プログラマ・マニュアル

Tektronix

DTG5078 型 / DTG5274 型 / DTG5334 型
データ・タイミング・ゼネレータ

071-1614-00

本マニュアルはファームウェア・バージョン
2.0.0 以降に対応しています。

www.tektronix.com

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

日本テクトロニクス株式会社　〒 108-6106　東京都港区港南 2-15-2　インターシティ B 棟 6F

目　次

図一覧

表一覧

第１章　はじめに

はじめに

DTG5000 シリーズは GPIB を装備しています。PC など外部コントローラのアプリケーションから前面パネルの設定とパターン・データの転送等をリモート・コントロールできます。

機器の機能や操作法などの詳細は、付属のユーザ・マニュアル 2 (071-1613-XX) を参照してください。

第２章　コマンドと構文

コマンドの構文

BNF 表記法の定義

このマニュアルでは、Backus-Naur Form (BNF) 表記法を用いてコマンドと問い合せを記述しています。

記　号	意　味
< >	定義された要素
::=	左辺を右辺として定義
|	排他的論理和
{ }	グループ（1 つの要素は必要です）
[ ]	オプション（省略可能）
…	前の要素の繰り返し
( )	コメント

ファイル名などの日本語使用について

ファイル名、パス名は日本語が使えます。

ただし、それ以外のブロック名などは英数字しか使えません。

SCPI コマンドと問い合せ

SCPI (Standard Commands for Programmable Instruments) は、計測機器のリモートプログラミングのガイドラインを定めるコンソシアムで作成された標準規格です。このガイドラインでは、機器のコントロールとデータ転送のためのプログラミング環境を実現しています。この環境では、メーカーによらず、すべての SCPI 機器で定義されたプログラミング・メッセージ、機器応答、およびデータ・フォーマットが使用できます。本機器では、この SCPI 標準を基にしたコマンド言語を使用しています。

SCPI 言語は、ツリー構造になっています。ツリーの上位レベルは、ルート・ノードで、その下に一つ、または複数の下位レベル・ノードが続きます。

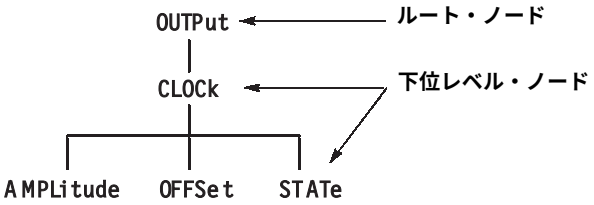


図 2-1 ：SCPI サブシステムのツリー構造

設定コマンドと問い合せコマンドは、これらサブシステムの階層ツリーから作成できます。設定コマンドを使い、機器の動作を指定します。また、問い合せコマンドを使い、測定データとパラメータ設定に関する情報を問い合せます。

コマンドの作成

SCPI コマンドは、サブシステムのノードと、各ノードを区切るコロン (:) で作成されます。

図 2-1 で、OUTPut はルート・ノードで、CLOCk、AMPLitude、OFFSet などは下位レベル・ノードです。SCPI コマンドを作成するには、ルート・ノードの OUTPut からツリー構造の下方に向ってノードを追加していきます。ほとんどのコマンドといくつかの問い合せはパラメータを持っており、パラメータ値を追加する必要があります。各コマンドのパラメータについては、「コマンドの詳細」を参照してください。

たとえば、OUTPut:CLOCk:AMPLitude 2.0 は、図 2-1 の階層ツリーから作成された有効な SCPI コマンドです。

問い合せコマンドの作成

問い合せコマンドを作成するには、ツリー構造のルート・ノードから下方に向かってノードを追加して行き、最後に疑問符 (?) を追加します。OUTPut:CLOCk:AMPLitude? は、図 2-1 の階層ツリーを使用した有効な SCPI 問い合せの例です。

応答

DTG5000 シリーズに問い合せコマンドを送ると、設定条件またはステータスが返されます。応答は、値だけが返されます。値がニーモニックの場合は、短縮形で表記されます。

表 2-1：問い合せ

問い合せ	応答
SYSTem:VERSion?	1999.0
DIAGnostic:SELect?	SYST

問い合せコマンドには、値を返す前に、ある操作を実行するものもあります。たとえば、*CAL?

問い合せコマンドは校正を実行します。

パラメータ・タイプ

コマンドと問い合せの記述内すべての引数は、独自のパラメータ・タイプを持っています。引数は、<file_name> などのように括弧で囲まれています。引数には、DTG5000 シリーズのコマンド･セットで定義されたものと SCPI で定義されたものがあります。パラメータ・タイプも <Numeric> のように括弧で囲まれて表されます。表 2-2 にパラメータ・タイプをまとめてあります。

表 2-2：構文記述で用いるパラメータ・タイプ

パラメータ・タイプ	記述	例
任意ブロック	**指定長の任意データ**	#512234xxxxx... **ここで、5 はそれに続く 5 桁 (12234) の数がデータ長 (バイト) を指定していることを表します。**xxxxx... **はデータを表します。** **または** #0xxxxx...<LF><&EOI>
ブーリアン (boolean)	**ブーリアン数または** NRf	ON **または** 0 **以外** OFF **または** 0
離散値	**特定値**	MIN、MAX
基数 (radix)	**特定値**	BINary、HEXadecimal、OCTal

表 2-2：構文記述で用いるパラメータ・タイプ （続き）

パラメータ・タイプ	記述	例
2 進	2 進数	#B0110
8 進	8 進数	#Q57、#Q3
16 進	16 進数	#HAA、#H1
NR1 数値	整数	0、1、15、-1
NR2 数値	小数	1.2、3.141516、-6.5
NR3 数値	浮動小数	3.1415E-9、-16.1E5
NRf 数値	NR1、NR2、NR3 のいずれも可能な 10 進数	NR1、NR2、NR3 の各例を参照してください。
Numeric	NR1、NR2、NR3 のいずれも可能な 10 進数、または特定値 (MIN、MAX)	NR1、NR2、NR3、離散値の各例を参照してください。
文字列 (string)	文字種はアスキーコード 32 (スペース) から 126(~) まで。(引用符で囲まれていることが必要。引用符を文字列内で使う場合は 2 回繰り返します。)	"Test 1, 2, 3" "AB""c""DE"

MIN、MAX に関して

Numeric パラメータを持つコマンドでは数値 (NR1、NR2、NR3) の他に MINimum、MAXimum というキーワードが使えます。

このキーワードを使って最大値、最小値に設定できます。

また問い合わせのときに使うと、その時点で設定可能な最大値、最小値を問い合わせることもできます。

特殊文字

改行 (LF、ASCII 10) と ASCII 127 ～ 255 の範囲の文字は、特殊文字として定義されています。これらの文字は任意ブロック引数だけで使います。コマンドの他の部分で使うと、予期されない結果が生じる場合があります。

コマンド、問い合せ、パラメータの短縮

SCPI コマンド、問い合せ、およびパラメータのほとんどは、短縮形で記述することができます。このマニュアルでは、これらの短縮形を大文字と小文字の組み合わせで示します。大文字はコマンドの短縮形を表します。図 2-2 に示すように、大文字だけでコマンドを記述できます。短縮したコマンドと短縮されないコマンドは等価で、機器に同じ動作を要求します。

短縮しない記述　　OUTPut:CLOCk:AMPLitude 2.0
　　下線のついた部分が短縮形での記述に最小限必要な情報です。
短縮した記述　　OUTP:CLOC:AMPL 2.0

図 2-2：短縮したコマンドの例

注：コマンドまたは問い合せの最後に付けられた数値 (サフィックス) は、短縮しない記述と短縮した記述のどちらにも含まれます。サフィックスを付けない場合には、デフォルトとして 1 が適用されます。

複数のコマンドと問い合せの連結

コマンドまたは問い合せは 1 つのメッセージ内で連結できます。連結したメッセージを作成するには、最初にコマンドまたは問い合せを作成し、セミコロン (;) を追加し、それからコマンドまたは問い合せを追加していきます。セミコロンに続くコマンドがルート・ノードの場合は、その前にコロン (:) を挿入してください。図 2-3 に複数のコマンドと問い合せを含む連結したメッセージを示します。連結したメッセージは、セミコロンでなくコマンドまたは問い合せで終わる必要があります。メッセージ内に含まれた問い合せに対する応答は、セミコロンで区切られます。

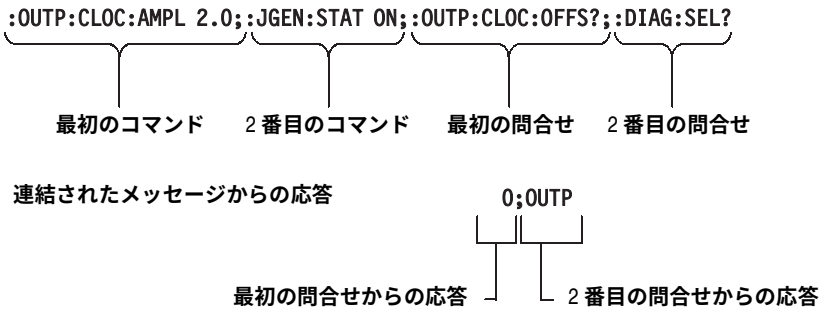


図 2-3：複数のコマンドと問い合せの連結

コマンドまたは問い合せが、前にあるコマンドまたは問い合せと共通のルート・ノードおよび下位レベル・ノードをもつ場合は、これらのノードを省略できます。図 2-4 では、2 番目のコマンドが最初のコマンドと共通の上位ノード (OUTP:DC) をもつため、これらのノードを省略できます。

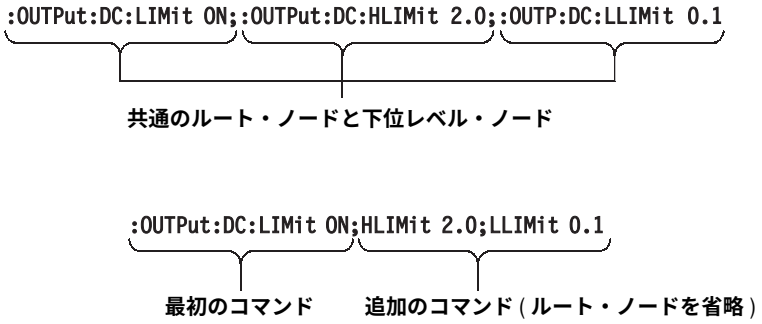


図 2-4：連結したメッセージ内での上位ノードと下位レベル・ノードの省略

単位と SI 接頭辞

引数の電圧、周波数、インピーダンス、および時間には、単位と SI 接頭辞を付加できます（SI は Systeme International d'Unites Standard に準拠した単位です）。たとえば、電圧 200e-3、周波数 1.2e+6 はそれぞれ、200mV、1.2MHz として指定できます。

単位として使用できる記号は、次のとおりです。

- V　— 電圧 (V)
- HZ　— 周波数 (Hz)
- OHM　— インピーダンス (ohm)
- S　— 時間 (s)
- DBM　— 電力比— (DBM)
- PCT　— %
- VPP　— 電圧の Peak to Peak(Vpp)
- UIPP　— 単位が UI の時の Peak to Peak (UIpp)
- UIRMS　— 単位が UI の時の実効値 (UIrms)
- SPP　— 単位が s の時の Peak to Peak (spp)
- SRMS　— 単位が s の時の実効値 (srms)
- V/NS　— SLEW コマンドで使用する単位 (V/ns)

角度の場合、単位として RADian と DEGree が使えます。単位を指定しない場合は、RADian になります。

SI 接頭辞として使用できる記号は、次のとおりです。

SI 接頭辞	定義
EX	1E18
P E	1E15
T	1E12
G	1E9
MA	1E6
K	1E3
M	1E-3
U	1E-6
N	1E-9
P	1E-12
F	1E-15
A	1E-18

注）SI 接頭辞 M は、HZ および OHM の場合には 1E6 として使われます。
注）SI 接頭辞 U は、" μ " の代わりに使用します。

単位および SI 接頭辞として使う記号は、大文字と小文字の両方が可能です。
たとえば、次の例は同じ結果になります。

170mhz、170mHz、170MHz など
250mv、250mV、250MV など

プログラムの記述は SI 単位系に合うように V の場合は mV を、Hz の場合は MHz を使用するようにしてください。

ただし、SI 接頭辞のみで使用することはできません。

正しい記述： 10MHz、10E+6Hz、10E+6
誤った記述： 10M

注：問い合せに対する返事では単位はつきません。

注：グループ名 "Group1" は、大文字と小文字はそれぞれ認識されます。

一般的な規則

SCPI コマンド、問い合せ、およびパラメータの使用について、以下の 三つの一般的な規則があります。

- 文字列を引用する場合には、引用符 (‘ ’) または二重引用符 ("") のいずれかを使用できますが、一つの文字列で両方を使用することはできません。

  正しい記述： " この文字列では、引用符を正しく使用しています "
  ‘ この文字列では、引用符を正しく使用しています ’

  誤った記述： " この文字列では、引用符を誤って使用しています '

- コマンド、問い合せ、およびパラメータを記述する場合には、大文字、小文字、または両方を混在して使用することができます。

```
OUTPUT:FILTER:LPASS:FREQUENCY 200MHZ
```

- このコマンドは、次のコマンドと同じ意味をもちます。

```
output:filter:lpass:frequency 200mhz
```

- さらに、次のコマンドとも同じ意味をもちます。

```
OUTPUT:filter:lpass:FREQUENCY 200MHz
```

注：引用符内の文字列 (たとえば、ファイル名) は、大文字と小文字が区別されます。

- ノード内またはノード間で、スペース (空白) は使用できません。

  正しい記述： OUTPUT:FILTER:LPASS:FREQUENCY 200MHZ
  誤った記述： OUTPUT: FILTER: LPASS:FREQ UENCY 200MHZ

IEEE 488.2 共通コマンド

概　要

ANSI/IEEE 488.2 規格では、コントローラと機器間のインタフェースで使用するコード、フォーマット、プロトコル、および共通コマンドと問い合せの使用方法について定義しています。本機器は、この規格に準拠しています。

コマンドと問い合せ

IEEE 488.2 共通コマンドは、アステリスク (*) の後にコマンドが続き、オプションとしてスペースとパラメータ値が続きます。IEEE 488.2 の問い合せは、アステリスクの後に問い合せコマンドと疑問符が続きます。

次は、IEEE 488.2 共通コマンドの例です。

- `*ESE 16`
- `*CLS`

次は、問い合せの例です。

- `*ESR?`
- `*IDN?`

物理チャンネルの指定方法について

本機器で、例えば High Level は

```
PGEN<x><m>:CH<n>:HIGH 2.0
```

のように設定します。

ここで <x> はスロット（A ～ H）を、<m> はメインフレーム番号 (1 ～ 3) を、<n> はチャンネル（1 ～ 4）を示します。

メインフレーム番号が 1 の時に <m> は省略する事ができます。

コマンドの分類

ここでは、最初に、機能ごとにコマンド一覧を示します。次に、「コマンドの詳細」で、アルファベット順にコマンドの詳細を説明します。

説明の中では "(?)" のマークを使用しています。コマンド・ヘッダの後ろにこのマークが付いている場合、そのコマンドは、問い合せコマンドを伴っていることを表します。それ以外のコマンドは、設定コマンドか問い合せコマンドのどちらかです。

本機器は、特に断りがない限り、SCPI (Standard Commands for Programmable Instruments) と IEEE Std 488.2-1992 に準拠しています。

このマニュアルで用いている表記法については、「コマンドの構文」を参照してください。

コマンドの機能別グループ分け

コマンドは共通コマンドとデバイス・コマンドに分かれます。

共通コマンド

共通コマンドは GPIB 機器などに対する一般的なコマンドです。

ヘッダ	説　明
*CAL?	すべての校正を実行し、その結果を返します
CALibration[:ALL](?)	すべての校正を実行します
*CLS	イベント関係のレジスタおよびキューをクリアします
DIAGnostic:DATA?	セルフテストの結果を読み取ります
DIAGnostic:IMMediate(?)	セルフテストを開始します
DIAGnostic:SELect(?)	実行するセルフテストの項目を選択します
*ESE(?)	Service Request Enable Register (SRER) の設定します
*ESR?	Standard Event Status Register (SESR) の問い合せをします
*IDN?	型名等の情報を返します
*OPC(?)	全ての処理が終了するのを待ちます
*OPT?	機器のオプションを問い合せます
*RST	機器の設定を初期状態にします
*SRE(?)	Service Request Enable Register (SRER) の設定を行います
*STB?	Status Byte Register (SBR) の値を問い合せます
SYSTem:ERRor[:NEXT]?	エラー / イベント・キューから次の項目を取り出します
SYSTem:KLOCk(?)	前面パネルとキーボードのコントロールをロックします
SYSTem:VERSion?	SCPI バージョンを問い合せます
*TRG	トリガを発生させます

ヘッダ	説　明
*TST?	セルフテストを実行し、結果を返します
*WAI	実行中のコマンドがすべて終了するまで待ちます

デバイス・コマンド

デバイス・コマンドは本機器固有のコマンドです。

ヘッダ	説　明
BLOCk:DELete	ブロックを削除します
BLOCk:DELete:ALL	全てのブロックを削除します
BLOCk:LENGth(?)	ブロック長を設定します
BLOCk:NEW	ブロックを新規作成します
BLOCk:SELect(?)	パターン・データ転送やインポートのためのブロックを選択します
GROup:DELete	グループの削除をします
GROup:DELete:ALL	全てのグループの削除をします
GROup:NEW	グループの新規作成をします
GROup:WIDTh(?)	グループのビット幅設定をします
JGENeration:AMPLitude(?)	ジッタ生成の振幅を設定します
JGENeration:AMPLitude:UNIT(?)	ジッタ生成の振幅のデフォルトの単位を設定します
JGENeration:EDGE(?)	ジッタ生成のエッジを設定します
JGENeration:FREQuency(?)	ジッタ生成の周波数を設定します
JGENeration:GSOurce(?)	ジッタ生成のゲーティング・ソースを設定します
JGENeration:MODE(?)	ジッタ生成のモードを設定します
JGENeration:PROFile(?)	ジッタ生成のプロフィールを設定します
JGENeration[:STATe] (?)	ジッタ生成のオン／オフを設定します
MMEMory:LOAD	設定ファイルを読み込みます
MMEMory:STORe	ファイルに現在の設定を保存します
OUTPut:CLOCk:AMPLitude(?)	クロック出力の振幅を設定します
OUTPut:CLOCk:OFFSet(?)	クロック出力のオフセットを設定します
OUTPut:CLOCk[:STATe](?)	クロック出力のオン／オフを設定します
OUTPut:CLOCk:TIMPedance(?)	クロック出力の終端インピーダンスを設定します
OUTPut:CLOCk:TVOLtage(?)	クロック出力の終端電圧を設定します
OUTPut:DC:HLIMit(?)	DC 出力の上限を設定します
OUTPut:DC:LEVel(?)	DC 出力の出力レベルを設定します
OUTPut:DC:LIMit(?)	DC 出力のリミットのオン／オフを設定します
OUTPut:DC:LLIMit(?)	DC 出力の下限を設定します
OUTPut:DC[:STATe](?)	DC 出力のオン／オフを設定します
OUTPut:STATe:ALL	全出力のオン／オフを設定します
PGEN<x>[m]:CH<n>:AMODe(?)	データ出力のチャンネル合成モードを設定します
PGEN<x>[m]:CH<n>:AMPLitude(?)	データ出力の振幅を設定します
PGEN<x>[m]:CH<n>:BDATa(?)	パターン・データをバイナリで転送します
PGEN<x>[m]:CH<n>:CPOint(?)	NRZ データ出力の Cross Point を設定します

ヘッダ	説　明
PGEN<x>[m]:CH<n>:DATA(?)	パターン・データを転送します
PGEN<x>[m]:CH<n>:DCYCle(?)	データ出力のデューティ・サイクルを設定します
PGEN<x>[m]:CH<n>:DTOFfset(?)	データ出力のディファレンシャル・タイミング・オフセット値を設定します
PGEN<x>[m]:CH<n>:DTOFfset:STATe(?)	データ出力のディファレンシャル・タイミングのオン／オフを設定します
PGEN<x>[m]:CH<n>:HIGH(?)	データ出力のハイ・レベルを設定します
PGEN<x>[m]:CH<n>:HLIMit(?)	データ出力のハイ・リミットを設定します
PGEN<x>[m]:CH<n>:IMPedance?	DTGM21 型の出力インピーダンスを調べます。
PGEN<x>[m]:CH<n>:JRANge(?)	DTGM32 型のジッタ・レンジを設定します。
PGEN<x>[m]:CH<n>:LDELay(?)	データ出力のリード・ディレイを設定します
PGEN<x>[m]:CH<n>:LHOLd(?)	データ出力のリーディング・エッジのホールド方法を指定します
PGEN<x>[m]:CH<n>:LIMit(?)	データ出力のリミットのオン／オフを設定します
PGEN<x>[m]:CH<n>:LLIMit(?)	データ出力レベルのロー・リミットを設定します
PGEN<x>[m]:CH<n>:LOW(?)	データ出力のロー・レベルを指定します
PGEN<x>[m]:CH<n>:OFFSet(?)	データ出力のオフセット・レベルを設定します
PGEN<x>[m]:CH<n>:OUTPut(?)	データ出力のオンオフを設定します
PGEN<x>[m]:CH<n>:PHASe(?)	データ出力の位相を設定します
PGEN<x>[m]:CH<n>:POLarity(?)	データ出力の極性を設定します
PGEN<x>[m]:CH<n>:PRATe(?)	パルス・レートを設定します
PGEN<x>[m]:CH<n>:SLEW(?)	データ出力のスルー・レートを設定します
PGEN<x>[m]:CH<n>:TDELay(?)	データ出力のトレイル・ディレイを設定します
PGEN<x>[m]:CH<n>:THOLd(?)	データ出力のトレーリング・エッジのホールド方法を指定します
PGEN<x>[m]:CH<n>:TIMPedance(?)	データ出力の終端インピーダンスを設定します
PGEN<x>[m]:CH<n>:TVOLtage(?)	データ出力の終端電圧を設定します
PGEN<x>[m]:CH<n>:TYPE(?)	DG モードでのデータ出力のフォーマットを設定します
PGEN<x>[m]:CH<n>:WIDTh(?)	データ出力のパルス幅を設定します
PGEN<x>[m]:ID?	モジュールを調べます。
SEQuence:DATA(?)	シーケンス一行分の設定をします
SEQuence:LENGth(?)	シーケンス長の設定をします
SIGNal:ASSign(?)	指定された論理チャンネル (グループ名＋ビット番号) に物理チャンネルをアサインします
SIGNal:<parameter>(?)	信号名を使って、データ出力の各種パラメータを設定します
SIGNal:BDATa(?)	パターン・データをバイナリで転送します
SIGNal:DATA(?)	パターン・データを転送します
SIGNal:IMPedance?	DTGM21 型の論理チャンネルの出力インピーダンスを調べます。
SIGNal:JRANge(?)	DTGM32 型の論理チャネルのジッタ・レンジを設定します。
SUBSequence:DATA(?)	サブシーケンスの一行分の設定をします
SUBSequence:DELete	サブシーケンス削除をします
SUBSequence:DELete:ALL	全サブシーケンスを削除します
SUBSequence:LENGth(?)	サブシーケンスの長さを変更します
SUBSequence:NEW	サブシーケンスを作成します

