プログラマ・マニュアル

Tektronix

AWG2000シリーズ
任意波形ゼネレータ

070-A518-50

www.tektronix.com

Copyright © Tektronix Japan, Ltd. All rights reserved.

当社の製品は、米国その他各国における登録特許および出願中特許の対象となっています。本書の内容は、すでに発行されている他の資料の内容に代わるものです。また製品仕様は、予告なく変更する場合がありますので、予めご了承ください。

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

Tektronix、Tek は Tektronix, Inc.の登録商標です。
また、本マニュアルに記載されている、その他の全ての商標は、各社所有のものです。

はじめに

本マニュアルでは、GPIB インタフェースまたは RS-232C インタフェースを通してAWG シリーズ (AWG2005 型、AWG2010 型、AWG2011 型、AWG2020 型、AWG2021 型、AWG2040 型、AWG2041 型) をリモート制御するための方法を説明します。各機器の詳細は、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参照ください。

なお、AWG2010 / 20 型は、ファームウェア・バージョン 3.00 以降に有効です。

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。

サンプル・プログラム

本マニュアルには、サンプル・プログラムを記録したフロッピー・ディスクが添付されています。本マニュアルの第4章に、このプログラムの使用方法とソース・リストが掲載されています。

目　次

第 1 章　概　要

第 2 章　シンタックスとコマンド

第 3 章　ステータス／イベント

第 4 章　プログラム例

付　録

第1章　概　要

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用いられています。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。GPIB のインタフェース・レイヤを図 1-1 に示します。

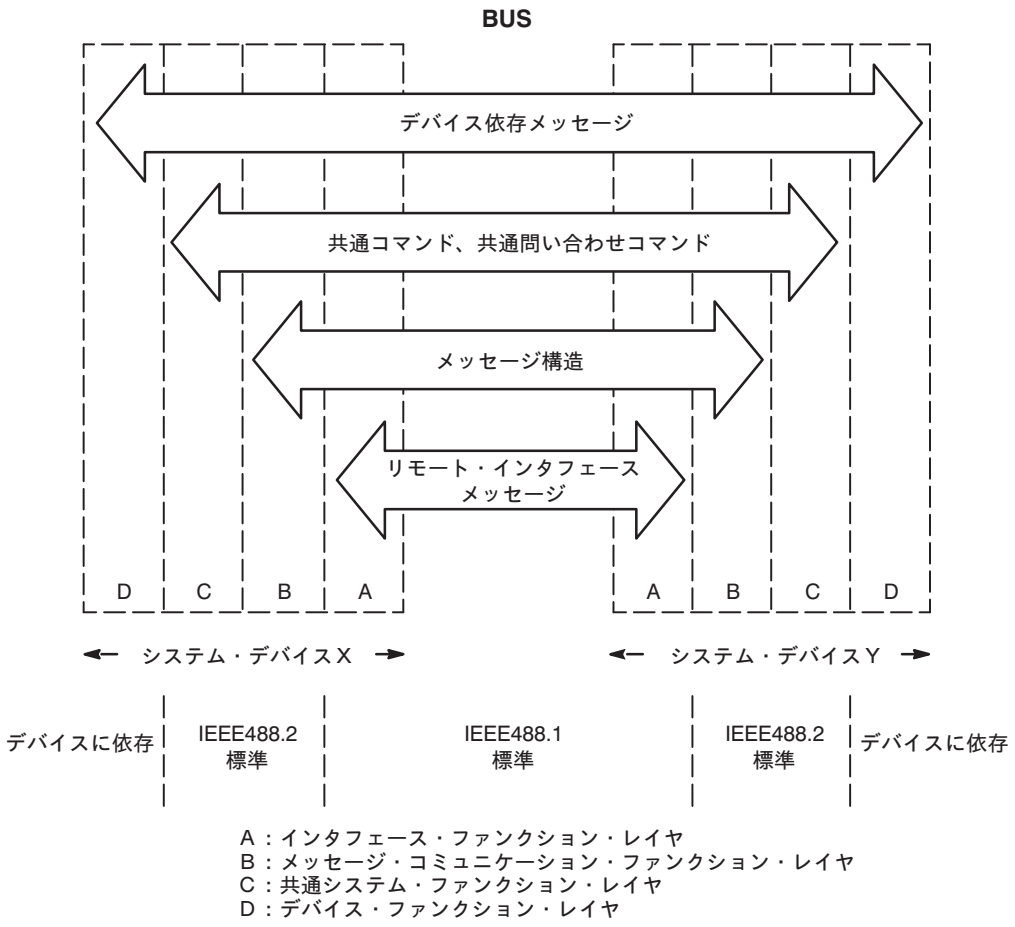


図 1-1：GPIBシステムのインタフェース・レイヤ

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。

本マニュアルでは、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、次の項目で説明します。

- 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」の項を参照ください。

本マニュアルに添付のフロッピー・ディスクで、GPIBインタフェースを使用した場合のサンプル・プログラムが提供されます。第 4 章のプログラム・リストを参照することにより、素早くコマンドやインタフェースの使用方法、あるいはプログラミングの方法を理解できるようになるものと思われます。また添付のソース・プログラムは、コンパイル、リンクすることにより、アプリケーション・プログラムとして直接使用することもできます。

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 を参照ください。

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しようとする場合、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てください。

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、以下で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。例えば、IBM PC あるいは 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、外部コントローラとすることができます。

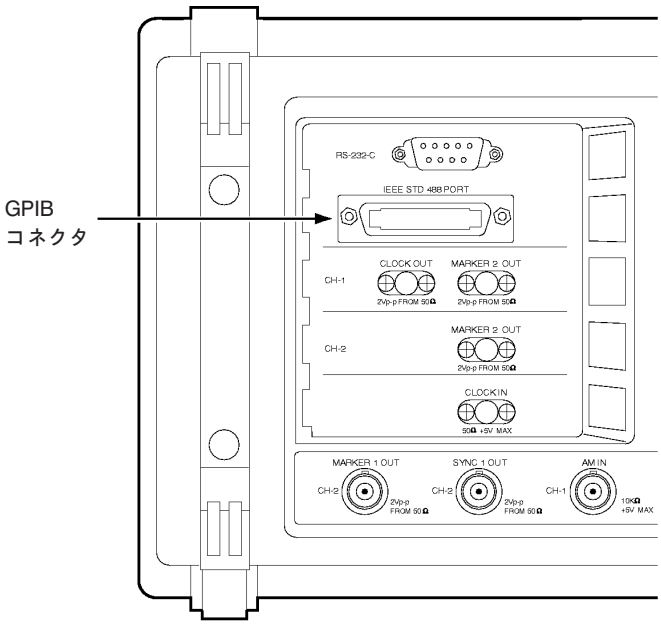


図 1-2：GPIBコネクタ

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 つの機器から他の全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。

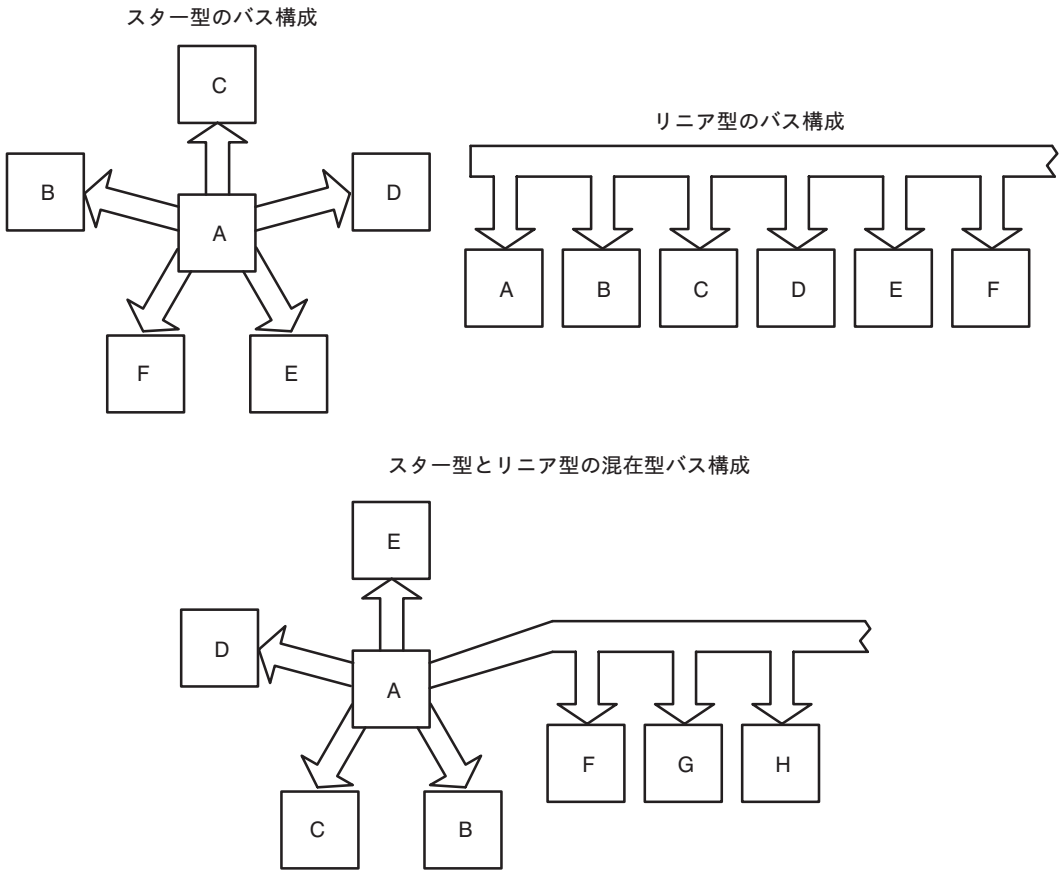


図 1-3：GPIB の接続形態

使用制限

GPIB システムへの接続は、下記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

以下の手順で GPIB パラメータを設定してください。

☐ **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 (図 1-4 参照)。

☐ **手順 2**：**GPIB** ボトム・キーを押し、**GPIB** サイド・メニューを表示します。本サイド・メニューでは、以下の GPIB パラメタを設定します。

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 3**：**Talk/Listen**、**Address** サイド・キーを押し、次に、ロータリ・ノブを左右に回して、目的のアドレスを表示します。これでコミュニケーション・モードが **Talk/Listen** に、また目的のプライマリ・アドレスが設定されます。

☐ **手順 4**：**Misc** ボトム・キーを押し、さらに **Config**..サイド・キーを押して **Config** サブ・メニューを表示します。

☐ **手順 5**：**Remote Port** サイド・キーを 1 回か 2 回押して **GPIB** の項目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。

これらの設定が完了すると、GPIB インタフェースをリモート・インタフェースとして使用するための準備が整い、管面のステータス表示ラインに GPIB インジケータが点灯します (図 1-11 参照)。

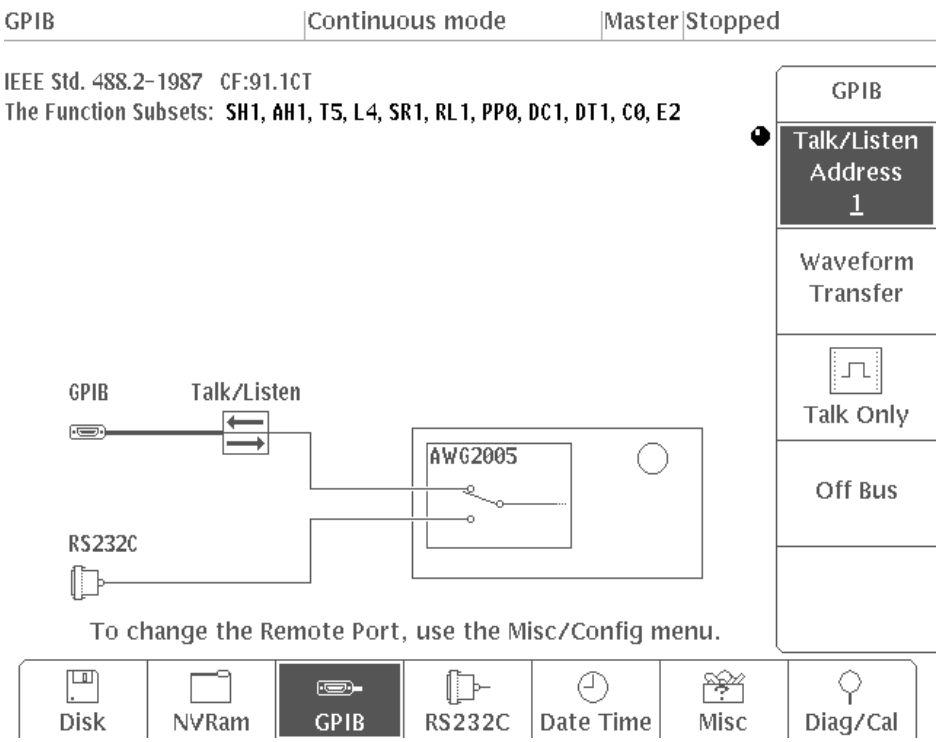


図 1-4：GPIBパラメタ設定メニュー (AWG2005型の場合)

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) 。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、このインタフェースではシリアルに転送します。しかも送信と受信が独立のバスになっていますので双方向同時にメッセージを転送することもできます。

本機器は、リア・パネルに、9 ピン D タイプ・コネクタを備えています。またオプショナル・アクセサリの 9 ピン・オス型－25 ピン・オス型の変換ケーブル (P/N 015-0554-00) を使用すると、9 ピン・メス型または 25 ピン・メス型のコネクタを持った外部コントローラあるいは延長ケーブルでも接続することができます。

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ますので、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。

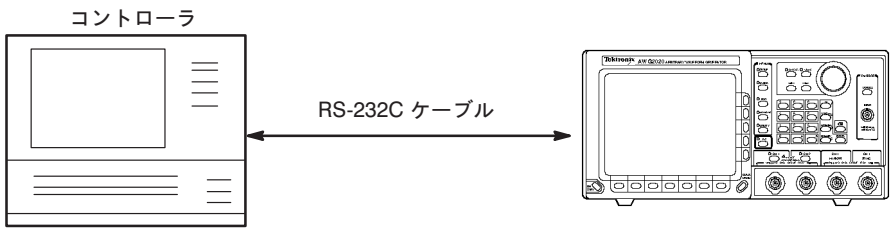


図 1-5：RS-232C ポイント・ツー・ポイント接続

AWG2005/10/11/20/21型の場合、オプションによっては RS-232C インターフェースが付かないことがあります。

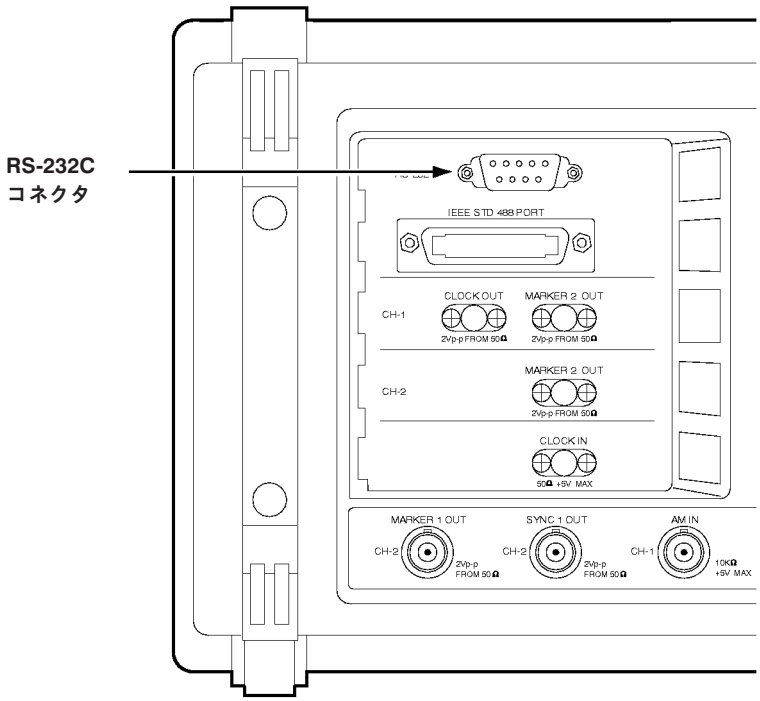


図 1-6 : RS-232C コネクタ

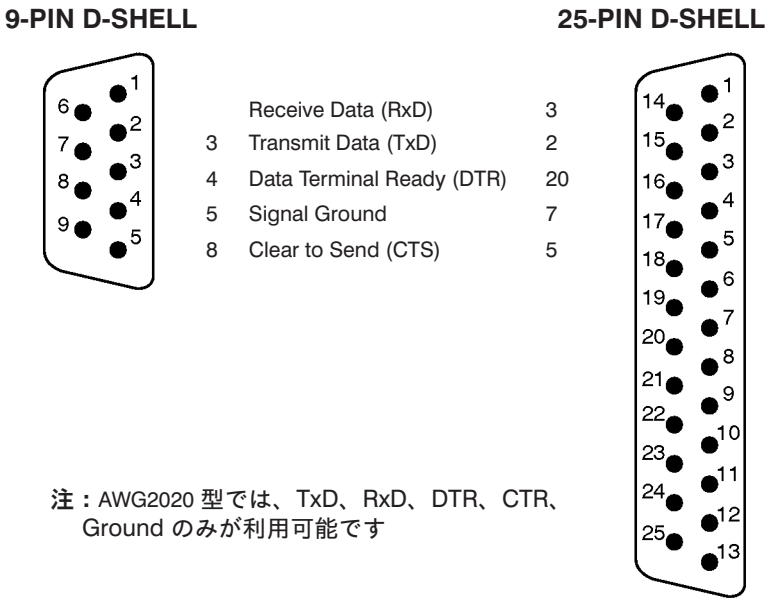
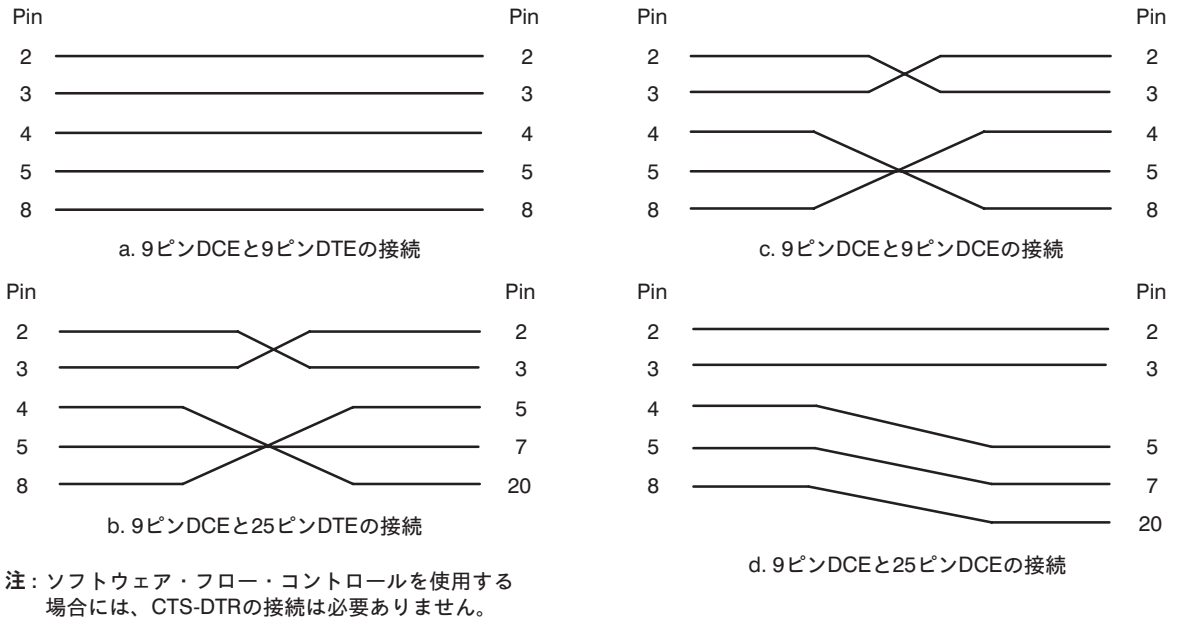


図 1-7 : 9 ピン、25 ピン D タイプ・コネクタのピン配置



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

以下の手順で RS-232C パラメータを設定してください。

☐ **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 (図 1-9 参照)。

☐ **手順 2**：**RS232C** ボトム・キーを押し、**RS232C** サイド・メニューを表示します。本サイド・メニューでは、以下の RS-232C パラメタを設定します。

Baud Rate — 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。

Data Bits — 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。

Parity — エラー・チェック・ビットを設定します。設定可能なパリティは、**None**、**Even**、**Odd** です。

Stop Bits — 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。

Flagging — 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 **None** (制御なし) が選択できます。

☐ **手順 3**：上記パラメタに対応するサイド・ソフト・キーを押し、次に、ロータリ・ノブを左右に回して、目的の項目を表示します。これで1つのパラメタに対する設定が完了しますので、必要なパラメタ数だけこの操作を繰り返して、全パラメタの設定を行います。

☐ **手順 4**：**Misc** ボトム・キー、**Config**..サイド・キーの順で押して、**Config** サブ・メニューを表示します。

☐ **手順 5**：**Remote Port** サイド・ソフト・キーを 1 回か 2 回押して **RS232C** の項目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。

これらの設定が完了すると、RS-232C インタフェースをリモート・インタフェースとして使用するための準備が整い、管面のステータス表示ラインに RS-232C インジケータが点灯します (図 1-9 参照)。

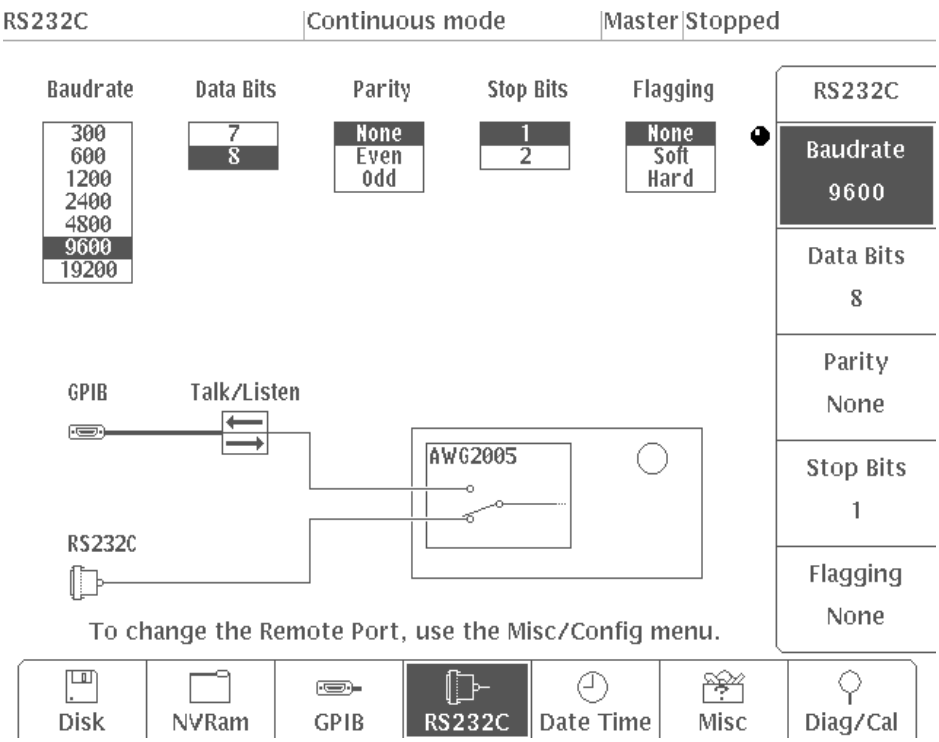


図 1-9：RS-232Cパラメタの設定メニュー (AWG2005型の場合)

設定の確認

Status の管面表示で GPIB/RS-232C の設定状態を確認することができます (図 1-10 参照) 。以下の手順でメニューを表示します。

☐ **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**：**Misc** ボトム・キーを押し、**Misc** サイド・メニューを表示します。

☐ **手順 3**：**Status**.サイド・キーを押して、**Status** サイド・メニューを表示します。

☐ **手順 4**：**System** サイド・キーを押して、**System** サブ・メニューを表示します (図 1-10 参照) 。

この管面表示では、以下のステータスが確認できます。

- **Address** — GPIB プライマリ・アドレス
- **Configure** — コミュニケーション・モード
- **PSC** — 電源投入時自動設定機構 (PSC: Power-on Status Clear) 。第 2 章の *PSCコマンドの説明を参照ください。
- **Header** — レスポンス・メッセージ中のコマンド・ヘッダの取り扱い。第 2 章の「コマンド・シンタックス」の「レスポンス・メッセージ」の項、または第 2 章の HEADer コマンドの説明を参照ください。
- **Verbose** — レスポンス・メッセージ中のコマンド・ヘッダの省略形。第 2 章の VERBose コマンドの説明を参照ください。
- **Data** — 波形転送に関する設定。第 2 章の DATA:SOURce、DATA:DESTination、DATA:ENCDG コマンドの説明を参照ください。
- **Debug** — デバック用パラメータに関する設定。DEBug コマンドの説明を参照ください。

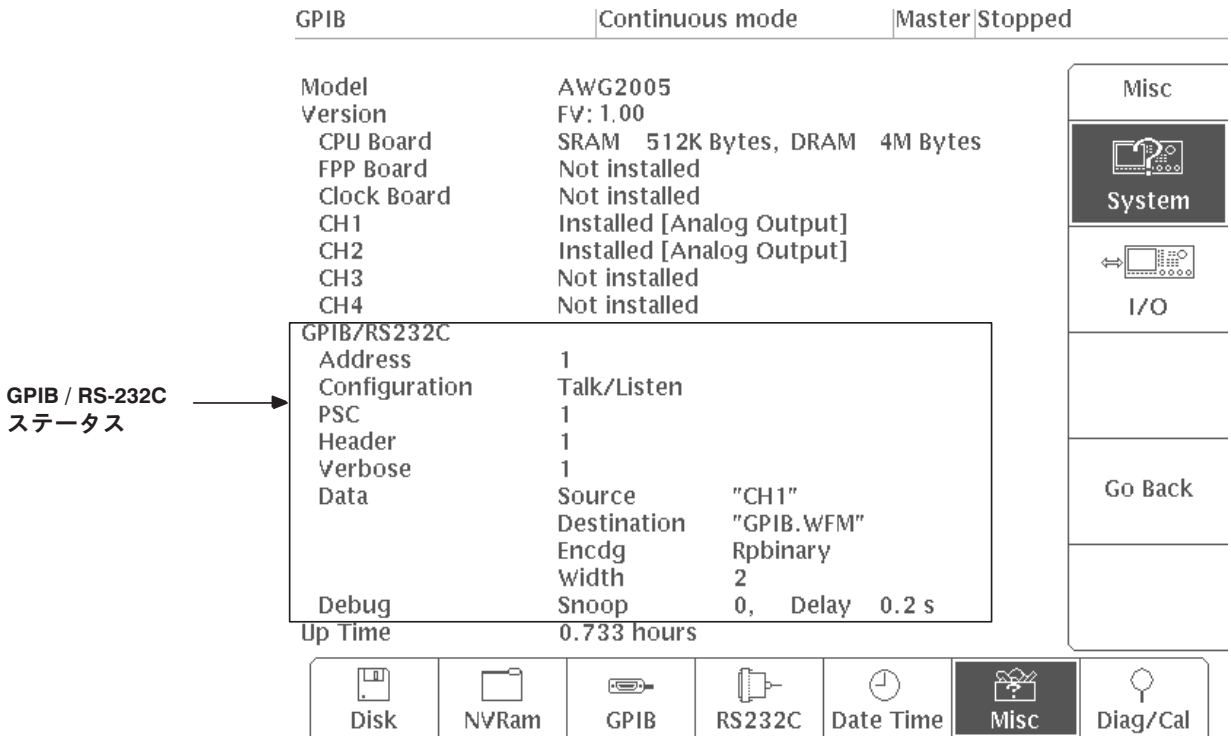


図 1-10：設定状態の確認

操　作

まず本機器のリア・パネルにある主電源スイッチを ON にし、次に、フロント・パネルの **ON/STBY** スイッチを ON にして初期メニュー画面が現れるのを待ちます。

電源投入後、本機器は、メニューおよびフロント・パネルからの全ての制御を受け付けます。この状態ではさらに、外部コントローラからリモート・インタフェースを通して送られてくるコマンドも同時に受け付けます。よってユーザは、ローカル・コントロールまたはリモート・コントロールの切り換えを意識することなくフロント・パネルからでも外部コントローラからでも、本機器を制御することができます。

図 1-11 に、管面に表示されるステータス・ラインを示します。リモート・コントロール・モードでは、**GPIB & RS232C** サブ・エリアの各種インジケータが下記の条件のときに表示されます。

GPIB — GPIB インタフェースがリモート・インタフェースとして選択されている場合に表示されます。

RS232C — RS-232C インタフェースがリモート・インタフェースとして選択されている場合に表示されます。

SRQ — 本機器が GPIB インタフェースを通して外部コントローラに SRQ を発行した場合に表示されます。

LOCK — フロント・パネル・コントロールがロック状態のときに表示されます。

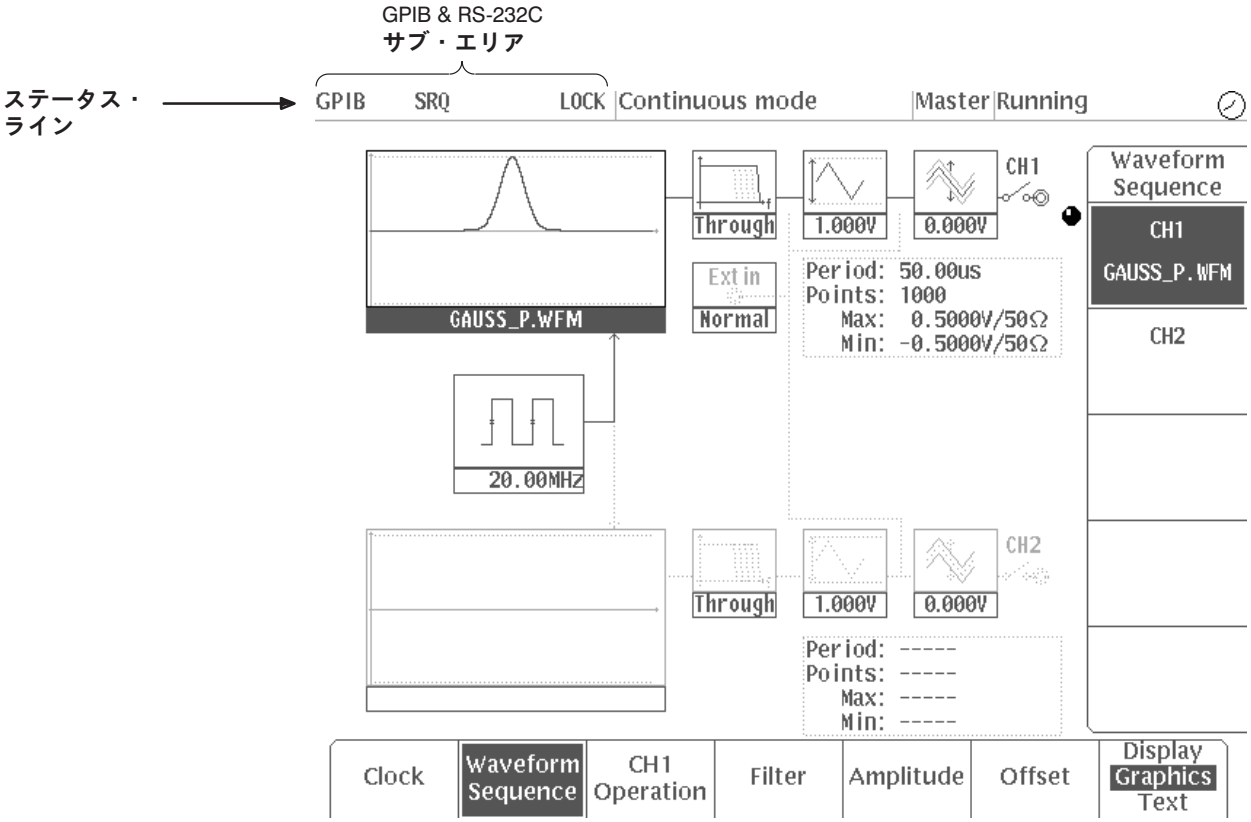


図 1-11：ステータス・ラインとGPIB＆RS232Cサブ・エリア

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。以下では、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

表 2-1 に示す BNF 記法を用いて、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。
{ }	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・メッセージとは、外部コントローラから本機器に転送される1つの設定コマンドを言い、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するように指示します。また、問合せコマンド・メッセージとは、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を言い、機器に対して、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。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。

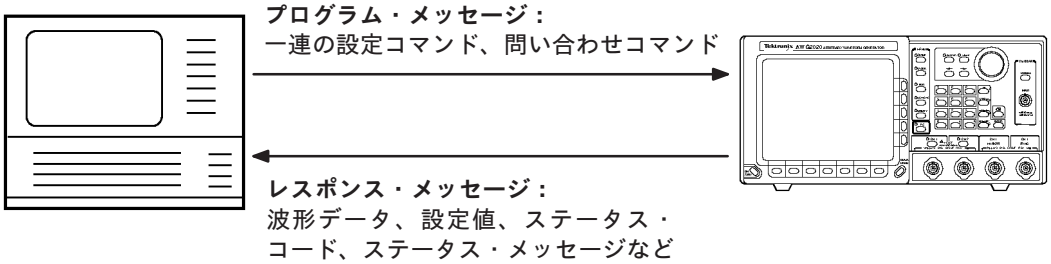


図 2-1：コマンド、問合せコマンド構造のフロー・チャート

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（本マニュアルでは、それぞれコマンドおよび問合せコマンドとよびます）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 2-2 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、後述の「ヘッダ」の項で詳しく説明します。

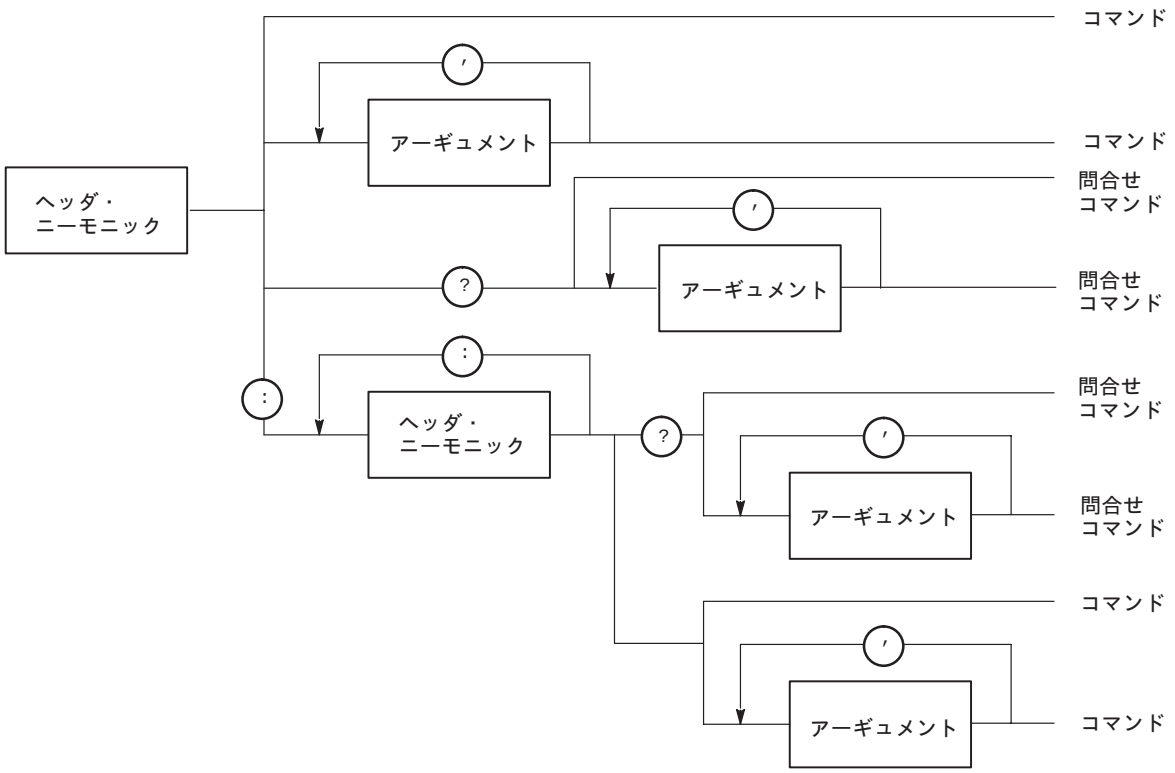


図 2-2：コマンド、問合せコマンド構造のフロー・チャート

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、全て 7 ビット ASCII コードを使用します。なお付録 A に ASCII キャラクタ・コード表を添付します。

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:ALOAD:MSIS、        OUTPUT:CH1:STATE
```

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。

```
DIAG:SELECT  ALL
MODE  BURST  4000
```

上記の場合、DIAG:SELECTとMODE がコマンド・ヘッダで、ALL と“BURST, 4000”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。上記の例では、“BURST, 4000”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の詳細については、後述「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。

なお本マニュアルでは、特別な場合を除き、スペースとホワイト・スペースの双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、次のような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

ANSI/IEEE Std.488.2 - 1987 で規定される NR1、NR2、NR3 と呼ばれる 3 つのタイプの10進数が利用できます（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。例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>
```

となります。

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>]...　または
　#0[<configuous 8–bit data byte>] <terminator>

それぞれ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。

<terminator>::=　ソフトウェア LF とハードウェア EOI（<LF><&EOI>と表記します）。

例　　#16AB4ZLT
例　　#0EHTGUILEDOM <LF> <&EOI>

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。例えば、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、次の通りです。

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、以下の通りです。

SI プリフィックスを表すシンボル	m/M	k/K	m/M	g/G
対応すべきベキ乗	10^{-3}	10^{+3}	10^{+6}	10^{+9}

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。例えば、下記の例は同じ結果となります。

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。

ポッド / チャンネル表現

コマンドまたは問合せコマンドでは、OUTPut:POD<s>:CH<n> ヘッダ・ニーモニックを使用してポッドおよびチャンネルを指定します。<s> は A 、B または C で、指定するポッドの接続されているパターン・データ出力コネクタを表します。また <n> は 0 ～ 11 の数値で、指定するチャンネルを表します。

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、以下のような 6 つのコマンドあるいは問合せコマンドに分類できます。

- 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　MMEMORY?
　　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　OUTPUT:CH1:STATE ON
DISK:FORMAT:TYPE HD1

ここで、OUTPUT:CH1:STATE、DISK:FORMAT:TYPE が複合コマンド・ヘッダで、ON と HD1 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　DISK:DIRECTORY?
MEMORY:CATALOG:ALL?
MEMORY:COMMENT?　”WAVE01.WFM”

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 をサポートする全ての機器に共通に使用されるものです。

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。

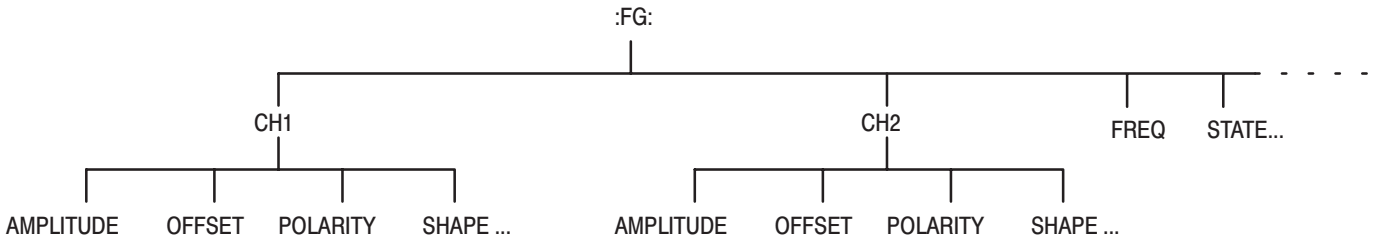
＜ヘッダ・ニーモニック＞?　[＜アーギュメント＞［、＜アーギュメント＞］...]

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 をサポートする全ての機器に共通に使用されるものです。

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ますので、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。

:FG:CH1:AMPLITUDE 3.5 ; :FG:CH1:OFFSET 1.5 ; :FG:CH1:POLARITY INVERTED ;
:FG:CH1:SHAPE SQUARE

2 番目から 4 番目まで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始まっていることを示すもので、省略することはできません。しかしながら、本機器のコマンド・パーザは、上位のヘッダ・パス:FG:CH1: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ますので、2番目以降のコマンド・ヘッダには完全なパスを記述する必要はありません。このため、上記のプログラム・メッセージは、次のように書き換えることができます。

:FG:CH1:AMPLITUDE 3.5 ; OFFSET 1.5 ; POLARITY INVERTED ; SHAPE SQUARE

以下は、正しい記述の例です。

:FG:CH1:AMPLITUDE 3.5 ; OFFSET 1.5 ; :FG:CH2:AMPLITUDE 3.5 ; OFFSET 1.5
:FG:STATE ON ; CH1:SHAPE SQUARE ; POLARITY INVERTED
:FG:CH1:AMPLITUDE 3.5 ; *SRE?; OFFSET 1.5 ; POLARITY INVERTED ;
SHAPE SQUARE

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。

以下は、誤った記述の例です。

:FG:CH1:AMPLITUDE 5.0 ; FG:CH2:AMPLITUDE 5.0
:FG:CH1:SHAPE SQUARE ; CH2:SHAPE SQUARE
:FG:CH1:AMPLITUDE 5.0 ; STATE ON

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。例えば、*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を 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となりますので、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るようになります。

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、値だけが返送されることになります。

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 OFF の場合	ヘッダが ON の場合
FG:CH1:AMPLITUDE?	5.000	:FG:CH1:AMPLITUDE 5.000
DIAG:SELECT?	WMEMORY	:DIAG:SELECT WMEMORY

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。例えば、

```
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

予約語と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、後述の「コマンド詳細説明」(2-25 ページ) 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。次のように記述されている場合、

