MT9085 シリーズ
アクセスマスタ
SCPI リモートコントロール
取扱説明書

第4版

・製品を適切・安全にご使用いただくために，製品をご使用になる前に，本書を必ずお読みください。
・本書に記載以外の各種注意事項は，MT9085シリーズアクセスマスタ取扱説明書に記載の事項に準じますので，そちらをお読みください。
・本書は製品とともに保管してください。

アンリツ株式会社

管理番号：M-W3972AW-4.0

安全情報の表示について

当社では人身事故や財産の損害を避けるために，危険の程度に応じて下記のようなシグナルワードを用いて安全に関する情報を提供しています。記述内容を十分に理解した上で機器を操作してください。
下記の表示およびシンボルは，そのすべてが本器に使用されているとは限りません。また，外観図などが本書に含まれるとき，製品に貼り付けたラベルなどがその図に記入されていない場合があります。

本書中の表示について

危険 回避しなければ，死亡または重傷に至る切迫した危険があることを示します。

警告 回避しなければ，死亡または重傷に至るおそれがある潜在的な危険があることを示します。

注意 回避しなければ，軽度または中程度の人体の傷害に至るおそれがある潜在的危険，または，物的損害の発生のみが予測されるような危険があることを示します。

機器に表示または本書に使用されるシンボルについて

機器の内部や操作箇所の近くに，または本書に，安全上および操作上の注意を喚起するための表示があります。
これらの表示に使用しているシンボルの意味についても十分に理解して，注意に従ってください。


禁止行為を示します。丸の中や近くに禁止内容が描かれています。


守るべき義務的行為を示します。丸の中や近くに守るべき内容が描かれています。


警告や注意を喚起することを示します。三角の中や近くにその内容が描かれています。


注意すべきことを示します。四角の中にその内容が書かれています。


このマークを付けた部品がリサイクル可能であることを示しています。

MT9085 シリーズ
アクセスマスタ
SCPI リモートコントロール取扱説明書

2018年 （平成30年） ９月４日 （初　版）
2020年 （令和２年） 10月21日 （第４版）

国外持出しに関する注意

1. 本製品は日本国内仕様であり，外国の安全規格などに準拠していない場合もありますので，国外へ持ち出して使用された場合，当社は一切の責任を負いかねます。
2. 本製品および添付マニュアル類は，輸出および国外持ち出しの際には，「外国為替及び外国貿易法」により，日本国政府の輸出許可や役務取引許可を必要とする場合があります。また，米国の「輸出管理規則」により，日本からの再輸出には米国政府の再輸出許可を必要とする場合があります。

   本製品や添付マニュアル類を輸出または国外持ち出しする場合は，事前に必ず当社の営業担当までご連絡ください。

   輸出規制を受ける製品やマニュアル類を廃棄処分する場合は，軍事用途等に不正使用されないように，破砕または裁断処理していただきますようお願い致します。

はじめに

MT9085 シリーズ アクセスマスタの取扱説明書は，本体，リモートコントロール，クイックガイドに分かれて構成されています。本書は，MT9085A/B/C アクセスマスタ（以下，アクセスマスタ）の SCPI コマンドによるリモート制御方法について記述したものです。

目次

コマンド

ここでは，アクセスマスタのリモートコントロールについて説明します。

1.1　リモートコントロールの紹介

リモートコントロール機能を使用することにより，制御用パーソナルコンピュータ（以下，PC）から通信インタフェースを介して測定器に命令を与え，測定器の設定をしたり，測定結果や測定器の状態を読み取ったりすることができます。

アクセスマスタは，通信インタフェースとしてイーサネットまたは Wi-Fi を使用します。

本書では，SCPI（Standard Commands for Programmable Instruments）規格の書式に従ったコマンドを説明します。
アクセスマスタを制御するときのコマンドをコントロールコマンドと呼び，アクセスマスタからデータを読み出すときのコマンドをクエリコマンドと呼びます。
コマンドは文字列で構成されます。アクセスマスタの波長を 1550 nm に設定するときは，次のコントロールコマンドを送信します。

```
SOURce:WAVelength 1550
```

クエリコマンドは文字列の最後に“?”（クエスチョンマーク）をつけます。アクセスマスタの距離レンジを読み取るときは，次のクエリコマンドを送信します。

```
SOURce:RANge?
```

クエリコマンドをアクセスマスタに送信すると，PC は次の文字列（レスポンス）を受信します。

```
100
```

アクセスマスタの距離レンジは，100 km であることがわかります。

アクセスマスタがリモート制御状態になると，ソフトキーに［ローカル操作］と表示されます。リモート制御中は，電源と［ローカル操作］を除くすべてのキーが操作できなくなります。リモート制御を解除するには，［ローカル操作］をタッチします。

1.2 用途

リモートコントロールの主な用途を以下に示します。

自動測定

パネルを操作する代わりに，プログラムを実行してアクセスマスタを制御することにより測定を自動化できます。

複数機器の制御

複数の機器をリモートコントロールすることにより，複数の被測定物の特性を同時に測定できます。

機器の遠隔操作

遠隔地にある測定器を通信回線を通じて制御することにより，測定データを収集できます。

ここでは，アクセスマスタのリモートコントロールを使用する前に準備することを説明します。

2.1　機器の準備

リモート制御をするには，次の機器が必要です。

- PC
- USB ネットワーク機器
- イーサネットケーブル
- 通信ソフトウェア

PC

イーサネットインタフェースがあり，通信ソフトウェアを実行できる PC が必要です。

USB ネットワーク機器

アクセスマスタの USB ポートに接続します。

種類	仕様
USB イーサネットコンバータ	USB1.1/2.0 対応，10/100BASE-T
Wi-Fi ドングル	USB1.1/2.0 対応，IEEE 802.11b/g/n

イーサネットケーブル

カテゴリ 5 以上のイーサネットケーブルを使用してください。

通信ソフトウェア

イーサネットアドレス，ポート番号を指定して通信するソフトウェアです。

2.2　機器の接続

アクセスマスタは，USB イーサネットコンバータまたは Wi-Fi ドングルを介してネットワークに接続します。

アクセスマスタの電源がオンのときに USB イーサネットコンバータを接続すると，測定が中断されてイーサネット設定画面が表示されます。
アクセスマスタの電源がオフのときに USB イーサネットコンバータを接続したときは，電源をオンにしたあとでトップメニューの［リモート設定］をタッチします。

IP アドレスなどの設定は電源を切った後も保存されます。次に電源を入れると前回の設定が読み込まれます。

2.2.1　イーサネットケーブルの接続

アクセスマスタの USB ポート（汎用）に USB イーサネットコンバータを介して，イーサネットケーブルを接続します。

図 2.2.1-1　イーサネットケーブルの接続

アクセスマスタと PC を直接接続する場合は，USB イーサネットコンバータの LAN ポートと PC の LAN ポートをクロスケーブルで接続します。複数の外部機器と接続する場合は，ネットワークハブを使用してストレートケーブルで接続します。

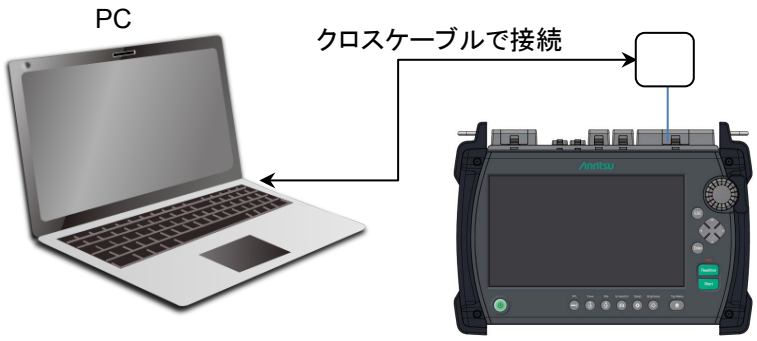


図 2.2.1-2　アクセスマスタと PC の直接接続例

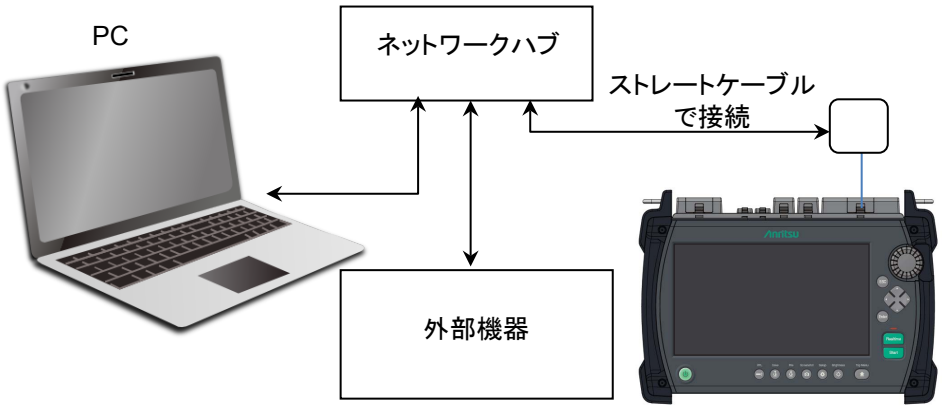


図 2.2.1-3　アクセスマスタと複数の外部機器の接続例

注：

外部機器同士の通信状態によっては，アクセスマスタと PC 間の通信がつながりにくくなる場合があります。安定した通信のためには，アクセスマスタと PC を直接接続することをお勧めします。

2.2.2　ドングルの接続

アクセスマスタの USB ポート（汎用）に Wi-Fi ドングルを接続します。

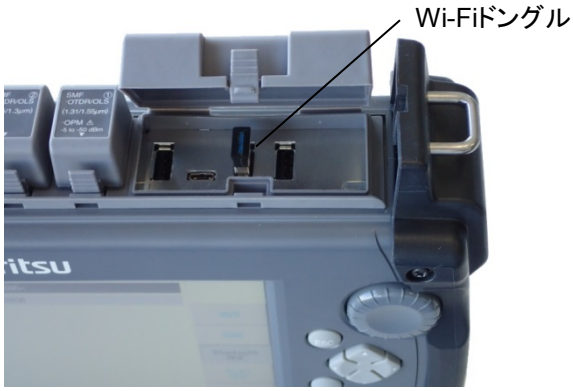


図 2.2.2-1　ドングルの接続

2.3　アクセスマスタのイーサネット設定

IP アドレスなどを，リモート設定画面で設定します。

アクセスマスタの IP アドレスを設定した後は，「2.4 PC の設定」を参照して，PC の IP アドレスも設定します。

2.3.1　イーサネットの設定

1. トップメニューの［リモート設定］をタッチします。
2. ［イーサネット設定］をタッチします。

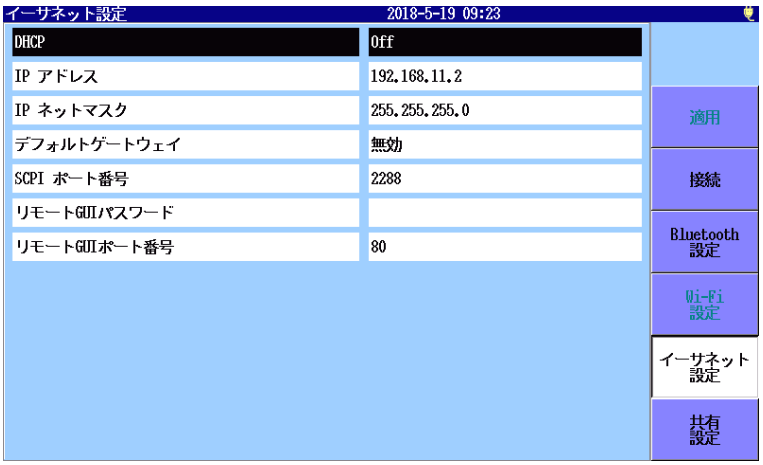


図 2.3.1-1　イーサネット設定画面

3. IP アドレスを自動で設定する場合は，［DHCP］をタッチして［On］に設定します。この場合は手順 8 に進みます。
4. ［IP アドレス］をタッチして，アクセスマスタの IP アドレスを設定します。
   設定範囲は 0.0.0.1 から 255.255.255.254 です。
   初期設定は 192.168.1.2 です。
5. ［IP ネットマスク］をタッチして，アクセスマスタの IP ネットマスクを設定します。
   設定範囲は 0.0.0.0 から 255.255.255.255 です。
   初期設定は 255.255.255.0 です。
6. ［デフォルトゲートウェイ］をタッチして，デフォルトゲートウェイの IP アドレスを設定します。
   設定範囲は 0.0.0.1 から 255.255.255.254 です。
   初期設定は無効です。

7. [SCPI ポート番号] をタッチして，SCPI コマンドの通信に使用するポート番号を設定します。
   設定範囲は 1 から 65535 です。初期設定は 2288 です。
8. 設定が終わったら，[適用] をタッチします。
9. [接続] をタッチします。
10. ネットワークに接続すると，アクセスマスタの画面にアイコンが表示されることを確認してください。

    手順 3 で DHCP を [On] に設定した場合は，イーサネット設定画面で IP アドレスと IP ネットマスクが確認できます。

図 2.3.1-2　ネットワーク接続アイコン

2.3.2　Wi-Fiの設定

次の手順で Wi-Fi を設定します。

1. トップメニューの [リモート設定] をタッチします。
2. [Wi-Fi 設定] をタッチします。
3. [選択されたネットワーク] をタッチします。

図 2.3.2-1　Wi-Fi 設定画面

4. [更新] をタッチすると，SSID の表示が更新されます。
接続するネットワークが表示されない場合は，[その他] をタッチして SSID を入力します。

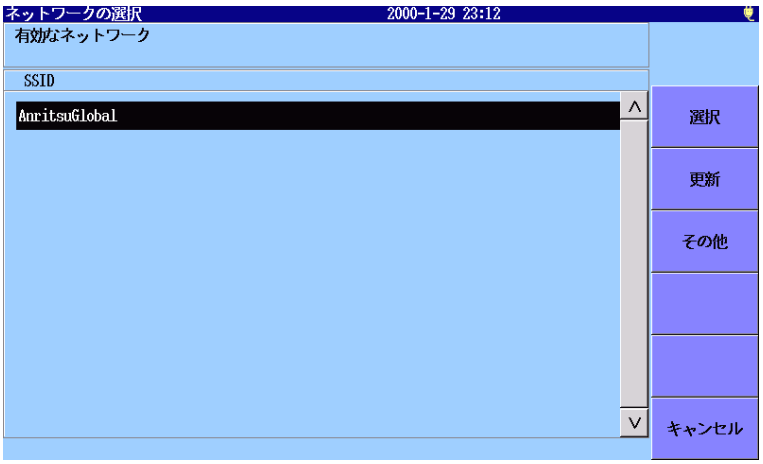


図 2.3.2-2　ネットワークの選択画面

5. 表示された SSID から接続するネットワークをタッチして，[選択] をタッチします。Wi-Fi 設定画面の [選択されたネットワーク] に SSID が表示されます。

6. IP アドレスを自動で設定する場合は，[DHCP] をタッチして [On] に設定します。この場合は手順 12 に進みます。

7. [IP アドレス] をタッチして，アクセスマスタの IP アドレスを設定します。
設定範囲は 0.0.0.1 から 255.255.255.254 です。
初期設定は 192.168.1.2 です。

8. [IP ネットマスク] をタッチして，アクセスマスタの IP ネットマスクを設定します。
設定範囲は 0.0.0.0 から 255.255.255.255 です。
初期設定は 255.255.255.0 です。

   IP ネットマスクは，設定範囲でも設定できない組み合わせがあります

9. [デフォルトゲートウェイ] をタッチして，デフォルトゲートウェイの IP アドレスを設定します。
設定範囲は 0.0.0.1 から 255.255.255.254 です。
初期設定は無効です。

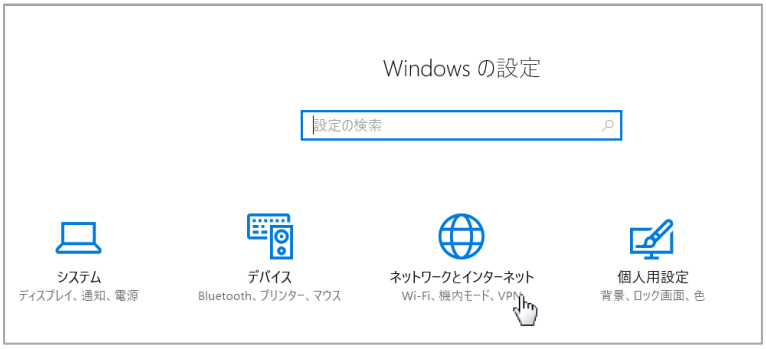
10. [SCPI ポート番号] をタッチして，SCPI コマンドの通信に使用するポート番号を設定します。
    設定範囲は 1 から 65535 です。初期設定は 2288 です。
11. 設定が終わったら，[適用] をタッチします
12. [接続] をタッチします。
13. ネットワークに接続すると，アクセスマスタの画面にアイコンが表示されることを確認してください。

    手順 6 で DHCP を [On] に設定した場合は，Wi-Fi 設定画面で IP アドレスと IP ネットマスクが確認できます。

2.4　PC の設定

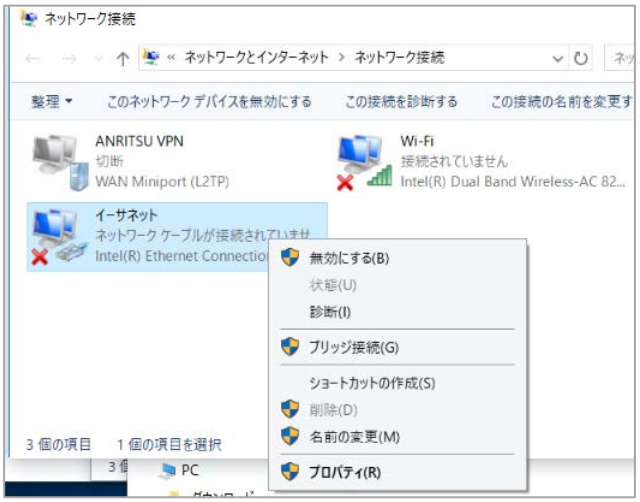
IP アドレスを設定する必要がある場合は，使用する PC の OS の IP アドレス設定手順に従ってください。以下は Windows 10 の画面イメージで紹介しています。

