プログラマ・マニュアル

Tektronix

DG2030 型
データ・ゼネレータ

070-A664-50

www.tektronix.com

日本テクトロニクス株式会社　〒141–0001　東京都品川区北品川 5–9–31

はじめに

このマニュアルでは、GPIB インタフェースまたは RS-232C インタフェースを通して本機器をリモート制御するための方法を説明します。本機器の詳細は、添付のユーザ・マニュアルで説明されていますので、ユーザ・マニュアルも同時にご参照ください。

関連マニュアル

- ユーザ・マニュアル　—　本機器を操作するために必要な情報を提供します。
- サービス・マニュアル　—　本機器の保守およびサービスのための情報を提供します。

目　次

第 1 章　概　要

はじめに

本機器には、GPIB インタフェースと RS-232C インタフェースの 2 つのリモート・インタフェース・ポートが用意されています。いずれかのインタフェースを通し、専用のプログラミング・コマンド・セットを使用して、外部のコントローラや端末から、本機器のメニューやフロント・パネル・コントロールの制御を行うことができます（ただし、エディット機能、GPIB と RS-232C のパラメータ設定機能、および前面パネルの ON/STBY スイッチの機能を除きます）。

GPIB インタフェースは、IEEE Std 488.1-1987 で、ハードウェア・インタフェース、基本プロトコル、およびファンクション・コードが定義されており、計測機器用の標準バスとして広く使用されています。また、IEEE Std 488.2-1987 では、さらに高度なシステム・アプリケーションをサポートするために、コード、フォーマット、共通コマンドを、上位のインタフェース・レイヤに定義しています。図 1-1 に、GPIB のインタフェース・レイヤを示します。

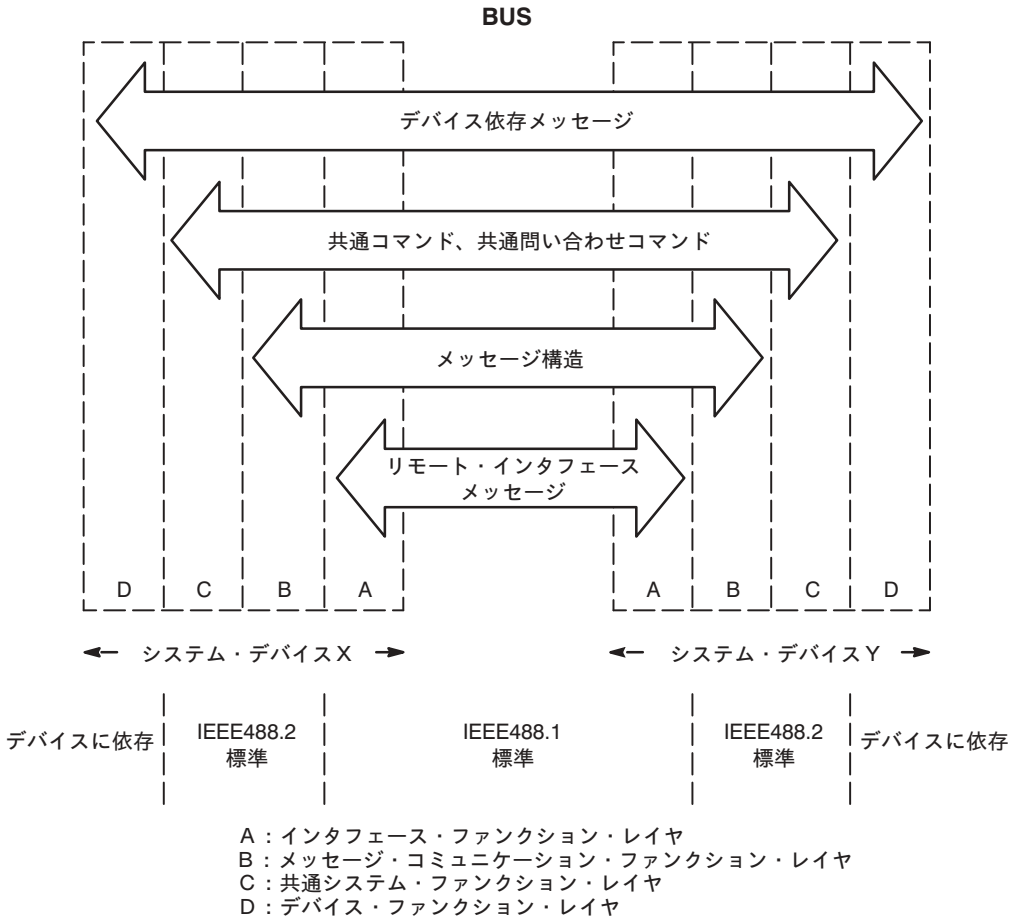


図 1-1: GPIBシステムのインタフェース・レイヤ

一方、RS-232C は、PC 、端末、モデム、プリンタなどの間で通信を行うために広く使用されているインタフェースです。本機器では、このインタフェースの上位レイヤに、IEEE Std 488.2-1987 のコード、フォーマット、共通コマンドを加えているため、GPIB と同様の制御を行うことができます。

このマニュアルでは、これらのインタフェースを使用して本機器を制御するための方法を、次の項目に従って説明します。

- インタフェースの選択
- リモート・コントロールのための準備
- コマンド・シンタックス
- コマンド・セット
- ステータス／イベント・レポーティング・システム

本機器で利用できるプログラミング・コマンド・セットについては、第2章の「コマンド詳細説明」の項をご参照ください。また、本機器は、ステータス／イベント・レポーティング機能を備えています。この機能については、第 3 章の「ステータス／イベント・レポーティング・システム」の項をご参照ください。

なお、本機器で利用可能な GPIB のインタフェース・ファンクションについては、「付録 B　GPIB インタフェース仕様」をご参照ください。

インタフェースの選択

本機器を外部コントローラで制御する場合には、まず初めに、GPIB インタフェースまたは RS-232C インタフェースのうち、どちらのインタフェースを使用するのかを決めなければなりません。GPIB インタフェースは、8 ビット・パラレル・バスを使用しているため高速データ転送が可能で、しかも同時に複数の機器を接続して制御することができます。一方、RS-232C インタフェースは、シリアルにデータを転送するため転送速度が遅く、しかも 1 対 1 の接続しかできないという欠点がありますが、低コストで接続が簡単であるという利点もあります。これらの点を考慮の上、使用するインタフェースを決定します。

表 1-1 に、GPIB インタフェースと RS-232C インタフェースの比較を示します。

表 1-1: GPIB と RS-232C の比較

特性項目	GPIB	RS-232C
接続形態	複数機器の接続（最大 15 台 / システム	1 対 1 の接続
インタラプト・レポート機能	サービス・リクエスト、ステータス&イベント・コード	ステータス&イベント・コード（サービス・リクエスト機能なし）
データ転送バス	8 ビット・パラレル	8 ビット・シリアル
同期	非同期	非同期
転送速度	200 k バイト / 秒	19,200 ビット / 秒
データ・フロー・コントロール	ハードウェア: 3 線式ハンドシェイク	ソフトウェア・フラッギング、ハードウェア・フラッギング、無制御
メッセージ・ターミネーション（受信）	ハードウェア EOI またはソフトウェア LF	ソフトウェア CR，LF 、または CR と LF
メッセージ・ターミネーション（送信）	ハードウェア EOI とソフトウェア LF	ソフトウェア LF
インタフェースの制御	低レベル・コントロール・メッセージ	なし
インタフェース・メッセージ	IEEE-488 Std. に準拠	ASCII 制御コード（ブレーク信号）による DCL（ Device Clear ）
接続ケーブル	IEEE-488 Std.	9 線（ DCE ）
接続ケーブル長	装置間 2 m、1 システムでトータル最大20 mまで	最大 15 mまで

リモート・コントロールのための準備

GPIB インタフェースまたは RS-232C インタフェースを使用する際には、次に説明する項目に従って本機器を設定します。

GPIB インタフェースの設定

接続と接続形態

GPIB インタフェースをリモート・インタフェースとして利用する場合には、本機器のリア・パネルにある IEEE STD 488 ポート（ GPIB コネクタ）に外部コントローラに接続された標準の GPIB ケーブルを接続します。外部コントローラとしては、MS-DOS コンパチブルなパーソナル・コンピュータなどに GPIB インタフェース・ボードをインストールして使用することができます。たとえば、IBM PC または NEC の PC-9800 シリーズに NATIONAL INSTRUMENTS の PC2/PC2A GPIB ボードをインストールして、外部コントローラにすることができます。

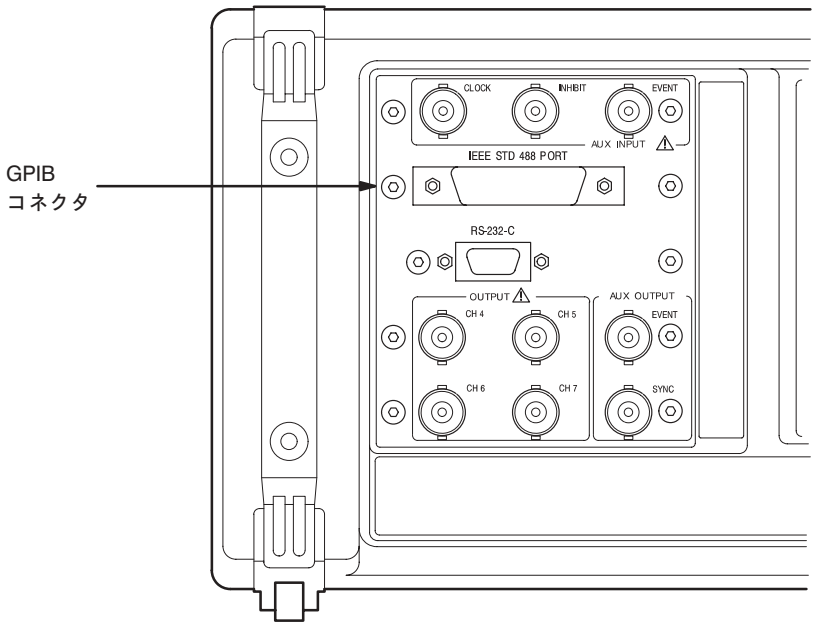


図 1-2: GPIBコネクタ

GPIB インタフェースを利用すると、リニア型、スター型、あるいはリニア型とスター型を組み合わせて機器を接続し、GPIB システムを構築できます。リニア型の接続は、A の機器から B の機器へ、B の機器から C の機器へと順番にケーブルを接続する方法です。またスター型の接続は、1 つの機器から他のすべての機器にケーブルを接続する方法です。図 1-3 に、接続形態を示します。

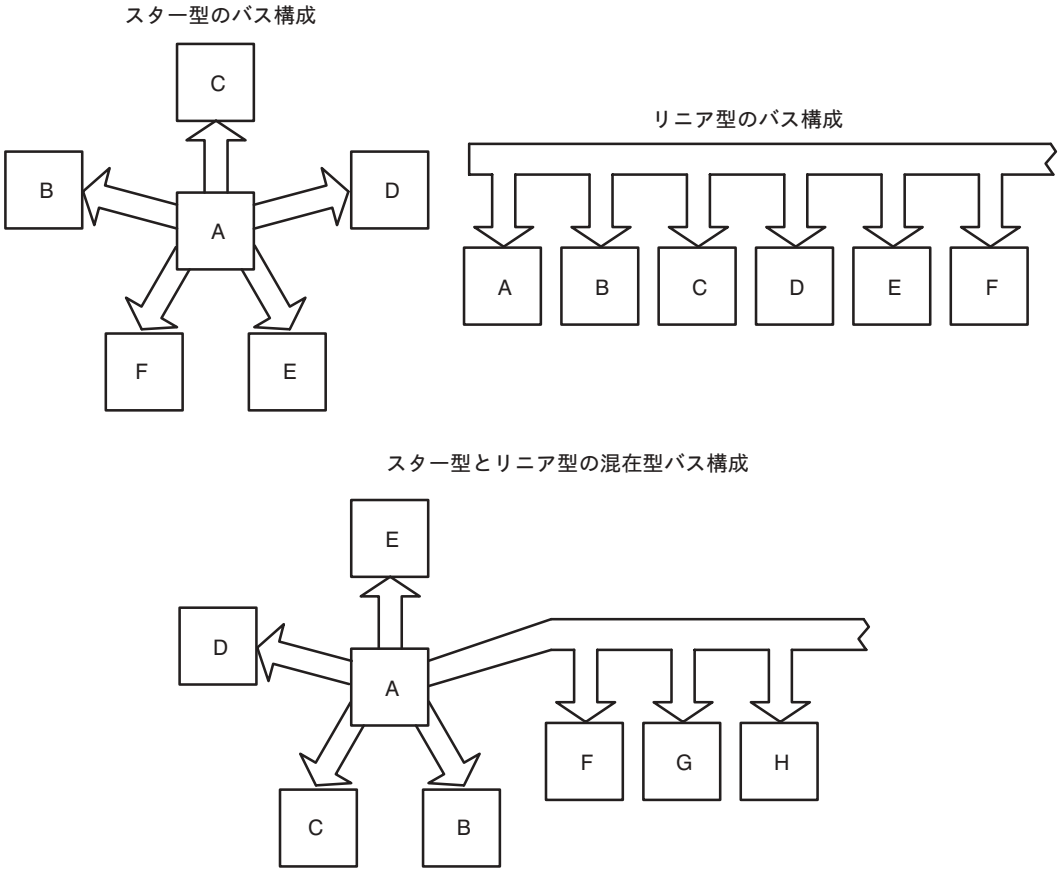


図 1-3: GPIB の接続形態

使用制限

GPIB システムへの接続は、次の規則に従って行ってください。

1. 1 つの GPIB システムの中に接続できる機器の数は、コントローラを含めて、15 台までです。

2. バスの電気的特性を維持するために、装置間は、2 m 以内の長さのケーブルで接続しなければなりません。

3. 1 つの GPIB システム中で、ケーブルのトータル長は、20 m を超えてはなりません。

4. システムが動作中には、少なくとも 2 / 3 の機器が電源 ON の状態になければなりません。

GPIBパラメータの設定

次に、GPIB パラメータの設定手順を示します。

☐ **手順 1**：フロント・パネルの MENU 欄にある UTILITY キーを押します。これにより UTILITY ボトム・メニューが、管面下部、ボトム・キーのすぐ上に表示されます。

☐ **手順 2**：System ボトム・キーを押し、System メニューを表示します（図 1-4 参照）。

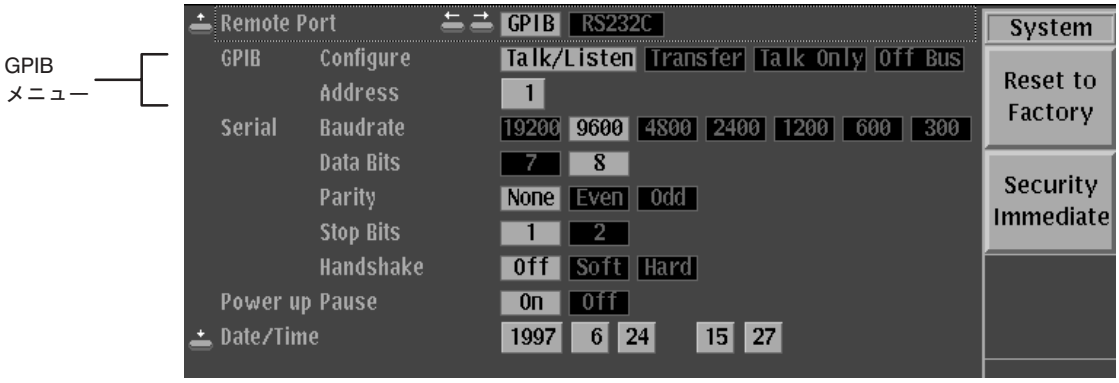


図 1-4: System メニュー

☐ **手順3**:上下矢印キーを使用して、GPIB メニューの Configure 項目を選択します。ここでは、左右矢印キーを使用して GPIB の動作モードを設定します。

Talk / Listen ― コミュニケーション・モードを Talk/Listen に設定します。

Talk Only ― コミュニケーション・モードをハードコピーの出力用の Talk Only に設定します。

Off Bus ― 本機器を論理的に GPIB システムから切り離します。

注：本機器では、ハードウェア EOI またはソフトウェア LF のいずれもターミネータとして認識します。EOI は、データの最終バイトが転送されるときに、EOI ラインを“Low”にして終了信号とする方式で、LF は、データの最終バイトとして ASCII コードの LF を付加する方式です。本機器からデータを転送する場合には、両方の方式を同時に使用します。

☐ **手順 4**：上下矢印キーを使用して、GPIB メニューの Address 項目を選択します。ここでは、ロータリ・ノブを使用してプライマリ・アドレスを 0 ～ 30 の間で設定します。

☐ **手順 5**：上下矢印キーを使用して Remote Port 項目を選択し、さらに左右矢印キーを使用して GPIB を強調表示にします。これにより、リモート・インタフェースとして GPIB インタフェースが選択されます。

RS-232C インタフェースの設定

コネクタとケーブル

RS-232C インタフェースをリモート・インタフェースとして利用する場合には、本機器のリア・パネルにある RS-232C コネクタと外部コントローラを、データ・フロー・コントロールの設定に応じて配線したケーブルで接続してください。

RS-232C インタフェースでは、機器間をポイント・ツー・ポイントで接続します（図 1-5 参照）。このインタフェースでは、GPIB インタフェースにおいてパラレルに転送されるメッセージをシリアルに転送します。

RS-232C インタフェースは、通常、DTE（Data Terminal Equipment）で 25 ピン・オス型 D タイプ・コネクタを、また DCE（Data Communication Equipment）で 25 ピン・メス型 D タイプ・コネクタを使用しますが、最近の機器では、9 ピン D タイプ・コネクタが多く使用されるようになっています。図 1-7 に、9 ピン D タイプ・コネクタと 25 ピン D タイプ・コネクタのピン配置を示します。

本機器は DCE として設計されているので、オス型－メス型コネクタを持つ 15 m までの長さのスルー・ケーブルで、DTE の機器と接続することができます。外部コントローラが DCE の場合には、特別なアダプタか、ヌル・モデム・ケーブルを使用して、DCE-DCE 通信用にケーブルを接続しなければなりません。図 1-8 に、ケーブルの接続例を示します。

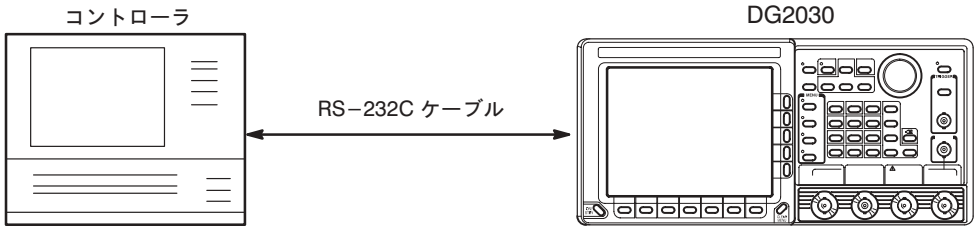


図 1-5: RS-232C ポイント・ツー・ポイント接続

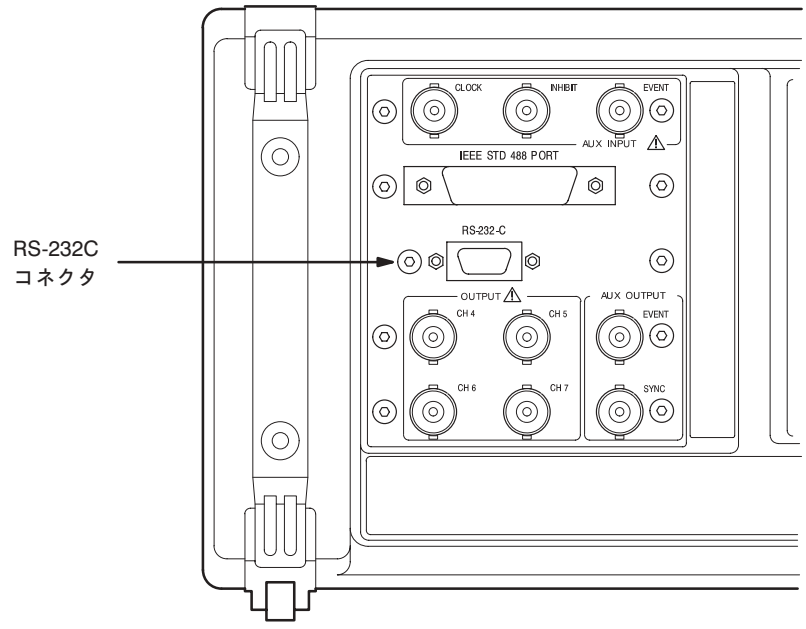


図 1-6: RS-232C コネクタ

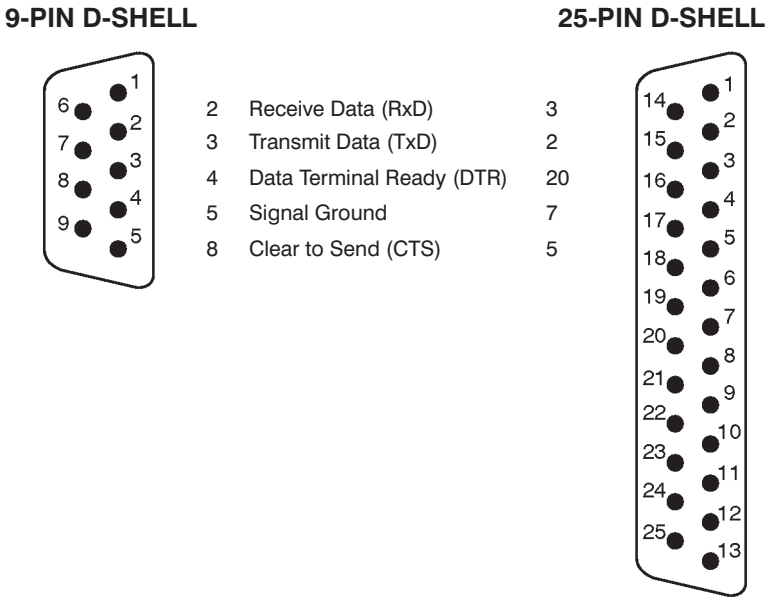
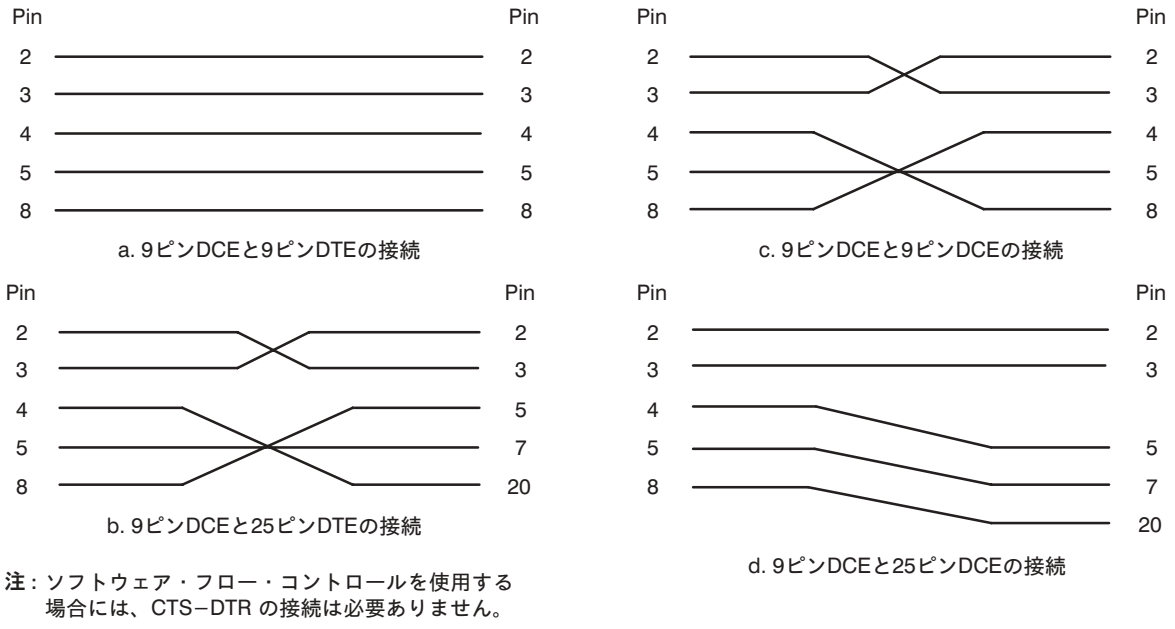


図 1-7: 9 ピン、25 ピン D タイプ・コネクタのピン配置



注 : ソフトウェア・フロー・コントロールを使用する場合には、CTS−DTR の接続は必要ありません。

図 1-8: RS-232C のケーブル接続例

RS-232C パラメータの設定

次に、RS-232C パラメータの設定手順を示します。

☐ **手順 1**：フロント・パネルの MENU 欄にある UTILITY キーを押します。これにより UTILITY ボトム・メニューが、管面下部、ボトム・キーのすぐ上に表示されます。

☐ **手順 2**：System ボトム・キーを押し、System メニューを表示します（図 1-9 参照）。

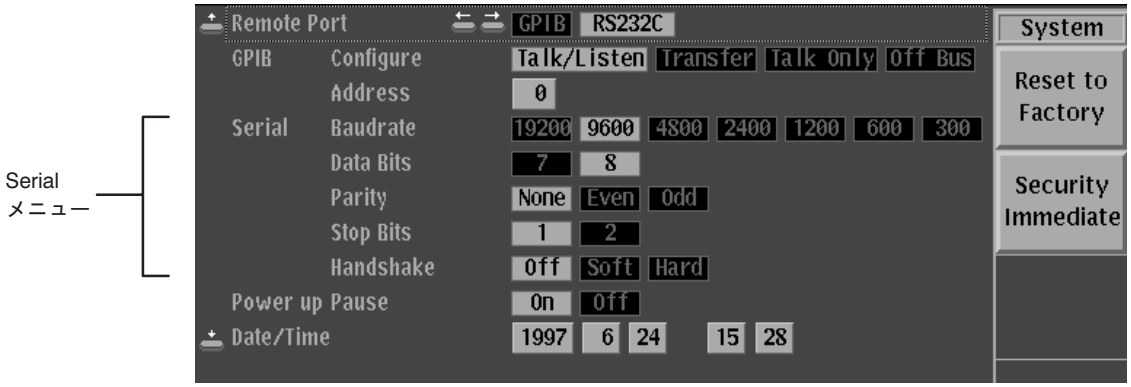


図 1-9: System メニュー

☐ **手順 3**：上下矢印キーを使用して、Serial メニューの Baudrate 項目を選択します。ここでは、左右矢印キーを使用してデータ転送レートを設定します。設定可能なレートは、300、600、1200、2400、4800、9600、19200 ボーです。

☐ **手順 4**：上下矢印キーを使用して、Serial メニューの Data Bits 項目を選択します。ここでは、左右矢印キーを使用して 1 キャラクタのデータ・ビット長を設定します。設定可能なビット長は、7 ビットまたは 8 ビットです。

☐ **手順 5**：上下矢印キーを使用して、Serial メニューの Parity 項目を選択します。ここでは、左右矢印キーを使用してパリティ・ビットを設定します。設定可能なパリティは、None、Even、Odd です。

☐ **手順 6**：上下矢印キーを使用して、Serial メニューの Stop Bits 項目を選択します。ここでは、左右矢印キーを使用してストップ・ビット長を設定します。設定可能なストップ・ビット長は、1 ビットまたは 2 ビットです。

☐ **手順 7**：上下矢印キーを使用して、Serial メニューの Handshake 項目を選択します。ここでは、左右矢印キーを使用してデータ・フロー・コントロールの方法を選択します。本機器では、Hard（DTR/CTS によるハードウェア制御）と Soft（XON/XOFF によるソフトウェア制御）あるいは Off（制御なし）が選択できます。

☐ **手順 8**：上下矢印キーを使用して Remote Port 項目を選択し、さらに左右矢印キーを使用して RS232C を強調表示にします。これによりリモート・インタフェースとして RS–232C インタフェースが選択されます。

第２章　シンタックスとコマンド

コマンド・シンタックス

本機器には、外部コントローラからリモート・コントロールを行うためのコマンド・セットが用意されています。次に、これらのコマンドを使用して本機器を制御するために必要なプログラムの記述方法について説明します。

記　法

ここでは、表 2-1 に示す BNF 記法を使用して説明を行います。

表 2-1: BNF シンボルと意味

シンボル	意味
＜＞	定義項目を表します。
[ ]	オプション（省略可）項目を表します。
[ ]...	オプション（省略可）または繰り返し項目を表します。
{ }	選択可能ないくつかの項目をグループ化します。
|	排他的に選択を行います。
::＝	左辺を右辺として定義します。

プログラム・メッセージとレスポンス・メッセージ

外部コントローラで作成されたプログラムは、リモート・インタフェースを通して、本機器にプログラム・メッセージとして送られます。

プログラム・メッセージは、プログラム・メッセージ・ユニット・デリミタ（;）で区切られた 0 個以上のプログラム・メッセージ・ユニットから構成されます。プログラム・メッセージ・ユニットは、設定コマンド・メッセージまたは問合せコマンド・メッセージを表します。設定コマンド・メッセージは、外部コントローラから本機器に転送される1つの設定コマンドで、機器に対して機能を実行したり、動作条件やモードを設定したりするように指示します。また、問合せコマンド・メッセージは、外部コントローラから本機器に転送される問合せコマンドで、機器に対して、パターン・データまたは設定値やモードなどのステータスを外部コントローラに返送するように指示します。なお、機器から外部コントローラに返送されるデータは、レスポンス・メッセージと呼ばれます。

コマンドと問合せコマンドの構造

コマンドは、設定コマンドと問合せコマンド（このマニュアルでは、それぞれコマンドおよび問合せコマンドと呼びます）からなります。ほとんどのコマンドは、設定の形式と問合せの形式の両方を持ち合わせているため、問合せの形式を、設定形式のヘッダの後ろに“？”を付加して作ることができます。

図 2-1 は、本機器で使用するコマンドおよび問合せコマンドの構造をフロー・チャートで表したものです。ヘッダの構造については、この後の「ヘッダ」の項で詳しく説明します。

