MP1761C

パルスパターン発生器

取 扱 説 明 書

(GPIBプログラミング)

初版

安全にお使い頂くための重要事項は、MP1761Cパルスパターン発生器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

管理番号: M-W1845AW-1.0

MP1761C パルスパターン発生器
取扱説明書 (GPIB プログラミング)

2001年 (平成13年) 4 月 (初版)

「HP Basic」はHewlett-Packard Corporationの登録商標です。

「HP」はHewlett-Packard Companyの登録商標です。

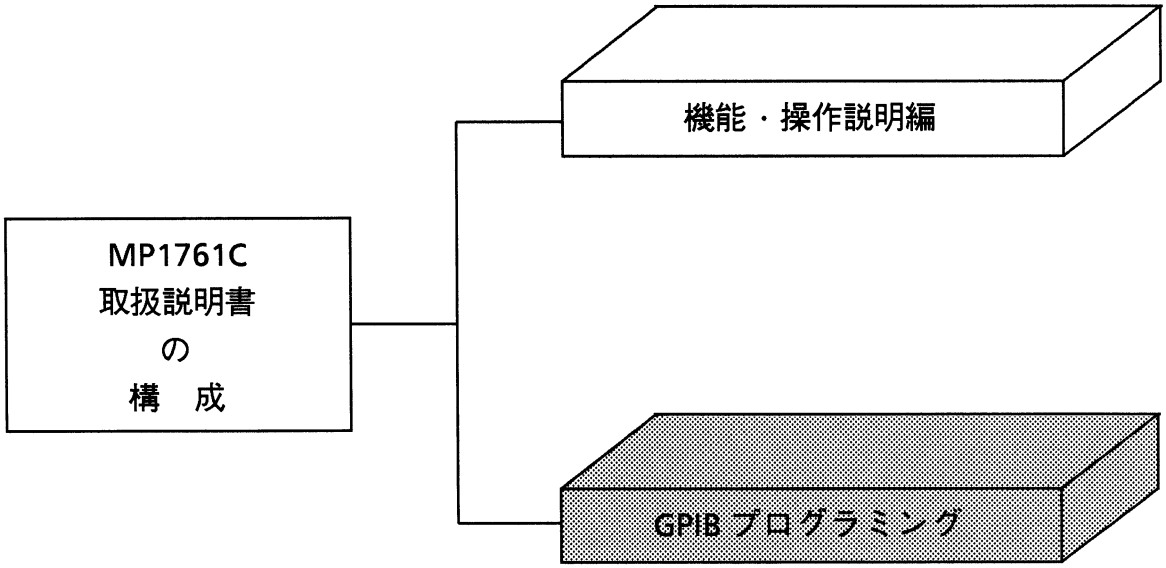
「MS-DOS」はMicrosoft Corporationの登録商標です。

「Quick Basic」はマイクロソフト株式会社の登録商標です。

（空白）

MP1761C取扱説明書の構成

MP1761Cパルスパターン発生器の取扱説明書は、下記の2冊で構成されています。利用目的に合わせて使い分けてください。



機能・操作説明編　：MP1761Cの概要、使用前の準備、パネル説明、規格、性能、機能および操作法を解説してあります。

GPIBプログラミング　：MP1761CはIEEE488.2対応機種ですのでIEEE488.2にもとづいたGPIBリモート制御について解説されています。プログラム例は、当社製PACKET Vシリーズ・パーソナルコンピュータ、HP9000シリーズ用HP-BASIC、マイクロソフト社製Quick Basicを適用しています。

（空白）

目　次

1 章
概　　要

この章では、GPIB 規格変遷の概要および MP1761C パルスパターン発生器の GPIB 機能の概要について説明します。

目　　次

（空白）

1章 概　要

GPIB規格変遷

MP1761C パルスパターン発生器は、外部コントローラと組み合わせて、システムバスを構築し、測定の自動化を提供します。このために使用する計測バスとして、本器は、GPIBインタフェースバス(IEEE std 488.2-1987)を第1インタフェースとして標準装備しています。

GPIB (General Purpose Interface Bus)は、1975年IEEE (Institute of Electric and Electronics Engineers米国電気電子学会)で制定されたプログラマブル計測機器用標準ディジタルインタフェースバス規格で、そのオリジナルは1975年に、IEEE Std 488-1975の名で発表されました。

このIEEE Std 488-1975は、ドキュメント上の編集・改訂を経て、1978年にIEEE Std 488-1978として発行されました。このIEEE Std 488-1978は、インタフェース側の仕様を定めたハードウェア規格のみであったため、1982年に、デバイス側の仕様を定めたソフトウェア規格として、IEEE Std 728-1982が加えられました。

このIEEE Std 728-1982は、デバイスメッセージの転送フォーマットを規格化したものですが、利用者側にたったソフトウェア共有化のコンセプトに欠けるところがありました。そこで、このコンセプト実現のためIEEE Std 728-1982の改訂版として1987年に、IEEE Std 488.2-1987 (これ以降、IEEE 488.2と記す)が登場することとなり、メッセージ交換プロトコル、メッセージデータコード、デバイス入出力フォーマット、共通コマンドの標準化が強化されました。

このIEEE 488.2の登場により、これまで運用してきたIEEE Std 488-1978(これ以降、IEEE 488と記す)は、IEEE Std 488.1-1987 (これ以降、IEEE 488.1と記す)と改名されることになりました。以上述べてきたGPIB規格の変遷を整理すると次のようになります。

規格対象	旧規格	新規格	備　　考
ハードウェア	IEEE 488	IEEE 488.1	IEEE 488.1はIEEE 488と同じ
ソフトウェア	IEEE 728	IEEE 488.2	IEEE 488.2はIEEE 728の改訂版

IEEE 488.2をサポートする機器は、IEEE 488.1とも互換性があることが必須条件ですが、IEEE 488.1 (IEEE 488)をサポートする機器は、IEEE 488.2への適合を保証されていません。

MP1761C GPIB 機能

MP1761C には、次のようなGPIB機能があります。

(1) 電源スイッチ およびLOCALキー等の一部を除くすべての機能の制御

(2) すべての設定条件の読み出し

(3) 割込み機能とシリアルポール動作

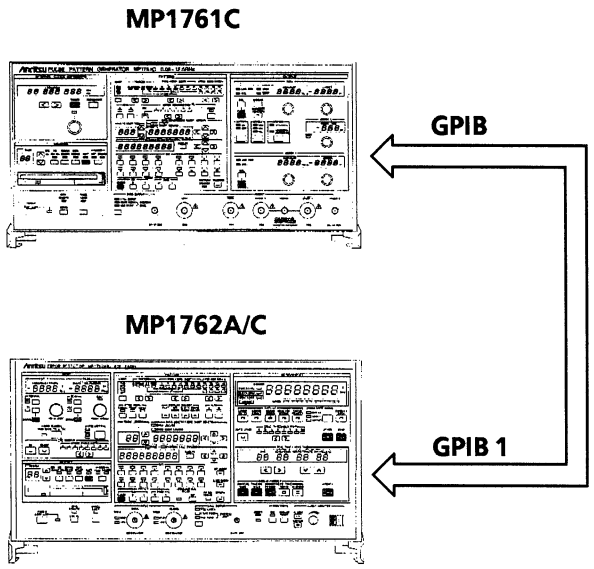
(4) パーソナルコンピュータやその他の測定器と組み合せて自動計測システムを構成できます。

GPIB 機能概要

GPIB ポートは従来の1 ポート GPIB を備えた測定器と同様に扱うことができます。したがってパワー ON 時や通常の測定状態ではデバイスポートとしての機能を行い、またトラッキング動作時においてはシステムコントローラの設定を行うことにより、MP1762A/C 誤り検出器を制御するためのシステムコントローラポートとして機能を行います。

GPIB を利用したシステムアップ例

(1) スタンドアロン方式……トラッキング動作

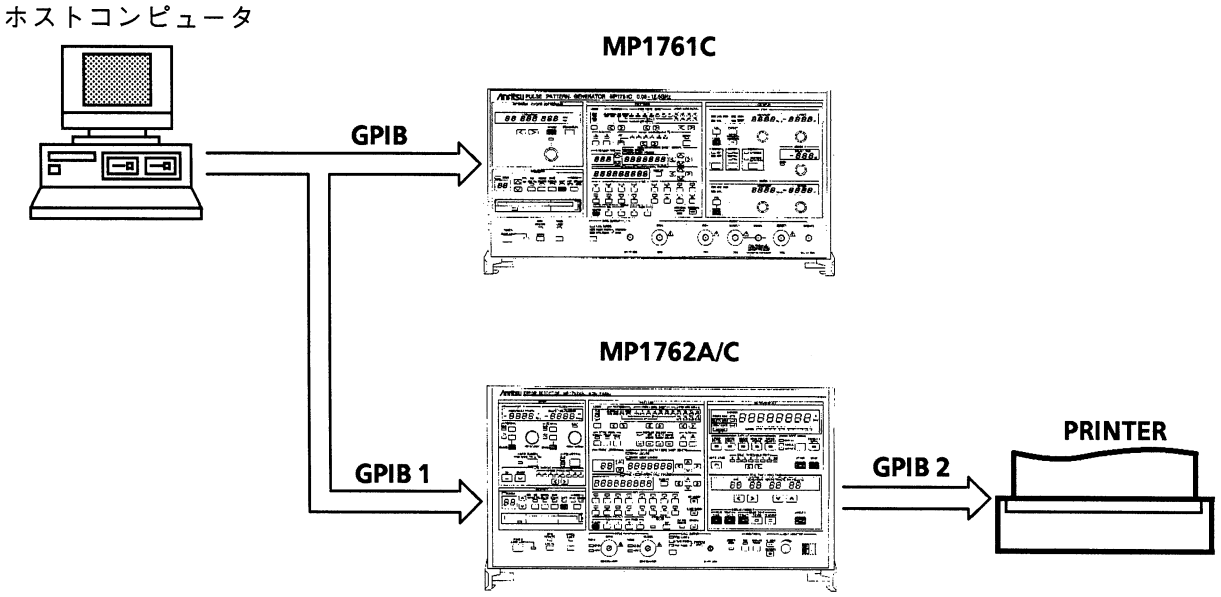


① 送信部の設定の一部が受信部の設定に連動します。このトラッキング動作時は、外部コントローラを接続できません。

② 受信部の設定の一部が送信部の設定に連動します。このトラッキング動作時は、外部コントローラを接続できません。

※ 本トラッキング動作でマスター(コントローラ)となれるのは、**MP1761C**もしくは**MP1762A/C**のどちらか一方のみです。

(2) ホストコンピュータ制御



ホストコンピュータより**GPIB 1**ポートを介して、**MP1761C**と**MP1762A/C**を制御します。このとき、**MP1762A/C GPIB 2**ポートよりプリンタへデータを出力します。

（空白）

2 章
規　　格

この章では、MP1761C の GPIB 規格としてインタフェースファンクション、およびデバイスメッセージリストについて説明します。

目　　次

（空白）

2章 規　格

インタフェースファンクション

IEEE 488.2では、IEEE 488.1で規定されているGPIBインタフェース機能の中から各測定器が備えるべき必要最小限のサブセットを決め、すくなくともシステム用として使えるレベルにしています。MP1761CのGPIBは、下表のコード欄に示すサブセットを備えています。

GPIBインタフェースファンクション

コード	インタフェースファンクション	IEEE 488.2 規定
SH1	ソースハンドシェイクの全機能有り。 データ送信のタイミングをとります。	全機能標準装備。デバイスは完全なソースハンドシェイク機能を有すること。
AH1	アクセプタハンドシェイクの全機能有り。 データ受信のタイミングをとります。	全機能標準装備。デバイスは完全なアクセプタ・ハンドシェイク機能を有すること。
T6	基本的トーカ機能有り。 シリアルポール機能有り。 トークオンリ機能無し。 MLAによるトーカ解除機能有り。	デバイスは、T5, T6, TE5, TE6いずれかのサブセットを有すること。トークオンリ機能はIEEE 488.2規定の規定外とする。
L4	基本的リスナ機能有り。 リスンオンリ機能無し。 MTAによるリスナ解除機能有り。	デバイスは、L3, L4, LE3, LE4いずれかのサブセットを有すること。リスンオンリ機能はIEEE 488.2規定の規定外とする。
SR1	サービスリクエスト、ステータスバイトの全機能有り。	全機能標準装備
RL1	リモート/ローカルの全機能有り。 ローカルロックアウトの機能有り。	RL0 (機能無し)またはRL1 (全機能)
PP0	パラレルポール機能無し。	PP0 (機能無し)またはPP1 (全機能)
DC1	デバイスクリアの全機能有り。	全機能標準装備。
DT1	デバイストリガ機能有り。	DT0 (機能無し)またはDT1 (全機能)
C1,C2, C3,C4, C7	コントローラ機能有り。 トラッキング動作時のみコントローラにすることが可能。	C0 (機能無し) もしくはC4とC5、または C7, C9, C11 のいずれか

デバイスメッセージリスト

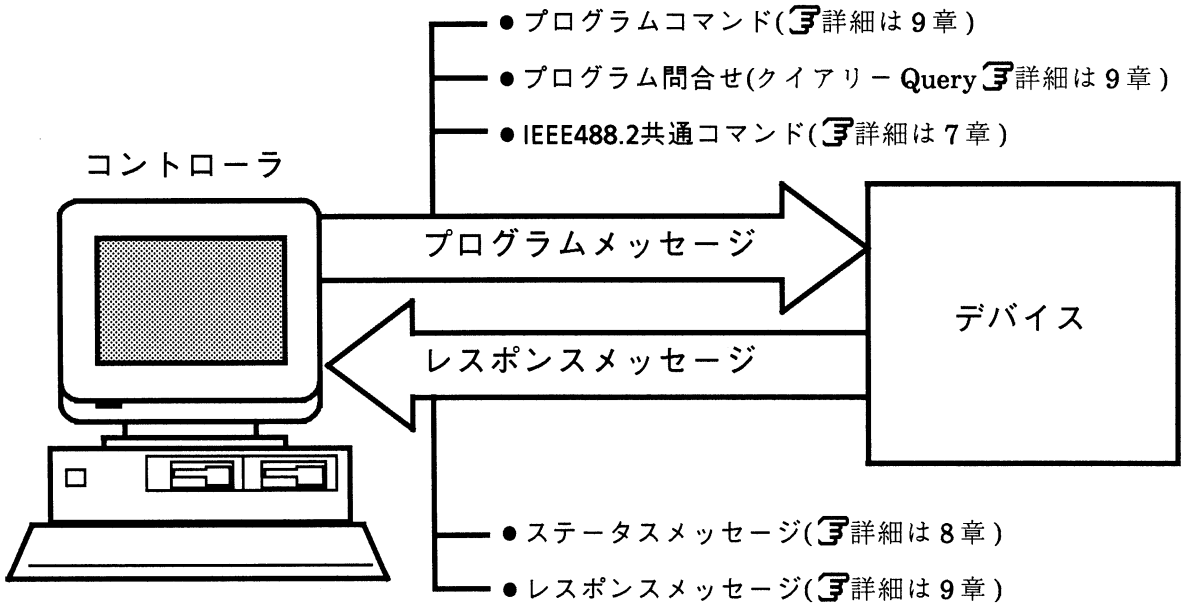
デバイスメッセージは、バスのモードすなわちATNラインが偽の時、システムインタフェースをとおしてコントローラとデバイス間で送受されるデータメッセージで、プログラムメッセージとレスポンスメッセージの二つがあります。

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(コマンドー command)およびプログラム問合せ(クイアリー 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、MP1761C制御専用に使用される装置固有のコマンドの他に、IEEE 488.2共通コマンドがあります。IEEE 488.2共通コマンドは、MP1761Cを含み、バス上に接続されたその他のIEEE 488.2対応測定器にも共通に適用されるプログラム命令です。

プログラム問合せは、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合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。



以上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言います。入力バッファ、出力バッファについて簡単に説明しておきます。

入力バッファ	出力キュー
DAB (プログラムメッセージや問合せメッセージ)の構文を解析し、実行する前に、それらのメッセージを一次的に蓄えておくFIFO (First in First out) タイプのメモリエリアです。 MP1761Cの入力バッファサイズは256バイトあります。	FIFOタイプの待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。 MP1761Cの出力キューサイズは256バイトあります。

IEEE 488.2 共通コマンドとサポート対象コマンド

下表に IEEE 488.2 規格で定められている 39 種類の共通コマンドをリストします。この中から MP1761C で使用される IEEE 488.2 共通コマンドを◎印で示します。

ニーモニック	コマンド・フルスペル名	IEEE488.2規定	当社サポートコマンド (MP1761C)
*AAD	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	◎

ニーモニック	コマンド・フルスペル名	IEEE488.2規定	当社サポートコマンド (MP1761C)
*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	◎

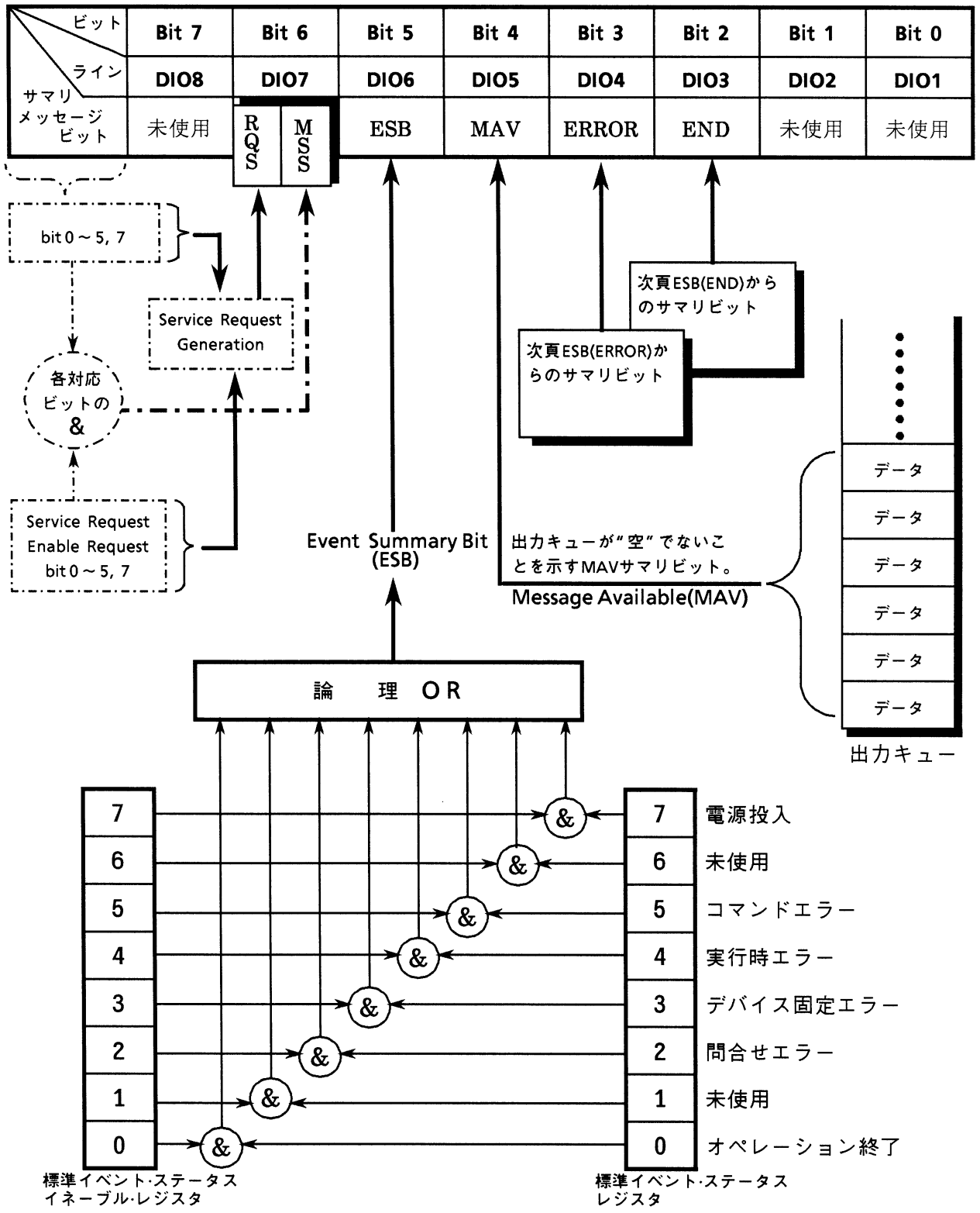
☞ IEEE488.2共通コマンドは、必ず*で始まります。詳細については、7章を参照してください。

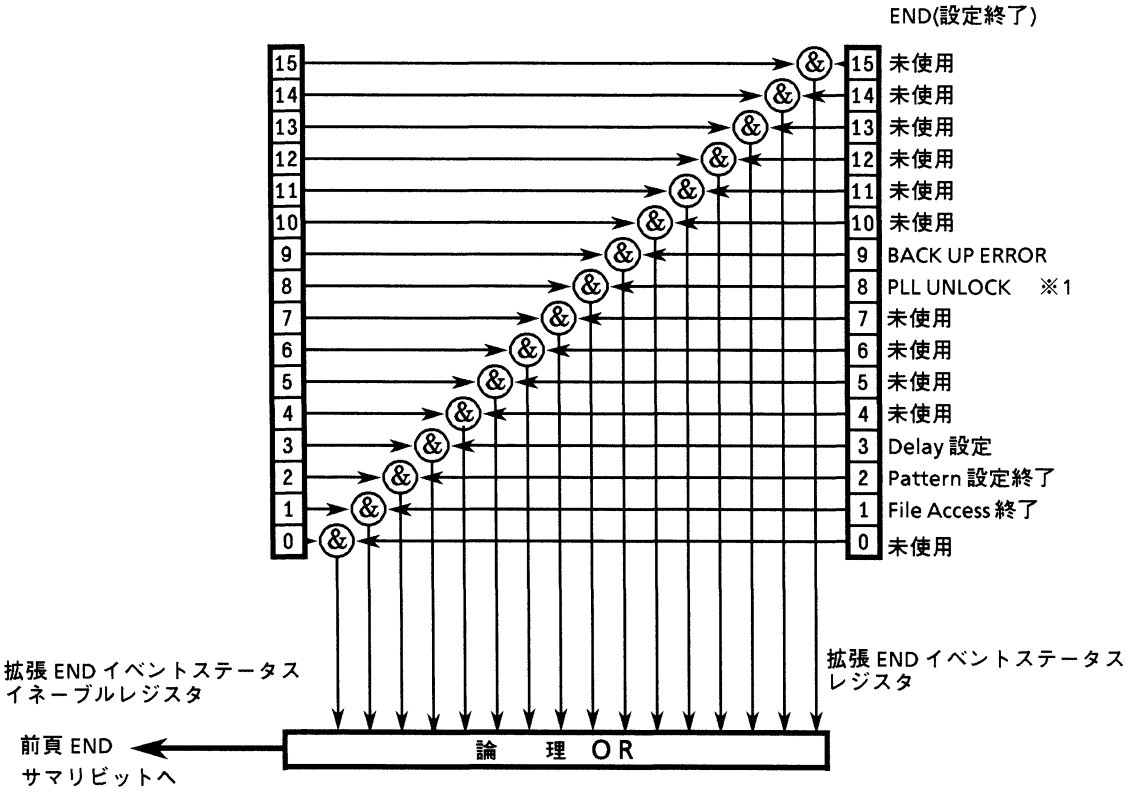
ステータスメッセージ

MP1761C で使用されるステータスバイト･レジスタのサービス要求用サマリメッセージの構造を下記に示します。

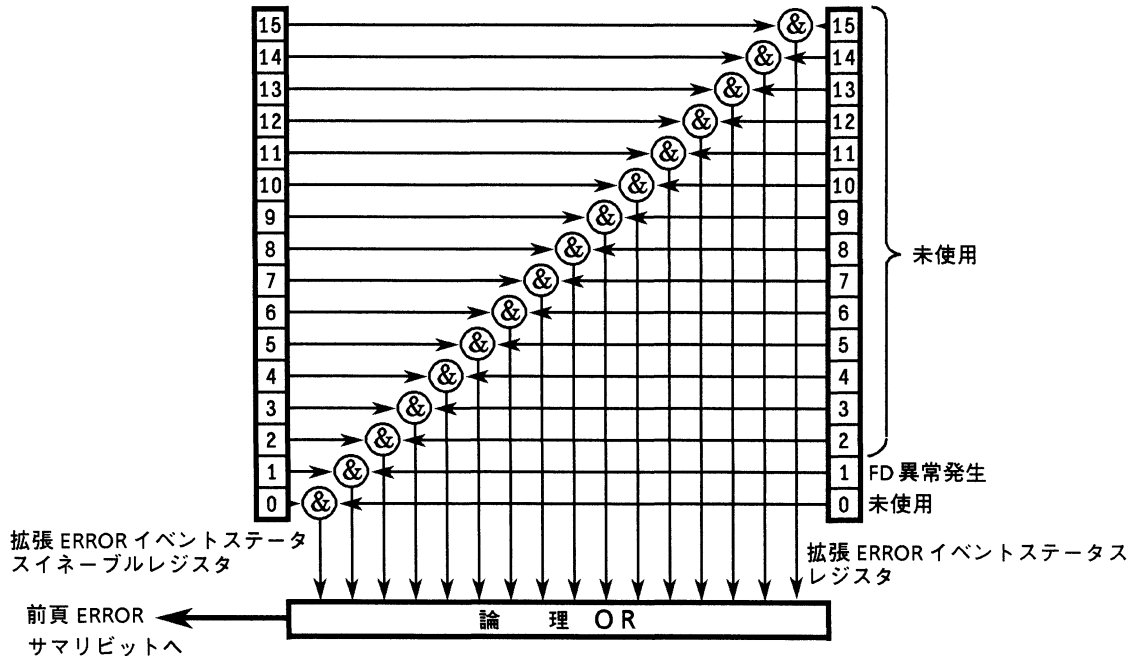
ステータスバイト･レジスタのサマリビット構成

ビット / ライン / サマリメッセージビット	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1
	未使用	RQS / MSS	ESB	MAV	ERROR	END	未使用	未使用





※1) PLL UNLOCK はOPTION-01 を実装している場合に限られます。



MP1761C デバイスメッセージ

MP1761C 固有のプログラムコマンド、クイアリ(問合せ)、およびレスポンスメッセージです。デバイスメッセージの一覧表および説明を9章に示します。

3 章
バス接続とアドレス設定

GPIBシステムインタフェース上に接続されたデバイスに対するリモート制御は、制御手順のパラメータとしてのアドレスを参照することから開始されます。
以下、この章では、GPIB使用前の準備として必ず行わなければならないGPIBケーブルの接続およびアドレス設定について説明します。

目　次

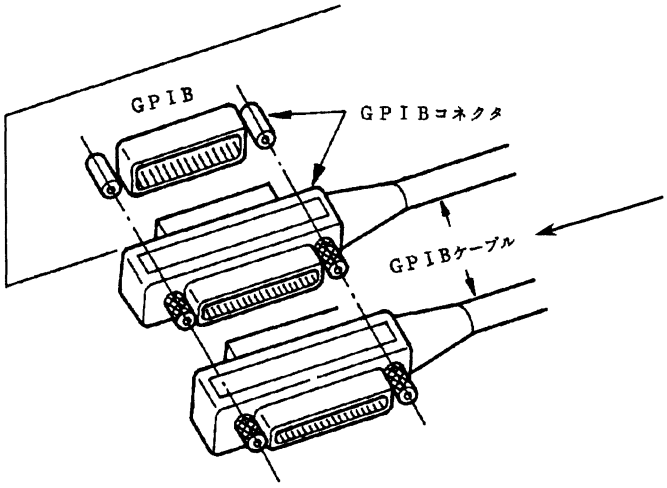
（空白）

3章
バス接続とアドレス設定

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、背面パネル上に取り付けられています。GPIBケーブルの接続は、必ず電源を投入する前に行ってください。

一つのシステムに接続可能なデバイス台数は、コントローラを含めて、最大15台までですが、その場合、下図右側に示した条件に従って接続してください。



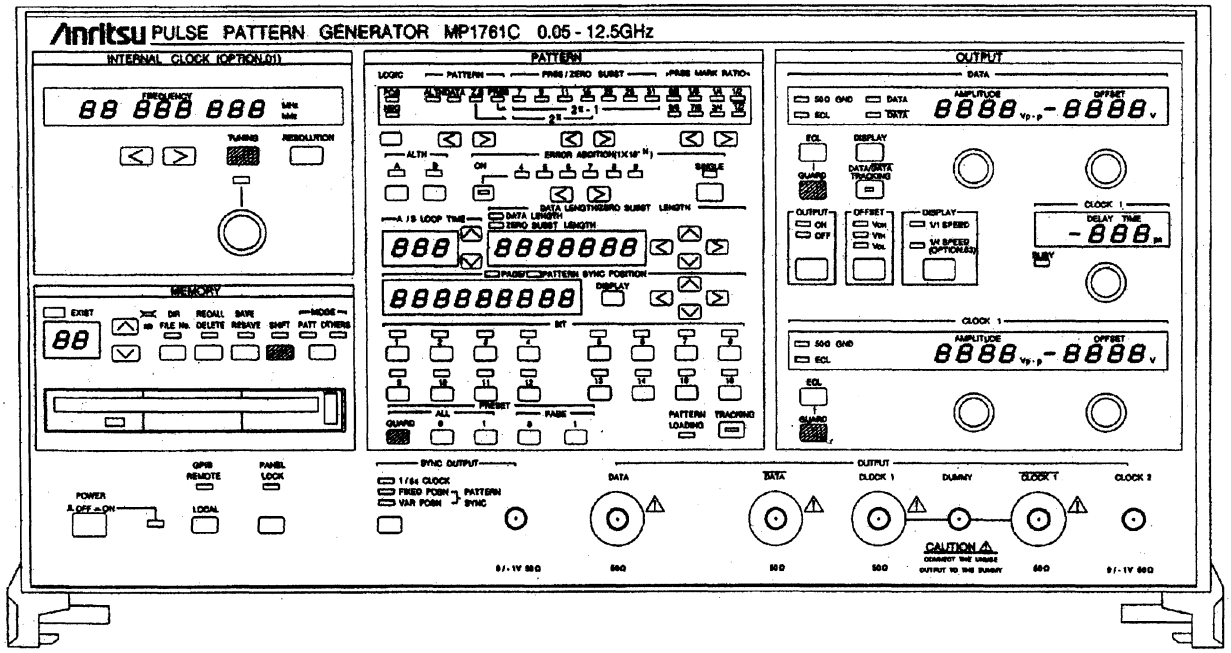
ケーブルの長さの総和	≦20 m
デバイス間のケーブルの長さ	≦4 m
接続可能なデバイス数	≦15

アドレスの確認および設定方法

MP1761C の GPIB アドレスは、電源投入前後に設定します。GPIB のアドレスは、工場出荷時に0番が背面パネル GPIB ADDRESS スイッチによって設定ずみです。アドレスを 0 番のまま使う場合には、アドレスを設定する必要はありません。アドレスを設定する場合は、MP1761C をローカル状態にしておき、背面パネルの GPIB ADDRESS スイッチによって設定します。一般に、電源投入時点では、GPIB 上のデバイスは、ローカル状態となります。

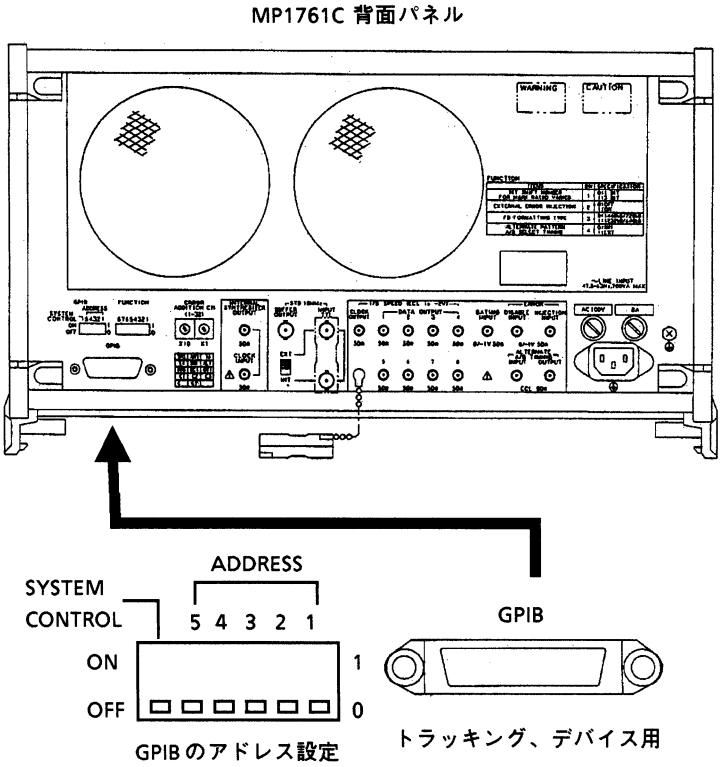
注: ● 本器は、通電中常時 GPIB の「ADDRESS」スイッチの設定状態をチェックして、自分自身のアドレスを決定しています。そのため、アドレスの変更は、本器がリモート状態の時以外は常時受け付けられます。

● 本器をデバイスとして外部コントローラから制御する場合は、GPIB アドレス・スイッチの「SYSTEM CONTROL」を OFF (0) にしてください。



アドレスの設定

MP1761C のGPIB ポートの GPIB アドレス設定は背面パネルのディップスイッチで行います。

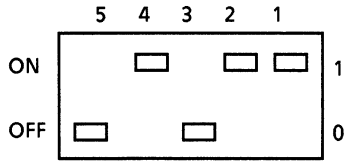


GPIB のアドレスとしては0~30が設定できます。5個のスイッチは、それぞれ重みが異なっており「5」は16、「4」は8、「3」は4、「2」は2、「1」は1の重みを持っています。

たとえば、アドレスを11に設定するには、

11 = 8 + 2+ 1

で下記のようにスイッチの「4」、「2」、「1」を ON にします。



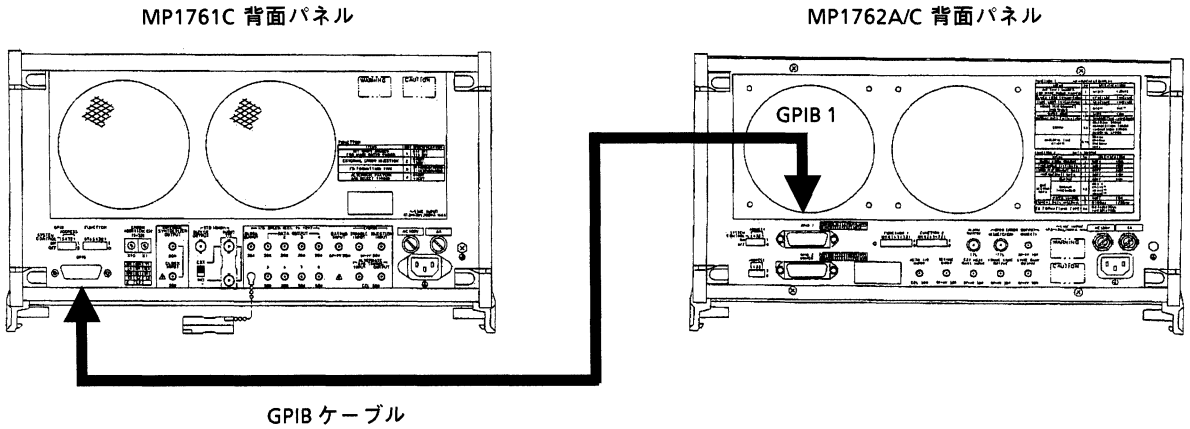
ただし、すべてのスイッチを ON としたアドレス31の場合は、アドレス0とみなします。

MP1762A/C とのトラッキング動作時の接続

トラッキング動作とは、MP1761C と MP1762A/C の間でパターン設定を同じにする機能です。MP1761C と MP1762A/C どちらか一方が Master となり、残りの一方が Slave となって Master 側の設定内容に Slave 側の設定をあわせます。

(1) MP1761C が Master となり MP1762A/C を制御する場合

MP1761C で設定された内容を MP1762A/C に GPIB を経由して設定する場合の設定方法と接続を以下に示します。



a) 図のように MP1761C の背面パネル GPIB コネクタと MP1762A/C の GPIB1 コネクタ間を GPIB ケーブル (添付されています。) で接続します。

b) MP1761C 背面パネルの GPIB アドレス SW にある「SYSTEM CONTROL」を ON (1) にします。

c) MP1762A/C 背面パネルの GPIB 1 アドレス SW の値を MP1761C の GPIB アドレス +2に設定します。

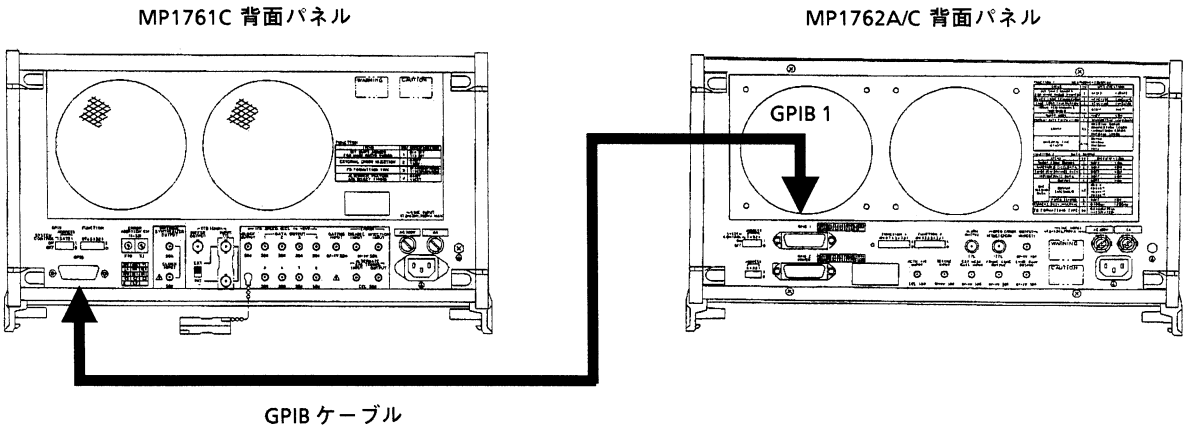
d) MP1761C の電源を再投入します。

e) MP1761C 正面パネルの TRACKING キーを ON にします。

これでパターントラッキングが可能となります。

(2) MP1762A/C が Master となり MP1761C を制御する場合

MP1762A/C で設定された内容を MP1761C に GPIB を経由して設定する場合の設定方法と接続を以下に示します。



a) 図のように MP1762A/C の背面パネル GPIB 1コネクタと MP1761C の GPIB コネクタ間を GPIB ケーブル (添付されています。) で接続します。

b) MP1762A/C 背面パネルの GPIB 1アドレス SW にある「SYSTEM CONTROL」を ON (1) にします。

c) MP1761C 背面パネルの GPIB 1アドレス SW の値を MP1762A/C の GPIB アドレス +2に設定します。

d) MP1762A/C の電源を再投入します。

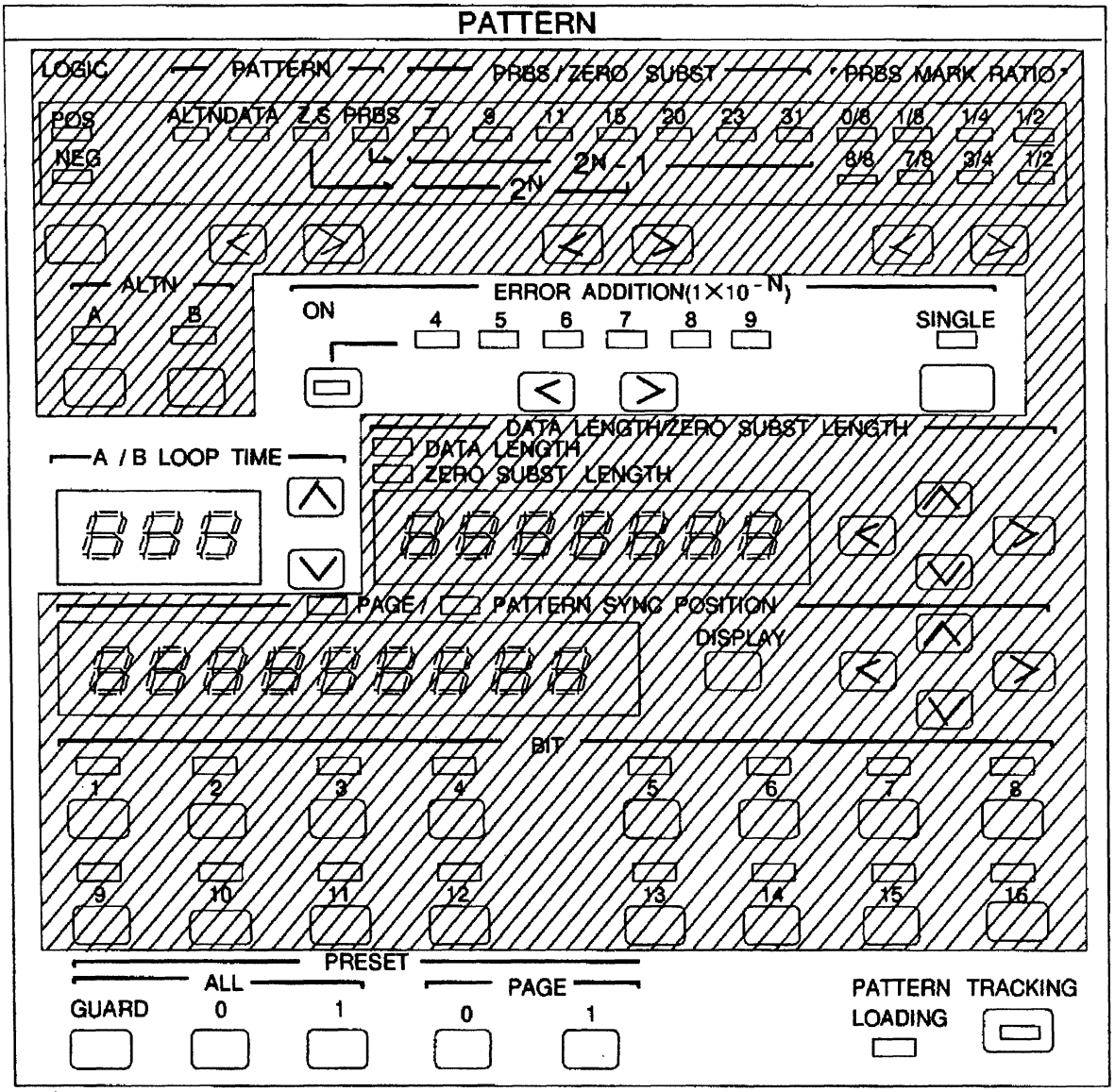
e) MP1762A/C 正面パネルの TRACKING キーを ON にします。

これでパターントラッキングが可能となります。

(3) MP1761C と MP1762A/C との間でトラッキングを行う項目

パターントラッキング機能でトラッキングを行う設定項目を以下に示します。

MP1761C 正面パネルのパターン設定部



トラッキングできる項目は、上図の網かけ部分のパターン設定についてです。

ただし、この中で MP1761C と MP1762A/C の設定が共通でない所(たとえば、MP1762A/C 誤り検出器内の誤り分析データ)は、パターン・トラッキングの対象となりません。

☞ パターン・トラッキングの対象となる設定詳細は、『付録 D　トラッキング項目一覧表』を参照してください。

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。

本章での書式、使用例は当社製 PACKET V シリーズ･パーソナルコンピュータによる制御コマンドを適用しています。

目　　次

（空白）

4章
イニシャル設定

GPIBシステムの初期化については、従来IEEE 488.1では、次の2点が規定されていました。

- バスの初期化 ..... コントローラからの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する。
- デバイスの初期化 .... 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、指定したデバイスだけをデバイスごとに定められている初期状態に戻す。

IEEE 488.2では、上記を3つのレベルに分け、第1レベルを『バスの初期化』とし、最も高位のレベルに位置付けました。『デバイスの初期化』は、さらに第2レベル『メッセージ交換の初期化』と第3レベル『デバイスの初期化』の2階層にわけて規定されると共に、電源投入時のデバイスの状態についても、ある既知の状態へ設定することが定められました。
以上のことをまとめると下表のようになります。

レベル	初期化の種類	概　要
1	バスの初期化	コントローラからの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。
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。
3	デバイスの初期化	*RSTまたはINIリセットコマンドによって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。

以下、レベル1, 2, 3については、これらを実行する命令およびその結果である初期化対象項目を中心に説明します。また、電源投入時に設定される既知の状態について説明します。

IFC @

IFCステートメントによるバスの初期化

■ 書　式

IFC△@セレクトコード

■ 使用例

```
IFC @1
```

■ 解　説

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、下表の中で○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。

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@**文の実行によって下図の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(パラメータの設定値、ランプのON/OFFなど)には影響を与えません。

前頁の表の中から**IFC**メッセージによるデバイスの状態をいくつか挙げます。

① 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。

② 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。

③ コントロール権の戻し ..... **IFC@**を実行した時、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。

④ 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。

⑤ 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。

DCL @

DCL, SDCバスコマンドによるメッセージ交換の初期化

■ 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]

■ 使用例

DCL @1	バス下の全デバイスのメッセージ交換の初期化(DCL送出)
DCL @1Ø3	アドレス3番のデバイスのみのメッセージ交換の初期化(SDC送出)

■ 解　説

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。

■ 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。

■ 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。

■ メッセージ交換の初期化対象項目

① 入力バッファと出力キュー	クリアされます。
② 構文解析部・実行制御部・応答作成部 ...	リセットされます。
③ *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。
④ 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合せもすべて捨てます。
⑤ *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。

⑥ ＊OPC?問合せの処理 ..... 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ビットはクリアされます。

⑦ 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。(MP1761Cでは、これらのコマンドをサポートしていません。)

⑧ 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。

注 意

下記事項は、デバイスクリアによって処理することを禁じられています。
①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。
②フロントパネルI/Oへの割り込み
③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。
④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。

■ 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の送出順序を下記に示します。

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	———

*RST

*RSTコマンドによるデバイスの初期化

■ 書　式

*RST

■ 使用例

```
WRITE @1Ø3:"*RST"
```

アドレス3番のデバイスのみをレベル3で初期化

■ 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。

■ 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。

バスコマンドの送出順序は、ATN ラインがこの期間に、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ます。ATN ラインが偽になると ***RST** コマンドを送出します。

■ デバイス初期化対象項目

① デバイス固有の機能・状態 .... それまでの来歴に関わらずある既知の状態に戻します。

② *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。

③ *OPC? 問合せの処理 ..... 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 ビットはクリアされます。

④ 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。

NOTE

*RSTコマンドは、下記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 Service Request Enableレジスタ
⑤ Standard Event Status Enable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 DMC (Define Macro Contents) コマンドで定義したマクロ
⑨ PUD (Protected User Data) 問合せに対するレスポンスメッセージ
⑩ RDT(Resource Description Transfer) 問合せに対するレスポンスメッセージ
この他、MP1761C固有のものとして時計、カレンダーに関する設定パラメータ等があります。
(☞③,④,⑤は8章を参照してください。なお、⑧～⑩については、MP1761Cではサポートしていません。)

前頁①デバイス固有の機能・状態について、MP1761C固有の初期設定一覧表を次頁に示します。

装置固有初期設定一覧表

グループ	初期設定	備　考
設定状態	工場出荷時の状態に初期化されます。	初期値は付録C　初期値一覧表を参照してください。
GPIBアドレス	初期化されません。	
時計、日付	初期化されません。	

INI

INIコマンドによるデバイスの初期化

■ 書　式

INI

■ 使用例(プログラムメッセージ)

```
WRITE @1Ø3:"INI"
```

アドレス3番のデバイスのみをレベル3で初期化

■ 解　説

INIコマンドはMP1761C固有のデバイスメッセージの一つで、デバイスをレベル3でリセットするため、そのコマンドを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ります。

■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。

コマンド送出順序は、ATNラインが真の期間に、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ます。ATNラインが偽になると、INIコマンドをプログラムメッセージとして、指定したリスナへ出力します。

■デバイス初期化対象項目

デバイス初期化対象項目は、設定状態および時計、日付です。

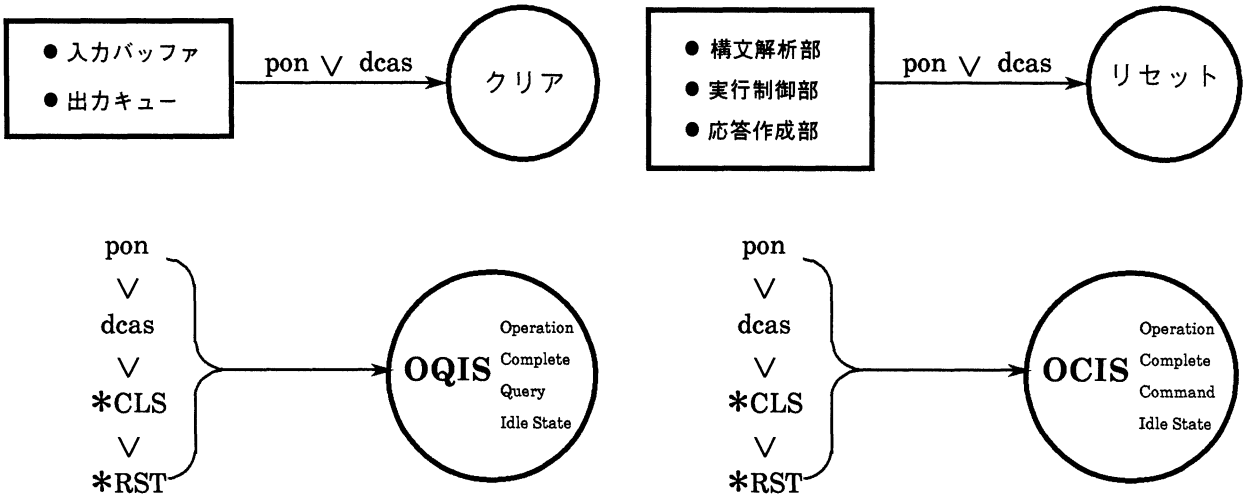
電源投入時のデバイスの状態

電源が投入されると：

① 最後に電源を OFF した時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 MP1761C は、***PSC** コマンドを用意していますので、もし PSC フラグが True の時すべてのイベント・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。

①の特別な場合として、MP1761C の工場出荷後、最初にその電源投入した時には、前述で示した初期設定一覧表 (☞付 C-1) のとおりに再現されます。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アグラムを下図に示します。



■ 電源投入時に変えてはいけない項目

① アドレス

② 関連するキャリブレーションデータ (MP1761C ではキャリブレーションデータはありません)

③ 以下のコモン疑問符に対するレスポンスで変化するデータやステート

*IDN?
*OPT?
*PSC?
*PUD?　(MP1761Cでは、サポートされていません)
*RDT?　(MP1761Cでは、サポートされていません)

■ POWER ON STATUS CLEAR (PSC) フラグに関係する項目

PSC フラグが False の時、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拡張イベント・ステータス・イネーブルレジスタは影響されません。
また、PSC フラグが True か、**＊PSC** コマンドが実行されていない時は、前記レジスタはクリアされます。

■ 電源投入時に変わってもよい項目

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
④ **＊DDT** コマンドで定義されたマクロ定義 (MP1761C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 (MP1761C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 (MP1761C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 (MP1761Cではサポートされていません)

5 章
リスナ入力フォーマット

バスのデータモードすなわちATNラインが偽の時に、システムインタフェースをとおして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には、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。以下、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。
本章でのプログラム例は、当社製PACKET Vシリーズ･パーソナルコンピュータを適用しています。

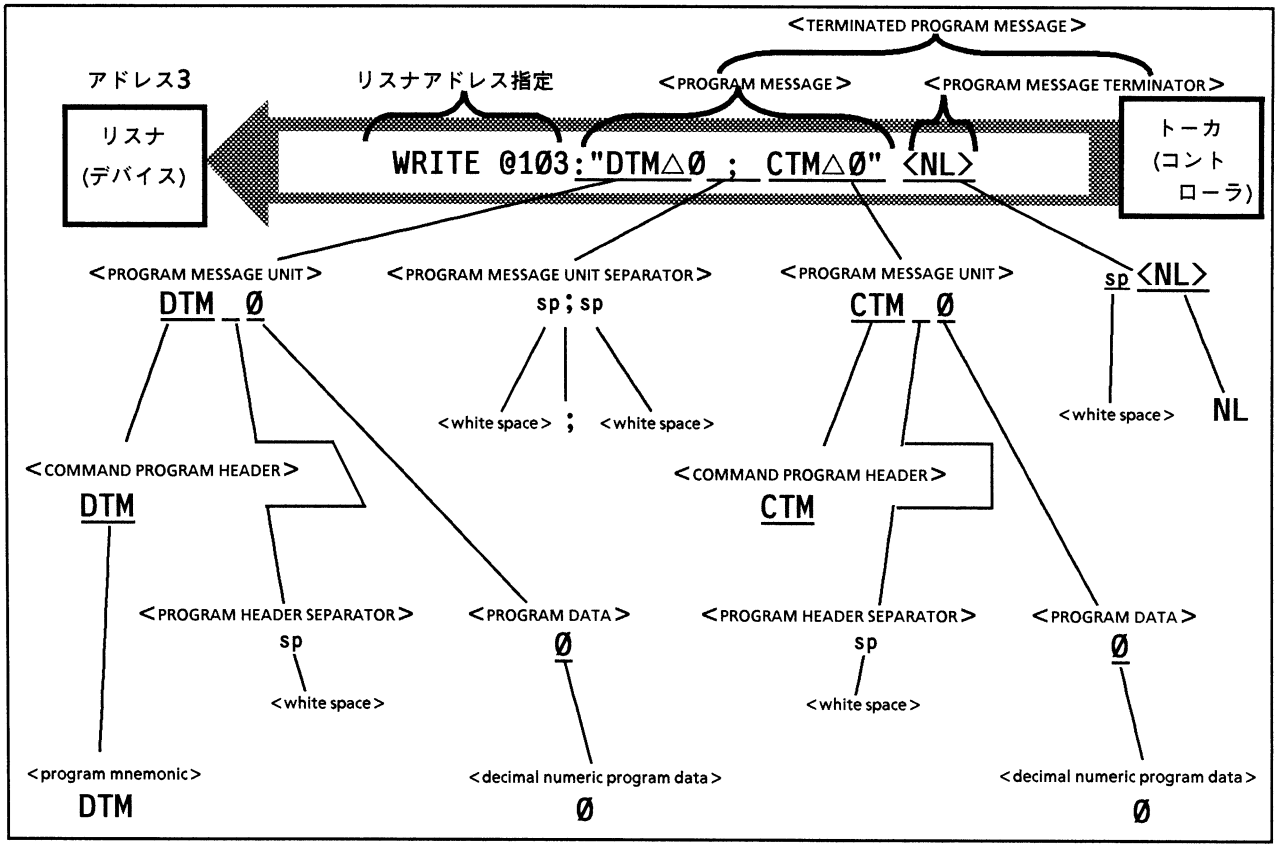
目　次

（空白）

5章
リスナ入力フォーマット

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、各々のユニットは、プログラム命令またはプログラム問合せです。

下図は、データ出力およびクロック出力終端電圧をGNDに設定するため、2つのプログラムメッセージ・ユニット**DTM△Ø**と**CTM△Ø**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。機能要素を更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したものを機能文法図と言います。また、特定の経路のコード化要素の選択を図で表したものをコード化文法図と言います。次ページからこの機能文法図・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に必要な実際のバスのコード化を表しています。機能要素のデータバイトを受信したリスナは、各々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。

リスナ入力プログラムメッセージ文法表記の要点

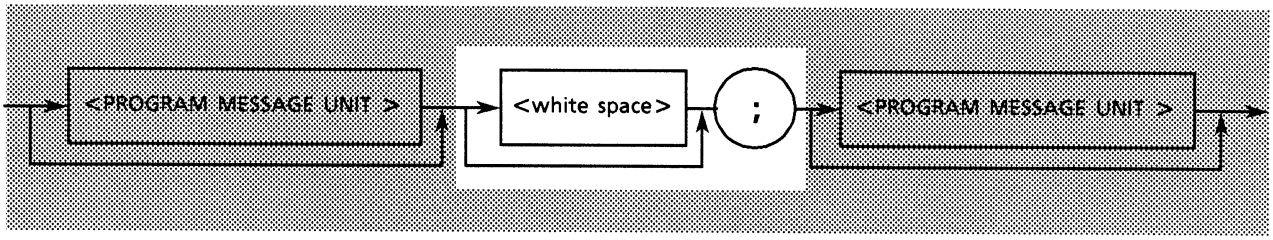
プログラムメッセージの機能要素およびプログラムデータのフォーマットの解説の要点を以下に示します。(複合コマンドと共通コマンドは省略)

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

複数のプログラムメッセージ･ユニットは、0個以上のスペース＋セミコロンで連結されます。

<例1> 2つのプログラムメッセージユニットの連結一般形式



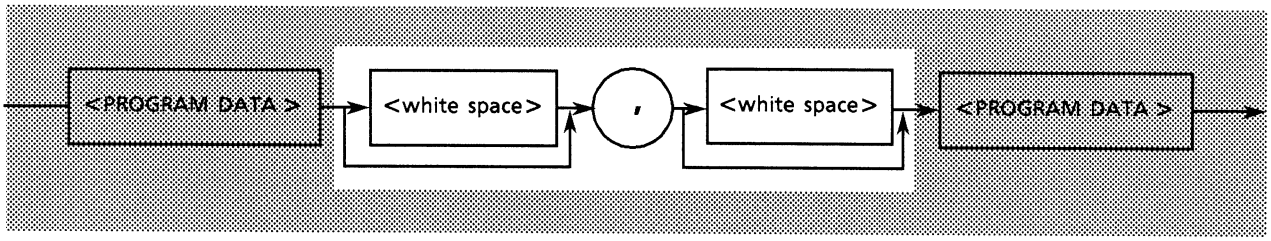
<例2> スペース1個 + セミコロン

DTM Ø△; CTM Ø

(2) PROGRAM DATA SEPARATOR

複数のプログラムデータは、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1> 2つのプログラムデータの区切り一般形式