1. スタートメニューから［設定］をクリックし，［ネットワークとインターネット］をクリックします。



2. ［アダプターのオプションを変更する］をクリックします。

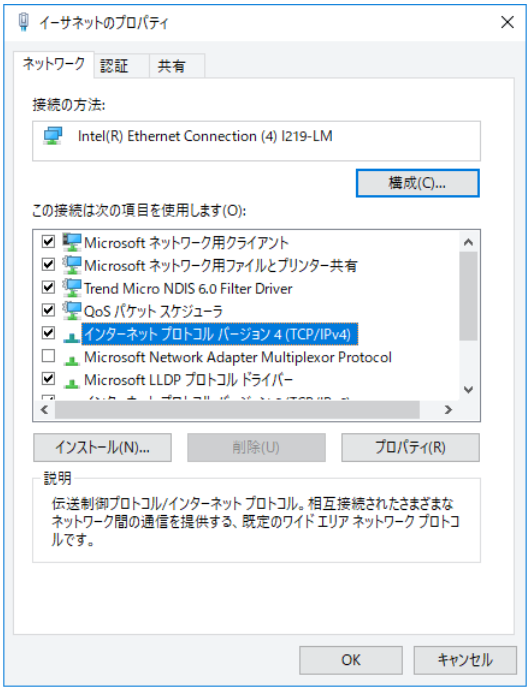
3. USB イーサネットコンバータを使用する場合は，［イーサネット］を右クリックし，［プロパティ］をクリックします。



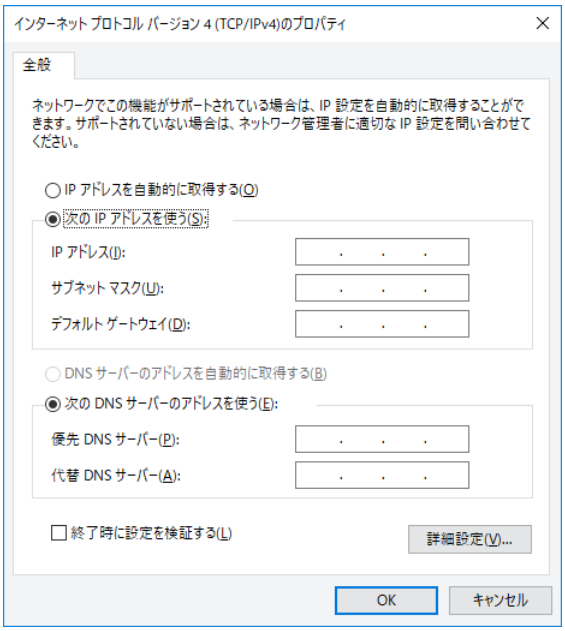
注：

Wi-Fi ドングルを使用する場合は，［Wi-Fi］を右クリックします。

4. [イーサネットのプロパティ] ダイアログボックスで, [インターネットプロトコルバージョン 4 (TCP/IPv4)] をクリックし, [プロパティ] をクリックします。

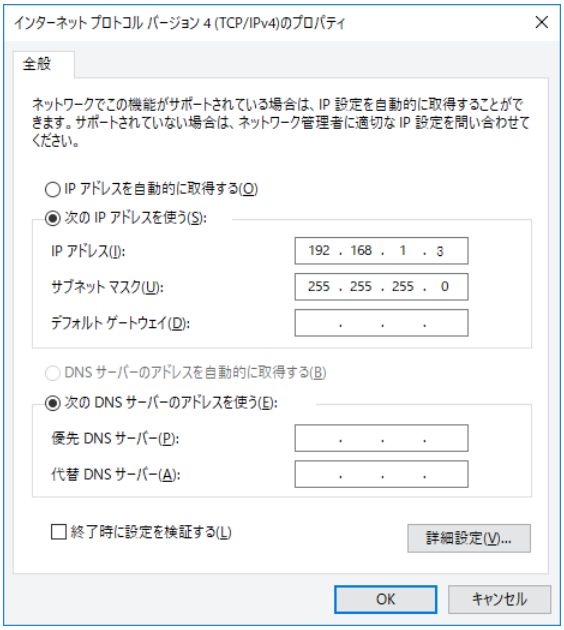


5. [インターネット プロトコル 4 (TCP/IPv4)のプロパティ] ダイアログボックスで, [次の IP アドレスを使う(S)] をクリックします。



6. IP アドレスには，アクセスマスタで設定した IP アドレスと異なるネットワークアドレスを設定します。ここでは，次のとおり設定し,[OK] をクリックします。

IP アドレス:　192.168.1.3
サブネットマスク:　255.255.255.0



7. [イーサネットのプロパティ] ダイアログボックスで，[OK] をクリックします。

2.5　接続の確認

1. スタートメニューの，[すべてのプログラム] をクリックします。
2. [アクセサリ] をクリックします。
3. [コマンド] をクリックします。
4. 次の画面のようにコマンドを入力します。
   この例ではアクセスマスタの IP アドレスを 192.168.1.2 に設定しています。

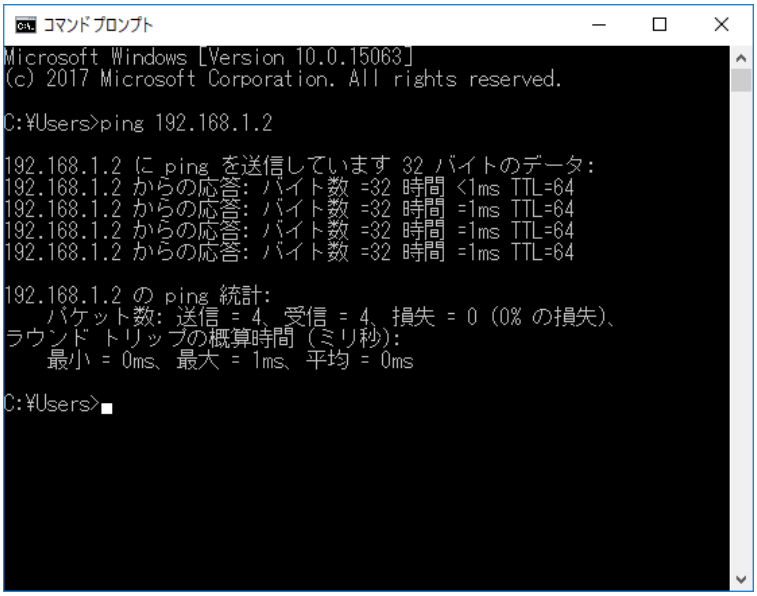


図 2.5-1　Ping コマンドの実行

5. 「タイムアウトしました」と表示されないことを確認します。
   タイムアウトが表示された場合は，以下の設定が正しいか確認してください。
   - IP アドレス, IP ネットマスク，デフォルトゲートウェイ, SCPI ポート番号
   - ケーブル接続
   - USB イーサネットコンバータの接続
   - Wi-Fi ドングルの接続

2.6　メッセージの書式

メッセージは，メッセージ本体の文字列とメッセージの終了を示す文字列で構成されます。アクセスマスタへのメッセージを送信するときは CR+LF で行を終了します。アクセスマスタから受信するメッセージも CR+LF で終了します。

2.6.1　メッセージの種類

メッセージには送信方向によって次の種類があります。

プログラムメッセージ:
PC からアクセスマスタに送信されるメッセージです。
プログラムメッセージは，次の 2 つに分類できます。

- コマンド
  測定条件の設定や，測定の開始などに使用します。
- クエリ
  アクセスマスタの状態や設定を問い合わせます。
  クエリを送信すると，アクセスマスタはクエリに対するレスポンスメッセージを作成します。

メッセージには，ヘッダ部分とデータ部分があり，その間には半角スペースが入ります。プログラムメッセージには，必ずヘッダーがありますがデータは無いことがあります。

レスポンスメッセージ:
アクセスマスタから PC に送信されるメッセージです。

レスポンスメッセージには，必ずデータがありますがヘッダは無いことがあります。

2.6.2　ヘッダの構成

ヘッダは英数字とアンダーバーで構成され，先頭文字は英文字です。IEEE 488.2 で定義されているコマンドは，ヘッダの先頭アステリスク (*) が付きます。アルファベットの大文字と小文字は区別されません。

ヘッダのみのコマンド

```
*RST
*CLS
INITiate:AUTo
```

ヘッダとデータからなるコマンド

```
*ESE 255
```

```
SOURce:WAVelength 1310
INSTrument:SELect OTDR_STD
```

データが複数あるコマンドは，データの区切りにコンマ(,)を使用します。

例:

```
SENSe:LSALeft 0.0,10.0
SENS:ANAL:PAR 0.05,-60.0,3.0
```

クエリのヘッダは，最後にクエスチョンマークをつけます。

例:

```
*ESR?
TOPMenu:UNIT:CATalog?
SOUR:PULS:ENH?
```

複数のプログラムメッセージを連結するときは，セミコロンでメッセージを区切ります。連結できるメッセージ数は 12 までです。メッセージを 13 個以上連結すると，13 番目以降のメッセージは処理されません。

例:

```
SOUR:WAV 1310;SOUR:RAN 100;SOUR:RES 0;INIT;*WAI
```

2.6.3 データの書式

データの書式には，文字データ，数値データ，バイナリデータがあります。

文字データ

文字データは，アスキーコードの文字列です。
次の例は，トップメニューから［光パルス試験（通常試験)］を選択するプログラムメッセージです。

例:

```
INSTrument:SELect OTDR_STD
```

数値データ

数値データは 10 進数で記述します。

例:

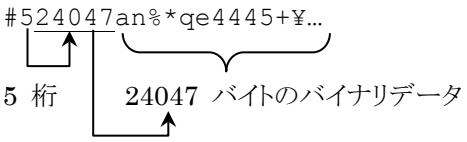
```
SYSTem:LOCK 1
TOPM:UNIT 2
SENSe:LOSS:MODE 5
SENSe:FIB:IOR 1.500000
SENSe:VOFF -10.0
```

整数，固定小数点または浮動小数点で書くことができます。

バイナリデータ

バイナリデータは，先頭文字が番号記号（#）で始まり，データ長を示す数字の後にデータを続けます。

例:



2.7　機器の状態確認

アクセスマスタには，エラーの発生やコマンドの実行状況など機器の状態を表示するレジスタがあります。ここではそのレジスタを説明します。

2.7.1　レジスタ構造

アクセスマスタの状態を表示するレジスタの構成を図 2.7.1-1 に示します

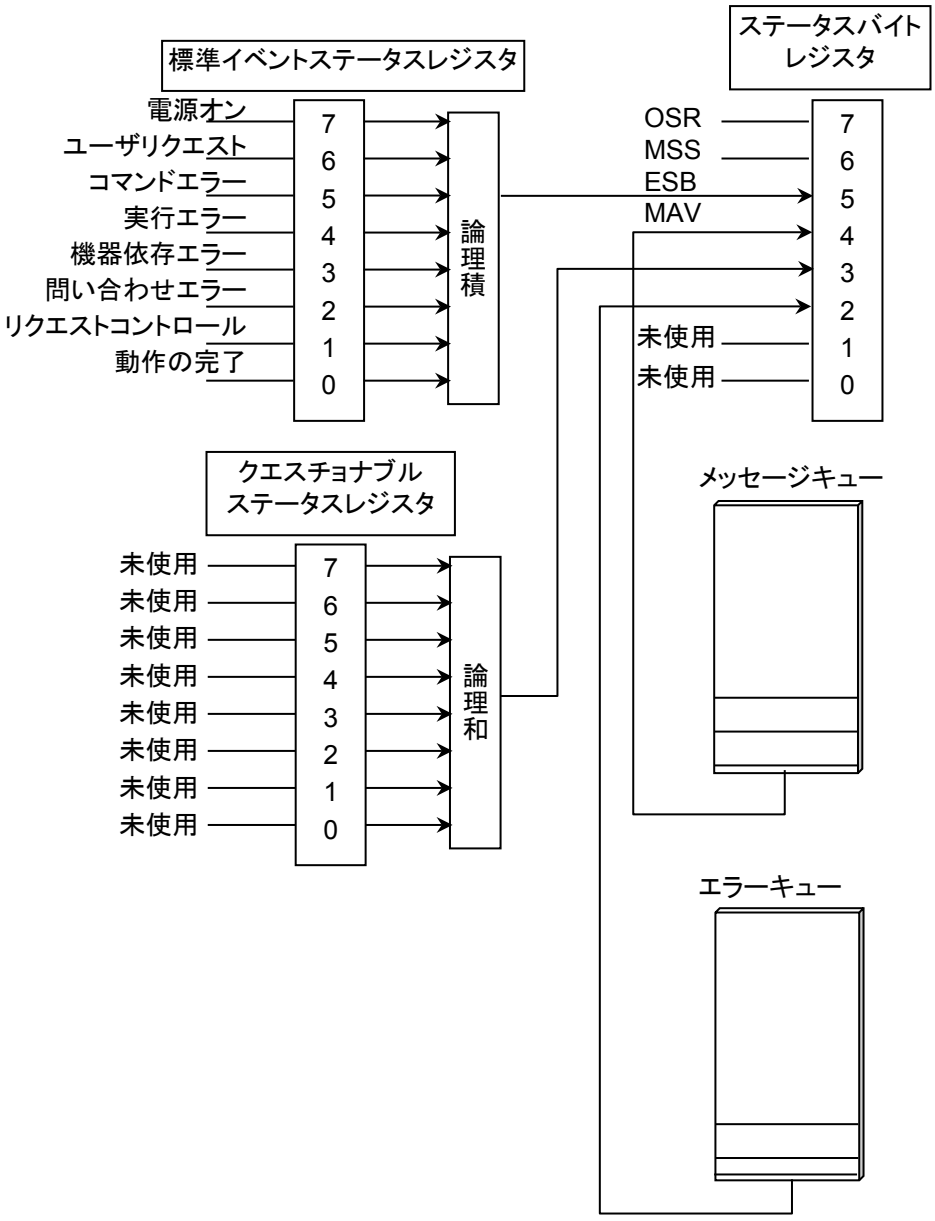


図 2.7.1-1　レジスタの構成

各レジスタは 8 ビットのデータを持ちます。レジスタの出力値は，表 2.7.1-1 に示す各ビットの 10 進数の値を合計した値です。

表 2.7.1-1　レジスタのビットの 10 進数変換値

ビット	10 進数の値	ビット	10 進数の値
0	1	4	16
1	2	5	32
2	4	6	64
3	8	7	128

ステータスバイトレジスタのデータを出力するためには，サービスリクエストイネーブルレジスタの該当するビットを，1 に設定してください。ステータスバイトレジスタを読み取る前に，サービスリクエストイネーブルレジスタを設定します。*STB コマンドで 2 つのレジスタの論理積を読み取ることができます。

標準イベンステータストレジスタには対応するイネーブルレジスタがあります。イベントレジスタとイベントイネーブルレジスタの論理積を求めて，それらの論理和がステータスバイトレジスタのビット 5 に出力されます。

クエスチョナブルステータスレジスタ
このレジスタは将来使用する予定で，未使用です。

メッセージキュー
メッセージキューは通常は空です。アクセスマスタからのレスポンスメッセージはただちに送信されます。PC からのメッセージは 12 個まで保持できます。

エラーキュー
アクセスマスタからのエラーメッセージは 12 個まで保持できます。

2.7.2　ステータスバイトレジスタ

ステータスバイトレジスタの各ビットの定義を，次の表に示します。

表 2.7.2-1　ステータスバイトレジスタのビット定義

ビット	説明
7	OSR (Operation Status Register) アクセスマスタの動作状態を表示します。光パルス試験実行中のときに 1 になります。実行状態レジスタと，そのイベントイネーブルレジスタとの論理積の，各ビットの論理和です。
6	MSS (Master Summary Register) ステータスバイトレジスタが 0 かそうでないかを表示します。ステータスバイトレジスタとサービスリクエストイネーブルレジスタとの論理積の，ビット 7 とビット 5～0 の論理和です。
5	ESB (Event Status Bit) 標準イベントレジスタと，標準イベントイネーブルレジスタとの論理積の，各ビットの論理和です。
4	MAV (Message AVailable summary) アクセスマスタではメッセージがただちに送信されるため，常に 0 になります。
3	クエスチョナブルステータスレジスタ 未使用：常に 0 になります。
2	エラー / イベントキュー エラーキューにエラーメッセージがあるときに 1 になります。
1	未使用：常に 0 になります。
0	未使用：常に 0 になります。

ステータスバイトレジスタのビット 7 からビット 0 までは，*STB で読むことができます。

サービスリクエストイネーブルレジスタの設定と読み取りには，共通コマンドの*SRE，*SRE?を使用します。ステータスバイトレジスタのデータを出力するためには，サービスリクエストイネーブルレジスタの該当するビットを，1 に設定してください。

ステータスバイトレジスタのビット 5，3，および 2 は，共通コマンド*CLS で 0 にできます。
コマンドの後に*CLS を送信したとき，または*CLS の後にクエリを送信したときは，送信キューがクリアされてビット 4 が 0 になります。

サービスリクエストイネーブルレジスタは，*CLS で 0 にできませんので *SRE を使用してください。

2.7.3　イベントレジスタ

標準イベントステータスレジスタの各ビットの定義は，次のとおりです。

表 2.7.3-1　標準イベントステータスレジスタのビット定義

ビット	説明
7	電源オン 電源が投入されると 1 になります。
6	ユーザリクエスト 未使用　常に 0 になります。
5	コマンドエラー 定義されていないプログラムメッセージ，または文法エラーやスペルミスで実行できないメッセージを受信すると 1 になります。
4	実行エラー パラメータの値が範囲外のため，プログラムメッセージを実行できないときに 1 になります。
3	機器依存エラー コマンドエラー，実行エラー，およびクエリエラー以外のエラーで 1 になります。
2	クエリエラー データ読み取り時に，出力キューにデータが存在しないか何らかの理由で出力キューにエラーがあるときに 1 になります。
1	リクエストコントロール 未使用　常に 0 になります。
0	操作の完了 イベントテーブルに対する操作が完了したかどうかを表示します。 読み取りができるのは*OPC コマンドだけです。

標準イベントステータスレジスタのビット 7 からビット 0 は，*ESR?で読み取ることができます。読み取ると，標準イベントステータスレジスタは 0 になります。

標準イベントステータスイネーブルレジスタの設定と読み取りには，*ESE と*ESE?を使用します。標準イベントステータスレジスタのデータを出力するためには，イネーブルレジスタの該当するビットを 1 に設定してください。

ビット 0 は，共通コマンド*OPC で読み取ることができます。

標準レジスタは，共通コマンド*CLS で 0 にできます。

2.8　メッセージの同期を制御する

アクセスマスタが以下のメッセージ（最大 12 種類）を測定中または測定結果の解析中に受信できます。しかし，アクセスマスタの前の処理が終了する前に，測定パラメータを変更するメッセージを送信してもメッセージは無視されて測定条件が正しく設定されません。

次のプログラムメッセージは，波長 1310 nm でアベレージ測定実行中に波長を 1550 nm に変更します。

```
SOUR:WAV 1310;INIT;SOUR:WAV 1550
```

このメッセージをアクセスマスタに送信したときの，メッセージの実行順序を図 2.8-1 に示します。
最初に波長を設定した後，INIT を送信すると測定を開始します。波長を 1550 nm に変更するコマンドをアクセスマスタに送信しても，測定中は処理されません。