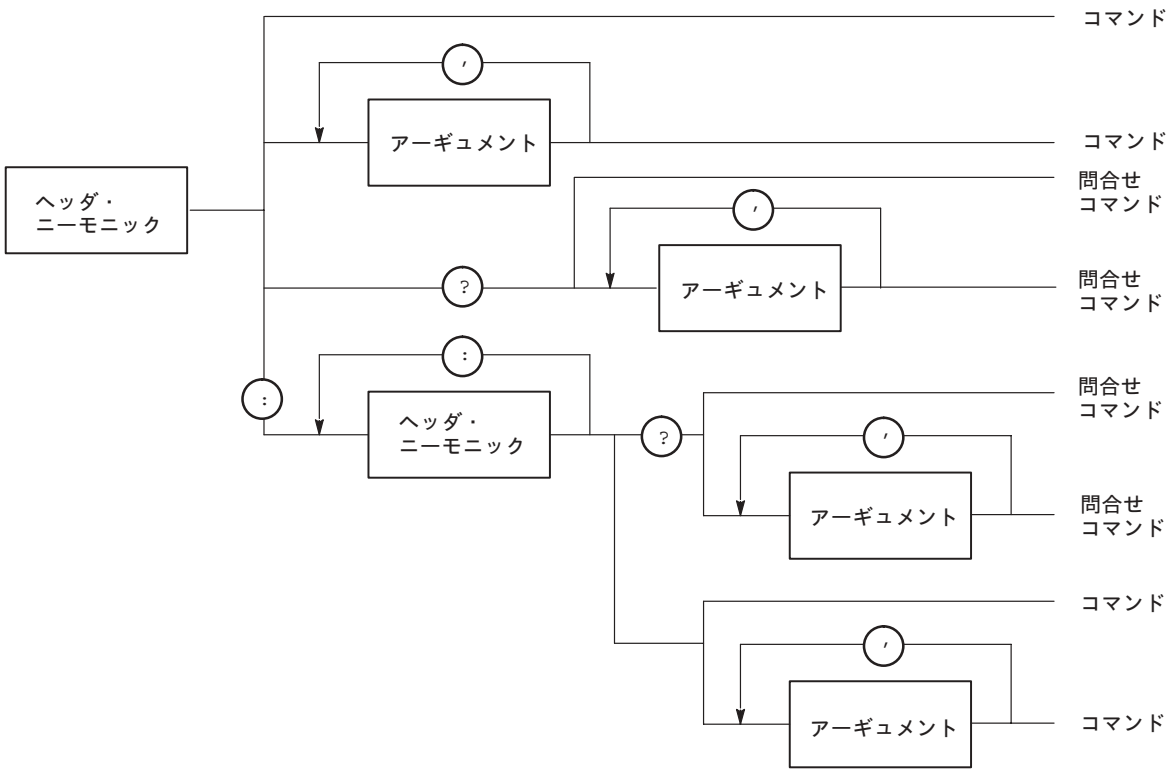


図 2-1: コマンド、問合せコマンド構造のフロー・チャート

ASCIIコード

プログラムは、ASCIIコードを使用して記述することができます。プログラムでは、例外的にアービトラリ・ブロック・データのみ 8 ビット ASCII コードを使用し、それ以外は、すべて 7 ビット ASCII コードを使用します。ASCII キャラクタ・コードについては、「付録 A　ASCII キャラクタ表」をご参照ください。

デリミタ

プログラム・メッセージ・ユニットを構成する文法要素は、コロン、ホワイト・スペース、コンマ、セミコロンで区切られます。

コロン（:） — 複合コマンド・ヘッダにおいて、個々のヘッダ・ニーモニックを接続するために使用します。

```
MMEMORY:DELETE:ALL
SOURCE:OSCILLATOR:SOURCE
```

ホワイト・スペース — ヘッダとアーギュメントを分離します。

```
HCOPY:PORT GPIB
SYSTEM:DATE 1997,7,7
```

この場合は、HCOPY:PORT と SYSTEM:DATE がコマンド・ヘッダで、GPIB と“ 1997,7,7 ”がアーギュメントになります。

コンマ（ , ） — 複数のアーギュメントを並べる場合に使用します。上の例では、“ 1997,7,7 ”がこの例です。

セミコロン（ ; ） — 複数のコマンド、問合せコマンドを接続する際に使用します。セミコロンの使用方法については、この後の「コマンドの接続」の項をご参照ください。

ホワイト・スペース

ホワイト・スペースは、ASCII コードの 0 ～ 9 および 11 ～ 32 の間のコードと定義され、文法要素を区切る際に使用されます。

なお、このマニュアルでは特別な場合を除き、スペースとホワイト・スペースの両方を単にスペースまたは空白と呼びます。

スペシャル・キャラクタ

ASCII の 127 ～ 255 の間のコードをスペシャル・コードと定義します。アービトラリ・ブロック・データに使用する場合を除き、これらのコードをプログラムに使用した場合には、結果が不定となります。

アーギュメント

アーギュメントは、プログラム・データとも呼ばれ、機器に値やモードを設定したり、コマンドの機能を制限したりするために使用します。コマンド・ヘッダの機能に応じて、次のタイプのデータがアーギュメントとして使用できます。

- 10 進数データ
- 文字列データ
- アービトラリ・ブロック・データ

10進数データ

10 進数データには、ANSI/IEEE Std.488.2 - 1987 で規定される NR1、NR2、NR3 と呼ばれる 3 つのタイプが利用できます（表 2-2 参照）。通常、この 3 つのいずれでも使用できる場合には、便宜的に NRf と記述します。

表 2-2: 数値表現

タイプ	フォーマット	例
NR1	整数	1、＋3、－2、＋10、－20
NR2	固定少数点実数	1.2、＋23.5、－0.15
NR3	浮動少数点実数	1E＋2、＋3.36E－2、－1.02E＋3

アーギュメントとして NR1 、NR2 、または NR3 のいずれかのタイプが要求されている場合でも、NRf の形式で入力できます。つまり、NR1 と指定されていても、NR2 や NR3 のタイプの形式で入力することができます。問合せコマンドによる問合せ結果の場合には、対応するコマンドのアーギュメントで要求される NR1 、NR2 、NR3 のタイプの形式で返送されます。たとえばシンタックスが、

```
:DESE <NR1>
```

と記述されている場合、次のいずれかの形でプログラムすることができます。

```
:DESE <NR1>
:DESE <NR2>
:DESE <NR3>
```

同じコマンドの問合せ形式に対するレスポンスは、シンタックスの記述に従って、

```
[:DESE] <NR1>
```

になります。

文字列データ

文字列データは、リテラル、あるいはストリングとも呼ばれるもので、文字列をダブルクォーテーション（ ” ）で囲んで表します。

” [＜文字列＞] ”

例　　”This is string constant.”

文字列データ中にダブルクォーテーション（”）を含める場合には，次の例のようにダブルクォーテーション（”）を続けて 2 個使用します。

例　　Serial Number ”B010000”

を文字列データとする場合、

”Serial Number ” ”B010000” ” ”

とします。

なお、ダブルクォーテーションの代わりにシングルクォーテーション（ ’ ）を使用することもできます。

例　　’Serial Nunber ”B010000’’ ’

アービトラリ・ブロック・データ

アービトラリ・ブロック・データは、次のように定義されます。

＃<byte count digit><byte count>[<contiguous 8-bit data byte>]...

ここで、各項目は次のように定義されます。

<byte count digit> ::＝ ‘0’を除く‘1’～‘9’までの ASCII キャラクタで、次に続く<byte count> の桁数を表します。

<byte count> ::＝ ‘0’～‘9’までのASCIIキャラクタで表現される 10 進数で、次に続く <contiguous 8-bit data byte> が何バイト続くかを表します。

<contiguous 8-bit data byte> ::＝ <byte count> で示されるバイト数の 8 ビット・データの集合です。

例　　＃16AB4ZLT

上の例では、

<byte count digit> ＝ 1
<byte count> ＝ 6
<contiguous 8-bit data byte> ＝AB4ZLT

単位と SI プリフィックス

数値データには、単位と SI プリフィックスを付加することができます。たとえば、200.0mV 、1.0MHz は、それぞれ 200E－3 、1.0E+6 の代わりにアーギュメントとして指定することができます。

単位として使用可能なシンボルは、次のとおりです。

V　　— 電圧を表します。
HZ　　— 周波数を表します。

また、SI プリフィックスとして使用可能なシンボルは、次のとおりです。

SI プリフィックスを表すシンボル	m/M	k/K	m/M
対応するベキ乗	10^{-3}	10^{+3}	10^{+6}

注： SI プリフィックス・シンボル m/M（ m または M ）は電圧に対しては 10^{-3} として、また周波数に対しては 10^{+6} として使用されます。

単位および SI プリフィックスとして使用されるシンボルは、大文字、小文字の両方が使用可能です。たとえば、次の例は同じ結果になります。

170mhz、170mHz、170MHz　など

250mv、250mV、250MV　など

ヘッダ

ヘッダ・ニーモニック

ヘッダ・ニーモニックは、ヘッダ・ノードまたはヘッダのサブ・ファンクションを表します。コマンドおよび問合せコマンドは、コロン（ : ）で区切られる 1 個以上のヘッダ・ニーモニックによって構成されます。

チャンネル表現

コマンドまたは問合せコマンドでは、OUTPut:CH<n> ヘッダ・ニーモニックを使用して、チャンネルを指定します。<n> は指定するチャンネルを表し、スタンダードで 0 ～ 3、オプション 01 型で 0 ～ 7 の数値を使用できます。

ヘッダの構造

本機器で使用するコマンドおよび問合せコマンドは、ヘッダの構造によって、次のような 6 つのコマンドあるいは問合せコマンドに分類されます。

- 単純・問合せコマンド・ヘッダ
- 複合コマンド・ヘッダ
- 複合・問合せコマンド・ヘッダ
- 共通コマンド・ヘッダ
- 共通・問合せコマンド・ヘッダ

単純コマンド・ヘッダ

単純コマンド・ヘッダから成るコマンドは、次のように、1 つのヘッダ・ニーモニック、または 1 つのヘッダ・ニーモニックとアーギュメントで構成されます。

[:]＜ヘッダ・ニーモニック＞ [＜アーギュメント＞［、＜アーギュメント＞］...]

例　START
　　STOP

単純・問合せコマンド・ヘッダ

単純・問合せコマンド・ヘッダから成るコマンドは、1つのヘッダ・ニーモニックに疑問符“？”を付加し、次のように表されます。

[:]＜ヘッダ・ニーモニック＞？ [＜アーギュメント＞［、＜アーギュメント＞］...]

例　HCOPY?
　　TRIGGER?

複合コマンド・ヘッダ

複合コマンド・ヘッダから成るコマンドは、複数のヘッダ・ニーモニックとアーギュメントを含み、次のように表されます。

[:]＜ヘッダ・ニーモニック＞:＜ヘッダ・ニーモニック＞[:＜ヘッダ・ニーモニック＞］...[＜アーギュメント＞［、＜アーギュメント＞］...]

例　MMEMORY:INITIALIZE　HD1
SYSTEM:SECURITY:STATE　ON

ここで、MMEMORY:INITIALIZE 、SYSTEM:SECURITY:STATE が複合コマンド・ヘッダで、HD1 と ON がアーギュメントです。

複合・問合せコマンド・ヘッダ

複合・問合せコマンド・ヘッダから成るコマンドは、複数のヘッダ・ニーモニックに疑問符“？”を付加し、次のように表されます。

[:]＜ヘッダ・ニーモニック＞:＜ヘッダ・ニーモニック＞[:＜ヘッダ・ニーモニック＞］...?[＜アーギュメント＞［、＜アーギュメント＞］...]

例　DATA:BLOCK:SIZE?　”BLOCK1”
DIAGNOSTIC:RESULT?
MMEMORY:CATALOG:ORDER?

共通コマンド・ヘッダ

共通コマンド・ヘッダから成るコマンドは、アスタリスク“＊”で始まるヘッダ・ニーモニックを含み、次のように表されます。

＜ヘッダ・ニーモニック＞　[＜アーギュメント＞［、＜アーギュメント＞］...]

例　*RST

アスタリスク“＊”を伴うコマンドは、IEEE Std 488.2 で定義されるコマンドです。これらのコマンドは、GPIB システムで IEEE Std 488.2 をサポートするすべての機器に共通に使用されます。

共通・問合せコマンド・ヘッダ

共通・問合せコマンド・ヘッダから成るコマンドは、アスタリスク“＊”で始まるヘッダ・ニーモニックに疑問符“？”を付加し、次のように表されます。

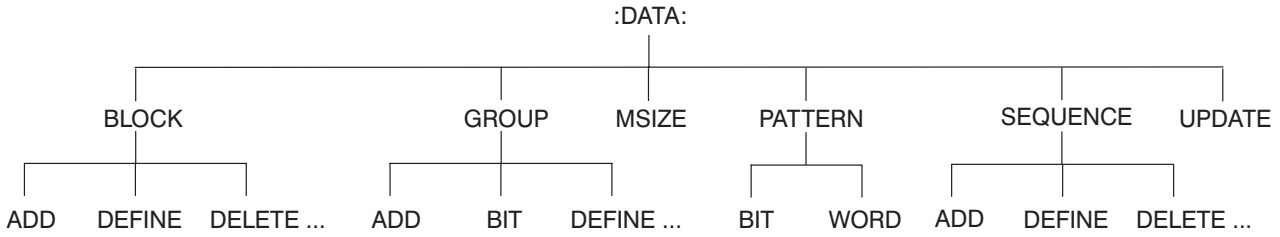
＜ヘッダ・ニーモニック＞?　[＜アーギュメント＞［、＜アーギュメント＞］...]

例　*IDN?

アスタリスク“＊”を伴う問合せコマンドは、IEEE Std 488.2 で定義されるコマンドです。これらの問合せコマンドは、GPIB システムで IEEE Std 488.2 をサポートするすべての機器に共通に使用されます。

コマンドの接続

複合コマンド・ヘッダと複合・問合せコマンド・ヘッダは、次の例のように、木構造になっています。



プログラム・メッセージは、プログラム・メッセージ・ユニット・デリミタ（ ; ）で区切られた 0 個以上のプログラム・メッセージ・ユニットから構成されるので、2 個以上のコマンドを組み合わせて、次のようにプログラム・メッセージを構成できます。

:DATA:BLOCK:ADD 512,”BLOCK3” ; :DATA:BLOCK:DELETE ”BLOCK2” ;
:DATA:BLOCK:SIZE ”BLOCK1”,512

2 番目と 3 番目のコマンドの前に付加されるコロン（ : ）は、木構造を持つ複合ヘッダの最上位ヘッダからコマンドが記述されていることを示すものです。ただし、本機器のコマンド・パーサは、上位のヘッダ・パス :DATA:BLOCK: を記憶し、次のコマンドのプリフィックスとして使用するように作られているので、2 番目以降のコマンド・ヘッダにすべてのパスを記述する必要はありません。上記のプログラム・メッセージは上位のヘッダ・パスが同じなので、次のように書き換えることができます。

:DATA:BLOCK:ADD 512,”BLOCK3” ; DELETE ”BLOCK2” ; SIZE ”BLOCK1”,512

次は、正しい記述の例です。

:DATA:BLOCK:ADD 512,”BLOCK3” ; DELETE ”BLOCK2” ; :DATA:GROUP:DELETE ”GROUP4”

:DATA:MSIZE 16384 ; BLOCK:ADD 512,”BLOCK3” ; DELETE ”BLOCK2”

:DATA:BLOCK:ADD 512,”BLOCK3” ; *SRE? ; DELETE ”BLOCK2” ; SIZE ”BLOCK1”,512

共通コマンドや共通・問合せコマンドを間に置いても、プリフィックスの規則には影響しません。

次は、誤った記述の例です。

:DATA:BLOCK:DELETE ”BLOCK2” ; DATA:GROUP:DELETE ”GROUP4”

:DATA:BLOCK:ADD 512,”BLOCK3” ; GROUP:DELETE ”GROUP4”

:DATA:BLOCK:DELETE ”BLOCK2” ; MSIZE 16384

レスポンス・メッセージ

問合せコマンドは、機器に対して、実行結果やパターン・データあるいは設定値やモードなどのステータスを外部コントローラに返送するように指示します（いくつかの問合せコマンドは、機器に対して何らかの機能を実行するように指示します。たとえば、*TST は機器に対してセルフ・テストを実行するように指示します）。

問合せコマンドに対するレスポンスは、特に説明がない限り、対応するコマンドの形式またはこのアーギュメントのみの形式で行われます。このレスポンスにヘッダを含めるかどうかは、 HEADER コマンドで定義することができます。

ヘッダが ON の場合には、問合せレスポンスの中にヘッダが含まれるようになります。この場合レスポンスは、正式な設定コマンドの形式になるので、後に、レスポンスをコマンドとして再利用することができます。

ヘッダが OFF の場合には、レスポンスの中にヘッダは含まれず、値だけが返送されます。

表 2-3 に、ヘッダが ON の場合と OFF の場合のレスポンスの違いを示します。ただし、VERBose が ON に設定されているものとします。

表 2-3: レスポンス・メッセージ中のヘッダ

問合せコマンド	ヘッダがONの場合	ヘッダがOFFの場合
DATA:MSIZE?	:DATA:MSIZE 16384	16384
DIAGNOSTIC:SELECT?	:DIAGNOSTIC:SELECT PMEMORY	PMEMORY

その他の規約

大文字と小文字の使用について

本機器は、大文字および小文字、または大文字と小文字を混ぜて使用することができます。たとえば、

```
HEADER  ON
```

は、次のように記述することもできます。

```
header on
```

または

```
Header On
```

省略について

予約語になっているヘッダおよびアーギュメントは、最低要求ワードを満たせば、省略して記述することができます。最低要求ワードは、この後の「コマンド詳細説明」の中で大文字で表され、省略可能な文字については小文字で表されています。たとえば、

TRIGger:SLOPe POSitive

は、次のように記述することができます。

TRIGGER:SLOPE POSITIVE
TRIG:SLOP POSIT
TRIG:SLOP POS

注： レスポンスに対する省略については、VERBose コマンドの説明をご参照ください。

コマンド・セット

この節では、コマンド・セットの構成について説明します。説明に当たっては” (?) ”のマークを使用しています。コマンド・ヘッダの後ろにこのマークが付いている場合、該当のコマンドは問合せコマンドを伴っていることを表します。それ以外のコマンドは、設定コマンドまたは問合せコマンドのいずれか一方を表します。

コマンド・グループ

本機器のコマンドは、その機能により、11 個のコマンド・グループに分類されます。

DATA サブシステム・コマンド

DATA サブシステムのコマンドは、ブロック、グループおよびシーケンスの定義、パターン・データの設定、ラン・モードが Enhanced のときに有効になるシーケンス・コントロールの設定などを行います。

表 2-4: DATA サブシステム・コマンド

ヘッダ	内　容
DATA?	データに関する設定状態の問合せ
DATA:BLOCk:ADD	ブロック定義の追加
DATA:BLOCk:DEFine(?)	ブロック定義の一括設定と問合せ
DATA:BLOCk:DELete	ブロック定義の削除
DATA:BLOCk:DELete:ALL	すべてのブロック定義の削除
DATA:BLOCk:REName	ブロック名の変更
DATA:BLOCk:SIZe(?)	ブロック・サイズの変更と問合せ
DATA:GROUp:ADD	グループ定義の追加
DATA:GROUp:BIT(?)	グループのビット構成の変更と問合せ
DATA:GROUp:DEFine(?)	グループ定義の一括設定と問合せ
DATA:GROUp:DELete	グループ定義の削除
DATA:GROUp:DELete:ALL	すべてのグループ定義の削除
DATA:GROUp:NAME?	グループ名の問合せ
DATA:GROUp:REName	グループ名の変更
DATA:MSIZe(?)	パターン・データのメモリ・サイズの設定と問合せ
DATA:PATTern:BIT(?)	パターン・データのビットごとの設定と問合せ
DATA:PATTern[:WORD](?)	パターン・データのワード単位での設定と問合せ
DATA:SEQuence:ADD	シーケンス・ステップの追加
DATA:SEQuence:DEFine(?)	シーケンス定義の一括設定と問合せ
DATA:SEQuence:DELete	シーケンス・ステップの削除
DATA:SEQuence:DELete:ALL	すべてのシーケンス定義の削除
DATA:SEQuence:EVJ(?)	イベント・ジャンプ ON/OFF の設定と問合せ
DATA:SEQuence:EVJTO(?)	イベント・ジャンプ先の設定と問合せ
DATA:SEQuence:LOOP(?)	無限ループ ON/OFF の設定と問合せ

表 2-4: DATA サブシステム・コマンド(続)

ヘッダ	内　容
DATA:SEQuence:REPeat(?)	リピート回数の設定と問合せ
DATA:SEQuence:TWAIT(?)	トリガ待ち ON/OFF の設定と問合せ
DATA:SUBSequence:ADD	サブ・シーケンス・ステップの追加
DATA:SUBSequence:CLEAr	すべてのサブ・シーケンス定義の削除
DATA:SUBSequence:DEFine(?)	サブ・シーケンス定義の設定と問合せ
DATA:SUBSequence:DELete	サブ・シーケンス・ステップの削除
DATA:SUBSequence:DELete:ALL	指定したサブ・シーケンス定義の削除
DATA:SUBSequence:REPeat(?)	サブ・シーケンス・ステップの繰り返し回数の設定と問い合せ
DATA:UPDate	強制的なパターン・データ等のアップデート

DEBUG サブシステム・コマンド

DEBUG サブシステムのコマンドは、デバッグ機能やデバッグ表示時間の設定を行います。

表 2-5: DEBAG サブシステム・コマンド

ヘッダ	内　容
DEBug?	デバッグ機能全般に関する問合せ
DEBug:SNOop?	デバッグ機能全般に関する問合せ
DEBug:SNOop:DELAy?	デバッグ機能の表示時間の問合せ
DEBug:SNOop:DELAy:TIME(?)	デバッグ機能表示時間に関する設定と問合せ
DEBug:SNOop:STATe(?)	デバッグ機能の ON/OFF 状態の設定と問合せ

DIAGNOSTIC サブシステム・コマンド

DIAGNOSTIC サブシステムのコマンドは、機能ごとに分類されたセルフ・テスト・ルーチンを選択し、実行します。

表 2-6: DIAGNOSTIC サブシステム・コマンド

ヘッダ	内　容
DIAGnostic?	セルフ・テストの種類と実行結果の問合せ
DIAGnostic:RESUlt?	セルフ・テストの実行結果の問合せ
DIAGnostic:SELect(?)	セルフ・テストの種類の設定と問合せ
DIAGnostic:STATe	セルフ・テストの実行開始

DISPLAY サブシステム・コマンド

DISPLAY サブシステムのコマンドは、メニュー選択の設定や管面の輝度調整などを行います。

表 2-7: DISPLAY サブシステム・コマンド

ヘッダ	内　容
DISPlay?	ディスプレイ全般に関する問合せ
DISPlay:BRIGhtness(?)	管面の輝度調整
DISPlay:CLOCk(?)	日付・時刻の表示状態の設定と問合せ
DISPlay:DIMmer(?)	ディスプレイのディマー動作の設定と問合せ
DISPlay:ENABle(?)	ディスプレイの ON/OFF 状態の設定と問合せ
DISPlay:MENU?	メニュー表示全般に関する問合せ
DISPlay:MENU[:NAME]	メニュー選択状態の設定
DISPlay:MENU:NAME?	メニュー選択状態の問合せ
DISPlay:MENU:STATe(?)	メニュー表示の ON/OFF 状態の設定と問合せ
DISPlay[:WINDow]:TEXT:CLEar	メッセージ表示エリアの消去
DISPlay[:WINDow]:TEXT[:DATA](?)	メッセージ表示内容の設定と問合せ

HARDCOPY サブシステム・コマンド

HARDCOPY サブシステムのコマンドは、ハードコピー機能の開始と停止、出力フォーマットとポートの選択を行います。

表 2-8: HARDCOPY サブシステム・コマンド

ヘッダ	内　容
HCOPy?	ハードコピー全般に関する問合せ
HCOPy:ABORt	ハードコピーの中止
HCOPy:DATA?	ハードコピーのデータ作成と返送
HCOPy:FORMat(?)	ハードコピーのデータ・フォーマットの設定と問合せ
HCOPy:PORT(?)	ハードコピーの出力ポートの設定と問合せ
HCOPy:STARt	ハードコピーの実行開始

MEMORY サブシステム・コマンド

MEMORY サブシステムのコマンドは、フロッピ・ディスクとファイル操作に関するすべての制御を行います。

表 2-9: MEMORY サブシステム・コマンド

ヘッダ	内　容
MMEMory:CATalog[:ALL]?	ディスクのファイル / ディレクトリ情報の問合せ
MMEMory:CATalog:ORDer(?)	ディスクのファイル / ディレクトリ情報の順序の設定と問合せ
MMEMory:CDIRectory(?)	ディスクのカレント・ディレクトリの設定と問合せ
MMEMory:COPY	ディスク・ファイルのコピー
MMEMory:DELete:ALL	ディスクのすべてのファイル / ディレクトリの削除
MMEMory:DELete[:NAME]	ディスクの指定したファイル / ディレクトリの削除
MMEMory:FREE?	ディスクの使用状況の問合せ
MMEMory:INITialize	ディスクの初期化
MMEMory:LOAD	ディスクのファイルからのデータ呼び出し
MMEMory:LOCK(?)	ディスクのファイルのロック状態の設定と問合せ
MMEMory:MDIRectory	ディスクのディレクトリ作成
MMEMory:RDIRectory	ディスクのディレクトリ削除
MMEMory:REName	ディスクのファイル / ディレクトリの名称変更
MMEMory:SAVE	ディスクのファイルへのデータ書き込み

MODE サブシステム・コマンド

MODE サブシステムのコマンドは、ラン・モードおよびアップデート・モードの設定を行います。

表 2-10: MODE サブシステム・コマンド

ヘッダ	内　容
MODE?	パターン発生に関する状態の問合せ
MODE:STATe(?)	パターン発生のラン・モードの設定と問合せ
MODE:UPDate(?)	データのアップデート・モードの設定と問合せ

OUTPUT サブシステム・コマンド

OUTPUT サブシステムのコマンドは、データ出力およびクロック出力に関するすべての設定を行います。データ出力チャンネルの指定は、ヘッダ・ニーモニック <n> で行います。

表 2-11: OUTPUT サブシステム・コマンド

ヘッダ	内　容
OUTPut?	出力に関する設定状態の問合せ
OUTPut:ELEVel(?)	イベント入力レベルの設定と問合せ
OUTPut:ILEVel(?)	インヒビット入力レベルの設定と問合せ
OUTPut:DEFine(?)	チャンネルのデータ・ビット割当ての一括設定と問合せ
OUTPut:CH<n>:ASSIGn(?)	データ出力のデータ・ビット割当ての設定と問合せ
OUTPut:CH<n>:DELAy(?)	データ出力のディレイ時間の設定と問合せ
OUTPut:CH<n>:DESKew(?)	データ出力のスキュー調整値の設定と問合せ
OUTPut:CH<n>:DESKew:RESET	データ出力のスキュー調整値のリセット
OUTPut:CH<n>:FALl(?)	データ出力の立ち下がり時間の設定と問合せ
OUTPut:CH<n>:FALl? RANge	データ出力の立ち下がり時間の有効設定範囲の問合せ
OUTPut:CH<n>:FALl? VALid	データ出力の立ち下がり時間とその設定値が有効範囲にあるかどうかの問合せ
OUTPut:CH<n>:HIGH(?)	データ出力の H レベル出力電圧の設定と問合せ
OUTPut:CH<n>:INHibit(?)	データ出力のハイ・インピーダンス制御方法の設定と問合せ
OUTPut:CH<n>:LOW(?)	データ出力の L レベル出力電圧の設定と問合せ
OUTPut:CH<n>:RELEase	データ出力のデータ・ビット割当ての解除
OUTPut:CH<n>:RISe(?)	データ出力の立ち上がり時間の設定と問合せ
OUTPut:CH<n>:RISe? RANge	データ出力の立ち上がり時間の有効設定範囲の問合せ
OUTPut:CH<n>:RISe? VALid	データ出力の立ち上がり時間とその設定値が有効範囲にあるかどうかの問合せ
OUTPut:CHCLK:FALL(?)	クロック出力の立ち下がり時間の有効設定範囲の問合せ
OUTPut:CHCLK:FALL? RANge	クロック出力の立ち下がり時間の有効設定範囲の問合せ
OUTPut:CHCLK:FALL? VALid	クロック出力の立ち下がり時間とその設定値が有効範囲にあるかどうかの問合せ
OUTPut:CHCLK:HIGH(?)	クロック出力の H レベル出力電圧の設定と問合せ
OUTPut:CHCLK:LOW(?)	クロック出力の L レベル出力電圧の設定と問合せ
OUTPut:CHCLK:RISE(?)	クロック出力の立ち上がり時間の設定と問合せ
OUTPut:CHCLK:RISE? RANge	クロック出力の立ち上がり時間の有効設定範囲の問合せ
OUTPut:CHCLK:RISE? VALid	クロック出力の立ち上がり時間とその設定値が有効範囲にあるかどうかの問合せ

SOURCE サブシステム・コマンド

SOURCE サブシステムのコマンドは、クロック信号源の選択、クロック周波数の設定などを行います。

表 2-12: SOURCEサブシステム・コマンド

ヘッダ	内　容
SOURce[:OSCillator]?	クロック信号に関するすべての問合せ
SOURce:OSCillator:EXTernal:FREQuency(?)	外部クロック周波数の入力と問合せ
SOURce:OSCillator[:INTernal]:FREQuency(?)	内部クロック周波数の設定と問合せ
SOURce:OSCillator[:INTernal]:PLLlock(?)	内部クロック発振回路の PLL 動作の設定と問合せ
SOURce:OSCillator:SOURce(?)	クロック信号の内部・外部選択の設定と問合せ
SOURce:EVENT:ENABle(?)	イベント入力の設定と問合せ

SYSTEM サブシステム・コマンド

SYSTEM サブシステムのコマンドは、時計の時刻、セキュリティ機能の ON/OFF など の設定を行います。

表 2-13: SYSTEM サブシステム・コマンド

ヘッダ	内　容
SYSTem:DATE(?)	時計の日付の設定と問合せ
SYSTem:PPAUse(?)	自己診断でエラーを検出した時の動作の設定と問合せ
SYSTem:SECurity:IMMediate	すべての設定値やデータの消去
SYSTem:SECurity:STATe(?)	セキュリティ ON/OFF の設定と問合せ
SYSTem:TIME(?)	時計の時刻の設定と問合せ

