User's
Manual

DL6000/DLM6000 シリーズ
ディジタルオシロスコープ /
ミックスドシグナルオシロスコープ
シリアルバス信号 トリガ / 解析機能

(I^2C バス信号 /CAN バス信号 /LIN バス信号 /SPI バス信号 /UART 信号のトリガ機能、解析機能)

YOKOGAWA
横河電機株式会社

IM DLM6054-51JA
初版

はじめに

このたびは、シリアルバストリガ機能 / 解析機能オプション付き DL6000/DLM6000 をお買い上げいただきましてありがとうございます。このユーザーズマニュアルは、シリアルバストリガ機能 / 解析機能 [*1] について説明しています。

*1 オプションによって、トリガ機能や解析機能の対象信号の種類が異なります。

/F3 オプション	I^2C バス信号、SPI バス 信号、および UART 信号のトリガ機能と解析機能付き
/F4 オプション	CAN バス信号、LIN バス信号、および UART 信号解析機能付き

その他の機能、操作方法、取り扱い上の注意などについては、以下の取扱説明書をご覧ください。

マニュアル名	マニュアル No.	内容
DL6000/DLM6000 シリーズ ディジタルオシロスコープ / ミックスドシグナルオシロスコープ ユーザーズマニュアル	IM DLM6054-01JA	DL6000/DLM6000 シリーズの通信機能を除く全機能とその操作方法について説明しています。
DL6000/DLM6000 シリーズ ディジタルオシロスコープ / ミックスドシグナルオシロスコープ 通信インタフェース ユーザーズマニュアル (CD 内)	IM DLM6054-17JA	DL6000/DLM6000 シリーズの通信インタフェースの機能について、その操作方法を説明しています。
DL6000/DLM6000 シリーズ ディジタルオシロスコープ / ミックスドシグナルオシロスコープ 電源解析機能ユーザーズマニュアル	IM DLM6054-61JA	オプションの電源解析の各機能と操作について説明しています。

ご注意

- 本書の内容は、性能・機能の向上などにより、将来予告なしに変更することがあります。また、実際の画面表示内容が本書に記載の画面表示内容と多少異なることがあります。
- 本書の内容に関しては万全を期していますが、万一ご不審の点や誤りなどお気づきのことがありましたら、お手数ですが、お買い求め先か、当社支社・支店・営業所までご連絡ください。
- 本書の内容の全部または一部を無断で転載、複製することは禁止されています。

1st Edition: Octorber 2009(YK)

商標

- DLM は横河電機株式会社の登録商標です。
- Adobe、Acrobat、および PostScript は、アドビシステムズ社の商標または登録商標です。
- 本文中の各社の登録商標または商標には、TM、® マークは表示していません。
- その他、本文中に使われている会社名、商品名は、各社の登録商標または商標です。

履歴

- 2009 年 10 月　　初版発行

このマニュアルで使用している記号と表記法

注記

このマニュアルでは、注記を以下のようなシンボルで区別しています。

Note 本機器を取り扱ううえで重要な情報が記載されています。

操作説明のページで使用している表記法

第 2 ～ 4 章で操作説明をしているページでは、説明内容を区別するために、次のような表記法を使用しています。

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

解　説 操作に関連する設定内容や限定事項について説明しています。

文字の表記法

太文字の操作キー名とソフトキー名
操作対象になるパネル上の操作キーの文字や、画面に表示されるソフトキー / メニューの文字を示します。

SHIFT+ 操作キー
SHIFT キーを押して、SHIFT キーのインジケータを点灯させてから、操作キーを押すという意味です。押した操作キーの下に紫色で記されている項目の設定メニューが画面に表示されます。

単位

k 「1000」の意味です。使用例：100kS/s(サンプルレート)
K 「1024」の意味です。使用例：720K バイト (ファイルのデータサイズ)

通信コマンドの記述

7 章の通信コマンドの詳細説明では、DLM6000 専用のコマンドを、青い斜体文字で記述しています。DL6000 では使用できないコマンドですので、ご注意ください。

目次

1.1　I²C バス信号

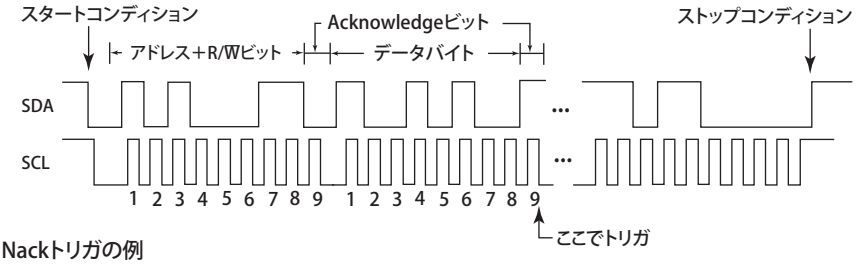
I²C バスとは、Inter Integrated Circuit Bus の略称で、IC 間の相互通信を目的とした双方向バスです。本機能を使うことにより、I²C バス信号の波形を表示しながら、データを解析できます。I²C バス信号解析機能には、大別すると次の 3 つの機能があります。

トリガ　<< 操作説明は 3.1 節 >>

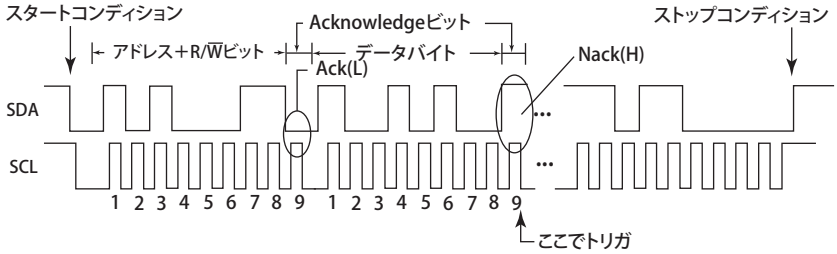
以下の条件でトリガをかけることができます。

- スタートコンディションを検出したとき
- Nack を検出したとき
- 設定したアドレス (7 ビットアドレス、7 ビットアドレス＋サブアドレス、10 ビットアドレス) パターンと一致したとき、
- データパターンと一致 / 不一致したとき
- ジェネラルコールアドレスと一致したとき

アドレス&データトリガの例
・アドレスパターン/データパターンでトリガをかける場合



Nackトリガの例
・Acknowledgeビットがないとき(SDA信号がHのとき)にトリガをかける場合



I²C バス信号のトリガ条件とアナログ信号のトリガ条件を組み合わせて、トリガをかけることもできます (Event Interval トリガ、B トリガ)。Event Interval トリガ、B トリガについての詳細は、ユーザーズマニュアル IMDLM6054-01JA の 6.11 節、6.12 節をご覧ください。

解析　<< 操作説明は 4.2 節 >>

I²C バス信号のデータを解析し、解析結果をリスト表示します。解析結果リストには、Simple(簡易表示) と Detail(詳細表示) の 2 種類があります。Simple では、解析番号、スタートコンディション / ストップコンディション、解析データ、アドレス / データの種類、Read/Write 信号、Acknowledge ビットの状態をバイトごとにリスト表示します。Detail では、Simple の項目に加えて、トリガポジションからの時間とデータ情報をリスト表示します。Detail のデータは、任意のストレージメディアに CSV 形式で保存できます。

また、解析結果リスト上の任意のバイトを選択し、そのバイトの先頭にズーム位置 (ズームボックスの中心) を移動することができます (ズームリンク)。

検索　<< 操作説明は 5.2 節 >>

I²C バス信号のデータに対して、特定のアドレスパターン / データパターン /Acknowledge ビットの状態と一致するデータを検索できます。検索を実行すると、条件と一致したデータにズームボックスが移動し、ズームウインドウ (Zoom1 または Zoom2) にデータを拡大表示します。

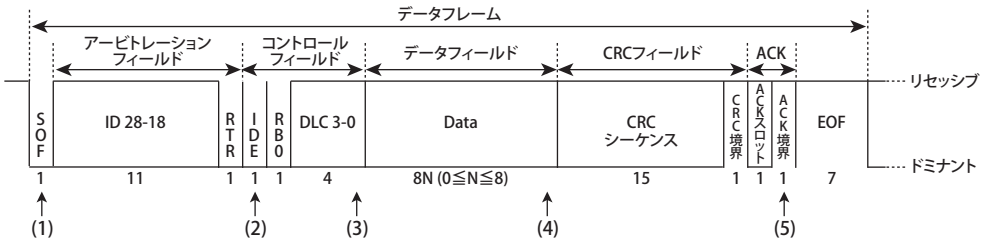
1.2 CAN バス信号

CAN とは、Controller Area Network の略称で、ISO(International Organization for Standardization) にて国際的に標準化されたシリアル通信プロトコルです。
本機能を使うことにより、CAN バス信号の波形をアナログ波形として表示しながらデータを解析できます。
CAN バス信号解析機能には、大別すると次の 3 つの機能があります。

トリガ　<< 操作説明は 3.2 節 >>

CAN バスの ID のビットパターン、DLC、Data、ACK スロットの状態を設定して、特定のデータフレーム / リモートフレームをトリガ条件としてトリガをかけることができます。最大 4 種類の ID/Data の条件を設定できるので、これらの OR 条件でトリガをかけることもできます。また、SOF(Start of Frame) やエラーフレームをトリガ条件にすることも可能です。

[例] データフレーム(標準フォーマット)の場合



(1) SOFをトリガ条件にしたときのトリガポイント
(2) IDのビットパターンだけをトリガ条件にしたときのトリガポイント
(3) IDのビットパターンとDLCをトリガ条件にしたときのトリガポイント
(4) IDのビットパターンとDataのビットパターンをトリガ条件にしたときのトリガポイント
(5) ACKスロットの状態をトリガ条件にしたときのトリガポイント

CAN バス信号のトリガ条件とアナログ信号のトリガ条件を組み合わせて、トリガをかけることもできます (Event Interval トリガ、B トリガ)。Event Interval トリガ、B トリガについての詳細は、ユーザーズマニュアル IMDLM6054-01JA の 6.11 節、6.12 節をご覧ください。

解析　<< 操作説明は 4.3 節 >>

CAN バス信号のデータを解析し、解析結果をリスト表示します。解析結果リストには、Simple(簡易表示) と Detail(詳細表示) の 2 種類があります。Simple では、解析番号、解析したフレームの種類、ID、Data、ACK スロットの状態をフレームごとにリスト表示します。Detail では、Simple の項目に加えて、トリガポジションからの時間、DLC、CRC シーケンスをリスト表示します。解析結果のデータは、任意のストレージメディアに CSV 形式で保存できます。また、解析結果リスト上の任意のフレームを選択し、そのフレームに対応した CAN バス信号を自動的に表示させることができます (ズームリンク)。さらに、フレームの指定したフィールドの先頭にズーム位置 (ズームボックスの中心) をジャンプさせることもできます (フィールドジャンプ)。

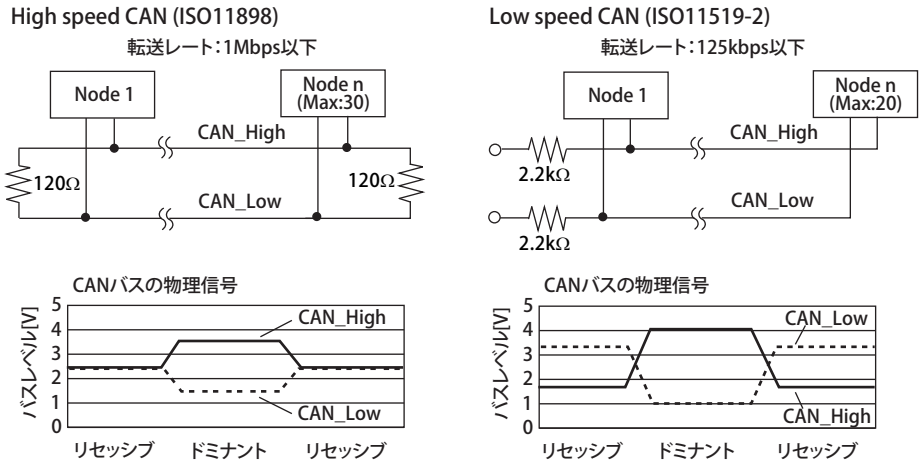
検索　<< 操作説明は 5.3 節 >>

CAN バス信号のデータに対して、特定のフレームやフィールドを検索できます。検索を実行すると、条件と一致したデータにズームボックスが移動し、ズームウインドウ (Zoom1 または Zoom2) にデータを拡大表示します。

High speed CAN (ISO11898) と Low speed CAN (ISO11519-2)

CAN の物理層の代表的な規格として、High speed CAN (ISO11898) と Low speed CAN (ISO11519-2) があります。

下図のように、High speed CAN/Low speed CAN のどちらの場合でも、2 本のバス (CAN_High と CAN_Low) の電位差によってバスのレベルを判断します。



プローブの接続方法

使用するプローブ

CAN バス信号を測定する場合は、差動プローブを使用します。

使用可能な差動プローブ：当社製 701920、701922、701924

リセッシブの電圧レベル ＞ ドミナントの電圧レベルで表示する場合 (Recessive：H)

- **2 線式 (差動) のとき**

  差動プローブの「−」を CAN_High に、「＋」を CAN_Low に接続します。

- **1 線式 (シングルエンド) のとき**

  差動プローブの「−」を CAN_High に、「＋」を GND(接地電位) に接続します。

リセッシブの電圧レベル ＜ ドミナントの電圧レベルで表示する場合 (Recessive：L)

- **2 線式 (差動) のとき**

  差動プローブの「−」を CAN_Low に、「＋」を CAN_High に接続します。

- **1 線式 (シングルエンド) のとき ***

  差動プローブの「−」を GND(接地電位) に、「＋」を CAN_High に接続します。

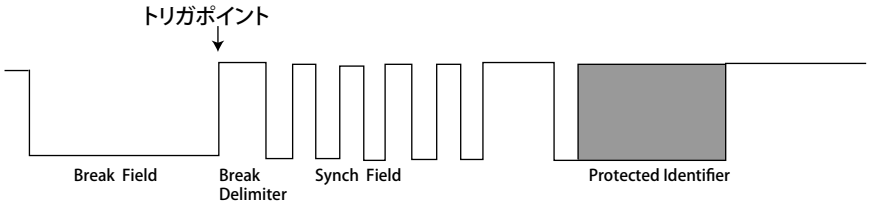
  * この場合、パッシブプローブ (形名：701939) を CAN_High に接続して使用することができます。

1.3 LIN バス信号

LIN とは、Local Interconnect Network の略称で、主に自動車などに使われるシリアル通信プロトコルです。
本機能を使うことにより、LIN バス信号の波形をアナログ波形として表示しながらデータを解析できます。
LIN バス信号解析機能には、大別すると次の 3 つの機能があります。

トリガ　<< 操作説明は 3.3 節 >>

Break delimiter の立ち上がりでトリガをかけます。ビットレートは 19200bps、9600bps、4800bps、2400bps、1200bps、または User から選択できます。



LIN バス信号のトリガ条件と CAN バス信号のトリガ条件を組み合わせたり、LIN バス信号のトリガ条件とアナログ信号のトリガ条件を組み合わせて、トリガをかけることもできます (Event Interval トリガ、B トリガ)。Event Interval トリガ、B トリガについての詳細は、ユーザーズマニュアル IMDLM6054-01JA の 6.11 節、6.12 節をご覧ください。

解析　<< 操作説明は 4.4 節 >>

LIN バス信号のデータを解析し、解析結果をリスト表示します。解析結果リストには、Simple(簡易表示) と Detail(詳細表示) の 2 種類があります。Simple では、解析番号、ID、Data、Checksum の状態をリスト表示します。Detail では、Simple の項目に加えて、トリガポジションからの時間、ID Field、ID Parity エラーと Checksum エラーをリスト表示します。解析結果のデータは、任意のストレージメディアに CSV 形式で保存できます。また、解析結果リスト上の任意のフィールドを選択し、そのフィールドに対応した LIN バス信号を自動的に表示させることができます (ズームリンク)。

検索　<< 操作説明は 5.4 節 >>

LIN バス信号のデータに対して、特定のフィールドを検索できます。検索を実行すると、条件と一致したデータにズームボックスが移動し、ズームウインドウ (Zoom1 または Zoom2) にデータを拡大表示します。

1.4 SPI バス信号

SPI とは、Serial Peripheral Interface の略称で、SPI バスは、IC 間通信やデータ通信などで広く採用されている同期式シリアルバスです。
本機能を使うことにより、SPI バス信号の波形を表示しながらデータを解析できます。
SPI バス信号解析機能には、大別すると次の 3 つの機能があります。

トリガ　<< 操作説明は 3.4 節 >>

入力信号に対し、1 バイト (8 ビット) を最小単位として、バイトごとに設定した条件と比較することで SPI バス信号を取り込むことができます。
比較するデータ位置を、チップセレクト信号 (CS) がアサートされてからのバイト数で指定できます。4 線式 SPI では 2 種類のデータパターン (Data 1、Data 2)、3 線式 SPI では 1 種類のデータパターンを設定できます。Data 1 と Data 2 では、あとからデータパターンが一致した位置でトリガがかかります。CS アサート後、4 バイト目から Data 1 を比較 (3 バイト分) し、さらに CS アサート後、10 バイト目から Data 2 を比較 (4 バイト分) して、両方のパターンが一致したときにトリガをかける場合の例を以下に示します。

CS:	Lowでアクティブ	
Data1	Condition:	True
	Data Size:	3
	Pattern:	121
	Data Position:	4
Data2	Condition:	True
	Data Size:	4
	Pattern:	FFFF
	Data Position:	10

SPI バス信号のトリガ条件とアナログ信号のトリガ条件を組み合わせて、トリガをかけることもできます (Event Interval トリガ、B トリガ)。Event Interval トリガ、B トリガについての詳細は、ユーザーズマニュアル IMDLM6054-01JA の 6.11 節、6.12 節をご覧ください。

解析 << 操作説明は 4.5 節 >>

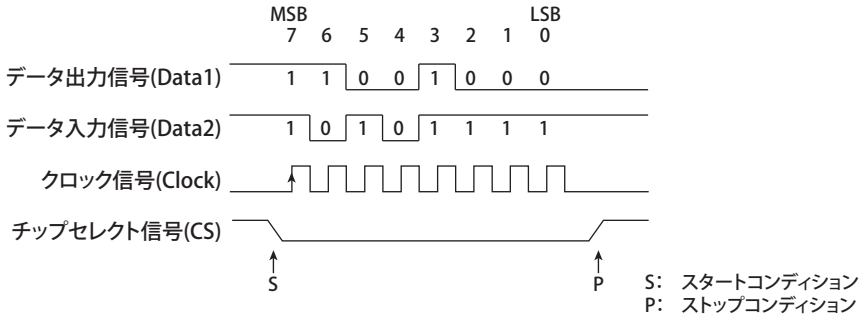
SPI バス信号のデータを解析し、解析結果をリスト表示します。解析は、クロック信号 (Clock) に同期して、フィールドサイズ (Field Size) と有効ビット範囲 (Enable MSB/LSB) で指定したビット単位で行われます。解析結果リストには、Simple(簡易表示) と Detail(詳細表示) の 2 種類があります。Simple では、解析番号、Data 1/Data 2(16 進数表示)、CS のステータスをバイトごとにリスト表示します。Detail では、Simple の項目に加えて、トリガポジションからの時間とアクティブ期間の開始位置 / 終了位置と Data 1/Data 2(2 進数表示) をリスト表示します。Detail のデータは、任意のストレージメディアに CSV 形式で保存できます。
また、解析結果リスト上の任意のバイトを選択し、そのバイトの先頭にズーム位置 (ズームボックスの中心) を移動することができます (ズームリンク)。

検索 << 操作説明は 5.5 節 >>

SPI バス信号のデータに対して、特定のデータパターンと一致するデータ / 一致しないデータを検索できます。検索を実行すると、条件と一致したデータにズームボックスが移動し、ズームウインドウ (Zoom1 または Zoom2) にデータを拡大表示します。

解析 / 検索例

下図の信号について解析 / 検索を実行すると、フィールドサイズ (Field Size) と有効ビット範囲 (Enable MSB/LSB) の設定に従い、下表のように解析 / 検索します。



解析 / 検索の条件

- データ区間は 8 ビット
- ビットオーダーは MSB First

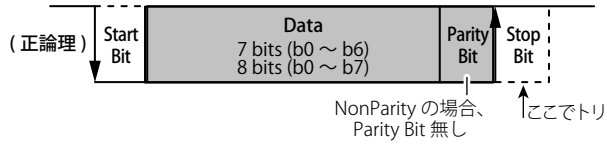
解析条件		解析結果		
Field Size	Enable MSB/LSB	Data1	Data2	CS
4bit	3 ～ 0 (4bit)	C, 8	A, F	L
		解析： データ区間を 4bit 単位で 2 回解析します。 検索： 比較開始フィールドから 4bit を検索します。		
6bit	5 ～ 0 (6bit)	3, 2	2, B	L
		解析： データ区間を 6bit 単位で 1 回解析します。(2bit,4bit でデコード) 検索： 比較開始フィールドから 6bit を検索します。		
8bit	5 ～ 0 (6bit)	0, 8	2, F	L
		解析： データ区間を 8bit 単位で下位 6bit を 1 回解析します。(2bit,4bit でデコード) 検索： 比較開始フィールドから 8bit 中下位 6bit を検索します。		
12bit		解析 / 検索しません。		

1.5 UART 信号

UART とは、Universal Asynchronous Receiver-Transmitter の略で、シリアル転送方式のデータとパラレル転送方式のデータを相互に変換するチップのことです。UART は、一般に、EIA RS-232 のような機器間の通信接続に利用されています。
本機能を使うことにより、UART 信号をアナログ波形として表示しながらデータを解析できます。
UART 信号解析機能には、大別すると次の 3 つの機能があります。

トリガ　<< 操作説明は 3.5 節 >>

すべてのデータの Stop Bit の位置でトリガがかかります。ビットレートは 115200bps、57600bps、38400bps、19200bps、9600bps、4800bps、2400bps、1200bps、または User から選択できます。



UART 信号のトリガ条件とアナログ信号のトリガ条件を組み合わせて、トリガをかけることもできます (B トリガ)。B トリガについての詳細は、ユーザーズマニュアル IMDLM6054-01JA の 6.12 節をご覧ください。

解析　<< 操作説明は 4.6 節 >>

UART 信号のデータを解析し、解析結果をリスト表示します。解析結果リストには、Simple(簡易表示) と Detail(詳細表示) の 2 種類があります。Simple では、解析番号、Data、エラーの状態をリスト表示します。Detail では、Simple の項目に加えて、トリガポジションからの時間をリスト表示します。解析結果のデータは、任意のストレージメディアに CSV 形式で保存できます。また、解析結果リスト上の任意のデータを選択し、そのデータに対応した UART 信号を自動的に表示させることができます (ズームリンク)。

検索　<< 操作説明は 5.6 節 >>

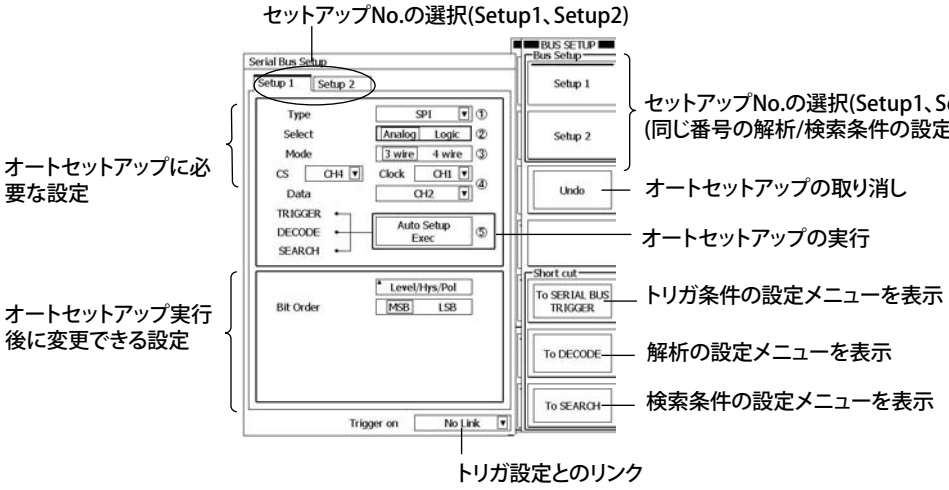
UART 信号のデータに対して、特定のデータパターンやエラーでデータを検索できます。検索を実行すると、条件と一致したデータにズームボックスが移動し、ズームウインドウ (Zoom1 または Zoom2) にデータを拡大表示します。

2.1　シリアルバス信号のセットアップをする

操　作

SETUP_Serial Bus Setup メニュー

SETUP キー >Serial Bus Setup のソフトキーを押します。次のメニューが表示されます。



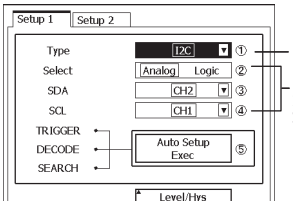
Note

Trigger on を Setup 1 または Setup 2 に設定すると、トリガの設定が、Setup 1 または Setup 2 で設定されているシリアルバストリガに変更されます。

オートセットアップに必要な設定

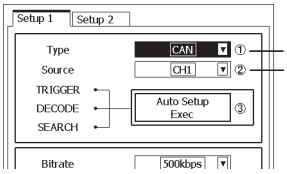
シリアルバス信号の種類によって選定内容が異なります。

I2C



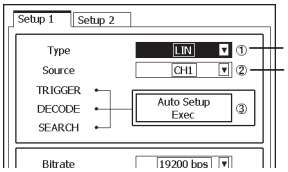
I2C を選択
SDA/SCL のソースの選択 (DL6000 シリーズの Select は Analog に固定 (表示しない))

CAN



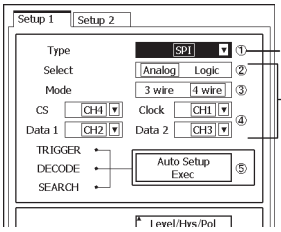
CAN を選択
ソースの選択

LIN



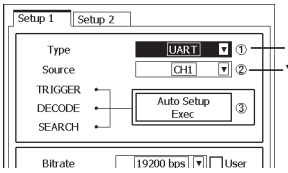
LIN を選択
ソースの選択

SPI



SPI を選択
線式、CS/Clock/ データのソースの選択 (DL6000 シリーズの Select は Analog に固定 (表示しない))

UART



UART を選択
ソースの選択

Note

Source に M1 ～ M4(演算波形) を選択した場合は、オートセットアップは実行できません。
トリガ、解析、検索条件の設定は共通です。

オートセットアップの実行

Auto Setup Exec を選択して、**SET** を押します。

Note

シリアルバス信号のタイプに SPI を選択して、CS のソースに None を選択した場合、オートセットアップは実行できません。

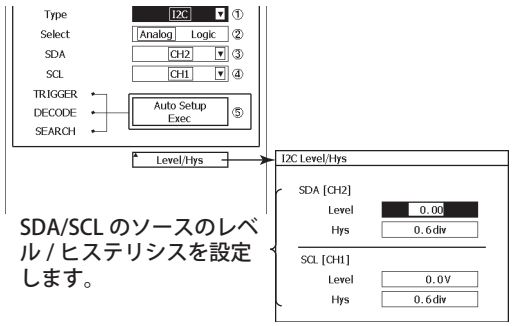
オートセットアップの取り消し

Undo のソフトキーを押します。オートセットアップ直前の設定に戻ります。

設定値の変更 (オートセットアップ後)