```
CLOCk:SOURce INTernal
```

次のように省略することができます。

```
CLOCK:SOURCE  INTERNAL
CLOC:SOUR  INTER
CLOC:SOUR  INT
```

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。

シンタックス・ダイアグラム

本章の「コマンド詳細説明」(2-25 ページ) の項では、コマンドのシンタックスを BNF 記法の他にシンタックス・ダイアグラムを使用して説明します。シンタックス・ダイアグラムは、下記のようなシンボルで表現されます。

- 楕円は、コマンド／問い合わせコマンドのヘッダ、ダブルクォートで囲まれていない文字列（ラベル）のような文法要素を含みます。コマンド名／問い合わせコマンド名やラベルは、必要最小限の文字の並びまで省略可能です。
- 円は、コンマ（, ）、コロン（:）疑問符（？）のようにセパレータなど特殊な文法要素を含みます。
- 長方形は、定義項目を表します。
- 各要素は、矢印で表されるフロー・ラインで結ばれます。このフロー・ラインをたどることにより、各文法要素が配置される順番を決めることができます。フロー・ラインが平行に流れている場合には選択を表し、フロー・ラインが文法要素をパスしている場合には省略可能を表し、ループになっている場合には繰り返しを表します。

注：　シンタックス・ダイアグラムの中では、コマンドのアーギュメントに添付可能な単位とSIプリフィックスについての明確な記述は行なっていません。

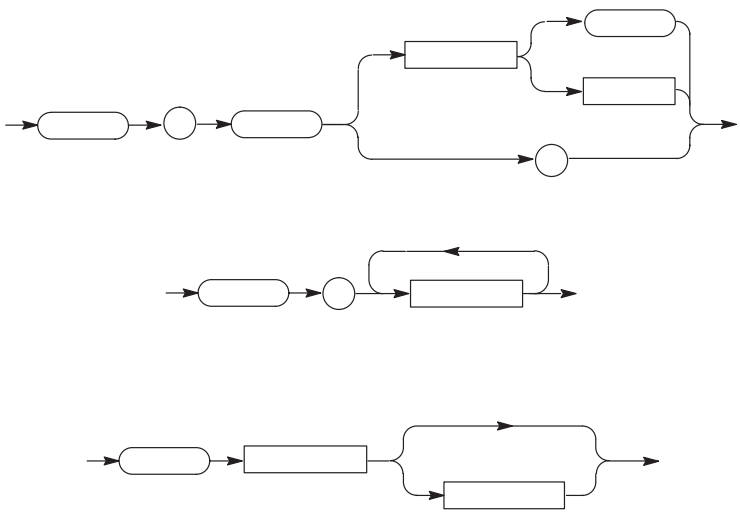


図 2-3：典型的なシンタックス・ダイアグラムの例

コマンド・セット

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、12 個のコマンド・グループに分類されます。

CALIBRATION＆DIAGNOSTICコマンド

CALIBRATION & DIAGNOSTIC グループのコマンドは、機能ごとに分類されたキャリブレーション・ルーチンまたはセルフ・テスト・ルーチンを選択し、実行します。

表 2-4：CALIBRATION & DIAGNOSTIC コマンド・グループ

ヘッダ		内　容
SELFcal:SELect	(?)	キャリブレーション・ルーチンの選択
SELFcal:STATe		キャリブレーションの実行
SELFcal:RESUlt?		キャリブレーション実行結果の問い合わせ
SELFcal?		キャリブレーションに関する設定中の全設定値の問い合わせ
*CAL?		キャリブレーションの実行
DIAG:SELect	(?)	セルフ・テスト・ルーチンの選択
DIAG:STATe		セルフ・テストの実行
DIAG:RESUlt?		セルフ・テスト実行結果の問い合わせ
DIAG?		セルフ・テストに関連する設定中の全設定値の問い合わせ
*TST?		セルフ・テストの実行

DISPLAYコマンド

DISPLAY グループのコマンドは、フロント・パネルのキーおよびノブに対応する機能の実行、管面の輝度調整などを行います。

表 2-5：DIAPLY コマンド・グループ

ヘッダ		内　容
ABSTouch		キーおよびノブに対応する機能の実行
DISPlay:BRIGhtness	(?)	管面の輝度調整
DISPlay:CATalog?		カタログ表示の条件を問い合わせ
DISPlay:CATalog:ORDer	(?)	カタログのファイル表示の順番を選択、問い合わせ
DISPlay:CLOCk(?)		日付けと時刻の表示と問い合わせ
DISPlay:MENU:SETUp:FORMat (?)		表示フォーマットの設定と問い合わせ
DISPlay:MENU:SETUp?		表示フォーマットの問い合わせ
DISPlay:MENU?		表示フォーマットの問い合わせ
DISPlay:MESSage	(?)	メッセージの消去と問い合わせ
DISPlay:MESSage:SHOW	(?)	メッセージの表示
DISPlay?		DISPLAY に関係する設定中の全設定値の問い合わせ

FGコマンド

FG グループのコマンドは、標準関数波形の選択、パラメタの設定（最大電圧レンジ、オフセット、極性など）、周波数の選択など FG モードに関係する設定を行います。また FG モードの ON/OFF も行います。

表 2-6：FG コマンド・グループ

ヘッダ	内　容
FG:CH<x>:AMPLitude(?)	標準関数波形の最大電圧レンジ設定
FG:CH<x>:OFFSet(?)	標準関数波形のオフセット設定
FG:CH<x>:POLarity(?)	標準関数波形の極性 (ポラリティ) 設定
FG:CH<x>:SHAPe(?)	標準関数波形の選択
FG:CH<x>?	FGモードに関する設定中の全設定値の問い合わせ
FG:FREQuency(?)	標準関数波形の周波数設定
FG:STATe(?)	FG モードの ON/OFF
FG?	FG モードに関する設定中の全設定値の問い合わせ

HARDCOPYコマンド

HARDCOPY グループのコマンドは、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 2-7：HARDCOPY コマンド・グループ

ヘッダ		内　容
HCOPy	(?)	ハードコピーの開始と停止及びハードコピーの全設定の問い合わせ
HCOPy:DATA?		ハードコピー・イメージ・データの問い合せ
HCOPy:FORMat	(?)	ハードコピーの出力フォーマットの選択
HCOPy:PORT	(?)	ハードコピーの出力ポートの選択

MEMORYコマンド

MEMORY グループ・コマンドには、内部メモリに関する操作、フロッピー・ディスクや NVRAM に関する操作が含まれます。MEMory、DISK、MMEMory サブ・グループのコマンドは、それぞれ内部メモリ、フロッピー・ディスク、マス・メモリの操作を行います。マス・メモリは、フロッピー・ディスクか NVRAM のいずれかを表しますので、マス・メモリ上のファイルを操作する場合には、まず初めに MMEMory:MSIS コマンドを使用してカレント・マス・メモリとして DISK または NVRAM のいずれかを選択してください。

表 2-8：MEMORY コマンド・グループ

ヘッダ		内　容
DISK:FORMat:TYPE	(?)	初期化時フロッピー・ディスク・フォーマット・タイプの選択
DISK:FORMat?		フォーマット・タイプの問い合わせ
DISK:FORMat:STATe		フロッピー・ディスクの初期化
DISK:CDIRectory		カレント・ワーキング・ディレクトリの変更
DISK:DIRectory?		カレント・ワーキング・ディレクトリの問い合わせ
DISK:MDIRectory		新規ディレクトリの作成
DISK?		DISKに関する設定中の全設定値の問い合わせ
MEMory:COPY		ファイルのコピー
MEMory:DELete		ファイルの削除
MEMory:REName		ファイルの名称変更
MEMory:COMMent	(?)	ファイルへのコメントの書き込み
MEMory:LOCk	(?)	ファイルのロック
MEMory:CATalog:ALL?		全ファイルのファイル情報参照
MEMory:CATalog:AST?		全オート・ステップ・ファイルのファイル情報参照
MEMory:CATalog:CLK? (AWG2005型)		全クロック・スイープ・ファイルの情報参照

表 2-8：MEMORY コマンド・グループ(続)

ヘッダ		内　容
MEMory:CATalog:EQU?		全イクエーション・ファイルのファイル情報参照
MEMory:CATalog:SEQ?		全シーケンス・ファイルのファイル情報参照
MEMory:CATalog:WFM?		全波形ファイルのファイル情報参照
MEMory:CATalog?		全ファイルのファイル情報参照
MEMory:FREE:ALL?		内部メモリ使用状況の問い合わせ
MEMory:FREE?		内部メモリ使用状況の問い合わせ
MEMory?		全ファイルのファイル情報と内部メモリの使用状況の問い合わせ
MMEMory:LOAD		マス・メモリから内部メモリへファイルのロード
MMEMory:DELete		ファイルの削除
MMEMory:MSIS	(?)	カレント・マス・メモリの選択
MMEMory:REName		ファイルの名称変更
MMEMory:SAVE		内部メモリからマス・メモリへファイルのセーブ
MMEMory:LOCk	(?)	ファイルのロック
MMEMory:ALOad:MSIS	(?)	オート・ロード用のマス・メモリの選択
MMEMory:ALOad:STATe	(?)	オート・ロードの実行制御
MMEMory:ALOad?		オート・ロードに関する全設定の問い合わせ
MMEMory:CATalog:ALL?		全ファイルのファイル情報参照
MMEMory:CATalog:AST?		全オート・ステップ・ファイルのファイル情報参照
MMEMory:CATalog:CLK? (AWG2005型)		全クロック・スイープ・ファイルの情報参照
MMEMory:CATalog:EQU?		全イクエーション・ファイルのファイル情報参照
MMEMory:CATalog:SEQ?		全シーケンス・ファイルのファイル情報参照
MMEMory:CATalog:WFM?		全波形ファイルのファイル情報参照
MMEMory:CATalog?		全ファイルのファイル情報参照
MMEMory:FREE:ALL?		マス・メモリ使用状況の問い合わせ
MMEMory:FREE?		マス・メモリ使用状況の問い合わせ
MMEMory?		全ファイルのファイル情報とマス・メモリの使用状況の問い合わせ

MODEコマンド

MODE グループのコマンドは、波形またはシーケンスを出力するためのモード選択、トリガ・イベントの生成、出力の停止などを行います。また外部トリガ・ソースに対するトリガ条件の設定も本コマンドで行います。

表 2-9：MODE コマンド・グループ

ヘッダ		内　容
CONFigure (AWG2005型)	(?)	システム・コンフィグレーションの選択
MODE	(?)	モードの選択
RUNNing?		出力状態の問い合わせ
STARt		トリガ・イベントの生成
*TRG		トリガ・イベントの生成
STOP		出力動作の停止と初期化
TRIGger:HOLDoff (AWG2010/11//20/21型)	(?)	トリガ・ホールドオフ時間の選択
TRIGger:IMPedance (AWG2010/11/20/21/40/41型)	(?)	外部トリガ信号に対する入力インピーダンス設定
TRIGger:LEVel	(?)	トリガ・イベント生成のための外部トリガ信号入力レベルの設定
TRIGger:POLarity	(?)	トリガ・イベント生成のための外部トリガ信号極性の設定
TRIGger:SLOPe	(?)	トリガ・イベント生成のための外部トリガ入力スロープの設定
TRIGger?		トリガに関する設定中の全設定値の問い合わせ

OUTPUTコマンド

OUTPUT グループのコマンドは、波形またはシーケンスの出力をコントロールしたり、Sync信号の生成位置を選択したりします。出力チャンネルの指定は、ヘッダ・ニーモニック CH<x>で行います。<x> は、チャンネル番号に応じた数字を取ります。

表 2-10：OUTPUT コマンド・グループ

ヘッダ		内　容
OUTPut:CH<x>:STATe	(?)	出力の ON/OFF
OUTPut:CH<x>?		出力状態の問い合わせ
OUTPut:CH1:NORMal:STATe (AWG2040/41型)	(?)	出力の ON/OFF
OUTPut:CH1:INVerted:STATe (AWG2040/41型)	(?)	出力の ON/OFF
OUTPut:CH1:INVerted? (AWG2040/41型)		出力の ON/OFF
OUTPut:CH1:NORMal? (AWG2040/41型)		出力の ON/OFF

表 2-10：OUTPUT コマンド・グループ(続)

ヘッダ		内　容
OUTPut:SYNC (AWG2010/11/20/21型)	(?)	Sync信号発生位置の選択
OUTPut?		OUTPUT に関連する設定中の全設定値の問い合わせ

SETUPコマンド

SETUP グループのコマンドは、クロック源の選択、クロック・パラメタの設定、出力波形パラメタの設定などを行います。出力波形パラメタは、ヘッダ・ニーモニック CH<x> で指定されるチャンネルに対して設定することができます。<x> は、チャンネル番号に応じた数字を表します。

表 2-11：SETUP コマンド・グループ

ヘッダ		内　容
CLOCk:FREQuency	(?)	クロック源の周波数設定
CLOCk:SOURce(?)		クロック源の選択
CLOCK:SWEep:DEFine (AWG2005型)	(?)	クロック・スイープのデータ転送とファイルへの書き込み
CLOCK:SWEep:DWELl(?) (AWG2005型)		クロック・スイープのドエル値の設定
CLOCK:SWEep:FREQuency:STARt (?) (AWG2005型)		クロック・スイープの開始周波数の設定
CLOCK:SWEep:FREQuency:STOP (?) (AWG2005型)		クロック・スイープの終了周波数の設定
CLOCK:SWEep:FREQuency? (AWG2005型)		クロック・スイープの開始・終了周波数の問い合わせ
CLOCK:SWEep:MODE (AWG2005型)	(?)	クロック・スイープ時のモード設定
CLOCK:SWEep:STATe (AWG2005型)	(?)	クロック・スイープの ON/OFF
CLOCK:SWEep:TIME (AWG2005型)	(?)	クロック・スイープの時間設定
CLOCK:SWEep:TYPE (AWG2005型)	(?)	クロック・スイープのタイプの選択
CLOCK:SWEep (AWG2005型)	(?)	クロック・スイープの全設定の問い合わせ
CLOCk:CH2:DIVider (AWG2010/11/20/21型)	(?)	分周比の設定
CLOCk:CH2? (AWG2010/11/20/21型)		チャンネル2のクロックに関連する設定中の全設定値の問い合わせ
CLOCk?		クロックに関連する設定中の全設定値の問い合わせ
CH1:OPERation (AWG2005/10/11/20/21型)	(?)	波形演算モードの選択
CH<x>:AMPLitude	(?)	波形の最大電圧レンジ設定

表 2-11：SETUP コマンド・グループ(続)

ヘッダ		内　容
CH<x>:FILTer	(?)	カット・オフ・フィルタの選択
[CH1]MARKERLEVEL1:LOW (AWG2040/41型)	(?)	マーカ1のローレベルの設定、問い合わせ
[CH1]MARKERLEVEL1:HIGH (AWG2040/41型)	(?)	マーカ1のハイレベルの設定、問い合わせ
[CH1]MARKERLEVEL2:LOW (AWG2040/41型)	(?)	マーカ2のローレベルの設定、問い合わせ
[CH1]MARKERLEVEL2:HIGH (AWG2040/41型)	(?)	マーカ2のハイレベルの設定、問い合わせ
[CH1]MARKERLEVEL1? (AWG2040/41型)		マーカ1のレベル設定の問い合わせ
[CH1]MARKERLEVEL2? (AWG2040/41型)		マーカ2のレベル設定の問い合わせ
CH<x>:OFFSet	(?)	オフセット電圧の設定
CH<x>:TRACk:AMPLitude (AWG2005/10/11/20/21型)	(?)	電圧レンジのトラッキング設定
CH<x>:TRACk:OFFSet (AWG2005/10/11/20/21型)	(?)	オフセット電圧のトラッキング設定
CH<x>:TRACk? (AWG2005/10/11/20/21型)		トラッキングの全設定値の問い合わせ
CH<x>:WAVeform	(?)	出力を行う波形ファイルまたはシーケンス・ファイルの指定
CH<x>?		指定チャンネルの SETUP に関連する設定中の全設定値の問い合わせ

STATUS＆EVENTコマンド

STATUS & EVENT グループのコマンドは、本機器の状態を調べたり、イベントの発生を制御したりするために、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レジスタとキューの設定、およびその問い合わせを行います。ステータス／イベント・レポーティング・システムについての詳細は、第 3 章を参照ください。

表 2-12：STATUS & EVENT コマンド・グループ

ヘッダ		内　容
ALLEv?		イベント・キューから全イベントの取り出し
*CLS		SESR、SBR、イベント・キューのクリア
DESE	(?)	DESER の設定
*ESE	(?)	ESER の設定
*ESR?		SESR の内容の問い合わせ
EVENT?		イベント・キューからイベントの取り出し
EVMsg?		イベント・キューからイベントの取り出し
EVQty?		イベント・キューにあるイベント数の問い合わせ
*PSC	(?)	PSC (power−on status clear) フラグの設定
*SRE	(?)	SRER の設定
*STB?		SBR の内容の問い合わせ

SYNCHRONIZATIONコマンド

SYNCHRONIZATION グループのコマンドは、ペンディング中の全ての操作が終了するのを監視します。全ての操作が終了するのを待ってからコマンドを実行する必要のある場合には、本コマンドを使用して、コマンド実行の同期制御を行います。使用方法についての詳細は、第 3 章の「実行の同期について」(3-9 ページ) を参照ください。

表 2-13：SYNCHRONIZATION コマンド・グループ

ヘッダ		内　容
*OPC	(?)	オペレーション・コンプリート
*WAI		コマンド実行待ち

SYSTEMコマンド

SYSTEM グループのコマンドは、日時の設定、フロント・パネル・コントロールのロック、レスポンス中のヘッダの取り扱い制御、ID 情報や設定情報などの問い合わせを行います。本グループは、他のグループに分類できないコマンドの集まりです。

表 2-14：SYSTEMコマンド・グループ

ヘッダ		内　容
DATE	(?)	日付の設定
DEBug:SNOop:STATe	(?)	デバッグ機能の ON/OFF
DEBug:SNOop:DELAy:TIME	(?)	デバッグ用遅延時間の設定
DEBug:SNOop:DELAy?		デバッグ用遅延時間の問い合わせ
DEBug:SNOop?		デバッグ機能の全設定の問い合わせ
DEBug?		デバッグ機能の全設定の問い合わせ
FACTory		工場出荷時設定へのリセット
HEADer	(?)	レスポンス・メッセージ中のヘッダの制御
HWSequencer? (AWG2041型)		ハードウェア・シーケンサの搭載の有無と使用中かどうかの問い合わせ
HWSequencer:INSTalled? (AWG2041型)		ハードウェア・シーケンサの搭載の有無の問い合わせ
HWSequencer:MODE (AWG2041型)	(?)	ハードウェア・シーケンサ機能の ON/OFF
ID?		機器 ID 情報の問い合わせ
*IDN?		機器 ID 情報の問い合わせ
LOCk	(?)	フロント・パネル・コントロールのロック
*LRN?		本機器全設定の問い合わせ
*OPT?		組み込みオプション情報の問い合わせ
*RST		本機器のリセット
SECUre		メモリのクリアと工場出荷時設定へのリセット
TIME	(?)	時刻の設定
UNLock		フロント・パネル・コントロールのロック状態解除
UPTime?		電源投入後経過時間の問い合わせ
VERBose	(?)	レスポンス・メッセージ中のヘッダ制御

WAVEFORMコマンド

WAVEFORM グループのコマンドは、本機器と外部コントローラとの間で、波形データ、マーカ・データ、イクエーション、オート・ステップ・データ、シーケンスの転送を行います。

波形データの転送は、CURVE コマンドで行います。転送される波形データは、アンスケールのデータですので、プリアンブル情報を使用して絶対スケールの波形データに変換します。プリアンブルは、波形データのフォーマットやスケーリングなどの情報、あるいは波形 ID や単位などの補助情報で、WFMPre をルート・ヘッダ・ニーモニックに持つコマンドによって設定・参照が可能です。

マーカ・データの転送は、MARKer:DATA コマンドで行います。また MARKER<x>:AOFF、MARKER<x>:POINt コマンドを使用すると、全マーカのリセットや指定位置マーカのセット、リセットが行えます。

波形データおよびマーカ・データの転送の際には、あらかじめ、DATA:DESTination、DATA:SOURce コマンドで転送先、転送元を指定します。転送手順やデータのフォーマットの詳細については、本章の「波形転送について」を参照ください。

イクエーションの転送は、EQUAtion:DEFine コマンドで行います。また EQUAtion: WPOints コマンドでポイント数の定義を、EQUAtion:COMPile コマンドでイクエーションから波形ファイルへのコンパイルを行うことができます。

オート・ステップ・データおよびシーケンスの転送は、それぞれ AUTOStep:DEFine コマンドおよび SEQUence:DEFine コマンドで行えます。SEQUence:DEFine コマンドを使用すると、シーケンスから波形データへの展開が可能です。

なおプリアンブル情報のうち、WFMPre:XINCR、WFMPre:YMULT、WFMPre:YZERO のコマンド／問い合わせコマンドで設定・参照される項目では、設定値と参照値が異なる場合があります。これは、コマンドが外部から波形を取り込むためのプリアンブルを設定するのに対して、問い合わせコマンドは外部に波形を転送する際のプリアンブルを参照するためです。従って設定値と参照値が一致するのは、DATA:DESTination コマンドで設定する転送先と DATA:SOURceで 設定する転送元が一致する場合のみです。

表 2-15：WAVEFORMコマンド・グループ

ヘッダ		内　容
AUTOStep:DEFine	(?)	オート・ステップ・データの転送とファイルへの書き込み
CURVe	(?)	本機器と外部コントローラ間の波形データ転送
DATA:DESTination	(?)	データ転送先の指定
DATA:ENCDG	(?)	波形データ転送フォーマットの指定
DATA:SOURce	(?)	データ転送元の指定
DATA:WIDTh	(?)	波形1ポイントあたりのバイト数設定
DATA	(?)	DATA に関する設定の初期化と DATA に関する設定中の全設定値の問い合わせ
EQUAtion:COMPile:STATe	(?)	イクエーション・ファイルのコンパイル

表 2-15：WAVEFORMコマンド・グループ(続)

ヘッダ		内　容
EQUAtion:COMPile	(?)	イクエーション・ファイルのコンパイル
EQUAtion:DEFine	(?)	イクエーションの転送とファイルへの書き込み
EQUAtion:WPOints	(?)	波形ポイント数のファイルへの書き込み
MARKER<x>:AOFF		全マーカ・データのリセット
MARKER<x>:POINt	(?)	指定位置マーカ・データの設定
MARKer:AOFF		全マーカ・データのリセット
MARKer:DATA	(?)	本機器と外部コントローラ間のマーカ・データ転送
MARKer:POINt	(?)	指定位置マーカ・データの設定
SEQUence:DEFine	(?)	シーケンスの転送とファイルへの書き込み
SEQUence:EXPAnd		シーケンスから波形データへの展開
WAVFrm?		本機器と外部コントローラ間の波形プリアンブルと波形データの転送
WFMPre:ENCDG	(?)	波形データのエンコード形式の指定
WFMPre:BN_FMT	(?)	バイナリ・データのフォーマット指定
WFMPre:BYT_NR	(?)	データ・ポイントのデータ長指定
WFMPre:BIT_NR	(?)	有効ビット数の指定
WFMPre:BYT_OR	(?)	バイト・オーダの指定
WFMPre:CRVCHK	(?)	エラー・チェック方法の指定
WFMPre:WFID	(?)	波形 ID 情報の設定
WFMPre:NR_PT	(?)	波形データ・ポイント数の設定
WFMPre:PT_FMT	(?)	データ・フォーマットの設定
WFMPre:XUNIT	(?)	X軸・単位表現の設定
WFMPre:XINCR	(?)	X軸インクリメント値の設定
WFMPre:PT_OFF	(?)	X軸データ・ポイント・オフセット値の設定
WFMPre:XZERO	(?)	X軸・原点オフセット値の設定
WFMPre:YUNIT	(?)	Y軸・単位表現の設定
WFMPre:YMULT	(?)	Y軸データ・ポイント乗数の設定
WFMPre:YZERO	(?)	Y軸・原点オフセット値の設定
WFMPre:YOFF	(?)	Y軸データ・ポイント・オフセット値の設定
WFMPre?		波形プリアンブルに関係する設定中の全設定値の問い合わせ

コマンド詳細説明

ABSTouch

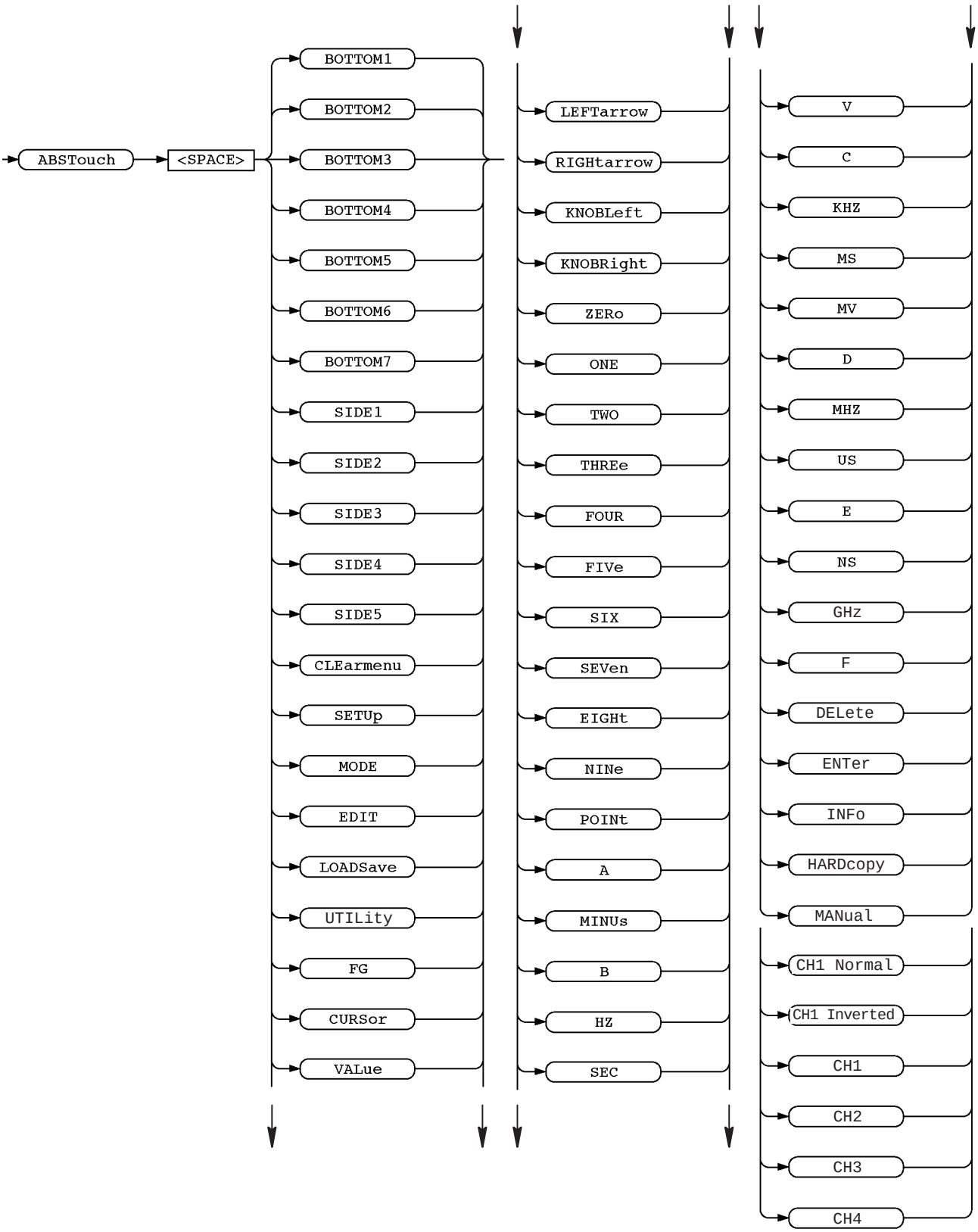
ABSTouchコマンドは、フロント・パネルのキーおよびノブに対応する機能を実行します。

グループ： DISPLAY

関連コマンド：

シンタックス： ABSTouch {BOTTOM1 | BOTTOM2 | BOTTOM3 | BOTTOM4 | BOTTOM5 | BOTTOM6 | BOTTOM7 | SIDE1 | SIDE2 | SIDE3 | SIDE4 | SIDE5 | CLEarmenu | SETUp | MODE | EDIT | LOADSave | UTILity | FG | CURSor | VALue | LEFTarrow | RIGHtarrow | KNOBLeft | KNOBRight | ZERo | ONE | TWO | THREe | FOUR | FIVe | SIX | SEVen | EIGHt | NINe | POINt | A | MINUs | B | HZ | SEC | V | C | KHZ | MS | MV | D | MHZ | US | E | NS |
GHz (AWG2040/41型) | F | DELete | ENTer |
HARDcopy (AWG2005/11/21/40/41型) | MANual |
CH1 (AWG2005/10/11/20/21型)|
CH1 Normal (AWG2040/41型) |
CH1 Inverted (AWG2040/41型) |
CH2 (AWG2005/10/11/20/21型) |
CH3 (AWG2005型) |
CH4 (AWG2005型) }

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 2-4 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。


ABSTouch
<SPACE>
BOTTOM1
BOTTOM2
BOTTOM3
BOTTOM4
BOTTOM5
BOTTOM6
BOTTOM7
SIDE1
SIDE2
SIDE3
SIDE4
SIDE5
CLEarmenu
SETUp
MODE
EDIT
LOADSave
UTILity
FG
CURSor
VALue
LEFTarrow
RIGHtarrow
KNOBLeft
KNOBRight
ZERo
ONE
TWO
THREe
FOUR
FIVe
SIX
SEVen
EIGHt
NINe
POINt
A
MINUs
B
HZ
SEC
V
C
KHZ
MS
MV
D
MHZ
US
E
NS
GHz
F
DELete
ENTer
INFo
HARDcopy
MANual
CH1 Normal
CH1 Inverted
CH1
CH2
CH3
CH4

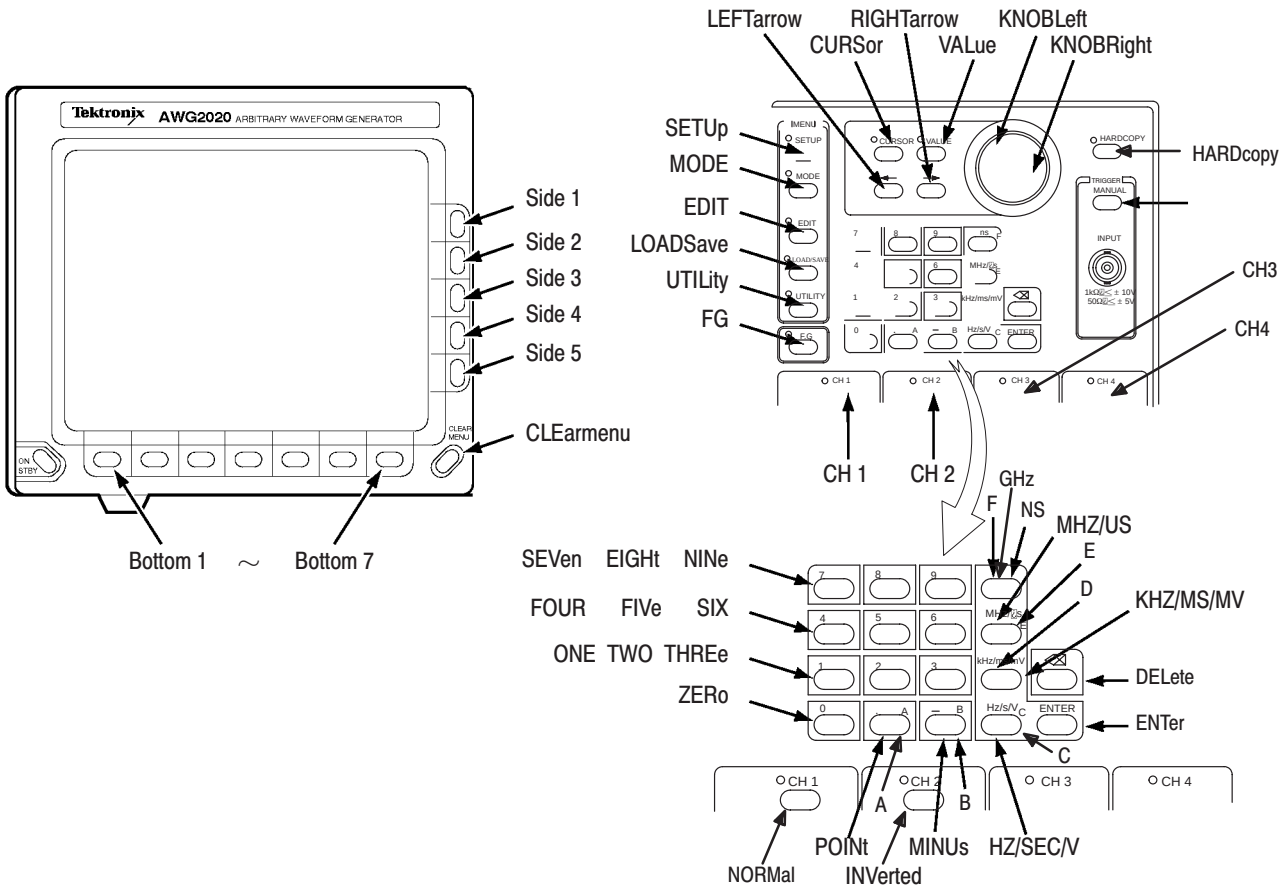


図 2-4：アーギュメントとフロント・パネル

使用例： 以下は、本機器の管面に **SETUP** メニューを表示する例です。**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
```

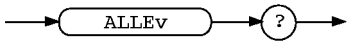
ALLEv?

ALLEv? 問い合わせコマンドは、イベント・キューから全てのイベント・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は、*ESR? 問い合わせコマンドを実行することにより可能になります。

グループ： STATUS & EVENT

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：

レスポンス： レスポンス・メッセージのフォーマットは、次の通りです。

[: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： 以下は :ALLEV? のレスポンス例です。

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

AUTOStep:DEFine (?)

AUTOStep:DEFine コマンドは、オートステップ・データを本機器の指定ファイルの指定チャンネル側に転送します。また AUTOStep:DEFine? 問い合わせコマンドは、本機器の指定ファイルの指定チャンネル側のオートステップ・データを問い合わせます。

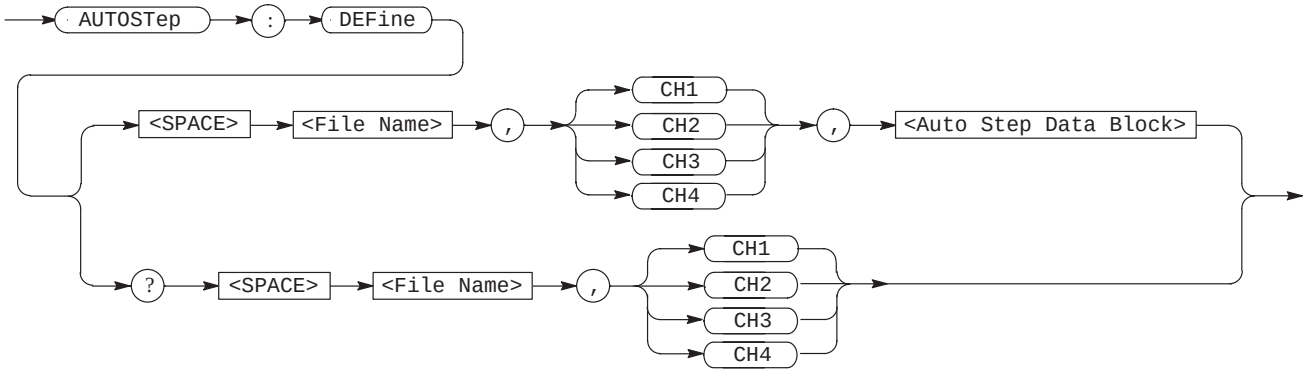
グループ： WAVEFORM

関連コマンド：

シンタックス： AWG2005/AWG2010/AWG2011/AWG2020/AWG2021型：

AUTOStep:DEFine <File Name>, {CH1 | CH2 | CH3 | CH4 }, <Autostep Data Block>

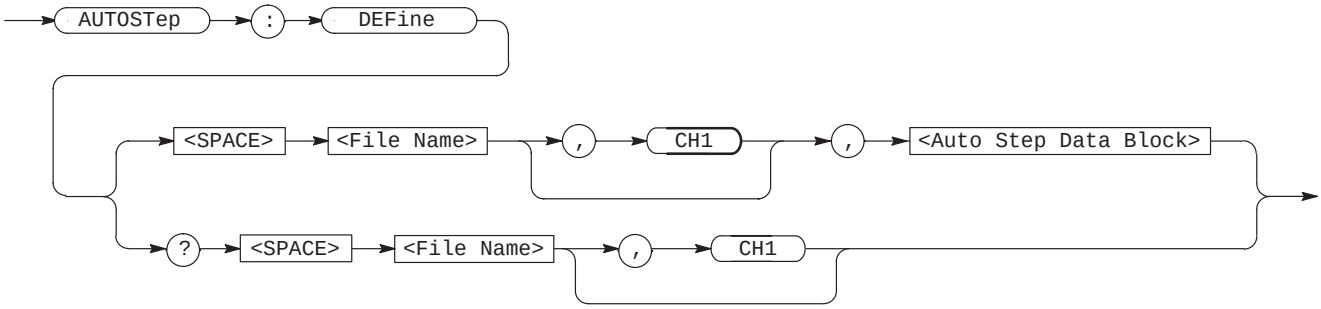
AUTOStep:DEFine? <File Name>, {CH1 | CH2 | CH3 | CH4}



AWG2040/AWG2041型：

AUTOStep:DEFine <File Name> [,CH1], <Autostep Data Block>

AUTOStep:DEFine? <File Name> [,CH1]



アーギュメント： <File Name>::=<文字列> オートステップ・データが転送されるファイル

CH1 — チャンネル 1 のオートステップ・データ

CH2 — チャンネル 2 のオートステップ・データ

CH3 — チャンネル 3 のオートステップ・データ

CH4 — チャンネル 4 のオートステップ・データ

<Auto Step Data Block>::=<Arbitrary Block> オートステップ・データ

オートステップ・データは ASCII コードで以下の例のように指定できます。各オートステップ・データは波形またはシーケンス・ファイル名 <waveform>、クロック・ソース <clocksource>、内部クロック周波数 <clock>、オペレーション・モード <operation> (AWG2005/10/11/20/21型)、周波数カットオフ・フィルタ <filter>、出力電圧レンジ <amplitude>、オフセット電圧 <offset>、マーカ1ハイ・レベル <mark1H> (AWG2040/41型)、マーカ 1 ロー・レベル <mark1L>(AWG2040/41型)、マーカ 2 ハイ・レベル <mark2H> (AWG2040/41型)、マーカ 2 ロー・レベル <mark2L> (AWG2040/41型) をカンマで続け、ライン・フィールド (LF) のコードで区切られます。ただし、AWG2010/11/20/21 型は波形またはシーケンス・ファイル名 <waveform> のみを <LF> で区切ったフォーマットも受け付ける事ができます。

<waveform>::=	文字列
<clock>::=	<NR3>[<unit1>]
<operation>::=	{INTernal | EXTernal}
<filter>::=	{THROUGH | THR | THRU | 500KHZ | K500 | 1MHZ | M1 | 2MHZ | M2 | 5MHZ |M5 | 10MHZ | M10 | 20MHZ | M20 | 50MHZ | M50 | 100 MHZ | M100}
<amplitude>::=	<NR2>[<unit2>]
<offset>::=	<NR2>[<unit2>]
<mark1H>::=	<NR2>[<unit2>]
<mark1L>:: =	<NR2>[<unit2>]
<mark2H>::=	<NR2>[<unit2>]
<mark2L>::=	<NR2>[<unit2>]

<unit1>::=[{Hz | KHz | MHz | GHz}]
<unit2>::=[{V | mV}]

AWG2005/AWG2010/AWG2011/AWG2020/AWG2021型：

#3109WAVE01.WFM, INTERNAL, 10.00000E+06,NORMAL,THROUGH, 1.000, 0.000 <LF> WAVE02.WFM, 2.00000E+06, NORMAL, THROUGH, 2.000, 0.000

AWG2040/AWG2041型：

#3135WAVE01.WFM, INTERNAL, 1.000000E+09, 10MHZ, 1.000,0.000, 2.0, 0.0, 2.0, 0.0 <LF>
WAVE02.WFM, INTERNAL, 1.000000E+09, THROUGH, 1.000, 0.000, 2.0, 0.0, 2.0, 0.0

使用例： 以下は、AWG2020型でオートステップ・データをファイル AUTOS01.AST のチャンネル 1 側に転送する例です。

AUTOSTEP:DEFINE ”AUTOS01.AST”, CH1, #287WAVE01.WFM, 10MHZ, NORMAL, THRU, 1.000, 0.000 <LF> WAVE02.WFM, 2.00000E+06, NORMAL, THRU, 2.000, 0.000

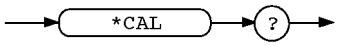
*CAL?

*CAL? 共通・問い合わせコマンドは、セルフ・キャリブレーションを実行し、キャリブレーションが正常に終了したかどうかの結果を戻します。キャリブレーション実行中にエラーが検出された場合には、実行が直ちに中断されます。

グループ： CALIBRATION & DIAGNOSTIC

関連コマンド： SELFcal:RESUlt、SELFcal:SELect、SELFcal:STATe

シンタックス： *CAL?



アーギュメント：

レスポンス： レスポンス・フォーマットは以下のようになります。

<Result>

ここで

<Result>::=<NR1>　以下のいずれかです。
0　— 正常に終了しました。
200　— CLOCK関係のエラーが発生しました (AWG2010/11/20/21型)。
600　— アナログ関係のエラーが発生しました。
800　— TRIGGER関係のエラーが発生しました (AWG2005型)。

注： キャリブレーションに要する時間は、最大 1 分程度です。この間に他のコマンドを実行しようとしても本機器は反応しません。

使用例： 以下は、キャリブレーションを実行する例です。

*CAL?

CH1:OPERation　(?)

(AWG2005/10/11/20/21型のみ)

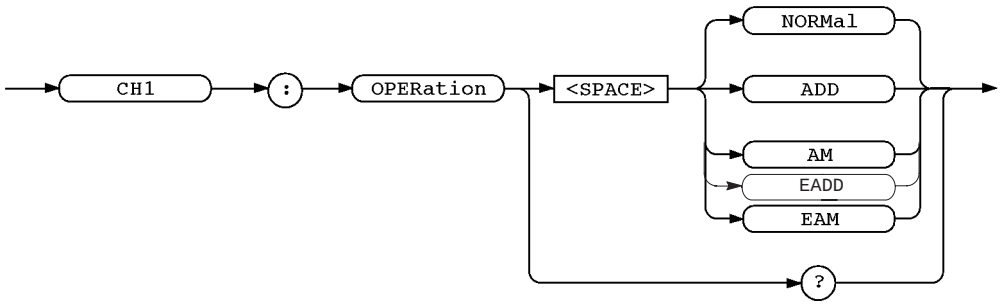
CH1:OPERation コマンドは、チャンネル1に対する波形演算モードを選択します。この場合には、CH1 のみが有効なヘッダ・ニーモニックです。

CH1:OPERation? 問い合わせコマンドは、選択中の波形演算モードを問い合わせます。

グループ： SETUP

関連コマンド： CH<x>:AMPLitude、CH<x>:FILTer、CH<x>:OFFSet、CH<x>:TRACk:AMPLitude、CH<x>:TRACk:OFFSet、CH<x>:WAVeform

シンタックス： CH1:OPERation　{NORMal | ADD | AM | EADD (AWG2005型) | EAM}
CH1:OPERation?



アーギュメント： 以下の表に従います。

アーギュメント	説　明
NORMal	演算を行いません。
ADD	チャンネル 2 の出力をチャンネル 1 の出力に加えます。この際チャンネル 2 側のコネクタ出力を停止します。ある時刻tにチャンネル1のコネクタに現れる出力電圧 Vout (t) は、以下の通りです。 Vout (t) = Vch1 (t) + Vch2 (t) + Voffset:ch1
EADD	EXTADD に加えた電圧がそのままチャンネル 1 に加えられ、出力されます。 EAM チャンネル1の出力に外部入力コネクタから入力される信号を乗算します。ある時刻 t にチャンネル 1 のコネクタに現れる出力電圧 Vout (t) は、以下の通りです。 Vout (t) = Vch1 (t) * (Vext (t) + 1) / 2 + Voffset:ch1
EAM	チャンネル 1 の出力にチャンネル2の出力を乗算します。この際チャンネル 2 側のコネクタ出力を停止します。ある時刻tにチャンネル 1 のコネクタに現れる出力電圧 Vout (t) は、以下の通りです。 Vout (t) = Vch1 (t) * Vch2 (t) + Voffset:ch1

*　Vch1 (t)、Vch2 (t)、Vext (t) は、それぞれ演算前の、ある時刻tにおけるチャンネル1、チャンネル2、外部入力信号の電圧を表します。また Voffset:ch1、Voffset:ch2 は、チャンネル 1 およびチャンネル 2 に対して設定されたオフセット電圧を表します。

注： 波形演算を行うと、チャンネル 1 から出力される電圧が最大出力電圧 (AWG2005型で 10 Vp-p、AWG2010/11型で 7.5 Vp-p、AWG2020/21型で 5.0Vp-p) を超えることがあります。この際、出力電圧の波形が歪むことがありますので御注意ください。

使用例： 以下は、チャンネル 2 の出力をチャンネル 1 の出力に加える波形演算モードを選択する例です。

```
:CH1:OPERATION  ADD
```

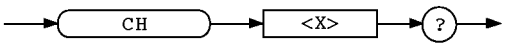
CH<x>?

CH<x>? 問い合わせコマンドは、SETUP グループのコマンドで可能な設定に対して、指定チャンネル上に設定中の値を問い合わせます。

グループ： SETUP

関連コマンド： CH<x>:AMPLitude、CH<x>:FILTer、CH<x>:OFFSet、CH1:OPERation、CH<x>:TRACk:AMPLitude、CH<x>:TRACk:OFFSet、CH<x>:WAVeform

シンタックス： CH<x>?



アーギュメント：

レスポンス： コマンドが連結された形式で戻されます。詳細は、使用例を参照ください。

使用例： 以下は :CH1? のレスポンス例です。

AWG2005/AWG2010/AWG2011/AWG2020/AWG2021型：

```
:CH1:WAVEFORM ”WAVE01.WFM”; OPERATION NORMAL;FILTER THRU;
AMPLITUDE1.000; OFFSET0.000
```

AWG2040/AWG2041型：

```
:CH1:WAVEFORM ”WAVE01.WFM”; FILTER THRU; AMPLITUDE 1.000;
OFFSET 0.000; MARKERLEVEL1:HIGH 2.0; LOW 0.0;:CH1:MARKERLEVEL2:
HIGH 2.0; LOW 0.0
```

CH<x>:AMPLitude　(?)

CH<x>:AMPLitude コマンドは、波形の最大電圧レンジ (Vp-p) を設定します。また CH<x>:AMPLitude? 問い合わせコマンドは、設定中の最大電圧レンジ Vp-p を問い合わせます。

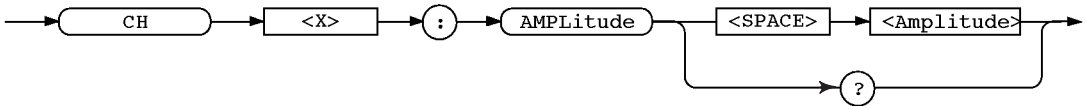
グループ： SETUP

関連コマンド： CH<x>:FILTer、CH<x>:OFFSet、CH1:OPERation、CH<x>:TRACk:AMPLitude、CH<x>:TRACk:OFFSet、CH<x>:WAVeform

シンタックス： AWG2005/AWG2010/AWG2011/AWG2020/AWG2021型：

CH<x>:AMPLitude <Amplitude>

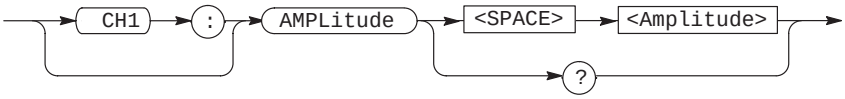
CH<x>:AMPLitude?



AWG2040/AWG2041型：

[CH1:] AMPLitude <Amplitude>

[CH1:] AMPLitude?



アーギュメント： <Amplitude>::=<NR2>[<unit>]

<unit>::={V | mV}

設定範囲：0.050 V ～ 10.000 V、ステップ 0.001 V　(AWG2005型)

設定範囲：0.050 V ～ 7.500 V、ステップ 0.001 V　(AWG2010/11型)

設定範囲：0.050 V ～ 5.000 V、ステップ 0.001 V　(AWG2020/21型)

設定範囲：0.020 V ～ 2.000 V、ステップ 0.001 V　(AWG2040/41型)

使用例： 以下は、最大電圧レンジを 230 mVp-p に設定する例です。

```
:CH1:AMPLITUDE  230.0mV
```

CH<x>:FILTer (?)

CH<x>:FILTer コマンドは、周波数カット・オフ・フィルタを選択します。また CH<x>:FILTer? 問い合わせコマンドは、選択中のカット・オフ・フィルタを問い合わせます。

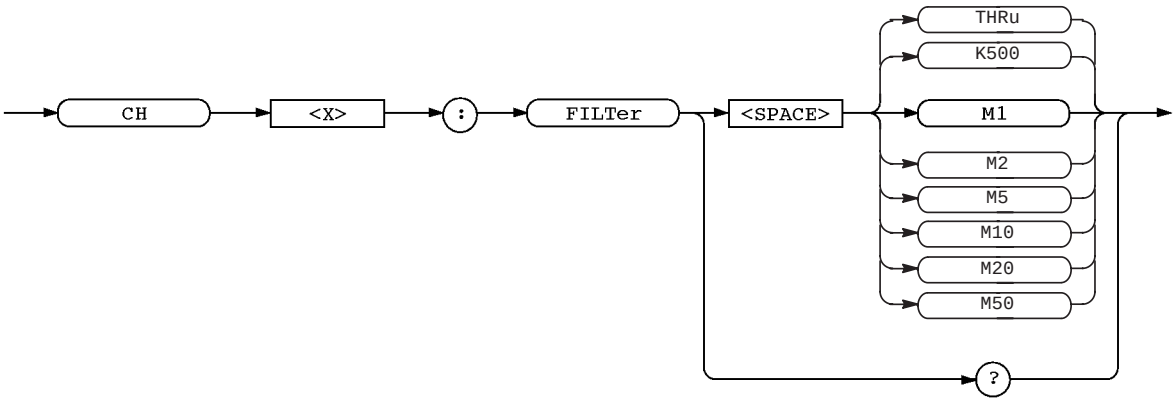
グループ： SETUP

関連コマンド： CH<x>:AMPLitude、CH<x>:OFFSet、CH1:OPERation、CH<x>:TRACk:AMPLitude、CH<x>:TRACk:OFFSet、CH<x>:WAVeform

シンタックス： AWG2005/AWG2010/AWG2011/AWG2020/AWG2021型：

CH<x>:FILTer {THRu | K50 | M1 | M2 | M5 | M20 | M50 }

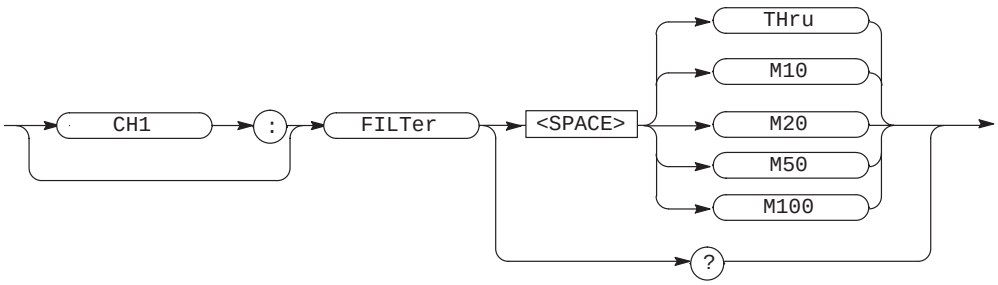
CH<x>:FILTer?



AWG2040/AWG2041型：

[CH1:] FILTer {M10 | M20 | M50 | M100}

[CH1:] FILTer?



アーギュメント：

THRu	—	フィルタを使用しません。
K500	—	500 kHz (AWG2005型)
M1	—	1 MHz (AWG2005/10/11/20/21型)
M2	—	2 MHz (AWG2005型)
M5	—	5 MHz (AWG2005/10/11/20/21型)
M10	—	10 MHz (AWG2040/41型)
M20	—	20 MHz (AWG2010/11/20/21/40/41型)
M50	—	50 MHz (AWG2020/21/40/41型)
M100	—	100 MHz((AWG2040/41型)

使用例： 以下は、20 MHz のカット・オフ周波数を持つフィルタを選択する例です。

```
:CH1:FILTER  M20
```

CH<x>:MARKERLEVEL1?

(AWG2040/41型のみ)

CH<x>:MARKERLEVEL1? 問い合わせコマンドは、マーカ 1 のマーカ・レベルの設定を問い合わせます。

グループ： SETUP

関連コマンド： CH<x>:MARKERLEVEL1:HIGH、CH<x>:MARKERLEVEL1:LOW

シンタックス： [CH1:]MARKERLEVEL1?

アーギュメント：

レスポンス： 詳しくは、使用例を参照ください。

使用例： 以下は、CH1:MARKERLEVEL1? のレスポンス例です。

```
:CH1:MARKERLEVEL1:HIGH  2.0; LOW  0.0
```

CH<x>:MARKERLEVEL1:HIGH (?)

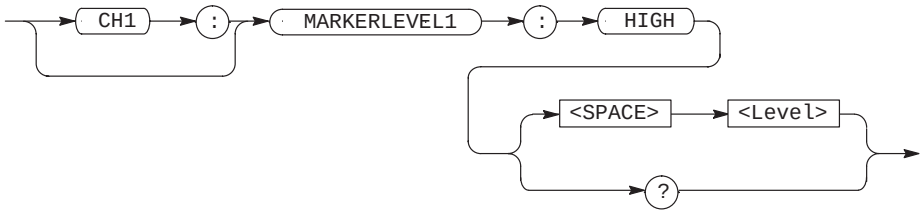
(AWG2040/41型のみ)

CH<x>:MARKERLEVEL1:HIGH コマンドは、 マーカ 1 のハイ・レベルを設定します。また、CH<x>:MARKERLEVEL1:HIGH? 問い合わせコマンドは、設定されているマーカ1のハイ・レベルを問い合わせます。

グループ： SETUP

関連コマンド： CH<x>:MARKERLEVEL1:LOW

シンタックス： [CH1:]MARKERLEVEL1:HIGH <Level>
[CH1:]MARKERLEVEL1:HIGH？



アーギュメント： <Level> ::= <NR2>[<unit>]
<unit> ::= {V | mV}
設定範囲： −1.9 V ～ 2.0 V、 ステップ 0.1V (ハイ・レベル > ロー・レベル)

使用例： 以下は、マーカ 1 のハイ・レベルを 1 V に設定します。

```
:CH1:MARKERLEVEL1:HIGH 1.0
```

CH<x>:MARKERLEVEL1:LOW (?)

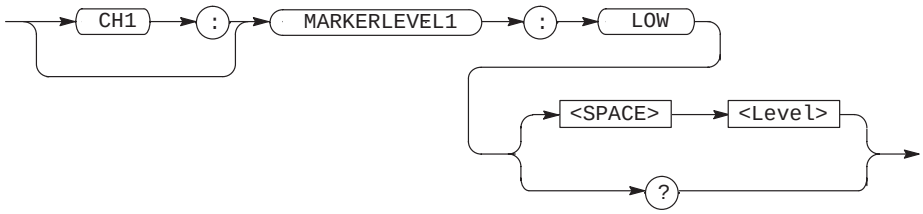
(AWG2040/41型のみ)

CH<x>:MARKERLEVEL1:LOW コマンドは、 マーカ 1 のロー・レベルを設定します。また、CH<x>:MARKERLEVEL1:LOW? 問い合わせコマンドは、設定されているマーカ 1 のロー・レベルを問い合わせます。

グループ： SETUP

関連コマンド： CH<x>:MARKERLEVEL1:HIGH

シンタックス： [CH1:]MARKERLEVEL1:LOW <Level>
[CH1:]MARKERLEVEL1:LOW?



アーギュメント： <Level> ::= <NR2>[<unit>]
<unit> ::= {V | mV}
設定範囲： −1.9 V ～ 2.0 V、ステップ 0.1 V (ハイ・レベル > ロー・レベル)

使用例： 以下は、マーカ 1 のロー・レベルを 0.5 V に設定します。

:CH1:MARKERLEVEL1:LOW 0.5

CH<x>:MARKERLEVEL2?

(AWG2040/41型のみ)

CH<x>:MARKERLEVEL2? 問い合わせコマンドは、マーカ2のマーカ・レベルの設定を問い合わせます。

グループ： SETUP

関連コマンド： CH<x>:MARKERLEVEL2:HIGH、CH<x>:MARKERLEVEL2:LOW

シンタックス： [CH1:]MARKERLEVEL2?

アーギュメント：

レスポンス： 詳しくは、使用例を参照ください。

使用例： 以下は、CH1:MARKERLEVEL2? のレスポンス例です。

:CH1:MARKERLEVEL2:HIGH 2.0; LOW 0.0

CH<x>:MARKERLEVEL2:HIGH (?)

(AWG2040/41型のみ)

CH<x>:MARKERLEVEL2:HIGH コマンドは、マーカ 2 のハイ・レベルを設定します。また、CH<x>:MARKERLEVEL2:HIGH? 問い合わせコマンドは、設定されているマーカ2のハイ・レベルを問い合わせます。

グループ： SETUP

関連コマンド： CH<x>:MARKERLEVEL2:LOW

シンタックス： [CH1:]MARKERLEVEL2:HIGH <Level>
[CH1:]MARKERLEVEL2:HIGH?

アーギュメント： <Level> ::= <NR2>[<unit>]
<unit> ::= {V | mV}
設定範囲： −1.9 V ～ 2.0 V 、ステップ 0.1 V (ハイ・レベル > ロー・レベル)

使用例： 以下は、マーカ 2 のハイ・レベルを 1 V に設定します。

:CH1:MARKERLEVEL2:HIGH 1.0

CH<x>:MARKERLEVEL2:LOW (?)

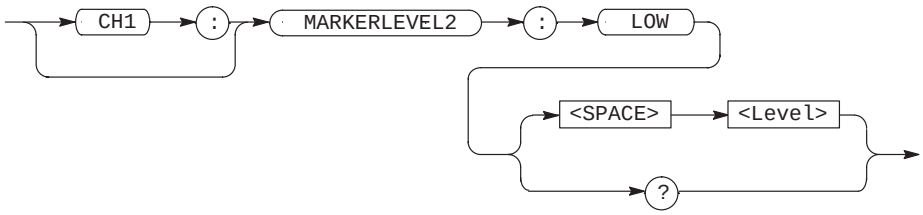
(AWG2040/41型のみ)

CH<x>:MARKERLEVEL2:LOW コマンドは、 マーカ2のロー・レベルを設定します。また、CH<x>:MARKERLEVEL2:LOW? 問い合わせコマンドは、設定されているマーカ 2 のロー・レベルを問い合わせます。

グループ： SETUP

関連コマンド： CH<x>:MARKERLEVEL2:HIGH

シンタックス： [CH1:]MARKERLEVEL2:LOW <Level>
[CH1:]MARKERLEVEL2:LOW?



アーギュメント： <Level> ::= <NR2>[<unit>]
<unit> ::= {V | mV}
設定範囲： −1.9V ～ 2.0V 、ステップ 0.1V
(設定条件：ハイ・レベル > ロー・レベル)

使用例：　以下は、マーカ 2 のロー・レベルを 1.5 V に設定します。

```
:CH1:MARKERLEVEL2:LOW 1.5
```

CH<x>:OFFSet (?)

CH<x>:OFFSet コマンドは、指定チャンネルに供給されるオフセット電圧を設定します。また CH<x>:OFFSet? 問い合わせコマンドは、設定中のオフセット電圧を問い合わせます。

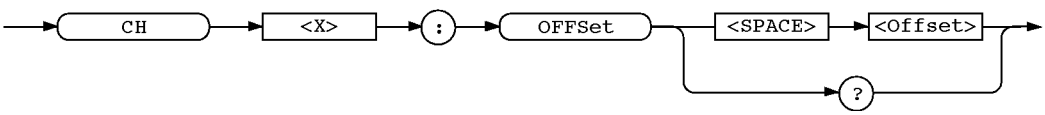
グループ：　SETUP

関連コマンド：　CH<x>:AMPLitude、CH<x>:FILTer、CH1:OPERation、CH<x>:TRACk:AMPLitude、CH<x>:TRACk:OFFSet、CH<x>:WAVeform

シンタックス：　AWG2005/10/11/20/21型

CH<x>:OFFSet <Offset>

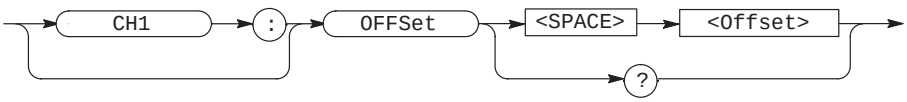
CH<x>:OFFSet?



AWG2040/AWG2041型：

[CH1:] OFFSet <Offset>

[CH1:] OFFSet?



アーギュメント：　<Offset>::=<NR2>[<unit>]

<unit>::={V | mV}

設定範囲：－5.000 V ～ 5.000 V、ステップ 0.005 V(AWG2005型)

設定範囲：－2.500 V ～ 2.500 V、ステップ 0.005 V(AWG2010/11型)

設定範囲：－2.500 V ～ 2.500 V、ステップ 0.005 V(AWG2020/21型)

設定範囲：－1.000 V ～ 1.000 V、ステップ 0.001 V(AWG2040/41型)

使用例：　以下は、オフセット電圧を 50 mV に設定する例です。

```
:CH1:OFFSET 50.0mV
```

CH<x>:TRACk?

(AWG2005/10/11/20/21型のみ)

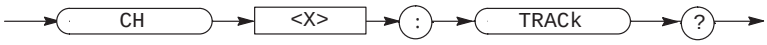
CH<x>:TRACk? 問い合わせコマンドは、指定チャネル上の振幅およびオフセットの連動の全設定を問い合わせます。

ただし、CH2、CH3、CH4 のみが有効なヘッダ・ニーモニックとなります。

グループ： SETUP

関連コマンド： CH<x>:TRACk:AMPLitude、CH<x>:TRACk:OFFSet

シンタックス： CH<x>:TRACk?



アーギュメント：

レスポンス： 詳しくは、使用例を参照ください。

使用例： 以下は、:CH2:TRACK? のレスポンス例です。

```
:CH2:TRACK:AMPLITUDE  OFF; OFFSET  OFF
```

CH<x>:TRACk:AMPLitude　(?)

(AWG2005/10/11/20/21型のみ)

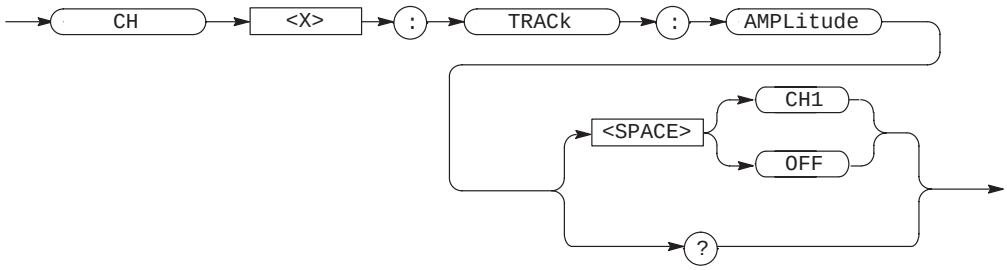
CH<x>:TRACk:AMPLitude コマンドは、ヘッダで指定するチャンネルの振幅の連動を設定します。また、CH<x>:TRACk:AMPLitude? 問い合わせコマンドは、設定中のヘッダで指定するチャンネルの振幅の連動を問い合わせます。

ただし、CH2、CH3、CH4 のみが有効なヘッダ・ニーモニックとなります。

グループ： SETUP

関連コマンド： CH<x>:TRACk?、CH<x>:TRACk:OFFSet

シンタックス： CH<x>:TRACk:AMPLitude {CH1 | OFF}
CH<x>:TRACk:AMPLitude?



アーギュメント： CH1 — CH1 の電圧レンジに連動します。
OFF — 振幅の連動機能は使用しません。

使用例： 以下は、CH2 の電圧レンジを CH1 の電圧レンジに連動させる例です。

```
:CH2:TRACK:AMPLITUDE  CH1
```

CH<x>:TRACk:OFFSet (?)

(AWG2005/10/11/20/21型のみ)

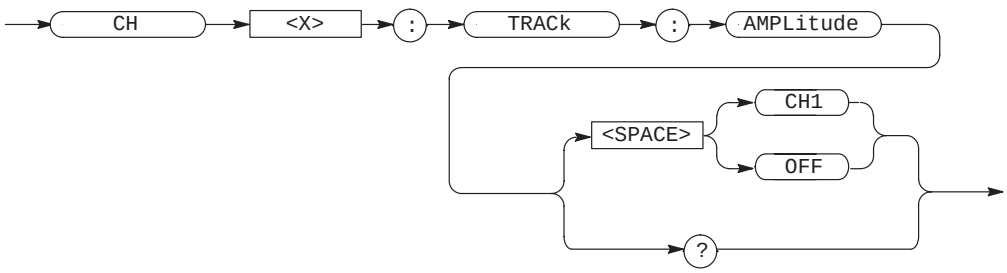
CH<x>:TRACk:OFFSet コマンドは、ヘッダで指定するチャンネルのオフセットの連動を設定します。また、CH<x>:TRACk:OFFSet? 問い合わせコマンドは、設定中のヘッダで指定するチャンネルのオフセットの連動を問い合わせます。

ただし、CH2、CH3、CH4 のみが有効なヘッダ・ニーモニックとなります。

グループ： SETUP

関連コマンド： CH<x>:TRACk?、CH<x>:TRACk:AMPLitude

シンタックス： CH<x>:TRACk:OFFSet {CH1 | OFF}
CH<x>:TRACk:OFFSet？



アーギュメント： CH1 — CH1 の電圧レンジに連動します。
OFF — オフセットの連動機能は使用しません。

使用例： 以下は、CH2 の電圧レンジを CH1 の電圧レンジに連動させる例です。

```
:CH2:TRACK:OFFSET  CH1
```

CH<x>:WAVeform　(?)

CH<x>:WAVeform コマンドは、ヘッダで指定するチャンネルから出力する波形またはシーケンスをファイルで指定します。また CH<x>:WAVeform? 問い合わせコマンドは、指定中の波形またはシーケンスのファイルを問い合わせます。

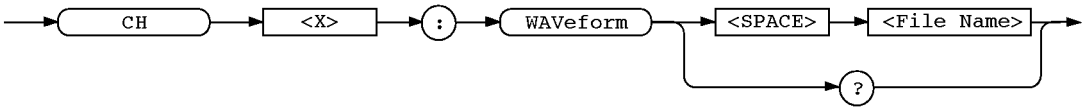
デュアル・チャンネル・オプション (オプション02型) がインストールされていない場合には、CH1 のみが有効なヘッダ・ニーモニックとなります。

グループ： SETUP

関連コマンド： CH<x>:AMPLitude、CH<x>:FILTer、CH<x>:OFFSet、CH<x>:TRACk:AMPLitude、CH<x>:TRACk:OFFSet、CH1:OPERation

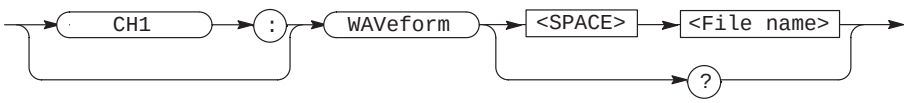
シンタックス： AWG2005/AWG2010/AWG2011/AWG2020/AWG2021型：

```
CH<x>:WAVeform　<File Name>
CH<x>:WAVeform?
```



AWG2040/AWG2041型：

```
[CH1:] WAVeform <File Name>
[CH1:] WAVeform?
```



アーギュメント： <File Name>::=<文字列> 波形ファイル名またはシーケンス・ファイル名

使用例： 以下は、波形ファイル SQUARE.WFM を指定する例です。

```
:CH1:WAVEFORM　”SQUARE.WFM”
```

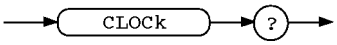
CLOCk?

CLOCk? 問い合わせコマンドは、クロックに関係する設定中の全設定値を問い合わせます。

グループ： SETUP

関連コマンド： CLOCk:FREQuency、CLOCk:SOURce、CLOCk:CH2:DIVider、CLOCk:SWEep

シンタックス： CLOCk?



アーギュメント：

使用例： 以下は、AWG2020型での CLOCK? のレスポンス例です。

```
:CLOCK:FREQUENCY 1.000E+8; SOURCE  INTERNAL; CH2:DIVIDER  1
```

CLOCk:CH2?

(AWG2010/11/20/21型のみ)

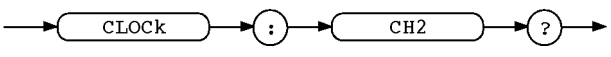
CLOCk:CH2? 問い合わせコマンドは、チャンネル 2 のクロックに関係するの設定中の全設定値を問い合わせます。

なお本コマンドは、オプション 02 型 (2 チャンネル出力) がインストールされている場合にのみ有効です。

グループ： SETUP

関連コマンド： CLOCk:CH2:DIVider

シンタックス： CLOCk:CH2?



アーギュメント：

使用例： 以下は、:CLOCK:CH2? のレスポンス例です。

```
:CLOCK:CH2:DIVIDER  1
```

CLOCk:CH2:DIVider (?)

(AWG2010/11/20/21型のみ)

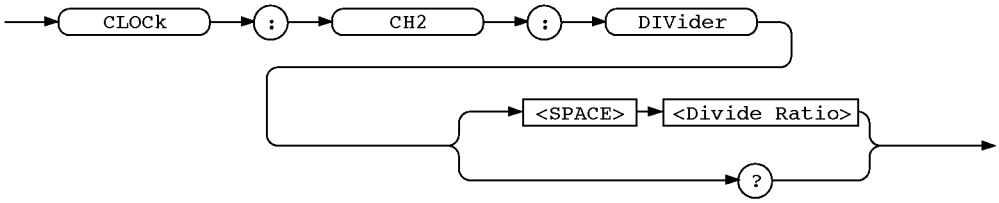
CLOCk:CH2:DIVider コマンドは、デバイダ(分周器)に対して分周比を設定します。分周された内部クロックの周波数は、チャンネル 2 のクロック周波数となります。CLOCk:CH2:DIVider? 問い合わせコマンドは、設定中の分周比を問い合わせます。

なお本コマンドは、オプション02型 (2チャンネル出力)がインストールされている場合にのみ有効です。

グループ： SETUP

関連コマンド： CLOCk？、CLOCk:FREQuency、CLOCk:SOURce

シンタックス： CLOCk:CH2:DIVider <Divide Ratio>
CLOCk:CH2:DIVider?



アーギュメント： <Divide Ratio>::=<NR1>
設定範囲：1 ～ 224、設定値は 2 のベキ乗でなければなりません。

使用例： 以下は、分周比を 256 (29) に設定する例です。

:CLOCK:CH2:DIVIDER 256

以下は、:CLOCK:CH2:DIVIDER? に対するレスポンス例です。

:CLOCK:CH2:DIVIDER 256

CLOCk:FREQuency (?)

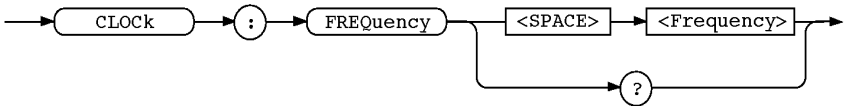
CLOCk:FREQuency コマンドは、内部クロックの周波数を設定します。また CLOCk:FREQuency? 問い合わせコマンドは、内部クロックの設定中の周波数を問い合わせます。

なお本コマンドは、クロック源として内部クロックが選択されている場合に設定してください (CLOCk:SOURceコマンド参照)。

グループ： SETUP

関連コマンド： CLOCk:SOURce、CLOCk:CH2:DIVider

シンタックス： CLOCk:FREQuency　<Frequency>
CLOCk:FREQuency?



