MT9810B
光テストセット
リモート制御
取扱説明書

第 2 版

製品をご使用前に必ず本書をお読みください。
本書は製品とともに保管してください。

安全にお使いいただくための重要事項は，
MT9810B光テストセット取扱説明書に記載して
ありますのでそちらをお読みください。
本書は製品とともに保管してください。

アンリツ株式会社
メジャメント ソリューションズ

管理番号：M-W1887AW-2.0

MT9810B
光テストセット　リモート制御
取扱説明書

2001年（平成１３年）　６月２８日（初　版）
2002年（平成１４年）　７月　１日（第２版）

商標

Visual BASIC，Windowsは米国マイクロソフト社の登録商標です。NI-488.2M，LabVIEWはナショナルインスツルメンツ社の登録商標です。

はじめに

この説明書では，MT9810B 光テストセットのリモート制御について説明します。
GPIB／RS-232Cインタフェースを通してMT9810Bを制御したり，測定結果を取り込んだりすることができます。

目次

第1章　概要

この章では，MT9810B 光テストセットのリモート制御機能の概要について説明します。

1.1　概要

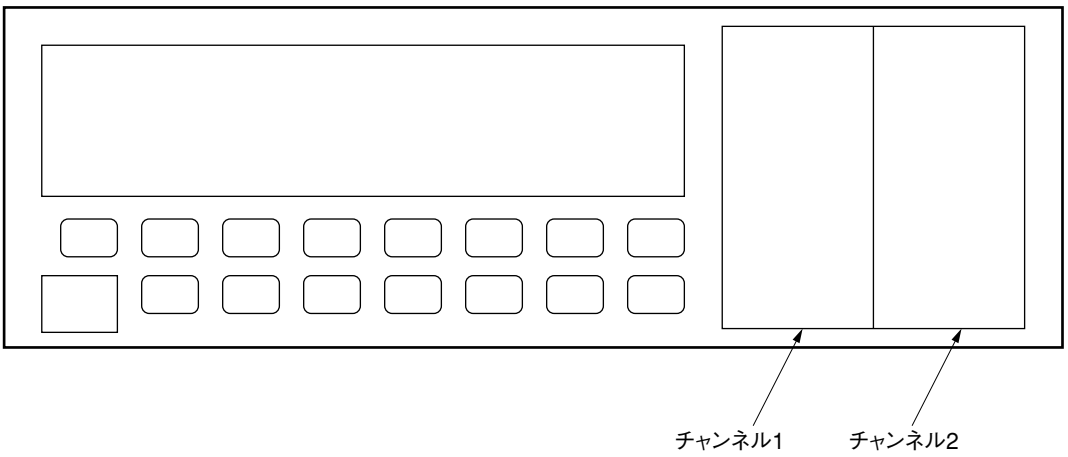
コンピュータなどを使って，MT9810B 光テストセット（以下，「本器」という。）のほとんどの操作をリモートで行うことができます。本器はGPIBインタフェース（IEEE Std 488.2-1987），およびRS-232Cインタフェースポートを装備しています。

1.2　インタフェースポートの選択

インタフェースポートの選択は，パネルで行います。二つのポートは同時に使用できません。
詳細については「第2章　接続方法」を参照してください。

1.3　ユニットのチャンネル番号

MT9810Bは最大2ユニットを装着可能です。コマンドの中には，ユニットの装着されたチャンネル番号を指定するものもあります。正面向かって左側がチャンネル1，右側がチャンネル2です。

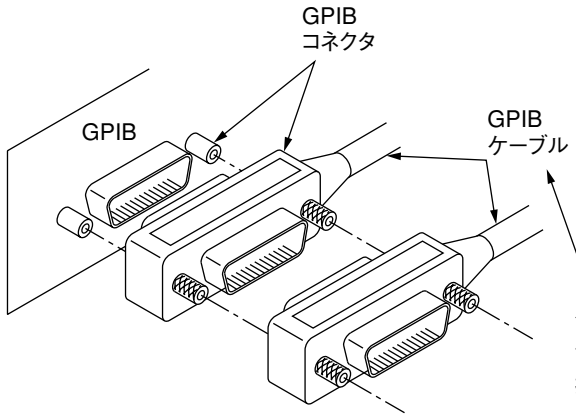


第2章　接続方法

この章では，ホストコンピュータ・パーソナルコンピュータ・プリンタなどの外部機器とのGPIB，RS-232Cケーブルの接続，および本器のインタフェース設定方法について説明します。

2.1　GPIBケーブルによるデバイスの接続

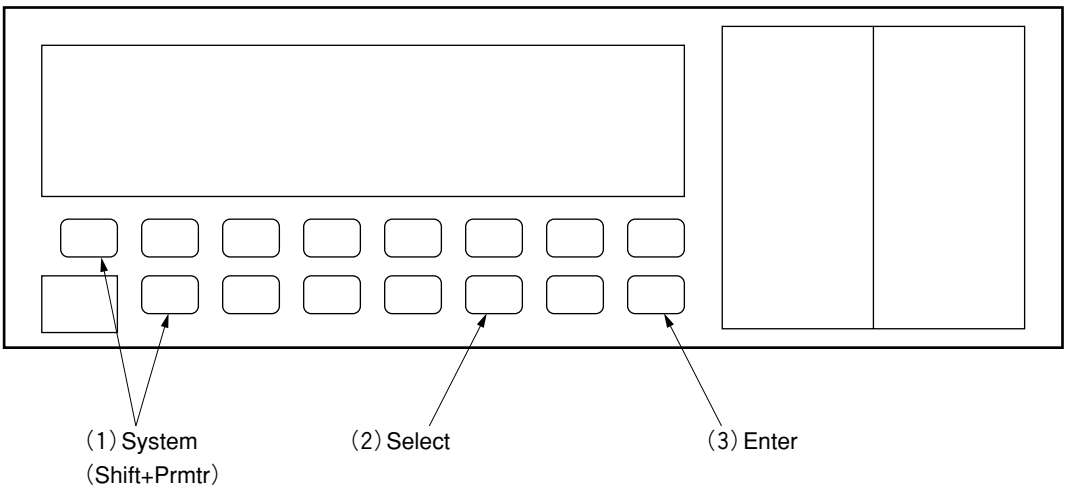
GPIBケーブル接続用コネクタは，背面パネルに取り付けられています。
GPIBケーブルは，電源を投入する前に接続してください。
1つのシステムに接続可能なデバイスの台数は，コントローラを含めて最大15台です。その場合，下図に示す条件に従って接続してください。



2.1.1　接続ポートのインタフェース設定

接続ポートのインタフェースをGPIBに設定します。設定方法は以下のとおりです。

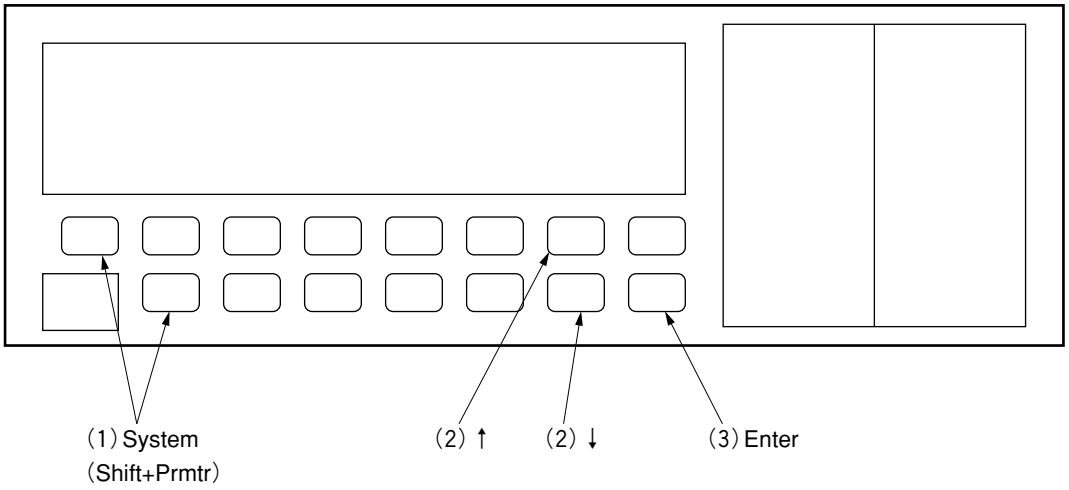
（1）Systemキーで“Remote Interface”を選択
（2）Selectキーで“GPIB”に切り換え
（3）Enterキーで確定



2.1.2　アドレスの確認および設定

MT9810BのGPIBアドレスは，電源投入後に設定します。アドレスの設定は，MT9810Bをローカル状態にしておき，パネル上で行います。

（1） Systemキーで“GPIB ADDRESS”を選択
（2） ↑キーまたは↓キーでアドレスを入力（入力範囲は0～30）
（3） Enterキーで確定



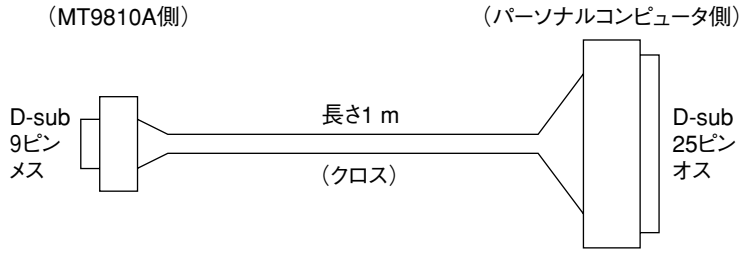
2.2　RS-232Cケーブルによるデバイスの接続

RS-232Cコネクタは，背面パネルに取り付けられています。

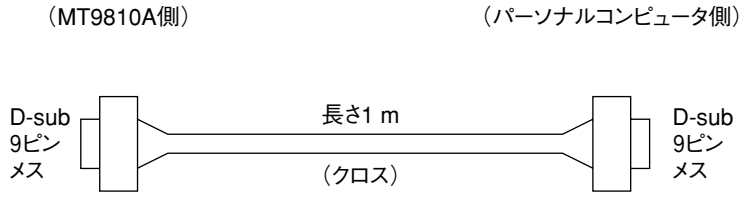
注：

RS-232Cコネクタは，9ピンと25ピンの2種類があります。9ピンはDOS/Vパーソナルコンピュータに，25ピンはNEC PC9801/PC9821シリーズなどに使われています。外部機器のコネクタを確認の上，RS-232Cケーブルを購入してください。なお，本器の応用部品として，下記の2種類のRS-232Cケーブルを用意しています。

●RS-232Cケーブル（25ピンタイプのパーソナルコンピュータ用）

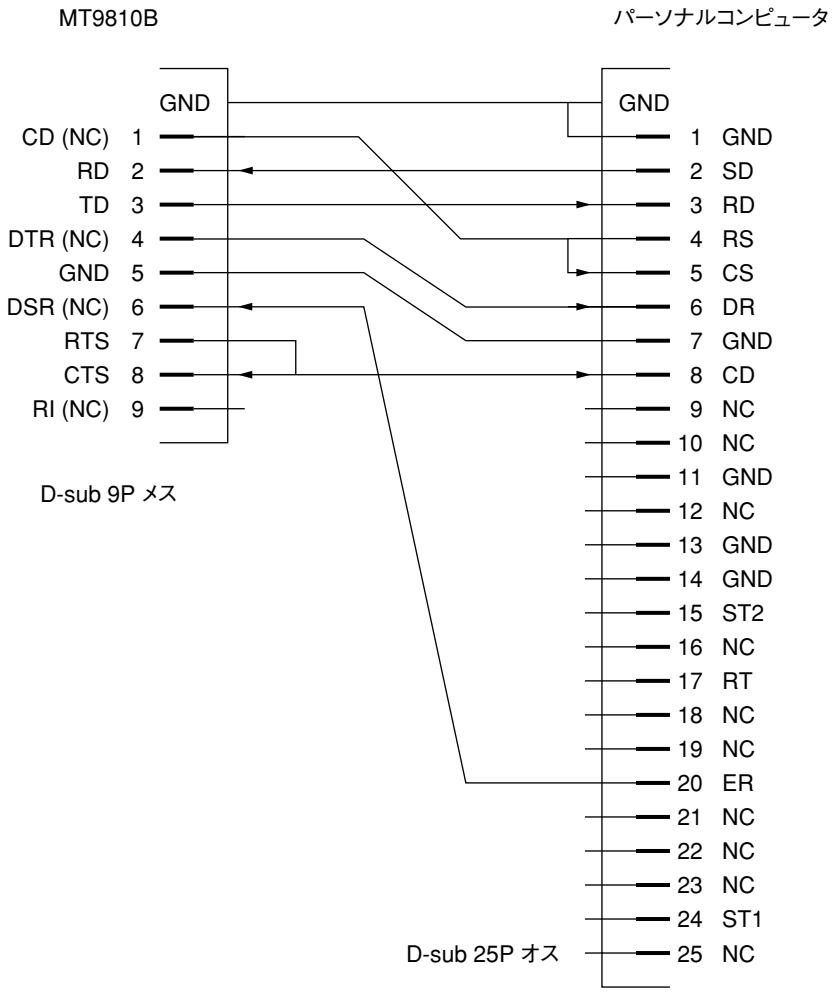


●RS-232Cケーブル（DOS/Vパーソナルコンピュータ用）

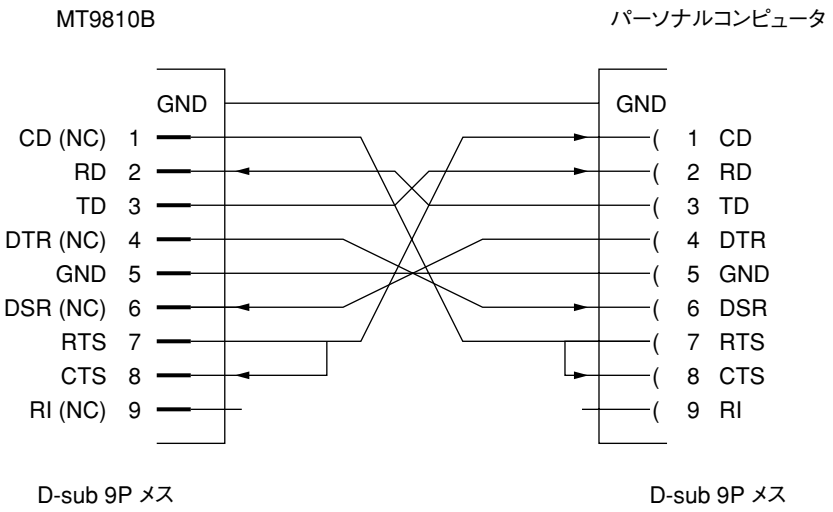


2.2.1 RS-232Cインタフェース信号の接続図

MT9810Bとパーソナルコンピュータの RS-232Cインタフェース信号の接続図を示します。



D-Sub25ピンのインタフェースを持つパーソナルコンピュータとの接続図

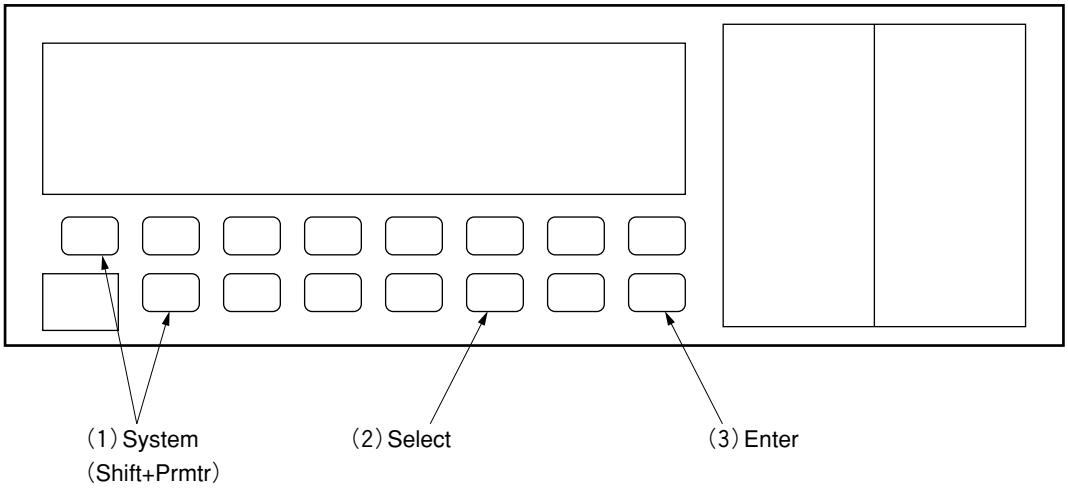


DOS/Vパーソナルコンピュータとの接続図

2.2.2 接続ポートのインタフェース設定

接続ポートのインタフェースをRS-232Cに設定します。設定方法は以下のとおりです。

（1） Systemキーで“Remote Interface”を選択
（2） Selectキーで“RS-232C”に切り換え
（3） Enterキーで確定



2.2.3 RS-232Cインタフェースの条件設定

本器のRS-232Cインタフェースの条件を，接続する外部機器の条件に合わせます。
設定方法は以下のとおりです。

（1） Systemキーで設定項目を選択
（2） Selectキーで設定値を入力
（3） Enterキーで確定

設定項目を表2-1に示します。

表2-1

項目	Systemキー	設定値
ボーレート	RS-232C Baudrate	1200/2400/4800/9600/14400/19200 bps
ストップビット	RS-232C StopBit	1/2 bit
パリティビット	RS-232C ParityBit	ODD/EVEN/NONE
キャラクタ長	RS-232C Character	7/8 bit

2.3　デフォルト値

工場出荷時の設定値を表2-2に示します。

表2-2

設定項目	デフォルト値
リモートインタフェース	GPIB
GPIBアドレス	15
RS-232Cボーレート	9600 bps
RS-232Cストップビット	1 bit
RS-232Cパリティビット	偶数（EVEN）
RS-232Cキャラクタ長	8 bit

第3章　規格

この章では，MT9810BのGPIBの規格とRS-232Cの規格，およびデバイスメッセージリストについて説明します。

3.1　GPIBの規格

MT9810BのGPIBの規格を表3-1に示します。

表3-1

項目	規格値と捕捉説明
機能	IEEE488.2対応 本器をデバイスとして,外部のコントローラから制御。
インタフェースファンクション	SH1:　ソースハンドシェイクの全機能あり。 データ送信のタイミングをとる。 AH1:　アクセプタハンドシェイクの全機能あり。 データ受信のタイミングをとる。 T6:　基本的トーカ機能あり。シリアルポート機能あり。 トークオンリ機能なし。MLAによるトーカ解除機能あり。 L4:　基本的リスナ機能あり。リスンオンリ機能なし。 MTAによるリスナ解除機能あり。 SR1:　サービスリクエスト／ステータスバイトの全機能あり。 RL1:　リモート／ローカル全機能あり。 ローカルロックアウトの機能あり。 PP0:　パラレルポール機能なし。 DC1:　デバイスクリアの全機能あり。 DT0:　デバイストリガの機能なし。 C0:　コントローラ機能なし。

3.2　RS-232Cの規格

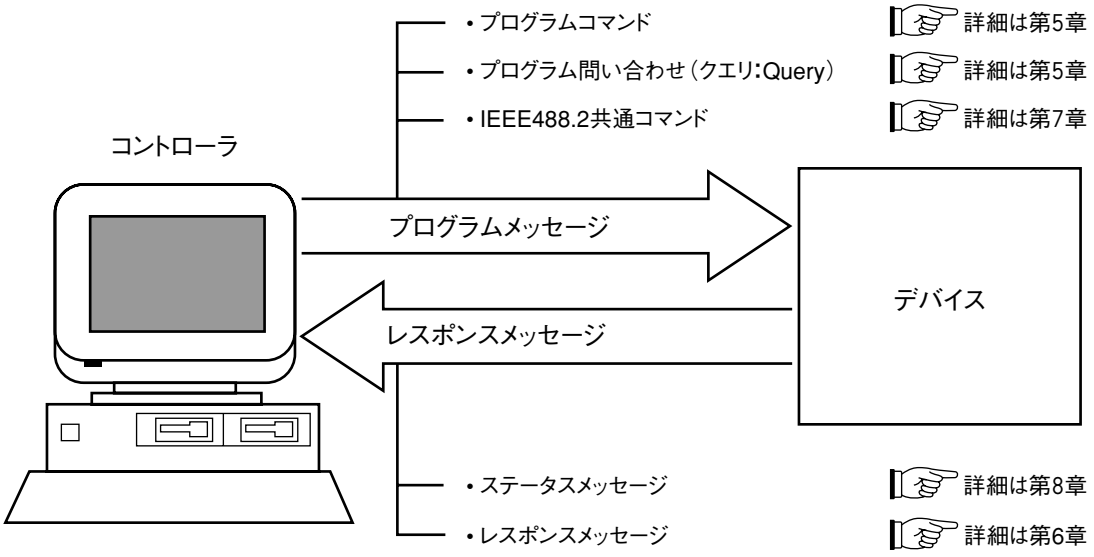
MT9810BのRS-232Cの規格を表3-2に示します。

表3-2

項目	規格値
機能	外部コントローラから制御
通信方式	非同期（調歩同期方式）,半2重
通信制御方式	フロー制御なし
ボーレイト	1200, 2400, 4800, 9600, 14400, 19200 bps
データビット	7 bit,8 bit
パリティ	奇数（ODD）,偶数（EVEN）,なし（NON）
スタートビット	1 bit
ストップビット	1 bit,2 bit
コネクタ	D-sub 9ピン,メス

3.3 デバイスメッセージリスト

デバイスメッセージは，コントローラとデバイス間で送受されるデータメッセージで，プログラムメッセージとレスポンスメッセージの二つがあります。
プログラムメッセージは，コントローラからデバイスへ転送されるASCIIデータメッセージです。プログラムメッセージには，プログラム命令(コマンド：command)およびプログラム問い合わせ(クエリ：Query)の二つがあります。この二つを次ページ以降でリストします。
プログラム命令には，MT9810Bの制御専用に使用される装置固有のコマンドの他に，IEEE488.2共通コマンドがあります。IEEE488.2共通コマンドは，MT9810Bを含み，GPIBインタフェースバス上に接続されたその他のIEEE488.2対応測定器にも共通に適用されるプログラム命令です。
プログラム問い合わせは，デバイスからレスポンスメッセージを得るためのコマンドであって，あらかじめコントローラからデバイスへ転送しておき，その後にデバイスからのレスポンスメッセージをコントローラでアクセプトします。
レスポンスメッセージは，デバイスからコントローラへ転送されるASCIIデータメッセージです。ここでは，その中からステータスメッセージおよびプログラム問い合わせに対応するレスポンスメッセージを次ページ以降でリストします。



なお，データメッセージの中で，プログラムデータおよびレスポンスメッセージでは，数値データの末尾にサフィクス(単位)を付ける場合があります。

以上で述べたメッセージはデバイスの入出力バッファを介してやりとりされます。出力バッファについては出力キューともいいます。入力バッファ，出力バッファについて簡単に下記に説明します。

入力バッファ

DAB（プログラムメッセージや問い合わせメッセージ）の構文を解析し，実行する前に，それらのメッセージを一時的に蓄えておくFIFO（First in First out）タイプのメモリエリアです。
MT9810Bの入力バッファサイズは256バイトあります。

出力キュー

FIFOタイプの持ち行列メモリエリアです。デバイスからコントローラへ出力するすべてのDAB（レスポンスメッセージ）は，コントローラがそれを読み終わるまでは，このメモリに蓄えられます。
MT9810Bの出力キューサイズは256バイトあります。

3.3.1 IEEE488.2共通コマンドとサポート対象コマンド

IEEE488.2規格で定められている39種類の共通コマンドを表3-3に示します。この中からMT9810Bで使用されるIEEE488.2共通コマンドを◎印で示します。

表3-3

ニーモニック	コマンドフルスペル名	IEEE488.2規定	MT9810Aサポートコマンド
*ADD	Accept Address Command	任意	
*CAL	Calibration Query	任意	
*CLS	Clear Status Command	必須	◎
*DDT	Define Device Trigger Command	任意	
*DDT?	Define Device Trigger Query	任意	
*DLF	Disable Listener Function Command	任意	
*DMC	Define Macro Command	任意	
*EMC	Enable Macro Command	任意	
*EMC?	Enable Macro Query	任意	
*ESE	Standard Event Status Enable Command	必須	◎
*ESE?	Standard Event Status Enable Query	必須	◎
*ESR?	Standard Event Status Register Query	必須	◎
*GMC?	Get Macro contents Query	任意	
*IDN?	Identification Query	必須	◎
*IST?	Individual Status Query	任意	
*LMC?	Learn Macro Query	任意	
*LRN?	Learn Device Setup Query	任意	
*OPC	Operation Complete Command	必須	◎
*OPC?	Operation Complete Query	必須	◎
*OPT?	Option Identification Query	任意	◎
*PCB	Pass Control Back Command	C0以外なら必須	
*PMC	Purge Macro Command	任意	
*PRE	Parallel Poll Register Enable Command	任意	
*PRE?	Parallel Poll Register Enable Query	任意	
*PSC	Power On Status Clear Command	任意	
*PSC?	Power On Status Clear Query	任意	
*PUD	Protected User Data Command	任意	
*PUD?	Protected User Data Query	任意	
*RCL	Recall Command	任意	
*RDT	Resource Description Transfer Command	任意	
*RDT?	Resource Description Transfer Query	任意	
*RST	Reset Command	必須	◎
*SAV	Save Command	任意	
*SRE	Service Request Enable Command	必須	◎
*SRE?	Service Request Enable Query	必須	◎
*STB?	Read Status Byte Query	必須	◎
*TRG	Trigger Command	DT1なら必須	
*TST?	Self Test Query	必須	◎
*WAI	Wait to Continue Command	必須	◎

注：

IEEE488.2共通コマンドは，必ず*で始まります。詳細については，「第7章　共通コマンド」を参照してください。

3.3.2　デバイスメッセージ一覧

MT9810B固有のデバイスメッセージ一覧表を表3-4，表3-5，表3-6に示します。コマンドには，HP系とSCPI準拠アンリツオリジナルの2種類があります。その区分も表中に示します。

表3-4　本体

機能	コマンド	HP	SCPI	参照
輝度設定	DISPlay:BRIGhtness	○		第9.1.1項
表示のON/OFF	DISPlay[:STATe]	○		第9.1.2項
カレンダーの設定	SYSTem:DATE	○		第9.1.7項
時刻設定	SYSTem:TIME	○		第9.1.9項
ブザー設定	SYSTem:BEEPer:STATe		○	第9.1.3項
ヘッダ付加の設定	SYSTem:COMMunicate:GPIB:HEAD		○	第9.1.5項
ヘッダ付加の設定	SYSTem:COMMunicate:SERial:HEAD		○	第9.1.6項
挿入ユニットの問い合わせ	SYSTem:CHANnel:STATe		○	第9.1.4項
エラー値の問い合わせ	SYSTem:ERRor		○	第9.1.8項

表3-5 光センサ

機能	コマンド	HP	SCPI	参照
ゼロセットの実行	SENSe[1|2]:CORRection:COLLect:ZERO	○		第9.2.6項
校正係数の設定	SENSe[1|2]:CORRection[:LOSS:[:INPut[:MAGNitude]]]	○		第9.2.7項
オートレンジの設定	SENSe[1|2]:POWer:RANGe:AUTO	○		第9.2.17項
マニュアルレンジの設定	SENSe[1|2]:POWer:RANGe:[UPPer]	○		第9.2.18項
リファレンス値の設定	SENSe[1|2]:POWer:REFerence	○		第9.2.19項
相対値表示	SENSe[1|2]:POWer:REFerence:DISPlay	○		第9.2.20項
リファレンス測定のON/OFF	SENSe[1|2]:POWer:REFernce:STATe	○		第9.2.21項
リファレンス選択	SENSe[1|2]:POWer:REFernce:STATe:RATio	○		第9.2.22項
単位系の切り替え	SENSe[1|2]:POWer:UNIT	○		第9.2.23項
波長の設定	SENSe[1|2]:POWer:WAVelength	○		第9.2.24項
波長の表示単位	SENSe[1|2]:POWer:WAVelength:UNI	○		第9.2.25項
測定データの問い合わせ	FETCh[1|2][:SCALar]:POWer[:DC]	○		第9.2.2項
平均化回数の設定	SENSe[1|2]:AVERage:COUNt		○	第9.2.3項
オート帯域設定	SENSe[1|2]:BANDwidth:AUTO		○	第9.2.5項
帯域設定	SENSe[1|2]:BANDwidth		○	第9.2.4項
変調周波数の設定	SENSe[1|2]:FILTer:BPASs:FREQuency		○	第9.2.11項
測定インターバルの設定	SENSe[1|2]:POWer:INTerval		○	第9.2.16項
測定回数の設定	SENSe[1|2]:TRIGger:COUNt		○	第9.2.26項
記録測定	SENSe[1|2]:INITiate[:IMMediate]		○	第9.2.12項
統計測定の再実行	SENSe[1|2]:TRIGger[:SEQuence][:IMMediate]		○	第9.2.27項
測定の停止	ABORt[1|2]		○	第9.2.1項
記録測定データの読み出し	SENSe[1|2]:MEMory:DATa		○	第9.2.14項
記録測定データ情報	SENSe[1|2]:MEMory:DATa:INFO		○	第9.2.15項
最大値読み出し	SENSe[1|2]:FETCh[:SCALar]:POWer[:DC]:MAXimum		○	第9.2.8項
最小値読み出し	SENSe[1|2]:FETCh[:SCALar]:POWer[:DC]:MINimum		○	第9.2.9項
最大／最小値差読み出し	SENSe[1|2]:FETCh[:SCALar]:POWer[:DC]:PTPeak		○	第9.2.10項
測定条件の保存／取り出し	SENSe[1|2]:MEMory:COPY[:NAME]		○	第9.2.13項
高速転送モード開始	READ[1|2]		○	第9.2.28項
高速転送モード終了	READ[1|2]:ABORt		○	第9.2.29項

表3-6　光源

機能	コマンド	HP	SCPI	参照
変調周波数の設定	SOURce[1|2]:AM[:INTerval]:FREQuency	○		第9.3.1項
減衰量の設定	SOURce[1|2]:POWer:ATTenuation	○		第9.3.3項
光出力の設定	SOURce[1|2]:POWer:STATe	○		第9.3.4項
波長の設定	SOURce[1|2]:POWer:WAVelengt	○		第9.3.5項
測定条件の保存／取り出し	SOURce[1|2]:MEMory[1|2]:COPY[:NAME]		○	第9.3.2項
波長の表示単位	SOURce[1|2]:POWer:WAVelength:UNIT		○	第9.3.6項

[1|2]と書かれた部分には，対象ユニットが挿入されたチャンネル番号(1または2)を入れます。
なお，カギカッコ([　])は不要です。
また，光センサに対して光源用コマンドを送った場合，コマンドエラーになります。逆も同様です。

第4章　イニシャル設定

GPIBインタフェースシステムは，3段階のレベルに分けて初期化されます。第1のレベルは，「バスの初期化」で，システムのバスをアイドル状態にします。第2のレベルは，「メッセージ交換の初期化」で，デバイスをプログラムメッセージ受信可能な状態にします。
第3のレベルは，「デバイスの初期化」で，デバイス特有の機能を初期化します。
これら初期化レベル1，2，3は，いわばデバイスの動作を開始させるための準備に相当します。

GPIBシステムの初期化については，IEEE488.2で，表4-1のように規定されています。

表4-1

レベル	初期化の種類	概要	レベルの組み合わせと順序
1	バスの初期化	コントローラからIFCメッセージによってバスに接続されたすべてのインタフェース機能を初期化します。	他のレベルと組み合わせて使用できますが,レベル1はレベル2の前に実行しなければなりません。
2	メッセージ交換の初期化	GPIBバスコマンドDCLによってGPIB上の全デバイス,またはGPIBバスコマンドSDCによって,指定したデバイスのメッセージ交換の初期化やオペレーションが終了したことをコントローラへ報告する機能を無効にします。	他のレベルと組み合わせて使用できますが,レベル2はレベル3の前に実行しなければなりません。
3	デバイスの初期化	*RSTコマンドによってGPIB上の指定したデバイスだけを,過去の使用状態に関係なく,そのデバイス固有の,既知の状態に戻します。	他のレベルと組み合わせて使用できますが,レベル3はレベル1,レベル2の後で実行しなければなりません。

本器ではRS-232Cインタフェースポートを使用してコントローラから制御する場合には，レベル3「デバイスの初期化」機能が使用可能です。レベル1，2の初期化機能は使用できません。
GPIBインタフェースバスを使用してコントローラから制御する場合には，レベル1，2，3すべての初期化機能が使用可能です。

4.1 IFCステートメントによるバスの初期化

（1）書式

IFC△セレクトコード

（2）解説

本機能はGPIBインタフェースバスを使用してコントローラから制御する場合に使用可能です。

指定したセレクトコードのGPIBにおいて，IFCラインを約100 μsの間アクティブ状態(電気的にLowレベルの状態)にします。IFCを実行すると指定したセレクトコードのGPIBバスラインに接続されているすべてのデバイスのインタフェース機能が初期化されます。システムコントローラのみが送信できます。

インタフェース機能の初期化とは，コントローラによって設定されているデバイスのインタフェース機能の状態(トーカ，リスナ，その他)を解除して初期状態に戻すもので，表4-2で○印の各ファンクションを初期化します。△印は，その一部を初期化します。

表4-2

No	ファンクション	記号	IFCでの初期化
1	ソースハンドシェイク	SH	○
2	アクセプタハンドシェイク	AH	○
3	トーカまたは拡張トーカ	TまたはTE	○
4	リスナまたは拡張リスナ	LまたはLT	○
5	サービス要求	SR	△
6	リモートローカル	RL	
7	パラレルポール	PP	
8	デバイスクリア	DC	
9	デバイストリガ	DT	
10	コントローラ	C	○

IFCステートメントがTrue(IFC文の実行によってIFCラインがLowレベル)でも，レベル2，3に対する初期化は除かれます。したがって，デバイスの動作状態には影響を与えません。

表4-2の中からIFCメッセージによるデバイスの状態のいくつかを表4-3に示します。

表4-3

項目	デバイスの状態
トーカ／リスナ	すべてのトーカ,すべてのリスナは,100 μs以内にアイドル状態（TIDS,LIDS）になります。
コントローラ	コントローラがアクティブ（SACS：System control ACtive State）でなければ,100 μs以内に,コントローラはアイドル状態（CIDS：Controller IDle State）になります。
コントロール権の戻し	IFCを実行したとき,もしシステムコントローラ（GPIB上で最初にコントローラになるよう定められている装置）が他の装置にコントローラとしての機能を移譲している状態であれば,システムコントローラとしての機能が戻されます。なお,一般には,システムコントローラのRESETキーを押せば,システムコントローラからIFCメッセージが出力されます。
サービス要求のデバイス	デバイスがコントローラにSRQメッセージを送信している状態（下図のSRQラインがデバイスによってLOWレベルに設定されている状態）は,解除されませんが,これによって,コントローラがシステムバス下の全デバイスをシリアルポールモード下においている状態は解除されます。
リモート状態のデバイス	現在リモート状態にあるデバイスは,IFCメッセージによって,リモート状態を解除されることはありません。

4.2 DCL，SDCバスコマンドによるメッセージ交換の初期化

(1) 書式

DCL△セレクトコード[プライマリアドレス][セカンダリアドレス]

(2) 解説

本機能はGPIBインタフェースバスを使用してコントローラから制御する場合に使用可能です。
指定したセレクトコードのGPIB上の全デバイス，または指定したデバイスだけの，メッセージ交換に関する初期化を行うステートメントです。

メッセージ交換の初期化の目的は，パネルの設定状態を変える必要はないが，デバイス内部のメッセージ交換に関係する部分が他のプログラムの実行などで，コントローラから制御するには不適当な状態になっている場合に，メッセージ交換の初期化を行うことによって，コントローラから新しい命令を送れるように準備を整えることにあります。

(3) セレクトコードのみ指定した場合

指定したセレクトコードのGPIB上のすべてのデバイスのメッセージ交換の初期化を行います。DCLはGPIBにDCL(Device Clear)バスコマンドを出力します。

(4) アドレスまで指定した場合

指定したデバイスに対するメッセージ交換の初期化を行います。指定したセレクトコードのGPIBにおいてリスナを解除した後，指定したデバイスだけをリスナに設定しSDC(Selected Device Clear)バスコマンドを出力します。

(5) メッセージ交換の初期化対象項目

表4-4

項目	デバイスの状態
入力バッファと出力キュー	クリアされます。
構文解析部･実行制御部･応答作成部	リセットされます。
*RSTを含むデバイスコマンド	これらのコマンドの実行を妨げるすべてのコマンドをクリアします。
対パラメータプログラムメッセージ	対パラメータのため,実行が延期されている部分のコマンドおよび問い合わせもすべて捨てます。
*OPCコマンドの処理	デバイスをOCISステート(Operation Complete Command Idle State)にします。この結果,オペレーション終了ビットを標準イベントステータスレジスタに立てることはできません。 第7章
*OPC?問い合わせの処理	デバイスをOQISステート(Operation Complete Query Idle State)にします。この結果,オペレーション終了ビット1を出力キューに立てることができません。MAV(Message Available)ビットはクリアされます。 第7章
システム構築の自動化	これを実行する*ADDと*DLF共通コマンドを無効にします。(本器では,これらのコマンドをサポートしていません。)
デバイスファンクション	メッセージ交換に関する部分は,すべてアイドル状態におかれます。デバイスは,コントローラからのメッセージを待ち続けます。