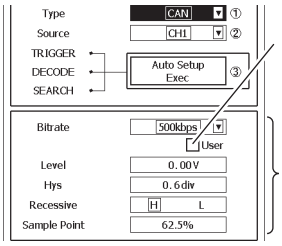
シリアルバス信号のタイプによって、変更できる項目が異なります。

I2C



SDA/SCL のソースのレベル / ヒステリシスを設定します。

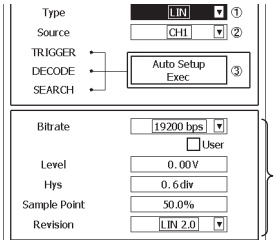
CAN



ビットレートを任意に設定する場合はここをチェック

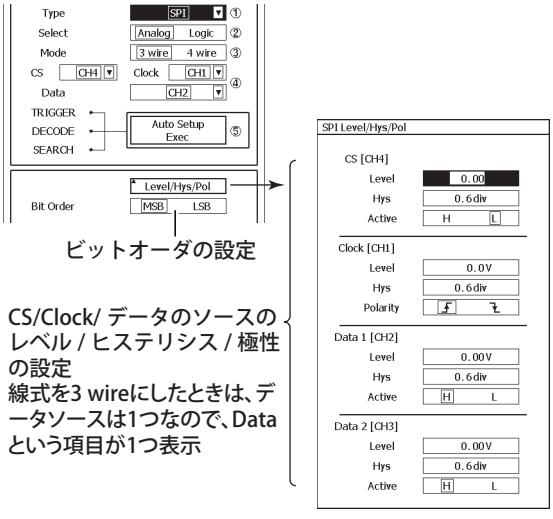
ビットレート / レベル / ヒステリシス / リセッシブ / サンプルポイントの設定

LIN



ビットレート / レベル / ヒステリシス / サンプルポイント / レビジョンの設定

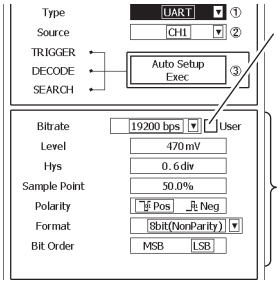
SPI



ビットオーダの設定

CS/Clock/ データのソースのレベル / ヒステリシス / 極性の設定
線式を3 wireにしたときは、データソースは1つなので、Dataという項目が1つ表示

UART



ビットレートを任意に設定する場合はここをチェック

ビットレート / レベル / ヒステリシス / サンプルポイント / 極性 / フォーマット / ビットオーダの設定

Note

オートセットアップに必要な設定以外の設定は、オートセットアップを実行すると、本機器が最適と判断した値または初期値に変更される場合があります。オートセットアップ実行した場合は、設定内容を確認してください。

解　説

I²C、CAN、LIN、SPI、および UART の各シリアルバス信号のオートセットアップができます。
オートセットアップを実行して、シリアルバス信号を認識したときは、入力信号に適した値が自動的にトリガ / 解析 / 検索の各機能の設定値に反映されます。

セットアップ No(Setup 1/Setup 2)

2 つのセットアップ条件を設定できます。セットアップ条件を、同じ番号の解析 (Analysis 1/Analysis 2) と検索条件 (Search 1/Search 2) に反映します。詳細は 2.2 節をご覧ください。
また、どちらかのセットアップ条件をトリガ条件に反映させることもできます。

オートセットアップをするときに必要な設定項目

ソース

オートセットアップを実行する対象信号 * を、シリアルバス信号のタイプに合わせて選択します。

I²C	SDA(シリアルデータ) と SCL(シリアルクロック) の各ソースを選択します。 Select 欄で Analog を選択したときは、CH1 ～ CH4 から選択します。 Select 欄で Logic を選択したときは、A0 ～ A7、B0 ～ B7、C0 ～ C7、および D0 ～ D7(16 ビットモデルは A0 ～ A7 と C0 ～ C7) から選択します。
CAN	CH1 ～ CH4、M1 ～ M4 から選択します。
LIN	CH1 ～ CH4、または A0 ～ A7、B0 ～ B7、C0 ～ C7、および D0 ～ D7(16 ビットモデルは A0 ～ A7 と C0 ～ C7) から選択します。
SPI	CS(チップセレクト)、Clock(クロック)、およびデータの各ソースを選択します。 Select 欄で Analog を選択したときは、CH1 ～ CH4、M1 ～ M4 から選択します。 Select 欄で Logic を選択したときは、A0 ～ A7、B0 ～ B7、C0 ～ C7、および D0 ～ D7(16 ビットモデルは A0 ～ A7 と C0 ～ C7) から選択します。
UART	CH1 ～ CH4、または A0 ～ A7、B0 ～ B7、C0 ～ C7、および D0 ～ D7(16 ビットモデルは A0 ～ A7 と C0 ～ C7) から選択します。

* ソースとして、A0 ～ A7、B0 ～ B7、C0 ～ C7、および D0 ～ D7 を選択したときは、スレショルドレベルを設定してください。設定操作については、ユーザーズマニュアル IM DLM6054-01JA-01 の 5.18 節をご覧ください。
また、ソースとして M1 ～ M4 を選択したときは、オートセットアップは実行できません。

線式

シリアルバス信号が SPI のときだけ、線式を選択します。

3 wire	Data ラインが 1 つ
4 wire	Data ラインが 2 つ

Note

設定した内容は、オートセットアップを実行しなくても、解析 / 検索条件に反映されます。詳細は 2.2 節をご覧ください。

トリガ設定とのリンク (Trigger on)

Setup 1 または Setup 2 の設定をトリガ設定とリンクできます。Trigger on の設定を Setup 1 または Setup 2 に変更すると同時に、トリガ設定が変更されます。
また、その状態でセットアップ条件を変更すると、変更内容がトリガ設定に反映されます。
オートセットアップを実行すると、Trigger on が実行したセットアップ No に設定され、オートセットアップ結果がトリガ設定に反映されます。

Note

- 直接トリガ設定を変更した場合は、Trigger on は No Link に変更され、リンクが解除されます。
- トリガが Edge トリガに設定されている場合でも、Trigger on を Setup 1 または Setup 2 に設定すると、Enhanced トリガに変更されます。

オートセットアップの実行

入力信号に合わせてオートセットアップを実行します。
セットアップした結果が、解析 / 検索条件に反映されます。詳細は 2.2 節をご覧ください。
オートセットアップを実行すると、セットアップ結果がトリガ設定に反映され、Trigger on が、実行したセットアップ No に設定されます。シリアルバス信号を認識できないときは、エラーメッセージが表示されます。

オートセットアップが可能な信号

シリアルバス信号で下記の条件を満たしている場合に、オートセットアップが可能です。

電圧	200mV 以上の振幅 (プローブの減衰比を 1：1 に設定したとき)
ビットレート	1200bps 以上
フレーム数	10 秒間に、少なくとも 5 フレーム以上

Note

- プローブの減衰比が正しく設定されていないと、正しい測定ができません。オートセットアップを実行する前にプローブの減衰比をご確認いただき、正しく設定してください。設定操作については、ユーザーズマニュアル IM DLM6054-01JA の 6.6 節をご覧ください。
- ソースとして、A0 ～ A7、B0 ～ B7、C0 ～ C7、および D0 ～ D7 を選択したときは、スレショルドレベルを超える電圧振幅が必要です。設定操作については、ユーザーズマニュアル IM DLM6054-01JA の 5.18 節をご覧ください。

オートセットアップ後の中心位置

オートセットアップ後の中心位置は 0V になります。

オートセットアップ前に表示されていた波形

オートセットアップをすると、アクイジションメモリにあるデータは上書きされ、オートセットアップ前に表示されていた波形は消去されます。

オートセットアップの取り消し

Undo のソフトキーを押すことで、オートセットアップ直前の設定に戻せます。ただし、電源を OFF にすると、オートセットアップ直前の設定内容は消えてしまうので、Undo 操作は無効です。

初期設定になる項目 / 検出し設定される項目

オートセットアップを実行すると、下表のように初期設定または信号から検出した値に設定されます。下表以外の項目は、既存の設定が保持されます。

I²C		
初期設定になる項目	モード	Every Start
	ヒステリシス	0.6div
	Qualification	対象にしない
検出し設定される項目	SDA/SCL のソースのレベル	
CAN		
初期設定になる項目	モード	SOF
	ヒステリシス	0.6div
	サンプルポイント	62.5%
検出し設定される項目	ビットレート	
	ソースのレベル	
	リセッシブ	
LIN		
初期設定になる項目	モード	Break
	ヒステリシス	0.6div
	サンプルポイント	50.0%
検出し設定される項目	ビットレート	
	ソースのレベル	
	レビジョン	
SPI		
初期設定になる項目	ヒステリシス	0.6div
検出し設定される項目	CS/Clock/ データのソースのレベル	
UART		
初期設定になる項目	モード	Every Data
	ヒステリシス	0.6div
	サンプルポイント	50.0%
検出し設定される項目	ビットレート	
	ソースのレベル	
	極性	

オートセットアップ後の設定変更

シリアルバス信号のタイプに合わせて、それぞれの項目を変更できます。設定範囲については、それぞれの参照先をご覧ください。

I²C	SDA/SCL のソースのレベル / ヒステリシス → 4.2 節の解説
CAN	ソースのビットレート / レベル / ヒステリシス、リセッシブ、サンプルポイント → 4.3 節の解説
LIN	ソースのビットレート / レベル / ヒステリシス、サンプルポイント、レビジョン → 4.4 節の解説
SPI	ビットオーダ、CS/Clock/Data のソースのレベル / ヒステリシス / 極性 → 4.5 節の解説
UART	ソースのビットレート / レベル / ヒステリシス、サンプルポイント、極性、フォーマット、ビットオーダ → 4.6 節の解説

トリガ / 解析 / 検索機能

トリガ / 解析 / 検索機能の操作説明については、それぞれの参照先をご覧ください。

	トリガ	解析	検索
I²C	3.1 節	4.2 節	5.2 節
CAN	3.2 節	4.3 節	5.3 節
LIN	3.3 節	4.4 節	5.4 節
SPI	3.4 節	4.5 節	5.5 節
UART	3.5 節	4.6 節	5.6 節

2.2 シリアルバス信号のトリガ / 解析 / 検索の設定値の共通化

本機器は、トリガ / 解析 / 検索の各機能の設定値を共通にしています。それぞれの操作メニューで設定すると、他の機能の同じ設定項目は同じ設定値になります。

セットアップ→トリガ / 解析 / 検索

セットアップ→解析 / 検索

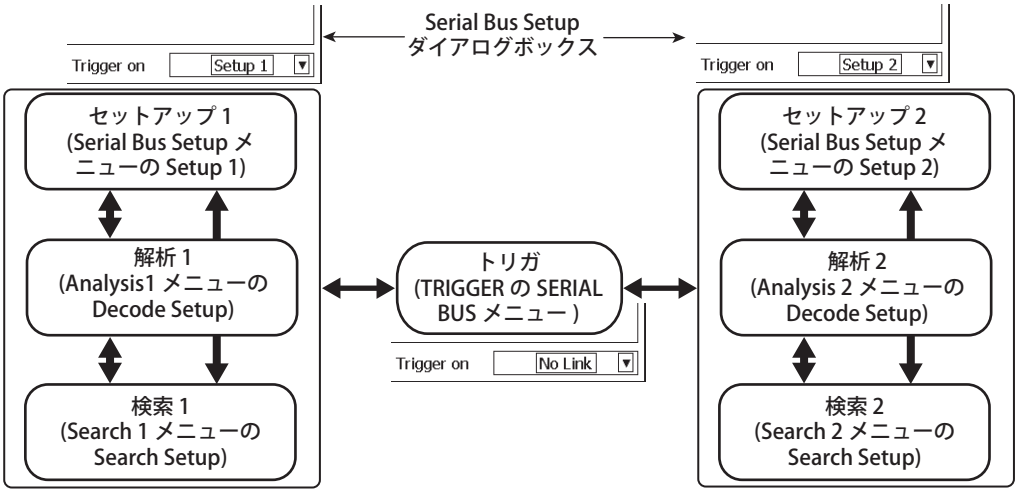
セットアップには Setup 1 と Setup 2、解析には Analysis 1 と Analysis 2、検索には Search 1 と Search 2 があり、同じ番号の設定が連動しています。オートセットアップを実行すると、セットアップした結果が同じ番号の解析、検索の設定に反映されます。

Note

Serial Bus Setup メニューで設定した内容は、オートセットアップを実行しなくても、解析、検索条件の設定に反映されます。

セットアップ→トリガ

- Serial Bus Setup メニューの「Trigger on」のボックスを「Setup 1」にすると、セットアップ 1 の設定値がトリガの設定に反映されます。同様に「Setup 2」にすると、セットアップ 2 の設定値がトリガの設定に反映されます。
- オートセットアップを実行したときは 、設定したシリアルバス信号のタイプ / ソースや検出値 (2-5 ページ参照) が 、トリガ (TRIGGER の SERIAL BUS メニュー) の設定に反映されます。同時に「Trigger on」のボックスにオートセットアップを実行したセットアップ No(Setup 1 または Setup 2) が表示されます。



解析 / 検索→ セットアップ / トリガ

- **解析 / 検索→セットアップ**
  解析 1 または検索 1 の設定を変更したときは、変更した設定値がセットアップ 1 の設定に反映されます。同様に解析 2 または検索 2 の設定を変更したときは、セットアップ 2 の設定に反映されます。
- **解析 / 検索→トリガ**
  - 「Trigger on」のボックスが「Setup 1」になっているとき、解析 1 または検索 1 の設定を変更すると、変更した設定値がトリガの設定に反映されます。解析 2 または検索 2 の設定を変更しても、トリガの設定には反映されません。同様に「Trigger on」のボックスが「Setup 2」になっているとき、解析 2 または検索 2 の設定を変更すると、トリガの設定に反映され、解析 1 または検索 1 の設定を変更しても、トリガの設定には反映されません。
  - 「Trigger on」のボックスが「No Link」になっているとき、解析 1、2 または検索 1、2 の設定を変更しても、トリガの設定には反映されません。

トリガ→セットアップ / 解析 / 検索

- 「Trigger on」のボックスが「Setup 1」になっているとき、トリガの設定を変更すると、変更した設定値がセットアップ 1、解析 1、および検索 1 の設定に反映されます。セットアップ 2、解析 2、および検索 2 の設定には反映されません。同様に「Trigger on」のボックスが「Setup 2」になっているとき、トリガの設定を変更すると、セットアップ 2、解析 2、および検索 2 の設定に反映され、セットアップ 1、解析 1、および検索 1 の設定には反映されません。
- 「Trigger on」のボックスが「No Link」になっているとき、トリガの設定を変更しても、変更した設定値はセットアップ / 解析 / 検索のどの設定にも反映されません。

Note

「Trigger on」のボックスが「Setup 1」または「Setup 2」になっていても、シリアルバス信号 I^2C、CAN、LIN、SPI、および UART 以外のトリガタイプを選択した時点で、「Trigger on」のボックスが「No Link」になります。具体的には、次の操作をしたときに「No Link」になります。

- TRIGGER の ENHANCED メニューで、TV または Serial を選択したとき
- フロントパネルの EDGE/STATE、WIDTH、または EVENT INTERVAL を押したとき

共通項目

シリアルバス信号のトリガタイプ別の共通項目を下表に示します。

トリガタイプ	I^2C	CAN	LIN	SPI	UART
トリガタイプ (Type)	選択したトリガタイプのメニューに変わります。				
ソース (Source)	○	○	○	○	○
ビットレート (Bitrate)	−	○	○	−	○
レベル (Level)	○	○	○	○	○
ヒステリシス (Hys)	○	○	○	○	○
サンプルポイント (Sample Point)	−	○	○*	−	○*
リセッシブ (Recessive)	−	○	−	−	−
レビジョン (Revision)	−	−	○*	−	−
極性 (Polarity/Active)	−	−	−	○	○
ビットオーダ (Bit Order)	−	−	−	○	○*
線式 (Mode)	−	−	−	○	−
フォーマット (Format)	−	−	−	−	○
パリティ (Parity)	−	−	−	−	○*

○：共通項目、−：項目なし

* トリガ機能には、設定メニューがありません。

ヒステリシス

トリガの設定のヒステリシスをセットアップ / 解析 / 検索の設定に反映させる場合は、以下の対応表に従います。

トリガの設定	セットアップ / 解析 / 検索の設定
	0.6div
	1.0div

セットアップ / 解析 / 検索のヒステリシスをトリガの設定に反映させる場合は、以下の対応表に従います。

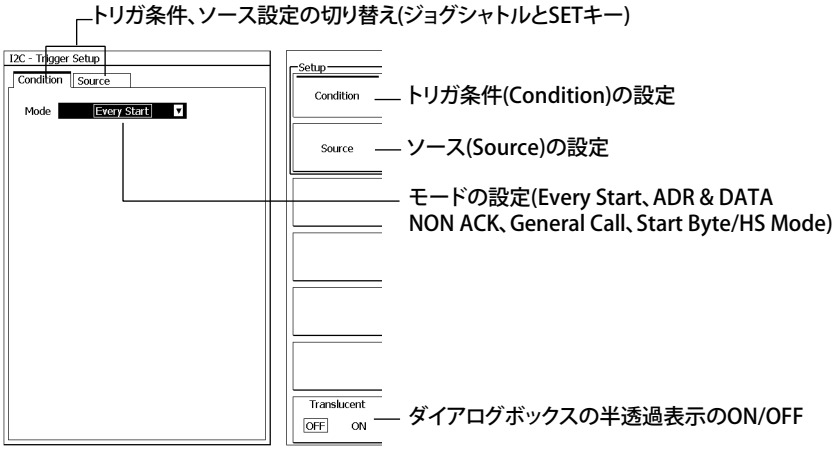
セットアップ / 解析 / 検索の設定	トリガの設定
0.9div 以下	
1.0div 以上	

3.1　I²C バス信号でトリガをかける

操　作

Setup メニュー

ENHANCED キー > Type のソフトキー > Serial Bus のソフトキー > I2C のソフトキー > Setup のソフトキーを押します。次のメニューが表示されます。

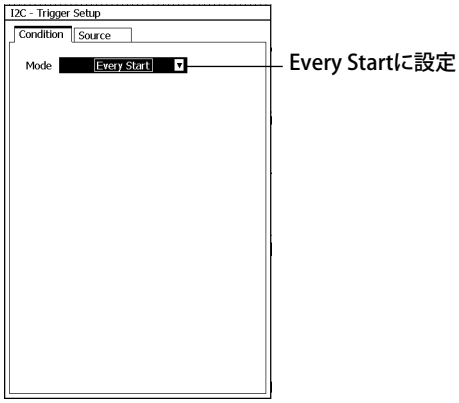


トリガ条件の設定 (Condition)

対象とするアドレスによって、5 つのモードがあります。

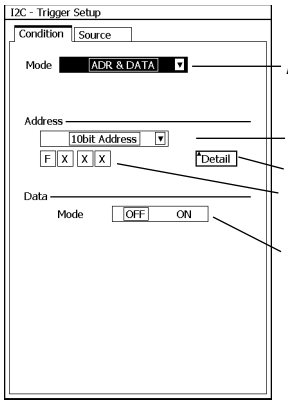
- Every Start
- ADR & DATA
- NON ACK
- General Call
- Start Byte/HS Mode

Every Start でトリガをかける場合



ADR & DATA でトリガをかける場合

10bit Addressのとき



ADR & DATAに設定

アドレスの設定

アドレスパターンの詳細設定

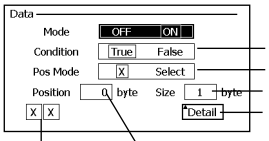
アドレスパターンの設定

データパターンをトリガ条件にする/しないの設定

7bit Addressのとき

7bit+Sub Addressのとき

データパターンをトリガ条件にする場合



比較条件の設定

比較方法の設定

データ長の設定

データパターンの詳細設定

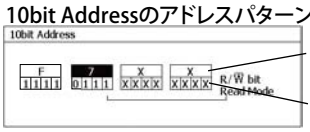
データパターンの設定

Pos ModeがSelectのときの比較開始点の設定

アドレスパターン/データパターンの詳細設定

Detailを選択して表示されるダイアログボックスで、ジョグシャトル、**SET**キー、ソフトキーを使ってアドレスパターン/データパターンを設定できます。

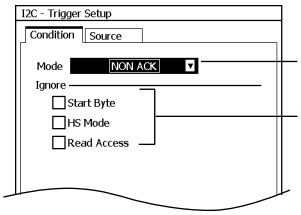
10bit Addressのアドレスパターン例



16進数

2進数

NON ACK でトリガをかける場合

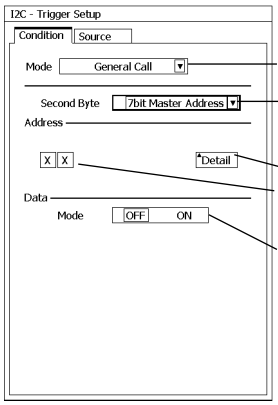


NON ACKに設定

対象にするAcknowledgeビットの選択

General Call でトリガをかける場合

7bit Master Addressのとき



General Callに設定

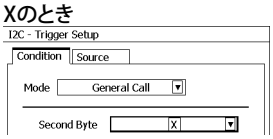
セカンドバイトの設定

アドレスパターンの詳細設定

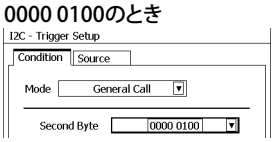
アドレスパターンの設定

データパターンをトリガ条件にする/しないの設定

Xのとき

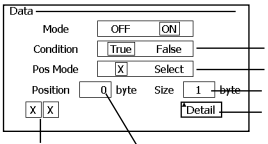


0000 0100のとき



0000 0110のとき

データパターンをトリガ条件にする場合



比較条件の設定

比較方法の設定

データ長の設定

データパターンの詳細設定

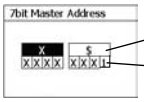
データパターンの設定

Pos ModeがSelectのときの比較開始点の設定

アドレスパターン、データパターンの詳細設定

Detailを選択して表示されるダイアログボックスで、ジョグシャトル、**SET**キー、ソフトキーを使ってアドレスパターン、データパターンを設定できます。

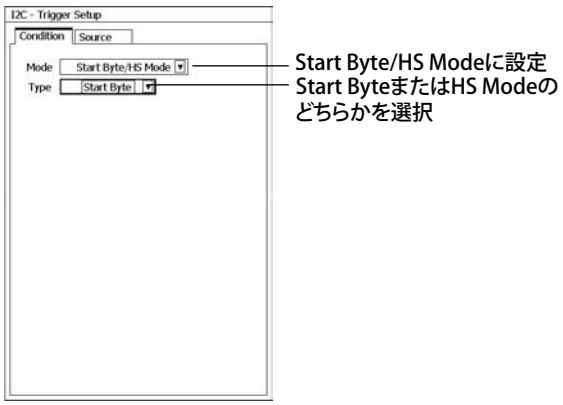
7bit Master Addressのアドレスパターンの例



16進数

2進数

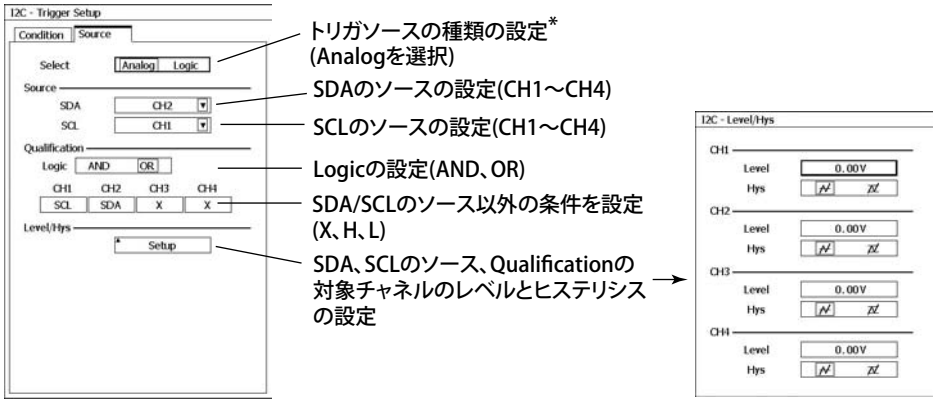
Start Byte/HS Mode でトリガをかける場合



トリガソースの設定 (Source)

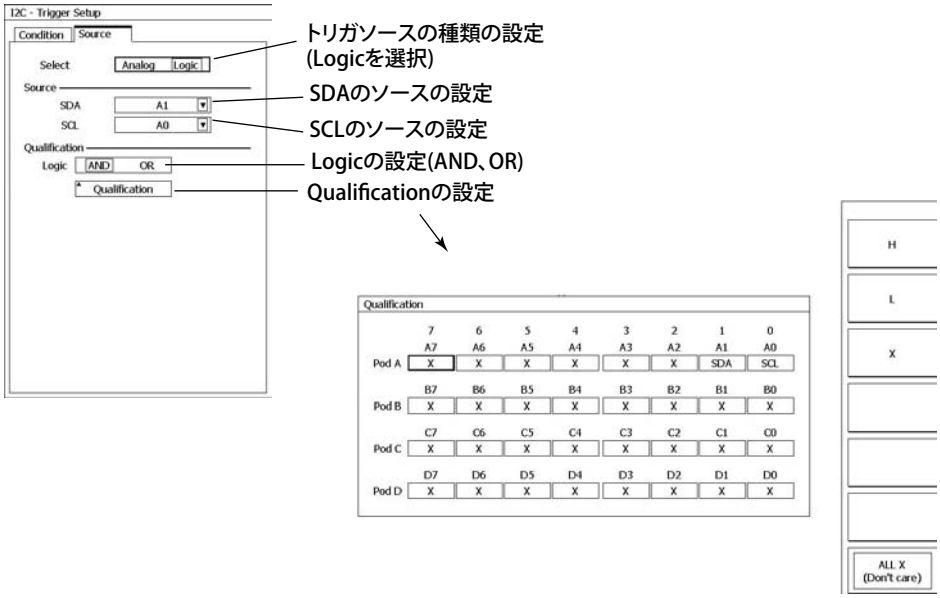
データパターンと比較するトリガソースや比較条件を設定します。

アナログ入力のとき



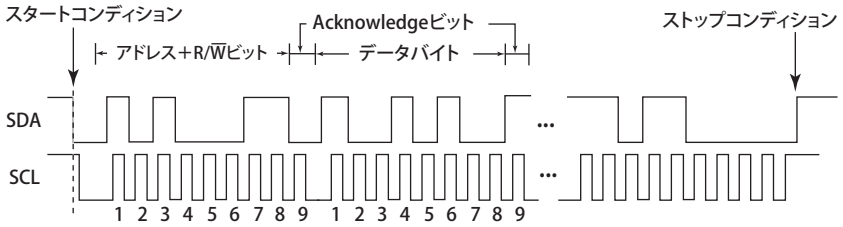
*：DL6000シリーズではAnalogに固定(設定無し)

ロジック入力のとき (DLM6000)



解　説

I²C バス信号でトリガをかける機能です。下図に I²C バス信号のデータフォーマットを示します。

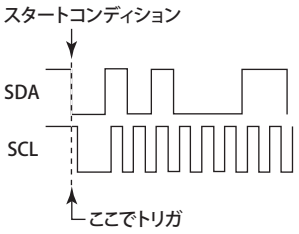


モード

I²C トリガの種類を、Every Start、ADR & DATA、NON ACK、General Call、および Start Byte/HS Mode の 5 つのモードから選択します。

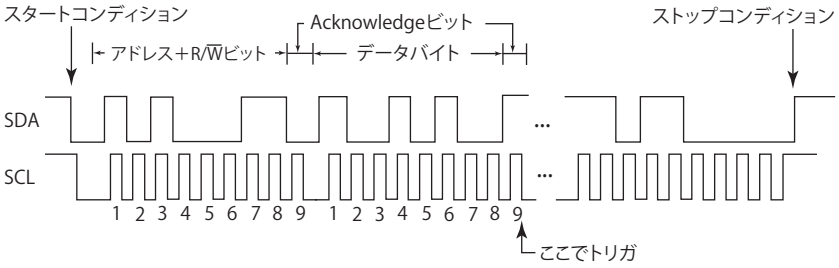
Every Start モード

スタートコンディションを検出すると、SDA 信号の立ち下がりでトリガがかかります。



ADR & DATA モード

Address と Data が一致すると、SCL 信号の 9 つ目のクロックの立ち下がりでトリガがかかります。



- **Address**
  - アドレスのタイプを 7bit Address、7bit + Sub Address、または 10bit Address から選択できます。
  - アドレスパターンを 16 進数または 2 進数で設定します。設定したアドレスパターンと 入力信号のアドレスパターンが一致したとき、Address のトリガ条件 が成立したことになります。
  - パターンに X を設定すると、対応するビットの状態にかかわらず条件を満たしていると見なされます。
  - 2 進のパターンに 1 つでも X があると、対応する 16 進の表示は「$」になります。