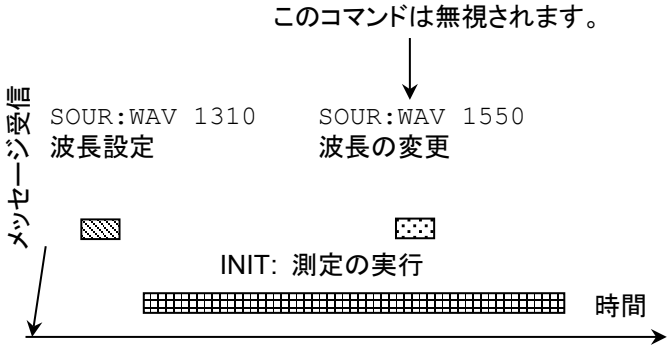


図 2.8-1　メッセージの実行順序

このようなときには，メッセージの実行順序を指定する必要があります。先に送信したメッセージの処理が完了してから，次のコマンドメッセージを処理する制御を同期制御と呼びます。

同期制御には，次の方法があります。

- *WAI コマンドの使用
- *OPC? クエリの使用
- *OPC コマンドと*ESR?クエリの使用
- 実行完了を問い合わせるクエリの使用

*WAI コマンド，*OPC?クエリ，*OPC コマンド，および*ESR?クエリは，すべてのメッセージに対して使用できます。

*WAI を使用する

共通コマンド*WAI は，*WAI の前に送信したメッセージの処理が終了するまで，*WAI の次に送信するコマンドの実行を待ちます。

使用例:　　INIT;*WAI;SOUR:WAV 1550;INIT

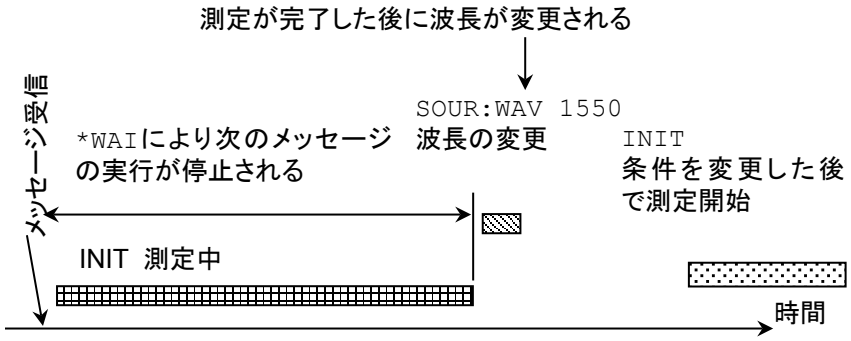


図 2.8-2　*WAI による同期制御

*OPC? を使用する

共通コマンド*OPC?は，メッセージ処理の実行完了を示すビット（OPC ビット）を問い合わせます。

使用例:

INIT	測定（平均化）を開始します。
*OPC?	操作が完了したか問い合わせ，1 になるまで待ちます。
> 1	レスポンスが 1 になると，測定終了です。
SENS:TRAC:READY?	測定したトレースデータが存在するか問い合わせます。
> 1	レスポンスが 1 になると，トレースデータの転送準備完了です。
TRAC:LOAD:SOR?	アクセスマスタからトレースデータを取得します。
SOUR:WAV 1550	波長を変更します。
INIT	新しい測定条件で測定を開始します。

*OPC と*ESR?を使用する

共通コマンド*OPC を実行すると，メッセージ処理の実行完了で標準イベントステータスレジスタのビット 0 (OPC ビット) が 1 となります。

使用例:

*CLS	OPC ビットを 0 にします。
*ESE 1	標準イベントステータスレジスタを 1 にします。
INIT	測定を実行します。
*OPC	標準イベントステータスレジスタの OPC ビット (ビット 0) を，測定が終了したら 1 になるように設定します。
*ESR?	標準イベントステータスレジスタの値を問い合わせます。
> 0	レスポンスが 0 の場合は測定中です。
*ESR?	標準イベントステータスレジスタの値を問い合わせます。
> 1	レスポンスが 1 になると，測定終了です。
SENS:TRAC:READY?	測定したトレースデータが存在するか問い合わせます。
> 1	レスポンスが 1 になると，トレースデータの転送準備完了です。
TRAC:LOAD:SOR?	アクセスマスタからトレースデータを取得します。
SOUR:WAV 1550	波長を変更します。
INIT	新しい測定条件で測定を開始します。

2.9　SM ポートと MM ポートを切り替える

オプション 063 では，シングルモードファイバ（測定波長 1310/1550 nm）とマルチモードファイバ（測定波長 850/1300 nm）の 2 つのポートがあります。

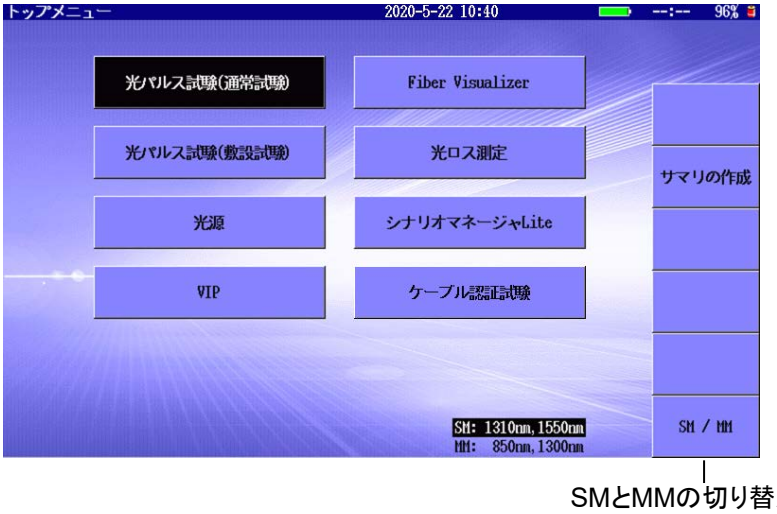


図 2.9-1　トップメニュー

以下の方法で SM と MM を切り替えます。

使用例 1: MT9085B-063 (MM 850/1300 nm, SM 1310/1550 nm)

`TOPMenu:UNIT:CATalog:FULL?`	ポートを問い合わせます。
`>MM,1,SM,2`	レスポンスが MM, 1, SM, 2: MM ポートが 1，SM ポートが 2 に割り当てられています。
`TOPMenu:UNIT:NSELect?`	現在の測定ポートを問い合わせます。
`>1`	レスポンスが 1 の場合，MM ポートが設定されています。
`TOPM:UNIT:NSEL 2`	SM ポートに変更します。

使用例 2: MT9085B-063 (MM 850/1300 nm, SM 1310/1550 nm)

TOPMenu:UNIT:CATalog?	ポートを問い合わせます。
>MM,SM	レスポンスが MM, SM: MM ポートと SM ポートがあります。
TOPMenu:UNIT?	現在の測定ポートを問い合わせます。
>MM	レスポンスが MM: MM ポートが設定されています。
TOPM:UNIT:SEL SM	SM ポートに変更します。

2.10 トップメニューから別の測定に移動する

アクセスマスタは，光パルス試験の他に光源やパワーメータなどの測定機能があります。これらの測定をするには，測定機能を切り替える必要があります。
現時点でアクセスマスタのリモートコントロールは，光パルス試験（通常測定）とトップメニューのみに対応しています。
この例では，光パルス試験（通常測定）をトップメニューから別の測定に切り替える方法を説明します。
トップメニューと光パルス試験（通常測定）の切り替えには以下が必要です。

使用例 1: MT9085C-053 の 1310 nm で測定するには

`INSTrument:CATalog:FULL?`	測定モードがリモートコントロールに対応しているか問い合わせます。
`>TOP_MENU,1,OTDR_STD,2`	TOP_MENU, 1, OTDR_STD, 2: モード 1 でトップメニュー， モード 2 で光パルス試験（通常測定）に対応します。
`INSTrument:NSELect?`	現在のモードを問い合わせます。
`>1`	レスポンス 1: トップメニュー
`INST:NSEL 2`	光パルス試験（通常測定）に切り替えます。 注：他の測定モードは操作できません。
`INST:STAT 1`	光パルス試験（通常測定）を有効にします。
`SOUR:WAV 1310`	波長を 1310 nm に設定します。

使用例 2: MT9085B-063 (MM 850/1300 nm, SM 1310/1550 nm) の SM ポートで測定した後で MM ポートで測定するには

`INSTrument:NSELect?`	現在のモードを問い合わせます。
`>2`	レスポンス 2: 光パルス試験 (通常測定)
`INST:NSEL 1`	トップメニューに移動します。
`TOPMenu:UNIT:CATalog:FULL?`	ポートを問い合わせます。
`>MM,1,SM,2`	レスポンスが MM, 1, SM, 2: MM ポートが 1, SM ポートが 2 に割り当てられています。
`TOPM:UNIT:NSEL 1`	MM ポートに変更します。

第3章 プラットフォーム SCPI コマンド

ここでは，アクセスマスタのプラットフォーム部の SCPI コマンドについて説明します。
文法の説明で <wsp> は，スペースを表します。

3.1　IEEE 488.2 共通コマンド

*CLS

機能

*CLS はステータスバイトに通知される，すべてのイベントレジスタをクリアします。出力キューを除いて，ステータスバイトに通知される，すべてのキューが空になります。エラーキューも空になります。

標準イベントステータスレジスタとサービスリクエスト・イネーブルレジスタは，*CLS コマンドによって変更されません。

文法

```
*CLS
```

パラメータ

無し

レスポンスデータ

無し

使用例

```
*CLS
```

*ESE

機能

*ESE コマンドは，標準イベントステータスレジスタ・イネーブルレジスタの値を設定します。
イネーブルレジスタのビットを 1 に設定すると，イネーブルレジスタのビット対応する標準イベントステータスレジスタを読み取ることができます。標準イベントステータスレジスタ・イネーブルレジスタは電源投入時にクリアされます。*RST と*CLS コマンドは標準イベントステータスレジスタ・イネーブルレジスタを変更しません。

文法

```
*ESE<wsp><value>
```

パラメータ

レジスタのビットの値（単精度整数):

7: (MSB) 電源オン	128
6: ユーザリクエスト	64
5: コマンドエラー	32

4: 実行エラー	16
3: 機器依存エラー	8
2: 問い合わせエラー	4
1: リクエストコントロール	2
0: (LSB) 動作の完了	1

レスポンスデータ

無し

使用例

```
*ESE 21
```

*ESE?

機能

*ESE? クエリは，標準イベントステータスレジスタ・イネーブルレジスタの値を問い合わせます。
レジスタの情報は*ESE を参照してください。

文法

```
*ESE?
```

パラメータ

無し

レスポンスデータ

レジスタの値 (単精度整数)

使用例

```
*ESE?
> 21<END>
```

*ESR?

機能

*ESR? は標準イベントステータスレジスタの値を問い合わせます。 レジスタの値を読み取ると，標準イベントステータスレジスタはクリアされます。

文法

```
*ESR?
```

パラメータ

無し

レスポンスデータ

レジスタのビットの値（単精度整数):

7: (MSB) 電源オン	128
6: ユーザリクエスト	64
5: コマンドエラー	32
4: 実行エラー	16
3: 機器依存エラー	8
2: 問い合わせエラー	4
1: リクエストコントロール	2
0: (LSB) 動作の完了	1

使用例

```
*ESR?
> 22<END>
```

*IDN?

機能

*IDN? は機器の情報を問い合わせます。

文法

```
*IDN?
```

パラメータ

無し

レスポンスデータ

機器情報は文字列 <END>で終了します。

使用例

```
*IDN?
> ANRITSU,MT9085C,6260123456<END>
```

*OPC

機能

機器が*OPC を実行した後では OPC コマンド受付状態（OCAS: Operation Complete Command Active State）になります。動作待ちフラグ（Operation Pending flag）が真でない場合は，機器は OPC コマ

ンド待ち状態(OCIS: Operation Complete Command Idle State)に戻り，同時に標準イベントステータスレジスタの OPC ビットを 1 にします。OPC コマンドアイドル状態になります。
以下のイベントは，機器をOPCコマンド待ち状態(OCIS)に変更します。この場合，動作待ちフラグは 1 に設定されませんし，標準イベントステータスレジスタの OPC ビットは 1 になりません。

- 電源オン
- *CLS
- *RST

文法

```
*OPC
```

パラメータ

無し

レスポンスデータ

無し

使用例

```
*OPC
```

*OPC?

機能

OPC?を実行した後，機器は実行完了の問い合わせが有効な状態(OQAS: Operation Complete Query Active State) になります。動作待ちフラグ (Operation Pending flag) が真でない場合は，機器は OPC コマンド待ち状態(OCIS: Operation Complete Command Idle State) に戻り，同時に出力キューに 1 を出力します。OPC コマンドアイドル状態になります。
以下のイベントは，OPC クエリの処理を中止し，機器を処理が完了した状態にします。

- 電源オン

文法

```
*OPC?
```

パラメータ

無し

レスポンスデータ

1 <END>.

使用例

```
*OPC?
> 1<END>
```

*RST

機能

*RST コマンドはアクセスマスタをトップメニューに変更します。処理待ちの*OPC 処理はキャンセルされます。アクセスマスタは，コマンドの受付待ちの状態になります。*RST コマンドはエラーキューをクリアします。

以下は変化しません。

- 出力キュー
- サービスリクエスト・イネーブルレジスタ（SRE）
- 標準イベントステータスレジスタ・イネーブルレジスタ（ESE）

文法

```
*RST
```

パラメータ

無し

レスポンスデータ

無し

使用例

```
*RST
```

*SRE

機能

*SRE コマンドは，サービスリクエスト・イネーブルレジスタの値を設定します。

イネーブルレジスタのビットを 1 に設定すると，イネーブルレジスタのビット対応するサービスリクエストレジスタを読み取ることができます。サービスリクエスト・イネーブルレジスタは電源投入時にクリアされます。*RST と *CLS コマンドは標準イベントステータスレジスタ・イネーブルレジスタを変更しません。

文法

```
*SRE<wsp><value>
```

パラメータ

レジスタのビットの値（単精度整数):

7: 実行状態レジスタ	128
6: マスタサマリステータス（MSS) / サービスリクエスト（RQS)	64
5: 標準イベントステータスレジスタ（ESB)	32
4: MAV 出力キュー（MAV)	16
3: クエスチョナブルステータスレジスタ	8
2: エラー / イベントキュー	4
1: 未使用	2
0: 未使用	1

レスポンスデータ

無し

使用例

```
*SRE 64
```

*SRE?

機能

*SRE? クエリは，サービスリクエスト・イネーブルレジスタの値を問い合わせます。

レジスタの情報は*SRE を参照してください。

文法

```
*SRE?
```

パラメータ

無し

レスポンスデータ

レジスタのビットの値（単精度整数):

7: 実行状態レジスタ	128
6: マスタサマリステータス（MSS) / サービスリクエスト（RQS)	64
5: 標準イベントステータスレジスタ（ESB)	32
4: MAV 出力キュー（MAV)	16
3: クエスチョナブルステータスレジスタ	8
2: エラー / イベントキュー	4

1: 未使用　2
0: 未使用　1

使用例

```
*SRE?
> 21<END>
```

*STB?

機能

*STB? はステータスバイトレジスタの値を問い合わせます。STB レジスタビットのどれか（ビット 6 を除く)がセットされると，マスタサマリステータス(MSS) ビットが 1 になります。MSS を含むステータスバイトレジスタは，*STB?クエリによって変わりません。

文法

```
*STB?
```

パラメータ

無し

レスポンスデータ

レジスタのビットの値（単精度整数):

7: 実行状態レジスタ　128
6: マスタサマリステータス（MSS) / サービスリクエスト（RQS)　64
5: 標準イベントステータスレジスタ（ESB)　32
4: MAV 出力キュー（MAV)　16
3: クエスチョナブルステータスレジスタ　8
2: エラー / イベントキュー　4
1: 未使用　2
0: 未使用　1

使用例

```
*STB?
> 78<END>
```

*TST?

機能

このクエリは現時点でアクセスマスタで使用していません。

自己診断クエリ*TST?は選択した機器で自己診断テストを実行し，テスト結果を出力キューに出します。テスト実行中は他のコマンドを実行できま

せん。自己診断テストが終了した後，機器の設定は*TST?コマンド処理開始前の設定に戻ります。

文法

```
*TST?
```

パラメータ

無し

レスポンスデータ

個別の試験結果の合計値（32 ビット符号付き整数）

使用例

```
*TST?
> 0<END>
```

*WAI

機能

*WAI コマンドの前に送信したメッセージの処理が完了するまで，次のメッセージの実行を待たせます。

文法

```
*WAI
```

パラメータ

無し

レスポンスデータ

無し

使用例

```
*WAI
```

3.2　System コマンド

SYSTem:ERRor?

機能

エラーキューの内容を問い合わせます。読み取ったエラーはエラーキューから削除されます。

文法

```
SYSTem:ERRor?
```

パラメータ

無し

レスポンスデータ

エラーコマンドを送信順に並び替えられた最後に発生したエラーの番号，エラーの内容

使用例

```
SYST:ERR?
> -100,"std_command,Command Parse Error"<END>
```

SYSTem:VERSion?

機能

対応している SCPI バージョンを問い合わせます。

文法

```
SYSTem:VERSion?
```

パラメータ

無し

レスポンスデータ

バージョンの年と数字の文字列

使用例

```
SYST:VERS?
> 1990.0<END>
```

SYSTem:LIGHt?

機能

バックライトが点灯しているか問い合わせます。

文法

```
SYSTem:LIGHt?
```

パラメータ

無し

レスポンスデータ

0: バックライト OFF
1: バックライト ON

使用例

```
SYST:LIGH?
> 0<END>
```

SYSTem:LIGHt

機能

バックライトの ON または OFF をします。

文法

```
SYSTem:LIGHt<wsp><value>
```

パラメータ

<value>
ブーリアン形式
範囲: 0 | 1
0: バックライトを OFF にします。
1: バックライト ON にします。

レスポンスデータ

無し

使用例

```
SYST:LIGH 1<END>
```

3.3　Status サブシステムコマンド

STATus:OPERation[:EVENt]?

機能

オペレーションイベントレジスタを問い合わせます。

文法

```
STATus:OPERation[:EVENt]?
```

パラメータ

無し

レスポンスデータ

オペレーションイベントレジスタのビットの値（単精度整数:0 .. +32767）

使用例

```
STAT:OPER?
> 0<END>
```

STATus:OPERation:CONDition?

機能

オペレーションコンディションレジスタを問い合わせます。

文法

```
STATus:OPERation:CONDition?
```

パラメータ

無し

レスポンスデータ

オペレーションコンディションレジスタのビットの値（単精度整数:0 .. +32767）

使用例

```
STAT:OPER:COND?
> 16<END>
```

STATus:OPERation:BIT<n>:CONDition?

機能

本コマンドはオペレーションレジスタのユーザ定義ビットを操作します。

<n> の値は 8 から 12 まででオペレーションステータスレジスタのビット 8 からビット 12 に対応します。

文法

```
STATus:OPERation:BIT<n>:CONDition?
```

パラメータ

無し

レスポンスデータ

オペレーションコンディションレジスタのビットの値（単精度整数:0 .. 1）

使用例

```
STAT:OPER:BIT8:COND?
> 1<END>
```

STATus:OPERation:BIT<n>:ENABle

機能

オペレーションイネーブルマスクの指定したイベントレジスタビットを設定します。
<n> の値は 8 から 12 まで, ビット 8 からビット 12 を指定します。

文法

```
STATus:OPERation:BIT<n>:ENABle<wsp><value>
```

パラメータ

オペレーションイネーブルマスクのビットの値（単精度整数:0 .. 1）

レスポンスデータ

無し

使用例

```
STAT:OPER:BIT11:ENAB 1
```

STATus:OPERation:BIT<n>:ENABle?

