ADVANTEST®
株式会社アドバンテスト

ネットワーク・アナライザ

プログラミング・マニュアル

MANUAL NUMBER OJA00 9506

適用機種
R3764A/B/C R3766A/B/C
R3765A/B/C R3767A/B/C

当社の製品が外国為替および外国貿易管理法の規定により、戦略物資あるいは役務等に該当する場合、輸出する際には日本国政府の許可が必要です。

初版1995年6月30日
Printed in Japan

この取扱説明書の使い方

この取扱説明書は、以下の構成で説明しています。

- 第 1部　内蔵BASIC について
- 第 2部　GPIBについて

（参考）　ネットワーク・アナライザの各部の名称、機能説明、キー操作など、詳細は別冊の取扱説明書を参照して下さい。

- R3764/66ネットワーク・アナライザ取扱説明書
  または
- R3765/67ネットワーク・アナライザ取扱説明書

第 1 部

第１部の目次

4. BASIC ステートメント

5. パラレル I/O ポート

6. エラー・メッセージ

1. はじめに

本器に内蔵されている BASIC言語は、汎用の BASICコマンド、GPIB制御用コマンドおよび専用ビルトイン関数を備え、小規模GPIBシステムを簡単に構築できます。

●コマンドとステートメントの構文について

本器で使われるコマンドとステートメントの構文は、本書の 3章と 4章で説明しますが、直観的に理解できるように図式表現と記述式表現を並記して解説しています。

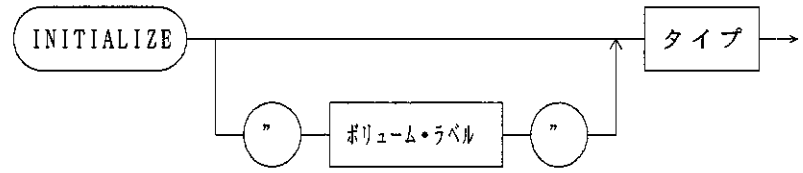
注意

各コマンドや各ステートメントの 構文 の読み方

(1) 図式表現

構文を各要素に分解し、直線で結んで表わしています。
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、それらのうちの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。

記述例）



(2) 記述式表現に用いている記号の意味

- 記号〔 〕で囲まれた部分 ： 省略できる。
- 記号 < > で囲まれた部分 ： 省略できない。
- 記号 { } で囲まれた部分 ： 0 回以上繰り返し用いることができる。
- 記号 | ： 「または」の意味。
  （例： A | B … AまたはB を用いる。）

記述例） INITIALIZE〔"ボリューム・ラベル"〕 <タイプ>

(3) 図式表現、記述式表現に用いている単語の意味

- 数値表現式 ： 数値定数、数値変数、数式のいずれか。
- 文字列表現式 ： 文字列定数、文字列変数、文字列関数、サブ・ストリングで構成される式。
- 装置アドレス ： GPIBに接続されている装置のアドレス。

●GPIBモードについて

本器は、2 種類のモード（ADDRESSABLE モードとSYSTEM CONTROLLER モード）で動作します。モードの切り換えは、CONTROL コマンド、または正面パネルで設定します。

CONTROL コマンドについては、[3. BASICコマンド] を参照して下さい。
正面パネルの設定については、各機器の取扱説明書を参照して下さい。

(1) ADDRESSABLE モード

通常のモードで、外部コントローラによりコントロールされます。
このモードで内蔵BASIC プログラムを実行すると、以下の 2種類の動作になります。

① BASIC コマンドの「CONTROL 7;4 」が設定されていない場合

内蔵BASIC と外部コントローラの間で、データの送受信ができます。
ただし、内蔵BASIC の ENTERとOUTPUT命令が優先されるため、外部コントローラからのGPIBコマンドによる設定はできなくなります。
外部コントローラからのGPIBコマンドによる設定をしたい場合は、内蔵BASIC プログラムを停止させるか、「CONTROL 7;4 」を設定して下さい。

② BASIC コマンドの「CONTROL 7;4 」が設定されている場合

①とは逆で、外部コントーラからのGPIBコマンドによる設定ができます。
つまり、内蔵BASIC 停止中の動作と同様です。ただし、内蔵BASIC と外部コントローラの間のデータ送受信はできません。

(2) SYSTEM CONTROLLER モード

内蔵BASIC プログラムにより、測定機能および外部接続した機器をコントロールすることができます。

注）このページでは本器に内蔵されているBASIC を、外部コントローラとの区別を明確にするために内部BASIC と呼びましたが、これ以降外部と区別する必要がない場合、BASIC と呼びます。

●フロッピー・ディスク

フロッピー・ディスクには、設定条件と測定データ、またBASIC プログラムとBASIC プログラム内からのデータ・ファイルなどを記録/ 再生できます。
フォーマット形式は、MS-DOSに準拠しているため、MS-DOS対応のパーソナル・コンピュータにて、プログラム作成やデータ解析などができます。
本器では、以下のフォーマットで初期化されたディスクが使用できます。

2DD(両面倍密度) : 720Kバイト (512バイト、 9セクタ)

2HD(両面高密度) : 1.2Mバイト (1024バイト、 8セクタ)
1.2Mバイト (512バイト、15セクタ)
1.4Mバイト (512バイト、18セクタ)

注意

本器では、2DD/2HD ディスクを自動認識するため、2DD ディスクで1.2Mまたは1.4Mバイト・フォーマットを行ったもの、2HD ディスクで720Kバイト・フォーマットを行ったものは使用できません。

(1) フロッピー・ディスクの外形と各名称

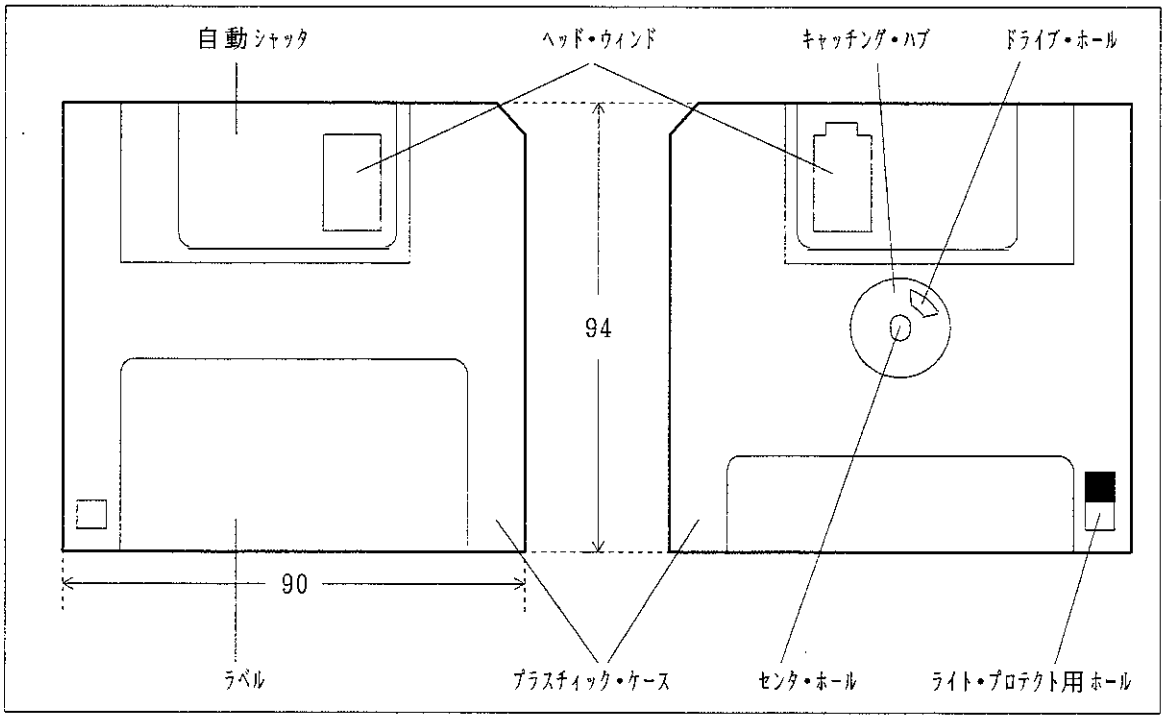


図 1 - 1 フロッピー・ディスクの外形と各名称

・ラベル　：フロッピー・ディスクを使用するときに、ユーザが貼り付けて使用する。

・ヘッド・ウィンド　：裏面にも同様な開口部があり、この部分にREAD/WRITEヘッドが位置する。
ヘッドは、このスロットの縦方向に沿って移動する。
フロッピー・ディスクをドライブ・スロットから抜き取った状態では、自動シャッタが閉じていて、ディスクの保護をする。

・キャッチング・ハブ（ドライブ・ホール、センタ・ホール）
：フロッピー・ディスクをドライブ・スロット内に挿入すると、ドライブ側にキャッチング用マグネットを使用したスピンドルがあり、ディスクを固定し、回転させる。

・ライト・プロテクト用ホール
：重要なデータを操作ミスなどによってデータを消去しないように書き込み禁止ができる。

(2) フロッピー・ディスクの装着および取扱方法

フロッピー・ディスクをフロッピー・ディスク・ドライブに装着する場合は、〔図2-2〕のようにフロッピー・ディスクのラベルが付いている側を上側にしてスロットに挿入します。このとき、指で押して完全に奥まで挿入してフロッピー・ディスクがドライブに固定されるのを確認して下さい。フロッピー・ディスクを取り出す場合は、イジェクト・ボタンを押すと、フロッピー・ディスクが自動的に出ます。

注意

フロッピー・ディスク・ドライブのランプが点滅中は、イジェクト・ボタンを押さないで下さい。誤動作やフロッピー・ディスク内のデータ破壊の原因になります。

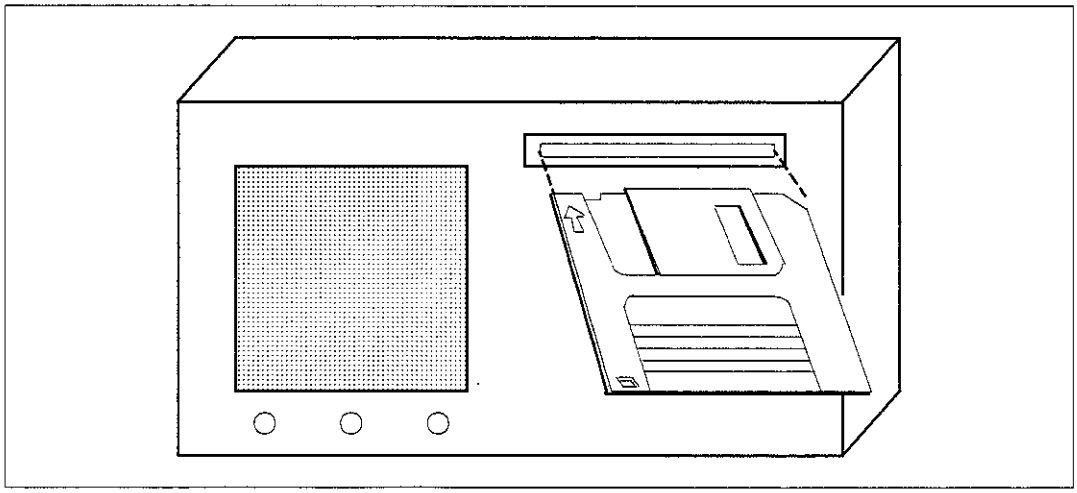


図 1 - 2 フロッピー・ディスクの装着方法（R3753 の場合）

フロッピー・ディスクの取扱いは、以下の事項に注意して下さい。

① 磁場および帯磁の原因となる強磁性材料に近づけないこと。

② 熱または直射日光にさらさないこと。

③ タバコの灰のような汚物に注意すること。

④ 磁気コーティングされた面に手を触れたり、手で清掃しないこと。

⑤ 重い物を載せないこと。

⑥ ダメージ（濡れ，折り目，歪みなど）を受けたり、異物で汚染されたフロッピー・ディスクは交換すること。（ドライブのヘッドを汚して、使用不可能にするだけでなく、他のフロッピー・ディスクをも汚染してしまいます。）

(3) 書き込み禁止（ライト・プロテクト）

フロッピー・ディスクは記録したデータを操作ミスなどで消去しないように、再度のデータの書き込みを禁止（Write Protect : ライト・プロテクト）できます。
書き込み禁止は、ライト・プロテクト用スライド（図2-3)で行います。通常、センタ・ホールに近い方にスライドがある場合は書き込みが可能です。また、ケースの端の方にスライドがある場合は書き込みが禁止されます。

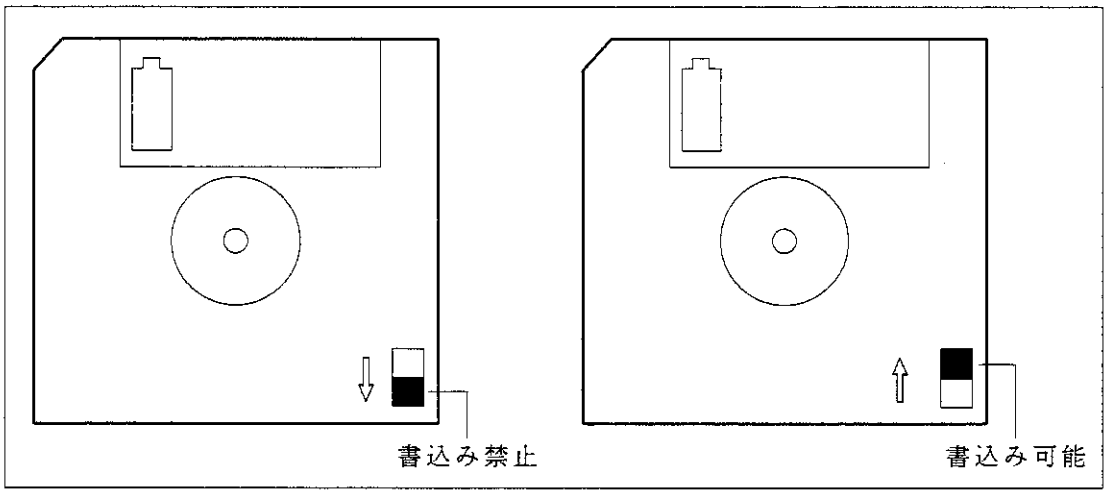


図 1 - 3 フロッピー・ディスクの書き込み禁止および解除

●ファイルの管理

本器のディスク・ファイルの管理方法は、MS-DOSで作成されたディスク・ファイルの管理方法と同じです。つまり、MS-DOSで作成されたフロッピー・ディスクはそのまま本器で使用でき、本器で作成したファイルもMS-DOSで参照できます。

(1) ファイル

通常、ひとまとまりの情報を「ファイル」と呼びます。パーソナル・コンピュータ(=パソコン)上で編集したBASIC プログラムやBASIC で作成したデータなどは、すべてファイルとして保存されます。

(2) ディレクトリ

ファイルは、ディレクトリごとに管理することができます。
本器では、ディレクトリを作成する機能はありませんが、サブ・ディレクトリにあるファイルを参照することはできます。

(3) ドライブ

ファイルの保存場所は、フロッピー・ディスクやメモリ・ディスクなどのディスクです。ディスクを読み書きする装置を「ドライブ」と呼び、ドライブごとにディスクを管理します。本器では、以下の4 つのドライブが割り付けられています。

A ：フロッピー・ディスク

MS-DOSで作成されたフロッピー・ディスクと同じです。

B ：バック・アップされないメモリ・ディスク

電源投入時に自動的にフォーマットします。電源をオフすると、ディスクの内容は失われます。
BASIC では最高約128Kバイトまで使用できますが、レジスタを使用すると使用できる容量は減ります。

C ：バック・アップされるメモリ・ディスク

電源をオフしてもディスクの内容は保持されます。
BASIC では最高約900Kバイトまで使用できますが、レジスタを使用すると使用できる容量は減ります。

D ：読み出し専用のメモリ・ディスク

本器のシステム・プログラムを保持しているディスクです。
BASIC では使用できません。

カレント・ドライブの選択方法については、各機種の『取扱説明書』を参照して下さい。

(4) ファイルの指定方法

ドライブ・ディレクトリを含めてファイルを指定する方法を、以下に示します。

"ドライブ名:/ディレクトリ名/ファイル名"

通常MS-DOSでは、ディレクトリの区切りに "¥" (英語モードのときは"\")を使用しますが、本器では"/" を使用します。[4. BASICステートメント] で説明しますが、本器では文字列中の"\" は特別な使い方をするため、"\" でなく"/" を使用します。

(5) フロッピー・ディスクの初期化

新品のフロッピー・ディスクを使用する際には、はじめに初期化 (フォーマット) という作業をする必要があります。
初期化には、以下の3 通りの方法があります。

① パソコン上でMS-DOSのFORMATコマンドを行い、そのフロッピー・ディスクを本器で使用する。

② 本器のパネル操作で行う (パネルの章を参照)。

③ 本器のBASIC で、INITIALIZEコマンドを実行する。

一般にフロッピー・ディスクのフォーマットには、以下の4 タイプがあります。

① 1.44M バイト形式(2HD、512 バイト、18セクタ)
② 1.2 M バイト形式(2HD、1024バイト、8 セクタ)
③ 1.2 M バイト形式(2HD、512 バイト、15セクタ)
④ 720 K バイト形式(2DD、512 バイト、9 セクタ)
⑤ 640 K バイト形式(2DD、512 バイト、8 セクタ)

本器では、このうち⑤を除く4 タイプの形式のフロッピー・ディスクを使用することができます。

(注) PC9801シリーズでは、FORMATコマンドで2DD をフォーマットするときのデフォルトは、⑤のフォーマットになります。
本器で使用するフロッピーは、④のフォーマットになるようにフォーマットして下さい。

●キーボード

OADG (PCオープン・アーキテクチャ推進協議会) の規定する101 型キーボードおよび106 型キーボードが接続できます。
R3765/67シリーズの場合、フロント・パネルのPROGRAM キーを押すと、BASIC に対してキーボードの入力が可能になります。

注意

キーボードの接続は、必ず電源投入前に行って下さい。
電源投入後に接続した場合には、正常な動作は保障できません。

MEMO

2. 基本操作

プログラムの作成，実行，終了の方法を以下に示します。

2.1 プログラムの作成

① パーソナル・コンピュータで行う場合

パーソナル・コンピュータ（または、MS-DOSフォーマットのフロッピー・ディスクにファイル・アクセスが可能な機器）で入力，編集を行い、ASCII 形式でフロッピー・ディスクにセーブします。

② キーボードから行う場合

プログラム行を行番号付きで入力し、フロッピー・ディスクに保存します。

注意

ファイルの拡張子についての制約はありませんが、BASIC プログラム・ファイル以外と区別がつきやすいように、拡張子は.BASとして下さい。

BASIC で扱える文字コードは、 7ビットASCII コードです。
ただし、以下の文字はBASIC で使用していないため、プログラム・ステートメントの中で使用した場合、プログラムのロードはこの行で中断します。
（ダブル・クォーテーションでくくられた場合を除きます。）

```
‘
,
/
{
}
|
```

2.2 プログラムの実行

● R3764/66シリーズ

① 実行したいプログラムが、セーブされているフロッピー・ディスクを本器のフロッピー・ディスク・ドライブに装着します。

② LOADキー（パネル・キー）を押して、フロッピー・ディスク内のファイルを表示させます。

③ ↑, ↓キー（パネル・キー）を使い、ロードしたいファイル名にカーソルを移動させます。

④ ENT キー（パネル・キー）を押すと、プログラムをロードします。

⑤ RUN キー（パネル・キー）を押すと、プログラムを実行します。

● R3765/67シリーズ

① 実行したいプログラムが、セーブされているフロッピー・ディスクを本器のフロッピー・ディスク・ドライブに装着します。

② RUN キー（パネル・キー）を押して、コントローラ・メニューを表示させます。

③ LOAD MENU キー（ソフト・キー）を押して、フロッピー・ディスク内のファイルを表示させます。

④ CURSOR↑, CORSOR↓キー（ソフト・キー）を使い、ロードしたいファイル名にカーソルを移動させます。

⑤ LOADキー（ソフト・キー）を押すと、プログラムをロードします。

⑥ RUN キー（ソフト・キー）を押すと、プログラムを実行します。

2.3 プログラムの終了

① R3764/66シリーズ

STOPキー（パネル・キー）を押すと、プログラムが終了します。

② R3765/67シリーズ

1) RUN キー（パネル・キー）を押して、コントローラ・メニューを表示させます。

2) STOPキー（ソフト・キー）を押すと、プログラムが終了します。

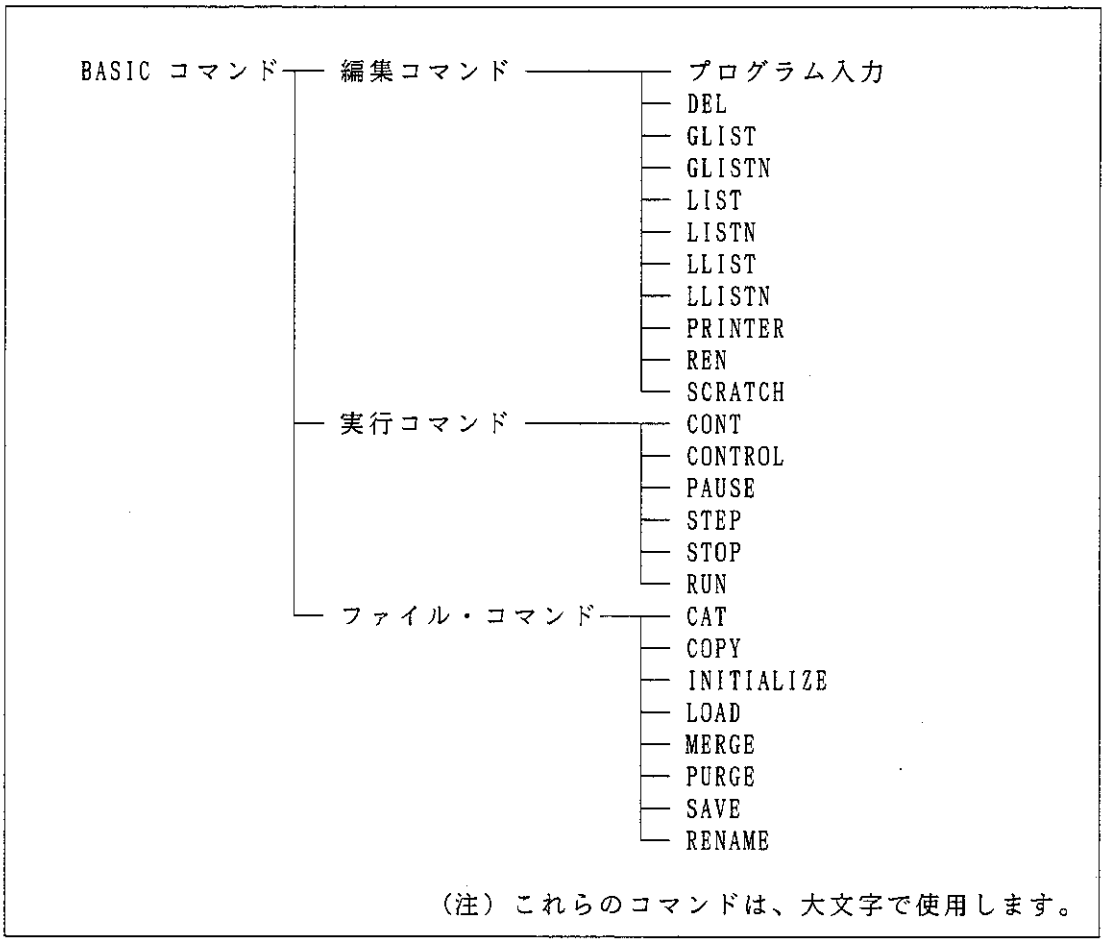
MEMO

3. BASIC コマンド

BASIC では、コマンドとステートメントを使用します。
コマンドとは、基本的に直接実行させる（プログラム中ではなく）もので、ステートメントとは、基本的にプログラム中で実行するものを言います。
ここではまず、コマンドの説明をします。

3.1 各種コマンド

BASIC には、プログラムの編集、実行およびファイル操作を行うためのコマンドがあります。BASIC コマンドの構成を以下に示します。



これらのコマンドは、ステートメントとしてプログラム中でも実行可能なものがあります。

3.1.1 コマンド機能一覧

	コマンド	機能	ｽﾃｰﾄﾒﾝﾄで実行可
編集コマンド	ﾌﾟﾛｸﾞﾗﾑ 入力	ステートメントをプログラムとして格納	×
	DEL	指定行番号を削除	×
	GLIST	プログラム・リストをGPIBに出力	○
	GLISTN	プログラム・リストをGPIBに出力	○
	LIST	プログラム・リストをディスプレイ上に表示	○
	LISTN	プログラム・リストをディスプレイ上に表示	○
	LLIST	プログラム・リストをシリアル・ポートに出力	○
	LLISTN	プログラム・リストをシリアル・ポートに出力	○
	PRINTER	プリンタのGPIBアドレスを設定	○
	REN	行番号の変更	○
	SCRATCH	すでに入力されているプログラムを消去	×
実行コマンド	CONT	プログラム実行を再開	×
	CONTROL	BASIC のコントロール変数を設定（環境設定）	○
	PAUSE	プログラム実行を一時停止（CONT可）	○
	STEP	プログラムを一行実行	×
	STOP	プログラム実行を停止　　（CONT不可）	○
	RUN	プログラムを実行	○
ファイル・コマンド	CAT	ｶﾚﾝﾄ･ﾄﾞﾗｲﾌﾞ 内のファイル名をディスプレイ上に表示	○
	COPY	ファイルの複写	○
	INITIALIZE	ﾌﾛｯﾋﾟｰ･ﾃﾞｨｽｸを初期化	○
	LOAD	プログラムをロード（呼び出す）	○
	MERGE	すでに入力済のプログラムに追加する形でロード（呼び出す）	○
	PURGE	ファイルの削除	○
	SAVE	プログラムのセーブ（格納）	○
	RENAME	ファイル名の変更	○

3.1.2　コマンド文法一覧

	コマンド	文法
編集コマンド	プログラム入力	行番号 ｽﾃｰﾄﾒﾝﾄ
	DEL	DEL 開始行〔, 終了行〕
	GLIST	GLIST〔開始行〕〔, 〔終了行〕〕
	GLISTN	GLISTN〔開始行〕〔, 〔行数〕〕
	LIST	LIST 〔開始行〕〔, 〔終了行〕〕
	LISTN	LISTN 〔開始行〕〔, 〔行数〕〕
	LLIST	LLIST〔開始行〕〔, 〔終了行〕〕
	LLISTN	LLISTN〔開始行〕〔, 〔行数〕〕
	PRINTER	PRINTER 装置ｱﾄﾞﾚｽ
	REN	REN 〔〔旧行番号〕〔, < 新行番号> 〔, < 増分値> 〕〕〕
	SCRATCH	SCRATCH 〔1 | 2 〕
実行コマンド	CONT	CONT〔行番号〕
	CONTROL	CONTROL <レジスタ番号> ; <値>
	PAUSE	PAUSE
	STEP	STEP〔行番号〕
	STOP	STOP
	RUN	RUN 〔行番号 | "ファイル名" 〕
ファイル・コマンド	CAT	CAT 〔"DATE"〕
	COPY	COPY "旧ﾌｧｲﾙ名" , "新ﾌｧｲﾙ名"
	INITIALIZE	INITIALIZE〔" ﾎﾞﾘｭｰﾑ･ﾗﾍﾞﾙ"〕 <ﾀｲﾌﾟ>
	LOAD	LOAD "ﾌｧｲﾙ名"
	MERGE	MERGE "ﾌｧｲﾙ名"
	PURGE	PURGE "ﾌｧｲﾙ名"
	SAVE	SAVE "ﾌｧｲﾙ名"
	RENAME	RENAME "旧ﾌｧｲﾙ名" , "新ﾌｧｲﾙ名"

3.1.3 コマンドの共通注意事項

内部BASIC コマンドには、以下に示す共通の注意事項があります。

(1) パラメータ

コマンドのパラメータには、文字列表現式または数値表現式が指定できます。つまり、BASIC で使用する変数が使えます。ここで、実数の場合は小数点以下が切り捨てられます。
しかし各コマンドの解説では、見やすさのため整数、文字列などの表現を使っています。

(2) 式の境界

原則としてBASIC コマンドは、式を複数個続けて指定する場合、その式の境界が構文上解釈できれば、カンマ(,) はスペースにしても構いません。

(3) LIST, LISTN, LLIST, LLISTN, GLIST, GLISTN における行番号

行番号の設定範囲は 1～65535 です。
0 またはプログラムの先頭行よりも小さい値を指定した場合、プログラムの先頭行を指定したと解釈されます。
65535 またはプログラムの最終行よりも大きい値を指定した場合、プログラムの最終行を指定したと解釈されます。
指定した番号がないときは、指定した行番号より大きい最も近い行番号が選択されます。行番号の代わりにラベルを指定することができます。

3.2 コマンド文法と活用

1. プログラム入力

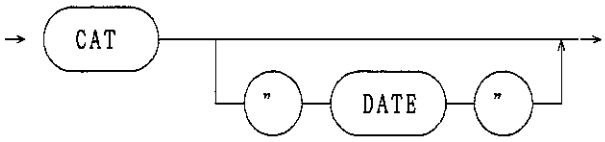
3 章と4 章に記載するコマンドとステートメントは、行番号を付けるとプログラムとして入力できます。

入力済のプログラムに同一の行番号が存在する場合は、置き換えとなります。また、存在しない場合は、追加または挿入となります。

2. CAT

概 要 カレント・ドライブに記憶されたファイルを表示します。

構 文 (1)-1



(1)-2
CAT 〔"DATE"〕

解 説 ・カレント・ドライブに記憶されたファイルとディレクトリをリスト表示します。

CAT の場合 : 左側から登録番号、ファイル名、使用バイト数、ファイルの属性が表示される。

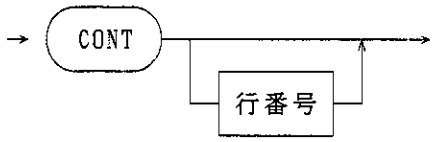
CAT "DATE"の場合 : 左側から登録番号、ファイル名、作成日時が表示される。

※ ファイルの取り扱いについては、「1. はじめに ●ファイルの管理」を参照して下さい。

3. CONT

概 要 BASIC プログラムの実行を再開させます。

構 文 (1)-1



(1)-2
CONT〔行番号〕

解 説
・PAUSE で停止したBASIC プログラムを、停止している行の次から再開できます。

・BASIC プログラムを指定した行から再開させます。
変数の初期化はしません。

・プログラム中でステートメントとして使用することはできません。

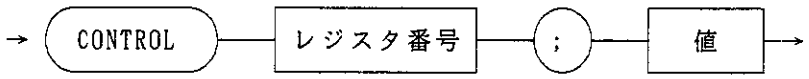
例
```
CONT

CONT 200
```

4. CONTROL

概 要 BASICのコントロールに関係する細かな部分の値を設定します（環境設定）

構 文 (1)-1



(1)-2
CONTROL　レジスタ番号　;　値

解 説

・レジスタ番号で設定するコントロール対象を指定し、セミコロン(;) の後の値が実際の設定値です。

・レジスタ番号には、 1～9 の値を設定します。各レジスタの内容を以下に示します。（ただし、レジスタ6 は内部の都合上未登録です。）

〈レジスタ1 〉 …初期値79
シリアルI/O ポートの設定です。以下の値の和で指定します。
下線のある値は、電源投入時に設定されている値です。

①ボーレート	: 0 ; 1200ボー	③パリティ	: 0 ; なし
	1 ; 2400ボー		16 ; 奇数
	2 ; 4800ボー		48 ; 偶数
	3 ; 9600ボー		
②キャラクタ長	: 0 ; 5 ビット	④ｽﾄｯﾌﾟ･ﾋﾞｯﾄ数	: 0 ; なし
	4 ; 6 ビット		64 ; 1 ビット
	8 ; 7 ビット		128 ; 1 1/2 ビット
	12 ; 8 ビット		192 ; 2 ビット

例）ボーレート9600ボー、キャラクタ長8 ビット、パリティ偶数、ストップ・ビット数 2ビットの場合

```
CONTROL 1;3+12+48+192
または
CONTROL 1;255
```

〈レジスタ2 〉 …初期値0
LLIST またはGLIST で左端からの印字位置をスペースの数で指定します。

例）リスト出力を右に 5文字分寄せる場合

CONTROL 2;5 を実行してからLLIST またはGLIST を実行すると、行番号の前にスペースが 5つ入り、その後にリストが表示されます。

<レジスタ3 > ····初期値0
BASIC プログラムをフルネーム表示にするか、ショートネーム表示にするかを指定します。

0 : フルネーム表示にする。
1 : ショートネーム表示にする。

フルネームとショートの対応表は [表4-2]を参照して下さい。

<レジスタ5 > ····初期値0
メンテナンス用コマンドのPOKEを有効にするか、無効にするか指定します。

0 : 無効
1 : 有効

<レジスタ7 > ····初期値0
GPIB関係の設定です。以下の値は各々設定します。

0 : GPIBモードをADDRESSABLE モードに設定する。
1 : GPIBモードをSYSTEM CONTROLLLERモードに設定する。
2 : REQUEST CONTROL （コントローラ権の要求）を送信する。
4 : BASIC 実行中に、外部コントローラからのGPIBコマンド設定を有効にする。

<レジスタ8 > ····初期値0
DMA 転送モードのON/OFFを設定します。

0 : OFF
1 : ON

<レジスタ9 > ····初期値1
PRINT の出力先の指定です。以下に示す値の和で設定します。

1 : デフォルト出力（各機種の正面パネル表示器）
2 : メンテナンス用ポート（端末）への出力
4 : 外部モニタおよびR3765/67液晶ディスプレイへの出力
8 : R3764/66用の蛍光表示管への出力

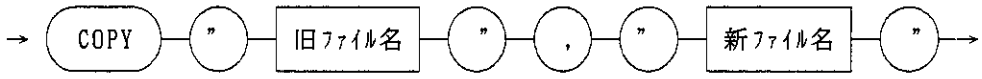
例1) デフォルト出力およびメンテナンス用ポートに出力したい場合
```
CONTROL 9;3
```

例2) デフォルト出力、メンテナンス用ポートおよび外部モニタへ出力したい場合
```
CONTROL 9;7
```

5.　COPY

概　要　ファイルを複写します。

構　文　(1)-1



(1)-2
COPY "旧ファイル名", "新ファイル名"

解　説

・旧ファイル名の内容を新ファイル名に複写します。

・新ファイル名が既に存在する場合、旧ファイル名の内容が上書きされます。

・新ファイル名と旧ファイル名が同じ場合はエラーとなります。

・2 つのファイル名とも文字列表現式を使って指定できます。

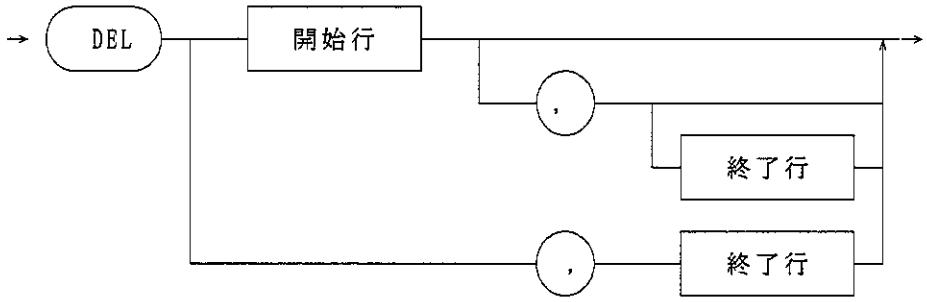
・ドライブの指定をすると、ドライブ間でのコピーができます。ドライブの指定のないときは、カレント・ドライブになります。

※　ファイルの取り扱いについては、「1.　はじめに　●ファイルの管理」を参照して下さい。

6.　DEL

概　要　プログラム中の行を削除します。

構　文　(1)-1



(1)-2
DEL <開始行〔, 〔終了行〕>|<,終了行>

注）カンマ(,)をスペースにしても構いません。
　　行番号の設定範囲は 1～65535 です。

解　説
・開始行から終了行までプログラムを削除します。
・行番号の指定がない場合は実行しません。
・プログラム中でステートメントとして使用することはできません。

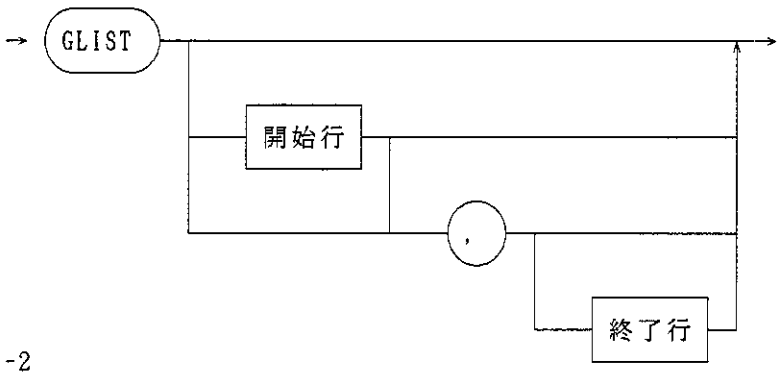
例

DEL 10	行番号10のみ削除
DEL 10 ,	行番号10～最終行まで削除
DEL 10, 100	行番号10～100 まで削除
DEL , 100	先頭行～行番号100 まで削除

7. GLIST

概 要 GPIB経由でプリンタ等にプログラム・リストを出力します。

構 文 (1)-1



(1)-2

GLIST 〔開始行〔, 〔終了行〕〕〕 | 〔, 〔終了行〕〕

注) カンマ(,) をスペースにしても構いません。
行番号の設定範囲は 1～65535 です。
行番号の代わりにラベルも使用できます。

解 説

・GPIBに接続されたプリンタ等に、BASICプログラム・リストを出力します。

・プリンタのGPIBアドレスは、PRINTER 文で、または本器のパネル操作で設定します。

・本器のパネル操作で、SYSTEM CONTROLLER にします。

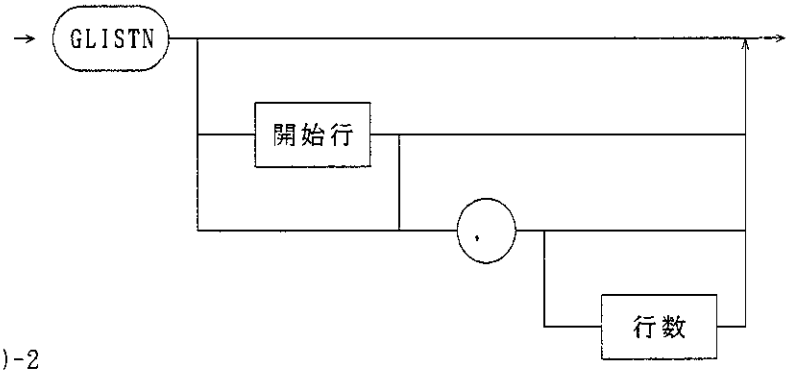
例

GLIST	全行を出力
GLIST 100	100 行目のみ出力
GLIST 100,	100 行目～最終行まで出力
GLIST 100, 200	100 行目～200 行目まで出力
GLIST ,	全行を出力 (GLISTと同じ)
GLIST ,200	先頭行～200 行目まで出力

8. GLISTN

概 要 GPIB経由でプリンタ等にプログラム・リストを出力します。

構 文 (1)-1



(1)-2
GLISTN〔開始行〔,〔行数〕〕〕|〔,〔行数〕〕

注）カンマ(,) はスペースにしても構いません。
行番号の設定範囲は 1～65535 です。
行番号の代わりにラベルも使用できます。

解 説

・GPIBに接続されたプリンタ等に、BASIC プログラム・リストを出力します。

・プリンタのGPIBアドレスは、PRINTER 文で、または本器のパネル操作で設定します。

・本器のパネル操作で、SYSTEM CONTROLLER にします。

・開始行で指定した行番号から行数で指定した行数分のプログラム・リストを出力します。

・行数が負の数の場合、行番号の小さい方に向かって指定行数がとられます。

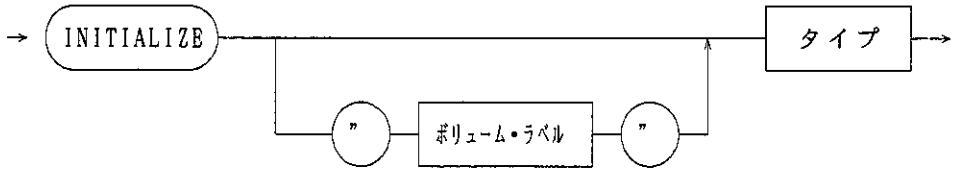
例

GLISTN	全行を出力
GLISTN 100	100 行目のみ出力
GLISTN 100,	100 行目～最終行まで出力
GLISTN 100, 20	100 行目～20行出力
GLISTN ,	全行を出力 (GLISTN と同じ)
GLISTN ,20	先頭行～20行出力

9.　INITIALIZE（INIT）

概　要　フロッピー・ディスクを初期化します。

構　文　(1)-1



(1)-2
INITIALIZE〔"ﾎﾞﾘｭｰﾑ･ﾗﾍﾞﾙ"〕タイプ

解　説

・新品のフロッピー・ディスクや、転用したいフロッピー・ディスクを、タイプで指定したフォーマットで初期化します。

・初期化時には、ボリューム・ラベルを指定できます。
省略すると、ボリューム・ラベルなしになります。

・タイプは、以下のように指定して下さい。

タイプ指定：　0；　720KB（512バイト、9セクタ）2DD
1；　1.2MB（1024バイト、8セクタ）2HD
2；　1.4MB（512バイト、18セクタ）2HD
3；　1.2MB（512バイト、15セクタ）2HD

注意

本器は、2DD/2HD を自動認識します。よって、上記のタイプ指定が、挿入されているフロッピー・ディスクに適合しない場合は、以下のデフォルト設定で初期化します。

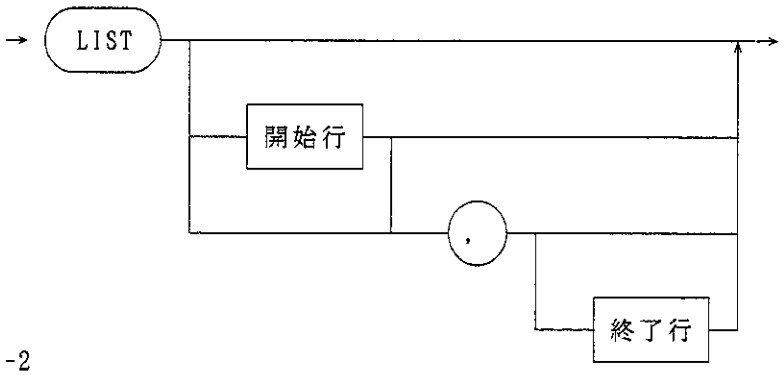
デフォルト設定：2DDの場合；720KB（タイプ0）
2HDの場合；1.2MB（タイプ1）

※　ファイルの取り扱いについては、「1.　はじめに　●ファイルの管理」を参照して下さい。

10. LIST

概 要 ディスプレイ上にプログラム・リストを表示します。

構 文 (1)-1



(1)-2

LIST〔開始行〔,〔終了行〕〕〕|〔,〔終了行〕〕

注）カンマ(,)をスペースにしても構いません。
行番号の設定範囲は 1～65535 です。
行番号の代わりにラベルも使用できます。

解 説

・ディスプレイ上にパラメータで指定した範囲の BASICプログラム・リストを表示します。

・STOPキーで表示を中止できます。しかし、プログラムの実行と異なるため、中断した行位置から再開できません。

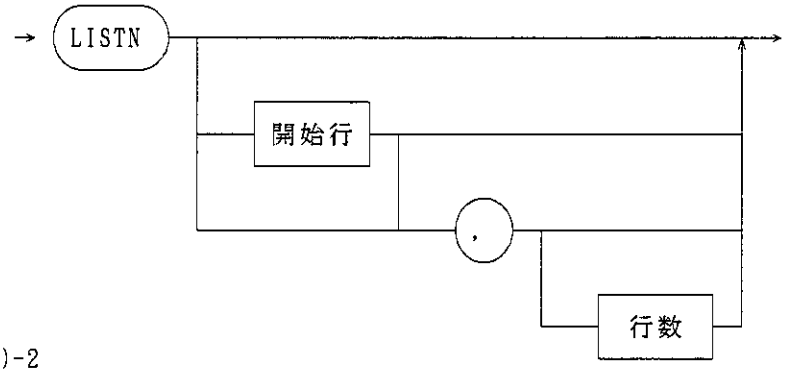
例

LIST	全行を出力
LIST 100	100 行目のみ出力
LIST 100,	100 行目～最終行まで出力
LIST 100, 200	100 行目～200 行目まで出力
LIST ,	全行を出力 (LIST と同じ)
LIST ,200	先頭行～200 行目まで出力

11. LISTN

概 要 ディスプレイ上にプログラム・リストを表示します。

構 文 (1)-1



(1)-2

LISTN 〔開始行〔, 〔行数〕〕 | 〔, 〔行数〕〕

注) カンマ(,) はスペースにしても構いません。
行番号の設定範囲は 1～65535 です。
行番号の代わりにラベルも使用できます。

解 説 ・ディスプレイにパラメータで指定した範囲のBASIC プログラム・リストを表示します。

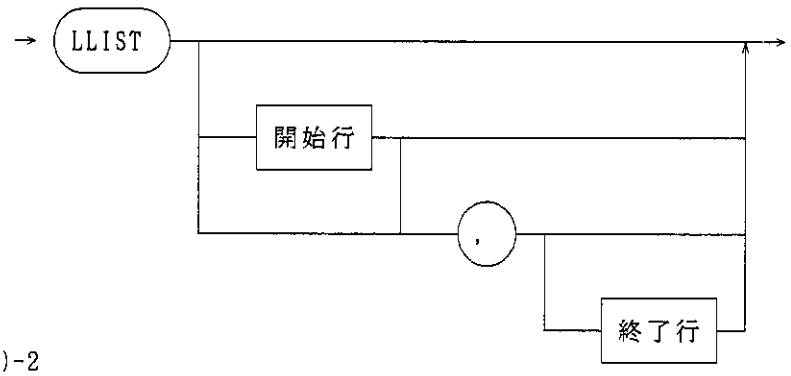
例

LISTN	全行を出力
LISTN 100	100 行目のみ出力
LISTN 100,	100 行目～最終行まで出力
LISTN 100, 20	100 行目～20行出力
LISTN ,	全行を出力 (LISTNと同じ)
LISTN ,20	先頭行～20行出力

12. LLIST

概 要 シリアル・ポート経由でプリンタ等にプログラム・リストを出力します。

構 文 (1)-1



(1)-2

LLIST 〔開始行〔, 〔終了行〕〕〕 | 〔, 〔終了行〕〕

注) カンマ(,) をスペースにしても構いません。
行番号の設定範囲は 1～65535 です。
行番号の代わりにラベルも使用できます。

解 説
- シリアル・ポートに接続されたプリンタ等に、BASIC プログラム・リストを出力します。

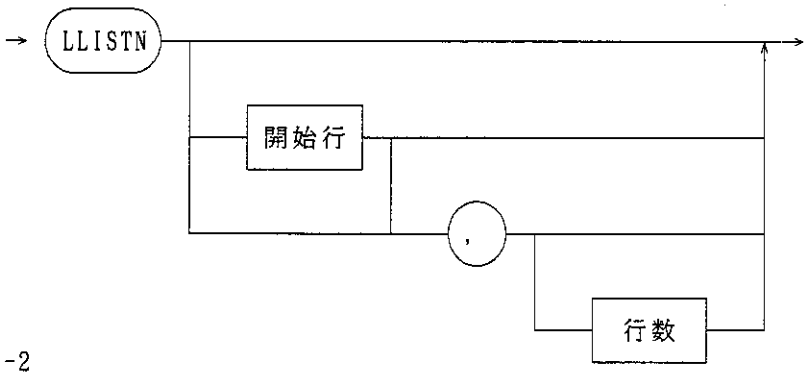
例

LLIST	全行を出力
LLIST 100	100 行目のみ出力
LLIST 100,	100 行目～最終行まで出力
LLIST 100, 200	100 行目～200 行目まで出力
LLIST ,	全行を出力 (LLISTと同じ)
LLIST ,200	先頭行～200 行目まで出力

13. LLISTN

概 要 シリアル・ポート経由でプリンタ等にプログラム・リストを出力します。

構 文 (1)-1



(1)-2

LLISTN〔開始行〔,〔行数〕〕〕|〔,〔行数〕〕

注）行番号の設定範囲は 1～65535 です。
行番号の代わりにラベルも使用できます。

解 説

・シリアル・ポートに接続されたプリンタ等に、BASIC プログラム・リストを出力します。

・開始行で指定した行番号から行数で指定した行数分のプログラム・リストを出力します。

・行数が負の数の場合、行番号の小さい方に向かって指定行数がとられます。

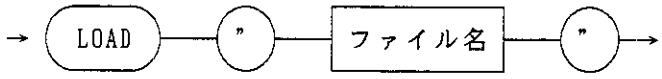
例

LLISTN	: 全行を出力
LLISTN 100	: 100 行目のみ出力
LLISTN 100,	: 100 行目～最終行まで出力
LLISTN 100, 20	: 100 行目～20行出力
LLISTN ,	: 全行を出力 (LLISTN と同じ)
LLISTN ,20	: 先頭行～20行出力

14. LOAD

概 要 BASIC プログラム・ファイルを呼び出します。

構 文 (1)-1



(1)-2
LOAD "ファイル名"

解 説

・ファイル名に指定されたファイルを呼び出します。BASIC 以外のファイルは呼び出さないで下さい。

・ドライブの指定のないときは、カレント・ドライブから呼び出します。

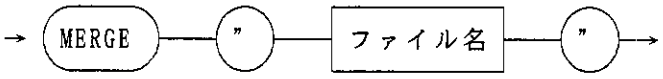
・行番号のないプログラムをロードした場合には、自動的に行番号が付きます。

※ ファイルの取り扱いについては、「1. はじめに ●ファイルの管理」を参照して下さい。

15. MERGE

概 要 BASIC プログラム・ファイルを呼び出し、メモリにあるプログラムに上書きします。

構 文 (1)-1



(1)-2
MERGE "ファイル名"

解 説

・LOADと異なり、ロードする前にBASIC メモリの初期化は行いません。

・すでにBASIC メモリに存在するプログラムは行番号が一致しない限り削除されません。

・行番号のないプログラムは、ロードできません。

・SCRATCH と MERGEの組み合わせがLOADの機能と同様になります。

※ ファイルの取り扱いについては、「1. はじめに ●ファイルの管理」を参照して下さい。

16. PAUSE

概 要 プログラムの実行を一時停止させます。

構 文 (1)-1

(1)-2
PAUSE

解 説
- BASIC プログラムの実行を一時的に停止、またはBASIC プログラム自身で一時的に停止させます。
- 停止した次の行から、CONTによりプログラムを継続できます。

例

```
10  FOR I=1 TO 9
20    GOTO 60
30    GOTO *PRT
40  NEXT I
50  PAUSE
60  !
70  X = I * I
80  GOTO 30
90  *PRT
100 PRINT I;"*";I;"=";X
110 GOTO 40
```

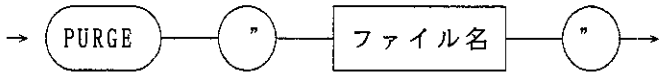
17.　PRINTER

4.3節の[44. PRINTER] を参照して下さい。

18.　PURGE

概　要　ファイルを消去します。

構　文　(1)-1



(1)-2
PURGE　"ファイル名"

解　説
・ファイルを消去します。消去されたファイルは、もとに戻りませんので注意して下さい。

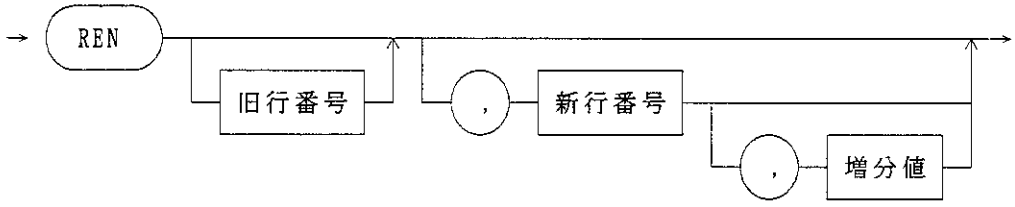
・ドライブの指定のないときは、カレント・ドライブになります。

※　ファイルの取り扱いについては、「1.　はじめに　●ファイルの管理」を参照して下さい。

19.　REN

概　要　プログラムの各行の行番号を付け直します。

構　文　(1)-1



(1)-2
REN 〔〔旧行番号〕〔, 新行番号〔, 増分値〕〕〕

注）カンマ(,)をスペースにしても構いません。
　　旧行番号、新行番号、増分値の設定範囲は 1～65535 です。
　　新行番号と増分値は、省略すると10に設定されます。

解　説
・旧行番号は、番号の付け替えが始められる現在のプログラム中の行番号です。

・新行番号は、新しく付ける先頭の行番号です。

・増分値は、新しく付ける行番号の増分量を示します。

・REN は、GOTO, GOSUB などで使っている行番号にも、新しい行番号に対応して変更します。

・REN は、65535 を超える行番号を使えません。また、プログラム行の順序を変えるような指定をしてはいけません。

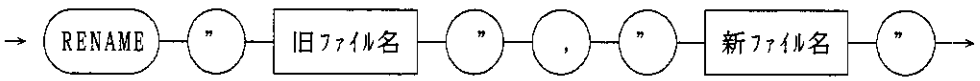
例

REN	: プログラムの先頭行を10にして、10ステップで最後まで行番号を付け直す
REN 30,50,3	: 行番号30を50にして3 ステップで最後まで行番号を付け直す

20. RENAME

概 要 ドライブに登録されているファイル名を変更します。

構 文 (1)-1



(1)-2
RENAME "旧ファイル名", "新ファイル名"

解 説

・ファイル名だけの変更で、内容は変更しません。

・新ファイル名と同じ名前のファイルが、既に存在する場合や、新ファイル名と旧ファイル名が同じ場合は、動作しません。

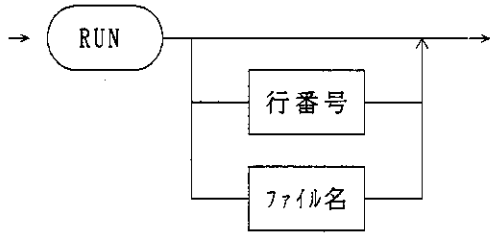
・異なるドライブ間でのRENAMEはできません。ドライブの指定のないときは、カレント・ドライブになります。

※ ファイルの取り扱いについては、「1. はじめに ●ファイルの管理」を参照して下さい。

21. RUN

概 要 BASIC プログラムを実行させます。

構 文 (1)-1



(1)-2

RUN 〔行番号 | ファイル名〕

解 説

・BASIC プログラムを指定した行から実行させます。

・行番号を指定しないときは、先頭行から実行します。

・ファイル名を指定したときには、指定ファイルをロードしたあと実行します。開始行の指定はできません。

・RUN を実行すると、プログラムを実行する前にすべての変数をクリアし、配列宣言なども強制的に無設定状態になります。

例

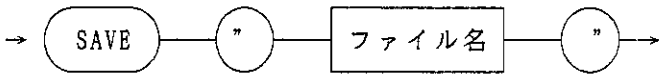
```
RUN

RUN 200
```

22. SAVE

概 要 BASIC プログラム・ファイルを保存します。

構 文 (1)-1



(1)-2
SAVE "ファイル名"

解 説

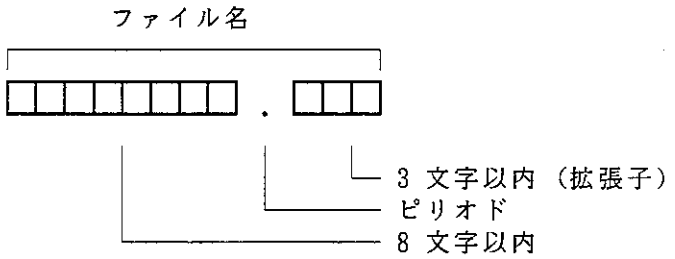
・編集したプログラム（行番号の付いている文の先頭から最後まで）をファイル名で指定した名前でファイルに登録します。

・既存のファイル名を指定すると、同一ファイルの更新とみなされ、その内容が更新されます。

・ドライブ指定がないときは、カレント・ドライブになります。

注意

ファイル名は、数字、英字および記号（ダブル・クォーテーション(") は除く）を使い、以下のように指定します。



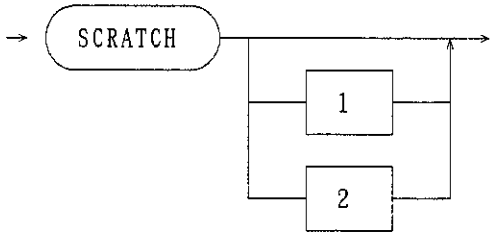
拡張子はできるだけ.BASとして下さい。

※ ファイルの取り扱いについては、「1. はじめに ●ファイルの管理」を参照して下さい。

23. SCRATCH

概 要 メモリに蓄えられたBASICプログラムを消去します。

構 文 (1)-1



(1)-2

SCRATCH 〔1 | 2 〕

例

SCRATCH : BASIC バッファ内のプログラムをすべて消去する。(データ・プロシージャ共に)

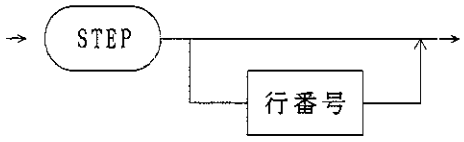
SCRATCH 1 : BASIC バッファ内のプログラムのデータのみ初期化する。

SCRATCH 2 : BASIC バッファ内のプログラムのプロシージャのみ初期化する。

24.　STEP

概　要　BASIC プログラムを一行のみ実行させます。

構　文　(1)-1



(1)-2
STEP〔行番号〕

解　説
・BASIC プログラムを指定した一行のみ実行しますが、FOR 文の中では実行しません。

・行番号の指定がない場合は、直前に終了した次の行を実行します。

例
```
STEP

STEP 100
```

25. STOP

概 要 BASIC プログラムの実行を停止させます。

構 文 (1)-1

(1)-2
STOP

解 説 ・BASIC プログラムの実行を停止、またはBASIC プログラム自身で停止させます。

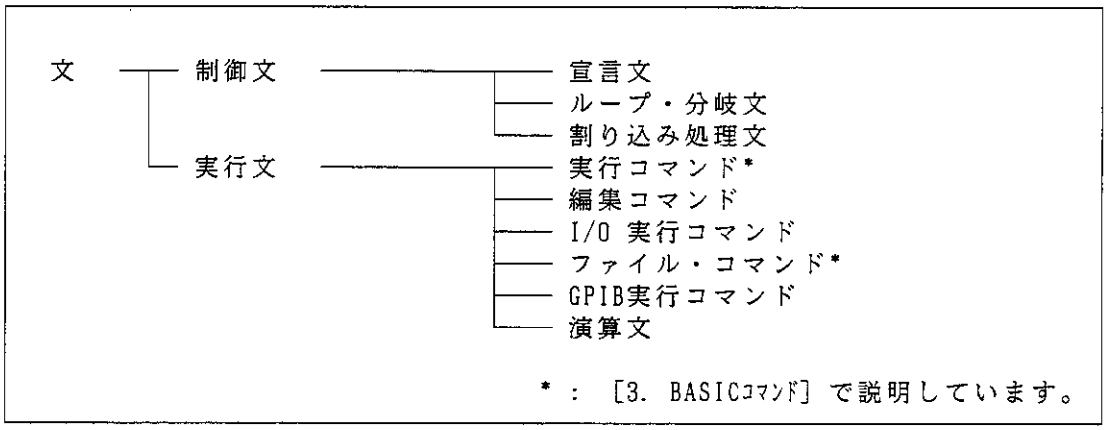
4. BASIC ステートメント

4.1　プログラミングのきまり

4.1.1　プログラム構造

(1) 文

BASIC プログラムは各種の文の集まりです。
文は大きく分けて、制御文と実行文によって構成されています。
各文は、キー・ワードと式から構成されます。この構成を取り決めたものが文法の構文規則です。



(2) キー・ワード

BASIC で、あらかじめ意味と用途を定めている言葉を「キー・ワード」と呼びます。キー・ワードと同じ名前を、その他の目的に使用できません。
キー・ワードの名前（フルネーム）で、使用頻度が高く、かつ長い名前のものにはショート・ネームがあります。
表示をフルネームからショート・ネームに変更するには、CONTROL コマンドでコントロール・レジスタ3 を 1にします。フルネームに戻すには、コントロール・レジスタ3 を 0にします。

キー・ワード一覧表は、[表4-1]を参照して下さい。
フルネームとショート・ネームの対応表は、[表4-2]を参照して下さい。

表 4 - 1　キー・ワード一覧

AND	APPEND	AS	ASCII	BAND	BASIC(*)
BINARY	BNOT	BOR	BREAK	BUZZER	BXOR
CASE	CAT	CHKDSK	CIRCLE(*)	CLEAR	CLOSE
CLS	CMD	COLOR(*)	CONSOLE	CONT	CONTINUE
CONTROL	COPY	DELAY	COUNT	CSR	CURSOR
DATA	DEL	ELSE	DELIMITER	DIM	DISABLE
DSTAT	DUMP	ERROR	ENABLE	END	ENT
ENTER	GLISTN	GOSUB	EVENT	FOR	FORMAT
GLIST	INITIALIZE	INP	GOTO	GPRINT	IF
INIT	ISRQ	KEY	INPUT	INTEGER	INTERFACE
INTR	LISTEN	LISTN	LABEL(*)	LINE(*)	LINETYPE(*)
LIST	LPRINT	LOAD	LLIST	LLISTN	LOCAL
LOCKOUT	NOT	OFF	MERGE	MOVE(*)	NEXT
OUTPUT	OUT	PRF	ON	OPEN	OR
PRINT	PRINTER	RENAME	PAUSE	PEEK	POKE
RESTORE	PURGE	RUN	PRINTF	READ	RECTANGLE(*)
REQUEST	RETURN	SRQ	REM	REMOTE	REN
SEND	SPRINTF	THEN	SAVE	SCRATCH	SELECT
TALK	TEXT	UNTIL(*)	STEP	STOP	SYSTEM(*)
UNL	UNT		TIME	TO	TRIGGER
WAIT	XOR		USE	USING	VIEWPORT(*)

（注）キー・ワードは、大文字で使用します。

(*): 予約されているキー・ワードです。使用されてませんが、変数名には使用できません。

表 4 - 2　フルネームとショートネームの対応表

フルネーム	ショートネーム
CURSOR	CSR
ENTER	ENT
INITIALIZE	INIT
INPUT	INP
OUTPUT	OUT
PRINTF	PRF
USING	USE
PRINT	?

(3)　式

式は、オブジェクトと演算子からなり、文法上、式を指定できるところならば、どこにでも置くことができます。（ただし、従来のBASIC との互換性を考えて、IF文の条件式では＝を等号と解釈するため、代入式を書くことはできません。）
式には、演算結果の最終値がどのデータ型を取るかによって、以下の 3通りあります。

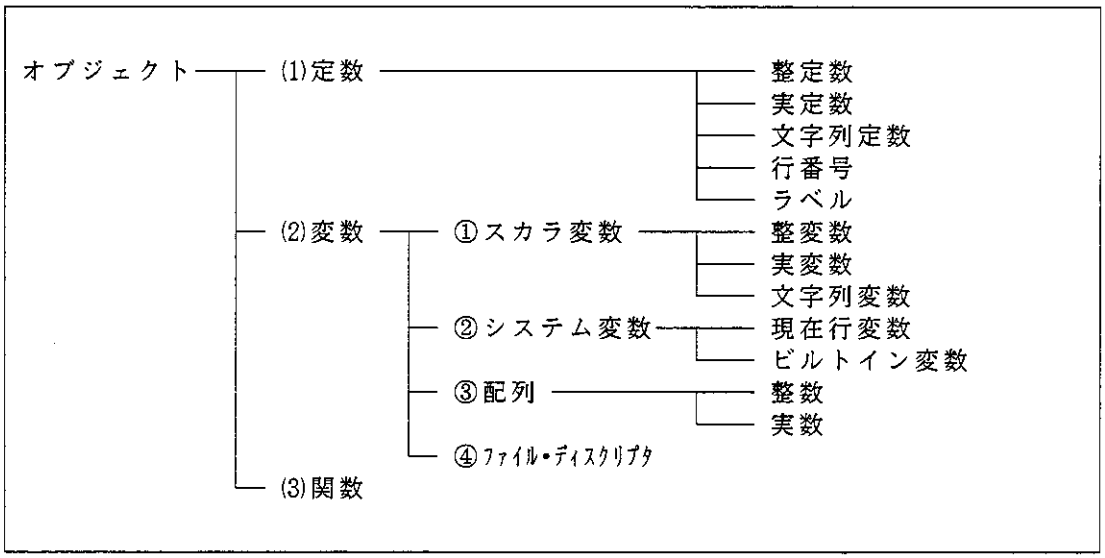
〈算術式 〉〈 文字列式 〉〈 論理式 〉

算術式 ： 整数値または実数値になる。

論理式 ： 内部に論理演算子を含むということには関係なく構文で決まり、最終値を論理値として評価する（つまり、0 が偽、0 以外が真になる）。

4.1.2　オブジェクト

BASICが処理する対象となるものを「オブジェクト」と呼びます。オブジェクトには定数、変数および関数があり、各データ型は以下のようになっています。



(1)
定数

・整定数

プログラム内部で小数点のない定数は整数とみなし、内部で 4バイト使い表現するので、－2,147,483,648 ～＋2,147,483,647 までの数値を表現できます。

・実定数

小数点付や1E＋20のような浮動小数点表現の定数は実数とみなし、内部で 8バイト(IEEE)を使って表現するので、約－1E＋308 ～1E＋308 まで表現でき、15桁の精度をもちます。

・文字列定数

文字列を表現するには、その文字列をダブル・クォーテーション(") で囲みます。
文字列は "" の空文字列から最大 128文字まで指定できます。構成する文字の単位は 8ビットで、 0～255 までの 256種類の文字単位を表現できます。
文字コードは ASCIIを使用し、 128～255 までは特殊なシンボルが登録されています。
コードがキー・ボードに割り当てられていないものをプログラムで表現するために、またはINPUT 文に対する入力のために、＼を使用して＼f(フォーム・フィールド) という方法があります。同様にして、ダブル・クォーテーション (")を文字列に含めるために＼" と書くことができます。
ASCII の制御文字を表現するために、以下のエスケープ・シーケンスが用意されています。

ｴｽｹｰﾌﾟ･ｼｰｹﾝｽ	意味	8進	10進
＼b	バック・スペース	010	8
＼t	水平タブ	011	9
＼n	ライン・フィード(ニューライン)	012	10
＼v	垂直タブ	013	11
＼f	フォーム・フィード(クリア・スクリーン)	014	12
＼r	キャリッジ・リターン	015	13

・行番号

1 ～65535 の整数で表され、BASIC プログラムの行を指定します。

・ラベル

行番号の代わりに、ラベルを使用することができます。プログラムの先頭でアスタリスク (*)を付けて宣言します。
使用できる文字は変数と同じですが、変数ではないので代入できません。またラベルが書ける位置は、[4.3　ステートメント文法と活用] で説明する行番号、またはラベルと書かれている部分です。

(2)　変数

変数名は、文字を先頭とするアルファニューメリックで構成し、最大20文字です。

変数名の最後が $の場合　　：文字列変数になる。
変数名の最後が(整数値)の場合　：配列型の変数とみなされる。
　変数は特に INTEGER文で宣言されなければ実数型になる。

表 4 - 3　アルファニューメリック

```
1, 2, 3, 4, 5, 6, 7, 8, 9, 0
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
 _
```

例）変数の型

```
value, v123          : 実変数
string$, s123$       : 文字列変数
array(3)             : 配列型実変数
INTEGER code         : 整変数
INTEGER week(7)      : 配列型整変数
```

① スカラ変数

・整変数
・実変数
・文字列変数

数値型の変数は、特に初期化しない限り 0が割りあてられます。したがって、特定の値に初期化すべき変数は、プログラム内で明示的に値を代入する必要があります。
各データ型に格納できる値は、定数の場合と同じ大きさです。
文字列変数には配列がありません。文字列には、文字列定数と同様に長さの属性があります。長さを宣言するには、DIM文を使用します。

```
DIM  string$[100]
```

宣言をせずに参照すると、18文字の文字列であるとみなされます。
文字列はサブ・ストリング演算子 ([ ])を使用して、文字列の一部を扱うことができます。

4.1.3 項の [(7)　サブ・ストリング演算子] を参照して下さい。

```
string$ ="ADVANTEST CORPORATION"
PRINT string$[1,14] ;"."
```

結果

```
ADVANTEST CORP.
```

② システム変数

・現在行変数　@

現在実行しているプログラムの行番号を格納しています。値は代入できません。

LIST @：現在実行している行を表示

・ビルトイン変数

BASICの起動時に自動的に登録される変数で、固有の値で初期化され、特定の値を代入することによって変更できます。起動時の値に戻すには明示的にその値を代入するか、SCRATCH 1, SCRACH で初期化します。

```
PI  : 3.14159 .......
EXP : 2.71828 .......
```

③　配列

配列の宣言は、DIM, INTEGER文を使用します。

・数値型配列

宣言せずに参照すると、その配列の大きさ、つまり要素数は10になります。以下のように宣言したものと同じになります。添字は必ず 1から割当られます。

```
DIM     array(10)
INTEGER array(10)
```

```
実数型配列    DIM       real(20)
整数型配列    INTEGER   int(30,40)
```

④　ファイル・ディスクリプタ

BASIC でファイルを読み書きするのは、ファイル・ディスクリプタを使用して行います。宣言は必要ありませんが、OPENによって実際のファイル名と結び付けられます。OPENされたあとは、ENTER やOUTPUTを使用してファイル・ディスクリプタを指定し、ファイルを参照します。ファイル・ディスクリプタは、特別な変数で他の変数のように、演算を行ったりPRINT したりすることはできません。

(3)　関数

関数はすべて組み込み型です。関数は、その戻り値で整数型、実数型または文字列型に分けられます。また、この関数呼び出しを演算式の中に記述できるため、変数と同じようになります。

```
string$ ="ADVANTEST"
PRINT string$
A = NUM("A")
a = NUM("a")
FOR idx = 1 TO LEN(string$)
      b = NUM(string$[idx;1]) - A + a
      string$[idx;1]=CHR$(b)
NEXT idx
PRINT string$
```

結果

```
ADVANTEST
advantest
```

・組み込み関数

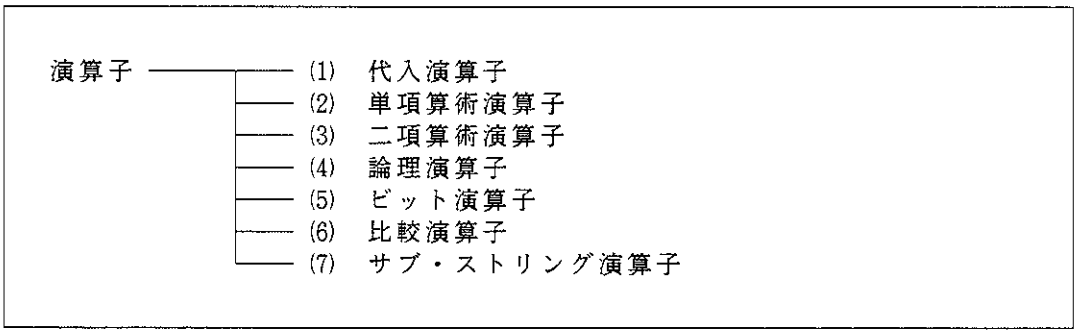
関数	機能
SIN(算術式) COS(算術式) TAN(算術式) ATN(算術式)	正弦 (sin) 余弦 (cos) 正接 (tan) 逆正接 ($\tan^{-1}$)　　角度の単位= ラジアン
LOG(算術式)	自然対数
SQR(算術式)	平方根
ABS(算術式)	絶対値
NUM(文字列式)	文字列式の先頭の 1文字の ASCIIコードを返す。 例)　NUM("A") ---> 65
CHR$ (算術式)	算術式の値に対応する ASCII文字 1文字の文字列式を返す。 例)　CHR$(65) ---> "A"
LEN(文字列)	文字列式の長さを返す。 例)　LEN("ADVANTEST") ---> 9
POS(文字列式1,文字列式2)	文字列式1の中から、文字列式2がある位置の先頭の位置を返す。 例)　POS("ADVANTEST" , "AN") ---> 4
ビルトイン関数	測定値を扱うための関数 詳細は [4.4 ビルトイン関数] を参照して下さい。

文字列から数値変数、数値変数から文字列への変換の組み込み関数はありませんが、代入文で変換されます。

例) A$=A
　　A="123.4"

4.1.3　演算子

オブジェクトを操作するものが演算子です。演算子とオブジェクトの組み合わせで式を構成します。



(1)　代入演算子

従来の BASICにあった LETというキー・ワードはありません。代入式はそれ自体の値をもち、 1つの式となります。

```
PRINT a=1                ---> 1.0
PRINT a$="ADVANTEST"     --->"ADVANTEST"
PRINT (a=1)+a            ---> 2.0
```

代入演算子を以下に示します。

```
=       : 普通の代入
          文字列変数への代入では、右辺の有効文字数分だけ転送します。
          例)   DIM string$ [20]
                PRINT LEN(string$ ="12345")
              結果
                        5
=       : =の左辺のデータ型に変換して代入
          例)   string$ = 123.456    ---> "123.456"
                numeric = "123"      ---> 123
                integer = 123.456    ---> 123
+=      :  a  += 10   --->  a = a  + 10
-=      :  a  -= 10   --->  a = a  - 10
*=      :  a  *= 10   --->  a = a  * 10
/=      :  a  /= 10   --->  a = a  / 10
%=      :  a  %= 10   --->  a = a  % 10
=<      : 文字列を左詰めで代入
=>      : 文字列を右詰めで代入
```

(2) 単項算術演算子

－　　：マイナス符号
＋　　：プラス符号
＋＋　：前置/ 後置インクリメント
　前置 b = ++a　　…… aに 1を加えてから、 bに代入する。
　後置 b = a++　　…… bに代入してから、 aに 1を加える。
－－　：前置/ 後置デクリメント
　前置 b = --a　　…… aから 1を引いてから、 bに代入する。
　後置 b = a--　　…… bに代入してから、 aから 1を引く。

例）　a = 10: PRINT a++: PRINT a: PRINT --a: PRINT --a: PRINT a
　結果
　　10.0
　　11.0
　　10.0
　　9.0
　　9.0

注）前置/ 後置インクリメント・デクリメントは、定数（実定数、整定数）に対して演算することはできません。

(3) 二項算術演算子

＋　　：加算
－　　：減算
＊　　：積算
／　　：割算
％　　：モジュロ（余り）
^　　：累乗
＆　　：文字列の連結

(4) 論理演算子

NOT	例	NOT 1	結果 0
AND	例	1 AND 0	結果 0
OR	例	1 OR 0	結果 1
XOR	例	1 XOR 1	結果 0

(5) ビット演算子

数式は整数型のみ有効で、実数型ではエラーになります。

BNOT	例	BNOT 0	結果 -1
BAND	例	2 BAND 3	結果 2
BOR	例	2 BOR 3	結果 3
BXOR	例	2 BXOR 3	結果 1

(6)　比較演算子

比較演算子には以下のものがあり、真ならば 1を、偽ならば 0を取ります。BASICの構文上で比較演算を行うところでは、演算結果として最終的に 0になったとき、偽と判定します。それ以外の値はすべて真になります。

＝　　：イコール
＜＞　　：ノットイコール（または！＝）
＜
＞
＜＝
＞＝

この比較演算はIF文の条件では、必らず論理演算を行うので、演算子の＝は無条件に比較演算子であるとみなします。したがって、代入式をIF文の条件式内に含めることはできません。

(7)　サブ・ストリング演算子

文字列式の一部を文字列として指定できます。

文字列式［算術式1,　算術式2］：文字列式の先頭から、算術式1 が表現する値だけ進んだ所から算術式2 が表わす値の所までをサブ・ストリングとする。

"ADVANTEST"［1,5］　---> "ADVAN"

文字列式［算術式1 ; 算術式2］：文字列式の先頭から、算術式1 が表現する値だけ進んだ所から算術式2 が表わす値の文字数分をサブ・ストリングとする。

"ADVANTEST"［6;4］　---> "TEST"

4.2　各種ステートメント

4.2.1　ステートメント機能一覧

(1)　基本ステートメント　(1/2)

ステートメント	機能
BUZZER	ブザーを鳴らす
CLS	ディスプレイ表示をクリア
CONSOLE	スクロール範囲を指定
CURSOR	カーソルを移動
DATA	READ文で読み込むための数値、文字列を定義
DATE$	本器に内蔵している時計(RTC) の日付の値を読み出す
DIM	配列変数または文字列変数を定義
DISABLE INTR	割り込みの受付を禁止
ENABLE INTR	割り込みの受付を許可
ERRM$	エラー・メッセージを返す
ERRN	エラー番号を返す
FOR-TO-STEP, NEXT, BREAK, CONTINUE	ループ処理を行う
FRE	BASIC プログラム・メモリの残量を返す
GOSUB, RETURN	サブルーチンへの分岐、復帰
GOTO	指定行への分岐
GPRINT	数値または文字列をGPIBへ出力
IF-THEN, ELSE, END IF	条件付分岐
INPUT	キーからの入力
INTEGER	変数が整数型であることを定義
KEY$	本器のパネル・キーのコードを返す
LPRINT	数値または文字列をシリアル・ポートへ出力
LET	変数への代入
OFF ERROR	BASIC エラーを検出したときの分岐の解除
OFF ISRQ	ISRQによる割り込み分岐を解除
OFF KEY	キー入力による割り込み分岐を解除
OFF SRQ	SRQ による割り込み分岐を解除
ON DELAY	指定時間経過後に分岐
ON ERROR	BASIC エラーを検出したときの分岐の定義
ON ISRQ	本器内部要因による割り込み分岐を定義
ON KEY	キー入力による割り込み分岐を定義
ON SRQ	GPIB外部SRQ 信号による割り込み分岐を定義
PRINT [USING]	数値または文字列を表示
PRINTER	プリンタのGPIBアドレスを設定
PRINTF	数値または文字列を表示
READ	DATA文の定数を変数に代入
REM	注釈
RESTORE	次のREAD文で読み込むDATA行を指定
SELECT, CASE, END SELECT	式の値を条件として複数の分岐を行う
SPRINTF	PRINTFの書式に従った結果を文字列へ代入

(2/2)

ステートメント	機能
TIME$	本器に内蔵されている時計(RTC) の時刻の値を返す
TIMER	内蔵システム時間の読み出しおよびリセット
WAIT	指定時間だけ待つ
WAIT EVENT	指定したイベントが発生するまで待つ

(2)　GPIB制御用ステートメント

ステートメント	機能
CLEAR	デバイス・クリア
DELIMITER	ブロック・デリミタを指定
ENTER	GPIBからの入力
INTERFACE CLEAR	GPIBインタフェース・クリア
LOCAL	リモート・コントロールを解除
LOCAL LOCKOUT	ローカル・ロックアウト
OUTPUT	GPIBへの出力
REMOTE	リモート・コントロール
REQUEST	ステータス・バイトを設定
SEND	GPIBへコマンド、データなどを出力
SPOLL	ステータス・バイトの読み出し
TRIGGER	グループ・エグゼキュート・トリガを出力

(3)　ファイル制御用ステートメント

ステートメント	機能
CLOSE	ファイルを閉じる
DSTAT	ｶﾚﾝﾄ･ﾄﾞﾗｲﾌﾞ のﾃｨﾚｸﾄﾘの内容を BASIC変数に取り込む
ENTER [USING]	ファイルからデータを読み込む
OFF END	ON ENDで指定した処理を解除
ON END	エンド・オブ・ファイル時の処理を定義
OPEN	ファイルをオープン
OUTPUT [USING]	ファイルにデータを出力（書き込む）

4.2.2　ステートメント文法一覧

(1)　基本ステートメント　(1/2)

ステートメント	文法
BUZZER	BUZZER <音程> <時間>
CLS	CLS
CONSOLE	CONSOLE　<開始行>,<終了行>
CURSOR	CURSOR <X軸>,<Y軸>
DATA	DATA　数値定数｜文字列定数 {, 数値定数｜文字列定数}
DATE$	(1) DATE$
	(2) DATE$ ="YY/MM/DD"
DIM	DIM　<B> ｜ <C> {,　<B>｜ <C>}
DISABLE INTR	DISABLE INTR
ENABLE INTR	ENABLE INTR
ERRM$	ERRM$(エラー番号)
ERRN	ERRN
FOR-TO-STEP,NEXT,BREAK,CONTINUE	FOR 数値変数= 数値表現式 TO 数値表現式 [STEP数値表現式]
	[BREAK]
	[CONTINUE]
	NEXT [数値変数]
FRE	FRE(数値)
GOSUB,RETURN	GOSUB 行番号｜ラベル
	RETURN
GOTO	GOTO　行番号｜ラベル
GPRINT	GPRINT [A　{,　｜;A}]
IF-THEN,ELSE,END IF	(1) IF <条件式> THEN <文>
	(2) IF <条件式> THEN
	[ELSE IF <条件式> THEN]
	[　複文　　]
	[ELSE　]
	[　複文]
	END IF
INPUT	INPUT ["< 文字列>",] A {,A}
INTEGER	INTEGER <B>　{,<B>}
KEY$	KEY$
LPRINT	LPRINT [A　{,　｜;A}]
LET	LET <D> ｜<E>　{:<D>｜<E> }
OFF ERROR	OFF ERROR
OFF ISRQ	OFF ISRQ
OFF KEY	OFF KEY [キーコード]
OFF SRQ	OFF SRQ
ON DELAY	ON DELAY　時間　GOTO｜GOSUB 行番号｜ラベル
ON ERROR	ON ERROR GOTO ｜GOSUB 行番号｜ラベル
ON ISRQ	ON ISRQ GOTO｜GOSUB 行番号｜ラベル

A : 数値表現式｜文字列表現式
B : 数値変数名 [(数値表現式　{, 数値表現式})]
C : 文字列変数 [数値表現式]
D : 数値変数= 数値表現式
E : 文字列変数= ｜=<｜=>文字列表現式

(2/2)