アーギュメント： <Frequency>::=<NR3>[<unit>]

<unit>::={Hz | kHz | MHz | GHz}
設定範囲：10.00E－3 ～ 20.00E+6 Hz　(AWG2005型)
設定範囲：10.0 ～ 100.0E+6 Hz　(AWG2010/11型)
設定範囲：10.0 ～ 250.0E+6 Hz　(AWG2020/21型)
設定範囲：1.000000E+3 ～ 1.024000E+9 Hz　(AWG2040/41型)

使用例： 以下は、クロック源として内部クロックを選択し、その周波数を 245 kHz に設定する例です。

```
:CLOCK:SOURCE  INTERNAL ;FREQUENCY   245.0KHz
```

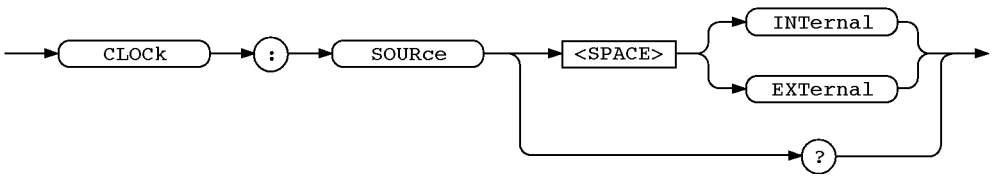
CLOCk:SOURce　(?)

CLOCk:SOURce コマンドは、クロック源を選択します。また CLOCk:SOURce? 問い合わせコマンドは、選択中のクロック源を問い合わせます。

グループ： SETUP

関連コマンド： CLOCk?、CLOCk:FREQuency、CLOCk:CH2:DIVider、CLOCk:SWEep:STATe

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



アーギュメント： INTernal　— 内部クロックをクロック源にします。
EXTernal　— 外部コネクタを通して供給される外部クロックをクロック源にします。

内部クロックを選択した場合には、CLOCk:FREQuency コマンドでクロック周波数を設定することができます。

使用例： 以下は、クロック源として、外部クロックを選択する例です。

```
:CLOCK:SOURCE  EXTERNAL
```

CLOCk:SWEep:DEFine　(?)

(AWG2005 型のみ)

CLOCk:SWEep:DEFine コマンドは、任意クロック・スイープ・データを、指定のファイルに書き込みます。

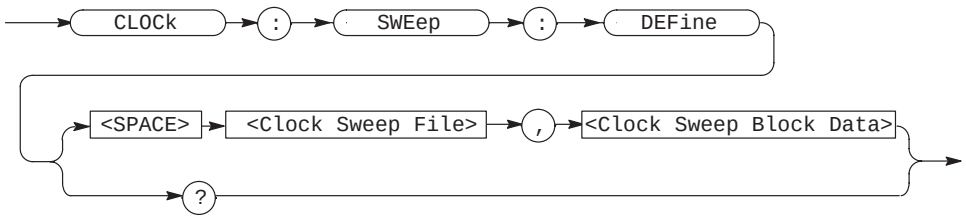
CLOCk:SWEep:DEFine? 問い合わせコマンドは、指定のファイルから、任意クロック・スイープ・データを読み出します。

なお本コマンドは、オプション 05 型がインストールされている場合にのみ有効です。

グループ： SETUP

関連コマンド： CLOCk:SWEep:DWELl、CLOCk:SWEep:TYPE

シンタックス：
CLOCk:SWEep:DEFine <Clock Sweep File>, <Clock Sweep Block Data>
CLOCk:SWEep:DEFine? <Clock Sweep File>



アーギュメント：
<Clock Sweep file>::=<string>　　クロック・スイープ・ファイル
<Clock Sweep Block Data>::=<Arbitrary Block　>　クロック・スイープ・データ

クロック・スイープ・データは、次の形式で指定します。

```
<dwell><clock (1)><event (1)><clock (2)><event (2)>  ...  <clock (N)><event (N)>
```

<dwell>::= 倍精度浮動小数点
<clock>::= 倍精度浮動小数点
<event>::= 符号なし 16 ビット整数

使用例： 以下は、クロック・スイープ・ファイル SWEEP.CLK に、1000 ステップのクロック・スイープ・データを書き込む例です。

```
:CLOCK:SWEEP:DEFINE  "SWEEP.CLK",  #510008...
```

CLOCk:SWEep:DWELl　(?)

(AWG2005型のみ)

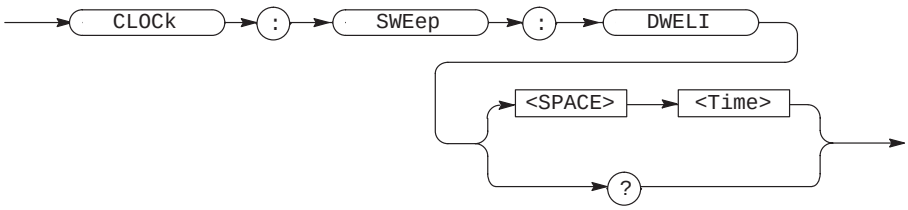
CLOCk:SWEep:DWELl コマンドは、クロック・スイープの種類がアービトラリの場合に一つの周波数をどれだけの間出力するかの時間を設定します。また、CLOCk:SWEep:DWELl? 問い合わせコマンドは、一つの周波数をどれだけの間出力するかを問い合わせます。

本コマンドは、オプション 05 型 (クロック・スイープ) がインストールされている場合にのみ有効です。

グループ： SETUP

関連コマンド： CLOCk:SWEep:TYPE

シンタックス： CLOCk:SWEep:DWELl <Time>
CLOCk:SWEep:DWELl?



アーギュメント： <Time> ::= <NR3>[<unit>]

<unit> ::= {s | ms | us}
設定範囲： 1us ～ 65.535ms

使用例： 以下は、一つの周波数を 1 ms の間出力する場合の例です。

:CLOCK:SWEEP:DWELl 1MS

CLOCk:SWEep:FREQuency?

(AWG2005型のみ)

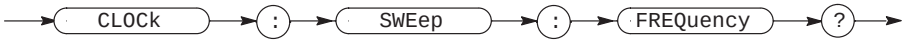
CLOCk:SWEep:FREQuency? 問い合わせコマンドは、クロック・スイープの開始点および終了点の周波数を問い合わせます。

本コマンドは、オプション 05 型 (クロック・スイープ) がインストールされている場合にのみ有効です。

グループ： SETUP

関連コマンド： CLOCk:SWEep:FREQuency:STARt、CLOCk:SWEep:FREQuency:STOP、
CLOCk:SWEep:TYPE

シンタックス： CLOCk:SWEep:FREQuency?



アーギュメント：

レスポンス： 詳しくは、使用例を参照ください。

使用例： 以下は、CLOCk:SWEep:FREQuency? のレスポンス例です。

:CLOCK:SWEEP:FREQUENCY:START 1.00000E+06; STOP 20.0000E+06

CLOCk:SWEep:FREQuency:STARt　(?)

(AWG2005型のみ)

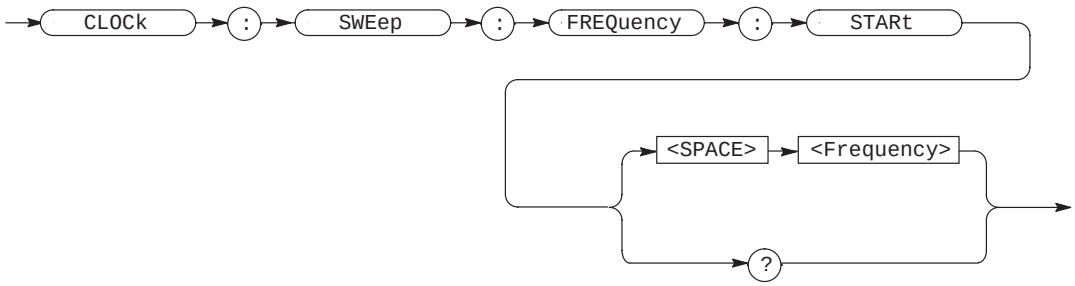
CLOCk:SWEep:FREQuency:STARt コマンドは、クロック・スイープの開始点の周波数を設定します。また、CLOCk:SWEep:FREQuency:STARt? 問い合わせコマンドは、クロック・スイープの開始点の周波数を問い合わせます。

本コマンドは、オプション 05 型 (クロック・スイープ) がインストールされている場合にのみ有効です。

グループ： SETUP

関連コマンド： CLOCk:SWEep:FREQuency?、CLOCk:SWEep:FREQuency:STOP、CLOCk:SWEep:TYPE

シンタックス： CLOCk:SWEep:FREQuency:STARt <Frequency>
CLOCk:SWEep:FREQuency:STARt?



アーギュメント： <Frequency> ::= <NR3>[<unit>]

<unit> ::= {Hz | KHz | MHz}
設定範囲： 0.01 Hz ～ 20.0000 MHz

使用例：　以下は、クロック・スイープの開始点の周波数を設定する例です。

```
:CLOCK:SWEEP:FREQUENCY:START  1000
```

CLOCk:SWEep:FREQuency:STOP　(?)

(AWG2005型のみ)

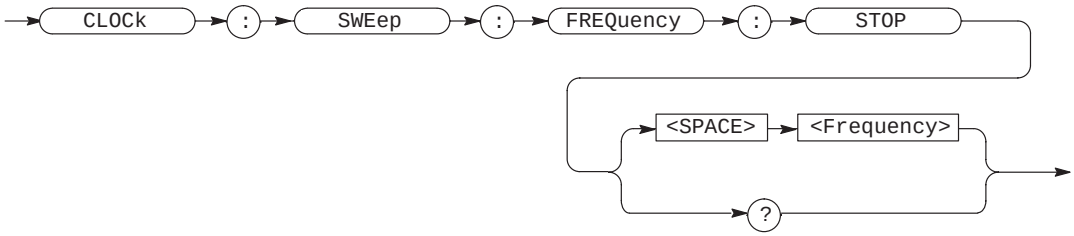
CLOCk:SWEep:FREQuency:STOP コマンドは、クロック・スイープの終了点の周波数を設定します。また、CLOCk:SWEep:FREQuency:STOP? 問い合わせコマンドは、クロック・スイープの終了点の周波数を問い合わせます。

本コマンドは、オプション 05 型 (クロック・スイープ) がインストールされている場合にのみ有効です。

グループ：　SETUP

関連コマンド：　CLOCk:SWEep:FREQuency?、CLOCk:SWEep:FREQuency:STARt、CLOCk:SWEep:TYPE

シンタックス：　CLOCk:SWEep:FREQuency:STOP <Frequency>
CLOCk:SWEep:FREQuency:STOP?



アーギュメント：　<Frequency> ::= <NR3>[<unit>]

<unit> ::= {Hz | KHz | MHz}
設定範囲： 0.01 Hz ～ 20.0000 MHz

使用例：　以下は、クロック・スイープの終了点の周波数を設定する例です。

```
:CLOCK:SWEEP:FREQUENCY:STOP  20KHZ
```

CLOCk:SWEep:MODE (?)

(AWG2005型のみ)

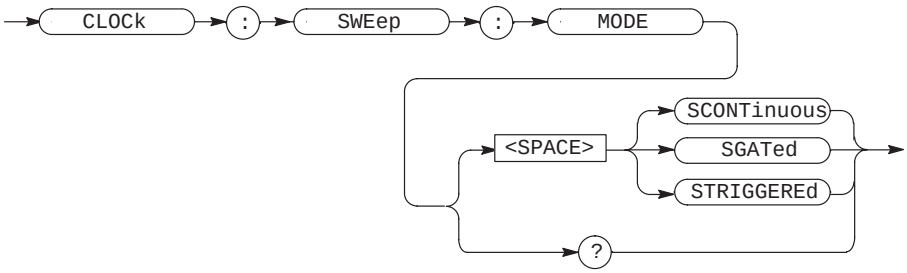
CLOCk:SWEep:MODE コマンドは、スイープ・モードを設定します。また、CLOCk:SWEep:MODE? 問い合わせコマンドは、設定中のスイープ・モードを問い合わせます。

本コマンドは、オプション 05 型 (クロック・スイープ) がインストールされている場合にのみ有効です。

グループ： SETUP

関連コマンド： CLOCk:SWEep:STAte、MODE

シンタックス： CLOCk:SWEep:MODE {SCONTinuous | SGATed | STRIGGERED}
CLOCk:SWEep:MODE?



アーギュメント：
- SCONTinuous — スイープ・モードを Continuous モードに設定します。Continuous モードは、連続的にスイープを行ないます。
- SGATed — スイープ・モードを Gated モードに設定します。Gated モードは、ゲート信号が有効である間だけスイープを行ないます。
- STRIGGERED — スイープ・モードを Triggered モードに設定します。Triggered モードは、トリガが発生した際にスイープを行ないます。

使用例： 以下は、スイープ・モードを Gated モードに設定する例です。

```
:CLOCK:SWEEP:MODE SGATED
```

CLOCk:SWEep:STATe (?)

(AWG2005型のみ)

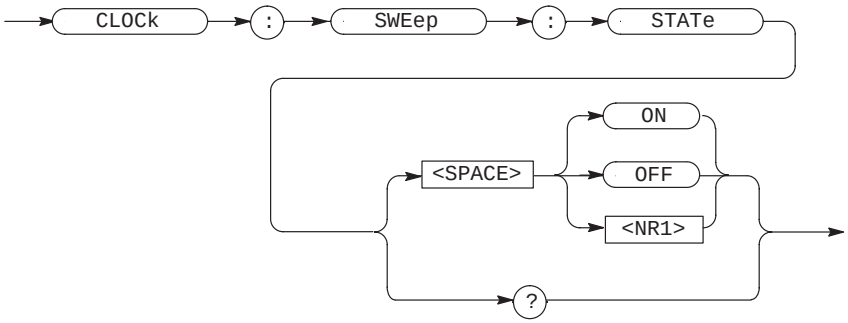
CLOCk:SWEep:STATe コマンドは、クロック・スイープを ON または OFF にします。また、CLOCk:SWEep:STATe? 問い合わせコマンドは、クロックのスイープが ON かどうかの問い合わせを行ないます。

本コマンドは、オプション 05 型 (クロック・スイープ) がインストールされている場合にのみ有効です。

グループ： SETUP

関連コマンド： CLOCk:SOURce、MODE

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



アーギュメント： ON または 0 以外の値　— クロックのスイープを ON にします。
OFF または 0　— クロックのスイープを OFF にします。

レスポンス： 問い合わせに対するレスポンスは以下の通りです。

1　— クロックのスイープが ON になっています。
0　— クロックのスイープが OFF になっています。

使用例： 以下は、クロックのスイープを ON にする例です。

```
:CLOCK:SWEEP:STATE ON
```

CLOCk:SWEep:TIME (?)

(AWG2005型のみ)

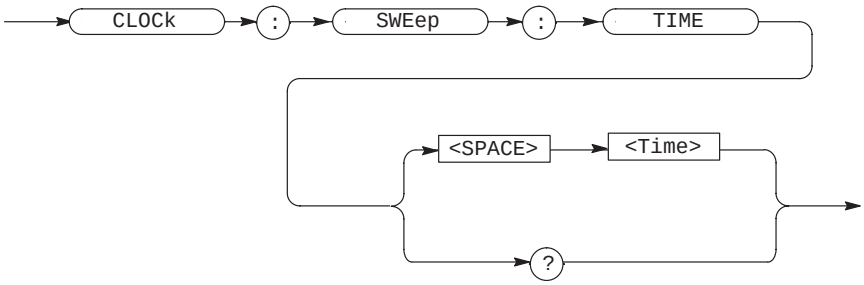
CLOCk:SWEep:TIME コマンドは、スイープを始めてから終わるまでの一周期の時間を設定します。また、CLOCk:SWEep:TIME? 問い合わせコマンドは、設定されているスイープを始めてから終わるまでの一周期の時間を問い合わせます。

本コマンドは、オプション05型（クロック・スイープ）がインストールされている場合にのみ有効です。

グループ： SETUP

関連コマンド： CLOCk:SWEep:TYPE

シンタックス： CLOCk:SWEep:TIME <Time>
CLOCk:SWEep:TIME?



アーギュメント： <Time> ::= <NR3>[<unit>]

<unit> ::= {s | ms | us}
設定範囲： 1 ms ～ 65.535 s

使用例： 以下は、スイープを始めてから終わるまでの一周期の時間を 5 ms に設定する場合の例です。

```
:CLOCK:SWEEP:TIME 5MS
```

CLOCk:SWEep:TYPE (?)

(AWG2005型のみ)

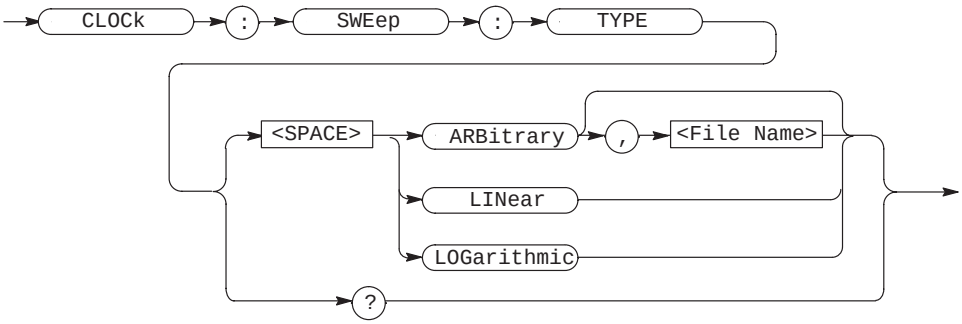
CLOCk:SWEep:TYPE コマンドは、クロック・スイープの種類を設定します。また、CLOCk:SWEep:TYPE? 問い合わせコマンドは、設定されているクロック・スイープの種類を問い合わせます。

本コマンドは、オプション 05 型 (クロック・スイープ) がインストールされている場合にのみ有効です。

グループ： SETUP

関連コマンド： CLOCk:SWEep:DWELl、CLOCk:SWEep:FREQuency:STARt、CLOCk:SWEep:FREQuency:STOP、CLOCk:SWEep:TIME

シンタックス： CLOCk:SWEep:TYPE {ARBitrary[,<File Name>] | LINear | LOGarithmic}
CLOCk:SWEep:TYPE?



アーギュメント： <File> ::= <文字列>　クロック・スイープ・ファイル (.CLK)

ARBitrary	—	クロック・スイープ・ファイルの内容に従い周波数を変化させてスイープを行ないます。クロック・スイープ・ファイルを指定しない場合は、前回指定したファイルとなります。
LINear	—	直線的に周波数を変化させてスイープを行ないます。
LOGarithmic	—	対数的に周波数を変化させてスイープを行ないます。

使用例： 以下は、クロック・スイープの種類を ARBitrary に設定する例です。

```
:CLOCK:SWEEP:TYPE  ARBITRARY, ”CLKSWEEP.CLK”
```

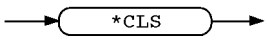
*CLS

*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)、イベント・キューをクリアします。

グループ： STATUS & EVENT

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

シンタックス： *CLS



使用例： 以下は、SESR、SBR およびイベント・キューをクリアする例です。

```
*CLS
```

CONFigure (?)

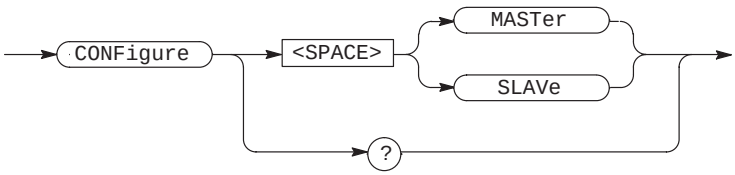
(AWG2005型のみ)

CONFigure コマンドは、AWG2005型を並列動作させる場合のコントロール信号およびクロック信号の入出力をコントロールします。また、CONFigure? 問い合わせコマンドは、並列動作させた場合の動作モードを問い合わせます。

グループ： MODE

関連コマンド： MODE、CLOCk:SOURce

シンタックス： CONFigure {MASTer | SLAVe}
CONFigure?



アーギュメント：
MASTer — 並列動作しているスレーブの AWG2005 型に対して、コントロール信号およびクロック信号を供給します。
SLAVe — 並列動作しているマスタの AWG2005 型から、コントロール信号およびクロック信号を受け取ります。

使用例： 以下は、CONFIGURE? のレスポンス例です。

```
:CONFIGURE  MASTER
```

CURVe (?)

CURVe コマンドは、外部コントローラから、本機器の DATA:DESTination コマンドで指定する場所に、アンスケールの波形データを転送します。

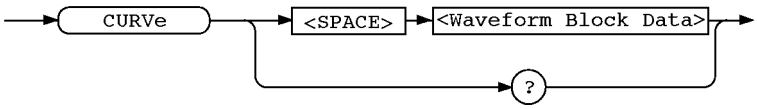
CURVe? 問い合わせコマンドは、本機器の DATA:SOURce コマンドで指定される場所から、外部コントローラに、アンスケールの波形データを転送します。

アンスケールの波形データは、プリアンブル情報も使って絶対スケールの波形データに変換することができます。波形転送については、本章の「波形転送について」（2-179 ページ）を参照ください。

グループ： WAVEFORM

関連コマンド： WAVFrm?、WFMPre?、DATA:SOURce、DATA:DESTination、DATA:ENCDG、DATA:WIDTH

シンタックス： CURVe　<Waveform Block Data>
CURVe?



アーギュメント： <Waveform Block Data>::=<Arbitrary Block> 波形データ。

使用例： 以下は、外部コントローラから本機器に波形を転送する例です。

:CURVE 3256 ...

#3256 は、256 バイトのバイナリ・データが転送されることを示しています。

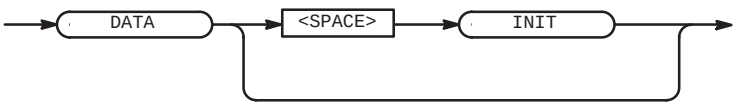
DATA　(?)

DATA コマンドは、波形転送またはマーカ・データ転送に関係する設定中の全設定をデフォルト設定に戻します。また、DATA? 問い合わせコマンドは、波形転送またはマーカ・データ転送に関係する設定中の全設定値を問い合わせます。

グループ： WAVEFORM

関連コマンド： DATA:DESTination、DATA:ENCDG、DATA:SOURce、DATA:WIDTh

シンタックス： DATA INIT
DATA?



アーギュメント： INIT　— 波形転送またはマーカ・データ転送に関係する設定中の全設定をデフォルト設定に戻します。

使用例： 以下は、DATA? に対するレスポンス例です。

:DATA:DESTINATION ”GPIB.WFM”;ENCDG RPBINARY;SOURCE”CH1”;WIDTH 2

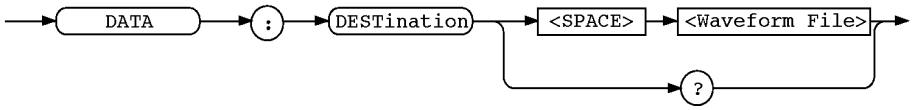
DATA:DESTination (?)

DATA:DESTination コマンドは、CURVe コマンドで本機器に転送される波形データ,または MARKer:DATA コマンドで本機器に転送されるマーカ・データの格納場所(デスティネーション)を指定します。また DATA:DESTination? 問い合わせコマンドは、設定中の波形またはマーカ・データの格納場所を問い合わせます。

グループ： WAVEFORM

関連コマンド： CURVe、MARKER<x>:AOFF、MARKER<x>:POINt、MAKer:DATA

シンタックス： DATA:DESTination <Waveform File>
DATA:DESTination?



アーギュメント： <Waveform File>::=<文字列> 波形ファイル

格納場所は、内部メモリの波形ファイルでなければなりません。波形またはマーカ・データが転送されると指定した波形ファイルが既に存在する場合には、内容が上書きされます。さらにその波形が波形メモリにロードされている場合には、出力内容も転送した波形データに変わります。

使用例： 以下は、転送波形データの格納場所を波形ファイル WAVE_EXT.WFM とする例です。

```
:DATA:DESTINATION ”WAVE_EXT.WFM”
```

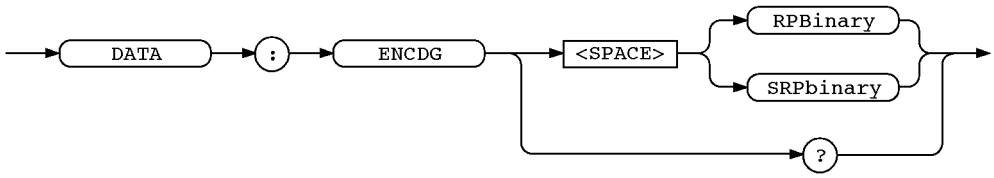
DATA:ENCDG　(?)

DATA:ENCDG コマンドは、データ幅が 2 バイトの時に CURVe コマンドや WAVFrm? 問い合わせコマンドで転送する波形データのフォーマットを選択します。また DATA:ENCDG? 問い合わせコマンドは、選択中の波形データのフォーマットを問い合わせます。

グループ： WAVEFORM

関連コマンド： CURVe、WAVFrm?、WFMPre:ENCDG、WFMPre:BYT_OR、WFMPre:BIT_NR、DATA:WIDTH

シンタックス： DATA:ENCDG　{RPBinary | SRPbinary}
DATA:ENCDG?



アーギュメント： RPBinary　— 正の整数表現が使用されます。この場合、上位バイトが先に転送されます。
SRPbinary　— 正の整数表現が使用されます。この場合、下位バイトが先に転送されます。

データ転送の際のバイト・オーダは、WFMPre:BYT_OR コマンドでも設定することができます。本コマンドおよび WFMPre:BYT_OR コマンドが同時に使用された場合には、一番最後の設定のみが有効となります。例えば、本コマンドにより、上位バイトが先に転送されるように設定されていても (DATA:ENCDG RPBinary)、WFMPre:BYT_OR LSB が実行されれば、下位バイトが先に転送されるように設定が変更されます。

使用例： 以下は、転送波形データのフォーマットを指定する例です。この例では、各データは正の整数値で表され、上位バイトが先に転送されます。

```
:DATA:ENCDG  RPBINARY
```

DATA:SOURce (?)

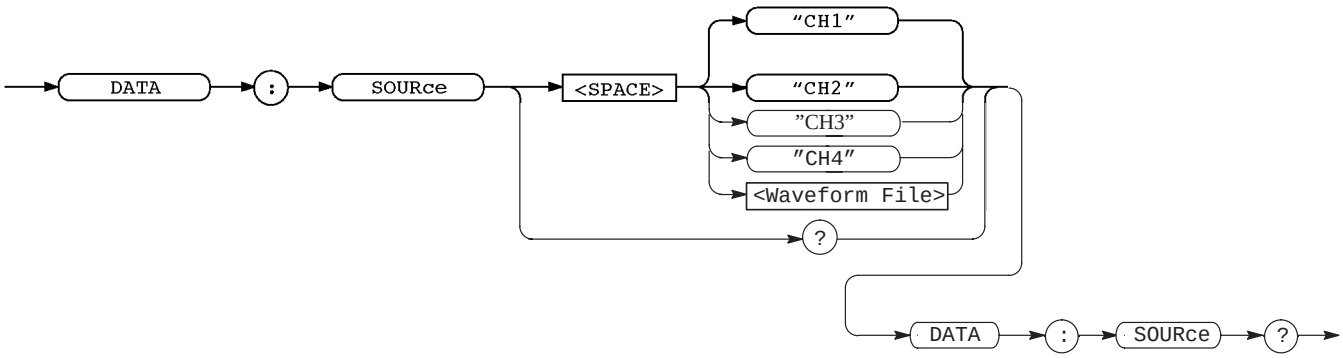
DATA:SOURce コマンドは、転送されるべき波形データが格納されている場所 (ソース)を指定します。また DATA:SOURce? 問い合わせコマンドは、設定中のソースを問い合わせます。

グループ： WAVEFORM

関連コマンド： CURVe?、WAVFrm?、MARKER<x>:POINt?、MARKer:DATA?

シンタックス： DATA:SOURce {"CH1"| "CH2" (AWG2005/10/11/20/21型) | "CH3" (AWG2005型) | "CH4" (AWG2005型) | <Waveform File>}

DATA:SOURce?



アーギュメント： <Waveform File>::=<文字列> 波形ファイル

"CH1" — チャンネル1
"CH2" — チャンネル2 (AWG2005/10/11/20/21型)
"CH3" — チャンネル3 (AWG2005型)
"CH4" — チャンネル4 (AWG2005型)

ファイルの格納場所 (ソース) は、いずれかのチャンネルまたは波形ファイルでなければなりません。チャンネルを指定する場合には、それぞれに波形がロードされていなければなりません (CH<x>:WAVeform コマンド参照)。

レスポンス： 以下は、ソースとして CH1 を指定する例です。

:DATA:SOURCE "CH1"

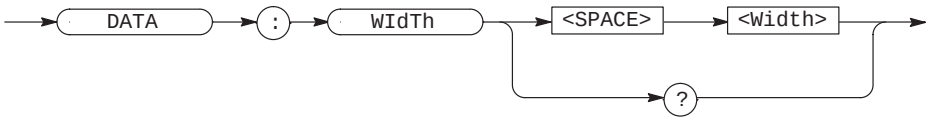
DATA:WIDTh　(?)

DATA:WIDTh コマンドは、波形データの転送時のデータ・ポイント当たりのバイト数を設定します。また、DATA:WIDTh? 問い合わせコマンドは、波形データの転送時のデータ・ポイント当たりのバイト数を問い合わせます。

グループ： WAVEFORM

関連コマンド： DATA:ENCDG、WFMPre:BIT_NR、WFMPre:BYT_NR、WFMPre:BIT_OR

シンタックス： DATA:WIDTh <Width>
DATA:WIDTh?



アーギュメント： <Width> ::= <NR1>　（設定範囲： 1 あるいは 2）

データ転送の際のデータ幅は、WFMPre:BYT_NR コマンドでも設定することができます。本コマンドおよび WFMPre:BYT_NR コマンドが同時に使用された場合には、一番最後の設定のみが有効となります。例えば、本コマンドによりデータ幅が 1 バイトと設定されても (DATA:WIDTH 1)、WFMPre:BYT_NR 2 が実行されれば、データ幅は 2 バイトです。

使用例： 以下は、波形データ転送時のデータ・ポイント当たりのバイト数を 1 バイトに設定する例です。

```
:DATA:WIDTH 1
```

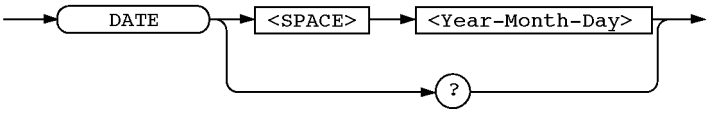
DATE　(?)

DATE コマンドは、日付を設定します。また DATE? 問い合わせコマンドは、現在の日付を問い合わせます。

グループ： SYSTEM

関連コマンド： TIME

シンタックス： DATE　<Year-Month-Day>
DATE?



アーギュメント： <Year-Month-Day>::=<文字列>　以下のフォーマットでなければなりません。

”YYYY–MM–DD”

ここで

YYYY	西暦（4桁表現）
MM	月　（1～12）
DD	日　（01～31ただし月の指定によって変わります）。

使用例： 以下は、日付を設定する例です。

:DATE　”1993–11–11”

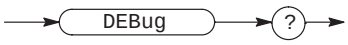
DEBug?

DEBug? 問い合わせコマンドは、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。本コマンドは、DEBug:SNOop? 問い合わせコマンドと全く同等です。

グループ： SYSTEM

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

シンタックス： DEBug?



アーギュメント：

レスポンス： 詳しくは、使用例を参照ください。

使用例： 以下は、DEBUG? のレスポンス例です。

:DEBUG:SNOOP:STATE　0;DELAY:TIME　0.2

DEBug:SNOop？

DEBug:SNOop? 問い合わせコマンドは、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。本コマンドは、DEBug? 問い合わせコマンドと全く同等です。

グループ： SYSTEM

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

シンタックス： DEBug:SNOop?

アーギュメント：

レスポンス： 詳しくは、使用例を参照ください。

使用例： 以下は、DEBUG:SNOOP? のレスポンス例です。

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

DEBug:SNOop:DELAy？

DEBug:SNOop:DELAy? 問い合わせコマンドは、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。本コマンドは、DEBug:SNOop:DELAy:TIME? 問い合わせコマンドと全く同等です。

グループ： SYSTEM

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：

レスポンス： レスポンス・フォーマットは以下の通りです。

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

ここで

<Delay time> ::= <NR2>

使用例： 以下は、DEBUG:SNOOP:DELAY? のレスポンス例です。

:DEBUG:SNOOP:DELAY:TIME 0.2

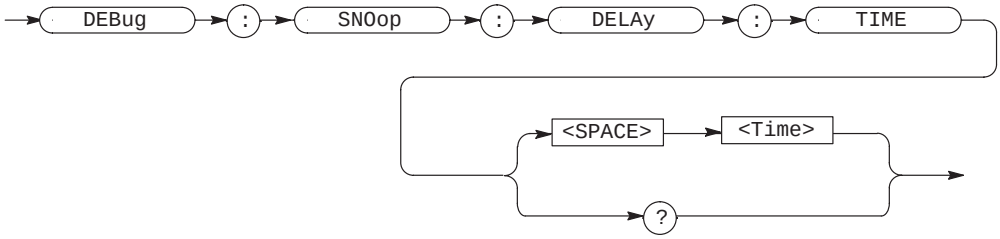
DEBug:SNOop:DELAy:TIME (?)

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? 問い合わせコマンドは、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： SYSTEM

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： 以下は、コマンドの表示時間を 0.5 s に設定する例です。

: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ないます。また、DEBug:SNOop:STATe? 問い合わせコマンドは、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、 ";" で区切られるつどに DEBug:SNOop:DELAy:TIME で設定される時間だけ各々表示されます。

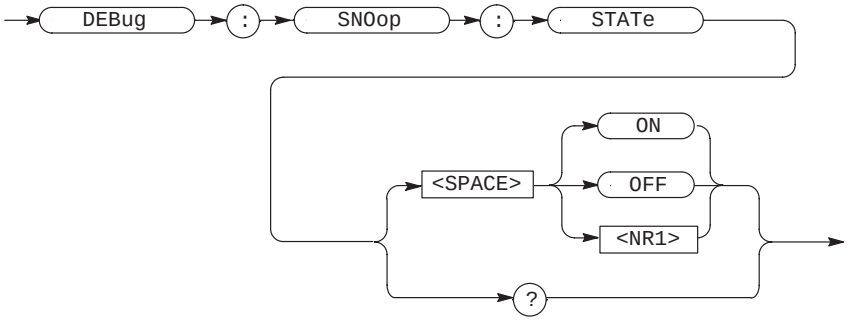
表示形式は、以下の通りです。

コントロール・コード	—	"<コードの10進表示>"：例えば、LF の場合は "<10>"。
英数字・シンボル	—	"<コードのASCII表示>"：例えば、"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>" および <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>"。例えば、コード80 (16進) の場合は、<128>。

グループ： SYSTEM

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

シンタックス： DEBug:SNOop:STATe {ON | OFF | <NR1>}
DEBug:SNOopp: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： 問い合わせに対するレスポンスは以下の通りです。

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す。

使用例： 以下は、デバッグ機能を設定する例です。

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

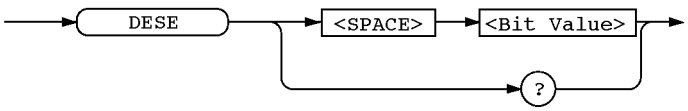
DESE　(?)

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? 問い合わせコマンドは、DESER の値を問い合わせます。

グループ： STATUS & EVENT

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、全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。

使用例： 以下は、DESER を 177 (10110001) に設定する例です。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
```

以下は、:DESE? のレスポンス例です。

```
:DESE  176
```

この場合 DESER の内容は、10110000 となります。

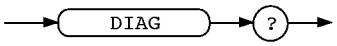
DIAG？

DIAG? 問い合わせコマンドは、選択中のセルフ・テスト・ルーチンとその実行結果を問い合わせます。

グループ： CALIBRATION & DIAGNOSTIC

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

シンタックス： DIAG?



アーギュメント：

レスポンス： レスポンス・フォーマットは以下の通りです。

[:DIAG:SELECT] <Self-test Routine>; [RESULT] <Result>[,<Result>]...

ここで <Self-test Routine>は、以下のいずれかです。

ALL	—	全てのセルフ・テスト・ルーチン
CPU	—	CPU ユニット・テスト・ルーチン
CLOCk	—	CLOCK ユニット・テスト・ルーチン
DISPlay	—	DISPLAY ユニット・テスト・ルーチン
FPP	—	FPP (Floating-Point Processor) ユニット・テスト・ルーチン
FPANel	—	FPC (Front Panel Control) ユニット・テスト・ルーチン
SETUp	—	SETUP 関連ユニット・テスト・ルーチン
TRIGger	—	TRIGGER ユニット・テスト・ルーチン
WMEMory	—	波形メモリ・テスト・ルーチン

<Result>::=<NR1>　以下のいずれかです。

0	—	正常終了
100	—	CPU ユニットでエラーを検出
200	—	CLOCK ユニットでエラーを検出
300	—	DISPLAY ユニットでエラーを検出
400	—	FPP ユニットでエラーを検出
500	—	FPC ユニットでエラーを検出
600	—	SETUP 関連ユニットでエラーを検出
700	—	波形メモリでエラーを検出
800	—	TRIGGER ユニットでエラーを検出

注： セルフ・テスト中に他のコマンドを実行しようとしても本機器は反応しません。

使用例： 以下は、:DIAG? のレスポンス例です。

:DIAG:SELECT ALL;RESULT 0

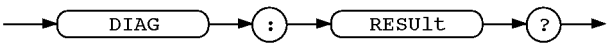
DIAG：RESUlt？

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

グループ： CALIBRATION & DIAGNOSTIC

関連コマンド： DIAG:SELect、DIAG:STATe

シンタックス： DIAG:RESUlt?



アーギュメント：

レスポンス： レスポンス・フォーマットは以下の通りです。

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

ここで <Result>::=<NR1> 以下のいずれかです。

- 0 — 正常終了
- 100 — CPU ユニットでエラーを検出
- 200 — CLOCK ユニットでエラーを検出
- 300 — DISPLAY ユニットでエラーを検出
- 400 — DSP ユニットでエラーを検出
- 500 — FPC ユニットでエラーを検出
- 600 — SETUP 関連ユニットでエラーを検出
- 700 — 波形メモリでエラーを検出
- 800 — TRIGGER ユニットでエラーを検出

使用例： 以下は、:DIAG:RESULT? のレスポンス例です。

:DIAG: RESULT 200, 600

上記の場合、CLOCK ユニットと波形メモリでエラーが検出されたことを示しています。

DIAG:SELect　(?)

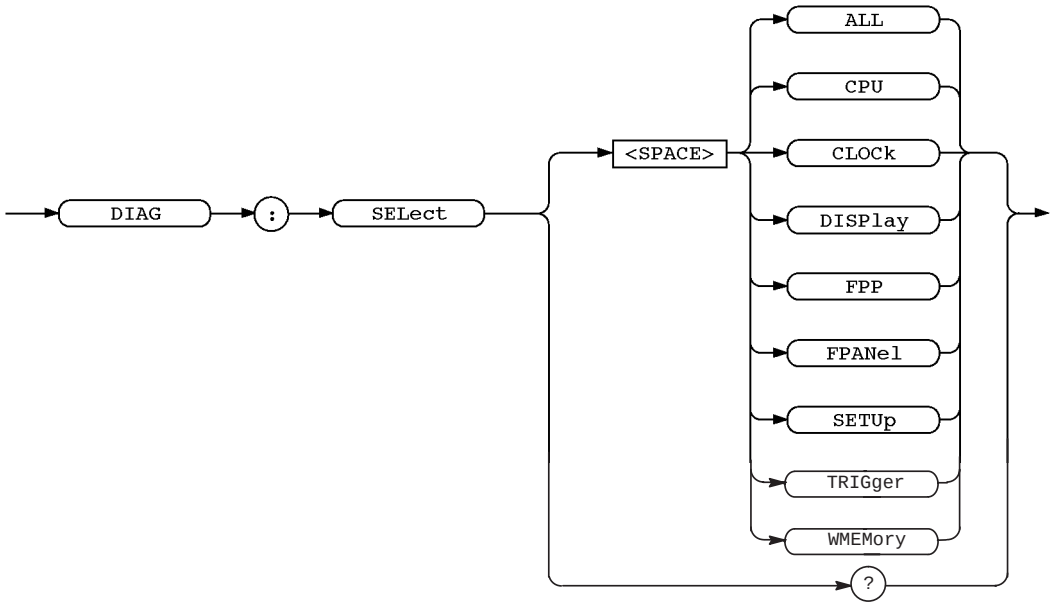
DIAG:SELect コマンドは、セルフ・テスト・ルーチンを選択します。また DIAG:SELect? 問い合わせコマンドは、選択中のセルフ・テスト・ルーチンを問い合わせます。

グループ： CALIBRATION & DIAGNOSTIC

関連コマンド： DIAG:STATe、DIAG:RESULt?

シンタックス： DIAG:SELect　{ALL | CPU | CLOCk | DISPlay | FPP | FPANel | SETUp | TRIGger | WMEMory}

DIAG:SELect?



アーギュメント：

ALL	—	以下の全てのセルフ・テスト・ルーチン
CPU	—	CPU ユニット・テスト・ルーチン
CLOCk	—	CLOCK ユニット・テスト・ルーチン
DISPlay	—	DISPLAY ユニット・テスト・ルーチン
FPP	—	FPP (Floating-Point Processor) ユニット・テスト・ルーチン
FPANel	—	FPC (Front Panel Control) ユニット・テスト・ルーチン
SETUp	—	SETUP 関連ユニット・テスト・ルーチン
TRIGger	—	TRIGGER ユニット・テスト・ルーチン
WMEMory	—	波形メモリ・テスト・ルーチン

使用例： 以下は、CPU ユニット・テスト・ルーチンを選択して実行する例です。

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

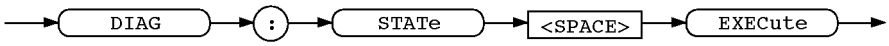
DIAG:STATe

DIAG:STATe コマンドは、DIAG:SELect コマンドで選択されたセルフ・テスト・ルーチンを実行します。各ルーチンは、実行中にエラーを検出すると実行を中止します。また全てのセルフ・テスト・ルーチンが選択されている場合には、エラーを検出したルーチンの実行を中止して、次のルーチンの実行に移ります。

グループ： CALIBRATION & DIAGNOSTIC

関連コマンド： DIAG:SELect、DIAG:RESUlt?

シンタックス： DIAG:STATe EXECute



アーギュメント： EXECute — 選択中のセルフ・テスト・ルーチンを実行します。

使用例： 以下は、全てのセルフ・テスト・ルーチンを実行する例です。

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

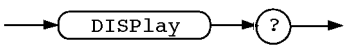
DISPlay?

DISPlay? 問い合わせコマンドは、DISPLAY コマンドで可能な全ての設定に対して、設定中の値を問い合わせます。

グループ： DISPLAY

関連コマンド：

シンタックス： DISPlay?



アーギュメント：

レスポンス： 詳しくは、使用例を参照ください。

使用例： 以下は、:DISPLAY? のレスポンス例です。

```
:DISPLAY:BRIGHTNESS 70; CATALOG:ORDER NAME1;DISPLAY:CLOCK0;
MENU:SETUP:FORMAT GRAPHICS;:DISPLAY:MESSAGE:SHOW ""
```

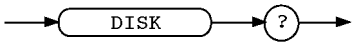
DISK?

DISK? 問い合わせコマンドは、フロッピー・ディスクに関連する全ての設定に対し、設定中の値を問い合わせます。

グループ： MEMORY

関連コマンド：

シンタックス： DISK?



アーギュメント：

レスポンス： 詳しくは、使用例を参照ください。

使用例： 以下は、:DISK? のレスポンス例です。

```
:DISK:FORMAT:TYPE  HD3
```

DISK:CDIRectory

DISK:CDIRectory コマンドは、フロッピー・ディスク上のカレント・ワーキング・ディレクトリを変更します。

グループ： MEMORY

関連コマンド： DISK:DIRectory?、DISK:MDIRectory

シンタックス： DISK:CDIRectory　<Directory>

アーギュメント： <Directory>::=<文字列>　移動後のカレント・ワーキング・ディレクトリ

使用例： 以下は、\FG\WORK3 を新しいカレント・ワーキング・ディレクトリとする例です。

```
:DISK:CDIRECTORY  "\FG\WORK3"
```

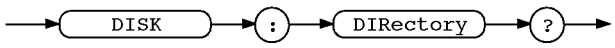
DISK:DIRectory？

DISK:DIRectory? 問い合わせコマンドは、フロッピー・ディスク上のカレント・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： DISK:CDIRectory、DISK:MDIRectory

シンタックス： DISK:DIRectory?



アーギュメント：

使用例： 以下は、:DISK:DIRECTORY? のレスポンス例です。

```
:DISK:DIRECTORY  "\FG\WORK3"
```

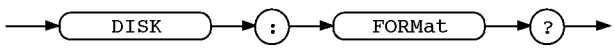
DISK:FORMat?

DISK:FORMat? 問い合わせコマンドは、DISK:FORMat:TYPE コマンドで設定中のフォーマット・タイプを問い合わせます。

グループ： MEMORY

関連コマンド： DISK:FORMat:TYPE、DISK:FORMat:STATe

シンタックス： DISK:FORMat?



アーギュメント：

レスポンス： 以下のフォーマット・タイプが戻されます。

DD1 — 2DD、720 KB、IBM PC および東芝 J3100 用フォーマット
DD2 — 2DD、640 KB、NEC PC-9800 用フォーマット
HD1 — 2HD、1232 KB、NEC PC-9800 用フォーマット
HD2 — 2HD、1200 KB、東芝 J3100 用フォーマット
HD3 — 2HD、1440 KB、IBM PC 用フォーマット

使用例：　以下は、:FORMAT:TYPE? に対するレスポンス例です。

```
:DISK:FORMAT:TYPE  HD3
```

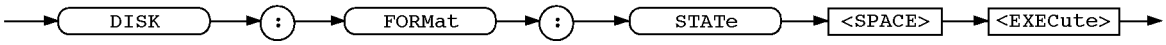
DISK:FORMat:STATe

DISK:FORMat:STATe コマンドは、DISK:FORMat:TYPE コマンドで指定するフォーマット・タイプに従って、フロッピー・ディスクを初期化します。

グループ：　MEMORY

関連コマンド：　DISK:FORMat:TYPE

シンタックス：　DISK:FORMat:STATe　EXECute



アーギュメント：　EXECute　—　フロッピー・ディスクを初期化します。

使用例：　以下は、フロッピー・ディスクを IBM PC 用 2DD にフォーマットする例です。

```
:DISK:FORMAT:TYPE  DD1 ;STATE  EXECUTE
```

DISK:FORMat:TYPE　(?)

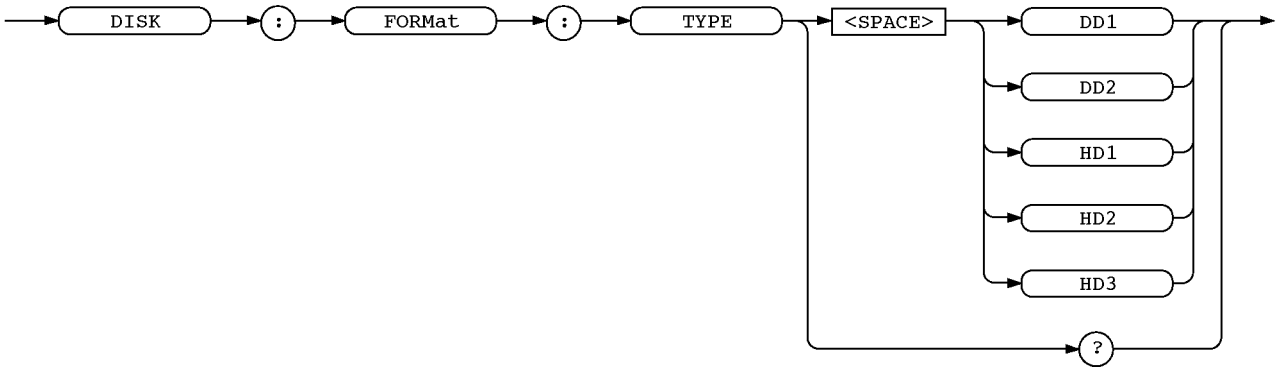
DISK:FORMat:TYPE コマンドは、フロッピー・ディスクを初期化する際のフォーマット・タイプを指定します。なおフロッピー・ディスクの初期化は、DISK:FORMat:STATe コマンドで行います。

また DISK:FORMat:TYPE? 問い合わせコマンドは、設定中のフォーマット・タイプを問い合わせます。

グループ：　MEMORY

関連コマンド：　DISK:FORMat:STATe

シンタックス：　DISK:FORMat:TYPE　{DD1 | DD2 | HD1 | HD2 | HD3}
DISK:FORMat:TYPE?



アーギュメント： フォーマット・タイプは以下の通りです。

使用例： 以下は、フロッピー・ディスクを IBM PC 用 2HD にフォーマットする例です。

```
:DISK:FORMAT:TYPE HD3 ;STATE  EXECUTE
```

DISK:MDIRectory

DISK:MDIRectory コマンドは、フロッピー・ディスク上に新しくディレクトリを作成します。

グループ： MEMORY

関連コマンド： DISK:CDIRectory、DISK:DIRectory？

シンタックス： DISK:MDIRectory　<Directory>

アーギュメント： <Directory>::=<文字列>　ディレクトリ名

使用例： 以下は、カレント・ワーキング・ディレクトリに WORK4 というディレクトリを作成する例です。

```
:DISK:MDIRECTORY  ”WORK4”
```

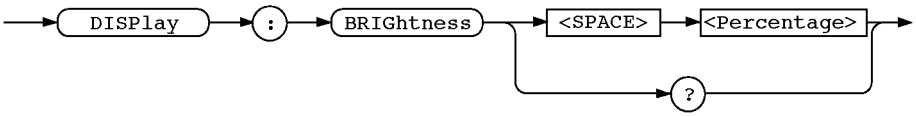
DISPlay:BRIGhtness　(?)

DISPlay:BRIGhtness コマンドは、管面の輝度をパーセンテージで設定します。また DISPlay:BRIGhtness? 問い合わせコマンドは、設定中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　<Percentage>
DISPlay:BRIGhtness?



アーギュメント： <Percentage>::=<NR1> [<unit>]　　輝度 (設定範囲: :　0～100 %、ステップ 1 %)
<unit>::=PCT

使用例： 以下は、管面の輝度を 80 % に設定する例です。

```
:DISPLAY:BRIGHTNESS  80
```

DISPlay:CATalog?

DISPlay:CATalog? 問い合わせコマンドは、設定されているカタログ表示の並び替えの条件を問い合わせます。本コマンドは、DISPlay:CATalog:ORDer? 問い合わせコマンドと全く同等です。

グループ： DISPLAY

関連コマンド： DISPlay:CATalog:ORDer

シンタックス： DISPlay:CATalog?

アーギュメント：

レスポンス： レスポンス・フォーマットは以下の通りです。

[:DISPLAY:CATALOG:ORDER] <Catalog order>

ここで <Catalog order> は、以下の 通りです。

NAME1 — ファイル名 (Name) のアスキ・コード順に並び替えます。
NAME2 — NAME1の逆順に並び替えます。
TIME1 — 日付時刻 (Date & Time) の新しい順に並び替えます。
TIME2 — 日付時刻 (Date & Time) の古い順に並び替えます。
TYPE1 — 拡張子 (Type) のアルファベット順に並び替えます。
TYPE2 — 拡張子 (Type) のアルファベット順で、かつ、ファイル名 (Name) のアスキ・コードの逆順に並び替えます。
TYPE3 — 拡張子 (Type) のアルファベット順で、かつ、日付時刻 (Date & Time) の新しい順に並び替えます。
TYPE4 — 拡張子 (Type) のアルファベット順で、かつ、日付時刻 (Date & Time) の古い順に並び替えます。

使用例： 以下は、DISPLAY:CATALOG? のレスポンス例です。

:DISPLAY:CATALOG:ORDER NAME1

DISPlay:CATalog:ORDer　(?)

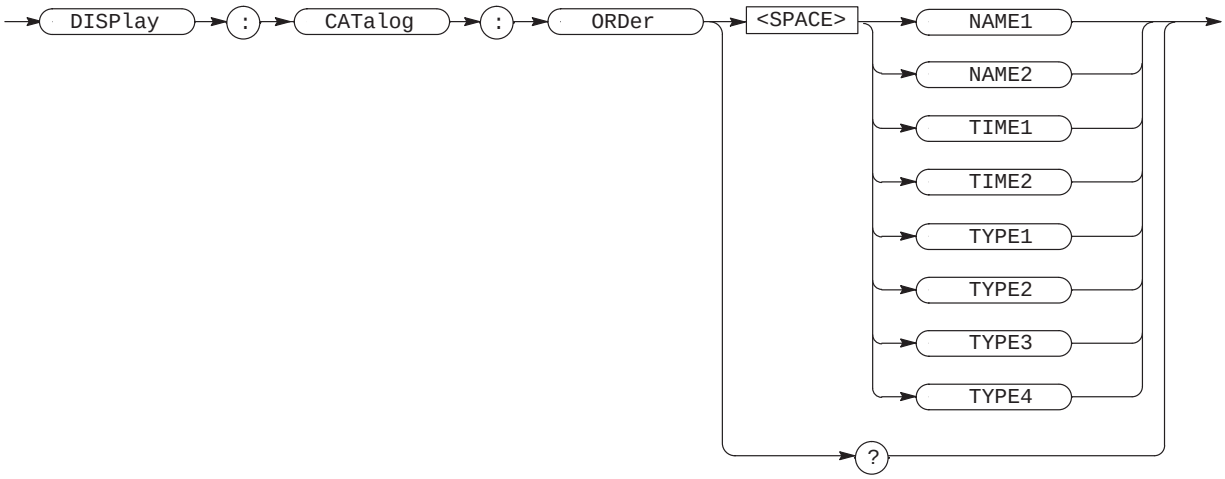
DISPlay:CATalog:ORDer コマンドは、カタログ表示の並び替えの条件を設定します。また、DISPlay:CATalog:ORDer? 問い合わせコマンドは、設定されているカタログ表示の並び替えの条件を問い合わせます。

グループ： DISPLAY

関連コマンド： DISPlay:CATalog?

シンタックス： DISPlay:CATalog:ORDer　{NAME1 | NAME2 | TIME1 | TIME2 | TYPE1 | TYPE2 | TYPE3 | TYPE4}

DISPlay:CATalog:ORDer?



アーギュメント：

NAME1	—	ファイル名 (Name) のアスキ・コード順に並び替えます。
NAME2	—	NAME1の逆順に並び替えます。
TIME1	—	日付時刻 (Date & Time) の新しい順に並び替えます。
TIME2	—	日付時刻 (Date & Time) の古い順に並び替えます。
TYPE1	—	拡張子 (Type) のアルファベット順に並び替えます。
TYPE2	—	拡張子 (Type) のアルファベット順で、かつ、ファイル名 (Name)のアスキ・コードの逆順に並び替えます。
TYPE3	—	拡張子 (Type) のアルファベット順で、かつ、日付時刻 (Date & Time) の新しい順に並び替えます。
TYPE4	—	拡張子 (Type) のアルファベット順で、かつ、日付時刻 (Date & Time) の古い順に並び替えます。

使用例： 以下は、カタログのファイル表示を日付時刻 (Date & Time) の新しい順に並び替えるよう設定します。

```
:DISPLAY:CATALOG:ORDER  TIME1
```

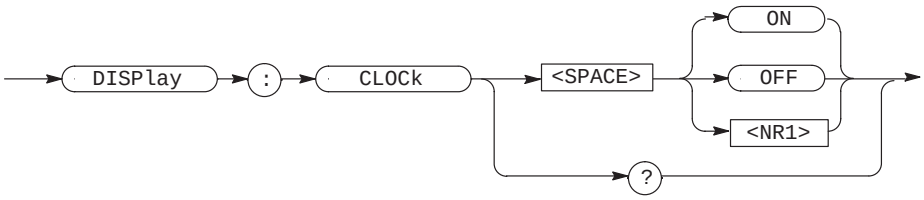
DISPlay:CLOCk (?)

DISPlay:CLOCk コマンドは、日付時刻を表示するかどうか設定します。また、DISPlay:CLOCk? 問い合わせコマンドは、日付時刻が表示されているか問い合わせを行ないます。

グループ： DISPLAY

関連コマンド：

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



アーギュメント： ON または 0 以外の値 — 日付時刻を表示します。
OFF または 0 — 日付時刻を表示しません。

レスポンス： 問い合わせに対するレスポンスは以下の通りです。

1 — 日付時刻が表示されます。
0 — 日付時刻が表示されます。

使用例： 以下の例は、日付時刻を表示するよう設定します。

