User's
Manual

AQ6150B、AQ6151B
光波長計
リモートコントロール
ユーザーズマニュアル

YOKOGAWA
横河計測株式会社

IM AQ6150B-17JA
初版

はじめに

このたびは、光波長計 AQ6150B、AQ6151B をお買い上げいただきましてありがとうございます。このリモートコントロールユーザーズマニュアルは、次の各インタフェースの機能やコマンドについて説明したものです。

- GP-IB インタフェース
- イーサネットインタフェース
- リモートコマンド

ご使用前にこのマニュアルをよくお読みいただき、正しくお使いください。
お読みになったあとは、ご使用時にすぐにご覧になれるところに、大切に保存してください。ご使用中に操作がわからなくなったときなどにきっとお役に立ちます。
なお、本機器のマニュアルとして、このマニュアルを含め次のものがあります。あわせてお読みください。

マニュアル名	マニュアル No.	内容
AQ6150B、AQ6151B 光波長計 ユーザーズマニュアル	IM AQ6150B-01JA	本機器のリモート機能を除く全機能とその操作方法について説明しています。付属の CD に PDF データが納められています。
AQ6150B、AQ6151B 光波長計 スタートガイド	IM AQ6150B-02JA	冊子で提供しています。本機器の取り扱い上の注意や基本的な操作の説明と、仕様を記載しています。付属の CD に PDF データが納められています。
AQ6150B、AQ6151B 光波長計 リモートコントロール ユーザーズマニュアル	IM AQ6150B-17JA	本書です。本機器の通信インタフェースの機能について、その操作方法を説明しています。付属の CD に PDF データが納められています。
AQ6150B, AQ6151B Optical Wavelength Meter	IM AQ6150B-92Z1	中国向けの文書です。

マニュアル No. の「JA」、「Z1」は言語コードです。

各国や地域の当社営業拠点の連絡先は、次のシートに記載されています。

ドキュメント No.	内容
PIM 113-01Z2	国内海外の連絡先一覧

ご注意

- 本書の内容は、性能・機能の向上などにより、将来、予告なしに変更することがあります。また、実際の表示内容が本書に記載の表示内容と多少異なることがあります。
- 本書の内容に関しては万全を期していますが、万一ご不審の点や誤りなどお気づきのことがありましたら、お手数ですが、当社支社・支店・営業所までご連絡ください。
- 本書の内容の全部または一部を無断で転載、複製することは禁止されています。
- 保証書が付いています。再発行はいたしません。よくお読みいただき、ご理解のうえ大切に保存してください。

商標

- Microsoft、Windows は、米国 Microsoft Corporation の、米国およびその他の国における登録商標または商標です。
- Adobe、Acrobat は、アドビシステムズ社の登録商標または商標です。
- 本文中の各社の登録商標または商標には、®、TM マークは表示していません。
- その他、本文中に使われている会社名、商品名は、各社の登録商標または商標です。

1st Edition: November 2018 (YMI)

履歴

・ 2018 年 11 月　初版発行

このマニュアルで使用している記号

注記

このマニュアルでは、注記を以下のようなシンボルで区別しています。

警　告　取り扱いを誤った場合に、使用者が死亡または重傷を負う危険があるときに、その危険を避けるための注意事項が記載されています。

注　意　取り扱いを誤った場合に、使用者が軽傷を負うか、または物的損害のみが発生する危険があるときに、それを避けるための注意事項が記載されています。

Note　本機器を取り扱ううえで重要な情報が記載されています。

操作説明ページで使用しているシンボル

各章で操作説明をしているページでは、説明内容を区別するために、次のようなシンボル / 表示文字 / 用語を使用しています。

操　作　数字で示す順序で各操作をしてください。ここでは、初めて操作をすることを前提に手順を説明しています。したがって設定内容を変更する場合は、すべての操作を必要としない場合があります。

解　説　操作に関連する設定内容や限定事項について説明しています。

操作説明中の表示文字と用語

操作キーとソフトキー
操作説明のところに記載されている太字の英数字は、操作対象のパネル上の操作キーの文字や、画面に表示されるソフトキー / メニューの文字を示します。

単位

k	「1000」の意味です。使用例：12kg、100kHz
K	「1024」の意味です。使用例：459K バイト (ファイルのデータサイズ)

このマニュアルの利用方法

このマニュアルの構成

このユーザーズマニュアルは、以下に示す第 1 章～第 5 章および付録で構成されています。

第 1 章 リモートコントロール機能
各種通信インタフェースの概要について説明しています。

第 2 章 GP-IB インタフェース
パーソナルコンピュータ (PC) をコントローラとして本機器をコントロールする GP-IB の機能・仕様などについて説明しています。

第 3 章 イーサネットインタフェース
イーサネットインタフェースの機能・仕様などについて説明しています。

第 4 章 ステータスレジスタ
ステータスバイトや各種レジスタ、キューなどについて説明しています。

第 5 章 リモートコマンド
使用できる全コマンドについて 1 つずつ説明しています。

目次

付録

1.1　リモートインタフェース

本機器の制御にはリモートコマンドを使用します。
リモートコマンドは SCPI(Standard Commands for Programmable Instruments) に準拠しています。以下のリモートインタフェースを備えています。

GP-IB(IEEE488.2　2 章参照)
PC などのコントローラにより、本機器をリモートコントロールするときに使用します。コントローラやそのコントローラで制御される他の機器を接続します。

イーサネット (3 章参照)
PC などのコントローラから、ネットワークを使って本機器をリモートコントロールするときに使用します。

1.2　リモート / ローカルの切り替え

ローカル→リモート切り替え時

- GP-IB の場合、ローカル状態のときにコントローラから REN(Remote Enable)、ATN を「True」にしたリスンアドレスを受け取ると、リモート状態になります。
- リモート状態の時は REMOTE ランプが点灯します。
- LOCAL キーおよび POWER スイッチ以外はキーが効かなくなります。
- ローカル状態での設定は、リモート状態になっても保持されます。
- GP-IB の場合、コントローラから LLO(Local Lock Out) のメッセージを受け取るとローカルロックアウト状態になります。ローカルロックアウト状態で LOCAL キーを押してもローカル状態に戻りません。ローカルロックアウト状態を解除してから LOCAL キーを押してください。ローカルロックアウト状態を解除するには、コントローラから REN を「False」にしてください。
- イーサネットの場合、認証が完了してログインするとリモート状態になります。

リモート→ローカル切り替え時

リモート状態のときに LOCAL キーを押すと、ローカル状態になります。ただし、ローカルロックアウト状態の時はローカル状態に戻りません。

- REMOTE ランプが消えます。
- キー操作が可能になります。
- リモート状態での設定は、ローカル状態になっても保持されます。
- GP-IB の場合、コントローラから GTL(Go To Local) メッセージを受け取るか、REN を「False」にしてもローカル状態になります。

1.3　リモートコマンドの送受信について

バッファについて

入力バッファ

本機器の入力バッファは 1 段で、バッファサイズは 4Mbytes です。
バッファサイズを超えるデータを受信した場合は、先頭の 4Mbytes よりも後ろのデータは破棄されます。このとき、先頭の 4Mbytes のデータのうち、最後のコマンドセパレータ以降のコマンドも削除してコマンド処理を行います。

出力バッファ

本機器の出力バッファは 1 段で、バッファサイズは 4Mbytes です。
最新のデータだけを保持します。
(データがバッファに保持されている状態でコマンドを受け付けると、保持していたデータをクリアして新しいデータを持ちます)
複数のトーカコマンドを組み合わせて実行してバッファサイズを超えるトーカデータが発生した場合は、以下の動作を行います。

- スタンダードイベントステータスレジスタのクエリエラービット (QYE) を 1 にセットします。
- 出力バッファをクリアします。
- 出力バッファをオーバーした以降も、受信済みのコマンドの処理を続けます。
  ただし、トーカコマンドにより発生するトーカデータは、出力バッファに格納されません。

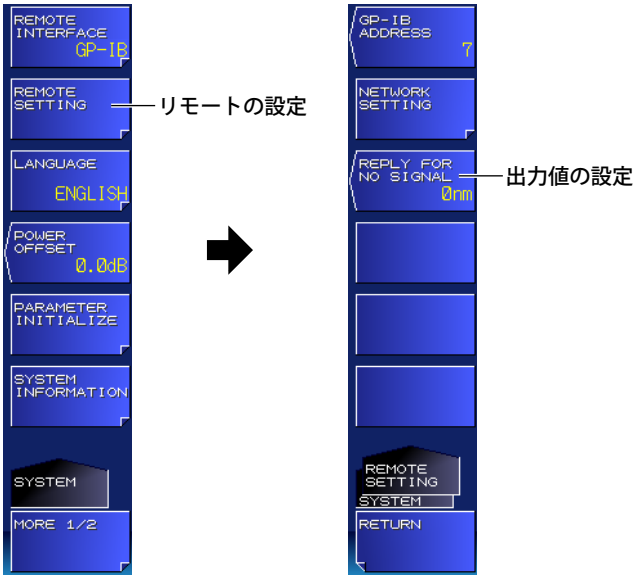
エラーバッファ

本機器のエラーバッファは 10 段です。

1.4　信号を検出していないときの応答について

信号が検出されていない状態で、PC から波長値の出力要求があった場合の出力値を 0nm ～ 300nm の範囲で、任意に設定できます。初期値は 0nm です。

1. **SYSTEM** を押します。
   システム設定メニューが表示されます。
2. **REMOTE SETTING** のソフトキーを押します。
   リモート設定する画面が表示されます。
3. **REPLY FOR NO SIGNAL** のソフトキーを押します。
   出力値を設定する画面が表示されます。
4. 矢印キーまたはテンキーで出力値を入力します。
5. **ENTER** キーを押します。
   ソフトキー上に設定した出力値が表示されます。



Note

本機能は、下記のコマンドに対して有効です。

```
:FETCh[:SCALar]:POWer:{FREQuency|WAVelength|WNUMber}?
:MEASure[:SCALar]:POWer:{FREQuency|WAVelength|WNUMber}?
:READ[:SCALar]:POWer:{FREQuency|WAVelength|WNUMber}?
```

2.1　GP-IB による接続

GP-IB ケーブル

本機器の GP-IB コネクタは、IEEE St'd 488-1978 規格の 24 ピンコネクタです。GP-IB ケーブルは、IEEE St'd 488-1978 に合ったものを使用してください。

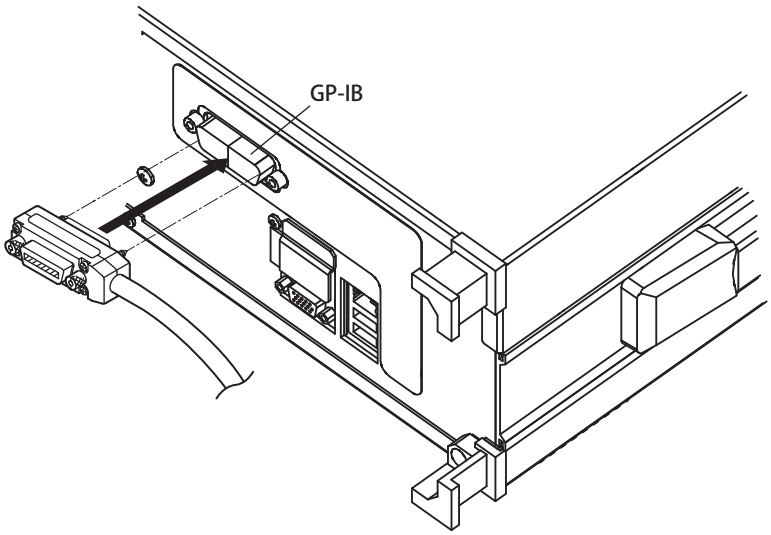
接続方法

GP-IB ポート：　PC と接続して本機器を PC からリモートコントロールできます。

本機器および本機器に接続する機器の電源を OFF にします。
本機器背面にある GP-IB ポートにケーブルを接続します。

注　意

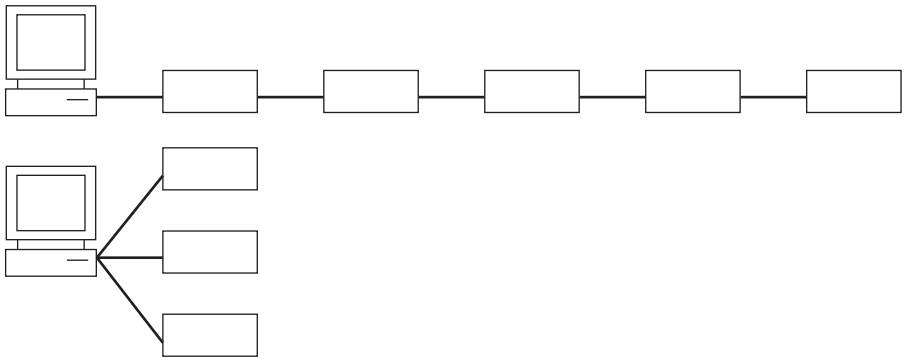
通信ケーブルを接続したり、取り外したりするときは、必ず PC および本機器の電源を OFF にしてください。OFF にしないと、誤動作が生じたり、内部回路を破損することがあります。



接続時の注意

- GP-IB ケーブルのコネクタに付いているねじは、しっかりと固定してください。
- 何本かのケーブルを接続して、複数の機器を接続することができます。ただし、1 つのバス上にコントローラを含め 15 台以上の機器を接続することはできません。
- 複数の機器を接続するときは、それぞれのアドレスを同じに設定することはできません。
- 機器間をつなぐケーブルは 2m 以下のものを使用してください。
- ケーブルの長さは合計で 20m を超えないようにしてください。
- 通信を行っているときは、少なくとも全体の 2/3 以上の機器の電源を ON にしておいてください。

・ 複数の機器を接続するときは、下図に示すようなリニア形またはスター形の結線にしてください。その組み合わせも可能です。ループ形の結線はできません。



2.2 GP-IB インタフェースの機能

GP-IB インタフェースの機能

リスナ機能

- 電源の ON/OFF、通信の設定、その他一部の設定を除き、本機器のキー操作で設定できる内容の設定ができます。
- コントローラからの出力指令で、設定データや測定データなどを受けることができます。
- その他、ステータスレポートに関するコマンドなどを受けることができます。

トーカ機能

- 設定データや測定データなどを出力できます。

Note

- リスンオンリ、トークオンリ、およびコントローラ機能はありません。
- GP-IB インタフェースは、イーサネット通信インタフェースと同時に使用できません。

メッセージターミネータ

本機器では、以下のメッセージターミネータを使用できます。

プログラムメッセージターミネータ

- EOI (End-Or-Identify) 信号のアサート
- LF(改行) 文字
- LF+EOI

ここで、LF を ASCII のラインフィード (0Ah)、CR+LF については CR(0Dh) を wsp として認識するため、結果として CR+LF もメッセージターミネータとして使用できます。

応答メッセージターミネータ

応答メッセージターミネータは LF+EOI を用います。

リモートコマンドの受信

- コマンド受信を完了すると、GP-IB バスを解放します。
- コマンド動作実行中に次のコマンドを受信した場合には、そのコマンドを取り込み、受信バッファに保存したあと、GP-IB バスを解放します。
- 受信バッファにコマンドが存在する場合には、それ以降のコマンドが GP-IB バス上に存在していても、取り込みは行いません。
- 先行コマンドの動作が完了すると、受信バッファのコマンド実行とクリアを行います。そして、次のコマンドがバス上にあれば受信バッファへの取り込み動作を行います。
- 1 つの出力ステートメントに複数のコマンドが含まれている場合は、全てのコマンドを取り込み、記述順に処理を行います。この場合、ステートメント最後のコマンド動作の実行を開始しない限り、次のコマンドを取り込めません。

データの問い合わせ

- 外部コントローラによるデータの問い合わせは、クエリコマンドとコントローラからのデータ出力要求により行います。
- クエリコマンドは、コマンドの後ろに "?" が付いた形式になっています。
- パラメータを持つクエリコマンドの場合には、"?" の後ろに <wsp> ＋ < パラメータ > の形式で指定します。
- 本機器がクエリコマンドを受信した場合、その時点でのクエリコマンドに対する応答を、出力バッファに準備します。
- 出力バッファのデータは、コントローラの入力ステートメントがあるか、もしくは新たなクエリコマンドを受信するまで保持されます。
- 複数のクエリコマンドが、セミコロン ";" により連続指定記述されている場合には、すべてのクエリコマンドに対応した応答を出力バッファに準備します。
  この場合、次に行われるデータ出力要求に対して、準備されているすべてのデータを一括で出力します。

デバイストリガ機能

GET(Group Execute Trigger) 受信時は、シングル測定を行います。

2.3 GP-IB インタフェースの仕様

GP-IB インタフェースの仕様

電気的・機械的仕様： IEEE St'd 488-1978 に準拠
機能的仕様： 下表
プロトコル： IEEE St'd 488.2-1992 に準拠
使用コード： ISO(ASCII) コード
モード： アドレッサブルモード
アドレス設定： SYSTEM メニューの GP-IB の設定画面で、0 ～ 30 のアドレスを設定可能。
リモート状態解除： LOCAL を押すことで、リモート状態の解除可能。ただし、コントローラにより Local Lockout されているときは無効。

機能的仕様

機能	サブセット名	内容
ソースハンドシェーク	SH1	送信ハンドシェークの全機能あり
アクセプタハンドシェーク	AH1	受信ハンドシェークの全機能あり
トーカ	T6	基本トーカ機能、シリアルポール、M L A(My Listen Address) によるトーカ解除機能あり、トークオンリ機能なし
リスナ	L4	基本リスナ機能、MTA(My Talk Address) によるリスナ解除機能あり、リスンオンリ機能なし
サービスリクエスト	SR1	サービスリクエストの全機能あり
リモートローカル	RL1	リモート / ローカルの全機能あり
パラレルポール	PP0	パラレルポール機能なし
デバイスクリア	DC1	デバイスクリアの全機能あり 出力バッファのクリアあり 入力バッファのクリア (未処理コマンドのクリア) あり エラーバッファのクリアあり STB、ESR のクリアあり
デバイストリガ	DT0	デバイストリガ機能あり
コントローラ	C0	コントローラ機能なし
電気特性	E1	オープンコレクタ

2.4 GP-IB アドレスの設定

操　作

通信インタフェースの選択

1. **SYSTEM** を押します。
   システム設定メニューが表示されます。
2. **REMOTE INTERFACE** のソフトキーを押します。
   リモートインタフェース設定メニューが表示されます。
3. **GP-IB** のソフトキーを押します。
   設定メニューの表示が 1 つ前に戻り、ソフトキー上に GP-IB が表示されます。

アドレスの設定

4. **REMOTE SETTING** のソフトキーを押します。
   リモート設定する画面が表示されます。
5. **GP-IB ADDRESS** のソフトキーを押します。
   GP-IB のアドレス値を設定する画面が表示されます。

6. 矢印キーまたはテンキーでアドレス値を入力します。

7. **ENTER** キーを押します。
ソフトキー上に設定したアドレスの値が表示されます。

解　説

本機器のキー操作で設定できる内容をコントローラで設定するときや、コントローラに設定データや測定データを出力するときは、下記の設定をします。

GP-IB アドレスの設定

アドレッサブルモードのときの、本機器のアドレスを次の範囲で設定します。
0 ～ 30
GP-IB で接続できる各装置は、GP-IB システム内で固有のアドレスを持ちます。このアドレスによって他の装置と識別されます。したがって、本機器を PC などに接続するときは、本機器のアドレスを他の機器と重ならないように設定する必要があります。

Note
コントローラが他のデバイスも含めて GP-IB を使用中は、アドレスを変更しないでください。

2.5　インタフェースメッセージに対する応答

インタフェースメッセージに対する応答

ユニラインメッセージに対する応答

IFC(Interface Clear)

トーカ、リスナを解除します。データ出力中のときは出力を中止します。

REN(Remote Enable)

リモート状態 / ローカル状態を切り替えます。

IDY(Identify) はサポートしていません。

マルチラインメッセージ (アドレスコマンド) に対する応答

GTL(Go To Local)

ローカル状態へ移行します。

SDC(Selected Device Clear)

- 受信中のプログラムメッセージ (コマンド) と、出力キューをクリアします。
- 実行中の *OPC、*OPC? は無効になります。
- *WAI は直ちに終了します。

PPC(Parallel Poll Configure)、TCT(Take Control) はサポートしていません。

マルチラインメッセージ (ユニバーサルコマンド) に対する応答

LLO(Local Lockout)

フロントパネルの LOCAL キーの操作を無効にし、ローカル状態への移行を禁止します。

DCL(Device Clear)

SDC と同じ動作をします。

SPE(Serial Poll Enable)

バス上のすべての機器のトーカ機能をシリアルポールモードにします。コントローラは各機器を順番にポーリングします。

SPD(Serial Poll Disable)

バス上のすべての機器のトーカ機能のシリアルポールモードを解除します。

PPU(Parallel Poll Unconfigure) はサポートしていません。

インタフェースメッセージとは

インタフェースメッセージは、インタフェースコマンドまたはバスコマンドとも呼ばれ、コントローラから発せられるコマンドのことです。次のような分類になっています。

ユニラインメッセージ

1 本の管理ラインを経由してメッセージを送ります。次の 3 種類があります。

IFC(Interface Clear)
REN(Remote Enable)
IDY(Identify)

マルチラインメッセージ

8 本のデータラインを経由してメッセージを送ります。次のように分類されます。

アドレスコマンド

機器がリスナあるいはトーカに指定されているときに有効なコマンドです。次の 5 種類があります。

リスナに指定している機器に有効なコマンド

GTL(Go To Local)
SDC(Selected Device Clear)
PPC(Parallel Poll Configure)
GET(Group Execute Trigger)

トーカに指定している機器に有効なコマンド

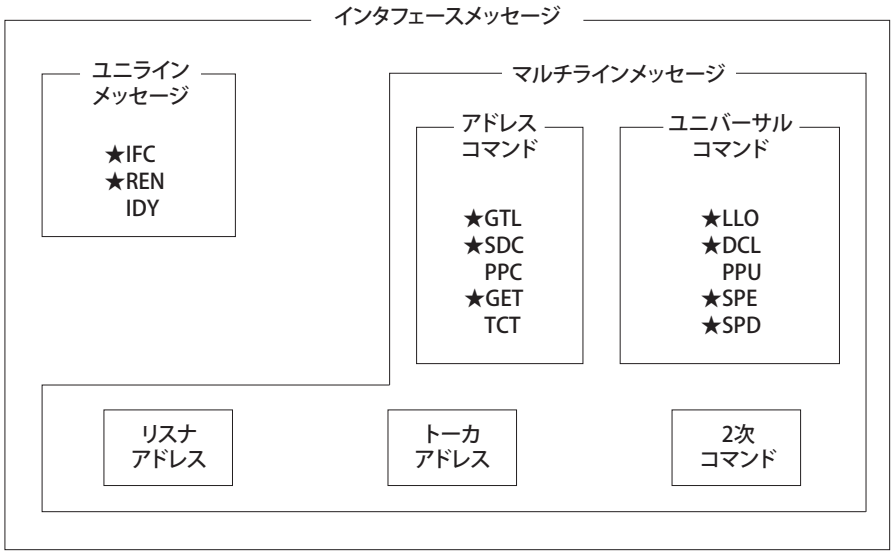
TCT(Take Control)

ユニバーサルコマンド

リスナ・トーカの指定の有無に関わらず、すべての機器に有効です。次の 5 種類があります。

LLO(Local Lockout)
DCL(Device Clear)
PPU(Parallel Poll Unconfigure)
SPE(Serial Poll Enable)
SPD(Serial Poll Disable)

その他、インタフェースメッセージとして、リスナアドレス、トーカアドレス、2 次コマンドがあります。



★印は本機器でサポートしているインタフェースメッセージです。

Note

SDC と DCL の違い

マルチラインメッセージのうち、SDC はトーカ・リスナの指定が必要なアドレスコマンド、DCL はトーカ・リスナの指定が不要なユニバーサルコマンドです。したがって、SDC はある特定の機器を対象にしますが、DCL はバス上のすべての機器を対象にします。

2.6　サンプルプログラム

GP-IB ポートを使用して本機器をリモートコントロールする例を示します。
対象モデル　：PC-AT 機
対象言語　　：Visual Basic 2008
GP-IB ボード：National Instruments 社製 GP-IB ボード
使用コンポーネント：NationalInstruments.Common
NationalInstruments.NI4882
.NET Framework 3.5

サンプルプログラム 1

GP-IB にて Single 測定を 1 回行い、結果 (波長情報、Power 情報、FP-LD 解析結果) を画面に出力するサンプルプログラム (仕様コード -MW: マルチ波長タイプの例です。)

ソースコード

```
Imports System
Imports System.IO
Imports NationalInstruments.NI4882

Module GpibSingleMeasure
'
' GP-IB にて Single 測定を 1 回行い、結果 ( 波長情報、Power 情報、FP-LD 解析結果 ) を画面に出力するサンプルプロ
' グラム
'
  Sub Main()
    Try
      Dim GpibDevice As Device
      Dim wlmAddr As Integer
      Dim replyString As String
      Dim wavArray As Double()
      Dim powArray As Double()
      Dim fwhm, ctrWl, totalPwr, sigma As Double
      Dim maxPeakPower, maxPeakWl As Double

      '=====================================================
      ' 波長計の情報
      '=====================================================
      wlmAddr = 7                                          ' 波長計の GP-IB アドレス
      GpibDevice = New Device(0, wlmAddr)                  'GP-IB の Open

      '=====================================================
      ' 波長計の測定条件設定
      '=====================================================
      Call GpibWrite("*RST", GpibDevice)                   ' 本機器のリセット
      Call GpibWrite(":CALC2:PTHR:MODE REL", GpibDevice)
                                                           ' ピーク検出しきい値設定を相対値モード
                                                           ' に設定
      Call GpibWrite(":CALC2:PTHR 15", GpibDevice)
                                                           ' ピーク検出しきい値を 15db に設定
      Call GpibWrite(":UNIT:WL NM", GpibDevice)            ' 波長単位を nm に設定
      Call GpibWrite(":UNIT:POW DBM", GpibDevice)          'Power 単位を dBm に設定
      Call GpibWrite(":DISP:WIND2:STAT ON", GpibDevice)    ' グラフ表示を有効化

      '=====================================================
      ' 測定の実行、データの取得
      '=====================================================
      ' 測定は READ コマンドにより実行し、データを取得する。
      ' さらに Power 情報は FETC コマンドで測定済みのデータを取得する。
      Call GpibWrite(":READ:ARR:POW:WAV?", GpibDevice)     'Single 測定の実行と波長データの取得
      replyString = GpibRead(GpibDevice)
      Call SplitArrayData(replyString, wavArray)           ' 波長情報を配列に格納

      Call GpibWrite(":FETC:ARR:POW?", GpibDevice)         ' 測定済み Power の取得
```

```
    replyString = GpibRead(GpibDevice)
    Call SplitArrayData(replyString, powArray)            'Power 情報を配列に格納

    '=====================================================
    ' 結果の表示 ( 波長、Power 情報 )
    '=====================================================
    Console.WriteLine("No.|Wavelength(m)|Power(dBm)")
    For idx As Integer = 1 To wavArray.Length
      Console.WriteLine((idx).ToString + "|" + wavArray(idx - 1).ToString() + "|" + _
                        powArray(idx - 1).ToString())
    Next

    '=====================================================
    ' 最大 Power ピーク情報の取得
    '=====================================================
    Call GpibWrite(":FETC:POW? MAX", GpibDevice)          ' 最大 Power のピークを指定して
                                                          'Power を取得
    replyString = GpibRead(GpibDevice)
    maxPeakPower = Convert.ToDouble(replyString)
    Call GpibWrite(":FETC:POW:WAV?", GpibDevice)          ':FETC:POW? MAX で指定された
                                                          ' ピークの波長を取得
    replyString = GpibRead(GpibDevice)
    maxPeakWl = Convert.ToDouble(replyString)
    Console.WriteLine("Highest Peak Power     :" + maxPeakPower.ToString + " dBm")
    Console.WriteLine("Highest Peak Wavelength:" + maxPeakWl.ToString + " nm")
    '=====================================================
    '  FP-LD 解析結果の取得
    '=====================================================
    Call GpibWrite(":CALC3:FPER ON", GpibDevice)          'FP-LD 解析の有効化
    Call GpibWrite(":CALC3:FPER:FWHM?", GpibDevice)       'FWHM の取得
    replyString = GpibRead(GpibDevice)
    fwhm = Convert.ToDouble(replyString)
    Call GpibWrite(":CALC3:FPER:MEAN?", GpibDevice)       'Center WL の取得
    replyString = GpibRead(GpibDevice)
    ctrWl = Convert.ToDouble(replyString)
    Call GpibWrite(":CALC3:FPER:POW?", GpibDevice)        'Total Power の取得
    replyString = GpibRead(GpibDevice)
    totalPwr = Convert.ToDouble(replyString)
    Call GpibWrite(":CALC3:FPER:SIGM?", GpibDevice)       'σ の取得
    replyString = GpibRead(GpibDevice)
    sigma = Convert.ToDouble(replyString)
    ' 結果の表示 (FP-LD 解析結果 )
    Console.WriteLine("====FP-LD Analysis====")
    Console.WriteLine("FWHM             : " + (fwhm * 1000000000).ToString + "nm")
    Console.WriteLine("Sigma            : " + (sigma * 1000000000).ToString + "nm")
    Console.WriteLine("CTR WL           : " + (ctrWl * 1000000000).ToString + "nm")
    Console.WriteLine("TOTAL PWR        : " + totalPwr.ToString + "dBm")

    '=====================================================
    ' 内部メモリへのデータ保存
    '=====================================================
    ' 画面イメージと結果データを内部メモリに保存する。
     Call GpibWrite(":MMEM:STOR SIM2,""\WLM_IMAGE"",INT", GpibDevice)
     Call GpibWrite(":MMEM:STOR TABL,""\WLM_TABLE"",INT", GpibDevice)

    '=====================================================
    ' 内部メモリに保存したデータを PC に転送
    '=====================================================
    Call GpibWrite(":MMEM:DATA? ""\WLM_IMAGE.BMP"",INT", GpibDevice)
    GpibReadBlockData2File(GpibDevice, "WLM_IMAGE.BMP")
    Call GpibWrite(":MMEM:DATA? ""\WLM_TABLE.CSV"",INT", GpibDevice)
    GpibReadBlockData2File(GpibDevice, "WLM_TABLE.CSV")
    Console.ReadLine()                                    ' 終了の Enter キー入力待ち
    Catch ex As Exception                                 ' 例外 ( エラー ) 処理
    Console.WriteLine(ex.Message)                         ' 発生したエラーのメッセージを表示
    Console.ReadLine()                                    ' 終了の Enter キー入力待ち
  End Try
End Sub
```

```
'=========================================================
'GP-IB に文字列を送信する関数
'=========================================================
Sub GpibWrite(ByVal commandStr As String, ByRef gpib As Device)
    gpib.Write(commandStr)                                  ' データの送信
End Sub

'=========================================================
'GP-IB からデータを 1 行読み込む関数
'=========================================================
Function GpibRead(ByRef gpib As Device) As String
  GpibRead = gpib.ReadString()                              ' データの受信
  Exit Function
End Function

'=========================================================
' ブロックデータを読み込んでファイルに保存する関数
'=========================================================
Function GpibReadBlockData2File(ByRef gpib As Device, ByVal filename As String) As Integer
  Dim headerLen As Integer
  Dim dataLen As Integer
  Dim dataByte As Byte()
  Dim file As New FileStream(filename, FileMode.Create, FileAccess.Write)

  If String.Compare(gpib.ReadString(1), "#") <> 0 Then     '1 文字目の取得
    GpibReadBlockData2File = -1                             '1 文字目は「# 以外」はエラー
    Exit Function
  End If
  headerLen = Integer.Parse(gpib.ReadString(1))             ' データ長情報が入っているエリアの
                                                            ' 大きさ
  dataLen = Integer.Parse(gpib.ReadString(headerLen))       ' データ長情報の取得

  While dataLen > 1024
    dataByte = gpib.ReadByteArray(1024)                     ' データを 1024byte づつ読み込み
    file.Write(dataByte, 0, dataByte.Length)                ' 取得したデータをファイルへ書き込む
    dataLen = dataLen - dataByte.Length
  End While

  dataByte = gpib.ReadByteArray(dataLen)                    ' 最後のデータを取得
  file.Write(dataByte, 0, dataByte.Length)                  ' 取得したデータをファイルへ書き込む

  file.Close()
  GpibReadBlockData2File = 0
End Function

'=========================================================
'READ/FETC/MEAS の結果を配列に分割
'=========================================================
Sub SplitArrayData(ByVal dataString As String, ByRef dataArray As Double())
  Dim peakNum As Integer
  Dim arrayDataStr As String() = dataString.Split(","c)    '「,」区切りでデータを分割
  peakNum = Integer.Parse(arrayDataStr(0))                  ' データ数の取得
  dataArray = New Double(peakNum - 1) {}
  For idx As Integer = 1 To arrayDataStr.Length - 1
    dataArray(idx - 1) = Convert.ToDouble _
                         (arrayDataStr(idx))                ' データ数分データを読み込む
  Next
 End Sub
End Module
```

実行例

```
No.|Wavelength(m) |Power(dBm)
  1|1.30678822E-06|-14.3279541
  2|1.30756963E-06|-9.42082105
  3|1.30835228E-06|-2.23592107
  4|1.30913555E-06|-3.93065804
  5|1.30991986E-06|-13.5578301
Highest Peak Power     :-2.23592107 dBm
Highest Peak Wavelength:1.30835228E-06 nm
====FP-LD Analysis====
FWHM             : 1.47415158nm
Sigma            : 0.625966702nm
CTR WL           : 1308.55169nm
TOTAL PWR        : 0.782282871dBm
```

サンプルプログラム 2

GP-IB にて Single 測定を 1 回行い、結果 (波長情報、Power 情報) を画面に出力するサンプルプログラム
(仕様コード -SW: シングル波長タイプの例です。)

ソースコード

```
Imports System
Imports System.IO
Imports NationalInstruments.NI4882

Module GpibSingleMeasure
'
' GP-IB にて Single 測定を 1 回行い、結果 ( 波長情報、Power 情報 ) を画面に出力するサンプルプログラム
'
  Sub Main()
    Try
      Dim GpibDevice As Device
      Dim wlmAddr As Integer
      Dim replyString As String
      Dim wavelength As Double
      Dim power As Double

      '=====================================================
      ' 波長計の情報
      '=====================================================
      wlmAddr = 7                                          ' 波長計の GP-IB アドレス
      GpibDevice = New Device(0, wlmAddr)                  'GP-IB の Open

      '=====================================================
      ' 波長計の測定条件設定
      '=====================================================
      Call GpibWrite("*RST", GpibDevice)                   ' 本機器のリセット
      Call GpibWrite(":UNIT:WL NM", GpibDevice)            ' 波長単位を nm に設定
      Call GpibWrite(":UNIT:POW DBM", GpibDevice)          'Power 単位を dBm に設定

      '=====================================================
      ' 測定の実行、データの取得
      '=====================================================
      ' 測定は READ コマンドにより実行し、データを取得する。
      ' さらに Power 情報は FETC コマンドで測定済みのデータを取得する。
      Call GpibWrite(":READ:POW:WAV?", GpibDevice)         'Single 測定の実行と波長データの取得
      replyString = GpibRead(GpibDevice)
      wavelength = Convert.ToDouble(replyString)           ' 応答文字列を数値に変換

      Call GpibWrite(":FETC:POW?", GpibDevice)             ' 測定済み Power の取得
      replyString = GpibRead(GpibDevice)
      power = Convert.ToDouble(replyString)                ' 応答文字列を数値に変換
```

```
    '=====================================================
    ' 結果の表示 ( 波長、Power 情報 )
    '=====================================================
    Console.WriteLine("Wavelength(m):" + wavelength.ToString())
    Console.WriteLine("Power(dBm)   :" + power.ToString())

    '=====================================================
    ' 内部メモリへのデータ保存
    '=====================================================
    ' 画面イメージと結果データを内部メモリに保存する。
     Call GpibWrite(":MMEM:STOR SIM2,""\WLM_IMAGE"",INT", GpibDevice)
     Call GpibWrite(":MMEM:STOR TABL,""\WLM_TABLE"",INT", GpibDevice)

    '=====================================================
    ' 内部メモリに保存したデータを PC に転送
    '=====================================================
    Call GpibWrite(":MMEM:DATA? ""\WLM_IMAGE.BMP"",INT", GpibDevice)
    GpibReadBlockData2File(GpibDevice, "WLM_IMAGE.BMP")
    Call GpibWrite(":MMEM:DATA? ""\WLM_TABLE.CSV"",INT", GpibDevice)
    GpibReadBlockData2File(GpibDevice, "WLM_TABLE.CSV")
    Console.ReadLine()                                      ' 終了の Enter キー入力待ち
  Catch ex As Exception                                     ' 例外 ( エラー ) 処理
    Console.WriteLine(ex.Message)                           ' 発生したエラーのメッセージを表示
    Console.ReadLine()                                      ' 終了の Enter キー入力待ち
  End Try
End Sub

'=========================================================
'GP-IB に文字列を送信する関数
'=========================================================
Sub GpibWrite(ByVal commandStr As String, ByRef gpib As Device)
    gpib.Write(commandStr)                                  ' データの送信
End Sub

'=========================================================
'GP-IB からデータを 1 行読み込む関数
'=========================================================
Function GpibRead(ByRef gpib As Device) As String
  GpibRead = gpib.ReadString()                              ' データの受信
  Exit Function
End Function
```

```
'=========================================================
' ブロックデータを読み込んでファイルに保存する関数
'=========================================================
Function GpibReadBlockData2File(ByRef gpib As Device, ByVal filename As String) As Integer
  Dim headerLen As Integer
  Dim dataLen As Integer
  Dim dataByte As Byte()
  Dim file As New FileStream(filename, FileMode.Create, FileAccess.Write)

  If String.Compare(gpib.ReadString(1), "#") <> 0 Then     '1 文字目の取得
    GpibReadBlockData2File = -1                            '1 文字目は「# 以外」はエラー
    Exit Function
  End If
  headerLen = Integer.Parse(gpib.ReadString(1))            ' データ長情報が入っているエリアの
                                                           ' 大きさ
  dataLen = Integer.Parse(gpib.ReadString(headerLen))      ' データ長情報の取得

  While dataLen > 1024
    dataByte = gpib.ReadByteArray(1024)                    ' データを 1024byte づつ読み込み
    file.Write(dataByte, 0, dataByte.Length)               ' 取得したデータをファイルへ書き込む
    dataLen = dataLen - dataByte.Length
  End While

  dataByte = gpib.ReadByteArray(dataLen)                   ' 最後のデータを取得
  file.Write(dataByte, 0, dataByte.Length)                 ' 取得したデータをファイルへ書き込む

  file.Close()
  GpibReadBlockData2File = 0
End Function
End Module
```