ヘッダ	説　明
SUBSequence:SELect(?)	サブシーケンスを選択します
TBAS:COUNt(?)	バースト・カウントを設定します
TBAS:CRANge(?)	クロック・レンジを設定します
TBAS:DOFFset(?)	ディレイ・オフセットを設定します
TBAS:EIN:IMMediate	イベントを発生させます
TBAS:EIN:IMPedance(?)	イベント入力のインピーダンスを設定します
TBAS:EIN:LEVel(?)	イベント入力レベルを設定します
TBAS:EIN:POLarity(?)	イベント入力極性を設定します。
TBAS:FREQuency(?)	周波数を設定します
TBAS:JMODe(?)	ジャンプ・モードを設定します
TBAS:JTIMing(?)	ジャンプ・タイミングを設定します
TBAS:JUMP	ソフトウェア・ジャンプを行います
TBAS:LDELay(?)	ロング・ディレイを設定します
TBAS:MODE(?)	PG のラン・モードを設定します
TBAS:OMODe(?)	動作モードを設定します
TBAS:PERiod(?)	周期を設定します
TBAS:PRATe?	PLL Multiplier Rate を問い合せます
TBAS:RSTate?	シーケンサ・ステータスを問い合せます
TBAS:RUN(?)	シーケンサをスタート、停止させます
TBAS:SMODe(?)	シーケンサ・モードを設定します
TBAS:SOURce(?)	クロック・ソースを設定します
TBAS:TIN:IMPedance(?)	トリガ入力インピーダンスを設定します
TBAS:TIN:LEVel(?)	トリガ入力レベルを設定します
TBAS:TIN:SLOPe (?)	トリガ入力極性を設定します
TBAS:TIN:SOURce(?)	トリガ入力ソースを設定します
TBAS:TIN:TIMer(?)	内部トリガ周期を設定します
TBAS:TIN:TRIGger	トリガを発生させます
TBAS:VRATe?	ベクタ・レイトを問い合せます
VECTor:BDATa(?)	バイナリ・フォーマットで、パターン・データを転送します
VECTor:BIOFormat(?)	VECTor:BDATa で転送されるデータ項目を設定します
VECTor:DATA(?)	アスキー・フォーマットで、パターン・データを転送します
VECTor:IMPort	ファイルからパターン・データを読み込みます
VECTor:IMPort:AWG	AWG シリーズ・ファイルからパターン・データを読み込みます
VECTor:IOFormat(?)	VECTor:DATA で転送するデータ項目及び書式を設定します

コマンドの詳細

この章では、コマンドをアルファベット順に挙げて、詳細を説明します。コマンドごとに、機能別分類、関連コマンド (ある場合)、構文、引数、応答、および使用例を示します。

ここでは、ヘッダ、ニーモニック、引数は、最小限表記しなければならない文字を大文字で示します。

【例】SIGNal:AMPLitude は、実際のプログラムでは SIGN:AMPL と表記できます。

“(?)” (括弧付き疑問符) の付いたコマンドは、設定と問い合せの両方に使います。コマンドの後に ”?” (疑問符) が付いているものは、ステータスの問い合せだけに使います。どちらの符号も付いていないコマンドは、設定だけに使います。

【例】

```
SIGNal:LIMit (?) ----- 設定および問い合せ
BLOCk:NEW   ---------- 設定のみ
SYSTem:VERsion? ------ 問い合せのみ
```

BLOCk:DELete （問い合せなし）

ブロックを削除します。

構　文：　BLOCk:DELete <block name>

引　数：　<block name> ::= <string> ― ブロック名

使用例：　"Block1" と名前のついたブロックを削除します。

```
BLOCk:DELete "Block1"
```

BLOCk:DELete:ALL （問い合せなし）

全てのブロックを削除します。

構　文：　BLOCk:DELete:ALL

使用例：　全てのブロックを削除します。

```
BLOCk:DELete:ALL
```

BLOCk:LENGth(?)

ブロック長を設定します。

構　文：　BLOCk:LENGth <block name>, <block length>
BLOCk:LENGth? <block name>

応　答：　<block length> = <NR1>

引　数：　<block name> ::= <string> ― ブロック名
<block length> ::= <Numeric> ― 範囲は

DTG5078：1 ～ 8,000,000
DTG5274：1 ～ 32,000,000
DTG5334：1 ～ 64,000,000

使用例：　ブロック "Block1" のブロック長を 960 に設定します。

```
BLOCk:LENGth "Block1",960
```

"Block2" のブロック長を問い合せます。

```
BLOCk:LENGth? "Block2"
```

ブロック名が存在しないときは次の応答が返ります。

```
-1
```

BLOCk:NEW （問い合せなし）

ブロックを作成します。

構　文： **BLOCk:NEW <block name>, <block length>**

引　数： <block name> ::= <string> ― ブロック名は 32 文字以内
<block length> ::= <Numeric> ― 範囲は

DTG5078：1 ～ 8,000,000
DTG5274：1 ～ 32,000,000
DTG5334：1 ～ 64,000,000

ブロックは 8000 個まで作成できます。

使用例： 名前を "Block1"、ブロック長を 960 でブロックを作成します。

```
BLOCk:NEW "Block1",960
```

BLOCk:SELect(?)

パターン・データ転送やインポートのためのブロックを選択します。

構　文： BLOCk:SELect <block name>
BLOCk:SELect?

応　答： <block name>

引　数： <block name> ::= <string> ― ブロック名

*RST で "" に戻ります。

使用例： "Block1" と言う名前のブロックを選択します。

```
BLOCk:SELect "Block1"
```

*CAL? （問い合せのみ）

Level Calibration を行い、校正が正常に終了したかどうかの結果を返します。

CALibration[:ALL]? 問い合せコマンドと同じ働きをします。

構　文： *CAL?

応　答： <NR1>

0　　正常終了。
-340　エラー検出。

使用例： 校正を行います。

```
*CAL?
```

正常終了すると、次の応答が返ります。

```
0
```

CALibration[:ALL](?)

Level Calibration を実行します。

構　文： CALibration[:ALL]
CALibration[:ALL]?

応　答： <NR1>

使用例： 校正を行います。

```
CALibration[:ALL] または CALibration[:ALL]?
```

詳しいエラーの状況を調べたい場合は、

```
CALibration[:ALL]
```

で Level Calibration を実行した後、

```
SYSTem:ERRor[:NEXT]?
```

コマンドでエラー情報を取り出します。

エラーがあるときは

```
-340, "Calibration failed"
```

の後に詳細情報が続きます。

*CLS （問い合せなし）

イベント関係のレジスタ及びキューをクリアします。

構　文： *CLS

使用例： すべてのイベント・レジスタ及びキューをクリアします。

```
*CLS
```

DIAGnostic:DATA? （問い合せのみ）

DIAGnostic:IMMediate で実行したセルフテストの結果を返します。

構　文： DIAGnostic:DATA?

応　答： <NR1>

使用例： セルフテストの結果を読み取ります。

```
DIAGnostic:DATA?
```

セルフテストが正常終了の場合、次の応答が返ります。

```
0
```

DIAGnostic:IMMediate(?)

セルフテストを実行します。

? が付く場合はセルフテストを実行し、結果を返します。

DIAGnostic:IMMediate? コマンドでは、エラーの詳細情報を得ることはできません。

DIAGnostic:IMMediate? により発生するエラーイベントは最大で一つです。最初のエラーが発生した時点でセルフテストは終了します。

? が付かない場合は単純にセルフテストを実行します。こちらの場合は、ダイアグエラーとなった場合にはイベントが発生します。イベント番号は -330, "Self test failed" で詳細情報がその後に続きます。

詳細情報はダイアグエラーコードと補助情報のセットで、画面に表示されるものと同じ内容です。

結果は DIAGnostic:DATA? で確認できます。

構　文： DIAGnostic:IMMediate
DIAGnostic:IMMediate?

応　答： <NR1>:=0 : セルフテストでエラーなし
-330 : セルフテストでエラーあり

使用例： 全てのセルフテスト・ルーチンを指定して、テストを開始し、終了後に結果を読み取ります。

```
DIAGnostic:SELect ALL;IMMediate?
```

セルフテストが正常終了の場合、次の応答が返ります。

```
0
```

DIAGnostic:SELect(?)

DIAGnostic:IMMediate で実行するセルフテスト・ルーチンを選択します。

構　文： DIAGnostic:SELect <diag item>
DIAGnostic:SELect?

応　答： <diag item>

引　数： <diag item> ::= { ALL | OUTPut | REGister | CLOCk | SMEMory | PMEMory }

引数	テスト回路
ALL	すべての回路
OUTPut	アウトプット
REGister	レジスタ
CLOCk	クロック
SMEMory	シーケンス・メモリ
PMEMory	パターン・メモリ

*RST で ALL に戻ります。

使用例： アウトプットを指定して、セルフテストを実行します。

```
DIAGnostic:SELect OUTPut;IMMediate
```

*ESE(?)

ステータス・レポーティング機能で使われる ESER(Event Status Enable Register) の値を設定または問い合せします。ステータス・レポーティングについての詳細は第 3 章を参照してください。

構　文：
```
*ESE <NR1>
*ESR?
```

応　答： <NR1>

引　数： <NR1> ― 設定範囲：0 ～ 255

ESER には、この値に対応するバイナリ・コードが設定されます。

*RST で 0 に戻ります。

使用例： ESER を 177(2 進 10110001) に設定します。この場合、ESER の PON、CME、EXE、OPC の各ビットがセットされます。

```
*ESE 177
```

次は、*ESE? に対する応答例です。

```
176
```

この場合、ESER の内容は、10110000 となります。

*ESR? （問い合せのみ）

ステータス・レポーティング機能で使われる SESR (Standard Event Status Register) の問い合せをします。SESR の内容は、読み出した後、クリアされます。

構　文： *ESR?

応　答： <NR1> ― SESR の内容が 0 ～ 255 の 10 進数で表されます。

使用例： *ESR? の応答例です。

181

この場合、SESR の内容は 2 進数で 10110101 です。

GROup:DELete （問い合せなし）

グループを削除します。

構　文： GROup:DELete <group name>

引　数： <group name> ::= <string> ― グループ名

使用例： "Group1" と言う名前のグループを削除します。

```
GROup:DELete "Group1"
```

GROup:DELete:ALL （問い合せなし）

全てのグループを削除します。

構　文： GROup:DELete:ALL

使用例： 全てのグループを削除します。

```
GROup:DELete:ALL
```

GROup:NEW （問い合わせなし）

グループを新規作成します。

構　文： GROup:NEW <group name>, <group width>

引　数： <group name> ::= <string> ― グループ名は 32 文字以内
<group width> ::= <Numeric> ― 範囲は 1 ～ 96

グループは 96 個まで作成できます。

使用例： 名前を "Group1"、ビット幅を 8 でグループを作成します。

```
GROup:NEW "Group1",8
```

GROup:WIDTh(?)

グループのビット幅を設定します。

構　文： GROup:WIDth <group name>, <group width>
GROup:WIDth? <group name>

応　答： <NR1>

引　数： <group name> ::= <string> ― グループ名
<group width> ::= <Numeric> ― 範囲は 1 ～ 96 （DTG5078 型 3 台同時使用時）

使用例： "Group1" のビット幅を 4 に設定します。

```
GROup:WIDTh "Group1",4
```

問い合せで、グループ名が存在しないときは次の応答が返ります。

```
-1
```

*IDN? （問い合せのみ）

型名等の情報を返します。

構　文：　*IDN?

応　答：　<manufacturer>,<model>, <serial number>,
<Firmware level>

引　数：　<manufacturer> ::= TEKTRONIX ― 製造元
<model> ::= DTG5334, DTG5274 または DTG5078 ― 機種名
<serial number> ::= JXXXXXX ― XXXXXX は実際の S/N
<Firmware level> ::= SCPI:99.0 FW:X.X.X ― システム・ソフトウェア・バージョン

使用例：　*IDN? の応答例です。

```
TEKTRONIX,DTG5078,0,SCPI:99.0 FW:1.0.0
```

JGENeration:AMPLitude(?)

ジッタ生成の振幅を設定します。

構　文：　JGENeration:AMPLitude <Numeric>
JGENeration:AMPLitude?

応　答：　<NR3>

引　数：　設定の時の単位は、SPP, SRMS, UIPP, UIRMS のいずれかが指定可能です。

単位を省略した時には JGENeration:AMPLitude:UNIT で指定した単位がついているものとみなされます。単位の意味については JGENeration:AMPLitude:UNIT の説明を参照してください。

設定範囲は計算が複雑なので、ユーザ・マニュアルを参照してください。MIN、MAX コマンドを使用して設定範囲の最小値と最大値を問い合せることもできます。

*RST で値は 0（単位は SPP）に戻ります。

使用例：　UNIT が SPP の時、ジッタ生成の振幅を 100ps に設定します。

```
JGENeration:AMPLitude 1e-10
```

現在の単位での設定できる最大のジッタ生成の振幅を問い合せます。

```
JGENeration:AMPLitude? MAX
```

JGENeration:AMPLitude:UNIT(?)

ジッタ生成の振幅のデフォルトの単位を設定します。

JGENeration:AMPLitudeで単位なしの数値が送られてきた時のデフォルトの単位を指定します。また JGENeration:AMPLITude? で問い合わせが行なわれた時の単位もこのコマンドで指定された単位になります。

更にこのコマンドは、Frequency を変えた時に second か UI (unit interval) か、どちらの単位での値を保持するかをも指定します。SPP、SRMS なら second で、UIPP, UIRMS なら UI です。

構　文：
```
JGENeration:AMPLitude:UNIT <amplitude unit>
JGENeration:AMPLitude:UNIT?
```

応　答：
```
<amplitude unit>
```

引　数：
```
<amplitude unit> ::= {SPP | SRMS | UIPP | UPRMS}
```

SPP ― Peak to Peak を second (秒) で表します。
SRMS ― Root Mean Square (実効値) を second (秒) で表します。
UIPP ― Peak to Peak を UI (unit interval) で表します。
UIRMS ― Root Mean Square (実効値) を UI (unit interval) で表します。

*RST で値は SPP に戻ります。

使用例： ジッタ生成の振幅のデフォルトの単位を SPP に設定します。

```
JGENeration:AMPLitude:UNIT SPP
```

JGENeration:EDGE(?)

ジッタ生成のエッジを設定します。

構　文： JGENeration:EDGE { RISe | FALL | BOTH }
JGENeration:EDGE?

応　答： { RISe | FALL | BOTH }

引　数： RISe：立上りに設定します。
FALL：立下りに設定します。
BOTH：立上り、立下り両方に設定します。

*RST で BOTH に設定されます。

使用例： ジッタ生成のエッジを立上がりにします。

```
JGENeration:EDGE RISe
```

JGENeration:FREQuency(?)

ジッタ生成の周波数を設定します（GNOise 以外）。

構　文： JGENeration:FREQuency <Numeric>
JGENeration:FREQuency?

応　答： <NR3>

引　数： 設定範囲：0.015Hz ～ 1.56MHz
ステップ：1e-3 Hz

*RST で 1e6 に戻ります。

使用例： ジッタ生成の周波数を 1MHz に設定します

```
JGENeration:FREQuency 1MHz
```

JGENeration:GSOurce(?)

どのグループの、どのビットにジッタをかけるか（Gating Source）を設定します。

構　文： JGENeration:GSOurce <logical channel>
JGENeration:GSOurce?

応　答： <logical channel>

引　数： <logical channel> ::= <string> ― 論理チャンネル。次のように表します。
<group name> ― 1 ビット幅のグループの場合
<group name>[<bit>] ― 指定されたグループ中の指定されたビット番号(この場合の[]は省略できません)

例：
CLK
Addr[0]

ジッタがかけられるのはマスタのスロット A のチャンネル 1 のみです。

*RST で "" に戻ります。

使用例： “Group1” のビット 0 にジッタ生成の Gating Source を設定します。

```
JGENeration:GSOurce "Group1[0]"
```

JGENeration:MODE(?)

ジッタ生成のモードを設定します。

構　文： JGENeration:MODE {ALL | PARTial}
JGENeration:MODE?

応　答： { ALL | PARTial }

引　数： ALL　　：出力信号全体にジッタをかけます。
PARTial：出力信号の一部分にジッタをかけます。

*RST で ALL に戻ります。

使用例： 出力信号全体にジッタをかけます。

```
JGENeration:MODE ALL
```

JGENeration:PROFile(?)

ジッタ生成の Profile を設定します。

構　文： JGENeration:PROFile <jitter profile>
JGENeration:PROFile?

応　答： <jitter profile>

引　数： <jitter profile> ― 波形の種類。

選択できるのは、次のとおりです。

SINusoid　：正弦波
SQUare　：方形波
TRIangle　：三角波
GNOise　：ガウス・ノイズ

*RST で SINusoid に戻ります。

使用例： 方形波でジッタを生成します。

```
JGENeration:PROFile SQUare
```

JGENeration[:STATe](?)

ジッタ生成のオン／オフを設定します。

構　文： JGENeration[:STATe] <boolean>
JGENeration[:STATe]?

応　答： <NR1>

引　数： OFF または <NRf> ＝ 0 ― ジッタ生成をオフにします。
ON または <NRf> ≠ 0 ― ジッタ生成をオンにします。

ただし、DG モードの Long Delay が Off で、マスタのスロット A にアウトプット・モジュールが入っていないとジッタをオンにできません。（ジッタ振幅等のパラメータ設定は可能です。）
ジッタがかけられるのはマスタのスロット A のチャンネル 1 のみです。またこの時チャンネル 2 は使用不能になります。

*RST で 0（オフ）に戻ります。

使用例： ジッタ生成をオンにします。

```
JGENeration:STATe ON
```

MMEMory:LOAD （問い合せなし）

設定ファイルを読み込みます。

構　文： MMEMory:LOAD <filename>

引　数： <filename> ― ファイル名：絶対パス

使用例： "C:¥tmp¥abc.dtg" と言う設定ファイルを読み込みます。

```
MMEMory:LOAD "C:¥tmp¥abc.dtg"
```

MMEMory:STORe（問い合せなし）

ファイルに現在の設定を保存します。

構　文： MMEMory:STORe <filename>

引　数： <filename> ― ファイル名：絶対パス

使用例： "C:¥tmp¥abc.dtg" に設定ファイルを保存します。

```
MMEMory:STORe "C:¥tmp¥abc.dtg"
```

*OPC(?)

このコマンドは、他の 二つのコマンドの間に入れ、次のコマンドを実行する前に、最初のコマンドの完了を確認するのに使います。本機器では、すべてのコマンドは外部コントローラから送られてきた順に処理されます。

構　文： *OPC
*OPC?

応　答： 1 ― 実行中の全てのコマンドが完了 (Operation Complete)

使用例： PGENA1:CH1:HIGH 2.0;*OPC

イベントが発生した時点で終了が確認できます。

```
PGENA1:CH1:HIGH 2.0;*OPC?
```

1 が返って来た時点で終了が確認できます。

*OPT? （問い合せのみ）

機器にインストールされているオプションを問い合せます。

構　文：　*OPT?

応　答：　0

使用例：　いつも 0 が返るので、実際のプログラムで使用する必要はありません

OUTPut:CLOCk:AMPLitude(?)

クロック出力の振幅を設定します。

構　文：　OUTPut:CLOCk:AMPLitude <Numeric>
OUTPut:CLOCk:AMPLitude?

応　答：　<NR3>

引　数：　設定範囲：0.03V ～ 1.25V
ステップ：10mV

*RST で 1.0V に戻ります。

使用例：　クロック出力の振幅を 0.5V に設定します。

```
OUTPut:CLOCk:AMPLitude 0.5
```

OUTPut:CLOCk:OFFSet(?)

クロック出力のオフセットを設定します。

構　文： OUTPut:CLOCk:OFFSet <Numeric>
OUTPut:CLOCk:OFFSet?

応　答： <NR3>

引　数： 設定範囲：-0.985V ～ 3.485V（振幅が 30mV の時）

ステップ：40mV

*RST で 0.48V に戻ります。

使用例： クロック出力のオフセットを 0.1V に設定します

```
OUTPut:CLOCk:OFFSet 0.1
```

OUTPut:CLOCk[:STATe](?)

クロック出力のオン／オフを設定します。

構　文： OUTPut:CLOCk[:STATe] <boolean>
OUTPut:CLOCk[:STATe]?

応　答： <NR1>

引　数： OFF または <NRf> ＝ 0 ― クロック出力をオフにします。
ON または <NRf> ≠ 0 ― クロック出力をオンにします。

*RST で 0（オフ）に戻ります。

使用例： クロック出力をオンにします。
OUTPut:CLOCk:STATe ON

OUTPut:CLOCk:TIMPedance (?)

クロック出力の終端インピーダンス（Termination Impedance）を設定します。

構　文：
```
OUTPut:CLOCk:TIMPedance <Numeric>
OUTPut:CLOCk:TIMPedance?
```

応　答：
```
<NR3>
```

引　数： 設定範囲：10ohm ～ 1Mohm、
0 以下の場合：オープン
ステップ：有効数字 3 桁、最小分解能は 1ohm

*RST で 50.0 ohm に戻ります。

使用例： クロック出力の終端インピーダンスを 40ohm に設定します。

```
OUTPut:CLOCk:TIMPedance 40
```

OUTPut:CLOCk:TVOLtage (?)

クロック出力の終端電圧（Termination Voltage）を設定します。

構　文：
```
OUTPut:CLOCk:TVOLtage <Numeric>
OUTPut:CLOCk:TVOLtage?
```

応　答：
```
<NR3>
```

引　数： 設定範囲：-2V ～ +5V
ステップ：0.1V

*RST で 0.0V に戻ります。

使用例： クロック出力の終端電圧を 1.1V にします。

```
OUTPut:CLOCk:TVOLtage 1.1
```

OUTPut:DC:HLIMit(?)

DC 出力の上限 (High Limit) を設定します。

構　文： OUTPut:DC:HLIMit <DC channel>, <Numeric>
OUTPut:DC:HLIMit? <DC channel>

応　答： <NR3>

引　数： <DC channel> ::= <NR1> (0 ～ 23)（３台同時使用のとき）
<Numeric> ::= 上限 ― ステップ：30mV、設定範囲：-3V ～ 5V

DC 出力の下限 (Low Limit) が上限 (High Limit) よりも大きな場合は、上限と同じ値に設定されます。

*RST で 1.0V に戻ります。

使用例： DC 出力 (0) の上限を 1.5V にします。

```
OUTPut:DC:HLIMit 0, 1.5
```

OUTPut:DC:LEVel(?)

DC 出力のレベルを設定します。

構　文： OUTPut:DC:LEVel <DC channel>, <Numeric>
OUTPut:DC:LEVel? <DC channel>

応　答： <NR3>

引　数： <DC channel> ::= <NR1> (0 ～ 23)（３台同時使用のとき）

<Numeric> ::= レベル ― ステップ:30mV、設定範囲:DC 出力の下限 (Low Limit) ～ DC 出力の上限 (High Limit)

*RST で 1.0 に戻ります。

使用例： DC 出力 (0) のレベルを 1.1V に設定します。

```
OUTPut:DC:LEVel 0,1.1
```

OUTPut:DC:LIMit(?)

DC 出力のリミットのオン／オフを設定します。

構　文：
```
OUTPut:DC:LIMit <DC channel>, <boolean>
OUTPut:DC:LIMit? <DC channel>
```

応　答：
```
<NR1>
```

引　数：
```
<DC channel> ::= <NR1> (0 ～ 23)（3 台同時使用のとき）
OFF または <NRf> = 0 — DC 出力のリミットをオフにします。
ON または <NRf> ≠ 0 — DC 出力のリミットをオンにします。

    *RST で 0 に戻ります。
```

使用例：
```
DC 出力 (1) のリミットをオンにします。

OUTPut:DC:LIMit 1,ON
```

OUTPut:DC:LLIMit(?)