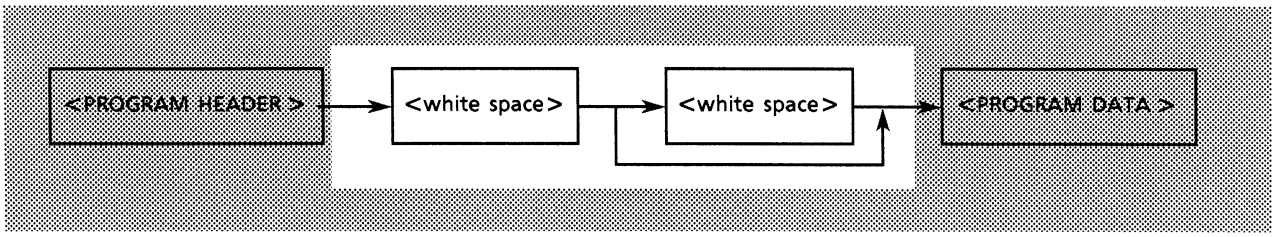
<例2> コンマのみ　<例3> コンマ + スペース1個

WRT 1,Ø　　WRT 1,△Ø

(3) PROGRAM HEADER SEPARATOR

プログラムヘッダとプログラムデータの間を1個のスペース＋0個以上のスペースで区切ります。

<例1> 単一コマンドプログラムヘッダ一般形式



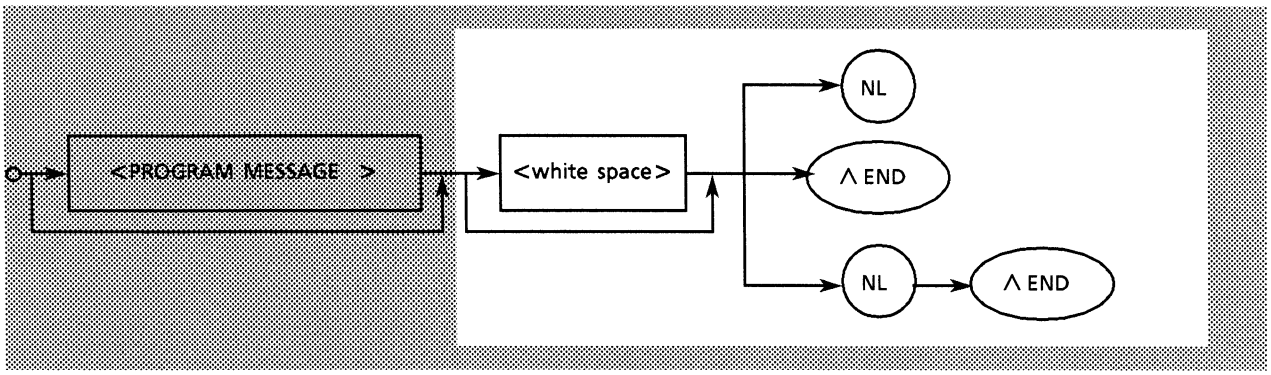
<例2> スペース1個

DTM△Ø

(4) PROGRAM MESSAGE TERMINATOR

プログラムメッセージの最後には、0個以上のスペース＋ {NL / EOI / NL + EOI} のいずれかを付加

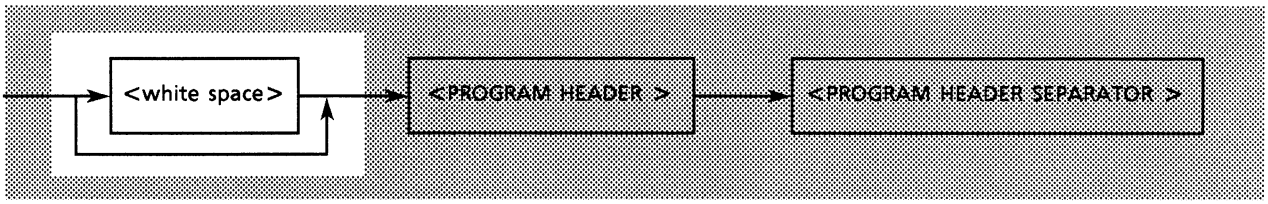
<一般形式>



(5) ヘッダ前置スペース

プログラムヘッダの前に、0個以上のスペースをおくことができます。

<一般形式>



<例> 2番目のプログラムヘッダ **SPF** の前にスペース1個

DTM Ø ;△CTM Ø

プログラムコマンドメッセージの一般形式

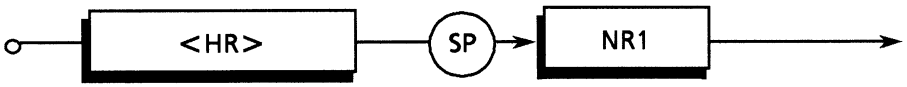
(1) データの指定を伴わないメッセージ

HR : COMMAND PROGRAM HEADER

<例>

INI　初期設定

(2) 整数データを伴うメッセージ

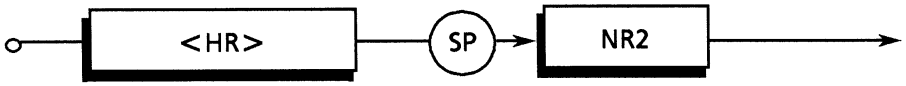


NR1 : 整数

<例>

PTS△0	ALTERNATE パターンの設定
PTS△1	DATA パターンの設定
PTS△2	ZERO SUBSTITUTIONパターン の設定
PTS△3	PRBS パターンの設定

(3) 実数を伴うメッセージ

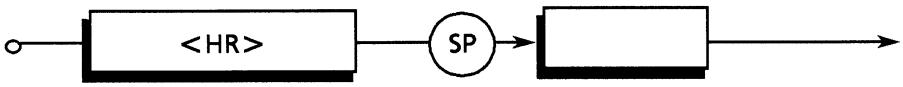


NR2 : 実数

<例>

CAP△2.000　　クロック出力振幅設定

(4) HEX (16進数) を伴うメッセージ

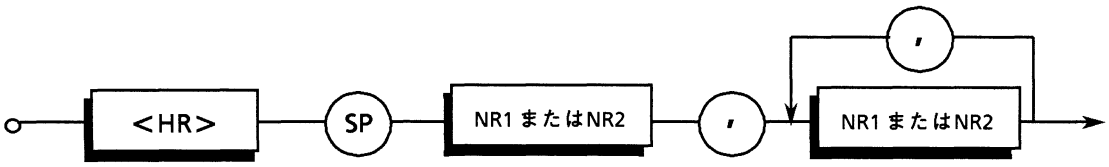


: HEX

<例>

BIT△#H FFFF

(5) 複数のプログラムデータを伴うメッセージ



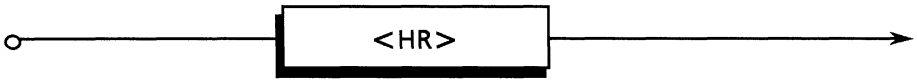
<例>

RTM△99,10,10,14,30,30　　内蔵タイマを99年10月10日14時30分30秒に設定

問合せメッセージの一般形式

問合せ PROGRAM HEADER は、コマンド PROGRAM HEADER の末尾に? を付けます。

(1) 問合せデータの指定を伴わないメッセージ



<例>

DTM?　　データ出力終端電圧データを要求する

(2) 問合せデータの指定を伴うメッセージ

<例>

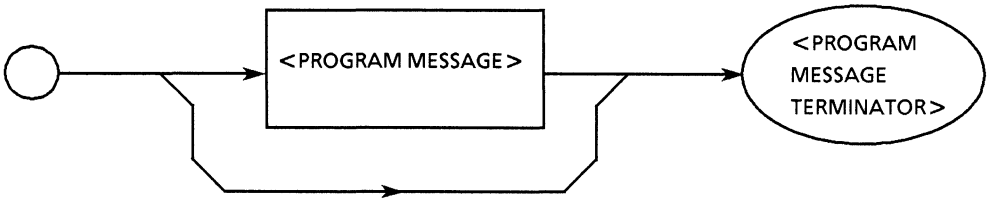
FSH? 1　　フロッピーディスク中の測定条件が格納されているファイルのうちファイル No. が51以降のファイル情報を要求する。

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、このプログラムメッセージの各機能要素を説明します。

<TERMINATED PROGRAM MESSAGE>

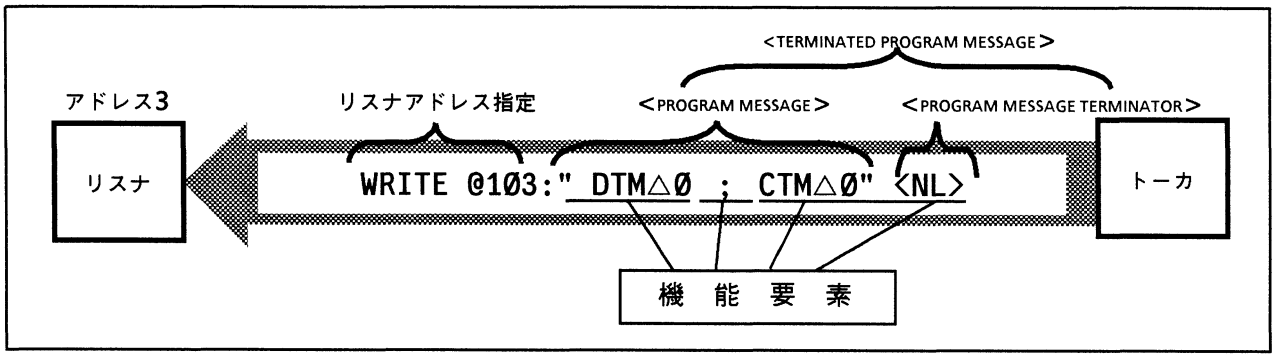
<TERMINATED PROGRAM MESSAGE> は、次のように定義されます。



<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。

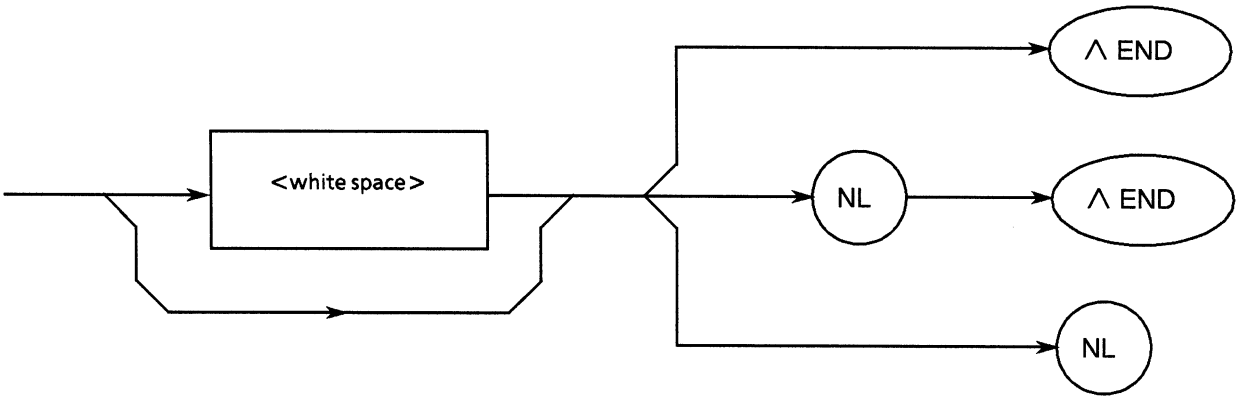
<PROGRAM MESSAGE >の転送を完了させるため、<PROGRAM MESSAGE >の最後には、<PROGRAM MESSAGE TERMINATOR>が付加されます。

<例> WRITE 文で2個の命令を送る <TERMINATED PROGRAM MESSAGE>



<PROGRAM MESSAGE TERMINATOR>

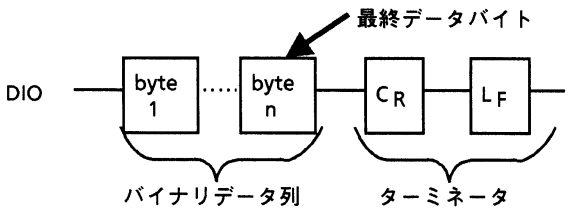
<PROGRAM MESSAGE TERMINATOR>は、次のように定義されます。



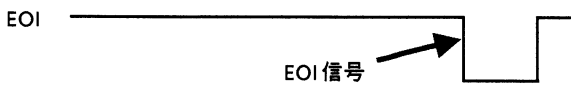
<PROGRAM MESSAGE TERMINATOR>は、一つ、またはそれ以上の一定の長さの<PROGRAM MESSAGE UNIT>要素のシーケンスを終了させます。

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、印字位置を次の行の同じキャラクタ位置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。**WRITE @**文で<PROGRAM MESSAGE >を送る場合、**WRITE @**文は、自動的にCR・LFを送出しますので、プログラム作成時にCR・LFコードを発生させる記述は必要ありません。この場合、LFコードのみを発生させるには、下記のステートメントをプログラムの初めで実行します。**TERM IS CHR$(1Ø)**

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。

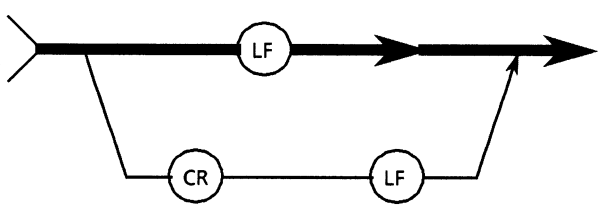


EOIラインを制御するステートメントに**EOI ON / OFF**文があります。デフォルトでは、**EOI OFF**を実行した状態と同じで、EOIラインの制御は行われませんが、あらかじめ**EOI ON**を実行しておけば、**WRITE @**文の最終データバイト送出時に、ターミネータLFと同時にEOI信号が送出されます。



LFを送らないで、END信号だけで<PROGRAM MESSAGE >を終了させる場合もあります。

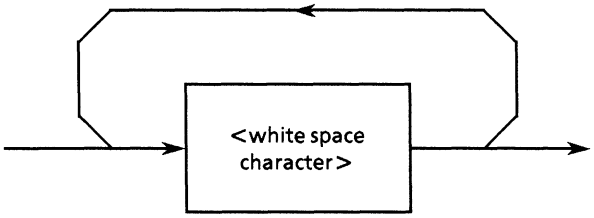
NOTE



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は、CRコードに続いてLFコードを出力する方式が多い。

<white space>

<white space>は、次のように定義されます。

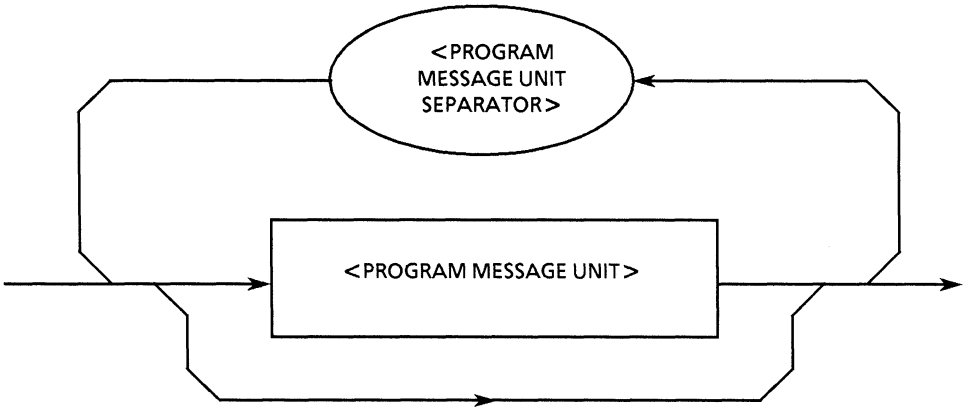


<white space character>は、ASCIIコードバイト00～09, 0B～20(10進数、0～9, 11～32)の範囲の中で、単一のASCIIコードバイトとして定義されます。

その範囲は、ニューラインを除き、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。

<PROGRAM MESSAGE>

<PROGRAM MESSAGE>は、次のように定義されます。

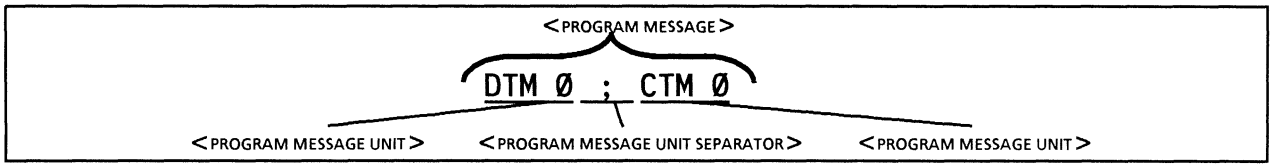


<PROGRAM MESSAGE>とは、それらはゼロであるか、1個の<PROGRAM MESSAGE UNIT>要素、または、より多くの<PROGRAM MESSAGE UNIT>要素のシーケンスです。<PROGRAM MESSAGE UNIT>要素は、コントローラからデバイスに送られるプログラミング命令か、データを意味しています。<PROGRAM MESSAGE UNIT SEPARATOR>要素は、複数の<PROGRAM MESSAGE UNIT>を区切るためのセパレータとして使用されます。

<例 1>　データ入力終端電圧をGNDに設定するプログラム・メッセージ

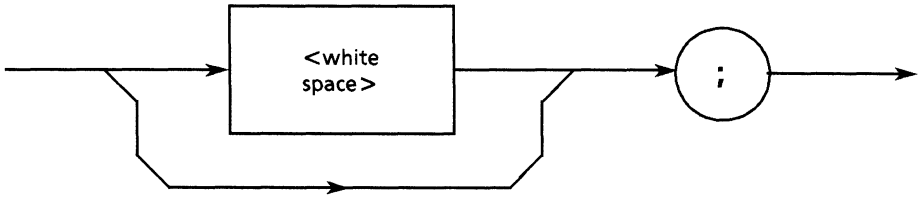
```
DTM Ø
```

<例 2>　上記設定に続き、クロック入力終端電圧をGNDに設定するプログラム・メッセージ

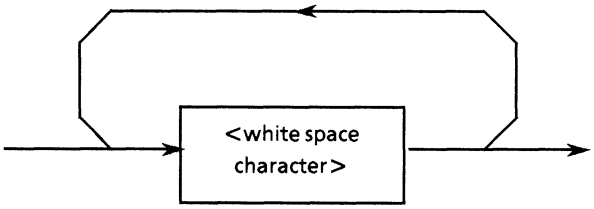


<PROGRAM MESSAGE UNIT SEPARATOR>

<PROGRAM MESSAGE UNIT SEPARATOR>は、次のように定義されます。



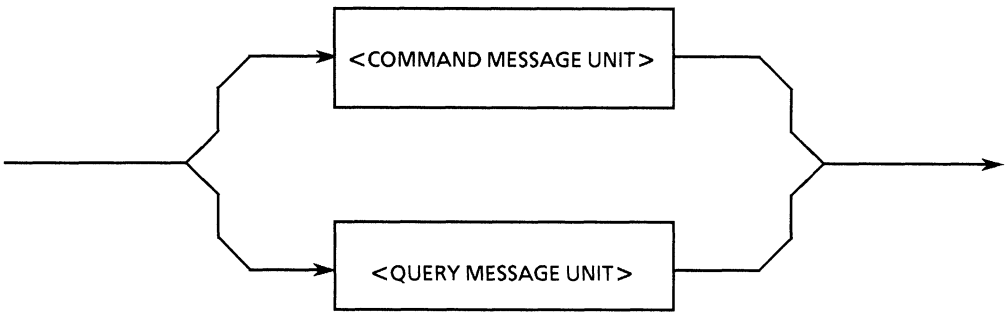
<white space>は、次のように定義されます。



<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、プログラムを読み易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、次のプログラムヘッダの前におかれた<white space>です。

<PROGRAM MESSAGE UNIT>

<PROGRAM MESSAGE UNIT>は、次のように定義されます。

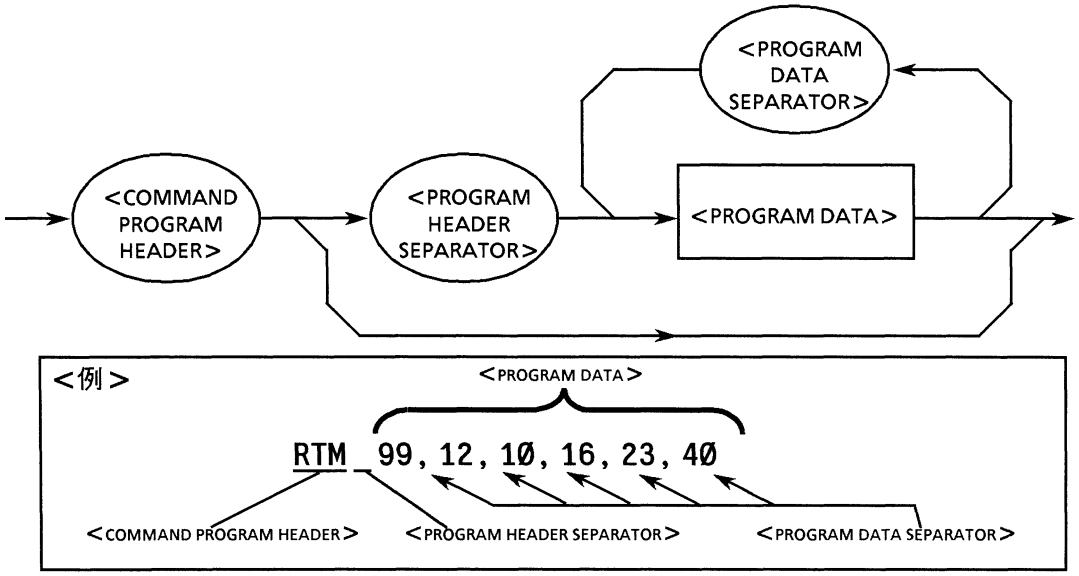


<PROGRAM MESSAGE UNIT>は、デバイスで受信される単一のコマンドメッセージである<COMMAND MESSAGE UNIT>または単一の問合せメッセージである<QUERY MESSAGE UNIT>から成ります。

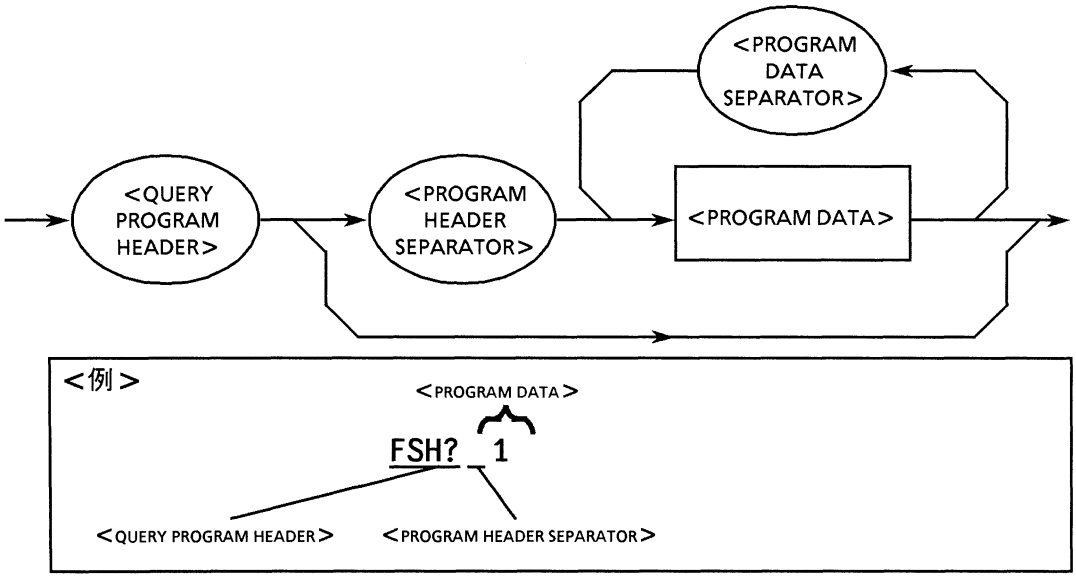
<COMMAND MESSAGE UNIT>と<QUERY MESSAGE UNIT>の詳細は、次ページで説明します。

<COMMAND MESSAGE UNIT>/<QUERY MESSAGE UNIT>

1)　<COMMAND MESSAGE UNIT>は、次のように定義されます。



2)　<QUERY MESSAGE UNIT>は、次のように定義されます。

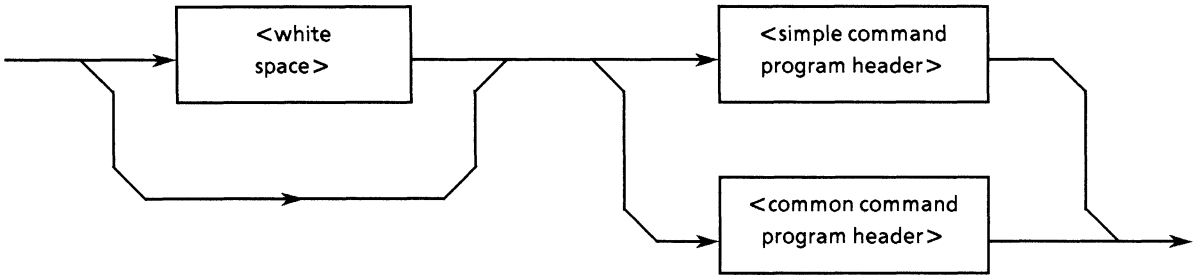


<COMMAND MESSAGE UNIT>も<QUERY MESSAGE UNIT>も、それぞれプログラムヘッダの次にプログラムデータが続く場合は、必ずその間にスペースが1個、セパレータとしてはいります。プログラムヘッダによって、プログラムデータの用途・機能・動作が分かります。プログラムデータが付かない場合は、ヘッダだけでデバイスの中で実行される用途・機能・動作を表します。

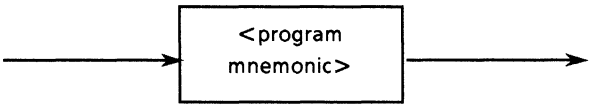
プログラムヘッダの中で、<COMMAND PROGRAM HEADER>は、コントローラからデバイスを制御するコマンドであり、<QUERY PROGRAM HEADER>は、コントローラがデバイスからレスポンスメッセージを受信するため、あらかじめコントローラからデバイスへ送る問合せ用コマンドです。そのヘッダの末尾には、かならず、問合せインジケータ?がつけられるのが特徴です。

<COMMAND PROGRAM HEADER>

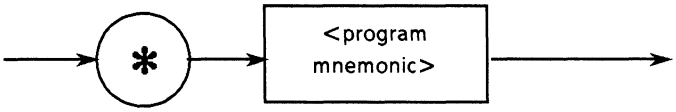
<COMMAND PROGRAM HEADER>は、次のように定義されます。
各ヘッダの前には<white space>をおくことができます。



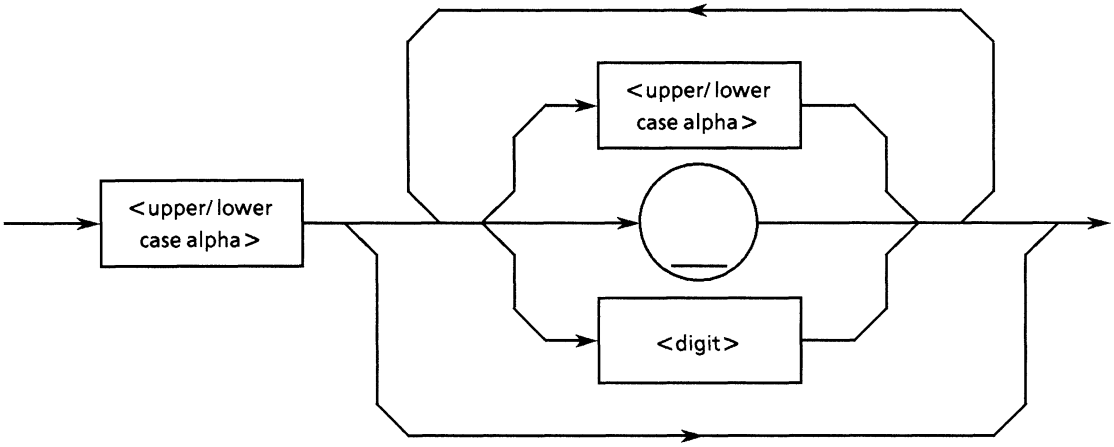
1)　<simple command program header>は、次のように定義されます。



2)　<common command program header>は、次のように定義されます。



3)　<program mnemonic>は、次のように定義されます。



■ <COMMAND PROGRAM HEADER>

デバイスが実行するプログラムデータの用途・機能・動作を表すもので、プログラムデータが付かない場合は、ヘッダだけでデバイスの中で実行される用途・機能・動作を表します。

それらの意味をASCIIコード文字で表したのが<program mnemonic>で、一般には、ニーモニックと呼ばれています。以下、ニーモニックの規定および上記1), 2)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"の任意の組合せが続きます。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。
- **<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>

上で述べた<program mnemonic>の規定がそのまま適用されます。

■ <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IEEE 488.2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。

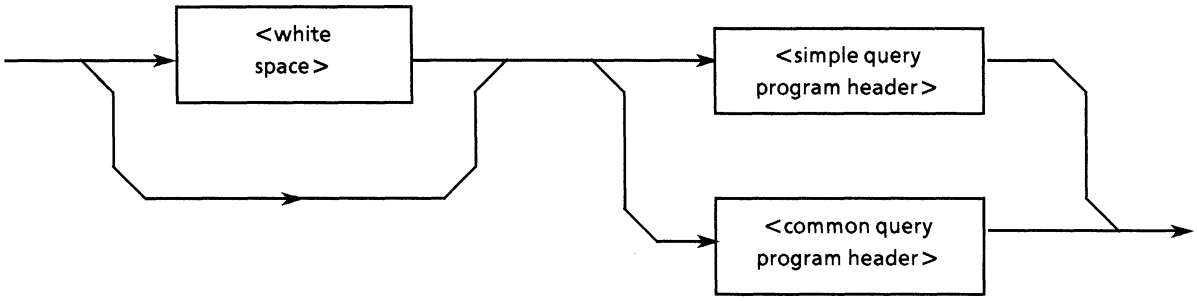
- **<例>**　セレクトコード1のGPIBインタフェースに接続されているアドレス3のデバイスのオペレーション終了をアイドルにし、各デバイスを、決められた固有の状態に初期設定する。

```
WRITE @1Ø3:"*RST" .....
```

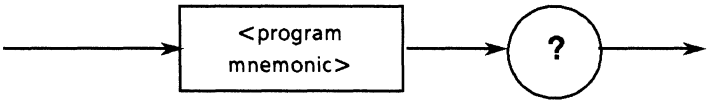
引用符" "の中がIEEE 488.2共通コマンド*RSTで、上記を実行します。

<QUERY PROGRAM HEADER>

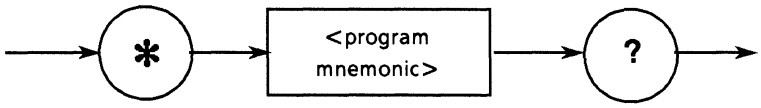
<QUERY PROGRAM HEADER>は、次のように定義されます。
各ヘッダの前には<white space>をおくことができます。



1)　<simple query program header>は、次のように定義されます。



2)　<common query program header>は、次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合せ用コマンドです。そのヘッダの末尾には、必ず、問合せインジケータ?がつけられるのが特徴です。

☞ 以上述べた<QUERY PROGRAM HEADER>の形式は、ヘッダの末尾に問合せインジケータ?が付けられる以外は、<COMMAND PROGRAM HEADER>に同じです。

<PROGRAM HEADER SEPARATOR>

<PROGRAM HEADER SEPARATOR>は、次のように定義されます。

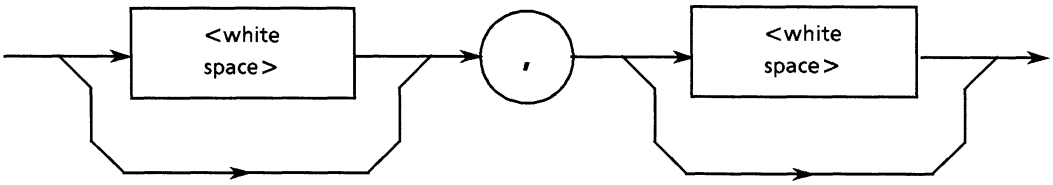
<PROGRAM HEADER SEPARATOR>は、<COMMAND PROGRAM HEADER>または<QUERY PROGRAM HEDADER>と<PROGRAM DATA>の間のセパレータとして使用されます。

プログラムヘッダとプログラムデータの間に複数の<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、プログラムヘッダの終わりであると同時にプログラムデータの始まりを示しています。

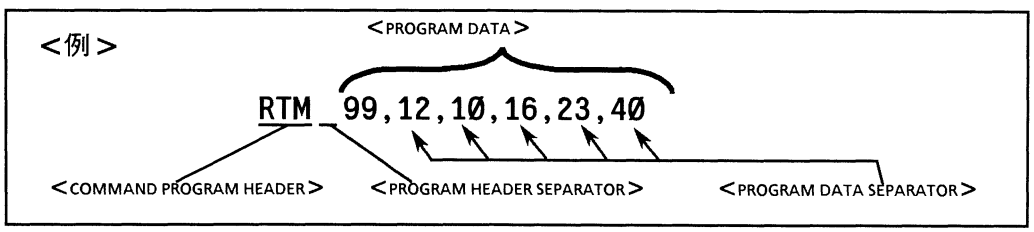
<PROGRAM DATA SEPARATOR>

<PROGRAM DATA SEPARATOR>は、次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。

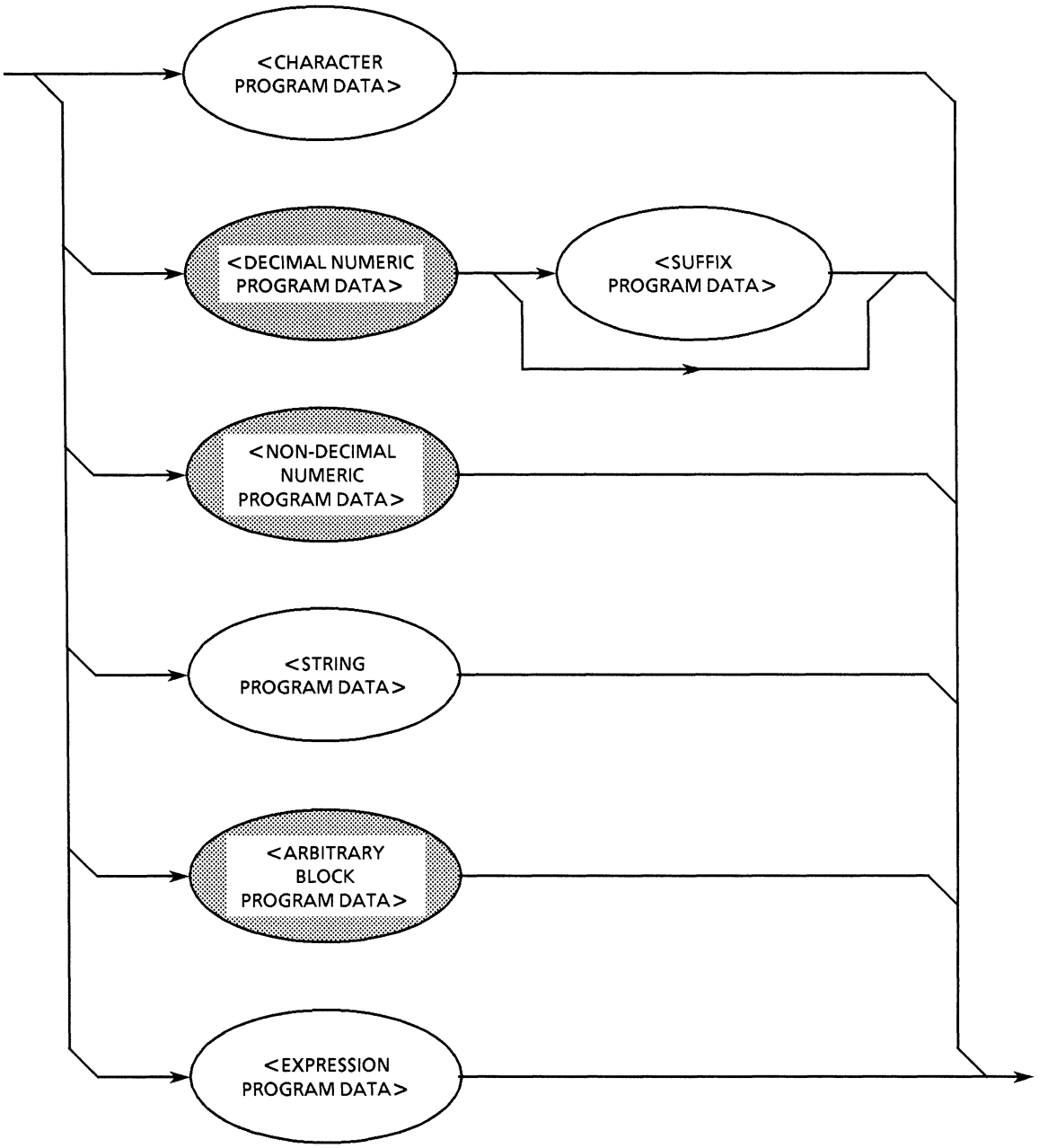
このデータセパレータを使う場合、コンマは必ず必要ですが、<white space character>は、必ずしも必要ではありません。コンマの前または後の<white space character>は、読み飛ばされます。ただし、<white space character>は、プログラムを読みやすくするためには、有用です。



プログラムデータのフォーマット

前述、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、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、プログラム･ヘッダ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。下図で、それらのプログラムデータの種類を示します。MP1761Cは、この中からアミで囲まれ、白抜きされたプログラムデータをアクセプトします。



<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プログラムデータです。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。浮動小数点形式について MP1761C では使用していません。

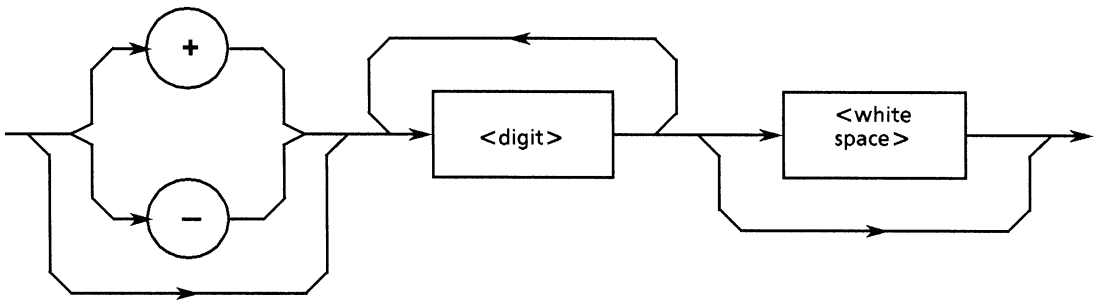
MP1761C で使用している整数形式、固定小数点形式のプログラムデータ伝送について説明します。

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、4捨5入を行います。
- レンジ外データ　<DECIMAL NUMERIC PROGRAM DATA>要素の値がプログラム・ヘッダとの関連において、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－NR1伝送

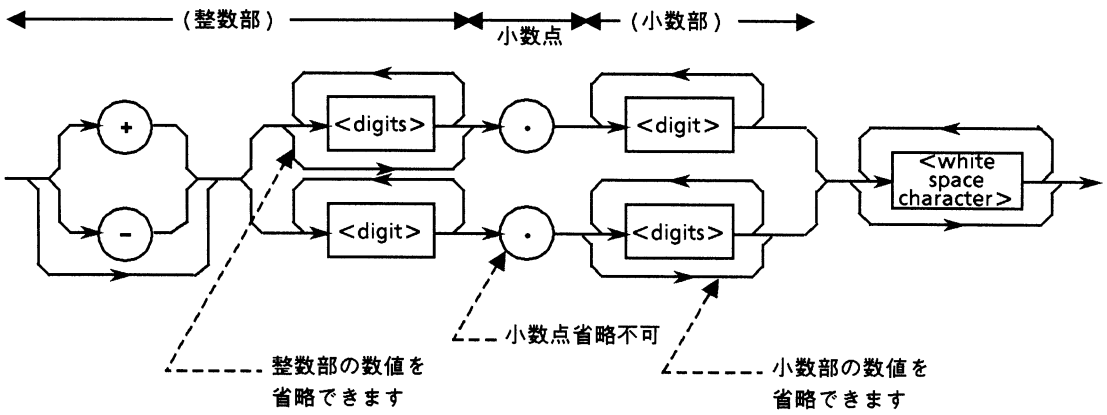
小数点や指数表現を含まない10進数値、すなわち実数の中の整数(NR1)を伝送します。



- 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

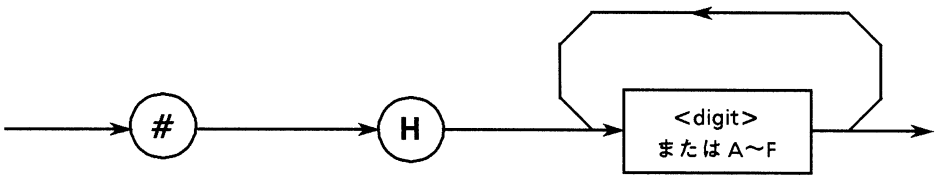
小数点以下の桁を持つ10進数値、すなわち整数および指数表現を除く実数(NR2)を伝送します。文法図は、(整数部)、小数点、(小数部)から成ります。



- (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不可→　+753△.123 (×)
- (小数部)の数字の後ろにスペース挿入可→　+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可。→　12.

<NON-DECIMAL NUMERIC PROGRAM DATA>

非10進数値として、16進数値データを伝送するプログラムデータです。非10進データは、必ず＃マークから始まります。下図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 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。

＜例＞　　#HABCD　　(43, 981 D)

6 章
トーカ出力フォーマット

バスのデータモードすなわちATNラインが偽の時に、システムインタフェースをとおして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には、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。
本章での書式、使用例は当社製PACKET Vシリーズ・パーソナルコンピュータによる制御コマンドを適用しています。

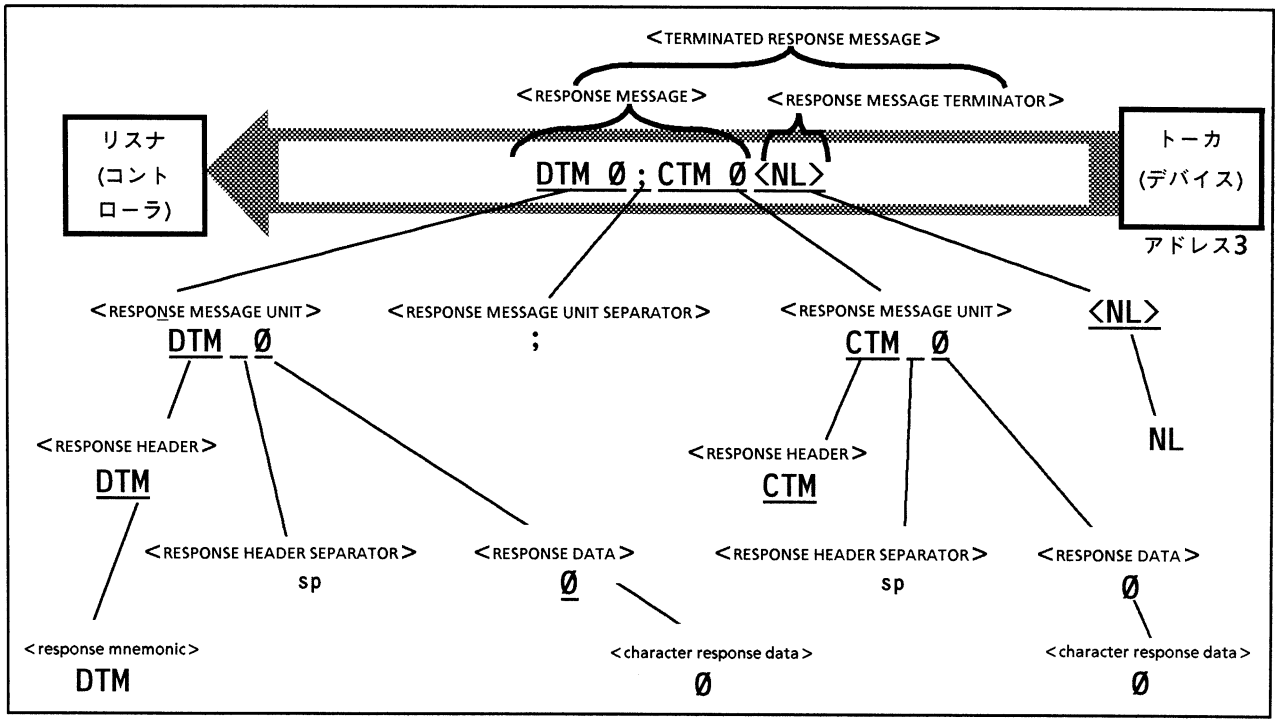
目　次

（空白）

6章
トーカ出力フォーマット

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無しで返されるレスポンスメッセージがあります。

下図は、データ出力終端電圧問合せメッセージユニット **DTM?** およびクロック出力終端電圧問合せメッセージユニット **CTM?** に対して、それぞれのレスポンスメッセージがヘッダ付き ASCII 文字列として、デバイスからコントローラへ送出され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せば、次のとおりとなります。

```
100 WRITE @103:"DTM? "!     データ出力終端電圧問合せメッセージユニット
110 READ @103:A$!           ターミネータ NL が検出されると、レスポンスメッセージ
                            DTM△Øが A$ へ読み込まれます。

120 WRITE @103:"CTM? "!     クロック出力終端電圧問合せメッセージユニット
130 READ @103:B$!           クロック出力終端電圧のレスポンスメッセージ
                            CTM△Øが B$ へ読み込まれます。
```

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。

以下、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。

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、

- 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、一つ機能に対しては一つ表記法しかありません。

下表は、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たものです。なお、表中の0/1個以上のスペースは<white space>を指します。

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る。 ただしヘッダについては大文字のみ。	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 ただしサポート無し	大文字 E のみ
NR3 指数部の+符号	省略可能　ただしサポート無し	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る。	不使用
メッセージユニット	①プログラムデータ付きヘッダ ②プログラムデータ無しヘッダ	①ヘッダ付きデータ ②ヘッダ無しデータ
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*
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} のいずれか	NL + EOI

* ASCII コードバイト20(10進数32 = ASCII 文字SP, スペース)

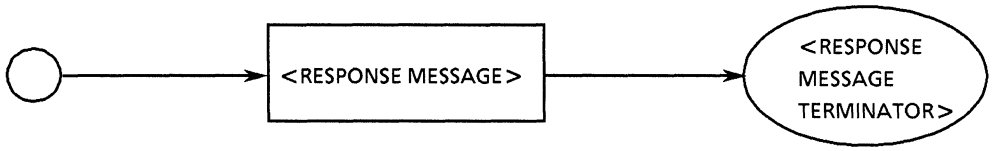
レスポンスメッセージの機能要素

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、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、5章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、5章を参照してください。

<TERMINATED RESPONSE MESSAGE>

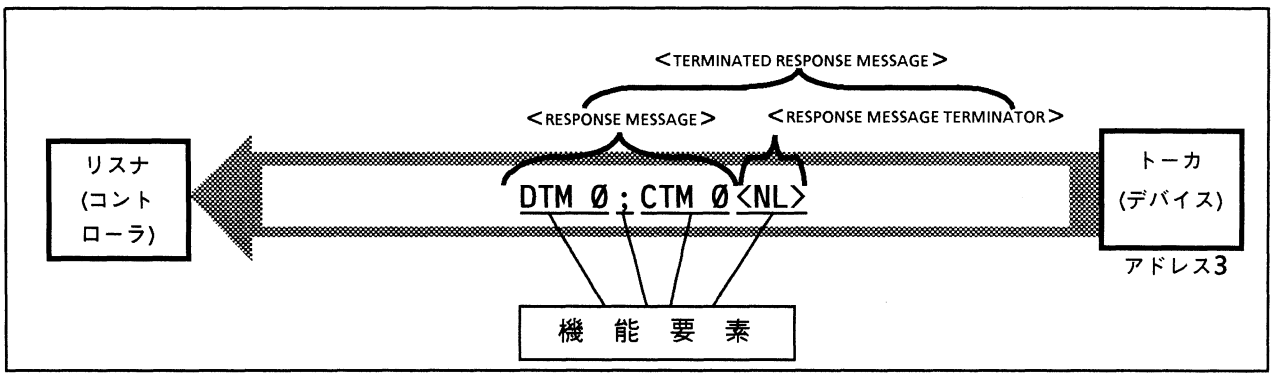
<TERMINATED RESPONSE MESSAGE>は、次のように定義されます。



<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。

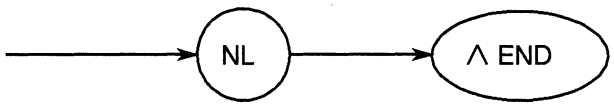
<RESPONSE MESSAGE>の転送を完了させるため、<RESPONSE MESSAGE>の最後には、<RESPONSE MESSAGE TERMINATOR>が付加されます。

<例> 2個の メッセージユニットを連結した<TERMINATED RESPONSE MESSAGE>



< RESPONSE MESSAGE TERMINATOR>

< RESPONSE MESSAGE TERMINATOR>は、次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。

NL^END は次のステートメントをプログラムの開始部分で実行しておけば、最終データバイト送出時にターミネータLFと同時にEND信号としてEOI信号が送出されます。

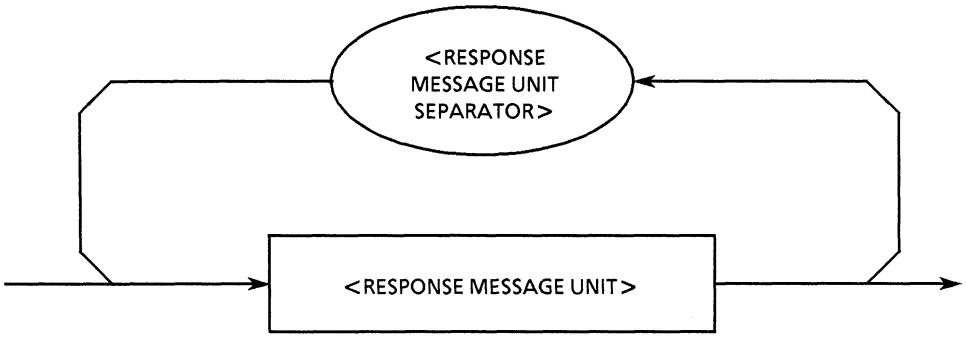
- NLに対しては、　　TERM IS CHR$(10)
- ENDに対しては、　　EOI ON

<例> 現在、設定されているアラーム付加を読み出す

```
10 LET ADR=101
20 TERM IS CHR$(10)! ..... ターミネータコードをLF(New Line)にする
30 EOI ON! ..... 最終データバイト送出時にEOIラインをTRUEとするEOI信号を出力
40 WRITE @ADR:"DTM?"! ..... データ出力終端電圧読み出し用問合せ
50 READ @ADR:A$! ..... レスポンスデータの読み込みをEOI信号によって終了させる
60 PRINT A$
70 END
```

<RESPONSE MESSAGE>

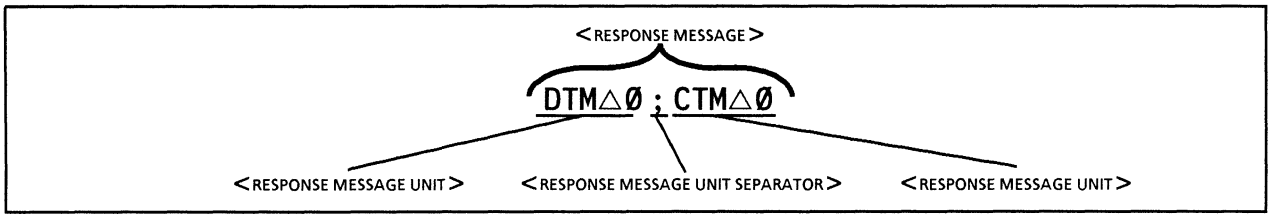
<RESPONSE MESSAGE>は、次のように定義されます。



<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。

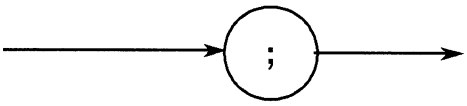
<RESPONSE MESSAGE UNIT>要素は、デバイスからコントローラに送られる単一のメッセージを意味しています。<RESPONSE MESSAGE UNIT SEPARATOR>要素は、複数の<RESPONSE MESSAGE UNIT>を区切るためのセパレータとして使用されます。

<例>　データ出力終端電圧およびクロック出力終端電圧にヘッダDTMとCTMを付け、それらのデータを1文字の固定フォーマットで送出する



<RESPONSE MESSAGE UNIT SEPARATOR>

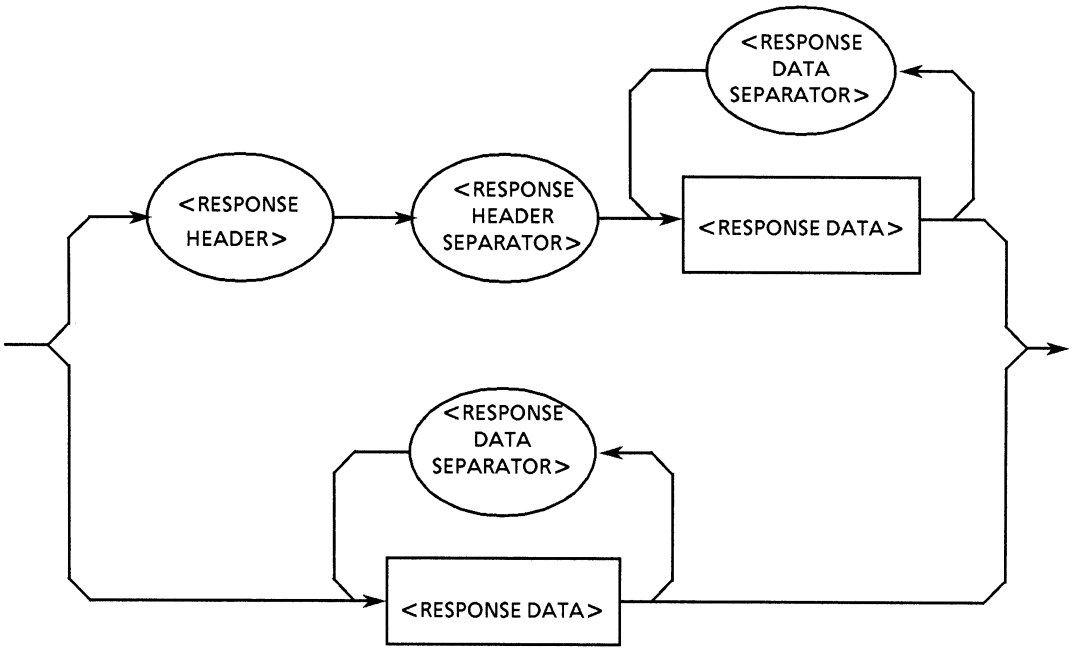
<RESPONSE MESSAGE UNIT SEPARATOR>は、次のように定義されます。



<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。

<RESPONSE MESSAGE UNIT>

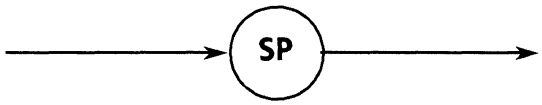
<RESPONSE MESSAGE UNIT>は、次のように定義されます。



<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。前者はヘッダ付きレスポンス・メッセージ・ユニットで、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。後者はヘッダ無しレスポンス・メッセージ・ユニットで、測定結果のデータだけを無駄なく返します。

< RESPONSE HEADER SEPARATOR >

<RESPONSE HEADER SEPARATOR>は、次のように定義されます。



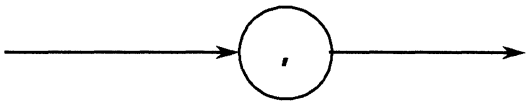
<RESPONSE HEADER SEPARATOR>は、<RESPONSE HEADER>の次に1個のスペースを置き、<RESPONSE HEADER>と<RESPONSE DATA>を分離します。

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、ヘッダ付きレスポンスメッセージでは、レスポンス・ヘッダ・セパレータ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、レスポンス・ヘッダの終わりであると同時にレスポンスデータの始まりを示しています。

< RESPONSE DATA SEPARATOR>

<RESPONSE DATA SEPARATOR>は、次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。

< RESPONSE HEADER >

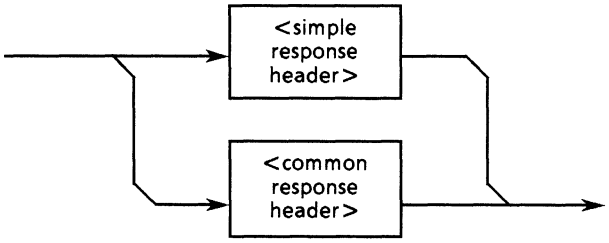
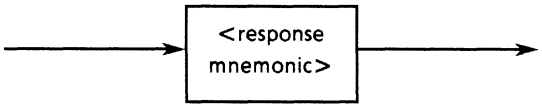
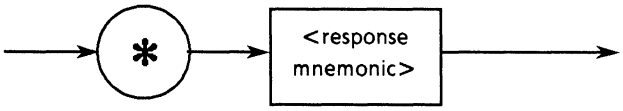
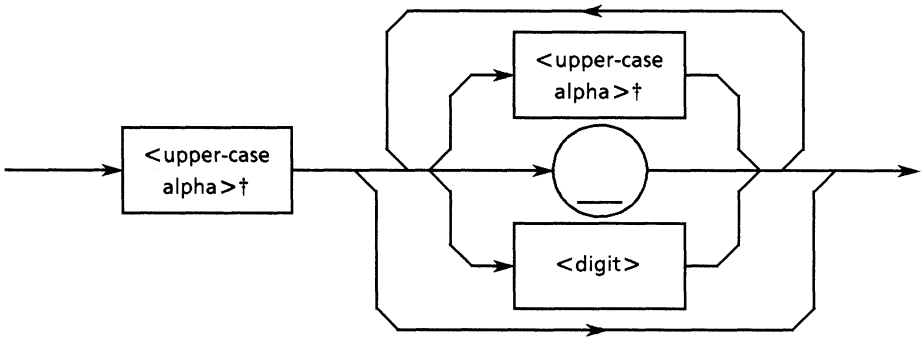
<RESPONSE HEADER>は、フォーマット表現上においては、次の3点を除き、<COMMAND PROGRAM HEADER>に同じです。

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、大文字のみを使用すること。その他は<program mnemonic>に同じ。