実行例

```
Wavelength(m): 1.55252398E-06
Power(dBm)   : 3.43060397
```

サンプルプログラム 3

GP-IB にて Drift 解析を行うサンプルプログラム (仕様コード -MW: マルチ波長タイプの例です。)

ソースコード

```
Imports System
Imports NationalInstruments.NI4882

Module GpibDriftMeasure
'
'GP-IB にて Drift 解析を行うサンプルプログラム
'
  Sub Main()
    Try
      Dim GpibDevice As Device
      Dim wlmAddr As Integer
      Dim replyString As String
      Dim peakNum As Integer
      Dim refPowData, refWavData As Double()
      Dim maxPowData, maxWavData As Double()
      Dim minPowData, minWavData As Double()
      Dim dropInfo As Double()

      '=====================================================
      ' 波長計の情報
      '=====================================================
      wlmAddr = 7                                          ' 波長計の GP-IB アドレス
      GpibDevice = New Device(0, wlmAddr)                  'GP-IB の Open

      '=====================================================
      ' 波長計の測定条件設定
      '=====================================================
      Call GpibWrite("*RST", GpibDevice)                   ' 本機器のリセット
      Call GpibWrite(":CALC2:PTHR:MODE REL", GpibDevice)
                                                           ' しきい値設定を相対値モードに
      Call GpibWrite(":CALC2:PTHR 15", GpibDevice)         ' しきい値を 15db に設定
      Call GpibWrite(":UNIT:WL NM", GpibDevice)            ' 波長単位を nm に設定
      Call GpibWrite(":UNIT:POW DBM", GpibDevice)          'Power 単位を dBm に設定

      ' ドリフト測定のリファレンスとするため Single 測定を行う。
      Call GpibWrite(":INIT;*OPC?", GpibDevice)
                                                           'Single 測定の実行と測定完了待ち
      GpibRead(GpibDevice)                                 ' 測定完了待ち (*OPC?) の応答を読む
      Call GpibWrite(":CALC3:DRIF ON", GpibDevice)         'DRIFT 解析の ON
      '=====================================================
      ' 測定の実行
      '=====================================================
      Call GpibWrite(":INIT:CONT ON", GpibDevice)          'Repeat 測定の開始

      For count As Integer = 1 To 60                       '1 分間待つ
        Threading.Thread.Sleep(1000)
        Console.Write(".")
      Next
      Console.WriteLine("")
      Call GpibWrite(":INIT:CONT OFF", GpibDevice)         'Repeat 測定の停止

      '=====================================================
      ' 測定結果の取得
      '=====================================================
      Call GpibWrite(":CALC3:POIN?", GpibDevice)           ' データ数の取得
      replyString = GpibRead(GpibDevice)
      peakNum = Integer.Parse(replyString)
      refPowData = New Double(peakNum - 1) {}
      refWavData = New Double(peakNum - 1) {}
      maxPowData = New Double(peakNum - 1) {}
      maxWavData = New Double(peakNum - 1) {}
      minPowData = New Double(peakNum - 1) {}
      minWavData = New Double(peakNum - 1) {}
      dropInfo = New Double(peakNum - 1) {}
```

```
' 結果の取得 ( リファレンス値 )
Call GpibWrite(":CALC3:DRIF:REF ON", GpibDevice)
Call GpibWrite(":CALC3:DATA? POW", GpibDevice)          'Ref Power の取得
replyString = GpibRead(GpibDevice)
SplitArrayData(replyString, refPowData)
Call GpibWrite(":CALC3:DATA? WAV", GpibDevice)          'Ref 波長の取得
replyString = GpibRead(GpibDevice)
SplitArrayData(replyString, refWavData)
' 結果の取得 (MAX 値 )
Call GpibWrite(":CALC3:DRIF:PRES", GpibDevice)
Call GpibWrite(":CALC3:DRIF:MAX ON", GpibDevice)
Call GpibWrite(":CALC3:DATA? POW", GpibDevice)          'MAX Power の取得
replyString = GpibRead(GpibDevice)
SplitArrayData(replyString, maxPowData)
Call GpibWrite(":CALC3:DATA? WAV", GpibDevice)          'MAX 波長の取得
replyString = GpibRead(GpibDevice)
SplitArrayData(replyString, maxWavData)
' 結果の取得 (MIN 値 )
Call GpibWrite(":CALC3:DRIF:PRES", GpibDevice)
Call GpibWrite(":CALC3:DRIF:MIN ON", GpibDevice)
Call GpibWrite(":CALC3:DATA? POW", GpibDevice)          'MIN Power の取得
replyString = GpibRead(GpibDevice)
SplitArrayData(replyString, minPowData)
Call GpibWrite(":CALC3:DATA? WAV", GpibDevice)          'MIN 波長の取得
replyString = GpibRead(GpibDevice)
SplitArrayData(replyString, minWavData)
'Drop 情報の取得
Call GpibWrite(":CALC3:DATA? DROP", GpibDevice)
replyString = GpibRead(GpibDevice)
SplitArrayData(replyString, dropInfo)

GpibDevice.Dispose()                                    'GP-IB の Close

'=======================================================
' 測定結果の表示
'=======================================================
Console.Write("No.          |")                          ' ピーク番号の表示
For idx As Integer = 0 To peakNum - 1
  Console.Write((idx + 1).ToString() + "            |")
Next
Console.WriteLine()
Console.Write("REF WL       |")                          ' リファレンス波長の表示
For idx As Integer = 0 To peakNum - 1
  Console.Write(refWavData(idx).ToString() + " | ")
Next
Console.WriteLine()
Console.Write("REF POWER    |")                          ' リファレンス Power の表示
For idx As Integer = 0 To peakNum - 1
  Console.Write(refPowData(idx).ToString() + " | ")
Next
Console.WriteLine()
Console.Write("MAX WL       |")                          ' 最大波長の表示
For idx As Integer = 0 To peakNum - 1
  If dropInfo(idx) <> 0 Then
    Console.Write("---------- | ")
  Else
    Console.Write(maxWavData(idx).ToString() + " | ")
  End If
Next
Console.WriteLine()
Console.Write("MAX POWER    |")                          ' 最大 Power の表示
For idx As Integer = 0 To peakNum - 1
  If dropInfo(idx) <> 0 Then
    Console.Write("---------- | ")
  Else
    Console.Write(maxPowData(idx).ToString() + " | ")
  End If
Next
```

```
      Console.WriteLine()
      Console.Write("MIN WL      |")                          ' 最小波長の表示
      For idx As Integer = 0 To peakNum - 1
        If dropInfo(idx) <> 0 Then
          Console.Write("---------- | ")
        Else
          Console.Write(minWavData(idx).ToString() + " | ")
        End If
      Next
      Console.WriteLine()
      Console.Write("MIN POWER   |")                          ' 最小 Power の表示
      For idx As Integer = 0 To peakNum - 1
        If dropInfo(idx) <> 0 Then
          Console.Write("---------- | ")
        Else
          Console.Write(minPowData(idx).ToString() + " | ")
        End If
      Next
      Console.WriteLine()
      Console.ReadLine()
    Catch ex As Exception                                     ' 例外（エラー）処理
      Console.WriteLine(ex.Message)                           ' 発生したエラーのメッセージを表示
      Console.ReadLine()                                      ' 終了の Enter キー入力待ち
    End Try
  End Sub

  '=========================================================
  'GP-IB に文字列を送信する関数
  '=========================================================
  Sub GpibWrite(ByVal commandStr As String, ByRef gpib As Device)
    gpib.Write(commandStr)                                    ' データの送信
  End Sub

  '=========================================================
  'GP-IB からデータを 1 行読み込む関数
  '=========================================================
  Function GpibRead(ByRef gpib As Device) As String
    GpibRead = gpib.ReadString()                              ' データの受信
    Exit Function
  End Function
  '=========================================================
  'CALC3 の結果を配列に分割する関数
  '=========================================================
  Sub SplitArrayData(ByVal dataString As String, ByRef dataArray As Double())
    Dim peakNum As Integer
    Dim arrayDataStr As String() = dataString.Split(","c)    '「,」区切りで文字列を分割する。
    peakNum = arrayDataStr.Length
    For idx As Integer = 0 To arrayDataStr.Length - 1
      dataArray(idx) = Convert.ToDouble(arrayDataStr(idx))   ' 分割した文字列を数値に変換
    Next
  End Sub
End Module
```

実行例

```
No.       |1              |2              |3              |4              |5              |
REF WL    | 1.30678832E-06| 1.30756981E-06| 1.30835238E-06| 1.30913541E-06| 1.30991969E-06|
REF POWER | -13.4899875   | -9.04694537   | -2.9512995    | -3.29214313   | -13.1556519   |
MAX WL    | --------------| 1.30757036E-06| 1.3083528E-06 | 1.30913604E-06| --------------|
MAX POWER | --------------| -8.81158076   | -0.665845116  | -3.21870974   | --------------|
MIN WL    | --------------| 1.30756953E-06| 1.30835221E-06| 1.30913538E-06| --------------|
MIN POWER | --------------| -10.2276251   | -3.02598662   | -6.67785905   | --------------|
```

サンプルプログラム 4

GP-IB にて Drift 解析を行うサンプルプログラム (仕様コード -SW: シングル波長タイプの例です。)

ソースコード

```
Imports System
Imports NationalInstruments.NI4882

Module GpibDriftMeasure
'
'GP-IB にて Drift 解析を行うサンプルプログラム
'
  Sub Main()
    Try
      Dim GpibDevice As Device
      Dim wlmAddr As Integer
      Dim replyString As String
      Dim peakNum As Integer
      Dim refPowData, refWavData As Double()
      Dim maxPowData, maxWavData As Double()
      Dim minPowData, minWavData As Double()
      Dim dropInfo As Double()

      '=====================================================
      ' 波長計の情報
      '=====================================================
      wlmAddr = 7                                          ' 波長計の GP-IB アドレス
      GpibDevice = New Device(0, wlmAddr)                  'GP-IB の Open

      '=====================================================
      ' 波長計の測定条件設定
      '=====================================================
      Call GpibWrite("*RST", GpibDevice)                   ' 本機器のリセット
      Call GpibWrite(":UNIT:WL NM", GpibDevice)            ' 波長単位を nm に設定
      Call GpibWrite(":UNIT:POW DBM", GpibDevice)          'Power 単位を dBm に設定

      ' ドリフト測定のリファレンスとするため Single 測定を行う。
      Call GpibWrite(":INIT;*OPC?", GpibDevice)
                                                           'Single 測定の実行と測定完了待ち
      GpibRead(GpibDevice)                                 ' 測定完了待ち (*OPC?) の応答を読む
      Call GpibWrite(":CALC3:DRIF ON", GpibDevice)         'DRIFT 解析の ON

      '=====================================================
      ' 測定の実行
      '=====================================================
      Call GpibWrite(":INIT:CONT ON", GpibDevice)          'Repeat 測定の開始

      For count As Integer = 1 To 60                       '1 分間待つ
        Threading.Thread.Sleep(1000)
        Console.Write(".")
      Next
      Console.WriteLine("")
      Call GpibWrite(":INIT:CONT OFF", GpibDevice)         'Repeat 測定の停止
```

```
'=====================================================
' 測定結果の取得
'=====================================================
Call GpibWrite(":CALC3:POIN?", GpibDevice)              ' データ数の取得
replyString = GpibRead(GpibDevice)
peakNum = Integer.Parse(replyString)
refPowData = New Double(peakNum - 1) {}
refWavData = New Double(peakNum - 1) {}
maxPowData = New Double(peakNum - 1) {}
maxWavData = New Double(peakNum - 1) {}
minPowData = New Double(peakNum - 1) {}
minWavData = New Double(peakNum - 1) {}
dropInfo = New Double(peakNum - 1) {}

' 結果の取得 ( リファレンス値 )
Call GpibWrite(":CALC3:DRIF:REF ON", GpibDevice)
Call GpibWrite(":CALC3:DATA? POW", GpibDevice)          'Ref Power の取得
replyString = GpibRead(GpibDevice)
SplitArrayData(replyString, refPowData)
Call GpibWrite(":CALC3:DATA? WAV", GpibDevice)          'Ref 波長の取得
replyString = GpibRead(GpibDevice)
SplitArrayData(replyString, refWavData)
' 結果の取得 (MAX 値 )
Call GpibWrite(":CALC3:DRIF:PRES", GpibDevice)
Call GpibWrite(":CALC3:DRIF:MAX ON", GpibDevice)
Call GpibWrite(":CALC3:DATA? POW", GpibDevice)          'MAX Power の取得
replyString = GpibRead(GpibDevice)
SplitArrayData(replyString, maxPowData)
Call GpibWrite(":CALC3:DATA? WAV", GpibDevice)          'MAX 波長の取得
replyString = GpibRead(GpibDevice)
SplitArrayData(replyString, maxWavData)
' 結果の取得 (MIN 値 )
Call GpibWrite(":CALC3:DRIF:PRES", GpibDevice)
Call GpibWrite(":CALC3:DRIF:MIN ON", GpibDevice)
Call GpibWrite(":CALC3:DATA? POW", GpibDevice)          'MIN Power の取得
replyString = GpibRead(GpibDevice)
SplitArrayData(replyString, minPowData)
Call GpibWrite(":CALC3:DATA? WAV", GpibDevice)          'MIN 波長の取得
replyString = GpibRead(GpibDevice)
SplitArrayData(replyString, minWavData)
'Drop 情報の取得
Call GpibWrite(":CALC3:DATA? DROP", GpibDevice)
replyString = GpibRead(GpibDevice)
SplitArrayData(replyString, dropInfo)

GpibDevice.Dispose()                                    'GP-IB の Close
```

```
    '=====================================================
    ' 測定結果の表示
    '=====================================================
    Console.Write("No.          |")                          ' ピーク番号の表示
    For idx As Integer = 0 To peakNum - 1
      Console.Write((idx + 1).ToString() + "             |")
    Next
    Console.WriteLine()
    Console.Write("REF WL       |")                          ' リファレンス波長の表示
    For idx As Integer = 0 To peakNum - 1
      Console.Write(refWavData(idx).ToString() + " | ")
    Next
    Console.WriteLine()
    Console.Write("REF POWER    |")                          ' リファレンス Power の表示
    For idx As Integer = 0 To peakNum - 1
      Console.Write(refPowData(idx).ToString() + " | ")
    Next
    Console.WriteLine()
    Console.Write("MAX WL       |")                          ' 最大波長の表示
    For idx As Integer = 0 To peakNum - 1
      If dropInfo(idx) <> 0 Then
        Console.Write("---------- | ")
      Else
        Console.Write(maxWavData(idx).ToString() + " | ")
      End If
    Next
    Console.WriteLine()
    Console.Write("MAX POWER    |")                          ' 最大 Power の表示
    For idx As Integer = 0 To peakNum - 1
      If dropInfo(idx) <> 0 Then
        Console.Write("---------- | ")
      Else
        Console.Write(maxPowData(idx).ToString() + " | ")
      End If
    Next

    Console.WriteLine()
    Console.Write("MIN WL       |")                          ' 最小波長の表示
    For idx As Integer = 0 To peakNum - 1
      If dropInfo(idx) <> 0 Then
        Console.Write("---------- | ")
      Else
        Console.Write(minWavData(idx).ToString() + " | ")
      End If
    Next
    Console.WriteLine()
    Console.Write("MIN POWER    |")                          ' 最小 Power の表示
    For idx As Integer = 0 To peakNum - 1
      If dropInfo(idx) <> 0 Then
        Console.Write("---------- | ")
      Else
        Console.Write(minPowData(idx).ToString() + " | ")
      End If
    Next
    Console.WriteLine()
    Console.ReadLine()
  Catch ex As Exception                                      ' 例外（エラー）処理
    Console.WriteLine(ex.Message)                            ' 発生したエラーのメッセージを表示
    Console.ReadLine()                                       ' 終了の Enter キー入力待ち
  End Try
End Sub
```

```
  '=========================================================
  'GP-IB に文字列を送信する関数
  '=========================================================
  Sub GpibWrite(ByVal commandStr As String, ByRef gpib As Device)
    gpib.Write(commandStr)                                    ' データの送信
  End Sub

  '=========================================================
  'GP-IB からデータを 1 行読み込む関数
  '=========================================================
  Function GpibRead(ByRef gpib As Device) As String
    GpibRead = gpib.ReadString()                              ' データの受信
    Exit Function
  End Function

  '=========================================================
  'CALC3 の結果を配列に分割する関数
  '=========================================================
  Sub SplitArrayData(ByVal dataString As String, ByRef dataArray As Double())
    Dim peakNum As Integer
    Dim arrayDataStr As String() = dataString.Split(","c)    '「,」区切りで文字列を分割する。
    peakNum = arrayDataStr.Length
    For idx As Integer = 0 To arrayDataStr.Length - 1
      dataArray(idx) = Convert.ToDouble(arrayDataStr(idx))   ' 分割した文字列を数値に変換
    Next
  End Sub
End Module
```

実行例

```
No.        |1              |
REF WL     | 1.55252293E-06|
REF POWER  | 3.37379314    |
MAX WL     | 1.55252431E-06|
MAX POWER  | 3.43271086    |
MIN WL     | 1.55252289E-06|
MIN POWER  | 3.3485737     |
```

3.1　イーサネットによる接続

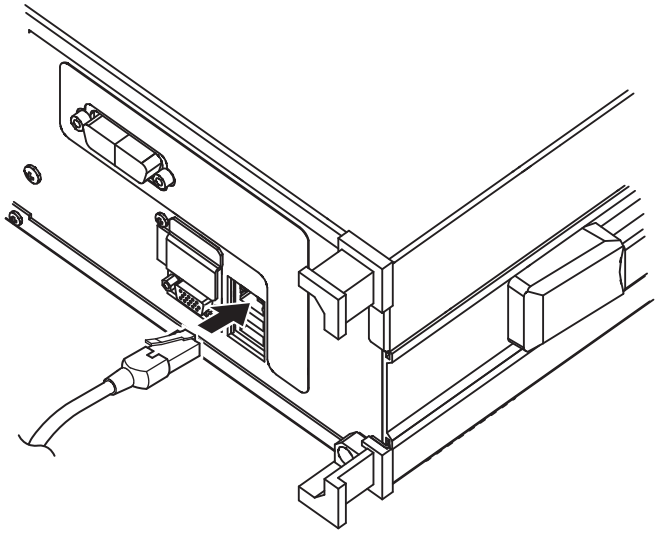
イーサネットポートを使ってネットワークに接続し、PC から本機器を制御できます。

イーサネットポートの仕様

通信ポート数：　1
電気的・機械的仕様：　IEEE802.3 準拠
伝送方式：　イーサネット (10BASE-T/100BASE-TX/1000BASE-T)
伝送速度：　10Mbps/100Mbps/1000Mbps
通信プロトコル：　TCP/IP
コネクタ形状：　RJ45 コネクタ
使用するポート番号：　1024 ～ 65535 の任意の値 (1025、20001 を除く)
同時接続数：　1

接続方法

ハブなどに接続された UTP(Unshielded Twisted-Pair) ケーブルまたは STP(Shielded Twisted-Pair) ケーブルを、本機器のリアパネルにあるイーサネットポートに接続します。



接続時の注意

- 本機器と PC との接続には、必ずハブを介してストレートケーブルを使用してください。
- UTP ケーブル (ストレートケーブル) を使用する場合は、必ずカテゴリー 5 以上のものを使用してください。

3.2　イーサネットポートの機能

リモート制御

イーサネットポートを使用して、ネットワーク経由で本機器をリモート制御することができます。リモート制御は GP-IB による制御と同じコマンドを使用します。
VXI-11 による制御にも対応しています。

リモートコマンド

メッセージターミネータ

本機器では、以下のメッセージターミネータを用いることができます。

プログラムメッセージターミネータ

LF(改行) 文字

ここで、LF を ASCII のラインフィード (0Ah)、CR+LF については CR(0Dh) を wsp として認識するため、結果として CR+LF もメッセージターミネータとして使用できます。

応答メッセージターミネータ

応答メッセージターミネータは CR+LF を用います。

データの問い合わせ

- クエリコマンドは、コマンドの後ろに "?" が付いた形式になっています。
- パラメータを持つクエリコマンドの場合には、"?" の後ろに <wsp> ＋ < パラメータ > の形式で指定します。
- 本機器がクエリコマンドを受信した場合、その時点でのクエリコマンドに対する応答を、出力バッファに準備します。
- 出力バッファのデータは、コントローラの入力ステートメントがあるか、もしくは新たなクエリコマンドを受信するまで保持されます。
- 複数のクエリコマンドが、セミコロン ";" により連続指定記述されている場合には、すべてのクエリコマンドに対応した応答を出力バッファに準備します。
  この場合、すべてのデータを一括で出力します。

リモートモニタ

イーサネットポートを使用して、PC からネットワーク経由で本機器画面のモニタリングや本機器の操作ができます。この機能を使用するには、別途リモートモニタ用のソフトウエアが必要です。
リモートモニタ用のソフトウエアについては、お買い求め先にお問い合わせください。

ディレクトリの共有

本機器の内部メモリのユーザー領域のディレクトリを PC 上で共有化できます。ユーザー領域のディレクトリを共有化すると、以下のファイルをネットワーク経由で PC から読み込めます。なお、本機器への保存はできません。

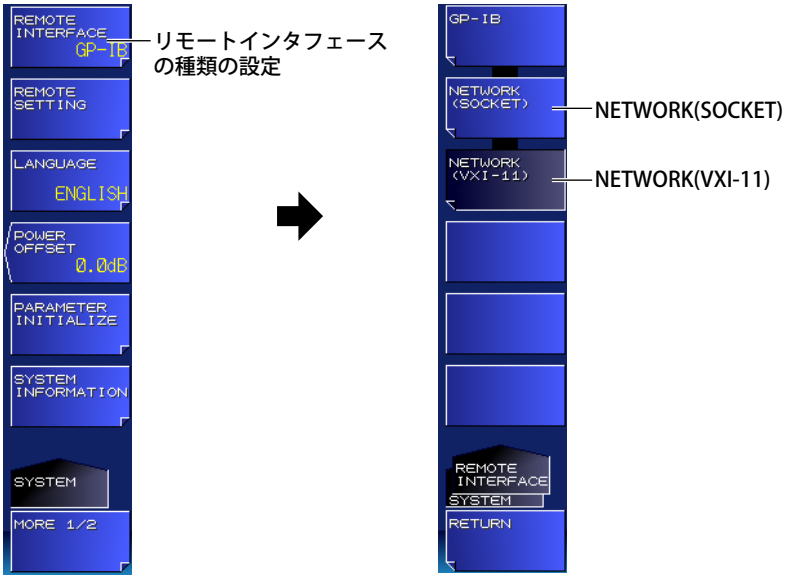
- 測定データ (CSV 形式)
- 設定データ (WS1 形式)
- 画像イメージデータ (BMP 形式)
- ロギングデータ (WG1 形式)

3.3　イーサネットポートの設定

操　作

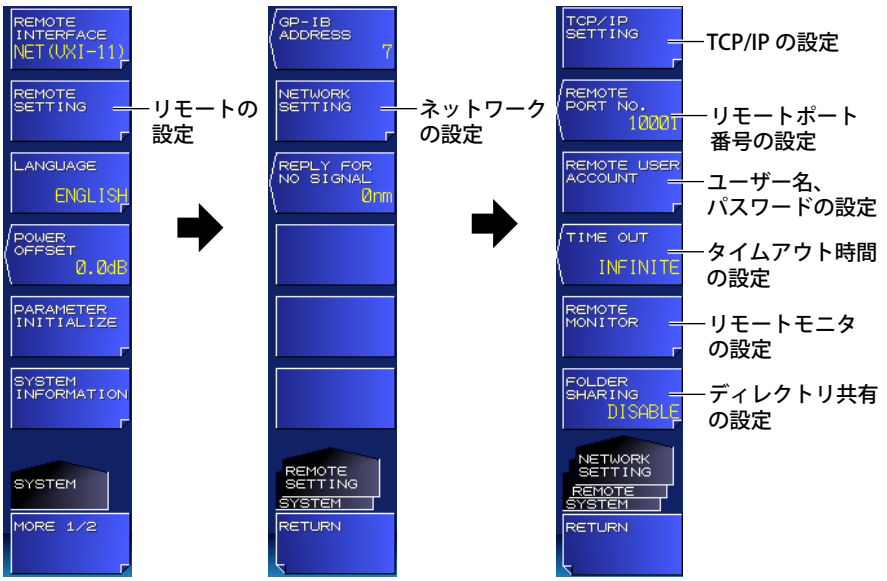
通信インタフェースの選択

1. SYSTEM を押します。
   システム設定メニューが表示されます。
2. REMOTE INTERFACE のソフトキーを押します。
   リモートインタフェース設定メニューが表示されます。
3. NETWORK(SOCKET) または NETWORK(VXI-11) のソフトキーを押します。
   設定メニューの表示が 1 つ前に戻り、ソフトキー上に設定したリモートインタフェースの種類が表示されます。



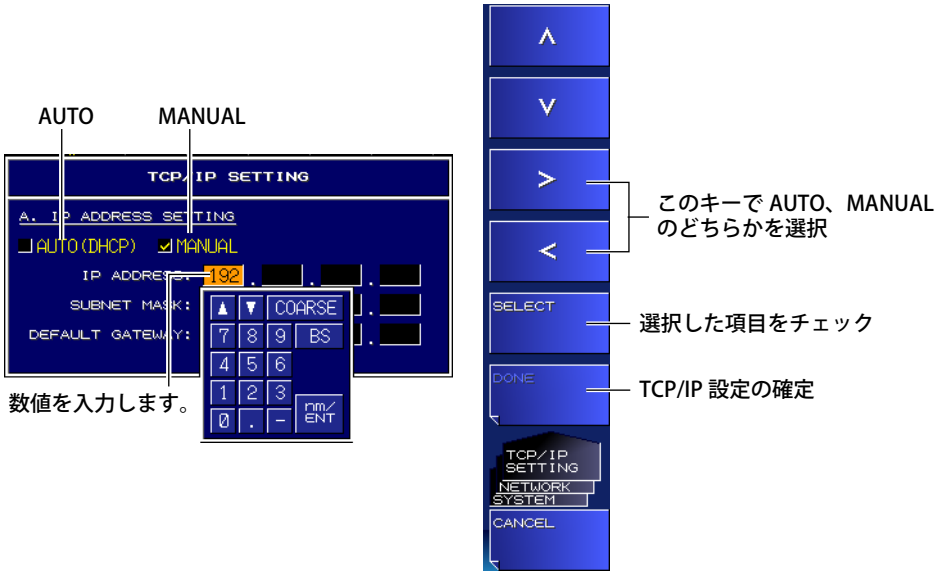
ネットワークの設定

4. REMOTE SETTING のソフトキーを押します。
   リモート設定する画面が表示されます。
5. NETWORK SETTING のソフトキーを押します。
   イーサネットポートに関する設定メニューが表示されます。



- **TCP/IP の設定**

6. **TCP/IP SETTING** のソフトキーを押します。
TCP/IP の設定メニューが表示されます。



7. <、>のソフトキーで、AUTO(DHCP) または MANUAL のどちらかを選択します。

8. **SELECT** のソフトキーを押します。
選択した項目がチェックされます。

9. MANUAL を選択した場合は、IP アドレス、サブネットマスク、デフォルトゲートウエイを設定します。<、>、∧、∨のソフトキーで入力位置を選択し、**ENTER** を押します。

AUTO を選択した場合は、操作 11 に進んでください。

10. 矢印キーまたはテンキーで数値を入力し、**ENTER** キーを押します。

11. すべての設定が終了したら、**DONE** のソフトキーを押します。

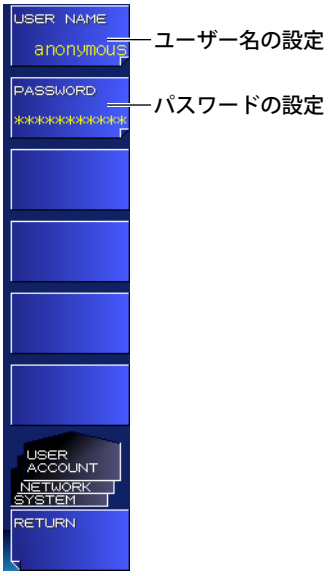
- **リモートポート番号の設定 (VXI-11 では、この設定は使用しません。)**

6. **REMOTE PORT NO.** のソフトキーを押します。
ポート番号の設定画面が表示されます。

7. 矢印キーまたはテンキーでポート番号を入力します。

- **ユーザー名、パスワードの設定 (VXI-11 では、この設定は使用しません。)**

  6. **REMOTE USER ACCOUNT** のソフトキーを押します。
  ユーザー名、パスワードの設定メニューが表示されます。

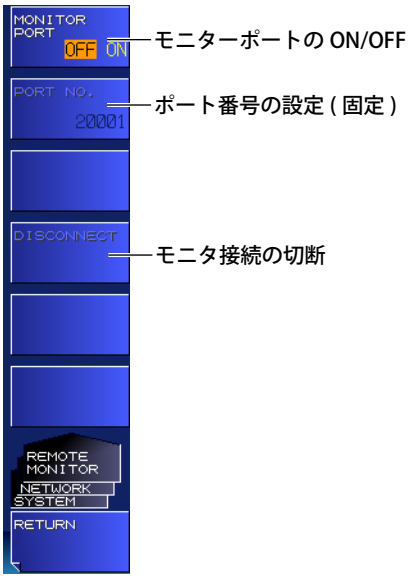


  7. **USER NAME** のソフトキーを押します。
  ユーザー名の設定画面が表示されます。文字列の入力方法については、スタートガイド IM AQ6150B-02JA の 3.3 節をご覧ください。

  8. **PASSWORD** のソフトキーを押します。
  パスワードの設定画面が表示されます。文字列の入力方法については、スタートガイド IM AQ6150B-02JA の 3.3 節をご覧ください。

- **リモートモニタの設定**

  6. **REMOTE MONITOR** のソフトキーを押します。
  リモートモニタの設定メニューが表示されます。



  7. **MONITOR PORT** のソフトキーを押します。
  ソフトキーを押すごとに ON/OFF が切り替わります。ON のときにリモートモニタができます。

- **リモートモニタ接続の切断**

  8. **DISCONNECT** のソフトキーを押します。
  PC からのモニタ接続状態が切断されます。

- **ディレクトリ共有の設定**

  6. **FOLDER SHARING** のソフトキーを押します。
  ディレクトリ共有の設定メニューが表示されます。

  7. **READ ONLY** のソフトキーを押します。
  本機器のユーザー領域ディレクトリが共有されます (読み込みだけ)。

- **ディレクトリ共有の解除**

  8. **DISABLE** のソフトキーを押します。
  ユーザー領域ディレクトリの共有が解除されます。

- **タイムアウト時間の設定 (Network(SOCKET) 通信用)**

  6. **TIME OUT** のソフトキーを押します。パラメータ入力ウィンドウが表示されます。

  7. パラメータ入力ウィンドウでタイムアウト時間を入力します。

解　説

本機器の TCP/IP を設定します。

TCP/IP の設定

本機能をネットワークに接続するためには、本機器の IP アドレスを正しく設定する必要があります。
本機器を接続するネットワーク上に DHCP サーバーが用意されている場合、本機器に与えられる IP アドレスは自動的に設定されます。その場合は、IP ADDRESS SETTING は "AUTO" にセットしてください。
本機器を接続するネットワークの詳細については、ネットワーク管理者にお問い合わせください。

REMOTE PORT NO. (VXI-11 では、この設定は使用されません。)

イーサネットポートからリモート制御するためのポート番号を設定します。値は 1024 ～ 65535 の間で任意に設定できます (1025、20001 を除く)。

ユーザー認証について (VXI-11 では、この設定は使用されません。)

イーサネットポートを使ってネットワーク経由で PC から本機器に接続する場合、ユーザー認証が必要です。ユーザー名が anonymous の場合は、パスワードは必要ありません。
本機器は、平文認証と MD5 アルゴリズム (RSA Data Security, Inc. MD5 Message Digest Algorithm) に対応しています。ユーザー名およびパスワードは 11 文字以内で設定してください。ユーザー名の初期値は anonymous です。

コマンドによるリモート制御 (SOCKET の場合)

インタフェースの切り替えについて

リモート制御に使用するインタフェースを GP-IB とイーサネットから選択します。
"GP-IB" に設定したとき、または LOCAL キーを押したときにイーサネットポートのリモート接続状態はリセットされます。これ以外には、コントローラ側からの切断操作がない限り接続は維持されます。

SRQ による割り込み

イーサネットポートによるリモート制御時は、SRQ による割り込みは発生しません。

ステータスレジスタ

ステータスレジスタは、GP-IB によるリモート制御時と同様に動作します。
ステータスレジスタは、"*STB?" コマンドを使用することで、GP-IB のシリアルポールと同様に読み出すことができます。

トーカデータの送信

本機器は、トーカデータを制御 PC から受信すると、制御 PC のバッファにデータを送信します。制御 PC のバッファデータを読み込んでデータを取得してください。

接続

本機器は、1 台のコントローラ (PC など) とだけ接続できます。すでにコントローラと接続した状態で、他のコントローラから接続要求があった場合、新しい接続は行わずに現状の接続が維持されます。

タイムアウト時間 (VXI-11 では、この設定は使用されません)

リモート状態で、通信していない時間が設定した時間を経過すると、自動的に通信を切断してローカル状態になります。
タイムアウト時間を変更すると、経過時間がリセットされます。
設定できる時間は、INFINITE(0s)、1 ～ 21600s(6 時間) です。

イーサネットポートによるリモート制御に必要なコマンド (SOCKET の場合)

イーサネットポートによるリモート接続には、OPEN コマンドによる認証が必要です。認証を受けないと接続できません。

OPEN

機能　ユーザー名を送信し、ユーザー認証を開始します。

構文　OPEN<wsp>"username"

username= ユーザー名

例　OPEN "yokogawa"

-> AUTHENTICATE CRAM-MD5.