ステートメント	文法
ON KEY	ON KEY キーコード GOTO | GOSUB 行番号|ラベル
ON SRQ	ON SRQ GOTO | GOSUB 行番号|ラベル
PRINT [USING]	(1) PRINT [A {, | ;A}]
	(2) PRINT USING 書式設定式 ; {,A}
PRINTER	PRINTER　数値表現式
PRINTF	PRINTF　書式表現式 {,A}
READ	READ　入力項目 {, 入力項目}
REM	REM [文字列] または　![文字列]
RESTORE	RESTORE 行番号|ラベル
SELECT,CASE,END SELECT	SELECT <数式|文字列式>
	CASE <数式|文字列式>
	複文
	[CASE ELSE]
	[複文]
	END SELECT
SPRINTF	SPRINTF 文字列変数　書式指定 {,A}
TIMER	TIMER(0|1)
TIME$	(1) TIME$
	(2) TIME$ ="HH:MM:SS"
WAIT	WAIT　時間
WAIT EVENT	WAIT EVENT <イベント番号>

A : 数値表現式 | 文字列表現式

●PRINT USING の書式指定は、以下に示すイメージ仕様をｺﾝﾏ (,) で区切って指定します。

ｲﾒｰｼﾞ仕様

D　: D の数で出力桁数を指定する。指定フィールドの余った部分にはスペースが入る。
Z　: Z の数で出力桁数を指定する。指定フィールドの余った部分には0 が入る。
K　: 式の値をそのまま表示する。
S　: S の位置に+ または- のサイン・フラグを付けて表示する。
M　: M の位置に、負のときは- 、正のときはスペースを付けて表示する。
.　: . の位置に小数点がくるように位置を合わせて表示する。
E　: 指数形式 (e, 符号, 指数) で表示する。
H　: K と同じ。ただし、小数点にカンマ(,) を使う。
R　: . と同じ。ただし、小数点にカンマ(,) を使う。
*　: * の数で出力桁数を指定する。指定フィールドの余った部分には *が入る。
A　: 1 文字を表示する。
k　: 文字列式をそのまま表示する。
X　: スペース 1文字を表示する。
ﾘﾃﾗﾙ　: 書式指定式にリテラルを書くときは＼" で囲む。
B　: 式の値をASCII コードとして表示する。
@　: 改ページする。
+　: 表示の位置を同じ行の先頭に移動させる。
-　: 改行する。
#　: 最後に改行しない。
n　: 数字で各イメージ仕様の繰り返し回数を指定できる。

●PRINTFの書式指定は、以下に示す方法で %に続けて指定します。

%〔-〕〔0〕〔m〕〔. n〕文字

- -　　：指定されたフィールド内で左詰めにする（指定がなければ右詰め）。
- 0　　：指定フィールドの余った部分に詰める文字を 0にする。
- m　　：m 文字分のフィールドを取る。
- . n　：n 桁の精度で出力する。文字列の場合は、この値が実際の文字列の長さになる。
- 文字　：d ; 符号付10進数　　s ; 文字列
  - o ; 8 進数　　e ; 浮動小数点表現（指数形式）
  - x ; 16進数　　f ; 浮動小数点表現

(2) GPIBステートメント

ステートメント	文法
CLEAR	CLEAR［装置ｱﾄﾞﾚｽ {, 装置ｱﾄﾞﾚｽ} ］
DELIMITER	DELIMITER 数値表現式
ENTER	ENTER 装置ｱﾄﾞﾚｽ ; B {,B}
INTERFACE CLEAR	INTERFACE CLEAR
LOCAL	LOCAL［装置ｱﾄﾞﾚｽ {, 装置ｱﾄﾞﾚｽ} ］
LOCAL LOCKOUT	LOCAL LOCKOUT
OUTPUT	OUTPUT 装置ｱﾄﾞﾚｽ {, 装置ｱﾄﾞﾚｽ} ; A {,A}
REMOTE	REMOTE［装置ｱﾄﾞﾚｽ {, 装置ｱﾄﾞﾚｽ} ］
REQUEST	REQUEST 整数
SEND	SEND <C> | <D> {,<C> | <D> }
SPOLL	SPOLL （装置ｱﾄﾞﾚｽ）
TRIGER	TRIGER［装置ｱﾄﾞﾚｽ {, 装置ｱﾄﾞﾚｽ} ］

A : 数値表現式 | 文字列表現式
B : 数値変数［文字列変数
C : <CMD | DATA | LISTEN | TALK>［ 数値表現式 {, 数値表現式} ］
D : UNL | UNT

(3)　ファイル制御用ステートメント

ステートメント	文法
CLOSE	CLOSE #FD | *
DSTAT	(1) DSTAT 0 <ファイル数> (2) DSTAT <index> < ﾌｧｲﾙ名 > <属性> <ｻｲｽﾞ> < ｾｸﾀ数 > < 年 > < 月 > <日> <時> <分> <開始ｾｸﾀ> (3) DSTAT ;SELECT <文字列> COUNT <変数>
ENTER [USING]	(1) ENTER #FD ; 入力項目 {, 入力項目} (2) ENTER #FD USING "ｲﾒｰｼﾞ仕様" ; 入力項目 {, 入力項目} }
OFF END	OFF END #FD
ON END	ON END #FD GOTO | GOSUB 整数 | ラベル式
OPEN	OPEN "ﾌｧｲﾙ名" FOR 処理ﾓｰﾄﾞ AS #FD [; ﾀｲﾌﾟ]
OUTPUT [USING]	(1) OUTPUT #FD ; 出力項目 {, 出力項目} (2) OUTPUT #FD USING " ｲﾒｰｼﾞ仕様" ; 出力項目 {, 出力項目}

FD　　：ファイル・ディスクリプタ
処理モード　：INPUT | OUTPUT
タイプ　　：BINARY | TEXT | ASCII

●ENTER USING のイメージ仕様

ｲﾒｰｼﾞ仕様

D　：D の数を入力桁と解釈して読み込み、入力項目の変数に代入する。
Z　：D と同じ。
K　：1 行読み込み、数値データに変換し、入力項目の変数に代入する。
S　：D と同じ。
M　：D と同じ。
.　：D と同じ。
E　：K と同じ。
H　：K と同じ。ただし、小数点にカンマ(,) を使う。
*　：D と同じ。
A　：A の数分の文字を読み込み、文字列変数に代入する。
k　：1 行読み込み文字列変数に代入する。
X　：1 文字のデータを読み飛ばす。
ﾘﾃﾗﾙ　：＼" で囲まれた文字列の数のデータを読み飛ばす。
B　：1 文字読み込み、入力項目に ASCIIコードとして代入する。
@　：1 バイトのデータを読み飛ばす。
+　：@ と同じ。
-　：@ と同じ。
#　：ENTER では無視される。
n　：数字で各イメージ仕様の繰り返し回数を指定できる。

●OUTPUT USINGのイメージ仕様

ｲﾒｰｼﾞ仕様

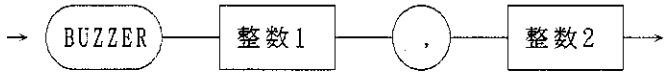
D	: D の数で出力桁数を指定する。指定フィールドの余った部分にはスペースが入る。
Z	: Z の数で出力桁数を指定する。指定フィールドの余った部分には0 が入る。
K	: 式の値をそのまま表示する。
S	: S の位置に+ または- のサイン・フラグを付けて出力する。
M	: M の位置に、負のときは- 、正のときはスペースを付けて出力する。
.	: . の位置に小数点がくるように位置を合わせて出力する。
E	: 指数形式（e, 符号, 指数）で出力する。
H	: K と同じ。ただし、小数点にカンマ(,) を使う。
R	: . と同じ。ただし、小数点にカンマ(,) を使う。
*	: * の数で出力桁数を指定する。指定フィールドの余った部分には* が入る。
A	: 1 文字を出力する。
k	: 文字列式をそのまま出力する。
X	: スペース 1文字を出力する。
ﾘﾃﾗﾙ	: ＼" で囲まれた文字列を出力項目とは無関係にそのまま出力する。
B	: 式の値をASCII コードとして出力する。
@	: フォーム・リード（改ページ）を出力する。
+	: キャリッジ・リターンを出力する。
-	: ライン・フィード（改行）を出力する。
#	: 最後の項目の後ろにライン・フィードを付けない。
n	: 数字で各イメージ仕様の繰り返し回数を指定できる。

4.3　ステートメント文法と活用

1.　BUZZER

概　要　ブザーを鳴らします。

構　文　(1)-1



(1)-2

```
BUZZER  整数1  ,  整数2
```

注）整数1 は、高低音を0(高音）～65535(低音）の範囲で指定します。
　　整数2 は、時間(ms 単位）を示します。

解　説　・本器内蔵のブザーを指定にしたがって鳴らします。

例

```
10 FOR I=0 TO  255
20   BUZZER I,  10
30 NEXT I
40 STOP
```

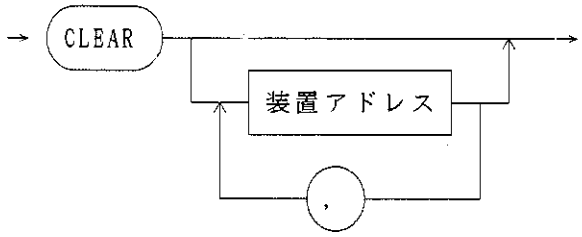
2.　CLEAR

概　要

GPIB上に接続されたすべての装置または選択された特定の装置を初期状態にします。

構　文

(1)-1



(1)-2

CLEAR 〔装置アドレス｛，装置アドレス｝〕

解　説

・装置アドレスを指定せずにCLEAR だけを実行すると、GPIB上にユニバーサル・コマンドのデバイス・クリア(DCL) を送ります。これによって、GPIBに接続されているすべての装置を初期状態にできます。

・CLEAR に続いて装置アドレスを指定すると、装置アドレスで指定された装置のみをアドレスし、アドレス・コマンドのセレクト・デバイス・クリア(SDC) を送ります。これによって、特定の装置のみを初期状態にします。なお、装置アドレスは複数指定できます。

・CLEAR で各装置に設定される初期状態の定義は、各装置に依存します。

例

```
10  CLEAR
20  CLEAR 2
30  CLEAR 1, 3, 5, 7
```

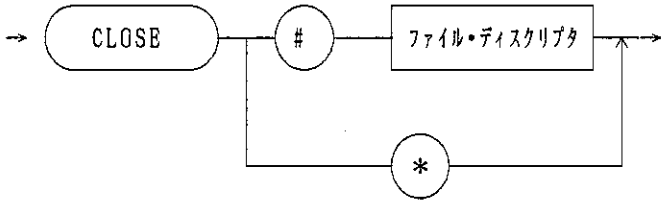
注　意

ADDRESSABLE モードでは機能しません。

3.　CLOSE

概　要　ファイル・ディスクリプタに割り当てられているファイルをクローズします。

構　文　(1)-1



(1)-2

CLOSE <#ファイル・ディスクリプタ | ＊>

解　説

・OPENコマンドでオープンしたファイルは、フロッピーを抜く前や、装置の電源をOFF する前に、必ずすべてのファイルをクローズしなければなりません。クローズしないとファイルは破壊されます。

・BASICプログラムでは、PAUSEやSTOPキーで停止させたときはファイルを自動的にはクローズしません。それ以外のときはプログラムの終了とともにすべてのファイルをクローズします。エラー終了時もクローズしますが、ON ERRORの設定がある場合は、クローズしません。

以上のような理由からエラー終了時には、以下の方法（コマンドですべてのファイルをクローズする指定方法）で明示的にクローズ動作を実行して下さい。

CLOSE *

・ファイルは、SCRATCH やLOADなどを実行したときにも自動的にクローズします。

※　ファイルの取り扱いについては、「1.　はじめに　●ファイルの管理」を参照して下さい。

4.　CLS

概　要　ディスプレイの表示をクリアします。

構　文　(1)-1

(1)-2
CLS

解　説
・ディスプレイに表示されているキャラクタをクリアすると同時に、カーソルをホーム・ポジションに戻します。

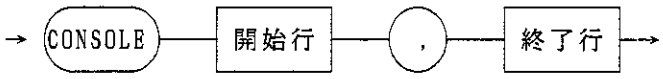
・CONSOLE によって指定されたスクロール範囲内のみクリアします。

例　10 CLS

5.　CONSOLE

概　要　スクロール範囲を指定します。

構　文　(1)-1



(1)-2
CONSOLE　開始行　,　終了行

（注）　開始行よりも小さい値を終了行に指定した場合、終了行= 開始行として実行します。

解　説
・テキスト画面のスクロール範囲を設定します。

・開始行、終了行の指定範囲は、以下の通りです。
R3764/66（蛍光表示管）：　0～7
R3764/66（外部モニタ）：　0～29
R3765/67　　：　0～29

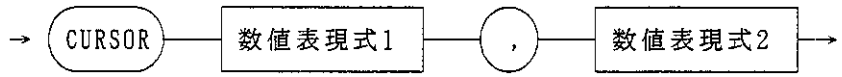
例

```
10 CONSOLE 0,5
20 PRINT "This is Network Analyzer"
30 PRINT "....Sweep Check Program...."
40 STOP
```

6.　CURSOR

概　要　指定された座標位置にカーソルを移動させます。

構　文　(1)-1



(1)-2
CURSOR　数値表現式1　,　数値表現式2

注）数値表現式1 ： X 軸指定、カラム方向
　　数値表現式2 ： Y 軸指定、行方向
　　カンマ(,) をスペースにしても構いません。

解　説
- ディスプレイの指定された位置にカーソルを移動させます。
- 数値表現式1 が X軸座標、数値表現式2 が Y軸座標を示します。
- X軸座標、Y軸座標の指定範囲を以下に示します。
  R3764/66（蛍光表示管）：　0≦X≦31　0≦Y≦7
  R3764/66（外部モニタ）：　0≦X≦79　0≦Y≦29
  R3765/67　　：　0≦X≦66　0≦Y≦29

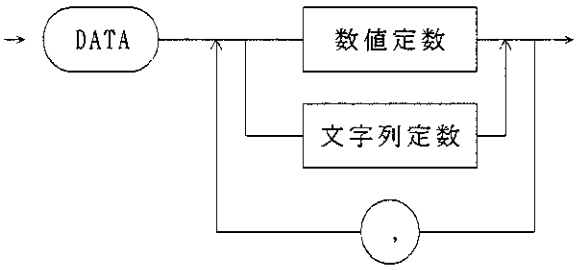
例

```
10 CLS
20 X=4:Y=4:X1=1:Y1=1
30 CURSOR X,Y:PRINT " ";
40 X=X+X1:Y=Y+Y1
50 CURSOR X,Y:PRINT "*";
60 IF X<=0 OR 67<=X THEN X1*=-1
70 IF Y<=0 OR 29<=Y THEN Y1*=-1
80 GOTO 30
90 STOP
```

7.　DATA

概　要　READ文で読み込むための数値、文字列を定義します。

構　文　(1)-1



(1)-2
DATA 〈数値定数 | 文字列定数〉 {, 〈数値定数 | 文字列定数〉 }

解　説
- DATA文は実行の対象とはならないため、どの文番号にあっても構いませんが、原則として、READ文で読み出す順序に従っている必要があります。
- READ文がプログラムの中からDATA文を捜して、対象となるデータを読み込むことになります。
- この順序を変更するには、RESTORE 文を使います。
- DATA文には、カンマ (,)かスペースで区切ることによって複数個の定数を指定できます。
  文字列は、文字列定数としてダブルクォーテーション (")で囲みます。
- DATAの後にコロン(:) によるマルチ・ステートメントは、使用できません。

注　意　DATA文に並べるパラメータは、変数を含む式ではいけません。

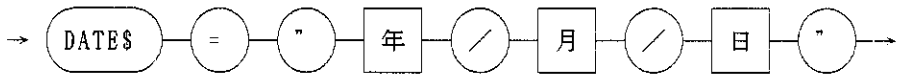
8.　DATE$

概　要　日付の読み出しと、日付の変更をします。

構　文　(1)-1

(1)-2
DATE$

(2)-1



(2)-2
DATE$=" 年／月／日"

解　説
・本器内蔵の時計（RTC)の日付を読み出します。

・読み出した日付は変更できます。
以下のように入力して下さい。

DATE$=" 93／1／1"
または
DATE$=" 93／01／01"

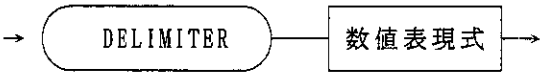
例

```
10 DIM D$[10]
20 D$=DATE$
30 PRINT "Date is ";D$
40 PRINT "Date Reset"
50 DATE$=" 93/1/1"
60 STOP
```

9.　DELIMITER

概　要　4種類のデリミタを選択し、設定するステートメントです。

構　文　(1)-1



(1)-2
DELIMITER 数値表現式

解　説

・数値表現式によって示される番号に対応したデリミタを設定します。
デリミタの選択番号および種類を下表に示します。

選択番号	デリミタの種類
0	CR、LFの 2バイト・コードを出力 LF出力と同時に、単線信号EOI も出力
1	LFの 1バイト・コードを出力
2	データの最終バイトと同時に、単線信号EOI を出力
3	CR、LFの 2バイト・コードを出力

・数値表現式の結果が 0～3の範囲を越えた場合は、エラーとなります。
また、小数点以下の数値は無視し、整数として取り扱います。

・電源投入時は、DELIMITER=0 が自動的に設定されます。

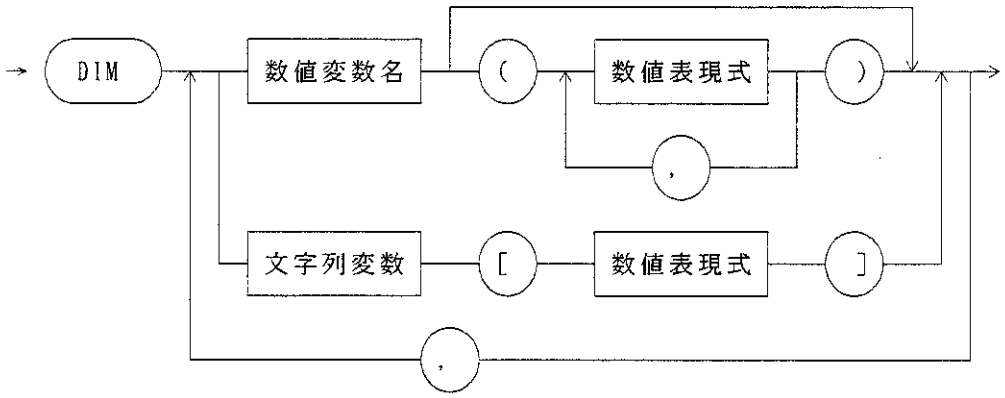
例

```
10   DELIMITER 0
20   DELIMITER 1
```

10.　DIM

概　要　配列変数または文字列変数の定義宣言を行います。

構　文　(1)-1



(1)-2
DIM <A | B> {, <A | B>}

注）A:数値変数名〔(数値表現式 {, 数値表現式}) 〕
　　B:文字列変数 [数値表現式]

解　説

・配列変数および文字列変数を使用するときは、DIM で配列変数名と配列の大きさを定義しなくてはなりません。定義をしないで配列変数を使うと、配列は 1次元で10個の要素数になり、文字列は18文字の長さになります。

・DIM で配列宣言をすると、指定された大きさの配列変数をメモリ上に確保します。したがって、大きすぎる配列宣言を行うと、BASIC プログラムの領域が足りなくなり、エラーとしてプログラムの実行を中止します。
(memory space full)

・配列変数の大きさを示す数値表現式は、演算の結果が実数表現となっても、小数点以下を切り捨て、整数として扱います。
配列変数に0 は使えません。

・文字列変数のとき、数値表現式は文字列の長さを宣言します。

例

```
10 DIM N(5)                          <実行結果>
20 FOR I = 1 TO 5                     0.5
30      N(I) = I*I/2                  2.0
40 NEXT I                             4.5
50 FOR I = 1 TO 5                     8.0
60      PRINT N(I)                    12.5
70 NEXT I
```

11. DISABLE INTR

概　要　割り込みの受付を禁止します。

構　文　(1)-1

(1)-2
DISABLE INTR

解　説

・DISABLE INTRは、ENABLE INTR で許可された割り込みを禁止します。

・DISABLE INTRの実行後に、再び割り込みを許可する場合は、ENABLE INTR を実行します。このとき、ON XXXステートメントで設定された分岐の条件は、以前の状態を保っています。
ただし、割り込み分岐の条件を変更する場合は、ENABLE INTR を実行する前にON XXX、またはOFF XXX の各ステートメントを用いて設定して下さい。

・プログラムを実行した直後は、ENABLE INTR を実行するまで割り込みは禁止状態になっています。

例

```
10  ON KEY 1 GOTO 60
20  ENABLE INTR
30  ! LOOP
40  GOTO 30
50  !
60  DISABLE INTR
70  PRINT "KEY 1 INTERRUPT"
80  STOP
```

12. DSTAT

概　要　ディレクトリの内容を BASIC変数に取り込みます。

構　文

(1)
```
DSTAT <index>  <変数>
```

(2)
```
DSTAT <index> <filename> <fileattribute> <size> <sectors>
              <year> <month> <day> <hour> <minutes> <start-sector>
```

(3)
```
DSTAT ; SELECT <文字列> COUNT  <変数>
```

解　説

・(1)の構文
ファイル・システムのディレクトリに登録されているファイルの数を調べます。1 つ目のパラメータ <index> には 0を指定します。
2 つ目のパラメータには数値変数を指定します。ここに結果が代入されます。

・(2)の構文
ファイル・システムのディレクトリ情報を BASIC変数に取り込みます。
1 つ目のパラメータ <index> で、ディレクトリ内のインデックスを指定します。指定可能な値は 1から登録ファイル数までです。（登録ファイル数は、(1)の構文で得られる値です）
2 つ目のパラメータには文字列変数を指定します。ここに結果のファイル名が格納されます。
3 つめのパラメータ以降には、すべて数値変数を指定します。ここに以下の内容が代入されます。

fileattribute	ファイル属性（複数の属性がある場合は、加算されて出力されます） 1.　READ ONLY 4.　SYSTEM FILE　　16. DIRECTORY 8.　VOLUME LABEL　　32. ARCHIVE FILE
size	ファイル・サイズ（バイト数）
sectors	セクタ数
year, month, day	ファイルの作成年月日
hour, minutes	ファイルの作成時刻
start-sector	ファイルの開始セクタ

・(3)の構文
〈文字列〉で指定したファイルの数を〈変数〉に代入します。
この構文は、指定ファイルがディレクトリ内に存在するかなどのファイルサーチとして使用できます。
指定した〈文字列〉の中に以下の文字がある場合は、特別な意味になります。

?	:	1文字と一致する。
*	:	1文字以上と一致する。
[ ]	:	[ ]で囲まれた文字列のいずれか 1文字と一致する。 [文字1 －文字2]で指定すると、文字1 から文字2 の範囲にある文字と一致する。

13.　ENABLE INTR

概　要　割り込みの受け付けを許可します。

構　文　(1)-1

(1)-2
ENABLE INTR

解　説

・ENABLE INTR は、割り込みの受け付けを許可し、ON XXXステートメント（ON ERROR は除く）で定義された割り込み分岐を有効にします。

・DISABLE INTRの実行後に再び割り込みを許可する場合は、ENABLE INTR を実行します。

・プログラムを実行させた直後は、ENABLE INTR を実行するまで割り込みは禁止状態になっています。(DISABLE INTR 状態)

例

```
10  ON KEY 1 GOTO 60
20  ENABLE INTR
30  ! LOOP
40  GOTO 30
50  !
60  PRINT "KEY 1"
70  GOTO 20
```

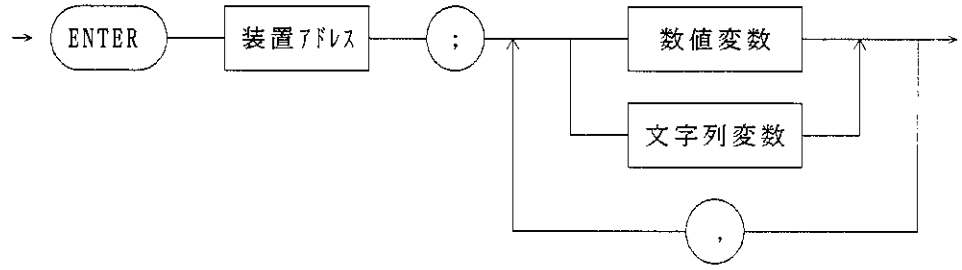
注意

ENABLE INTR を実行しても、ON XXXで定義された割り込みが発生すると、プログラムが分岐した時点で割り込み禁止状態となります。(DISABLE INTR 実行と同等)
これは、割り込み処理中に次の割り込みが発生した場合、割り込み処理が入れ子(Nest)にならないようにしているためです。
よって、続けて割り込み分岐を有効にしたい場合は、再度ENABLE INTR により割り込みを許可する必要があります。

14.　ENTER

概　要　(1)GPIBおよびパラレルI/O からデータを取り込みます。
(2)ファイルからデータを読み込み、入力項目に代入します。

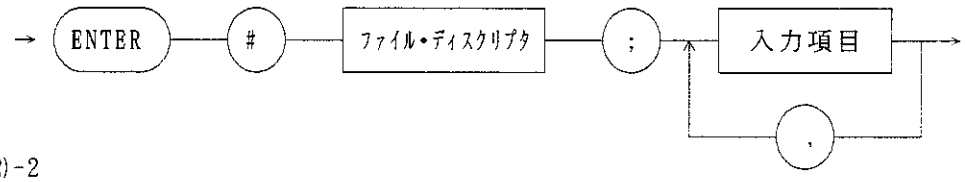
構　文　(1)-1



(1)-2
ENTER 装置アドレス;<数値変数 | 文字列変数> {,<数値変数 | 文字列変数>}

注）装置アドレス：0～30;　外部GPIB接続機器のアドレス
31　;　本器の測定部からのデータ入力
34　;　パラレル・ポートのFlip/Flop の状態の読み出し
35　;　パラレル・ポートの Cポートのデータ読み出し
36　;　パラレル・ポートの Dポートのデータ読み出し
37　;　パラレル・ポートのCDポートのデータ読み出し

(2)-1



(2)-2
ENTER # ﾌｧｲﾙ･ﾃﾞｨｽｸﾘﾌﾟﾀ ;　入力項目 {, 入力項目}

解　説　(1)の構文

・装置アドレスによって指定された装置からGPIBを通してデータを入力し、数値または文字列としてBASIC の変数内に蓄えます。ただし、装置アドレスによって指定された装置にトーカ機能がない場合、コントローラはハンドシェークを完了できずに停まってしまうので、注意して下さい。
また、文字列変数を使用する場合は、あらかじめ DIM文によって文字列変数を宣言しておかなければなりません。

・文字列で入力するときは、デスティネーションに使用する文字列変数の長さが十分でないと、入力データがオーバーフローを起こし、文字列変数に入りきらないデータは無視されるので、注意して下さい。

・例
```
10 ENTER 1;A
20 DIM A$(100), B$(20)
30 ENTER 2;A$
40 ENTER 3;B$
```

・注意
SYSTEM CONTROLLER モード時は、指定アドレスの機器をトーカに指定し、データを取り込みます。

⑵の構文
・ファイル・ディスクリプタに割り当てられているファイルから、データを対応する入力項目のデータ・タイプ形式で読み込んで、その入力項目に代入します。

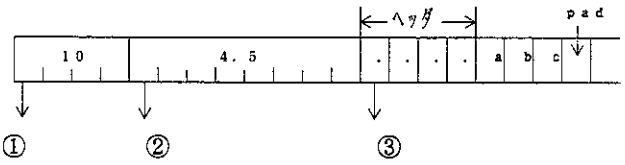
※　ファイルの取り扱いについては、「I.　はじめに　●ファイルの管理」を参照して下さい。

・例1）　BINARYファイル

内部データをそのそのままの形で代入します。入力項目が整数のとき 4バイト、実数は 8バイト、文字列は 4バイトのヘッダをそれぞれ読み込んだ後、ヘッダの内容が示すバイト数のデータを読み込みます。
読み込むバイト数は入力項目の型で決まるので、OUTPUTのときと同じ型で入力しないと、データの内容が違ってしまいます。

```
10 INTEGER I
20 DIM R
30 OPEN "FILE" FOR INPUT AS #FD
40 ENTER #FD;I,R,S$
```

代入する変数のタイプによって読み込むバイト数が違ってきます。

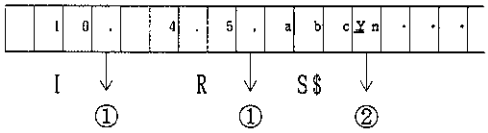


① ： 変数が整数のときは、 4バイトのデータを読み込み、そのままのデータを変数に代入します。
② ： 変数が実数のときは、 8バイトのデータを読み込み、そのままのデータを変数に代入します。
③ ： 変数が文字列のときは、ヘッダ 4バイトを読み込み、ヘッダが示す長さ分だけ読み込み、文字列変数に代入します。

・例2)　TEXTファイル

入力項目の数に関わらず、ライン・フィードまで読み込みます。カンマ(,)までが 1つのデータとなり、入力項目の型に変換して代入されます。入力項目の数が実際のデータより多いときは、多い分の変数には代入されません。
したがって、それらは前に格納されていた値が残ります。
逆に変数の数が実際のデータ数より少ないときは、データが捨てられます。

```
10 INTEGER I
20 DIM R
30 OPEN "FILE" FOR INPUT AS #FD;TEXT
40 ENTER #FD;I,R,S$
```

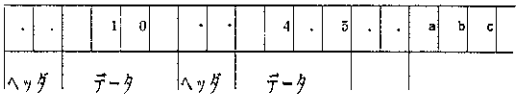


①：　各項目はカンマ(,)で区切られます。
②：　最後の項目の後にはライン・フィードがあります。

・例3)　ASCII ファイル

ヘッダ 2バイトを読み込み、ヘッダが示す長さのデータを読み込みます。変数に型にデータを変換し代入します。

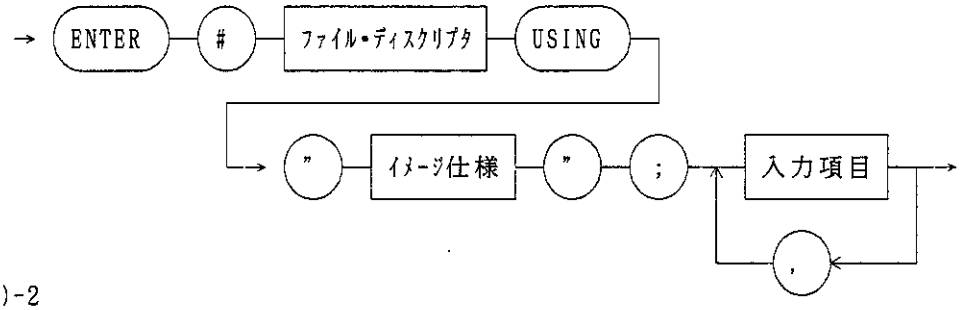
```
10 INTEGER I
20 DIM R
30 OPEN "FILE" FOR INPUT #FD;ASCII
40 ENTER #FD;I,R,S$
```



15.　ENTER USING

概　要　ファイルからイメージ仕様のフォーマットで入力項目に入力します。

構　文　(1)-1



(1)-2
ENTER #ファイル・ディスクリプタ USING "イメージ仕様" ; 入力項目 {, 入力項目}

注）ENTERはENT 、USING はUSE と省略できます。

解　説　ファイル・ディスクリプタに割り当てられているファイルからイメージ仕様のフォーマットで入力項目にデータを入力します。TEXTファイルとしてOPENされた場合のみ有効です。

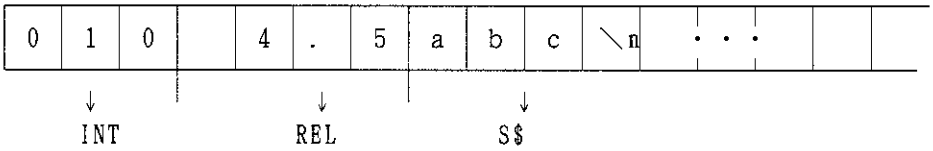
イメージ仕様

D	:	D の数を数値の桁数と解釈して数値を読み込み、入力項目の変数に代入する。
Z	:	D と同じ。
K	:	1 行読み込み、数値データに変換し、入力項目の変数に代入する。
S	:	D と同じ。
M	:	D と同じ。
.	:	D と同じ。
E	:	K と同じ。
H	:	K と同じ。ただし、小数点にカンマ(,) を使う。
*	:	D と同じ。
A	:	A の数分の文字を読み込み、文字列変数に代入する。
k	:	1 行読み込み文字列変数に代入する。
X	:	1 文字のデータを読み飛ばす。
リテラル	:	＼”で囲まれた文字列の数のデータを読み飛ばす。
B	:	1 文字読み込み、入力項目に ASCIIコードとして代入する。
@	:	1 バイトのデータを読み飛ばす。
+	:	@ と同じ。
-	:	@ と同じ。
#	:	ENTER では無視される。
n	:	数字で各イメージ仕様の繰り返し回数を指定できる。 3D.2D はDDD.DDと同じで、4AはAAAAと同じ。

※　ファイルの取り扱いについては、「1.　はじめに　●ファイルの管理」を参照して下さい。

例

```
10 INTEGER INT
20 DIM     REL
30 ENTER #FD USING "ZZZ,DD.D,3A";INT,REL,S$
```

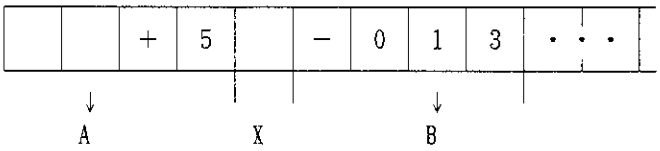


INT :　3 バイトのデータを読み込み、INT のデータタイプの整数型に変換し、INTに代入します。実行後の INTの値は10になります。

REL :　イメージ仕様の DD.D が入力項目の RELに対応します。
4 バイトのデータを読み込み、データを実数型に変換し RELに代入します。実行後の RELは 4.5になります。

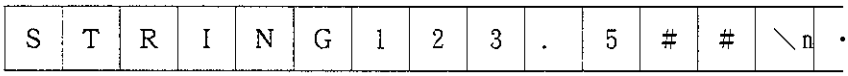
S$:　3 バイトのデータを読み込み、S$に代入します。
実行後のS$は abcです。

```
10  DIM A,B
20  ENTER #FD USING "SDDD,X,MZZZ";A,B
```



A,B : 4 バイトのデータを読み込み、実数型に変換して A,Bに代入します。
実行結果、A=5.0 , B=-13.0
イメージ仕様のXは1バイトを読み込みますが、変数へ代入しません。
SDDDのフォーマットで入力するデータを実数型に変換して Aに代入します。
X は変数を必要とせず、1 文字分が読み飛ばされます。
MZZZZ が Bに対応し、4 バイトを入力し実数に変換して Bに代入されます。

```
10  DIM A
20  ENTER #FD USING "K";A
```



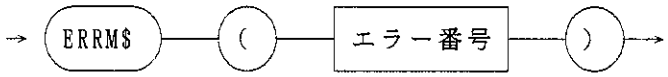
実行結果　　A=123.5

STRING123.5## を読み込んで入力変数A の実数型に変換します。
入力項目が実数型の場合、先行する数値、符号(+, -)、指数のE,e 以外の文字は無視され、数値のみを取り込みます。数値を検出してから、数値でない文字になったところで数値への変換をやめます。
イメージ仕様の K, E, k, H は、ライン・フィードがターミネータになるので、現在のファイル・ポインタからライン・フィードまでを 1つのデータとして変数に代入します。

16.　ERRM$

概　要　指定する番号のエラーメッセージを返すシステム関数です。

構　文　(1)-1



(1)-2
ERRM$(エラー番号)

解　説

・パラメータで指定されたエラー・メッセージを返します。
特にパラメータとして0を指定すると、直前に表示されたエラー・メッセージを返します。

・エラー番号は、以下のような構造になっています。

エラークラス * 256 +　エラー・メッセージ番号

エラークラス：1；　データ入出力関係
2；　データ演算処理関係
3；　ビルトイン関数関係
4；　BASIC 構文関係
5；　その他

・エラークラスを含む番号を指定しても、内部ではエラーメッセージ番号だけを参照します。したがって、エラー番号にERRNを指定できます。

17.　ERRN

概　要　エラー番号を保持しているシステム変数です。

構　文　(1)-1

(1)-2
ERRN

解　説

・BASICプログラムの実行の際に発生したエラーの番号を保持するシステム変数です。

・BASIC プログラムの開始時に 0に初期化され、エラーが発生するとその値が代入されます。この値は、明示的に 0を代入するか、BASIC プログラムを再び実行させると 0に初期化します。

・エラー番号は、以下のような構造になっています。

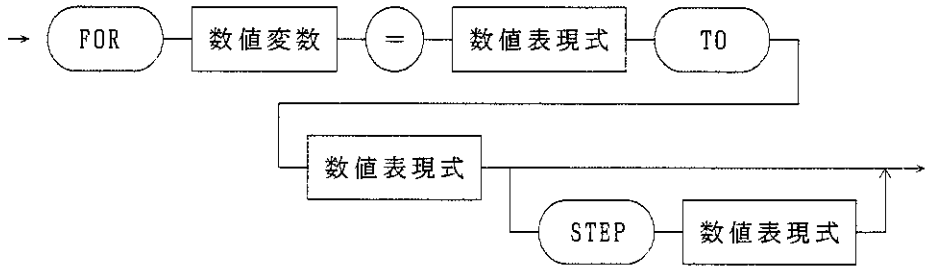
エラークラス * 256 + エラー・メッセージ番号

エラークラス : 1 ;　データ入出力関係
2 ;　データ演算処理関係
3 ;　ビルトイン関数関係
4 ;　BASIC 構文関係
5 ;　その他

18.　FOR - TO - STEP, NEXT, BREAK, CONTINUE

概　要　FOR 文とNEXT文の一対でプログラムのループ（繰り返し処理）を構成します。

構　文　(1)-1



(1)-2
FOR 数値変数＝数値表現式　TO　数値表現式〔STEP　数値表現式〕

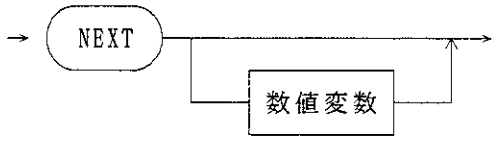
(2)-1

(2)-2
BREAK

(3)-1

(3)-2
CONTINUE

(4)-1



(4)-2
NEXT〔数値変数〕

解　説

・指定された数値変数をループ（繰り返し）のカウンタとして用い、初期値から最終値まで増加分ずつ変化させていきます。カウンタの値が最終値より大きくなったとき、ループは終了します。カウンタの増減はNEXT文で行います。したがって、FOR 文～NEXT文までの間に組まれたプログラムを繰り返し処理します。

・初期値、最終値、増加分は、以下のように指示します。
　FOR A=(初期値)　　TO (最終値)　　STEP (増加分)

・STEP(増加分)を省略した場合、自動的に+1となります。

・FOR 文～NEXT文は入れ子(Nest)にできます。

・一対のFOR 文とNEXT文で使用するループ・カウンタの数値変数名は、同じにして下さい。数値変数名が異なっているとエラーになります。

・FOR 文～NEXT文で繰り返し処理をしているときに、ループ・カウンタに使っている数値変数の値を変えると、正常な繰り返し処理をしなくなりますので注意して下さい。

・NEXT文の後の数値変数を省略した場合、自動的に直前の FOR文と対応します。

・FOR-NEXTのループは、BREAK 文で抜け出すことができます。

・CONTINUE文では、FOR-NEXTのループ内で次のステップ値のループに分岐します。

・例えば、FOR I=0 TO 10 STEP -1 のようなループの指定の場合には、ループ内の行は 1度も実行せずに終了します。

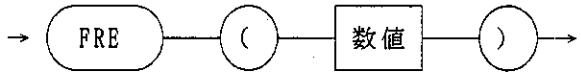
例

```
10 FOR R=11 TO 0 STEP -5
20   FOR I=0 TO PI STEP PI/180
30     X=SIN(I)*R+23
40     Y=COS(I)*R+15
50     CURSOR X,Y:PRINT "*"
60   NEXT I
70 NEXT R
80 STOP
```

19.　FRE

概　要　BASIC のメモリ残量を返すシステム関数です。

構　文　(1)-1



(1)-2
FRE(数値)

解　説

1. 数字が0 のとき

・BASIC が使用できるメモリのおおよその残りの容量を、バイト数で返します。

・これはおおよその容量を判断するためのもので、厳密にメモリ内の再構成はしません。したがって、一度セーブしてロードし直すと、より大きな容量を得ることがあります。

2. 数字が1 のとき

・ビルトイン関数の使用するメモリ領域のおよその残量を、バイト数で返します。

3. その他

・0 を返します。

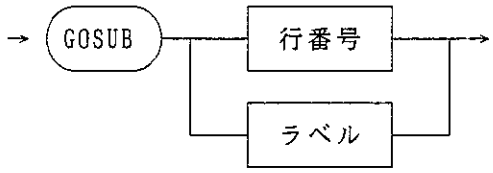
例

```
PRINT FRE(0)
```

20.　GOSUB, RETURN

概　要　指定されたサブルーチンへの分岐、復帰を行います。

構　文　(1)-1



(1)-2
GOSUB　〈行番号｜ラベル〉

(2)-1

(2)-2
RETURN

解　説

・指示された行番号のサブルーチンへ処理の制御を移し、RETURN文でGOSUB文の次の文へ戻ります。

・サブルーチンの最後には必ずRETURN文を入れて、処理をメイン・プログラムへ戻して下さい。

・サブルーチンの分岐をせずにRETURN文を実行するとエラーになります。

・GOSUB 文～RETURN文は入れ子(Nest)にできるので、サブルーチンの中から別のサブルーチンへ分岐できます。ただし、あまり入れ子を大きくするとメモリ容量がなくなり、エラーになることがあります。

・GOTO、GOSUB などで指定された行番号、ラベルがプログラム中に存在しない場合は、プログラムが実行されません。
RUN したときにUndefined LABEL と表示され、プログラムは1 行も実行されずにエラーで止まります。

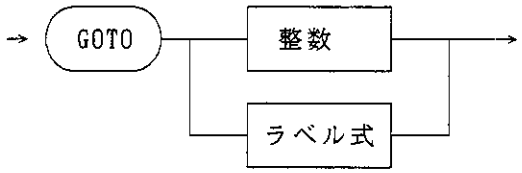
例

```
10   FOR I=1 TO 9
20     GOSUB 60
30     GOSUB *PRT
40   NEXT I
50   STOP
60   ! SUB ROUTINE
70   X = I * I
80   RETURN
90   *PRT ! SUB ROUTINE
100  PRINT I;"*";I;"=";X
110  RETURN
```

21. GOTO

概　要　指定された行番号へ分岐します。

構　文　(1)-1



(1)-2
GOTO <整数 | ラベル式>

解　説
・指定された行番号へ無条件の分岐をします。

・GOTO、GOSUB などで指定された行番号、ラベルがプログラム中に存在しない場合は、プログラムが実行されません。
RUN したときにUndefined LABEL と表示され、プログラムは1 行も実行されずにエラーで止まります。

例

```
10  FOR I=1 TO 9
20    GOTO 60
30    GOTO *PRT
40  NEXT I
50  STOP
60  !
70  X = I * I
80  GOTO 30
90  *PRT
100 PRINT I;"*";I;"=";X
110 GOTO 40
```

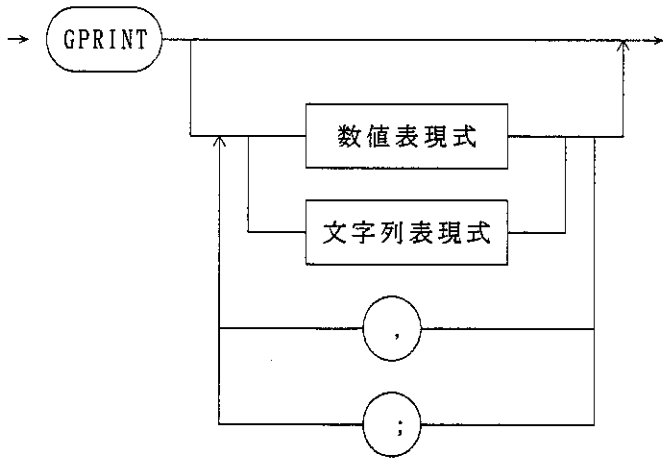
22.　GPRINT, LPRINT

概　要

数値または文字列を出力します。
GPRINT :　GPIB 出力
LPRINT :　ｼﾘｱﾙ 出力

構　文

(1)-1



(1)-2
GPRINT〔<数値表現式｜文字列表現式>｛,｜;<数値表現式｜文字列表現式>｝

(2)
LPRINTは GPRINT と同様です。

解　説

・GPRINTまたはLPRINTで指定された数値、文字列を表示します。

・数値、文字列をカンマ(,) で区切って複数を指定すると、改行せずに数値、文字列を次々に出力します。

・GPRINTまたはLPRINTの最後に、セミコロン(;) を置いた場合は、プリント出力が終っても改行されません。
したがって、次のGPRINTまたはLPRINTを実行すると、以前にプリントした行に続いてプリントします。

・GPIBプリンタに出力するためにGPRINTを使用する場合には、必ず本器のパネル操作でSYSTEM CONTROLLER にし、プリンタのアドレスを設定して下さい。

例

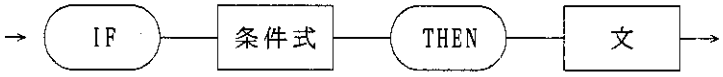
```
100 PRINTER 1
110 FOR I=0 TO 20
120   GPRINT I
130   LPRINT I
140 NEXT I
150 STOP
```

23.　IF-THEN, ELSE, END IF

概　要　条件判断による分岐、指定された文の実行をします。

構　文

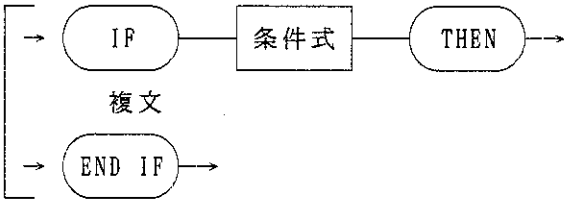
(1)-1



(1)-2

```
IF　条件式　THEN　文
```

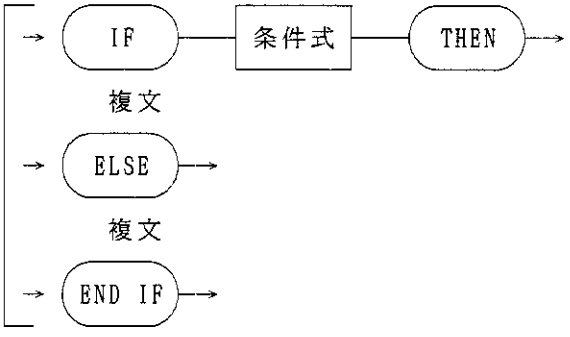
(2)-1



(2)-2

```
IF　条件式　THEN
  複文
END IF
```

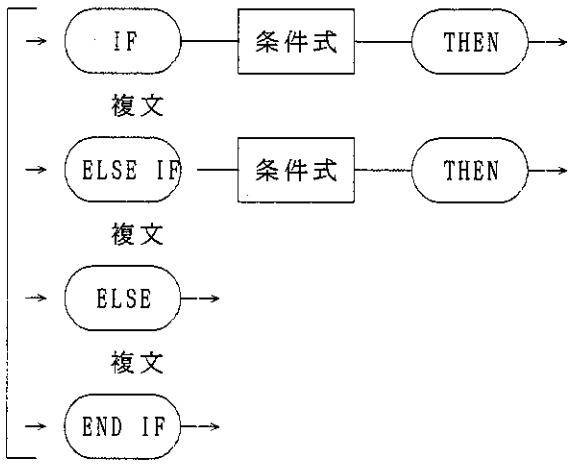
(3)-1



(3)-2

```
IF　条件式　THEN
  複文
ELSE
  複文
END IF
```

(4)-1



(4)-2

```
IF  条件式  THEN
  複文
ELSE IF 条件式  THEN
  複文
ELSE
  複文
END IF
```

解　説

・条件式は論理式ですが、ここには比較演算子を用いた論理式以外に数値表現式を書くこともできます。この場合、演算結果が 0になったときのみ(false)とし、それ以外の値は全て真(true)と解釈します。

・論理式の条件によってプログラムの分岐、処理などを行います。

・論理式の関係が成立すると、THEN文を実行します。THEN文には文を続けることができ、次文を実行します。

・論理式の関係が不成立の場合は、そのまま次の行に進みます。

・比較演算子には、以下の 6種類があります。

A=B	AとBが等しいとき成立
A>B	AがBより大きいとき成立
A<B	AがBより小さいとき成立
A>=B	AがBと等しいか大きいとき成立
A<=B	AがBと等しいか小さいとき成立
A<>B	AとBが等しくないとき成立

上の論理式でA, Bはともに数値表現式で構成できます。
ただし、数値表現式と文字列表現式を比較することもできます。

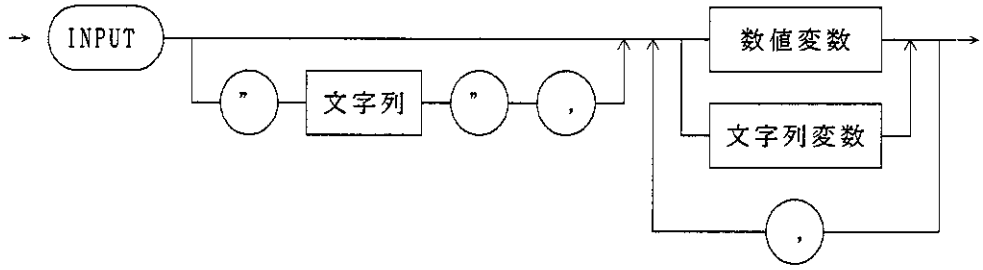
例

```
10 FLG = 0
20 FOR I=0 TO 10
30  PRINT I;
40  IF (I % 2) =0 THEN FLG = 1
50  IF FLG = 1 THEN
60                    PRINT "  EVEN";
70                    FLG = 0
80                    END IF
90  PRINT
100 NEXT I
110 STOP
```

24.　INPUT

概　要　キー入力したデータを数値変数に代入します。

構　文　(1)-1



(1)-2
INPUT〔“文字列”,〕<数値変数｜文字列変数>{,<数値変数｜文字列変数>}

解　説

・INPUT を実行すると、プログラムは一時停止して、キー入力待ちとなります。キー入力待ちは ENTERキーが押されるまで続きます。データをキー入力後ENTER キーを押すと、データが変数に代入されます。

・INPUT では、数値変数、文字列変数のいずれも扱えるようになっていますが、数値変数を入力しようとしているときに数字以外の文字（英文字、英記号など）を入力させると数字以外の文字は無視し、もし数字が一字もないときは 0が変数に入力されます。また、ENTER キーのみが押されたときには変数への代入は行いません。つまり、INPUT 前の値がそのまま残ります。

・文字定数を入力するときは、引用符で囲む必要はありません。

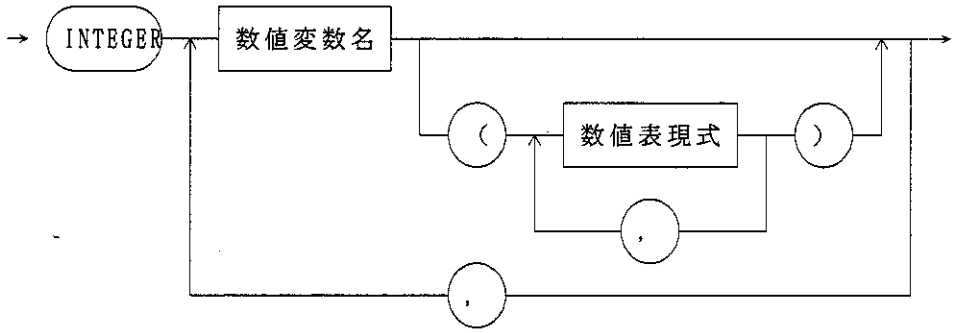
例

```
10  OUTPUT 31;"OLDC OFF"
20  OUTPUT 31;"INIT:CONT OFF"
30  ENABLE INTR
40  INPUT  "CENTER FREQUENCY(MHz)?",CF
50  OUTPUT 31;"FREQ:CENT ",CF,"MHz"
60  OUTPUT 31;"FREQ:SPAN ",SF,"KHz"
70  OUTPUT 31;"INIT"
80  PRINT  "MAX =",MAX(0,1200,0)
90  STOP
```

25.　INTEGER

概　要　変数または配列変数が整数型であることを宣言します。

構　文　(1)-1



(1)-2

```
INTEGER  A〔B 〕 {, A〔B 〕}
```

A : 数値変数名
B : (数値表現式 {, 数値表現式})

解　説

・INTEGER 文で、数値変数、配列変数を指定すると、以後その変数は、整数型となります。

・整数型変数で扱える数値は、整定数で扱える範囲と同じです。
-2147483648～+2147483647

・整数しか扱わない変数では、INTEGER 文で宣言した方が処理時間が短くなります。

・INTEGER 命令で配列宣言すると、指定された大きさの配列変数をメモリ上に確保します。したがって、大きすぎる配列宣言をするとメモリ領域が足りなくなり、エラーとしてプログラムの実行を中止します。
(memory space full)

・添字を複数個指定すると個数分の次元を持つ配列変数の指定となります。
(次元数は、メモリ容量が許す限り)

例

```
10  INTEGER ARRAY(2,3)
20  PRINT "J/I";
30  PRINT USING "X,3D,3D,3D";1,2,3
40  PRINT " ";
50  FOR I = 1 TO 2
60    FOR J = 1 TO 3
70        ARRAY(I,J) = I*10 + J
80    NEXT J
90  NEXT I
100 FOR I = 1 TO 2
110 PRINT
120 PRINT USING "2D,2X,#";I
130   FOR J = 1 TO 3
140       PRINT USING "3D,#";ARRAY(I,J)
150   NEXT J
160 NEXT I
```

<実行結果>

```
J/I   1  2  3

 1   11 12 13
 2   21 22 23
```

注意

1. INTEGER 文で一度整数型に指定された変数はDEL やコメント文で命令を削除しても、整数型のままです。

2. 実数型に再指定したい場合は、DIM 命令を追加するか、SAVE/LOAD を一度行ってからRUN させます。

26.　INTERFACE CLEAR

概　要　本器に接続されているすべてのGPIBインタフェースを初期化します。

構　文　(1)-1

(1)-2
INTERFACE CLEAR

解　説　・INTERFACE CLEAR を実行すると、GPIBの単線信号IFC を約100μs の間出力します。
本器のGPIBに接続されている装置のすべてのGPIBインタフェースは、IFC信号を受け取ると、トーカまたはリスナの状態が解除されます。

例　10 INTERFACE CLEAR

注　意　ADDRESSABLE モードでは機能しません。

27.　KEY$

概　要　パネル・キーのコードを返します。

構　文　(1)-1

(1)-2
KEY$

解　説　・本器のパネル・キーのコードで、一番最後に押されたコードを返します。
一度参照すると、この変数の内容はクリアされます。

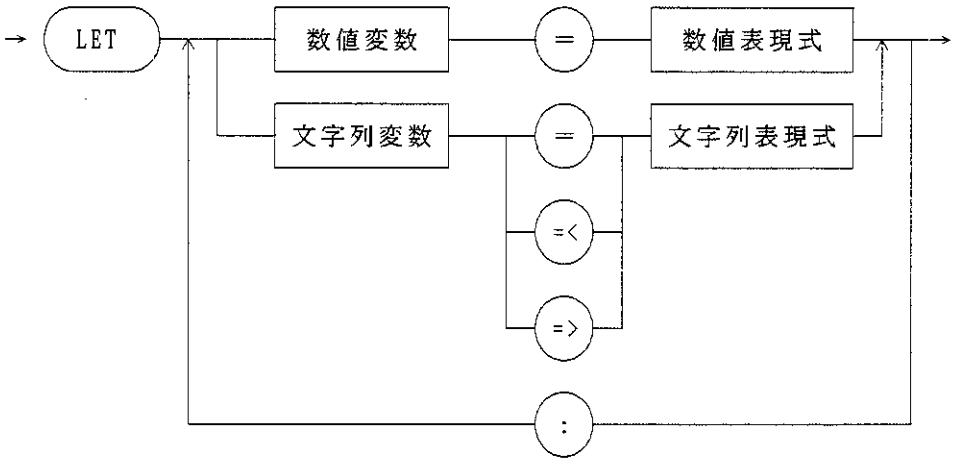
例

```
10   A$=KEY$
20   IF A$="1" THEN
30     GOSUB *TEST1
40   ELSE IF A$="2" THEN
50     GOSUB *TEST2
60   END IF
70   GOTO 10
80   STOP
100  *TEST1
110    PRINT "Check1 Start !!"
120    .....
130    RETURN
200  *TEST2
210    PRINT "Check2 Start !!"
220    .....
230    RETURN
```

28.　LET

概　要　（プログラム上ではLET は使用せず、直接代入文を記述します。）
変数に代入を行います。

構　文　(1)-1



(1)-2
LET　<A | B> {: <A　|　B> }

A : 数値変数=数値表現式
B : 文字列変数 = | =< | =>文字列表現式

解　説
・ここで用いる等号（=)は、代入を意味するもので、数学的な等号とは意味が異なります。

・等号の左辺が数値ならば文字列も数値の部分を変換して代入します。
特に文字列を代入する場合
= のとき: 高々右辺の長さ分だけ代入されます。
=>のとき: 左辺の文字列に比べ右辺の文字列が短いと、頭にスペースをつめて左辺に長さ分だけ代入します。
=<のとき: 後ろにスペースをつめます。
したがって、=>と =< は文字列にのみ有効な代入演算子です。

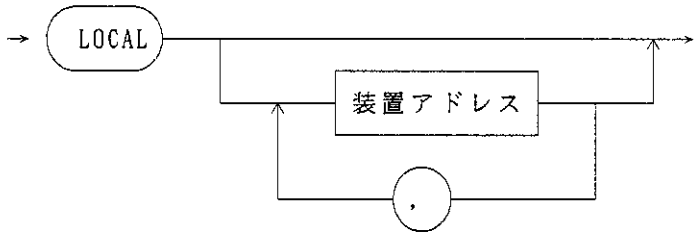
例

```
10  DIM STR$                                  <実行結果>
20  PRINT "123456789012345678"                123456789012345678
30  STR$ = "ABC":PRINT STR$                   ABC
40  STR$ =< "OPQ":PRINT STR$                  OPQ
50  STR$ => "XYZ":PRINT STR$                                  XYZ
```

29.　LOCAL

概　要　指定した装置をリモート状態から解除するか、またはリモート・イネーブル（REN)ラインを偽にします。

構　文　(1)-1



(1)-2
LOCAL 〔装置アドレス｛，装置アドレス｝〕

解　説
・装置アドレスを指定せずにLOCAL だけを実行した場合、GPIBリモート・イネーブル（REN)ラインが偽（High level）となり、GPIB上のすべての装置がローカル状態となります。
RENが偽のときは、OUTPUT命令でのGPIB機器の設定はできなくなる(GPIBでコントロールできない）ので注意が必要です。

・再び RENを真（Low level)にするためには、REMOTEを実行して下さい。

・LOCAL に続いて装置アドレスを指定した場合、装置アドレスで指定された装置のみをアドレスし、リモート状態を解除します。

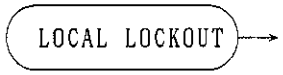
例
```
10 LOCAL
20 LOCAL 1
30 LOCAL 1, 2, 3
```

注　意　ADDRESSABLE モードでは機能しません。

30. LOCAL LOCKOUT

概　要　GPIBに接続されている装置を、パネル面からローカル状態にする機能を禁止します。

構　文　(1)-1



(1)-2
LOCAL LOCKOUT

解　説

・GPIB上の各装置がリモート状態（GPIBによってリモート・コントロールされている）のときは、各装置のパネル・キーは LOCALキーを除きロックされ、パネルからデータ設定ができません。

リモート状態のときLOCAL キーを押すと、各装置は自分自身をローカル状態にするので、データ設定が可能な状態になります。このため、リモート制御中に種々の障害が生じ、正確なコントロールができなくなります。

この場合にLOCAL LOCKOUT を実行すると、GPIB上の全装置のローカル・キーをロックして、完全に各装置のパネル面からの設定を禁止します。

・LOCAL LOCKOUT を実行すると、GPIBにユニバーサル・コマンドのローカル・ロックアウト (LLO)を送ります。

・ローカル・ロックアウト状態を解除するには、LOCAL コマンドを用いてRENラインを偽(High level)にして下さい。

例

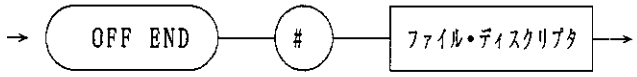
```
10 LOCAL LOCKOUT
```

注　意　ADDRESSABLE モードでは機能しません。

31.　OFF END

概　要　ON END文で指定したエンド・オブ・ファイル時の処理を解除します。

構　文　(1)-1



(1)-2
OFF END　#ファイル・ディスクリプタ

解　説　・ファイル・ディスクリプタに定義してあった分岐先を解除した後に、エンド・オブ・ファイルが起こった場合、以下のエラー・メッセージを表示して終了します。

```
end of "DATAFILE" file
```

※　ファイルの取り扱いについては、「1.　はじめに　●ファイルの管理」を参照して下さい。

32.　OFF ERROR

概　要　エラーが発生したときの分岐の機能、定義を解除します。

構　文　(1)-1

(1)-2

```
OFF ERROR
```

解　説　・ON ERRORステートメントによって定義されたエラー分岐を禁止します。

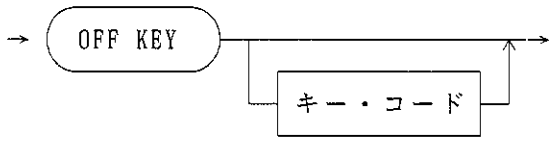
例

```
10  ON ERROR GOTO 100
      :
100  OFF ERROR
110  PRINT "Error Code",ERRN
120  STOP
```

33.　OFF KEY

概　要　KEY 入力の割り込みによる分岐の機能、定義を解除します。

構　文　(1)-1



(1)-2
OFF KEY 〔キー・コード〕

解　説
・ON KEYステートメントによって許可された本器の KEY入力割り込みによる分岐を禁止します。

例

```
10   ON KEY 2 GOTO 100
20   ENABLE INTR
30   ! LOOP
40   GOTO 30
100  OFF KEY
110  PRINT "OFF KEY"
120  STOP
```

34. OFF SRQ, OFF ISRQ

概　要

SRQ またはISRQの割り込みによる分岐の機能、定義を解除します。
(OFF SRQはコントローラ・モード時のみ有効)

構　文

(1)-1

(1)-2
OFF SRQ

(2)
OFF ISRQはOFF SRQ と同様です。

解　説

・OFF SRQ
ON SRQによって許可された割り込みによる分岐を禁止します。

・OFF ISRQ
ON ISRQ によって許可された割り込みによる分岐を禁止します。

例

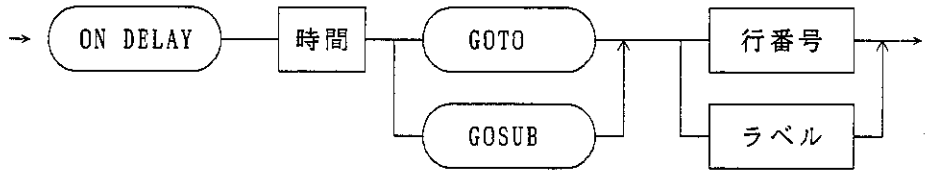
```
100 OUTPUT 31;"OLDC OFF"
110 OUTPUT 31;"STAT:OPER:ENAB 8;*SRE 128":SPOLL(31)
120 ON ISRQ GOTO *MAX
130 OUTPUT 31;"INIT:CONT OFF;:ABOR;:INIT"
140 ENABLE INTR
150 ! LOOP
160 GOTO 140
170 *MAX
180 DISABLE INTR
190 OFF ISRQ
200 PRINT MAX(0,1200,0)
210 STOP
```

ｱﾄﾞﾚｽ	内容
100	SRQ をENABLE
110	内部SRQ の割り込み分岐を設定
120	シングル掃引
130	割り込み受け付け
170	割り込み禁止
180	内部SRQ の割り込み分岐を解除
190	最大レベルを表示

35.　ON DELAY

概　要　指定時間経過後に分岐します。

構　文　(1)-1



(1)-2

ON DELAY　時間　<GOTO | GOSUB> <行番号 | ラベル>

注）時間の単位はmsecで、設定範囲は 0～65535 です。

解　説
- 指定時間経過後、その後のステートメントに従って分岐します。
- ENABLE INTR で割り込みの受け付けを許可しておく必要があります。

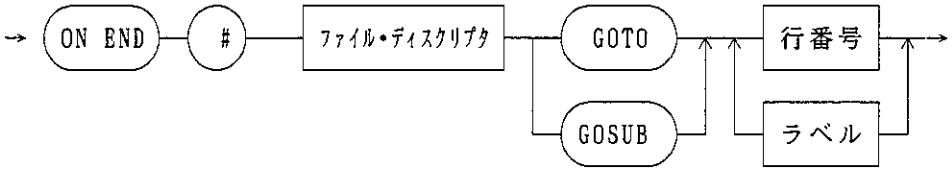
例

```
10  INTEGER T
20  T=50
30  ENABLE INTR
40  ON DELAY T GOSUB *TEST
50  STOP
100 *TEST
110   PRINT T;"[msec] Delay"
120 RETURN
```

36.　ON END

概　要　エンド・オブ・ファイル時の処理（分岐先）を定義します。

構　文　(1)-1



(1)-2
ON END #ﾌｧｲﾙ･ﾃﾞｨｽｸﾘﾌﾟﾀ <GOTO | GOSUB> <行番号 | ラベル>

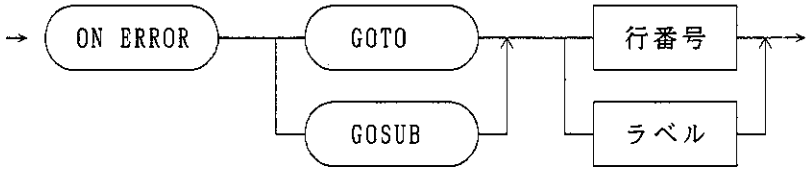
解　説
・ENTER でファイルからデータを読み込みますが、ファイルの終わりまで読み込んで入力するデータがない場合に、エンド・オブ・ファイルになります。
ON END文で処理を宣言しておかないと、ファイルをクローズした後に、エラー・メッセージを表示して実行を停止します。

※　ファイルの取り扱いについては、「1.　はじめに　●ファイルの管理」を参照して下さい。

37.　ON ERROR

概　要　エラーが発生したときの分岐を許可します。

構　文　(1)-1



(1)-2
ON ERROR <GOTO | GOSUB> <行番号 | ラベル>

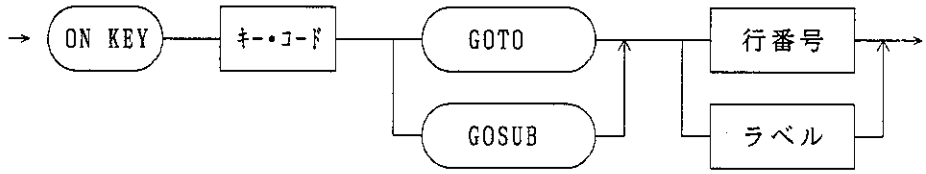
解　説
- BASICプログラムの実行中にエラーが発生すると、そこの文番号とエラー・メッセージを表示してプログラムを停止します。
  特に、計測器のサービスを要求するビルトイン関数のエラーの際には、エラー・メッセージを表示するだけで実行し続けます。これらを検出して分岐する場合には、ON ERROR文を使用します。
- 発生したエラーを分類するために、エラー番号を記憶したERRNシステム変数が用意されています。
- エラーが発生した後に、そのエラー処理で確実に回復できないと永久ループになってしまいます。これを防ぐには、OFF ERROR文を入れます。

例
```
ON ERROR GOTO 1000
```

38. ON KEY

概　要　KEY 入力の割り込みによる分岐を許可します。

構　文　(1)-1



(1)-2

ON KEY　ｷｰｺｰﾄﾞ　<GOTO | GOSUB> <行番号 | ラベル>

解　説

・プログラム実行中にKEY入力の割り込みで分岐します。

・分岐は、割り込みが発生したときに、実行していたステートメントの処理が終了してから行われます。

・サブルーチンへ分岐した場合の戻り先は、割り込みが発生したときに実行していたステートメントの次のステートメントとなります。

・キー・コードは1 ～6 までの数値で、正面パネル上のファンクション・キーとキー・ボード上のF1～F6に対応しています。また、本器にキー・ボードを接続した場合には、キー・ボードのF1～F6に対応します。

・ENABLE INTR で割り込みの受け付けを許可しておく必要があります。

例

```
10    CLS
20    ON KEY 1 GOTO 1000
30    ON KEY 2 GOTO 1100
40    ON KEY 3 GOTO 1200
50    ON KEY 4 GOTO 1300
60    ON KEY 5 GOTO 1400
70    ON KEY 6 GOTO 1500
75    CNT = 10
80    *HERE:
85    I = 0: PRINT " "
90    IF I=CNT THEN GOTO *HERE
100   ++I: PRINT ">";
110   ENABLE INTR
120   GOTO 90
1000  PRINT "FIRST KEY"
1001  CNT = 1
1010  GOTO *HERE
1100  PRINT "SECOND KEY"
1101  CNT = 10
1110  GOTO *HERE
1200  PRINT "THIRD KEY"
1201  CNT = 20
1210  GOTO *HERE
1300  PRINT "FOURTH KEY"
1301  CNT = 30
1310  GOTO *HERE
1400  PRINT "FIFTH KEY"
1401  CNT = 40
1410  GOTO *HERE
1500  PRINT "SIXTH KEY"
1501  CNT = 50
1510  GOTO *HERE
```

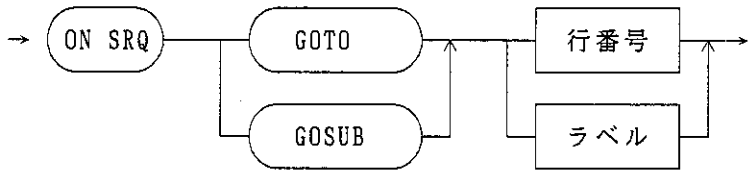
39.　ON SRQ, ON ISRQ

概　要

ON SRQは、GPIB外部SRQ 信号による割り込み分岐を許可します。
（ON SRQコントローラ・モード時のみ有効）
ON ISRQ は、内部割り込み要因が発生したときの割り込み分岐を許可します。

構　文

(1)-1



(1)-2
ON SRQ <GOTO | GOSUB> <行番号 | ラベル>

(2)
ON ISRQ はON SRQと同様です。

解　説

・プログラム実行中の割り込みで分岐します。

・分岐は割り込みが発生したときに実行していたステートメントの処理が終了してから行われます。

・サブルーチンへ分岐した場合の戻り先は、割り込みが発生したときに実行していたステートメントの次のステートメントになります。

・ON SRQは、コントローラ・モードで実行時のみGPIB外部からのSRQ 信号で割り込み分岐します。

・ENABLE INTR で割り込みの受け付けを許可しておく必要があります。

例　シングル掃引ごとに、MAX をサーチするプログラム

```
100 OUTPUT 31;"OLDC OFF"
110 ON ISRQ GOTO *MAX
120 OUTPUT 31;"STAT:OPER:ENAB 8;*SRE 128":SPOLL(31)
130 ENABLE INTR
135 OUTPUT 31;"INIT:CONT OFF;:ABOR;:INIT"
140 ! LOOP
150 GOTO 140
160 *MAX
170 DISABLE INTR:SPOLL(31)
180 PRINT MAX(0,1200,0)
190 GOTO 130
```

ｱﾄﾞﾚｽ	内容
110	内部SRQ の割り込み分岐を設定
120	SRQ をENABLE
130	割り込み受け付け
135	シングル掃引
170	割り込み禁止
180	最大レベルを表示

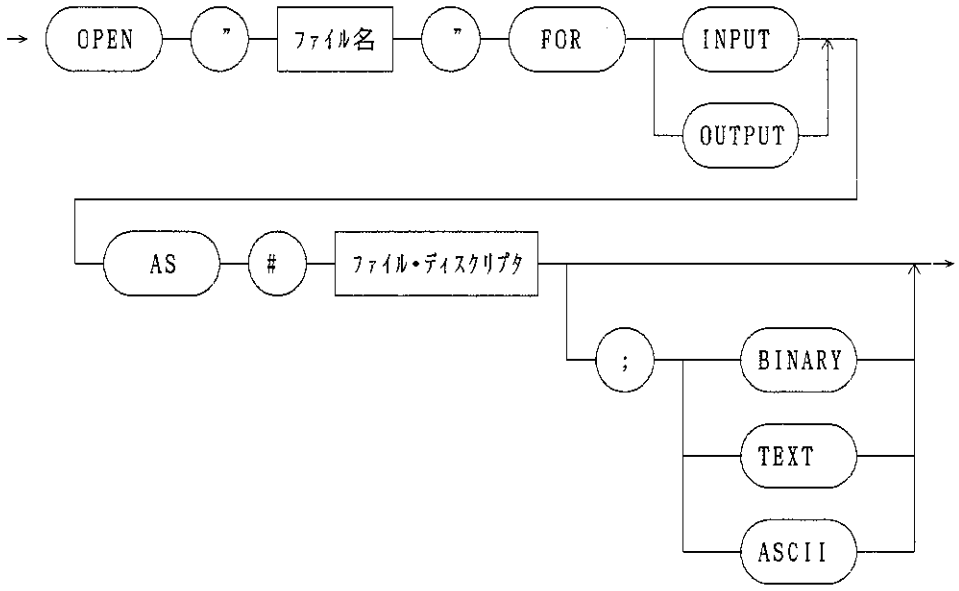
40. OPEN

概　要

ファイルに対しファイル・ディスクリプタを割り当て、指定した処理モードでオープンします。

構　文

(1)-1



(1)-2

OPEN "ファイル名" FOR 処理モード AS #ファイル・ディスクリプタ 〔; ファイル・タイプ〕

注）処理モード ： INPUT | OUTPUT
　　ファイル・タイプ　： BINARY | TEXT | ASCII

解　説

・ファイルをプログラムに認識させるために、ファイルに対してファイル・ディスクリプタを割り当て、指定した処理モードでオープンします。

処理モード
処理モードには、OUTPUTとINPUT があります。
OUTPUT ：　ファイルにデータを書き込むときに使います。
INPUT　：　ファイルからデータを読み込むときに使います。

#ファイル・ディスクリプタ
実際のファイルに対する読み書きは、ENTER またはOUTPUTを使いますが、これらのコマンドに対して、対象となるファイルを認識させるために、ファイル・ディスクリプタを使います。
ファイル・ディスクリプタ名は #の後に英数字で記述します。

ファイル・タイプ
ファイル・タイプには、BINARY、TEXT、ASCII の 3種類あります。
ファイル・タイプを指定しないとBINARYになります。

BINARY :　データを内部の表現のまま記録します。整数のときは 4バイト、実数のときは 8バイト、文字列はヘッダ 4バイトの後にASCIIデータが続きます。データ文字数が奇数の場合はデータの後に 1 バイトのスペースをとります。

TEXT　:　データをそのまま ASCIIコードに変換して出力しますが、数値の前に－かスペースをとります。
TEXTファイルでは USING指定ができます。

ASCII :　入力、出力項目を 2バイトのヘッダの後に ASCIIで表現します。数値の前に－かスペースをとります。データ文字数が奇数の場合はデータの後に 1バイトのスペースをとります。

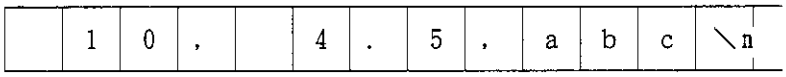
・既に他のファイルに割り当てられているファイル・ディスクリプタをオープンすると、前に割り当てられていたファイルをクローズして、指定されたファイルを新しくオープンします。

・同じファイルを同時点で複数のファイル・ディスクリプタでオープンすることはできません。

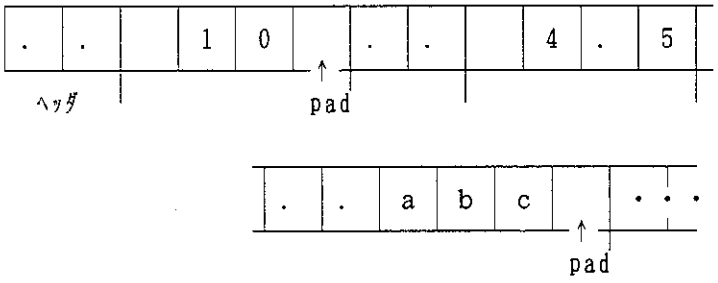
※　ファイルの取り扱いについては、「1.　はじめに　●ファイルの管理」を参照して下さい。

例

```
10 OPEN "DATA.BAS" FOR OUTPUT AS #FD ; TEXT
20 OUTPUT #FD;10,4.5,"abc"
```



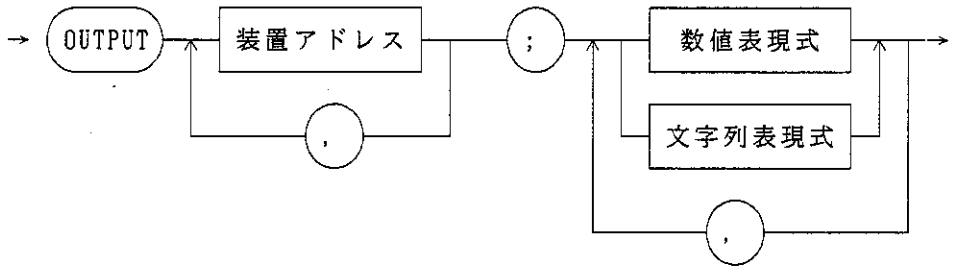

```
10 OPEN "DATA.BAS" FOR OUTPUT AS #FD ; ASCII
20 OUTPUT #FD;10,4.5,"abc"
```



41.　OUTPUT

概　要　(1)GPIBやパラレル・ポートへデータを送出します。
(2)ファイルにデータを出力（書き込み）します。

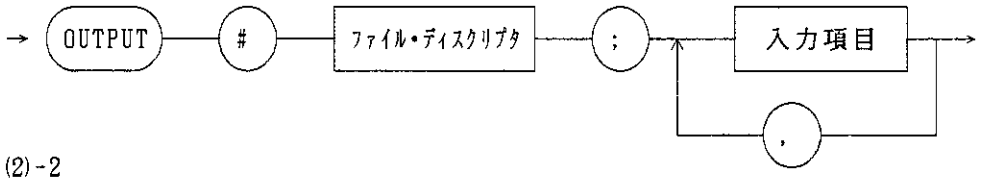
構　文　(1)-1



(1)-2
OUTPUT　装置アドレス｛，装置アドレス｝；<数値表現式｜文字列表現式>
｛，<数値表現式｜文字列表現式>｝

注）装置アドレス：0～30；外部GPIB接続機器のアドレス
31　；本器の測定部への出力
33　；パラレル・ポートのA ポートへの出力
34　；パラレル・ポートのB ポートへの出力
35　；パラレル・ポートのC ポートへの出力および
Flip/Flop のセット/ リセット
36　；パラレル・ポートのD ポートへの出力および
ポートのモード設定
37　；パラレル・ポートのCDポートへの出力
複数の装置アドレスが指定できるのは、 0～30のときだけです。

(2)-1



(2)-2
OUTPUT # ﾌｧｲﾙ･ﾃﾞｨｽｸﾘﾌﾟﾀ ; 入力項目｛,入力項目｝

解　説　(1)の構文
・装置アドレスによって指定された装置へ、数値および文字列をASCII データとして送ります。
装置アドレスは、カンマ（，）で区切って複数を指定できます。
また、数値表現式と文字列表現式もカンマで区切ると、混在して使えます。

・REN ラインが真（Low level)のときにOUTPUTステートメントを実行すると、装置アドレスで指定された装置は、自動的にリモート状態になります。リモート状態をプログラムで解除するときは、LOCAL ステートメントを実行して下さい。

・例
```
10 A=5
20 B=10
30 OUTPUT A;"STARTF",B,"MHz"
```

・注意
SYSYEM CONTROLLER モード時は、指定アドレスの機器をリスナに指定し、データを出力します。

外部に指定したリスナがない場合、このコマンドは実行しません。

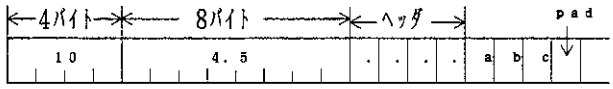
⑵の構文
・ファイル・ディスクリプタに割り当てられているファイルに、出力項目をBASIC の標準の書式に変換してから出力します。
このデータタイプの形式で読み込んで、その入力項目に代入します。

・例1)　BINARYファイル

データを内部表現と同じ型で出力します。文字列は 4バイトの文字列の長さを示すヘッダを付けて出力します。文字列が奇数の長さである場合は、最後に 1文字分のスペースをとります。

```
10 OPEN "FILE" FOR OUTPUT AS #FD
20 OUTPUT #FD;10,4.5,"abc"
```

※　ファイルの取り扱いについては、「1.　はじめに　●ファイルの管理」を参照して下さい。

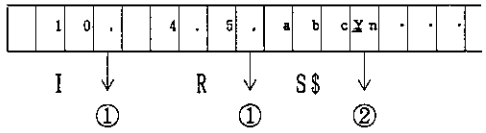


ヘッダはデータの長さを持っています。

・例2)　TEXTファイル

データをASCII コードに変換して出力します。
数値データには、スペースかマイナスの符号が頭に付きます。

```
10 OPEN "FILE" FOR  OUTUT AS #FD;TEXT
20 OUTPUT #FD;10,4.5,"abc"
```



①：　各項目はカンマ(,) で区切られます。
②：　最後の項目の後にはライン・フィードが出力されます。

・例3)　ASCII ファイル

データをASCII コードに変換して出力します。
数値データには、スペースかマイナスの符号が頭に付きます。データのバイト数が奇数の場合は、最後にスペースが入ります。

```
10 OPEN "FILE" FOR INPUT #FD;ASCII
20 OUTPUT #FD;10,4.5,"abc"
```