② プログラム・ヘッダの前にスペースがおけましたが、レスポンス・ヘッダ前には、スペースはおけません。

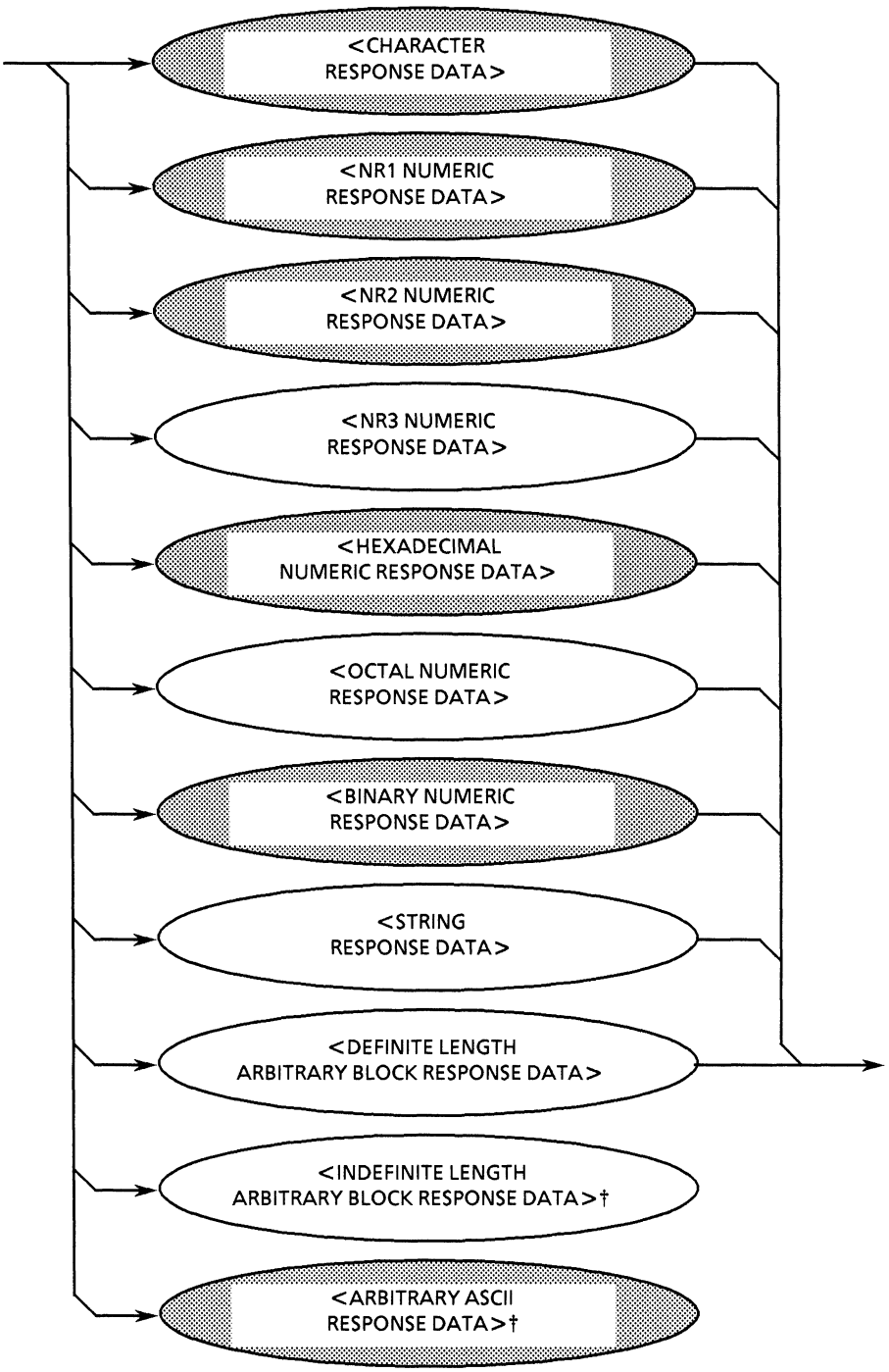
③ プログラム・ヘッダの後には、複数のスペースがおけましたが、レスポンス・ヘッダ後には、1個のスペースしかおけません。

次ページで、<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<program mnemonic>に同じです)

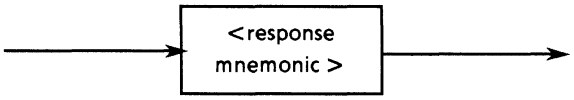
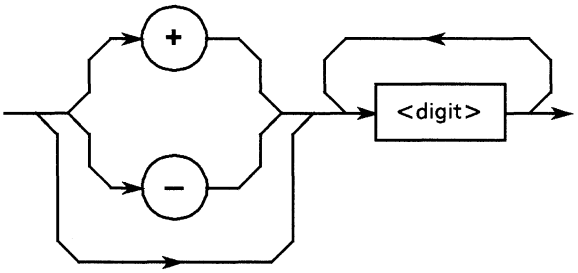
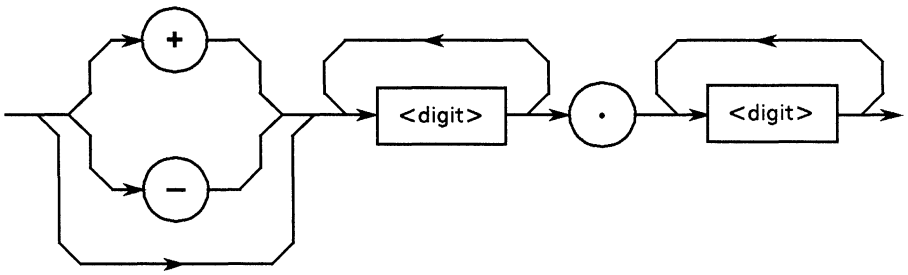
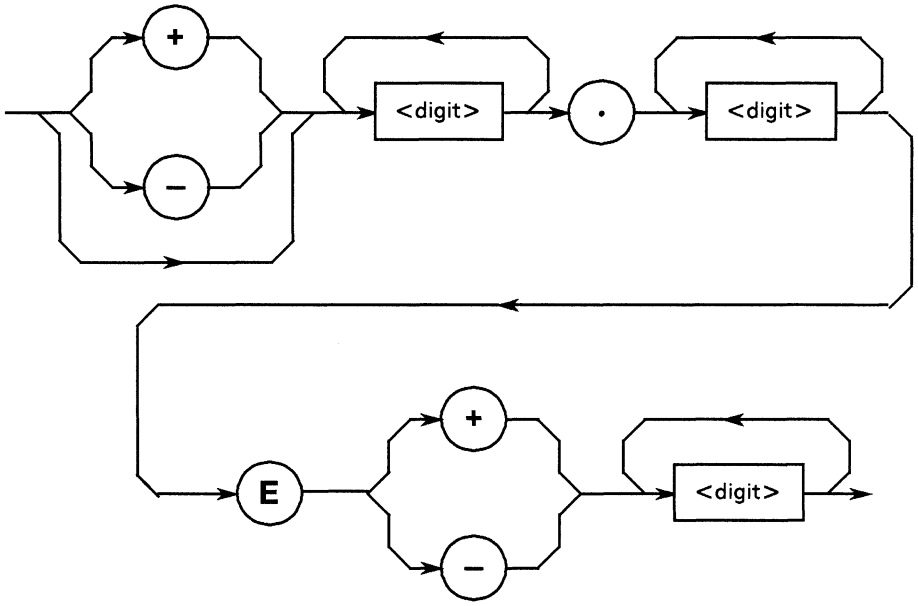
要　素	機　　能
RESPONSE HEADER	ヘッダは、レスポンス・データの機能を表すもので、英大文字から始まる12文字以内の英大文字、数字、アンダーラインのいずれか、それらの組合せから成る<response mnemonic>で、その語義が示されます。  1) <simple response header>は、次のように定義されます。  2) <common response header>は、次のように定義されます。  3) <response mnemonic>は、次のように定義されます。  †<upper-case alpha>　ASCIIコードバイト41~5A (10進数65~90＝英大文字A~Z)

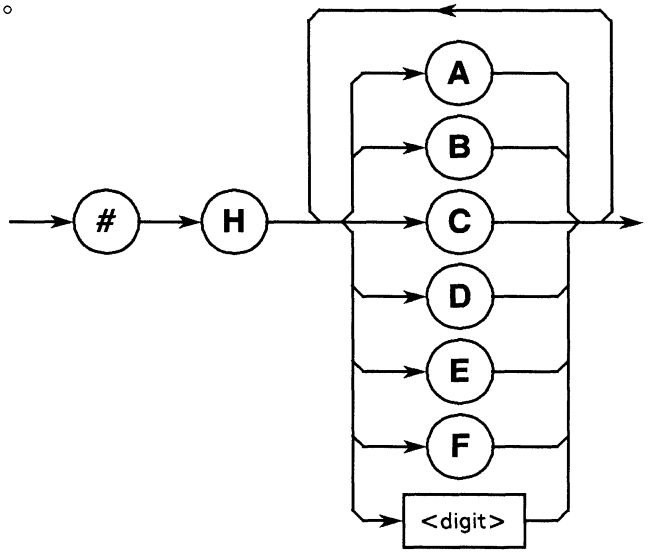
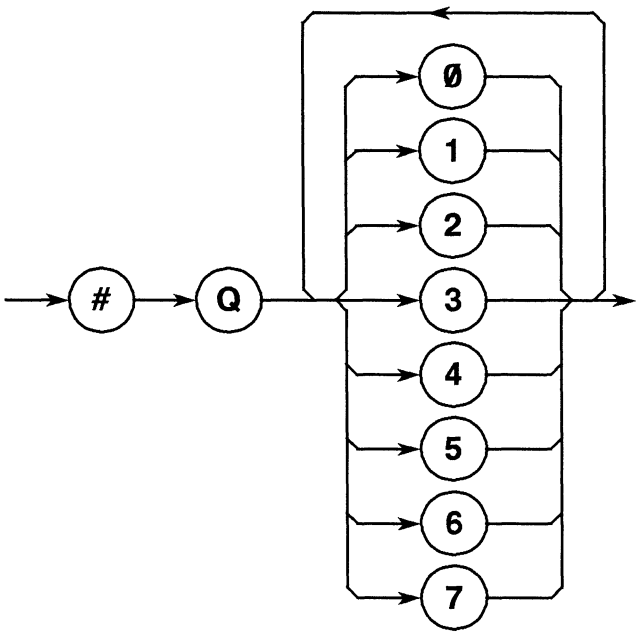
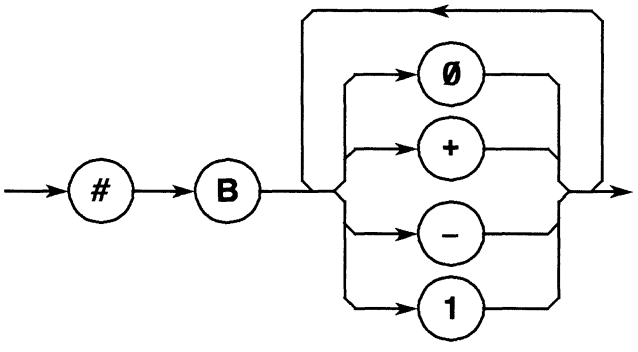
<RESPONSE DATA>

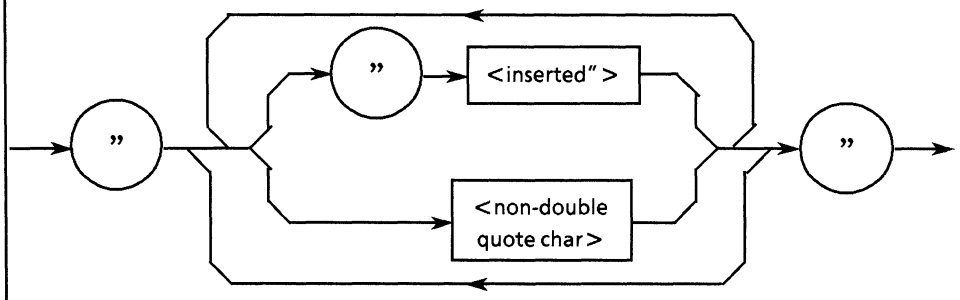
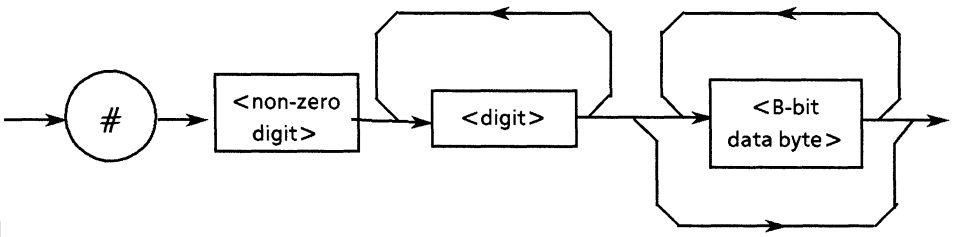
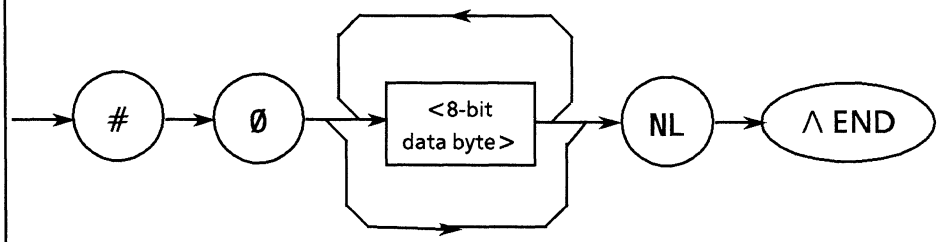
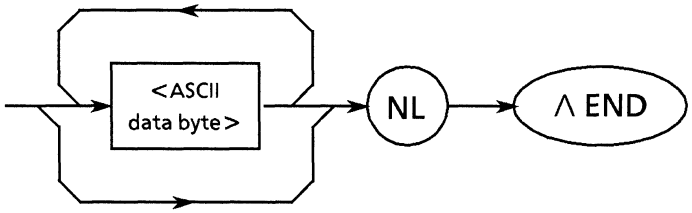
<RESPONSE DATA>は11種類あり、その中でMP1761Cは、アミで囲まれているレスポンスデータをコントローラへ送出します。どのレスポンスデータが返されるかは、問い合わせメッセージによって決定されます。



† <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。

要　素	機　　能
(1) CHARACTER RESPONSE DATA <例> AAT2_AUTO AAT2_MANUAL	<response mnemonic>と同じ文字例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> 12.3E + 4 +12.34E − 5 −12.345 E + 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 

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。 

要　素	機　　能
(8) STRING RESPONSE DATA ＜例＞ “This is a text” “Say,” “Hello” “.”	ASCII 7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、引用符一個に付き、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。 
(10)INDEFINITE 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**#Ø**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。 

（空白）

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合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ラインが偽の時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。
本章での書式、使用例は当社製PACKET Vシリーズ･パーソナルコンピュータによる制御コマンドを適用しています。

目　次

（空白）

7章
共通コマンド

MP1761Cサポート共通コマンドのグループ機能別分類

MP1761Cサポート IEEE 488.2共通コマンドのグループ機能別分類を以下に示します。
サポート対象コマンドについては、アルファベット順にリストされています。

サポートコマンドの分類とリファレンス

前ページで示したMP1761C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説明を下表に示します。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。

グループ	グループ別機能	ニーモニック
システムデータ	GPIB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・形名・シリアル番号などのデータを知ることができます。	*IDN?
内部オペレーション	デバイス内部の制御：① デバイスをレベル3でリセット ② デバイス内部のセルフテストとエラー有無の検知	*RST *TST?
同期	デバイスとコントローラの同期を① サービスリクエスト待ち ② デバイスの出力キュー応答待ち ③ シーケンシャル実行の強制により行います。	*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合せによって知るため、4個のコマンド、5個の問い合わせが用意されています。	*CLS *ESE *ESE? *ESR? *PSC *PSC? *SRE *SRE? *STB?
デバイストリガ	IEEE488.2のGETバスコマンドをデバイスが受信したとき、実行すべきコマンドシーケンスを定義します。	*TRG
オプション情報	実装されているオプション情報を知ることができます。	*OPT?

*CLS　Clear Status Command

(ステータスバイト・レジスタのクリア)

■ 書　式

*CLS

■ 使用例

```
30 WRITE @103:"*CLS"
40 WRITE @103:"DTM△0;CTM△0;*CLS"
```

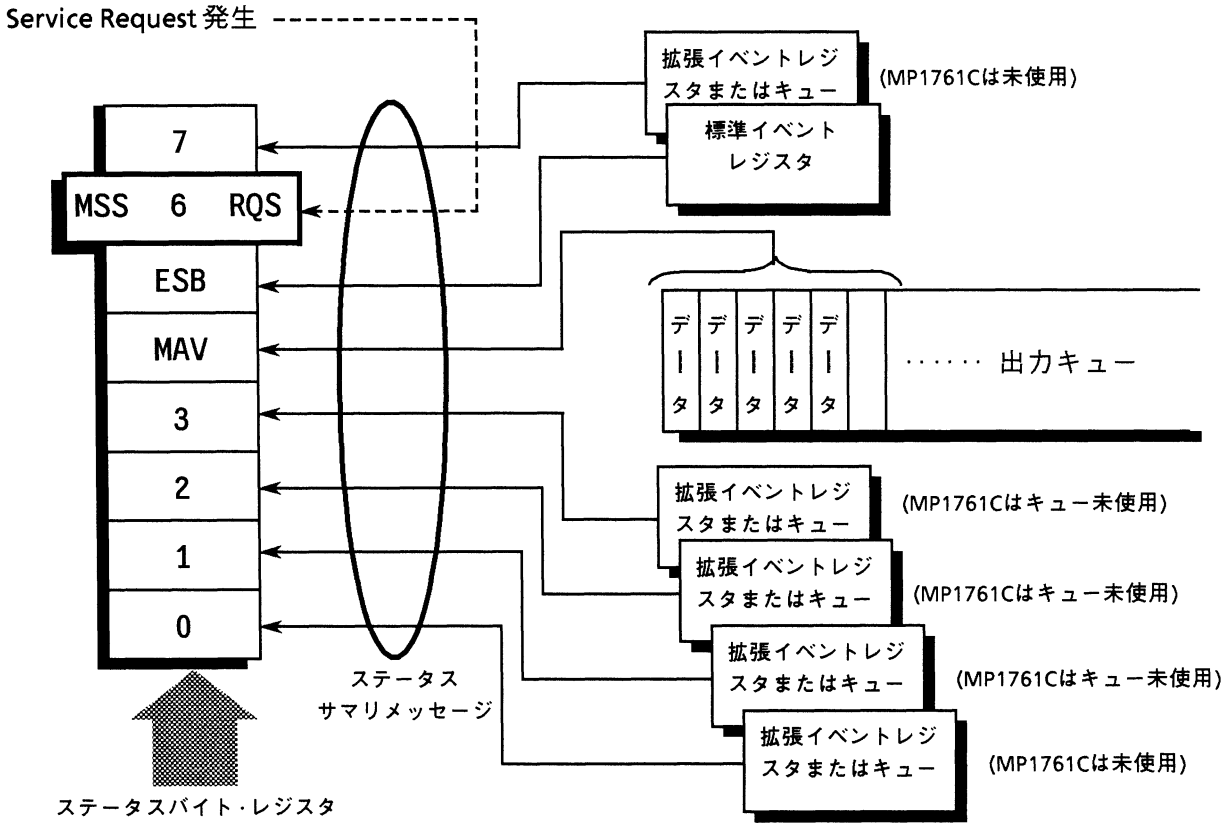
■ 解　説

***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。

出力キューとその **MAV** サマリメッセージも、下例の場合はクリアされます。

```
30 WRITE @103:"DTM△0;CTM△0"
40 WRITE @103:"*CLS;DTM?"
```

すなわち、<PROGRAM MESSAGE TERMINATOR>の後、そして何らかの<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)

■ 書　式

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

本書式において、

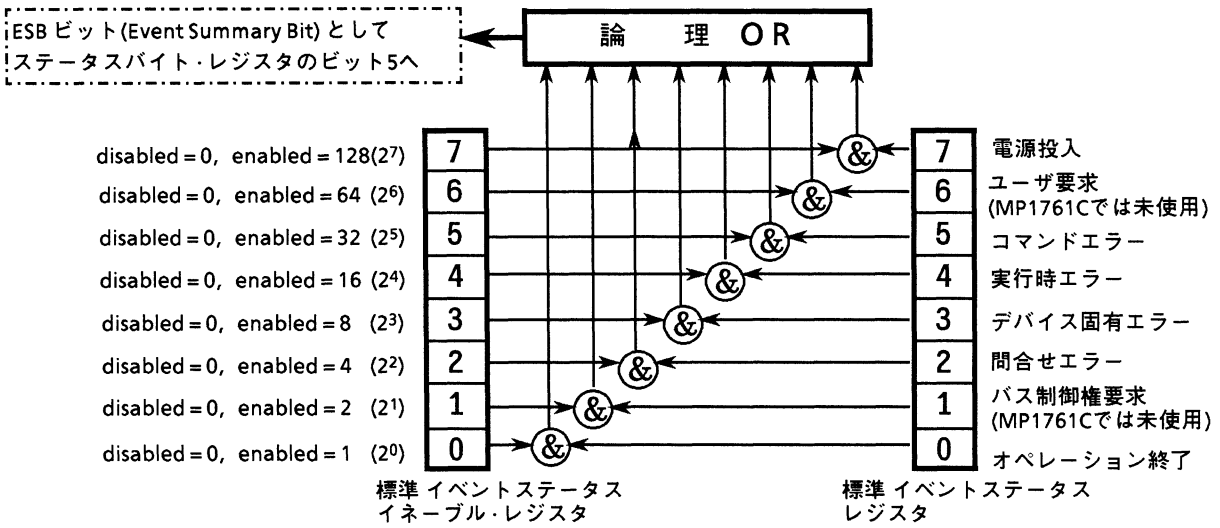
<DECIMAL NUMERIC PROGRAM DATA> = 0～255の整数に丸められ数値(2を底としてバイナリで重み付けされていること)

■ 使用例

```
WRITE @103:"*ESE 20"!
```
イネーブルレジスタのビット2,4をセットします。

■ 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, 1, 2, 3, 4,5, 6, 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 Query

(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)

■ 書　式

*ESE?

■ 使用例

*ESE 20実行後、*ESE?を送ると、20がレスポンスされます。

■ 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。

■ 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)

■ 書　式

*ESR?

■ 使用例

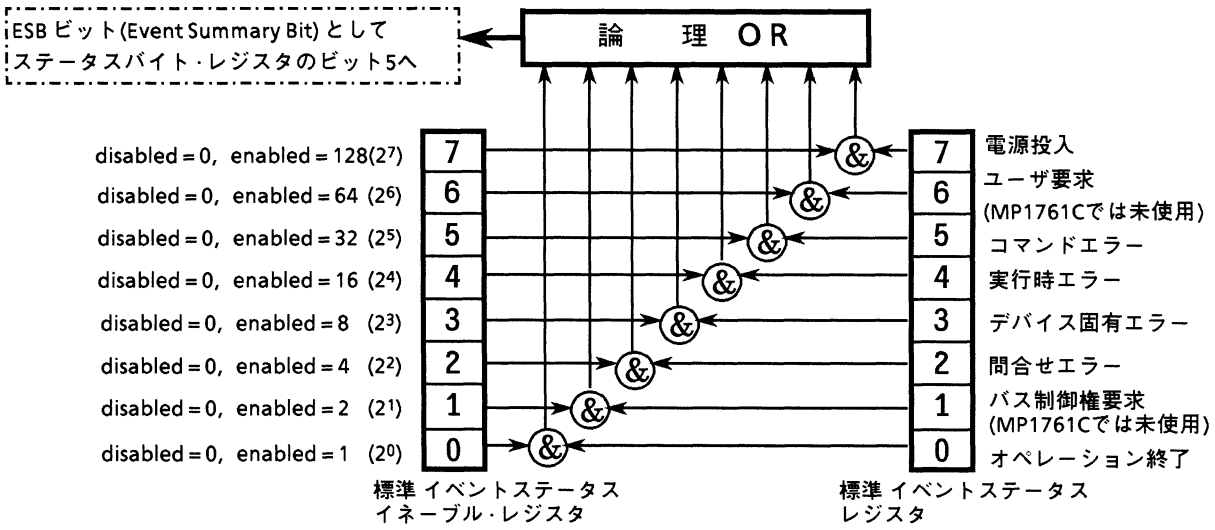
```
30 WRITE @103:"*ESR?"
40 READ @103:STEVET
50 PRINT STEVET
```

■ レスポンスメッセージ

NR1＝0～255

■ 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, 1, 2, 3, 4, 5, 6, 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に対して、標準イベントステータス・イネーブルレジスタによってenabledされているビットに対応するビット桁値の総和がNR1となります。値はレスポンスが読み取られると(たとえば、行40)、このレジスタはクリアされます。



*IDN? Identification Query

(製品のメーカ名・形名等を返す。)

■ 書　式

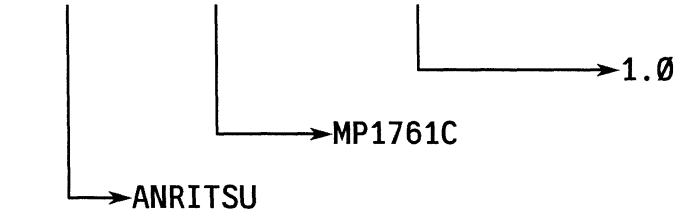
```
*IDN?
```

■ 使用例

```
30 WRITE @103:"*IDN?"
40 READ @103:IDEN$! メーカ名・形名等を格納
```

■ 解　説

製品メーカ名・形名・Ø・バージョンNO.を返します。



形名 MP1761C の製造メーカがアンリツで、ソフトウェアまたはハードウェアのバージョン 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4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。

①フィールド1 ..... 製品メーカ名(当社の場合、ANRITSU)
②フィールド2 ..... 形名
③フィールド3 ..... (常にØ)
④フィールド4 ..... バージョン No.

■ レスポンスメッセージ

上記4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>
解説の例では、
ANRITSU,MP1761C,Ø,1.Øとなります。

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)

■ 書　式

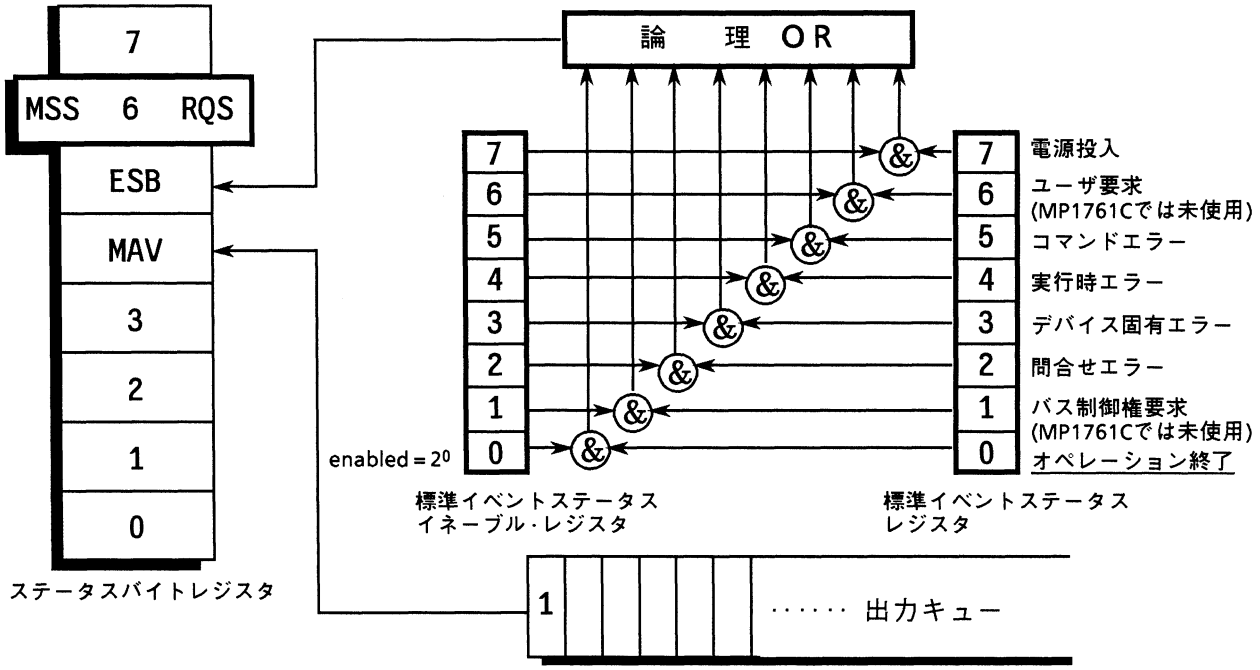
*OPC

■ 使用例

```
WRITE @103:"*OPC"
```

■ 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。



*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。)

■ 書　式

*OPC?

■ 使用例

```
WRITE @103:"*OPC?"
```

■ 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。

■ レスポンスメッセージ

1を<NR1 NUMERIC RESPONSE DATA>で返します。

*OPT? Option Query

(実装されているOption情報を出力)

■ 書　式

*OPT?

■ 使用例

```
30 WRITE @103:"*OPT?"
40 READ:OPTI
```

■ 解　説

*OPT問い合わせは、実装されているOption情報を出力します。

■ レスポンスメッセージ

<ARBITRARY ASCII RESPONSE DATA>
オプションに対応したキャラクタをカンマで区切って返します。

0	:オプション無し
OPT01	:MP1761C-01 内蔵シンセサイザ
OPT03	:MP1761C-03 1/4スピード出力

*PSC　Power-on Status Clear Command

(ステータス報告部のイネーブルレジスタを電源ON時にクリアーするかどうかを決定)

■ 書　式

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

本書式において、　<HEADER> = *PSC
<DECIMAL NUMERIC PROGRAM DATA> = Ø(クリアしない場合)
Øを除く1～255の整数(クリアする場合)

■ 使用例

```
WRITE @1Ø3:"*PSC Ø;*SRE 32;*ESE 128"!クリアしないでSRQ ON
```

■ 解　説

*PSC共通コマンドはステータス報告部のサービスリクエスト・標準イベントステータスおよびパラレルポールの各イネーブルレジスタを電源ON時にクリアするかどうかを決定します。

<DECIMAL NUMERIC PROGRAM DATA>の値がパワーON・ステータス・フラグの真/偽を決定します。この値が整数値Øに丸められる場合は、フラグは偽にセットされますので、イネーブルレジスタはクリアされません。*PSC Øによって、デバイスは電源ON後にSRQを発生することができます。上記の例は、電源ONのイベントをコントローラに伝えることを示しています。

<DECIMAL NUMERIC PROGRAM DATA>の値が0でない整数1～255に丸められる場合は、フラグは真にセットされますのでイネーブルレジスタはクリアされます。*PSC 1を実行すると、パワーONクリアを可能にしますが、SRQの発生は許されません。

プログラムデータの整数値が0～255の範囲外であれば、実行エラーが発生します。

*PSC?　Power-on Status Clear Query

(パワーONステータスフラグの真/偽を返す)

■ 書　式

*PSC?

■ 使用例

```
3Ø WRITE @1Ø3:"*PSC?"
4Ø READ:POWF
```

■ 解　説

*PSC?共通問合せを実行すると、パワーONステータスフラグが真であれば1を返し、偽であればØを返します。

■ レスポンスメッセージ

NR1＝1 (パワーONステータスフラグ真), Ø (パワーONステータスフラグ偽)

*RST　Reset Command

(デバイスをレベル3でリセット)

■ 書　式

*RST

■ 使用例

```
WRITE @103:"*RST"
```

アドレス3番のデバイスのみを初期化

■ 解　説

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

① デバイス固有の機能・状態をそれまでの来歴に関わらず、ある既知の状態に戻します。

② 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。

③ 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。

④ 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 標準イベントステータス・イネーブルレジスタ
⑥ Power-on-status-clear フラグ設定

*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)

■ 書　式

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

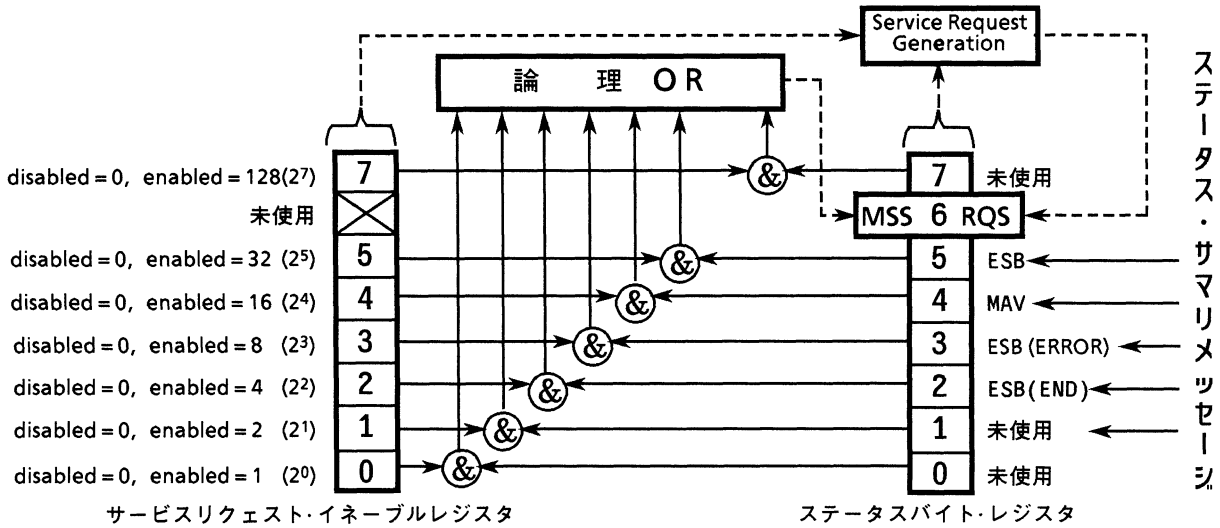
本書式において、
<DECIMAL NUMERIC PROGRAM DATA> = 0～255の整数に丸められ数値 (2を底としてバイナリで重み付けされていること)

■ 使用例

```
WRITE @103:"*SRE 16"!
```
イネーブルレジスタのビット4をセットします。

■ 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, 1, 2, 3, 4, 5, 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)

■ 書　式

*SRE?

■ 使用例

*SRE 16実行後、*SRE?を送ると、16がレスポンスされます。

■ 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。

■ レスポンスメッセージ

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)

■ 書　式

*STB?

■ 使用例

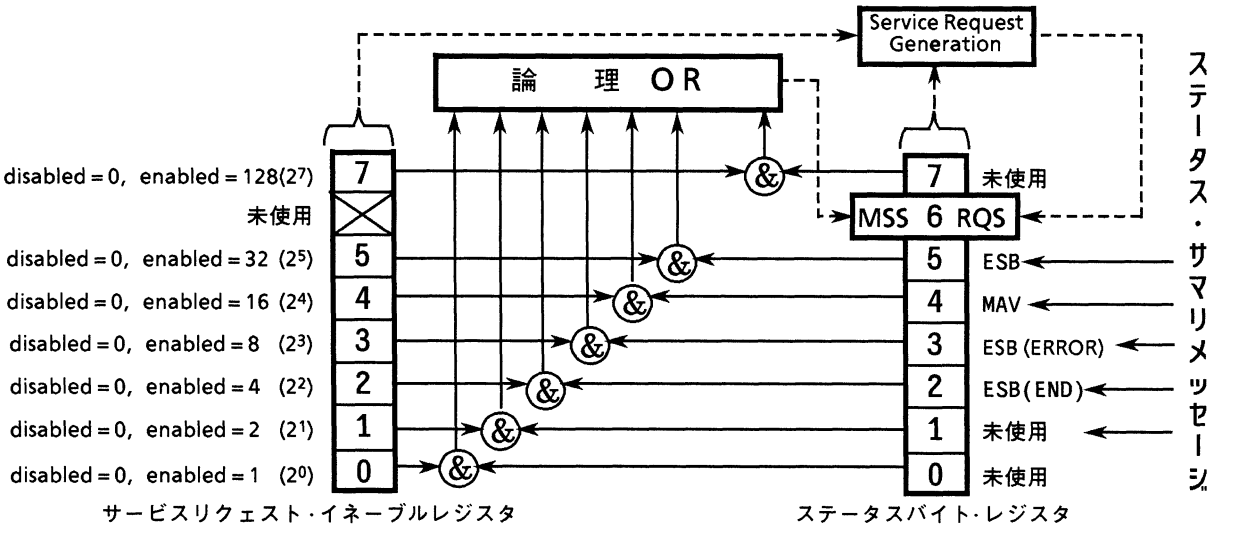
```
30 WRITE @103:"*STB?"
40 READ @103:STBV
50 PRINT STBV
```

■ 解　説

***STB?** 問合せは、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 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。

■ 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。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 (Master Summary Status) ビットは64に重み付けされています。MSSはサービスをリクェストする原因を少なくとも一つあることを示します。下表に、MP1761Cステータスバイト・レジスタの条件を示します。



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	——	0＝使用せず
6	64	MSS	0＝サービスを要求していない。　1＝サービスを要求している。
5	32	ESB	0＝イベントステータス発生せず。　1＝イベントステータス発生
4	16	MAV	0＝出力キューにデータなし。　1＝出力キューにデータあり。
3	8	ESB(ERROR)	0＝イベントステータス発生せず。　1＝イベントステータス発生
2	4	ESB(END)	0＝イベントステータス発生せず。　1＝イベントステータス発生
1	2	——	0＝使用せず
0	1	——	0＝使用せず

*TRG　Trigger Command

(IEEE488.1 の GET − Group Execute Trigger バスコマンドと同一の機能)

■ 書　式

*TRG

■ 使用例

```
WRITE @103:"*TRG"
```

■ 解　説

***TRG** 共通コマンドは、IEEE488.1の **GET** − Group Execute Trigger バスコマンドと同一の機能を提供します。MP1761C の場合には ***DDT** コマンドはサポートしていません。

MP1761C の場合は、***TRG** 共通コマンドを実行しても何も変化はありません。

```
WRITE @103:"*TRG"
```

*TST?　Self-Test Query

(内部セルフテストのエラーの有無を返す)

■ 書　式

*TST?

■ 使用例

```
3Ø WRITE @1Ø3:"*TST?"
4Ø READ @1Ø3:TEST
5Ø PRINT TEST
```

■ 解　説

***TST?** 問合せは、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。

MP1761C では電源 ON 時のセルフテスト結果を出力します。

■ 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＝Ø ..... エラー無しを示します。

NR1≠Ø ..... エラーが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)

■ 書　式

```
*WAI
```

■ 使用例

```
WRITE @103:"*WAI"
```

■ 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合せがデバイスで何かを実行している間でも、次に送られてく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合せをオーバーラップコマンドと言います。

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。

（空白）

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。

IEEE488.2では、IEEE488.1に比べて、より詳しいステータス情報を得るために、共通コマンドおよび共通問い合わせが追加されていますが、これらの詳細については、7章を参照してください。

本章での書式、使用例は当社製PACKET Vシリーズ･パーソナルコンピュータによる制御コマンドを適用しています。

目　次

目　　次(つづき)

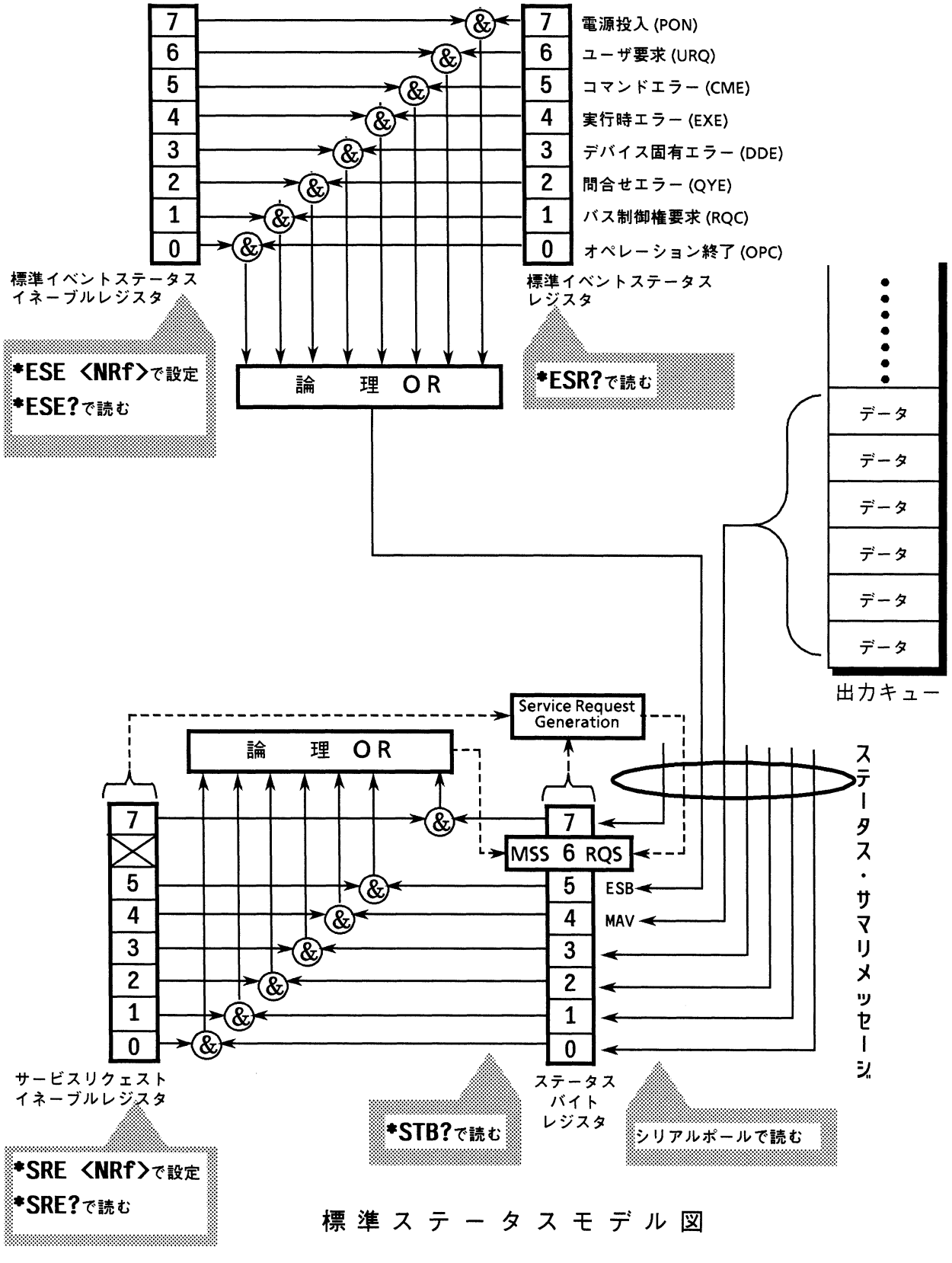
8章
ステータス・ストラクチャー

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したものです。

以下、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、並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。

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。



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	キューモデル
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 (register-model) 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。	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 (queue-model) といいます。キュー構造では、キューにデータがある時だけ対応ビットが1となり、キューが空であれば0となります。

以上、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・レジスタと標準イベントステータス・イネーブルレジスタ
②ステータスバイト・レジスタとサービスリクエスト・イネーブルレジスタ
③出力キュー

標準イベントステータス･レジスタ (Standard Event Status Register)	ステータスバイト･レジスタ (Status Byte Register)	出力キュー (Output Queue)
これ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、8種類の事象 (①電源投入、②ユーザ要求、③コマンドエラー、④実行時エラー、⑤デバイス固有エラー、⑥問合わせエラー、⑦バス制御権要求、⑧オペレーション終了) 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。論理OR出力ビットは、Event Status Bit (ESB) サマリメッセージとして、ステータスバイト・レジスタのbit 5 (DI06) に要約表示されます。	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。この時のステータスバイト・レジスタのbit 6 (DIO7) は、RQS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有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。	これは前記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有ることを知らせるMessage Available (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bit 4 (DI05) に要約表示されます。

ステータスバイト(STB)レジスタ

STB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bit 6に送出されるMaster Summary Status (MSS)について定義しています。

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、STBレジスタのbit 5に現れます。このbit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1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一つもないときにFALSEとなります。

(2) MAVサマリメッセージ

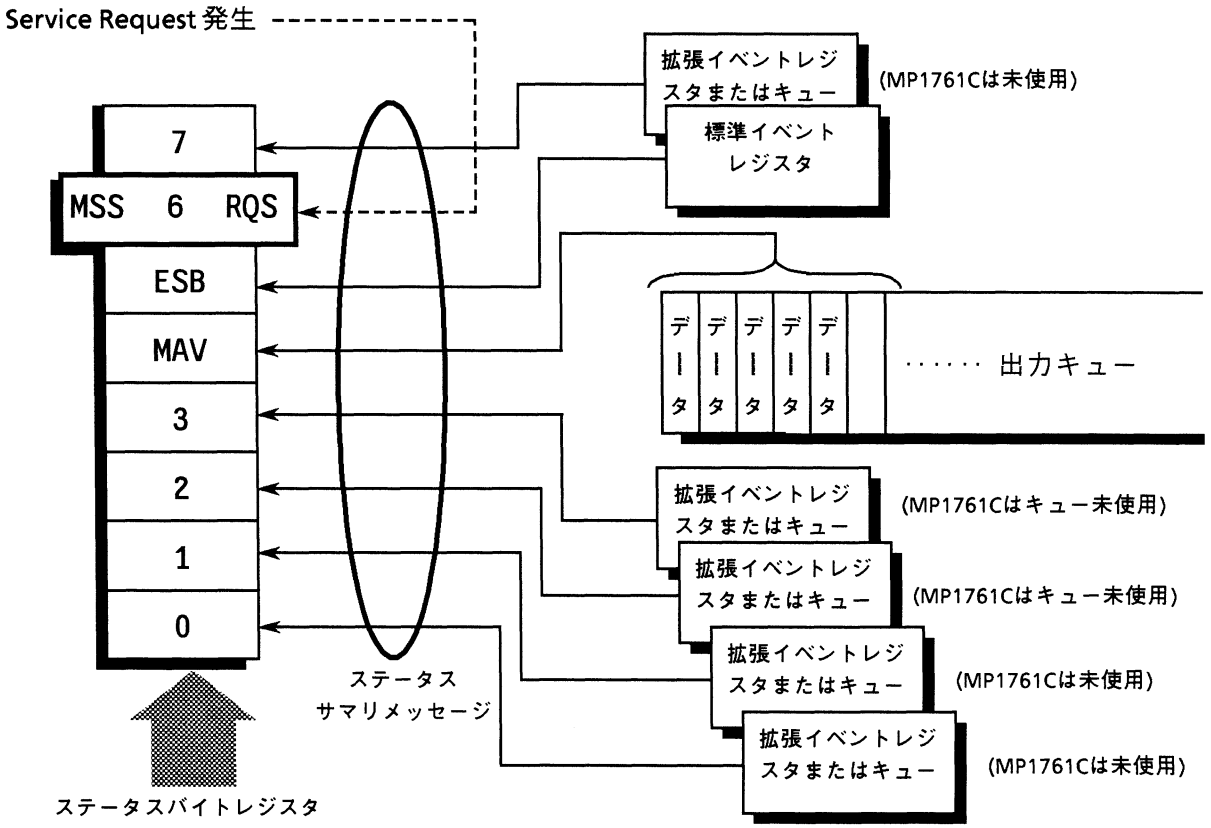
MAV (Message Available)サマリビットは、IEEE488.2で定義されたメッセージで、STBレジスタのbit 4に現れます。このbitの状態は、出力キューが"空"であるかどうかを示します。デバイスがコントローラからレスポンスメッセージの送出要求を受け付ける用意ができているときに、MAVサマリメッセージビットは1 (TRUE)となり、出力キューが"空"のときに0 (FALSE)となります。このメッセージはコントローラとの情報交換に同期を取るために利用されます。たとえば、コントローラがデバイスに問合せコマンドを送り、MAVがTRUEになるのを待つというように使うことができます。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。

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bit 7 (DIO8), bit 3 (DIO4) ~ bit 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、キューにデータの有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一個のキューです。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。

MP1761Cでは下記に示すように、bit 0, bit 1およびbit7を未使用とし、bit 2およびbit 3の3ビットをステータスレジスタのサマリビットとして使っていますので、レジスタモデルは全部で5種類(拡張3種類)、キューモデルは拡張なしで、出力キューの一種類となっています。



STB レジスタの読み出しとクリア

STB 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 メッセージを読みとりますが、bit 6 (位置) に送られる値はその方法によって異なります。
STB 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

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 STB レジスタの内容と MSS (Master Summary Status) 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 STB レジスタの値と **MSS** サマリメッセージの値の総和を表します。STB レジスタの bit 0 ～ 5と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 bit 6位置に現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 Summmary 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に現れますが、シリアルポールに対する応答としては現れません。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 STB レジスタと SRQ イネーブル (SRE) レジスタのビットの組合せによる総合的 OR により構成されます。これを具体的に示すと、結局 MSS は以下のように定義されます。

(STB Register bit0 AND SRE Register bit0)
OR
(STB Register bit1 AND SRE Register bit1)
OR
:
:
(STB Register bit5 AND SRE Register bit5)
OR
(STB Register bit7 AND SRE Register bit7)

MSSの定義において、STBレジスタと、SRQイネーブルレジスタ双方のbit 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bit 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。

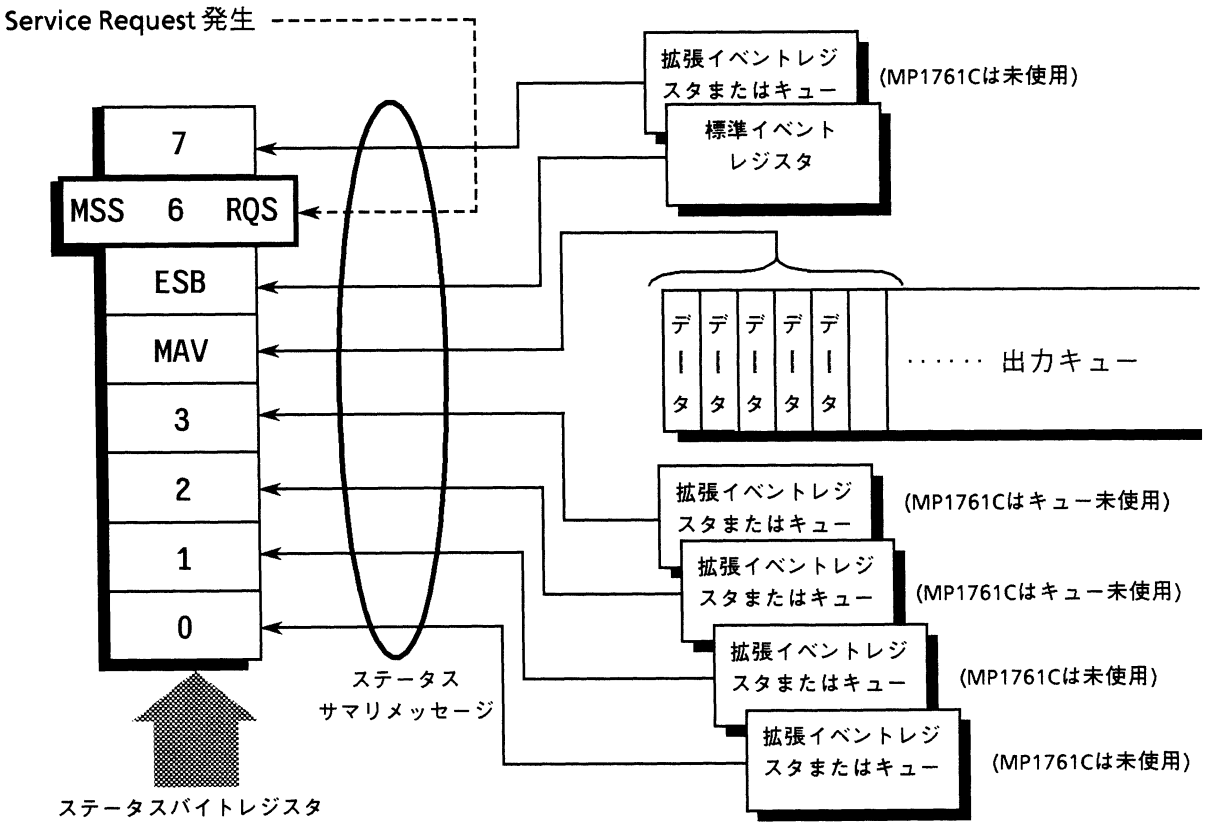
(4) *CLS共通コマンドによるSTBレジスタのクリア

***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。

出力キューとその**MAV**サマリメッセージも、下例の場合はクリアされます。

```
3Ø WRITE @1Ø3:"DTM△Ø;CTM△Ø"
4Ø WRITE @1Ø3:"*CLS;DTM?"
```

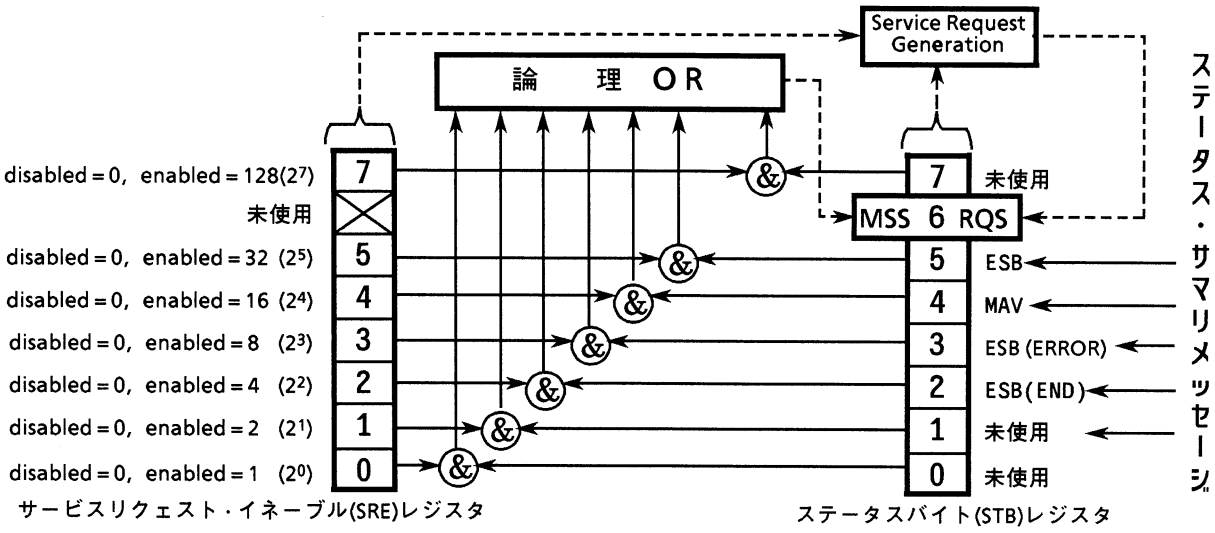
すなわち、<PROGRAM MESSAGE TERMINATOR>の後、そして何らかの<Query MESSAGE UNIT>要素の前に***CLS**コマンドを送出すると、すべてのステータスバイトはクリアされます。この方法により出力キューのすべての未読み出しのメッセージはクリアされ、**MAV**メッセージはFALSEとなります。また、***STB?**に対する応答時、**MSS**メッセージもFALSEになります。なお、各イネーブル･レジスタの設定値については、***CLS**によって影響されません。



SRQのイネーブル

SRQのイネーブルにより、STB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･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が立つ度に、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) SREレジスタの読み出し

SREレジスタの内容は、***SRE?**共通問合せを使って読み出せます。この問合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と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) SREレジスタの更新

SRE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、SREレジスタの各ビット桁値(ウェイト値)の総和を表します。このbit値は、1がenabledの状態を表し、0がdisabledの状態を表します。ビット6の値は常に無視しなければなりません。

(3) SRE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、SREレジスタのクリアをクリアできます。

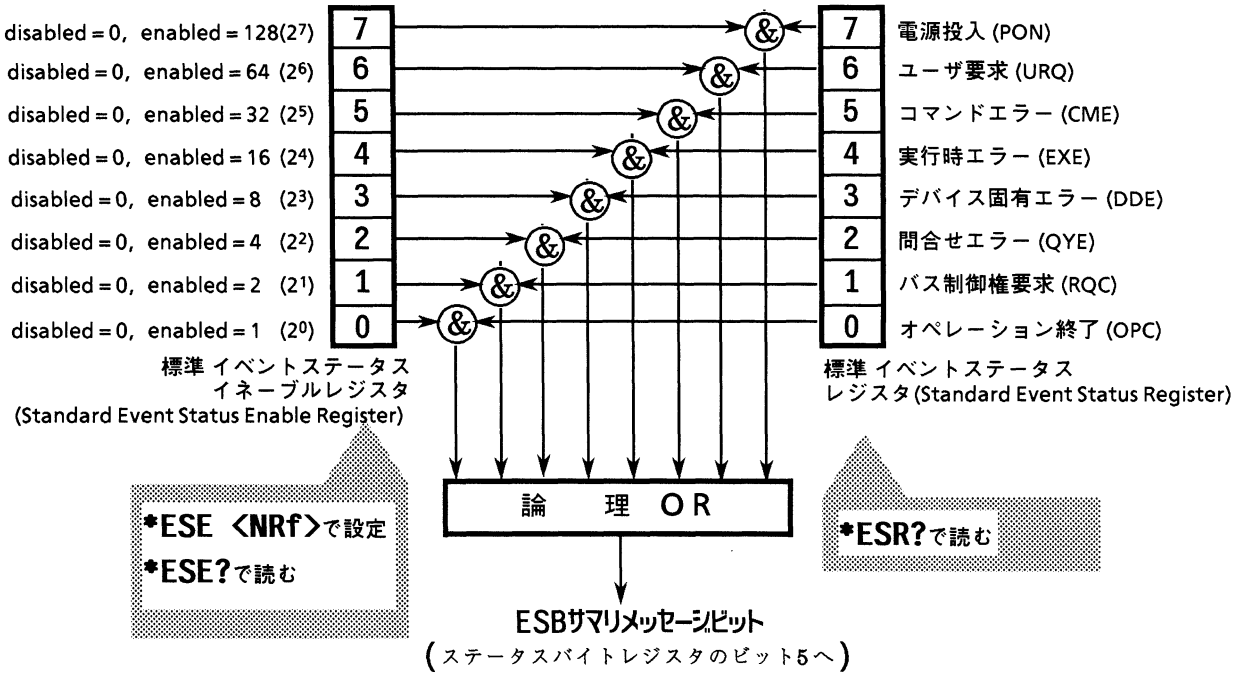
***SRE**共通コマンドの場合は、<DECIMAL NUMERIC PROGRAM DATA>の要素の値を0にすれば、SREレジスタをクリアすることができます。レジスタをクリアすることで、ステータス情報がrsvローカルメッセージを発生することを禁止できますので、この結果サービス要求は発生しなくなります。