解説　OPEN コマンドにより、以下の手順で認証が行われます。

平文認証の場合

1. 本機器に OPEN "username" を送信し、本機器からの応答メッセージを取得します。
2. 取得したメッセージが "AUTHENTICATE CRAM-MD5." であることを確認します。
3. 本機器にパスワードを送信します (user name が anonymous のときはパスワードは何でも可)。
4. 本機器から "READY" のメッセージを取得すると認証が完了します。本機器の REMOTE ランプが点灯し、リモートコマンド送信が可能になります。
   ユーザー名やパスワードが違う場合は認証に失敗し、接続が解除されます。

暗号化認証の場合

1. 本機器に OPEN "username" を送信し、本機器からの応答メッセージを取得します。
2. 取得したメッセージが "AUTHENTICATE CRAM-MD5." であることを確認します。
3. 本機器に "AUTHENTICATE CRAM-MD5 OK." を送信し、本機器の応答メッセージ (チャレンジ文字列) を取得します。
4. 取得したチャレンジ文字列とパスワードを、MD5 方式でハッシュ計算します (user name が anonymous のときはパスワードは何でも可)。
5. 得られたハッシュデータ (16 進数小文字 x 32 文字) を本機器に送信し、応答メッセージを取得します。
6. 本機器から "READY" のメッセージを取得すると認証が完了します。本機器の REMOTE ランプが点灯し、リモートコマンド送信が可能になります。
   ユーザー名やパスワードが違う場合は認証に失敗し、接続が解除されます。

CLOSE

機能　コネクションを OFF にし、ローカル状態に切り替えます。

構文　CLOSE

例　CLOSE

Note

- 本機器をネットワークに接続して起動する場合、起動処理に数分間かかる場合があります。起動処理中は、画面の下側に初期化動作の進行状況を示す「STEP 1/6」～「STEP 6/6」が表示されます。
- 起動処理が完了して測定画面が表示されたあとに、ネットワーク経由で PC から本機器へ接続できるようになるまで数分間かかる場合があります。

- **リモートモニタ機能を使用中の測定時間**

  リモートモニタ機能を使用して PC 上のリモートモニタ用のソフトウェアから本機器画面のモニタリングや本機器を操作すると、リモートモニタ機能を使用しないときに比べて、測定時間が長くなる場合があります。

3.4　サンプルプログラム (SOCKET の場合)

イーサネットポートを使用して本機器をリモートコントロールする例を示します。
対象モデル　：PC-AT 機
対象言語　　：Visual Basic 2008
使用コンポーネント：.NET Framework 3.5

サンプルプログラム 1

イーサネットにて Single 測定を 1 回行い、結果 (波長情報、Power 情報、FP-LD 解析結果) を画面に出力するサンプルプログラム (仕様コード -MW: マルチ波長タイプの例です。)

ソースコード

```
Imports System
Imports System.IO
Imports System.Net.Sockets
Imports System.Text

Module EtherSingleMeasure
'
' Ether にて Single 測定を 1 回行い結果 ( 波長情報、Power 情報、FP-LD 解析結果 ) を画面に出力する
' サンプルプログラム
'
  Sub Main()
    Try
      Dim wlmAddr As String
      Dim wlmPort As Integer
      Dim sockStream As NetworkStream
      Dim tcpObj As TcpClient
      Dim replyString As String
      Dim wavArray As Double()
      Dim powArray As Double()
      Dim fwhm, ctrWl, totalPwr, sigma As Double
      Dim username, passwd As String
      Dim maxPeakPower, maxPeakWl As Double

      '=================================================
      ' 波長計の情報
      '=================================================
      wlmAddr = "192.168.0.1"                          ' 波長計の IP アドレス
      wlmPort = 10001                                  ' リモートポート番号
      username = "anonymous"                           ' ユーザ名
      passwd = ""                                      ' パスワード

      '=================================================
      'TCP 接続
      '=================================================
      tcpObj = New TcpClient
      tcpObj.Connect(wlmAddr, wlmPort)                 'TCP の接続
      sockStream = tcpObj.GetStream()
      tcpObj.NoDelay = True                            'TCP_NODELAY を有効にする

      '=================================================
      ' 認証の実行
      '=================================================
      Dim recvBuffer As String
      TcpWriteLine("open """ + username + """", sockStream)'OPEN コマンド、ユーザ名の送信
      recvBuffer = TcpReadLine(sockStream)
      If String.Compare(recvBuffer, "AUTHENTICATE CRAM-MD5") <> 0 Then
        sockStream.Dispose()
        Exit Sub                                       ' 応答が「AUTHENTICATE CRAM-MD5」でなかっ
                                                       ' たらエラー
      End If
      TcpWriteLine(passwd, sockStream)                 ' パスワードの送信
      recvBuffer = TcpReadLine(sockStream)
```

```
If String.Compare(recvBuffer, "ready") <> 0 Then
  sockStream.Dispose()
  Exit Sub                                                  ' 認証失敗
End If

'=====================================================
' 波長計の測定条件設定
'=====================================================
Call TcpWriteLine("*RST", sockStream)                       ' 本機器のリセット
Call TcpWriteLine(":CALC2:PTHR:MODE REL", sockStream)       ' ピーク検出しきい値設定を相対値モード
                                                            ' に設定
Call TcpWriteLine(":CALC2:PTHR 15", sockStream)             ' ピーク検出しきい値を 15db に設定
Call TcpWriteLine(":UNIT:WL NM", sockStream)                ' 波長単位を nm に設定
Call TcpWriteLine(":UNIT:POW DBM", sockStream)              'Power 単位を dBm に設定
Call TcpWriteLine(":DISP:WIND2:STAT ON", sockStream)        ' グラフ表示を有効化

'=====================================================
' 測定の実行、データの取得
'=====================================================
' 測定は READ コマンドにより実行し、データを取得する。
' さらに Power 情報は FETC コマンドで測定済みのデータを取得する。
Call TcpWriteLine(":READ:ARR:POW:WAV?", sockStream)         'Single 測定の実行と波長データの取得
replyString = TcpReadLine(sockStream)
Call SplitArrayData(replyString, wavArray)                  ' 波長情報を配列に格納

Call TcpWriteLine(":FETC:ARR:POW?", sockStream)             ' 測定済み Power の取得
replyString = TcpReadLine(sockStream)
Call SplitArrayData(replyString, powArray)                  'Power 情報を配列に格納

'=====================================================
' 結果の表示 ( 波長、Power 情報 )
'=====================================================
Console.WriteLine("No.|Wavelength(m)|Power(dBm)")
For idx As Integer = 1 To wavArray.Length
  Console.WriteLine((idx).ToString + "|" + wavArray(idx - 1).ToString() + "|" + _
                    powArray(idx - 1).ToString())
Next

'=====================================================
' 最大 Power ピーク情報の取得
'=====================================================
Call TcpWriteLine(":FETC:POW? MAX", sockStream)             ' 最大 Power のピークを指定して Power
                                                            ' を取得
replyString = TcpReadLine(sockStream)
maxPeakPower = Convert.ToDouble(replyString)
Call TcpWriteLine(":FETC:POW:WAV?", sockStream)             ':FETC:POW? MAX で指定されたピークの
                                                            ' 波長を取得
replyString = TcpReadLine(sockStream)
maxPeakWl = Convert.ToDouble(replyString)
Console.WriteLine("Highest Peak Power     :" + maxPeakPower.ToString + " dBm")
Console.WriteLine("Highest Peak Wavelength:" + maxPeakWl.ToString + " nm")

'=====================================================
'   FP-LD 解析結果の取得
'=====================================================
Call TcpWriteLine(":CALC3:FPER ON", sockStream)             'FP-LD 解析の有効化
Call TcpWriteLine(":CALC3:FPER:FWHM?", sockStream)          'FWHM の取得
replyString = TcpReadLine(sockStream)
fwhm = Convert.ToDouble(replyString)
Call TcpWriteLine(":CALC3:FPER:MEAN?", sockStream)          'Center WL の取得
replyString = TcpReadLine(sockStream)
ctrWl = Convert.ToDouble(replyString)
Call TcpWriteLine(":CALC3:FPER:POW?", sockStream)           'Total Powr の取得
replyString = TcpReadLine(sockStream)
totalPwr = Convert.ToDouble(replyString)
Call TcpWriteLine(":CALC3:FPER:SIGM?", sockStream)          'σ の取得
replyString = TcpReadLine(sockStream)
sigma = Convert.ToDouble(replyString)
```

```
    ' 結果の表示 (FP-LD 解析結果 )
    Console.WriteLine("====FP-LD Analysis====")
    Console.WriteLine("FWHM             : " + (fwhm * 1000000000).ToString + "nm")
    Console.WriteLine("Sigma            : " + (sigma * 1000000000).ToString + "nm")
    Console.WriteLine("CTR WL           : " + (ctrWl * 1000000000).ToString + "nm")
    Console.WriteLine("TOTAL PWR        : " + totalPwr.ToString + "dBm")

    '=====================================================
    ' 内部メモリへのデータ保存
    '=====================================================
    ' 画面イメージと結果データを内部メモリに保存する。
    Call TcpWriteLine(":MMEM:STOR SIM2,""\WLM_IMAGE"",INT", sockStream)
    Call TcpWriteLine(":MMEM:STOR TABL,""\WLM_TABLE"",INT", sockStream)

    '=====================================================
    ' 内部メモリに保存したデータを PC に転送
    '=====================================================
    Call TcpWriteLine(":MMEM:DATA? ""\WLM_IMAGE.BMP"",INT", sockStream)
    TcpReadBlockData2File(sockStream, "WLM_IMAGE.BMP")
    Call TcpWriteLine(":MMEM:DATA? ""\WLM_TABLE.CSV"",INT", sockStream)
    TcpReadBlockData2File(sockStream, "WLM_TABLE.CSV")
    sockStream.Dispose()                                    'TCP の Close

    Console.ReadLine()
  Catch ex As Exception                                     ' 例外 ( エラー ) 処理
    Console.WriteLine(ex.Message)                           ' 発生したエラーのメッセージを表示
    Console.ReadLine()                                      ' 終了の Enter キー入力待ち
  End Try
End Sub

'=====================================================
'TCP Socket に文字列を送信する関数
'=====================================================
Sub TcpWriteLine(ByVal commandStr As String, ByRef stream As NetworkStream)
  Dim writer As StreamWriter = New StreamWriter(stream, Encoding.ASCII)
  Dim ByteLf As Byte() = New Byte() {10}
  writer.NewLine = Encoding.ASCII.GetString(ByteLf)         ' 改行コードは LF
  writer.AutoFlush = True
  writer.WriteLine(commandStr)                              ' データの送信
End Sub

'=====================================================
'TCP Socket からデータを 1 行読み込む関数
'=====================================================
Function TcpReadLine(ByRef stream As NetworkStream) As String
  Dim reader As StreamReader = New StreamReader(stream, Encoding.ASCII)
  TcpReadLine = reader.ReadLine()                           ' データの受信
  Exit Function
End Function

'=====================================================
'TCP Socket からブロックデータを読み込んでファイルに保存する関数
'=====================================================
Function TcpReadBlockData2File(ByRef stream As NetworkStream, _
         ByVal filename As String) As Integer
  Dim headerLen As Integer
  Dim dataLen As Integer
  Dim readLen As Integer
  Dim file As New FileStream(filename, FileMode.Create, FileAccess.Write)
  Dim recvBuffer As Byte() = New Byte(1024) {}
  Dim ByteSharp As Byte = Asc("#")
  stream.Read(recvBuffer, 0, 1)                             '1 文字目の取得
  If recvBuffer(0) <> ByteSharp Then                        '1 文字目は「# 以外」はエラー
    TcpReadBlockData2File = -1
    Exit Function
  End If
  stream.Read(recvBuffer, 0, 1)
```

```
    headerLen = Integer.Parse(Encoding.ASCII.GetString(recvBuffer)) ' データ長情報が入っている
                                                                    ' エリアの大きさ
    stream.Read(recvBuffer, 0, headerLen)                    ' データ長情報エリアの読み込み
    dataLen = Integer.Parse(Encoding.ASCII.GetString(recvBuffer))
                                                             ' データ長情報の取得

    While dataLen > 1024
      readLen = stream.Read(recvBuffer, 0, 1024)             ' データを 1024byte づつ読み込み
      file.Write(recvBuffer, 0, readLen)                     ' 取得したデータをファイルへ書き込む
      dataLen = dataLen - readLen
    End While
    readLen = stream.Read(recvBuffer, 0, recvBuffer.Length) ' 最後のデータを取得
    file.Write(recvBuffer, 0, dataLen)                       ' 取得したデータをファイルへ書き込む

    file.Close()
    TcpReadBlockData2File = 0
  End Function

  '=========================================================
  'READ/FETC/MEAS の結果を配列に分割する関数
  '=========================================================
  Sub SplitArrayData(ByVal dataString As String, ByRef dataArray As Double())
    Dim peakNum As Integer
    Dim arrayDataStr As String() = dataString.Split(","c)    '「,」区切りでデータを分割
    peakNum = Integer.Parse(arrayDataStr(0))                 ' データ数の取得
    dataArray = New Double(peakNum - 1) {}
    For idx As Integer = 1 To arrayDataStr.Length - 1
      dataArray(idx - 1) = Convert.ToDouble(arrayDataStr(idx)) ' データ数分データを読み込む
    Next
  End Sub
End Module
```

実行例

```
No.|Wavelength(m) |Power(dBm)
  1|1.30678822E-06|-14.3279541
  2|1.30756963E-06|-9.42082105
  3|1.30835228E-06|-2.23592107
  4|1.30913555E-06|-3.93065804
  5|1.30991986E-06|-13.5578301
Highest Peak Power     :-2.23592107 dBm
Highest Peak Wavelength:1.30835228E-06 nm
====FP-LD Analysis====
FWHM             : 1.47415158nm
Sigma            : 0.625966702nm
CTR WL           : 1308.55169nm
TOTAL PWR        : 0.782282871dBm
```

サンプルプログラム 2

イーサネットにて Single 測定を 1 回行い、結果 (波長情報、Power 情報) を画面に出力するサンプルプログラム
(仕様コード -SW: シングル波長タイプの例です。)

ソースコード

```
Imports System
Imports System.IO
Imports System.Net.Sockets
Imports System.Text

Module EtherSingleMeasure
'
' Ether にて Single 測定を 1 回行い結果 ( 波長情報、Power 情報 ) を画面に出力するサンプルプログラム
'
  Sub Main()
    Try
      Dim wlmAddr As String
      Dim wlmPort As Integer
      Dim sockStream As NetworkStream
      Dim tcpObj As TcpClient
      Dim replyString As String
      Dim wavelength As Double
      Dim power As Double
      Dim username, passwd As String

      '================================================
      ' 波長計の情報
      '================================================
      wlmAddr = "192.168.0.1"                          ' 波長計の IP アドレス
      wlmPort = 10001                                  ' リモートポート番号
      username = "anonymous"                           ' ユーザ名
      passwd = ""                                      ' パスワード

      '================================================
      'TCP 接続
      '================================================
      tcpObj = New TcpClient
      tcpObj.Connect(wlmAddr, wlmPort)                 'TCP の接続
      sockStream = tcpObj.GetStream()
      tcpObj.NoDelay = True                            'TCP_NODELAY を有効にする

      '================================================
      ' 認証の実行
      '================================================
      Dim recvBuffer As String
      TcpWriteLine("open """ + username + """", sockStream)'OPEN コマンド、ユーザ名の送信
      recvBuffer = TcpReadLine(sockStream)
      If String.Compare(recvBuffer, "AUTHENTICATE CRAM-MD5") <> 0 Then
        sockStream.Dispose()
        Exit Sub                                       ' 応答が「AUTHENTICATE CRAM-MD5」でなかっ
                                                       ' たらエラー
      End If
      TcpWriteLine(passwd, sockStream)                 ' パスワードの送信
      recvBuffer = TcpReadLine(sockStream)
      If String.Compare(recvBuffer, "ready") <> 0 Then
        sockStream.Dispose()
        Exit Sub                                         ' 認証失敗
      End If
```

```
    '=====================================================
    ' 波長計の測定条件設定
    '=====================================================
    Call TcpWriteLine("*RST", sockStream)                  ' 本機器のリセット
    Call TcpWriteLine(":UNIT:WL NM", sockStream)           ' 波長単位を nm に設定
    Call TcpWriteLine(":UNIT:POW DBM", sockStream)         'Power 単位を dBm に設定

    '=====================================================
    ' 測定の実行、データの取得
    '=====================================================
    ' 測定は READ コマンドにより実行し、データを取得する。
    ' さらに Power 情報は FETC コマンドで測定済みのデータを取得する。
    Call TcpWriteLine(":READ:POW:WAV?", sockStream)        'Single 測定の実行と波長データの取得
    replyString = TcpReadLine(sockStream)
    wavelength = Convert.ToDouble(replyString)             ' 応答文字列を数値に変換

    Call TcpWriteLine(":FETC:POW?", sockStream)            ' 測定済み Power の取得
    replyString = TcpReadLine(sockStream)
    power = Convert.ToDouble(replyString)                  ' 応答文字列を数値に変換

    '=====================================================
    ' 結果の表示 ( 波長、Power 情報 )
    '=====================================================
    Console.WriteLine("Wavelength(m):" + wavelength.ToString())
    Console.WriteLine("Power(dBm)   :" + power.ToString())

    '=====================================================
    ' 内部メモリへのデータ保存
    '=====================================================
    ' 画面イメージと結果データを内部メモリに保存する。
    Call TcpWriteLine(":MMEM:STOR SIM2,""\WLM_IMAGE"",INT", sockStream)
    Call TcpWriteLine(":MMEM:STOR TABL,""\WLM_TABLE"",INT", sockStream)

    '=====================================================
    ' 内部メモリに保存したデータを PC に転送
    '=====================================================
    Call TcpWriteLine(":MMEM:DATA? ""\WLM_IMAGE.BMP"",INT", sockStream)
    TcpReadBlockData2File(sockStream, "WLM_IMAGE.BMP")
    Call TcpWriteLine(":MMEM:DATA? ""\WLM_TABLE.CSV"",INT", sockStream)
    TcpReadBlockData2File(sockStream, "WLM_TABLE.CSV")
    sockStream.Dispose()                                   'TCP の Close

    Console.ReadLine()
  Catch ex As Exception                                    ' 例外 ( エラー ) 処理
    Console.WriteLine(ex.Message)                          ' 発生したエラーのメッセージを表示
    Console.ReadLine()                                     ' 終了の Enter キー入力待ち
  End Try
End Sub

'=========================================================
'TCP Socket に文字列を送信する関数
'=========================================================
Sub TcpWriteLine(ByVal commandStr As String, ByRef stream As NetworkStream)
  Dim writer As StreamWriter = New StreamWriter(stream, Encoding.ASCII)
  Dim ByteLf As Byte() = New Byte() {10}
  writer.NewLine = Encoding.ASCII.GetString(ByteLf)        ' 改行コードは LF
  writer.AutoFlush = True
  writer.WriteLine(commandStr)                             ' データの送信
End Sub
```

```
'=========================================================
'TCP Socket からデータを 1 行読み込む関数
'=========================================================
Function TcpReadLine(ByRef stream As NetworkStream) As String
  Dim reader As StreamReader = New StreamReader(stream, Encoding.ASCII)
  TcpReadLine = reader.ReadLine()                          ' データの受信
  Exit Function
End Function

'=========================================================
'TCP Socket からブロックデータを読み込んでファイルに保存する関数
'=========================================================
Function TcpReadBlockData2File(ByRef stream As NetworkStream, _
         ByVal filename As String) As Integer
  Dim headerLen As Integer
  Dim dataLen As Integer
  Dim readLen As Integer
  Dim file As New FileStream(filename, FileMode.Create, FileAccess.Write)
  Dim recvBuffer As Byte() = New Byte(1024) {}
  Dim ByteSharp As Byte = Asc("#")
  stream.Read(recvBuffer, 0, 1)                            '1 文字目の取得
  If recvBuffer(0) <> ByteSharp Then                       '1 文字目は「# 以外」はエラー
    TcpReadBlockData2File = -1
    Exit Function
  End If
  stream.Read(recvBuffer, 0, 1)
  headerLen = Integer.Parse(Encoding.ASCII.GetString(recvBuffer)) ' データ長情報が入っている
                                                                  ' エリアの大きさ
  stream.Read(recvBuffer, 0, headerLen)                    ' データ長情報エリアの読み込み
  dataLen = Integer.Parse(Encoding.ASCII.GetString(recvBuffer))
                                                           ' データ長情報の取得

  While dataLen > 1024
    readLen = stream.Read(recvBuffer, 0, 1024)             ' データを 1024byte づつ読み込み
    file.Write(recvBuffer, 0, readLen)                     ' 取得したデータをファイルへ書き込む
    dataLen = dataLen - readLen
  End While
  readLen = stream.Read(recvBuffer, 0, recvBuffer.Length) ' 最後のデータを取得
  file.Write(recvBuffer, 0, dataLen)                       ' 取得したデータをファイルへ書き込む

  file.Close()
  TcpReadBlockData2File = 0
 End Function
End Module
```

実行例

```
Wavelength(m): 1.55252398E-06
Power(dBm)   : 3.43060397
```

サンプルプログラム 3

イーサネットにてドリフト解析を行うサンプルプログラム (仕様コードが -MW: マルチ波長の機種の例です)

ソースコード

```
Imports System
Imports System.IO
Imports System.Net.Sockets
Imports System.Text

Module EtherDriftMeasure
'Ether にて Drift 解析を行うサンプルプログラム
  Sub Main()
    Try
      Dim wlmAddr As String
      Dim wlmPort As Integer
      Dim sockStream As NetworkStream
      Dim tcpObj As TcpClient
      Dim replyString As String
      Dim peakNum As Integer
      Dim refPowData, refWavData As Double()
      Dim maxPowData, maxWavData As Double()
      Dim minPowData, minWavData As Double()
      Dim dropInfo As Double()
      Dim username, passwd As String

      '=====================================================
      ' 波長計の情報
      '=====================================================
      wlmAddr = "192.168.0.1"                              ' 波長計の IP アドレス
      wlmPort = 10001                                      ' リモートポート番号
      username = "anonymous"                               ' ユーザ名
      passwd = ""                                          ' パスワード

      '=====================================================
      ' TCP 接続
      '=====================================================
      tcpObj = New TcpClient
      tcpObj.Connect(wlmAddr, wlmPort)
      sockStream = tcpObj.GetStream()
      tcpObj.NoDelay = True                                'TCP_NODELAY を有効にする

      '=====================================================
      ' 認証の実行
      '=====================================================
      Dim recvBuffer As String
      TcpWriteLine("open """ + username + """", sockStream) 'OPEN コマンド、ユーザ名の送信
      recvBuffer = TcpReadLine(sockStream)
      If String.Compare(recvBuffer, "AUTHENTICATE CRAM-MD5") <> 0 Then
        sockStream.Dispose()
        Exit Sub                                           ' 応答が「AUTHENTICATE CRAM-MD5」
                                                           ' でなかったらエラー
      End If
      TcpWriteLine(passwd, sockStream)                     ' パスワードの送信
      recvBuffer = TcpReadLine(sockStream)

      If String.Compare(recvBuffer, "ready") <> 0 Then
        sockStream.Dispose()
        Exit Sub                                           ' 認証失敗
      End If

      '=====================================================
      ' 波長計の測定条件設定
      '=====================================================
      Call TcpWriteLine("*RST", sockStream)                ' 本機器のリセット
      Call TcpWriteLine(":CALC2:PTHR:MODE REL", sockStream) ' しきい値設定を相対値モードに
      Call TcpWriteLine(":CALC2:PTHR 15", sockStream)      ' しきい値を 15db に設定
      Call TcpWriteLine(":UNIT:WL NM", sockStream)         ' 波長単位を nm に設定
      Call TcpWriteLine(":UNIT:POW DBM", sockStream)       'Power 単位を dBm に設定
```

```
' ドリフト測定のリファレンスとするため Single 測定を行う。
Call TcpWriteLine(":INIT;*OPC?", sockStream)          'Single 測定の実行と測定完了待ち
TcpReadLine(sockStream)                               ' 測定完了待ち (*OPC?) の応答を読む
Call TcpWriteLine(":CALC3:DRIF ON", sockStream)       'DRIFT 解析の ON

'=====================================================
' 測定の実行
'=====================================================
Call TcpWriteLine(":INIT:CONT ON", sockStream)        'Repeat 測定の開始

For count As Integer = 1 To 60                        '1 分間待つ
  Threading.Thread.Sleep(1000)
  Console.Write(".")
Next
Console.WriteLine("")
Call TcpWriteLine(":INIT:CONT OFF", sockStream)       'Repeat 測定の停止

'=====================================================
' 測定結果の取得
'=====================================================
Call TcpWriteLine(":CALC3:POIN?", sockStream)         ' データ数の取得
replyString = TcpReadLine(sockStream)
peakNum = Integer.Parse(replyString)
refPowData = New Double(peakNum - 1) {}
refWavData = New Double(peakNum - 1) {}
maxPowData = New Double(peakNum - 1) {}
maxWavData = New Double(peakNum - 1) {}
minPowData = New Double(peakNum - 1) {}
minWavData = New Double(peakNum - 1) {}
dropInfo = New Double(peakNum - 1) {}

' 結果の取得 ( リファレンス値 )
Call TcpWriteLine(":CALC3:DRIF:REF ON", sockStream)
Call TcpWriteLine(":CALC3:DATA? POW", sockStream)     'Ref Power の取得
replyString = TcpReadLine(sockStream)
SplitArrayData(replyString, refPowData)
Call TcpWriteLine(":CALC3:DATA? WAV", sockStream)     'Ref 波長の取得
replyString = TcpReadLine(sockStream)
SplitArrayData(replyString, refWavData)
' 結果の取得 (MAX 値 )
Call TcpWriteLine(":CALC3:DRIF:PRES", sockStream)
Call TcpWriteLine(":CALC3:DRIF:MAX ON", sockStream)
Call TcpWriteLine(":CALC3:DATA? POW", sockStream)     'MAX Power の取得
replyString = TcpReadLine(sockStream)
SplitArrayData(replyString, maxPowData)
Call TcpWriteLine(":CALC3:DATA? WAV", sockStream)     'MAX 波長の取得
replyString = TcpReadLine(sockStream)
SplitArrayData(replyString, maxWavData)
' 結果の取得 (MIN 値 )
Call TcpWriteLine(":CALC3:DRIF:PRES", sockStream)
Call TcpWriteLine(":CALC3:DRIF:MIN ON", sockStream)
Call TcpWriteLine(":CALC3:DATA? POW", sockStream)     'MIN Power の取得
replyString = TcpReadLine(sockStream)
SplitArrayData(replyString, minPowData)
Call TcpWriteLine(":CALC3:DATA? WAV", sockStream)     'MIN 波長の取得
replyString = TcpReadLine(sockStream)
SplitArrayData(replyString, minWavData)
'Drop 情報の取得
Call TcpWriteLine(":CALC3:DATA? DROP", sockStream)
replyString = TcpReadLine(sockStream)
SplitArrayData(replyString, dropInfo)

sockStream.Dispose()                                  'TCP の Close
```

```
    '=====================================================
    ' 測定結果の表示
    '=====================================================
    Console.Write("No.         |")                          ' ピーク番号の表示

    For idx As Integer = 0 To peakNum - 1
      Console.Write((idx + 1).ToString() + "            |")
    Next
    Console.WriteLine()
    Console.Write("REF WL      |")                          ' リファレンス波長の表示
    For idx As Integer = 0 To peakNum - 1
      Console.Write(refWavData(idx).ToString() + " | ")
    Next
    Console.WriteLine()
    Console.Write("REF POWER   |")                          ' リファレンス Power の表示
    For idx As Integer = 0 To peakNum - 1
      Console.Write(refPowData(idx).ToString() + " | ")
    Next
    Console.WriteLine()
    Console.Write("MAX WL      |")                          ' 最大波長の表示
    For idx As Integer = 0 To peakNum - 1
      If dropInfo(idx) <> 0 Then
        Console.Write("---------- | ")
      Else
        Console.Write(maxWavData(idx).ToString() + " | ")
      End If
    Next
    Console.WriteLine()
    Console.Write("MAX POWER   |")                          ' 最大 Power の表示
    For idx As Integer = 0 To peakNum - 1
      If dropInfo(idx) <> 0 Then
        Console.Write("---------- | ")
      Else
        Console.Write(maxPowData(idx).ToString() + " | ")
      End If
    Next
    Console.WriteLine()
    Console.Write("MIN WL      |")                          ' 最小波長の表示
    For idx As Integer = 0 To peakNum - 1
      If dropInfo(idx) <> 0 Then
        Console.Write("---------- | ")
      Else
        Console.Write(minWavData(idx).ToString() + " | ")
      End If
    Next
    Console.WriteLine()
    Console.Write("MIN POWER   |")                          ' 最小 Power の表示
    For idx As Integer = 0 To peakNum - 1
      If dropInfo(idx) <> 0 Then
        Console.Write("---------- | ")
      Else
        Console.Write(minPowData(idx).ToString() + " | ")
      End If
    Next
    Console.WriteLine()
    Console.ReadLine()
  Catch ex As Exception                                     ' 例外 ( エラー ) 処理
    Console.WriteLine(ex.Message)                           ' 発生したエラーのメッセージを表示
    Console.ReadLine()                                      ' 終了の Enter キー入力待ち
  End Try
End Sub

'=====================================================
'TCP Socket に文字列を送信する関数
'=====================================================
Sub TcpWriteLine(ByVal commandStr As String, ByRef stream As NetworkStream)
  Dim writer As StreamWriter = New StreamWriter(stream, Encoding.ASCII)
```

```
    Dim ByteLf As Byte() = New Byte() {10}
    writer.NewLine = Encoding.ASCII.GetString(ByteLf)        ' 改行コードは LF
    writer.AutoFlush = True
    writer.WriteLine(commandStr)                             ' データの送信
  End Sub

  '=========================================================
  'TCP Socket からデータを 1 行読み込む関数
  '=========================================================
  Function TcpReadLine(ByRef stream As NetworkStream) As String
    Dim reader As StreamReader = New StreamReader(stream, Encoding.ASCII)
    TcpReadLine = reader.ReadLine()                          ' データの受信
    Exit Function
  End Function

  '=========================================================
  'CALC3 の結果を配列に分割する関数
  '=========================================================
  Sub SplitArrayData(ByVal dataString As String, ByRef dataArray As Double())
    Dim peakNum As Integer
    Dim arrayDataStr As String() = dataString.Split(","c)
                                                             '「,」区切りで文字列を分割する。
    peakNum = arrayDataStr.Length
    For idx As Integer = 0 To arrayDataStr.Length - 1
      dataArray(idx) = Convert.ToDouble(arrayDataStr(idx))
                                                             ' 分割した文字列を数値に変換
    Next
  End Sub
End Module
```

実行例

```
No.       |1              |2              |3              |4              |5              |
REF WL    | 1.30678832E-06| 1.30756981E-06| 1.30835238E-06| 1.30913541E-06| 1.30991969E-06|
REF POWER| -13.4899875    | -9.04694537   | -2.9512995    | -3.29214313   | -13.1556519   |
MAX WL    | ----------     | 1.30757036E-06| 1.3083528E-06 | 1.30913604E-06| ----------    |
MAX POWER| ----------     | -8.81158076   | -0.665845116  | -3.21870974   | ----------    |
MIN WL    | ----------     | 1.30756953E-06| 1.30835221E-06| 1.30913538E-06| ----------    |
MIN POWER| ----------     | -10.2276251   | -3.02598662   | -6.67785905   | ----------    |
```

サンプルプログラム 4

イーサネットにてドリフト解析を行うサンプルプログラム (仕様コード -SW: シングル波長タイプの例です。)

ソースコード

```
Imports System
Imports System.IO
Imports System.Net.Sockets
Imports System.Text

Module EtherDriftMeasure
'Ether にて Drift 解析を行うサンプルプログラム
  Sub Main()
    Try
      Dim wlmAddr As String
      Dim wlmPort As Integer
      Dim sockStream As NetworkStream
      Dim tcpObj As TcpClient
      Dim replyString As String
      Dim peakNum As Integer
      Dim refPowData, refWavData As Double()
      Dim maxPowData, maxWavData As Double()
      Dim minPowData, minWavData As Double()
      Dim dropInfo As Double()
      Dim username, passwd As String

      '=====================================================
      ' 波長計の情報
      '=====================================================
      wlmAddr = "192.168.0.1"                                  ' 波長計の IP アドレス
      wlmPort = 10001                                          ' リモートポート番号
      username = "anonymous"                                   ' ユーザ名
      passwd = ""                                              ' パスワード

      '=====================================================
      ' TCP 接続
      '=====================================================
      tcpObj = New TcpClient
      tcpObj.Connect(wlmAddr, wlmPort)
      sockStream = tcpObj.GetStream()
      tcpObj.NoDelay = True                                    'TCP_NODELAY を有効にする

      '=====================================================
      ' 認証の実行
      '=====================================================
      Dim recvBuffer As String
      TcpWriteLine("open """ + username + """", sockStream) 'OPEN コマンド、ユーザ名の送信
      recvBuffer = TcpReadLine(sockStream)
      If String.Compare(recvBuffer, "AUTHENTICATE CRAM-MD5") <> 0 Then
        sockStream.Dispose()
        Exit Sub                                               ' 応答が「AUTHENTICATE CRAM-MD5」
                                                               ' でなかったらエラー
      End If
      TcpWriteLine(passwd, sockStream)                         ' パスワードの送信
      recvBuffer = TcpReadLine(sockStream)

      If String.Compare(recvBuffer, "ready") <> 0 Then
        sockStream.Dispose()
        Exit Sub                                               ' 認証失敗
      End If
```

```
'=====================================================
' 波長計の測定条件設定
'=====================================================
Call TcpWriteLine("*RST", sockStream)                  ' 本機器のリセット
Call TcpWriteLine(":UNIT:WL NM", sockStream)           ' 波長単位を nm に設定
Call TcpWriteLine(":UNIT:POW DBM", sockStream)         'Power 単位を dBm に設定

' ドリフト測定のリファレンスとするため Single 測定を行う。
Call TcpWriteLine(":INIT;*OPC?", sockStream)           'Single 測定の実行と測定完了待ち
TcpReadLine(sockStream)                                ' 測定完了待ち (*OPC?) の応答を読む
Call TcpWriteLine(":CALC3:DRIF ON", sockStream)        'DRIFT 解析の ON

'=====================================================
' 測定の実行
'=====================================================
Call TcpWriteLine(":INIT:CONT ON", sockStream)         'Repeat 測定の開始

For count As Integer = 1 To 60                         '1 分間待つ
  Threading.Thread.Sleep(1000)
  Console.Write(".")
Next
Console.WriteLine("")
Call TcpWriteLine(":INIT:CONT OFF", sockStream)        'Repeat 測定の停止

'=====================================================
' 測定結果の取得
'=====================================================
Call TcpWriteLine(":CALC3:POIN?", sockStream)          ' データ数の取得
replyString = TcpReadLine(sockStream)
peakNum = Integer.Parse(replyString)
refPowData = New Double(peakNum - 1) {}
refWavData = New Double(peakNum - 1) {}
maxPowData = New Double(peakNum - 1) {}
maxWavData = New Double(peakNum - 1) {}
minPowData = New Double(peakNum - 1) {}
minWavData = New Double(peakNum - 1) {}
dropInfo = New Double(peakNum - 1) {}
' 結果の取得 ( リファレンス値 )
Call TcpWriteLine(":CALC3:DRIF:REF ON", sockStream)
Call TcpWriteLine(":CALC3:DATA? POW", sockStream)      'Ref Power の取得
replyString = TcpReadLine(sockStream)
SplitArrayData(replyString, refPowData)
Call TcpWriteLine(":CALC3:DATA? WAV", sockStream)      'Ref 波長の取得
replyString = TcpReadLine(sockStream)
SplitArrayData(replyString, refWavData)
' 結果の取得 (MAX 値 )
Call TcpWriteLine(":CALC3:DRIF:PRES", sockStream)
Call TcpWriteLine(":CALC3:DRIF:MAX ON", sockStream)
Call TcpWriteLine(":CALC3:DATA? POW", sockStream)      'MAX Power の取得
replyString = TcpReadLine(sockStream)
SplitArrayData(replyString, maxPowData)
Call TcpWriteLine(":CALC3:DATA? WAV", sockStream)      'MAX 波長の取得
replyString = TcpReadLine(sockStream)
SplitArrayData(replyString, maxWavData)
' 結果の取得 (MIN 値 )
Call TcpWriteLine(":CALC3:DRIF:PRES", sockStream)
Call TcpWriteLine(":CALC3:DRIF:MIN ON", sockStream)
Call TcpWriteLine(":CALC3:DATA? POW", sockStream)      'MIN Power の取得
replyString = TcpReadLine(sockStream)
SplitArrayData(replyString, minPowData)
Call TcpWriteLine(":CALC3:DATA? WAV", sockStream)      'MIN 波長の取得
replyString = TcpReadLine(sockStream)
SplitArrayData(replyString, minWavData)
'Drop 情報の取得
Call TcpWriteLine(":CALC3:DATA? DROP", sockStream)
replyString = TcpReadLine(sockStream)
SplitArrayData(replyString, dropInfo)
sockStream.Dispose()                                   'TCP の Close
```

```
    '=====================================================
    ' 測定結果の表示
    '=====================================================
    Console.Write("No.         |")                          ' ピーク番号の表示

    For idx As Integer = 0 To peakNum - 1
      Console.Write((idx + 1).ToString() + "            |")
    Next
    Console.WriteLine()
    Console.Write("REF WL      |")                          ' リファレンス波長の表示
    For idx As Integer = 0 To peakNum - 1
      Console.Write(refWavData(idx).ToString() + " | ")
    Next
    Console.WriteLine()
    Console.Write("REF POWER   |")                          ' リファレンス Power の表示
    For idx As Integer = 0 To peakNum - 1
      Console.Write(refPowData(idx).ToString() + " | ")
    Next
    Console.WriteLine()
    Console.Write("MAX WL      |")                          ' 最大波長の表示
    For idx As Integer = 0 To peakNum - 1
      If dropInfo(idx) <> 0 Then
        Console.Write("---------- | ")
      Else
        Console.Write(maxWavData(idx).ToString() + " | ")
      End If
    Next
    Console.WriteLine()
    Console.Write("MAX POWER   |")                          ' 最大 Power の表示
    For idx As Integer = 0 To peakNum - 1
      If dropInfo(idx) <> 0 Then
        Console.Write("---------- | ")
      Else
        Console.Write(maxPowData(idx).ToString() + " | ")
      End If
    Next
    Console.WriteLine()
    Console.Write("MIN WL      |")                          ' 最小波長の表示
    For idx As Integer = 0 To peakNum - 1
      If dropInfo(idx) <> 0 Then
        Console.Write("---------- | ")
      Else
        Console.Write(minWavData(idx).ToString() + " | ")
      End If
    Next
    Console.WriteLine()
    Console.Write("MIN POWER   |")                          ' 最小 Power の表示
    For idx As Integer = 0 To peakNum - 1
      If dropInfo(idx) <> 0 Then
        Console.Write("---------- | ")
      Else
        Console.Write(minPowData(idx).ToString() + " | ")
      End If
    Next
    Console.WriteLine()
    Console.ReadLine()
  Catch ex As Exception                                     ' 例外 ( エラー ) 処理
    Console.WriteLine(ex.Message)                           ' 発生したエラーのメッセージを表示
    Console.ReadLine()                                      ' 終了の Enter キー入力待ち
  End Try
End Sub
```

```
'=========================================================
'TCP Socket に文字列を送信する関数
'=========================================================
Sub TcpWriteLine(ByVal commandStr As String, ByRef stream As NetworkStream)
  Dim writer As StreamWriter = New StreamWriter(stream, Encoding.ASCII)
  Dim ByteLf As Byte() = New Byte() {10}
  writer.NewLine = Encoding.ASCII.GetString(ByteLf)        ' 改行コードは LF
  writer.AutoFlush = True
  writer.WriteLine(commandStr)                             ' データの送信
End Sub

'=========================================================
'TCP Socket からデータを 1 行読み込む関数
'=========================================================
Function TcpReadLine(ByRef stream As NetworkStream) As String
  Dim reader As StreamReader = New StreamReader(stream, Encoding.ASCII)
  TcpReadLine = reader.ReadLine()                          ' データの受信
  Exit Function
End Function

'=========================================================
'CALC3 の結果を配列に分割する関数
'=========================================================
Sub SplitArrayData(ByVal dataString As String, ByRef dataArray As Double())
  Dim peakNum As Integer
  Dim arrayDataStr As String() = dataString.Split(","c)
                                                           '「,」区切りで文字列を分割する。
  peakNum = arrayDataStr.Length
  For idx As Integer = 0 To arrayDataStr.Length - 1
    dataArray(idx) = Convert.ToDouble(arrayDataStr(idx))
                                                           ' 分割した文字列を数値に変換
  Next
 End Sub
End Module
```

実行例

```
No.       |1              |
REF WL    | 1.55252293E-06|
REF POWER | 3.37379314    |
MAX WL    | 1.55252431E-06|
MAX POWER | 3.43271086    |
MIN WL    | 1.55252289E-06|
MIN POWER | 3.3485737     |
```

4.1　ステータスレジスタについて

本機器は、下表のステータスレジスタを備えています。ステータスレジスタの全体図を次ページに記載します。
本機器は、IEEE488.2 および SCPI で規定された下記のレジスタを備えています。