```
:DISPLAY:CLOCK ON
```

DISPlay:MENU?

DISPlay:MENU? 問い合わせコマンドは、SETUP メニューの表示フォーマットを問い合わせます。本コマンドは、DISPlay:MENU:SETUp:FORMat? 問い合わせコマンドと全く同等です。

グループ： DISPLAY

関連コマンド： DISPlay:MENU:SETUp:FORMat

シンタックス： DISPlay:MENU?

アーギュメント：

レスポンス： レスポンス・フォーマットは以下の通りす。

[:DISPLAY:MENU:SETUP:FORMAT] <Menu format>

ここで <Menu format> は、
GRAPHICS　— SETUP メニューは、グラフィック表示です。
TEXT　— SETUP メニューは、テキスト表示です。

使用例： 以下は、SETUP メニューが テキスト表示時の DISPLAY:MENU? のレスポンス例です。

:DISPLAY:MENU:SETUP:FORMAT TEXT

DISPlay:MENU:SETUp?

DISPlay:MENU:SETUp? 問い合わせコマンドは、SETUP メニューの表示フォーマットを問い合わせます。本コマンドは、DISPlay:MENU:SETUp:FORMat? 問い合わせコマンドと全く同等です。

グループ： DISPLAY

関連コマンド： DISPlay:MENU:SETUp:FORMat

シンタックス： DISPlay:MENU:SETUp?

アーギュメント：

レスポンス： レスポンス・フォーマットは以下の通りす。

[:DISPLAY:MENU:SETUP:FORMAT] <Menu format>

ここで <Menu format> は、
GRAPHICS　— SETUP メニューは、グラフィック表示です。
TEXT　— SETUP メニューは、テキスト表示です。

使用例： 以下は、SETUP メニューが グラフィック 表示時の DISPLAY:MENU:SETUP? のレスポンス例です。

:DISPLAY:MENU:SETUP:FORMAT GRAPHICS

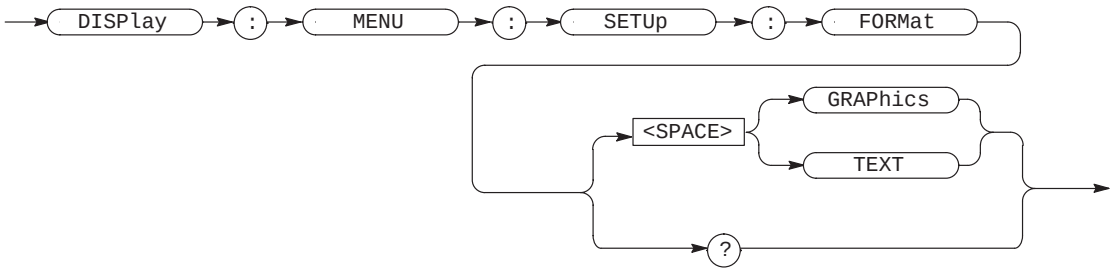
DISPlay:MENU:SETUp:FORMat (?)

DISPlay:MENU:SETUp:FORMat コマンドは、SETUP メニューの表示フォーマットを設定します。また、DISPlay:MENU:SETUp:FORMat? 問い合わせコマンドは、SETUP メニューの表示フォーマットを問い合わせます。

グループ： DISPLAY

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

シンタックス： DISPlay:MENU:SETUp:FORMat {GRAPhics | TEXT}
DISPlay:MENU:SETUp:FORMat?



アーギュメント：
GRAPhics — SETUP メニューは、グラフィック表示です。
TEXT — SETUP メニューは、テキスト表示です。

使用例： 以下は、SETUPメニューの表示をテキスト表示に 設定します。

```
:DISPLAY:MENU:SETUP:FORMAT  TEXT
```

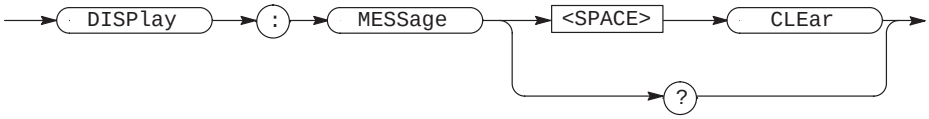
DISPlay:MESSage (?)

DISPlay:MESSage コマンドは、メッセージ領域に表示されているメッセージを消去します。また、DISPlay:MESSage? 問い合わせコマンドは、メッセージ領域に表示されているメッセージを問い合わせます。

グループ： DISPLAY

関連コマンド： DISPlay:MESSage:SHOW

シンタックス： DISPlay:MESSage CLEar
DISPlay:MESSage?



アーギュメント： CLEar　　—　メッセージ領域に表示されているメッセージを消去します。

使用例： 以下は、メッセージ領域に表示されているメッセージを消去する例です。

```
:DISPLAY:MESSAGE  CLEAR
```

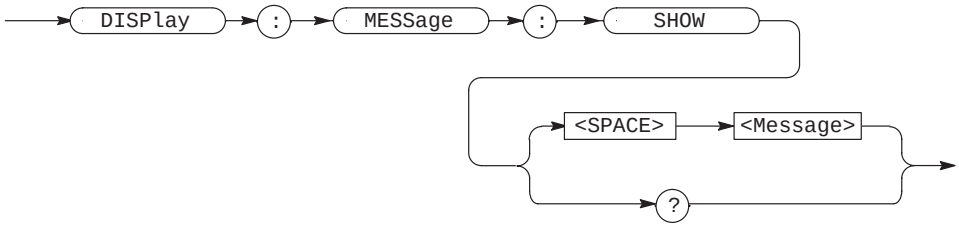
DISPlay:MESSage:SHOW　(?)

DISPlay:MESSage:SHOW コマンドは、メッセージ領域に指定されたメッセージを表示します。また、DISPlay:MESSage:SHOW? 問い合わせコマンドは、メッセージ領域に表示されているメッセージを問い合わせます。

グループ： DISPLAY

関連コマンド： DISPlay:MESSage

シンタックス：
```
DISPlay:MESSage:SHOW  <Message>
DISPlay:MESSage:SHOW?
```



アーギュメント： <Message> ::= <文字列>　最大 60 文字までのメッセージ。

使用例： 以下は、メッセージ領域に ”TEST No.1” を表示する例です。

```
:DISPLAY:MESSAGE:SHOW  ”TEST  No.1”
```

EQUAtion:COMPile　(?)

EQUAtion:COMPile コマンドは、指定するイクエーション・ファイルをコンパイルして波形ファイルに変換します。また、EQUAtion:COMPile:? 問い合わせコマンドは、コンパイル実行中かどうかを問い合わせます。本コマンドは EQUAtion:COMPile:STATe コマンドと全く同等です。

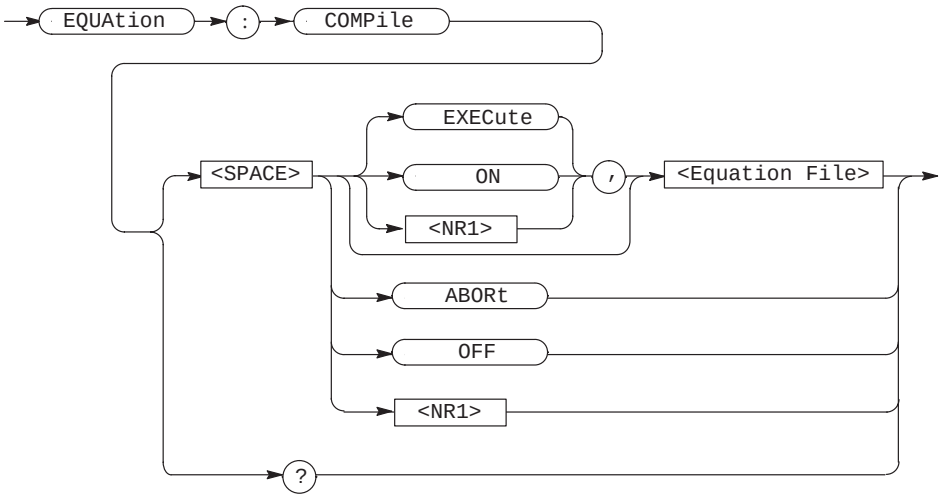
グループ： WAVEFORM

関連コマンド： EQUAtion:COMPile:STATe

シンタックス：

EQUAtion:COMPile { [EXECute, | ON, | <NR1>,] <Equation File> | ABORt | OFF | <NR1>}

EQUAtion:COMPile:STATe?



アーギュメント： <Equation File> ::= <文字列>

<Equation File> は、内部メモリのイクエーション・ファイルでなければなりません。コンパイルの結果として作成される波形データは、波形ファイルに格納されます。波形ファイルのベース名は、イクエーション・ファイルのベース名と同じです。

EXECute	—	指定されたイクエーション・ファイルをコンパイルします。
ON または 0 以外の値	—	指定されたイクエーション・ファイルをコンパイルします。
ABORt	—	実行中のコンパイルを強制終了させます。
OFF または 0	—	実行中のコンパイルを強制終了させます。

レスポンス： レスポンスは以下の通りです。

[:EQUATION:COMPILE] 1, <Equation File>	—	実行中
[:EQUATION:COMPILE] 0	—	実行中以外

使用例： 以下は、イクエーション・ファイル ”EXP_SAMP.EQU” をコンパイルして、生成された波形データを ”EXP_SAMP.WFM” に格納する例です。

```
:EQUATION:COMPILE:STATE  EXECUTE,  ”EXP_SAMP.EQU”
```

EQUAtion:COMPile:STATe　(?)

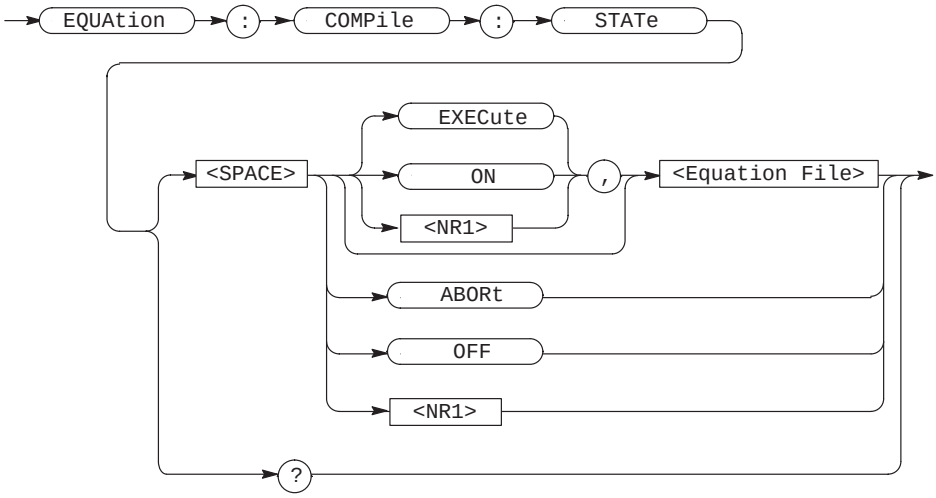
EQUAtion:COMPile:STATe コマンドは、指定するイクエーション・ファイルをコンパイルして波形ファイルに変換します。また、EQUAtion:COMPile:STATe? 問い合わせコマンドは、コンパイル実行中かどうかを問い合わせます。

グループ： WAVEFORM

関連コマンド： EQUAtion:COMPile

シンタックス： EQUAtion:COMPile:STATe { [EXECute, | ON, | <NR1>,] <Equation File> | ABORt | OFF | <NR1>}

EQUAtion:COMPile:STATe？



アーギュメント： <Equation File> ::= <文字列>

<Equation File> は、内部メモリのイクエーション・ファイルでなければなりません。コンパイルの結果として作成される波形データは、波形ファイルに格納されます。波形ファイルのベース名は、イクエーション・ファイルのベース名と同じです。

EXECute	—	指定されたイクエーション・ファイルをコンパイルします。
ON または 0 以外の値	—	指定されたイクエーション・ファイルをコンパイルします。
ABORt	—	実行中のコンパイルを強制終了させます。
OFF または 0	—	実行中のコンパイルを強制終了させます。

レスポンス： レスポンスは以下の通りです。

[:EQUATION:COMPILE:STATE]　1, <Equation File>	—	実行中
[:EQUATION:COMPILE:STATE]　0	—	実行中以外

使用例： 以下は、イクエーション・ファイル ”EXP_SAMP.EQU” をコンパイルして、生成された波形データを ”EXP_SAMP.WFM” に格納する例です。

```
:EQUATION:COMPILE:STATE  EXECUTE,  ”EXP_SAMP.EQU”
```

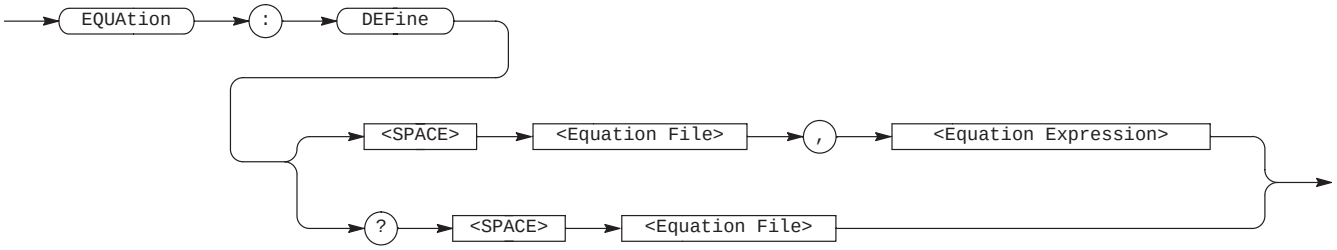
EQUAtion:DEFine　(?)

EQUAtion:DEFine コマンドは、指定するイクエーション・ファイルに、一連の演算式(コメントを含む)を書き込みます。また EQUAtion:DEFine? 問い合わせコマンドは、指定するイクエーション・ファイルに書き込まれた演算式を問い合わせます。

グループ： WAVEFORM

関連コマンド： EQUAtion:COMPile:STATe、EQUAtion:WPOints

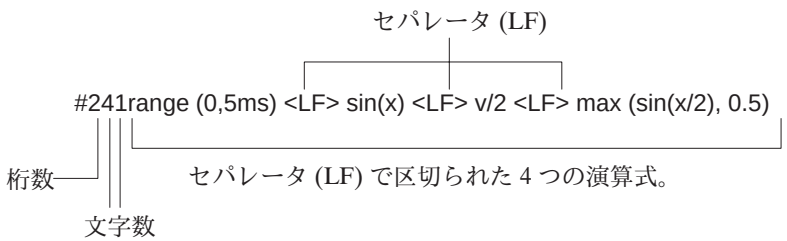
シンタックス： EQUAtion:DEFine　<Equation File>, <Equation Block Data>
EQUAtion:DEFine?　<Equation File>



アーギュメント：

<Equation File>::=<文字列>	イクエーション・ファイル
<Equation Block Data>::=<Arbitrary Block>	イクエーション

<Equation File>は、内部メモリのイクエーション・ファイルでなければなりません。演算式は、ASCII 形式で記述し、各演算式は LF コードで区切ります。イクエーション・ファイルに記述できる内容は、ユーザ・マニュアルを参照ください。なお、コメント行には印字可能な全ての ASCII コード 32 ～ 126 (10 進数) が使用可能です。



イクエーション・ファイルは、EQUAtion:COMPile:STATe コマンドで波形ファイルにコンパイルすることができます。イクエーション・ファイルに対して、EQUAtion:WPOints コマンドで波形ポイント数 (コンパイル後に使用) を設定できます。

使用例：　以下は、イクエーション・ファイル EXP_SAMP.EQU に演算式を書き込む例です。

```
:EQUATION:DEFINE  ”EXP_SAMP.EQU”, #241range(0,5ms)<LF>sin(x)<LF>
v/2<LF>max(sin ...
```

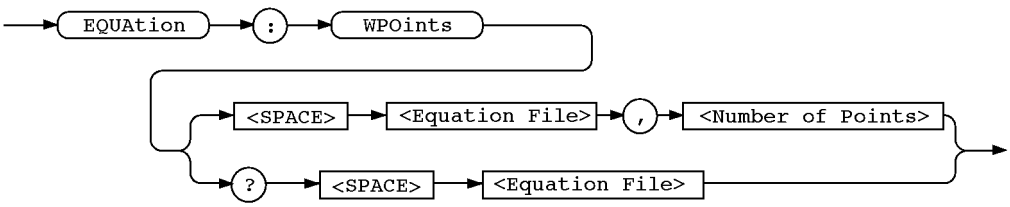
EQUAtion:WPOints　(?)

EQUAtion:WPOints コマンドは、イクエーション・データをコンパイルした場合の波形ポイント数を設定します。また EQUAtion:WPOints? 問い合わせコマンドは、設定中の波形ポイント数を問い合わせます。

グループ：　WAVEFORM

関連コマンド：　EQUAtion:COMPile:STATe、EQUAtion:DEFine

シンタックス：　EQUAtion:WPOints　<Equation File>, <Number of Points>
EQUAtion:WPOints?　<Equation File>



アーギュメント：　<Equation File>::=<文字列>　イクエーション・ファイル名

<Number of Points>::=<NR1>　　波形ポイント数 (設定範囲：　1 ～ 32768 (32K))

<Equation File> は、内部メモリのイクエーション・ファイルでなければなりません。イクエーション・ファイルは、EQUAtion:COMPiles コマンドで波形ファイルにコンパイルすることができます。

使用例：　以下は、イクエーション・ファイル EXP_SAMP.EQU に波形ポイント数 1000 を設定する例です。

```
:EQUATION:WPOINTS”EXP_SAMP.EQU”, 1000
```

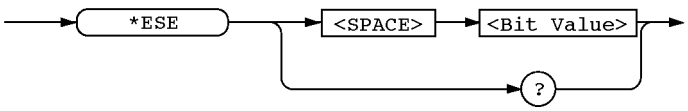
*ESE (?)

*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・問い合わせコマンドは、ESER の内容を問い合わせます。.

グループ： STATUS & EVENT

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)

アーギュメントは、0 から 255 の範囲の10進数で指定します。ESER 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 ESER の値は、PSC フラグが TRUE の場合、全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。

使用例： 以下は、ESER を 177 (10110001) に設定する例です。この場合、PON、CME、EXE、OPC の各ビットがセットされます。

```
*ESE 177
```

以下は、*ESE? のレスポンス例です。

```
176
```

この場合 ESER の内容は、10110000 となります。

*ESR?

*ESR? 共通・問い合わせコマンドは、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： STATUS & EVENT

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：

使用例： 以下は、*ESR? のレスポンス例です。

```
181
```

この場合 SESR の内容は 10110101 となります。

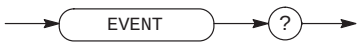
EVENT?

EVENT? 問い合わせコマンドは、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・問い合わ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”を参照ください。

グループ： STATUS & EVENT

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：

使用例： 以下は、:EVENT? のレスポンス例です。

```
:EVENT  113
```

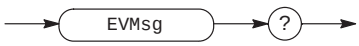
EVMsg?

EVMsg? 問い合わせコマンドは、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・問い合わ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」を参照ください。

グループ： STATUS & EVENT

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

シンタックス： EVMsg?



使用例： 以下は、:EVMSG? のレスポンス例です。

:EVMSG 420, ”Query UNTERMINATED”

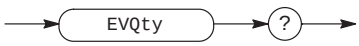
EVQty?

EVQty? 問い合わせコマンドは、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： STATUS & EVENT

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

シンタックス： EVQty?



アーギュメント：

使用例： 以下は、:EVQTY? に対するレスポンス例です。

:EVQTY 5

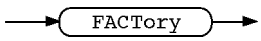
FACTory

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 を参照ください。

グループ： SYSTEM

関連コマンド： *RST、SECUre

シンタックス： FACTory



アーギュメント：

使用例： 以下は、本機器を、工場出荷時のデフォルト設定状態に戻す例です。

```
:FACTORY
```

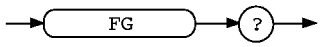
FG?

FG? 問い合わせコマンドは、FG グループのコマンドで可能な全ての設定に対して、現在設定中の値を問い合わせます。

グループ： FG

関連コマンド：

シンタックス： FG?



アーギュメント：

レスポンス： コマンドが連結された形式で戻されます。詳しくは、使用例を参照ください。

使用例： 以下は、:FG? のレスポンス例です。

```
:FG:STATE 0;FREQUENCY 2.500E+06;CH1:AMPLITUDE 1.000;OFFSET 0.000;
POLARITY NORMAL;SHAPE SINUSOID;:FG:CH2:AMPLITUDE 1.000;OFFSET
0.000; POLARITY NORMAL;SHAPE SINUSOID
```

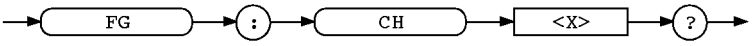
FG:CH<x>?

FG:CH<x>? 問い合わせコマンドは、関数波形のパラメタに関する全ての設定に対して、ヘッダで指定するチャンネルに設定中の値を問い合わせます。

グループ： FG

関連コマンド： FG:CH<x>:AMPLitude、FG:CH<x>:OFFSet、FG:CH<x>:POLarity、FG:CH<x>:SHAPe

シンタックス： FG:CH<x>?



アーギュメント：

レスポンス： コマンドが連結された形式で戻されます。詳しくは、使用例を参照ください。

使用例： 以下は、:FG:CH1? のレスポンス例です。

```
:FG:CH1:AMPLITUDE 1.000; OFFSET  0.000; POLARITY NORMAL;
SHAPE  SINUSOID
```

FG:CH<x>:AMPLitude　(?)

FG:CH<x>:AMPLitude コマンドは、ヘッダで指定するチャンネルに対して、関数波形の最大振幅（Vp-p）を設定します。また FG:CH<x>:AMPLitude? 問い合わせコマンドは、指定チャンネルに対して、設定中の関数波形の最大振幅 (Vp-p) を問い合わせます。

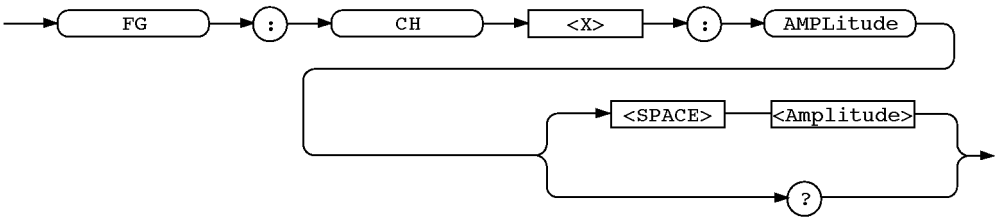
グループ： FG

関連コマンド： FG:CH<x>:OFFSet、FG:CH<x>:POLarity、FG:CH<x>:SHAPe、FG:CH<x>?

シンタックス：　AWG2005/AWG2010/AWG2011/AWG2020/AWG2021型：

FG:CH<x>:AMPLitude　<Amplitude>

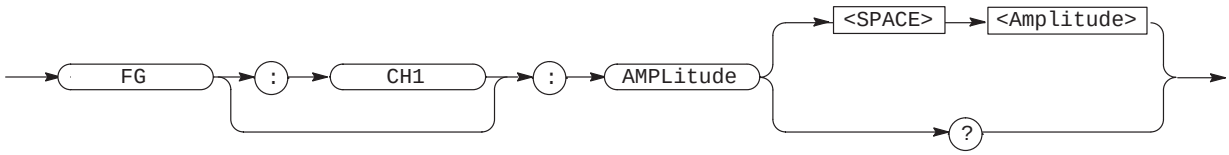
FG:CH<x>:AMPLitude?



AWG2040/AWG2041型：

FG [:CH1] AMPLitude　<Amplitude>

FG [:CH1] AMPLitude?



アーギュメント：　<Amplitude>::=<NR2>[<unit>]　　最大振幅 (Vp-p)

<unit>::={V | mV}

設定範囲：　0.050 ～ 10.000 V、ステップ 0.001 V　　(AWG2005型)

設定範囲：　0.050 ～ 7.500 V、ステップ 0.001 V　　(AWG2010/11型)

設定範囲：　0.050 ～ 5.000 V、ステップ 0.001 V　　(AWG2020/21型)

設定範囲：　0.020 ～ 2.000 V、ステップ 0.001 V　　(AWG2040/41型)

使用例：　以下は、最大振幅を 100 mVp-p に設定する例です。

:FG:CH1:AMPLITUDE　100.0mV

FG:CH<x>:OFFSet (?)

FG:CH<x>:OFFSet コマンドは、ヘッダで指定するチャンネルに対して、関数波形のオフセット電圧を設定します。また FG:CH<x>:OFFSet? 問い合わせコマンドは、指定チャンネルに対して、設定中のオフセット電圧を問い合わせます。

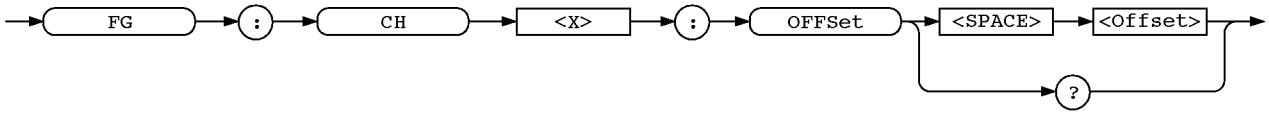
グループ： FG

関連コマンド： FG:CH<x>:AMPLitude、FG:CH<x>:POLarity、FG:CH<x>:SHAPe、FG:CH<x>?

シンタックス： AWG2005/AWG2010/AWG2011/AWG2020/AWG2021型：

FG:CH<x>:OFFSet <Offset>

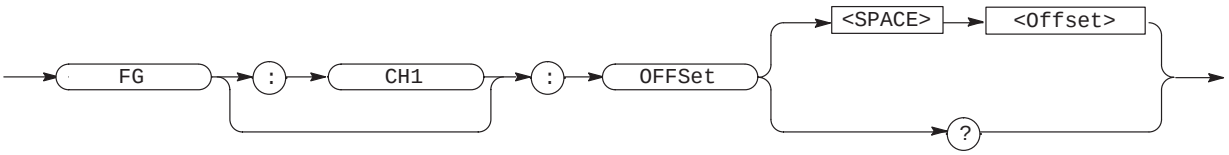
FG:CH<x>:OFFSet?



AWG2040/AWG2041型：

FG [:CH1] OFFSet <offset>

FG [:CH1] OFFSet?



アーギュメント： <Offset>::=<NR2>[<unit>] オフセット電圧

<unit>::={V | mV}

設定範囲： - 5.000 V ～ 5.000 V、ステップ 0.005 V (AWG2005型)

設定範囲： - 2.500 V ～ 2.500 V、ステップ 0.005 V (AWG2010/11型)

設定範囲： - 2.500 V ～ 2.500 V、ステップ 0.005 V (AWG2020/21型)

設定範囲： - 1.000 V ～ 1.000 V、ステップ 0.001 V (AWG2040/41型)

使用例： 以下は、チャンネル1側のオフセット電圧を 5 mV に設定する例です。

```
:FG:CH1:OFFSET 5mV
```

FG:CH<x>:POLarity　(?)

FG:CH<x>:POLarity コマンドは、ヘッダで指定するチャンネルに対して、関数波形の極性(ポーラリティ)を設定します。また FG:CH<x>:POLarity? 問い合わせコマンドは、指定チャンネルに対して、設定中の極性を問い合わせます。

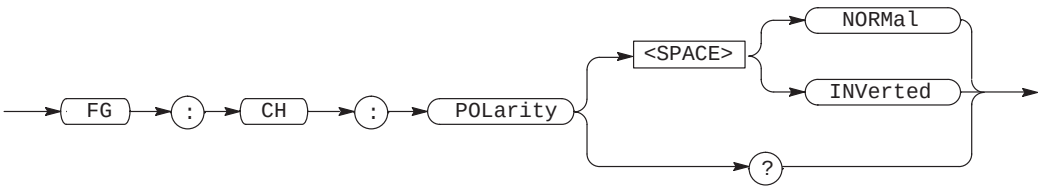
グループ： FG

関連コマンド： FG:CH<x>:AMPLitude、FG:CH<x>:OFFSet、FG:CH<x>:SHAPe、FG:CH<x>?

シンタックス： AWG2005/AWG2010/AWG2011/AWG2020/AWG2021型：

FG:CH<x>:POLarity {NORMal | INVerted}

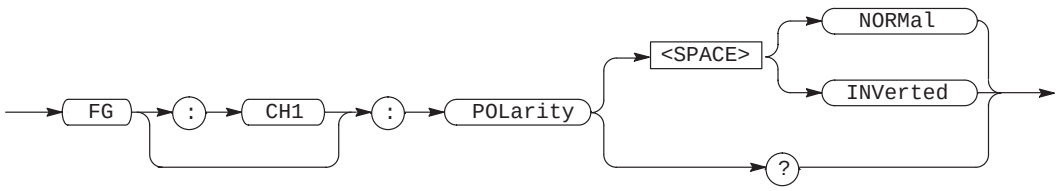
FG:CH<x>:POLarity?



AWG2040/AWG2041型：

FG [:CH1] POLarity {NORMal | INVerted}

FG [:CH1] POLarity?



アーギュメント： NORMal　— 通常の極性に設定します。

INVerted　— 極性を反転します。

使用例： 以下は、チャンネル1側の関数波形の極性を反転する例です。

```
:FG:CH1:POLARITY  INVERTED
```

FG:CH<x>:SHAPe (?)

FG:CH<x>:SHAPe コマンドは、ヘッダで指定するチャンネルに対して、標準の関数波形を選択します。関数波形が指定されると、関数波形は設定中のパラメタに従ってヘッダで指定するチャンネルから出力されます。

また FG:CH<x>:SHAPe? 問い合わせコマンドは、指定チャンネルの選択中の標準関数波形を問い合わせます。

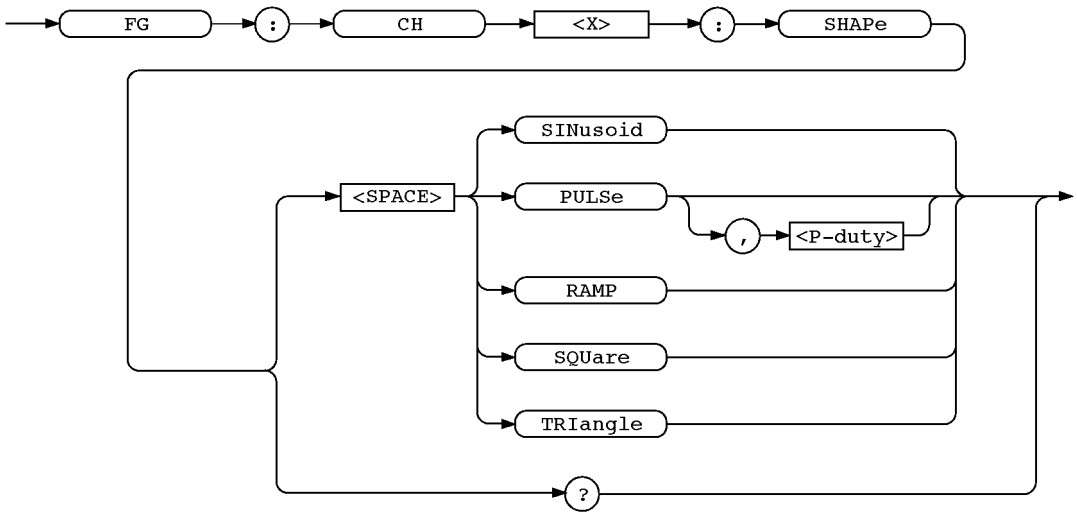
グループ： FG

関連コマンド： FG:CH<x>:AMPLitude、FG:CH<x>:POLarity、FG:CH<x>:OFFSet、FG:CH<x>?

シンタックス： AWG2005/AWG2010/AWG2011/AWG2020/AWG2021型：

FG:CH<x>:SHAPe {SINusoid | PULSe[,<P–duty>] | RAMP | SQUare | TRIangle}

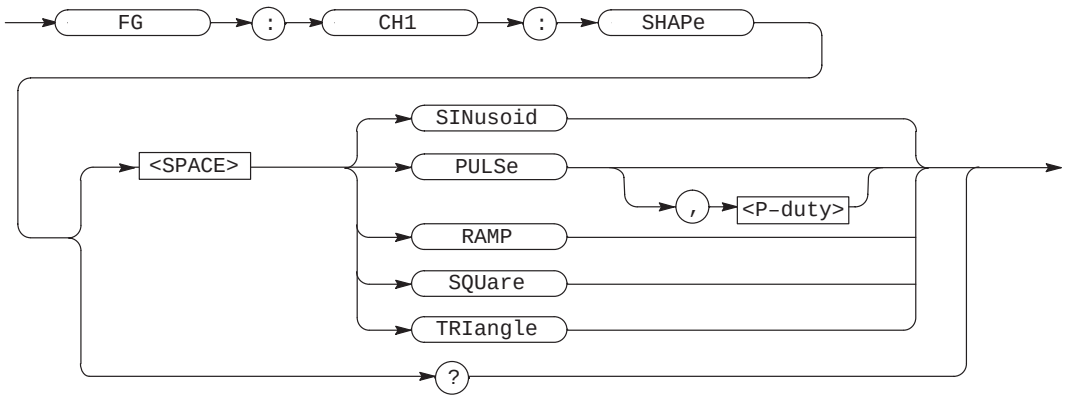
FG:CH<x>:SHAPe?



AWG2040/AWG2041型：

FG[:CH1] SHAPe

FG[:CH1] SHAPe?



アーギュメント： SINusoid　— サイン波
PULS　— パルス波。同時に、デューティ・サイクルをパルス周期の割合で指定することができます。

<P-duty>::=<NR1>[<NR1>]　(設定範囲: 0～100%、ステップ1%)
<unit>::=PCT

RAMP　— ランプ波
SQUare　— 方形波
TRIangle　— 三角波

使用例： 以下は、標準関数波形としてパルス波を選択し、デューティ・サイクルを 40 % に設定する例です。

```
:FG:CH1:SHAPE  PULSE, 40
```

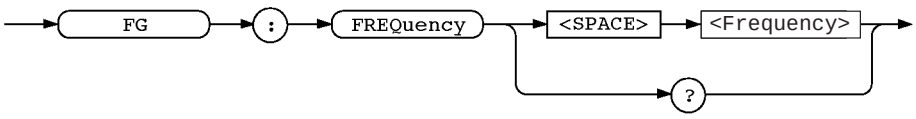
FG:FREQuency　(?)

FG:FREQuency コマンドは、選択されている関数波形に対して周波数を設定します。また FG:FREQuency? 問い合わせコマンドは、設定中の周波数を問い合わせます。

グループ： FG

関連コマンド： FG:STATe, FG?

シンタックス： FG:FREQuency　<Frequency>
FG:FREQuency?



アーギュメント： <Frequency>::=<NR3>[<unit>] 周波数
<unit>::={Hz | kHz | MHz}
設定範囲： 1.000 Hz ～ 200.0 kHz (AWG2005型)
設定範囲： 1.000 Hz ～ 1.000 MHz (AWG2010/11型)
設定範囲： 1.000 Hz ～ 2.500 MHz (AWG2020/21型)
設定範囲： 1.000000 Hz ～ 10.00000 MHz (AWG2040/41型)

使用例： 以下は、選択されている関数波形に対して、1.2 MHz の周波数を設定する例です。

```
:FG:FREQ  1.2MHz
```

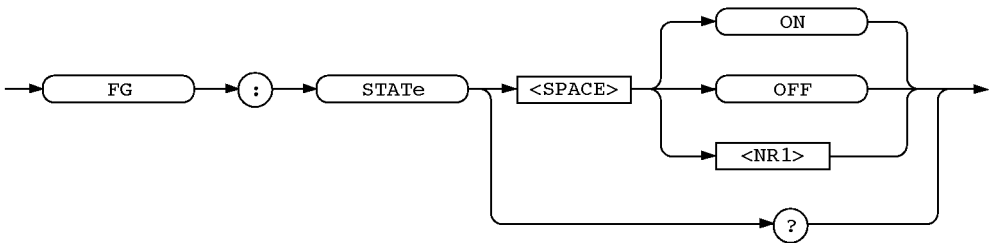
FG:STATe (?)

FG:STATe コマンドは、FG (Function Generator) モードの ON/OFF を行います。また FG:STATe? 問い合わせコマンドは、FG モードの ON/OFF 状態を問い合わせます。

グループ： FG

関連コマンド： FG?

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



アーギュメント： ONまたはゼロ以外の値 — FG モードを ON にします。
OFFまたはゼロ — FG モードを OFF にします。

レスポンス： 問い合わせに対するレスポンスは以下の通りです。

1 — FG モードが ON になっています。
0 — FG モードが OFF になっています。

使用例： 以下は、FG モードを ON にする例です。

```
:FG:STATE  1
```

HCOPy　(?)

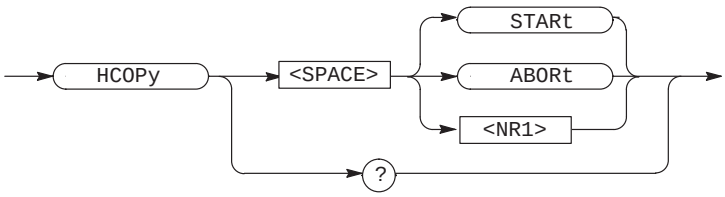
HCOPy コマンドは、設定されている出力ポートに対してハードコピー出力を開始あるいは終了させます。また、HCOPy? 問い合わせコマンドは、ハードコピーの全設定を問い合わせます。

注：　このコマンドは、IEEE-488.2-1987 規格との互換性はありません。

グループ：　HARDCOPY

関連コマンド：　HCOPy:FORMat、HCOPy:PORT、HCOPy:DATA?

シンタックス：　HCOPy {STARt | ABORt | <NR1>}
HCOPy?



アーギュメント：　STARt または 0 以外の値　—　ハードコピーを開始します。
ABORt または 0　—　ハードコピーを終了させます。

注：　HARDCopy START コマンド使用中は、*WAI コマンドで最初のハードコピーの終了を確認してから、次のハードコピーを始めてください。

使用例：　以下は、設定されている出力先に対してハードコピー出力を開始する例です。

```
:HCOPY  START
```

HCOPy:DATA?

HCOPy:DATA? 問い合わせコマンドは、ハードコピー・データを出力キューに出力します。ただし、ハードコピー出力ポートの設定には関係ありません。

グループ：　HARDCOPY

関連コマンド：　HCOPy、HCOPy:PORT

シンタックス： HCOPy:DATA?

使用例： 以下は、ハードコピー・データを出力キューに出力します。

```
:HCOPY:DATA?
```

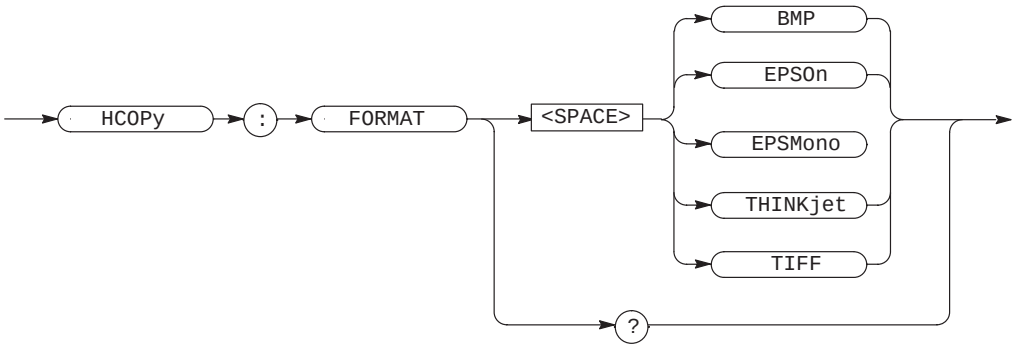
HCOPy:FORMat (?)

HCOPy:FORMat コマンドは、ハードコピー出力のフォーマットを設定します。また、HCOPy:FORMat? 問い合わせコマンドは、設定されているハードコピーの出力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のフォーマットです。

使用例： 以下は、TIFF フォーマットでハードコピーが出力されるよう設定します。

```
:HCOPY:FORMAT  TIFF
```

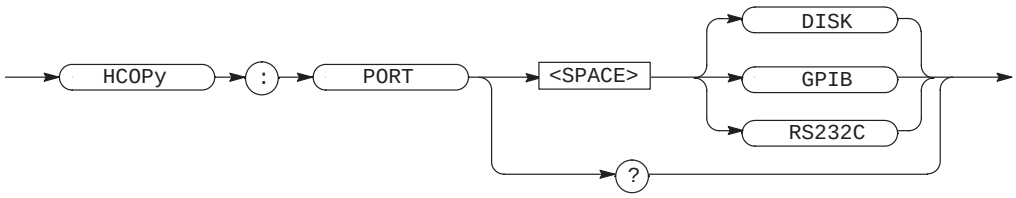
HCOPy:PORT　(?)

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? 問い合わせコマンドは、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 のポートへ出力します。
RS232c　— RS-232Cのポートへ出力します。

使用例： 以下は、ハードコピーの出力をフロッピ・ディスク上のファイルとする例です。

```
:HCOPY:PORT  DISK
```

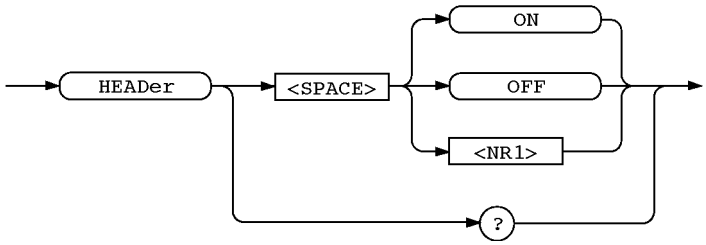
HEADer　(?)

HEADer コマンドは、IEEE std. 488.2 の共通コマンドを除く全ての問い合わ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? 問い合わせコマンドは、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： SYSTEM

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： 問い合わせコマンドに対するレスポンスは以下の通りです。

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 を参照ください。

使用例： 以下は、レスポンス中にヘッダを含める設定の例です。

```
:HEADER ON
```

以下は、:HEADER? に対するレスポンス例です。

```
: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。

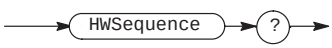
HWSequencer?
(AWG2041型のみ)

HWSequencer? コマンドはハードウェア・シーケンサを搭載しているかどうかと、現在ハードウェア・シーケンサを使用しているかどうかを問い合わせます。

グループ： SYSTEM

関連コマンド： HWSequencer:INSTalled?, HWSequencer:MODE, HWSequencer:MODE?,

シンタックス： HWSequencer?



アーギュメント：

レスポンス： レスポンス・フォーマットは以下の通りです。

[:HWSEQUENCER:INSTALLED] <Installed State>;[MODE] <Mode State>

ここで、<Installed State>::={1|0}
1 — ハードウェア・シーケンサを搭載しています。
0 — ハードウェア・シーケンサを搭載していません。

ここで、<Mode State>::={1|0}
1 — ハードウェア・シーケンサを使用しています。
0 — ハードウェア・シーケンサを使用していません。

使用例： 以下は、HWSequencer? のレスポンス例です。

:HWSEQUENCER:INSTALLED 1;MODE 0

HWSequencer:INSTalled?

(AWG2041型のみ)

HWSequencer:INSTalled?コマンドはハードウェア・シーケンサを搭載しているかどうかを問い合わせます。

グループ： SYSTEM

関連コマンド： HWSequencer?, HWSequencer:MODE, HWSequencer:MODE?,

シンタックス： HWSequencer:INSTalled?

アーギュメント：

レスポンス： レスポンス・フォーマットは以下の通りです。

[:HWSEQUENCER:INSTALLED] <Installed State>

ここで、<Installed State>::={1|0}
1 — ハードウェア・シーケンサを搭載しています。
0 — ハードウェア・シーケンサを搭載していません。

使用例： 以下は、HWSequencer:INSTalled? のレスポンス例です。

:HWSEQUENCER:INSTALLED 1

HWSequencer:MODE (?)

(AWG2041型のみ)

HWSequencer:MODE コマンドは、ハードウェア・シーケンサの機能を使うかどうかを設定します。ハードウェア・シーケンサを搭載していない場合は何も影響しません。また、設定変更後は電源投入時と同じ初期化が行なわれます。

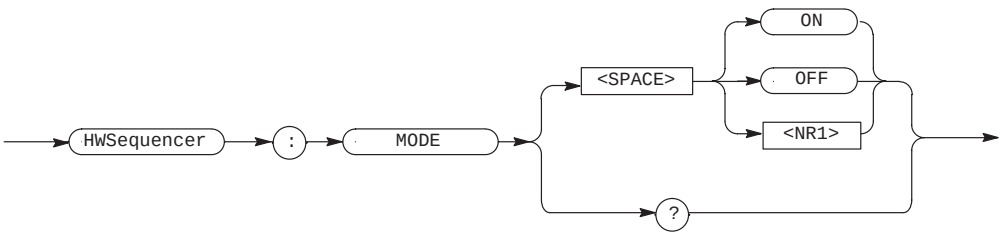
HWSequencer:MODE? 問い合わせコマンドは、現在ハードウェア・シーケンサを使う設定になっているかどうか問い合わせます。ハードウェア・シーケンサを搭載していない場合は 0 (ゼロ) を返します。

> **注：** ハードウェア・シーケンサ・モードを変更すると、本機器の内部メモリのデータが失われます。必要なデータはあらかじめディスクまたは NVRAM に保存しておいてください。

グループ： SYSTEM

関連コマンド： HWSequencer?, HWSequencer:INSTalled?

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



アーギュメント： ON またはゼロ以外の値 — ハードウェア・シーケンサを使用します。
OFF またはゼロ — ハードウェア・シーケンサは使用しません。

レスポンス： 問い合わせコマンドに対するレスポンス・フォーマットは以下の通りです。

[:HWSEQUENCER:MODE] <Mode State>

ここで、<Mode State>::={1|0}
1 — ハードウェア・シーケンサを使用しています。
0 — ハードウェア・シーケンサを使用していません。

使用例： 以下は、ハードウェア・シーケンサを使用する例です。

:HWSEQUENCER:MODE ON

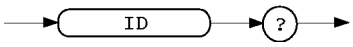
ID?

ID? 問い合わせコマンドは、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*IDN? 共通・問い合わせコマンドを使用できます。

グループ： SYSTEM

関連コマンド： *IDN?

シンタックス： ID?



アーギュメント：

レスポンス： レスポンス・フォーマットは以下の通りです。

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

詳細は、以下の通りです。

<Manufacturer>::=SONY_TEK
<Model>::=AWG2005 | AWG2010 | AWG2011 | AWG2020 | AWG2021 | AWG2040 | AWG2041
<Firmware Level>::=CF:<Code and Format Version>,
FV:<Firmware Version>

使用例： 以下は、AWG2020 型に対する:ID? のレスポンス例です。

ID SONY_TEK/AWG2020, CF:91.1CT, FV:1.00

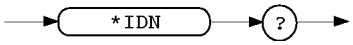
*IDN?

*IDN? 共通・問い合わせコマンドは、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? 問い合わせコマンドを使用できます。

グループ： SYSTEM

関連コマンド： ID?

シンタックス： *IDN?



アーギュメント：

レスポンス： レスポンス・フォーマットは以下の通りです。

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

詳細は、以下の通りです。

<Manufacturer>::=SONY/TEK
<Model>::=AWG2005 | AWG2010 | AWG2011 | AWG2020 | AWG2021 | AWG2040 | AWG2041
<Serial Number>::=0
<Firmware Level>::=CF:<Code and Format Version>
<sp>FV:<Firmware Version>
<sp>::= 空白

使用例： 以下は、AWG2020型に対する *IDN? のレスポンス例です。

SONY/TEK, AWG2020, 0, CF:91.1CT FV:1.00

LOCk (?)

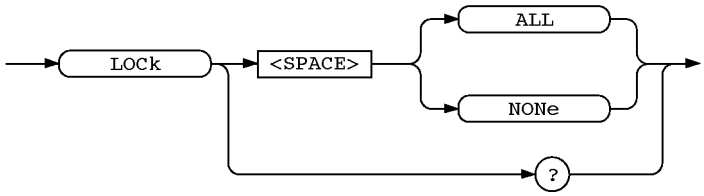
LOCk コマンドは、フロント・パネル上のキーやノブの機能をロックしたりロックを解除したりします。また LOCk? 問い合わせコマンドは、フロント・パネル上のキーやノブがロックされているかどうかを問い合わせます。

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なえます。外部コントローラで操中に、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、本コマンドで、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。

グループ： SYSTEM

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： 以下は、キーおよびノブをロックする例です。

```
:LOCK  ALL
```

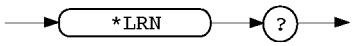
*LRN?

*LRN? 共通・問い合わせコマンドは、本機器の全ての設定を問い合わせます。設定値は、全ての関連するコマンドが連結された形式で得られますので (使用例参照)、このレスポンスをコマンドとして機器に送れば、このコマンドを実行した時の設定状態に機器を戻すことができます。

グループ： SYSTEM

関連コマンド：

シンタックス： *LRN?



アーギュメント：

レスポンス： 関連する全てのコマンドが連結された形式。使用例を参照ください。

使用例： 以下は、*LRN? のレスポンス例です。

:HEADER 1; :VERBOSE 1; :DIAG:SELECT ALL; :SELFCAL:SELECT ALL; :DISPLAY:BRIGHTNESS 70; CATALOG:ORDER NAME1; :DISPLAY:CLOCK 0; MENU:SETUP:FORMAT GRAPHICS; :DISPLAY:MESSAGE:SHOW””; :FG:STATE 0; FREQUENCY 10.00000E+06; CH1:AMPLITUDE 1.000; OFFSET 0.000; POLARITY NORMAL; SHAPE SINUSOID; :HCOPY:FORMAT BMP; PORT DISK; :DISK: FORMAT:TYPE HD3; TRIGGER:IMPEDANCE HIGH; LEVEL 1.4; POLARITY POSITIVE; SLOPE POSITIVE; :CH1:WAVEFORM””; FILTER THRU; AMPLITUDE 1.000; OFFSET 0.000; MARKERLEVEL1:HIGH 2.0; LOW 0.0; :CH1: MARKERLEVEL2:HIGH 2.0; LOW 0.0; :CLOCK:FREQUENCY 1.000000E+09; SOURCE INTERNAL; :OUTPUT:CH1:NORMAL:STATE 0; :OUTPUT:CH1: INVERTED:STATE 0; :DEBUG:SNOOP:STATE 0; DELAY:TIME 0.2; :DATA: DESTINATION ”GPIB.WFM”; ENCDG RPBINARY; SOURCE ”CH1”; WIDTH 2

MARKer:DATA (?)

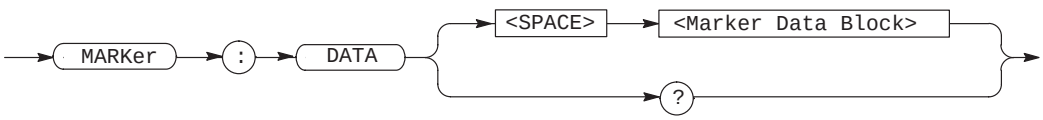
MARKer:DATA コマンドは、DATA:DESTination コマンドで指定するファイルにマーカ・データを書き込みます。

MARKer:DATA? 問い合わせコマンドは、DATA:SOURce コマンドで指定するマーカ・データを読みます。

グループ： WAVEFORM

関連コマンド： MARKER<x>:AOFF、MARKER<x>:POINt、DATA:DESTination、DATA:SOURce

シンタックス： MARKer:DATA <Marker Data Block>
MARKer:DATA?



アーギュメント： <Marker Data Block>::=<Arbitrary Block>　　マーカ・データ

<Marker Data Block> のフォーマットは、次の通りです。

#<x><yyy><marker(1)><marker(2)><marker(3)> . . . . <marker(n)>

ここで <yyy> は、後に続くマーカ・データのバイト数 (ASCII 形式) n を、<x> は、<yyy> の桁数 (ASCII 形式) を表します。マーカ・データ <marker(i)> は、1 バイトで構成され、下位 2 ビットのみが有効ビットとして次の表のようなバイナリの値を取ります。なお上位 6 ビットには 0 を設定します。

上記でマーカ 2 は AWG2010/11/20/21/40/41型のみです。

本コマンドは、マーカ・データを一括して設定します。部分的にマーカ・データを設定しなおす場合には、MARKER<x>:POINt コマンドを使用してください。

使用例： 以下は、マーカ・データを設定する例です。

:MARKER:DATA #41000 ...

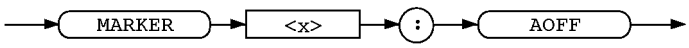
MARKER<x>:AOFF

MARKER<x>:AOFF コマンドは、DATA:DESTination コマンドで指定されるファイルの指定マーカ側のデータをリセットします。

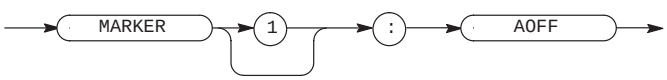
グループ： WAVEFORM

関連コマンド： MARKER<x>:POINt、MARKer:DATA、DATA:DESTination

シンタックス： AWG2010/AWG2011/AWG2020/AWG2021/AWG2040/AWG2041型：
MARKER<x>:AOFF



AWG2005型：
MARKER [1]:AOFF



アーギュメント：

使用例： 以下は、ファイル WAVE01.WFM のマーカ 1 のデータをリセットする例です。

:DATA:DESTINATION ”WAVE01.WFM”; :MARKER1:AOFF

MARKER<x>:POINt (?)

MARKER<x>:POINt コマンドは、DATA:DESTination コマンドで指定されるファイルの指定マーカ側のデータのうち、指定位置のマーカ・データを設定し直します。

MARKER<x>:POINt? 問い合わせコマンドは、DATA:SOURce コマンドで指定されるファイルの指定マーカ側のデータのうち、指定位置のマーカ・データを問い合わせます。

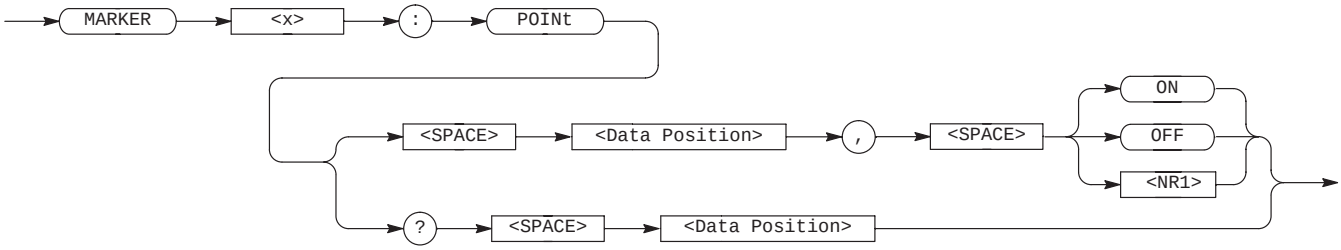
グループ： WAVEFORM

関連コマンド： MARKER<x>:AOFF、MARKer:DATA、DATA:DESTination、DATA:SOURce

シンタックス： AWG2010/AWG2011/AWG2020/AWG2021/AWG2040/AWG2041型:

MARKER<x>:POINt <Data Position>,{ON | OFF | <NR1>}

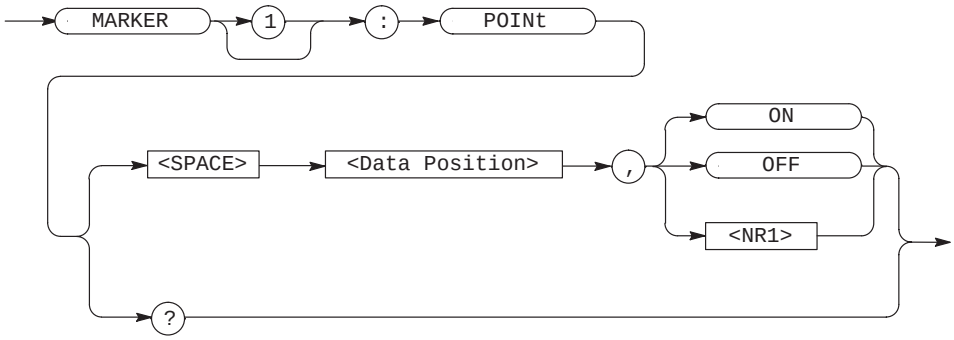
MARKER<x>:POINt? <Data Position>



AWG2005型：

MARKER[1]:POINt <Data Position>, {ON|OFF|<NR1>}

MARKER[1]:POINt?



アーギュメント： <Data Position>::=<NR1> データ位置

ON または 0 以外の値 — マーカをセットします。

OFF または 0 — マーカをリセットします。

なお、マーカ・データを一括して設定したい場合には、MARKer:DATA コマンドを使用してください。

レスポンス： 次のフォーマットで返送されます。

AWG2010/AWG2011/AWG2020/AWG2021/AWG2040/AWG2041型：
[:MARKER {1|2} POINT] <Data Position>, {0|1}

AWG2005型：
[:MARKER [1]:POINT] <Data Point>, {0|1}

使用例： 以下は、ファイル WAVE01.WFM のマーカ 1 側の 2001 番目のデータを設定し直す例です。

:DATA:DESTINATION ”WAVE01.WFM”; :MARKER1:POINT 2001, ON

以下は、:DATA:SOURCE ”WAVE02.WFM”; :MARKER1:POINT? 1400 に対するレスポンス例です。

:MARKER1:POINT 1400, 1

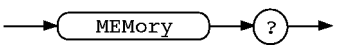
MEMory?

MEMory? 問い合わせコマンドは、内部メモリにある全てのファイルのファイル情報と内部メモリの使用状況を問い合わせます。本問い合わせコマンドは、MEMory:CATalog:ALL? と MEMory:FREE:ALL? 問い合わせコマンドの機能に相当します。

グループ： MEMORY

関連コマンド： MEMory:CATalog:ALL?、MEMory:FREE:ALL?

シンタックス： MEMory?



アーギュメント：

レスポンス： レスポンス・フォーマットは、以下の通りです。

[:MEMORY:CATALOG:ALL] <File Entry>[,<File Entry>]...
;[MEMORY:FREE:ALL] <Unused Size>,<Used Size>

ここで

<File Entry>::=<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>
<Unused Size>::=<NR1>　　未使用領域バイト数
<Used Size>::=<NR1>　　使用済領域バイト数

使用例： :MEMORY? のレスポンス例です。

:MEMORY:CATALOG:ALL ”AUTOSTEP.AST”, 142, ”93–11–11 12:12”, ”EQUATION.EQU”, 296, ”93–11–11 12:12”, ”SEQUENCE.SEQ”, 960, ”93–11–11 12:12”, ”WAVE2.WFM”, 2948, ”93–11–11 12:12”, ”WAVEFORM.WFM”, 2948, ”93–11–11 12:12”;
:MEMORY:FREE:ALL 1696220, 28500

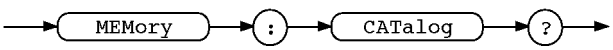
MEMory:CATalog?