MP1761Cは***PSC**コマンドを用意していますので、もし電源投入時pscフラグがTrueの時SREレジスタはクリアされます。

標準イベントステータス·レジスタ

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、ここでは、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。



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 へと変化した。
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。MP1761C 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 コマンドを受信した。(ヘッダパラメータの文法のミス、パラメータの個数ミス)
4	実行時エラー (EXE–Execution Error)	文法に問題はないが、実行できないプログラムメッセージを受信した。(パラメータの範囲外)
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た。 (現在のデバイスの状態により、実行不可)
2	問合せエラー (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。

ビット	イベント名	説　　明
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る。MP1761C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る。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。

問合せエラーの詳細

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、問合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に問合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、問合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に問合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	コントローラが問合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。8と同じように問合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	問合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、まだ問合せメッセージが入力され、それに伴いレスポンスメッセージを出力しなければいけない時、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、問合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。

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

読み取り	***ESR?** 共通問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き、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。 ③ ***ESR?** 問合せコマンドに対して、イベントが読み込まれた。

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

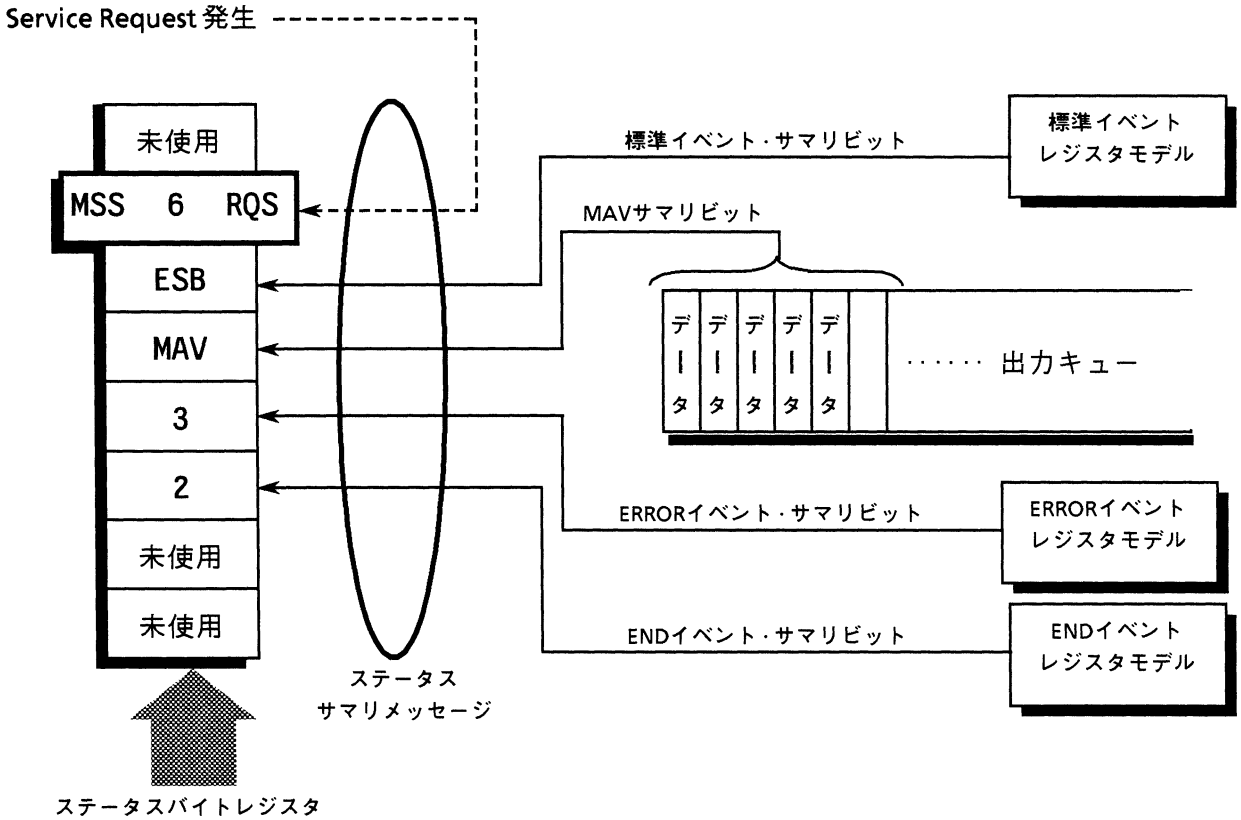
読み取り	***ESE?** 共通問合わせにより非破壊的に読み取られます。 すなわち、読み取られた後も、クリアされません。レスポンスメッセージ は、2進数の重みを付けて2進-10進変換された<NR1>で返されます。
書き込み	***ESE** 共通コマンドによって書き込まれます。レジスタのビット0～7は、それ ぞれ1, 2, 4, 8, 16, 32, 64, 128に重み付けされていますので、 書き込みデータは、その中から希望のビット桁値を合計した <10進数値プログラムデータ>で送ります。
クリア	次の場合にクリアされます。 ① データ値0の***ESE** コマンドを受信 ② 電源 ON ステータス･クリア･フラグが真の状態での電源 ON 時。 標準イベントステータス･イネーブルレジスタは、 下記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

拡張イベントステータス・レジスタ

IEEE488.2対応機種において、これまで説明したイネーブルレジスタを含むステータスバイト・レジスタおよび標準イベントステータス・レジスタの各レジスタモデルは、必須のものとなっています。

IEEE488.2では、ステータスバイト・レジスタのbit 7 (DIO8), bit 3 (DIO4) ~ bit 0 (DIO1)を拡張レジスタモデルまたは拡張キューモデルから供給されるステータスサマリビット用に割当てています。

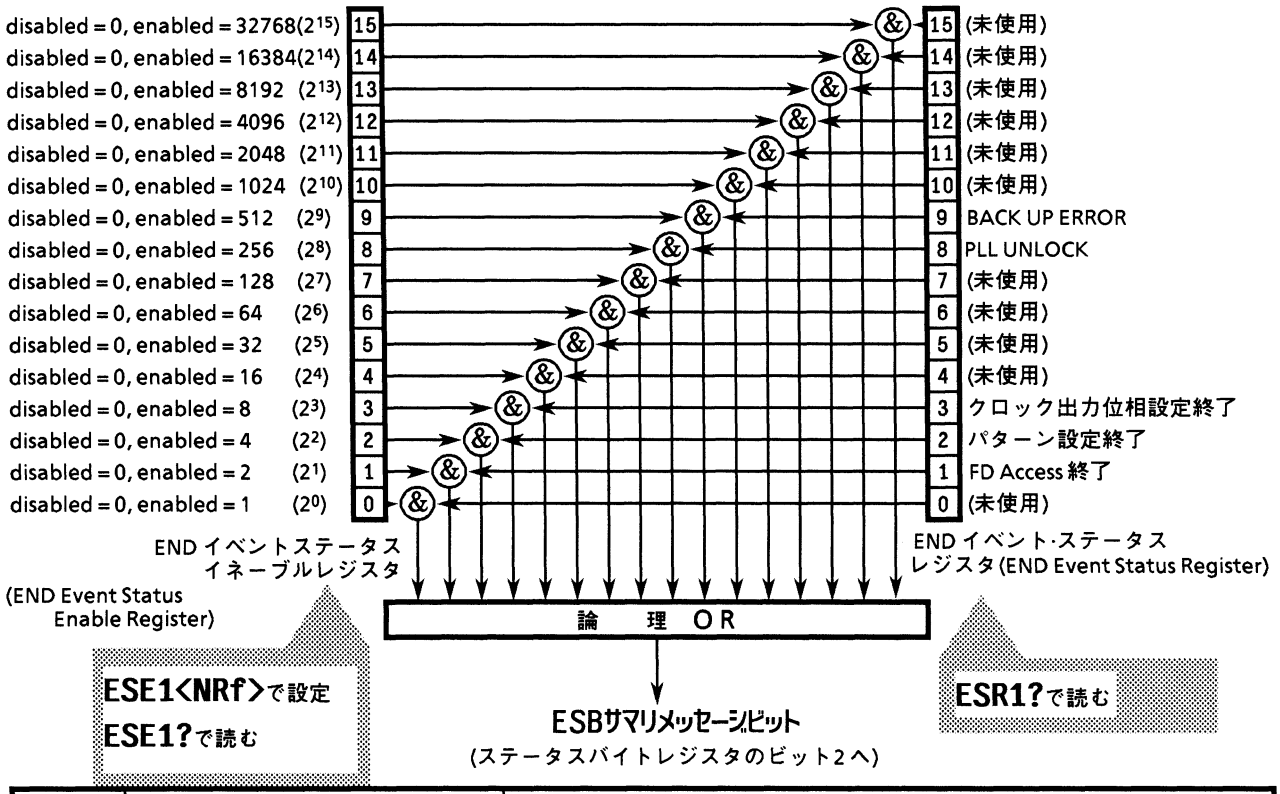
MP1761Cでは、下記に示すように、bit 0, bit 1, bit 7を未使用とし、bit 2, bit 3の2ビットを、拡張レジスタモデルから供給されるステータスサマリビット用として、ERRORおよびENDサマリビットに割当てています。したがって、キューモデルは拡張されてないので、出力キューの一種類となっています。



以下、END, ERROR各々の拡張イベントレジスタ・モデルのビットの定義、読み取り・書き込み・クリアについて説明します。

ENDイベントステータス･レジスタのビット定義

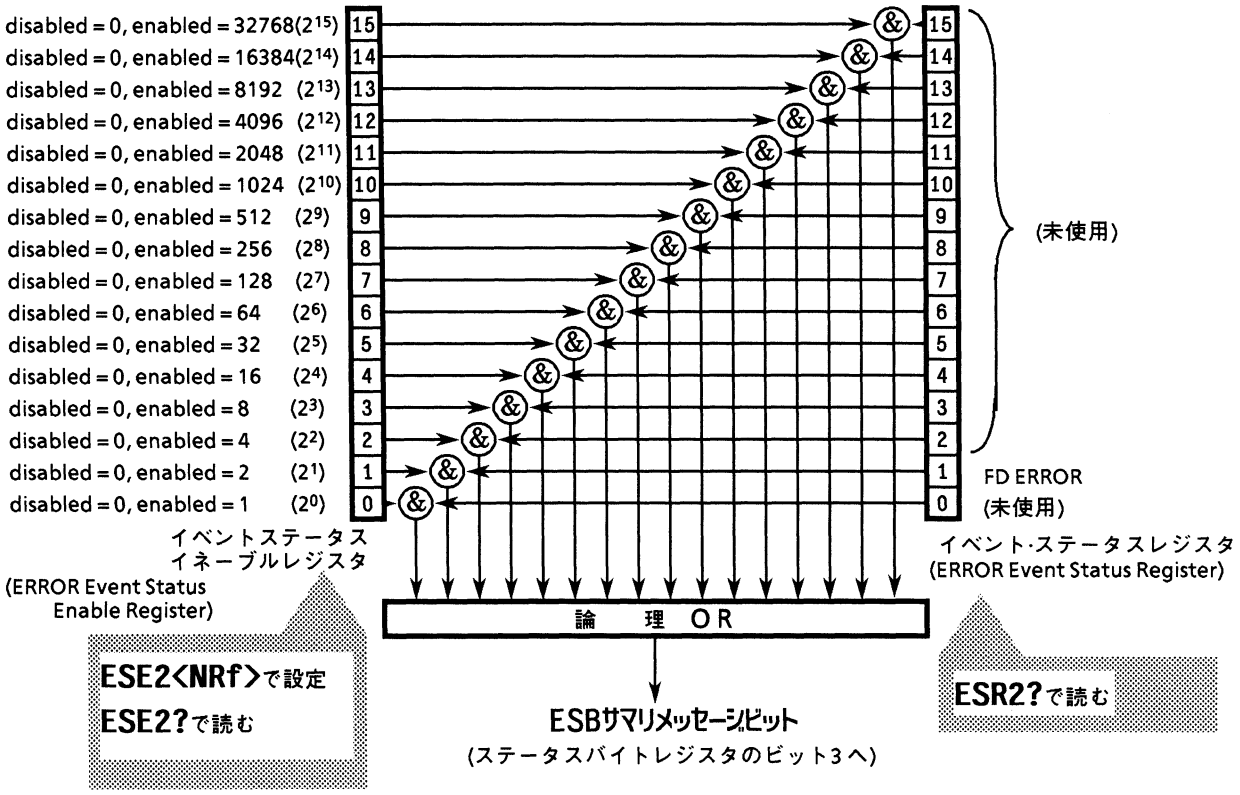
下記に、ENDイベントステータス･レジスタモデルの動作、イベントビット名およびその意味について説明します。



ビット	イベント名	説　明
15	(未使用)	(未使用)
14	(未使用)	(未使用)
13	(未使用)	(未使用)
12	(未使用)	(未使用)
11	(未使用)	(未使用)
10	(未使用)	(未使用)
9	BACK UP ERROR	バックアップデータに誤りを検出した。
8	PLL UNLOCK	内蔵シンセサイザのPLLロック外れを検出した(オプション01実装時のみ)。
7	(未使用)	(未使用)
6	(未使用)	(未使用)
5	(未使用)	(未使用)
4	(未使用)	(未使用)
3	クロック出力位相設定終了	クロック出力位相のサーボ回路がBUSYからREADY状態に変化した。
2	パターン設定終了	プログラマブル・パターンの設定が終了した。
1	FD Access終了	フロッピーのアクセスが終了した。
0	(未使用)	(未使用)

ERROR イベントステータス･レジスタのビット定義

下記に、ERROR イベントステータス･レジスタモデルの動作、イベントビット名およびその意味について説明します。



ビット	イベント名	説　　明
15	(未使用)	(未使用)
14	(未使用)	(未使用)
13	(未使用)	(未使用)
12	(未使用)	(未使用)
11	(未使用)	(未使用)
10	(未使用)	(未使用)
9	(未使用)	(未使用)
8	(未使用)	(未使用)
7	(未使用)	(未使用)
6	(未使用)	(未使用)
5	(未使用)	(未使用)
4	(未使用)	(未使用)
3	(未使用)	(未使用)
2	(未使用)	(未使用)
1	FD ERROR	FD異常発生
0	(未使用)	(未使用)

拡張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

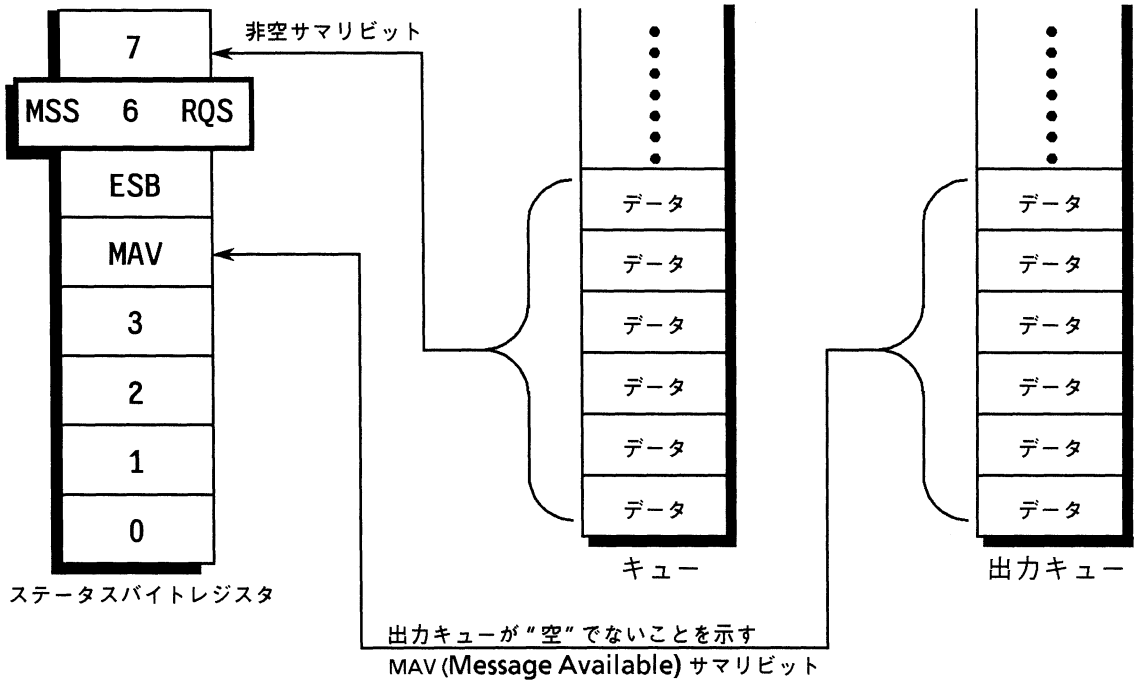
読み取り	問合わせにより破壊的に読み取られます。すなわち、読み取られた後、クリアされます。END, ERRORの各々のイベントステータス･レジスタに対しては、ESR1?, ESR2?の問い合わせで読み取ります。その値はイベントビットに2進数の重みを付けて10進数変換した<NR1>です。
書き込み	クリアすることを除き、外部から書き込みは行えません。
クリア	次の場合にクリアされます。 ① *CLSコマンド受信 ② 電源ONステータス･クリア･フラグが真ならば、電源ONのとき。 ③ 問合せコマンドに対して、イベントが読み込まれた。

拡張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

読み取り	問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。END, ERRORの各々のイベントステータス･イネーブル･レジスタに対しては、ESE1?, ESE2?の問い合わせで読み取ります。その値は、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	END, ERRORの各々のイベントステータス･イネーブル･レジスタに対しては、ESE1, ESE2プログラムコマンドによって書き込まれます。 レジスタのビット0～7は、それぞれ1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768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総和した<10進数値プログラムデータ>で送ります。
クリア	次の場合にクリアされます。 ① END, ERRORの各々のイベントステータス･イネーブル･レジスタに対してはデータ値0のESE1, ESE2, プログラムコマンドを受信 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。 拡張イベントステータス･イネーブルレジスタは、 下記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

キュー (待ち行列) モデル

下図右側に、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) にある時、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のいず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。

当社では、ステータスバイト･レジスタのビット7を『キュー』からのサマリ･メッセージビット用に当てていますが、『出力キュー』で間に合う場合は、特に『キュー』を使用していませんので、ステータスバイト･レジスタのビット7は未使用となっています。

ここでは、『出力キュー』と一般のキューを、次ページで比較して示します。

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。
読出し	IEEE 488.2で定義された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、問合せによって決定されます。	装置固有の問合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。 IEEE 488.2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時にはTRUE (1)、空の時には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。	キューが空でない時にはTRUE (1)、空の時にはFALSE (0)となります。
クリア	次のいずれかが発生した時にはクリアされます。 ① キューの中のすべてのアイテムが読み取られた。 ② 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。 ③ 電源投入で、ponがTrueとなった。	次のいずれかが発生した時にはクリアされます。 ① キューの中のすべてのアイテムが読み取られた。 ② *CLSコマンドを受信した。 ③ 装置固有のこの他の手段。

デバイスとコントローラ間の同期テクニック

デバイスとコントローラ間の同期をとるには3つの方法があります。

① シーケンシャル実行の制御 ..... (***WAI?** コマンドによる)
② デバイスの出力キュー応答待ち ..... (***OPC?** 問合せによる)
③ サービスリクエスト待ち ..... (***OPC** コマンド / ***OPC?** 問合せによる)

シーケンシャル実行の強制

デバイス固有のコマンドは、シーケンシャルコマンドとオーバラップコマンドの2つのタイプに分けられます。

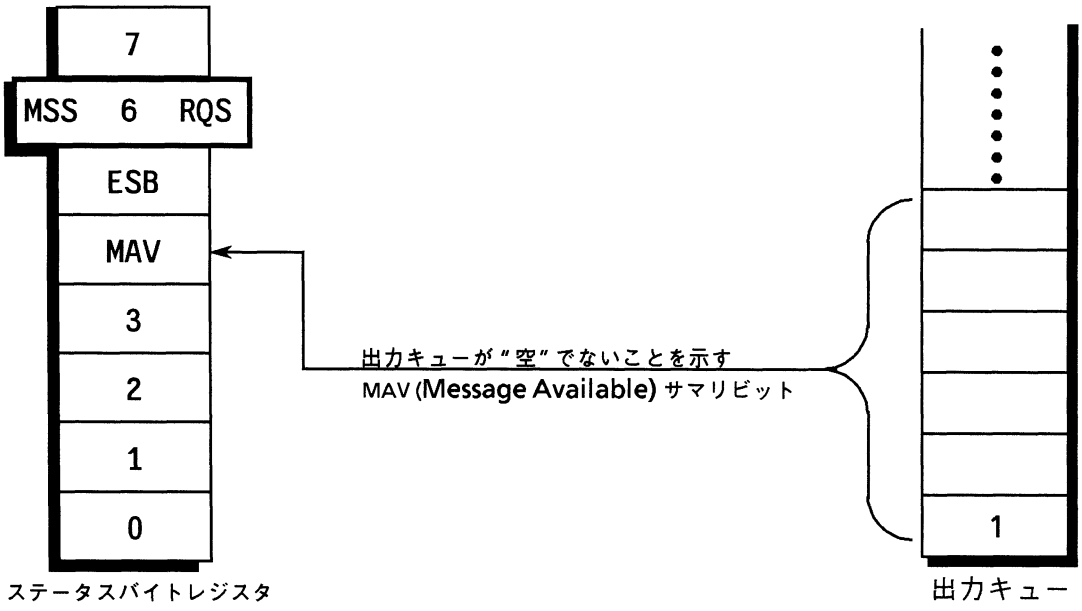
- シーケンシャルコマンド　　コントローラから送られてくるコマンドまたは問合せで、デバイスが何かを実行している間、次に送られてくるコマンドの実行を開始しないコマンドまたは問合せをシーケンシャルコマンドと言います。
- オーバラップコマンド　　コントローラから送られてくるコマンドまたは問合せで、デバイスが何かを実行している間でも、次に送られてくるコマンドの実行を開始するコマンドまたは問合せをオーバラップコマンドと言います。

シーケンシャル実行の強制とは、本来オーバラップコマンドとして振る舞うコマンドを強制的にシーケンシャルに動作させ、一つの処理終了を合図に、次の処理を行う同期テクニックを言います。この同期テクニックには、***WAI** コマンドを使用します。

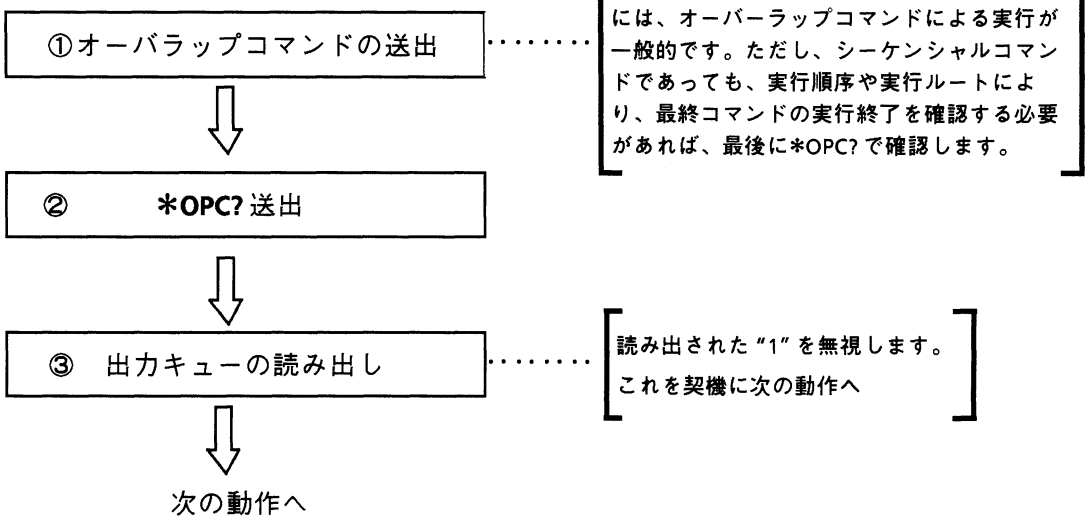
デバイスの出力キュー応答待ち

＊OPC? 問合せを実行すると、ペンディング中の、指定した動作をデバイスがすべて終了したら、出力キューに **"1"** を立て、**MAV** サマリメッセージを発生させます。

出力キュー応答待ちによる同期とは、上記の結果によって得られた出力キューの中のデータ**1**を読み出すか、または**MAV**サマリメッセージビットを読み出すことによって、デバイスとコントローラ間の同期をとる方法です。
MAVサマリメッセージビットの利用は、『サービスリクエスト待ちによる方法』に属するため、そこで説明します。ここでは、『出力キューの読み出しによる方法』について説明します。



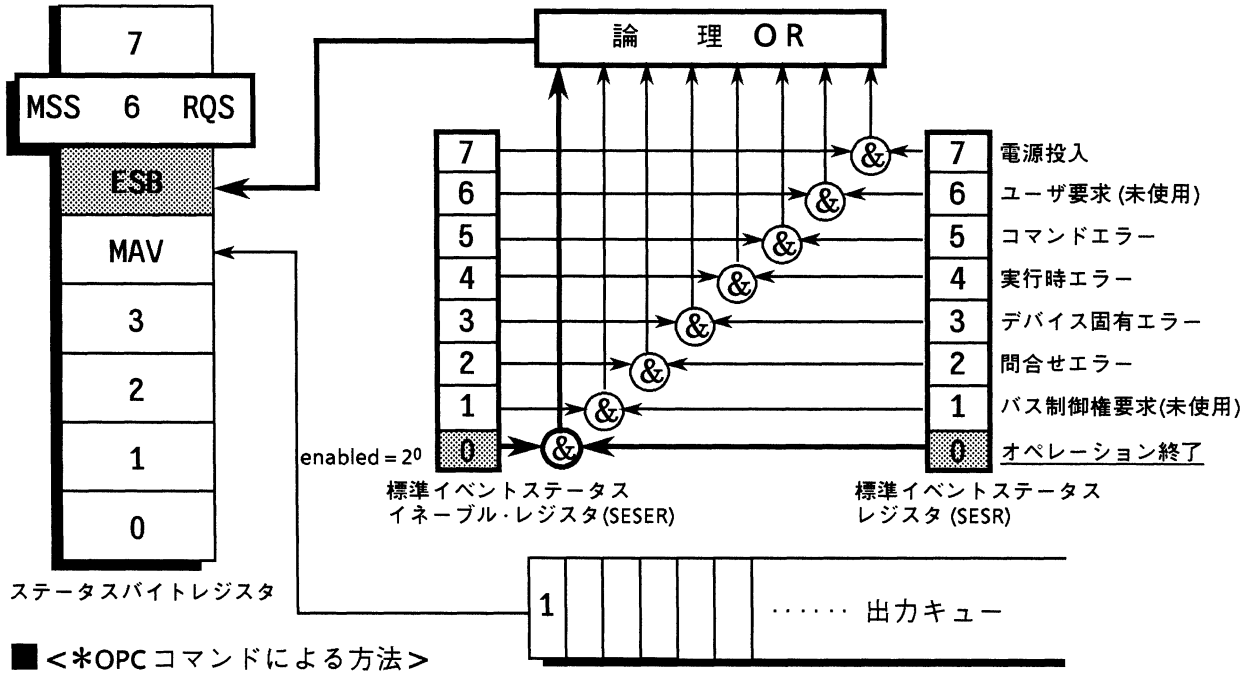
<出力キュー読み出し手順>



サービスリクエスト待ち

サービスリクエスト待ちとは、通常の処理を行っているコントローラがデバイスのSRQ信号による割り込みのため、その通常の処理を一時中断し、デバイスのステータスメッセージの条件に対応した処理を行うことです。

本来の割り込み処理は、コントローラが現在どんな処理をやっているかにはおかまいなくデバイス側が要求してくるものですが、デバイスとコントローラ間の同期テクニックにおいては、あらかじめコントローラがデバイスの動作が終了したかどうかを確認するため、***OPC**コマンドまたは***OPC?**問合せをデバイスに送ります。そして、動作終了イベントによるSRQ信号を待つ間、コントローラは、何か別の有用な仕事を進め、動作終了イベントを検出すると、指定された処理をするというのが、サービスリクエスト待ちによる同期テクニックです。



■<*OPCコマンドによる方法>

①標準イベントステータス･イネーブルレジスタの2^0ビットをイネーブルにする。　WRITE @1Ø3:"*ESE 1"

⇩

②サービスリクエスト･イネーブルレジスタの2^5ビットをイネーブルにする。　WRITE @1Ø3:"*SRE 32"

⇩

③デバイスに指定した動作を実行させる。　WRITE @1Ø3:<PROGRAM MESSAGE >

⇩

④*OPCコマンド実行　WRITE @1Ø3:"*OPC"

⇩

⑤SRQ割り込み待ち (ESBサマリメッセージ)　····ステータスバイトの値は$2^6+2^5=96$

■ <＊OPC? 問合せによる方法>

①サービスリクエスト･イネーブルレジスタの2^4ビットをイネーブルにする。　WRITE @1Ø3:"＊SRE 16"

⇩

②デバイスに指定した動作を実行させる。　WRITE @1Ø3:<PROGRAM MESSAGE>

⇩

③＊OPC? 問合せの送出 (②の動作終了まで待つ)　WRITE @1Ø3:"＊OPC?"

⇩

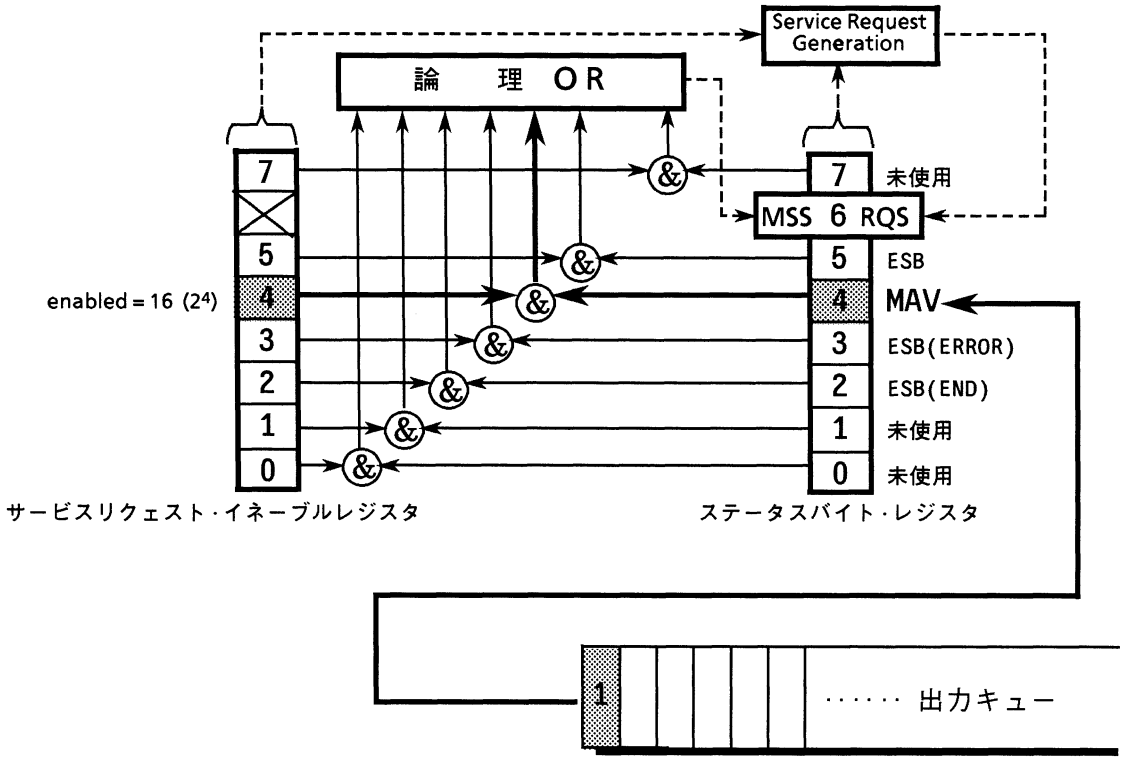
④出力キューの ASCII キャラクタ "1" を読み込み、これを捨てる。

⇩

⑤SRQ 割り込み待ち (MAV サマリメッセージ)　・・・・ステータスバイトの値は$2^6+2^4=80$

⇩

次の動作へ



9 章
デバイス・メッセージの詳細

この章では、各デバイス・メッセージの一覧表と各デバイス・メッセージの詳細について説明します。

本書での書式、使用例はヒューレット・パッカード社 HP9000シリーズの HP-BASIC で説明しています。

目　次

（空白）

9章
デバイス・メッセージの詳細

この章では、各デバイスメッセージをグループごとに説明します。各グループは、MP1761Cの正背面パネルに対応しているので、設定または要求の内容によりグループを選択して参照します。

デバイス・メッセージ一覧表

MP1761Cで規定される、コントロール・メッセージおよびデータ・リクエスト・メッセージについて一覧表の形で説明します。

各コマンドについての詳細は一覧表内のページにしたがって「デバイス・メッセージの詳細説明」の所で確認して下さい。

デバイス・メッセージ一覧表(アルファベット順)

各コントロール・メッセージおよびデータ・リクエスト・メッセージについてアルファベット順にした一覧表を表9-1に示します。

表9-1　デバイス・メッセージ一覧表(アルファベット順)

機　能　項　目	コントロール・メッセージ		データ・リクエスト・メッセージ	デバイス・メッセージ詳細	
	ヘッダ部	数値データ部	ヘッダ部	セクション	ページ
ページ数/パターン同期トリガ位置	ADR	NR1形式	ADR?	PATTERN	P9-47
パターン・データ・プリセット(全ページ・全ビット)	ALL	NR1形式	−	PATTERN	P9-51
オルタネート・パターンA/B表示切替	ALT	NR1形式	ALT?	PATTERN	P9-41
オルタネート・パターンA/B切替信号選択	APS	NR1形式	APS?	その他	P9-82
パターン・ビット	BIT	NR1形式 HEX形式	BIT?	PATTERN	P9-49
クロック1出力振幅	CAP	NR2形式	CAP?	OUTPUT	P9-70
クロック1出力遅延時間	CDL	NR2形式	CDL?	OUTPUT	P9-69
クロック1出力終端電圧	CTM	NR1形式	CTM?	OUTPUT	P9-58
クロック1出力オフセット	COS	NR2形式	COS?	OUTPUT	P9-71
データ出力振幅	DAP	NR2形式	DAP?	OUTPUT	P9-60
データ/$\overline{\text{データ}}$表示切替	DDS	NR1形式	DDS?	OUTPUT	P9-74
フロッピー・データ・デリート	DEL	NR1形式	−	MEMORY	P9-26
データ長	DLN	NR1形式	DLN?	PATTERN	P9-44
ディレイ状態	−	−	DLY?	その他	P9-89
データ出力オフセット	DOS	NR2形式	DOS?	OUTPUT	P9-62
データ出力終端電圧	DTM	NR1形式	DTM?	OUTPUT	P9-57
誤り挿入	EAD	NR1形式	EAD?	PATTERN	P9-42
誤り挿入チャンネル	ECH	NR1形式	ECH?	その他	P9-79
外部誤り挿入	EEI	NR1形式	EEI?	その他	P9-81
フロッピー・フォーマット	FDF	−	−	MEMORY	P9-30
ファイルNo./ディレクトリモード切替	FIL	NR1形式	FIL?	MEMORY	P9-24
フロッピー・エラーメッセージ	−	−	FDE?	MEMORY	P9-35
メモリ・フロッピーモード	−	−	FMD?	MEMORY	P9-33
内部クロック周波数	FRQ	NR1形式	FRQ?	INTERNAL CLOCK	P9-20
ファイル内容の検索	−	−	FSH?	MEMORY	P9-31

表9-1　デバイス・メッセージ一覧表(アルファベット順: つづき)

機　能　項　目	コントロール・メッセージ		データ・リクエスト・メッセージ	デバイス・メッセージ詳細	
	ヘッダ部	数値データ部	ヘッダ部	セクション	ページ
初期化	INI	−	−	その他	P9-83
パターン論理	LGC	NR1形式	LGC?	PATTERN	P9-37
オルタネート・A/B・ループ回数	LPT	NR1形式	LPT?	PATTERN	P9-43
フロッピー・アクセス状態	−	−	MAC?	MEMORY	P9-34
メモリ・モード切替	MEM	NR1形式	MEM?	MEMORY	P9-29
PRBSマーク率	MRK	NR1形式	MRK?	PATTERN	P9-40
$\overline{\text{データ}}$出力振幅	NAP	NR2形式	NAP?	OUTPUT	P9-61
$\overline{\text{データ}}$出力オフセット	NOS	NR2形式	NOS?	OUTPUT	P9-67
オフセット基準値	OFS	NR1形式	OFS?	OUTPUT	P9-59
出力オン/オフ	OON	NR1形式	OON?	OUTPUT	P9-73
ページ数/パターン同期トリガ位置	PAG	NR1形式	PAG?	PATTERN	P9-47
PLLロック状態	−	−	PLL?	その他	P9-86
ページ数/パターン同期トリガ位置表示切替	PPD	NR1形式	PPD?	PATTERN	P9-55
パターン同期トリガ位置	PSP	NR2形式	PSP?	PATTERN	P9-53
パターン・データ・プリセット(1ページ・全ビット)	PST	NR1形式	−	PATTERN	P9-52
ZERO SUBST / PRBS段数	PTN	NR1形式	PTN?	PATTERN	P9-39
発生パターン切替	PTS	NR1形式	PTS?	PATTERN	P9-38
電源断、回復状態	−	−	PWI?	その他	P9-88
フロッピー・データ・リコール	RCL	NR1形式	−	MEMORY	P9-25
パターン・データ出力バイト数	RED	NR1形式	−	その他	P9-85
内部クロック分解能切替	RES	NR1形式	RES?	INTERNAL CLOCK	P9-22
フロッピー・データ・リセーブ	RSV	NR1形式	−	MEMORY	P9-28
内部タイマ設定	RTM	NR1形式	RTM?	その他	P9-87
フロッピー・データ・セーブ	SAV	NR1形式	−	MEMORY	P9-27
マーク率のANDビットシフト数	SFT	NR1形式	SFT?	その他	P9-80
同期信号出力選択	SOP	NR1形式	SOP?	その他	P9-78
1/1 SPEED, 1/4 SPEED表示切替	SPD	NR1形式	SPD?	OUTPUT	P9-76

表9-1　デバイス・メッセージ一覧表(アルファベット順: つづき)

機　能　項　目	コントロール・メッセージ		データ・リクエスト・メッセージ	デバイス・メッセージ詳細	
	ヘッダ部	数値データ部	ヘッダ部	セクション	ページ
データ/データ・トラッキング	TRK	NR1形式	TRK?	OUTPUT	P9-75
終端コード切り替え	TRM	NR1形式	TRM?	その他	P9-90
パターン・データ入力バイト数	WRT	NR1形式	−	その他	P9-84
ZERO SUBST長	ZLN	NR1形式	ZLN?	PATTERN	P9-46

デバイス・メッセージ一覧表 (パネルとの対応)

各コントロール・メッセージおよびデータ・リクエスト・メッセージについてパネル上のキーに対応した形で図9-2-(1)～(6), および表9-2-(1)～(5)に示します。

● INTERNAL CLOCK セクション

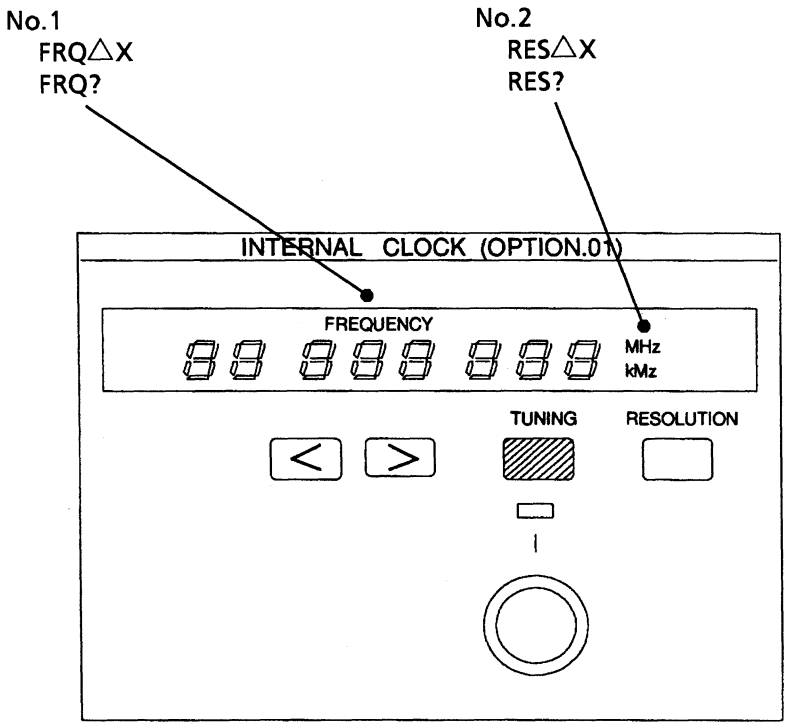
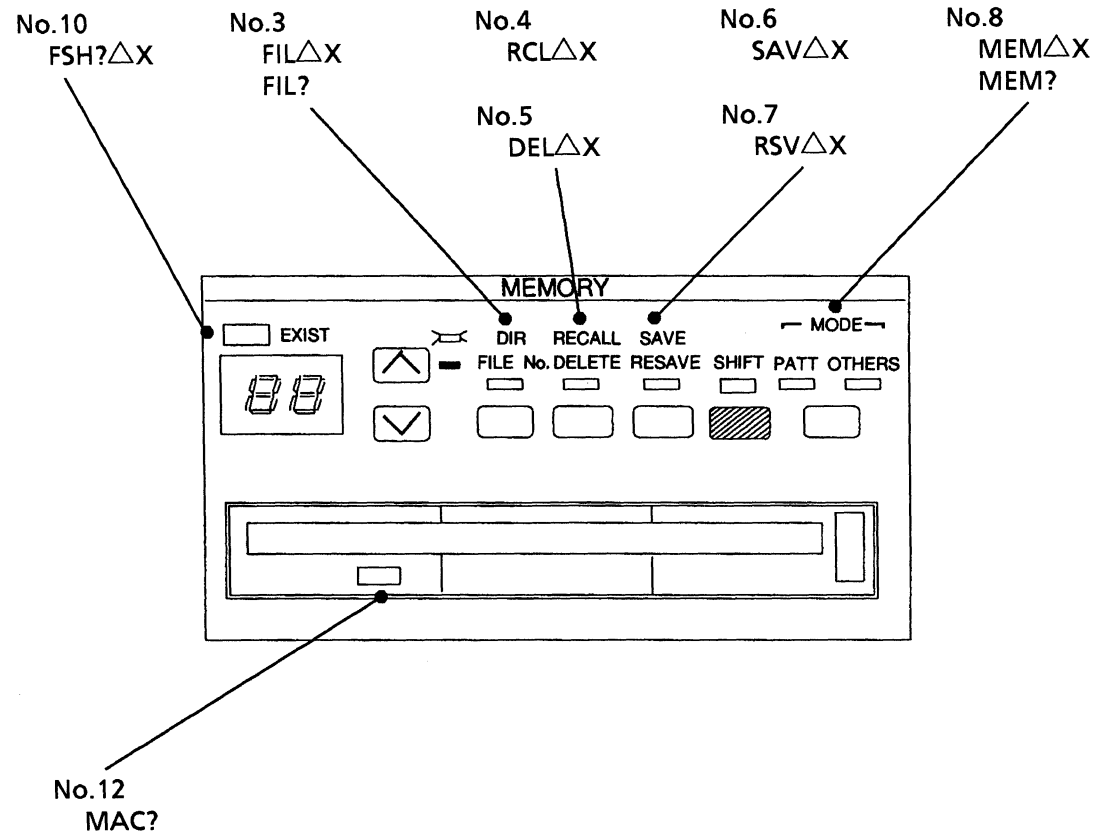


図9-2-(1)　INTERNAL CLOCK セクション

表9-2-(1)　デバイス・メッセージ一覧表 (INTERNAL CLOCK セクション)

機　能　項　目	コントロール・メッセージ		データ・リクエスト・メッセージ	デバイス・メッセージ詳細	
	ヘッダ部	数値データ部	ヘッダ部	項目 No.	ページ
● INTERNAL CLOCK セクション					
内部クロック周波数	FRQ	NR1形式	FRQ?	1	P9-20
内部クロック分解能切替	RES	NR1形式	RES?	2	P9-22

● MEMORY セクション



- FD mode : No. 11 FMD?
- FD error message : No. 13 FDE?
- FD format : No. 9 FDF

図9-2-(2)　MEMORY セクション

表9-2-(2)　デバイス・メッセージ一覧表(MEMORYセクション)

機　能　項　目	コントロール・メッセージ		データ・リクエスト・メッセージ	デバイス・メッセージ詳細	
	ヘッダ部	数値データ部	ヘッダ部	項目 No.	ページ
● MEMORYセクション					
ファイルNo./ディレクトリモード切替	FIL	NR1形式	FIL?	3	P9-24
フロッピー・データ・リコール	RCL	NR1形式	−	4	P9-25
フロッピー・データ・デリート	DEL	NR1形式	−	5	P9-26
フロッピー・データ・セーブ	SAV	NR1形式	−	6	P9-27
フロッピー・データ・リセーブ	RSV	NR1形式	−	7	P9-28
メモリ・モード切替	MEM	NR1形式	MEM?	8	P9-29
フロッピー・フォーマット	FDF	−	−	9	P9-30
ファイル内容の検索	−	−	FSH?	10	P9-31
メモリ・フロッピーモード	−	−	FMD?	11	P9-33
フロッピー・アクセス状態	−	−	MAC?	12	P9-34
フロッピー・エラーメッセージ	−	−	FDE?	13	P9-35

● PATTEERN セクション

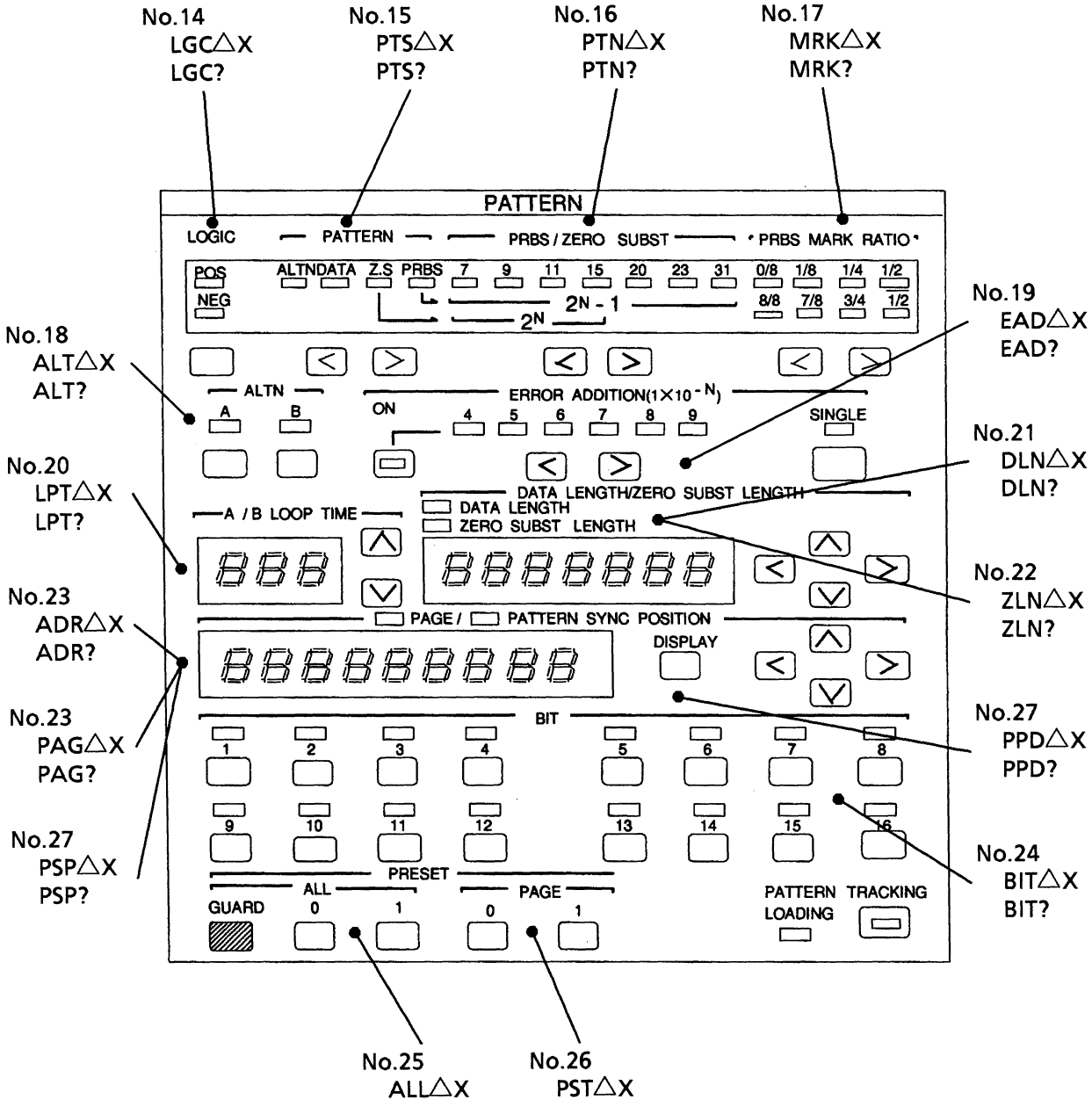


図9-2-(3)　PATTERN セクション

表9-2-(3)　デバイス･メッセージ一覧表 (PATTEERN セクション)

機　能　項　目	コントロール･メッセージ		データ･リクエスト･メッセージ	デバイス･メッセージ詳細	
	ヘッダ部	数値データ部	ヘッダ部	項目 No.	ページ
● PATTERN セクション					
パターン論理	LGC	NR1形式	LGC?	14	P9-37
発生パターン切替	PTS	NR1形式	PTS?	15	P9-38
ZERO SUBST / PRBS段数	PTN	NR1形式	PTN?	16	P9-39
PRBSマーク率	MRK	NR1形式	MRK?	17	P9-40
オルタネート･パターン A/B表示切替	ALT	NR1形式	ALT?	18	P9-41
誤り挿入	EAD	NR1形式	EAD?	19	P9-42
オルタネート･A/B･ループ回数	LPT	NR1形式	LPT?	20	P9-43
データ長	DLN	NR1形式	DLN?	21	P9-44
ZERO SUBST長	ZLN	NR1形式	ZLN?	22	P9-46
ページ数/パターン同期トリガ位置	PAG ADR	NR1形式 NR1形式	PAG? ADR?	23	P9-47
パターン･ビット	BIT	NR1形式 HEX形式	BIT?	24	P9-49
パターン･データ･プリセット (全ページ･全ビット)	ALL	NR1形式	−	25	P9-51
パターン･データ･プリセット (1ページ･全ビット)	PST	NR1形式	−	26	P9-52
パターン同期トリガ位置	PSP	NR2形式	PSP?	27	P9-53
ページ数/パターン同期トリガ位置表示切替	PPD	NR1形式	PPD?	28	P9-55

● OUTPUT セクション

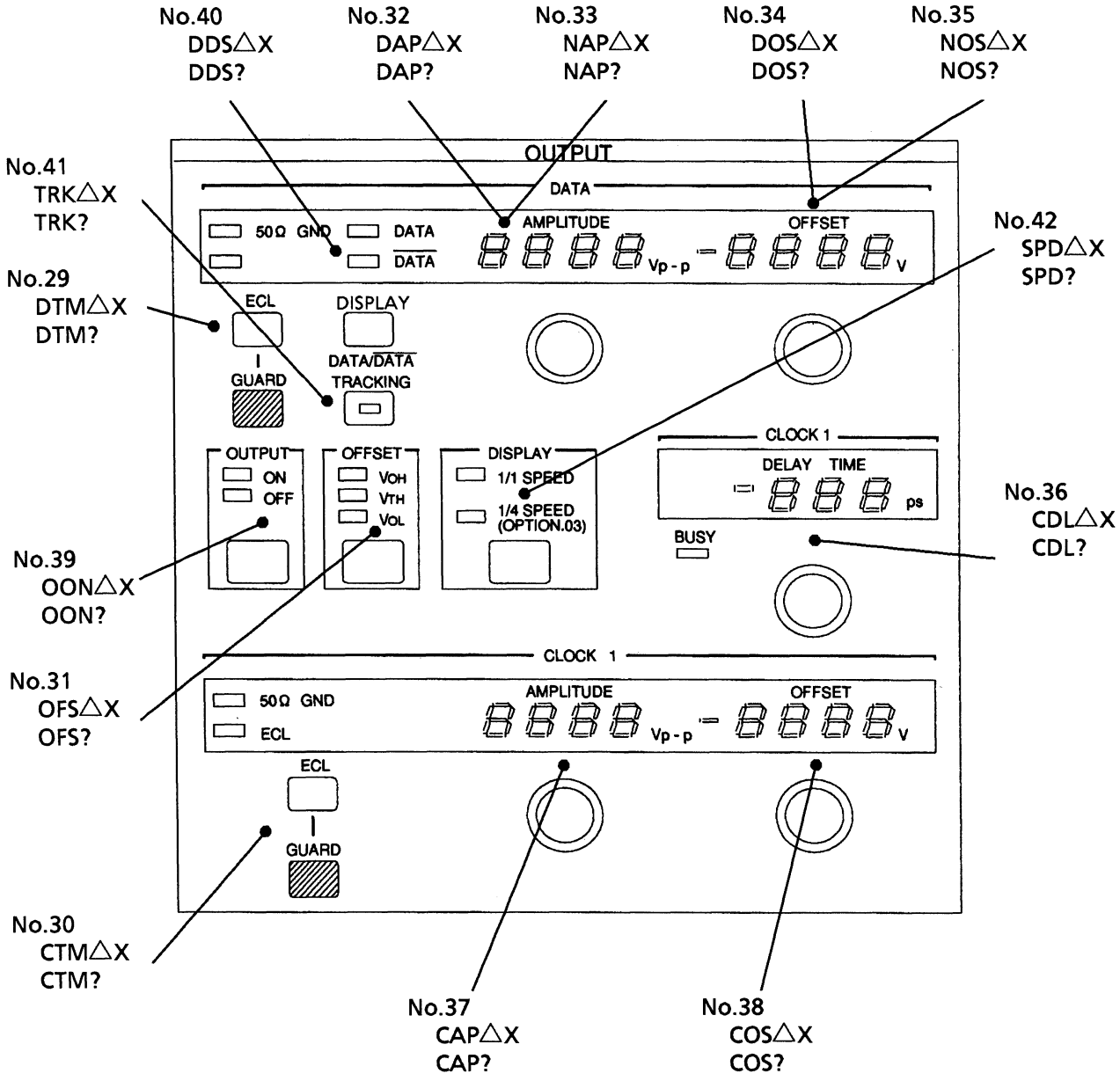


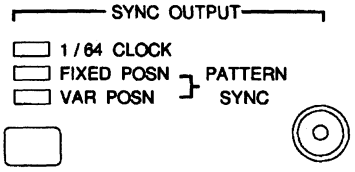
図9-2-(4)　OUTPUT セクション

表9-2-(4)　デバイス・メッセージ一覧表 (OUTPUT セクション)

機　能　項　目	コントロール・メッセージ		データ・リクエスト・メッセージ	デバイス・メッセージ詳細	
	ヘッダ部	数値データ部	ヘッダ部	項目 No.	ページ
● OUTPUTセクション					
データ出力終端電圧	DTM	NR1形式	DTM?	29	P9-57
クロック1出力終端電圧	CTM	NR1形式	CTM?	30	P9-58
オフセット基準値	OFS	NR1形式	OFS?	31	P9-59
データ出力振幅	DAP	NR2形式	DAP?	32	P9-60
$\overline{\text{データ}}$出力振幅	NAP	NR2形式	NAP?	33	P9-61
データ出力オフセット	DOS	NR2形式	DOS?	34	P9-62
$\overline{\text{データ}}$出力オフセット	NOS	NR2形式	NOS?	35	P9-67
クロック1出力遅延時間	CDL	NR2形式	CDL?	36	P9-69
クロック1出力振幅	CAP	NR2形式	CAP?	37	P9-70
クロック1出力オフセット	COS	NR2形式	COS?	38	P9-71
出力オン/オフ	OON	NR1形式	OON?	39	P9-73
データ/$\overline{\text{データ}}$表示切替	DDS	NR1形式	DDS?	40	P9-74
データ/$\overline{\text{データ}}$・トラッキング	TRK	NR1形式	TRK?	41	P9-75
1/1 SPEED, 1/4 SPEED表示切替	SPD	NR1形式	SPD?	42	P9-76

● その他のセクション

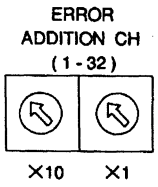
(正面パネル)



No.43
SOP△X
SOP?

図9-2-(5)　その他のセクション (正面パネル)

(背面パネル)



No.44
ECH△X
ECH?

(ファンクションSW)

FUNCTION

ITEMS	SW	SPECIFICATION
BIT SHIFT NUMBER FOR MARK RATIO VARIED	1	0:1 BIT 1:3 BIT
EXTERNA:L ERROR INJECTION	2	0: OFF 1: ON
FD FORMATTING TYPE	3	0: 1440kB/720kB 1: 1232kB/640kB
ALTERNATE PATTERN A/B SELECT TIMING	4	0: INT 1: EXT

No.45
SFT△X
SFT?

No.46
EEI△X
EEI?

No.9
FDF△X

No.47
APS△X
APS?