TRIGGER サブシステム・コマンド

TRIGGER サブシステムのコマンドは、トリガ入力に関するすべての設定を行います。

表 2-14: TRIGGER サブシステム・コマンド

ヘッダ	内　容
TRIGger?	トリガ入力に関するすべての問合せ
TRIGger:IMPedance(?)	トリガ・インピーダンスの設定と問合せ
TRIGger:INTERVal?	周期的に発生する内部トリガに関する問合せ
TRIGger:INTERVal:STATe(?)	周期的に発生する内部トリガ状態の設定と問合せ
TRIGger:INTERVal:TIME(?)	周期的に発生する内部トリガ時のインターバル時間の設定と問合せ
TRIGger:LEVel(?)	トリガ入力レベルの設定と問合せ
TRIGger:SLOPe(?)	トリガ入力の検出エッジの設定と問合せ
TRIGger:SOURce(?)	トリガ信号源の設定と問合せ

その他のコマンド

その他のコマンドは、前述したコマンド・グループに分類できないコマンドです。

表 2-15: その他のコマンド

ヘッダ	内　容
ABSTouch	キーおよびノブに対応する機能の実行
ALLEv?	イベント・キューのイベント・コード / メッセージ取出し
*CAL?	キャリブレーションの実行と結果の問合せ
*CLS	ステータス / イベントのクリア
DESE(?)	ステータス / イベント動作の設定と問合せ
*ESE(?)	ステータス / イベント動作の設定と問合せ
*ESR?	ステータス / イベント動作の問合せ
EVENT?	イベント・キューからのイベント・コード取出し
EVMsg?	イベント・キューからのイベント・メッセージ取出し
EVQty?	イベント・キューのイベント数の問合せ
FACTory	工場出荷時の状態への復帰
HEADer(?)	レスポンス中のコマンド・ヘッダの設定と問合せ
ID?	機器の ID 情報の問合せ
*IDN?	機器の ID 情報の問合せ
LOCk(?)	フロント・パネル操作のロックの設定と問合せ
*OPC(?)	ペンディング中の操作の終了の確認
*OPT?	オプション構成の問合せ
*PSC(?)	電源投入時のステータス / イベント・レポート動作の設定と問合せ
*RST	デフォルト状態へのリセット
RUNNing?	パターンまたはシーケンスの出力状態の問合せ
*SRE(?)	ステータス / イベント動作の設定と問合せ
STARt	パターンまたはシーケンス出力の開始
*STB?	ステータス・バイトの取出し
STOP	パターンまたはシーケンス出力の停止
*TST?	セルフ・テストの実行と結果の問合せ
*TRG	トリガ・イベントの生成
UNLock	フロント・パネルのロック解除
UPTime?	電源投入後の経過時間の問合せ
VERBose(?)	レスポンス中のコマンド・ヘッダの設定と問合せ
*WAI	ペンディング中の操作の終了待ち

コマンド詳細説明

ABSTouch

ABSTouchコマンドは、フロント・パネルのキーおよびノブに対応する機能を実行します。

グループ： DISPLAY

関連コマンド： なし

シンタックス： ABSTouch {BOTTOM1 | BOTTOM2 | BOTTOM3 | BOTTOM4 | BOTTOM5 | BOTTOM6 | BOTTOM7 | SIDE1 | SIDE2 | SIDE3 | SIDE4 | SIDE5 | CLEARMenu | SETUp | EDIT | APPLication | UTILity | CURSor | EXECute | UParrow | DOWNarrow | LEFTarrow | RIGHtarrow | KNOBLeft | KNOBRight | RUN | STEp | ZERo | ONE | TWO | THREe | FOUR | FIVe | SIX | SEVen | EIGHt | NINe | POINt | A | MINUs | B | HZ | S | V | C | KHZ | MS | MV | D | MHZ | US | E | NS | F | DELete | ENTer | HARDcopy | MANual }

アーギュメント： 各アーギュメントで指定されたコマンドの実行は、フロント・パネルの対応するキーを1回押すこと、またはノブを 1/25 回転だけ回すことに相当します。各アーギュメントは、図 2-2 のようにフロント・パネルのキーおよびノブに対応します。

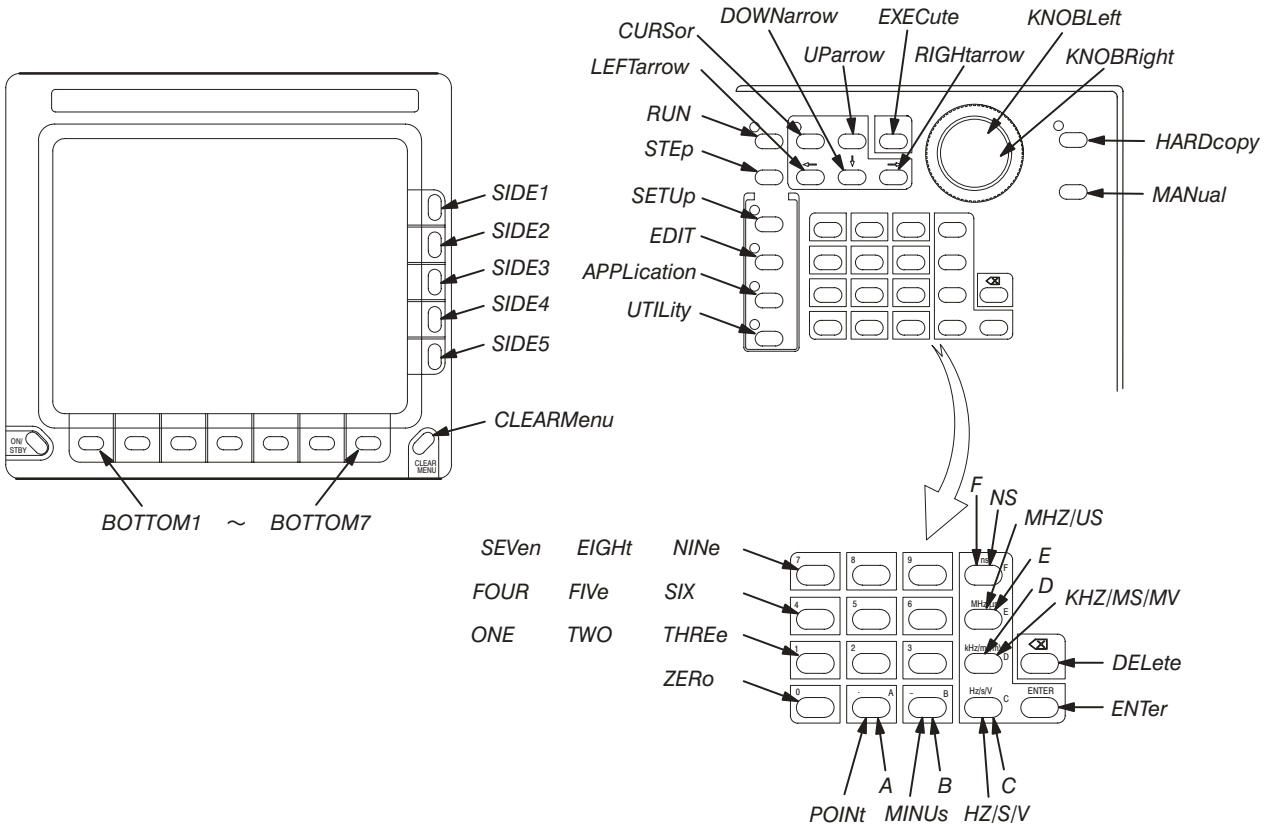


図 2-2: アーギュメントとフロント・パネル

使用例： 次に、本機器の管面に SETUP メニューを表示する例を示します。SETUP メニューは、マニュアル操作では、MENU 欄の SETUP キーを押すことによって表示できます。

```
:ABSTOUCH SETUP
```

ALLEv?

ALLEv? 問合せコマンドは、イベント・キューからすべてのイベント・コードと対応するイベント・メッセージを取り出します。なお、イベント・キューからの取り出しを可能にするには、*ESR? 問合せコマンドをあらかじめ実行することが必要です。

グループ： その他

関連コマンド： *CLS、DESE、*ESE、*ESR?、EVENT?、EVMsg?、EVQty?、*SRE、*STB?

シンタックス： ALLEv?

アーギュメント： なし

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:ALLEV] <イベント・コード>, ”<イベント・メッセージ：二次メッセージ>”[;<イベント・コード>, ”<イベント・メッセージ：二次メッセージ>”］...

使用例： 次に、:ALLEV？のレスポンス例を示します。

```
:ALLEV 113, ”Undefined header;unrecognized command – FG:CH1:AMP”; 420,
”Query UNTERMINATED”
```

*CAL?

*CAL? 共通・問合せコマンドは、機器のキャリブレーションを実行し、その結果を問い合わせます。なお、このコマンドではクロックのキャリブレーションのみが実行され、出力に関する補正などのキャリブレーションは実行されません。

グループ： その他

関連コマンド： なし

シンタックス： *CAL?

アーギュメント： なし

レスポンス： <Result>
<Result> ::= <NR1> は、次のいずれかです。
0–正常終了
800–クロックのキャリブレーション失敗

使用例： 次に、キャリブレーションを実行し、その結果を問い合わせる例を示します。

```
*CAL?
```

*CLS

*CLS 共通コマンドは、ステータス／イベント・レポーティング・システムで使用する SESR（ Standard Event Status Register ）、SBR（ Status Byte Register ）、およびイベント・キューをクリアします。

グループ： その他

関連コマンド： DESE、*ESE、*ESR?、EVENT?、EVMsg?、EVQty?、*SRE、*STB?

シンタックス： *CLS

使用例： 次に、SESR、SBR、およびイベント・キューをクリアする例を示します。

```
*CLS
```

DATA?

DATA? 問合せコマンドは、出力されるデータに関する設定を問い合わせます。

グループ： DATA

関連コマンド： OUTPut?

シンタックス： DATA?

使用例： 次に、:DATA? に対するレスポンス例を示します。

```
:DATA:MSIZE378;BLOCK:DEFINE#2440,BLOCK_1<LF>99,BLOCK_2<LF>189,
BLOCK_3<LF>288,BLOCK_4;:DATA:SUBSEQUENCE:DEFINE #217UNNAMED,1;
:DATA:SEQUENCE:DEFINE #271BLOCK_1,1,0,1,0,0<LF>BLOCK_2,1,0,0,1,0<LF>
BLOCK_3,1,0,0,0,0<LF>BLOCK_4,1,0,0,0:DATA:GROUP:DEFINE #279DATA7,7,7<LF>
DATA6,6,6<LF>DATA5,5,5<LF>DATA4,4,4<LF>DATA3,3,3<LF>DATA2,2,2<LF>DATA
1,1,1<LF>DATA0,0,0
```

DATA:BLOCk:ADD

DATA:BLOCk:ADD コマンドは、ブロックを追加します。これにより、ブロック定義セクションに新たなブロックが 1 つ定義されます。

グループ： DATA

関連コマンド： DATA:BLOCk:DEFine、DATA:BLOCk:DELete、DATA:BLOCk:DELete:ALL、DATA:BLOCk:REName、DATA:BLOCk:SIZe

シンタックス： DATA:BLOCk:ADD <Position>,<Name>

アーギュメント： <Position> ::= <NR1>　追加するブロックの先頭位置
<Name> ::= <string>　追加するブロックの名称

使用例： 次に、アドレス 512 から始まる BLOCK1 という名称のブロックを追加する例を示します。

```
:DATA:BLOCK:ADD  512,”BLOCK1”
```

DATA:BLOCk:DEFine(?)

DATA:BLOCk:DEFine コマンドは、ブロック定義セクション全体の情報をアスキー形式で設定します。また、DATA:BLOCk:DEFine? 問合せコマンドは、ブロック定義セクション全体の情報を問い合わせます。

グループ： DATA

関連コマンド： DATA:BLOCk:ADD、DATA:BLOCk:DELete、DATA:BLOCk:DELete:ALL、DATA:BLOCk:REName、DATA:BLOCk:SIZe

シンタックス： DATA:BLOCk:DEFine <Blockinfo>
DATA:BLOCk:DEFine?

アーギュメント： <Blockinfo> ::= <blockheader><Blkdef>[<LF><Blkdef>][<LF><Blkdef>]...
ブロック定義のアービトラリ・ブロック・データ

<blockheader> ::= <byte count digit><byte count>
<Blkdef> ::= <APosition>,<AName>
　　<Aposition>　アスキー文字で記述したブロックの先頭位置（最初のブロックの先頭位置は必ずゼロでなければなりません。）
　　<AName>　アスキー文字で記述したブロック名（大文字）
<LF> ::= ASCII 改行コード（10）

レスポンス： レスポンス・フォーマットは次のとおりです。

[:DATA:BLOCk:DEFINE] <Blockinfo>

ここで <Blockinfo> はアーギュメントと同形式のデータ・ブロックを表します。

使用例： 次に、BLOCK0、BLOCK1、BLOCK2 の 3 つのデータ・ブロックを定義する例を示します。

```
:DATA:BLOCK:DEFINE #2320,BLOCK0<LF>512,BLOCK1<LF>1024,BLOCK2
```

DATA:BLOCk:DELete

DATA:BLOCk:DELete コマンドは、指定したブロックを削除します。ただし、最初のブロックは削除できません。

グループ： DATA

関連コマンド： DATA:BLOCk:ADD、DATA:BLOCk:DEFine、DATA:BLOCk:DELete:ALL、DATA:BLOCk:REName、DATA:BLOCk:SIZe

シンタックス： DATA:BLOCk:DELete <Name>

アーギュメント： <Name> ::= <string>　　削除するブロックの名称

使用例： 次に、BLOCK2 という名称のブロックを削除する例を示します。

```
:DATA:BLOCK:DELETE ”BLOCK2”
```

DATA:BLOCk:DELete:ALL

DATA:BLOCk:DELete:ALL コマンドは、すべてのブロック定義を削除します。このコマンドを実行すると、全メモリ領域が ”UNNAMED” という名称のひとつのブロックになります。

グループ： DATA

関連コマンド： DATA:BLOCk:ADD、DATA:BLOCk:DEFine、DATA:BLOCk:DELete、DATA:BLOCk:REName、DATA:BLOCk:SIZe

シンタックス： DATA:BLOCk:DELete:ALL

アーギュメント： なし

DATA:BLOCk:REName

DATA:BLOCk:REName コマンドは、ブロック名を変更します。

グループ： DATA

関連コマンド： DATA:BLOCk:ADD、DATA:BLOCk:DEFine、DATA:BLOCk:DELete、DATA:BLOCk:DELete:ALL、DATA:BLOCk:SIZe

シンタックス： DATA:BLOCk:REName <Oldname>,<Newname>

アーギュメント： <Oldname> ::= <string>　　現在のブロック名
<Newname> ::= <string>　　新しいブロック名

使用例： 次に、ブロック名を BLOCK3 から BLOCK4 に変更する例を示します。

```
:DATA:BLOCK:RENAME ”BLOCK3”,”BLOCK4”
```

DATA:BLOCk:SIZe(?)

DATA:BLOCk:SIZe コマンドは、ブロック・サイズを変更します。また、DATA:BLOCk:SIZe? 問合せコマンドは、ブロック・サイズを問い合わせます。

グループ： DATA

関連コマンド： DATA:BLOCk:ADD、DATA:BLOCk:DEFine、DATA:BLOCk:DELete、DATA:BLOCk:DELete:ALL、DATA:BLOCk:REName

シンタックス：
DATA:BLOCk:SIZe <Name>,<Size>
DATA:BLOCk:SIZe? <Name>

アーギュメント：
<Name> ::= <string>　　ブロック名
<Size> ::= <NR1>　　新しいブロック・サイズ

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:DATA:BLOCK:SIZE] <Name>,<Size>

使用例： 次に、ブロック名 BLOCK1 のブロック・サイズを 512 に変更する例を示します。

```
:DATA:BLOCK:SIZE  ”BLOCK1”,512
```

DATA:GROUp:ADD

DATA:GROUp:ADD コマンドは、グループを追加します。

グループ： DATA

関連コマンド： DATA:GROUp:BIT、DATA:GROUp:DEFine、DATA:GROUp:DELete、DATA:GROUp:DELete:ALL、DATA:GROUp:NAME?、DATA:GROUp:REName

シンタックス： DATA:GROUp:ADD <Name>,<MSB>,<LSB>

アーギュメント：
<Name> ::= <string>　　追加するグループの名称
<MSB> ::= <NR1>　　グループの最上位ビット
<LSB> ::= <NR1>　　グループの最下位ビット

使用例： 次に、DATA00 から DATA03 までの 4 ビットで構成される GROUP01 という名称のグループを追加する例を示します。

```
:DATA:GROUP:ADD  ”GROUP01”,3,0
```

DATA:GROUp:BIT(?)

DATA:GROUp:BIT コマンドは、グループのビット構成を変更します。また、DATA:GROUp:BIT? 問合せコマンドは、設定されているグループのビット構成を問い合わせます。

グループ： DATA

関連コマンド： DATA:GROUp:ADD、DATA:GROUp:DEFine、DATA:GROUp:DELete、DATA:GROUp:DELete:ALL、DATA:GROUp:NAME?、DATA:GROUp:REName

シンタックス：
DATA:GROUp:BIT <Name>,<MSB>,<LSB>
DATA:GROUp:BIT? <Name>

アーギュメント：
<Name> ::= <string>　変更または問合せを行うグループの名称
<MSB> ::= <NR1>　グループの最上位ビット
<LSB> ::= <NR1>　グループの最下位ビット

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:DATA:GROUP:BIT] <Name>,<MSB>,<LSB>

使用例： 次に、GROUP02 という名称のグループのビット構成を DATA04 から DATA07 までに変更する例を示します。

```
:DATA:GROUP:BIT  ”GROUP02”,7,4
```

DATA:GROUp:DEFine(?)

DATA:GROUp:DEFine コマンドは、グループ定義セクション全体の情報をアスキー形式で設定します。また DATA:GROUp:DEFine? 問合せコマンドは、グループ定義セクション全体の情報を問い合わせます。

グループ： DATA

関連コマンド： DATA:GROUp:ADD、DATA:GROUp:BIT、DATA:GROUp:DELete、DATA:GROUp:DELete:ALL、DATA:GROUp:NAME?、DATA:GROUp:REName

シンタックス：
DATA:GROUp:DEFine <Groupblock>
DATA:GROUp:DEFine?

アーギュメント： <Groupblock> ::= <blockheader><Group>[<LF><Group>][<LF><Group>]...
グループ定義のアービトラリ・ブロック・データ

ここで

<blockheader> ::= <byte count digit><byte count>

<Group> ::= <AName>,<AMSB>,<ALSB>

<AName> 、<AMSB> および <ALSB> はアスキー文字で記述した次の情報です。

<AName>	グループ名（大文字）
<AMSB>	グループの最上位ビット
<ALSB>	グループの最下位ビット

<LF> ::= ASCII 改行コード（10）

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:DATA:GROUP:DEFINE] <Groupblock>

ここで <Groupblock> はアーギュメントと同形式のデータ・ブロックを表します。

使用例： 次に、GROUP01、GROUP02、GROUP03 の 3 つのグループを定義する例を示します。

```
:DATA:GROUP:DEFINE #238GROUP01,7,0<LF>GROUP02,11,8<LF>GROUP03,15,12
```

DATA:GROUp:DELete

DATA:GROUp:DELete コマンドは、指定したグループを削除します。

グループ： DATA

関連コマンド： DATA:GROUp:ADD、DATA:GROUp:BIT、DATA:GROUp:DEFine、
DATA:GROUp:DELete:ALL、DATA:GROUp:NAME?、DATA:GROUp:REName

シンタックス： DATA:GROUp:DELete <Name>

アーギュメント： <Name> ::= <string>　　削除するグループの名称

使用例： 次に、GROUP02 という名称のグループを削除する例を示します。

```
:DATA:GROUP:DELETE ”GROUP02”
```

DATA:GROUp:DELete:ALL

DATA:GROUp:DELete:ALL コマンドは、すべてのグループ定義を削除します。

グループ： DATA

関連コマンド： DATA:GROUp:ADD、DATA:GROUp:BIT、DATA:GROUp:DEFine、DATA:GROUp:DELete、
DATA:GROUp:NAME?、DATA:GROUp:REName

シンタックス： DATA:GROUp:DELete:ALL

アーギュメント： なし

DATA:GROUp:NAME?

DATA:GROUp:NAME? 問合せコマンドは、指定ビットを含むグループの名称を問い合わせます。

グループ： DATA

関連コマンド： DATA:GROUp:ADD、DATA:GROUp:BIT、DATA:GROUp:DEFine、DATA:GROUp:DELete、DATA:GROUp:DELete:ALL、DATA:GROUp:REName

シンタックス： DATA:GROUp:NAME? <Bit>

アーギュメント： <Bit> ::= <NR1>　　問い合わせるビットの番号（0～35）

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:DATA:GROUP:NAME] <Bit>,<Name>

ここで

<Bit> ::= <NR1>　　ビットの番号（0～35）
<Name> ::= <string>　グループ名

使用例： 次に、:DATA:GROUP:NAME? のレスポンス例を示します。

```
:DATA:GROUP:NAME  6,”GROUP02”
```

この場合、DATA06 ビットを含むグループの名称が GROUP02 であることを示しています。

DATA:GROUp:REName

DATA:GROUp:REName コマンドは、グループ名を変更します。

グループ： DATA

関連コマンド： DATA:GROUp:ADD、DATA:GROUp:BIT、DATA:GROUp:DEFine、DATA:GROUp:DELete、DATA:GROUp:DELete:ALL、DATA:GROUp:NAME?

シンタックス： DATA:GROUp:REName <Oldname>,<Newname>

アーギュメント： <Oldname> ::= <string>　　変更するグループ名
<Newname> ::= <string>　　新しいグループ名

使用例： 次に、グループ名を GROUP03 から GROUP04 に変更する例を示します。

```
:DATA:GROUP:RENAME  ”GROUP03”,”GROUP04”
```

DATA:MSIZe(?)

DATA:MSIZe コマンドは、ビット・パターン・セクションのメモリ領域を設定します。また、DATA:MSIZe? 問合せコマンドは、ビット・パターン・セクションのメモリ領域の設定を問い合わせます。

グループ： DATA

関連コマンド： なし

シンタックス：
DATA:MSIZe <Memsize>
DATA:MSIZe?

アーギュメント：
<Memsize> ::= <NR1>　　メモリ・サイズを表わす数値（ワード数）

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:DATA:MSIZE] <Memsize>

DATA:PATTern:BIT(?)

DATA:PATTern:BIT コマンドは、データ・メモリのビット・パターン・セクションを設定します。データはビット単位で与えます。また、DATA:PATTern:BIT? 問合せコマンドは、データ・メモリのビット・パターン・セクションの内容を問い合わせます。

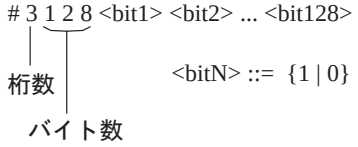
グループ： DATA

関連コマンド： DATA:PATTern[:WORD]

シンタックス：
DATA:PATTern:BIT <Bitpos>,<Address>,<Length>,<Data>
DATA:PATTern:BIT? <Bitpos>,<Address>,<Length>

アーギュメント：
<Bitpos> ::= <NR1>　　ビット位置（0～7）
<Address> ::= <NR1>　　開始アドレス（0～262143）
<Length> ::= <NR1>　　データ長（1～262144）
<Data> ::= <block>　　ビット・パターン・セクションのアービトラリ・ブロック・データ

データ長 = 128 の例：



指定されたアドレスから指定のデータ・ビットの値がアスキー文字の 1 または 0 で表わされ、以後アドレス順に指定データ長のデータ・ビットが同様にアスキー文字で格納されます。ブロック・ヘッダのバイト数は指定されたデータ長に等しくなります。

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:DATA:PATTERN:BIT] <Bitpos>,<Address>,<Length>,<Data>

DATA:PATTern[:WORD](?)

DATA:PATTern[:WORD] コマンドは、データ・メモリのビット・パターン・セクションを設定します。データは、ワード単位で与えます。また、DATA:PATTern:WORD? 問合せコマンドは、データ・メモリのビット・パターン・セクションの内容を問い合わせます。

グループ： DATA

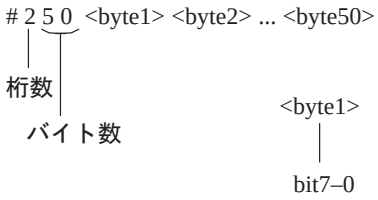
関連コマンド： DATA:PATTern:BIT

シンタックス： DATA:PATTern[:WORD] <Address>,<Length>,<Data>
DATA:PATTern:WORD? <Address>,<Length>

アーギュメント：

<Address> ::= <NR1>	開始アドレス（0～262143）
<Length> ::= <NR1>	データ長（1～262144）
<Data> ::= <block>	ビット・パターン・セクションのアービトラリ・ブロック・データ

データ長 = 50 の例：



各バイトが、ビット・パターン・データの 1 ワード（8 ビット）を表します。各バイトでは、LSB 側が bit 0 に、MSB 側が bit 7 に対応します。この繰り返しによって、指定の開始アドレスのデータから順に各ワードをデータ・ブロックにします。

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:DATA:PATTERN:WORD] <Address>,<Length>,<Data>

DATA:SEQuence:ADD

DATA:SEQuence:ADD コマンドは、シーケンス・ステップを追加します。

グループ： DATA

関連コマンド： DATA:SEQuence:DEFine、DATA:SEQuence:DELete、DATA:SEQuence:DELete:ALL

シンタックス： DATA:SEQuence:ADD <LineN>,<Name>,<Repeat>,<To>,<WaitE>,<JumpE>,<LoopE>

アーギュメント：

<LineN> ::= <NR1>	シーケンス・ステップ行番号 既にある行とつながるかオーバラップする番号を使用します。番号を飛ばすことはできません。
<Name> ::= <string>	ブロック名（「”」または「’」で囲む。）
<Repeat> ::= <NR1>	繰り返し回数（ 1 ～ 65536 ）
<To> ::= <NR1>	イベント・ジャンプ先の行番号
<WaitE> ::= {ON | OFF | 1 | 0}	トリガ・ウェイト ON/OFF（ {ON | 1}:ON, {OFF | 0}:OFF ）
<JumpE> ::= {ON | OFF | 1 | 0}	イベント・ジャンプON/OFF（ {ON | 1}:ON, {OFF | 0}:OFF ）
<LoopE> ::= {ON | OFF | 1 | 0}	無限ループON/OFF（ {ON | 1}:ON, {OFF | 0}:OFF ）

ラインの挿入によって、以前からあったその行番号以降の行は、1 つ後ろにずれます。新たに挿入されるシーケンス・ステップ内のジャンプ先行番号は、上記の行番号更新の結果の行番号をあらかじめ指定します。

使用例： 次に、シーケンスの行番号 4 の位置に BLOCK3 というブロックから成るシーケンス・ステップを追加する例を示します。

```
:DATA:SEQUENCE:ADD  4,”BLOCK3”,16,0,0,1,1
```

エンハンスド・モードでこのシーケンスを実行すると、無限ループが ON に指定されているので BLOCK3 を繰り返し実行します。ただし、イベント・ジャンプが ON に指定されているので、外部イベントが発生したときにシーケンスの行番号 0 にジャンプします。エンハンスド・モード以外のラン・モードでは BLOCK3 が 16 回実行され、次の行番号に移ります。

DATA:SEQuence:DEFine(?)

DATA:SEQuence:DEFine コマンドは、シーケンス定義セクション全体の情報をアスキー形式で設定します。また、DATA:SEQuence:DEFine? 問合せコマンドは、シーケンス定義セクション全体の情報を問い合わせます。

グループ： DATA

関連コマンド： DATA:SEQuence:ADD、DATA:SEQuence:DELete、DATA:SEQuence:DELete:ALL

シンタックス： DATA:SEQuence:DEFine <Seqblock>
DATA:SEQuence:DEFine?

アーギュメント： <Seqblock> ::= <blockheader><Step>[<LF><Step>][<LF><Step>]...
シーケンス定義のアービトラリ・ブロック・データ

ここで

<blockheader> ::= <byte count digit><byte count>

<Step> ::= <AName>,<ARepeat>,<ATo>,<AWaitE>,<AJumpE>,<ALoopE>

各項目はアスキー文字で記述した次の情報

<AName>	ブロック名（引用符なし）
<ARepeat>	繰り返し回数（ 1～65536 ）
<ATo>	イベント・ジャンプ先の行番号
<AWaitE>	トリガ・ウェイト ON/OFF（{ON | 1}:ON, {OFF | 0}:OFF）
<AJumpE>	イベント・ジャンプ ON/OFF（{ON | 1}:ON, {OFF | 0}:OFF）
<ALoopE>	無限ループ ON/OFF（{ON | 1}:ON, {OFF | 0}:OFF）

<LF> ::= ASCII 改行コード（10）

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:DATA:SEQUENCE:DEFINE] <Seqblock>

ここで、<Seqblock> はアーギュメントと同形式のデータ・ブロックですが、<AWaitE>、<AJumpE> および <ALoopE> に ON、OFF のキー・ワードを使用せず、1 、0 のみを使用します。

使用例： 次に、BLOCK1、BLOCK2 という 2 つのブロックで構成される 2 ステップのシーケンスを定義する例を示します。

:DATA:SEQUENCE:DEFINE #235BLOCK1,16,0,1,0,0<LF>BLOCK2,32,0,0,1,1

DATA:SEQuence:DELete

DATA:SEQuence:DELete コマンドは、指定したシーケンス・ステップを削除します。

グループ： DATA

関連コマンド： DATA:SEQuence:ADD、DATA:SEQuence:DEFine、DATA:SEQuence:DELete:ALL

シンタックス： DATA:SEQuence:DELete <Linenum>

アーギュメント： <Linenum> ::= <NR1>　　削除するシーケンス・ステップの行番号

使用例： 次に、行番号 3 のシーケンス・ステップを削除する例を示します。

```
:DATA:SEQUENCE:DELETE  3
```

DATA:SEQuence:DELete:ALL

DATA:SEQuence:DELete:ALL コマンドは、すべてのシーケンス定義を削除します。

グループ： DATA

関連コマンド： DATA:SEQuence:ADD、DATA:SEQuence:DEFine、DATA:SEQuence:DELete

シンタックス： DATA:SEQuence:DELete:ALL

アーギュメント： なし

DATA:SEQuence:EVJ(?)