ヘッダはデータの長さを持ちます。

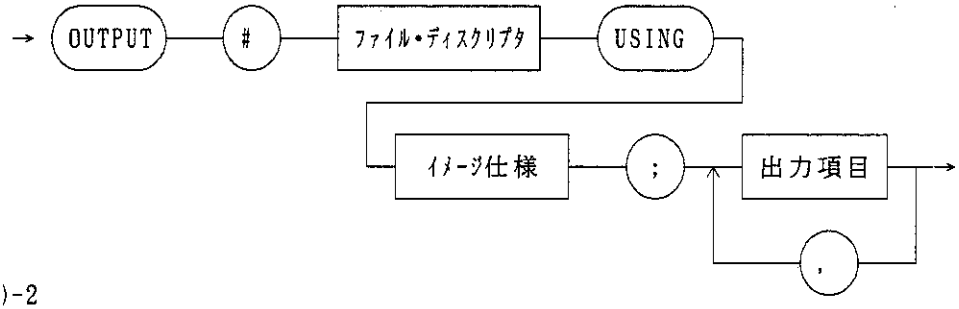
42. OUTPUT USING

概　要

#ファイル・ディスクリプタに割り当てられているファイルにデータを指定された形式で出力（書き込み）します。TEXTファイルのみ有効です。

構　文

(1)-1



(1)-2

OUTPUT　#ﾌｧｲﾙ･ﾃﾞｨｽｸﾘﾌﾟﾀ　USING　イメージ仕様；出力項目 {, 出力項目}

注）OUTPUTは OUT、USING はUSE と省略できます。

解　説

・USING とイメージ仕様を指定すると、自由に書式を変換して出力します。イメージ仕様は、文字列式で指定します。

・TEXTファイルとしてOPENされた場合のみ有効です。

・ファイル・ディスクリプタは、フィイル・オープン時に指定したものを使います。オープン時に、処理の対象になるファイルに対して、ファイル・ディスクリプタを割り当てます。以後、そのファイルに対する処理はすべてこのファイル・ディスクリプタを介して行います。

イメージ仕様

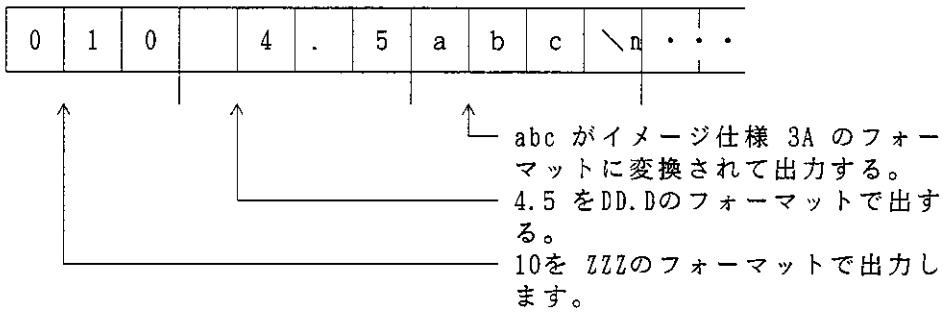
D	：	D の数で数値を出力するときの桁数を指定する。 指定したフィールドで空いた部分にはスペースが入る。
Z	：	Z の数で数値を出力するときの桁数を指定する。 指定したフィールドで空いた部分には 0が入る。
K	：	式の値を BASICの標準形式（PRINTと同じ）で出力する。
S	：	S の位置にプラス(+) かマイナス(-) を出力する。
M	：	M の位置に、負のときはマイナス(-) を、正ならばスペースを出力する。
.	：	. の位置に小数点がくるように位置を合わせる。
E	：	e 符号　指数　という書式で出力する。
H	：	K と同じ。ただし、小数点にカンマ (,)を使う。
R	：	. と同じ。ただし、小数点にカンマ (,)を使う。
*	：	* の数で数値の出力時の桁数を指定する。 指定したフィールドで空いた部分には *を出力する。
A	：	A の部分に 1文字出力する。
k	：	文字列式の値をそのまま出力する。
ﾘﾃﾗﾙ	：	＼” で囲まれた文字列を出力項目とは無関係にそのまま出力する。

X　：X の位置に 1つスペースをとる。
B　：式の値を ASCIIコードとして出力する。
@　：フォーム・フィードを出力する。
+　：キャリージ・リターンを出力する。
-　：ライン・フィードを出力する。
#　：最後の項目の後ろには、自動的にライン・フィードが付くが、このイメージ仕様を指定するとライン・フィードが付かない。
n　：数字で各イメージ仕様の繰り返し指定の回数を指定できる。
3D. 2DはDDD.DDと同じで、4AはAAAAと同じ。

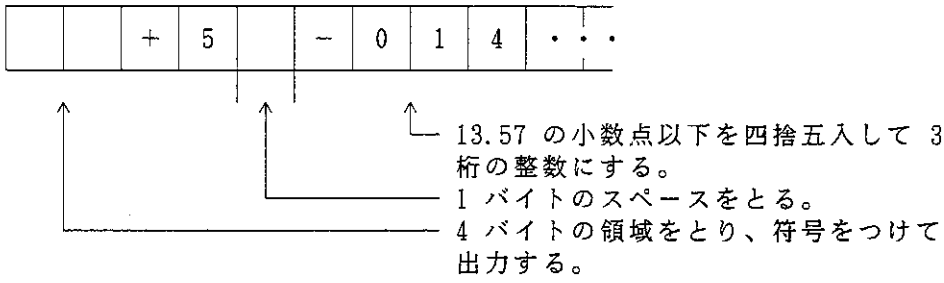
※ ファイルの取り扱いについては、「1. はじめに ●ファイルの管理」を参照して下さい。

例

```
OUTPUT #FD USING "ZZZ,DD.D,3A";10,4.5,"abc"
```



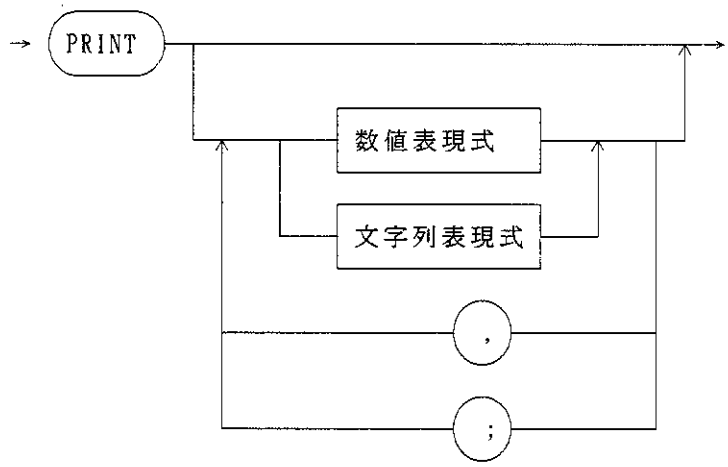

```
OUTPUT #FD USING "SDDD,X,MZZZ";+5,-13.57
```



43.　PRINT［USING］

概　要　数値または文字列を表示します。

構　文　(1)-1



(1)-2
PRINT〔数値表現式｜文字列表現式 {, | ; 数値表現式｜文字列表現式}〕

解　説
- 指定された数値、文字列を表示します。
- 数値、文字列をカンマ(,) で区切って複数を指定すると、改行せずに数値、文字列を次々に出力します。
- PRINT の最後にセミコロン(;) を置いた場合は、プリント出力が終っても改行されません。したがって、次のPRINT を実行すると、以前にプリントした行に続いてプリントします。

例

```
10 PRINT 123*456
20 PRINT "ABC"
30 PRINT "Freq.=",A,"Hz"
40 PRINT I,
```

●PRINT USING 書式指定式 ;〔〔式〔••••〕〕

書式指定式は文字列表現式で、イメージ仕様をコンマで区切って、書式を指定します。最後は自動的に改行します。

ｲﾒｰｼﾞ仕様
D　：　指定フィールドの余った部分にスペースを表示する。
Z　：　指定フィールドの余った部分に 0を表示する。
K　：　式の値をそのまま表示する。
S　：　常に＋または－のサイン・フラグを付ける。
M　：　－のサイン・フラグを付けるか、正のときはスペースを取る。
.　：　小数点を表示する。
E　：　指数形式（e，符号，指数）で表示する。
H　：　式の値をそのまま表示する。ただし、小数点にカンマ(,) を使う。
R　：　ヨーロッパ・タイプの小数点を表示する（小数点にカンマ(,) を使う）。
*　：　指定フィールドの余った部分に *を表示する。
A　：　1 文字を表示する。
k　：　式の文字列をそのまま表示する。
X　：　スペースを表示する。
ﾘﾃﾗﾙ：　書式指定式にリテラルを書くときは＼" で囲む。
B　：　式の値をASCII コードとして表示する。
@　：　改ページする。（フォーム・フィード）
+　：　表示の位置を同じ行の先頭に移動させる。（キャリッジ・リターン）
-　：　表示の位置を次の行に移動させる。（ライン・フィールド）
#　：　最後に改行されない。
n　：　数字で各イメージ仕様の繰り返し回数を指定できる。
　　3D.2D はDDD.DDと同じ、4AはAAAAと同じ。

例 1

```
10  PRINT USING "4Z,2X,5D,2X,5*";123,-444,567
```

<実行結果>
```
 0123    -444  **567
```

例 2

```
10  PRINT USING "S3D,X,S3D";-4.5,465
20  PRINT USING "M3Z.Z,X,M3ZR3Z";1.26,-5.452
```

<実行結果>
```
  -5 +465
 001.3 -005,452
```

例 3

```
10  PRINT USING "K,X,H";5.03884e+22,4.5563
```

<実行結果>
```
 5.03884e+22 4,5563
```

例 4

```
10  PRINT USING "k,#";"character:"
20  PRINT USING "B";69
```

<実行結果>
```
 character:E
```

例 5

```
10  PRINT USING "\".....\",+,A";"*"
20  PRINT USING "k,-,\".END.\"";"string"
```

<実行結果>

```
*.....
string
.END.
```

例 6

```
100  PRINT USING "DDD.DD";1.2
110  PRINT USING "ZZZ.ZZ";1.2
120  PRINT USING "K";1.2
130  PRINT USING "SDDD.DD";1.2
140  PRINT USING "MDDD.DD";1.2
150  PRINT USING "MDDD.DD";-1.2
160  PRINT USING "H";1.2
170  PRINT USING "DDDRDD";1.2
180  PRINT USING "***.**";1.2
190  PRINT USING "A";"a"
200  PRINT USING "k";"string"
210  PRINT USING "B";42
220  PRINT USING "3D.2D";1.2
```

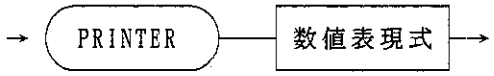
<実行結果>

```
  1.20
001.20
1.2
  +1.20
   1.20
  -1.20
1,2
  1,20
**1.20
a
string
*
  1.20
```

44. PRINTER

概　要　プリンタに送る装置アドレスを指定します。

構　文　(1)-1



(1)-2
PRINTER 数値表現式

解　説
- GPIBに接続されるプリンタの装置アドレスを設定します。
- GPRINT、GLIST 、GLISTNを実行する前に、必ずPRINTER でプリンタの装置アドレスを本器に指示して下さい。
- 装置アドレスは、0 ～30までの整数です。

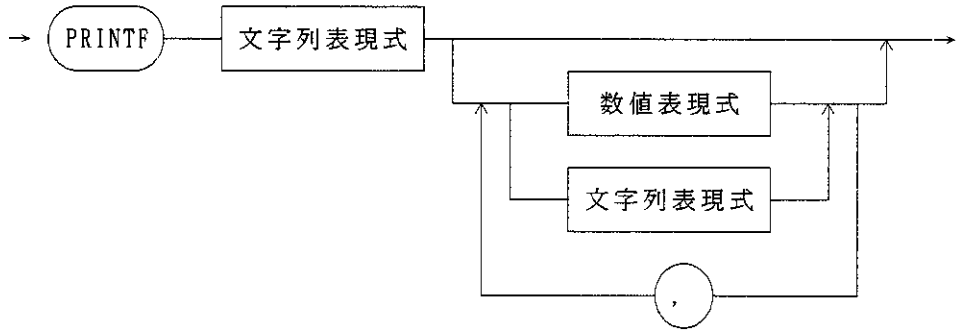
例

```
10 PRINTER 1
```

45. PRINTF

概　要　数値または文字列を表示します。

構　文　(1)-1



(1)-2
PRINTF　文字列表現式〔数値表現式｜文字列表現式
{, 数値表現式｜文字列表現式}〕

解　説
・指定された数値、文字列を表示します。

・数値、文字列をカンマ(,) で区切って複数を指定すると、改行せずに数値、文字列を次々に出力します。改行する場合は＼n を書式指定式の中で指定します。

・第 1パラメータの文字列表現式が、その後のパラメータの書式を指定するために使われます。

・書式指定の方法は以下の通りです。

●PRINTF　書式指定式〔〔式〔式〔・・・〕〕〕

書式指定の方法はC言語のPrintf関数に似ています。
書式指定式は文字列型であって、% に続けて以下の方法で出力の書式を指定します。この書式以外の文字列は単純に出力されます。% を出力したい場合は、%%と続けます。

% 〔-〕〔0〕〔m〕〔. n〕文字
- …… 指定されたフィールド内で左詰めにする。この指定がなければ右詰めにする。
0 …… 指定フィールドの余った部分に詰める文字を、スペースでなく 0にする。
m …… m 文字分のフィールドを取る。
. n …… n 桁の精度で出力する。文字列に対して指定すると、この値が実際の文字列の長さになる。
文字…… d ; 符号付10進数　　s ; 文字列
o ; 8 進数　　e ; 浮動小数点表現（指数形式）
x ; 16進数　　f ; 浮動小数点表現

例

```
10 N = 500000
20 U = LOG(1+1/N)
30 V = U - 1 / N
40 PRINTF "%7d %16.5e %16.5e\n",N,U,V
50 PRINTF "%s\n","end"
```

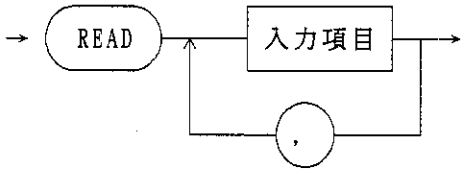
<実行結果>

```
 500000 2.00000e-06 -1.99994e-12
end
```

46.　READ

概　要　DATA文の定数を、変数に代入します。

構　文　(1)-1



(1)-2

READ　入力項目 {, 入力項目}

解　説

・DATA文で定義されている数値、文字列を、引数で指定してある変数に読み込みます。

・READ文が現れたところで、プログラムの中からDATA文を捜します。

・最初のREADでは、原則として（RESTORE 文で変更されていなければ）、プログラムの先頭行から行番号順に捜して、最初に発見した値を引き数並びの変数に代入します。
その後、順に対応するDATA文の定数を捜して代入します。

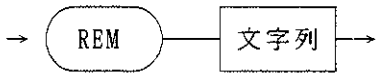
・READの変数に対して、DATAで指定する定数の数の方が少ない場合には、エラーとなります。

・READで読み出す変数の数と、DATA文の 1行の定数の数が一致する必要はありません。

47.　REM

概　要　プログラムの注釈です。

構　文　(1)-1



(1)-2
REM　文字列

解　説

・プログラム中に注釈をつけたいときに使用します。

・REM は非実行ステートメントですから、REM に続く文字列はいかなるものでも構いません。すべての文字、数字、記号が使用できます。

・REM は、感嘆符(!) で代用できます。

・REM の後にコロン(:) によるマルチ・ステートメントは使用できません。すべて注釈文として見なされます。

例

```
10 REM "PROGRAM 1"
20 ! 1983-JUN-02
30 A=A+1:! INCREMENT A
```

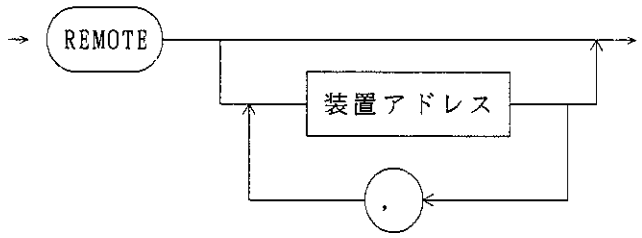
48.　REMOTE

概　要

指定した装置をリモート状態にするか、またはリモート・イネーブル(REN)ラインを真にします。

構　文

(1)-1



(1)-2

REMOTE〔装置アドレス｛,装置アドレス　｝〕

解　説

・装置アドレスを指定せずにREMOTEだけを実行した場合、GPIBのリモート・イネーブル（REN)ラインが真（Low level)となり、GPIB上に接続された装置をリモート・コントロール可能な状態にします。
REN ラインを偽（Highlevel)にするためには、LOCAL を実行して下さい。

・REMOTEに続いて装置アドレスを指定した場合、装置アドレスで指定された装置のみをリモート状態にします（ただし、RENラインが真のときのみ）。
装置アドレスは複数を指定できます。
またリモート状態を解除するためには、LOCAL を実行して下さい。

・REMOTEは、選択した装置をリモート状態にするものですが、以下に示すステートメントを実行したときは、REMOTEを実行しなくても自動的に指定した装置をリモート状態にします（ただし、REN ラインが真のときのみ）。

CLEAR 〔装置ｱﾄﾞﾚｽ｛, 装置ｱﾄﾞﾚｽ｝〕
OUTPUT 装置ｱﾄﾞﾚｽ｛, 装置ｱﾄﾞﾚｽ｝; < 出力データ>｛, <出力データ>｝
REMOTE〔装置ｱﾄﾞﾚｽ｛, 装置ｱﾄﾞﾚｽ｝〕
SEND LISTEN 装置ｱﾄﾞﾚｽ｛, 装置ｱﾄﾞﾚｽ｝
TRIGGER 装置ｱﾄﾞﾚｽ｛, 装置ｱﾄﾞﾚｽ｝

例

```
10 REMOTE 1
20 REMOTE 5
30 REMOTE 1  2  3
```

注　意

ADDRESSABLE モードでは機能しません。

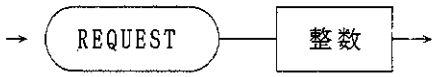
49. REQUEST

概　要

ADDRESSABLE モード時に外部GPIBコントローラへ送信するステータス・バイトを設定します。

構　文

(1)-1



(1)-2

REQUEST 整数

注）整数の設定範囲は 0～255 です。

解　説

・ADDRESSABLE モード時に外部GPIBコントローラへ送信するステータス・バイトを設定します。

・サービス・リクエストを発信する場合は、64～127 または 192～255(ビット6 が1)の値を設定します。

例

10 REQUEST 65

注　意

・SYSTEM CONTROLLER モードでは機能しません。

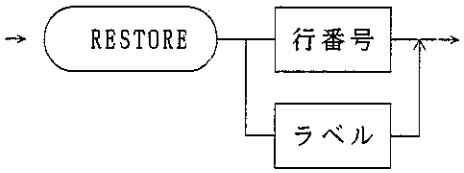
・外部コントローラからはシリアルポールを用いて読んで下さい。GPIBコマンドの*STB? では読めません。

・GPIBコマンドのSRQDが実行されている場合は、ステータス・バイトのビット6 は常に 0で送信されます。
したがって、サービス・リクエストも発信されません。

50.　RESTORE

概　要　次のREAD文で読み込むDATA行を指定します。

構　文　(1)-1



(1)-2
RESTORE

解　説

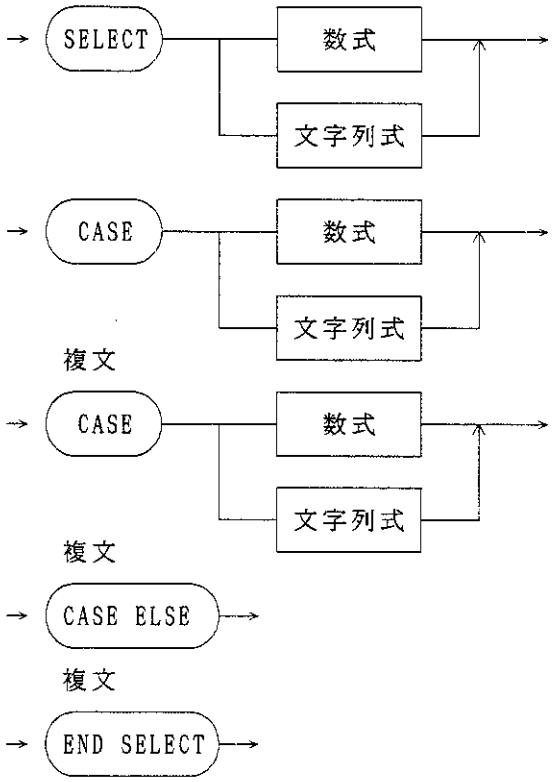
・行番号またはラベルで指定します。
特に指定がなければ、プログラムの先頭行から順番にDATA文の定数が読み込まれ、RESTORE で次のREADの対象となるDATA文を指定できます。

・引数の行番号がDATA文を捜し始める先頭行という判断をするので、その行以降の最初のDATA文が指定するものであれば構いません。

51.　SELECT, CASE, END SELECT

概　要　　1つの式の値を条件として、複数の分岐を行います。

構　文　(1)-1



(1)-2

```
SELECT <数式 | 文字列式>
CASE   <数式 | 文字列式>
  複文
CASE   <数式 | 文字列式>
  複文
CASE ELSE
  複文
END SELECT
```

解　説

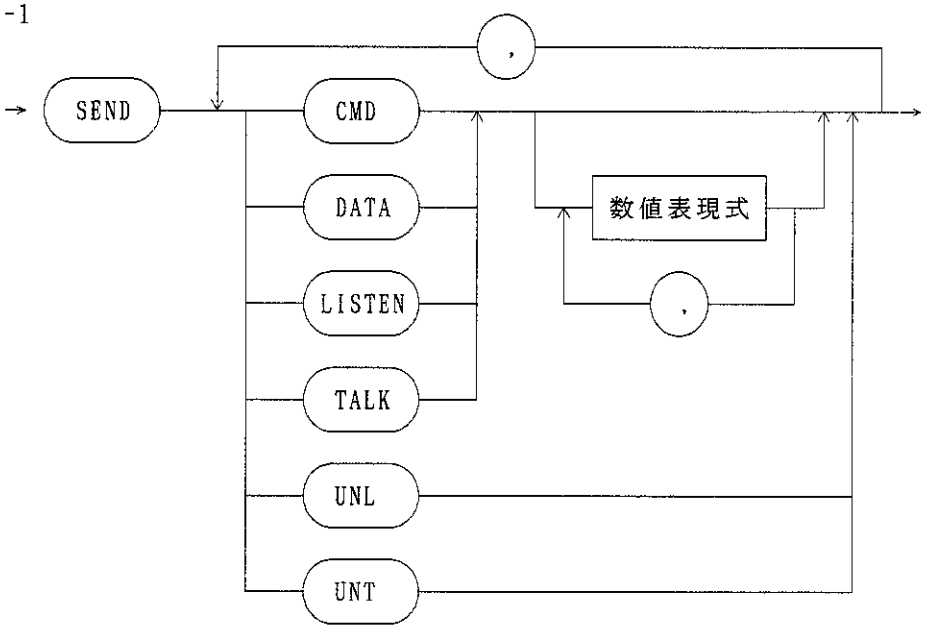
・SELECTで指定した式の値に、一致するCASE以下の文（複文）を実行します。実行の対象は、次のCASE、CASE ELSE、またはEND SELECTまでです。

・SELECT構文自体の入れ子ができます。この場合内部のSELECTは、完全に外部のものを含む形になります。

52.　SEND

概　要　GPIBにコマンドおよびデータを出力します。

構　文　(1)-1



(1)-2

SEND <A | B> { , <A | B> }

注）A:<CMD | DATA | LISTEN | TALK> 〔数値表現式 {, 数値表現式} 〕
　　B:UNL | UNT

解　説

・GPIB上にユニバーサル・コマンド、アドレス・コマンド、およびデータなどを独立に送ります。

CMD　：アテンション（ATN)ラインを真（Low level)にして、与えられた数値をGPIBに送ります。ただし、数値は8ビットのバイナリ・データに変換されて、GPIBに出力されます。したがって、扱う数値は0 ～255 の範囲内で、また小数点表現の数値は自動的に整数に変換されます。

DATA　：ANT ラインを偽（High level）にして、与えられた数値をGPIBに送ります。ただし、ここで扱う数値は CMDで扱われるものと同様です。

LISTEN：与えられた数値を、リスナ・アドレス・グループ（LAG)としてGPIB上に送ります。数値は複数を指定できます。

TALK　：与えられた数値をトーカ・アドレス・グループ（TAG)としてGPIB上に送ります。ただし、数値は複数を指定できません。

UNT　：アントーク（UNT)コマンドをGPIBに送ります。このコマンドを実行する前にトーカに指定されていた装置は、トーカを解除されます。

UNL　：アンリスン（UNL)コマンドをGPIBに送ります。このコマンドを実行する前にリスナに指定されていた装置は、リスナを解除されます。

例

```
10 SEND UNT UNL LISTEN 1, 2, 3 TALK 4
20 SEND UNT CMD 63, 33 DATA 30, 54
```

注　意

ADDRESSABLE モードでは機能しません。

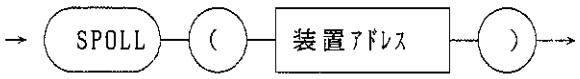
53.　SPOLL

概　要

指定した装置のシリアル・ポールを行い、ステータス・バイトを読み込みます。

構　文

(1)-1



(1)-2

SPOLL（装置アドレス）

解　説

・本器がSYSTEM CONTROLLER モードのとき、他のGPIB装置に対してシリアル・ポールを行います。

・装置アドレスが 0～30のときは、各アドレスに対応した装置のシリアル・ポールを行います。

・装置アドレスが31のとき、SYSTEM CONTROLLER モード、ADDRESSABLE モードに関係なく、本器に対してステータス・バイトを取り出します。

例

```
10    OUTPUT 31;"OLDC ON"
20    ON ISRQ GOTO 70
30    ENABLE INTR
40    OUTPUT 31;"SRQE"
50    OUTPUT 31;"SINGLE"
60    GOTO 60
70    PRINT SPOLL(31)
80    STOP
```

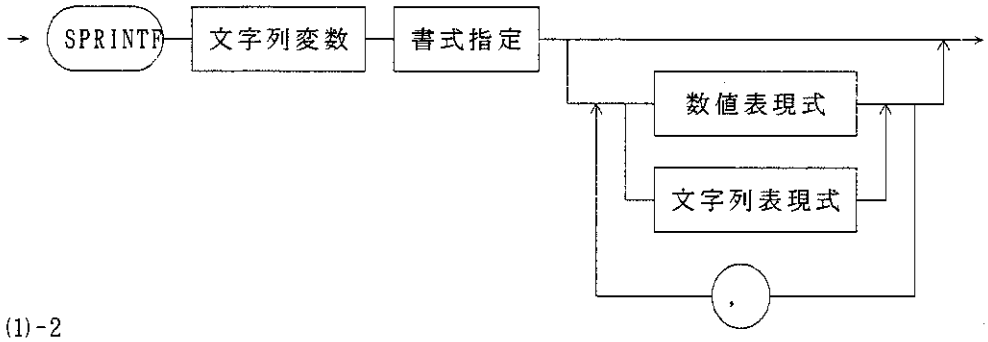
注　意

ADDRESSABLE モード時に装置アドレス 0～30を指定し、SPOLL を行った場合は、0 が返ります。

54. SPRINTF

概　要　PRINTFコマンドの書式変換仕様にしたがって書式を変換し、文字列変数に結果を代入します。

構　文　(1)-1



(1)-2

SPRINTF 文字列変数　書式指定〔数値表現式｜文字列表現式
{,数値表現式｜文字列表現式}〕

解　説

・PRINTFの書式変換の方法で式の値を変換して、最初のパラメータの文字列変数に結果を代入します。

・書式指定の方法と式の数、それに結果を入れる文字列変数の大きさには十分な注意が必要です。
特に結果をいれる文字列が結果に対して十分な大きさがないと、BASIC バッファを破壊する恐れがあります。

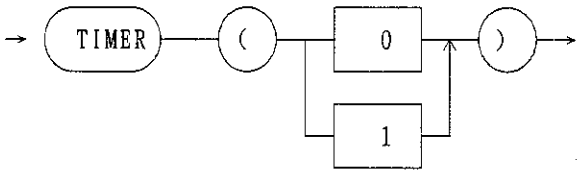
書式の指定方法は、4.3 節の[45. PRINTF]を参照して下さい。

55.　TIMER

概　要　内部システム時間の読み出しおよびリセットをします。

構　文

(1)-1



(1)-2

```
TIMER (0 | 1)
```

解　説

・内部システム時間をsec 単位の値で返す組み込み関数です。
この関数は、主に実行時間の測定などに使います。

引数 0を指定した場合 ：　内部システム時間の読み出し
引数 1を指定した場合 ：　内部システム時間のリセット

・読み出した値は10msecの分解能で、±10msecの誤差があります。

例

```
10  INTEGER I
20  TIMER(1)
30  FOR I=0 TO 10000
40  NEXT I
50  T1=TIMER(0)
60  !
70  TIMER(1)
80  FOR I=0 TO 10000
90    PRINT I
100 NEXT I
110 T2=TIMER(0)
120 !
130 PRINT "PRINT Command execute time is ";T2-T1
140 STOP
```

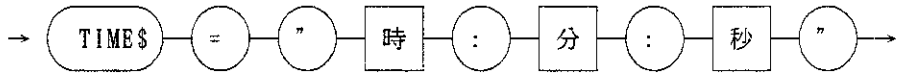
56.　TIME$

概　要　時刻の読み出しおよび設定をします。

構　文　(1)-1

(1)-2
TIME$

(2)-1



(2)-2
TIME$=" 時: 分: 秒"

解　説　・本器内蔵の時計(RTC) の時刻を読み出します。

・読み出した時刻は変更できます。
以下のように入力して下さい。

```
TIME$="23:43:12"
TIME$="11:5:6"
```

例

```
10 DIM T$[10]
20 T$=TIME$
30 PRINT "Time is ";T$
40 PRINT "Time Reset"
50 TIME$="0:0:0"
60 STOP
```

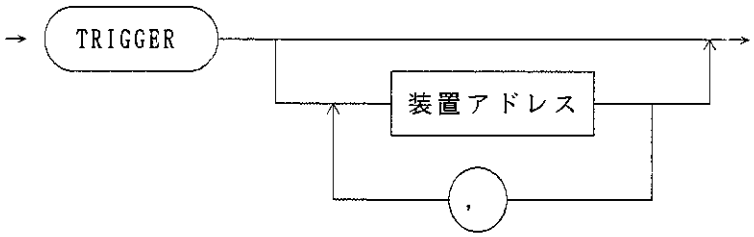
57.　TRIGGER

概　要

GPIB上に接続されているすべての装置、または選択された特定の装置にアドレス・コマンド・グループ（ACG)のグループ・エグゼキュート・トリガ(GET）を送ります。

構　文

(1)-1



(1)-2

TRIGGER 〔装置アドレス｛，装置アドレス　｝〕

解　説

・装置アドレスを指定せずにTRIGGER だけを実行すると、GPIBにはアドレス・コマンドのグループ・エクグゼュート・トリガ（Group Execute Trigger-GET)のみが送られます。この場合、トリガをかけたい装置はあらかじめリスナに設定しておかなければなりません。

・TRIGGER に続いて装置アドレスを指定すると、装置アドレスで指定された装置のみに GETコマンドを送ります。

例

```
10 TRIGGER 1
20 TRIGGER
```

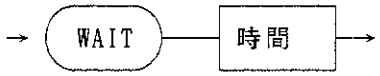
注　意

ADDRESSABLE モードでは機能しません。

58. WAIT

概　要　指定時間だけ待ちます。

構　文　(1)-1



(1)-2

```
WAIT　時間
```

解　説
・指定された時間待ちます。時間の単位はmsecです。
時間の設定範囲は 0～65535 です。

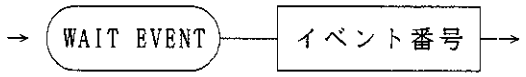
例

```
10 INTEGER T
20 T=30
30 PRINT T;"[msec] Wait !!"
40 WAIT T
50 STOP
```

59.　WAIT EVENT

概　要　指定したイベントが発生するまで待ちます。

構　文　(1)-1



(1)-2

```
WAIT EVENT　イベント番号
```

解　説
- 指定されたイベント番号のイベントが発生するまで待ちます。
  イベント番号 : 1; スイープエンド

例

```
10 INTEGER EV
20 EV=1
25 OUTPUT 31;"OLDC OFF"
30 OUTPUT 31;"INIT:CONT OFF;:ABOR;INIT"
40 WAIT EVENT EV
50 PRINT "SWEEP FINISHED"
60 STOP
```

4.4　ビルトイン関数

4.4.1　概要

ビルトイン関数は、ネットワーク・アナライザで測定したデータを、高速処理できる組み込み関数で、測定データの解析や判定に使用します。
従来のネットワーク・アナライザと基本的な関数は同じですが、一部追加や削除した関数がありますので注意して下さい。また、処理スピードも向上しています。
ビルトイン関数で扱う数値には単位を指定できません。すべて基準単位として処理されます。

例　100MHzのアドレス・ポイントを求める場合

```
P = POINT2(1E+8,0)
```

また、ビルトイン関数が返す応答データも同様に基準単位の数値となります。

(1)　測定ポイントとアドレス・ポイント

測定データの解析範囲の指定や、測定データの中の位置指定には、アドレス・ポイントを使用します。アドレス・ポイントは、0 ～1200の値で測定データの指定を行います。以下のような対応づけになります。

●測定ポイント数が1201のとき

最初のデータ	アドレス・ポイント 0
2 番目のデータ	アドレス・ポイント 1
3 番目のデータ	アドレス・ポイント 2
⋮	
n 番目のデータ	アドレス・ポイント n-1
⋮	
1201番目のデータ	アドレス・ポイント 1200

●測定ポイント数が 601のとき

最初のデータ	アドレス・ポイント 0
2 番目のデータ	アドレス・ポイント 2
3 番目のデータ	アドレス・ポイント 4
⋮	
n 番目のデータ	アドレス・ポイント 2(n-1)
⋮	
601 番目のデータ	アドレス・ポイント 1200

●測定ポイント数が 301のとき

最初のデータ	アドレス・ポイント 0
2 番目のデータ	アドレス・ポイント 4
3 番目のデータ	アドレス・ポイント 8
⋮	
n 番目のデータ	アドレス・ポイント 4(n-1)
⋮	
301 番目のデータ	アドレス・ポイント 1200

このように、測定ポイントが1201のときはアドレス・ポイントが1 増加し、それ以外のときは1 以上増加します。

測定ポイント数とアドレス・ポイントの増加値の関係は、以下のようになります。

測定ポイント数	アドレス・ポイントの増加値	測定ポイント数	アドレス・ポイントの増加値
1201	1	101	12
801 *	1	51	24
601	2	21	60
401	3	11	120
301	4	6	240
201	6	3	600

ユーザ掃引、プログラム掃引時も、この関係に準じます。測定ポイント数が1201でユーザ掃引、プログラム掃引にしたときは、アドレス・ポイントの増加値は必ず1 になります。アドレス・ポイント0 からつめてデータを配置します。測定データ・ポイント数を601 にし、セグメントのポイント数の合計が601 を越えない限り、測定データは1 ポイントおきに配置されます。また、アドレス・ポイントで指定すると測定ポイント数を変えた場合でも、ビルトイン関数の指定を変更しないですむメリットがあります。

* : 測定ポイントが801 ポイントの場合、アドレス・ポイントの増加値は1 になります。801 ～1200ポイントを指定するとエラーになります。

(2) 解析チャンネル

解析チャンネルでは、ビルトイン関数で解析するデータを指定します。解析対象となる本器内部のデータには以下のものがあります。複素数データは解析に使用することはできませんが、データ転送にのみ使用することができます。

①表示データ
②メイン・トレース・データ
③サブ・トレース・データ
④メイン・トレース複素数データ
⑤サブ・トレース複素数データ

これらのデータの解析チャンネル指定は、以下のようになります。

① 表示データ

表示データには、表示されているデータが格納されています。表示のフォーマットやメジャーの指定で格納されるデータが変わります。表示されていないチャンネルやメモリへのコピーを行っていないときのメモリ・データの内容は不定です。

各測定チャンネルと解析チャンネル

CH1	CH2	CH3	CH4	
0	1	4	5	測定表示第1 波形データ *1
8	9	12	13	測定表示第2 波形データ *2
2	3	6	7	メモリ表示第1 波形データ *3
10	11	14	15	メモリ表示第2 波形データ *4

*1 : 1画面に 1波形表示される場合には表示データが格納されます。 1画面に 2波形表示される場合には第 1波形が格納されます。
第 1波形：フォーマットがLOGMAG&PHASEのときのLOGMAG
メジャーがS11&S21 のときのS11

*2 : 1画面に 1波形表示される場合には内容は不定です。 1画面に 2波形が表示される場合には第 2波形が格納されます。
第 2波形：フォーマットがLOGMAG&PHASEのときのPHASE
メジャーがS11&S21 のときのS21

*3 : メモリへのコピーがされていないときには内容は不定です。

*4 : メモリへコピーした場合でも、そのときに 2波形表示でないと内容は不定になります。

② メイン・トレース・データ

トレース・データとは、表示データになる前のデータです。LOGMAG、位相、実数部、虚数部のデータを内部データとして保持しています。これらの内部データは表示フォーマットとは関係なく持っているので、表示データにないデータを解析する場合に有効です。表示データに対してスムージングを演算させた場合でもこのデータは変わりません。

メジャーがS11&S21 などのように 1画面に 2個の測定データがある場合には、区別するためにそれぞれの波形を以下のように呼び分けます。

第 1波形に相当するトレース・データ：メイン・トレース・データ

第 2波形に相当するトレース・データ：サブ・トレース・データ

S11 、S21 などのメジャーの場合には、常にメイン・トレース・データとなります。

各測定チャンネルと解析チャンネル

CH1	CH2	CH3	CH4	
32	36	48	52	LOGMAGデータ *1
33	37	49	53	位相データ *1
34	38	50	54	実数部データ *1
35	39	51	55	虚数部データ *1
40	44	56	60	メモリのLOGMAGデータ *2
.41	45	57	61	メモリの位相データ *2
42	46	58	62	メモリの実数部データ *2
43	47	59	63	メモリの虚数部データ *2

*1 ：指定されたチャンネルで測定していないときには内容は不定になります。

*2 ：メモリへのコピーが行われていないときには、内容は不定になります。

③ サブ・トレース・データ

各測定チャンネルと解析チャンネル

CH1	CH2	CH3	CH4	
64	68	80	84	LOGMAGデータ *1
65	69	81	85	位相データ *1
66	70	82	86	実数部データ *1
67	71	83	87	虚数部データ *1
72	76	88	92	メモリのLOGMAGデータ *2
73	77	89	93	メモリの位相データ *2
74	78	90	94	メモリの実数部データ *2
75	79	91	95	メモリの虚数部データ *2

*1 ：指定されたチャンネルで測定していないときには内容は不定になります。

*2 ：メモリへのコピーが行われていないときには、内容は不定になります。

④　メイン・トレース複素数データ

内部の複素数データを扱う場合、TRANSRやTRANSWなどのデータ転送だけが可能になります。

各測定チャンネルと解析チャンネル

CH1	CH2	CH3	CH4	
128	192	256	320	トレース・データ　*1
132	196	260	324	トレース・メモリ・データ　*2
129	193	257	321	補正演算後のデータ　*1
130	194	258	322	補正演算後のメモリ・データ　*2
131	195	259	323	補正演算前のデータ　*1
133	197	261	325	ノーマライズ基準データ　*3
134	198	262	326	1ポート補正：方向性エラー係数　*3
135	199	263	327	1ポート補正：ソース・マッチ・エラー係数　*3
136	200	264	328	1ポート補正：反射ﾄﾗｯｷﾝｸﾞ･ｴﾗｰ係数　*3
137	201	265	329	2ポート補正：順方向方向性エラー係数　*4
138	202	266	330	2ポート補正：順方向ｿｰｽ･ﾏｯﾁ･ｴﾗｰ 係数　*4
139	203	267	331	2ポート補正：順方向反射ﾄﾗｯｷﾝｸﾞ･ｴﾗｰ係数　*4
140	204	268	332	2ポート補正：順方向ﾛｰﾄﾞ･ﾏｯﾁ･ｴﾗｰ 係数　*4
141	205	269	333	2ポート補正：順方向伝送ﾄﾗｯｷﾝｸﾞ･ｴﾗｰ係数　*4
142	206	270	334	2ポート補正：順方向ｱｲｿﾚｰｼｮﾝ･ｴﾗｰ係数　*4
143	207	271	335	2ポート補正：逆方向方向性エラー係数　*4
144	208	272	336	2ポート補正：逆方向ｿｰｽ･ﾏｯﾁ･ｴﾗｰ 係数　*4
145	209	273	337	2ポート補正：逆方向反射ﾄﾗｯｷﾝｸﾞ･ｴﾗｰ係数　*4
146	210	274	338	2ポート補正：逆方向ﾛｰﾄﾞ･ﾏｯﾁ･ｴﾗｰ 係数　*4
147	211	275	339	2ポート補正：逆方向伝送ﾄﾗｯｷﾝｸﾞ･ｴﾗｰ係数　*4
148	212	276	340	2ポート補正：逆方向ｱｲｿﾚｰｼｮﾝ･ｴﾗｰ係数　*4
149	213	277	341	ﾉｰﾏﾗｲｽﾞ&ｱｲｿﾚｰｼｮﾝ 補正：ﾉｰﾏﾗｲｽﾞ基準データ　*3
150	214	278	342	ﾉｰﾏﾗｲｽﾞ&ｱｲｿﾚｰｼｮﾝ 補正：ｱｲｿﾚｰｼｮﾝ･ｴﾗｰ係数　*3

*1 ：指定されたチャンネルで測定していないときには内容は不定になります。

*2 ：メモリへのコピーが行われていないときには、内容は不定になります。

*3 ：補正を行っていないときには内容は不定になります。

*4 ：補正を行っていないときには内容は不定になります。また、CH1 とCH3 、CH2 とCH4 の補正データの内容は同じになります。

⑤ サブ・トレース複素数データ

サブ・トレースの複素数データは次のような割り付けになっています。

各測定チャンネルと解析チャンネル

CH1	CH2	CH3	CH4	
160	224	288	352	トレース・データ *1
164	228	292	356	トレース・メモリ・データ *2
161	225	289	353	補正演算後のデータ *1
162	226	290	354	補正演算後のメモリ・データ *2
163	227	291	355	補正演算前のデータ *1
165	229	293	357	ノーマライズ基準データ *3
166	230	294	358	1ポート補正：方向性エラー係数 *3
167	231	295	359	1ポート補正：ソース・マッチ・エラー係数 *3
168	232	296	360	1ポート補正：反射ﾄﾗｯｷﾝｸﾞ･ｴﾗｰ係数 *3
169	233	297	361	2ポート補正：順方向方向性エラー係数 *4
170	234	298	362	2ポート補正：順方向ｿｰｽ･ﾏｯﾁ･ｴﾗｰ 係数 *4
171	235	299	363	2ポート補正：順方向反射ﾄﾗｯｷﾝｸﾞ･ｴﾗｰ係数 *4
172	236	300	364	2ポート補正：順方向ﾛｰﾄﾞ･ﾏｯﾁ･ｴﾗｰ 係数 *4
173	237	301	365	2ポート補正：順方向伝送ﾄﾗｯｷﾝｸﾞ･ｴﾗｰ係数 *4
174	238	302	366	2ポート補正：順方向ｱｲｿﾚｰｼｮﾝ･ｴﾗｰ係数 *4
175	239	303	367	2ポート補正：逆方向方向性エラー係数 *4
176	240	304	368	2ポート補正：逆方向ｿｰｽ･ﾏｯﾁ･ｴﾗｰ 係数 *4
177	241	305	369	2ポート補正：逆方向反射ﾄﾗｯｷﾝｸﾞ･ｴﾗｰ係数 *4
178	242	306	370	2ポート補正：逆方向ﾛｰﾄﾞ･ﾏｯﾁ･ｴﾗｰ 係数 *4
179	243	307	371	2ポート補正：逆方向伝送ﾄﾗｯｷﾝｸﾞ･ｴﾗｰ係数 *4
180	244	308	372	2ポート補正：逆方向ｱｲｿﾚｰｼｮﾝ･ｴﾗｰ係数 *4
181	245	309	373	ﾉｰﾏﾗｲｽﾞ&ｱｲｿﾚｰｼｮﾝ 補正：ﾉｰﾏﾗｲｽﾞ基準データ *3
182	246	310	374	ﾉｰﾏﾗｲｽﾞ&ｱｲｿﾚｰｼｮﾝ 補正：ｱｲｿﾚｰｼｮﾝ･ｴﾗｰ係数 *3

*1 ：指定されたチャンネルで測定していないときには内容は不定になります。

*2 ：メモリへのコピーが行われていないときには、内容は不定になります。

*3 ：補正を行っていないときには内容は不定になります。

*4 ：補正を行っていないときには内容は不定になります。メイン・トレースの補正データと同じになります。

(3)　ビルトイン関数の応答形式

ビルトイン関数の応答形式には3種類あります。

●測定ポイント　　：測定データのあるアドレス・ポイント。
（例）　MAX関数

●アドレス・ポイント：測定ポイント以外のときは、補間してアドレス・ポイントでの値にする。
（例）　VALUE関数

●コンペンセート　　：アドレス・ポイントの間でも、補間して値を求める。
（例）　CVALUE 関数

4.4.2　ビルトイン関数一覧

●アドレス・ポイント関係

```
POINT1(F,C)       : meas point     ; 指定周波数に最も近い測定ポイント
POINT2(F,C)       : address point  ; 指定周波数に最も近いアドレス・ポイント
DPOINT(F0,F1,C)   : address point  ; 指定周波数幅に対応するアドレス・ポイント幅
POINT1L(F,C)      : meas point     ; 指定周波数を越えない最も大きい測定ポイント
POINT1H(F,C)      : meas point     ; 指定周波数より大きな最も小さな測定ポイント
POINT2L(F,C)      : address point  ; 指定周波数を越えない最も大きいアドレス・ポイント
POINT2H(F,C)      : address point  ; 指定周波数より大きな最も小さなアドレス・ポイント
SWPOINT(C)        : meas point     ; 最新の測定ポイント
```

●周波数関係

```
FREQ(P,C)         : address point  ; 指定アドレス・ポイントに対応する周波数
DFREQ(P0,P1,C)    : address point  ; 指定アドレス・ポイント幅に対応する周波数幅
SWFREQ(C)         : meas point     ; 最新の測定周波数
```

●レスポンス関係

```
VALUE(P,C)        : address point  ; 指定アドレス・ポイントでのレスポンス値
DVALUE(P0,P1,C)   : address point  ; 指定アドレス・ポイント間のレスポンス値差
CVALUE(F,C)       : compensate     ; 指定周波数でのレスポンス値
DCVALUE(F0,F1,C)  : compensate     ; 指定周波数間でレスポンス値差
SWVALUE(C)        : meas point     ; 最新のレスポンス値
```

● 最大値・最小値関係

MAX(P0,P1,C) : meas point ; 指定アドレス・ポイント間の最大レスポンス値
FMAX(P0,P1,C) : meas point ; 指定アドレス・ポイント間の最大レスポンスの周波数
PMAX(P0,P1,C) : meas point ; 指定アドレス・ポイント間の最大レスポンスの測定ポイント
MIN(P0,P1,C) : meas point ; 指定アドレス・ポイント間の最小レスポンス値
FMIN(P0,P1,C) : meas point ; 指定アドレス・ポイント間の最小レスポンスの周波数
PMIN(P0,P1,C) : meas point ; 指定アドレス・ポイント間の最小レスポンスの測定ポイント

● 帯域幅関係

BND(P,X,C) : compensate ; 指定アドレス・ポイントから指定データ減衰した帯域幅
BNDL(P,X,C) : compensate ; 指定アドレス・ポイントから指定データ減衰した低周波数側の周波数
BNDH(P,X,C) : compensate ; 指定アドレス・ポイントから指定データ減衰した高周波数側の周波数
CBND(F,X,C) : compensate ; 指定周波数から指定データ減衰した帯域幅
CBNDL(F,X,C) : compensate ; 指定周波数から指定データ減衰した低周波数側の周波数
CBNDH(F,X,C) : compensate ; 指定周波数から指定データ減衰した高周波数側の周波数
MBNDI(P0,P1,P,N,La,Fa,C) : compensate ; 指定アドレス・ポイント間で、指定アドレス・ポイントから指定データ減衰した低周波数側の周波数、高周波数側の周波数、中心周波数、帯域幅
MBNDO(P0,P1,P,N,La,Fa,C) : compensate ; 指定アドレス・ポイント間で、指定アドレス・ポイントから指定データ減衰した低周波数側の周波数、高周波数側の周波数、中心周波数、帯域幅

● リップル関係-1

RPL1(P0,P1,dX,dY,C) : meas point ; 指定アドレス・ポイント間の極大値と極小値の差
RPL2(P0,P1,dX,dY,C) : meas point ; 指定アドレス・ポイント間の隣あう極大値と極小値の差の最大値
RPL3(P0,P1,dX,dY,C) : meas point ; 指定アドレス・ポイント間の隣あう極大と極小の差を加算した中の最大値
RPL4(P0,P1,dX,dY,C) : meas point ; 指定アドレス・ポイント間の隣あう極大値と極小値の差の最大値
RPL5(P0,P1,dX,dY,C) : meas point ; 指定アドレス・ポイント間の極大値の最大値
RPL6(P0,P1,dX,dY,C) : meas point ; 指定アドレス・ポイント間の極大値の最小値
RPLF(P0,P1,dX,dY,C) : meas point ; 指定アドレス・ポイント間の最初の極大点と極小点の周波数差
RPLR(P0,P1,dX,dY,C) : meas point ; 指定アドレス・ポイント間の最初の極大点と極小点のレスポンス差
RPLH(P0,P1,dX,dY,C) : meas point ; 指定アドレス・ポイント間の最初の極大値のレスポンス値
FRPLH(P0,P1,dX,dY,C) : meas point ; 指定アドレス・ポイント間の最初の極大値の周波数
PRPLH(P0,P1,dX,dY,C) : meas point ; 指定アドレス・ポイント間の最初の極大値の測定ポイント
RPLL(P0,P1,dX,dY,C) : meas point ; 指定アドレス・ポイント間の最初の極小値のレスポンス値
FRPLL(P0,P1,dX,dY,C) : meas point ; 指定アドレス・ポイント間の最初の極小値の周波数
PRPLL(P0,P1,dX,dY,C) : meas point ; 指定アドレス・ポイント間の最初の極小値の測定ポイント

● リップル関係-2

NRPLH(P0,P1,dX,dY,C) : meas point ; 指定アドレス・ポイント間の極大点の数
NRPLL(P0,P1,dX,dY,C) : meas point ; 指定アドレス・ポイント間の極小点の数
PRPLHN(N,C) : meas point ; NRPLH で求めたN 番目の極大値の測定ポイント
PRPLLN(N,C) : meas point ; NRPLL で求めたN 番目の極小値の測定ポイント
FRPLHN(N,C) : meas point ; NRPLH で求めたN 番目の極大値の周波数
FRPLLN(N,C) : meas point ; NRPLL で求めたN 番目の極小値の周波数
VRPLHN(N,C) : meas point ; NRPLH で求めたN 番目の極大値のレスポンス値
VRPLLN(N,C) : meas point ; NRPLL で求めたN 番目の極小値のレスポンス値
PRPLHM(Pa,C) : meas point ; NRPLH で求めた極大値の測定ポイント配列
PRPLLM(Pa,C) : meas point ; NRPLL で求めた極小値の測定ポイント配列
FRPLHM(Xa,C) : meas point ; NRPLH で求めた極大値の周波数配列
FRPLLM(Xa,C) : meas point ; NRPLL で求めた極小値の周波数配列
VRPLHM(Xa,C) : meas point ; NRPLH で求めた極大値のレスポンス値配列
VRPLLM(Xa,C) : meas point ; NRPLL で求めた極小値のレスポンス値配列

●ダイレクトサーチ関係

```
DIRECT(P0,P1,X,C)    : address point ; 指定アドレス・ポイント間で最初に検出されたデータの最も近いアドレス・ポイント
DIRECTL(P0,P1,X,C)   : meas point    ; 指定アドレス・ポイント間で低周波数側からサーチして最初に検出されたデータの測定ポイント
DIRECTH(P0,P1,X,C)   : meas point    ; 指定アドレス・ポイント間で高周波数側からサーチして最初に検出されたデータの測定ポイント
CDIRECT(F0,F1,X,C)   : compensate    ; 指定周波数間で最初に検出されたデータの周波数
CDIRECTL(F0,F1,X,C)  : compemsate    ; 指定周波数間で低周波数側からサーチして最初に検出されたデータの周波数
CDIRECTH(F0,F1,X,C)  : compemsate    ; 指定周波数間で高周波数側からサーチして最初に検出されたデータの周波数
DDIRECT(P0,P1,X,C)   : address point ; 指定アドレス・ポイント間で指定データのアドレス・ポイント幅
CDDIRECT(F0,F1,X,C)  : compensate    ; 指定周波数間で指定データの帯域幅
ZEROPHS(P0,P1,C)     : compensate    ; 指定アドレス・ポイント間のゼロ位相の周波数
```

●データ転送関係

```
TRANSR(P0,P1,Xa,C) : meas point ; 指定アドレス・ポイント間の測定データの配列への転送
TRANSW(P0,P1,Xa,C) : meas point ; 配列から指定アドレス・ポイントへの転送
```

```
P,P0,P1  : アドレス・ポイント指定
F,F0,F1  : 周波数指定
C        : 解析チャンネル指定
dX       : 傾きの横軸指定
dY       : 傾きの縦軸指定
X        : レベル指定
N        : 個数や何番目の指定
Xa,La,Fa : 配列の指定
Pa       : 整数配列の指定
```

4.4.3　アドレス・ポイントを求める関数

(1)　測定ポイントを求める関数　POINT1, POINT1L, POINT1H

```
POINT1(周波数, 解析ﾁｬﾝﾈﾙ)
POINT1L(周波数, 解析ﾁｬﾝﾈﾙ)
POINT1H(周波数, 解析ﾁｬﾝﾈﾙ)
```

説明　　指定された周波数の測定ポイントを求めます。

POINT1関数　：指定周波数に最も近い測定ポイントを求めます。すなわち測定ポイントへの変換で四捨五入を行います。
POINT1L 関数：指定周波数を越えない最も大きい測定ポイントを求めます。すなわち測定ポイントへの変換で切り捨てを行います。
POINT1H 関数：指定周波数より大きな最も小さい測定ポイントを求めます。すなわち測定ポイントへの変換で切り上げを行います。

用途　　多くのビルトイン関数は、アドレス・ポイントを引き数にしています。他のビルトイン関数を使用するために、周波数を測定ポイントに変換します。解析範囲を指定するときに切り上げ、切り捨てた方が範囲の指定が正確になります。

例）　　
```
P0=POINT1L(F0,0)
P1=POINT1H(F1,0)
X =MAX(P0,P1,0)    周波数F0,F1 を含む範囲で最大値を探します。

P =POINT1(F,0)
Y =VALUE(P,0)      周波数F に最も近い測定データを読み出します。
```

(2)　アドレス・ポイントを求める関数　POINT2, POINT2L, POINT2H

```
POINT2(周波数, 解析ﾁｬﾝﾈﾙ)
POINT2L(周波数, 解析ﾁｬﾝﾈﾙ)
POINT2H(周波数, 解析ﾁｬﾝﾈﾙ)
```

説明　　指定された周波数のアドレス・ポイントを求めます。

POINT2関数　：指定周波数に最も近いアドレス・ポイントを求めます。すなわちアドレス・ポイントへの変換で四捨五入を行います。
POINT2L 関数：指定周波数を越えない最も大きいアドレス・ポイントを求めます。すなわちアドレス・ポイントへの変換で切り捨てを行います。
POINT2H 関数：指定周波数より大きな最も小さいアドレス・ポイントを求めます。すなわちアドレス・ポイントへの変換で切り上げを行います。

用途　　多くのビルトイン関数は、アドレス・ポイントを引き数にしています。他のビルトイン関数を使用するために、周波数をアドレス・ポイントに変換します。

例　　P =POINT2(F,0)
　　Y =VALUE(P,0)

周波数F に最も近い測定データを、測定ポイントのときには測定データ、そうでないときは補間して読み出します。

(3) アドレス・ポイント幅を求める関数　DPOINT

DPOINT(周波数1,　周波数2,　解析ﾁｬﾝﾈﾙ)

説明　　周波数幅に対応するアドレス・ポイント幅を求めます。

(4) 最新の測定ポイントを求める関数　SWPOINT

SWPOINT(解析ﾁｬﾝﾈﾙ)

説明　　測定中に最新の測定ポイントを求めます。

用途　　SWPOINT(解析チャンネル) を使用して掃引の状態を知ることができます。以下の例のように、掃引中にすでに掃引されたデータの解析もできます。

例　　*SWEEPING1
　　IF SWPOINT(0)<P1 THEN GOTO *SWEEPING1
　　X=MAX(P0,P1,0)

注意

本器が高速に掃引しているとき、測定ポイントは間欠読み出しになります。

4.4.4　周波数を求める関数

(1) 周波数を求める関数　FREQ

FREQ(ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ,　解析ﾁｬﾝﾈﾙ)

説明　　アドレス・ポイントを周波数に変換します。

用途　　アドレス・ポイントを返す関数の値を、周波数に変換します。

例　　P =PMAX(0,1200,0)
　　F =FREQ(P,0)
　　X =VALUE(P,0)

最大値の周波数とレスポンス値を求めます。MAX,FMAXを使用するより、サーチが一度で済むので、より高速に処理できます。

(2)　周波数幅を求める関数　DFREQ

DFREQ(ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ1,　ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ2,　解析ﾁｬﾝﾈﾙ)

説明　　指定したアドレス・ポイントから周波数幅へ変換します。

(3)　最新の周波数を求める関数　SWFREQ

SWFREQ(解析ﾁｬﾝﾈﾙ)

説明　　測定中に最新の測定周波数を求めます。

用途　　SWFREQ(解析チャンネル）を使用して、掃引している周波数を知ることができます。

例　　
```
*SWEEPING1
IF SWFREQ(0)<F1 THEN GOTO *SWEEPING1
X=CVALUE(F1)
```

注意

本器が高速に掃引しているとき、測定周波数は間欠読み出しになります。

4.4.5　レスポンスを求める関数

(1)　レスポンスを求める関数　VALUE

VALUE(ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ,　解析ﾁｬﾝﾈﾙ)

説明　　指定アドレス・ポイントのレスポンスを読み出します。アドレス・ポイントが測定ポイントでないときは、補間して求めます。

用途　　アドレス・ポイントを返す関数の値をレスポンス値に変換します。

例　　
```
P =PMAX(0,1200,0)
F =FREQ(P,0)
X =VALUE(P,0)
```

最大値の周波数とレスポンス値を求めます。MAX,FMAXを使用するより、サーチが一度で済むので、より高速に処理できます。

(2)　レスポンス差を求める関数　DVALUE

DVALUE(ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ1,　ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ2,　解析ﾁｬﾝﾈﾙ)

説明　　指定アドレス・ポイントのそれぞれのレスポンス値の差を求めます。

(3) レスポンス値を求める関数 CVALUE

```
CVALUE(周波数, 解析ﾁｬﾝﾈﾙ)
```

説明 指定した周波数に対応するレスポンス値を求めます。

(4) レスポンス差を求める関数 DCVALUE

```
DCVALUE(周波数1, 周波数2, 解析ﾁｬﾝﾈﾙ)
```

説明 指定周波数のそれぞれのレスポンス値の差を求めます。

(5) 最新のレスポンス値を求める関数 SWVALUE

```
SWVALUE(解析ﾁｬﾝﾈﾙ)
```

説明 測定中に最新の測定レスポンス値を求めます。

用途 レスポンス値をモニタして、調整などに使用できます。

例
```
*ADJUST
IF SWVALUE(33)<=PHASE1 THEN GOTO *ADJUST_END
OUTPUT 33;C
GOTO *ADJUST
*ADJUST_END
```

位相値がある値以下になるまでパラレルI/O へ出力します。

注意

本器が高速に掃引しているとき、測定レスポンス値は間欠読み出しになります。

4.4.6 最大値、最小値を求める関数

(1) 最大レスポンス値を求める関数 MAX

MAX(開始ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 終了ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 解析ﾁｬﾝﾈﾙ)

説明 指定アドレス・ポイント間で、最大レスポンス値をサーチします。

用途 共振点のレスポンス値を求めるときなどに使用します。

例 X =MAX(0,1200,0)

(2) 最大レスポンスの周波数を求める関数 FMAX

FMAX(開始ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 終了ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 解析ﾁｬﾝﾈﾙ)

説明 指定アドレス・ポイント間で、最大レスポンスの周波数を求めます。

用途 共振点の周波数を求めるときなどに使用します。

例 F =FMAX(0,1200,0)

(3) 最大レスポンスの測定ポイントを求める関数 PMAX

PMAX(開始ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 終了ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 解析ﾁｬﾝﾈﾙ)

説明 指定アドレス・ポイント間で、最大レスポンスの測定ポイントを求めます。

用途 共振点の周波数、レスポンス値を求めるときなどに使用します。また、他の解析のアドレス・ポイントなどに使用します。

例1
```
P =PMAX(0,1200,0)
F =FREQ(P,0)
X =VALUE(P,0)
```

最大値の測定ポイントから周波数、レスポンス値を求めます。FMAX,MAX使用時に比べ、一度のサーチなので、高速になります。

例2
```
P =PMAX(0,1200,0)
FB=BND(P,3,0)
```

ピークから3dB ダウンの帯域幅を求めます。

(4) 最小レスポンス値を求める関数 MIN

MIN(開始ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 終了ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 解析ﾁｬﾝﾈﾙ)

説明 指定アドレス・ポイント間で、最小レスポンス値をサーチします。

用途 反共振点のレスポンス値を求める場合などに使用します。

例 X =MIN(0,1200,0)

(5) 最小レスポンスの周波数を求める関数 FMIN

FMIN(開始ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 終了ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 解析ﾁｬﾝﾈﾙ)

説明 指定アドレス・ポイント間で、最小レスポンスの周波数を求めます。

用途 反共振点の周波数を求めるときなどに使用します。

例 F =FMIN(0,1200,0)

(6) 最小レスポンスの測定ポイントを求める関数 PMIN

PMIN(開始ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 終了ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 解析ﾁｬﾝﾈﾙ)

説明 指定アドレス・ポイント間の最小レスポンスの測定ポイントを求めます。

用途 反共振点の周波数、レスポンス値を求めるときなどに使用します。

（例） P =PMIN(0,1200,0)
F =FREQ(P,0)
X =VALUE(P,0)

最小値の測定ポイントから、周波数・レスポンス値を求めます。FMIN,MINの使用時に比べ、一度のサーチなので高速になります。

4.4.7　帯域幅などを求める関数

(1)　帯域幅を求める関数　BND

BND(ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ,　減衰ﾚﾍﾞﾙ,　解析ﾁｬﾝﾈﾙ)

説明　　指定アドレス・ポイントから、指定された減衰レベルだけ減衰したポイントをサーチし、帯域幅を求めます。サーチは指定アドレス・ポイントから外側に行います。

用途　　3dB ダウンの帯域幅などを求めます。

例　　
```
P =PMAX(0,1200,0)
F =BND(P,3,0)
```

3dB ダウンの帯域幅を求めます。

(2)　帯域幅の低周波数側の周波数を求める関数　BNDL

BNDL(ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ,　減衰ﾚﾍﾞﾙ,　解析ﾁｬﾝﾈﾙ)

説明　　指定アドレス・ポイントから、指定された減衰レベルだけ減衰したポイントを低周波数側にサーチし、その周波数を求めます。

用途　　BNDHと組み合わせて中心周波数を求めることができます。

(3)　帯域幅の高周波数側の周波数を求める関数　BNDH

BNDH(ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ,　減衰ﾚﾍﾞﾙ,　解析ﾁｬﾝﾈﾙ)

説明　　指定アドレス・ポイントから、指定された減衰レベルだけ減衰したポイントを高周波数側にサーチし、その周波数を求めます。

用途　　BNDLと組み合わせて中心周波数を求めることができます。

例　　
```
P =PMAX(0,1200,0)
FH=BNDH(P,3,0)
FL=BNDL(P,3,0)
FB=FH-FL
FC=(FL+FH)*0.5
```

(4) 帯域幅を求める関数 CBND

CBND(周波数, 減衰レベル, 解析チャンネル)

説明 指定周波数から、指定された減衰レベルだけ減衰したポイントをサーチし、帯域幅を求めます。サーチは指定アドレス・ポイントから外側に行います。

用途 3dB ダウンの帯域幅などを求めます。

例 F =BND(F,3,0)

3dB ダウンの帯域幅を求めます。

(5) 帯域幅の低周波数側の周波数を求める関数 CBNDL

CBNDL(周波数, 減衰レベル, 解析チャンネル)

説明 指定周波数から、指定された減衰レベルだけ減衰したポイントを低周波数側にサーチし、その周波数を求めます。

用途 CBNDH と組み合わせて中心周波数を求めることができます。

(6) 帯域幅の高周波数側の周波数を求める関数 CBNDH

CBNDH(周波数, 減衰レベル, 解析チャンネル)

説明 指定周波数から、指定された減衰レベルだけ減衰したポイントを高周波数側にサーチし、その周波数を求めます。

用途 CBNDL と組み合わせて中心周波数を求めることができます。

例
```
FH =CBNDH(F,3,0)
FL =CBNDL(F,3,0)
FB =FH-FL
FC =(FL+FH)*0.5
```

(7) 複数の減衰レベルの帯域幅解析を行う関数 MBNDI

MBNDI(開始ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 終了ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 基準ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 減衰ﾚﾍﾞﾙの 数, 減衰ﾚﾍﾞﾙの配列, 帯域幅などの解析結果を格納する配列, 解析ﾁｬﾝﾈﾙ)

説明　複数の減衰レベルの解析を一度に行います。1 つの減衰レベルに対し、低周波数側の周波数、高周波数側の周波数、中心周波数、帯域幅の 4つを出力します。
減衰レベルは配列で指定し、解析結果も配列に格納されます。サーチは、指定アドレス・ポイントから外側に向かって行います。減衰レベルの配列は、レベルの小さい順になっている必要があります。

用途　複数の減衰レベルで解析を行うときに、高速処理できます。減衰レベルが 1 つのときでも、4 つの周波数が必要なときに便利です。

例
```
DIM L(3),F(3,4)
L(1) =1.0
L(2) =3.0
L(3) =10.0
P    =PMAX(0,1200,0)
N    =MBNDI(0,1200,P,3,L(1),F(1,1),0)
```

このとき配列 Fには、以下のものが格納されます。

```
F(1,1)  減衰レベル1.0 のときの低周波数側周波数
F(1,2)  減衰レベル1.0 のときの高周波数側周波数
F(1,3)  減衰レベル1.0 のときの中心周波数
F(1,4)  減衰レベル1.0 のときの帯域幅
F(2,1)  減衰レベル3.0 のときの低周波数側周波数
F(2,2)  減衰レベル3.0 のときの高周波数側周波数
F(2,3)  減衰レベル3.0 のときの中心周波数
F(2,4)  減衰レベル3.0 のときの帯域幅
F(3,1)  減衰レベル10.0のときの低周波数側周波数
F(3,2)  減衰レベル10.0のときの高周波数側周波数
F(3,3)  減衰レベル10.0のときの中心周波数
F(3,4)  減衰レベル10.0のときの帯域幅
```

なお、サーチできなかった場合には0.0 が入ります。N には、サーチできた減衰レベルの数が入ります。

(8) 複数の減衰レベルの帯域幅解析を行う関数 MBNDO

```
MBNDO(開始ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 終了ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 基準ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 減衰ﾚﾍﾞﾙの数,
      減衰ﾚﾍﾞﾙの配列, 帯域幅などの解析結果を格納する配列, 解析ﾁｬﾝﾈﾙ)
```

説明 機能は MBNDIと同様ですが、サーチは外側から内側へ行います。

用途 サーチを外側から内側に行うときに使用します。

例
```
DIM L(3),F(3,4)
L(1) =1.0
L(2) =3.0
L(3) =10.0
P    =PMAX(0,1200,0)
N    =MBNDO(0,1200,P,3,L(1),F(1,1),0)
```

このとき配列 Fに格納されるのは、 MBNDIのときと同じです。

4.4.8 リップル解析関数-1

(1) 最大の極大値と最小の極小値の差を求める関数 RPL1

```
RPL1(開始ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 終了ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 横軸の傾き係数, 縦軸の傾き係数,
     解析ﾁｬﾝﾈﾙ)
```

説明 横軸と縦軸の傾き係数に従って極大、極小を指定アドレス・ポイント間で検出し、最大の極大値と最小の極小値の差を求めます。

用途 測定対象のリップルを解析します。

例
```
X =RPL1(0,1200,1,0.5,0)
```

1 ポイントあたり0.5dB 上がる/ 下がるものをリップルとし、その最大と最小の差を求めます。

(2) 隣接する極大値と極小値の差の最大値を求める関数 RPL2

```
RPL2(開始ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 終了ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 横軸の傾き係数, 縦軸の傾き係数,
     解析ﾁｬﾝﾈﾙ)
```

説明 横軸と縦軸の傾き係数に従って極大、極小を指定アドレス・ポイント間で検出します。隣接する極大値と極小値の差を求め、さらにその最大値を求めます。隣接する極大、極小では、極大の方が低周波数側になります。

用途 測定対象のリップルを解析します。

例　　P =PMAX(0,1200,0)
X =RPL2(0,P,1,0.5,0)

1 ポイントあたり0.5dB 上がる/ 下がるものをリップルとし、ピーク点の左側で隣接する極大、極小の差の最大値を求めます。

(3) 隣接する極大値と極小値の差を加算した値の最大値を求める関数 RPL3

RPL3(開始ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 終了ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 横軸の傾き係数, 縦軸の傾き係数, 解析ﾁｬﾝﾈﾙ)

説明　　横軸と縦軸の傾き係数に従って極大、極小を指定アドレス・ポイント間で検出します。隣接する極大値と極小値の差、極小値と極大値の差、を加算し、その最大値を求めます。

用途　　測定対象のリップルを解析します。

例　　X =RPL3(0,1200,1,0.5,0)

1 ポイントあたり0.5dB 上がる/ 下がるものをリップルとし、解析します。

(4) 隣接する極大値と極小値の差の最大値を求める関数 RPL4

RPL4(開始ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 終了ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 横軸の傾き係数, 縦軸の傾き係数, 解析ﾁｬﾝﾈﾙ)

説明　　横軸と縦軸の傾き係数に従って極大、極小を指定アドレス・ポイント間で検出し、隣接する極大値と極小値の差を求め、その最大値を求めます。隣接する極大・極小では、極大の方が高周波数側になります。RPL2とは、極大、極小のペアの取り方が逆になります。

用途　　測定対象のリップルを解析します。

例　　P =PMAX(0,1200,0)
X =RPL4(P,1200,1,0.5,0)

1 ポイントあたり0.5dB 上がる/ 下がるものをリップルとし、ピーク点の右側で隣接する極大、極小の差の最大値を求めます。

(5) 極大値の最大値を求める関数 RPL5

```
RPL5(開始ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 終了ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 横軸の傾き係数, 縦軸の傾き係数,
     解析ﾁｬﾝﾈﾙ)
```

説明 横軸と縦軸の傾き係数に従って極大を指定アドレス・ポイント間で検出し、最大の極大値を求めます。

用途 測定対象のリップル・スプリアスを解析します。

例 X =RPL5(P0,P1,1,0.5,0)

1 ポイントあたり0.5dB 上がる/ 下がるものをリップルとし、最大の極大値を求めます。

(6) 極大値の最小値を求める関数 RPL6

```
RPL6(開始ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 終了ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 横軸の 傾き係数, 縦軸の傾き係数,
     解析ﾁｬﾝﾈﾙ)
```

説明 横軸と縦軸の傾き係数にしたがって極大を指定アドレス・ポイント間で検出し、最小の極大値を求めます。

用途 測定対象のリップル・スプリアスを解析します。

例 X =RPL6(P0,P1,1,0.5,0)

1 ポイントあたり0.5dB 上がる/ 下がるものをリップルとし、最小の極大値を求めます。

(7) 極大、極小の周波数差を求める関数 RPLF

```
RPLF(開始ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 終了ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 横軸の傾き係数, 縦軸の傾き係数,
     解析ﾁｬﾝﾈﾙ)
```

説明 横軸と縦軸の傾き係数に従って極大を指定アドレス・ポイント間で検出し、最初に見つかった極大値と次の極小値の周波数差を求めます。

用途 測定対象のリップルを解析します。

例 X =RPLF(P0,P1,1,0.5,0)

1 ポイントあたり0.5dB 上がる/ 下がるものをリップルとし、極大、極小の周波数差を求めます。

(8)　極大、極小のレスポンス差を求める関数　RPLR

RPLR(開始ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ,　終了ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ,　横軸の傾き係数, 縦軸の傾き係数,
解析ﾁｬﾝﾈﾙ)

説明　横軸と縦軸の傾き係数に従って極大を指定アドレス・ポイント間で検出し、最初に見つかった極大値と次の極小値のレスポンス差を求めます。

用途　測定対象のリップルを解析します。

例　X =RPLR(P0, P1, 1, 0.5, 0)

1 ポイントあたり0.5dB 上がる/ 下がるものをリップルとし、極大、極小のレスポンス差を求めます。

(9)　極大値のレスポンス値を求める関数　RPLH

RPLR(開始ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ,　終了ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ,　横軸の傾き係数, 縦軸の傾き係数,
解析ﾁｬﾝﾈﾙ)

説明　横軸と縦軸の傾き係数に従って極大を指定アドレス・ポイント間で検出し、最初に見つかった極大値のレスポンス値を求めます。

用途　測定対象のリップルを解析します。

例　X =RPLH(P0, P1, 1, 0.5, 0)

1 ポイントあたり0.5dB 上がる/ 下がるものをリップルとし、極大のレスポンス値を求めます。

(10)　極大値の周波数を求める関数　FRPLH

FRPLH(開始ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ,　終了ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ,　横軸の傾き係数, 縦軸の傾き係数,
解析ﾁｬﾝﾈﾙ)

説明　横軸と縦軸の傾き係数に従って極大を指定アドレス・ポイント間で検出し、最初に見つかった極大値の周波数を求めます。

用途　測定対象のリップルを解析します。

例　X =FRPLH(P0, P1, 1, 0.5, 0)

1 ポイントあたり0.5dB 上がる／下がるものをリップルとし、極大の周波数を求めます。

(11) 極大値の測定ポイントを求める関数 PRPLH

```
PRPLH(開始アドレス・ポイント, 終了アドレス・ポイント, 横軸の傾き係数, 縦軸の傾き係数,
      解析チャンネル)
```

説明 横軸と縦軸の傾き係数に従って極大を指定アドレス・ポイント間で検出し、最初に見つかった極大値の測定ポイントを求めます。

用途 測定対象のリップルを解析します。

例 X =PRPLH(P0,P1,1,0.5,0)

1 ポイントあたり0.5dB 上がる/ 下がるものをリップルとし、極大の測定ポイントを求めます。

(12) 極小値のレスポンス値を求める関数 RPLL

```
RPLL(開始アドレス・ポイント, 終了アドレス・ポイント, 横軸の傾き係数, 縦軸の傾き係数,
     解析チャンネル)
```

説明 横軸と縦軸の傾き係数に従って極小を指定アドレス・ポイント間で検出し、最初に見つかった極小値のレスポンス値を求めます。

用途 測定対象のリップルを解析します。

例 X =RPLL(P0,P1,1,0.5,0)

1 ポイントあたり0.5dB 上がる/ 下がるものをリップルとし、極小のレスポンス値を求めます。

(13) 極小値の周波数を求める関数 FRPLL

```
FRPLL(開始アドレス・ポイント, 終了アドレス・ポイント, 横軸の傾き係数, 縦軸の傾き係数,
      解析チャンネル)
```

説明 横軸と縦軸の傾き係数に従って極小を指定アドレス・ポイント間で検出し、最初に見つかった極小値の周波数を求めます。

用途 測定対象のリップルを解析します。

例 X =FRPLL(P0,P1,1,0.5,0)

1 ポイントあたり0.5dB 上がる/ 下がるものをリップルとし、極小の周波数を求めます。

(14)　極小値の測定ポイントを求める関数　PRPLL

```
PRPLL(開始アドレス・ポイント,　終了アドレス・ポイント,　横軸の傾き係数,　縦軸の傾き係数,
      解析チャンネル)
```

説明　　横軸と縦軸の傾き係数に従って極小を指定アドレス・ポイント間で検出し、最初に見つかった極小値の測定ポイントを求めます。

用途　　測定対象のリップルを解析します。

例　　X =PRPLH(P0,P1,1,0.5,0)

1 ポイントあたり0.5dB 上がる/ 下がるものをリップルとし、極小の測定ポイントを求めます。

4.4.9　リップル解析関数-2

(1)　極大値の数を求める関数　NRPLH

```
NRPLH(開始アドレス・ポイント,　終了アドレス・ポイント,　横軸の傾き係数,　縦軸の傾き係数,
      解析チャンネル)
```

説明　　横軸と縦軸の傾き係数に従って極大を指定アドレス・ポイント間で検出し、極大値の情報を内部に保存して極大値の数を求めます。

用途　　測定対象のリップルを解析します。

例　　NH =NRPLH(0,1200,1,0.5,0)

1 ポイントあたり0.5dB 上がる/ 下がるものをリップルとし、極大値の個数を求めます。

(2)　極小値の数を求める関数　NRPLL

```
NRPLL(開始アドレス・ポイント,　終了アドレス・ポイント,　横軸の傾き係数,　縦軸の傾き係数,
      解析チャンネル)
```

説明　　横軸と縦軸の傾き係数に従って極小を指定アドレス・ポイント間で検出し、極小値の情報を内部に保存して極小値の数を求めます。

用途　　測定対象のリップルを解析します。

例　　NL =NRPLL(0,1200,1,0.5,0)

1 ポイントあたり0.5dB 上がる/ 下がるものをリップルとし、極小値の個数を求ます。

(3) 極大値、極小値の測定ポイントを求める関数 PRPLHN, PRPLLN

```
PRPLHN(リップルの番号指定, 解析チャンネル)
PRPLLN(リップルの番号指定, 解析チャンネル)
```

説明 PRPLHN : NRPLHで求めたN 番目の極大値の測定ポイントを求めます。
PRPLLN : NRPLLで求めたN 番目の極小値の測定ポイントを求めます。

例
```
NH  =NRPLH(0,1200,1,0.5,0)
NL  =NRPLL(0,1200,1,0.5,0)
PH2 =PRPLHN(2,0)
PL2 =PRPLLN(2,0)
```

NRPLH, NRPLL を実行し、2 番目の極大値・極小値の測定ポイントを求めます。

(4) 極大値、極小値の周波数を求める関数 FRPLHN, FRPLLN

```
FRPLHN(リップルの番号指定, 解析チャンネル)
FRPLLN(リップルの番号指定, 解析チャンネル)
```

説明 FRPLHN : NRPLHで求めたN 番目の極大値の周波数を求めます。
FRPLLN : NRPLLで求めたN 番目の極小値の周波数を求めます。

用途 測定対象のリップルを解析します。

例
```
NH  =NRPLH(0,1200,1,0.5,0)
NL  =NRPLL(0,1200,1,0.5,0)
FH2 =FRPLHN(2,0)
FL2 =FRPLLN(2,0)
```

NRPLH, NRPLL を実行し、2 番目の極大値、極小値の周波数を求めます。

(5) 極大値、極小値のレスポンス値を求める関数 VRPLHN, VRPLLN

```
VRPLHN(リップルの番号指定, 解析チャンネル)
VRPLLN(リップルの番号指定, 解析チャンネル)
```

説明 VRPLHN : NRPLHで求めたN 番目の極大値のレスポンス値を求めます。
VRPLLN : NRPLLで求めたN 番目の極小値のレスポンス値を求めます。

用途 測定対象のリップルを解析します。

例
```
NH  =NRPLH(0,1200,1,0.5,0)
NL  =NRPLL(0,1200,1,0.5,0)
XH2 =VRPLHN(2,0)
XL2 =VRPLLN(2,0)
```

NRPLH, NRPLL を実行し、2 番目の極大値、極小値のレスポンス値を求めます。

(6) 極大値、極小値の測定ポイントを一括で求める関数 PRPLHM, PRPLLM

```
PRPLHM(整数配列,  解析ﾁｬﾝﾈﾙ)
PRPLLM(整数配列,  解析ﾁｬﾝﾈﾙ)
```

説明 PRPLHM : NRPLHで求めた極大値の測定ポイントを求めます。
PRPLLM : NRPLLで求めた極小値の測定ポイントを求めます。

用途 測定対象のリップルを解析します。

例
```
INTEGER PH(600), PL(600)
NH =NRPLH(0, 1200, 1, 0.5, 0)
NL =NRPLL(0, 1200, 1, 0.5, 0)
NH =PRPLHM(PH(1), 0)
NL =PRPLLM(PL(1), 0)
```

NRPLH, NRPLL を実行し、極大値、極小値の測定ポイントを配列に取り込みます。

(7) 極大値、極小値の周波数を一括で求める関数 FRPLHM, FRPLLM

```
FRPLHM(実数配列,  解析ﾁｬﾝﾈﾙ)
FRPLLM(実数配列,  解析ﾁｬﾝﾈﾙ)
```

説明 FRPLHM : NRPLHで求めた極大値の周波数を求めます。
FRPLLM : NRPLLで求めた極小値の周波数を求めます。

用途 測定対象のリップルを解析します。

例
```
DIM FH(600), FL(600)
NH =NRPLH(0, 1200, 1, 0.5, 0)
NL =NRPLL(0, 1200, 1, 0.5, 0)
NH =FRPLHM(FH(1), 0)
NL =FRPLLM(FH(1), 0)
```

NRPLH, NRPLL を実行し、極大値、極小値の周波数を配列に取り込みます。

(8) 極大値、極小値のレスポンス値を一括で求める関数 VRPLHM,VRPLLM

```
VRPLHM(実数配列,  解析ﾁｬﾝﾈﾙ)
VRPLLM(実数配列,  解析ﾁｬﾝﾈﾙ)
```

説明 VRPLHM : NRPLHで求めた極大値のレスポンス値を求めます。
VRPLLM : NRPLLで求めた極小値のレスポンス値を求めます。

用途 測定対象のリップルを解析します。

例
```
DIM XH(600),XL(600)
NH =NRPLH(0,1200,1,0.5,0)
NL =NRPLL(0,1200,1,0.5,0)
NH =VRPLHM(XH(1),0)
NL =VRPLLM(XL(1),0)
```