- **Data**

  データパターンをトリガ条件にする (ON)/ しない (OFF) の選択ができます。

  - 比較条件 (Condition)

    設定したパターンと入力信号のパターンを比較して、選択した比較条件を満たしたとき、Data のトリガ条件が成立したことになります。

True	パターンが一致したとき
False	パターンが一致しないとき

  - 比較開始点 (Position)

    Pos Mode の設定で、比較開始点を設定した点 (Select)/ 設定にかかわらない (X) から選択できます。Select を選択すると、設定した分のバイト数だけスキップした次のデータから比較します。

    設定範囲：0 ～ 9999 バイト
  - データ長 (Size)

    連続したデータを、何バイト分比較するかを設定します。

    設定範囲：1 ～ 4 バイト
  - データパターン

    Size で設定した長さのデータに対して、データパターンを 16 進数または 2 進数で設定します。
  - パターンに X を設定すると、対応するビットの状態にかかわらず条件を満たしていると見なされます。
  - 2 進のパターンに 1 つでも X があると、対応する 16 進の表示は「$」になります。

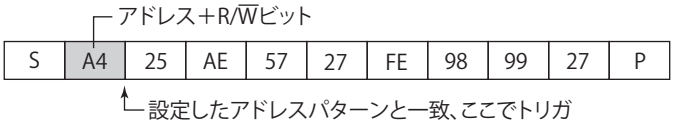
- **設定例**

  ここでは、データ列をバイト単位 (16 進) で表示し、トリガがかかる位置を示します。図中で使用する記号は、次のとおりです。

  S：スタートコンディション、P：ストップコンディション、網掛け：パターン比較対象

  アドレスパターンだけでトリガ

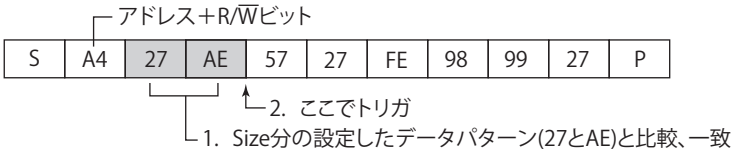
Mode	ADR & DATA
Address	7bit address、A4
Data	Mode：OFF



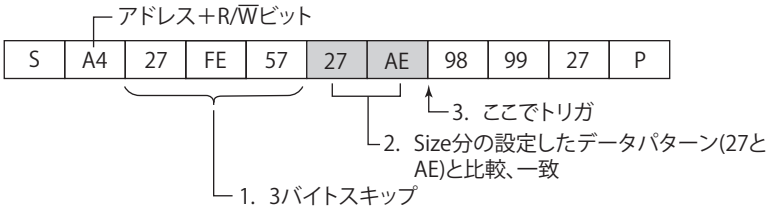
  データパターンだけでトリガ

Mode	ADR & DATA
Address	対象外
Data	Mode：ON、Condition：True、Size：2 bytes、データパターン：27 と AE

  < Pos Mode：X >

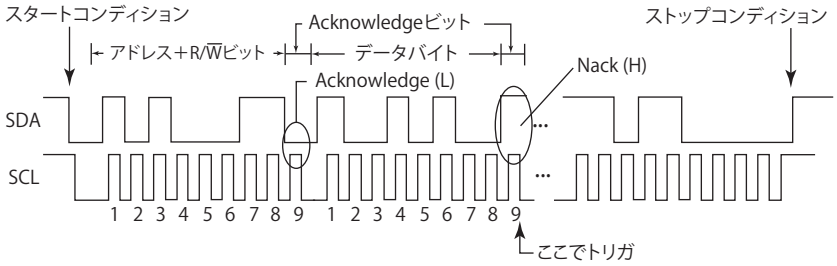


  < Pos Mode：Select、Position：3 >



NON ACK モード

Acknowledge ビットが Nack とき (SDA 信号が H のとき) にトリガがかかります。
スタートバイト (Start Byte)、HS モードマスターコード (HS Mode)、リードアクセスバイト (Read Access) の Acknowledge ビットを対象にするか、対象外 (Ignore) にするかを選択できます。



General Call モード

ゼネラルコールアドレス (0000 0000) で、トリガがかかります。

- **Second Byte**
  ゼネラルコールアドレスのあとのバイト (Second Byte) のアドレスパターンを、トリガ条件にできます。設定したパターンと入力信号のパターンが一致したとき、Second Byte のトリガ条件が成立したことになります。

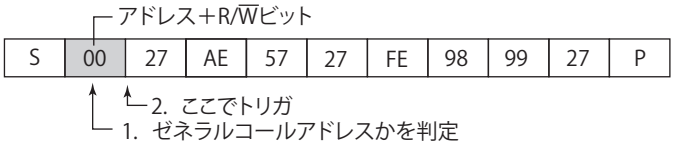
X	対象にしない
0000 0100	パターン 0000 0100 (0x04) と一致したとき
0000 0110	パターン 0000 0110 (0x06) と一致したとき
7bit Master Address	任意に設定したパターンと一致したとき 7bit Master Address を設定すると、次項のデータパターン (Data) をトリガ条件にできます。

- **Data**
  トリガ条件や設定項目は、3-5 ページで説明しているものと同じです。それぞれの説明をご覧ください。

- **設定例**
  ここでは、データ列をバイト単位 (16 進) で表示し、トリガがかかる位置を示します。図中で使用する記号は、次のとおりです。
  S：スタートコンディション、P：ストップコンディション、網掛け：パターン比較対象

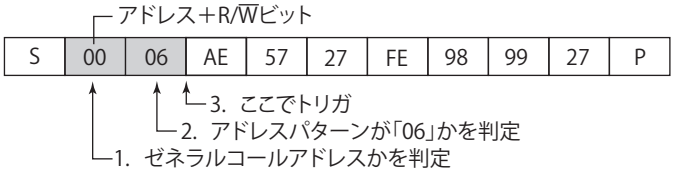
ゼネラルコールアドレスだけでトリガ

Mode	General Call
Second Byte	X



Second Byte のアドレスが「06」のパターンでトリガ

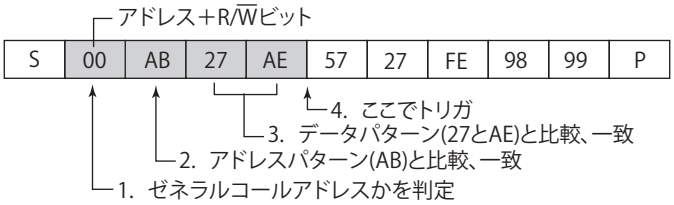
Mode	General Call
Second Byte	0000 0110



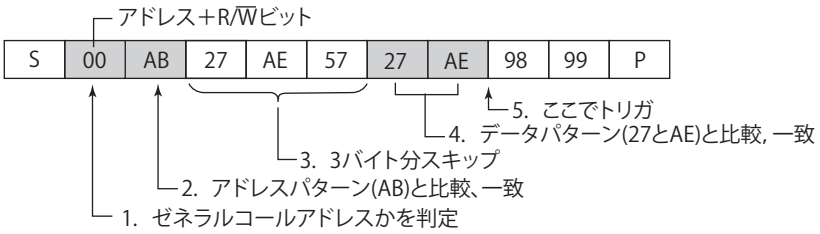
Second Byte のアドレスが任意のパターンでトリガ

Mode	General Call
Second Byte	7bit Master Address、アドレスパターン：1010 1011 (0xAB)
Data	Mode：On、Condition：True、Size：2 bytes、データパターン：27 と AE

< Pos Mode：X >



< Pos Mode：Select、Position：3 >

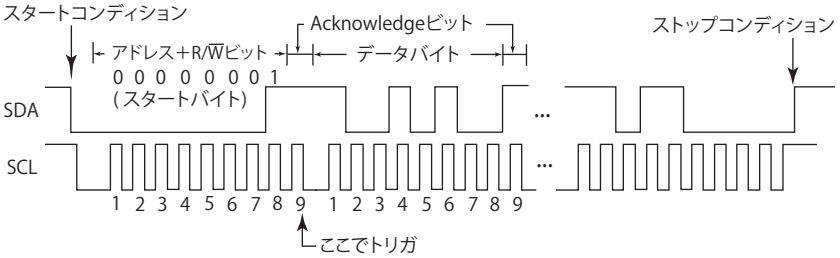


Start Byte/HS Mode モード

Start Byte か HS Mode のマスタコードで、トリガがかかります。

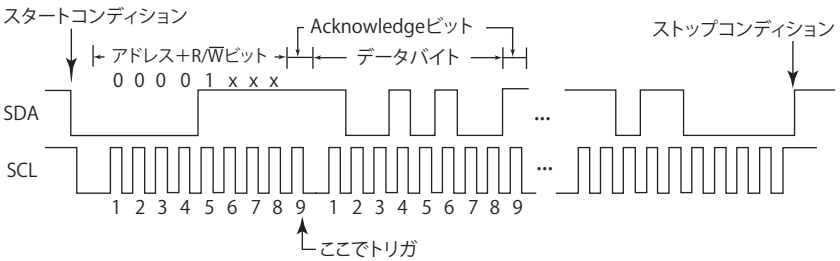
- **Start Byte**

  スタートバイト (パターン：0000 0001) を検出すると、トリガがかかります。



- **HS Mode**

  HS モード (ハイスピードモード) のマスタコード (パターン：0000 1XXX) を検出すると、トリガがかかります。



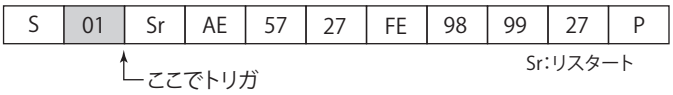
- **設定例**

  ここでは、データ列をバイト単位 (16 進) で表示し、トリガがかかる位置を示します。図中で使用する記号は、次のとおりです。

  S：スタートコンディション、P：ストップコンディション、網掛け：パターン比較対象

  Start Byte でトリガ

Mode	Start Byte/HS Mode
Type	Start byte



SDA/SCL/Qualification

SDA/SCL のソース

SDA(シリアルデータ)/ SCL(シリアルクロック) のソースを選択できます。

- DL6000 シリーズの場合、CH1 ～ CH4 から選択します。
- DLM6000 シリーズの場合、CH1 ～ CH4、A0 ～ A7、B0 ～ B7、C0 ～ C7、および D0 ～ D7 (16 ビットモデルは A0 ～ A7 と C0 ～ C7) から選択します。

トリガレベル (Level)

CH1 ～ CH4 ごとに、I^2C バス信号のトリガレベルを設定できます。

- 設定範囲は画面内 8div 分で、設定分解能は 0.01div です。たとえば、2V/div のときの設定分解能は 0.02V です。
- RESET キーを押すことで、トリガレベルを現在のオフセット電圧値にリセットすることもできます。
- ソースが A0 ～ D7 のときのトリガレベルは、ユーザーズマニュアル IM DLM6054-01JA の 5.2 節で設定したスレショルドレベルです。

ヒステリシス (Hys)

トリガレベルに幅を持たせて、小さな変動ではトリガがかからないようにします。

	トリガレベルを中心に、約 0.3div* のヒステリシス
	トリガレベルを中心に、約 1div* のヒステリシス

* 上記の数値は、おおよその値です。厳密に保証するものではありません。

セットアップ / 解析 / 検索のヒステリシスとの関係は 2-7 ページをご覧ください。

Qualification と論理

- **必要条件**

  トリガをかけるための必要条件として、SDA と SCL に選択したソース以外の各信号の状態を H、L、または X から選択します。選択した状態と入力信号の状態が一致したとき、必要条件が成立したことになります。

H	High レベルのとき
L	Low レベルのとき
X	対象にしない (Don't care)

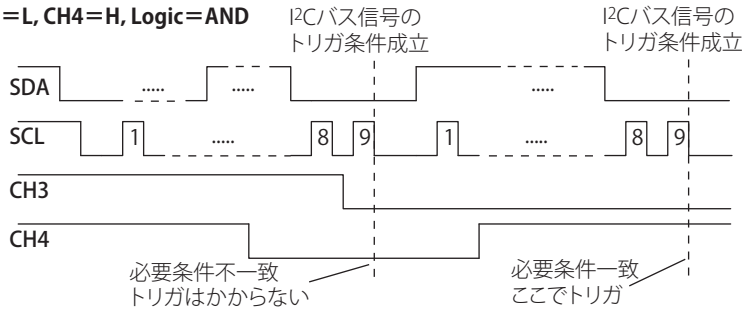
* High か Low かの判定レベルは、信号が CH1 ～ CH4 のとき、上記で設定したトリガレベルです。A0 ～ D7 のときは、ユーザーズマニュアル IM DLM6054-01JA の 5.2 節で設定したスレショルドレベルです。

- **論理条件 (Logic)**

  上項の必要条件と各モードで設定した I²C バス信号のトリガ条件の、論理条件を選択できます。
  論理条件を満たしたとき、トリガがかかります。

AND	必要条件 と I²C バス信号のトリガ条件が、両方成立したとき
OR	必要条件のどれかが成立したときに、I²C バス信号のトリガ条件が成立したとき

CH3＝L, CH4＝H, Logic＝AND



Note

I²C バス信号のトリガ条件 (SDA 信号 /SCL 信号) だけでトリガをかける場合は、次のように設定します。

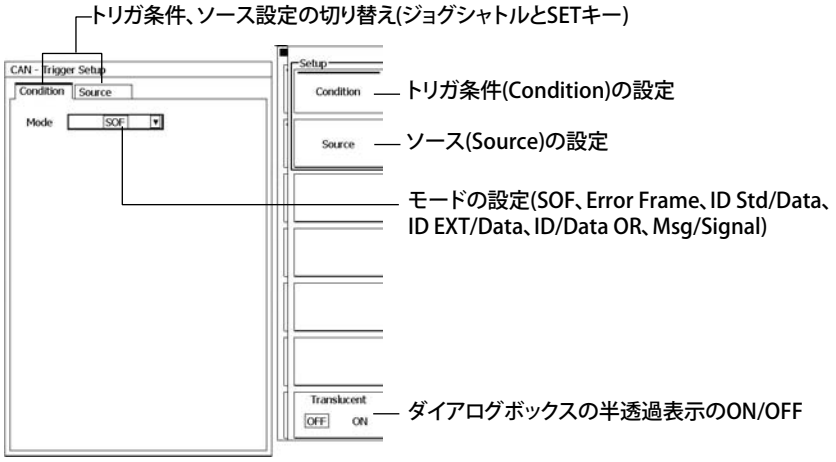
- SDA と SCL に選択したソース以外の各信号の状態：X(トリガソースにしない)
- 論理：AND

3.2 CAN バス信号でトリガをかける

操　作

Setup メニュー

ENHANCED キー > Type のソフトキー > Serial Bus のソフトキー > CAN のソフトキー > Setup のソフトキーを押します。次のメニューが表示されます。

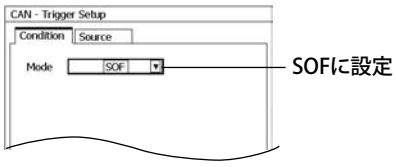


トリガ条件の設定 (Condition)

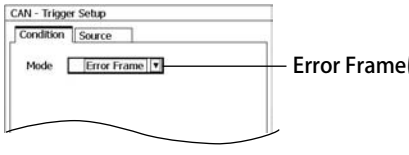
対象とするアドレスによって、6 つのモードがあります。

- SOF
- Error Frame
- ID Std/Data
- ID EXT/Data
- ID/Data OR
- Msg/Signal

SOF でトリガをかける場合

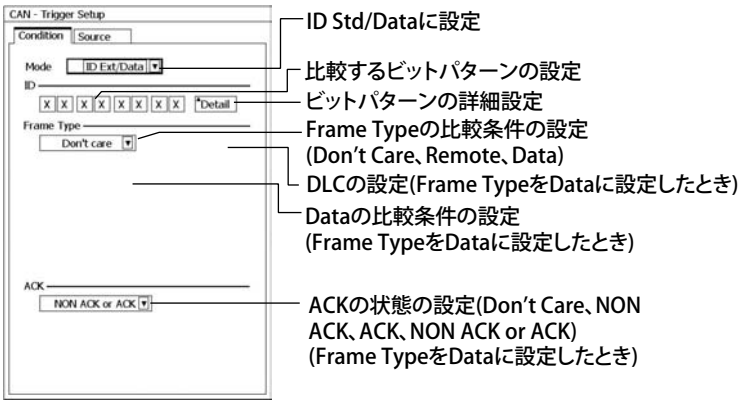


Error Frame でトリガをかける場合



ID Std/Data または ID Ext/Data でトリガをかける場合

ID Std/Data

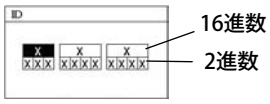


ID Ext/Data

ビットパターンの詳細設定

Detailを選択して表示されるダイアログボックスで、ジョグシャトル、**SET**キー、ソフトキーを使ってビットパターンを設定できます。

ID Std/DataのIDビットパターンの例

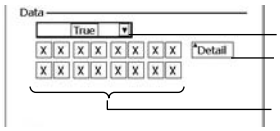


Frame Type の Data をトリガ条件にする場合

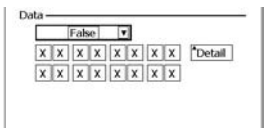
Data の比較条件によって設定内容が異なります。

Data の比較条件が True または False のとき

True

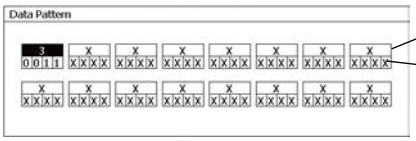


False



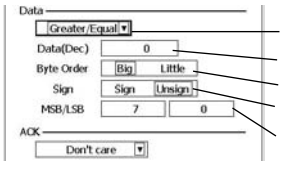
データパターンの詳細設定

Detailを選択して表示されるダイアログボックスで、ジョグシャトル、**SET**キー、ソフトキーを使ってデータパターンを設定できます。

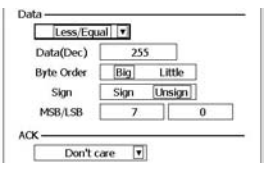


Data の比較条件が Greater/Equal または Less/Equal のとき

Greater/Equal

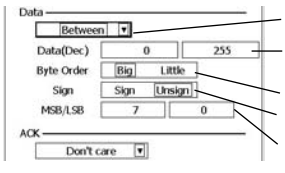


Less/Equal

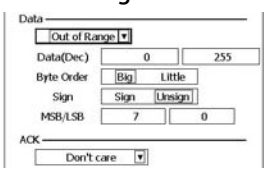


Data の比較条件が Between または Out of Range のとき

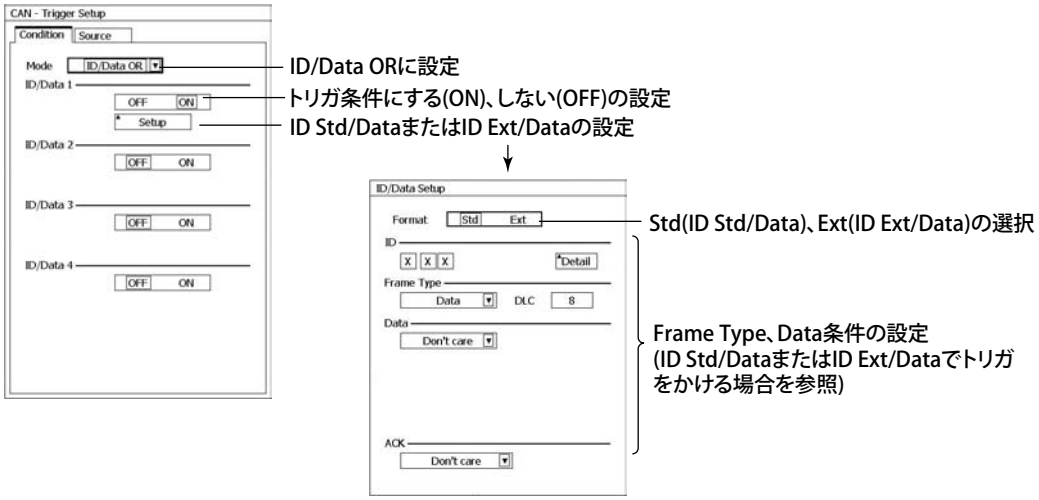
Between



Out of Range

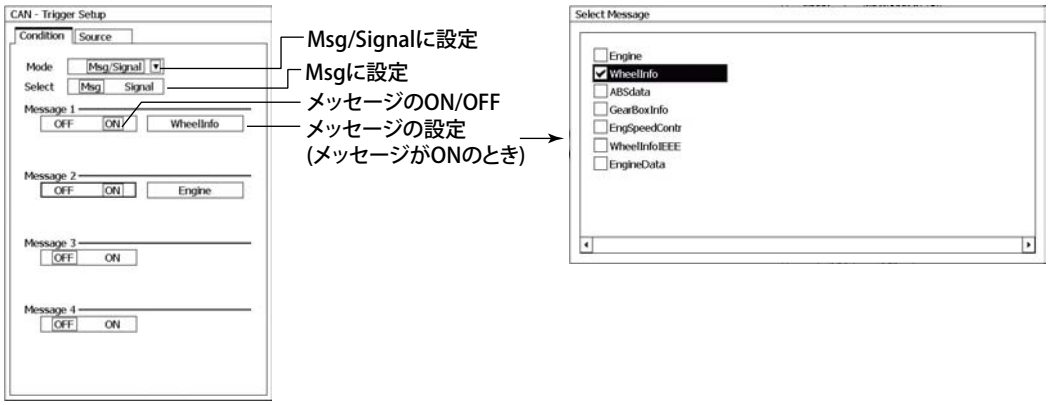


ID/Data OR でトリガをかける場合

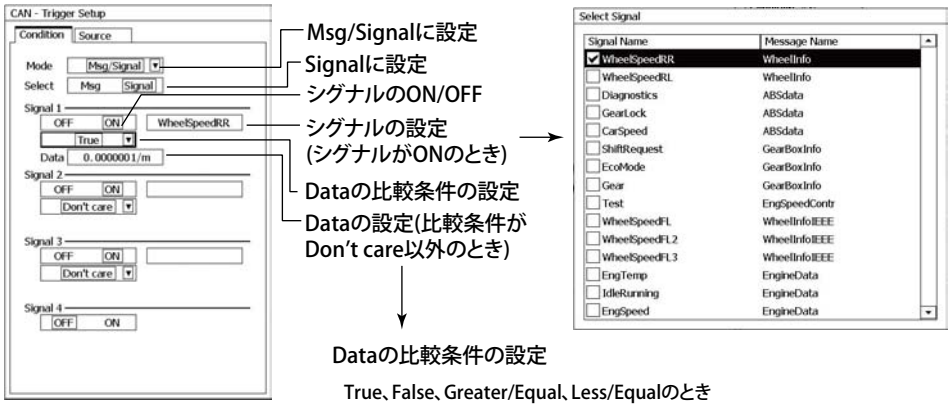


Msg/Signal でトリガをかける場合

Message をトリガ条件にする場合

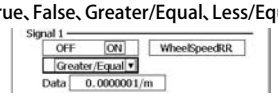


Signal をトリガ条件にする場合

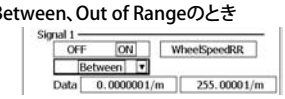


Dataの比較条件の設定

True、False、Greater/Equal、Less/Equalのとき

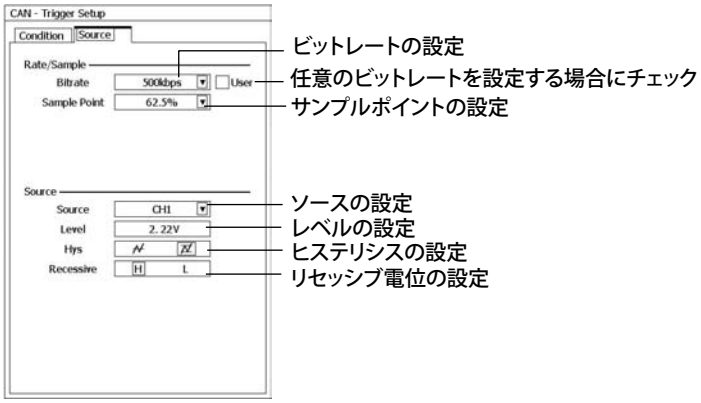


Between、Out of Rangeのとき



トリガソースの設定 (Source)

Source タブを選択するか、Source のソフトキーを押します。次のメニューが表示されます。



解　説

CANバス信号でトリガをかける機能です。CANバス信号のフレームフォーマットについては、3-18ページをご覧ください。

モード

CAN トリガの種類を、SOF、Error Frame、ID Std/Data、ID Ext/Data、ID/Data OR、および Msg/Signal の 6 つのモードから選択します。

SOF モード

CAN バス信号のフレームの開始を検出して、トリガがかかります。

SOF：Start of Frame

Error Frame モード

Error Frame の Error Flag がアクティブエラーフラグのとき、トリガがかかります。

ID Std/Data モード、ID Ext/Data モード

ID Std/Data モードは、標準フォーマットの場合の Data Frame や Remote Frame に対して、トリガをかけるモードです。

ID Ext/Data モードは、拡張フォーマットの場合の Data Frame や Remote Frame に対して、トリガをかけるモードです。

ID、Frame Type、Data、および ACK の AND 条件でトリガがかかります。

ID Std/Data モードの設定内容と、ID Ext/Data モードの設定内容は共通です。

ID

標準フォーマットの場合 は 11 ビットの、拡張フォーマットの場合は 29 ビットの ID の ビットパターンを 16 進数または 2 進数で設定します。設定したビットパターンと 入力信号の ID のビットパターンが一致したとき、ID のトリガ条件が成立したことになります。

- パターンに X を設定すると、対応するビットの状態にかかわらず条件を満たしていると見なされます。
- 2 進のパターンに 1 つでも X があると、対応する 16 進の表示は「$」になります。

Frame Type

Remote Frame と Data Frame をトリガ対象として設定できます。

・ Frame の選択

CAN バス信号フレームには、Remote Frame か Data Frame かを識別する RTR(Remote Transmission Request bit) があります。どちらのフレームをトリガ対象にするかを選択します。

Don't care	Remote Frame と Data Frame の両方がトリガ対象になります。
Remote	Remote Frame がトリガ対象になります。
Data Frame	Data Frame がトリガ対象になります。

Don't care または Remote を選択したときは、次項の DLC や Data のトリガ条件は無視されます。

・ DLC(Data Length Code)

Data Field のデータ長を設定します。設定値と入力信号の DLC の値が一致したとき、DLC のトリガ条件が成立したことになります。Data Frame を選択したときだけ設定します。

設定範囲：0 ～ 8 バイト

「0」のときは、次項の Data のトリガ条件は無視されます。

Data

Data Field の値をトリガ条件として設定できます。Data Frame を選択したときだけ設定します。

・ 比較条件

判定値と入力信号の Data Field の値を比較して、選択した比較条件を満たしたとき、Data のトリガ条件が成立したことになります。

Don't care	対象にしない
True	判定値と一致したとき
False	判定値と一致しないとき
Greater/Equal	判定値以上のとき
Less/Equal	判定値以下のとき
Between	判定範囲内のとき (判定値を含む)
Out of Range	判定範囲外のとき (判定値を除く)

・ データパターン

DLC で設定した長さのデータに対して、データパターンを 16 進数または 2 進数で設定します。比較条件が True または False のときだけ有効です。

- パターンに X を設定すると、対応するビットの状態にかかわらず条件を満たしていると見なされます。
- 2 進のパターンに 1 つでも X があると、対応する 16 進の表示は「$」になります。

・ 判定値 (Data(Dec))

- 比較条件が Greater/Equal または Less/Equal の場合は、判定値を 1 つ設定します。
- 比較条件が Between または Out of Range の場合は、判定範囲を設定する必要があるため、判定値を 2 つ設定します。下限値≦上限値になるように自動的に調整されます。
- 比較条件が True または False の場合は、データパターンを判定値として使用します。
- 設定範囲