DATA:SEQuence:EVJ コマンドは、シーケンス・ステップのイベント・ジャンプを ON または OFF にします。また、DATA:SEQuence:EVJ? 問合せコマンドは、シーケンス・ステップのイベント・ジャンプの ON/OFF 状態を問い合わせます。

グループ： DATA

関連コマンド： DATA:SEQuence:EVJTO、DATA:SEQuence:LOOP、DATA:SEQuence:REPeat、DATA:SEQuence:TWAIT

シンタックス：
DATA:SEQuence:EVJ <Linenum>,<Flag>
DATA:SEQuence:EVJ? <Linenum>

アーギュメント：
<Linenum> ::= <NR1>　　シーケンス・ステップの行番号
<Flag> ::= {ON | OFF | 1 | 0}　　ON/OFF の状態（ {ON | 1}:ON, {OFF | 0}:OFF ）

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:DATA:SEQUENCE:EVJ] <Linenum>,{1 | 0}

使用例： 次に、行番号 8 のシーケンス・ステップのイベント・ジャンプを ON に設定する例を示します。

```
:DATA:SEQUENCE:EVJ  8,ON
```

DATA:SEQuence:EVJTO(?)

DATA:SEQuence:EVJTO コマンドは、シーケンス・ステップのイベント・ジャンプ先を設定します。また、DATA:SEQuence:EVJTO? 問合せコマンドは、シーケンス・ステップの設定されているイベント・ジャンプ先を問い合わせます。

グループ： DATA

関連コマンド： DATA:SEQuence:EVJ、DATA:SEQuence:LOOP、DATA:SEQuence:REPeat、DATA:SEQuence:TWAIT

シンタックス：
DATA:SEQuence:EVJTO <Linenum>,<Target>
DATA:SEQuence:EVJTO? <Linenum>

アーギュメント：
<Linenum> ::= <NR1>　　シーケンス・ステップの行番号
<Target> ::= <NR1>　　ジャンプ先のステップの行番号

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:DATA:SEQUENCE:EVJTO] <Linenum>,<Target>

使用例： 次に、行番号 5 のシーケンス・ステップのイベント・ジャンプ先を行番号 0 に設定する例を示します。

```
:DATA:SEQUENCE:EVJTO  5,0
```

DATA:SEQuence:LOOP(?)

DATA:SEQuence:LOOP コマンドは、シーケンス・ステップの無限ループを ON または OFF に設定します。また、DATA:SEQuence:LOOP? 問合せコマンドは、シーケンス・ステップの無限ループの ON/OFF 状態を問い合わせます。

グループ： DATA

関連コマンド： DATA:SEQuence:EVJ、DATA:SEQuence:EVJTO、DATA:SEQuence:REPeat、DATA:SEQuence:TWAIT

シンタックス：
DATA:SEQuence:LOOP <Linenum>,<Flag>
DATA:SEQuence:LOOP? <Linenum>

アーギュメント：
<Linenum> ::= <NR1>　　シーケンス・ステップの行番号
<Flag> ::= {ON | OFF | 1 | 0}　　ON/OFF の状態（ {ON | 1}:ON, {OFF | 0}:OFF ）

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:DATA:SEQUENCE:LOOP] <Linenum>,{1 | 0}

使用例： 次に、行番号 9 のシーケンス・ステップの無限ループを OFF に設定する例を示します。

```
:DATA:SEQUENCE:LOOP  9,OFF
```

DATA:SEQuence:REPeat(?)

DATA:SEQuence:REPeat コマンドは、シーケンス・ステップの繰り返し回数を設定します。また、DATA:SEQuence:REPeat? 問合せコマンドは、シーケンス・ステップの設定されている繰り返し回数を問い合わせます。

グループ： DATA

関連コマンド： DATA:SEQuence:EVJ、DATA:SEQuence:EVJTO、DATA:SEQuence:LOOP、DATA:SEQuence:TWAIT

シンタックス：
DATA:SEQuence:REPeat <Linenum>,<Times>
DATA:SEQuence:REPeat? <Linenum>

アーギュメント：
<Linenum> ::= <NR1>　　シーケンス・ステップの行番号
<Times> ::= <NR1>　　繰り返し回数（ 1～65536 ）

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:DATA:SEQUENCE:REPEAT] <Linenum>,<Times>

使用例： 次に、行番号 5 のシーケンス・ステップの繰り返し回数を 8 に設定する例を示します。

```
:DATA:SEQUENCE:REPEAT  5,8
```

DATA:SEQuence:TWAIT(?)

DATA:SEQuence:TWAIT コマンドは、シーケンス・ステップのトリガ・ウェイトを ON または OFF に設定します。また、DATA:SEQuence:TWAIT? 問合せコマンドは、シーケンス・ステップのトリガ・ウェイトの ON/OFF 状態を問い合わせます。

グループ： DATA

関連コマンド： DATA:SEQuence:EVJ、DATA:SEQuence:EVJTO、DATA:SEQuence:LOOP、DATA:SEQuence:REPeat

シンタックス：
DATA:SEQuence:TWAIT <Linenum>,<Flag>
DATA:SEQuence:TWAIT? <Linenum>

アーギュメント：
<Linenum> ::= <NR1>　　シーケンス・ステップの行番号
<Flag> ::= {ON | OFF | 1 | 0}　　ON/OFF の状態（ {ON | 1}:ON, {OFF | 0}:OFF ）

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:DATA:SEQUENCE:TWAIT] <Linenum>,{1 | 0}

使用例： 次に、行番号 5 のシーケンス・ステップのトリガ・ウェイトを ON に設定する例を示します。

```
:DATA:SEQUENCE:TWAIT  5,ON
```

DATA:SUBSequence:ADD

DATA:SUBSequence:ADD コマンドは、サブ・シーケンス・ステップを追加します。

グループ： DATA

関連コマンド： DATA:SUBSequence:DEFine、DATA:SUBSequence:DELete、DATA:SUBSequence:DELete:ALL

シンタックス： DATA:SUBSequence:ADD <Sname>,<LineN>,<Name>,<Repeat>

アーギュメント：
<Sname> ::= <String>　　サブ・シーケンス名（「”」または「’」で囲む。）
<LineN> ::= <NR1>　　サブ・シーケンス・ステップ行番号
<Name> ::= <String>　　ブロック名（「”」または「’」で囲む。）
<Repeat> ::= <NR1>　　繰り返し回数（1～65536）

使用例： 次に、サブ・シーケンス名 SUB1 の行番号 2 の位置に、BLOCK3 という名称のブロックを 10 回繰り返すサブ・シーケンス・ステップを追加する例を示します。

```
:DATA:SUBSEQUENCE:ADD ”SUB1”,2,”BLOCK3”,10
```

DATA:SUBSequence:CLEAr

DATA:SUBSequence:CLEAr コマンドは、すべてのサブ・シーケンス定義を削除します。

グループ： DATA

関連コマンド： DATA:SUBSequence:ADD、DATA:SUBSequence:DEFine、DATA:SUBSequence:DELete、DATA:SUBSequence:DELete:ALL

シンタックス： DATA:SUBSequence:DELete:CLEAr

アーギュメント： なし

DATA:SUBSequence:DEFine(?)

DATA:SUBSequence:DEFine コマンドは、サブ・シーケンス定義セクション全体の情報をアスキー形式で設定します。また、DATA:SUBSequence:DEFine? 問合せコマンドは、サブ・シーケンス定義セクション全体の情報を問い合わせます。

グループ： DATA

関連コマンド： DATA:SUBSequence:ADD、DATA:SUBSequence:CLEAr、DATA:SUBSequence:DELete、DATA:SUBSequence:DELete:ALL

シンタックス： DATA:SUBSequence:DEFine <SubSeq block>
DATA:SUBSequence:DEFine?

アーギュメント： <SubSeqb lock> ::=
<blockheader><SName>,<step>[,<step>...][<LF><SName>,<step>[,<step>...]...]
シーケンス定義のアービトラリ・ブロック・データ

ここで

<step> ::= <AName>,<ARepeat>

各項目はアスキー文字で記述した次の情報

<SName>	サブ・シーケンス名（引用符なし）
<AName>	ブロック名（引用符なし）
<ARepeat>	繰り返し回数（ 1～65536 ）

<LF> ::= ASCII 改行コード（10）

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:DATA:SUBSEQUENCE:DEFINE] <SubSeq block>

ここで、<SubSeq block> はアーギュメントと同形式のデータ・ブロックです。

使用例： 次に、ブロック名 B1、B2 を持つ サブシーケンス SUB1、およびブロック名 B3、B4 を持つサブシーケンス SUB2 という 2 つのブロックで構成される 2 ステップのサブシーケンスを定義する例を示します。

```
:DATA:SUBSEQUENCE:DEFINE #233SUB1,B1,16,B2,32<LF>SUB2,B3,2,B4,3
```

DATA:SUBSequence:DELete

DATA:SUBSequence:DELete コマンドは、指定したサブ・シーケンス・ステップを削除します。

グループ： DATA

関連コマンド： DATA:SUBSequence:ADD、DATA:SUBSequence:CLEAr、DATA:SUBSequence:DEFine、DATA:SUBSequence:DELete:ALL

シンタックス： DATA:SUBSequence:DELete <SName>,<Linenum>

アーギュメント： <SName> ::= <String>　サブ・シーケンス名（「”」または「’」で囲む。）
<Linenum> ::= <NR1>　削除するシーケンス・ステップの行番号

使用例： 次に、サブ・シーケンス名 SUB2 の行番号 7 にあるサブ・シーケンス・ステップを削除する例を示します。

```
:DATA:SUBSEQUENCE:DELETE  ”SUB2”,7
```

DATA:SUBSequence:DELete:ALL

DATA:SUBSequence:DELete:ALL コマンドは、指定したサブ・シーケンス定義を削除します。

グループ： DATA

関連コマンド： DATA:SUBSequence:ADD、DATA:SUBSequence:CLEAr、DATA:SUBSequence:DEFine、DATA:SUBSequence:DELete

シンタックス： DATA:SUBSequence:DELete:ALL <SName>

アーギュメント： <SName> ::= <String>　サブ・シーケンス名（「”」または「’」で囲む。）

使用例： 次に、サブシーケンス定義 SUB1 を削除する例を示します。

```
:DATA:SUBSEQUENCE:DELETE:ALL ”SUB1”
```

DATA:SUBSequence:REPeat(?)

DATA:SUBSequence:REPeat コマンドは、サブ・シーケンス・ステップの繰り返し回数を設定します。また、DATA:SUBSequence:REPeat? 問合せコマンドは、サブ・シーケンス・ステップの繰り返し回数を問い合わせます。

グループ： DATA

関連コマンド：

シンタックス： DATA:SUBSequence:REPeat <SName>,<Linenum>,<Times>
DATA:SUBSequence:REPeat? <SName>,<Linenum>

アーギュメント： <SName> ::= <String>　サブ・シーケンス名（「”」または「’」で囲む。）
<Linenum> ::= <NR1>　シーケンス・ステップの行番号
<TImes> ::= <NR1>　繰り返し回数（1～65536）

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:DATA:SUBSEQUENCE:REPEAT] <SName>,<Linenum>,<Times>

使用例： 次に、サブ・シーケンス名 SUB1 の行番号 5 のシーケンス・ステップの回数を 8 に設定する例を示します。

```
:DATA:SUBSEQUENCE:REPEAT ” SUB1”,5,8
```

DATA:UPDate

DATA:UPDate コマンドは、データ・メモリの内容をパターン発生メモリに転送し、最新のデータを出力に反映させます。このコマンドは、モードがマニュアルに設定されている場合だけ有効です。モードがオートに設定されている場合、このコマンドの処理はデータの変更に応じて自動的に行われます。

グループ： DATA

関連コマンド： なし

シンタックス： DATA:UPDate

アーギュメント： なし

DEBug?

DEBug? 問合せコマンドは、リモート・コマンドのデバッグ機能の全設定を問い合わせます。このコマンドは、DEBug:SNOop? 問合せコマンドと同等です。

グループ： DEBUG

関連コマンド： DEBug:SNOop?、DEBug:SNOop:DELAy?、DEBug:SNOop:DELAy:TIME、DEBug:SNOop:STATe

シンタックス： DEBug?

アーギュメント： なし

レスポンス： 詳しくは、使用例をご参照ください。

使用例： 次に、:DEBUG? のレスポンス例を示します。

```
:DEBUG:SNOOP:STATE 0;DELAY:TIME 0.2
```

DEBug:SNOop?

DEBug:SNOop? 問合せコマンドは、リモート・コマンドのデバッグ機能の全設定を問い合わせます。このコマンドは、DEBug? 問合せコマンドと同等です。

グループ： DEBUG

関連コマンド： DEBug?、DEBug:SNOop:DELAy?、DEBug:SNOop:DELAy:TIME、DEBug:SNOop:STATe

シンタックス： DEBug:SNOop?

アーギュメント： なし

レスポンス： 詳しくは、使用例をご参照ください。

使用例： 次に、:DEBUG:SNOOP? のレスポンス例を示します。

```
:DEBUG:SNOOP:STATE 0;DELAY:TIME 0.2
```

DEBug:SNOop:DELAy?

DEBug:SNOop:DELAy? 問合せコマンドは、リモート・インターフェイスより入力されたコマンドが、”;” で接続されている場合における各々のコマンドの表示時間を問い合わせます。このコマンドは、DEBug:SNOop:DELAy:TIME? 問合せコマンドと同等です。

グループ： DEBUG

関連コマンド： DEBug?、DEBug:SNOop?、DEBug:SNOop:DELAy:TIME、DEBug:SNOop:STATe

シンタックス： DEBug:SNOop:DELAy?

アーギュメント： なし

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:DEBUG:SNOOP:DELAY] <Delay time>

ここで

<Delay time> ::= <NR2>

使用例： 次に、:DEBUG:SNOOP:DELAY? のレスポンス例を示します。

```
:DEBUG:SNOOP:DELAY:TIME 0.2
```

DEBug:SNOop:DELAy:TIME(?)

DEBug:SNOop:DELAy:TIME コマンドは、リモート・インターフェイスより入力されたコマンドが、”;” で接続されている場合における各々のコマンドの表示時間を設定します。また、DEBug:SNOop:DELAy:TIME? 問合せコマンドは、コマンドが ”;” で接続されている場合における各々のコマンドの表示時間を問い合わせます。

グループ： DEBUG

関連コマンド： DEBug?、DEBug:SNOop?、DEBug:SNOop:DELAy?、DEBug:SNOop:STATe

シンタックス： DEBug:SNOop:DELAy:TIME <Time>
DEBug:SNOop:DELAy:TIME?

アーギュメント： <Time> ::= <NR2>[<unit>]
<unit> ::= {s | ms | us}
設定範囲：0.0 s ～ 10.0 s ステップ 0.1 s

使用例： 次に、コマンドの表示時間を 0.5 s に設定する例を示します。

```
:DEBUG:SNOOP:DELAY:TIME 0.5
```

DEBug:SNOop:STATe(?)

DEBug:SNOop:STATe コマンドは、リモート・コマンドのデバッグ機能の設定および解除を行います。また、DEBug:SNOop:STATe? 問合せコマンドは、リモート・コマンドのデバッグ機能の設定状況を問い合わせます。

デバッグ機能は、リモート・インターフェイスより入力されるメッセージを管面上のメッセージ領域に表示します。コマンドが ”;” で接続されている場合は、” ; ” で区切られるごとに DEBug:SNOop:DELAy:TIME で設定される時間だけ各々表示されます。

表示形式は、次のとおりです。

コントロール・コード　—　”<コードの 10 進表示>” 。たとえば、LF の場合は ”<10>” 。
英数字・シンボル　—　”<コードの ASCII 表示>” たとえば、”A”の場合は ”A” 。
メセージ・ターミネーション　—　”<PMT>” 。
インターフェース・メッセージ　—　”<DCL>” と <GET>。他は、”<コードの10進表示>”。
ブロック・データ　—　”#0” 。
上記以外　—　”<コードの 10 進表示>” 。たとえば、コード 80 （ 16 進）の場合は、<128> 。

グループ： DEBUG

関連コマンド： DEBug?、DEBug:SNOop?、DEBug:SNOop:DELAy?、DEBug:SNOop:DELAy:TIME

シンタックス：
DEBug:SNOop:STATe {ON | OFF | <NR1>}
DEBug:SNOop:STATe?

アーギュメント：
ON または 0 以外の値　— デバッグ機能を設定します。
OFF または 0　— デバッグ機能を解除します。

レスポンス： 問合せに対するレスポンスは、次のとおりです。
1　— デバッグ機能が設定されています。
0　— デバッグ機能は設定されていません。

使用例： 次に、デバッグ機能を設定する例を示します。

```
:DEBUG:SNOOP:STATE ON
```

DESE(?)

DESE コマンドは、ステータス／イベント・レポーティング・システムで使用する DESER（ Device Event Status Enable Register ）に値を設定します。また、DESE? 問合せコマンドは、DESER の値を問い合わせます。

グループ： その他

関連コマンド： *CLS、*ESE、*ESR?、EVENT?、EVMsg?、EVQty?、*SRE、*STB?

シンタックス：
DESE <Bit Value>
DESE?

アーギュメント： <Bit Value> ::= <NR1>　（ 設定範囲:　0 ～ 255 ）

アーギュメントは、0 から 255 の範囲の 10 進数でなければなりません。DESER には、この値に対応する 2 進数が設定されます。

電源投入時の DESER の値は、PSC フラグが TRUE の場合、すべてのビットがセットされます。PSC フラグが FALSE の場合、電源の ON/OFFとは無関係に、DESER の値が保持されます。PSC フラグについては、*PSC コマンドの項をご参照ください。

使用例： 次に、DESER を 177（ 10110001 ）に設定する例を示します。この場合、PON 、CME 、EXE、および OPC の各ビットがセットされます。

```
:DESE 177
```

次に、:DESE? のレスポンス例を示します。

```
:DESE 176
```

この場合、DESERの内容は、10110000 になります。

DIAGnostic?

DIAGnostic? 問合せコマンドは、選択されているセルフ・テストの種類と実行結果を問い合わせます。

グループ： DIAGNOSTIC

関連コマンド： DIAGnostic:RESUlt?、DIAGnostic:SELect、DIAGnostic:STATe

シンタックス： DIAGnostic?

アーギュメント： なし

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:DIAGNOSTIC:SELECT] <Testtype>;[RESULT] <Result>[,<Result]...

ここで

<Testtype> は、次のいずれかです。

ALL	—	すべてのセルフ・テスト
CPU	—	CPUユニットのテスト
DISPlay	—	ディスプレイ・ユニットのテスト
FPANel	—	フロント・パネルのテスト
CLOCk	—	クロック発生ユニットのテスト
TRIGger	—	トリガ信号処理ユニットのテスト
SMEMory	—	シーケンス・メモリのテスト
PMEMory	—	パターン・メモリのテスト

<Result> ::= <NR1>　次のいずれかです。

0	—	正常終了
100	—	CPUユニットでエラーを検出
200	—	ディスプレイ・ユニットでエラーを検出
300	—	フロント・パネルでエラーを検出
400	—	クロック発生ユニットでエラーを検出
500	—	トリガ処理ユニットでエラーを検出
600	—	シーケンス・メモリでエラーを検出
700	—	パターン・メモリでエラーを検出

使用例： 次に、:DIAGNOSTIC? のレスポンス例を示します。

```
:DIAGNOSTIC:SELECT ALL;RESULT 0
```

DIAGnostic:RESUlt?

DIAGnostic:RESUlt? 問合せコマンドは、セルフ・テストの実行結果を問い合わせます。

グループ： DIAGNOSTIC

関連コマンド： DIAGnostic?、DIAGnostic:SELect、DIAGnostic:STATe

シンタックス： DIAGnostic:RESUlt?

アーギュメント： なし

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:DIAGNOSTIC:RESULT] <Result> [,<Result>]...

ここで

<Result> ::= <NR1>　次のいずれかです。

0	—	正常終了
100	—	CPUユニットでエラーを検出
200	—	ディスプレイ・ユニットでエラーを検出
300	—	フロント・パネルでエラーを検出
400	—	クロック発生ユニットでエラーを検出
500	—	トリガ処理ユニットでエラーを検出
600	—	シーケンス・メモリでエラーを検出
700	—	パターン・メモリでエラーを検出

使用例： 次に、:DIAGNOSTIC:RESULT? のレスポンス例を示します。

```
:DIAGNOSTIC:RESULT  200, 400
```

この場合、ディスプレイ・ユニットとクロック発生ユニットでエラーが検出されたことを示します。

DIAGnostic:SELect(?)

DIAGnostic:SELect コマンドは、セルフ・テストの種類を選択します。また、DIAGnostic:SELect? 問合せコマンドは、選択されているセルフ・テストの種類を問い合わせます。

グループ： DIAGNOSTIC

関連コマンド： DIAGnostic?、DIAGnostic:RESUlt、DIAGnostic:STATe

シンタックス： DIAGnostic:SELect {ALL | CPU | DISPlay | FPANel | CLOCk | TRIGger | SMEMory | PMEMory}
DIAGnostic:SELect?

アーギュメント：
ALL — すべてのセルフ・テスト
CPU — CPUユニットのテスト
DISPlay — ディスプレイ・ユニットのテスト
FPANel — フロント・パネルのテスト
CLOCk — クロック発生ユニットのテスト
TRIGger — トリガ信号処理ユニットのテスト
SMEMory — シーケンス・メモリのテスト
PMEMory — パターン・メモリのテスト

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:DIAGNOSTIC:SELect] {ALL | CPU | DISPlay | FPANel | CLOCk | TRIGger | SMEMory | PMEMory}

使用例： 次に、CPU ユニットのテストを選択して実行する例を示します。

```
:DIAG:SELECT  CPU ; STATE  EXECUTE
```

DIAGnostic:STATe

DIAGnostic:STATe コマンドは、DIAGnostic:SELect コマンドで選択されたセルフ・テストを実行します。各セルフ・テストは、実行中にエラーを検出すると実行を中止します。また、すべてのセルフ・テストが選択されている場合には、エラーを検出したテストの実行を中止して、次のテストの実行に移ります。

グループ： DIAGNOSTIC

関連コマンド： DIAGnostic?、DIAGnostic:SELect、DIAGnostic:STATe

シンタックス： DIAGnostic:STATe EXECute

アーギュメント： EXECute

使用例： 次に、すべてのセルフ・テストを選択して実行する例を示します。

```
:DIAG:SELECT  ALL ; STATE  EXECUTE
```

DISPlay?

DISPlay? 問合せコマンドは、画面表示に関する基本的な設定状態を問い合わせます。

グループ： DISPLAY

関連コマンド： なし

シンタックス： DISPlay?

アーギュメント： なし

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:DISPLAY:BRIGHTNESS] <NR2>;[CLOCK] {1 | 0};[DIMMER] {1 | 0};[ENABLE] {1 | 0};
[MENU:NAME] {SETUP | EDIT | APPLICATION | UTILITY};[STATE] {1 | 0};
:DISPLAY:WINDOW:TEXT:DATA ”メッセージ文字列”

使用例： 次に、:DISPlAY? のレスポンス例を示します。

```
:DISPLAY:BRIGHTNESS 0.7;CLOCK 0;DIMMER 1;ENABLE 1;MENU:NAME SETUP;
STATE 1;:DISPLAY:WINDOW:TEXT:DATA ” ”
```

DISPlay:BRIGhtness(?)

DISPlay:BRIGhtness コマンドは、画面の輝度を設定します。また、DISPlay:BRIGhtness? 問合せコマンドは、設定されている画面の輝度を問い合わせます。

グループ： DISPLAY

関連コマンド： DISPlay?

シンタックス：
DISPlay:BRIGhtness <Value>
DISPlay:BRIGhtness?

アーギュメント： <Value> ::= <NRf>　　輝度（ 0 ～ 1 の範囲の実数値）

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:DISPlAY:BRIGHTNESS] <NR2>

使用例： 次に、画面の輝度を 0.7（約 70%）に設定する例を示します。

```
:DISPlAY:BRIGHTNESS  0.7
```

DISPlay:CLOCk(?)

DISPlay:CLOCk コマンドは、画面右上部に日付時刻を表示するかどうか設定します。また、 DISPlay:CLOCk? 問合せコマンドは、画面右上部に日付時刻が表示されているかどうかを問い合わせます。

グループ： DISPLAY

関連コマンド： DISPlay?

シンタックス： DISPlay:CLOCk {ON | OFF | 1 | 0}
DISPlay:CLOCk?

アーギュメント： {ON | 1} — 日付時刻を表示します。
{OFF | 0} — 日付時刻を表示しません。

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:DISPLAY:CLOCK] {1 | 0}

使用例： 次に、日付時刻を表示するように設定する例を示します。

```
:DISPLAY:CLOCK  ON
```

DISPlay:DIMmer(?)

DISPlay:DIMmer コマンドは、画面のディマー動作を行うかどうかを設定します。また、 DISPlay:DIMmer? 問合せコマンドは、画面のディマー動作の ON/OFF 状態を問い合わせます。ディマー動作が ON に設定されている場合、フロント・パネルを操作しないで約 10 分間経過すると、自動的に画面の輝度が下がります。

グループ： DISPLAY

関連コマンド： DISPlay?

シンタックス： DISPlay:DIMmer {ON | OFF | 1 | 0}
DISPlay:DIMmer?

アーギュメント： {ON | 1} — ディマー動作を行います。
{OFF | 0} — ディマー動作を行いません。

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:DISPLAY:DIMMER] {1 | 0}

使用例： 次に、ディマー動作を ON に設定する例を示します。

```
:DISPLAY:DIMMER  ON
```

DISPlay:ENABle(?)

DISPlay:ENABle コマンドは、ディスプレイを ON または OFF にします。セキュリティが ON の場合はディスプレイを OFF にした後で再び ON にすることはできません。また、DISPlay:ENABle? 問合せコマンドは、ディスプレイの ON/OFF 状態を問い合わせます。

グループ： DISPLAY

関連コマンド： DISPlay?、DISPlay:MENU:STATe

シンタックス： DISPlay:ENABle {ON | OFF | 1 | 0}
DISPlay:ENABle?

アーギュメント： {ON | 1} — ディスプレイを ON にします。
{OFF | 0} — ディスプレイを OFF にします。

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:DISPLAY:ENABLE] {1 | 0}

使用例： 次に、ディスプレイを OFF に設定する例を示します。

```
:DISPLAY:ENABLE  OFF
```

DISPlay:MENU?

DISPlay:MENU? 問合せコマンドは、選択されているメニューの種類と表示状態を問い合わせます。

グループ： DISPLAY

関連コマンド： DISPlay?、DISPlay:MENU[:NAME]、DISPlay:MENU:NAME?

シンタックス： DISPlay:MENU?

アーギュメント： なし

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:DISPLAY:MENU:NAME] {SETUP | EDIT | APPLICATION | UTILITY};[STATE] {1 | 0}

使用例： 次に、SETUP メニューが画面に表示されているときの :DISPLAY:MENU? のレスポンス例を示します。

```
:DISPLAY:MENU:NAME  SETUP ; STATE  1
```

DISPlay:MENU[:NAME]

DISPlay:MENU[:NAME] コマンドは、画面に表示するメニューを選択します。

グループ： DISPLAY

関連コマンド： DISPlay?、DISPlay:MENU?、DISPlay:MENU:NAME?

シンタックス： DISPlay:MENU[:NAME] {SETUp | EDIT | APPLication | UTILity}

アーギュメント：
SETUp — セットアップ・メニュー
EDIT — エディット・メニュー
APPLication — アプリケーション・メニュー
UTILity — ユーティリティ・メニュー

使用例： 次に、UTILITY メニューを選択する例を示します。

```
:DISPLAY:MENU:NAME  UTILITY
```

DISPlay:MENU:NAME?

DISPlay:MENU:NAME? 問合せコマンドは、選択されているメニューの種類を問い合わせます。

グループ： DISPLAY

関連コマンド： DISPlay?、DISPlay:MENU?、DISPlay:MENU:[:NAME]

シンタックス： DISPlay:MENU:NAME?

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:DISPLAY:MENU:NAME] {SETUP | EDIT | APPLICATION | UTILITY}

使用例： 次に、EDIT メニューが選択されているときの :DISPLAY:MENU:NAME? のレスポンス例を示します。

```
:DISPLAY:MENU:NAME  EDIT
```

DISPlay:MENU:STATe(?)

DISPlay:MENU:STATe コマンドは、画面にメニューを表示するかどうかを設定します。また、DISPlay:MENU:STATe? 問合せコマンドは、画面にメニューが表示されているかどうかを問い合わせます。このコマンドは、DISPlay:ENABle コマンドと同等です。

グループ： DISPLAY

関連コマンド： DISPlay?、DISPlay:ENABle、DISPlay:MENU?、DISPlay:MENU:[:NAME]

シンタックス： DISPlay:MENU:STATe {ON | OFF | 1 | 0}
DISPlay:MENU:STATe?

アーギュメント： {ON | 1} — メニューを表示します。
{OFF | 0} — メニューを表示しません。

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:DISPLAY:MENU:STATE] {1 | 0}

使用例： 次に、画面にメニューが表示されるように設定する例を示します。

```
:DISPLAY:MENU:STATE  ON
```

DISPlay[:WINDow]:TEXT:CLEar

DISPlay[:WINDow]:TEXT:CLEar コマンドは、画面のメッセージ表示エリアをクリアします。

グループ： DISPLAY

関連コマンド： DISPlay?、DISPlay[:WINDow]:TEXT[:DATA]

シンタックス： DISPlay[:WINDow]:TEXT:CLEar

アーギュメント： なし

使用例： 次に、メッセージ表示エリアをクリアする例を示します。

```
:DISPLAY:WINDOW:TEXT:CLEAR
```

DISPlay[:WINDow]:TEXT[:DATA](?)

DISPlay[:WINDow]:TEXT[:DATA] コマンドは、画面のメッセージ表示エリアに表示するメッセージを入力します。入力されたメッセージは、ただちに表示されます。また、DISPlay[:WINDow]:TEXT[:DATA]? 問合せコマンドは、入力されているスクリーン・メッセージの内容を問い合わせます。

注： メッセージ表示エリアの内容は自動的にスクロールします。表示内容を完全に更新したい場合は、DISPlay[:WINDow]:TEXT:CLEar コマンドでメッセージ表示エリアを前もってクリアしてください。

グループ： DISPLAY

関連コマンド： DISPlay?、DISPlay[:WINDow]:TEXT:CLEar

シンタックス：
DISPlay[:WINDow]:TEXT[:DATA] <Message>
DISPlay[:WINDow]:TEXT[:DATA]?