図9-2-(6)　その他のセクション (背面パネル・ファンクションSW)

NOTE

背面ファンクションSWはリモート状態で設定された値と背面のスイッチの値が異なる場合は、以下の処理をしています。

リモートで設定された値は、本器がリモート状態を継続中は保持されます。(ただし、イニシャライズコマンドのINI,*RSTを送出した場合は初期化されます。)

しかし、いったん本器がローカル状態になると背面のファンクションSWの設定状態を優先して、リモート時の設定内容は無効となります。

表9-2-(5)　デバイス・メッセージ一覧表(その他のセクション)

機　能　項　目	コントロール・メッセージ		データ・リクエスト・メッセージ	デバイス・メッセージ詳細	
	ヘッダ部	数値データ部	ヘッダ部	項目 No.	ページ
● 正面パネル 同期信号出力選択	SOP	NR1形式	SOP?	43	P9-78
● 背面パネル 誤り挿入チャンネル	ECH	NR1形式	ECH?	44	P9-79
● ファンクションスイッチ マーク率のANDビットシフト数	SFT	NR1形式	SFT?	45	P9-80
外部誤り挿入	EEI	NR1形式	EEI?	46	P9-81
オルタネート・パターン A/B切替信号選択	APS	NR1形式	APS?	47	P9-82
● その他 初期化	INI	−	−	48	P9-83
パターン・データ入力バイト数	WRT	NR1形式	−	49	P9-84
パターン・データ出力バイト数	RED	NR1形式	−	50	P9-85
内蔵シンセPLL	−	−	PLL?	51	P9-86
内部タイマ設定	RTM	NR1形式	RTM?	52	P9-87
電源断、回復状態	−	−	PWI?	53	P9-88
ディレイ状態	−	−	DLY?	54	P9-89
メモリ・フロッピーモード	−	−	FMD?	11	P9-33
終端コード切り替え	TRM	NR1形式	TRM?	55	P9-90

デバイス・メッセージの詳細説明

MP1761Cの個々のコントロール・メッセージおよびデータ・リクエスト・メッセージについて詳細な説明をします。

以下の例は、ヒューレット・パッカード社HP9000シリーズのHP-BASICで説明しています。

● INTERNAL CLOCK セクション

次のページより、**INTERNAL CLOCK** セクションの各コントロール・メッセージについて示します。なお、文中の△はスペースを意味します。

1) FRQ　　内部クロック周波数(FReQuency)

■ 機　能　　内部クロック周波数を設定します。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
FRQ	FRQ△m	FRQ?	FRQ△m　(kHz単位: FIX 8) (MHz単位: FIX 5)

■ mの値　　内部クロック周波数設定分解能がkHz単位の場合

最小値:　50000
最大値:　12500000
ステップ:　1

レスポンスは下記の様になります。

FRQ△△△△**50000**(最小値)
　　:
FRQ△**12500000**(最大値)

内部クロック周波数設定分解能がMHz単位の場合

最小値:　50
最大値:　12500
ステップ:　1

レスポンスは下記の様になります。

FRQ△△△△**50**　　(最小値)
　　:
FRQ△**12500**　　(最大値)

■ コマンド種別　　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム:　オプション01(内蔵シンセサイザ)が未搭載の場合
　　フロッピー・ディスクをアクセス中の場合

問い合わせ:　次の設定条件の場合は、無効となりERR(CR/LF)を出力します。オプション01が未搭載の場合

■ 使用例

プログラム ： 分解能がMHz単位で内部クロック周波数を500MHzにする場合

```
OUTPUT△700;"FRQ△500"
```

分解能がkHz単位で内部クロック周波数を500MHzにする場合

```
OUTPUT△700;"FRQ△500000"
```

問い合わせ ： 周波数が1GHzに設定してある場合

```
OUTPUT△700;"FRQ?"
ENTER△700;B$
PRINT△B$
```

↓

周波数分解能がMHz単位の場合

```
FRQ△△△△△1000 (CR/LF)
```

周波数分解能がkHz単位の場合

```
FRQ△△1000000 (CR/LF)
```

■ Note

オプション01内蔵の場合に外部からクロックを入力している場合には、表示上の設定はできますが、実際の出力に変化はありません。また、問い合わせに対しては表示の値を出力します。

2) RES　　内部クロック分解能切替 (RESolution)

■ 機　能　　内部クロック周波数の分解能を設定します。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
RES	RES△m	RES?	RES△m　(FIX 1)

■ mの値　　Ø: kHz単位
1: MHz単位

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム ： オプション01未搭載の場合
フロッピー・ディスクをアクセス中の場合

問い合わせ ： 次の設定条件の場合は、無効となりERR(CR/LF)を出力します。
オプション01未搭載の場合

■ 使用例　　プログラム ： 内部クロック周波数分解能をkHz単位にする場合

```
OUTPUT△700;"RES△Ø"
```

問い合わせ ： 内部クロック周波数分解能がMHz単位の場合

```
OUTPUT△700;"RES?"
ENTER△700;B$
PRINT△B$
          ↓
RES△1(CR/LF)
```

■ Note　　オプション01内蔵の場合に外部からクロックを入力している場合には、表示上の設定はできますが、実際の出力に変化はありません。また、問い合わせに対しては表示の値を出力します。

● MEMORYセクション

次のページより、**MEMORY**セクションの各コントロール・メッセージについて示します。なお、文中の△はスペースを意味します。

3) FIL　　ファイルNo./ディレクトリモード切替 (FILe no. / directory mode)

■ 機　能　　ファイルNo.・モードとディレクトリ・モードの切替設定をします。

ヘッダ	プログラム	問い合わせ	レスポンス (文字数)
FIL	FIL△m	FIL?	FIL△m (FIX 1)

■ mの値
Ø： File No.モード
1： DIRモード

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　次の設定条件の場合は、無効となります。

プログラム ： フロッピー・ディスクをアクセス中の場合

問い合わせ ： 無し

■ 使用例

プログラム ： DIRモードに設定を行う場合

```
OUTPUT△700;"FIL△1"
```

この時、挿入されているフロッピー・ディスクをアクセスし、DIR情報を内部メモリに格納します。

問い合わせ ： File No.モードに設定されている場合

```
OUTPUT△700;"FIL?"
ENTER△700;B$
PRINT△B$
```

↓

FIL△Ø(CR/LF)を出力します。

■ Note　ファイル・アクセス時にエラーが発生した場合は、エラーとなりMEMORY表示器にエラー情報を表示します。
このエラー表示は、下記の設定を行った場合にクリアされます。

- ファイルNo./ディレクトリモード切替 (No. 3)
- フロッピー・データ・リコール (No. 4)
- フロッピー・データ・デリート (No. 5)
- フロッピー・データ・セーブ (No. 6)
- フロッピー・データ・リセーブ (No. 7)
- メモリ・モード切替 (No. 8)
- フロッピー・フォーマット (No. 9)

4) RCL　　フロッピー・データ・リコール (ReCaLl)

■ 機　能　　フロッピー・ディスクの内容を本器に設定します。

ヘッダ	プログラム	問い合わせ	レスポンス (文字数)
RCL	RCL△m	無　し	無　し

■ mの値　　ファイル名を0～99の範囲で設定します。

数値範囲 :　最大値　99
　　最小値　0
　　ステップ　1

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム : フロッピー・ディスクをアクセス中の場合

■ 使用例　　プログラム : ファイル No. が9のファイル内容を本器に設定する場合

```
OUTPUT△700;"RCL△9"
```

■ Note　　指定したファイルが存在しない場合には、エラーとなり MEMORY 表示器にエラー情報を表示します。
このエラー表示は、下記の設定を行った場合にクリアされます。

- ファイル No./ディレクトリモード切替 (No. 3)
- フロッピー・データ・リコール (No. 4)
- フロッピー・データ・デリート (No. 5)
- フロッピー・データ・セーブ (No. 6)
- フロッピー・データ・リセーブ (No. 7)
- メモリ・モード切替 (No. 8)
- フロッピー・フォーマット (No .9)

また、拡張イベント・ステータス・レジスタ ESR2 (ERROR) bit 1の FD 異常発生ビットを立てます。

メモリ・モードの設定状態によって下記のファイルを読みだします。

PATT モードの場合　: RR**.PTN もしくは TT**.PTN
OTHERS モードの場合 : TT**.OTH

ただし、PATTモードでRR**.PTNとTT**.PTNの**の部分が等しいファイルがあった場合にはTT**.PTNを優先的に読みだします。

5) DEL　　フロッピー・データ・デリート (DELete)

■ 機　能　　フロッピー・ディスクの指定されたファイル No. を削除します。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
DEL	DEL△m	無　し	無　し

■ mの値　　ファイル名を0～99の範囲で設定します。

数値範囲 :　最大値　99
　　最小値　0
　　ステップ　1

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム : フロッピー・ディスクをアクセス中の場合

■ 使用例　　プログラム : ファイル No. が9のファイル内容を削除する場合

```
OUTPUT△700;"DEL△9"
```

■ Note　　指定したファイルが存在しない場合には、エラーとなり MEMORY 表示器にエラー情報を表示します。
このエラー表示は、下記の設定を行った場合にクリアされます。

① ファイル No. / ディレクトリモード切替 (No. 3)
　フロッピー・データ・リコール　(No. 4)
　フロッピー・データ・デリート　(No. 5)
　フロッピー・データ・セーブ　(No. 6)
　フロッピー・データ・リセーブ　(No. 7)
　メモリ・モード切替　(No. 8)
　フロッピー・フォーマット　(No. 9)

また、拡張イベント・ステータス・レジスタ ESR2 (ERROR) bit 1 の FD 異常発生ビットを立てます。

本器では、MP1762A/Cで保存したファイルの削除はできません。

メモリ・モードの設定状態によって下記のファイルを削除します。
　PATT モードの場合　: TT**.PTN
　OTHERS モードの場合 : TT**.OTH

6) SAV　　フロッピー・データ・セーブ (SAVe)

■ 機　能　　本器の設定内容をフロッピー・ディスクにセーブします。

ヘッダ	プログラム	問い合わせ	レスポンス (文字数)
SAV	SAV△m	無　し	無　し

■ mの値　　ファイル名を0～99の範囲で設定します。

数値範囲 :　最大値　99
　　最小値　0
　　ステップ　1

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。
プログラム : フロッピー・ディスクをアクセス中の場合

■ 使用例　　プログラム : ファイル No. 9で本器の設定内容を保存する場合
OUTPUT△700;"SAV△9"

■ Note　　指定したファイルが存在する場合には、エラーとなり MEMORY 表示器にエラー情報を表示します。
このエラー表示は、下記の設定を行った場合にクリアされます。

ファイル No./ディレクトリモード切替 (No. 3)
フロッピー・データ・リコール (No. 4)
フロッピー・データ・デリート (No. 5)
フロッピー・データ・セーブ (No. 6)
フロッピー・データ・リセーブ (No. 7)
メモリ・モード切替 (No. 8)
フロッピー・フォーマット (No. 9)

また、拡張イベント・ステータス・レジスタ ESR2 (ERROR) bit 1の FD 異常発生ビットを立てます。

メモリ・モードの設定状態によって下記のファイルを保存します。
PATT モードの場合　: TT**.PTN
OTHERS モードの場合 : TT**.OTH

7) RSV　　フロッピー・データ・リセーブ (ReSaVe)

■ 機　能　　フロッピー・ディスクの内容を本器に設定します。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
RSV	RSV△m	無　し	無　し

■ mの値　　ファイル名を0～99の範囲で設定します。

数値範囲 :　最大値　99
　　最小値　0
　　ステップ　1

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム : フロッピー・ディスクをアクセス中の場合

■ 使用例　　プログラム : ファイル No. が9のファイル内容を上書きする場合

```
OUTPUT△700;"RSV△9"
```

■ Note　　指定したファイルが存在しない場合には、エラーとなり MEMORY 表示器にエラー情報を表示します。
このエラー表示は、下記の設定を行った場合にクリアされます。

- ファイル No./ディレクトリモード切替 (No. 3)
- フロッピー・データ・リコール (No. 4)
- フロッピー・データ・デリート (No. 5)
- フロッピー・データ・セーブ (No. 6)
- フロッピー・データ・リセーブ (No. 7)
- メモリ・モード切替 (No. 8)
- フロッピー・フォーマット (No. 9)

また、拡張イベント・ステータス・レジスタ ESR2 (ERROR) bit 1の FD 異常発生ビットを立てます。

メモリ・モードの設定状態によって下記のファイルを再保存します。

PATT モードの場合　: RR**.PTN もしくは TT**.PTN
OTHERS モードの場合 : TT**.OTH

8) MEM　　メモリ・モード切替 (MEMory mode)

■ 機　能　　PATT / OTHERS の切替設定を行います。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
MEM	MEM△m	MEM?	MEM△m　(FIX 1)

■ mの値　　Ø : PATT モード
1 : OTHERS モード

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム : フロッピー・ディスクをアクセス中の場合

問い合わせ : 無し

■ 使用例　　プログラム : メモリ・モードを PATT に設定する場合

```
OUTPUT△700;"MEM△Ø"
```

問い合わせ : メモリ・モードが OTHERS に設定されている場合

```
OUTPUT△700;"MEM?"
ENTER△700;B$
PRINT△B$
```

↓

MEM△1 (CR / LF) を出力します。

■ Note　　PATT モードとは、PATTern モードを意味します。
この場合は、メモリの内容は PATTERN セクションの内容を対象とします。
OTHERS モードとは、上記以外の部分の内容をメモリの対象とします

ファイル・アクセス等でエラーが発生した場合には、MEMORY 表示器にエラー情報を表示します。
このエラー表示は、下記の設定を行った場合にクリアされます。

- ファイル No. / ディレクトリモード切替 (No. 3)
- フロッピー・データ・リコール　(No. 4)
- フロッピー・データ・デリート　(No. 5)
- フロッピー・データ・セーブ　(No. 6)
- フロッピー・データ・リセーブ　(No. 7)
- メモリ・モード切替　(No. 8)
- フロッピー・フォーマット　(No. 9)

また、拡張イベント・ステータス・レジスタ ESR2 (ERROR) bit 1 の FD 異常発生ビットを立てます。

メモリ・モードの設定状態によって下記のファイルを再保存します。

- PATT モードの場合　: RR**.PTN もしくは TT**.PTN
- OTHERS モードの場合 : TT**.OTH

9) FDF　　フロッピー・フォーマット (FD Format)

■ 機　能　　フロッピー・ディスクのフォーマットを行います。
フォーマット形式の選択は、背面 FUNCTION SW3で行って下さい。
また、2HD / 2DD の判定は挿入された FD に対して自動的に検出し判定します。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
FDF	FDF	無　し	無　し

■ mの値　　無し

■ コマンド種別　　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム ： フロッピー・ディスクをアクセス中の場合
ファイル No. / ディレクトリモード切替がディレクトリ・モードの場合

■ 使用例　　プログラム ： フロッピー・ディスクのフォーマットを行う。

```
OUTPUT△700;"FDF"
```

■ Note　　ファイル・アクセス等でエラーが発生した場合には、MEMORY 表示器にエラー情報を表示します。
このエラー表示は、下記の設定を行った場合にクリアされます。

ファイル No. / ディレクトリモード切替 (No. 3)
フロッピー・データ・リコール (No. 4)
フロッピー・データ・デリート (No. 5)
フロッピー・データ・セーブ (No. 6)
フロッピー・データ・リセーブ (No. 7)
メモリ・モード切替 (No. 8)
フロッピー・フォーマット (No. 9)

また、拡張イベント・ステータス・レジスタ ESR2 (ERROR) bit 1の FD 異常発生ビットを立てます。

10) FSH?　　ファイル内容の検索 (File SearcH?)

■ 機　能　　フロッピー・ディスク内に保存されているデータの情報を出力します。
対象となるファイル名は、下記の3種類です。

TT**.PTN
RR**.PTN　　RR**.OTH

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
FSH	無　し	FSH?△m1	FSH△m2,m3,m4,m5, m2　(FIX 7) m3　(FIX 7) m4　(FIX 2) m5　(各FIX 2)

■ mの値
m1 : ファイル No. の前後半の選択
0 : 前半 (No. 0 ~ No. 49)
1 : 後半 (No. 50 ~ No. 99)
m2 : Unused size
m3 : Used size
m4 : File 数
m5 : 前半もしくは後半の File No. (対象となるファイル名のみ)

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

問い合わせ : 無し

■ 使用例　　問い合わせ : フロッピー・ディスク内にファイル No. 1 ~ 10のファイルが存在する場合 (Unused size および Used size は一例です。)

```
OUTPUT△700;"FSH?△0"
ENTER△700;B$
PRINT△B$
```

↓

FSH△△72294,△△18132,10,01,02,03,04,05,06,07,08,09,10(CR/LF)を出力します。

フロッピー・ディスク内に対象となるファイルが存在しない場合

```
OUTPUT△700;"FSH?△0"
ENTER△700;B$
PRINT△B$
```

↓

FSH△△723968,△△△6144,△0,--(CR/LF)を出力します。

■ **Note**

ファイル内容の検索は、メモリ・モード切替 (PATT / OTHERS) の設定状態に従っています。

(1) メモリ・モード切替が PATT の場合
　TT**.PTN および RR**.PTN のファイルを検索します。

(2) メモリ・モード切替が OTHERS の場合
　TT**.OTH のファイルを検索します。

MP1761C で保存した PATT ファイルと MP1762A/C で保存した PATT ファイルが同一ファイル名でフロッピー・ディスクに保管されている場合は MP1761C で保存されたファイル名を優先して出力します。

ファイル内容の検索は、その直前のディレクトリ情報を出力しています。
また本器はフロッピーの挿入検出機能がないため、フロッピー・ディスク交換時に以前のディレクトリ情報が更新されません。
そのため、フロッピー・ディスク挿入および交換時には、必ず DIR モードに切替をして、ディレクトリ情報の更新を行って下さい。

11) FMD?　　メモリ・フロッピー モード (memory Fd MoDe?)

■ 機　能　　フロッピー・ディスクのフォーマットタイプを出力します。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
FMD	無　し	FMD?	FMD△m　(FIX 1)

■ mの値

Ø : 1440 k
1 : 720 k
2 : 1232 k
3 : 640 k

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　次の設定条件の場合は、無効となります。

問い合わせ : 無し

■ 使用例　問い合わせ : 挿入されている FD が 2DD で1440 kB / 720 k のフォーマット形式が設定されている場合

```
OUTPUT△700;"FMD?"
ENTER△700;B$
PRINT△B$
```

↓

FMD△1 (CR / LF) を出力します。

■ Note　MS-DOS フォーマット形式の選択は、背面 FUNCTION SW3で行います。この設定は電源投入時の状態で決定するため、電源投入後に設定を変更することはできません。

- FUNCTION SW3 = 'OFF'の時

フォーマット時の容量	セクタ長 [バイト/セクタ]	セクタ数 [セクタ/トラック]	トラック数 [トラック/サイド]
1440 kB	512	18	80
720 kB	512	9	80

- FUNCTION SW3 = 'ON' の時

フォーマット時の容量	セクタ長 [バイト/セクタ]	セクタ数 [セクタ/トラック]	トラック数 [トラック/サイド]
1232 kB	1024	8	77
640 kB	512	8	80

フロッピー・ディスクが挿入されていない時に本問い合わせを実行すると現在設定されている FDD の状態 (FUNCTION SW3) に従って1440 kB もしくは1232 kB 時の情報を出力します。

12) MAC?　　フロッピー・アクセス状態 (Memory Access Condition?)

■ 機　能　　フロッピー・ディスクのアクセス状態を出力します。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
MAC	無　し	MAC?	MAC△m　(FIX 1)

■ mの値　　Ø：ノンアクセス状態
1：アクセス状態

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

問い合わせ：無し

■ 使用例　　問い合わせ：フロッピーがアクセス中で無い場合

```
OUTPUT△700;"MAC?"
ENTER△700;B$
PRINT△B$
```

↓

MAC△Ø(CR/LF)を出力します。

13) FDE?　　フロッピー・エラーメッセージ (FD Error message?)

■ 機　能　　フロッピー・ディスクのエラー情報を出力します。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
FDE	無　し	FDE?	FDE△m　(FIX 2)

■ mの値　　エラー・メッセージ

Ø： E0 (フォーマット形式の違いによるエラー)
1： E1 (書き込み時にライトプロテクトされている)
2： E2 (書き込み領域が不足している)
3： E3 (読み出し時に指定されたファイル名が存在しない)
4： E4 (同一ファイル名でセーブしようとした)
5： E5 (書き込み障害エラー)
6： E6 (読み出し障害エラー)
7： E7 (DMA転送エラー)
8： E8 (その他のエラー)
9： E9 (ハードウェア・トラブル・エラー)
1Ø： No error

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

問い合わせ ： 無し

■ 使用例　　問い合わせ ： ハードウェア・トラブル・エラーが発生している場合

```
OUTPUT△7ØØ;"FDE?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

FDE△9 (CR/LF) を出力します。

■ Note　　このエラー表示は、下記の設定を行った場合にクリアされます。

ファイル No./ディレクトリモード切替 (No. 3)
フロッピー・データ・リコール (No. 4)
フロッピー・データ・デリート (No. 5)
フロッピー・データ・セーブ (No. 6)
フロッピー・データ・リセーブ (No. 7)
メモリ・モード切替 (No. 8)
フロッピー・フォーマット (No. 9)

また、拡張イベント・ステータス・レジスタ ESR2 (ERROR) bit 1のFD異常発生ビットを立てます。

● PATTERNセクション

次のページより、PATTERNセクションの各コントロール・メッセージについて示します。なお、文中の△はスペースを意味します。

14) LGC　　パターン論理 (LoGiC mode)

■ 機　能　　データの論理設定を行います。

論理と実際のデータ出力の関係は、ALTERNATE / DATA / ZERO SUBST の場合と PRBS の場合で異なります。(機能・操作編取扱説明書を参照してください。)

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
LGC	LGC△m	LGC?	LGC△m　(FIX 1)

■ mの値　　Ø : Positive
1 : Negative

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム : フロッピー・ディスクをアクセス中の場合

問い合わせ : 無し

■ 使用例　　プログラム : パターン論理を正論理 (Positive) に設定を行う場合

```
OUTPUT△700;"LGC△Ø"
```

問い合わせ : パターン論理が負論理 (Negative) に設定されている場合

```
OUTPUT△700;"LGC?"
ENTER△700;B$
PRINT△B$
```

↓

LGC△1 (CR / LF) を出力します。

■ Note　　パターンが PRBS モードの場合、パターン論理を設定すると、パターン・マーク率も論理に対応して切替わります。

- 正論理　0/8, 1/8, 1/4, 1/2
- 負論理　8/8, 7/8, 3/4, $\overline{1/2}$

15) PTS　　発生パターン切替 (PaTtern Select)

■ 機　能　　発生パターンの設定を行います。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
PTS	PTS△m	PTS?	PTS△m　(FIX 1)

■ mの値

Ø : Alternate
1 : Data
2 : Zero subst
3 : PRBS

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　次の設定条件の場合は、無効となります。

プログラム : フロッピー・ディスクをアクセス中の場合

問い合わせ : 無し

■ 使用例

プログラム : 発生パターンを Alternate に設定する場合

```
OUTPUT△700;"PTS△Ø"
```

問い合わせ : 発生パターン論理が Data に設定されている場合

```
OUTPUT△700;"PTS?"
ENTER△700;B$
PRINT△B$
```

↓

PTS△1 (CR/LF) を出力します。

■ Note　発生パターンを切替えると、各発生パターンでの前回の設定状態が復帰します。
例えば、PRBS パターンに設定すると PRBS パターンの段数とパターン・マーク率の設定が復帰します。

16) PTN　　ZERO SUBST / PRBS 段数 (zero subst / prbs PaTterN mode)

■ 機　能　　ZERO SUBST / PRBS パターンの場合の設定を行います。

ヘッダ	プログラム	問い合わせ	レスポンス (文字数)
PTN	PTN△m	PTN?	PTN△m (FIX 1)

■ mの値

ZERO SUBST の場合
- 2 : 2^7
- 3 : 2^9
- 5 : 2^{11}
- 6 : 2^{15}

PRBS の場合
- 2 : 2^7-1
- 3 : 2^9-1
- 5 : $2^{11}-1$
- 6 : $2^{15}-1$
- 7 : $2^{20}-1$
- 8 : $2^{23}-1$
- 9 : $2^{31}-1$

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　次の設定条件の場合は、無効となります。

プログラム : フロッピー・ディスクをアクセス中の場合
発生パターンが Alternate、Data の場合

問い合わせ : 次の場合は無効となり、ERR (CR / LF) を出力します。
発生パターンが Alternate、Data の場合

■ 使用例

プログラム : 発生パターンを PRBS 2^7-1 に設定する場合

```
OUTPUT△700;"PTN△2"
```

問い合わせ : 発生パターン論理が PRBS $2^{31}-1$ に設定されている場合

```
OUTPUT△700;"PTN?"
ENTER△700;B$
PRINT△B$
```

↓

PTN△9 (CR / LF) を出力します。

: 発生パターンが DATA に設定されている場合

```
OUTPUT△700;"PTN?"
ENTER△700;B$
PRINT△B$
```

↓

ERR (CR / LF) を出力します。

17) MRK　　PRBS マーク率 (MaRK ratio mode)

■ 機　能　　PRBS パターンの場合のマーク率設定を行います。

ヘッダ	プログラム	問い合わせ	レスポンス (文字数)
MRK	MRK△m	MRK?	MRK△m (FIX 1)

■ mの値

	正論理 (Positive) の場合	負論理 (Negative) の場合
Ø :	0/8	8/8
1 :	1/8	7/8
2 :	1/4	3/4
3 :	1/2	1/2

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム ： フロッピー・ディスクをアクセス中の場合
発生パターンが Alternate、Data、Zero subst の場合

問い合わせ ： 次の場合は無効となり、ERR (CR / LF) を出力します。
発生パターンが Alternate、Data、Zero subst の場合

■ 使用例

プログラム ： パターン・マーク率を0/8に設定する場合

```
OUTPUT△700;"MRK△Ø"
```

問い合わせ ： パターン・マーク率が1/8に設定されている場合

```
OUTPUT△700;"MRK?"
ENTER△700;B$
PRINT△B$
```

↓

MRK△1 (CR / LF) を出力します。

： 発生パターンが DATA に設定されている場合

```
OUTPUT△700;"MRK?"
ENTER△700;B$
PRINT△B$
```

↓

ERR (CR / LF) を出力します。

18) ALT　　オルタネート・パターンA/B表示切替 (ALTernate a / b)

■ 機　能　　オルタネート・パターン時のA/B表示切替の設定を行います。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
ALT	ALT△m	ALT?	ALT△m　(FIX 1)

■ mの値
Ø : Aパターン
1 : Bパターン

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　次の設定条件の場合は、無効となります。

プログラム : フロッピー・ディスクをアクセス中の場合
発生パターンがDATA、ZERO SUBST、PRBSの場合

問い合わせ : 次の場合には無効となり、ERR (CR / LF) を出力します。
発生パターンがDATA、ZERO SUBST、PRBSの場合

■ 使用例　プログラム : オルタネート表示をAに切替える場合

```
OUTPUT△700;"ALT△Ø"
```

問い合わせ : オルタネート表示をAに設定されている場合

```
OUTPUT△700;"ALT?"
ENTER△700;B$
PRINT△B$
```

↓

ALT△1 (CR / LF) を出力します。

: 発生パターンがPRBSに設定されている場合

```
OUTPUT△700;"ALT?"
ENTER△700;B$
PRINT△B$
```

↓

ERR (CR / LF) を出力します。

19) EAD　　誤り挿入(Error ADditon)

■ 機　能　　32ルート中の1ルートに指定の割合の符号誤りを挿入します。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
EAD	EAD△m	EAD?	EAD△m　(FIX 1)

■ mの値

内部誤り挿入の場合(EEI△0)

Ø : OFF(誤り挿入無し)
1 : 1×10^{-4}
2 : 1×10^{-5}
3 : 1×10^{-6}
4 : 1×10^{-7}
5 : 1×10^{-8}
6 : 1×10^{-9}
7 : SINGLE

外部誤り挿入の場合(EEI△1)

Ø : OFF(誤り挿入無し)
1 : ON(誤り挿入有り)

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　次の設定条件の場合は、無効となります。

プログラム : フロッピー・ディスクをアクセス中の場合

問い合わせ : 無し

■ 使用例

プログラム : 1×10^{-5}の誤りを挿入する場合

```
OUTPUT△700;"EAD△2"
```

問い合わせ : 1×10^{-4}の誤りが挿入されている場合

```
OUTPUT△700;"EAD?"
ENTER△700;B$
PRINT△B$
```

↓

EAD△1 (CR/LF) を出力します。

■ Note　誤り挿入の外部、内部を切り替えると、前設定状態になります。例えば、内部誤り挿入で1×10^{-8}を設定した後に、外部誤り挿入をONにして、内部誤り挿入に切り替えると、誤り挿入率は1×10^{-8}になります。

20) LPT　　オルタネート・A/B・ループ回数 (LooP Time)

■ 機　能　　AあるいはBパターンのうち、現在表示されているパターンのループ回数を設定します。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
LPT	LPT△m	LPT?	LPT△m　(FIX 3)

■ mの値　　ループ回数を下記の範囲内で設定します。

最小値:　1
最大値:　127
ステップ:　1

レスポンスは下記の様になります。

LPT△△△1　(最小値)
　　:
LPT△127　(最大値)

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム ： フロッピー・ディスクをアクセス中の場合
発生パターンがALTERNATE以外の設定の場合

問い合わせ ： 次の設定条件の場合には、無効となりERR(CR/LF)を出力します。
発生パターンがALTERNATE以外の設定の場合

■ 使用例　　プログラム ： ループ回数を10回にする場合

```
OUTPUT△700;"LPT△10"
```

問い合わせ ： ループ回数が17回に設定されている場合

```
OUTPUT△700;"LPT?"
ENTER△700;B$
PRINT△B$
```

↓

LPT△△17 (CR/LF) を出力します。

： 発生パターンがDATAに設定されている場合

```
OUTPUT△700;"LPT?"
ENTER△700;B$
PRINT△B$
```

↓

ERR (CR/LF) を出力します。

21) DLN　　データ長 (Data LeNgth)

■ 機　能　　発生パターンがALTERNATE、DATAの場合にデータ長の設定を行います。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
DLN	DLN△m	DLN?	DLN△m　(FIX 7)

■ mの値　　データ長の設定を下記の範囲内で設定します。

● オルタネートパターンの場合
　最大値　：4194304
　最小値　：128
　ステップ：128

● データパターンの場合
　最大値　：8388608
　最小値　：2
　ステップ：データ長によって下記の様に分類されます。

データ長		
2～	65536ビット/ステップ	1ビット
65536～	131072ビット/ステップ	2ビット
131072～	262144ビット/ステップ	4ビット
262144～	524288ビット/ステップ	8ビット
524288～	1048576ビット/ステップ	16ビット
1048576～	2097152ビット/ステップ	32ビット
2097152～	4194304ビット/ステップ	64ビット
4194304～	8388608ビット/ステップ	128ビット

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム：フロッピー・ディスクをアクセス中の場合
　発生パターンがZERO SUBST、PRBSの場合

問い合わせ：次の場合には無効となり、ERR (CR/LF) を出力します。
　発生パターンがZERO SUBST、PRBSの場合

■ 使用例　　プログラム：データ長を4ビットに設定する場合

```
OUTPUT△700;"DLN△4"
```

問い合わせ：データ長が32ビットに設定されている場合

```
OUTPUT△700;"DLN?"
ENTER△700;B$
PRINT△B$
```

↓

DLN△△△△△△32 (CR/LF) を出力します。

：発生パターンがPRBSに設定されている場合

```
OUTPUT△700;"DLN?"
ENTER△700;B$
PRINT△B$
```

↓

ERR (CR/LF) を出力します。

■ **Note**

データ長の設定値が存在しない値を設定した場合はデータ長を下記の要領で最適化します。

入力した値よりも小さく、しかも入力した値に最も近い値に変更します。

例) ● 入力したデータ長　　● 設定されるデータ長

131075　→→→　**131072**

22) ZLN　　ZERO SUBST 長 (Zero subst LeNgth)

■ 機　能　　測定パターンが ZERO SUBST の場合にゼロ挿入ビット長の設定を行います。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
ZLN	ZLN△m	ZLN?	ZLN△m　(FIX 5)

■ m の値　　ゼロ挿入ビット長の設定を下記の範囲内で設定します。

最大値　：ZERO SUBST 段数によって以下の範囲となります。

2^7 ：　127
2^9 ：　511
2^{11} ：　2047
2^{15} ：　32767

最小値　：1
ステップ：1

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム ：フロッピー・ディスクをアクセス中の場合
発生パターンが ALTERNATE、DATA、PRBS の場合

問い合わせ ：次の場合には無効となり、ERR (CR / LF) を出力します。
発生パターンが ALTERNATE、DATA、PRBS の場合

■ 使用例　　プログラム ：ZERO SUBST 長を1ビットに設定する場合

```
OUTPUT△700;"ZLN△1"
```

問い合わせ ：ZERO SUBST 長が127ビットに設定されている場合

```
OUTPUT△700;"ZLN?"
ENTER△700;B$
PRINT△B$
```

↓

ZLN△△△127 (CR / LF) を出力します。

：測定パターンが PRBS に設定されている場合

```
OUTPUT△700;"ZLN?"
ENTER△700;B$
PRINT△B$
```

↓

ERR (CR / LF) を出力します。

23) ADR　PAG　ページ数/パターン同期トリガ位置 (ADdRess / PAGe)

■ 機　能　　ページ数の設定を行います。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
ADR	ADR△m	ADR?	ADR△m　(FIX 9)
PAG	PAG△m	PAG?	PAG△m　(FIX 9)

■ mの値　　ページ数の設定を下記の範囲内で設定します。

最大値　：134217728
最小値　：1
ステップ：1

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム：フロッピー・ディスクをアクセス中の場合
表示がパターン同期位置の場合

問い合わせ：次の設定条件の場合は、無効になり、ERR (CR/LF) を出力します。
表示がパターン同期位置の場合

■ 使用例　　プログラム：ページを1ページに設定する場合

```
OUTPUT△700;"PAG△1"
```

問い合わせ：ページが16000ページに設定されている場合

```
OUTPUT△700;"PAG?"
ENTER△700;B$
PRINT△B$
```

↓

PAG△△△△△16000 (CR / LF) を出力します。

：表示がパターン同期位置の場合

```
OUTPUT△700;"PAG?"
ENTER△700;B$
PRINT△B$
```

↓

ERR (CR / LF) を出力します。

■ Note

ページ数の設定コマンドは、“ADR”と“PAG”の2種類ありますが、機能は同じです。
但し、設定できる最大ページ数は、設定されている測定パターンやデータ長の設定値によって異なります。
また、上記mの最大値である134217728を越えない範囲で、設定可能な最大ページ数を越えるページ数を入力すると、その時の最大ページ数に変更します。

例) データ長＝32、表示しているページ数＝1、最大ページ数は2ページですが、PAG△3を入力すると、表示しているページ数は2ページとなります。

ページの最大値は、データ長 ÷ 16を越えない値までで、余りがある場合は左記の商 ＋ 1までとなります。

パターンセクション中央部付近にDISPLAYキーが付いていない器については、問い合わせにおいてERRが出力されることはありません。
また、プログラムする場合はフロッピーディスクがアクセス中の場合以外有効になります。

24) BIT　　パターン・ビット (pattern BIT)

■ 機　能　　ビット・パターンの設定を行います。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
BIT	● NR1形式 BIT△m ● HEX形式 BIT△#Hm	BIT?	設定されているページ数から最大8ページ分までのビットの内容を、最大パターン設定ページまで、以下の形式で出力します。 PAG△**********: BIT△#H****,#H****,..., #H**** ビット・パターンを出力先頭ページとともにHEXで表示する。 最大8ページ分 (各 FIX 4)

■ mの値　　ビット・パターンを下記の範囲内で設定します。

- NR1形式
  - 最大値　：65535
  - 最小値　：0
  - ステップ：1
- HEX形式
  - 最大値　：FFFF
  - 最小値　：0
  - ステップ：1

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム：フロッピー・ディスクをアクセス中の場合
発生パターンがZERO SUBST、PRBSの場合

問い合わせ：無し

■ 使用例　　プログラム：現在設定されているページから3ページ分のビット・パターンを設定する場合

```
OUTPUT△700;"BIT△10,20,30"
OUTPUT△700;"BIT△#HFFFF,#H1000,#H2000"
```

データとデータの間をコンマ(,)で区切ることにより連続したページのパターン・ビットを設定することができます

ページ数を設定し、そのページから4ページ分のパターン・ビットの設定をする場合は

```
OUTPUT 700;"PAG△10;BIT△10,20,30,40"
OUTPUT 700;"PAG△10;BIT△#HFFFF,#H1000,
           #H2000,#H3000"
```

問い合わせ ： 表示ページが1、取りうる最大ページ数が10までとされている場合

1ページから8ページまでのデータを読みだします。

```
OUTPUT△700;"BIT?"
ENTER△700;B$
PRINT△B$
          ↓
PAG△△△△△△△△△1;   BIT△#H0000,#H0000,
                   #H0000,#H0000,#H0000,
                   #H0000,#H0000,#H0000
```

(CR/LF)を出力します。

■ **Note**

NR1またはHEX部のデータとデータの間をコンマ(,)で区切ることによって連続したページのパターン・ビットを設定することができます。(最大8ページ分)

ビット表示器のbit 1をLSB、bit 16をMSBとして設定・レスポンスを行います。

例えば、32768を設定すると最上位ビット(bit 16)が1となります。

25) ALL　　パターン・データ・プリセット (全ページ・全ビット) (prest ALL)

■ 機　能　　パターンデータ全ページの全ビットを0または1に設定します。

ヘッダ	プログラム	問い合わせ	レスポンス (文字数)
ALL	ALL△m1	無　し	無　し

■ mの値　　Ø　：　全ページ、クリア
1　：　全ページ、セット

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム　：　フロッピー・ディスクをアクセス中の場合
発生パターンがZERO SUBSTもしくはPRBSの場合

■ 使用例　　プログラム　：　発生パターンがDATAの場合で全ページをクリアする場合

OUTPUT△7ØØ;"ALL△Ø"

全ページのデータをクリアします。

■ Note　　発生パターンがALTERNATEパターンの場合はA/B表示切替 (No. 18) の状態によってAもしくはBのパターンについてプリセットを行います。

例えば、Aパターンを表示している場合は、本コマンドを実行するとAのパターンのみプリセットされます。

26) PST　　パターン・データ・プリセット (1ページ・全ビット) (PreST)

■ 機　能　　パターンデータ1ページの全ビットを0または1に設定します。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
PST	PST△m	無　し	無　し

■ mの値　　Ø　：1ページ、クリア
1　：1ページ、セット

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム　：　フロッピー・ディスクをアクセス中の場合
発生パターンがZERO SUBSTもしくはPRBSの場合

■ 使用例　　プログラム　：　発生パターンがDATAの場合で1ページをクリアする場合

```
OUTPUT△7ØØ;"PST△Ø"
```

設定されているページの1ページ分のデータをクリアします。

■ Note　　発生パターンがALTERNATEパターンの場合はA/B表示切替 (No. 18) の状態によってAもしくはBのパターンについてプリセットを行います。

例えば、Aパターンを表示している場合は、本コマンドを実行するとAのパターンのみプリセットされます。

27) PSP　　パターン同期トリガ位置(Pattern Sync Position)

■ 機　能　　パターンシンク(可変)のトリガ同期位置を設定します。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
PSP	PSP△m	PSP?	PSP△m　(FIX 9)

■ mの値　　パターン同期位置を下記の範囲内で設定します。

最大値：　発生パターンによって異なります。

PRBS7,	ZERO SUBST.7	: 8
PRBS9,	ZERO SUBST.9	: 32
PRBS11,	ZERO SUBST.11	: 128
PRBS15,	ZERO SUBST.15	: 2048
PRBS20		: 65536
PRBS23		: 524288
PRBS31		: 134217728
DATA		: (ビット長÷16)+α ビット長÷16の商が整数でない場合はα=1 商が整数の場合はα=0となります。

最小値：　1
ステップ: 1

レスポンスは下記の様になります。

PSP△△△△△△△△△1　　(最小値の場合)
　　:
PSP△134217728　　(最大値の場合)

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム ： フロッピー・ディスクをアクセス中の場合
　　表示がページの場合

問い合わせ ： 次の設定の場合は、無効となりERR(CR/LF)を出力します。
　　表示がページ数の場合

■ 使用例

プログラム ： パターン同期位置を3ページに設定する場合

```
OUTPUT△700;"PSP△3"
```

問い合わせ ： パターン同期位置が189ページに設定されている場合

```
OUTPUT△700;"PSP?"
ENTER△700;B$
PRINT B$
```

↓

PSP△△△△△△△189 (CR/LF)を出力します。

： 表示がページの場合

```
OUTPUT△700;"PSP?"
ENTER△700;B$
PRINT B$
```

↓

ERR (CR/LF)を出力します。

28) PPD　　ページ数/パターン同期トリガ位置表示切替 (Page / Pattern sync positon Display)

■ 機　能　　ページ数とパターン同期位置の表示を切り替えます。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
PPD	PPD△m	PPD?	PPD△m　(FIX 1)

■ mの値　　Ø：ページ数の表示
1：パターン同期位置の表示

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム：フロッピー・ディスクをアクセス中の場合

問い合わせ：無し

■ 使用例

プログラム：パターン同期位置を表示する場合

```
OUTPUT△700;"PPD△1"
```

問い合わせ：ページ数が表示されている場合

```
OUTPUT△700;"PPD?"
ENTER△700;B$
PRINT△B$
```

↓

PPD△Ø(CR/LF)を出力します。

● OUTPUT セクション

次のページより、OUTPUT セクションの各コントロール・メッセージについて示します。なお、文中の△はスペースを意味します。

29) DTM　　データ出力終端電圧 (Data TerMination)

■ 機　能　　データ出力の終端電圧を設定します。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
DTM	DTM△m	DTM?	DTM△m　(FIX 1)

■ mの値　　Ø : GND
1 : −2V (ECL)

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム : フロッピー・ディスクをアクセス中の場合

問い合わせ : 無し

■ 使用例　　プログラム : データ出力終端電圧をGNDに設定する場合

```
OUTPUT△700;"DTM△Ø"
```

問い合わせ : データ出力終端電圧の設定値が−2Vの場合

```
OUTPUT△700;"DTM?"
ENTER△700;B$
PRINT△B$
```

↓

DTM△1 (CR/LF) を出力します。

30) CTM　　クロック出力終端電圧 (Clock TerMination)

■ 機　能　　クロック出力の終端電圧を設定します。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
CTM	CTM△m	CTM?	CTM△m　(FIX 1)

■ mの値　　Ø : GND
1 : −2V (ECL)

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム : フロッピー・ディスクをアクセス中の場合

問い合わせ : 無し

■ 使用例　　プログラム : クロック出力終端電圧をGNDに設定する場合

```
OUTPUT△700;"CTM△Ø"
```

問い合わせ : クロック出力終端電圧の設定値が−2Vの場合

```
OUTPUT△700;"CTM?"
ENTER△700;B$
PRINT△B$
```

↓

CTM△1 (CR/LF) を出力します。

31) OFS　　オフセット基準値 (OFSset)

■ 機　能　　データ/データ/クロック出力のオフセット基準値を設定します。

ヘッダ	プログラム	問い合わせ	レスポンス (文字数)
OFS	OFS△m	OFS?	OFS△m (FIX 1)

■ mの値

Ø : オフセット基準値VOH
1 : オフセット基準値VTH
2 : オフセット基準値VOL

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　次の設定条件の場合は、無効となります。

プログラム : フロッピーディスクをアクセス中の場合

問い合わせ : 無し

■ 使用例

プログラム : オフセット基準値をVOLにする場合

```
OUTPUT△700;"OFS△2"
```

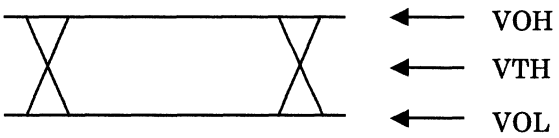
問い合わせ : オフセット基準値がVTHの場合

```
OUTPUT△700;"OFS?"
ENTER△700;B$
PRINT△B$
```

↓

OFS△1 (CR/LF) を出力します。

■ Note　各出力オフセット基準値の概念を次に示します。



オフセット基準値を設定すると、各出力オフセット電圧の表示は、その時の最適値に変更されます。

32) DAP　　データ出力振幅 (Data AmPlitude)

■ 機　能　　データ/$\overline{\text{データ}}$出力の振幅を設定します。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
DAP	DAP△m	DAP?	DAP△m　(FIX 5)

■ mの値　　データの出力振幅設定範囲である0.25 V~2.0 Vまでの値を設定します。

数値範囲 : 最大値　: 2.000
最小値　: 0.250
ステップ : 0.002

なお、オプション03 (1/4出力) 搭載時に、表示が1/4SPEEDの場合は、以下 の数値範囲で1/4出力の振幅設定を行います。

数値範囲 : 最大値　: 2.000
最小値　: 0.500
ステップ : 0.002

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム : フロッピー・ディスクをアクセス中の場合
オプション03未搭載時は、1/4出力振幅の設定は無効になります。

問い合わせ : 無し

■ 使用例　　プログラム : データ振幅を1.2Vにする場合

```
OUTPUT△700;"DAP△1.2"
```

問い合わせ : データ振幅が0.5Vに設定してある場合

```
OUTPUT△700;"DAP?"
ENTER△700;B$
PRINT△B$
```

↓

DAP△0.500 (CR/LF) を出力します。

■ Note　　データ/$\overline{\text{データ}}$・トラッキング・オフの場合は、データ出力のみの振幅を設定します。

33) NAP　　データ出力振幅 (iNverted data AmPlitude)

■ 機　能　　データ出力の振幅を設定します。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
NAP	NAP△m	NAP?	NAP△m　(FIX 5)

■ mの値　　データの出力振幅設定範囲である0.25 V~2.0 Vまでの値を設定します。

数値範囲 :　最大値　: 2.000
　　最小値　: 0.250
　　ステップ : 0.002

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム :　データ/データトラッキングオンの場合
　　オプション03搭載時に、表示が1/4SPEEDの場合
　　フロッピーディスクをアクセス中の場合

問い合わせ :　次の設定条の場合は無効になり、ERRを出力します。
　　データ/データトラッキングオンの場合
　　オプション03搭載時に、表示が1/4SPEEDの場合

■ 使用例

プログラム:　データ振幅を1.2Vにする場合

```
OUTPUT△700;"NAP△1.2"
```

問い合わせ :　データ振幅が0.5Vに設定してある場合

```
OUTPUT△700;"NAP?"
ENTER△700;B$
PRINT△B$
```

↓

NAP△0.500(CR/LF)を出力します。

34) DOS　　データ出力オフセット (Data OfSet)

■ 機　能　　データ/データ出力のオフセットを設定します。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
DOS	DOS△m	DOS?	DOS△m　(FIX 6)

■ mの値

データ出力オフセット設定範囲は、オフセット基準値の設定によって異なります。オフセット基準値VOHの場合は−2.0V〜+2.0Vまでの値を設定します。

数値範囲：最大値　：　2.000
　　最小値　：　−2.000
　　ステップ：　0.001

オフセット基準値VTHの場合は−3.0V〜+1.875Vまでの値を設定します。

数値範囲：最大値　：　1.875
　　最小値　：　−3.000
　　ステップ：　0.001

オフセット基準値VOLの場合は−4.0V〜+1.75Vまでの値を設定します。

数値範囲：最大値　：　1.750
　　最小値　：　−4.000
　　ステップ：　0.001

オプション03搭載時に、表示が1/4SPEEDなら、1/4データ出力のオフセットを次の範囲で設定します。

オフセット基準値VOHの場合は−1.5V〜+1.5Vまでの値を設定します。

数値範囲：最大値　：　1.500
　　最小値　：　−1.500
　　ステップ：　0.001

オフセット基準値VTHの場合は−2.5V〜+1.25Vまでの値を設定します。

数値範囲：最大値　：　1.250
　　最小値　：　−2.500
　　ステップ：　0.001

オフセット基準値VOLの場合は−3.5V〜+1.0Vまでの値を設定します。

数値範囲：最大値　：　1.000
　　最小値　：　−3.500
　　ステップ：　0.001

■ **コマンド種別**　シーケンシャル・コマンド

■ **使用制限**　次の設定条件の場合は、無効となります。

プログラム ： フロッピー・ディスクをアクセス中の場合

問い合わせ ： 無し

■ **使用例**　プログラム ： オフセットを−1.2Vにする場合

```
OUTPUT△700;"DOS△−1.2"
```

問い合わせ ： オフセットが0.5Vに設定してある場合

```
OUTPUT△700;"DOS?"
ENTER△700;B$
PRINT△B$
```

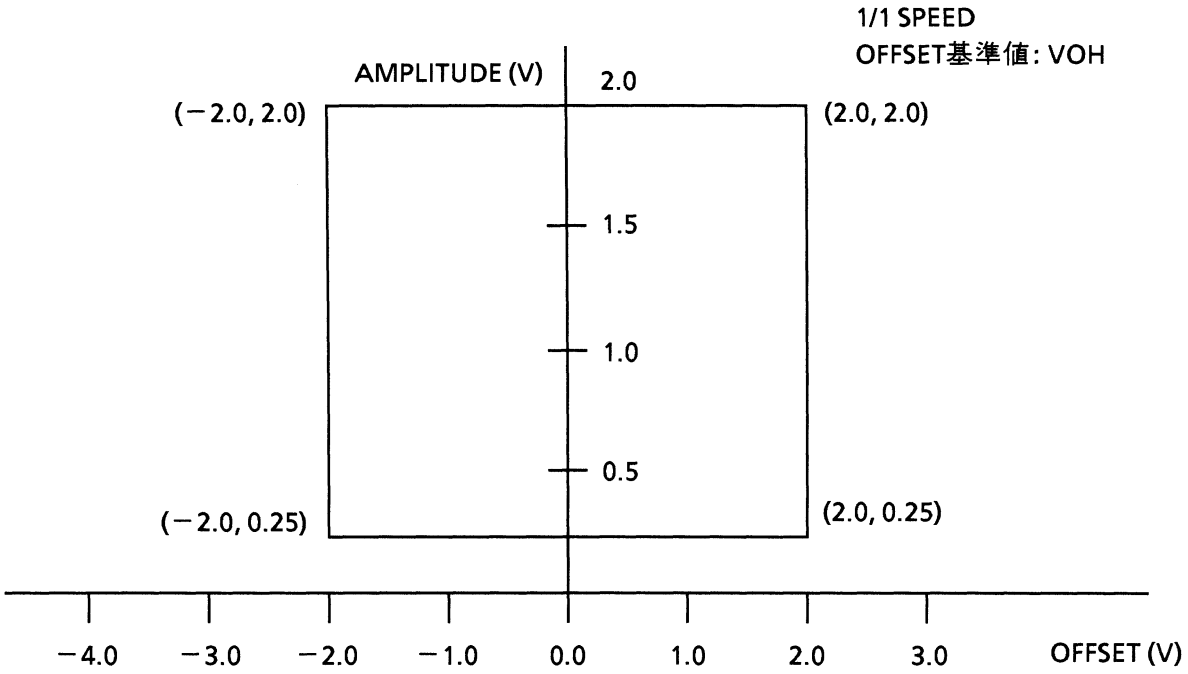
↓

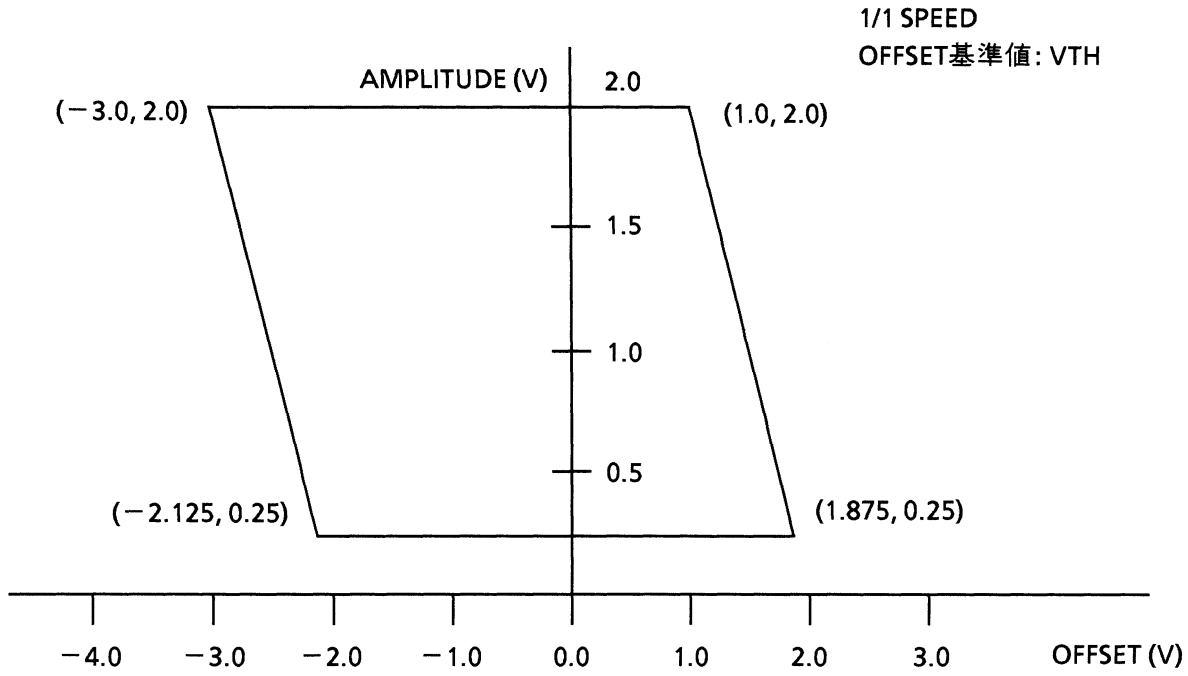
DOS△△Ø.5ØØ(CR/LF)を出力します。

■ **Note**　データ/$\overline{\text{データ}}$・トラッキング・オフの場合は、データ出力のみのオフセットを設定します。

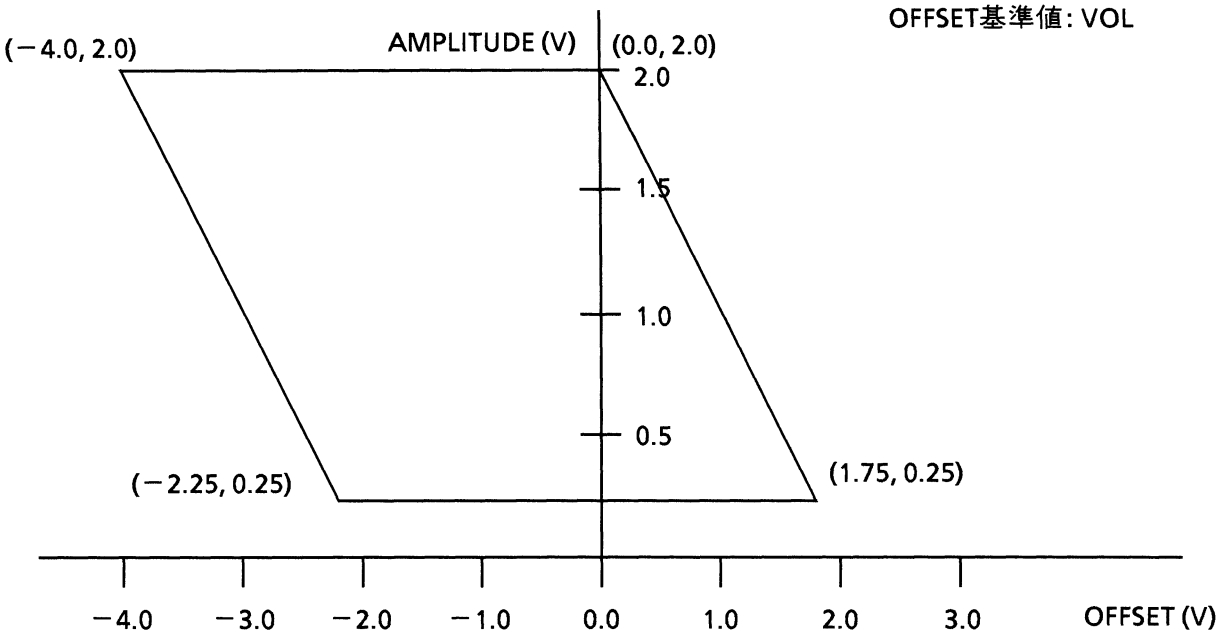
オプション03(1/4出力)搭載時、表示が1/4SPEEDの場合は、1/4出力のオフセットを設定します。

オフセットの設定できる範囲は、次ページから示す様に、出力振幅設定値によっても異なります。

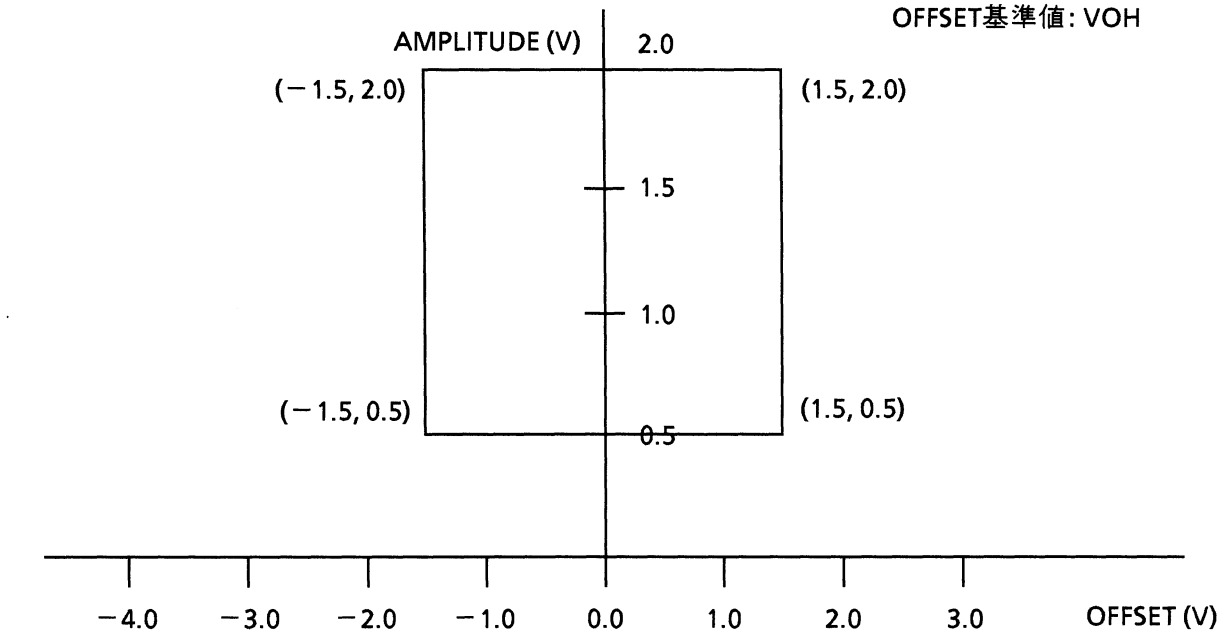

1/1 SPEED
OFFSET基準値: VOH
AMPLITUDE (V)
2.0
(−2.0, 2.0)
(2.0, 2.0)
1.5
1.0
0.5
(−2.0, 0.25)
(2.0, 0.25)
−4.0
−3.0
−2.0
−1.0
0.0
1.0
2.0
3.0
OFFSET (V)

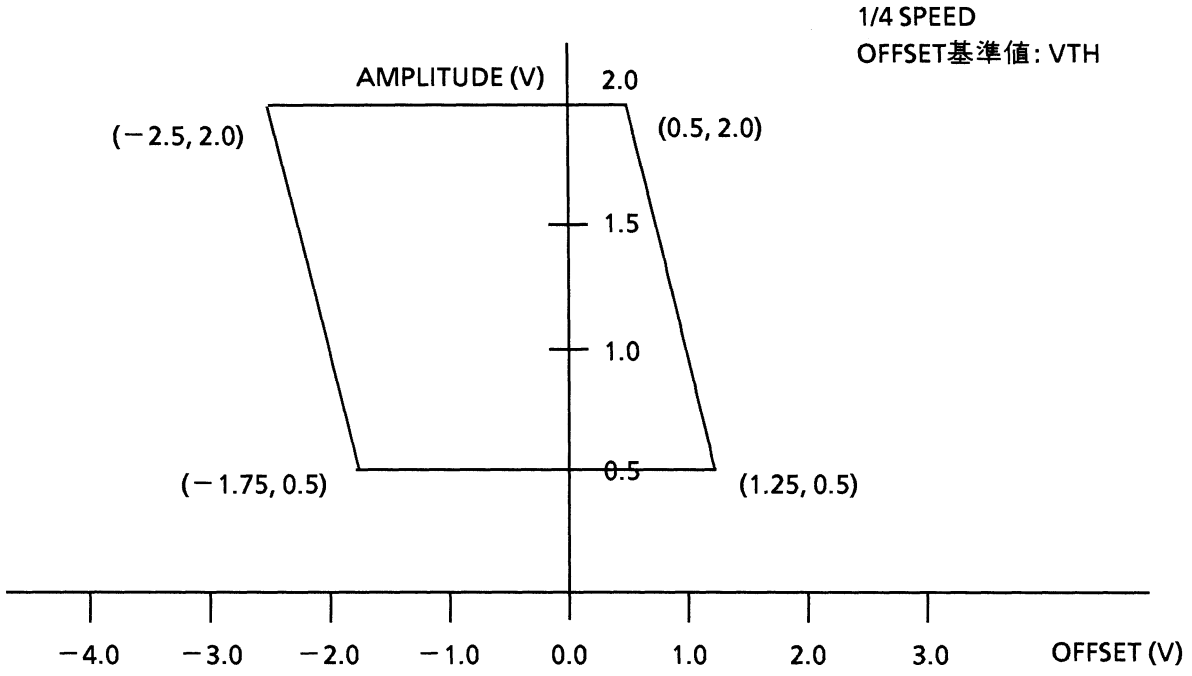

1/1 SPEED
OFFSET基準値: VTH
AMPLITUDE (V)
2.0
(−3.0, 2.0)
(1.0, 2.0)
1.5
1.0
0.5
(−2.125, 0.25)
(1.875, 0.25)
−4.0
−3.0
−2.0
−1.0
0.0
1.0
2.0
3.0
OFFSET (V)

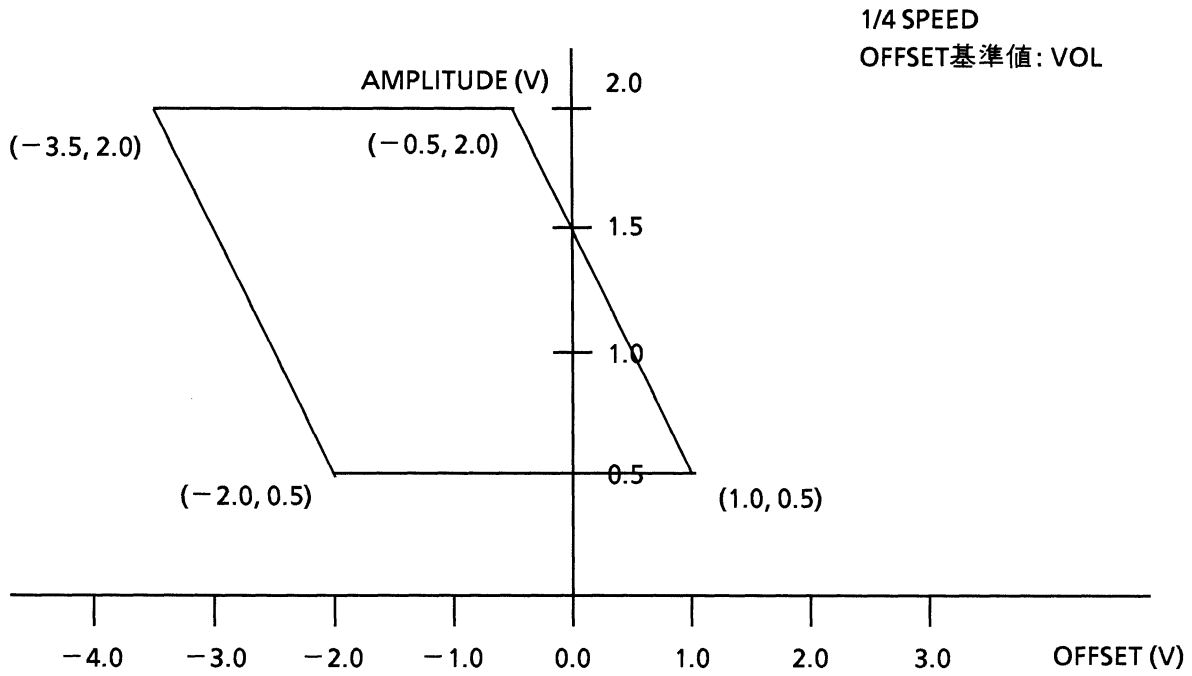
1/1 SPEED
OFFSET基準値: VOL



1/4 SPEED
OFFSET基準値: VOH




1/4 SPEED
OFFSET基準値: VTH
AMPLITUDE (V)
2.0
(−2.5, 2.0)
(0.5, 2.0)
1.5
1.0
0.5
(−1.75, 0.5)
(1.25, 0.5)
−4.0
−3.0
−2.0
−1.0
0.0
1.0
2.0
3.0
OFFSET (V)


1/4 SPEED
OFFSET基準値: VOL
AMPLITUDE (V)
2.0
(−3.5, 2.0)
(−0.5, 2.0)
1.5
1.0
0.5
(−2.0, 0.5)
(1.0, 0.5)
−4.0
−3.0
−2.0
−1.0
0.0
1.0
2.0
3.0
OFFSET (V)

35) NOS　　データ出力オフセット (iNverted data OffSet)

■ 機　能　　データ出力のオフセットを設定します。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
NOS	NOS△m	NOS?	NOS△m　(FIX 6)

■ mの値　　データ出力オフセット設定範囲は、オフセット基準値の設定によって異なります。
オフセット基準値VOHの場合は−2.0V〜+2.0Vまでの値を設定します。

数値範囲 : 最大値　:　2.000
　　最小値　:　−2.000
　　ステップ :　0.001

オフセット基準値VTHの場合は−3.0V〜+1.875Vまでの値を設定します。

数値範囲 : 最大値　:　1.875
　　最小値　:　−3.000
　　ステップ :　0.001

オフセット基準値VOLの場合は−4.0V〜+1.75Vまでの値を設定します。

数値範囲 : 最大値　:　1.750
　　最小値　:　−4.000
　　ステップ :　0.001

■ コマンド種別

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム : データ/データトラッキングオンの場合
　　オプション03搭載時に、表示が1/4SPEEDの場合
　　フロッピー・ディスクをアクセス中の場合

問い合わせ : 次の設定条の場合は無効になり、ERRを出力します。
　　データ/データトラッキングオンの場合
　　オプション03搭載時に、表示が1/4SPEEDの場合

■ 使用例

プログラム：　データオフセットを1.2Vにする場合