下記の事項は，デバイスクリア(DCL)によって処理することを禁じられています。

(a) 現在のデバイスの設定データやストアされているデータを変えること。
(b) フロントパネルI/Oへの割り込み。
(c) 出力キューのクリアにおいて，MAVビットクリア以外に他のステータスバイトを変えること。
(d) 現在進行中のデバイスの動作に影響を与えたり，割り込みを行うこと。

(6) DCL文によるGPIBバスコマンド送出順序

DCL文のGPIBバスコマンドDCL，SDCの送出順序を表4-5に示します。

表4-5

ステートメント	バスコマンド送出順序(ATNライン“LOW”)	データ(ATNライン“HIGH”)
DCLセレクトコード	UNL,DCL	――――――
DCL装置番号	UNL,LISTENアドレス,[2次アドレス],SDC	――――――

4.3 *RSTコマンドによるデバイスの初期化

(1) 書式

*RST

(2) 解説

*RST(Reset)コマンドはIEEE488.2共通コマンドの一つで，デバイスをレベル3でリセットします。
一般にデバイスは装置固有のコマンド(デバイスメッセージ)を使って，さまざまな状態に設定されています。*RSTコマンドは，それらの中でデバイスをある特定の既知の状態を再現するのに使用されます。なお，デバイスのオペレーションの終了を無効にすることについては，レベル2の場合と同じです。

(3) WRITE文の装置番号の指定

指定したアドレスのデバイスをレベル3で初期化します。

(4) デバイス初期化対象項目

表4-6

項目	デバイスの状態
デバイス固有の機能･状態	それまでの来歴にかかわらず,ある既知の状態に戻します。(次ページでリスト表示されています。)
*OPCコマンドの処理	デバイスをOCISステート(Operation Complete Command Idle State)にします。この結果,オペレーション終了ビットを標準イベントステータスレジスタに立てることはできません。第7章
*OPC?問い合わせの処理	デバイスをOQISステート(Operation Complete Query Idle State)にします。この結果,オペレーション終了ビット1を出力キューに立てることができません。MAV(Message Available)ビットはクリアされます。第7章
マクロコマンド	マクロ動作を禁止し,マクロコマンドを受け付けないモードにします。また,マクロ定義を設計者が示す状態に戻します。

注：

*RSTコマンドは，下記の事項には影響を与えません。

1. IEEE488.1インタフェースの状態
2. デバイスアドレス
3. 出力キュー
4. サービスリクエストイネーブルレジスタ
5. 標準ステータスイネーブルレジスタ
6. Power-on-status-clearフラグ設定
7. デバイスの規格に影響する校正データ
8. RS-232Cインタフェース条件

4.4　電源投入時のデバイスの状態

電源が投入されると：

(1)　最後に電源をOFFしたときの状態に設定されます。

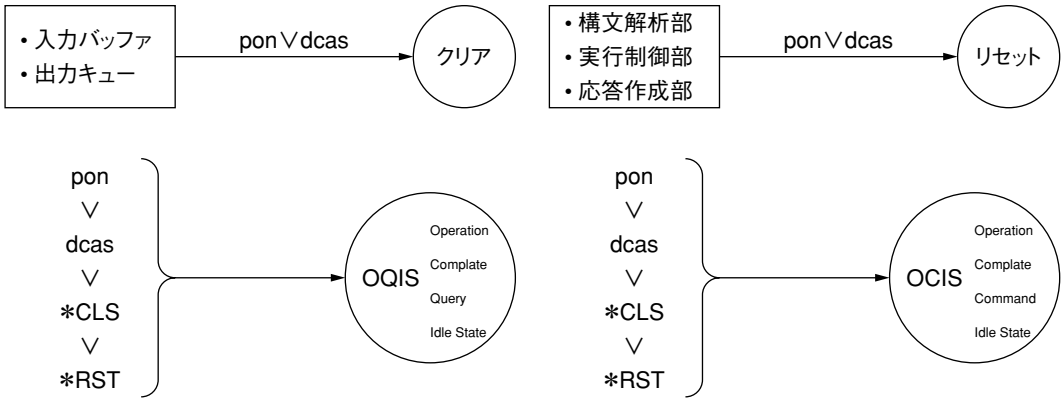
(2)　入力バッファと出力キューは，クリアされます。

(3)　構文解析部・実行制御部・応答作成部は，リセットされます。

(4)　デバイスをOCISステート(Operation Complete Command Idle State)にします。

(5)　デバイスをOQISステート(Operation Complete Query Idle State)にします。

(6)　MT9810Bは，*PSCコマンドを用意していないので，標準イベントステータスレジスタおよび標準イベントステータスイネーブルレジスタは，クリアされます。
イベントはクリア後に記録されます。
(2)～(5)は電源投入以外でも，この状態に設定されますのでそのステートダイヤグラムを下図に示します。



4.4.1　電源投入時に変えない項目

(1) アドレス
(2) 関連するキャリブレーションデータ
(3) 下記のコモンクエリコマンドに対するレスポンスで変化するデータやステート

*IDN? ..... 詳細については，「第7章　共通コマンド」を参照してください。
*OPT? ..... 詳細については，「第7章　共通コマンド」を参照してください。
*PSC? ..... MT9810Bでは，サポートされていません。
*PUD? ..... MT9810Bでは，サポートされていません。
*RDT? ..... MT9810Bでは，サポートされていません。

4.4.2　PSCフラグに関係する項目

PSC(POWER ON STATUS CLEAR)フラグがFalseのとき，サービスリクエストイネーブルレジスタ，標準イベントステータスイネーブルレジスタ，およびパラレルポールイネーブルレジスタは影響されません。
サービスリクエストイネーブルレジスタについては「第8.3項　SRQのイネーブル」，標準イベントステータスイネーブルレジスタについては「第8.4項　標準イベントステータスレジスタ」を参照してください。
また，PSCフラグがTrueか，*PSCコマンドが実行されていないときは，前記レジスタはクリアされます。

注：

PSCコマンドは本器ではサポートされていません。

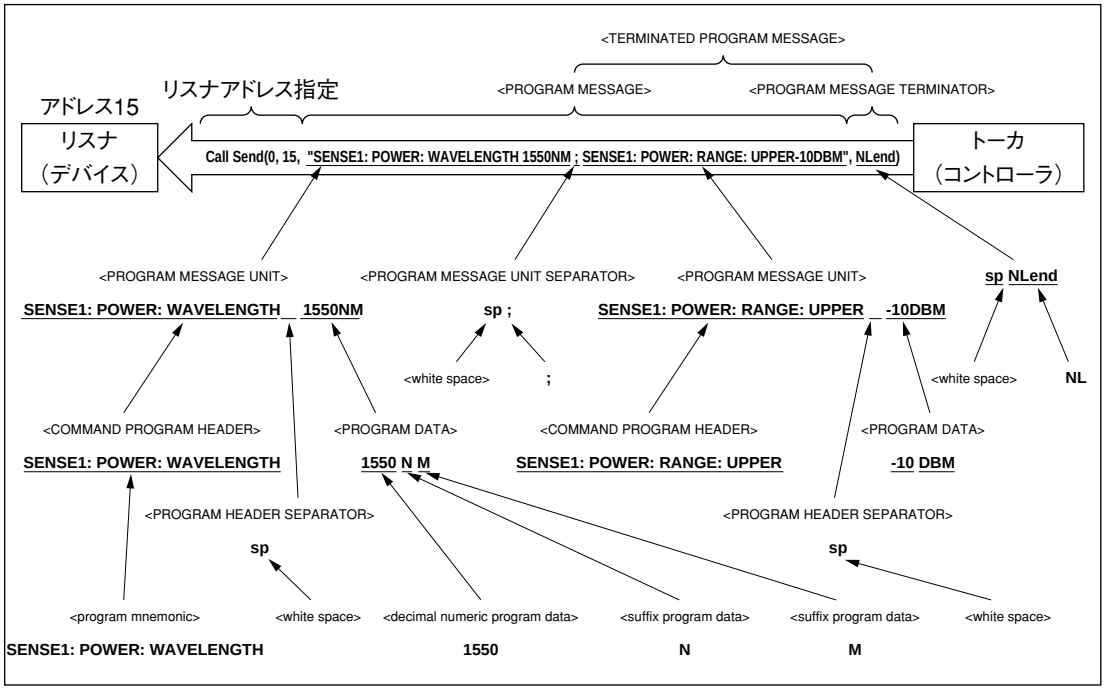
4.4.3　電源投入時に変わる項目

(1) カレントデバイスファンクションステート
(2) ステータス情報
(3) *SAV／*RCLレジスタ(本器ではサポートされていません。)
(4) *DDTコマンドで定義されたマクロ定義(本器ではサポートされていません。)
(5) *DMCコマンドで定義されたマクロ定義(本器ではサポートされていません。)
(6) *EMCコマンドで可能となったマクロ(本器ではサポートされていません。)
(7) *PCBコマンドで受信したアドレス(本器ではサポートされていません。)

第5章　リスナ入力フォーマット

デバイスメッセージはコントローラとデバイス間で送受されるデータメッセージで，プログラムメッセージとレスポンスメッセージの二つがあります。この章では，リスナが受信するプログラムメッセージの書式について説明します。

プログラムメッセージは，プログラムメッセージユニットのシーケンスで構成されており，おのおののユニットは，プログラム命令またはプログラム問い合わせです。
下図は，チャンネル1に挿入したパワーメータユニットを，波長：1550 nm，測定レンジ：－10 dBmに設定するため，二つのプログラムメッセージユニットSENSE1:POWER:WAVELENGTH 1550NMとSENSE1:POWER:RANGE:UPPER －10 DBMをプログラムメッセージユニットセパレータで結び，一つのプログラムメッセージとしてコントローラからデバイスへ送出していることを示しています。



プログラムメッセージの書式は，機能を表わすことのできる最小レベルの単位まで分割した機能要素のシーケンスから構成されます。上図でカギカッコ< >で囲まれた英大文字が機能要素の例です。機能要素をさらに分割したものをコード化要素と呼びます。上図でカギカッコ< >で囲まれた英小文字がコード化要素の例です。
特定の経路の機能要素の選択を図で表わしたものを機能文法図といいます。また，特定の経路のコード化要素の選択を図で表わしたものをコード化文法図といいます。「第5.1項　リスナ入力プログラムメッセージ文法表記の要点」よりこの機能文法図・コード化文法図を使ってプログラムメッセージの書式を説明します。
コード化要素は，機能要素のデータバイトをデバイスに送るのに必要な実際のバスのコード化を表わしています。機能要素のデータバイトを受信したリスナは，おのおのの要素がコード化文法のルールに正しく従っているかどうかを解釈し，もし違反しているならば，機能要素と解釈することなくコマンドエラーを発生します。

5.1　リスナ入力プログラムメッセージ文法表記の要点

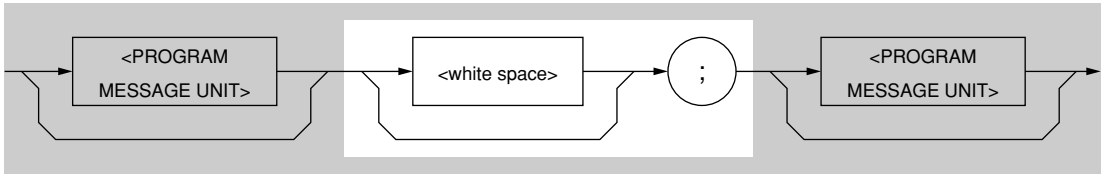
「第5.2項　プログラムメッセージの機能要素」，および「第5.3項　プログラムデータのフォーマット」の解説の要点を下記に示します。（複合コマンドと共通コマンドは省略）

5.1.1　セパレータ・ターミネータ・ヘッダ前置スペース

（1）<PROGRAM MESSAGE UNIT SEPARATOR>

複数の<PROGRAM MESSAGE UNIT>は，0個以上のスペース+セミコロンで連結されます。

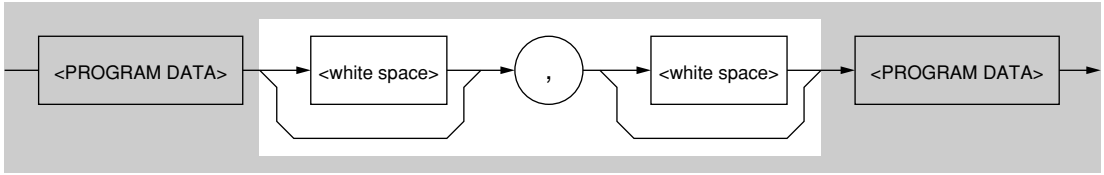
＜例＞ 二つの<PROGRAM MESSAGE UNIT>の連結一般形式



（2）<PROGRAM DATA SEPARATOR>

複数の<PROGRAM DATA>は，0個以上のスペース＋コンマ+0個以上のスペースで区切ります。

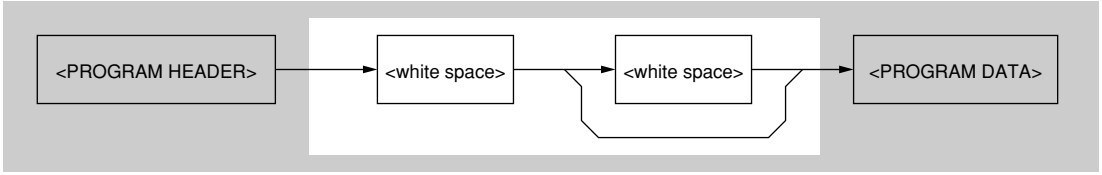
＜例＞ 二つの<PROGRAM DATA>の区切り一般形式



（3）<PROGRAM HEADER SEPARATOR>

<PROGRAM HEADER>と<PROGRAM DATA>の間を1個のスペース+0個以上のスペースで区切ります。

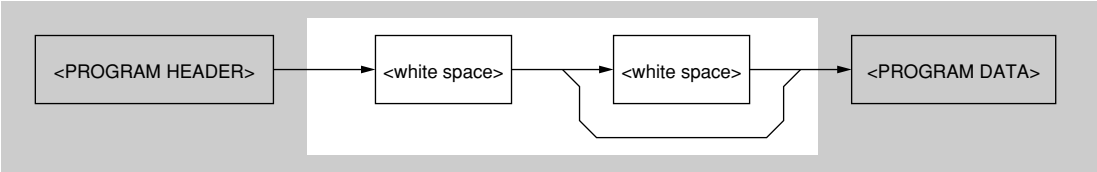
＜例＞ 単一コマンド<PROGRAM HEADER>一般形式



（4）<PROGRAM MESSAGE TERMINATOR>

<PROGRAM MESSAGE>の最後には，0個以上のスペース+{NL / EOI / NL+EOI}のどれかを付加します。

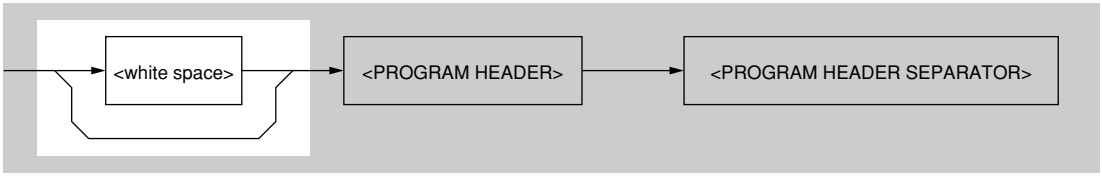
＜一般形式＞



（5）ヘッダ前置スペース

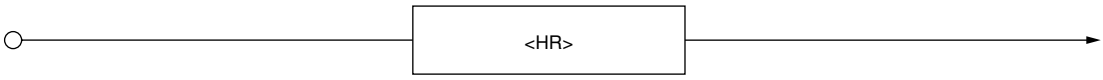
<PROGRAM HEADER>の前に，0個以上のスペースをおくことができます。

＜一般形式＞



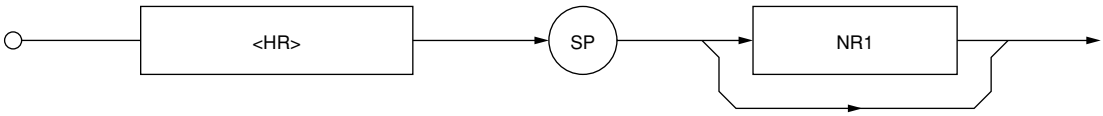
5.1.2　プログラムコマンドメッセージの一般形式

（1）データの指定をともなわないメッセージ



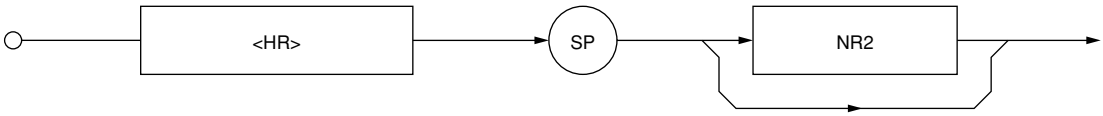
HR: COMMAND PROGRAM HEADER

（2）整数データをともなうメッセージ



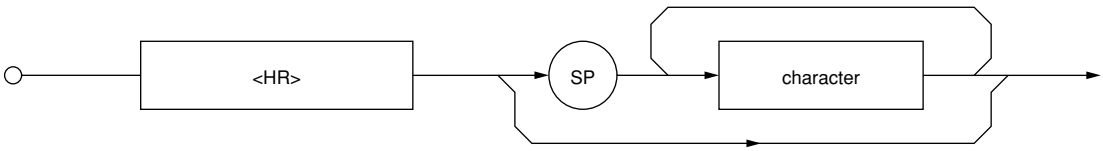
NR1: 整数

（3）実数をともなうメッセージ

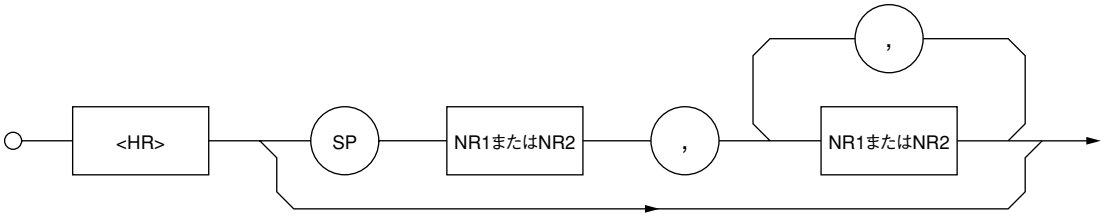


NR2: 整数

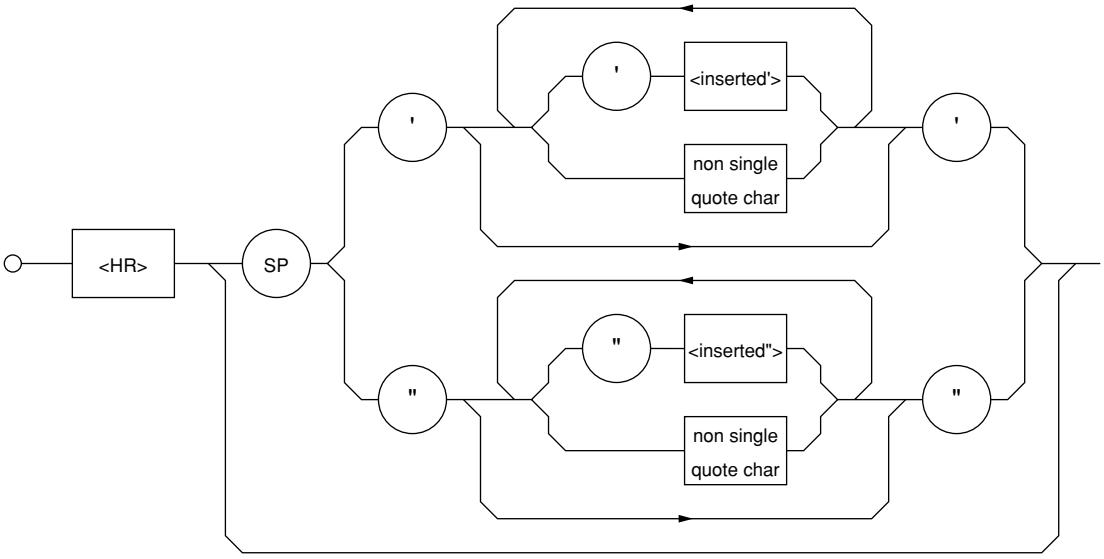
（4）固定または任意の文字列データをともなうメッセージ（データ長≦12文字）



（5）複数のプログラムデータをともなうメッセージ（先頭　NR1）



（6）ASCII 7ビットすべてが使用可能な文字列専用メッセージ

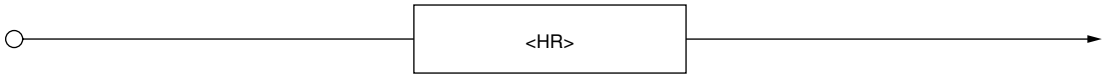


<inserted'>：　値27の単一のASCII記号を表わします。
non-sigle quote char：　値27以外のどちらかの値の単一のASCII記号を表わします。
<inserted">：　値22の単一のASCII記号を表わします。
non-sigle quote char：　値22以外のどちらかの値の単一のASCII記号を表わします。

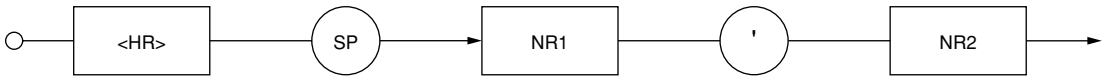
5.1.3　問い合わせメッセージの一般形式

問い合わせ<PROGRAM HEADER>は，コマンド<PROGRAM HEADER>の末尾に“?”を付けます。

（1）問い合わせデータの指定をともなわないメッセージ



（2）問い合わせデータの指定をともなうメッセージ

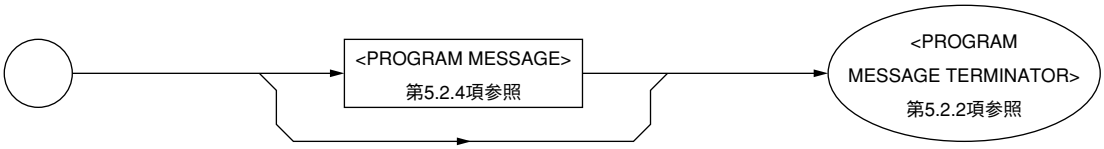


5.2 プログラムメッセージの機能要素

デバイスは，プログラムメッセージの最後にあるターミネータを検出することによりプログラムメッセージをアクセプトします。以下に，このプログラムメッセージの各機能要素を説明します。

5.2.1 <TERMINATED PROGRAM MESSAGE>

<TERMINATED PROGRAM MESSAGE>は，下図のように定義されます。

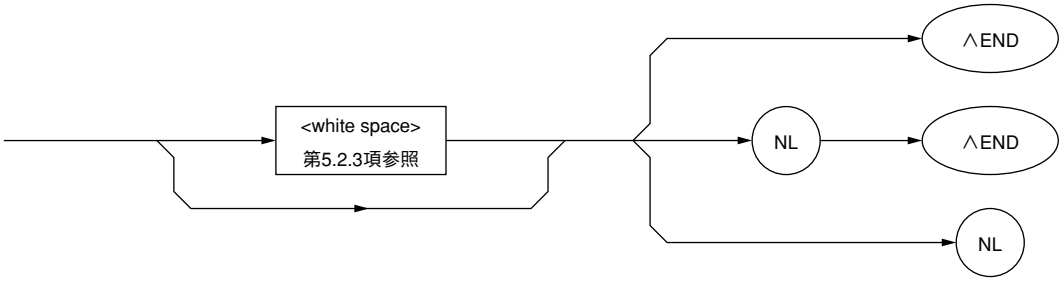


<TERMINATED PROGRAM MESSAGE>は，コントローラからリスナデバイスに送るのに必要なすべての機能要素を満たしたデータメッセージです。

<PROGRAM MESSAGE>の転送を完了させるため，<PROGRAM MESSAGE>の最後には，<PROGRAM MESSAGE TERMINATOR>が付加されます。

5.2.2 <PROGRAM MESSAGE TERMINATOR>

<PROGRAM MESSAGE TERMINATOR>は，下図のように定義されます。



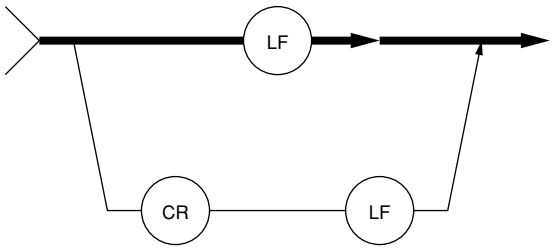
<PROGRAM MESSAGE TERMINATOR>は，一つ，またはそれ以上の一定の長さの<PROGRAM MESSAGE UNIT>要素のシーケンスを終了させます。

NL　単一のASCIIコードバイト0A（10進の10）として定義されます。すなわち，ASCII制御キャラクタLF（Line Feed）であって，印字位置を次の行へ戻す復帰改行動作を行います。これによって，新しい行からスタートするのでNL（New Line）とも呼ばれます。

END　GPIB管理バスの一つ，EOIラインをTRUE（LOWレベル）にすることにより，EOI信号を発生することができます。

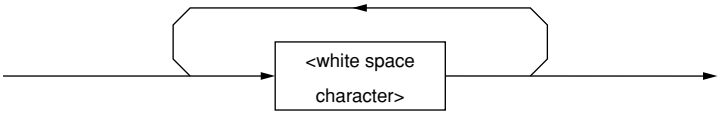
注：

CRは，印字位置を同じ行の最初の文字へ戻す復帰動作を行いますが，リスナ側では，一般には無視されます。しかし，すでに世に出回っている製品の多くは，CR-LFコードを使用している場合もあるので，たいていのコントローラの多くは，CRコードに続いてLFコードを出力する方式です。



5.2.3 <white space>

<white space>は，下図のように定義されます。

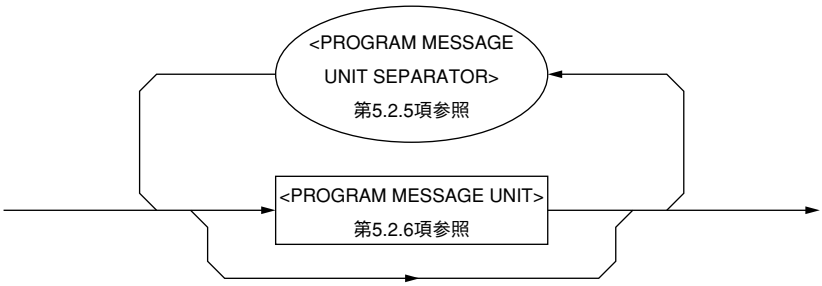


<white space character>は，ASCIIコードバイト00～09，0B～20（10進数，0～9，11～32）の範囲の中で，単一のASCIIコードバイトとして定義されます。

その範囲は，NLを除き，ASCIIコントロール記号およびスペース信号を含みますが，デバイスは，これらをASCIIコントロール記号の意味として解釈せずに，単にスペースとして処理するか，読み飛ばします。

5.2.4 <PROGRAM MESSAGE>

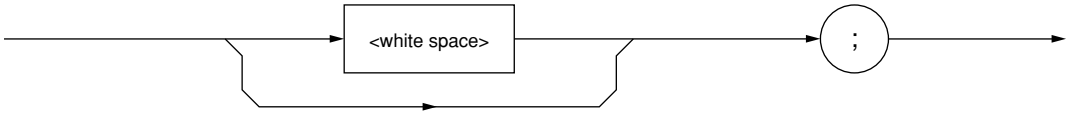
<PROGRAM MESSAGE>は，下図のように定義されます。



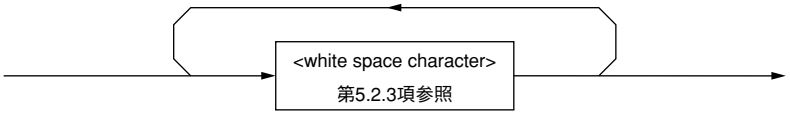
<PROGRAM MESSAGE>とは，それらは0であるか，1個の<PROGRAM MESSAGE UNIT>要素，または，より多くの<PROGRAM MESSAGE UNIT>要素のシーケンスです。<PROGRAM MESSAGE UNIT>要素は，コントローラからデバイスに送られるプログラミング命令か，データを意味しています。<PROGRAM MESSAGE UNIT SEPARATOR>要素は，複数の<PROGRAM MESSAGE UNIT>を区切るためのセパレータとして使用されます。

5.2.5　<PROGRAM MESSAGE UNIT SEPARATOR>

<PROGRAM MESSAGE UNIT SEPARATOR>は，下図のように定義されます。



<white space>は下図のように定義されます。

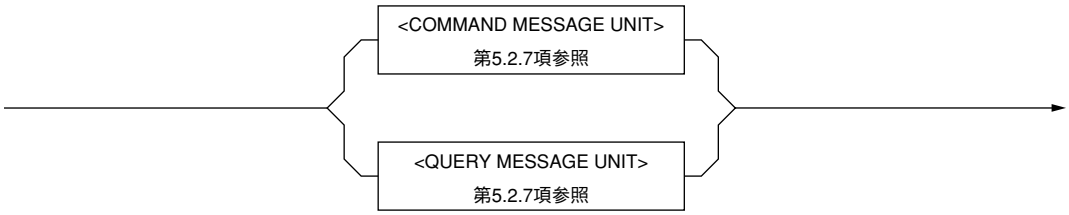


<PROGRAM MESSAGE UNIT SEPARATOR>は，<PROGRAM MESSAGE UNIT>要素のシーケンスを<PROGRAM MESSAGE>の範囲で分割します。
デバイスはセミコロン“；”を<PROGRAM MESSAGE UNIT>のセパレータとして解釈します。したがって，セミコロン“；”の前後の<white space character>は読み飛ばされます。ただし，<white space character>は，プログラムを読みやすくするためには有用です。なお，セミコロンの後に<white space>がある場合は，次の<PROGRAM HEADER>の前におかれた<white space>です。

第5.2.8項

5.2.6　<PROGRAM MESSAGE UNIT>

<PROGRAM MESSAGE UNIT>は，下図のように定義されます。

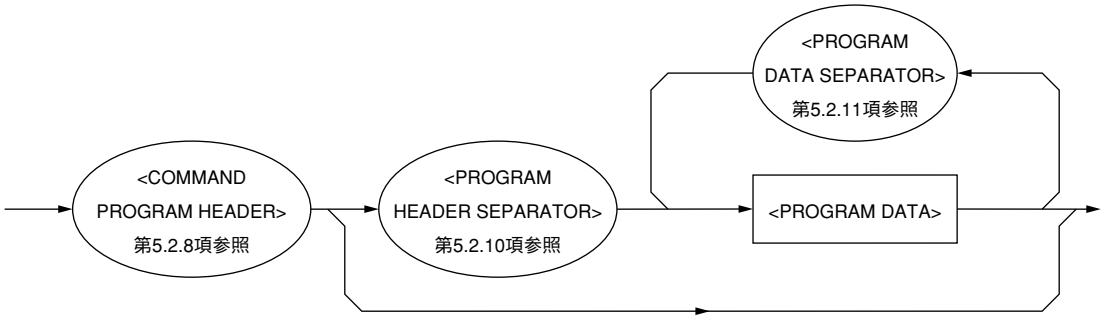


<PROGRAM MESSAGE UNIT>は，デバイスで受信される単一のコマンドメッセージである
<COMMAND MESSAGE UNIT>，または単一の問い合わせメッセージである<QUERY MESSAGE UNIT>から成ります。

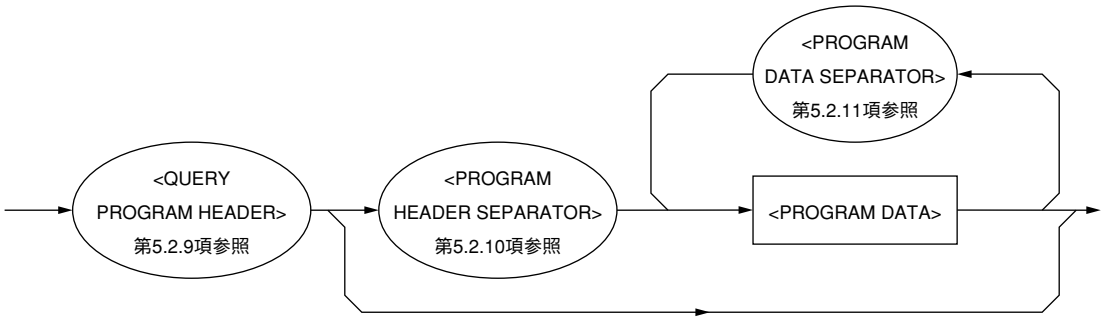
<COMMAND MESSAGE UNIT>と<QUERY MESSAGE UNIT>の詳細については，「第5.2.7項　<COMMAND MESSAGE UNIT>／<QUERY MESSAGE UNIT>」で説明します。

5.2.7 <COMMAND MESSAGE UNIT>／<QUERY MESSAGE UNIT>

<COMMAND MESSAGE UNIT>は下図のように定義されます。



<QUERY MESSAGE UNIT>は下図のように定義されます。

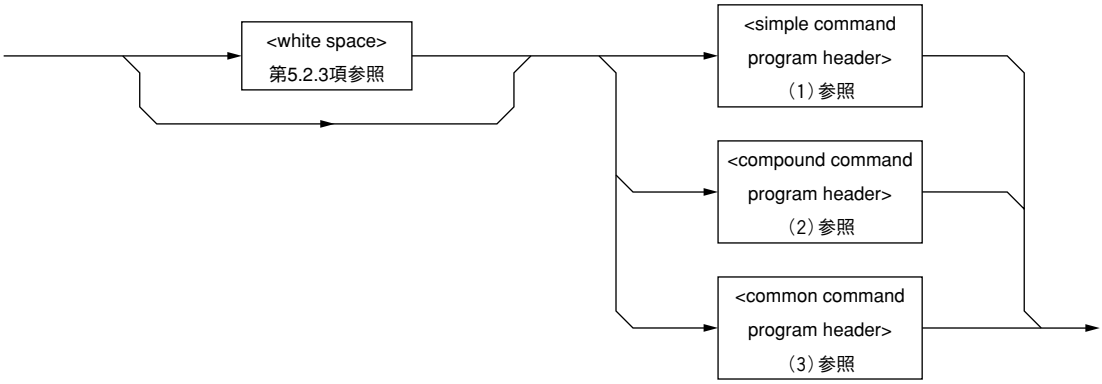


<COMMAND MESSAGE UNIT>も<QUERY MESSAGE UNIT>も，それぞれ<PROGRAM HEADER>の次に<PROGRAM DATA>が続く場合は，必ずその間にスペースが1個，セパレータとしてはいります。<PROGRAM HEADER>によって，<PROGRAM DATA>の用途・機能・動作がわかります。<PROGRAM DATA>が付かない場合は，ヘッダだけでデバイスの中で実行される用途・機能・動作を表わします。

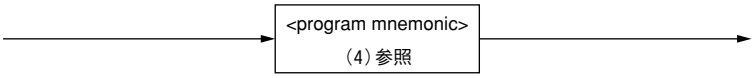
<PROGRAM HEADER>の中で，<COMMAND PROGRAM HEADER>は，コントローラからデバイスを制御するコマンドであり，<QUERY PROGRAM HEADER>は，コントローラがデバイスからレスポンスメッセージを受信するため，あらかじめコントローラからデバイスへ送る問い合わせ用コマンドです。そのヘッダの末尾には，必ず，問い合わせインジケータ“?”がつけられるのが特徴です。

5.2.8　<COMMAND PROGRAM HEADER>

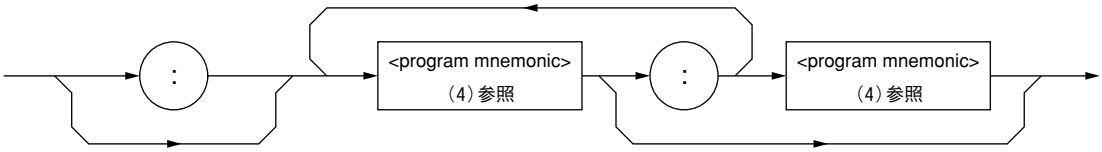
<COMMAND PROGRAM HEADER>は，下図のように定義されます。
各ヘッダの前には<white space>をおくことができます。



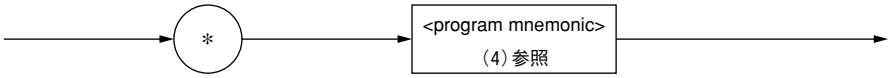
（1） <simple command program header>は下図のように定義されます。