アーギュメント： <Message> ::= <string>　　メッセージ文字列

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:DISPLAY:WINDOW:TEXT:DATA] <Message>

使用例： 次に、メッセージ表示エリアにテキスト ”ABCD” を入力する例を示します。

```
:DISPLAY:WINDOW:TEXT:DATA  ”ABCD”
```

*ESE(?)

*ESE 共通コマンドは、ステータス／イベント・レポーティング・システムで使用する ESER（ Event Status Enable Register ）に値を設定します。また、*ESE? 共通・問合せコマンドは、ESER の内容を問い合わせます。.

グループ： その他

関連コマンド： *CLS、DESE、*ESR?、EVENT?、EVMsg?、EVQty?、*SRE、*STB?

シンタックス： *ESE <Bit Value>
*ESE?

アーギュメント： <Bit Value> ::= <NR1> （ 設定範囲: 0 ～ 255 ）

電源投入時の ESER の値は、PSC フラグが TRUE の場合、すべてのビットがリセットされます。PSC フラグが FALSE の場合、電源の ON/OFF とは無関係に、ESER の値が保持されます。PSC フラグについては、*PSC コマンドの項をご参照ください。

使用例： 次に、ESER を 177（ 10110001 ）に設定する例を示します。この場合、PON 、CME 、EXE 、および OPC の各ビットがセットされます。

```
*ESE 177
```

次に、*ESE? のレスポンス例を示します。

```
176
```

この場合 ESER の内容は、10110000 になります。

*ESR?

*ESR? 共通・問合せコマンドは、ステータス／イベント・レポーティング・システムで使用する SESR（ Standard Event Status Register ）の内容を問い合わせます。

グループ： その他

関連コマンド： *CLS、DESE、*ESE?、EVENT?、EVMsg?、EVQty?、*SRE、*STB?

シンタックス： *ESR?

アーギュメント： なし

使用例： 次に、*ESR? のレスポンス例を示します。

```
181
```

この場合 SESR の内容は、10110101 になります。

EVENT?

EVENT? 問合せコマンドは、イベント・キューから、取り出し可能なイベントのうち最も古いイベントのコードを取り出します。イベント・キューのイベントは、*ESR? 共通・問合せコマンドによって取り出し可能となります。詳しくは、第 3 章の“ステータス／イベント・レポーティング・システム”をご参照ください。

グループ： その他

関連コマンド： *CLS、DESE、*ESE、*ESR?、EVMsg?、EVQty?、*SRE、*STB?

シンタックス： EVENT?

アーギュメント： なし

使用例： 次に、:EVENT? のレスポンス例を示します。

```
:EVENT  113
```

EVMsg?

EVMsg? 問合せコマンドは、イベント・キューから、取り出し可能なイベントのうち、最も古いイベントのコードと、そのコードに対応するメッセージを取り出します。イベント・キューのイベントは、*ESR? 共通・問合せコマンドによって取り出し可能になります。詳しくは、第 3 章の“ステータス／イベント・レポーティング・システム”をご参照ください。

グループ： その他

関連コマンド： *CLS、DESE、*ESE、*ESR?、EVENT?、EVQty?、*SRE、*STB?

シンタックス： EVMsg?

使用例： 次に、:EVMSG? のレスポンス例を示します。

```
:EVMSG 420,”Query UNTERMINATED”
```

EVQty?

EVQty? 問合せコマンドは、イベント・キューにスタックされているイベント数を問い合わせます。

グループ： その他

関連コマンド： *CLS、DESE、*ESE、*ESR?、EVENT?、EVMsg?、*SRE、*STB?

シンタックス： EVQty?

アーギュメント： なし

レスポンス： 使用例をご参照ください。

使用例： 次に、:EVQTY? に対するレスポンス例を示します。

```
:EVQTY 5
```

FACTory

FACTory コマンドは、本機器を、工場出荷時のデフォルト設定状態に戻します。デフォルト設定値については、「付録 C　デフォルト設定値」をご参照ください。

グループ： その他

関連コマンド： *RST、SYSTem:SECurity:IMMediate

シンタックス： FACTory

アーギュメント： なし

使用例： 次に、本機器を、工場出荷時のデフォルト設定状態に戻す例を示します。

```
:FACTORY
```

HCOPy?

HCOPy? 問合せコマンドは、設定されているハードコピーのイメージ・データ・フォーマットおよび出力ポートを問い合わせます。

グループ： HARDCOPY

関連コマンド： HCOPy:FORMat、HCOPy:PORT

シンタックス： HCOPy?

アーギュメント： なし

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:HCOPY:FORMAT] {BMP | EPSON | EPSMONO | THINKJET | TIFF};[:PORT] {DISK | GPIB | RS232C}

ここで

BMP — Windows のモノクロ・イメージ・ファイルのフォーマットです。
EPSON — ESC/P グラフィック・モードの 9 ピンおよび 24 ピン・ドット・マトリックス・プリンタ用のフォーマットです。
EPSMONO — Encapsulated Postscript 形式のモノクロ・イメージ・ファイルのフォーマットです。
THINKJET— HP のインクジェット・プリンタ用のフォーマットです。
TIFF — TIFF フォーマットです。

DISK — フロッピ・ディスク上にファイルとして出力します。
GPIB — GPIB ポートへ出力します。
RS232C — RS-232C ポートへ出力します。

使用例： 次に、:HCOPY? のレスポンス例を示します。

```
:HCOPY:FORMAT  TIFF ; PORT  DISK
```

この場合、ハードコピー・データが TIFF フォーマットでフロッピ・ディスク上にファイルとして出力されるよう設定されています。

HCOPy:ABORt

HCOPy:ABORt コマンドは、ハードコピー出力を中止します。

グループ： HARDCOPY

関連コマンド： HCOPy:STARt

シンタックス： HCOPy:ABORt

アーギュメント： なし

使用例： 次に、ハードコピー出力を中止する例を示します。

```
:HCOPY:ABORT
```

HCOPy:DATA?

HCOPy:DATA? 問合せコマンドは、ハードコピーのイメージ・データを出力キューに出力します。

グループ： HARDCOPY

関連コマンド： なし

シンタックス： HCOPy:DATA?

アーギュメント： なし

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:HCOPY:DATA] <Image>

ここで

<Image> ::= <block>　　ハードコピーのイメージ・データのブロック

使用例： 次に、ハードコピーのイメージ・データを出力キューに出力する例を示します。

```
:HCOPY:DATA?
```

HCOPy:FORMat(?)

HCOPy:FORMat コマンドは、ハードコピーのイメージ・データ・フォーマットを設定します。また、HCOPy:FORMat? 問合せコマンドは、設定されているハードコピーのイメージ・データ・フォーマットを問い合わせます。

グループ： HARDCOPY

関連コマンド： HCOPy?

シンタックス：
HCOPy:FORMat {BMP | EPSOn | EPSMono | THINkjet | TIFF}
HCOPy:FORMat?

アーギュメント：
BMP — Windows のモノクロ・イメージ・ファイルのフォーマットです。
EPSOn — ESC/P グラフィック・モードの 9 ピンおよび 24 ピン・ドット・マトリクス・プリンタ用のフォーマットです。
EPSMono — Encapsulated Postscript 形式のモノクロ・イメージ・ファイルのフォーマットです。
THINkjet — HP のインクジェット・プリンタ用のフォーマットです。
TIFF — TIFF フォーマットです。

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:HCOPY:FORMAT] {BMP | EPSON | EPSMONO | THINKJET | TIFF}

使用例： 次に、ハードコピーのイメージ・データが TIFF フォーマットで出力されるように設定する例を示します。

```
:HCOPY:FORMAT  TIFF
```

HCOPy:PORT(?)

HCOPy:PORT コマンドは、ハードコピーの出力ポートを設定します。また、HCOPy:PORT? 問合せコマンドは、設定されているハードコピーの出力ポートを問い合わせます。

グループ： HARDCOPY

関連コマンド： HCOPy?

シンタックス： HCOPy:PORT {DISK | GPIB | RS232c}
HCOPy:PORT?

アーギュメント： DISK — フロッピ・ディスク上にファイルとして出力します。
GPIB — GPIB ポートへ出力します。
RS232c — RS-232C ポートへ出力します。

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:HCOPY:PORT] {DISK | GPIB | RS232C}

使用例： 次に、ハードコピーのイメージ・データが RS-232C ポートへ出力されるように設定する例を示します。

```
:HCOPY:PORT  RS232C
```

HCOPy:STARt

HCOPy:STARt コマンドは、ハードコピー出力を開始させます。

グループ： HARDCOPY

関連コマンド： HCOPy:ABORt

シンタックス： HCOPy:STARt

アーギュメント： なし

使用例： 次に、ハードコピー出力を開始させる例を示します。

```
:HCOPY:START
```

HEADer(?)

HEADer コマンドは、IEEE std. 488.2 の共通コマンドを除くすべての問合せコマンドに対するレスポンスに、コマンド・ヘッダを含めるかどうかを設定します。また、HEADer? 問合せコマンドは、レスポンス・メッセージにコマンド・ヘッダが含まれるかどうかを問い合わせます。

グループ： その他

関連コマンド： VERBose

シンタックス： HEADer {ON | OFF | <NR1>}
HEADer?

アーギュメント： ON またはゼロ以外の値 — レスポンスにコマンド・ヘッダを含めます。
OFF またはゼロ — レスポンスからコマンド・ヘッダを除きます。

レスポンス： 問合せコマンドに対するレスポンスは次のとおりです。
1 — コマンド・ヘッダが含まれます。
0 — コマンド・ヘッダは含まれません。

具体例については、表 2-3 をご参照ください。

使用例： 次に、レスポンス中にヘッダを含める設定例を示します。

```
:HEADER ON
```

次に、:HEADER? に対するレスポンス例を示します。

```
:HEADER 1
```

この場合、レスポンス中にヘッダが含まれることを示しています。

ID?

ID? 問合せコマンドは、本機器の ID 情報を問い合わせます。特にソフトウェア・バージョン（ファームウェア・バージョン）を確認する場合には、このコマンド、または *IDN? 共通・問合せコマンドを使用できます。

グループ： その他

関連コマンド： *IDN?

シンタックス： ID?

アーギュメント： なし

レスポンス： レスポンス・フォーマットは、次のとおりです。

ID <Manufacturer>/<Model>, <Firmware Level>

ここで

<Manufacturer> ::= SONY_TEK
<Model> ::= DG2030
<Firmware Level> ::= CF:<Code and Format Version>, FV:<Firmware Version>

使用例： 次に、:ID? のレスポンス例を示します。

```
:ID SONY_TEK/DG2030,CF:91.1CN,FV:1.00
```

*IDN?

*IDN? 共通・問合せコマンドは、本機器の ID 情報を問い合わせます。特にソフトウェア・バージョン（ファームウェア・バージョン）を確認する場合には、本コマンド、または ID? 問合せコマンドを使用できます。

グループ： その他

関連コマンド： ID?

シンタックス： *IDN?

アーギュメント： なし

レスポンス： レスポンス・フォーマットは、次のとおりです。

<Manufacturer>，<Model>，<Serial Number>，<Firmware Level>

ここで

<Manufacturer> ::= SONY/TEK
<Model> ::= DG2030
<Serial Number> ::= 0
<Firmware Level> ::= CF:<Code and Format Version><sp>FV:<Firmware Version>
<sp> ::= 空白

使用例： 次に、*IDN? のレスポンス例を示します。

:SONY/TEK,DG2030,0,CF:91.1CN FV:1.00

LOCk(?)

LOCk コマンドは、フロント・パネル上のキーやノブの機能をロックしたり、ロックを解除したりします。また、LOCk? 問合せコマンドは、フロント・パネル上のキーやノブがロックされているかどうかを問い合わせます。

本機器にはリモート・コントロール／ローカル・コントロールの切り替えはなく、外部コントローラからもフロント・パネルからも同時に設定が行えます。外部コントローラで操作中に、あるいは、外部コントローラでソフトウェアを実行中にフロント・パネルからの操作を禁止したい場合には、このコマンドで、フロント・パネルのキーおよびノブの機能をロックしてください。

注： LOCk コマンドによってフロント・パネル・コントロールがロック状態になると、本機器の管面右上部に ” FP: LOCKED ” の文字が表示されます。

グループ： その他

関連コマンド： UNLock

シンタックス：
LOCk {ALL | NONe}
LOCk?

アーギュメント：
ALL — フロント・パネルのキーおよびノブの機能をロックします。
NONe — フロント・パネルのキーおよびノブのロックを解除します。

使用例： 次に、キーおよびノブをロックする例を示します。

```
:LOCK ALL
```

MMEMory:CATalog[:ALL]?

MMEMory:CATalog[:ALL]? 問合せコマンドは、ディスクのカレント・ディレクトリに存在するファイル／ディレクトリの情報を問い合わせます。

グループ： MEMORY

関連コマンド： MMEMory:CATalog:ORDer

シンタックス： MMEMory:CATalog[:ALL]?

アーギュメント： なし

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:MMEMORY:CATALOG:ALL] <Fileinfo>[,<Fileinfo>]

ここで

<Fileinfo> ::= <ファイル名>,<サイズ>,<日時>
<ファイル名> ::= <文字列>
<サイズ> ::= <NR1>
<日時> ::= <文字列>

注： サブディレクトリのファイルサイズは、0 になります。

MMEMory:CATalog:ORDer(?)

MMEMory:CATalog:ORDer コマンドは、ディスクのディレクトリ・リストに記録するファイル情報の順序を設定します。また、MMEMory:CATalog:ORDer? 問合せコマンドは、ディスクのディレクトリ・リストに記録するファイル情報の順序を問い合わせます。

グループ： MEMORY

関連コマンド： MMEMory:CATalog[:ALL]?

シンタックス： MMEMory:CATalog:ORDer {NAME1 | NAME2 | TIME1 | TIME2}
MMEMory:CATalog:ORDer?

アーギュメント： NAME1 — 名前のアルファベット順
NAME2 — NAME1 の逆順
TIME1 — 作成時刻の古い順
TIME2 — 作成時刻の新しい順

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:MMEMORY:CATALOG:ORDER] {NAME1 | NAME2 | TIME1 | TIME2}

使用例： 次に、ディスクのディレクトリ・リストに記録するファイル情報の順序を名前のアルファベット順に設定する例を示します。

```
:MMEMORY:CATALOG:ORDER  NAME1
```

MMEMory:CDIRectory(?)

MMEMory:CDIRectory コマンドは、カレント・ワーキング・ディレクトリを変更します。また、MMEMory:CDIRectory? 問合せコマンドは、カレント・ワーキング・ディレクトリを問い合わせます。

グループ： MEMORY

関連コマンド： MMEMory:MDIRectory

シンタックス： MMEMory:CDIRectory <Pathname>
MMEMory:CDIRectory?

アーギュメント： <Pathname> ::= <文字列>　　ディレクトリを示す文字列

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:MMEMORY:CDIRECTORY] <Pathname>

使用例： 次に、\DG\WORK3 を新しいカレント・ワーキング・ディレクトリにする例を示します。

```
:MMEMORY:CDIRECTORY  "\DG\WORK3"
```

MMEMory:COPY

MMEMory:COPY コマンドは、ディスク上のファイルをコピーし、新規にファイルを作成します。コピー先のファイルが既に存在すれば、上書きしないでエラーになります。

グループ： MEMORY

関連コマンド： MMEMory:DELete:ALL、MMEMory:DELete[:NAME]

シンタックス： MMEMory:COPY <Path1>,<Path2>

アーギュメント： <Path1> ::= <文字列>　　コピー元のファイルのパス名
<Path2> ::= <文字列>　　コピー先のファイルのパス名

使用例： 次に、カレント・ワーキング・ディレクトリの MYDATA.PDA というファイルをコピーし、MYWORK.PDA というファイルをカレント・ワーキング・ディレクトリに作成する例を示します。

```
:MMEMORY:COPY  ”MYDATA.PDA”,”MYWORK.PDA”
```

MMEMory:DELete:ALL

MMEMory:DELete:ALL コマンドは、ディスク上のカレント・ディレクトリのファイルとサブディレクトリをすべて削除します。ただし、空でないサブディレクトリは削除しません。

グループ： MEMORY

関連コマンド： MMEMory:DELete[:NAME]

シンタックス： MMEMory:DELete:ALL

アーギュメント： なし

MMEMory:DELete[:NAME]

MMEMory:DELete[:NAME] コマンドは、指定されたパス名のファイルまたはサブディレクトリを削除します。ただし、空でないサブディレクトリは削除しません。

グループ： MEMORY

関連コマンド： MMEMory:DELete:ALL

シンタックス： MMEMory:DELete[:NAME] <Pathname>

アーギュメント： <Pathname> ::= <文字列>　　ファイルのパス名

使用例： 次に、カレント・ワーキング・ディレクトリの NOMORE.PDA というファイルを削除する例を示します。

```
:MMEMORY:DELETE  ”NOMORE.PDA”
```

MMEMory:FREE?

MMEMory:FREE? 問合せコマンドは、ディスクの使用状況を問い合わせます。

グループ： MEMORY

関連コマンド： なし

シンタックス： MMEMory:FREE?

アーギュメント： なし

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:MMEMORY:FREE] <Used>,<Unused>

ここで

<Used> ::= <NR1>　　使用済領域のバイト数
<Unused> ::= <NR1>　　未使用領域のバイト数

使用例： 次に、:MMEMORY:FREE? コマンドのレスポンス例を示します。

```
:MMEMORY:FREE  104584,1352704
```

MMEMory:INITialize

MMEMory:INITialize コマンドは、ディスクを初期化します。フォーマット・タイプはアーギュメントで指定します。

グループ： MEMORY

関連コマンド： なし

シンタックス： MMEMory:INITialize {DD1 | DD2 | HD1 | HD2 | HD3}

アーギュメント： フォーマット・タイプは、次のとおりです。

アーギュメント	内　容
DD1	2DD、720 KB、80 トラック、9 セクタ／トラック、512 バイト／セクタ。IBM PC 2DD、東芝 J3100 用 2DD フォーマット
DD2	2DD、640 KB、80 トラック、8 セクタ／トラック、512 バイト／セクタ。NEC PC－9800 2DD 用フォーマット
HD1	2HD、1,232 KB、77 トラック、15 セクタ／トラック、1,024 バイト／セクタ。NEC PC－9800 用 2HD フォーマット
HD2	2HD、1,200 KB、80 トラック、15 セクタ／トラック、512 バイト／セクタ。東芝 J3100 用 2HD フォーマット
HD3	2HD、1,440 KB、80 トラック、18 セクタ／トラック、512 バイト／セクタ。IBM PC 用 2HD フォーマット

使用例： 次に、フロッピ・ディスクを IBM PC 用 2HD フォーマットで初期化する例を示します。

```
:MMEMORY:INITIALIZE  HD3
```

MMEMory:LOAD

MMEMory:LOAD コマンドは、マス・メモリから本機器の内部メモリに DG2030 型オリジナル・フォーマットのパターン・データ、ブロック、グループ、シーケンス、およびセットアップ情報を読み込みます。

グループ： MEMORY

関連コマンド： MMEMory:SAVE

シンタックス： MMEMory:LOAD <Pathname>

アーギュメント： <Pathname> ::= <文字列>　　ファイルの名称

使用例： 次に、カレント・ワーキング・ディレクトリの MYDATA.PDA というファイルから内部メモリにすべての情報を読み込む例を示します。

```
:MMEMORY:LOAD  ”MYDATA.PDA”
```

MMEMory:LOCK(?)

MMEMory:LOCK コマンドは、ファイルをロックしたり、ロックを解除したりします。ファイルがロックされると、削除や書き込みができなくなります。また、MMEMory:LOCK? 問合せコマンドは、ファイルがロックされているかどうかを問い合わせます。

グループ： MEMORY

関連コマンド： なし

シンタックス： MMEMory:LOCK <Pathname>,<Flag>
MMEMory:LOCK? <Pathname>

アーギュメント： <Pathname> ::= <文字列>　設定または問合わせの対象となるファイルのパス名
<Flag> ::= {ON | OFF | 1 | 0}
{ON | 1}　— ファイルをロックします。
{OFF | 0}　— ファイルのロックを解除します。

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:MMEMORY:LOCK] {1 | 0}

使用例： 次に、カレント・ワーキング・ディレクトリの COUNT1.PDA というファイルをロックする例を示します。

```
:MMEMORY:LOCK  ”COUNT1.PDA”,ON
```

MMEMory:MDIRectory

MMEMory:MDIRectory コマンドは、新しいサブディレクトリを作成します。すでに同名のサブディレクトリが存在するときは、無効になります。

グループ： MEMORY

関連コマンド： MMEMory:CDIRectory、MMEMory:RDIRectory

シンタックス： MMEMory:MDIRectory <Pathname>

アーギュメント： <Pathname> ::= <文字列>　作成するサブディレクトリのパス名

使用例： 次に、カレント・ワーキング・ディレクトリに WORK4 というサブディレクトリを作成する例を示します。

```
:MMEMORY:MDIRECTORY  ”WORK4”
```

MMEMory:RDIRectory

MMEMory:RDIRectory コマンドは、サブディレクトリを削除します。サブディレクトリ内にファイルが存在するときは、削除は実行されません。

グループ： MEMORY

関連コマンド： MMEMory:CDIRectory、MMEMory:MDIRectory

シンタックス： MMEMory:RDIRectory <Pathname>

アーギュメント： <Pathname> ::= <文字列>　　削除するサブディレクトリのパス名

使用例： 次に、カレント・ワーキング・ディレクトリの WORK4 というサブディレクトリを削除する例を示します。

```
:MMEMORY:RDIRECTORY ”WORK4”
```

MMEMory:REName

MMEMory:REName コマンドは、指定ファイルのファイル名を変更します。

グループ： MEMORY

関連コマンド： MMEMory:COPY

シンタックス： MMEMory:REName <Oldname>,<Newname>

アーギュメント： <Oldname> ::= <文字列>　　現在のファイル名
<Newname> ::= <文字列>　　新しいファイル名

使用例： 次に、カレント・ワーキング・ディレクトリ内でファイル名を COUNT1.PDA から COUNT2.PDA に変更する例を示します。

```
:MMEMORY:RENAME ”COUNT1.PDA”,”COUNT2.PDA”
```

MMEMory:SAVE

MMEMory:SAVE コマンドは、本機器の内部メモリからマス・メモリにパターン・データ、ブロック、グループ、シーケンス、およびセットアップ情報を DG2030 型オリジナル・フォーマットでセーブします。

グループ： MEMORY

関連コマンド： MMEMory:LOAD

シンタックス： MMEMory:SAVE <Pathname>

アーギュメント： <Pathname> ::= <文字列>　　ファイルのパス名

使用例： 次に、内部メモリからカレント・ワーキング・ディレクトリの NEWDATA.PDA というファイルにすべての情報をセーブする例を示します。

```
:MMEMORY:SAVE  ”NEWDATA.PDA”
```

MODE?

MODE? 問合せコマンドは、パターン発生のモードに関するすべての設定状態を問い合わせます。

グループ： MODE

関連コマンド： MODE:STATe、MODE:UPDate

シンタックス： MODE?

アーギュメント： なし

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:MODE:STATE] {REPEAT | SINGLE | STEP | ENHANCED};[UPDATE] {AUTO | MANUAL}

使用例： 次に、:MODE? のレスポンス例を示します。

```
:MODE:STATE  REPEAT ; UPDATE  AUTO
```

この場合、ラン・モードが Repeat に設定されていて、出力パターンの更新方法が Auto に設定されていることを示します（MODE:STATe コマンドおよび MODE:UPDate コマンドの項を参照）。

MODE:STATe(?)

MODE:STATe コマンドは、パターン発生のラン・モードを設定します。また、MODE:STATe? 問合せコマンドは、設定されているパターン発生のラン・モードを問い合わせます。

グループ： MODE

関連コマンド： MODE?

シンタックス： MODE:STATe {REPeat | SINGle | STEp | ENHanced}
MODE:STATe?

アーギュメント：
REPeat — パターン・データ出力を繰り返し行います。
SINGle — パターン・データ出力を 1 回だけ行います。
STEp — 内部クロックを用いずに STEP キーでクロックを生成してパターン・データ出力を行います。
ENHanced — シーケンスに定義したとおりにパターン・データ出力を行います。

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:MODE:STATE] {REPEAT | SINGLE | STEP | ENHANCED}

使用例： 次に、ラン・モードを Single に設定する例を示します。

```
:MODE:STATE  SINGLE
```

MODE:UPDate(?)

MODE:UPDate コマンドは、パターン発生に関連するデータが変更された場合の、出力パターンの更新方法を設定します。また、MODE:UPDate? 問合せコマンドは、パターン発生に関連するデータが変更された場合の、出力パターンの更新方法を問い合わせます。

グループ： MODE

関連コマンド： MODE:STATe、MODE:UPDate

シンタックス： MODE:UPDate {AUTO | MANual}
MODE:UPDate?

アーギュメント：
AUTO — データに何らかの変更があるたびに、パターン出力に反映させます。
MANual — データに変更があっても強制的に更新するコマンドを受け取るまで、パターン出力を更新しません。

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:MODE:UPDATE] {AUTO | MANUAL}

使用例： 次に、出力パターンの更新方法を Auto に設定する例を示します。

```
:MODE:UPDATE  AUTO
```

*OPC(?)

*OPC共通コマンドは、ペンディング中のすべての操作が終了した場合、SESR（Standard Event Status Register）のビット 0 をセットします。また、*OPC? 共通・問合せコマンドは、ペンディング中のすべての操作が終了した場合、ASCII コード ”1” を返します。

グループ： その他

関連コマンド： *WAI

シンタックス： *OPC
*OPC?

アーギュメント： なし

使用例： 次に、ハードコピーの終了を待つ例を示します。

```
:HCOPY:PORT DISK;HCOPY START;*OPC
```

*OPT?

*OPT共通・問合せコマンドは、本機器に組み込まれたオプション情報を問い合わせます。

グループ： その他

関連コマンド： なし

シンタックス： *OPT?

アーギュメント： なし

レスポンス： レスポンス・フォーマットは、次のとおりです。

<Option>［, <Option>]...

次のオプションが認識されます。

0 — オプションが組み込まれていません。
CH4–CH7 — オプション 01 型が組み込まれています。

使用例： 次に、*OPT? のレスポンス例を示します。

```
CH4–CH7
```

この場合、オプション 01 型が組み込まれていることを示します。

OUTPut?

OUTPut? 問合せコマンドは、データ出力、クロック出力に関するすべての設定を問い合わせます。

グループ： OUTPUT

関連コマンド： DATA?

シンタックス： OUTPut?

アーギュメント： なし

使用例： 次に、:OUTPUT? に対するレスポンス例を示します。

```
:OUTPUT:ELEVEL 0.5;ILEVEL 0.5;CH0:HIGH 0.500;LOW –0.500;DELAY 0.00E–0.9;
INHIBIT 0;ASSIGN 0;RISE 0.00E–9;FALL 0.00E–9;DESKEW 0.00E–09;
:OUTPUT:CH1:HIGH 0.500;LOW –0.500;DELAY 0.00E–09;INHIBIT 0
;ASSIGN 1;RISE 0.00E–9;FALL 0.00E–9;DESKEW 0.00E–09;
:OUTPUT:CH2:HIGH 0.500;LOW –0.500;DELAY 0.00E–09;INHIBIT 0
;ASSIGN 1;RISE 0.00E–9;FALL 0.00E–9;DESKEW 0.00E–09;
:OUTPUT:CH3:HIGH 0.500;LOW –0.500;DELAY 0.00E–09;INHIBIT 0
;ASSIGN 1;RISE 0.00E–9;FALL 0.00E–9;DESKEW 0.00E–09;
:OUTPUT:CH4:HIGH 0.500;LOW –0.500;DELAY 0.00E–09;INHIBIT 0
;ASSIGN 1;RISE 0.00E–9;FALL 0.00E–9;DESKEW 0.00E–09;
:OUTPUT:CH5:HIGH 0.500;LOW –0.500;DELAY 0.00E–09;INHIBIT 0
;ASSIGN 1;RISE 0.00E–9;FALL 0.00E–9;DESKEW 0.00E–09;
:OUTPUT:CH6:HIGH 0.500;LOW –0.500;DELAY 0.00E–09;INHIBIT 0
;ASSIGN 1;RISE 0.00E–9;FALL 0.00E–9;DESKEW 0.00E–09;
:OUTPUT:CH7:HIGH 0.500;LOW –0.500;DELAY 0.00E–09;INHIBIT 0
;ASSIGN 1;RISE 0.00E–9;FALL 0.00E–9;DESKEW 0.00E–09;
:OUTPUT:CHCLK:HIGH 0.500;LOW –0.500;INHIBIT 0;RISE 0.00E–9;FALL 0.00E–9;
:OUTPUT:DEFINE #2470,0,0<LF>1,1,0<LF>2,2,0<LF>3,3,0<LF>4,4,0<LF>5,5,0<LF>
6,6,0<LF>7,7,0
```

OUTPut:CH<n>:ASSIGn(?)

OUTPut:CH<n>:ASSIGn コマンドは、ヘッダで指定するチャンネルにデータ・ビットを割り当てます。また、OUTPut:CH<n>:ASSIGn? 問合せコマンドは、指定チャンネルに割り当てられているデータ・ビットを問い合わせます。チャンネルにデータ・ビットが割り当てられていない場合、レスポンスのビット番号として –1 が返されます。

グループ： OUTPUT

関連コマンド： OUTPut:CH<n>:RELEase、OUTPut:DEFine

シンタックス：
OUTPut:CH<n>:ASSIGn <Bit>
OUTPut:CH<n>:ASSIGn?
（<n>:0～7）

アーギュメント： <Bit> ::= <NR1>　　データ・ビットの番号（ 0 ～ 7 ）

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:OUTPUT:CH<n>:ASSIGN] <Bit>

使用例： 次に、チャンネル 1 にデータ・ビット D03 を割り当てる例を示します。

```
:OUTPUT:CH1:ASSIGN  3
```

OUTPut:CH<n>:DELAy(?)

OUTPut:CH<n>:DELAy コマンドは、ヘッダで指定するチャンネルのディレイ時間を設定します。また、OUTPut:CH<n>:DELAy? 問合せコマンドは、指定チャンネルのディレイ時間の設定値を問い合わせます。

グループ： OUTPUT

関連コマンド： なし

シンタックス：
OUTPut:CH<n>:DELAy <Time>
OUTPut:CH<n>:DELAy?