機能

オペレーションイネーブルマスクの指定したイベントレジスタビットを問い合わせます。
<n> の値は 8 から 12 まででオペレーションステータスレジスタのビット 8 からビット 12 に対応します。

文法

```
STATus:OPERation:BIT<n>:ENABle?
```

パラメータ

無し

レスポンスデータ

オペレーションイネーブルマスクのビットの値（単精度整数:0 .. 1）

使用例

```
STAT:OPER:BIT9:ENAB?
> 0<END>
```

STATus:OPERation:BIT<n>[:EVENt]?

機能

オペレーションイベントレジスタの指定したビットを問い合わせます。
<n> の値は 8 から 12 までででオペレーションステータスレジスタのビット 8 からビット 12 に対応します。

文法

```
STATus:OPERation:BIT<n>[:EVENt]?
```

パラメータ

無し

レスポンスデータ

オペレーションイベントレジスタのビットの値（単精度整数:0 .. 1）

使用例

```
STAT:OPER:BIT10:EVEN?
> 0<END>
```

STATus:OPERation:ENABle

機能

イベントレジスタのオペレーションイネーブルマスクレジスタを設定します。

文法

```
STATus:OPERation:ENABle<wsp><value>
```

パラメータ
オペレーションイネーブルマスクのビットの値（単精度整数：0 .. +32767）

レスポンスデータ
無し

使用例
```
STAT:OPER:ENAB 128
```

STATus:OPERation:ENABle?

機能
イベントレジスタのオペレーションイネーブルマスクレジスタを問い合わせます。

文法
```
STATus:OPERation:ENABle?
```

パラメータ
無し

レスポンスデータ
オペレーションイネーブルマスクのビットの値（単精度整数：0 .. +32767）

使用例
```
STAT:OPER:ENAB?
> 128<END>
```

STATus:OPERation:INSTrument:CONDition?

機能
オペレーションコンディションレジスタを問い合わせます。

文法
```
STATus:OPERation:INSTrument:CONDition?
```

パラメータ
無し

レスポンスデータ
オペレーションコンディションレジスタのビットの値（単精度整数：0 .. +32767）

使用例

```
STAT:OPER:INST:COND?
> 16<END>
```

STATus:OPERation:INSTrument:ENABle

機能

イベントレジスタのオペレーションイネーブルマスクを設定します。

文法

```
STATus:OPERation:INSTrument:ENABle<wsp><value>
```

パラメータ

オペレーションイネーブルマスクのビットの値（単精度整数：0 .. +32767）

レスポンスデータ

無し

使用例

```
STAT:OPER:INST:ENAB 128
```

STATus:OPERation:INSTrument:ENABle?

機能

イベントレジスタのオペレーションイネーブルマスクを問い合わせます。

文法

```
STATus:OPERation:INSTrument:ENABle?
```

パラメータ

無し

レスポンスデータ

オペレーションイネーブルマスクのビットの値（単精度整数：0 .. +32767）

使用例

```
STAT:OPER:INST:ENAB?
> 128<END>
```

STATus:OPERation:INSTrument[:EVENt]?

機能

オペレーションイベントレジスタを問い合わせます。

文法

```
STATus:OPERation:INSTrument[:EVENt]?
```

パラメータ

無し

レスポンスデータ

オペレーションイベントレジスタのビットの値（単精度整数：0 .. +32767）

使用例

```
STAT:OPER:INST?
> 0<END>
```

STATus:OPERation:INSTrument:ISUMmary<n>:CONDition?

機能

指定した機器のオペレーションコンディションレジスタを問い合わせます。<n>は 1 から 14 までの値で，INSTrument サブシステムによって SCPI で制御される機器に割り当てられた論理機器の識別番号を表します。

文法

```
STATus:OPERation:INSTrument:ISUMmary<n>:CONDition?
```

パラメータ

無し

レスポンスデータ

オペレーションコンディションレジスタのビットの値（単精度整数：0 .. +32767）

使用例

```
STAT:OPER:INST:ISUM4:COND?
> 16<END>
```

STATus:OPERation:INSTrument:ISUMmary<n>:ENABle

機能

指定した機器のイベントレジスタに対するオペレーションイネーブルマスク

を設定します。
<n>は 1 から 14 までの値で, INSTrument サブシステムによって SCPI で制御される機器に割り当てられた論理機器の識別番号を表します。

文法

```
STATus:OPERation:INSTrument:ISUMmary<n>:ENABle<wsp><value>
```

パラメータ

オペレーションイネーブルマスクのビットの値（単精度整数:0 .. +32767）

レスポンスデータ

無し

使用例

```
STAT:OPER:INST:ISUM1:ENAB 128
```

STATus:OPERation:INSTrument:ISUMmary<n>:ENABle?

機能

指定した機器のイベントレジスタに対するオペレーションイネーブルマスクを問い合わせます。
<n>は 1 から 14 までの値で, INSTrument サブシステムによって SCPI で制御される機器に割り当てられた論理機器の識別番号を表します。

文法

```
STATus:OPERation:INSTrument:ISUMmary<n>ENABle?
```

パラメータ

無し

レスポンスデータ

オペレーションイネーブルマスクのビットの値（単精度整数:0 .. +32767）

使用例

```
STAT:OPER:INST:ISUM4:ENAB?
> 128<END>
```

STATus:OPERation:INSTrument:ISUMmary<n>[:EVENt]?

機能

指定した機器のオペレーションイベントレジスタを問い合わせます。

<n>は 1 から 14 までの値で, INSTrument サブシステムによって SCPI で制御される機器に割り当てられた論理機器の識別番号を表します。

文法

```
STATus:OPERation:INSTrument:ISUMmary<n>[:EVENt]?
```

パラメータ

無し

レスポンスデータ

オペレーションイベントレジスタのビットの値（単精度整数:0 .. +32767）

使用例

```
STAT:OPER:INST:ISUM3?
> 0<END>
```

STATus:QUEStionable[:EVENt]?

機能

クエスチョナブルイベントレジスタを問い合わせます。

文法

```
STATus:QUEStionable[:EVENt]?
```

パラメータ

無し

レスポンスデータ

クエスチョナブルイベントレジスタのビットの値（単精度整数:0 .. +32767）

使用例

```
STAT:QUES?
> 0<END>
```

STATus:QUEStionable:CONDition?

機能

クエスチョナブルコンディションレジスタを問い合わせます。

文法

```
STATus:QUEStionable:CONDition?
```

パラメータ
無し

レスポンスデータ
クエスチョナブルコンディションレジスタのビットの値（単精度整数:0 .. +32767）

使用例
```
STAT:QUES:COND?
> 8<END>
```

STATus:QUEStionable:BIT<n>:CONDition?

機能
クエスチョナブルコンディションレジスタの指定したビットを問い合わせます。
<n> の値は 9 から 12 までででオペレーションステータスレジスタのビット 9 からビット 12 に対応します。

文法
```
STATus:QUEStionable:BIT<n>:CONDition?
```

パラメータ
無し

レスポンスデータ
クエスチョナブルコンディションレジスタのビットの値（単精度整数:0 .. 1）

使用例
```
STAT:QUES:BIT9:COND?
> 0<END>
```

STATus:QUEStionable:BIT<n>:ENABle

機能
イベントレジスタに対する，クエスチョナブルイネーブルマスクの指定したビットを設定します。
<n> の値は 9 から 12 までででクエスチョナブルステータスレジスタのビット 9 からビット 12 に対応します。

文法
```
STATus:QUEStionable:BIT<n>:ENABle<wsp><value>
```

パラメータ
クエスチョナブルイネーブルマスクのビットの値（単精度整数:0 .. 1）

レスポンスデータ
無し

使用例
```
STAT:QUES:BIT11:ENAB 1
```

STATus:QUEStionable:BIT<n>:ENABle?

機能
イベントレジスタに対する，クエスチョナブルイネーブルマスクの指定したビットを問い合わせます。
<n> の値は 9 から 12 まででクエスチョナブルステータスレジスタのビット 9 からビット 12 に対応します。

文法
```
STATus:QUEStionable:BIT<n>:ENABle?
```

パラメータ
無し

レスポンスデータ
クエスチョナブルイネーブルマスクのビットの値（単精度整数:0 .. 1）

使用例
```
STAT:QUES:BIT10:ENAB?
> 1<END>
```

STATus:QUEStionable:BIT<n>[:EVENt]?

機能
クエスチョナブルイベントレジスタの指定したビットを問い合わせます。
<n> の値は 9 から 12 まででクエスチョナブルステータスレジスタのビット 9 からビット 12 に対応します。

文法
```
STATus:QUEStionable:BIT<n>[:EVENt]?
```

パラメータ
無し

レスポンスデータ
クエスチョナブルイベントレジスタのビットの値（単精度整数：0 .. 1）

使用例
```
STAT:QUES:BIT9:EVEN?
> 0<END>
```

STATus:QUEStionable:ENABle

機能
イベントレジスタ用のクエスチョナブルイネーブルマスクを設定します。

文法
```
STATus:QUEStionable:ENABle<wsp><value>
```

パラメータ
クエスチョナブルイネーブルマスクのビットの値（単精度整数：0 .. +32767）

レスポンスデータ
無し

使用例
```
STAT:QUES:ENAB 128
```

STATus:QUEStionable:ENABle?

機能
イベントレジスタ用のクエスチョナブルイベントレジスタイネーブルマスクを問い合わせます。

文法
```
STATus:QUEStionable:ENABle?
```

パラメータ
無し

レスポンスデータ
クエスチョナブルイネーブルマスクのビットの値（単精度整数：0 ..

+32767)

使用例

```
STAT:QUES:ENAB?
> 128<END>
```

STATus:QUEStionable:INSTrument:CONDition?

機能

クエスチョナブル機器コンディションレジスタを問い合わせます。

文法

```
STATus:QUEStionable:INSTrument:CONDition?
```

パラメータ

無し

レスポンスデータ

クエスチョナブルコンディションレジスタのビットの値（単精度整数:0 .. +32767)

使用例

```
STAT:QUES:INST:COND?
> 8<END>
```

STATus:QUEStionable:INSTrument:ENABle

機能

イベントレジスタ用のクエスチョナブル機器イネーブルマスクを設定します。

文法

```
STATus:QUEStionable:INSTrument:ENABle<wsp><value>
```

パラメータ

クエスチョナブルイネーブルマスクのビットの値（単精度整数:0 .. +32767)

レスポンスデータ

無し

使用例

```
STAT:QUES:INST:ENAB 128
```

STATus:QUEStionable:INSTrument:ENABle?

機能

イベントレジスタ用のクエスチョナブル機器イネーブルマスクを問い合わせます。

文法

```
STATus:QUEStionable:INSTrument:ENABle?
```

パラメータ

無し

レスポンスデータ

クエスチョナブルイネーブルマスクのビットの値（単精度整数：0 .. +32767）

使用例

```
STAT:QUES:INST:ENAB?
> 128<END>
```

STATus:QUEStionable:INSTrument[:EVENt]?

機能

クエスチョナブル機器イベントレジスタを問い合わせます。

文法

```
STATus:QUEStionable:INSTrument[:EVENt]?
```

パラメータ

無し

レスポンスデータ

クエスチョナブルイベントレジスタのビットの値（単精度整数：0 .. +32767）

使用例

```
STAT:QUES:INST:EVEN?
> 0<END>
```

STATus:QUEStionable:INSTrument:ISUMmary<n>:CONDition?

機能

指定した機器のクエスチョナブルステータスレジスタを問い合わせます。
<n>は 1 から 14 までの値で，INSTrument サブシステムによって SCPI で制御される機器に割り当てられた論理機器の識別番号を表します。

文法

```
STATus:QUEStionable:INSTrument:ISUMmary<n>:CONDition?
```

パラメータ

無し

レスポンスデータ

クエスチョナブルコンディションレジスタのビットの値（単精度整数：0 .. +32767）

使用例

```
STAT:QUES:INST:ISUM2:COND?
> 0<END>
```

STATus:QUEStionable:INSTrument:ISUMmary<n>:ENABle

機能

指定した機器のイベントレジスタに対するクエスチョナブルイネーブルマスクを設定します。
<n>は 1 から 14 までの値で，INSTrument サブシステムによって SCPI で制御される機器に割り当てられた論理機器の識別番号を表します。

文法

```
STATus:QUEStionable:INSTrument:ISUMmary<n>:ENABle<wsp><value>
```

パラメータ

クエスチョナブルイネーブルマスクのビットの値（単精度整数：0 .. +32767）

レスポンスデータ

無し

使用例

```
STAT:QUES:INST:ISUM3:ENAB 128
```

STATus:QUEStionable:INSTrument:ISUMmary<n>:ENABle?

機能

指定した機器のイベントレジスタに対するクエスチョナブルイネーブルマスクを問い合わせます。

<n>は 1 から 14 までの値で，INSTrument サブシステムによって SCPI で制御される機器に割り当てられた論理機器の識別番号を表します。

文法

```
STATus:QUEStionable:INSTrument:ISUMmary<n>:ENABle?
```

パラメータ

無し

レスポンスデータ

クエスチョナブルイネーブルマスクのビットの値（単精度整数：0 .. +32767）

使用例

```
STAT:QUES:INST:ISUM3:ENAB?
> 136<END>
```

STATus:QUEStionable:INSTrument:ISUMmary<n>[:EVENt]?

機能

指定した機器のクエスチョナブルイベントレジスタを問い合わせます。

<n>は 1 から 14 までの値で，INSTrument サブシステムによって SCPI で制御される機器に割り当てられた論理機器の識別番号を表します。

文法

```
STATus:QUEStionable:INSTrument:ISUMmary<n>[:EVENt]
?
```

パラメータ

無し

レスポンスデータ

クエスチョナブルイベントレジスタのビットの値（単精度整数：0 .. +32767）

使用例

```
STAT:QUES:INST:ISUM4:EVEN?
> 0<END>
```

STATus:PRESet

機能

オペレーションイベントマスクとクエスチョナブルイベントマスクの両方を 0 にします。

文法

```
STATus:PRESet
```

パラメータ

無し

レスポンスデータ

無し

使用例

```
STAT:PRES
```

3.4　Instrument サブシステムコマンド

INSTrument:CATalog?

機能

SCPI で制御できる装置として検出されたアクセスマスタ内の機器リストを問い合わせます。

文法

```
INSTrument:CATalog?
```

パラメータ

無し

レスポンスデータ

コンマで区切られた，SCPI で制御できるすべての機器の識別文字リスト

TOP_MENU	トップメニュー
OTDR_STD	光パルス試験（通常試験）

使用例

```
INST:CAT?
> TOP_MENU,OTDR_STD<END>
```

INSTrument:CATalog:FULL?

機能

SCPI で制御できる装置として検出されたアクセスマスタ内の機器リストを問い合わせます。

文法

```
INSTrument:CATalog:FULL?
```

パラメータ

無し

レスポンスデータ

論理機器の識別文字と数字が組みになったリスト

文字列は論理機器を識別する名称を含みます。その後の単精度整数値は，そのポートの論理機器番号です。すべてのレスポンスデータの要素はコンマで区切られています。

使用例

```
INSTrument:CATalog:FULL?
> TOP_MENU,1,OTDR_STD,2<END>
```

INSTrument:NSELect

機能

制御対象の機器として指定した論理機器の識別番号を設定します。

文法

```
INSTrument:NSELect<wsp><num_id>
```

パラメータ

制御対象となっている論理機器の識別番号（単精度整数）

レスポンスデータ

無し

使用例

```
INST:NSEL 2
```

INSTrument:NSELect?

機能

現在選択されている機器の識別番号を問い合わせます。

文法

```
INSTrument:NSELect?
```

パラメータ

無し

レスポンスデータ

制御対象となっている論理機器の識別番号（単精度整数）

使用例

```
INST:NSEL?
> 2<END>
```

INSTrument[:SELect]

機能

指定した論理機器を制御対象として選択します。

文法

```
INSTrument:SELect<wsp><string_id>
```

パラメータ

Instrument サブシステムで割り当てられた論理機器を選択するための識別文字列

レスポンスデータ

無し

使用例

```
INST:SEL OTDR_STD
```

INSTrument[:SELect]?

機能

選択されている論理機器の識別文字を問い合わせます。

文法

```
INSTrument:SELect?
```

パラメータ

無し

レスポンスデータ

現在選択されている論理機器の識別文字

使用例

```
INST:SEL?
> OTDR_STD<END>
```

INSTrument:STATe

機能

選択している論理機器の状態を On または Off にします。

文法

```
INSTrument:STATe<wsp><boolean>
```

パラメータ

ブーリアン形式

On: 1 または ON
Off: 0 または OFF

レスポンスデータ
無し

使用例
```
INST:STAT ON
```

INSTrument:STATe?

機能
選択されている論理機器の状態を問い合わせます。

文法
```
INSTrument:STATe?
```

パラメータ
無し

レスポンスデータ
論理機器の状態
ブーリアン形式
1: 機器が On
0: 機器が Off

使用例
```
INST:STAT?
> 1<END>
```

3.5　TOPMenu サブシステムコマンド

TOPMenu:UNIT:CATalog?

機能

アクセスマスタに搭載されているポートのリストを問い合わせます。

文法

```
TOPMenu:UNIT:CATalog?
```

パラメータ

無し

レスポンスデータ

コンマで区切られた，搭載されているすべてのポートの識別文字リスト

使用例

```
TOPM:UNIT:CAT?
> MM,SM<END>
```

TOPMenu:UNIT:CATalog:FULL?

機能

アクセスマスタに搭載されているポートのリストを問い合わせます。

文法

```
TOPMenu:UNIT:CATalog:FULL?
```

パラメータ

無し

レスポンスデータ

ポートの識別文字と数字が組みになったリスト
文字列は搭載されているポート名称を含みます。その後の単精度整数値は，そのポートの論理ポート番号です。すべてのレスポンスデータの要素はコンマで区切られています。

使用例

```
TOPM:UNIT:CAT:FULL?
> MM,1,SM,2<END>
```

TOPMenu:UNIT[:SELect]

機能

指定したポートを選択ポートに設定します。

文法

```
TOPMenu:UNIT:SELect<wsp><string_id>
```

パラメータ

TOPMenu サブシステムで割り当てられたポートを選択するための識別文字列

レスポンスデータ

無し

使用例

```
TOPM:UNIT:SEL SM
```

TOPMenu:UNIT[:SELect]?