DC 出力の下限 (Low Limit) を設定します。

構　文：
```
OUTPut:DC:LLIMit <DC channel>, <Numeric>
OUTPut:DC:LLIMit? <DC channel>
```

応　答：
```
<NR3>
```

引　数：
```
<DC channel> ::= <NR1> (0 ～ 23)（3 台同時使用のとき）
<Numeric> ::= 下限 — ステップ：30mV、設定範囲：-3V ～ 5V

DC 出力の上限 (High Limit) が下限 (Low Limit) よりも小さな場合は、下限と同じ値に
設定されます。

    *RST で 0 に戻ります。
```

使用例：
```
DC 出力 (0) の下限を -1V に設定します。

    OUTPut:DC:LLIMit 0, -1
```

OUTPut:DC[:STATe](?)

DC 出力のオン／オフを設定します。

構　文： OUTPut:DC[:STATe] <boolean>
OUTPut:DC[:STATe]?

応　答： <NR1>

引　数： OFF または <NRf> = 0 ― DC 出力をオフにします。
ON または <NRf> ≠ 0 ― DC 出力をオンにします。

*RST で 0 に戻ります。

使用例： DC 出力をオンにします。

```
OUTPut:DC:STATe ON
```

OUTPut:STATe:ALL

全出力（アサインされているされている全データ出力、クロック出力、DC 出力）のオン／オフを、まとめて設定します。

構　文： OUTPut:STATe:ALL <boolean>
OUTPut:STATe?

引　数： OFF または <NRf> = 0 ― 全出力をオフにします。
ON または <NRf> ≠ 0 ― 全出力をオンにします。

使用例： 全出力をオンにします。

```
OUTPut:STATe:ALL ON
```

PGEN<x>[<m>]:CH<n>:AMODe(?)

指定されたチャンネルのデータ出力のチャンネル合成モードを設定します。

構　文：
```
PGEN<x>[<m>]:CH<n>:AMODe <channel addition mode>
PGEN<x>[<m>]:CH<n>:AMODe?
```

応　答：
```
<channel addition mode>
```

引　数：
```
<channel addition mode> ::= {NORMal | XOR | AND}
```

NORMal ：加算しません。
XOR　：排他的論理和で合成します。
（奇数物理チャンネルのみ、選択できます。）
AND　：論理積で合成します。
（偶数物理チャンネルのみ、選択できます。）

*RST で NORMal に戻ります。

使用例： メインフレーム 1、スロット A、チャンネル 1 とチャネル 2 を AND モードで合成します。

```
PGENA:CH2:AMODE AND
```

PGEN<x>[<m>]:CH<n>:AMPLitude(?)

指定されたチャンネルのデータ出力の振幅を設定します。

構　文：
```
PGEN<x>[<m>]:CH<n>:AMPLitude <Numeric>
PGEN<x>[<m>]:CH<n>:AMPLitude?
```

応　答：
```
<NR3>
```

引　数： 設定範囲：0.1V ～ 3.5V
ステップ：5mV

*RST で 1V に戻ります。

使用例： メインフレーム 1、スロット A、チャンネル 1 の振幅を 1.2V に設定します。

```
PGENA:CH1:AMPLitude 1.2
```

PGEN<x>[<m>]:CH<n>:BDATa(?)

指定されたチャンネルのパターン・データをバイナリで転送します。

構　文： PGEN<x>[<m>]:CH<n>:BDATa <start vector>, <vector size>, <binary pattern data>
PGEN<x>[<m>]:CH<n>:BDATa? <start vector>, <vector size>

応　答： <binary pattern data>

引　数： <start vector> ::= <NR1> ― データのスタート・アドレス
<vector size> ::= <NR1> ― データのサイズ
<binary pattern data> ::= <block data> ― バイナリ・バイトのブロック

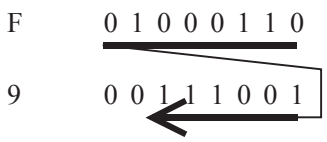
注：一回で転送できるパターン・データの量は最大 1MB です。

使用例： PGENB2:CH2:BDATa 0,14,#12F9

このデータは

#：ブロックのスタート・キャラクタ
1：長さのフィールドの長さが「1」であることを示します。
2：データの長さが「2」であることを示します。
F：01000110
9：00111001

ですので、メインフレーム 2、スロット B、チャンネル 2 の頭から 14 ベクタ分のデータがそれぞれ、0,1,1,0,0,0,1,0,1,0,0,1,1,1 に設定されます。



PGENB2:CH2:BDATa? 2,10

このコマンドで、メインフレーム 2、スロット B、チャンネル 2 のアドレス 2 から 10 ベクタ分のデータを読み取ることができます。

PGEN<x>[<m>]:CH<n>:CPOint(?)

NRZ データ出力の Cross Point を設定します。

構　文： PGEN<x>[<m>]:CH<n>:CPOint <Numeric>
PGEN<x>[<m>]:CH<n>:CPOint?

応　答： <NR3>

引　数： 設定範囲：30% ～ 70%
ステップ：2%

*RST で 50% に戻ります。

使用例： メインフレーム 1、スロット A、チャンネル 1 のクロス・ポイントを 30% に設定します。

```
PGENA:CH1:CPOint 30
```

PGEN<x>[<m>]:CH<n>:DATA(?)

指定されたチャンネルのパターン・データを転送します。

構　文： PGEN<x>[<m>]:CH<n>:DATA <start vector>, <vector size>, <ascii pattern data>
PGEN<x>[<m>]:CH<n>:DATA? <start vector>, <vector size>

応　答： <ascii pattern data>

引　数： <start vector> ::= <NR1> ― データのスタート・アドレス
<vector size> ::= <NR1> ― データのサイズ
<ascii pattern data> ::= <string> ― データ文字列

注： 一回で転送できるパターン・データの量は最大 1MB です。

使用例：

```
PGENB:CH2:DATA 0,16,"0100011100111001"
```

というコマンドはメインフレーム 1、スロット B、チャンネル 2 にアドレス 0 から 16 ベクタ分データをそれぞれ
0,1,0,0,0,1,1,1,0,0,1,1,1,0,0,1 に設定します。

```
PGENB:CH2:DATA? 2,10
```

このコマンドで、メインフレーム 1、スロット B、チャンネル 2 のアドレス 2 から 10 ベクタ分のデータを読み取ることができます。

PGEN<x>[<m>]:CH<n>:DCYCle(?)

指定されたチャンネルのデータ出力のデューティ・サイクルを設定します。

構　文： PGEN<x>[<m>]:CH<n>:DCYCle <Numeric>
PGEN<x>[<m>]:CH<n>:DCYCle?

応　答： <NR3>

引　数： 設定範囲：0% ～ 100%（0% と 100% は含まれません。）
DG モード（ロング・ディレイ・オフ）の場合：
パルス幅が 290ps から Period 290ps の範囲にあること
DG モード（ロング・ディレイ・オン）の場合：
パルス幅が 290ps から Period 290ps の範囲にあること
PG モード：
パルス幅が 290ps から Period * Pulse Rate 290ps の範囲にあること
ステップ：0.1%

*RST で 50% に戻ります。

使用例： メインフレーム 1、スロット B、チャンネル 2 のデューティ・サイクルを 1% に設定します。

```
PGENB:CH2:DCYCle 1
```

PGEN<x>[<m>]:CH<n>:DTOFfset(?)

指定されたチャンネルのデータ出力のディファレンシャル・タイミング・オフセット値を設定します。

構　文：
```
PGEN<x>[<m>]:CH<n>:DTOFfset <Numeric>
PGEN<x>[<m>]:CH<n>:DTOFfset?
```

応　答：
```
<NR3>
```

引　数： 設定範囲：-1ns ～ 1ns
ただし、Lead Delay の設定範囲を超えることはできません。

ステップ：
DTG5078：1ps
DTG5274/5334：0.2ps

*RST で 0 に戻ります。

使用例： メインフレーム 1、スロット B、チャンネル 2 のデータ出力のディファレンシャル・タイミング・オフセット値を 1ps に設定します。

```
PGENB:CH2:DTOFfset 1ps
```

PGEN<x>[<m>]:CH<n>:DTOFfset:STATe(?)

指定されたチャンネルの、データ出力のディファレンシャル・タイミング・オフセット (Differential Timing Offset) のオン／オフを設定します。

構　文：
```
PGEN<x>[<m>]:CH<n>:DTOFfset:STATe <boolean>
PGEN<x>[<m>]:CH<n>:DTOFfset:STATe?
```

応　答：
```
<NR1>
```

引　数： OFF または <NRf> ＝ 0 ― ディファレンシャル・タイミング・オフセットをオフにします。
ON または <NRf> ≠ 0 ― ディファレンシャル・タイミング・オフセットをオンにします。

*RST で 0 に戻ります。

使用例： メインフレーム 2、スロット A、チャンネル 1 のディファレンシャル・タイミング・オフセットをオンにします。

```
PGENA2:CH1:DTOFset:STATe ON
```

PGEN<x>[<m>]:CH<n>:HIGH(?)

指定されたチャンネルのデータ出力のハイ・レベル (High Level) を設定します。

構　文： PGEN<x>[<m>]:CH<n>:HIGH <Numeric>
PGEN<x>[<m>]:CH<n>:HIGH?

応　答： <NR3>

引　数： ステップは 5mV。
設定範囲は計算が複雑なので、ユーザ・マニュアルを参照してください。MIN、MAX コマンドを使用して設定範囲の最小値と最大値を問い合せることもできます。

*RST で 1.0 に戻ります。

使用例： メインフレーム 1、スロット A、チャンネル 1 のデータ出力のハイ・レベルを 1.05V に設定します。

```
PGENA:CH1:HIGH 1.05
```

現在の設定できる最大のデータ出力のハイ・レベルを問い合せます。

```
PGENA:CH1:HIGH? MAX
```

PGEN<x>[<m>]:CH<n>:HLIMit(?)

データ出力のハイ・リミット (High Limit) を設定します。

構　文： PGEN<x>[<m>]:CH<n>:HLIMit <Numeric>
PGEN<x>[<m>]:CH<n>:HLIMit?

応　答： <NR3>

引　数： ステップは 5mV。
設定範囲は計算が複雑なので、ユーザ・マニュアルを参照してください。MIN、MAX コマンドを使用して設定範囲の最小値と最大値を問い合せることもできます。

*RST で 1.0 に戻ります。

使用例： メインフレーム 2、スロット A、チャンネル 1 のデータ出力のハイ・リミットを 1.05V に設定します。

```
PGENA2:CH1:HLIMit 1.05
```

現在の設定できる最大のデータ出力のハイ・リミットを問い合せます。

```
PGENA2:CH1:HLIMit? MAX
```

PGEN<x>[<m>]:CH<n>:IMPedance?（問い合せのみ）

DTGM21 型の出力インピーダンスを調べます。

構　文： PGEN<x>[<m>]:CH<n>:IMPedance?

応　答： <NR3>
50.0 または 23.0 が返ります。

使用例： メインフレーム 2、スロット A、チャンネル 1 の出力インピーダンスを調べます。

```
PGENA2:CH1:IMPedance?
```

インピーダンスが 50 ohm に設定されている場合、次の応答が返ります。

```
50.0
```

PGEN<x>[<m>]:CH<n>:JRANge(?)

DTGM32 型のジッタ・レンジを設定します。

構　文：
```
PGEN<x>[<m>]:CH<n>:JRANge <Numeric>
PGEN<x>[<m>]:CH<n>:JRANge?
```

応　答： <NR3>

引　数： 設定範囲：1e-9 s または 2e-9 s

*RST で 2e-9 s に戻ります。

使用例： メインフレーム 2、スロット A、チャンネル 1 のジッタ・レンジを 1 ns に設定します。

```
PGENA2:CH1:JRANge 1ns
```

PGEN<x>[<m>]:CH<n>:LDELay(?)

指定されたチャンネルのデータ出力のリード・ディレイ (Lead Delay) を設定します。

構　文：
```
PGEN<x>[<m>]:CH<n>:LDELay <Numeric>
PGEN<x>[<m>]:CH<n>:LDELay?
```

応　答： <NR3>

引　数： ステップは
DTG5078：1ps
DTG5274：0.2ps

設定範囲は計算が複雑なので、ユーザ・マニュアルを参照してください。MIN、MAX コマンドを使用して設定範囲の最小値と最大値を問い合せることもできます。

*RST で 0 に戻ります。

使用例： メインフレーム 2、スロット A、チャンネル 1 のデータ出力のリード・ディレイを 1ps に設定します。

```
PGENA2:CH1:LDELay 1ps
```

現在の設定できる最大のデータ出力のリード・ディレイを問い合せます。

```
PGENA2:CH1:LDELay? MAX
```

PGEN<x>[<m>]:CH<n>:LHOLd(?)

指定されたチャンネルのデータ出力のリーディング・エッジ (Leading Edge) のホールド方法を指定します。

構　文：
```
PGEN<x>[<m>]:CH<n>:LHOLd {LDELay | PHASe}
PGEN<x>[<m>]:CH<n>:LHOLd?
```

応　答：
```
<lead hold>
```

引　数：
```
<lead hold> ::= {LDELay | PHASe}
```
LDELay ：リード・ディレイ (Lead Delay)
PHASe ：位相 (Phase)

注： Phase=Lead Delay / Period * 100 (%)

*RST で LDELay に戻ります。

使用例： メインフレーム 2、スロット A、チャンネル 1 のデータ出力のリーディング・エッジのホールド方法を位相にします。

```
PGENA2:CH1:LHOLd PHAse
```

PGEN<x>[<m>]:CH<n>:LIMit(?)

指定されたチャンネルのリミット (Limit) が適用されるかどうかを設定します。

構　文： PGEN<x>[<m>]:CH<n>:LIMit <boolean>
PGEN<x>[<m>]:CH<n>:LIMit?

応　答： <NR1>

引　数： OFF または <NRf> = 0 ― リミットをオフにします。
ON または <NRf> ≠ 0 ― リミットをオンにします。

*RST で 0 に戻ります。

使用例： メインフレーム 2、スロット A、チャンネル 1 にリミットを適用させます。

```
PGENA2:CH1:LIMit ON
```

PGEN<x>[<m>]:CH<n>:LLIMit(?)

指定されたチャンネルのデータ出力レベルのロー・リミット (Low Limit) を設定します。

構　文： PGEN<x>[<m>]:CH<n>:LLIMit <Numeric>
PGEN<x>[<m>]:CH<n>:LLIMit?

応　答： <NR3>

引　数： ステップは 5mV。
設定範囲は計算が複雑なので、ユーザ・マニュアルを参照してください。MIN、MAX コマンドを使用して設定範囲の最小値と最大値を問い合せることもできます。

*RST で 0.0 に戻ります。

使用例： メインフレーム 1、スロット A、チャンネル 1 のデータ出力レベルのロー・リミットを設定できる最大値に設定します。

```
PGENA:CH1:LLIMit MAX
```

PGEN<x>[<m>]:CH<n>:LOW(?)

指定されたチャンネルのデータ出力のロー・レベル (Low Level) を指定します。

構　文： PGEN<x>[<m>]:CH<n>:LOW <Numeric>
PGEN<x>[<m>]:CH<n>:LOW?

応　答： <NR3>

引　数： ステップは 5mV。
設定範囲は計算が複雑なので、ユーザ・マニュアルを参照してください。MIN、MAX コマンドを使用して設定範囲の最小値と最大値を問い合せることもできます。

*RST で 0.0 に戻ります。

使用例： メインフレーム 1、スロット A、チャンネル 1 のデータ出力レベルのロー・レベルを設定できる最小値に設定します。

```
PGENA:CH1:LOW MIN
```

PGEN<x>[<m>]:CH<n>:OFFSet(?)

指定されたチャンネルのデータ出力のオフセット・レベルを設定します。

構　文： PGEN<x>[<m>]:CH<n>:OFFSet <Numeric>
PGEN<x>[<m>]:CH<n>:OFFSet?

応　答： <NR3>

引　数： ステップは 5mV。
設定範囲は計算が複雑なので、ユーザ・マニュアルを参照してください。MIN、MAX コマンドを使用して設定範囲の最小値と最大値を問い合せることもできます。

*RST で 0.5 に戻ります。

使用例： メインフレーム 1、スロット A、チャンネル 1 のデータ出力のオフセット・レベルを 0.6V に設定します。

```
PGENA:CH1:OFFSet 0.6
```

現在の設定できる最大のデータ出力のオフセット・レベルを問い合せます。

```
PGENA:CH1:OFFSet? MAX
```

PGEN<x>[<m>]:CH<n>:OUTPut(?)

指定されたチャンネルのデータ出力のオン／オフを設定します。

構　文： PGEN<x>[<m>]:CH<n>:OUTPut <boolean>
PGEN<x>[<m>]:CH<n>:OUTPut?

応　答： <NR1>

引　数： OFF または <NRf> = 0 ― データ出力オフにします。
ON または <NRf> ≠ 0 ― データ出力をオンにします。

*RST で 0 に戻ります。

使用例： メインフレーム 2、スロット A、チャンネル 1 のデータ出力をオンにします。

```
PGENA2:CH1:OUTPut ON
```

PGEN<x>[<m>]:CH<n>:PHASe(?)

指定されたチャンネルのデータ出力の位相 (Phase) を設定します。

構　文： PGEN<x>[<m>]:CH<n>:PHASe <Numeric>
PGEN<x>[<m>]:CH<n>:PHASe?

応　答： <NR3>

引　数： Lead Delay と Phase は、どちらもパルスの Leading Edge の位置を示すものですが、表現方法が違うだけで、「実質的な」設定範囲は同一です。

ステップは 0.1%
設定範囲は計算が複雑なので、ユーザ・マニュアルを参照してください。MIN、MAX コマンドを使用して設定範囲の最小値と最大値を問い合せることもできます。

*RST で 0.0 に戻ります。

PHASe を設定するときは、LHOLd を PHASe にしてください。

使用例： メインフレーム 2、スロット A、チャンネル 1 のデータ出力の位相を 1% に設定します。

```
PGENA2:CH1:PHASe 1
```

現在の設定できる最大のデータ出力の位相を問い合せます。

```
PGENA2:CH1:PHASe? MAX
```

PGEN<x>[<m>]:CH<n>:POLarity(?)

指定されたチャンネルのデータ出力の極性を設定します。

構　文： PGEN<x>[<m>]:CH<n>:POLarity <output polarity>
PGEN<x>[<m>]:CH<n>:POLarity?

応　答： <output polarity>

引　数： <polarity> ::= { NORMal | INVert }
NORMal：極性を正にします。
INVert：極性を負にします。

*RST で NORMal に戻ります。

使用例： メインフレーム 2、スロット A、チャンネル 1 のデータ出力の極性を負にします。

```
PGENA2:CH1:POLarity INVert
```

PGEN<x>[<m>]:CH<n>:PRATe(?)

指定されたチャンネルのパルス・レート (Pulse Rate) を設定します。

構　文： PGEN<x>[<m>]:CH<n>:PRATe <pulse rate>
PGEN<x>[<m>]:CH<n>:PRATe?

応　答： <pulse rate>

引　数： <pulse rate> ::= { NORMal | HALF | QUARter | EIGHth | SIXTeenth | OFF }
NORMal　：パルス・レートをノーマルにします。
HALF　：パルス・レートを 1/2 にします。
QUARter　：パルス・レートを 1/4 にします。
EIGHth　：パルス・レートを 1/8 にします。
SIXTeenth　：パルス・レートを 1/16 にします。
OFF　：パルス・レートをオフにします。

*RST で NORMal に戻ります。

使用例： メインフレーム 2、スロット A、チャンネル 1 のパルス・レートを 1/2 にします。

```
PGENA2:CH1:PRATe HALF
```

PGEN<x>[<m>]:CH<n>:SLEW(?)

指定されたチャンネルのデータ出力のスルー・レート (Slew Rate) を設定します。

構　文：
```
PGEN<x>[<m>]:CH<n>:SLEW <Numeric>
PGEN<x>[<m>]:CH<n>:SLEW?
```

応　答：
```
<NR3>
```

引　数： ステップ：0.1V/ns。
単位は V/ns です。
設定範囲は計算が複雑なので、ユーザ・マニュアルを参照してください。MIN、MAX コマンドを使用して設定範囲の最小値と最大値を問い合せることもできます。

*RST で 2.25V/ns に戻ります。

使用例： メインフレーム 2、スロット A、チャンネル 1 のスルー・レートを 5.1V/ns に設定します。

```
PGENA2:CH1:SLEW 5.1
```

PGEN<x>[<m>]:CH<n>:TDELay(?)

指定されたチャンネルのデータ出力のトレイル・ディレイ (Trail Delay) を設定します。

構　文：
```
PGEN<x>[<m>]:CH<n>:TDELay <Numeric>
PGEN<x>[<m>]:CH<n>:TDELay?
```

応　答：
```
<NR3>
```

引　数： ステップは 5ps
設定範囲は計算が複雑なので、ユーザ・マニュアルを参照してください。MIN、MAX コマンドを使用して設定範囲の最小値と最大値を問い合せることもできます。

*RST で 5e-9 に戻ります。

TDELay を設定するときは、THOLd を TDELay にしてください。

使用例： メインフレーム 1、スロット A、チャンネル 1 のデータ出力のトレイル・ディレイを 0.5ns に設定します。

```
PGENA:CH1:TDELay 0.5ns
```

現在の設定できる最大のデータ出力のトレイル・ディレイを問い合せます。

```
PGENA:CH1:TDELay? MAX
```

PGEN<x>[<m>]:CH<n>:THOLd(?)

指定されたチャンネルのデータ出力のトレーリング・エッジ (Trailing Edge) のホールド方法を指定します。

構　文： PGEN<x>[<m>]:CH<n>:THOLd <trail hold>
PGEN<x>[<m>]:CH<n>:THOLd?

応　答： <trail hold>

引　数： <trail hold> ::= {TDELay | DCYCle | WIDTh}
TDELay ：トレーリング・エッジのホールド方法を TDELay にします。
DCYCle ：トレーリング・エッジのホールド方法を DCYCle にします。
WIDTh ：トレーリング・エッジのホールド方法を WIDTh にします。

*RST で DCYCle に戻ります。

使用例： メインフレーム 1、スロット A、チャンネル 1 のトレーリング・エッジのホールド方法を TDELay にします。

```
PGENA:CH1:THOLd TDELay
```

PGEN<x>[<m>]:CH<n>:TIMPedance(?)

指定されたチャンネルのデータ出力の終端インピーダンス (Termination Impedance) を設定します。

構　文： PGEN<x>[<m>]:CH<n>:TIMPedance <Numeric>
PGEN<x>[<m>]:CH<n>:TIMPedance?

応　答： <NR3>

引　数： 設定範囲：10ohm ～ 1Mohm、
0 以下の場合：オープン
オープン時、応答は –1 を返します。
ステップ：有効数字は 3 桁、最小分解能は 1ohm

*RST で 50ohm に戻ります。

使用例： メインフレーム 1、スロット A、チャンネル 1 の終端インピーダンスをオープンに設定します。

```
PGENA:CH1:TIMPedance -1
```

PGEN<x>[<m>]:CH<n>:TVOLtage(?)

指定されたチャンネルのデータ出力の終端電圧 (Termination Voltage) を設定します。

構　文： PGEN<x>[<m>]:CH<n>:TVOLtage <Numeric>
PGEN<x>[<m>]:CH<n>:TVOLtage?