アーギュメント： <Time> ::= <NR2>[<Unit>]　　ディレイ時間
<Unit> ::= {s | ms | us | ns}

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:OUTPUT:CH<n>:DELAY] <NR3>

使用例： 次に、チャンネル 2 のディレイ時間を 10 ns に設定する例を示します。

```
:OUTPUT:CH2:DELAY 10 ns
```

OUTPut:CH<n>:DESKew(?)

OUTPut:CH<n>:DESKew コマンドは、ヘッダで指定するチャンネルのスキュー調整値を設定します。また、OUTPut:CH<n>:DESKew? 問い合せコマンドは、指定チャンネルのスキュー調整値を問い合わせます。

グループ： OUTPUT

関連コマンド： OUTPut:CH<n>:DESKew:RESET

シンタックス：
OUTPut:CH<n>:DESKew <Time>
OUTPut:CH<n>:DESKew?

アーギュメント：
<Time> ::= <NR2>[<Unit>]　　スキュー調整値
<Unit> ::= {s | ms | us | ns]

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:OUTPUT:CH<n>:DESKEW] <NR3>

使用例： 次に、チャンネル 1 のスキュー調整値を 2ns に設定する例を示します。

```
:OUTPUT:CH1:DESKEW 2 ns
```

OUTPut:CH<n>:DESKew:RESET

OUTPut:CH<n>:DESKew:RESET コマンドは、ヘッダで指定するチャンネルのスキュー調整値をリセットします。

グループ： OUTPUT

関連コマンド： OUTPut:CH<n>:DESKew

シンタックス： OUTPut:CH<n>:DESKew:RESET

アーギュメント： なし

使用例： 次に、チャンネル 2 のスキュー調整値をリセットする例を示します。

```
:OUTPUT:CH2:DESKEW:RESET
```

OUTPut:CH<n>:FALl(?)

OUTPut:CH<n>:FALl コマンドは、ヘッダで指定するチャンネルの立ち下がり時間を設定します。また、OUTPut:CH<n>:FALl? 問い合わせコマンドは、指定したチャンネルの立ち下がり時間を問い合わせます。なお、アーギュメントで FAST または 0.0 を指定した場合は、立ち下がり時間が限りなく速くなります。

グループ： OUTPUT

関連コマンド： OUTPut:CH<n>:FALl?RANge、OUTPut:CH<n>:FALl?VALid

シンタックス：
OUTPut:CH<n>:FALl <Time>
OUTPut:CH<n>:FALl?

アーギュメント：
<Time> ::= <NR2>[<Unit>] | FAST
<Unit> ::= {s | ms | us | ns}

レスポンス： レスポンス・フォーマットは、次のとおりです。FAST の場合は、0.0 が返されます。

[:OUTPUT:CH<n>:FALL] <NR3>

使用例： 次に、チャンネル 1 の立ち下がり時間を1 ns に設定する例を示します。

```
:OUTPUT:CH1:FALL 1ns
```

OUTPut:CH<n>:FALl? RANge

OUTPut:CH<n>:FALl? RANge 問い合せコマンドは、指定したチャンネルの立ち下がり時間の有効設定範囲を問い合わせます。

グループ： OUTPUT

関連コマンド： OUTPut:CH<n>:FALl?、OUTPut:CH<n>:FALl? VALid

シンタックス： OUTPut:CH<n>:FALl? RANge

アーギュメント： なし

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:OUTPUT:CH<n>:FALL RANGE,]<min>,<max>

ここで

<min> ::= <NR3>　　最小値
<max> ::= <NR3>　　最大値

使用例： 次に、:OUTPUT:CH0:FALL?RANGE のレスポンス例を示します。

```
:OUTPUT:CH0:FALL RANGE,2.0E–9,7.0E–9
```

OUTPut:CH<n>:FALl? VALid

OUTPut:CH<n>:FALl? VALid 問い合せコマンドは、指定したチャンネルの立ち下がり時間とその設定値が有効であるかどうかを問い合わせます。

グループ： OUTPUT

関連コマンド： OUTPut:CH<n>:FALl?、OUTPut:CH<n>:FALl? RANge

シンタックス： OUTPut:CH<n>:FALl? VALid

アーギュメント： なし

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:OUTPUT:CH<n>:FALL VALID,]<NR3>,{1 | 0 }　　1：有効、0：無効

使用例： 次に、:OUTPUT:CH0:FALL? VALID のレスポンス例を示します。

:OUTPUT:CH0:FALL VALID,1.0E–9,1

OUTPut:CH<n>:HIGH(?)

OUTPut:CH<n>:HIGH コマンドは、ヘッダで指定するチャンネルの H レベル出力電圧を設定します。また、OUTPut:CH<n>:HIGH? 問合せコマンドは、指定チャンネルの H レベル出力電圧の設定値を問い合わせます。

グループ： OUTPUT

関連コマンド： OUTPut:CH<n>:LOW

シンタックス： OUTPut:CH<n>:HIGH <Volt>
OUTPut:CH<n>:HIGH?
（ <n>:0～7）

アーギュメント： <Volt> ::= <NR2>[<Unit>]　　H レベル出力電圧
<Unit> ::= {V | mV}

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:OUTPUT:CH<n>:HIGH] <NR2>

使用例： 次に、チャンネル 2 の H レベル出力電圧を 1 V に設定する例を示します。

:OUTPUT:CH2:HIGH 1 V

OUTPut:CH<n>:INHibit(?)

OUTPut:CH<n>:INHibit コマンドは、ヘッダで指定するチャンネルの出力インピーダンスをどのように制御するかを設定します。また、OUTPut: CH<n>:INHibit? 問合せコマンドは、指定チャンネルの出力インピーダンスがどのように制御されるかを問い合わせます。

グループ： OUTPUT

関連コマンド： なし

シンタックス：
OUTPut:CH<n>:INHibit {OFF | INTernal | EXTernal | BOTH | 0 | 1 | 2 | 3}
OUTPut:CH<n>:INHibit?
（ <n>:0～7）

アーギュメント：
{OFF | 0} — 出力インピーダンスを制御しません。
{INTernal | 1} — 出力インピーダンスをチャンネル 0 信号で制御します。
{EXTernal | 2} — 出力インピーダンスを外部入力（INH）信号で制御します。
{BOTH | 3} — チャンネル 0 信号と外部入力（INH）信号の論理和で制御します。

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:OUTPUT:CH<n>:INHIBIT] {0 | 1 | 2 | 3}

使用例： 次に、チャンネル 3 の 出力インピーダンスを外部入力（INH）信号で制御するための設定例を示します。

```
:OUTPUT:CH3:INHIBIT  EXTERNAL
```

OUTPut:CH<n>:LOW(?)

OUTPut:CH<n>:LOW コマンドは、ヘッダで指定するチャンネルの L レベル出力電圧を設定します。また、OUTPut:CH<n>:LOW? 問合せコマンドは、指定チャンネルの L レベル出力電圧の設定値を問い合わせます。

グループ： OUTPUT

関連コマンド： OUTPut:CH<n>:HIGH

シンタックス：
OUTPut:CH<n>:LOW <Volt>
OUTPut:CH<n>:LOW?
（ <n>:0～7）

アーギュメント：
<Volt> ::= <NR2>[<Unit>] L レベル出力電圧
<Unit> ::= {V | mV}

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:OUTPUT:CH<n>:LOW] <NR2>

使用例： 次に、チャンネル 2 の L レベル出力電圧を −0.5 V に設定する例を示します。

```
:OUTPUT:CH2:LOW  –0.5 V
```

OUTPut:CH<n>:RELEase

OUTPut:CH<n>:RELEase コマンドは、ヘッダで指定するチャンネルへのデータ・ビット割当てを解除します。

グループ： OUTPUT

関連コマンド： OUTPut:CH<n>:ASSIGn、OUTPut:DEFine

シンタックス： OUTPut:CH<n>:RELEase
（ <n>:0～7）

アーギュメント： なし

使用例： 次に、チャンネル 3 へのデータ・ビット割当てを解除する例を示します。

```
:OUTPUT:CH3:RELEASE
```

OUTPut:CH<n>:RISe(?)

OUTPut:CH<n>:RISe コマンドは、ヘッダで指定するチャンネルの立ち上がり時間を設定します。また、OUTPut:CH<n>:RISe? 問い合わせコマンドは、指定したチャンネルの立ち上がり時間を問い合わせます。なお、アーギュメントで FAST または 0.0 を指定した場合は、立ち上がり時間が限りなく速くなります。

グループ： OUTPUT

関連コマンド： OUTPut:CH<n>:RISe? RANge、OUTPut:CH<n>:RISe? VALid

シンタックス： OUTPut:CH<n>:RISe <Time>
OUTPut:CH<n>:RISe?
（ <n>:0～7）

アーギュメント： <Time> ::= <NR2>[<Unit>] | FAST
<Unit> ::= {s | ms | us | ns}

レスポンス： レスポンス・フォーマットは、次のとおりです。FAST の場合は、0.0 が返されます。

[:OUTPUT:CH<n>:RISE] <NR3>

使用例： 次に、チャンネル 1 の立ち上がり時間を1 ns に設定する例を示します。

```
:OUTPUT:CH1:RISE 1ns
```

OUTPut:CH<n>:RISe? RANge

OUTPut:CH<n>:RISe?RANge 問い合せコマンドは、指定したチャンネルの立ち上がり時間の有効設定範囲を問い合わせます。

グループ： OUTPUT

関連コマンド： OUTPut:CH<n>:RISe?、OUTPut:CH<n>:RISe? VALid

シンタックス： OUTPut:CH<n>:RISe? RANge
（ <n>:0～7）

アーギュメント： なし

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:OUTPUT:CH<n>:RISE RANGE,]<min>,<max>

ここで

<min> ::= <NR3>　　最小値
<max> ::= <NR3>　　最大値

使用例： 次に、:OUTPUT:CH0:RISE?RANGE のレスポンス例を示します。

:OUTPUT:CH0:RISE RANGE,1.0E–9,3.0E–9

OUTPut:CH<n>:RISe? VALid

OUTPut:CH<n>:RISe?VALid 問い合せコマンドは、指定したチャンネルの立ち上がり時間とその設定値が有効であるかどうかを問い合わせます。

グループ： OUTPUT

関連コマンド： OUTPut:CH<n>:RISe?、OUTPut:CH<n>:RISe? RANge

シンタックス： OUTPut:CH<n>:RISe? VALid
（ <n>:0～7）

アーギュメント： なし

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:OUTPUT:CH<n>:RISE VALID,]<NR3>,{0 | 1}　　1：有効、0：無効

使用例： 次に、:OUTPUT:CH0:RISE? VALID のレスポンス例を示します。

:OUTPUT:CH0:RISE VALID,1.0E–9,1

OUTPut:CHCLK:FALl(?)

OUTPut:CHCLK:FALl コマンドは、クロック出力の立ち下がり時間を設定します。また、OUTPut:CHCLK:FALl? 問い合わせコマンドは、クロック出力の立ち下がり時間を問い合わせます。なお、アーギュメントで FAST または 0.0 を指定した場合は、立ち下がり時間が限りなく速くなります。

グループ： OUTPUT

関連コマンド： OUTPut:CHCLK:FALl?RANge、OUTPut:CHCLK:FALl?VALid

シンタックス：
OUTPut:CHCLK:FALl <Time>
OUTPut:CHCLK:FALl?

アーギュメント：
<Time> ::= <NR2>[<Unit>] | FAST
<Unit> ::= {s | ms | us | ns}

レスポンス： レスポンス・フォーマットは、次のとおりです。FAST の場合は、0.0 が返されます。

[:OUTPUT:CHCLK:FALL] <NR3>

使用例： 次に、クロック出力の立ち下がり時間を1 ns に設定する例を示します。

```
:OUTPUT:CHCLK:FALL 1ns
```

OUTPut:CHCLK:FALl? RANge

OUTPut:CHCLK:FALl? RANge 問い合せコマンドは、クロック出力の立ち下がり時間の有効設定範囲を問い合わせます。

グループ： OUTPUT

関連コマンド： OUTPut:CHCLK:FALl?、OUTPut:CHCLK:FALl? VALid

シンタックス： OUTPut:CHCLK:FALl? RANge

アーギュメント： なし

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:OUTPUT:CHCLK:FALL RANGE,]<min>,<max>

ここで

<min> ::= <NR3>　　最小値
<max> ::= <NR3>　　最大値

使用例： 次に、:OUTPUT:CHCLK:FALL?RANGE のレスポンス例を示します。

```
:OUTPUT:CHCLK:FALL RANGE,2.0E–9,7.0E–9
```

OUTPut:CHCLK:FALl? VALid

OUTPut:CHCLK:FALl? VALid 問い合せコマンドは、クロック出力の立ち下がり時間とその設定値が有効であるかどうかを問い合わせます。

グループ： OUTPUT

関連コマンド： OUTPut:CHCLK:FALl?、OUTPut:CHCLK:FALl? RANge

シンタックス： OUTPut:CHCLK:FALl? VALid

アーギュメント： なし

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:OUTPUT:CHCLK:FALL VALID,]<NR3>,{1 | 0 }　　1：有効、0：無効

使用例： 次に、:OUTPUT:CHCLK:FALL? VALID のレスポンス例を示します。

```
:OUTPUT:CHCLK:FALL VALID,1.0E–9,1
```

OUTPut:CHCLK:HIGH(?)

OUTPut:CHCLK:HIGH コマンドは、クロック出力の H レベル出力電圧を設定します。また、OUTPut:CHCLK:HIGH? 問合せコマンドは、クロック出力の H レベル出力電圧の設定値を問い合わせます。

グループ： OUTPUT

関連コマンド： OUTPut:CHCLK:LOW

シンタックス：
OUTPut:CHCLK:HIGH <Volt>
OUTPut:CHCLK:HIGH?

アーギュメント：
<Volt> ::= <NR2>[<Unit>]　　H レベル出力電圧
<Unit> ::= {V | mV}

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:OUTPUT:CHCLK:HIGH] <NR2>

使用例： 次に、クロック出力の H レベル出力電圧を 2.0 V に設定する例を示します。

```
:OUTPUT:CHCLK:HIGH  2.0 V
```

OUTPut:CHCLK:LOW(?)

OUTPut:CHCLK:LOW コマンドは、クロック出力の L レベル出力電圧を設定します。また、OUTPut:CHCLK:LOW? 問合せコマンドは、クロック出力の L レベル出力電圧の設定値を問い合わせます。

グループ： OUTPUT

関連コマンド： OUTPut:CHCLK:HIGH

シンタックス：
OUTPut:CHCLK:LOW <Volt>
OUTPut:CHCLK:LOW?

アーギュメント：
<Volt> ::= <NR2>[<Unit>]　L レベル出力電圧
<Unit> ::= {V | mV}

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:OUTPUT:CHCLK:LOW] <NR2>

使用例： 次に、クロック出力の L レベル出力電圧を −1.5 V に設定する例を示します。

```
:OUTPUT:CHCLK:LOW  –1.5 V
```

OUTPut:CHCLK:RISe(?)

OUTPut:CHCLK:RISe コマンドは、クロック出力の立ち上がり時間を設定します。また、OUTPut:CHCLK:RISe? 問い合わせコマンドは、クロック出力の立ち上がり時間を問い合わせます。なお、アーギュメントで FAST または 0.0 を指定した場合は、立ち上がり時間が限りなく速くなります。

グループ： OUTPUT

関連コマンド： OUTPut:CHCLK:RISe? RANge、OUTPut:CHCLK:RISe? VALid

シンタックス：
OUTPut:CHCLK:RISe <Time>
OUTPut:CHCLK:RISe?

アーギュメント：
<Time> ::= <NR2>[<Unit>] | FAST
<UNit> ::= {s | ms | us | ns}

レスポンス： レスポンス・フォーマットは、次のとおりです 。FAST の場合は、0.0 が返されます。

[:OUTPUT:CHCLK:RISE] <NR3>

使用例： 次に、クロック出力の立ち上がり時間を1 ns に設定する例を示します。

```
:OUTPUT:CH1:RISE 1 ns
```

OUTPut:CHCLK:RISe? RANge

OUTPut:CHCLK:RISe?RANge 問い合せコマンドは、クロック出力の立ち上がり時間の有効設定範囲を問い合わせます。

グループ： OUTPUT

関連コマンド： OUTPut:CHCLK:RISe?、OUTPut:CHCLK:RISe? VALid

シンタックス： OUTPut:CHCLK:RISe? RANge

アーギュメント： なし

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:OUTPUT:CHCLK:RISE RANGE,]<min>,<max>

ここで

<min> ::= <NR3>　　最小値
<max> ::= <NR3>　　最大値

使用例： 次に、:OUTPUT:CHCLK:RISE?RANGE のレスポンス例を示します。

:OUTPUT:CHCLK:RISE RANGE,1.0E–9,3.0E–9

OUTPut:CHCLK:RISe? VALid

OUTPut:CHCLK:RISe?VALid 問い合せコマンドは、クロック出力の立ち上がり時間とその設定値が有効であるかどうかを問い合わせます。

グループ： OUTPUT

関連コマンド： OUTPut:CHCLK:RISe?、OUTPut:CHCLK:RISe? RANge

シンタックス： OUTPut:CHCLK:RISe? VALid

アーギュメント： なし

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:OUTPUT:CHCLK:RISE VALID,]<NR3>,{1 | 0 }　　1：有効、0：無効
<Time> ::= <NR2>

使用例： 次に、:OUTPUT:CHCLK:RISE? VALID のレスポンス例を示します。

:OUTPUT:CHCLK:RISE VALID,1.0E–9,1

OUTPut:DEFine(?)

OUTPut:DEFine コマンドは、ヘッダで指定するすべてのチャンネルにデータ・ビットを割り当てます。アーギュメントで割当ての記述がないデータ・ビットは割当てが解除されます。また、OUTPut:DEFine? 問合せコマンドは、指定のチャンネルに割り当てられているデータ・ビットを問い合わせます。

グループ： OUTPUT

関連コマンド： OUTPut:CH<n>:ASSIGn、OUTPut:CH<n>:RELEase

シンタックス： OUTPut:DEFine <Assigninfo>
OUTPut:DEFine?

アーギュメント： <Assigninfo> ::= <blockheader><Assign>[<LF><Assign][<LF><Assign>]...
チャンネルを定義するアービトラリ・ブロック・データ

ここで

<blockheader> ::= <byte count digit><byte count>

<Assign> ::= <AChannel>,<ABit>,<AHoldE>

<AChannel> 、<ABit> および <AHoldE> はアスキー文字で記述した次の情報です。

<AChannel>	チャンネル番号（0 ～ 7）
<ABit>	データ・ビット番号（0 ～ 7）
<AHoldE>	ハイ・インピーダンス制御の選択 （0 ：制御しない、1 ：内部信号、2 ：外部入力信号、 3 ：内部信号と外部入力信号の論理和） （OUTPut:CH<n>:INHibit コマンドを参照）

<LF> ::= ASCII 改行コード（10）

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:OUTPUT:DEFINE] <Assigninfo>

ここで、<Assigninfo> はアーギュメントと同形式のアービトラリ・ブロック・データを表します。

使用例： OUTPUT:DEFINE #2170,4,1<LF>1,5,2<LF>2,7,0

このコマンドを実行するとチャンネル割当ては次のようになります。

チャンネル0:	ビット 4、内部信号でハイ・インピーダンス制御
チャンネル1:	ビット 5、外部入力信号でハイ・インピーダンス制御
チャンネル2:	ビット 7、ハイ・インピーダンス制御 OFF
他のチャンネル:	割当解除

OUTPut:ELEVel(?)

OUTPut:ELEVel コマンドは、イベント入力（ event input ）のスレッショルド・レベルを設定します。また、OUTPut:ELEVel? 問合せコマンドは、イベント入力のスレッショルド・レベルの設定値を問い合わせます。

グループ： OUTPUT

関連コマンド： なし

シンタックス：
OUTPut:ELEVel <Volt>
OUTPut:ELEVel?

アーギュメント：
<Volt> ::= <NR2>[<Unit>]　スレッショルド・レベル（ –5 V ～ +5 V、0.1 Vステップ）
<Unit> ::= {V | mV}

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:OUTPUT:ELEVEL] <NR2>

使用例： 次に、イベント入力のスレッショルド・レベルを 500 mVに設定する例を示します。

```
:OUTPUT:ELEVEL  500 mV
```

OUTPut:ILEVel(?)

OUTPut:ILEVel コマンドは、高インピーダンス制御入力（ inhibit input ）のスレッショルド・レベルを設定します。また、OUTPut:ILEVel? 問合せコマンドは、高インピーダンス制御入力のスレッショルド・レベルの設定を問い合わせます。

グループ： OUTPUT

関連コマンド： なし

シンタックス：
OUTPut:ILEVel <Volt>
OUTPut:ILEVel?

アーギュメント：
<Volt> ::= <NR2>[<Unit>]　スレッショルド・レベル（ –5 V ～ +5 V、0.1 Vステップ）
<Unit> ::= {V | mV}

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:OUTPUT:ILEVEL] <NR2>

使用例： 次に、高インピーダンス制御入力のスレッショルド・レベルを 300 mVに設定する例を示します。

```
:OUTPUT:ILEVEL  300 mV
```

*PSC(?)

*PSC 共通コマンドは、ステータス／イベント・レポーティング・システムで使用する SRER（ Service Request Enable Register ）、DESER（ Device Event Status Enable Register ）および ESER （ Event Status Enable Register ）の電源投入時自動設定機構を制御します。また、*PSC? 共通・問合せコマンドは、電源投入時自動設定機構の設定状態（ PSC フラグ : power‐on status clear flag ）を問い合わせます。

グループ： その他

関連コマンド： DESE、*ESE、FACTory、*SRE

シンタックス：
*PSC　<Power‐On Status Clear>
*PSC?

アーギュメント： <Power‐On Status Clear> ::= <NR1>
（設定範囲:　‐32767～32767 ）

0　　— PSC フラグ（ power‐on status clear flag ）を FALSE に設定します。この状態では、電源投入時、DESER 、SESR 、ESER の値を、電源を OFF にする直前の状態に戻します。さらに、電源投入後の SRQ の発行が可能になります。

0以外の値 — PSC フラグ（ power‐on status clear flag ）を TRUE に設定します。この状態では、電源投入時、DESER をセット（ 255 に設定）し、SESR とESER をリセット（ 0 に設定）します。さらに、電源投入後の SRQ の発行を禁止します。

レスポンス： 問合せコマンドに対するレスポンスは、次のとおりです。
1　— PSC フラグが TRUE に設定されています。
0　— PSC フラグが FALSE に設定されています。

使用例： 次に、PSC フラグをTRUEに設定する例を示します。

```
*PSC 1
```

次に、*PSC? のレスポンス例を示します。

```
0
```

この場合、PSC フラグが FALSE に設定されてます。

*RST

*RST共通コマンドは、本機器をデフォルトの設定に戻します。

グループ： その他

関連コマンド： FACTory、SYSTem:SECurity:IMMediate

シンタックス： *RST

アーギュメント： なし

使用例： 次に、機器をデフォルトの設定に戻す例を示します。

```
*RST
```

RUNNing?

RUNNing? 問合せコマンドは、パターン・データまたはシーケンスが出力中かどうかを問い合わせます。

グループ： その他

関連コマンド： STARt、STOP、*TRG

シンタックス： RUNNing?

アーギュメント： なし

レスポンス： レスポンスは、次のとおりです。

1 — パターン・データまたはシーケンスが出力中です。
0 — 出力は行われていません。

使用例： 次に、:RUNNING? のレスポンス例を示します。

```
:RUNNING 1
```

この場合、パターン・データまたはシーケンスが出力中であることを示します。

SOURce:EVENT:ENABle(?)

SOURce:EVENT:ENABle コマンドは、イベント入力を有効または無効にします。また、SOURce:EVENT:ENABle? 問合せコマンドは、イベント入力が有効か無効かを問い合わせます。

グループ： SOURCE

関連コマンド： なし

シンタックス： SOURce:EVENT:ENABle {ON | OFF | 1 | 0}
SOURce:EVENT:ENABle?

アーギュメント： {ON | 1} — イベント入力を有効にします。
{OFF | 0} — イベント入力を無効にします。

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:SOURCE:EVENT:ENABLE] {1 | 0}

使用例： 次に、イベント入力を無効にする例を示します。

```
:SOURCE:EVENT:ENABLE OFF
```

SOURce[:OSCillator]?

SOURce[:OSCillator]? 問合せコマンドは、クロック信号に関するすべての設定状態を問い合わせます。

グループ： SOURCE

関連コマンド： SOURce:OSCillator:EXTernal:FREQuency、SOURce:OSCillator[:INTernal]:FREQuency、SOURce:OSCillator[:INTernal]:PLLlock、SOURce:OSCillator:SOURce

シンタックス： SOURce[:OSCillator]?

アーギュメント： なし

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:SOURCE:OSCILLATOR:SOURCE] {INTERNAL | EXTERNAL};[EXTERNAL:FREQUENCY] <NR3>HZ;[:SOURCE:OSCILLATOR:INTERNAL:FREQUENCY] <NR3>HZ;[PLLLOCK] {1 | 0}

使用例： 次に、:SOURCE:OSCILLATOR? のレスポンス例を示します。

```
:SOURCE:OSCILLATOR:SOURCE  INTERNAL ; EXTERNAL:FREQUENCY  1.000E＋8HZ ;
:SOURCE:OSCILLATOR:INTERNAL:FREQUENCY  2.000E＋8HZ ; PLLLOCK  1
```

SOURce:OSCillator:EXTernal:FREQuency(?)

SOURce:OSCillator:EXTernal:FREQuency コマンドは、外部クロック入力に供給するクロック信号の周波数を設定します。また、SOURce:OSCillator:EXTernal:FREQuency? 問合せコマンドは、外部クロック入力に供給するクロック信号の周波数を問い合わせます。

グループ： SOURCE

関連コマンド： SOURce[:OSCillator]?、SOURce:OSCillator[:INTernal]:FREQuency、
SOURce:OSCillator[:INTernal]:PLLlock、SOURce:OSCillator:SOURce

シンタックス： SOURce:OSCillator:EXTernal:FREQuency <Frequency>
SOURce:OSCillator:EXTernal:FREQuency?

アーギュメント： <Frequency> ::= <NR3>[<Unit>]
<Unit> ::= {Hz | kHz | MHz}
設定範囲： 0.1 ～ 400.0E+6 Hz

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:SOURCE:OSCILLATOR:EXTERNAL:FREQUENCY] <NRf>HZ

使用例： 次に、外部クロック入力に供給するクロック信号の周波数を 10 MHz に設定する例を示します。

```
:SOURCE:OSCILLATOR:EXTERNAL:FREQUENCY  10.0MHZ
```

SOURce:OSCillator[:INTernal]:FREQuency(?)

SOURce:OSCillator[:INTernal]:FREQuency コマンドは、内部クロック発振器の周波数を設定します。また、SOURce:OSCillator[:INTernal]:FREQuency? 問合せコマンドは、内部のクロック発振器の周波数設定を問い合わせます。

グループ： SOURCE

関連コマンド： SOURce[:OSCillator]?、SOURce:OSCillator:EXTernal:FREQuency、
SOURce:OSCillator[:INTernal]:PLLlock、SOURce:OSCillator:SOURce

シンタックス： SOURce:OSCillator[:INTernal]:FREQuency <Frequency>
SOURce:OSCillator[:INTernal]:FREQuency?

アーギュメント： <Frequency> ::= <NR3>[<Unit>]
<Unit> ::= {Hz | kHz | MHz}
設定範囲： 0.1 ～ 400.0E+6 Hz

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:SOURCE:OSCILLATOR:INTERNAL:FREQUENCY] <NRf>HZ

使用例： 次に、内部クロック発振器の周波数を 100 MHz に設定する例を示します。

```
:SOURCE:OSCILLATOR:INTERNAL:FREQUENCY  100MHZ
```

SOURce:OSCillator[:INTernal]:PLLlock(?)

SOURce:OSCillator[:INTernal]:PLLlock コマンドは、内部クロック発振器を基準発振器に位相同期（ PLL 作動）させるかどうかを設定します。また、SOURce:OSCillator[:INTernal]:PLLlock? 問合せコマンドは、内部のクロック発振器を基準発振器に位相同期（ PLL 作動）させる設定かどうかを問い合わせます。

グループ： SOURCE

関連コマンド： SOURce[:OSCillator]?、SOURce:OSCillator:EXTernal:FREQuency、
SOURce:OSCillator[:INTernal]:FREQuency、SOURce:OSCillator:SOURce

シンタックス：
SOURce:OSCillator[:INTernal]:PLLlock {ON | OFF | 1 | 0}
SOURce:OSCillator[:INTernal]:PLLlock?

アーギュメント：
{ON | 1} — 位相同期します（ PLL ON ）。
{OFF | 0} — 位相同期しません（ PLL OFF ）。

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:SOURCE:OSCILLATOR:INTERNAL:PLLLOCK] {1 | 0}

使用例： 次に、内部クロック発振器を基準発振器に位相同期させる例を示します。

```
:SOURCE:OSCILLATOR:INTERNAL:PLLLOCK  ON
```

SOURce:OSCillator:SOURce(?)

SOURce:OSCillator:SOURce コマンドは、内部クロック発振器と外部クロック入力信号のどちらをクロック信号源に使用するかを設定します。また、SOURce:OSCillator:SOURce? 問合せコマンドは、内部クロック発振器と外部クロック入力信号のどちらがクロック信号源として使用されているかを問い合わせます。

グループ： SOURCE

関連コマンド： SOURce[:OSCillator]?、SOURce:OSCillator:EXTernal:FREQuency、
SOURce:OSCillator[:INTernal]:FREQuency、SOURce:OSCillator[:INTernal]:PLLlock

シンタックス：
SOURce:OSCillator:SOURce {INTernal | EXTernal}
SOURce:OSCillator:SOURce?

アーギュメント：
INTernal — 内部クロックをクロック信号源にします。
EXTernal — 外部クロック入力をクロック信号源にします。

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:SOURCE:OSCILLATOR:SOURCE] {INTERNAL | EXTERNAL}

使用例： 次に、内部クロック発振器をクロック信号源として使用するための設定例を示します。

```
:SOURCE:OSCILLATOR:SOURCE  INTERNAL
```

*SRE(?)

*SRE 共通コマンドは、ステータス／イベント・レポーティング・システムで使用する SRER（ Service Request Enable Register ）に値を設定します。また、*SRE? 共通・問合せコマンドは、SRER の内容を問い合わせます。

グループ： その他

関連コマンド： *CLS、DESE、*ESE、*ESR?、EVENT?、EVMsg?、EVQty?、*STB?