機能

選択されているポートの識別文字を問い合わせます。

文法

```
TOPMenu:UNIT:SELect?
```

パラメータ

無し

レスポンスデータ

選択されているポートの識別文字

使用例

```
TOPM:UNIT:SEL?
> MM<END>
```

TOPMenu:UNIT:NSELect

機能

指定したポートを，光パルス試験を実行するポートに設定します。

文法

```
TOPMenu:NSELect<wsp><num_id>
```

パラメータ

TOPMenu サブシステムで割り当てられたポートの識別番号(単精度整数)

レスポンスデータ

無し

使用例

```
TOPM:UNIT:NSEL 2
```

TOPMenu:UNIT:NSELect?

機能

選択されているポートの識別番号を問い合わせます。

文法

```
TOPMenu:UNIT:NSELect?
```

パラメータ

無し

レスポンスデータ

選択されているポートの識別番号

使用例

```
TOPM:UNIT:NSEL?
> 1<END>
```

この章では，アクセスマスタの光パルス試験（通常試験）で使用するコマンドの詳細を説明します。コマンドサマリの節では，各コマンドの短い説明をします。詳細な説明はその後の節で記載しています。
文法の説明で<wsp>は，スペースを表します。

4.1　コマンドサマリ

この節では，アクセスマスタの光パルス試験（通常試験）のコマンドリストと各コマンドの簡単な説明をします。

この節のコマンドとクエリは，光パルス試験（通常試験）のときだけ受け付けられます。

4.1.1　ルートレベルコマンド

ABORt	試験を中止します。トレースデータは破棄されます。
STOP	試験を停止します。トレースは画面に表示されたままになります。
INITiate	手動の設定で試験を開始します。
INITiate:AUTo	自動で試験を開始します。
INITiate:RTIMe	手動の設定でリアルタイム測定を開始します。
INITiate?	試験実行中か問い合わせます。

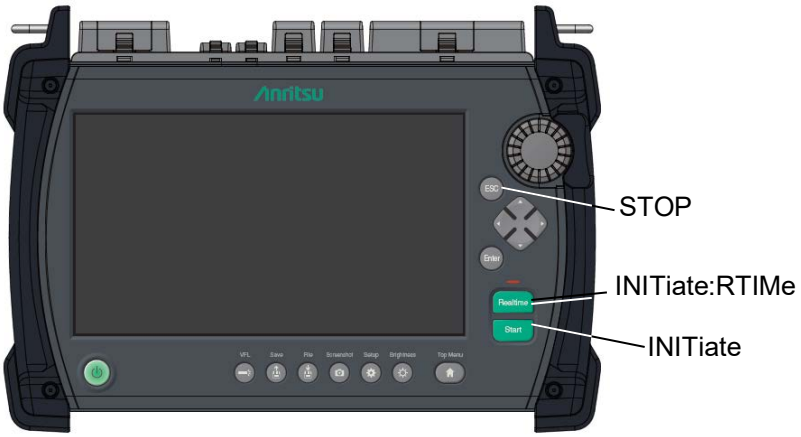


図 4.1.1-1　測定に関するコマンド

4.1.2 Source サブシステムコマンド

SOURce:WAVelength:AVAilable?	波長のリストを問い合わせます。
SOURce:WAVelength	波長を設定します。
SOURce:WAVelength?	設定されている波長を問い合わせます。
SOURce:RANge:AVAilable?	距離レンジのリストを問い合わせます。
SOURce:RANge	距離レンジを設定します。
SOURce:RANge?	設定されている距離レンジを問い合わせます。
SOURce:RESo:AVAilable?	分解能のリストを問い合わせます。
SOURce:RESo	分解能を設定します。
SOURce:RESo?	設定されている分解能を問い合わせます。
SOURce:PULSe:AVAilable?	パルス幅のリストを問い合わせます。
SOURce:PULSe	パルス幅を設定します。
SOURce:PULSe?	設定されているパルス幅を問い合わせます。
SOURce:PULSe:ENHanced:AVAilable?	設定されているパルス幅に対してデッドゾーンの高ダイナミックレンジが設定できるか問い合わせます。
SOURce:PULSe:ENHanced	デッドゾーンの高ダイナミックレンジを設定します。
SOURce:PULSe:ENHanced?	デッドゾーンが高ダイナミックレンジに設定されているか，問い合わせます。
SOURce:AVERages:TIMe	次回の測定で使用する平均化時間を設定します。
SOURce:AVERages:TIMe?	次回の測定で使用する平均化時間を問い合わせます。
SOURce:AVERages:COUNt	平均化回数を設定します。
SOURce:AVERages:COUNt?	平均化回数を問い合わせます。
SOURce:AVERages:EXPonent	平均化回数を 2 の指数形式で設定します。
SOURce:AVERages:EXPonent?	平均化回数を 2 の指数形式で問い合わせます。

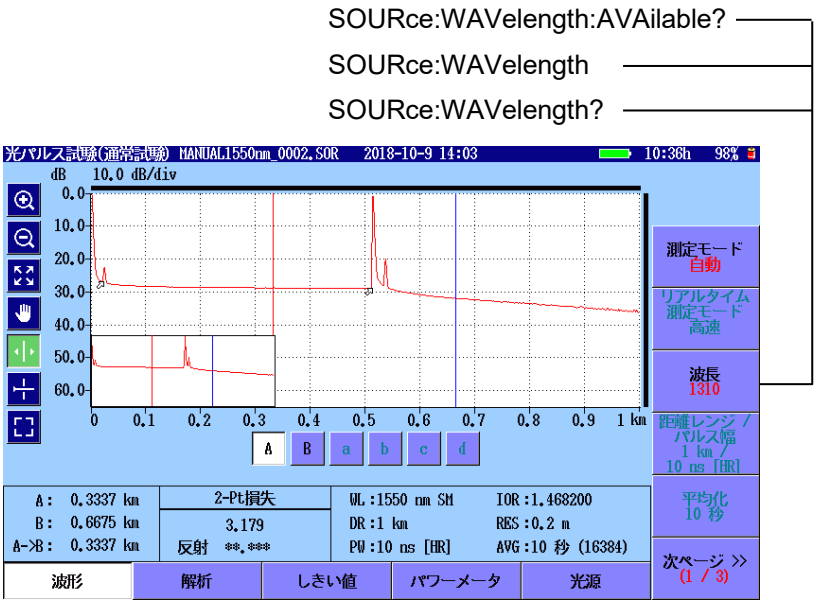


図 4.1.2-1　波長に関するコマンド

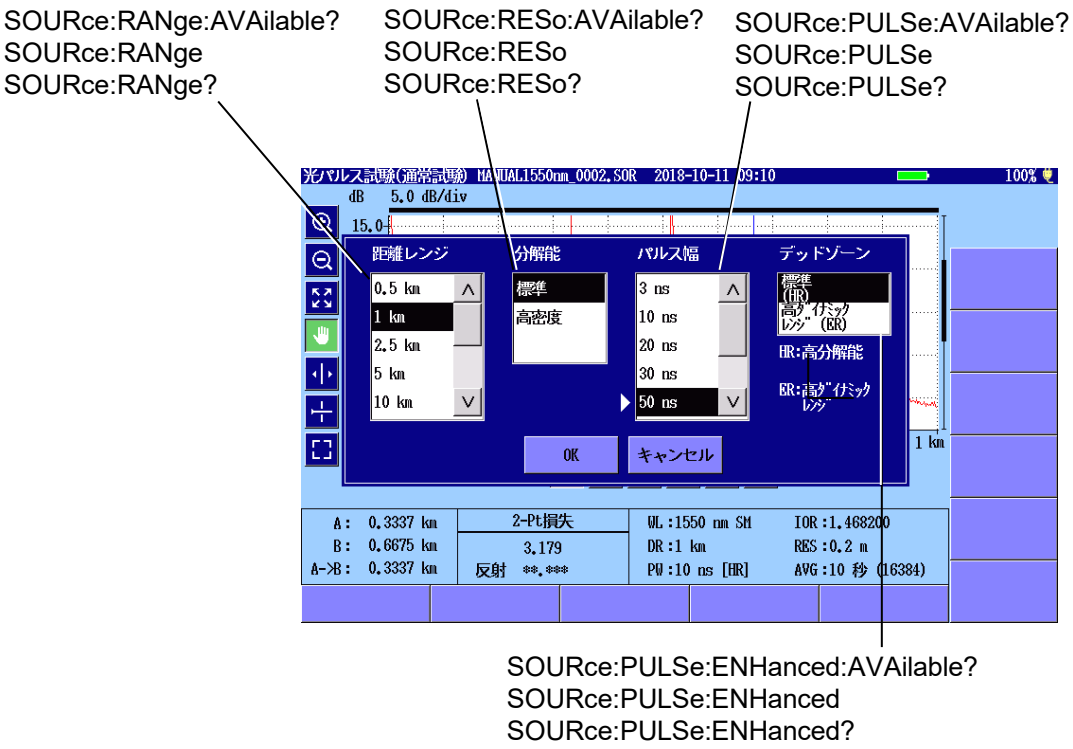


図 4.1.2-2　測定条件に関するコマンド

SOURce:AVERages:TIMe
SOURce:AVERages:TIMe?
SOURce:AVERages:COUNt
SOURce:AVERages:COUNt?
SOURce:AVERages:EXPonent
SOURce:AVERages:EXPonent?

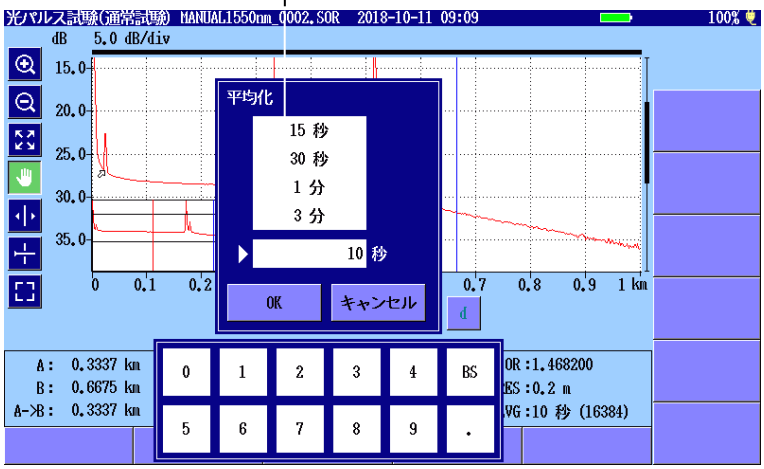


図 4.1.2-3　平均化時間に関するコマンド

4.1.3　Sense サブシステムコマンド

SENSe:AVERages?	測定を開始してから平均化した回数を問い合わせます。
SENSe:AVERages:TIMe?	測定を開始してから平均化した時間を問い合わせます。
SENSe:TRACe:READY?	トレースデータの準備ができているか問い合わせます。
SENSe:CONCheck	接続チェックを On または Off に設定します。
SENSe:CONCheck?	接続チェックが On かどうか問い合わせます。
SENSe:LIVCheck	通信光チェックを On または Off に設定します。
SENSe:LIVCheck?	通信光チェックが On かどうか問い合わせます。
SENSe:FIBCheck	ファイバ長チェックを On または Off に設定します。
SENSe:FIBCheck?	ファイバ長チェックが On かどうか問い合わせます。
SENSe:FIBer:IOR	ファイバの IOR を設定します。
SENSe:FIBer:IOR?	ファイバの IOR を問い合わせます。
SENSe:FIBer:BSC	ファイバの BSC を設定します。
SENSe:FIBer:BSC?	ファイバの BSC を問い合わせます。
SENSe:HOFFset	水平オフセット値を設定します。
SENSe:HOFFset?	水平オフセット値を問い合わせます。
SENSe:VOFFset	垂直オフセット値を設定します。
SENSe:VOFFset?	垂直オフセット値を問い合わせます。
SENSe:LSALeft	左側 LSA マーカ位置の値を設定します。
SENSe:LSALeft?	左側 LSA マーカ位置の値を問い合わせます。
SENSe:LSARight	右側 LSA マーカ位置の値を設定します。
SENSe:LSARight?	右側 LSA マーカ位置の値を問い合わせます。
SENSe:ACURsor	カーソル A の位置を設定します。
SENSe:ACURsor?	カーソル A の位置を問い合わせます。
SENSe:BCURsor	カーソル B の位置を設定します。
SENSe:BCURsor?	カーソル B の位置を問い合わせます。
SENSe:LOSS:MODE	損失計算方法を設定します。
SENSe:LOSS:MODE?	損失計算方法を問い合わせます。
SENSe:ORL:MODE	全反射減衰量計算を設定します。
SENSe:ORL:MODE?	全反射減衰量計算を問い合わせます。
SENSe:ANALyze:PARameters	自動検出のパラメータを設定します。
SENSe:ANALyze:PARameters?	自動検出のパラメータを問い合わせます。
SENSe:ANALyze:PARameters:ENDRefl	自動検出のパラメータを設定します。
SENSe:ANALyze:PARameters:ENDRefl?	自動検出のパラメータを問い合わせます。
SENSe:ANALyze:AUTO	自動解析を On または Off に設定します。
SENSe:ANALyze:AUTO?	自動解析が On かどうか問い合わせます。