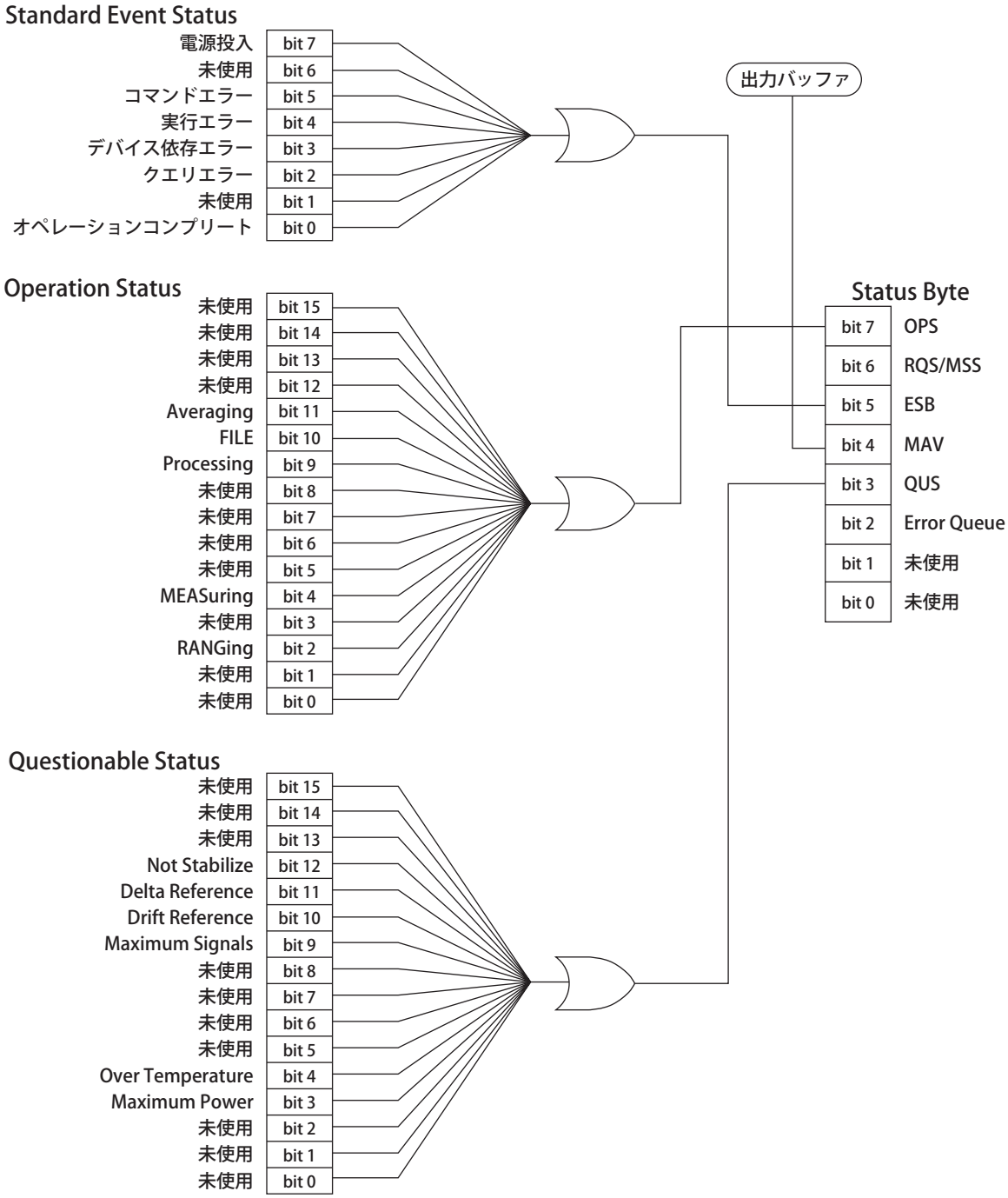
- ステータスバイトレジスタ
- スタンダードイベントステータスレジスタ
- オペレーションステータスレジスタ
- クエッショナブルステータスレジスタ

また、各レジスタのサマリ情報として、オペレーションステータスビット (OPS) とクエッショナブルステータスビット (QUS) を、ステータスバイトレジスタの拡張ビットに割り当てています。

ステータスレジスター一覧

レジスタ名	内容
ステータスバイトレジスタ	IEEE488.2 で規定されたレジスタ
STB: Status Byte Register	同上
SRE: Service Request Enable Register	同上
スタンダードイベントステータスレジスタ	IEEE488.2 で規定されたレジスタ
ESR: Standard Event Status Register	同上
ESE: Standard Event Status Register	同上
オペレーションステータスレジスタ	動作の実行情報を提供 (測定中、平均化処理中、等)
Operation Event Register	イベントの有無を表すレジスタ。イベントはラッチされる
Operation Event Enable Register	サマリビット (OPS) 生成時の条件マスク用レジスタ
クエッショナブルステータスレジスタ	本機器の動作状態情報を提供
Questionable Event Register	イベントの有無を表すレジスタ。イベントはラッチされる
Questionable Event Enable Register	サマリビット (QUS) 生成時の条件マスク用レジスタ

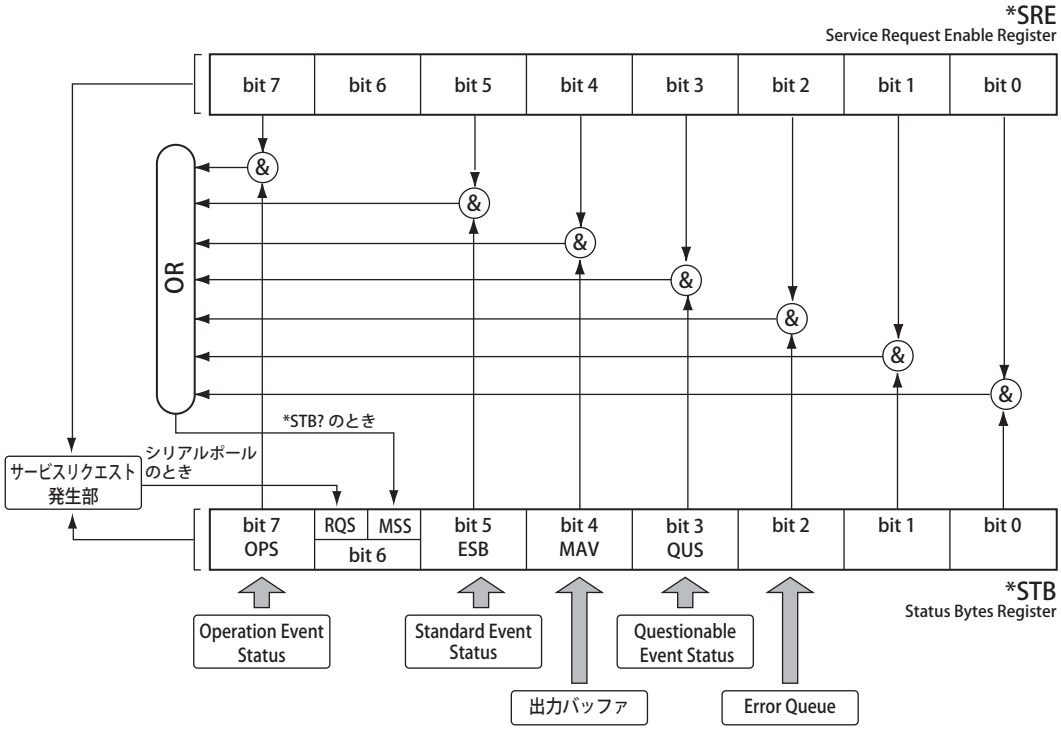
ステータスレジスタ全体図



4.2 ステータスバイトレジスタ

構成

以下にステータスバイトレジスタの構成を示します。本レジスタの内容や動作は、IEEE488.2 に準じています。
また、本機器では、OPS ビットと QUS ビットを拡張しています。



Status Byte Register の内容

Bit	イベント名	説明	Decimal value
Bit 7	OPS	オペレーションステータスのサマリビット	128
Bit 6	RQS, MSS	1 つ以上のサービス要求があるときに "1"	64
Bit 5	ESB	Standard Event Status Register のサマリビット	32
Bit 4	MAV	出力バッファにデータが存在するときに "1"	16
Bit 3	QUS	クエッショナブルステータスのサマリビット	8
Bit 2	Error Queue	エラーが存在するときに "1"	4
Bit 1	None	未使用 (常に 0)	0
Bit 0	None	未使用 (常に 0)	0

ステータスバイトレジスタ

読み取り

シリアルポールや *STB? 共通クエリで読み取ることができます。ただし、読み取り方法の違いにより、bit 6 の情報が変わります。

- シリアルポールで読み取った場合
  RQS メッセージが bit 6 の情報として読み取られます。
  読み取り後に RQS メッセージはクリアされます。
- *STB? 共通クエリで読み取った場合
  MSS サマリメッセージが bit 6 の情報として読み取られます。
  読み取り後も MSS メッセージは変化しません。

bit 6 以外の内容は変化しません。
読み取り動作は、IEEE488.2 規格に準じます。

書き込み

割り当てられたステータスデータ構造の状態が変化したときにだけ、書き換えられます。
書き込み動作は、IEEE488.2 規格に準じます。

クリア

*CLS 共通コマンドにより、出力キューと MAV ビットを除くすべてのイベントレジスタとキューがクリアされます。
クリア動作は、IEEE488.2 規格に準じます。

サービスリクエストイネーブルレジスタ

読み取り

*SRE? 共通クエリで読み取ることができます。
読み取っても内容はクリアされません。
読み取り動作は、IEEE488.2 規格に準じます。

書き込み

*SRE 共通コマンドで書き込むことができます。
また、未使用ビットの bit 6 の設定値は常に無視されます。
書き込み動作は、IEEE488.2 規格に準じます。

クリア

次の条件でクリアされます。

- *SRE 共通コマンドでデータ "0" をセットする
- 電源投入

本レジスタは、以下のときはクリアされません。

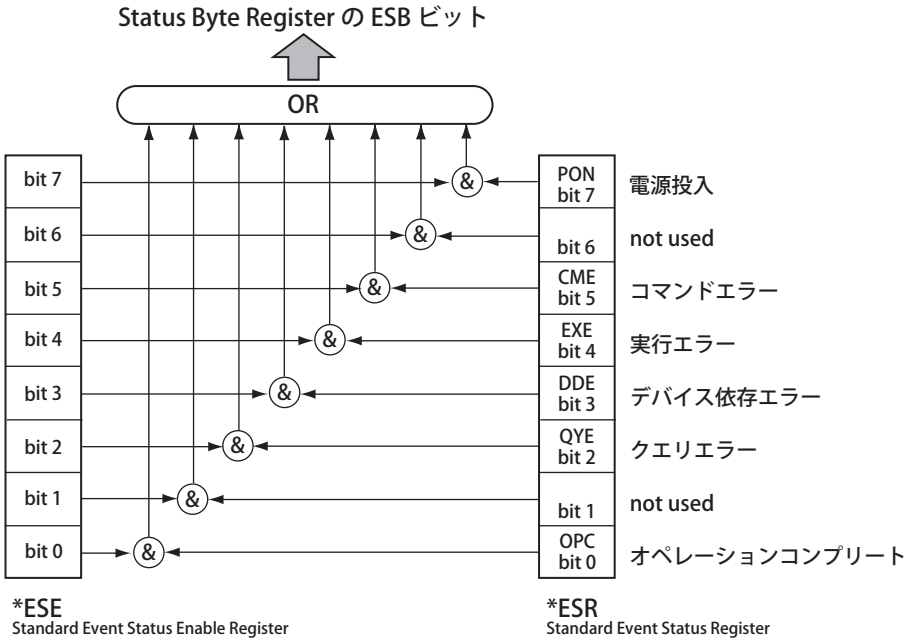
- *RST コマンドの受信
- *CLS コマンドの受信
- デバイスクリア (DCL, SDC)

クリア動作は、IEEE488.2 規格に準じます。

4.3 スタンダードイベントステータスレジスタ

構成

以下にスタンダードイベントステータスレジスタの構成を示します。本レジスタの内容や動作は、IEEE488.2 に準じています。



Standard Event Status Register の内容

Bit	イベント名	説明	Decimal value	備考
Bit 7	PON(パワー ON)	電源が OFF → ON へ遷移起動時に "1" になる	128	
Bit 6	None	未使用 (常に 0)	0	
Bit 5	CME (コマンドエラー)	構文エラーを検出・認識不可能なコマンドを検出 プログラムメッセージの 1 番目のバイトとプログラムメッセージターミネータの間で GET に遭遇した	32	
Bit 4	EXE (実行エラー)	プログラムヘッダーに続くプログラムデータが有効範囲外 デバイスのステートに矛盾したプログラムメッセージを受信	16	メッセージ No.200 で "1" になります
Bit 3	DDE	CME、EXE、QYE 以外の原因によるエラー (デバイス固有エラー)	8	メッセージ No.70 番代 (測定処理エラー)、80 番台 (ハードの動作不良) で "1" になります
Bit 2	QYE (クエリエラー)	出力が存在しない状態で出力キューにアクセスした出力キューデータが失われた	4	メッセージ No.410、440 で "1" になります
Bit 1	None	未使用 (常に 0)	0	
Bit 0	OPC (オペレーションコンプリート)	コマンド動作完了 *OPC 時のみ有効 *OPC? の場合には無効	1	コマンド動作完了のタイミングについては 4-8 ページをご覧ください。

Note

メッセージ一覧の詳細はユーザーズマニュアル IM AQ6150B-02JA の 4.1 節をご覧ください。

スタンダードイベントステータスレジスタ

読み取り

*ESR? 共通クエリで読み取ることができます。
読み取りにより、レジスタの内容はクリアされます。
読み取り動作は、IEEE488.2 規格に準じます。

書き込み

レジスタの内容をクリアできます。クリア以外、書き込むことはできません。

クリア

次の条件でクリアされます。

- *CLS 共通コマンド
- *ESR? 共通クエリ

クリア動作は、IEEE488.2 規格に準じます。

スタンダードイベントステータスイネーブルレジスタ

読み取り

*ESE? 共通クエリで読み取ることができます。
読み取り動作は、IEEE488.2 規格に準じます。

書き込み

*ESE 共通コマンドで書き込むことができます。
書き込み動作は、IEEE488.2 規格に準じます。

クリア

次の条件でクリアされます。

- *ESE 共通コマンドでデータ "0" をセットする
- 電源投入

以下のときにはクリアされません。

- *RST コマンドの受信
- *CLS コマンドの受信
- デバイスクリア (DCL, SDC)

クリア動作は、IEEE488.2 規格に準じます。

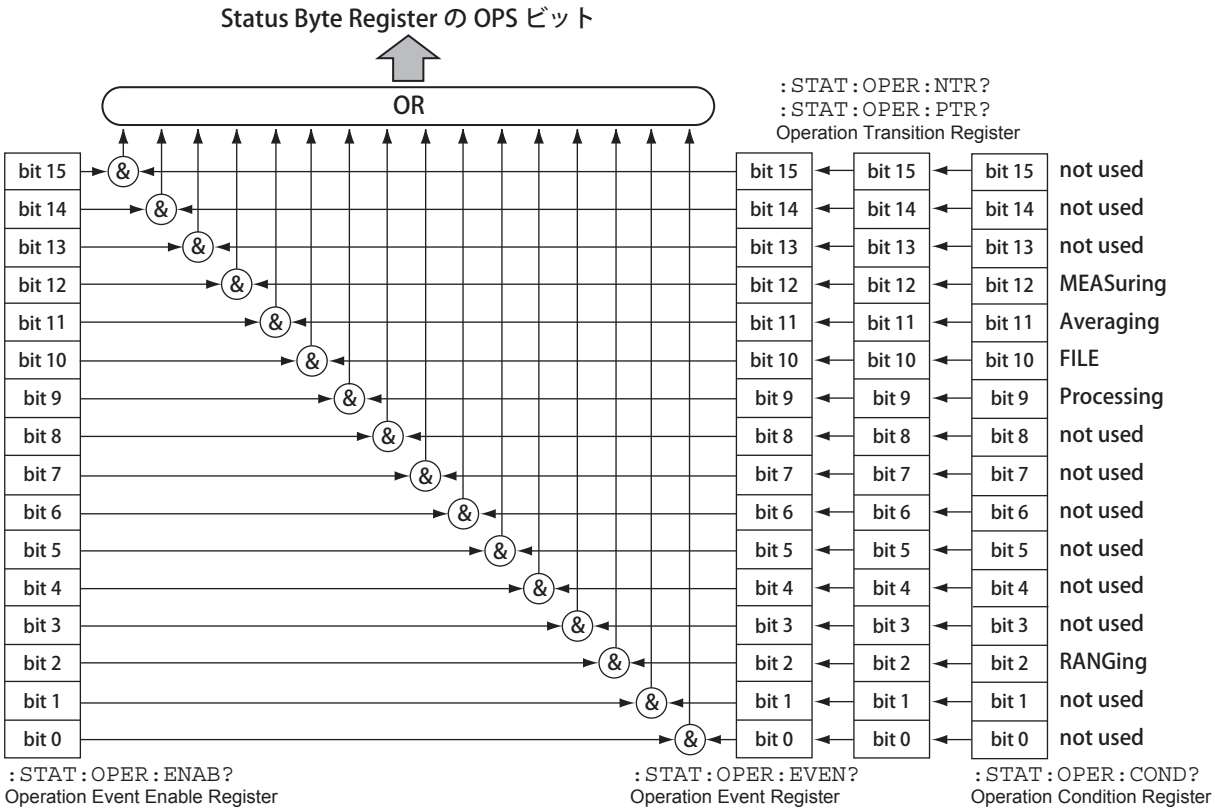
4.4 オペレーションステータスレジスタ

オペレーションステータスレジスタは、本機器の動作ステータスをレポートします。
本機器の状態はオペレーションコンディションレジスタで示されます。
オペレーションコンディションレジスタでの変化は、オペレーションイベントレジスタに反映されます。オペレーションステータスレジスタを参照することで、動作ステータスの変化をとらえられます。
また、オペレーションイベントレジスタのサマリ情報が、ステータスバイトレジスタの OPS ビットにセットされます。ここでは、オペレーションイベントイネーブルレジスタが "1" に指定されたビットに対応するステータスだけがサマリ情報に含まれます。

構成

以下にオペレーションステータスレジスタの構成を示します。

Operation Status Register の構造



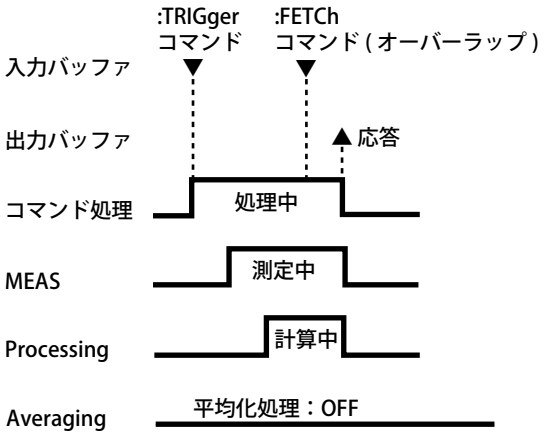
Operation Status Register の内容

Bit	イベント名	説明	Decimal value	備考
Bit 15	Not used	予備 (常に 0)	0	
Bit 14	Not used	予備 (常に 0)	0	
Bit 13	Not used	予備 (常に 0)	0	
Bit 12	Not used	予備 (常に 0)	0	
Bit 11	Averaging	平均化処理中	2048	平均化回数が 1 より大きい設定をすると "1" になります
Bit 10	FILE	ファイル操作中	1024	ファイル操作 (Read/Write/Copy/Delete/Rename) 中で "1" になります。
Bit 9	Processing	計算中	512	計算中で "1" になります
Bit 8	Not used	予備 (常に 0)	0	
Bit 7	Not used	予備 (常に 0)	0	
Bit 6	Not used	予備 (常に 0)	0	
Bit 5	Not used	予備 (常に 0)	0	
Bit 4	MEASuring	測定中	16	測定中で "1" になります
Bit 3	Not used	予備 (常に 0)	0	
Bit 2	RANGing	レンジ切替中	4	アンダーレンジまたはオーバーレンジで "1" になります
Bit 1	Not used	予備 (常に 0)	0	
Bit 0	Not used	予備 (常に 0)	0	

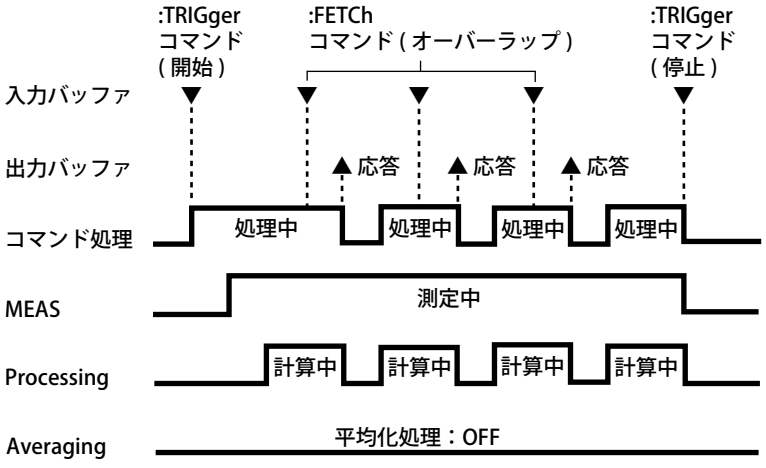
動作ステータス変化の例

測定開始のコマンドを受けたときの各ステータスの変化と、測定中に測定結果の問い合わせを受けたときの応答を返すタイミングを以下に示します。

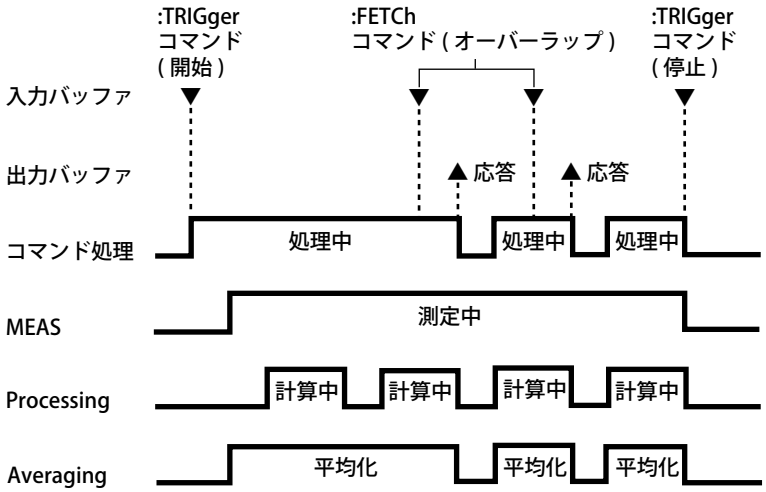
- Single 測定



- Repeat 測定



・ 平均化測定 (Repeat 測定時、平均化の回数が 2 のとき)



入力バッファにコマンドを受信すると、コマンド処理状態となります。コマンド処理状態は、測定、計算、平均化の処理がすべて完了するまで継続します。この間はオーバーラップコマンドだけを実行できます。図の例では、測定結果を問い合わせるコマンドをオーバーラップ動作をさせて、コマンド処理が完了した時点で応答 (測定結果) を返しています。Repeat 測定と平均化測定では、測定中は継続して MEAS が 1 になります。
Processing はその都度計算するときだけ 1 になります。
Averaging は 1 回目の平均化処理時は、平均化の回数だけ処理する間 1 になります。
2 回目以降は、既に測定した値と新しく測定した値を平均化するため、Processing と同じタイミングになります。*OPC、*OPC? コマンドは、コマンド処理の状態を参照します。

オペレーションコンディションレジスタ

読み取り

:STATus:OPERation:CONDition? クエリ・コマンドで、読み取ることができます。
読み取りを行っても、レジスタの内容はクリアされません。

書き込み

本レジスタは、本機器の状態が変化した場合にだけ、その状態の変化に対応したビットがセット / リセットされます。
書き込むことはできません。

クリア

クリアはできません。

オペレーションイベントレジスタ

読み取り

:STATus:OPERation[:EVENt]? クエリ・コマンドで、読み取ることができます。
読み取りにより、レジスタの内容はクリアされます。

書き込み

レジスタの内容をクリアできます。クリア以外、書き込むことはできません。
<クリア>
次の条件でクリアされます。
・ :STATus:OPERation[:EVENt]? クエリ・コマンドによる読み取り時
・ :STATus:PRESet コマンドによる初期化時
・ *CLS 共通コマンド
・ 電源投入

オペレーションイベントイネーブルレジスタ

読み取り

:STATus:OPERation:ENABle? クエリ・コマンドで、読み取ることができます。

書き込み

:STATus:OPERation:ENABle コマンドで、書き込むことができます。

クリア

次の条件でクリアされます。

- :STATus:OPERation:ENABle コマンドでデータ "0" をセット
- 電源投入

以下のときにはクリアされません。

- *RST コマンドの受信
- *CLS コマンドの受信
- デバイスクリア (DCL, SDC)

オペレーションポジティブトランジションフィルタ

読み取り

:STATus:OPERation:PTRansition? クエリ・コマンドで、読み取ることができます。

書込み

:STATus:OPERation:PTRansition コマンドで、書き込むことができます。

クリア

次の条件でクリアされます。

- :STATus:OPERation:PTRansition コマンドでデータ "0" をセット
- 電源投入

以下のときにはクリアされません。

- *RST コマンドの受信
- *CLS コマンドの受信
- デバイスクリア (DCL, SDC)

オペレーションネガティブトランジションフィルタ

読み取り

:STATus:OPERation:NTRansition? クエリ・コマンドで、読み取ることができます。

書込み

:STATus:OPERation:NTRansition コマンドで、書き込むことができます。

クリア

次の条件でクリアされます。

- :STATus:OPERation:NTRansition コマンドでデータ "0" をセット
- 電源投入

以下のときにはクリアされません。

- *RST コマンドの受信
- *CLS コマンドの受信
- デバイスクリア (DCL, SDC)

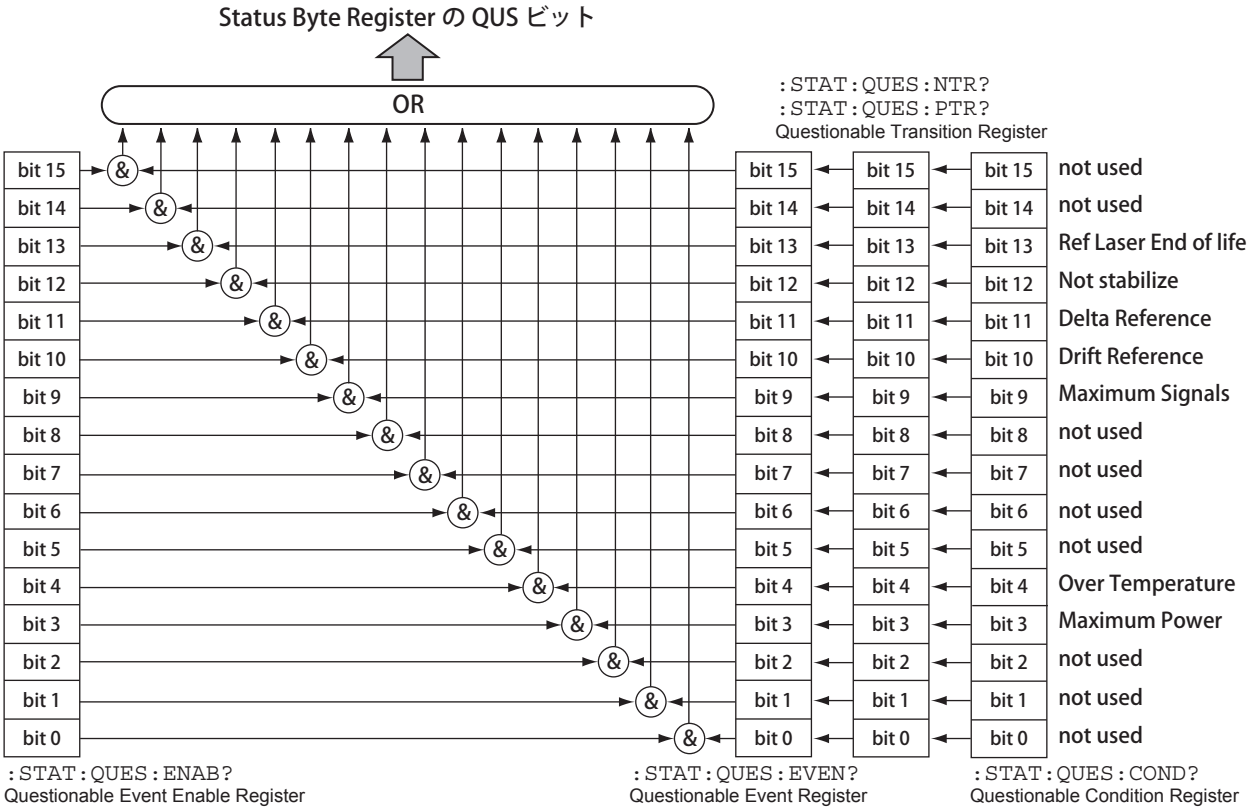
4.5 クエッショナブルステータスレジスタ

クエッショナブルステータスレジスタは、本機器のクエッショナブルステータスをレポートします。
イベントレジスタのサマリ情報は、ステータスバイトレジスタの QUS ビットにセットされます。

構成

以下にクエッショナブルステータスレジスタの構成と内容を示します。

クエッショナブルステータスレジスタの構成



クエッショナブルステータスの内容

Bit	イベント名	説明	Decimal value	備考
Bit 15	Not used	予備 (常に 0)	0	
Bit 14	Not used	予備 (常に 0)	0	
Bit 13	Ref Laser End of Life	基準光源の寿命	8192	基準光源の寿命を検出すると "1" になります
Bit 12	Not stabilize	基準光源の状態	4096	基準光源が安定していないときに "1" になります
Bit 11	Delta Reference	リファレンスの消滅	2048	デルタ測定時に、リファレンスのピークが消滅すると "1" になります。仕様コード -SW：シングル波長タイプでは常に "0"。
Bit 10	Drift Reference	ピークの数が異なる	1024	ドリフト測定時に、リファレンスのピークの数と測定時のピークの数が異なると "1" になります
Bit 9	Maximum Signals	ピークの最大検出数 (1024) を超えている	512	ピークの最大検出数を超えると "1" になります。仕様コード -SW：シングル波長タイプでは常に "0"。
Bit 8	Not used	予備 (常に 0)	0	
Bit 7	Not used	予備 (常に 0)	0	
Bit 6	Not used	予備 (常に 0)	0	
Bit 5	Not used	予備 (常に 0)	0	
Bit 4	Over Temperature	温度上昇エラー	16	本機器内部の温度が異常に上昇したときに "1" になります
Bit 3	Maximum Power	光入力パワーオーバー	8	光入力のパワーが許容パワーを超えているときに "1" になります
Bit 2	Not used	予備 (常に 0)	0	
Bit 1	Not used	予備 (常に 0)	0	
Bit 0	Not used	予備 (常に 0)	0	

クエッショナブルコンディションレジスタ

読み取り

:STATus:QUEStionable:CONDition? クエリ・コマンドで、読み取ることができます。読み取りを行っても、レジスタの内容はクリアされません。

書き込み

本レジスタは、本機器の状態が変化した場合だけ、その状態の変化に対応したビットがセット / リセットされます。書き込むことはできません。

クリア

クリアはできません。

クエッショナブルイベントレジスタ

読み取り

:STATus:QUEStionable[:EVENt]? クエリ・コマンドで、読み取ることができます。
読み取りにより、レジスタの内容はクリアされます。

書き込み

レジスタの内容をクリアできます。クリア以外、書き込むことはできません。

クリア

次の条件でクリアされます。

- :STATus:QUEStionable[:EVENt]? クエリ・コマンドによる読み取り時
- :STATus:PRESet コマンドによる初期化時
- *CLS 共通コマンド
- 電源投入

クエッショナブルイベントイネーブルレジスタ

読み取り

:STATus:QUEStionable:ENABle? クエリ・コマンドで、読み取ることができます。

書き込み

:STATus:QUEStionable:ENABle コマンドで、書き込むことができます。

クリア

次の条件でクリアされます。

- :STATus:QUTStionable:ENABle コマンドでデータ "0" をセット
- 電源投入

以下のときはクリアされません。

- *RST コマンドの受信
- *CLS コマンドの受信
- デバイスクリア (DCL, SDC)

クエッショナブルポジティブトランジションフィルタ

読み取り

:STATus:QUEStionable:PTRansition? クエリ・コマンドで、読み取ることができます。

書き込み

:STATus:QUEStionable:PTRansition コマンドで、書き込むことができます。

クリア

次の条件でクリアされます。

- :STATus:QUEStionable:PTRansition コマンドでデータ "0" をセット
- 電源投入

以下のときはクリアされません。

- *RST コマンドの受信
- *CLS コマンドの受信
- デバイスクリア (DCL, SDC)

クエッショナブルネガティブトランジションフィルタ

読み取り

:STATus:QUEStionable:NTRansition? クエリ・コマンドで、読み取ることができます。

書き込み

:STATus:QUEStionable:NTRansition コマンドで、書き込むことができます。

クリア

次の条件でクリアされます。

- :STATus:QUEStionable:NTRansition コマンドでデータ "0" をセット
- 電源投入

以下のときはクリアされません。

- *RST コマンドの受信
- *CLS コマンドの受信
- デバイスクリア (DCL, SDC)

5.1　シンタックス記述の規則とコマンドの種類

以下の内容は、本書に記載する共通コマンドおよび機器固有コマンドを対象としています。
測定値とパラメータは、特殊なコマンドを除き、すべて ASCII 文字列で送受信します。

シンタックス記述の規則

規則	説明
|	リスト中の要素をどれか 1 つ選ぶことを示す。 例： A|B|C= A か B か C のいずれか 1 つ
[　]	括弧内のアイテムは任意指定
{　}	括弧内のアイテムはコマンド中に複数回指定可能
<wsp> *	ホワイト・スペース
<integer>	整数値
<NRf>	指数表記値、整数値または小数表記値
<"file name">	ファイル名はディレクトリ部分を除いて、拡張子込みで最大 56 文字。 ダブルクォーテーション ("") で文字列を囲う
<"string">	文字列 ダブルクォーテーション ("") で文字列を囲う

*　ホワイト・スペース (<wsp>) について
ASCII 文字セットの 00h ～ 20h (0Ah (LF) を除く) に対応する文字をホワイト・スペースと定義します。
ホワイト・スペースは、パラメータを指定する場合のコマンドとパラメータの間に入れるとき、およびパラメータにおけるファイル名等の文字列中のスペースを除いて任意に指定でき、プログラムを読みやすくするために使用できます。

コマンドの種類

本機器のコマンドには、以下の 3 種類があります。被オーバーラップコマンドとオーバーラップコマンドについては、5.4 節、5.5 節の解説に明記しています。

シーケンシャルコマンド

- 当コマンドの動作が完了するまで、他のコマンドの動作を実行しません。
- 他のコマンド動作が完了するまで、当コマンドの動作を実行しません。

被オーバーラップコマンド

- 当コマンドの動作が完了する前に、他のオーバーラップコマンドの動作を実行します。
- 当コマンドの動作が完了するまで、シーケンシャルコマンドの動作を実行しません。
- 他のコマンド動作が完了するまで、当コマンドの動作を実行しません。

オーバーラップコマンド

- 被オーバーラップコマンドの動作が完了する前に、当コマンドの動作が実行できます。
- 当コマンドの動作が完了するまで、他のコマンドの動作を実行しません。
- シーケンシャルコマンドの動作が完了するまで、当コマンドの動作を実行しません。

複数のコマンドの一括送信

「5.4 共通コマンド」および「5.5 機器固有コマンド」に記載のコマンドを使用して、コマンド文字列を作成し、本機器に送信します。
セミコロン ";" で各コマンドを区切り、単一の出力ステートメントに複数のコマンドを記述した場合、コマンドは記述順に実行されます。

リモートコマンドの書式について

短形式と長形式

本機器のリモートコマンドは、短形式、長形式の両方に対応しています。
本書に記載のコマンドにおいて、大文字で記載されている部分は当該コマンドの短形式です。
`INITiate` コマンドの短形式は `INIT`、長形式は `INITIATE` です。

大文字と小文字

本機器では、大文字と小文字の区別をしません。
読み取り値はすべて大文字で記述します。

数値

- 本機器が受信する場合には、複数の表記方法に対応しています。
- 本機器が送信する場合には、基本単位のみ使用します。

実数部の桁数は、整数部 1 桁 (符号付き)、小数点以下 8 桁固定。
指数部の桁数は、 3 桁固定。

例： 受信可能数値 (1550nm の場合)
1550nm、1.55um、1550E-9、1.55E-6 など
例： 送信数値 (1550nm の場合)
+1.55000000E-006 のみ

- 受け取った数値が内部で取り扱う数値範囲より高い精度の場合は、下位の切り捨てではなく四捨五入を行います。
- 本機器が対応する乗数サフィックスは以下のとおりです。

乗数	ニーモニック	乗数	ニーモニック
1E18	EX (エクサ)	1E-3	M (ミリ)
1E15	PE (ペタ)	1E-6	U (マイクロ)
1E12	T (テラ)	1E-9	N (ナノ)
1E9	G (ギガ)	1E-12	P (ピコ)
1E6	MA (メガ)	1E-15	F (フェムト)
1E3	K (キロ)	1E-18	A (アト)

コマンド中のパラメータ指定

コマンドの中にパラメータを使用する場合、コマンドとパラメータの間にはスペースを入れる必要があります。
パラメータとパラメータの間は、カンマ "," で区切ります。コマンドを読みやすくするために、カンマの前後にスペースを入れることもできます。

5.2 ソフトキーとリモートコマンドの対応表

本機器を操作するときに使用するソフトキーに対応するリモートコマンドを示します。各コマンドのパラメータについては、5.3 節または 5.5 節をご覧ください。
パラメータの詳細な説明や解説については 5.5 節をご覧ください。

SETUP

ソフトキー	リモートコマンド	備考
DEVICE TYPE	[:SENSe]:CORRection:DEVice	
PEAK THRESH TYPE*1	:CALCulate2:PTHReshold:MODe	
PEAK THRESH VALUE*1	:CALCulate2:PTHReshold[:RELative]	相対値
	:CALCulate2:PTHReshold:ABSolute	絶対値
PEAK EXCURSION	:CALCulate2:PEXCursion	
WAVELENGTH LIMIT		
LIMITTING MODE	:CALCulate2:WLIMit[:STATe]	
LIMIT START WL	:CALCulate2:WLIMit:STARt:FREQuency	周波数
	:CALCulate2:WLIMit:STARt[:WAVelength]	波長
	:CALCulate2:WLIMit:STARt:WNUMber	波数
LIMIT STOP WL	:CALCulate2:WLIMit:STOP:FREQuency	周波数
	:CALCulate2:WLIMit:STOP[:WAVelength]	波長
	:CALCulate2:WLIMit:STOP:WNUMber	波数
SET PRESET LIMITS	—	
AVERAGE TIMES	:CALCulate2:COUNt	
WAVELENGTH UNIT	:UNIT:WL	
POWER UNIT	:UNIT[:POWer]	
MEAS WL	[:SENSe]:CORRection:MEDium	
CH MATCHING THRESH FREQ*1	:CALCulate2:MTHResh	
UPDATE RATE	[:SENSe]:URATe	