シンタックス：
*SRE <Bit Value>
*SRE?

アーギュメント： <Bit Value> ::= <NR1> （設定範囲: 0 ～ 255 ）

アーギュメントは、0 から 255 までの間の 10 進数でなければなりません。SRER には、この値に対応する 2 進数が設定されます。

電源投入時の SRER の値は、PSC フラグが TRUE の場合、すべてのビットがリセットされます。PSC フラグが FALSE の場合、電源の ON/OFF とは無関係に、SRER の値が保持されます。PSC フラグについては、*PSC コマンドの項をご参照ください。

使用例： 次に、SRER を 48（ 00110000 ）に設定する例を示します。この場合、ESB と MAV ビットがセットされます。

```
*SRE 48
```

次は、*SRE? のレスポンス例です。

```
32
```

この場合、SRER は 00100000 に設定されています。

STARt

STARt コマンドは、本機器を START 状態にします。ラン・モードが Repeat または Step に設定されている場合、パターン・データまたはシーケンスの出力を開始します。また、ラン・モードが Single に設定されている場合、トリガ待ちの状態になります。

グループ： その他

関連コマンド： RUNNing?、STOP、*TRG

シンタックス： STARt

アーギュメント： なし

使用例： 次に、本機器を START 状態にする例を示します。

```
:START
```

*STB?

*STB?共通・問合せコマンドは、ステータス／イベント・レポーティング・システムで使用される SBR（ Status Byte Register ）の内容を問い合わせます。この場合、SBR のビット 6 は、MSS（ Master Status Summary ）ビットとして、読み取られます。

グループ： その他

関連コマンド： *CLS、DESE、*ESE、*ESR?、EVENT?、EVMsg?、EVQty?、*SRE

シンタックス： *STB?

アーギュメント： なし

レスポンス： レスポンスは、<NR1> のタイプで戻されます。

使用例： 次に、*STB? のレスポンス例を示します。

```
96
```

この場合、SBR の内容は 01100000 になります。

STOP

STOP コマンドは、パターン・データまたはシーケンスの出力を停止します。ラン・モードが Single に設定されている場合は、トリガ待ちの状態がキャンセルされます。

グループ： その他

関連コマンド： RUNNing?、STARt、*TRG

シンタックス： STOP

アーギュメント： なし

使用例： 次に、パターン・データまたはシーケンスの出力を停止する例を示します。

```
:STOP
```

SYSTem:DATE(?)

SYSTem:DATE コマンドは、内蔵時計の日付を設定します。また、SYSTem:DATE? 問合せコマンドは、内蔵時計の日付を問い合わせます。

グループ： SYSTEM

関連コマンド： SYSTem:TIME

シンタックス： SYSTem:DATE <Year>,<Month>,<Day>
SYSTem:DATE?

アーギュメント： <Year> ::= <NR1>　西暦年
<Month> ::= <NR1>　月
<Day> ::= <NR1>　日

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:SYSTEM:DATE] <Year>,<Month>,<Day>

使用例： 次に、日付を設定する例を示します。

```
:SYSTEM:DATE  97, 7, 7
```

SYSTem:PPAUse(?)

SYSTem:PPAUse コマンドは、パワーアップ時の自己診断でエラーが検出されたり、出力チャンネルが接続されていない場合に、オペレータのキー入力待ちの状態（パワーアップ・ポーズ）にするかどうかを設定します。また、SYSTem:PPAUse? 問合せコマンドは、パワーアップ・ポーズが ON / OFFのいずれに設定されているかを問い合わせます。

グループ： SYSTEM

関連コマンド： なし

シンタックス： SYSTem:PPAUse {ON | OFF | 1 | 0}
SYSTem:PPAUse?

アーギュメント： {ON | 1}　— パワーアップ・ポーズを ON にします。
{OFF | 0}　— パワーアップ・ポーズを OFF にします。

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:SYSTEM:PPAUSE?] {1 | 0}

使用例： 次に、パワーアップ・ポーズを ON に設定する例を示します。

```
:SYSTEM:PPAUSE  ON
```

SYSTem:SECurity:IMMediate

SYSTem:SECurity:IMMediate コマンドは、内部設定を工場出荷状態（ FACTory コマンドを実行した場合と同じ状態）にし、データ類を完全に消去します。消去されるデータには、ビット・パターン、グループ、ブロック、シーケンスなどが含まれます。GPIB と RS232C の設定、日付および時刻はリセットされません。

グループ： SYSTEM

関連コマンド： FACTory、*RST

シンタックス： SYSTem:SECurity:IMMediate

アーギュメント： なし

SYSTem:SECurity:STATe(?)

SYSTem:SECurity:STATe コマンドは、セキュリティを ON または OFF にします。また、SYSTem:SECurity:STATe? 問合せコマンドは、セキュリティが ON か OFF かを問い合わせます。セキュリティを ON から OFF にすると、内部メモリーの内容は完全に消去されます。FACTory コマンドを実行しても、セキュリティの ON/OFF の設定は変化しません。

グループ： SYSTEM

関連コマンド： SYSTem:SECurity:IMMediate

シンタックス： SYSTem:SECurity:STATe {ON | OFF | 1 | 0}
SYSTem:SECurity:STATe?

アーギュメント： {ON | 1} — セキュリティを ON にします。
{OFF | 0} — セキュリティを OFF にします。

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:SYSTEM:SECURITY:STATE] {1 | 0}

使用例： 次に、セキュリティを ON に設定する例を示します。

```
:SYSTEM:SECURITY:STATE  ON
```

SYSTem:TIME(?)

SYSTem:TIME コマンドは、内蔵時計の時刻を設定します。また、SYSTem:TIME? 問合せコマンドは、内蔵時計の時刻を問い合わせます。

グループ： SYSTEM

関連コマンド： SYSTem:DATE

シンタックス： SYSTem:TIME <Hour>,<Minute>,<Second>
SYSTem:TIME?

アーギュメント： <Hour> — 時
<Minute> — 分
<Second> — 秒

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:SYSTEM:TIME] <Hour>,<Minute>,<Second>

使用例： 次に、時刻を設定する例を示します。

```
:SYSTEM:TIME  11, 23, 58
```

*TRG

*TRG 共通コマンドは、トリガ・イベントを生成します。

グループ： その他

関連コマンド： RUNNing?、STARt、STOP

シンタックス： *TRG

アーギュメント： なし

使用例： 次に、トリガ・イベントを生成する例を示します。

```
*TRG
```

TRIGger?

TRIGger? 問合せコマンドは、トリガ入力に関するすべての設定を問い合わせます。

グループ： TRIGGER

関連コマンド： TRIGger:IMPedance、TRIGger:LEVel、TRIGger:SLOPe、TRIGger:SOURce

シンタックス： TRIGger?

アーギュメント： なし

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:TRIGGER:IMPEDANCE] {HIGH | LOW};[LEVEL] <NR2>;
[SLOPE] {POSITIVE | NEGATIVE};
[SOURCE] {EXTERNAL | INTERNAL};
[INTERVAL:TIME] <Time>;
[STATE] {1 | 0}

使用例： 次に、:TRIGGER? に対するレスポンス例を示します。

:TRIGGER:IMPEDANCE LOW ; LEVEL 1.400 ; SLOPE POSITIVE;SOURCE INTERNAL;
INTERVAL:TIME 1.0us;STATE 1

TRIGger:IMPedance(?)

TRIGger:IMPedance コマンドは、トリガ入力端子のインピーダンスを設定します。また、TRIGger:IMPedance? 問合せコマンドは、トリガ入力端子のインピーダンス設定を問い合わせます。

グループ： TRIGGER

関連コマンド： TRIGger:LEVel、TRIGger:SLOPe、TRIGger:SOURce

シンタックス： TRIGger:IMPedance {HIGH | LOW}
TRIGger:IMPedance?

アーギュメント： HIGH — 負荷インピーダンスを 1 kΩ に設定します。
LOW — 負荷インピーダンスを 50 Ω に設定します。

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:TRIGGER:IMPEDANCE] {HIGH | LOW}

使用例： 次に、トリガ入力のインピーダンスを 50 Ω に設定する例を示します。

:TRIGGER:IMPEDANCE LOW

TRIGger:INTERVal?

TRIGger:INTERVal? 問い合わせコマンドは、周期的に発生する内部トリガの情報 (インターバル時間と有効／無効)を問い合わせます。

グループ： TRIGGER

関連コマンド： TRIGger:INTERVal:STATe、TRIGger:INTERVal:TIME

シンタックス： TRIGger:INTERVal?

アーギュメント： なし

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:TRIGGER:INTERVAL:TIME] <Time>;[STATE] {1 | 0 }

使用例： 次に、:TRIGGER:INTERVAL?に対するレスポンス例を示します。

:TRIGGER:INTERVAL:TIME 1.0MS;STATE 1

TRIGger:INTERVal:STATe(?)

TRIGger:INTERVal:STATe コマンドは、周期的に発生する内部トリガを ON または OFF に設定します。また、TRIGger:INTERVal:STATe ? 問合せコマンドは、内部トリガの設定が ON か OFF かを問い合わせます。

グループ： TRIGGER

関連コマンド： TRIGger:INTERVal?、TRIGger:INTERVal:TIME

シンタックス： TRIGger:INTERVal:STATe {ON | OFF | 1 | 0 }
TRIGger:INTERVal:STATe?

アーギュメント： {ON | 1} — 周期的な内部トリガを有効にします。
{OFF | 0} — 周期的な内部トリガを無効にします。

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:TRIGGER:INTERVAL:STATE] {0 | 1}

使用例： 次に、内部トリガを ON に設定する例を示します。

:TRIGGER:INTERVAL:STATE ON

TRIGger:INTERVal:TIME(?)

TRIGger:INTERVal:TIME コマンドは、周期的に発生する内部トリガのインターバル時間を設定します。また、TRIGger:INTERVal:TIME? 問合せコマンドは、内部トリガのインターバル時間を問い合わせます。なお、この設定は、トリガ・ソースの選択が外部の場合でも行えます。

グループ： TRIGGER

関連コマンド： TRIGger:INTERVal?、TRIGger:INTERVal:STATe

シンタックス：
TRIGger:INTERVal:TIME <Time>
TRIGger:INTERVal:TIME?

アーギュメント：
<Time> ::= <NR2>[<Unit>]　　インターバル時間 (有効範囲：1.0 µs～10.0 s 、有効 3 桁)
<Unit> ::= {s | ms | µs | ns}

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:TRIGGER:INTERVAL:TIME] <NR3>

使用例： 次に、内部トリガのインターバル時間を1 µs に設定する例を示します。

```
:TRIGGER:INTERVAL:TIME 1us
```

TRIGger:LEVel(?)

TRIGger:LEVel コマンドは、トリガ入力のスレッショルド電圧を設定します。また、TRIGger:LEVel? 問合せコマンドは、設定されているトリガ入力のスレッショルド電圧を問い合わせます。

グループ： MODE

関連コマンド： TRIGger:IMPedance、TRIGger:SLOPe、TRIGger:SOURce

シンタックス：
TRIGger:LEVel <Threshold>
TRIGger:LEVel?

アーギュメント：
<Threshold> ::= <NR2>[<Unit>]　　スレッショルド電圧値
<Unit> ::= {V | mV}
設定範囲： －5.0～＋5.0

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:TRIGGER:LEVEL] <NR2>

ここで、<NR2> は現在のスレッショルド電圧を表わす実数値（単位 V ）です。

使用例： 次に、スレッショルド電圧を 200 mV に設定する例を示します。

```
:TRIGGER:LEVEL  200 mV
```

TRIGger:SLOPe(?)

TRIGger:SLOPe コマンドは、トリガ入力の検出エッジの極性を設定します。また、TRIGger:SLOPe? 問合せコマンドは、トリガ入力の検出エッジが立ち上がり／立ち下がりエッジのどちらに設定されているかを問い合わせます。

グループ： TRIGGER

関連コマンド： TRIGger:IMPedance、TRIGger:LEVel、TRIGger:SOURce

シンタックス： TRIGger:SLOPe {POSitive | NEGative}
TRIGger:SLOPe?

アーギュメント： POSitive — トリガ入力の検出エッジを立ち上がりエッジに設定します。
NEGative — トリガ入力の検出エッジを立ち下がりエッジに設定します。

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:TRIGGER:SLOPE] {POSITIVE | NEGATIVE}

使用例： 次に、トリガ入力の検出エッジを立ち上がりエッジに設定する例を示します。

```
:TRIGGER:SLOPE  POSITIVE
```

TRIGger:SOURce(?)

TRIGger:SOURce コマンドは、トリガ信号源として外部信号または内部信号のどちらを使用するかを設定します。また、TRIGger:SOURce? 問合せコマンドは、トリガ信号源が外部信号／内部信号のどちらに設定されているかを問い合わせます。

グループ： TRIGGER

関連コマンド： TRIGger:IMPedance、TRIGger:LEVel、TRIGger:SLOPe

シンタックス： TRIGger:SOURce{EXTernal | INTernal}
TRIGger:SOURce?

アーギュメント： EXTernal — トリガ信号源として、外部信号を使用します。
INTernal — トリガ信号源として、内部信号を使用します。

レスポンス： レスポンス・フォーマットは、次のとおりです。

[:TRIGGER:SOURCE] {EXTERNAL | INTERNAL}

使用例： 次に、トリガ信号源として外部信号を使用する例を示します。

```
:TRIGGER:SOURCE EXTERNAL
```

*TST?

*TST? 共通・問合せコマンドは、セルフ・テストを実行し、結果を戻します。セルフ・テスト実行中にエラーが検出されると、実行は直ちに中止されます。

グループ： その他

関連コマンド： DIAGnostic:RESUlt?、DIAGnostic:SELect、DIAGnostic:STATe

シンタックス： *TST?

アーギュメント： なし

レスポンス： レスポンス・フォーマットは、次のとおりです。

<Result>

ここで

<Result> ::= <NR1>　次のいずれかです。

0	—	正常終了
100	—	CPUユニットでエラーを検出
200	—	ディスプレイ・ユニットでエラーを検出
300	—	フロント・パネルでエラーを検出
400	—	クロック発生ユニットでエラーを検出
500	—	トリガ処理ユニットでエラーを検出
600	—	シーケンス・メモリでエラーを検出
700	—	パターン・メモリでエラーを検出

注： セルフ・テストには、最大 90 秒程度必要です。この間に次のコマンドを送っても受け付けられません。

使用例： 次に、*TST? のレスポンス例を示します。

```
200
```

この場合、ディスプレイ・ユニットでエラーが検出されたことを示します。

UNLock

UNLock コマンドは、フロント・パネルのキーおよびノブがロックされているのを解除します。フロント・パネルのキーおよびノブのロックは、LOCk ALL コマンドで行うことができます。なお、このコマンドは、LOCk NONe コマンドと同等です。

グループ： その他

関連コマンド： LOCk

シンタックス： UNLock ALL

アーギュメント： ALL — フロント・パネルのキーおよびノブのロックを解除します。

使用例： 次に、キーおよびノブのロックを解除する例を示します。

```
:UNLOCK ALL
```

UPTime?

UPTime? 問合せコマンドは、電源投入後の経過時間を問い合わせます。

グループ： その他

関連コマンド： なし

シンタックス： UPTime?

アーギュメント： なし

使用例： 次に、:UPTIME? に対するレスポンス例を示します。

```
:UPTIME 7.016
```

この場合、電源投入後 7.016 時間が経過していることを示します。

VERBose(?)

VERBose コマンドは、レスポンスにコマンド・ヘッダが含まれる場合、省略形（ショート・コマンド・ヘッダ）を使用するかどうかを指定します。省略形を使用した場合には、バス転送速度が向上し、省略しない場合（ロング・コマンド・ヘッダ）には、“読み易さ”が向上するなど、それぞれメリットがあります。

グループ： その他

関連コマンド： HEADer

シンタックス： VERBose {ON | OFF | <NR1>}
VERBose?

アーギュメント： ON または 0 以外の値 — ロング・コマンド・ヘッダを使用します。
OFF または 0 — ショート・コマンド・ヘッダを使用します。
.

レスポンス： レスポンス・フォーマットは、次のとおりです。
1 — ロング・コマンド・ヘッダが使用されています。
0 — ショート・コマンド・ヘッダが使用されています。

使用例： 次に、ロング・コマンド・ヘッダに設定する例を示します。

```
:VERBOSE ON
```

次の例は、:VERBOSE? のレスポンス例です。

```
:VERBOSE 1
```

この場合、ロング・コマンド・ヘッダが使用されています。

*WAI

*WAI共通コマンドは、ペンディング中のすべての操作が完了するまで、以後のコマンドまたは問合せコマンドの実行を待ちます。

グループ： その他

関連コマンド： *OPC

シンタックス： *WAI

アーギュメント： なし

レスポンス・メッセージの取り出しについて

GPIB インタフェースを使用した場合と RS-232C インタフェースを使用した場合では、レスポンス・メッセージの取り出し方が異なります。図 2-3 および図 2-4 に、それぞれの場合の概要を示します。

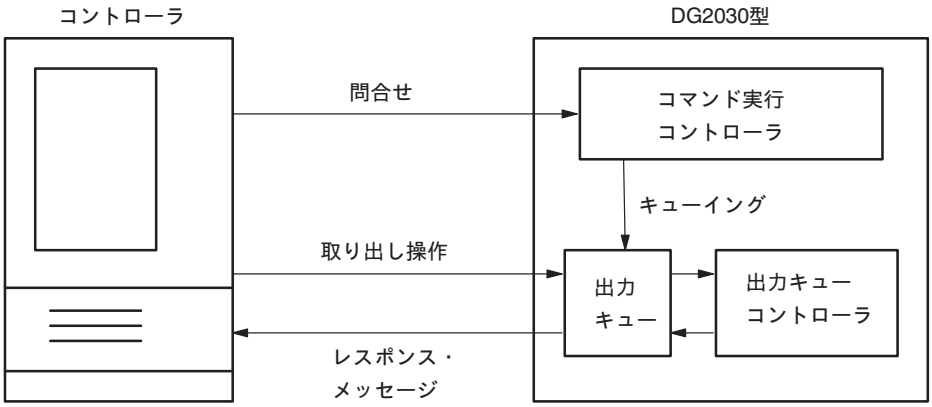


図 2-3: GPIB : レスポンス・メッセージの取り出し

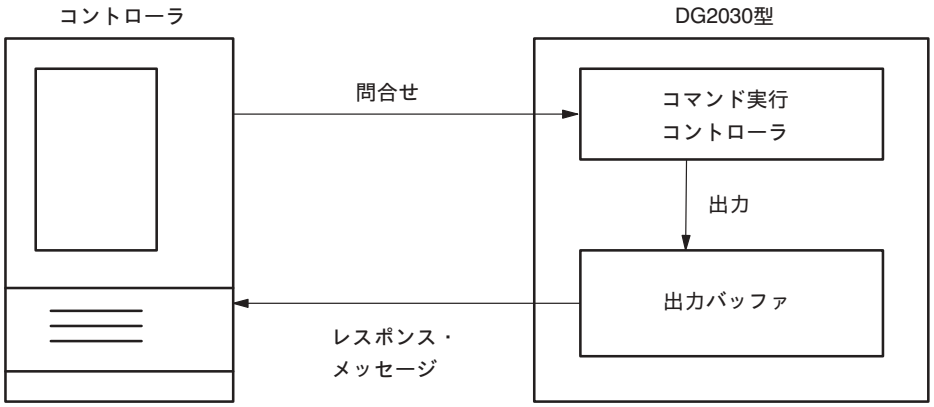


図 2-4: RS-232C : レスポンス・メッセージの取り出し

図 2-3 は、GPIB インタフェースを使用した場合のレスポンス・メッセージの取り出しを示しています。外部コントローラから問合せコマンドが転送されると、本機器は、問合せに対するレスポンス・メッセージを出力キューに入れます。このレスポンス・メッセージは、ユーザが外部コントローラを通して取り出し操作を行わない限り、取り出すことはできません。

レスポンス・メッセージが出力キューにキューイングされている状態で、取り出し操作が行われる前に再び外部コントローラから問合せコマンドが転送された場合、本機器はキューイング中のレスポンス・メッセージを消去し、新たに転送された問合せコマンドに対するレスポンス・メッセージを出力キューに入れます。

レスポンス・メッセージのキューイング状態は、SBR (Status Byte Register) の MAV ビットを使用して調べることができます。出力キュー、SBR、またはそのコントロール方法については、「第 3 章　ステータス／イベント」をご参照ください。

図 2-4 は、RS-232C インタフェースを使用した場合のレスポンス・メッセージの取り出しを示しています。外部コントローラから問合せコマンドが転送されると、本機器は、出力バッファを経由し、ただちにレスポンス・メッセージを外部コントローラに転送します。したがって、外部コントローラとして ダム・ターミナル（Dumb Terminal）を使用している場合や、PC 上で端末ユーティリティ・ソフトウェアを使用しているような場合には、問合せコマンドをタイプした後、CRT にレスポンス・メッセージがただちに表示されます。

RS-232C インタフェースの場合には、GPIB インタフェースの場合と異なり、次々に問合せコマンドを送っても、レスポンス・メッセージが消去されることはありません。

第３章　ステータス／イベント

ステータス／イベント・レポーティング・システム

本機器には、GPIB と RS-232C のインタフェースに、ステータス／イベント・レポーティング機能が組み込まれており、本機器内で発生する重要なイベント情報を知ることができます。

ステータス／イベント・レポーティング・システムには、IEEE Std. 488.2-1987 に準拠する 4 つのレジスタと 1 つのキュー、および Tektronix 仕様のレジスタとキューが各 1 個づつ、合計 5 つのレジスタと 2 つのキューが使用されています。次に、これらのレジスタとキューについて述べ、さらにステータス／イベント処理の流れを簡単に説明します。

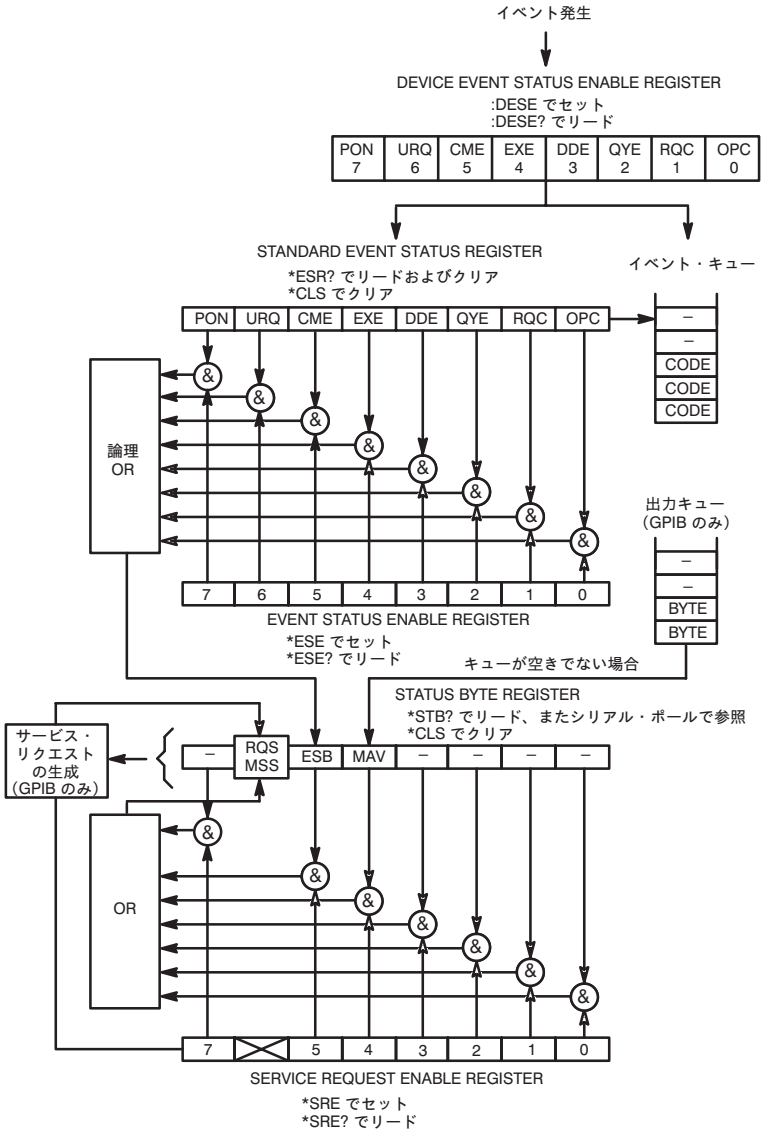


図 3-1: ステータス／イベント処理シーケンス概要

レジスタ

レジスタは、その機能によって、次のように 2 つに大別されます。

- **ステータス・レジスタ** — 本機器のステータスについての情報をストアします。
- **イネーブル・レジスタ** — 本機器で発生したイベントを、イベントのタイプに応じて、ステータス・レジスタとイベント・キューに設定するかどうかを決定します。

ステータス・レジスタ

ステータス・レジスタには、SBR（Status Byte Register）と SESR（Standard Event Status Register）があります。ステータス・レジスタの各ビットには、実行エラーやサービス要求などのように、ある特定のタイプのイベントが記録されます。イベントが発生すると、対応するビットがセットされるため、レジスタの内容を調べることによって、どのタイプのイベントが発生したかを知ることができます。

SBR（Status Byte Register）

ステータス・バイト・レジスタ SBR は、8 ビットで構成されるレジスタで、この内ビット 4、5、6 は、IEEE Std. 488.2-1987 で下記のように定義されており、それぞれ出力キュー（Output Queue）、SESR（Standard Event Status Register）、サービス要求をモニタするのに使用されます。ビット 0 ～ 3 および 7 は、ユーザが定義可能なビットですが、本機器では使用していないので、常に 0 に設定されています。なお、RS-232C インタフェースの場合には、ビット 5 のみが有効です。

- **ビット 4** — **MAV**（**Message Available**）は、出力キューにメッセージがスタックされ、取り出し可能な状態にあることを示します。RS‐232Cインタフェースの場合は、使用されません。
- **ビット 5** — **ESB**（**Event Status Bit**）は、前回のSESR（Standard Event Status Register）のクリアまたはイベント読み出し操作後に、新たなイベントが発生したかどうかを示します。
- **ビット 6** — **RQS**（**Request Service**）／**MSS**（**Master Status Summary**）. GPIB のシリアル・ポール・コマンドによってアクセスされる場合、このビットは、RQS（Request Service）ビットと呼ばれます。RQS は、サービス要求が発生したこと、つまり GPIB バスの SRQ ラインが“L”になっていることをコントローラに示すものです。RQS ビットは、シリアル・ポールが終了するとクリアされます。

  また、*STB? 問い合わせコマンドによってアクセスされる場合、このビットは、MSS（Master Status Summary）ビットと呼ばれます。MSS は、機器が少なくとも 1 つ以上の理由によりサービス・リクエストを行っていることを示します。MSS ビットは、*STB? 問い合わせコマンドによってゼロ・クリアされることはありません。

  なお、RS-232C インタフェースの場合、サービス・リクエストおよびシリアル・ポールの機能がないので、このビットは使用されません。

SESR（Standard Event Status Register）

スタンダード・イベント・ステータス・レジスタ SESR は、8 ビットからなるレジスタで、各ビットは次に説明する 8 つのタイプのイベントの発生を記録します。

- **ビット 7** — **Power On**（**PON**）. 本機器の電源が ON になったことを示します。

- **ビット 6** — **User Request**（**URQ**）. 機器がオペレータに対して何らかの要求を行うイベント、あるいは注意を促すイベントが発生したことを示します。

- **ビット 5** — **Command ERROR**（**CME**）. コマンド・パーサで、パーズ中にコマンド・エラーが検出されたことを示します。

- **ビット 4** — **Execution ERROR**（**EXE**）. コマンド実行中に、実行エラーが検出されたことを示します。実行エラーは、次の 2 つの理由によって発生します。

  a. アーギュメントに指定された値が、機器で許される範囲外の場合や、機器の能力に矛盾するような場合。

  b. 本来要求されるべき実行条件と異なる条件で実行したため、正しく動作しなかった場合。

- **ビット 3** — **Device-Specific ERROR**（**DDE**）. 機器に依存したエラーが検出されたことを示します。

- **ビット 2** — **Query ERROR**（**QYE**）. 出力キュー・コントローラによって、問い合わせエラーが検出されたことを示します。エラーは、次の 2 つの理由によって発生します。

  a. 出力キューが空またはペンディング状態であるにもかかわらず、出力キューからメッセージを取り出そうとした場合。

  b. 出力キューのメッセージが、取り出し操作を行っていないのに、クリアされた場合。

  RS‐232C インタフェースの場合は、出力キューを使用しないので、このビットは無効です。

- **ビット 1** — **Request Control**（**RQC**）. 機器がコントローラに対して、IEEE Std. 488.1 で定めるバスの管理権を譲るように要求していることを示します。ただし、本機器ではいずれのインタフェースでもこのビットを使用しません。

- **ビット 0** — **Operation Complete**（**OPC**）. このビットは、*OPC コマンドの実行結果として設定されるもので、ペンディングされたすべてのコマンドの実行が完了したことを示します。

イネーブル・レジスタ

イネーブル・レジスタには、DESER（Device Event Status Enable Register）、ESER（Event Status Enable Register）、SRER（Service Request Enable Register）があります。

イネーブル・レジスタの各ビットは、制御するステータス・レジスタのビットに対応しています。オペレータは、このイネーブルレジスタの各ビットをセットまたはリセットすることによって、イベント発生時に、そのイベントをステータス・レジスタやキューに記録するかどうかを決めることができます。つまり、ステータス・レジスタをマスクする働きをします。

DESER（Device Event Status Enable Register）

デバイス・イベント・ステータス・イネーブル・レジスタ DESERは、SESR のビット 0 から 7 の内容と同じ定義のビットで構成されます。このレジスタは、イベント発生時に、どのイベントを SESR およびイベント・キューへ記録し、どのイベントを無視するのかを、オペレータが指定するためのものです。たとえば、DESER のすべてのビットを 0 にリセットした場合、SESR およびイベント・キューにエラーが記録されなくなります。

イベントを SESR にセットし、イベント・キューにスタックするためには、DESER のイベントに対応するビットをセットします。また、イベントを無視する場合には、SESR のイベントに対応するビットをリセットしておきます。

DESER の値の設定は DESE コマンドで、DESER の内容の参照は DESE? 問い合わせコマンドで行います。

ESER（Event Status Enable Register）

イベント・ステータス・イネーブル・レジスタ ESER は、SESR のビット 0 から 7 の内容と同じ定義のビットで構成されます。このレジスタは、イベントが発生して、対応する SESR のビットがセットされた時、SBR の ESB ビットをセットするかどうかを、オペレータが指定するためのものです。

SESR のビットがセットされた時、SBR の ESB ビットをセットするには、イベントに対応する ESER のビットをセットし、ESB ビットをセットしないようにするには、イベントに対応する ESER のビットをリセットします。たとえば、ESER のすべてのビットを 0 にリセットした場合、どのようなエラーが発生しても、SBR の ESB にはビットがセットされることはありません。

ESER の内容は、*ESE コマンドで設定できます。また、*ESE? 問い合わせコマンドで ESER の内容を問い合わせることができます。

SRER（Service Request Enable Register）

サービス・リクエスト・イネーブル・レジスタ SRER は、ステータス・バイト・レジスタ SBR のビット 6 を制御します。このレジスタをセットしておくと、対応する SBR のビットがセットされた時、SBR の RQS ビットがセットされ、サービス・リクエスト（SRQ）が生成されます。

サービス・リクエストの生成とは、機器が GPIB バスの SRQ ラインを“Low”にしてコントローラにサービス・リクエストを行い、コントローラが行うシリアル・ポーリングに対して、RQS を設定したステータス・バイトを返送することです。

SRER の内容は、*SRE コマンドで設定できます。また、SRER の内容は、*SRE? 問い合わせコマンドで問い合わせることができます。ビット 6 は、常に 0 に設定されなければなりません。

なお、RS‐232C インタフェースの場合には、サービス・リクエスト（SRQ）の機能はありません。

キュー

ステータス／イベント・レポーティング・システムでは、出力キューとイベント・キューの 2 つのキューが使用されています。

出力キュー（Output Queue）

出力キューは、FIFO キューで、問い合わせコマンドなどに対するレスポンス・メッセージがスタックされ、取り出されるのを待ちます。メッセージがスタックされる際には、SBR（Status Byte Register）の MAV ビットもセットされます。

GPIB インタフェースの場合、出力キューは、コマンドまたは問い合わせコマンドを受け取るごとに空になるので、次のコマンドまたは問い合わせコマンドを発行する前に取り出す必要があります。取り出さずにコマンドまたは問い合わせコマンドを発行すると、エラーとなり、出力キューが空になります（ただし、エラーになっても実行には影響しません）。

なお、RS-232C インタフェースは、この出力キューを使用しません。このため、MAV ビットも影響を受けません。RS‐232C インタフェースの場合には、出力バッファに送られたレスポンス・メッセージは、ただちに出力ラインを通して、外部コントローラに送られます。GPIB インタフェースの場合のように、外部コントローラがメッセージを取り出す操作をする必要はありません。

イベント・キュー（Event Queue）

イベント・キューは、FIFO キューで、機器で発生したイベントが最大 20 個までスタックされます。イベント数が 20 を超えると、20 番目のイベントは、イベント・コード 350 の“Queue Overflow”に置き換えられます。

イベントの取り出しは、*ESR? 問い合わせコマンドで同期を取りながら、 ALLEv?、EVENT?、または EVMsg? 問い合わせコマンドを使用して次のように行います。

まず、 *ESR? を発行して、SESR の内容を読み出します。SESR の内容が読み出されると、SESR がクリアされ、同時にイベント・キューからイベントが取り出し可能な状態になります。 ALLEv? は、取り出しが可能になったすべてのイベントを取り出し、イベント・コードとメッセージ・テキストで返送します。 EVENT? と EVMsg? は、取り出し可能なイベントの内、最も古いイベントを取り出し、 EVENT? はイベント・コードのみを、EVMsg? はイベント・コードとメッセージ・テキストを返送します。

イベントが取り出される前に、新しいイベントが発生すると、イベントに対応する SESR のビットがセットされると共に、イベントがイベント・キューにスタックされます。ただし、イベント・キューから取り出せるイベントは、*ESR? によって取り出しが可能になったイベントのみです。イベントを取り出す前に、さらに *ESR? を発行した場合には、取り出し可能なイベントは削除され、代わって、前回の *ESR? 発行後に発生したイベントが取り出し可能になります。

例　*ESR?
　　ALLEV?