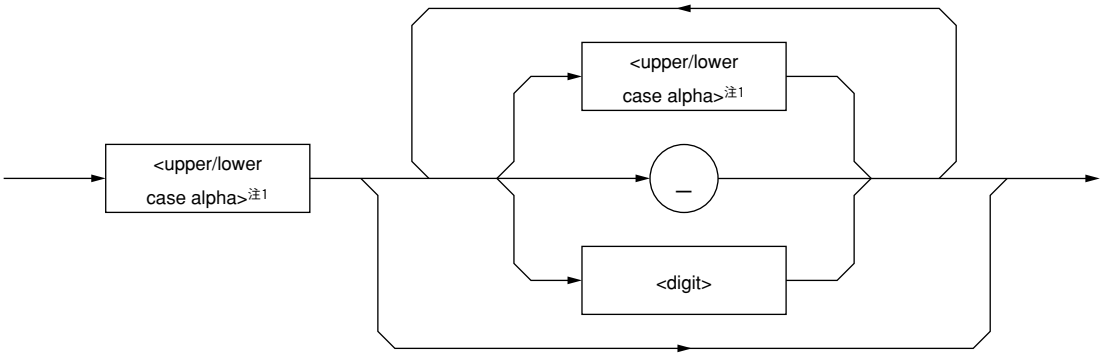
（2） <compound command program header>は下図のように定義されます。



（3） <common command program header>は下図のように定義されます。



（4） <program mnemonic>は，下図のように定義されます。



<COMMAND PROGRAM HEADER>

デバイスが実行する<PROGRAM DATA>の用途・機能・動作を表わすもので，<PROGRAM DATA>が付かない場合は，ヘッダだけでデバイスの中で実行される用途・機能・動作を表わします。
それらの意味をASCIIコード文字で表わしたのが<program mnemonic>で，一般には，ニーモニックと呼ばれています。以下に，ニーモニックの規定および上記の(1)，(2)，(3)について説明します。

<program mnemonic>

ニーモニックの先頭は，必ず英大文字または英小文字で始まります。その後は，英大文字“A～Z”／英小文字“a～z”，アンダーライン“_”，数字“0～9”の任意の組み合わせが続きます。ニーモニックの最大長は，12文字ですが，一般には3～4文字の英大文字が多用されます。文字と文字の間にスペースは含まれません。

<upper/lower case alpha>	ASCIIコードバイト41～5A，61～7A(10進数，65～90，97～122＝英大文字A～Z，英小文字a～z)の範囲の中で，単一のASCIIコードバイトとして規定されます。したがって，ヘッダは大文字で送っても，小文字で送ってもデバイスは受け付けます。
<digit>	ASCIIコードバイト30～39(10進数，48～57＝数値0～9)の範囲の中で，単一のASCIIコードバイトとして規定されます。
(_)	ASCIIコードバイト5F(10進数，95＝アンダーライン)を示し，単一のASCIIコードバイトとして規定されます。

<simple command program header>

上記の<program mnemonic>の規定がそのまま適用されます。

<compound command program header>

<compound command program header>は，複合的な機能を実行する<COMMAND PROGRAM HEADER>です。<program mnemonic>の前には，<compound command program header>のセパレータとして必ずコロン“：”が付けられます。このヘッダを1個だけ使用する場合は，後の“：”を省略することができます。

機能:

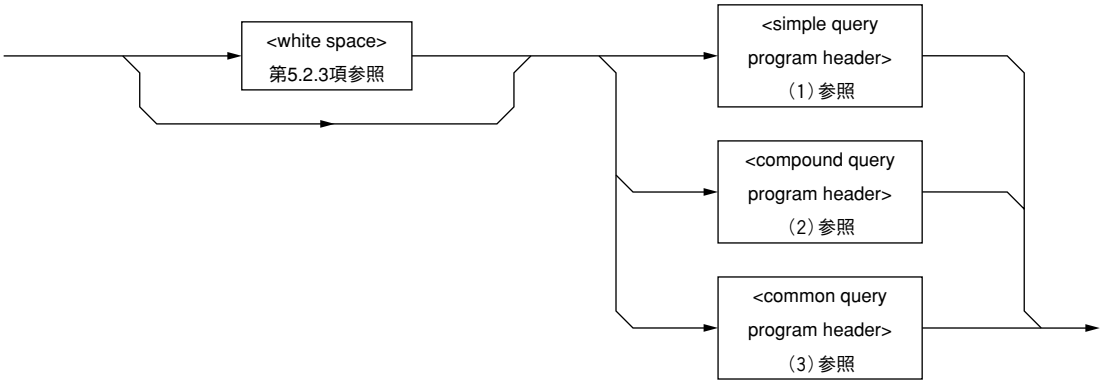
複雑なデバイスにおいて，独自のヘッダの数を制限する代わりに，複合的な機能を持たせて，デバイスコマンドセットを論理的に構成するために使用されます。階層関係にあるコマンド構造を扱うのに有用です。

<common command program header>

<common command program header>は，<program mnemonic>の前に必ずアスタリスク“*”が付けられます。このコマンドは，バス上に接続されたその他のIEEE488.2対応測定器にも共通に適用されるプログラムコマンドであるためcommonの名が付けられています。

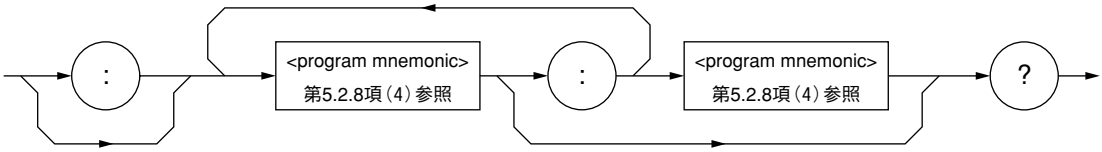
5.2.9　<QUERY PROGRAM HEADER>

<QUERY PROGRAM HEADER>は，下図のように定義されます。
各ヘッダの前には<white space>をおくことができます。

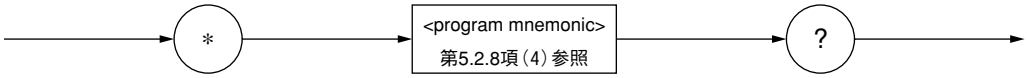


（1） <simple query program header>は下図のように定義されます。

（2） <compound cquery program header>は下図のように定義されます。



（3） <common query program header>は下図のように定義されます。



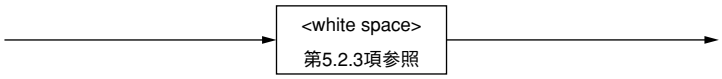
<QUERY PROGRAM HEADER>

<QUERY PROGRAM HEADER>は，コントローラがデバイスからレスポンスメッセージを受信するため，あらかじめコントローラからデバイスへ送る問い合わせ用コマンドです。そのヘッダの末尾には，必ず，問い合わせインジケータ“?”がつけられるのが特徴です。

以上で述べた<QUERY PROGRAM HEADER>の形式は，ヘッダの末尾に問い合わせインジケータ“?”が付けられる以外は，<COMMAND PROGRAM HEADER>に同じなので，「第5.2.8項　<COMMAND PROGRAM HEADER>」を参照してください。

5.2.10 <PROGRAM HEADER SEPARATOR>

<PROGRAM HEADER SEPARATOR>は，下図のように定義されます。



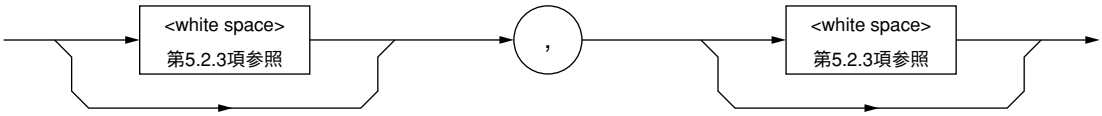
<PROGRAM HEADER SEPARATOR>は，<COMMAND PROGRAM HEADER>または
<QUERY PROGRAM HEDADER>と<PROGRAM DATA>の間のセパレータとして使用されます。

<PROGRAM HEADER>と<PROGRAM DATA>の間に複数の<white space character>がある場合は，最初の<white space character>がセパレータとして解釈され，残りの<white space character>は，読み飛ばされます。ただし，<white space character>は，プログラムを読みやすくするためには有用です。

すなわち，ヘッダセパレータは，ヘッダとデータの間に1個だけ必ず存在し，<PROGRAM HEADER>の終わりであると同時に<PROGRAM DATA>の始まりを示しています。

5.2.11 <PROGRAM DATA SEPARATOR>

<PROGRAM DATA SEPARATOR>は，下図のように定義されます。



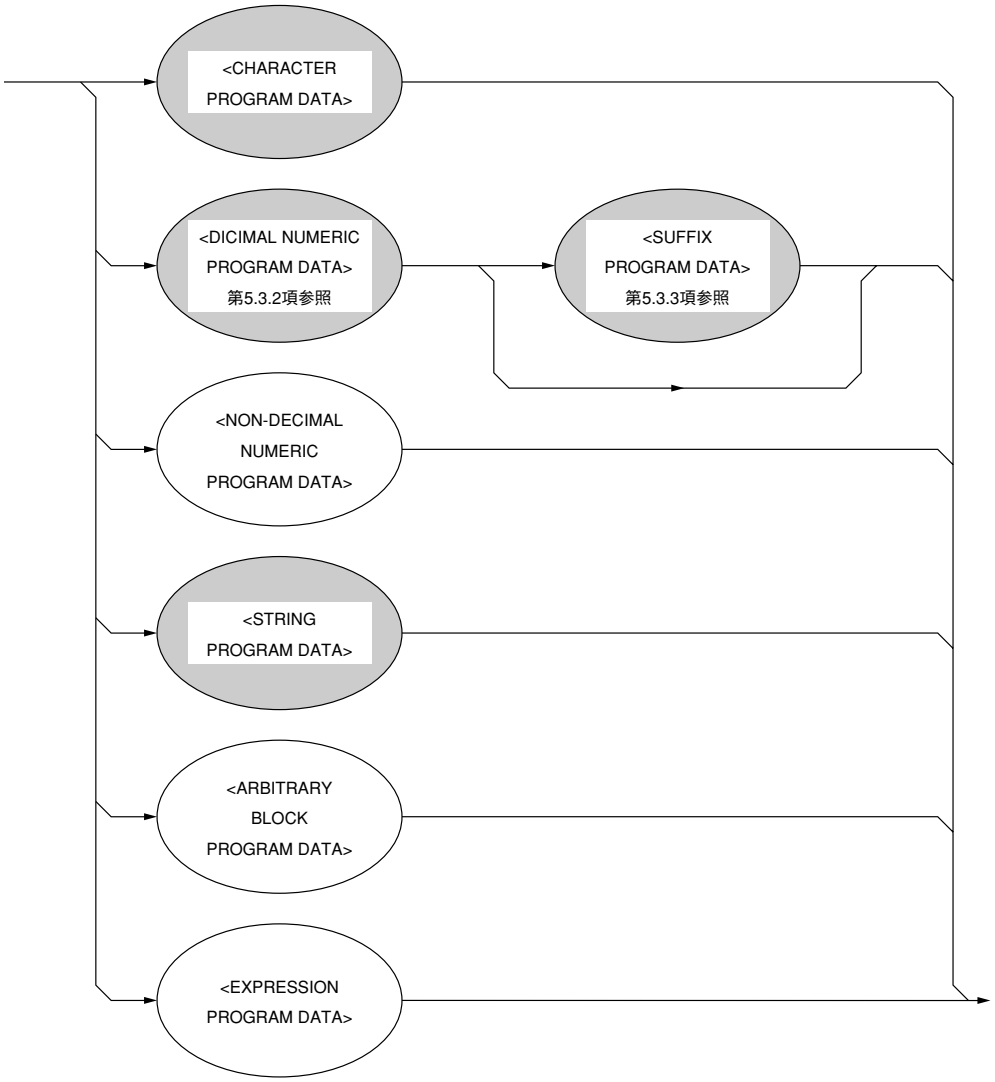
<PROGRAM DATA SEPARATOR>は，<COMMAND PROGRAM HEADER>または<QUERY PROGRAM HEADER>が多数のパラメータを持つ場合に，それらを区切るために使用されます。

この<DATA SEPARATOR>を使う場合，コンマは必ず必要ですが，<whit space character>は，必ずしも必要ではありません。コンマの前または後の<whit space character>は，読み飛ばされます。ただし，<white space character>は，プログラムを読みやすくするためには有用です。

5.3　プログラムデータのフォーマット

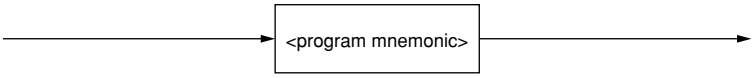
前記の，ターミネイトされたプログラムメッセージのフォーマット体系の中から，「第5.2.7項　<COMMAND MESSAGE UNIT>／<QUERY MESSAGE UNIT>」の機能文法図で示されている<PROGRAM DATA>のフォーマットを説明します。

<PROGRAM DATA>の機能要素は，<PROGRAM HEADER>に関連したいろいろなタイプのパラメータを伝送するのに使用されます。下図に，それらの<PROGRAM DATA>の種類を示します。本器は，この中からアミで囲まれ，白抜きされた<PROGRAM DATA>をアクセプトします。本器で使用されていない<PROGRAM DATA>については，参考としてお読みください。

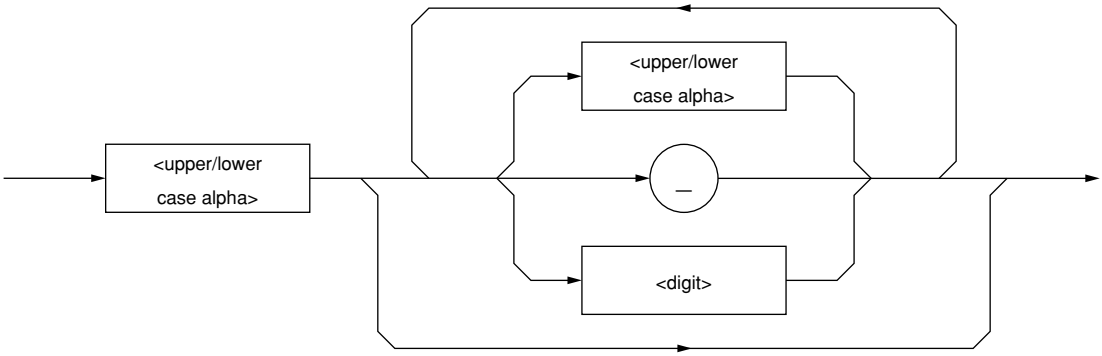


5.3.1 <CHARACTER PROGRAM DATA>

<CHARACTER PROGRAM DATA>の要素は，短い英文字または英数字データを伝達することによるリモート制御を目的とし，下図のように定義されます。



文字データの内容は<program mnemonic>と同じです。これまで，制御データといえば，数値データが主体でしたが，この文字プログラムデータによる制御も可能です。コード化文法図の詳細は下図のとおりとなります。



データの先頭は必ず英大文字または英小文字で始まります。その後，英大文字“A～Z”／英小文字“a～z”，アンダーライン“_”，数字“0～9”の任意の組み合わせが続きます。これら英数字の組み合わせは，ニーモニックと同じようなシンボルとしての使い方を目的としますので，データの最大長は，12文字です。文字と文字の間にスペースはありません。

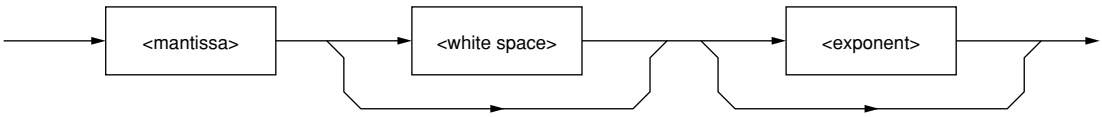
<upper/lower case alpha>	ASCIIコードバイト41～5A，61～7A（10進数，65～90，97～122＝英大文字A～Z，英小文字a～z）の範囲の中で，単一のASCIIコードバイトとして規定されます。したがって，ヘッダは大文字で送っても，小文字で送ってもデバイスは受け付けます。
<digit>	ASCIIコードバイト30～39（10進数，48～57＝数値0～9）の範囲の中で，単一のASCIIコードバイトとして規定されます。
（_）	ASCIIコードバイト5F（10進数，95＝アンダーライン）を示し，単一のASCIIコードバイトとして規定されます。

したがって，<CHARACTER PROGRAM DATA>は，比較的短いニーモニックタイプの英数字記号を送ることを目的とする<PROGRAM DATA>です。

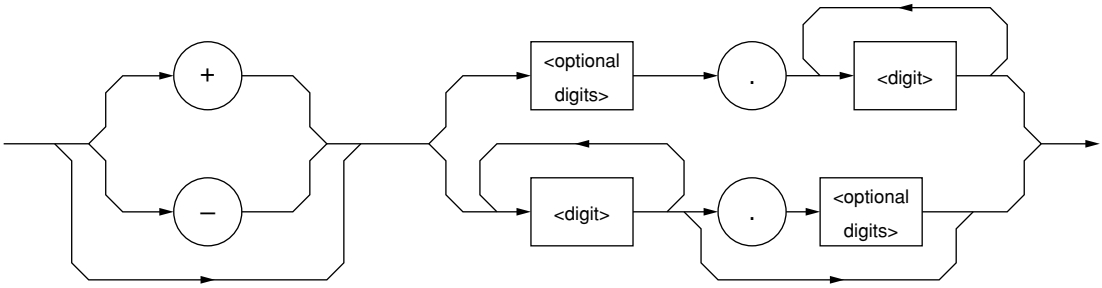
5.3.2　<DECIMAL NUMERIC PROGRAM DATA>

<DECIMAL NUMERIC PROGRAM DATA>は，10進で表現される数値定数を伝送する<PROGRAM DATA>です。10進数値の表現形式には，「整数形式」「固定小数点形式」「浮動小数点形式」の3種類があります。

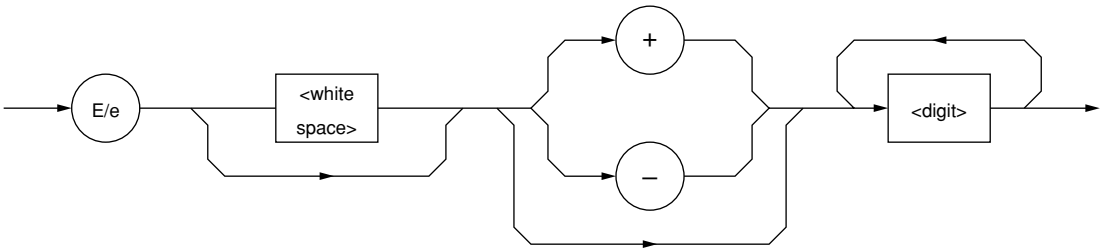
これら3種類の数値は，スペースを含むことを可とする「10進数値プログラムデータ」をフレキシブルに数値表現（NRf：flexible numeric representation）するため，下図に示すコード化文法図のように定義されます。



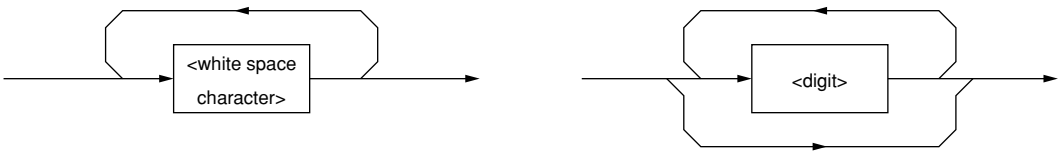
<mantissa>（仮数部）は，下図のように定義されます。



<expornent>（指数部）は，下図のように定義されます。



<white space>および<optional digits>は，下図のように定義されます。



<white space>については「第5.2.3項　<white space>」を<digit>については「第5.3.1項　<CHARACTER PROGRAM DATA>」を参照してください。

以上に説明した10進数値プログラムデータのコード化文法図を「整数形式」,「固定小数点形式」,「浮動小数点形式」に分けて，それらのプログラムデータ伝送について説明します。

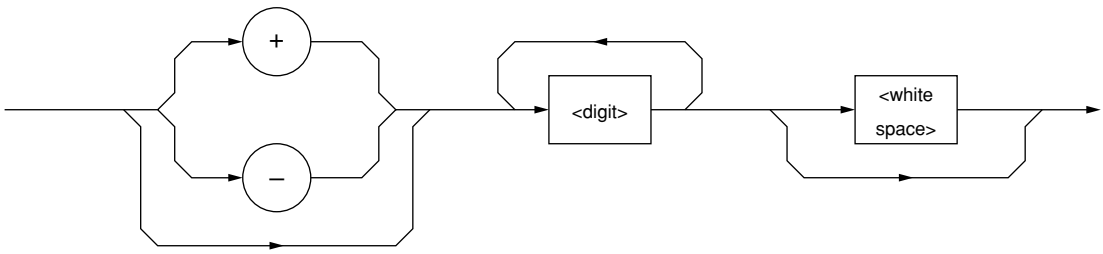
なお，どの形式の伝送においても下記の処理がなされますので注意してください。

数値要素の丸め　デバイスは，その内部で扱えるよりも桁数の多い<DECIMAL NUMERIC PROGRAM DATA>の要素を受け取った場合は，その数のサインを無視して，四捨五入を行います。

レンジ外データ　<DECIMAL NUMERIC PROGRAM DATA>要素の値が<PROGRAM HEADER>との関連において，許されているレンジ外の場合は，実行エラーが報告されます。

(1) 整数形式ーNR1伝送

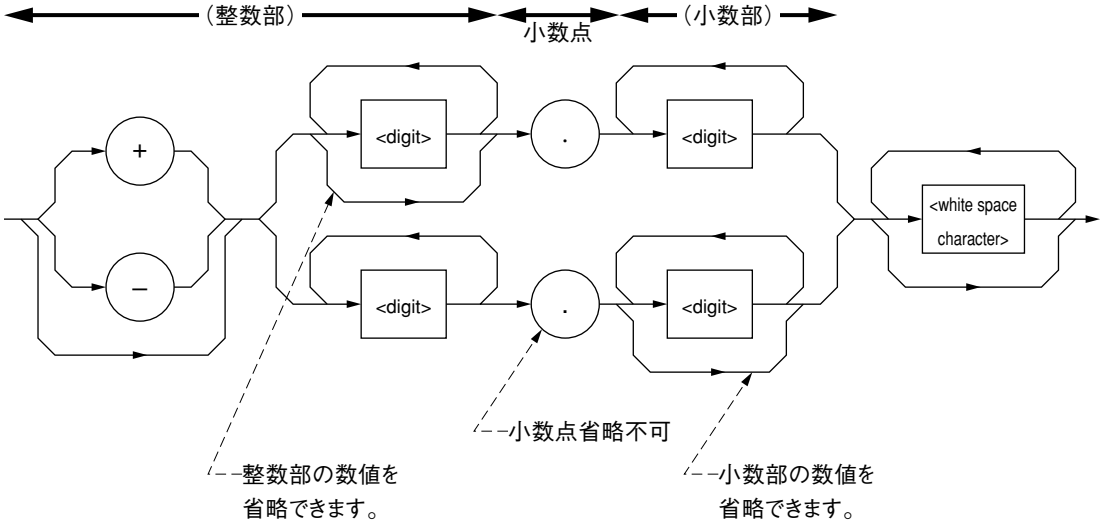
小数点や指数表現を含まない10進数値，すなわち実数の中の整数(NR1)を伝送します。



- 先頭に0挿入可　→ 005，+000045
- 符号(+または−)と数字の間にスペース挿入不可　→ +5，+△5(×)
- 数字の後ろにスペース挿入可　→ ＋5△△△
- +符号は，付けても付けなくてもかまいません。　→ +5，5
- 桁区切りにコンマは使用できません。　→ 1,234,567(×)

(2) 固定小数点形式－NR2伝送

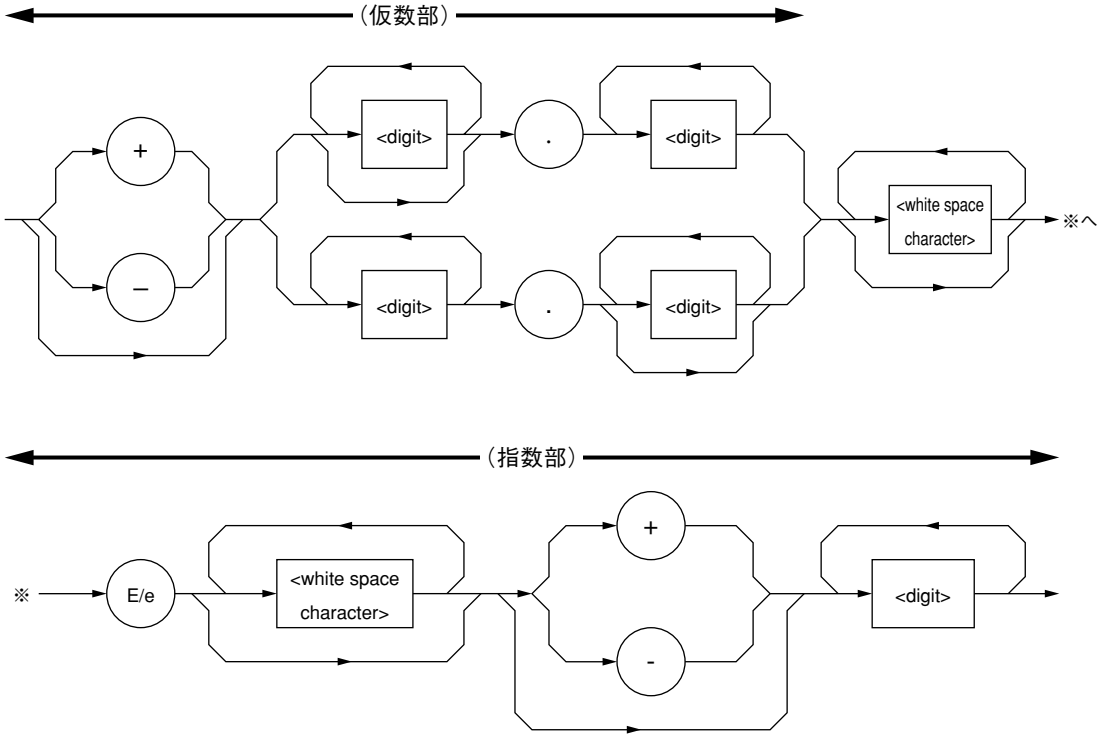
小数点以下の桁を持つ10進数値，すなわち整数および指数表現を除く実数(NR2)を伝送します。文法図は，(整数部)，小数点，(小数部)から成ります。



- (整数部)は整数形式の数値表現が適用されます。
- 数字と小数点の間にスペース挿入はできません。　→ +753△.123(×)
- (小数部)の数字の後ろにスペース挿入できます。　→ +753.123 △△△△
- 小数点の前に数値がなくてもかまいません。　→ .05
- 小数点の前に符号がおけます。　→ +.05，－.05
- 小数点で終わることもできます。　→ 12.

（3） 浮動小数点形式－NR3伝送

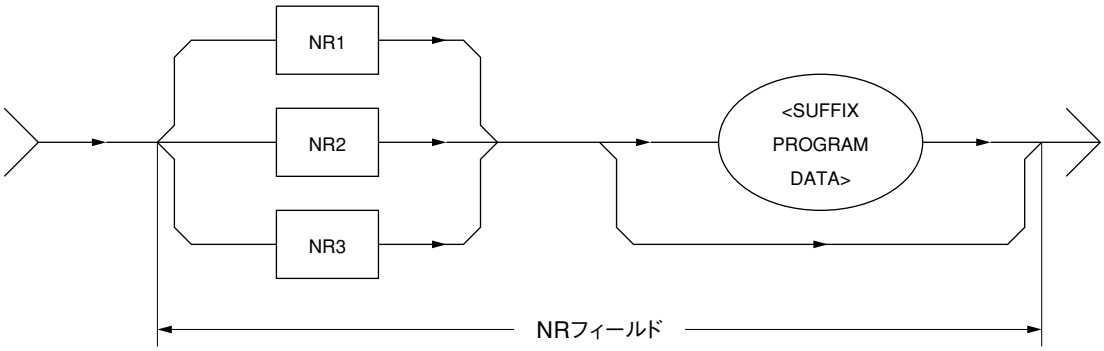
指数表現の桁を持つ10進数値，すなわち浮動小数点形式によって表わされた実数（NR3）を伝送します。文法図は，（仮数部）と（指数部）から成ります。仮数部は数値の精度を表わすため，整数形式または固定小数点形式で表現されます。指数部はEから始まり，その右側に10のべき乗の数値がおかれます。



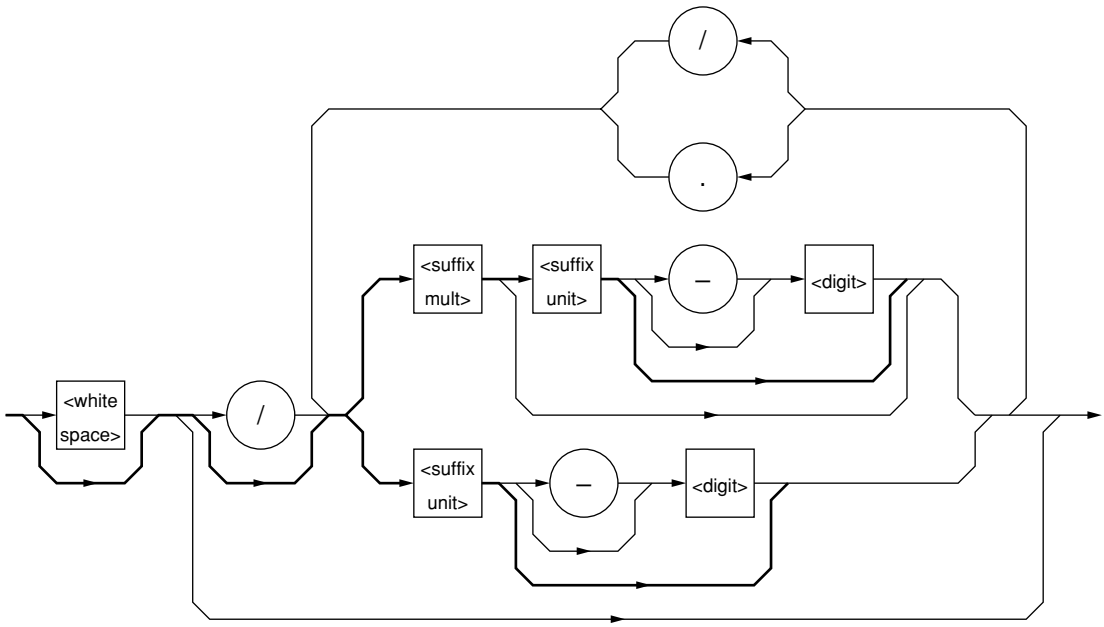
- Eは10のべき乗を意味し，指数部（Exponent part）の始まりを示します。
- Eは大文字・小文字どちらでもかまいません。　→　1.234E+12，1.234e+12
- E/eの前もしくは後ろにスペースがおけます。　→　1.234 △E△+12
- 符号が+なら仮数部も指数部も，＋を省略できます。　→　+1.234E+4，1.234E4
- 仮数部で数字を省略できません。　→　－1E2，－E2（×），－.E2（×）

5.3.3 <SUFFIX PROGRAM DATA>

<SUFFIX PROGRAM DATA>は，「第5.3.2項　<DECIMAL NUMERIC PROGRAM DATA>」で説明した<DECIMAL NUMERIC PROGRAM DATA>（整数形式NR1，固定小数点形式NR2，または浮動小数点形式NR3）に続いて用いられるもので，各形式の末尾にはサフィクスをおくこともできます。



サフィクスは，10進数値プログラムデータで測定単位が必要な場合のみ，そのデータの末尾につけて使用するもので，サフィクス単位（suffix unit）またはサフィクス乗数（suffix multipliers）とサフィクス単位を組み合わせて用います。文法図は，下図に示すとおりですが，一般には太線のルートがよく使用されます。



- サフィクス乗数は，英大文字または英小文字で表現します。
  たとえば，1E3 Hzは1E3＝kとして1 kHzで表わします。

- サフィクス単位は，英大文字または英小文字で表現します。
- <SUFFIX PROGRAM DATA>の先頭にEを置くことは，浮動小数点形式で使用されるEと混同される恐れがあるので，禁止されています。

表5-1に，サフィクス乗数と単位を示します。

(1) サフィクス乗数

表5-1　サフィクス乗数

乗数	ニーモニック	名前
1E18	EX	EXA
1E15	PE	PETA
1E12	T	TERA
1E9	G	GIGA
1E6	MA (NOTE)	MEGA
1E3	K	KILO
1E-3	M (NOTE)	MILLI
1E-6	U	MICRO
1E-9	N	NANO
1E-12	P	PICO
1E-15	F	FEMTO
1E-18	A	ATTO

注：

従来からの慣習により，Hzの10^6をMHz(megahertz)，OHMの10^6をMOHM(megohm)としています。これらは，表5-1のサフィクス乗数表に入れないで，表5-2のサフィクス単位表にリストされています。

(2) 相対的単位(dB)

1 μVに関するデシベル　　DBUV
1 μWに関するデシベル　　DBUW
1 mWに関するデシベル　　DBMW

慣習により，DBMWの代わりにDBMの使用が認められています。

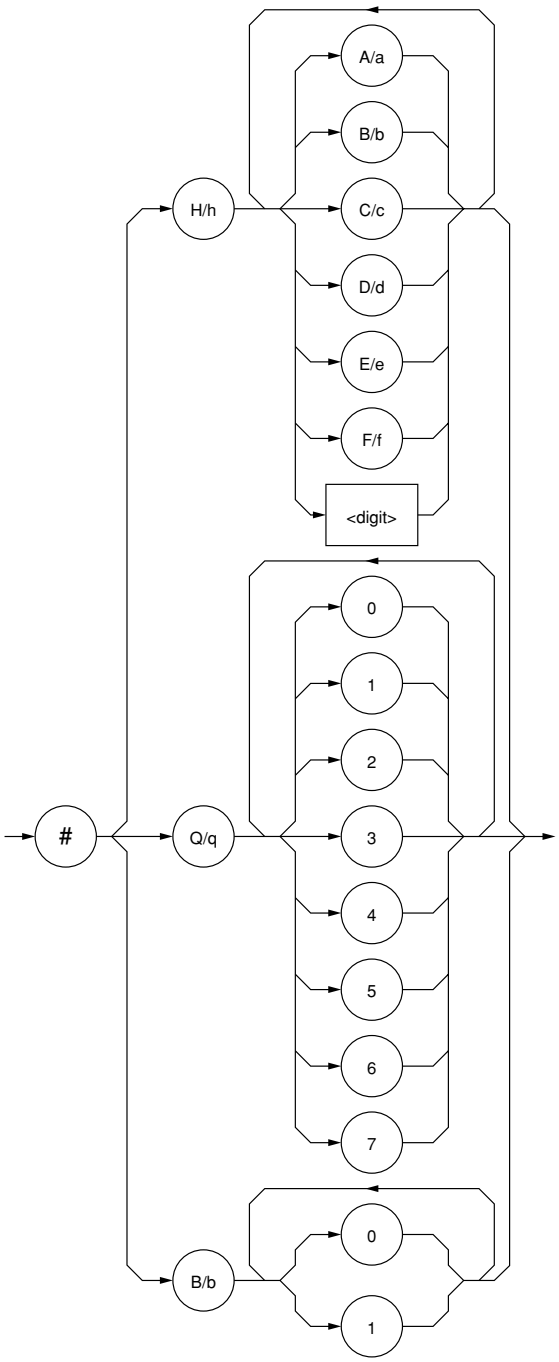
（3） サフィクス単位

表5-2　サフィクス単位

項目	推奨単位 ニーモニック	準推奨単位 ニーモニック	名前
電流	A		Ampere
気圧	ATM		Atmosphere
電荷	C		Coulomb
照度	CD		Candela
デシベル	DB		Decibel
電力	DBM		Decibel milliwatt
キャパシタンス	F		Farad
質量		G	Gram
インダクタンス	H		Henry
周波数（ヘルツ）	HZ		Hertz
水銀柱	INHG		Inches of mercury
ジュール	J		Joule
温度	K	 CEL FAR	Degree Kelvin Degree Celsius Degree Fahrenheit
容積	L		Liter
照度	LM		Lumen
照度	LX		Lux
長さ（メートル）	M	 FT IN	Meter Feet Inch
周波数（1E3 Hz）		MHZ	Megahertz
抵抗		MOHM	Megaohm
力	N		Newton
抵抗	OHM		Ohm
圧力	PAL		Pascal
比（パーセント）	PCT		Percent
角度（ラジアル）	RAD		Radian
角度（度）		DEG MNT	Degree Minute (of arc)
時間（秒）	S	SEC	Second
コンダクタンス	SIE		Siemens
自動速度	T		Tesla
圧力	TORR		Torr
電圧	V		Volt
電力（ワット）	W		Watt
時速	WB		Weber
照度	LM		Lumen

5.3.4 <NON-DECIMAL NUMERIC PROGRAM DATA>

<NON-DECIMAL NUMERIC PROGRAM DATA>は，非10進数値として，16進・8進・2進数値データを伝送する<PROGRAM DATA>です。非10進データは，必ず#マークから始まります。下図の左側に示すコード化文法図のように定義されます。
指定した文字列以外の並びが送られるとコマンドエラーとなります。



#Hまたは#hに続く文字は,符号付きでない16進数としてデバイスで受け付けられます。
（ ）内は,対応する10進数を示します。

#Habc1230	(11,256,099D)
#hAbC123	
#H2DC3	(11,715D)
#h2dc3	
#H8301	(33,537D)
#h8301	

#Qまたは#qに続く文字は,符号付きでない8進数としてデバイスで受け付けられます。

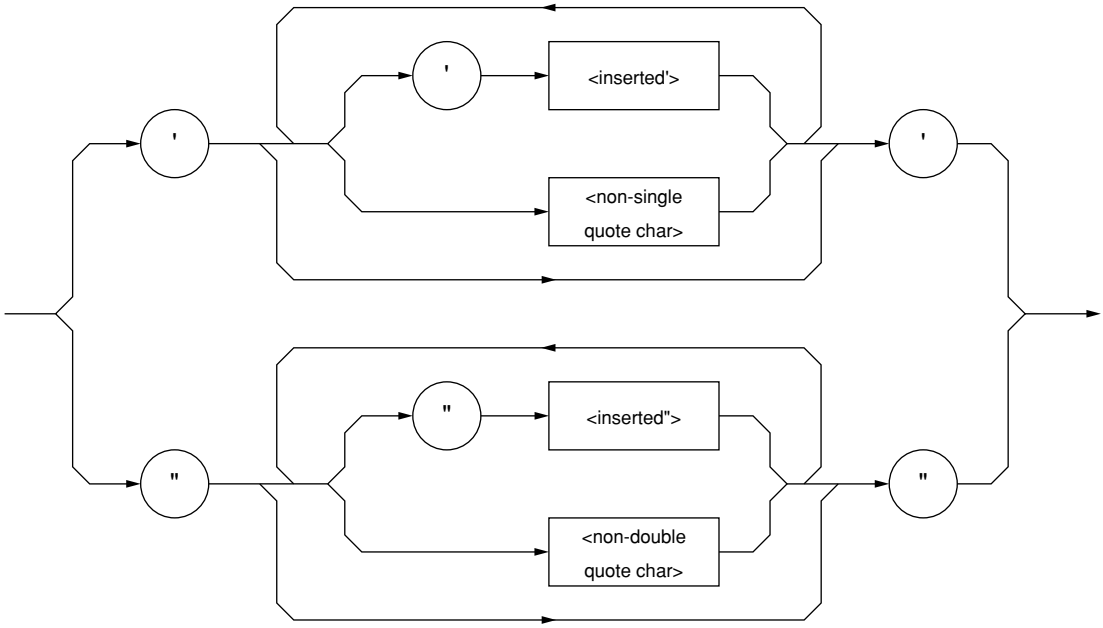
#Q37	(31D)
#q37	
#Q26703	(11,715D)
#q26703	

#Bまたは#bに続く文字は,符号付きでない2進数としてデバイスで受け付けられます。

#B10101011110000010010011	(11,256,099D)
#b0010110111000011	(11,715D)

5.3.5 <STRING PROGRAM DATA>

<STRING PROGRAM DATA>は，文字列専用の<PROGRAM DATA>です。ASCII 7 ビットコードのすべてを使用することができます。ただし，文字列中にシングル引用符“'”またはダブル引用符“"”がある場合は，どちらかの引用符1個につき，同じ引用符を2個続けて記述しなければなりません。



（1） 文字列の両端は，文字列中の引用符の有無にかかわらず，必ずシングル引用符またはダブル引用符で囲みます。たとえば，下記のように記述します。

It's a nice day.　→ "It's a nice day."
→ 'It' 's a nice day.'