*1　仕様コード -MW：マルチ波長タイプ用

SYSTEM

ソフトキー	リモートコマンド	備考
REMOTE SETTING		
REPLY FOR NO SIGNAL	:FORMat:NDATa[:WAVelength]	
LANGUAGE	:SYSTem:LANGuage	
POWER OFFSET	[:SENSe]:CORRection:OFFSet[:MAGNitude]	
PARAMETER INITIALIZE		
MEAS PARAM CLEAR	:SYSTem:PRESet	
BUZZER		
CLICK	:SYSTem:BUZZer[:CLICk]	
WARNING	:SYSTem:BUZZer:WARNing	
SET CLOCK	:SYSTem:DATE	日付
	:SYSTem:TIME	時刻
COLOR MODE	:DISPlay:COLor	

DISPLAY

ソフトキー	リモートコマンド	備考
VIEW MODE	:CONFigure[:SCALar]:POWer	SINGLE-WL
	:CONFigure:ARRay:POWer*1	MULTI-WL
	:CALCulate3:DELTa:WPOWer[:STATe]*1	DELTA-WL
	:CALCulate3:GRID[:STATe]	GRID
GRID PARAMETER		
START WL	:CALCulate3:GRID:STARt[:WAVelength]	
START FREQ	:CALCulate3:GRID:STARt:FREQuency	
START WNUM	:CALCulate3:GRID:STARt:WNUMber	
STOP WL	:CALCulate3:GRID:STOP[:WAVelength]	
STOP FREQ	:CALCulate3:GRID:STOP:FREQuency	
STOP WNUM	:CALCulate3:GRID:STOP:WNUMber	
SEARCH AREA	:CALCulate3:GRID:SAREa:FREQuency	
REF FREQ	:CALCulate3:GRID:REFerence:FREQuency	
SPACING	:CALCulate3:GRID:SPACing:FREQuency	
SHOW ALL*1	:CALCulate3:GRID:DISPlay:ALL	
PREV PEAK*1	—	
NEXT PEAK*1	—	
LIST ONLY*1	:DISPlay[:WINDow]:STATe	
SPECTRUM DISPLAY*1	:DISPlay:WINDow2:STATe	
AUTO SCALE*1	:DISPlay:WINDow2:TRACe[:SCALe]:ASCale	
SCALE*1		
CENTER WL	:DISPlay:WINDow2:TRACe[:SCALe]:CENTer[:WAVelength]	中心波長
CENTER FREQ	:DISPlay:WINDow2:TRACe[:SCALe]:CENTer:FREQuency	中心周波数
CENTER WNUM	:DISPlay:WINDow2:TRACe[:SCALe]:CENTer:WNUMber	中心波数
SPAN WL	:DISPlay:WINDow2:TRACe[:SCALe]:SPAN[:WAVelength]	波長スパン
SPAN FREQ	:DISPlay:WINDow2:TRACe[:SCALe]:SPAN:FREQuency	周波数スパン
SPAN WNUM	:DISPlay:WINDow2:TRACe[:SCALe]:SPAN:WNUMber	波数スパン
START WL	:DISPlay:WINDow2:TRACe[:SCALe]:LEFT[:WAVelength]	開始波長
START FREQ	:DISPlay:WINDow2:TRACe[:SCALe]:LEFT:FREQuency	開始周波数
START WNUM	:DISPlay:WINDow2:TRACe[:SCALe]:LEFT:WNUMber	開始波数
STOP WL	:DISPlay:WINDow2:TRACe[:SCALe]:RIGHt[:WAVelength]	終了波長
STOP FREQ	:DISPlay:WINDow2:TRACe[:SCALe]:RIGHt:FREQuency	終了周波数
STOP WNUM	:DISPlay:WINDow2:TRACe[:SCALe]:RIGHt:WNUMber	終了波数
PEAK CENTER	:DISPlay:WINDow2:TRACe[:SCALe]:CENTer:PEAK	
INITIAL	:DISPlay:WINDow2:TRACe[:SCALe]:INITialize	
LIST BY*1	:CONFigure:ARRay:POWer	パワー順にソート
	:CONFigure:ARRay:POWer:WAVelength	波長順にソート
DIGIT	:DISPlay:RESolution[:WAVelength]	
	:DISPlay:RESolution:FREQuency	
	:DISPlay:RESolution:WNUMber	
OVER VIEW DISPLAY*1	—	
LABEL	:DISPlay[:WINDow]:TEXT:DATA	
DISPLAY OFF	:DISPlay[:WINDow]	

*1 仕様コード -MW：マルチ波長タイプ用

SEARCH(仕様コード -MW: マルチ波長タイプ用)

ソフトキー	リモートコマンド	備考
PEAK	:DISPlay:MARKer:MAXimum	
NEXT POWER	:DISPlay:MARKer:MAXimum:NEXT	
PREV POWER	:DISPlay:MARKer:MAXimum:PREVious	
NEXT WL	:DISPlay:MARKer:MAXimum:RIGHt	
PREV WL	:DISPlay:MARKer:MAXimum:LEFT	
LIST BY	—	
AUTO PEAK SEARCH	:CALCulate2:ASEarch	

ANALYSIS

ソフトキー	リモートコマンド	備考
FABRY-PEROT LASER*1	:CALCulate3:FPERot[:STATe]	
DRIFT MEASUREMENT/ PARAMETER SETTING	:CALCulate3:DRIFt[:STATe]	
REF SET	:CALCulate3:DRIFt:REFerence:RESEt	
DISPLAY MODE		
DELTA	:CALCulate3:DRIFt:PRESet	
MAX	:CALCulate3:DRIFt:MAXimum[:STATe]	
MIN	:CALCulate3:DRIFt:MINimum[:STATe]	
MAX-MIN	:CALCulate3:DRIFt:DIFFerence[:STATe]	
WAVELENGTH	:CALCulate3:DRIFt:WAVelength[:STATe]	
POWER	:CALCulate3:DRIFt:POWer[:STATe]	
DATA LOGGING	:CALCulate3:DLOGging:STATe	
LOGGING	:CALCulate3:DLOGging:MEASure:STATe	
CURSOR/SCALE	—	
SETUP		
LOGGING PARAMETER	:CALCulate3:DLOGging:LPARameter:ITEM	ロギングアイテム
	:CALCulate3:DLOGging:LPARameter:LMODe	ロギングモード
	:CALCulate3:DLOGging:LPARameter:INTerval	ロギング間隔
	:CALCulate3:DLOGging:LPARameter:TDURation	ロギング時間
	:CALCulate3:DLOGging:LPARameter:ASAVe[:STATe]	データ自動保存
GRAPH ITEM	—	
GRAPH CHANNEL	—	
CURSOR DATA FORMAT	—	
DATA DISPLAY	—	
DATA VIEW	—	
LOGGING DATA CLEAR	—	
SPECTRUM DISPLAY	—	
FILE	—	
WDM(OSNR)	:CALCulate3:SNR[:STATe]	
NOISE ALGO		
AUTO-CTR	:CALCulate3:SNR:AUTO	ON のとき
MANUAL-FIX	:CALCulate3:SNR:AUTO	OFF のとき
NOISE AREA	:CALCulate3:SNR:REFerence[:WAVelength]:RELative	
NOISE BW	:CALCulate3:SNR:REFerence:BWIDth	

*1 仕様コード -MW：マルチ波長タイプ用

FILE

ソフトキー	リモートコマンド	備考
WRITE		
MEMORY	:MMEMory:CDRive	
MAKE DIRECTORY	:MMEMory:MDIRectory	
FILE SORT	—	
EXECUTE	:MMEMory:STORe	
READ		
MEMORY	:MMEMory:CDRive	
FILE SORT	—	
EXECUTE	:MMEMory:LOAD	
ITEM SELECT	—	WRITE の EXECUTE 時に指定
REMOVE USB STORAGE	:MMEMory:REMove	
FILE OPERATION		
MEMORY	:MMEMory:CDRive	
DELETE	:MMEMory:DELete	
COPY	:MMEMory:COPY	
RENAME	:MMEMory:REName	
MAKE DIRECTORY	:MMEMory:MDIRectory	
FILE SELECT	—	他の各コマンドにファイル名の指定あり

5.3 リモートコマンドツリー

各コマンドに対するパラメータを示します。各コマンドに対応するソフトキーについては 5.2 節をご覧ください。パラメータの詳細な説明や解説については 5.4 節、5.5 節をご覧ください。

共通コマンド

コマンド	パラメータ	参照ページ
*CLS	none	5-13
*ESE	<integer>	5-13
*ESR?	none	5-13
*IDN?	none	5-13
*OPC	none	5-13
*RCL	1|2|3|4	5-13
*RST	none	5-13
*SAV	1|2|3|4	5-13
*SRE	<integer>	5-13
*STB?	none	5-14
*TRG	none	5-14
*TST?	none	5-14
*WAI	none	5-14

CALCulate2

コマンド	パラメータ	参照ページ
:CALCulate2		
:ASEarch[*1]	ON|OFF|0|1	5-15
:COUNt	<integer>|MINimum|MAXimum	5-15
:DATA?	FREQuency|POWer|WAVelength|WNUMber|{ALL[,WAVelength|FREQuency|WNUMber]}	5-15
:MTHResh[*1]	<thresh>	5-15
:PEXCursion	MINimum|MAXimum|DEFault|<integer>	5-15
:POINts?	none	5-15
:PTHReshold[*1]		
:ABSolute	<NRf>|MINimum|MAXimum|DEFault	5-15
[:RELative]	MINimum|MAXimum|DEFault|<integer>	5-16
:MODe	RELative|ABSolute	5-16
:WLIMit		
:STARt		
:FREQuency	<NRf>|MINimum|MAXimum	5-16
[:WAVelength]	<NRf>|MINimum|MAXimum	5-16
:WNUMber	<NRf>|MINimum|MAXimum	5-16
[:STATe]	ON|OFF|0|1	5-16
:STOP		
:FREQuency	<NRf>|MINimum|MAXimum	5-17
[:WAVelength]	<NRf>|MINimum|MAXimum	5-17
:WNUMber	<NRf>|MINimum|MAXimum	5-17

*1 仕様コード -MW：マルチ波長タイプ用

CALCulate3

コマンド	パラメータ	参照ページ
:CALCulate3		
:ASNR[*1]		
:COUNt	<integer>|MINmum|MAXimum	5-17
:DATA?	**Drift のとき** POWer|FREQuency|WAVelength|WNUMber|DROPped| {ALL[,WAVelength|FREQuency|WNUMber]} **Delta のとき** POWer|FREQuency|WAVelength|WNUMber **GRID のとき** STATus|{GRID[,WAVelength|FREQuency|WNUMber]}| {PEAK[,WAVelength|FREQuency|WNUMber|POWer]}| {DEViation[,WAVelength|FREQuency|WNUMber]}| {ALL[,WAVelength|FREQuency|WNUMber]} **WDM(OSNR) のとき** POWer|SIGNal|NOISe|{ALL[,WAVelength|FREQuency| WNUMber]}	5-17
:DELTa[*1]		
:POWer[:STATe]	0|OFF|1|ON	5-18
:PRESet	none	5-19
:REFerence		
:FREQuency	<NRf>|MINimum|MAXimum	5-19
:POWer?	none	5-19
[:WAVelength]	<NRf>|MINimum|MAXimum	5-19
:WNUMber	<NRf>|MINimum|MAXimum	5-20
:WAVelength[:STATe]	0|OFF|1|ON	5-20
:WPOWer[:STATe]	0|OFF|1|ON	5-209
:DLOGging		
:ETIMe?	none	5-20
:LPARameter		
:ASAVe		
:FNAMe?	none	5-20
[:STATe]	OFF|INTernal|EXTernal	5-20
:INTerval	MINimum|<NRf>	5-20
:ITEM	PEAK|FPLD	5-20
:LMODe	MODE1|MODE2|MODE3	5-21
:TDURation	<integer>	5-21
:MEASure		
:STATe	0|OFF|1|ON	5-21
:STATe	0|OFF|1|ON	5-21
:DRIFt		
:DIFFerence[:STATe]	0|OFF|1|ON	5-21
:ETIMe	none	5-21
:MAXimum[:STATe]	0|OFF|1|ON	5-21
:MINimum[:STATe]	0|OFF|1|ON	5-21
:POWer[:STATe]	0|OFF|1|ON	5-22
:WAVelength[:STATe]	0|OFF|1|ON	5-22
:PRESet	none	5-22
[:STATe]	0|OFF|1|ON	5-22
:REFerence		
:RESet	none	5-22
[:STATe]	0|OFF|1|ON	5-22
:FPERot[*1]		
[:STATe]	0|OFF|1|ON	5-22
:FWHM		
[:WAVelength]?	none	5-22
:FREQuency?	none	5-22
:WNUMber?	none	5-22
:MEAN		
[:WAVelength]?	none	5-22
:FREQuency?	none	5-22
:WNUMber?	none	5-22

コマンド	パラメータ	参照ページ
:MODE:SPACing		
[:WAVelength]?	none	5-23
:FREQuency?	none	5-23
:WNUMber?	none	5-23
:PEAK		
[:WAVelength]?	none	5-23
:FREQuency?	none	5-23
:WNUMber?	none	5-23
:POWer		
[:DBM]?	none	5-23
:WATTs?	none	5-23
:POWer		
[:DBM]?	none	5-23
:WATTs?	none	5-23
:SIGMa		
[:WAVelength]?	none	5-23
:FREQuency?	none	5-23
:WNUMber?	none	5-23
:GRID		
:DISPlay		
:ALL	0|OFF|1|ON	5-23
:REFerence		
:FREQuency	DEFault|<NRf>	5-23
:STARt		
[:WAVelength]	<NRf>	5-23
:FREQuency	<NRf>	5-23
:WNUMber	<NRf>	5-23
[:STATe]	0|OFF|1|ON	5-24
:STOP		
[:WAVelength]	<NRf>	5-24
:FREQuency	<NRf>	5-24
:WNUMber	<NRf>	5-24
:SPACing		
:FREQuency	<NRf>	5-24
:SAREa		
:FREQuency	<NRf>	5-24
:POINts?	none	5-24
:PRESet	none	5-24
:SNR		
:AUTO	0|OFF|1|ON	5-24
:REFerence		
[:WAVelength]		
:RELative	<NRf>	5-24
:BWIDth	<NRf>	5-25
[:STATE]	0|OFF|1|ON	5-25

*1 仕様コード -MW：マルチ波長タイプ用

CONFigure

コマンド	パラメータ	参照ページ
:CONFigure?	none	5-25
[:SCALar]		
:POWer	MAXimum|MINimum|DEFault|<NRf>	5-25
:FREQuency	MAXimum|MINimum|DEFault|<NRf>	5-26
:WAVelength	MAXimum|MINimum|DEFault|<NRf>	5-26
:WNUMber	MAXimum|MINimum|DEFault|<NRf>	5-26
:ARRay*1		
:POWer	MAXimum|MINimum|DEFault|<NRf>	5-26
:FREQuency	MAXimum|MINimum|DEFault|<NRf>	5-26
:WAVelength	MAXimum|MINimum|DEFault|<NRf>	5-26
:WNUMber	MAXimum|MINimum|DEFault|<NRf>	5-27

*1 仕様コード -MW：マルチ波長タイプ用

DISPlay

コマンド	パラメータ	参照ページ
:DISPlay		
:COLor	0|1	5-27
[:WINDow]	0|OFF|1|ON	5-27
:MARKer*1		
:MAXimum	none	5-27
:LEFT	none	5-27
:NEXT	none	5-27
:PREVious	none	5-27
:RIGHt	none	5-28
:RESolution		
[:WAVelength]	R0.0001|R0.001|R0.01|R0.1|MAXimum|MINimum	5-28
:FREQuency	R0.00001|R0.0001|R0.001|R0.01|MAXimum|MINimum	5-28
:WNUMber	R0.001|R0.01|R0.1|R1|MAXimum|MINimum	5-28
:UNIT		
:WAVelength	NM|THZ|ICM	5-28
[:WINDow]		
:TEXT		
:DATA	<"string">	5-28
:STATe	0|OFF|1|ON	5-28
:WINDow2*1		
:STATe	0|OFF|1|ON	5-28
:TRACe		
[:SCALe]		
:AUTOmeasure	none	5-29
:ASCale	none	5-29
:INITialize	none	5-29
:LEFT		
[:WAVelength]	<NRf>|MINimum|MAXimum	5-29
:FREQuency	<NRf>|MINimum|MAXimum	5-29
:WNUMber	<NRf>|MINimum|MAXimum	5-29
:RIGHt		
[:WAVelength]	<NRf>|MINimum|MAXimum	5-30
:FREQuency	<NRf>|MINimum|MAXimum	5-30
:WNUMber	<NRf>|MINimum|MAXimum	5-30
:CENTer		
[:WAVelength]	<NRf>	5-30
:FREQuency	<NRf>	5-30
:WNUMber	<NRf>	5-31
:PEAK	none	5-31
:SPAN		
[:WAVelength]	<NRf>|MAXimum	5-31
:FREQuency	<NRf>|MAXimum	5-31
:WNUMber	<NRf>|MAXimum	5-31

*1 仕様コード -MW：マルチ波長タイプ用

FETCh

コマンド	パラメータ	参照ページ
:FETCh?	none	5-32
:ARRay		
:POWer?	MAXimum|MINimum|DEFault|<NRf>	5-32
:FREQuency?	MAXimum|MINimum|DEFault|<NRf>	5-32
:WAVelength?	MAXimum|MINimum|DEFault|<NRf>	5-33
:WNUMber?	MAXimum|MINimum|DEFault|<NRf>	5-33
[:SCALar]		
:POWer?	MAXimum|MINimum|DEFault|<NRf>	5-33
:FREQuency?	MAXimum|MINimum|DEFault|<NRf>	5-33
:WAVelength?	MAXimum|MINimum|DEFault|<NRf>	5-34
:WNUMber?	MAXimum|MINimum|DEFault|<NRf>	5-34

FORMat

コマンド	パラメータ	参照ページ
:FORMat		
:NDATa		
[:WAVelength]	<NRf>	5-34

MEASure

コマンド	パラメータ	参照ページ
:MEASure		
:ARRay		
:POWer?	MAXimum|MINimum|DEFault|<NRf>	5-35
:FREQuency?	MAXimum|MINimum|DEFault|<NRf>	5-35
:WAVelength?	MAXimum|MINimum|DEFault|<NRf>	5-35
:WNUMber?	MAXimum|MINimum|DEFault|<NRf>	5-36
[:SCALar]		
:POWer?	MAXimum|MINimum|DEFault|<NRf>	5-36
:FREQuency?	MAXimum|MINimum|DEFault|<NRf>	5-36
:WAVelength?	MAXimum|MINimum|DEFault|<NRf>	5-36
:WNUMber?	MAXimum|MINimum|DEFault|<NRf>	5-37

MMEMory

コマンド	パラメータ	参照ページ
:MMEMory		
:CATalog?	[<"directory">|ROOT[,INTernal|EXTernal]]	5-37
:CDIRectory	<"directory">|ROOT[,INTernal|EXTernal]	5-38
:CDRive	INTernal|EXTernal	5-38
:COPY	<"source_file_name">,[INTernal|EXTernal], <"dest_file_name">[,INTernal|EXTernal]	5-38
:DATA?	<"filename">,[INTernal|EXTernal]	5-38
:DELete	<"filename">[,INTernal|EXTernal]	5-38
:INFormation?	<"filename">[,INTernal|EXTernal]	5-38
:LOAD	<"filename">[,INTernal|EXTernal]	5-38
:MDIRectory	<"directory_name">[,INTernal|EXTernal]	5-38
:PWDirectory?	none	5-38
:REMove	none	5-39
:REName	<"new_file_name">,<"old_file_name">[,INTernal| EXTernal]	5-39
:STORe	TABLe|SETup|SIMage1|SIMage2|SIMage3|DLOGging1| DLOGging2,<"filename">[,INTernal|EXTernal]	5-39

READ

コマンド	パラメータ	参照ページ
:READ?	none	5-39
:ARRay		
:POWer?	MAXimum|MINimum|DEFault|<NRf>	5-40
:FREQuency?	MAXimum|MINimum|DEFault|<NRf>	5-40
:WAVelength?	MAXimum|MINimum|DEFault|<NRf>	5-40
:WNUMber?	MAXimum|MINimum|DEFault|<NRf>	5-40
[:SCALar]		
:POWer?	MAXimum|MINimum|DEFault|<NRf>	5-41
:FREQuency?	MAXimum|MINimum|DEFault|<NRf>	5-41
:WAVelength?	MAXimum|MINimum|DEFault|<NRf>	5-41
:WNUMber?	MAXimum|MINimum|DEFault|<NRf>	5-41

SENSe

コマンド	パラメータ	参照ページ
[:SENSe]		
:CORRection		
:DEVice	NARRow|BROad	5-42
:MEDium	AIR|VACuum	5-42
:OFFSet		
[:MAGNitude]	<NRf>|MINimum|MAXimum	5-42
:URATe	NORMal|FAST	5-42

STATus

コマンド	パラメータ	参照ページ
:STATus		
:OPERation		
:CONDition?	none	5-42
:ENABle	<integer>	5-42
[:EVENt]?	none	5-42
:NTRansition	<integer>	5-42
:PTRansition	<integer>	5-42
:PRESet	none	5-43
:QUEStionable		
:CONDition?	none	5-43
:ENABle	<integer>	5-43
[:EVENt]?	none	5-43
:NTRansition	<integer>	5-43
:PTRansition	<integer>	5-43

SYSTem

コマンド	パラメータ	参照ページ
:SYSTem		
:BUZZer		
[:CLICk]	0|OFF|1|ON	5-43
:WARNing	0|OFF|1|ON	5-43
:CAPability		
:WAVelength?	none	5-43
:DATE	<year>,<month>,<day>	5-43
:ENVironment?	none	5-43
:ERRor?	none	5-44
:INFormation?	0|1	5-44
:LANGuage	ENGLish|CHINese|JAPanese	5-44
:PRESet	none	5-44
:REFLaser		
:CONDition?	none	5-44
:COUNter?	none	5-44
:OTIMe?	none	5-44
:TIME	<hour>,<minute>,<second>	5-44
:VERSion?	none	5-44

TRIGger

コマンド	パラメータ	参照ページ
[:TRIGger]		
:ABORt	none	5-45
:INITiate		
:CONTinuous	0|OFF|1|ON	5-45
[:IMMediate]	none	5-45

UNIT

コマンド	パラメータ	参照ページ
:UNIT		
[:POWer]	W|DBM	5-45
:WL	THZ|NM|ICM	5-45

5.4 共通コマンド

共通コマンドグループは、IEEE 488.2-1991 で規定されている、機器固有の機能に依存しないコマンドのグループです。このグループに相当するフロントパネルのキーはありません。

*CLS(Clear Status)

機能　エラーキュー、標準イベントレジスタ、ステータスバイトレジスタをクリアします。
構文　*CLS
例　*CLS
解説　オーバーラップコマンドです。

*ESE(Standard Event Status Enable)

機能　標準イベントイネーブルレジスタを設定 / 問い合わせします。
構文　*ESE<wsp><integer>
*ESE?
<integer>：0 ～ 255
例　*ESE 255
*ESE? -> +255<END>
解説　オーバーラップコマンドです。

*ESR?(Standard Event Status Register)

機能　標準イベントステータスレジスタの値を問い合わます。
構文　*ESR?
例　*ESR? -> +128<END>
解説　*ESR? で問い合わせると、標準イベントレジスタの内容がクリアされます。
解説　オーバーラップコマンドです。

*IDN?(Identification)

機能　機器の形名、シリアル番号、ファームウエアバージョンを問い合わせます。
構文　*IDN?
応答
YOKOGAWA,AQ615xB,<SerialNo>,<Version>
AQ615xB：形名
<SerialNo>：シリアル番号
<Version>：ファームウエアバージョン
例　*IDN? -> YOKOGAWA,AQ6151B,012345678,
01.00<END>
解説　オーバーラップコマンドです。

*OPC(Operation Complete)

機能　オーバーラップ動作が終了したときに、標準イベントレジスタのビット 0(OPC ビット) を設定 / 問い合わせします。
構文　*OPC
*OPC?
例　*OPC
*OPC? -> 1<END>
解説
- 設定時はビット 0 に 1 をセットします。問い合わせ時に 1 が返ってくる場合は、オーバーラップ動作が終了しています。
- オーバーラップコマンドです。
  動作完了のタイミングは 4-8 ページをご覧ください。

*RCL(Recall Command)

機能　本機器の設定状態を *SAV コマンドで保存した内容に戻します。
構文　*RCL<wsp>1|2|3|4
1|2|3|4：設定番号
例　*RCL 1
解説　戻したい設定状態 (1 ～ 4 の状態のどれか) を選んでください。

*RST(Reset)

機能　設定を初期化します。
構文　*RST
例　*RST

*SAV(Save Command)

機能　現在の本機器の設定状態を保存します。
構文　*SAV<wsp>1|2|3|4
1|2|3|4：設定番号
例　*SAV 1
解説　最大 4 種類の設定状態を保存できます。

*SRE(Service Request Enable)

機能　サービスリクエストイネーブルレジスタを設定 / 問い合わせします。
構文　*SRE<wsp><integer>
*SRE?
<integer>：0 ～ 255
例　*SRE 255
*SRE? -> +255<END>
解説　オーバーラップコマンドです。

*STB?(Read Status Byte)

機能　ステータスバイトレジスタの値を問い合わせます。
構文　*STB?
例　*STB? -> +12<END>
解説　オーバーラップコマンドです。

*TRG(Trigger)

機能　シングル測定を開始します。
構文　*TRG
例　*TRG
解説　被オーバーラップコマンドです。

*TST?(Self Test)

機能　本機器の自己診断を実行し、診断結果を問い合わせます。
構文　*TST?
応答　0：異常なし
0 以外：異常あり (エラーコード表示)
例　*TST? -> 0<END>
解説　・ 本機器では、常に 0 の応答を返します。
・ オーバーラップコマンドです。

*WAI(Wait to Continue)

機能　現在のコマンドの実行が終了するまで、本機器が他のコマンドを実行しないようにします。
構文　*WAI
例　*WAI
解説　オーバーラップコマンドです。
動作完了のタイミングは 4-8 ページをご覧ください。

5.5 機器固有コマンド

本機器の各個別の機能を操作するコマンドのパラメータや書式例を示します。
各コマンドに対応するソフトキーについては 5.2 節をご覧ください。

CALCulate2 Sub System コマンド

:CALCulate2:ASEarch

機能　自動ピーク (波長 /Power) 検出の ON/OFF を設定 / 問い合わせします。

構文　:CALCulate2:ASEarch<wsp>ON|OFF|0|1
:CALCulate2:ASEarch?
ON|1：自動ピーク検出を ON
OFF|0：自動ピーク検出を OFF

例　:CALC2:ASE ON
:CALC2:ASE? -> 1<END>

解説　仕様コード -MW：マルチ波長タイプで有効。

:CALCulate2:COUNt

機能　ピーク検出の平均化の回数を設定 / 問い合わせします。

構文　:CALCulate2:COUNt<wsp><average_times>
:CALCulate2:COUNt?
<average_times>(平均化の回数)：
<integer>|MINimum|MAXimum
MINimum：1
MAXimum：100

例　:CALC2:COUN 10
:CALC2:COUN? -> +10<END>

:CALCulate2:DATA?

機能　検出しているすべてのピークの測定値を問い合わせます。

構文　:CALCulate2:DATA?<wsp>FREQuency|POWer|WAVelength|WNUMber|{ALL[,WAVelength|FREQuency|WNUMber]}
FREQuency：波長を周波数の単位で問い合わせ
WAVelength：波長を波長の単位で問い合わせ
WNUMber：波長を波数の単位で問い合わせ
POWer：パワーの値を問い合わせ
ALL：全てのパワーと波長 (周波数、波長または波数) を問い合わせ

例　:CALC2:DATA? FREQ -> +1.93596570E+014,+1.93738272E+014,+1.93880006E+014<END>

解説
- 検出しているピークの数分すべてをカンマ区切りの浮動小数点数で返します。
- パワーの値は設定されている単位に従い返します。
- ピークを検出していない時 (No signal) は次の値を返します。
  波長、パワー (mw、μw)：0.000000E+000
  パワー (dBm)：-2.000000E+002
- オーバーラップコマンドです。

:CALCulate2:MTHResh

機能　チャネルマッチングを判定する周波数のしきい値を設定 / 問い合わせします。

構文　:CALCulate2:MTHResh<wsp><thresh>
:CALCulate2:MTHResh?
<thresh>：<NRf> チャネルマッチング判定しきい値を Hz 単位で指定 (1GHz ～ 99GHz)

例　:CALC2:MTHR 2GHZ
:CALC2:MTHR? -> +2.00000000E+009<END>

解説　仕様コード -MW：マルチ波長タイプで有効。

:CALCulate2:PEXCursion

機能　ピーク検出に用いるパワー差を設定 / 問い合わせします。

構文　:CALCulate2:PEXCursion<wsp>
<pexcursion_value>
:CALCulate2:PEXCursion?
<pexcursion_value>(パワー差)：
MINimum|MAXimum|DEFault|<integer>
MINimum：1dB
MAXimum：30dB
DEFault：15dB

例　:CALC2:PEXC 10
:CACL2:PEXC? -> +10<END>

:CALCulate2:POINts?

機能　検出しているピークの数を問い合わせます。

構文　:CALCulate2:POINts?

例　:CALC2:POIN? -> +3<END>

解説
- ピークの最大検出数は 1024 です。
- オーバーラップコマンドです。

:CALCulate2:PTHReshold:ABSolute

機能　ピーク検出のしきい値を絶対値で設定 / 問い合わせします。

構文　:CALCulate2:PTHReshold:ABSolute
<wsp><thresh>
:CALCulate2:PTHReshold:ABSolute?
<thresh>(しきい値)：
<NRf>|MINimum|MAXimum|DEFault
MINimum：-40dBm
MAXimum：10dBm
DEFault：-20dBm

例　:CALC2:PTHR:ABS -20
:CALC2:PTHR:ABS?
-> -2.00000000E+001<END>

解説　仕様コード -MW：マルチ波長タイプで有効。

:CALCulate2:PTHReshold[:RELative]

機能　ピーク検出のしきい値を最大パワーピークの相対値で設定 / 問い合わせします。

構文　:CALCulate2:PTHReshold[:RELative]<wsp><thresh>
:CALCulate2:PTHReshold[:RELative]?
<thresh>(しきい値) :
MINimum/MAXimum/DEFault/<integer>
MINimum：0dB
MAXimum：40dB
DEFault：10dB

例　:CALC2:PTHR 9
:CALC2:PTHR? -> +9<END>

解説　仕様コード -MW：マルチ波長タイプで有効。

:CALCulate2:PTHReshold:MODe

機能　ピーク検出のしきい値の定義を設定 / 問い合わせします。

構文　:CALCulate2:PTHReshold:MODe<wsp>
RELative|ABSolute
:CALCulate2:PTHReshold:MODe?
RELative：しきい値を相対値で定義
ABSolute：しきい値を絶対値で定義

例　:CALC2:PTHR:MOD REL
:CALC2:PTHR:MODE? -> REL<END>

解説　仕様コード -MW タイプで有効。

:CALCulate2:WLIMit:STARt:FREQuency

機能　ピーク検出の測定範囲制限の開始周波数を設定 / 問い合わせします。

構文　:CALCulate2:WLIMit:STARt
:FREQuency<wsp><freq>
:CALCulate2:WLIMit:STARt:FREQuency?
<freq>：(開始周波数)
<NRf>|MINimum|MAXimum
MINimum：　181.69THz(仕様コード -10 タイプ)
176.35THz(仕様コード -20 タイプ)
176.35THz(仕様コード -30 タイプ)
MAXimum：　終了周波数ー 0.1THz

例　:CALC2:WLIM:STAR:FREQ 191THZ
:CALC2:WLIM:STAR:FREQ?
-> +1.91000000E+014<END>

解説　問い合わせ結果は Hz 単位で返します。

:CALCulate2:WLIMit:STARt[:WAVelength]

機能　ピーク検出の測定範囲制限の開始波長を設定 / 問い合わせします。

構文　:CALCulate2:WLIMit:STARt[:WAVelength]
<wsp><wavelength>
:CALCulate2:WLIMit:STARt
[:WAVelength]?
<wavelength>(開始波長) :
<NRf>|MINimum|MAXimum
MINimum：　1270nm(仕様コード -10 タイプ)
1200nm(仕様コード -20 タイプ)
900nm(仕様コード -30 タイプ)
MAXimum：終了波長ー 1nm

例　:CALC2:WLIM:STAR 1500NM
:CALC2:WLIM:STAR?
-> +1.50000000E-006<END>

解説　問い合わせ結果は m 単位で返します。

:CALCulate2:WLIMit:STARt:WNUMber

機能　ピーク検出の測定範囲制限の開始波数を設定 / 問い合わせします。

構文　:CALCulate2:WLIMit:STARt:WNUMber<wsp>
<wnumber>
:CALCulate2:WLIMit:STARt:WNUMber?
<wnumber>(波数) :
<NRf>|MINimum|MAXimum
MINimum：　6060.0cm^{-1}(仕様コード -10 タイプ)
5882.4cm^{-1}(仕様コード -20 タイプ)
5882.4cm^{-1}(仕様コード -30 タイプ)
MAXimum：終了波数ー 1cm^{-1}

例　:CALC2:WLIM:STAR:WNUM 6400ICM
:CALC2:WLIM:STAR:WNUM? ->
+6.40000000E+005<END>

解説　問い合わせ結果は m^{-1} 単位で返します。

:CALCulate2:WLIMit[:STATe]

機能　ピーク検出の測定範囲制限の ON/OFF を設定 / 問い合わせします。

構文　:CALCulate2:WLIMit[:STATe]<wsp>
0|OFF|1|ON
:CALCulate2:WLIMit[:STATe]?
0|OFF：測定範囲制限を OFF
1|ON：測定範囲制限を ON

例　:CALC2:WLIM ON
:CALC2:WLIM? -> 1<END>

:CALCulate2:WLIMit:STOP:FREQuency

機能　ピーク検出の測定範囲制限の終了周波数を設定 / 問い合わせします。

構文　:CALCulate2:WLIMit:STOP:FREQuency <wsp><frequency>
:CALCulate2:WLIMit:STOP:FREQuency?
<frequency>(終了周波数) :
<NRf>|MINimum|MAXimum
MINimum：開始周波数 +0.1THz
MAXimum： 230.06THz(仕様コード -10 タイプ)
249.83THz(仕様コード -20 タイプ)
333.11THz(仕様コード -30 タイプ)

例　:CALC2:WLIM:STOP:FREQ 195THZ
:CALC2:WLIM:STOP:FREQ?
-> +1.95000000E+014<END>

解説　問い合わせ結果は Hz 単位で返します。

:CALCulate2:WLIMit:STOP[:WAVelength]

機能　ピーク検出の測定範囲制限の終了波長を設定 / 問い合わせします。

構文　:CALCulate2:WLIMit:STOP[:WAVelength] <wsp><wavelength>
:CALCulate2:WLIMit:STOP[:WAVelength]?
<wavelength>(終了波長) :
<NRf>|MINimum|MAXimum
MINimum：開始波長 +1nm
MAXimum： 1650nm(仕様コード -10 タイプ)
1700nm(仕様コード -20 タイプ)
1700nm(仕様コード -30 タイプ)

例　:CALC2:WLIM:STOP 1640NM
:CALC2:WLIM:STOP?
-> +1.64000000E-006<END>

解説　問い合わせ結果は m 単位で返します。

:CALCulate2:WLIMit:STOP:WNUMber

機能　ピーク検出の測定範囲制限の終了波数を設定 / 問い合わせします。

構文　:CALCulate2:WLIMit:STOP:WNUMber<wsp><wnumber>
:CALCulate2:WLIMit:STOP:WNUMber?
<wnumber>(波数) :
<NRf>|MINimum|MAXimum
MINimum：開始波数 +1cm^{-1}
MAXimum： 7875.0cm^{-1}(仕様コード -10 タイプ)
8333.3cm^{-1}(仕様コード -20 タイプ)
11111.1cm^{-1}(仕様コード -30 タイプ)

例　:CALC2:WLIM:STOP:WNUM 7800ICM
:CALC2:WLIM:STOP:WNUM? ->
+7.80000000E+005<END>

解説　問い合わせ結果は m^{-1} 単位で返します。

CALCulate3 Sub System コマンド

:CALCulate3:ASNR:COUNt

機能　OSNR 計算の平均化回数を設定 / 問い合わせします。

構文　:CALCulate3:ASNR:COUNt<wsp><count>|MINimum|MAXimum
:CALCulate3:ASNR:COUNt?
<count>：<integer> 形式、1 ～ 100 の平均化回数
MINimum：1
MAXimum：100

例　:CALC3:ASNR:COUN 3
:CALC3:ASNR:COUN? -> 3<END>

解説
- オーバーラップコマンドです。
- 本設定は、平均化回数設定 (:CALC2:COUNT) と共通です。
- 仕様コード -MW：マルチ波長タイプで有効。

:CALCulate3:DATA?