MEMory:CATalog? 問い合わせコマンドは、内部メモリにある全てのファイルのファイル情報を問い合わせます。本問い合わせコマンドは、MEMory:CATalog:ALL? 問い合わせコマンドと全く同じ機能です。

グループ： MEMORY

関連コマンド： MEMory:CATalog:ALL?、MEMory?

シンタックス： MEMory:CATalog?



アーギュメント：

レスポンス： レスポンス・フォーマットは以下の通りです。

[:MEMORY:CATALOG:ALL] <File Entry>[,<File Entry>]...

ここで

<File Entry>::=<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>

使用例：　以下は、:MEMORY:CATALOG? のレスポンス例です。

:MEMORY:CATALOG:ALL ”AUTOSTEP.AST”, 142, ”93–11–11 12:12”, ”EQUATION.EQU”, 296, ”93–11–11 12:12”, ”SEQUENCE.SEQ”, 960, ”93–11–11 12:12”, ”WAVE2.WFM”, 2948, ”93–11–11 12:12”, ”WAVEFORM.WFM”, 2948, ”93–11–11 12:12”

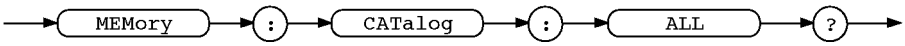
MEMory:CATalog:ALL?

MEMory:CATalog:ALL? 問い合わせコマンドは、内部メモリにある全てのファイルのファイル情報を問い合わせます。

グループ：　MEMORY

関連コマンド：　MEMory:CATalog?、MEMory:CATalog:AST?、MEMory:CATalog:CLK?、MEMory:CATalog:EQU?、MEMory:CATalog:SEQ?、MEMory:CATalog:WFM?、MEMory?

シンタックス：　MEMory:CATalog:ALL?



アーギュメント：

レスポンス：　レスポンス・フォーマットは、以下の通りです。

[:MEMORY:CATALOG:ALL] <File Entry>[,<File Entry>]...

ここで

<File Entry>::=<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>

使用例：　以下は、:MEMORY:CATALOG:ALL? のレスポンス例です。

:MEMORY:CATALOG:ALL ”AUTOSTEP.AST”, 142, ”93–11–11 12:12”, ”EQUATION.EQU”, 296, ”93–11–11 12:12”, ”SEQUENCE.SEQ”, 960, ”93–11–11 12:12”, ”WAVE2.WFM”, 2948, ”93–11–11 12:12”, ”WAVEFORM.WFM”, 2948, ”93–11–11 12:12”

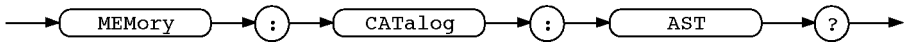
MEMory:CATalog:AST?

MEMory:CATalog:AST? 問い合わせコマンドは、内部メモリにある全てのオート・ステップ・ファイルのファイル情報を問い合わせます。

グループ： MEMORY

関連コマンド： MEMory:CATalog:ALL?、MEMory?

シンタックス： MEMory:CATalog:AST?



アーギュメント：

レスポンス： レスポンス・フォーマットは、以下の通りです。

[:MEMORY:CATALOG:AST] <File Entry>[,<File Entry>]...

ここで

<File Entry>::=<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>

使用例： 以下は、:MEMORY:CATALOG:AST? のレスポンス例です。

:MEMORY:CATALOG:AST "AUTOSTEP.AST", 142, "93–11–11 12:12"

MEMory:CATalog:CLK?

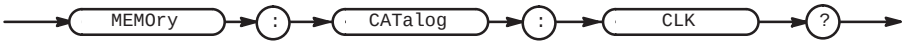
(AWG2005型のみ)

MEMory:CATalog:CLK? 問い合わせコマンドは、内部メモリにある全てのクロック・スイープ・ファイルのファイル情報を問い合わせます。

グループ： MEMORY

関連コマンド： MEMory:CATalog:ALL?、MEMory?

シンタックス： MEMory:CATalog:CLK?



アーギュメント：

レスポンス：　レスポンス・フォーマットは以下の通りです。

[:MEMORY:CATALOG:CLK] <File Entry> [,<File Entry>]...

ここで

<File Entry> ::=<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>

使用例：　以下は、:MEMORY:CATALOG:CLK? のレスポンス例です。

:MEMORY:CATALOG:CLK　”CLKSWEEP.CLK”, 10876, ”93–11–11 12:12”

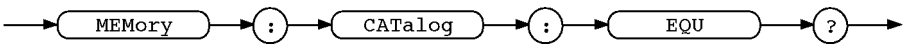
MEMory:CATalog:EQU?

MEMory:CATalog:EQU? 問い合わせコマンドは、内部メモリにある全てのイクエーション・ファイルのファイル情報を問い合わせます。

グループ：　MEMORY

関連コマンド：　MEMory:CATalog:ALL?、MEMory?

シンタックス：　MEMory:CATalog:EQU?



アーギュメント：

レスポンス：　レスポンス・フォーマットは、以下の通りです。

[:MEMORY:CATALOG:EQU] <File Entry>[,<File Entry>]...

ここで

<File Entry>::=<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>

使用例：　以下は、:MEMORY:CATALOG:EQU? のレスポンス例です。

:MEMORY:CATALOG:EQU ”EQUATION.EQU”, 296, ”93–11–11 12:12”

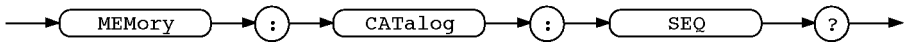
MEMory:CATalog:SEQ?

MEMory:CATalog:SEQ? 問い合わせコマンドは、内部メモリにある全てのシーケンス・ファイルのファイル情報を問い合わせます。

グループ： MEMORY

関連コマンド： MEMory:CATalog:ALL?、MEMory?

シンタックス： MEMory:CATalog:SEQ?



アーギュメント：

レスポンス： レスポンス・フォーマットは、以下の通りです。

[:MEMORY:CATALOG:SEQ] <File Entry>[,<File Entry>]...

ここで

<File Entry>::=<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>

使用例： 以下は、:MEMORY:CATALOG:SEQ? のレスポンス例です。

:MEMORY:CATALOG:SEQ "SEQUENCE.SEQ", 960, "93–11–11 12:12"

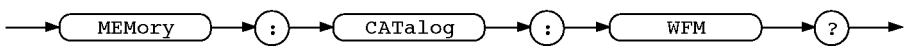
MEMory:CATalog:WFM?

MEMory:CATalog:WFM? 問い合わせコマンドは、内部メモリにある全ての波形ファイルのファイル情報を問い合わせます。

グループ： MEMORY

関連コマンド： MEMory:CATalog:ALL?、MEMory?

シンタックス： MEMory:CATalog:WFM?



アーギュメント：

レスポンス：　レスポンス・フォーマットは、以下の通りです。

[:MEMORY:CATALOG:WFM] <File Entry>[,<File Entry>]...

ここで

<File Entry>::=<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>

使用例：　以下は、:MEMORY:CATALOG:WFM? のレスポンス例です。

:MEMORY:CATALOG:WFM ”WAVE2.WFM”, 2948, ”93–11–11 12:12”, ”WAVEFORM.WFM”, 2948, ”93–11–11 12:12”

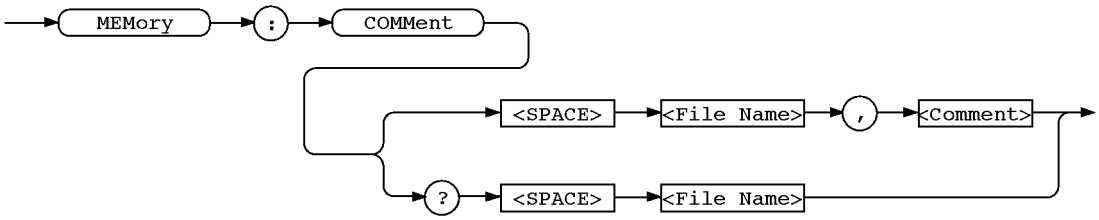
MEMory:COMMent　(?)

MEMory:COMMent コマンドは、指定するファイルのコメント欄に、コメント情報を書き込みます。また MEMory:COMMent? 問い合わせコマンドは、指定するファイルのコメント欄に書き込まれたコメント情報を問い合わせます。なお指定するファイルは、内部メモリにあるファイルに限られます。

グループ：　MEMORY

関連コマンド：　MEMory:COPY、MEMory:DELete、MEMory:REName、MEMory:LOCk

シンタックス：　MEMory:COMMent　<File Name>,<Comment>
MEMory:COMMent?　<File Name>



アーギュメント：　<File Name>::=<文字列> コメントを書き込むファイル。ファイルは、内部メモリになければなりません。
<Comment> ::=<文字列> 最大 24 文字までのコメント

注：　MEMory:LOCk コマンドでファイルがロックされている場合は、そのファイルにコメント情報を書き込むことはできません。

使用例： 以下は、ファイル TDS_REF.WFM にコメントを書き込む例です。

:MEMORY:COMMENT ”TDS_REF.WFM”, ”COPIED FROM TDS REF”

MEMory:COPY

MEMory:COPY コマンドは、内部メモリ上でファイルをコピーします。

グループ： MEMORY

関連コマンド： MEMory:DELete、MEMory:REName、MEMory:COMMent、MEMory:LOCk

シンタックス： MEMory:COPY <From-file>, <To-file>

アーギュメント： <From-file>::=<文字列> コピー元ファイル名
<To-file>::=<文字列> コピー先ファイル名

<To-file> が既に存在する場合には、内容が上書きされます。ただし、<To-file> が MEMory:LOCk コマンドでロックされていればコピーはできません。ファイルがない場合には、新しく作られます。

<From-file> と <To-file> は、ファイル拡張子が同じでなければなりません。違う拡張子を指定した場合には、エラーとなります。

使用例： 以下は、波形ファイルをコピーする例です。

:MEMORY:COPY ”TDS_REF.WFM”, ”AWGCH1.WFM”

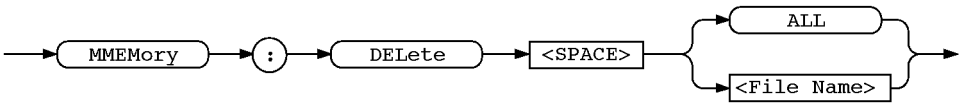
MEMory:DELete

MEMory:DELete コマンドは、内部メモリにあるファイルを削除します。ただし、MEMory:LOCk コマンドでロックされたファイルを削除することはできません。

グループ： MEMORY

関連コマンド： MEMory:COPY、MEMory:REName、MEMory:COMMent、MEMory:LOCk

シンタックス： MEMory:DELete {<File Name>|ALL}



アーギュメント： <File Name>::=<文字列>　　削除ファイル名
ALL　　—　内部メモリにある全てのファイルを削除

使用例： 以下は、波形ファイル AWGCH2.WFM を削除する例です。

```
:MEMORY:DELETE ”AWGCH2.WFM”
```

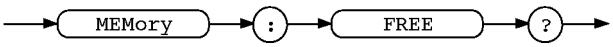
MEMory:FREE?

MEMory:FREE? 問い合わせコマンドは、内部メモリの使用状況 (used size と unused size) を問い合わせます。本コマンドは、MEMory:FREE:ALL? 問い合わせコマンドと全く同じです。

グループ： MEMORY

関連コマンド： MEMory:FREE:ALL?、MEMory?

シンタックス： MEMory:FREE?



アーギュメント：

レスポンス： レスポンス・フォーマットは、以下の通りです。

```
[:MEMORY:FREE:ALL]  <Unused Size>,<Used Size>
```

ここで

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

使用例： 以下は、:MEMORY:FREE? のレスポンス例です。

```
:MEMORY:FREE  1696220, 28500
```

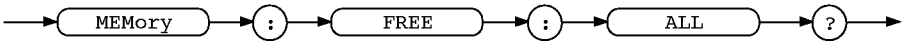
MEMory:FREE:ALL?

MEMory:FREE:ALL? 問い合わせコマンドは、内部メモリの使用状況 (used size と unused size) を問い合わせます。本コマンドは、MEMory:FREE? 問い合わせコマンドと全く同じです。

グループ： MEMORY

関連コマンド： MEMory:FREE?、MEMory?

シンタックス： MEMory:FREE:ALL?



アーギュメント：

レスポンス： レスポンス・フォーマットは、以下の通りです。

[:MEMORY:FREE:ALL] <Unused Size>,<Used Size>

ここで

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

使用例： 以下は、:MEMORY:FREE:ALL? のレスポンス例です。

:MEMORY:FREE:ALL 1696220, 28500

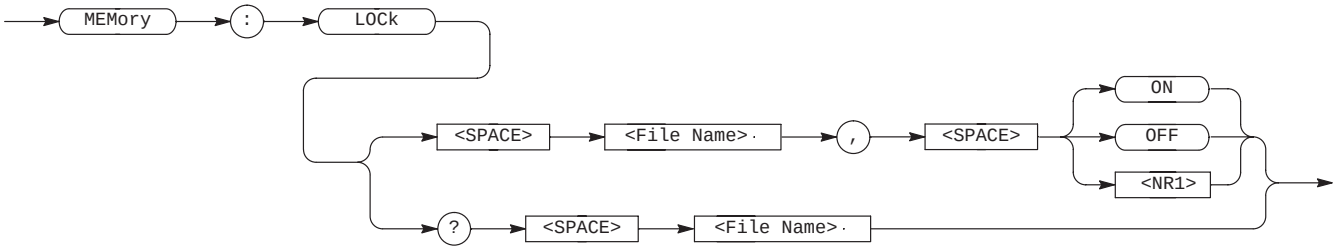
MEMory:LOCk (?)

MEMory:LOCk マンドは、内部メモリの指定する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、削除や書き込みなどができなくなります。また MEMory:LOCk? 問い合わせコマンドは、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： MEMory:DELete、MEMory:COPY、MEMory:RENname、MEMory:COMMent

シンタックス： MEMory:LOCk <File Name>,{ON | OFF | <NR1>}
MEMory:LOCk? <File Name>



アーギュメント： <File Name>::=<文字列> ロックまたはアンロックするファイルのファイル名
ON または 0 以外の値　— 指定ファイルをロックします。
OFF または 0　— 指定ファイルのロックを解除します。

注： ファイルがロックされた場合には、次の処理ができなくなります。

- □ ファイルの削除
- □ コピーやロードによる上書き
- □ コメント情報の書き込み
- □ ファイル名の変更

レスポンス： レスポンスは、以下の通りです。

1 — ファイルがロックされています。
0 — ファイルがアンロックされています。

使用例： 以下は、ファイル RAMP_W1.WFM をロックする例です。

```
:MEMORY:LOCK ”RAMP_W1.WFM”, 1
```

MEMory:REName

MEMory:REName コマンドは、内部メモリにある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： MEMory:COPY、MEMory:DELete、MEMory:COMMent、MEMory:LOCk

シンタックス： MEMory:REName　<From-filename>, <To-filename>

アーギュメント： <From-filename>::=<文字列 > 変更前のファイル名
<To-filename>::=<文字列 > 変更後のファイル名

<From-filename> と <To-filename> の拡張子は、同じでなければなりません。異なった拡張子を指定した場合には、エラーになります。

注： MEMory:LOCk コマンドでファイルがロックされている場合には、ファイル名の変更を行なうことはできません。

使用例： 以下は、ファイル TDS_REF.WFM を AWGCH2.WFM に名前を変更する例です。

```
:MEMORY:RENAME  ”TDS_REF.WFM”, ”AWGCH2.WFM”
```

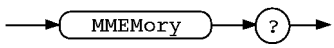
MMEMory?

MMEMory? 問い合わせコマンドは、カレント・マス・メモリにある全てのファイルのファイル情報、カレント・マス・メモリの使用状況およびオート・ロードの設定状況を問い合わせます。本問い合わせコマンドは、MMEMory:ALOad?、MMEMory:MISIS?、MMEMory:CATalog:ALL? と MMEMory:FREE:ALL? の各問い合わせコマンドを合わせた機能に相当します。

グループ： MEMORY

関連コマンド： MMEMory:ALOad?、MMEMory:MSIS?、MMEMory:CATalog:ALL?、MMEMory:FREE:ALL?

シンタックス： MMEMory?



アーギュメント：

レスポンス：　レスポンス・フォーマットは、以下の通りです。

[:MMEMORY:MSIS] <Current Mass Memory>;[CATALOG:ALL] <File Entry>[,<File Entry>]... ;[:MMEMORY:ALOAD:MSIS] <Auto Load Mass Memory>;[STATE] <Auto Load State> ;[:MMEMORY:FREE:ALL] <Unused Size>, <Used Size>

ここで

<Current Mass Memory>::={DISK | NVRAM}
<File Entry>::=<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>
<Auto Load Mass Memory>::={DISK | NVRAM}
<Auto Load State>::={0 | 1}
<Unused Size>::=<NR1>　　未使用領域バイト数
<Used Size>::=<NR1>　　使用済領域バイト数

拡張子に .BMP、.EPS、.EQA、.ESC、.ISF、.TIF、.TJ、.WFB、.WVNを持つフロッピ・ディスク上のファイルは、本問い合わせコマンドまたは MMEMory:CATalog? 問い合わせコマンド、MMEMory:CATalog:ALL? 問い合わせコマンドのみで参照できます。

使用例：　以下は、:MMEMORY? のレスポンス例です。

:MMEMORY:MSIS DISK;CATALOG:ALL　”AUTOSTEP.AST”, 142, ”93–11–11 12:12”, ”EQUATION.EQU”, 296, ”93–11–11 12:12”, ”SEQUENCE.SEQ”, 960, ”93–11–11 12:12”, ”WAVE2.WFM”, 2948, ”93–11–11 12:12”, ”WAVEFORM.WFM”, 2948, ”93–11–11 12:12; :MMEMORY:ALOAD:MSIS DISK;STATE 0; :MMEMORY:FREE:ALL　801792, 672760

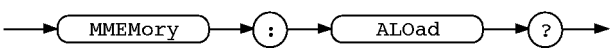
MMEMory:ALOad?

MMEMory:ALOad? 問い合わせコマンドは、電源投入時に実行するオート・ロードの設定状況を問い合わせます。

グループ：　MEMORY

関連コマンド：　MMEMory:ALOad:MSIS、MMEMory:ALOad:STATe

シンタックス：　MMEMory:ALOad?



アーギュメント：

レスポンス： レスポンス・フォーマットは、以下の通りです。

[:MMEMORY:ALOAD:MSIS]<Auto Load Mass Memory>;[STATE] <Auto Load State>

ここで

<Auto Load Mass Memory>::={DISK | NVRAM}　　オート・ロード時のロード元マス・メモリ

<Auto Load State>::={1 | 0}

1　—　電源投入時にオート・ロードが実行されます。

0　—　オート・ロードは実行されません。

使用例： 以下は、:MMEMORY:ALOAD? のレスポンス例です。

:MMEMORY:ALOAD:MSIS DISK; STATE 0

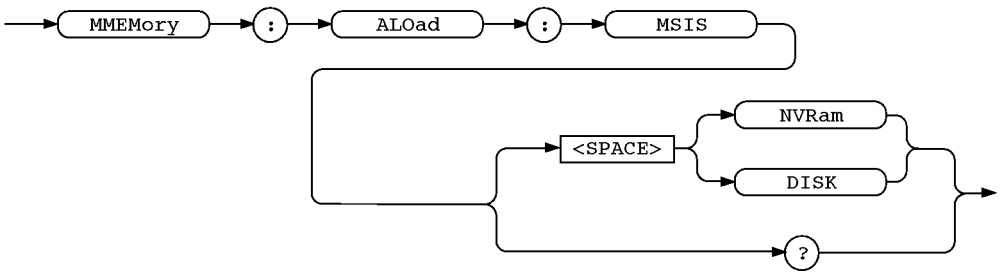
MMEMory:ALOad:MSIS　(?)

MMEMory:ALOad:MSIS コマンドは、オート・ロード時のマス・メモリとして、NVRAMまたは DISK を選択します。オート・ロード実行時には、この指定したマス・メモリにあるファイルを内部メモリにロードします。また MMEMory:ALOad:MSIS? 問い合わせコマンドは、設定中のオート・ロード用のマス・メモリを問い合わせます。

グループ： MEMORY

関連コマンド： MMEMory:ALOad:STATe、MMEMory:ALOad?

シンタックス： MMEMory:ALOad:MSIS　{NVRam | DISK}
MMEMory:ALOad:MSIS?



アーギュメント： NVRam　—　Non Volatile RAM を選択します。
DISK　—　フロッピー・ディスクを選択します。

マス・メモリとして DISKを選択した場合、オート・ロード時、“\AWG2005” (AWG2005型)、“\AWG2010” (AWG2010型)、“\AWG2011” (AWG2011型)、“\AWG2020” (AWG2020型)、“\AWG2021” (AWG2021型)、“\AWG2040” (AWG2040型)、“\AWG2041” (AWG2041型) のディレクトリの下にある全てのファイルがロードされます。

注：　フロッピー・ディスクから内部メモリにファイルがロードされる際、拡張子 .ISF、.WFB、.WVN、.EQAを持つファイルは、それぞれ .WFM、.WFM、.WFM、.EQU の拡張子を持つファイル名に変更されることに注意してください。

使用例：　以下は、マス・メモリとして DISK を選択する例です。

```
:MMEMORY:ALOAD:MSIS  DISK
```

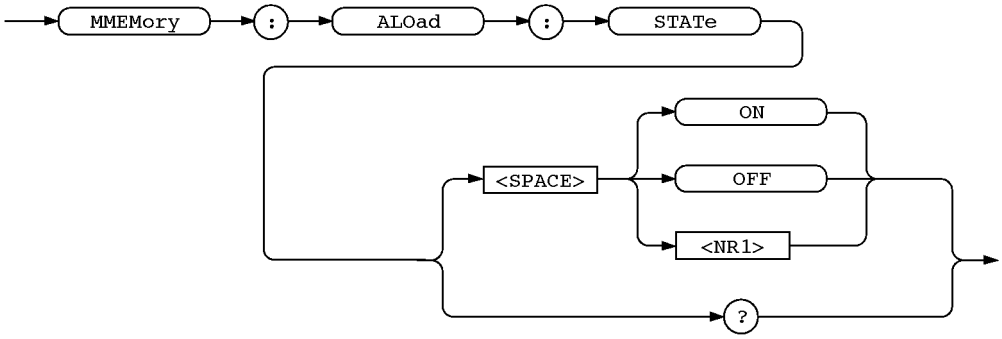
MMEMory:ALOad:STATe　(?)

MMEMory:ALOad:STATe コマンドは、電源投入時に、オート・ロードを実行するかどうかを定義します。また MMEMory:ALOad:STATe? 問い合わせコマンドは、電源投入時に、オート・ロードを実行するかどうかを問い合わせます。

グループ：　MEMORY

関連コマンド：　MMEMory:ALOad:MSIS、MMEMory:ALOad?

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



アーギュメント：　ON または 0 以外の値 ― 電源投入時にオート・ロードを実行するように設定します。
OFF または 0　　― オート・ロードを実行しないように設定します。

レスポンス：　問い合わせコマンドに対するレスポンスは、以下の通りです。

1　― 電源投入時にオート・ロードが実行されます。
0　― オート・ロードは実行されません。

使用例：　以下は、電源投入時にオート・ロードが実行されるように設定する例です。

```
:MMEMORY:ALOAD:STATE  1
```

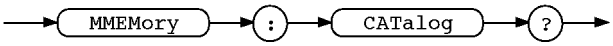
MMEMory:CATalog?

MMEMory:CATalog? コマンドは、カレント・マス・メモリにある全てのファイルのファイル情報を問い合わせます。本問い合わせコマンドは、MMEMory:CATalog:ALL? 問い合わせコマンドと全く同等です。

グループ： MEMORY

関連コマンド： MMEMory:MSIS、MMEMory:CATalog:ALL?、MMEMory?

シンタックス： MMEMory:CATalog?



アーギュメント：

レスポンス： レスポンス・フォーマットは、以下の通りです。

[:MMEMORY:CATALOG:ALL] <File Entry>[,<File Entry>]...

ここで

<File Entry>::=<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>

拡張子に .BMP、.EPS、.EQA、.ESC、.ISF、.TIF、.TJ、.WFB、.WVN を持つフロッピ・ディスク上のファイルは、本問い合わせコマンドまたは MMEMory? 問い合わせコマンド、MMEMory:CATalog:ALL? 問い合わせコマンドのみで参照できます。

使用例： 以下は、:MMEMORY:CATALOG? のレスポンス例です。

:MMEMORY:CATALOG:ALL ”AUTOSTEP.AST”, 142, ”93–11–11 12:12”,
”EQUATION.EQU”, 296, ”93–11–11 12:12”, ”SEQUENCE.SEQ”, 960,
”93–11–11 12:12”, ”WAVE2.WFM”, 2948, ”93–11–11 12:12”, ”WAVEFORM.WFM”,
2948, ”93–11–11 12:12”, ”TDSWFM.ISF”, 2948, ”93–11–11 12:12”, ”DSAWFM.WFB”,
2948, ”93–11–11 12:12”, ”PCEQU.EQA”, 296, ”93–11–11 12:12”

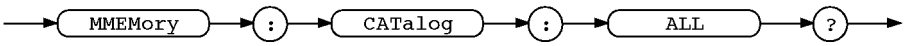
MMEMory:CATalog:ALL?

MMEMory:CATalog:ALL? 問い合わせコマンドは、カレント・マス・メモリにある全てのファイルのファイル情報を問い合わせます。本問い合わせコマンドは、MMEMory:CATalog? 問い合わせコマンドと全く同等です。

グループ： MEMORY

関連コマンド： MMEMory:MSIS、MMEMory:CATalog?、MMEMory:CATalog:AST?、MMEMory:CATalog:CLK?、MMEMory:CATalog:EQU?、MMEMory:CATalog:SEQ?、MMEMory:CATalog:WFM?、MMEMory?

シンタックス： MMEMory:CATalog:ALL?



アーギュメント：

レスポンス： レスポンス・フォーマットは、以下の通りです。

[:MMEMORY:CATALOG:ALL] <File Entry>[,<File Entry>]...

ここで

<File Entry>::=<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>

拡張子に .BMP、.EPS、.EQA、.ESC、.ISF、.TIF、.TJ、.WFB、.WVN を持つフロッピ・ディスク上のファイルは、本問い合わせコマンドまたは MMEMory? 問い合わせコマンド、MMEMory:CATalog? 問い合わせコマンドのみで参照できます。

使用例： 以下は、:MMEMORY:CATALOG:ALL? のレスポンス例です。

:MMEMORY:CATALOG:ALL ”AUTOSTEP.AST”, 142, ”93–11–11 12:12”,
”EQUATION.EQU”, 296, ”93–11–11 12:12”, ”SEQUENCE.SEQ”, 960,
”93–11–11 12:12”, ”WAVE2.WFM”, 2948, ”93–11–11 12:12”, ”WAVEFORM.WFM”,
2948, ”93–11–11 12:12”, ”TDSWFM.ISF”, 2948, ”93–11–11 12:12”, ”DSAWFM.WFB”,
2948, ”93–11–11 12:12”, ”PCEQU.EQA”, 296, ”93–11–11 12:12”

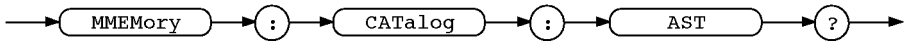
MMEMory:CATalog:AST?

MMEMory:CATalog:AST? 問い合わせコマンドは、カレント・マス・メモリにある全てのオート・ステップ・ファイルのファイル情報を問い合わせます。

グループ： MEMORY

関連コマンド： MMEMory:MSIS、MMEMory:CATalog:ALL?、MMEMory?

シンタックス： MMEMory:CATalog:AST?



アーギュメント：

レスポンス： レスポンス・フォーマットは、以下の通りです。

[:MMEMORY:CATALOG:AST] <File Entry>[,<File Entry>]...

ここで

<File Entry>::=<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>

使用例： 以下は、:MMEMORY:CATALOG:AST? のレスポンス例です。

:MMEORY:CATALOG:AST ”AUTOSTEP.AST”, 142, ”93–11–11 12:12”

MMEMory:CATalog:CLK?

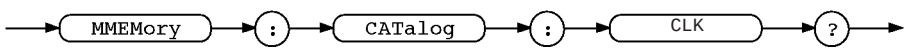
(AWG2005型のみ)

MMEMory:CATalog:CLK? 問い合わせコマンドは、カレント・マス・メモリにある全てのクロック・スイープ・ファイルのファイル情報を問い合わせます。

グループ： MEMORY

関連コマンド： MMEMory:MSIS、MMEMory:CATalog:ALL?、MMEMory?

シンタックス： MMEMory:CATalog:CLK?



アーギュメント：

レスポンス： レスポンス・フォーマットは以下の通りす。

[:MMEMORY:CATALOG:CLK] <File Entry> [,<File Entry>]...

ここで

<File Entry> ::=<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>

使用例： 以下は、:MMEMORY:CATALOG:CLK? のレスポンス例です。

:MMEMORY:CATALOG:CLK ”CLKSWEEP.CLK”, 10876, ”93–09–28 12:53”

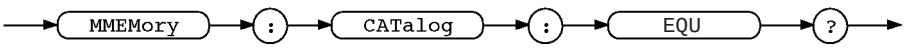
MMEMory:CATalog:EQU?

MMEMory:CATalog:EQU? 問い合わせコマンドは、カレント・マス・メモリにある全てのイクエーション・ファイルのファイル情報を問い合わせます。

グループ： MEMORY

関連コマンド： MMEMory:MSIS、MMEMory:CATalog:ALL?、MMEMory?

シンタックス： MMEMory:CATalog:EQU?



アーギュメント：

レスポンス： レスポンス・フォーマットは、以下の通りです。

[:MMEMORY:CATALOG:EQU] <File Entry>[,<File Entry>]...

ここで

<File Entry>::=<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>

使用例： 以下は、:MMEMORY:CATALOG:EQU? のレスポンス例です。

:MMEMORY:CATALOG:EQU ”EQUATION.EQU”, 296, ”93–11–11 12:12”

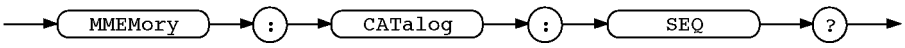
MMEMory:CATalog:SEQ?

MMEMory:CATalog:SEQ? 問い合わせコマンドは、カレント・マス・メモリにある全てのオート・ステップ・ファイルのファイル情報を問い合わせます。

グループ： MEMORY

関連コマンド： MMEMory:MSIS、MMEMory:CATalog:ALL?、MMEMory?

シンタックス： MMEMory:CATalog:SEQ?



アーギュメント：

レスポンス： レスポンス・フォーマットは、以下の通りです。

[:MMEMORY:CATALOG:SEQ] <File Entry>[,<File Entry>]...

ここで

<File Entry>::=<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>

使用例： 以下は、:MMEMORY:CATALOG:SEQ? のレスポンス例です。

:MMEMORY:CATALOG:SEQ ”SEQUENCE.SEQ”, 960, ”93–11–11 12:12”

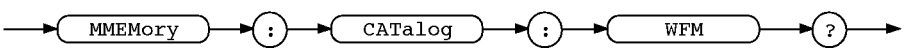
MMEMory:CATalog:WFM?

MMEMory:CATalog:WFM? 問い合わせコマンドは、カレント・マス・メモリにある全ての波形ファイルのファイル情報を問い合わせます。

グループ： MEMORY

関連コマンド： MMEMory:MSIS、MMEMory:CATalog:ALL?、MMEMory?

シンタックス： MMEMory:CATalog:WFM?



アーギュメント：

レスポンス：　レスポンス・フォーマットは、以下の通りです。

[:MMEMORY:CATALOG:WFM] <File Entry> [,<File Entry>]...

ここで

<File Entry>::=<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>

使用例：　以下は、:MMEMORY:CATALOG:WFM? のレスポンス例です。

:MMEMORY:CATALOG:WFM ”WAVE2.WFM”, 2948, ”93–11–11 12:12”, ”WAVEFORM.WFM”, 2948, ”93–11–11 12:12”

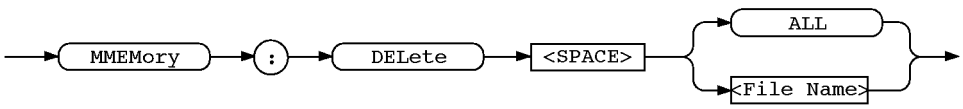
MMEMory:DELete

MMEMory:DELete コマンドは、カレント・マス・メモリの指定ファイルを削除します。ただし、MMEMory:LOCk コマンドでファイルがロックされている場合には、削除することはできません。

グループ：　MEMORY

関連コマンド：　MMEMory:REName、MMEMory:MSIS、MMEMory:LOCk

シンタックス：　MMEMory:DELete　{<File Name> | ALL}



アーギュメント：　<File Name>::=<文字列>　削除すべきファイル。
ALL　　—　カレント・マス・メモリにある全てのファイルを削除します。

使用例：　以下は、ファイル AWG2.WFM を削除する例です。

:MMEMORY:DELETE ”AWG2.WFM”

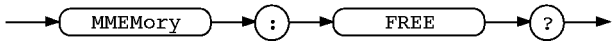
MMEMory:FREE?

MMEMory:FREE? 問い合わせコマンドは、カレント・マス・メモリの使用状況 (used size と unused size) を問い合わせます。本問い合わせコマンドは、MMEMory:FREE:ALL? 問い合わせコマンドと全く同等です。

グループ： MEMORY

関連コマンド： MMEMory:MSIS、MMEMory:FREE:ALL?、MMEMory?

シンタックス： MMEMory:FREE?



アーギュメント：

レスポンス： レスポンス・フォーマットは、以下の通りです。

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

ここで

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

使用例： 以下は、:MMEMORY:FREE? のレスポンス例です。

:MMEMORY:FREE:ALL 801792, 672760

MMEMory:FREE:ALL?

MMEMory:FREE:ALL? 問い合わせコマンドは、カレント・マス・メモリの使用状況 (used size と unused size) を問い合わせます。

グループ： MEMORY

関連コマンド： MMEMory:MSIS、MMEMory:FREE?、MMEMory?

シンタックス： MMEMory:FREE:ALL?

アーギュメント：

レスポンス： レスポンス・フォーマットは、以下の通りです。

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

ここで

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

使用例： 以下は、:MMEMORY:FREE:ALL? のレスポンス例です。

:MMEMORY:FREE:ALL 1696220, 28500

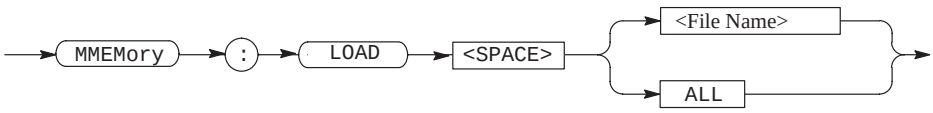
MMEMory:LOAD

MMEMory:LOAD コマンドは、カレント・マス・メモリにあるファイルを内部メモリにロードします。

グループ： MEMORY

関連コマンド： MMEMory:MSIS、MMEMory:SAVE、MMEMory:LOCk

シンタックス： MMEMory:LOAD　{<File Name> | ALL}



アーギュメント： <File Name>::=<文字列>　　ロードするファイル
ALL　　—　全てのファイルをロードします。

ロードするファイルが既に内部メモリに存在すれば、ファイルは上書きされます。ただし既存ファイルがロックされていれば上書きされることはありません。存在しなければ、新規に作成されます。

フロッピー・ディスクから内部メモリにファイルがロードされる際、拡張子 .ISF、.WFB、.WVN、.EQA を持つファイルは、データ型式が変換されるとともに、それぞれ .WFM、.WFM、.WFM、.EQU の拡張子を持つファイル名に変更されます。なお、データ型式の変換、ファイル名の変更によってカレント・マス・メモリのファイルが影響受けることはありません。

使用例： 以下は、カレント・マス・メモリの全てのファイルを内部メモリにロードする例です。

:MMEMORY:LOAD ALL

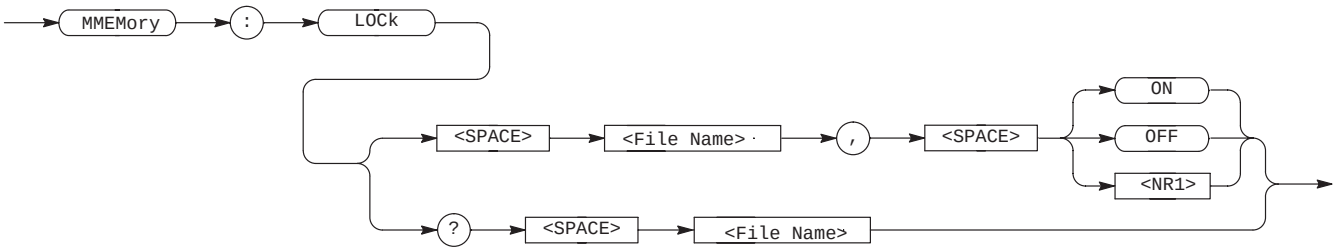
MMEMory:LOCk (?)

MMEMory:LOCk コマンドは、カレント・マス・メモリの指定する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? 問い合わせコマンドは、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： MMEMory:DELete、MMEMory:LOAD、MMEMory:REName、MMEMory:MSIS

シンタックス： MMEMory:LOCk <File Name>,{ON | OFF | <NR1>}
MMEMory:LOCk? <File Name>



アーギュメント： <File Name>::=<文字列> ロックするファイルを指定します。
ON または 0 以外の値 — ファイルをロックします。
OFF または 0 — ファイルのロックを解除します。

注： ファイルがロックされた場合には、次の処理ができなくなります。

- ファイルの削除
- セーブによる上書き
- コメント情報の書き込み
- ファイル名の変更

使用例： 以下は、カレント・マス・メモリにあるファイル SINE_W1.WFM をロックする例です。

```
:MMEMORY:LOCK "SINE_W1.WFM", 1
```

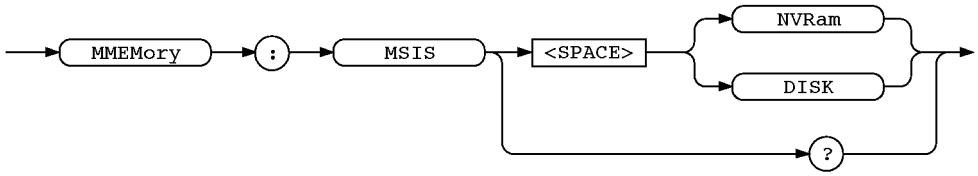
MMEMory:MSIS　(?)

MMEMory:MSIS コマンドは、カレント・マス・メモリとして、NVRAM または DISK を選択します。また MMEMory:MSIS? 問い合わせコマンドは、カレント・マス・メモリとして、NVRAM または DISK のいずれが選択されているかを問い合わせます。

グループ： MEMORY

関連コマンド： DISK:CDIRectory

シンタックス： MMEMory:MSIS　{NVRam | DISK}
MMEMory:MSIS?



アーギュメント： NVRam　— Non Volatile RAM を選択します。
DISK　— フロッピー・ディスクを選択します。

カレント・マス・メモリとして DISK を選択した場合、必要ならば、DISK:CDIRectory コマンドでカレント・ワーキング・ディレクトリを移動することができます。

使用例： 以下は、カレント・マス・メモリとして DISK を選択し、カレント・ワーキング・ディレクトリを \SAMPL1 に移動する例です。

```
:MMEMORY:MSIS  DISK ; :DISK:CDIRECTORY  "\SAMPL1"
```

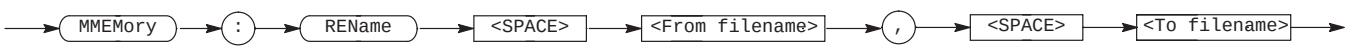
MMEMory:REName

MMEMory:REName コマンドは、カレント・マス・メモリの指定ファイルの名称を変更します。

グループ： MEMORY

関連コマンド： MMEMory:DELete、MMEMory:MSIS、MMEMory:LOCk

シンタックス： MMEMory:REName　<From-filename>, <To-filename>



アーギュメント： <From-filename>::=<文字列　>　　変更前のファイル名
<To-filename>::=<文字列>　　変更後のファイル名

<From-filename> と <To-filename> は、ファイルの拡張子が同じでなければなりません。異なったファイル拡張子を指定するとエラーになります。

使用例： 以下は、ファイルの名称を TDS_REF.WFM から AWGCH2.WFM に変更する例です。

```
:MMEMORY:RENAME  ”TDS_REF.WFM”, ”AWGCH2.WFM”
```

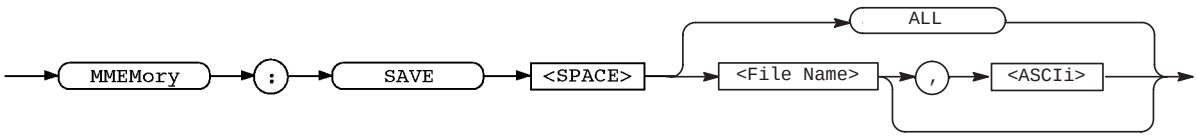
MMEMory:SAVE

MMEMory:SAVE コマンドは、内部メモリにあるファイルをカレント・マス・メモリにセーブします。

グループ： MEMORY

関連コマンド： MMEMory:LOAD、MMEMory:MSIS、MMEMory:LOCk

シンタックス： MMEMory:SAVE　{<File Name>[,<ASCIi>] | ALL}



アーギュメント： <File Name>::=<文字列>　　指定する 1 個のファイルをセーブします。
ALL　　— 内部メモリの全てのファイルをセーブします。
ASCIi　　— <File Name> がイクエーション・ファイル (.EQU) であり、かつカレント・マス・メモリがDISKのときのみ有効で、イクエーション・ファイルの内容を ASCII コードに変換してセーブします。このとき、セーブされたファイルの拡張子 (Type) は “.EQA” となります。

使用例： 以下は、内部メモリにある全てのファイルをマス・メモリにセーブする例です。

```
:MMEMORY:SAVE  ALL
```

MODE　(?)

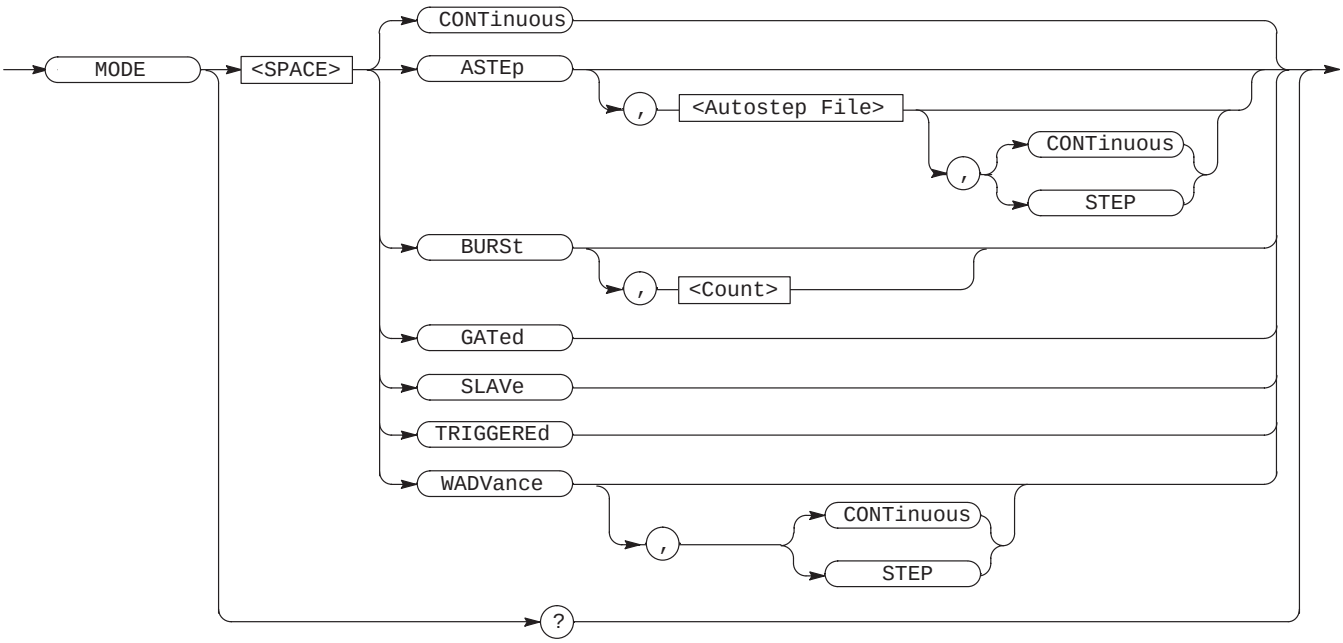
MODE コマンドは、トリガ・イベントが発生した際に波形を出力するモードを指定します。また MODE? 問い合わせコマンドは、指定中のモードを問い合わせます。

グループ： MODE

関連コマンド： STARt、STOP、*TRG

シンタックス： MODE {CONTinuous | ASTEp [,<Autostep File> [,CONTinuous | ,STEP] | BURst [,<Count>] | GATed | SLAVe | TRIGGERED | WADVance [,CONTinuous | ,STEP] }

MODE?



アーギュメント： 選択は、以下の通りです。

アーギュメント	説　明
CONTinuous	Contモードに設定します。Contモードは、連続的に波形またはシーケンスを出力します。
ASTep	Autostepモードに設定します。Autostepモードはトリガーが発生する毎に、オート・ステップ・ファイルに記述されるステップを1ステップづつ進めながら波形またはシーケンスを出力します。 <Autostep File>::=<文字列> AWG2005/40/41型の場合は、次のオプショナル・アーギュメントが付けられます。 COUTinuous — そのステップを繰り返し行ないます。 STEP — そのステップを1回だけ行ないます。

アーギュメント	説　明　(続き)
BURSt	Burstモードに設定します。Burstモードは、トリガが発生する毎に、<Count>サイクルの波形またはシーケンスを出力します。 <Count>::=<NR1>　Burstカウント (設定範囲: 1～65535) (AWG2010/11/20/21/40/41型のみ。ただし　AWG2041　型では、HWSequencer:MODEがOFFに設定されている場合にのみ有効です。)
GATed	Gated モードに設定します。Gated モードは、トリガが有効である間(フロント・パネルの **TRIGGER MANUAL** キーが押されている間、外部ゲート信号が有効である間、または STARt/*TRG コマンドが実行されてから STOP コマンドが実行されるまでの間)だけ、波形またはシーケンスを出力します。
SLAVe	Slaveモードに設定します。Slaveモードは、AWG2040/41型がスレーブ側として動作し、マスタとするAWG2040/41型と同じ動作モードになります。また、マスタとするAWG2040/41型のトリガ、ゲートおよびストップ信号の発生に同期して動作します。 (AWG2040/41型のみ)
TRIGGEREd	Triggeredモードに設定します。Triggerdモードは、トリガが発生する毎に、1サイクルの波形またはシーケンスを出力します。
WADVance	Waveform Advanceモードに設定します。Waveform Advanceモードは、トリガが発生する毎に、1 波形づつ進ませながら各波形を繰り返し連続的に出力します。 AWG2041 型の場合で、HWSequencer:MODE が OFF に設定されている場合には、トリガが発生する毎に各波形を 1 回のみ出力します。 AWG2005型の場合は、次のオプショナル・アーギュメントが付けられます。 CONTinuous　— その波形を繰り返します。 STEP　— その波形を1回だけ出力します。

*　トリガ・イベントが発生してから出力が開始されるタイミング、出力が停止するタイミングについての詳細は、ユーザ・マニュアルを参照ください。

使用例：　以下は、Burst モードに設定し、Burst カウントを 200 に設定する例です。

```
:MODE  BURST ,200
```

*OPC　(?)

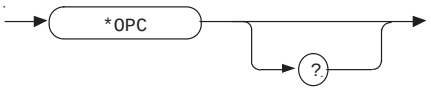
*OPC 共通コマンドは、ペンディング中の全ての操作が終了した場合、SES (Standard Event Status Register) のビット 0 をセットします。また *OPC? 共通・問い合わせコマンドは、ペンディング中の全ての操作が終了した場合、ASCII コード ”1” を戻します。

使用方法については、第 3 章の 「実行の同期について」、および第 4 章の 「プログラム例 3」を参照ください。

グループ：　SYNCHRONIZATION

関連コマンド：　*WAI

シンタックス： *OPC
*OPC?



アーギュメント：

使用例： 以下は、イクエーションのコンパイルの終了を待つ例です。

EQUATION:COMPILE:STATE EXECUTE, ”SAMPLE.EQU”; *OPC

以下は、ハードコピーの終了を待つ例です。

HCOPY:PORT DISK; HCOPY START; *OPC

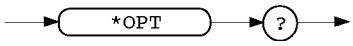
*OPT?

*OPT 共通・問い合わせコマンドは、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： SYSTEM

関連コマンド：

シンタックス： *OPT?



アーギュメント：

レスポンス： レスポンス・フォーマットは、以下の通りです。

<Option>[,<Option>]...

以下のオプションが認識されます。

0	—	オプションは組み込まれていません。
CH2	—	オプション 02 型 (2 チャンネル出力) が組み込まれています。(AWG2010/11/20/21型)
CH3/4	—	オプション02型 (4チャンネル出力) が組み込まれています。(AWG2005型)
DDO	—	オプション 03 型 (Digital Data Out) が組み込まれています。(AWG2010/11/20/21/40/41型) オプション 04 型 (Digital Data Out) が組み込まれています。(AWG2005型)
SWP	—	オプション 05 型 (クロック・スイープ) が組み込まれています。(AWG2005型)
FPP	—	オプション 09 型 (Floating Point Processor) が組み込まれています。
4M	—	オプション 01 型 (4Mワード波形メモリ) が組み込まれています。(AWG2040/41型)

使用例： 以下は、*OPT? のレスポンス例です。

CH2, FPP

この場合、オプション 02 型とオプション 09 型が組み込まれていることを示しています。

OUTPut:CH<x>?

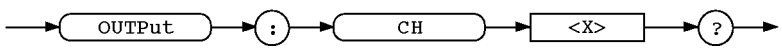
OUTPut:CH<x>? 問い合わせコマンドは、波形またはシーケンスが出力中かどうかを問い合わせます。

AWG2040/41型の場合、CH1 のみが有効なヘッダ・ニーモックとなります。

グループ： OUTPUT

関連コマンド： OUTPut:CH<x>:STATe

シンタックス： OUTPut:CH<x>?



アーギュメント：

レスポンス：　1　—　波形またはシーケンスが出力中です。
0　—　出力が OFF になっています。

出力の ON/OFF はフロント・パネルの出力コネクタにつながるリレーを切り替えるもので、OUTPut:CH<x>:STATe コマンドで行なうことができます。本コマンドは、このリレーの切り替え状態を調べるもので、OUTPut:CH<x>:STATe? 問い合わせコマンドと同じ機能です。

使用例：　以下は、:OUTPUT:CH1? に対する出力例です。

AWG2005/AWG2010/AWG2011/AWG2020/AWG2021型
```
OUTPUT:CH1:STATE
```

AWG2040/AWG2041型
```
TPUT:CH1:INVERTED:STATE 1 ;OUTPUT:CH1:NORMAL:STATE 1
```

OUTPut:CH<x>:STATe　(?)

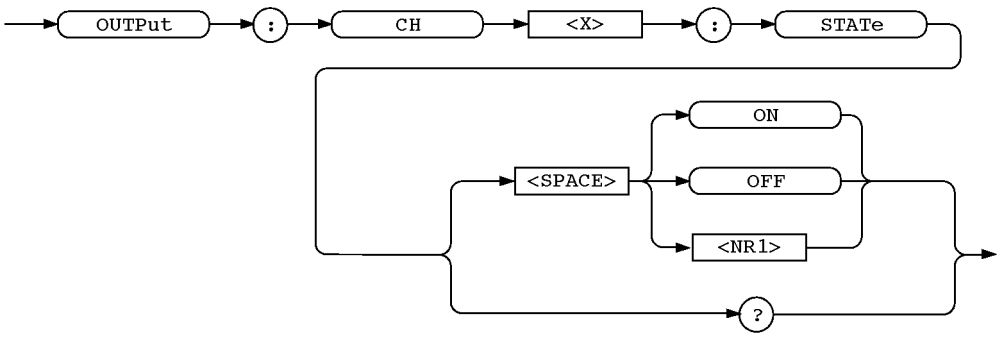
(AWG2005/10/11/20/21型のみ)

OUTPut:CH<x>:STATe コマンドは、指定チャンネルの出力を ON または OFF にします。また OUTPut:CH<x>:STATe? 問い合わせコマンドは、出力が ON かどうかの問い合わせを行います。

グループ：　OUTPUT

関連コマンド：　OUTPut:CH<x>?、OUTPut:CH1:NORMal:STATe、OUTPut:CH1:INVerted:STATe

シンタックス：　OUTPut:CH<x>:STATe　{ON | OFF | <NR1>}
OUTPut:CH<x>:STATe?



アーギュメント：　ON または 0 以外の値　—　出力を ON にします。
OFF または 0　—　出力を OFF にします。

出力の ON/OFF は、フロント・パネルの出力コネクタにつながるリレーを切り替えるものです。

レスポンス： 問い合わせコマンドに対するレスポンスは、以下の通りです。

1 — 出力が ON になっています。
0 — 出力が OFF になっています。

使用例： 以下は、チャンネル1の出力を ON にする例です。

```
:OUTPUT:CH1:STATE 1
```

OUTPut:CH1:INVerted?

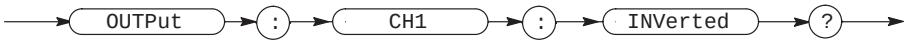
(AWG2040/41型のみ)

OUTPut:CH1:INVerted? 問い合わせコマンドは、反転出力が ON かどうかの問い合わせを行ないます。

グループ： OUTPUT

関連コマンド： OUTPut:CH1:INVerted:STATe

シンタックス： OUTPut:CH1:INVerted?



アーギュメント：

レスポンス： レスポンス・フォーマットは以下のようになります。

```
[:OUTPUT:CH1:INVERTED:STATE] <State>
```

ここで <State> ::=<NR1> は以下の通りです

1 — 反転出力が ON になっています。
0 — 反転出力が OFF になっています。

使用例： 以下は、反転出力の状態を問い合わせます。

```
:OUTPUT:CH1:INVERTED?
```

OUTPut:CH1:INVerted:STATe　(?)

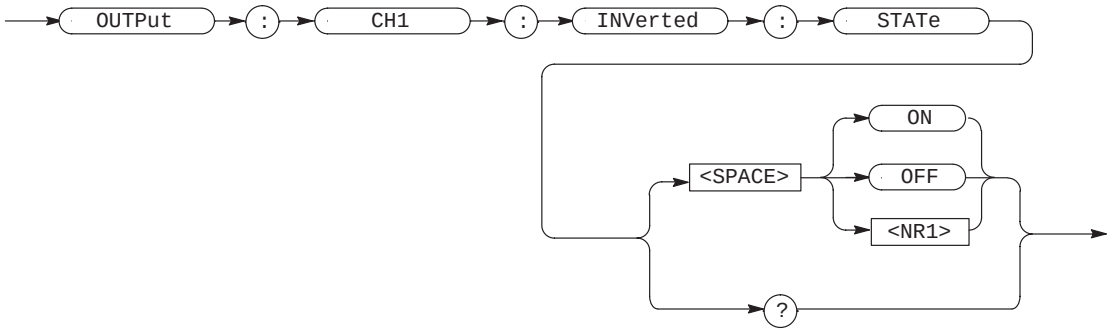
(AWG2040/41型のみ)

OUTPut:CH1:INVerted:STATe コマンドは、 反転出力を ON または OFF にします。また、OUTPut:CH1:INVerted:STATe? 問い合わせコマンドは、反転出力が ON かどうかの問い合わせを行ないます。

グループ： OUTPUT

関連コマンド： OUTPut:CH1:STATe

シンタックス： OUTPut:CH1:INVerted:STATe {ON | OFF | <NR1>}
OUTPut:CH1:INVerted:STATe?



アーギュメント： ON または 0 以外の値　— 反転出力を ON にします。
OFF または 0　— 反転出力を OFF にします。

レスポンス： 問い合わせに対するレスポンスは以下の通りです。

1　— 反転出力が ON になっています。
0　— 反転出力が OFF になっています。

使用例： 以下は、反転出力を ON にする例です。

```
:OUTPUT:CH1:INVERTED:STATE  ON
```

OUTPut:CH1:NORMal?

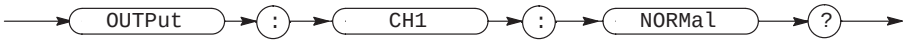
(AWG2040/41型のみ)

OUTPut:CH1:NORMal? 問い合わせコマンドは、非反転出力が ON かどうかの問い合わせを行ないます。

グループ： OUTPUT

関連コマンド： OUTPut:CH1:NORMal:STATe

シンタックス： OUTPut:CH1:NORMal?



アーギュメント：

レスポンス： レスポンス・フォーマットは以下のようになります。

[:OUTPUT:CH1:NORMAL:STATE] <State>

ここで <State> ::=<NR1> は以下の通りです

1 — 非反転出力が ON になっています。

0 — 非反転出力が OFF になっています。

使用例： 以下は、非反転出力の状態を問い合わせます。

:OUTPUT:CH1:NORMAL?