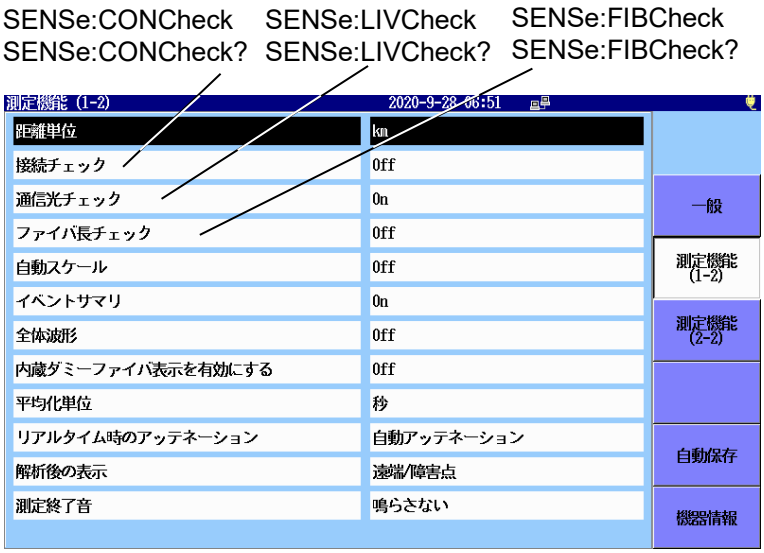


図 4.1.3-1　測定機能 (1-2) のコマンド

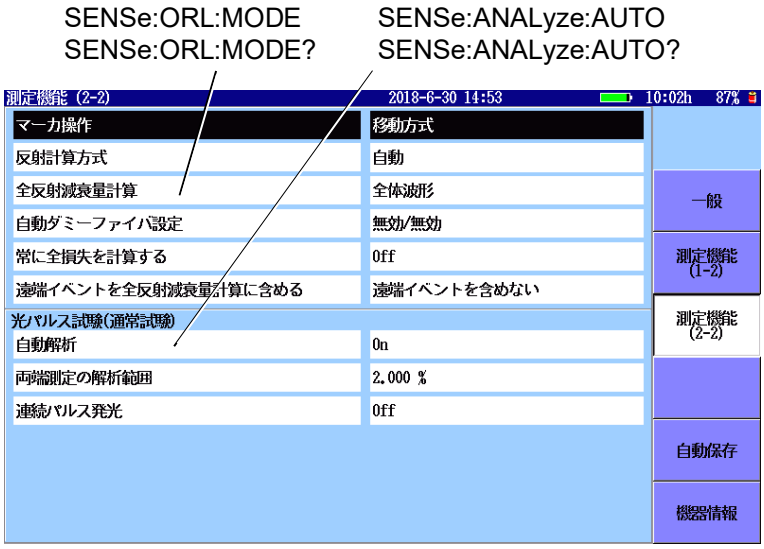


図 4.1.3-2　測定機能 (2-2) のコマンド

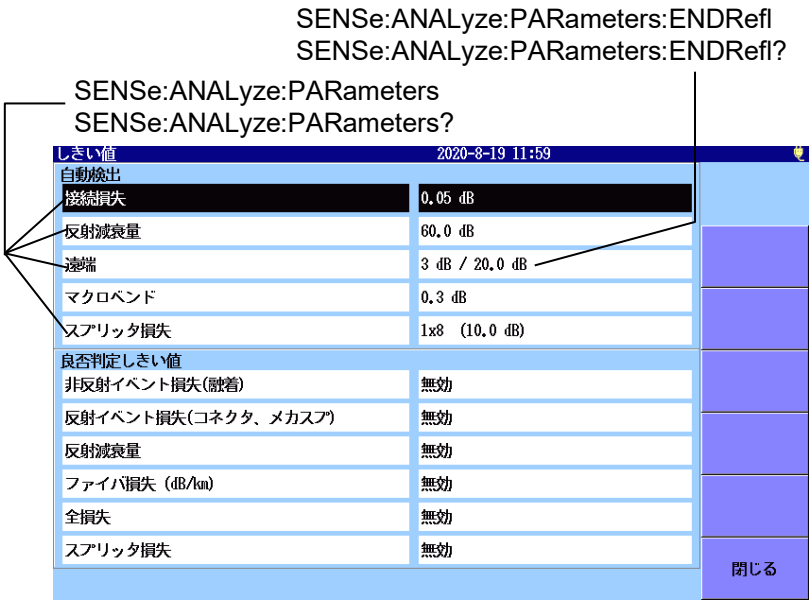


図 4.1.3-3　しきい値のコマンド

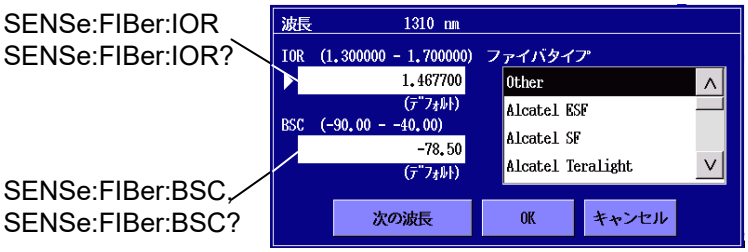


図 4.1.3-4　IOR と BSC のコマンド

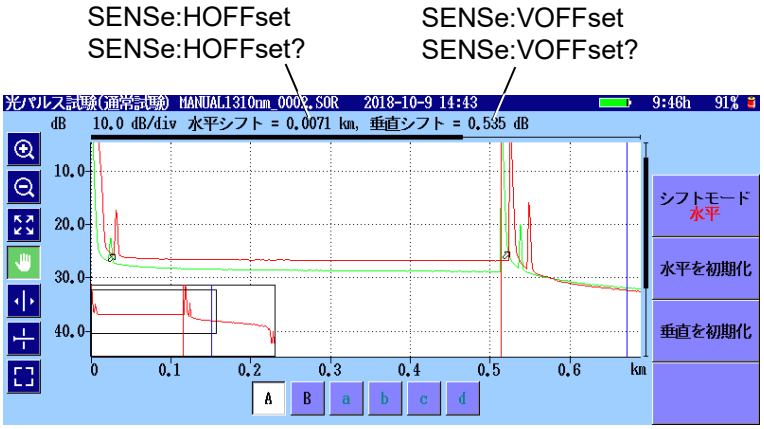


図 4.1.3-5　オフセットのコマンド

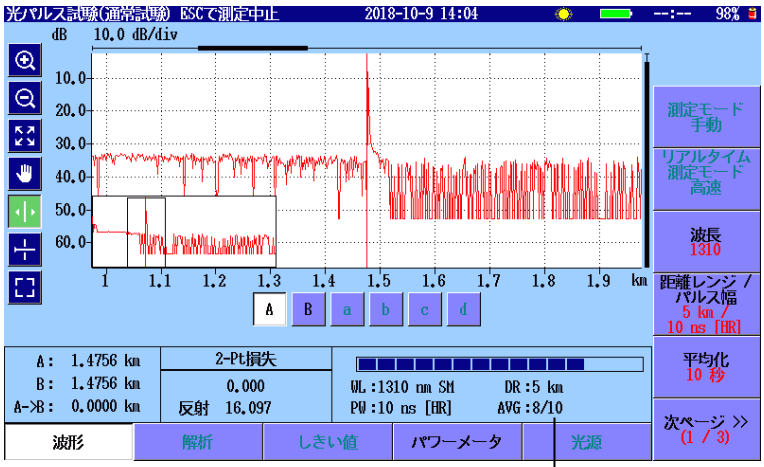


図 4.1.3-6 平均化処理のコマンド

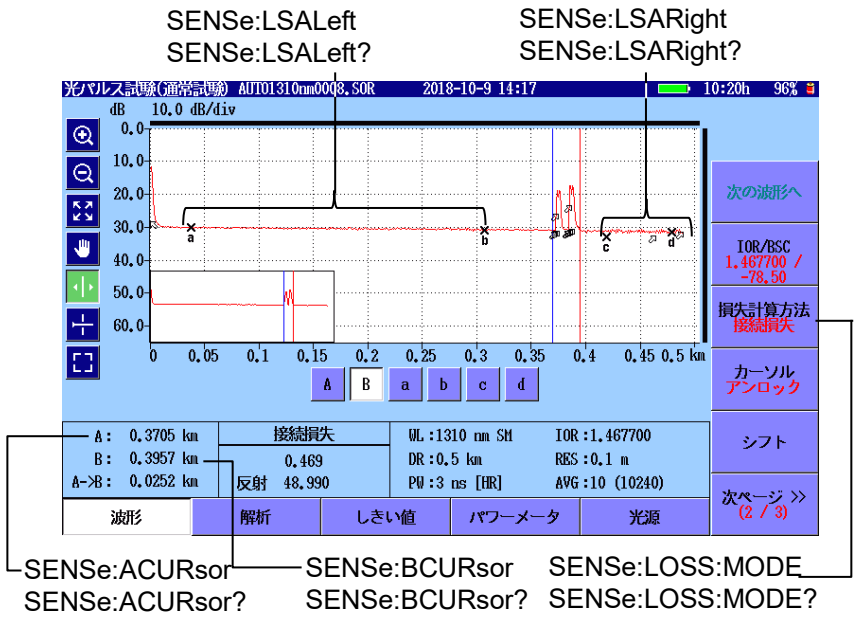


図 4.1.3-7 カーソルのコマンド

4.1.4　Trace サブシステムコマンド

TRACe:PARameters?	トレースのパラメータをテキスト形式で問い合わせます。
TRACe:ANALyze	トレースの解析を実行します。
TRACe:ANALyze?	トレースの解析が終了したかを問い合わせます。
TRACe:ANALyze:ORL	トレースの全反射減衰量を計算します。
TRACe:MDLOss?	現在の損失計算方法による損失の値を問い合わせます。
TRACe:EELOss?	全損失を問い合わせます。
TRACe:LOAD:SOR?	トレースデータを SOR ファイル形式で問い合わせます。
TRACe:LOAD:TEXT?	トレースデータを ASCII テキスト形式で問い合わせます。
TRACe:LOAD:DATA?	トレースデータポイントをバイナリ形式で問い合わせます。
TRACe:HEADer	ファイルヘッダを設定します。
TRACe:HEADer?	ファイルヘッダを問い合わせます。
TRACe:STORe:SOR	アクセスマスタの内蔵メモリに SOR ファイルを保存します。

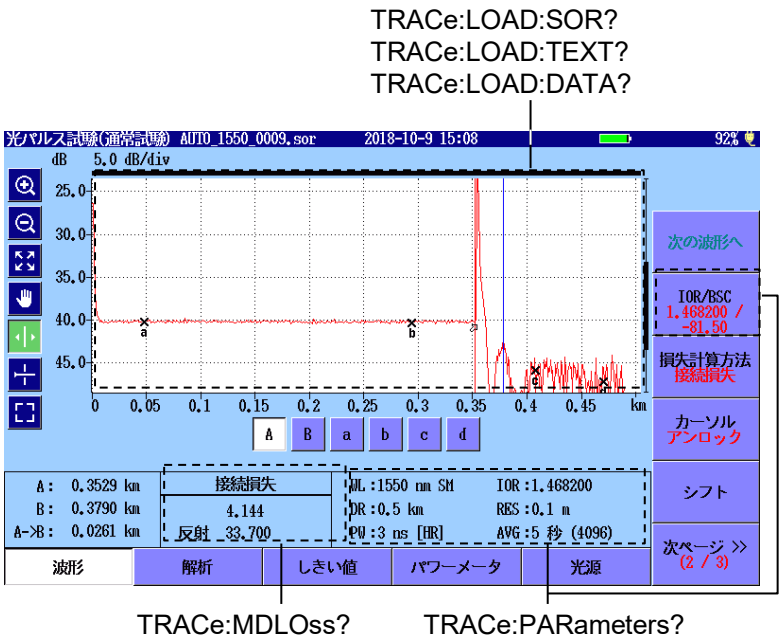


図 4.1.4-1　トレースデータのコマンド

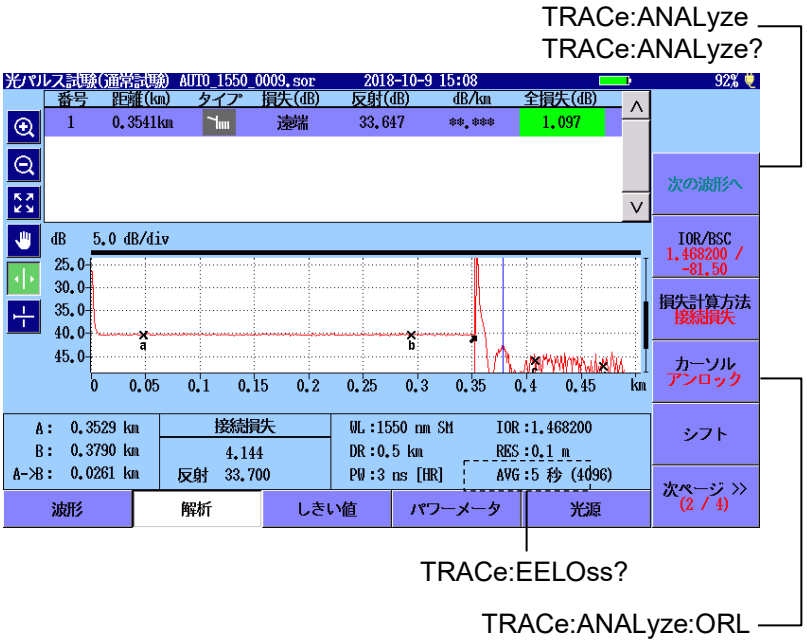


図 4.1.4-2　解析のコマンド

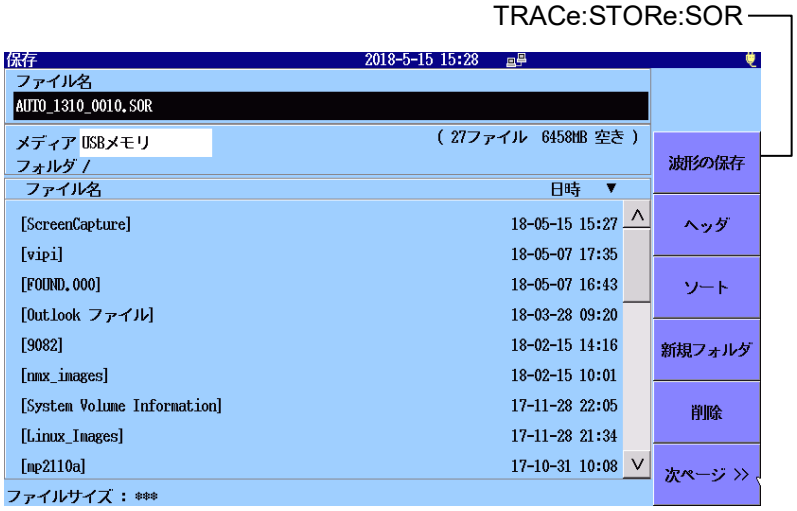


図 4.1.4-3　波形保存のコマンド

TRACe:HEADer
TRACe:HEADer?

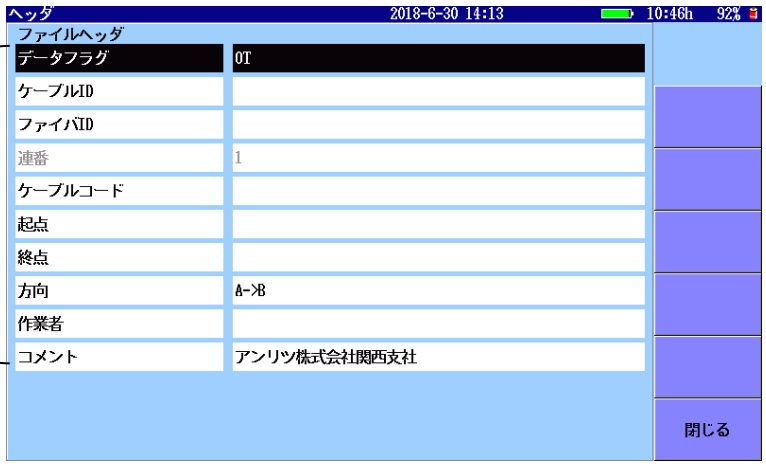


図 4.1.4-4　ヘッダのコマンド

4.1.5 Display サブシステムコマンド

DISPlay:MODE	距離表示の基準カーソルを設定します。
DISPlay:MODE?	距離表示の基準カーソルを問い合わせます。
DISPlay:ZOOM:FULL	全体波形が表示されるようにズームを設定します。
DISPlay:ZOOM:HORIzontal	水平スケールのズームレベルを設定します。
DISPlay:ZOOM:HORIzontal?	水平スケールのズームレベルを問い合わせます。
DISPlay:ZOOM:VERTical	垂直スケールのズームレベルを設定します。
DISPlay:ZOOM:VERTical?	垂直スケールのズームレベルを問い合わせます。
DISPlay:SCALe:HORIzontal	水平スケールを設定します。
DISPlay:SCALe:HORIzontal?	水平スケールを問い合わせます。
DISPlay:SCALe:VERTical	垂直スケールを設定します。
DISPlay:SCALe:VERTical?	垂直スケールを問い合わせます。

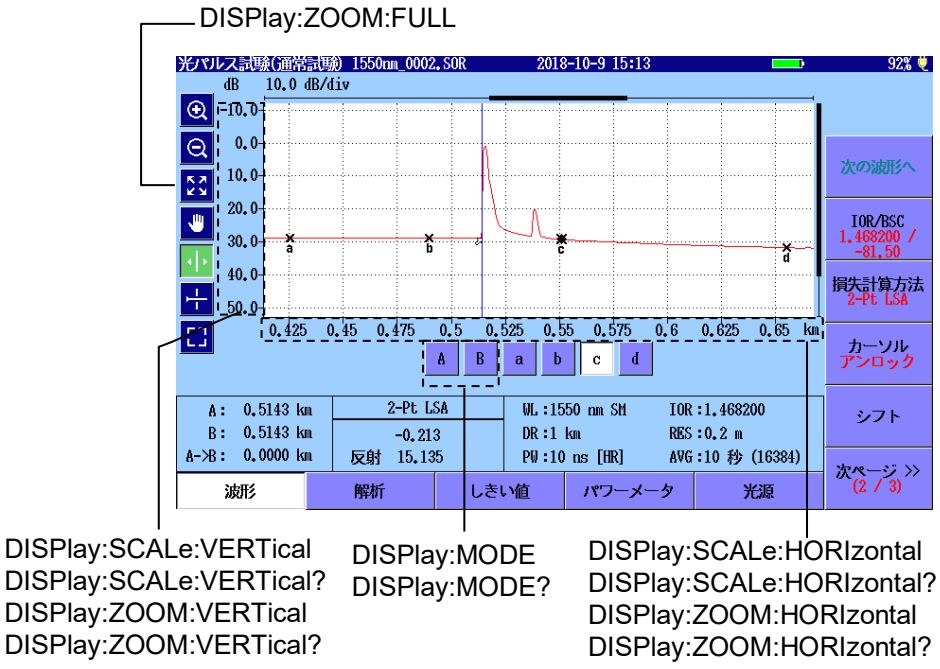


図 4.1.5-1 表示に関するコマンド

4.2　ルートレベルコマンド

ABORt

機能

試験を中止します。トレースデータは破棄されます。

文法

```
ABORt
```

パラメータ

無し

レスポンスデータ

無し

エラー

(–200, "std_execGen, Test is Inactive")
(–200, "std_execGen, Instrument is Busy")

使用例

```
Abor
```

STOP

機能

試験を停止します。トレースデータは保持されます。

文法

```
STOP
```

パラメータ

無し

レスポンスデータ

無し

エラー

(–200, "std_execGen, Test is Inactive")
(–200, "std_execGen, Instrument is Busy")

使用例

```
Stop
```

INITiate

機能

設定されている測定条件で光パルス試験を開始します。測定モードは[手動] になります。
本コマンドは，他のコマンドと同時に実行できるオーバーラップコマンドです。
全波長を設定しているときに本コマンドを実行すると，一番短い波長で測定を開始しますが，その他の波長では測定しません。

文法

```
INITiate
```

パラメータ

無し

レスポンスデータ

無し

エラー

(–200, "std_execGen, Test is Active")
(–200, "std_execGen, Instrument is Busy")
(–200, "std_execGen, Start Test Failed")
(–200, "std_execGen, Connection Check Failed")
(–200, "std_execGen, Live Fiber Check Failed")

使用例

```
init
```

INITiate:AUTo

機能

光パルス試験の自動試験を開始します。測定パラメータは，測定開始前に試しの測定をすることにより自動で設定されます。
本コマンドは他のコマンドと同時に実行できるオーバーラップコマンドです。
全波長を設定しているときに本コマンドを実行すると，一番短い波長で測定を開始しますが，その他の波長では測定しません。

文法

```
INITiate:AUTo
```

パラメータ

無し

レスポンスデータ

無し

エラー

(–200, "std_execGen, Test is Active")
(–200, "std_execGen, Instrument is Busy")
(–200, "std_execGen, Start Test Failed")
(–200, "std_execGen, Connection Check Failed")
(–200, "std_execGen, Live Fiber Check Failed")

使用例

```
init:auto
```

INITiate:RTIMe

機能

設定されている測定条件で光パルス試験のリアルタイム測定を開始します。
本コマンドは他のコマンドと同時に実行できるオーバーラップコマンドです。
全波長を設定しているときに本コマンドを実行すると，一番短い波長で測定を開始しますが，その他の波長では測定しません。

文法

```
INITiate:RTIMe
```

パラメータ

無し

レスポンスデータ

無し

エラー

(–200, "std_execGen, Test is Active")
(–200, "std_execGen, Instrument is Busy")
(–200, "std_execGen, Start Test Failed")

(–200, "std_execGen, Connection Check Failed")
(–200, "std_execGen, Live Fiber Check Failed")

使用例

```
init:rtim
```

INITiate?

機能

試験が開始されたか問い合わせます。

文法

```
INITiate?
```

パラメータ

無し

レスポンスデータ

0 | 1
0: 試験は実施されていません。
1: 試験中

エラー

無し

使用例

```
init?
> 0 <End>
```

4.3　SOURce サブシステムコマンド

SOURce サブシステムは，光パルス試験の光源パラメータを設定または問い合わせをします。

SOURce:WAVelength:AVAilable?

機能

使用できる波長のリストを問い合わせます。

文法

```
SOURce:WAVelength:AVAilable?
```

パラメータ

無し

レスポンスデータ

使用できる波長のリスト（nm 単位）

エラー

無し

使用例

```
sour:wav:ava?
> 1310, 1550<END>
```

SOURce:WAVelength

機能

波長（nm 単位）を設定します。設定する波長が，使用できる波長のリストに存在する場合に設定されます。

文法

```
SOURce:WAVelength<wsp><value>
```

パラメータ

<value>
整数値
範囲: SOURce:WAVelength:AVAilable?で取得した波長（整数値）
全波長は設定できません。

レスポンスデータ

無し

エラー

(–200, "std_execGen, Test is Active")
(–224, "std_illegalParmValue, Invalid Parameter Value")

使用例

```
sour:wave 1310
```

SOURce:WAVelength?

機能

波長を問い合わせます。
[全波長] が設定されている場合，最初の波長が返ります。

文法

```
SOURce:WAVelength?
```

パラメータ

無し

レスポンスデータ

現在の波長の値

エラー

無し

使用例

```
sour:wav?
> 1310<END>
```

SOURce:RANge:AVAilable?

機能

設定されている波長に対して，使用可能な距離レンジのリストを問い合わせます。

文法

```
SOURce:RANge:AVAilable?
```

パラメータ

無し

レスポンスデータ
設定できる距離レンジのリスト, 単位 km

エラー
無し

使用例
```
sour:ran:ava?
> 5.0, 10.0, 20.0, 50.0, 100.0, 200.0, 300.0<END>
```

SOURce:RANge

機能
距離レンジを設定します。

文法
```
SOURce:RANge<wsp><value>
```

パラメータ
<value>
数値形式
範囲: 機種に依存します。SOURce:RANge:AVAilable?で取得した距離レンジの数値

レスポンスデータ
無し

エラー
(–200, "std_execGen, Test is Active")
(–224, "std_illegalParmValue, Invalid Parameter Value")

使用例
```
sour:ran 100
```

SOURce:RANge?

機能
設定されている距離レンジを問い合わせます。

文法
```
SOURce:RANge?
```

パラメータ
無し

レスポンスデータ
現在の距離レンジの数値, 単位 km

エラー
無し

使用例
```
sour:ran?
> 50.0<END>
```

SOURce:RESo:AVAilable?

機能
設定されている距離レンジに対して設定できる分解能のリストを問い合わせます。

文法
```
SOURce:RESo:AVAilable?
```

パラメータ
無し

レスポンスデータ
設定できる分解能のリスト:
0: 標準
1: 高密度
2: 超高密度

エラー
無し

使用例
```
sour:res:ava?
> 0, 1, 2<END>
```

SOURce:RESo

機能
分解能を設定します。

文法

SOURce:RESo<wsp><value>

パラメータ

<value>
整数値
範囲: 0|1|2
0: 標準
1: 高密度
2: 超高密度
設定できる分解能は，設定されている距離レンジによって変わります。

レスポンスデータ

無し

エラー

(–200, "std_execGen, Test is Active")
(–224, "std_illegalParmValue, Invalid Parameter Value")

使用例

sour:res 1

SOURce:RESo?