機能　ドリフト測定、デルタ測定 (仕様コード -MW：マルチ波長タイプで有効)、グリッド測定、WDM(OSNR) 測定の結果を問い合わせます。

構文　ドリフト測定時
:CALCulate3:DATA?<wsp>POWer|FREQuency|WAVelength|WNUMber|DROPped|{ALL[,WAVelength|FREQuency|WNUMber]}
デルタ測定時
:CALCulate3:DATA?<wsp>POWer|FREQuency|WAVelength|WNUMber
グリッド測定時
:CALCulate3:DATA?<wsp>STATus|{GRID[,WAVelength|FREQuency|WNUMber]}|{PEAK[,WAVelength|FREQuency|WNUMber|POWer]}|{DEViation[,WAVelength|FREQuency|WNUMber]}|{ALL[,WAVelength|FREQuency|WNUMber]}
WDM(OSNR) 時
:CALCulate3:DATA?<wsp>POWer|SIGNal|NOISe|{ALL[,WAVelength|FREQuency|WNUMber]}

例　:CALC3:DATA? POW -> 4.80000000E-001,-3.60000000E-001,+5.70000000E-001<END>

解説
- オーバーラップコマンドです。
- ドリフト、デルタ、グリッド、WDM(OSNR) の設定で測定が ON になっているほうの結果を返します。
- ドリフト測定の ON/OFF については、:CALCulate3:DRIFt[:STATe] コマンドをご覧ください。
- デルタ測定の ON/OFF については、以下のコマンドをご覧ください。
:CALCulate3:DELTa:POWer[:STATe] コマンド
:CALCulate3:DELTa:WAVelength[:STATe] コマンド
:CALCulate3:DELTa:WPOWer[:STATe] コマンド
- グリッド測定の ON/OFF については、:CALCulate3:GRID[:STATe] コマンドをご覧ください。
- WDM(OSNR) 測定の ON/OFF については、:CALCulate3:SNR[:STATe] コマンドをご覧ください。

ドリフト測定のとき
- 各パラメータの応答は、以下の項目の中から ON (測定をする) になっている項目だけ、検出しているピークの数分をカンマ区切りで返します。
  MAX 値、MIN 値、MAX-MIN 値、DELTA 値、Wavelength 値、Power 値、Ref 値
  例) MAX 値が ON になっていて、ピークが 3 つ検出されているときは、MAX 値の値が 3 つ返ります。
  各項目の ON/OFF はコマンドで設定できます。詳細は各コマンドをご覧ください。
- ALL のパラメータを指定したときは、DROPped、MAX POWer、MIN POWer、MAX-MIN POWer、REF POWer、POWer、MAX WAVelength、MIN WAVelength、MAX-MIN WAVelength、REF WAVelength、WAVelength の順番で、応答をカンマ区切りで返します。
- ALL,WAVlength のパラメータを指定したときは、ALL の場合と同じです。
- ALL,FREQuency のパラメータを指定したときは、DROPped、MAX POWer、MIN POWer、MAX-MIN POWer、REF POWer、POWer、MAX FREQuency、MIN FREQuency、MAX-MIN FREQuency、REF FREQuency、FREQuency の順番で、応答をカンマ区切りで返します。
- ALL,WNUMber のパラメータを指定したときは、DROPped、MAX POWer、MIN POWer、MAX-MIN POWer、REF POWer、POWer、MAX WNUMber、MIN WNUMber、MAX-MINWNUMber、REF WNUMber、WNUMber の順番で、応答をカンマ区切りで返します。
- 応答のデータ形式は以下のとおりです。
  DROPped：
  0：通常データ、1：ドロップデータ
  POWer、WAVelength、FREQuency、WNUMber：
  浮動小数点数

デルタ測定のとき (仕様コード -MW：マルチ波長タイプで有効)
- パラメータで指定した項目だけ、検出しているピークの数分をカンマ区切りの浮動小数点数で返します。
  Power 値、Wavelength 値、Frequency 値、Wnumber 値

グリッド測定のとき
- パラメータで指定した項目の測定値を、グリッド順にカンマ区切りで返します。
  STATus PEAK の有無
  0：ピークなし
  1：ピークあり
  2：複数のピークあり
  GRID グリッド波長 (単位は波長単位による)
  GRID,FREQuency グリッド周波数
  GRID,WAVelength グリッド波長
  GRID,WNUMber グリッド波数
  DEViation グリッドから最も近いピークとグリッドとの偏差 (単位は波長単位による)
  DEViation,FREQuency
  グリッドから最も近いピークとグリッドとの偏差 (周波数)
  DEViation,WAVelength
  グリッドから最も近いピークとグリッドとの偏差 (波長)
  DEViation,WNUMber
  グリッドから最も近いピークとグリッドとの偏差 (波数)
  PEAK グリッドにあるピークの波長 (単位は波長単位による)
  PEAK,FREQuency
  グリッドにあるピークの周波数
  PEAK,WAVelength
  グリッドにあるピークの波長
  PEAK,WNUMber：
  グリッドにあるピークの波数
  PEAK,POWer
  グリッドにあるピークの Power
  ALL グリッド No、Status、グリッド波長、Deviation、Peak 波長 (単位は波長単位による)、Peak Power
  ALL,FREQuency グリッド No、Status、グリッド周波数、Deviation、Peak 周波数、,Peak Power
  ALL,WAVelength グリッド No、Status, グリッド波長、Deviation、Peak 波長、Peak Power
  ALL,WNUMber グリッド No、Status、グリッド波数、Deviation、Peak 波数、Peak Power
- SHOW ALL の設定が ON のときは、全てのグリッドの測定値を返します。
  SHOW ALL の設定が OFF のときは、ピークのあるグリッドの測定値を返します。

WDM(OSNR) 測定のとき
- パラメータで指定した項目の測定値を、カンマ区切りで返します。
  POWer OSNR の配列
  SIGNal Signal Power の配列
  NOISe Noise Power の配列
  ALL 波長、OSNR、Signal Power、Noise Power の配列
  ALL,FREQuency 周波数、OSNR、Signal Power、Noise Power
  ALL,WAVelength 波長、OSNR、Signal Power、Noise Power
  ALL,WNUMber 波数、OSNR、Signal Power、Noise Power
- ピーク、パワーなどの測定値を取得するときは、:CALC2:DATA? コマンドを使用してください。

:CALCulate3:DELTa:POWer[:STATe]

機能 デルタ測定 ON/OFF を設定 / 問い合わせします。

構文
```
:CALCulate3:DELTa:POWer[:STATe]
<wsp>0|OFF|1|ON
:CALCulate2:DELTa:POWer[:STATe]?
```
0|OFF： デルタ測定をしない
1|ON： デルタ測定をする

例
```
:CALC3:DELT:POW ON
:CALC3:DELT:POW? -> 1<END>
```

解説 仕様コード -MW：マルチ波長タイプで有効。

:CALCulate3:DELTa:PRESet

機能　デルタ測定を中止します。
構文　:CALCulate3:DELTa:PRESet
例　:CALC3:DELT:PRES
解説　仕様コード -MW：マルチ波長タイプで有効。

:CALCulate3:DELTa:REFerence:FREQuency

機能　デルタ測定のリファレンスピークを周波数で設定 / 問い合わせします。
構文　:CALCulate3:DELTa:REFerence:FREQuency <wsp><frequency>
:CALCulate3:DELTa:REFerence:FREQuency?
<frequency>(周波数)：
<NRf>/MINimum/MAXimum
MINimum：　181.69THz(仕様コード -10 タイプ)
176.35THz(仕様コード -20 タイプ)
176.35THz(仕様コード -30 タイプ)
MAXimum：　230.06THz(仕様コード -10 タイプ)
249.83THz(仕様コード -20 タイプ)
333.11THz(仕様コード -30 タイプ)
例　:CALC3:DELT:REF:FREQ 193.8THZ
:CALC3:DELT:REF:FREQ?
-> +1.93878971E+014<END>
解説　・ 問い合わせ結果は Hz 単位で返します。
コマンドで設定した周波数値にもっとも近いピークがリファレンスとなります。
このため、入力した周波数値と問い合わせ結果が異なる場合があります。
・ 仕様コード -MW：マルチ波長タイプで有効。

:CALCulate3:DELTa:REFerence:POWer?

機能　デルタ測定のリファレンスピークのパワーを問い合わせます。
構文　:CALCulate3:DELTa:REFerence:POWer?
例　:CALC3:DELT:REF:POW?
-> -1.02600000E+001<END>
解説　・ 問い合わせ結果は、設定により dBm または W 単位で返します。
・ 仕様コード -MW：マルチ波長タイプで有効。

:CALCulate3:DELTa:REFerence[:WAVelength]

機能　デルタ測定のリファレンスピークを波長で設定 / 問い合わせします。
構文　:CALCulate3:DELTa:REFerence[:WAVelength]<wsp><wavelength>
:CALCulate3:DELTa:REFerence[:WAVelength]?
<wavelength>(波長)：<NRf>/MINimum/MAXimum
MINimum：　1270nm(仕様コード -10 タイプ)
1200nm(仕様コード -20 タイプ)
900nm(仕様コード -30 タイプ)
MAXimum：　1650nm(仕様コード -10 タイプ)
1700nm(仕様コード -20 タイプ)
1700nm(仕様コード -30 タイプ)
例　:CALC3:DELT:REF 1547.4NM
:CALC3:DELT:REF? ->
+1.54741791E-006<END>
解説　・ 問い合わせ結果は m 単位で返します。
コマンドで設定した波長の値にもっとも近いピークがリファレンスとなります。
このため、入力した波長の値と問い合わせ結果が異なる場合があります。
・ 仕様コード -MW：マルチ波長タイプで有効。

:CALCulate3:DELTa:REFerence:WNUMber

機能　デルタ測定のリファレンスピークを波数で設定 / 問い合わせします。
構文　:CALCulate3:DELTa:REFerence:WNUMber<wsp><wnumber>
:CALCulate3:DELTa:REFerence:WNUMber?
<wnumber>(波数)：
<NRf>/MINimum/MAXimum
MINimum：　6060.0cm^{-1}(仕様コード -10 タイプ)
5882.4cm^{-1}(仕様コード -20 タイプ)
5882.4cm^{-1}(仕様コード -30 タイプ)
MAXimum：　7875.0cm^{-1}(仕様コード -10 タイプ)
8333.3cm^{-1}(仕様コード -20 タイプ)
11111.1cm^{-1}(仕様コード -30 タイプ)
例　:CALC3:DELT:REF:WNUM 646700
:CALC3:DELT:REF:WNUM?
-> +6.46710630E+005<END>
解説　・ 問い合わせ結果は m^{-1} 単位で返します。
コマンドで設定した波数の値にもっとも近いピークがリファレンスとなります。
このため、入力した波数の値と問い合わせ結果が異なる場合があります。
・ 仕様コード -MW：マルチ波長タイプで有効。

:CALCulate3:DELTa:WAVelength[:STATe]

機能　デルタ測定 ON/OFF を設定 / 問い合わせします。
構文　:CALCulate3:DELTa:WAVelength[:STATe]<wsp>0|OFF|1|ON
:CALCulate3:DELTa:WAVelength[:STATe]?
0|OFF： デルタ測定をしない
1|ON： デルタ測定をする
例　:CALC3:DELT:WAV ON
:CALC3:DELT:WAV? -> 1<END>
解説　仕様コード -MW：マルチ波長タイプで有効。

:CALCulate3:DELTa:WPOWer[:STATe]

機能　デルタ測定 ON/OFF を設定 / 問い合わせします。
構文　:CALCulate3:DELTa:WPOWer[:STATe]<wsp>0|OFF|1|ON
:CALCulate3:DELTa:WPOWer[:STATe]?
0|OFF： デルタ測定をしない
1|ON： デルタ測定をする
例　:CALC3:DELT:WPOW ON
:CALC3:DELT:WPOW? -> 1<END>
解説　仕様コード -MW：マルチ波長タイプで有効。

:CALCulate3:DLOGging:ETIMe?

機能　データロギングの経過時間(秒)を問い合わせます。
構文　:CALCulate3:DLOGging:ETIMe?
例　:CALC3:DLOG:ETIM? -> 30<END>
解説　・ オーバーラップコマンドです。
・ データロギングが停止しているときは、本コマンドは無効になります。

:CALCulate3:DLOGging:LPARameter:ASAVe:FNAMe?

機能　データロギングの自動保存で、最後に保存されたファイル名を問い合わせます。
構文　:CALCulate3:DLOGging:LPARameter:ASAVe:FNAMe?
例　:CALC3:DLOG:LPAR:ASAV:FNAM?
-> L0001.WG1,EXT<END>
解説　・ 問い合わせ結果は、<filename>,INT|EXT で返します。
<filename>：ファイル名
INT：内部メモリ
EXT：USB メモリ
・ 現在自動保存している途中のファイルも含みます。
・ 自動保存されたファイルが存在しない場合は「,」を返します。

:CALCulate3:DLOGging:LPARameter:ASAVe[:STATe]

機能　データロギングのデータ自動保存の ON/OFF、および自動保存先のメディアを設定 / 問い合わせします。
構文　:CALCulate3:DLOGging:LPARameter:ASAVe[:STATe]<wsp><mode>
:CALCulate3:DLOGging:LPARameter:ASAVe[:STATe]?
<mode>：OFF|INTernal|EXTernal
OFF：自動保存しない
INTernal：内部メモリに自動保存する
EXTernal：USB メモリに自動保存する
例　:CALC3:DLOG:LPAR:ASAV EXT
:CALC3:DLOG:LPAR:ASAV? -> EXT<END>

:CALCulate3:DLOGging:LPARameter:INTerval

機能　データロギングの測定間隔を設定 / 問い合わせします。
構文　:CALCulate3:DLOGging:LPARameter:INTerval<wsp>MINimum|<NRf>[S|MS]
:CALCulate3:DLOGging:LPARameter:INTerval?
例　:CALC3:DLOG:LPAR:INT 5S
:CALC3:DLOG:LPAR:INT?
-> +5.0000000E+000
解説　・ 任意の値を入力できますが、200ms、500ms、1s、2s、5s、10s、30s、1m、2m、5m、10m のうち、近い値に設定されます。200ms を入力した場合は、MINimum のパラメータと同じ動作になります。
・ 問い合わせ結果は、浮動小数点形式の秒単位で返します。200ms または MINimum を設定した場合は「MIN」を返します。
・ データロギング実行中は本コマンドは無効になります。

:CALCulate3:DLOGging:LPARameter:ITEM

機能　データロギングの対象を設定 / 問い合わせします。
構文　:CALCulate3:DLOGging:LPARameter:ITEM<wsp>PEAK|FPLD
:CALCulate3:DLOGging:LPARameter:ITEM?
PEAK：各 PEAK の波長、Power をロギングする
FPLD：PEAK を FP-LD 解析した結果をロギングする
例　:CALC3:DLOG:LPAR:ITEM PEAK
:CALC3:DLOG:LPAR:ITEM? -> PEAK<END>
解説　・ データロギング実行中は本コマンドは無効になります。
・ 仕様コード -MW：マルチ波長タイプで有効。

:CALCulate3:DLOGging:LPARameter:LMODe

機能　データロギングのモード (最大チャネル数とロギング回数) を設定 / 問い合わせします。

構文
```
:CALCulate3:DLOGging:LPARameter:LMODe<wsp>MODE1|MODE2|MODE3
:CALCulate3:DLOGging:LPARameter:LMODe?
```
MODE1：最大 1024ch を 5001 回分ロギングする
MODE2：最大 256ch を 20001 回分ロギングする
MODE3：最大 64ch を 100001 回分ロギングする

例
```
:CALC3:DLOG:LPAR:LMOD MODE3
:CALC3:DLOG:LPAR:LMOD? -> MODE3<END>
```

解説
- データロギング実行中は本コマンドは無効になります。
- 仕様コード -MW：マルチ波長タイプで有効。

:CALCulate3:DLOGging:LPARameter:TDURation

機能　データロギングの測定時間 (秒) を設定 / 問い合わせします。

構文
```
:CALCulate3:DLOGging:LPARameter:TDURation<wsp><integer>[S]
:CALCulate3:DLOGging:LPARameter:TDURation
```
<integer>：1 ～ 8639999

例
```
:CALC3:DLOG:LPAR:TDURation 86400
:CALC3:DLOG:LPAR:TDUR? -> 86400<END>
```

解説　データロギング実行中は本コマンドは無効になります。

:CALCulate3:DLOGging:MEASure:STATe

機能　データロギングの開始 / 停止を実行 / 問い合わせします。

構文
```
:CALCulate3:DLOGging:MEASure:STATe<wsp>0|OFF|1|ON
:CALCulate3:DLOGging:MEASure:STATe?
```
0|OFF：データロギングの停止
1|ON： データロギングの開始

例
```
:CALC3:DLOG:MEAS:STAT ON
:CALC3:DLOG:MEAS:STAT? -> 1<END>
```

解説
- データロギング実行中は本コマンドは無効になります。
- 被オーバーラップコマンドです

:CALCulate3:DLOGging:STATe

機能　解析画面でのロギングデータ表示の ON/OFF を設定 / 問い合わせします。

構文
```
:CALCulate3:DLOGging:STATe<wsp>0|OFF|1|ON
:CALCulate3:DLOGging:STATe?
```
0|OFF：ロギングデータ表示の OFF
1|ON： ロギングデータ表示の ON

例
```
:CALC3:DLOG:STAT ON
:CALC3:DLOG:STAT? -> 1<END>
```

解説　データロギング実行中は本コマンドは無効になります。

:CALCulate3:DRIFt:DIFFerence[:STATe]

機能　ドリフト測定の MAX-MIN 値測定の ON/OFF を設定 / 問い合わせします。

構文
```
:CALCulate3:DRIFt:DIFFerence[:STATe]<wsp>0|OFF|1|ON
:CALCulate3:DRIFt:DIFFerence[:STATe]?
```
0|OFF： MAX-MIN 値を測定しない
1|ON： MAX-MIN 値を測定する

例
```
:CALC3:DRIF:DIFF ON
:CALC3:DRIF:DIFF? -> 1<END>
```

解説　すでに MAX 値、MIN 値、Ref 値、Power 値、Wavelength 値のどれかを測定している (測定 ON) ときは、ON に設定できません。

:CALCulate3:DRIFt:ETIMe?

機能　ドリフト測定の経過時間を問い合わせます。

構文
```
:CALCulate3:DRIFt:ETIMe?
```

例
```
:CALC3:DRIF:ETIM? -> 312<END>
```

解説　秒単位の経過時間を整数で返します。

:CALCulate3:DRIFt:MAXimum[:STATe]

機能　ドリフト測定の MAX 値測定の ON/OFF を設定 / 問い合わせします。

構文
```
:CALCulate3:DRIFt:MAXimum[:STATe]<wsp>0|OFF|1|ON
:CALCulate3:DRIFt:MAXimum[:STATe]?
```
0|OFF： MAX 値を測定しない
1|ON： MAX 値を測定する

例
```
:CALC3:DRIF:MAX ON
:CALC3:DRIF:MAX? -> 1<END>
```

解説　すでに MAX-MIN 値、MIN 値、Ref 値、Power 値、Wavelength 値のどれかを測定している (測定 ON) ときは、ON に設定できません。

:CALCulate3:DRIFt:MINimum[:STATe]

機能　ドリフト測定の MIN 値測定の ON/OFF を設定 / 問い合わせします。

構文
```
:CALCulate3:DRIFt:MINimum[:STATe]<wsp>0|OFF|1|ON
:CALCulate3:DRIFt:MINimum[:STATe]?
```
0|OFF： MIN 値を測定しない
1|ON： MIN 値を測定する

例
```
:CALC3:DRIF:MIN ON
:CALC3:DRIF:MIN? -> 1<END>
```

解説　すでに MAX-MIN 値、MAX 値、Ref 値、Power 値、Wavelength 値のどれかを測定している (測定 ON) ときは、ON に設定できません。

:CALCulate3:DRIFt:POWer[:STATe]

機能　ドリフト測定の Power 値測定の ON/OFF を設定 / 問い合わせします。

構文　:CALCulate3:DRIFt:POWer[:STATe]
<wsp>0|OFF|1|ON
:CALCulate3:DRIFt:POWer[:STATe]?
0|OFF： POWER 値を測定しない
1|ON： POWER 値を測定する

例　:CALC3:DRIF:POW ON
:CALC3:DRIF:POW? -> 1<END>

解説　すでに MAX-MIN 値、MAX 値、MIN 値、Ref 値、Wavelength 値のどれかを測定している (測定 ON) ときは、ON に設定できません。

:CALCulate3:DRIFt:WAVelength[:STATe]

機能　ドリフト測定の Wavelength 値測定の ON/OFF を設定 / 問い合わせします。

構文　:CALCulate3:DRIFt:WAVelength[:STATe]
<wsp>0|OFF|1|ON
:CALCulate3:DRIFt:WAVelength[:STATe]?
0|OFF： Wavelength 値を測定しない
1|ON： Wavelength 値を測定する

例　:CALC3:DRIF:WAV ON
:CALC3:DRIF:WAV? -> 1<END>

解説　すでに MAX-MIN 値、MAX 値、MIN 値、Ref 値、Power 値のどれかを測定している (測定 ON) ときは、ON に設定できません。

:CALCulate3:DRIFt:PRESet

機能　ドリフト測定の MAX-MIN 値、MAX 値、MIN 値、Ref 値の測定を OFF にします。

構文　:CALCulate3:DRIFt:PRESet

例　:CALC3:DRIF:PRES

解説　このコマンドを実行した後は、:CALCulate3:DATA? コマンドの応答はデルタ測定の結果が返ります。

:CALCulate3:DRIFt[:STATe]

機能　ドリフト測定の ON/OFF を設定 / 問い合わせします。

構文　:CALCulate3:DRIFt[:STATe]<wsp>
0|OFF|1|ON
:CALCulate3:DRIFt[:STATe]?
0|OFF： ドリフト測定をしない
1|ON： ドリフト測定をする

例　:CALC3:DRIF ON
:CALC3:DRIF? -> 1<END>

:CALCulate3:DRIFt:REFerence:RESet

機能　現在のピーク検出値をリファレンスとして、ドリフト測定を再度実行します。

構文　:CALCulate3:DRIFt:REFerence:RESet

例　:CALC3:DRIF:REF:PRES

解説　ドリフト測定結果は、このコマンドを実行した時点からの値になります。

:CALCulate3:DRIFt:REFerence[:STATe]

機能　ドリフト測定時の、:CALCulate3:DATA? コマンドに対する Ref 値の応答をする / しないを設定 / 問い合わせします。

構文　:CALCulate3:DRIFt:REFerence[:STATe]
<wsp>0|OFF|1|ON
:CALCulate3:DRIFt:REFerence[:STATe]?
0|OFF： Ref 値の応答をしない
1|ON： Ref 値の応答をする

例　:CALC3:DRIF:REF ON
:CALC3:DRIF:REF? -> 1<END>

:CALCulate3:FPERot[:STATe]

機能　FP-LD 解析の ON/OFF を設定 / 問い合わせします。

構文　:CALCulate3:FPERot[:STATe]<wsp>
0|OFF|1|ON
:CALCulate3:FPERot[:STATe]?
0|OFF： FP-LD 解析をしない
1|ON： FP-LD 解析をする

例　:CALC3:FPER ON
:CALC3:FPER? -> 1<END>

解説　仕様コード -MW：マルチ波長タイプで有効。

:CALCulate3:FPERot:FWHM?

機能　FP-LD 解析の FWHM 値を問い合わせます。

構文　:CALCulate3:FPERot:FWHM
{[:WAVelength]|:FREQuency|:
WNUMber}?
WAVelength：波長
FREQuency：周波数
WNUMber：波数

例　:CALC3:FPER:FWHM?
-> +3.12095579E-009<END>

解説
- 問い合わせ結果は、波長は m、周波数は Hz、波数は m^{-1} 単位で返します。
- オーバーラップコマンドです。
- 仕様コード -MW：マルチ波長タイプで有効。

:CALCulate3:FPERot:MEAN?

機能　FP-LD 解析の中心値を問い合わせます。

構文　:CALCulate3:FPERot:MEAN
{[:WAVelength]|:FREQuency|:WNUMber}?
WAVelength：波長
FREQuency：周波数
WNUMber：波数

例　:CALC3:FPER:MEAN?
-> +1.54721566E-006<END>

解説
- 問い合わせ結果は、波長は m、周波数は Hz、波数は m^{-1} 単位で返します。
- オーバーラップコマンドです。
- 仕様コード -MW：マルチ波長タイプで有効。

:CALCulate3:FPERot:MODE:SPACing?

機能　FP-LD 解析のチャネル間隔を問い合わせます。
構文　:CALCulate3:FPERot:MODE:SPACing
{[:WAVelength]|:FREQuency|:WNUMber}?
WAVelength：波長
FREQuency：周波数
WNUMber：波数
例　:CALC3:FPER:MODE:SPAC?
-> +1.50681284E-009<END>
解説
- 問い合わせ結果は、波長は m、周波数は Hz、波数は m^{-1} 単位で返します。
- オーバーラップコマンドです。
- 仕様コード -MW：マルチ波長タイプで有効。

:CALCulate3:FPERot:PEAK?

機能　FP-LD 解析のピーク値を問い合わせます。
構文　:CALCulate3:FPERot:PEAK
{[:WAVelength]|:FREQuency|:WNUMber|:POWer{[:DBM]|:WATTs}}?
WAVelength：波長
FREQuency：周波数
WNUMber：波数
POWer：パワー
例　:CALC3:FPER:PEAK?
-> +1.54742260E-006<END>
解説
- 問い合わせ結果は、波長は m、周波数は Hz、波数は m^{-1} 単位で返します。
- パワーは、dBm または W で返します。
- オーバーラップコマンドです。
- 仕様コード -MW：マルチ波長タイプで有効。

:CALCulate3:FPERot:POWer?

機能　FP-LD 解析のトータルパワー値を問い合わせます。
構文　:CALCulate3:FPERot:POWer
{[:DBM]|:WATTs}?
例　:CALC3:FPER:POW?
-> -1.21722665E+000<END>
解説
- 問い合わせ結果は、dBm または W で返します。
- オーバーラップコマンドです。
- 仕様コード -MW：マルチ波長タイプで有効。

:CALCulate3:FPERot:SIGMa?

機能　FP-LD 解析のσ値を問い合わせます。
構文　:CALCulate3:FPERot:SIGMa
{[:WAVelength]|:FREQuency|:WNUMber}?
WAVelength：波長
FREQuency：周波数
WNUMber：波数
例　:CALC3:FPER:SIGM?
-> +1.32524662E-009<END>
解説
- 問い合わせ結果は、波長は m、周波数は Hz、波数は m^{-1} 単位で返します。
- オーバーラップコマンドです。
- 仕様コード -MW：マルチ波長タイプで有効。

:CALCulate3:GRID:DISPlay:ALL

機能　全てのグリッドを表示するかしないかを設定 / 問い合わせします。
構文　:CALCulate3:GRID:DISPlay:ALL<wsp>0|OFF|1|ON
:CALCulate3:GRID:DISPlay:ALL?
0|OFF：　ピークのあるグリッドだけを表示する
1|ON：　全てのグリッドを表示する
例　:CALC3:GRID:DISP:ALL ON
:CALC3:GRID:DISP:ALL? -> 1<END>
解説　仕様コード -MW：マルチ波長タイプで有効。

:CALCulate3:GRID:REFerence:FREQuency

機能　グリッド表示のリファレンス周波数を設定 / 問い合わせします。
構文　:CALCulate3:GRID:REFerence:FREQuency<wsp><freq>
:CALCulate3:GRID:REFerence:FREQuency?
<freq>：基準周波数 (186THz ～ 202THz)
DEFault|<NRf>
DEFault：193.1THz
例　:CALC3:GRID:REF:FREQ 195THZ
:CALC3:GRID:REF:FREQ?
-> +1.95000000E+014<END>

:CALCulate3:GRID:STARt[:WAVelength]

機能　グリッド開始波長を設定 / 問い合わせします。
構文　:CALCulate3:GRID:STARt[:WAVelength]<wsp><wavelength>
:CALCulate3:GRID:STARt[:WAVelength]?
<wavelength>：　m 単位の開始波長
<NRf>
例　:CALC3:GRID:STAR 1500NM
:CALC3:GRID:STAR?
-> +1.50000000E-006<END>

:CALCulate3:GRID:STARt:FREQuency

機能　グリッド開始周波数を設定 / 問い合わせします。
構文　:CALCulate3:GRID:STARt:FREQuency<wsp><freq>
:CALCulate3:GRID:STARt:FREQuency?
<freq>：Hz 単位の開始周波数
<NRf>
例　:CALC3:GRID:STAR:FREQ 191THZ
:CALC3:GRID:STAR:FREQ?
-> +1.91000000E+014<END>

:CALCulate3:GRID:STARt:WNUMber

機能　グリッド開始波数を設定 / 問い合わせします。
構文　:CALCulate3:GRID:STARt:WNUMber<wsp><wnumber>
:CALCulate3:GRID:STARt:WNUMber?
<wnumber>：　m^{-1} 単位の開始波数
<NRf>
例　:CALC3:GRID:STAR:WNUM 640000
:CALC3:GRID:STAR:WNUM?
-> +6.40000000E+005

:CALCulate3:GRID[:STATe]

機能　グリッド表示 ON/OFF を設定 / 問い合わせします。
構文　:CALCulate3:GRID[:STATe]<wsp>0|OFF|1|ON
:CALCulate3:GRID[:STATe]?
0|OFF： グリッド表示を無効にする
1|ON： グリッド表示を有効にする
例　:CALC3:GRID ON
:CALC3:GRID? -> 1<END>

:CALCulate3:GRID:STOP[:WAVelength]

機能　グリッド終了波長を設定 / 問い合わせします。
構文　:CALCulate3:GRID:STOP[:WAVelength]<wsp><wavelength>
:CALCulate3:GRID:STOP[:WAVelength]?
<wavelength>： m 単位の終了波長
<NRf>
例　:CALC3:GRID:STOP 1500NM
:CALC3:GRID:STOP?
-> +1.50000000E-006<END>

:CALCulate3:GRID:STOP:FREQuency

機能　グリッド終了周波数を設定 / 問い合わせします。
構文　:CALCulate3:GRID:STOP:FREQuency<wsp><freq>
:CALCulate3:GRID:STOP:FREQuency?
<freq>：Hz 単位の終了周波数
<NRf>
例　:CALC3:GRID:STOP:FREQ 195THZ
:CALC3:GRID:STOP:FREQ?
-> +1.95000000E+014<END>

:CALCulate3:GRID:STOP:WNUMber

機能　グリッド終了波数を設定 / 問い合わせします。
構文　:CALCulate3:GRID:STOP:WNUMber<wsp><wnumber>
:CALCulate3:GRID:STOP:WNUMber?
<wnumber>： m^{-1} 単位の終了波数
<NRf>
例　:CALC3:GRID:STOP:WNUM 640000
:CALC3:GRID:STOP:WNUM?
-> +6.40000000E+005<END>

:CALCulate3:GRID:SPACing:FREQuency

機能　グリッド間隔を設定 / 問い合わせします。
構文　:CALCulate3:GRID:SPACing:FREQuency<wsp><freq>
:CALCulate3:GRID:SPACing:FREQuency?
<freq>：Hz 単位の開始周波数 (5G ～ 1000GHz)
<NRf>
例　:CALC3:GRID:SPAC:FREQ 100GHZ
:CALC3:GRID:SPAC:FREQ?
-> +1.00000000E+011<END>

:CALCulate3:GRID:SARea:FREQuency

機能　ピークの検索範囲を設定 / 問い合わせします。
構文　:CALCulate3:GRID:SARea:FREQuency<wsp><freq>
:CALCulate3:GRID:SARea:FREQuency?
<freq>：Hz 単位の検索範囲 (1G ～ 100GHz)
<NRf>
例　:CALC3:GRID:SAR:FREQ 1GHZ
:CALC3:GRID:SAR:FREQ?
-> +1.00000000E+009<END>
解説　グリッド間隔より大きな値は設定できません。

:CALCulate3:POINts?

機能　:CALCulate3:DATA? コマンド問い合わせに対する応答のデータ数を問い合わせます。
構文　:CALCulate3:POINts?
例　:CALC3:POIN? -> +4<END>
解説　・ 応答データは最大で 1024 個です。
・ デルタ測定モード、ドリフト測定モードのどちらでもない場合は、常に 0 を返します。
・ オーバーラップコマンドです。

:CALCulate3:PRESet

機能　デルタ測定、ドリフト測定、FP-LD 解析、WDM 解析、Grid 解析のすべてを OFF にします。
構文　:CALCulate3:PRESet
例　:CALC3:PRES

:CALCulate3:SNR:AUTO

機能　SNR のノイズ検出方法を設定 / 問い合わせします。
構文　:CALCulate3:SNR:AUTO<wsp>0|OFF|1|ON
:CALCulate3:SNR:AUTO?
0|OFF： ノイズレベルを MANUAL-FIX で計算する
1|ON： ノイズレベルを AUTO-CENTER で計算する
例　:CALC3:SNR:AUTO ON
:CALC3:SNR:AUTO? -> 1<END>

:CALCulate3:SNR:REFerence[:WAVelength]:RELative

機能　ノイズ検出方法が MANUAL-FIX のときのノイズ測定点を設定 / 問い合わせします。
構文　:CALCulate3:SNR:REFerence[:WAVelength]:RELative<wsp><ref>
:CALCulate3:SNR:REFerence[:WAVelength]:RELative?
<ref>: <nrf>m 単位の波長
例　:CALC3:SNR:REF:REL 10nm
:CALC3:SNR:REF:REL?
-> +1.00000000E-008<END>
解説　・ 測定点はピークからの相対波長で設定します。

:CALCulate3:SNR:REFerence:BWIDth

機能　ノイズ計算する帯域幅を設定 / 問い合わせします。
構文　:CALCulate3:SNR:REFerence:BWIDth<wsp>
<band>
:CALCulate3:SNR:REFerence:BWIDth?
<ref>:　m 単位の波長
<NRf>
例　:CALC3:SNR:REF:BWID 0.1nm
:CALC3:SNR:REF:BWID?
-> +1.00000000E-010<END>

:CALCulate3:SNR[:STATe]

機能　OSNR 解析の ON/OFF を設定 / 問い合わせします。
構文　:CALCulate3:SNR[:STATe]<wsp>0|OFF|1|ON
:CALCulate3:SNR[:STATe]?
0|OFF：　OSNR 解析を無効にする
1|ON：　OSNR 解析を有効にする
例　:CALC3:SNR ON
:CALC3:SNR? -> 1<END>

CONFigure Sub System コマンド

概要
- 本サブシステムは、本機器のピーク検出結果の画面表示方法を設定 / 問い合わせる機能です。操作キーから実行したときと同じように、コマンド実行により本機器の画面設定の View Mode を切替えるため、画面表示内容が切り替わります。
- CONFigure[:SCALar] コマンドを実行すると、本機器の画面表示がシングル表示になります。
- CONFigure:ARRay コマンドを実行すると、本機器の画面表示がマルチ表示になります。このコマンドは、仕様コード -MW：マルチ波長タイプで有効です。

:CONFigure?