```
OUTPUT△700;"NOS△1.2"
```

問い合わせ：　データ振幅が0.5Vに設定してある場合

```
OUTPUT△700;"NOS?"
ENTER△700;B$
PRINT△B$
```

↓

NOS△△0.500(CR/LF)を出力します。

：　オプション02未搭載の場合

```
OUTPUT△700;"NOS?"
ENTER△700;B$
PRINT△B$
```

↓

NOS△△0.500(CR/LF)を出力します。

■ Note

オフセット設定のできる範囲は、1/1SPEEDのデータオフセット設定(DOS)と同様です。

36) CDL　　クロック1出力遅延時間 (Clock1 DeLay)

■ 機　能　　クロック1とデータ/データ間の遅延時間を設定します。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
CDL	CDL△m	CDL?	CDL△m　(FIX 5)

■ mの値　　クロック1の出力位相設定範囲である500ps～－500psまでの値を設定します。

数値範囲：最大値　：　500
最小値　：　－500
ステップ：　1

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム：オプション03搭載時に、表示が1/4SPEEDの場合
フロッピー・ディスクをアクセス中の場合

問い合わせ：次の設定条の場合は無効になり、ERRを出力します。
オプション03搭載時に、表示が1/4SPEEDの場合

■ 使用例　　プログラム：クロック遅延時間を－100psにする場合

```
OUTPUT△700;"CDL△－100"
```

問い合わせ：クロック遅延時間が100psに設定してある場合

```
OUTPUT△700;"CDL?"
ENTER△700;B$
PRINT B$
```

↓

CDL△△△100(CR/LF)を出力します。

37) CAP　　クロック1出力振幅 (Clock1 AmPlitude)

■ 機　能　　クロック1の出力振幅を設定します。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
CAP	CAP△m	CAP?	CAP△m　(FIX 5)

■ mの値　　クロック1の出力振幅設定範囲である0.25V~2.0Vまでの値を設定します。

数値範囲 : 最大値 : 2.000
最小値 : 0.250
ステップ : 0.002

オプション03搭載時、表示が1/4SPEEDに設定されている場合の出力振幅範囲は、0.5V~2.0Vになります。

数値範囲 : 最大値 : 2.000
最小値 : 0.500
ステップ : 0.002

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム : フロッピー・ディスクをアクセス中の場合

問い合わせ : 無し

■ 使用例　　プログラム : クロック1出力振幅を1.5Vにする場合

```
OUTPUT△700;"CAP△1.5"
```

問い合わせ : クロック1出力振幅が0.25Vに設定してある場合

```
OUTPUT△700;"CAP?"
ENTER△700;B$
PRINT△B$
```

↓

CAP△0.250(CR/LF)を出力します。

38) COS　　クロック1出力オフセット (Clock1 OffSet)

■ 機　能　　クロック1の出力振幅を設定します。

ヘッダ	プログラム	問い合わせ	レスポンス (文字数)
COS	COS△m	COS?	COS△m (FIX 6)

■ mの値

クロック1出力オフセット設定範囲は、オフセット基準値の設定によって異なります。
オフセット基準値VOHの場合は−2.0V～+2.0Vまでの値を設定します。

数値範囲：最大値　：　2.000
　　最小値　：　−2.000
　　ステップ：　0.001

オフセット基準値VTHの場合は−3.0V～+1.875Vまでの値を設定します。

数値範囲：最大値　：　1.875
　　最小値　：　−3.000
　　ステップ：　0.001

オフセット基準値VOLの場合は−4.0V～+1.75Vまでの値を設定します。

数値範囲：最大値　：　1.750
　　最小値　：　−4.000
　　ステップ：　0.001

オプション03搭載時に、表示が1/4SPEEDなら、1/4データ出力のオフセットを次の範囲で設定します。

オフセット基準値VOHの場合は−1.5V～+1.5Vまでの値を設定します。

数値範囲：最大値　：　1.500
　　最小値　：　−1.500
　　ステップ：　0.001

オフセット基準値VTHの場合は−2.5V～+1.25Vまでの値を設定します。

数値範囲：最大値　：　1.250
　　最小値　：　−2.500
　　ステップ：　0.001

オフセット基準値VOLの場合は−3.5V～+1.0Vまでの値を設定します。

数値範囲：最大値　：　1.000
　　最小値　：　−3.500
　　ステップ：　0.001

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限

次の設定条件の場合は、無効となります。

プログラム ： フロッピー・ディスクをアクセス中の場合

問い合わせ ： 無し

■ 使用例

プログラム ： クロック1出力オフセットを1.5Vにする場合

```
OUTPUT△700;"COS△1.5"
```

問い合わせ ： クロック1出力オフセットが－0.25Vに設定してある場合

```
OUTPUT△700;"COS?"
ENTER△700;B$
PRINT△B$
```

↓

COS△－0.250(CR/LF)を出力します。

■ Note

CLOCK1出力オフセットの設定できる範囲は、データ出力オフセットと同様に、CLOCK1出力振幅の設定値によっても異なります。

39) OON　　出力オン/オフ (Output ON/off)

■ 機　能　　データ、$\overline{データ}$、クロック1、$\overline{クロック1}$、1/4出力の出力を0Vにします。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
OON	OON△m	OON?	OON△m　(FIX 1)

■ mの値　　Ø：出力オフ
1：出力オン

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム：フロッピー・ディスクをアクセス中の場合

問い合わせ：無し

■ 使用例　　プログラム：出力をオフにする場合

```
OUTPUT△700;"OON△Ø"
```

問い合わせ：出力がオンの場合

```
OUTPUT△700;"OON?"
ENTER△700;B$
PRINT△B$
```

↓

OON△1 (CR/LF) を出力します。

40) DDS　　データ/データ表示切替 (Data/data Display Select)

■ 機　能　　データ/データの表示設定値を選択します。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
DDS	DDS△m	DDS?	DDS△m　(FIX 1)

■ mの値　　Ø : データ設定値表示
1 : データ設定値表示

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム : データ/データ・トラッキングオンの場合
オプション03搭載時、表示が1/4SPEEDの場合
フロッピー・ディスクをアクセス中の場合

問い合わせ : 次の設定条件の場合には、無効となりERRを出力します。
データ/データ・トラッキングオンの場合
オプション03搭載時、表示が1/4SPEEDの場合

■ 使用例　　プログラム : データの設定値を表示する場合

```
OUTPUT△700;"DDS△Ø"
```

問い合わせ : データの設定値を表示している場合

```
OUTPUT△700;"DDS?"
ENTER△700;B$
PRINT△B$
```

↓

DDS△1 (CR / LF) を出力します。

41) TRK　　データ/$\overline{データ}$・トラッキング (data/$\overline{data}$ TRacKing)

■ 機　能　　データ/$\overline{データ}$・トラッキングのオン・オフを設定します。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
TRK	TRK△m	TRK?	TRK△m　(FIX 1)

■ mの値　　Ø：トラッキングオフ
1：トラッキングオン

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム：オプション03搭載時、表示が1/4SPEEDの場合
フロッピー・ディスクをアクセス中の場合

問い合わせ：次の設定条件の場合には、無効となりERRを出力します。
オプション03搭載時、表示が1/4SPEEDの場合

■ 使用例　　プログラム：トラッキングをオフにする場合

```
OUTPUT△700;"TRK△Ø"
```

問い合わせ：トラッキングオンに設定されている場合

```
OUTPUT△700;"TRK?"
ENTER△700;B$
PRINT△B$
```

↓

TRK△1 (CR/LF) を出力します。

42) SPD　1/1 SPEED・1/4 SPEED表示切替 (1/1 SPeeD・1/4 speed display select)

■ 機　能　1/1SPEED・1/4SPEEDの設定値の表示を選択します。

ヘッダ	プログラム	問い合わせ	レスポンス (文字数)
SPD	SPD△m	SPD?	SPD△m (FIX 1)

■ mの値
Ø : 1/1SPEED表示
1 : 1/4SPEED表示

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　次の設定条件の場合は、無効となります。

プログラム : オプション03未搭載の場合
フロッピー・ディスクをアクセス中の場合

問い合わせ : 次の設定条件の場合には、無効となりERRを出力します。
オプション03未搭載の場合

■ 使用例

プログラム : 1/1SPEEDの表示にする場合

```
OUTPUT△700;"SPD△Ø"
```

問い合わせ : 1/4SPEEDの表示にされている場合

```
OUTPUT△700;"SPD?"
ENTER△700;B$
PRINT△B$
```

↓

SPD△1 (CR / LF) を出力します。

: オプション03未搭載の場合

```
OUTPUT△700;"SPD?"
ENTER△700;B$
PRINT B$
```

↓

ERR (CR/LF) を出力します。

■ Note　オプション03搭載時に有効になります。

● その他のセクション

次のページより、その他のセクションの各コントロール・メッセージについて示します。なお、文中の△はスペースを意味します。

■ Note

本章の背面ファンクションスイッチの設定内容はRemote時はコマンド設定された内容を優先しますが、Local状態になると背面のファンクションスイッチの設定内容に戻ります。

注意してください。

43) SOP　　同期信号出力選択 (Sync OutPut)

■ 機　能　　同期信号出力の制御を行います。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
SOP	SOP△m	SOP?	SOP△m　(FIX 1)

■ mの値
- Ø : 1/64 CLOCK
- 1 : PATTERN SYNC (FIXED)
- 2 : PATTERN SYNC (VARIABLE)

■ コマンド種別　　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム : フロッピー・ディスクをアクセス中の場合

問い合わせ : 無し

■ 使用例

プログラム : 同期信号出力を1/64 CLOCK に設定する場合

```
OUTPUT△700;"SOP△Ø"
```

問い合わせ: 同期信号出力が PATTERN SYNC (FIXED) に設定されている場合

```
OUTPUT△700;"SOP?"
ENTER△700;B$
PRINT△B$
```

↓

SOP△1 (CR/LF) を出力します。

44) ECH　　誤り挿入チャンネル (Error addition CHannel)

■ 機　能　　誤りを挿入するチャンネルを選択します。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
ECH	ECH△m	ECH?	ECH△m　(FIX 2)

■ mの値　　誤り挿入するチャンネルをch1~ch32の範囲内で設定します。

(ch1)　最小値：　1
(ch32)　最大値：　32

レスポンスは下記の様になります。

(ch1の場合)　ECH△△1
:
(ch32の場合)　ECH△32

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム ： フロッピー・ディスクをアクセス中の場合

問い合わせ ： 無し

■ 使用例　　プログラム ： ch3に誤りを挿入する場合

```
OUTPUT△700;"ECH△3"
```

問い合わせ ： 誤り挿入がch8に設定されている場合

```
OUTPUT△700;"ECH?"
ENTER△700;B$
PRINT△B$
```

↓

ECH△△8 (CR/LF) を出力します。

45) SFT　　マーク率のANDビットシフト数(mark ratio and bit ShiFT)

■ 機　能　　PRBSマーク率のANDビットシフト数を設定します。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
SFT	SFT△m	SFT?	SFT△m　(FIX 1)

■ mの値　　Ø : 1 bitシフト
1 : 3 bitシフト

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム : フロッピー・ディスクをアクセス中の場合
発生パターンがALTERNATE、DATA、ZERO SUBSTの場合

問い合わせ : 次の場合は無効となり、ERR(CR/LF)を出力します。
発生パターンがALTERNATE、DATA、ZERO SUBSTの場合

■ 使用例　　プログラム : マーク率のANDビットシフトを1 bitに設定する場合

```
OUTPUT△700;"SFT△Ø"
```

問い合わせ : マーク率のANDビットシフトが1 bitに設定されている場合

```
OUTPUT△700;"SFT?"
ENTER△700;B$
PRINT△B$
```

↓

SFT△1(CR/LF)を出力します。

: 発生パターンがALTERNATE、DATA、ZERO SUBSTに設定されている場合

```
OUTPUT△700;"SFT?"
ENTER△700;B$
PRINT△B$
```

↓

ERR(CR/LF)を出力します。

46) EEI　　外部誤り挿入 (External Error Injection)

■ 機　能　　誤り挿入の方法を外部挿入、内部挿入で切り替えます。

ヘッダ	プログラム	問い合わせ	レスポンス (文字数)
EEI	EEI△m	EEI?	EEI△m (FIX 1)

■ mの値　　Ø： 内部誤り挿入
1： 外部誤り挿入

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム ： フロッピー・ディスクをアクセス中の場合

問い合わせ ： 無し

■ 使用例　　プログラム ： 外部誤り挿入をオンにする場合

```
OUTPUT△700;"EEI△1"
```

問い合わせ ： 外部誤り挿入がオフにされている場合

```
OUTPUT△700;"EEI?"
ENTER△700;B$
PRINT△B$
```

↓

EEI△Ø(CR/LF)を出力します。

47) APS　オルタネート・パターン A/B切替信号選択 (Alternate Pattern A/B Select timming)

■ 機　能　オルタネート・パターンのA/B切替信号の出力を、内部発生したものを使用するか、外部からの入力信号を使用するか選択します。

ヘッダ	プログラム	問い合わせ	レスポンス (文字数)
APS	APS△m	APS?	APS△m (FIX 1)

■ mの値
Ø： 内部で発生した信号を出力します。
1： 外部からの信号を出力します。

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　次の設定条件の場合は、無効となります。

プログラム ： フロッピー・ディスクをアクセス中の場合

問い合わせ ： 無し

■ 使用例

プログラム ： 外部からの入力信号を使用する場合

```
OUTPUT△700;"APS△1"
```

問い合わせ ： 内部発生の信号を使用している場合

```
OUTPUT△700;"APS?"
ENTER△700;B$
PRINT△B$
```

↓

APS△Ø(CR/LF)を出力します。

48) INI　　初期化 (INItialize)

■ 機　能　　工場出荷時の状態に強制初期化します。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
INI	INI	無　し	無　し

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　フロッピー・ディスクをアクセス中の場合

■ 使用例　プログラム ： 工場出荷時の状態に初期化する場合

```
OUTPUT△700;"INI"
```

■ Note　このコマンドはLOCALキーを押しながら電源投入をした場合の動作と同じです。

49) WRT　　パターン・データ入力バイト数 (pattern data WRiTe)

■ 機　能　　パターンデータのDMA転送を行うバイト数とスタートアドレスを設定します。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
WRT	WRT△m1,m2	無　し	無　し

■ mの値

m1 : パターン転送バイト数
　数値範囲 : 最大値　1048376
　　最小値　1
　　ステップ　1

m2 : パターン入力先頭アドレス
　数値範囲 : 最大値　524288
　　最小値　0
　　ステップ　1

発生パターンがALTERNATEパターンの場合は、上記最大値の半分となります。

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム : フロッピー・ディスクをアクセス中の場合
発生パターンがZERO SUBSTもしくはPRBSの場合
パターン転送バイト数＋パターン先頭アドレス×2>1048376の場合

■ 使用例　　プログラム : 発生パターンがDATAの場合で1ページから10ページまでのデータを設定する場合

```
DIM△B(9)
READ△B(*)
DATA△1,2,4,8,16,32,64,128,256,512
OUTPUT△700;"WRT△20,0"
OUTPUT△700△USING△"W,#";B(*)
```

1~10ページのデータを設定します。

■ Note　　本器はNR部の各値から、DMA転送するパターン・データの必要なバイト数及び入力先頭アドレスを定義し、DMAモードの切替、内部RAMエリアへの格納アドレスを定義します。
パターン先頭アドレスと設定される実際のページとの関係は、
(パターン先頭アドレス＋1)＝実際のページ数
となります。なお、パターン・データの転送終了後はDMAモードを解除します。
パターン・データのDMA転送については、付録のパターン・データのDMA転送を参照してください。

50) RED?　　パターン・データ出力バイト数 (pattern data REaD ?)

■ 機　能

パターンデータのDMA転送を使用した読みとりを行うバイト数とスタートアドレスを設定します。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
RED	無　し	RED?△m1,m2	データパターン列　(m1による)

■ mの値

m1 : パターン転送バイト数

数値範囲	最大値	1048376
	最小値	1
	ステップ	1

m2 : パターン出力先頭アドレス

数値範囲	最大値	524288
	最小値	0
	ステップ	1

発生パターンがALTERNATEパターンの場合は、上記最大値の半分となります。

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限

次の設定条件の場合は、無効となりERR(CR/LF)を出力します。

問い合わせ : 発生パターンがZERO SUBSTもしくはPRBSの場合
パターン転送バイト数＋パターン先頭アドレス×2>1048376の場合

■ 使用例

問い合わせ : 発生パターンがDATAの場合で1ページから10ページまでのデータを設定する場合

```
DIM△B(9)
OUTPUT△700;"RED?△20,0"
ENTER△700△USING△"W";B(*)
PRINT△B(*)
```

1~10ページのデータを印字します。

■ Note

本器はNR部の各値から、DMA転送するパターン・データの必要なバイト数及び入力先頭アドレスを定義し、DMAモードの切替、内部RAMエリアへの格納アドレスを定義します。

パターン先頭アドレスと設定される実際のページとの関係は、
(パターン先頭アドレス＋1)＝実際のページ数
となります。なお、パターン・データの転送終了後はDMAモードを解除します。

パターン・データのDMA転送については、付録のパターン・データのDMA転送を参照してください。

51) PLL?　　PLLロック状態 (PLL unlock?)

■ 機　能　　内蔵シンセサイザのPLLがロック状態か、アンロック状態かを調べます。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
無し	無　し	PLL?	PLL△m　(FIX 1)

■ mの値　　0 : ロック状態
1 : アンロック状態

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となり、ERR(CR/LF)を出力します。

問い合わせ : オプション01未搭載の場合

■ 使用例　　問い合わせ : 内蔵シンセサイザがアンロック状態の場合

```
OUTPUT△700;"PLL?"
ENTER△700;B$
PRINT△B$
    ↓
PLL△1 (CR/LF) を出力します。
```

: オプション01未搭載の場合

```
OUTPUT△700;"PLL?"
ENTER△700;B$
PRINT△B$
    ↓
ERR (CR/LF) を出力します。
```

52) RTM　　内部タイマ設定 (Real TiMe setting)

■ 機　能　　内蔵のタイマ設定を行います。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
RTM	RTM△m1,m2,m3,m4,m5,m6	RTM?	RTM△m1,m2,m3,m4,m5,m6 (各FIX 2)

■ mの値　　設定ステップはm1~m6まで、全て1です。

		機能	設定範囲	
m1	:	年	0~99	(西暦)
m2	:	月	1~12	
m3	:	日	1~31	(閏年:1~29)
m4	:	時	0~23	
m5	:	分	0~59	
m6	:	秒	0~59	

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム : フロッピー・ディスクをアクセス中の場合

問い合わせ : 無し

■ 使用例

プログラム : 67年5月28日8時23分45秒に設定する場合

```
OUTPUT△700;"RTM△67, 5,28, 8,23, 45"
```

設定終了後タイマがスタートします。

問い合わせ : 内部タイマが94年4月23日11時30分0秒の場合

```
OUTPUT△700;"RTM?"
ENTER△700;B$
PRINT△B$
```

↓

RTM△94,△4,23,11,30,△0(CR/LF)を出力します。

53) PWI　　電源断、回復状態 (PoWer fail Interval)

■ 機　能　　電源断時刻、回復時刻、およびインターバル時間を出力します。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
無し	無　し	PWI?	PWF△m1,m2,m3,m4,m5,m6, PWR△m1,m2,m3,m4,m5,m6, PWI△△△△m7,m4,m5,m6 (CR/LF)

■ mの値　　PWFは電源断時刻を年月日時分秒で表します。
PWRは回復時刻を年月日時分秒で表します。
PWIは電源断インターバルを日時分秒で表します。
m1~m6は、2文字、m7は5文字固定です。

		機能	設定範囲	
m1	:	年	0~99	(西暦)
m2	:	月	1~12	
m3	:	日	1~31	(閏年:1~29)
m4	:	時	0~23	
m5	:	分	0~59	
m6	:	秒	0~59	
m7	:	日	0~99999	

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となり、ERR(CR/LF)を出力します。

問い合わせ : デバイス・クリア受信後の場合
イニシャライズ後の場合

■ 使用例　　問い合わせ :
```
OUTPUT△700;"PWI?"
ENTER△700;B$
PRINT△B$
   ↓
PWI△95,04,14,00,00,00,
PWR△95,04,15,12,00,00,
PWI△△△△△△△△1,12,00,00 (CR/LF+EOI) を出力します。
```

: デバイスクリア受信後または、イニシャライズ後の場合
```
OUTPUT△700;"PWI?"
ENTER△700;B$
PRINT△B$
        ↓
ERR(CR/LF) を出力します。
```

■ Note　　問い合わせに対するレスポンスは、改行されずに一つの文字列で出力されます。

54) DLY　　ディレイ状態 (DeLaY unlock?)

■ 機　能　　クロック遅延回路のサーボ回路がREADY状態か、BUSY状態かを調べます。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
無し	無　し	DLY?	DLY△m　(FIX 1)

■ mの値　　Ø : READY状態
1 : BUSY状態

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となり、ERR(CR/LF)を出力します。

問い合わせ : オプション03搭載時、表示が1/4SPEEDの場合

■ 使用例　　問い合わせ : クロック遅延回路のサーボ回路がREADY状態の場合

```
OUTPUT△7ØØ;"DLY?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

DLY△Ø(CR/LF)を出力します。

: 表示が1/4SPEEDの場合

```
OUTPUT△7ØØ;"DLY?"
ENTER△7ØØ;B$
PRINT△B$
```

↓

ERR(CR/LF)を出力します。

55) TRM　　終端コード切り替え (TeRMination select)

■ 機　能　　データリクエストコマンドに対するレスポンスデータの終端コードを切り替えます。

ヘッダ	プログラム	問い合わせ	レスポンス　(文字数)
TRM	TRM△m	TRM?	TRM△m　(FIX 1)

■ mの値　　Ø : LF+EOI
1 : CR+LF+EOI
※イニシャライズ時、電源投入時はTRM 0状態で起動します。

■ コマンド種別　シーケンシャル・コマンド

■ 使用制限　　次の設定条件の場合は、無効となります。

プログラム : フロッピー・ディスクをアクセス中の場合

問い合わせ : 無し

■ 使用例　　プログラム : レスポンスデータの終端コードをLF+EOIに設定を行う場合

```
OUTPUT△700;"TRM△Ø"
```

問い合わせ : レスポンスデータの終端コードがCR+LF+EOIに設定されている場合

```
OUTPUT△700;"LGC?"
ENTER△700;B$
PRINT△B$
```

↓

TRM△1 (CR+LF+EOI) を出力します。

10　章
プログラム作成例

目　　次

（空白）

10章
プログラム作成例

この章では、MP1761Cにおけるプログラム作成例を示します。

ホストコントローラとして、IBM-PC/ATまたはその互換機にナショナル・インスツルメント社(N.I.)社製GPIBインタフェース・カードを付加したものを想定して作成しました。

記述言語は、マイクロ・ソフト社QUICK-BASICです。

プログラムの検証は、DECpc, N.I.社GPIBインタフェース・カード, QUICK-BASIC V4.50で行いました。

プログラム例は、

(1)　周波数の設定1(ポイント設定)

(2)　周波数の設定2(一定範囲の設定)

(3)　発生パターンの設定

(4)　出力信号の設定

(5)　フロッピー・ディスクからのファイル情報読みだし

(6)　フロッピー・ディスクの操作

(7)　ステータス・バイトのチェック

(8)　パターン・データのDMA転送

プログラムの実行にあたっての準備を、表10-1に示します。

表10-1　サンプル・プログラムの実行にあたっての準備 (1 / 2)

使用コントローラ	プログラム実行の準備
DEC pc	● 本器の GPIB アドレスを "1" に設定します。 ● IBCONF の設定の一例 (但し、御使用される機器の状態によって設定は異なります。) ① <Board Characteristics> Board : GPIB Ø (Board を "GPIB Ø" とします。) Primary GPIB Address　0 Secondary GPIB Address　NONE Timeout setting　1000 sec Terminate Read on EOS　Yes Set EOI with EOS on Writes　Yes Type of Compare on EOS　7-Bit EOS byte　ØAH Send EOI at end of Write　Yes System Controller　Yes Assert REN when SC　No Enable Auto Serial Polling　Yes Enable CIC Protocol　No Bus timing　500 nsec Cable Length for High Speed　off Parallel Poll Duration　Default Use this GPIB interface　Yes Base I / O Address　Ø2CØh Interrupt Level　11 DMA Channel　5 DMA Transfer Mode　Demand

表10-1　サンプル・プログラムの実行にあたっての準備 (2 / 2)

使用コントローラ	プログラム実行の準備
DEC pc	②　＜Device Characteristics＞ Device : PPG (デバイス名を "PPG" とします。) Primary GPIB Address　1 Secondary GPIB Address　NONE Timeout setting　1000 sec Serial Poll Timeout　1 sec Terminate Read on EOS　Yes Set EOI with EOS on Writes　Yes Type of compare on EOS　7-Bit EOS byte　ØAh Send EOI at end of Write　Yes Enable Repeat Addressing　No ③　GPIB Device Map で②のデバイスを①の GPIB Ø に接続します。 ● 本器と DEC pc を GPIB ケーブルで接続します。

DECpcを用いたサンプル・プログラム例

<プログラム共通部分の説明>

Quick Basic Ver4.50およびナショナル・インストゥル・メント社製GPIBボードを使用して、以降のサンプルプログラムは作成されています。(☞ Quick BasicおよびNI社製GPIBドライバに関する詳細は、それぞれの取扱説明書を参照してください。)

サンプル・プログラムで必要な共通関数を下記の2種類のプログラムにまとめてあります。

- COMMON.BAS
- ACS_GPIB.BAS

サンプル・プログラムを実行する際には、上記のプログラムが必要となるので前もって準備しておいてください。

また、使用されるアプリケーションに必要な関数のみを作成されてもけっこうです。

2種類の共通関数について、まず説明します。

＜COMMON.BASの説明＞

COMMON.BASは下記の7種類の関数で構成されています。

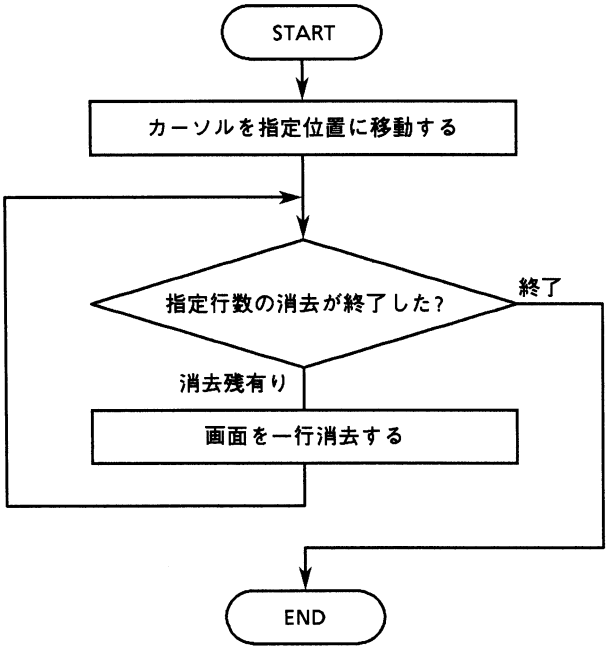
表10-2　COMMON.BASの関数表

モジュール番号	関　数　名	処　　理　　概　　要
1.1	SUB ClearDisp (p%, l%)	画面を行単位で消去する p% : 消去開始行番号 l% : 消去する行数
1.2	FUNC Exchange% (i%)	単精度整数をビット・パターンとするデータの上位、下位をバイト単位で交換する i% : ビット・パターン・データ
1.3	FUNC itob$ (l%, v%)	単精度整数をLSBから指定されたビット長の2進文字列に変換する ただし、出力文字長は16文字固定とする。 l% : 2進文字列長 v% : 変換データ
1.4	SUB waidly (tim)	指定時間 (秒) waitする tim : 指定時間 (秒) (入力)

次ページ以降に、各関数のフローチャートとプログラム・リストを掲載します。

(1.1)　SUB ClearDisp (p%, l%) : 画面を消去する

● フローチャート

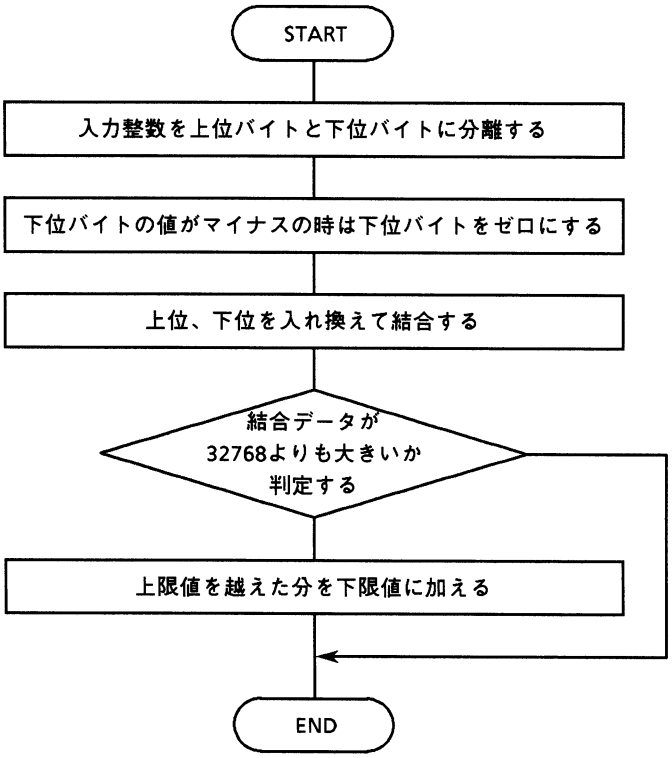


● プログラム・リスト

```
' ---- Procedure for Clear display ----
' in    p%:Location line number
'       l%:clear line count
'
SUB ClearDisp (p%, l%)
    LOCATE p%, 1
    FOR i% = 0 TO (l% - 1)
        PRINT "
            "
    NEXT i%
END SUB
```

(1.2)　FUNCTION Exchange (i%) : 16-BIT整数データをバイト単位で交換する

● フローチャート

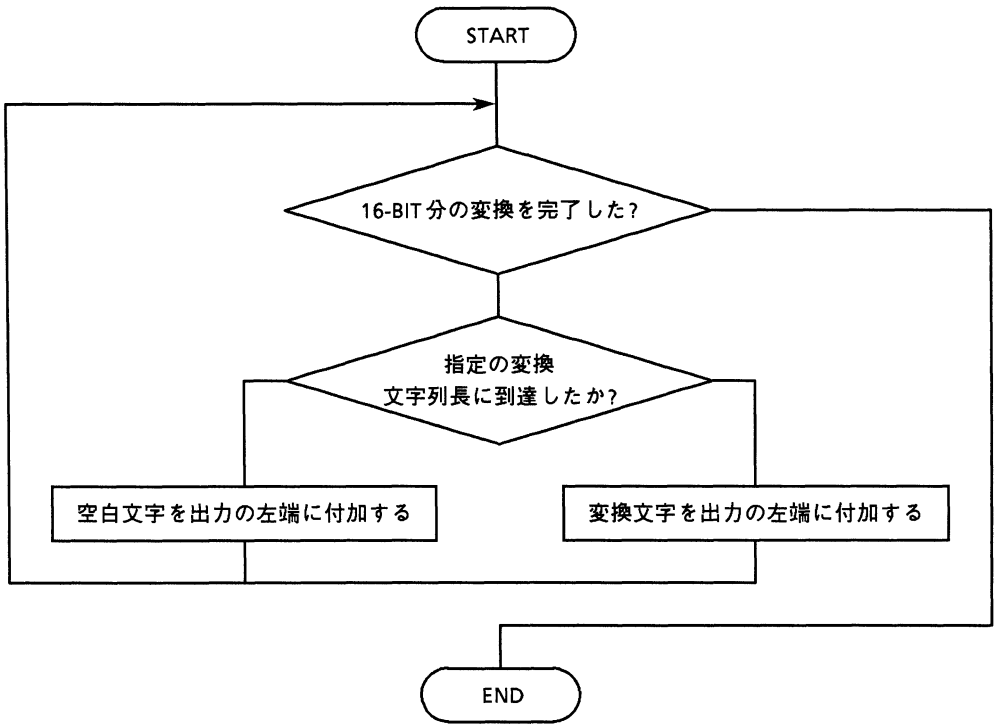


● プログラム・リスト

```
' ---- Exchange 16-bits pattern data ----
' In   i%:16bits pattern data (used integer)
'
' Procedure for swap of low byte and high byte .
' This program is bit manipulation of integer valu. Why this program used
' real value because one is overflow detect on bit manipulation of integer
' value, another one is internal manipulation by real value although input
' parameter is integer. And integer declear value is same operation.
'
FUNCTION Exchange% (i%)
    h = i% AND &HFF
    l = i% AND &HFF00
    IF h < 0 THEN
        h = 0
    END IF

    a = INT(h * 256) + ((l ¥ 256) AND &HFF)
    IF a >= 32768 THEN
        b = a - 32768
        a = -32768 + b
    END IF
    Exchange% = a
END FUNCTION
```

(1.3)　itob$ (l%, v%) : 整数を2進文字列に変換する

● フローチャート

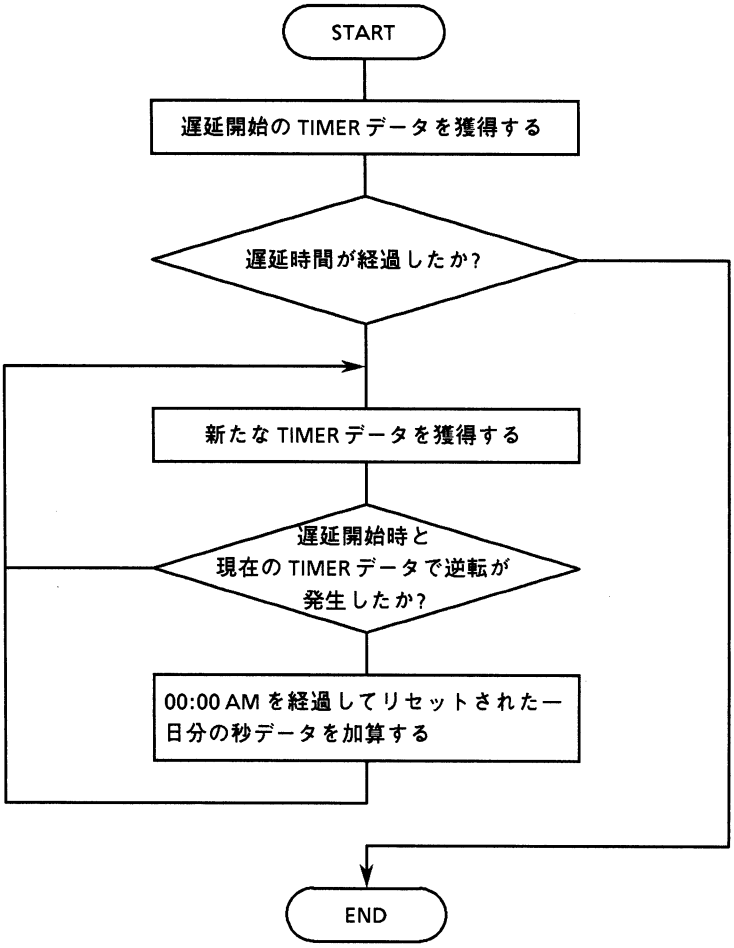


● プログラム・リスト

```
' ---- Interger convert to binary strings ----
' in    l%:convert binary length
'       v%:convert value
'
' This convert binary strings is always possession 16-character field.
'
FUNCTION itob$ (l%, v%)
b$ = ""
FOR i% = 1 TO 16
    IF i% <= l% THEN
        IF v% AND &H1 THEN
            b$ = "1" + b$
        ELSE
            b$ = "0" + b$
        END IF
        v% = INT(v% / 2)
    ELSE
        b$ = " " + b$
    END IF
NEXT i%
itob$ = b$
END FUNCTION
```

(1.4)　waidly (tim!) : 遅延時間を作る

● フローチャート



● プログラム・リスト

```
' ---- Make a timing delay ----
'in    tim:wait time length (unit is seconds)
'
SUB waidly (tim)
    stm = TIMER
    etm = TIMER
    WHILE etm - stm < tim
        etm = TIMER
        IF etm < stm THEN etm = etm + 86400
    WEND
END SUB
```

<ACS_GPIB.BASの説明>

ACS_GPIB.BASは下記の15種類の関数で構成されています。

表10-3　ACS_GPIB.BASの関数表　(1/2)

モジュール番号	関数名	処理概要
2.1	SUB wrtcmd1 (w$)	PPGにコマンドを発行する w$: 発行するコマンド文字列 (入力)
2.2	FUNC readcmd1$ ( )	EDからメッセージをリードする readcmd1$: メッセージの文字列 (戻り値)
2.3	SUB dmawrt (w% ( ) , i%)	PPGへのDMA転送処理をする w% ( ) : 転送するパターン・データの整数配列 i% : 整数配列の要素数
2.4	SUB EndPoll ( )	MSSステータスレジスタのENDビットをポーリングする
2.5	SUB SRQPoll ( )	MSSステータスレジスタのERRORビットおよびSRQビットをポーリングする
2.6	SUB StatusMask (sre%, ese%, ese1%, ese2%)	ステイタス・レジスタ、イベント・レジスタおよび拡張イベント・レジスタのマスク・レジスタを設定する sre% : ステイタス・レジスタのマスク・パターン ese% : 標準イベント・レジスタのマスク・パターン ese1%: 拡張イベント・レジスタ-1のマスク・パターン ese2%: 拡張イベント・レジスタ-2のマスク・パターン
2.7	SUB StatusDisp (stb%, esr%, esr1%, esr2%)	ステイタス・レジスタ、イベント・レジスタおよび拡張イベント・レジスタの設定状態を表示する、読み出したデータは引き数に設定され呼出側に渡される stb% : ステイタス・レジスタの設定状態パターン esr% : 標準イベント・レジスタの設定状態パターン esr1%: 拡張イベント・レジスタ-1の設定状態パターン esr2%: 拡張イベント・レジスタ-2の設定状態パターン

表10-3　ACS_GPIB.BASの関数表　(2/2)

モジュール番号	関　数　名	処　　理　　概　　要
2.8	FUNC gpinit%()	GPIBの初期化を実行し、その結果を関数値として返す 0(False):設定に異常が有り、GPIBの初期化に失敗した 1(True):PPGの初期化が完了した
2.9	SUB trap()	システムエラー時の処理を行う
2.10	SUB gpiberr()	National Instruments Cop.の提供するGPIBサンプル・プログラムに含まれる内部エラー、ステイタス情報の表示処理

次ページ以降に各関数のフローチャートとプログラム・リストを掲載します。

また本モジュールの先頭に、

```
REM $INCLUDE:'C:¥at-gbib¥qbasic¥qbdecl.bas'..... ①
COMMON SHARED DEV%,GPIBØ%,PPG% ..... ②
```

と記述してください。

①は、NI社のGPIBドライバで提きょうされているNI-488関数の定義をロードしています。

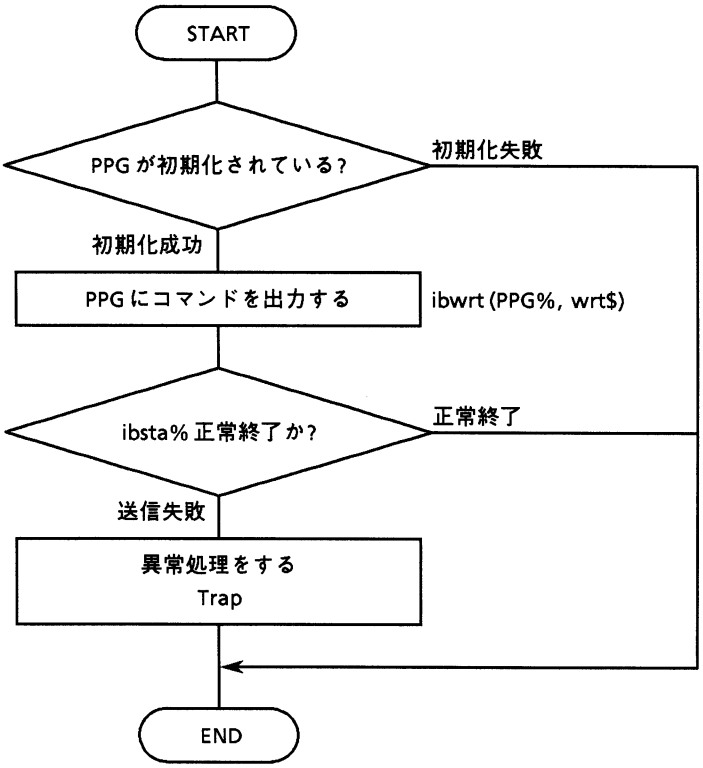
実際に御使用される為は、'qbdecl. bas'が存在しているディレクトリを指定してください。

②は、Quick Basicのステートメントで、複数モジュール間で共有する変数の定義をしています。

■ Note：　①に関してはお使いになられるGPIBドライバおよび使用環境によって異なりますのでご注意ください。

(2.1)　SUB wrtcmd1 (w$) : PPG にコマンドを送信する

- フローチャート

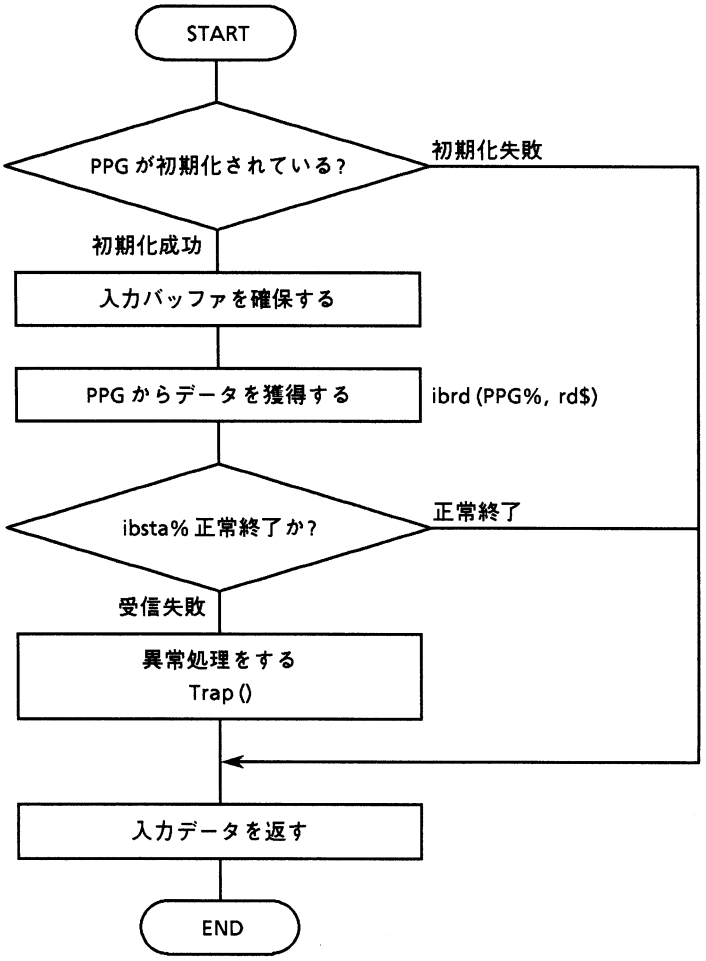


- プログラム・リスト

```
' ---- Procedure for command write to PPG ----
'
SUB wrtcmd1 (WRT$)
    IF DEV% = 1 OR DEV% = 3 THEN
        WRT$ = WRT$ + CHR$(13) + CHR$(10)
        CALL IBWRT(PPG%, WRT$)              'write command(ppg)
        IF IBSTA% < 0 THEN CALL trap        'call trap if illegal end
    END IF
END SUB
```

(2.2) FUNCTION readcmd1$ () : PPGから別に送信したコマンドに対応するデータを獲得する

● フローチャート

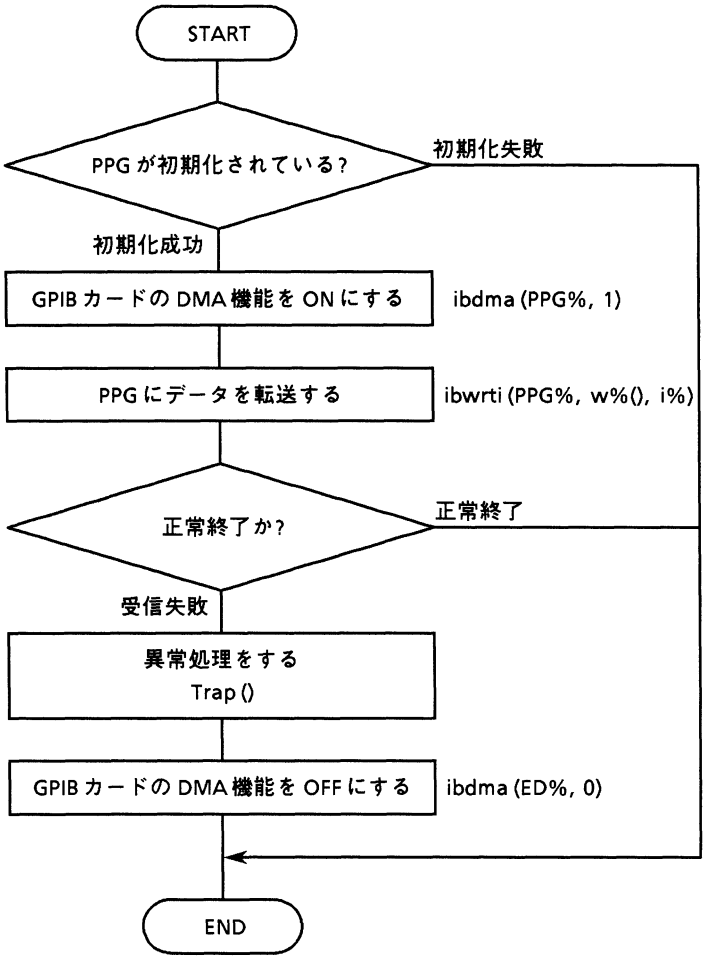


● プログラム・リスト

```
' ---- Procedure for data read from PPG ----
'
FUNCTION readcmd1$
        IF DEV% = 1 OR DEV% = 3 THEN
                r$ = SPACE$(256)
                CALL IBRD(PPG%, r$)              ' Read data from PPG%
                IF IBSTA% < 0 THEN CALL trap
'
                readcmd1$ = r$
        END IF
END FUNCTION
```

(2.3)　SUB dmawrt (w%, i%) : PPG にデータを DMA 転送する

● フローチャート

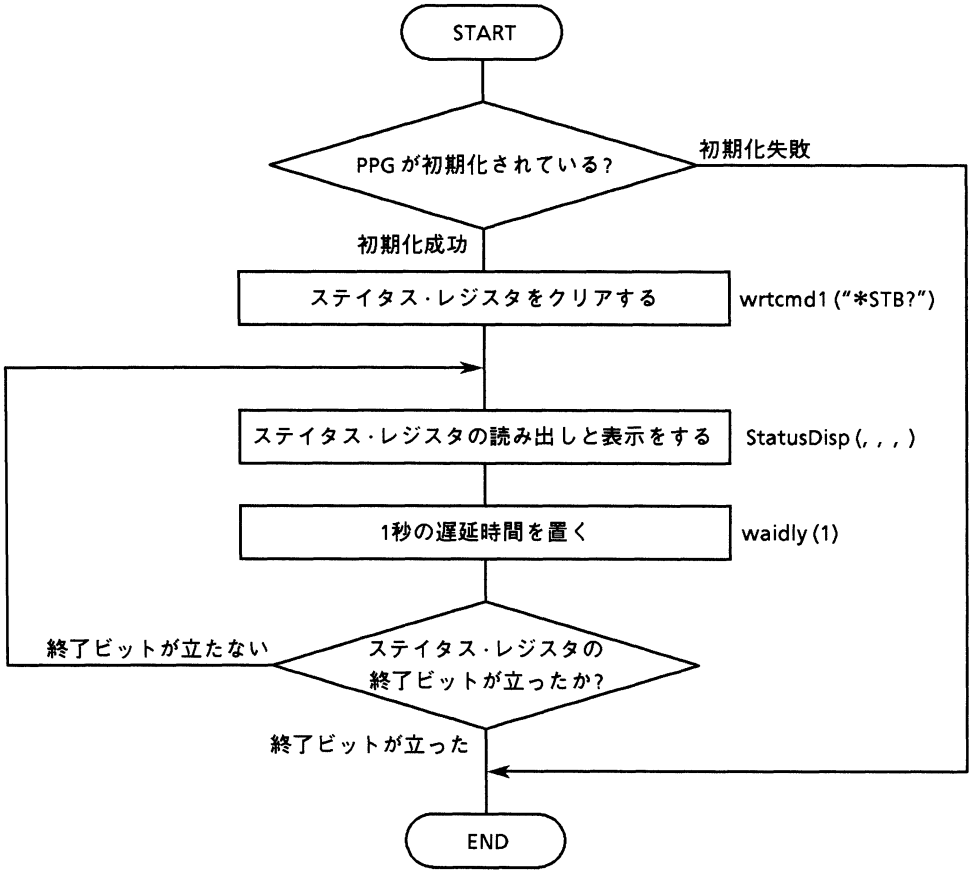


● プログラム・リスト

```
' --- Procedure for DMA transfer ----
' in    w%():Transmit data pattern of integer arry
'       i%  :length count for integer arry
'
SUB dmawrt (w%(), i%)
         IF DEV% = 1 OR DEV% = 3 THEN
                  CALL IBDMA(PPG%, 1)          ' DMA enable

                  i% = i% * 2 + 1              ' make up to a byte count
                  CALL IBWRTI(PPG%, w%(), i%)
                  IF IBSTA% < 0 THEN CALL trap ' call trap if illegal end

                  CALL IBDMA(PPG%, 0)          ' DMA disable
         END IF
END SUB
```