NRPLH,NRPLL を実行し、極大値、極小値のレスポンス値を配列に取り込みます。

4.4.10 ダイレクト・サーチ

(1) 指定レスポンスに対応するアドレス・ポイントを求める関数 DIRECT

```
DIRECT(開始ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ,  終了ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ,  ﾚｽﾎﾟﾝｽ値,  解析ﾁｬﾝﾈﾙ)
```

説明 指定アドレス・ポイント間で、指定レスポンス値をサーチし、対応するアドレス・ポイントを求めます。サーチの方向は低周波から高周波です。

例
```
P =DIRECT(0,1200,-10.0,0)
```

-10dB のデータの位置を探します。

(2) 指定レスポンスに対応する測定ポイントを求める関数 DIRECTL,DIRECTH

```
DIRECTL(開始ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ,  終了ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ,  ﾚｽﾎﾟﾝｽ値,  解析ﾁｬﾝﾈﾙ)
DIRECTH(開始ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ,  終了ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ,  ﾚｽﾎﾟﾝｽ値,  解析ﾁｬﾝﾈﾙ)
```

説明 指定アドレス・ポイント間で指定レスポンス値をサーチし、対応する測定ポイントを求めます。サーチの方向は低周波から高周波が DIRECTL、高周波から低周波が DIRECTHです。指定レスポンスに一致したレスポンスがあったときは、その測定ポイントを返します。一致しないときは指定レスポンス値を超えた測定ポイントを返します。そのため、連続サーチが行いやすくなっています。

例
```
P0 =DIRECTL(0,1200,-3.0,0)
P1 =DIRECTH(0,1200,-3.0,0)
F  =DFREQ(P0,P1,0)
```

外側からサーチして帯域幅を求めます。

(3) 指定レスポンスに対応する周波数を求める関数 CDIRECT

```
CDIRECT(開始周波数, 終了周波数, レスポンス値, 解析チャンネル)
```

説明 指定周波数間で、指定レスポンス値をサーチし、対応する周波数を求めます。サーチの方向は低周波から高周波です。

例
```
F =CDIRECT(F0,F1,-10.0,0)
```

-10dB のデータの周波数を求めます。

(4) 指定レスポンスに対応する周波数を求める関数 CDIRECTL,CDIRECTH

```
CDIRECTL(開始周波数, 終了周波数, レスポンス値, 解析チャンネル)
CDIRECTH(開始周波数, 終了周波数, レスポンス値, 解析チャンネル)
```

説明 指定周波数間で、指定レスポンス値をサーチし、対応する周波数を求めます。サーチの方向は低周波から高周波がCDIRECTL、高周波から低周波がCDIRECTHです。

例
```
F0 =CDIRECTL(F0,F1,-3.0,0)
F1 =CDIRECTH(F0,F1,-3.0,0)
F  =F1-F0
```

外側からサーチして帯域幅を求めます。

(5) 指定レスポンスのアドレス・ポイント幅を求める関数 DDIRECT

```
DDIRECT(開始ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 終了ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, レスポンス値, 解析チャンネル)
```

説明 指定アドレス・ポイント間で、指定レスポンス値を高周波数側へサーチし、検出した 2つの測定ポイントからアドレス・ポイント幅を求めます。

(6) 指定レスポンスの帯域幅を求める関数 CDDIRECT

```
CDDIRECT(開始周波数, 終了周波数, レスポンス 値, 解析チャンネル)
```

説明 指定周波数間で、指定レスポンス値を高周波数側へサーチし、検出した 2つの測定ポイントから帯域幅を求めます。

(7) ゼロ位相の周波数を求める関数 ZEROPHS

ZEROPHS(開始ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 終了ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, ﾚｽﾎﾟﾝｽ値, 解析ﾁｬﾝﾈﾙ)

説明 指定アドレス・ポイント間で位相ゼロを検出し、その周波数を求めます。

4.4.11 データ転送

(1) 指定解析チャンネルのデータを配列に読み込む関数 TRANSR

TRANSR(開始ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 終了ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 実数配列, 解析ﾁｬﾝﾈﾙ)

説明 指定した解析チャンネルの測定データを、アドレス・ポイントを指定して BASIC の配列に読み込み、データ数を返します。

用途 測定データを２次処理するときに使用します。

例
```
DIM X(1201)
N =TRANSR(0,1200,X(1),0)
```

(2) 指定解析チャンネルに配列の内容を書き込む関数 TRANSW

TRANSW(開始ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 終了ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾄ, 実数配列, 解析ﾁｬﾝﾈﾙ)

説明 指定した解析チャンネルに、BASIC の配列の内容を書き込みます。

用途 測定データを２次処理するときに使用します。

例
```
DIM X(1201)
N =TRANSW(0,1200,X(1),0)
```

5. パラレルI/O ポート

5.1 概要

パラレルI/O ポートは、ハンドラおよび周辺機器と通信するためのI/O(インプット/アウトプット) ポートです。
通信は、背面パネルのパラレルI/O コネクタを用いて行います。[図 5-1] にコネクタの内部ピン配置と信号を示してあります。これらのI/O ポートのコントロールは、ENTER とOUTPUTを用いて行われます。

●入出力ポート

出力ポートと入出力ポートが 2組づつあります。

・出力専用ポート : A ポート ; 8ビット幅
B ポート ; 8ビット幅

・入出力ポート : C ポート ; 4ビット幅
D ポート ; 4ビット幅

●ポートC ステータス出力、ポートD ステータス出力

入出力ポートC,D の入力の設定状態を示します。
C,D ポートが入力に設定されているときにLOW 、出力に設定されているときにHIGHになります。

●出力ポート用ライト・ストローブ出力

このライト・ストローブ出力に負パルスを出力することにより、出力ポートのいずれかにデータが出力されていることを示します。
[図5-1] にライト・ストローブ出力とデータ出力のタイミング・チャートを示します。

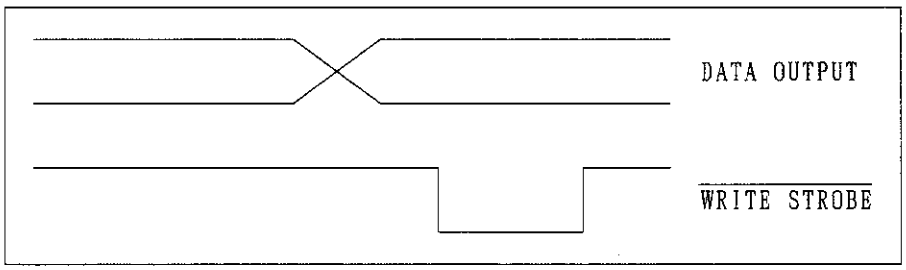


図 5 - 1 WRITE STROBE のタイミング・チャート

● INPUT 1 入力

この入力に負パルスを入力することにより、OUTPUT 1およびOUTPUT 2の出力状態をLOW にします。INPUT 1 に入力する信号のパルス幅は 1μs 以上必要です。

● OUTPUT 1出力、OUTPUT 2出力

この2 つの信号ラインは、INPUT 1 への負パルス入力によりLOW にセットされるラッチ出力端子です。BASIC コマンド(OUTPUT)によりLOW またはHIGHにセットすることができます。

● PASS/FAIL 出力

リミット・テストの結果がPASSのときLOW 、FAILのときHIGHの信号を発生します。
リミット・テスト機能がONのときのみ有効です。

● PASS/FAIL 出力用ライト・ストローブ出力

PASS/FAIL 出力ラインにリミット・テストの結果が出力されると、負パルスが出力されます。

● SWEEP END

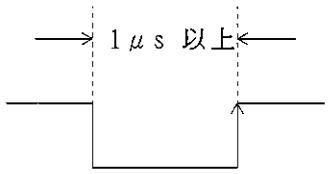
本器が掃引を終了したときに、負パルスを出力します。パルス幅は10μs です。

● +5V 出力

外部機器のために+5V 出力が用意されています。供給可能な最大電流は100mA です。このラインにはヒューズがあり、過電流が流れた場合遮断され、回路は保護されますが交換が必要です。

● EXT TRIG入力

この入力に負パルスを入力することにより、掃引測定のトリガをかけることができます。パルス幅は 1μs 以上必要で、パルスの立ち上がりエッジで掃引を開始します。この信号ラインを使用する場合は、トリガ・ソースを外部(External)に設定します。



5.2　コネクタの内部ピン配置と信号規格

ピンNo.	信号名称	機能
1	GND	グランド
2	INPUT 1	TTL レベルの負論理パルス入力（幅 1μs 以上）
3	OUTPUT 1	TTL レベルの負論理ラッチ出力
4	OUTPUT 2	TTL レベルの負論理ラッチ出力
5	出力ポート A0	TTL レベルの負論理ラッチ出力
6	出力ポート A1	TTL レベルの負論理ラッチ出力
7	出力ポート A2	TTL レベルの負論理ラッチ出力
8	出力ポート A3	TTL レベルの負論理ラッチ出力
9	出力ポート A4	TTL レベルの負論理ラッチ出力
10	出力ポート A5	TTL レベルの負論理ラッチ出力
11	出力ポート A6	TTL レベルの負論理ラッチ出力
12	出力ポート A7	TTL レベルの負論理ラッチ出力
13	出力ポート B0	TTL レベルの負論理ラッチ出力
14	出力ポート B1	TTL レベルの負論理ラッチ出力
15	出力ポート B2	TTL レベルの負論理ラッチ出力
16	出力ポート B3	TTL レベルの負論理ラッチ出力
17	出力ポート B4	TTL レベルの負論理ラッチ出力
18	EXT TRIG	EXTERNAL TRIGGER入力（パルス幅 1μs 以上）、負論理
19	出力ポート B5	TTL レベルの負論理ラッチ出力
20	出力ポート B6	TTL レベルの負論理ラッチ出力
21	出力ポート B7	TTL レベルの負論理ラッチ出力
22	入出力ポート C0	TTL レベルの負論理ステート入力/ ラッチ出力
23	入出力ポート C1	TTL レベルの負論理ステート入力/ ラッチ出力
24	入出力ポート C2	TTL レベルの負論理ステート入力/ ラッチ出力
25	入出力ポート C3	TTL レベルの負論理ステート入力/ ラッチ出力
26	入出力ポート D0	TTL レベルの負論理ステート入力/ ラッチ出力
27	入出力ポート D1	TTL レベルの負論理ステート入力/ ラッチ出力
28	入出力ポート D2	TTL レベルの負論理ステート入力/ ラッチ出力
29	入出力ポート D3	TTL レベルの負論理ステート入力/ ラッチ出力
30	ポートC ｽﾃｰﾀｽ	TTL レベル、入力モード : LOW、出力モード : HIGH
31	ポートD ｽﾃｰﾀｽ	TTL レベル、入力モード : LOW、出力モード : HIGH
32	ﾗｲﾄ･ｽﾄﾛｰﾌﾞ 信号	TTL レベル、負論理、パルス出力
33	PASS/FAIL 信号	TTL レベル、PASS : LOW、FAIL : HIGH 、ラッチ出力
34	SWEEP END 信号	TTL レベル、負論理、パルス出力（幅10μs 以上）
35	+5V	+5V 100mA MAX
36	ﾗｲﾄ･ｽﾄﾛｰﾌﾞ 信号 (PASS/FAIL用)	TTL レベル、負論理、パルス出力

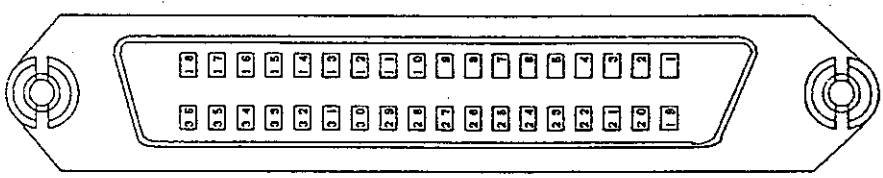
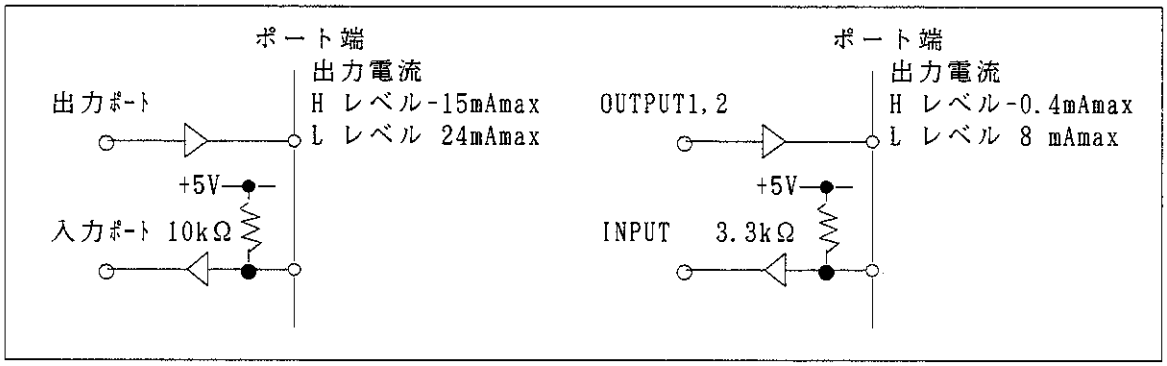


図 5 - 1　36ピン・コネクタの内部ピン配置と信号



5.3　ポートのモード設定

コマンド	出力ポート	入力ポート
OUTPUT 36；16	A, B, C, D	
OUTPUT 36；17	A, B, D	C
OUTPUT 36；18	A, B, C	D
OUTPUT 36；19	A, B	CD

パラレルI/O を使用するには、まずポートのモード設定をします。設定コマンドおよび入出力ポートは上の表の組合せになります。

例
```
10  OUTPUT 36；19
20  OUTPUT 33；255
30  ENTER 37；A
      ⋮
```

出力ポートをA, Bポート、入力ポートをCDポートにします。

5.4　各ポートの操作方法

R3752/3753の内蔵 BASICによる操作方法を説明します。
データの入出力には、OUTPUT文（出力）、ENTER文（入力）を使用します。
各ポートと BASICコマンドとの関係には、それぞれOUTPUT文、ENTER 文に使用されるアドレスを区別することにします。

(1)　BASIC 書式

OUTPUT（アドレス）；（データ）

ENTER（アドレス）；［変数名］

（入力データは変数名の数値となる）

(2)　アドレスおよびデータ範囲

アドレス	使用ポート
33	Aポート（出力専用：OUTPUT文のみ）
34	Bポート（出力専用：OUTPUT文のみ）
35	Cポート（入出力：ENTER，OUTPUT）
36	Dポート（入出力：ENTER，OUTPUT）
37	CDポート（入出力：ENTER，OUTPUT）

- OUTPUT 33，34，37

  OUTPUT ××；0～255（8bit）

- OUTPUT 35，36

  OUTPUT ××；0～15（4bit）

  注）OUTPUT 35 は Flip Flopの Set/Resetにも関与します。
  （後述 Flip Flop部）

- ENTER 35，36

  ENTER ××；数値変数（4bit）（0～15までのデータが代入される）

- ENTER 37

  ENTER 37；数値変数（8bit）（0～255までのデータが代入される）

5.5　INPUT 1, OUTPUT 1, OUTPUT 2 端子について

INPUT 1,OUTPUT 1,OUTPUT 2 の信号ラインを組み合わせて用いることにより、外部機器の制御を容易に行う機能が用意されています。これは、OUTPUT 1,2の2 つのラッチ出力をINPUT 1 へのパルス入力によりLOW にセットする機能と、INPUT 1 により変化するOUTPUT 1の状態を検出する機能です。また、OUTPUT 1,2の状態をOUTPUTコマンドによりコントロールできます。

(1)　OUTPUT 1, OUTPUT 2のセットおよびリセット

セットとリセットは 1と2 が別々に行われるので 4通りとなります。

- OUTPUT 1のセット　：　OUTPUT 35 ; 16
- OUTPUT 2のセット　：　OUTPUT 35 ; 48
- OUTPUT 1のリセット ：　OUTPUT 35 ; 80
- OUTPUT 2のリセット ：　OUTPUT 35 ; 112

(2)　INPUT 1（外部入力）

INPUT 1 により変化するOUTPUT 1の状態を、ENTER 文で見ることができます。

ENTER 34 ; （数値変数）

数値変数が 1であるとOUTPUT 1がON(Low Level…負論理であるため）で、0 であるとOFF(High Level) となっています。

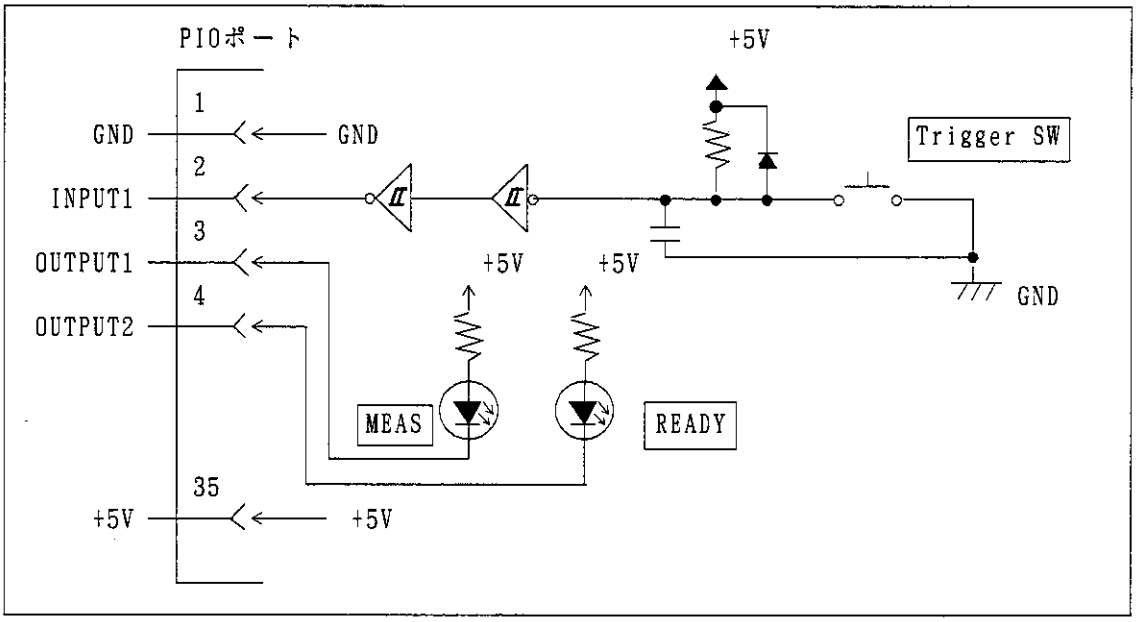
例
```
10  OUTPUT 36 ; 16
20  ENTER 34 ; A
30  IF A <> 1 THEN GOTO 20
40  OUTPUT 33 ; 1
        ⋮
```

OUTPUT 1の状態を見て、OUTPUT 1がONであったならば、そのあとA ポートに 1を出力します。

① INPUT1, OUTPUT1, OUTPUT2の使用例

トリガ・スイッチによってプログラムを動作させる場合

・回路例



・プログラム例

測定開始待ち : READY とします。

測定中 : MEAS とします。

```
10 OUTPUT 35;80
20 OUTPUT 35;112
 :
 :
 :
100 OUTPUT 35;48
110 ENTER 34;A
120 IF A<>1 THEN GOTO 110
130 OUTPUT 35;112
 :
 :
 :
500 OUTPUT 35;80
510 GOTO 100
520 STOP
```

READY MEAS OFFする

ネットワーク・アナライザ初期設定

READY ONする

Trigger SWの認識

READY OFFする

測定ルーチン

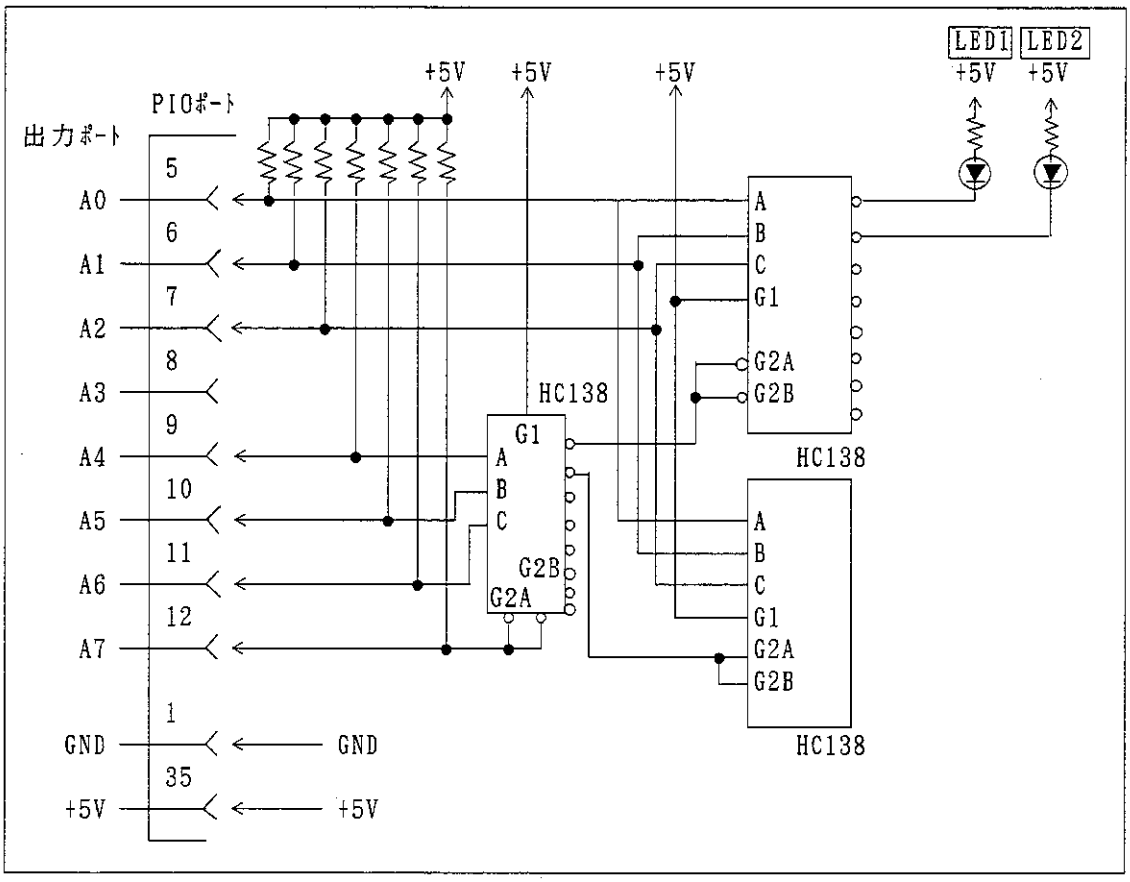
MEAS OFFする

測定を繰り返す場合

② 出力ポートＡおよびＢの使用例

デバイスの選別をLED を使って行う場合（A ポート使用時）

・回路例



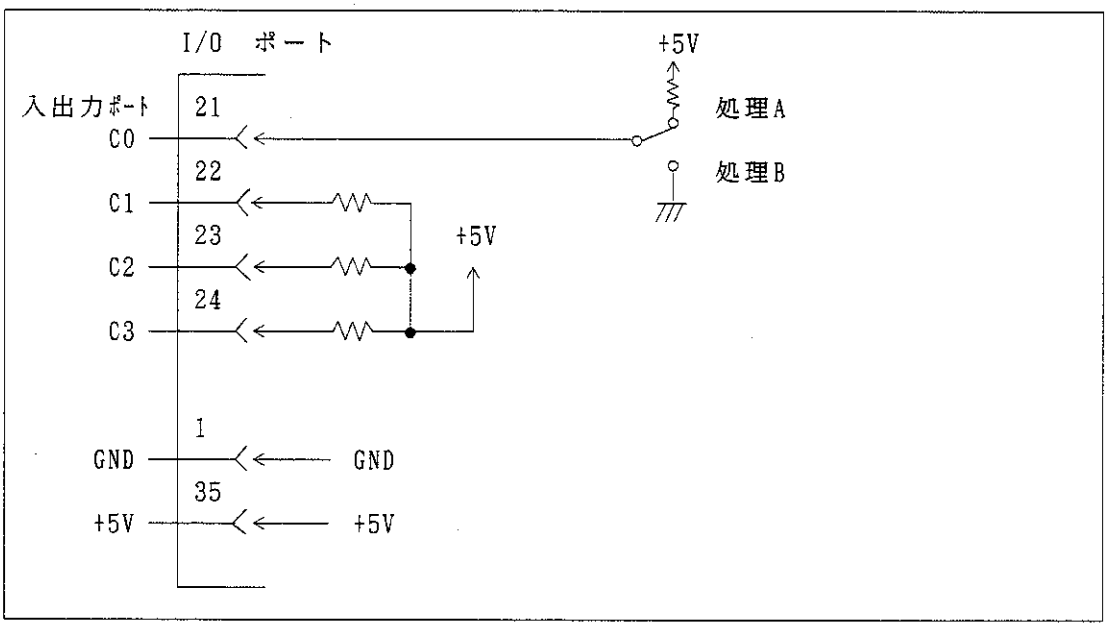
・プログラム例

```
 10 OUTPUT 36;16              A,B,C,D ポートを出力ポートとする
 20 OUTPUT 33;0               LED を初期化する
 30
  ⋮                           測定および判定
  ⋮                            (判定変数；A
  ⋮                            (判定範囲;JED0 ～JED1, JED1～JED2…)
  ⋮
500 IF A>=JED0 AND A<JED1 THEN OUTPUT 33;0xFF
                              (JED0 ～JED1の場合はLED1を点灯させる)
510 IF A>=JED1 AND A<JED2 THEN OUTPUT 33;0xFB
                              (JED1 ～JED2の場合はLED2を点灯させる)
  ⋮
800 GOTO 30
810 STOP
```

③　入出力ポートC およびD の使用例

入出力ポートC のビット0 が、0 か1 かによって処理ルーチンを変える例

・回路例



・プログラム例（①の Trigger SW を押してC ポートをチェックする）

```
 10 OUTPUT 36;19                    A,B ポートを出力ポートとする
 20 OUTPUT 35;80                    C,D ポートを入力ポートとする
 30 OUTPUT 35;112
  :
                                    ネットワーク・アナライザ初期設定
100 *TRIG
110 ENTER 34;A
120 IF A<>1 THEN GOTO *TRIG
130 ENTER 35;B                      C ポートの値をとる
140 IF B=1 THEN GOTO *ROUT_B
150 *ROUT_A
  :
                                    処理A
490 GOTO *TRIG
500 *ROUT_B
  :
                                    処理B
900 GOTO *TRIG
910 STOP
```

MEMO

6.　エラー・メッセージ

6.1　エラー・メッセージを知る方法

PRINT ERRM$(0)を実行すると、エラーメッセージとエラーを発生した行番号が表示されます。

6.2　プログラムの現在位置を知る方法

@ はシステム変数で、プログラムを実行している行番号が格納されています。
これを利用すると、現在の行番号、プログラムの現在位置、プログラムの停止位置などを知ることができます。

例）PRINT @ …… プログラムの停止位置を表示します。

6.3　エラー・メッセージ一覧

注1）表はエラー・メッセージをエラー・クラス（エラー番号）順にしています。
（一覧表の後に、アルファベット順対応表もあります。）
文字列をXXX と表します。
数値をYYY と表します。

注2）エラー・クラス内容：1；データ入出力関係
2；データ演算処理関係
3；ビルトイン関数関係
4；BASIC 構文関係
5；その他

(1/5)

ｴﾗｰ･ｸﾗｽ (ｴﾗｰ番号)	エラー・メッセージ	内容
1(22)	xxx1(xxx2) error	xxx2のファイルに対してxxx1の命令が実行できない
1(23)	xxx1(xxx2, xxx3) error	xxx2, xxx3のファイルに対してxxx1の命令が実行できない
1(64)	"xxx" file cannot be opened.	オープンしようとするファイルがない（オープンできない）
1(65)	xxx: "xxx" file was opened with xxx mode.	オープンしたときのアクセス・モードと違うアクセスを行った
1(66)	cannot read data from "xxx" file.	xxx ファイルから指定した文字数を読み込めない
1(67)	cannot write data into "xxx" file.	xxx ファイルにデータを書き込めない

(2/5)

エラー・クラス (エラー番号)	エラー・メッセージ	内容
1(69)	"xxx" file is already opened with another PATH.	すでにオープンしてあるファイルをオープンしようとした
1(72)	file is NOT open.	指定のディスクリプタにファイルが登録されていない（ファイルがオープンされていない）
1(74)	end of "xxx" file.	EOF(End Of File)まで読み込んだ
1(75)	"xxx" file already exists.	存在しているファイルをOUTPUTモードでオープンしようとした
1(77)	Already 8 files are opened.	8 つを越えてファイルをオープンしようとした
1(79)	CANNOT assigned into this token	文字列変数に代入できない
1(95)	GPIB SYNTAX ERROR	GPIBコマンドが間違っている
1(96)	Abort	GPIB制御ステートメントが中断された、またはGPIBのバス上でエラーが発生した
1(98)	Not controller	コントローラ・モードでのみ使用できる命令をアドレッサブル・モードで使用した
1(99)	Not Talker/Listener	アドレッサブル・モードでのみ使用できる命令をコントローラ・モードで使用した
2(1)	0 divide	0 による除算(n/0) が行われた
2(10)	xxx: CANNOT convert into string	文字列に変換できない
2(32)	string length is too long	文字列変数の宣言が多すぎる（最大128）
2(33)	Array's range error	配列変数の添字が宣言の範囲外である
2(41)	yyy: UNIT addr error in xxx	GPIBアドレス指定が違う
2(43)	yyy is invalid value in xxx	xxx の命令ではyyy が無効である

(3/5)

ｴﾗｰ･ｸﾗｽ (ｴﾗｰ番号)	エラー・メッセージ	内容
2(48)	CANNOT move line.	REN 命令にて最終行が65535 を越えるような指定をした
2(51)	Overflow value	数値が扱える範囲を越えている
2(60)	yyy: Undefined Control Register	CONTROL 命令のレジスタ番号が違う
2(63)	Unmatched DATA's values and READ variable	READ文で読み込むデータがない
2(85)	file format error	256 文字以内にターミネータがない
3(11)	xxx function error	ビルトイン関数のパラメータ・エラー
3(94)	xxx function error. ﾒｯｾｰｼﾞ	ビルトイン関数のエラー
4(2)	xxx: invalid type in xxx	xxx の中に無効なものがある
4(3)	NO operand in xxx	xxx の演算書式が違う
4(5)	Program does NOT exist	プログラムが存在しないのに実行した
4(6)	xxx: Syntax error	文法が違う
4(7)	undefined ON condition	ON条件の定義が間違っている
4(9)	xxx: Invalid TARGET operand in xxx	xxx の中の書式に無効なものがある
4(12)	Unbalanced NEXT statement	FOR 文があるのにNEXT文がない
4(13)	FOR's nest is abnormal.	FOR 文が入れ子(nest)できない
4(14)	FOR variable does NOT exist.	FOR 文のカウンタ変数がない
4(15)	FOR <init value> does NOT exist.	FOR 文の初期値がない

(4/5)

ｴﾗｰ･ｸﾗｽ (ｴﾗｰ番号)	エラー・メッセージ	内容
4(16)	Unbalanced FOR variable in NEXT	FOR 文とNEXT文の関係がおかしい
4(17)	Unbalanced BREAK	BREAK 文が FOR～NEXT間にない
4(18)	Uninstalled type (xxx)	変数の書式が間違っている
4(19)	Label xxx already exists.	xxx のラベルはすでに存在している
4(20)	Unbalanced xxx	構文上、バランスがとれない
4(21)	Not available ASCII char(yyy)	ASCII コードが有効でない
4(24)	xxx: invalid first type in xxx	コマンドの構文の最初が違う
4(25)	xxx: invalid second type in xxx	コマンドの構文の2 番目が違う
4(26)	xxx: invalid source type in xxx	式の代入でソース側のタイプが違う
4(27)	xxx: invalid target type in xxx	代入しようとする変数のタイプが違う
4(29)	Invalid dimension parameter	配列変数のパラメータが間違っている
4(31)	string declaration error	数値変数に [ ] を使った
4(34)	Unbalanced line No.	指定した行がない
4(37)	Undefined label	指定のラベルが存在しない
4(38)	label not found	指定したラベルがない
4(39)	Unknown line No.	指定行がない
4(40)	expression format error	書式の表現が間違っている

(5/5)

ｴﾗｰ･ｸﾗｽ (ｴﾗｰ番号)	エラー・メッセージ	内容
4(43)	yyy is invalid value in xxx	xxx の命令ではyyy は無効である
4(44)	Unbalanced xxx block	xxx ブロック(FOR、IF文等) が合わない
4(45)	Not found THEN in xxx	IF文の後ろにTHENがない
4(47)	Not found line No. yyy	yyy の行が存在しない
4(49)	Substring error	サブストリング指定が間違っている
4(50)	parameter error	パラメータが間違っている
4(52)	Unmatched IMAGE-spec in USING	USING のイメージ仕様が違う
4(54)	yyy error(s) appeared.	ラベル行番号のエラー
4(55)	Program CANNOT be continued.	終了したプログラムを再実行しようとした
4(56)	Line No.yyy is out of range.	行番号の指定がプログラムの範囲を越えている
4(68)	cannot specify "USING"	指定のファイルのタイプではUSING の指定はできない
4(70)	Not found DATA statement	RESTORE する先にDATA文がない
4(71)	xxx nest overflow	xxx の入れ子(nest)が多すぎる
4(78)	SELECT nesting overflow	SELECT文の入れ子(nest)が多すぎる
4(93)	Program cannot changed	プログラムRUN 中にプログラム入力しようとした

アルファベット順対応表

(1/2)

エラー・メッセージ	エラー・クラス（エラー番号）
Abort	1(96)
Already 8 files are opened.	1(77)
Array's range error	2(33)
CANNOT assigned into this token	1(79)
CANNOT move line.	2(48)
cannot read data from "xxx" file.	1(66)
cannot specify "USING"	4(68)
cannot write data into "xxx" file.	1(67)
end of "xxx" file.	1(74)
expression format error	4(40)
file format error	2(85)
file is NOT open.	1(72)
FOR <init value> does NOT exist.	4(15)
FOR variable does NOT exist.	4(14)
FOR's nest is abnormal.	4(13)
GPIB SYNTAX ERROR	1(95)
Invalid dimension parameter	4(29)
label not found	4(38)
Label xxx already exists.	4(19)
Line No.yyy is out of range.	4(56)
NO operand in xxx	4(3)
Not available ASCII char(yyy)	4(21)
Not controller	1(98)
Not found DATA statement	4(70)
Not found line No. yyy	4(47)
Not found THEN in xxx	4(45)
Not Talker/Listener	1(99)
Overflow value	2(51)
parameter error	4(50)
Program CANNOT be continued.	4(55)
Program cannot changed	4(93)
Program does NOT exist	4(5)
SELECT nesting overflow	4(78)
string declaration error	4(31)
string length is too long	2(32)
Substring error	4(49)
Unbalanced BREAK	4(17)
Unbalanced FOR variable in NEXT	4(16)
Unbalanced line No.	4(34)
Unbalanced NEXT statement	4(12)
Unbalanced xxx	4(20)
Unbalanced xxx block	4(44)

(2/2)

エラー・メッセージ	エラー・クラス (エラー番号)
Undefined label	4(37)
undefined ON condition	4(7)
Uninstalled type (xxx)	4(18)
Unknown line No.	4(39)
Unmatched DATA's values and READ variable	2(63)
Unmatched IMAGE-spec in USING	4(52)
xxx function error	3(11)
xxx function error. ﾒｯｾｰｼﾞ	3(94)
xxx nest overflow	4(71)
xxx1(xxx2) error	1(22)
xxx1(xxx2, xxx3) error	1(23)
xxx: CANNOT convert into string	2(10)
xxx: invalid first type in xxx	4(24)
xxx: invalid second type in xxx	4(25)
xxx: invalid source type in xxx	4(26)
xxx: Invalid TARGET operand in xxx	4(9)
xxx: invalid target type in xxx	4(27)
xxx: invalid type in xxx	4(2)
xxx: Syntax error	4(6)
xxx: "xxx" file was opened with xxx mode.	1(65)
"xxx" file cannot be opened.	1(64)
"xxx" file already exists.	1(75)
"xxx" file is already opened with another PATH.	1(69)
yyy error(s) appeared.	4(54)
yyy is invalid value in xxx	2(43),4(43)
yyy: Undefined Control Register	2(60)
yyy: UNIT addr error in xxx	2(41)
0 divide	2(1)

索引

— アルファベット順 —

【O】

【P】

【R】

【S】

【T】

【W】

— 50音順 —

【せ】

【そ】

【た】

【て】

【は】

【ひ】

【ふ】

【へ】

【ほ】

【め】

【り】

【れ】

【ろ】

【わ】

第 2 部

第 2 部の目次

1. はじめに

2. GPIBバスの機能

3. コマンド文法

4. ステータス・バイト

1.　はじめに

本器は、IEEE規格488.1-1987および488.2-1987に準拠したGPIB(General Purpose Interface Bus)を標準装備し、外部コントローラによるリモート・コントロールが可能です。また、内蔵コントローラ機能により小規模GPIBシステムを簡単に構築できます。

以下、GPIBリモート・コントロール機能を用いたコントロール方法について説明します。

1.1　GPIBとは

GPIB(General Purpose Interface Bus) は、コンピュータと計測器を統合する高性能のバスを提供します。
このGPIBの動作はIEEE規格488.1-1987によって定義されています。GPIBはバス構造のインタフェースのため、各機器が固有の互いに異なる機器アドレスを持つことによって、特定の機器を指定します。これらの機器は 1つのバスに15台まで並列に接続できます。GPIB機器は、以下の機能のうち 1つ以上を備えています。

・トーカ　：　バスにデータを送信するために指定された機器を「トーカ」と呼びます。GPIBバス上では、一台の機器のみがアクティブ・トーカとして動作します。

・リスナ　：　バスのデータを受信するために指定された機器を「リスナ」と呼びます。アクティブなリスナ機器はGPIBバス上に複数存在できます。

・コントローラ：　トーカ、リスナを指定する機器を「コントローラ」と呼びます。GPIBバス上では一台の機器のみがアクティブ・コントローラとして動作します。これらのコントローラのうち、IFC 、およびREN のメッセージをコントロールできる機器を特に「システム・コントローラ」と呼びます。

システム・コントローラは、GPIBバス上に一台だけ許されます。バス上に複数のコントローラがある場合、システム起動時にはシステム・コントローラがアクティブ・コントローラとなり、その他のコントローラ能力を持つ機器はアドレッサブル機器として動作します。

その他のコントローラをアクティブ・コントローラにするにはTake Control(TCT)インタフェース・メッセージを用います。そのとき自分はノンアクティブ・コントローラとなります。

コントローラはインタフェース・メッセージ、またはデバイス・メッセージを各測定器に送ってシステム全体をコントロールします。
それぞれ以下の役目を果たします。

・インタフェース・メッセージ　：　GPIBバスをコントロールする
・デバイス・メッセージ　　：　測定器をコントロールする

内蔵BASIC の取扱方法は、本書第 1部を参照して下さい。

1.2　コマンド・モード

1.2.1　IEEE488.2-1987コマンド・モード

R3764/66、R3765/67シリーズでは、 2つのコマンド・モードでの動作が可能です。

・IEEE規格488.2-1987コマンド・モード
・IEEE規格488.1-1987コマンド・モード

R3762/63シリーズは、IEEE規格488.1-1987コマンド・モードのみ動作可能です。

この488.2-1987は、以下に示すものを488.1-1987に拡張して定義しています。

・測定器をプログラミングする際の文法
・コマンドとデータの通信プロトコル（手順）
・共通コマンド *
・ステータスデータ構造
・システムの同期プロトコル

* : 共通コマンドとは、すべての測定器で同じ動作をするコマンドのことです。

1.2.2　IEEE488.1-1987コマンド・モード

IEEE488.1-1987コマンド・モードでは、コマンドの文法および通信の手順に関して、R3762/63シリーズとのコンパチビリティを保ち、R3764/66、R3765/67シリーズへのスムーズな移行を可能にします（ただし、製品仕様の変更に伴い、一部動作の違うコマンドがあります）。

1.2.3　コマンド・モードの切り換え

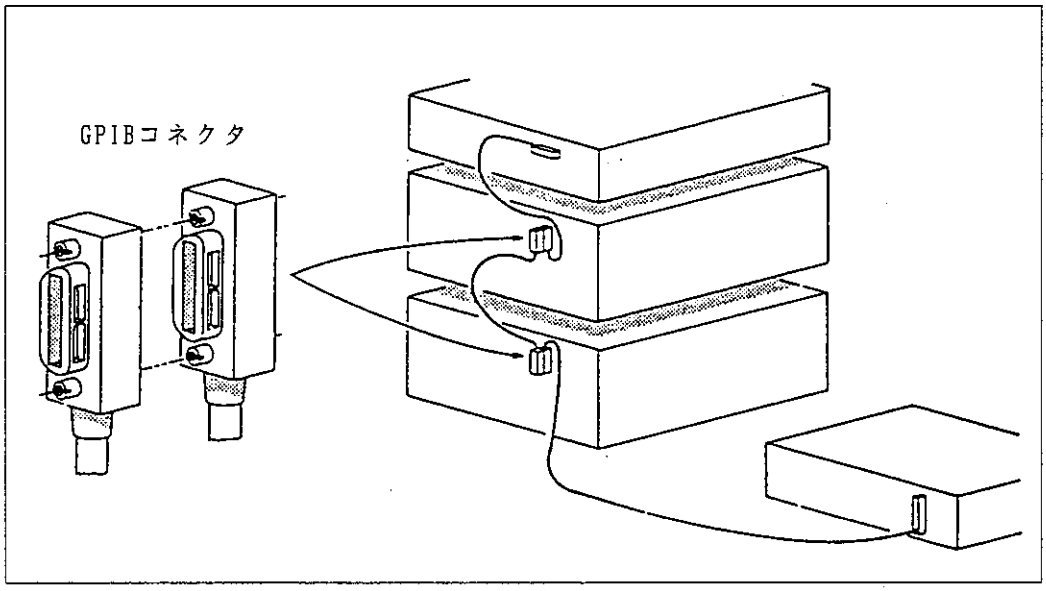
本器の起動時は、IEEE488.1-1987コマンド・モードになっています。
IEEE488.1-1987コマンド・モードとIEEE488.2-1987コマンド・モードの切り換えは、以下のように行います。

・OLDC OFFを送る　→　IEEE488.2-1987コマンド・モードになります。
・OLDC ON を送る　→　IEEE488.1-1987コマンド・モードになります。

1.3　GPIBのセット・アップ

(1)　GPIBの接続

以下に標準的なGPIBの接続を示します。GPIBコネクタは2 本のねじでしっかり固定して、使用中にゆるむことがないように注意して下さい。



GPIBインタフェースの使用時においては、以下のようなことに注意して下さい。

・1 つのバス・システムで使われるGPIBケーブルの全ケーブル長は、2m×｛接続される機器の数(GPIB コントローラも1 つの機器として数える)｝以下です。また、ケーブルの全ケーブル長は20m 以下とします。

・1 つのバス・システムに接続できる機器の数は、最高15台です。

・ケーブル間の接続方法には制限はありません。ただし、1 台の機器上に4 個以上のGPIBコネクタを重ねないで下さい。4 個以上重ねるとコネクタの取り付け部に過度の力が加わり、破損することがあります。

たとえば、5 台の機器から構成されるシステムで使用できる全ケーブル長は、10m 以下(5台×2m/ 台=10m) です。全ケーブル長が許容最大長を超えない範囲で、自由に分配することができます。ただし、10台以上の機器を接続する場合は、何台かの機器を2m以下のケーブルで接続して、全ケーブル長が20m を超えないようにする必要があります。

(2)　GPIBアドレスの設定

GPIBアドレスは、正面パネル・キーで設定します。キー操作は機種ごと(R3764/66 、R3765/67) に異なります。詳細は、各機種の取扱説明書を参照して下さい。

MEMO

2.　GPIBバスの機能

2.1　GPIBインタフェース機能

ｺｰﾄﾞ	説明
SH1	ソース・ハンドシェーク機能あり
AH1	アクセプタ・ハンドシェーク機能あり
T6	基本的トーカ機能、シリアル・ポール機能、リスナ指定によるトーカ解除機能
TE0	拡張トーカ機能なし
L4	基本的リスナ機能、トーカ指定によるリスナ解除機能
LE0	拡張リスナ機能なし
SR1	サービス・リクエスト機能あり
RL1	リモート機能、ローカル機能、ローカル・ロック・アウト機能
PP0	パラレル・ポール機能なし
DC1	デバイス・クリア機能
DT1	デバイス・トリガ機能
C1	システム・コントローラ機能
C2	IFC 送信、コントローラ・イン・チャージ機能
C3	REN 送信機能
C4	SRQ に対する応答機能
C12	インタフェース・メッセージの送信、コントロールの受け渡し機能
E1	オープン・コレクタ・バス・ドライバを使用

2.2　コントローラ機能

R3764/66、R3765/67には、システム・コントローラ・モードとアドレッサブル・モードがあります。それぞれ特徴を以下に示します。

	ｼｽﾃﾑ･ｺﾝﾄﾛｰﾗ･ﾓｰﾄﾞ	ｱﾄﾞﾚｯｻﾌﾞﾙ･ﾓｰﾄﾞ
起動時	ｱｸﾃｨﾌﾞ･ｺﾝﾄﾛｰﾗ	ﾉﾝｱｸﾃｨﾌﾞ･ｺﾝﾄﾛｰﾗ
IFC	ｺﾝﾄﾛｰﾙ可	ｺﾝﾄﾛｰﾙ不可
REN	ｺﾝﾄﾛｰﾙ可	ｺﾝﾄﾛｰﾙ不可

アドレッサブル・モードでアクティブ・コントローラになるには、TCT インタフェース・メッセージを受信しなければなりません。

システム・コントローラは、GPIBバス上に 1台だけ許されます。GPIBバスで接続されたシステムの起動時には、システム・コントローラがアクティブ・コントローラとなります。同時にアクティブ・コントローラは、GPIBバス上に 1台だけ許されます。このアクティブ・コントローラがGPIBバス上の機器のコントロールを実行します。具体的にはインタフェース・メッセージの送信（IFC およびREN はシステム・コントローラだけが送信する）およびサービス・リクエスト(SRQ) の受信を実行します。

インタフェース・メッセージは、トーカとリスナの指示、シリアル・ポール、デバイスクリア、トリガ、ローカルなどを計測器に伝え、サービス・リクエストで計測器からの割り込みを受信します。

アクティブ・コントローラは、コントロール権を他のノンアクティブ・コントローラに渡すことができます。コントロール権を渡したい機器をトーカにして、TCT インタフェース・メッセージを発行すると、コントロール権がその機器に渡ります。これを「パス・コントロール」と呼びます。

アクティブ・コントローラが持っているコントロール権は、システム・コントローラがIFC インタフェース・メッセージを発行すると、システム・コントローラに戻ります。

2.3　インタフェース・メッセージに対する応答

この節で説明するインタフェース・メッセージに対する本器の応答は、IEEE規格488.1-1987および488.2-1987で定義されています。

インタフェース・メッセージの本器への送り方は、使用するコントローラの取扱説明書を参照して下さい。

2.3.1　インタフェース・クリア(IFC)

このメッセージは、本器へ直接信号線で送られてきます。
このメッセージによって本器はGPIBバスの動作を停止します。すべての入/ 出力を停止しますが、入出力バッファはクリアされません（クリアは DCLで実行される）。このとき本器がアクティブ・コントローラに指定されている場合、GPIBバスのコントロール権は解除され、システム・コントローラがコントロール権を得ます。

2.3.2　リモート・イネーブル(REN)

このメッセージは、本器へ直接信号線で送られてきます。
このメッセージが真のとき、本器がリスナに指定されるとリモート状態になります。
この状態はGTL を受けとるか、REN が偽になるか、LOCAL キーを押すまで続きます。
本器は、ローカル状態のとき、すべての受信データを無視します。
リモート状態のとき、LOCAL キーを除くすべてのキー入力を無視します。
ローカル・ロック・アウト状態(LLO : [2.3.8 項] を参照）のとき、すべてのキー入力を無視します。

2.3.3　シリアル・ポール・イネーブル(SPE)

本器はこのメッセージを外部から受信すると、シリアル・ポール・モードになります。このモードでは、トーカに指定されると通常のメッセージではなくステータス・バイトを送信します。このモードはシリアル・ポール・ディセーブル(SPD) メッセージを受信するか、IFC メッセージを受信するまで続きます。

本器がサービス・リクエスト(SRQ) メッセージをコントローラに送信しているときには、応答データのbit6(RQS bit) が 1(TRUE)になります。送信が終了後、RQS bit は 0 (FALSE) になります。
サービス・リクエスト(SRQ) メッセージは、直接信号線で送ります。

2.3.4　グループ・エグゼキュート・トリガ(GET)

このメッセージは本器にトリガをかけ、以下の条件が満たされていれば、本器は測定を始めます。

・トリガ・ソースがGPIBバスになっている。　（TRIG:SOUR BUS である）
・本器がトリガ待ちステートになっている。（[5. トリガ・システム] を参照）

GET は、*TRGと同一の動作を行いますが、TRIG:IMM、TRIG:SIGとは異なります。
GET 、*TRG、TRIG:IMM、およびTRIG:SIGは、入力バッファ上につまれて受信した順番に実行されます。

2.3.5　デバイス・クリア(DCL)

本器は DCLを受け取ったときに、以下のことを実行します。

・入力バッファと出力バッファのクリア
・構文解析部、実行コントロール部、応答データ生成部のリセット
・次に実行するリモート・コマンドを妨げる全コマンドのキャンセル
・他のパラメータを待つため一時停止されているコマンドのキャンセル
・*OPCと*OPC? のキャンセル

以下のことは実行しません。

・本器に設定または格納されているデータの変更
・正面パネル操作の中断
・実行中の本器の動作への影響や中断
・MAV を除くステータス・バイトの変更（MAV は出力バッファのクリアの結果として 0 になる）

2.3.6　セレクテッド・デバイス・クリア(SDC)

DCL と同一の動作を行います。ただし、SDC は本器がリスナの場合だけ実行されます。その他の場合は無視されます。

2.3.7　ゴー・トゥ・ローカル(GTL)

このメッセージは、本器をローカル状態にします。ローカル状態になると、正面パネル操作がすべて有効になります。

2.3.8　ローカル・ロック・アウト(LLO)

このメッセージは、本器をローカル・ロック・アウト状態にします。この状態で本器がリモート状態になると、正面パネル操作はすべて禁止されます（通常のリモート状態では、LOCAL キーで正面パネル操作ができる）。

このとき本器をローカル状態にする方法は、以下の 3通りあります。

・GTL メッセージを本器に送る
・REN メッセージを偽にする（このときローカル・ロック・アウト状態も解除される）
・電源を再投入する

2.3.9　テイク・コントロール(TCT)

本器がトーカに指示されているとき、このメッセージを受けとると、パス・コントロールされ、アクティブ・コントローラになります。IFC メッセージの受信で本器はアドレッサブル・モードに戻ります。

2.4　メッセージ交換プロトコル

本器は、コントローラやその他の機器からGPIBバスを通じてプログラム・メッセージを受け取り、応答データを発生します。プログラム・メッセージには、コマンド、クエリ（応答データを問い合わせるコマンドのことを特に「クエリ」と呼ぶ）、データが含まれています。それらのデータのやりとりには手順があります。この節ではその手順について説明します。

2.4.1　GPIB各種バッファ

本器にはバッファが 3つあります。

・入力バッファ

コマンド解析をするために一時的にデータを貯めておくバッファです。
(1024 バイトの長さをもつ)

入力バッファのクリア方法は、2 通りあります。
・電源投入
・DCL または SDCの実行

・出力バッファ

コントローラからデータを読まれるまでデータを貯めておくバッファです。
(1024 バイトの長さをもつ)

出力バッファのクリア方法は、2 通りあります。
・電源投入
・DCL または SDCの実行

・エラー・キュー

IEEE488.2-1987コマンド・モードでのみ存在します。
これはリモート・コマンドのエラー・メッセージを蓄えておくキューで、深さは10です。リモート・コマンドの解析/ 実行でエラーが発生するたびに、メッセージがキューにつまれます。
SYST:ERRコマンドで読み出すことができ、 1つ読み出すとキューから 1つメッセージを削除します。

エラー・キューのクリア方法は、2 通りあります。
・電源投入
・*CLSの実行

2.4.2　IEEE488.2-1987コマンド・モード

IEEE488.2-1987コマンド・モードは、IEEE規格488.2-1987に適合したメッセージ交換プロトコルに従ってメッセージの送受信を実行します。

このモードで、他のコントローラや機器がメッセージを本器から受信するときに特に重要な項目を、以下に示します。

・クエリの受信によって応答データを生成する
・クエリを実行した順にデータが生成される

パーサー
入力バッファから受信した順序通りにコマンド・メッセージを受け取り、構文解析を実行し、受け取ったコマンドがどんな内容の実行を行うのかを決定します。

コマンドの構文解析時にコマンドの木構造の追跡も行っています。
木構造のどの部分から解析すべきなのかを次のコマンドの解析のために覚えています。
この情報はパーサーがクリアされると木構造の頭まで戻ります。

パーサーのクリア方法は、4 通りあります。
・電源投入
・DCL またはSDC の受信
・';' の次の':' の受信
・ターミネータまたはEOI の受信

応答データ生成
本器はパーサーがクエリを実行すると、その応答としてデータを出力バッファ上に生成します（つまりデータを出力するにはその直前に必ずクエリを送る必要がある）。これはクエリで生成されるデータをコントローラがリードしなければデータがクリアされないことを意味します。

コントローラのリード以外でデータがクリアされる条件は 2通りあり、これらの状態はQuery Error を発生します。

・Unterminated condition ；　クエリをターミネート(ASCIIのLFコードまたはGPIBのEND メッセージ）せずにコントローラが応答データをリードしたか、クエリを送らずにコントローラが応答データをリードした場合

・Interrupted condition ；　コントローラが応答データをリードする前に次のプログラム・メッセージを受け取った場合

2.4.3　IEEE488.1-1987コマンド・モード

IEEE488.1-1987コマンド・モードでは、R3762/63と同一のメッセージ交換のプロトコルを使用します。このモードでは入力バッファに入ったコマンドを解析し、入力バッファよりも長いコマンド列の受信はできません（コマンドとして無視する）。
応答データは、トーカに指定されたときに生成します。応答データの項目はあらかじめクエリよって指示される必要があります。応答データはトーカに指定されるたびに生成し、出力バッファ上にフォーマッティングされます。一度に複数のクエリに応答することはできません。

2.4.4　BASIC モード

本器は内蔵BASIC インタープリタによって本器自身および外部機器をプログラミングする機能がサポートされています。このBASIC インタープリタが実行しているときには本器のGPIBインタフェースは特別なモードに入り、外部からのコマンド・メッセージや本器からのデータ出力はBASIC インタープリタがコントロールします。

データ入出力については、本書第 1部の ENTERとOUTPUTの説明を参照して下さい。
GPIBをBASIC インタープリタがコントロールしない方法については、本書第 1部のCONTROL コマンドの説明を参照して下さい。

本器はアドレッサブル・モードで、この内蔵BASIC インタープリタをコントロールするための特別の方法を提供しています。

@BASICステートメント

（注）'@' 文字は入力メッセージの先頭になければなりません。

この方法で実行するBASIC ステートメントには制限はありません。
また、ここでいうBASIC ステートメントは、コマンドに限りません。つまり、以下の例のようなステートメントも実行できます。

・@100 PRINT "Hello World"
・@VAR=1000

この方法で外部コントローラからGPIBバスを通して内蔵BASIC プログラムのダウンロードが可能です。

BASIC インタープリタ実行中には、GPIBバスはインタープリタ自身がコントロールしていますが、このときも前記の方法によって外部コントローラからステートメントを実行できます（BASIC の実行中のコマンドに対する制限は受けます）。
逆にいえば、@ が先頭についた文字列はアドレッサブル・モードではGPIBバスから受信できません（システム・コントローラ・モードではこの制限はなく、アドレッサブル・モードでは回避する手段はない）。

MEMO

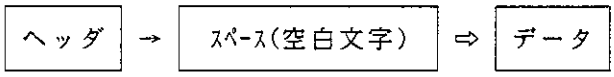
3.　コマンド文法

3.1　IEEE488.2-1987コマンド・モード

IEEE488.2-1987コマンド・モードで入力する文字は、文字列データとブロック・データを除き英字の大文字・小文字の区別はありません。

3.1.1　コマンド文法

R3752/53コマンド・モードのコマンド文法は、以下のフォーマットで定義されています。



（注）⇨は繰り返しを意味します。

(1)　ヘッダ

ヘッダは、コロン(:) で区切られた複数のニーモニックからなる階層構造を持ちます。 4文字以上からなるニーモニックは 4文字（または 3文字）の「ショート・フォーム」をもちます（省略しないニーモニックを「ロング・フォーム」と呼ぶ）。どちらのフォームをどのように組み合わせても構いません。

ヘッダの直後に ?を付けるとクエリ・コマンドになります。

(2)　スペース（空白文字）

１文字分以上のスペースが必要です。スペース以外ではエラーとなります。

(3)　データ

コマンドが複数のデータを必要とするときは、データをカンマ(,) で区切って複数並べます。カンマ(,) の前後にスペース（空白文字）を入れても構いません。

データ・タイプの詳細については、[3.1.2　データ・フォーマット] を参照して下さい。

(4)　複数のコマンドの記述

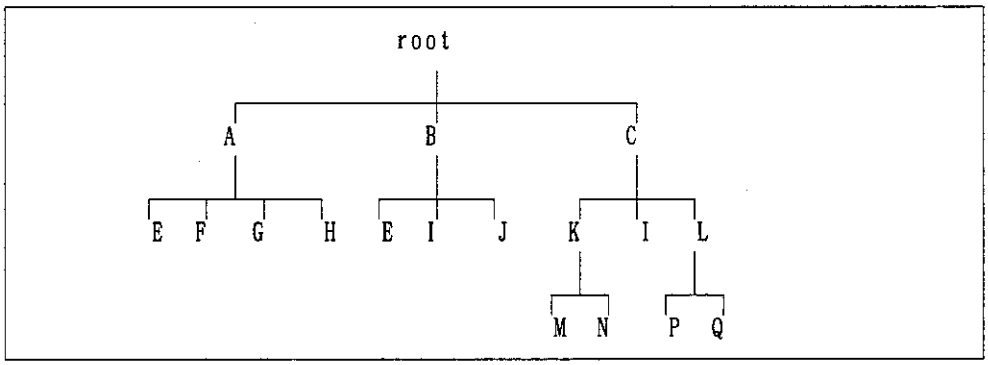
IEEE488.2-1987コマンド・モードでは複数のコマンドをセミコロン(;) で区切って１行で記述することが可能です。
このようにコマンドを記述した場合には、ヘッダの持つ階層構造の中でカレント・パスを移動しながらコマンドを実行していきます。

(5)　カレント・パスの移動

以下の規則に従ってカレント・パスは移動します。

・電源投入時　　：カレント・パスはrootにセットされる
・ターミネータ　：カレント・パスはrootにセットされる
・コロン（:）　　：カレント・パスをコマンド・ツリーの中で 1階層下に移動する
　コロン（:)がコマンドの先頭の文字の場合、コロン（:)はカレント・パスをrootにする
・セミコロン(;)：カレント・パスを変更しない
・共通コマンド　：カレント・パスに関係なく実行できます。*RSTコマンドを実行するとカレント・パスはrootにセットされる（* 以下の例を参照）

例）以下のヘッダ構造とします。



このとき、以下のカレント・パス動作になります。

①:A:E;:B:E
　2 つ目のコマンドの :はカレント・パスをrootに移動するので、A:EとB:Eはどちらも正しいコマンドです。

②:A:E<END> B:E
　<END>（ターミネータ）はカレントパスをrootに移動するので、 A:EとB:Eはどちらも正しいコマンドです。

③:A:E;F;G;H
　; はカレントパスを移動しないので、 :A:E;F;G;H は結果的に A:E、A:F、A:G 、A:H の 4つのコマンドと等しくなります。

④:C:I;K:N;M
　: がカレントパスを移動するので、 K:Nは :C:の階層から見ることになります。従って K:Nは C:K:Nとなります。また同時に、K:N は :を含むためカレント・パスを :C:K:に変更し、最後の Mは C:K:Mと解釈されます。

⑤:A:E;*ESR 16
　共通コマンドはカレント・パスに関係ないので、 *ESR 16は正しく実行されます。

⑥:A:E;*ESR 16;F;G;H
共通コマンドはカレント・パスを変更しないので、 3つ目の Fは 1つ目の :A:Eで設定されたカレント・パスの :A: で探されます。
したがって、 Fは A:F、G はA:G 、H はA:H になります。

以下の例では、文法エラーとなります。

①:A:E;B:E
A:E はカレント・パスを :A:に変更しています。
したがって、B:E は :A:の階層で探されるが、 Bというニーモニックが見つからないのでエラーとなります。

②:C:K:M;L:P
:C:K:Mはカレント・パスを :C:K:に変更しています。
したがって、L:P は :C:K:で探されるが、 Lというニーモニックが見つからないのでエラーとなります。

3.1.2　データ・フォーマット

IEEE488.2-1987コマンド・モードでは、この項で示すデータ・タイプをデータの入出力で使用します。

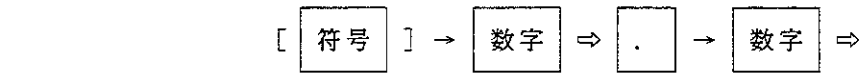
(1)　数値データ

数値データには以下の 3つのフォーマットがあり、本器に対する数値の入力では、どれを用いても構いません（入力するデータの型に応じて四捨五入される）。
また、コマンドによっては入力時に単位を付けられます。単位に関しては、後述(5)を参照して下さい。

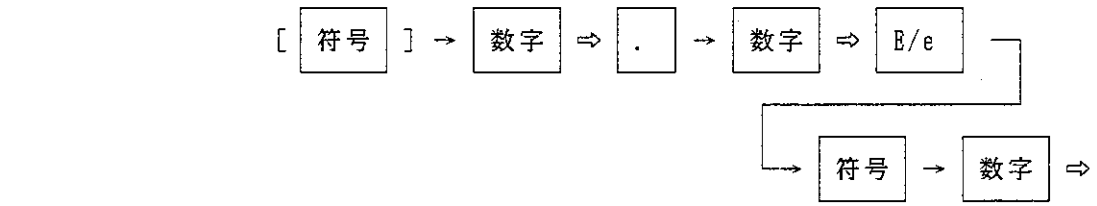
・整数型　　：NR1フォーマット

（注）⇨は繰り返しを意味します。
先頭の符合は省略可能です。

・固定小数点型 ：NR2フォーマット

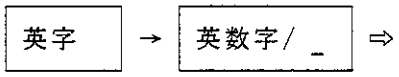


・浮動小数点型 ：NR3フォーマット



(2)　文字データ

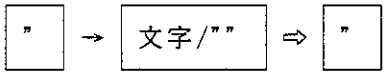
文字データのフォーマットを以下に示します。

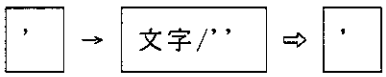


（注）⇨は繰り返しを意味します。

(3)　文字列データ

文字列データには、2 つのフォーマットがあります。





文字列データ中では、ASCII 7bitコード文字として使用できます。

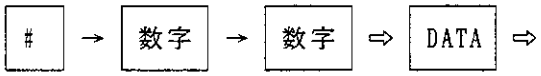
（注）" で始まる文字列データ中では "を "" で表現しなければなりません。
' で始まる文字列データ中では 'を '' で表現しなければなりません。
⇨は繰り返しを意味します。

応答データが文字列データの場合、"で始まる文字列データを必ず出力します。

(4)　ブロック・データ

ブロック・データには、2 つのフォーマットがあります。本器への入力時には、どちらのフォーマットを用いても構いません。

固定長フォーマット：



不定長フォーマット：

（注）⇨は繰り返しを意味します。

固定長のフォーマットでは、# の後の 1文字の数字でその後に続くデータのバイト数の桁数を表します。0 は使えません（不定長になる）。

例）#3128 <data byte> というブロックデータの場合

の後の 3がその後に続く文字列(128) の桁数を表し、128 はその後に続く<data byte> のバイト数を表します。

(5)　単位

単位は数値の後に続く接尾語です。また、単位にはサフィックスを接頭語として使用できます。

使用可能なサフィックスと単位の一覧表を以下に示します。

サフィックス		単位	使用可能なコマンド
1E18	EX	HZ	[SENSe:]BANDwidth[:RESolution] [SOURce:]FREQuency:CENTer [SOURce:]FREQuency:CW [SOURce:]FREQuency:SPAN [SOURce:]FREQuency:STARt [SOURce:]FREQuency:STOP [SOURce:]PSWeep:FREQuency
1E15	PE		
1E12	T		
1E9	G	DEG	[SENSe:]CORRection:OFFSet:PHASe
1E6	MA	DB	INPut:ATTenuation OUTPut:ATTenuation
1E3	K		
1E-3	M *	DBM	[SOURce:]POWer[:LEVel][:AMPLitude] [SOURce:]POWer:STARt [SOURce:]POWer:STOP
1E-6	U		
1E-9	N	M	[SENSe:]CORRection:EDELay:DISTance
1E-12	P	S	[SENSe:]CORRection:EDELay[:TIME] [SENSe:]CORRection:PEXTension:TIME [SOURce:]SWEep:TIME TRIGger[:SEQuence]:DELay
1E-15	F		
1E-18	A		
		OHM	CALCulate:TRANsform:IMPedance:CIMPedance INPut:IMPedance

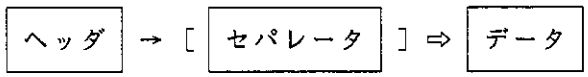
（注）上記の表に載っていないコマンドは、サフィックスのみ使用可能です。
　　* ：単位がHZとOHM の場合、サフィックスは1E6(MAと同等）として動作します。

3.2　IEEE488.1-1987コマンド・モード

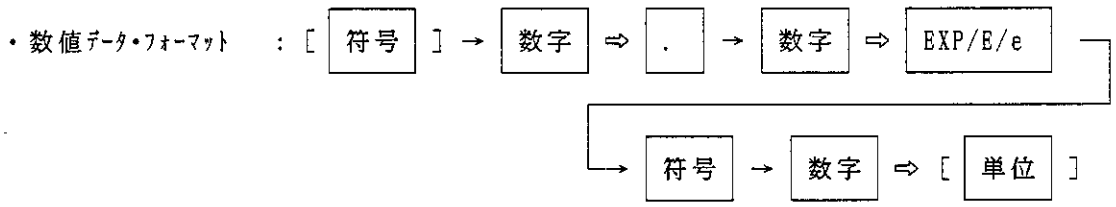
IEEE488.1-1987コマンド・モードのプログラム・メッセージの構造を以下に示します。IEEE488.1-1987コマンド・モードでは、文字列データを除き小文字はセパレータとして扱われます。

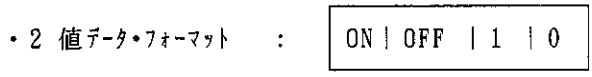
3.2.1　コマンド文法

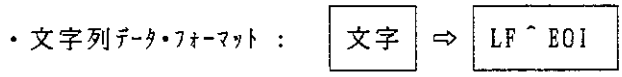
IEEE488.1-1987コマンド・モードの文法は、以下のフォーマットで定義されています。



セパレータは、0 文字以上のスペース、カンマ(,) またはセミコロン(;) です。
データには、3 つのフォーマットがあります。







（注）⇨は繰り返しを意味します。

数値データには、以下の単位が使えます。

GHZ	MHZ	KHZ	HZ
DEG			
DP	DM	DB	
METER	CM		
SEC	MSEC	USEC	NSEC
VOLT	MV	UV	NV
MOHM	KOHM	OHM	
UNIT			
DIV			
PER			

文字列データは、ヘッダの直後の文字から入力データの最後までを文字列とみなします。ヘッダの直後に ?をつけるとクエリ・コマンドになります。

4.　ステータス・バイト

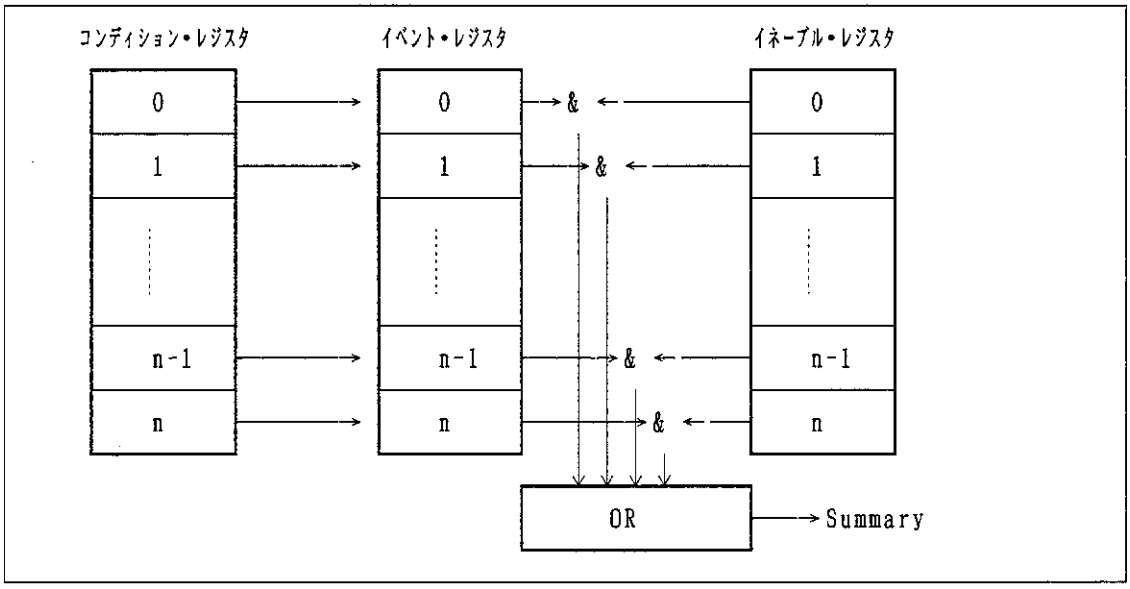
本器ではIEEE規格488.2-1987に適合した階層化されたステータス・レジスタ構造をもち、機器の様々な状態をコントローラへ送信できます。本章ではこのステータス・バイトの動作モデルと、イベントの割当を説明します。

（注）ステータス構造は、コマンド・モードに関係なくR3762/63シリーズと異なります。

4.1　ステータス・レジスタ

4.1.1　ステータス・レジスタの構造

本器は、IEEE規格488.2-1987で定義されたステータス・レジスタのモデルを採用しており、コンディション・レジスタ、イベント・レジスタ、イネーブル・レジスタから構成されています。



(1)　コンディション・レジスタ

コンディションレジスタは、機器のステータスを常に監視しています。つまり、このレジスタには常に最新の機器のステータスが保持されています。
このレジスタにデータを書き込むことはできません。

(2)　イベント・レジスタ

イベント・レジスタは、コンディション・レジスタからのステータスをラッチして保持します（変化を保持する場合もある）。このレジスタがセットされると、クエリで読み出されるか、*CLSでクリアされるまでセットされたままです。
このレジスタにデータを書き込むことはできません。

(3)　イネーブル・レジスタ

イネーブル・レジスタは、イベント・レジスタのどのビットを有効なステータスとしてサマリを生成するのかを指定します。イネーブル・レジスタはイベント・レジスタとAND をとられ、その結果のORがサマリとして生成されます。サマリは次のステータス・レジスタに書き込まれます。
このレジスタはデータを書き込めます。

4.1.2　ステータス・レジスタの種類

本器のステータス・レジスタは、以下の 8種類があります。

(1)　ステータス・バイト・レジスタ　：　4.2節を参照

(2)　スタンダード・イベント・レジスタ　：　4.3節を参照

(3)　スタンダード・オペレーション・ステータス・レジスタ　：　4.4節を参照

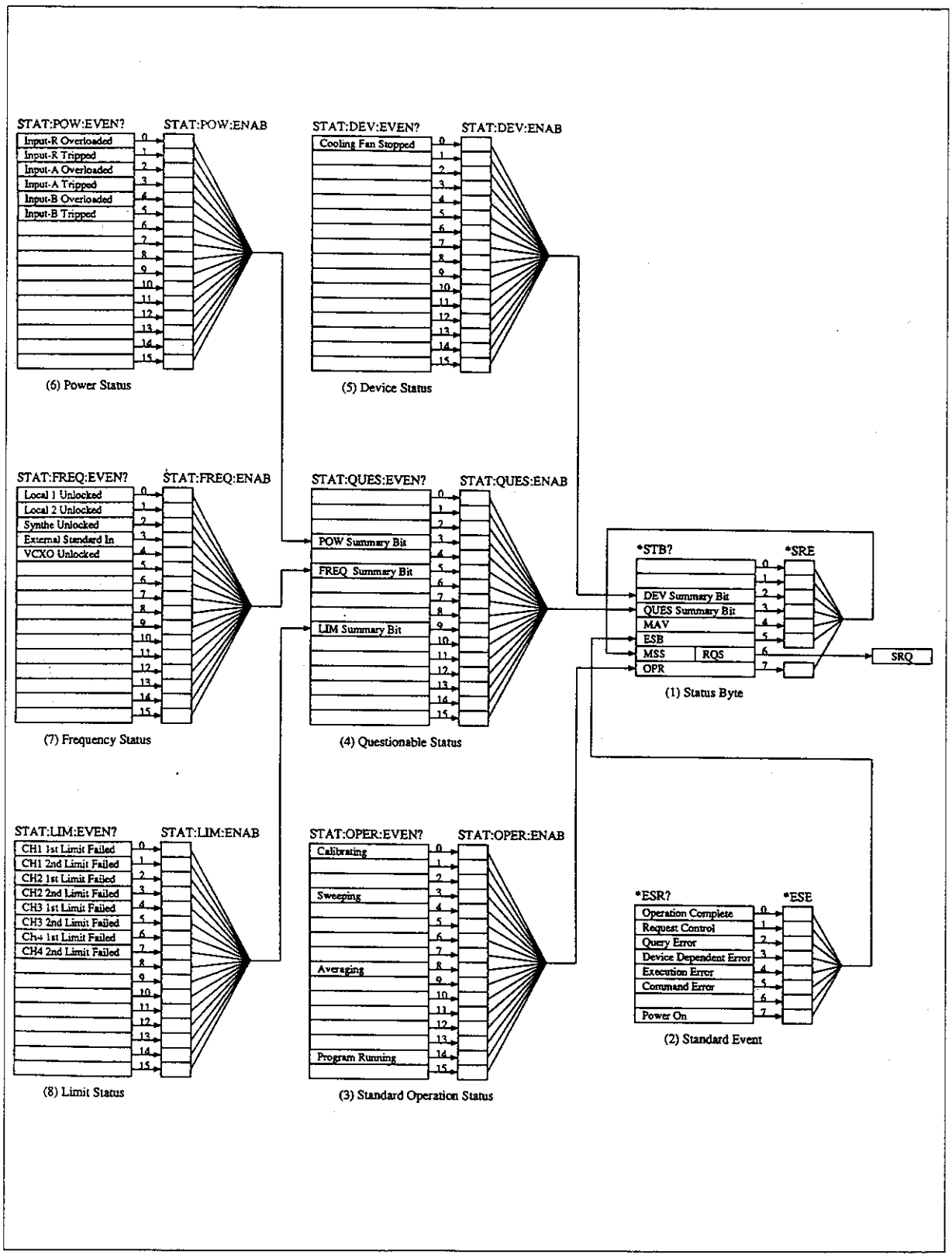
(4)　クエスチョナブル・ステータス・レジスタ

(5)　デバイス・ステータス・レジスタ　：　4.5節を参照

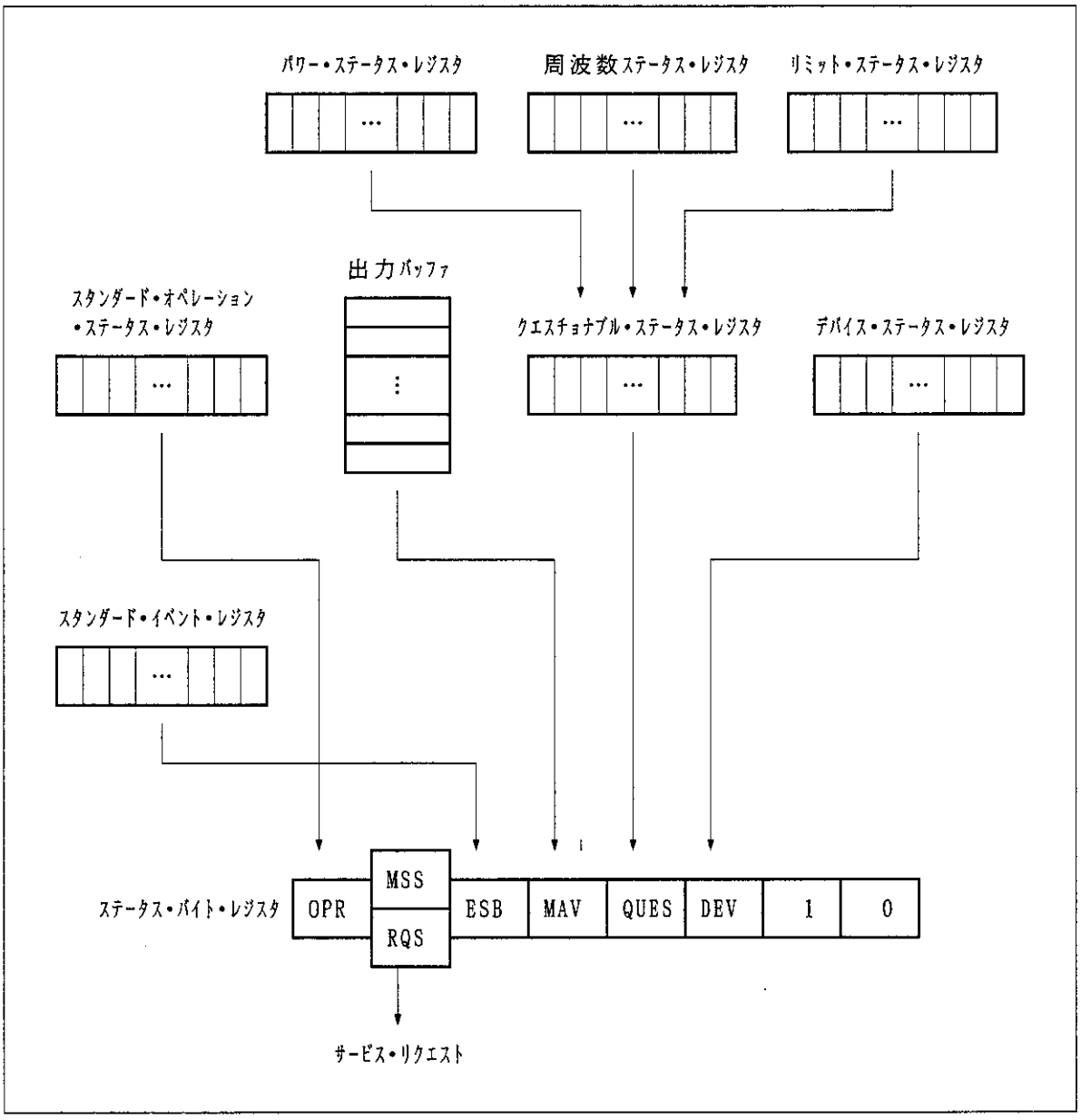
(6)　パワー・ステータス・レジスタ　：　4.6節を参照

(7)　周波数ステータス・レジスタ　：　4.7節を参照

(8)　リミット・ステータス・レジスタ　：　4.8節を参照


STAT:POW:EVEN?
STAT:POW:ENAB
Input-R Overloaded
Input-R Tripped
Input-A Overloaded
Input-A Tripped
Input-B Overloaded
Input-B Tripped
(6) Power Status
STAT:DEV:EVEN?
STAT:DEV:ENAB
Cooling Fan Stopped
(5) Device Status
STAT:FREQ:EVEN?
STAT:FREQ:ENAB
Local 1 Unlocked
Local 2 Unlocked
Synthe Unlocked
External Standard In
VCXO Unlocked
(7) Frequency Status
STAT:QUES:EVEN?
STAT:QUES:ENAB
POW Summary Bit
FREQ Summary Bit
LIM Summary Bit
(4) Questionable Status
*STB?
*SRE
DEV Summary Bit
QUES Summary Bit
MAV
ESB
MSS
RQS
OPR
SRQ
(1) Status Byte
STAT:LIM:EVEN?
STAT:LIM:ENAB
CH1 1st Limit Failed
CH1 2nd Limit Failed
CH2 1st Limit Failed
CH2 2nd Limit Failed
CH3 1st Limit Failed
CH3 2nd Limit Failed
CH4 1st Limit Failed
CH4 2nd Limit Failed
(8) Limit Status
STAT:OPER:EVEN?
STAT:OPER:ENAB
Calibrating
Sweeping
Averaging
Program Running
(3) Standard Operation Status
*ESR?
*ESE
Operation Complete
Request Control
Query Error
Device Dependent Error
Execution Error
Command Error
Power On
(2) Standard Event

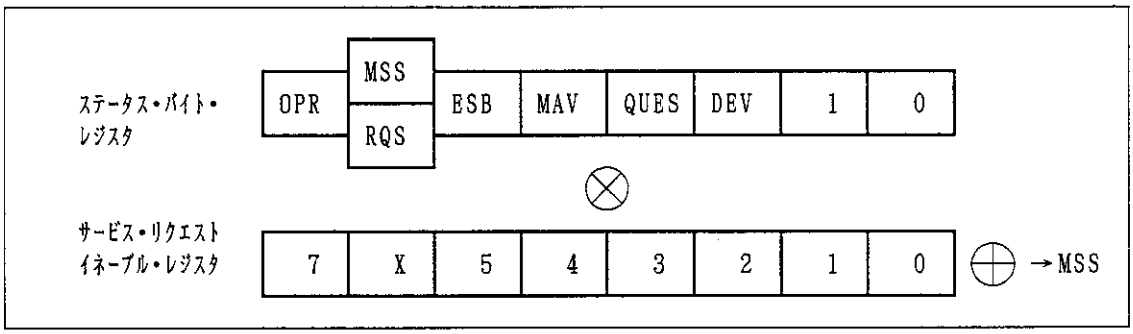
本器のステータス・レジスタの配置を、以下に示します。



4.2　ステータス・バイト・レジスタ

ステータス・バイト・レジスタは、ステータス・レジスタ(4.1.1項を参照）からの情報を要約しています。また、このステータス・バイト・レジスタのサマリがサービス・リクエストとしてコントローラに送信されます。そのため、ステータス・バイト・レジスタは、ステータス・レジスタ構造とは若干違った動作を行います。この節ではステータス・バイト・レジスタに関して説明をします。