機能　画面表示のカレントの設定内容を問い合わせます。
構文　:CONFigure?
例　:CONF? -> "ARR:POW DEF,DEF"<END>
解説
- CONFigure コマンドで設定した表示条件の内容をコマンド書式で返します。
(シングル表示 / マルチ表示):(波長 / 周波数 / 波数)< 値 >,(分解能)
シングル表示：POW
マルチ表示：ARR:POW
波長：WAV
周波数：FREQ
波数：WNUM
値：最大値 (MAX)| 最小値 (MIN)| カレント値 (DEF)| 指定値にもっとも近いピーク (浮動小数点数)
分解能：カレント値 (DEF)
- オーバーラップコマンドです。
- マルチ表示は、仕様コード -MW：マルチ波長タイプで有効です。

:CONFigure[:SCALar]:POWer

機能　画面 (View Mode) 上にシングル表示するピークをパワーで設定します。
構文　:CONFigure[:SCALar]:POWer<wsp>
[<expected_value>]
<expected_value>(ピークを指定するパワー):
MAXimum|MINimum|DEFault|<NRf>
MAXimum：最大パワーのピーク
MINimum：最小パワーのピーク
DEFault：現在選択されているピーク
<NRf>：指定したパワーに最も近いピーク
例　:CONF:POW -4dbm
解説
- パラメータを省略すると、DEF が設定されます。
- パラメータをパワー、DEF 以外に指定すると、ピークの自動検索機能は OFF になります。
- パラメータは、仕様コード -MW：マルチ波長タイプで有効です。

:CONFigure[:SCALar]:POWer:FREQuency

機能　画面 (View Mode) 上にシングル表示するピークを周波数で設定します。

構文　:CONFigure[:SCALar]:POWer:FREQuency <wsp>[<expected_value>]
<expected_value>(ピークを指定する周波数) :
MAXimum|MINimum|DEFault|<NRf>
MAXimum：最大周波数のピーク
MINimum：最小周波数のピーク
DEFault：現在選択されているピーク
<NRf>：指定した周波数に最も近いピーク

例　:CONF:POW:FREQ 193.6THZ

解説
- パラメータを省略すると、DEF が設定されます。
- パラメータに DEF 以外を指定すると、ピークの自動検索機能は OFF になります。
- パラメータは、仕様コード -MW：マルチ波長タイプで有効です。

:CONFigure[:SCALar]:POWer:WAVelength

機能　画面 (View Mode) 上にシングル表示するピークを波長で設定します。

構文　:CONFigure[:SCALar]:POWer:WAVelength <wsp>[<expected_value>]
<expected_value>(ピークを指定する波長) :
MAXimum|MINimum|DEFault|<NRf>
MAXimum：最大波長のピーク
MINimum：最小波長のピーク
DEFault：現在選択されているピーク
<NRf>：指定した波長に最も近いピーク

例　:CONF:POW:WAV 1547.4nm

解説
- パラメータを省略すると、DEF が設定されます。
- パラメータに DEF 以外を指定すると、ピークの自動検索機能は OFF になります。
- パラメータは、仕様コード -MW：マルチ波長タイプで有効です。

:CONFigure[:SCALar]:POWer:WNUMber

機能　画面 (View Mode) 上にシングル表示するピークを波数で設定します。

構文　:CONFigure[:SCALar]:POWer:WNUMber <wsp>[<expected_value>]
<expected_value>(ピークを指定する波数) :
MAXimum|MINimum|DEFault|<NRf>
MAXimum：最大波数のピーク
MINimum：最小波数のピーク
DEFault：現在選択されているピーク
<NRf>：指定した波数に最も近いピーク

例　:CONF:POW:WNUM 646710

解説
- パラメータを省略すると、DEF が設定されます。
- パラメータに DEF 以外を指定すると、ピークの自動検索機能は OFF になります。
- パラメータは、仕様コード -MW：マルチ波長タイプで有効です。

:CONFigure:ARRay:POWer

機能　画面 (View Mode) 上にマルチ表示するピークをパワーで設定します。

構文　:CONFigure:ARRay:POWer<wsp>[<expected_value>]
<expected_value>(ピークを指定するパワー) :
MAXimum|MINimum|DEFault|<NRf>
MAXimum：最大パワーのピーク
MINimum：最小パワーのピーク
DEFault：現在選択されているピーク
<NRf>：指定したパワーに最も近いピーク

例　:CONF:ARR:POW -4DBM

解説
- パラメータを省略すると、DEF が設定されます。
- パラメータに DEF 以外を指定すると、ピークの自動検索機能は OFF になります。
- 仕様コード -MW：マルチ波長タイプで有効。

:CONFigure:ARRay:POWer:FREQuency

機能　画面 (View Mode) 上にマルチ表示するピークを周波数で設定します。

構文　:CONFigure:ARRay:POWer:FREQuency<wsp>[<expected_value>]
<expected_value>(ピークを指定する周波数) :
MAXimum|MINimum|DEFault|<NRf>
MAXimum：最大周波数のピーク
MINimum：最小周波数のピーク
DEFault：現在選択されているピーク
<NRf>：指定した周波数に最も近いピーク

例　:CONF:ARR:POW:FREQ 193.6THZ

解説
- パラメータを省略すると、DEF が設定されます。
- パラメータに DEF 以外を指定すると、ピークの自動検索機能は OFF になります。
- 仕様コード -MW：マルチ波長タイプで有効。

:CONFigure:ARRay:POWer:WAVelength

機能　画面 (View Mode) 上にマルチ表示するピークを波長で設定します。

構文　:CONFigure:ARRay:POWer:WAVelength <wsp>[<expected_value>]
<expected_value>(ピークを指定する波長) :
MAXimum|MINimum|DEFault|<NRf>
MAXimum：最大波長のピーク
MINimum：最小波長のピーク
DEFault：現在選択されているピーク
<NRf>：指定した波長に最も近いピーク

例　:CONF:ARR:POW:WAV 1548.5NM

解説
- パラメータを省略すると、DEF が設定されます。
- パラメータに DEF 以外を指定すると、ピークの自動検索機能は OFF になります。
- 仕様コード -MW：マルチ波長タイプで有効。

:CONFigure:ARRay:POWer:WNUMber

機能　画面 (View Mode) 上にマルチ表示するピークを波数で設定します。

構文　:CONFigure:ARRay:POWer:WNUMber<wsp>[<expected_value>]
<expected_value>(ピークを指定する波数) :
MAXimum|MINimum|DEFault|<NRf>
MAXimum：最大波数のピーク
MINimum：最小波数のピーク
DEFault：現在選択されているピーク
<NRf>：指定した波数に最も近いピーク

例　:CONF:ARR:POW:WNUM 645760

解説
- パラメータを省略すると、DEF が設定されます。
- パラメータに DEF 以外を指定すると、ピークの自動検索機能は OFF になります。
- 仕様コード -MW：マルチ波長タイプで有効。

DISPlay Sub System コマンド

:DISPlay:COLor

機能　画面の表示色を設定 / 問い合わせします。

構文　:DISPlay:COLor<wsp>0|1
:DISPlay:COLor?
0：白黒
1：カラー

例　:DISP:COL 1
:DISP:COL? -> 1<END>

解説　オーバーラップコマンドです。

:DISPlay[:WINDow]

機能　画面表示の ON/OFF を設定 / 問い合わせします。

構文　:DISPlay[:WINDow]<wsp>OFF|0|ON|1
:DISPlay[:WINDow]?
0|OFF： 画面表示 OFF
1|ON： 画面表示 ON

例　:DISP OFF
:DISP? -> 0<END>

解説　オーバーラップコマンドです。

:DISPlay:MARKer:MAXimum

機能　カレントピークを最大パワーピークに設定します。

構文　:DISPlay:MARKer:MAXimum

例　:DISP:MARK:MAX

解説
- オーバーラップコマンドです。
- 仕様コード -MW：マルチ波長タイプで有効。

:DISPlay:MARKer:MAXimum:LEFT

機能　カレントピークを現在のカレントピークの 1 つ左隣に移動します。

構文　:DISPlay:MARKer:MAXimum:LEFT

例　:DISP:MARK:MAX:LEFT

解説
- オーバーラップコマンドです。
- 仕様コード -MW：マルチ波長タイプで有効。

:DISPlay:MARKer:MAXimum:NEXT

機能　カレントピークを現在のカレントピークパワーの次に低いパワーのピークに移動します。

構文　:DISPlay:MARKer:MAXimum:NEXT

例　:DISP:MARK:MAX:NEXT

解説
- オーバーラップコマンドです。
- 仕様コード -MW：マルチ波長タイプで有効。

:DISPlay:MARKer:MAXimum:PREVious

機能　カレントピークを現在のカレントピークパワーの次に高いパワーのピークに移動します。

構文　:DISPlay:MARKer:MAXimum:PREVious

例　:DISP:MARK:MAX:PREV

解説
- オーバーラップコマンドです。
- 仕様コード -MW：マルチ波長タイプで有効。

:DISPlay:MARKer:MAXimum:RIGHt

機能　カレントピークを現在のカレントピークの 1 つ右隣に移動します。
構文　:DISPlay:MARker:MAXimum:RIGHt
例　:DISP:MARK:MAX:RIGH
解説　・ オーバーラップコマンドです。
・ 仕様コード -MW：マルチ波長タイプで有効。

:DISPlay:RESolution[:WAVelength]

機能　画面に表示する波長の小数点以下の桁数を設定 / 問い合わせします。
構文　:DISPlay:RESolution[:WAVelength]
<wsp>R0.0001|R0.001|R0.01|R0.1|
MAXimum|MINimum
:DISPlay:RESolution[:WAVelength]?
R0.0001：小数点以下 4 桁
R0.001：小数点以下 3 桁
R0.01：小数点以下 2 桁
R0.1：小数点以下 1 桁
MAXimum：最大値の小数点以下 4 桁
MINimum：最小値の小数点以下 1 桁
例　:DISP:RES R0.01
:DISP:RES? -> R0.01<END>
解説　オーバーラップコマンドです。

:DISPlay:RESolution:FREQuency

機能　画面に表示する周波数の小数点以下の桁数を設定 / 問い合わせします。
構文　:DISPlay:RESolution:FREQuency
<wsp>R0.00001|R0.0001|R0.001|R0.01|
MAXimum|MINimum
:DISPlay:RESolution:FREQuency?
R0.00001：小数点以下 5 桁
R0.0001：小数点以下 4 桁
R0.001：小数点以下 3 桁
R0.01：小数点以下 2 桁
MAXimum：最大値の小数点以下 5 桁
MINimum：最小値の小数点以下 2 桁
例　:DISP:RES:FREQ R0.01
:DISP:RES:FREQ? -> R0.01<END>
解説　オーバーラップコマンドです。

:DISPlay:RESolution:WNUMber

機能　画面に表示する波数の小数点以下の桁数を設定 / 問い合わせします。
構文　:DISPlay:RESolution::WNUMber
<wsp>R0.001|R0.01|R0.1|R1
MAXimum|MINimum
:DISPlay:RESolution::WNUMber?
R0.001：小数点以下 3 桁
R0.01：小数点以下 2 桁
R0.1：小数点以下 1 桁
R1：整数
MAXimum：最大値の小数点以下 3 桁
MINimum：最小値の整数
例　:DISP:RES:WNUM R0.01
:DISP:RES:WNUM? -> R0.01<END>
解説　オーバーラップコマンドです。

:DISPlay:UNIT:WAVelength

機能　波長の単位を設定 / 問い合わせします。
構文　:DISPlay:UNIT:WAVelength<wsp>NM|THZ|ICM
:DISPlay:UNIT:WAVelength?
NM：波長 (nm)、 THZ：周波数 (THz)
ICM：波数 (cm^{-1})
例　:DISP:UNIT:WAV NM
:DISP:UNIT:WAV? -> NM<END>
解説　オーバーラップコマンドです。

:DISPlay[:WINDow]:TEXT:DATA

機能　画面表示上のラベル文字を設定 / 問い合わせします。
構文　:DISPlay[:WINDow]:TEXT:DATA
<wsp><"string">
:DISPlay[:WINDow]:TEXT:DATA?
<"string">：ラベルの文字列 (ダブルクォート文字を除いて最大 52 文字)
例　:DISP:TEXT:DATA "AQ6150B Optical Wavelength Meter"
:DISP:TEXT:DATA? -> AQ6150B Optical Wavelength Meter<END>
解説　オーバーラップコマンドです。

:DISPlay[:WINDow]:STATe

機能　マルチ波長表示ウインドウの ON/OFF を設定 / 問い合わせします。
構文　:DISPlay[:WINDow]:
STATe<wsp>0|OFF|1|ON
:DISPlay[:WINDow]:STATe?
0|OFF： マルチ波長表示ウインドウ OFF
1|ON： マルチ波長表示ウインドウ ON
例　:DISP:STAT ON
:DISP:STAT? -> 1<END>
解説　・ オーバーラップコマンドです。
・ 仕様コード -MW：マルチ波長タイプで有効。

:DISPlay:WINDow2:STATe

機能　スペクトルウインドウ表示の ON/OFF を設定 / 問い合わせします。
構文　:DISPlay:WINDow2:STATe<wsp>0|OFF|1|ON
:DISPlay:WINDow2:STATe?
0|OFF： スペクトルウインドウ表示 OFF
1|ON： スペクトルウインドウ表示 ON
例　:DISP:WIND2:STAT ON
:DISP:WIND2:STAT? -> 1<END>
解説　・ オーバーラップコマンドです。
・ 仕様コード -MW：マルチ波長タイプで有効。

:DISPlay:WINDow2:TRACe[:SCALe]:AUTOmeasure

機能　Single 測定を 1 回実行後にオートスケールを実行します。

構文
```
:DISPlay:WINDow2:TRACe[:SCALe]
:AUTOmeasure
```

例
```
:DISP:WIND2:TRAC:AUTO
```

解説
- オーバーラップコマンドです。
- 仕様コード -MW：マルチ波長タイプで有効。

:DISPlay:WINDow2:TRACe[:SCALe]:ASCale

機能　スペクトルウインドウ上の波形を最適化表示 (オートスケール) します。

構文
```
:DISPlay:WINDow2:TRACe[:SCALe]:ASCale
```

例
```
:DISP:WIND2:TRAC:ASC
```

解説
- オーバーラップコマンドです。
- 仕様コード -MW：マルチ波長タイプで有効。

:DISPlay:WINDow2:TRACe[:SCALe]:INITialize

機能　スペクトルウインドウ上の横軸スケール (周波数 / 波長 / 波数) を初期化します。

構文
```
:DISPlay:WINDow2:TRACe[:SCALe]:
INITialize
```

例
```
:DISP:WIND2:TRAC:INIT
```

解説
- スケールの左端は開始波長に、右端は終了波長になります。
- オーバーラップコマンドです。
- 仕様コード -MW：マルチ波長タイプで有効。

:DISPlay:WINDow2:TRACe[:SCALe]:LEFT[:WAVelength]

機能　スペクトルウインドウ上の横軸スケールの開始波長を設定 / 問い合わせします。

構文
```
:DISPlay:WINDow2:TRACe[:SCALe]:LEFT
[:WAVelength]<wsp><wavelength>
:DISPlay:WINDow2:TRACe[:SCALe]:LEFT
[:WAVelength]?
```
<wavelength>(開始波長)：
<NRf>|MINimum|MAXimum
MINimum：　1270nm(仕様コード -10 タイプ)
1200nm(仕様コード -20 タイプ)
900nm(仕様コード -30 タイプ)
MAXimum：終了波長ー 1nm

例
```
:DISP:WIND2:TRAC:LEFT 1550NM
:DISP:WIND2:TRAC:LEFT? ->
+1.55000000E-006<END>
```

解説
- 問い合わせ結果は、m 単位で返します。
- オーバーラップコマンドです。
- 仕様コード -MW：マルチ波長タイプで有効。

:DISPlay:WINDow2:TRACe[:SCALe]:LEFT:FREQuency

機能　スペクトルウインドウ上の横軸スケールの開始周波数を設定 / 問い合わせします。

構文
```
:DISPlay:WINDow2:TRACe[:SCALe]:LEFT:
FREQuency<wsp><frequency>
:DISPlay:WINDow2:TRACe[:SCALe]:LEFT:
FREQuency?
```
<frequency>(開始周波数)：
<NRf>|MINimum|MAXimum
MINimum：　181.69THz(仕様コード -10 タイプ)
176.35THz(仕様コード -20 タイプ)
175.35THz(仕様コード -30 タイプ)
MAXimum：終了周波数ー 0.1THz

例
```
:DISP:WIND2:TRAC:LEFT:FREQ 190THZ
:DISP:WIND2:TRAC:LEFT:FREQ? ->
+1.90000000E+014<END>
```

解説
- 問い合わせ結果は、Hz 単位で返します。
- オーバーラップコマンドです。
- 仕様コード -MW：マルチ波長タイプで有効。

:DISPlay:WINDow2:TRACe[:SCALe]:LEFT:WNUMber

機能　スペクトルウインドウ上の横軸スケールの開始波数を設定 / 問い合わせします。

構文
```
:DISPlay:WINDow2:TRACe[:SCALe]:LEFT:
WNUMber<wsp><wnumber>
:DISPlay:WINDow2:TRACe[:SCALe]:LEFT:
WNUMber?
```
<wnumber>(開始波数)：
<NRf>|MINimum|MAXimum
MINimum：　6060.0cm^{-1}(仕様コード -10 タイプ)
5882.4cm^{-1}(仕様コード -20 タイプ)
5882.4cm^{-1}(仕様コード -30 タイプ)
MAXimum：終了波数ー 1cm^{-1}

例
```
:DISP:WIND2:TRAC:LEFT:WNUM 609000
:DISP:WIND2:TRAC:LEFT:WNUM?
-> +6.09000000E+004<END>
```

解説
- 問い合わせ結果は、m^{-1} 単位で返します。
- オーバーラップコマンドです。
- 仕様コード -MW：マルチ波長タイプで有効。

:DISPlay:WINDow2:TRACe[:SCALe]:RIGHt[:WAVelength]

機能　スペクトルウインドウ上の横軸スケールの終了波長を設定 / 問い合わせします。

構文
```
:DISPlay:WINDow2:TRACe[:SCALe]:
RIGHt[:WAVelength]<wsp><wavelength>
:DISPlay:WINDow2:TRACe[:SCALe]:
RIGHt[:WAVelength]?
```
<wavelength>(終了波長) :
<NRf>|MINimum|MAXimum
MINimum：開始波長 +1nm
MAXimum： 1650nm(仕様コード -10 タイプ)
1700nm(仕様コード -20 タイプ)
1700nm(仕様コード -30 タイプ)

例
```
:DISP:WIND2:TRAC:RIGH 1600NM
:DISP:WIND2:TRAC:RIGH?
-> +1.60000000E-006<END>
```

解説
- 問い合わせ結果は、m 単位で返します。
- オーバーラップコマンドです。
- 仕様コード -MW：マルチ波長タイプで有効。

:DISPlay:WINDow2:TRACe[:SCALe]:RIGHt:FREQuency

機能　スペクトルウインドウ上の横軸スケールの終了周波数を設定 / 問い合わせします。

構文
```
:DISPlay:WINDow2:TRACe[:SCALe]:RIGHt:
FREQuency<wsp><frequency>
:DISPlay:WINDow2:TRACe[:SCALe]:RIGHt:
FREQuency?
```
<frequency>(終了周波数) :
<NRf>|MINimum|MAXimum
MINimum：開始周波数 +0.1THz
MAXimum： 230.06THz(仕様コード -10 タイプ)
249.83THz(仕様コード -20 タイプ)
333.11THz(仕様コード -30 タイプ)

例
```
:DISP:WIND2:TRAC:RIGH:FREQ 190THZ
:DISP:WIND2:TRAC:RIGH:FREQ?
-> +1.90000000E+014<END>
```

解説
- 問い合わせ結果は、Hz 単位で返します。
- オーバーラップコマンドです。
- 仕様コード -MW：マルチ波長タイプで有効。

:DISPlay:WINDow2:TRACe[:SCALe]:RIGHt:WNUMber

機能　スペクトルウインドウ上の横軸スケールの終了波数を設定 / 問い合わせします。

構文
```
:DISPlay:WINDow2:TRACe[:SCALe]:RIGHt:
WNUMber<wsp><wnumber>
:DISPlay:WINDow2:TRACe[:SCALe]:RIGHt:
WNUMber?
```
<wnumber>(終了波数) :
<NRf>|MINimum|MAXimum
MINimum：開始波数
MAXimum： 7875.0cm^{-1}(仕様コード -10 タイプ)
8333.3cm^{-1}(仕様コード -20 タイプ)
11111.1cm^{-1}(仕様コード -30 タイプ)

例
```
:DISP:WIND2:TRAC:RIGH:WNUM 609000
:DISP:WIND2:TRAC:RIGH:WNUM?
-> +6.09000000E+005<END>
```

解説
- 問い合わせ結果は、m^{-1} 単位で返します。
- オーバーラップコマンドです。
- 仕様コード -MW：マルチ波長タイプで有効。

:DISPlay:WINDow2:TRACe[:SCALe]:CENTer[:WAVelength]

機能　スペクトルウインドウ上の横軸スケールの中心波長を設定 / 問い合わせします。

構文
```
:DISPlay:WINDow2:TRACe[:SCALe]:
CENTer[:WAVelength]<wsp><wavelength>
:DISPlay:WINDow2:TRACe[:SCALe]:
CENTer[:WAVelength]?
```
<wavelength>(中心波長) :<NRf>

例
```
:DISP:WIND2:TRAC:CENT 1550NM
:DISP:WIND2:TRAC:CENT?
-> +1.55000000E-006<END>
```

解説
- 問い合わせ結果は、m 単位で返します。
- オーバーラップコマンドです。
- 仕様コード -MW：マルチ波長タイプで有効。

:DISPlay:WINDow2:TRACe[:SCALe]:CENTer:FREQuency

機能　スペクトルウインドウ上の横軸スケールの中心周波数を設定 / 問い合わせします。

構文
```
:DISPlay:WINDow2:TRACe[:SCALe]:
CENTer:FREQuency<wsp><frequency>
:DISPlay:WINDow2:TRACe[:SCALe]:
CENTer:FREQuency?
```
<frequency>(中心周波数) :<NRf>

例
```
:DISP:WIND2:TRAC:CENT:FREQ 190THZ
:DISP:WIND2:TRAC:CENT:FREQ?
-> +1.90000000E+014<END>
```

解説
- 問い合わせ結果は、Hz 単位で返します。
- オーバーラップコマンドです。
- 仕様コード -MW：マルチ波長タイプで有効。

:DISPlay:WINDow2:TRACe[:SCALe]:CENTer:WNUMber

機能　スペクトルウインドウ上の横軸スケールの中心波数を設定 / 問い合わせします。

構文
```
:DISPlay:WINDow2:TRACe[:SCALe]:
CENTer:WNUMber<wsp><wnumber>
:DISPlay:WINDow2:TRACe[:SCALe]:
CENTer:WNUMber?
```
<wnumber>(中心波数)：<NRf>

例
```
:DISP:WIND2:TRAC:CENT:WNUM 609000
:DISP:WIND2:TRAC:CENT:WNUM? ->
+6.09000000E+005<END>
```

解説
- 問い合わせ結果は、m^{-1} 単位で返します。
- オーバーラップコマンドです。
- 仕様コード -MW：マルチ波長タイプで有効。

:DISPlay:WINDow2:TRACe[:SCALe]:CENTer:PEAK

機能　カレントピークを横軸スケールの中央に表示します。

構文
```
:DISPlay:WINDow2:TRACe[:SCALe]:
CENTer:PEAK
```

例
```
:DISP:WIND2:TRAC:CENT:PEAK
```

解説
- オーバーラップコマンドです。
- 仕様コード -MW：マルチ波長タイプで有効。

:DISPlay:WINDow2:TRACe[:SCALe]:SPAN[:WAVelength]

機能　スペクトルウインドウ上の横軸スケールの表示幅 (スパン) の波長を設定 / 問い合わせします。

構文
```
:DISPlay:WINDow2:TRACe[:SCALe]:SPAN
[:WAVelength]<wsp><wavelength>
:DISPlay:WINDow2:TRACe[:SCALe]:SPAN
[:WAVelength]?
```
<wavelength>(スパン波長)：<NRf>|MAXimum
MAXimum： 380nm(仕様コード -10 タイプ)
500nm(仕様コード -20 タイプ)
800nm(仕様コード -30 タイプ)

例
```
:DISP:WIND2:TRAC:SPAN 50NM
:DISP:WIND2:TRAC:SPAN? ->
+5.00000000E-008<END>
```

解説
- 問い合わせ結果は、m 単位で返します。
- オーバーラップコマンドです。
- 仕様コード -MW：マルチ波長タイプで有効。

:DISPlay:WINDow2:TRACe[:SCALe]:SPAN:FREQuency

機能　スペクトルウインドウ上の横軸スケールの表示幅 (スパン) の周波数を設定 / 問い合わせします。

構文
```
:DISPlay:WINDow2:TRACe[:SCALe]:SPAN:
FREQuency<wsp><frequency>
:DISPlay:WINDow2:TRACe[:SCALe]:SPAN:
FREQuency?
```
<frequency>(スパン周波数)：<NRf>|MAXimum
MAXimum： 48.37THz(仕様コード -10 タイプ)
73.48THz(仕様コード -20 タイプ)
156.76THz(仕様コード -30 タイプ)

例
```
:DISP:WIND2:TRAC:SPAN:FREQ 20THZ
:DISP:WIND2:TRAC:SPAN:FREQ? ->
+2.00000000E+014<END>
```

解説
- 問い合わせ結果は、Hz 単位で返します。
- オーバーラップコマンドです。
- 仕様コード -MW：マルチ波長タイプで有効。

:DISPlay:WINDow2:TRACe[:SCALe]:SPAN:WNUMber

機能　スペクトルウインドウ上の横軸スケールの表示幅 (スパン) の波数を設定 / 問い合わせします。

構文
```
:DISPlay:WINDow2:TRACe[:SCALe]:SPAN:
WNUMber<wsp><wnumber>
:DISPlay:WINDow2:TRACe[:SCALe]:SPAN:
WNUMber?
```
<wnumber>(スパン波数)：<NRf>|MAXimum
MAXimum： 1815cm^{-1}(仕様コード -10 タイプ)
2450.9cm^{-1}(仕様コード -20 タイプ)
5228.7cm^{-1}(仕様コード -30 タイプ)

例
```
:DISP:WIND2:TRAC:SPAN:WNUM 10000
:DISP:WIND2:TRAC:SPAN:WNUM? ->
+1.00000000E+003<END>
```

解説
- 問い合わせ結果は、m^{-1} 単位で返します。
- オーバーラップコマンドです。
- 仕様コード -MW：マルチ波長タイプで有効。

FETCh Sub System コマンド

概要
- 本サブシステムは、本機器で直近に測定した結果を問い合わせる機能です。ただし、測定動作中は測定の完了を待って結果を返します。詳細は 4.4 節の「動作ステータス変化の例」をご覧ください。
- 本機器の動作には影響を与えません。(関連コマンド：MEAS Sub System、READ Sub System)

:FETCh?

機能　直近のピークの測定結果を問い合わせます。
構文　:FETC?
例　:FETC? -> 3,+6.46241320E+005,
+6.45768650E+005,+6.46714090E+005
<END>
解説
- 直近の問い合わせコマンドがシングル表示用 (コマンドの階層に [:SCALar] があるもの) 場合は、測定結果が 1 つ返ります。
- 直近の問い合わせコマンドがマルチ表示用 (コマンドの階層に :ARRay があるもの) の場合は、測定結果をデータの数だけ返します。
  仕様コード -SW：シングル波長タイプでは、測定結果を 1 つ返します。
  パワーの場合
  <peak_num>,<power1>,<power2>,･･･
  波長の場合
  <peak_num>,<wav1>,<wav2>,･･･
  周波数の場合
  <peak_num>,<freq1>,<freq2>,･･･
  波数の場合
  <peak_num>,<wnum1>,<wnum2>,･･･
  <peak_num>：ピークの数 0 ～ 1024
  <power1>,<power2>,･･･：ピークのパワー
  <wav1>,<wav2>,･･･：ピークの波長
  <freq1>,<freq2>,･･･：ピークの周波数
  <wnum1>,<wnum2>,･･･：ピークの波数
- 電源起動時は、問い合わせ結果は、m 単位の波長で返します。
- オーバーラップコマンドです。

:FETCh:ARRay:POWer?

機能　直近のピークのパワーをマルチ表示で問い合わせます。
構文　:FETCh:ARRay:POWer?<wsp>
[<expected_value>]
<expected_value>(パワー)：
MAXimum|MINimum|DEFault|<NRf>
MAXimum：最大パワーのピークを指定
MINimum：最小パワーのピークを指定
DEFault：現在選択されているピークを指定
<NRf>：指定したパワーに最も近いピークを指定。
例　:FETC:ARR:POW? -> ,-3.99000000E+000,
-7.28000000E+000,-1.08300000E+001<END>
解説
- 測定結果をデータの数だけ返します。
  仕様コード -SW：シングル波長タイプでは、測定結果を 1 つ返します。
  <peak_num>,<power1>,<power2>,･･･
  <peak_num>：ピークの数 0 ～ 1024
  <power1>,<power2>,･･･ ピークのパワー
- パラメータを指定すると、本機器の画面表示上のカレントピーク (選択表示しているピーク) が変わります。
- パラメータをパワー、DEF 以外に指定すると、ピークの自動検索機能は OFF になります。
- 問い合わせ結果は、パラメータにより、dBm または W で返します。
- オーバーラップコマンドです。
- パラメータは、仕様コード -MW：マルチ波長タイプで有効です。

:FETCh:ARRay:POWer:FREQuency?

機能　直近のピークの周波数をマルチ表示で問い合わせます。
構文　:FETCh:ARRay:POWer:FREQuency?<wsp>
[<expected_value>]
<expected_value>(周波数)：
MAXimum|MINimum|DEFault|<NRf>
MAXimum：最大周波数のピークを指定
MINimum：最小周波数のピークを指定
DEFault：現在選択されているピークを指定
<NRf>：指定した周波数に最も近いピークを指定。
例　:FETC:ARR:POW:FREQ? -> 3,
+1.93738272E+014,+1.93596570E+014,
+1.93880006E+014<END>
解説
- 測定結果をデータの数だけ返します。
  仕様コード -SW：シングル波長タイプでは測定結果を 1 つ返します。
  <peak_num>,<freq1>,<freq2>,･･･
  <peak_num>：ピークの数 0 ～ 1024
  <freq1>,<freq2>,･･･　ピークの周波数
- パラメータを指定すると、本機器の画面表示上のカレントピーク (選択表示しているピーク) が変わります。
- パラメータに DEF 以外を指定すると、ピークの自動検索機能は OFF になります。
- 問い合わせ結果は、Hz 単位で返します。
- オーバーラップコマンドです。
- パラメータは、仕様コード -MW：マルチ波長タイプで有効です。

:FETCh:ARRay:POWer:WAVelength?

機能　直近のピークの波長をマルチ表示で問い合わせます。

構文　:FETCh:ARRay:POWer:WAVelength?<wsp>
[<expected_value>]
<expected_value>(波長) :
MAXimum|MINimum|DEFault|<NRf>
MAXimum：最大波長のピークを指定
MINimum：最小波長のピークを指定
DEFault：現在選択されているピークを指定
<NRf>：指定した波長に最も近いピークを指定。

例　:FETC:ARR:POW:WAV? -> 3,
+1.54740958E-006,+1.54854220E-006,
+1.54627836E-006<END>

解説
- 測定結果をデータの数だけ返します。
  仕様コード -SW：シングル波長タイプでは、測定結果を 1 つ返します。
  <peak_num>,<wav1>,<wav2>,･･･
  <peak_num>：ピークの数 0 ～ 1024
  <wav1>,<wav2>,･･･　ピークの波長
- パラメータを指定すると、本機器の画面表示上のカレントピーク (選択表示しているピーク) が変わります。
- パラメータに DEF 以外を指定すると、ピークの自動検索機能は OFF になります。
- 問い合わせ結果は、m 単位で返します。
- オーバーラップコマンドです。
- パラメータは、仕様コード -MW：マルチ波長タイプで有効です。

:FETCh:ARRay:POWer:WNUMber?

機能　直近のピークの波数をマルチ表示で問い合わせます。

構文　:FETCh:ARRay:POWer:WNUMber?<wsp>
[<expected_value>]
<expected_value>(波数) :
MAXimum|MINimum|DEFault|<NRf>
MAXimum：最大波数のピークを指定
MINimum：最小波数のピークを指定
DEFault：現在選択されているピークを指定
<NRf>：指定した波数に最も近いピークを指定。

例　:FETC:ARR:POW:WNUM? -> 3,
+6.46241320E+005,+6.45768650E+005,
+6.46714090E+005<END>

解説
- 測定結果をデータの数だけ返します。
  仕様コード -SW：シングル波長タイプでは、測定結果を 1 つ返します。
  <peak_num>,<wnum1>,<wnum2>,･･･
  <peak_num>：ピークの数 0 ～ 1024
  <wnum1>,<wnum2>,･･･　ピークの波数
- パラメータを指定すると、本機器の画面表示上のカレントピーク (選択表示しているピーク) が変わります。
- パラメータに DEF 以外を指定すると、ピークの自動検索機能は OFF になります。
- 問い合わせ結果は、m^{-1} 単位で返します。
- オーバーラップコマンドです。
- パラメータは、仕様コード -MW：マルチ波長タイプで有効です。

:FETCh[:SCALar]:POWer?

機能　直近のピークのパワーをシングル表示で問い合わせます。

構文　:FETCh[:SCALar]:POWer?<wsp>
[<expected_value>]
<expected_value>(パワー) :
MAXimum|MINimum|DEFault|<NRf>
MAXimum：最大パワーのピークを指定
MINimum：最小パワーのピークを指定
DEFault：現在選択されているピークを指定
<NRf>：指定したパワーに最も近いピークを指定。

例　:FETC:POW? -> -7.28000000E+000<END>

解説
- パラメータに DEF 以外を指定すると、ピークの自動検索機能は OFF になります。
- 問い合わせ結果は、パラメータにより、dBm または W で返します。
- オーバーラップコマンドです。
- パラメータは、仕様コード -MW：マルチ波長タイプで有効です。

:FETCh[:SCALar]:POWer:FREQuency?

機能　直近のピークの周波数をシングル表示で問い合わせます。

構文　:FETCh[:SCALar]:POWer:FREQuency?<wsp>
[<expected_value>]
<expected_value>(周波数) :
MAXimum|MINimum|DEFault|<NRf>
MAXimum：最大周波数のピークを指定
MINimum：最小周波数のピークを指定
DEFault：現在選択されているピークを指定
<NRf>：指定した周波数に最も近いピークを指定。

例　:FETC:POW:FREQ?
-> +1.93596570E+014<END>

解説
- パラメータに DEF 以外を指定すると、ピークの自動検索機能は OFF になります。
- 問い合わせ結果は、Hz 単位で返します。
- オーバーラップコマンドです。
- パラメータは、仕様コード -MW：マルチ波長タイプで有効です。

:FETCh[:SCALar]:POWer:WAVelength?

機能　直近のピークの波長をシングル表示で問い合わせます。

構文　:FETCh[:SCALar]:POWer:WAVelength?<wsp>[<expected_value>]
<expected_value>(波長) :
MAXimum|MINimum|DEFault|<NRf>
MAXimum：最大波長のピークを指定
MINimum：最小波長のピークを指定
DEFault：現在選択されているピークを指定
<NRf>：指定した波長に最も近いピークを指定。

例　:FETC:POW:WAV? ->
+1.54854220E-006<END>

解説
- パラメータに DEF 以外を指定すると、ピークの自動検索機能は OFF になります。
- 問い合わせ結果は、m 単位で返します。
- オーバーラップコマンドです。
- パラメータは、仕様コード -MW：マルチ波長タイプで有効です。

:FETCh[:SCALar]:POWer:WNUMber?

機能　直近のピークの波数をシングル表示で問い合わせます。

構文　:FETCh[:SCALar]:POWer:WNUMber?<wsp>[<expected_value>]
<expected_value>(波数) :
MAXimum|MINimum|DEFault|<NRf>
MAXimum：最大波数のピークを指定
MINimum：最小波数のピークを指定
DEFault：現在選択されているピークを指定
<NRf>：指定した波数に最も近いピークを指定。

例　:FETC:POW:WNUM?
-> +6.45768650E+005<END>

解説
- パラメータに DEF 以外を指定すると、ピークの自動検索機能は OFF になります。
- 問い合わせ結果は、m^{-1} 単位で返します。
- オーバーラップコマンドです。
- パラメータは、仕様コード -MW：マルチ波長タイプで有効です。

FORMat Sub System コマンド

:FORMat:NDATa[:WAVelength]

機能　ピークを検出していないときの応答値を設定 / 問い合わせします。

構文　:FORMat:NDATa[:WAVelength]<wsp><wavelength>
:FORMat:NDATa[:WAVelength]?
<wavelength>: 0 ～ 300nm
<NRf>

例　:FORM:NDAT 100NM
:FORM:NDAT? -> +1.00000000E-007<END>

解説
- 下記コマンドに有効です。
  :FETC:SCAL:{FREQ|WAV|WNUM}?
  :MEAS:SCAL:{FREQ|WAV|WNUM}?
  :READ:SCAL:{FREQ|WAV|WNUM}?

MEASure Sub System コマンド

概要

- 本サブシステムは、本機器を測定動作させて測定が完了した測定結果を問い合わせる機能です。操作キーから実行したときと同じように、コマンド実行により本機器の画面設定の View Mode を切替えるため、画面表示内容が切り替わります。
- 本機器が測定停止中の場合は、Single 測定を実行し完了してから測定結果を返します。
- 本測定器が測定動作中 (Repeat 測定) の場合は、実行エラーを返します。
- 平均化測定を実行した場合は、平均化された測定結果を返します。
- MEASure[:SCALar] コマンドを実行すると、本機器の画面表示がシングル表示になり、測定結果を 1 つ返します。
- MEASure:ARRay コマンドを実行すると、本機器の画面表示がマルチ表示になり、測定結果をデータの数だけ返します (最大 1024 個)。仕様コード -SW：シングル波長タイプでは、測定結果を 1 つ返します。
- 本機器の設定内容を変更しないで測定結果を問い合わせをする場合は、READ Sub System コマンドをご使用ください。(関連コマンド：FETCh Sub System、READ Sub System)

:MEASure:ARRay:POWer?

機能　Single 測定時 (View Mode が MULTI に設定される) のピークのパワーをマルチ表示で問い合わせます。

構文　`:MEASure:ARRay:POWer?<wsp>[<expected_value>]`
<expected_value>(パワー)：
MAXimum|MINimum|DEFault|<NRf>
MAX：最大パワーのピークを指定
MIN：最小パワーのピークを指定
DEF：現在選択されているピークを指定
<NRf>：指定したパワーに最も近いピークを指定。dBm 単位でも W 単位でも指定可能で、単位を省略すると W 単位になります。

例　`:MEAS:ARR:POW? -> 3,`
`-3.97000000E+000,-7.31000000E+000,`
`-1.08700000E+001<END>`

解説
- 測定結果をデータの数だけ返します。
  仕様コード -SW：シングル波長タイプでは、測定結果を 1 つ返します。
  <peak_num>,<power1>,<power2>,･･･
  <peak_num>：ピークの数 0 ～ 1024
  <power1>,<power2>,･･･：ピークのパワー
- パラメータを指定すると、本機器の画面表示上のカレントピーク (選択表示しているピーク) が変わります。
- パラメータをパワー、DEF 以外に指定すると、ピークの自動検索機能は OFF になります。
- 問い合わせ結果は、パラメータにより、dBm または W で返します。
- パラメータは、仕様コード -MW：マルチ波長タイプで有効です。

:MEASure:ARRay:POWer:FREQuency?

機能　Single 測定時 (View Mode が MULTI に設定される) のピークの周波数をマルチ表示で問い合わせます。

構文　`:MEASure:ARRay:POWer:FREQuency?`
`<wsp>[<expected_value>]`
<expected_value>(周波数)：
MAXimum|MINimum|DEFault|<NRf>
MAX：最大周波数のピークを指定
MIN：最小周波数のピークを指定
DEF：現在選択されているピークを指定
<NRf>：指定した周波数に最も近いピークを指定

例　`:MEAS:ARR:POW:FREQ? -> 3,`
`+1.93738414E+014,+1.93596724E+014,`
`+1.94163516E+014<END>`

解説
- 測定結果をデータの数だけ返します。
  仕様コード -SW：シングル波長タイプでは、測定結果を 1 つ返します。
  <peak_num>,<freq1>,<freq2>,･･･
  <peak_num>：ピークの数 0 ～ 1024
  <freq1>,<freq2>,･･･：ピークの周波数
- パラメータを指定すると、本機器の画面表示上のカレントピーク (選択表示しているピーク) が変わります。
- パラメータに DEF 以外を指定すると、ピークの自動検索機能は OFF になります。
- 問い合わせ結果は、Hz 単位で返します。
- パラメータは、仕様コード -MW：マルチ波長タイプで有効です。

:MEASure:ARRay:POWer:WAVelength?

機能　Single 測定時 (View Mode が MULTI に設定される) のピークの波長をマルチ表示で問い合わせます。

構文　`:MEASure:ARRay:POWer:WAVelength?<wsp>[<expected_value>]`
<expected_value>(波長)：
MAXimum|MINimum|DEFault|<NRf>
MAX：最大波長のピークを指定
MIN：最小波長のピークを指定
DEF：現在選択されているピークを指定
<NRf>：指定した波長に最も近いピークを指定

例　`:MEAS:ARR:POW:WAV? -> 3,`
`+1.54740844E-006,+1.54854097E-006,`
`+1.54402055E-006<END>`

解説
- 測定結果をデータの数だけ返します。
  仕様コード -SW：シングル波長タイプでは、測定結果を 1 つ返します。
  <peak_num>,<wav1>,<wav2>,･･･
  <peak_num>：ピークの数 0 ～ 1024
  <wav1>,<wav2>,･･･：ピークの波長
- パラメータを指定すると、本機器の画面表示上のカレントピーク (選択表示しているピーク) が変わります。
- パラメータに DEF 以外を指定すると、ピークの自動検索機能は OFF になります。
- 問い合わせ結果は、m 単位で返します。
- パラメータは、仕様コード -MW：マルチ波長タイプで有効です。

:MEASure:ARRay:POWer:WNUMber?

機能　Single 測定時 (View Mode が MULTI に設定される) のピークの波数をマルチ表示で問い合わせます。

構文　:MEASure:ARRay:POWer:WNUMber?<wsp>[<expected_value>]
<expected_value>(波数) :
MAXimum|MINimum|DEFault|<NRf>
MAX：最大波数のピークを指定
MIN：最小波数のピークを指定
DEF：現在選択されているピークを指定
<NRf>：指定した波数に最も近いピークを指定

例　:MEAS:ARR:POW:WNUM? -> 3,
+6.46241790E+005,+6.45769160E+005,
+6.47659780E+005<END>

解説
- 測定結果をデータの数だけ返します。
  仕様コード -SW：シングル波長タイプでは、測定結果を 1 つ返します。
  <peak_num>,<wnum1>,<wnum2>,･･･
  <peak_num>：ピークの数 0 ～ 1024
  <wnum1>,<wnum2>,･･･：ピークの波数
- パラメータを指定すると、本機器の画面表示上のカレントピーク (選択表示しているピーク) が変わります。
- パラメータに DEF 以外を指定すると、ピークの自動検索機能は OFF になります。
- 問い合わせ結果は、m^{-1} 単位で返します。
- パラメータは、仕様コード -MW：マルチ波長タイプで有効です。

:MEASure[:SCALar]:POWer?