ステータス／イベント処理シーケンス

図 3-2 に、ステータス／イベントの処理の流れを示します。

イベントが発生すると、まず、DESER の内容を調べます。イベントに対応するDESER のビットがセットされている場合は、イベントに対応する SESR のビットがセットされ、イベント・キューにイベントがスタックされます。また、ESER でもイベントに対応するビットがセットされていれば、SBR の ESB ビットもセットされます。

メッセージが出力キューに送られた場合には、SBR の MAV ビットがセットされます（GPIB インタフェースのみ）。

SBR のいずれかのビットがセットされ、かつ対応する SRER のビットがセットされていると、SBR の MSS ビットがセットされ、サービス要求が生成されます。ただし、サービス要求が生成されるのは、GPIB インタフェースの場合のみです。

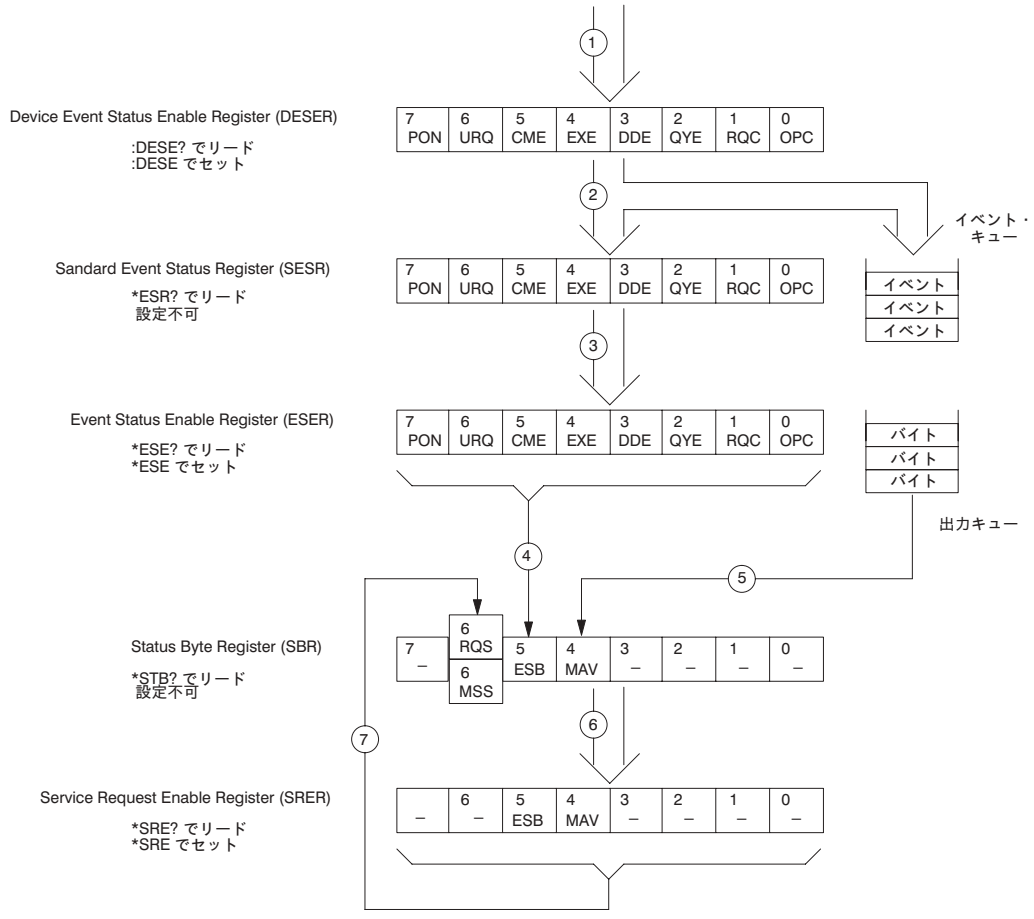


図 3-2: ステータス／イベントの処理過程

メッセージ

表 3-2 から表 3-8 に、GPIB および RS-232C インタフェースのステータス／イベント・レポーティング・システムで使用されるコードとメッセージを示します。

ほとんどのメッセージでは、メッセージの内容をより詳しく説明したり、エラーの原因をより詳細に知らせるために、イベント・メッセージとセミコロンに続いて、二次メッセージ（このマニュアルでは説明していません）が添えられます。イベント・コードとメッセージは、EVMsg?、ALLEv? で問い合わせができ、次のフォーマットで返送されます。

＜イベント・コード＞、”＜イベント・メッセージ;二次メッセージ＞”

また、EVENT? 問い合わせコマンドは、イベント・コードのみを返送します。これらの問い合わせコマンドを使用する場合には、*ESR? 問い合わせコマンドの使用と同期を取らなければならないことに注意してください。

例　*ESR?
　　ALLEV?

イベント・コードは、表 3-1 のように分類されています。エラーが発生した場合、コードのレンジを参照するだけで、どのクラスのエラーが起きているかを知ることができます。なお、本機器で使用するイベントの詳細については、クラスごとに分類して、表 3-2 から表3-9 に示します。

表 3-1: イベント・コードの分類

イベント・クラス	イベント・コード・レンジ	意　味
No Events	0 – 1	イベントやステータスのない状態
Reserved	2 – 99	（未使用）
Command Errors	100 – 199	コマンド・エラー
Execution Errors	200 – 299	コマンド実行エラー
Device-Specific Errors	300 – 399	内部機器エラー
Query Errors	400 – 499	システム・イベントと問い合わせエラー
Execution Warnings	500 – 599	実行ワーニング（実行の中断なし）
Reserved	600 – 1999	（未使用）
Extended Execution Errors	2000 – 2999	機器依存コマンド実行エラー
Extended Device-Specific Errors	3000 – 3999	機器依存機器エラー
Reserved	4000 –	（未使用）

表 3-2 は、イベントやステータスがない状態のメッセージです。対応する SESR のビットはありません。

表 3-2: 正常状態

コード	メッセージ
0	No events to report – queue empty
1	No events to report – new events pending *ESR?

表 3-3 は、コマンドのシンタックスに誤りがある場合に生成されるメッセージです。この場合、コマンドが正しく記述されているかどうかをチェックしてください。

表 3-3: コマンド・エラー（CMEビット：5）

コード	メッセージ
100	Command error
101	Invalid character
102	Syntax error
103	Invalid separator
104	Data type error
105	GET not allowed
106	Invalid program data separator
108	Parameter not allowed
109	Missing parameter
110	Command header error
111	Header separator error
112	Program mnemonic too long
113	Undefined header
114	Header suffix out of range
118	Query not allowed
120	Numeric data error
121	Invalid character in number
123	Exponent too large
124	Too many digits
128	Numeric data not allowed
130	Suffix error
131	Invalid suffix
134	Suffix too long
138	Suffix not allowed
140	Character data error
141	Invalid character data
144	Character data too long
148	Character data not allowed
150	String data error
151	Invalid string data
152	String data too long
158	String data not allowed
160	Block data error
161	Invalid block data
168	Block data not allowed

表 3-3: コマンド・エラー（CMEビット：5）（続き）

コード	メッセージ
170	Expression error
171	Invalid expression
178	Expression data not allowed
180	Macro error
181	Invalid outside macro definition
183	Invalid inside macro definition
184	Macro parameter error

表 3-4 は、コマンド実行中に検出されたエラーのエラー・メッセージです。

表 3-4: 実行エラー（EXEビット：4）

コード	メッセージ
200	Execution error
201	Invalid while in local
202	Settings lost due to rtl
203	Command protected
210	Trigger error
211	Trigger ignored
212	Arm ignored
213	Init ignored
214	Trigger deadlock
215	Arm deadlock
220	Parameter error
221	Settings conflict
222	Data out of range
223	Too much data
224	Illegal parameter value
225	Parameter underrange
226	Parameter overrange
227	Parameter rounded
230	Data corrupt or stale
231	Data questionable
240	Hardware error
241	Hardware missing

表 3-4: 実行エラー（EXEビット：4）（続き）

コード	メッセージ
250	Mass storage error
251	Missing mass storage
252	Missing media
253	Corrupt media
254	Media full
255	Directory full
256	File name not found
257	File name error
258	Media protected
260	Expression error
261	Math error in expression
262	Expression syntax error
263	Expression execution error
270	Macro error
271	Macro syntax
272	Macro execution error
273	Illegal macro label
274	Macro parameter error
275	Macro definition too long
276	Macro recursion error
277	Macro redefinition not allowed
278	Macro header not found
280	Program error
281	Cannot create program
282	Illegal program name
283	Illegal variable name
284	Program currently running
285	Program syntax error
286	Program runtime error

表 3-5 は、機器の内部エラーを示すエラー・メッセージです。この種のエラーが発生した場合には、ハードウェアに問題が発生している可能性があります。

表 3-5: 内部機器エラー（DDEビット：3）

コード	メッセージ
300	Device-specific error
310	System error
311	Memory error
312	PUD memory lost
313	Calibration memory lost
314	Save/recall memory lost
315	Configuration memory lost
330	Self-test failed
350	Queue overflow （注：DDE ビットをセットしません。）

表 3-6 は、システム・イベントのメッセージです。この種のメッセージは、機器がメッセージが示す状態になったときに生成されます。

表 3-6: システム・イベントと問い合わせエラー

コード	メッセージ
401	Power on
402	Operation complete
403	User request
404	Power fail
405	Request control
410	Query INTERRUPTED
420	Query UNTERMINATED
430	Query DEADLOCKED
440	Query UNTERMINATED after indefinite response

表 3-7 は、コマンドの実行を中断させないレベルのエラーのワーニング・メッセージです。この種のメッセージは、予期した処理結果が得られない可能性があることをオペレータに知らせるものです。

表 3-7: ワーニング（EXEビット：4）

コード	メッセージ
500	Execution warning

表 3-8 は、本機器に特有のエラーのエラー・メッセージです。このメッセージは、本機器特有な機能を実行中に検出されるエラーを示しています。

表 3-8: 機器依存コマンド実行エラー（EXEビット：4）

コード	メッセージ
2000	File error
2001	Directory not empty
2002	Too much files
2003	File locked
2004	File already exists
2005	File already opened
2006	Invalid file type
2007	File type mismatch
2008	Internal memory full
2009	Invalid file format
2010	Comment error
2012	Invalid data in comment string
2020	Pattern data error
2021	To much pattern data
2022	Pattern data byte count error
2023	Pattern data load error
2024	Internal pattern memory full
2025	Invalid pattern size
2026	Invalid pattern data
2030	Sequence error
2032	Too much sequence data
2033	Invalid sequence repeat count
2034	Invalid sequence syntax
2035	Sequence load error
2036	Internal sequence memory full
2037	No sequence
2038	Invalid sequence number
2039	Sequence incomplete
2040	Data error
2041	Invalid data syntax
2042	Invalid data value
2050	Time error
2051	Invalid time syntax
2052	Invalid time value

表 3-8: 機器依存コマンド実行エラー（EXEビット：4）（続き）

コード	メッセージ
2060	Invalid group name
2061	Group name is empty
2062	Same name already exists
2063	Too much group
2064	Group name not found
2065	Group number is not found
2066	Invalid group data
2067	Invalid group syntax
2070	Invalid block position
2071	To much block
2072	Block already exists
2073	Block is not found
2074	Illegal block name
2075	Illegal block size
2076	Block name already exists
2077	Block is not defined
2078	Too much block data
2079	Invalid block syntax
2080	Import error
2081	Code table syntax error
2082	Too much table data
2100	Hardcopy error
2101	Hardcopy busy
2102	Hardcopy timeout error
2103	Sub Sequence eroor
2132	Too much sub sequence data
2133	Invalid sub sequence repeat count
2134	Invalid sub sequence syntax
2135	Sub Sequence load error
2136	Internal sub sequence memory full
2137	No sub sequence
2138	Sub Sequence line is not found
2139	Sub Sequence incomplete
2140	Too many sub sequence
2141	Sub Sequence name error
2200	Message error

表 3-9 は、本機器に特有の機器エラーのエラー・メッセージです。

表 3-9: 機器依存機器エラー

コード	メッセージ
3001	RS-232C input buffer overflow
3002	Internal divider error
3003	Internal EEPROM init error
3004	Internal EEPROM write error
3005	Internal EEPROM read error
3006	Internal FPGA configration error

付録 A　ASCII キャラクタ表

	0	1	2	3	4	5	6	7
0	0 NUL 0 0	20 DLE 16 10	40 SP 32 20	60 0 48 30	100 @ 64 40	120 P 80 50	140 ‘ 96 60	160 p 112 70
1	1 GTL SOH 1 1	21 LLO DC1 17 11	41 ! 33 21	61 1 49 31	101 A 65 41	121 Q 81 51	141 a 97 61	161 q 113 71
2	2 STX 2 2	22 DC2 18 12	42 ” 34 22	62 2 50 32	102 B 66 42	122 R 82 52	142 b 98 62	162 r 114 72
3	3 ETX 3 3	23 DC3 19 13	43 # 35 23	63 3 51 33	103 C 67 43	123 S 83 53	143 c 99 63	163 s 115 73
4	4 SDC EOT 4 4	24 DCL DC4 20 14	44 $ 36 24	64 4 52 34	104 D 68 44	124 T 84 54	144 d 100 64	164 t 116 74
5	5 PPC ENQ 5 5	25 PPU NAK 21 15	45 % 37 25	65 5 53 35	105 E 69 55	125 U 85 55	145 e 101 65	165 u 117 75
6	6 ACK 6 6	26 SYN 22 16	46 & 38 26	66 6 54 36	106 F 70 46	126 V 86 56	146 f 102 66	166 v 118 76
7	7 BEL 7 7	27 ETB 23 17	47 ’ 39 27	67 7 55 37	107 G 71 47	127 W 87 57	147 g 103 67	167 w 119 77
8	10 GET BS 8 8	30 SPE CAN 24 18	50 (40 28	70 8 56 38	110 H 72 48	130 X 88 58	150 h 104 68	170 x 120 78
9	11 TCT HT 9 9	31 SPD EM 25 19	51) 41 29	71 9 57 39	111 I 73 49	131 Y 89 59	151 i 105 69	171 y 121 79
A	12 LF 10 A	32 SUB 26 1A	52 * 42 2A	72 : 58 3A	112 J 74 4A	132 Z 90 5A	152 j 106 6A	172 z 122 7A
B	13 VT 11 B	33 ESC 27 1B	53 + 43 2B	73 ; 59 3B	113 K 75 4B	133 [91 5B	153 k 107 6B	173 { 123 7B
C	14 FF 12 C	34 FS 28 1C	54 , 44 2C	74 < 60 3C	114 L 76 4C	134 \ 92 5C	154 l 108 6C	174 | 124 7C
D	15 CR 13 D	35 GS 29 1D	55 – 45 2D	75 = 61 3D	115 M 77 4D	135] 93 5D	155 m 109 6D	175 } 125 7D
E	16 SO 14 E	36 RS 30 1E	56 . 46 2E	76 > 62 3E	116 N 78 4E	136 ^ 94 5E	156 n 110 6E	176 ~ 126 7E
F	17 SI 15 F	37 US 31 1F	57 / 47 2F	77 UNL ? 63 3F	117 O 79 4F	137 UNT – 95 5F	157 o 111 6F	177 DEL (RUBOUT) 127 7F
	アドレス・コマンド	ユニバーサル・コマンド	リスン・アドレス		トーク・アドレス		セカンダリ・アドレスまたはコマンド	

KEY

8 進 / 16 進

25	PPU
NAK	
15	21

GPIB コード
ASCII キャラクタ
10 進

付録 B　GPIB インタフェース仕様

インタフェース・ファンクション

インタフェース・ファンクションは、IEEE Std 488.1-1987 で定義されているもので、メッセージを送／受信したり、メッセージに従って機器を制御したりする機能です。表 B-1 は、本機器に組み込まれたインタフェース・ファンクションを示しています。括弧で囲まれた略号は、IEEE Std 488.1-1987 で定義され、広く使用されているインタフェース・ファンクションを示す記号です。

表 B-1: GPIBインタフェース・ファンクションと組み込みサブセット

インタフェース・ファンクション	組み込みサブセット	サブセットの機能
Acceptor Handshake（AH）	AH1	AH の機能をすべて満足します。
Source Handshake（SH）	SH1	SH の機能をすべて満足します。
Listener（L）	L4	基本リスナ、MTA によるアクティブ・リスナの解除、Listen Only モードなし
Talker（T）	T5	基本トーカ、シリアル・ポール、MLA によるアクティブ・トーカの解除、Talk Only モードなし。
Device Clear（DC）	DC1	DC の機能をすべて満足します。
Remote/Local（RL）	RL1	RL の機能をすべて満足します。
Service Request（SR）	SR1	SR の機能をすべて満足します。
Parallel Poll（PP）	PP0	サポートしません。
Device Trigger（DT）	DT1	DT の機能をすべて満足します。
Controller（C）	C0	サポートしません。
Electrical Interface	E2	3 ステート・ドライバ

- **Acceptor Handshake**（**AH**）— データを確実に受信するためのハンドシェイク機能です。この機能は、機器が次のデータの受信準備が完了するまで、データの送出開始と完了を遅らせます。

- **Source Handshake**（**SH**）— データを確実に転送するために、AH との間でハンドシェイクを行う機能です。この機能は、バイト単位にデータの送出開始と完了を制御します。

- **Listener**（**L**）— バスを通して、デバイス依存データを受信できる機能です。ただし、データを受信できるのは、受信指定されたアドレスを持つリスナに限ります。本機器でのアドレス指定は、1 バイトで行われます。

- **Talker**（**T**）— バスを通して、デバイス依存データを送出できる機能です。ただし、データを送出できるのは、送信指定されたアドレスを持つトーカに限ります。本機器でのアドレス指定は、1 バイトで行われます。

- **Device Clear**（**DC**）—　システムに接続された機器に対して、個々に、またはまとめて初期化を行う機能です。

- **Remote / Local**（**RL**）—　機器を操作する方法を選択します。機器の制御には、フロント・パネルの操作（ローカル・コントロール）による方法と、インタフェースを通して、コントローラから操作（リモート・コントロール）する方法の、2 つの方法があります。

- **Service Request**（**SR**）—　コントローラに対して、非同期のサービスを要求する機能です。

- **Controller**（**C**）—　バスを通して他の機器に、デバイス・アドレス、ユニバーサル・コマンド、アドレス・コマンドを送出できる機能です。デバイス・アドレス、ユニバーサル・コマンド、アドレス・コマンドについては、この後の“インタフェース・メッセージ”をご参照ください。

なお、Electorical Interface は、電気的インタフェースのタイプを示すもので、インタフェース・ファンクションには含まれません。記号としては E1 および E2 が使用され、インタフェースのタイプが、それぞれ 3 ステート・ドライバ、オープン・コレクタ・ドライバであることを示します。

インタフェース・メッセージ

表 B-2 に、本機器に組み込まれた GPIB ユニバーサル・コマンドとアドレス・コマンドを示します。

表 B-2: GPIB インタフェース・メッセージ

インタフェース・メッセージ	種別	組み込み
Device Clear（DCL）	UC	Yes
Local Lockout（LLO）	UC	Yes
Serial Poll Disable（SPD）	UC	Yes
Serial Poll Enable（SPE）	UC	Yes
Parallel Poll Unconfigure（PPU）	UC	No
Go To Local（GTL）	AC	Yes
Selected Device Clear（SDC）	AC	Yes
Group Execute Trigger（GET）	AC	Yes
Take Control（TCT）	AC	No
Parallel Poll Configure（PPC）	AC	No

※　UC、AC は、それぞれユニバーサル・コマンド、アドレス・コマンドを表します。

- **Device Clear**（**DCL**）—　DCLインタフェース・メッセージが組み込まれたすべての機器を初期化します。

- **Local Lockout**（**LLO**）—　ローカル状態に戻る機能を無効にします。この場合、フロント・パネルからの操作はできなくなります。

- **Serial Poll Enable**（**SPE**）—　サービス要求機能を持つすべての機器を、シリアル・ポール・モードにします。このモードの機器は、コントローラが送出するトーク・アドレスを受け取ると、コントローラにステータス・バイトを戻します。コントローラは、シリアル・ポーリングによって、サービス要求を行った機器を特定することができます。

- **Serial Poll Disable**（**SPD**）—　サービス要求機能を持つすべての機器に対して、シリアル・ポール・モードを解除し、通常の動作モードに戻します。

- **Go To Local**（**GTL**）—　リモート・コントロール状態を解除して、ローカル・コントロール状態に戻します。

- **Select Device Clear**（**SDC**）—　DCLインタフェース・メッセージが組み込まれた機器を初期化します。

- **Group Execute Trigger**（**GET**）—　特定の機器、またはあるグループの機器に対してトリガをかけ、プログラムされた機能を実行します。

- **Take Control**（**TCT**）—　バスを管理しているコントローラから、コントローラの機能を有する他の機器に、バス管理権を移します。

- **Parallel Poll Configure**（**PPC**）—　PPC コマンドに続いて送出される PPE（Parallel Poll Enable）コマンドと PPD（Parallel Poll Disable）コマンドに従って、パラレル・ポールのモードを設定・解除します。

付録 C　デフォルト設定値

表 C-1 に、コマンドのデフォルト設定値を示します。

表 C-1: デフォルト設定値

ヘッダ	設定値
DATA サブシステム・コマンド	
DATA:MSIZe	1000
DEBUG サブシステム・コマンド	
DEBug:SNOop:DELAy:TIME	0.2
DEBug:SNOop:STATe	0
DIAGNOSTIC サブシステム・コマンド	
DIAG:SELect	ALL
DISPLAY サブシステム・コマンド	
DISPlay:BRIGhtness	0.7
DISPlay:CLOCk	0
DISPlay:DIMmer	0
DISPlay:ENABle	1
DISPlay:MENU[:NAME]	EDIT
DISPlay:MENU:STATe	1
HARDCOPY サブシステム・コマンド	
HCOPy:FORMat	BMP
HCOPy:PORT	DISK
MEMORY サブシステム・コマンド	
MMEMory:CATalog:ORDer	NAME1
MODE サブシステム・コマンド	
MODE:STATe	REPEAT
MODE:UPDate	AUTO
OUTPUT サブシステム・コマンド	
OUTPut:ELEVel	1.4 [V]
OUTPut:ILEVel	1.4 [V]
OUTPut:CH<n>:DELAy	0.0 [ns]
OUTPut:CH<n>:DESKew	0.0 [ns]
OUTPut:CH<n>:FALl	FAST
OUTPut:CH<n>:HIGH	1.5 [V]
OUTPut:CH<n>:INHibit	0
OUTPut:CH<n>:LOW	0.0 [V]
OUTPut:CH<n>:RISe	FAST
OUTPut:CHCLK:FALl	FAST
OUTPut:CHCLK:HIGH	1.5 [V]
OUTPut:CHCLK:LOW	0.0 [V]
OUTPut:CHCLK:RISe	FAST

表 C-1: デフォルト設定値（続き）

ヘッダ	設定値
SOURCE サブシステム・コマンド	
SOURce:OSCillator:EXTernal:FREQuency	1.0E+8 [Hz]
SOURce:OSCillator[:INTernal]:FREQuency	1.0E+8 [Hz]
SOURce:OSCillator[:INTernal]:PLLlock	ON
SOURce:OSCillator:SOURce	INTERNAL
SOURce:EVENT:ENABle	1
SYSTEM サブシステム・コマンド	
SYSTem:PPAUse	1
SYSTem:SECurity:STATe	0
TRIGGER サブシステム・コマンド	
TRIGger:IMPedance	HIGH
TRIGger:INTERVal:STATe	OFF
TRIGger:INTERVal:TIME	10.0 [s]
TRIGger:LEVel	1.4 [V]
TRIGger:SLOPe	POSITIVE
TRIGger:SOURce	EXTERNAL
その他のコマンド	
DESE	255
*ESE	0
HEADer	1
LOCk	NONE
*PSC	1
*SRE	0
VERBose	1

付録 D　コマンド索引

A

C

D

E

F

H

I

L

M

O

P

R

S

T

U

V

W

保証規定

保証期間（納入後 1 年間）内に通常の取り扱いによって生じた故障は無料で修理します。

1. 取扱説明書、本体ラベルなどの注意書きに従った正常な使用状況で保証期間内に故障した場合には、販売店または当社に修理をご依頼下されば無料で修理いたします。なお、この保証の対象は製品本体に限られます。
2. 転居、譲り受け、ご贈答品などの場合で販売店に修理をご依頼できない場合には、当社にお問い合せください。
3. 保証期間内でも次の事項は有料となります。
   - 使用上の誤り、他の機器から受けた障害、当社および当社指定の技術員以外により修理、改造などから生じた故障および損傷の修理
   - 当社指定以外の電源（電圧・周波数）使用または外部電源の以上により故障および損傷の修理
   - 移動時の落下などによる故障および損傷の修理
   - 火災、地震，風水害、その他の天変地異，公害、塩害、異常電圧などによる故障および損傷の修理
   - 消耗品、付属品などの消耗による交換
   - 出張修理（ただし故障した製品の配送料金は、当社負担）
4. 本製品の故障またはその使用によって生じた直接または間接の損害について、当社はその責任を負いません。
5. この規定は、日本国内においてのみ有効です。（ This warranty is valid only in Japan. ）
   - この保証規定は本書に明示された条件により無料修理をお約束するもので、これによりお客様の法律上の権利を制限するものではありません。
   - ソフトウェアは、本保証の対象外です。
   - 保証期間経過後の修理は有料となります。詳しくは、販売店または当社までお問い合せください。

お問い合わせ

製品についてのご相談・ご質問につきましては、下記までお問い合わせください。

お客様コールセンター

TEL 03-6714-3010　　FAX 0120-046-011

東京都港区港南 2-15-2　品川インターシティ B 棟 6F　〒108-6106

電話受付時間/9:00～12:00・13:00～19:00　月曜～金曜（休祝日を除く）

E-Mail: ccc.jp@tektronix.com

URL:　http://www.tektronix.co.jp

修理・校正につきましては、お買い求めの販売店または下記サービス受付センターまでお問い合わせください。

（ご連絡の際には，型名、故障状況を簡単にお知らせください）

サービス受付センター

TEL 0120-74-1046　　FAX 0550-89-8268

静岡県御殿場市神場 143-1　〒412-0047

電話受付時間/9:00～12:00・13:00～19:00　月曜～金曜（休祝日を除く）