OUTPut:CH1:NORMal:STATe (?)

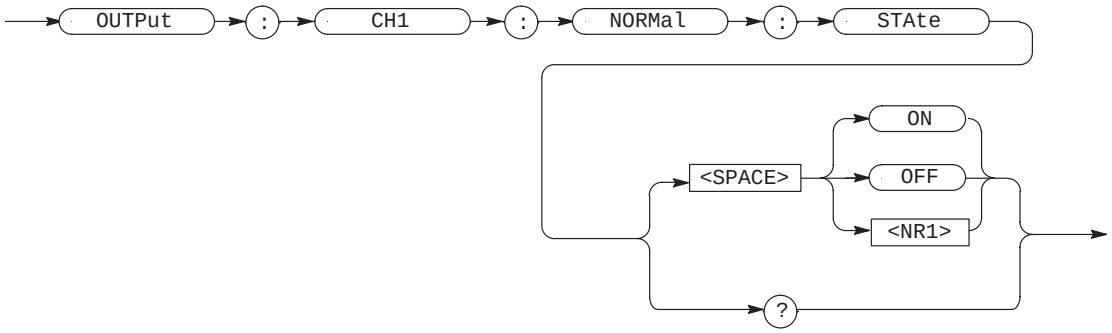
(AWG2040/41型のみ)

OUTPut:CH1:NORMal:STATe コマンドは、非反転出力を ON または OFF にします。また、OUTPut:CH1:NORMal:STATe? 問い合わせコマンドは、非反転出力が ON かどうかの問い合わせを行ないます。本コマンドは、OUTPut:CH1:STATe と全く同等です。

グループ： OUTPUT

関連コマンド： OUTPut:CH1:STATe

シンタックス： OUTPut:CH1:NORMal:STATe {ON | OFF | <NR1>}
OUTPut:CH1:NORMal:STATe?



アーギュメント： ON または 0 以外の値　— 非反転出力を ON にします。
OFF または 0　— 非反転出力を OFF にします。

レスポンス： 問い合わせに対するレスポンスは以下の通りです。

1 — 非反転出力が ON になっています。
0 — 非反転出力が OFF になっています。

使用例： 以下は、非反転出力を ON にする例です。

```
:OUTPUT:NORMAL:STATE  ON
```

OUTPut:SYNC (?)

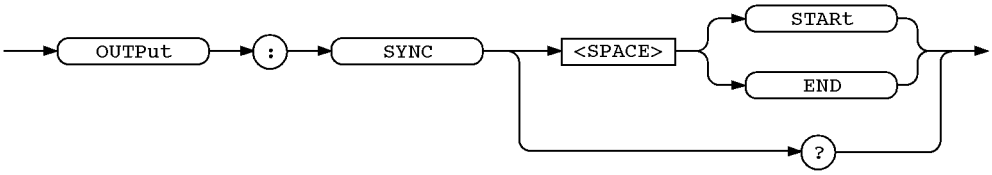
(AWG2010/11/20/21型のみ)

OUTPut:SYNC コマンドは、フロント・パネルの SYNC コネクタから出力する Sync 信号を生成するタイミングを指定します。また OUTPUT:SYNC? 問い合わせコマンドは、Sync 信号を生成するタイミングを問い合わせます。

グループ： OUTPUT

関連コマンド：

シンタックス： OUTPut:SYNC {STARt | END}
OUTPut:SYNC?



アーギュメント： STARt — 波形またはシーケンスの出力開始時 (トリガ発生時)、Sync 信号を生成します。
END — 波形またはシーケンスの出力終了時、Sync 信号を生成します。

使用例： 以下は、波形またはシーケンスの出力終了時に Sync 信号が生成されるように設定する例です。

```
:OUTPUT:SYNC  END
```

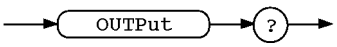
OUTPut?

OUTPut? 問い合わせコマンドは、OUTPUT グループのコマンドによって可能な全ての設定に対して、設定中の値を問い合わせます。

グループ： OUTPUT

関連コマンド：

シンタックス： OUTPut?



アーギュメント：

レスポンス： コマンドが連結されたフォーマットで戻されます。詳しくは、使用例を参照ください。

使用例： 以下は、:OUTPUT? のレスポンス例です。

```
:OUTPUT:CH1:STATE 0; :OUTPUT:CH2:STATE 0; :OUTPUT:SYNC END
```

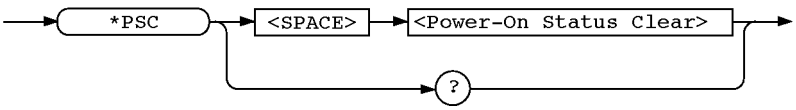
*PSC　(?)

*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・問い合わせコマンドは、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： STATUS & EVENT

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 にする直前の状態の設定に戻します。さらに、電源投入後の SRQ の発行が可能となります。

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： 問い合わせコマンドに対するレスポンスは、以下の通りです。

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： 以下は、PSC フラグを TRUE に設定する例です。

```
*PSC 1
```

次は、*PSC? のレスポンス例です。

```
0
```

この場合、PSC フラグが FALSE に設定されていることになります。

*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： SYSTEM

関連コマンド： FACTory、SECUre

シンタックス： *RST

アーギュメント：

使用例： 以下は、機器をデフォルトの設定に戻す例です。

```
*RST
```

RUNNing?

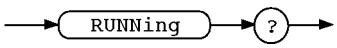
RUNNing? 問い合わせコマンドは、波形またはシーケンスが出力中かどうかを問い合わせます。

グループ： MODE

関連コマンド： STARt、STOP、*TRG

シンタックス： RUNNing?

アーギュメント： <<MD-02>>



アーギュメント：

レスポンス： レスポンスは、以下の通りです。

1 — 波形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： 以下は、:RUNNING? のレスポンス例です。

:RUNNING 1

この場合、波形またはシーケンスが出力中であることを示します。

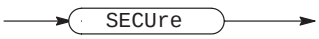
SECUre

SECUre コマンドは、機器の内部メモリと内部不揮発性メモリをすべて初期化し、全設定を工場出荷時のデフォルト設定に戻します。

グループ： SYSTEM

関連コマンド： *RST、FACTory

シンタックス： SECUre



使用例： 以下は、機器の内部メモリと内部不揮発性メモリを初期化し、全設定を工場出荷時の設定に戻す例です。

:SECURE

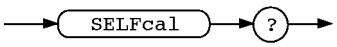
SELFcal?

SELFcal? 問い合わせコマンドは、選択中のキャリブレーション・ルーチンと実行結果を問い合わせます。

グループ： CALIBRATION & DIAGNOSTIC

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

シンタックス： SELFcal?



アーギュメント：

レスポンス： レスポンス・フォーマットは、以下の通りです。

[:SELFCAL:SELECT] <Calibration Routine> ;[RESULT] <Result>[,<Result>]...

ここで <Calibration Routine> は、次のいずれかです。

ALL	—	全てのルーチン
CLOCk	—	CLOCK ユニット・キャリブレーション・ルーチン (AWG2010/11/20/21型)
SETUp	—	SETUP 関係のキャリブレーション・ルーチン
TRIGger	—	TRIGGERユニット・キャリブレーション・ルーチン (AWG2005/10/11/20/21型)

<Results>::=<NR1> 次のいずれかです。

0	—	正常に終了しました。
200	—	CLOCK ユニットでエラーが検出されました (AWG2010/11/20/21型)。
600	—	SETUP 関係のエラーが検出されました。
800	—	TRIGGER ユニット・エラーが検出されました (AWG2005/10/11/20/21型)。

使用例： 以下は、:SELFCAL? に対するレスポンス例です。

:SELFCAL:SELECT ALL; RESULT 0

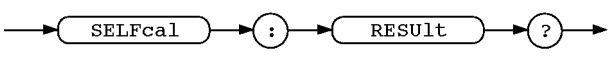
SELFcal:RESUlt?

SELFcal:RESUlt? 問い合わせコマンドは、キャリブレーションの実行結果を問い合わせます。

グループ： CALIBRATION & DIAGNOSTIC

関連コマンド： SELFcal:SELect、SELFcal:STATe

シンタックス： SELFcal:RESUlt?



アーギュメント：

レスポンス：　レスポンス・フォーマットは、以下の通りです。

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

<Results>::=<NR1>　次のいずれかです。

0　— 正常に終了しました。
200　— CLOCK ユニットでエラーが検出されました (AWG2010/11/20/21型)。
600　— SETUP 関係のエラーが検出されました。
800　— TRIGGER ユニット・エラーが検出されました (AWG2005/10/11/20/21型)。

使用例：　以下は、:SELFCAL:RESULT? のレスポンス例です。

:SELFCAL:RESULT 200, 600

上記の場合、CLOCK ユニットと SETUP 関係のユニットでエラーが検出されたことを示しています。

SELFcal:SELect　(?)

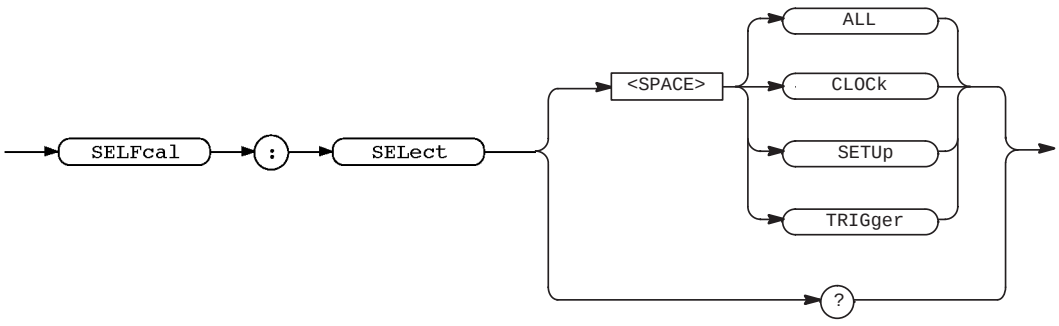
SELFcal:SELect コマンドは、キャリブレーション・ルーチンを選択します。また SELFcal:SELect? 問い合わせコマンドは、選択中のキャリブレーション・ルーチンを問い合わせます。

グループ：　CALIBRATION & DIAGNOSTIC

関連コマンド：　SELFcal:STATe、SELFcal:RESUlt?

シンタックス：　SELFcal:SELect {ALL |
CLOCK (AWG2010/11/20/21型) | SETUp | TRIGger (AWG2005/10/11/20/21型) }

SELFcal:SELect?



アーギュメント： ALL — 全てのキャリブレーション・ルーチンを選択します。
CLOCk — CLOCK ユニット・キャリブレーション・ルーチンを選択します (AWG2010/11/20/21型)。
SETUp — SETUP 関係のキャリブレーション・ルーチンを選択します。
TRIGger — TRIGGER ユニット・キャリブレーション・ルーチンを選択します (AWG2005/10/11/20/21型)。

使用例： 以下は、CLOCK ユニット・キャリブレーション・ルーチンを選択し、実行する例です。

```
:SELFCAL:SELECT  CLOCK ;  STATE  EXECUTE
```

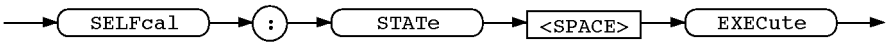
SELFcal:STATe

SELFcal:STATe コマンドは、SELFcal:SELect コマンドで選択中のキャリブレーション・ルーチンを実行します。各ルーチンは、キャリブレーション実行中にエラーを検出すると実行を中止します。また全てのキャリブレーション・ルーチンが選択されている場合には、実行中にエラーを検出するとそのルーチンの実行を中止して、次のルーチンを実行します。

グループ： CALIBRATION & DIAGNOSTIC

関連コマンド： SELFcal:SELect、SELFcal:RESUlt?

シンタックス： SELFcal:STATe EXECute



アーギュメント： EXECute — 選択中のキャリブレーション・ルーチンを実行します。

使用例： 以下は、全てのキャリブレーション・ルーチンを選択し、実行する例です。

```
:SELFCAL:SELECT  ALL ; STATE  EXECUTE
```

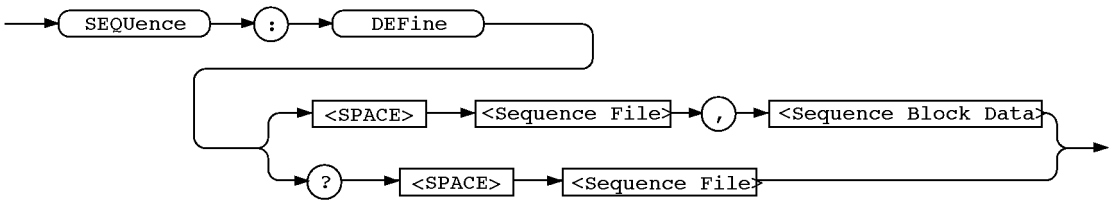
SEQUence:DEFine　(?)

SEQUence:DEFine コマンドは、指定ファイルにシーケンス・データを書き込みます。また SEQUence:DEFine? 問い合わせコマンドは、指定ファイルに書き込まれているシーケンス・データを問い合わせます。

グループ： WAVEFORM

関連コマンド：

シンタックス： SEQUence:DEFine　<Sequence File>,<Sequence Block Data>
SEQUence:DEFine?　<Sequence File>



アーギュメント： <Sequence File>::=<文字列>　シーケンス・ファイル
<Sequence Block Data>::=<Arbityary Block>

<Sequence Block Data> は、以下の例のように ASCII コードで記述され、各シーケンスとシーケンス・カウントは、ライン・フィード(LF)で区切られます。

使用例： 以下は、ファイル SQWAVE.SEQ にシーケンス・データを書き込む例です。

:SEQUENCE:DEFINE　”SQWAVE.SEQ”, #255WAVE01.WFM, 10 <LF>
WAVE02.WFM, 10 <LF> WAVE03.WFM, 10 <LF> WAVE04.WFM, 10

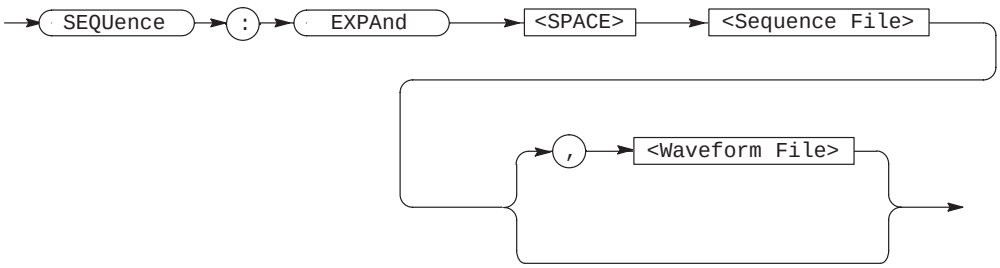
SEQUence:EXPAnd

SEQUence:EXPand コマンドは、指定シーケンス・ファイルに記録されたシーケンスを波形データに展開し、波形ファイルを生成します。

グループ： WAVEFORM

関連コマンド： SEQUence:DEFine

シンタックス： SEQUence:EXPAnd <Sequence File>[, <Waveform File>]



アーギュメント：
<Sequence File>::=<文字列> シーケンス・ファイル名
<Waveform File>::=<文字列> 波形ファイル名

シーケンス・ファイルおよび波形ファイルは、内部メモリ上のファイルです。波形ファイル <Waveform File> の指定を省略すると、シーケンス・ファイルのベース名に拡張子 .WFM を添えた波形ファイルが生成されます。生成されるべき波形ファイルと同じファイル名を持つ波形ファイルが既に存在する場合には、エラーとなります。

展開後の波形ファイルの波形ポイント数は、シーケンスに記述された各波形ファイルのポイント数と繰り返し数の積を加算したものです。

使用例： 以下は、シーケンス・ファイル SQWAVE.SEQ を波形ファイルに展開する例です。この場合、生成される波形ファイルは、SQWAVE.WFM となります。

```
:SEQUENCE:EXPAND "SQWAVE.SEQ"
```

次は、波形ファイルを SWAVE01.WFM として生成する例です。

```
:SEQUENCE:EXPAND "SQWAVE.SEQ", "SWAVE01.WFM"
```

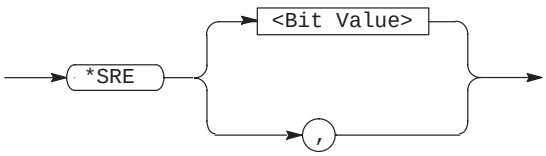
*SRE　(?)

*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・問い合わせコマンドは、SRER の内容を問い合わせます。

グループ： STATUS & EVENT

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、全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。

使用例： 以下は、SRER を 48 (00110000) に設定する例です。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、波形またはシーケンスを出力するために、トリガ・イベントを生成します。

グループ： MODE

関連コマンド： MODE、RUNNing?、STOP、*TRG

シンタックス： STARt

アーギュメント：

使用例： 以下は、トリガ・イベントを生成する例です。

```
:START
```

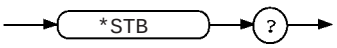
*STB?

*STB? 共通・問い合わせコマンドは、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： STATUS & EVENT

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：

レスポンス： 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： 以下は、*STB? のレスポンス例です。

```
96
```

この場合、SBR の内容は 01100000 となります。

STOP

STOP コマンドは、波形またはシーケンスの出力を停止します。また Autostep モードまたは Waveform advance モードのときには、シーケンス・ポインタをリセットし、次のトリガでシーケンスの最初から出力されるようにします。

グループ： MODE

関連コマンド： MODE、RUNNing?、STARt、*TRG

シンタックス： STOP

アーギュメント：

使用例： 以下は、波形またはシーケンスの出力を停止する例です。

```
:STOP
```

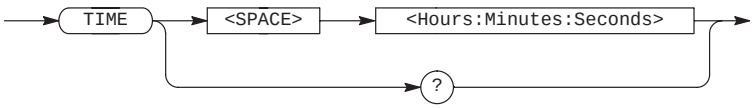
TIME　(?)

TIME コマンドは、時刻を設定します。また TIME? 問い合わせコマンドは、時刻を問い合わせます。

グループ： SYSTEM

関連コマンド： DATE

シンタックス： TIME　<Hours:Minutes:Seconds>
TIME?



アーギュメント： <Hours:Minutes:Seconds>::=<文字列>　以下のフォーマットでなければなりません。

```
"HH:MM:SS"
```

ここで

HH　— 時間 (0 ～ 23)
MM　— 分 (0 ～ 59)
SS　— 秒 (0 ～ 59)

使用例： 以下は、時刻を設定する例です。

```
:TIME  ”11:23:58”
```

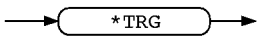
*TRG

*TRG 共通コマンドは、トリガ・イベントを生成します。本コマンドは、STARt コマンドと全く同等です。

グループ： MODE

関連コマンド： MODE、RUNNing?、STARt、STOP

シンタックス： *TRG



アーギュメント：

使用例： 以下は、トリガ・イベントを生成する例です。

```
*TRG
```

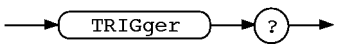
TRIGger?

TRIGger? 問い合わせコマンドは、トリガに関連する設定のうち、設定中の全設定値を問い合わせます。

グループ： MODE

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

シンタックス： TRIGger?



アーギュメント：

レスポンス： 使用例を参照ください。

使用例：　以下は、:TRIGGER? に対するレスポンス例です。

:TRIGGER:IMPEDANCE␣LOW; LEVEL␣1.400; POLARITY␣POSITIVE; SLOPE␣POSITIVE

TRIGger:HOLDoff　(?)

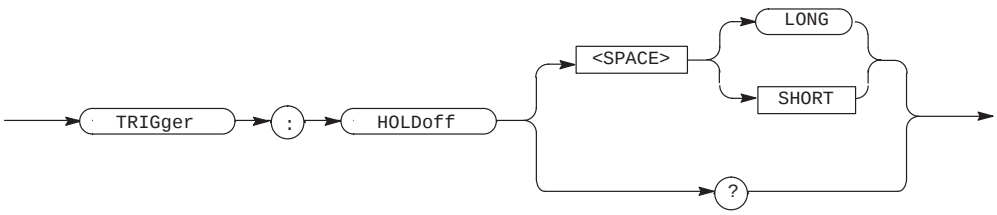
(AWG2010/11/20/21型のみ)

TRIGger:HOLDoff コマンドは、トリガ・ホールドオフの時間を選択します。TRIGger:HOLDoff? 問い合わせコマンドは、選択されているトリガ・ホールドオフ時間を問い合わせます。

グループ：　MODE

関連コマンド：　TRIGer:IMPedance、TRIGger:LEVel、TRIGger:POLarity、TRIGger:SLOPe

シンタックス：　TRIGger:HOLDoff {LONG | SHORt}
TRIGger:HOLDoff?



アーギュメント：　LONG　—　長い (最大約 1 秒) ホールドオフ時間を選択します。
SHORt　—　短い (最大約 0.002 秒以下) ホールドオフ時間を選択します。
ただし、トリガ信号の繰返しが速く、波形出力時間が短い場合には、フロント・パネルおよびリモートからの操作が遅くなることがあります。

使用例：　以下は、トリガ・ホールドオフ時間を短くする例です。

:TRIGGER:HOLDOFF SHORT

TRIGger:IMPedance　(?)

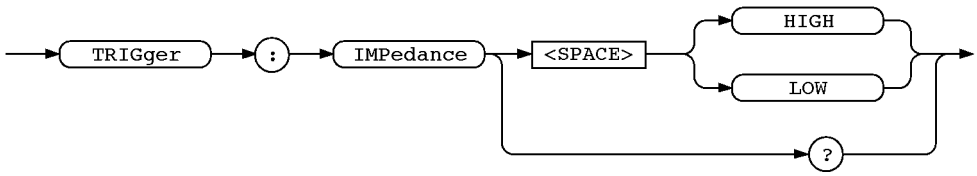
(AWG2010/11/20/21/40/41型のみ)

TRIGger:IMPedance コマンドは、外部トリガ信号に対する入力インピーダンスを選択します。TRIGger:IMPedance? 問い合わせコマンドは、選択中の外部トリガ・ソースに対する入力インピーダンスを問い合わせます。

グループ：　MODE

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

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



アーギュメント： HIGH — 高インピーダンス (1 MΩ：AWG2010/11/20/21型、1kΩ：AWG2040/41型) を選択します。
LOW — 低インピーダンス (50 Ω) を選択します。

使用例： 以下は、低インピーダンス (50 Ω) を選択する例です。

```
:TRIGGER:IMPEDANCE LOW
```

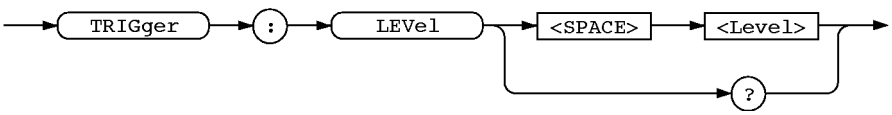
TRIGger:LEVel (?)

TRIGger:LEVel コマンドは、外部トリガ信号のトリガ・レベルを設定します。また TRIGger:LEVel? 問い合わせコマンドは、設定中のトリガ・レベルを問い合わせます。

グループ： MODE

関連コマンド： TRIGger:IMPedance、TRIGger:POLarity、TRIGger:SLOPe

シンタックス： TRIGger:LEVel <Level>
TRIGger:LEVel?



アーギュメント： <Level>::=<NR2>[<unit>]
<unit>::={V | MV} (設定範囲：−5.0 ~ 5.0 V、ステップ 0.1 V)

使用例： 以下は、トリガ・レベルを 200 mV に設定する例です。

```
:TRIGGER:LEVEL 200 mV
```

TRIGger:POLarity　(?)

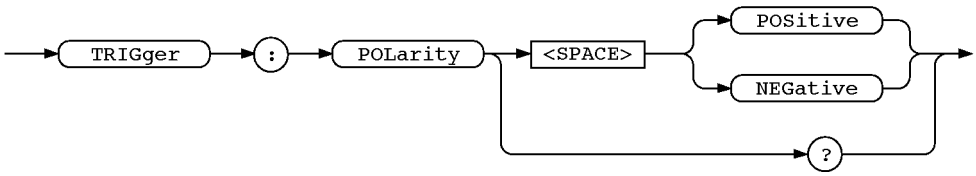
TRIGger:POLarity コマンドは、外部トリガ・ソースのトリガ極性を設定します。また TRIGger:POLarity? 問い合わせコマンドは、設定中のトリガ極性を問い合わせます。

この極性パラメタは、Gated モードの場合のみ有効です。

グループ： MODE

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

シンタックス： TRIGger:POLarity　{POSitive | NEGative}
TRIGger:POLarity?



アーギュメント： POSitive　— 正極性を選択します。
NEGative　— 負極性を選択します。

使用例： 使用例:　以下は、負極性を選択する例です。

```
:TRIGGER:POLARITY  NEGATIVE
```

TRIGger:SLOPe　(?)

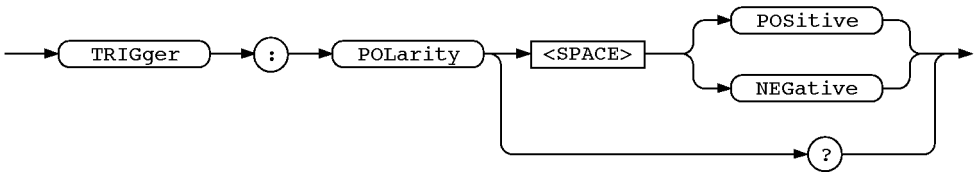
TRIGger:SLOpe コマンドは、外部トリガ信号に対してトリガ・イベントが、立ち上がりエッジで生成されるか立ち下がりエッジで生成されるか(スロープ)を選択します。また TRIGger:SLOPe? 問い合わせコマンドは、トリガ・イベントがいずれのエッジで生成されるのかを問い合わせます。

このスロープ・パラメタは、Cont モードと Gated モードを除くその他のモードに対してのみ有効です。

グループ： MODE

関連コマンド： TRIGger:IMPedance、TRIGger:LEVel、TRIGger:POLarity

シンタックス： TRIGger:SLOPe {POSitive | NEGative}
TRIGger:SLOPe?



アーギュメント： POSitive — 立ち上がりエッジでトリガ・イベントを生成します。
NEGative — 立ち下がりエッジでトリガ・イベントを生成します。

使用例： 以下は、立ち上がりエッジにトリガ・イベントの生成を設定する例です。

```
:TRIGGER:SLOPE  POSITIVE
```

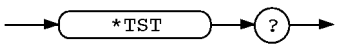
*TST?

*TST? 共通・問い合わせコマンドは、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： CALIBRATION & DIAGNOSTIC

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

シンタックス： *TST?



アーギュメント：

レスポンス： レスポンスは、以下の通りです。

<Result>

ここで <Result>::=<NR1> は、以下のいずれかです。

0 — 正常に終了しました。
100 — CPU ユニットでエラーが検出されました。
200 — CLOCK ユニットでエラーが検出されました。
300 — DISPLAY ユニットでエラーが検出されました。
400 — FPP (Flating-Point Processor) ユニットでエラーが検出されました。
500 — FP (Front Panel) ユニットでエラーが検出されました。
600 — SETUP 関係のエラーが検出されました。
700 — 波形メモリでエラーが検出されました。
800 — TRIGGER ユニットでエラーが検出されました。

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実行させようとしても本機器は反応しません。

使用例：　以下は、*TST? のレスポンス例です。

```
200
```

この場合、CLOCK ユニットでエラーが検出されたことを示しています。

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 コマンドで行なうことができます。なお本コマンドは、LOCk NONe コマンドと全く同等です。

グループ：　SYSTEM

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：　以下は、キーおよびノブのロックを解除する例です。

```
:UNLOCK  ALL
```

UPTime?

UPTime? 問い合わせコマンドは、電源投入後の経過時間を問い合わせます。

グループ：　SYSTEM

関連コマンド：

シンタックス：　UPTime?

アーギュメント：

使用例： 以下は、:UPTIME? に対するレスポンス例です。

:UPTIME 7.016

上記の場合、電源投入後 7.016 時間経過していることを示しています。

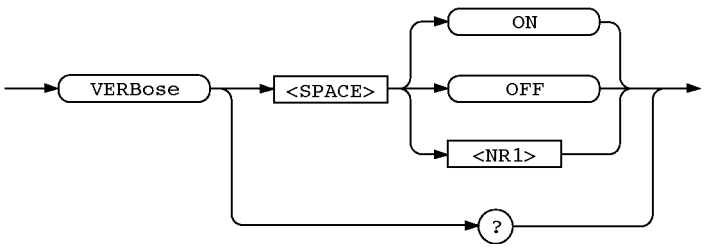
VERBose (?)

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： SYSTEM

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、以下の通りです。

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： 以下は、ロング・コマンド・ヘッダに設定する例です。

```
: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、ペンディング中の全ての操作が完了するまで、以後のコマンドまたは問い合わせコマンドの実行を待ちます。

グループ： SYNCHRONIZATION

関連コマンド： *OPC

シンタックス： *WAI

アーギュメント：

使用例： 3-9 ページの「実行の同期について」を参照ください。

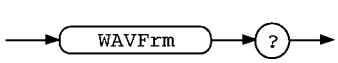
WAVFrm?

WAVFrm? 問い合わせコマンドは、波形データと波形プリアンブルを本機器から外部コントローラに転送します。本問い合わせコマンドは、WFMPre? 問い合わせコマンドと CURVe? 問い合わせコマンドを合わせた機能と同等です。

グループ： WAVEFORM

関連コマンド： CURVe?、DATA:SOURce、DATA:ENCDG、WFMPre?

シンタックス： WAVFrm?



アーギュメント：

使用例： 以下は、:WAVEFRM? のレスポンス例です。

:WFMPRE:ENCDG BIN; BN_FMT RP; BYT_NR 2; BIT_NR 12; BYT_OR MSB; CRVCHK NONE; WFI ”WAVEFORM.WFM, 1000 points, clock: 100.0MHz, amplitude: 1.000V, offset: 0.000V”; NR_PT 1000; PT_FMT Y; XUNI ”S”; XINCR 1.0000E–08; PT_OFF 0; XZERO 0.000; YUNIT ”V”; YMULT 2.442E–04; YZERO 0.000; YOFF 2.047E+03; :CURVE #42000 ...

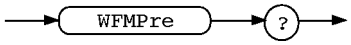
WFMPre?

WFMPre? 問い合わせコマンドは、波形プリアンブルの全ての設定のうち、設定中の値を問い合わせます。この問い合わせコマンドで参照できるプリアンブル情報は、DATA:SOURce コマンドで設定されたソース(波形データの転送元)に対応する情報です。

グループ： WAVEFORM

関連コマンド： WFMPRE サブ・グループの全てのコマンド、DATA:SOURce

シンタックス： WFMPre?



アーギュメント：

使用例： 以下は、:WFMPRE? のレスポンス例です。

:WFMPRE:ENCDG BIN;BN_FMT RP; BYT_NR 2; BIT_NR 12; BYT_OR MSB; CRVCHK NONE; WFID ”WAVEFORM.WFM, 1000 points, clock: 100.0MHz, amplitude: 1.000V, offset: 0.000V”; NR_PT 1000; PT_FMT Y; XUNIT ”S”; XINCR 1.0000E–08;PT_OFF 0;XZERO 0.000; YUNIT ”V”; YMULT 2.442E–04; YZERO 0.000; YOFF 2.047E+03

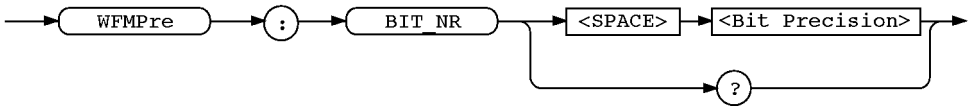
WFMPre:BIT_NR (?)

WFMPre:BIT_NR コマンドは、各バイナリ・データ・ポイントのビット精度を設定します。また WFMPre:BIT_NR? 問い合わせコマンドは、設定中のビット精度を問い合わせます。

グループ： WAVEFORM

関連コマンド： WFMPre:BN_FMT、WFMPre:BYT_NR、WFMPre:PT_FMT、WFMPre:BYT_OR、WFMPre:ENCDG、DATA:ENCDG、DATA:WIDTh

シンタックス： WFMPre:BIT_NR <Bit Precision>
WFMPre:BIT_NR?



アーギュメント： <Bit Precision>::=<NR1>

本機器の場合、ビット精度はデータ幅が 1 バイトのときは 8、2 バイトのときは12と解釈します。それ以外の値が指定されても無視されます。

使用例： 以下は、:WFMPRE:BIT_NR? のレスポンス例です。

```
:WFMPRE:BIT_NR  12
```

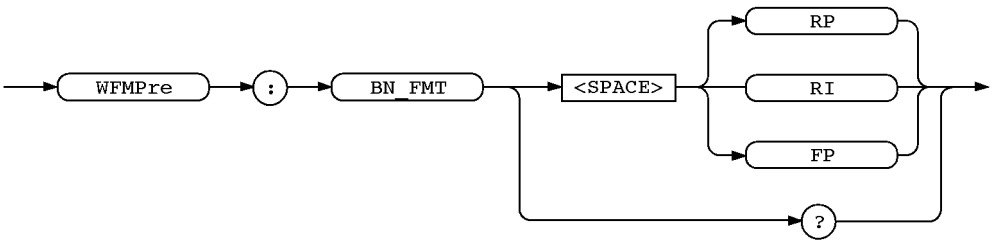
WFMPre:BN_FMT　(?)

WFMPre:BN_FMT コマンドは、バイナリ・データのフォーマットを指定します。また WFMPre:BN_FMT? 問い合わせコマンドは、設定中のバイナリ・データ・フォーマットを問い合わせます。

グループ： WAVEFORM

関連コマンド： WFMPre:BYT_NR、WFMPre:BIT_NR、WFMPre:BYT_OR、WFMPre:ENCDG、DATA:ENCDG

シンタックス： WFMPre:BN_FMT　{RP | RI | FP}
WFMPre:BN_FMT?



アーギュメント： RP　—　符号無し整数コード
RI　—　符号付き整数コード
FP　—　単精度浮動小数点コード

本機器では RP のみを取り扱います。よって RP (デフォルト) 以外の選択が行われても無視されます。

使用例： 以下は、:WFMPRE:BN_FMT? のレスポンス例です。

```
:WFMPRE:BN_FMT  RP
```

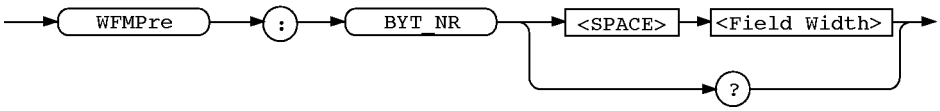
WFMPre:BYT_NR (?)

WFMPre:BYT_NR コマンドは、バイナリ・データ・ポイントのデータ幅 (バイト長) を設定します。また WFMPre:BYT_NR? 問い合わせコマンドは、設定中のデータ幅を問い合わせます。

グループ： WAVEFORM

関連コマンド： WFMPre:BN_FMT、WFMPre:BIT_NR、WFMPre:BYT_OR,WFMPre:ENCDG、DATA:ENCDG、DATA:WIDTh

シンタックス： WFMPre:BYT_NR <Field Width>
WFMPre:BYT_NR?



アーギュメント： <Field Width>::=<NR1>

本機器では、データ幅は 2 バイトまたは1バイトです。データ転送の際のデータ幅は、DATA:WIDTh コマンドでも設定することができます。本コマンドおよび DATA:WIDTh コマンドが同時に使用された場合には、一番最後の設定のみが有効となります。例えば、本コマンドにより、データ幅が 1 バイトに設定されていても (WFMPre:BYT_NR1)、DATA:WIDTh2 が実行されれば、データ幅 2 で転送されるように設定が変更されます。

使用例： 以下は、:WFMPRE:BYT_NR のレスポンス例です。

```
:WFMPRE:BYT_NR  2
```

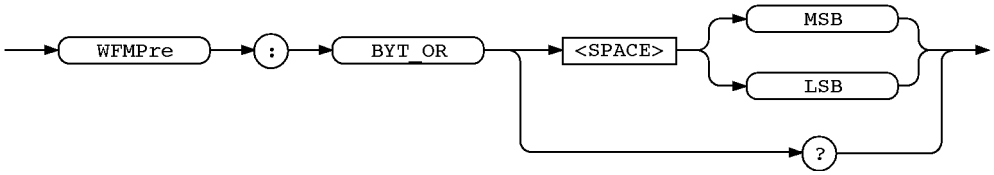
WFMPre:BYT_OR (?)

WFMPre:BYT_OR コマンドは、バイナリ・データ幅が2バイトのとき、どちらのバイトを先に転送するのかを設定します。また WFMPre:BYT_OR? 問い合わせコマンドは、どちらのバイトが先に転送されるのかを問い合わせます。

グループ： WAVEFORM

関連コマンド： WFMPre:BN_FMT、WFMPre:BYT_NR、WFMPre:BIT_NR、WFMPre:ENCDG、DATA:ENCDG、DATA:WIDTh

シンタックス： WFMPre:BYT_OR {MSB | LSB}
WFMPre:BYT_OR?



アーギュメント： MSB　— 上位バイトを先に転送します。
LSB　— 下位バイトを先に転送します。

データ転送の際のバイト・オーダは、DATA:ENCDG コマンドでも設定することができます。本コマンドおよび DATA:ENCDG コマンドが同時に使用された場合には、一番最後の設定のみが有効となります。例えば、本コマンドにより、下位バイトが先に転送されるように設定されていても (WFMPre:BYT_OR LSB)、DATA:ENCDG RPBinary が実行されれば、上位バイトが先に転送されるように設定が変更されます。

使用例： 以下は、:WFMPRE:BYT_OR? のレスポンス例です。

```
:WFMPRE:BYT_OR  MSB
```

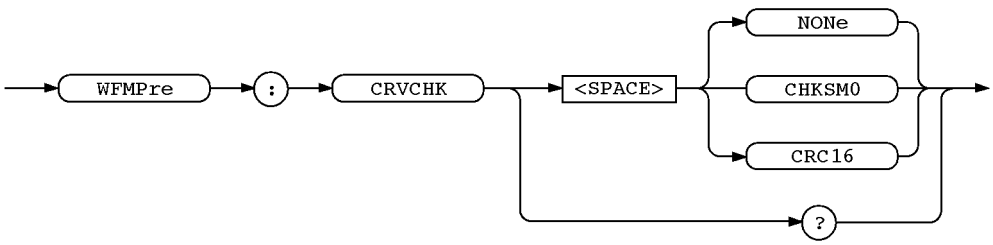
WFMPre:CRVCHK　(?)

WFMPre:CRVCHK コマンドは、バイナリ・データのエラー・チェック方法を選択します。また WFMPre:CRVCHK? 問い合わせコマンドは、選択中のエラー・チェック方法を問い合わせます。

グループ： WAVEFORM

関連コマンド： WFMPre:ENCDG、DATA:ENCDG

シンタックス： WFMPre:CRVCHK　{NONe | CHKSM0 | CRC16}
WFMPre:CRVCHK?



アーギュメント： NONe — エラー・チェックを行いません。データ・ブロック中の全てのバイナリ・データは、波形データのみを表し、チェックサム・コードを含みません。

CHKSM0 — バイナリ・データの最終バイトが、チェックサム・コードです。チェックサムは、チェックサムを除くバイナリ・データと ASCII コードの合計を 256 で割り、余りを 2 の補数表現したものです。

CRC16 — バイナリ・データの最後の 2 バイトが、16 ビット冗長チェック・コードです。

本機器では、エラー・チェックを行いません。よって NONe (デフォルト) 以外の選択が行われても無視されます。

使用例： 以下は、:WFMPRE:CRVCHK? のレスポンス例です。

```
:WFMPRE:CRVCH  NONE
```

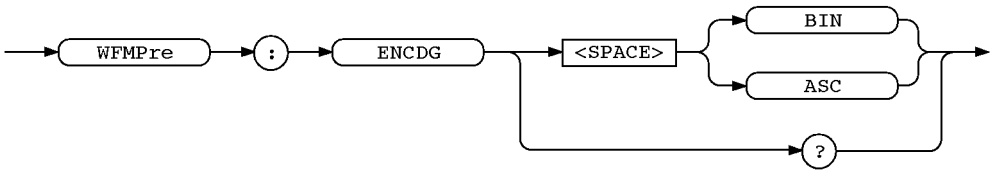
WFMPre:ENCDG (?)

WFMPre:ENCDG コマンドは、CURVe コマンドで転送される波形データが ASCII 形式かバイナリ形式かのエンコーディング形式を選択します。また WFMPre:ENCDG? 問い合わせコマンドは、設定中のエンコーディング情報を問い合わせます。

グループ： WAVEFORM

関連コマンド： DATA:ENCDG

シンタックス： WFMPre:ENCDG {BIN | ASC}
WFMPre:ENCDG?



アーギュメント： BIN — バイナリ・エンコーディング形式を選択します。

ASC — ASCII エンコーディング形式を選択します。

本機器では、バイナリ・エンコーディング形式のみが有効です。BIN (デフォルト) 以外の選択が行われても無視されます。

使用例： 以下は、:WFMPre:ENCDG? のレスポンス例です。

```
:WFMPRE:ENCDG  BIN
```

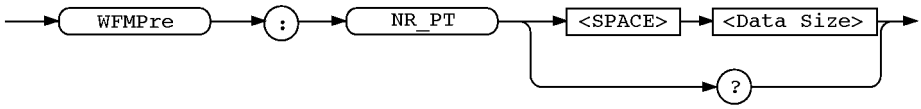
WFMPre:NR_PT　(?)

WFMPre:NR_PT コマンドは、波形データのデータ・サイズ(データ・ポイント数)を設定します。また WFMPre:NR_PT? 問い合わせコマンドは、設定中のデータ・サイズを問い合わせます。

グループ： WAVEFORM

関連コマンド： DATA:SOURce、DATA:DESTination

シンタックス： WFMPre:NR_PT　<Data Size>
WFMPre:NR_PT?



アーギュメント： <Data Size>::=<NR1>　　データ・ポイント数は本機器が自動的に設定します。よって入力値は無視されます。

使用例： 以下は、:WFMPRE:NR_PT? に対するレスポンス例です。

:WFMPRE:NR_PT 131072

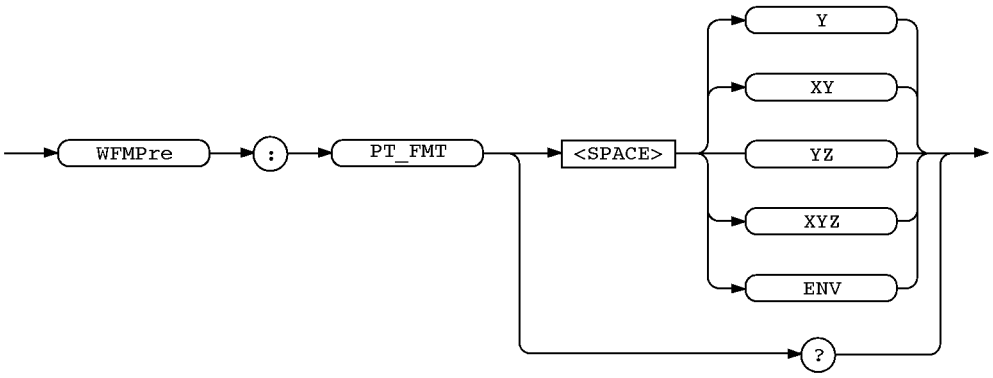
WFMPre:PT_FMT　(?)

WFMPre:PT_FMT コマンドは、データ・ポイントのフォーマットを指定します。また WFMPre:PT_FMT? 問い合わせコマンドは、設定中のデータ・ポイント・フォーマットを問い合わせます。

グループ： WAVEFORM

関連コマンド： WFMPre:PT_OFF、WFMPre:XINCR、WFMPre:XMULT、WFMPre:XZERO、WFMPre:XOFF、WFMPre:YMULT、WFMPre:YZERO、WFMPre:YOFF

シンタックス： WFMPre:PT_FMT {Y | XY | YZ | XYZ | ENV}
WFMPre:PT_FMT?



アーギュメント： Y — Y 軸成分のデータを転送します。完全な X 軸成分と Y 軸成分のデータは、転送データの各ポイント yn,yn+1,yn+2 ... に対して、次式で与えられます。

Xn = <XZERO-value> + <XINCR-value> × (n − <PT_OFF-value>)
Yn = <YZERO-value> + <YMULT-value> × (yn − <YOFF-value>)

なお、完全な X 軸成分と Y 軸成分のデータへの具体的な変換方法については、本章の“波形転送について”を参照ください。

XY — X 軸・Y 軸成分の値を転送します。
YZ — Y 軸・Z 軸成分の値を転送します。
XYZ — X 軸・Y 軸・Z 軸成分の値を転送します。
ENV — 各データ・ポイント毎に、最大値と最小値の 2 つの Y 軸成分を転送します。

本機器は、Y フォーマットのみを取り扱います。Y (デフォルト) 以外の選択を行っても無視されます。

使用例： 以下は、:WFMPRE:PT_FMT? のレスポンス例です。

:WFMPRE:PT_FMT Y

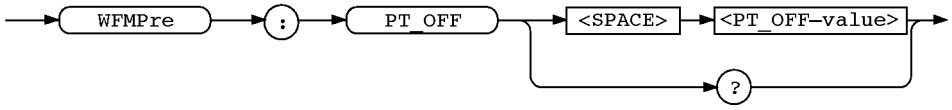
WFMPre:PT_OFF (?)

WFMPre:PT_OFF コマンドは、X軸の各データ・ポイントに適用するポイント・オフセット値を定義します。また WFMPre:PT_OFF? 問い合わせコマンドは、X軸のポイント・オフセット値を問い合わせます。

グループ： WAVEFORM

関連コマンド： WFMPre:PT_FMT、WFMPre:XINCR、WFMPre:XZERO

シンタックス： WFMPre:PT_OFF　<PT_OFF-value>
WFMPre:PT_OFF?



アーギュメント： <PT_OFF-value>::=<NR1>

本機器では、0 (デフォルト) 以外の値を指定しようとしても無視されます。<PT-OFF-value> の使用方法については、WFMPre:PT_FMT コマンドを参照ください。

使用例： 以下は、:WFMPRE:PT_OFF? のレスポンス例です。

```
:WFMPRE:PT_OFF  0
```

WFMPre:XINCR　(?)

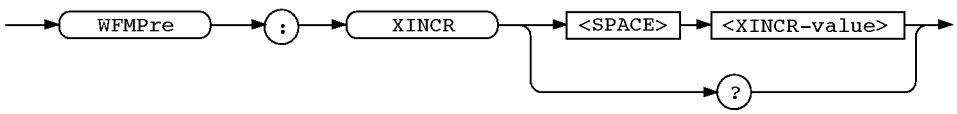
WFMPre:XINCR コマンドは、X 軸データ・ポイントのインクリメント値を定義します。また WFMPre:XINCR? 問い合わせコマンドは、設定中の X 軸データ・ポイント・インクリメント値を問い合わせます。

なお、インクリメント値の設定は、DATA:DESTination コマンドで設定される波形ファイルの転送先に対して行なわれます。、またインクリメント値の参照は、DATA:SOURce コマンドで設定される波形ファイルの転送元に対して行なわれます。

グループ： WAVEFORM

関連コマンド： WFMPre:PT_FMT、WFMPre:PT_OFF、WFMPre:XZERO、WFMPre:XUNIT、DATA:DESTination、DATA:SOURce

シンタックス： WFMPre:XINCR　<XINCR-value>
WFMPre:XINCR?



アーギュメント： <XINCR-value>::=<NR3>

設定範囲： 5E−8 ～ 1E−1	(AWG2005型)
設定範囲： 1E−8 ～ 1E−1	(AWG2010/11型)
設定範囲： 4E−9 ～ 1E−1	(AWG2020/21型)
設定範囲： 9.7646E−10 ～ 1E−3	(AWG2040/41型)

<XINCR-value> の使用方法については、WFMPre:PT_FMT コマンドを参照ください。

使用例： 以下は、X 軸データ・ポイント・インクリメント値を 0.01 に設定する例です。

:WFMPRE:XINCR 0.01

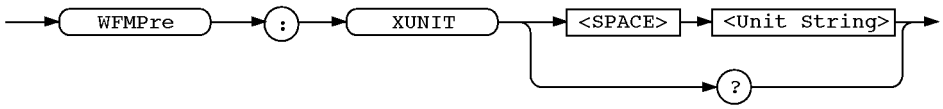
WFMPre:XUNIT (?)

WFMPre:XUNIT コマンドは、X軸データの単位を表現する文字列を定義します。また WFMPre:XUNIT? 問い合わせコマンドは、設定中のX軸データの単位表現を問い合わせます。

グループ： WAVEFORM

関連コマンド： WFMPre:PT_OFF、WFMPre:XINCR、WFMPre:XZERO

シンタックス： WFMPre:XUNIT <Unit String>
WFMPre:XUNIT?



アーギュメント： <Unit String>::=<文字列>

本機器では、X 軸は常に “時間” を表します。よって単位表現は “S” になりますので、他の文字列が設定されても無視されます。

使用例： 以下は、:WFMPRE:XUNIT? のレスポンス例です。

:WFMPRE:XUNIT ”S”

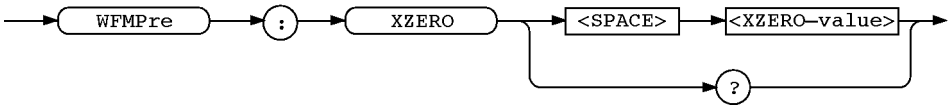
WFMPre:XZERO (?)

WFMPre:XZERO コマンドは、X軸の原点オフセット値を定義します。また WFMPre:XZERO? 問い合わせコマンドは、設定中のX軸原点オフセット値を問い合わせます。

グループ： WAVEFORM

関連コマンド： WFMPre:PT_OFF、WFMPre:XUNLT、WFMPre:XINCR

シンタックス： WFMPre:XZERO　<XZERO-value>
WFMPre:XZERO?



アーギュメント： <XZERO-value>::=<NR2>

本機器では、X 軸の原点オフセットは常に 0.0 (デフォルト) に設定されなければなりません。他の値を設定しようとしても無視されます。<XZERO-value> の使用方法については、WFMPre:PT_FMT コマンドを参照ください。

使用例： 以下は、:WFMPRE:XZERO? のレスポンス例です。

:WFMPRE:PT_OFF 0.0

WFMPre:YMULT　(?)

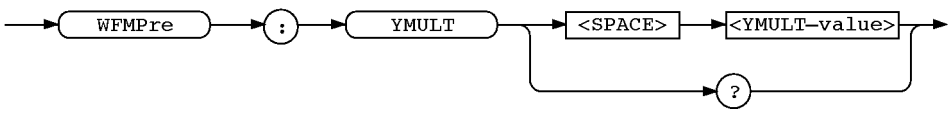
WFMPre:YMULT コマンドは、Y 軸のデータ・ポイントに対する乗数を定義します。また WFMPre:YMULT? 問い合わせコマンドは、設定中の Y 軸データ・ポイントに対する乗数を問い合わせます。

なお、乗数の設定は、DATA:DESTination コマンドで設定される波形ファイルの転送先に対して行なわれます。、また乗数の参照は、DATA:SOURce コマンドで設定される波形ファイルの転送元に対して行なわれます。

グループ： WAVEFORM

関連コマンド： WFMPre:YOFF、WFMPre:YZERO、WFMPre:YUNIT、DATA:DESTination、DATA:SOURce

シンタックス： WFMPre:YMULT　<YMULT-value>
WFMPre:YMULT?



アーギュメント： <YMULT-value>::=<NR3>

<YMULT-value> の使用方法については、WFMPre:PT_FMT コマンドを参照ください。

使用例： 以下は、Y 軸のデータ・ポイントに対する乗数を 0.0012 と定義する例です。

```
:WFMPRE:YMULT  0.0012
```

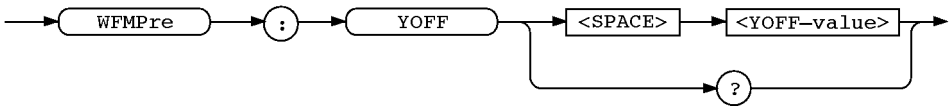
WFMPre:YOFF　(?)

WFMPre:YOFF コマンドは、Y 軸の各データ・ポイントに対するオフセット値を定義します。また WFMPre:YOFF? 問い合わせコマンドは、設定中の Y 軸データ・ポイントに対するオフセット値を問い合わせます。

グループ： WAVEFORM

関連コマンド： WFMPre:YMULT、WFMPre:YZERO、WFMPre:YUNIT

シンタックス：
```
WFMPre:YOFF　<YOFF-value>
WFMPre:YOFF?
```



アーギュメント： <YOFF-value>::=<NR3>

本機器ではデータ幅が 1 バイトのときは 127 を、2 バイトのときは 2047 を Y 軸データ・ポイントに対するオフセットと定義しています。よって、これ以外の値が設定されても無視されます。<YOFF-value> の使用方法については、WFMPre:PT_FMT コマンドを参照ください。

使用例： 以下は、:WFMPRE:YOFF? のレスポンス例です。

```
:WFMPRE:YOFF  2.047E+03
```

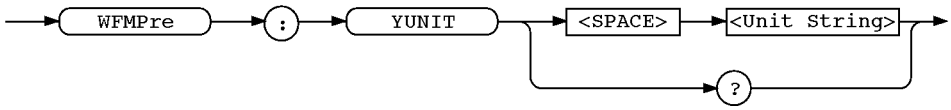
WFMPre:YUNIT　(?)

WFMPre:YUNIT コマンドは、Y 軸データの単位を表現する文字列を設定します。また WFMPre:YUNIT? 問い合わせコマンドは、設定中の Y 軸データの単位表現を問い合わせます。

グループ： WAVEFORM

関連コマンド： WFMPre:YMULT、WFMPre:YZERO、WFMPre:YOFF

シンタックス：
```
WFMPre:YUNIT　<Unit String>
WFMPre:YUNIT?
```



アーギュメント： <Unit String>::=<文字列>

本機器では、Y 軸は常に “電圧” を表します。よって単位表現は “V” になりますので、他の文字列が設定されても無視されます。

使用例： 以下は、:WFMPRE:YUNIT? のレスポンス例です。

```
:WFMPRE:YUNIT  ”V”
```

WFMPre:YZERO　(?)

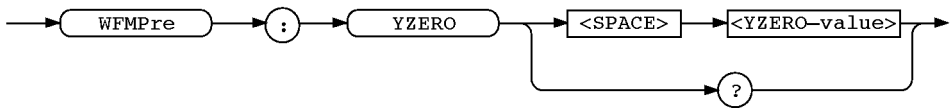
WFMPre:YZERO コマンドは、Y 軸の原点オフセット値を定義します。また WFMPre:YZERO? 問い合わせコマンドは、設定中の Y 軸原点オフセット値を問い合わせます。

なお、オフセット値の設定は、DATA:DESTination コマンドで設定される波形データの転送先に対して行なわれます。またオフセット値の参照は、DATA:SOURce コマンドで設定される波形データの転送元に対して行なわれます。

グループ： WAVEFORM

関連コマンド： WFMPre:PT_OFF、WFMPre:YMULT、WFMPre:YUNIT、WFMPre:YOFF、DATA:DESTination、DATA:SOUce

シンタックス： WFMPre:YZERO　<YZERO-value>
WFMPre:YZERO?



アーギュメント： <YZERO-value>::=<NR2>
設定範囲：−5.000 V ~ 5.000 V、ステップ 0.005 V　(AWG2005型)
設定範囲：−2.500 V ~ 2.500 V、ステップ 0.005 V　(AWG2010/11/20/21型)
設定範囲：−1.000 V ~ 1.000 V、ステップ 0.001 V　(AWG2040/41型)

<YZERO-value> の使用方法については、WFMPre:PT_FMT コマンドを参照ください。

使用例： 以下は、Y 軸の原点オフセット値を 0.225 V に設定する例です。