ステータス・バイト・レジスタの構造を、以下に示します。



このステータス・バイト・レジスタは、以下の 3点を除くと[4.1.1項] のステータス・レジスタ構造に従います。

①ステータス・バイト・レジスタのサマリが、ステータス・バイト・レジスタの bit6 に書き込まれます。

②イネーブル・レジスタの bit6 は常に有効で変更できません。

③ステータス・バイト・レジスタのbit6(MSS) が、サービス・リクエスト要求のRQS を書き込みます。

このレジスタがコントローラからのシリアル・ポールに対して応答します。シリアル・ポールに対して応答するときには、ステータス・バイト・レジスタのbit0～5 、bit7およびRQS が読み出され、その後にRQS は0 にリセットされます。その他のビットはそれぞれの要因が 0になるまでクリアされません。

ステータス・バイト・レジスタ、RQS 、MSS は、*CLSを実行するとクリアできます。

ステータス・バイト・レジスタの各ビットの意味を、以下に示します。

bit		説明
7	OPR	・OPR は、ｽﾀﾝﾀﾞｰﾄﾞ･ｵﾍﾟﾚｰｼｮﾝ･ｽﾃｰﾀｽ･ﾚｼﾞｽﾀ のサマリである
6	MSS	・RQS は、ステータス・バイト・レジスタの MSSが 1になったときTRUEになるが、そのMSS はすべてのステータス・データ構造のサマリ・ビットになっている ・MSS は、サービス・リクエストでは読めない（ただし、RQS が1 のときは MSSが1 であることがわかる） ・MSS を読むには、共通コマンド*STB? を用いる *STB? ではステータス・バイト・レジスタのbit0～5 、bit7およびMSS が読み出される この場合ステータス・バイト・レジスタとMSS はクリアされない ・MSS はステータス・レジスタ構造のすべてのマスクされていない要因がクリアされるまで0 にならない
5	ESB	・ESB は、スタンダード・イベント・レジスタのサマリである
4	MAV	・MAV は出力バッファの要約ビット ・出力バッファに出力データがある間 1になり、データが読み出されると0 になる
3	QUES	・QUESは、クエスチョナブル・ステータス・レジスタのサマリである
2	DEV	・DEV は、デバイス・ステータス・レジスタのサマリである
1～0		・常に0

4.3　スタンダード・イベント・レジスタ

スタンダード・イベント・レジスタの割り当てを、以下に示します。

bit		説明
7	Power on	・電源投入で 1になる
6		・常に0
5	Command Error	・パーサーが文法エラーを見つけたときに 1にセットされる
4	Execution Error	・GPIBコマンドとして受け取った命令の実行を何らかの理由（パラメータが範囲外など）で失敗すると 1にセットされる
3	Device Dependent Error	・Command Error 、Execution Error 、Query Error 以外のエラーが発生したとき 1にセットされる
2	Query Error	・コントローラが本器からデータを読み出そうとしたときに、データが存在しない、またはデータが消失していると 1にセットされる
1	Request Control	・本器がアクティブ・コントローラになる必要があるときに 1にセットされる
0	Operation Control	・*OPCコマンドを受け取った後、かつ本器が実行しているコマンドがなくなると 1にセットされる

4.4　スタンダード・オペレーション・ステータス・レジスタ

(1)　コンディション・レジスタ

スタンダード・オペレーション・ステータスのコンディション・レジスタの割り当てを、以下に示します。

bit		説明
15		・常に0
14	Program running	・内蔵BASIC 言語がRUN していると 1にセットされる
13～ 4		・常に0
3	Sweeping	・掃引実行中に 1にセットされる
2～ 1		・常に0
0	Calibrating	・補正データ取得中に 1にセットされる

(注)　イベント・レジスタと違い、bit8(Averaging) は常に0 となります。

(2)　イベント・レジスタ

スタンダード・オペレーション・ステータスのイベント・レジスタは、対応するコンディション・レジスタが 1→0 へ変化するときをラッチしています。
スタンダード・オペレーション・ステータスのイベント・レジスタの割り当てを、以下に示します。

bit		説明
15		・常に0
14	Program running	・内蔵BASIC 言語が停止すると 1にセットされる
13～ 9		・常に0
8	Averaging	・アベレージ終了時に1 にセットされる
7～ 4		・常に0
3	Sweeping	・掃引終了時に1 にセットされる
2～ 1		・常に0
0	Calibrating	・補正データ取得終了時に1 にセットされる

4.5　デバイス・ステータス・レジスタ

コンディション・レジスタの割り当てを、以下に示します。

bit		説明
0	Cooling Fan Stopped	・冷却用ファンが止まっていると 1にセットされる
その他		・常に0

4.6　パワー・ステータス・レジスタ

コンディション・レジスタの割り当てを、以下に示します。

bit		説明
0	Input-R Overloaded	・入力R に過入力レベルが入っていると1 にセットされる
1	Input-R Tripped	・入力R の保護回路が動作していると1 にセットされる
2	Input-A Overloaded	・入力A に過入力レベルが入っていると1 にセットされる
3	Input-A Tripped	・入力A の保護回路が動作していると1 にセットされる
4	Input-B Overloaded	・入力B に過入力レベルが入っていると1 にセットされる
5	Input-B Tripped	・入力B の保護回路が動作していると1 にセットされる
その他		・常に0

イベント・レジスタは、対応するコンディション・レジスタの 0→1 への変化をラッチしています。つまり、過入力レベルが入った（または保護回路が作動した）場合に、 1にセットされます。

4.7　周波数ステータス・レジスタ

コンディション・レジスタの割り当てを、以下に示します。

bit		説明
0	Local 1 Unlocked	・ローカル1 のロックが外れていると1 にセットされる
1	Local 2 Unlocked	・ローカル2 のロックが外れていると1 にセットされる
2	Synthe Unlocked	・シンセのロックが外れていると1 にセットされる
3	External Standard In	・外部基準周波数が入力されていると1 にセットされる
4	VCXO Unlocked	・VCXOのロックが外れていると1 にセットされる
その他		・常に0

イベント・レジスタは、対応するコンディション・レジスタの 0→1 への変化をラッチしています。つまり、ロック外れが起こった場合などに、 1にセットされます。

4.8　リミット・ステータス・レジスタ

コンディション・レジスタの割り当てを、以下に示します。

bit		説明
0	CH1 1st Limit Failed	・チャンネル1 の第1 波形がFAILになっていると 1 にセットされる
1	CH1 2nd Limit Failed	・チャンネル1 の第2 波形がFAILになっていると 1 にセットされる
2	CH2 1st Limit Failed	・チャンネル2 の第1 波形がFAILになっていると 1 にセットされる
3	CH2 2nd Limit Failed	・チャンネル2 の第2 波形がFAILになっていると 1 にセットされる
4	CH3 1st Limit Failed	・チャンネル3 の第1 波形がFAILになっていると 1 にセットされる
5	CH3 2nd Limit Failed	・チャンネル3 の第2 波形がFAILになっていると 1 にセットされる
6	CH4 1st Limit Failed	・チャンネル4 の第1 波形がFAILになっていると 1 にセットされる
7	CH4 2nd Limit Failed	・チャンネル4 の第2 波形がFAILになっていると 1 にセットされる

イベント・レジスタは、対応するコンディション・レジスタの 0→1 への変化をラッチしています。つまり、各波形でFAILが発生した場合に、 1にセットされます。

4.9　SRQE/SRQD の動作

本器ではIEEE規格488.2-1987には規定されていない拡張をR3762/63コンパチブル・モードのためにサービス・リクエストのシステムに組み込んでいます。この節で説明する項目はIEEE488.2-1987コマンド・モードには適用されません。

R3762/63ではサービス・リクエストの許可／禁止はSRQE/SRQD というコマンドで実行しましたが、IEEE規格488.2-1987のステータス・データ構造の採用にともなって本来リクエストの許可／禁止はイネーブル・レジスタによって実現する項目になりました。
しかし、イネーブル・レジスタの性格上完全に同じことを実現することができない（イネーブルレジスタでは、要因が1 のときにイネーブルにするとリクエストが発生する）のでSRQE/SRQD に関してはIEEE488.1-1987コマンド・モードに限ってIEEE規格488.2-1987を拡張してあります。

IEEE488.1-1987コマンド・モードのSRQE/SRQD は、ステータス・データ構造の RQSのイネーブル／ディセーブルとして動作します。SRQEは既にあるリクエストは無視し、RQS を発行しません。新たに発生するMSS に対してのみRQS メッセージをコントローラに発信します。
SRQDは常にRQS の発信を停止します。したがって、RQS がTRUEの状態で SRQD SRQEと連続的に実行すると、 RQSは FALSEになります。このときコントローラはRQS の状態を読み出せなくるので、RQS が必要な場合は SRQD を実行する前にシリアル・ポールを本器に対して実行して下さい。

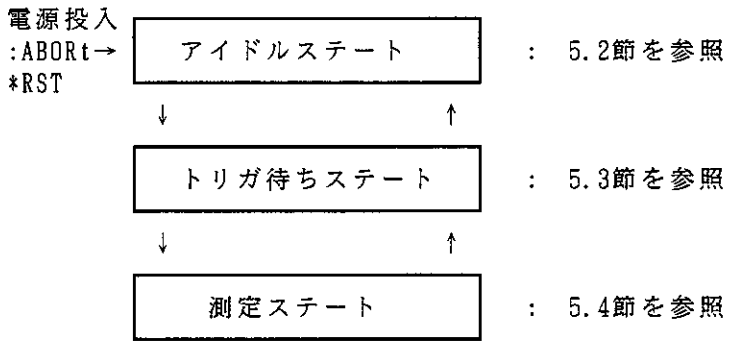
MEMO

5.　トリガ・システム

この章ではトリガ・システムについて説明をします。
トリガ・システムは、測定を指定したイベントに同期させるために使用します。このイベントは GETインタフェース・メッセージ、*TRGコマンドなどのGPIBコマンド、外部トリガ信号を指します。イベントから測定開始までの遅延時間などもトリガ・システムを用いて指定できます。

5.1　トリガ・モデル

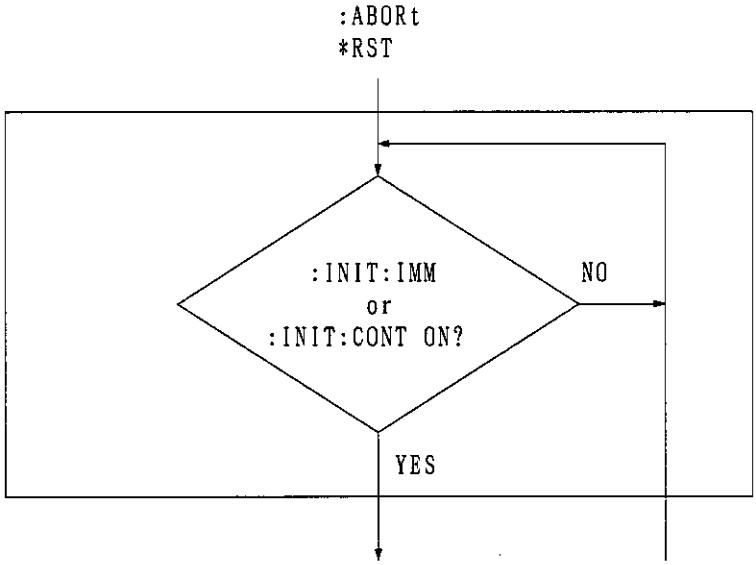
以下に本器のトリガ・システムのモデルを示します。



トリガ・ステートは、電源投入、あるいは:ABORtまたは*RSTコマンドの実行で、トリガ・ステートがアイドル・ステートになります。
アイドル・ステートとトリガ待ちステートは、測定を実行するために満たす必要のある条件を待ちます。

5.2　アイドル・ステート

本器のトリガ・システムは、電源投入でアイドル・ステートになります。
また、:ABORtまたは*RSTコマンドを実行すると、強制的にアイドル・ステートになります。このステートの動作は、以下のようになります。

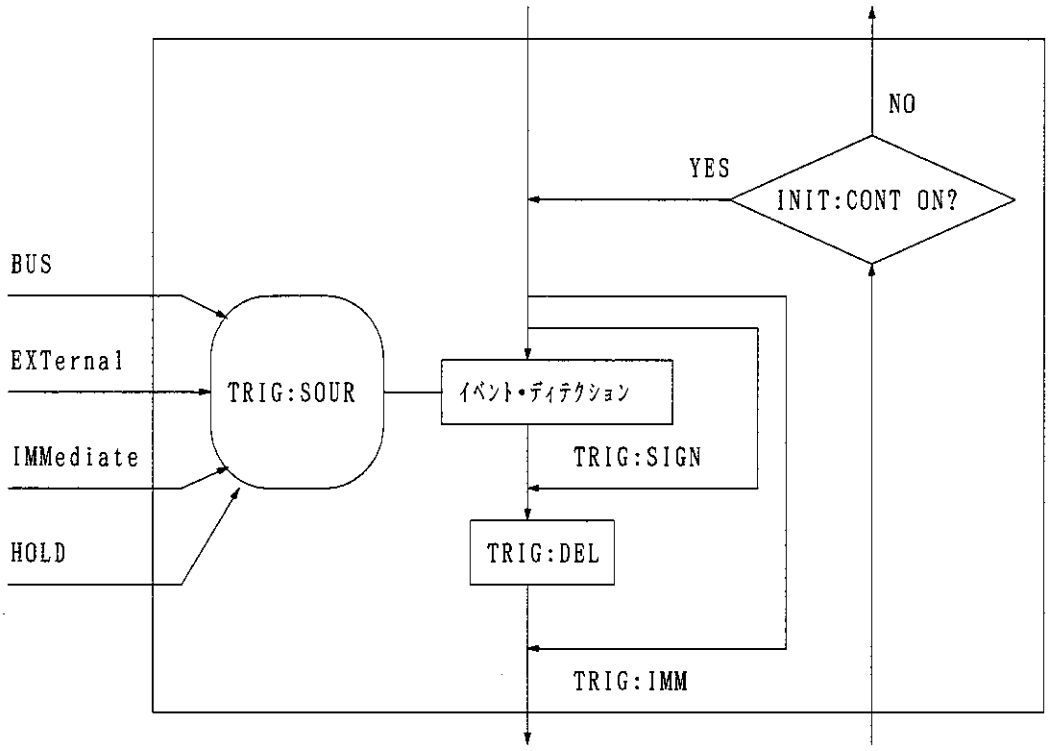


トリガ・システムは、INITiate[:IMMediate]またはINITiate:CONTinuous ONになるまでこのステートから抜け出しません。これらの条件でトリガ・システムはトリガ待ちステートに移行します。

（注）*RSTではINITiate:CONTinuous OFF になるので、測定は止まります。

トリガ・システムがアイドル・ステートを抜けると、本器のオペレーション・ペンディング・フラグが常にセットされます。
また、本器がアイドル・ステートになると、オペレーション・ペンディング・フラグはクリアされます。*OPC、*OPC? 、*WAIはこのオペレーション・ペンディング・フラグを参照します。

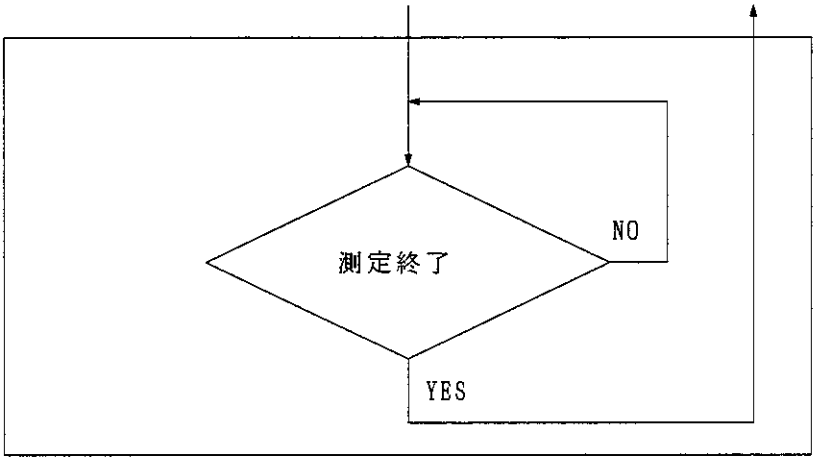
5.3　トリガ待ちステート



本器のトリガ待ちステートは上図のような構成になっています。TRIGger:SOURceコマンドでトリガ・ソースを設定し、イベント・ディテクション部で要因を待ちます。
トリガがかかり、イベント・ディテクションを抜けるとTRIGger:DELay コマンドで設定した時間を待った後、次の測定ステートに進みます。
このトリガ待ちステートでTRIGger:SIGNalコマンドを受け取ると、イベント・ディテクション部をパスします。また、TRIGger[:IMMediate] コマンドを受け取るとTRIGger:DELay もパスし、即座に測定ステートに入ります。

測定ステートを終了して戻ってきたとき、INITiate:CONTinuous ONに設定されているとアイドル・ステートに戻らずに即座に次のトリガ待ちになります。

5.4　測定ステート



実際に測定を実行するステートです。測定ステートになると、本器は掃引を行い、測定データを取得します。

5.5　IEEE488.1-1987コマンド・モード

本器がIEEE488.1-1987コマンド・モードのときは、これまで説明したトリガ・システムの全機能は利用できません。以下に示すトリガ・システムのマクロ的な 4つのコマンドが使用できるだけです。
右側に各コマンドの実際の動作をIEEE488.2-1987コマンド・コードで記述しています（多少実際の動作と異なります）。

```
CONT      INITiate:CONTinuous ON
SINGLE    INITiate:CONTinuous OFF;:ABORt;INITiate
MEAS      ABORt;INITiate
SWPHLD    INITiate:CONTinuous OFF;:ABORt
```

6. サンプル・プログラム

サンプル・プログラム例を 3つ記載します。

(1) プログラム 1 ： 中心周波数とスパン周波数を入力し、波形の全ポイントのレベルを取り込み、変数に代入します。
すべて取り込み終わると 1～1201まで順にレベルを表示します。

(2) プログラム 2 ： 一度掃引させ、掃引終了のSRQ を受け取るまでループして待ち、SRQ を受信するとループを抜けて先に進む処理の基本形です。

(3) プログラム 3 ： 中心周波数、スパン周波数を入力して、波形の最大レベルと最大レベルの周波数をサーチし、結果を表示します。

(1)

```
100 !************************
110 !*                      *
120 !* BINARY DATA TRANSFER *
130 !*     TEST PROGRAM     *
140 !*                      *
150 !************************
160 !
170 DIM DA(1201)
180 INTEGER N,LP
190 ADD=31
195 OUTPUT ADD;"OLDC OFF"
200 OUTPUT ADD;"DISP:ACT 1;:CALC:FORM MLOG"
210 OUTPUT ADD;"SWE:POIN 1201"
220 OUTPUT ADD;"INIT:CONT OFF"
230 CLS
240 INPUT "CENTER FREQ ? [MHz] =",CF
250 INPUT "SPAN   FREQ ? [KHz] =",SP
260 OUTPUT ADD;"FREQ:CENT ",CF,"MHz"
270 OUTPUT ADD;"FREQ:SPAN ",SP,"KHz"
280 OUTPUT ADD;"FREQ:STAR?"
290 ENTER ADD;STA
300 OUTPUT ADD;"FREQ:STOP?"
310 ENTER ADD;STP
320   P1=POINT1(STA,0)
330   P2=POINT1(STP,0)
340   N=TRANSR(P1,P2,DA(1),0)
350   FOR LP=1 TO 1201
360     PRINT "POINT ";(LP-1);" = ";DA(LP)
370   NEXT LP
380 PRINT "DATA COUNT = ";N
390 STOP
```

解説

```
100 ～160 コメント行
170       変数の配列を宣言する（波形データが代入される）
180       変数を整数に宣言する
190       ネットワーク・アナライザ本体のアドレスを変数に代入する
195       IEEE488.2-1987コマンド・モードする
200       チャンネル1 のフォーマットをLOGMAGに設定する
210       測定ポイントを1201ポイントに設定する
220       掃引をシングル・モードに設定する
230       管面の文字を消す
240       中心周波数を入力し、変数に代入する（単位:MHz）
250       スパン周波数を入力し、変数に代入する（単位:KHz）
260       入力した中心周波数に設定する
270       入力したスパン周波数に設定する
280       本器からスタート周波数を取り込む
290       取り込んだ値を変数に代入する
300       本器からストップ周波数を取り込む
310       取り込んだ値を変数に代入する
320       取り込んだスタート周波数をアドレス・ポイントに変換する
330       取り込んだストップ周波数をアドレス・ポイントに変換する
340       波形データ(LOGMAG)を変数に代入する
          ［アドレス・ポイント0 のデータ＝DA(1) : 以下1200まで］
350       1 ～1201のデータを順に表示する
360       波形データを代入した変数DA(1～1201) を管面に表示する
370       LPが1201になるまで繰り返す
380       最後にデータ転送が行われた数を表示する（1201個）
390       プログラム終了
```

(2)

```
100 !******************
110 !*                *
120 !* SRQ SWEEP TEST *
130 !*                *
140 !******************
150 !
160 CLS
162 OUTPUT 31;"OLDC OFF"
165 OUTPUT 31;"STAT:OPER:ENAB 8"
170 OUTPUT 31;"SWE:POIN 1201"
180 OUTPUT 31;"SWE:TIME 1S"
190 OUTPUT 31;"INIT:CONT OFF;:ABOR"
200 INPUT "HIT ENT KEY TO SWEEP START !",DUMMY$
210  GOSUB *SWP
220 PRINT "SWEEP TEST FINISHED !!!"
230 STOP
240 !
250 !******************
260 !
270 *SWP
280   ON ISRQ GOTO *PATH
290   OUTPUT 31;"*SRE 128":SPOLL(31)
300   ENABLE INTR
310   OUTPUT 31;"INIT"
320  *LOOP
330   GOTO *LOOP
340 !
350  *PATH
360   SPOLL(31):DISABLE INTR
370   OUTPUT 31;"*SRE 0"
380 RETURN
```

解説
100 ～150 コメント行
160 管面上の文字を消す
162 IEEE488.2-1987コマンド・モードにする
165 OPERステータスのビット3(Sweep End)をイネーブルにする
170 ネットワーク・アナライザの測定ポイントを1201ポイントに設定する
180 掃引時間を 1秒に設定する
190 掃引をシングル・モードに設定する
200 コメントをCRT 上に表示する (ENT キーで次に進ませる)
210 サブルーチン(*SWP)を呼ぶ
220 コメントをCRT 上に表示する
230 プログラム終了
240 コメント行
250 コメント行
260 コメント行
270 サブルーチン(*SWP)
280 ISRQを受信したら*PATH に行く
290 ｽﾀﾝﾀﾞｰﾄﾞ･ｵﾍﾟﾚｰｼｮﾝ･ｽﾃｰﾀｽ･ﾚｼﾞｽﾀ のSRQ の送信を許可する
300 割り込みの受け付けを許可する
310 掃引をシングル・モードに設定する (ここの場合は一度掃引させる)
320 *LOOP (ループ)
330 *LOOP に行く (ISRQが来るまでループさせておく)
340 コメント行
350 *PATH (ISRQを受信したときの飛び先名)
360 割り込みの受け付けを禁止する
370 すべてのSRQ の送信を禁止する
380 サブルーチン(*SWP)が呼ばれた所に戻る

(3)

```
100 !*****************************
110 !*                           *
120 !* MAX SEARCH SAMPLE PROGRAM *
130 !*                           *
140 !*****************************
150 !
155 OUTPUT 31;"OLDC OFF"
160 OUTPUT 31;"DISP:ACT 1;:CALC:FORM MLOG"
170 OUTPUT 31;"SWE:POIN 1201"
180 OUTPUT 31;"SWE:TIME 1S"
190 CLS
200 INPUT "ENTER CENTER FREQ ? [MHz] =",CF
210 INPUT "ENTER SPAN   FREQ ? [KHz] =",SF
220 OUTPUT 31;"FREQ:CENT ",CF,"MHz"
230 OUTPUT 31;"FREQ:SPAN ",SF,"KHz"
240 OUTPUT 31;"FREQ:STAR?"
250 ENTER 31;S1
260 OUTPUT 31;"FREQ:STOP?"
270 ENTER 31;S2
280   PO1=POINT1(S1,0)
290   PO2=POINT1(S2,0)
300   FR=FMAX(PO1,PO2,0)
310   LV=MAX(PO1,PO2,0)
320   FR=FR/10^6
330 PRINT "***** PROGRAM RESULT *****"
340 PRINT "MAX FREQ  [MHz] = ";FR
350 PRINT "MAX LEVEL [dB]  = ";LV
360 STOP
```

解説

100 ～150 コメント行
155 IEEE488.2-1987コマンド・モードにする
160 ネットワーク・アナライザのチャンネル1 をLOGMAGに設定
170 測定ポイント数を1201ポイントに設定
180 掃引時間を 1秒に設定
190 管面上の文字を消す
200 中心周波数を入力し、変数に代入する（単位：MHz）
210 スパン周波数を入力し、変数に代入する（単位：KHz）
220 入力した中心周波数に設定する
230 入力したスパン周波数に設定する
240 本器からスタート周波数を取り込む
250 取り込んだ値を変数に代入する
260 本器からストップ周波数を取り込む
270 取り込んだ値を変数に代入する
280 取り込んだスタート周波数をアドレス・ポイントに変換する
290 取り込んだストップ周波数をアドレス・ポイントに変換する
300 帯域内で最大レスポンス（レベル）の周波数をサーチ
310 帯域内で最大のレスポンス（レベル）をサーチ
320 サーチした値を MHz単位に変換する
330 コメントを管面に表示する
340 コメントと最大レスポンスの周波数値を表示する
350 コメントと最大レスポンス値を表示する
360 プログラムの終了

MEMO

7.　コマンド・リファレンス

この章では、本器のすべてのリモート・コマンドの文法（コマンド文法、クエリ文法、または両方）、応答データ・フォーマット（クエリの存在するとき）、およびコマンドの詳細の説明をします。

（注）・コマンドを参照する場合、コマンド・ニーモニックの一部を省略可能なことを考慮に入れて下さい。

例）以下の 2つのコマンドは、表記は違いますが同じものです。

```
SOURCE:SWEEP:TIME 1S
SWEEP:TIME 1S
```

・SWEEP:TIMEという記述からコマンドのリファレンスを参照できなかった場合、付録のコマンド・リストからコマンドの完全な記述を探し、そこからリファレンスを参照して下さい。コマンドの完全な記述がわかっている場合は、目次からの検索が可能です。

7.1　コマンド記述のフォーマットの説明

以降の節でIEEE488.2-1987とIEEE488.1-1987の各コマンド・モードの詳細を説明をします。
以下の注意を参照して下さい。

注意

1. コマンドと応答データ・フォーマットは、以下の記号を用いて記述します。

   記号<>　：文法の構成要素を示す
   　　その内容は、その後に記述される
   記号｜　：複数の中から一つを選択することを示す
   　　例）A ｜B ｜C　　これはA 、B またはC という意味
   記号[ ]：囲まれた項目は、オプション（省略可能）であることを示す
   記号 {}：囲まれた項目は、グループを表し、 {} の中で ｜で区切られた複数の項目の 1つを選択することを示す

2. コマンドとクエリの存在を以下のように記述します。

   Command/Query　:コマンドとクエリがどちらも存在する
   Command　　:コマンドだけが存在する
   Query　　:クエリだけが存在する

3. 4 文字以上のニーモニックはショート・フォームをもちます。本文中では大文字で記述した部分がショート・フォームになります。

   例）SOURce:SWEep:TIME
   　　ｼｮｰﾄ･ﾌｫｰﾑ　：SOUR, SWE
   　　ﾛﾝｸﾞ･ﾌｫｰﾑ　：SOURCE, SWEEP
   　　TIMEは 4文字なのでｼｮｰﾄ･ﾌｫｰﾑ と ﾛﾝｸﾞ･ﾌｫｰﾑ の区別はありません。

続く→

注意

→続き

4. クエリは、コマンドのヘッダに ?をつけます。パラメータを必要とするクエリは、クエリのフォーマットも記述します。

5. この章で共通に用いているパラメータの書式を、以下に示します。

<int> : 数値データでNR1 、NR2 、NR3 の各フォーマットで入力できる
本器が受け取った後、整数に丸め込まれる
<real> : 数値データでNR1,NR2,NR3 の各フォーマットで入力できる
本器が受け取った後、有効な桁数の実数に丸め込まれる
<bool> : 0 ｜OFF ｜1 ｜ON のスイッチ
0 とOFF 、1 とONがそれぞれ対応する
<str> : 文字列
"または 'で囲まれた英数記号を示す(IEEE488.1-1987 コマンド・モードでは "または 'は付けない)
<block> : ブロック・データ型
データの内容は 8bit のバイナリ・データ列

フォーマットは、IEEE488.2-1987コマンド・モードの説明を参照して下さい。

6. パラメータ・ヘッダの一部に付加するパラメータを、以下に示します。
これらは各コマンド共通に使用します。

<chno> : 0 ; アクティブチャンネル
1 ; チャンネル1
2 ; チャンネル2
3 ; チャンネル3
4 ; チャンネル4

(注) サブ・メジャーがOFF のときに、<chno>に 3または4 を指定した場合はエラーとなります。

<trace> : 解析チャンネル

(注) この指定ができるコマンドでは、<chno>の指定は無視されます。
これらの解析チャンネルのうち、コマンドの種類にによって指定可能なチャンネルは限定されます。

CH1	CH2	CH3	CH4	
0	1	4	5	; 表示データ (第1波形)
2	3	6	7	; メモリ・データ (第1波形)
8	9	12	13	; 表示データ (第2波形)
10	11	14	15	; メモリ・データ (第2波形)
32	36	48	52	; LOGMAGデータ
33	37	49	53	; 位相データ
34	38	50	54	; メモリのLOGMAGデータ

続く→

注意

→続き

```
                35  39  51  55 ; メモリの位相データ
                40  44  56  60 ; 実数部
                41  45  57  61 ; 虚数部
                42  46  58  62 ; メモリの実数部
                43  47  59  63 ; メモリの虚数部
                                （以下、複素数データ）
               128 192 256 320 ; フォーマット前のデータ配列
               129 193 257 321 ; データ配列
               130 194 258 322 ; メモリ配列
               131 195 259 323 ; 生データ配列
               133 197 261 325 ; ノーマライズ基準データ配列
               134 198 262 326 ; 方向性エラー係数配列
               135 199 263 327 ; ソース・マッチ・エラー係数配列
               136 200 264 328 ; 反射トラッキング・エラー係数配列
               137 201 265 329 ; 順方向：方向性エラー係数配列
               138 202 266 330 ; 順方向：ソース・マッチ・エラー係数配列
               139 203 267 331 ; 順方向：反射トラッキング・エラー係数配列
               140 204 268 332 ; 順方向：ロード・マッチ・エラー係数配列
               141 205 269 333 ; 順方向：伝送トラッキング・エラー係数配列
               142 206 270 334 ; 順方向：アイソレーション・エラー係数配列
               143 207 271 335 ; 逆方向：方向性エラー係数配列
               144 208 272 336 ; 逆方向：ソース・マッチ・エラー係数配列
               145 209 273 337 ; 逆方向：反射トラッキング・エラー係数配列
               146 210 274 338 ; 逆方向：ロード・マッチ・エラー係数配列
               147 211 275 339 ; 逆方向：伝送トラッキング・エラー係数配列
               148 212 276 340 ; 逆方向：アイソレーション・エラー係数配列
   <input> : 1 ; R チャンネル
             2 ; A チャンネル
             3 ; B チャンネル
   <port>  : 1 ; PORT1
           : 2 ; PORT2
   <eport> : 1 ; R チャンネル
             2 ; A チャンネル
             3 ; B チャンネル
             4 ; PORT1
             5 ; PORT2
   <n>     : n ; 各コマンドによって定義される整数値

             例）CALCulate[<chno>]:FORMat　で、チャンネル1 の測定フォーマ
                 ットを MLOG にするには、以下のようにします。

                 CALCulate1:FORMat MLOG

   <parano>: 表示フォーマットが直交座標系の場合
             0 ; メイン・トレース
             1 ; サブ・トレース
             表示フォーマットが極座標系の場合
             0 ; 振幅または実数部
             1 ; 位相または虚数部
```

7.2　共通コマンド

1. *CLS　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ *CLS

- ●機能　ステータス・バイトと関連データのクリア
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command
- ●コマンド　*CLS
- ●説明　*CLSはステータス・データ構造をクリアし、強制的に*OPCと*OPC? をキャンセルします。また、エラー・キューもクリアします。しかし、このコマンド自身は出力バッファをクリアしないので、出力データがある場合 MAVビットはクリアされません。ただし、行の最初にこのコマンドを実行するとデータがクリアされるので、MAV を含めてすべてのステータスがクリアされます。
  *CLSは、エラー・キューのクリアも実行します。

2. *DDT

●機能	GET に対するマクロ定義
●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在	Command/Query
●コマンド	*DDT <block>
●パラメータ	<block>
●応答形式	<block>
●説明	*DDTは*TRG、またはGET インタフェース・メッセージが受信されたときに実行するコマンド・シーケンスを定義します。 つまり、*TRGの動作を<block> データ中に記述された一連のコマンドと置き換えます。定義できるシーケンスの長さは255 文字以内です。 *DDTで 0の長さのブロック・データ(#10) を定義すると、*TRGおよびGET インタフェース・メッセージで何も実行しないことを定義することになります。また、*RSTの実行でマクロをキャンセルします。 クエリに対する応答は、ブロック・データで応答します。マクロが未定義の状態で*DDT? を実行すると、0 の長さのブロックデータ(#10) が返ります。
●注意	この定義中に*TRGは用いないで下さい。*DDTで定義中に*TRGを用いるとトリガではなく、*DDTで設定したシーケンスを呼び出し、無限ループとなります（実際にはネスティングの制限にかかり、マクロ・エラーになります）。
●例	*DDT #214INIT;TRIG:SIGN のとき *TRG→INIT;TRIG:SIGN　と置き換えます。

3. *DMC

● 機能　　マクロ定義

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　　Command

● コマンド　　*DMC <str>,<block>

● パラメータ　　<str>
<block>

● 説明　　*DMCは<str> で指定されたマクロ・ラベルにコマンド・シーケンスを定義します。この定義で本器は<str> を受信したときに<block> の本体を受信したのと同じ動作を実行するようになります（ただし、*EMC 1でなければならない）。

このマクロ・ラベルには階層コマンドも使用できます。また、あらかじめ定義されているR3764/66、R3765/67コマンドにマクロを上書きすることもできます（ただし、共通コマンドには上書きできない）。このときは*EMC 1でマクロをイネーブルにするとマクロで置き換えた一連のコマンドを、*BMC 0でディセーブルにすると本来の動作を行います。*DMCで定義したマクロの削除は*PMCを用いて下さい。一度登録したマクロは*PMCでクリアされるまで再登録できません。

マクロの本体はR3764/66、R3765/67コマンドの文法に従って記述して下さい。マクロ・コマンドに与えたパラメータは$1～$9で9 個まで表現できます。数字はマクロコマンドに続くパラメータが 1、次が 2と進みます。また、マクロ定義にマクロを含むことができます。最大9 レベルまでのネスティングをサポートしています。登録可能な新規マクロは最大30です（条件によって変化する）。

*PMC, *GMC?, *LMC?, *EMCも参照して下さい。

● 例　　*DMC "SWPINIT",#221FREQ:START $1;STOP $2　のとき

SWPINIT 100MHZ,500MHZ →FREQ:START 100MHZ;STOP 500MHZ
と置き換えます。

4. *EMC

- ●機能　マクロの実行許可
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query
- ●コマンド　*EMC <int>
- ●パラメータ　<int>
- ●応答形式　0 | 1
- ●説明　マクロの実行の許可(1) または禁止(0) をします。このコマンドはマクロの定義内容に影響を与えません。

  このコマンドはマクロで上書きされた本来のコマンドを実行するなどのために使用します。

  *RSTでマクロの実行は禁止されます。

  *DMC, *PMC, *GMC?, *LMC?も参照して下さい。

5. *ESE　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ *ESE

- ●機能　標準イベント・ステータス・イネーブル・レジスタの設定
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query
- ●コマンド　*ESE <int>
- ●パラメータ　<int>
- ●応答形式　NR1 （整数値）
- ●説明　標準イベント・ステータス・レジスタのイネーブル・レジスタを設定します。このレジスタの1 に設定されたbit に対応する標準イベント・ステータス・レジスタが、有効ビットとしてステータス・バイト・レジスタに反映します。

  詳細はステータス・データ構造の説明を参照して下さい。
  *ESR? も参照して下さい。
- ●例　Operation Control(bit3) とDevice Dependent Error(bit0) をイネーブルにセットするとき

  $2^3 + 2^0 = 8 + 1 = 9$ と計算し、*ESE 9とセットします。

6. *ESR?　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ *ESR?

- ●機能　　標準イベント・ステータス・レジスタの読み出し
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　　Query
- ●クエリ　　*ESR?
- ●応答形式　　NR1 （整数値）
- ●説明　　標準イベント・ステータス・レジスタの値を読み出します。
  標準イベント・ステータス・レジスタは読み出すとクリアされ、
  対応するステータスバイトのビット(bit5) もクリアされます。

  詳細はステータス・データ構造の説明を参照して下さい。

スタンダード・イベント・レジスタの割り当て

bit		説明
7	Power on	・電源ONで 1になる
6		・常に 0
5	Command Error	・パーサーが文法エラーを見つけたときに 1にセットされる
4	Execution Error	・GPIBコマンドとして受け取った命令の実行を何らかの理由（パラメータが範囲外など）で失敗すると 1にセットされる
3	Device Dependent Error	・Command Error 、Execution Error 、Quer Error 以外のエラーが発生したとき 1にセットされる
2	Query Error	・コントローラが本器からデータを読み出そうとしたときに、データが存在しない、またはデータが消失していると 1にセットされる
1	Request Control	・本器がアクティブ・コントローラになる必要があるときに 1にセットされる
0	Operation Control	・*OPCコマンドを受け取った後で、かつ本器が実行しているコマンドがなくなると 1にセットされる

7. *GMC?

● 機能　マクロ定義の問い合わせ

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Query

● クエリ　*GMC? <name>

● パラメータ　<name>

● 応答形式　<block>

● 説明　<name>で指定したマクロの定義を読み出します。
未定義の<name>マクロを*GMC?で読み出すと、0 の長さのブロック・データ(#10) が返ります。

*DMC, *PMC, *LMC?, *EMC も参照して下さい。

8. *IDN?　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
IDNT?

● 機能　機器の問い合わせ

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Query

● クエリ　*IDN?
IDNT?

● 応答形式　"<manufacturer>,<model>,<serial number>,<firmware level>"
<manufacturer> = ADVANTEST
<model> = 機種名
<serial number> = シリアル番号
<firmware level> =システム・バージョン

● 説明　本器の識別情報を取り出します。上記の応答形式の項目で記述している4項目を文字列形式で出力します。

9. *LMC?

- ●機能　すべてのマクロ定義の読み出し
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Query
- ●クエリ　*LMC?
- ●応答形式　"<macro label>"[,"<macro label>"...]
  <macro label> = マクロ・ヘッダ
- ●説明　すべてのマクロ・ヘッダを文字列形式で応答します。複数のマクロが定義されているときは ,で区切って並べます。定義されているマクロがない場合は、0文字長の文字列（""）で応答します。

  *DMC, *PMC, *GMC?, *EMC も参照して下さい。

10. *OPC　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ *OPC

- ●機能　実行中のすべての動作の終了の通知
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query
- ●コマンド　*OPC
- ●応答形式　1
- ●説明　*OPCは現在実行中のすべてのコマンドが終了したときに標準イベント・ステータス・レジスタの 'Operation Control' bit を 1に設定します。"現在実行中のすべてのコマンド"が終了する前に次のコマンドを受けとると、そのコマンド実行の終了も待ちます。つまり、*OPCを受けとった後に本器が何も実行していない状態になったときにステータス・レジスタの設定をします。
  *OPC? は上記の*OPCで設定する 'Operation Control' bit の代わりに出力バッファに 1を書き込みます。つまり、コントローラが本器からの応答を受けとるタイミングでコマンド終了のタイミングをとれます。

  *OPC、*OPC? ともに DCLインタフェース・メッセージ、*CLS、および*RSTで解除されます。

  *WAIも参照して下さい。

11. *PCB　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ *PCB

- ●機能　コントローラ権を返すGPIBアドレスの設定
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command
- ●コマンド　*PCB <primary>[,<secondary>]
- ●パラメータ　<primary>
  <secondary>

  （注）IEEE488.1-1987コマンド・モードでは<secondary> は入力できません。常に省略して下さい。
- ●説明　*PCBは本器を接続する外部コントローラのアドレスを設定します。

12. *PMC

- ●機能　すべてのマクロ定義の削除
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command
- ●コマンド　*PMC
- ●説明　*PMCはすべてのマクロ定義を削除します。このコマンドで本器のメモリからすべてのマクロ・ヘッダとマクロ本体が削除され、新しいマクロの登録が可能になります。

  *DDT, *DMC, *GMC?, *LMC?, *EMCも参照して下さい。

13. *RCL　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
RECLREG{1|2|3|4|5|6|7|8|9|10}
RECLPOFF

● 機能　　機器の設定のリコール

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　　Command

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　*RCL {<int> | POFF}

パラメータ　　<int> = レジスタ番号
POFF =前回のパワーオフ時の設定

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　RECLREG{1|2|3|4|5|6|7|8|9|10}
RECLPOFF

● 説明　　本器の設定条件を指定した内部レジスタから呼び出します。
レジスタ番号0 またはPOFF（またはRECLPOFF）は前回のパワーオフ時の設定値を呼び出します。

14. *RST　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
*RST

● 機能　　機器のリセット

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　　Command

● コマンド　　*RST

● 説明　　*RSTは本器のリセットを実行します。実際には以下のことを実行します。
① 本器の設定を初期状態にする（[A3. 初期設定] を参照）
② *DDTで定義されるマクロを初期状態にする
③ マクロを無効にする(*EMC 0 と同じ)
④ *OPC、*OPC? を無効にする。
⑤ トリガ・システムのリセット

以下への影響はありません。
① GPIBバスの状態
② GPIBアドレス
③ 出力バッファ
④ ステータスデータ構造
⑤ *DMCで定義するマクロ
⑥ デバイスの校正データ

SYSTem:PRESet (IP)も参照して下さい。

15. *SAV　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ SAVEREG{1|2|3|4|5|6|7|8|9|10}

● 機能　　機器の設定のセーブ

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　　Command

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　*SAV <int>

　パラメータ　　<int>

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　SAVEREG{1|2|3|4|5|6|7|8|9|10}

● 説明　　本器の設定条件を指定した番号の内部レジスタに記憶します。
内部レジスタは内蔵の電池によりバックアップされます。
ただし校正データはバックアップされません。電源OFF すると校正データおよび関連する設定はクリアされます。
既にレジスタに記憶している場合は、その上にオーバー・ライトします。

16. *SRE　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ *SRE

● 機能　　サービス・リクエスト・イネーブル・レジスタの設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　　Command/Query

● コマンド　　*SRE <int>

● パラメータ　　<int>

● 応答形式　　NR1 （整数値）

● 説明　　サービス・リクエスト・イネーブル・レジスタを設定します。
このレジスタの1 に設定されたbit に対応するステータス・バイト・レジスタが有効ビットとしてMSS に反映します。

クエリ時の応答データbit6は、常に 0となります。

詳細はステータス・データ構造の説明を参照して下さい。
*STB? も参照して下さい。

● 例　　OPR(bit7) 、ESB(bit5) およびMAV(bit4) をイネーブルにセットするとき

$2^7 + 2^5 + 2^4 = 128 + 32 + 16 = 176$
と計算し、*SRE 176　とセットします。

17. *STB?　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ *STB?

● 機能　　ステータス・バイト・レジスタの読み出し

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　　Query

● クエリ　　*STB?

● 応答形式　　NR1 （整数値）

● 説明　　ステータス・バイト・レジスタの内容を読み出します。
ここで読み出されるリクエストの要約ビットはMSS です。
このレジスタとMSS は読み出されてもクリアされません。

詳細はステータス・データ構造の説明を参照して下さい。

ステータス・バイト・レジスタの割り当て

bit		説明
7	OPR	・OPR は、ｽﾀﾝﾀﾞｰﾄﾞ･ｵﾍﾟﾚｰｼｮﾝ･ｽﾃｰﾀｽ･ﾚｼﾞｽﾀ のサマリである
6	MSS	・RQS はステータス・バイト・レジスタの MSSが 1になったときにTRUEとなるが、そのMSS はすべてのステータス・データ構造のサマリ・ビットになっている ・MSS は、サービス・リクエストでは読めない（ただし、RQS が 1のときは MSSが 1であることがわかる） ・MSS を読むには、共通コマンドの*STB? を用いる *STB? ではステータス・バイト・レジスタのbit0～5 、bit7 およびMSS が読み出される この場合ステータス・バイト・レジスタと MSSはクリアされない ・MSS は、ステータス・レジスタ構造のすべてのマスクされていない要因がクリアされるまで0 にならない
5	ESB	・ESB は、スタンダード・イベント・レジスタのサマリである
4	MAV	・MAV は出力バッファの要約ビット ・出力バッファに出力データがある間 1になり、データが読み出されると0 になる
3	QUES	・QUESは、ｸｴｽﾁｮﾅﾌﾞﾙ･ｽﾃｰﾀｽ･ﾚｼﾞｽﾀ のサマリである
2	DEV	・DEV は、デバイス・ステータス・レジスタのサマリである
1～0		・常に0

18. *TRG　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ *TRG

- ●機能　機器にトリガをかける
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command
- ●コマンド　*TRG
- ●説明　*TRGは機器にトリガをかけます。これはGET インタフェース・メッセージと全く同じ効果を発生します。TRIG:SOUR がBUS で本器がトリガ待ちステートになっているときに（[5. トリガ・システム] を参照）*TRGを受けとると、本器は測定を開始します。それ以外のときはこのコマンドは無視されます。

  *TRG、GET インタフェース・メッセージともに入力バッファにつまれ、入力順に処理されます。

19. *TST?　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ *TST?

- ●機能　セルフテストの結果の問い合わせ
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Query
- ●クエリ　*TST?
- ●応答形式　0 | エラー・コード
- ●説明　*TST? は本器にセルフテストを実行させ、その結果を応答します。0 の応答はセルフテストの成功を意味し、それ以外の応答はエラー・コードを意味します。本器の場合、*TST? に対して 0 以外の応答をしません。

20. *WAI　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ *WAI

- ●機能　実行中のすべての動作の終了を待つ
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command
- ●コマンド　*WAI
- ●説明　*WAIは現在実行中のすべてのコマンドが終了するのを待ちます。このコマンドを実行すると、これ以降のすべてのコマンドは現在実行中のコマンドの終了まで遅延されます。

  *WAIは DCLインタフェース・メッセージでキャンセルされます。

7.3　ABORt サブシステム

1. ABORt

● 機能　　トリガ・モジュールのリセット

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　　Command

● コマンド　　ABORt

● 説明　　ABORt コマンドはトリガ・システムをリセットし、トリガ・ステートをアイドル・ステートに強制的にセットします。これに伴い測定は中断し、アベレージ・カウントがリセットされます。また、機器のオペレーション・ペンディング・フラグもこれに伴いクリアされます。

このコマンドはINITiate:CONTinuous を変更しません。そのためCONTinuousがONのときは即座に次のトリガ待ちに移行します。

INITiateサブシステム、TRIGger サブシステムも参照して下さい。

7.4　CALCulate サブシステム

1. CALCulate[<chno>]:FORMat　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
LOGMAG, PHASE, DELAY, LINMAG, SWR, REAL, IMAG,
UNWRAP, LINMP, LOGMP, LOGMD, POLAR, SRJX, SGJB

● 機能　　測定フォーマットの選択

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

コマンド　　CALCulate[<chno>]:FORMat <format>

パラメータ　　<format> = {MLOGarithmic | PHASe | GDELay | MLINear | SWR | REAL | IMAGinary | UPHase | MLIPhase | MLOPhase | MLODelay | POLar | SCHart | ISCHart}

応答形式　　MLOG | PHAS | GDEL | MLIN | SWR | REAL | IMAG | UPH | MLIP | MLOP | MLOD | POL | SCH | ISCH

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

コマンド　　LOGMAG
PHASE
DELAY
LINMAG
SWR
REAL
IMAG
UNWRAP
LINMP
LOGMP
LOGMD
POLAR
SRJX
SGJB

応答形式　　0 | 1

● 説明　　振幅、位相、群遅延等の測定フォーマットを指定します。

初期値 MLOPhase

入力信号は、X+jY型の複素数として測定されますが、測定フォーマットの指定により、次表のように演算処理されます。

R3762/63 コマンド	R3764/66、R3765/67 コマンド・パラメータ	演算式:(単位・比測定/ 絶対値)	内容
LOGMAG	MLOG	$10\ \log_{10}(X^2+Y^2)$:(dB/dBm)	振幅 (対数)
PHASE	PHAS	arctan(Y/X):(deg/deg)	位相
DELAY	GDEL	$\frac{-\Delta\ (\text{位相})}{360\times\Delta\ (\text{周波数})}$:(Sec/Sec)	群遅延
LINMAG	MLIN	$\sqrt{X^2+Y^2}$:(Unit/Vrms)	振幅
SWR	SWR	$\frac{1+\Gamma}{1-\Gamma}$: (Unit/Unit) $\Gamma = \sqrt{X^2+Y^2}$	反射係数
REAL	REAL	X:(Unit/Unit)	実数部
IMAG	IMAG	Y:(Unit/Unit)	虚数部
UNWRAP	UPH	arctan(Y/X):(deg/deg)	位相 PHASE は±180° の範囲内であり、UNWRAPは±180° では折り返さずに最初の測定ポイントを基準とした連続値となる
LINMP	MLIP	対(r1, r2) r1= $\sqrt{X^2+Y^2}$:(Unit/Vrms) r2= arctan (Y/X) :(deg/deg)	振幅、位相の対 直行座標表示
LOGMP	MLOP	対(r1, r2) r1= $10\ \log_{10}(X^2+Y^2)$:(dB/dBm) r2= arctan (Y/X) :(deg/deg)	振幅 (対数)、位相の対 直行座標表示
LOGMD	MLOD	対(r1, r2) r1= $10\ \log_{10}(X^2+Y^2)$:(dB/dBm) r2= $\frac{-\Delta\ (\text{位相})}{360\times\Delta\ (\text{周波数})}$:(Sec/Sec)	振幅 (対数)、群遅延の対 直行座標表示
POLAR	POLar	X:(Unit/Unit) Y:(Unit/Unit)	実数部 虚数部
SRJX	SCHart	X:(Unit/Unit) Y:(Unit/Unit)	実数部 虚数部
SGJB	ISCHart	X:(Unit/Unit) Y:(Unit/Unit)	実数部 虚数部

2. CALCulate[<chno>]:GDAPerture:APERture　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ APERTP

● 機能　群遅延のアパーチャの設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● コマンド　CALCulate[<chno>]:GDAPerture:APERture <real>
APERTP<real>

● パラメータ　<real>

● 応答形式　NR3 （実数値）

● 説明　群遅延のアパーチャを設定します。

初期値　10%
設定範囲　0.01% ～50%
設定分解能　0.01%

群遅延は、以下の計算式で演算されますが、Δ（周波数）を「アパーチャ」と言います。

$$\text{群遅延} = \frac{-\Delta(\text{位相})}{360 \times \Delta(\text{周波数})}$$

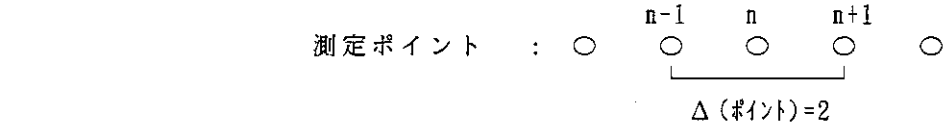
アパーチャ（Δ(周波数)）は、測定ポイント（横軸）に変換され、設定値<real>に対して、以下の式で決定されます。

$$\Delta(\text{周波数}) = \Delta(\text{ﾎﾟｲﾝﾄ}) = \frac{\text{測定ﾎﾟｲﾝﾄ数}-1}{100} \times <\text{real}>$$

つまり、設定値<real>は測定ポイント数に対する% で設定されます。
測定ポイント数が変更されてもこの数値は保持され、変更後の測定ポイント数でΔポイントを内部的に再計算します。

● 例　測定ポイント数 ： 101ポイント

$$\text{アパーチャ} : 2(\%) \rightarrow \Delta(\text{ﾎﾟｲﾝﾄ}) = \frac{101-1}{100} \times 2 = 2$$



3. CALCulate[<chno>]:MATH[:EXPRession]:NAME　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ DISPDDM

● 機能　　データ（+ - × /）メモリの設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

コマンド　　CALCulate[<chno>]:MATH[:EXPRession]:NAME <type>

パラメータ　　<type> = {NONE | DDM | DMM | DAM | DSM }

応答形式　　NONE | DDM | DMM | DAM | DSM

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

コマンド　　DISPDDM<bool>

パラメータ　　<bool> = {ON | OFF }

応答形式　　0 | 1

● 説明　　測定データとメモリ・データ間の演算をします。

R3762/63 コマンド	R3764/66、R3765/67 ｺﾏﾝﾄﾞ･ﾊﾟﾗﾒｰﾀ	演算
DISPDDM ON	DDM	÷
	DMM	×
	DAM	+
	DSM	−
DISPDDM OFF	NONE	なし

● 注意　　演算は、同一チャンネル内のデータとメモリ間にだけ有効です。（異なるチャンネル間での演算はできない）

データの正規化（ノーマライズ）はDDM(÷）です。

演算は、フォーマット前のベクトル量（複素数データ）に対して行われます。

4. CALCulate[<chno>]:SMOothing:APERture　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ SMOOAPER

● 機能　スムージングの区間の設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● コマンド　CALCulate[<chno>]:SMOothing:APERture <real>
SMOOAPER<real>

● パラメータ　<real>

● 応答形式　NR3 （実数値）

● 説明　スムージング・アパーチャを設定します。

初期値　10%
設定範囲　0.01% ～50%
設定分解能　0.01%

スムージングは、以下のアルゴニズムで与えられます。
(2m)を「アパーチャ」と言います。

スムージングのアルゴリズム

$$\bar{D}(n) = \frac{D(n-m) + \cdots + D(n) + \cdots + D(n+m)}{2m+1}$$

$\bar{D}(n)$ ： スムージングされた n番目のﾌｫｰﾏｯﾄ後のデータ
D(n) ： スムージングされる前の n番目のデータ
2m ： スムージング・アパーチャ

アパーチャは、設定値<real>に対して以下の式で与えられます。

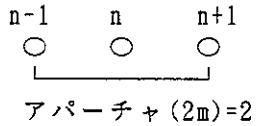
$$\text{アパーチャ}(2m) = \frac{(\text{測定ポイント数})-1}{100} \times \langle real \rangle$$

つまり、設定値<real>は測定ポイント数に対する% で設定されます。
測定ポイント数が変更されてもこの設定値<real>は保持され、変更後の測定ポイント数でアパーチャ(2m)を内部的に再計算します。

● 例　測定ポイント数 ： 101ポイント

アパーチャ　：2(%) →アパーチャ(2m)= $\frac{101-1}{100} \times 2 = 2$

測定ポイント　：



アパーチャ(2m)=2

5. CALCulate[<chno>]:SMOothing:STATe　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ SMOO

●機能　スムージングのON/OFF

●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

●コマンド　CALCulate[<chno>]:SMOothing:STATe <bool>
SMOO<bool>

●パラメータ　<bool>

●応答形式　0 | 1

●説明　スムージングを実行します。
スムージングは、フォーマットされたデータの隣接データ間の移動平均を行います。

ノイズ成分にスムージングを行うと、ノイズの平均値を求めることになります。
これに対して、アベレージングは、フォーマット前のデータ（ベクトル量）の時間平均を求めるため、ノイズは平均値ではなく減少します。

スムージングのアルゴリズム

$$\bar{D}(n) = \frac{D(n-m) + \cdots + D(n) + \cdots + D(n+m)}{2m+1}$$

$\bar{D}(n)$ ： スムージングされた n番目のﾌｫｰﾏｯﾄ後のデータ
$D(n)$ ： スムージングされる前の n番目のデータ
$2m$ ： スムージング・アパーチャ

●注意　測定フォーマットの設定が 2トレース(MLOP,MLOD,MLIP)の場合、またはメモリ・トレースONの場合は、それらすべてのトレースに対してスムージング機能が実行されます。

●例　測定ポイント数 ： 101ポイント

アパーチャ　： 2(%) →アパーチャ(2m)= $\frac{101-1}{100} \times 2 = 2$

測定ポイント　： ○ ○(n-1) ○(n) ○(n+1) ○

アパーチャ(2m)=2

6. CALCulate[<chno>]:TRANsform:IMPedance:CIMPedance　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
SETZ0
MKRZ0{50|75}

● 機能　　Z 変換の特性インピーダンスの設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　　Command/Query

● コマンド　　CALCulate[<chno>]:TRANsform:IMPedance:CIMPedance <real>
SETZ0<real>
MKRZ0{50|75}

● パラメータ　　<real>

● 応答形式　　NR3 （実数値）
0 | 1　（MKRZ0{50|75} のとき）

● 説明　　インピーダンス測定を行う場合の特性インピーダンスを設定します。

初期値　　50Ω
設定範囲　　100pΩ～1GΩ
設定分解能　0.001pΩ

測定値は、測定系の特性インピーダンスで正規化(1Ω）された値で求めます。
したがって、絶対値を求めるには、測定系の特性インピーダンスを指定する必要があります。

● 例　　反射係数よりインピーダンスを求める場合

正規化インピーダンス　：$\frac{1+\Gamma}{1-\Gamma}\times 1(\Omega)$

絶対値インピーダンス　：$\frac{1+\Gamma}{1-\Gamma}\times Z_0$

Γ ： 反射係数
Z_0 ： 特性ｲﾝﾋﾟｰﾀﾞﾝｽ

7. CALCulate[<chno>]:TRANsform:IMPedance:TYPE　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
CONV{OFF|RZ|RY|TZ|TY|1DS}

● 機能　　Z 変換の型の設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　CALCulate[<chno>]:TRANsform:IMPedance:TYPE <type>

パラメータ　　<type> = {NONE | ZREFlection | YREFlection | ZTRansmit | YTRansmit | INVersion }

応答形式　　NONE | ZREF | YREF | ZTR | YTR | INV

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　CONV{OFF|RZ|RY|TZ|TY|1DS}

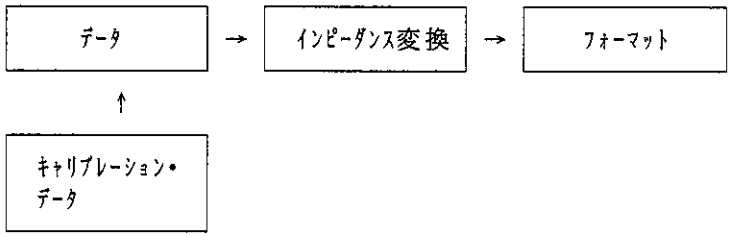
応答形式　　0 | 1

● 説明　　反射係数、伝送特性よりインピーダンスを以下の表から求めます。

R3762/63 3 マンド	R3764/66、R3765/67 ｺﾏﾝﾄﾞ･ﾊﾟﾗﾒｰﾀ	変換した値	変換式
CONVOFF	NONE	変換なし	
CONVRZ	ZREF	反射ｲﾝﾋﾟｰﾀﾞﾝｽ	$\frac{1+\Gamma}{1-\Gamma}\times Z_0$
CONVRY	YREF	反射ｱﾄﾞﾐｯﾀﾝｽ	$\frac{1-\Gamma}{1+\Gamma}\times \frac{1}{Z_0}$
CONVTZ	ZTR	伝送ｲﾝﾋﾟｰﾀﾞﾝｽ	$\frac{2(1-T)}{T}\times Z_0$
CONVTY	YTR	伝送ｱﾄﾞﾐｯﾀﾝｽ	$\frac{T}{2(1-T)}\times \frac{1}{Z_0}$
CONV1DS	INV	逆S ﾊﾟﾗﾒｰﾀ	$\frac{1}{S}$

Γ : 反射係数
T : 利得
S : Γまたは T
Z_0: 特性ｲﾝﾋﾟｰﾀﾞﾝｽ

●注意　　データ処理フローは、以下のようになります。



7.5　DISPlay サブシステム

1. DISPlay:ACTive　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
CH{1|2|3|4}

● 機能　　アクティブ・チャンネルの指定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　DISPlay:ACTive <int>

パラメータ　　<int>

応答形式　　NR1(整数値)

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　CH{1|2|3|4}

応答形式　　0 | 1

● 説明　　アクティブ・チャンネルを選択します。（初期設定チャンネル1）
本器には 4つの測定チャンネルがあり、測定、データ表示など独立して行えます。
チャンネルに依存する機能では、IEEE488.2-1987コマンドのヘッダ・パラメータ<chno>で指定が可能です。<chno>を省略した場合、またはIEEE488.1-1987コマンド使用時には、コマンドはここで設定したアクティブ・チャンネルに対して適用されます。

R3762/63 コマンド	R3764/65、R3765/67 ｺﾏﾝﾄﾞ･ﾊﾟﾗﾒｰﾀ	動作
CH1	1	チャンネル1 がアクティブ
CH2	2	チャンネル2 がアクティブ
CH3	3	チャンネル3 がアクティブ
CH4	4	チャンネル4 がアクティブ

（注）サブ・メジャーがOFF の場合には、サブ・チャンネルをアクティブに切り換えることができません。あらかじめサブ・メジャーをONにしておく必要があります。
サブ・メジャーをON/OFFした際に、アクティブ・チャンネルが自動的に切り換わる場合があります。
(7.10.19　[SENSe:]FUNCtion[<chno>][:ON] 、
7.10.20　[SENSe:]FUNCtion[<chno>]:POWerを参照して下さい。)

2. DISPlay:DUAL　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ DUAL

● 機能　　デュアル・チャンネルのON/OFF

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　　Command/Query

● コマンド　　DISPlay:DUAL <bool>
DUAL<bool>

● パラメータ　　<bool>

● 応答形式　　0 | 1

● 説明　　2 つの測定チャンネル(CH1、CH2)を同時に表示するか、または片方のチャンネルだけを表示するかを選択します。
サブ・メジャーが選択されている場合は、チャンネル3 、チャンネル4 も表示されます。

初期値　DUAL OFF

3. DISPlay:FORMat　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ SPLIT

● 機能　　スプリット／オバーラップの選択

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

コマンド　　DISPlay:FORMat <type>

パラメータ　　<type> = {ULOWer | FBACk }

応答形式　　ULOW | FBAC

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

コマンド　　SPLIT<bool>

パラメータ　　<bool> = {ON | OFF }

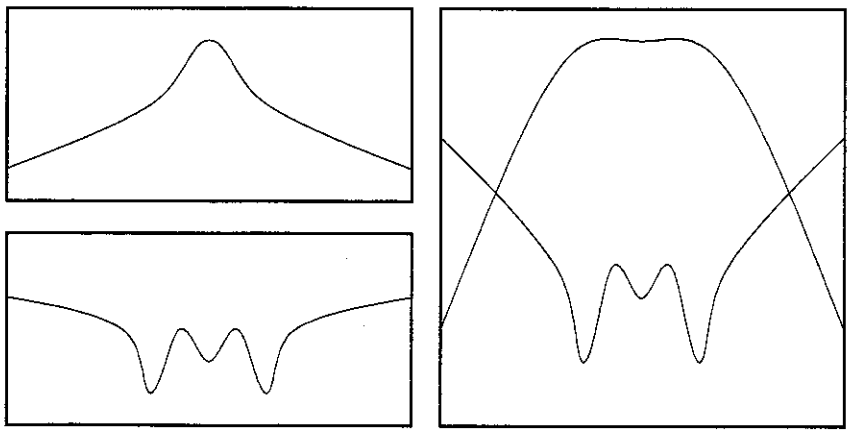
応答形式　　0 | 1

● 説明　　スプリット表示、オーバラップ表示の選択を行います。

初期値　SPLIT OFF

R3762/63 コマンド	R3764/66、R3765/67 ｺﾏﾝﾄﾞ･ﾊﾟﾗﾒｰﾀ	動作
SPLIT ON	ULOW	スプリット表示
SPLIT OFF	FBAC	オーバラップ表示

● 例　　スプリット表示の場合　　オーバラップ表示の場合



4. DISPlay[:WINDow[<chno>]]:TEXT[:DATA]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ LABEL

- ●機能　ラベルの設定
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query
- ●コマンド　DISPlay[:WINDow[<chno>]]:TEXT[:DATA] {<str> | <block> }
  LABEL<str>
- ●パラメータ　{<str> | <block> }
- ●応答形式　<str> = 文字列データ
- ●説明　ラベルを設定します。
  ラベルはアクティブ・チャンネルに対して設定されます。
  設定可能文字数　80文字

5. DISPlay[:WINDow[<chno>]]:TRACe:ASSign　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ DISP{DATA|MEM|DM}

- ●機能　トレース表示のON/OFF
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query
- ●IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
  - コマンド　DISPlay[:WINDow[<chno>]]:TRACe:ASSign <type>
  - パラメータ　<type> = {DATA | MEMory | DMEMory }
  - 応答形式　DATA | MEM | DMEM
- ●IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
  - コマンド　DISP{DATA|MEM|DM}
  - 応答形式　0 | 1
- ●説明　トレース表示の種類を指定します。
  初期値　DISPDATA

R3762/63 コマンド	R3764/66、R3765/67 ｺﾏﾝﾄﾞ･ﾊﾟﾗﾒｰﾀ	動作
DISPDATA	DATA	ﾃﾞｰﾀ･ﾄﾚｰｽのみ表示
DISPMEM	MEM	ﾒﾓﾘ･ﾄﾚｰｽのみ表示
DISPDM	DMEM	ﾃﾞｰﾀ･ﾄﾚｰｽとﾒﾓﾘ･ﾄﾚｰｽを両方表示

6. DISPlay[:WINDow[<chno>]]:TRACe:GRATicule[:STATe] IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ GRAT

● 機能　目盛表示のON/OFF

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● コマンド　DISPlay[:WINDow[<chno>]]:TRACe:GRATicule[:STATe] <bool>
GRAT<bool>

● パラメータ　<bool>

● 応答形式　0 | 1

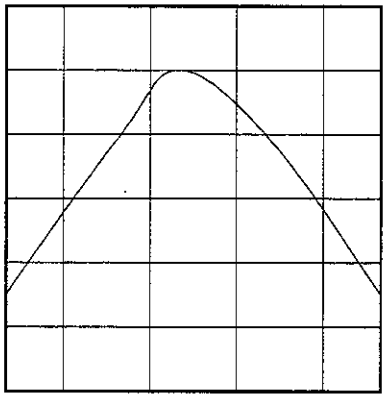
● 説明　目盛を表示する/ 表示しないを選択します。

初期値　GRAT ON

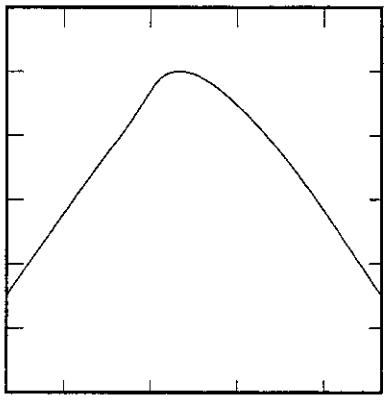
R3762/63 コマンド	R3764/66、R3765/67 ｺﾏﾝﾄﾞ･ﾊﾟﾗﾒｰﾀ	動作
GRAT ON	ON	目盛を表示する
GRAT OFF	OFF	目盛を表示しない

● 例

GRAT ON の場合



GRAT OFFの場合



7. DISPlay[:WINDow[<chno>]]:Y[<trace>]:RLINe　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ REFL

● 機能　　Y 軸リファレンス・ライン表示のON/OFF

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　　Command/Query

● コマンド　　DISPlay[:WINDow[<chno>]]:Y[<trace>]:RLINe <bool>
REFL<bool>

● パラメータ　　<bool>

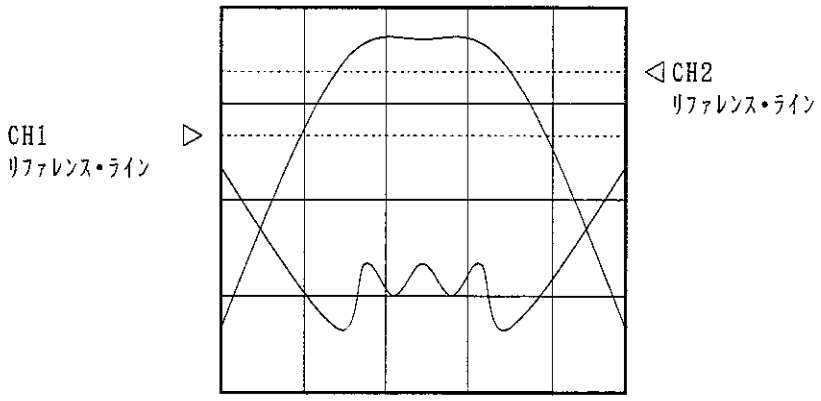
● 応答形式　　0 | 1

● 説明　　Y 軸リファレンス・ライン表示のON/OFFを選択します。
Y 軸リファレンス・ラインは、Y 軸目盛の基準値を示します。

初期値　REFL ON

R3762/63 コマンド	R3764/66、R3765/67 ｺﾏﾝﾄﾞ･ﾊﾟﾗﾒｰﾀ	動作
REFL ON	ON	Y 軸ﾘﾌｧﾚﾝｽ･ﾗｲﾝを表示する
REFL OFF	OFF	Y 軸ﾘﾌｧﾚﾝｽ･ﾗｲﾝを表示しない

● 例



8. DISPlay[:WINDow[<chno>]]:Y[<trace>][:SCALe]:AUTO　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
AUTO
SCALF{1ST|2ND}

● 機能　　Y 軸の自動設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　　Command

● コマンド　　DISPlay[:WINDow[<chno>]]:Y[<trace>][:SCALe]:AUTO ONCE
AUTO
SCALF{1ST|2ND}

● パラメータ　　ONCE

● 説明　　Y 軸の設定を自動調整します。
このコマンド実行直前の全表示データが、スケール画面内に入る最適値に設定されます。（設定更新されるのはPDIV,RLEV のみ）
<trace> およびIEEE488.1-1987コマンド・モードの
SCALF{1ST|2ND} は、測定フォーマットが 2トレース(MLOP,MLOD,MLIP) の場合のスケール変更対象のトレース指定です。測定フォーマットが 2トレースでない場合はこの指定は無視されます。

<trace>=0　CH1の第1 波形 ┐
=1　CH2の第1 波形 │
=4　CH3の第1 波形 ┘ SCALF1ST
=5　CH4の第1 波形
=8　CH1の第2 波形 ┐
=9　CH2の第2 波形 │
=12 CH3の第2 波形 ┘ SCALF2ND
=13 CH4の第2 波形

第1 波形…表示フォーマットがLOGMAG&PHASE、LOGMAG&DELAYのときのLOGMAG、
LINMAG&PHASEのときのLINMAG、
メジャー・モードがS11&S21(FWD)のときのS11 、
S22&S12(REV)のときのS22

第2 波形…表示フォーマットがLOGMAG&PHASEのときのPHASE 、
LOGMAG&DELAYのときのDELAY 、
LINMAG&DELAYのときのDELAY 、
メジャー・モードがS11&S21(FWD)のときのS21 、
S22&S12(REV)のときのS12

9. DISPlay[:WINDow[<chno>]]:Y[<trace>][:SCALe]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
:PDIVision　SDIV
SCALF{1ST|2ND}

● 機能　Y 軸グリッド間隔の設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● コマンド　DISPlay[:WINDow[<chno>]]:Y[<trace>][:SCALe]:PDIVision <real>
SDIV<real>
SCALF{1ST|2ND}

● パラメータ　<real>

● 応答形式　NR3 （実数値）

● 説明　Y 軸グリッド間隔（1 目盛当たりのスケール）の値を設定します。極座標およびスミス・チャート表示では、機能しません。
<trace> およびIEEE488.1-1987コマンド・モードの SCALF{1ST|2ND} は、測定フォーマットが 2トレース(MLOP,MLOD,MLIP) の場合のスケール変更対象のトレース指定です。測定フォーマットが 2トレースでない場合はこの指定は無視されます。

<trace>=0　CH1の第1 波形 ┐
=1　CH2の第1 波形 ┤
=4　CH3の第1 波形 ┤ SCALF1ST
=5　CH4の第1 波形 ┘
=8　CH1の第2 波形 ┐
=9　CH2の第2 波形 ┤
=12 CH3の第2 波形 ┤ SCALF2ND
=13 CH4の第2 波形 ┘

第1 波形…表示フォーマットがLOGMAG&PHASE、LOGMAG&DELAYのときのLOGMAG、
LINMAG&PHASEのときのLINMAG、
メジャー・モードがS11&S21(FWD)のときのS11 、
S22&S12(REV)のときのS22

第2 波形…表示フォーマットがLOGMAG&PHASEのときのPHASE 、
LOGMAG&DELAYのときのDELAY 、
LINMAG&DELAYのときのDELAY 、
メジャー・モードがS11&S21(FWD)のときのS21 、
S22&S12(REV)のときのS12

初期値は測定フォーマットごとに異なります。[A3.　初期設定] を参照して下さい。

10. DISPlay[:WINDow[<chno>]]:Y[<trace>][:SCALe]:RLEVel　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
REFV
SCALF{1ST|2ND}

●機能　Y 軸リファレンス・レベルの設定

●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

●コマンド
DISPlay[:WINDow[<chno>]]:Y[<trace>][:SCALe]:RLEVel <real>
REFV<real>
SCALF{1ST|2ND}

●パラメータ　<real>

●応答形式　NR3 （実数値）

●説明
Y 軸リファレンス・ラインのレベルを設定します。
Y 軸リファレンス・ラインは、Y 軸目盛の基準値を示します。
極座標およびスミスチャート表示では、外周円におけるフルスケール値となります。
<trace> およびIEEE488.1-1987コマンド・モードの
SCALF{1ST|2ND} は、測定フォーマットが 2トレース(MLOP,MLOD,MLIP) の場合のスケール変更対象のトレース指定です。測定フォーマットが 2トレースでない場合はこの指定は無視されます。

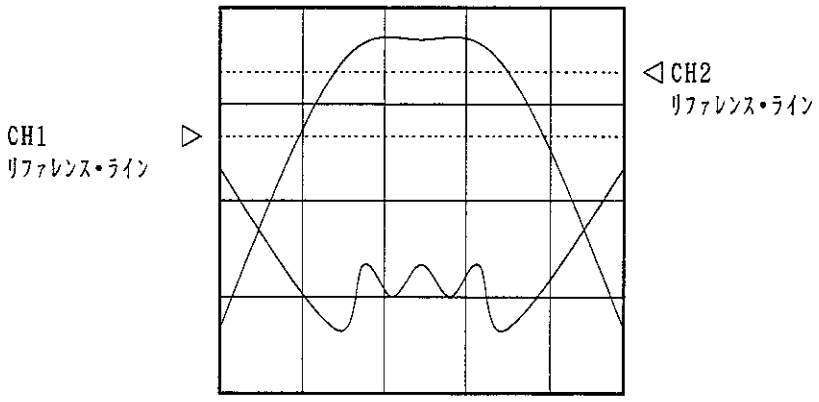
<trace>=0　CH1の第1 波形 ┐
=1　CH2の第1 波形 ┤
=4　CH3の第1 波形 ┘ SCALF1ST
=5　CH4の第1 波形
=8　CH1の第2 波形 ┐
=9　CH2の第2 波形 ┤
=12 CH3の第2 波形 ┘ SCALF2ND
=13 CH4の第2 波形

第1 波形…表示フォーマットがLOGMAG&PHASE、LOGMAG&DELAYのときのLOGMAG、
LINMAG&PHASEのときのLINMAG、
メジャー・モードがS11&S21(FWD)のときのS11 、
S22&S12(REV)のときのS22

第2 波形…表示フォーマットがLOGMAG&PHASEのときのPHASE 、
LOGMAG&DELAYのときのDELAY 、
LINMAG&DELAYのときのDELAY 、
メジャー・モードがS11&S21(FWD)のときのS21 、
S22&S12(REV)のときのS12

初期値は測定フォーマットごとに異なります。[A3.　初期設定]を参照して下さい。

● 例



11. DISPlay[:WINDow[<chno>]]:Y[<trace>][:SCALe] IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
:RPOSition REFP
SCALF{1ST|2ND}

● 機能　Y 軸リファレンス・ラインの位置指定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● コマンド
DISPlay[:WINDow[<chno>]]:Y[<trace>][:SCALe]:RPOSition <real>
REFP<real>
SCALF{1ST|2ND}

● パラメータ　<real> = 0 - 100

● 応答形式　NR3 （実数値）

● 説明
Y 軸リファレンス・ラインの位置を指定します。
<trace> およびIEEE488.1-1987コマンド・モードの
SCALF{1ST|2ND} は、測定フォーマットが 2トレース(MLOP,MLOD,
MLIP) の場合のスケール変更対象のトレース指定です。測定フォーマットが 2トレースでない場合はこの指定は無視されます。

<trace>=0 CH1の第1 波形 ┐
=1 CH2の第1 波形 │
=4 CH3の第1 波形 ┘ SCALF1ST
=5 CH4の第1 波形
=8 CH1の第2 波形 ┐
=9 CH2の第2 波形 │
=12 CH3の第2 波形 ┘ SCALF2ND
=13 CH4の第2 波形

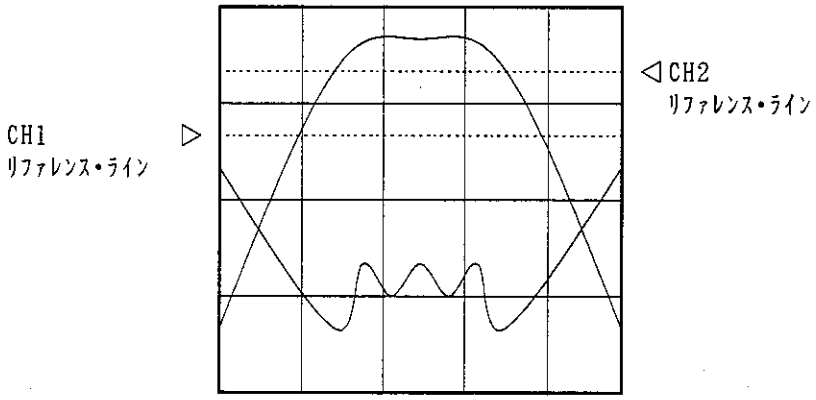
第1 波形…表示フォーマットがLOGMAG&PHASE、LOGMAG&DELAYのときのLOGMAG、
LINMAG&PHASEのときのLINMAG、
メジャー・モードがS11&S21(FWD)のときのS11 、
S22&S12(REV)のときのS22

第2 波形…表示フォーマットがLOGMAG&PHASEのときのPHASE 、
LOGMAG&DELAYのときのDELAY 、
LINMAG&DELAYのときのDELAY 、
メジャー・モードがS11&S21(FWD)のときのS21 、
S22&S12(REV)のときのS12

初期値は測定フォーマットごとに異なります。[A3. 初期設定]を参照して下さい。

画面のトップを100%として、パーセンテージで指定します。画面のセンタが50% 、一番下が0%になります。

● 例



7.6　FILEサブシステム

1. FILE:DELete　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ PURGE

- ●機能　　ストア・ファイルの消去
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command
- ●IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
  コマンド　　FILE:DELete <str>

  パラメータ　　<str> = ファイル名
- ●IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
  コマンド　　PURGE<str>

  パラメータ　　<str> = ファイル名
- ●説明　　FILE:STORe、またはSTFILEによって保存されたファイルを消去します。

2. FILE:LOAD　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ LDFILE

● 機能　　ストア・ファイルのロード（再生）

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　FILE:LOAD <str>

　パラメータ　　<str> = ファイル名

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　LDFILE<str>

　パラメータ　　<str> = ファイル名

● 説明　　FILE:STORe、またはSTFILEによって保存されたファイルを再生します。

指定したファイルがFILE:STATe:RAW、またはFILE:STATe:DATA がONで保存されたものである場合、測定波形データも再生するため、再生後は無条件に掃引がホールド・モードとなります。

3. FILE:STATe:CONDition　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ DSSTATE

- ●機能　　ストア・ファイル保存情報の定義
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　　Command/Query
- ●コマンド　　FILE:STATe:CONDition <bool>
  DSSTATE <bool>
- ●パラメータ　　<bool>
- ●応答形式　　0 | 1
- ●説明　　FILE:STOReでの保存にて、設定条件を保存する／しないを選択します。

4. FILE:STATe:CORRection　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ CORARY

- ●機能　　ストア・ファイル保存情報の定義
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　　Command/Query
- ●コマンド　　FILE:STATe:CORRection <bool>
  CORARY <bool>
- ●パラメータ　　<bool>
- ●応答形式　　0 | 1
- ●説明　　FILE:STOReでの保存にて、校正データを保存する／しないを選択します。

5. FILE:STATe:DATA　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
DATAARY

- ●機能　ストア・ファイル保存情報の定義
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query
- ●コマンド　FILE:STATe:DATA <bool>
  DATAARY <bool>
- ●パラメータ　<bool>
- ●応答形式　0 | 1
- ●説明　FILE:STOReでの保存にて、測定波形データを保存する／しないを選択します。

6. FILE:STATe:MEMory　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ MEMARY

- ●機能　ストア・ファイル保存情報の定義
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query
- ●コマンド　FILE:STATe:MEMory <bool>
  MEMARY <bool>
- ●パラメータ　<bool>
- ●応答形式　0 | 1
- ●説明　FILE:STOReでの保存にて、メモリ波形データを保存する／しないを選択します。

7. FILE:STATe:RAW　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
RAWARY

- ●機能　ストア・ファイル保存情報の定義
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query
- ●コマンド　FILE:STATe:RAW <bool>
  RAWARY <bool>
- ●パラメータ　<bool>
- ●応答形式　0 | 1
- ●説明　FILE:STOReでの保存にて、測定波形の生データを保存する／しないを選択します。