(2.4)　SUB EndPoll () : ステイタスの終了ビットが設定される迄待つ

● フローチャート

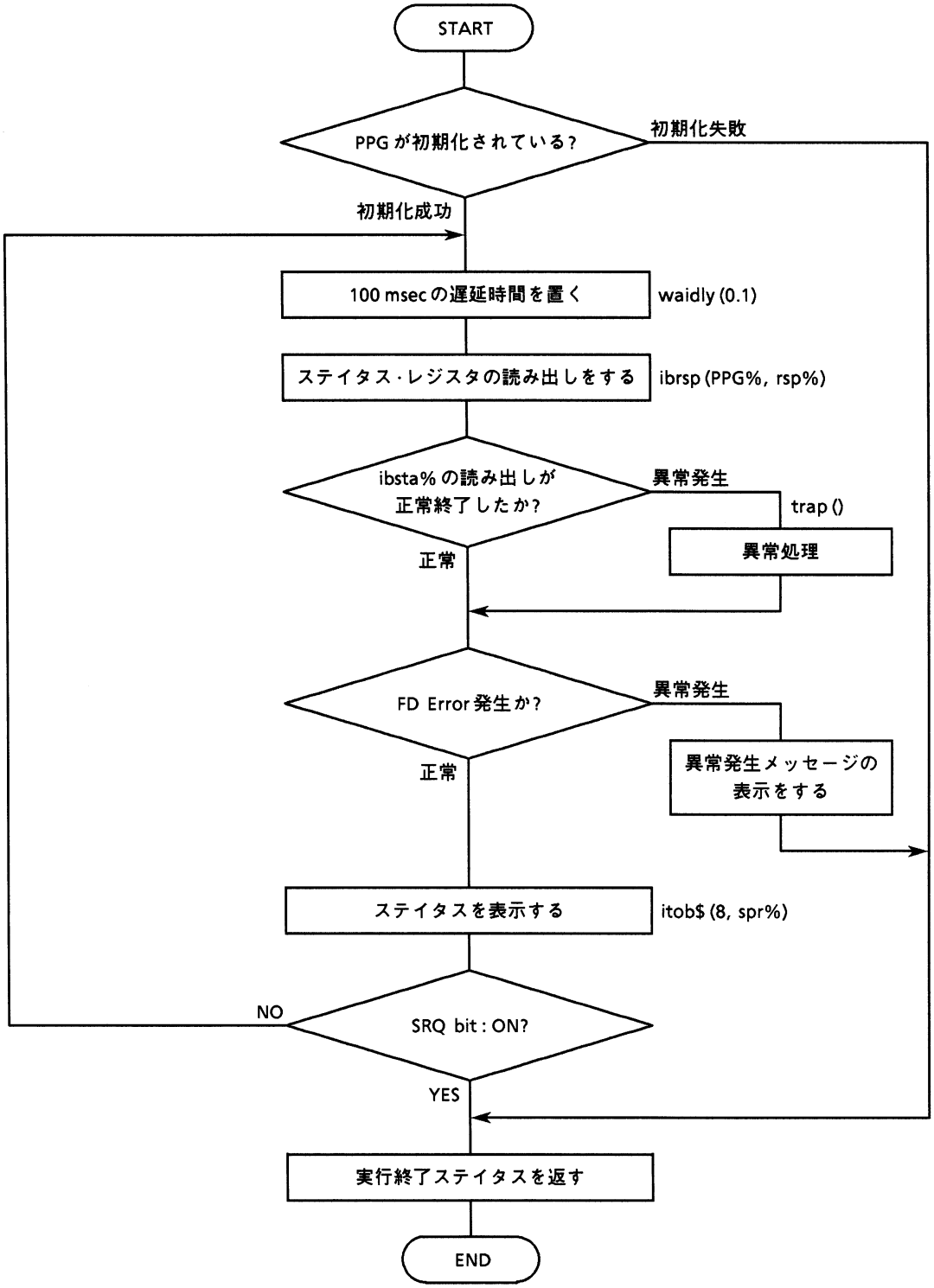


● プログラム・リスト

```
' ---- Procedure for judgement of Measurement end ----
'
SUB EndPoll
        IF DEV% = 1 OR DEV% = 3 THEN
                CALL wrtcmd1("*STB?")              ' reset event flag
                RD$ = LEFT$(readcmd1$, IBCNT% - 1)

                DO
                        CALL StatusDisp(reg%, dmy%, dmy2%, dmy3%)

                        waidly (1)
                LOOP UNTIL reg% AND &H4
        END IF
END SUB
```

(2.5)　FUNCTION SRQPoll () : SRQ bit と Error bit の判定をする

● フローチャート

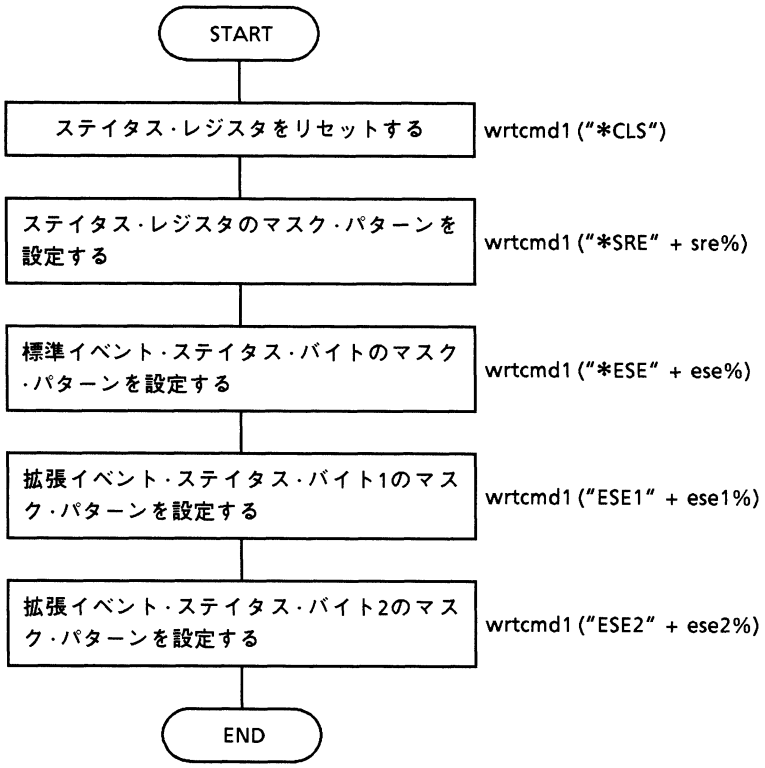


● プログラム・リスト

```
' ---- Procedure for Seliall poll with SRQ bit ----
'
FUNCTION SRQPoll%
        IF DEV% = 1 OR DEV% = 3 THEN
                exe% = 1
                DO
                        waidly (.1)

                        CALL IBRSP(PPG%, SPR%)
                        IF IBSTA < 0 THEN CALL trap
                        srq = SPR% AND &H40
                        esr1 = SPR% AND &H4
                        esr2 = SPR% AND &H8

                        IF esr2 = &H8 THEN      'Output warning message,if
                                                 error detect
                                LOCATE 12, 35
                                PRINT "FD error detect!!"
                                exe% = 0
                                EXIT DO
                        END IF

                        sta$ = itob$(8, SPR%)
                        LOCATE 1, 60
                        PRINT "*STB:"; sta$
                LOOP UNTIL srq = &H40 AND esr1 = &H4
        END IF
        '
        SRQPoll% = exe%
END FUNCTION
```

(2.6) SUB StatusMask (sre%, ese%, ese1%, ese2%) : ステイタス・マスク・レジスタの設定

● フローチャート

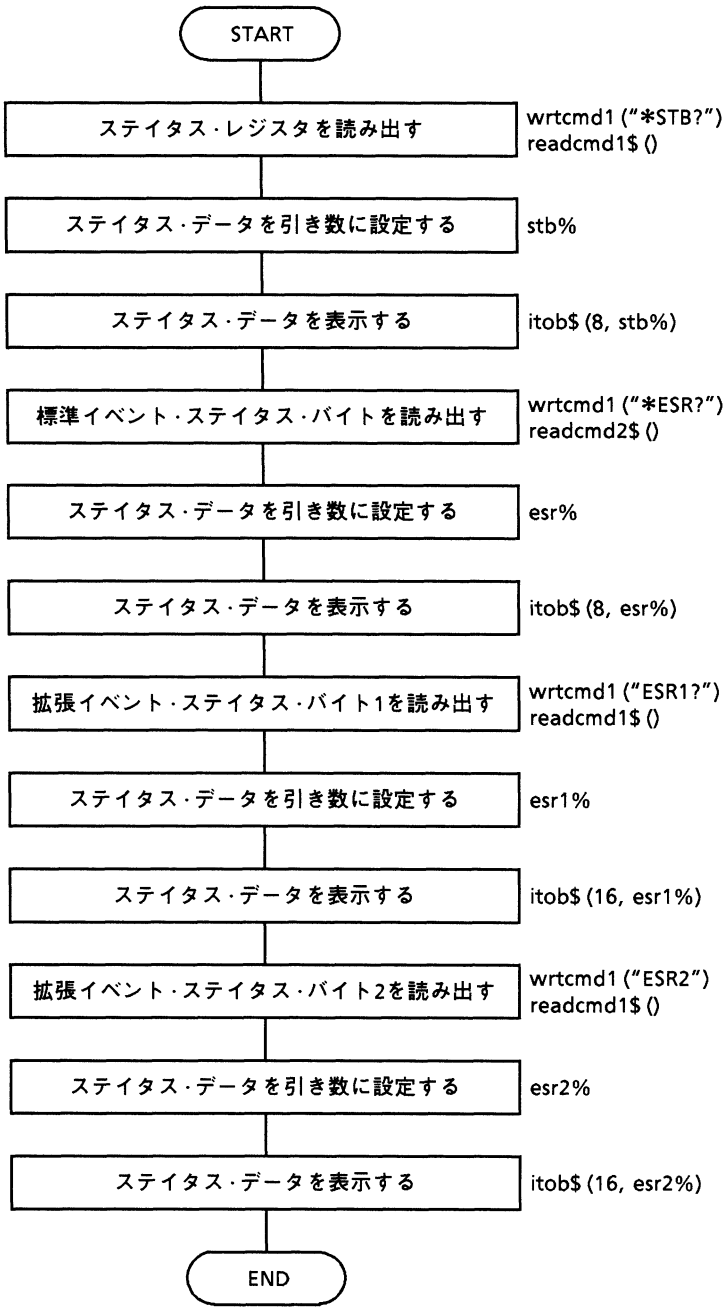


● プログラム・リスト

```
' ---- Procedure for set status mask pattern ----
' in    s0%:status byte enable register mask pattern
'       s1%:normal event status enable register mask pattern
'       s2%:Extend event status enable register-1 mask pattern
'       s3%:Extend event status enable register-2 mask pattern
'
SUB StatusMask (s0%, s1%, s2%, s3%)
         wrtcmd1 ("*CLS")
         wrtcmd1 ("*SRE " + STR$(s0%))
         wrtcmd1 ("*ESE " + STR$(s1%))
         wrtcmd1 ("ESE1 " + STR$(s2%))
         wrtcmd1 ("ESE2 " + STR$(s3%))
END SUB
```

(2.7)　SUB StatusDisp (stb%, esr%, esr1%, esr2%) :
ステイタス・レジスタの読み出しおよび表示をする

● フローチャート

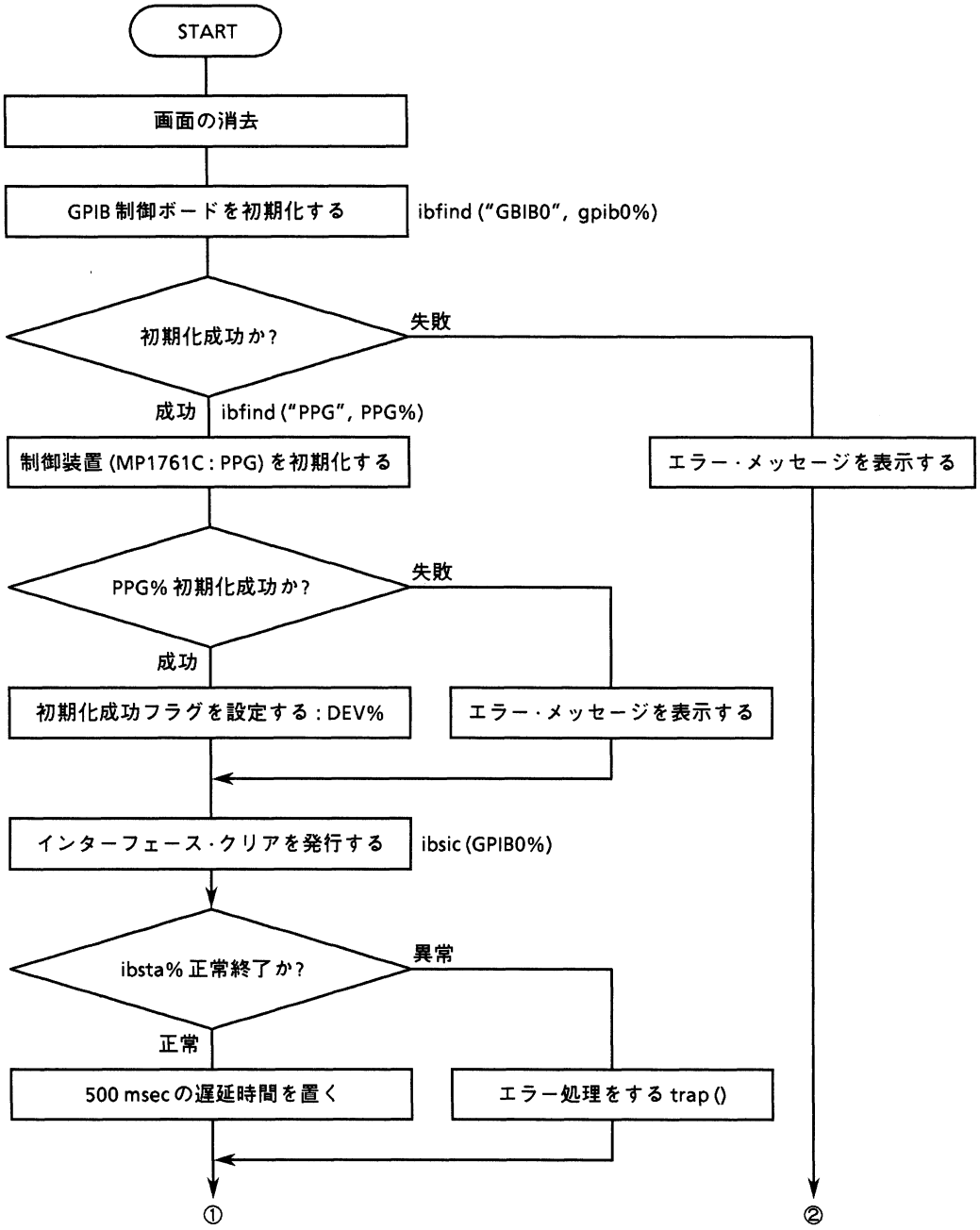


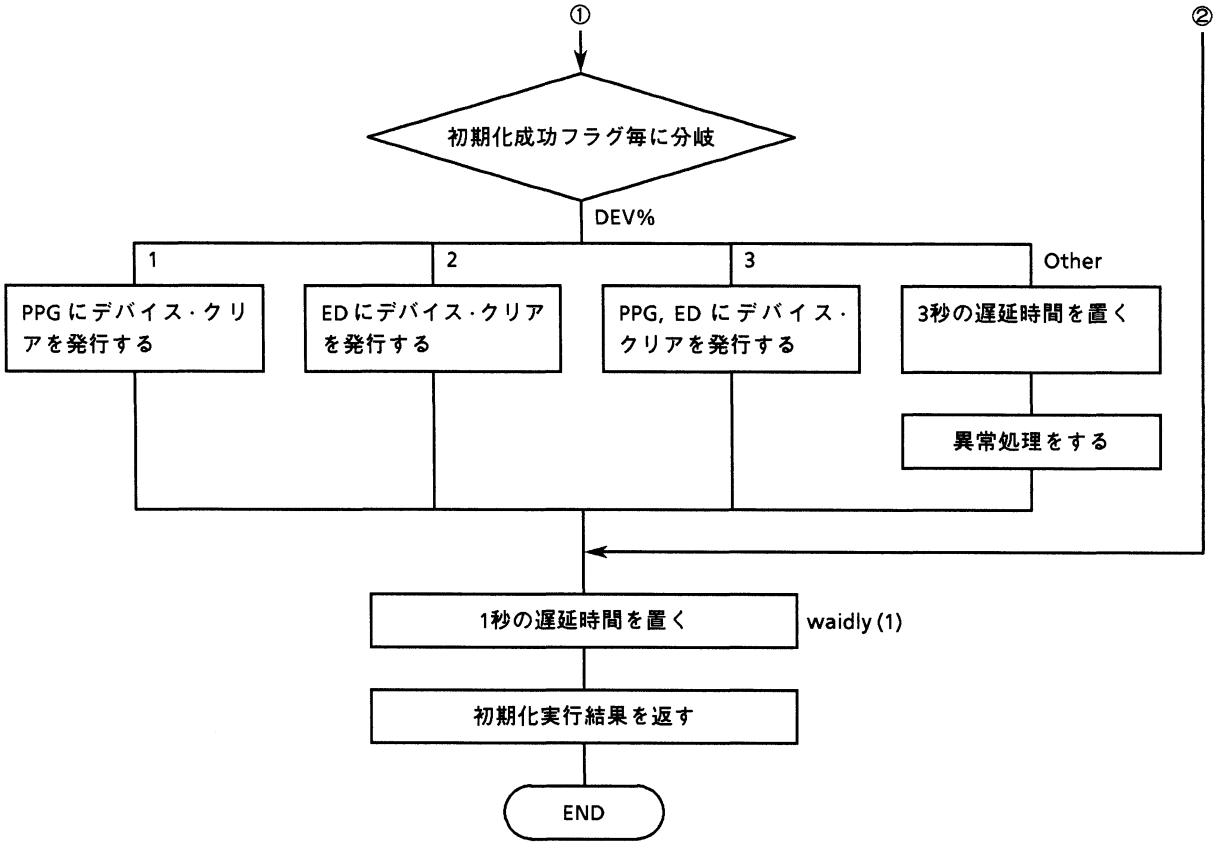
● プログラム・リスト

```
' ---- Procedure for status byte display ----
' out   stb% :Status byte
'       esr% :Normal event status byte
'       esr1%:Extend event-1 status byte
'       esr2%:Extend event-2 status byte
'
SUB StatusDisp (stb%, esr%, esr1%, esr2%)
         CALL wrtcmd1("*STB?")
         RD$ = LEFT$(readcmd1$, IBCNT% - 1)
         stb% = VAL(RD$)
         sta$ = itob$(8, VAL(RD$))
         LOCATE 1, 60
         PRINT "*STB:"; sta$

         CALL wrtcmd1("*ESR?")
         RD$ = LEFT$(readcmd1$, IBCNT% - 1)
         esr% = VAL(RD$)
         sta$ = itob$(8, VAL(RD$))

         LOCATE 2, 60
         PRINT "*ESR:"; sta$

         CALL wrtcmd1("ESR1?")
         RD$ = LEFT$(readcmd1$, IBCNT% - 1)
         esr1% = VAL(MID$(RD$, 6, 5))
         sta$ = itob$(16, VAL(MID$(RD$, 6, 5)))
         LOCATE 3, 60
         PRINT "ESR1:"; sta$

         CALL wrtcmd1("ESR2?")
         RD$ = LEFT$(readcmd1$, IBCNT% - 1)
         esr2% = VAL(MID$(RD$, 6, 5))
         sta$ = itob$(16, VAL(MID$(RD$, 6, 5)))
         LOCATE 4, 60
         PRINT "ESR2:"; sta$
END SUB
```

(2.8)　FUNCTION gpinit () : GPIB 制御環境の初期化をする

● フローチャート



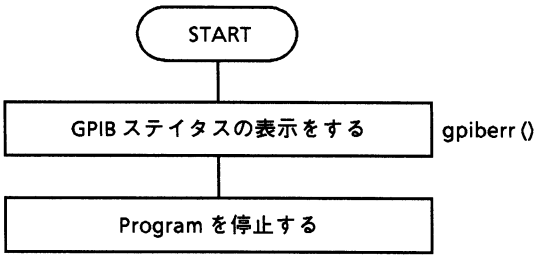

①
②
初期化成功フラグ毎に分岐
DEV%
1
2
3
Other
PPGにデバイス・クリアを発行する
EDにデバイス・クリアを発行する
PPG, EDにデバイス・クリアを発行する
3秒の遅延時間を置く
異常処理をする
1秒の遅延時間を置く
waidly (1)
初期化実行結果を返す
END

● プログラム・リスト

```
' ---- Procedure for initialize equipments and interface board ----
'
FUNCTION gpinit%
        CLS
        CALL IBFIND("GPIB0", GPIB0%)        'Open DEVice (GPIB0)
        IF GPIB0% < 0 THEN
                PRINT "Configration fail!!"
                PRINT "You need verify are hardware condition, and try
                again."
                ret% = 0
        ELSE
                CALL IBFIND("PPG", PPG%)        'Open DEVice (PPG)
                IF PPG% < 0 THEN
                        PRINT "Lost PPG address!!"
                        PRINT "If you use a PPG, then verify are
                        configration and environment."
                        DEV% = 0
                ELSE
                        DEV% = 1
                END IF
                '
                CALL IBSIC(GPIB0%)              'Interface clear
                IF IBSTA% < 0 THEN CALL trap
                '
                CALL waidly(.5)                 '500ms wait
                '
                SELECT CASE DEV%
                CASE 1
                        CALL IBCLR(PPG%)            'DEVice clear (PPG)
                CASE 2
                        CALL IBCLR(ED%)             'DEVice clear (ED)
                CASE 3
                        CALL IBCLR(PPG%)            'DEVice clear (PPG)
                        CALL IBCLR(ED%)             'DEVice clear (ED)
                CASE ELSE
                        waidly (3)
                        CALL trap
                END SELECT
                ret% = 1
        END IF

        waidly (1)
        CLS
        gpinit% = ret%                          ' set Execution status
END FUNCTION
```

(2.9)　SUB trap (msg$) : 異常処理をする

- フローチャート

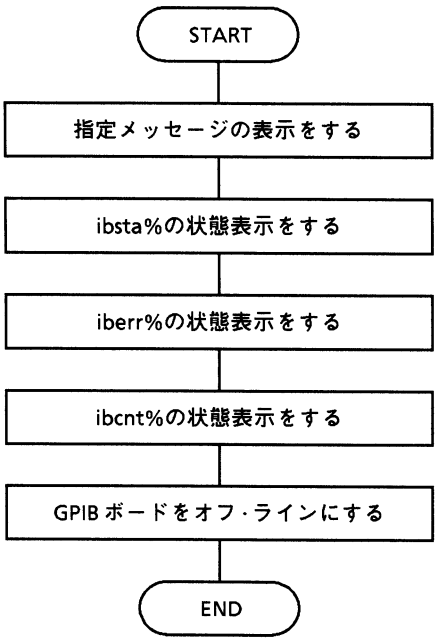


- プログラム・リスト

```
' ---- Procedure for illegal process trap ----
' This subroutine, call on illegal execution or fatal error detect.
' And, you will get are status condition by presented NI-488 function.
'
SUB trap
    CALL gpiberr("Program trap condition.")   ' call NI subroutine
    STOP
END SUB
```

(2.10) SUB gpiberr (msg$) STATIC : GPIB ステイタスの表示をする

- フローチャート



● プログラム・リスト

```
'=============================================================================
'                        Subroutine GPIBERR
'   This subroutine will notify you that a NI-488 function failed by printing
'   an error message.  The status variable IBSTA% will also be printed
'   in hexadecimal along with the mnemonic meaning of the bit position.
'   The status variable IBERR% will be printed in decimal along with the
'   mnemonic meaning of the decimal value.  The status variable IBCNT% will
'   be printed in decimal.
'
'   The NI-488 function IBONL is called to disable the hardware and software.
'
'   The STOP command will terminate this program.
'=============================================================================
'
SUB gpiberr (msg$) STATIC

   PRINT msg$

   PRINT "ibsta = &H"; HEX$(IBSTA%); " <";
   IF IBSTA% AND EERR THEN PRINT " ERR";
   IF IBSTA% AND TIMO THEN PRINT " TIMO";
   IF IBSTA% AND EEND THEN PRINT " END";
   IF IBSTA% AND SRQI THEN PRINT " SRQI";
   IF IBSTA% AND RQS THEN PRINT " RQS";
   IF IBSTA% AND SPOLL THEN PRINT " SPOLL";
   IF IBSTA% AND EEVENT THEN PRINT " EVENT";
   IF IBSTA% AND CMPL THEN PRINT " CMPL";
   IF IBSTA% AND LOK THEN PRINT " LOK";
   IF IBSTA% AND RREM THEN PRINT " REM";
   IF IBSTA% AND CIC THEN PRINT " CIC";
   IF IBSTA% AND AATN THEN PRINT " ATN";
   IF IBSTA% AND TACS THEN PRINT " TACS";
   IF IBSTA% AND LACS THEN PRINT " LACS";
   IF IBSTA% AND DTAS THEN PRINT " DTAS";
   IF IBSTA% AND DCAS THEN PRINT " DCAS";
   PRINT " >"

   PRINT "iberr = "; IBERR%;
   IF IBERR% = EDVR THEN PRINT " EDVR <DOS Error>"
   IF IBERR% = ECIC THEN PRINT " ECIC <Not CIC>"
   IF IBERR% = ENOL THEN PRINT " ENOL <No Listener>"
   IF IBERR% = EADR THEN PRINT " EADR <Address error>"
   IF IBERR% = EARG THEN PRINT " EARG <Invalid argument>"
   IF IBERR% = ESAC THEN PRINT " ESAC <Not Sys Ctrlr>"
   IF IBERR% = EABO THEN PRINT " EABO <Op. aborted>"
   IF IBERR% = ENEB THEN PRINT " ENEB <No GPIB board>"
   IF IBERR% = EOIP THEN PRINT " EOIP <Async I/O in prg>"
   IF IBERR% = ECAP THEN PRINT " ECAP <No capability>"
   IF IBERR% = EFSO THEN PRINT " EFSO <File sys. error>"
   IF IBERR% = EBUS THEN PRINT " EBUS <Command error>"
   IF IBERR% = ESTB THEN PRINT " ESTB <Status byte lost>"
   IF IBERR% = ESRQ THEN PRINT " ESRQ <SRQ stuck on>"
   IF IBERR% = ETAB THEN PRINT " ETAB <Table Overflow>"

   PRINT "ibcnt = "; IBCNT%

'  Call the IBONL function to disable the hardware and software.

   CALL IBONL(dvm%, 0)
END SUB
```

<プログラムの起動>

前述の共通関数と(1)～(8)のサンプル・プログラムを起動する手順を以下に記述します。

(手順1):メニューバーのFileをオープンして、"Load File. . . . "を選択します。
次にLoadするファイル名(共通関数の)COMMON.BASをロードします。

(手順2):手順1と同様にして、ACS_GPIB.BASをロードします。

(手順3):手順1と同様にして、サンプル・プログラムをロードします。

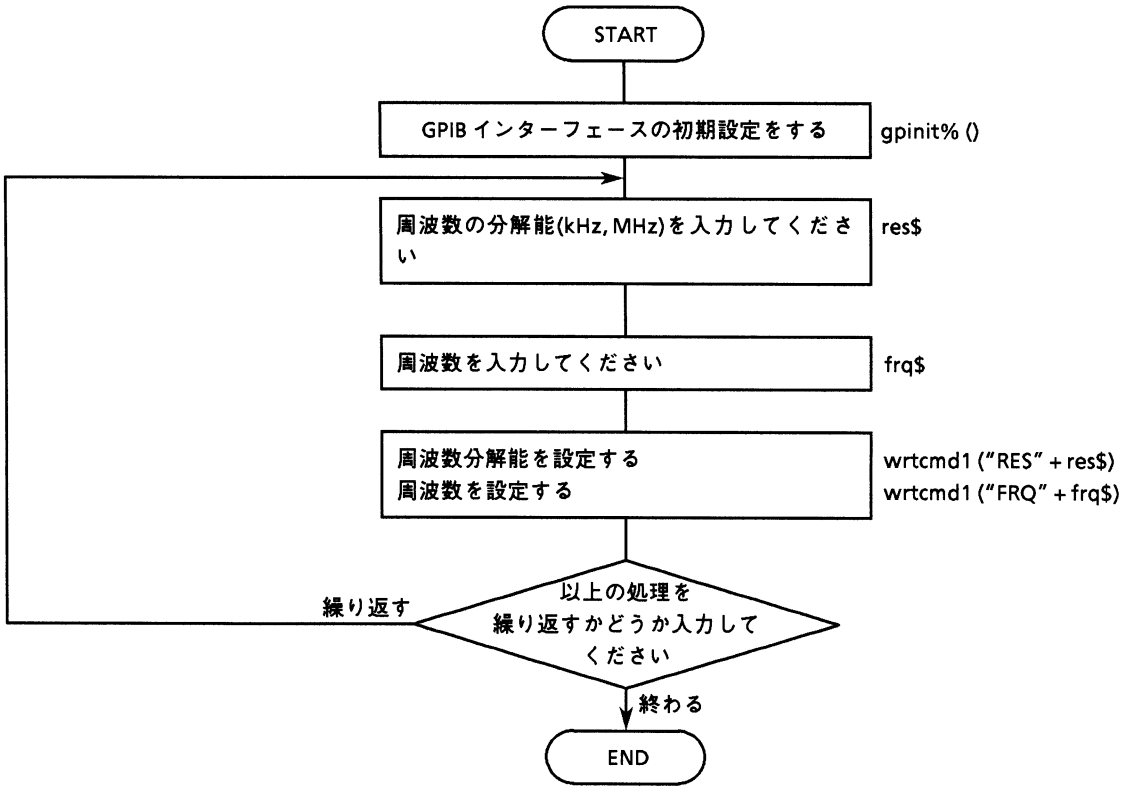
(手順4):メニューバーのRunをオープンして"Set Main Module"を選択し、手順3でロードしたサンプル・プログラムをMain Moduleとします。

(手順5):メニューバーのRunをオープンして"Start"を実行するとプログラムがスタートします。

(☞詳細はQuick Basicの取扱説明書を参照してください。)

(1)　周波数の設定1 (ポイントの設定)

本プログラムは内蔵シンセサイザの周波数および周波数の分解能を設定します。

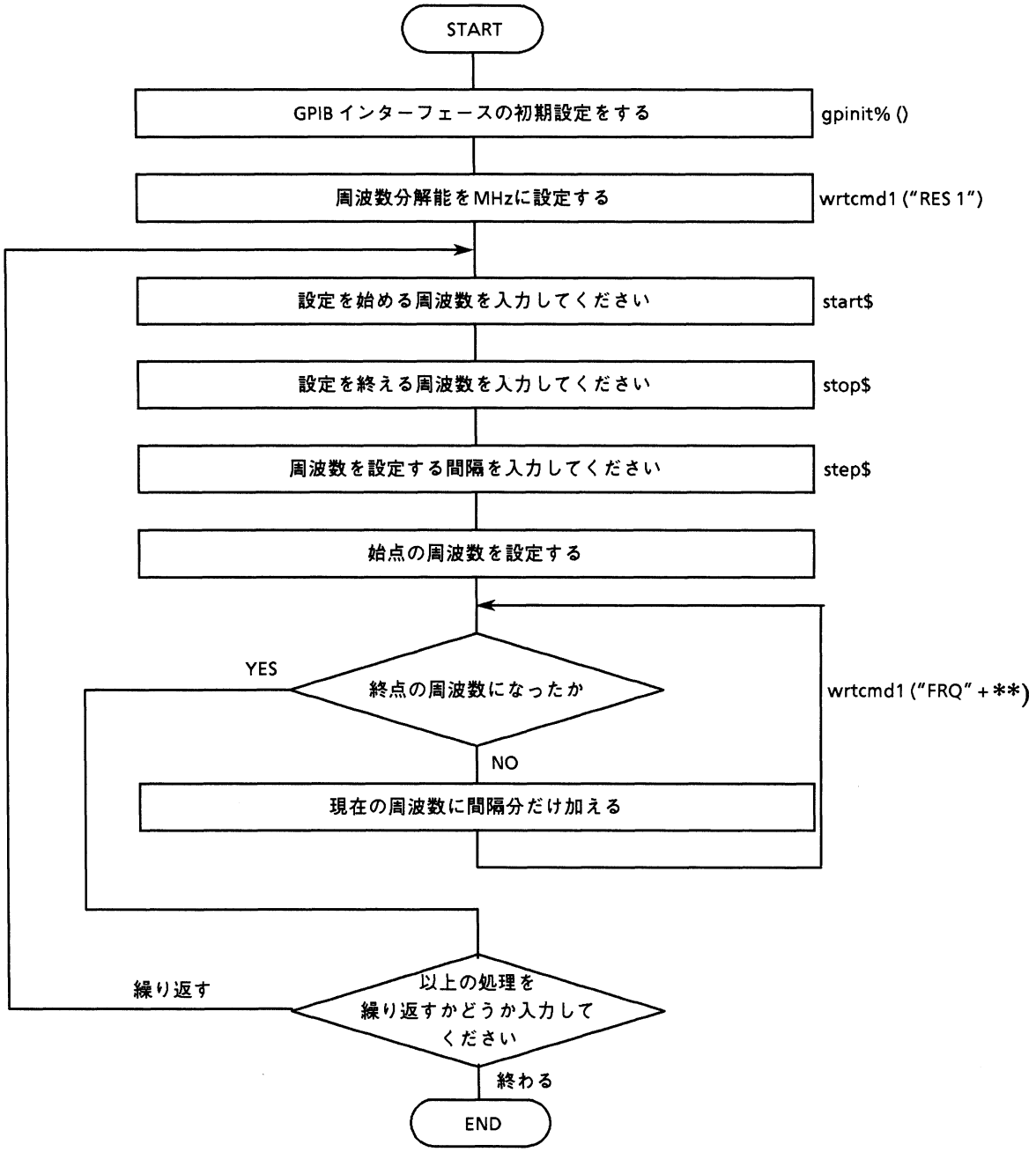


● プログラム・リスト

```
REM $INCLUDE: 'c:¥at-gpib¥qbasic¥qbdecl.bas'

COMMON SHARED DEV%, GPIB0%, PPG%

DECLARE SUB waidly (tim!)
DECLARE SUB wrtcmdl (w$)
DECLARE FUNCTION gpinit% ()

'*********************************************************
'*     MP1761C / MP1763C  FREQUENCY SAMPLE SOFT_1        *
'*********************************************************
'
'-------------------------------------------------------
'                    MAIN ROUTINE
'-------------------------------------------------------
'
CLS
IF gpinit% <> 0 THEN       'setup interface
'
   DO
            CLS
            LOCATE 3, 1
            GOSUB setfrql
            GOSUB dset
            '
            PRINT ""
            INPUT " SET NEXT DATA [ Y or N ] ?", loop$
            '
   LOOP UNTIL loop$ = "n" OR loop$ = "N"
END IF
STOP
'
'-------------------------------------------------------
'                    SUB ROUTINE
'-------------------------------------------------------
'
setfrql: '----------------------- INPUT DATA CONDITIONS
'
PRINT ""
PRINT " ****** MP1761C / MP1763C FREQUENCY_1 SAMPLE PROGRAM ******"
PRINT ""
'
INPUT " FREQUENCY RESOLUTION [ KHz = 0, MHz = 1 ]?", res$
PRINT ""
   IF res$ = "0" THEN
            INPUT " FREQUENCY DATA [ KHz : 50000 to 12500000 ] ?", frq$
   ELSE
            INPUT " FREQUENCY DATA [ MHz : 50 to 12500 ] ?", frq$
   END IF
RETURN
'
'
dset:    '----------------------------- OUTPUT DATA CONDITIONS
'
CALL wrtcmdl("RES " + res$)          '  Set Resolution mode
CALL wrtcmdl("FRQ " + frq$)          '  Set Frequency
RETURN
'
END
```

(2)　周波数の設定2 (一定範囲の設定)

本プログラムは指定された2点の周波数間を指定された間隔で設定します。

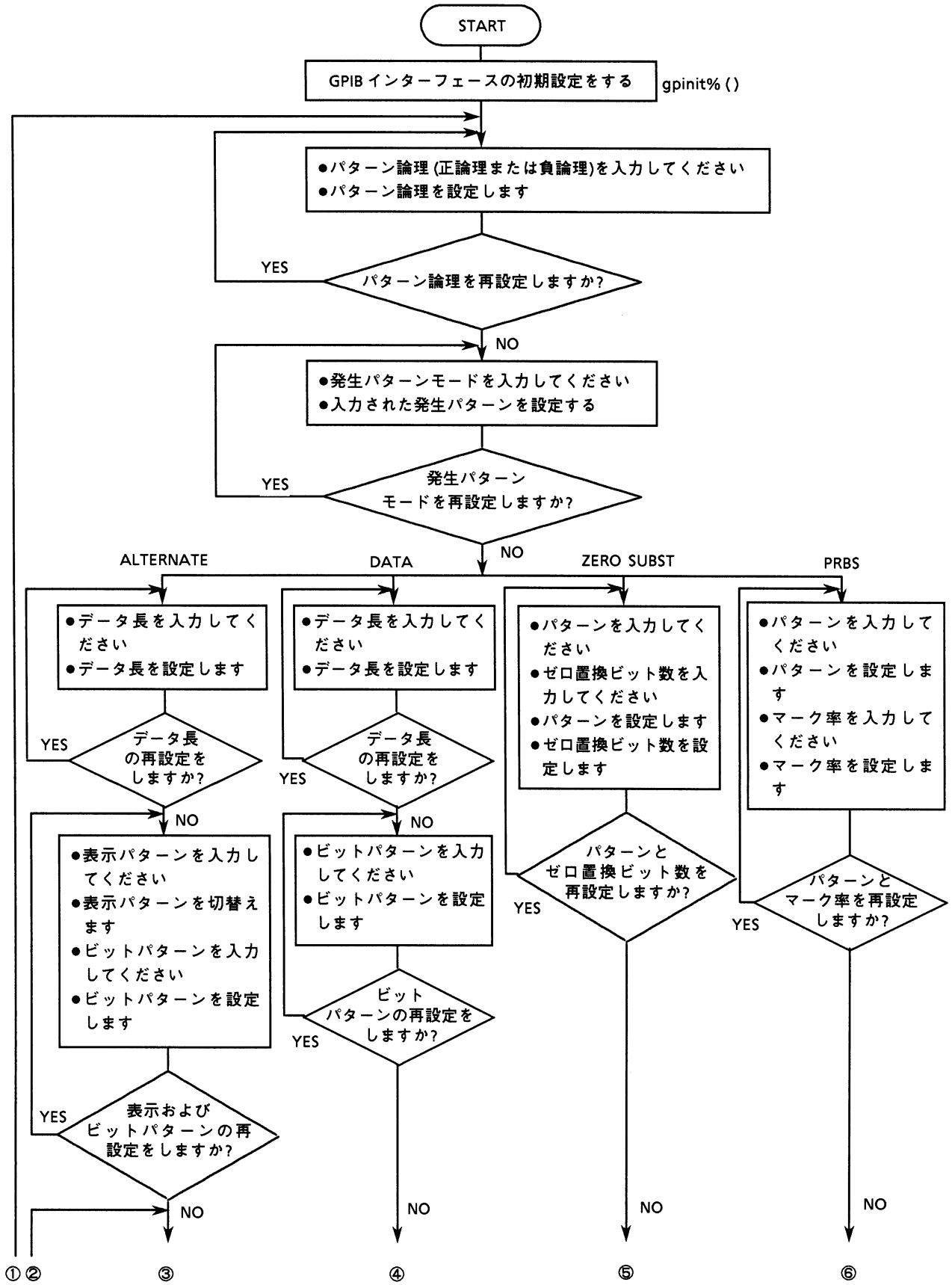


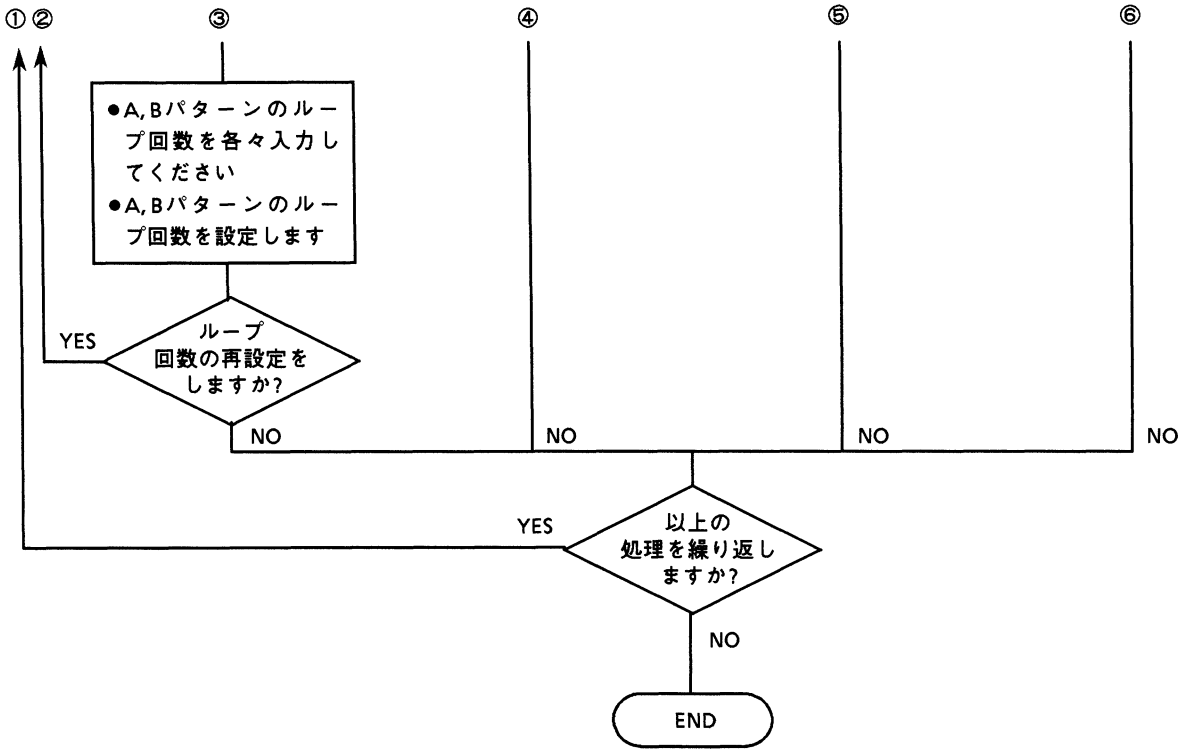
● プログラム・リスト

```
REM $INCLUDE: 'c:¥at-gpib¥qbasic¥qbdecl.bas'

COMMON SHARED DEV%, GPIB0%, PPG%

DECLARE SUB waidly (tim!)
DECLARE SUB wrtcmd1 (w$)
DECLARE FUNCTION gpinit% ()

'*********************************************************
'*     MP1761C / MP1763C  FREQUENCY SAMPLE SOFT          *
'*********************************************************
'
'---------------------------------------------------------
'                    MAIN ROUTINE
'---------------------------------------------------------
CLS
IF gpinit% <> 0 THEN      'setup interface
'
CALL wrtcmd1("RES 1")
'
   DO
            CLS
            LOCATE 3, 1
            GOSUB setfrq2
            GOSUB dset
            '
            PRINT ""
            INPUT " SET NEXT DATA [ Y or N ] ?", loop$
            '
   LOOP UNTIL loop$ = "N" OR loop$ = "n"
END IF
STOP
'
'---------------------------------------------------------
'                    SUB ROUTINE
'---------------------------------------------------------
'
setfrq2: '----------------------- INPUT DATA CONDITIONS
'
PRINT ""
PRINT " ****** MP1761C / MP1763C FREQUENCY_2 SAMPLE PROGRAM ******"
PRINT ""
'
INPUT " START FREQUENCY [ 50 to 12500 : MHz ]? ", startf$
PRINT ""
INPUT " STOP  FREQUENCY [ 50 to 12500 : MHz ]? ", stopf$
PRINT ""
INPUT " FREQUENCY STEP  [ MHz ]? ", stepf$
   '
RETURN
'
dset:    '----------------------------- OUTPUT DATA CONDITIONS
'
FOR I = VAL(startf$) TO VAL(stopf$) STEP VAL(stepf$)
           CALL wrtcmd1("FRQ " + STR$(I))          '  Set Frequency
           CALL waidly(.1)
NEXT I
RETURN
'
END
```

(3)　発生パターンの設定

本プログラムは、発生パターン関係の設定を行います。まず始めにパターン論理と発生パターンを選択し、選択したパターンに対して各種の設定を行います。




①②
③
④
⑤
⑥
●A,Bパターンのループ回数を各々入力してください
●A,Bパターンのループ回数を設定します
YES
ループ回数の再設定をしますか?
NO
NO
NO
NO
YES
以上の処理を繰り返しますか?
NO
END

● プログラム・リスト

```
DECLARE SUB ClearDisp (p%, l%)
REM $INCLUDE: 'c:¥at-gpib¥qbasic¥qbdecl.bas'

COMMON SHARED DEV%, GPIB0%, PPG%

DECLARE SUB waidly (tim!)
DECLARE SUB wrtcmdl (w$)
DECLARE FUNCTION gpinit% ()

'*********************************************************
'*                                                       *
'*     MP1761C / MP1763C PATTERN SET SAMPLE SOFT         *
'*                                                       *
'*********************************************************
'
'-------------------------------------------------------
'                    MAIN ROUTINE
'-------------------------------------------------------
'
IF gpinit% <> 0 THEN       'setup interface
'
   DO
            GOSUB pattern
            '
            PRINT ""
            INPUT " SET NEXT DATA [ YES = ENTER, NO = 1 ] ?", loop$
            '
   LOOP UNTIL loop$ = "1"
END IF
STOP
'
'-------------------------------------------------------
'                    SUB ROUTINE
'-------------------------------------------------------
'
pattern: '---------------------- INPUT DATA CONDITIONS
'
L = 4
CLS
PRINT ""
PRINT " ****** MP1761C / MP1763C PATTERN SET SAMPLE PROGRAM ******"
'
DO
   LOCATE L, 1
   INPUT " LOGIC MODE [ POSITIVE = 0, NEGATIVE = 1 ]?", lgc$
   wrtcmdl ("LGC " + lgc$)
   INPUT " Do you wish to change LOGIC (Y OR N) ?", yes$
LOOP UNTIL yes$ = "N" OR yes$ = "n"

CALL ClearDisp(5, 1)
L = L + 2

DO
   DO
            LOCATE L, 1
            INPUT " PATTERN MODE [ ALTN=0, DATA=1, Z.S.=2, PRBS=3 ] ?", pat$
   LOOP UNTIL pat$ = "0" OR pat$ = "1" OR pat$ = "2" OR pat$ = "3"
   wrtcmdl ("PTS " + pat$)
   '
   INPUT " Do you wish to change PATTERN (Y OR N) ?", yes$
LOOP UNTIL yes$ = "N" OR yes$ = "n"
```

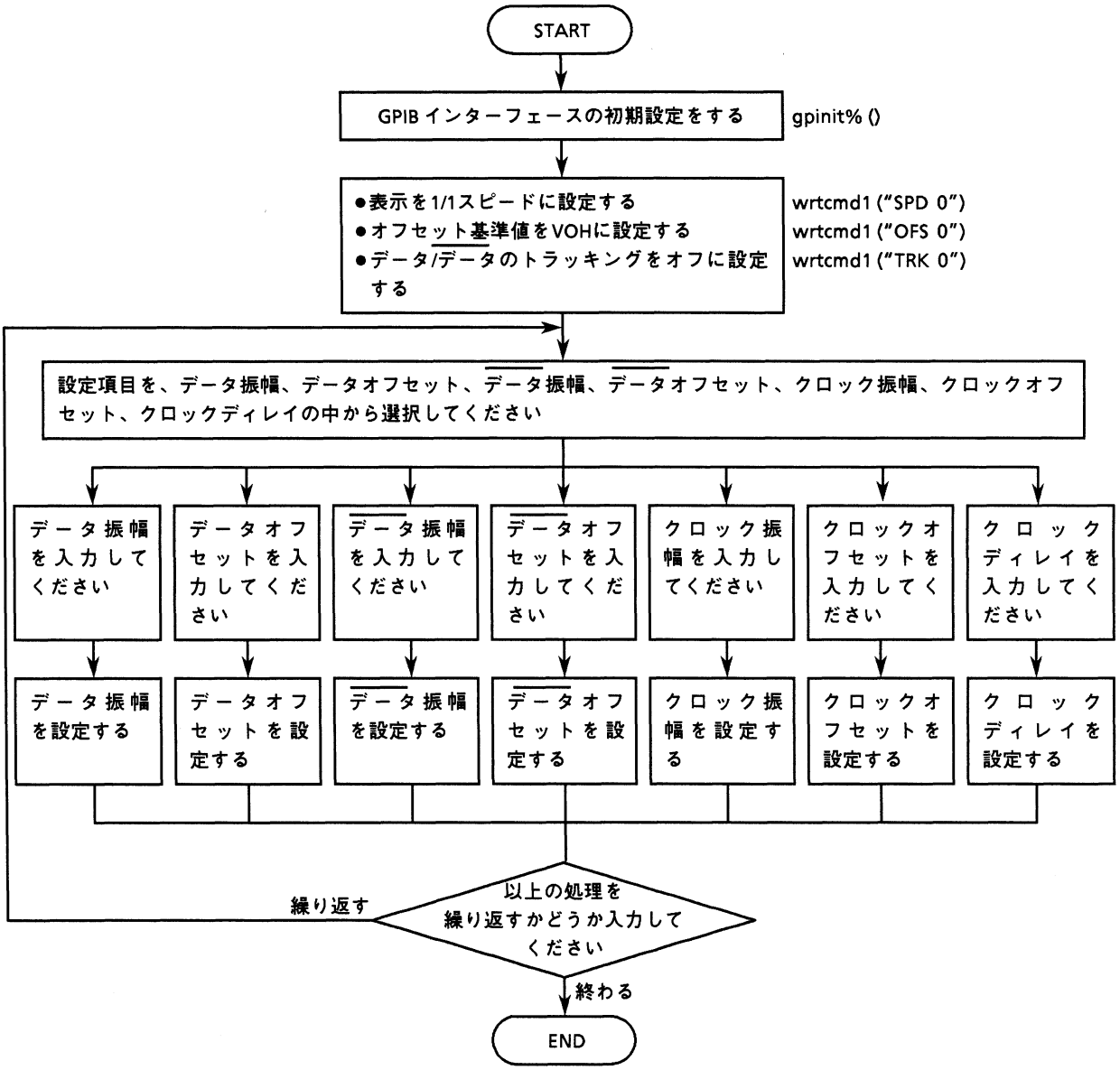
```
CALL ClearDisp(7, 1)
L = L + 2

IF pat$ = "3" OR pat$ = "2" THEN                    '   PRBS OR Z.S. PATTERN
   DO
          LOCATE L, 1
          IF pat$ = "3" THEN
                  PRINT " PRBS MODE [PN7 =2, PN9 =3, PN11=5, PN15=6,"
                  INPUT "            PN20=7, PN23=8, PN31=9         ]?",ptn$
                  wrtcmd1 ("PTN " + ptn$)
                  '
                  PRINT " MARK RATIO (POSITIVE) [0/8=0,1/8=1,1/4=2,1/2=3]"
                  INPUT "            (NEGATIVE) [8/8=0,7/8=1,3/4=2,1/2=3]?",
mrk$
                  wrtcmd1 ("MRK " + mrk$)
                  '
                  INPUT " Do you wish to change PRBS MODE & MARK RATIO (Y OR
                  N) ?", y$
          END IF
          IF pat$ = "2" THEN
                  INPUT " Z.S. MODE [2^7=2, 2^9=3, 2^11=5, 2^15=6]?", ptn$
                  INPUT " ZERO SUBSTITUTION LENGTH [ 1 to 32767 ] ?", zln$
                  wrtcmd1 ("PTN " + ptn$)
                  wrtcmd1 ("ZLN " + zln$)
                  '
                  INPUT " Do you wish to change Z.S MODE & Z.S. LENGTH (Y OR
                  N) ?", y$
          END IF
   LOOP UNTIL y$ = "N" OR y$ = "n"
ELSE                                                '   DATA OR ALTN PATTERN
   IF pat$ = "0" THEN GOSUB setaltn
   IF pat$ = "1" THEN GOSUB setdata
END IF
RETURN
'
setaltn: '-------------------------- SET ALTERNATE A/B PATTERN ---------
'=========== SET DATA LENGTH ==========================
DO
   DO
          LOCATE L, 1
          INPUT " DATA LENGTH [ 128 to 4194304 ] ?", dln
   LOOP UNTIL dln >= 128 AND dln <= 4194304
   wrtcmd1 ("DLN " + STR$(dln))
   '
   INPUT " Do you wish to change DATA LENGTH (Y OR N) ?", yes$
LOOP UNTIL yes$ = "n" OR yes$ = "N"

CALL ClearDisp(9, 1)
L = L + 1
'
'============ SET ALTN PATTERN ================================
DO
   LOCATE L + 2, 1
   FOR c1 = 1 TO 6: PRINT SPACE$(78): NEXT c1
   LOCATE L + 1, 1
   INPUT " CHOSE ALTERNATE [ A = 0, B = 1 ] ?", alt$
   wrtcmd1 ("ALT " + alt$)
   GOSUB setbit
   '
   INPUT " Do you wish to change A/B pattern & bit pattern (y or n) ?", yes$
LOOP UNTIL yes$ = "n" OR yes$ = "N"
L = L + 2
'
```

```
DO
   LOCATE L + 1, 1: FOR cl = 1 TO 2
   PRINT SPACE$(78)
   NEXT cl
'
   LOCATE L, 1
   INPUT " A PATTERN LOOP TIME [ 1 to 127 ] ?", lpt$
   wrtcmd1 ("ALT 0")
   wrtcmd1 ("LPT " + lpt$)
   INPUT " B PATTERN LOOP TIME [ 1 to 127 ] ?", lpt$
   wrtcmd1 ("ALT 1" + ";LPT " + lpt$)
   '
   INPUT " Do you wish to change LOOP TIME (Y OR N) ?", yes$
LOOP UNTIL yes$ = "n" OR yes$ = "N"
L = L + 3
'
RETURN
'
setdata: '--------------------------- SET DATA PATTERN ------------------
DO
   DO
           LOCATE L, 1
           INPUT " DATA LENGTH [ 2 to 8388608 ] ?", dln
   LOOP UNTIL dln >= 2 AND dln <= 8388608
   wrtcmd1 ("DLN " + STR$(dln))
   '
   INPUT " Do you wish to change DATA LENGTH (Y OR N) ?", yes$
LOOP UNTIL yes$ = "n" OR yes$ = "N"
LOCATE L + 1, 1: PRINT SPACE$(78)
L = L + 2
'=========== SET DATA PATTEN ============================
DO
   LOCATE L + 1, 1: PRINT SPACE$(78)
   LOCATE L + 1, 1
   GOSUB setbit
   '
   INPUT " Do you wish to change BIT PATTERN (Y OR N) ?", yes$
   LOCATE L + 2, 1: FOR cl = 1 TO 5
   PRINT SPACE$(78)
   NEXT cl
LOOP UNTIL yes$ = "N" OR yes$ = "n"
'
RETURN
'
```

```
setbit: '------------------------------ SET BIT PATTERN --------------------
INPUT " TOP PAGE  TO SET ?:", page
'
j = 8
PRINT " You are able to choice data format of Hexadecimal or Decimal."
PRINT " Default data format is Hexadecimal."
'
BIT$ = ""
FOR k = 0 TO j - 1
   LOCATE L + 4, 1
   PRINT " Do you set bit-pattern of" + STR$(page + k) + " PAGE ? ";
   INPUT "(y or n)", yes$
   IF yes$ = "n" OR yse$ = "N" THEN
            EXIT FOR
   END IF
   '
   IF k <> 0 THEN BIT$ = BIT$ + ","
   '
   INPUT " Which do you choice format? (Hexadecimal / Decimal)", ftype$
   LOCATE L + 2, 1:
   FOR cl = 1 TO 4
            PRINT SPACE$(78)
   NEXT cl
   LOCATE L + 2, 1
   PRINT SPACE$(78)
   IF ftype$ = "d" OR ftype$ = "D" THEN
            PRINT " Enter " + STR$(page + k) + " PAGE pattern [0 to 65535]";
            INPUT b$
   ELSE
            PRINT " Enter " + STR$(page + k) + " PAGE pattern [0 to FFFF]";
            INPUT b$
            b$ = "&H" + b$
   END IF
   BIT$ = BIT$ + b$
   '
NEXT k
PRINT ""
'
LOCATE L + 1, 1
FOR cl = 1 TO 5: PRINT SPACE$(78): NEXT cl
LOCATE L + 2, 1
PRINT " PAG " + STR$(page) + ";BIT " + BIT$
wrtcmd1 ("PAG " + STR$(page) + ";BIT " + BIT$)
RETURN

END
```