機能

設定されている分解能を問い合わせます。

文法

SOURce:RESo?

パラメータ

無し

レスポンスデータ

現在の分解能を示す数値:
0: 標準
1: 高密度
2: 超高密度

エラー

無し

使用例

```
sour:res?
> 1<END>
```

SOURce:PULSe:AVAilable?

機能

設定されている距離レンジおよび分解能に対して，設定可能なパルス幅のリストを問い合わせます。

文法

```
SOURce:PULSe:AVAilable?
```

パラメータ

無し

レスポンスデータ

設定できるパルス幅のリスト，ns 単位

エラー

無し

使用例

```
sour:puls:ava?
> 10,20,50,100<END>
```

SOURce:PULSe

機能

パルス幅を設定します。

文法

```
SOURce:PULSe<wsp><value>
```

パラメータ

<value>
整数値
範囲: SOURce:PULSe:AVAilable?で取得したパルス幅の数値
設定できるパルス幅は，距離レンジおよび分解能の設定によって変わります。

レスポンスデータ
無し

エラー
(–200, "std_execGen, Test is Active")
(–224, "std_illegalParmValue, Invalid Parameter Value")

使用例
```
sour:puls 100
```

SOURce:PULSe?

機能
設定されているパルス幅を問い合わせます。

文法
```
SOURce:PULSe?
```

パラメータ
無し

レスポンスデータ
現在設定されているパルス幅の数値, ns 単位

エラー
無し

使用例
```
sour:puls?
> 100<END>
```

SOURce:PULSe:ENHanced:AVAilable?

機能
設定されているパルス幅に対して, デッドゾーンの高ダイナミックレンジが設定できるか問い合わせます。

文法
```
SOURce:PULSe:ENHanced:AVAilable?
```

パラメータ
無し

レスポンスデータ

ブーリアン形式

1 または 0

0: 設定不可

1: 設定可能

エラー

無し

使用例

```
sour:puls:enh:ava?
> 1<END>
```


SOURce:PULSe:ENHanced

機能

設定されているパルス幅に対して，デッドゾーンの設定をします。

文法

```
SOURce:PULSe:ENHanced<wsp><value>
```

パラメータ

<value>

ブーリアン形式

範囲: 1|0

0: 標準（HR）

1: 高ダイナミックレンジ（ER）

レスポンスデータ

無し

エラー

(–200, "std_execGen, Test is Active")

(–104, "std_wrongParamType, Data Type Error")

(–224, "std_illegalParmValue, Invalid Parameter Value")

使用例

```
sour:puls:enh 0
```

SOURce:PULSe:ENHanced?

機能

設定されているパルス幅に対して，デッドゾーンの設定を問い合わせます。

文法

```
SOURce:PULSe:ENHanced?
```

パラメータ

無し

レスポンスデータ

ブーリアン形式
0: 標準（HR）
1: 高ダイナミックレンジ（ER）

エラー

無し

使用例

```
sour:puls:enh?
> 1<END>
```

SOURce:AVERages:TIMe

機能

次回の測定で使用する平均化時間（秒数）を設定します。
また，平均化単位を秒数に変更します。

文法

```
SOURce:AVERages:TIMe<wsp><value>
```

パラメータ

<value>
整数値
範囲: 1～3600

レスポンスデータ

無し

エラー

(–200, "std_execGen, Test is Active")
(–224, "std_illegalParmValue, Invalid Parameter Value")

使用例

```
sour:aver:tim 120<END>
```

SOURce:AVERages:TIMe?

機能

次回の測定で使用する平均化時間（秒数）を問い合わせます。

文法

```
SOURce:AVERages:TIMe?
```

パラメータ

無し

レスポンスデータ

平均化秒数
平均化単位が秒数でない場合はエラーとなります。

エラー

(–400, "std_queryGen, Invalid Average Mode")

使用例

```
sour:aver:tim?
> 120<END>
```

SOURce:AVERages:COUNt

機能

次回の測定で使用する平均化回数（掃引回数）を設定します。
また，平均化単位を回数に変更します。
1 回の試験中に掃引する数で平均化します。
平均化回数（1～3600）を設定します。

文法

```
SOURce:AVERages:COUNt<wsp><value>
```

パラメータ

<value>

整数値
範囲: 1～3600

レスポンスデータ
無し

エラー
(–200, "std_execGen, Test is Active")
(–224, "std_illegalParmValue, Invalid Parameter Value")

使用例
```
sour:aver:coun 1000<END>
```

SOURce:AVERages:COUNt?

機能
次回の測定で使用する平均化回数（掃引回数）を問い合わせます。

文法
```
SOURce:AVERages:COUNt?
```

パラメータ
無し

レスポンスデータ
平均化回数（掃引回数)
平均化単位が回数でない場合はエラーとなります。

エラー
(–400, "std_queryGen, Invalid Average Mode")

使用例
```
sour:aver:coun?
> 1000<END>
```

SOURce:AVERages:EXPonent

機能
次回の測定で使用する平均化回数（掃引回数）を総パルス送出回数から設定します。
また，平均化単位を回数に変更します。
総パルス送出回数 ＝ 掃引回数 × 1 回の掃引で送出するパルス数

総パルス送出回数は 2 の指数形式で設定します。

文法

```
SOURce:AVERages:EXPonent<wsp><value>
```

パラメータ

<value>
文字列
範囲: 2^N　N:8～21

注:

総パルス送出回数は，距離レンジなどの測定条件により設定範囲が制限されることがあります。指定した回数が設定されているか，SOURce:AVERages:EXPonent?で確認してください。

また，測定波形によっては測定結果の総パルス送出回数が設定値と異なることがあります。測定結果の総パルス送出回数はSENSe:AVERages?で確認できます。

レスポンスデータ

無し

エラー

(–200, "std_execGen, Test is Active")
(–224, "std_illegalParmValue, Invalid Parameter Value")

使用例

```
sour:aver:exp 10<END>
```

SOURce:AVERages:EXPonent?

機能

総パルス送出回数を 2 の指数形式で問い合わせます。

文法

```
SOURce:AVERages:EXPonent?
```

パラメータ

無し

レスポンスデータ

平均化回数

<2^N　　総パルス送出回数が 2^{N-1} を超えて 2^N 未満のとき
2^N　　総パルス送出回数が 2^N のとき
平均化単位が回数ではない場合はエラーとなります。

エラー

(–400, "std_queryGen, Invalid Average Mode")

使用例

```
sour:aver:exp 16
sour:aver:exp?
> 2^16<END>
```

4.4 SENSe サブシステムコマンド

SENSe:AVERages?

機能

測定を開始してから平均化した回数を問い合わせます。

文法

```
SENSe:AVERages?
```

パラメータ

無し

レスポンスデータ

平均化した回数

エラー

(–400, "std_queryGen, Trace Not Ready")

使用例

```
sens:aver?
> 4096<END>
```

SENSe:AVERages:TIMe?

機能

測定を開始してから平均化した時間（秒数）を問い合わせます。
距離レンジと分解能の設定によっては，設定した平均化時間よりも実際の平均化時間が長くなることがあります。

文法

```
SENSe:AVERages:TIMe?
```

パラメータ

無し

レスポンスデータ

秒数

エラー

(–400, "std_queryGen, Trace Not Ready")

使用例

```
sens:aver:tim?
> 28<END>
```

SENSe:TRACe:READY?

機能

トレースデータを出力する準備ができているか問い合わせます。

文法

```
SENSe:TRACe:READY?
```

パラメータ

無し

レスポンスデータ

1: トレースデータの準備が完了してデータの転送ができる。
0: メモリにトレースデータが存在しない。

エラー

無し

使用例

```
sens:trac:ready?
> 1<END>
```

SENSe:CONCheck

機能

接続チェックを On または Off に設定します。

文法

```
SENSe:CONCheck<wsp><value>
```

パラメータ

<value>
ブーリアン形式
1 または on: 接続チェック On
0 または off: 接続チェック Off

レスポンスデータ

無し

エラー

(–200, "std_execGen, Test is Active")
(–104, "std_wrongParamType, Data Type Error")

使用例

```
sens:conc ON<END>
```

SENSe:CONCheck?

機能

接続チェックが On かどうか問い合わせます。

文法

```
SENSe:CONCheck?
```

パラメータ

無し

レスポンスデータ

1: 接続チェック On
0: 接続チェック Off

エラー

無し

使用例

```
sens:conc?
> 1<END>
```

SENSe:LIVCheck

機能

通信光チェックを On または Off に設定します。

文法

```
SENSe:LIVCheck<wsp><value>
```

パラメータ

<value>
ブーリアン形式
1 または on: 通信光チェック On
0 または off: 通信光チェック Off

レスポンスデータ
無し

エラー
(–200, "std_execGen, Test is Active")
(–104, "std_wrongParamType, Data Type Error")

使用例
```
sens:livc ON<END>
```

SENSe:LIVCheck?

機能
通信光チェックが On かどうか問い合わせます。

文法
```
SENSe:LIVCheck?
```

パラメータ
無し

レスポンスデータ
1: 接続チェック On
0: 接続チェック Off

エラー
無し

使用例
```
sens:livc?
> 1<END>
```

SENSe:FIBCheck

機能
ファイバ長チェックを On または Off に設定します。

文法
```
SENSe:FIBCheck<wsp><value>
```

パラメータ
<value>

ブーリアン形式
1 または on: ファイバ長チェック On
0 または off: ファイバ長チェック Off

レスポンスデータ
無し

エラー
(–200, "std_execGen, Test is Active")
(–104, "std_wrongParamType, Data Type Error")

使用例
```
sens:fibc ON<END>
```

SENSe:FIBCheck?

機能
ファイバ長チェックが On かどうか問い合わせます。

文法
```
SENSe:FIBCheck?
```

パラメータ
無し

レスポンスデータ
1: ファイバ長チェック On
0: ファイバ長チェック Off

エラー
無し

使用例
```
sens:fibc?
> 1<END>
```

SENSe:FIBer:IOR

機能
ファイバの IOR (群屈折率) を設定します。この値は, 次回の測定から使用されます。

文法

SENSe:FIBer:IOR<wsp><value>

パラメータ

<value>
浮動小数点形式
範囲: 1.3～1.7

レスポンスデータ

無し

エラー

(–200, "std_execGen, Test is Active")
(–224, "std_illegalParmValue, Invalid Parameter Value")

使用例

```
sens:fib:ior 1.45
```

SENSe:FIBer:IOR?

機能

ファイバの IOR を問い合わせます。

文法

SENSe:FIBer:IOR?

パラメータ

無し

レスポンスデータ

1.3～1.7

エラー

(–200, "std_execGen, Test is Inactive")

使用例

```
sens:fib:ior?
> 1.450000<END>
```

SENSe:FIBer:BSC

機能

ファイバの BSC（後方散乱光係数）を設定します。この値は，次回の測定から使用されます。

文法

```
SENSe:FIBer:BSC<wsp><value>
```

パラメータ

<value>
浮動小数点形式
範囲: –90.0～–40.0

レスポンスデータ

無し

エラー

(–200, "std_execGen, Test is Active")
(–224, "std_illegalParmValue, Invalid Parameter Value")

使用例

```
sens:fib:bsc -83.0
```

SENSe:FIBer:BSC?

機能

ファイバの BSC を問い合わせます。

文法

```
SENSe:FIBer:BSC?
```

パラメータ

無し

レスポンスデータ

–90.0～–40.0

エラー

無し

使用例

```
sens:fib:bsc?
> -83.0<END>
```

SENSe:HOFFset

機能

水平オフセット値を設定します。 オフセット値の単位は km です。

文法

```
SENSe:HOFFset<wsp><value>
```

パラメータ

<value>
浮動小数点形式
範囲: オフセット値は正負の最大距離レンジまでを設定できます。

レスポンスデータ

無し

エラー

(–224, "std_illegalParmValue, Invalid Parameter Value")

使用例

```
sens:hoff 10.0
```

SENSe:HOFFset?

機能

表示されているトレースの水平オフセット値を問い合わせます。

文法

```
SENSe:HOFFset?
```

パラメータ

無し

レスポンスデータ

水平オフセット値

エラー

無し

使用例

```
sens:hoff?
> 0<END>
```

SENSe:VOFFset

機能

表示されているトレースの垂直オフセット値を設定します。単位は dB です。

文法

```
SENSe:VOFFset<wsp><value>
```

パラメータ

<value>
浮動小数点形式
範囲: オフセット値は，正負のダイナミックレンジまでを設定できます。
(–64.0～64.0)

レスポンスデータ

無し

エラー

(–224, "std_illegalParmValue, Invalid Parameter Value")

使用例

```
sens:voff -5.0
```

SENSe:VOFFset?

機能

表示されているトレースの垂直オフセット値を問い合わせます。

文法

```
SENSe:VOFFset?
```

パラメータ

無し

レスポンスデータ

垂直オフセット値

エラー
無し

使用例
```
sens:voff?
> 0<END>
```

SENSe:ACURsor

機能
カーソル A の位置を設定します。単位は km です。
注:
SCPI コマンドによるリモート制御をすると，マーカ操作は強制的に[移動方式] に切り替わります。

文法
```
SENSe:ACURsor<wsp><value>
```

パラメータ
<value>
浮動小数点形式
範囲: 0.0～設定されている距離レンジ

レスポンスデータ
無し

エラー
(–224, "std_illegalParmValue, Invalid Parameter Value")

使用例
```
sens:acur 20.5
```

SENSe:ACURsor?

機能
カーソル A の位置を問い合わせます。

文法
```
SENSe:ACURsor?
```

パラメータ
無し

レスポンスデータ
カーソル A の位置

エラー
無し

使用例
```
sens:acur?
> 20.5<END>
```

SENSe:BCURsor

機能
カーソル B の位置を設定します。単位は km です。
注:
SCPI コマンドによるリモート制御をすると，マーカ操作は強制的に [移動方式] に切り替わります。

文法
```
SENSe:BCURsor<wsp><value>
```

パラメータ
<value>
浮動小数点形式
範囲: 0.0～設定されている距離レンジ

レスポンスデータ
無し

エラー
(–224, "std_illegalParmValue, Invalid Parameter Value")

使用例
```
sens:bcur 20.5
```

SENSe:BCURsor?

機能
カーソル B の位置を問い合わせます。

文法
```
SENSe:BCURsor?
```

パラメータ
無し

レスポンスデータ
カーソル B の位置

エラー
無し

使用例
```
sens:bcur?
> 20.5<END>
```

SENSe:LSALeft

機能
左側 LSA マーカの開始位置と終了位置を設定します。開始位置と終了位置の単位は km です。
注:
SCPI コマンドによるリモート制御をすると，マーカ操作は強制的に [移動方式] に切り替わります。

文法
```
SENSe:LSALeft<wsp><start>,<stop>
```

パラメータ
<start>
浮動小数点形式
範囲: –100.0～400.0.
<stop>
浮動小数点形式
範囲: –100.0～400.0.
開始位置の数値は，終了位置の数値以下にしてください

レスポンスデータ
無し

エラー
(–200, "std_execGen, LSA Inactive State")
(–224, "std_illegalParmValue, Invalid Parameter Value")

使用例

```
sens:lsal 0.0,0.5
```

SENSe:LSALeft?

機能

左側 LSA マーカ位置の値を問い合わせます。

文法

```
SENSe:LSALeft?
```

パラメータ

無し

レスポンスデータ

左側 LSA マーカの開始位置と終了位置。 開始位置と終了の単位は km です。

エラー

(–400, "std_queryGen, LSA Inactive State")

使用例

```
sens:lsal?
> 0.0,0.5<END>
```

SENSe:LSARight

機能

右側 LSA マーカの開始位置と終了位置を設定します。開始位置と終了位置の単位は km です。

注:

SCPI コマンドによるリモート制御をすると，マーカ操作は強制的に[移動方式] に切り替わります。

文法

```
SENSe:LSARight<wsp><start>,<stop>
```

パラメータ

<start>

浮動小数点形式

範囲: –100.0～400.0.

<stop>

浮動小数点形式
範囲: –100.0～400.0.
開始位置の数値は, 終了位置の数値以下にしてください

レスポンスデータ
無し

エラー
(–200, "std_execGen, LSA Inactive State")
(–224, "std_illegalParmValue, Invalid Parameter Value")

使用例
```
sens:lsal 0.0,0.5
```

SENSe:LSARight?

機能
右側 LSA マーカ位置の値を問い合わせます。

文法
```
SENSe:LSARight?
```

パラメータ
無し

レスポンスデータ
右側 LSA マーカの開始位置と終了位置。 開始位置と終了の単位は km です。

エラー
(–400, "std_queryGen, LSA Inactive State")

使用例
```
sens:lsar?
> 0.0, 0.5<END>
```

SENSe:LOSS:MODE

機能

損失計算方法を設定します。

文法

```
SENSe:LOSS:MODE<wsp><value>
```

パラメータ

<value>
整数値
範囲: 0|1|2|3|4|5|6
0: 接続損失
1: 2-Pt 損失
2: 2-Pt LSA
3: dB/km 損失
4: dB/km LSA
5: 2-Pt, dB/km
6: 全反射減衰量

レスポンスデータ

無し

エラー

(–224, "std_illegalParmValue, Invalid Parameter Value")

使用例

```
sens:loss:mode 0
```

SENSe:LOSS:MODE?

機能

損失計算方法を問い合わせます。

文法

```
SENSe:LOSS:MODE?
```

パラメータ

無し

レスポンスデータ

0: 接続損失

1: 2-Pt 損失
2: 2-Pt LSA
3: dB/km 損失
4: dB/km LSA
5: 2-Pt, dB/km
6: 全反射減衰量

エラー
無し

使用例
```
sens:loss:mode?
> 0<END>
```

SENSe:ORL:MODE

機能
測定機能（2-2）の全反射減衰量計算を設定します。

文法
```
SENSe:ORL:MODE<wsp><value>
```

パラメータ
<value>
整数値
範囲: 0 | 1 | 2
0: A カーソル
1: 近端
2: 全体波形

レスポンスデータ
無し

エラー
(–224, "std_illegalParmValue, Invalid Parameter Value")

使用例
```
sens:loss:mode 0
```

SENSe:ORL:MODE?

機能

測定機能（2-2）の全反射減衰量計算を問い合わせます。

文法

```
SENSe:ORL:MODE?
```

パラメータ

無し

レスポンスデータ

0: A カーソル
1: 近端
2: 全体波形

エラー

無し

使用例

```
sens:orl:mode?
> 0<END>
```

SENSe:ANALyze:PARameters

機能

次回の試験で使用する自動検出のパラメータを設定します。

文法

```
SENSe:ANALyze:PARameters<wsp><splice loss>,
<reflectance>,<end loss>,<pon loss>
```

パラメータ

倍精度:

<splice loss>	接続損失	範囲: 0.01～9.99
<reflectance>	反射減衰量	範囲: –70.0～–20.0
<end loss>	遠端損失	範囲: 1～99
<pon loss>	スプリッタ損失	範囲: 1.0～30.0

レスポンスデータ

無し

エラー

(–224, "std_illegalParmValue, Invalid Parameter Value")

使用例

```
sens:anal:par 0.05,-60.0,3.0,10.0
```

SENSe:ANALyze:PARameters?