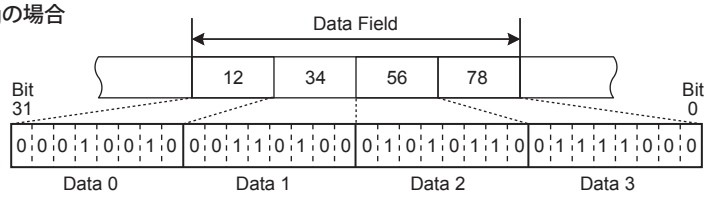
  10 進数で設定します。

符号が付かないとき (Unsign)	0 ～ 9E ＋ 18 ただし、設定可能な最大値は、DLC と MSB/LSB の設定で決まるデータ長とビット位置によって制限されます。
符号が付くとき (Sign)	－ 9E ＋ 18 ～ 9E ＋ 18 ただし、設定可能な最小値 / 最大値は、DLC と MSB/LSB の設定で決まるデータ長とビット位置によって制限されます。

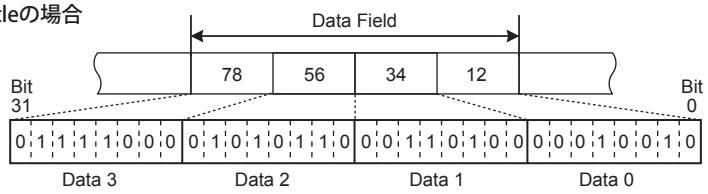
設定値は、7 桁を超えると指数で表示されます (例：1234567E ＋ 10)。

・ バイトオーダー (Byte Order)
Data のバイトオーダー (転送順序) をビッグエンディアン (Big)/ リトルエンディアン (Little) から選択します。たとえば、4 バイトの Data(12345678：16 進数) がバス上を流れるイメージは、下図のようになります。

Bigの場合



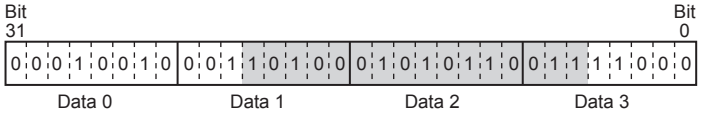
Littleの場合



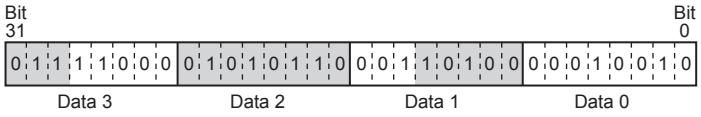
・ 符号 (Sign)
Data に符号を付ける (Sign)/ 付けない (Unsign) を選択します。
符号が付くときと付かないときで、Data の判定値の設定範囲が変わります。

・ 最上位ビット / 最下位ビット (MSB/LSB)
比較する Data の最上位ビット (MSB)/ 最下位ビット (LSB) の位置を設定します。 たとえば、4 バイトのデータ (12345678：16 進数) のビット 5 ～ビット 20 の範囲を比較する場合は、MSB を 20、LSB を 5 に設定します。この場合、Byte Order の設定 (Big または Little) によって、比較するビットは、下図の網掛けの部分になります。
設定範囲：0 ～ (データ長のバイト数× 8 － 1)、最大値は 63 です。

Bigの場合



Littleの場合



ACK

ACK スロットの状態をトリガ条件として設定できます。選択した状態と入力信号の ACK スロットの状態が一致したとき、ACK のトリガ条件が成立したことになります。

Don't care	対象にしない
NON ACK	リセッシブのとき
ACK	ドミナントのとき
NON ACK or ACK	リセッシブまたはドミナントのとき

ID/Data OR モード

複数の ID Std/Data の OR 条件、または複数の ID Ext/Data の OR 条件で、トリガがかかります。4 つまでの ID Data を設定できます。ID Std/Data の設定内容と、ID Ext/Data の設定内容は共通です。

- 各 ID/Data に対して、トリガ条件にする (ON)/ しない (OFF) の選択ができます。
- 各 ID/Data のそれぞれのトリガ条件や設定項目は、3-14 ～ 3-16 ページにかけて説明しているものと同じです。それぞれの説明をご覧ください。

Note

モードに ID/Data OR を選択した場合、トリガ点が同じになる条件を設定してください。トリガ点が異なる条件を混在して設定すると、正しい位置でトリガがかからない場合があります。

Msg/Signal モード

本機器に読み込んだ物理値 / シンボル定義ファイル (.sbl)* に収納されている、Message や Signal のデータをトリガ条件にするモードです。

* 物理値 / シンボル定義ファイル (.sbl) は、CANdb ファイル (.dbc) を変換したものです。ファイルの読み込み操作については、別冊のユーザーズマニュアル (IM DLM6054-01JA) の 13.9 節をご覧ください。

Msg/Signal モードのトリガ条件の設定内容は次のようになります。

Msg モード		
ID		sbl ファイルから読み込み
その他の条件		対象にしない
Signal モード		
ID		sbl ファイルから読み込み
Frame Type		Data Frame に固定
Data		
	比較条件	選択した条件
	判定値	設定値
	Byte Order	sbl ファイルから読み込み
	Sign	sbl ファイルから読み込み
	MSB/LSB	sbl ファイルから読み込み
	ACK	対象にしない

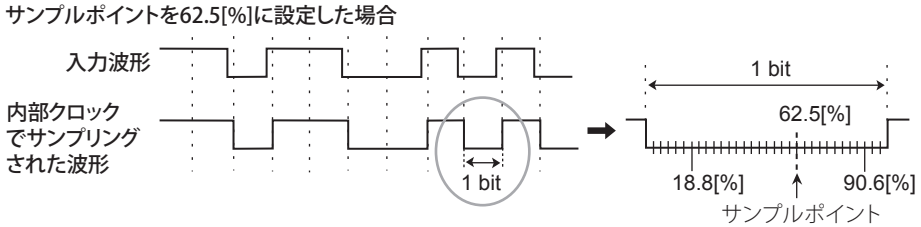
ソースのビットレート / サンプルポイント / トリガレベル / ヒステリシス / リセッシブ電位

ビットレート (Bitrate)

CAN バス信号の転送レートを次の中から選択できます。
1Mbps、500kbps、250kbps、125kbps、83.3kbps、33.3kbps
User 設定を選択した場合は、10.0kbps ～ 1.000Mbps の範囲 (設定分解能 0.1kbps) で設定できます。

サンプルポイント (Sample Point)

バスレベル (リセッシブ / ドミナント) を判定するポイントを 18.8 ～ 90.6[%] の範囲 (設定分解能 3.1%) で設定できます。
本機器の CAN バス信号のトリガ回路では、入力された CAN バス信号を内部クロックでサンプリングして、リセッシブからドミナントへの変化点を検出しています。検出された変化点を 0% とし、変化点からビットタイム (設定したビットレートの逆数) が経過したところを 100% として、サンプルポイントを % で設定します。



トリガレベル (Level)

CH1 ～ CH4 ごとに、CAN バス信号のトリガレベルを設定できます。

- 設定範囲は画面内 8div 分で、設定分解能は 0.01div です。たとえば、2V/div のときの設定分解能は 0.02V です。
- RESET キーを押すことで、トリガレベルを現在のオフセット電圧値にリセットすることもできます。

ヒステリシス (Hys)

トリガレベルに幅を持たせて、小さな変動ではトリガがかからないようにします。

⩘	トリガレベルを中心に、約 0.3div* のヒステリシス
⩛	トリガレベルを中心に、約 1div* のヒステリシス

* 上記の数値は、おおよその値です。厳密に保証するものではありません。
セットアップ / 解析 / 検索のヒステリシスとの関係は 2-7 ページをご覧ください。

リセッシブ電位 (Recessive)

リセッシブ電位を次の中から選択します。どちらの設定でも論理値は、
リセッシブ＝ 1、ドミナント＝ 0 です。

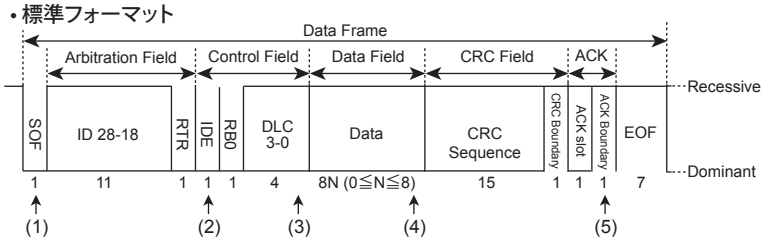
H	リセッシブ電位がドミナント電位より高い
L	リセッシブ電位がドミナント電位より低い

フレームのフォーマットとトリガ点

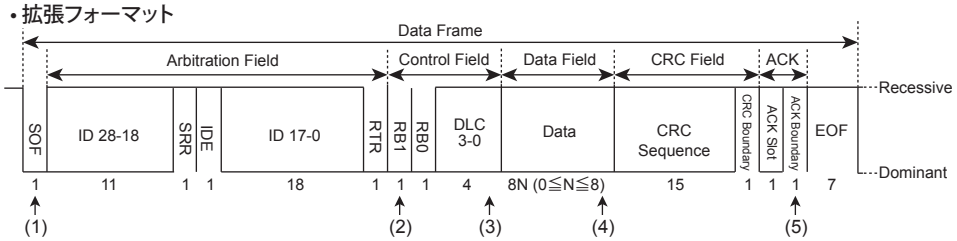
各フレームのフォーマットとトリガ点は、下図のとおりです。

データフレーム

• 標準フォーマット



• 拡張フォーマット

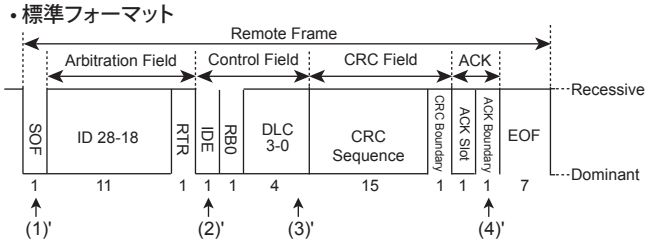


上記の(1)～(5)は、下記の条件のときのトリガ点です。
(1) モード: SOF
(2) モード: ID X*、 Frame(RTR): Don't care、 ACK: Don't care
(3) モード: ID X*、 Frame(RTR): Data、 Data Field: Don't care、 ACK: Don't care
(4) モード: ID X*, Frame(RTR): Data、 Data Field: Don't care以外、ACK: Don't care
(5) ACK: Don't care以外

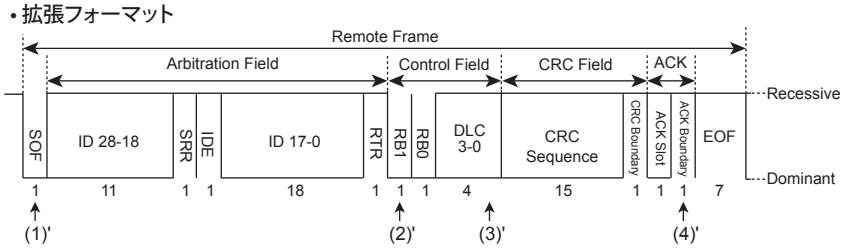
* ID X: ID Std/Data, ID Ext/Data, またはID/Data ORのとき

リモートフレーム

• 標準フォーマット



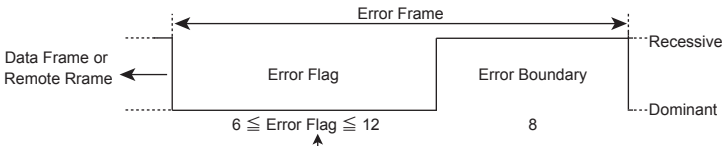
• 拡張フォーマット



上記の(1)'～(5)'は、下記の条件のときのトリガ点です。
(1)' モード: SOF
(2)' モード: ID X*、 Frame(RTR): Don't care、 ACK: Don't care
(3)' モード: ID X*、 Frame(RTR): Remote、 ACK: Don't care
(4)' ACK: Don't care以外

* ID X: ID Std/Data, ID Ext/Data, またはID/Data ORのとき

エラーフレーム



モードがError Frameのとき、エラーフラグの
6ビット目がトリガ点になります。

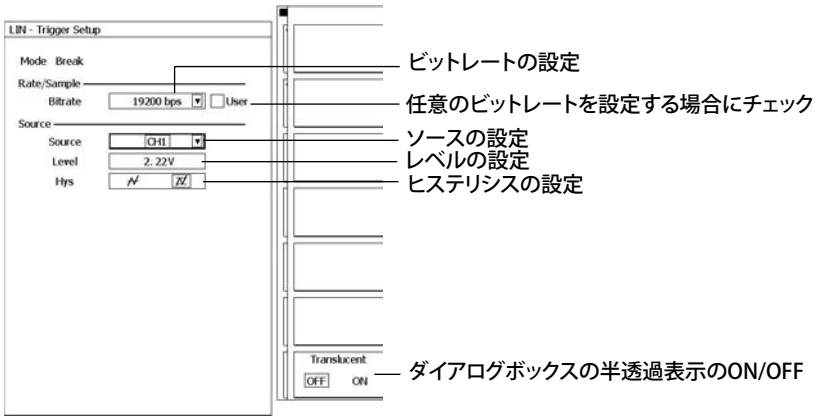
3.3 LIN バス信号でトリガをかける

操　作

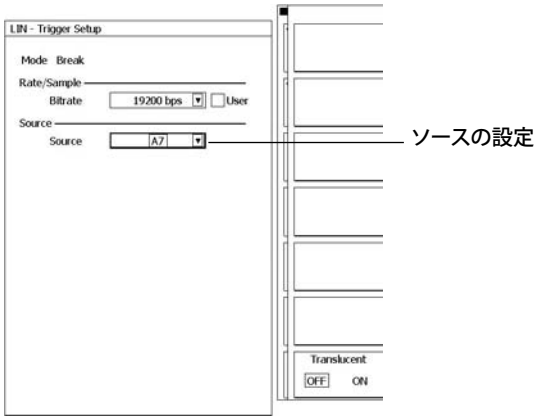
Setup メニュー

ENHANCED キー > Type のソフトキー > Serial Bus のソフトキー > LIN のソフトキー > Setup のソフトキーを押します。次のメニューが表示されます。

ソースがアナログ信号(CH1～CH4)のとき



ソースがロジック信号(A0～D7)のとき

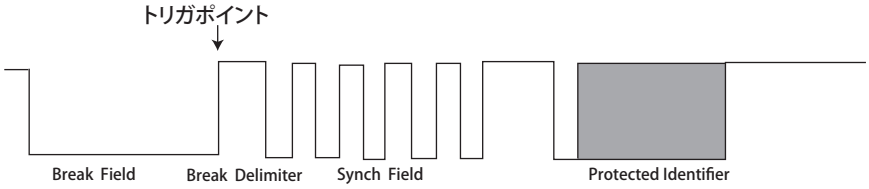


解　説

LIN バスの Break delimiter の立ち上がりでトリガをかける機能です。

モード (Mode)

モードは、Break に固定です。LIN バス信号の Break delimiter の立ち上がりでトリガがかかります。



ソース (Source)

ソースを選択できます。

- DL6000 シリーズの場合、CH1 ～ CH4 から選択します。
- DLM6000 シリーズの場合、CH1 ～ CH4、A0 ～ A7、B0 ～ B7、C0 ～ C7、および D0 ～ D7 (16 ビットモデルは A0 ～ A7 と C0 ～ C7) から選択します。

ビットレート (Bitrate)

LIN バス信号の転送レートを次の中から選択できます。

19200bps、9600bps、4800bps、2400bps、1200bps

User 設定を選択した場合は、1000bps ～ 20000bps の範囲 (設定分解能 10bps) で設定できます。

トリガレベル (Level)

CH1 ～ CH4 ごとに、LIN バス信号のトリガレベルを設定できます。

- 設定範囲は画面内 8div 分で、設定分解能は 0.01div です。たとえば、2V/div のときの設定分解能は 0.02V です。
- RESET キーを押すことで、トリガレベルを現在のオフセット電圧値にリセットすることもできます。
- ソースが A0 ～ D7 のときのトリガレベルは、ユーザーズマニュアル IM DLM6054-01JA の 52 節で設定したスレショルドレベルです。

ヒステリシス (Hys)

トリガレベルに幅を持たせて、小さな変動ではトリガがかからないようにします。

⩘	トリガレベルを中心に、約 0.3div* のヒステリシス
⩛	トリガレベルを中心に、約 1div* のヒステリシス

*　上記の数値は、おおよその値です。厳密に保証するものではありません。

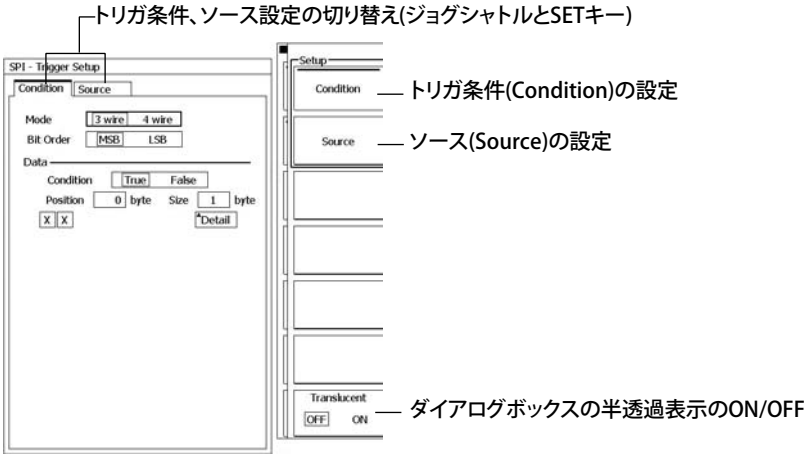
セットアップ / 解析 / 検索のヒステリシスとの関係は 2-7 ページをご覧ください。

3.4 SPI バス信号でトリガをかける

操　作

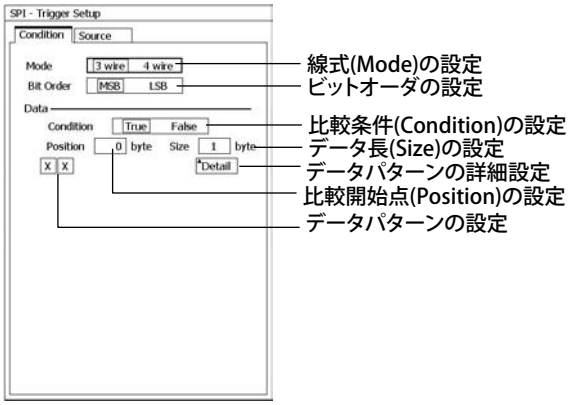
Setup メニュー

ENHANCED キー > Type のソフトキー > Serial Bus のソフトキー > SPI のソフトキー > Setup のソフトキーを押します。次のメニューが表示されます。

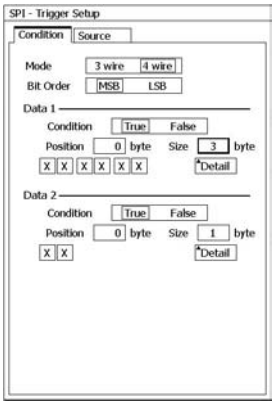


トリガ条件の設定 (Condition)

線式が3Wireのとき

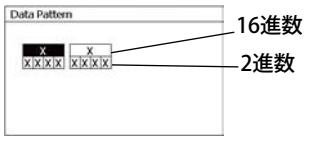


線式が4Wireのとき



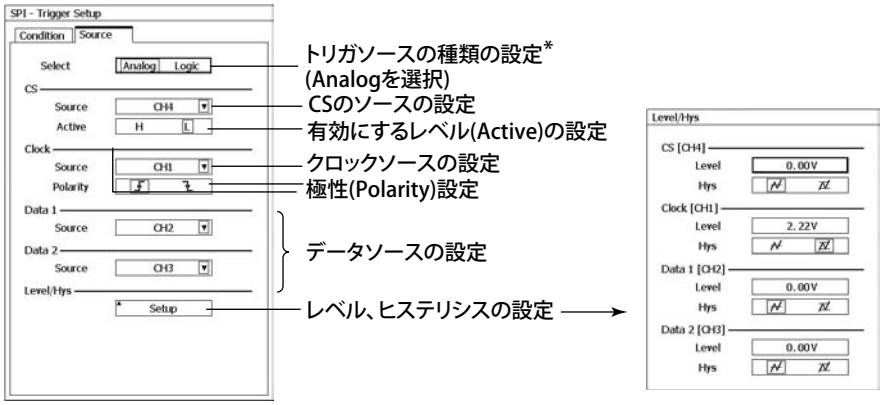
データパターンの詳細設定

Detailを選択して表示されるダイアログボックスで、ジョグシャトル、SETキー、ソフトキーを使ってデータパターンを設定できます。



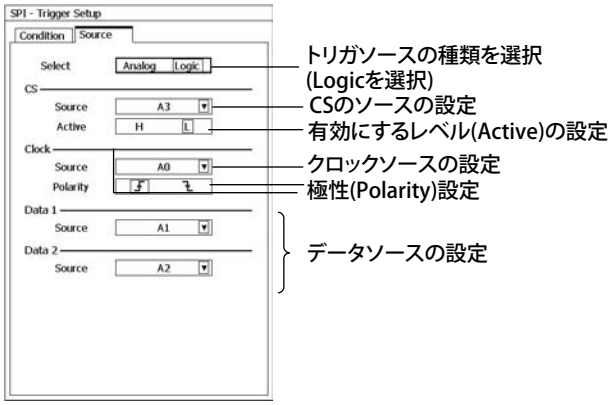
トリガソースの設定 (Source)

アナログ入力のとき



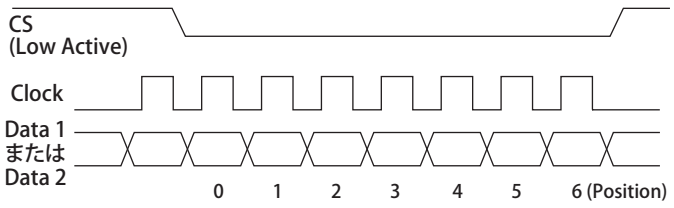
*：DL6000シリーズではAnalogに固定(設定無し)

ロジック入力のとき (DLM6000)



解　説

SPI バス信号でトリガをかける機能です。下図に SPI バス信号のタイムチャートを示します。



線式 (Mode)

次の中から選択できます。

3 wire	1 つの Data ラインのデータパターンを条件にトリガをかけます。
4 wire	Data 1 ラインと Data 2 ラインの 2 つのデータパターンを条件にトリガをかけます。 Data 1 と Data 2 それぞれを単独でトリガ条件にすることもできます。

ビットオーダ (Bit Order)

データの信号の流れに合わせて、ビットオーダを選択できます。

MSB	MSB からデータの信号が流れているとき
LSB	LSB からデータの信号が流れているとき

- 2 進数でデータパターンを設定するときは、ビットオーダの設定に関係なく、流れるデータの順でパターンを設定します。
- 16 進数でデータパターンを設定するときは、ビットオーダの設定に従って、流れるデータの順に 4 ビットずつ区切ってパターンを設定します。

データ (Data)

データパターンをトリガ条件として設定できます。

比較条件 (Condition)

設定したパターンと入力信号のパターンを比較して、選択した比較条件を満たしたとき、データのトリガ条件が成立したことになります。

True	パターンが一致したとき
False	パターンが一致しないとき

比較開始点 (Position)

比較開始点を設定します。たとえば CS 信号が Active になった最初のデータから開始するときは、「0」を設定します。

設定範囲：0 ～ 9999 バイト

データ長 (Size)

連続したデータを、何バイト分比較するかを設定します。

設定範囲：1 ～ 4 バイト

データパターン

Size で設定した長さのデータに対して、データパターンを 16 進数または 2 進数で設定します。

- パターンに X を設定すると、対応するビットの状態にかかわらず条件を満たしていると見なされます。
- 2 進のパターンに 1 つでも X があると、対応する 16 進の表示は「$」になります。

CS/Clock/ データ

CS(チップセレクト)/ Clock(クロック)/ データのソースを選択できます。

- DL6000 シリーズの場合、CH1 ～ CH4 から選択します。
- DLM6000 シリーズの場合、CH1 ～ CH4、A0 ～ A7、B0 ～ B7、C0 ～ C7、および D0 ～ D7 (16 ビットモデルは A0 ～ A7 と C0 ～ C7) から選択します。

CS

CS のレベルがどちらの状態のときに、データを有効 (Active) にするかを選択できます。

H	High レベルのとき
L	Low レベルのとき

Clock

Clock のどちらのエッジのタイミングで、データパターンを比較するかを選択できます。

⤴	立ち上がりのとき
⤵	立ち下がりのとき

トリガレベル (Level)

CS/Clock/ データ * が Analog(CH1 ～ CH4) のとき、ソースごとに、トリガレベルを設定できます。

- 設定範囲は画面内 8div 分で、設定分解能は 0.01div です。たとえば、2V/div のときの設定分解能は 0.02V です。
- RESET キーを押すことで、トリガレベルを現在のオフセット電圧値にリセットすることもできます。

* Logic(A0 ～ D7) のときは、ユーザーズマニュアル IMDLM6054-01JA の 5.2 節で設定したスレショルドレベルです。

ヒステリシス (Hys)

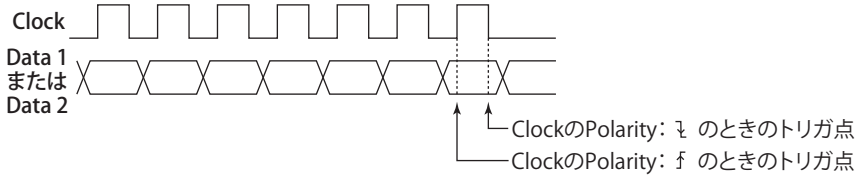
トリガレベルに幅を持たせて、小さな変動ではトリガがかからないようにします。

⩘	トリガレベルを中心に、約 0.3div* のヒステリシス
△	トリガレベルを中心に、約 1div* のヒステリシス

* 上記の数値は、おおよその値です。厳密に保証するものではありません。
セットアップ / 解析 / 検索のヒステリシスとの関係は 2-7 ページをご覧ください。

トリガ点

Clock の Polarity の設定によって、次の位置になります。

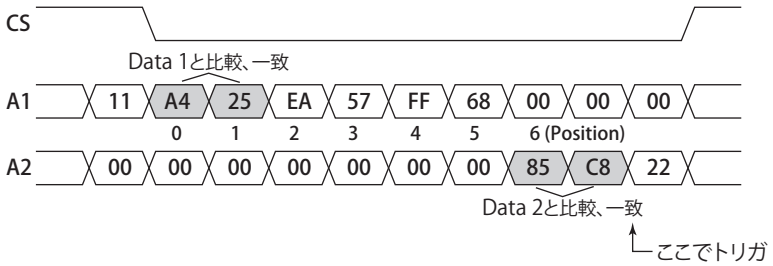


設定例

ここでは、データ列をバイト単位でヘキサ表示し、トリガのかかる位置を示します。
Data 1 のパターン比較対照として A1、Data 2 のパターン比較対照として A2 を選択しています。

網掛け：パターン比較対象

CS	Active：L
Data 1	Condition：True、Position：0、Size：2 bytes、データパターン：A4 と 25
Data 2	Condition：True、Position：6、Size：2 bytes、データパターン：85 と C8



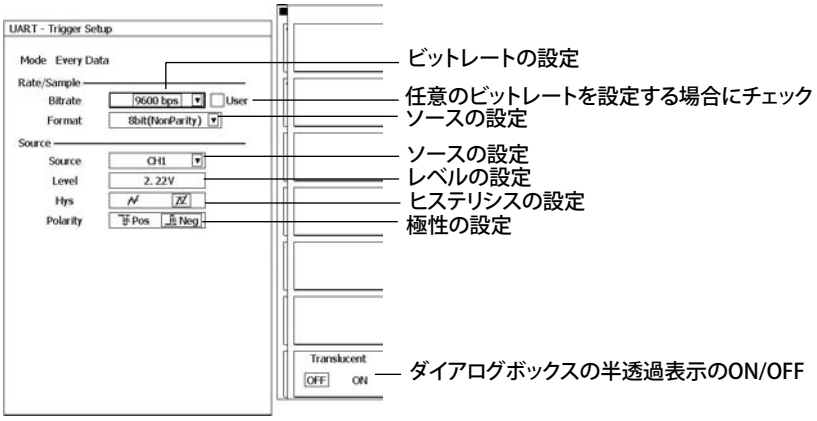
3.5 UART 信号でトリガをかける

操　作

Setup メニュー

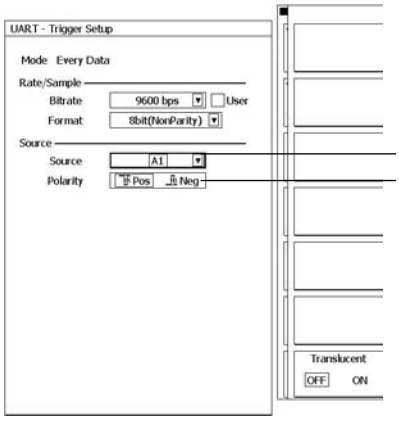
ENHANCED キー > Type のソフトキー > Serial Bus のソフトキー > UART のソフトキー > Setup のソフトキーを押します。次のメニューが表示されます。