8. FILE:STORe　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ STFILE

● 機能　　ストア・ファイルの保存

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　FILE:STORe <str>

　パラメータ　　<str> = ファイル名

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　STFILE<str>

　パラメータ　　<str> = ファイル名

● 説明　　本器の設定条件、校正データ、および波形データ等をフロッピー・ディスクに保存します。

　　保存する情報は、FILE:STATeによって定義します。詳細は、FILE:STATeコマンドを参照して下さい。

7.7　FORMatサブシステム

1. FORMat:BORDer　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ FORM{0|2|3|5|6|7|8}

● 機能　　バイト順序の設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　FORMat:BORDer <border>

　パラメータ　　<border> = {NORMal | SWAPped }

　応答形式　　NORM | SWAP

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　FORM{0|2|3|5|6|7|8}

　応答形式　　なし

● 説明　　FORMat:BORDer(FORM{0|2|3|5|6|7|8}) コマンドは、TRACe:DATA コマンドで入出力するデータのフォーマットを設定します。このコマンドの説明はFORMat[:DATA] でまとめて行います。

　次項2.の FORMat[:DATA]の説明を参照して下さい。

2. FORMat[:DATA]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
FORM{0|2|3|5|6|7|8}

● 機能　　データ・フォーマット

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　FORMat[:DATA] <format>,<len>

パラメータ　　<format> = {ASCii | REAL | MBINary}
<len> = {32 | 64}

応答形式　　{ASC | REAL | MBIN} ,<int>
<int> = NR1 （整数値）

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　FORM{0|2|3|5|6|7|8}

応答形式　　なし

● 説明　　FORMat[:DATA] コマンドは、FORMat:BORDer コマンドと組み合わせて使用します。これらのコマンドを用いることによって、TRACe:DATAコマンドによるトレース・データ入出力のフォーマットが変更されます（IEEE488.1-1987コマンド・モードでは FORM {0 | 2 | 3 | 5 | 6 | 7 | 8 } コマンドを用い、IN {1 | 2 } etc. 、OT {1 | 2 } etc.コマンドの入出力フォーマットを変更する）。
これらのコマンドの組合せによるデータ転送のフォーマットは、下表のようになります。BORDerが NORMal のときはデータのハイ・バイトから順に、SWAPped のときはロー・バイトから順に転送します。

（注） NEC 製パーソナル・コンピュータでN88BASICをお使いになる場合のバイナリ・フォーマットは、マイクロソフト浮動小数点フォーマットを使用して下さい。

FORM:DATA	FORM:BORD	
	NORMal	SWAPped
ASCii	ASCII(FORM0)	
REAL,32	IEEE 32bit ﾊﾞｲﾅﾘ(FORM2)	IEEE 32bit ﾊﾞｲﾅﾘ-順序入換え(FORM5)
REAL,64	IEEE 64bit ﾊﾞｲﾅﾘ(FORM3)	IEEE 64bit ﾊﾞｲﾅﾘ-順序入換え(FORM6)
MBIN,32	マイクロソフト単精度浮動小数点数バイナリ(FORM7)	
MBIN,64	マイクロソフト倍精度浮動小数点数バイナリ(FORM8)	

7.8　INITiateサブシステム

1. INITiate:CONTinuous

● 機能　トリガ・システム・コンティニューのON/OFF

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● コマンド　INITiate:CONTinuous <bool>

● パラメータ　<bool>

● 応答形式　0 | 1

● 説明　INITiate:CONTinuous コマンドは、トリガ・システムのスタートをコントロールします。

CONTinuousがONのときは、アイドル・ステートに戻らずにトリガ待ちになります。
CONTinuousがOFF のときは、アイドル・ステートを経由します。このときトリガ待ちステートになるには、INITiate[:IMMediate]コマンドを使います。

詳細は[5. トリガ・システム] を参照して下さい。

2. INITiate[:IMMediate]

● 機能　トリガ・システムのスタート

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command

● コマンド　INITiate[:IMMediate]

● 説明　INITiate[:IMMediate]コマンドは、トリガ・システムをスタートさせます。

トリガ・システムは、アイドル・ステートからトリガ待ちステートになり、イベントの発生を待ち始めます。

詳細は[5. トリガ・システム] を参照して下さい。

7.9　REGisterサブシステム

1. REGister:CLEar　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
CLRREG{1|2|3|4|5|6|7|8|9|10}

● 機能　　レジスタの消去

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　REGister:CLEar <int>

　パラメータ　　<int>

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　CLRREG{1|2|3|4|5|6|7|8|9|10}

● 説明　　*SAV、REGister:SAVE <int>、SAVEREG{1|2|3|4|5|6|7|8|9|10}
によって保存されたレジスタを消去します。

2. REGister:RECall　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ RECLREG{1|2|3|4|5|6|7|8|9|10}

●機能　　レジスタのリコール（再生）

●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command

●IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　REGister:RECall {<int>|POFF}

　パラメータ　　<int> = レジスタ番号
　　POFF = 前回の電源オフ時の設定

●IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　RECLREG{1|2|3|4|5|6|7|8|9|10}

●説明　　*SAV、REGister:SAVE <int>、SAVEREG{1|2|3|4|5|6|7|8|9|10}によって保存されたレジスタの内容を再生します。

　　このコマンドは、*RCLと同じ動作です。

3. REGister:SAVE　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
SAVEREG{1|2|3|4|5|6|7|8|9|10}

● 機能　　レジスタのセーブ（保存）

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　REGister:SAVE <int>

　パラメータ　　<int>

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　SAVEREG{1|2|3|4|5|6|7|8|9|10}

● 説明　　本器の設定条件、および校正データを指定した番号のレジスタに保存します。

　このコマンドは、*SAVと同じ動作です。

7.10　SENSeサブシステム

1. [SENSe:]AVERage[<chno>]:COUNt　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
AVERFACT
AVR{2|4|8|16|32|64|128}

● 機能　　アベレージ回数の設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　[SENSe:]AVERage[<chno>]:COUNt <int>

パラメータ　　<int>

応答形式　　NR1 （整数値）

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　AVERFACT<int>
AVR{2|4|8|16|32|64|128}

パラメータ　　<int>

応答形式　　NR1 （AVERFACTコマンド）
0 | 1 （AVR{2|4|8|16|32|64|128} コマンド）

● 説明　　アベレージ回数を設定します。

アベレージは、測定されたフォーマット前のデータに時間的な重みを付けて平均化します。ベクトル量による平均化を行っているのでノイズ・レベルを下げる効果もあります。

アベレージのプロセスを以下に示します。

$$\overline{Y}(n)=\frac{n-1}{n}\cdot\overline{Y}(n-1)+\frac{1}{n}\cdot Y(n)\quad(n\leqq N)$$

$$\overline{Y}(n)=\frac{N-1}{N}\cdot\overline{Y}(n-1)+\frac{1}{N}\cdot Y(n)\quad(n>N)$$

$\overline{Y}(n)$ ： n 回目のアベレージされたデータ
$Y(n)$ ： n 回目のデータ
N ： アベレージ回数

2. [SENSe:]AVERage[<chno>]:RESTart　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ AVERREST

● 機能　　アベレージの再スタート

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　　Command

● コマンド　　[SENSe:]AVERage[<chno>]:RESTart
AVERREST

● 説明　　アベレージ・カウンタをクリアし、アベレージを再スタートします。

アベレージは、測定されたフォーマット前のデータに時間的な重みを付けて平均化します。ベクトル量による平均化を行っているのでノイズ・レベルを下げる効果もあります。

アベレージのプロセスを以下に示します。

$$\overline{Y}(n)=\frac{n-1}{n}\cdot\overline{Y}(n-1)+\frac{1}{n}\cdot Y(n) \qquad (n \leq N)$$

$$\overline{Y}(n)=\frac{N-1}{N}\cdot\overline{Y}(n-1)+\frac{1}{N}\cdot Y(n) \qquad (n > N)$$

$\overline{Y}(n)$:　n 回目のアベレージされたデータ
$Y(n)$:　n 回目のデータ
N　:　アベレージ回数

3. [SENSe:]AVERage[<chno>][:STATe]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
AVERAGE
AVER

● 機能　　アベレージのON/OFF

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　　Command/Query

● コマンド　　[SENSe:]AVERage[<chno>][:STATe] <bool>
AVERAGE
AVER<bool>

● パラメータ　　<bool>

● 応答形式　　0 | 1

● 説明　　アベレージのON/OFFを設定します。

初期設定　OFF

アベレージは、測定されたフォーマット前のデータに時間的な重みを付けて平均化します。ベクトル量による平均化を行っているのでノイズ・レベルを下げる効果もあります。

アベレージのプロセスを以下に示します。

$$\overline{Y}(n) = \frac{n-1}{n} \cdot \overline{Y}(n-1) + \frac{1}{n} \cdot Y(n) \quad (n \leq N)$$

$$\overline{Y}(n) = \frac{N-1}{N} \cdot \overline{Y}(n-1) + \frac{1}{N} \cdot Y(n) \quad (n > N)$$

$\overline{Y}(n)$:　n 回目のアベレージされたデータ
$Y(n)$:　n 回目のデータ
N :　アベレージ回数

R3762/63コマンドのAVERAGE はAVER OFFと同等です。

● 参考　　スムージングは、フォーマットされた隣接データ間の移動平均を求めます。スカラ量の平均のため、ノイズ幅は小さくなりますが、ノイズ・レベルを下げる効果はありません。

4. [SENSe:]BANDwidth[<chno>][:RESolution]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
RBW
RBW{1K|300|100|30|10}HZ

● 機能　　バンド幅の設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　[SENSe:]BANDwidth[<chno>][:RESolution] <int>

パラメータ　　<int>

応答形式　　NR1 （整数値）

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　RBW<int>
RBW{1K|300|100|30|10}HZ

パラメータ　　<int>

応答形式　　NR1 (RBWコマンド)
0 | 1 （RBW{1K|300|100|30|10}HZコマンド）

● 説明　　受信部の分解能帯域幅を設定します。

初期設定　10KHz

下表のように、分解能帯域幅は 10kHz～ 3Hzの範囲で選択します。また、分解能帯域幅により、1 ポイント当たりの最高掃引速度と雑音レベルが決定されます。

分解能帯域幅	1ﾎﾟｲﾝﾄ当たりの最高掃引速度
10kHz	0.1ms/POINT
3kHz	0.35ms/POINT
1kHz	1.0ms/POINT
300Hz	3.5ms/POINT
100Hz	10ms/POINT
30Hz	35ms/POINT
10Hz	100ms/POINT
3Hz	350ms/POINT

● 説明　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞで、分解能帯域幅 10kHz、3kHz、3Hz を設定するには、RBW コマンドで設定、クエリして下さい。

5. [SENSe:]BANDwidth[<chno>][:RESolution]:AUTO　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ RBWAUTO

● 機能　　バンド幅の自動設定

● ｺﾏﾝﾄﾞ と ｸｴﾘ の存在　　Command/Query

● コマンド　　[SENSe:]BANDwidth[<chno>][:RESolution]:AUTO <bool>
RBWAUTO

● パラメータ　　<bool>

● 応答形式　　0 | 1

● 説明　　分解能帯域幅を測定周波数に応じて、自動設定します。

分解能帯域幅により、1 ポイント当たりの最高掃引速度と雑音レベルが決定されます。

分解能帯域幅	1ﾎﾟｲﾝﾄ当たりの最高掃引速度
10kHz	0.1ms/POINT
3kHz	0.35ms/POINT
1kHz	1.0ms/POINT
300Hz	3.5ms/POINT
100Hz	10ms/POINT
30Hz	35ms/POINT
10Hz	100ms/POINT
3Hz	350ms/POINT

● 注意　　分解能帯域幅により、1 ポイント当たりの最高掃引速度が決定されます。特に低周波数では、分解能帯域幅も小さい値となり、掃引速度が遅くなるので、必要以上に周波数を低く設定しないで下さい。

6. [SENSe:]CORRection[<chno>]:COLLect[:ACQuire]

IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
NORM, NORMS
OPEN, SHORT, LOAD
S11OPEN, S11SHORT, S11LOAD,
S22OPEN, S22SHORT, S22LOAD,
FWDTRNS, FWDMATCH,
REVTRNS, REVMATCH,
OMITISO, FWDISO, REVISO

● 機能　校正データの取得

● ｺﾏﾝﾄﾞ と ｸｴﾘ の存在　Command

● コマンド　[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] <standard>
{NORM | SNOR} ,S11O, S11S, S11L, S22O, S22S, S22L, FTR, FMAT, RTR, RMAT, GTHRU, OIS, FIS, RIS <bool>
OPEN, SHORT, LOAD, S11OPEN, S11SHORT, S11LOAD, S22OPEN, S22SHORT, S22LOAD, FWDTRNS, FWDMATCH, REVTRNS, REVMATCH, OMITISO, FWDISO, REVISO

● パラメータ　<standard> = {NORMalize | SNORmalize | OPEN | SHORt | LOAD}

● 説明　校正データを取得します。

このコマンドを実行すると、掃引を再スタートして、校正データを取得します。
アベレージ機能がONに設定されている場合は、アベレージ指定回数分掃引した後、校正データを取得します。
すでに校正データが取得済の場合は、校正データが更新されます。
ただし、1 ポート・フル・キャリブレーション、2 ポート・フル・キャリブレーションが実行状態では、校正データは更新されません。このとき1 度校正データをクリアする必要があります。

(1/2)

R3762/63 コマンド	R3764/66、R3765/67 ｺﾏﾝﾄﾞ･ﾊﾟﾗﾒｰﾀ	動作 (取得データ)
NORM ON	NORM	ﾉｰﾏﾗｲｽﾞ 取得と完了を同時に行う
NORMS ON	SNOR	ｼｮｰﾄ･ﾉｰﾏﾗｲｽﾞ 取得と完了を同時に行う
OPEN	OPEN	1ﾎﾟｰﾄ･ﾌﾙ･ｷｬﾘﾌﾞﾚｰｼｮﾝ ｵｰﾌﾟﾝ･ﾃﾞｰﾀ
SHORT	SHOR	1ﾎﾟｰﾄ･ﾌﾙ･ｷｬﾘﾌﾞﾚｰｼｮﾝ ｼｮｰﾄ･ﾃﾞｰﾀ
LOAD	LOAD	1ﾎﾟｰﾄ･ﾌﾙ･ｷｬﾘﾌﾞﾚｰｼｮﾝ ﾛｰﾄﾞ･ﾃﾞｰﾀ
S11OPEN	S11O	2ﾎﾟｰﾄ･ﾌﾙ･ｷｬﾘﾌﾞﾚｰｼｮﾝ ｵｰﾌﾟﾝ･ﾃﾞｰﾀ(S11)
S11SHORT	S11S	2ﾎﾟｰﾄ･ﾌﾙ･ｷｬﾘﾌﾞﾚｰｼｮﾝ ｼｮｰﾄ･ﾃﾞｰﾀ(S11)
S11LOAD	S11L	2ﾎﾟｰﾄ･ﾌﾙ･ｷｬﾘﾌﾞﾚｰｼｮﾝ ﾛｰﾄﾞ･ﾃﾞｰﾀ (S11)
S22OPEN	S22O	2ﾎﾟｰﾄ･ﾌﾙ･ｷｬﾘﾌﾞﾚｰｼｮﾝ ｵｰﾌﾟﾝ･ﾃﾞｰﾀ(S22)
S22SHORT	S22S	2ﾎﾟｰﾄ･ﾌﾙ･ｷｬﾘﾌﾞﾚｰｼｮﾝ ｼｮｰﾄ･ﾃﾞｰﾀ(S22)
S22LOAD	S22L	2ﾎﾟｰﾄ･ﾌﾙ･ｷｬﾘﾌﾞﾚｰｼｮﾝ ﾛｰﾄﾞ･ﾃﾞｰﾀ (S22)

(2/2)

R3762/63 コマンド	R3764/66、R3765/67 ｺﾏﾝﾄﾞ･ﾊﾟﾗﾒｰﾀ	動作（取得データ）
FWDTRNS	FTR	2ﾎﾟｰﾄ･ﾌﾙ･ｷｬﾘﾌﾞﾚｰｼｮﾝ 順方向ｽﾙｰ 特性ﾃﾞｰﾀ
FWDMATCH	FMAT	2ﾎﾟｰﾄ･ﾌﾙ･ｷｬﾘﾌﾞﾚｰｼｮﾝ 順方向ﾎﾟｰﾄ･ﾏｯﾁﾝｸﾞ 特性ﾃﾞｰﾀ
REVTRNS	RTR	2ﾎﾟｰﾄ･ﾌﾙ･ｷｬﾘﾌﾞﾚｰｼｮﾝ 逆方向ｽﾙｰ 特性ﾃﾞｰﾀ
REVMATCH	RMAT	2ﾎﾟｰﾄ･ﾌﾙ･ｷｬﾘﾌﾞﾚｰｼｮﾝ 逆方向ﾎﾟｰﾄ･ﾏｯﾁﾝｸﾞ 特性ﾃﾞｰﾀ
——	GTHRU	2ﾎﾟｰﾄ･ﾌﾙ･ｷｬﾘﾌﾞﾚｰｼｮﾝ 上記の4つ(伝送特性)をまとめて取得
OMITISO	OIS	2ﾎﾟｰﾄ･ﾌﾙ･ｷｬﾘﾌﾞﾚｰｼｮﾝ ｱｲｿﾚｰｼｮﾝ･ﾃﾞｰﾀ(OMIT)
FWDISO	FIS	2ﾎﾟｰﾄ･ﾌﾙ･ｷｬﾘﾌﾞﾚｰｼｮﾝ ｱｲｿﾚｰｼｮﾝ･ﾃﾞｰﾀ(Foword)
REVISO	RIS	2ﾎﾟｰﾄ･ﾌﾙ･ｷｬﾘﾌﾞﾚｰｼｮﾝ ｱｲｿﾚｰｼｮﾝ･ﾃﾞｰﾀ(Reverse)

7. [SENSe:]CORRection[<chno>]:COLLect:DELete　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
CLEAR

● 機能　　校正データのクリア

● ｺﾏﾝﾄﾞ と ｸｴﾘ の存在　　Command

● コマンド　　[SENSe:]CORRection[<chno>]:COLLect:DELete
CLEAR

● 説明　　校正データをクリアします。

1 ポート・フル・キャリブレーション、2 ポート・フル・キャリブレーションでは、一度校正を終了すると、校正データをクリアするまで、再度校正データの取得ができません。したがって、再度校正を取り直すには、校正データをクリアしてから実行します。なお、校正データをクリアするには、補正測定OFF の状態で実行します。

8. [SENSe:]CORRection[<chno>]:COLLect:SAVE　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
DONE
DONE1PORT
DONE2PORT

● 機能　　校正データから誤差係数を算出

● ｺﾏﾝﾄﾞ と ｸｴﾘ の存在　　Command

● コマンド　　[SENSe:]CORRection[<chno>]:COLLect:SAVE
DONE
DONE1PORT
DONE2PORT

● 説明　　取得した校正データから誤差係数を算出し、補正測定機能をONにします。

9. [SENSe:]CORRection[<chno>]:CSET:STATe　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ CORRECT

- ●機能　補正測定のON/OFF
- ●ｺﾏﾝﾄﾞ と ｸｴﾘ の存在　Command/Query
- ●コマンド　[SENSe:]CORRection[<chno>]:CSET:STATe <bool>
  CORRECT<bool>
- ●パラメータ　<bool>
- ●応答形式　0 | 1
- ●説明　校正データを用いた補正測定のON/OFFを選択します。

  すでに校正データが取得済の場合は、このコマンドで補正測定を実行します。また、OFF にしても、校正データは保存しているので、随時ONにして補正測定を実行できます。

10. [SENSe:]CORRection[<chno>]:EDELay:DISTance　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ LENGVAL

- ●機能　電気長（距離）の設定
- ●ｺﾏﾝﾄﾞ と ｸｴﾘ の存在　Command/Query
- ●コマンド　[SENSe:]CORRection[<chno>]:EDELay:DISTance <real>
  LENGVAL<real>
- ●パラメータ　<real>
- ●応答形式　NR3 （実数値）
- ●説明　電気長補正の値を距離で設定します。

$$\text{補正量}\Phi(\text{deg}) = \frac{L}{c} \times \frac{1}{V_f} \times f \times 360 = S \times f \times 360$$

L　：電気長（距離）
V_f：伝搬定数
c　：光速度
f　：周波数
S　：電気長（時間）

11. [SENSe:]CORRection[<chno>]:EDELay:STATe　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ LENGTH

● 機能　電気長補正のON/OFF

● ｺﾏﾝﾄﾞ と ｸｴﾘ の存在　Command/Query

● コマンド　[SENSe:]CORRection[<chno>]:EDELay:STATe <bool>
LENGTH<bool>

● パラメータ　<bool>

● 応答形式　0 | 1

● 説明　電気長補正のON/OFFを選択します。

設定された電気長に応じた位相変化量を、測定データに対して補正します。
接続ケーブルの位相変化を取り除き、被測定物だけの位相変化を測定するために使用します。

$$\text{補正量}\Phi(\text{deg}) = \frac{L}{c} \times \frac{1}{V_f} \times f \times 360 = S \times f \times 360$$

L　：電気長（距離）
V_f：伝搬定数
c　：光速度
f　：周波数
S　：電気長（時間）

12. [SENSe:]CORRection[<chno>]:EDELay[:TIME]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ ELED

● 機能　電気長（時間）の設定

● ｺﾏﾝﾄﾞ と ｸｴﾘ の存在　Command/Query

● コマンド　[SENSe:]CORRection[<chno>]:EDELay[:TIME] <real>
ELED<real>

● パラメータ　<real>

● 応答形式　NR3 （実数値）

● 説明　電気長の値を時間で設定します。

$$\text{補正量}\Phi(\text{deg}) = \frac{L}{c} \times \frac{1}{V_f} \times f \times 360 = S \times f \times 360$$

L　：電気長（距離）
V_f：伝搬定数
c　：光速度
f　：周波数
S　：電気長（時間）

13. [SENSe:]CORRection[n]:GPHase:STATe　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ SRCCOR

● 機能　受信部の周波数特性補正のON/OFF

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　[SENSe:]CORRection[n]:GPHase:STATe <bool>

　パラメータ　<bool>

　応答形式　0 | 1

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　INPCOR<bool>

　パラメータ　<bool>

　応答形式　0 | 1

● 説明　受信部の周波数特性の補正をするか(ON)、しないか(OFF) を選択します。

14. [SENSe:]CORRection[<chno>]:OFFSet:PHASe　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ PHAO

● 機能　位相オフセット値の設定

● ｺﾏﾝﾄﾞ と ｸｴﾘ の存在　Command/Query

● コマンド　[SENSe:]CORRection[<chno>]:OFFSet:PHASe <real>
PHAO<real>

● パラメータ　<real>

● 応答形式　NR3 （実数値）

● 説明　位相オフセットの値を設定します。

位相データに一定の値を加算します。電気長補正と異なり、周波数に関係なく設定された値を常に補正します。

● 注意　0 を設定すると、CORR:OFFS:STATが自動的にOFF となります。
0 以外を設定すると、CORR:OFFS:STATが自動的にONとなります。

15. [SENSe:]CORRection[<chno>]:OFFSet:STATe　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ PHAOFS

- ●機能　位相オフセット機能のON/OFF
- ●ｺﾏﾝﾄﾞ と ｸｴﾘ の存在　Command/Query
- ●コマンド　[SENSe:]CORRection[<chno>]:OFFSet:STATe <bool>
  PHAOFS<bool>
- ●パラメータ　<bool>
- ●応答形式　0 | 1
- ●説明　位相オフセット機能のON/OFFを選択します。

  位相データに一定の値を加算します。電気長補正と異なり、周波数に関係なく設定された値を常に補正します。
- ●注意　OFF を設定すると、CORR:OFFS:PHASが自動的に0 となります。

16. [SENSe:]CORRection[<chno>]:PEXTension:TIME[<eport>]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ EPORT{R|A|B|1|2}

- ●機能　測定端面延長補正値の設定
- ●ｺﾏﾝﾄﾞ と ｸｴﾘ の存在　Command/Query
- ●コマンド　[SENSe:]CORRection[<chno>]:PEXTension:TIME[<eport>] <real>
  EPORT{R|A|B|1|2} <real>
- ●パラメータ　<real>
- ●応答形式　NR3 （実数値）
- ●説明　測定端面延長の値を設定します。

  入力ポートに対応した延長補正をします。電気長補正では、単純に設定された値に対して補正をしますが、このコマンドでは入力ポートに対応した値を設定すると、そのときの入力ポート条件にあった値で補正します。

  例えば、反射測定の場合はポート延長値の 2倍、伝送測定の場合はポート延長値の 1倍と、補正値を自動設定します。

17. [SENSe:]CORRection[<chno>]:PEXTension:STATe　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ PORE

● 機能　測定端面延長補正のON/OFF

● ｺﾏﾝﾄﾞ と ｸｴﾘ の存在　Command/Query

● コマンド　[SENSe:]CORRection[<chno>]:PEXTension:STATe <bool>
PORE<bool>

● パラメータ　<bool>

● 応答形式　0 | 1

説明　測定端面延長補正機能のON/OFFを選択します。

入力ポートに対応した延長補正をします。電気長補正では、単純に設定された値に対して補正をしますが、このコマンドでは入力ポートに対応した値を設定すると、そのときの入力ポート条件にあった値で補正します。

例えば、反射測定の場合はポート延長値の 2倍、伝送測定の場合はポート延長値の 1倍と、補正値を自動設定します。

18. [SENSe:]CORRection[<chno>]:RVELocity:COAX　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ VELOFACT

● 機能　ケーブル伝搬定数の設定

● ｺﾏﾝﾄﾞ と ｸｴﾘ の存在　Command/Query

● コマンド　[SENSe:]CORRection[<chno>]:RVELocity:COAX <real>
VELOFACT<real>

● パラメータ　<real>

● 応答形式　NR3 （実数値）

● 説明　ケーブル伝搬定数の値を設定します。

$$\text{補正量}\Phi(\text{deg}) = \frac{L}{c}\times\frac{1}{V_f}\times f\times 360 = S\times f\times 360$$

$$V_f = \frac{1}{\sqrt{\varepsilon_R}}$$

L ： 電気長（距離）
V_f ： 伝搬定数
c ： 光速度
f ： 周波数
S ： 電気長（時間）
ε_R ： 比誘電率

19. [SENSe:]FUNCtion[<chno>][:ON]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
{R|A|B|AR|BR|AB|BDC|BDCR}IN,
S11,S12,S21,S22,SFWD,SREV
SMEAS

● 機能　　メジャー・モードの指定、およびサブ・メジャーのON/OFF

● ｺﾏﾝﾄﾞ と ｸｴﾘ の存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

コマンド　　[SENSe:]FUNCtion[<chno>][:ON] <input>

パラメータ　　<input> = {"POWer:{AC | DC} {1 | 2 | 3 } " |
"POWer:{AC | DC}:RATio {2,1 | 3,1 | 3,2 } " |
"POWer:{S11 | S12 | S22 | S21 | SFWD | SREV} " |
"POWer:NONE" }

応答形式　　"POW:{AC | DC} {1 | 2 | 3 } " |
"POW:{AC | DC}:RAT {2,1 | 3,1 | 3,2 } " |
"POW:{S11 | S12 | S22 | S21 | SFWD | SREV} " |
"POW:NONE"

● IEEE488.1-1987ｺﾏﾝﾄﾞ･･ﾓｰﾄﾞ

コマンド　　{R|A|B|AR|BR|AB|BDC|BDCR}IN
S11,S12,S21,S22,SFWD,SREV
SMEAS<bool>

応答形式　　0 | 1

● 説明　　測定・解析の対象となるメジャー・モードの指定、およびサブ・メジャーのON/OFFの切り換えを行います。

IEEE488.2-1987コマンド・モードでは、チャンネルを<chno>で指定してメジャー・モードを指定します。サブ・メジャーがOFF のときに、<chno>に 3または4 を指定するとサブ・メジャーがONになります。
サブ・メジャーのモード設定は、<chno>に 3または4 を指定するか、アクティブ・チャンネルを 3または4 にしてから行います。
サブ・メジャーをOFF する場合は、アクティブ・チャンネルを 3または4 にするか、<chno>に 3または4 を指定してパラメータ"POW:NONE"を設定します。
このとき、アクティブ・チャンネルは対応するメイン・チャンネルに切り換わります。

IEEE488.1-1987コマンド・モードでは、アクティブ・チャンネルに対して設定が行われます。あらかじめアクティブ・チャンネルを切り換えてから設定して下さい。
サブ・メジャーをONする場合は、SMEASON を送ります。この際、サブ・メジャーのモードは、対応するメイン・メジャーのモードと同じになり、アクティブ・チャンネルはサブ・チャンネルに切り換わります。
サブ・メジャーをOFF する場合は、SMEASOFFを送ります。アクティブ・チャンネルは対応するメイン・チャンネルに切り換わります。

（注）サブ・メジャーがOFF のときは、サブ・チャンネルをアクティブに切り換えることができません。

R3762/63 コマンド	R3764/66、R3765/67 ｺﾏﾝﾄﾞ･ﾊﾟﾗﾒｰﾀ	動作（入力ポート）
RIN	"POW:AC 1"	R 入力を設定する
AIN	"POW:AC 2"	A 入力を設定する
BIN	"POW:AC 3"	B 入力を設定する
ARIN	"POW:AC:RAT 2,1"	A/R 入力（比測定）を設定する
BRIN	"POW:AC:RAT 3,1"	B/R 入力（比測定）を設定する
ABIN	"POW:AC:RAT 2,3"	A/B 入力（比測定）を設定する
BDCIN	"POW:DC 3"	B(DC) 入力を設定する
BDCRIN	"POW:DC:RAT 3,1"	B(DC)/R 入力を設定する
S11	"POW:S11"	S11 を設定する
S12	"POW:S12"	S12 を設定する
S21	"POW:S21"	S21 を設定する
S22	"POW:S22"	S22 を設定する
SFWD	"POW:SFWD"	S11&S21(REFL&TRANS) を設定する
SREV	"POW:SREV"	S22&S12 を設定する
SMEASON	<chno> に3または4を指定	サブ・メジャーをONする
SMEASOFF	"POW:NONE"	サブ・メジャーをOFF する

* [7.5.1　DISPlay:ACTive]も参照して下さい。

20. [SENSe:]FUNCtion[<chno>]:POWer　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
{R|A|B|AR|BR|AB|BDC|BDCR}IN,
S11,S12,S21,S22,SFWD,SREV
SMEAS

● 機能　　メジャー・モードの指定、およびサブ・メジャーのON/OFF

● ｺﾏﾝﾄﾞ と ｸｴﾘ の存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　[SENSe:]FUNCtion[<chno>]:POWer <input>

パラメータ　　<input> = {R|A|B|AR|BR|AB|BDC|BDCR|S11|S12|S21|S22|
SFWD|SREV|NONE}

応答形式　　R|A|B|AR|BR|AB|BDC|BDCR|S11|S12|S21|S22|SFWD|SREV|NONE

● IEEE488.1-1987ｺﾏﾝﾄﾞ･･ﾓｰﾄﾞ
コマンド　　{R|A|B|AR|BR|AB|BDC|BDCR}IN
S11,S12,S21,S22,SFWD,SREV
SMEAS<bool>

応答形式　　0 | 1

● 説明　　測定・解析の対象となるメジャー・モードの指定、およびサブ・メジャーのON/OFFの切り換えを行います。

IEEE488.2-1987コマンド・モードでは、チャンネルを<chno>で指定してメジャー・モードを指定します。サブ・メジャーがOFF のときに、<chno>に 3または4 を指定するとサブ・メジャーがONになります。
サブ・メジャーのモード設定は、<chno>に 3または4 を指定するか、アクティブ・チャンネルを 3または4 にしてから行います。
サブ・メジャーをOFF する場合は、アクティブ・チャンネルを 3または4 にするか、<chno>に 3または4 を指定してパラメータNONEを設定します。
このとき、アクティブ・チャンネルは対応するメイン・チャンネルに切り換わります。

IEEE488.1-1987コマンド・モードでは、アクティブ・チャンネルに対して設定が行われます。あらかじめアクティブ・チャンネルを切り換えてから設定して下さい。
サブ・メジャーをONする場合は、SMEASON を送ります。この際、サブ・メジャーのモードは、対応するメイン・メジャーのモードと同じになり、アクティブ・チャンネルはサブ・チャンネルに切り換わります。
サブ・メジャーをOFF する場合は、SMEASOFFを送ります。アクティブ・チャンネルは対応するメイン・チャンネルに切り換わります。

（注）サブ・メジャーがOFF のときは、サブ・チャンネルをアクティブに切り換えることができません。

R3762/63 コマンド	R3764/66、R3765/67 ｺﾏﾝﾄﾞ･ﾊﾟﾗﾒｰﾀ	動作（入力ポート）
RIN	R	R 入力を設定する
AIN	A	A 入力を設定する
BIN	B	B 入力を設定する
ARIN	AR	A/R 入力（比測定）を設定する
BRIN	BR	B/R 入力（比測定）を設定する
ABIN	AB	A/B 入力（比測定）を設定する
BDCIN	BDC	B(DC) 入力を設定する
BDCRIN	BDCR	B(DC)/R 入力を設定する
S11	S11	S11 を設定する
S12	S12	S12 を設定する
S21	S21	S21 を設定する
S22	S22	S22 を設定する
SFWD	SFWD	S11&S21(REFL&TRANS) を設定する
SREV	SREV	S22&S12 を設定する
SMEASON	<chno> に3または4を指定	サブ・メジャーをONする
SMEASOFF	NONE	サブ・メジャーをOFF する

＊ [7.5.1　DISPlay:ACTive]も参照して下さい。

7.11　SOURce サブシステム

1. [SOURce:]CORRection[n]:GAIN:STATe　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ SRCCOR

● 機能　　信号源部の周波数特性補正のON/OFF

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　[SOURce:]CORRection[n]:GAIN:STATe <bool>

　パラメータ　　<bool>

　応答形式　　0 | 1

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　SRCCOR<bool>

　パラメータ　　<bool>

　応答形式　　0 | 1

● 説明　　信号源部の周波数特性の補正をするか(ON)、しないか(OFF) を選択します。

2. [SOURce:]COUPle　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ COUPLE

● 機能　出力信号のチャンネル間結合のON/OFF

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● コマンド　[SOURce:]COUPle <bool>
COUPLE<bool>

● パラメータ　<bool>

● 応答形式　0 | 1

● 説明　測定チャンネル1 、2 の測定条件を同じにするか、または、別々に設定するかを選択します。

初期設定　COUPLE ON

ここでの測定条件とは、以下の項目です。
・掃引タイプ
・周波数
・出力レベル
・掃引時間
・測定ポイント数
・分解能帯域幅

COUPLE OFFの場合、測定チャンネル1 の測定を実行して、次に測定チャンネル2 の測定を実行します。
つまり、チャンネル1 、2 を交互に測定します。
サブ・メジャーが選択されているとき、チャンネル3 とチャンネル1 、チャンネル4 とチャンネル2 は、COUPLE ON/OFF に関係なく常に同時測定します。

COUPLE ON の場合、測定チャンネル1 、2 を同時に測定します。
サブ・メジャーが選択されているときは、 4画面同時測定します。

3. [SOURce:]FREQuency[<chno>]:CENTer　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ CENTERF

- ●機能　中心周波数の設定
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query
- ●コマンド　[SOURce:]FREQuency[<chno>]:CENTer <real>
  CENTERF<real>
- ●パラメータ　<real>
- ●応答形式　NR3 （実数値）
- ●説明　周波数掃引時の中心周波数を設定します。

  初期設定　1.92GHz(R3764/66)
  4.02GHz(R3765/67)
  設定範囲　20MHz～3.8GHz(R3764/66)
  20MHz～8.0GHz(R3765/67)
  設定分解能 1Hz

4. [SOURce:]FREQuency[<chno>]:CW　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ CWFREQ

- ●機能　固定周波数の設定
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query
- ●コマンド　[SOURce:]FREQuency[<chno>]:CW <real>
  CWFREQ<real>
- ●パラメータ　<real>
- ●応答形式　NR3 （実数値）
- ●説明　レベル掃引時の周波数を設定します。

  初期設定　1GHz(R3764/66)
  1GHz(R3765/67)
  設定範囲　20MHz～3.8GHz(R3764/66)
  20MHz～8.0GHz(R3765/67)
  設定分解能 1Hz

5. [SOURce:]FREQuency[<chno>]:MODE　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
LINFREQ
LOGFREQ

● 機能　　掃引タイプの設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　[SOURce:]FREQuency[<chno>]:MODE <mode>

パラメータ　　<mode> = SWEep

応答形式　　CW | SWE | PSW

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　LINFREQ
LOGFREQ

応答形式　　0 | 1

● 説明　　このコマンドは下表のように、組みで設定します。
初期設定　リニア周波数掃引

ｺﾏﾝﾄﾞ	PSW:MODE	FREQ:MODE	POW:MODE	SWE:SPAC	掃引タイプ	対応する R3762/63 ｺﾏﾝﾄﾞ
ﾊﾟﾗﾒｰﾀ	(NONE)	SWE	(FIX)	LIN	リニア周波数掃引	LINFREQ
				LOG	ログ周波数掃引	LOGFREQ
		(CW)	SWE	(LIN)	レベル掃引	LEVEL
	FREQ	(PSW)	(FIX)	(LIN)	ﾌﾟﾛｸﾞﾗﾑ掃引(周波数のみ)	USRFSWP
	ALL	(PSW)	(PSW)	(LIN)	プログラム掃引	USRARWP

(注) 表中の( ) はクエリ時に返る値であり、その値での設定はできません。

掃引タイプ　：リニア周波数掃引…固定レベルにて周波数を等間隔で掃引
：ログ周波数掃引　…固定レベルにて周波数を対数間隔で掃引
：レベル掃引　　…固定周波数にて出力レベルを掃引
：プログラム掃引　…区間ごとに周波数のみを任意に設定
(周波数のみ)
：プログラム掃引　…区間ごとに周波数、出力レベル、分解能帯域幅、セットリング時間を任意に設定

ただし、R3764 、R3766 ではログ周波数掃引の設定はできません。

6. [SOURce:]FREQuency[<chno>]:SPAN　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
SPANF

- ●機能　スパン周波数の設定
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query
- ●コマンド　[SOURce:]FREQuency[<chno>]:SPAN <real>
  SPANF<real>
- ●パラメータ　<real>
- ●応答形式　NR3 （実数値）
- ●説明　周波数掃引のスパン周波数を設定します。

  初期設定　3.76GHz(R3764/66)
  7.96GHz(R3765/67)
  設定範囲　0～3.78GHz(R3764/66)
  0～7.98GHz(R3765/67)
  設定分解能 1Hz

7. [SOURce:]FREQuency[<chno>]:STARt　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
STARTF

- ●機能　スタート周波数の設定
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query
- ●コマンド　[SOURce:]FREQuency[<chno>]:STARt <real>
  STARTF<real>
- ●パラメータ　<real>
- ●応答形式　NR3 （実数値）
- ●説明　周波数掃引のスタート周波数を設定します。

  初期設定　40MHz(R3764/66)
  40MHz(R3765/67)
  設定範囲　20MHz～3.8GHz(R3764/66)
  20MHz～8.0GHz(R3765/67)
  設定分解能 1Hz

8. [SOURce:]FREQuency[<chno>]:STOP　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ STOPF

● 機能　ストップ周波数の設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● コマンド　[SOURce:]FREQuency[<chno>]:STOP <real>
STOPF<real>

● パラメータ　<real>

● 応答形式　NR3 （実数値）

● 説明　周波数掃引のストップ周波数を設定します。

初期設定　3.8GHz(R3764/66)
8.0GHz(R3765/67)
設定範囲　20MHz～3.8GHz(R3764/66)
20MHz～8.0GHz(R3765/67)
設定分解能 1Hz

9. [SOURce:]POWer[<chno>][:LEVel][:AMPLitude]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ OUTLEV

● 機能　出力レベルの設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● コマンド　[SOURce:]POWer[<chno>][:LEVel][:AMPLitude] <real>
OUTLEV<real>

● パラメータ　<real>

● 応答形式　NR3 （実数値）

● 説明　周波数掃引時の出力レベルを設定します。

	初期設定	設定範囲	
		SRC COR ON	SRC COR OFF
A タイプ	0dBm	-13dBm～+17dBm	-16dBm～+24.95dBm
B タイプ	0dBm	-15dBm～+15dBm	-13dBm～+22.95dBm
C タイプ (Aﾀｲﾌﾟ+ Sﾊﾟﾗﾒｰﾀ)	10dBm	-20dBm～+15dBm	-23dBm～+17.95dBm

設定分解能 0.01dB

10. [SOURce:]POWer[<chno>]:MODE　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ LEVEL

● 機能　　掃引タイプの設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　[SOURce:]POWer[<chno>]:MODE <mode>

　パラメータ　　<mode> = {SWEep }

　応答形式　　FIX | SWE | PSW

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　ｺﾏﾝﾄﾞ･ﾌｫｰﾏｯﾄ　LEVEL

　応答形式　　0 | 1

● 説明　　このコマンドは下表のように、組みで設定します。
初期設定　リニア周波数掃引

ｺﾏﾝﾄﾞ	PSW:MODE	FREQ:MODE	POW:MODE	SWE:SPAC	掃引タイプ	対応する R3762/63 ｺﾏﾝﾄﾞ
ﾊﾟﾗﾒｰﾀ	(NONE)	SWE	(FIX)	LIN	リニア周波数掃引	LINFREQ
				LOG	ログ周波数掃引	LOGFREQ
		(CW)	SWE	(LIN)	レベル掃引	LEVEL
	FREQ	(PSW)	(FIX)	(LIN)	ﾌﾟﾛｸﾞﾗﾑ掃引(周波数のみ)	USRFSWP
	ALL	(PSW)	(PSW)	(LIN)	プログラム掃引	USRARWP

（注）表中の( ) はクエリ時に返る値であり、その値での設定はできません。

掃引タイプ　：リニア周波数掃引…固定レベルにて周波数を等間隔で掃引
　：ログ周波数掃引　…固定レベルにて周波数を対数間隔で掃引
　：レベル掃引　　…固定周波数にて出力レベルを掃引
　：プログラム掃引　…区間ごとに周波数のみを任意に設定
　（周波数のみ）
　：プログラム掃引　…区間ごとに周波数、出力レベル、分解能帯域幅、セットリング時間を任意に設定

ただし、R3764 、R3766 ではログ周波数掃引の設定はできません。

11. [SOURce:]POWer[<chno>]:STARt　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ STLEVEL

● 機能　スタート・レベルの設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● コマンド　[SOURce:]POWer[<chno>]:STARt <real>
STLEVEL<real>

● パラメータ　<real>

● 応答形式　NR3 （実数値）

● 説明　レベル掃引のスタート・レベルを設定します。

	初期設定		設定範囲	
	ｽﾀｰﾄ	ｽﾄｯﾌﾟ	SRC COR ON	SRC COR OFF
A タイプ	-13dBm	0dBm	-13dBm～+17dBm	-16dBm～+24.95dBm
B タイプ	-15dBm	0dBm	-15dBm～+15dBm	-13dBm～+22.95dBm
C タイプ (Aﾀｲﾌﾟ+ Sﾊﾟﾗﾒｰﾀ)	-20dBm	0dBm	-20dBm～+15dBm	-23dBm～+17.95dBm

設定分解能 0.01dB

12. [SOURce:]POWer[<chno>]:STOP　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ SPLEVEL

● 機能　ストップ・レベルの設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● コマンド　[SOURce:]POWer[<chno>]:STOP <real>
SPLEVEL<real>

● パラメータ　<real>

● 応答形式　NR3 （実数値）

● 説明　レベル掃引のストップ・レベルを設定します。

	初期設定		設定範囲	
	ｽﾀｰﾄ	ｽﾄｯﾌﾟ	SRC COR ON	SRC COR OFF
A タイプ	-13dBm	0dBm	-13dBm～+17dBm	-16dBm～+24.95dBm
B タイプ	-15dBm	0dBm	-15dBm～+15dBm	-13dBm～+22.95dBm
C タイプ (Aﾀｲﾌﾟ+ Sﾊﾟﾗﾒｰﾀ)	-20dBm	0dBm	-20dBm～+15dBm	-23dBm～+17.95dBm

設定分解能 0.01dB

13. [SOURce:]PSWeep[<chno>]:BANDwidth[<n>]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
USEG
URBW

● 機能　　プログラム掃引のセグメント帯域幅の入力

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　[SOURce:]PSWeep[<chno>]:BANDwidth[<n>] <int>

パラメータ　　<int>

応答形式　　NR1(整数値)

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　USEG<int>
URBW<int>

パラメータ　　<int>

応答形式　　NR1(整数値)

● 説明　　プログラム掃引のセグメント帯域幅を設定します。

R3762/63 コマンド	R3764/66、R3765/67 ｺﾏﾝﾄﾞ･ﾊﾟﾗﾒｰﾀ	動作
USEG	<n>	セグメント番号を指定する
URBW	<int>	帯域幅を設定する

● 注意　　この帯域幅の設定は、PSWeep[<chno>]:MODEがALLのとき(USRASWP)のみ反映され、FREQのとき(USRFSWP) には反映されません。

14. [SOURce:]PSWeep[<chno>]:CLEar[<n>]

● 機能　　プログラム掃引の指定セグメントのクリア

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　[SOURce:]PSWeep[<chno>]:CLEar[<n>]

● 説明　　プログラム掃引の<n> 番のセグメントの設定条件をクリアします。

15. [SOURce:]PSWeep[<chno>]:CLEar[<n>]:ALL　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ USEGCL

- ●機能　　プログラム掃引の全セグメントのクリア
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　　Command
- ●コマンド　　[SOURce:]PSWeep[<chno>]:CLEar[<n>]:ALL
  USEGCL
- ●説明　　プログラム掃引の全セグメントの設定条件をクリアします。

16. [SOURce:]PSWeep[<chno>]:FREQuency[<n>]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
USEG
UFREQ
U{START|STOP}

● 機能　　プログラム掃引のセグメント周波数の入力

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　[SOURce:]PSWeep[<chno>]:FREQuency[<n>] <start>[,<stop>]

パラメータ　　<start>
<stop>

応答形式　　<start>,<stop>
<start>=<stop>=NR3（実数値）

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　USEG<int>
UFREQ<real>
U{START|STOP}<real>

応答形式　　NR1(USEGコマンド)
NR3(UFREQ|USTART|USTOPコマンド)

● 説明　　プログラム掃引のセグメント周波数を設定します。

R3762/63 コマンド	R3764/66、R3765/67 ｺﾏﾝﾄﾞ･ﾊﾟﾗﾒｰﾀ	動作
USEG	<n>	セグメント番号を指定する
UFREQ	* 1	固定周波数を設定する
USTART	<start>	スタート周波数を設定する
USTOP	<stop>	ストップ周波数を設定する

*1 : <stop>を省略した場合の<start> に相当します。
<stop>が省略されると<stop>=<start>で、さらにセグメントポイント数(PSWeep[<chno>]:POINts[<n>])に1 が自動的に設定されます。

17. [SOURce:]PSWeep[<chno>]:MODE　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
USR{FSWP|ASWP}

● 機能　　掃引タイプの設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　[SOURce:]PSWeep[<chno>]:MODE <mode>

パラメータ　　<mode>= {FREQuency|ALL}

応答形式　　NONE|FREQ|ALL

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　USR{FSWP|ASWP}

応答形式　　0 | 1

● 説明　　このコマンドは下表のように、組みで設定します。
初期設定　リニア周波数掃引

PSW:MODEをFREQまたはALL に設定すると、入力済みのセグメントを探し出し、周波数の低い順にセグメントを内部的に並べ換えて実行します。
このとき、並べ換えられたセグメント間で、そのセグメントのSTOP周波数が次のセグメントのSTART 周波数よりも大きい場合は、エラーとなります。

ｺﾏﾝﾄﾞ	PSW:MODE	FREQ:MODE	POW:MODE	SWE:SPAC	掃引タイプ	対応する R3762/63ｺﾏﾝﾄﾞ
ﾊﾟﾗﾒｰﾀ	(NONE)	SWE	(FIX)	LIN	リニア周波数掃引	LINFREQ
				LOG	ログ周波数掃引	LOGFREQ
		(CW)	SWE	(LIN)	レベル掃引	LEVEL
	FREQ	(PSW)	(FIX)	(LIN)	ﾌﾟﾛｸﾞﾗﾑ掃引(周波数のみ)	USRFSWP
	ALL	(PSW)	(PSW)	(LIN)	プログラム掃引	USRARWP

（注）表中の( ) はクエリ時に返る値であり、その値での設定はできません。

掃引タイプ　：リニア周波数掃引…固定レベルにて周波数を等間隔で掃引
：ログ周波数掃引　…固定レベルにて周波数を対数間隔で掃引
：レベル掃引　…固定周波数にて出力レベルを掃引
：プログラム掃引　…区間ごとに周波数のみを任意に設定
（周波数のみ）
：プログラム掃引　…区間ごとに周波数、出力レベル、分解能帯域幅、セットリング時間を任意に設定

ただし、R3764 、R3766 ではログ周波数掃引の設定はできません。

18. [SOURce:]PSWeep[<chno>]:POINts[<n>]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
USEG
UPOINT

●機能　　プログラム掃引のセグメントポイント数の入力

●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

●IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　[SOURce:]PSWeep[<chno>]:POINts[<n>] <int>

パラメータ　　<int>

応答形式　　NR1 （整数値）

●IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　USEG<int>
UPOINT<int>

パラメータ　　<int>

応答形式　　NR1 （整数値）

●説明　　プログラム掃引のセグメントポイント数を設定します。

R3762/63 コマンド	R3764/66、R3765/67 ｺﾏﾝﾄﾞ･ﾊﾟﾗﾒｰﾀ	動作
USEG	<n>	セグメント番号を指定する
UPOINT	<int>	ポイント数を設定する

19. [SOURce:]PSWeep[<chno>]:POWer[<n>]

IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
USEG
ULEVEL

● 機能　プログラム掃引のセグメント出力レベルの入力

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

コマンド　[SOURce:]PSWeep[<chno>]:POWer[<n>] <real>

パラメータ　<real>

応答形式　NR3 （実数値）

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

コマンド　USEG<int>
ULEVEL<real>

パラメータ　<int>
<real>

応答形式　NR1 （USEGコマンド）
NR3 （ULEVELコマンド）

● 説明　プログラム掃引のセグメント出力レベルを設定します。

R3762/63 コマンド	R3764/66、R3765/67 ｺﾏﾝﾄﾞ･ﾊﾟﾗﾒｰﾀ	動作
USEG	<n>	セグメント番号を指定する
ULEVEL	<real>	出力レベルを設定する

● 注意　この出力レベルの設定値は、PSWeep[<chno>]:MODE がALL のとき(USRASWP) のみ反映され、FREQのとき(USRFSWP) には反映されません。

20. [SOURce:]PSWeep[<chno>]:SETTling[<n>]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
USEG
USETLT

● 機能　　プログラム掃引のセグメントセットリング時間の入力

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　[SOURce:]PSWeep[<chno>]:SETTling[<n>] <real>

パラメータ　　<real>

応答形式　　NR3 （実数値）

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　USEG<int>
USETLT<real>

パラメータ　　<int>
<real>

応答形式　　NR1 （USEGコマンド）
NR3 （USETLTコマンド）

● 説明　　プログラム掃引のセグメントセットリング時間を設定します。

R3762/63 コマンド	R3764/66、R3765/67 ｺﾏﾝﾄﾞ･ﾊﾟﾗﾒｰﾀ	動作
USEG	<n>	セグメント番号を指定する
USETLT	<real>	セットリング時間を設定する

● 注意　　このセットリング時間の設定値は、PSWeep[<CHNO>]:MODE がALL のとき(USRASWP) のみ反映され、FREQのとき(USRFSWP) には反映されません。

21. [SOURce:]SWEep[<chno>]:POINts　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
POIN
M{1201|601|301|201|101|51|21|11|6|3}P

● 機能　　掃引ポイント数の設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　[SOURce:]SWEep[<chno>]:POINts <int>

パラメータ　　<int>

応答形式　　NR1 （整数値）

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　POIN<int>
M{1201|601|301|201|101|51|21|11|6|3}P

パラメータ　　<int>

クエリ　　POIN?
M{1201|601|301|201|101|51|21|11|6|3}P?

応答形式　　NR1 （POIN? コマンド）
0 | 1(M{1201|601|301|201|101|51|21|11|6|3}P?コマンド)

● 説明　　掃引ポイント数を設定します。

設定可能なポイント数は、3, 6, 11, 21, 51, 101, 201, 301, 401, 601, 801, 1201 ポイントです。

22. [SOURce:]SWEep[<chno>]:SPACing　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
LINFREQ
LOGFREQ

● 機能　　掃引タイプの指定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　[SOURce:]SWEep[<chno>]:SPACing <mode>

パラメータ　　<mode> = {LINear | LOGarithmic}

応答形式　　LIN | LOG

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　LINFREQ
LOGFREQ

応答形式　　0 | 1

● 説明　　このコマンドは下表のように、組みで設定します。
初期設定　リニア周波数掃引

ｺﾏﾝﾄﾞ	PSW:MODE	FREQ:MODE	POW:MODE	SWE:SPAC	掃引タイプ	対応する R3762/63ｺﾏﾝﾄﾞ
ﾊﾟﾗﾒｰﾀ	(NONE)	SWE	(FIX)	LIN	リニア周波数掃引	LINFREQ
				LOG	ログ周波数掃引	LOGFREQ
		(CW)	SWE	(LIN)	レベル掃引	LEVEL
	FREQ	(PSW)	(FIX)	(LIN)	ﾌﾟﾛｸﾞﾗﾑ掃引(周波数のみ)	USRFSWP
	ALL	(PSW)	(PSW)	(LIN)	プログラム掃引	USRARWP

(注) 表中の( ) はクエリ時に返る値であり、その値での設定はできません。

掃引タイプ　：リニア周波数掃引…固定レベルにて周波数を等間隔で掃引
：ログ周波数掃引　…固定レベルにて周波数を対数間隔で掃引
：レベル掃引　　…固定周波数にて出力レベルを掃引
：プログラム掃引　…区間ごとに周波数のみを任意に設定
(周波数のみ)
：プログラム掃引　…区間ごとに周波数、出力レベル、分解能帯域幅、セットリング時間を任意に設定

ただし、R3764 、R3766 ではログ周波数掃引の設定はできません。

23. [SOURce:]SWEep[<chno>]:TIME　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ STIME

● 機能　　掃引時間の設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

コマンド　　[SOURce:]SWEep[<chno>]:TIME <real>
STIME<real>

パラメータ　　<real>

応答形式　　NR3 （実数値）

● 説明　　掃引時間を設定します。0 に設定した場合はAUTOとなります。

初期設定　30ms
設定範囲　0.2ms ～3932.1s
設定分解能　0.05ms

24. [SOURce:]SWEep[<chno>]:TIME:AUTO　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ STIMEAUTO

● 機能　　掃引時間の自動設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

コマンド　　[SOURce:]SWEep[<chno>]:TIME:AUTO <bool>
STIMEAUTO

パラメータ　　<bool>

応答形式　　0 | 1

● 説明　　掃引時間を分解能帯域幅で決定される最小掃引時間に自動設定します。
AUTOモードで掃引時間を設定すると、AUTOモードが解除されます。

7.12　STATus サブシステム

1. STATus:DEVice:CONDition?

●機能　DEV ステータスの参照

●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Query

●クエリ　STATus:DEVice:CONDition?

●応答形式　NR1 （整数値）

●説明　デバイス・ステータス・レジスタのコンディション・レジスタの内容を応答します。このレジスタは読み出してもクリアされません。

詳細は[4. ステータス・バイト] を参照して下さい。

コンディション・レジスタの割り当て

bit		説明
15		・常に0
14	Program running	・内蔵BASIC プログラムが実行されていると 1にセットされる
13～4		・常に0
3	Sweeping	・掃引実行中に 1にセットされる
2～1		・常に0
0	Cooling Fan Stopped	・冷却用ファンがとまっていると、1 にセットされる
その他		・常に0

2. STATus:DEVice:ENABle

● 機能　OPERステータスのイネーブル・レジスタの設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● コマンド　STATus:DEVice:ENABle <int>

● パラメータ　<int>

● 応答形式　NR1 （整数値）

● 説明　デバイス・ステータス・レジスタのイネーブル・レジスタの内容を設定します。このレジスタの1 に設定されたbit に対応するイベント・レジスタが、有効ビットとしてステータス・バイト・レジスタの2 に反映されます。

詳細は[4. ステータス・バイト] を参照して下さい。

● 例　Cooling Fan Stopped(ビット1)をイネーブルにセットする場合、STAT:DEV:ENAB 1 とセットします。

3. STATus:DEVice[:EVENt]?

● 機能　OPERステータスの問い合わせ(with clear)

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Query

● クエリ　STATus:DEVice[:EVENt]?

● 応答形式　NR1 （整数値）

● 説明　デバイス・ステータス・レジスタのイベント・レジスタの内容を応答します。このレジスタは読み出されるとクリアされ、対応するステータス・バイト・レジスタのビット2 もクリアされます。

詳細は[4. ステータス・バイト] を参照して下さい。

イベント・レジスタの割り当て

bit		説明
15		・常に0
14	Program running	・内蔵BASIC プログラムの実行が停止すると 1にセットされる
13～9		・常に0
8	Averaging	・アベレージ終了時に1 にセットされる
7～4		・常に0
3	Sweeping	・掃引終了時に1 にセットされる
2～1		・常に0
0	Cooling Fan Stopped	・冷却用ファンが停止した場合、1 にセットされる
その他		・常に0

4. STATus:FREQuency:CONDition?

● 機能　FREQステータスの参照

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Query

● クエリ　STATus:FREQuency:CONDition?

● 応答形式　NR1 （整数値）

● 説明　周波数ステータス・レジスタのコンディション・レジスタの内容を応答します。このレジスタは読み出してもクリアされません。

詳細は[4. ステータス・バイト] を参照して下さい。

コンディション・レジスタの割り当て

bit		説明
0	Local 1 Unlocked	・ローカル1 のロックが外れていると1 にセットされる
1	Local 2 Unlocked	・ローカル2 のロックが外れていると1 にセットされる
2	Synthe Unlocked	・シンセのロックが外れていると1 にセットされる
3	External Standard In	・外部基準周波数が入力されていると1 にセットされる
4	VCXO Unlocked	・VCXOのロックが外れていると1 にセットされる
その他		・常に0

5. STATus:FREQuency:ENABle?

- 機能　FREQステータスのイネーブル・レジスタの設定
- ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query
- コマンド　STATus:FREQuency:ENABle <int>
- パラメータ　<int>
- 応答形式　NR1 （整数値）
- 説明　周波数ステータス・レジスタのイネーブル・レジスタの内容を設定します。このレジスタの1 に設定されたビットに対応するイベント・レジスタが、有効ビットとしてクエスチョナブル・ステータス・レジスタのビット5 に反映されます。

  詳細は[4. ステータス・バイト] を参照して下さい。
- 例　External Standard In （ビット3)をイネーブルに設定する場合、2**3 = 8を計算し、STAT:FREQ:ENAB 8とセットします。

6. STATus:FREQuency[:EVENt]?

● 機能　FREQステータスの読み出し

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Query

● クエリ　STATus:FREQuency[:EVENt]?

● 応答形式　NR1 （整数値）

● 説明　周波数ステータス・レジスタのイベント・レジスタの内容を応答します。このレジスタは読み出されるとクリアされ、対応するクエスチョナブル・ステータス・レジスタのビット5 もクリアされます。

詳細は[4. ステータス・バイト] を参照して下さい。

イベント・レジスタの割り当て

bit		説明
0	Local 1 Unlocked	・ローカル1 のロックが外れたときに1 にセットされる
1	Local 2 Unlocked	・ローカル2 のロックが外れたときに1 にセットされる
2	Synthe Unlocked	・シンセのロックが外れたときに1 にセットされる
3	External Standard In	・外部基準周波数が入力されたときに1 にセットされる
4	VCXO Unlocked	・VCXOのロックが外れたときに1 にセットされる
その他		・常に0

7. STATus:LIMit:CONDition?

● 機能　LIM ステータスの参照

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Query

● クエリ　STATus:LIMit:CONDition?

● 応答形式　NR1 （整数値）

● 説明　リミット・ステータス・レジスタのコンディション・レジスタの内容を応答します。このレジスタは読み出してもクリアされません。

詳細は[4. ステータス・バイト] を参照して下さい。

コンディション・レジスタの割り当て

bit		説明
0	CH1 1st Limit Failed	・ﾁｬﾝﾈﾙ1の第1 波形がFAILになっていると1 にセットされる
1	CH1 2nd Limit Failed	・ﾁｬﾝﾈﾙ1の第2 波形がFAILになっていると1 にセットされる
2	CH2 1st Limit Failed	・ﾁｬﾝﾈﾙ2の第1 波形がFAILになっていると1 にセットされる
3	CH2 2nd Limit Failed	・ﾁｬﾝﾈﾙ2の第2 波形がFAILになっていると1 にセットされる
4	CH3 1st Limit Failed	・ﾁｬﾝﾈﾙ3の第1 波形がFAILになっていると1 にセットされる
5	CH3 1st Limit Failed	・ﾁｬﾝﾈﾙ3の第2 波形がFAILになっていると1 にセットされる
6	CH4 2nd Limit Failed	・ﾁｬﾝﾈﾙ4の第1 波形がFAILになっていると1 にセットされる
7	CH4 1st Limit Failed	・ﾁｬﾝﾈﾙ4の第2 波形がFAILになっていると1 にセットされる
その他		・常に0

8. STATus:LIMit:EVABle

● 機能　LIM ステータスのイネーブル・レジスタの設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● コマンド　STATus:LIMit:EVABle <int>

● パラメータ　<int>

● 応答形式　NR1 （整数値）

● 説明　リミット・ステータス・レジスタのイネーブル・レジスタの内容を設定します。このレジスタの1 に設定されたビットに対応するイベント・レジスタが、有効ビットとしてクエスチョナブル・ステータス・レジスタのビット9 に反映されます。

詳細は[4. ステータス・バイト] を参照して下さい。

● 例　CH1 1st Limit Failed (ビット0)とCH3 1st Limit Failed (ビット4)をイネーブルに設定する場合、2**0 + 2**4 = 17を計算し、STAT:LIN:ENAB 17とセットします。

9. STATus:LIMit[:EVENt]?

● 機能　　LIM ステータスの読み出し

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Query

● クエリ　　STATus:LIMit[:EVENt]?

● 応答形式　　NR1 （整数値）

● 説明　　リミット・ステータス・レジスタのイベント・レジスタの内容を応答します。このレジスタは読み出されるとクリアされ、対応するクエスチョナブル・ステータス・レジスタのビット9 もクリアされます。

詳細は[4. ステータス・バイト] を参照して下さい。

イベント・レジスタの割り当て

bit		説明
0	CH1 1st Limit Failed	・ﾁｬﾝﾈﾙ1の第1 波形がFAILになっていると1 にセットされる
1	CH1 2nd Limit Failed	・ﾁｬﾝﾈﾙ1の第2 波形がFAILになっていると1 にセットされる
2	CH2 1st Limit Failed	・ﾁｬﾝﾈﾙ2の第1 波形がFAILになっていると1 にセットされる
3	CH2 2nd Limit Failed	・ﾁｬﾝﾈﾙ2の第2 波形がFAILになっていると1 にセットされる
4	CH3 1st Limit Failed	・ﾁｬﾝﾈﾙ3の第1 波形がFAILになっていると1 にセットされる
5	CH3 1st Limit Failed	・ﾁｬﾝﾈﾙ3の第2 波形がFAILになっていると1 にセットされる
6	CH4 2nd Limit Failed	・ﾁｬﾝﾈﾙ4の第1 波形がFAILになっていると1 にセットされる
7	CH4 1st Limit Failed	・ﾁｬﾝﾈﾙ4の第2 波形がFAILになっていると1 にセットされる
その他		・常に0

10. STATus:OPERation:CONDition?

● 機能　OPERステータスの参照

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Query

● クエリ　STATus:OPERation:CONDition?

● 応答形式　NR1 （整数値）

● 説明　オペレーション・ステータス・レジスタのコンディション・レジスタの内容を応答します。このレジスタは読み出してもクリアされません。

詳細は[4. ステータス・バイト] を参照して下さい。

コンディション・レジスタの割り当て

bit		説明
0	Calibrating	・補正実行中に1 にセットされる
3	Sweeping	・掃引実行中に1 にセットされる
14	Program Running	・内蔵BASIC でプログラムが実行されていると1 にセットされる
その他		・常に0

11. STATus:OPERation:ENABle

● 機能　OPERステータスのイネーブル・レジスタの設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● コマンド　STATus:OPERation:ENABle <int>

● パラメータ　<int>

● 応答形式　NR1 （整数値）

● 説明　オペレーション・ステータス・レジスタのイネーブル・レジスタの内容を設定します。このレジスタの1 に設定されたビットに対応するイベント・レジスタが、有効ビットとしてステータス・バイト・レジスタのビット7 に反映されます。

詳細は[4. ステータス・バイト] を参照して下さい。

● 例　Program Running(ビット14) とSweeping （ビット3)をイネーブルに設定する場合、2**14 + 2**3 = 16392を計算し、
STAT:OPER:ENAB 16392とセットします。

12. STATus:OPERation[:EVENt]?

● 機能　OPERステータスの読み出し

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Query

● クエリ　STATus:OPERation[:EVENt]?

● 応答形式　NR1 （整数値）

● 説明　オペレーション・ステータス・レジスタのイベント・レジスタの内容を応答します。このレジスタは読み出されるとクリアされ、対応するステータス・バイト・レジスタのビット7 もクリアされます。

詳細は[4. ステータス・バイト] を参照して下さい。

イベント・レジスタの割り当て

bit		説明
0	Calibrating	・補正終了時に1 にセットされる
3	Sweeping	・掃引終了時に1 にセットされる
14	Program Running	・内蔵BASIC でプログラムが停止すると1 にセットされる
その他		・常に0

13. STATus:POWer:CONDition?

● 機能　POW ステータスの参照

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Query

● クエリ　STATus:POWer:CONDition?

● 応答形式　NR1 （整数値）

● 説明　パワー・ステータス・レジスタのコンディション・レジスタの内容を応答します。このレジスタは読み出してもクリアされません。

詳細は[4. ステータス・バイト] を参照して下さい。

コンディション・レジスタの割り当て

bit		説明
0	Input-R Overloaded	・入力R に過入力レベルが入っていると1 にセットされる
1	Input-R Tripped	・入力R の保護回路が動作していると1 にセットされる
2	Input-A Overloaded	・入力A に過入力レベルが入っていると1 にセットされる
3	Input-A Tripped	・入力A の保護回路が動作していると1 にセットされる
4	Input-B Overloaded	・入力B に過入力レベルが入っていると1 にセットされる
5	Input-B Tripped	・入力B の保護回路が動作していると1 にセットされる
その他		・常に0

14. STATus:POWer:ENABle

- ●機能　POW ステータスのイネーブル・レジスタの設定
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query
- ●コマンド　STATus:POWer:ENABle <int>
- ●パラメータ　<int>
- ●応答形式　NR1 （整数値）
- ●説明　パワー・ステータス・レジスタのイネーブル・レジスタの内容を設定します。このレジスタの1 に設定されたビットに対応するイベント・レジスタが、有効ビットとしてクエスチョナブル・ステータス・レジスタのビット3 に反映されます。

  詳細は[4. ステータス・バイト] を参照して下さい。
- ●例　Input-A Overloaded (ビット2)をイネーブルに設定する場合、2**2 = 4を計算し、STAT:POW:ENAB 4 とセットします。

15. STATus:POWer[:EVENt]?

● 機能　POW ステータスの読み出し

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Query

● クエリ　STATus:POWer[:EVENt]?

● 応答形式　NR1 （整数値）

● 説明　パワー・ステータス・レジスタのイベント・レジスタの内容を応答します。このレジスタは読み出されるとクリアされ、対応するクエスチョナブル・ステータス・レジスタのビット3 もクリアされます。

詳細は[4. ステータス・バイト] を参照して下さい。

イベント・レジスタの割り当て

bit		説明
0	Input-R Overloaded	・入力R に過入力レベルが入ると1 にセットされる
1	Input-R Tripped	・入力R の保護回路が動作すると1 にセットされる
2	Input-A Overloaded	・入力A に過入力レベルが入ると1 にセットされる
3	Input-A Tripped	・入力A の保護回路が動作すると1 にセットされる
4	Input-B Overloaded	・入力B に過入力レベルが入ると1 にセットされる
5	Input-B Tripped	・入力B の保護回路が動作すると1 にセットされる
その他		・常に0

16. STATus:QUEStionable:ENABle

● 機能　QUESステータスのイネーブル・レジスタの設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● コマンド　STATus:QUEStionable:ENABle <int>

● パラメータ　<int>

● 応答形式　NR1 （整数値）

● 説明　クエスチョナブル・ステータス・レジスタのイネーブル・レジスタの内容を設定します。このレジスタの1 に設定されたビットに対応するイベント・レジスタが、有効ビットとしてステータス・バイト・レジスタのビット3 に反映されます。

詳細は[4. ステータス・バイト] を参照して下さい。

● 例　POW(ビット3)とLIM(ビット9)のサマリ・ビットをイネーブルに設定する場合、2**3 + 2**9 = 520 を計算し、STAT:QUES:ENAB 520 とセットします。

17. STATus:QUEStionable[:EVENt]?

● 機能　QUESステータスの読み出し

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Query

● クエリ　STATus:QUEStionable[:EVENt]?

● 応答形式　NR1 （整数値）

● 説明　クエスチョナブル・ステータス・レジスタのイベント・レジスタの内容を応答します。このレジスタは読み出されるとクリアされ、対応するステータス・バイト・レジスタもクリアされます。

詳細は[4. ステータス・バイト] を参照して下さい。

イベント・レジスタの割り当て

bit		説明
3	POW Summary Bit	・パワー・ステータス・レジスタのサマリが1 になったとき、1 にセットされる
5	FREQ Summary Bit	・周波数ステータス・レジスタのサマリが1 になったとき、1 にセットされる
9	LIM Summary Bit	・リミット・ステータス・レジスタのサマリが1 になったとき、1 にセットされる
その他		・常に0

7.13　SYSTem サブシステム

1. SYSTem:DATE　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
YEAR
MONTH
DAY

● 機能　　日付の設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　SYSTem:DATE <year>,<month>,<day>

　パラメータ　　<year> = 1900 - 2099の数値データ
　　<month> = 1 - 12の数値データ
　　<day> = 1 - 31の数値データ

　応答形式　　<year>,<month>,<day>
　　<year> = <month> = <day> = NR1（整数値）

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　YEAR<int>
　　MONTH<int>
　　DAY<int>

　パラメータ　　<int>

　応答形式　　NR1(整数値)

● 説明　　本器に内蔵の時計の日付を設定します。

　　年の設定は西暦(4桁）で表して下さい（例えば1990、1993）。

2. SYSTem:ERRor?

● 機能　　エラーの問い合わせ

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　　Query

● クエリ　　SYSTem:ERRor?

● 応答形式　　<errno>,<errmsg>
<errno> = NR1 （整数値）
<errmsg> =エラーメッセージ

● 説明　　本器は最大10個まで発生したエラーをエラーキューに記憶します。10個以上のエラーが発生したときは、10個目のエラーが以下のように置き変わります。

-350,"Queue overflow"

この10個目以降のエラーは保持されません。
SYSTem:ERRor? は、このエラー・キューからエラーを取り出します。
エラー・キューは、FIFO (First in First Out) 方式でエラーをストアするのでエラーは発生した順序通りに取り出されます。
エラー・キューからエラーが取り出されると、エラーはエラー・キューから消え、次に発生するエラーを記憶できるようになります。

エラーがないときには、以下のように応答します。

0,"No error"

エラー・キューは*CLSでクリアされます。

3. SYSTem:PRESet

IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
IP

● 機能　システムの初期化

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command

● コマンド
SYSTem:PRESet
IP

● 説明
SYSTem:PRESet (IP)コマンドは、本器の設定を初期化し、トリガ・システムのリセットを実行します。
このコマンドで設定される初期値は、*RSTで設定される初期値とは違います。
実際に設定される値は[A3. 初期設定] を参照して下さい。
このコマンドが実行する内容は、正面パネルのPRESETキーと同じです。

4. SYSTem:TIME

IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
HOUR
MINUTE
RTC30ADJ

● 機能　時刻の設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　SYSTem:TIME <hour>,<minute>,<second>

パラメータ
<hour> = 0-23 の数値データ
<minute> = 0-59 の数値データ
<second> = 0-59 の数値データ

応答形式
<hour>,<minute>,<second>
<hour> = <minute> = <second> = NR1 (整数値)

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド
HOUR<int>
MINUTE<int>
RTC30ADJ

パラメータ　<int>

応答形式
NR1(整数値)
RTC30ADJコマンドはクエリが存在しません。

● 説明
本器に内蔵の時計の時刻を設定します。時間は24時間制です。
R3751 コマンド・モードのRTC30ADJは、秒設定を必ず 0にするコマンドです。

7.14　TRACeサブシステム

1. TRACe[<chno>]:COPY　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
DTOM

● 機能　トレースのコピー

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command

● コマンド　TRACe[<chno>]:COPY <name>
DTOM

● パラメータ　<name> = DATA

● 説明　このコマンドは、データ波形をメモリ波形にコピーします。

2. TRACe[<chno>][:DATA]?　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
OT{1|2|3|4}{DRAT|CORED|MRAT|NORED|DFOR|
MFOR|CORNR|CORDI|CORSO|CORTR}

● 機能　トレースのクエリ（出力）

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
クエリ　TRACe[<chno>][:DATA]? {<name>|<trace>}[,{<name>|<trace>}...]