応　答： <NR3>

引　数： 設定範囲：-2V ～ +5V
ステップ：0.1V

*RST で 0V に戻ります。

使用例： メインフレーム 1、スロット A、チャンネル 1 の終端電圧を 1V に設定します

```
PGENA:CH1:TVOLtage 1
```

PGEN<x>[<m>]:CH<n>:TYPE(?)

DG モードで指定されたチャンネルのデータ出力のフォーマット (Format) を設定します。

構　文： PGEN<x>[<m>]:CH<n>:TYPE <data format>
PGEN<x>[<m>]:CH<n>:TYPE?

応　答： <data format>

引　数： <data format> ::= {NRZ | RZ | R1}
NRZ：信号を「NRZ 形式」に設定します。
RZ ：信号を「RZ 形式」に設定します。
R1 ：信号を「R1 形式」に設定します。

*RST で NRZ に戻ります。

使用例： DG モードでメインフレーム 1、スロット A、チャンネル 1 のフォーマットを「R1 形式」に設定します。

```
PGENA:CH1:TYPE R1
```

PGEN<x>[<m>]:CH<n>:WIDTh(?)

指定されたチャンネルのデータ出力のパルス幅 (Pulse Width) を設定します。

構　文： PGEN<x>[<m>]:CH<n>:WIDTh <Numeric>
PGEN<x>[<m>]:CH<n>:WIDTh?

応　答： <NR3>

引　数： ステップ：5ps
設定範囲：
Trail Delay あるいは Duty の設定範囲から以下の換算式で求められます。
Pulse Width = Duty * (Period * Pulse Rate) / 100
あるいは
Pulse Width = Trail Delay – Lead Delay
MIN、MAX コマンドを使用して設定範囲の最小値と最大値を問い合せることもできます。

*RST で 5e-9 に戻ります。

WIDTh を設定するときは、THOLd を WIDTh にしてください。

使用例： メインフレーム 1、スロット A、チャンネル 1 のパルス幅を 6ns に設定します。

```
PGENA:CH1:WIDTh 6e-9
```

PGEN<x>[<m>]:ID? （問い合せのみ）

指定されたスロットにどのようなモジュールが入っているか調べます。

構　文： PGEN<x>[<m>]:ID?

応　答： <opt>

引　数： <opt>
-1：入っていない
1：DTGM10
2：DTGM20
3：DTGM30
4：DTGM21
5：DTGM31
6：DTGM32

使用例： メインフレーム 1、スロット B に入っているモジュールを調べます。

```
PGENB:ID?
```

モジュールが入っていない場合、次の応答が返ります。

```
-1
```

*RST （問い合せなし）

機器をデフォルト設定に戻します。（ただし、GPIB アドレス等コマンドで変更できないものは初期化されません。）

構　文： *RST

使用例： 機器をリセットします。

```
*RST
```

SEQuence:DATA(?)

シーケンスの一行分の設定をします。

構　文：
```
SEQuence:DATA <line number>, <label>, <wait trigger>, <block/subsequence name>, <repeat count>, <jump to>, <go to>
SEQuence:DATA? <line number>
```

応　答：
```
<label>, <wait trigger>, <block/subsequence name>, <repeat count>, <jump to>, <go to>
```

引　数：
```
<line number> ::= <NR1> ― 0 から始まります。
<label> ::= <string> ― 16 文字以内。
<wait trigger> ::= { ON | OFF | <NRf> }
<block/subsequence name> ::= <string> ― 32 文字以内。
<repeat count> ::= <NR1> ― 1 から 65536 まで。
                           0 の場合は無限ループ。
<jump to> ::= <string> ― この行を出している途中でイベントが発生したときの飛び先。
<go to> ::= <string> ― この行を出し終わった後の無条件の飛び先。
```

使用例： ライン・ナンバ 0、ラベル ""、ウェイト・トリガ OFF、ブロック名 ""、リピート回数 1、ジャンプ先 "", goto “Label2” の場合、次のようになります。

```
SEQuence:DATA 0, "", OFF, "", 1, "", "Label2"
```

*RST 後の応答は "", 0, "Block1", 0, "", "" となります。

SEQuence:LENGth(?)

シーケンスの長さを設定します。

構　文：
```
SEQuence:LENGth <NR1>
SEQuence:LENGth?
```

応　答： <NR1>

引　数： 設定範囲：1 ～ 8000。
長さが増えた場合、その部分の内容（データ定義）は不定です。

*RST で 1 に戻ります。

使用例： シーケンスの長さ（行数）を 1000 に設定します。

```
SEQuence:LENGth 1000
```

もとの長さが 1000 より小さな場合は、新規部分のデータは不定です。

SIGNal:ASSign(?)

論理チャンネルに物理チャンネルをアサインします。

構　文：
```
SIGNal:ASSign <logical channel>, <physical channel>
SIGNal:ASSign? <logical channel>
```

応　答： <physical channel>

引　数： <logical channel> の中身は、
<group name> 1 ビット幅のグループの場合
<group name>[<bit>] 指定されたグループ中の指定されたビット番号（この場合の [] は省略できません。）

例：
CLK
Addr[0]

<physical channel>
メインフレーム番号、スロット名、CH 番号で指定します。
"1A4" はメインフレーム 1、スロット A、チャンネル 4。
"" の場合はアサインが解除されます。

*RST で自動アサインされます。

使用例： メインフレーム 1、スロット B、チャンネル 4 の物理チャンネルをグループ名 Addr、ビット番号 1 の論理チャンネルに割りあてます。

```
SIGNal:ASSign "Addr[1]","1B4"
```

SIGNal:<parameter>(?)

信号名を使って、データ出力の各種パラメータを設定します。

構　文： SIGNal:<parameter> <signal>, <value>
SIGNal:<parameter>? <signal>

応　答： <value>

引　数： <parameter> ::= {AMODe | AMPLitude | CPOint | DCYCle | DTOFfset | DTOFfset:STATe | HIGH | HLIMit | LDELay | LHOLd | LIMit | LLIMit | LOW | OFFSet | OUTPut | PHASe | POLarity | PRATe | SLEW | TDELay | THOLd | TIMPedance | TVOLtage | TYPE | WIDTh}

<signal>:= 論理チャンネル、またはバス。
Addr[]
Addr[0:3]
Addr[0..3]
Addr[3..0]
のように書きます。

注： Addr[] の様に、[] の中身を省略した場合は、Addr[<msb>:<lsb>] として処理されます。（例えば Addr が 8 ビット幅なら Addr[] は Addr[7:0] とみなされます。）

<value> は <parameter> によって変わります。
具体的には PGEN<x>[<m>]:CH<n> コマンドを参照してください。

複数のチャンネルに対して問い合わせが行なわれた場合には、最初のチャンネルの値が返ります。例えば、SIGN:HIGH? "DATA[2..4]" では DATA[2] の値が返ります。

使用例： SIGNal: AMPLitude "Addr[1]",1.1

Addr[1] で指定されたチャンネルの振幅を 1.1V にします。

SIGNal:TYPE "Addr[2]",R1

Addr[2] で指定されたチャンネルのデータ出力のフォーマットを R1 にします。

SIGNal:BDATa(?)

パターン・データをバイナリで転送します。

構　文： SIGNal:BDATa <logical channel>, <start vector>, <vector size>, <binary pattern data>
SIGNal:BDATa? <logical channel>, <start vector>, <vector size>

応　答： <binary pattern data>

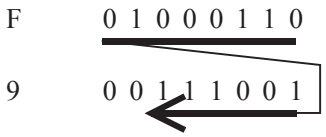
引　数： <logical channel> ::= SIGNal:ASSign でアサインした論理チャンネル
<start vector> ::= データのスタート・アドレス
<vector size> ::= データのサイズ
<binary pattern data> ::= バイナリ・バイトのブロック

注：一回で転送できるパターン・データの量は最大 1MB です。

使用例： SIGNal:BDATa "Addr[1]",0,14,#12F9

このデータは
#：ブロックのスタート・キャラクタ
1：長さのフィールドの長さが「1」であることを示します。
2：データの長さが「2」であることを示します。
F：01000110
9：00111001

ですので、Addr[1] で指定されたチャンネルの頭から 14 ベクタ分のデータがそれぞれ、0,1,1,0,0,0,1,0,1,0,0,1,1,1 に設定されます。



SIGNal:BDATa "Addr[1]",2,10

このコマンドで、Addr[1] で指定されたチャンネルのアドレス 2 から 10 ベクタ分のデータを読み取ることができます。

SIGNal:DATA(?)

パターン・データを転送します。

構　文： SIGNal:DATA <logical channel>, <start vector>, <vector size>, <ascii pattern data>
SIGNal:DATA? <logical channel>, <start vector>, <vector size>

応　答： <ascii pattern data>

引　数： <logical channel> ::= SIGNal:ASSign でアサインした論理チャンネル
<start vector> ::= データのスタート・アドレス
<vector size> ::= データのサイズ
<ascii pattern data> ::= データ文字列

注： 一回で転送できるパターン・データの量は最大 1MB です。

使用例：

```
SIGNal:DATA "Test[2]",0,16, "0100011100111001"
```

というコマンドは Test[2] チャンネルのアドレス 0 から 16 ベクタ分データをそれぞれ 0,1,0,0,0,1,1,1,0,0,1,1,1,0,0,1 に設定します。

```
SIGNal:DATA "Test[2]",2,10
```

というコマンドは Test[2] チャンネルのアドレス 2 から 10 ベクタ分のデータを読み取ることができます。

SIGNal:IMPedance?（問い合せのみ）

DTGM21 型の物理チャンネルにアサインされている論理チャンネルの出力インピーダンスを調べます。

構　文： SIGNal:IMPedance? <signal>

応　答： <NR3>
5 e+1 または 2.3 e+1 が返ります。

引　数： <signal>::=<string> ― SIGNal:ASSign コマンドでアサインした論理チャンネル名を指定します。

使用例： 論理チャンネル名 Group1[0] の出力インピーダンスを調べます。

```
SIGNAL:IMPedance? "Group1[0]"
```

インピーダンスが 50 ohm に設定されている場合、5 e+1 が返ります。

SIGNal:JRANge(?)

DTGM32 型の物理チャンネルにアサインされている論理チャンネルのジッタ・レンジを設定します。

構　文： SIGNal:JRANge <signal>,<Numeric>
SIGNal:JRANge? <signal>

応　答： <NR3>

引　数： <signal>::=<string> ― SIGNal:ASSign コマンドでアサインした論理チャンネル名を指定します。
<Numeric>::=<NR3>
設定範囲：1e-9 s または 2e-9 s

*RST で 2e-9 s に戻ります。

使用例： 論理チャンネル名 Group1[0] のジッタ・レンジを 1 ns に設定します。

```
SIGNal:JRANge "Group1[0]",1ns
```

*SRE(?)

Service Request Enable Register (SRER) の設定を行います。

構　文： *SRE <NR1>
*SRE?

応　答： <NR1>

引　数： <NR1> ― SRER のビット値。範囲：0 ～ 255。SRER のバイナリ・ビットは、この値によってセットされ、範囲外の値を代入すると実行エラーが発生します。

*RST でも値は保持されます。

使用例： SRER のビットを、2 進数の 00110000 にセットします。

```
*SRE 48
```

次は、問い合せの例です。

```
*SRE?
```

SRER のビットが 2 進数の 00100000 にセットされていると、値 32 が返されます。

*STB? （問い合せのみ）

Status Byte Register (SBR) の値を問い合せます。

構　文： *STB?

応　答： <NR1> ― SBR の 2 進数の値が 10 進数で返されます。

使用例： *STB? に対する応答例です。

```
96
```

この場合、SBR の内容は 2 進数で 0110 0000 です。

SUBSequence:DATA(?)

サブシーケンス一行分の設定をします。

構　文：
```
SUBSequence:DATA <line number>, <block name>, <repeat count>
SUBSequence:DATA? <line number>
```

応　答： <block name>, <repeat count>

引　数：
<line number> ::= <NR1> ― 0 から始まります。
<block name> ::= <string> ― 32 文字以内。
<repeat count> ::= <NR1> ― 1 から 65536 まで。

使用例： 0 行目にブロック名 "Block0"、リピート・カウント 1 のサブシーケンスを設定します。

```
SUBSequence:DATA 0, "Block0",1
```

SUBSequence:DELete （問い合せなし）

サブシーケンスを削除します。

構　文： SUBSequence:DELete <subsequence name>

引　数： <subsequence name> ::= <string>

使用例： "Sub1" と言うサブシーケンスを削除します。

```
SUBSequence:DELete "Sub1"
```

SUBSequence:DELete:ALL （問い合せなし）

全てのサブシーケンスを削除します。

構　文： SUBSequence:DELete:ALL

使用例： 全てのサブシーケンスを削除します。

```
SUBSequence:DELete:ALL
```

SUBSequence:LENGth(?)

サブシーケンスの長さを設定します。

構　文：
```
SUBSequence:LENGth <subsequence name>, <length>
SUBSequence:LENGth? <subsequence name>
```

応　答： <length>

引　数： <subsequence name> ::= <string>
<length> ::= <NR1> ― 設定範囲は 1 ～ 256。指定されたサブシーケンスが存在しないと、-1 が返されます。
長さが増えた場合、その部分の内容（データ定義）は不定です。

*RST で -1 に戻ります。

使用例： サブシーケンス “Sub1” の長さを 128 に設定します。

```
SUBSequence:LENGth "Sub1",128
```

SUBSequence:NEW （問い合せなし）

サブシーケンスを作成します。

構　文： SUBSequence:NEW <subsequence name>, <length>

引　数： <subsequence name> ::= <string> ― 32 文字以内。
<length> ::= <NR1> ― 設定範囲は 1 ～ 256。

データ定義（内容）は不定です。

使用例： "Sub02" と言う名前のサブシーケンスを長さ 100 で作成します。

```
SUBSequence:NEW "Sub02",100
```

SUBSequence:SELect(?)

SUBSequence:DATA で設定するサブシーケンスを選択します。

構　文： SUBSequence:SELect <subsequence name>
SUBSequence:SELect?

応　答： <subsequence name>

引　数： <subsequence name> ::= <string>

*RST で "" に戻ります。

使用例： SUBSequence:DATA で使うサブシーケンスを "Sub03" に設定します。

```
SUBSequence:SELect "Sub03"
```

SYSTem:ERRor[:NEXT]? （問い合せのみ）

エラー / イベント・キューから一つエラー / イベントを取り出し、返します。

構　文： SYSTem:ERRor[:NEXT]?

応　答： <Error/event number>, "<Error/event description>"

<Error/event number> ― エラー / イベント・コード：-32768 ～ 0 の整数値。
0　　：エラー / イベントは発生していません。
負の値：SCPI で定められたエラー / イベント・コード。

<Error/event description> ― エラー / イベントの内容。

使用例： エラーを取り出します。

```
SYSTem:ERRor[:NEXT]?
```

エラーがない場合、次の応答が返ります。

```
0
```

SYSTem:KLOCk(?)

前面パネルをロック、またはロックを解除します。本機器を外部コントローラから制御するために前面パネルの操作を無効にする場合、このコマンドを使います。

構　文： SYSTem:KLOCk <boolean>
SYSTem:KLOCk?

応　答： <NR1>

引　数： OFF または <NRf> = 0 — 前面パネルのロックを解除します。
ON または <NRf> ≠ 0 — 前面パネルをロックします。

*RST で 0 に戻ります。

使用例： 前面パネルをロックします。

```
SYSTem:KLOCk ON
```

SYSTem:VERSion? （問い合せのみ）

本機器が準拠している SCPI のバージョンを問い合せます。

構　文： SYSTem:VERSion?

応　答： <NR2> ::= 1999.0

使用例： SYSTem:VERSion? の応答例です。

```
1999.0
```

TBAS:COUNt(?)

バースト・カウント (Burst Count) を設定します。

構　文：
```
TBAS:COUNt <Numeric>
TBAS:COUNt?
```

応　答：
```
<NR1>
```

引　数： 設定範囲：1 ～ 65536。

*RST で 1 に戻ります。

使用例： バースト・カウントを 10 に設定します。

```
TBAS:COUNt 10
```

TBAS:CRANge(?)

クロック・レンジ（Clock Range）を設定します

構　文：
```
TBAS:CRANge <NR1>
TBAS:CRANge?
```

応　答：
```
<NR1>
```

引　数：
0：　50k　～　100kHz
1：　100k　～　200kHz
2：　200k　～　400kHz
3：　250k　～　500kHz
4：　500k　～　1MHz
5：　1M　～　2MHz
6：　2M　～　4MHz
7：　2.5M　～　5MHz
8：　5M　～　10MHz
9：　10M　～　20MHz
10：　20M　～　40MHz
11：　25M　～　50MHz
12：　50M　～　100MHz
13：100M　～　200MHz
14：200M　～　400MHz（RZ/R1 があるときには 200MHz <）
15：400M <

*RST で 12 に戻ります。

使用例： クロック・レンジを "25M ～ 50MHz" にします。

```
TBAS:CRANge 11
```

クロック・レンジを問い合せます。

```
TBAS:CRANge?
```

応答例です。

```
11
```

TBAS:DOFFset(?)

ディレイ・オフセット (Delay Offset) を設定します。

構　文：
```
TBAS:DOFFset <Numeric>
TBAS:DOFFset?
```

応　答： <NR3>

引　数： ステップ：
DTG5078：1ps
DTG5274：0.2ps

設定範囲は計算が複雑なので、ユーザ・マニュアルを参照してください。MIN、MAX コマンドを使用して設定範囲の最小値と最大値を問い合せることもできます。

*RST で 0 に戻ります。

使用例： ディレイ・オフセットを 1ps に設定します。

```
TBAS:DOFFset 1ps
```

TBAS:EIN:IMMediate （問い合せなし）

イベントを発生させます。

構　文：
```
TBAS:EIN:IMMediate
```

使用例： イベントを発生させます。

```
TBAS:EIN:IMMediate
```

TBAS:EIN:IMPedance(?)

イベント入力インピーダンスを設定します。

構　文：
```
TBAS:EIN:IMPedance <Numeric>
TBAS:EIN:IMPedance?
```

応　答：
```
<NR3>
```

引　数： 設定範囲：50 または 1e3

*RST で 1e3 に戻ります。

使用例： イベント入力インピーダンスを 50ohm に設定します

```
TBAS:EIN:IMPedance 50
```

TBAS:EIN:LEVel(?)

イベント入力レベルを設定します。

構　文：
```
TBAS:EIN:LEVel <Numeric>
TBAS:EIN:LEVel?
```

応　答：
```
<NR3>
```

引　数： 設定範囲：-5V ～ +5V
ステップ：0.1V

*RST で 1.4V に戻ります。

使用例： イベント入力レベルを 1.1V に設定します。

```
TBAS:EIN:LEVel 1.1
```

TBAS:EIN:POLarity (?)

イベント入力極性を設定します。

構　文：
```
TBAS:EIN:POLarity <input slope>
TBAS:EIN:POLarity?
```

応　答：
```
<input slope>
```

引　数： <input slope> ::= {NORMal | INVert}
NORMal：イベントの入力極性をポジティブにします。
INVert：イベントの入力極性をネガティブにします。

*RST で NORMal に戻ります。

使用例： イベントの入力極性をネガティブにします。

```
TBAS:EIN:POLarity INVert
```

TBAS:FREQuency(?)

周波数を設定します。

構　文：
```
TBAS:FREQuency <Numeric>
TBAS:FREQuency?
```

応　答： <NR3>

引　数： 設定範囲：
DG モード（NRZ のみ）
DTG5078：50 kbps ～ 750 Mbps
DTG5274：50 kbps ～ 2.7 Gbps
DTG5334：50 kbps ～ 3.35 Gbps
DG モード（RZ/R1 使用）
DTG5078：50 kbps ～ 375 Mbps
DTG5274：50 kbps ～ 1.35 Gbps
DTG5334：50 kbps ～ 1.67 Gbps
PG モード
DTG5078：50 kHz ～ 375 MHz
DTG5274：50 kHz ～ 1.35 GHz
DTG5334：50 kHz ～ 1.675 GHz

周波数及び周期の分解能は 8 桁です。
ただし、Clock Source が External Clock Input と External PLL Input の時には、4 桁になります。

*RST で 100e6（100MHz）に戻ります。

使用例： 周波数を 200MHz に設定します

```
TBAS:FREQuency 200MHz
```

TBAS:JMODe(?)

ジャンプ・モード (Jump Mode) を設定します。

構　文： TBAS:JMODe {COMMand | EVENt}
TBAS:JMODe?

応　答： {COMMand | EVENt}

引　数： COMMand：ジャンプ・モードをコマンドにします。
EVENt　：ジャンプ・モードをイベントにします。

*RST で EVENt に戻ります。

使用例： ジャンプ・モードをコマンドに設定します

```
TBAS:JMODe COMMand
```

TBAS:JTIMing(?)

ジャンプ・タイミング (Jump Timing) を設定します。

構　文： TBAS:JTIMing {ASYNc | SYNC}
TBAS:JTIMing?

応　答： {ASYNc | SYNC}

引　数： ASYNc　：ジャンプ・タイミングを非同期モードにします。
SYNC　：ジャンプ・タイミングを同期モードにします。

*RST で SYNC に戻ります。

使用例： ジャンプ・タイミングを非同期モードにします。

```
TBAS:JTIMing ASYNc
```

TBAS:JUMP （問い合せなし）

ソフトウェア・ジャンプを行います。

構　文： TBAS:JUMP <string>

引　数： <string> は Main Sequence 内のラベルです。

使用例： シーケンス内のラベル ”test1” へジャンプします。

```
TBAS:JUMP "test1"
```

TBAS:LDELay(?)

ロング・ディレイ (Long Delay) の設定をします。
ロング・ディレイがオンの時はコマンド・ジャンプもイベント・ジャンプも使用できません。

構　文：
```
TBAS:LDELay <boolean>
TBAS:LDELay?
```

応　答： <NR1>

引　数： OFF または <NRf> ＝ 0 ― ロング・ディレイをオフにします。
ON または <NRf> ≠ 0 ― ロング・ディレイをオンにします。

*RST で 0 に戻ります。

使用例： ロング・ディレイをオンにします。

```
TBAS:LDELay ON
```

TBAS:MODE(?)

PG のラン・モード (Run Mode) を設定します。

構　文：
```
TBAS:MODE {BURSt | CONTinuous}
TBAS:MODE?
```

応　答： {BURSt | CONTinuous}

引　数： BURSt：バースト・モードに設定します。（トリガを待って指定した回数分パルスを出力します。）
CONTinuous：コンティニュアス・モードに設定します。（トリガを待たずにただ連続的にパルスを出力します）

*RST で CONTinuous に戻ります。

使用例： PG のラン・モードを BURSt に設定します

```
TBAS:MODE BURSt
```

TBAS:OMODe(?)

動作モード (Operating Mode) を設定します。

構　文：
```
TBAS:OMODe { DATA | PULSe }
TBAS:OMODe?
```

応　答：
```
{ DATA | PULSe }
```

引　数：
```
DATA    ：Data Generator モードに設定します。
PULSe   ：Pulse Generator モードに設定します。
```

*RST で DATA に戻ります。

使用例： PG モードに設定します。

```
TBAS:OMODe PULSe
```

TBAS:PERiod(?)