機能

自動検出のパラメータを問い合わせます。

文法

```
SENSe:ANALyze:PARameters?
```

パラメータ

無し

レスポンスデータ

<event loss>,<reflectance>,<end loss>,<pon loss><END>

エラー

無し

使用例

```
sens:anal:par?
> 0.050000,-60.000000,3.000000,10.000000<END>
```

SENSe:ANALyze:PARameters:ENDRefl

機能

次回の試験で使用する自動検出のパラメータを設定します。

文法

```
SENSe:ANALyze:PARameters:ENDRefl<wsp>
<end reflectance>
```

パラメータ

倍精度:

<end reflectance>　遠端反射減衰量　範囲: –70.0～–10.0, 0
0 は無効を設定します。

レスポンスデータ
無し

エラー
(–224, "std_illegalParmValue, Invalid Parameter Value")

使用例
```
sens:anal:par:endr -25.0
```

SENSe:ANALyze:PARameters:ENDRefl?

機能
自動検出のパラメータを問い合わせます。

文法
```
SENSe:ANALyze:PARameters:ENDRefl?
```

パラメータ
無し

レスポンスデータ
<end reflectance><END>

エラー
無し

使用例
```
sens:anal:par:endr?
> -25.000000<END>
```

SENSe:ANALyze:AUTO

機能
試験が完了した後の自動解析を On または Off に設定します。

文法
```
SENSe:ANALyze:AUTO<wsp><value>
```

パラメータ
<value>
ブーリアン形式
範囲: 0|1

0:自動解析 Off
1:自動解析 On

レスポンスデータ
無し

エラー
(–104, "std_wrongParamType, Data Type Error")

使用例
```
sens:anal:auto 0
```

SENSe:ANALyze:AUTO?

機能
自動解析が On かどうか問い合わせます。

文法
```
SENSe:ANALyze:AUTO?
```

パラメータ
無し

レスポンスデータ
0: 自動解析 Off
1: 自動解析 On

エラー
無し

使用例
```
sens:anal:auto?
> 0<END>
```

4.5 TRACe サブシステムコマンド

TRACe サブシステムは, トレースデータの取得とトレースの解析を制御できます。

TRACe:PARameters?

機能

トレースのパラメータをテキスト形式で問い合わせます。

文法

```
TRACe:PARameters?
```

パラメータ

無し

レスポンスデータ

トレースのパラメータ。コンマで区切られた数値です。

<wave>, <range>, <pulse>, <avg>, <reso>, <ior>, <bsc>, <enh><END>

エラー

(–400, "std_queryGen, Trace Not Ready")

使用例

```
trac:par?
> 1310, 16.415554, 50, 6144, 0.656621, 1.467700, -
78.500000, 1<END>
```

TRACe:ANALyze

機能

トレースの解析を実行します。

自動解析のパラメータを変更した後, または自動解析の設定が Off の時に, 本コマンドを使用できます。

本コマンドは他のコマンドと同時に実行できるオーバーラップコマンドです。

文法

```
TRACe:ANALyze
```

パラメータ

無し

レスポンスデータ
無し

エラー
(–200, "std_execGen, Test is Active")
(–200, "std_execGen, Trace Not Ready")

使用例
```
trac:anal
```

TRACe:ANALyze?

機能
トレースの解析が終了したかを問い合わせます。

文法
```
TRACe:ANALyze?
```

パラメータ
無し

レスポンスデータ
0: トレースは未解析
1: トレースの解析済み

エラー
(–400, "std_queryGen, Trace Not Ready")

使用例
```
trac:anal?
> 0<END>
```

TRACe:ANALyze:ORL

機能
トレースの全反射減衰量を計算します。
本コマンドは他のコマンドと同時に実行できるオーバーラップコマンドです。

文法
```
TRACe:ANALyze:ORL
```

パラメータ
無し

レスポンスデータ
無し

エラー
(–200, "std_execGen, Test is Active")
(–200, "std_execGen, Trace Not Ready")
(–200, "std_execGen, Invalid Loss Mode")

使用例
```
trac:anal:orl
```

TRACe:MDLOss?

機能
損失の値を問い合わせます。読み取った値は，設定されている損失計算方法によって変わります。損失の値が 1 つだけの場合は，最初の数値のみが有効な値です。損失が計算されていない場合，レスポンスの値は–99.99 になります。

文法
```
TRACe:MDLOss?
```

パラメータ
無し

レスポンスデータ
コンマで区切られた 2 つの損失計算値

エラー
(–200, "std_execGen, Test is Active")
(–400, "std_queryGen, Trace Not Ready")

使用例
```
trac:mdlo?
> -4.610,-99.99<END>
```

TRACe:EELOss?

機能

トレースの全損失を問い合わせます。
全損失が計算できない場合，レスポンスは–99.99 になります。

文法

```
TRACe:EELOss?
```

パラメータ

無し

レスポンスデータ

計算した全損失の値

エラー

(–200, "std_execGen, Test is Active")
(–400, "std_queryGen, Trace Not Ready")

使用例

```
trac:eelo?
> -4.610<END>
```

TRACe:LOAD:SOR?

機能

トレースデータを SOR ファイル形式で問い合わせます。
データ形式の詳細については，「2.6.3 データの書式」のバイナリデータを参照してください。

文法

```
TRACe:LOAD:SOR?
```

パラメータ

無し

レスポンスデータ

SCPI バイナリ転送の規格に従ったバイト単位の配列の SOR トレースファイル

エラー

(–200, "std_execGen, Test is Active")

(–400, "std_queryGen, Trace Not Ready")

使用例

```
trac:load:sor?
> "BINARY ARRAY"
```

#524047 //バイナリデータ転送用の SCPI データサイズメッセージ
後に SOR ファイルのバイト列が続きます。

TRACe:LOAD:TEXT?

機能

TRACe:LOAD:TEXT? コマンドには 3 つの方式があります。

TRACe:LOAD:TEXT?

テキスト形式の SOR ファイル情報と，すべてのトレースデータを取得します。

TRACe:LOAD:TEXT? <start>

<start>: トレースデータの開始点，距離の単位は km
指定した開始点から距離レンジの最終点までのトレースデータを取得します。

TRACe:LOAD:TEXT? <start>,<end>

<start>: トレースデータの開始点，距離の単位は km
<end>: トレースデータの終了点，距離の単位は km
指定した開始点から終了点までのトレースデータを取得します。

データ形式の詳細については，「2.6.3 データの書式」のバイナリデータを参照してください。

文法

```
TRACe:LOAD:TEXT?
TRACe:LOAD:TEXT?<wsp><start>
TRACe:LOAD:TEXT?<wsp><start>,<end>
```

パラメータ

<start>
浮動小数点形式
範囲: 0.000000～設定されている距離レンジ
<end>
浮動小数点形式
範囲: 0.000000～設定されている距離レンジ
<start>の値は，<end>以下にしてください。

レスポンスデータ

SOR トレースデータ。
SCPI バイナリ転送の規格に従ったバイト単位の配列です。

エラー

(–108, "std_tooManyParameters, Parameter not Allowed")
(–200, "std_execGen, Test is Active")
(–224, "std_illegalParmValue, Invalid Parameter Value")
(–400, "std_queryGen, Trace Not Ready")

使用例

```
trac:load:text?
>#6140479 //SCPI data size message for binary data
transfer
WL   = 1310 nm //Wavelength
FBR  = SM //Fiber Type
DR   = 5 km //
PW   = 50 ns [ER] //Pulse Width and resolution type
AVG  = 6144 //Number of hardware averages
IOR  = 1.467700 //IOR value
BSC  = -78.50 //BSC value
DATE = 08/13/09 //Date of test
TIME = 10:19 PM //Time of test
MXDB = 64 dB //dB Range
RESO = 0.200 m //Resolution value
DX   = 0.20440100621721 m //Point spacing
PTS  = 25001 //Number of data points in the trace

//Start of trace data points
1000
9741
41291
41923
.....

9741
9741
0
//End of trace data points
```

```
Events  1 //Number of events found by analysis

Dist              1.0505 km //Event distance
Type              E //Event type
Loss              >3.00 dB //Event loss value
Reflectance       N/A //Event reflectance value
dB / km           59.420 dB //dB/km Loss value
Cumulative Loss   1.81 dB //Cumulative loss value
```

TRACe:LOAD:DATA?

機能

TRACe:LOAD:DATA?は 4 つの方式があります。

TRACe:LOAD:DATA?

すべてのトレースデータを取得します。

TRACe:LOAD:DATA? <start>

<start>: トレースデータの開始点, 距離の単位は km

指定した開始点から距離レンジの最終点までのトレースデータを取得します。

TRACe:LOAD:DATA? <start>,<end>

<start>: トレースデータの開始点, 距離の単位は km

<end>: トレースデータの終了点, 距離の単位は km

指定した開始点から終了点までのトレースデータを取得します。

TRACe:LOAD:DATA? <start>,<end>,<space>

<start>: トレースデータの開始点, 距離の単位は km

<end>: トレースデータの終了点, 距離の単位は km

<space>: 分解能で表したデータ間隔

指定した開始点から終了点までのトレースデータを, 指定したデータ点間隔ごとに取得します。

データ形式の詳細については,「2.6.3 データの書式」のバイナリデータを参照してください。

文法

```
TRACe:LOAD:DATA?<wsp><start>,<end>,<space>
```

パラメータ

<start> 開始点の距離 (km)

浮動小数点形式

範囲: 0.0～(設定されている距離レンジ–<space>*分解能)

<end> 終了点の距離 (km)
浮動小数点形式
範囲: (<start>+<space>*分解能)～(設定されている距離レンジ)

<space>分解能で表したデータ間隔
数値形式
範囲: 1～(<start> と <end>の間の最大データポイント数) <start>開始点の距離 (km)
数値形式
範囲: 0.0～(設定されている距離レンジ–<space>*分解能)

開始点の値は終了点の値以下にしてください。

レスポンスデータ
指定したデータ点
SCPI バイナリ転送の規格に従ったバイト単位の配列です。

エラー
(–225, "std_illegalState, Trace Not Ready")
(–224, "std_illegalValue, Invalid Parameter Value")

使用例
```
trac:load:data? 0.0,10.0,1
> "BINARY ARRAY"
```
#510006 //バイナリデータ転送用の SCPI データサイズメッセージ
後にデータポイントのバイト列が続きます。
最初の 4 バイト – 以降のデータポイント数を表す符号なし整数,
以降の各 2 バイト– 要求したデータポイントの符号なし単精度整数

TRACe:HEADer

機能
ファイルヘッダを設定します。空欄に設定する場合は, パラメータに文字を設定しません。
ただし, データフラグ<Data Flag>と方向<Direction>は空欄にできません。

コメントを空欄に設定する場合は, 最後の引数にコンマを追加します。

文法

```
TRACe:HEADer<wsp><Data Flag>,<Cable ID>,<Fiber ID>,
<Cable Code>,<Start location>,<Terminal Location>,
<Direction>,<Operator>,<Comment>
```

パラメータ

<Data Flag>	BC, RC, OT
<Cable ID>	30 文字まで
<Fiber ID>	30 文字まで
<Cable Code>	30 文字まで
<Start location>	30 文字まで
<Terminal Location>	30 文字まで
<Direction>	0: A->B, 1: B->A
<Operator>	30 文字まで
<Comment>	30 文字まで

レスポンスデータ

無し

エラー

(–108, "std_tooManyParameters, Parameter not Allowed")
(–109, "std_tooFewParameters, Missing Parameter")
(–224, "std_illegalParmValue, Invalid Parameter Value")

使用例

```
trac:head OT,1,1,ABC,Tokyo,Yokohama,1,Anritsu,TEST
trac:head BC,,,,,,0,,,
```

TRACe:HEADer?

機能

ファイルヘッダを問い合わせます。

文法

```
TRACe:HEADer?
```

パラメータ

無し

レスポンスデータ

9 つのヘッダ

エラー

(–108, "std_tooManyParameters, Parameter not Allowed")

使用例

```
trac:head?
> OT, 1,1,ABC,Tokyo,Yokohama,1,Anritsu,TEST<END>
```

TRACe:STORe:SOR

機能

アクセスマスタの内蔵メモリに SOR ファイルを保存します。

文法

```
TRACe:STORe:SOR<wsp><filepath&name>
```

パラメータ

<filepath&name>　　SOR ファイルを保存するファイルパスとファイル名

レスポンスデータ

無し

エラー

(–108, "std_tooManyParameters, Parameter not Allowed")
(–109, "std_tooFewParameters, Missing Parameter")
(–224, "std_illegalParmValue, Invalid Parameter Value")
(–400, "std_queryGen, Trace Not Ready")

使用例

```
trac:stor:sor test.sor
```

test.sor ファイルは内蔵メモリのルートフォルダに保存されます。

```
trac:stor:sor TEST/test.sor
```

test.sor ファイルは内蔵メモリの TEST フォルダに保存されます。

4.6 DISPlay サブシステムコマンド

The DISPlay サブシステムは, 表示モードと表示のズームとスケール設定を提供します。

DISPlay:MODE

機能

距離表示の基準カーソルを設定します。

文法

```
DISPlay:MODE<wsp><mode>
```

パラメータ

<mode> 基準カーソル
数値形式
範囲: 0|1
0: カーソル A
1: カーソル B

レスポンスデータ

無し

エラー

(–224, "std_illegalValue, Invalid Parameter Value")

使用例

```
disp:mode 1
```

DISPlay:MODE?

機能

距離表示の基準カーソルを問い合わせます。

文法

```
DISPlay:MODE?
```

パラメータ

無し

レスポンスデータ

基準カーソルの数値

エラー
無し

使用例
```
disp:mode?
> 1
```

DISPlay:ZOOM:FULL

機能
全体波形が表示されるようにズームを設定します。

文法
```
DISPlay:ZOOM:FULL
```

パラメータ
無し

レスポンスデータ
無し

エラー
無し

使用例
```
disp:zoom:full
```

DISPlay:ZOOM:HORIzontal

機能
水平スケールのズームレベルを設定します。

文法
```
DISPlay:ZOOM:HORIzontal<wsp><level>
```

パラメータ
<level> ズームレベル
数値形式
範囲: 0～(6～16) 設定されている距離レンジによって変わります。
0: 最大ズームイン (拡大表示)
6～16: 最大ズームアウト (全体表示)

注:

距離レンジごとのズームレベル最大値を以下に示します。

0.5 km:	6	25 km:	11
1 km:	7	50 km:	12
2.5 km:	8	100 km:	13
5 km:	9	200 km:	14
10 km:	10	300 km:	16

レスポンスデータ

無し

エラー

(–224, "std_illegalValue, Invalid Parameter Value")

使用例

```
disp:zoom:hori 2
```

DISPlay:ZOOM:HORIzontal?

機能

水平スケールのズームレベルを問い合わせます。

文法

```
DISPlay:ZOOM:HORIzontal?
```

パラメータ

無し

レスポンスデータ

水平ズームレベルの数値

エラー

無し

使用例

```
disp:zoom:hori?
> 5<END>
```

DISPlay:ZOOM:VERTical

機能

垂直スケールのズームレベルを設定します。

文法

```
DISPlay:ZOOM:VERTical <wsp><level>
```

パラメータ

<level> ズームレベル
数値形式
範囲: 0～6
0: 最大ズームイン (拡大表示)
6: 最大ズームアウト (全体表示)

レスポンスデータ

無し

エラー

(–224, "std_illegalValue, Invalid Parameter Value")

使用例

```
disp:zoom:vert 2
```

DISPlay:ZOOM:VERTical?

機能

垂直スケールのズームレベルを問い合わせます。

文法

```
DISPlay:ZOOM:VERTical?
```

パラメータ

無し

レスポンスデータ

垂直ズームレベルの数値

エラー

無し

使用例

```
disp:zoom:vert?
```

```
> 5<END>
```

DISPlay:SCALe:HORIzontal

機能

水平スケールを指定した距離に設定します。
水平スケールに指定した距離が画面に表示されます。

文法

```
DISPlay:SCALe:HORIzontal<wsp><scale>
```

パラメータ

<scale>水平スケールの距離（km）
浮動小数点形式
範囲: 0.0124～(0.5022～300.0056)
0.0124: 最大ズームイン
0.5022～300.0056: 最大ズームアウト
距離レンジの設定（0.5～300 km）によって変わります。

注:

あらかじめ定義済みのスケールが 17 あります（0.0124, 0.0186, 0.0310, 0.0558, 0.1054, 0.2542, 0.5022, 1.0044, 2.5048, 5.0034, 10.0006, 25.0046, 50.0030, 100.0060, 200.0058, 250.0026, 300.0056）。指定した <scale> の値は，その値よりも大きくて最も近い定義済みスケールに変更されます。
距離レンジごとの最大スケールを以下に示します。

0.5 km:	0.5022	25 km:	25.0046
1 km:	1.0054	50 km:	50.0030
2.5 km:	2.5048	100 km:	100.0060
5 km:	5.0034	200 km:	200.0058
10 km:	10.0006	300 km:	300.0056

レスポンスデータ

無し

エラー

(–224, "std_illegalValue, Invalid Parameter Value")

使用例

```
disp:scal:hori 5.0
```

DISPlay:SCALe:HORIzontal?

機能

水平スケールの距離（km）を問い合わせます。

文法

```
DISPlay:SCALe:HORIzontal?
```

パラメータ

無し

レスポンスデータ

水平スケールの距離を表す数値（km）

エラー

無し

使用例

```
disp:scal:hori?
> 5.0034<END>
```

DISPlay:SCALe:VERTical

機能

垂直スケールを指定した値に設定します。
垂直スケールに指定した値が画面に表示されます。

文法

```
DISPlay:SCALe:VERTical<wsp><scale>
```

パラメータ

<scale>垂直スケールの値（dB）
浮動小数点形式
範囲: 0.5～65
0.5: 最大ズームイン（拡大表示）
65: 最大ズームアウト（全体表示）

注:

あらかじめ定義済みのスケールが 7 あります（0.5, 1.0, 2.5, 5.0, 10.0, 25.0, 65.0）。指定した <scale> の値は，その値よりも大きくて最も近い定義済みスケールに変更されます。

レスポンスデータ
無し

エラー
(–224, "std_illegalValue, Invalid Parameter Value")

使用例
```
disp:scal:vert 0.5
```

DISPlay:SCALe:VERTical?

機能
垂直スケールの値（dB）を問い合わせます。

文法
```
DISPlay:SCALe:VERTical?
```

パラメータ
無し

レスポンスデータ
垂直スケールを表す数値（dB）

エラー
無し

使用例
```
disp:scal:vert?
> 10.0<END>
```