パラメータ　<name> = {RAW|DATA|MEM|UDAT|FDAT1|FDAT2|FMEM1|FMEM2|
NORM|EDIR|ESM|ERTR|EDF|ESF|ERF|
ELF|ETF|EXF|EDR|ESR|ERR|ELR|ETR|EXR}

<trace>=　解析チャンネル

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
クエリ　OT{1|2|3|4}{DRAT|CORED|MRAT|NORED|DFOR|MFOR|CORNR|CORDI|
|CORSO|CORTR}

● 説明　指定したトレースデータを出力します。
<name>または<trace> はカンマ(,) で区切って複数指定できます。
その場合、指定した順にトレース単位でデータが出力されます。
(1トレース分の出力が終了してから次のトレースに進む)

3. TRACe[<chno>][:DATA]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
IN{1|2|3|4}{DRAT|CORED|MRAT|NORED|DFOR|MFOR|CORNR|CORDI|CORSO|CORTR}

● 機能　　トレースの入力

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

コマンド　　TRACe[<chno>][:DATA] {<name>|<trace>},{<block>|<real>[,<real>...]}

パラメータ　　<name> = {RAW|DATA|MEM|UDAT|FDAT1|FDAT2|FMEM1|FMEM2|NORM|EDIR|ESM|ERTR}
EDF|ESF|ERF|ELF|ETF|EXF|EDR|ESR|ERR|ELR|ETR|EXR}

<trace> = 解析チャンネル

● IEEE481.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

コマンド　　IN{1|2|3|4}{DRAT|CORED|MRAT|NORED|DFOR|MFOR|CORNR|CORDI|CORSO|CORTR}

● 説明　　指定したトレースにデータを入力します。
トレース出力と違い、<name>または<trace> の複数指定はできません。

※　トレース入出力コマンドのパラメータについて

R3762/63コマンド	R3764/66、R3765/67ｺﾏﾝﾄﾞ･ﾊﾟﾗﾒｰﾀ		対象となるトレース	データ形式*2
	<name> *1	<trace>		
{OT|IN}{1|2|3|4}DRAT	RAW	{131|195|259|323}	生データ配列	複素数
{OT|IN}{1|2|3|4}CORED	DATA	{129|193|257|321}	データ配列	複素数
{OT|IN}{1|2|3|4}MRAT	MEMory	{130|194|258|322}	メモリ配列	複素数
{OT|IN}{1|2|3|4}NORED	UDATa	{128|192|256|320}	フォーマット前のデータ配列	複素数
{OT|IN}{1|2|3|4}DFOR	FDATa1	{0|1|4|5}	フォーマット後のデータ配列1	第 1波形
	FDATa2	{8|9|12|13}	フォーマット後のデータ配列2	第 2波形
{OT|IN}{1|2|3|4}MFOR	FMEMory1	{2|3|6|7}	フォーマット後のメモリ配列1	第 1波形
	FMEMory2	{10|11|14|15}	フォーマット後のメモリ配列2	第 2波形
{OT|IN}{1|2|3|4}CORNR	NORMalize	{133|197|261|325}	ノーマライズ基準データ配列	複素数
{OT|IN}{1|2|3|4}CORDI	EDIRectivity	{134|198|262|326}	方向性エラー係数配列	複素数
{OT|IN}{1|2|3|4}CORSO	ESMatch	{135|199|263|327}	ｿｰｽ･ﾏｯﾁ･ｴﾗｰ 係数配列	複素数
{OT|IN}{1|2|3|4}CORTR	ERTRacking	{136|200|264|328}	反射ﾄﾗｯｷﾝｸﾞ･ｴﾗｰ係数配列	複素数
	EDForward	{137|201|265|329}	順方向：方向性ｴﾗｰ 係数配列	複素数
	ESForward	{138|202|266|330}	順方向：ｿｰｽ･ﾏｯﾁ･ｴﾗｰ 係数配列	複素数
	ERForward	{139|203|267|331}	順方向：反射ﾄﾗｯｷﾝｸﾞ･ｴﾗｰ係数配列	複素数
	ELForward	{140|204|268|332}	順方向：ﾛｰﾄﾞ･ﾏｯﾁ･ｴﾗｰ 係数配列	複素数
	ETForward	{141|205|269|333}	順方向：伝送ﾄﾗｯｷﾝｸﾞ･ｴﾗｰ係数配列	複素数
	EXForward	{142|206|270|334}	順方向：ｱｲｿﾚｰｼｮﾝ･ｴﾗｰ係数配列	複素数
	EDReverse	{143|207|271|335}	逆方向：方向性ｴﾗｰ 係数配列	複素数
	ESReverse	{144|208|272|336}	逆方向：ｿｰｽ･ﾏｯﾁ･ｴﾗｰ 係数配列	複素数
	ERReverse	{145|209|273|337}	逆方向：反射ﾄﾗｯｷﾝｸﾞ･ｴﾗｰ係数配列	複素数
	ELReverse	{146|210|274|338}	逆方向：ﾛｰﾄﾞ･ﾏｯﾁ･ｴﾗｰ 係数配列	複素数
	ETReverse	{147|211|275|339}	逆方向：伝送ﾄﾗｯｷﾝｸﾞ･ｴﾗｰ係数配列	複素数
	EXReverse	{148|212|276|340}	逆方向：ｱｲｿﾚｰｼｮﾝ･ｴﾗｰ係数配列	複素数

*1: R3764/66, R3765/67コマンドにおいて<name>で指定する場合は、パラメータ<chno>にてチャンネル指定します。

*2: データ形式は、トレースによって変わります。（下記参照）

複素数　：real, imag, real, imag, …の順に出力されます。したがって、総出力データ数は2 倍になります。

第 1波形: フォーマットが LOGMAG&PHASE または LOGMAG&DELAY のときはLOGMAG、LINMAG&PHASEのときはLINMAG、SMITH またはPOLAR のときは実数部(real)、メジャー・モードが S11&S21のときはS11 、S22&S12 のときはS22となります。

第 2波形: フォーマットが LOGMAG&PHASE または LINMAG&PHASE のときは PHASE、LOGMAG&DELAYのときは DELAY、SMITH またはPOLAR のときは虚数部(imag)、メジャー・モードが S11&S21のときはS21 、S22&S12 のときはS12 となります。
その他の場合は、有効なデータとなりません。

7.15 TRIGgerサブシステム

1. TRIGger[:SEQuence]:DELay　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ SETLTIME

- ●機能　トリガ・ディレイの設定
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query
- ●コマンド　TRIGger[:SEQuence]:DELay <real>
  SETLTIME<real>
- ●パラメータ　<real>
- ●応答形式　NR3 （実数値）
- ●説明　このコマンドはトリガが検出されてから実際に測定が開始されるまでの遅延時間を設定します。
  この遅延時間はTRIGger[:SEQuence]:DELay:STATEがONになっているときのみ有効となります。

  TRIGger[:SEQuence]:DELay:STATEも参照して下さい。
- ●注意　0 を設定すると、TRIG:DEL:STAT が自動的にOFF となります。
  0 以外を設定すると、TRIG:DEL:STAT が自動的にONとなります。

2. TRIGger[:SEQuence]:DELay:STATe　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ SETLVARI

- ●機能　トリガ・ディレイのON/OFF
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query
- ●コマンド　TRIGger[:SEQuence]:DELay:STATe <bool>
  SETLVARI<bool>
- ●パラメータ　<bool>
- ●応答形式　0 | 1
- ●説明　このコマンドは、TRIGger[:SEQuence]:DELay (SETLTIME) コマンドで設定したトリガ・ディレイ時間の有効／無効を設定します。
  このコマンドでOFF に設定するとTRIGger[:SEQuence]:DELay 0 (SETLTIME 0)と同じになります。
- ●注意　OFF を設定すると、TRIG:DELが自動的に0 となります。

3. TRIGger[:SEQuence][:IMMediate]

● 機能　イベント・ディテクションのパス(not delay)

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command

● コマンド　TRIGger[:SEQuence][:IMMediate]

● 説明　このコマンドは、トリガ待ちステートをバイパスします。
トリガ・システムがトリガ待ちステートになっていれば、このコマンドは直ちに測定を開始します。
このとき、TRIGger[:SEQuence]:DELay(SETLTIME)コマンドで設定した遅延時間は無効となります。

詳細は[5. トリガ・システム] を参照して下さい。

4. TRIGger[:SEQuence]:SIGNal

● 機能　イベント・ディテクションのパス(with delay)

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command

● コマンド　TRIGger[:SEQuence]:SIGNal

● 説明　このコマンドは、トリガ待ちステートのイベント・ディテクションをバイパスします。トリガ・システムがトリガ待ちステートになっていれば、このコマンドは、TRIGger[:SEQuence]:DELay (SETLTIME)コマンドで設定した遅延時間を待った後に測定を開始します。

詳細は[5. トリガ・システム] を参照して下さい。

5. TRIGger[:SEQuence]:SOURce

IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
FREE
EXTERN

● 機能　トリガ・ソースの設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

コマンド　TRIGger[:SEQuence]:SOURce <source>

パラメータ　<source> = {IMMediate | EXTernal | BUS | HOLD}

応答形式　IMM | EXT | BUS | HOLD

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

コマンド　FREE
EXTERN

応答形式　0 | 1

● 説明

このコマンドはトリガのソースを選択します。それぞれ以下の条件がそろったときにイベント・ディテクションを終了します。

IMMediate : イベントを待ちません。
すぐにトリガ待ちステートのイベント・ディテクションを終了します。
EXTernal : 外部同期信号を待ちます。
BUS : *TRGまたはGET インタフェース・メッセージを待ちます。
HOLD : トリガ待ちステートのイベント・ディテクションを終了しません。

本器がトリガ待ちステートで、TRIGger[:IMMediate] または TRIGger:SIGNalを受け取ると、トリガ・ソースの設定に関わらず測定を開始します。

詳細は[5. トリガ・システム] を参照して下さい。

IEEE488.1-1987コマンド・モードのFREE, EXTERNは、それぞれIEEE488.2-1987コマンド・モードのIMMediate , EXTernalと等しいトリガ・ソースを選択します。

7.16　R3762/63 コマンド

1. CONT

- ●機能　掃引モードをCONTにする
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command
- ●コマンド　CONT
- ●説明　連続に掃引、測定をします。

2. MEAS

- ●機能　測定を実行する
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command
- ●コマンド　MEAS
- ●説明　掃引中の場合、掃引をリセット後、1 回掃引、測定をします。
  掃引モードがCONTである場合は、続けて連続に掃引、測定します。

3. SINGLE

- ●機能　掃引モードをSINGLEにする
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command
- ●コマンド　SINGLE
- ●説明　1 回掃引、測定をします。

4. SWPHLD

- ●機能　掃引をホールドする
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command
- ●コマンド　SWPHLD
- ●説明　掃引を即座に停止します。

7.17　R3765/67 MARKerサブシステム

1. MARKer[<chno>]:ACTivate[:NUMBer]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
MKR{1|2|3|4|5|6|7|8|9|10}A

● 機能　　アクティブ・マーカの設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　MARKer[<chno>]:ACTivate[:NUMBer] <n>[,<real>]

パラメータ　　<n> = 1～10（マーカ番号）
<real> = 設定値（スティミュラス値）

応答形式　　NR1(整数値) : 0 ～10（マーカ番号）
NR3(実数値) : 設定値（スティミュラス値）

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　MKR{1|2|3|4|5|6|7|8|9|10}A

応答形式　　NR3(実数値) : 設定値（スティミュラス値）
NR3(実数値) : 測定値（データA,B,C）
NR1(整数値) : ステータス

● 説明　　アクティブ・マーカの番号を指定します。指定されたマーカは自動的にONになります。

クエリは、IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞの場合は、マーカ番号と設定値が返ります。マーカが 1つもONされていなければ、マーカ番号に 0が返ります。
測定値は、FETCh?クエリで取得できます。
IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞの場合は、設定値と測定値が返ります。
データおよびフォーマットは、[7.18 FETCh?サブシステム] を参照して下さい。

2. MARKer[<chno>]:ACTivate:STATe　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
MKROFF

● 機能　　マーカのON/OFF

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　　Command/Query

● コマンド　　MARKer[<chno>]:ACTivate:STATe <bool>
MKROFF

● パラメータ　　<bool>

● 応答形式　　0|1

● 説明　　アクティブ・マーカを OFFし、それ以外のマーカがONになっていると、その中で最も小さい番号のマーカをアクティブ・マーカに変更します。

IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞの場合で、パラメータがONの場合は、全てのマーカが OFFのときのみ、マーカ1 をONにします。

3. MARKer[<chno>]:ACTivate:STIMulus　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
MKR{1|2|3|4|5|6|7|8|9|10}A

● 機能　　マーカのスティミュラス値の指定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　MARKer[<chno>]:ACTivate:STIMulus <real>

パラメータ　　<real> = スティミュラス値

応答形式　　NR3(実数値) : スティミュラス値

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　MKR{1|2|3|4|5|6|7|8|9|10}A <real>

パラメータ　　<real> = スティミュラス値

応答形式　　NR3(実数値) : 設定値 (スティミュラス値)
NR3(実数値) : 測定値 (データA,B,C)
NR1(整数値) : ステータス

● 説明　　アクティブ・マーカのスティミュラス値を設定します。

クエリは、IEEE488.-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ の場合、設定値が返ります。
測定値は、FETch?クエリで取得できます。
IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞの場合は、設定値と測定値が返ります。

4. MARKer[<chno>]:AOFF　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ MKRAOFF

- ●機能　全マーカのOFF
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command
- ●コマンド　MARKer[<chno>]:AOFF
MKRAOFF
- ●説明　全マーカをOFF します。

5. MARKer[<chno>]:COMPensate　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
MKRCMP
MKRUCMP

● 機能　　マーカ補間モードのON/OFF

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　MARKer[<chno>]:COMPensate <bool>

パラメータ　　<bool>

応答形式　　0|1

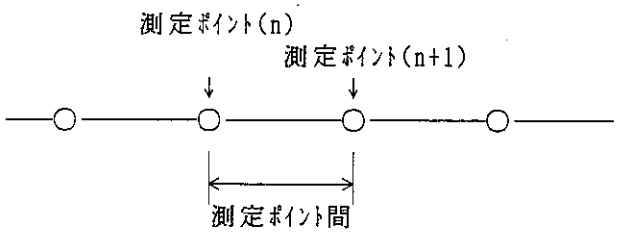
● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　MKRCMP　-----> ON
MKRUCMP　-----> OFF

応答形式　　0|1

● 説明　　マーカ補間モードとは、測定ポイント間のデータを直線近似により補間して求める機能です。

OFF : 測定ポイントだけにマーカ設定ができます。測定ポイント以外のスティミュラス値を指定した場合は、近傍の測定ポイントに自動的に変更されます。

ON　: 測定ポイント間も補間によりマーカ設定ができます。



6.　MARKer[<chno>]:CONVert[:MODE]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
ZYMK{DFLT|LIN|RI|LC}

● 機能　　マーカ・コンバージョン・モードの設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　MARKer[<chno>]:CONVert[:MODE] <format>

　パラメータ　　<format> = {DEFault|LINear|RIMaginary}

　応答形式　　DEF|LIN|RIM

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　ZYMK{DFLT|LIN|RI|LC}

　応答形式　　0|1

● 説明　　測定フォーマットと関係なく、マーカの測定値のフォーマットを設定します。測定値のパラメータ・コンバージョンを実行している場合に有効です。

R3762/63 コマンド	R3764/66、R3765/67 ｺﾏﾝﾄﾞ･ﾊﾟﾗﾒｰﾀ	ﾏｰｶ･ﾌｫｰﾏｯﾄ
ZYMKDFLT	DEFault	測定フォーマットと同じ
ZYMKLIN	LINear	リニア・インピーダンス
ZYMKRI	RIMaginary	虚数インピーダンス

7.　MARKer[<chno>]:COUPle　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
MKRCOUP
MKRUCOUP

● 機能　　マーカ・カップル・モードのON/OFF

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　MARKer[<chno>]:COUPle <bool>

パラメータ　　<bool>

応答形式　　0|1

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　MKRCOUP -----> ON
MKRUCOUP -----> OFF

応答形式　　0|1

● 説明　　チャンネル 1、2 、3 、4 のマーカ・カップリングのON/OFFを設定します。

ON ：アクティブ・チャンネルで設定したマーカが、他のチャンネルにも自動設定されます。

OFF ：チャンネル 1、2 、3 、4 を別々に設定します。

8. MARKer[<chno>]:DELTa[:MODE]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ DMKR{C|A|F|OF}

● 機能　　デルタ・マーカの設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

コマンド　　MARKer[<chno>]:DELTa[:MODE] <type>

パラメータ　　<type> = {OFF|CHILd|COMPare|FIXed}

応答形式　　OFF|CHIL|COMP|FIX

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

コマンド　　DMKRC
DMKRA
DMKRF
DMKROF

応答形式　　0|1

● 説明　　デルタ・マーカのモードを設定します。

R3762/63 コマンド	R3764/66、R3765/67 ｺﾏﾝﾄﾞ･ﾊﾟﾗﾒｰﾀ	モード
DMKRC	CHIL	ｱｸﾃｨﾌﾞ･ﾏｰｶ位置にﾁｬｲﾙﾄﾞ･ﾏｰｶ を設定し、ｱｸﾃｨﾌﾞ･ﾏｰｶ とﾁｬｲﾙﾄﾞ･ﾏｰｶ 間の差を求めます。
DMKRA	COMP	ｱｸﾃｨﾌﾞ･ﾏｰｶと他のマーカ間の差を求めます。
DMKRF	FIX	固定ﾏｰｶ（FIX MKR）とｱｸﾃｨﾌﾞ･ﾏｰｶ間の差を求めます。
DMKROF	OFF	ﾃﾞﾙﾀ･ﾏｰｶ･ﾓｰﾄﾞ を OFFします。

（注）デルタ・モードをCOMPに設定するには、その前にコンペア・マーカを指定して下さい。

IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞの場合もデルタ・スティミュラスの設定はできなくなっています。

9.　MARKer[<chno>]:DELTa:COMPare　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
DMKR{1|2|3|4|5|6|7|8|9|10}O

● 機能　　コンペア・マーカの指定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　MARKer[<chno>]:DELTa:COMPare <n>[,<real>]

パラメータ　　<n> = 1～10（マーカ番号）
<real> =スティミュラス値（ｱｸﾃｨﾌﾞ･ﾏｰｶからの相対値）

応答形式　　<NR1>(整数値) : 0～10（マーカ番号）
<NR3>(実数値) : スティミュラス値（ｱｸﾃｨﾌﾞ･ﾏｰｶからの相対値）

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　DMKR{1|2|3|4|5|6|7|8|9|10}O<real>

パラメータ　　<real> =スティミュラス値（ｱｸﾃｨﾌﾞ･ﾏｰｶからの相対値）

応答形式　　0|1

● 説明　　デルタ・マーカがCOMPare モードに設定されている場合、比較するマーカを指定します。またその位置をアクティブ・マーカからの相対値で設定します。

10. MARKer[<chno>]:FANalysis:DIRection　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
TIN
TOUT

● 機能　　フィルタ解析の方向設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　MARKer[<chno>]:FANalysis:DIRection <type>

パラメータ　　<type> = {IN|OUT}

応答形式　　IN|OUT

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　TIN
TOUT

応答形式　　0|1

● 説明　　フィルタ解析の方向を設定します。

R3762/63 コマンド	R3764/66、R3765/67 ｺﾏﾝﾄﾞ･ﾊﾟﾗﾒｰﾀ	方向
TIN	IN	ｱｸﾃｨﾌﾞ･ﾏｰｶ より外側へサーチします。
TOUT	OUT	外側からｱｸﾃｨﾌﾞ･ﾏｰｶ へサーチします。

11. MARKer[<chno>]:FANalysis[:STATe]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ FLTANA

● 機能　フィルタ解析のON/OFF

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● コマンド　MARKer[<chno>]:FANalysis[:STATe] <bool>
FLTANA <bool>

● パラメータ　<bool>

● 応答形式　0|1

● 説明　フィルタ解析のON/OFFを設定します。

フィルタ解析では以下の項目を測定します。
・アクティブ・マーカからの解析幅（ロス）で指定された通過帯域の中心周波数
・通過帯域幅
・通過帯域の左側周波数
・通過帯域の右側周波数
・品質係数(Qファクタ)
・選択度（ｼｪｰﾋﾟﾝｸﾞ･ﾌｧｸﾀ）

品質係数(Qファクタ)/選択度（ｼｪｰﾋﾟﾝｸﾞ･ﾌｧｸﾀ)については、損失最小値からデータを求めています。

12. MARKer[<chno>]:FANalysis:WIDTh

IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
T{3|6|60|X}DB
T{3|6|X}DEG

● 機能　　フィルタ解析の解析幅の設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

コマンド　　MARKer[<chno>]:FANalysis:WIDTh <real>

パラメータ　　<real> = 解析幅（通過帯域幅）

応答形式　　NR3(実数値) : 解析幅（通過帯域幅）

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

コマンド

T3DB	T3DEG
T6DB	T6DEG
T60DB	TXDEG<real>
TXDB<real>	

パラメータ　　<real> = 解析幅（通過帯域幅）

応答形式

NR3(実数値) : CENTER
NR3(実数値) : LEFT
NR3(実数値) : RIGHT
NR3(実数値) : BAND
NR3(実数値) : QUALITY FACTOR
NR3(実数値) : SHAPE FACTOR
NR1(整数値) : ステータス

● 説明　　フィルタ解析の解析幅（通過帯域幅）を設定します。

IEEE488.1-1987コマンド・モードの場合、3dB, 6dB, 60dBを設定するときは、それぞれT3DB, T6DB, T60DB のコマンドで実行します。
TXDBのときだけ<real>にて任意の値を設定します。

また位相で設定する場合も同様に、3deg, 6degのときは、T3DEG, T6DEG にて実行します。TXDEG のときだけ<real>にて任意の値を設定します。

13. MARKer[<chno>]:FIXed:STIMulus　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ FMKRS

- ●機能　固定マーカ (FIX MKR) X軸の設定
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query
- ●コマンド　MARKer[<chno>]:FIXed:STIMulus <real>
  FMKRS <real>
- ●パラメータ　<real> =　X軸値
- ●応答形式　<NR3> 実数値 : X軸値
- ●説明　直交座標表示の場合、固定マーカ (FIX MKR)の X軸値を設定します。

  固定マーカ (FIX MKR)は、パラメータ・コンバージョンがOFF または1/S のときだけ有効です。

14. MARKer[<chno>]:FIXed:VALue　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ FMKRV

- ●機能　固定マーカ（FIX MKR）Y軸の設定
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query
- ●コマンド　MARKer[<chno>]:FIXed:VALue <real>
  FMKRV <real>
- ●パラメータ　<real> =　Y軸値
- ●応答形式　<NR3> 実数値 ： Y軸値
- ●説明　直交座標表示の場合は、固定マーカ（FIX MKR)の Y軸値を設定します。

  極座標表示の場合は、実数部の値となります。

15.　MARKer[<chno>]:FIXed:AVALue

●機能	固定マーカ (FIX MKR)　虚数部の設定
●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在	Command/Query
●コマンド	MARKer[<chno>]:FIXed:AVALue <real>
●パラメータ	<real> =　虚数部
●応答形式	<NR3> 実数値 :　虚数部
●説明	極座標表示の場合に、固定マーカ (FIX MKR)の虚数部を設定します。

16. MARKer[<chno>]:LET　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ MKR{REF|CENT|STAR|STOP|SPAN|FIX}

● 機能　　マーカ代入機能

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　MARKer[<chno>]:LET <type>

　パラメータ　　<type> = {STARt|STOP|CENTer|SPAN|RLBVel|FIXed}

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　MKR{REF|CENT|STAR|STOP|SPAN|FIX}

● 説明　　アクティブ・マーカの設定値、測定値を各種設定パラメータに代入します。

R3762/63 コマンド	R3764/66、R3765/67 ｺﾏﾝﾄﾞ･ﾊﾟﾗﾒｰﾀ	動作
MKRREF	RLEV	ｱｸﾃｨﾌﾞ･ﾏｰｶ の Y軸値（測定値）をﾘﾌｧﾚﾝｽ値に代入します。
MKRCENT	CENT	ｱｸﾃｨﾌﾞ･ﾏｰｶ の X軸値（設定値）を掃引の ｾﾝﾀへ代入します。
MKRSTAR	STAR	ｱｸﾃｨﾌﾞ･ﾏｰｶ の X軸値（設定値）を掃引のｽﾀｰﾄへ代入します。
MKRSTOP	STOP	ｱｸﾃｨﾌﾞ･ﾏｰｶ の X軸値（設定値）を掃引のｽﾄｯﾌﾟへ代入します。
MKRSPAN	SPAN	ｱｸﾃｨﾌﾞ･ﾏｰｶ の X軸値（設定値）を掃引の ｽﾊﾟﾝへ代入します。
MKRFIX	FIX	ｱｸﾃｨﾌﾞ･ﾏｰｶ の位置を固定ﾏｰｶ（FIX MKR）へ代入します。

17.　MARKer[<chno>]:LIST

- ●機能　　マーカリスト表示のON/OFF
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query
- ●コマンド　　MARKer[<chno>]:LIST <bool>
- ●パラメータ　　<bool>
- ●応答形式　　0|1
- ●説明　　マーカリスト表示のON/OFFを選択します。

18. MARKer[<chno>]:POLar　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ PMKR{LIN|LOG|RI }

● 機能　　ポーラ表示の場合のマーカ・モード設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　MARKer[<chno>]:POLar <type>

　パラメータ　　<type> = {MLINear|MLOGarithmic|RIMaginary}

　応答形式　　MLIN|MLOG|RIM

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　PMKR{LIN|LOG|RI}

　応答形式　　0|1

● 説明　　ポーラ表示の場合のマーカ・モードを設定します。

R3762/63 コマンド	R3764/66、R3765/67 ｺﾏﾝﾄﾞ･ﾊﾟﾗﾒｰﾀ	モード
PMKRLIN	MLIN	リニア値
PMKRLOG	MLOG	ログ値
PMKRRI	RIM	複素数値

19.　MARKer[<chno>]:SMITh　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ SMKR{LIN|LOG|RI|RX|GB}

●機能　　スミス・チャート表示のマーカ・モード設定

●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

●IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　MARKer[<chno>]:SMITh <type>

　パラメータ　　<type> = {MLINear|MLOGarithmic|RIMaginary |IMPedance |ADMittance}

　応答形式　　MLIN|MLOG|RIM|IMP|ADM

●IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　SMKR{LIN|LOG|RI|RX|GB}

　応答形式　　0|1

●説明　　スミス・チャート表示のマーカ・モードを設定します。

R3762/63 コマンド	R3764/66、R3765/67 ｺﾏﾝﾄﾞ･ﾊﾟﾗﾒｰﾀ	モード
SMKRLIN	MLIN	リニア値
SMKRLOG	MLOG	ログ値
SMKRRI	RIM	複素数値
SMKRRX	IMP	インピーダンス値
SMKRGB	ADM	アドミッタンス値

20.　MARKer[<chno>]:SEARch[:MODE]

IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
SRCHOFF
{MAX|MIN}SRCH
ZRPSRCH
DRIPPL1

● 機能　　マーカ・サーチ機能

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● R3752/53ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

コマンド　　MARKer[<chno>]:SEARch[:MODE] <type>

パラメータ　　<type> = {OFF|MAX|MIN|TARGet|RIPPle}

応答形式　　OFF|MAX|MIN|TARG|RIPP

● R3751ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

コマンド

SRCHOFF
{MAX|MIN}SRCH
ZRPSRCH
DRIPPL1

応答形式

SRCHOFF　……: 0|1
{MAX|MIN}SRCH　……: NR3(実数値) : 設定値 (ｽﾃｨﾐｭﾗｽ値)
ZRPSRCH　……: NR3(実数値) : 測定値 (データA,B,C)
DRIPPL1　……: NR1(整数値) : ステータス

● 説明　　マーカ・サーチ機能を指定します。

R3762/63 コマンド	R3764/66、R3765/67 ｺﾏﾝﾄﾞ･ﾊﾟﾗﾒｰﾀ	サーチ・モード
SRCHOFF	OFF	OFF
MAXSRCH	MAX	最大値
MINSRCH	MIN	最小値
ZRPSRCH	TARG	指定値
DRIPPL1	RIPP1	リップル

クエリは、IEEE488.2-1987コマンド・モードの場合、サーチ・モードを返します。測定値は、FETch?クエリで取得できます。
IEEE488.1-1987コマンド・モードの場合は、測定値を返します。

21.　MARKer[<chno>]:SEARch:PARTial:SRANge

● 機能　　部分マーカ・サーチ範囲の指定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　MARKer[<chno>]:SEARch:PARTial:SRANge

● 説明　　部分マーカ・サーチの範囲を指定します。デルタ・マーカで設定されているマーカ間をサーチの範囲とします。
デルタ・マーカがOFF の場合は無効です。

このコマンドは範囲の指定のみで、部分サーチのON/OFFは"MARK:SEAR:PART:STAT" コマンドで設定します。

（注）IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞでは、MKRPART ONで自動的に実行します。

22. MARKer[<chno>]:SEARch:PARTial[:STATe]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ MKRPART

- ●機能　部分マーカ・サーチのON/OFF
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query
- ●コマンド　MARKer[<chno>]:SEARch:PARTial[:STATe] <bool>
  MKRPART <bool>
- ●パラメータ　<bool>
- ●応答形式　0|1
- ●説明　部分マーカ・サーチのON/OFFを指定します。

23.　MARKer[<chno>]:SEARch:RIPPle[:MODE]　　IEEE488.1-1987ｺﾏﾝﾄﾞ・ﾓｰﾄﾞ
DRIPPL1
DMAXMIN

● 機能　　リップル・サーチのモード指定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ・ﾓｰﾄﾞ
　コマンド　　MARKer[<chno>]:SEARch:RIPPle[:MODE] <type>

　パラメータ　　<type> = {MAX|MIN|BOTH|PPEak}

　応答形式　　MAX|MIN|BOTH|PPEak

● 説明　　リップル・サーチのモードを指定します。

R3762/63 コマンド	R3764/66、R3765/67 ｺﾏﾝﾄﾞ・ﾊﾟﾗﾒｰﾀ	モード
	MAX	極大値の最大値を求めます。
	MIN	極小値の最小値を求めます。
DRIPPL1	BOTH	極大値の最大値と極小値の最小値との差を求めます。
DMAXMIN	PPEak	最大値と最小値との差を求めます。

（注）DRIPPL2 は、サポートしていません。

24.　MARKer[<chno>]:SEARch:RIPPle{:DX|:DY}　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ DLT{X|Y}

● 機能　　リップル・サーチ検出感度の設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

コマンド　　MARKer[<chno>]:SEARch:RIPPle{:DX|:DY} <real>
DLT{X|Y} <real>

パラメータ　　<real> = 設定値

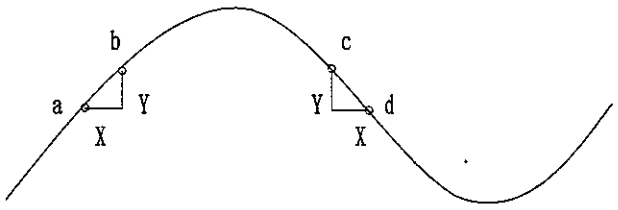
応答形式　　<NR3> 実数値 : 設定値

● 説明　　リップル・サーチの検出感度を設定します。

検出感度をΔY/ΔX とした場合、リップルを求めるには、まず波形の傾き(Y/X) がΔY/ΔX 以上になる a点を求め、次に逆の傾きがΔY/ΔX 以上になる d点を求め、a, d点間での最大値を極大値として求めます。
極小値は、逆の傾きで同様に求めます。

IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ : DX → ΔX の設定
DY → ΔY の設定

IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ : DLTX→ ΔX の設定
DLTY→ ΔY の設定



25.　MARKer[<chno>]:SEARch:TARGet[:MODE]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ ZRPSRCH

● 機能　　ターゲット・サーチのモード設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　MARKer[<chno>]:SEARch:TARGet[:MODE] <type>

　パラメータ　　<type> = {ZERO|PI|VALue}

　応答形式　　ZERO|PI|VALue

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　ZRPSRCH

　応答形式　　NR3(実数値) : 設定値 (スティミュラス値)
　　NR3(実数値) : 測定値 (データA,B,C)
　　NR1(整数値) : ステータス

● 説明　　ターゲット・サーチのモードを指定します。

R3762/63 コマンド	R3764/66、R3765/67 ｺﾏﾝﾄﾞ･ﾊﾟﾗﾒｰﾀ	モード
ZRPSRCH	ZERO	位相 0deg をサーチします。
	PI	位相±180degをサーチします。
	VAL	指定した値をサーチします。

26. MARKer[<chno>]:SEARch:TARGet:VALue

- ●機能　ターゲット・サーチの値指定
- ●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query
- ●コマンド　MARKer[<chno>]:SEARch:TARGet:VALue <real>
- ●パラメータ　<real>
- ●応答形式　<NR3> 実数値
- ●説明　ターゲット・サーチのモードを指定値サーチに設定したときの、指定値を設定します。

27.　MARKer[<chno>]:SEARch:TARGet:LEFT

● 機能　　左側周波数サーチ

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command

● コマンド　　MARKer[<chno>]:SEARch:TARGet:LEFT

● 説明　　ターゲット・サーチの指定値サーチ・モードにて、左側へ次のサーチを実行します。

28. MARKer[<chno>]:SEARch:TARGet:RIGHt

● 機能　右側周波数サーチ

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command

● コマンド　MARKer[<chno>]:SEARch:TARGet:RIGHt

● 説明　ターゲット・サーチの指定値サーチ・モードにて、右側へ次のサーチを実行します。

29.　MARKer[<chno>]:SEARch:TRACking　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
MKRTRAC

●機能　トラッキング・モードのON/OFF

●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

●コマンド　MARKer[<chno>]:SEARch:TRACking <bool>
MKRTRAC <bool>

●パラメータ　<bool>

●応答形式　0|1

●説明　トラッキング・モードが
ONの場合　：掃引終了ごとにマーカ・サーチを実行します。

（注）トラッキングをONにしてから、マーカ・サーチを指定して下さい。

OFF の場合：マーカ・サーチを指定したとき 1回だけマーカ・サーチを実行します。

7.18　FETCh? サブシステム

1. FETCh[<chno>][:MARKer][:ACTivate]?

● 機能　アクティブ・マーカ出力

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Query

● コマンド　FETCh[<chno>][:MARKer][:ACTivate]?

● 説明　アクティブ・マーカの最新データを出力します。
出力データは、ASCII フォーマットで転送します。

出力フォーマット

SN.NNNNNNNNNNNNNNESNN, SN.NNNNNNNNNNNNNNESNN,
<スティミュラス> <データA>

SN.NNNNNNNNNNNNNNESNN, SN.NNNNNNNNNNNNNNESNN,
<データB> <データC>

SN NL ^ END
<ステータス>

<スティミュラス> マーカ・ポイント位置の X軸値を示します。

フォーマットは、以下に示す22文字の固定長フォーマットです。

SN.NNNNNNNNNNNNNNESNN
(S:+/-, N: 0～9, E: 指数符号)

アクティブ・マーカが無効なときは、+1.000000000000000E+38になります。

デルタ・マーカが有効なときは、マーカ間のスティミュラス差となります。

<データA, B> データA は、第 1波形の演算データです。データB は、第 2波形の演算データです。
メモリ波形は、データB となります。

極座標およびスミスチャート設定になっている場合は、データA が実数部、データB が虚数部の値となります。

データ・フォーマットは、<スティミュラス> と同じです。
有効なデータがない場合は、+1.000000000000000E+38になります。

〈データC〉　極座標およびスミスチャート設定の場合のみ有効です。
このとき、リアクタンス値または容量値となります。

データ・フォーマットは、〈スティミュラス〉と同じです。
有効なデータがない場合は、+1.000000000000000E+38になります。

〈ステータス〉　以下の演算データのステータスを表します。

-1 :　データなし
 0 :　正常演算のデータ
 1 :　演算不可能な測定データ
 2 :　フィルタ解析のレベル1 エラー
 3 :　フィルタ解析のレベル2 エラー
 4 :　フィルタ解析のレベル3 エラー
 5 :　フィルタ解析のレベル4 エラー

フォーマットは、 1文字または 2文字の整数値です。

2.　FETCh[<chno>][:MARKer]:FANalysis?

●機能　　フィルタ解析データ出力

●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Query

●コマンド　　FETCh[<chno>][:MARKer]:FANalysis?

●説明　　フィルタ解析の結果を出力します。

フィルタ解析は、第1 波形データで実行します。ただし、データ波形がOFF のときは、メモリ波形データを使います。

出力データは、ASCII フォーマットで転送されます。

出力フォーマット

SN.NNNNNNNNNNNNNNESNN, <CENTER FREQ>	SN.NNNNNNNNNNNNNNESNN, <LEFT FREQ>
SN.NNNNNNNNNNNNNNESNN, <RIGHT FREQ>	SN.NNNNNNNNNNNNNNESNN, <BANDWIDTH>
SN.NNNNNNNNNNNNNNESNN, <QUALITYFACTOR>	SN.NNNNNNNNNNNNNNESNN, <SHAPEFACTOR>

SN　　NL^END
<ステータス>

<CENTER FREQ>　　フィルタの中心周波数です。

フォーマットは、以下の22文字の固定長フォーマットです。

SN.NNNNNNNNNNNNNNESNN
(S:+/-, N:0～9, E:指数符号)

アクティブ・マーカが無効なときは、+1.000000000000000E+38になります。
デルタ・マーカが有効でも、マーカ間の周波数差は転送されません。

<LEFT FREQ>　　サーチした帯域幅の左側周波数です。

フォーマットは、<CENTER FREQ> と同じです。
有効なデータがない場合は、+1.000000000000000E+38になります。

<RIGHT FREQ>　　サーチした帯域幅の右側周波数です。

フォーマットは、<CENTER FREQ> と同じです。
有効なデータがない場合は、+1.000000000000000E+38になります。

<BANDWIDTH>　サーチした帯域幅です。

フォーマットは、<CENTER FREQ> と同じです。
有効なデータがない場合は、+1.000000000000000E+38になります。

<QUALITYFACTOR>　品質係数です。

フォーマットは、<CENTER FREQ> と同じです。
有効なデータがない場合は、+1.000000000000000E+38になります。

<SHAPEFACTOR>　選択度です。

フォーマットは、<CENTER FREQ> と同じです。
有効なデータがない場合は、+1.000000000000000E+38になります。

<ステータス>　以下の演算データのステータスを表します。

-1 :　データなし
 0 :　正常演算のデータ
 1 :　演算不可能な測定データ

フォーマットは、 1文字または 2文字の整数値です。

3. FETCh[<chno>][:MARKer]:NUMBer<n>?

● 機能　指定番号マーカ・データ出力

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Query

● コマンド　FETCh[<chno>][:MARKer]:NUMBer<n>?

● パラメータ　<n> = 0～10

● 説明　指定した番号のマーカ・データを出力します。

番号 0は、アクティブ・マーカです。

フォーマットは、アクティブ・マーカ出力フォーマットと同じです。

7.19　LIMitサブシステム

1. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:BEEP　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ FAILBEEP

● 機能　　リミットFAIL時のビープ音のON/OFF

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:BEEP <bool>

　パラメータ　　<bool>

　応答形式　　0 | 1

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　FAILBEEP<bool>

　パラメータ　　<bool>

　応答形式　　0 | 1

● 説明　　リミット・テストFAIL時のビープ音の有無を選択します。

　リミット・テスト機能(DISP:LIM)がONで、ビープ機能(SYST:BEEP:STAT)がONの場合に、このコマンドをONにするとビープ音が許可されます。
　パラメータ<parano>の指定は無効です。

2. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:CLEar　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ LSEGCL

● 機能　　リミット・テーブルの全セグメントのクリア

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:CLEar

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　LSEGCL

● 説明　　リミット・テーブルの全セグメントの内容をクリアします。

リミット・テーブルは、チャンネルごとに 2 組（パラメータ）存在します。第2 パラメータのテーブルを対象にする場合には、<parano>に 2を指定して下さい。
セグメントを部分的に削除するには、DISP:LIM:SEGM:DEL を用います。

3. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:DATA<block>

●機能　　リミット・テーブルの全セグメントの情報の設定

●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

コマンド　　DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:DATA<block>

パラメータ　　<block> = #<byte><length><data>
<byte> = ブロック長を示す次の文字列のバイト数をアスキー数字(1文字) で記述します。
<length> = データ列を示す文字列の文字数をアスキー数字で記述します。
<data> = セグメントの各要素を、<stimulus>,<upper>,<lower>,<type>,<color>,<wcolor>,... の順で必要なセグメント全部を記述します。
<stimulus> = スティミュラス値
<upper> = 上限値
<lower> = 下限値
<type> = ﾗｲﾝ･ﾀｲﾌﾟ {SLINe|FLINe|SPOint}
<color> = ﾘﾐｯﾄ･ﾗｲﾝ表示色 {1-7}
<wcolor> = 信号波形の表示色 {1-7}

応答形式　　<block>

●説明　　リミット・テーブルの全セグメント情報を、完結した形で設定します。前回のセグメント情報は失われます。
すべてのデータを受けると、スティミュラス値の昇順にセグメントの並び変えを行います。
データの途中に記述の誤りが見つかった場合、そこまでのセグメントを有効とし、残りは無視します。
ブロック・データ<block> については、[3.1.2 データ・フォーマット] を参照して下さい。

●例　　DISP:LIM:DATA #2463GHz,5dB,-5dB,SLIN,2,6,6GHz,10dB,-10dB,SPO,2,6

4. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:LINE　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ LIMILINE

● 機能　　リミット・ライン画面表示のON/OFF

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:LINE<bool>

　パラメータ　　<bool>

　応答形式　　0 | 1

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　　LIMILINE<bool>

　パラメータ　　<bool>

　応答形式　　0 | 1

● 説明　　リミット・ラインの画面表示のON/OFFを選択します。
このコマンドをONにすると、ディスプレイのスケール上にリミット・ラインが表示されます。
実際にリミット・テストを実行するには、DISP:LIMをONする必要があります。

5. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
:OFFSet:AMPLitude　　LIMIAMPO

●機能　　全セグメントのリミット値にオフセット値を加減算する。

●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

●IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　DISPlay[:WINDow[<chno>]]:LIMit[<parano>]
:OFFSet:AMPLitude <real>

パラメータ　　<real>

応答形式　　NR3 （実数値）

●IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　LIMIAMPO

パラメータ　　<real>

応答形式　　NR3 （実数値）

●説明　　リミット・ラインを、指定したオフセット値に従って上下に移動します。
スティミュラス値にオフセット値を加えるには、DISP:LIM:OFFS:STIMコマンドを用います。

6. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:OFFSet:STIMulus <real>　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ LIMISTIO

● 機能　　全セグメントのスティミュラス値にオフセット値を加減算する。

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

コマンド　　DISPlay[:WINDow[<chno>]]:LIMit[<parano>]
:OFFSet:STIMulus <real>

パラメータ　　<real>

応答形式　　NR3　(実数値)

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

コマンド　　LIMISTIO<real>

パラメータ　　<real>

応答形式　　NR3　(実数値)

● 説明　　リミット・ラインを、指定したオフセット値に従って上下に移動します。
レスポンス値にオフセット値を加えるには、DISP:LIM:OFFS:AMPL コマンドを用います。

7. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
:ParallelIO　　LIMPIO

● 機能　　パラレルI/O へのリミット・ライン判定結果出力のON/OFF

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:ParallelIO <bool>

パラメータ　　<bool>

応答形式　　0 | 1

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　LIMPIO<bool>

パラメータ　　<bool>

応答形式　　0 | 1

● 説明　　リミット・テスト結果のパラレルI/O(PIO)出力のON/OFFを選択します。
リミット・テスト(DISP:LIM)がONの状態で、このコマンドをONにすると、PIO への結果出力が許可されます。

8. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
:PARameter:PolarLIMit　　LIMPLIN|LIMPLOG

● 機能　　ポーラ表示フォーマット時の判定パラメータの組み合わせを選択

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　DISPlay[:WINDow[<chno>]]:LIMit[<parano>]
:PARameter:PolarLIMit <select>

パラメータ　　<select> = {LINear|LOGarithmic}

応答形式　　LIN|LOG

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　LIMPLIN|LIMPLOG

応答形式　　0 | 1

● 説明　　表示フォーマットにポーラ表示(CALCulate[:FORMat]POLar) が選択されている場合、判定パラメータは振幅と位相の組み合わせになります。
このコマンドでは、振幅をリニア形式にするか対数（ログ）形式にするかを選択します。

R3762/63 コマンド	R3764/66、R3765/67 ｺﾏﾝﾄﾞ･ﾊﾟﾗﾒｰﾀ	判定ﾊﾟﾗﾒｰﾀ <parano>
LIMPLIN	LINear	0;振幅(ﾘﾆｱ)　1;位相
LIMPLOG	LOGarithmic	0;振幅（対数）1;位相

対応するチャンネルの表示フォーマットが直交座標系の場合、ここでの設定内容は影響しません。

9. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
:PARameter:SmithLIMit　　LIMSLIN|LIMSLOG

●機能　　スミス表示フォーマット時の判定パラメータの組み合わせを選択

●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

●IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　DISPlay[:WINDow[<chno>]]:LIMit[<parano>]
:PARameter:SmithLIMit <select>

パラメータ　　<select> = {LINear|LOGarithmic}

応答形式　　LIN|LOG

●IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　LIMSLIN|LIMSLOG

応答形式　　0 | 1

●説明　　表示フォーマットにスミスチャート(CALCulate[:FORMat]SCHart|ISCHart)が選択されている場合、判定パラメータは振幅と位相の組み合わせになります。
このコマンドでは、振幅をリニア形式にするか対数（ログ）形式にするかを選択します。

R3762/63 コマンド	R3764/66、R3765/67 ｺﾏﾝﾄﾞ･ﾊﾟﾗﾒｰﾀ	判定ﾊﾟﾗﾒｰﾀ <parano>
LIMPLIN	LINear	0;振幅(ﾘﾆｱ)　1;位相
LIMPLOG	LOGarithmic	0;振幅（対数）1;位相

対応するチャンネルの表示フォーマットが直交座標系の場合、ここでの設定内容は影響しません。

10. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
:PARameter[:STATe]　　LIMPAR

● 機能　　個々の判定パラメータ設定のON/OFF

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　DISPlay[:WINDow[<chno>]]:LIMit[<parano>]
:PARameter[:STATe] <bool>

パラメータ　　<bool>

応答形式　　0 | 1

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　LIMPAR<bool>

パラメータ　　<bool>

応答形式　　0 | 1

● 説明　　個々の判定パラメータ<parano>のON/OFFを設定します。

<parano>	判定パラメータ
0	ﾒｲﾝ･ﾄﾚｰｽ/ 実数部/ 振幅
1	ｻﾌﾞ･ﾄﾚｰｽ/虚数部/ 位相

リミット・テストを実際に実行するには、リミットを設定した後に、DISP:LIM ON でテストをONにして下さい。指定したパラメータをONにしても、セグメントが1 つも設定されていなければ、ここでの設定は無効になります。

11. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:REPort?

●機能　全セグメントについてのPASS/FAIL 情報を報告

●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Query

●IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

クエリ　DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:REPort?

応答形式　<block>

データ・フォーマット設定(FORMat[:DATA]) によって出力形式が異なります。

アスキー・フォーマット(FORMat[:DATA] ASCii) の場合
<block> = [,,....]
 = 0 ～30の数字(ASCII文字列)

バイナリ・フォーマット(FORMat[:DATA] {REAL|MBIN},{32|64}) の場合
<block> = #<byte>[<length>]<data>
<byte> = ブロック長を示す文字列の文字数をアスキー数字(1文字で指示します。
<length> = データ長を示す文字列の文字数をアスキー数字で指示します。
<data> = FAIL となったセグメント番号
(1バイト整数の並び、昇順)

●説明　全セグメントについてのPASS/FAIL 情報を、まとめて報告します。テストの結果のみを知りたい場合は、DISP:LIM:RES? を使用します。

ブロック・データ<block> については、[3.1.2 データ・フォーマット] を参照して下さい。
データ・フォーマットについては、[7.7.2 FORMat[:DATA]] を参照して下さい。

12. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:RESult?　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ LIMRES?

● 機能　　テスト結果のPASS/FAIL 情報を報告

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　クエリ　　DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:RESult?

　応答形式　　PASS|FAIL|OFF|UND

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　クエリ　　LIMRES?

　応答形式　　PASS|FAIL|OFF|UND

● 説明　　テスト結果のPASS/FAIL を返します。
ただし、リミット・テストがOFF の場合にはOFF を、リミット値が未定義の場合にはUNDefined を返します。

13. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>

● 機能　　指定セグメントの全情報を一度に設定。

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

クエリ　　DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n> <block>

パラメータ

<block> = #<byte><length><data>
<byte> = ブロック長を示す次の文字列のバイト数をアスキー数字(1文字) で記述します。
<length> = データ列を示す文字列の文字数をアスキー数字で記述します。
<data> = セグメントの各要素を、<stimulus>,<upper>,<lower>,<type>,<color>,<wcolor>,... の順で必要なセグメント全部を記述します。
<stimulus> = スティミュラス値
<upper> = 上限値
<lower> = 下限値
<type> = ﾗｲﾝ･ﾀｲﾌﾟ {SLINe|FLINe|SPOint}
<color> = ﾘﾐｯﾄ･ﾗｲﾝ表示色 {1-7}
<wcolor> = 信号波形の表示色 {1-7}

応答形式　　<block>

● 説明

1 つのセグメントに必要な情報を一度に設定します。指定したセグメントが空でない場合、新しい内容が上書きされます。
セグメント番号<n> は、0 ～30の範囲で設定します。
すべてのデータを受けると、スティミュラス値の昇順にセグメントの並び変えを行います。
それまでに設定されているセグメント数よりも、指定セグメントが大きい場合には、セグメントの指定は、無視し、最初の空のセグメントに追加します。

ブロック・データ<block> については、[3.1.2　データ・フォーマット] を参照して下さい。

● 例　　DISP:LIM:SEGM1 #2224GHz,5dB,-5dB,SLIN,2,6

14. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
:SEGMent<n>:COLor　　LIMC

● 機能　　指定セグメントのライン色を設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　DISPlay[:WINDow[<chno>]]:LIMit[<parano>]
:SEGMent<n>:COLor<int>

パラメータ　　<int>

応答形式　　NR1 （整数値）

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　LIMC<int>

パラメータ　　<int>

応答形式　　NR1 （整数値）

● 説明　　指定セグメントのリミット・ラインの表示色を設定します。

パラメータ	表示色
1	灰色
2	赤
3	紫
4	緑
5	水色
6	黄
7	白

IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞでは、セグメント番号<n> を0 ～30の範囲で指定します。IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞでは、セグメント番号をあらかじめLSEGコマンドで指定します。

15. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]
:SEGMent<n>:DELete

● 機能　指定セグメントの内容を削除

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>:DELete

● 説明　指定セグメントを削除し、次のセグメントを前につめます。
前セグメントを削除するには、DISP:LIM:CLEarを使用します。

16. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
:SEGMent<n>:LOWer　LIML

● 機能　指定セグメントの下限値を設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>
:LOWer<real>

パラメータ　<real>

応答形式　NR3 （実数値）

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　LIML<real>
＊ [7.19.25 LSEG] も参照して下さい。

パラメータ　<real>

応答形式　NR3 （実数値）

● 説明　指定セグメントのリミット判定下限値を設定します。
指定した下限値が上限値よりも大きい場合、下限値は上限値と同じになります。
IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞでは、セグメント番号<n> を0 ～30の範囲で指定します。IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞでは、セグメント番号をあらかじめLSEGコマンドで指定します。

17. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>:LOWer:REPort?

● 機能　　指定セグメントの下限値でFAILとなったポイント情報を報告

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

クエリ　　DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>:LOWer
:REPort?

応答形式　　<block>

データ・フォーマット設定(FORMat[:DATA]) によって出力形式がが異なります。

アスキー・フォーマット(FORMat[:DATA] ASCii) の場合
<block> = <point>[,<point>,....]
<point> = <stimulus>,<amplitude>,<failed> (ASCII文字列)

バイナリ・フォーマット(FORMat[:DATA] {REAL|MBIN},{32|64}) の場合
<block> = #<byte><length>[<point>...]
<byte> = ブロック長を示す文字列の文字数をアスキー数字(1文字で指示します。
<length> = データ長を示す文字列の文字数をアスキー数字で指示します。
<point> = <stimulus>,<amplitude>,<failed> (ﾊﾞｲﾅﾘ･ﾌｫｰﾏｯﾄ)
<stimulus> = FAIL 点のスティミュラス値 <real>
<amplitude> = FAIL 点のレスポンス値 <real>
<failed> = レスポンス値と下限値との差 <real>

● 説明　　指定セグメントの下限値でFAILとなったポイントについての情報を報告します。
出力データ・フォーマットは、FORM[:DATA] コマンドでの指定に従います。
スティミュラス値およびレスポンス値の単位系は、その時点での表示フォーマットに対応します。

ブロック・データ<block> については、[3.1.2 データ・フォーマット] を参照して下さい。
データ・フォーマットについては、[7.7.2 FORMat[:DATA]] を参照して下さい。

18. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>:REPort?

● 機能　　指定セグメントのFAILとなったポイント情報を報告

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
クエリ　　DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>:REPort?

応答形式　　<block>
データ・フォーマット設定(FORMat[:DATA]) によって出力形式が が異なります。

アスキー・フォーマット(FORMat[:DATA] ASCii) の場合
<block> = <point>[,<point>,....]
<point> = <stimulus>,<amplitude>,<failed> (ASCII文字列)

バイナリ・フォーマット(FORMat[:DATA] {REAL|MBIN},{32|64}) の場合
<block> = #<byte><length>[<point>...]
<byte> = ブロック長を示す文字列の文字数をアスキー数字(1文字で指示します。
<length> = データ長を示す文字列の文字数をアスキー数字で指示します。
<point> = <stimulus>,<amplitude>,<failed> (ﾊﾞｲﾅﾘ･ﾌｫｰﾏｯﾄ)
<stimulus> = FAIL 点のスティミュラス値 <real>
<amplitude> = FAIL 点のレスポンス値 <real>
<failed> = レスポンス値と下限値との差 <real>

● 説明　　指定セグメントのFAILとなったポイントについての情報を報告します。
<failed>の値が正ならば上限値でのFAIL、負ならば下限値でのFAILです。
出力データ・フォーマットは、FORM[:DATA] コマンドでの指定に従います。
スティミュラス値およびレスポンス値の単位系は、その時点での表示フォーマットに対応します。

ブロック・データ<block> については、[3.1.2 データ・フォーマット] を参照して下さい。
データ・フォーマットについては、[7.7.2 FORMat[:DATA]] を参照して下さい。

19. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
:SEGMent<n>:STIMulus　LSTIM

● 機能　　指定セグメントのスティミュラス値の設定

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　DISPlay[:WINDow[<chno>]]:LIMit[<parano>]
:SEGMent<n>:STIMulus<real>

パラメータ　　<real>

応答形式　　NR3　(実数値)

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　LSTIM<real>
＊[7.19.25　LSEG] も参照して下さい。

パラメータ　　<real>

応答形式　　NR3　(実数値)

● 説明　　指定セグメントのスティミュラス値を設定します。
IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞでは、セグメント番号<n> を0 ～30の範囲で指定します。IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞでは、セグメント番号をあらかじめLSEGコマンドで指定します。

20. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
:SEGMent<n>:TYPE　　LIMTFLT|LIMTSLP|LIMTSP

●機能　　指定セグメントのライン・タイプの設定

●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

●IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　DISPlay[:WINDow[<chno>]]:LIMit[<parano>]
:SEGMent<n>:TYPE<type>

パラメータ　　<type> = SLINe|FLINe|SPOint

応答形式　　SLIN|FLIN|SPO

●IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　LIMTFLT|LIMTSLP|LIMTSP

応答形式　　0 | 1

●説明　　指定セグメントのライン・タイプを設定します。

R3762/63 コマンド	R3764/66、R3765/67 ｺﾏﾝﾄﾞ･ﾊﾟﾗﾒｰﾀ	種類
LIMTFLT	FLINe	フラット・ライン
LIMTSLP	SLINe	スロープ・ライン
LIMTSP	SPOint	シングル・ライン

極座標の表示フォーマットでシングル・ポイント以外が選択されると、セグメント内の測定ポイントには、すべて同じリミット値が適応されます。
IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞでは、セグメント番号<n> を0 ～30の範囲で指定します。IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞでは、セグメント番号をあらかじめLSEGコマンドで指定します。

21. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
:SEGMent<n>:UPPer　　LIMU

●機能　　指定セグメントの上限値を設定

●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

●IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　DISPlay[:WINDow[<chno>]]:LIMit[<parano>]
:SEGMent<n>:UPPer<real>

パラメータ　　<real>

応答形式　　NR3　(実数値)

●IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　LSTIM<real>
* [7.19.25　LSEG] も参照して下さい。

パラメータ　　<real>

応答形式　　NR3　(実数値)

●説明　　指定セグメントのリミット判定上限値を設定します。
指定した上限値が下限値よりも小さい場合、上限値は下限値と同じになります。
IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞでは、セグメント番号<n> を0 ～30の範囲で指定します。IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞでは、セグメント番号をあらかじめLSEGコマンドで指定します。

22. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>:UPPer:REPort?

● 機能　　指定セグメントの上限値でFAILとなったポイント情報を報告

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

クエリ　　DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>
:UPPer:REPort?

応答形式　　<block>
データ・フォーマット設定(FORMat[:DATA]) によって出力形式が
が異なります。

アスキー・フォーマット(FORMat[:DATA] ASCii) の場合
<block> = <point>[,<point>,....]
<point> = <stimulus>,<amplitude>,<failed> (ASCII文字列)

バイナリ・フォーマット(FORMat[:DATA] {REAL|MBIN},{32|64})
の場合
<block> = #<byte><length>[<point>...]
<byte> = ブロック長を示す文字列の文字数をアスキー数字(1文
字で指示します。
<length> = データ長を示す文字列の文字数をアスキー数字で指
示します。
<point> = <stimulus>,<amplitude>,<failed> (ﾊﾞｲﾅﾘ･ﾌｫｰﾏｯﾄ)
<stimulus> = FAIL 点のスティミュラス値 <real>
<amplitude> = FAIL 点のレスポンス値 <real>
<failed> = レスポンス値と上限値との差 <real>

● 説明　　指定セグメントの上限値でFAILとなったポイントについての情報
を報告します。
出力データ・フォーマットは、FORM[:DATA] コマンドでの指定に
従います。
スティミュラス値およびレスポンス値の単位系は、その時点での
表示フォーマットに対応します。

ブロック・データ<block> については、[3.1.2 データ・フォー
マット] を参照して下さい。
データ・フォーマットについては、[7.7.2 FORMat[:DATA]] を
参照して下さい。

23. DISPlay[:WINDow[<chno>]]:LIMit[<parano>]　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
:SEGMent<n>:WCOLor　　LIMWC

●機能　　指定セグメント内での波形の色を設定

●ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　DISPlay[:WINDow[<chno>]]:LIMit[<parano>]
:SEGMent<n>:WCOLor<int>

パラメータ　　<int>

応答形式　　NR1　(整数値)

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
コマンド　　LIMWC<int>
＊[7.19.25　LSEG] も参照して下さい。

パラメータ　　<int>

応答形式　　NR1　(整数値)

●説明　　指定セグメント内での測定波形の表示色を設定します。
該当セグメントのスティミュラス範囲のうち、判定がPASSの範囲では、ここで指定した色で測定波形が描画されます。また、判定がFAILとなっている範囲では、ここでの設定に関わらず赤で描画されます。

パラメータ	表示色
1	灰色
2	赤
3	紫
4	緑
5	水色
6	黄
7	白

IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞでは、セグメント番号<n> を0 ～30の範囲で指定します。IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞでは、セグメント番号をあらかじめLSEGコマンドで指定します。

24. DISPlay[:WINDow[<chno>]]:LIMit[<parano>][:STATe]　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ LIMITEST

● 機能　リミット・テスト機能のON/OFF

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.2-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　DISPlay[:WINDow[<chno>]]:LIMit[<parano>][:STATe] <bool>

　パラメータ　<bool>

　応答形式　0 | 1

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
　コマンド　LIMITEST<bool>

　パラメータ　<bool>

　応答形式　0 | 1

● 説明　リミット・テストの機能がONされると、設定されたリミット値を使用してトレース・データの判定を行います。
リミット・ラインを画面に表示するためには、DISP:LIM:LINE をONにする必要があります。
パラメータ<parano>の指定は無効です。

25.

IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ
LSEG

● 機能　　セグメント番号を選択する。

● ｺﾏﾝﾄﾞと ｸｴﾘ の 存在　Command/Query

● IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞ

コマンド　　LSEG<int>

パラメータ　　<int> = 0 ～30

応答形式　　NR1 （整数値）

● 説明　　IEEE488.1-1987ｺﾏﾝﾄﾞ･ﾓｰﾄﾞにおいて、セグメント番号を指定します。セグメント関連の設定コマンドLIMC, LIML, LSTIM, LIMIT, LIMU, LIMWC では、ここで指定されたセグメント番号を対象に設定が行われます。
IEEE488.2-1987コマンド・モードでは、各コマンド中のヘッダ・パラメータ<n> でのセグメントに従うため、このコマンドでの設定は無効です。

付録

A1.　コマンド一覧

A1.1　共通コマンド

```
*CLS

*DDT <blk>
*DMC <str>,<blk>

*EMC <num>
*ESE <num>
*ESR?

*GMC? <name>

*IDN?

*LMC?

*OPC

*PCB <primary>[,<secondary>]
*PMC

*RCL {<num> | POFF}
*RST

*SAV <num>
*SRE <num>
*STB?

*TRG
*TST?

*WAI
```

A1.2　R3764/66, R3765/67コマンド

```
ABORt

CALCulate[<chno>]:FORMat {MLOGarithmic | PHASe | GDELay | POLar | MLINear | SWR | REAL
                          | IMAGinary | UPHase | SCHart | ISCHart | MLIPhase
                          | MLOPhase | MLODeley}
CALCulate[<chno>]:GDAPerture:APERture <real>
CALCulate[<chno>]:MATH[:EXPRession]:NAME {NONE | DDM | DMM | DAM | DSM}
CALCulate[<chno>]:SMOothing:APERture <real>
CALCulate[<chno>]:SMOothing:STATe <bool>
CALCulate[<chno>]:TRANsform:IMPedance:CIMPedance <real>
CALCulate[<chno>]:TRANsform:IMPedance:TYPE {NONE | ZREFlection | YREFlecion | ZTRansmit |
                                            | YTRansmit | INVersion}

DISPlay:ACTive <int>
DISPlay:DUAL <bool>
DISPlay:FORMat {ULOWer | FBACk}
DISPlay[:WINDow[<chno>]]:LIMit[pn]:BEEP <bool>
DISPlay[:WINDow[<chno>]]:LIMit[pn]:CLEar
DISPlay[:WINDow[<chno>]]:LIMit[pn]:DATA <block>
DISPlay[:WINDow[<chno>]]:LIMit[pn]:LINE <bool>
DISPlay[:WINDow[<chno>]]:LIMit[pn]:OFFSet:AMPLitude <real>
DISPlay[:WINDow[<chno>]]:LIMit[pn]:OFFSet:AMPLitude <real>
DISPlay[:WINDow[<chno>]]:LIMit[pn]:OFFSet:STIMulus <real>
DISPlay[:WINDow[<chno>]]:LIMit[pn]:OFFSet:STIMulus <real>
DISPlay[:WINDow[<chno>]]:LIMit[pn]:PARameter[:STATe] <bool>
DISPlay[:WINDow[<chno>]]:LIMit[pn]:PARameter:PLIMit {LINear | LOGarithmic}
DISPlay[:WINDow[<chno>]]:LIMit[pn]:PARameter:SLIMit {LINear | LOGarithmic}
DISPlay[:WINDow[<chno>]]:LIMit[pn]:ParallelIO <bool>
DISPlay[:WINDow[<chno>]]:LIMit[pn]:REPort?
DISPlay[:WINDow[<chno>]]:LIMit[pn]:RESult?
DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n> <block>
DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:COLor <int>
DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:DEL
DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:LOWer <real>
DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:LOWer:REPort?
DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:REPort?
DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:STIMulus <real>
DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:TYPE {SLINe | FLINe | SPOint}
DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:UPPer <real>
DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:UPPer:REPort?
DISPlay[:WINDow[<chno>]]:LIMit[pn][:STATe] <bool>
DISPlay[:WINDow[<chno>]]:TEXT[:DATA] {<str> | <block>}
DISPlay[:WINDow[<chno>]]:TRACe:ASSign {DATA | MEMory | DMEMory}
DISPlay[:WINDow[<chno>]]:TRACe:GRATiclue[:STATe] <bool>
DISPlay[:WINDow[<chno>]]:Y[trace]:RLINe <bool>
DISPlay[:WINDow[<chno>]]:Y[trace][:SCALe]:AUTO ONCE
DISPlay[:WINDow[<chno>]]:Y[trace][:SCALe]:PDIVision <real>
DISPlay[:WINDow[<chno>]]:Y[trace][:SCALe]:RLEVel <real>
DISPlay[:WINDow[<chno>]]:Y[trace][:SCALe]:RPOSition <real>
```

```
FETCh[<chno>][:MARKer]:FANalysis?
FETCh[<chno>][:MARKer]:NUMBer<n>?
FETCh[<chno>][:MARKer][:ACTivate]?
FILE:DBLete <str>
FILE:LOAD <str>
FILE:STATe:CONDition <bool>
FILE:STATe:CORRection <bool>
FILE:STATe:DATA <bool>
FILE:STATe:MEMory <bool>
FILE:STATe:RAW <bool>
FILE:STORe <str>
FORMat:BORDer {NORMal | SWAPped}
FORMat[:DATA] {ASCii | REAL,32 | REAL,64 | MBINary,32 | MBINary,64}

INITiate:CONTinuous <bool>
INITiate[:IMMediate]

MARKer[<chno>]:ACTivate:STATe <bool>
MARKer[<chno>]:ACTivate:STIMulus <real>
MARKer[<chno>]:ACTivate[:NUMBer] <n>[,<real>]
MARKer[<chno>]:AOFF
MARKer[<chno>]:COMPensate <bool>
MARKer[<chno>]:CONVert[:MODE] {DEFault | LINear | RIMaginary}
MARKer[<chno>]:COUPle <bool>
MARKer[<chno>]:DELTa:COMPare <n>[,<real>]
MARKer[<chno>]:DELTa[:MODE] {OFF | CHIld | COMPare | FIXed}
MARKer[<chno>]:FANalysis:DIRection {IN | OUT}
MARKer[<chno>]:FANalysis:WIDTh <real>
MARKer[<chno>]:FANalysis[:STATe] <bool>
MARKer[<chno>]:FIXed:AVALue <real>
MARKer[<chno>]:FIXed:STIMulus <real>
MARKer[<chno>]:FIXed:VALue <real>
MARKer[<chno>]:LBT {STARt | STOP | CENTer | SPAN | RLEVel | FIXed}
MARKer[<chno>]:LIST <bool>
MARKer[<chno>]:POLar {MLINear | MLOGarithmic | RIMaginary}
MARKer[<chno>]:SEARch:PARTial:SRANge
MARKer[<chno>]:SEARch:PARTial[:STATe] <bool>
MARKer[<chno>]:SEARch:RIPPle:DX <real>
MARKer[<chno>]:SEARch:RIPPle:DY <real>
MARKer[<chno>]:SEARch:RIPPle[:MODE] {MAX | MIN | BOTH | PPEak}
MARKer[<chno>]:SEARch:TARGet:LEFT
MARKer[<chno>]:SEARch:TARGet:RIGHt
MARKer[<chno>]:SEARch:TARGet:VALue <real>
MARKer[<chno>]:SEARch:TARGet[:MODE] {ZERO | PI | VALue}
MARKer[<chno>]:SEARch:TRACking <bool>
MARKer[<chno>]:SEARch[:MODE] {OFF | MAX | MIN | TARGet | RIPPle}
MARKer[<chno>]:SMITH {MLINear | MLOGarithmic | RIMaginary | IMPedance | ADMittance}
```

```
REGister:CLEar <int>
REGister:RECall {<int> | POFF}
REGister:SAVE <int>

[SENSe:]AVERage[<chno>]:COUNt <int>
[SENSe:]AVERage[<chno>]:RESTart
[SENSe:]AVERage[<chno>][:STATe] <bool>
[SENSe:]BANDwidth[<chno>][:RESolution] <int>
[SENSe:]BANDwidth[<chno>][:RESolution]:AUTO <bool>
[SENSe:]CORRection[<chno>]:CKIT:TERMinal[port] {MALe | FEMale}
[SENSe:]CORRection[<chno>]:CKIT[:TYPE] {0-4}
[SENSe:]CORRection[<chno>]:COLLect:DELete
[SENSe:]CORRection[<chno>]:COLLect:SAVE
[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] {NORMalize | SNORromalize | OPEN | SHORt
                                             | LOAD | S11Oopen | S11Sshort | S11Load
                                             | S22Oopen | S22Sshort | S22Load | FTRansmit
                                             | FMATch | RTRansmit | RMATch | GTHRU
                                             | OISolation | FISolation | RISolation}
[SENSe:]CORRection[<chno>]:CSET:INTerpolate <bool>
[SENSe:]CORRection[<chno>]:CSET:STATe <bool>
[SENSe:]CORRection[<chno>]:EDELay:DISTance <real>
[SENSe:]CORRection[<chno>]:EDELay:STATe <bool>
[SENSe:]CORRection[<chno>]:EDELay[:TIME] <real>
[SENSe:]CORRection[n]:GPHase:STATe <bool>
[SENSe:]CORRection[<chno>]:OFFSet:PHASe <real>
[SENSe:]CORRection[<chno>]:OFFSet:STATe <bool>
[SENSe:]CORRection[<chno>]:PEXTension:STATe <bool>
[SENSe:]CORRection[<chno>]:PEXTension:TIME[eport] <real>
[SENSe:]CORRection[<chno>]:RVELocity:COAX <real>
[SENSe:]CORRection[n]:GPHase:STATe <bool>
[SENSe:]FUNCtion[<chno>]:POWer {AR | BR | AB | R | A | B | BDC | BDCR | S11 | S21 | S12 | S22
                                  | SF  WD | SREV | NONE}
[SENSe:]FUNCtion[<chno>][:ON] {"POWer: {AC | DC}  {1 | 2 | 3} " | "POWer:RATio: {AC | DC}
                              {2,1 | 3,1 | 2,3} " | "POWer: {S11 | S12 | S21 | S22} "
                              | "POWer: {SFWD | SREV} " | "POWer:NONE"}
[SOURce:]POWer[<chno>]:BANDwidth[n] <int>
[SOURce:]CORRection[n]:GAIN:STATe <bool>
[SOURce:]COUPle <bool>
[SOURce:]FREQuency[<chno>]:CENTer <real>
[SOURce:]FREQuency[<chno>]:CW <real>
[SOURce:]FREQuency[<chno>]:MODE SWEep
[SOURce:]FREQuency[<chno>]:SPAN <real>
[SOURce:]FREQuency[<chno>]:STARt <real>
[SOURce:]FREQuency[<chno>]:STOP <real>
[SOURce:]POWer[<chno>]:STARt <real>
[SOURce:]POWer[<chno>]:STOP <real>
[SOURce:]POWer[<chno>]MODE SWEep
[SOURce:]POWer[<chno>][:LEVel][:AMPlitude] <real>
[SOURce:]PSWeep[<chno>]:CLEar
[SOURce:]PSWeep[<chno>]:CLEar:ALL
```

```
[SOURce:]PSWeep[<chno>]:FREQuency<n> <real>[,<real>]
[SOURce:]PSWeep[<chno>]:MODE {FREQuency | ALL}
[SOURce:]PSWeep[<chno>]:POINts<n> <int>
[SOURce:]PSWeep[<chno>]:POWer<n> <real>
[SOURce:]PSWeep[<chno>]:SETTling<n> <real>
[SOURce:]SWEep[<chno>]:POINts <num>
[SOURce:]SWEep[<chno>]:SPACing {LINear | LOGarithmic}
[SOURce:]SWEep[<chno>]:TIME <real>
[SOURce:]SWEep[<chno>]:TIME:AUTO <bool>
STATus:DEVice:CONDition?
STATus:DEVice:ENABle
STATus:DEVice[:EVENt]?
STATus:FREQuency:CONDition?
STATus:FREQuency:ENABle
STATus:FREQuency[:EVENt]?
STATus:LIMit:CONDition?
STATus:LIMit:ENABle
STATus:LIMit[:EVENt]?
STATus:OPERation:CONDition?
STATus:OPERation:ENABle <num>
STATus:OPERation[:EVENt]?
STATus:POWer:CONDition?
STATus:POWer:ENABle
STATus:POWer[:EVENt]?
STATus:QUEStionable:ENABle
STATus:QUEStionable[:EVENt]?
SYSTem:DATE <year>,<month>,<day>
SYSTem:ERRor?
SYSTem:PRESet
SYSTem:TIME <hour>,<minute>,<second>

TRACe[<chno>]:COPY DATA
TRACe[<chno>][:DATA] {<name> | <trace>} , {<block> | <real>[,<real>...]}
TRACe[<chno>][:DATA]? {<name> | <trace>} [, {<name> | <trace>} ...]
TRIGger[:SEQuence]:DELay <real>
TRIGger[:SEQuence]:DELay:STATe <bool>
TRIGger[:SEQuence]:SIGNal
TRIGger[:SEQuence]:SOURce {IMMediate | EXTernal | BUS | HOLD}
TRIGger[:SEQuence][:IMMediate]
```

A1.3　R3762/63 コマンド

R3762/63コマンド	対応するR3764/66, R3765/67コマンド
AB	[SENSe:]FUNCtion[<chno>]:POWer AB
ABIN	[SENSe:]FUNCtion[<chno>]:POWer AB
ADDRCONT<int>	*PCB <int>
AIN	[SENSe:]FUNCtion[<chno>]:POWer A
ALTAB	[SOURCe:]COUPle OFF
APERTP<real>	CALCulate[<chno>]:GDAPerture:APERture <real>
AR	[SENSe:]FUNCtion[<chno>]:POWer AR
ARIN	[SENSe:]FUNCtion[<chno>]:POWer AR
AUTO	DISPlay[:WINDow[<chno>]]:Y[trace][:SCALe]:AUTO ONCE
AVER<bool>	[SENSe:]AVERage[<chno>][:STATe] <bool>
AVERAGE	[SENSe:]AVERage[<chno>][:STATe] OFF
AVERFACT<int>	[SENSe:]AVERage[<chno>]:COUNt <int>
AVERREST	[SENSe:]AVERage[<chno>]:RESTart
AVR128	[SENSe:]AVERage[<chno>]:COUNt 128; STATe ON
AVR16	[SENSe:]AVERage[<chno>]:COUNt 16; STATe ON
AVR2	[SENSe:]AVERage[<chno>]:COUNt 2; STATe ON
AVR32	[SENSe:]AVERage[<chno>]:COUNt 32; STATe ON
AVR4	[SENSe:]AVERage[<chno>]:COUNt 4; STATe ON
AVR64	[SENSe:]AVERage[<chno>]:COUNt 64; STATe ON
AVR8	[SENSe:]AVERage[<chno>]:COUNt 8; STATe ON
BDCIN	[SENSe:]FUNCtion[<chno>]:POWer BDC
BDCRIN	[SENSe:]FUNCtion[<chno>]:POWer BDCR
BEEPFAIL<bool>	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:BEEP <bool>
BIN	[SENSe:]FUNCtion[<chno>]:POWer B
BR	[SENSe:]FUNCtion[<chno>]:POWer BR
BRIN	[SENSe:]FUNCtion[<chno>]:POWer BR
CALN	[SENSe:]CORRection[<chno>]:CSET:STATe OFF
CENT<real>	[SOURce:]FREQuncy[<chno>]:CENTer <real>
CENTERF<real>	[SOURce:]FREQuncy[<chno>]:CENTer <real>
CH1	DISPlay:ACTive 1
CH2	DISPlay:ACTive 2
CH3	DISPlay:ACTive 3
CH4	DISPlay:ACTive 4
CHAN1	DISPlay:ACTive 1
CHAN2	DISPlay:ACTive 2
CKIT0	[SENSe:]CORRection[<chno>]:CKIT[:TYPE] 0
CKIT1	[SENSe:]CORRection[<chno>]:CKIT[:TYPE] 1
CKIT2	[SENSe:]CORRection[<chno>]:CKIT[:TYPE] 2
CKIT3	[SENSe:]CORRection[<chno>]:CKIT[:TYPE] 3
CKIT4	[SENSe:]CORRection[<chno>]:CKIT[:TYPE] 4
CKIT5	[SENSe:]CORRection[<chno>]:CKIT[:TYPE] 5
CLEA1	REGister:CLEar 1
CLEA2	REGister:CLEar 2
CLEA3	REGister:CLEar 3
CLEA4	REGister:CLEar 4
CLEA5	REGister:CLEar 5

R3762/63コマンド	対応するR3764/66, R3765/67コマンド
CLEAR	[SENSe:]CORRection[<chno>]:COLLect:DELete
CLES	*CLS
CLRREG1	REGister:CLEar 1
CLRREG10	REGister:CLEar 10
CLRREG2	REGister:CLEar 2
CLRREG3	REGister:CLEar 3
CLRREG4	REGister:CLEar 4
CLRREG5	REGister:CLEar 5
CLRREG6	REGister:CLEar 6
CLRREG7	REGister:CLEar 7
CLRREG8	REGister:CLEar 8
CLRREG9	REGister:CLEar 9
CLS	*CLS
CONT	INITiate:CONTinuous ON
CONV1DS	CALCulate[<chno>]:TRANsform:IMPedance:TYPE INVersion
CONVOFF	CALCulate[<chno>]:TRANsform:IMPedance:TYPE NONE
CONVRY	CALCulate[<chno>]:TRANsform:IMPedance:TYPE YREFlection
CONVRZ	CALCulate[<chno>]:TRANsform:IMPedance:TYPE ZREFlection
CONVTY	CALCulate[<chno>]:TRANsform:IMPedance:TYPE YTRansmit
CONVTZ	CALCulate[<chno>]:TRANsform:IMPedance:TYPE ZTRansmnit
CONVYREF	CALCulate[<chno>]:TRANsform:IMPedance:TYPE YREFlection
CONVYTRA	CALCulate[<chno>]:TRANsform:IMPedance:TYPE YTRansmit
CONVZREF	CALCulate[<chno>]:TRANsform:IMPedance:TYPE ZREFlection
CONVZTRA	CALCulate[<chno>]:TRANsform:IMPedance:TYPE ZTRansmnit
CORARY<bool>	FILE:STATe:CORRection <bool>
CORR<bool>	[SENSe:]CORRection[<chno>]:CSET:STATe <bool>
CORRECT<bool>	[SENSe:]CORRection[<chno>]:CSET:STATe <bool>
COUC<bool>	[SOURCe:]COUPle <bool>
COUPLE<bool>	[SOURCe:]COUPle <bool>
CWFREQ<real>	[SOURce:]FREQuency[<chno>]:CW <real>
DATAARY<bool>	FILE:STATe:DATA <bool>
DATI	TRACe[<chno>]:COPY DATA
DAY<int>	SYSTem:DATE <year>,<month>,<day>
DELA	CALCualte[<chno>]:FORMat GDELay
DELAY	CALCualte[<chno>]:FORMat GDELay
DELO	MARKer[<chno>]:DELTa[:MODE] OFF
DELR1	MARKer[<chno>]:DELTa:COMPare 1
DELR2	MARKer[<chno>]:DELTa:COMPare 2
DELR3	MARKer[<chno>]:DELTa:COMPare 3
DELR4	MARKer[<chno>]:DELTa:COMPare 4
DELRFIXM	MARKer[<chno>]:DELTa[:MODE] FIXed
DISM<bool>	MARKer:LIST <bool>
DISPDATA	DISPlay[:WINDow[<chno>]]:TRACe:ASSign DATA
DISPDATM	DISPlay[:WINDow[<chno>]]:TRACe:ASSign DMEMory
DISPDDM<bool>	CALCulate[<chno>]:MATH[:EXPRession]:NAME {DDM | NONE}
DISPDM	DISPlay[:WINDow[<chno>]]:TRACe:ASSign DMEMory
DISPDMM	CALCulate:MATH[:EXPRession]:NAME DSM
DISPMEM	DISPlay[:WINDow[<chno>]]:TRACe:ASSign MEMory

R3762/63コマンド	対応するR3764/66, R3765/67コマンド
DISPMEMO	DISPlay[:WINDow[<chno>]]:TRACe:ASSign MEMory
DIVI	CALCulate[<chno>]:MATH[:EXPRession]:NAME DDM
DL0	(CR+LF/EOI; none)
DL1	(LF; none)
DL2	(EOI; none)
DL3	(CR+LF; none)
DLTX<real>	MARKer[<chno>]:SEARch:RIPPle:DX <real>
DLTY<real>	MARKer[<chno>]:SEARch:RIPPle:DY <real>
DMAXMIN	MARKer[<chno>]:SEARch:RIPPle[:MODE] PPEak
DMKR100[real]	MARKer[<chno>]:DELTa:COMPare 10[,<real>]
DMKR10[real]	MARKer[<chno>]:DELTa:COMPare 1[,<real>]
DMKR20[real]	MARKer[<chno>]:DELTa:COMPare 2[,<real>]
DMKR30[real]	MARKer[<chno>]:DELTa:COMPare 3[,<real>]
DMKR40[real]	MARKer[<chno>]:DELTa:COMPare 4[,<real>]
DMKR50[real]	MARKer[<chno>]:DELTa:COMPare 5[,<real>]
DMKR60[real]	MARKer[<chno>]:DELTa:COMPare 6[,<real>]
DMKR70[real]	MARKer[<chno>]:DELTa:COMPare 7[,<real>]
DMKR80[real]	MARKer[<chno>]:DELTa:COMPare 8[,<real>]
DMKR90[real]	MARKer[<chno>]:DELTa:COMPare 9[,<real>]
DMKRA	MARKer[<chno>]:DELTa[:MODE] COMPare
DMKRC	MARKer[<chno>]:DELTa[:MODE] CHILd
DMKRF	MARKer[<chno>]:DELTa[:MODE] FIXed
DMKROF	MARKer[<chno>]:DELTa[:MODE] OFF
DONE	[SENSe:]CORRection[<chno>]:COLLect:SAVE
DONE	[SENSe:]CORRection[<chno>]:COLLect:SAVE
DONE1PORT	[SENSe:]CORRection[<chno>]:COLLect:SAVE
DONE2PORT	[SENSe:]CORRection[<chno>]:COLLect:SAVE
DONEISO	[SENSe:]CORRection[<chno>]:COLLect:SAVE
DONEREFL	[SENSe:]CORRection[<chno>]:COLLect:SAVE
DONETRNS	[SENSe:]CORRection[<chno>]:COLLect:SAVE
DRIPPL1	MARKer[<chno>]:SEARch[:MODE] RIPPle
DSSTATE<bool>	FILE:STATe:CONDition <bool>
DTOM	TRACe[<chno>]:COPY DATA
DUAC<bool>	DISPlay:DUAL <bool>
DUAL<bool>	DISPlay:DUAL <bool>
ELED<real>	[SENSe:]CORRection[<chno>]:EDELay[:TIME] <real>
ELED<val>	[SENSe:]CORRection[<chno>]:EDELay:DISTance <real>
EPORT1<real>	[SENSe:]CORRection[<chno>]:PEXTension:TIME4 <real>
EPORT2<real>	[SENSe:]CORRection[<chno>]:PEXTension:TIME5 <real>
EPORTA<real>	[SENSe:]CORRection[<chno>]:PEXTension:TIME2 <real>
EPORTB<real>	[SENSe:]CORRection[<chno>]:PEXTension:TIME3 <real>
EPORTR<real>	[SENSe:]CORRection[<chno>]:PEXTension:TIME1 <real>
ESE	*ESE
ESR?	*ESR?
EXTERN	TRIGger[:SEQuence]:SOURce EXTernal
EXTTOFF	TRIGger[:SEQuence]:SOURce IMMediate
EXTTON	TRIGger[:SEQuence]:SOURce EXTernal

R3762/63コマンド	対応するR3764/66, R3765/67コマンド
FAILBEEP<bool>	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:BEEP <bool>
FLTANA<bool>	MARKer[<chno>]:FANnalsis[:STATe] <bool>
FMKRS<real>	MARKer[<chno>]:FIXed:STIMulus <real>
FMKRV<real>	MARKer[<chno>]:FIXed:VALue <real>
FORM0	FORMat:DATA ASCii;BORDer NORMal
FORM2	FORMat:DATA REAL,32;BORDer NORMal
FORM3	FORMat:DATA REAL,64;BORDer NORMal
FORM4	FORMat:DATA ASCii;BORDer NORMal
FORM5	FORMat:DATA REAL,32;BORDer SWAPped
FORM6	FORMat:DATA REAL,64;BORDer SWAPped
FORM7	FORMat:DATA MBINary,32;BORDer NORMal
FORM8	FORMat:DATA MBINary,64;BORDer NORMal
FREE	TRIGger[:SEQuence]:SOURce IMMediate
FRER	INITiate:CONTinuous ON
FWDISO	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] FISolation
FWDMATCH	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] FMATch
FWDTRNS	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] FTRansmit >|
GRAT<bool>	ISPlay[:WINDow[<chno>]]:TRACe:GRATicule[:STATe] <bool>
GRPTHRU	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] GTHRU
HOLD	INITiate:CONTinuous OFF;:ABORt
HOUR<int>	SYSTem:TIME <hour>,<minute>,<second>
IDN?	*IDN?
IDNT	*IDN?
IFBW<int>	[SENSe:]BANDwidth[:RESolution] <int>
IMAG	CALCulate[<chno>]:FORMat IMAGinary
IN1CORDI	TRACe[<chno>][:DATA] {EDIRectivity | 134} , {<block> | <real>[,<real>...]}
IN1CORDI	TRACe[<chno>][:DATA] {EDIRrectivity | 134} , {<block> | <real>[,<real>...]}
IN1CORED	TRACe[<chno>][:DATA] {DATA | 129} , {<block> | <real>[,<real>...]}
IN1CORNR	TRACe[<chno>][:DOTA] {NORMalize | 133} , {<block> | <real>[,<real>...]}
IN1CORNR	TRACe[<chno>][:DATA] {NORMalize | 133} , {<block> | <rea」 [,<real>...]}
IN1CORSO	TRACe[<chno>][:DATA] {ESMatch | 135} , {<block> | <real>[,<real>...]}
IN1CORSO	TRACe[<chno>][:DATA] {ESMatch | 135} , {<block> | <real>[,<real>...]}
IN1CORTR	TRACe[<chno>][:DATA] {ERTRacking | 136} , {<block> | <real>[,<0eal>...]}
IN1CORTR	TRACe[<chno>][:DATA] {ERTRacking | 136} , {<block> | <real>[,<real>...]}
IN1DFOR	TRACe[<chno>][:DATA] {FDATa1 | 0} , {<block> | <real>[,<real>...]}

R3762/63コマンド	対応するR3764/66, R3765/67コマンド
IN1DRAT	TRACe[<chno>][:DATA] {RAW | 131} , {<block> | <real> [,<real>...]}
IN1MFOR	TRACe[<chno>][:DATA] {FMEMory1 | 2} , {<block> | <real> [,<real>...]}
IN1MRAT	TRACe[<chno>][:DATA] {MEMory | 130} , {<block> | <real> [,<real>...]}
IN1NORED	TRACe[<chno>][:DATA] {UDATa | 128} , {<block> | <real> [,<real>...]}
IN2CORDI	TRACe[<chno>][:DATA] {EDIRectivity | 198} , {<block> | <real>[,<real>...]}
IN2CORDI	TRACe[<chno>][:DATA] {EDIRrectivity | 198} , {<block> | <real>[,<real>...]}
IN2CORED	TRACe[<chno>][:DATA] {DATA | 193} , {<block> | <real> [,<real>...]}
IN2CORNR	TRACe[<chno>][:DATA] {NORMalize | 197} , {<block> | <real>[,<real>...]}
IN2CORNR	TRACe[<chno>][:DATA] {NORMalize | 197} , {<block> | <real>[,<real>...]}
IN2CORSO	TRACe[<chno>][:DATA] {ESMatch | 199} , {<block> | <real> [,<real>...]}
IN2CORSO	TRACe[<chno>][:DATA] {ESMatch | 199} , {<block> | <real> [,<real>...]}
IN2CORTR	TRACe[<chno>][:DATA] {ERTRacking | 200} , {<block> | <real>[,<real>...]}
IN2CORTR	TRACe[<chno>][:DATA] {ERTRacking | 200} , {<block> | <real>[,<real>...]}
IN2DFOR	TRACe[<chno>][:DATA] {FDATa1 | 1} , {<block> | <real> [,<real>...]}
IN2DRAT	TRACe[<chno>][:DATA] {RAW | 195} , {<block> | <real> [,<real>...]}
IN2MFOR	TRACe[<chno>][:DATA] {FMEMory1 | 3} , {<block> | <real> [,<real>...]}
IN2MRAT	TRACe[<chno>][:DATA] {MEMory | 194} , {<block> | <real> [,<real>...]}
IN2NORED	TRACe[<chno>][:DATA] {UDATa | 192} , {<block> | <real> [,<real>...]}
IN3CORDI	TRACe[<chno>][:DATA] {EDIRectivity | 262} , {<block> | <real>[,<real>...]}
IN3CORED	TRACe[<chno>][:DATA] {DATA | 257} , {<block> | <real> [,<real>...]}
IN3CORNR	TRACe[<chno>][:DATA] {NORMalize | 261} , {<block> | <real>[,<real>...]}
IN3CORSO	TRACe[<chno>][:DATA] {ESMatch | 263} , {<block> | <real>[,<real>...]}
IN3CORTR	TRACe[<chno>][:DATA] {ERTRacking | 264} , {<block> | <real>[,<real>...]}
IN3DFOR	TRACe[<chno>][:DATA] {FDATa1 | 4} , {<block> | <real> [,<real>...]}
IN3DRAT	TRACe[<chno>][:DATA] {RAW | 259} , {<block> | <real> [,<real>...]}