周期を設定します。

構　文：
```
TBAS:PERiod <Numeric>
TBAS:PERiod?
```

応　答：
```
<NR3>
```

引　数： PERiod ::= 1/FREQuency です。
設定範囲等は FREQuency に準拠します。
MIN、MAX コマンドを使用して設定範囲の最小値と最大値を問い合せることもできます。

*RST で 10e-9s に戻ります。

使用例： 周期を 2ns に設定します。

```
TBAS:PERiod 2ns
```

現在の設定できる最小の周期を問い合せます。

```
TBAS:PERiod? MIN
```

TBAS:PRATe? （問い合せのみ）

PLL Multiplier Rate を問い合せます。

構　文： TBAS:PRATe?

応　答： <NR1>

引　数： <NR1>：PLL Multiplier Rate

使用例： PLL Multiplier Rate を問い合せます

```
TBAS:PRATe?
```

次のような応答が返ります。

```
1000
```

TBAS:RSTate? （問い合せのみ）

シーケンサ・ステータスを問い合せます。

構　文： TBAS:RSTate?

応　答： { RUN | STOP | WAIT| PUNLocked | ERMissing | EPMissing | ECMissing }

引　数： RUN ― 実行中です。
STOP ― 停止中です。
WAIT ― ウェイト中です。
PUNLocked ― PLL はアンロックされています。
ERMissing ― 外部 10M リファレンス・クロックが見つかりません。
EPMissing ― 外部 PLL 入力が見つかりません。
ECMissing ― 外部クロックが見つかりません。

使用例： シーケンサ・ステータスを問い合せます。

```
TBAS:RSTate?
```

応答例です。実行中は次の応答が返ります。

```
RUN
```

TBAS:RUN(?)

シーケンサをスタート、停止させます。

構　文： TBAS:RUN <boolean>
TBAS:RUN?

応　答： <NR1>

引　数： OFF または <NRf> = 0 — シーケンサをストップします。
ON または <NRf> ≠ 0 — シーケンサをスタートします。

使用例： シーケンサをスタートさせます。

```
TBAS:RUN ON
```

TBAS:SMODe(?)

シーケンサ・モード (Sequencer Mode) を設定します。

構　文： TBAS:SMODe {HARDware | SOFTware}
TBAS:SMODe?

応　答： {HARDware | SOFTware}

引　数： HARDware：シーケンサ・モードをハードウェアに設定します。
SOFTware：シーケンサ・モードをソフトウェアに設定します。

*RST で HARDware に戻ります。

使用例： シーケンサ・モードをソフトウェアに設定します

```
TBAS:SMODe SOFTware
```

TBAS:SOURce(?)

クロック・ソース (Clock Source) を設定します。
クロック・ソースが変更されると、クロックは自動再スタートせず、停止します。

構　文： TBAS:SOURce <clock source>
TBAS:SOURce?

応　答： <clock source>

引　数： <clock source> ::= {INTernal | EXTReference | EXTPll | EXTernal }
INTernal ― 内部 10M (Internal 10MHz Reference)
EXTReference ― 外 部 10M (External 10MHz Reference)
EXTPll ― 外部 PLL (External PLL Input)
EXTernal ― 外部クロック (External Clock Input)

*RST で INTernal に戻ります。

使用例： クロック・ソースを外部 PLL に設定します

```
TBAS:SOURce EXTPll
```

TBAS:TIN:IMPedance(?)

トリガ入力インピーダンスを設定します。

構　文：
```
TBAS:TIN:IMPedance <Numeric>
TBAS:TIN:IMPedance?
```

応　答： <NR3>

引　数： 設定範囲：50 と 1e3

*RST で 1e3 に戻ります。

使用例： トリガ入力インピーダンスを 50ohm に設定します。

```
TBAS:TIN:IMPedance 50
```

TBAS:TIN:LEVel(?)

トリガ入力レベルを設定します。

構　文：
```
TBAS:TIN:LEVel <Numeric>
TBAS:TIN:LEVel?
```

応　答： <NR3>

引　数： 設定範囲：-5V ～ +5V
ステップ：0.1V

*RST で 1.4V に戻ります。

使用例： トリガ入力レベルを 1V に設定します

```
TBAS:TIN:LEVel 1
```

TBAS:TIN:SLOPe(?)

トリガ入力極性を設定します。

構　文： TBAS:TIN:SLOPe {POSitive | NEGative}
TBAS:TIN:SLOPe?

応　答： {POSitive | NEGative}

引　数： POSitive：ポジティブ
NEGative：ネガティブ

*RST で POSitive に戻ります。

使用例： トリガ入力極性をネガティブ設定します

```
TBAS:TIN:SLOPe NEGative
```

TBAS:TIN:SOURce(?)

トリガ入力ソースを設定します。

構　文： TBAS:TIN:SOURce {INTernal | EXTernal}
TBAS:TIN:SOURce?

応　答： {INTernal | EXTernal}

引　数： INTernal：内部トリガ
EXTernal：外部入力

*RST で EXTernal に戻ります。

使用例： トリガ入力ソースを内部に設定します

```
TBAS:TIN:SOURce INTernal
```

TBAS:TIN:TIMer(?)

内部トリガ周期を設定します。

構　文： TBAS:TIN:TIMer <Numeric>
TBAS:TIN:TIMer?

応　答： <NR3>

引　数： 設定範囲：1.0 μ s ～ 10.0s
ステップ：0.1 μ s（有効数字 3 桁）

*RST で 1e-3 に戻ります。

使用例： 内部トリガ周期を 1.01 秒に設定します。

```
TBAS:TIN:TIMer 1.01
```

TBAS:TIN:TRIGger （問い合せなし）

トリガを発生させます。
*TRG と同じ動作です。

構　文： TBAS:TIN:TRIGger

使用例： トリガ信号を発生させます。

```
TBAS:TIN:TRIGger
```

TBAS:VRATe? （問い合せのみ）

ベクタ・レイト (Vector Rate) を問い合せます。

構　文： TBAS:VRATe?

応　答： <NR1>

引　数： <NR1> ::= ベクタ・レイト (HW のクロック周波数とユーザ周波数との比) を応答します。

使用例： ベクタ・レイトを問い合せます

```
TBAS:VRATe?
```

次のように応答が返ります。

```
8
```

*TRG （問い合せなし）

トリガを発生させます。前面パネル・キーの Force Trigger と同じ動作を行います。

構　文： *TRG

使用例： トリガ信号を発生させます。

```
*TRG
```

*TST? （問い合せのみ）

セルフ・テストを実行し、結果を返します。

構　文： *TST?

応　答： <NR1>

使用例： *TST? の応答例です。エラーがなかったことを示しています。

```
0
```

VECTor:BDATa(?)

バイナリ・フォーマットで、複数チャンネル分のパターン・データを転送します。

構　文：
```
VECTor:BDATa <start vector>, <vector size>, <binary pattern data>
VECTor:BDATa? <start vector>, <vector size>
```

応　答： <binary pattern data>

引　数：
<start vector> ::= <NR1> ― データのスタート・アドレス
<vector size> ::= <NR1> ― データのサイズ
<binary pattern data> ::= <block data> ― バイナリ・バイトのブロック

注： 一回で転送できるパターン・データの量は最大 1MB です。

使用例：

```
VECTor:BIOFormat "G1[2:10]", "G2[1]"

VECTor:BDATa 1,2, #16abCDEF
```

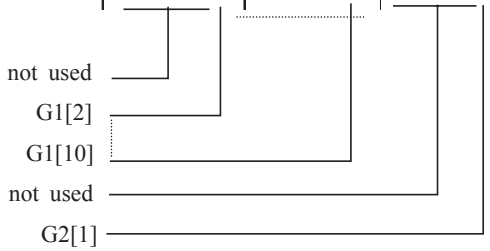
VECTor:BIOFormat コマンドでデータの転送するバスを指定して、VECTor:BDATa で、複数チャンネル分のデータをまとめて転送します。このとき、

- まず各 <signal> 毎に必要なビット数を求めそこから必要なバイト数を求めます。
- そのバイト数分づつデータを取り出します。
- 不要な MSB が捨てられ、残った部分に対し、MSB 側から順に指定された論理チャンネルにデータを書き込まれます。

この場合、データは
#：ブロックのスタート・キャラクタ
1：長さのフィールドの長さが「1」であることを示します。
6：データの長さが「6」であることを示します。
abCDEF：01100001 01100010 01000011 01000100 01000101 01000110

を表すので、以下のようにデータが設定されます。

	ベクタ・アドレス 1			ベクタ・アドレス 2		
グループ	G1		G2	G1		G2
<binary pattern data>	a	b	C	D	E	F
バイナリ値	01100001	01100010	01000011	01000100	01000101	01000110



上の例では、G1[2:10] という指定をしましたので、6 ビット分のデータの中の MSB が G1[2] に、LSB が G1[10] に入りますが、これを G1[10:2] とした場合はその逆になります。

```
VECTor:BDATa? 2,10
```

このコマンドで、指定されたアドレスの、アドレス 2 から 10 ビットのデータを読み取ることができます。

VECTor:BIOFormat(?)

VECTor:BDATa で転送されるデータ項目を設定します。

構　文： VECTor:BIOFormat <signal> [, <signal>...]
VECTor:BIOFormat?

応　答： <signal> [, <signal>...]

引　数： <signal>:= 論理チャンネル、またはバス。
Addr[]
Addr[0:3]
Addr[0..3]
Addr[3..0]
のように書きます。

注： Addr[] の様に、[] の中身を省略した場合は、Addr[<msb>:<lsb>] として処理されます。（例えば Addr が 8 ビット幅なら Addr[] は Addr[7:0] とみなされます。）

複数のチャンネルに対して問い合わせが行なわれた場合には、最初のチャンネルの値が返ります。例えば、SIGN:HIGH? "DATA[2..4]" では DATA[2] の値が返ります。

使用例： BDATa コマンドで転送するデータのバスを ”Addr[1:3]” に指定します。

```
VECTor:BIOFormat "Addr[1:3]"
```

VECTor:DATA(?)

アスキー・フォーマットで、パターン・データを転送します。

構　文： VECTor:DATA <start vector>, <vector size>, <ascii pattern data>
VECTor:DATA? <start vector>, <vector size>

応　答： <ascii pattern data>

引　数： <start vector> ::= <NR1> ― データのスタート・アドレス
<vector size> ::= <NR1> ― データのサイズ
<ascii pattern data> ::= <string> ― データ文字列

注： 一回で転送できるパターン・データの量は最大 1MB です。

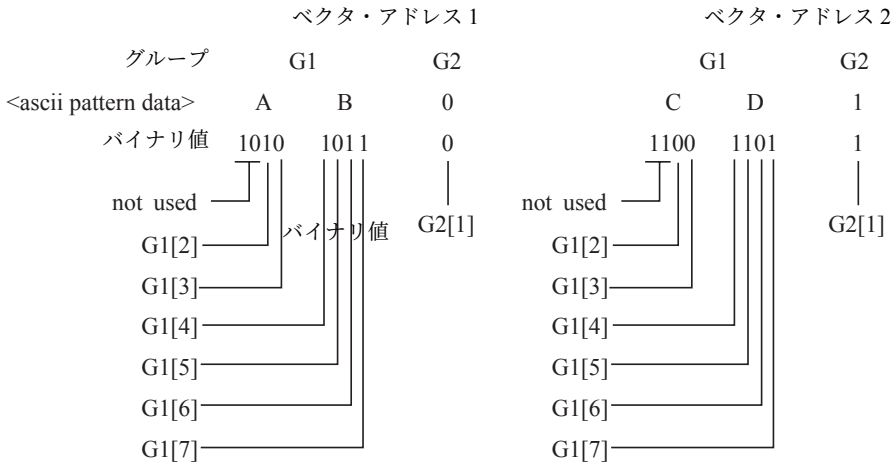
使用例：

```
VECTor:IOFormat "G1[2:7]", HEX, "G2[1]", BIN

VECTor:DATA 1,2, "ABOCD1"
```

パターンデータを受け取る時には以下の処理が行なわれます。

- 送られてきた <ascii pattern data> をその文字数分づつ取り出します。
- 文字からバイナリ値に変換し、その中の不要な MSB が捨てられるます。
- 残った部分に対し、MSB 側から順に指定された論理チャンネルにデータを書き込みます。



Vector	Group G1									Group G2		
	8	7	6	5	4	3	2	1	0	2	1	0
1	X	X	X	X	X	X	X	X	X	X	X	X
2	X	1	1	0	1	0	1	X	X	X	0	X
3	X	1	0	1	1	0	0	X	X	X	1	X
4	X	X	X	X	X	X	X	X	X	X	X	X

上の例では、G1[2:7] という指定をしましたので、6 ビット分のデータの中の MSB が G1[2] に、LSB が G1[7] に入りますが、これを G1[7:2] とした場合はその逆になります

```
VECTor:IOFormat "DT", OCT

VECTor:DATA? 2,6
```

このコマンドで、DT のアドレス 2 から 6 ビットのデータを OCT 形式で読み取ることができます。

VECTor:IMPort （問い合せなし）

ファイルからパターン・データを BLOCk:SELect で選択したブロックに対して読み込みます。

構　文： VECTor:IMPort <format>, <filename>

引　数： <format> ::= {TLA | VCA | VCB | DG }

TLA：Tektronix TLA Data Exchange Format (*.txt)
VCA：HFS Vector Files (*.vca)
VCB：HFS Vector Files (*.vcb)
DG：DG2000 Series Files (*.pda)

<filename> ::= <string> ― 絶対パスのファイル名

DG ファイルが読み込まれた場合、シーケンス、すべてのサブシーケンス、およびすべてのブロックが読み込まれます。なお、読み込まれる前に存在していたシーケンス、すべてのサブシーケンス、およびすべてのブロックは削除されます。また、グループ定義（グループ名およびビット幅）も読み込まれます。なお、チャンネル・アサインは無視されます。

使用例： TLA フォーマットで作成した DTG5000 内の “C:¥tmp¥tla2.txt” と言うファイルを読み込みます。

```
VECTor:IMPort TLA, "C:¥tmp¥tla2.txt"
```

VECTor:IMPort:AWG（問い合せなし）

AWG ファイルからパターン・データを読み込みます。

構　文：
```
VECTor:IMPort:AWG <format>, <filename>, <import bits>, <group name prefix>,
                  <group clear if the group does not exist state>
```

引　数： <format>::= { 0 | 1 | 2 | 3 | 4 }
0 : AWG400 シリーズ
1 : AWG500 シリーズ
2 : AWG600/700 シリーズ
3 : AWG202x/2005 型
4 : AWG204x 型

<filename> ::= <string> ― 絶対パスのファイル名

<import bits>:: = <string> ― AWG ファイルから読み込むビット
読み込むビットは、次のように書きます。

“D0D1D3D5M1M2”
全有効ビットを読み込む場合は、ALL を使用することもできます。

<group name prefix>::= <string> ― グループ名の接頭詞

注： グループ名は、<group name prefix> D0_、<group name prefix> D1_、・・・となり 1 グループ 1 チャンネルのグループが作成されます。

<group clear if the group does not exist state>::= <boolean>

ONまたは0以外:インポートするファイル内にDTGで定義されていないグループ名が存在するとき、および DTG 内に同じ名前でビット幅の異なるグループ名が存在する場合、既存のグループをすべて削除して新たなグループを作成します。

OFFまたは0:インポートするファイル内にDTGで定義されていないグループ名が存在する場合、そのグループを追加します。同じ名前でビット幅の異なるグループ名が存在する場合、そのグループのみ再定義を行ないます。

使用例： AWG400 シリーズで作成された test.pat というファイルから、D0D1M1M2 ビットのデータを、接頭詞 AWG400 を持つグループとして読み込みます。

```
VECTor:IMPort:AWG 0,"test.pat","D0D1M1M2","AWG400",ON
```

VECTor:IOFormat(?)

VECTor:DATA で転送するデータ項目及び書式を設定します。

構　文：
```
VECTor:IOFormat <signal>, <radix> [, <signal>, <radix> ...]
VECTor:IOFormat?
```

応　答：
```
<signal>, <radix> [, <signal>, <radix> ...]
```

引　数： <signal> ::= <string> ― DATA コマンドで転送するデータのバスまたは論理チャンネルを指定します。
Addr[]
Addr[0:3]
Addr[0..3]
Addr[3..0]
のように書きます。

注： Addr[] の様に、[] の中身を省略した場合は、Addr[<msb>:<lsb>] として処理されます。（例えば Addr が 8 ビット幅なら Addr[] は Addr[7:0] とみなされます。）

複数のチャンネルに対して問い合わせが行なわれた場合には、最初のチャンネルの値が返ります。例えば、SIGN:HIGH? "DATA[2..4]" では DATA[2] の値が返ります。

<radix> ::= { BINary | HEXadecimal | OCTal }

使用例：

```
VECTor:IOFormat "Adr[0:3]", HEX
```

論理チャンネル Adr0 から Adr3 までに HEX 形式でデータを転送するための設定をします。

```
VECTor:IOFormat?
```

次のような応答が返ります。

```
"Adr[0:3]", HEX
```

*WAI （問い合せなし）

実行中または実行待ちのコマンドの全処理が完了するまで、後のコマンドまたは問い合せコマンドの実行を待ちます。本機器は、すべてのコマンドは外部コントローラから送られてきた順に処理されます。

構　文： *WAI

使用例：

```
PGENA:CH1:HIGH 2.0;*WAI
```

```
PGENA:CH1:HIGH?
```

HIGH? の返事が返って来た時点で終了が確認できます。

第 3 章　ステータスとイベント

ステータス／イベント・レポーティング

ステータス・レポーティング機能

本機器には、SCPI および IEEE-488.2 規格に準拠したステータス・レポーティング機能があります。この機能は、機器にどのイベントが発生したか、また機器がどのような状態にあるかを調べるものです。

ステータス・レポーティング機構には、次のブロックが含まれます。

■ スタンダード・イベント・ステータス

このブロックで行われる処理は、ステータス・バイトに集約され、ユーザに必要なステータス / イベント情報を提供します。

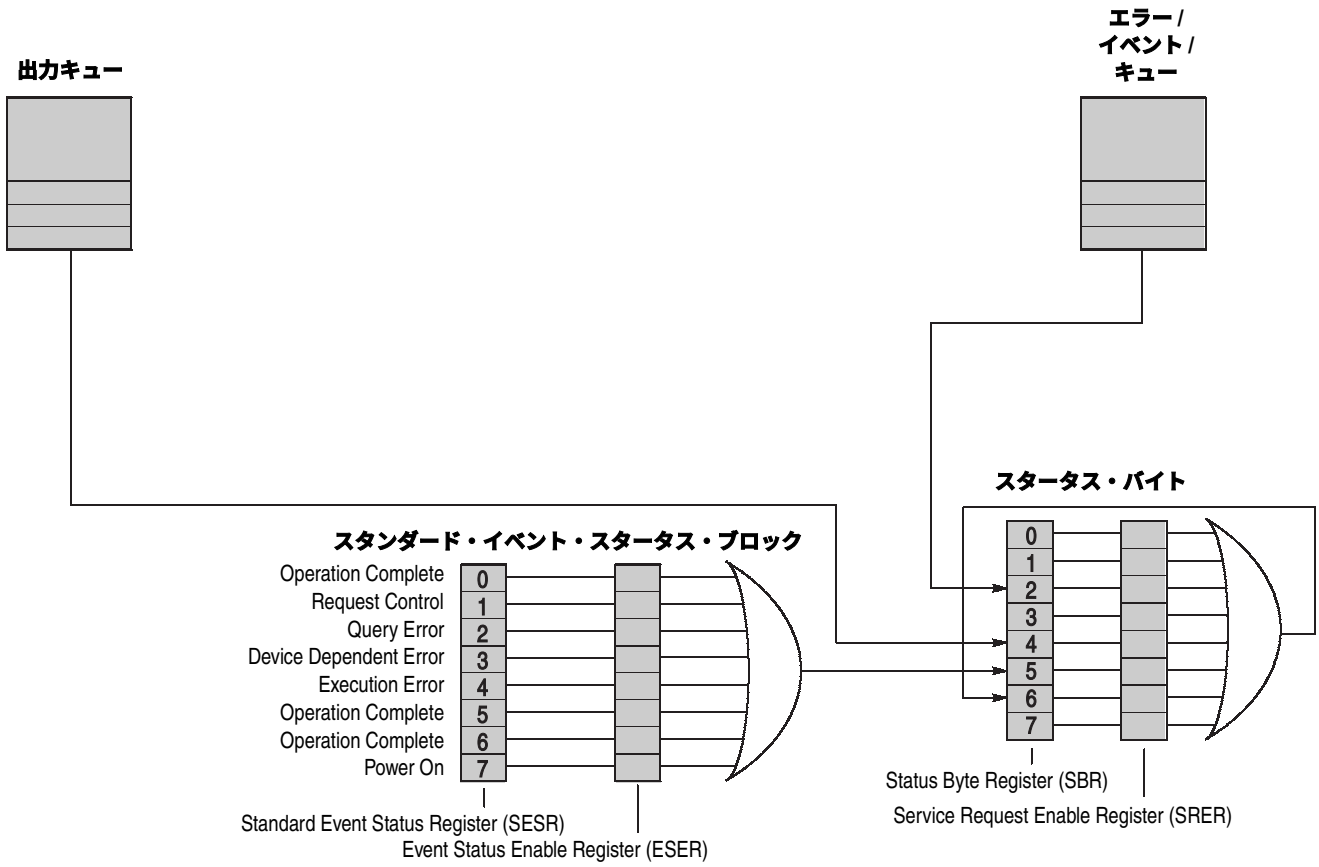


図 3-1：ステータス・レポーティング機構

スタンダード・イベント・ステータス・ブロック

電源のオン / オフ、コマンドのエラー、および実行状態をレポートするブロックです。

図 3-1 のスタンダード・イベント・ステータス・ブロックを参照してください。

このブロックは、次の 二つのレジスタで構成されています。

- イベント・ステータス・レジスタ (SESR: Standard Event Status Register)
  8 ビットのステータス・レジスタです。機器にエラーその他のイベントが発生すると、対応するビットがセットされます。ユーザの書き込みはできません。

- イベント・ステータス・イネーブル・レジスタ (SESR: Standard Event Status Register)
  8 ビットのイネーブル・レジスタで、SESR にマスクをかける働きをします。マスクは、ユーザが設定できます。SESR と論理積をとって、SBR (ステータスバイト・レジスタ) の ESB (bit 5：イベント・ステータスビット) をセットするかどうかを決定できます。

レジスタのビット内容については、次ページの「レジスタ」を参照してください。

処理の流れ

イベントが発生すると、イベントに対応する SESR のビットがセットされ、エラー / イベント・キューにイベントがスタックされます。SBR の EAV(bit 2) ビットもセットされます。イベントに対応するビットが ESER にもセットされていれば、SBR の ESB ビットもセットされます。

メッセージが出力キューに送られると、SBR の MAV ビットがセットされます。

レジスタ

レジスタは、大別すると次の 2 種類に分類されます。

- **ステータス・レジスタ**：機器のステータスに関するデータを保存します。このレジスタは、DTG5000 シリーズにより設定されます。

- **イネーブル・レジスタ**：機器内で発生するイベントを、ステータス・レジスタとイベント・キューの対応するビットにセットするかどうかを決めます。

ステータス・レジスタ

ステータス・レジスタには、次の 2 種類があります。

- ステータス・バイト・レジスタ (SBR)

- スタンダード・イベント・ステータス・レジスタ (SESR)

エラーと機器の状態を調べるときには、これらのレジスタの内容を読み出してください。

ステータス・バイト・レジスタ (SBR : Status Byte Register)