ソースがアナログ信号(CH1～CH4)のとき



ビットレートの設定
任意のビットレートを設定する場合にチェック
ソースの設定
ソースの設定
レベルの設定
ヒステリシスの設定
極性の設定
ダイアログボックスの半透過表示のON/OFF

ソースがロジック信号(A0～D7)のとき



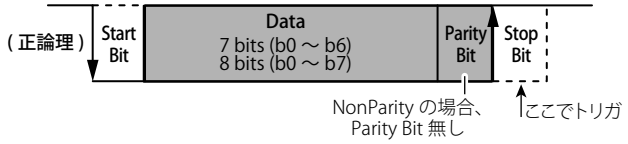
ソースの設定
極性の設定

解　説

UART 信号でトリガをかける機能です。

モード (Mode)

モードは、Every Data に固定です。すべてのデータの Stop Bit の位置でトリガがかかります。



ソース (Source)

ソースを選択できます。

- DL6000 シリーズの場合、CH1 ～ CH4 から選択します。
- DLM6000 シリーズの場合、CH1 ～ CH4、A0 ～ A7、B0 ～ B7、C0 ～ C7、および D0 ～ D7 (16 ビットモデルは A0 ～ A7 と C0 ～ C7) から選択します。

ビットレート (Bitrate)

UART バス信号の転送レートを次の中から選択できます。

115200bps、57600bps、38400bps、19200bps、9600bps、4800bps、2400bps、1200bps

User 設定を選択した場合は、1000bps ～ 200000bps の範囲 (設定分解能 100bps) で設定できます。

フォーマット (Foramat)

フォーマットを次の中から選択できます。

8bit + Parity	8 ビットのデータ＋ Parity Bit
7bit + Parity	7 ビットのデータ＋ Parity Bit
8bit(NonParity)	8 ビットのデータ (Parity Bit 無し)

トリガレベル (Level)

CH1 ～ CH4 ごとに、UART 信号のトリガレベルを設定できます。

- 設定範囲は画面内 8div 分で、設定分解能は 0.01div です。たとえば、2V/div のときの設定分解能は 0.02V です。
- RESET キーを押すことで、トリガレベルを現在のオフセット電圧値にリセットすることもできます。
- ソースが A0 ～ D7 のときのトリガレベルは、ユーザーズマニュアル IM DLM6054-01JA の 5.2 節で設定したスレショルドレベルです。

ヒステリシス (HyS)

トリガレベルに幅を持たせて、小さな変動ではトリガがかからないようにします。

⩘	トリガレベルを中心に、約 0.3div* のヒステリシス
△	トリガレベルを中心に、約 1div* のヒステリシス

*　上記の数値は、おおよその値です。厳密に保証するものではありません。

セットアップ / 解析 / 検索のヒステリシスとの関係は 2-7 ページをご覧ください。

極性 (Polarity)

ビットのどちらの状態を論理 1 として認識するかを選択できます。

Pos	正論理
Neg	負論理

Note

UART 信号トリガ機能使用時は、ホールドオフ時間の設定はできません。ホールドオフ時間については、ユーザーズマニュアル IM DLM6054-01JA の 6.1 節をご覧ください。

4.1　シリアルバス信号を選択する、解析結果を表示 / 保存する

操　作

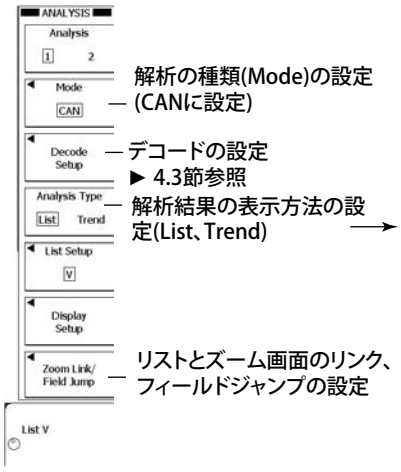
ANALYSIS メニュー

ANALYSIS キー > Mode のソフトキー > Serial Bus のソフトキー > 各シリアルバスのソフトキーを押します。以下のメニューが表示されます。

I2Cのとき

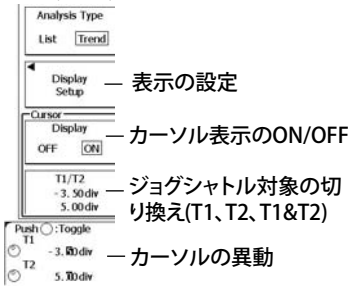
— 解析番号の設定(1、2)
— 解析の種類(Mode)の設定(I2Cに設定)
— デコードの設定 ▶ 4.2節参照
— リスト操作(List Setup)の設定
— 表示(Display Setup)の設定
— リストとズーム画面のリンクの設定
— リストのスクロール

CANのとき



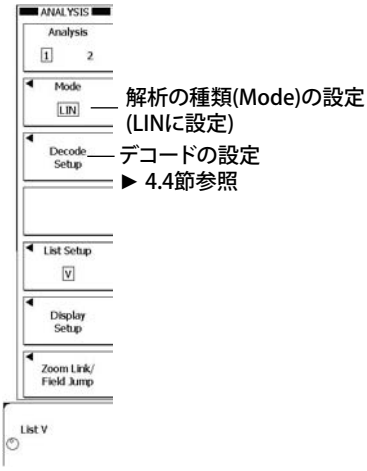
解析の種類(Mode)の設定(CANに設定)
— デコードの設定 ▶ 4.3節参照
— 解析結果の表示方法の設定(List、Trend)
— リストとズーム画面のリンク、フィールドジャンプの設定

Analysis TypeがTrendのとき



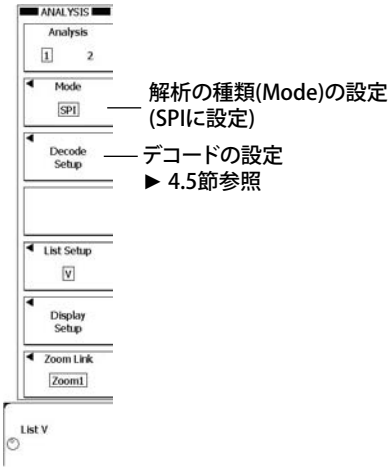
— 表示の設定
— カーソル表示のON/OFF
— ジョグシャトル対象の切り換え(T1、T2、T1&T2)
— カーソルの異動

LINのとき



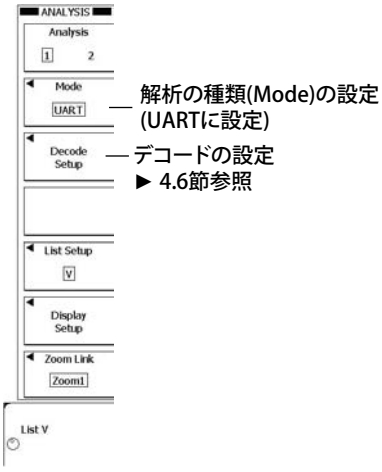
— 解析の種類(Mode)の設定(LINに設定)
— デコードの設定 ▶ 4.4節参照

SPIのとき



— 解析の種類(Mode)の設定(SPIに設定)
— デコードの設定 ▶ 4.5節参照

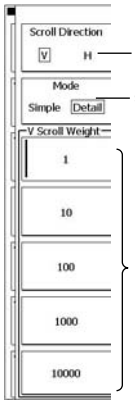
UARTのとき



— 解析の種類(Mode)の設定(UARTに設定)
— デコードの設定 ▶ 4.6節参照

リスト操作の設定 (List Setup)

List Setup のソフトキーを押します。次のメニューが表示されます。



スクロール方向の設定(V、H)

表示内容の設定(Simple、Detail)

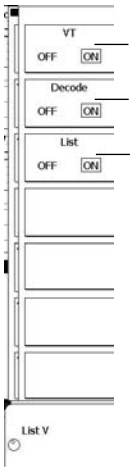
スクロール量の設定

Note

CAN バス信号で、AnalysisType の設定を Trend に設定した場合は操作できません。

表示の設定 (Display Setup)

Display Setup のソフトキーを押します。次のメニューが表示されます。

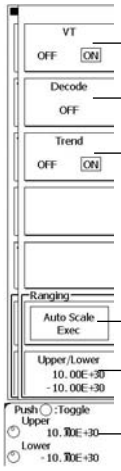


時間軸波形表示(VT)のON/OFF

デコード表示のON/OFF

リスト表示のON/OFF

CANバス信号の解析で、Analysis TypeがTrendのとき



時間軸波形表示のON/OFF

デコード表示のON/OFF

トレンド表示のON/OFF

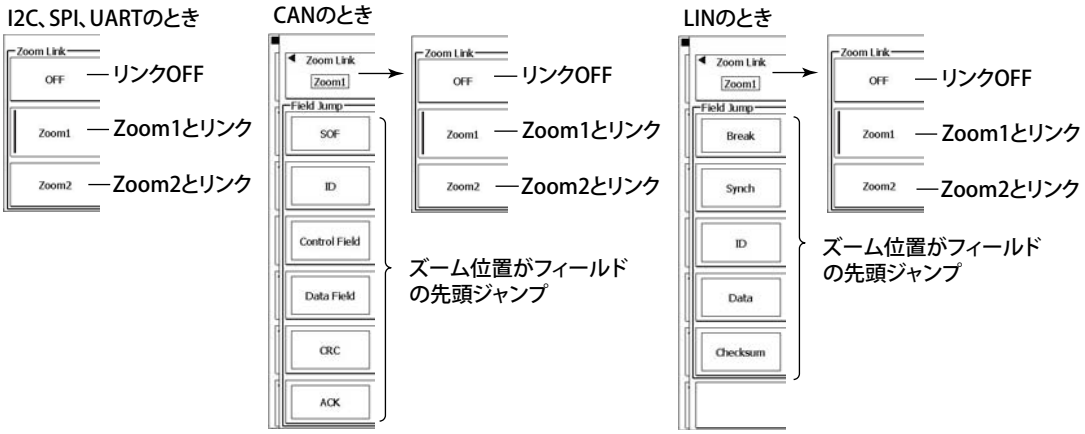
トレンド表示のオートスケール実行

ジョグシャトル対象の切り換え
(Upper、Lower)

トレンド表示の上限値(Upper)、
下限値(Lower)の設定

ズームリンク、フィールドジャンプ (CAN/LIN) の設定

I2C、SPI、UART のときは、**Zoom Link** のソフトキーを押します。
CAN、LIN のときは、**Zoom Link/Field Jump** のフトキーを押します。次の画面が表示されます。



Note

CAN バス信号で、AnalysisType の設定を Trend に設定した場合は操作できません。

解析結果を保存する

I^2C、CAN、LIN、SPI、および UART などのシリアスバスのリストを保存するには、ユーザーズマニュアル IM DLM6054-01JA の 13.6 節で、データタイプ「Serial Bus」を選択してください。

解　説

解析の種類 (Mode)

I²C、CAN、LIN、SPI、および UART のシリアルバス信号を解析します。

解析対象

次のデータを解析できます。

- 波形データ
  信号の取り込みをスタート / ストップしているときのどちらでも解析できます。信号の取り込みをスタートしているときは、表示波形と同期して解析結果が更新されます。
  また、ヒストリメモリに保存された波形データ (HISTORY メニュー > Select で選択したレコード No. の波形データ) も解析対象です。
- ロードしたアクイジションデータ (ACQ データ)

解析結果の表示

I²C、CAN、LIN、SPI、および UART の各信号の解析結果の内容について説明します。

I²C

- **解析可能データ数**

  最大 40000 バイト分 (解析基準点前後 20000 バイト)

- **簡易表示 (Simple)**

No.	解析番号
S/P	データの状態を表示します。S：スタートコンディション、P：ストップコンディション
Hex	Data の 16 進数表示
Form	アドレス / データ
R/W	信号の種類
ACK	Acknowledge ビットの状態

それぞれの説明は、「・詳細表示」をご覧ください。

- **詳細表示 (Detail)**

No.	解析番号。解析基準点 (Reference Point) よりも前は、No. − 1、No. − 2、・・・、解析基準点よりも後は、No.0、No.1、No.2、・・・となります。解析結果数は、− 19999 ～ 20000 までの範囲で最大 40000 データ表示できます。 RESET キーを押すと、No. 0 のデータがハイライト表示されます。
S/P	データの状態を表示します。S：スタートコンディション、P：ストップコンディション
Time(ms)	トリガポジションからバイトの先頭までの時間を ms で表示します。
Binary	Data を 2 進数で表示します。
Hex	Data を 16 進数で表示します。
Form	アドレスかデータかを表示します。A：アドレス、D：データ
R/W	信号の種類を表示します。R：Read、W：Write
ACK	Acknowledge が検出された場合は「1」、Acknowledge が検出されなかった場合は「0」になります。
Info	データのタイプを表示します。

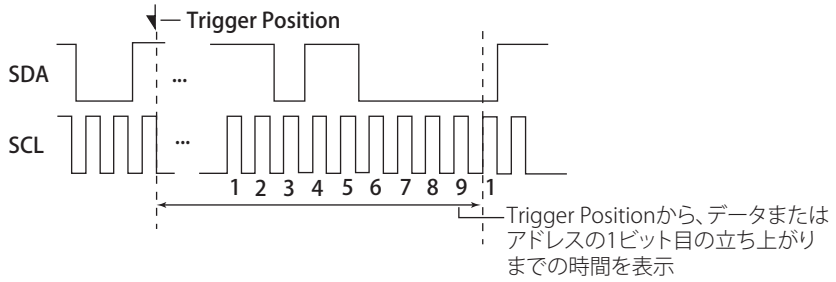
簡易表示

No.	S/P	Hex	Form	R/W	ACK
9	P	4B	D		1
10	S	F8	A	W	0
11		EB	D		0
12	P	21	D		0
13	S	9D	A	R	0
14		66	D		0
15	P	E2	D		1
16	S	38	A	W	0
17		9B	D		0
18	P	2C	D		0
19	S	6D	A	R	0
20		D7	D		0
21	P	4E	D		1

詳細表示

No.	S/P	Time(ms)	Binary	Hex	Form	R/W	ACK	Info
1	S	-2.46700	01110101	75	A	R	0	7-bit
2		-2.43100	10111100	BC	D		0	
3	P	-2.39500	11101111	EF	D		1	
4	S	-1.32304	00111000	38	A	W	0	7-bit
5		-1.28704	01010011	53	D		0	
6	P	-1.25104	10101001	A9	D		0	
7	S	-1.13904	10011101	9D	A	R	0	7-bit
8		-1.10304	00010000	10	D		0	
9	P	-1.06704	01001011	4B	D		1	
10	S	0.00496	11111000	F8	A	W	0	7-bit
11		0.04096	11101011	EB	D		0	
12	P	0.07696	00100001	21	D		0	
13	S	0.18896	10011101	9D	A	R	0	7-bit

Time(ms) の表示時間について



- **デコード (復号) 表示**

  各データをデコードして、それぞれの色で表示します。ソースが CH1 ～ CH4 または M1 ～ M4 のときに適用できます。

Adr	淡緑色 (Light Green) の 16 進数の値
Data	青緑色 (Cyan) の 16 進数の値
R/W	桃色 (Pink)
Ack	黄色 (Yellow)
General Call	緑色 (Green)
Start Byte	橙色 (Orange)
HS Mode	橙色 (Orange)

CAN

・ 解析可能フレーム数
最大 3000 フレーム分 (解析基準点前後 1500 フレーム)

・ **解析対象フレーム**

リモートフレーム	ID の値、Data の値、CRC の値、ACK の有無を検出します。
データフレーム	ID の値、CRC の値、ACK の有無を検出します。
エラーフレーム	エラーフラグを検出します。
オーバーロードフレーム	オーバーロードフラグを検出します。

・ **簡易表示 (Simple)**

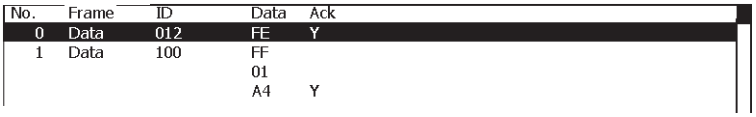
No.	解析番号
Frame	フレームの種類
ID	ID の 16 進数表示
Data	Data の 16 進数表示
Ack	ACK スロットの状態

それぞれの説明は、「・詳細表示」をご覧ください。

・ **詳細表示 (Detail)**

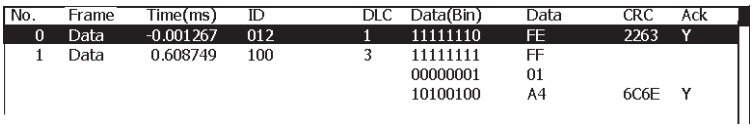
No.	解析番号。解析基準点 (Reference Point) よりも前は、No. − 1、No. − 2、・・・、解析基準点よりも後は、No.0、No.1、No.2、・・・となります。解析結果数は、− 1499 ～ 1500 までの範囲で最大 3000 フレーム表示できます。 RESET キーを押すと、No. 0 のフレームがハイライト表示されます
Frame	フレームの種類を示します。解析できるフレームは、データフレーム (Data)、リモートフレーム (Remote)、エラーフレーム (Error)、およびオーバーロードフレーム (Over load) の 4 種類です。
Time(ms)	トリガポジションからフレームの先頭までの時間を ms で表示します。
ID	標準フォーマット (11 ビット) または拡張フォーマット (29 ビット) の ID の値を 16 進数で表示します。
DLC	有効バイト数を 16 進数表示します。
Data(Bin)	フレームの種類がデータフレームの場合だけ、Data を 2 進数で表示します。1 バイトごとに改行されて表示されます。
Data	フレームの種類がデータフレームの場合だけ、Data を 16 進数で表示します。1 バイトごとに改行されて表示されます。
CRC	CRC シーケンスを 16 進数で表示します。フレームの種類がデータフレームまたはリモートフレームの場合に表示されます。
Ack	ACK が検出された場合は「Y」、ACK が検出されなかった場合は「N」になります。

簡易表示



No.	Frame	ID	Data	Ack
0	Data	012	FE	Y
1	Data	100	FF	
			01	
			A4	Y

詳細表示



No.	Frame	Time(ms)	ID	DLC	Data(Bin)	Data	CRC	Ack
0	Data	-0.001267	012	1	11111110	FE	2263	Y
1	Data	0.608749	100	3	11111111	FF		
					00000001	01		
					10100100	A4	6C6E	Y

・ **デコード (復号) 表示**

各フィールドの値をデコードして、それぞれの色で表示します。

ID	淡緑色 (Light Green)
DLC	桃色 (Pink)
Data	青緑色 (Cyan)
CRC シーケンス	淡青色 (Light Blue)
アラームフレーム	赤色 (Red)
オーバーロードフレーム	緑色 (Green)
フレームの背景	灰色 (Gray)
スタッフビット	灰色 (Gray) の塗りつぶし

LIN

- **解析可能フレーム数**

  最大 3000 フレーム分 (解析基準点前後 1500 フレーム)

- **解析対象フィールド**

  Break、Synch、ID、Data、Checksum

- **簡易表示 (Simple)**

No.	解析番号
ID	ID の 16 進数表示
Data	Data の 16 進数表示
Checksum	Checksum の 16 進数表示

それぞれの説明は、「・詳細表示」をご覧ください。

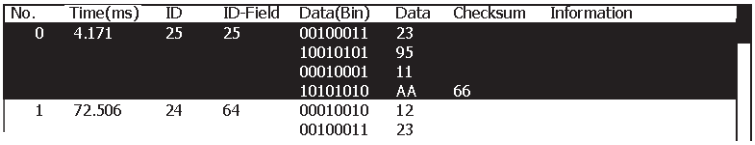
- **詳細表示 (Detail)**

No.	解析番号。解析基準点 (Reference Point) よりも前は、No. − 1、No. − 2、・・・、解析基準点よりも後は、No.0、No.1、No.2、・・・となります。解析結果数は、− 1499 ～ 1500 までの範囲で最大 3000 フレーム表示できます。 RESET キーを押すと、No. 0 のフレームがハイライト表示されます。
Time(ms)	トリガポジションからフレームの先頭までの時間を ms で表示します。
ID	ID の値を 16 進数で表示します。
ID-Field	ID の値をパリティ (2 ビット) を含めて 16 進数で表示します。
Data(Bin)	Data を 2 進数で表示します。1 バイトごとに改行されて表示されます。
Data	Data を 16 進数で表示します。1 バイトごとに改行されて表示されます。
Checksum	Checksum の値を 16 進数で表示します。
Information	それぞれの情報を検出して、次の文字を表示します。WakeUp 信号を検出した場合は、WakeUp を表示します。1 つのフレームで複数のエラーを検出した場合、下記の順に優先順位が高いエラーを 1 つだけ表示します。 Timeout Error、Framing Error 、Checksum Error、Synch Error、Parity Error

簡易表示

No.	ID	Data	Checksum
0	25	23	
		95	
		11	
		AA	66
1	24	12	
		23	

詳細表示



No.	Time(ms)	ID	ID-Field	Data(Bin)	Data	Checksum	Information
0	4.171	25	25	00100011	23		
				10010101	95		
				00010001	11		
				10101010	AA	66	
1	72.506	24	64	00010010	12		
				00100011	23		

- **デコード (復号) 表示**

  各フィールドの値をデコードして、それぞれの色で表示します。

Break	橙色 (Orange)
Synch	桃色 (Pink)
ID	淡緑色 (Light Green)
Data	青緑色 (Cyan)
Checksum	淡青色 (Light Blue)
WakeUp	緑色 (Green)
Start Bit	灰色 (Gray) の塗りつぶし
Stop Bit	灰色 (Gray) の塗りつぶし
Error	赤色 (Red) ・ Timeout Error　エラーが発生した区間の連結ラインを赤色の太線で表示。 ・ Framing Error　エラーが発生したフィールドに「Framing Error」の文字を黒色で、背景は赤色で表示。Checksum Error/Synch Error/Parity Error よりも優先して表示される。 ・ Checksum Error、Synch Error、Parity Error　エラーが発生した Synch/ID/Checksum のフィールドの文字を黒色で、背景は赤色で表示。

SPI

- **解析可能データ数**

  最大 40000 バイト分 (解析基準点前後 20000 バイト)

- **簡易表示 (Simple)**

No.	解析番号
Data 1(H)	Data 1 の 16 進数表示
Data 2(H)	Data 2 の 16 進数表示
CS	CS のステータスを示します。

それぞれの説明は、「・詳細表示」をご覧ください。

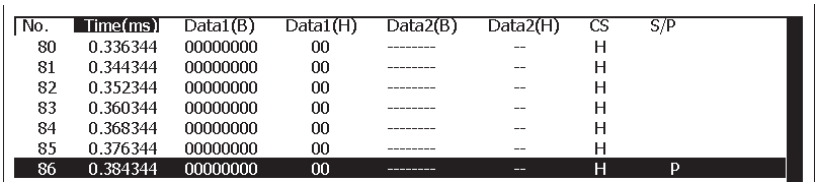
- **詳細表示 (Detail)**

No.	解析番号。解析基準点 (Reference Point) よりも前は、No. − 1、No. − 2、・・・、解析基準点よりも後は、No.0、No.1、No.2、・・・となります。解析結果数は、− 19999 ～ 20000 までの範囲で最大 40000 データ表示できます。 RESET キーを押すと、No. 0 のデータがハイライト表示されます。
Time(ms)	トリガポジションからバイトの先頭までの時間を ms で表示します。
Data 1(B)	Data 1 を 2 進数で表示します。
Data 1(H)	Data 1 を 16 進数で表示します。
Data 2(B)	Data 2 を 2 進数で表示します。
Data 2(H)	Data 2 を 16 進数で表示します。
CS	CS のステータスを示します。
S/P	アクティブ期間を示します。S: 開始位置、P：終了位置

簡易表示

No.	Data1(H)	Data2(H)	CS
80	00	--	H
81	00	--	H
82	00	--	H
83	00	--	H
84	00	--	H
85	00	--	H
86	00	--	H

詳細表示



No.	Time(ms)	Data1(B)	Data1(H)	Data2(B)	Data2(H)	CS	S/P
80	0.336344	00000000	00	--------	--	H	
81	0.344344	00000000	00	--------	--	H	
82	0.352344	00000000	00	--------	--	H	
83	0.360344	00000000	00	--------	--	H	
84	0.368344	00000000	00	--------	--	H	
85	0.376344	00000000	00	--------	--	H	
86	0.384344	00000000	00	--------	--	H	P

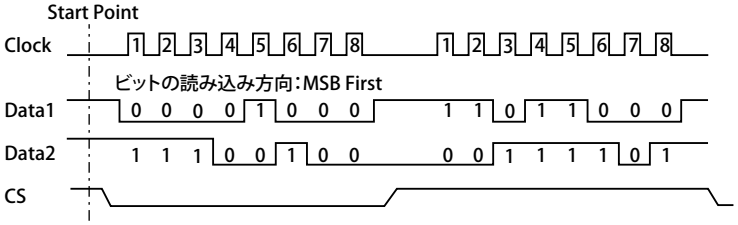
- **デコード (復号) 表示**

  各データをデコードして、それぞれの色で表示します。ソースが CH1 ～ CH4 または M1 ～ M4 のときに適用できます。

Data	青緑色 (Cyan) の 16 進数の値
グループの背景	灰色 (Gray)

- **表示例**

  解析条件を変えて解析したときの例を以下に示します。



解析条件　Clock(CH1)= ↓、CS(CH4)=L のとき	
項目	表示
解析番号 (No.)	0
Data 1 の 16 進数表示	08
Data 2 の 16 進数表示	E4
CS 信号のステータス	

解析条件　Clock(CH1)= ↓、CS(CH4)=H のとき	
項目	表示
解析番号 (No.)	0
Data 1 の 16 進数表示	D8
Data 2 の 16 進数表示	3D
CS 信号のステータス	H

Note

CS に H → L または L → H に切り替わる変化点がないとき、入出力データは解析されません。

UART

- **解析可能データ数**

  最大 3000 バイト分 (解析基準点前後 1500 バイト)

- **解析対象フィールド**

  Data

- **簡易表示 (Simple)**

No.	解析番号
Data/ASCII	Data の 16 進数表示 /ASCII 表示
Information	Error を表示

それぞれの説明は、「・詳細表示」をご覧ください。

- **詳細表示 (Detail)**

No.	解析番号。解析基準点 (Reference Point) よりも前は、No. − 1、No. − 2、・・・、解析基準点よりも後は、No.0、No.1、No.2、・・・となります。解析結果数は、− 1499 ～ 1500 までの範囲で最大 3000 データ表示できます。 RESET キーを押すと、No. 0 のデータがハイライト表示されます。
Time(ms)	トリガポジションからバイトの先頭までの時間を ms で表示します。
Size	表示モードのグルーピングを ON にしたときだけ、Data のバイト数を表示します。
Data(Bin)	Data を 2 進数で表示します。
Data/ASCII	Data を 16 進数で表示します。表示モードを「ASCII」にすると、 ASCII 表示になります。
Information	次のエラー情報を表示します。1 つのデータで複数のエラーを検出した場合、下記の順に優先順位が高いエラーを 1 つだけ表示します。 Framing Error、Parity Error

簡易表示

No.	Data	Information
0	30	
1	53	
2	74	
3	61	
4	72	
5	74	

簡易表示 - ASCII

No.	ASCII	Information
0	0	
1	S	
2	t	
3	a	
4	r	
5	t	

簡易表示 - グルーピングON

No.	Data(Hex)
0	47 50 53 32 30 30
1	53 74 61 72 74 5F 41
2	33 36 3A 31 36 35
3	31 34 32 3A 35 33 31
4	47 50 53 32 30 30
5	53 74 61 72 74 5F 41