R3762/63コマンド	対応するR3764/66, R3765/67コマンド
IN3MFOR	TRACe[<chno>][:DATA] {FMEMory1 | 6} , {<block> | <real>[,<real>...]}
IN3MRAT	TRACe[<chno>][:DATA] {MEMory | 258} , {<block> | <real>[,<real>...]}
IN3NORED	TRACe[<chno>][:DATA] {UDATa | 256} , {<block> | <real>[,<real>...]}
IN4CORDI	TRACe[<chno>][:DATA] {EDIRectivity | 326} , {<block> | <real>[,<real>...]}
IN4CORED	TRACe[<chno>][:DATA] {DATA | 321} , {<block> | <real>[,<real>...]}
IN4CORNR	TRACe[<chno>][:DATA] {NORMalize | 325} , {<block> | <real>[,<real>...]}
IN4CORSO	TRACe[<chno>][:DATA] {ESMatch | 327} , {<block> | <real>[,<real>...]}
IN4CORTR	TRACe[<chno>][:DATA] {ERTRacking | 328} , {<block> | <real>[,<real>...]}
IN4DFOR	TRACe[<chno>][:DATA] {FDATa1 | 5} , {<block> | <real>[,<real>...]}
IN4DRAT	TRACe[<chno>][:DATA] {RAW | 323} , {<block> | <real>[,<real>...]}
IN4MFOR	TRACe[<chno>][:DATA] {FMEMory1 | 7} , {<block> | <real>[,<real>...]}
IN4MRAT	TRACe[<chno>][:DATA] {MEMory | 322} , {<block> | <real>[,<real>...]}
IN4NORED	TRACe[<chno>][:DATA] {UDATa | 320} , {<block> | <real>[,<real>...]}
INPCOR	[SENSe:]CORRection[n]:GPHase:STATe <bool>
INTERPOL	[SENSe:]CORRection[<chno>]:CSET:INTerpolate <bool>
IP	SYSTem:PRESet
LABEL<str>	DISPlay[:WINDow[<chno>]]:TEXT[:DATA] {<str> | <block>}
LDFILE<str>	FILE:LOAD <str>
LENGTH<bool>	[SENSe:]CORRection[<chno>]:EDELay:STATe <bool>
LENGVAL<real>	[SENSe:]CORRection[<chno>]:EDELay:DISTance <real>
LEVEL	[SOURce:]POWer[<chno>]:MODE SWEep
LIMC<int>	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>:COLor <int>
DLIMIAMPO<real>	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:OFFSet:AMPLitude <real>
LIMILINE<bool>	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:LINE <bool>
LIMISTIO<real>	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:OFFSet:STIMulus <real>
LIMITEST<bool>	DISPlay[:WINDow[<chno>]]:LIMit[<parano>][:STATe] <bool>
LIML<real>	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n>:LOWer <real>
LIMPLIN	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:PARameter:PLIMit LINear
LIMPLOG	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:PARameter:PLIMit LOGarithmic

R3762/63コマンド	対応するR3764/66, R3765/67コマンド
LIMSLIN	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:PARameter :SLIMit LINear
LIMSLOG	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:PARameter :SLIMit LOGarithmic
LIMS<real>	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n> :STIMulus <real>
LIMTFL	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n> :TYPE FLINe
LIMTFLT	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n> :TYPE FLINe
LIMTSL	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n> :TYPE SLINe
LIMTSLP	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n> :TYPE SLINe
LIMTSP	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n> :TYPE SPOint
LIMU<real>	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n> :UPPer <real>
LINFREQ	[SOURce:]FREQuency[<chno>]:MODE SWEep;:[SOURce:]SWEep [<chno>]:SPACing LINear
LINM	CALCulate[<chno>]:FORMat MLINear
LINMAG	CALCulate[<chno>]:FORMat MLINear
LINMP	CALCulate[<chno>]:FORMat MLIPhase
LISFREQ	[SOURce:]PSWeep[<chno>]:MODE FREQuency
LOAD	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] LOAD
LOGFREQ	[SOURce:]FREQuency[<chno>]:MODE SWEep;:[SOURce:]SWEep [<chno>]:SPACing LOGarithmic
LOGM	CALCulate[<chno>]:FORMat MLOGarithmic
LOGMAG	CALCulate[<chno>]:FORMat MLOGarithmic
LOGMD	CALCulate[<chno>]:FORMat MLODelay
LOGMP	CALCulate[<chno>]:FORMat MLOPhase
LSEG	(segment number is specified by <n> in each command)
LSEGCL	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:CLEar
LSTIM<real>	DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMent<n> :STIMulus <real>
M101P	[SOURce:]SWEep[<chno>]:POINts 101
M11P	[SOURce:]SWEep[<chno>]:POINts 11
M1201P	[SOURce:]SWEep[<chno>]:POINts 1201
M201P	[SOURce:]SWEep[<chno>]:POINts 201
M21P	[SOURce:]SWEep[<chno>]:POINts 21
M301P	[SOURce:]SWEep[<chno>]:POINts 301
M3P	[SOURce:]SWEep[<chno>]:POINts 3
M51P	[SOURce:]SWEep[<chno>]:POINts 51
M601P	[SOURce:]SWEep[<chno>]:POINts 601
M6P	[SOURce:]SWEep[<chno>]:POINts 6
MARK1<val>	MARKer[<chno>]:ACTivate[:NUMBer] 1[,<real>]
MARK2<val>	MARKer[<chno>]:ACTivate[:NUMBer] 2[,<real>]
MARK3<val>	MARKer[<chno>]:ACTivate[:NUMBer] 3[,<real>]
MARK4<val>	MARKer[<chno>]:ACTivate[:NUMBer] 4[,<real>]

R3762/63コマンド	対応するR3764/66, R3765/67コマンド
MARKCONT	MARKer[<chno>]:COMPensate OFF
MARKCOUP	MARKer[<chno>]:COUPle ON
MARKCW	MARKer[<chno>]:LET CENTer
MARKDISC	MARKer[<chno>]:COMPensate ON
MARKFAUV<val>	MARKer:FIXed:AVALue <val>
MARKFSTI<val>	MARKer[<chno>]:FIXed:STIMulus <real>
MARKFVAL<val>	MARKer[<chno>]:FIXed:VALue <real>
MARKMAXI	MARKer[<chno>]:SEARch[:MODE] MAX
MARKMINI	MARKer[<chno>]:SEARch[:MODE] MIN
MARKOFF	MARKer[<chno>]:AOFF
MARKREF	MARKer[<chno>]:LET RLEVel
MARKSPAN	MARKer[<chno>]:LET SPAN
MARKSTAR	MARKer[<chno>]:LET STARt
MARKSTOP	MARKer[<chno>]:LET STOP
MARKUNCO	MARKer[<chno>]:COUPle OFF
MARKZERO	MARKer[<chno>]:LET FIXed
MAXSRCH	MARKer[<chno>]:SEARch[:MODE] MAX
MEAS	ABORt;INITiate[:IMMediate]
MEASA	[SENSe:]FUNCtion[<chno>]:POWer A
MEASB	[SENSe:]FUNCtion[<chno>]:POWer B
MEASR	[SENSe:]FUNCtion[<chno>]:POWer R
MEMARY<bool>	FILE:STATe:MEMory <bool>
MINSRCH	MARKer[<chno>]:SEARch[:MODE] MIN
MINU	CALCulate:MATH[:EXPRession]:NAME DSM
MINUTE<int>	SYSTem:TIME <hour>,<minute>,<second>
MKR10A<real>	MARKer[<chno>]:ACTivate[:NUMBer] 10[,<real>]
MKR1A<real>	MARKer[<chno>]:ACTivate[:NUMBer] 1[,<real>]
MKR2A<real>	MARKer[<chno>]:ACTivate[:NUMBer] 2[,<real>]
MKR3A<real>	MARKer[<chno>]:ACTivate[:NUMBer] 3[,<real>]
MKR4A<real>	MARKer[<chno>]:ACTivate[:NUMBer] 4[,<real>]
MKR5A<real>	MARKer[<chno>]:ACTivate[:NUMBer] 5[,<real>]
MKR6A<real>	MARKer[<chno>]:ACTivate[:NUMBer] 6[,<real>]
MKR7A<real>	MARKer[<chno>]:ACTivate[:NUMBer] 7[,<real>]
MKR8A<real>	MARKer[<chno>]:ACTivate[:NUMBer] 8[,<real>]
MKR9A<real>	MARKer[<chno>]:ACTivate[:NUMBer] 9[,<real>]
MKRAOFF	MARKer[<chno>]:AOFF
MKRCENT	MARKer[<chno>]:LET CENTer
MKRCMP	MARKer[<chno>]:COMPensate ON
MKRCOUP	MARKer[<chno>]:COUPle ON
MKRFIX	MARKer[<chno>]:LET FIXed
MKROFF	MARKer[<chno>]:ACTivate:STATe OFF
MKRPART<bool>	MARKer[<chno>]:SEARch:PARTial[:STATe] <bool>
MKRREF	MARKer[<chno>]:LET RLEVel
MKRSPAN	MARKer[<chno>]:LET SPAN
MKRSTAR	MARKer[<chno>]:LET STARt
MKRSTOP	MARKer[<chno>]:LET STOP
MKRTRAC<bool>	MARKer[<chno>]:SEARch:TRACking <bool>
MKRUCMP	MARKer[<chno>]:COMPensate OFF
MKRUCOUP	MARKer[<chno>]:COUPle OFF

R3762/63コマンド	対応するR3764/66, R3765/67コマンド
MKRZ050	CALCulate[<chno>]:TRANsform:IMPedance:CIMPedance 50OHM
MKRZ075	CALCulate[<chno>]:TRANsform:IMPedance:CIMPedance 75OHM
MONTH<int>	SYSTem:DATE <year>,<month>,<day>
NORM<ON>	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] NORMalize
NORMS<ON>	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] SNORmalize
OMITISO	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] OISolation
OPC	*OPC
OPEN	[SENSe:]CORRection[<chno>]:COLLect[:ACQire] OPEN
OT1CORDI	TRACe[<chno>][:DATA]? {EDIRectivity | 134}
OT1CORED	TRACe[<chno>][:DATA]? {DATA | 129}
OT1CORNR	TRACe[<chno>][:DATA]? {NORMalize | 133}
OT1CORSO	TRACe[<chno>][:DATA]? {ESMatch | 135}
OT1CORTR	TRACe[<chno>][:DATA]? {ERTRacking | 136}
OT1DFOR	TRACe[<chno>][:DATA]? {FDATa1 | 0}
OT1DRAT	TRACe[<chno>][:DATA]? {RAW | 131}
OT1MFOR	TRACe[<chno>][:DATA]? {FMEMory1 | 2}
OT1MRAT	TRACe[<chno>][:DATA]? {MEMory | 130}
OT1NORED	TRACe[<chno>][:DATA]? {UDATa | 128}
OT2CORDI	TRACe[<chno>][:DATA]? {EDIRectivity | 198}
OT2CORED	TRACe[<chno>][:DATA]? {DATA | 193}
OT2CORNR	TRACe[<chno>][:DATA]? {NORMalize | 197}
OT2CORSO	TRACe[<chno>][:DATA]? {ESMatch | 199}
OT2CORTR	TRACe[<chno>][:DATA]? {ERTRacking | 200}
OT2DFOR	TRACe[<chno>][:DATA]? {FDATa1 | 1}
OT2DRAT	TRACe[<chno>][:DATA]? {RAW | 195}
OT2MFOR	TRACe[<chno>][:DATA]? {FMEMory1 | 3}
OT2MRAT	TRACe[<chno>][:DATA]? {MEMory | 194}
OT2NORED	TRACe[<chno>][:DATA]? {UDATa | 192}
OT3CORDI	TRACe[<chno>][:DATA]? {EDIRectivity | 262}
OT3CORED	TRACe[<chno>][:DATA]? {DATA | 257}
OT3CORNR	TRACe[<chno>][:DATA]? {NORMalize | 261}
OT3CORSO	TRACe[<chno>][:DATA]? {ESMatch | 263}
OT3CORTR	TRACe[<chno>][:DATA]? {ERTRacking | 264}
OT3DFOR	TRACe[<chno>][:DATA]? {FDATa1 | 4}
OT3DRAT	TRACe[<chno>][:DATA]? {RAW | 259}
OT3MFOR	TRACe[<chno>][:DATA]? {FMEMory1 | 6}
OT3MRAT	TRACe[<chno>][:DATA]? {MEMory | 258}
OT3NORED	TRACe[<chno>][:DATA]? {UDATa | 256}
OT4CORDI	TRACe[<chno>][:DATA]? {EDIRectivity | 326}
OT4CORED	TRACe[<chno>][:DATA]? {DATA | 321}
OT4CORNR	TRACe[<chno>][:DATA]? {NORMalize | 325}
OT4CORSO	TRACe[<chno>][:DATA]? {ESMatch | 327}
OT4CORTR	TRACe[<chno>][:DATA]? {ERTRacking | 328}
OT4DFOR	TRACe[<chno>][:DATA]? {FDATa1 | 5}
OT4DRAT	TRACe[<chno>][:DATA]? {RAW | 323}
OT4MFOR	TRACe[<chno>][:DATA]? {FMEMory1 | 7}

R3762/63コマンド	対応するR3764/66, R3765/67コマンド
OT4MRAT	TRACe[<chno>][:DATA]? {MEMory | 322}
OT4NORED	TRACe[<chno>][:DATA]? {UDATa | 320}
OUTLEV<real>	[SOURce:]POWer[<chno>][:LEVel][:AMPLitude] <real>
PCB<int>	*PCB <int>
PHAO<real>	[SENSe:]CORRection[<chno>]:OFFSet:PHASe <real>
PHAOFS<bool>	[SENSe:]CORRection[<chno>]:OFFSet:STATe <bool>
PHAS	CALCulate[<chno>]:FORMat PHASe
PHASE	CALCulate[<chno>]:FORMat PHASe
PMKRLIN	MARKER[<chno>]:POLar MLINear
PMKRRI	MARKER[<chno>]:POLar RIMaginary
PMKRRLOG	MARKER[<chno>]:POLar MLOGarithmic
POIN<int>	[SOURce:]SWEep[<chno>]:POINts <int>
POLA	CALCulate[<chno>]:FORMat POLar
POLAR	CALCulate[<chno>]:FORMat POLar
POLMLIN	MARKER[<chno>]:POLar MLINear
POLMLOG	MARKER[<chno>]:POLar MLOGarithmic
POLMRI	MARKER[<chno>]:POLar RIMaginary
PORE<bool>	[SENSe:]CORRection[<chno>]:PEXTension:STATe <bool>
PORT1FEM	[SENSe:]CORRection[<chno>]:CKIT:TERMinal1 FEMale
PORT1MAL	[SENSe:]CORRection[<chno>]:CKIT:TERMinal1 MALe
PORT2FEM	[SENSe:]CORRection[<chno>]:CKIT:TERMinal2 FEMale
PORT2MAL	[SENSe:]CORRection[<chno>]:CKIT:TERMinal3 MALe
PORTA<real>	[SENSe:]CORRection[<chno>]:PEXTension:TIME2 <real>
PORTB<real>	[SENSe:]CORRection[<chno>]:PEXTension:TIME3 <real>
POWE<real>	[SOURce:]POWer[<chno>][:LEVel][:AMPLitude] <real>
POWS	[SOURce:]POWer[<chno>]:MODE SWEep
POWTOFF	[SENSe:]POWer:AC:PROTection:CLEar
PRES	SYSTem:PRESet
PURGE<str>	FILE:DELete <str>
RAWARY<bool>	FILE:STATe:RAW <bool>
RBW100HZ	[SENSe:]BANDwidth[:RESolution] 100HZ
RBW10HZ	[SENSe:]BANDwidth[:RESolution] 10HZ
RBW1KHZ	[SENSe:]BANDwidth[:RESolution] 1KHZ
RBW300HZ	[SENSe:]BANDwidth[:RESolution] 300HZ
RBW30HZ	[SENSe:]BANDwidth[:RESolution] 30HZ
RBW<int>	[SENSe:]BANDwidth[:RESolution] <int>
RBWAUTO	[SENSe:]BANDwidth[:RESolution]:AUTO ON
REAL	CALCulate[<chno>]:FORMat REAL
RECA1	REGister:RECall 1
RECA2	REGister:RECall 2
RECA3	REGister:RECall 3
RECA4	REGister:RECall 4
RECA5	REGister:RECall 5
RECLPOFF	REGister:RECall {0 | POFF}

R3762/63コマンド	対応するR3764/66, R3765/67コマンド
RECLREG1	REGister:RECall 1
RECLREG10	REGister:RECall 10
RECLREG2	REGister:RECall 2
RECLREG3	REGister:RECall 3
RECLREG4	REGister:RECall 4
RECLREG5	REGister:RECall 5
RECLREG6	REGister:RECall 6
RECLREG7	REGister:RECall 7
RECLREG8	REGister:RECall 8
RECLREG9	REGister:RECall 9
REFL<bool>	DISPlay[:WINDow[<chno>]]:Y[trace]:RLINe <bool>
REFP<real>	DISPlay[:WINDow[<chno>]]:Y[trace][:SCALe]:RPOSition <real>
REFV<real>	DISPlay[:WINDow[<chno>]]:Y[trace][:SCALe]:RLEVel <real>
REST	ABORt;INITiate[:IMMediate]
REVISO	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] RISolation
REVMATCH	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] RMATch
REVTRNS	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] RTRansmit
RIN	[SENSe:]FUNCtion[<chno>]:POWer R
RST	*RST
RTC30ADJ	SYSTem:TIME <hour>,<minute>,<second>
S11	[SENSe:]FUNCtion[<chno>]:POWer S11
S11LOAD	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] S11Load
S11OPEN	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] S11Open
S11SHORT	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] S11Short
S12	[SENSe:]FUNCtion[<chno>]:POWer S12
S21	[SENSe:]FUNCtion[<chno>]:POWer S21
S22	[SENSe:]FUNCtion[<chno>]:POWer S22
S22LOAD	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] S22Load
S22OPEN	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] S22Open
S22SHORT	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] S22Short
SAVE1	REGister:SAVE 1
SAVE2	REGister:SAVE 2
SAVE3	REGister:SAVE 3
SAVE4	REGister:SAVE 4
SAVE5	REGister:SAVE 5
SAVEREG1	REGister:SAVE 1
SAVEREG10	REGister:SAVE 10
SAVEREG2	REGister:SAVE 2
SAVEREG3	REGister:SAVE 3
SAVEREG4	REGister:SAVE 4
SAVEREG5	REGister:SAVE 5
SAVEREG6	REGister:SAVE 6
SAVEREG7	REGister:SAVE 7
SAVEREG8	REGister:SAVE 8
SAVEREG9	REGister:SAVE 9

R3762/63コマンド	対応するR3764/66, R3765/67コマンド
SCAL<real>	DISPlay[:WINDow[<chno>]]:Y[trace][:SCALe]:PDIVision <real>
SCALF1ST	DISPlay[:WINDow[<chno>]]:Y[trace]...
SCALF2ND	DISPlay[:WINDow[<chno>]]:Y[trace]...
SDIV<real>	DISPlay[:WINDow[<chno>]]:Y[trace][:SCALe]:PDIVision <real>
SEAMAX	MARKer[<chno>]:SEARch[:MODE] MAX
SEAMIN	MARKer[<chno>]:SEARch[:MODE] MIN
SEAOFF	MARKer[<chno>]:SEARch[:MODE] OFF
SETLTIME<real>	TRIGger[:SEQuence]:DELay <real>
SETLVARI<bool>	TRIGger[:SEQuence]:DELay:STATe <bool>
SETZ	CALCulate[<chno>]:TRANsform:IMPedance:CIMPedance <real>
SETZ0<real>	CALCulate[<chno>]:TRANsform:IMPedance:CIMPedance <real>
SFWD	[SENSe:]FUNCtion[<chno>]:POWer SFWD
SGJB	CALCulate[<chno>]:FORMat ISCHart
SHORT	[SENSe:]CORRection[<chno>]:COLLect[:ACQuire] SHORt
SING	INITiate:CONTinuous OFF;:ABORt;INITiate
SINGLE	INITiate:CONTinuous OFF;:ABORt;INITiate
SMEAS<bool>	[SENSe:]FUNCtion[<chno>]:POWer <input>
SMIC	CALCulate[<chno>]:FORMat SCHart
SMIMGB	MARKer[<chno>]:SMITh ADMittance
SMIMLIN	MARKer[<chno>]:SMITh MLINear
SMIMLOG	MARKer[<chno>]:SMITh MLOGarithmic
SMIMRI	MARKer[<chno>]:SMITh RIMaginary
SMIMRX	MARKer[<chno>]:SMITh IMPedance
SMKRGB	MARKer[<chno>]:SMITh ADMittance
SMKRLIN	MARKer[<chno>]:SMITh MLINear
SMKRLOG	MARKer[<chno>]:SMITh MLOGarithmic
SMKRRI	MARKer[<chno>]:SMITh RIMaginary
SMKRRX	MARKer[<chno>]:SMITh IMPedance
SMOO<bool>	CALCulate[<chno>]:SMOothing:STATe <bool>
SMOOAPER<real>	CALCulate[<chno>]:SMOothing:APERture <real>
SPAN<real>	[SOURce:]FREQuency[<chno>]:SPAN <real>
SPANF<real>	[SOURce:]FREQuency[<chno>]:SPAN <real>
SPLD<bool>	DISPlay:FORMat {ULOWer | FBACk}
SPLEVEL<real>	[SOURce:]POWer[<chno>]:STOP <real>
SPLIT<bool>	DISPlay:FORMat {ULOWer | FBACk}
SRCCOR	[SOURce:]CORRection[n]:GAIN:STATe <bool>
SRCHOFF	MARKer[<chno>]:SEARch[:MODE] OFF
SRE	*SRE
SREV	[SENSe:]FUNCtion[<chno>]:POWer SREV
SRJX	CALCulate[<chno>]:FORMat SCHart
SRQD	(none)
SRQE	(none)
STAR<real>	[SOURce:] {FREQuency | POWer} [<chno>]:STARt <real>
STARTF<real>	[SOURce:]FREQuency[<chno>]:STARt <real>
STB?	*STB?

R3762/63コマンド	対応するR3764/66, R3765/67コマンド
STFILE<str>	FILE:STORe <str>
STIME<real>	[SOURce:]SWEep[<chno>]:TIME <real>
STIMEAUTO	[SOURce:]SWEep[<chno>]:TIME:AUTO ON
STLEVEL<real>	[SOURce:]POWer[<chno>]:STARt <real>
STOP<real>	[SOURce:] {FREQuency | POWer} [<chno>]:STOP <real>
STOPF<real>	[SOURce:]FREQuency[<chno>]:STOP <real>
SWEA	[SOURce:]SWEep[<chno>]:TIME:AUTO ON
SWET<real>	[SOURce:]SWEep[<chno>]:TIME <real>
SWPHLD	INITiate:CONTinuous OFF;:ABORt
SWR	CALCulate[<chno>]:FORMat SWR
T3DB	MARKer[<chno>]:FANalysis:WIDTh 3DB
T3DEG	MARKer[<chno>]:FANalysis:WIDTh 3DEG
T6ODB	MARKer[<chno>]:FANalysis:WIDTh 6ODB
T6DB	MARKer[<chno>]:FANalysis:WIDTh 6DB
T6DEG	MARKer[<chno>]:FANalysis:WIDTh 6DEG
TIN	MARKer[<chno>]:FANalysis:DIRection IN
TITL<str>	DISPlay[:WINDow[<chno>]]:TEXT[:DATA] <str>
TOUT	MARKer[<chno>]:FANalysis:DIRection OUT
TRACK<bool>	MARKer[<chno>]:SEARch:TRACking <bool>
TST?	*TST?
TXDB<real>	MARKer[<chno>]:FANalysis:WIDTh <real>
TXDEG<real>	MARKer[<chno>]:FANalysis:WIDTh <real>;:MARKer[<chno>] :SEARch[:MODE] TARGet
UFREQ<real>	[SOURce:]PSWeep[<chno>]:FREQuency[n] <real>
ULEVEL<real>	[SOURce:]PSWeep[<chno>]:POWer[n] <real>
UNWARP	CALCulate[<chno>]:FORMat UPHase
UPOINT<int>	[SOURce:]PSWeep[<chno>]:POINts[n] <int>
URBW<int>	[SOURce:]PSWeep[<chno>]:BANDwidth[n] <int>
USEG<int>	[SOURce:]PSWeep[<chno>]:FREQuency[n] <real>[,<real>]
USEGCL	[SOURce:]PSWeep[<chno>]:CLEar[n]:ALL
USETLT<real>	[SOURce:]PSWeep[<chno>]:SETTling[n] <real>
USPLEV	[SOURce:]PSWeep[<chno>]:POWer[n] <real>[,<real>]
USRASWP	[SOURce:]PSWeep[<chno>]:MODE ALL
USRFSWP	[SOURce:]PSWeep[<chno>]:MODE FREQuency
USRSWP	[SOURce:]PSWeep[<chno>]:MODE FREQuency
USTART<real>	[SOURce:]PSWeep[<chno>]:FREQuency[n] <real>[,<real>]
USTLEV	[SOURce:]PSWeep[<chno>]:POWer[n] <real>[,<real>]
USTOP<real>	[SOURce:]PSWeep[<chno>]:FREQuency[n] <real>[,<real>]
VELOFACT<real>	[SENSe:]CORRection[<chno>]:RVELocity:COAX <real>
WAIT	*WAI
WIDT<bool>	MARKer[<chno>]:FANnalsis[:STATe] <bool>
WIDV<real>	MARKer[<chno>]:FANalysis:WIDTh <real>
YEAR<int>	SYSTem:DATE <year>,<month>,<day>

R3762/63コマンド	対応するR3764/66, R3765/67コマンド
ZRPSRCH	MARKer[<chno>]:SEARch:TARGet[:MODE] ZERO
ZYMKDFLT	MARKer[<chno>]:CONVert[:MODE] DEFault
ZYMKLIN	MARKer[<chno>]:CONVert[:MODE] LINear
ZYMKRI	MARKer[<chno>]:CONVert[:MODE] RIMaginary

MEMO

A2.　パネル・キー／ソフト・キーに対応するGPIBコマンド一覧

パネル・キー/ ソフト・キーに対応するGPIBコマンドを示します。

- 以下の項目順に記載します。

  1. ACTIVE CHANNELブロック
  2. STIMULUSブロック
  3. RESPONSEブロック
  4. INSTRUMENT STATEブロック
  5. GPIBブロック

- 説明の「O 」と「N 」について

  O : IEEE488.1-1987コマンド・モード
  N : IEEE488.2-1987コマンド・モード

A2. 1　ACTIVE CHANNELブロック

(1)　CH1

CH1	O : CH1 N : DISPlay:ACTive { 1 | 3 }

(2)　CH2

CH2	O : CH2 N : DISPlay:ACTive { 2 | 4 }

A2. 2　STIMULUSブロック

(1)　MENU
信号源メニュー

POWER	パワー・メニューへ (1-1)
SWEEP TIME	O : STIME <real> STIMEAUTO N : [SOURce:]SWEep[<chno>]:TIME <real> [SOURce:]SWEep[<chno>]:TIME:AUTO <bool>
SWEEP TYPE [　　]	掃引タイプ・メニューへ (1-3)
TRIGGER [　　]	トリガ・メニューへ (1-2)
POINTS	O : M {1201 | 601 | 301 | 201 | 101 | 51 | 21 | 11 | 6 | 3}P / POIN<int> POIN<int> N : [SOURce:]SWEep[<chno>]:POINts <int>
COUPLED CH ON/OFF	O : COUPLE <bool> N : [SOURce:]COUPle <bool>
CW FREQ	O : CWFREQ <real> N : [SOURce:]FREQuency[<chno>]:CW <real>
RESTART	O : MEAS N : ABORt;INITiate[:IMMediate]

(1-1) パワー・メニュー

POWER	O : OUTLEV <real> N : [SOURce:]POWer[<chno>][:LEVel][:AMPLitude] <real>
Return	信号源メニューへ　(1)

(1-2) トリガ・メニュー

CONTINUOUS	O : CONT N : INITiate:CONTinuous ON
SINGLE	O : SINGLE N : INITiate:CONTinuous OFF;:ABORt;INITiate
HOLD	O : SWPHLD N : INITiate:CONTinuous OFF;:ABORt
INT TRIG	O : FREE N : TRIGger[:SEQuence]:SOURce IMMediate
EXT TRIG	O : EXTERN N : TRIGger[:SEQuence]:SOURce EXTernal
TRIGGER DELAY	O : SETLTIME <real> N : TRIGger[:SEQuence]:DELay <real>
Return	信号源メニューへ　(1)

(1-3) 掃引タイプ・メニュー

LIN FREQ	O : LINFREQ N : [SOURce:]FREQuency[<chno>]:MODE SWEep; [SOURce:]SWEep[<chno>]:SPACing LINear
LOG FREQ	O : LOGFREQ N : [SOURce:]FREQuency[<chno>]:MODE SWEep; [SOURce:]SWEep[<chno>]:SPACing LOGarithmic
USER SWEEP	O : USRFSWP N : [SOURce:]PSWeep[<chno>]:MODE FREQuency
PROGRAM SWEEP	O : USRARWP N : [SOURce:]PSWeep[<chno>]:MODE ALL
POW SWEEP	O : LEVEL N : [SOURce:]POWer[<chno>]:MODE SWEep
EDIT USER SWEEP	ユーザ周波数掃引セグメント編集メニューへ (1-3-1)
EDIT PROG SWEEP	プログラム掃引セグメント編集メニューへ (1-3-2)
Return	信号源メニューへ (1)

2行のコマンドを併せて使用して下さい。

(1-3-1) ユーザ周波数掃引セグメント編集メニュー

SEGMENT: NUMBER	O : USEG <n> N : （注）
START	O : USTART<start> N : [SOURce:]PSWeep[<chno>]:FREQuency[<n>] <start>[,<stop>]
STOP	O : USTOP<stop> N : [SOURce:]PSWeep[<chno>]:FREQuency[<n>] <start>[,<stop>]
FREQ	O : UFREQ<real> N : [SOURce:]PSWeep[<chno>]:FREQuency[<n>] <start>
POINT	O : UPOINT <int> N : [SOURce:]PSWeep[<chno>]:POINts[<n>] <int>
CLEAR SEG	O : 該当するGPIBコマンドはありません。 N : [SOURce:]PSWeep[<chno>]:CLEar[<n>]
CLEAR ALL SEG	O : USEGCL N : [SOURce:]PSWeep[<chno>]:CLEar[<n>]:ALL
Return	

<start>,<stop> は共に<real>です。

（注）IEEE488.2-1987コマンド・モードでは、セグメント番号を各GPIBコマンド中のパラメータ<n> によって指定します。

(1-3-2) プログラム掃引セグメント編集メニュー(1/2)

SEGMENT: NUMBER	O : USEG <n> N : (注)
START	O : USTART<start> / UFREQ<real> N : [SOURce:]PSWeep[<chno>]:FREQuency[<n>] <start>[,<stop>]
STOP	O : USTOP<stop> N : [SOURce:]PSWeep[<chno>]:FREQuency[<n>] <start>[,<stop>]
POINT	O : UPOINT <int> N : [SOURce:]PSWeep[<chno>]:POINts[<n>] <int>
CLEAR SEG	O : 該当するGPIBコマンドはありません。 N : [SOURce:]PSWeep[<chno>]:CLEar[<n>]
CLEAR ALL SEG	O : USEGCL N : [SOURce:]PSWeep[<chno>]:CLEar[<n>]:ALL
Return	掃引タイプ・メニューへ (1-3)
More 1/2	プログラム掃引セグメント編集メニュー (2/2)へ

<start>,<stop>は共に<real>です。

(注) IEEE488.2-1987コマンド・モードでは、セグメント番号を各GPIBコマンド中のパラメータ<n> によって指定します。

プログラム掃引セグメント編集メニュー(2/2)

SEGMENT: POWER	O : ULEVEL <real> N : [SOURce:]PSWeep[<chno>]:POWer[<n>] <real>
IF RBW	O : URBW <int> N : [SOURce:]PSWeep[<chno>]:BANDwidth[<n>] <int>
SETTLING TIME	O : USETLT <real> N : [SOURce:]PSWeep[<chno>]:SETTling[<n>] <real>
Return	掃引タイプ・メニューへ (1-3)
More 2/2	プログラム掃引セグメント編集メニューへ (1/2)へ

(2) START

START	O : STARTF <real> STLEVEL <real> N : [SOURce:]FREQuency[<chno>]:STARt <real> [SOURce:]POWer[<chno>]:STARt <real>

(3) STOP

STOP	O : STOPF <real> STLEVEL <real> N : [SOURce:]FREQuency[<chno>]:STOP <real> [SOURce:]POWer[<chno>]:STOP <real>

(4) CENTER

CENTER	O : CENTERF <real> N : [SOURce:]FREQuency[<chno>]:CENTer <real>

(5) SPAN

SPAN	O : SPANF <real> N : [SOURce:]FREQuency[<chno>]:SPAN <real>

A2. 3　RESPONSEブロック

(1) MEAS
メジャー・メニュー

① R3765A/67A+Sパラメータ, R3765C/67Cの場合

S11(A/R) REFL FWD	O : S11 N : [SENSe:]FUNCtion[<chno>][:ON] 'POWer:S11' [SENSe:]FUNCtion[<chno>]:POWer S11
S21(B/R) TRANS FWD	O : S21 N : [SENSe:]FUNCtion[<chno>][:ON] 'POWer:S21' [SENSe:]FUNCtion[<chno>]:POWer S21
S12(A/R) TRANS REV	O : S12 N : [SENSe:]FUNCtion[<chno>][:ON] 'POWer:S12' [SENSe:]FUNCtion[<chno>]:POWer S12
S22(B/R) REFL REV	O : S22 N : [SENSe:]FUNCtion[<chno>][:ON] 'POWer:S22' [SENSe:]FUNCtion[<chno>]:POWer S22
S11&S21 FWD	O : 該当するGPIBコマンドはありません。 N : [SENSe:]FUNCtion[<chno>][:ON] 'POWer:SFWD' [SENSe:]FUNCtion[<chno>]:POWer SFWD
S22&S12 REV	O : 該当するGPIBコマンドはありません。 N : [SENSe:]FUNCtion[<chno>][:ON] 'POWer:SREV' [SENSe:]FUNCtion[<chno>]:POWer SREV
SUB MEAS ON/OFF	
CONVERSION [　　]	パラメータ変換メニューへ (1-1)

② R3765A/67Aの場合

A/R	O : ARIN N : [SENSe:]FUNCtion[<chno>][:ON] 'POWer:AC:RATio 2,1' [SENSe:]FUNCtion[<chno>]:POWer AR
B/R	O : BRIN N : [SENSe:]FUNCtion[<chno>][:ON] 'POWer:AC:RATio 3,1' [SENSe:]FUNCtion[<chno>]:POWer BR
R	O : RIN N : [SENSe:]FUNCtion[<chno>][:ON] 'POWer:AC1' [SENSe:]FUNCtion[<chno>]:POWer R
A	O : AIN N : [SENSe:]FUNCtion[<chno>][:ON] 'POWer:AC2' [SENSe:]FUNCtion[<chno>]:POWer A
B	O : BIN N : [SENSe:]FUNCtion[<chno>][:ON] 'POWer:AC3' [SENSe:]FUNCtion[<chno>]:POWer B
SUB MEAS ON/OFF	
CONVERSION [ ]	パラメータ変換メニューへ (1-1)

③　R3765B/67Bの場合

REFLECTION	0 : S11 N : [SENSe:]FUNCtion[<chno>][:ON] 'POWer:S11' [SENSe:]FUNCtion[<chno>]:POWer S11
TRANS MISSION	0 : S21 N : [SENSe:]FUNCtion[<chno>][:ON] 'POWer:S21' [SENSe:]FUNCtion[<chno>]:POWer S21
TRANS & REFL	0 : 該当するGPIBコマンドはありません。 N : [SENSe:]FUNCtion[<chno>][:ON] 'POWer:SFWD' [SENSe:]FUNCtion[<chno>]:POWer SFWD
SUB MEAS ON/OFF	
CONVERSION [　　]	パラメータ変換メニューへ (1-1)

(1-1) パラメータ変換メニュー

Z(REFL)	O : CONVRZ N : CALCulate[<chno>]:TRANsform:IMPedance:TYPE ZREFlection
Z(TRANS)	O : CONVTZ N : CALCulate[<chno>]:TRANsform:IMPedance:TYPE ZTRansmit
Y(REFL)	O : CONVRY N : CALCulate[<chno>]:TRANsform:IMPedance:TYPE YREFlection
Y(TRANS)	O : CONVTY N : CALCulate[<chno>]:TRANsform:IMPedance:TYPE YTRansmit
1/S	O : CONV1DS N : CALCulate[<chno>]:TRANsform:IMPedance:TYPE INVersion
OFF	O : CONVOFF N : CALCulate[<chno>]:TRANsform:IMPedance:TYPE NONE
Z0 VALUE	O : SETZ0 <real> / MKRZ0 {50 | 75} N : CALCulate[<chno>]:TRANsform:IMPedance:CIMPedance <real>
Return	メジャー・メニューへ (1)

(2) FORMAT
フォーマット・メニュー (1/2)

LOG MAG	O : LOGMAG N : CALCulate[<chno>]:FORMat MLOGarithmic
PHASE	O : PHASE N : CALCulate[<chno>]:FORMat PHASe
DELAY	O : DELAY N : CALCulate[<chno>]:FORMat GDELay
SMITH (R+jX)	O : SRJX N : CALCulate[<chno>]:FORMat SCHart
SMITH (G+jB)	O : SGJB N : CALCulate[<chno>]:FORMat ISCHart
POLAR	O : POLAR N : CALCulate[<chno>]:FORMat POLar
LIN MAG	O : LINMAG N : CALCulate[<chno>]:FORMat MLINear
More 1/2	フォーマット・メニュー (2/2)へ

フォーマット・メニュー(2/2)

SWR	O : SWR N : CALCulate[<chno>]:FORMat SWR
REAL	O : REAL N : CALCulate[<chno>]:FORMat REAL
IMAG	O : IMAG N : CALCulate[<chno>]:FORMat IMAGinary
PHASE -∞, +∞	O : UNWRAP N : CALCulate[<chno>]:FORMat UPHase
LOG MAG & PHASE	O : LOGMP N : CALCulate[<chno>]:FORMat MLOPhase
LOG MAG & DELAY	O : LOGMD N : CALCulate[<chno>]:FORMat MLODelay
LIN MAG & PHASE	O : LINMP N : CALCulate[<chno>]:FORMatM LIPhase
More 2/2	フォーマット・メニュー(1/2) へ

(3) SCALE
スケール・メニュー

AUTO SCALE	O : AUTO N : DISPlay[:WINDow[<chno>]]:Y[<trace>][:SCALe]:AUTO ONCE
/DIV	O : SDIV <real> N : DISPlay[:WINDow[<chno>]]:Y[<trace>][:SCALe]:PDIVision <real>
REF VALUE	O : REFV <real> N : DISPlay[:WINDow[<chno>]]:Y[<trace>][:SCALe]:RLEVel <real>
REF POS	O : REFP <real> N : DISPlay[:WINDow[<chno>]]:Y[<trace>][:SCALe]:RPOSition <real>
REF LINE	O : REFL <bool> N : DISPlay[:WINDow[<chno>]]:Y[<trace>]RLINe <bool>
SCALE FOR 2nd / 1st	O : SCALF {1ST | 2ND} N : （注）

（注）IEEE488.2-1987コマンド・モードでは、各GPIBコマンド中のパラメータ<trace>によってトレースを選択します。

<trace> = 0,1,4,5,8,9,12,13
（ただし、0:CH1 TRACE 1st,
1:CH2 TRACE 1st,
4:CH3 TRACE 1st,
5:CH4 TRACE 1st,
8:CH1 TRACE 2nd,
9:CH2 TRACE 2nd,
12:CH3 TRACE 2nd,
13:CH4 TRACE 2nd)

(4) DISPLAY
ディスプレイ・メニュー (1/2)

DUAL CH ON/OFF	O : DUAL <bool> N : DISPlay:DUAL <bool>
SPLIT CH ON/OFF	O : SPLIT <bool> N : DISPlay:FORMat {ULOWer | FBACk} (注)
DISPLAY DATA	O : DISPDATA N : DISPlay[:WINDow[<chno>]]:TRACe:ASSign DATA
DISPLAY MEMORY	O : DISPMEM N : DISPlay[:WINDow[<chno>]]:TRACe:ASSign MEMory
DISPLAY DATA & MEM	O : DISPDM N : DISPlay[:WINDow[<chno>]]:TRACe:ASSign DMEMory
DEFINE TRACE [ ]	トレース演算メニューへ (4-2)
DATA→ MEMORY	O : DTOM N : TRACe[<chno>]:COPY DATA
More 1/2	ディスプレイ・メニュー (2/2)へ

(注) SPLIT CHの {ULOWer | FBACk} はULOWer: スプリット表示、FBACk:オーバラップ表示です。

ディスプレイ・メニュー (2/2)

GRATICULE ON/OFF	O : GRAT <bool> N : DISPlay[:WINDow[<chno>]]:TRACe:GRATicule[:STATe] <bool>
LABEL	ラベル・メニューへ (4-1)
More 2/2	ディスプレイ・メニュー (1/2)へ

(4-1) ラベル・メニュー

DONE	O : LABEL <str> N : DISPlay[:WINDow[<chno>]]:TEXT[:DATA]　{<str> | <block>}
CURSOR →	該当するGPIBコマンドはありません。
CURSOR ←	該当するGPIBコマンドはありません。
BACKSPACE	該当するGPIBコマンドはありません。
DELETE CHAR	該当するGPIBコマンドはありません。
CLEAR LINE	該当するGPIBコマンドはありません。
CANCEL	ディスプレイ・メニュー (2/2)へ　(4)

(4-2) トレース演算メニュー

DATA/MEM	O : DISPDDM ON N : CALCulate[<chno>]:MATH[:EXPRession]:NAME DDM
DATA-MEM	O : 該当するGPIBコマンドはありません。 N : CALCulate[<chno>]:MATH[:EXPRession]:NAME DSM
DATA*MEM	O : 該当するGPIBコマンドはありません。 N : CALCulate[<chno>]:MATH[:EXPRession]:NAME DMM
DATA+MEM	O : 該当するGPIBコマンドはありません。 N : CALCulate[<chno>]:MATH[:EXPRession]:NAME DAM
OFF	O : DISPDDM OFF N : CALCulate[<chno>]:MATH[:EXPRession]:NAME NONE
Return	ディスプレイ・メニュー (1/2)へ　(4)

(5) AVG
アベレージ・メニュー

AVG STATE ON/OFF	O : AVER <bool> N : [SENSe:]AVERage[<chno>][:STATe] <bool>
AVG COUNT	O : AVERFACT <int>/ AVR{2 | 4 | 8 | 16 | 32 | 64 | 128} N : [SENSe:]AVERage[<chno>]:COUNt <int>
AVG RESTART	O : AVERREST N : [SENSe:]AVERage[<chno>]:RESTart
GROUP DELAY APERTURE	O : APERTP <real> N : CALCulate[<chno>]:GDAPerture:APERture <real>
SMOOTHING ON/OFF	O : SMOO <bool> N : CALCulate[<chno>]:SMOothing:STATe <bool>
SMOOTHING APERTURE	O : SMOOAPER <REAL> N : CALCulate[<chno>]:SMOothing:APERture <real>
IF RBW [　　]	O : RBW <int> / RBW{1K | 300 | 100 | 30 | 10}HZ / RBWAUTO N : [SENSe:]BANDwidth[<chno>][:RESolution] <real> [SENSe:]BANDwidth[<chno>][:RESolution]:AUTO <bool>

(6) CAL
校正メニュー(1/2)

NORMALIZE (THRU)	O : NORM ON N : [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] NORMalize
NORMALIZE (SHORT)	O : NORMS ON N : [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] SNORmalize
CAL MENU	フルキャル選択メニューへ (6-1)
CORRECT ON/OFF	O : CORRECT <bool> N : [SENSe:]CORRection[<chno>]:CSET:STATe <bool>
INTERPOLATE ON/OFF	O : INTERPOL N : [SENSe:]CORRection[<chno>]:CSET:INTerpolate <bool>
Z0 VALUE	O : SETZ0 <real> / MKRZ0{50 | 75} N : CALCulate[<chno>]:TRANsform:IMPedance:CIMPedance <real>
More 1/2	校正メニュー (2/2)へ

校正メニュー (2/2)

ELEC DELAY ON/OFF	O : LENGTH <bool> N : [SENSe:]CORRection[<chno>]:EDELay:STATe <bool>
ELECTRICAL DELAY	O : ELED <real> N : [SENSe:]CORRection[<chno>]:EDELay[:TIME] <real>
ELECTRICAL LENGTH	O : LENGVAL <real> N : [SENSe:]CORRection[<chno>]:EDELay:DISTance <real>
VELOCITY FACTOR	O : VELOFACT <real> N : [SENSe:]CORRection[<chno>]:RVELocity:COAX <real>
PHASE OFFSET VALUE	O : PHAO N : [SENSe:]CORRection[<chno>]:OFFSet:PHASe <real>
PORT EXTENSION	ポート延長メニューへ (6-4)
More 2/2	校正メニュー (1/2)へ

(6-1) フルキャル選択メニュー

1PORT FULL CAL	1 ポート・フルキャル・メニューへ (6-1-1)
2PORT FULL CAL	2 ポート・フルキャル・メニューへ (6-2-1)
CAL KIT [ ]	キャル・キット・メニューへ (6-3-1)
CLEAR CAL DATA	O : CLEAR N : [SENSe:]CORRection[<chno>]:COLLect:DELete
Return	校正メニュー (1/2)へ (6)

(6-1-1) 1ポート・フルキャル・メニュー

OPEN	O : OPEN N : [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] OPEN
SHORT	O : SHORT N : [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] SHORt
LOAD	O : LOAD N : [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] LOAD
DONE 1-PORT	O : DONE / DONE1PORT N : [SENSe:]CORRection[<chno>]:COLLect:SAVE

(6-2-1)　2ポート・フルキャル・メニュー

REFLECT'N	リフレクション・メニューへ (6-2-2)
TRANS- MISSION	トランスミッション・メニューへ (6-2-3)
ISOLATION	アイソレーション・メニューへ (6-2-4)
DONE 2-PORT	O : DONE N : [SENSe:]CORRection[<chno>]:COLLect:SAVE

(6-2-2) リフレクション・メニュー

S11: OPEN	O : S11OPEN N : [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] S11Open
S11: SHORT	O : S11SHORT N : [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] S11Short
S11: LOAD	O : S11LOAD N : [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] S11Load
S22: OPEN	O : S22OPEN N : [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] S22Open
S22: SHORT	O : S22SHORT N : [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] S22Short
S22: LOAD	O : S22LOAD N : [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] S22Load
DONE REFLECT'N	O : DONEREFL N : [SENSe:]CORRection[<chno>]:COLLect:SAVE

(6-2-3) トランスミッション・メニュー

FWD.TRANS THRU	O : FWDTRNS N : [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] FTRansmit
FWD.MATCH THRU	O : FWDMATCH N : [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] FMATch
REV.TRANS THRU	O : REVTRNS N : [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] RTRansmit
REV.MATCH THRU	O : REVMATCH N : [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] RMATch
GROUP THRU	O : 該当するGPIBコマンドはありません。 N : [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] GTHRU
DONE TRANS	O : DONETRNS N : [SENSe:]CORRection[<chno>]:COLLect:SAVE

(6-2-4) アイソレーション・メニュー

OMIT ISOLATION	O : OMITISO N : [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] OISolation
FWD.ISOL'N	O : FWDISO N : [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] FISolation
REV.ISOL'N	O : REVISO N : [SENSe:]CORRection[<chno>]:COLLect[:ACQuire] RISolation
DONE ISOLATION	O : DONEISO N : [SENSe:]CORRection[<chno>]:COLLect:SAVE

(6-3-1) キャル・キット・メニュー

N(50Ω)	O : CKIT1 N : [SENSe:]CORRection[<chno>]:CKIT[:TYPE] 1
N(75Ω)	O : CKIT2 N : [SENSe:]CORRection[<chno>]:CKIT[:TYPE] 2
3.5mm	O : CKIT3 N : [SENSe:]CORRection[<chno>]:CKIT[:TYPE] 3
7mm	O : CKIT4 N : [SENSe:]CORRection[<chno>]:CKIT[:TYPE] 4
DONT CARE	O : CKIT0 N : [SENSe:]CORRection[<chno>]:CKIT[:TYPE] 0
Return	校正メニューへ (6)

(6-3-2) FEMAL/MAL 選択メニュー

PORT 1 FEMAL/MAL	O : PORT1 FEM/PORT1 MAL N : [SENSe:]CORRection[<chno>]:CKIT:TERMinal1 FEMale N : [SENSe:]CORRection[<chno>]:CKIT:TERMinal1 MALe
PORT 2 FEMAL/MAL	O : PORT2 FEM/PORT2 MAL N : [SENSe:]CORRection[<chno>]:CKIT:TERMinal2 FEMale N : [SENSe:]CORRection[<chno>]:CKIT:TERMinal2 MALe
Return	キャル・キット・メニュー(6-3-1) へ

(6-4) ポート延長メニュー

EXTENSION ON/OFF	O : PORE <bool> N : [SENSe:]CORRection[<chno>]:PEXTension:STATe <bool>
EXTENSION INPUT R	O : EPORTR <real> N : [SENSe:]CORRection[<chno>]:PEXTension:TIME1 <real>
EXTENSION INPUT A	O : EPORTA <real> N : [SENSe:]CORRection[<chno>]:PEXTension:TIME2 <real>
EXTENSION INPUT B	O : EPORTB <real> N : [SENSe:]CORRection[<chno>]:PEXTension:TIME3 <real>
EXTENSION PORT 1(注)	O : EPORT1 <real> N : [SENSe:]CORRection[<chno>]:PEXTension:TIME4 <real>
EXTENSION PORT 2(注)	O : EPORT2 <real> N : [SENSe:]CORRection[<chno>]:PEXTension:TIME5 <real>
Return	校正メニュー (2/2)へ

(注) R3765A/67A+Sﾊﾟﾗﾒｰﾀ, R3765C/67C, R3765B/67B の場合に設定できます。

(7)　MKR
　　マーカ・メニュー

ACTIVATE MARKER [　　]	アクティブ・マーカ・メニュー (1/2)へ　(7-1)
MARKER ALL OFF	O : MKRAOFF N : MARKer[<chno>]:AOFF
ΔMODE MENU	デルタ・モード・メニューへ (7-2)
MKR LIST ON/OFF	O : 該当するGPIBコマンドはありません。 N : MARKer[<chno>]:LIST <bool>
MARKER MODE MENU	マーカ・モード・メニューへ (7-3)

マーカ・データの取得には
O : MKR {1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10} A?
N : FETch[<chno>][:MARKer][:ACTivate]?
　　FETch[<chno>][:MARKer]:NUMBer<n>?
が使用できます。

(7-1) アクティブ・マーカ・メニュー (1/2)

MARKER 1	O : MKR1A <real> N : MARKer[<chno>]:ACTivate[:NUMBer] 1[,<real>]
MARKER 2	O : MKR2A <real> N : MARKer[<chno>]:ACTivate[:NUMBer] 2[,<real>]
MARKER 3	O : MKR3A <real> N : MARKer[<chno>]:ACTivate[:NUMBer] 3[,<real>]
MARKER 4	O : MKR4A <real> N : MARKer[<chno>]:ACTivate[:NUMBer] 4[,<real>]
MARKER 5	O : MKR5A <real> N : MARKer[<chno>]:ACTivate[:NUMBer] 5[,<real>]
ACTIVATE MKR OFF	O : MKROFF N : MARKer[<chno>]:ACTivate:STATe <bool>
Return	マーカ・メニューへ (7)
More 1/2	アクティブ・マーカ・メニュー (2/2)へ

アクティブ・マーカ・メニュー (2/2)

MARKER 6	O : MKR6A <real> N : MARKer[<chno>]:ACTivate[:NUMBer] 6[,<real>]
MARKER 7	O : MKR7A <real> N : MARKer[<chno>]:ACTivate[:NUMBer] 7[,<real>]
MARKER 8	O : MKR8A <real> N : MARKer[<chno>]:ACTivate[:NUMBer] 8[,<real>]
MARKER 9	O : MKR9A <real> N : MARKer[<chno>]:ACTivate[:NUMBer] 9[,<real>]
MARKER 10	O : MKR10A <real> N : MARKer[<chno>]:ACTivate[:NUMBer] 10[,<real>]
ACTIVATE MKR OFF	O : MKROFF N : MARKer[<chno>]:ACTivate:STATe <bool>
Return	マーカ・メニューへ (7)
More 2/2	アクティブ・マーカ・メニュー (1/2)へ

(7-2) デルタ・モード・メニュー

ΔMODE OFF	O : DMKROF N : MARKer[<chno>]:DELTa[:MODE] OFF
ΔREF= ΔMKR	O : DMKRC N : MARKer[<chno>]:DELTa[:MODE] CHILd
ΔREF= ACT MKR	ACT MKR メニューへ (7-2-1)　(注) O : DMKRA N : MARKer[<chno>]:DELTa[:MODE] COMPare
ΔREF= FIXED MKR	O : DMKRF N : MARKer[<chno>]:DELTa[:MODE] FIXed
FIXED MKR POSITION	FIXED MKR 設定メニュー (7-2-2)
Return	マーカ・メニューへ (7)

(注) コンペア・マーカを選択してから、デルタ・モードをΔREF=ACT MKR に設定して下さい。(ACT MKR メニューを参照して下さい。)

(7-2-1) ACT MKR メニュー (1/2)

COMPARE MARKER 1	O : DMKR1O <real> N : MARKer[<chno>]:DELTa:COMPare 1[,<real>]
COMPARE MARKER 2	O : DMKR2O <real> N : MARKer[<chno>]:DELTa:COMPare 2[,<real>]
COMPARE MARKER 3	O : DMKR3O <real> N : MARKer[<chno>]:DELTa:COMPare 3[,<real>]
COMPARE MARKER 4	O : DMKR4O <real> N : MARKer[<chno>]:DELTa:COMPare 4[,<real>]
COMPARE MARKER 5	O : DMKR5O <real> N : MARKer[<chno>]:DELTa:COMPare 5[,<real>]
ACTIVATE MARKER [　　]	アクティブ・マーカ・メニュー (1/2)へ (7-1)
Return	デルタ・モード・メニューへ (7-2)
More 1/2	ACT MKR メニュー (2/2)へ

ACT MKR メニュー (2/2)

COMPARE MARKER 6	O : DMKR6O <real> N : MARKer[<chno>]:DELTa:COMPare 6[,<real>]
COMPARE MARKER 7	O : DMKR7O <real> N : MARKer[<chno>]:DELTa:COMPare 7[,<real>]
COMPARE MARKER 8	O : DMKR8O <real> N : MARKer[<chno>]:DELTa:COMPare 8[,<real>]
COMPARE MARKER 9	O : DMKR9O <real> N : MARKer[<chno>]:DELTa:COMPare 9[,<real>]
COMPARE MARKER 10	O : DMKR10O <real> N : MARKer[<chno>]:DELTa:COMPare 10[,<real>]
ACTIVATE MARKER [ ]	アクティブ・マーカ・メニュー (1/2)へ (7-1)
Return	デルタ・モード・メニューへ (7-2)
More 2/2	ACT MKR メニュー (1/2)へ

(7-2-2) FIXED MKR 設定メニュー

FIXED MKR STIMULUS	O : FMKRS <real> N : MARKer[<chno>]:FIXed:STIMulus <real>
FIXED MKR VALUE	O : FMKRV <real> N : MARKer[<chno>]:FIXed:VALue <real>
FIXED MKR AUX VALUE	O : 該当するGPIBコマンドはありません。 N : MARKer[<chno>]:FIXed:AVALue <real>
FIXED MKR→ ACTIVE MKR	O : MKRFIX N : MARKer[<chno>]:LET FIXed
Return	デルタ・モード・メニューへ (7-2)

(7-3) マーカ・モード・メニュー

MKR CMP/UNCMP	O : MKRCMP/ MKRUCMP N : MARKer[<chno>]:COMPensate <bool>
MKR CPL/UNCPL	O : MKRCOUP/ MKRUCOUP N : MARKer[<chno>]:COUPle <bool>
CONVERSION MKR MENU [ ]	コンバージョン・マーカ・メニューへ (7-3-1)
SMITH MKR MENU [ ]	スミスマーカ・メニューへ (7-3-2)
POLAR MKR MENU [ ]	ポーラマーカ・メニューへ (7-3-3)
Return	マーカ・メニューへ (7)

(7-3-1) コンバージョン・マーカ・メニュー

DEFAULT	O : ZYMKDFLT N : MARKer[<chno>]:CONVert[:MODE] DEFault
LIN MKR	O : ZYMKLIN N : MARKer[<chno>]:CONVert[:MODE] LINear
Re/Im	O : ZYMKRI N : MARKer[<chno>]:CONVert[:MODE] RIMaginary
Return	マーカ・モード・メニューへ (7-3)

(7-3-2) スミスマーカ・メニュー

LIN MKR	O : SMKRLIN N : MARKer[<chno>]:SMITh MLINear
LOG MKR	O : SMKRLOG N : MARKer[<chno>]:SMITh MLOGarithmic
Re/Im MKR	O : SMKRRI N : MARKer[<chno>]:SMITh RIMaginary
R+jX MKR	O : SMKRRX N : MARKer[<chno>]:SMITh IMPedance
G+jB MKR	O : SMKRGB N : MARKer[<chno>]:SMITh ADMittance
Z0 VALUE	O : SETZO <real> / MKRZO{50 | 75} N : CALCulate[<chno>]:TRANsform:IMPedance:CIMPedance <real>
Return	マーカ・モード・メニューへ (7-3)

(7-3-3) ポーラマーカ・メニュー

LIN MKR	O : PMKRLIN N : MARKer[<chno>]:POLar MLINear
LOG MKR	O : PMKRLOG N : MARKer[<chno>]:POLar MLOGarithmic
Re/Im MKR	O : PMKRRI N : MARKer[<chno>]:POLar RIMaginary
Z0 VALUE	O : SETZO <real> / MKRZO{50 | 75} N : CALCulate[<chno>]:TRANsform:IMPedance:CIMPedance <real>
Return	マーカ・モード・メニューへ (7-3)

(8)　MKR →
マーカ・サーチ・メニュー

MARKER→ START	O : MKRSTAR N : MARKer[<chno>]:LET STARt
MARKER→ STOP	O : MKRSTOP N : MARKer[<chno>]:LET STOP
MARKER→ CENTER	O : MKRCENT N : MARKer[<chno>]:LET CENTer
MARKER→ SPAN	O : MKRSPAN N : MARKer[<chno>]:LET SPAN
MARKER→ REF.VALUE	O : MKRREF N : MARKer[<chno>]:LET RLEVel
PART SRCH [　　]	部分サーチ・メニューへ (8-1)
MKR SEARCH [　　]	サーチ・メニューへ (8-2)

(8-1) 部分サーチ・メニュー

ΔMODE MENU	デルタ・モード・メニューへ (7-2)
SET RANGE	O : 該当するGPIBコマンドはありません。 N : MARKer[<chno>]:SEARch:PARTial:SRANge
PART SRCH ON/OFF	O : MKRPART <bool> N : MARKer[<chno>]:SEARch:PARTial[:STATe] <bool>
Return	マーカ・サーチ・メニューへ　(8)

(8-2) サーチ・メニュー

MKR SEARCH OFF	O : SRCHOFF N : MARKer[<chno>]:SEARch[:MODE] OFF
MAX	O : MAXSRCH N : MARKer[<chno>]:SEARch[:MODE] MAX
MIN	O : MINSRCH N : MARKer[<chno>]:SEARch[:MODE] MIN
TARGET	ターゲット・メニューへ (8-2-1) O : ZRPSRCH (0° SEARCH) N : MARKer[<chno>]:SEARch[:MODE] TARGet
RIPPLE	リップル・メニューへ (8-2-2) O : DRIPPL1 N : MARKer[<chno>]:SEARch[:MODE] RIPPle
FLTR ANAL	フィルタ解析メニューへ (8-2-3)
TRACKING ON/OFF	O : MKRTRAC <bool> N : MARKer[<chno>]:SEARch:TRACking <bool>
Return	マーカ・サーチ・メニューへ (8)

(8-2-1) ターゲット・メニュー

TARGET VALUE	O : 該当するGPIBコマンドはありません。 N : MARKer[<chno>]:SEARch:TARGet[:MODE] VALue MARKer[<chno>]:SEARch:TARGet:VALue <real>
0°	O : ZRPSRCH N : MARKer[<chno>]:SEARch:TARGet[:MODE] ZERO
±180°	O : 該当するGPIBコマンドはありません。 N : MARKer[<chno>]:SEARch:TARGet[:MODE] PI
LEFT SEARCH	O : 該当するGPIBコマンドはありません。 N : MARKer[<chno>]:SEARch:TARGet:LEFT
RIGHT SEARCH	O : 該当するGPIBコマンドはありません。 N : MARKer[<chno>]:SEARch:TARGet:RIGHt
Return	サーチ・メニューへ (8-2)

(8-2-2) リップル・メニュー

MAX∩	O : 該当するGPIBコマンドはありません。 N : MARKer[<chno>]:SEARch:RIPPle[:MODE] MAX
MIN∪	O : 該当するGPIBコマンドはありません。 N : MARKer[<chno>]:SEARch:RIPPle[:MODE] MIN
ΔMAX∩ -MIN∪	O : DRIPPL1 N : MARKer[<chno>]:SEARch:RIPPle[:MODE] BOTH
MAX-MIN	O : DMAXMIN N : MARKer[<chno>]:SEARch:RIPPle[:MODE] PPEak
ΔX	O : DLTX <real> N : MARKer[<chno>]:SEARch:RIPPle:DX <real>
ΔY	O : DLTY <real> N : MARKer[<chno>]:SEARch:RIPPle:DY <real>
Return	サーチ・メニューへ (8-2)

(8-2-3) フィルタ解析メニュー

WIDTH VALUE	O : T{3 | 6 | 60}DB/ T{3 | 6}DEG/ TXDB <real>/ TXDEG <real> N : MARKer[<chno>]:FANalysis:WIDTh <real>
SEARCH IN/OUT	O : TIN/ TOUT N : MARKer[<chno>]:FANalysis:DIRection {IN | OUT}
FILTER ANAL ON/OFF	O : FLTANA <bool> N : MARKer[<chno>]:FANalysis[:STATe] <bool>
Return	サーチ・メニューへ (8-2)

フィルタ解析のデータは
O : TXDB?/ TXDEG?
N : FETch[<chno>][:MARKer]:FANalysis?
で取得できます。

A2.4　INSTRUMENT STATEブロック

(1) SAVE
セーブ・メニュー

SAVE REGISTER	セーブ・レジスタ・メニュー (1/2)へ (1-1)
CLEAR REGISTER	クリア・レジスタ・メニュー (1/2)へ (1-2)
STORE FILE	ストア・ファイル・メニューへ (1-3)
PURGE FILE	パージ・ファイル・メニューへ (1-4)
FORMAT DISK	該当するGPIBコマンドはありません。

(1-1) セーブ・レジスタ・メニュー (1/2)

SAVE REG-1	O : SAVEREG1 N : *SAV 1/ REGister:SAVE 1
SAVE REG-2	O : SAVEREG2 N : *SAV 2/ REGister:SAVE 2
SAVE REG-3	O : SAVEREG3 N : *SAV 3/ REGister:SAVE 3
SAVE REG-4	O : SAVEREG4 N : *SAV 4/ REGister:SAVE 4
SAVE REG-5	O : SAVEREG5 N : *SAV 5/ REGister:SAVE 5
RENAME REG	該当するGPIBコマンドはありません。
Return	セーブ・メニューへ (1)
More 1/2	セーブ・レジスタ・メニュー (2/2)へ

セーブ・レジスタ・メニュー (1/2)

SAVE REG-6	O : SAVEREG6 N : *SAV 6/ REGister:SAVE 6
SAVE REG-7	O : SAVEREG7 N : *SAV 7/ REGister:SAVE 7
SAVE REG-8	O : SAVEREG8 N : *SAV 8/ REGister:SAVE 8
SAVE REG-9	O : SAVEREG9 N : *SAV 9/ REGister:SAVE 9
SAVE REG-10	O : SAVEREG10 N : *SAV 10/ REGister:SAVE 10
RENAME REG	該当するGPIBコマンドはありません。
Return	セーブ・メニューへ (1)
More 2/2	セーブ・レジスタ・メニュー (2/2)へ

(1-2) クリア・レジスタ・メニュー (1/2)

CLEAR REG-1	O : CLRREG1 N : REGister:CLEar 1
CLEAR REG-2	O : CLRREG2 N : REGister:CLEar 2
CLEAR REG-3	O : CLRREG3 N : REGister:CLEar 3
CLEAR REG-4	O : CLRREG4 N : REGister:CLEar 4
CLEAR REG-5	O : CLRREG5 N : REGister:CLEar 5
Return	セーブ・メニューへ (1)
More 1/2	クリア・レジスタ・メニュー (2/2)へ

クリア・レジスタ・メニュー (2/2)

CLEAR REG-6	O : CLRREG6 N : REGister:CLEar 6
CLEAR REG-7	O : CLRREG7 N : REGister:CLEar 7
CLEAR REG-8	O : CLRREG8 N : REGister:CLEar 8
CLEAR REG-9	O : CLRREG9 N : REGister:CLEar 9
CLEAR REG-10	O : CLRREG10 N : REGister:CLEar 10
Return	セーブ・メニューへ (1)
More 1/2	クリア・レジスタ・メニュー (1/2)へ

(1-3) ストア・ファイル・メニュー

STORE	O : STFILE <str> N : FILE:STORe <str>
ROLL ↑	該当するGPIBコマンドはありません。
ROLL ↓	該当するGPIBコマンドはありません。
DEFINE STORE	ファイル・データ・メニューへ (1-3-1)
EDIT NAME	該当するGPIBコマンドはありません。
NAME ↑	該当するGPIBコマンドはありません。
NAME ↓	該当するGPIBコマンドはありません。
CANCEL	該当するGPIBコマンドはありません。

STOREの<str> はファイル・ネームです。

(1-3-1) ファイル・データ・メニュー

STATE ON/OFF	O : DSSTATE <bool> N : FILE:STATe:CONDition <bool>
RAY ARRAY ON/OFF	O : RAWARY <bool> N : FILE:STATe:RAW <bool>
CORR COEF ON/OFF	O : CORARY <bool> N : FILE:STATe:CORRection <bool>
DATA ARRAY ON/OFF	O : DATAARY <bool> N : FILE:STATe:DATA <bool>
MEM ARRY ON/OFF	O : MEMARY <bool> N : FILE:STATe:MEMory <bool>
Return	セーブ・メニューへ (1)

(1-4) パージ・ファイル・メニュー

PURGE	O : PURGE <str> N : FILE:DELete <str>
CURSOR ↑	該当するGPIBコマンドはありません。
CURSOR ↓	該当するGPIBコマンドはありません。
Return	セーブ・メニューへ (1)

PURGE の<str> はファイル・ネームです。

(2) RECALL
リコール・メニュー (1/2)

RECALL REG-1	O : RECLREG1 N : *RCL 1/ REGister:RECall 1
RECALL REG-2	O : RECLREG2 N : *RCL 2/ REGister:RECall 2
RECALL REG-3	O : RECLREG3 N : *RCL 3/ REGister:RECall 3
RECALL REG-4	O : RECLREG4 N : *RCL 4/ REGister:RECall 4
RECALL REG-5	O : RECLREG5 N : *RCL 5/ REGister:RECall 5
RECALL POWER OFF	O : RECLPOFF N : *RCL POFF/ REGister:RECall POFF
LOAD FILE	O : LDFILE <str> N : FILE:LOAD <str>
More 1/2	リコール・メニュー (2/2)へ

LOAD FILE の<str> はファイル・ネームです。

リコール・メニュー (2/2)

RECALL REG-6	O : RECLREG6 N : *RCL 6/ REGister:RECall 6
RECALL REG-7	O : RECLREG7 N : *RCL 7/ REGister:RECall 7
RECALL REG-8	O : RECLREG8 N : *RCL 8/ REGister:RECall 8
RECALL REG-9	O : RECLREG9 N : *RCL 9/ REGister:RECall 9
RECALL REG-10	O : RECLREG10 N : *RCL 10/ REGister:RECall 10
RECALL POWER OFF	O : RECLPOFF N : *RCL POFF/ REGister:RECall POFF
LOAD FILE	O : LDFILE <str> N : FILE:LOAD <str>
More 2/2	リコール・メニュー (1/2)へ

LOAD FILE の<str> はファイル・ネームです。

(3)　SYSTEM
システム・メニュー

SYSTEM DRIVE	該当するGPIBコマンドはありません。 （注）
SET CLOCK	リアルタイム・クロック・メニューへ (3-1)
LIMIT MENU	リミット・メニューへ (3-2-1)

（注）ドライブ名は、ファイル名の中に記述します。"[ドライブ名:] <ファイル名>"

(3-1) リアルタイム・クロック・メニュー

YEAR	O : YEAR <int> N : SYSTem:DATE <year>,<month>,<day>
MONTH	O : MONTH <int> N : SYSTem:DATE <year>,<month>,<day>
DAY	O : DAY <int> N : SYSTem:DATE <year>,<month>,<day>
HOUR	O : HOUR <int> N : SYSTem:TIME <hour>,<minute>,<second>
MINUTE	O : MINUTE <int> N : SYSTem:TIME <hour>,<minute>,<second>
SECOND	O : SECOND <int> N : SYSTem:TIME <hour>,<minute>,<second>
Return	システム・メニューへ (3)

(3-2-1) リミット・メニュー

LIMIT LINE ON/OFF	O : LIMILINE N : DISPlay[:WINDow[<chno>]]:LIMit[pn]:LINE <bool>
LIMIT TEST ON/OFF	O : LIMITEST N : DISPlay[:WINDow[<chno>]]:LIMit[pn][:STATe] <bool>
BEEP FAIL ON/OFF	O : BEEPFAIL N : DISPlay[:WINDow[<chno>]]:LIMit[pn]:BEEP <bool>
LIMIT MODE MENU	リミット・モード・メニューへ (3-2-2)
EDIT LIMIT LINE	エディット・リミッツ・メニューへ (3-2-4)
SELECT DATA 1ST/2ND	O : 該当するGPIBコマンドはありません。 N : 該当するGPIBコマンドはありません。
LIMIT LINE OFFSETS	オフセット・リミッツ・メニューへ (3-2-8)
Return	システム・メニューへ (3)

(3-2-2) リミット・モード・メニュー

1ST DATA ON/OFF	O : 該当するGPIBコマンドはありません。 N : DISPlay[:WINDow[<chno>]]:LIMit[<pn>]:PARameter[:STATe]
2ND DATA ON/OFF	O : 該当するGPIBコマンドはありません。 N : DISPlay[:WINDow[<chno>]]:LIMit[<pn>]:PARameter[:STATe]
SMITH LIMIT MENU	リミット・パラメータ・メニューへ (3-2-3)
POLAR LIMIT MENU	リミット・パラメータ・メニューへ (3-2-3)
Return	リミット・メニューへ (3-2-1)

(3-2-3) リミット・パラメータ・メニュー

RF/IM LIMIT	O : 該当するGPIBコマンドはありません。　スミス表示 N : DISPlay[:WINDow[<chno>]]:LIMit[<pn>]:PARameter:Smith LIMit {RIMaginaly | RhoTHeta}　ポーラ表示 DISPlay[:WINDow[<chno>]]:LIMit[<pn>]:PARameter:Polar LIMit {RIMaginaly | RhoTHeta}
MAG/PHASE LIMIT	O : 該当するGPIBコマンドはありません。　スミス表示 N : DISPlay[:WINDow[<chno>]]:LIMit[<pn>]:PARameter:Smith LIMit {RIMaginaly | RhoTHeta}　ポーラ表示 DISPlay[:WINDow[<chno>]]:LIMit[<pn>]:PARameter:Polar LIMit {RIMaginaly | RhoTHeta}
Return	リミット・モード・メニューへ (3-2-2)

(3-2-4) エディット・リミッツ・メニュー

SEGMENT	O : LSEG N : 該当するGPIBコマンドはありません。
EDIT SEGMENT	O : 該当するGPIBコマンドはありません。 N : 該当するGPIBコマンドはありません。
DELETE	O : 該当するGPIBコマンドはありません。 N : DISPlay[:WINDow[<chno>]]:LIMit[<pn>]:SEGMent<n>:DELete
ADD SEGMENT	O : 該当するGPIBコマンドはありません。 N : 該当するGPIBコマンドはありません。
CLEAR	クリア・リミット・メニューへ (3-2-6)
LINE TYPE	リミット・タイプ・メニューへ (3-2-7)
DONE	O : 該当するGPIBコマンドはありません。 N : 該当するGPIBコマンドはありません。

(3-2-5) エディット・セグメント・メニュー

STIMULUS VALUE	O : LIMS N : DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:STIMulus <real>
MARKER TO STIMULUS	O : 該当するGPIBコマンドはありません。 N : 該当するGPIBコマンドはありません。
UPPER LIMIT	O : LIMU N : DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:UPPer <real>
LOWER LIMIT	O : LIML N : DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:LOWer <real>
DELTA LIMIT	O : 該当するGPIBコマンドはありません。 N : 該当するGPIBコマンドはありません。
MIDDLE VALUE	O : 該当するGPIBコマンドはありません。 N : 該当するGPIBコマンドはありません。
MARKER TO MIDDLE	O : 該当するGPIBコマンドはありません。 N : 該当するGPIBコマンドはありません。
Return	O : 該当するGPIBコマンドはありません。 N : 該当するGPIBコマンドはありません。

(3-2-6) クリア・リミット・メニュー

YES	O : LSEGCL N : DISPlay[:WINDow[<chno>]]:LIMit[pn]:CLEar
NO	O : 該当するGPIBコマンドはありません。 N : 該当するGPIBコマンドはありません。

(3-2-7) リミット・タイプ・メニュー

SLOPING LINE	O : LIMTSLP N : DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:TYPE Slope LINe
FLAT LINE	O : LIMTFLT N : DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:TYPE Flat LINe
SINGLE POINT	O : LIMTSP N : DISPlay[:WINDow[<chno>]]:LIMit[pn]:SEGMent<n>:TYPE Single Point
LIMIT COLOR	O : LIMC N : LIMC:COLor
WAVE COLOR	O : LIMWC N : LIMWC:WCOLor
Return	エディット・リミッツ・メニューへ (3-2-4)

(3-2-8) オフセット・リミッツ・メニュー

STIMULUS OFFSET	O : 該当するGPIBコマンドはありません。 N : DISPlay[:WINDow[<chno>]]:LIMit[pn]:OFFSet:STIMulus <real>
AMPLITUDE OFFSET	O : 該当するGPIBコマンドはありません。 N : DISPlay[:WINDow[<chno>]]:LIMit[pn]:OFFSet:AMPLitude <real>
MARKER TO AMP.OFS	O : 該当するGPIBコマンドはありません。 N : 該当するGPIBコマンドはありません。
Return	リミット・メニューへ (3-2-1)

(4) PRESET

PRESET	O : IP N : SYSTem:PRESet

A2.5　GPIBブロック

(1) PROGRAM

PROGRAM

このキーで呼び出される以下のメニューに該当するGPIBコマンドはありません。
- コントローラ・メニュー
- ロード・メニュー
- ドライブ・メニュー

(2) REMOTE/LCL
GPIBメニュー

SYSTEM CONTROLLER	該当するGPIBコマンドはありません。
TALKER LISTENER	該当するGPIBコマンドはありません。
SET ADDRESSES	アドレス・メニューへ (2-1)

(2-1) アドレス・メニュー

ADDRESS R3765 (注)	該当するGPIBコマンドはありません。 (注) R3767 の場合R3767 と表示します。
ADDRESS PLOTTER	該当するGPIBコマンドはありません。
ADDRESS PRINTER	該当するGPIBコマンドはありません。
Return	GPIBメニューへ (2)

MEMO

A3.　初期設定

表 A3 - 1　初期設定(1/3)

機能	初期化方法	
	電源投入またはﾌﾟﾘｾｯﾄ	*RST
スティミュラス		
掃引タイプ	リニア周波数掃引	同左
連続掃引	ON	OFF
トリガソース	内部(FREE RUN)	同左
トリガ遅延	OFF(0sec)	同左
掃引時間	190.95msec(AUTO) (R3764/65ｼﾘｰｽﾞ)	240.2msec(AUTO) (R3764/65ｼﾘｰｽﾞ)
	402.0msec(AUTO) (R3766/67ｼﾘｰｽﾞ)	420.35msec(AUTO) (R3766/67ｼﾘｰｽﾞ)
測定ポイント数	201	1201
スタート周波数	40MHz	同左
ストップ周波数	3.8GHz(R3764/65ｼﾘｰｽﾞ)	同左
	8.0GHz(R3766/67ｼﾘｰｽﾞ)	同左
中心周波数	1.92GHz(R3764/65ｼﾘｰｽﾞ)	同左
	4.02Hz (R3766/67ｼﾘｰｽﾞ)	同左
周波数スパン	3.76GHz(R3764/65ｼﾘｰｽﾞ)	同左
	7.96GHz(R3766/67ｼﾘｰｽﾞ)	同左
周波数表示	スタート／ストップ	同左
ﾚﾍﾞﾙ 掃引の固定周波数	1GHz	同左
出力レベル	*1	同左
スタートレベル	*2	同左
ストップレベル	*2	同左
2チャンネル連動	ON	同左
ﾌﾟﾛｸﾞﾗﾑ 掃引ｾｸﾞﾒﾝﾄ	すべてクリア	同左
レスポンス		
デュアルチャンネル	OFF	同左
アクティブチャンネル	CH 1	同左
分解能帯域幅	10kHz	同左
入力ポートの選択条件	*3	同左
アベレージ	OFF(回数16)	同左
トレース演算	NONE	同左
コンバージョン	NONE	同左
特性インピーダンスZ0	50Ω	同左
測定フォーマット	*4	同左
群遅延アパーチャ	10%	0.01%
スムージング	OFF(アパーチャ10%)	OFF(アパーチャ 0.01%)
ディスプレイ	データ	同左
スプリット／重ね表示	重ね表示	同左
ラベル	なし	同左

*1 : 出力レベル

タイプ	電源投入またはプリセット	*RST
A	0dBm	同左
B	0dBm	同左
C A+Sパラメータ	10dBm	同左

*2 : スタート／ストップ・レベル

タイプ	電源投入またはプリセット		*RST	
	スタート	ストップ	スタート	ストップ
A	-13dBm	0dBm	同左	22dBm
B	-15dBm	0dBm	同左	20dBm
C A+Sパラメータ	-20dBm	0dBm	同左	10dBm

*3 : 入力ポートの選択条件

タイプ ＼ チャンネル	CH1	CH2	CH3	CH4
A	A/R	B/R	A/R	B/R
B	REFLECTION	TRANSMISSION	REFLECTION	TRANSMISSION
C A+Sパラメータ	S11	S21	S11	S21

*4 : 測定フォーマット

タイプ ＼ チャンネル	CH1	CH2	CH3	CH4
A	LOGMAG	LOGMAG	LOGMAG	LOGMAG
B	LOGMAG	LOGMAG	POLAR	LOGMAG
C A+Sパラメータ	LOGMAG	LOGMAG	POLAR	LOGMAG

(2/3)

機能	初期化方法	
	電源投入またはﾌﾟﾘｾｯﾄ	*RST
リファレンスの値		
ログ振幅	0dB	同左
位相	0°	同左
群遅延	0sec	同左
スミスチャート	1	同左
極座標	1	同左
リニア振幅	0	同左
SWR	1	同左
実数部	10	同左
虚数部	10	同左
連続位相	0°	同左
Y 軸1 目盛当たりのスケール		
ログ振幅	*5	同左
位相	45°	同左
群遅延	100nsec	同左
スミスチャート	−	同左
極座標	−	同左
リニア振幅	100m	同左
SWR	1	同左
実数部	1	同左
虚数部	1	同左
連続位相	360°	同左
リファレンスの位置		
ログ振幅	*6	同左
位相	50%	同左
群遅延	50%	同左
スミスチャート	−	同左
極座標	−	同左
リニア振幅	0%	同左
SWR	0%	同左
実数部	100%	同左
虚数部	100%	同左
連続位相	50%	同左

*5 : ログ振幅(Y軸1 目盛当たりのスケール)

タイプ＼チャンネル	CH1	CH2	CH3	CH4
A	10dB	10dB	1dB	1dB
B	5dB	10dB	1 UNIT	1dB
C A+Sパラメータ	5dB	10dB	1 UNIT	1dB

*6 : ログ振幅 (リファレンスの位置)

タイプ＼チャンネル	CH1	CH2	CH3	CH4
A	90%	90%	90%	90%
B	90%	90%	—	90%
C A+Sパラメータ	90%	90%	—	90%

(3/3)

機能	初期化方法	
	電源投入またはﾌﾟﾘｾｯﾄ	*RST
校正		
補正測定	OFF	同左
校正データ	クリア	同左
電気長補正	OFF(0sec)	同左
位相オフセット	OFF(0°)	同左
測定端面延長補正	OFF	同左
R 入力	0sec	同左
A 入力	0sec	同左
B 入力	0sec	同左
ポート1	0sec	同左
ポート2	0sec	同左
伝搬定数	1	同左

表 A3 - 2　バックアップメモリの設定（工場出荷時）

項目	初期値
本器のGPIBアドレス	11
ｼｽﾃﾑｺﾝﾄﾛｰﾗ/ｱﾄﾞﾚｯｻﾌﾞﾙ	アドレッサブル
プリンタGPIBアドレス	18
プロッタGPIBアドレス	5
セーブ・レジスタ	すべてクリア

MEMO

A4.　マルチライン・インタフェース・メッセージ

	PCG												SCG			
	ACG		UCG		LAG				TAG							
	0		1		2		3		4		5		6		7	
	ascii	msg	ascii	msg	ascii	msg	ascii	msg	ascii	msg	ascii	msg	ascii	msg	ascii	msg
0	NUL		DEL		SP		0		@		P		‘		p	
1	SOH	GTL	DC1	LLO	!		1		A		Q		a		q	
2	STX		DC2		"		2		B		R		b		r	
3	ETX		DC3		#		3		C		S		c		s	
4	EOT	SDC	DC4	DCL	$		4		D		T		d		t	
5	ENQ	PPC	NAK	PPU	%		5		E		U		e		u	
6	ACK		SYN		&	(1)	6	(1)	F	(2)	V	(2)	f		v	
7	BEL		ETB		'		7		G		W		g		w	
8	BS	GET	CAN	SPE	(		8		H		X		h		x	
9	HT	TCT	EM	SPD	)		9		I		Y		i		y	
10	LF		SUB		*		:		J		Z		j		z	
11	VT		ESC		+		;		K		[		k		{	
12	FF		FS		,		<		L		\		l		|	
13	CR		GS		-		=		M		]		m		}	
14	SO		RS		.		>		N		^		n		~	
15	SI		US		/		?	UNL	O		_	UNT	o		DEL	

（注）　PCG : 1 次コマンド・グループ
ACG : アドレス・コマンド・グループ
UCG : ユニバーサル・コマンド・グループ
LAG : リスン・アドレス・グループ
TAG : トーク・アドレス・グループ
SCG : 2 次コマンド・グループ（意味はPCG によって定義される）
(1) : 機器に割り当てられるリスナ・アドレス
(2) : 機器に割り当てられるトーカ・アドレス

MEMO

索引

— 数字、アルファベット順 —

— 50順 —

【R】

【S】

【T】

【Y】

【Z】

【あ】

【い】

【え】

【お】

【か】

【き】

【く】

【け】

【こ】

【さ】

【し】

【す】

【せ】

【そ】

【た】

【ち】

【て】

【と】

【は】

【ひ】

【ふ】

【へ】

【ほ】

【ま】

【み】

【め】

【も】

【ゆ】

【ら】

【り】

【れ】

【ろ】

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