（2） 文字列の両端をシングル引用符で囲んだ場合，文字列中のシングル引用符を2連とします。その他の文字は，ダブル引用符を含みそのまま記述します。たとえば，下記のように記述します。

"I shouted,'Shame'."　→ ' "I shouted,' 'Shame' '." '

（3） 文字列の両端をダブル引用符で囲んだ場合，文字列中のダブル引用符を2連とします。その他の文字は，シングル引用符を含みそのまま記述します。たとえば，下記のように記述します。

"I shouted,'Shame'."　→ " " "I shouted,'Shame'." " "

（4） <inserted'>はASCIIコードバイト27（10進の39＝記号'）の単一のASCII記号，また<inserted">はASCIIコードバイト22（10進の34＝記号"）の単一のASCII記号です。

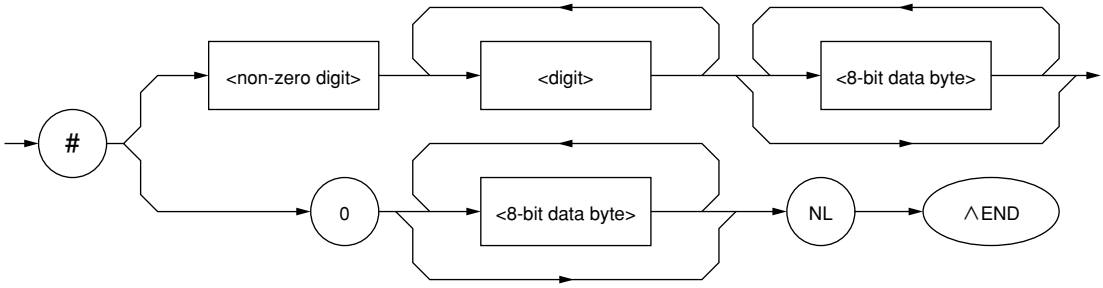
<non-single quote char>および<non-double quote char>は，それぞれシングル引用符，ダブル引用符以外の単一のASCII記号です。

5.3.6 <ARBITRARY BLOCK PROGRAM DATA>

<ARBITRARY BLOCK PROGRAM DATA>は，#マークから始まる非10進プログラムデータです。データ形式を変更することなく，1バイト=8ビットを最小のブロックとして，バイナリデータを直接に伝送します。「第5.3.4項　<NON-DECIMAL NUMERIC PROGRAM DATA>」の非10進数値プログラムデータ<NON-DECIMAL NUMERIC PROGRAM DATA>との違いは，以下のとおりです。

- データは，数値データに限定せず，文字列データでも数値データでも扱います。
- ＃と先頭データの間に，送出するデータバイト数などを記述します。

このように，この非10進データは，転送すべきデータバイトを任意に指定できるプログラムデータで，下図のように定義されます。



<digit>	ASCIIコードバイト30～39（10進数，48～57＝数値0～9）の範囲の中で，単一のASCIIコードバイトとして規定されます。
<non-zero digit>	ASCIIコードバイト31～39（10進数，49～57＝数値1～9）の範囲の中で，単一のASCIIコードバイトとして規定されます。
<8-bit data byte>	00～FF（10進数の0～255）の範囲の8ビットバイト

（1） 送出すべきデータバイト数が既知の場合

上の文法図で，右上のルートが適用されます。転送する<8-bit data byte>のバイト数は，図の<digit>の位置，すなわち，データを書き始める直前で指定します。そして，指定したバイト数の桁数を＃と<digit>の間，すなわち<non-zero digit>の位置に書きます。たとえば，4バイトのデータバイト（DAB）を送るには，下記のように記述します。

4バイト送るので<digit>の位置に4と指定します。

↓

＃14<DAB><DAB><DAB><DAB>

↑

右の<digit>の位置にある4は，1桁なので<non-zero digit>の値は1です。

4バイト送るので<digit>の位置に4と指定します。先行0付きでも指定できます。

↓

＃3004<DAB><DAB><DAB><DAB>

↑

右の<digit>の位置にある4は，3桁なので<non-zero digit>の値は3です。

（2）送出すべきデータバイト数が未知の場合

上記の文法図で，右下のルートが適用されます。最初のデータの前に#0を置きます。また，最後のデータの次にNL＾ENDを置きますので出口なしでターミネートされます。

＃0<DAB><DAB><DAB><DAB><DAB>NL＾END

（3）整数精度バイナリデータの扱い

整数精度バイナリデータは，プログラムデータの場合もレスポンスデータの場合も，<ARBITRARY BLOCK>形式の転送データとして用いられ，表5-3のような仕様となっています。負の数は2の補数形として処理されます。

表5-3

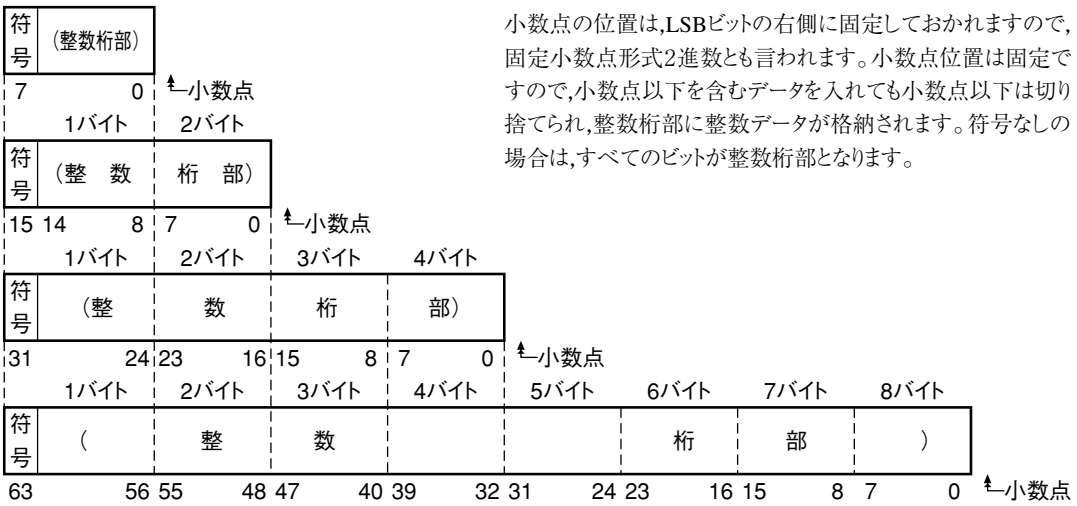
転送バイト数	1,2,4または8バイト
転送順序	最上位から順に転送する
符号付きバイナリーコード	LSD……右端から右づめ MSB……サインビットとする データ長がフィールドより小さいときは残ったフィールドをMSBで埋める
符号なしバイナリーコード	LSD……右端から右づめ MSB……サインビットではない 使用しない上位ビットは0で埋める

下記に，よく使用される1バイト8ビットおよび2バイト16ビット整数データの符号付きと符号なしの範囲を示します。

8-Bit Binary	With Sign	No Sign
10000000	–128	128
10000001	–172	129
10000010	–126	130
11111101	–3	253
11111110	–2	254
11111111	–1	255
00000000	0	0
00000001	1	1
00000010	2	2
00000011	3	3
01111101	125	125
01111110	126	126
01111111	127	127

16-Bit Binary	With Sign	No Sign
1000000000000000	–32768	32768
1000000000000001	–32767	32769
1000000000000010	–32766	32770
1111111111111101	–3	65533
1111111111111110	–2	65534
1111111111111111	–1	65535
0000000000000000	0	0
0000000000000001	1	1
0000000000000010	2	2
0000000000000011	3	3
0111111111111101	32765	32765
0111111111111110	32766	37266
0111111111111111	32767	32767

下図に符号付き1，2，3，4，8バイト整数データの内部表現を示します。符号ビットは0のときに正のデータを，1のときに負のデータを表わします。



小数点の位置は,LSBビットの右側に固定しておかれますので,固定小数点形式2進数とも言われます。小数点位置は固定ですので,小数点以下を含むデータを入れても小数点以下は切り捨てられ,整数桁部に整数データが格納されます。符号なしの場合は,すべてのビットが整数桁部となります。

(4) 浮動小数点形式バイナリデータの扱い

浮動小数点形式バイナリデータは，<PROGRAM DATA>の場合も<RESPONSE DATA>の場合も，<ARBITRARY BLOCK>形式の転送データとして用いられます。なお，当社製造のデバイスには浮動小数点形式バイナリデータは採用されていませんが，一般的な仕様を下記に示します。

この形式の数値は，下記の三つのフィールドで構成されなければなりません。

(a)　サインフィールド　（サインビット）
(b)　指数フィールド　（指数ビット）
(c)　仮数フィールド　（仮数ビット）

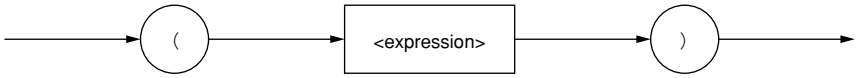
ここで扱う数値は，小数を含む数値データであって，その数値精度として単精度と倍精度があります。表5-4にそのフィールド構成と転送順位を示します。ここに，S(サインビット)，EM(最上位の指数ビット)，EL(最下位の指数ビット)，FM(最上位の仮数ビット)，FL(最下位の仮数ビット)とします。

表5-4

<table>
<tr><th>精度</th><th>転送バイト数</th><th>フィールド構造と転送順位</th></tr>
<tr><td>単
精
度</td><td>4バイト</td><td>
<table>
<tr><th rowspan="2">転送バイト</th><th colspan="8">DIOライン</th></tr>
<tr><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th></tr>
<tr><td>1バイト目</td><td>S</td><td>EM</td><td>E</td><td>E</td><td>E</td><td>E</td><td>E</td><td>E</td></tr>
<tr><td>2バイト目</td><td>EL</td><td>FM</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td></tr>
<tr><td>3バイト目</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td></tr>
<tr><td>4バイト目</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>FL</td></tr>
</table>
サインビット:　1ビット
指数ビット:　8ビット(+127～−126)
仮数ビット:　23ビット
</td></tr>
<tr><td>倍
精
度</td><td>8バイト</td><td>
<table>
<tr><th rowspan="2">転送バイト</th><th colspan="8">DIOライン</th></tr>
<tr><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th></tr>
<tr><td>1バイト目</td><td>S</td><td>EM</td><td>E</td><td>E</td><td>E</td><td>E</td><td>E</td><td>E</td></tr>
<tr><td>2バイト目</td><td>E</td><td>E</td><td>E</td><td>EL</td><td>FM</td><td>F</td><td>F</td><td>F</td></tr>
<tr><td>3～7バイト目</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td></tr>
<tr><td>8バイト目</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>FL</td></tr>
</table>
サインビット:　1ビット
指数ビット:　11ビット(+1023～−1022)
仮数ビット:　52ビット
</td></tr>
</table>

5.3.7 <EXPRESSION PROGRAM DATA>

<EXPRESSION PROGRAM DATA>要素は，スカラー，ベクトル，行列，またはストリング値を求める式をデバイスに送り，デバイスがコントローラに代わって，それらの式を計算し，値を求めようとするものです。コード化文法図では，下図のように定義されます。



<expression>　<expression>は，ASCIIコードバイト20～7E(10進数表現で32～126)の範囲のASCII文字をシーケンスに使用します。ただし，下記の6文字は除外されます。

" （ダブル引用符）
（番号記号）
' （シングル引用符）
(（左括弧）
) （右括弧）
; （セミコロン）

<expression>を，たとえば，a+b+c とおけば，上の文法図は下記のようにおけます。

(a+b+c)

これをデバイスへ転送する手段としては，<ARBITARARY BLOCK PROGRAM DATA>の不定フォーマット以外ならば，これまで説明してきたP4-16からP4-35までの<PROGRAM DATA>を使用することができます。(<expression>)を受け取ったデバイスは，この式の計算を行い，その数値を求める処理を行います。

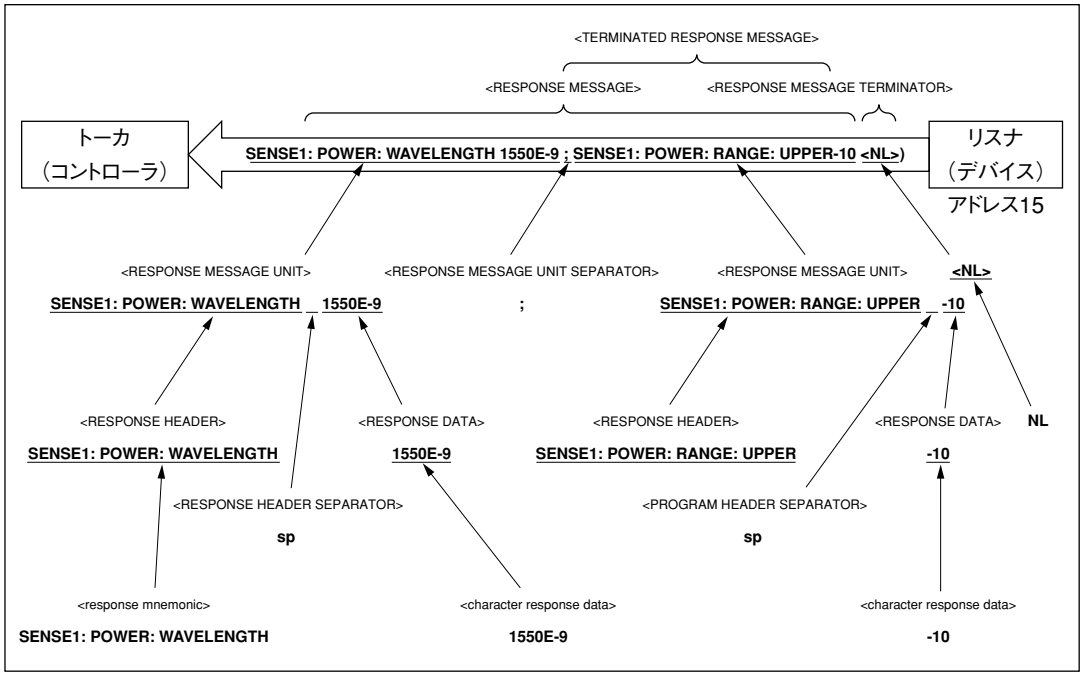
注：

MT9810Bでは，この<expression>の機能はありません。式の計算が必要ならば，あらかじめコントローラで式の値を求め，その結果，算出された数値データを<PROGRAM DATA>としてデバイスへ転送します。

第6章　トーカ出力フォーマット

デバイスメッセージは，コントローラとデバイス間で送受されるデータメッセージで，プログラムメッセージとレスポンスメッセージの二つがありますが，この章では，トーカデバイスからコントローラへ送出するレスポンスメッセージの書式について説明します。

レスポンスメッセージの代表的なものとして，測定結果・設定状態・ステータス情報などがあり，ヘッダ付きで返されるレスポンスメッセージとヘッダなしで返されるレスポンスメッセージがあります。
下図は，チャンネル1に挿入したパワーメータユニットに対して，設定波長問い合わせと測定レンジ問い合わせのメッセージユニットを送出した場合の，それぞれのレスポンスメッセージがヘッダ付きASCII文字列でデバイスからコントローラへ送出していることを示しています。



以上の動作部分のみをプログラムで示すと，下記のようになります。
Call Send (0,15,"SENSE1:POWER:WAVELENGTH?;SENSE1:POWER:RANGE:UPPER?",NLend)[注1]
Call Receive (0,15,buf1,NLend)[注2]

注1：

設定波長と測定レンジの問い合わせメッセージユニットを送出します。

注2：

ターミネータNLが検出されると，レスポンスメッセージSENSE1:POWER:WAVELENGTH 1550E-9;SENSE1:POWER:RANGE:UPPER-10がbuf1に読み込まれます。

レスポンスメッセージの書式は，プログラムメッセージの場合と同様に，機能を表わすことのできる最小レベルの単位まで分割した機能要素のシーケンスから構成されます。上図でカギカッコ<　>で囲まれた英大文字が機能要素の例です。機能要素をさらに分割したものをコード化要素と呼びます。上図でカギカッコ<　>で囲まれた英小文字がコード化要素の例です。したがって，文法図表記法は，トーカもリスナも同じです。

「第6.1項　リスナ入力とトーカ出力フォーマットの文法上の相違点」より，リスナ装置入力フォーマットとの相違点を中心にトーカ装置の出力フォーマットを説明します。

6.1　リスナ入力とトーカ出力フォーマットの文法上の相違点

リスナ装置の入力フォーマットとトーカ装置の出力フォーマットに関する文法上の最も大きな相違点は以下のとおりです。

リスナフォーマット　コントローラからのプログラムメッセージをデバイスが容易にアクセプトできるようにするため，柔軟性を持ったプログラム作成が意図されています。したがって，プログラムメッセージに記述上の多少の違いがあっても，それらのプログラムメッセージは，同じ機能を発揮することができます。たとえば，セパレータやターミネータには，<white space>を好きなだけジョイントできるため，読みやすいプログラムを作成することができます。

トーカフォーマット　デバイスから出力されるレスポンスメッセージをコントローラが容易にアクセプトできるようにするため，厳格に定められた文法に従って出力メッセージは送り出されます。したがって，上記とは逆にレスポンスメッセージの文法は，一つの機能に対しては一つの表記法しかありません。

リスナ装置の入力フォーマットとトーカ装置の出力フォーマットの相違点を要約し，表6-1に示します。なお，表中の0／1個以上のスペースは<white space>を指します。

表6-1

項目	リスナ入力プログラムメッセージ文法	トーカ出力レスポンスメッセージ文法
特性	（柔軟）	（厳格）
英文字	大文字も小文字も同じ意味に使えます。	大文字のみ
NR3指数部Eの前後	0個以上のスペース+E/e+0個以上のスペース	大文字Eのみ
NR3指数部の+符号	省略可能	省略不可
<white space>	セパレータ前後やターミネータの前に複数おけます。	不使用
メッセージユニット	(a) プログラムデータ付きヘッダ (b) プログラムデータなしヘッダ	(a) ヘッダ付きデータ (b) ヘッダなしデータ
ユニットセパレータ	0個以上のスペース+セミコロン	セミコロンのみ
ヘッダ前置スペース	0個以上のスペース+ヘッダ	ヘッダのみ
ヘッダセパレータ	ヘッダ+1個以上のスペース	ヘッダ+1個の$20[注1]
データセパレータ	0個以上のスペース+コンマ+0個以上のスペース	コンマのみ
ターミネータ	0個以上のスペース+{NL, EOI, NL+EOI}のどれか	NL+EOI

注1：

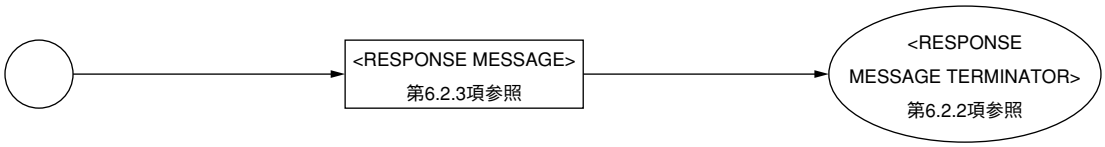
ASCIIコードバイト20（10進数32＝ASCII文字SP，スペース）

6.2　レスポンスメッセージの機能要素

トーカから出力されるレスポンスメッセージは，NL^END信号でターミネイトされることにより，コントローラでアクセプトされます。以下に，このレスポンスメッセージの各機能要素について説明します。
文法図の表記規定については，プログラムメッセージの場合と同じなので，「第5章　リスナ入力フォーマット」を参照してください。また，機能要素やコード化要素の説明についても，プログラムメッセージの場合と重複するものについては，説明を省略していますので必要があれば，「第5章　リスナ入力フォーマット」を参照してください。

6.2.1　<TERMINATED RESPONSE MESSAGE>

<TERMINATED RESPONSE MESSAGE>は，下図のように定義されます。

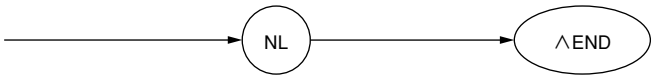


<TERMINATED RESPONSE MESSAGE>は，トーカデバイスからコントローラに送るのに必要なすべての機能要素を満たしたデータメッセージです。

<RESPONSE MESSAGE>の転送を完了させるため，<RESPONSE MESSAGE>の最後には，<RESPONSE MESSAGE TERMINATOR>が付加されます。

6.2.2　<RESPONSE MESSAGE TERMINATOR>

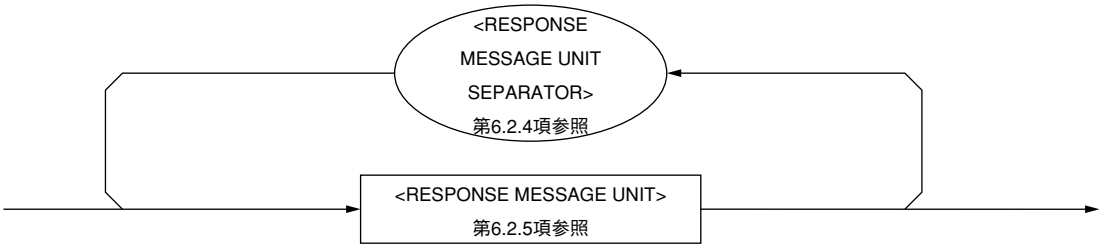
<RESPONSE MESSAGE TERMINATOR>は，下図のように定義されます。



<RESPONSE MESSAGE TERMINATOR>は，最後の<RESPONSE MESSAGE UNIT>の次に置かれ，一つ，またはそれ以上の一定の長さの<RESPONSE MESSAGE UNIT>要素のシーケンスを終了させます。

6.2.3 <RESPONSE MESSAGE>

<RESPONSE MESSAGE>は，下図のように定義されます。

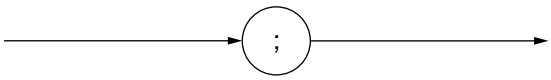


<RESPONSE MESSAGE>とは，1個の<RESPONSE MESSAGE UNIT>要素，またはより多くの<RESPONSE MESSAGE UNIT>要素のシーケンスです。

<RESPONSE MESSAGE UNIT>要素は，デバイスからコントローラに送られる単一のメッセージを意味しています。<RESPONSE MESSAGE UNIT SEPARATOR>要素は，複数<RESPONSE MESSAGE UNIT>を区切るためのセパレータとして使用されます。

6.2.4 <RESPONSE MESSAGE UNIT SEPARATOR>

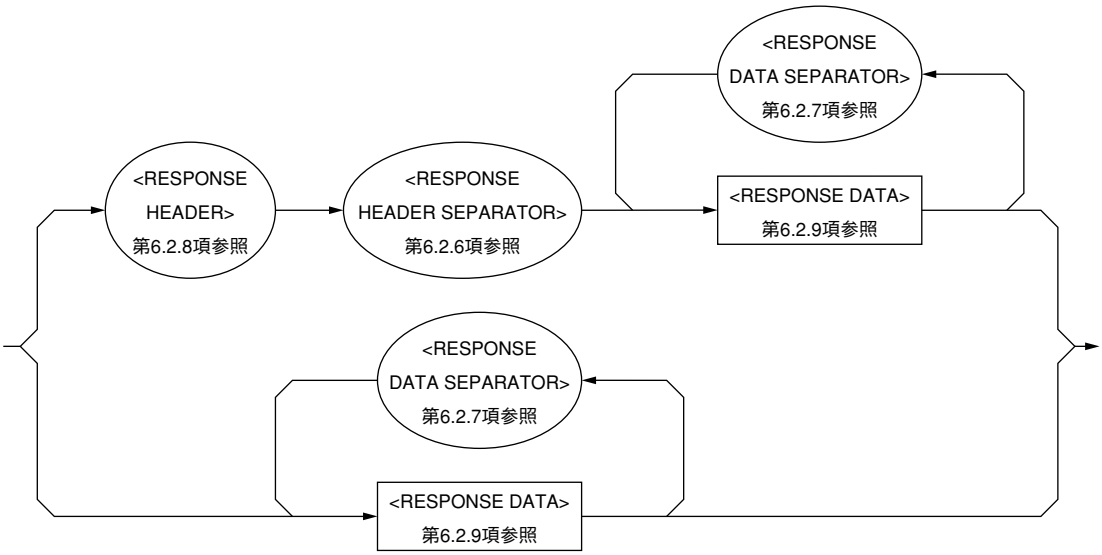
<RESPONSE MESSAGE UNIT SEPARATOR>は，下記のように定義されます。



<RESPONSE MESSAGE UNIT SEPARATOR>は，複数の<RESPONSE MESSAGE UNIT>要素のシーケンスを一つの<RESPONSE MESSAGE>として，出力する場合に，<RESPONSE MESSAGE UNIT>要素を<UNIT SEPARATOR>セミコロン“；”で分割します。

6.2.5 <RESPONSE MESSAGE UNIT>

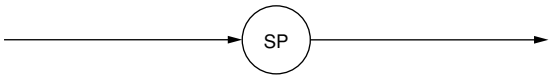
<RESPONSE MESSAGE UNIT>は，下図のように定義されます。



<RESPONSE MESSAGE UNIT>は二つの基本的なシンタックスから成っています。
前者はヘッダ付き<RESPONSE MESSAGE UNIT>で，プログラムメッセージでセットした情報の処理結果を正確に返します。後者はヘッダなし<RESPONSE MESSAGE UNIT>で，測定結果のデータだけを無駄なく返します。

6.2.6 <RESPONSE HEADER SEPARATOR>

<RESPONSE HEADER SEPARATOR>は，下図のように定義されます。



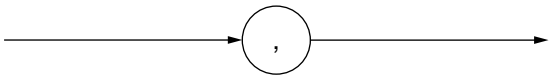
<RESPONSE HEADER SEPARATOR>は，<RESPONSE HEADER>の次に1個のスペースを置き，<RESPONSE HEADER>と<RESPONSE DATA>を分離します。

スペースSPは，ASCIIコードバイト20(10進数の32)です。

すなわち，ヘッダ付き<RESPONSE MESSAGE>では，<RESPONSE HEADER SEPARATER>としてスペースがヘッダとデータの間に1個だけ必ず存在し，<RESPONSE HEADER>の終わりであると同時に<RESPONSE DATA>の始まりを示しています。

6.2.7　<RESPONSE DATA SEPARATOR>

<RESPONSE DATA SEPARATOR>は，下図のように定義されます。



<RESPONSE DATA SEPARATOR>は，複数の<RESPONSE DATA>を出力する場合に，データとデータの間に置き，それらを区切るために使用されます。

6.2.8　<RESPONSE HEADER>

<RESPONSE HEADER>は，フォーマット表現上においては，次の3点を除き，「第5.2.8項　<COMMAND PROGRAM HEADER>」の説明と同じです。

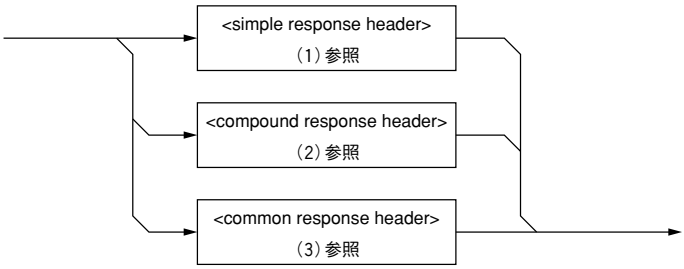
（1）<response mnemonic>で使用文字が定められており，その中で英文字については，大文字のみを使用してください。その他は<program mnemonic>に同じです。

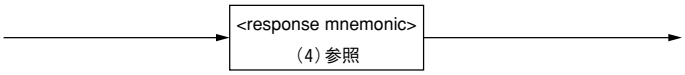
（2）<PROGRAM HEADER>の前にスペースがおけましたが，<RESPONSE HEADER>前には，スペースはおけません。

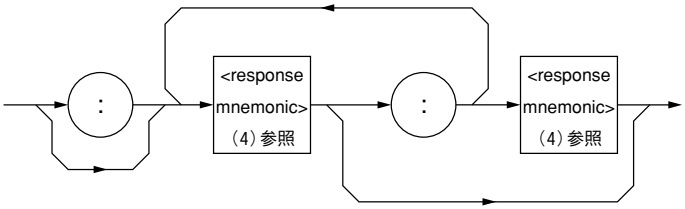
（3）<PROGRAM HEADER>の後には，複数のスペースがおけましたが，<RESPONSE HEADER>後には，1個のスペースしかおけません。

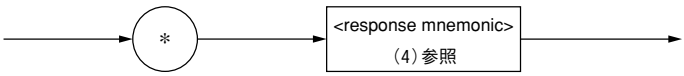
表6-2に，<response mnemonic>までを一括して示します。
（<response mnemonic>で使用される文字の中で，英字は大文字のみとなりますが，その他は「第5.2.8項　<COMMAND PROGRAM HEADER>」の説明と同じです。）

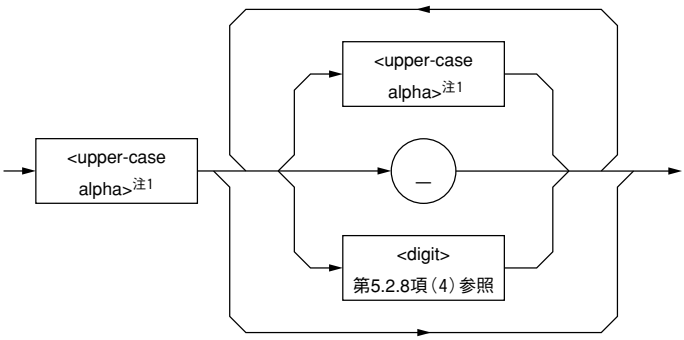
表6-2

<table>
<tr><th>要素</th><th>機能</th></tr>
<tr><td>RESPONSE HEADER</td><td>ヘッダは,<RESPONSE DATA>の機能を表わすもので,英大文字から始まる12文字以内の英大文字,数字,アンダーラインのうちどれか,それらの組み合わせから成る<response mnemonic>で,その語義が示されます。
 

(1) <simple response header>は,下図のように定義されます。
 

(2) <compound response header>は,下図のように定義されます。
 

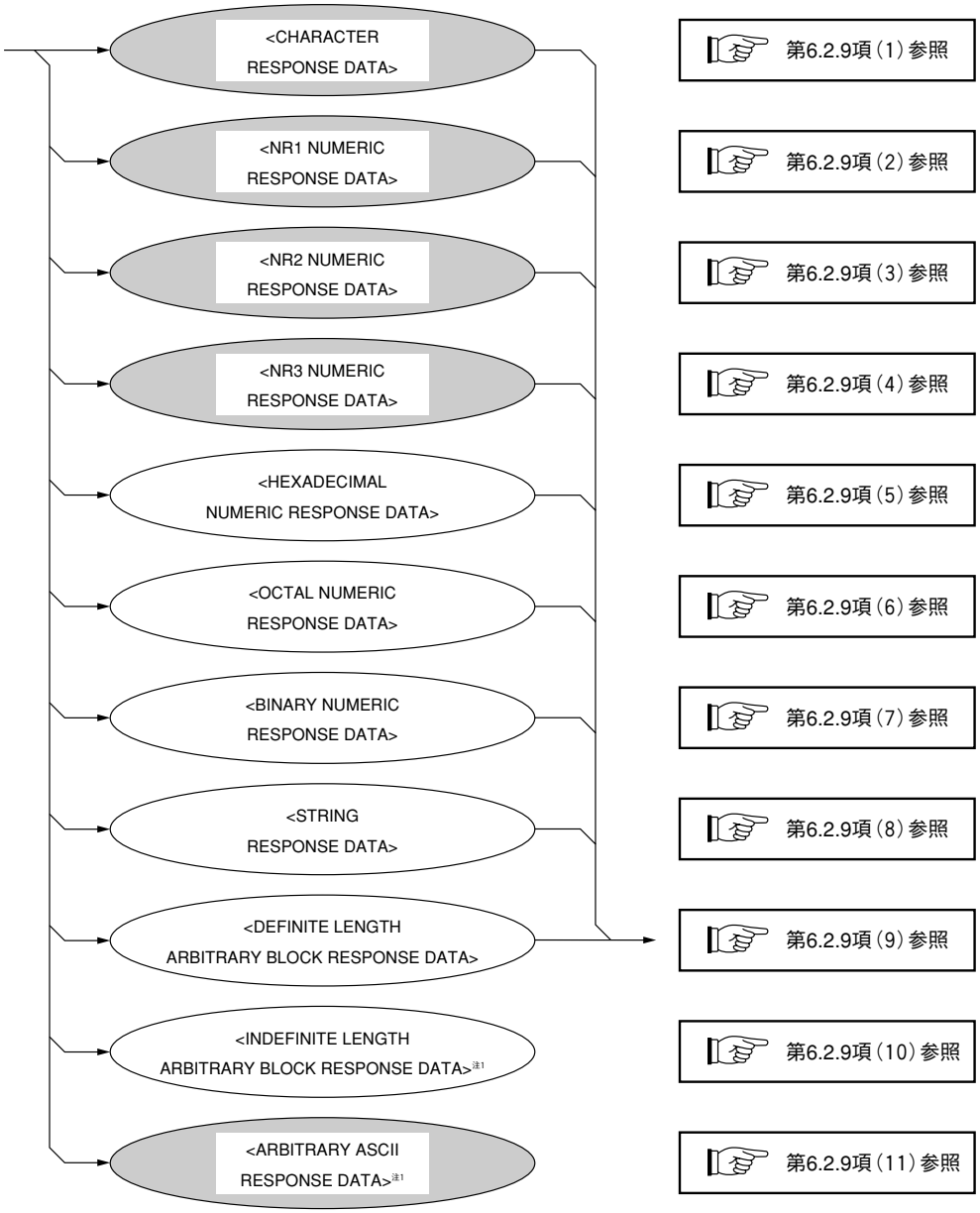
(3) <common response header>は,下図のように定義されます。
 