BR は、8 ビットで構成されます。ビット 4、5、6 は、IEEE Std 488.2-1992 に準拠しています。これらのビットはそれぞれ出力待ち行列、SESR、およびサービス・リクエストをモニタするために使用されます。このレジスタの内容は、問い合せ *STB? が送られたときに返されます。

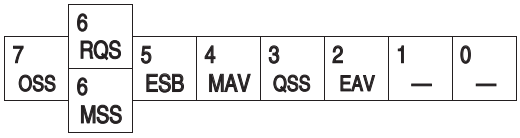


図 3-2：ステータス・バイト・レジスタ (SBR)

表 3-1：SRB のビット機能

ビット	機能
7	未使用
6	RQS (Request Service) / MSS (Master Summary Status)： 機器が GPIB シリアル・ポール・コマンドでアクセスされたとき、このビットはリクエスト・サービス (RQS) ビットとして機能し、コントローラに対してサービス・リクエストが発生 (GPIB バスの SRQ ラインが“L”) にしたことを示します。RQS ビットは、シリアル・ポールが終了したときにクリアされます。 機器が、問い合せ *STB? によりアクセスされた場合、このビットはマスタ・サマリ・ステータス (MSS) ビットとして機能し、機器が何かの理由によりサービス・リクエストを要求していることを示します。MSS ビットは、問い合せ *STB? で 0 になることはありません。
5	ESB (Event Status Bit)：前のスタンダード・イベント・ステータス・レジスタ (SESR) がクリアされた後、またはイベントの読み出しが実行された後に、新しいイベントが発生しているかどうかを示します。
4	MAV (Message Available Bit)：このビットは、メッセージが出力キュー内に置かれ、検索できることを示します。
3	未使用
2	EAV (Event Queue Available)
1-0	未使用

スタンダード・イベント・ステータス・レジスタ (SESR : Standard Event Status Register)

SESR は、8 ビットで構成されます。各ビットは、下に示すように様々なイベントの発生を記録します。このレジスタの内容は、問い合せ *ESR? を送ったときに返されます。

7	6	5	4	3	2	1	0
PON	—	CME	EXE	DDE	QYE	—	OPC

図 3-3：スタンダード・イベント・ステータス・レジスタ (SESR)

表 3-2：SESR のビット機能

ビット	機能
7	PON (Power On)：機器の電源がオンになっていることを示します。
6	未使用
5	CME (Command Error)：コマンドの構文解析でコマンド・テーブル検索中にコマンド・エラーが発生したことを示します。
4	EXE (Execution Error)：コマンド実行中にエラーが発生したことを示します。実行エラーは、次のいずれかの原因により発生します。 ■ 引数で指定された値が機器の許容範囲を超えているとき、または値が機器の仕様に合わないとき。 ■ 実行条件が不適切で、コマンドが正しく実行されなかったとき。
3	DDE (Device-Dependent Error)：機器固有のエラーが検出されたことを示します。
2	QYE (Query Error)：出力キュー・コントローラで、問い合せエラーが検出されたことを示します。このエラーは、次のいずれかの原因で発生します。 ■ 出力キューが空の状態またはステータスが未処理にもかかわらず、出力キューからメッセージを読み出そうとしたとき。 ■ 出力キュー・メッセージが、検索されていないにもかかわらず、クリアされたとき。
1	未使用
0	OPC (Operation Complete)：このビットは、*OPC コマンドの実行結果によりセットされます。未処理のすべての操作が完了したことを示します。

イネーブル・レジスタ

イネーブル・レジスタには、次の 4 種類があります。

- イベント・ステータス・イネーブル・レジスタ (ESER)
- サービス・リクエスト・イネーブル・レジスタ (SRER)

これらのイネーブル・レジスタの各ビットは、ステータス・レジスタの各ビットに対応しています。イネーブル・レジスタのビットをセット / リセットすることにより、発生したイベントをステータス・レジスタとキューに記録するかどうか決めます。

イベント・ステータス・イネーブル・レジスタ (ESER : Event Status Enable Register)

ESER は、SESR のビット 0 ～ 7 と全く同じビットで構成されています (下図参照)。このレジスタは、イベントが発生したときに SBR レジスタの ESB ビットをセットするか、また対応する SESR のビットをセットするかを指定するときに使います。

SBR レジスタの ESB ビットをセットするには (SESR ビットがセットされたとき)、SESR に対応する ESER のビットをセットします。ESB ビットがセットされるのを防ぐには、そのイベントに対応した ESER ビットをリセットします。

ESER のビットをセットするときは、*ESR コマンドを使います。また、ESER の内容を読み出すときは、問合せ *ESE? を使います。

7	6	5	4	3	2	1	0
PON	—	CME	EXE	DDE	QYE	—	OPC

図 3-4 :イベント・ステータス・イネーブル・レジスタ (ESER)

サービス・リクエスト・イネーブル・レジスタ (SRER : Service Request Enable Register)

SRER は、SBR のビット 0 ～ 7 に対応したビットで構成されています。このレジスタは、どのイベントでサービス・リクエストを発生するか指定するときに使います。

SRER のビット 6 は、セットできません。また、RQS はマスクできません。

GPIB インタフェースでのサービス・リクエスト発生には、SRQ ラインを “ L ” に変更すること、およびコントローラにサービス・リクエストを要求することが含まれます。この結果、コントローラからのシリアル・ポーリングに応答して、RQS がセットされたステータス・バイトが返されます。

SRER のビットをセットするときは、*SRE コマンドを使います。また、SRER の内容を読み出すときは、問合せ *SRE? を使います。ビット 6 は、通常 0 にセットされています。

7	6	5	4	3	2	1	0
OSS		ESB	MAV	QSS	EAV	—	—

図 3-5：サービス・リクエスト・イネーブル・レジスタ (SRER)

キュー

DTG5000 シリーズで使用されているステータス・レポーティング・システムには、出力キューとイベント・キューの 2 種類のキューがあります。

出力キュー

出力キューは FIFO (先入れ先出し方式) キューで、問合せに対する応答メッセージを保持します。このキューにメッセージがあるときは、SBR の MAV ビットがセットされます。

出力キューは、コマンドまたは問合せを受け取るごとに空になります。このため、コントローラは、次のコマンドまたは問合せが発生する前に出力キューを読み取る必要があります。もし、この動作が実行されないと、エラーが発生し、出力キューは空になります。ただし、エラーが発生しても、動作は継続されます。

エラー／イベント・キュー

イベント・キューは FIFO (先入れ先出し方式) キューで、機器内で発生したイベントを保持します。イベント・キューに溜められるイベントの数は最大で 100 です。

- :SYSTem:ERRor[:NEXT]?

ステータスとイベントの処理

ここでは、ブロックごとにステータスとイベントの処理の流れを示します。

スタンダード・イベント・ステータス・ブロック

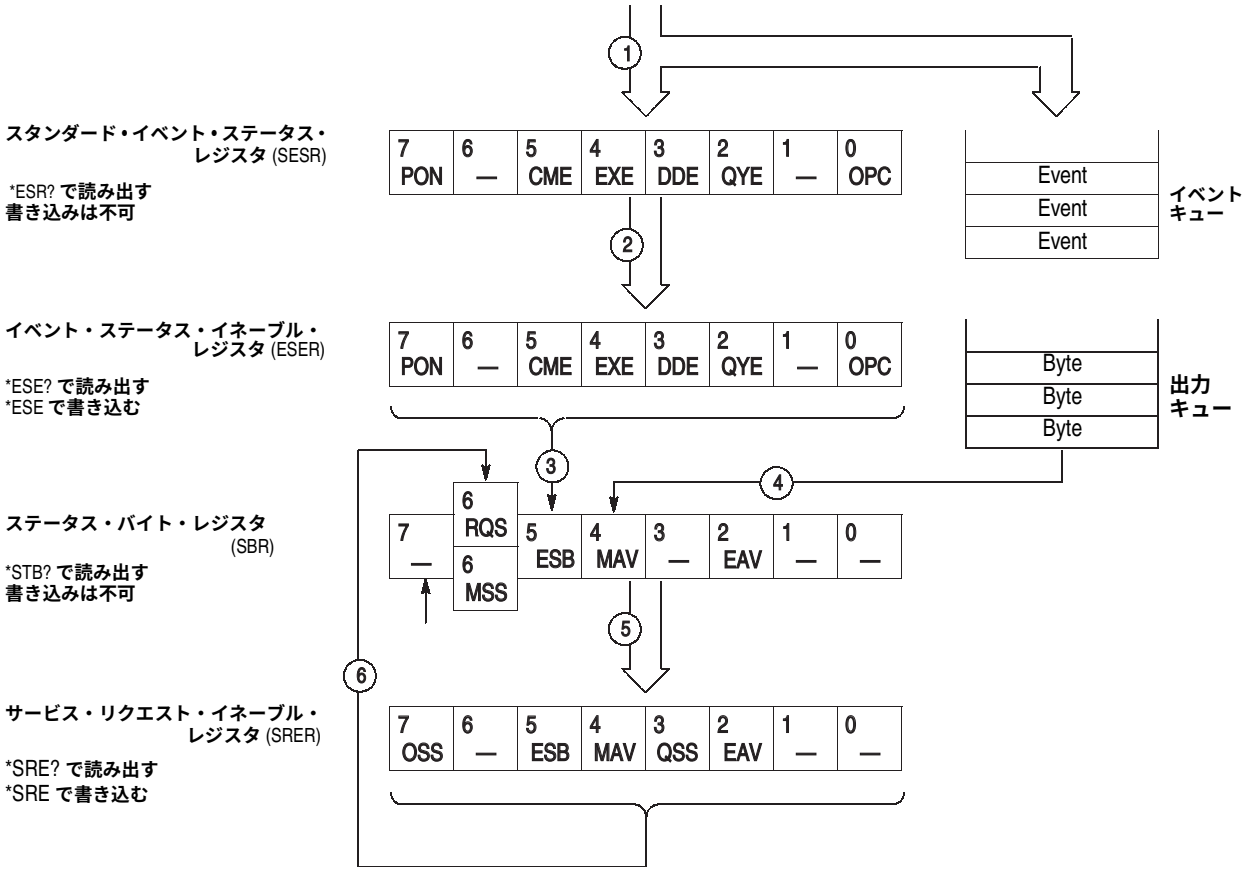


図 3-6：ステータスとイベントの処理 ― スタンダード・イベント・ステータス・ブロック

1. イベントが発生すると、そのイベントに対応する SESR ビットがセットされ、イベントがイベント・キューに記録されます。
2. そのイベントに対応した ESER のビットがセットされます。
3. ESER のステータスによって SBR ESB ビットがセットされます。
4. メッセージが出力キューに送られると、SBR MAV ビットがセットされます。
5. SBR の ESB ビットまたは MAV ビットのいずれかがセットされることで、SRER のそれぞれのビットがセットされます。
6. SRER ビットがセットされていれば、SBR MSS ビットがセットされ、GPIB を使用している場合はサービス・リクエストが発生します。

コマンドの同期実行

本機器ではコマンドの実行はシーケンシャルに行なわれますが、コマンドの実行が完全にハードウェアに対して行なわれたかどうかは以下の方法では確認できません。

```
PGENA:CH1:HIGH 2.0
PGENA:CH1:HIGH?
```

上記の例で HIGH? の返事が返ってきても、その時点でその前のコマンドのハードウェアへの設定は終了していない可能性があります。以下の方法によりハードウェアへの設定終了を確認することが出来ます。

```
PGENA1:CH1:HIGH 2.0;*OPC
```

--> Operation Complete のイベントが発生した時点で終了が確認できます。

```
PGENA1:CH1:HIGH 2.0;*OPC?
```

--> 1 が返って来た時点で終了が確認できます。

```
PGENA:CH1:HIGH 2.0;*WAI
PGENA:CH1:HIGH?
```

--> HIGH? の返事が返って来た時点で終了が確認できます。

メッセージ

次ページ以降に、イベント・レポーティング・システムで使われているコードとメッセージを示します。

イベント・コードとメッセージは、問合せ SYSTem:ERRor[:NEXT]? を使って得られます。レスポンスは、次の書式で返されます。

```
<event code>,"<event message>"
```

表 3-4 は、コマンド内にシンタックス・エラーがあるときに発生するメッセージを示しています。

表 3-5 は、コマンドが実行されたにもかかわらず、エラーが検出されたときに発生するメッセージを示しています。

表 3-6 は、内部機器エラーが検出されたときに発生するメッセージを示しています。

表 3-7 は、システム・イベントに対するメッセージを示しています。これらのメッセージは、機器の状態が変化したときに発生します。

エラー／イベント・コードとメッセージ

この節では、エラーおよびイベント・コードとメッセージを示します。

表に示すように、エラー / イベント・コードはクラスごとに分類されています。コードは、SCPI 規格です。

表 3-3：エラー・コードの定義

クラス	コードの範囲	説　明
エラーなし	0	イベントまたはステータスなし
コマンド・エラー	-100 ～ -199	コマンド構文エラー
実行エラー	-200 ～ -299	コマンド実行エラー
デバイス固有エラー	-300 ～ -399	機器内部 (ハードウェア) エラー
問合せエラー	-400 ～ -499	システム・イベントおよび問合せエラー
電源投入時イベント	-500 ～ -599	電源投入時に発生するイベント
ユーザ・リクエスト・イベント	-600 ～ -699	ユーザ・リクエスト検出時に発生するイベント
リクエスト・コントロール・イベント	-700 ～ -799	コントロール要求時に発生するイベント
操作終了時イベント	-800 ～ -899	コマンド実行終了時に発生するイベント

コマンド・エラー

コマンド・エラーは、コマンド中に構文エラーが存在する場合に発生します。

表 3-4：コマンド・エラー

エラー・コード	エラー・メッセージ
-100	Command error（コマンド・エラー）
-101	Invalid character（文字が不適当）
-102	Syntax error（構文エラー）
-103	Invalid separator（セパレータが不適当）
-104	Data type error （データ・タイプ・エラー）
-105	GET not allowed（GET は使用不可）
-108	Parameter not allowed（パラメータは使用不可）

表 3-4：コマンド・エラー （続き）

エラー・コード	エラー・メッセージ
-109	Missing parameter（パラメータが見つからない）
-110	Command header error（コマンド・ヘッダ・エラー）
-111	Header separator error（ヘッダ・セパレータ・エラー）
-112	Program mnemonic too long（プログラム・ニーモニックが長すぎる）
-113	Undefined header（ヘッダが未定義）
-114	Header suffix out of range（ヘッダ・サフィックスが範囲外）
-115	Unexpected number of parameters（パラメータの数が不正）
-120	Numeric data error（数値データ・エラー）
-121	Invalid character in number（数値データで不当なキャラクタを使用）
-123	Exponent too large（指数が大きすぎる）
-124	Too many digits（桁が多すぎる）
-128	Numeric data not allowed（数値データは使用不可）
-130	Suffix error（サフィックス・エラー）
-131	Invalid suffix（サフィックスが不適当）
-134	Suffix too long（サフィックスが長すぎる）
-138	Suffix not allowed（サフィックスは使用不可）
-140	Character data error（文字データ・エラー）
-141	Invalid character data（文字データが不適当）
-144	Character data too long（文字データが長すぎる）
-148	Character data not allowed（文字データは使用不可）
-150	String data error（ストリング・データ・エラー）
-151	Invalid string data（ストリング・データが不適当）
-158	String data not allowed（ストリング・データは使用不可）
-160	Block data error（ブロック・データ・エラー）
-161	Invalid block data（ブロック・データが不適当）
-168	Block data not allowed（ブロック・データは使用不可）
-170	Command expression error（コマンド式エラー）
-171	Invalid expression（表現式が不適当）
-178	Expression data not allowed（表現式データは使用不可）
-180	Macro error（マクロ・エラー）
-181	Invalid outside macro definition（マクロ定義の最大が不適当）
-183	Invalid inside macro definition（マクロ定義の最小が不適当）
-184	Macro parameter error（マクロ・パラメータ・エラー）

実行エラー

これらのエラー・コードは、コマンドが実行されている間にエラーが検出されたときに発生します。

表 3-5：実行エラー

エラー・コード	エラー・メッセージ
-200	Execution error（実行エラー）
-201	Invalid while in local（ローカル制御では無効）
-202	Settings lost due to RTL（RTL のために設定が消失）
-203	Command protected（コマンドが保護されている）
-210	Trigger error（トリガ・エラー）
-211	Trigger ignored（トリガを無視）
-212	Arm ignored（アーミングを無視）
-213	Init ignored（初期化を無視）
-214	Trigger deadlock（トリガ停止）
-215	Arm deadlock（アーミング停止）
-220	Parameter error（パラメータ・エラー）
-221	Settings conflict（設定の矛盾）
-222	Data out of range（データが範囲外）
-223	Too much data（データが多すぎる）
-224	Illegal parameter value（パラメータ値が無効）
-225	Out of memory（メモリ容量が不足）
-226	Lists not same length（リストが同じ長さでない）
-230	Data corrupt or stale（データが破壊または消失）
-231	Data questionable（データに問題がある）
-232	Invalid format（フォーマット・エラー）
-233	Invalid version（バージョン・エラー）
-240	Hardware error（ハードウェア・エラー）
-241	Hardware missing（ハードウェアが見つからない）
-250	Mass storage error（マス・ストレージ・エラー）
-251	Missing mass storage（マス・ストレージが見つからない）
-252	Missing media（メディアが見つからない）
-253	Corrupt media（メディアが破壊）
-254	Media full（メディアに空きがない）

表 3-5：実行エラー （続き）

エラー・コード	エラー・メッセージ
-255	Directory full（ディレクトリに空きがない）
-256	File name not found（ファイル名が見つからない）
-257	File name error（ファイル名エラー）
-258	Media protected（メディア書き込み禁止）
-260	Execution expression error（実行式エラー）
-261	Math error in expression（式の演算エラー）
-270	Execution macro error（マクロ式エラー）
-271	Macro syntax error（マクロ構文エラー）
-272	Macro execution error（マクロ実行エラー）
-273	Illegal macro label（マクロ・ラベルが無効）
-274	Execution macro parameter error（実行マクロ・パラメータ・エラー）
-275	Macro definition too long（マクロ定義が長過ぎ）
-276	Macro recursion error（マクロ反復エラー）
-277	Macro redefinition not allowed（マクロの再定義は不可）
-278	Macro header not found（マクロ・ヘッダが見つからない）
-280	Program error（プログラム・エラー）
-281	Cannot create program（プログラムが作成できない）
-282	Illegal program name（プログラム名が無効）
-283	Illegal variable name（変数名が無効）
-284	Program currently running（プログラム実行中）
-285	Program syntax error（プログラム構文エラー）
-286	Program runtime error（プログラム実行エラー）
-290	Memory use error（メモリ使用エラー）
-291	Out of memory（メモリ範囲外）
-292	Referenced name does not exist（参照名が存在しない）
-293	Referenced name already exist（参照名が既に存在する）
-294	Incompatible type（不適合タイプ）

デバイス固有エラー

これらのエラー・コードは、機器の内部でエラーが検出されたときに発生します。デバイス固有エラーは、ハードウェアに問題があることを示します。

表 3-6：デバイス固有エラー

エラー・コード	エラー・メッセージ
-300	Device specific error（デバイス固有エラー）
-310	System error（システム・エラー）
-311	Memory error（メモリ・エラー）
-312	PUD memory lost（PUD メモリの内容が消失）
-313	Calibration memory lost（校正メモリの内容が消失）
-314	Save/recall memory lost（保存 / 呼び出しでメモリの内容が消失）
-315	Configuration memory lost （コンフィギュレーション・メモリの内容消失）
-320	Storage fault（保存できない）
-321	Out of memory（メモリ範囲外）
-330	Self test failed（セルフテスト・エラー）
-340	Calibration failed（校正エラー）
-350	Queue overflow（キュー・オーバフロー）
-360	Communication error（通信エラー）
-361	Parity error in program message （プログラム・メッセージのパリティ・エラー）
-362	Framing error in program message （プログラム・メッセージのフレーム・エラー）
-363	Input buffer overrun（入力バッファ超過）
-364	Time out error（タイムアウト・エラー）

問合せエラー

これらのエラー・コードは、応答できない問合せに対して発生します。

表 3-7：問合せエラー

エラー・コード	エラー・メッセージ
-400	Query error（問合せエラー）
-410	Query interrupted（問合せの中断）
-420	Query unterminated（問合せが終了していない）
-430	Query deadlocked（問合せの処理が停止）
-440	Query unterminated after indefinite response （不定長のレスポンス発生後の問い合せ中断）

電源投入時イベント

このイベント・コードは、本機器の電源がオフからオンに切り替わったときに発生します。

表 3-8：電源投入時イベント

エラー・コード	エラー・メッセージ
-500	Power on（電源オン）

ユーザ・リクエスト時イベント

本イベントは DTG5000 シリーズでは使用されません。

表 3-9：ユーザ・リクエスト時イベント

エラー・コード	エラー・メッセージ
-600	User request（ユーザ・リクエスト）

リクエスト・コントロール時イベント

本イベントは DTG5000 シリーズでは使用されません。

表 3-10：リクエスト・コントロール時イベント

エラー・コード	エラー・メッセージ
-700	Request control（リクエスト・コントロール）

操作終了時イベント

このイベント・コードは、*OPC コマンドで同期をとる場合、前のコマンドが完了したときに発生します。

表 3-11：操作終了時イベント

エラー・コード	エラー・メッセージ
-800	Operation complete（操作終了）

第４章　プログラム例

プログラム例

DTG5000 シリーズをコントロールするプログラム例を下に示します。

このプログラム例は、GPIB 使用したリモート・コントロールのプログラム例で、Visual Basic V6.0 で書かれています。

GPIB プログラムは、ナショナル・インスツルメンツ社製 GPIB ボードおよびドライバ・ソフトウェアを装備した PC 上で動作します。また、ナショナル・インスツルメンツ社 LabVIEW でも使用できます。

サンプル・プログラム

４ビットのバイナリ・カウンタを繰り返し出力する Visual Basic V6.0 サンプル・プログラムです。

```
=======================================================

Option Explicit

Private Sub Command1_Click()
    ´Following is the example to
    Dim dev%, i&, count&
    Dim s$, ss$, blockSize&

    blockSize = 1024

    ´select GPIB Board 0
    ´Primary Address 1
    ´Secondary Address None
    ´Timeout 3s
    ´Assert EOI at the end of ibwrt
    ´Do not handle EOS character automatically
    ibdev 0, 1, 0, T10s, 1, 0, dev

    ibwrt dev, "*CLS"
    ibwrt dev, "*RST"
    ibwrt dev, "GROUP:DELETE:ALL"
    ibwrt dev, "GROUP:NEW ""GRP1"",4"
    ibwrt dev, "BLOCK:DELETE:ALL"
    ibwrt dev, "BLOCK:NEW ""BLK1"", " & blockSize
    ibwrt dev, "BLOCK:SELECT ""BLK1"""
    ibwrt dev, "VECTOR:IOFORMAT ""GRP1"", HEX"

    ´Define 4bit counter pattern
    count = blockSize / 16
```

```
    s = "0123456789ABCDEF"
    For i = 0 To count - 1
        ss = ss & s
    Next
    ´Send block data
    ibwrt dev, "VECTOR:DATA 0, " & blockSize & ", """ & ss & """"

    ibwrt dev, "SEQUENCE:LENGTH 1"
´Define Sequence
    ibwrt dev, "SEQUENCE:DATA 0, """", 0, ""BLK1"",0,"""","""""
´Infinite Loop of BLK1
    ibwrt dev, "SIGNAL:ASSIGN ""GRP1[3]"", ""A1"""
´Channel Assign
    ibwrt dev, "SIGNAL:ASSIGN ""GRP1[2]"", ""A2"""
    ibwrt dev, "SIGNAL:ASSIGN ""GRP1[1]"", ""B1"""
    ibwrt dev, "SIGNAL:ASSIGN ""GRP1[0]"", ""B2"""
    ibwrt dev, "TBAS:FREQ 100e6"
´Set Frequency
    ibwrt dev, "SIGNAL:HIGH ""GRP1[]"", 0.5"
´Set High Level
    ibwrt dev, "SIGNAL:LOW ""GRP1[]"", -0.0"
´Set Low Level
    For i = 0 To 3
        ibwrt dev, "SIGNAL:OUTPUT ""GRP1[" & i & "]"", 1"
´Output On
    Next
    ibwrt dev, "TBAS:RUN 1"
´Start Sequencer

End Sub

=======================================================
```