```
:WFMPRE:YZERO  0.225
```

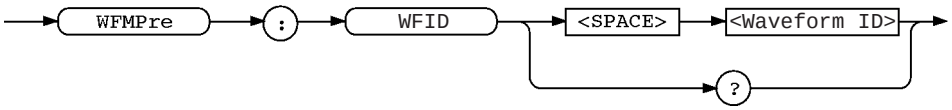
WFMPre:WFID (?)

WFMPre:WFID コマンドは、転送データの ID 情報を設定します。

グループ： WAVEFORM

関連コマンド：

シンタックス： WFMPre:WFID <Waveform ID>
WFMPre:WFID?



アーギュメント： ID 情報は、本機器が自動的に設定しますので、ユーザによる設定は無視されます。

レスポンス： 使用例を参照ください。

使用例： 以下は、:WFMPRE:WFID? のレスポンス例です。

:WFMPRE:WFID ”WAVEFORM.WFM, 1000 POINTS, CLOCK: 100.0MHZ, AMPLITUDE: 1.000V, OFFSET: 0.000V”

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 2-5 に、それぞれの場合の概要を示します。

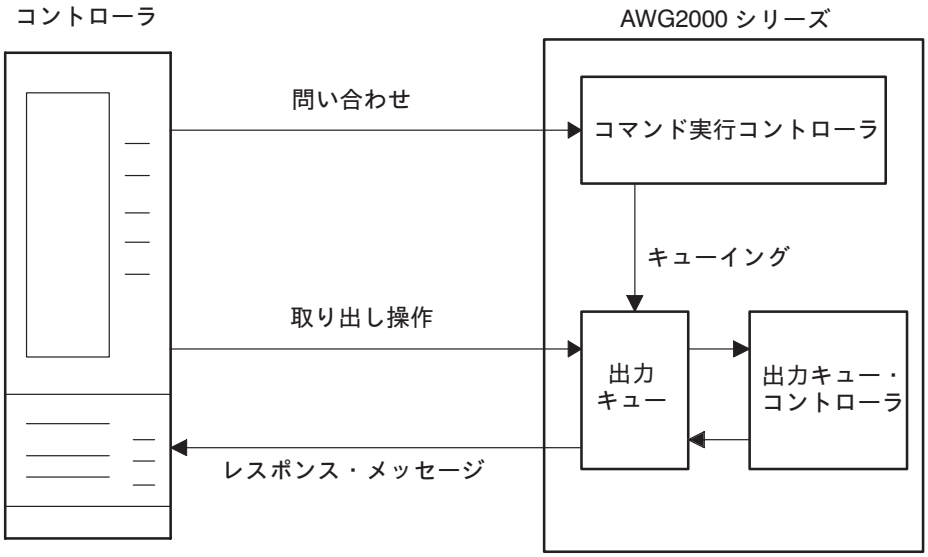


図 2-5 : GPIB:レスポンス・メッセージの取り出し

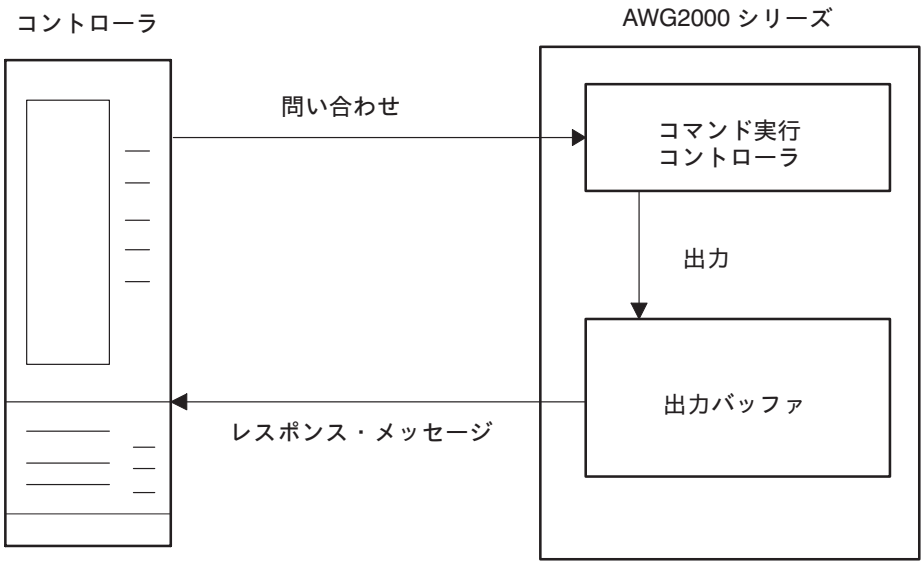


図 2-6 : RS-232C:レスポンス・メッセージの取り出し

図 2-5 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い合わせコマンドが転送されると、本機器は、問い合わ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。「第 4 章のプログラム例」の中では、awgRead() サポート関数によってレスポンス・メッセージの取り出し操作を行っています。取り出し操作の詳細は、第 4 章の「プログラム例」および「サポート関数」を参照ください。

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い合わ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い合わ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 ビットを使って調べることができます。出力キュー、SBR、あるい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」を参照ください。

図 2-6 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い合わ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。従って、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、問い合わ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ることになります。

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、次々に問い合わ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。

波形転送について

波形転送機能は、本機器と外部コントローラとの間で波形の転送を行う機能です。この機能を使用すると、本機器で作成した波形を外部コントローラで保存したり、外部コントローラからさらに他の機器に転送したり、あるいは外部コントローラで新規に作成した波形や一部を修正した波形を本機器に戻したりすることができます。

波形の転送は、Tektronix Std Codes and Formats の波形フォーマット仕様に従って行われます。以下で、本機器と外部コントローラの間で波形転送を行うための方法を説明します。

本機器にはさらに、GPIB インタフェースを使用し、当社デジタル・オシロスコープ等との間で直接波形の転送を行う、ダイレクト波形転送機能が用意されています。使用方法については、ユーザ・マニュアルを参照ください。

なお本機器は、外部コントローラとの間で、イクエーションやマーカ・データの転送も行えます。イクエーションの転送については EQUAtion:DEFine コマンドの項を、マーカ・データの転送については MARKer:DATA コマンドの項をそれぞれ参照ください。

ソースとデスティネーション

波形の転送に先立って、ソースまたはデスティネーションを指定します。

ソースは、本機器から外部コントローラに波形またはマーカ・データを転送する際の波形の転送元を意味します。本機器が外部に転送できる波形またはマーカ・データは、波形メモリにロードされたもの、あるいは内部メモリにある波形ファイルに保存されたものでなければなりません。ソースの指定は、DATA:SOURce コマンドで行ってください。

デスティネーションは、外部コントローラから本機器に波形またはマーカ・データを転送する際の波形の転送先を意味します。転送先は、本機器の内部メモリにある波形ファイルに限られます。指定の波形ファイルが内部メモリに無い場合には新規に波形ファイルが作成され、既に同じ名前のファイルが存在する場合には上書きされます。なおソースの指定は、DATA:DESTination コマンドで行ってください。

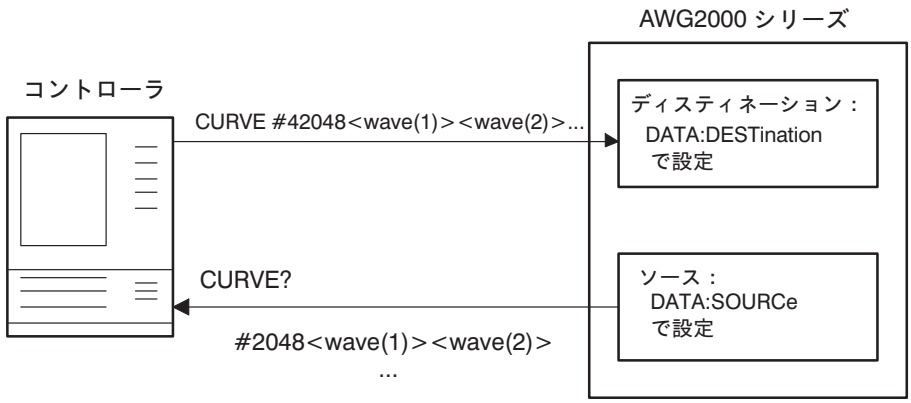
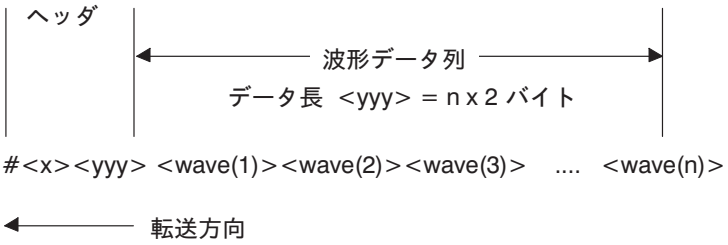


図 2-7：ソースとデスティネーション

プリアンブルとカーブ

転送波形は、プリアンブルとカーブのデータで構成されます。プリアンブルには、カーブ・データのサイズ、スケール、フォーマットなどの情報や、波形 ID や単位などの補助情報が含まれます。一方カーブは、アンスケールの波形データ列で、波形メモリに記述されるべきデータを表しています。このアンスケールの波形データとプリアンブルの情報をもとに、スケールされた完全なデータを得ることができます。

カーブは、CURVE? 問い合わせコマンドにより、下記のようなアービトラリ・ブロック形式のレスポンス・メッセージとして本機器から外部コントローラに転送されます (レスポンス・ヘッダが OFF の場合)。また下記の形式のアービトラリ・ブロックを CURVE コマンドのアーギュメントとすることにより、外部コントローラから本機器にアンスケールの波形データを転送できます。



ここで <yyy> は、後に続く波形データ列のバイト数 (ASCII 形式) を、<x> は、<yyy> の桁数 (ASCII 形式) を、また <wave(i)> は、i 番目の波形データを表します。i番目のデータ・ポイント (X(i)、Y(i)) は、次の式に従って、スケールされた波形データに変換できます。

$$X(i) = i \times \langle XINCR\text{−}value \rangle$$
$$Y(i) = \langle YZERO\text{−}value \rangle + (\langle wave(i) \rangle - \langle YOFF\text{−}value \rangle) \times \langle YMULT\text{−}value \rangle$$

<XINCR–value>、<YZERO–value>、<YOFF–value>、<YMULT–value> は、次の表ようなプリアンブルに含まれる情報です。波形データを読み出す場合には、問い合わせコマンドによって、本機器からこれらの情報を引き出すことができます。また波形データを本機器に転送する場合には、コマンドによって、これらの情報を本機器に設定します (ただし <YOFF–value> は、2047 (データ幅が 2)、または 127 (データ幅が 1) 以外の設定ができません)。

パラメータ	コマンド	意　味
<XINCR–value>	WFMPre:XINCR	X 軸データ・ポイント・インクリメント
<YZERO–value>	WFMPre:YZERO	Y 軸原点オフセット
<YOFF–value>	WFMPre:YOFF	Y 軸データ・ポイント・オフセット (2047または127)
<YMULT–value>	WFMPre:YMULT	Y 軸データ・ポイント乗数

各データ・ポイント <wave(i)> は、データ幅2バイト、有効ビット数 12 ビットまたはデータ幅1バイト、有効ビット数8ビットの符号なし整数コードで転送されます。データ幅2バイトで転送の際には、バイト・オーダ(上位バイトまたは下位バイトのどちらを先に転送するか) を、WFMPre:BYT_OR コマンドまたは DATA:ENCDG コマンドで指定することができます。

バイト・オーダの指定は、使用する外部コントローラの CPU が Little−Endian または Big−Endian 方式かに従って使い分けると、データをメモリに取り込み易くなります。例えば、PC-9800 シリーズや IBM-PC を外部コントローラに使用している場合には、下位バイトを先に転送するように設定しておきます。詳しくは、コマンドの詳細説明を参照ください。

X 軸と Y 軸は、それぞれ時間 (S) と電圧 (V) で表されます。なお転送波形データのフォーマットや関連情報(プリアンブル)は、WAVEFORM コマンド・グループの WFMPre をルート・ヘッダ・ニーモニックに持つコマンド、問い合わせコマンドでそれぞれ設定、参照ができます。

データ転送手順

以下に本機器から外部コントローラへ、あるいは外部コントローラから本機器に波形を転送するための手順例を示します。

本機器から外部コントローラへの転送

☐ **手順 1**：ソースを指定します。

```
DATA:SOURCE　”CH1”
```

上記は、チャンネル 1 にロードされた波形を指定する例です。また次は、波形ファイルを指定する例です。

```
DATA:SOURCE　”SAMPLE–1.WFM”
```

☐ **手順 2**：波形データ・ポイントのデータ幅とバイト・オーダを指定します。次の例は、データ幅 2 で下位バイトを先に転送する例です。

```
DATA:WIDTH 2; ENCDG SRPBINARY
```

上位バイトを先に転送する場合には、SRPBINARY に代えて RPBINARY を指定します。また同じ指定を WFMPRE:BYT_OR コマンドで行うこともできます。

☐ **手順 3**：レスポンス・ヘッダを OFF にします。

```
HEADER　OFF
```

☐ **手順 4**：次のプリアンブル・データを読み込み、バイナリ形式に変換してからメモリに格納しておきます (手順 7 参照)。

a. データの数 <NR_PT> を読みます。

```
WFMPRE:NR_PT?
```

b. Y 軸原点オフセット <YZERO–value> を読みます。

```
WFMPRE:YZERO?
```

c. Y 軸データ・ポイントの乗数 <YMULT–value> を読みます。

```
WFMPRE:YMULT?
```

d. X 軸データ・ポイントのインクリメント値 <XINCR–value> を読みます。

```
WFMPRE:XINCR?
```

☐ **手順 5**：波形データの転送開始を指示します。

```
CURVE?
```

☐ **手順 6**：出力キューから波形データを読み取ります。RS-232C インタフェースを使用している場合には、出力キューがありませんので、ただちに波形データが返送される点に注意してください。

a. アービトラリ・ブロックのヘッダ部を読み取ります。

b. 波形データを配列に読み込みます。波形データは、データ数が <NR.PT> ですので、メモリとして <NR.PT> 個のデータ幅長分の 1 次元配列が必要となります。データ幅の設定が 2 の場合、データ・ポイントのバイト・オーダは、外部コントローラの CPU に依存して決定されなければなりませんが、CPU に依存させないようにするには、1 バイトずつデータを読み取り、2 バイトのデータに再構成する方法を用いてください。

☐ **手順 7**：スケール波形データに変換します。i 番目のデータ・ポイント (X(i)、Y(i)) は、次のように変換します。なお wave(i) は、アンスケール波形データの i 番目の要素です。

$$X(i) = i \times \text{<XINCR−value>}$$
$$Y(i) = (\text{wave}(i) - \text{<YOFF−value>}) \times \text{<YMULT−value>}$$

☐ **手順 8**：レスポンス・ヘッダを ON に戻します。

```
HEADER ON
```

以上で、本機器から外部コントローラへの波形ファイルの転送は終了です。

外部コントローラから本機器への転送

☐ **手順 1**：デスティネーションを設定します。

DATA:DESTINATION　”SAMPLE–1.WFM”

上記の例は、内部メモリ上の波形ファイル SAMPLE−1.WFM を指定する例です。

☐ **手順 2**：波形データ・ポイントのデータ幅とバイト・オーダを指定します。

DATA:WIDTH 2; ENCDG　SRPBINARY

上記の例は、データ幅 2 で下位バイトを先に転送する例です。上位バイトを先に転送する場合には、SRPBINARY に代えて RPBINARY を指定します。また同じ指定を WFMPRE:BYT_OR コマンドで行うこともできます。

☐ **手順 3**：次のプリアンブル・データを設定します。

a. Y 軸原点オフセット <YZERO–value> を設定します。

例：　WFMPRE:YZERO　0.0

b. Y 軸データ・ポイントの乗数　<YMULT–value> を設定します。

例：　WFMPRE:YMULT　4.8E–04

c. X 軸データ・ポイントのインクリメント値　<XINCR–value> を設定します。

例：　WFMPRE:XINCR　5.0E–9

プリアンブル情報を設定しない場合には、デフォルト値が設定されます(手順 1 のソースの設定で、内部メモリに存在する波形ファイルを指定した場合には、そのファイルに記録されたプリアンブル情報がデフォルト値となります)。

☐ **手順 4**：波形データを転送します。

CURVE　#42048 <wave(1)><wave(2)> ... <wave(1024)>

以上で、外部コントローラから本機器への波形ファイルの転送は終了です。

第 3 章
ステータス／イベント

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オペレータに知らせることができます。

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。以下、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。

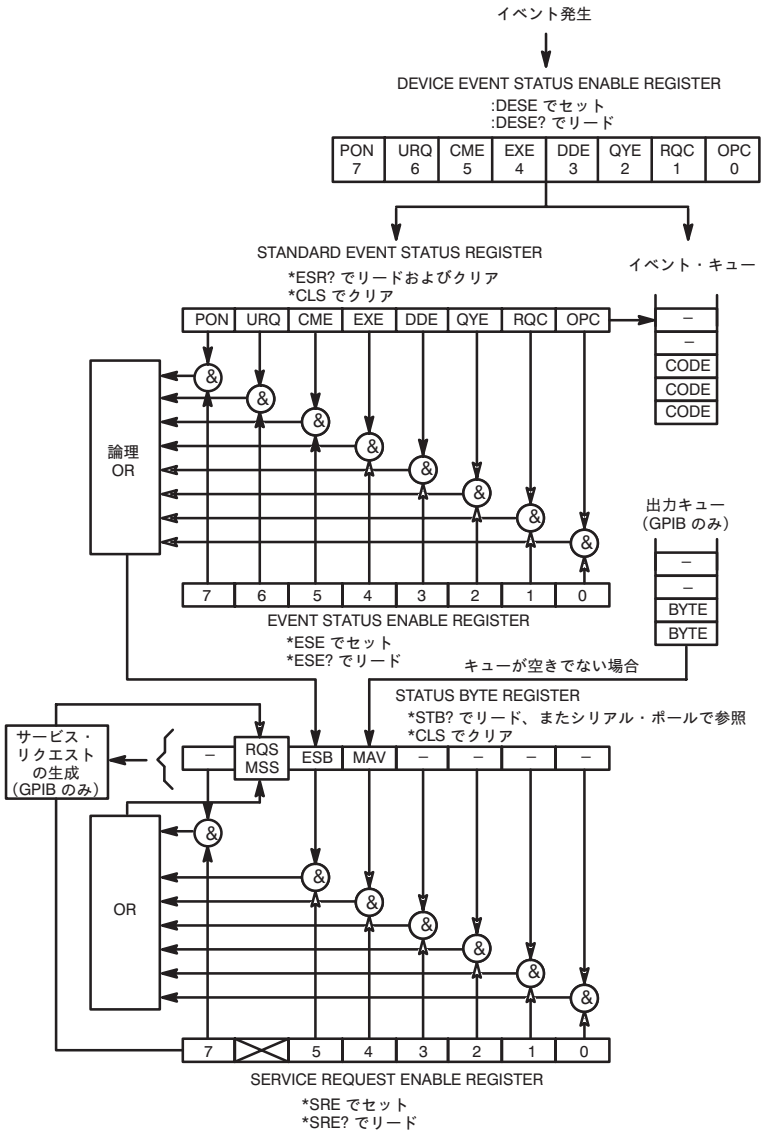


図 3-1：ステータス／イベント処理シーケンス概要

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ますので、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の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ませんので常に 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、機器が少なくても 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・ポールの機能がありませんので、本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、各ビットに以下で説明する 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ませんので、本ビットは無効です。

- **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全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、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 の内容と全く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、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。例えば、DESER の全てのビットを 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 の内容と全く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。例えば、ESER の全てのビットを 0 にリセットした場合、いかなる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。本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 *SRE? 問い合わせコマンドで SRER の内容を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りますので、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。よって 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。イベントの数が 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全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。

イベントを取り出す前に、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。しかしイベント・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となります。

例：
```
*ESR?
ALLEV?
```

ステータス／イベント処理シーケンス

図 3-2 は、ステータス/イベントの処理の流れを表しています。

イベントが発生すると、まず DESER の内容を調べます。もしイベントに対応する DESER のビットがセットされていれば、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。なお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。

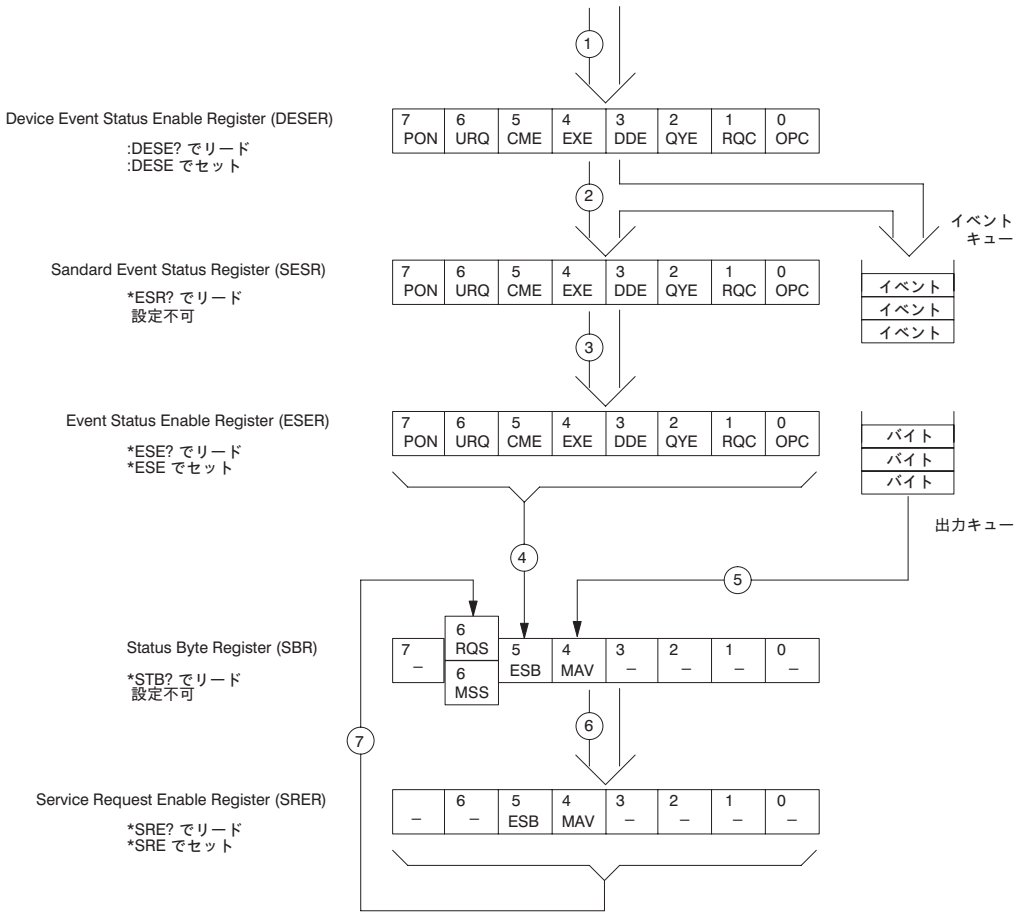


図 3-2：ステータス／イベントの処理過程

I/Oステータス・スクリーン

イベント／ステータス・レポーティング・システムのレジスタとキューの内容は、本機器の管面に表示できます。図 3-3 は、管面に表示されたイベント／ステータス・レポーティング・システムの内容です。以下の手順で表示できます。

☐ **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**：**Misc** ボトム・キーを押して **Misc** サイド・メニューを表示します。

☐ **手順 3**：**Status...** サイド・キーを押して、**Status** サブ・メニューの内容を管面に表示します。

☐ **手順 4**：**I/O** サイド・キーを押して、**I/O** サブ・メニューを管面に表示します。

図 3-3 で DESER、SESR、ESER、SBR、SRER の各レジスタの内容は、[ ] に囲まれた 10 進数の値と共に表示されます。イベント・キューから取り出し可能状態にあるイベントは、**Event Queue** の左側の **Avail** に表示されます。またイベント・キューにスタック中のイベントは、**Event Queue** の右側の **Pend** に表示されます。

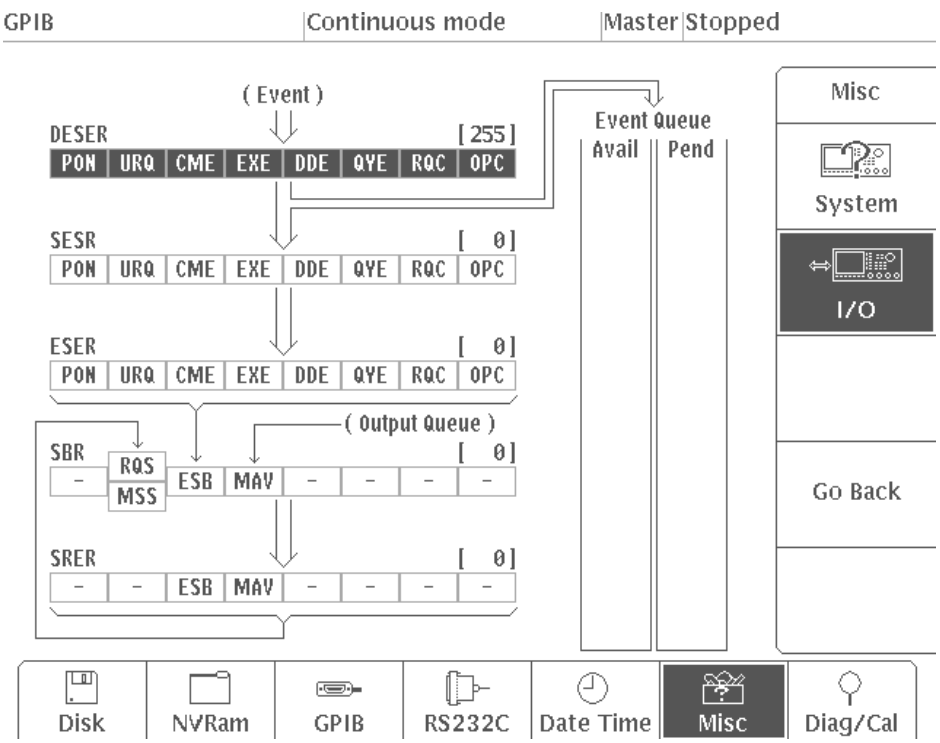


図 3-3：ステータス／イベント・レポーティング・スクリーン

実行の同期について

本機器の GPIB コマンドは、外部コントローラから本機器に転送された順番に実行処理されるように設計されています。ただし、いくつかのコマンドはその実行を完了するのにある程度の時間を必要とするため、その実行終了を待たずして、次に転送されるコマンドが同時に実行できるように設計されています。このようなコマンドの場合、実行が完了するのを待ってから次のコマンドを実行しなければならないことがあります。

以下のコマンドは実行が完了する前に他のコマンドも同時に実行できるコマンドです。

```
EQUAtion:COMPile[:STATe EXECute,]<Equation File>
HCOPy STARt
```

本機器には、同期の制御を行うため形式的に、以下のコマンドが用意されています。

```
*WAI
*OPC
*OPC?
```

*WAIコマンド

一般に *WAI コマンドを使用すると最も簡単に実行の同期を取ることができます。同期をとるには以下の例のように *WAI コマンドを転送してから次のコマンドを転送するだけです。

```
:EQUATION:COMPILE:STATE EXECUTE  ”SAMPL.EQU”;
*WAI;:CH1:WAVEFORM  ”SAMPL.WFM”
```

*OPCコマンドを使用する方法

*OPC コマンドは、ペンディング中の全ての処理が完了すると、SESR (Standard Event Status Register) の OPC ビットをセットします。このコマンドは、シリアル・ポールかサービス・リクエストの機能と一緒に使用することによって実行終了の監視を最も効果的に行うことができるようになります。以下の例に示すように、どちらの方法を使用しても、ほとんど同じシーケンスで処理できます。

対応するステータス・レジスタをイネーブルにします。

:DESE 1
*ESE 1

*SRE 0 （シリアル・ポールを使用する場合）

または

*SRE 32 （サービス・リクエストを使用する場合）

例えば、イクエーション・ファイルのコンパイルを開始します。さらに続けて、コンパイルの終了を待ちます。

:EQUATION:COMPILE ”SAMPL.EQU” ; *OPC
(serial poll が 0 の間待ちます。または、サービス・リクエストが発生するのを待ちます)

コンパイルによって生成された波形ファイルをチャンネル1に設定します。

:CH1:WAVEFORM ”SAMPL.WFM”

なお上記の具体的な使用方法については、第 4 章の「プログラム例 3 」を参照ください。

*OPC？問い合わせコマンド

*OPC? 問い合わせコマンドは、ペンディング中の全ての処理が終了すると、レスポンスとしてASCIIコードの “1” を戻します。実行終了の監視は、以下の例のように行うことができます。

例えば、イクエーション・ファイルのコンパイルを開始します。さらに続けて、コンパイルの終了を待ちます。

:EQUATION:COMPILE ”SAMPL.EQU” ; *OPC?
(レスポンスとして “1” が戻されるのを待ちます。GPIB インタフェースの場合、出力キューからデータの取り出しを待っていると、データが出力キューに書き込まれる前にタイム・アウトになることもあります。)

コンパイルによって生成された波形ファイルをチャンネル 1 に設定します。

:CH1:WAVEFORM ”SAMPL.WFM”

メッセージ

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表 3-2 から表 3-8 に示します。

たいていのメッセージでは、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、セミコロンに続いて、二次メッセージ（本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の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無い状態
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無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

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
152	String data too long
158	String data not allowed
160	Block data error
161	Invalid block data
168	Block data not allowed
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
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

表 3-4：実行エラー（EXEビット：4）（続き）

コード	メッセージ
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
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

第４章　プログラム例

プログラム例

はじめに

本章では、サンプル・プログラムを通し、GPIB インタフェースを使用して本機器を制御するための具体的な方法を示します。

本マニュアルに添付のフロッピー・ディスクには、Microsoft QuickC 2.0 および Microsoft Quick BASIC 4.5 で取り扱えるサンプル・プログラムのソース・ファイルが含まれていますので、このソース・ファイルをコンパイル、リンクすることにより実行可能なコマンドとして利用できます。またこのソースを変更したり、さらに大きなプログラムに発展させることもできます。

サンプル・プログラムは、NECのPC-9800 シリーズまたは IBM PC 互換機に National Instruments 社の GPIB ボードと関連のドライバがインストールされたシステムの上で動作します(ただしフロッピー・ディスクは、PC-9800 用フォーマットになっています)。また GPIB システムは、本機器、外部コントローラの機器名がそれぞれ、DEV1、GPIB0、に設定されているものとしています。

サンプル・プログラムには、以下のものが含まれます。

getwfm 本機器から波形を読み出し、プリアンブルと共に波形データを外部コントローラのファイルに格納したり、あるいは絶対スケールを持つ波形データに変換して画面に表示したりします。

putwfm getwfmでファイルにセーブされた波形データとプリアンブルを本機器に戻します。

equset イクエーション・データを本機器に転送し、コンパイルします。コンパイル後、得られた波形をチャンネル 1 に設定して出力します。このプログラムは、*OPC 共通コマンドとシリアル・ポールを使用してコンパイルの終了を確認してから次の設定に進みます。

intrv 外部コントローラからインタラクティブにコマンドを送信して、本機器を制御します (Quick C 用のみ)。

サンプル・プログラムのコンパイル

フロッピー・ディスクで提供されるソース・ファイルは、QuickC 2.0 または Quick BASIC 4.5 で取り扱うことができます。

ソース・ファイルと MAKE ファイルは、それぞれのディレクトリに置かれていますので、Quick C 用と Quick BASIC 用の使用するコンパイラに合わせてこれらのファイルをハード・ディスクにコピーしてから使用してください。

Quick C のサンプル　　\C
Quick BASIC のサンプル　\BASIC

以下の手順に従って、実行形式のプログラム（コマンド）を作ります。

Quick Cの場合

☐ **手順 1**：QuickC をインストールします。SMALL サイズのメモリ・モデルを使用します。コマンドのパスを QuickC が実行できるように設定してください。

☐ **手順 2**：National Instruments 社の PC2/PC2A ボードと関連のドライバをインストールします。外部コントローラには、IBCONF.EXE で、本機器を DEV1、外部コントローラを GPIB0 として認識できるように設定してください。

☐ **手順 3**：フロッピー・ディスクからハード・ディスクにファイルをコピーします。例えば、ハード・ディスク・ドライバをC、フロッピー・ディスク・ドライバをBとします。Cドライバ上で作業中であるとすれば、以下のように行います。

```
mkdir  examples
cd  example
copy  B:\C\*.*  .
```

☐ **手順 4**：さらに National Instruments 社の PC2/PC2A GPIB ドライバのディレクトリから decl.h と mcibs.obj をコピーします。

```
copy  \gpib–pc\decl.h  .
copy  \gpib–pc\mcibs.obj  .
```

なお PC‐9800 用の場合は、mcibs.obj の代わりに mcib.obj を使用することができます。

☐ **手順 5**：MAKE ファイルを使って、サンプル・プログラムをコンパイル、リンクします。

```
nmake  /F  samp.mak
```

Quick BASIC の場合

☐ **手順 6**：Quick BASIC をインストールします。コマンドのパスを Quick BASIC が実行できるように設定してください。

☐ **手順 7**：National Instruments 社の PC2/PC2A ボード（PC－9801 用の場合は、GPIB－98 Turbo）ボードと関連のドライバをインストールします。外部コントローラには、IBCONF.EXE で、本機器をDEV1、外部コントローラを GPIB0 として認識できるように設定してください。

☐ **手順 8**：フロッピー・ディスクからハード・ディスクにファイルをコピーします。例えば、ハード・ディスク・ドライバを C、フロッピー・ディスク・ドライバを B とします。C ドライバ上で作業中であるとすれば、以下のように行います。

```
mkdir  examples
cd  examples
copy  B:\basic\*.*  .
```

☐ **手順 9**：さらにNational Instruments社のPC2/PC2A GPIBドライバのディレクトリからgpib.objとgpdecl.basをコピーします。

```
copy  \gpib–pc\qbasic\gpib.obj  .
copy  \gpib–pc\qbasic\gpdecl.obj  .
```

そして、Quick BASIC のディレクトリから bc.exe、link.exe、bcom45.lib（PC－9801 用の場合は、bcom45a.lib）をコピーします。

```
copy  \gpib–pc\bin\bc.exe  .
copy  \gpib–pc\bin\link.exe  .
copy  \gpib–pc\bin\bcom45.lib  .
```

☐ **手順 10**：BAT ファイルを使って、サンプル・プログラムをコンパイル、リンクします。

```
makeexe.bat
```

サンプル・プログラムの実行

サンプル・プログラムの実行は、以下のように行います。

Getwfm

getwfm <source> [<out–file>]

このプログラムは、本機器から波形を読み出し、プリアンブルと共に波形データを外部コントローラのファイルに格納したり、あるいは絶対スケールを持つ波形データに変換して画面に表示したりします。第一アーギュメント <source> は、転送すべき波形が格納された内部メモリ上の波形ファイルを示します。第二アーギュメント <out–file> は、外部コントローラで読み出された波形データとプリアンブルをセーブするファイルのファイル名です。この <out–file > が省略された場合には、ファイルにセーブされることなく、プリアンブルのデータから絶対スケールを持つ波形データに変換されて画面に表示されます。なおファイルにセーブされた波形データは、putwfm プログラムにより本機器に戻すことができます。

Putwfm

putwfm <destination> <waveform–file>

このプログラムは、getwfm でファイルにセーブされた波形データとプリアンブルを本機器に戻します。第一アーギュメント <destination> は、転送される波形データを格納する波形ファイルで、本機器の内部メモリに生成されるファイルあるいは内部メモリに既に存在するファイル（この場合ロックされていなければ上書きされます）でなければなりません。第二アーギュメント <waveform–file>は、外部コントローラ上のファイルで getwfm により波形データがセーブされたものです。

Equset

equset

このプログラムは、イクエーション・データを本機器に転送し、コンパイルします。コンパイル後、得られた波形をチャンネル 1 に設定して出力します。このプログラムは、*OPC 共通コマンドとシリアル・ポールを使用してコンパイルの終了を確認してから次の設定に進みます。

Intrv

intrv

このプログラムは、外部コントローラからインタラクティブにコマンドを送信して、本機器を制御します。プログラムが起動されると、プロンプト・メッセージ 'AWG >> ' が出力されてユーザからのコマンドの入力を待ちます。このプログラムに対しては、以下で説明するように、本プログラマ・マニュアルで定義される全てのコマンドとこのプログラムの内部コマンド（ビルトイン・コマンド）が使用できます。なお、本プログラムは Quick C 用のみです。

- GPIBコマンド： 本プログラマ・マニュアルで定義される全てのコマンドと問い合わせコマンドが利用できます。問い合わせコマンドに対するレスポンスは、ただちに読み出されて画面に表示されます。またエラーが発生した場合にも、イベント・キューからイベント・コードとイベント・メッセージが読み出されて画面に表示されます。

- ビルトイン・コマンド： 以下のコマンドが使用できます。

  help　ヘルプ・メッセージを出力します。

  view　アーギュメントで指定される外部コントローラ上のファイル <path–name> の内容を画面に表示します。

  シンタックス： view <path–name>

  exec　外部コントローラ上のファイル <description–file> からコマンド・ラインを読みとり実行します。ファイルが終了すると標準入力に切り替わります。

  シンタックス： exec <description–file>

  status　本機器からステータス・バイトを読みとり、16 進数で表示します。

  resetes　イベント／ステータス・レポーティング・システムで使用するレジスタの内容を、このプログラムのディフォルトの状態にリセットします。このコマンドは、DESE、*ESE などでレジスタの内容を書き換えた後に使用します。

- リダイレクト： 標準入力または標準出力を次のシンタックスで切り替えることができます。

  < name

  ファイル name の内容を直接本機器に転送します。GPIB のコマンド・シーケンス・ファイルや getwfm プログラムでセーブしたファイルの内容を本機器に転送する際に使用します。

> name
標準出力の内容をファイル name に出力します。name で示されるファイルが既に存在していれば内容が上書きされます。存在してなければ新規に作成されます。

>> name
'>' の場合と同じですが、ファイルが既に存在していれば、後ろに追加されます。

■ その他：　コマンド・ラインに '!!' があれば、直前に入力したコマンド・ラインの内容と置き換えられます。

プログラム例

プログラム例1： 波形の転送（その1）

最初のプログラムは、本機器から外部コントローラに波形を転送する例です。このプログラムでは、波形データの転送方法と、絶対スケールを持つ波形データへの変換方法などが取り扱われています。

Quick Cの場合

```
/*
 * getwfm.c – a simple waveform transfer program that converts to
 * scaled waveform in ASCII format, or save raw waveform and preamble
 * to a file in the external controller.
 */
# include    <stdio.h>
# include    <stdlib.h>
# include    ”decl.h”
# include    ”exit.h”

# define     MAX_DATA2     4000
# define     MAX_DATA      (MAX_DATA2 / 2)
# define     CMD_LEN       80
# define     LEN12         12
# define     FILE_OUT      1
# define     STD_OUT       –1

typedef      float  FLOAT;
typedef      double DOUBLE;
typedef      long   LONG;
typedef      short  SHORT;

void   checkarg();
char   *awgWR();

SHORT  wfm[MAX_DATA + 1];  /* Array for raw waveform data input    */
LONG   nr_pt;              /* Preamble: number of data points      */
LONG   pt_off;             /* Preamble: point offset               */
FLOAT  yzero;              /* Preamble: Y origin - offset          */
FLOAT  yoff;               /* Preamble: Y offset                   */
FLOAT  ymult;              /* Preamble: Y multiple                 */
FLOAT  xincr;              /* Preamble: X increment                */
char   xunit[LEN12 + 1];   /* Preamble: X unit representation      */
char   yunit[LEN12 + 1];   /* Preamble: Y unit representation      */

char   *outfile;           /* Output file descriptor                     */
char   *source;            /* Source from which waveform is transferred  */
int    fflag = STD_OUT;
```

```
main(argc, argv)
int    argc;
char   *argv[];
{
       printf("\n\n");
       printf("GETWFM      –  simple waveform transfer program.\n");
       printf("Copyright (c) Tektronix Japan, Ltd.");
       printf(" All Rights Reserved.\n\n");

       checkarg(argc, argv);      /* Check for arguments and open output
                                                              device       */
       open_dev();                /* Find GPIB devices                    */
       SrcSetup();                /* Define source in the instrument      */
       if (fflag == STD_OUT)
              ReadandStdout();    /* Get waveform and convert to
                                     scaled waveform to display           */
       else
              ReadandFileout();   /* Get waveform and preamble, and
                                     then save into a file                */
       close_dev();
}

/*
 * Check whether the arguments are valid.
 */
void   checkarg(argc, argv)
int    argc;
char   *argv[];
{
       /*
        * Check for command line argument count.
        */
       if ((argc < 2)      || (argc > 3)){
              fprintf(stderr, "usage: getwfm <source> [<out-file>]\n");
              fprintf(stderr, "\twhere:\n");
              fprintf(stderr, "\t\t<source>\t\tis the source channel or");
              fprintf(stderr, " file to read.\n");
              fprintf(stderr, "\t\t[<outfile>]\tis the optional save");
              fprintf(stderr, " output file.\n");
              EXIT(1);
       }

       /*
        * Check for valid source channel, or waveform file name.
        */
       source = argv[1];
       if (strcmp(argv[1], "CH3") != 0 && strcmp(argv[1], "CH4") != 0 &&
          wfmfile(argv[1]) != 0)
       {
              fprintf(stderr, "ERROR: Invalid source: \"%s\":", argv[1]);
              fprintf(stderr, " No waveform.\n");
              EXIT(1);
       }
```

```
        /*
         * Open output file if specified otherwise use stdout.
         */
        if (argc == 3)
        {
                outfile = argv[2];
                fflag = FILE_OUT;
        }
}

/*
 * Check whether the file extension is '.wfm'.
 */
wfmfile(name)
char   *name;
{
        int    dlen = strlen(name);

        if (dlen < 4 || dlen > 12)
                return -1;
        if (strcmp(&name[dlen - 4], ".WFM") == 0 ||
            strcmp(&name[dlen - 4], ".wfm") == 0)
                return 0;
        return -1;
}

/*
 * Define source in the instrument, and set encoding format and byte order.
 *
 * WARNING - This program assumes that the CPU with Little-Endian is used
 * in the external controller, so that the byte order in waveform transfer
 * is set to SRPBINARY with :DATA:ENCDG command.  If the CPU with Big-Endian
 * is used, byte order must be set to RPBINARY.
 */
SrcSetup()
{
        char cmd[CMD_LEN + 1];

        sprintf(cmd, ":DATA:SOURCE \"%s\";ENCDG SPBINARY;WIDTH2" , source);
        if (awgWrite(cmd) < 0)
        {
                gpiberr("Write Error: Unable to setup waveform parameters");
                EXIT(1);
        }
}
```

```
/*
 * Read preamble and waveform, then save them into a file.
 */
ReadandFileout()
{
       if (awgWrite(":HEADER  ON") < 0)
       {
              gpiberr("Write Error: Unable to turn header on\n");
              EXIT(1);
       }
       if (awgWrite(":WAVFRM?") < 0)
       {
              gpiberr("Write Error: Unable to write :WAVFRM? query");
              EXIT(1);
       }
       if (WrtGtoF(outfile) < 0)
       {
              gpiberr("Read Error/File Open Error:");
              EXIT(1);
       }
}

/*
 * Read waveform data and convert to scaled waveform.
 *
 * The waveform is formatted as #<x><yyy><data> where
 *      <x> is the number of y bytes; for example if yyy = 500, then
 *           x = 3.
 *     <yyy> is the number of bytes to transfer;
 *        if width is 1 then all bytes on bus are single data
 *        points; if width is 2 then bytes on bus are
 *        2-byte pairs; this program uses width of 2.
 *    <data> is the curve data.
 */
ReadandStdout()
{
       char cmd[CMD_LEN + 1];
       LONG   llen;               /* Data size              */
       LONG   li;                 /* Loop index             */
       int    dlen;               /* Size                   */
       int    c;
       int    i;                  /* Loop index             */

       /*
        * Get some parameters in preamble.
        */
       if (awgWrite(":HEADER  OFF") < 0)
       {
              gpiberr("Write Error: Unable to turn header off\n");
              EXIT(1);
       }
       nr_pt = atol(awgWR("WFMPRE:NR_PT?", cmd, CMD_LEN));
       yzero  = atof(awgWR("WFMPRE:YZERO?", cmd, CMD_LEN));
       yoff  = atof(awgWR("WFMPRE:YOFF?", cmd, CMD_LEN));
       ymult = atof(awgWR("WFMPRE:YMULT?", cmd, CMD_LEN));
       xincr = atof(awgWR("WFMPRE:XINCR?", cmd, CMD_LEN));
```

```
pt_off = atol(awgWR(”WFMPRE:PT_OFF?”, cmd, CMD_LEN));
awgWR(”WFMPRE:XUNIT?”, xunit, LEN12);
awgWR(”WFMPRE:YUNIT?”, yunit, LEN12);

/*
 * Read the header information in <Arbitrary Block>.
 * The header includes #<x><yyy>.
 */
if (awgWrite(”:CURVE?”) < 0)
{
       gpiberr(”Write Error: CURVE?”);
       EXIT(1);
}
awgRead(cmd, 1);    /* Read the ’#’ symbol                          */
awgRead(cmd, 1);    /* Read string length of num bytes to transfer */
c = atoi(cmd);      /* Convert string to integer                    */
awgRead(cmd, c);    /* Read string containing number of bytes
                       to transfer                                  */
llen = ldiv(atol(cmd), 2L).quot; /* Two bytes per one data point   */

/*
 * Read the raw waveform data, and then process them.
 */
fprintf(stdout, ”%s,%s,＼”%s＼”, max. number of data point (%ld)＼n”,
                                  xunit, yunit, source, nr_pt);
for (li = 0; li < llen;)
{
       if (awgRead(wfm, MAX_DATA2) < 0)
       {
              gpiberr(”Read Error: WAVEFORM”);
              EXIT(1);
       }
       /*
        * Output scaled x, y values in (Sec, Volts)
        * Time[li] = (li – PTOFF) * XINCR
        * Volts[li] = YZERO + (point value – YOFF) * YMULT
        */
       dlen = ibcnt / 2; /* Two bytes per one data point */
       for(i = 0; i < dlen; i++)
       {
              fprintf(stdout, ”%.2e,%.2e＼n＼r”,
                     (FLOAT)(li – pt_off)*(FLOAT)(xincr), yzero +
                     (FLOAT)(((FLOAT)wfm[i] – (FLOAT)yoff) * ymult));
              li++;
       }
```

```
        }
        /*
         * Cleanup.
         */
        if (awgWrite(":HEADER ON") < 0)
        {
               gpiberr("Write Error: Unable to turn header off\n");
               EXIT(1);
        }
        fprintf(stdout, "\n");
        fprintf(stdout, "Waveform from %s successfully transferred!\n", source);
        return 0;
}

/*
 * Write GPIB query, and immediately read the response.
 */
char   *awgWR(cmd, resp, cnt)
char   *cmd, *resp;
int    cnt;
{
        if (awgWrite(cmd) < 0)
        {
               gpiberr("Write Error: WFMPRE?");
               EXIT(1);
        }
        if (awgRead(resp, cnt) < 0)
        {
               gpiberr("Read Error: WFMPRE");
               EXIT(1);
        }
        resp[ibcnt -1] = '\0';    /* Replace '\n' at the end of the response
                                  with '\0'.                          */
        return resp;
}
```

Quick BASIC の場合

```
DECLARE SUB GPIB2ASC (DEV%, FLNAME$)
DECLARE SUB GPIB2ISF (DEV%, FLNAME$)
DECLARE SUB CHKSTAT (DEV%, ESR%, EVENT$)
DECLARE SUB FINDDEV (KEYNAME$, DEV%)
DECLARE SUB EXTOPT (OPTION$, SOURCE$, FLNAME$)
DECLARE FUNCTION DISKERR$ ()
'$INCLUDE: 'QBDECL.BAS'
PRINT
PRINT "GETWFM Ver.1.0 "
PRINT "     Sample Progarm for AWG2000 series"
PRINT "                 Copyright(C)Tektronix Japan, Ltd. All rights reserved."
PRINT "             No warranty."
'
'Check COMMAND Arguments and extract source & filename
'
       OPTION$ = COMMAND$
       CALL EXTOPT(OPTION$, SOURCE$, FLNAME$)
'
'GPIB address search
'
       KEYNAME$ = "SONY/TEK,AWG2"
       CALL FINDDEV(KEYNAME$, DEV%)
       PRINT KEYNAME$
       IF DEV% = 0 THEN BEEP: END
'
'Check DATA source
'
       WRT$ = "HEADER ON;:DATA:SOURCE '" + SOURCE$ + "';:WFMPRE?"
       CALL IBWRT(DEV%, WRT$)
       RD$ = SPACE$(500): CALL IBRD(DEV%, RD$)
       IF INSTR(RD$, "WFID") = 0 THEN
              BEEP
              PRINT "ERROR. "; SOURCE$; " data is none."
              END
       END IF
'
'Set DATA ENCDG to SRPBINARY. It's a signed integer and transfer the LSB data
fi'rst.
'
       CALL IBWRT(DEV%, "data:encdg srpbin;width 2")
'
'Choose saved data type with extention.
'
       IF INSTR(FLNAME$, ".CSV") OR INSTR(FLNAME$, "CONS:") THEN
              CALL GPIB2ASC(DEV%, FLNAME$)
       ELSE
              CALL GPIB2ISF(DEV%, FLNAME$)
       END IF
```

```
'
'Check GPIB Status.
'
       DO
              CALL CHKSTAT(DEV%, ESR%, EVENT$)
              IF ESR% <> 0 THEN
                     BEEP
                     PRINT "Worning."
                     PRINT EVENT$
              END IF
       LOOP UNTIL ESR% = 0

END
'
'ERROR Trap routine.
'
ERRHANDLER:
       BEEP
       PRINT "ERROR. "; DISKERR$
       END
END
'- - - - - - - - - - - - - - - - - - - - - - - - End of Main procedure

SUB CHKSTAT (DEV%, ESR%, EVENT$)

       CALL IBRSP(DEV%, sta%)
       CALL IBWRT(DEV%, "*esr?")
       RD$ = SPACE$(16)
       CALL IBRD(DEV%, RD$)
       ESR% = VAL(RD$)

       CALL IBWRT(DEV%, "allev?")
       RD$ = SPACE$(500)
       CALL IBRD(DEV%, RD$)
       EVENT$ = LEFT$(RD$, IBCNT% - 1)

END SUB

FUNCTION DISKERR$
       SELECT CASE ERR
              CASE 54
                     DISKERR$ = "Bad file mode"
              CASE 64
                     DISKERR$ = "Bad file name"
              CASE 52
                     DISKERR$ = "Bad name or number"
              CASE 25
                     DISKERR$ = "Device fault"
```

```
                CASE 57
                       DISKERR$ = ”Device I/O error”
                CASE 24
                       DISKERR$ = ”Device timeout”
                CASE 68
                       DISKERR$ = ”Device unavailable”
                CASE 61
                       DISKERR$ = ”Disk full”
                CASE 72
                       DISKERR$ = ”Disk - media error”
                CASE 71
                       DISKERR$ = ”Disk not ready”
                CASE 53
                       DISKERR$ = ”File not found”
                CASE 62
                       DISKERR$ = ”Input past end of file”
                CASE 76
                       DISKERR$ = ”Path not found”
                CASE 75
                       DISKERR$ = ”Path/File sccess error”
                CASE 70
                       DISKERR$ = ”Permission denied”
                CASE 67
                       DISKERR$ = ”Too many files”
                CASE ELSE
                       DISKERR$ = ”???”
                END SELECT
END FUNCTION

SUB EXTOPT (OPTION$, SOURCE$, FLNAME$)

       IF OPTION$ = ”” THEN GOTO DISPUSAGE
       OPTION$ = OPTION$ + ” ”
       SOURCE$ = ””
       FLNAME$ = ””
’
’Extract string between spaces.
’
       FOR I% = 1 TO LEN(OPTION$)
              A$ = MID$(OPTION$, I%, 1)
              IF A$ = ” ” THEN EXIT FOR
              SOURCE$ = SOURCE$ + A$
       NEXT I%

       FOR J% = I% + 1 TO LEN(OPTION$)
              A$ = MID$(OPTION$, J%, 1)
              IF A$ = ” ” THEN EXIT FOR
              FLNAME$ = FLNAME$ + A$
       NEXT J%
```

```
'
'clean up DATA source.
'
       IF FLNAME$ = "" THEN FLNAME$ = "CONS:"
       IF SOURCE$ = "CH1" THEN EXIT SUB
       IF SOURCE$ = "CH2" THEN EXIT SUB
       IF INSTR(SOURCE$, ".WFM") THEN EXIT SUB
       BEEP
       PRINT "Invalid argument."