機能　Single 測定時 (View Mode が SINGLE に設定される) のピークのパワーをシングル表示で問い合わせます。

構文　:MEASure[:SCALar]:POWer?<wsp>[<expected_value>]
<expected_value>(パワー) :
MAXimum|MINimum|DEFault|<NRf>
MAX：最大パワーのピークを指定
MIN：最小パワーのピークを指定
DEF：現在選択されているピークを指定
<NRf>：指定したパワーに最も近いピークを指定。dBm 単位でも W 単位でも指定可能で、単位を省略すると W 単位になります。

例　:MEAS:POW? -> -7.84000000E+000<END>

解説
- パラメータを指定すると、本機器の画面表示上のカレントピーク (選択表示しているピーク) が変わります。
- パラメータをパワー、DEF 以外に指定すると、ピークの自動検索機能は OFF になります。
- 問い合わせ結果は、パラメータにより、dBm または W で返します。
- パラメータは、仕様コード -MW：マルチ波長タイプで有効です。

:MEASure[:SCALar]:POWer:FREQuency?

機能　Single 測定時 (View Mode が SINGLE に設定される) のピークの周波数をシングル表示で問い合わせます。

構文　:MEASure[:SCALar]:POWer:FREQuency?<wsp>[<expected_value>]
<expected_value>(周波数) :
MAXimum|MINimum|DEFault|<NRf>
MAX：最大周波数のピークを指定
MIN：最小周波数のピークを指定
DEF：現在選択されているピークを指定
<NRf>：指定した周波数に最も近いピークを指定

例　:MEAS:POW:FREQ?
-> +1.93596757E+014<END>

解説
- パラメータを指定すると、本機器の画面表示上のカレントピーク (選択表示しているピーク) が変わります。
- パラメータに DEF 以外を指定すると、ピークの自動検索機能は OFF になります。
- 問い合わせ結果は、Hz 単位で返します。
- パラメータは、仕様コード -MW：マルチ波長タイプで有効です。

:MEASure[:SCALar]:POWer:WAVelength?

機能　Single 測定時 (View Mode が SINGLE に設定される) のピークの波長をシングル表示で問い合わせます。

構文　:MEASure[:SCALar]:POWer:WAVelength?<wsp>[<expected_value>]
<expected_value>(波長) :
MAXimum|MINimum|DEFault|<NRf>
MAX：最大波長のピークを指定
MIN：最小波長のピークを指定
DEF：現在選択されているピークを指定
<NRf>：指定した波長に最も近いピークを指定

例　:MEAS:POW:WAV?
-> +1.54854010E-006<END>

解説
- パラメータを指定すると、本機器の画面表示上のカレントピーク (選択表示しているピーク) が変わります。
- パラメータに DEF 以外を指定すると、ピークの自動検索機能は OFF になります。
- 問い合わせ結果は、m 単位で返します。
- パラメータは、仕様コード -MW：マルチ波長タイプで有効です。

:MEASure[:SCALar]:POWer:WNUMber?

機能　Single 測定時 (View Mode が SINGLE に設定される) のピークの波数をシングル表示で問い合わせます。

構文　:MEASure[:SCALar]:POWer:WNUMber?<wsp>[<expected_value>]
<expected_value>(波数) :
MAXimum|MINimum|DEFault|<NRf>
MAX：最大波数のピークを指定
MIN：最小波数のピークを指定
DEF：現在選択されているピークを指定
<NRf>：指定した波数に最も近いピークを指定

例　:MEAS:POW:WNUM?
-> +6.45769370E+005<END>

解説
- パラメータを指定すると、本機器の画面表示上のカレントピーク (選択表示しているピーク) が変わります。
- パラメータに DEF 以外を指定すると、ピークの自動検索機能は OFF になります。
- 問い合わせ結果は、m^{-1} 単位で返します。
- パラメータは、仕様コード -MW：マルチ波長タイプで有効です。

MMEMory Sub System コマンド

概要
- <"filename"> にディレクトリ名を含む場合は、以下の方法で指定します。
  - 絶対パス指定
    <"file name"> の先頭が文字 "\" の場合は絶対パス指定。
  - 相対パス指定
    <"file name"> の先頭が文字 "\" 以外の場合は、現在のカレントディレクトリからの相対パス指定。
    カレントディレクトリは、:MMEMory:CDIRectory コマンドで指定します。
- INTernal| EXTernal の指定が省略された場合は、カレントドライブに対するアクセスになります。
  カレントドライブは、:MMEMory:CDRive コマンドで指定します。
- ファイルの保存時にファイル名の拡張子を省略した場合、データの種類に応じた拡張子が付加されます。

:MMEMory:CATalog?

機能　カレントディレクトリの全ファイルリストを問い合わせます。

構文　:MMEMory:CATalog?<wsp>[{<"directory">|ROOT}[,INTernal|EXTernal]]
<"directory">：任意のディレクトリ。下の階層のディレクトリは、"\" で記述
ROOT：ルートディレクトリ
INTernal：内部メモリ
EXTernal：USB メモリ

例
```
:MMEM:CAT? "\Data\test" ->
3
\Data\test

<DIR> result
24.5KB data.csv
12.3KB image.bmp
<END>
```

解説　上記の例では、\Data\test のディレクトリの中に 1 つのディレクトリと 2 つのファイルがあります。ファイルの場合は、ファイル名の前にファイル容量が表示されます。
応答フォーマット：(改行されて表示します)
- ファイル / ディレクトリの数
- カレントディレクトリ
- 空の行 (1 行空ける)
- ファイル / ディレクトリ名 (1 つ表示すると改行して次を表示)
- オーバーラップコマンドです。

:MMEMory:CDIRectory

機能　カレントディレクトリを変更します。
構文　:MMEMory:CDIRectory<wsp>
<"directory">|ROOT[,INTernal|
EXTernal]
<"directory">：任意のディレクトリ。下の階層の
ディレクトリは、"\" で記述
ROOT：ルートディレクトリ
INTernal：内部メモリ
EXTernal：USB メモリ
例　:MMEM:CDIR "MYDIRECTORY"
解説　オーバーラップコマンドです。

:MMEMory:CDRive

機能　カレントドライブを設定 / 問い合わせします。
構文　:MMEMory:CDRive<wsp>[INTernal|
EXTernal]
:MMEMory:CDRive?
INTernal：内部メモリ
EXTernal：USB メモリ
例　:MMEM:CDR EXT
:MMEM:CDR? -> EXT<END>
解説　オーバーラップコマンドです。

:MMEMory:COPY

機能　指定したファイルをコピーします。
構文　:MMEMory:COPY<wsp><"source_file_
name">
[INTernal|EXTernal],<"dest_file_name">
[,INTernal|EXTernal]
<"source_file_name">：コピー元ファイル名
<"dest_file_name">：コピー先ファイル名
INTernal：内部メモリ
EXTernal：USB メモリ
例　:MMEM:COPY "test1.csv",,"test2.csv"
解説　オーバーラップコマンドです。

:MMEMory:DATA?

機能　指定したファイルのデータを読み込みます。
構文　:MMEMory:DATA?<wsp><"filename">
[,INTernal|EXTernal]
<"filename">：データを読み込むファイル名
INTernal：内部メモリ
EXTernal：USB メモリ
例　:MMEM:DATA? "data2.csv"
-> #238YOKOGAWA AQ6151
Data1, 2,3
Data2, 3,4
<END>
解説　・ 応答データはブロックデータ形式で返します。
・ データがバッファに入りきらない場合は Too much data エラー (223) を返します。
・ オーバーラップコマンドです。

:MMEMory:DELete

機能　指定したファイルを削除します。
構文　:MMEMory:DELete<wsp><"filename">
[,INTernal|EXTernal]
<"filename">：消去対象のファイル名
INTernal：内部メモリ
EXTernal：USB メモリ
例　:MMEM:DEL "data1.txt"
解説　オーバーラップコマンドです。

:MMEMory:INFormation?

機能　指定したファイルの情報を問い合わせます。
構文　:MMEMory:INFormation?<wsp>
<"filename">[,INTernal|EXTernal]
<"filename">：取得対象のファイル名
INTernal：内部メモリ
EXTernal：USB メモリ
例　:MMEM:INF? "data1.txt"
-> 1024,2014/09/01 11:55:23<END>
解説　・ <"filename"> ファイルのサイズとタイムスタンプを返します。
<file_size>,<time_stamp>
<file_size>：対象ファイルのサイズを byte 単位で 10 進数表記
<time_stamp>：
更新日時を yyyy/mm/dd hh:mm:ss 表記
・ 指定したファイルが無い場合には USB Storage not inserted(30)、USB Storage not initialized(31) または File not found(33) のエラーを返します。
・ オーバーラップコマンドです。

:MMEMory:LOAD

機能　指定した設定ファイルを本機器に読み込みます。
構文　:MMEMory:LOAD<wsp><"filename">
[,INTernal|EXTernal]
INTernal：内部メモリ
EXTernal：USB メモリ
例　:MMEM:LOAD "SETTING1"
解説　・ ファイル名の拡張子は省略できます。
・ オーバーラップコマンドです。

:MMEMory:MDIRectory

機能　ディレクトリを新規に作成します。
構文　:MMEMory:MDIRectory<wsp>
<"directory_name">[,INTernal|EXTernal]
<"directory_name">：作成するディレクトリ名
INTernal：内部メモリ
EXTernal：USB メモリ
例　:MMEM:MDIR "MYDIR"
解説　オーバーラップコマンドです。

:MMEMory:PWDirectory?

機能　カレントディレクトリを問い合わせます。
構文　:MMEMory:PWDirectory?
例　:MMEM:PWD? -> \MYDIR<END>
解説　オーバーラップコマンドです。

:MMEMory:REMove

機能　USB メモリを取り外せる状態にします。
また、USB ストレージメディアを取り外せる状態かを問い合わせます。

構文　:MMEMory:REMove
:MMEMory:REMove?
応答　0：取り外し可能
1：取り外し不可

例　:MMEM:REM
:MMEM:REM? -> 1<END>

解説　オーバーラップコマンドです。

:MMEMory:REName

機能　指定したファイルのファイル名を変更します。

構文　:MMEMory:REName<wsp><"new_file_name">,<"old_file_name">[,INTernal|EXTernal]
<"new_file_name">：新しいファイル名
<"old_file_name">：古いファイル名
INTernal：内部メモリ
EXTernal：USB メモリ

例　:MMEM:REN "test1.csv","test2.csv"

解説　オーバーラップコマンドです。

:MMEMory:STORe

機能　波長データ、設定データ、画面イメージまたはロギングデータをファイルに保存します。

構文　:MMEMory:STORe<wsp><source>,<"filename">[,INTernal|EXTernal]
<source>(データの種類)：
TABLe|SETup|SIMage1|SIMage2|SIMage3|DLOGging1|DLOGging2
TABLe：波長データ
SETup：設定データ
SIMage1：画面イメージ (白黒)
SIMage2：画面イメージ (カラー)
SIMage3：画面イメージ (カラー、背景色なし)
DLOGging1：ロギングデータ (バイナリ)
DLOGging2：ロギングデータ (CSV)
<"filename">：ファイル名
INTernal：内部メモリ
EXTernal：USB メモリ

例　:MMEM:STOR SET,"SETTINGS"

解説
- ファイル名の拡張子は自動的に付与されます。
- オーバーラップコマンドです。

READ Sub System コマンド

概要
- 本サブシステムは、本機器を測定動作させて測定が完了した測定結果を問い合わせる機能です。本機器の設定内容を変更しないで問い合わせができます (コマンド実行による本機器の画面設定の View Mode は切り替わりません)。
- 本機器が測定停止中の場合は、Single 測定を実行し完了してから測定結果を返します。
- 本測定器が測定動作中 (Repeat 測定) の場合は、実行エラーを返します。
- 平均化測定を実行した場合は、平均化された測定結果を返します。
- READ[:SCALar] コマンドを実行すると、測定結果を 1 つ返します。
- READ:ARRay コマンドを実行すると、測定結果をデータの数だけ返します (最大 1024 個)。仕様コード -SW：シングル波長タイプでは、測定結果を 1 つ返します。
- コマンドを実行しても、本機器の画面表示 (シングル表示 / マルチ表示) は切り替わりません。(関連コマンド：FETCh Sub System、MEASure Sub System)

:READ?

機能　Single 測定時のピークの測定結果を問い合わせます。

構文　:READ?

例　:READ? -> 3,+6.46241450E+005,+6.45768920E+005,+6.47659390E+005<END>

解説
- 前回の問い合わせコマンドがシングル表示用 (コマンドの階層に [:SCALar] があるもの) の場合は、測定結果が 1 つ返ります。
- 前回の問い合わせコマンドがマルチ表示用 (コマンドの階層に :ARRay があるもの) の場合は、測定結果をデータの数だけ返します。
仕様コード -SW：シングル波長タイプでは、測定結果を 1 つ返します。
パワーの場合
<peak_num>,<power1>,<power2>,･･･
波長の場合
<peak_num>,<wav1>,<wav2>,･･･
周波数の場合
<peak_num>,<freq1>,<freq2>,･･･
波数の場合
<peak_num>,<wnum1>,<wnum2>,･･･
<peak_num>：ピークの数 0 ～ 1024
<power1>,<power2>,･･･：ピークのパワー
<wav1>,<wav2>,･･･：ピークの波長
<freq1>,<freq2>,･･･：ピークの周波数
<wnum1>,<wnum2>,･･･：ピークの波数
- 電源起動時は、問い合わせ結果は、m 単位の波長で返します。

:READ:ARRay:POWer?

機能　Single 測定時のピークのパワーをマルチ表示で問い合わせます。

構文　:READ:ARRay:POWer?<wsp>
[<expected_value>]
<expected_value>(パワー) :
MAXimum|MINimum|DEFault|<NRf>
MAX：最大パワーのピークを指定
MIN：最小パワーのピークを指定
DEF：現在選択されているピークを指定
<NRf>：指定したパワーに最も近いピークを指定

例　:READ:ARR:POW? -> 3,-3.77000000E+000,-7.72000000E+000,-1.04900000E+001<END>

解説
- 測定結果をデータの数だけ返します。
  仕様コード -SW：シングル波長タイプでは、測定結果を 1 つ返します。
  <peak_num>,<power1>,<power2>,･･･
  <peak_num>：ピークの数 0 ～ 1024
  <power1>,<power2>,･･･：ピークのパワー
- パラメータを指定すると、本機器の画面表示上のカレントピーク (選択表示しているピーク) が変わります。
- パラメータを DEF 以外に指定すると、ピークの自動検索機能は OFF になります。
- 問い合わせ結果は、パラメータにより、dBm または W で返します。
- パラメータは、仕様コード -MW：マルチ波長タイプで有効です。

:READ:ARRay:POWer:FREQuency?

機能　Single 測定時のピークの周波数をマルチ表示で問い合わせます。

構文　:READ:ARRay:POWer:FREQuency?<wsp>
[<expected_value>]
<expected_value>(周波数) :
MAXimum|MINimum|DEFault|<NRf>
MAX：最大周波数のピークを指定
MIN：最小周波数のピークを指定
DEF：現在選択されているピークを指定
<NRf>：指定した周波数に最も近いピークを指定

例　:READ:ARR:POW:FREQ? -> 3,
+1.93738284E+014,+1.93596611E+014,
+1.94163376E+014<END>

解説
- 測定結果をデータの数だけ返します。
  仕様コード -SW：シングル波長タイプでは、測定結果を 1 つ返します。
  <peak_num>,<freq1>,<freq2>,･･･
  <peak_num>：ピークの数 0 ～ 1024
  <freq1>,<freq2>,･･･：ピークの周波数
- パラメータを指定すると、本機器の画面表示上のカレントピーク (選択表示しているピーク) が変わります。
- パラメータを DEF 以外に指定すると、ピークの自動検索機能は OFF になります。
- 問い合わせ結果は、Hz 単位で返します。
- パラメータは、仕様コード -MW：マルチ波長タイプで有効です。

:READ:ARRay:POWer:WAVelength?

機能　Single 測定時のピークの波長をマルチ表示で問い合わせます。

構文　:READ:ARRay:POWer:WAVelength?<wsp>
[<expected_value>]
<expected_value>(波長) :
MAXimum|MINimum|DEFault|<NRf>
MAX：最大波長のピークを指定
MIN：最小波長のピークを指定
DEF：現在選択されているピークを指定
<NRf>：指定した波長に最も近いピークを指定

例　:READ:ARR:POW:WAV? -> 3,
+1.54740962E-006,+1.54854218E-006,
+1.54402171E-006<END>

解説
- 測定結果をデータの数だけ返します。
  仕様コード -SW：シングル波長タイプでは、測定結果を 1 つ返します。
  <peak_num>,<wav1>,<wav2>,･･･
  <peak_num>：ピークの数 0 ～ 1024
  <wav1>,<wav2>,･･･：ピークの波長
- パラメータを指定すると、本機器の画面表示上のカレントピーク (選択表示しているピーク) が変わります。
- パラメータを DEF 以外に指定すると、ピークの自動検索機能は OFF になります。
- 問い合わせ結果は、m 単位で返します。
- パラメータは、仕様コード -MW：マルチ波長タイプで有効です。

:READ:ARRay:POWer:WNUMber?

機能　Single 測定時のピークの波数をマルチ表示で問い合わせます。

構文　:READ:ARRay:POWer:WNUMber?<wsp>
[<expected_value>]
<expected_value>(波数) :
MAXimum|MINimum|DEFault|<NRf>
MAX：最大波数のピークを指定
MIN：最小波数のピークを指定
DEF：現在選択されているピークを指定
<NRf>：指定した波数に最も近いピークを指定

例　:READ:ARR:POW:WNUM? -> 3,
+6.46241320E+005,+6.45768650E+005,
+6.46714090E+005<END>

解説
- 測定結果をデータの数だけ返します。
  仕様コード -SW：シングル波長タイプでは、測定結果を 1 つ返します。
  <peak_num>,<wnum1>,<wnum2>,･･･
  <peak_num>：ピークの数 0 ～ 1024
  <wnum1>,<wnum2>,･･･：ピークの波数
- パラメータを指定すると、本機器の画面表示上のカレントピーク (選択表示しているピーク) が変わります。
- パラメータを DEF 以外に指定すると、ピークの自動検索機能は OFF になります。
- 問い合わせ結果は、m^{-1} 単位で返します。
- パラメータは、仕様コード -MW：マルチ波長タイプで有効です。

:READ[:SCALar]:POWer?

機能　Single 測定時のピークのパワーをシングル表示で問い合わせます。

構文　:READ[:SCALar]:POWer?<wsp>
[<expected_value>]
<expected_value>(パワー) :
MAXimum|MINimum|DEFault|<NRf>
MAX：最大パワーのピークを指定
MIN：最小パワーのピークを指定
DEF：現在選択されているピークを指定
<NRf>：指定したパワーに最も近いピークを指定

例　:READ:POW? -> -7.43000000E+000<END>

解説
- パラメータを指定すると、本機器の画面表示上のカレントピーク (選択表示しているピーク) が変わります。
- パラメータを DEF 以外に指定すると、ピークの自動検索機能は OFF になります。
- 問い合わせ結果は、パラメータにより、dBm または W で返します。
- パラメータは、仕様コード -MW：マルチ波長タイプで有効です。

:READ[:SCALar]:POWer:FREQuency?

機能　Single 測定時のピークの周波数をシングル表示で問い合わせます。

構文　:READ[:SCALar]:POWer:FREQuency?<wsp>
[<expected_value>]
<expected_value>(周波数) :
MAXimum|MINimum|DEFault|<NRf>
MAX：最大周波数のピークを指定
MIN：最小周波数のピークを指定
DEF：現在選択されているピークを指定
<NRf>：指定した周波数に最も近いピークを指定

例　:READ:POW:FREQ?
-> +1.93596574E+014<END>

解説
- パラメータを指定すると、本機器の画面表示上のカレントピーク (選択表示しているピーク) が変わります。
- パラメータを DEF 以外に指定すると、ピークの自動検索機能は OFF になります。
- 問い合わせ結果は、Hz 単位で返します。
- パラメータは、仕様コード -MW：マルチ波長タイプで有効です。

:READ[:SCALar]:POWer:WAVelength?

機能　Single 測定時のピークの波長をシングル表示で問い合わせます。

構文　:READ[:SCALar]:POWer:WAVelength?<wsp>
[<expected_value>]
<expected_value>(波長) :
MAXimum|MINimum|DEFault|<NRf>
MAX：最大波長のピークを指定
MIN：最小波長のピークを指定
DEF：現在選択されているピークを指定
<NRf>：指定した波長に最も近いピークを指定

例　:READ:POW:WAV?
-> +1.54854253E-006<END>

解説
- パラメータを指定すると、本機器の画面表示上のカレントピーク (選択表示しているピーク) が変わります。
- パラメータを DEF 以外に指定すると、ピークの自動検索機能は OFF になります。
- 問い合わせ結果は、m 単位で返します。
- パラメータは、仕様コード -MW：マルチ波長タイプで有効です。

:READ[:SCALar]:POWer:WNUMber?

機能　Single 測定時のピークの波数をシングル表示で問い合わせます。

構文　:READ[:SCALar]:POWer:WNUMber?<wsp>
[<expected_value>]
<expected_value>(波数) :
MAXimum|MINimum|DEFault|<NRf>
MAX：最大波数のピークを指定
MIN：最小波数のピークを指定
DEF：現在選択されているピークを指定
<NRf>：指定した波数に最も近いピークを指定

例　:READ:POW:WNUM?
-> +6.45768740E+005<END>

解説
- パラメータを指定すると、本機器の画面表示上のカレントピーク (選択表示しているピーク) が変わります。
- パラメータを DEF 以外に指定すると、ピークの自動検索機能は OFF になります。
- 問い合わせ結果は、m^{-1} 単位で返します。
- パラメータは、仕様コード -MW：マルチ波長タイプで有効です。

SENSe Sub System コマンド

[:SENSe]:CORRection:DEVice

機能　測定光の種類 (Device Mode) を設定 / 問い合わせします。
構文　[:SENSe]:CORRection:DEVice<wsp>NARRow|BROad
[:SENSe]:CORRection:DEVice?
NARRow：CW 光
BROad：Modulation 光
例　:CORR:DEV NARR
:CORR:DEV? -> NARR<END>

[:SENSe]:CORRection:MEDium

機能　光の通過媒体 (MEAS WL) を設定 / 問い合わせします。
構文　[:SENSe]:CORRection:MEDium<wsp>AIR|VACuum
[:SENSe]:CORRection:MEDium?
AIR：標準空気
VACuum：真空
例　:SENS:CORR:MED AIR
:SENS:CORR:MED? -> AIR<END>

[:SENSe]:CORRection:OFFSet[:MAGNitude]

機能　パワーオフセットを設定 / 問い合わせします。
構文　[:SENSe]:CORRection:OFFSet[:MAGNitude]<wsp><offset>
[:SENSe]:CORRection:OFFSet[:MAGNitude]?
<offset>(パワーオフセット)：
<NRf>|MINimum|MAXimum
MINimum：－10db
MAXimum：+10db
例　:CORR:OFFS 1.2
:CORR:OFFS? -> +1.20000000E+000<END>
解説　問い合わせ結果は、dB 単位で返します。

[:SENSe]:URATe

機能　更新レート (波長の測定速度) を設定 / 問い合わせします。
構文　[:SENSe]:URATe<wsp>NORMal|FAST
[:SENSe]:URATe?
NORMal：通常の測定速度でデータを更新
FAST：高速の測定速度でデータを更新
例　:SENS:URAT FAST
:SENS:URAT? -> FAST<END>

STATus Sub System コマンド

概要
STATus コマンドは、ステータスレポートに関する設定と問い合わせを行うグループです。
このグループに相当するフロントパネルのキーはありません。
ステータスレポートについては、第 4 章をご覧ください。

:STATus:OPERation:CONDition?

機能　Operation ステータスの Condition レジスタの内容を問い合わせます。
構文　:STATus:OPERation:CONDition?
例　:STAT:OPER:COND? -> +2048<END>
解説　オーバーラップコマンドです。

:STATus:OPERation:ENABle

機能　Operation ステータスの Enable レジスタの内容を設定 / 問い合わせます。
構文　:STATus:OPERation:ENABle<wsp><integer>
:STATus:OPERation:ENABle?
<integer>：0 ～ 65535
例　:STAT:OPER:ENAB 4095
:STAT:OPER:ENAB? -> +4095<END>
解説　オーバーラップコマンドです。

:STATus:OPERation[:EVENt]?

機能　Operation ステータスの Event レジスタの内容を問い合わせます。
構文　:STATus:OPERation[:EVEnt]?
例　:STAT:OPER? -> +4096<END>
解説　オーバーラップコマンドです。

:STATus:OPERation:NTRansition

機能　Operation ステータスの N Transition レジスタの内容を設定 / 問い合わせします。
構文　:STATus:OPERation:NTRansition<wsp><integer>
:STATus:OPERation:NTRansition?
<integer>：0 ～ 65535
例　:STAT:OPER:NTR 4096
:STAT:OPER:NTR? -> +4096<END>
解説　オーバーラップコマンドです。

:STATus:OPERation:PTRansition

機能　Operation ステータスの P Transition レジスタの内容を設定 / 問い合わせします。
構文　:STATus:OPERation:PTRansition<wsp><integer>
:STATus:OPERation:PTRansition?
<integer>：0 ～ 65535
例　:STAT:OPER:PTR 4096
:STAT:OPER:PTR? -> +4096<END>
解説　オーバーラップコマンドです。

:STATus:PRESet

機能　イベントレジスタをクリアし、イネーブルレジスタの全ビットをセットします。
構文　:STATus:PRESet
例　:STAT:PRES
解説　・ ENABle、NTRansition、PTRansition のレジスタの内容をクリアします。
・ オーバーラップコマンドです。

:STATus:QUEStionable:CONDition?

機能　Questionable ステータスの Condition レジスタの内容を問い合わせます。
構文　:STATus:QUESionable:CONDition?
例　:STAT:QUES:COND? -> +24<END>
解説　オーバーラップコマンドです。

:STATus:QUEStionable:ENABle

機能　Questionable ステータスの Enable レジスタの内容を設定 / 問い合わせします。
構文　:STATus:QUEStionable:ENABle<wsp><integer>
:STATus:QUEStionable:ENABle?
<integer>：0 ～ 65535
例　:STAT:QUES:ENAB 4095
:STAT:QUES:ENAB? -> +4095<END>
解説　オーバーラップコマンドです。

:STATus:QUEStionable[:EVENt]?

機能　Questionable ステータスの Event レジスタの内容を問い合わせます。
構文　:STATus:QUEStionable[:EVENt]?
例　:STAT:QUES? -> +8<END>
解説　オーバーラップコマンドです。

:STATus:QUEStionable:NTRansition

機能　Questionable ステータスの N Transition レジスタの内容を設定 / 問い合わせします。
構文　:STATus:QUEStionable:NTRansition<wsp><integer>
:STATus:QUEStionable:NTRansition?
<integer>：0 ～ 65535
例　:STAT:QUES:NTR 24
:STAT:QUES:NTR? -> +24<END>
解説　オーバーラップコマンドです。

:STATus:QUEStionable:PTRansition

機能　Questionable ステータスの P Transition レジスタの内容を設定 / 問い合わせします。
構文　:STATus:QUEStionable:PTRansition<wsp><integer>
:STATus:QUEStionable:PTRansition?
<integer>：0 ～ 65535
例　:STAT:QUES:PTR 24
:STAT:QUES:PTR? -> +24<END>
解説　オーバーラップコマンドです。

SYSTem Sub System コマンド

:SYSTem:BUZZer[:CLICk]

機能　キーを押下したときのクリック音 (ブザー) の ON/OFF を設定 / 問い合わせします。
構文　:SYSTem:BUZZer[:CLICk]<wsp>0|OFF|1|ON
:SYSTem:BUZZer[:CLICk]?
0|OFF：クリック音 OFF
1|ON：クリック音 ON
例　:SYST:BUZZ ON
:SYST:BUZZ? -> 1<END>
解説　オーバーラップコマンドです。

:SYSTem:BUZZer:WARNing

機能　アラームを検出したときのブザーの ON/OFF を設定 / 問い合わせします。
構文　:SYSTem:BUZZer:WARNing<wsp>0|OFF|1|ON
:SYSTem:BUZZer:WARNing?
0|OFF：ブザー OFF
1|ON：ブザー ON
例　:SYST:BUZZ:WARN ON
:SYST:BUZZ:WARN? -> 1<END>
解説　オーバーラップコマンドです。

:SYSTem:CAPability:WAVelength?

機能　測定可能な波長範囲を問い合わせます。
構文　:SYSTem:CAPability:WAVelength?
例　:SYST:CAP:WAV? ->
+1.27000000E-006,+1.65000000E-006<END>
解説　応答は、測定可能な最小波長 (m)、測定可能な最大波長 (m) の順で返します。

:SYSTem:DATE

機能　日付を設定 / 問い合わせします。
構文　:SYSTem:DATE<wsp><year>,<month>,<day>
:SYSTem:DATE?
<year>：年
<month>：月
<day>：日
例　:SYST:DATE 2012,04,09
:SYST:DATE? -> 2012,04,09<END>
解説　・ 応答は、年、月、日の順で返します。
・ オーバーラップコマンドです。

:SYSTem:ENVironment?

機能　本機器の状態を問い合わせます。
構文　:SYSTem:ENVironment?
例　:SYST:ENV? -> 2.30000000E+001,
2.05000000E+001,9.90000000E+002,-1
<END>
解説　・ 応答は、機内温度 (℃)、干渉計内温度 (℃)、気圧 (hPa)、－ 1(本機器には冷却用ファンはありません) の順で返します。
・ オーバーラップコマンドです。

:SYSTem:ERRor?

機能　本機器のエラー情報を問い合わせます。
構文　:SYSTem:ERRor?
例　:SYST:ERR? -> +0,"No error"<END>
解説　・ 応答は、エラー番号、エラーメッセージの順で返します。
メッセージの詳細は、ユーザーズマニュアル IM AQ6150B-02JA の 4.1 節をご覧ください。
・ オーバーラップコマンドです。

:SYSTem:INFormation?

機能　機種固有情報 (MODEL コードや SPECIAL コード) を問い合わせます。
構文　:SYSTem:INFormation?<wsp>0|1
0：MODEL コード
1：SPECIAL コード
応答　<string>：MODEL コードまたは SPECIAL コード
例　:SYST:INF? 0 -> AQ6150B-10-MW<END>
解説　SPECIAL コード情報が無い場合は "NONE" を返します。

:SYSTem:LANGuage

機能　言語を設定 / 問い合わせします。
構文　:SYSTem:LANGuage<wsp>ENGLish|CHINese|JAPanese
:SYSTem:LANGuage?
ENGLish: 英語
CHINese: 中国語
JAPanese: 日本語
例　:SYSTem:LANG ENGL
:SYSTem:LANG? -> ENGL<END>
解説　オーバーラップコマンドです。

:SYSTem:PRESet

機能　本機器の測定に関する設定条件を初期化します。
構文　:SYSTem:PRESet
例　:SYST:PRES
解説　ブザーやネットワークなどの設定は初期化されません。
初期化の範囲についての詳細は、ユーザーズマニュアル IM AQ6150B-01JA の 7.5 節をご覧ください。

:SYSTem:REFLaser:CONDition?

機能　内蔵基準光源の状態を問い合わせます。
構文　:SYSTem:REFLaser:CONDition?
応答　0：レーザ出力 OFF
1：レーザ起動待ち
2：レーザ安定待ち
3：通常状態
4：交換時期
5：レーザ故障
例　:SYST:REFL:COND? -> 3<END>
解説　・ 本機器では、状態が 2 ～ 4 の場合に測定ができます。
交換時期については、スタートガイド IM AQ6150B-02JA の 2.8 節をご覧ください。
・ オーバーラップコマンドです。

:SYSTem:REFLaser:COUNter?

機能　内蔵基準光源が ON になった回数を問い合わせます。
構文　:SYSTem:REFLaser:COUNter?
例　:SYST:REFL:COUN? -> 40<END>

:SYSTem:REFLaser:OTIMe?

機能　内蔵基準光源の合計稼働時間 (時) を問い合わせます。
構文　:SYSTem:REFLaser:OTIMe?
例　:SYST:REFL:OTIM? -> 100<END>
解説　工場出荷以降に内蔵基準光源が ON になった時間の積算値を表示します。内蔵基準光源の交換の目安となる時間を確認できます。
時間については、スタートガイド IM AQ6150B-02JA の 2.8 節をご覧ください。

:SYSTem:TIME

機能　時刻を設定 / 問い合わせします。
構文　:SYSTem:TIME<wsp><hour>,<minute>,<second>
:SYSTem:TIME?
<hour>：時
<minute>：分
<second>：秒
例　:SYST:TIME 17,20,00
:SYST:TIME? -> 17,20,00<END>
解説　・ 応答は、時、分、秒の順で返します。
・ オーバーラップコマンドです。

:SYSTem:VERSion?

機能　SCPI(Standard Commands for Programmable Interfaces) の版数を問い合わせます。
構文　:SYSTem:VERSion?
例　:SYST:VERS? -> 1999.0<END>
解説　オーバーラップコマンドです。

TRIGger Sub System コマンド

[:TRIGger]:ABORt

機能　測定の動作を停止します。
構文　[:TRIGger]:ABORt
例　:ABOR
解説　オーバーラップコマンドです。

[:TRIGger]:INITiate:CONTinuous

機能　リピート測定を実行 / 問い合わせします。
構文　[:TRIGger]:INITiate:CONTinuous<wsp>0|OFF|1|ON
[:TRIGger]:INITiate:CONTinuous?
0|OFF：リピート測定停止
1|ON：リピート測定実行
例　:INIT:CONT ON
:INIT:CONT? -> 1<END>
解説　リピート測定停止中は被オーバーラップコマンドです。
リピート測定中はオーバーラップコマンドです。

[:TRIGger]:INITiate[:IMMediate]

機能　シングル測定を実行します。
構文　[:TRIGger]:INITiate[:IMMediate]
例　:INIT
解説　・ リピート測定中は本コマンドは無視されます。
・ 被オーバーラップコマンドです。

UNIT Sub System コマンド

:UNIT[:POWer]

機能　パワーの単位を設定 / 問い合わせします。
構文　:UNIT[:POWer]<wsp>W|DBM
:UNIT[:POWer]?
W：W 単位 (ワット)
DBM：dBm 単位
例　:UNIT DBM
:UNIT? -> DBM<END>
解説　・ W に設定したときは、本機器画面上の表示は mW になります。
・ オーバーラップコマンドです。

:UNIT:WL

機能　波長の単位を設定 / 問い合わせします。
構文　:UNIT:WL<wsp>THZ|NM|ICM
:UNIT:WL?
THZ：Hz 単位 (周波数)
NM：m 単位 (波長)
ICM：cm^{-1} 単位 (波数)
例　:UNIT:WL THZ
:UNIT:WL? -> THZ<END>
解説　オーバーラップコマンドです。

付録 1　IEEE 488.2-1992 について

本機器の GP-IB インタフェースは、IEEE 488.2-1992 規格に準じています。この規格では、以下の 23 の項目について「ドキュメントに記載しなければならない」としています。ここでは、これらについて説明しています。

(1) IEEE 488.1 インタフェース機能のうち、サポートしているサブセット
「2.3　GP-IB インタフェースの仕様」を参照してください。

(2) アドレスが 0 ～ 30 以外に設定されたときのデバイスの動作
本機器では、アドレスを 0 ～ 30 以外に設定することはできません。

(3) ユーザーがアドレス変更をしたときの動作
アドレスの変更は SYSTEM キー -> GP-IB ADDRESS ソフトキーでアドレスを設定した時点で行われます。設定したアドレスは、次に変更するまで有効です。

(4) 電源 ON 時のデバイスのセッティング。電源 ON 時に使用可能なコマンド
基本的には、以前の設定 (その前に電源を OFF にしたときの設定) になります。
電源 ON 時に実行を制限されるコマンドはありません。

(5) メッセージ交換のオプション

(a) 入力バッファのサイズ
2M バイト

(b) 複数の応答メッセージユニットを返すクエリ
5 章の各コマンドの例を参照してください。

(c) 構文解析時に応答データを作成するクエリ
すべてのクエリは、構文を解析すると応答データを作成します。

(d) 受信時に応答データを作成するクエリ
コントローラが受信する時点で応答データを作成するクエリはありません。

(e) 制限しあうパラメータを有するコマンド
相互に制限を与えるものはありません。

(6) コマンドを構成する機能エレメントおよび複合ヘッダのエレメントに含まれるもの
5 章を参照してください。

(7) ブロックデータの転送に影響するバッファのサイズ
ヘッダ長も含めて 2M バイト

(8) 演算式で使えるプログラムデータのエレメントの一覧と、そのネストの制限
演算式は使えません。

(9) 各問い合わせに対する応答の構文
5 章の各コマンドの例を参照してください。

(10) 応答の文法に従わないデバイス間の通信について
本機器では、サポートしていません。

(11) 応答データのブロックデータのサイズ
ヘッダ長も含めて 2M バイト

(12) サポートしている共通コマンドの一覧
「5.4　共通コマンド」を参照してください。

(13) キャリブレーション正常終了時のデバイスの状態
*CAL? はサポートしていません。

(14) *DDT のトリガマクロの定義で使用できるブロックデータの最大長
サポートしていません。

(15) マクロ定義のマクロラベルの最大長、マクロ定義で使用できるブロックデータの最大長、マクロ定義で再帰を使ったときの処理
マクロ機能は対応していません。

(16) *IDN? に対する返送
「5.4　共通コマンド」を参照してください。

(17) *PUD、*PUD? のプロテクトユーザーデータの保存エリアのサイズ
*PUD、*PUD? はサポートしていません。

(18) *RDT、*RDT? のリソース名の長さ
*RDT、*RDT? はサポートしていません。

(19) *RST、*LRN?、*RCL、*SAV による状態の変化

***RST、*RCL、*SAV**
「5.4　共通コマンド」を参照してください。

***LRN?**
この共通コマンドはサポートしていません。

(20) *TST? によるセルフテストの実行範囲
「5.4　共通コマンド」を参照してください。

(21) 拡張されたリターンステータスの構造
4 章を参照してください。

(22) 各コマンドの処理がオーバラップするか、シーケンシャルに行われるか
5 章を参照してください。

(23) 各コマンドの実行内容
5 章の各コマンドの機能と、ユーザーズマニュアル IM AQ6150B-01JA、スタートガイド IM AQ6150B-02JA を参照してください。