付　録

付録 A　GPIB インタフェース仕様

インタフェース機能

インタフェース機能は、IEEE Std 488.2-1992 で定義されているもので、メッセージを送信したり、メッセージを受信したり、あるいはメッセージに従って機器を制御する機能です。表 A-1 に、組み込まれたインタフェース機能を示します。括弧で囲んだ略号は、IEEE Std 488.2-1992 で定義され、広く使用されているインタフェース・ファンクションを示す記号です。

表 A-1：GPIB インタフェース機能と組み込みサブセット

インタフェース機能	組み込みサブセット	サブセットの機能
Acceptor Handshake（AH）	AH1	AH の全機能
Source Handshake（SH）	SH1	SH の全機能
Talker（T）	T6	基本トーカ、シリアル・ポール MLA によるアクティブ・トーカの解除 Talk Only モードなし
Listener（L）	L4	基本リスナ MTA によるアクティブ・リスナの解除 Listen Only モードなし
Device Clear（DC）	DC1	DC の全機能
Remote/Local（RL）	RL1	RL の全機能
Service Request（SR）	SR1	SR の全機能
Parallel Poll（PP）	PP0	サポートしません
Device Trigger（DT）	DT1	DT の全機能
Controller（C）	C0	サポートしません
Electrical Interface	E2	3 ステート・ドライバ

- Acceptor Handshake (AH)
  データを確実に受信するためのハンドシェイク機能です。この機能は、機器が次のデータの受信準備が完了するまで、データの送出開始と完了を遅らせます。

- Source Handshake (SH)
  データを確実に鄭 H との間でハンドシェイクを行う機能です。この機能は、バイト単位にデータの送出開始と完了を制御します。

- Listener (L)
  バスを通して、デバイス依存データを受信できる機能です。ただし、データを受信できるのは、受信指定されたアドレスを持つリスナに限ります。

- Talker (T)
  バスを通して、デバイス依存データを送出できる機能です。ただし、データを送出できるのは、送信指定されたアドレスを持つトーカに限ります。

- Device Clear (DC)
  システムに接続された機器を、個々に、またはまとめて初期化を行う機能です。

- Remote / Local (RL)
  機器を操作する方法を選択します。機器の制御には、前面パネルの操作（ローカル・コントロール）による方法と、インタフェースを通して、コントローラから操作（リモート・コントロール）する方法の、二つの方法があります。

- Service Request (SR)
  コントローラに対して、非同期のサービスを要求する機能です。

- Controller (C)
  バスを通して、他の機器に、デバイス・アドレス、ユニバーサル・コマンド、アドレス・コマンドを送出する機能です。デバイス・アドレス、ユニバーサル・コマンド、アドレス・コマンドについては、次項の「インタフェース・メッセージ」を参照ください。

- Electrical Interface (E)
  電気的インタフェースのタイプを示すもので、インタフェース機能には含まれません。記号としては E1 および E2 が使用され、インタフェースのタイプが、それぞれ 3 ステート・ドライバ、オープン・コレクタ・ドライバであることを示します。

インタフェース・メッセージ

次の表に、本機器に組み込まれた GPIB ユニバーサル・コマンドとアドレス・コマンドを示します。

表 A-2：GPIB インタフェース・メッセージ

インタフェース・メッセージ	種別	組み込み
Device Clear（DCL）	UC	Yes
Local Lockout（LLO）	UC	Yes
Serial Poll Disable（SPD）	UC	Yes
Serial Poll Enable（SPE）	UC	Yes
Parallel Poll Unconfigure（PPU）	UC	No
Go To Local（GTL）	AC	Yes
Selected Device Clear（SDC）	AC	Yes
Group Execute Trigger（GET）	AC	Yes
Take Control（TCT）	AC	No
Parallel Poll Configure（PPC）	AC	No

* UC、AC は、それぞれユニバーサル・コマンド、アドレス・コマンドを表します。

- Device Clear (DCL)
  DCL インタフェース・メッセージが組み込まれたすべての機器を初期化します。

- Local Lockout (LLO)
  ローカル状態に戻る機能を無効にします。この場合、前面パネルからの操作はできなくなります。

- Serial Poll Enable (SPE)
  サービス要求機能を持つすべての機器を、シリアル・ポール・モードにします。このモードの機器は、コントローラが送出するトーク・アドレスを受け取ると、コントローラにステータス・バイトを戻します。コントローラは、シリアル・ポーリングによって、サービス要求を行った機器を特定することができます。

- Serial Poll Disable (SPD)
  サービス要求機能を持つすべての機器に対して、シリアル・ポール・モードを解除し、通常の動作モードに戻します。

- Go To Local (GTL)
  リモート・コントロール状態を解除して、ローカル・コントロール状態に戻します。

- Select Device Clear (SDC)
  DCL インタフェース・メッセージが組み込まれた機器を初期化します。

- Group Execute Trigger (GET)
  特定の機器、またはあるグループの機器に対してトリガをかけ、プログラムされた機能を実行します。

- Take Control (TCT)
  バスを管理しているコントローラから、コントローラの機能を有する他の機器に、バス管理権を移します。

- Parallel Poll Configure (PPC)
  PPC コマンドに続いて送出される PPE (Parallel Poll Enable) コマンドと PPD (Parallel Poll Disable) コマンドに従い、パラレル・ポールのモードを設定および解除します。

付録 B　工場出荷時設定

下表にコマンドのデフォルト設定値を示します。

*RST コマンドはステータス・コマンドには影響しません。

表 B-1：デフォルト設定値

ヘッダ	設定値
BLOCk:SELect	“”
DIAGnostic:SELect	ALL
JGENeration:AMPLitude	0
JGENeration:AMPLitude:UNIT	SPP
JGENeration:EDGE	BOTH
JGENeration:FREQuency	1e6
JGENeration:GSOurce	“”
JGENeration:MODE	ALL
JGENeration:PROFile	SINusoid
JGENeration[:STATe]	0
OUTPut:CLOCk:AMPLitude	1.0
OUTPut:CLOCk:OFFSet	0.48
OUTPut:CLOCk[:STATe]	0
OUTPut:CLOCk:TIMPedance	50
OUTPut:CLOCk:TVOLtage	0
OUTPut:DC:HLIMit	1.0
OUTPut:DC:LEVel	1.0
OUTPut:DC:LIMit	0
OUTPut:DC:LLIMit	0
OUTPut:DC[:STATe]	0
PGEN<x>[m]:CH<n>:AMODe	NORMal
PGEN<x>[m]:CH<n>:AMPLitude	1
PGEN<x>[m]:CH<n>:CPOint	50
PGEN<x>[m]:CH<n>:DCYCle	50
PGEN<x>[m]:CH<n>:DTOFfset	0
PGEN<x>[m]:CH<n>:DTOFfset:STATe	0

表 B-1：デフォルト設定値 （続き）

ヘッダ	設定値
PGEN<x>[m]:CH<n>:HIGH	1.0
PGEN<x>[m]:CH<n>:HLIMit	1.0
PGEN<x>[m]:CH<n>:JRANge	2e-9
PGEN<x>[m]:CH<n>:LDELay	0
PGEN<x>[m]:CH<n>:LHOLd	LDELay
PGEN<x>[m]:CH<n>:LIMit	0
PGEN<x>[m]:CH<n>:LLIMit	0
PGEN<x>[m]:CH<n>:LOW	0
PGEN<x>[m]:CH<n>:OFFSet	0.5
PGEN<x>[m]:CH<n>:OUTPut	0
PGEN<x>[m]:CH<n>:PHASe	0
PGEN<x>[m]:CH<n>:POLarity	NORMal
PGEN<x>[m]:CH<n>:PRATe	NORMal
PGEN<x>[m]:CH<n>:SLEW	2.25
PGEN<x>[m]:CH<n>:TDELay	5e-9
PGEN<x>[m]:CH<n>:THOLd	DCYCle
PGEN<x>[m]:CH<n>:TIMPedance	50
PGEN<x>[m]:CH<n>:TVOLtage	0
PGEN<x>[m]:CH<n>:TYPE	NRZ
PGEN<x>[m]:CH<n>:WIDTh	5e-9
SEQuence:LENGth	1
SIGNal:ASSign	自動アサイン
SIGNal:JRANge	2e-9
SUBSequence:LENGth	-1
SUBSequence:SELect	“”
SYSTem:KLOCk	0
TBAS:COUNt	1
TBAS:CRANge	12
TBAS:DOFFset	0
TBAS:EIN:IMPedance	1e3
TBAS:EIN:LEVel	1.4
TBAS:EIN:POLarity	NORMal
TBAS:FREQuency	1e8
TBAS:JMODe	EVENt
TBAS:JTIMing	SYNC
TBAS:LDELay	0
TBAS:MODE	CONTinuous

表 B-1：デフォルト設定値 （続き）

ヘッダ	設定値
TBAS:OMODe	DATA
TBAS:PERiod	1e-8
TBAS:SMODe	HARDware
TBAS:SOURce	INTernal
TBAS:TIN:IMPedance	1e3
TBAS:TIN:LEVel	1.4
TBAS:TIN:SLOPe	POSitive
TBAS:TIN:SOURce	EXTernal
TBAS:TIN:TIMer	1e-3

付録 C　ファイル・フォーマット

付録 C では、次の項目について説明します。

- ファイル・フォーマットとレコード・フォーマット
- レコード ID
- ファイル・ロードについて
- チャンネル・アサインについて

ファイル・フォーマットとレコード・フォーマット

DTG で作成されるファイルは、複数のレコードが羅列されたバイナリ形式のデータで、ファイルの拡張子には .dtg が使用されます。

レコードは図 C-1 に示すように、レコード ID、データ・サイズ、またはデータにより構成されます。レコード ID は、ツリー構造になっており（図 C-2 参照）、中間ノードのレコードとリーフ・ノードのレコードでその構造が異なります。中間ノードでのレコードはデータを持ちませんが、リーフ・ノードのレコードは必ずデータ・サイズ分のデータを持ちます。中間ノードのレコードのデータ・サイズには、中間ノード全体のデータ・サイズが設定されます。

中間ノードの場合

レコード ID (16 bit 符号なし整数)	データ・サイズ (32 bit 符号なし整数)

リーフ・ノードの場合

レコード ID (16 bit 符号なし整数)	データ・サイズ (32 bit 符号なし整数)	データ

図 C-1：レコード・フォーマット

レコード ID、データ・サイズ、および数値データは、ファイルの入／出力時にリトルエンディアンで扱われます。

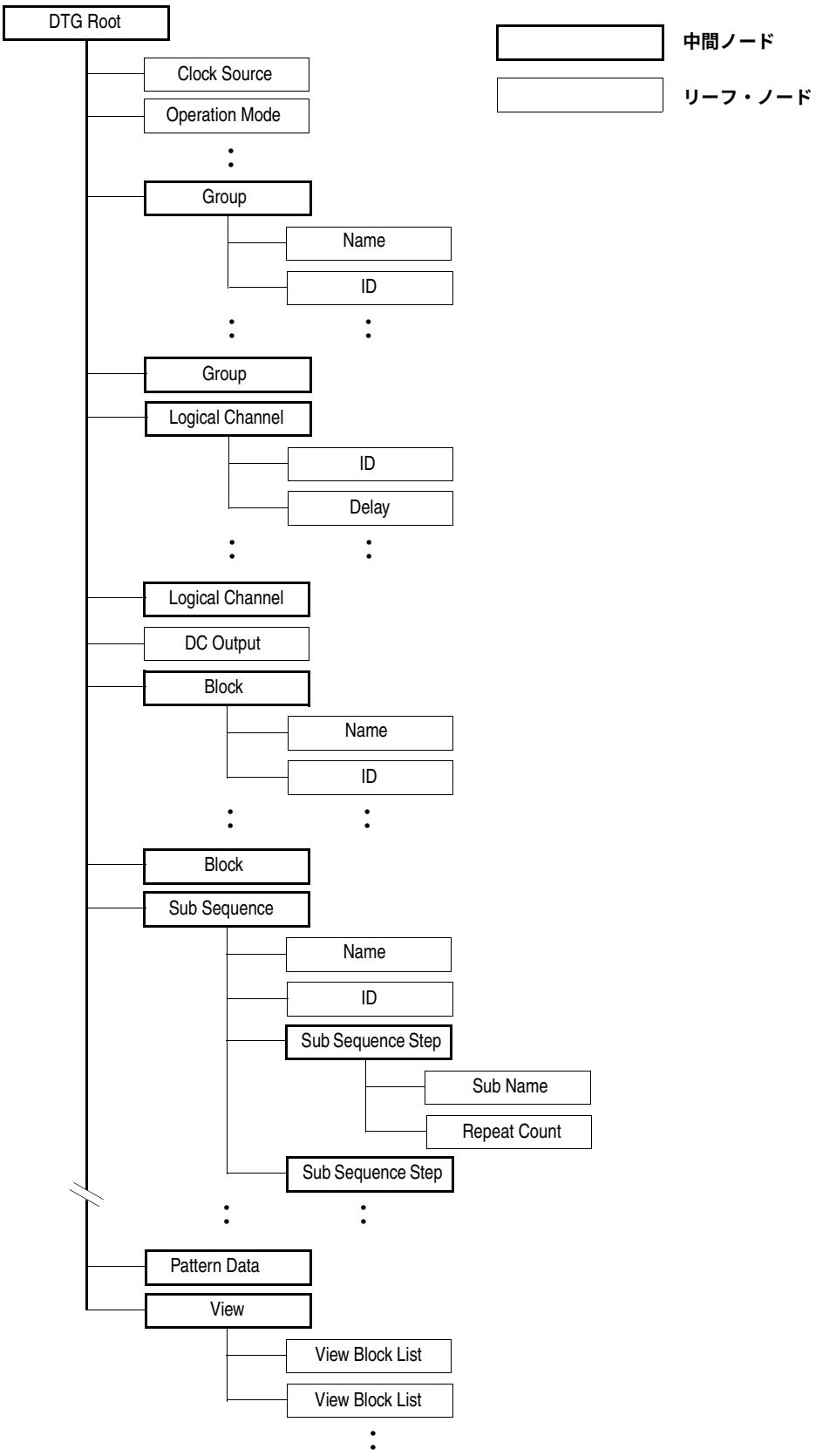


図 C-2：レコード ID のツリー構造

レコード ID 一覧

表 C-1 から表 C-10 に、DTG5000 シリーズで使用されているレコード ID を示します。各レコード ID の実際の値は、saveLoadRecid.h ヘッダ・ファイルを参照してください。また、初期値に具体的な数値が明記されていない箇所については、dtgDefinition.h ヘッダ・ファイルを参照してください。これらのファイルは、C:\Program Files\Tektronix\DTG5000\Samples フォルダに収められています。

中間ノード

表 C-1 に、中間ノードのレコード ID を示します。

表 C-1：中間ノードのレコード ID

ID	データ型	サイズ (byte)	初期値	備考
DTG_ROOT_RECID				
DTG_GROUP_RECID				
DTG_LOGICALCH_RECID				
DTG_BLOCK_RECID				
DTG_SUBSEQUENCE_RECID				= 1 SubSequence。DTG_SUBSEQUENCESTEP_RECID の集合体。
DTG_SUBSEQUENCESTEP_RECID				= 1 step。
DTG_MAINSEQUENCE_RECID				= 1 step。
DTG_PATTERN_RECID				
DTG_VIEW_RECID				

リーフ・ノード

表 C-2 から表 C-10 に、リーフ・ノードのレコード ID を示します。

Root

表 C-2：レコード ID-Root

ID	データ型	サイズ (byte)	パラメータ	初期値	備考
DTG_MAGIC_RECID	short	2		7000	この値が 7000 ～ 7999 以外の場合、DTG 以外のファイルと見なされ、ロードできません。
DTG_TB_OPERATION_RECID	short	2	OPE_MODE_DG OPE_MODE_PG	OPE_MODE_DG	DG または PG。
DTG_TB_RUNMODE_RECID	short	2	RUN_MODE_CONTINUOUS RUN_MODE_BURST	RUN_MODE_CONTINUOUS	TimeBase。PG のみ。
DTG_BURSTCOUNT_RECID	long	4			TimeBase。PG のみ。

表 C-2：レコード ID-Root （続き）

ID	データ型	サイズ (byte)	パラメータ	初期値	備考
DTG_TB_CLOCKSOURCE_RECID	short	2	CLK_SOURCE_INT CLK_SOURCE_EXT_10MHZ CLK_SOURCE_EXT_PLL CLK_SOURCE_EXT	CLK_SOURCE_INT	TimeBase
DTG_TB_CLOCKSETTYPE_RECID	short	2	CLOCK_SETUP_MODE_FREQUENCY CLOCK_SETUP_MODE_PERIOD	CLOCK_SETUP_MODE_FREQUENCY	TImeBase。FREQUENCY または PERIOD を指定する。
DTG_TB_CLOCK_RECID	double	8	CLOCK_SETUP_MODE_PERIOD	100.000000 MHz	TimeBase。DTG_TB_CLOCKSETTYPE_RECID の値により、単位が変わる。 FREQUENCY: Hz PERIOD: sec
DTG_TB_JITTERINPUT_RECID	short	2	FALSE TRUE	FALSE	TimeBase。 DG モードのみ。
DTG_TB_LONGDELAY_RECID	short	2	FALSE TRUE	FALSE (PG モードの時は TRUE)	TimeBase。 DG モードのみ。
DTG_TB_CLOCKRANGE_RECID	short	2	CLOCK_RANGE_1 CLOCK_RANGE_2 CLOCK_RANGE_3 CLOCK_RANGE_4 CLOCK_RANGE_5 CLOCK_RANGE_6 CLOCK_RANGE_7 CLOCK_RANGE_8 CLOCK_RANGE_9 CLOCK_RANGE_10 CLOCK_RANGE_11 CLOCK_RANGE_12 CLOCK_RANGE_13 CLOCK_RANGE_14 CLOCK_RANGE_15 CLOCK_RANGE_16	CLOCK_RANGE_13	TimeBase。DG モードかつ LongDelay=ON の場合のみ。
DTG_TB_DELAYOFFSET_RECID	double	8		0.000 s	TimeBase。
DTG_TB_CLOCKOUTPUT_RECID	short	2	FALSE TRUE	FALSE	TimeBase。
DTG_TB_CLOCKOUTPUTAMPLITUDE_RECID	double	8		1.000 Vp-p	TimeBase。
DTG_TB_CLOCKOUTPUTOFFSET_RECID	double	8		0.5 V	TimeBase。
DTG_TB_CLOCKOUTPUTTERMV_RECID	double	8		0 V	TimeBase。
DTG_TB_CLOCKOUTPUTTERMZ_RECID	long	4		50 ohm	TimeBase。
DTG_TRIGGER_SOURCE_RECID	short	2	INTERNAL EXTERNAL	EXTERNAL	Trigger。

表 C-2：レコード ID-Root （続き）

ID	データ型	サイズ (byte)	パラメータ	初期値	備考
DTG_TRIGGER_SLOPE__RECID	short	2	POLARITY_PLUS POLARITY_MINUS	Positive (POLARITY_ PLUS)	Trigger。TriggerSource が External の場合のみ。
DTG_TRIGGER_LEVEL_RECID	double	8		1.4 V	Trigger。TriggerSource が External の場合のみ。
DTG_TRIGGER_IMPEDANCE_ RECID	short	2	IMPEDANCE_HI IMPEDANCE_50 IMPEDANCE_1K	IMPEDANCE_ 1K	Trigger。TriggerSource が External の場合のみ。
DTG_TRIGGER_INTERVAL_RECID	double	8		1.000 ms	Trigger。TriggerSource が Internal の場合のみ。
DTG_EVENT_INPUTPOLARITY_ RECID	short	2	POLARITY_PLUS POLARITY_MINUS	Positive (POLARITY_ PLUS)	Event。
DTG_EVENT_INPUTTHRESHOLD_ RECID	double	8		1.4 V	Event。
DTG_EVENT_INPUTIMPEDANCE_ RECID	short	2	IMPEDANCE_HI IMPEDANCE_50 IMPEDANCE_1K	IMPEDANCE_ 1K	Event。
DTG_SEQ_MODE_RECID	short	2	SEQUENCER_MODE _HW SEQUENCER_MODE _SW	SEQUENCER_ MODE_HW (PG の時は SEQUENCER_ MODE_SW)	Sequence。DG モードのみ。
DTG_SEQ_JUMPMODE_RECID	short	2	JUMP_MODE_EVENT JUMP_MODE_COMM AND	JUMP_MODE_ EVENT	Sequence。DG モードかつ SequenceMode が HW の場合のみ。
DTG_SEQ_JUMPTIMING_RECID	short	2	JUMP_TIMING_ ASYNC JUMP_TIMING_SYNC	JUMP_TIMING_ SYNC	Sequence。DG モードかつ SequenceMode が HW の場合のみ。
DTG_JITTER_GENERATION_RECID	short	2	FALSE TRUE	FALSE	JitterGen。DG モードかつ LongDelay=OFF の場合のみ。
DTG_JITTER_MODE_RECID	short	2	FALSE (ALL) TRUE (PART)	FALSE(ALL)	JitterGen。DG モードかつ LongDelay=OFF の場合のみ。
DTG_JITTER_GATINGSOURCE_ RECID	short	2		-1 (None)	JitterGen。DG モードかつ LongDelay=OFF かつ JitterMode=Partial の場合のみ。
DTG_JITTER_PATTERN_RECID	short	2	WFM_SINE WFM_SQUARE WFM_TRIANGLE WFM_GAUSSIAN	WFM_SINE	JitterGen。DG モードかつ LongDelay=OFF の場合のみ。
DTG_JITTER_EDGE_RECID	short	2	EDGE_RISE EDGE_FALL EDGE_BOTH	EDGE_BOTH	JitterGen。DG モードかつ LongDelay=OFF の場合のみ。
DTG_JITTER_FREQUENCY_RECID	double	8		1 MHz	JitterGen。DG モードかつ LongDelay=OFF の場合のみ。