(4) <response mnemonic>は,下図のように定義されます。
 

注1:
<upper-case alpha> ASCIIコードバイト41～5A(10進数65～90=英大文字A～Z)</td></tr>
</table>

6.2.9　<RESPONSE DATA>

<RESPONSE DATA>は11種類あり，その中で本器は，アミで囲まれている<RESPONSE DATA>をコントローラへ送出します。どの<RESPONSE DATA>が返されるかは，問い合わせメッセージによって決定されます。



注1：

<INDEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA>および<ARBITRARY ASCII RESPONSE DATA>は，それ自身の最後のデータバイトの次はNL^ENDでターミネイトされます。

表6-3

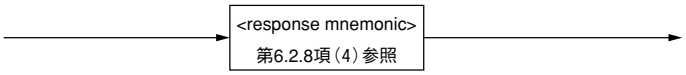
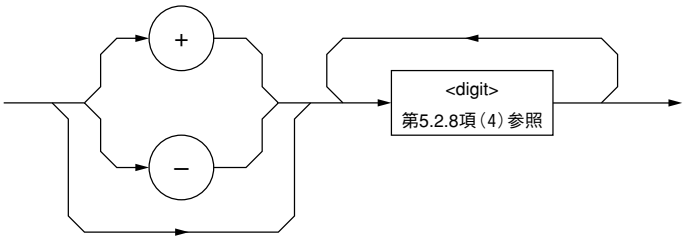
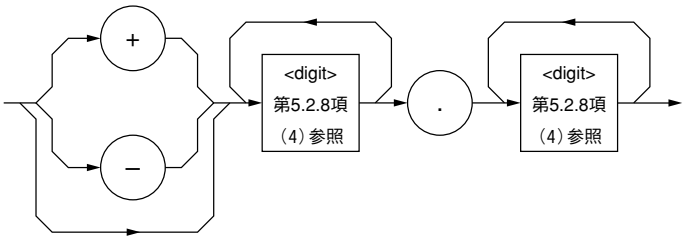
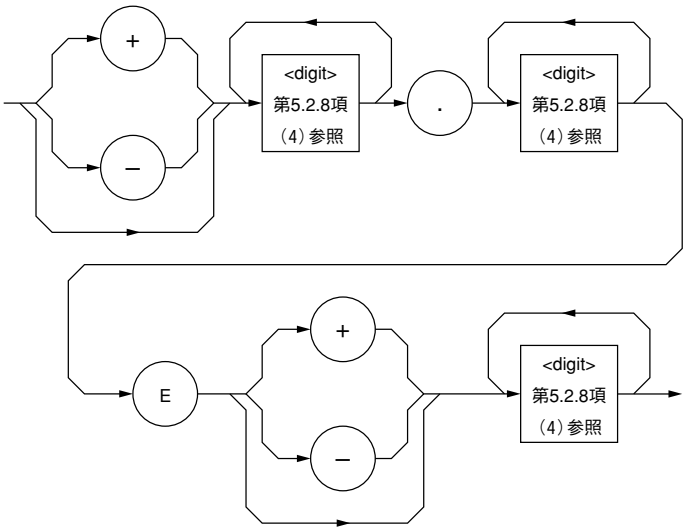
要素	機能
(1) CHARACTER RESPONSE DATA	<response mnemonic>と同じ文字列の構成から成るデータです。したがって,文字列の先頭は必ず英大文字から始まり,文字列の長さは12文字以内です。数値パラメータの使用は適当でありません。 
(2) NR1 NUMERIC RESPONSE DATA ＜例＞ 123 ＋123 －1234	整数形式データ,すなわち小数点や指数表現を含まない整数の10進数値 
(3) NR2 NUMERIC RESPONSE DATA ＜例＞ 12.3 ＋12.34 －12.345	固定小数点形式データ,すなわち整数および指数表現を除く10進数値 
(4) NR3 NUMERIC RESPONSE DATA ＜例＞ 1.23E＋4 ＋12.34E－5 －12.345E＋6 • Eに小文字は使えません。 • Eの前後のスペースは不可 • 指数部の＋は省略できません。 • 仮数部の＋は省略可	浮動小数点形式データ,すなわち指数表現の桁を持つ10進数値 

表6-3（つづき）

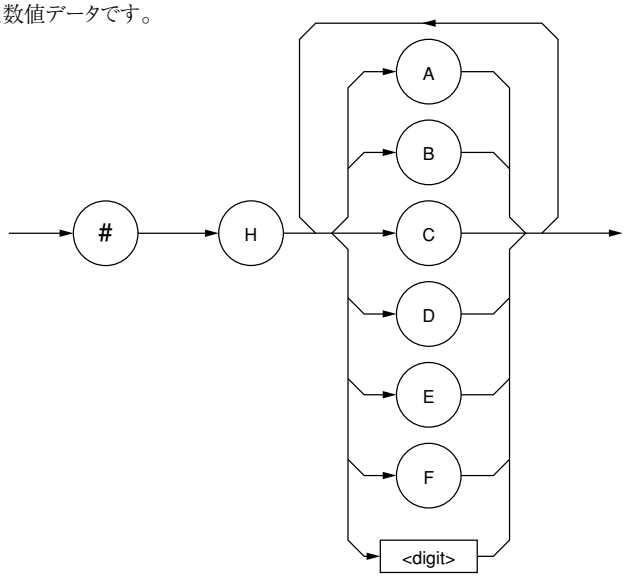
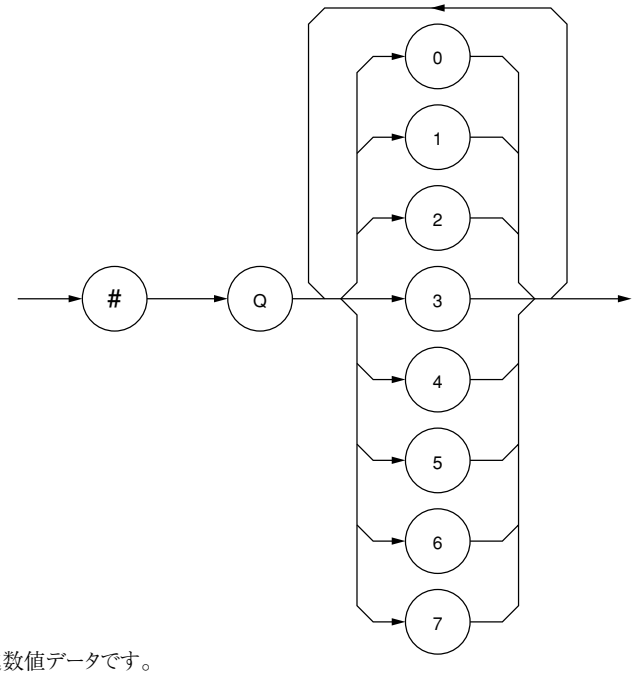
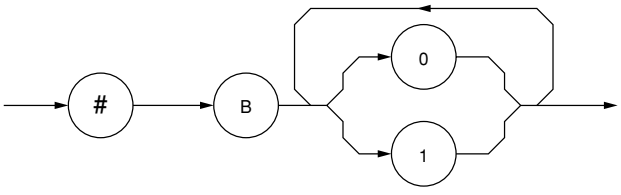
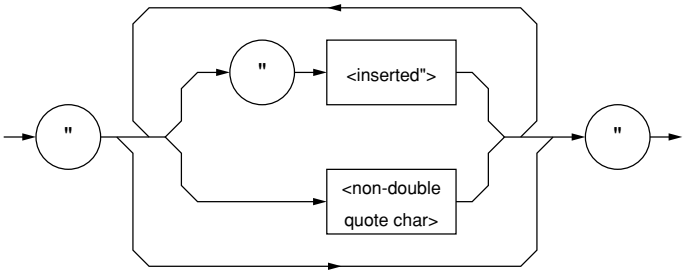
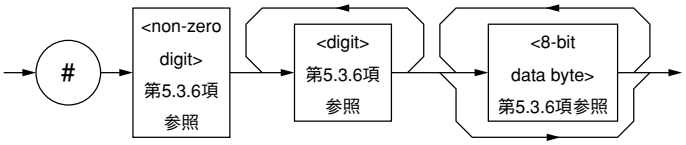
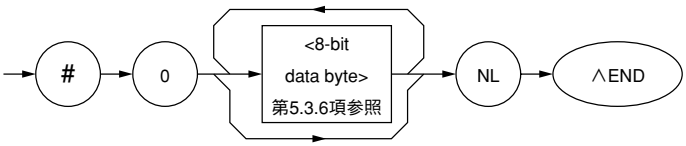
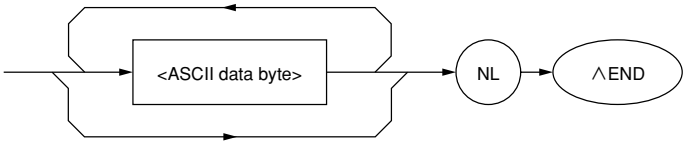
要素	機能
（5）HEXADECIMAL NUMERIC RESPONSE DATA ＜例＞ #HABC123 #H2DC3 #H8301	16進数値データです。 
（6）OCTAL NUMERIC RESPONSE DATA ＜例＞ #Q37 #Q26703 #Q30562	8進数値データです。 
（7）BINARY NUMERIC RESPONSE DATA ＜例＞ #B011101 #B1011 #B1011	2進数値データです。 

表6-3（つづき）

要素	機能
（8） STRING RESPONSE DATA ＜例＞ "This is a text" "Say,""Hello""."	ASCII7ビットコードのすべてが使えます。文字列の両端は,必ずダブル引用符で囲まれます。文字列中のダブル引用符は,引用符1個につき,同じ引用符が2個連続した2連引用符となります。CR,LF,スペースが使えるので,テキストをプリンタやCRTへ出力するのに適しています。 
（9） DEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA ＜例＞ 11256099Dを 4バイトで転送 ↓ #1400ABC123	固定長8ビットバイナリのブロックデータです。 大量なデータ,8ビット拡張ASCIIコード,非ディスプレイデータなどの転送に適しています。（各要素の説明は,「第5.3.6項　<ARBITRARY BLOCK PROGRAM DATA>」を参照してください。） 
（10） NIDEFINITE LENGTH ARBITRARY BLOCK RESPONSE DATA ＜例＞–250, –50, 120,…を不定長転送 ↓ #0FF06FFCE0078	不定長8ビットバイナリのブロックデータです。このため,最初のデータの前に#0をおきます。また,最後のデータの次はNL＾ENDでターミネートされます。 
（11） ARBITRARY ASCII RESPONSE DATA ＜例1＞ <ASCII Byte><ASCII Byte>NL＾END ＜例2＞ NL＾END	NL文字を除いたASCIIデータバイトを区切らないで送ります。このため,最後のデータの次はNL＾ENDでターミネートされます。 

第7章　共通コマンド

この章では，IEEE488.2で規定されている共通コマンドと共通問い合わせコマンドについて説明します。これら共通コマンドは，インタフェースメッセージで使用されるバスコマンドではありません。デバイスメッセージと同様に，共通コマンドは，バスのデータモードすなわちATNラインが偽のときに使用されるデータメッセージの一つであって，IEEE488.2対応機種であれば，他社の製品を含むすべての測定器に共通に使用することができます。IEEE488.2共通コマンドは，必ず“*”で始まります。

7.1　サポートコマンドの分類とリファレンス

本器に対する当社サポート対象コマンドの機能グループ別の証明を表7-1に示します。各コマンドの説明は，次ページからアルファベット順で示します。

表7-1

グループ	グループ別機能	ニーモニック
システムデータ	システムに接続されているデバイス固有の情報,たとえば,そのデバイスの製造メーカ名･形式･シリアル番号などを返します。	*IDN? *OPT?
内部オペレーション	デバイス内部の制御: (a) デバイスをレベル3でリセット (b) デバイス内部のセルフテストとエラーの有無の検知	*RST *TST?
同期	デバイスとコントローラの同期を下記により行います。 (a) サービスリクエスト待ち (b) デバイスの出力キュー応答待ち (c) シーケンシャル実行の強制	*OPC *OPC? *WAI
ステータス&イベント	ステータスバイトは,ステータスサマリメッセージによって構成されています。そのメッセージの個々のサマリビットは,標準イベントレジスタ,出力キュー,および拡張イベントレジスタまたは拡張キューから供給されます。そこで,これらのレジスタやキューにあるデータをセット･クリア･有効化･無効化,さらにはレジスタの設定状況を問い合わせによって知るため,3個のコマンド,4個の問い合わせが用意されています。	*CLS *ESE *ESE? *ESR? *SRE *SRE? *STB?

*CLS　Clear Status Command

（ステータスバイトレジスタのクリア）

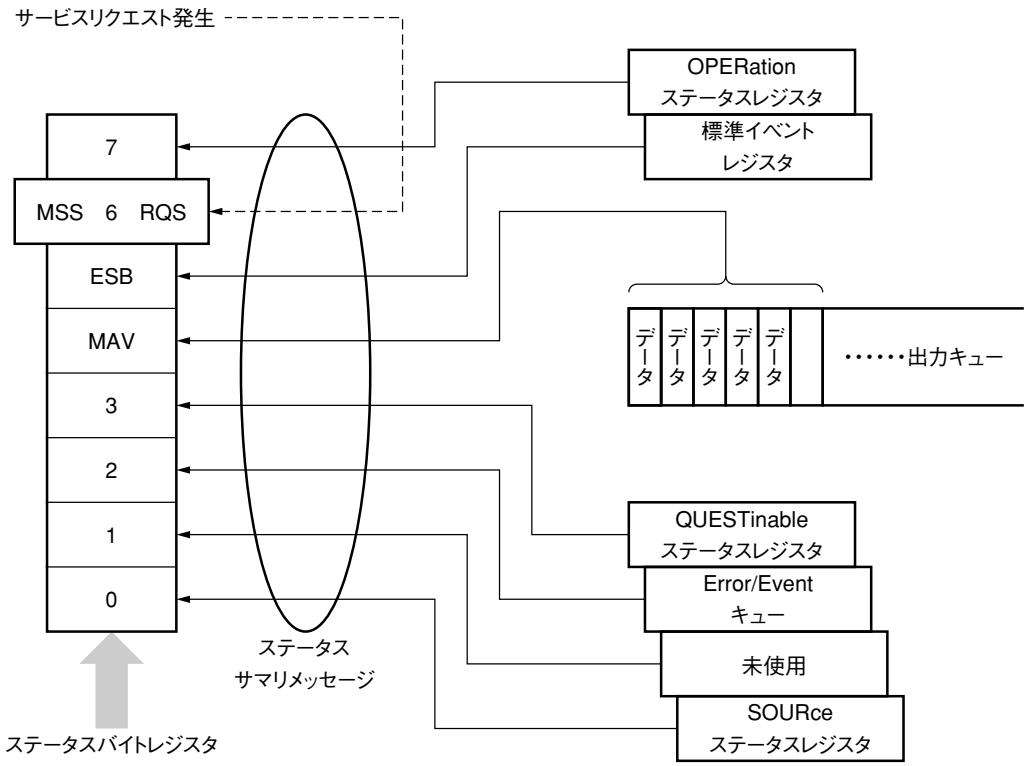
（1）書式

*CLS

（2）解説

*CLS共通コマンドは，出力キューとそのMAVサマリメッセージを除くすべてのステータスデータ構造(すなわち，これらのイベントレジスタおよびキュー)をクリアし，これに応じてそれらに対応するサマリメッセージもクリアします。

<PROGRAM MESSAGE TERMINATOR>の後，あるいは<Query MESSAGE UNIT>要素の前に*CLSコマンドを送出すると，すべてのステータスバイトはクリアされます。この方法により出力キューは，すべての未読み出しのメッセージもクリアされます。なお，各イネーブルレジスタの設定値については，*CLSによって影響されません。



*ESE　Standard Event Status Enable Command

（標準イベントステータスイネーブルレジスタのセットまたはクリア）

（1）書式

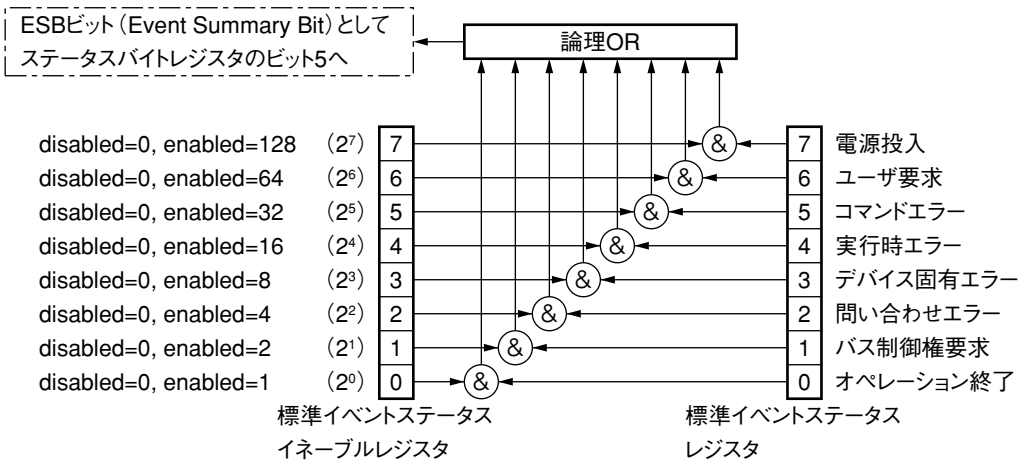
*ESE<HEADER SEPARATOR><DECIMAL NUMERIC PROGRAM DATA>

本書式において，

<DECIMAL NUMERIC PROGRAM DATA>＝ 0～255の整数に丸められ数値(2を底としてバイナリで重み付けされていること）

（2）解説

標準イベントステータスイネーブルレジスタのビット0～7に対応する値2^0=1，2^1=2，2^2=4，2^3=5，2^4=16，2^5=32，2^6=64，2^7=128の中から，enabledにしたいビットを選択したときのビット桁値の総和がプログラムデータとなります。disabledにしたいビット桁値は，0となります。



*ESE?　Standard Event Status Enable Quer

（標準イベントステータスイネーブルレジスタの現在値をレスポンス）

（1）書式

*ESE?

（2）解説

標準イベントステータスイネーブルレジスタの値である“NR1”を返します。

（3）レスポンスメッセージ

NR1=0～255

*ESR?　Standard Event Status Register Query

（標準イベントステータスレジスタの現在値を返す）

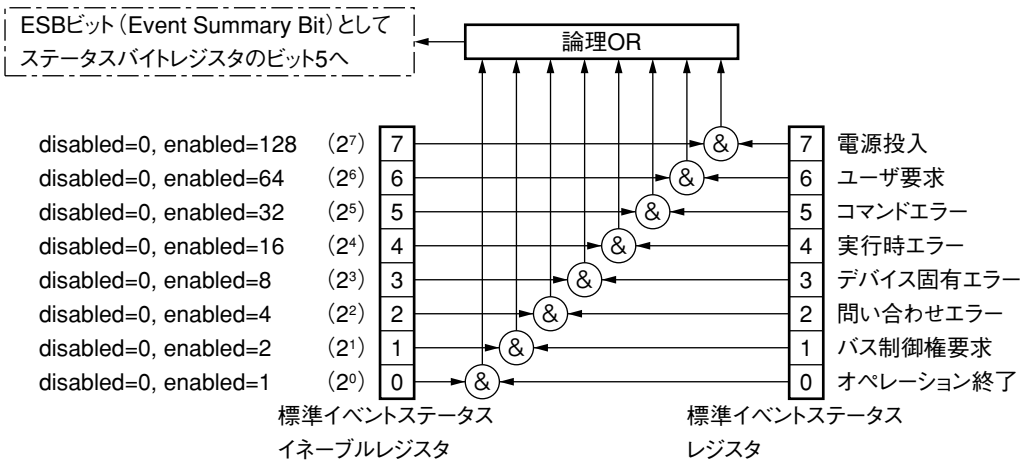
（1）書式

*ESR?

（2）解説

標準イベントステータスレジスタの現在値をNR1で返します。標準イベントステータスレジスタのビット0～7に対応する値2^0=1，2^1=2，2^2=4，2^3=8，2^4=16，2^5=32，2^6=64，2^7=128の中で，イベントが発生したビット桁値の総和がNR1となります。たとえば，ビット2とビット4でイベントが発生したときは，値は20(=2^2+2^4)になります。この値が読み取られると，このレジスタはクリアされます。また，標準イベントステータスイネーブルレジスタ(*ESE)によってenabledされているビットの論理OR値がイベントサマリビットとしてステータスバイトレジスタのビット5へ送られます。

標準イベントステータスレジスタの各ビットの内容は第8.4.1項で説明します。各ビットに対応したイベントが発生すると，そのビットに"1"がセットされます。



（3）レスポンスメッセージ　NR1

NR1=0～255

*IDN?　Identification Query

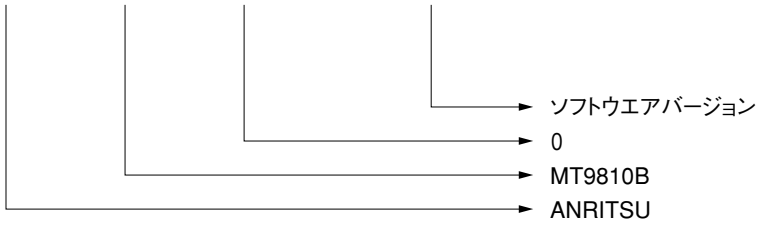
（製品のメーカ名・形名・シリアル番号・ファームウエアレベルを返す）

（1）書式

*IDN?

（2）解説

製品メーカ名・形名・シリアル番号・ファームウエアレベルを返します。



形名MT9810Bの製造メーカがアンリツで，そのシリアル番号が0，ソフトウエアまたはハードウエアのバージョンNo.が1の場合，*IDN?共通問い合わせをデバイスに送ると，上記に示した四つのフィールドから成るレスポンスメッセージを返します。

フィールド1 ..... 製品メーカ名（当社の場合，ANRITSU）
フィールド2 ..... 形名
フィールド3 ..... 製造番号－シリアル番号（当社の場合，0）
フィールド4 ..... ファームウエアバージョンNo.（制御ソフトウエアバージョン&光学ソフトウエアバージョン）

フィールド3，4のシリアル番号およびファームウエアバージョンNo.に情報を提供する意図がなければ，ASCII文字“0”を返すことができます。

（3）レスポンスメッセージ

上記の四つのフィールドをコンマで区切って構成したレスポンスメッセージを，<ARBITRARY ASCII RESPONSE DATA>で送ります。

<フィールド1>,<フィールド2>,<フィールド3>,<フィールド4>
レスポンスメッセージの全長は，≦72文字です。

*OPC　Operation Complete Command

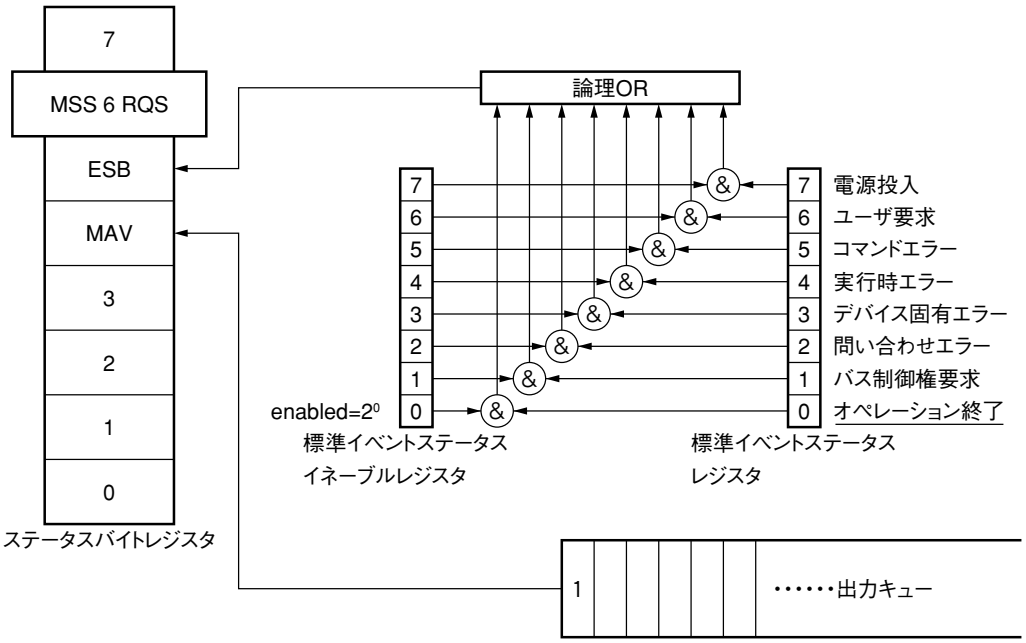
（デバイス動作が終了すると，標準イベントステータスレジスタのビット0をセット）

（1）書式

*OPC

（2）解説

選択されたペンディング中のデバイス動作がすべて終了したら，標準イベントステータスレジスタの中のビット0，すなわち「オペレーション終了ビット」をセットします。ただし，本器の場合にはオーバーラップコマンドは存在しませんのでこのコマンドは意味を持ちません。



*OPC?　Operation Complete Query

（デバイス動作が終了すると，出力キューに“1”を立てMAVサマリメッセージを発生させる）

（1）書式

*OPC?

（2）解説

選択されたペンディング中のデバイス動作がすべて終了したら，出力キューに“1”を立てMAVサマリメッセージが発生するまで待ちます。

（3）レスポンスメッセージ

“1”を<NR1 NUMERIC RESPONSE DATA>で返します。

*RST　Reset Command

（デバイスをレベル3でリセット）

（1）書式

*RST

（2）解説

*RST（Reset）コマンドは，デバイスをレベル3（表4-1参照）でリセットします。レベル3における初期化対象項目は，下記のとおりです。

(a) デバイス固有の機能・状態をそれまでの来歴にかかわらず，ある既知の状態に戻します。
(b) *DDTコマンドによって定義されたマクロをデバイスで定義された状態にします。
(c) マクロ動作を禁止し，マクロコマンドを受け付けないモードにします。また，マクロ定義を設計者が示す状態に戻します。
(d) デバイスをOCISステート（Operation Complete Command Idle State）にします。この結果，オペレーション終了ビットを標準イベントステータスレジスタに立てることはできません。

第8.1項

(e) デバイスをOQISステート（Operation Complete Query Idle State）にします。この結果，オペレーション終了ビット1を出力キューに立てることができません。MAVビットはクリアされます。

*RSTコマンドは，下記の事項には影響を与えません。

(a) IEEE488.1インタフェースの状態
(b) デバイスアドレス
(c) 出力キュー
(d) サービスリクエストイネーブルレジスタ
(e) 標準イベントステータスイネーブルレジスタ
(f) Power-on-status-clearフラグ設定
(g) デバイスの規格に影響する校正データ
(h) RS-232Cインタフェース条件

*OPT?　Option Identification Query

（実装されているオプションリストを報告する）

（1）書式

*OPT?

（2）解説

実装されているオプションの状態を0または1で返します。

（3）レスポンスメッセージ

上記三つのフィールドをコンマで区切って構成したレスポンスメッセージを<ARBITRARY ASCII RESPONSE DATA>で送ります。
現在はオプションがありませんので，“0”を返します。

*SRE　Service Request Enable Command

（サービスリクエストイネーブルレジスタのビットをセット）

（1）書式

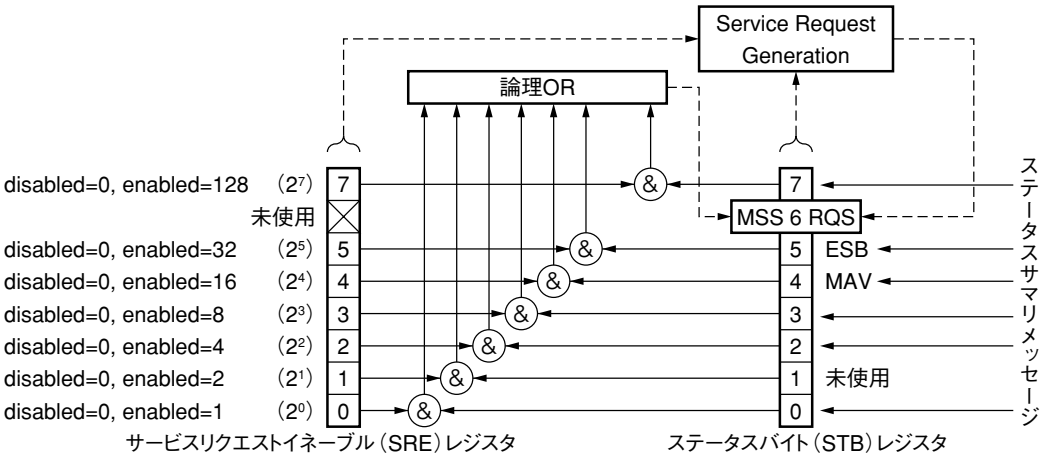
*SRE<HEADER SEPARATOR><DECIMAL NUMERIC PROGRAM DATA>

本書式において，

<DECIMAL NUMERIC PROGRAM DATA>＝ 0～255の整数に丸められ数値（2を底としてバイナリで重み付けされていること）

（2）解説

サービスリクエストイネーブルレジスタのビット0～7に対応する値 2^0=1，2^1=2，2^2=4，2^3=8，2^4=16，2^5=32，2^7=128の中から，enabledにしたいビットを選択したときのビット桁値の総和がプログラムデータとなります。disabledにしたいビット桁値は，0となります。



*SRE?　Service Request Enable Query

（サービスリクエストイネーブルレジスタの現在値を返す）

（1）書式

*SRE?

（2）解説

サービスリクエストイネーブルレジスタの値である“NR1”を返します。

（3）レスポンスメッセージNR1

NR1＝ビット6（RQSビット）はセットできないので，NR1=0～63 or 128～191

*STB?　Read Status Byte Command

（MSSビットを含むステータスバイトの現在値を返す）

（1）書式

*STB?

（2）解説

*STB?問い合わせは，バイナリで重み付けされたステータスバイトレジスタの値とMSS（Master Summary Status）サマリメッセージの値の総和を<NR1 NUMERIC RESPONSE DATA>として返します。

（3）レスポンスメッセージ

レスポンスメッセージは，<NR1 NUMERIC RESPONSE DATA>=0～255の整数で，ステータスバイトレジスタの各ビット桁値の総和になります。ステータスバイトレジスタのビット0～5と7はそれぞれ1，2，4，8，16，32および128に，またMSSビットは64に重み付けされています。MSSはサービスをリクエストする原因を少なくとも一つあることを示します。表7-2に，本器のステータスバイトレジスタの条件を示します。

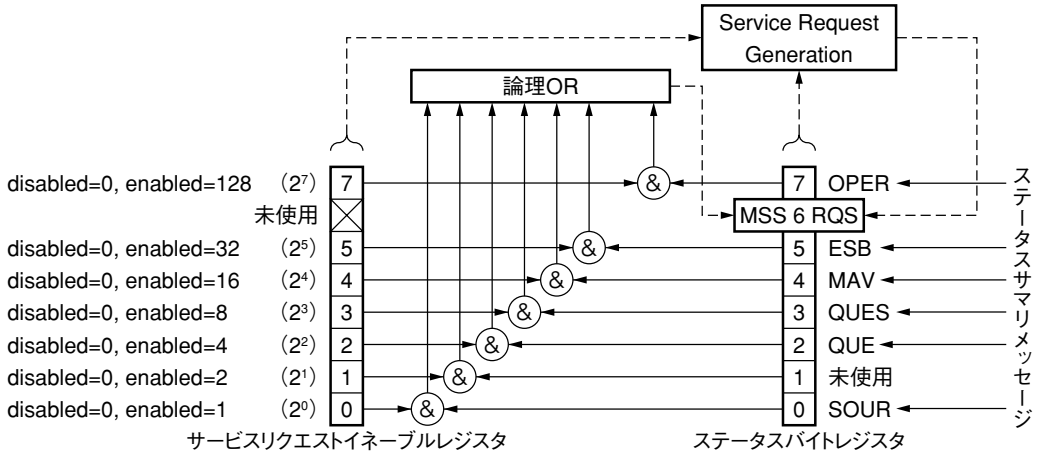


表7-2

ビット	ビットの重み	ビット名	ステータスバイトレジスタの条件	
7	128	OPER	1=OPERationステータスレジスタに状態遷移あり	0=なし
6	64	MSS/RQS	1=ビット7,ビット5～0に関するサービス要求あり	0=なし
5	32	ESB	1=標準イベントステータスレジスタに状態遷移あり	0=なし
4	16	MAV	1=出力キューにデータあり	0=なし
3	8	QUES	1=QUWStionableステータスレジスタに状態遷移あり	0=なし
2	4	QUE	1=エラー/イベントキューにデータあり	0=なし
1	2	————		0=未使用
0	1	SOUR	1=SOURceステータスレジスタに状態遷移あり	0=なし

各ステータスレジスタの詳細は第8章をご覧ください。

*TST?　Self-Test Query

（内部セルフテストを実行し，エラーの有無を返す）

（1）書式

*TST?