(4)　出力信号の設定

本プログラムはクロック信号のディレイ、データ/$\overline{\text{データ}}$/クロック信号の振幅およびオフセットを設定します。



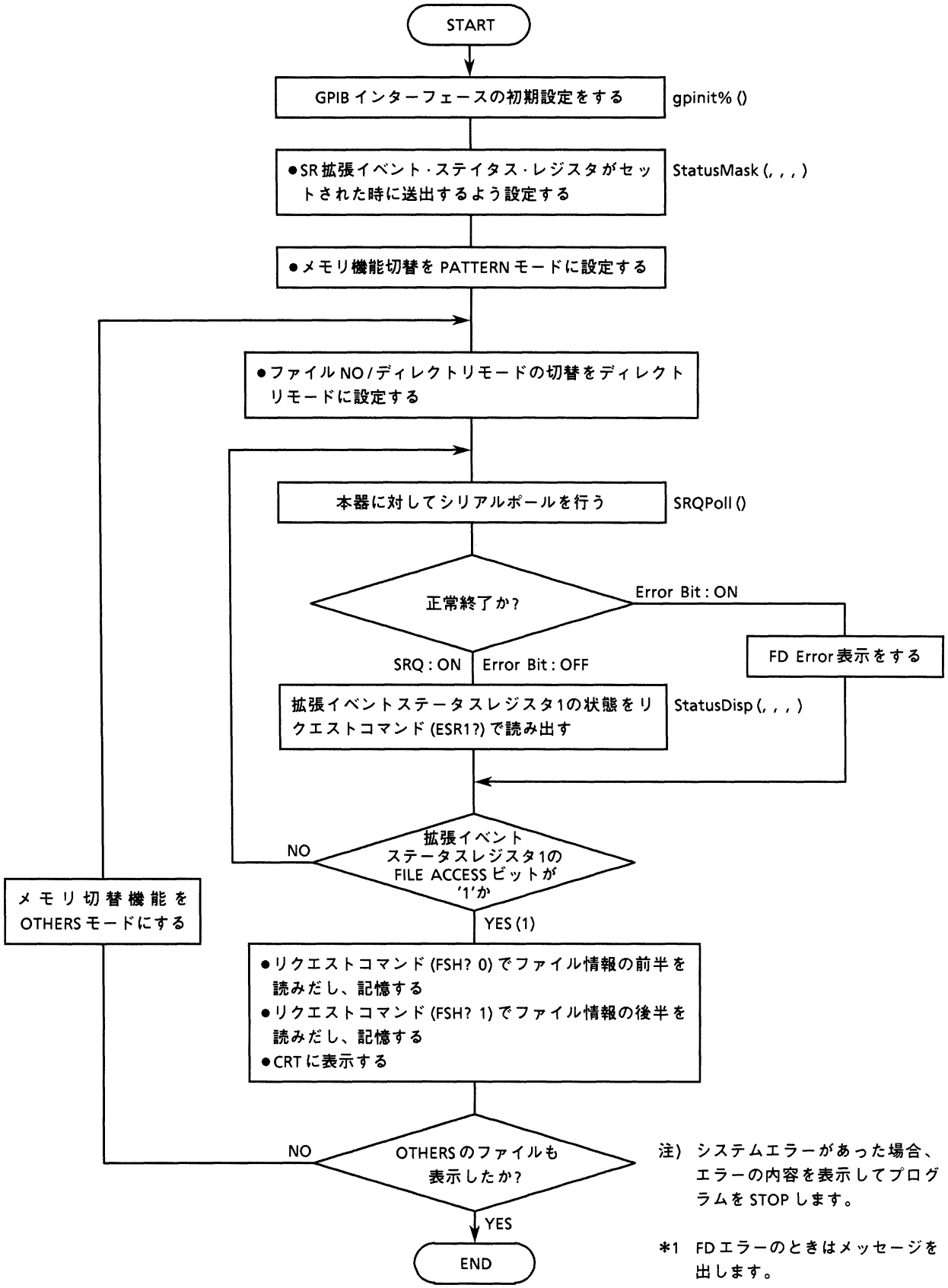
● プログラム・リスト

```
REM $INCLUDE: 'c:¥at-gpib¥qbasic¥qbdecl.bas'

COMMON SHARED DEV%, GPIB0%, PPG%

DECLARE SUB waidly (tim!)
DECLARE SUB wrtcmdl (w$)
DECLARE FUNCTION gpinit% ()

'*********************************************************
'*                                                       *
'*     MP1761C / MP1763C  OUTPUT  SAMPLE SOFT            *
'*    ----- 1/1 SPEED, VOH, TRACKING OFF ------          *
'*                                                       *
'*********************************************************
'
'-------------------------------------------------------
'                    MAIN ROUTINE
'-------------------------------------------------------
'
CLS
IF gpinit% <> 0 THEN       'setup interface
'
CALL wrtcmdl("SPD 0")      ' 1/1 SPEED
CALL wrtcmdl("OFS 0")      ' OFFSET VOH
CALL wrtcmdl("TRK 0")      ' DATA / NDATA TRACKING OFF
'
   DO
            CLS
            LOCATE 3, 1
            GOSUB setout
            GOSUB dset
            '
            PRINT ""
            INPUT " SET NEXT DATA [ Y or N ] ?", loop$
            '
   LOOP UNTIL loop$ = "N" OR loop$ = "n"
END IF
STOP
```

```
'
'-------------------------------------------------------
'                    SUB ROUTINE
'-------------------------------------------------------
'
setout: '----------------------- OUTPUT SIGNAL CONDITIONS
'
PRINT ""
PRINT " ****** MP1761C / MP1763C OUTPUT SIGNAL SET SAMPLE PROGRAM ******"
PRINT ""
'
PRINT " SETTING ITEM [ DATA  AMPLITUDE = 1, DATA  OFFSET = 2"
PRINT "                NDATA AMPLITUDE = 3, NDATA OFFSET = 4"
PRINT "                CLOCK AMPLITUDE = 5, CLOCK OFFSET = 6"
INPUT "                CLOCK DELAY = 7 ] ? ", ITM
PRINT ""
'
RETURN
'
'
dset:    '----------------------------- SET OUTPUT CONDITIONS
'
IF ITM = 1 OR ITM = 2 THEN CALL wrtcmd1("DDS 0")
IF ITM = 3 OR ITM = 4 THEN CALL wrtcmd1("DDS 1")
'
IF ITM = 1 THEN
           INPUT " INPUT DATA AMPLITUDE [ 2.000 to 0.25 ] ", DAP$
           CALL wrtcmd1("DAP " + DAP$)
END IF
IF ITM = 2 THEN
           INPUT " INPUT DATA OFFSET [ 2.000 to -2.000 ] ", DOS$
           CALL wrtcmd1("DOS " + DOS$)
END IF
IF ITM = 3 THEN
           INPUT " INPUT NDATA AMPLITUDE [ 2.000 to 0.25 ] ", NAP$
           CALL wrtcmd1("NAP " + NAP$)
END IF
IF ITM = 4 THEN
           INPUT " INPUT NDATA OFFSET [ 2.000 to -2.000 ] ", NOS$
           CALL wrtcmd1("NOS " + NOS$)
END IF
IF ITM = 5 THEN
           INPUT " INPUT CLOCK AMPLITUDE [ 2.000 to 0.25 ] ", CAP$
           CALL wrtcmd1("CAP " + CAP$)
END IF
IF ITM = 6 THEN
           INPUT " INPUT CLOCK OFFSET [ 2.000 to -2.000 ] ", COS$
           CALL wrtcmd1("COS " + COS$)
END IF
IF ITM = 7 THEN
           INPUT " INPUT CLOCK DELAY [ 500 to -500 ] ", CDL$
           CALL wrtcmd1("CDL " + CDL$)
END IF
           '
RETURN
'
END
```

(5)　フロッピーディスクのファイル情報読みだし

本プログラムはフロッピーディスクに格納されているファイルのディレクトリ情報をCRTに表示します。



注）システムエラーがあった場合、エラーの内容を表示してプログラムをSTOPします。

＊1　FDエラーのときはメッセージを出します。

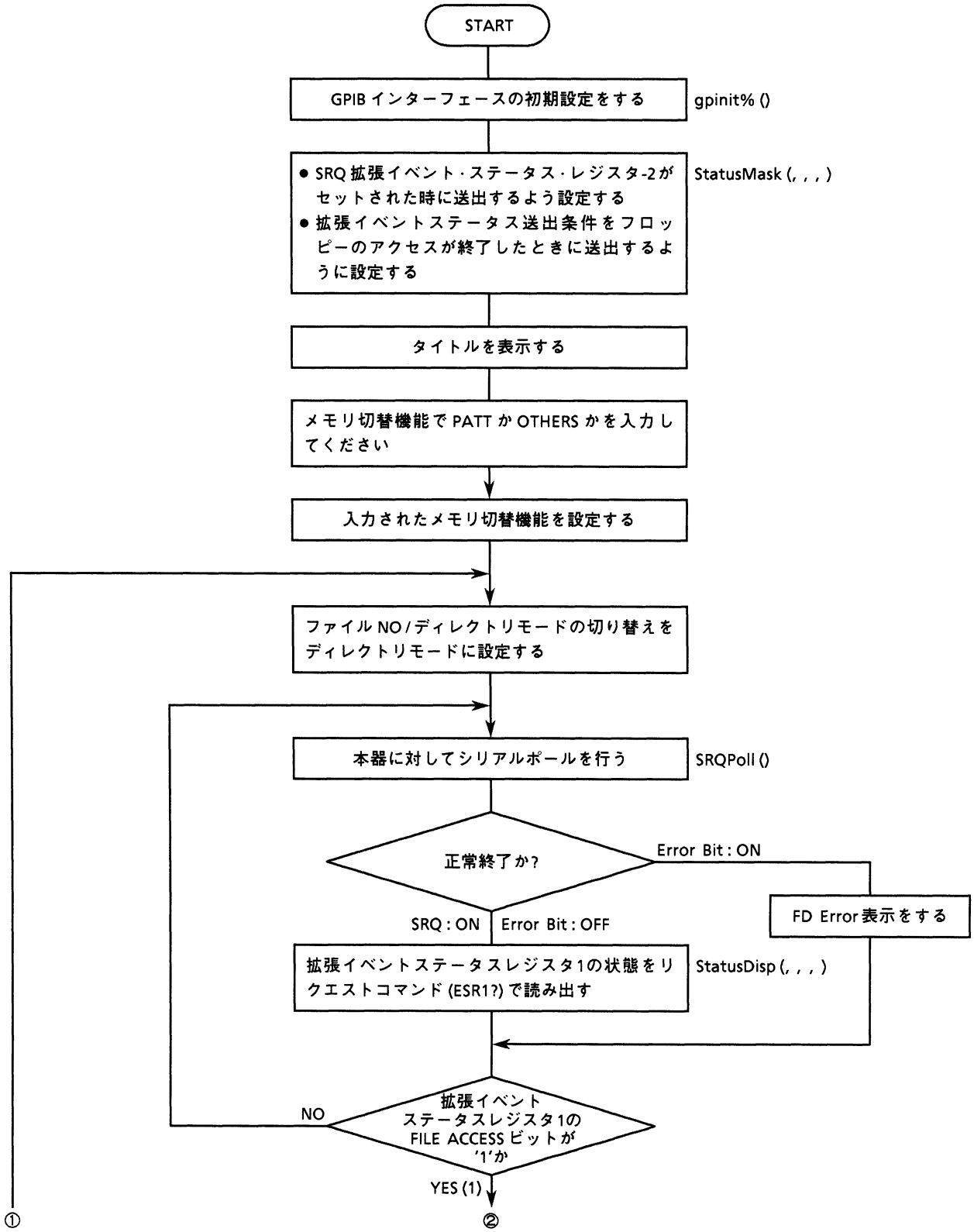
● プログラム・リスト

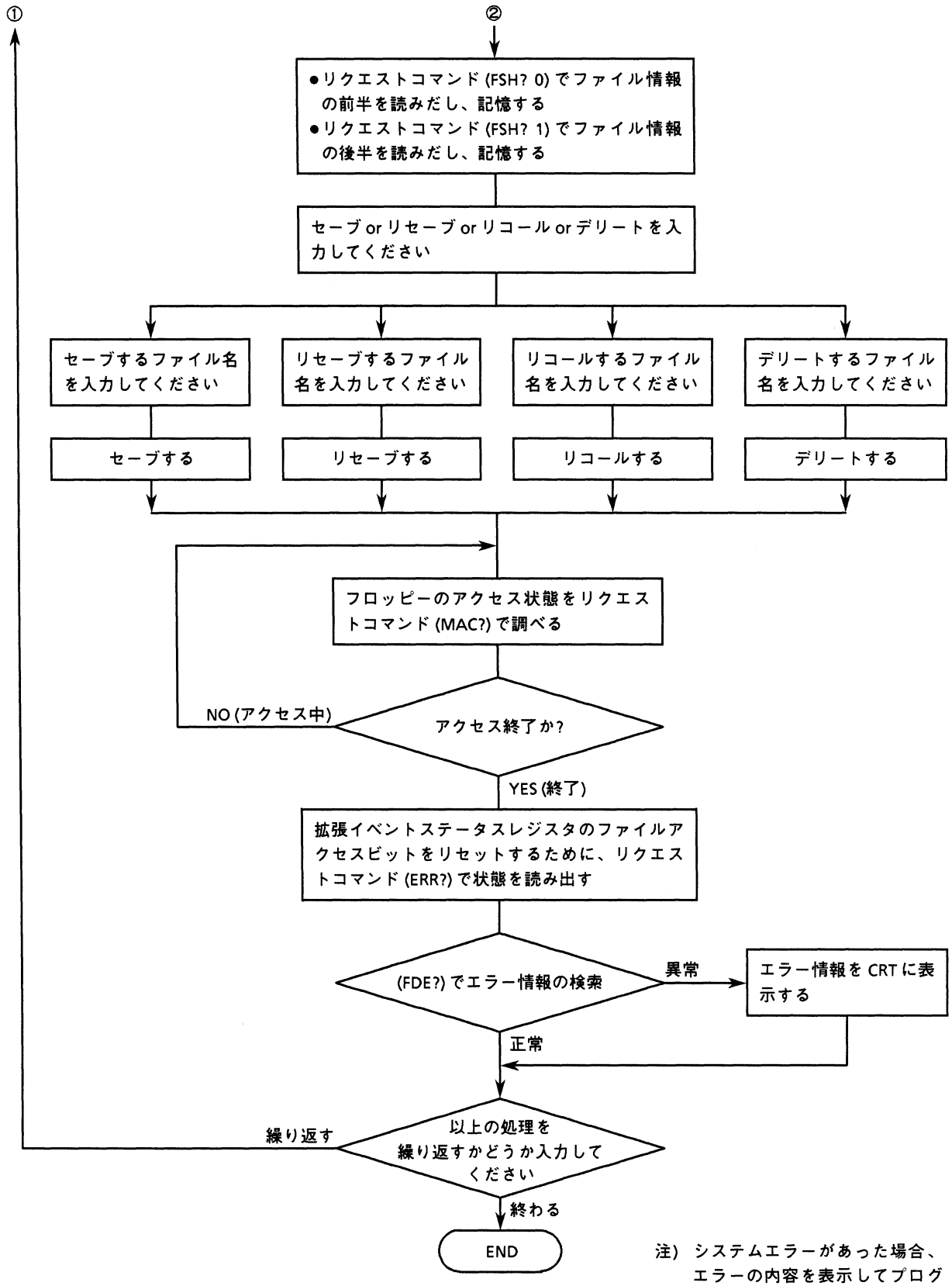
```
REM $INCLUDE: 'c:¥at-gpib¥qbasic¥qbdecl.bas'

COMMON SHARED DEV%, GPIB0%, PPG%

DECLARE SUB wrtcmdl (w$)
DECLARE SUB ErrPoll ()
DECLARE SUB StatusDisp (stb%, esr%, esr2%, esr3%)
DECLARE SUB StatusMask (s0%, s1%, s2%, s3%)
DECLARE FUNCTION itob$ (l%, V%)
DECLARE FUNCTION SRQPoll% ()
DECLARE FUNCTION gpinit% ()
DECLARE FUNCTION readcmdl$ ()

IF gpinit% <> 0 THEN                              'Setup interface
         '====== Set MSS status byte register =======
         CALL StatusMask(&H4, &H0, &H2, &H2)

         FOR i = 0 TO 1
                  '======== Set memory mode Pattern/Others =========
                  wrtcmdl ("MEM " + STR$(i))

                  '========== Set FILE DIR mode ===========
                  wrtcmdl ("FIL 1")

                  '========= Polling FILE ACCESS bit =========
                  DO
                           IF SRQPoll% <> 0 THEN
                                    CALL StatusDisp(dmy%, dmy1%, reg%, dmy3%)
                           ELSE
                                    LOCATE 12, 35
                                    PRINT "FD error detect!!"
                                    EXIT DO
                           END IF
                   LOOP UNTIL reg% AND &H2
```

```
                    '=========== Read FD infomation ============
                     wrtcmd1 ("FDE?")
                     rd1$ = LEFT$(readcmd1$, IBCNT% - 1)
                     IF rd1$ <> "FDE 10" THEN
                             LOCATE 1, 1
                             SELECT CASE VAL(MID$(rd1$, 5, 2))
                                     CASE 0
                                          PRINT "<<E0:Media error           >>"
                                     CASE 1
                                          PRINT "<<E1:Write protection error>>"
                                     CASE 2
                                          PRINT "<<E2:File full             >>"
                                     CASE 3
                                          PRINT "<<E3:File not found        >>"
                                     CASE 4
                                          PRINT "<<E4:File already exists
                                          error>>"
                                     CASE 5
                                          PRINT "<<E5:Write error           >>"
                                     CASE 6
                                          PRINT "<<E6:Read error            >>"
                                     CASE 7
                                          PRINT "<<E7:File type,File error  >>"
                                     CASE 8
                                          PRINT "<<E8:FD error              >>"
                                     CASE 9
                                          PRINT "<<E9:Hardware error        >>"
                             END SELECT
                             LOCATE 23, 1
                             INPUT "End of FD analize.Press 'Enter' to fin.";f$
                             STOP
                    END IF
                    wrtcmd1 ("FSH? 0")
                    rd1$ = LEFT$(readcmd1$, IBCNT% - 1)
                    wrtcmd1 ("FSH? 1")
                    rd2$ = LEFT$(readcmd1$, IBCNT% - 1)
                    '=============  Output CRT  ================
                    IF i = 0 THEN
                             LOCATE 4, 1
                             PRINT "Pattern directory data."
                    ELSE
                             LOCATE 11, 1
                             PRINT "Others directory data."
                    END IF
                    PRINT "Unused size:"+MID$(rd1$,5,7)      'print unused size
                    PRINT "Used size  :"+MID$(rd1$,13,7)      'print used size
                    PRINT "File count :";VAL(MID$(rd1$,21,22))'print file num

                    'print file no
                    PRINT "File name  :";MID$(rd1$,24)+"," + MID$(rd2$,24)
            NEXT i
            '
            PRINT
            INPUT "End of FD analize. Press 'Enter' to fin."; f$
END IF

STOP
```

(6)　FDのオペレーション(データのセーブ、リセーブ、リコール)

本プログラムはフロッピーディスクへのデータセーブ、リセーブ、またフロッピーディスクのデータリコールを行います。





注) システムエラーがあった場合、エラーの内容を表示してプログラムを STOP します。

*1 FD エラーの場合はメッセージを出します。

● プログラム・リスト

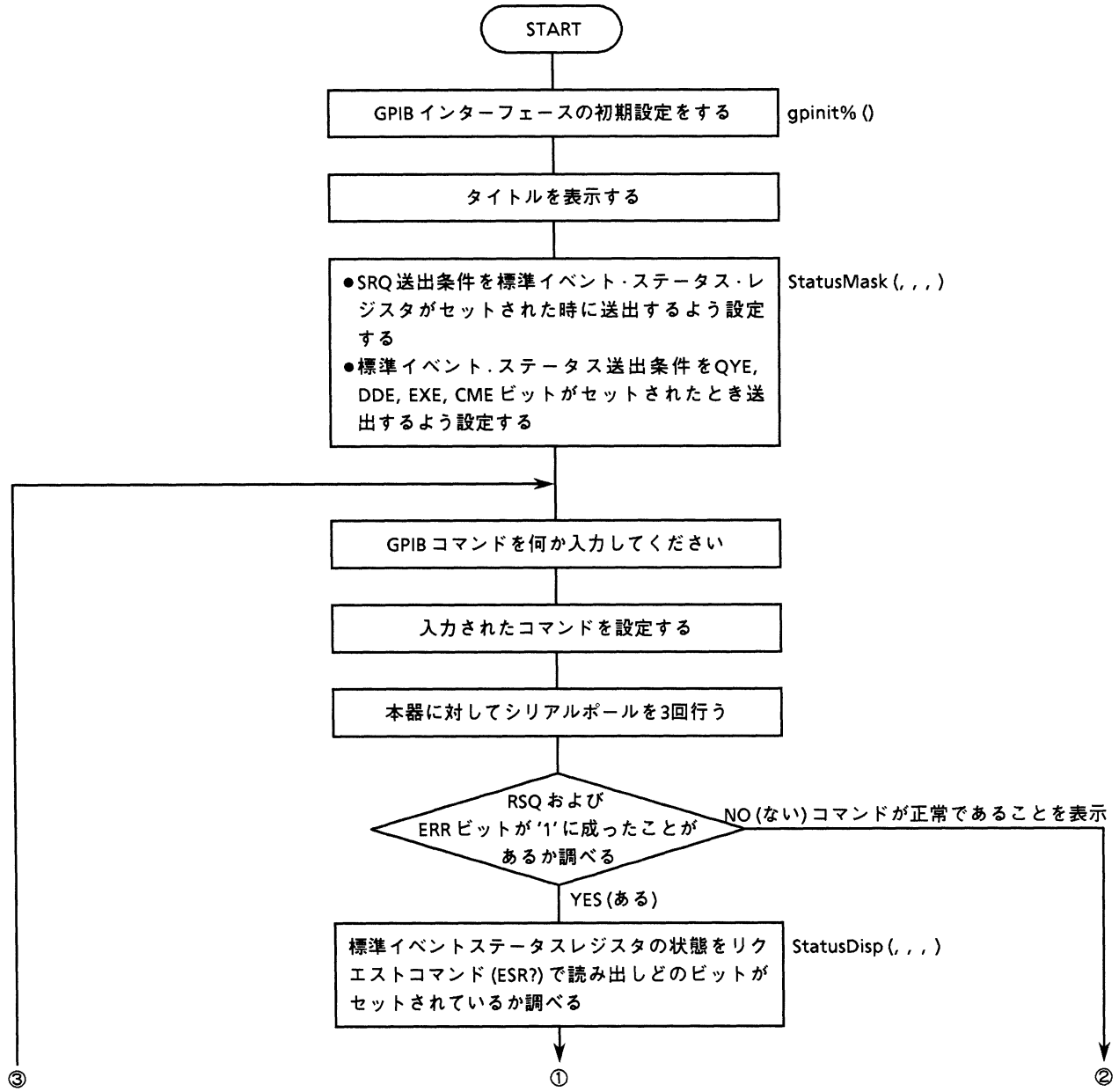
```
DECLARE SUB ClearDisp (p%, l%)
REM $INCLUDE: 'c:¥at-gpib¥qbasic¥qbdecl.bas'

COMMON SHARED DEV%, GPIB0%, PPG%, ED%

DECLARE SUB waidly (tim!)
DECLARE SUB wrtcmdl (w$)
DECLARE SUB ErrPoll ()
DECLARE SUB StatusDisp (stb%, esr%, esr2%, esr3%)
DECLARE SUB StatusMask (s0%, s1%, s2%, s3%)
DECLARE FUNCTION itob$ (l%, V%)
DECLARE FUNCTION SRQPoll% ()
DECLARE FUNCTION gpinit% ()
DECLARE FUNCTION readcmdl$ ()

IF gpinit% <> 0 THEN                          'Setup interface

        '====== Set MSS status byte register =======
        CALL StatusMask(&HC, &H0, &H2, &H2)

        DO
                CLS
                PRINT "** MP1761C/MP1763C FD OPERATION PROGRAM ** "
                '
                '============  Select PTN/OTHERS  ==============
                DO
                        LOCATE 17, 1
                        INPUT"Memory mode select[PATTERN:0,OTHERS:1]";mem$
                        IF mem$ <> "0" AND mem$ <> "1" THEN
                                LOCATE 16, 1
                                PRINT "Wrong chosen number!! Please
                                select a correct number"
                        END IF
                        CALL ClearDisp(16, 2)
                LOOP UNTIL mem$ = "0" OR mem$ = "1"
                wrtcmdl ("MEM " + mem$)

                '========== Set FILE DIR mode ===========
                wrtcmdl ("FIL 1")

                '========= Polling FILE ACCESS bit =========
                DO
                        IF SRQPoll% <> 0 THEN
                                CALL StatusDisp(dmy%, dmy1%, reg%, dmy3%)
                        ELSE
                                GOSUB Fderr
                                GOTO jump
                        END IF
                LOOP UNTIL reg% AND &H2

                '==== Save, Resave, Recall or Delete? =====
                DO
                        LOCATE 17, 1
                        INPUT "Choose function
                        [SAVE:0,RESAVE:1,RECALL:2,DELEAT:3]";op%
                        IF op% < 0 OR op% > 3 THEN
                                LOCATE 16, 1
                                PRINT "Wrong chosen number!! Please
                                select correct function."
                        END IF
                LOOP UNTIL op% >= 0 AND op% <= 3
```

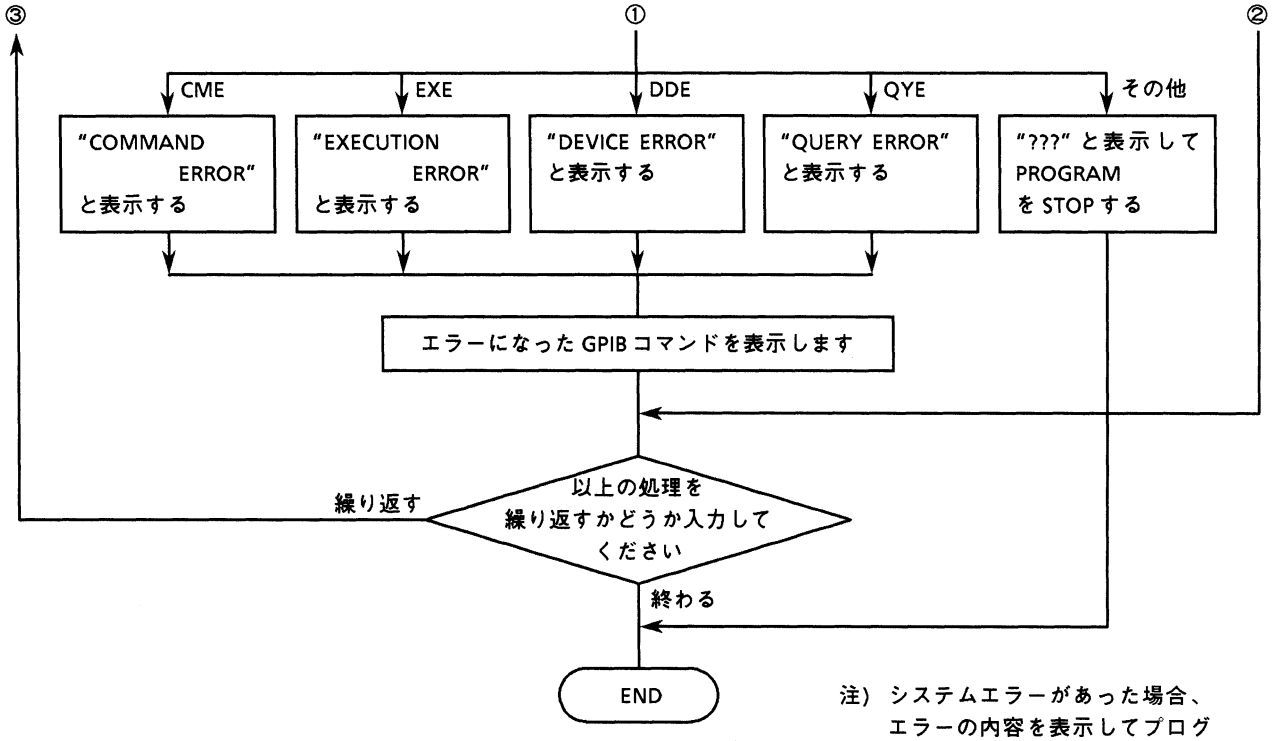
```
                    CALL ClearDisp(16, 2)
                    LOCATE 17, 1
                    SELECT CASE op%
                    CASE 0
                              INPUT "Enter file number for SAVE:"; NO$
                              wrtcmd1 ("SAV " + NO$)
                              '
                    CASE 1
                              INPUT "Enter file number for RESAVE:"; NO$
                              wrtcmd1 ("RSV " + NO$)
                              '
                    CASE 2
                              INPUT "Enter file number for RECALL:"; NO$
                              wrtcmd1 ("RCL " + NO$)
                              '
                    CASE 3
                              INPUT "Enter file number for DELEAT:"; NO$
                              wrtcmd1 ("DEL " + NO$)
                              '
                    END SELECT
                    GOSUB Faccess
                    GOSUB Fderr
                    CALL ClearDisp(17, 1)

                    '======= Reset EventStatusRegister1 ========
jump:    CALL StatusDisp(dmy%, dmy1%, dmy2%, dmy3%)
                    '
                    LOCATE 17, 1
                    INPUT "Do you more test another function? [Yes/No]"; loop$
          LOOP UNTIL loop$ = "n" OR loop$ = "N"
END IF
'
STOP
'
Faccess: '============ FD access end ?  =============
                    '
                    DO
                              CALL StatusDisp(dmy%, dmy1%, dmy2%, dmy3%)
                              waidly (1)
                              wrtcmd1 ("MAC?")
                              RD$ = LEFT$(readcmd1$, IBCNT% - 1)
                    LOOP UNTIL MID$(RD$, 1, 5) = "MAC 0"
                    '
RETURN
'
```

```
Fderr: '=========== FD error message ===============
            '
            CALL wrtcmdl("FDE?")
            RD$ = LEFT$(readcmdl$, IBCNT% - 1)
            LOCATE 10, 1
            IF RD$ <> "FDE 10" THEN
                     PRINT "FD error occuerd!! "
                     SELECT CASE MID$(RD$, 6, 1)
                              CASE "0"
                                       PRINT "E0:Media error                "
                              CASE "1"
                                       PRINT "E1:Write protection error     "
                              CASE "2"
                                       PRINT "E2:File full                  "
                              CASE "3"
                                       PRINT "E3:File not found             "
                              CASE "4"
                                       PRINT "E4:File already exists error  "
                              CASE "5"
                                       PRINT "E5:Write error                "
                              CASE "6"
                                       PRINT "E6:Read error                 "
                              CASE "7"
                                       PRINT "E7:File type , File error     "
                              CASE "8"
                                       PRINT "E8:FD error                   "
                              CASE "9"
                                       PRINT "E9:Hardware error             "
                     END SELECT
            ELSE
                     PRINT "<<  ** FD operation complete!! **    >>"
                     PRINT "<<  ** Accept file number is " + NO$ + ". **  >>"
            END IF
            '
RETURN
```

(7)　標準ステータスバイト(4種類)のチェック

本プログラムは標準ステータスバイト(QYE, DDE, EXE, CMEビット)のチェックをします。





注）システムエラーがあった場合、エラーの内容を表示してプログラムを STOP します。

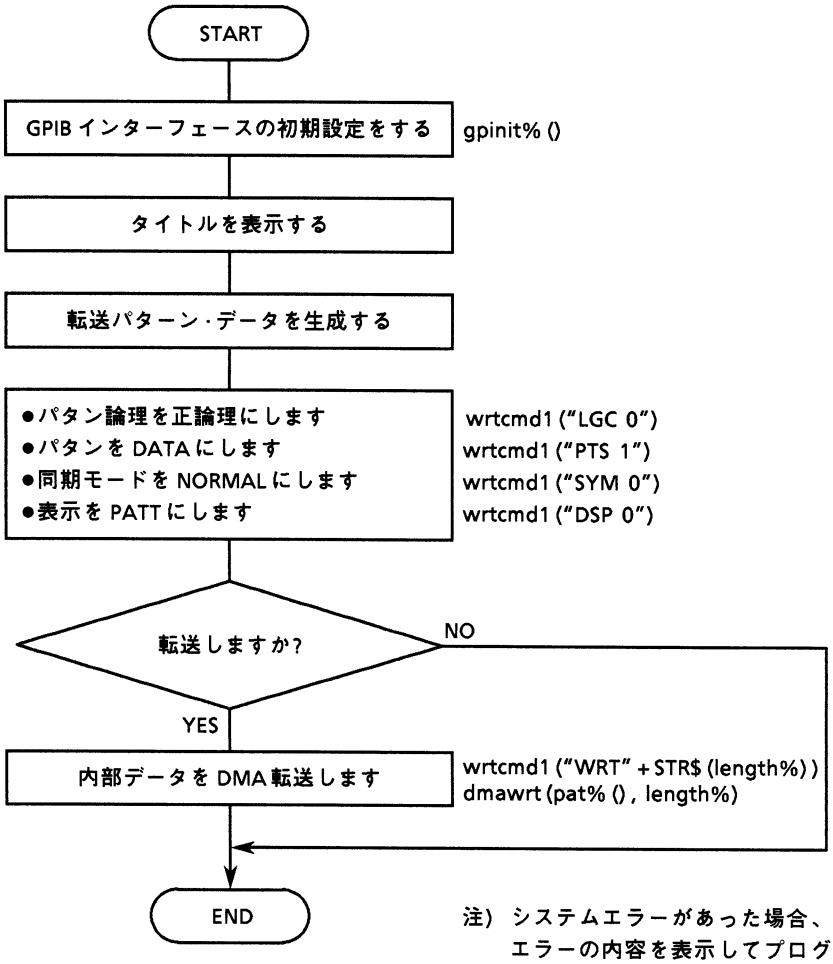
● プログラム・リスト

```
REM $INCLUDE: 'c:¥at-gpib¥qbasic¥qbdecl.bas'

COMMON SHARED DEV%, GPIB0%, PPG%

DECLARE SUB waidly (tim!)
DECLARE SUB wrtcmdl (WRT$)
DECLARE SUB trap ()
DECLARE SUB ClearDisp (p%, l%)
DECLARE SUB StatusDisp (stb%, esr%, esr2%, esr3%)
DECLARE SUB StatusMask (s0%, s1%, s2%, s3%)
DECLARE FUNCTION itob$ (l%, V%)
DECLARE FUNCTION gpinit% ()
DECLARE FUNCTION readcmdl$ ()

CLS
IF gpinit% <> 0 THEN                              'Setup interface

        PRINT "** MP1761C/MP1763C STANDARD STATUS REGISTER CHECK **"
        PRINT

        '====== Set MSS status byte register =======
        CALL StatusMask(&H3C, &H7E, &H77F, &H3)

        DO
                CALL ClearDisp(5, 15)
                LOCATE 5, 1
                INPUT "Please enter some GPIB command(s):"; com$
                length% = LEN(com$)
                CALL wrtcmdl(com$)
                LOCATE 5, 1
                PRINT "Please enter some GPIB command(s):
                      "

                sta% = 0
                FOR i = 0 TO 2
                        CALL IBRSP(PPG%, SPR%)
                        IF IBSTA < 0 THEN CALL trap
                        sta% = sta% OR SPR%
                        sta$ = itob$(8, SPR%)
                        LOCATE 1, 60
                        PRINT "*SRE:"; sta$

                        waidly (.1)
                NEXT i
```

```
                IF (sta% AND &H20) THEN
                        LOCATE 7, 1
                        PRINT "Execution command(s) fail of ''
                                        "
                        IF length% > 0 THEN
                                LOCATE 7, 1
                                PRINT "Execution command(s) fail of '";
                                LEFT$(com$, length%); "'"
                        END IF

                        CALL StatusDisp(dmy%, reg%, dmy2%, dmy3%)

                        CALL ClearDisp(8, 6): LOCATE 8, 1
                        IF (reg% AND &H2) OR (reg% AND &H40) THEN PRINT
                        " ??? ": STOP
                        IF reg% AND &H4 THEN PRINT "* QUERY ERROR *"
                        IF reg% AND &H8 THEN PRINT "* DEVICE ERROR *"
                        IF reg% AND &H10 THEN PRINT "* EXECUTION ERROR *"
                        IF reg% AND &H20 THEN PRINT "* COMMAND ERROR *"
                ELSE
                        LOCATE 7, 1
                        PRINT "Command succed execution.                "
                        CALL ClearDisp(9, 6)
                        CALL ClearDisp(2, 3)
                END IF

                LOCATE 15, 36: PRINT "                                  "
                LOCATE 15, 1
                INPUT "Do you test other command? [Yes/No] "; loop$
        LOOP UNTIL loop$ = "n" OR loop$ = "N"
END IF
'
STOP
```

(8)　パターンデータの DMA 転送処理

本プログラムはパターンデータの DMA 転送処理を行います



注）システムエラーがあった場合、エラーの内容を表示してプログラムを STOP します。

● プログラム・リスト

```
REM $INCLUDE: 'c:¥at-gpib¥qbasic¥qbdecl.bas'

COMMON SHARED DEV%, gpib0%, PPG%

DECLARE SUB StatusMask (s0%, s1%, s2%, s3%)
DECLARE SUB StatusDisp (stb%, esr%, esr2%, esr3%)
DECLARE SUB wrtcmd1 (w$)
DECLARE SUB dmawrt (w%(), i%)
DECLARE SUB gpiberr (msg$)
DECLARE FUNCTION gpinit% ()
DECLARE FUNCTION Exchange% (i%)

DIM pat%(302)

IF gpinit% <> 0 THEN            'Setup interface
        CLS
        PRINT "** MP1761C/MP1763C DMA(pattern data) SAMPLE PROGRAM ** "
        PRINT

        CALL StatusMask(&H0, &H0, &H0, &H0)
        '
        '===========   Table   ============
        ' Test pattern set and swap data.
        ' if you use ibconfig() swap function,
        ' then don't call Exchange() function. Because, its same operation.
        '
        Dlength% = 300                              ':Max Page
        j% = 0
        FOR i% = 0 TO Dlength% - 1
                pat%(i%) = Exchange(j%)
                j% = j% + 1
        NEXT i%
        pat%(i%) = &HA

        '=============  initial  ===============
        CALL wrtcmd1("LGC 0")       'Pattern logic        : positive
        CALL wrtcmd1("PTS 1")       'Pattern              : data
        CALL wrtcmd1("DLN " + STR$(Dlength% * 16)): DATA Length

        CALL StatusDisp(dmy%, dmy1%, dmy2%, dmy3%)

        INPUT "Do you wish transmit PATTERN data? [Yes/No]:"; a$
        IF a$ = "y" OR a$ = "Y" THEN
                CALL wrtcmd1("WRT " + STR$(Dlength% * 2) + ",0")

           ' CALL ibconfig(gpib0%, 20, 1)
           ' CALL gpiberr("'ibconfig' execute status")

                CALL dmawrt(pat%(), Dlength%)
        END IF
END IF
'
STOP
```

付　録　A　従来器とのコンパチビリティ

従来器(ここでは、MP1701B / MP1608A / MP1650A を示します。)で用いていたプログラムで本器を実行する場合、以下に示す項目について、多少プログラムの再編集が必要となります。

(1) ステータス・共通コマンド構成

MP1761C では488.2に準拠した共通コマンドの一部をサポートしていますが、MP1701B ではこの共通コマンドを*無しで表現していました。

コ　マ　ン　ド	MP1701B	MP1761C
サービス・リクエスト	STB?	*STB?
サービス・リクエスト・イネーブル・レジスタ	SRQ	*SRE
標準イベント・ステータス・レジスタ	ESR?	*ESR?
標準イベント・ステータス・イネーブル・レジスタ	ESE	*ESE
拡張イベント・ステータス・レジスタ	EER?	ESR1? ESR2?
拡張イベント・ステータス・イネーブル・レジスタ	EES	ESE1? ESE2?

その他の共通コマンド、およびステータス構成は「第8章　ステータス・ストラクチャー」を参照してください。

(2) その他の GPIB コマンド

表 A-1に MP1701B 等との GPIB コマンド対応表を示します。
各機器の GPIB 取扱説明書を見比べて、プログラムの再編集をお願いします。

○ : MP1701B と共通のコマンド
× : MP1701B と共通でないコマンド

表A-1 デバイス・メッセージ一覧表

機能項目	コントロール・メッセージ		データ・リクエスト・メッセージ	デバイス・メッセージ詳細		MP1701Bとのコンパチビリティ
	ヘッダ部	数値データ部	ヘッダ部	No.	ページ	
● INTERNAL CLOCK セクション						
内部クロック周波数	FRQ	NR1 形式	FRQ?	1	P9-20	○
内部クロック分解能切替	RES	NR1 形式	RES?	2	P9-22	○
● MEMORY セクション						
ファイルNo./ディレクトリモード切替	FIL	NR1 形式	FIL?	3	P9-24	○
フロッピー・データ・リコール	RCL	NR1 形式	−	4	P9-25	○
フロッピー・データ・デリート	DEL	NR1 形式	−	5	P9-26	×
フロッピー・データ・セーブ	SAV	NR1 形式	−	6	P9-27	○
フロッピー・データ・リセーブ	RSV	NR1 形式	−	7	P9-28	○
メモリ・モード切替	MEM	NR1 形式	MEM?	8	P9-29	○
フロッピー・フォーマット	FDF	−	−	9	P9-30	×
ファイル内容の検索	−	−	FSH?	10	P9-31	○
メモリ・フロッピーモード	−	−	FMD?	11	P9-33	×
フロッピー・アクセス状態	−	−	MAC?	12	P9-34	○
フロッピー・エラーメッセージ	−	−	FDE?	13	P9-35	×
● PATTERNセクション						
パターン論理	LGC	NR1 形式	LGC?	14	P9-37	○
発生パターン切替	PTS	NR1 形式	PTS?	15	P9-38	○
ZERO SUBST / PRBS段数	PTN	NR1 形式	PTN?	16	P9-39	×
PRBSマーク率	MRK	NR1 形式	MRK?	17	P9-40	○
オルタネート・パターン A/B表示切替	ALT	NR1 形式	ALT?	18	P9-41	×
誤り挿入	EAD	NR1 形式	EAD?	19	P9-42	×
オルタネート・A/B・ループ回数	LPT	NR1 形式	LPT?	20	P9-43	×
データ長	DLN	NR1 形式	DLN?	21	P9-44	○
ZERO SUBST長	ZLN	NR1 形式	ZLN?	22	P9-46	×

表A-1 デバイス・メッセージ一覧表(つづき)

機能項目	コントロール・メッセージ		データ・リクエスト・メッセージ	デバイス・メッセージ詳細		MP1701Bとのコンパチビリティ
	ヘッダ部	数値データ部	ヘッダ部	No.	ページ	
● PATTERNセクション(つづき)						
ページ数/パターン同期トリガ位置	PAG ADR	NR1形式 NR1形式	PAG? ADR?	23	P9-47	○
パターン・ビット	BIT	NR1形式 HEX形式	BIT?	24	P9-49	○
パターン・データ・プリセット(全ページ・全ビット)	ALL	NR1形式	−	25	P9-51	○
パターンデータ・プリセット(1ページ・全ビット)	PST	NR1形式	−	26	P9-52	○
パターン同期トリガ位置	PSP	NR2形式	PSP?	27	P9-53	×
ページ数/パターン同期トリガ位置表示切替	PPD	NR1形式	PPD?	28	P9-55	×
● OUTPUTセクション						
データ出力終端電圧	DTM	NR1形式	DTM?	29	P9-57	○
クロック1出力終端電圧	CTM	NR1形式	CTM?	30	P9-58	○
オフセット基準値	OFS	NR1形式	OFS?	31	P9-59	○
データ出力振幅	DAP	NR2形式	DAP?	32	P9-60	○
$\overline{\text{データ}}$出力振幅	NAP	NR2形式	NAP?	33	P9-61	○
データ出力オフセット	DOS	NR2形式	DOS?	34	P9-62	○
$\overline{\text{データ}}$出力オフセット	NOS	NR2形式	NOS?	35	P9-67	○
クロック1出力遅延時間	CDL	NR2形式	CDL?	36	P9-69	○
クロック1出力振幅	CAP	NR2形式	CAP?	37	P9-70	○
クロック1出力オフセット	COS	NR2形式	COS?	38	P9-71	○
出力オン/オフ	OON	NR1形式	OON?	39	P9-73	×
データ/$\overline{\text{データ}}$表示切替	DDS	NR1形式	DDS?	40	P9-74	×
データ/$\overline{\text{データ}}$・トラッキング	TRK	NR1形式	TRK?	41	P9-75	○
1/1 SPEED, 1/4 SPEED表示切替	SPD	NR1形式	SPD?	42	P9-76	○
● 正面パネル						
同期信号出力選択	SOP	NR1形式	SOP?	43	P9-78	×

表A-1 デバイス・メッセージ一覧表(つづき)

機能項目	コントロール・メッセージ		データ・リクエスト・メッセージ	デバイス・メッセージ詳細		MP1701Bとのコンパチビリティ
	ヘッダ部	数値データ部	ヘッダ部	No.	ページ	
● 背面パネル 誤り挿入チャンネル	ECH	NR1形式	ECH?	44	P9-79	○
● ファンクションスイッチ マーク率のANDビットシフト数	SFT	NR1形式	SFT?	45	P9-80	○
外部誤り挿入	EEI	NR1形式	EEI?	46	P9-81	×
オルタネート・パターンA/B切替信号選択	APS	NR1形式	APS?	47	P9-82	×
● その他 初期化	INI	−	−	48	P9-83	○
パターン・データ入力バイト数	WRT	NR1形式	−	49	P9-84	○
パターン・データ出力バイト数	RED	NR1形式	−	50	P9-85	○
内蔵シンセPLL	−	−	PLL?	51	P9-86	○
内部タイマ設定	RTM	NR1形式	RTM?	52	P9-87	○
電源断、回復状態	−	−	PWI?	53	P9-88	○
ディレイ状態	−	−	DLY?	54	P9-89	○
メモリ・フロッピーモード	−	−	FMD?	11	P9-33	×
終端コード切り替え	TRM	NR1形式	TRM?	55	P9-90	○

付　録　B　パターンのDMA転送

DMAとは、Direct Memory Accessの略で、大量のデータ転送を高速に行うことです。

本器には、DMA転送コマンドを2種類もうけています。この2種類の転送方法について説明します。

(1) パターンデータのDMA転送コマンド(WRT)

測定パターンがALTERNATE, DATAの場合にプログラム・パターンのDMA転送内容を前もって送出するコマンドです。

本コマンドでは、

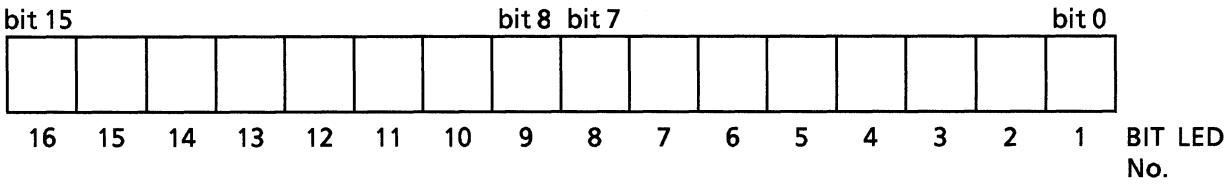
1：何バイトのデータを転送するのか?
2：転送したパターンデータを本器内部RAMエリアの何番目のアドレスから格納するのか?

の情報をWRTをもちいて本器に知らせます。

1) 何バイトのデータを転送するのか?

本器は16 bitのパターン構成になっています。そのため通常表示されているBIT表示器1ページ(16 bit)に対して2バイトの転送を行います。

パターンデータとビットの対応は、以下の様になっています。



奇数バイト転送の場合は、上位ビット(bit 15 ~ bit 8)のみ設定されます。

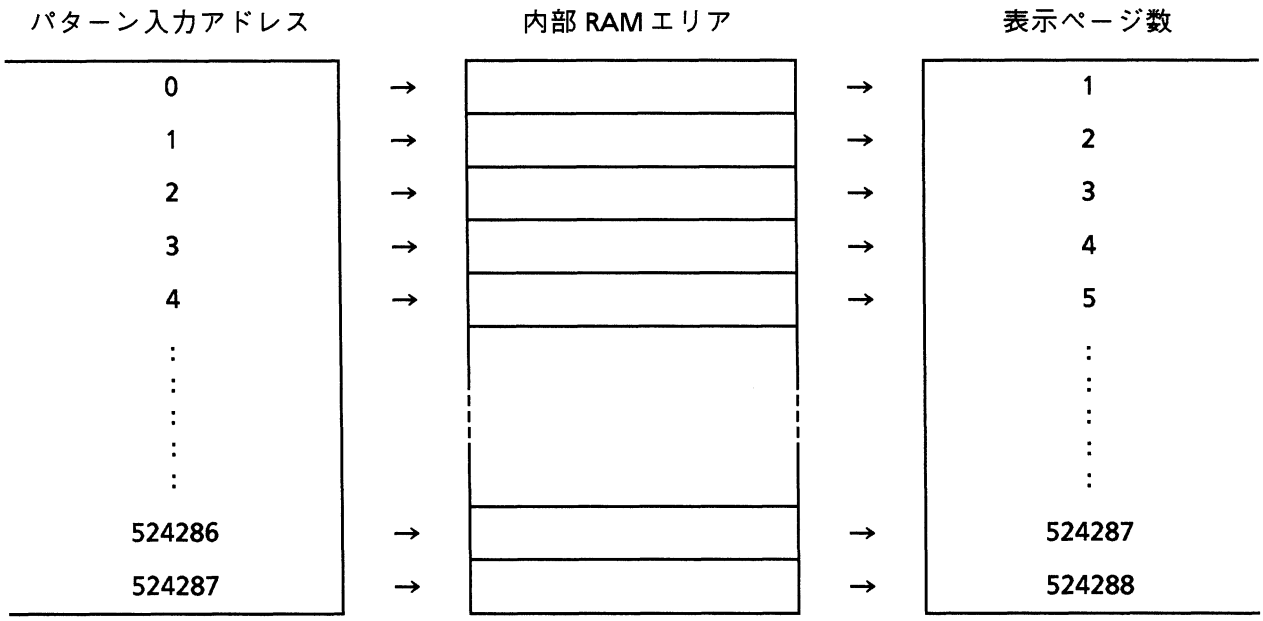
2) 転送したパターンデータを本器内部 RAM エリアの何番目のアドレスから格納するのか?

本器の内部 RAM アドレスは、

DATA の場合　　：0～524287
ALTERNATE の場合：0～262143

で、ALTERNATE の場合は A パターン/B パターンの2種類で内部 RAM エリアを2分割しています。(A/B 表示切替の状態によって A パターン・B パターンの転送を選択します。)

また、以下に実際のページ数とアドレスとの関係を示します。



(2) パターンデータの DMA 転送コマンド (RED?)

測定パターンが ALTERNATE, DATA の場合にプログラム・パターンの DMA 転送内容を前もって送出するコマンドです。

本コマンドでは、

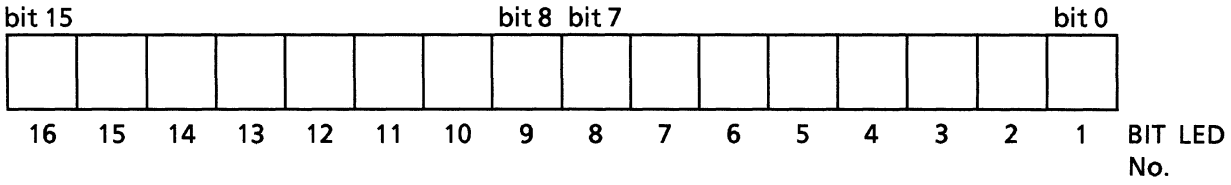
1：何バイトのデータを転送するのか?
2：転送するパターンデータを本器内部 RAM エリアの何番目のアドレスから出力するのか?

の情報を RED? をもちいて本器に知らせます。

1) 何バイトのデータを転送するのか?

本器は16 bit のパターン構成になっています。そのため通常表示されている BIT 表示器 1ページ (16 bit) に対して2バイトの転送を行います。

パターンデータとビットの対応は、以下のようになっています。



奇数バイト転送の場合は、上位ビット (bit 15~bit 8) のみ設定されます。

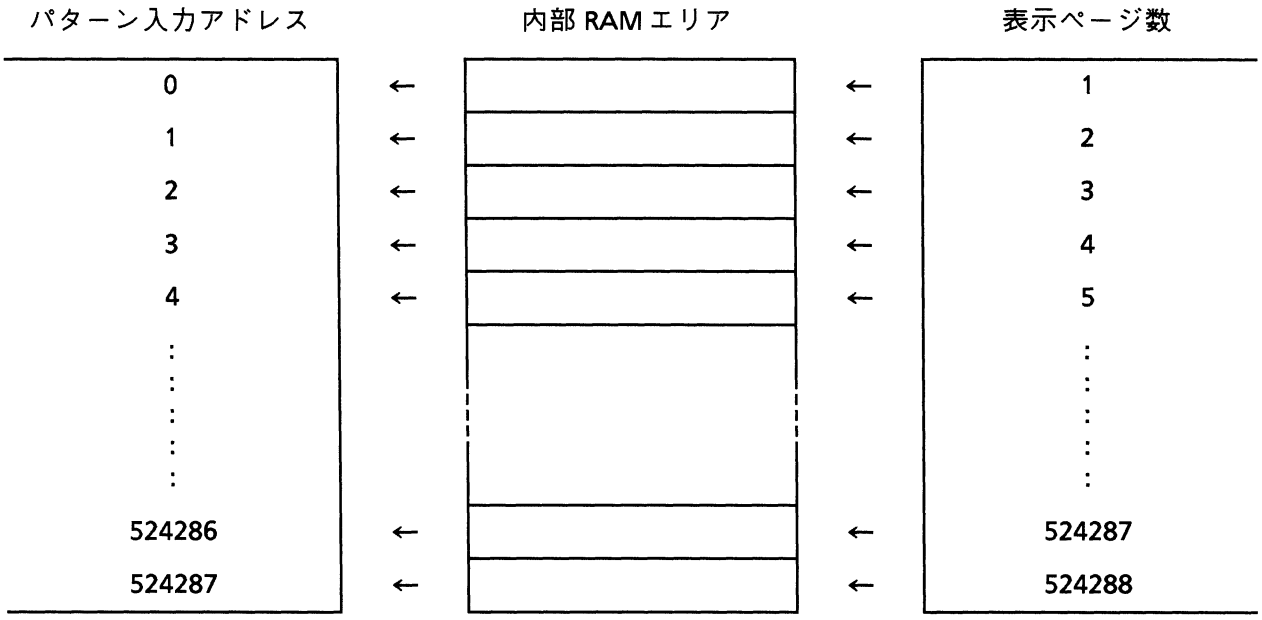
2) 転送したパターンデータを本器内部 RAM エリアの何番目のアドレスから出力するのか?

本器の内部 RAM アドレスは、

DATA の場合　　：0～524287
ALTERNATE の場合：0～262143

で、ALTERNATE の場合は A パターン/B パターンの2種類で内部 RAM エリアを2分割しています。(A/B 表示切替の状態によって A パターン・B パターンの転送を選択します。)

また、以下に実際のページ数とアドレスとの関係を示します。



付　録　C　初期設定一覧

付録の表C1からC5に、MP1761C工場出荷時の初期設定状態を示します。

表 C-1　INTERNAL CLOCK セクション初期設定状態

機　能　項　目	ヘッダ	初　期　値	デバイス・メッセージ詳細	
			項目 No.	ページ
● INTERNAL CLOCK セクション				
内部クロック周波数	FRQ	12.5 GHz	1	P9-20
内部クロック分解能切替	RES	MHz単位	2	P9-22

表C-2 MEMORYセクション初期設定状態

機能項目	ヘッダ	初期値	デバイス・メッセージ詳細	
			項目 No.	ページ
● MEMORY セクション				
ファイルNo./ディレクトリモード切替	FIL	ファイルNo.	3	P9-24
フロッピー・データ・リコール	RCL	−	4	P9-25
フロッピー・データ・デリート	DEL	−	5	P9-26
フロッピー・データ・セーブ	SAV	−	6	P9-27
フロッピー・データ・リセーブ	RSV	−	7	P9-28
メモリ・モード切替	MEM	PATT	8	P9-29
フロッピー・フォーマット	FDF	−	9	P9-30
ファイル内容の検索	FSH	データ無し	10	P9-31
メモリ・フロッピーモード	FMD	1440 K	11	P9-33
フロッピー・アクセス状態	MAC	アクセス無し	12	P9-34
フロッピー・エラーメッセージ	FDE	エラー無し	13	P9-35

表C-3 PATTERNセクション初期設定状態

機能項目	ヘッダ	初期値	デバイス・メッセージ詳細	
			項目 No.	ページ
● PATTERNセクション				
パターン論理	LGC	正論理	14	P9-37
発生パターン切替	PTS	PRBS	15	P9-38
ZERO SUBST / PRBS段数	PTN	PRBS : $2^{15}-1$ Z.S. : 2^{7}	16	P9-39
PRBSマーク率	MRK	1/2	17	P9-40
オルタネート・パターン A/B表示切替	ALT	Aパターン	18	P9-41
誤り挿入	EAD	外部誤り挿入無し	19	P9-42
オルタネート・A/B・ループ回数	LPT	A, Bパターンとも1回	20	P9-43
データ長	DLN	Alternate : 128 bits Data : 2 bits	21	P9-44
ZERO SUBST長	ZLN	1ビット	22	P9-46
ページ数/パターン同期トリガ位置	PAG ADR	1ページ	23	P9-47
パターン・ビット	BIT	全ビット0	24	P9-49
パターン同期トリガ位置	PSP	1ページ	27	P9-53
ページ数/パターン同期トリガ位置表示切替	PPD	ページ数表示	28	P9-55

表C-4 OUTPUTセクション初期設定状態

機能項目	ヘッダ	初期値	デバイス・メッセージ詳細	
			項目 No.	ページ
● OUTPUTセクション				
データ出力終端電圧	DTM	GND	29	P9-57
クロック1出力終端電圧	CTM	GND	30	P9-58
オフセット基準値	OFS	VOH	31	P9-59
データ出力振幅	DAP	1.0Vp-p	32	P9-60
$\overline{\text{データ}}$出力振幅	NAP	1.0Vp-p	33	P9-61
データ出力オフセット	DOS	0.0V	34	P9-62
$\overline{\text{データ}}$出力オフセット	NOS	0.0V	35	P9-67
クロック1出力遅延時間	CDL	0ps	36	P9-69
クロック1出力振幅	CAP	1.0Vp-p	37	P9-70
クロック1出力オフセット	COS	0.0V	38	P9-71
出力オン/オフ	OON	出力オフ	39	P9-73
データ/$\overline{\text{データ}}$表示切替	DDS	データ表示	40	P9-74
データ/$\overline{\text{データ}}$・トラッキング	TRK	トラッキング・オフ	41	P9-75
1/1 SPEED, 1/4 SPEED表示切替	SPD	1/1スピード	42	P9-76

表C-5　その他のセクション初期設定状態

機　能　項　目	ヘッダ	初　期　値	デバイス・メッセージ詳細	
			項目 No.	ページ
● 正面パネル 同期信号出力	SOP	1/64CLOCK	43	P9-78
● 背面パネル 誤り挿入チャンネル	ECH	CH.1	44	P9-79
● ファンクションスイッチ マーク率のANDビットシフト数	SFT	1bit shift	45	P9-80
外部誤り挿入	EEI	外部誤り挿入オフ	46	P9-81
オルタネート・パターン A/B切替信号選択	APS	切替信号内部発生	47	P9-82
● その他 内部タイマ設定	RTM	現在時刻	52	P9-87
終端コード切り替え	TRM	LF＋EOI	55	P9-90

（空白）

付　録　D　トラッキング項目一覧表

ここでは、MP1761Cのトラッキング項目について表D-1に示します。

トラッキングとは、MP1761Cの設定状態をMP1762A/CにGPIB経由で送信する機能で接続方法は、「3章　バス接続とアドレス設定」を参照してください。

また、トラッキング項目は測定パターンによってことなります。

表D-1　トラッキング項目一覧表

測定パターン	トラッキング項目
Alternateパターン	1：LOGIC 2：発生パターン(Alternate) 3：A/B表示切替 4：ページ設定 5：パターンビット(DMA転送、A/Bともに送出)
Dataパターン	1：LOGIC 2：発生パターン(Data) 3：データ長 4：ページ設定 5：パターンビット(DMA転送)
Zero subst.パターン	1：LOGIC 2：発生パターン 3：Zero subst.段数 4：Zero subst長 5：ページ設定
PRBSパターン	1：LOGIC 2：発生パターン 3：PRBS段数 4：PRBSマーク率 5：ページ設定

（空白）