表 C-2：レコード ID-Root （続き）

ID	データ型	サイズ (byte)	パラメータ	初期値	備考
DTG_AMPUNITSETTYPE_RECID	short	2	JITTER_SETUP_UNIT_UI JITTER_SETUP_UNIT_SEC	JITTER_SETUP_UNIT_SEC	JitterGen。SEC または UI を指定。
DTG_JITTER_AMPMODESETTYPE_RECID	short	2	JITTER_SETUP_MODE_PP JITTER_SETUP_MODE_RMS	PP	JitterGen。PP (peek to peak) または RMS(root mean square) を指定。
DTG_JITTER_AMPLITUDE_RECID	double	8		0	JitterGen。DG モードかつ LongDelay=OFF の場合のみ。1 ns p-p は 0.1 UI p-p と同じ。DTG_JITTER_AMPMODESETTYPE_RECID/ DTG_JITTER_AMPUNITSETTYPE_RECID の値により単位が変る。 SEC/PP: s pp SEC/RMS: s rms UI/PP: UI pp UI/RMS: UI rms
DTG_DCOUTPUT_RECID	short	2	FALSE TRUE	FALSE	
DTG_DCOUTPUTTABLE_RECID		26*8 (ch)			1 ch=26 bytes のデータが 8 ch 分隙間なく並んでいる。Load はデータを 26 bytes ずつ先頭から区切って各チャンネルに設定していく。 1 ch 分のデータ (26 bytes) の構成は**注 1** を参照。
DTG_ASSIGN_RECID		(2*5)*Grouping 済み論理チャンネル数			1 ch=10 bytes のデータが Grouping 済み論理チャンネル数分隙間なく並んでいる 1 ch 分のデータ (10 bytes) の構成は**注 2** を参照。

注 1：1 ch 分のデータ構成 (26 bytes)

オフセット	型	初期値	説明
0-7	double	1 V	アウトプット・レベル
8-9	short	FALSE	リミットの ON/OFF
10-17	double	1 V	リミット値 (high)
18-25	double	0 V	リミット値 (low)

注 2：1 ch 分のデータ構成 (10 bytes)

オフセット	型	説明
0-1	short	グループ番号
2-3	short	論理チャンネル番号
4-5	short	アサイン先メインフレーム番号
6-7	short	アサイン先スロット番号
8-9	short	アサイン先チャンネル番号 (アサイン先がなければ -1 がセットされる。)

Group

表 C-3：レコード ID-Group

ID	データ型	サイズ (byte)	パラメータ	初期値	備考
DTG_ID_RECID	short	2		-1	値は、0 以上であれば不問 (最大グループ数を超えても良い)。ただし、Pattern の DTG_GROUPID_RECID および View の DTG_GROUPLIST_RECID および DTG_BYGROUPLIST_RECID のデータと整合性がとれている必要があります。
DTG_NAME_RECID	char	可変		Group 1	NULL を含む。
DTG_NBIT_RECID	short	2		8 またはメインフレーム 1 の実装チャンネル数	DG モードの場合は 8。PG モードの場合は、メインフレーム 1 のスロット A ～ D に実装されているチャンネル数。
DTG_LOGICALCHANNEL_RECID	short	2*nbits			グループに属する論理チャンネル番号 (1 ch=2 bytes) が n bits 分隙間無く並んでいる。
DTG_RADIX_RECID	short	2	VIEW_RADIX_BINARY VIEW_RADIX_OCTAL VIEW_RADIX_DECIMAL VIEW_RADIX_HEX	HEX	
DTG_SIGNED_RECID	short	2	FALSE TRUE	FALSE	
DTG_MAGNITUDE_RECID	short	2	FALSE TRUE	FALSE	

Logical Channel

表 C-4：レコード ID-Logical Channel

ID	データ型	サイズ (byte)	パラメータ	初期値	備考
DTG_ID_RECID	short	2		-1	値は 0 以上であれば不問 (最大チャンネル数を超えてもよい)。ただし、View の DTG_BYCHANNELLIST_RECID のデータと整合性が取れていることが必要。
DTG_TERMZ_RECID	long	4		50 ohm	データ出力 Level。
DTG_TERMV_RECID	double	8		0.0 V	データ出力 Level。
DTG_LIMIT_RECID	short	2	FALSE TRUE	FALSE	データ出力 Level。
DTG_LIMITHIGH_RECID	double	8		1 V	データ出力 Level。
DTG_LIMITLOW_RECID	double	8		0 V	データ出力 Level。
DTG_LEVELSETTYPE_RECID	short	2	LEVEL_SETUP_ MODE_HIGH_LOW LEVEL_SETUP_MOD E_AMP_OFFSET	HIGH_LOW	データ出力 Level。 HIGH_LOW または AMP_OFFSET を指定する。
DTG_LEVELHIGHORAMP_RECID	double	8		1.000 V	データ出力 Level。 DTG_LEVELSETTYPE_RECID の値により値の意味が変る。 HIGH_LOW: V: この値は HIGH を意味する。 AMP_OFFSET: Vpp: この値は AMPLITUDE を意味する。
DTG_LEVELLOWOROFFSET_ RECID	double	8		0.00 V	データ出力 Level。 DTG_LEVELSETTYPE_RECID の値により値の意味が変る。 HIGH_LOW: この値は LOW を意味する。 AMP_OFFSET: この値は OFFSET を意味する。
DTG_OUTPUT_RECID	short	2	FALSE TRUE	FALSE	データ出力 Level。
DTG_POLARITY_RECID	short	2	POLARITY_PLUS POLARITY_MINUS	Normal (POLARITY_ PLUS)	データ出力 Level。
DTG_JITTERRANGE_RECID	short	2	JITTERRANGE_1NS JITTERRANGE_2NS	2NS	Timing。DTGM32 用のパラメータ。
DTG_FORMAT_RECID	short	2	DATA_FORMAT_NRZ DATA_FORMAT_RZ DATA_FORMAT_R1	NRZ	Timing。DG モードのみ。スロット A ～ D のみ有効。
DTG_DELAYUNIT_RECID	short	2	LOCK_LEAD_DELAY LOCK_PHASE LOCK_DUTY LOCK_WIDTH LOCK_TRAIL_DELAY	Lead Delay	Timing。

表 C-4：レコード ID-Logical Channel （続き）

ID	データ型	サイズ (byte)	パラメータ	初期値	備考
DTG_DELAYDATA_RECID	double	8		Lead Delay: 0s Pulse: 0%	Timing。
DTG_WIDTHUNIT_RECID	short	2	LOCK_LEAD_DELAY LOCK_PHASE LOCK_DUTY LOCK_WIDTH LOCK_TRAIL_DELAY	Duty	Timing。 スロット A ～ D のみ有効。
DTG_WIDTHDATA__RECID	double	8		Pulse Width: 5 ns Trail Duty: 5 ns Duty: 50%	Timing。 スロット A ～ D のみ有効。
DTG_SLEWRATE_RECID	double	8		2.25 V/ns	Timing。
DTG_CHMIX_RECID	short	2	CH_MIX_NORMAL CH_MIX_AND CH_MIX_XOR	Normal	Timing。 スロット A ～ D のみ有効。
DTG_DTO_RECID	short	2	FALSE TRUE	FALSE	Timing。
DTG_DTOOFFSET_RECID	double	8		0 ns	Timing。
DTG_PULSERATE_RECID	short	2		1/1	Timing。PG モードのみ。
DTG_CROSS_POINT_RECID	short	2		50%	Timing。NRZ かつ、DLY10 かつ、DTGM30 のチャンネルにのみ設定可。
DTG_JITTERRANGE_RECID	short	2	JITTERRANGE_1NS JITTERRANGE_2NS	JITTERRANGE_2NS	

Block

表 C-5：レコード ID-Block

ID	データ型	サイズ (byte)	パラメータ	初期値	備考
DTG_ID_RECID	short	2		-1	値は 0 以上であれば不問 (最大ブロック数を超えてもよい)。ただし、Pattern の DTG_BLOCKID_RECID および View の DTG_BLOCKLIST_RECID のデータと整合性が取れている必要があります。
DTG_NAME_RECID	char	可変		Block 1	NULL を含む。
DTG_SIZE_RECID	long	4		1000	

Sub Sequence

表 C-6：レコード ID-Sub Sequence

ID	データ型	サイズ (byte)	パラメータ	初期値	備考
DTG_NAME_RECID	char	可変		NULL	NULL を含む。

Sub Sequence Step

表 C-7：レコード ID-Sub Sequence Step

ID	データ型	サイズ (byte)	パラメータ	初期値	備考
DTG_ID_RECID	short	2		-1	値は 0 以上であれば不問 (最大サブシーケンス・ステップ数を超えてもよい)。ただし、View の DTG_SUBSEQLIST_RECID のデータと整合性が取れている必要があります。
DTG_SUBNAME_RECID	char	可変		NULL	Block Name。NULL を含む。
DTG_REPEATCOUNT_RECID	long	4		1	

Main Sequence

表 C-8：レコード ID-Main Sequence

ID	データ型	サイズ (byte)	パラメータ	初期値	備考
DTG_LABEL_RECID	char	可変		NULL	NULL を含む。
DTG_WAITTRIGGER_RECID	short	2		OFF	
DTG_SUBNAME_RECID	char	可変		Block 1	Block/SubSeq Name。NULL を含む。
DTG_REPEATCOUNT_RECID	long	4		Inf.	
DTG_JUMPTO_RECID	char	可変		NULL	Label Name。NULL を含む。
DTG_GOTO_RECID	char	可変		NULL	Label Name。NULL を含む。

Pattern

表 C-9：レコード ID-Pattern

ID	データ型	サイズ (byte)	パラメータ	初期値	備考
DTG_TERMZ_RECID	short	2		-1	Group の DTG_ID_RECID との整合性が取れていることが必要。
DTG_TERMV_RECID	short	2		-1	Block の DTG_ID_RECID との整合性が取れていることが必要。
DTG_LIMIT_RECID	void				

View

表 C-10：レコード ID-View

ID	データ型	サイズ (byte)	パラメータ	初期値	備考
DTG_BLOCKLIST_RECID	short	可変 (2*n)			Block の DTG_ID_RECID との整合性が取れていることが必要。
DTG_GROUPLIST_RECID	short	可変 (2*n)			Group の DTG_ID_RECID との整合性が取れていることが必要。
DTG_BYCHANNELLIST_RECID	short	可変 (2*n)			Logical Channel の DTG_ID_RECID との整合性が取れていることが必要。
DTG_BYGROUPLIST_RECID	short	可変 (2*n)			Group の DTG_ID_RECID との整合性が取れていることが必要。
DTG_SUBSEQLIST_RECID	short	可変 (2* n)			Sub Sequence Step の DTG_ID_RECID との整合性が取れていることが必要。

定数定義

次に、DTG5000 シリーズで使用されているパラメータの定数定義を示します。

```
/*********************************************************************
*
*       Difinitions in DTG5000 file format for V2.0.
*/
/*********************************************************************/

/*****************************
*   DG mode, RunningStateID
*/
    RUN_STATE_STOPPED = 0,
    RUN_STATE_RUNNING = 1,
    RUN_STATE_WAIT_TRIGGER = 2,
    RUN_STATE_CLK_MISSING = 3,
    RUN_STATE_UNKNOWN = 4

/*****************************
*   PG mode, RunModeID
*/
    RUN_MODE_CONTINUOUS = 0,
    RUN_MODE_BURST = 1,

/*****************************
*   ClockSourceID
*/
    CLK_SOURCE_INT = 0,
    CLK_SOURCE_EXT_10MHZ = 1,
    CLK_SOURCE_EXT_PLL = 2,
    CLK_SOURCE_EXT = 3

/*************************
*   ImpedanceID for Trigger and Event
*/
    IMPEDANCE_HI = -1,
    IMPEDANCE_50 = 0,
    IMPEDANCE_1K = 1

/*********************************
*   LevelSettingTypeID
*/
    LEVEL_SETUP_MODE_HIGH_LOW = 0,
    LEVEL_SETUP_MODE_AMP_OFFSET = 1

/*********************************
*   ClockSettingTypeID
*/
```

```
    CLOCK_SETUP_MODE_FREQUENCY = 0,
    CLOCK_SETUP_MODE_PERIOD = 1

/*********************************
*   SettingTypeID in Jitter Amplitude
*/
    JITTER_SETUP_MODE_PP = 0,
    JITTER_SETUP_MODE_RMS = 1

/*********************************
*   SettingTypeID in Jitter Units
*/
    JITTER_SETUP_UNIT_UI = 0,
    JITTER_SETUP_UNIT_SEC = 1

/************************
*   IntExtID
*/
    INTERNAL = 0,
    EXTERNAL = 1

/****************************
*   DataFormatID
*/
    DATA_FORMAT_NRZ = 0,
    DATA_FORMAT_RZ = 1,
    DATA_FORMAT_R1 = 2

/****************************
*   LockTimingID
*/
    LOCK_LEAD_DELAY = 0,
    LOCK_PHASE = 1,
    LOCK_DUTY = 2,
    LOCK_WIDTH = 3,
    LOCK_TRAIL_DELAY = 4

/*********************************
*   ChannelMixedModeID
*/
    CH_MIX_NORMAL = 0,
    CH_MIX_AND = 1,
    CH_MIX_XOR= 2

/*************************
*   PolarityID
*/
    POLARITY_PLUS = 0,
    POLARITY_MINUS = 1
```

```
/*************************
*   JitterRangeID
*/
    JITTERRANGE_1NS = 0,
    JITTERRANGE_2NS = 1

/*************************
*   ViewModeID
*/
    VIEW_MODE_CHANNEL = 0,
    VIEW_MODE_GROUP = 1

/*************************
*   RadixID
*/
    VIEW_RADIX_BINARY = 0,
    VIEW_RADIX_OCTAL = 1,
    VIEW_RADIX_DECIMAL = 2,
    VIEW_RADIX_HEX = 3

/*************************
*   EdgeID
*/
    EDGE_RISE = 0,
    EDGE_FALL = 1,
    EDGE_BOTH = 2

/*************************
*   WaveformID
*/
    WFM_SINE = 0,
    WFM_SQUARE = 1,
    WFM_TRIANGLE = 2,
    WFM_GAUSSIAN = 3

/*************************
*   SequencerModeID
*/
    SEQUENCER_MODE_HW = 0,
    SEQUENCER_MODE_SW = 1

/*************************
*   JumpModeID
*/
    JUMP_MODE_EVENT = 0,
    JUMP_MODE_COMMAND = 1
```

```
/*************************
*   JumpTimingID
*/
    JUMP_TIMING_SYNC = 0,
    JUMP_TIMING_ASYNC = 1

/*************************
*   ClockRangeID
*/
    CLOCK_RANGE_1 = 0,//  50K - 100KHz
    CLOCK_RANGE_2 = 1,// 100K - 200KHz
    CLOCK_RANGE_3 = 2,// 200K - 400KHz
    CLOCK_RANGE_4 = 3,// 250K - 500KHz
    CLOCK_RANGE_5 = 4,// 500K -   1MHz
    CLOCK_RANGE_6 = 5,//   1M -   2MHz
    CLOCK_RANGE_7 = 6,//   2M -   4MHz
    CLOCK_RANGE_8 = 7,//   2.5M - 5MHz
    CLOCK_RANGE_9 = 8,//   5M -  10MHz
    CLOCK_RANGE_10 = 9,//  10M -  20MHz
    CLOCK_RANGE_11 = 10,//  20M -  40MHz
    CLOCK_RANGE_12 = 11,//  25M -  50MHz
    CLOCK_RANGE_13 = 12,//  50M - 100MHz
    CLOCK_RANGE_14 = 13,// 100M - 200MHz
    CLOCK_RANGE_15 = 14,// 200M - 400MHz
    CLOCK_RANGE_16 = 15// 400MHz <

/*************************
*   ClockRangePGID
*/
    CLOCK_RANGE_PG_1 = 0,//  50K - 100KHz
    CLOCK_RANGE_PG_2 = 1,// 100K - 200KHz
    CLOCK_RANGE_PG_3 = 2,// 200K - 400KHz
    CLOCK_RANGE_PG_4 = 3,// 250K - 500KHz
    CLOCK_RANGE_PG_5 = 4,// 500K -   1MHz
    CLOCK_RANGE_PG_6 = 5,//   1M -   2MHz
    CLOCK_RANGE_PG_7 = 6,//   2M -   4MHz
    CLOCK_RANGE_PG_8 = 7,//   2.5M - 5MHz
    CLOCK_RANGE_PG_9 = 8,//   5M -  10MHz
    CLOCK_RANGE_PG_10 = 9,//  10M -  20MHz
    CLOCK_RANGE_PG_11 = 10,//  20M -  40MHz
    CLOCK_RANGE_PG_12 = 11,//  25M -  50MHz
    CLOCK_RANGE_PG_13 = 12,//  50M - 100MHz
    CLOCK_RANGE_PG_14 = 13,// 100M - 200MHz
    CLOCK_RANGE_PG_15 = 14// 200M <
```

ファイル・ロードについて

同一の中間ノード下に同じ ID を持つリーフ・ノードが存在する場合、最初に現れたリーフ・ノードのみがロードされ、それ以降のリーフ・ノードは読み飛ばされます。たとえば、DTG_GROUP_RECID の下に DTG_NAME_RECID = “Group Name 1” と DTG_NAME_RECID = “Group Name 2” がこの順序で存在した場合、ロードされるグループ名は “Group Name 1” になります。また、同じ ID の中間ノードについては、各中間ノードの上限値を超えたものについては、読み飛ばされます。たとえば、DTG_SUBSEQUENCE_RECID は先頭から 50 個目まではロードされますが、51 個目は読み飛ばされ、ロードされません。

PG モードでのファイル・ロードおよびデフォルト・セットアップ

ファイルがロードされるときに、オペレーション・モードが PG に設定されている場合、Block/Sequence/Subsequence/Pattern/View のロードは行なわれません。また、デフォルト・セットアップの際に、オペレーション・モードが PG に設定されている場合、Block/Sequence/Subsequence/Pattern のセットアップは行なわれません。したがって、PG モードでの Block/Sequence/Subsequence は空になります。

レコード ID の出現順序

ファイル内でのレコード ID の出現順序は、同一の中間ノード内であればロード時の動作に影響しません。たとえば、DTG_LOGICALCH_RECID、DTG_TERMZ_RECID の順でも、またその逆でもファイル・ロード時の動作は同じです。

チャンネル・アサインについて

ここでは、各オーペレーション・モードにおいて、どのようにチャンネル・アサインが行なわれるかについて説明します。

デフォルト時のチャンネル・アサインおよびグループ作成

- オペレーション・モードが DG の場合：グループは以下の計算式により作成されます。

  グループ数＝ (実装されているチャンネル数 + (8-1)) ÷ 8

  グループは、実装されているチャンネル数を 8 チャンネル 1 グループとして作成されます。残りのチャンネル (1 ～ 7) は、まとめて 1 グループとして作成されます。実装チャンネルがない場合、グループは作成されません。チャンネル・アサインは、以下の順序で行ないます。

  グループ：Group1, Group2, Group3, ・・・の順
  グループ・ビット：MSB から LSB
  物理チャンネル：メインフレーム 1 のスロット A のチャンネル 1 からメインフレーム 3 のスロット H のチャンネル 4 の順

未アサインのチャンネルは存在しません。

■ オペレーション・モードが PG の場合：

グループは、メインフレーム 1 のスロット A ～ D に実装されているチャンネル数を 1 グループとして作成されます。スロット A ～ D に何も実装されていない場合、グループは作成されません。チャンネル・アサインは、グループ・ビットの MSB から LSB の順、スロット A のチャンネル 1 からスロット D のチャンネル 4 の順にアサインされます。未アサインのチャンネルは存在しません。

ファイル・ロード時のチャンネル・アサイン

■ オペレーション・モードが DG の場合：

チャンネルは、ロードされたファイルに従ってアサインされます。アサイン先の物理チャンネルが存在しない場合は、次の項目のようになります。

■ オペレーション・モードが PG の場合：

グループ・ビットの MSB から LSB の順、スロット A のチャンネル 1 からスロット D のチャンネル 4 の順にアサインされます。
グループ・ビット数＞物理チャンネル数の場合：グループ・ビットの LSB 側は無視されます。
グループ・ビット数＜物理チャンネル数の場合：スロット D のチャンネル 4 側の物理チャンネルはデフォルト設定されます。
未アサインのチャンネルは存在しません。

ファイル・ロード時にアサイン先が存在しない場合

ファイル・ロード時にアサイン先の物理チャンネルが存在しない場合は、デアサインされます。

オペレーション・モードが DG に設定されているときにデアサインが発生した場合は、警告メッセージが表示されます（DbLoadSetup() はリターン・コードで DBERR_WAR_DISCONNECT を返します）。

たとえば、次のような場合を仮定します。
セーブした状態：DTG5078 型で、スロット A/B の全チャンネルにアサイン (スロット A/B のアウトプット・モジュールには DTGM20 型がインストールされている）
ロード先構成 (1)：DTG5078 型で、スロット A/B にのみアウトプット・モジュールがインストールされている（スロット A/B のアウトプット・モジュールは DTGM30 型）。
ロード先構成 (2)：DTG5078 型で、スロット A/B にのみアウトプット・モジュールがインストールされている（スロット A/B のアウトプット・モジュールは DTGM10 型）。
この場合、ロード先構成 (1) はロードに成功しますが、「デアサイン発生」を示す警告メッセージが表示されます。ロード先構成 (2) はロードに成功します。

オペレーション・モードが PG の場合、デアサインは発生しません。

保証規定

保証期間 (納入後 1 年間) 内に、通常の取り扱いによって生じた故障は無料で修理いたします。

1. **取扱説明書、本体ラベルなどの注意書きに従った正常な使用状況で保証期間内に故障した場合には、販売店または当社に修理をご依頼下されば無料で修理いたします。なお、この保証の対象は製品本体に限られます。**
2. **転居、譲り受け、ご贈答品などの場合で販売店に修理をご依頼できない場合には、当社にお問い合せください。**
3. **保証期間内でも次の事項は有料となります。**
   - **使用上の誤り、他の機器から受けた障害、当社および当社指定の技術員以外による修理、改造などから生じた故障および損傷の修理**
   - **当社指定外の電源 (電圧・周波数) 使用または外部電源の異常による故障および損傷の修理**
   - **移動時の落下などによる故障および損傷の修理**
   - **火災、地震、風水害、その他の天変地異、公害、塩害、異常電圧などによる故障および損傷の修理**
   - **消耗品、付属品などの消耗による交換**
   - **出張修理 (ただし故障した製品の配送料金は、当社負担)**
4. **本製品の故障またはその使用によって生じた直接または間接の損害について、当社はその責任を負いません。**
5. **この規定は、日本国内においてのみ有効です。**(This warranty is valid only in Japan.)

- **この保証規定は本書に明示された条件により無料修理をお約束するもので、これによりお客様の法律上の権利を制限するものではありません。**
- **ソフトウェアは、本保証の対象外です。**
- **保証期間経過後の修理は有料となります。詳しくは、販売店または当社までお問い合せください。**

お問い合わせ

製品についてのご相談・ご質問につきましては、下記までお問い合わせください。

お客様コールセンター

TEL 03-6714-3010　FAX 0120-046-011

東京都港区港南 2-15-2　**品川インターシティ** B **棟** 6F　〒 108-6106

電話受付時間／ 9:00 ～ 12:00　13:00 ～ 19:00　**月曜～金曜 (休祝日を除く)**

E-mail: ccc.jp@tektronix.com

URL:　http://www.tektronix.co.jp

修理・校正につきましては、お買い求めの販売店または下記サービス受付センターまでお問い合わせください。

（ご連絡の際に、型名、故障状況等を簡単にお知らせください。）

サービス受付センター

TEL 0120-741-046　FAX 0550-89-8268

静岡県御殿場市神場 143-1　〒 412-0047

電話受付時間／ 9:00 ～ 12:00　13:00 ～ 19:00 **月曜～金曜 (休祝日を除く)**

プログラマ・マニュアル
DTG5078 **型** /DTG5274 **型** /DTG5334 **型**
データ・タイミング・ゼネレータ
(P/N 071-1614-00)

● 2004 **年** 12 **月　初版発行**
● **仕様は予告無く変更する場合がありますので、あらかじめご了承ください。**