詳細表示

No.	Time(ms)	Data(Bin)	Data	Information
0	-0.4944	00110000	30	
1	1.0168	01010011	53	
2	1.5872	01110100	74	
3	2.1616	01100001	61	
4	2.7348	01110010	72	
5	3.3072	01110100	74	

詳細表示 - ASCII

No.	Time(ms)	ASCII	Information
0	-0.4944	0	
1	1.0168	S	
2	1.5872	t	
3	2.1616	a	
4	2.7348	r	
5	3.3072	t	

詳細表示 - グルーピングON

No.	Time(ms)	Size	Data(Hex)
0	-80.912	6	47 50 53 32 30 30
1	-80.339	7	53 74 61 72 74 5F 41
2	-79.765	6	33 36 3A 31 36 35
3	-79.192	7	31 34 32 3A 35 33 31
4	-78.621	6	47 50 53 32 30 30
5	-77.110	7	53 74 61 72 74 5F 41

- **デコード (復号) 表示**

  各フィールドの値をデコードして、それぞれの色で表示します。

Data	青緑色 (Cyan)
Parity	黄色 (Yellow)
Start Bit	灰色 (Gray) の塗りつぶし
Stop Bit	灰色 (Gray) の塗りつぶし
Error	赤色 (Red) ・ Framing Error　エラーが発生したフィールドに「Framing Error」の文字を黒色で、背景は赤色で表示。Parity Error よりも優先して表示される。 ・ Parity Error　エラーが発生したフィールドの文字を黒色で、背景は赤色で表示。

ズームリンク (Zoom Link)

次の中から選択します。

OFF	ズームリンク機能を無効にします。
Zoom1	Zoom1 にリンクします。
Zoom2	Zoom2 にリンクします。

初期設定は、Analysis1 では Zoom1、Analysis 2 では Zoom2 です。Zoom1 または Zoom2 を選択した場合、解析結果のリストで任意のバイトを選択する (ハイライト表示させる) と、そのバイトの先頭に Zoom1 または Zoom2 のズーム位置 (ズームボックスの中心) が移動します。逆に、Zoom1 または Zoom2 のズーム位置を変更すると、解析結果リストのハイライト表示が Zoom1 または Zoom2 のズームボックス内に表示されているバイトに移動します。

フィールドジャンプ (Field Jump)

解析タイプが CAN/LIN のとき、ズームリンク機能が有効の場合、解析結果のリストでハイライト表示されたフレームの指定したフィールドの先頭にズーム位置をジャンプさせることができます。

解析結果の保存

ストレージメディアに解析結果 (簡易表示 (Simple) と詳細表示 (Detail) のデータ) を CSV 形式で保存できます。拡張子は .csv です。
表示されるそれぞれのリストに準じたフォーマットで保存されます。

保存対象

ユーザーズマニュアル IM DLM6054-01JA の 13.8 節で、データタイプに「Serial Bus」を選択したときに表示されるメニューで、保存対象を Analysis1 または Analysis2 から選択します。

Analysis1	Analysis1 で設定した条件で求められた解析結果を保存対象にします。
Analysis2	Analysis2 で設定した条件で求められた解析結果を保存対象にします。

データサイズ

I^2C	(解析結果のバイト数＋ 4) × 65 [バイト]
CAN	(解析結果のフレーム数＋ 4) × 155 [バイト]
LIN	(解析結果のフレーム数＋ 4) × 170 [バイト]
SPI	(解析結果のバイト数＋ 4) × 79 [バイト]
UART	(解析結果のフレーム数＋ 4) × 40 [バイト]

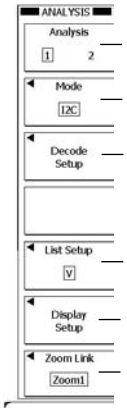
* データサイズは参考値です。厳密に保証するものではありません。データを保存する際の目安としてご利用ください。

4.2 I²C バス信号を解析する

操　作

ANALYSIS メニュー

ANALYSIS キー > Mode のソフトキー > Serial Bus のソフトキー > I2C のソフトキーを押します。以下のメニューが表示されます。

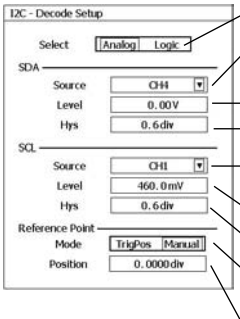


- 解析番号の設定(1、2)
- 解析の種類(Mode)の設定 (I2Cに設定)
- デコードの設定
- リスト操作(List Setup)の設定 ► 4.1節参照
- 表示(Display Setup)の設定 ► 4.1節参照
- リストとズーム画面のリンクの設定 ► 4.1節参照
- リストのスクロール ► 4.1節参照

デコードの設定 (Decode Setup)

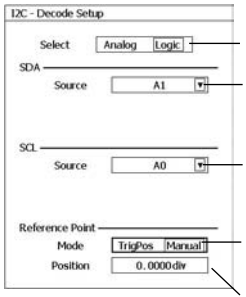
Decode Setup のソフトキーを押します。以下のメニューが表示されます。

アナログ信号のとき



- アナログ/ロジックの設定*(Analogに設定)
- SDAのソースの設定 (CH1～CH4、M1～M4)
- レベルの設定
- ヒステリシスの設定
- SCLのソースの設定 (CH1～CH4、M1～M4)
- レベルの設定
- ヒステリシスの設定
- 解析基準点の設定 (TrigPos、Manual)
- 解析基準点をManualにしたときの基準点の設定(±5.00div)

ロジック信号のとき(DLM6000)



- アナログ/ロジックの設定*(Logicに設定)
- SDAのソースの設定
- SCLのソースの設定
- 解析基準点の設定 (TrigPos、Manual)
- 解析基準点をManualにしたときの基準点の設定(±5.00div)

*：DL6000シリーズではAnalogに固定(設定無し)

解　説

SDA/SCL のソース (Source)

SDA(シリアルデータ)/ SCL(シリアルクロック) のソースを選択できます。

- DL6000 シリーズの場合、、CH1 ～ CH4 または M1 ～ M4 から選択します。
- DLM6000 シリーズの場合、CH1 ～ CH4、M1 ～ M4、A0 ～ A7、B0 ～ B7、C0 ～ C7、および D0 ～ D7 (16 ビットモデルは A0 ～ A7 と C0 ～ C7) から選択します。

レベル (Level)

CH1 ～ CH4 と M1 ～ M4 に対して、信号レベルが 0 か 1 かを判定するレベル * を設定します。
設定範囲は垂直ポジションを中心に± 10div 分で、設定分解能は 0.01div です。 たとえば、2V/div のときの設定分解能は 0.02V です。

* Logic(A0 ～ D7) のときは、ユーザーズマニュアル IM DLM6054-01JA の 5.2 節で設定したスレショルドレベルです。

ヒステリシス (Hys)

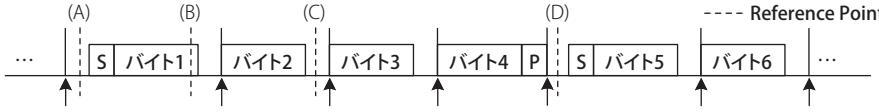
CH1 ～ CH4 と M1 ～ M4 に対して、設定します。
設定範囲は 0.0 ～ 4.0div で、設定分解能は 0.1div です。
トリガのヒステリシスとの関係は 2-7 ページをご覧ください。

解析基準点 (Reference Point)

解析基準点を次の中から選択します。

Trig Pos	トリガポジションを解析基準点に設定します。
Manual	手動で解析基準点を設定します。設定範囲：± 5.00div、設定分解能：0.01div

解析基準点(Reference Point)の位置によって、解析結果リストのNo. 0のバイトは下記のようになります。



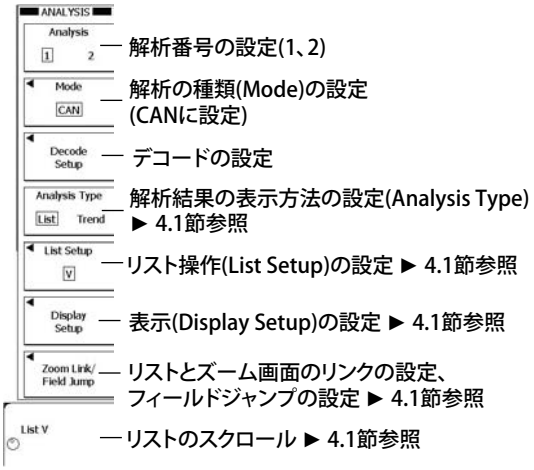
(A)：No. 0のバイト→バイト1 (バイト2がNo. 1、バイト3がNo. 2、・・・)
(B)：No. 0のバイト→バイト1 (バイト2がNo. 1、バイト3がNo. 2、・・・)
(C)：No. 0のバイト→バイト2 (バイト1がNo. -1、バイト3がNo. 1、バイト4がNo. 2、・・・)
(D)：No. 0のバイト→バイト5 (バイト1がNo. -4、・・・、バイト4がNo. -1、バイト6がNo. 1、・・・)
S：スタートコンディション、**P**：ストップコンディション

4.3 CAN バス信号を解析する

操　作

ANALYSIS メニュー

ANALYSIS キー > Mode のソフトキー > Serial Bus のソフトキー > CAN のソフトキーを押します。以下のメニューが表示されます。



解析番号の設定(1、2)

解析の種類(Mode)の設定
(CANに設定)

デコードの設定

解析結果の表示方法の設定(Analysis Type)
▶ 4.1節参照

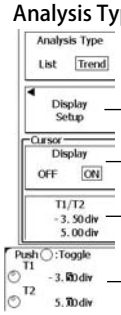
リスト操作(List Setup)の設定 ▶ 4.1節参照

表示(Display Setup)の設定 ▶ 4.1節参照

リストとズーム画面のリンクの設定、
フィールドジャンプの設定 ▶ 4.1節参照

リストのスクロール ▶ 4.1節参照

Analysis TypeがTrendのとき



表示の設定 ▶ 4.1節参照

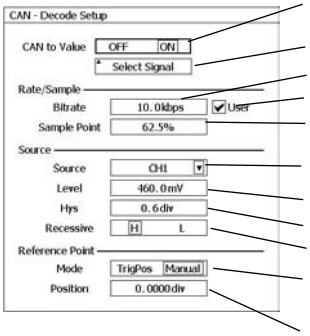
カーソル表示のON/OFF ▶ 4.1節参照

ジョグシャトル対象の切り換え
(T1、T2、T1&T2) ▶ 4.1節参照

カーソルの異動 ▶ 4.1節参照

デコードの設定 (Decode Setup)

Decode Setup のソフトキーを押します。以下のメニューが表示されます。



Signalを比較する(ON)/しない(OFF)

Signalの選択

ビットレートの設定

任意のビットレートを設定する場合にチェック

サンプルポイントの設定

ソースの設定(CH1～CH4、M1～M4)

レベルの設定

ヒステリシスの設定

判定基準の設定(H、L)

解析基準点の設定
(TrigPos、Manual)

解析基準点をManualにしたとき
の基準点の設定(±5.00div)

解　説

Signal のシンボル表示 (CAN to Value)

Can to Value を ON にすると、本機器に読み込まれている物理値 / シンボル定義ファイル (.sbl)* を指定できます。指定した物理値 / シンボル定義ファイルの Signal と合致した場合、合致した Signal のシンボルを表示できます。

整数型 (Integer) と浮動小数点型 (Float、32 ビットと 16 ビット) のデータタイプに対応しています。

*：物理値 / シンボル定義ファイルは、CANdb ファイル (.dbc) を変換したものです。当社のフリーソフトウエア「Symbol Editor」で変換できます。Symbol Editor は、当者 Web ページからダウンロードできます。

ビットレート (Bitrate)

CAN バス信号の転送レートを次の中から選択できます。

1Mbps、500kbps、250kbps、125kbps、83.3kbps、33.3kbps

User 設定を選択した場合は、10.0kbps ～ 1.000Mbps の範囲 (設定分解能 0.1kbps) で設定できます。

サンプルポイント (Sample Point)

バスレベル (リセッシブ / ドミナント) を判定するポイントを 18.8 ～ 90.6[%] の範囲 (設定分解能 3.1%) で設定できます。

本機器の CAN バス信号のトリガ回路では、入力された CAN バス信号を内部クロックでサンプリングして、リセッシブからドミナントへの変化点を検出しています。検出された変化点を 0% とし、変化点からビットタイム (設定したビットレートの逆数) が経過したところを 100% として、サンプルポイントを % で設定します。 3-17 ページの説明図をご覧ください。

ソース (Source)

CH1 ～ CH4 または M1 ～ M4 から選択できます。

レベル (Level)

信号レベルが 0 か 1 かを判定するレベルを設定します。

設定範囲は垂直ポジションを中心に± 10div 分で、設定分解能は 0.01div です。 たとえば、2V/div のときの設定分解能は 0.02V です。

ヒステリシス (Hys)

設定範囲は 0.0 ～ 4.0div で、設定分解能は 0.1div です。

トリガのヒステリシスとの関係は 2-7 ページをご覧ください。

リセッシブ電位 (Recessive)

リセッシブ電位を次の中から選択します。どちらの設定でも論理値は、リセッシブ＝ 1、ドミナント＝ 0 です。

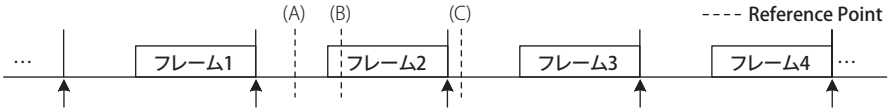
H	リセッシブ電位がドミナント電位より高い
L	リセッシブ電位がドミナント電位より低い

解析基準点 (Reference Point)

解析基準点を次の中から選択します。

Trig Pos	トリガポジションを解析基準点に設定します。
Manual	手動で解析基準点を設定します。設定範囲：± 5.00div、設定分解能：0.01div

解析基準点(Reference Point)の位置によって、解析結果リストのNo. 0のフレームは下記のようになります。



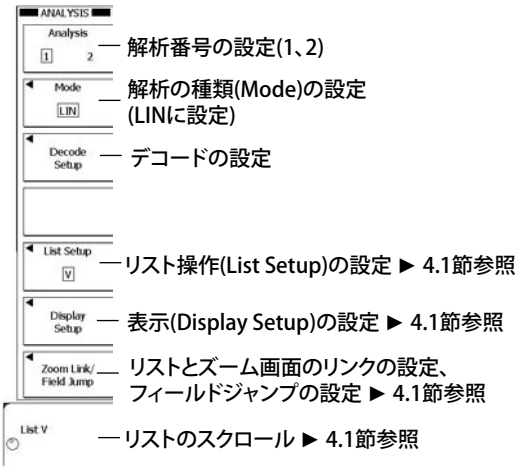
(A)： No. 0のフレーム→フレーム2 (フレーム1がNo. -1、フレーム3がNo. 1、フレーム4がNo. 2)
(B)： No. 0のフレーム→フレーム2 (フレーム1がNo. -1、フレーム3がNo. 1、フレーム4がNo. 2)
(C)： No. 0のフレーム→フレーム3 (フレーム1がNo. -2、フレーム2がNo. -1、フレーム4がNo. 1)

4.4 LIN バス信号を解析する

解　析

ANALYSIS メニュー

ANALYSIS キー > Mode のソフトキー > Serial Bus のソフトキー > LIN のソフトキーを押します。以下のメニューが表示されます。



解析番号の設定(1、2)

解析の種類(Mode)の設定
(LINに設定)

デコードの設定

リスト操作(List Setup)の設定 ► 4.1節参照

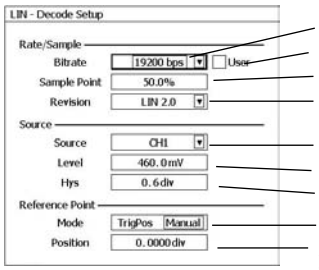
表示(Display Setup)の設定 ► 4.1節参照

リストとズーム画面のリンクの設定、
フィールドジャンプの設定 ► 4.1節参照

リストのスクロール ► 4.1節参照

デコードの設定 (Decode Setup)

Decode Setup のソフトキーを押します。以下のメニューが表示されます。



ビットレートの設定

任意のビットレートを設定する場合にチェック

サンプルポイントの設定

レビジョンの設定(LIN 2.0、LIN 1.3、Both)

ソースの設定(CH1～CH4、M1～M4、A0～D7)

レベルの設定

ヒステリシスの設定

解析基準点の設定(TrigPos、Manual)

解析基準点をManualにしたとき
の基準点の設定(±5.00div)

解　説

ビットレート (Bitrate)

LIN バス信号の転送レートを次の中から選択できます。
19200bps、9600bps、4800bps、2400bps、1200bps
User 設定を選択した場合は、1000bps ～ 20000bps の範囲 (設定分解能 10bps) で設定できます。

サンプルポイント (Sample Point)

バスレベルを判定するポイントを 18.8 ～ 90.6[%] の範囲 (設定分解能 3.1%) で設定できます。
本機器の LIN バス信号のトリガ回路では、入力された LIN バス信号を内部クロックでサンプリングして、レベルの変化点を検出しています。検出された変化点を 0% とし、変化点からビットタイム (設定したビットレートの逆数) が経過したところを 100% として、サンプルポイントを % で設定します。
3-17 ページの説明図をご覧ください。

レビジョン (Revision)

レビジョンを次の中から選択できます。Enhanced Checksum または Classic Checksum のどちらのエラーを検知するかを選択します。エラーの詳細については、5.4 節の解説をご覧ください。

LIN 2.0	保護 ID を含む Enhanced Checksum のエラーを検知します。 (ただし、ID=60(0x3c) ～ 63(0x3f) は Classic Checksum のエラーを検知します。)
LIN 1.3	Data Field だけの Classic Checksum のエラーを検知します。
Both	LIN 2.0 と LIN 1.3 の両方の Checksum エラーを検知したとき、エラーとします。どちらかだけの場合は、エラーになりません

ソース (source)

ソースを選択できます。

- DL6000 シリーズの場合、CH1 ～ CH4 または M1 ～ M4 から選択します。
- DLM6000 シリーズの場合、CH1 ～ CH4、M1 ～ M4、A0 ～ A7、B0 ～ B7、C0 ～ C7、および D0 ～ D7 (16 ビットモデルは A0 ～ A7 と C0 ～ C7) から選択します。

レベル (lLevel)

信号レベルが 0 か 1 かを判定するレベル * を設定します。
設定範囲は垂直ポジションを中心に± 10div 分で、設定分解能は 0.01div です。 たとえば、2V/div のときの設定分解能は 0.02V です。

* Logic(A0 ～ D7) のときは、ユーザーズマニュアル IM DLM6054-01JA の 5.2 節で設定したスレショルドレベルです。

ヒステリシス (Hys)

設定範囲は 0.0 ～ 4.0div で、設定分解能は 0.1div です。
トリガのヒステリシスとの関係は 2-7 ページをご覧ください。

解析基準点 (Reference Point)

解析基準点を次の中から選択します。

Trig Pos	トリガポジションを解析基準点に設定します。
Manual	手動で解析基準点を設定します。設定範囲：± 5.00div、設定分解能：0.01div

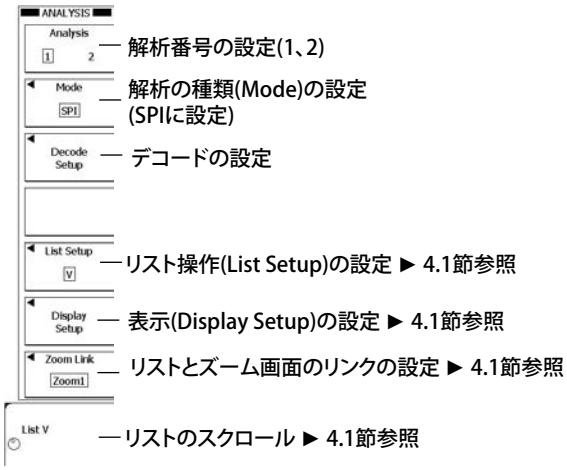
解析基準点と解析結果リストの No. については、4-16 ページの説明図をご覧ください。

4.5 SPI バス信号を解析する

操　作

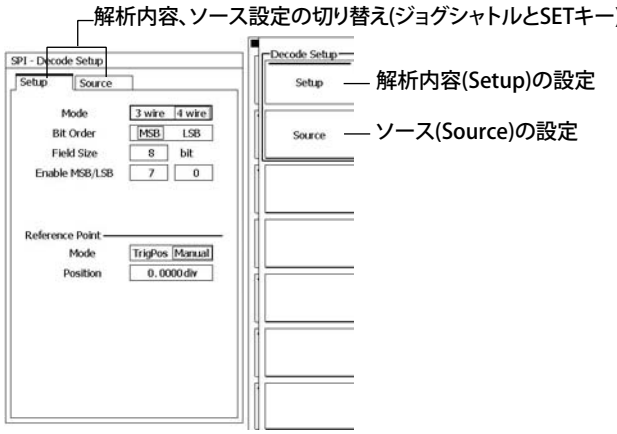
ANALYSIS メニュー

ANALYSIS キー > Mode のソフトキー > Serial Bus のソフトキー > SPI のソフトキーを押します。以下のメニューが表示されます。

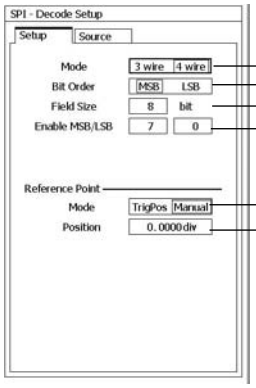


デコードの設定 (Decode Setup)

Decode Setup のソフトキーを押します。以下のメニューが表示されます。



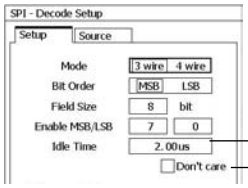
解析内容の設定 (Setup)



線式の設定(3 wire、4 wire)
ビットオーダの設定(MSB、LSB)
フィールドサイズの設定(4～32ビット)
有効ビット範囲の設定(設定範囲はフィールドサイズの設定による)

解析基準点の設定(TrigPos、Manual)
解析基準点をManualにしたときの
基準点の設定(±5.00div)

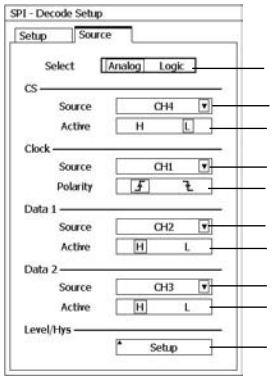
Source設定内のCSのソースがNoneのとき



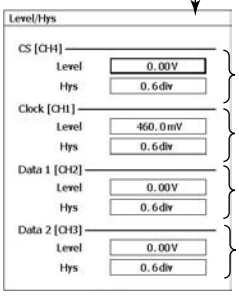
Clockのアイドル時間の設定
アイドル時間の設定を無視して、画面左端から
解析する場合にチェック

ソースの設定 (Source)

Analogのとき

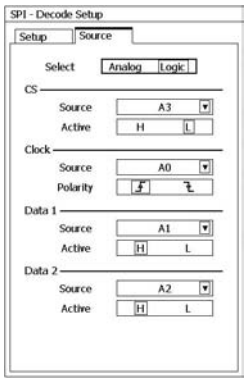


ソースの種類の設定*(Analogを選択)
CSのソースの設定(None、CH1～CH4、M1～M4)
有効にするレベル(Active)の設定(H、L)
Clockのソースの設定(CH1～CH4、M1～M4)
極性(Polarity)の設定
Data 1のソースの設定(None、CH1～CH4、M1～M4)
有効にするレベル(Active)の設定(H、L)
Data 2のソースの設定(None、CH1～CH4、M1～M4)
有効にするレベル(Active)の設定(H、L)
レベル、ヒステリシスの設定



CSのソースのレベル、ヒステリシスの設定
Clockのソースのレベル、ヒステリシスの設定
Data 1のソースのレベル、ヒステリシスの設定
Data 2のソースのレベル、ヒステリシスの設定

Logicのとき(DLM6000)



*：DL6000シリーズではAnalogに固定(設定無し)

解　説

線式 (Mode)

次の中から選択できます。

3 wire	1 つの Data ラインのデータを解析します。
4 wire	Data 1 ラインと Data 2 ラインの 2 つのデータを解析します。

ビットオーダ (Bit Order)

データの信号の流れに合わせて、ビットオーダを選択できます。

MSB	MSB からデータの信号が流れているとき
LSB	LSB からデータの信号が流れているとき

フィールドサイズ (Fiels Size)

フィールドサイズを設定できます。設定したビット単位でデコードします。
設定範囲は 4 ～ 32 ビットです。

有効ビットの範囲 (Enable NSB/LSB)

フィールドサイズの有効ビットの範囲を指定できます。左側の入力ボックスが MSB、右側の入力ボックスが LSB です。
有効ビット以外のデータは解析されません。

CS/Clock/ データ

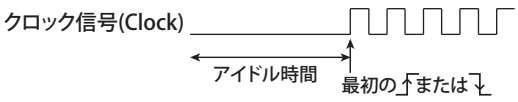
CS(チップセレクト)/ Clock(クロック)/ データのソースを選択できます。

- DL6000 シリーズの場合、None*1、CH1 ～ CH4 または M1 ～ M4 から選択します。
- DLM6000 シリーズの場合、None*1、CH1 ～ CH4、M1 ～ M4、A0 ～ A7、B0 ～ B7、C0 ～ C7、および D0 ～ D7(16 ビットモデルは None*1、CH1 ～ CH4、M1 ～ M4、A0 ～ A7 と C0 ～ C7) から選択します。

  *1 None は CS にだけ選択できます。CS に None を選択した場合、Idle Time からデータ開始点を決定します。

Clock のアイドル時間

CS の Source に None を選択した場合、設定したアイドル時間以上経過後の最初のクロックをデータ開始点とします。Don't care を選択した場合は、アイドル時間の設定値にかかわらず、画面左端から解析します。



CS

CS のレベルがどちらの状態のときに、データを解析するかを選択できます。

H	High レベルのとき
L	Low レベルのとき

Clock

Clock のどちらのエッジのタイミングで、データのステータスを判定するかを選択できます。

↑	立ち上がりのとき
↓	立ち下がりのとき

Data 1、Data 2

データのステータスのどちらを 1(Active)、0 にするかを選択できます。

H	設定したレベル以上のとき 1、レベル未満のとき 0
L	設定したレベル以下のとき 1、レベルを超えたとき 0

レベル (Level)

CH1 ～ CH4 と M1 ～ M4 に対して、判定レベル *2 を設定できます。
設定範囲は垂直ポジションを中心に± 10div 分で、設定分解能は 0.01div です。 たとえば、2V/div のときの設定分解能は 0.02V です。

*2 Logic(A0 ～ D7) のときは、ユーザーズマニュアル IM DLM6054-01JA の 5.2 節で設定したスレショルドレベルです。

ヒステリシス (Hys)

CH1 ～ CH4 と M1 ～ M4 に対して、設定します。
設定範囲は 0.0 ～ 4.0div で、設定分解能は 0.1div です。
トリガのヒステリシスとの関係は 2-7 ページをご覧ください。

解析基準点 (Reference Point)

解析基準点を次の中から選択します。

Trig Pos	トリガポジションを解析基準点に設定します。
Manual	手動で解析基準点を設定します。設定範囲：± 5.00div、設定分解能：0.01div

4.6 UART 信号を解析する

解　析

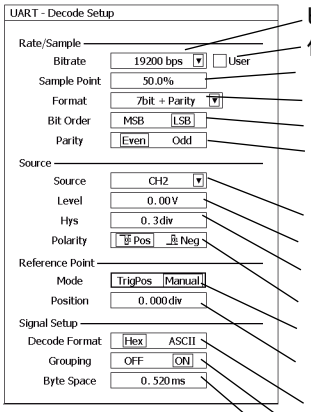
ANALYSIS メニュー

ANALYSIS キー > Mode のソフトキー > Serial Bus のソフトキー > UART のソフトキーを押します。以下のメニューが表示されます。

- Analysis：解析番号の設定(1、2)
- Mode：解析の種類(Mode)の設定 (UARTに設定)
- Decode Setup：デコードの設定
- List Setup：リスト操作(List Setup)の設定 ► 4.1節参照
- Display Setup：表示(Display Setup)の設定 ► 4.1節参照
- Zoom Link：リストとズーム画面のリンクの設定 ► 4.1節参照
- List V：リストのスクロール ► 4.1節参照