（2）解説

*TST?問い合わせは，デバイス内部のセルフテストを実行します。テスト結果は出力キューにおかれます。出力キューのデータは，エラーを起こすことなくテストが完了したかどうかを示します。セルフテストの実行にあたっては，オペレータの介入を必要としません。

（3）レスポンスメッセージ

レスポンスメッセージは，<NR1 NUMERIC RESPONSE DATA>で送ります。
データの範囲=－32767～32767

NR1=－ ... エラーなしで，テストを完了したことを示します。
NR1=1 ..... テストが実行されなかったか，実行されてもエラーを発生したことを示します。

*WAI　Wait-to-Continue Command

（デバイスがコマンド実行中は，次のコマンドを待機させる）

(1) 書式

*WAI

(2) 解説

*WAI共通コマンドは，オーバーラップコマンドをシーケンシャルコマンドとして実行します。
コントローラから送られてくるコマンドまたは問い合わせがデバイスでなにかを実行している間でも，次に送られるコマンドが実行を開始することができれば，最初に実行中のコマンドまたは問い合わせをオーバーラップコマンドといいます。
オーバーラップコマンドの次に，*WAI共通コマンドを実行しますと，デバイスがコマンド実行中は，次のコマンドを待機させ，実行が終了してから，次のコマンドの実行を許します。これは，シーケンシャルコマンドと同じ動作です。
ただし，本器の場合にはオーバーラップコマンドは存在しませんのでこのコマンドは意味を持ちません。

第8章　ステータスストラクチャー

この章では，IEEE488.2規格で定義されているデバイスのステータス報告とそのデータ構造およびデバイスとコントローラ間の同期テクニックについて説明します。

コントローラに送るステータスバイト（STB：Status Byte）は，IEEE488.1規格に基づいていますが，その構成ビットはステータスサマリメッセージと呼ばれ，レジスタやキュー（待ち行列）に蓄えられたデータの現在の内容を要約して表わしたものです。

以下に，このステータスサマリメッセージビットおよびこのステータスサマリメッセージビットを生成するためのステータスデータ構造，ならびにこのステータスメッセージを使ったデバイスとコントローラ間の同期テクニックについて説明します。

本機能はGPIBインタフェースバスを使用して外部コントローラから制御を行う際の機能ですが，RS-232Cインタフェースを使用して外部コントローラから制御を行う場合も，一部の機能を除いて，本機能を使用することができます。

8.1　IEEE488.2標準ステータスのモデル

下図にIEEE488.2で定められているステータスデータ構造の標準モデル図を示します。

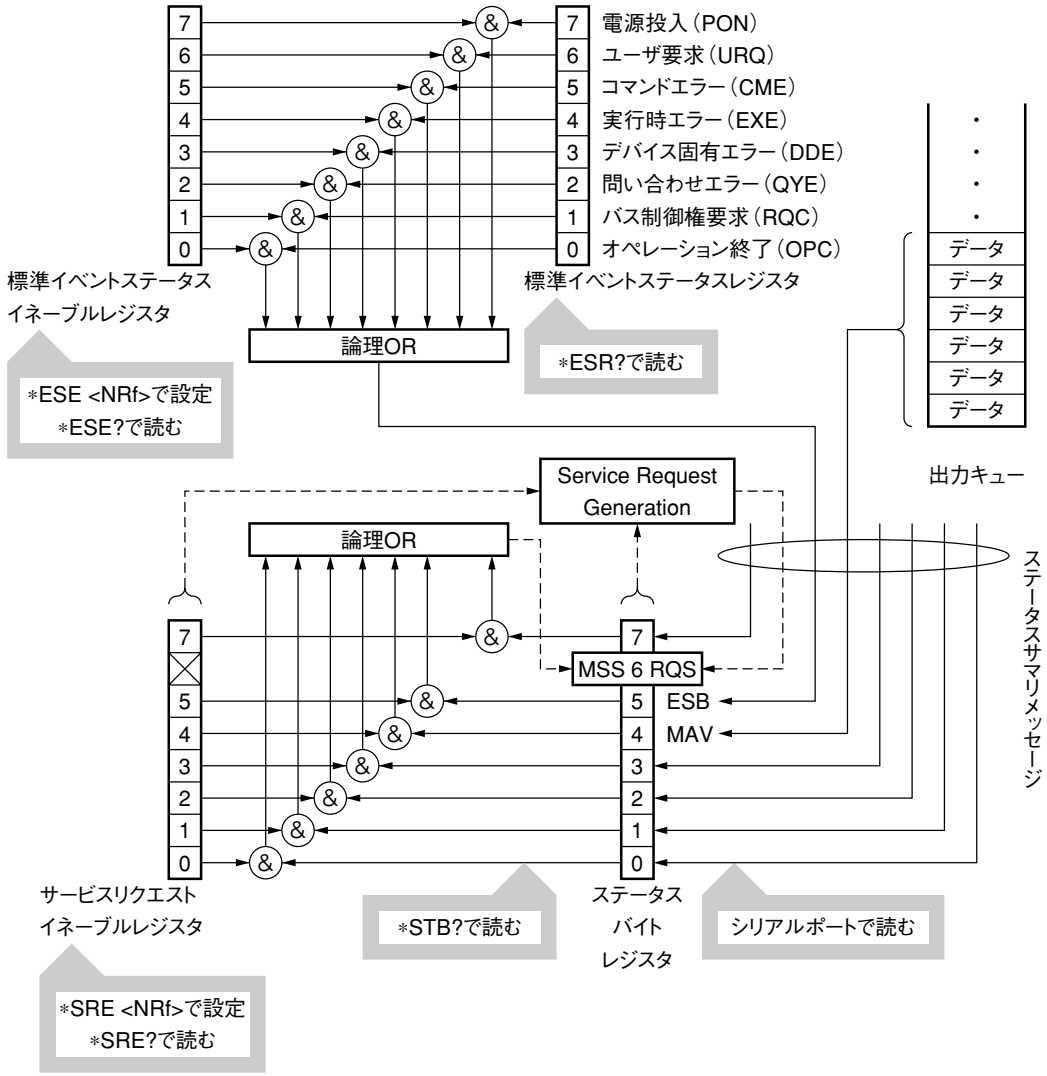


図8-1　標準ステータスモデル図

ステータスモデルでは，IEEE488.1ステータスバイトが使用されます。そのステータスバイトは，ステータスデータ構造から供給される7個のサマリメッセージビットで構成されます。これらのサマリメッセージビットを生成するため，ステータスデータ構造は，レジスタモデルとキューモデルの2種類から構成されます。

レジスタモデル

デバイスの遭遇した事象(event)および状態(condition)を記録するための一組のレジスタ，これをレジスタモデル(registermodel)といいます。その構造はイベントステータスレジスタ(Event Status Register)とイベントステータスイネーブルレジスタ(Event Status Enable Register)とから構成され，両者のANDが0でないとき，ステータスビットの対応ビットが1となります。それ以外の場合は0となります。そして，それらの論理ORの結果が1であれば，サマリメッセージビットは，1となります。論理ORの結果が0であれば，サマリメッセージビットは，0となります。

キューモデル

順序を待つ状態値または情報をシーケンシャルに記録するための待ち行列で，これをキューモデル(queuemodel)といいます。キュー構造では，キューにデータがあるときだけ対応ビットが1となり，キューが“空”であれば0となります。

上記で説明したレジスタモデルとキューモデルをもとに，IEEE488.2のステータスデータ構造の標準モデルは，2種類のレジスタモデルと1個のキューモデルから構成されています。

(1) 標準イベントステータスレジスタ(Standard Event Status Register)と標準イベントステータスイネーブルレジスタ(Standard Event Status Enable Register)

これらは前記のレジスタモデルの構造を持ち，この内容はデバイスが遭遇する事象の中で，下記の8種類の事象の各ビットを標準事象として，標準イベントステータスレジスタに立てます。

(a) 電源投入
(b) ユーザ要求
(c) コマンドエラー
(d) 実行時エラー
(e) デバイス固有エラー
(f) 問い合わせエラー
(g) バス制御権要求
(h) オペレーション終了

論理OR出力ビットは，イベントステータスビット(ESB：Event Status Bit)サマリメッセージとして，ステータスバイト(STB：Status Byte)レジスタのビット5(DI06)に要求表示されます。

(2) ステータスバイト(STB：Status Byte)レジスタとサービスリクエストイネーブル(SRE：Service Request Enable)レジスタ

ステータスバイトレジスタは，RQSビットおよびステータスデータ構造からの7個のサマリメッセージビットがセット可能なレジスタで，サービスリクエストイネーブルレジスタと組で使用され，両者のORが0でないときSRQをONにします。このときのステータスバイトレジスタのビット(DI07)は，RSQビットとしてシステム予約されており，このビットにより外部コントローラにサービス要求のあることを報告します。このSRQの仕組みはIEEE488.1の規格に従っています。

(3) 出力キュー(Output Queue)

これはキューモデルの構造を持ち，この内容は出力バッファにデータのあることを知らせるMAVサマリメッセージとしてステータスバイトレジスタのビット4(DI05)に要約表示されます。

8.2　ステータスバイトレジスタ

ステータスバイトレジスタは，デバイスのSTBとRQS（またはMSS）メッセージから構成されます。IEEE488.1では，STBとRQSメッセージの伝達（reporting）方法については定義していますが，セットおよびクリアのプロトコルとSTBの意味については定義していません。IEEE488.2では，デバイスのステータスサマリメッセージおよび*STB?共通問い合わせに応じて，STBと共にビット6に送出されるMSS（Master Summary Status）について定義しています。

8.2.1　ESBおよびMAVサマリメッセージ

ESBサマリメッセージおよびMAVサマリメッセージについて説明します。

（1）ESBサマリメッセージ

ESB（Event Summary Bit）サマリビットは，IEEE488.2で定義されたメッセージで，ステータスバイトレジスタのビット5に現われます。このビットの状態は，標準イベントステータスレジスタを最後にリード後またはクリア後において，イベント発生が有効となるようにサービスリクエストイネーブルレジスタを設定した状態で，IEEE488.2で定義された事象が少なくとも一つ以上発生したかどうかを示すものです。ESBサマリメッセージビットは，イベント発生が有効となるように設定された状態で，標準イベントステータスレジスタに登録されたイベントが一つでもTRUEにセットされれば，TRUEとなります。逆にESBサマリビットは，イベント発生が有効となるように設定された状態でも，登録されたイベントの発生がひとつもないときにFALSEとなります。

（2）MAVサマリメッセージ

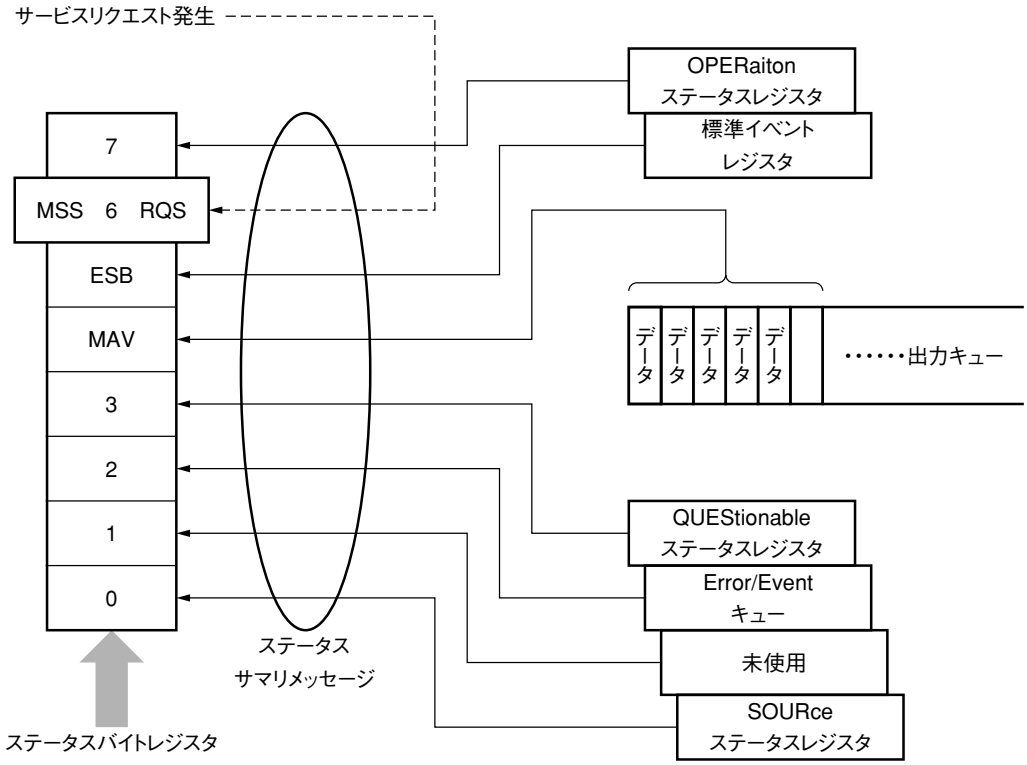
MAV（Message Available）サマリビットは，IEEE488.2で定義されたメッセージで，ステータスバイトレジスタのビット4に現われます。このビットの状態は，出力キューが“空”であるかどうかを示します。デバイスがコントローラからレスポンスメッセージの送出要求を受け付ける用意ができているときに，MAVサマリメッセージビットは1（TRUE）となり，出力キューが“空”のときに0（FALSE）となります。このメッセージはコントローラとの情報交換に同期を取るために利用されます。たとえば，コントローラがデバイスに問い合わせコマンドを送り，MAVがTRUEになるのを待つというように使うことができます。そして，デバイスが応答をするのを待つ間，他の処理をすることができます。もし，初めにMAVをチェックすることなしに出力キューを読み取り始めた場合は，すべてのシステムバス動作はデバイスが応答するまで待たされます。

8.2.2 装置固有のサマリメッセージ

IEEE488.2では，ステータスバイトレジスタのビット7(DIO8)，ビット3(DIO4)～ビット0(DIO1)をステータスレジスタのサマリビットとして使うか，キューにデータのあることを知らせるビットとして使うかは，決められていません。これらのビットは，装置固有のサマリメッセージとして利用することができます。

装置固有サマリメッセージは，それぞれレジスタモデルまたはキューモデルのステータスデータ構造を持ちます。すなわち，このステータスデータ構造は事象および状態を並列的に報告する一組のレジスタであるか，または状態および情報を順次報告する1個のキューです。サマリビットは対応するステータスデータ構造の現在の状態を要約表示します。レジスタモデルの場合は，一つ以上のTRUEの発生が有効となるように設定された事象が存在するとき，またキューモデルの場合は，キューが“空”でないときサマリメッセージはTRUEとなります。

本器では下記に示すようになっています。SCPI規格によりビット7をOPERationステータスレジスタのイベントサマリビットに，ビット3をQUEStionableステータスレジスタのイベントサマリビットに，ビット2をError/Event キューのサマリビットに割り当てています。また，bit1を未使用とし，bit0を本器固有のサマリメッセージとして，SOURceステータスレジスタのイベントサマリビットに割り当てています。



8.2.3　ステータスバイトレジスタの読み出しとクリア

ステータスバイトレジスタの内容は，シリアルポール，または*STB?共通問い合わせを使って読み取ります。どちらの方法でもIEEE488.1のSTBメッセージを読み取りますが，ビット6(位置)に送られる値はその方法によって異なります。

ステータスバイトレジスタの内容は，*CLSコマンドによってクリアすることができます。

(1) シリアルポールを使って読む(GPIBインタフェースバス使用時のみ)

IEEE488.1によるシリアルポールが行われた場合，デバイスは7ビットのステータスバイトと，IEEE488.1によるRQSメッセージビットを返送しなければなりません。IEEE488.1によれば，RQSメッセージはデバイスがSRQをTRUEで送出していたかどうかを示します。ステータスバイトの値は，シリアルポールを行っても変化しません。デバイスは，ポーリングされた直後rsvメッセージをFALSEにセットしなければなりません。これにより，新たなサービス要求のための原因が発生する前に，再度デバイスがポーリングされた場合，RQSメッセージはFALSEとなっています。

(2) *STB?共通問い合わせを使って読む

*STB?共通問い合わせは，デバイスにステータスバイトレジスタの内容とMSSサマリメッセージからの一つの<NR1 NUMERIC RESPONSE DATA>を送出させます。応答はバイナリで重み付けされたステータスバイトレジスタの値とMSSサマリメッセージの値の総和を表わします。ステータスバイトレジスタのビット0～5，7はそれぞれ1，2，4，8，16，32，および128に，またMSSは64に重み付けされます。これにより，RQSメッセージの代わりにMSSサマリメッセージがビット6位置に現われることを除いては，*STB?に対する応答は，シリアルポールに対する対応と一致します。

(3) MSS(Master Summary Status)の定義

デバイスに少なくとも一つのサービスを要求する原因があることを示します。MSSメッセージは*STB?問い合わせに対するデバイスの応答の中でビット6に現われますが，シリアルポールに対する応答としては現われません。また，IEEE488.1のステータスバイトの一部とみなしてはなりません。MSSはステータスバイトレジスタとSRQイネーブル(SRE)レジスタのビットの組み合わせによる総合的ORにより構成されます。これを具体的に示すと，結局MSSは下記のように定義されます。

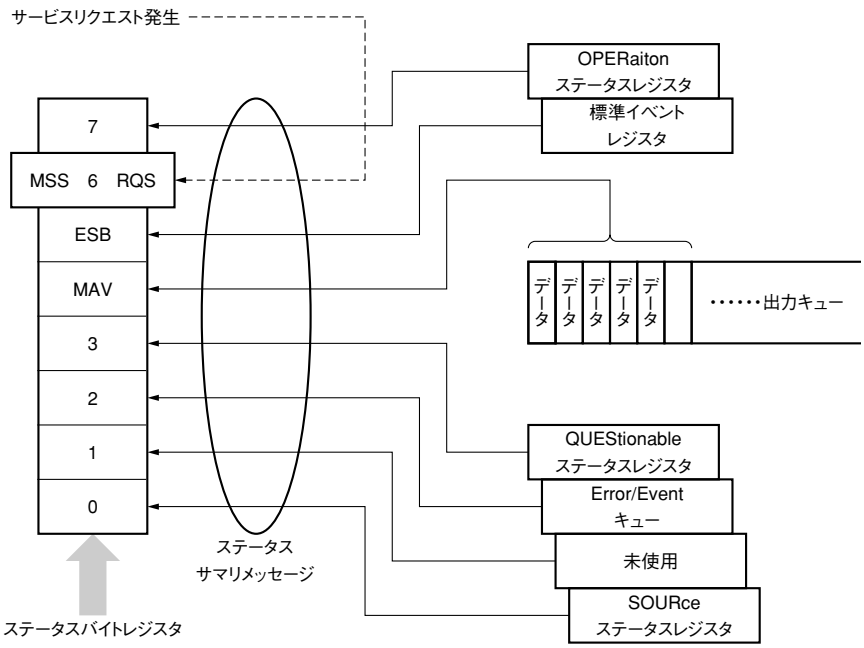
（STB Register bit 0 AND SRE Register bit 0）
OR
（STB Register bit 1 AND SRE Register bit 1）
OR
：
：
（STB Register bit 5 AND SRE Register bit 5）
OR
（STB Register bit 7 AND SRE Register bit 7）

MSSの定義において，ステータスバイトレジスタと，SRQイネーブルレジスタ双方のビット6の状態を無視していますので，MSSの値を算出するに当たっては，ステータスバイトをビット6が常に0である8ビットの値として取り扱ってかまいません。

（4）*CLS共通コマンドによるステータスバイトレジスタのクリア

*CLS共通コマンドは，出力キューとそのMAVサマリメッセージを除くすべてのステータスデータ構造(すなわち，これらのイベントレジスタおよびキュー)をクリアし，これに応じてそれらに対応するサマリメッセージもクリアします。

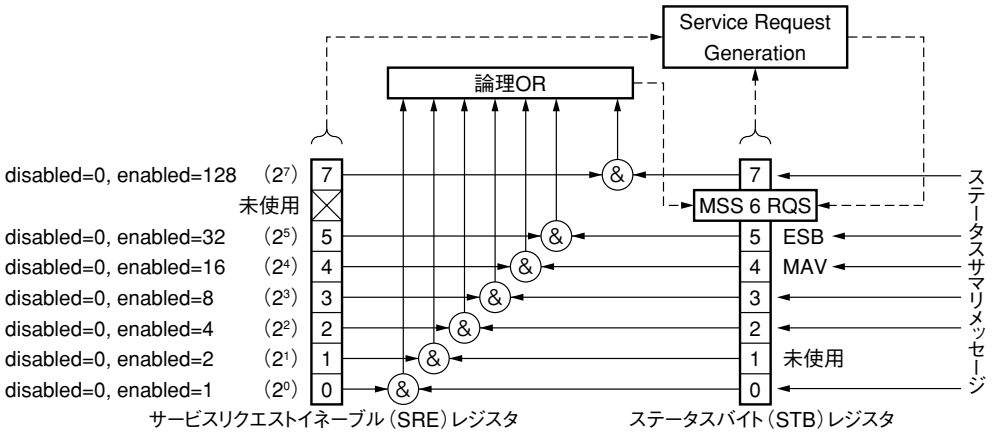
<PROGRAM MESSAGE TERMINATOR>の後，あるいは<Query MESSAGE UNIT>要素の前に*CLSコマンドを送出すると，すべてのステータスバイトはクリアされます。この方法により出力キューのすべての未読み出しのメッセージはクリアされ，MAVメッセージはFALSEとなります。また，*STB?に対する応答時，MSSメッセージもFALSEになります。なお，各イネーブルレジスタの設定値については，*CLSによって影響されません。



8.3　SRQのイネーブル

SRQのイネーブルにより，ステータスバイトレジスタの中のどのサマリメッセージをサービスリクエストに対して有効にするか無効にするかを選択できます。下図で示すサービスリクエストイネーブル(SRE)レジスタがサマリメッセージを選択する手段として使用されます。

サービスリクエストイネーブルレジスタ上のビットは，ステータスバイト(STB)レジスタ上のビットと対応しています。サービスリクエストイネーブルレジスタ上の有効なビットに対応するステータスバイト中のビットに1が立つと，デバイスは，RQSビットを1として，サービスリクエストをコントローラに対して行います。たとえば，サービスリクエストイネーブルレジスタのビット4をイネーブルにセットしておくと，出力キューにデータがあれば，MAVビットに1が立つたびに，サービスリクエストをコントローラに対して行います。



(1)　サービスリクエストイネーブルレジスタの読み出し

サービスリクエストイネーブルレジスタの内容は，*SRE?共通問い合わせを使って読み出せます。この問い合わせに対するレスポンスメッセージは，<NR1 NUMERIC RESPONSE DATA>=0～255の整数で，サービスリクエストイネーブルレジスタの各ビット桁値の総和となります。サービスリクエストイネーブルレジスタのビット0～5，7はそれぞれ1，2，4，8，16，32および128に重み付けされています。使用されないビット6は，常に0でなければなりません。

(2)　サービスリクエストイネーブルレジスタの更新

サービスリクエストイネーブルレジスタは，*SRE共通命令を使って書き込まれます。*SRE共通命令の後には<DECIMAL NUMERIC PROGRAM DATA>要素が続きます。<DECIMAL NUMERIC PROGRAM DATA>は整数に丸められ，2を基数としてバイナリで表現され，サービスリクエストイネーブルレジスタの各ビット桁値(ウェイト値)の総和を表わします。このビット値は，1がenabledの状態を表わし，0がdisabledの状態を表わします。ビット6の値は常に無視しなければなりません。

(3) サービスリクエストイネーブルレジスタのクリア

*SRE共通コマンドの実行または電源再投入によって，サービスリクエストイネーブルレジスタをクリアできます。

*SRE共通コマンドの場合は，<DECIMAL NUMERIC PROGRAM DATA>の要素の値を0にすれば，サービスリクエストイネーブルレジスタをクリアすることができます。レジスタをクリアすることで，ステータス情報がrsvローカルメッセージを発生することを禁止できますので，この結果サービス要求は発生しなくなります。

電源再投入の場合は，電源ONステータスクリアフラグがTRUEで，*PSCコマンドが未装備のためクリア阻止の実行がされていなければ，電源再投入時にサービスリクエストイネーブルレジスタはクリアされます。

8.4　標準イベントステータスレジスタ

8.4.1　標準イベントステータスレジスタのビット定義

標準イベントステータスレジスタは，IEEE488.2対応機種であれば，すべてのデバイスが装備しなければならないイベントステータスレジスタです。下図に，標準イベントステータスレジスタモデルの動作を示します。レジスタモデルの動作それ自身は，これまでに説明してきたのと同じなので，表8-1では，標準イベントステータスレジスタの各ビットの意味について，IEEE488.2の定義を説明します。

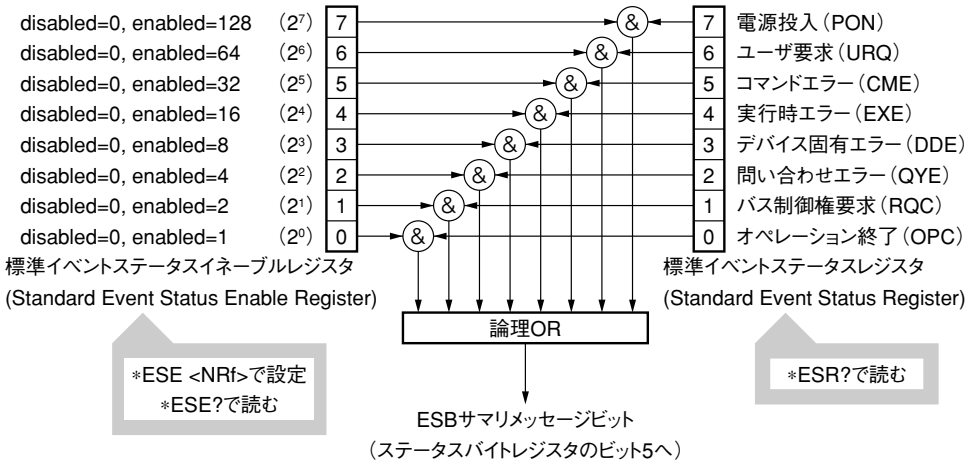


表8-1

ビット	イベント名	説明
7	電源投入（PON-Power on）	電源投入がOFFからONへと変化しています。
6	ユーザ要求 （URQ-User Request）	ローカル制御（rtl）を要求しています。 このビットは,デバイスのリモート／ローカル状態とは無関係に発生します。 本器では使用しておりませんので常に0となります。
5	コマンドエラー （CME-Command Error）	文法に従わないプログラムメッセージ,ミススペルのコマンドまたはプログラムメッセージの中でGETコマンドを受信しています。
4	実行時エラー （EXE-Execution Error）	文法に問題はありませんが,実行できないプログラムメッセージを受信しています。
3	デバイス固有エラー （DDE-Device-dependent Error）	CME,EXE,QYE以外の原因によるエラーが発生しています。
2	問い合わせエラー （QYE-Query Error）	出力キューにデータがないのに,出力キューからデータを読もうとした,または出力キューのデータがなんらかの原因,たとえばオーバーフローなどで失われた,などです。
1	バス制御権要求 （RQC-Request Control）	自らがアクティブコントローラになることを要求しています。 本器では使用しておりませんので常に0となります。
0	オペレーション終了 （OPC-Operation Complete）	デバイスが,ペンディング中の,指定した動作を終了して,新しい命令を受ける準備ができています。 このビットは,*OPCコマンドに対してのみ応答し,オペレーション終了ビットを立てます。

8.4.2 問い合わせエラーの詳細

表8-2

No.	項目	説明
1	不完全なプログラムメッセージ	デバイスがプログラムメッセージを受信中に,プログラムメッセージターミネータを受信する前にコントローラからMTAを受信した場合,デバイスはそれまでに入力した不完全なプログラムメッセージを破棄し,次のプログラムメッセージを待ちます。不完全なプログラムメッセージの破棄の動作では,デバイスは入出力バッファをクリアし,問い合わせエラーをステータス報告部に伝え,標準ステータスレジスタのビット2に問い合わせエラービットをセットします。
2	レスポンスメッセージの出力の中断	デバイスがレスポンスメッセージを送信中で,レスポンスメッセージターミネータを転送し終わる前にコントローラからMLAを受信した場合には,デバイスは自動的にレスポンスメッセージ出力中断動作を行い,次のプログラムメッセージを待ちます。レスポンスメッセージ出力中断動作では,デバイスは出力バッファをクリアし,問い合わせエラーをステータス報告部に伝え,標準ステータスレジスタのビット2に問い合わせエラービットをセットします。
3	レスポンスメッセージを読まないで次のプログラムメッセージを送信した場合	コントローラが問い合わせメッセージを含むプログラムメッセージの送信に続いて,さらに次のプログラムメッセージを送信したためにデバイスがレスポンスメッセージの出力をできなかった場合,デバイスはレスポンスメッセージの破棄を行い,次のプログラムメッセージを待ちます。2と同じように問い合わせエラーをステータス報告部に伝えます。
4	出力キューのオーバーフロー	問い合わせメッセージを多数含むプログラムメッセージを実行していくとき,出力キュー(256バイト)に入りきらないほど多くのレスポンスメッセージが発生することがあります。出力キューが満杯になっても,まだ問い合わせメッセージが入力され,それにともないレスポンスメッセージを出力しなければいけないとき,出力キューがオーバーフロー状態になります。出力キューがオーバーフローすると,デバイスは出力キューをクリアし,レスポンスメッセージ作成部をリセットします。 また,問い合わせエラービットをステータス報告部の標準イベントステータスレジスタのビット2にセットします。

8.4.3　標準イベントステータスレジスタの読み取り・書き込み・クリア

表8-3

読み取り	*ESR?共通問い合わせにより破壊的に読み取られます。 すなわち,読み取られた後,クリアされます。レスポンスメッセージは,イベントビットに2進数の重みを付けて10進数変換した<NR1>です。
書き込み	クリアすることを除く,外部から書き込みは行えません。
クリア	次の場合にのみクリアされます。 （1）*CLSコマンド受信 （2）電源ONステータスクリアフラグが真ならば,電源ONのとき。電源ONシーケンス実行中のデバイスは最初,標準イベントステータスレジスタをクリアしますが,その後,このシーケンス中に発生するイベントを記録。（たとえばPONイベントビットのセットなど） （3）*ESR?問い合わせコマンドに対して,イベントが読み込まれた。

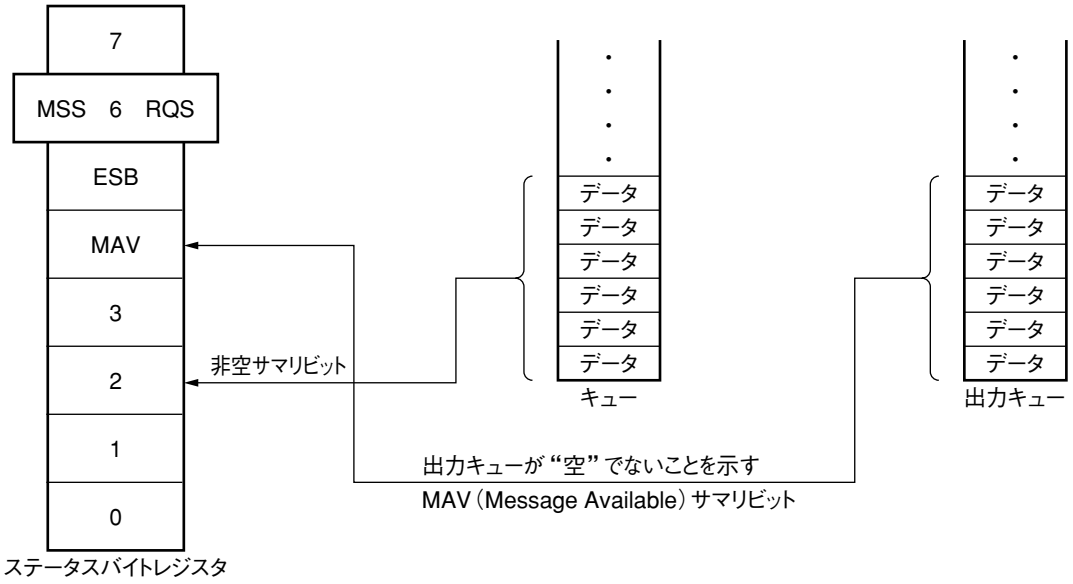
8.4.4　標準イベントステータスイネーブルレジスタの読み取り・書き込み・クリア

表8-4

読み取り	*ESR?共通問い合わせにより非破壊的に読み取られます。 すなわち,読み取られた後も,クリアされません。レスポンスメッセージは,2進数の重みを付けて2進～10進変換された<NR1>で返されます。
書き込み	*ESS共通コマンドによって書き込まれます。レジスタのビット0～8は,それぞれ1,2,4,8,16,32,64,および128に重み付けされていますので,書き込みデータは,その中から希望のビット桁値を合計した<10進数値プログラムデータ>で送ります。
クリア	下記の場合にのみクリアされます。 （1）データ値0の*ESEコマンドを受信 （2）電源ONステータスクリアフラグが真の状態での電源ON時,および*PSCコマンドが用意されていない場合の電源ON時。 標準イベントステータスイネーブルレジスタは,下記の事項に影響されません。 （1）IEEE488.1のデバイスクリアファンクションの状態変化 （2）*RST共通コマンドの受信 （3）*CLS共通コマンドの受信

8.5　キュー（待ち行列）モデル

下図の右側に，ステータスデータ構造のキューモデルを示します。キューとは，順番に並べられた情報リストを含むデータ構造で，シーケンシャルステータスその他の情報を報告する手段を提供します。そうした情報がキューの中に存在することは，サマリメッセージに要約表示されます。キューの内容は，デバイスがトーカアクティブステート（TACS）にあるとき，ハンドシェークにより読み取られます。



サマリメッセージの中で，MAVサマリメッセージをステータスバイトのビット4へ出力するキューは，「出力キュー」と呼ばれ，必須となっています。MAVサマリメッセージをステータスバイトレジスタのビット0～3，7のうちのどれかへ出力可能なキューは，単に「キュー」と呼ばれ，オプションとなっています。ステータスバイトレジスタのビット0～3，7には，レジスタモデルからのサマリメッセージも接続可能ですので，サマリメッセージの種類は，デバイスごとに相違します。

ここでは，「出力キュー」と一般のキューを，表8-5で比較して示します。

表8-5　出力キューとキューの比較表

項目	出力キュー	キュー
データ入出力方式	FIFO形	必ずしもFIFO形である必要はありません。
読み出し	IEEE488.2メッセージ交換プロトコルを通じてのみ読み出されます。読み出されるレスポンスメッセージユニットのタイプは,問い合わせによって決定されます。	装置固有 の問い合わせコマンドによって読み出されます。読み出されるレスポンスメッセージユニットは同じタイプでなければなりません。
書き込み	プログラムメッセージ要素が直接書き込まれることはありません。 IEEE488.2メッセージ交換プロトコルを通じてのみシステムインタフェースとやりとりされます。	プログラムメッセージ要素が直接書き込まれることはありません。 コード化されたデバイスの情報を示します。
サマリメッセージ	出力キューが“空”でないときにはTRUE (1),“空”のときにはFALSE (0) となります。 MAVサマリメッセージは,デバイスとコントローラとの情報交換の同期に用いられます。	キューが“空”でないときにはTRUE (1) ,“空”のときにはFALSE (0) となります。
クリア	下記のうちのどれかが発生したときにはクリアされます。 (a) キューの中のすべてのアイテムが読み取られた。 (b) メッセージ交換初期化のため,DCLバスコマンドを受信した。 (c) 電源投入で,ponがTrueとなった。 (d) UNTERMINATEDまたはINTERRUPTED動作	下記のどれかが発生したときには,クリアされます。 (a) キューの中のすべてのアイテムが読み取られた。 (b) *CLSコマンドを受信した。 (c) 装置固有のこの他の手段。

8.6　拡張ステータスバイト

SCPIの規格では，ステータスバイトのビット7を“OPERation Status”，ビット3を“QUEStionable Status”として使用することを規定しています。また，ビット2を“Error/EventQueue”に割り当てます。また，本器固有のメッセージとして,ビット0を"SOURce Status"のステータスサマリビットとして割り当てます。
各ステータスレジスタは下記のような構成をとります。

(1) CONDITION REGISTER

コンディションレジスタは，外部(コントローラ)からの読み出し後も変化しません。また，外部(コントローラ)からのいかなるコマンドによっても設定できず，測定器内部の状態変化によってのみセットされます。

(2) TRANSITION FILTER

トランジションフィルタは，コンディションレジスタに通知された状態変化をイベントステータスレジスタに通知するかどうかを決定するためのフィルタです。
0→1の状態変化のためのフィルタをPトランジションフィルタ，1→0の状態変化のためのフィルタをNトランジションフィルタと呼びます。これらのフィルタは，外部(コントローラ)からの要求に従ってマスクパターンとして書き換えられます。(ビット別にセット／クリア可)これらのマスクパターンは，コントローラからの読み出し後も変化しません。

(3) EVENT REGISTER

イベントレジスタは，測定器内部からコンディションレジスタ，P／Nトランジションフィルタを介して間接的にセットできます。アプリケーションプログラムからイベントレジスタへ直接にアクセスできません。

(4) EVENT ENABLE REGISTER

イベントレジスタに対するイベントイネーブルレジスタです。

(5) ERROR/EVENT QUEUE

このキューにメッセージが蓄えられている間，ステータスバイトレジスタの対応ビットがセットされます。メッセージキューにメッセージがなくなると，ステータスバイトレジスタの対応ビットはクリアされます。

8.6.1　ステータスレジスタ

STATus:PRESet

(1) 機能

イネーブルレジスタ，トランジションフィルタの初期化

(2) プログラムメッセージ

```
STATus:PRESet
```

(3) 解説

イネーブルレジスタ，トランジションフィルタの初期化を行います。各レジスタは表8-6のように設定されます。

表8-6

Register	Filter/Enable	Preset Value
Operation	Enable	all 0
	PTR	all 1
	NTR	all 0
Questionable	Enable	all 0
	PTR	all 1
	NTR	all 0

<node>:CONDition

(1) 機能

コンディションレジスタの確認

(2) プログラムメッセージ

```
<node>:CONDition?
```

(3) レスポンスメッセージ

```
<code>
```

(4) パラメータ

```
<code>:= {n | 0≦n≦32767}
```

(5) 解説

コンディションレジスタの値の総和を返します。
<node>で，指定するコンディションレジスタの項目を決定します。

<node>:ENABle

(1) 機能

イベントイネーブルレジスタの設定

(2) プログラムメッセージ

```
<node>:ENABle <mask>
<node>:ENABle?
```

(3) レスポンスメッセージ

```
<mask>
```

(4) パラメータ

<mask>:= {n | 0≦n≦32767}

(5) 解説

イベントイネーブルレジスタのenableにしたいビットを選択したときのビット桁値の総和がパラメータになります。disableにしたいビット桁値は0になります。
<node>で，指定するイベントイネーブルレジスタの項目を決定します。

<node>[:EVENt]

(1) 機能

イベントレジスタの確認

(2) プログラムメッセージ

```
<node>[:EVENt]?
```

(3) レスポンスメッセージ

```
<code>
```

(4) パラメータ

<code>:= {n | 0≦n≦32767}

(5) 解説

イベントレジスタの値の総和を返します。
<node>で，指定するイベントレジスタの項目を決定します。

<node>:NTRansition

（1）機能

Nトランジションレジスタの設定

（2）プログラムメッセージ

```
<node>:NTRansition <mask>
<node>:NTRansition?
```

（3）レスポンスメッセージ

```
<mask>
```

（4）パラメータ

<mask>:= {n | 0≦n≦32767}

（5）解説

Nトランジションレジスタのenableにしたいビットを選択したときのビット桁値の総和がパラメータになります。disableにしたいビット桁値は0になります。
<node>で，指定するNトランジションレジスタの項目を決定します。

<node>:PTRansition

（1）機能

Pトランジションレジスタの設定

（2）プログラムメッセージ

```
<node>:PTRansition <mask>
<node>:PTRansition?
```

（3）レスポンスメッセージ

```
<mask>
```

（4）パラメータ

<mask>:= {n ¦ 0≦n≦32767}

（5）解説

Pトランジションレジスタのenableにしたいビットを選択したときのビット桁値の総和がパラメータになります。disableにしたいビット桁値は0になります。
<node>で，指定するPトランジションレジスタの項目を決定します。

8.6.2 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装置の状態を示します。
コマンドは以下のとおりです。このコマンドを，ステータスレジスタの<node>部にはめ込んで使用します。

コマンド	内容
STATus:OPERation	オペレーションステータスレジスタ
STATus:OPERation:SETTling	光源ユニットの温度状態
STATus:OPERation:MEASuring	センサユニットの測定状態
STATus:OPERation:CORRection	センサユニットのオフセット処理状態
STATus:OPERation:AVERage	センサユニットの平均化処理状態

STATus:OPERation

(1) 機能

オペレーションステータスレジスタ

(2) 解説

オペレーションステータスレジスタの参照をします。
装置の状態をビットに割り付けて表わします。各ビットの内容は以下のとおりです。

ビット	内容
1	光源ユニットの温度状態
4	センサユニットの測定状態
7	センサユニットのオフセット処理状態
8	センサユニットの平均化処理状態

STATus:OPERation:SETTling

(1) 機能

光源ユニットの温度状態

(2) 解説

光源ユニットの温度状態を表わし，使用できるか否かを示します。
各ビットは，ビット0をチャンネル1として順番にチャンネルに対応しており，その状態により，光源ユニットが使用できるか否かを示します。