DISPUSAGE:
       PRINT
       PRINT "Usage:GETWFM <source> [<filename>]"
       PRINT
       PRINT "         <source>:Waveform data to transfer"
       PRINT "                  CH1/CH2/Wafeform file."
       PRINT "                  Wafeform file have a '.WFM' extention."
       PRINT
       PRINT "         [<filename>]:Output filename"
       PRINT "                  If no spec, dislay on the screen."
       PRINT "                  Specially '.CSV' extention is given, convert to
the ascii"
       PRINT "                  data for spread sheet software."
END
END SUB

SUB FINDDEV (KEYNAME$, DEV%)

       CALL IBFIND("GPIB0", BD%)
       IF BD% < 0 THEN
              KEYNAME$ = "'GPIB0' not found."
              DEV% = 0
              EXIT SUB
       END IF

       CALL IBFIND("DEV1", DEV%)
       IF DEV% <= 0 THEN
              KEYNAME$ = "'DEV1' not found, Please run IBCONF and define."
              DEV% = 0
              EXIT SUB
       END IF
       CALL IBSRE(BD%, 0)
       CALL IBSRE(BD%, 1)  V% = 11: CALL IBTMO(DEV%, V%)
       AD% = 0
```

```
'
'GPIB Address Search
'      DO
              CALL IBPAD(DEV%, AD%)
              CALL IBWRT(DEV%, "*IDN?")
              IF IBSTA% AND &H8000 THEN
                     AD% = AD% + 1
              ELSE
                     id$ = SPACE$(100): CALL IBRD(DEV%, id$)
                     IF INSTR(id$, UCASE$(KEYNAME$)) THEN
                            EXIT DO
                     ELSE
                            AD% = AD% + 1
                            CALL IBCLR(DEV%)
                     END IF
              END IF
              IF 30 < AD% THEN
                     KEYNAME$ = "Specified instrument not found."
                     DEV% = 0
                     EXIT SUB
              END IF
       LOOP

       V% = 13: CALL IBTMO(DEV%, V%)
       KEYNAME$ = LEFT$(id$, IBCNT% - 1) + "(GPIB Address =" + STR$(AD%) + ")"

       CALL IBWRT(DEV%, ":DESE 255;*CLS")

END SUB

SUB GPIB2ASC (DEV%, FLNAME$)
'
'Request to send the waveform data.
'
       CALL IBWRT(DEV%, "CURVE?")
'
'Read the waveform data
'
'The waveform data is formatted as #<x><yyy><data><newline> where
'  <x> is the number of bytes of <yyy>, for example if yyy = 500, then x = 3.
'  <yyy> is the number of bytes to transfer inclulude checksum.
'   (The AWG don't send cheksum.)
'   The resolution in the AWG2000 is 12 bits/point, then the number of bytes at
'   one point data is two. The Length of waveform data is the harf of yyy.
'  <data> is the curve data.
'  <newline> End of data block.(=0AH(linefeed character))
'
       RD$ = SPACE$(1)                'define buffer to 1 byte
       DO
              CALL IBRD(DEV%, RD$)    'read and discard until '#' symbol
```

```
        LOOP UNTIL RD$ = "#"
        CALL IBRD(DEV%, RD$)            'read <x>
        RD$ = SPACE$(VAL(RD$))          'set buffer to x bytes
        CALL IBRD(DEV%, RD$)            'read <yyy>
        BYTCNT& = VAL(RD$)              'get the number of bytes to transfer
'
'Define an arrey for raw data. It's the twe bytes signed integer arrey.
'The length of the arrey is the harf of total bytes count.
'
        NRPT& = BYTCNT& / 2
'
'Limit the data length to 32k bytes.
'
        IF 32767 <= NRPT& THEN
                BEEP
                PRINT "Data length is too long. Set to till 32k words."
                DO
                        CALL CHKSTAT(DEV%, ESR%, EVENT$)
                LOOP UNTIL ESR% = 0
                END
        END IF

        NRPT% = NRPT&
        BYTCNT% = BYTCNT&

        DIM WFM%(NRPT% - 1)     'Option base is 0.
'
'Read the waveform data at two bytes pair by IBRDI.
'
        CALL IBRDI(DEV%, WFM%(), BYTCNT%)
        IF IBSTA% < 0 THEN
                BEEP
                PRINT "Error on Reaf waveform data."
                END
        END IF
'
'Read the End charctor
'
        RD$ = SPACE$(2)
        CALL IBRD(DEV%, RD$)

'
'Read Scale data and Convert the raw data to voltae value.
'
        CALL IBWRT(DEV%, ":HEADER OFF;:WFMPRE:YOFF?")
        RD$ = SPACE$(40)
        CALL IBRD(DEV%, RD$)
        YOFF! = VAL(RD$)
```

```
        CALL IBWRT(DEV%, "WFMPRE:YZERO?")
        RD$ = SPACE$(40)
        CALL IBRD(DEV%, RD$)
        YZERO! = VAL(RD$)

        CALL IBWRT(DEV%, "WFMPRE:YMULT?")
        RD$ = SPACE$(40)
        CALL IBRD(DEV%, RD$)
        YMULT! = VAL(RD$)

        CALL IBWRT(DEV%, "WFMPRE:XINCR?")
        RD$ = SPACE$(40)
        CALL IBRD(DEV%, RD$)
        XINCR! = VAL(RD$)

        CALL IBWRT(DEV%, "WFMPRE:PT_OFF?")
        RD$ = SPACE$(40)
        CALL IBRD(DEV%, RD$)
        PTOFF! = VAL(RD$)

'
'  X axsis unit, Y axsis unit, date, time
'
        ON ERROR GOTO ERRHANDLER

        OPEN FLNAME$ FOR OUTPUT AS #1
              WRITE #1, "sec", "Volts", DATE$, TIME$
'
'Scaling method
'       Time[i] = (i  - PT_OFF) * XINCR
'       Volts[i] = (point value  - YOFF) * YMULT + YZERO
'
              FOR I% = 0 TO NRPT%  - 1
                    TTT! = (I%  - PTOFF!) * XINCR!
                    VOLTS! = (WFM%(I%)  - YOFF!) * YMULT! + YZERO!
                    PRINT #1, TTT!; ",";  VOLTS!
              NEXT I%
        CLOSE #1

        ON ERROR GOTO 0
        PRINT NRPT%; "points data is written to"; FLNAME$

END SUB
```

```
SUB GPIB2ISF (DEV%, FLNAME$)
'
'Request to send Preamble and waveorm data
'
      CALL IBWRT(DEV%, "WAVFRM?")
'
'Read the waveform data and write to file.
'The IBRDF transfer from GPIB to file directory.
'
'more file error check.(because IBRDF can't check the disk cache)
'
      ON ERROR GOTO ERRHANDLER
      OPEN FLNAME$ FOR OUTPUT AS #1
      CLOSE #1

      CALL IBRDF(DEV%, FLNAME$)
      IF IBSTA% AND &H8000 THEN
            BEEP
            PRINT "Error on writing data."
            END
      ELSE
            PRINT IBCNTL&; "bytes data is written to "; FLNAME$
      END IF

      ON ERROR GOTO 0

END SUB
```

プログラム例2： 波形の転送（その2）

2番目のプログラムは、外部コントローラから本機器に波形データとプリアンブルを転送する例です。プログラム例1でファイルに保存したデータを本機器に戻す方法が取り扱われています。

Quick Cの場合

```
/*
 * putwfm.c – a simple waveform transfer program that restores waveform
 * to the instrument. The waveform must be one obtained with the getwfm
 * program.
 */
# include    <stdio.h>
# include    ”decl.h”
# include    ”exit.h”

void   checkarg();

char   *infile;     /* Output file descriptor                       */
char   *destination;/* Destination from which a waveform is transferred */

main(argc, argv)
int    argc;
char   *argv[];
{
       printf(”\n\n”);
       printf(”PUTWFM –  simple waveform transfer program.\n”);
       printf(”Copyright (c) Tektronix Japan, Ltd.”);
       printf(” All Rights Reserved.\n\n”);

       checkarg(argc, argv);      /* Check whether the arguments are valid.*/
       open_dev();         /* Find GPIB devices                    */
       DestSetup();        /* Define destination in the instrument */
       FtoGPIBwrite();     /* Read preamble and waveform, and
                              write them to the instrument.        */
       close_dev();
}

/*
 * Check whether the arguments are valid.
 */
void   checkarg(argc, argv)
int    argc;
char   *argv[];
{
       /*
        * Check for command line argument count.
        */
       if(argc != 3)
       {
              fprintf(stderr, ”usage: putwfm <destination> <in–file>\n”);
              fprintf(stderr, ”\twhere:\n”);
              fprintf(stderr, ”\t\t<destination>\t\tis the destination”);
              fprintf(stderr, ” waveform file to be written\n”);
```

```
                fprintf(stderr, "\t\t<in-file>\tis the input file\n");
                EXIT(1);
        }
        /*
         * Check for valid destination.
         */
        destination = argv[1];
        if(wfmfile(argv[1]) != 0)
        {
                fprintf(stderr, "ERROR: Invalid destination: \"%s\":", argv[1]);
                EXIT(1);
        }
        infile = argv[2];
}

/*
 * Check whether the file extension is '.wfm'.
 */
wfmfile(name)
char   *name;
{
        int    dlen = strlen(name);

        if (dlen < 4 || dlen > 12)
                return -1;
        if (strcmp(&name[dlen - 4], ".WFM") == 0 ||
            strcmp(&name[dlen - 4], ".wfm") == 0)
                return 0;
        return -1;
}

/*
 * Define destination to be written in the instrument.
 */
DestSetup()
{
        char cmd[100];

        sprintf(cmd, ":DATA:DESTINATION \"%s\"", destination);
        if(awgWrite(cmd) < 0)
        {
                gpiberr("Write Error: Unable to setup waveform parameters");
                EXIT(1);
        }
}
```

```
/*
 * Read waveform and preamble from a file, and then write them
 * to the instrument.
 */
FtoGPIBwrite()
{
       if (WrtFtoG(infile) < 0)
       {
          gpiberr(”Read error/File open error:”);
          EXIT(1);
       }
}
```

Quick BASIC の場合

```
DECLARE SUB CHKSTAT (DEV%, ESR%, EVENT$)
DECLARE SUB FINDDEV (KEYNAME$, DEV%)
DECLARE SUB EXTOPT (OPTION$, FLNAME$, DESTINATION$)
DECLARE FUNCTION DISKERR$ ()
DECLARE SUB ISF2GPIB (DEV%, FLNAME$)
'$INCLUDE: 'QBDECL.BAS'
PRINT
PRINT "PUTWFM Ver.1.0 "
PRINT "     Sample Program for AWG2000 series"
PRINT "                Copyright(C) Tektronix Japan, Ltd. All rights reserved."
PRINT "             No warranty."
'
'Check COMMAND Arguments and extract source & filename
'
       OPTION$ = COMMAND$
       CALL EXTOPT(OPTION$, FLNAME$, DESTINATION$)
'
'GPIB Adress search
'
       KEYNAME$ = "SONY/TEK,AWG2"
       CALL FINDDEV(KEYNAME$, DEV%)
       PRINT KEYNAME$
       IF DEV% = 0 THEN BEEP: END
'
'Check file name.
'
       WRT$ = ":DATA:DESTINATION '" + DESTINATION$ + "'"
       CALL IBWRT(DEV%, WRT$)
       CALL CHKSTAT(DEV%, ESR%, EVENT$)
       IF ESR% <> 0 THEN
              BEEP
              PRINT "Error on file name."
              PRINT EVENT$
              END
       END IF
'
'Data transfer.
'
       CALL ISF2GPIB(DEV%, FLNAME$)
'
'Check GPIB Status
'
       DO
              CALL CHKSTAT(DEV%, ESR%, EVENT$)
              IF ESR% <> 0 THEN
                     BEEP         PRINT "Worning."
                     PRINT EVENT$
              END IF
       LOOP UNTIL ESR% = 0

END
```

```
'
'ERROR Trap routine
'
ERRHANDLER:
       BEEP
       PRINT "ERROR. "; DISKERR$
       END
END
'- - - - - - - - - - - - - - - - - - - - - - - End fo Main procedure

SUB CHKSTAT (DEV%, ESR%, EVENT$)

       CALL IBRSP(DEV%, sta%)
       CALL IBWRT(DEV%, "*esr?")
       RD$ = SPACE$(16)
       CALL IBRD(DEV%, RD$)
       ESR% = VAL(RD$)

       CALL IBWRT(DEV%, "allev?")
       RD$ = SPACE$(500)
       CALL IBRD(DEV%, RD$)
       EVENT$ = LEFT$(RD$, IBCNT% - 1)

END SUB

FUNCTION DISKERR$
       SELECT CASE ERR
              CASE 54
                     DISKERR$ = "Bad file mode"
              CASE 64
                     DISKERR$ = "Bad file name"
              CASE 52
                     DISKERR$ = "Bad name or number"
              CASE 25
                     DISKERR$ = "Device fault"
              CASE 57
                     DISKERR$ = "Device I/O error"
              CASE 24
                     DISKERR$ = "Device timeout"
              CASE 68
                     DISKERR$ = "Device unavailable"
              CASE 61
                     DISKERR$ = "Disk full"
              CASE 72
                     DISKERR$ = "Disk - media error"
              CASE 71
                     DISKERR$ = "Disk not ready"
              CASE 53
                     DISKERR$ = "File not found"
              CASE 62
                     DISKERR$ = "Input past end of file"
              CASE 76
                     DISKERR$ = "Path not found"
              CASE 75
                     DISKERR$ = "Path/File sccess error"
              CASE 70
```

```
                DISKERR$ = ”Permission denied”
            CASE 67
                DISKERR$ = ”Too many files”
            CASE ELSE
                DISKERR$ = ”???”
            END SELECT
END FUNCTION

SUB EXTOPT (OPTION$, FLNAME$, DESTINATION$)

    IF OPTION$ = ”” THEN GOTO DISPUSAGE
    OPTION$ = OPTION$ + ” ”
    FLNAME$ = ””
    DESTINATION$ = ””
’
’Extract string between spaces.
’
    FOR I% = 1 TO LEN(OPTION$)
        A$ = MID$(OPTION$, I%, 1)
        IF A$ = ” ” THEN EXIT FOR
        DESTINATION$ = DESTINATION$ + A$
    NEXT I%

    FOR J% = I% + 1 TO LEN(OPTION$)
        A$ = MID$(OPTION$, J%, 1)
        IF A$ = ” ” THEN EXIT FOR
        FLNAME$ = FLNAME$ + A$
    NEXT J%
’
’Check argumants
’
    IF INSTR(DESTINATION$, ”.WFM”) AND FLNAME$ <> ”” THEN EXIT SUB
    BEEP
    PRINT ”Invalid argument.”

DISPUSAGE:
    PRINT
    PRINT ”Usage:PUTWFM <destination> <filename>”
    PRINT
    PRINT ”         <destination>:Waveform filename to distination”
    PRINT ”                    extention is ’.WFM’”
    PRINT
    PRINT ”         <filename>:Waveform file to transfer”
    PRINT ”                    must be instument specified format.”
    PRINT
    PRINT ”  This program read the waveform file form disk and send to the
AWG2000.”
    PRINT ”  If same as <destination> is already exist in the memory of the
AWG2000,”
    PRINT ”  and if the file isn’t locked, It’s overwrited.”

END

END SUB
```

```
SUB FINDDEV (KEYNAME$, DEV%)

       CALL IBFIND(”GPIB0”, BD%)
       IF BD% < 0 THEN
              KEYNAME$ = ”’GPIB0’ not found.”
              DEV% = 0
              EXIT SUB
       END IF

       CALL IBFIND(”DEV1”, DEV%)
       IF DEV% <= 0 THEN
              KEYNAME$ = ”’DEV1’ not found.”
              DEV% = 0
              EXIT SUB
       END IF
       CALL IBSRE(BD%, 0)
       CALL IBSRE(BD%, 1)

       V% = 11: CALL IBTMO(DEV%, V%)
       AD% = 0
’
’GPIB Address search
’
       DO
              CALL IBPAD(DEV%, AD%)
              CALL IBWRT(DEV%, ”*IDN?”)
              IF IBSTA% AND &H8000 THEN
                     AD% = AD% + 1
              ELSE
                     id$ = SPACE$(100): CALL IBRD(DEV%, id$)

                     IF INSTR(id$, UCASE$(KEYNAME$)) THEN
                            EXIT DO
                     ELSE
                            AD% = AD% + 1
                            CALL IBCLR(DEV%)
                     END IF
              END IF
              IF 30 < AD% THEN
                     KEYNAME$ = ”Specified instrument not found.”
                     DEV% = 0
                     EXIT SUB
              END IF
       LOOP

       V% = 13: CALL IBTMO(DEV%, V%)
       KEYNAME$ = LEFT$(id$, IBCNT% - 1) + ”  (GPIB Address =” + STR$(AD%) + ”)”
       CALL IBWRT(DEV%, ”:DESE 255;*CLS”)

END SUB

SUB ISF2GPIB (DEV%, FLNAME$)
’
’Read waveform from file and transfer to GPIB.
’The IBWRTF function transfer from file to GPIB directry
’No problem on binary file.
```

```
        '
        'more file error check
        '
               ON ERROR GOTO ERRHANDLER
               OPEN FLNAME$ FOR INPUT AS #1
               CLOSE #1

               CALL IBWRTF(DEV%, FLNAME$)
               IF IBSTA% < 0 THEN
                      BEEP
                      PRINT "Error when transfer data."
                      END
               ELSE
                      PRINT FLNAME$; "is tarnsfered (at"; IBCNTL&; "bytes)."
               END IF

               ON ERROR GOTO 0

        END SUB
```

プログラム例3： イクエーション・データの転送と設定

3番目のプログラムは、イクエーション・データを転送し、その波形出力を得る例です。このプログラムでは、イクエーション・データの転送、波形データへのコンパイル、実行終了待ち、セットアップ、波形出力開始の方法などが取り扱われています。

Quick Cの場合

```
/*
 * equset.c  –      equation data processing program that writes and compiles
 * equation data, sets the instrument up, and turns output on.
 */
# include    <stdio.h>
# include    <stdlib.h>
# include    ”decl.h”
# include    ”exit.h”

# define            CMD_LEN       100

typedef             long   LONG;

LONG   wpoints = 8000;
char   *equfile = ”equsampl.equ”; /* Equation file to be created   */
char   *wfmfile = ”equsampl.wfm”; /* Waveform file to be created   */
char   *equation =                /* Equation data                 */
”range(0,50ms)\n\
K0=100e–3\n\
K1=63.3e–9\n\
K2=K0*K1\n\
K3=10e–3\n\
exp(–t/K3)*sin(1/sqrt(K2)*t)\n\
range(51ms,100ms)\n\
exp(–t/K3)*sin(1/sqrt(K2)*t)\n\
range(101ms,150ms)\n\
exp(–t/K3)*sin(1/sqrt(K2)*t)\n\
range(151ms,200ms)\n\
exp(–t/K3)*sin(1/sqrt(K2)*t)\n\
range(201ms,250ms)\n\
exp(–t/K3)*sin(1/sqrt(K2)*t)\n\
norm()\n”;

main()
{
       char   cmd[CMD_LEN + 1];

       open_dev();               /*  Find GPIB devices            */

       printf(”\n\n”);
       printf(”EQUSET      –  equation data processing program.\n”);
       printf(”Copyright (c) Tektronix Japan, Ltd.”);
       printf(” All Rights Reserved.\n\n”);
       printf(”Start processing ...\n\n”);
       printf(”Lock front panel controls.\n”);
       awgWrite(”:LOCK  ALL”);
```

```
        /*
         * Process equation data.
         */
        WriteCompEqu(); /* Write equation data and number of waveform
                           points, and then compile              */
        WaveOutput();  /* Setup for output and turn output on       */

        /*
         * Cleanup.
         */
        printf("Recover front panel controls.\n\n");
        awgWrite(":LOCK  NONE");
        close_dev();
}

/*
 * Write equation data and number of waveform points, and then compile.
 */
WriteCompEqu()
{
        int    l;                       /* Size                          */
        char   cmd[CMD_LEN + 1];        /* Command buffer          */

        awgWrite("ABSTOUCH  EDIT");/* Display EDIT screen           */

        /*
         * Check whether the file exists
         */
        if (awgWrite("*CLS ;:DESE 255 ;*ESE 16 ;*SRE 0") < 0)
        {                               /* Set for serial poll     */
                gpiberr("Write Error:");
                EXIT(3);
        }
        sprintf(cmd, ":MEMORY:LOCK? \"%s\"", equfile);
        if (awgWrite(cmd) < 0)
        {
                gpiberr("Write Error:\n");
                EXIT(3);
        }
        if (!(serialp() & 0x20))        /* Check for the ESB bit in the SRB */
        {
                awgtmo(T1s);        /* Wait for further 100us           */
                awgwait(TIMO | SRQI | RQS | END);
                awgtmo(T10s);       /* Reset to 10s                     */
                if (!(serialp() & 0x20))
                {
                        fprintf(stderr,
                                "Equation file (%s) already exists\n", equfile);
                        EXIT(3);
                }
        }

        /*
         * Write equation data.
         */
        if (awgWrite("*CLS ;:DESE 255 ;*ESE 1 ;*SRE 0") < 0)
```

```
        {                                   /* Set for serial poll           */
               gpiberr("Write Error:");
               EXIT(3);
        }
        printf("Write equation data.\n");
        l = strlen(equation);
        eotcont(0);                /* Turn off sending terminator          */
        sprintf(cmd, ":EQUATION:DEFINE \"%s\", #%d%d",
                                    equfile, DigitCount((LONG)l), l);
        if (awgWrite(cmd) < 0)
        {
               gpiberr("Equation Definition Error:");
               EXIT(3);
        }
        eotcont(1);                /* Turn on sending terminator           */
        if (awgWrite(equation) < 0)
        {
               gpiberr("Equation Definition Error:");
               EXIT(3);
        }

        /*
         * Write number of waveform points.
         */
        printf("Write number of waveform points.\n");
        sprintf(cmd, ":EQUATION:WPOINTS \"%s\",%ld", equfile, wpoints);
        if (awgWrite(cmd) < 0)
        {
               gpiberr("Waveform Point Write Error:");
               EXIT(3);
        }

        /*
         * Compile.
         */
        printf("Start compiling...\n");
        sprintf(cmd, ":EQUATION:COMPILE \"%s\" ;*OPC", equfile);
        if (awgWrite(cmd) < 0)
        {
               gpiberr("Equation Compile Command Write Error:");
               EXIT(3);
        }
        /*
         * Wait for the termination by checking status byte.
         */
        printf("Wait for its termination.\n\n");
        while (!serialp())  /* Keep looping while serial_poll = 0           */
               ;

        awgWrite(":DESE 255 ;*ESE 0 ;*SRE 0");  /* Set back          */
}
```

```
/*
 * Set up the instrument for output, and turns output on.
 */
WaveOutput()
{
       int    l;                          /* Size                    */
       char   cmd[CMD_LEN + 1];           /* Command buffer          */

       awgWrite(”ABSTOUCH  SETUP”);            /* Display SETUP screen */
       awgWrite(”:OUTPUT:CH1:STATE OFF”);      /* Turn output off  */

       /*
        * Set waveform to CH1
        */
       printf(”Set waveform file (%s) to CH1.\n”, wfmfile);
       sprintf(cmd, ”CH1:WAVEFORM  \”%s\””, wfmfile);
       if (awgWrite(cmd) < 0)
       {
              gpiberr(”Write Error:”);
              EXIT(4);
       }

       /*
        * Set mode to triggered
        */
       printf(”Set mode to triggered.\n”);
       if (awgWrite(”MODE  TRIGGERED”) < 0)
       {
              gpiberr(”Write Error:”);
              EXIT(4);
       }

       /*
        * Set output parameters and ready to start output by trigger.
        */
       printf(”Set amplitude to 2.0V, and frequency to 20MHz.\n\n”);
       if (awgWrite(”:CH1:AMPLITUDE 2.0V”) < 0 ||
           awgWrite(”:CLOCK:FREQUENCY   20MHz”) < 0 ||
           awgWrite(”:OUTPUT:CH1:STATE   ON”) < 0)
       {
              gpiberr(”Write Error:”);
              EXIT(4);
       }
}
```

```
/*
 * Count digits.
 */
DigitCount(n)
LONG   n;
{
       int    cc = 1;
       while (n = ldiv(n, 10L).quot)
              cc++;
       return cc;
}
```

Quick BASIC の場合

```
DECLARE SUB WAVEOUTPUT (DEV%, WFMFILE$)
DECLARE SUB WRITECOMPEQU (DEV%, WPOINTS&, EQUFILE$, EQUATION$)
DECLARE SUB CHKSTAT (DEV%, ESR%, EVENT$)
DECLARE SUB FINDDEV (KEYNAME$, DEV%)
'$INCLUDE: 'QBDECL.BAS'
PRINT
PRINT "EQUSET Ver.1.0 "
PRINT "     Sample Program for AWG2000 series"
PRINT "              Copyright(C) Tektronix Japan, Ltd. All rights reserved."
PRINT "         No warranty."
'
'Define equation data and file names
'
       WPOINTS& = 8000             'number of waveform points
       EQUFILE$ = "EQUSAMPL.EQU"   'Equation file to created
       WFMFILE$ = EQUFILE$         'Waveform file to Created
       MID$(WFMFILE$, INSTR(WFMFILE$, ".EQU")) = ".WFM"
       EQUATION$ = ""              'Equation data

       EQUATION$ = EQUATION$ + "range(0,50ms)" + CHR$(10)
       EQUATION$ = EQUATION$ + "K0=100e-3" + CHR$(10)
       EQUATION$ = EQUATION$ + "K1=63.3e-9" + CHR$(10)
       EQUATION$ = EQUATION$ + "K2=K0*K1" + CHR$(10)
       EQUATION$ = EQUATION$ + "K3=10e-3" + CHR$(10)
       EQUATION$ = EQUATION$ + "exp(-t/K3)*sin(1/sqrt(K2)*t)" + CHR$(10)
       EQUATION$ = EQUATION$ + "range(51ms,100ms)" + CHR$(10)
       EQUATION$ = EQUATION$ + "exp(-t/K3)*sin(1/sqrt(K2)*t)" + CHR$(10)
       EQUATION$ = EQUATION$ + "range(101ms,150ms)" + CHR$(10)
       EQUATION$ = EQUATION$ + "exp(-t/K3)*sin(1/sqrt(K2)*t)" + CHR$(10)
       EQUATION$ = EQUATION$ + "range(151ms,200ms)" + CHR$(10)
       EQUATION$ = EQUATION$ + "exp(-t/K3)*sin(1/sqrt(K2)*t)" + CHR$(10)
       EQUATION$ = EQUATION$ + "range(201ms,250ms)" + CHR$(10)
       EQUATION$ = EQUATION$ + "exp(-t/K3)*sin(1/sqrt(K2)*t)" + CHR$(10)
       EQUATION$ = EQUATION$ + "norm()"
'
'Search GPIB Address
'
       KEYNAME$ = "SONY/TEK,AWG2"
       CALL FINDDEV(KEYNAME$, DEV%)
       PRINT KEYNAME$
       IF DEV% = 0 THEN BEEP: END
'
'Lock the front pannel controls
'
       PRINT "Processing..."
       PRINT "Lock front pannel controls."
       CALL IBWRT(DEV%, ":LOCK ALL")
'
'Write the equation data and number of points and compile
'
       CALL WRITECOMPEQU(DEV%, WPOINTS&, EQUFILE$, EQUATION$)
'
'Setup fot output and turns output on
'
       CALL WAVEOUTPUT(DEV%, WFMFILE$)
```

```
'
'Check GPIB Status
'
       DO
              CALL CHKSTAT(DEV%, ESR%, EVENT$)
              IF ESR% <> 0 THEN
                     BEEP
                     PRINT "Worning."
                     PRINT EVENT$
              END IF
       LOOP UNTIL ESR% = 0
'
'UNLock front pannel controls
'
       PRINT "Recover front panel controls."
       CALL IBWRT(DEV%, ":LOCK NONE")
END
'- - - - - - - - - - - - - - - - - - - - - - - - - - End of Main procedure

SUB CHKSTAT (DEV%, ESR%, EVENT$)

       CALL IBRSP(DEV%, STB%)
       CALL IBWRT(DEV%, "*ESR?")
       RD$ = SPACE$(16)
       CALL IBRD(DEV%, RD$)
       ESR% = VAL(RD$)

       CALL IBWRT(DEV%, "ALLEV?")
       RD$ = SPACE$(500)
       CALL IBRD(DEV%, RD$)
       EVENT$ = LEFT$(RD$, IBCNT% - 1)

END SUB

SUB FINDDEV (KEYNAME$, DEV%)

       CALL IBFIND("GPIB0", BD%)
       IF BD% < 0 THEN
              KEYNAME$ = "'GPIB0' not found."
              DEV% = 0
              EXIT SUB
       END IF

       CALL IBFIND("DEV1", DEV%)
       IF DEV% <= 0 THEN
              KEYNAME$ = "'DEV1' not found, Please run IBCONF and define."
              DEV% = 0
              EXIT SUB
       END IF
       CALL IBSRE(BD%, 0)
       CALL IBSRE(BD%, 1)
```

```
'
'GPIB Address search
'
       V% = 11: CALL IBTMO(DEV%, V%)
       AD% = 0
       DO
              CALL IBPAD(DEV%, AD%)
              CALL IBWRT(DEV%, "*IDN?")
              IF IBSTA% AND &H8000 THEN
                     AD% = AD% + 1
              ELSE
                     ID$ = SPACE$(100): CALL IBRD(DEV%, ID$)
                     IF INSTR(ID$, UCASE$(KEYNAME$)) THEN
                            EXIT DO
                     ELSE
                            AD% = AD% + 1
                            CALL IBCLR(DEV%)
                     END IF
              END IF
              IF 30 < AD% THEN
                     KEYNAME$ = "Specified instrument not found."
                     DEV% = 0
                     EXIT SUB
              END IF
       LOOP

       V% = 13: CALL IBTMO(DEV%, V%)
       KEYNAME$ = LEFT$(ID$, IBCNT% - 1) + "  (GPIBア Address =" + STR$(AD%) +
")"
       CALL IBWRT(DEV%, ":DESE 255;*CLS")

END SUB

SUB WAVEOUTPUT (DEV%, WFMFILE$)
'
'Dispaly SETUP screen to see the operation
'
       CALL IBWRT(DEV%, "ABSTOUCH SETUP")
'
'Turns output off
'
       CALL IBWRT(DEV%, "OUTPUT:CH1:STATE OFF")
'
'Set waveform file to CH1
'
       PRINT "Set the "; WFMFILE$; " to CH1."
       WRT$ = "CH1:WAVEFORM '" + WFMFILE$ + "'"
       CALL IBWRT(DEV%, WRT$)
'
'Set mode to triggered
'
       PRINT "Set mode to triggered."
       CALL IBWRT(DEV%, "MODE TRIGGERED")
```

```
'
'Set output parameters and turns output on
'
       PRINT "Set amplitude to 2.0V, and frequency to 20MHz."
       CALL IBWRT(DEV%, "CH1:AMPLITUDE 2.0V")
       CALL IBWRT(DEV%, "CLOCK:FREQUENCY 20MHz")
       CALL IBWRT(DEV%, "OUTPUT:CH1:STATE ON")

END SUB

SUB WRITECOMPEQU (DEV%, WPOINTS&, EQUFILE$, EQUATION$)
'
'Check file name to transfer
'
       CALL IBWRT(DEV%, "ABSTOUCH EDIT")'Display EDIT screen to see operation

       CALL IBWRT(DEV%, ":MEMORY:CATALOG:EQU?")
       RD$ = SPACE$(5000)
       CALL IBRD(DEV%, RD$)
       IF INSTR(RD$, UCASE$(EQUFILE$)) THEN
              BEEP
              PRINT EQUFILE$; "is already exist."
              INPUT "overwrite(y/[n])"; SURE$
              IF UCASE$(SURE$) <> "Y" THEN
                     CALL IBWRT(DEV%, "UNLOCK ALL")
                     END
              END IF
       END IF

'
'Write the equation data
'
       EQULENGTH$ = LTRIM$(RTRIM$(STR$(LEN(EQUATION$))))
       DIGCOUNT$ = "#" + LTRIM$(RTRIM$(STR$(LEN(EQULENGTH$)))) + EQULENGTH$
       WRT$ = ":EQUATION:DEFINE '" + EQUFILE$ + "',"  + DIGCOUNT$ + EQUATION$
       CALL IBWRT(DEV%, WRT$)
       CALL CHKSTAT(DEV%, ESR%, EVENT$)
       IF ESR% <> 0 THEN
              BEEP
              PRINT "Error on write the equation data."
              PRINT EVENT$
              CALL IBWRT(DEV%, "UNLOCK ALL")
              END
       END IF
```

```
'
'Write the number of points
'
       WRT$ = ":EQUATION:WPOINTS '" + EQUFILE$ + "'," + STR$(WPOINTS&)
       CALL IBWRT(DEV%, WRT$)
       CALL CHKSTAT(DEV%, ESR%, EVENT$)
       IF ESR% <> 0 THEN
              BEEP
              PRINT "Error on write the number of points."
              PRINT EVENT$
              CALL IBWRT(DEV%, "UNLOCK ALL")
              END
       END IF
'
'Set the event report resisters to know the operation complete
'Not use the SRQ and check by serial polling.
'
       CALL IBWRT(DEV%, ":DESE 255;*ESE 1;*SRE 0;*CLS")
'
'Compile
'
       PRINT "Compile in progress..."
       WRT$ = "EQUATION:COMPILE '" + EQUFILE$ + "';*OPC"
       CALL IBWRT(DEV%, WRT$)
'
'Wait to the operation complete
'
       DO
              CALL IBRSP(DEV%, STB%)
       LOOP UNTIL STB%
'
'Check the standard event status resister, if it's 1, that's OK but...
'

       CALL CHKSTAT(DEV%, ESR%, EVENT$)
       IF ESR% <> 1 AND ESR% <> 0 THEN
              BEEP
              PRINT "Error at Compiling."
              PRINT EVENT$
              CALL IBWRT(DEV%, "UNLOCK ALL")
              END
       END IF

END SUB
```

プログラム例4： インタラクティブ・コミュニケーション

4番目のプログラムは、本機器と外部コントローラとの間でインタラクティブに通信を行う例です。このプログラム例では、GPIBコマンドの転送、出力キューからの読み出し、イベント／ステータスの制御方法などが取り扱われています。

Quick Cの場合

```
/*
 * intrv.c  –  interactive communication program between the external
 * controller and the instrument.
 */
# include    <stdio.h>
# include    ”decl.h”

# define            MAX_BUF      128
# define            MAX_ARG      10
# define            FILE_LEN     15
# define            RD0          0
# define            RD1          1
# define            RD2          2
# define            RD3          3
# define            ON           1
# define            OFF          –1
# define            ERROR        –1
# define            NORMAL 1
# define            QUERY        ’?’
# define            NOQUERY      ’N’

extern int    iostatus;          /* Same as ibsta                  */

void   viewfile();
void   execfile();
void   helpmessg();
void   process();
void   bnull();
void   redirect();
void   chkstatus();
void   ReadOutputbuf();
char   *getfile();

int    argc;
char   *argv[MAX_ARG];
char   readbuf[MAX_BUF + 1];
char   replace[MAX_BUF + 1];
char   assmble[MAX_BUF + 1];
char   stack[MAX_BUF + 1];

FILE   *ifd;        /* Input file descriptor               */
FILE   *ofd;        /* Output file descriptor              */
char   *rfile;
int    rflag;       /* > : RD1, >> : RD2,  < : RD3, none : RD0    */
int    ropnum;
int    sflag;
```

```
main()
{
       open_dev();                /* Find GPIB devices             */

       printf("\n\n");
       printf("INTRV  –  interactive communication program.\n");
       printf("Copyright (c) Tektronix Japan, Ltd.");
       printf(" All Rights Reserved.\n");
       printf("Type 'help' to display help messages.\n");
       printf("Type 'q' or 'quit' to exit from the program.\n\n");

       printf("Start interactive communication\n\n");
       process();                 /* Communication process   */
       close_dev();
}

/*
 * Communication process.
 *
 */
void   process()
{
       /*
        * Initialize.
        */
       stack[0] = '\0';
       replace[0] = '\0';
       ofd = stdout;       /* Output to standard output           */
       ifd = stdin;        /* Input from standard input           */
       resetes(1);         /* Set enable registers                */

       /*
        * Start interactive communication.
        */
       for (;;)
       {
              chkstatus(); /* Check for the status byte, and dequeue
                                          events if stacked */
              tcerider();  /* Reset ofd to stdout                    */

              /*
               * Get and parse command line.
               */
              inputline();

              /*
               * Terminate interactive process.
               */
              if (strcmp(argv[0], "q") == 0 || strcmp(argv[0], "quit") == 0)
                     break;       /* Terminate interactive process */

              /*
               * Redirect.
               */
              if (rflag == RD1 || rflag == RD2)
                     redirect();
```

```
                /*
                 * Execute built-in command.
                 */
                if (strcmp(argv[0], "exec") == 0)
                       execfile(argc, argv[1]);
                else if (strcmp(argv[0], "view") == 0)
                       viewfile(argc, argv[1]);
                else if (strcmp(argv[0], "help") == 0)
                       helpmessg(argc);
                else if (strcmp(argv[0], "status") == 0)
                       statusbyte(argc);
                else if (strcmp(argv[0], "resetes") == 0)
                       resetes(argc);

                /*
                 * Transmit GPIB command or query. If query, the output
                 * queue is immediately read.
                 */
                else if (argc > 0)
                {
                       if (rflag == RD3)
                       {
                              fprintf(stderr,
                                     "syntax error: (%s)\n", assmble);
                       }
                       else if (strcmp(assmble, "?") == 0)
                       {
                              ReadOutputbuf();
                       }
                       else
                       {
                              if (awgWrite(assmble) < 0)
                              {
                                     gpiberr("Write Error:");
                                     continue;
                              }
                              if (chkquery(assmble) == QUERY)
                                     ReadOutputbuf();
                       }
                }
                else if (rflag == RD3)
                {
                       WrtFtoG(rfile);
                }
        }
}
```

```
/*
 * Input command line is parsed and stored into following memory.
 *  *argv[]   – command and arguments delimited by space.
 *  argc      – separated arguments count.
 *  assemble  – concatenation of *argv[].
 *  rflag     – '>', '>>', '<', or none.
 *  rfile     – input or output file to be redirected.
 *
 * If the string '!!' exists in command line, it is replaced with
 * previous command line.
 */
inputline()
{
      int    i;           /* loop index       */

      for (;;)
      {
            printf("AWG >> ");
            if (fgets(readbuf, MAX_BUF, ifd) == NULL)      /* Read one line */
            {
                  if (ifd == stdin)
                  {
                        fprintf(stderr, "Detected system error:");
                        fprintf(stderr, "restart the program\n");
                        exit (1);
                  }
                  fclose(ifd);
                  ifd = stdin;
                  continue;
            }
            if (ifd != stdin)
                  printf("%s", readbuf);
            chkreplace(readbuf);
            setarg(replace);                        /* Parse input line */
            if (!(argc < 1 && rflag == RD0))        /* Check input
*/
                  break;
      }

      assmble[0] = '\0';
      for (i = 0; i < argc; i++)              /* Assemble line    */
      {
            strcat(assmble, argv[i]);
            if ((i+1) < argc)
                  strcat(assmble, " ");
      }
      strcpy(stack, assmble);
      if (rflag != RD0)
      {
            strcat(stack, " ");
            strcat(stack, (rflag == RD1)?">":(rflag == RD2)?">>":"<");
            strcat(stack, " ");
            strcat(stack, rfile);
      }
}
```

```
/*
 * Check for '!!' and replace if existed.
 */
chkreplace(s)
char   *s;
{
       char   *p = stack;
       char   *r = replace;
       int    cc = 0;                    /* flag: replaced or not   */

       /*
        * Replace !! with previous input line.
        */
       while (*s)
       {
              if (strncmp(s, "!!", 2) == 0)
              {
                     for (p = stack; *p;)
                            *r++ = *p++;
                     s++; s++; p= stack;
                     cc++;
              }
              else
                     *r++ = *s++;
       }
       *r = '\0';

       if (cc != 0)
              printf("%s", replace);
}

/*
 * Check whether '?' is included in input line.
 */
chkquery(s)
char   *s;
{
       for (; *s; s++)
              if (*s == '?')
                     return QUERY;       /* May be query or query is
                                             included in a line        */
       return NOQUERY;                   /* Set command(s) only       */
}

/*
 * Read from output queue, and write stdin or file.
 */
void   ReadOutputbuf()
{
       FILE   *tfd;
       char   *p;
       char   sc;          /* Store one character              */
       int    i;           /* Loop index                */
```

```
/*
 * Check for the MAV bit in the SBR.
 */
if (!(serialp() & 0x10))
{
       awgtmo(T1s);              /* Wait for further 100us  */
       awgwait(TIMO | SRQI | RQS | END);
       awgtmo(T10s);             /* Reset to 10s            */
       if (!(serialp() & 0x10))
       {
              fprintf(stderr,
                     "Nothing to take out in Output Queue!!\n");
              return;
       }
}

/*
 * Take out
 */
if (rflag == RD1 || rflag == RD2)
{
       for (;;)
       {
              if (awgRead(readbuf, MAX_BUF) < 0)
              {
                     gpiberr("Read Error:");
                     break;
              }
              for (p = readbuf, i = 0; i < ibcnt; i++)
                     putc(*p++, ofd);
              if (iostatus & (ERR | TIMO | END))
                     break;
       }
}
else                  /* Read Output Queue, and write to stdout     */
{
       sc = ' ';
       for (;;)
       {
              if (awgRead(readbuf, MAX_BUF) < 0)
              {
                     gpiberr("Read Error:");
                     break;
              }
              for (p = readbuf, i = 0; i < ibcnt; i++, p++)
              {
                     if (*p == ';')
                     {
                            sc = *p;
                     }
                     else
                     {
                            if (*p == ':' && sc == ';')
                                   putc('\n', ofd);
                            else if (sc == ';')
                                   putc(sc, ofd);
                            putc(*p, ofd);
```

```
                                    sc = *p;
                              }
                        }
                        if (iostatus & (ERR | TIMO | END))
                              break;
                  }
            }
}

/*
 * Parse command line.
 */
setarg(str)
char   *str;
{
      char   *s = str;
      char   *p = str;

      sflag = OFF;
      rflag = RD0;
      argc = 0;
      for (; *s; s++)
      {
            switch ((int)*s)
            {
            case ' ' :
                  sflag = OFF;
                  bnull(p, s);
                  break;
            case '\n' :
                  sflag = OFF;
                  bnull(p, s);
                  --s;
                  break;
            case '>' :
                  ropnum = argc - 1;
                  bnull(p, s);
                  if ( *(s+1) == '>')
                  {
                        rflag = RD2;
                        *++s = '\0';
                  }
                  else
                  {
                        rflag = RD1;
                  }
                  sflag = OFF;
                  s = getfile(++s);
                  s--;
                  break;
            case '<' :
                  ropnum = argc - 1;
                  bnull(p, s);
                  if ( *(s+1) == '<')
                        *++s = '\0';
                  rflag = RD3;
                  sflag = OFF;
```

```
                        s = getfile(++s);
                        s--;
                        break;
                default :
                        if (sflag == OFF)
                        {
                                sflag = ON;
                                argv[argc] = p = s;
                                ++argc;
                        }
                }
        }
}

/*
 * Put '\0' at the end of the command and arguments.
 */
void   bnull(p, s)
char   *p, *s;
{
        *s-- = '\0';
        for (; p < s; s--)
        {
                if (*s == ' ')
                        *s = '\0';
                else
                        return;
        }
}

/*
 * Extract file name placed after '<', '>', or '>>'.
 */
char   *getfile(s)
char   *s;
{
        for (; *s == ' '; s++)
                ;
        rfile = s;
        for (;*s && *s != '\n' && *s != ' ' && *s != '>' && *s != '<'; s++)
                ;
        *s = '\0';
        return ++s;
}
```

```
/*
 * 'view' built-in command.
 */
void   viewfile(n, name)
int    n;
char   *name;
{
       FILE   *tfd;
       int    c;
       if (n != 2)
       {
              fprintf(stderr, "usage:     view  file-name\n");
              return;
       }

       if ((tfd = fopen(name, "r")) == NULL)
       {
              fprintf(stderr, "can't open file (%s)\n", name);
              return;
       }
       while ((c = getc(tfd)) != EOF)
              putc(c, ofd);
       fclose(tfd);
}

/*
 * 'exec' built-in command.
 */
void   execfile(n, name)
int    n;
char   *name;
{
       FILE   *tfd;

       if (n != 2)
       {
              fprintf(stderr, "usage: exec  file-name\n");
              return;
       }

       if ((tfd = fopen(name, "r")) == NULL)
       {
              fprintf(stderr, "can't open file (%s)\n", name);
              return;
       }
       ifd = tfd;
}
```

```
/*
 * 'status' built-in command.
 */
statusbyte(n)
int    n;
{
       if (n != 1)
              fprintf(stderr, "Arguments are neglected!!\n\n");
       fprintf(ofd, "Status byte: (%X)h\n", serialp());
}

static char   hmessg[] = {"\n\
Intrv is a interactive communication program that sends GPIB commands to\n\
the instrument or executes built-in commands read from a terminal or a file\n\
in the external controller.\n\n\n\
[] GPIB commands:  all commands and queries defined in Programmer Manual\n\
are available.\n\
 The response message to the query is immediately output to the standard\n\
 output.\n\n\
[] Built-in commands:  following built-in commands are available:\n\n\
 help   -  displays this message.\n\
 view   -  displays the contents of the file in the external controller,\n\
        which is specified by an argument.\n\n\
\t\tusage:  view  path-name\n\n\
 exec   -  reads command lines from the file specified by an argument, instead\n
\
        of the standard input. If the EOF is detected, command lines are\n\
        read from standard input again.\n\n\
\t\tusage:  exec  description-file\n\n\
 status - reads status byte from the instrument.\n\n\
 resetes - resets the registers in the event/status reporting system to\n\
         default values for this program. Resetes must be used after you\n\
         change the values of the registers with the GPIB commands such\n\
         as :DESE, *ESE, etc.\n\n\
[] Redirection:      the standard input or standard output may be redirected
with\n\
 the following syntax:\n\n\
 < name\n\
\tOpen file name as a standard input. If the GPIB command is followed\n\
\tby, the contents of the file are transferred to the instrument,\n\
\twithout sending terminator, the continuation of sending the GPIB\n\
\tcommand.\n\n\
 > name\n\
\tThe file name is used as standard output. If the file does not exist\n\
\tthen it is created, if the file exists, its is overwritten (its\n\
\tprevious contents being lost.\n\n\
 >> name\n\
\tThe file name is used as standard output like '>', but its contents are\n\
\tplaced at the end of the file if it exists, or it is created if not\n\
\texisted.\n\n\
[] Others:  '!!' in a command line is replaced with previously input line.\n\n
\
"};
```

```
/*
 * 'help' built-in command.
 */
void   helpmessg(n)
int    n;
{
       char   *p = hmessg;

       if (n != 1)
       {
              fprintf(stderr, "Arguments are neglected!!\n\n");
       }
       while (*p)
              putc(*p++, ofd);
}

/*
 * 'resetes' built-in command.
 */
resetes(n)
int    n;
{
       if (n != 1)
              fprintf(stderr, "Arguments are neglected!!\n\n");
       if (awgWrite("*CLS;:DESE 59;*ESE 58;*SRE 48") < 0)
       {
              gpiberr("Write Error: can't set enable registers");
       }
}

/*
 * Read event queue if event is being stacked.
 *
 * '\n' is added after event code and event message.
 */
void   chkstatus()
{
       int    ccount = 0;
       int    i;
       char   *p;

       /*
        * Check for the ESB bit in the SBR.
        */
       if (!(serialp() & 0x20))
       {
              awgtmo(T1s);              /* Wait for further 100us        */
              awgwait(TIMO | SRQI | RQS | END);
              awgtmo(T10s);             /* Reset to 10s                  */
              if (!(serialp() & 0x20))
                     return;
       }
```

```
        /*
         * Prepare to take out.
         */
        if (awgWrite("HEADER OFF;*ESR?") < 0 || awgWrite(":ALLEV?") < 0)
        {
               gpiberr("Write Error:");
               return;
        }

        /*
         * Read the Event Queue.
         */
        for (;;)
        {
               if (awgRead(readbuf, MAX_BUF) < 0)
               {
                      gpiberr("Read Error:");
                      break;
               }
               for (p = readbuf, i = 0; i < ibcnt; i++, p++)
               {
                      putc(*p, ofd);
                      if (*p == ',' && ccount == 1)
                      {
                             putc('\n', ofd);
                             ccount = 0;
                      }
                      else if (*p == ',')
                             ccount++;
               }
               if (iostatus & (ERR | TIMO | END))
                      break;
        }
        if (awgWrite("HEADER ON") < 0)
        {
               gpiberr("Write Error:");
               return;
        }
}
```

```
/*
 * Open file for output.
 */
void   redirect()
{
       FILE   *tfd;        /* temporally file descriptor     */
       if (rflag == RD1)
       {
              if ((tfd = fopen(rfile, ”w”)) == NULL)
              {
                     fprintf(stderr, ”can’t open file (%s)＼n”, rfile);
                     return;
              }
       }
       else if (rflag == RD2)
       {
              if ((tfd = fopen(rfile, ”a”)) == NULL)
              {
                     fprintf(stderr, ”can’t open file (%s)＼n”, rfile);
                     return;
              }
       }
       ofd = tfd;
}

/*
 * Close file after redirection completes.
 */
tcerider()
{
       if (ofd != stdout)
       {
              fclose(ofd);
              ofd = stdout;
       }
}
```

サポート関数

本マニュアルに添付された全てのプログラムは、以下にリスト・アップする関数を使用しています。

```
/*
 * awglib.c  –      libraries of GPIB interfaces.
 */
# include    ”decl.h”

int    awgdev;             /* Gpib descriptor of AWG        */
int    extcdev;            /* Gpib descriptor of GPIB0      */
int    iostatus;           /* Save a value of ibsta                */

/*
 * Find GPIB devices
 */
open_dev()
{
       /*
        * Assign unique identifiers to the device DEV1 and to the board GPIB0,
        * store them in the variables ”awgdev” and ”extcdev”, respectively, and
        * check for errors. If DEV1 or GPIB0 is not defined, ibfind returns –1.
        */
       if((awgdev = ibfind(”DEV1”)) < 0 || (extcdev = ibfind(”GPIB0”)) < 0)
       {
              gpiberr(”Ibfind Error: Unable to find device/board!”);
              exit(0);
       }

       /*
        *  Clear the device and check for errors.
        */
       if(ibclr(awgdev) < 0 || ibsre(extcdev, 0) < 0)
       {
              gpiberr(”ibclr/ibsre Error: Unable to clear device/board!”);
              exit(0);
       }

       /*
        * Set up the Device Event Status Enable Register, Event Status Enable
        * Register, and Service Request Enable Register to enable status
        * events.
        */
       if (awgWrite(”DESE 255”) < 0 || awgWrite(”*ESE 255”) < 0 ||
           awgWrite(”*SRE 48”) < 0)
       {
              gpiberr(”GPIBWRITE Error: Unable to initialize device!”);
              exit(0);
       }
}

close_dev()
{
}
```

```
/*
 *  Read into the string from the device and wait for the
 *  read to finish.
 */
awgRead(resp, cnt)
char   *resp;
int    cnt;
{
       /*
        * Set the timeout for 10 seconds, send the command, and
        * wait for the instrument to finish processing the command.
        */
       ibtmo(awgdev, T10s);
       ibrd(awgdev, resp, cnt);
       iostatus = ibsta;
       resp[ibcnt] = '\0';
       /*
        * If ibwrt was successful, wait for completion.
        */
       if(ibsta >=0)
              ibwait(awgdev, CMPL);

       return ibsta;
}

/*
 *  Send the contents of the string to the device and wait
 *  for the write to finish.
 */
awgWrite(cmd)
char   *cmd;
{
       int    cnt = strlen(cmd);

       /*
        * Set the timeout for 10 seconds, send the command, and
        * wait for the instrument to finish processing the command.
        */
       ibtmo(awgdev, T10s);
       ibwrt (awgdev, cmd, cnt);

       /*
        * If ibwrt was successful, wait for completion.
        */
       if(ibsta >=0)
              ibwait(awgdev, CMPL);

       return ibsta;
}
```

```
/*
 * Read from GPIB device, and Write into a file.
 */
WrtGtoF(name)
char   *name;
{
       return ibrdf(awgdev, name);
}

/*
 * Read from a file, and write into GPIB device.
 */
WrtFtoG(name)
char   *name;
{
       return ibwrtf(awgdev, name);
}

/*
 * Get status byte.
 */
serialp()
{
       char   serial_poll = 0;

       ibrsp(awgdev, &serial_poll);
       return serial_poll & 0xff;
}

/*
 * Set time out.
 */
awgtmo(tm)
int    tm;
{
       ibtmo(awgdev, tm);
}

/*
 * Wait.
 */
awgwait(wt)
int    wt;
{
       ibwait(awgdev, wt);
}

/*
 * EOI control.
 */
eotcont(status)
{
       ibeot(awgdev, status);
}
```

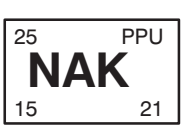
```
/*
 *  gpiberr.c – display error from defined error codes based on what
 *  is contained in ibsta. This routine would notify you that an IB
 *  call failed.
 */
#include ”decl.h”
#include <stdio.h>

void gpiberr(msg)
char   *msg;
{
       fprintf(stderr, ”%s\n”, msg);
       fprintf(stderr,      ”ibsta=(%X)h  <”, ibsta);

       if (ibsta & ERR ) fprintf(stderr, ” ERR”);
       if (ibsta & TIMO) fprintf(stderr, ” TIMO”);
       if (ibsta & END ) fprintf(stderr, ” END”);
       if (ibsta & SRQI) fprintf(stderr, ” SRQI”);
       if (ibsta & RQS ) fprintf(stderr, ” RQS”);
       if (ibsta & CMPL) fprintf(stderr, ” CMPL”);
       if (ibsta & LOK ) fprintf(stderr, ” LOK”);
       if (ibsta & REM ) fprintf(stderr, ” REM”);
       if (ibsta & CIC ) fprintf(stderr, ” CIC”);
       if (ibsta & ATN ) fprintf(stderr, ” ATN”);
       if (ibsta & TACS) fprintf(stderr, ” TACS”);
       if (ibsta & LACS) fprintf(stderr, ” LACS”);
       if (ibsta & DTAS) fprintf(stderr, ” DTAS”);
       if (ibsta & DCAS) fprintf(stderr, ” DCAS”);
       fprintf(stderr, ” >\n”);

       fprintf(stderr, ”iberr= %d”, iberr);
       if (iberr == EDVR) fprintf(stderr, ” EDVR <DOS Error>\n”);
       if (iberr == ECIC) fprintf(stderr, ” ECIC <Not CIC>\n”);
       if (iberr == ENOL) fprintf(stderr, ” ENOL <No Listener>\n”);
       if (iberr == EADR) fprintf(stderr, ” EADR <Address error>\n”);
       if (iberr == EARG) fprintf(stderr, ” EARG <Invalid argument>\n”);
       if (iberr == ESAC) fprintf(stderr, ” ESAC <Not Sys Ctrlr>\n”);
       if (iberr == EABO) fprintf(stderr, ” EABO <Op. aborted>\n”);

       if (iberr == ENEB) fprintf(stderr, ” ENEB <No GPIB board>\n”);
       if (iberr == EOIP) fprintf(stderr, ” EOIP <Async I/O in prg>\n”);
       if (iberr == ECAP) fprintf(stderr, ” ECAP <No capability>\n”);
       if (iberr == EFSO) fprintf(stderr, ” EFSO <File sys. error>\n”);
       if (iberr == EBUS) fprintf(stderr, ” EBUS <Command error>\n”);
       if (iberr == ESTB) fprintf(stderr, ” ESTB <Status byte lost>\n”);
       if (iberr == ESRQ) fprintf(stderr, ” ESRQ <SRQ stuck on>\n”);
/*
       if (iberr == ETAB) fprintf(stderr, ” ETAB <Table Overflow>\n”);
*/
}

/*
 * exit.h
 */
# define            exit(x)                    {awgWrite(”:HEADER    ON;:UNLOCK
ALL”);exit(x);}
```

付　録

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	40 SP 32 0	60 0 48 16	100 @ 64 0	120 P 80 16	140 ‘ 96 0	160 p 112 16
1	1 GTL SOH 1 1	21 LLO DC1 17	41 ! 33 1	61 1 49 17	101 A 65 1	121 Q 81 17	141 a 97 1	161 q 113 17
2	2 STX 2 2	22 DC2 18	42 ” 34 2	62 2 50 18	102 B 66 2	122 R 82 18	142 b 98 2	162 r 114 18
3	3 ETX 3 3	23 DC3 19	43 # 35 3	63 3 51 19	103 C 67 3	123 S 83 19	143 c 99 3	163 s 115 19
4	4 SDC EOT 4 4	24 DCL DC4 20	44 $ 36 4	64 4 52 20	104 D 68 4	124 T 84 20	144 d 100 4	164 t 116 20
5	5 PPC ENQ 5 5	25 PPU NAK 21	45 % 37 5	65 5 53 21	105 E 69 5	125 U 85 21	145 e 101 5	165 u 117 21
6	6 ACK 6 6	26 SYN 22	46 & 38 6	66 6 54 22	106 F 70 6	126 V 86 22	146 f 102 6	166 v 118 22
7	7 BEL 7 7	27 ETB 23	47 ’ 39 7	67 7 55 23	107 G 71 7	127 W 87 23	147 g 103 7	167 w 119 23
8	10 GET BS 8 8	30 SPE CAN 24	50 (40 8	70 8 56 24	110 H 72 8	130 X 88 24	150 h 104 8	170 x 120 24
9	11 TCT HT 9 9	31 SPD EM 25	51) 41 9	71 9 57 25	111 I 73 9	131 Y 89 25	151 i 105 9	171 y 121 25
A	12 LF 10 A	32 SUB 26	52 * 42 10	72 : 58 26	112 J 74 10	132 Z 90 26	152 j 106 10	172 z 122 26
B	13 VT 11 B	33 ESC 27	53 + 43 11	73 ; 59 27	113 K 75 11	133 [91 27	153 k 107 11	173 { 123 27
C	14 FF 12 C	34 FS 28	54 , 44 12	74 < 60 28	114 L 76 12	134 \ 92 28	154 l 108 12	174 | 124 28
D	15 CR 13 D	35 GS 29	55 – 45 13	75 = 61 29	115 M 77 13	135] 93 29	155 m 109 13	175 } 125 29
E	16 SO 14 E	36 RS 30	56 . 46 14	76 > 62 30	116 N 78 14	136 ^ 94 30	156 n 110 14	176 ~ 126 30
F	17 SI 15 F	37 US 31	57 / 47 15	77 ? 63 UNL	117 O 79 15	137 – 95 UNT	157 o 111 15	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



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、メッセージを送信したり、メッセージを受信したり、あるいはメッセージに従って機器を制御する機能です。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 の機能を全て満足します。
Source Handshake（SH）	SH1	SH の機能を全て満足します。
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 の機能を全て満足します。
Remote/Local（RL）	RL1	RL の機能を全て満足します。
Service Request（SR）	SR1	SR の機能を全て満足します。
Parallel Poll（PP）	PP0	サポートしません。
Device Trigger（DT）	DT1	DT の機能を全て満足します。
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、後述「インタフェース・メッセージ」を参照ください。

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全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全ての機器を、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全ての機器に対して、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	
DIAG:SELect	ALL
SELFcal:SELect	ALL
DISPLAY コマンド	
DISPlay:BRIGhtness	70
DISPlay:CATalog:ORDER	NAME1
DISPlay:CLOCk	0
DISPlay:MENU:STEUP:FORMat	GRAPHICS
DISPlay:MESSAGE:SHOW	GRAPHICS
FG コマンド	
FG:CHx:AMPLitude	1.000
FG:CHx:OFFSet	0.000
FG:CHx:POLarity	NORMAL
FG:CHx:SHAPe	SINUSOID
FG:STATe	0
FG:FREQuency	2.500E+06 (AWG2020/21型) 1.000E+06 (AWG2010/11型) 200.0E+03 (AWG2005型) 200.000E+03 (AWG2005 opt.05型) 10.00000E+06 (AWG2040/41型)
HARDCOPY コマンド	
HCOPy:FORMat	BMP
HCOPy:PORT	DISK
MEMORY コマンド	
DISK:FORMat:TYPE	HD3
MMEMory:MSIS	DISK
MMEMory:ALOad:MSIS	DISK
MMEMory:ALOad:STATe	0

表 C-1：デフォルト設定値（続き）

ヘッダ	設定値
MODE コマンド	
CONFigure	MASTER (AWG2005型のみ)
MODE	CONTINUOUS
TRIGger:HOLDoff (AWG2010/11/20/21型)	LONG
TRIGger:IMPedance	HIGH
TRIGger:LEVel	1.4
TRIGger:POLarity	POSITIVE
TRIGger:SLOPe	POSITIVE
OUTPUT コマンド	
OUTPut:CHx:STATe	0 (AWG2005/10/11/20/21型)
OUTPut:CH1:INVerted:STATE	0 (AWG2040/41型)
OUTPut:CH1:NORMal:STATE	0 (AWG2040/41型)
OUTPut:SYNC	END
SETUP コマンド	
CLOCk:FREQuency	100.0E+06 (AWG2020/21型) 50.00E+06 (AWG2010/11型) 10.00E+06 (AWG2005型) 10.00000E+06 (AWG2005 opt.05型) 1.000000E+09 (AWG204/410型)
CLOCk:SOURce	INTERNAL
CLOCk:CH2:DIVider	1 (AWG2010/11/20/21型)
CLOCk:SWEep:DWELl	1.000E−03 (AWG2005 opt.05型)
CLOCk:SWEep:MODE	SCONTINUOUS (AWG2005 opt.05型)
CLOCk:SWEep:STATe	0 (AWG2005 opt.05型)
CLOCk:SWEep:TIME	1.000E+00 (AWG2005 opt.05型)
CLOCk:SWEep:TYPE	LINEAR (AWG2005 opt.05型)
CLOCk:SWEep:FREQuency:STARt	1.00000E+06 (AWG2005 opt.05型)
CLOCk:SWEep:FREQuency:STOP	20.0000E+06 (AWG2005 opt.05型)
CH1:MARKERLEVEL1:HIGH	2.0 (AWG2040/41型)
CH1:MARKERLEVEL1:LOW	0.0 (AWG204/410型)
CH1:MARKERLEVEL2:HIGH	2.0 (AWG2040/41型)
CH1:MARKERLEVEL2:LOW	0.0 (AWG2040/41型)
CH1:OPERation	NORMAL (AWG2005/10/11/20/21型)
CHx:AMPLitude	1.000
CHx:FILTer	THRU
CHx:OFFSet	0.000
CHx:TRACk:AMPLitude	OFF (AWG2005型)
CHx:TRACk:OFFSet	OFF (AWG2005型)
CHx:WAVeform	""

表 C-1：デフォルト設定値（続き）

ヘッダ	設定値
STATUS & EVENT コマンド	
DESE	255
*ESE	0
*PSC	1
*SRE	0
SYSTEM コマンド	
DEBug:SNOop:STATe	0
DEBug:SNOop:DELAy:TIME	0.2
HEADer	1
HWSequencer:MODE	ON (AWG2041型)
VERBose	1
WAVEFORM コマンド	
DATA:DESTination	”GPIB.WFM”
DATA:ENCDG	RPBINARY
DATA:SOURce	”CH1”
DATA:WIDTh	2
WFMPre:ENCDG	BIN
WFMPre:BN_FMT	RP
WFMPre:BYT_NR	2
WFMPre:BIT_NR	12
WFMPre:BYT_OR	MSB
WFMPre:CRVCHK	NONE