デコードの設定 (Decode Setup)

Decode Setup のソフトキーを押します。以下のメニューが表示されます。



- Bitrate：ビットレートの設定
- User：任意のビットレートを設定する場合にチェック
- Sample Point：サンプルポイントの設定
- Format：解析データのフォーマットの設定(8bit + Parity、7bit + Parity、(8bit(NonParity)
- Bit Order：ビットオーダの設定(MSB、LSB)
- Parity：パリティの設定(Even、Odd) (Formatを8bit + Parity、7bit + Parityにしたとき)
- Source：ソースの設定(CH1～CH4、M1～M4、A0～D7)
- Level：レベルの設定(SourceをCH1～CH4、M1～M4に設定したとき)
- Hys：ヒステリシスの設定(SourceをCH1～CH4、M1～M4に設定したとき)
- Polarity：極性の設定
- Mode：解析基準点の設定(TrigPos、Manual)
- Position：解析基準点をManualにしたときの基準点の設定(±5.00div)
- Decode Format：解析結果の表示方法の設定(Hex、ASCII)
- Grouping：グルーピングのON/OFF
- Byte Space：グルーピングする時間の設定 (グルーピングをONにしたとき)

解　説

ビットレート (Bitrate)

UART バス信号の転送レートを次の中から選択できます。
115200bps、57600bps、38400bps、19200bps、9600bps、4800bps、2400bps、1200bps
User 設定を選択した場合は、1000bps ～ 200000bps の範囲 (設定分解能 100bps) で設定できます。

サンプルポイント (Sample Point)

信号レベルを判定するポイントを 18.8 ～ 90.6[%] の範囲 (設定分解能 3.1%) で設定できます。
本機器の UART 信号のトリガ回路では、入力された UART 信号を内部クロックでサンプリングして、レベルの変化点を検出しています。検出された変化点を 0% とし、変化点からビットタイム (設定したビットレートの逆数) が経過したところを 100% として、サンプルポイントを % で設定します。
3-19 ページの説明図をご覧ください。

フォーマット (Format)

フォーマットを次の中から選択できます。

8bit + Parity	8 ビットのデータ＋ Parity Bit
7bit + Parity	7 ビットのデータ＋ Parity Bit
8bit(NonParity)	8 ビットのデータ (Parity Bit 無し)

ビットオーダ (Bit Order)

入力信号のデータパターンを、どちら側から読み込むかを選択します。

MSB	MSB 側からデータパターンを読み込みます。
LSB	LSB 側からデータパターンを読み込みます。

パリティ (Parity)

format を 8bit+Parity または 7bit+Parity に設定したときに設定します。
Parity Bit を Even(偶数) または Odd(奇数) に設定します

ソース (Source)

ソースを選択できます。

- DL6000 シリーズの場合、CH1 ～ CH4 または M1 ～ M4 から選択します。
- DLM6000 シリーズの場合、CH1 ～ CH4、M1 ～ M4、A0 ～ A7、B0 ～ B7、C0 ～ C7、および D0 ～ D7 16 ビットモデルは A0 ～ A7 と C0 ～ C7) から選択します。

レベル (Level)

信号レベルが 0 か 1 かを判定するレベル * を設定します。
設定範囲は垂直ポジションを中心に± 10div 分で、設定分解能は 0.01div です。 たとえば、2V/div のときの設定分解能は 0.02V です。

* Logic(A0 ～ D7) のときは、ユーザーズマニュアル IM DLM6054-01JA の 5.2 節で設定したスレショルドレベルです。

ヒステリシス (Hys)

設定範囲は 0.0 ～ 4.0div で、設定分解能は 0.1div です。
トリガのヒステリシスとの関係は 2-7 ページをご覧ください。

極性 (Polarity)

ビットのどちらの状態を論理 1 として認識するかを選択できます。

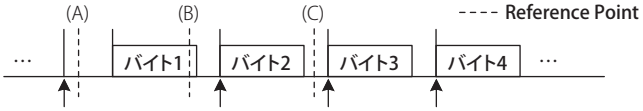
Pos	正論理
Neg	負論理

解析基準点 (Reference Point)

解析基準点を次の中から選択します。

Trig Pos	トリガポジションを解析基準点に設定します。
Manual	手動で解析基準点を設定します。設定範囲：± 5.00div、設定分解能：0.01div

解析基準点(Reference Point)の位置によって、解析結果リストのNo. 0のバイトは下記のようになります。



(A)： No. 0のバイト→バイト1 (バイト2がNo. 1、バイト3がNo. 2、・・・)
(B)： No. 0のバイト→バイト1 (バイト2がNo. 1、バイト3がNo. 2、・・・)
(C)： No. 0のバイト→バイト2 (バイト1がNo. -1、バイト3がNo. 1、バイト4がNo. 2、・・・)

解析結果の表示モード (Signal Setup)

解析結果の表示方法を設定します。

デコードの表示方法 (Decord Format)

Data の表示フォーマットを、16 進数 (Hex) または ASCII から選択できます。

- ASCII の場合、LF のようなコントロールコードの場合は、コード名 (LF) を表示します。
- Data が 16 進数の 7F 以上の場合、ASCII を選択していても 16 進数で表示します。

グルーピング (Grouping、Byte Space)

設定したバイトスペース (Byte Space) の時間未満の Data を、1 つのまとまったグループデータとして表示する (ON)/ しない (OFF) の選択ができます。

バイトスペース (Byte Space) の時間

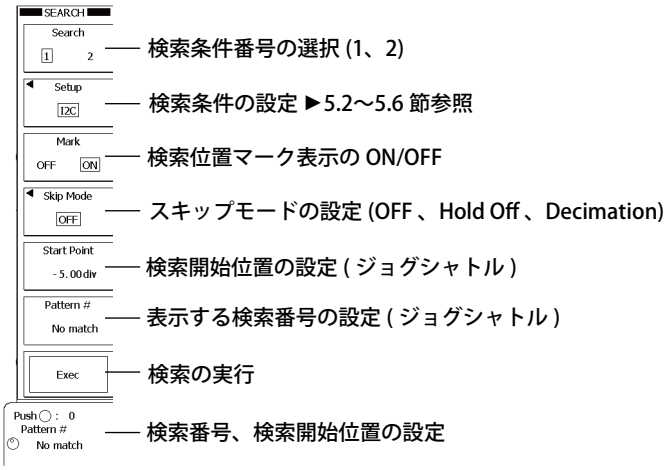
設定範囲	(UART 信号のデータフォーマットのビット数＋ 2 ビット) の時間～ 100ms 上記の「2 ビット」は、Start Bit と Stop Bit のビット数です。 たとえば データフォーマット が 8bit + Parity の場合、次の時間になります。 Data(8) ＋ Parity Bit(1) ＋ Start Bit(1) ＋ Stop Bit(1) ＝ 11 ビット分の時間
設定分解能	1bps の時間
初期値	(UART 信号のデータフォーマットのビット数＋ 2 ビット) の時間

5.1　シリアルバス信号 / スキップモードを選択する、検索を実行し結果を表示する

操　作

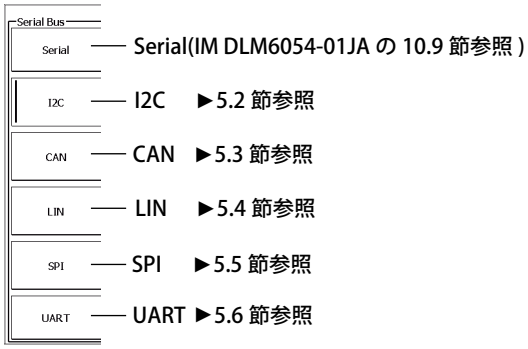
SEARCH メニュー

SEARCH キーを押します。次のメニューが表示されます。



シリアルバスの種類の設定

Setup のソフトキー > Type のソフトキー > Serial Bus のソフトキー を押します。次のメニューが表示されます。



解　説

検索タイプ (Type)

このマニュアルでは、I^2C、CAN、LIN、SPI、および UART のシリアルバスの信号を検索する機能について説明しています。所定の条件を満たす部分を検索し、拡大表示します。他の検索機能については、ユーザーズマニュアル IM DLM6054-01JA をご覧ください。

検索位置マーク (Mark)

検索点にマークを表示する / しないを選択します。ON にすると、Main ウインドウ上部に検索点マークが表示されます。

ON：　マークを表示する

OFF：　マークを表示しない

スキップモード (Skip Mode)

検索条件成立位置 (検索点) を検索後、設定した時間または設定した回数分ずつ、検索をスキップします。

OFF	検索点をすべて検索します。
Hold Off	設定した時間、検索をスキップします。 設定範囲：0.1ns ～ 1.00000s(有効数字 6 桁)、設定分解能：0.1ns
Decimation	設定した回数分、検索点をスキップします。 設定範囲：1 ～ 9999 回

検索開始点 (Start Point)

設定範囲は ± 5.00 div で、設定分解能は 0.01div です。

検索結果の表示 (Pattern #)

検索点に番号が付けられます。1 個目に「0」、2 個目に「1」・・・・というように順番に番号が付けられます。

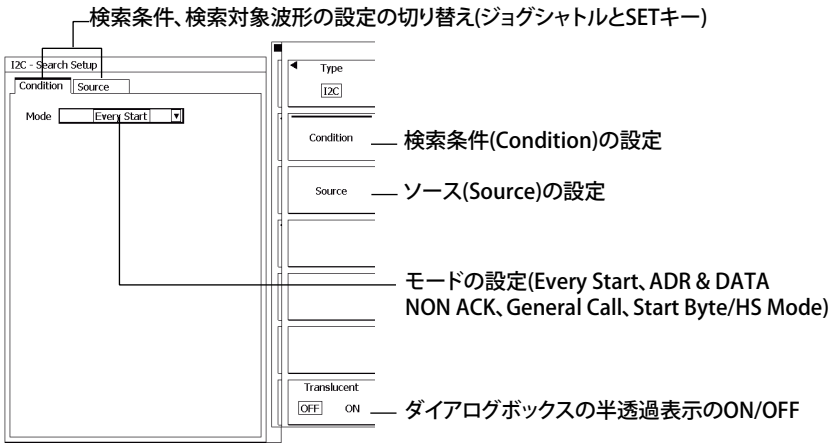
- 検索番号の最大値は 4999 です。
- 選択した検索番号の箇所の波形をズーム波形エリアに表示できます。

5.2 I²C バス信号を検索する

操　作

Setup メニュー

SEARCH キー > Setup のソフトキー > Type のソフトキー > Serial Bus のソフトキー > I2C のソフトキーを押します。次のメニューが表示されます。

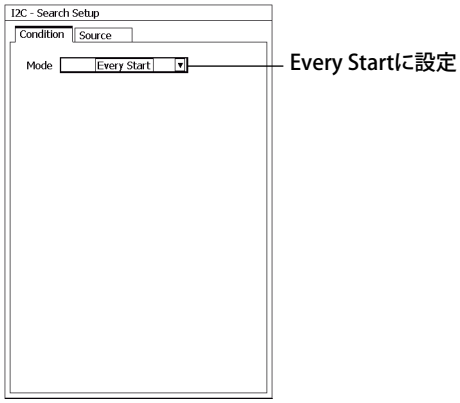


検索条件の設定 (Condition)

対象とするアドレスによって、5 つのモードがあります。

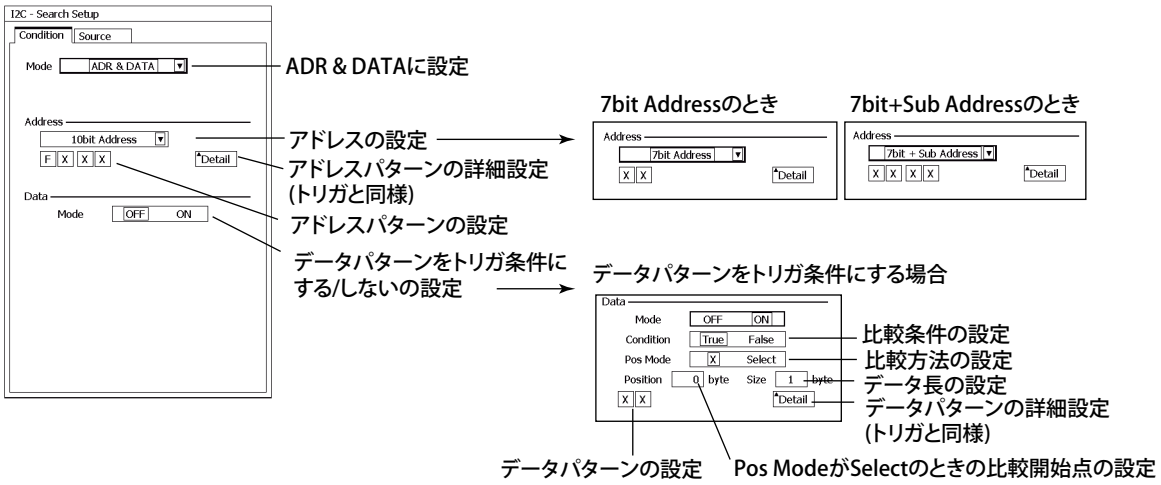
- Every Start
- ADR & DATA
- NON ACK
- General Call
- Start Byte/HS Mode

Every Start で検索する場合

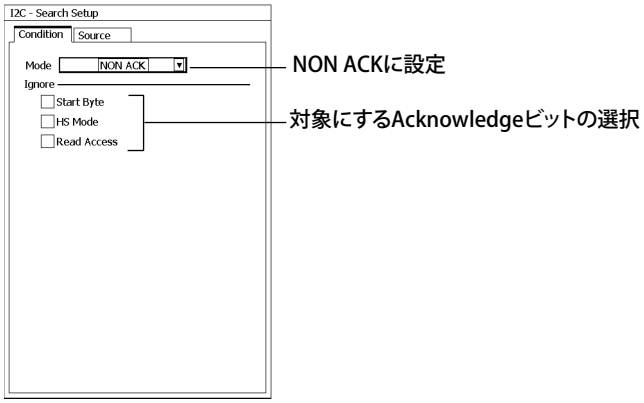


ADR & DATA で検索する場合

10bit Addressのとき

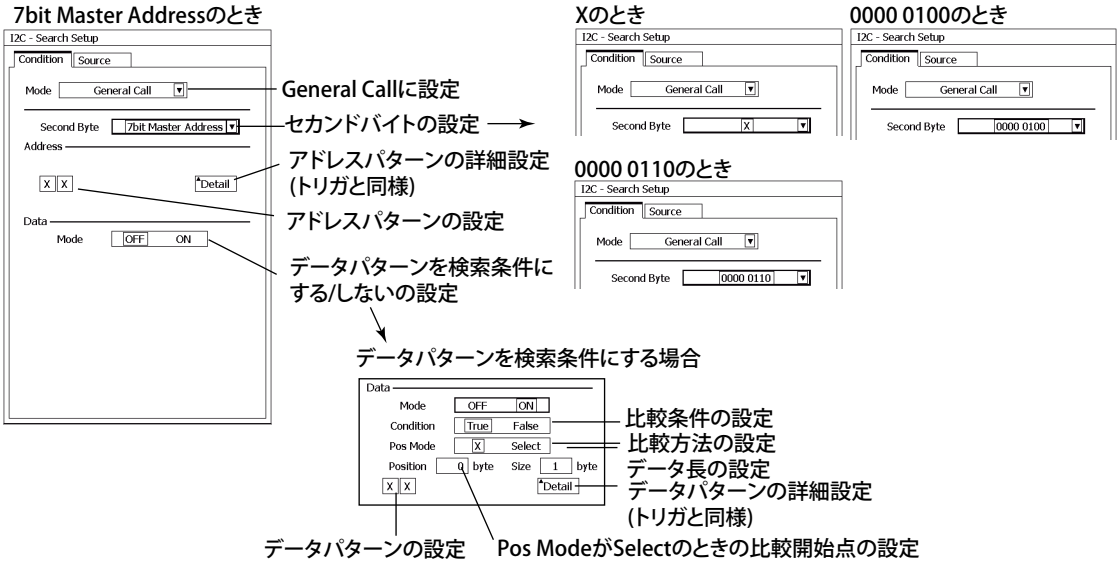


NON ACK で検索する場合

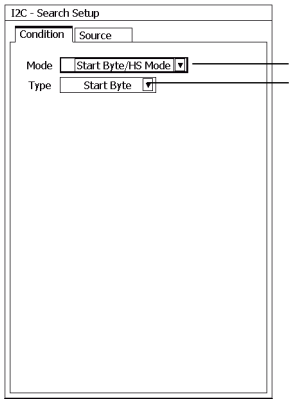


General Call で検索する場合

7bit Master Addressのとき



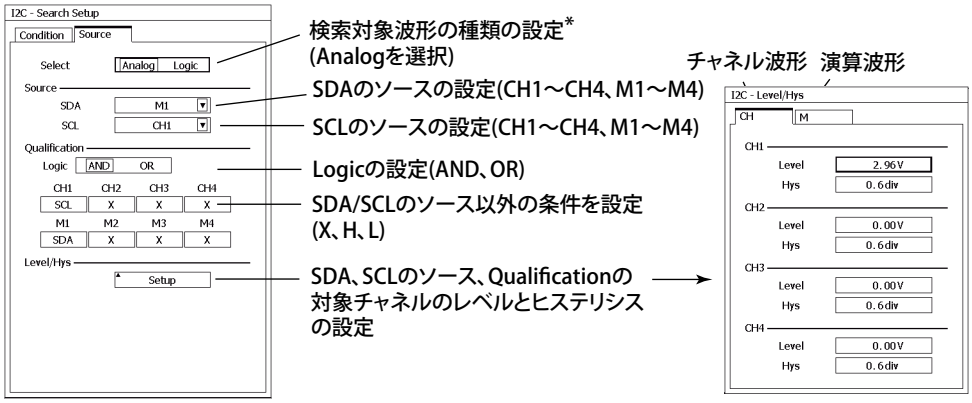
Start Byte/HS Mode で検索する場合



検索対象波形の設定 (Source)

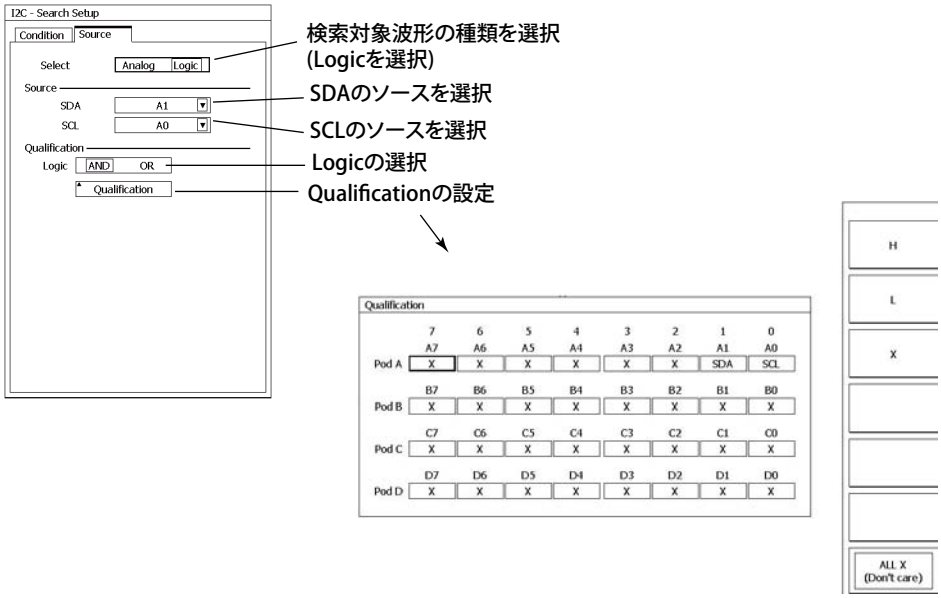
データパターンと比較する波形や比較条件を設定します。

アナログ入力のとき



*：DL6000シリーズではAnalogに固定(設定無し)

ロジック入力のとき (DLM6000)



検索を実行する

5.1 節に従って、操作してください。

解　説

I^2C バス信号を検索する機能です。I^2C バス信号のデータフォーマットについては、3.1 節の「解説」をご覧ください。

モード (Mode)

I^2C 検索の種類を、Every Start、ADR & DATA、NON ACK、General Call、および Start Byte/HS Mode の 5 つのモードから選択します。

Address

- アドレスのタイプを 7bit Address、7bit + Sub Address、または 10bit Address から選択できます。
- アドレスパターンを 16 進数または 2 進数で設定します。設定したアドレスパターンと 入力信号のアドレスパターンが一致したとき、Address の検索条件 が成立したことになります。
- パターンに X を設定すると、対応するビットの状態にかかわらず条件を満たしていると見なされます。
- 2 進のパターンに 1 つでも X があると、対応する 16 進の表示は「$」になります。

Data

データパターンを検索条件にする (ON)/ しない (OFF) の選択ができます。

- **比較条件 (Condition)**
  設定したパターンと入力信号のパターンを比較して、選択した比較条件を満たしたとき、Data の検索条件が成立したことになります。

True	パターンが一致したとき
False	パターンが一致しないとき

- **比較開始点 (Position)**
  Pos Mode の設定で、比較開始点を設定した点 (Select)/ 設定にかかわらない (X) から選択できます。
  Select を選択すると、設定した分のバイト数だけスキップした次のデータから比較します。
  設定範囲：0 ～ 9999 バイト

- **データ長 (Size)**
  連続したデータを、何バイト分比較するかを設定します。
  設定範囲：1 ～ 4 バイト

- **データパターン**
  Size で設定した長さのデータに対して、データパターンを 16 進数または 2 進数で設定します。
  - パターンに X を設定すると、対応するビットの状態にかかわらず条件を満たしていると見なされます。
  - 2 進のパターンに 1 つでも X があると、対応する 16 進の表示は「$」になります。

SDA/SCL/Qualification

SDA/SCL のソース

SDA(シリアルデータ)/ SCL(シリアルクロック) のソースを選択できます。

- DL6000 の場合、CH1 ～ CH4 または M1 ～ M4 から選択します。
- DLM6000 の場合、CH1 ～ CH4、M1 ～ M4、A0 ～ A7、B0 ～ B7、C0 ～ C7、および D0 ～ D7(16 ビットモデルは A0 ～ A7 と C0 ～ C7) から選択します。

- **レベル (Level)**

  CH1 ～ CH4 と M1 ～ M4 に対して、信号レベルが 0 か 1 かを判定するレベル * を設定します。
  設定範囲は垂直ポジションを中心に± 10div 分で、設定分解能は 0.01div です。 たとえば、2V/div のときの設定分解能は 0.02V です。

  * Logic(A0 ～ D7) のときは、ユーザーズマニュアル IM DLM6054-01JA の 5.2 節で設定したスレショルドレベルです。

- **ヒステリシス (Hys)**

  CH1 ～ CH4 と M1 ～ M4 に対して、設定します。
  設定範囲は 0.0 ～ 4.0div で、設定分解能は 0.1div です。
  トリガのヒステリシスとの関係は 2-7 ページをご覧ください。

Qualification と論理

- **必要条件**

  検索するための必要条件として、SDA と SCL に選択したソース以外の各信号の状態を H、L、または X から選択します。選択した状態と入力信号の状態が一致したとき、必要条件が成立したことになります。

H	High レベルのとき
L	Low レベルのとき
X	対象にしない (Don't care)

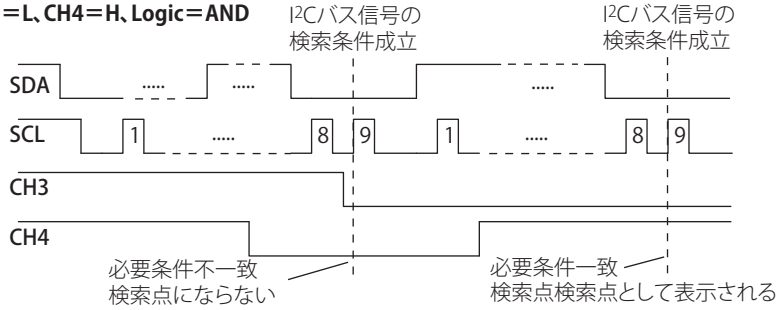
  * High か Low かの判定レベルは、信号が CH1 ～ CH4 と M1 ～ M4 のとき、上記で設定したレベルです。A0 ～ D7 のときは、ユーザーズマニュアル IM DLM6054-01JA の 5.2 節で設定したスレショルドレベルです。

- **論理条件**

  上項の必要条件と各モードで設定した I²C バス信号の検索条件の、論理条件を選択できます。論理条件を満たした点を検索します。

AND	必要条件 と I²C バス信号の検索条件が、両方成立したとき
OR	必要条件のどれかが成立したときに、I²C バス信号の検索条件が成立したとき

CH3＝L、CH4＝H、Logic＝AND



Note

I²C バス信号の検索条件 (SDA 信号 /SCL 信号) だけで、検索する場合は、次のように設定します。

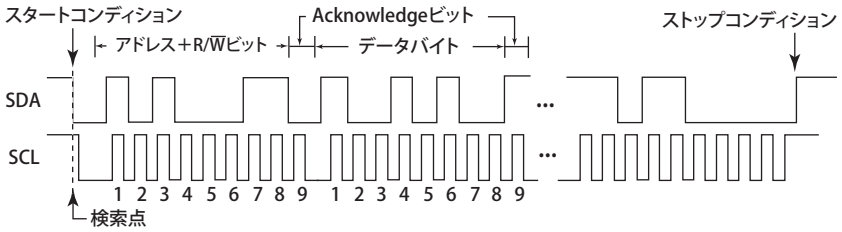
- SDA と SCL に選択したソース以外の各信号の状態：X(ソースにしない)
- 論理：AND

検索点

モードよって、次のようになります。

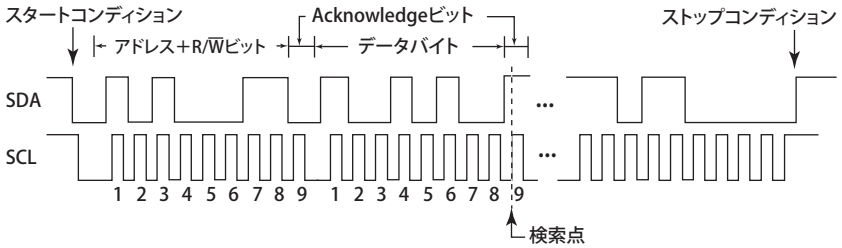
・Mode：Every Start

トリガ点と同様に、SDA信号の立ち下がりが検索点になります。



・Mode：Every Start以外

設定した条件に一致したあとのAcknowledgeビットの SCLの立ち上がりが検索点になります。
下記は、Mode：ADR & DATAの場合の例ですが、他のサーチモードの場合も同様です。

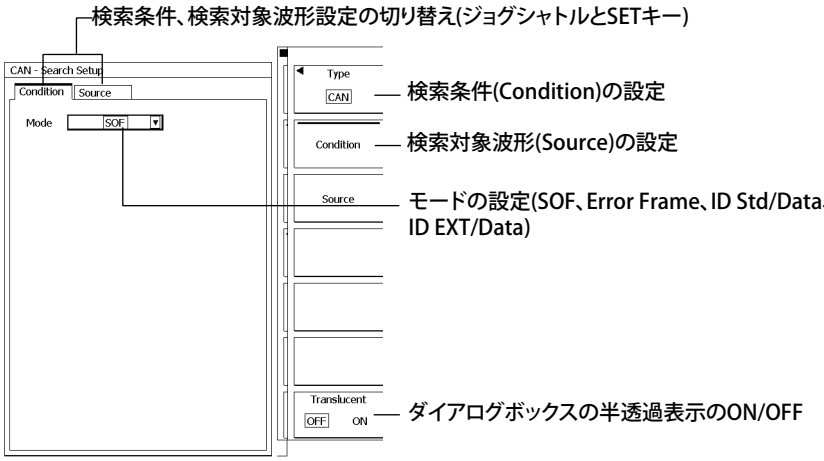


5.3 CAN バス信号を検索する

操　作

Setup メニュー

SEARCH キー > Setup のソフトキー > Type のソフトキー > Serial Bus のソフトキー > CAN のソフトキーを押します。次のメニューが表示されます。