ビット	対応チャンネル
0	チャンネル1
1	チャンネル2

状態	内容
0	使用不可
1	使用可

STATus:OPERation:MEASuring

(1) 機能

センサユニットの測定状態

(2) 解説

センサユニットの測定状態を表わします。
各ビットは，ビット0をチャンネル1として順番にチャンネルに対応しており，その状態により，センサユニットが測定中であるか否かを示します。

ビット	対応チャンネル
0	チャンネル1
1	チャンネル2

状態	内容
0	測定していない
1	測定中

STATus:OPERation:CORRecting

(1) 機能

センサユニットのオフセット処理状態

(2) 解説

センサユニットのオフセット処理状態を表わします。
各ビットは，ビット0をチャンネル1として順番にチャンネルに対応しており，その状態により，センサユニットがオフセット中であるか否かを示します。

ビット	対応チャンネル
0	チャンネル1
1	チャンネル2

状態	内容
0	オフセットを実行していない
1	オフセット実行中

STATus:OPERation:AVERaging

(1) 機能

センサユニットの平均化処理状態

(2) 解説

センサユニットの平均化処理状態を表わします。
各ビットは，ビット0をチャンネル1として順番にチャンネルに対応しており，その状態により，センサユニットが平均化処理を実行しているか否かを示します。

ビット	対応チャンネル
0	チャンネル1
1	チャンネル2

状態	内容
0	平均化処理を実行していない
1	平均化処理実行中

8.6.3　QUESTIONABLEステータスレジスタ

QUESTIONABLEステータスレジスタのコマンドは以下のとおりです。このコマンドを，ステータスレジスタの<node>部にはめ込んで使用します。

コマンド	内容
STATus:QUEStionable:POWer	QUESTIONABLEステータスレジスタ
STATus:QUEStionable:POWer:OVERRange	センサユニットのオーバーレンジ
STATus:QUEStionable:POWer:UNDerrange	センサユニットのアンダーレンジ
STATus:QUEStionable:POWer:CURRent	電流異常
STATus:QUEStionable:POWer:ENVTemp	温度異常
STATus:QUEStionable:POWer:POWer	電源異常

STATus:QUEStionable:POWer

（1）機能

QUESTIONABLEステータスレジスタ

（2）解説

QUESTIONABLEステータスレジスタの参照をします。
装置の状態をビットに割り付けて表わします。各ビットの内容を下記に示します。

ビット	内容
0	センサユニットのオーバーレンジ
1	センサユニットのアンダーレンジ
2	リモートインタロック
6	電流異常
7	温度異常
8	電源異常

STATus:QUEStionable:POWer:OVERRange

(1) 機能

センサユニットのオーバーレンジ

(2) 解説

センサユニットのオーバーレンジを表わします。
各ビットは，ビット0をチャンネル1として順番にチャンネルに対応しており，その状態により，センサユニットがオーバーレンジであるか否かを示します。

ビット	対応チャンネル
0	チャンネル1
1	チャンネル2

状態	内容
0	オーバーレンジでない
1	オーバーレンジ

STATus:QUEStionable:POWer:UNDerrange

(1) 機能

センサユニットのアンダーレンジ

(2) 解説

センサユニットのアンダーレンジを表わします。
各ビットは，ビット0をチャンネル1として順番にチャンネルに対応しており，その状態により，センサユニットがアンダーレンジであるか否かを示します。

ビット	対応チャンネル
0	チャンネル1
1	チャンネル2

状態	内容
0	アンダーレンジでない
1	アンダーレンジ

STATus:QUEStionable:POWer:CURRent

(1) 機能

電流異常

(2) 解説

電流異常の発生状態を表わします。
各ビットは，ビット0をチャンネル1として順番にチャンネルに対応しており，その状態により，電流異常が発生しているか否かを示します。

ビット	対応チャンネル
0	チャンネル1
1	チャンネル2

状態	内容
0	電流異常なし
1	電流異常発生

STATus:QUEStionable:POWer:ENVTemp

(1) 機能

温度異常

(2) 解説

温度異常の発生状態を表わします。
各ビットは，ビット0をチャンネル1として順番にチャンネルに対応しており，その状態により，温度異常が発生しているか否かを示します。

ビット	対応チャンネル
0	チャンネル1
1	チャンネル2

状態	内容
0	温度異常なし
1	温度異常発生

STATus:QUEStionable:POWer:POWer

(1) 機能

電源異常

(2) 解説

電源異常の発生状態を表わします。
各ビットは，ビット0をチャンネル1として順番にチャンネルに対応しており，その状態により，電源異常が発生しているか否かを示します。

ビット	対応チャンネル
0	チャンネル1
1	チャンネル2

状態	内容
0	電源異常なし
1	電源異常発生

8.6.4　SOURCE ステータスレジスタ

SOURCEステータスレジスタは光源ユニットの光出力状態を示します。コマンドは以下のとおりです。このコマンドをステータスレジスタの<node>部にはめ込んで使用します。

コマンド	内容
STATus:SOURce:	SOURCEステータスレジスタ
STATus:SOURce:SLOT	光源ユニットの光出力状態

STATus:SOURce

（1）機能

SOURCEステータスレジスタ

（2）解説

SOURCEステータスレジスタの参照をします。
装置の状態をビットに割り付けて表わします。各ビットの内容を下記に示します。

ビット	対応チャンネル
0	光源ユニットの光出力状態

STATus:SOURce:SLOT

（1）機能

光源ユニットの光出力状態

（2）解説

光源ユニットの測定状態を表わします。
各ビットは，ビット0をチャンネル1として順番にチャンネルに対応しており，その状態により光源ユニットの光出力状態を示します。

ビット	対応チャンネル
0	チャンネル1
1	チャンネル2

状態	内容
0	光出力OFF状態
1	光出力ON状態

第9章　デバイスメッセージ詳細

9.1　本体関係

9.1.1　DISPlay:BRIGhtness

（1）機能

輝度設定

（2）プログラムメッセージ

```
DISPlay:BRIGhtness <ratio>
DISPlay:BRIGhtness?
```

（3）レスポンスメッセージ

```
DISPLAY:BRIGHTNESS <ratio>
```

（4）パラメータ

```
<ratio>:= {f|0.1 ≤ f ≤ 1.0}
```

（5）解説

ディスプレイの輝度を割合で設定します。
<ratio>を0.1としたときに最も暗く，1としたときに最も明るくなります。
輝度は10段階に丸められます。

0.1← <ratio> → 1
暗い　　明るい

9.1.2　DISPlay[:STATe]

（1）機能

表示のON/OFF

（2）プログラムメッセージ

```
DISPlay[:STATe] <sw>
DISPlay[:STATe]?
```

（3）レスポンスメッセージ

```
<status>
```

（4）パラメータ

```
<sw>:= {ON,OFF,1,0}
<status>:= {1,0}
1 ..... ON
0 ..... OFF
```

（5）解説

ディスプレイの表示／非表示を切り替えます。

9.1.3 SYSTem:BEEPer:STATe

(1) 機能

ブザー設定

(2) プログラムメッセージ

```
SYSTem:BEEPer:STATe <level>
SYSTem:BEEPer:STATe?
```

(3) レスポンスメッセージ

```
SYSTEM:BEEPER:STATE <level>
```

(4) パラメータ

```
<level>:= {0,1,2,3,4}
```

(5) 解説

ブザー音のレベルを設定します。
ブザー音は<level>によって下記のように設定されます。

<level>	意味
0	ブザーOFF
1	レベル小
2	
3	
4	レベル大

9.1.4　SYSTem:CHANnel:STATe

（1）機能

挿入ユニットの問い合わせ

（2）プログラムメッセージ

```
SYSTem:CHANnel:STATe?
```

（3）レスポンスメッセージ

```
SYSTEM:CHANNEL:STATE <uid> (@ <uno>){,<uid>
(@ <uno>)}
```

（4）パラメータ

```
<uid>:= {OPM,OLS}
<uno>:= {1,2}
```

（5）解説

現在，挿入されているすべてのユニットについて種類と識別番号を出力します。ユニットがひとつも挿入されていないときは，レスポンスデータとして“NOUNIT”が返されます。
ユニットの種類は<uid>で示され，下記のように解釈されます。

<uid>	ユニット名
OPM	光センサユニット
OLS	光源ユニット

9.1.5　SYSTem:COMMunicate:GPIB:HEAD

（1）機能

ヘッダ付加の設定

（2）プログラムメッセージ

```
SYSTem:COMMunicate:GPIB:HEAD <flag>
SYSTem:COMMunicate:GPIB:HEAD?
```

（3）レスポンスメッセージ

```
SYSTEM:COMMUNICATE:GPIB:HEAD <status>
```

（4）パラメータ

```
<flag>:= {ON,OFF,1,0}
<status>:= {1,0}
1 ..... ON
0 ..... OFF
```

（5）解説

レスポンスメッセージにヘッダを付けるか，付けないかを設定します。デフォルトはヘッダを付けません。
GPIBとシリアルに同じ設定項目がありますが（SYSTem:COMMunicate:GPIB:HEADと SYSTem:COMMunicate:SERial:HEAD），これらは独立しておらず一方を設定すると他方も同じ状態に設定されます。

9.1.6 SYSTem:COMMunicate:SERial:HEAD

(1) 機能

ヘッダ付加の設定

(2) プログラムメッセージ

```
SYSTem:COMMunicate:SERial:HEAD <flag>
SYSTem:COMMunicate:SERial:HEAD?
```

(3) レスポンスメッセージ

```
SYSTEM:COMMUNICATE:SERIAL:HEAD <status>
```

(4) パラメータ

```
<flag>:= {ON,OFF,1,0}
<status>:= {1,0}
1 ..... ON
0 ..... OFF
```

(5) 解説

レスポンスメッセージにヘッダを付けるか，付けないかを設定します。
デフォルトはヘッダを付けません。
GPIBとシリアルに同じ設定項目がありますが(SYSTem:COMMunicate:GPIB:HEADとSYSTem:COMMunicate:SERial:HEAD)，これらは独立しておらず一方を設定すると他方も同じ状態に設定されます。

9.1.7 SYSTem:DATE

(1) 機能

カレンダの設定

(2) プログラムメッセージ

```
SYSTem:DATE <year>,<month>,<day>
SYSTem:DATE?
```

(3) レスポンスメッセージ

```
SYSTEM:DATE <year>,<month>,<day>
```

(4) パラメータ

```
<year>:= {n|1990 ≤ n ≤ 2089}
<month>:= {n|1 ≤ n ≤ 12}
<day>:= {n|1 ≤ n ≤ 31}
```

(5) 解説

システムのカレンダを設定します。
<year>，<month>，<day>は，それぞれ年，月，日を表わします。

9.1.8　SYSTem:ERRor

（1）機能

エラー値の問い合わせ

（2）プログラムメッセージ

```
SYSTem:ERRor?
```

（3）レスポンスメッセージ

```
SYSTEM:ERROR <code>
```

（4）解説

SYSTem:ERRorに対するレスポンスとして，SCPIではエラーに対応したコードとメッセージを規定しています。本器がサポートするエラーメッセージの詳細については「第9.4項　エラーメッセージ」を参照してください。

9.1.9　SYSTem:TIME

（1）機能

時刻設定

（2）プログラムメッセージ

```
SYSTem:TIME <hour>,<minute>,<second>
SYSTem:TIME?
```

（3）レスポンスメッセージ

```
SYSTEM:TIME <hour>,<minute>,<second>
```

（4）パラメータ

```
<hour>:= {n|0 ≤ n ≤ 23}
<minute>:= {n|0 ≤ n ≤ 59}
<second>:= {n|0 ≤ n ≤ 59}
```

（5）解説

システムの時計を指定された時刻に設定します。
時刻は24時間制で指定し，<hour>，<minute>，<second>はそれぞれ時，分，秒を表わします。

9.2 光センサ

[1¦2]は，制御する光センサが挿入されているチャンネル番号を示します。チャンネル1の場合は，省略可能です。なお，カギカッコ([ ])は不要です。

例）

```
ABORT1  FETCH2:SCALAR:POWER:DC  SENSE:CORRECTION:COLLECT:ZERO
```

など

9.2.1 ABORt[1|2]

（1）機能

測定の停止

（2）プログラムメッセージ

```
ABORt[1|2]
```

（3）解説

記録測定を停止します。

9.2.2 FETCh[1|2][:SCALar]:POWer[:DC]

（1）機能

測定データの問い合わせ

（2）プログラムメッセージ

```
FETCh[1|2][:SCALar]:POWer[:DC]?
```

（3）レスポンスメッセージ

```
FETCH1|2 <level>
```

（4）パラメータ

```
<level>:= <NR3>
```

（5）解説

現在の測定データ値を返します。
測定データは，現在の単位に従いdBm値，W値またはdB値となります。

9.2.3　SENSe[1|2]:AVERage:COUNt

（1）機能

平均化回数の設定

（2）プログラムメッセージ

```
SENSe[1|2]:AVERage:COUNt <count>
SENSe[1|2]:AVERage:COUNt?
```

（3）レスポンスメッセージ

```
SENSE1|2:AVERAGE:COUNT <count>
```

（4）パラメータ

```
<count>:= {1,2,5,10,20,50,100,200,500,1000}
```

（5）解説

平均化処理における平均化回数を設定します。

9.2.4　SENSe[1|2]:BANDwidth

（1）機能

帯域設定

（2）プログラムメッセージ

```
SENSe[1|2]:BANDwidth <bw> [<unit>]
SENSe[1|2]:BANDwidth?
```

（3）レスポンスメッセージ

```
SENSE1|2:BANDWIDTH <bw>
```

（4）パラメータ

```
<bw>:= {0.1,1,10,100,1000,10000,20000,100000}
```
（単位：Hz）

```
<unit>:= {HZ,KHZ}
```

（5）解説

帯域を<bw>で指定された値にします。
<bw>はユニットに依存し，ユニットによっては設定できない値もあります。
プログラムメッセージでは，補助単位の使用ができます。
レスポンスメッセージでは，常にHz値で出力されます。

9.2.5 SENSe[1|2]:BANDwidth:AUTO

(1) 機能

オート帯域設定

(2) プログラムメッセージ

```
SENSe[1|2]:BANDwidth:AUTO <sw>
SENSe[1|2]:BANDwidth:AUTO?
```

(3) レスポンスメッセージ

```
SENSE1|2:BANDWIDTH:AUTO <status>
```

(4) パラメータ

```
<sw>:= {ON,OFF,1,0}
<status>:= {1,0}
1 ..... ON
0 ..... OFF
```

(5) 解説

帯域設定を自動設定とします。

9.2.6 SENSe[1|2]:CORRection:COLLect:ZERO

(1) 機能

ゼロセットの実行

(2) プログラムメッセージ

```
SENSe[1|2]:CORRection:COLLect:ZERO
SENSe[1|2]:CORRection:COLLect:ZERO?
```

(3) レスポンスメッセージ

```
SENSE1|2:CORRECTION:COLLECT <result>
```

(4) パラメータ

```
<result>:= <NR1>
```

(5) 解説

ゼロセットを実行します。
レスポンスメッセージは，下記の値を持ちます。
エラーコードの詳細については「第9.4項　エラーメッセージ」を参照してください。

<result>	状態
0	正常終了
1	リモートによるゼロセット処理が未実行
2	ゼロセット実行中
負数	エラー

9.2.7　SENSe[1|2]:CORRection[:LOSS[:INPut[:MAGNitude]]]

（1）機能

校正係数の設定

（2）プログラムメッセージ

```
SENSe[1|2]:CORRection[:LOSS[:INPut[:MAGNitude]]]
<cal>[DB]
SENSe[1|2]:CORRection[:LOSS[:INPut[:MAGNitude]]]?
```

（3）レスポンスメッセージ

```
SENSE1|2:CORRECTION:LOSS:INPUT:MAGNITUDE <cal>
```

（4）パラメータ

```
<cal>:= {f| –199.99 ≤ n ≤ 199.99}
```

（5）解説

校正係数を<cal>とします。
<cal>は常にdB値で受け付けます。単位は省略してもかまいません。

9.2.8　SENSe[1|2]:FETCh[:SCALar]:POWer[:DC]:MAXimum

（1）機能

最大値読み出し

（2）プログラムメッセージ

```
SENSe[1|2]:FETCh[:SCALar]:POWer[:DC]:MAXimum?
```

（3）レスポンスメッセージ

```
SENSE1|2:FETCH:SCALAR:POWER:DC:MAXIMUM <level>
```

（4）パラメータ

```
<level>:= <NR3>
```

（5）解説

統計測定の実行から現在までの最大測定データ値を出力します。
測定データの単位は，現在の測定値に従い“dBm”または“W”となります。

9.2.9 SENSe[1|2]:FETCh[:SCALar]:POWer[:DC]:MINimum

(1) 機能

最小値読み出し

(2) プログラムメッセージ

```
SENSe[1|2]:FETCh[:SCALar]:POWer[:DC]:MINimum?
```

(3) レスポンスメッセージ

```
SENSE1|2:FETCH:SCALAR:POWER:DC:MINIMUM <level>
```

(4) パラメータ

```
<level>:= <NR3>
```

(5) 解説

統計測定の実行から現在までの最小測定データ値を出力します。
測定データの単位は，現在の測定値に従い“dBm”または“W”となります。

9.2.10 SENSe[1|2]:FETCh[:SCALar]:POWer[:DC]:PTPeak

(1) 機能

最大／最小値差読み出し

(2) プログラムメッセージ

```
SENSe[1|2]:FETCh[:SCALar]:POWer[:DC]:PTPeak?
```

(3) レスポンスメッセージ

```
SENSE1|2:FETCH:SCALAR:POWER:DC:PTPEAK <level>
```

(4) パラメータ

```
<level>:= <NR3>
```

(5) 解説

統計測定の実行から現在までの最大測定値と最小測定値の差を出力します。
単位は，“dB”となります。

9.2.11 SENSe[1|2]:FILTer:BPASs:FREQuency

（1）機能

変調周波数の設定

（2）プログラムメッセージ

```
SENSe[1|2]:FILTer:BPASs:FREQuency CW|<freq>[<unit>]
SENSe[1|2]:FILTer:BPASs:FREQuency?
```

（3）レスポンスメッセージ

```
SENSE1|2:FILTER:BPASS:FREQUENCY <freq>
```

（4）パラメータ

```
<freq>:= {0,270,1000,2000}
```
（単位：Hz）
```
<unit>:= {HZ,KHZ}
```

（5）解説

測定する入力光の変調周波数を設定します。
<freq>の単位が省略されたときはHz値，<unit>により指定されたときはその単位で設定します。
0 HzはCWが設定されます。
レスポンスメッセージは，常にHz値で出力します。

9.2.12 SENSe[1|2]:INITiate[:IMMediate]

（1）機能

記録測定

（2）プログラムメッセージ

```
SENSe[1|2]:INITiate[:IMMediate]
```

（3）解説

設定された回数分の測定を行います。
回数は，“SENSe:TRIGger:COUNt”で設定します。

9.2.13 SENSe[1|2]:MEMory:COPY[:NAME]

(1) 機能

測定条件の保存／取り出し

(2) プログラムメッセージ

```
SENSe[1|2]:MEMory:COPY[:NAME] MC,<no>|<no>,MC
```

(3) パラメータ

```
<no>:= {0,1,2,3,4,5,6,7,8,9}
```

(4) 解説

<no>で指定されるメモリ番号で，測定条件の保存または読み出しをします。
“MC,<no>”は測定条件の保存，“<no>,MC”は測定条件の読み出しをします。
なお，<no>に“0”を指定したときは，初期条件の設定であるので読み出しのみ有効です。

9.2.14 SENSe[1|2]:MEMory:DATa

(1) 機能

記録測定データの読み出し

(2) プログラムメッセージ

```
SENSe[1|2]:MEMory:DATa? MD[,<start>[,<number>]]
```

(3) レスポンスメッセージ

```
SENSE1|2:MEMORY:DATA <number>{,<level>}*
```

(4) 解説

記録測定で測定したデータを読み出します。
<start>，<number>はそれぞれ読み出しの開始ポイント，読み出し個数です。これらが省略されたときには，<start>を1，<number>を測定したデータ数，として動作します。
<start>は測定データ数よりも大きな値を指定できません。<number>は測定データ数または<start>以降のデータ数より大きな値を指定してもかまいませんが，実際に取り出せるデータ数は有効な個数となります。
レスポンスメッセージは，実際に取り出したデータ数とその数分の測定データを出力します。測定データに単位は付きませんが，記録されたときの単位(“dBm” または “W”)での値となります。
この単位は“SENSe:MEMory:DATa:INFO”で知ることができます。単位が“W”であるとき補助単位は使用せず，指数表記となります。
測定データが存在しないときのレスポンスメッセージは，<number>のみを“0”として出力します。

9.2.15 SENSe[1|2]:MEMory:DATa:INFO

（1）機能

記録測定データ情報

（2）プログラムメッセージ

```
SENSe[1|2]:MEMory:DATa:INFO?
```

（3）レスポンスメッセージ

```
SENSE1|2:MEMORY:DATA:INFO V1.0,"<info>"
```

（4）解説

記録測定データの詳細情報を読み出します。
レスポンスメッセージの最初にある“V1.0”は，後に続く情報の識別に使用します。現在は“V1.0”のみです。
<info>には以下の情報が順番に並べられ，“;”で区切られています。

情報	内容
ユニット型名	測定した時のユニットの型名
測定日時	測定した日にちと時刻を以下の形式で表す YY/MM/DD,hh:mm:ss
アベレージ回数	測定した時のアベレージ回数
インターバル時間	測定した時の測定間隔
測定データ数	測定したデータ数
データ単位	“DBM”または“W”で表す
データ統計	最大値，最小値，peak-to-peak値，平均値を“,”で区切って並べてある

データ統計の単位は省略されていますが，データ単位に従った値で出力されます。データ単位が“DBM”であるとき，最大値，最小値，平均値は“dBm”で，peak-to-peak値は“W”で出力します。データ単位が“W”であるとき，最大値，最小値，平均値は“W”で，peak-to-peak値は“%”で出力します。
データ単位が“W”であるときのワット値は補助単位を使用せず，指数表記とします。
測定データが存在しないときのレスポンスメッセージは，<info>部を空文として出力します。

```
SENSE1:MEMORY:DATA:INFO V1.0,""
```

9.2.16 SENSe[1|2]:POWer:INTerval

(1) 機能

測定インターバルの設定

(2) プログラムメッセージ

```
SENSe[1|2]:POWer:INTerval <time>
SENSe[1|2]:POWer:INTerval?
```

(3) レスポンスメッセージ

```
SENSE1|2:POWER:INTERVAL <time>
```

(4) パラメータ

```
<time>:= {f|0.001 ≤ f ≤ 359999}
```

(5) 解説

測定間隔を設定します。
<time>は秒単位で設定し，分解能に丸められます。

9.2.17 SENSe[1|2]:POWer:RANGe:AUTO

(1) 機能

オートレンジの設定

(2) プログラムメッセージ

```
SENSe[1|2]:POWer:RANGe:AUTO <sw>
SENSe[1|2]:POWer:RANGe:AUTO?
```

(3) レスポンスメッセージ

```
SENSE1|2:POWER:RANGE:AUTO <status>
```

(4) パラメータ

```
<sw>:= {ON,OFF,1,0}
<status>:= 1,0
1 ..... ON
0 ..... OFF
```

(5) 解説

測定レンジを自動設定にするか，しないかの切り替えをします。
“ON”または“1”であるとき自動設定が有効，
“OFF”または“0”であるとき自動設定が無効になります。

9.2.18 SENSe[1|2]:POWer:RANGe[:UPPer]

（1）機能

マニュアルレンジの設定

（2）プログラムメッセージ

```
SENSe[1|2]:POWer:RANGe[:UPPer] <levle>[DBM]
SENSe[1|2]:POWer:RANGe[:UPPer]?
```

（3）レスポンスメッセージ

```
SENSE1|2:POWER:RANGE:UPPER <level>
```

（4）パラメータ

```
<level>:= {40,30,2 0,10,0,-10,-20,-30,-40,-50,-
60,-70,-80,-90,-100,-110}
```

（5）解説

測定レンジを<level>に固定して測定します。
<level>は，ユニットに依存しユニットによっては設定できない値もあります。
単位は“DBM”のみを受け付け，省略することもできます。

9.2.19 SENSe[1|2]:POWer:REFerence

（1）機能

リファレンス値の設定

（2）プログラムメッセージ

```
SENSe[1|2]:POWer:REFerence <type>,<level>[<unit>]
SENSe[1|2]:POWer:REFerence? <type>
```

（3）レスポンスメッセージ

```
SENSE1|2:POWER:REFERENCE <level>
```

（4）パラメータ

```
<type>:= {TOA,TOB,TOREF,0,1,2}
```

“TOREF”または“2”のとき

```
<level>:= {f(W)|1 × 10-16 ≤ f ≤ 99.999}
<level>:= {f(dBm)| -199.999 ≤ f ≤ +199.999}
<unit>:= {PW,NW,UW,MW,W,DBM}
```

“TOA”または“0”，“TOB”または“1”のとき

```
<level>:= {f(dB)| -199.999 ≤ f ≤ 199.999}
<unit>:= {DB}
```

(5) 解説

リファレンス測定時のリファレンス値を設定します。
“TOREF”は，指定されたレベル値をそのチャンネルのリファレンス値として設定します。SENSe1およびSENSe2のどちらでも使用できます。
リファレンス値は，W値，dBm値で設定でき，単位が省略されたときはdBm値として扱います。単位を“W”としたときは0.0001 pW～99.999 Wまで，“DBM”としたときは－199.999 dBm ～+199.999 dBmまで設定できます。
“TOA”または“TOB”は，センサユニットが2個挿入されている場合のみ有効であり，2チャンネル間のレベル差に対してのリファレンス値を設定します。つまり表示内容は，（レベル差－基準値）となります。（ここで，基準値=リファレンス値+相対値）
リファレンス値はdB値でのみ設定できます。したがって，単位を付加するときは“DB”のみが有効です。
“TOA”はSENSe2であるとき，“TOB”はSENSe1であるときにのみ有効となります。
“TOA”または“TOB”が指定されたときのレスポンスメッセージは，常にdB値で返します。
“TOREF”が指定されたときのレスポンスメッセージは，現在の単位に従ってW値またはdBm値で返します。

9.2.20 SENSe[1|2]:POWer:REFerence:DISPlay

(1) 機能

相対値表示

(2) プログラムメッセージ

```
SENSe[1|2]:POWer:REFerence:DISPlay
```

(3) 解説

表示値が0 dBとなるような相対値が設定され相対値表示となります。
このコマンドは現在の表示値を0 dBとしての相対測定であるので，絶対値表示，リファレンス表示であっても設定できます。表示値は，下記の式により求めることができます。
表示値=測定値－リファレンス値－相対値
ここで，絶対値表示から相対値表示となったときは，リファレンス値を0として扱います。

9.2.21 SENSe[1|2]:POWer:REFerence:STATe

（1）機能

リファレンス測定のON/OFF

（2）プログラムメッセージ

```
SENSe[1|2]:POWer:REFerence:STATe <sw>
SENSe[1|2]:POWer:REFerence:STATe?
```

（3）レスポンスメッセージ

```
SENSE1|2:POWER:REFERENCE:STATE <status>
```

（4）パラメータ

```
<sw>:= {ON,OFF,1,0}
<status>:= {1,0}
1 ..... ON
0 ..... OFF
```

（5）解説

リファレンス測定のON/OFFを設定します。

9.2.22 SENSe[1|2]:POWer:REFerence:STATe:RATio

(1) 機能

リファレンス選択

(2) プログラムメッセージ

```
SENSe[1|2]:POWer:REFerence:STATe:RATio <sel>
SENSe[1|2]:POWer:REFerence:STATe:RATio?
```

(3) レスポンスメッセージ

```
SENSE1|2:POWER:REFERENCE:STATE:RATIO <status>
```

(4) パラメータ

```
<sel>:= {TOA,TOB,TOREF,0,1,2}
<status>:= {0,1,2}
1 ..... TOA
1 ..... TOB
0 ..... TOREF
```

(5) 解説

リファレンス測定に方式を設定します。
<sel>によりリファレンスの測定方式を下記のようにします。

<sel>	測定方式
TOA(0)	(チャンネル2の測定値)−(チャンネル1の測定値)
TOB(1)	(チャンネル1の測定値)−(チャンネル2の測定値)
TOREF(2)	(指定チャンネルの測定値)−(基準値)

ここで基準値は，(リファレンス値)+(相対値)です。
“TOA”はSENSe2であるとき，“TOB”はSENSe1であるときにのみ有効です。
“TOREF”はSENSe1，SENSe2ともに有効です。

9.2.23 SENSe[1|2]:POWer:UNIT

(1) 機能

単位系の切り替え

(2) プログラムメッセージ

```
SENSe[1|2]:POWer:UNIT <unit>
SENSe[1|2]:POWer:UNIT?
```

(3) レスポンスメッセージ

```
SENSE1|2:POWER:UNIT <unit>
```

(4) パラメータ

```
<unit>:= {DBM,W}
```

(5) 解説

受光パワーの表示単位系を切り替えます。

9.2.24 SENSe[1|2]:POWer:WAVelength

（1）機能

波長の設定

（2）プログラムメッセージ

```
SENSe[1|2]:POWer:WAVelength <wavelength>[<unit>]
SENSe[1|2]:POWer:WAVelength?
```

（3）レスポンスメッセージ

```
SENSE1|2:POWER:WAVELENGTH <wavelength>
```

（4）パラメータ

<wavelength>:= {f(m)|380 × 10^{-9} ≤ f ≤ 1800 × 10^{-9}}
<wavelength>:= {f(Hz)|166.551 × 10^{12} ≤ f ≤ 788.927 × 10^{12}}
<unit>:= {NM,UM,M,HZ}

（5）解説

波長補正を<wavelength>の波長に設定します。
波長の設定範囲および分解能は，センサユニットに依存します。
プログラムメッセージにおいて単位が省略されたときは，m値として扱います。
レスポンスメッセージは，現在設定されている単位系（mまたはHz）に従って出力します。

9.2.25 SENSe[1|2]:POWer:WAVelength:UNIT

（1）機能

波長の表示単位

（2）プログラムメッセージ

```
SENSe[1|2]:POWer:WAVelength:UNIT <unit>
SENSe[1|2]:POWer:WAVelength:UNIT?
```

（3）レスポンスメッセージ

```
SENSE1|2:POWER:WAVELENGTH:UNIT <unit>
```

（4）パラメータ

```
<unit>:= {M,HZ}
```

（5）解説

波長の表示単位を切り替えます。

9.2.26 SENSe[1|2]:TRIGger:COUNt

(1) 機能

測定回数の設定

(2) プログラムメッセージ

```
SENSe[1|2]:TRIGger:COUNt <count>
```

(3) レスポンスメッセージ

```
SENSE1|2:TRIGGER:COUNT <count>
```

(4) パラメータ

```
<count>:= {n|1 ≤ n ≤ 1000}
```

(5) 解説

記録測定における測定データ数を設定します。

9.2.27 SENSe[1|2]:TRIGger[:SEQuence][:IMMediate]

(1) 機能

統計測定の再実行

(2) プログラムメッセージ

```
SENSe[1|2]:TRIGger[:SEQuence][:IMMediate]
```

(3) 解説

測定データの最小値・最大値・peak-to-peak値を計測します。

9.2.28 READ[1|2]

(1) 機能

高速転送モードの開始

(2) プログラムメッセージ

```
READ[1|2]?
```

(3) レスポンスメッセージ

```
<level>
```

(4) パラメータ

```
<level>: <NR3>
```

(5) 解説

高速転送モードに切り替わり，現在の測定データを高速に返します。FETCh[1|2][:SCALar]:POWer[:DC]に比べて高速にデータ転送を行います。
測定データは絶対値で，単位はdBmです。
1度このコマンドを実行すると高速転送モードになります。高速転送モード中は，データ読み出しと高速転送モード終了コマンド以外のコマンドは無効になります。高速転送モードを終了する場合は，READ[1|2]:ABORtコマンド(第9.2.29項参照)を使用します。
また，高速転送モードは1つのチャンネルのみ有効です。サンプルプログラムを第10.2項の(2)に示します。

9.2.29 READ[1|2]:ABORt

(1) 機能

高速転送モードの終了

(2) プログラムメッセージ

```
READ[1|2]:ABORt
```

(3) 解説

高速転送モードを終了します。

9.3 光源

[1|2]は，制御する光源が挿入されているチャンネル番号を示します。チャンネル1の場合は，省略可能です。なお，カギカッコ([ ])は不要です。

例） `SOURCE1:POWER:STATE ON SOURCE2:POWER:STATE?  SOURCE:POWER:STATE 0` など

9.3.1 SOURce[1|2]:AM[:INTerval]:FREQuency

（1） 機能

変調周波数の設定

（2） プログラムメッセージ

```
SOURce[1|2]:AM[:INTerval]:FREQuency CW|<freq>[<unit>]
SOURce[1|2]:AM[:INTerval]:FREQuency?
```

（3） レスポンスメッセージ

```
SOURCE1|2:AM[:INTERVAL:FREQUENCY <freq>
```

（4） パラメータ

`<freq>:= {0,270,1000,2000}`（単位：Hz）
`<unit>:= {HZ,KHZ}`

（5） 解説

光出力をCWまたは<freq>で指定される変調周波数に設定します。
<freq>の単位が省略されたときはHz値，<unit>により指定されたときはその単位で設定します。
0 HzはCWが設定されます。
レスポンスメッセージは，常にHz値で出力します。

9.3.2 SOURce[1|2]:MEMory:COPY[:NAME]

（1） 機能

測定条件の保存／取り出し

（2） プログラムメッセージ

```
SOURce[1|2]:MEMory:COPY[:NAME] MC,<no> | <no>,MC
```

（3） パラメータ

```
<no>:= {0,1,2,3,4,5,6,7,8,9}
```

（4） 解説

<no>で指定されるメモリ番号で，測定条件の保存または読み出しをします。
“MC,<no>”は測定条件の保存，“<no>,MC”は測定条件の読み出しです。
なお，<no>に“0”を指定したときは，初期条件の設定なので読み出しのみ有効です。

9.3.3　SOURce[1|2]:POWer:ATTenuation

（1）機能

減衰量の設定

（2）プログラムメッセージ

```
SOURce[1|2]:POWer:ATTenuation <level>[DB]
SOURce[1|2]:POWer:ATTenuation?
```

（3）レスポンスメッセージ

```
SOURCE1|2:POWER:ATTENUATIOn <level>
```

（4）パラメータ

```
<level>:= {f(dB)|0.00 ≤ f ≤ 6.00}
```

（5）解説

光出力を<level>で指定された値だけ最大出力レベルより減少させます。
<level>の設定範囲および設定分解能はその光源ユニットに依存します。
<level>は設定分解能に丸められます。
減衰量は常にdB値として扱います。また単位は省略できます。

9.3.4　SOURce[1|2]:POWer:STATe

（1）機能

光出力の設定

（2）プログラムメッセージ

```
SOURce[1|2]:POWer:STATe <sw>
SOURce[1|2]:POWer:STATe?
```

（3）レスポンスメッセージ

```
SOURCE1|2:POWER:STATE <status>
```

（4）パラメータ

```
<sw>:= {ON,OFF,1,0}
<status>:= {1,0}
1 ..... ON
0 ..... OFF
```

（5）解説

光出力のON/OFFを設定します。

9.3.5 SOURce[1|2]:POWer:WAVelength

(1) 機能

波長の設定

(2) プログラムメッセージ

```
SOURce[1|2]:POWer:WAVelength UPPer|LOWer|CENTer|
<wavelength>[<unit>]
SOURce[1|2]:POWer:WAVelength?
```

(3) レスポンスメッセージ

```
SOURCE1|2:POWER:WAVELENGTH <wavelength>
```

(4) パラメータ

```
<wavelength>:= {f(m)|380 × 10-9 ≤ f ≤ 1800 × 10-9}
<wavelength>:= {f(Hz)|166.551 × 1012 ≤ f ≤ 788.927
× 1012}
<unit>:= {NM,UM,M,Hz}
```

(5) 解説

波長を<wavelength>に設定します。
波長の範囲および分解能は，その光源ユニットに依存します。実際の設定は分解能に丸められて設定されます。
プログラムメッセージで単位が省略されたときはm値として設定します。
単位が付加されたときはその単位で設定します。
レスポンスメッセージは現在設定されている単位系(mまたはHz)に従って出力します。
2波長光源であるときにのみ，パラメータに“UPPer”または“LOWer”が指定でき，それぞれ長波長側，短波長側の波長を設定します。
波長設定が“UPPer”，“LOWer”であってもレスポンスメッセージは長波長側の波長，短波長側の波長値を返します。
DFB-LD光源のときのみ，パラメータに“CENTer”を指定できます。このとき，波長を中心波長に設定します。

9.3.6 SOURce[1|2]:POWer:WAVelength:UNIT

(1) 機能

波長の表示単位

(2) プログラムメッセージ

```
SOURce[1|2]:POWer:WAVelength:UNIT<unit>
SOURce[1|2]:POWer:WAVelength:UNIT?
```

(3) レスポンスメッセージ

```
<unit>
```

(4) パラメータ

```
<unit>:= {M,Hz}
```

(5) 解説

波長の表示単位を切り替えます。

9.4　エラーメッセージ

(1) コマンドエラー[-100～-199]

エラーコード[-100～-199]は，IEEE488.2のシンタックスエラーが発生したことを示します。このとき，イベントステータスレジスタのビット5がセットされます。

エラーは，下記のイベントが起こった場合に発生します。

(a) IEEE488.2の標準に反したメッセージを装置が受け取った場合

(b) 装置固有コマンド，共通コマンドの規定に従わないヘッダを受け取った場合

(c) GET(Group Execute Trigger)がプログラムメッセージ内に送られた場合

コード	メッセージ	エラー検出条件
-101	Invalid character	ヘッダやパラメータに無効な文字が含まれる
-104	Data type error	パラメータのタイプが指定されたタイプと異なる
-105	Get not allowed	GET(Group Execute Trigger)がプログラムメッセージに送られた
-108	Parameter not allowed	パラメータの数が指定した数より多い
-112	Program mnemonic too long	プログラムニーモニックが12文字以上ある
-113	Undefined header	ヘッダの文法は正しいが,装置で定義されていない
-120	Numeric data error	数値データ内にエラーがある
-121	Invalid character in number	数値データ内に不適当な文字が含まれる
-130	Suffix error	サフィックス内にエラーがある
-144	Character data too long	キャラクタデータが12文字以上ある

(2) 実行時エラー[-200～-299]

エラーコード[-200～-299]は，装置の実行制御部でエラーが生じたことを示します。エラーが発生すると，イベントステータスレジスタのビット4がセットされます。

エラーは，下記のイベントが起こった場合に発生します。

(a) ヘッダに続く<PROGRAM DATA>が装置の規定外の場合

(b) プログラムメッセージが装置の状態により実行できない場合

コード	メッセージ	エラー検出条件
-220	Parameter error	パラメータ内にエラーがある
-221	Setting conflict	パラメータとしては正しいが,装置の状態により実行できない
-222	Data out of range	数値データが装置の規定外
-224	Illegal parameter value	受け取ったパラメータは使用不可能
-240	Hardware error	ハードウエア故障のため,コマンド実行不可

(3) デバイス固有エラー[-300～-399]

エラーコード[-300～-399]は，装置がコマンドエラー・問い合わせエラー・実行エラー以外のエラーが発生したことを示します。ハードウエア／ファームウエアの故障や自己診断エラーが含まれます。
装置エラーが発生すると，イベントステータスレジスタのビット3がセットされます。

コード	メッセージ	エラー検出条件
-310	System error	システムにエラーが起こった
-315	Configuration memory error	リジュームメモリが失われた
-350	Queue overflow	自己診断に異常があった

(4) 問い合わせエラー[-400～-499]

エラーコード[-400～-499]は，装置の出力キュー制御でメッセージ交換制御プロトコルに関するエラーが発生したことを示します。このエラーが発生すると，イベントステータスレジスタのビット2がセットされます。
エラーは，下記のイベントが起こった場合に発生します。

(a) 出力がないのに出力キューから読み込みを実行した場合
(b) 出力キューのデータが失われた場合

コード	メッセージ	エラー検出条件
-410	Query interrupted	装置が応答メッセージを送り終える前に,新たなコマンドによって割り込みが発生
-420	Query unterminated	読み込もうとする応答メッセージに対応した問い合わせが送られていない
-430	Query deadlocked	格納空きエリアを超えるデータのバッファリングを行おうとしている

第10章　プログラム作成例

この章では，リモート制御プログラムの作成について説明します。
ここでは，Visual BASICを使用して作成した例を示します。また，GPIBについてはナショナルインスツルメンツ社のハードウエアおよびNI-488.2Mソフトウエアの使用を前提にしています。
Visual BASICやNI-488.2Mの取り扱いについては，それぞれの取扱説明書を参照してください。

10.1 プログラム作成上の注意

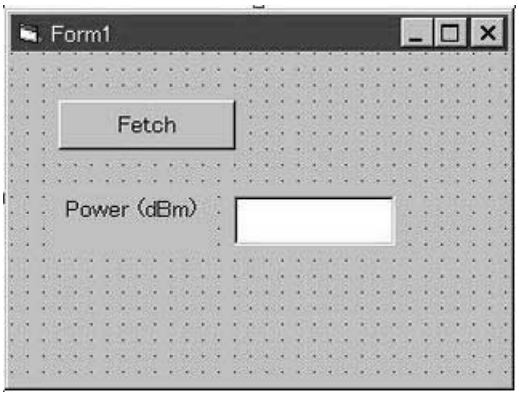
リモート制御プログラムを作成する場合，表10-1の留意事項に注意してください。

表10-1

No.	留意事項	説明
1	各デバイスの初期化を必ず行う。	各デバイスは,デバイス自身のパネル上の操作や,他のプログラムの実行などで,実際に使用する時点での状態が必ずしも適正でない場合が多いと考えられます。各デバイスの初期化を必ず行うことにより,一定の条件で使用を開始する必要があります。
2	問い合わせ直後に,結果読み込み以外のコマンドを送らない。	問い合わせ結果を読み込む前に読み込みコマンド以外のコマンドをコントローラに送ったとき,MLAが受信されると出力バッファがクリアされてしまうため,レスポンスメッセージが消失してしまいます。したがって,問い合わせ直後には必ず結果を読み込むコマンドを記述してください。
3	プロトコルの例外処理を避ける。	予想される例外については,プログラムに例外処理部を設けて,エラーによる実行停止を避けてください。
4	各デバイスのインタフェース機能（サブセット）の確認（GPIB）	必要なサブセットを用意しないデバイスに対してプログラムを実行しても処理は進みません。必ず各デバイスのサブセットを確認してください。また,IEEE488.2対応機種であることも確認してください。
5	バッファオーバーフローを防止する（RS-232C）	本器のRS-232Cインタフェースでは,内部の受信バッファとして256バイトのデータエリアを持っていますが,処理内容によってはバッファオーバーフローが発生する場合があります。オーバーフローによる不具合を防止するため,RS-232Cインタフェースを利用してリモート制御を行う場合は,一度に大量データ（制御コマンド）を送信しないようにしてください。一連のオマンドを送信した後,“OPC?”コマンドを送信し,レスポンスの受信を待って次のコマンドを送信して同期をとる方法もあります。

10.2 プログラム作成例

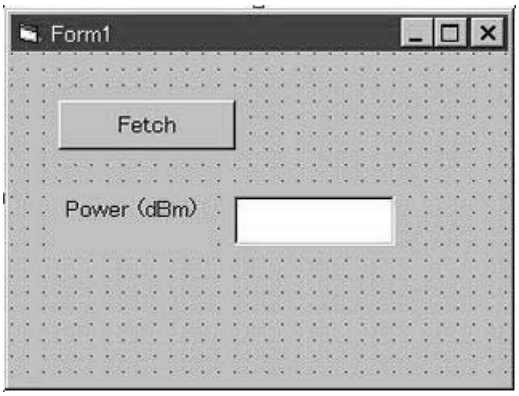
(1) 光センサユニットの測定データを読み出します。
本器のチャンネル1に光センサユニットを挿入し，外部光源の光パワーを測定します。
GPIBでその測定データを読み出し，表示します。
本器のGPIBアドレスは15です。




```
Sub cmdfetch_Click()                                                    *1
    Dim buf1 As String*20                                               *2
    Call Send(0,15,"SYSTEM:COMMUNICATE:GPIB:HEAD 0",NLend)              *3
    Call Send(0,15,"SENSE1:POWER:UNIT DBM",NLend)                       *4
    Call Send(0,15,"FETCH1:SCALAR:POWER:DC?",NLend)                     *5
    Call Receive(0,15,buf1,STOPend)                                     *6
    lblPwr.Caption=buf1                                                 *7
End Sub                                                                 *8
```

*3-*4　OTS初期設定
*5-*6　データ読み出し
*7　結果出力

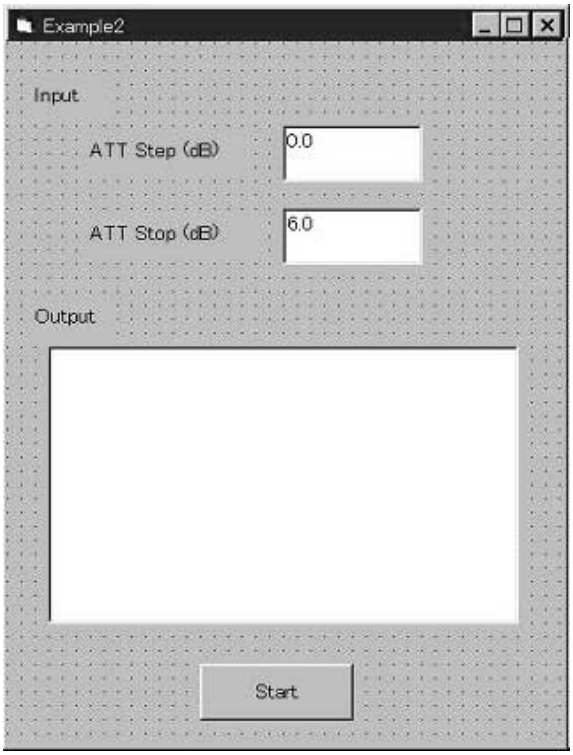
(2) 光センサユニットの測定データを読み出します。(高速転送モード)
本器のチャンネル1に光センサユニットを挿入し，外部光源の光パワーを測定します。
GPIBで1000回その測定データを読み出し，表示します。
本器のGPIBアドレスは15です。




```
Sub cmdfetch_Click()                                          *1
    Dim buf1 As String*20                                     *2
    Call Send(0,15,"READ1?",NLend)                            *3
    For I=0 To 1000                                           *4
        Call Receive(0,15,buf1,STOPend)                       *5
        lblpwr.Caption=buf1                                   *6
    Next I                                                    *7
    Call Send(0,15,"READ1:ABOR",NLend)                        *8
End Sub                                                       *9
```

*3　　高速転送モードへ切り替え
*4-*7　1000回データ読み出しと表示
*8　　高速転送モード終了

(3) 光源ユニットのアッテネータ値を光センサユニットで実測します。本器のチャンネル1に光源ユニットを，チャンネル2に光センサユニットをそれぞれ挿入し，光ファイバで接続します。光源ユニットのアッテネータ値を順次変化させたときの減衰量を光センサユニットで相対値測定し，結果を表示します。
本器のGPIBアドレスは15です。




```
Sub cmdstart_Click()                                                    *1
    Dim buf1 As String*15                                               *2
    Dim strAttStep As String*5                                          *3
    Dim strAttStop As String*5                                          *4
    Dim strAtt As String*5                                              *5
    Dim sglAttStep As Single                                            *6
    Dim sglAttStop As Single                                            *7
    Dim sglAtt As Single                                                *8
'
    chrAttStep=txtStep.Text                                             *9
    chrAttStop=txtStop.Text                                            *10
    sglAttStep=val(chrAttStep)                                         *11
    sglAttStop=val(chrAttStop)                                         *12
'
    Call Send(0,15,"SOURCE1:POWER:STATE 1",NLend)                      *13
    Call Send(0,15,"SOURCE1:POWER:ATTENUATION 0",NLend)                *14
```

```
    Call Send(0,15,"FETCH2:SCALAR:POWER:DC?",NLend)                  *15
    Call Receive(0,15,buf1,STOPend)                                  *16
    lblResult.Caption="ATT=0.0 dB   P0="+buf1                        *17
'
    Call Send(0,15,"SENSE2:POWER:REFERENCE:DISPLAY",NLend)           *18
    sglAtt=sglAttStep                                                *19
    Do                                                               *20
        chrAtt=str(sglAtt)                                           *21
        Call Send(0,15,"SOURCE1:POWER:ATTENUATION"+chrAtt,NLend)     *22
        Call Send(0,15,"FETCH2:SCALAR:POWER:DC?",NLend)              *23
        Call Receive(0,15,buf1,STOPend)                              *24
        lblResult.Caption=lblResult.Caption+chr(13)                  *25
        lblResult.Caption=lblResult.Caption+"ATT="+chrAtt+"dB  Pr="+buf1   *26
        sglAtt=sglAtt + sglAttStep                                   *27
        If sglAtt > sglAttStop Then                                  *28
            Exit do                                                  *29
        End If                                                       *30
    Loop                                                             *31
End Sub                                                              *32
```

*13　　光源ユニットの光出力ON
*14　　光源ユニットの減衰量を0 dBに設定
*15-*16　光センサユニットでパワー測定
*17　　測定結果の表示
*18　　光センサユニットを相対値測定モードに設定
*22　　光源ユニットの減衰量を設定
*23-*24　光センサユニットでパワー測定
*25-*26　測定結果の表示
*28-*30　繰り返し条件の判定

注：

実際の測定では，光源ユニットの出力安定のため，*14と*15，および*22と*23の間に5秒程の待ち時間を入れます。

第11章　LabVIEW計測器ドライバ

この章では，LabVIEW上で本器をリモート制御するための計測器ドライバ（MX981001A）について説明します。
LabVIEW計測器ドライバとは，米国ナショナルインスツルメンツ社のグラフィックプログラミングシステム"LabVIEW"上で計測器を制御する際に，コマンドの送信・受信部を機能ごとににまとめ，モジュール化したものです。本計測器ドライバを用いることにより，制御コマンドを憶えておかなくても，本器をリモート制御することが可能です。
本計測器ドライバを使用するには，ナショナルインスツルメンツ社のWindows版LabVIEWソフトウエアがインストールされているコントローラが必要となります。
本計測器ドライバはWindows版LabVIEWバージョン4.1で作成しました。
LabVIEWの操作方法については，LabVIEWの説明書を参照してください。
LabVIEWは，米国ナショナルインスツルメンツ社の登録商標です。
Windowsは米国マイクロソフト社の登録商標です。

LabVIEWについて

LabVIEW は計測器の制御，データの収録・解析に適したグラフィカルプログラム言語です。
LabVIEW では回路図を描くようにしてプログラムを作成しますので，テキストベースのプログラミング言語に比べなじみやすいものになっています。また，実効速度もC言語とほぼ同等です。
LabVIEW には計測器の制御とデータの収録・解析・表示に関するライブラリが豊富に用意されています。これと計測器ドライバを用いることにより，グラフィカルユーザインタフェース(GUI)プログラムを容易に作成することができます。

11.1 インストール

添付されているフロッピーディスクMX981001Aには，下記のファイルが収納されています。

```
MT9810.LLB
```

インストール例

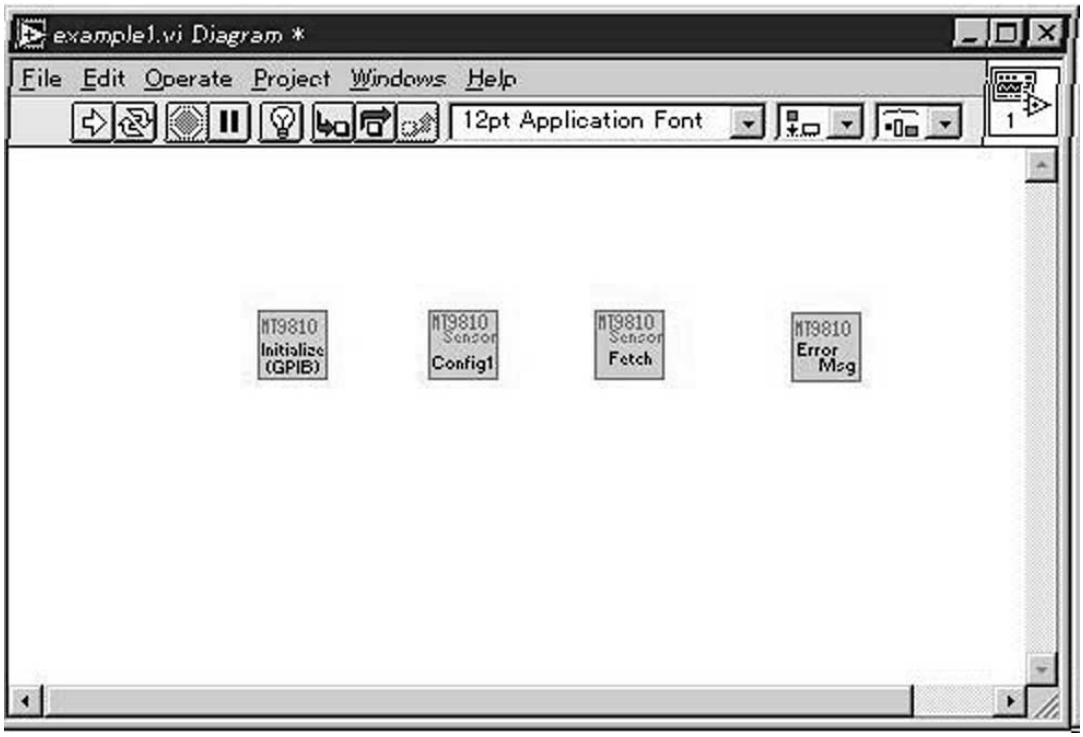
(1) X:LABVIEW(“X”はLabVIEWがインストールされているドライブ名を示します)上に，ディレクトリMT9810.LIBを作成します。
(2) このディレクトリに，MT9810.LLBをコピーします。

11.2 プログラム例

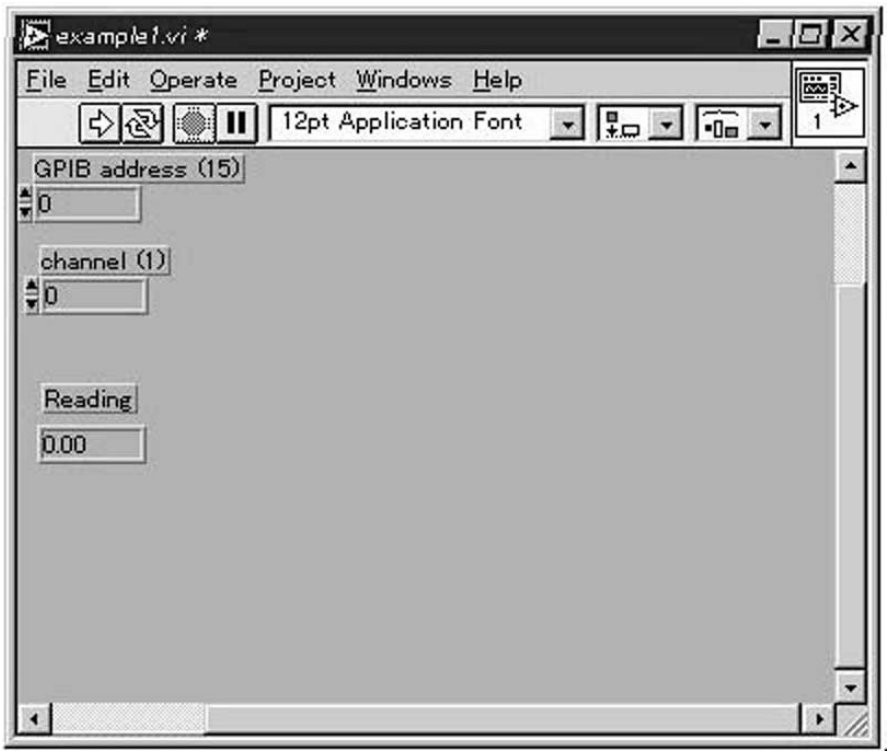
本計測器ドライバを使ったプログラム例を示します。
ここでは，「第10.2項　プログラム例」と同様に，GPIB制御で光パワー測定を行うプログラムを作成します。このプログラム例では，本器のGPIBアドレスを15番にしています。
ここでは，下記の四つのドライバを使用します。

MT9810 Initialize(GPIB).vi	GPIBによる通信準備
MT9810 Config.Sensor.vi	光センサユニットの設定
MT9810 Sensor.Fetch.vi	光センサユニットから測定データ読み込み
MT9810 Error.message.vi	エラーメッセージ表示

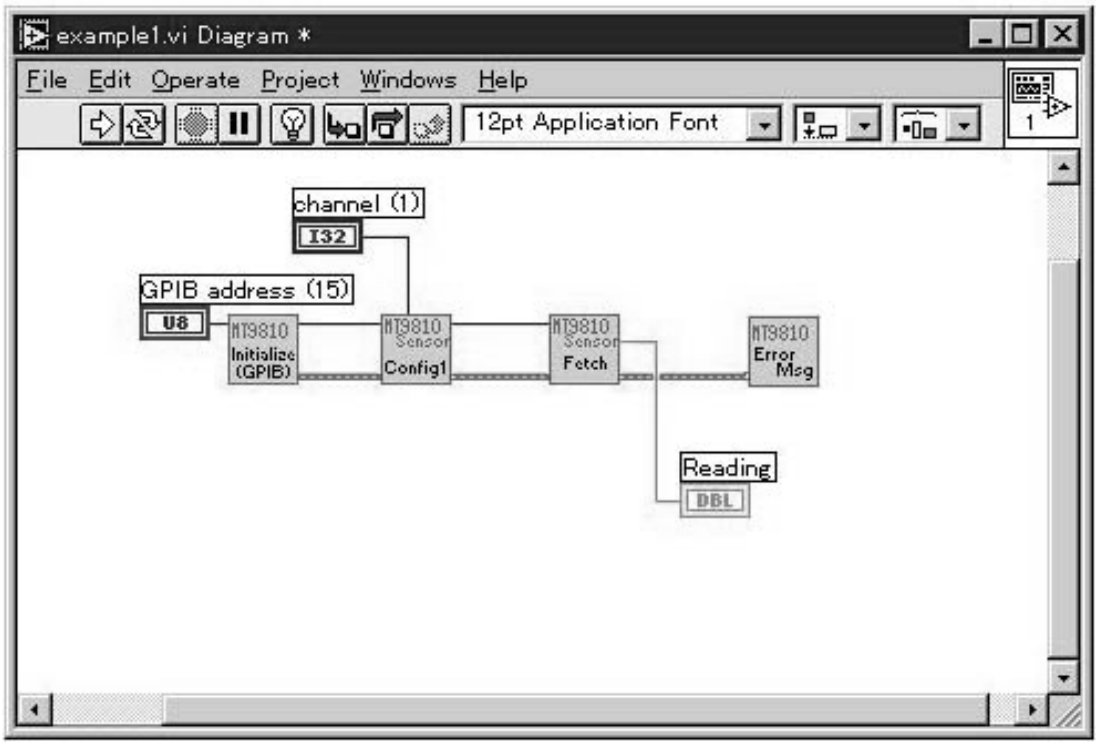
(1) ブロックダイヤグラム上にドライバを配置
上記のドライバを順番に配置します。



（2）フロントパネルウインドウ上に制御器・表示器を配置
ダイヤグラムウインドウ上のMT9810 Initialize(GPIB).viのアイコンをダブルクリックすると，計測器ドライバのウインドウが開きます。このウインドウの中から，GPIBアドレスを入力する制御器をフロントパネル上にコピーします。
同様にして，MT9810 Sensor.Fetch.viのアイコンから測定データを表示する表示器をフロントパネル上にコピーします。



（3） 各表示器・制御器とアイコンの端子を接続
図のように，各ドライバ間をワイヤで接続します。



11.3 計測器ドライバー覧

計測器ドライバVIのファイル名は，MT9810（機能名称）.viとなっています。
また，GPIBとRS-232Cは準備（Initialize）以外共通です。

表11-1　サンプル／ユーティリティ

ファイル名	機能
MT9810 VI tree.vi	すべての計測器ドライバをロード
MT9810 Example1.vi	簡単なプログラム例
MT9810 Example2.vi	簡単なプログラム例
MT9810 Example3.vi	簡単なプログラム例
MT9810 Interactive.vi	デバイスメッセージレベルで通信
MT9810 Error Message.vi	エラーコードと詳細情報

表11-2　本体

ファイル名	機能
MT9810 Initialize(GPIB).vi	GPIBの準備
MT9810 Initialize(RS232C).vi	RS-232Cの準備
MT9810 Reset.vi	本体のリセット
MT9810 Self-Test.vi	内部セルフテスト
MT9810 Config Instrument.vi	本体のパラメータ設定／問い合わせ

表11-3　光センサユニット

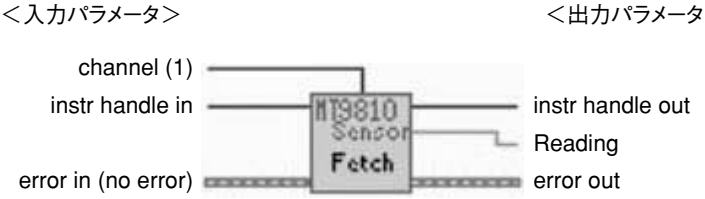
ファイル名	機能
MT9810 Config Sensor Zeroing.vi	ゼロセット
MT9810 Config Sensor_1.vi	パラメータの設定／問い合わせ
MT9810 Config Sensor_2.vi	パラメータの設定／問い合わせ
MT9810 Config Sensor Wavelength.vi	測定波長の設定／問い合わせ
MT9810 Config Sensor Ranging.vi	測定レンジの設定／問い合わせ
MT9810 Config Sensor Reference.vi	リファレンス測定の設定／問い合わせと実行／停止
MT9810 Sensor Fetch.vi	測定データの問い合わせ
MT9810 Config Logging Parameter.vi	記録測定の実行／停止
MT9810 Read Logging Values.vi	記録測定による測定データを出力
MT9810 MinMax Values.vi	最大値／最小値の測定データをリセットするか出力

表11-4　光源ユニット

ファイル名	機能
MT9810 Config Source.vi	パラメータの設定／問い合わせ
MT9810 Config Source Output.vi	光出力のON/OFF

11.4 計測器ドライバの機能説明

計測器ドライバの機能と入出力パラメータについて説明します。
計測器ドライバは，アイコンの左側の端子からデータと設定値を受け取り，その入力パラメータの値により各ドライバで決められた処理を行います。処理終了後，アイコン右側の端子から処理結果を出力します。



MT9810 Sensor Fetch.vi

本章のパラメータの説明で，変数名の後の[　]は変数の型を示します。

11.4.1 共通パラメータ

計測器ドライバのほとんどで使用される入出力パラメータについて説明します。

instr handle in [I32]
GPIBアドレスやシリアルのポート番号，通信パラメータの設定を格納するMT9810 Global（グローバル変数，クラスタの1次配列）の指示子です。Initialize.viにはありません。

instr handle out [I32]
Instl handle inの値を出力します。Close.viにはありません。

error in [clust]
VIを実行させる前のエラーの発生状況を出力します。
status [bool] ..... エラー有無を示します。“True”のとき，エラーの発生を示します。
code [I32] ..... エラーが発生したとき（statusがTrueになったとき）の，エラーコードを示します。
source [str] ..... エラーの発生したVIを示します。

error out [clust]
VIを実行させた後のエラーの発生状況を出力します。クラスタの内容はerror inと同じです。

Channel [I32]
ユニットのチャンネル番号を示します（ユニットに対するVIのみ）。

11.4.2 機能説明

(1) サンプル／ユーティリティVI

MT9810 VI tree.vi

このVIのダイヤグラム上にすべての計測器ドライバをロードしたものです。(ただし，SubVIは含みません)一覧として利用できます。

MT9810 Example1.vi

計測器ドライバを使った簡単なプログラム例です。
センサユニットの表示値をMeasurement Intervalで設定した間隔で取り込み，Reading表示器，チャートに表示します。sensor channnel，GPIB／RS-232制御器を設定・実行します。(GPIB，RS-232Cのパラメータはウインドウの右端に隠れて配置されています)clearボタンによりチャートをクリアします。終了するときはExitボタンを押します。

MT9810 Example2.vi

計測器ドライバを使った簡単なプログラム例です。
リファレンス測定の様子を表示します。(センサユニットは二つ必要)GPIB／RS-232制御器を設定・実行します。(GPIB，RS-232Cのパラメータはウインドウの左端に隠れて配置されています)sensor channnel，Absolute Unit，Reference State，Level Valueを適当に変えてfetchボタンを押すと，チャンネル1と2の測定値とリファレンス測定値が表示されます。終了するときはExitボタンを押します。

MT9810 Example3.vi

計測器ドライバを使った簡単なプログラム例です。
最大・最小値測定の値を表示します。sensor channnel，GPIB／RS-232制御器を設定・実行します。(GPIB，RS-232Cのパラメータはウインドウの右端に隠れて配置されています)最大値，最小値，最大値と最小値の差，経過時間が表示されます。Resetボタンにより最大値，最小値の値はリセットされます。終了するときはExitボタンを押します。

MT9810 Interactive.vi

本器との通信をデバイスメッセージレベルで行います。GPIB／RS-232Cを設定し，Write Buffer1～4のどれかに本器に送信するデバイスメッセージを入力します。スイッチで実際に送信するデバイスメッセージのWiteBufferの番号を指定し，実行します。クエリコマンドを送信した場合はRead Bufferにレスポンスメッセージが表示されます。

MT9810 Error Message.vi

エラーコードとそれらの詳細情報の報告を行います。いくつかの計測器ドライバVIを実行した後，これを実行してエラー情報を確認します。

パラメータの説明

- type of dialog [int] ..... エラーが発生したときに表示するダイアログのスタイルを選択します。
  0: ダイアログを表示しません。
  1: OKボタンダイアログ
  2: 継続・停止ボタンダイアログ
- status [bool] ..... エラーが発生した場合，TRUEになります。
- code [int] ..... エラーに該当するエラーコードを出力します
  0の場合，エラーがないことを示します。
  負の場合，エラーが発生したことを示します。
  正の場合，警告を示します。
- error message [str] ..... 発生したエラーの説明を出力します。

（2）本体関係のVI

MT9810 Initialize (GPIB).vi

MT9810 Initialize (GPIB)

GPIBによる計測器との通信を開始するための準備を行います。

1. デバイスクリアを送信。
2. 本体のIDの確認。（選択可）
3. リセット（レベル3）の実行。（選択可）
4. レスポンスメッセージのヘッダをOFFに設定。

パラメータの説明

- GPIB address [V8] ..... GPIBアドレス
- Reset [bool] ..... リセット動作を行うかどうかの切り替えをします。
- ID Query [bool] ..... IDの照合を行うかどうかの切り替えをします。

MT9810 Initialize(RS232C).vi

MT9810 Initialize (RS232)

RS-232Cによる計測器との通信を開始するための準備を行います。

1. シリアルポートのパラメータを設定。
2. 本体のIDの確認。（選択可）
3. リセット（レベル3）の実行。（選択可）
4. レスポンスメッセージのヘッダをOFFに設定。

パラメータの説明

- RS232C Parameter[clust] ..... シリアルポートの設定値です。
  Port No. (0:COM1) [V8] ..... シリアルポート番号
  baud rate (bps) [V16] ..... ボーレート
  stop bit [V16] ..... ストップビット
  parity bit [V16] ..... パリティビット
  character (bit) [V16] ..... キャラクタ長
- Reset [bool] ..... リセット動作を行うかどうかの切り替えをします。
- ID Query [bool] ..... IDの照合を行うかどうかの切り替えをします。

MT9810 Reset.vi

MT9810
Reset

本体のリセットを行います。

パラメータの説明
（なし）

MT9810 Self-Test.vi

MT9810
Self-Test

内部セルフテストを実行し，エラーの有無を返します。error outにテスト結果は出力しません。

パラメータの説明

- Self-test Error [bool] ... テストの結果エラーの場合はTrueとなります。

MT9810 Config Instrment.vi

MT9810
config
instr

本体のパラメータ（表示のON/OFF，輝度，日付，時刻，ブザーのレベル）を設定／問い合わせを行います。

パラメータの説明

- Display Brightness [I32] ..... 管面の輝度の設定をします。
- Set Date [clust] ..... 日付の設定をします。設定する場合は，Year，Month，Dayすべてに値を入力してください。

  Year [I32]
  Month [I32]
  Day [I32]
- Set Time [clust] ..... 時刻の設定をします。設定する場合は，Hours，Minutes，Secondsすべてに値を入力をしてください。

  Hours [I32]
  Minutes [I32]
  Seconds [I32]
- Beep Level [I32] ..... ブザー音のレベル設定をします。

(3) 光センサユニット関係のVI

MT9810 Config Sensor Zeroing.vi

ゼロセットを実行し，正常終了かエラーかの出力を行います。ゼロセット処理終了，またはエラー発生の後，VIを終了します。エラーはerror outクラスタに出力されます。

パラメータの説明

(なし)

MT9810 Config Sensor_1.vi

パラメータの設定／問い合わせ(表示単位系・校正係数・変調周波数)を行います。

パラメータの説明

- Absolute Units [I32] ..... 表示単位系の切り替えをします。
- Calibration Factor [double] .... 校正係数です。
- Modulation Frequency [I32] ... 変調周波数の設定値です。

MT9810 Config Sensor_2.vi

パラメータの設定／問い合わせ(測定インターバル・帯域・平均化回数)を行います。

パラメータの説明

- Measurement Interval [double] ..... 測定間隔の設定値です。分解能に丸められます。
- Bandwidth [double] ..... 帯域の設定値です。
- Averaging Time [I32] ..... 平均化回数の設定値です。

MT9810 Config Sensor Wavelength.vi

パラメータの設定／問い合わせ(測定波長)を行います。

パラメータの説明

- Wavelength Value [double] .... 波長の設定値です。波長の設定範囲および分解能は，センサユニットに依存します。単位はWavelength Unitで指定した単位となります。
- Wavelength Units [bool] ..... 波長の単位の切り替えをします。

MT9810 Config Sensor Ranging.vi

測定レンジを設定／問い合わせを行います。

パラメータの説明

- Power Range [I32] ..... レンジ範囲の設定値です。センサユニットによって設定できない値もあります。

MT9810 Config Sensor Reference.vi

リファレンス測定の設定／問い合わせ(リファレンス測定方式，リファレンス値)と実行／停止を行います。

パラメータの説明

- Reference State [I32] .. リファレンス測定方式の切り替えをします。“Reference to the other”は(測定値)－(他チャンネルの測定値)－(リファレンス値)，“Reference to Value”は(測定値)－(リファレンス値)となります。
- Level Value [double] .. リファレンス値です。設定範囲は，リファレンス測定方式により異なります。設定値の単位は表示単位系に依存します。“Reference to the other”のときは－199.999～199.999 dB，“Reference to Value”のときは 1E-16～99.999 Wまたは－199.999～199.999 dBmで指定します。

MT9810 Sensor Fetch.vi

測定データの問い合わせを行います。データの単位は表示単位系やリファレンスの設定により“dBm”，“W”，“dB”，“%”となります。

パラメータの説明

- Reading [double] ..... 測定値です。

MT9810 Config Logging Parameters.vi

MT9810 Sensor Logging Read

記録測定の実行／停止を行います。実行のときはNumber of Samples（測定データ数）の設定を行います。

パラメータの説明

- Start/Stop [bool] ..... 記録測定の実行／停止の切り替えをします。
- Number of Samples [I32] ..... 測定データ数の設定値です。

MT9810 Read Logging Values.vi

MT9810 Sensor Logging Read

記録測定による測定データの出力を行います。

パラメータの説明

- Number of Samples [I32] ..... 測定データ数の設定値です。
- Number of Samples taken [I32] .... 読み出したデータの個数です。
- Result Array [double array] ..... 測定データ（ログ値）です。
- Data [str] ..... データの識別子，ユニット形名，測定日時，アベレージ回数，インターバル時間，測定データ数，データ統計（最大値“dBm”，最小値“dBm”，peak-to-peak値“dB”，平均値“dBm”）を下記のように出力します。

```
V1.0,"XXXXXXXXX;YY/MM/DD,hh:mm:ss;XXX;XXX;XXX;
XXXXXX,XXX,XXX"
```

MT9810 MinMax Values.vi

最大値／最小値の測定データをリセットするか，最大値／最小値の測定データの出力を行います。

パラメータの説明

- Reset [bool] ..... 最大値／最小値のリセット動作を行うかどうかの切り替えをします。
- Minimum [double] ..... 最小値測定データです。(dBm，W) リセット動作を行った場合は問い合わせしません。
- Maximam [double] ..... 最大値測定データです。(dBm，W) リセット動作を行った場合は問い合わせしません。
- Change in Power Level [double] ... 最小値と最大値の差です。(dB，W) リセット動作を行った場合は問い合わせしません。

(4) 光源ユニット関係のVI

MT9810 Config Source.vi

パラメータ(変調周波数，減衰量，波長の選択・設定)の設定／問い合わせを行います。

パラメータの説明

- Frequency [I32] ..... 変調周波数の設定値です。
- Attenuation Level [double] ..... 減衰量の設定設定値です。設定範囲および設定分解能は光源ユニットに依存します。
- Wavelength Level [double] .... 波長の設定値です。2波長光源のときは，長波長側・短波長側の選択を行います。波長の範囲および分解能は，光源ユニットに依存します。

MT9810 Config Source Output.vi

光出力のON/OFFの切り替えを行います。

パラメータの説明

- Source Output Signal State [bool] ..... 光出力のON/OFFの切り替えをします。

Anritsu

MT9810B

光テストセット
リモート制御

取扱説明書

■ 製品をご使用前に必ず本書をお読みください。本書は製品とともに保管してください。

Anritsu

MT9810B　光テストセット　リモート制御　取扱説明書

Anritsu

アンリツ株式会社　東京都港区南麻布5 - 10 - 27 〒106 - 8570　TEL 03 - 3446 - 1111

管理番号： M-W1887AW
Printed in Japan