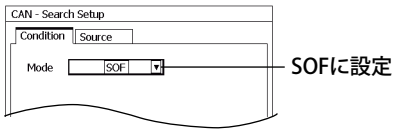


検索条件の設定 (Condition)

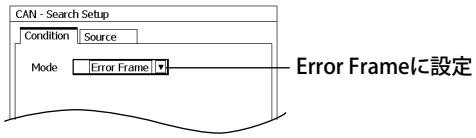
対象とするアドレスによって、5 つのモードがあります。

- SOF
- Error Frame
- ID Std/Data
- ID EXT/Data
- Msg/Signal

SOF で検索する場合

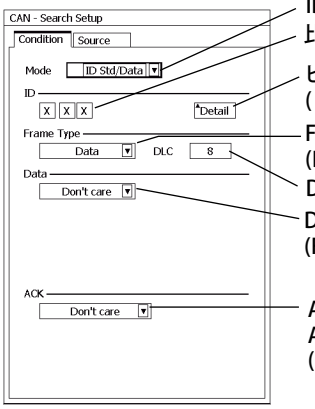


Error Frame で検索する場合



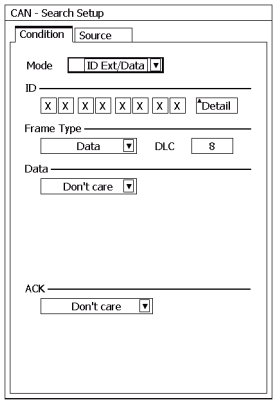
ID Std/Data または ID Ext/Data で検索する場合

ID Std/Data



ID Std/Datalに設定
比較するビットパターンの設定
ビットパターンの詳細設定
(トリガと同様)
Frame Typeの比較条件の設定
(Don't Care、Remote、Data)
DLCの設定(Frame TypeをDataに設定したとき)
Dataの比較条件の設定
(Frame TypeをDataに設定したとき)
ACKの状態の設定(Don't Care、NON ACK、ACK、NON ACK or ACK)
(Frame TypeをDataに設定したとき)

ID Ext/Data

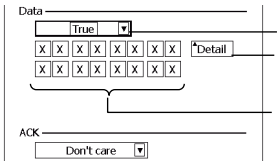


Frame Type の Data を検索条件にする場合

Data の比較条件によって設定内容が異なります。

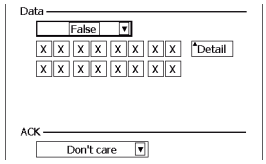
Data 条件が True または False のとき

True



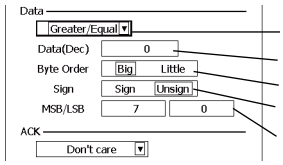
Trueに設定
データパターンの詳細設定
(トリガと同様)
データパターンの設定

False



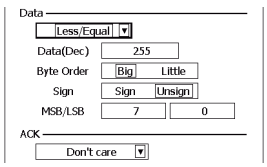
Data 条件が Greater/Equal または Less/Equal のとき

Greater/Equal



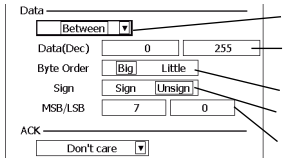
Greater/Equalに設定
判定値の設定
バイトオーダー(エンディアン)の設定
符号の設定
最上位、最下位のビット位置の設定(左側がMSB、左側がLSB)

Less/Equal



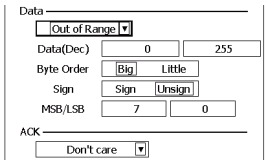
Data 条件が Between または Out of Range のとき

Between



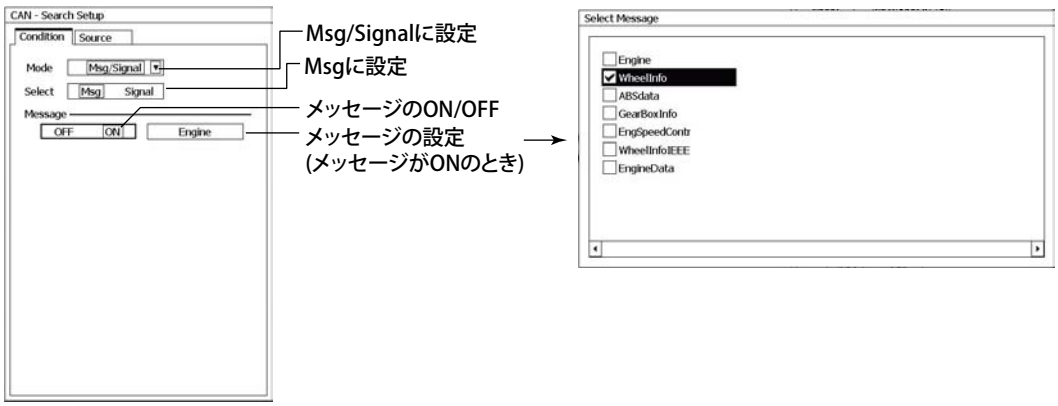
Betweenに設定
判定範囲の下限値(左側)、上限値(右側)の設定
バイトオーダー(エンディアン)の設定
符号の設定
最上位、最下位のビット位置の設定(左側がMSB、左側がLSB)

Out of Range

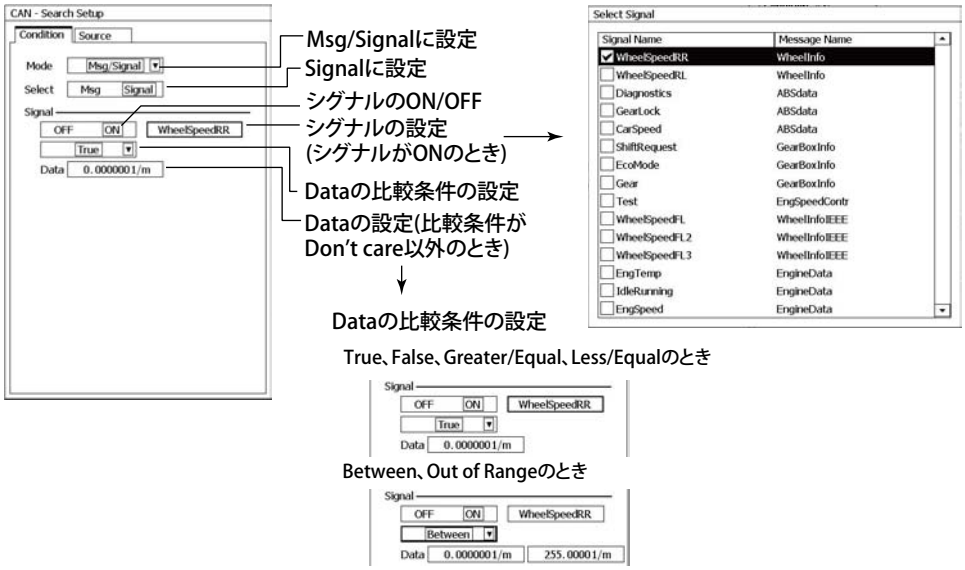


Msg/Signal で検索する場合

Message を検索条件にする場合



Signal を検索条件にする場合



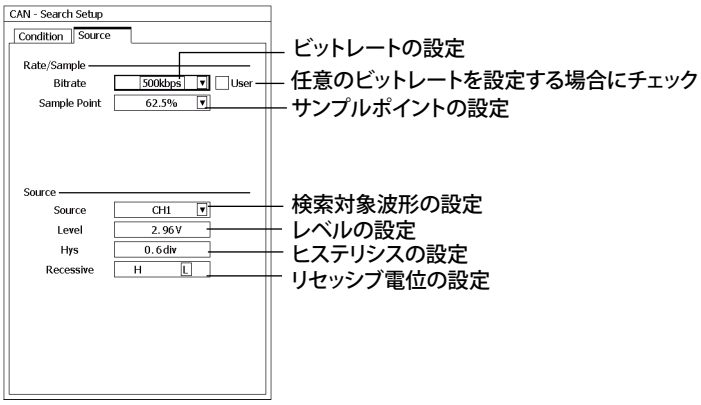
Dataの比較条件の設定

True、False、Greater/Equal、Less/Equalのとき

Between、Out of Rangeのとき

検索対象波形の設定 (Source)

Source タブを選択するか、Source のソフトキーを押します。次のメニューが表示されます。



検索を実行する

5.1 節に従って、操作してください。

解　説

CAN バス信号を検索する機能です。CAN バス信号のフレームフォーマットについては、3.2 節の「解説」をご覧ください。

モード

CAN 検索の種類を、SOF、Error Frame、ID Std/Data、および ID Ext/Data の 4 つのモードから選択します。

SOF モード

CAN バス信号のフレームの開始点を検索します。
SOF：Start of Frame

Error Frame モード

Error Frame の Error Flag がアクティブエラーフラグのとき、発生点を検索します。

ID Std/Data モード、ID Ext/Data モード

ID Std/Data モードは、標準フォーマットの場合の Data Frame や Remote Frame に対して、検索するモードです。
ID Ext/Data モードは、拡張フォーマットの場合の Data Frame や Remote Frame に対して、検索するモードです。
ID、Frame Type、Data、および ACK の AND 条件で検索します。
ID Std/Data モードの設定内容と、ID Ext/Data モードの設定内容は共通です。

- **ID**
  標準フォーマットの場合 は 11 ビットの、拡張フォーマットの場合は 29 ビットの ID の ビットパターンを 16 進数または 2 進数で設定します。設定したビットパターンと 入力信号の ID のビットパターンが一致したとき、ID の検索条件が成立したことになります。
  - パターンに X を設定すると、対応するビットの状態にかかわらず条件を満たしていると見なされます。
  - 2 進のパターンに 1 つでも X があると、対応する 16 進の表示は「$」になります。
- **Frame Type**
  Remote Frame と Data Frame を検索対象として設定できます。
  Frame の選択
  CAN バス信号フレームには、Remote Frame か Data Frame かを識別する RTR(Remote Transmission Request bit) があります。どちらのフレームを検索対象にするかを選択します。

Don't care	Remote Frame と Data Frame の両方が検索対象になります。
Remote	Remote Frame が検索対象になります。
Data Frame	Data Frame が検索対象になります。

  Don't care または Remote を選択したときは、次項の DLC や Data の検索条件は無視されます。
  DLC(Data Length Code)
  Data Field のデータ長を設定します。設定値と入力信号の DLC の値が一致したとき、DLC の検索条件 が成立したことになります。Data Frame を選択したときだけ設定します。
  設定範囲：0 ～ 8 バイト
  「0」のときは、次項の Data の検索条件は無視されます。

- **Data**

  Data Field の値を検索条件として設定できます。Data Frame を選択したときだけ設定します。

  比較条件

  判定値と入力信号の Data Field の値を比較して、選択した比較条件を満たしたとき、Data の検索条件が成立したことになります。

Don't care	対象にしない
True	判定値と一致したとき
False	判定値と一致しないとき
Greater/Equal	判定値以上のとき
Less/Equal	判定値以下のとき
Between	判定範囲内のとき (判定値を含む)
Out of Range	判定範囲外のとき (判定値を除く)

  データパターン

  DLC で設定した長さのデータに対して、データパターンを 16 進数または 2 進数で設定します。比較条件が True または False のときだけ有効です。

  - パターンに X を設定すると、対応するビットの状態にかかわらず条件を満たしていると見なされます。
  - 2 進のパターンに 1 つでも X があると、対応する 16 進の表示は「$」になります。

  判定値 (Data(Dec))

  - 比較条件が Greater/Equal または Less/Equal の場合は、判定値を 1 つ設定します。
  - 比較条件が Between または Out of Range の場合は、判定範囲を設定する必要があるため、判定値を 2 つ設定します。下限値≦上限値になるように自動的に調整されます。
  - 比較条件が True または False の場合は、データパターンを判定値として使用します。
  - 設定範囲

    10 進数で設定します。

符号が付かないとき (Unsign)	0 ～ 9E ＋ 18 ただし、設定可能な最大値は、DLC と MSB/LSB の設定で決まるデータ長とビット位置によって制限されます。
符号が付くとき (Sign)	－ 9E ＋ 18 ～ 9E ＋ 18 ただし、設定可能な最小値 / 最大値は、DLC と MSB/LSB の設定で決まるデータ長とビット位置によって制限されます。

    設定値は、7 桁を超えると指数で表示されます (例：1234567E ＋ 10)。

  バイトオーダー (Byte Order)

  Data のバイトオーダー (転送順序) をビッグエンディアン (Big)/ リトルエンディアン (Little) から選択します。具体例については、3-15 ページをご覧ください。

  符号：Sign

  Data に符号を付ける (Sign)/ 付けない (Unsign) を選択します。

  符号が付くときと付かないときで、Data の判定値の設定範囲が変わります。

  最上位 / 最下位のビットの位置：MSB/LSB

  比較する Data の最上位ビット (MSB)/ 最下位ビット (LSB) の位置を設定します。 具体例については、3-16 ページをご覧ください。

  設定範囲：0 ～ (データ長のバイト数× 8 － 1)、最大値は 63 です。

- **ACK**
  ACK スロットの状態を検索条件として設定できます。選択した状態と入力信号の ACK スロットの状態が一致したとき、ACK の検索条件が成立したことになります。

Don't care	対象にしない
NON ACK	リセッシブのとき
ACK	ドミナントのとき
NON ACK or ACK	リセッシブまたはドミナントのとき

Msg/Signal モード

本機器に読み込んだ物理値 / シンボル定義ファイル (.sbl)* に収納されている、Message や Signal のデータを検索条件にするモードです。整数型 (Integer) と浮動小数点型 (Float、32 ビットと 64 ビット) の両方のデータタイプに対応しています。Msg/Signal モードの検索条件の設定内容は、トリガ条件と同じです。

* 物理値 / シンボル定義ファイル (.sbl) は、CANdb ファイル (.dbc) を変換したものです。ファイルの読み込み操作については、別冊のユーザーズマニュアル (IM DLM6054-01JA) の 13.9 節をご覧ください。

ビットレート / サンプルポイント / レベル / ヒステリシス / リセッシブ電位

ビットレート (Bitrate)

CAN バス信号の転送レートを次の中から選択できます。
1Mbps、500kbps、250kbps、125kbps、83.3kbps、33.3kbps
User 設定を選択した場合は、10.0kbps ～ 1.000Mbps の範囲 (設定分解能 0.1kbps) で設定できます。

サンプルポイント (Sample Point)

バスレベル (リセッシブ / ドミナント) を判定するポイントを 18.8 ～ 90.6[%] の範囲 (設定分解能 3.1%) で設定できます。
本機器の CAN バス信号のトリガ回路では、入力された CAN バス信号を内部クロックでサンプリングして、リセッシブからドミナントへの変化点を検出しています。検出された変化点を 0% とし、変化点からビットタイム (設定したビットレートの逆数) が経過したところを 100% として、サンプルポイントを % で設定します。 3-17 ページの説明図をご覧ください。

ソース (Source)

CH1 ～ CH4 または M1 ～ M4 から選択できます。

- **レベル (Level)**
  信号レベルが 0 か 1 かを判定するレベルを設定します。
  設定範囲は垂直ポジションを中心に± 10div 分で、設定分解能は 0.01div です。 たとえば、2V/div のときの設定分解能は 0.02V です。

- **ヒステリシス (Hys)**
  設定範囲は 0.0 ～ 4.0div で、設定分解能は 0.1div です。
  トリガのヒステリシスとの関係は 2-7 ページをご覧ください。

- **リセッシブ電位 (Recessive)**
  リセッシブ電位を次の中から選択します。どちらの設定でも論理値は、リセッシブ＝ 1、ドミナント＝ 0 です。

H	リセッシブ電位がドミナント電位より高い
L	リセッシブ電位がドミナント電位より低い

検索点

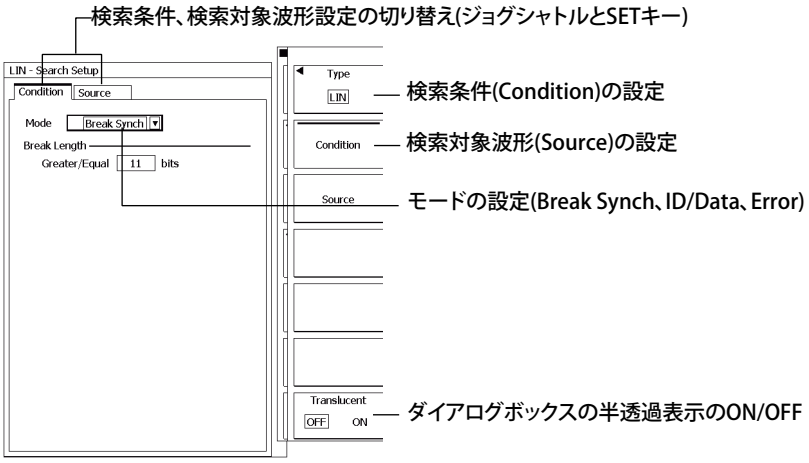
検索点の位置は、トリガ点と同じ位置です。トリガ点については、3.2 節の「解説」をご覧ください。

5.4 LIN バス信号を検索する

操　作

Setup メニュー

SEARCH キー > Setup のソフトキー > Type のソフトキー > Serial Bus のソフトキー > LIN のソフトキーを押します。次のメニューが表示されます。

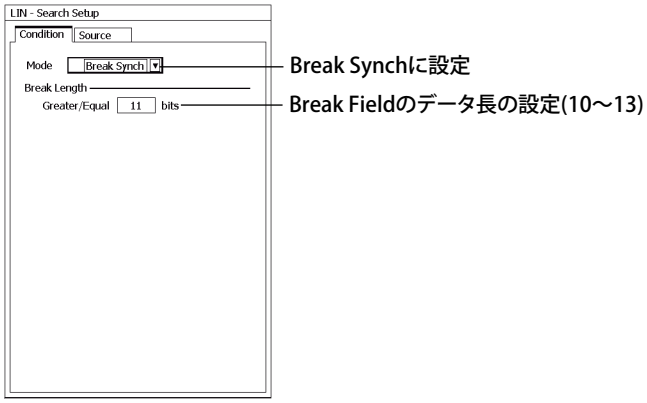


検索条件の設定 (Condition)

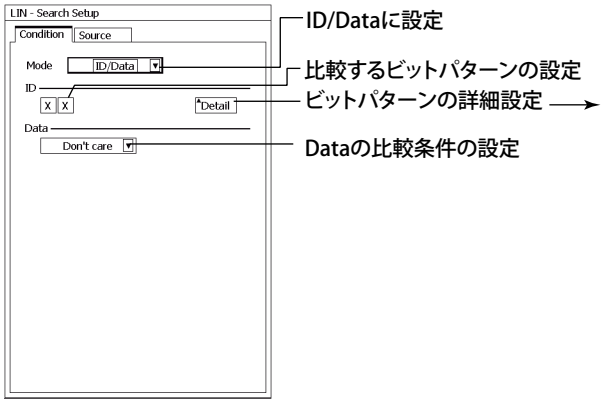
対象とするアドレスによって、3 つのモードがあります。

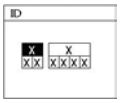
- Break Synch
- ID/Data
- Error

Break Synch で検索する場合



ID/Data で検索する場合

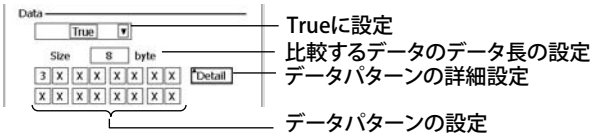




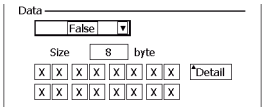
ビットパターンの詳細設定
Detailを選択して表示されるダイアログボックスで、ジョグシャトル、**SET**キー、ソフトキーを使ってビットパターンを設定できます。

Data の比較条件が True または False のとき

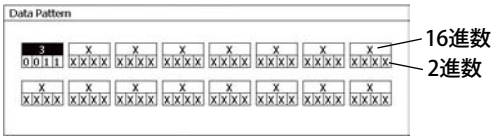
True



False

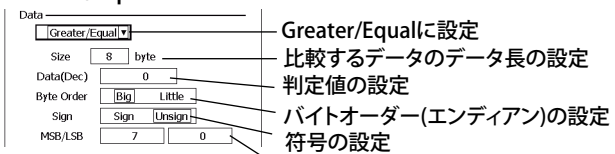


データパターンの詳細設定
Detailを選択して表示されるダイアログボックスで、ジョグシャトル、**SET**キー、ソフトキーを使ってデータパターンを設定できます。

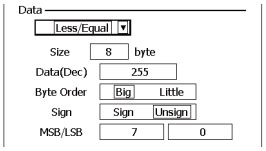


Data の比較条件が Greater/Equal または Less/Equal のとき

Greater/Equal

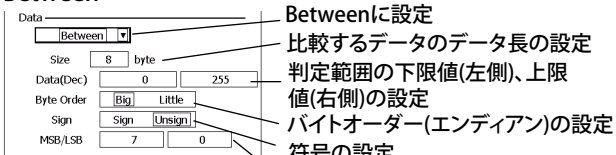


Less/Equal

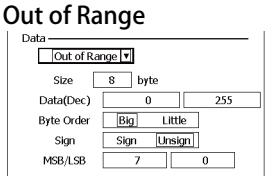


Data の比較条件が Between または Out of Range のとき

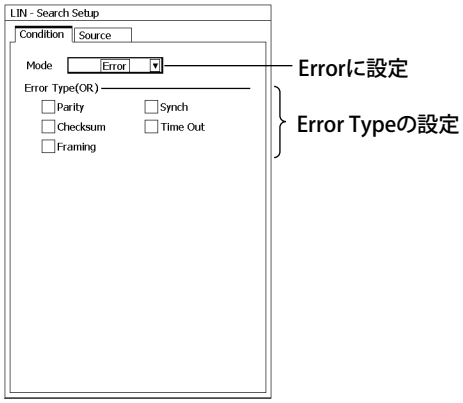
Between



Out of Range

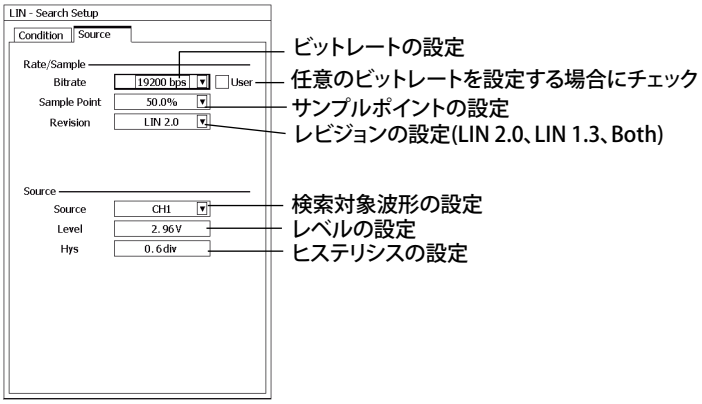


Error で検索する場合



検索対象波形の設定 (Source)

Source タブを選択するか、Source のソフトキーを押します。次のメニューが表示されます。



検索を実行する

5.1 節に従って、操作してください。

解　説

LIN バス信号を検索する機能です。

モード

LIN 検索の種類を、Break Synch、ID/Data、および Error の 3 つのモードから選択します。

Break Synch モード

Break Field に続いて Synch Field を検出 (Break Field + Synch Field) したとき、その点を検索します。
Break Field のデータ長を次の中から選択します。
10 以上、11 以上、12 以上、13 以上

ID/Data モード

ID と Data の AND 条件で検索します。

- **ID**

  Protected Identifier Field にある 6 ビットの保護 ID(ID0 ～ ID5) のビットパターンを 16 進数または 2 進数で設定します。設定したビットパターンと 入力信号の ID のビットパターンが一致したとき、ID の検索条件 が成立したことになります。
  - パターンに X を設定すると、対応するビットの状態にかかわらず条件を満たしていると見なされます。
  - 2 進のパターンに 1 つでも X があると、対応する 16 進の表示は「$」になります。

- **Data**

  Data 1 ～ Data 8 の値を検索条件として設定できます。
  比較条件
  判定値と入力信号の Data の値を比較して、選択した比較条件を満たしたとき、Data の検索条件 が成立したことになります。

Don't care	対象にしない
True	判定値と一致したとき
False	判定値と一致しないとき
Greater/Equal	判定値以上のとき
Less/Equal	判定値以下のとき
Between	判定範囲内のとき (判定値を含む)
Out of Range	判定範囲外のとき (判定値を除く)

  データ長 (Size)
  検索する Data の長さを設定します。
  設定範囲：1 ～ 8 バイト
  データパターン
  Size で設定した長さのデータに対して、データパターンを 16 進数または 2 進数で設定します。
  比較条件が True または False のときだけ有効です。
  - パターンに X を設定すると、対応するビットの状態にかかわらず条件を満たしていると見なされます。
  - 2 進のパターンに 1 つでも X があると、対応する 16 進の表示は「$」になります。

判定値 (Data(Dec)

- 比較条件が Greater/Equal または Less/Equal の場合は、判定値を 1 つ設定します。
- 比較条件が Between または Out of Range の場合は、判定範囲を設定する必要があるため、判定値を 2 つ設定します。下限値≦上限値になるように自動的に調整されます。
- 比較条件が True または False の場合は、データパターンを判定値として使用します。
- 設定範囲

  10 進数で設定します。

符号が付かないとき (Unsign)	0 ～ 9E ＋ 18 ただし、設定可能な最大値は、Size と MSB/LSB の設定で決まるデータ長とビット位置によって制限されます。
符号が付くとき (Sign)	－ 9E ＋ 18 ～ 9E ＋ 18 ただし、設定可能な最小値 / 最大値は、DataSize と MSB/LSB の設定で決まるデータ長とビット位置によって制限されます。

  設定値は、7 桁を超えると指数で表示されます (例：1234567E ＋ 10)。

バイトオーダー (Byte Order)

Data のバイトオーダー (転送順序) をビッグエンディアン (Big)/ リトルエンディアン (Little) から選択します。具体例については、3-17 ページをご覧ください。

符号：Sign

Data に符号を付ける (Sign)/ 付けない (Unsign) を選択します。

符号が付くときと付かないときで、Data の判定値の設定範囲が変わります。・

最上位 / 最下位のビットの位置：MSB/LSB

比較する Data の最上位ビット (MSB)/ 最下位ビット (LSB) の位置を設定します。 具体例については、3-16 ページをご覧ください。

設定範囲：0 ～ (データ長のバイト数× 8 － 1)、最大値は 63 です。

Error モード

エラーの発生点を検索します。

次の中から、検出するエラータイプを選択できます。

- 複数のエラータイプを選択できます。
- 選択したエラーすべてを検索します。

Parity	Protected Identifier Field の Parity 計算をして、次の式を満たさないとき、エラーになります。 ・Even Parity Check：ID0 xor ID1 xor ID2 xor ID4 xor P0 = 0 P0 = ID0 xor ID1 xor ID2 xor ID4 ・ODD Parity Check：ID1 xor ID3 xor ID4 xor ID5 xor P1 = 1 P1 = ¬(ID1 xor ID3 xor ID4 xor ID5)
Checksum	レビジョン LIN 2.0 の場合 (Enhanced Checksum) Protected Identifier Field、全 Data Field、および Checksum 値の合計値 *1 が 0xFF でないとき、エラーになります。ただし、Protected Identifier Field の ID が 0x60 ～ 0x63 の場合は、Classic Checksum の演算結果で判定します。 レビジョン LIN 1.3 の場合 (Classic Checksum) 全 Data Field と Checksum 値の演算結果が 0xFF でないとき、エラーになります。
Synch	Synch Field が 0x55 でないとき、エラーになります。0x55 のときでも、入力信号のビットレートが、設定したビットレート (次項参照) に対して－ 5.6% ～ 6.3% 以上ずれているとエラーになります。
Timeout	・Slave Not Responding Error Break 検出のあと、次の式の時間を経過しても Frame が終了していないとき、エラーになります。 $1.4 \times (T_{Header}{}^{*2} + T_{Response}{}^{*3})$ ・Header Timeout Error Break 検出のあと、次の式の時間を経過しても Header が終了していないとき、エラーになります。 $1.4 \times T_{Header}{}^{*2}$ ・Response Timeout Error Break 検出のあと、次の式の時間を経過しても Response が終了していないとき、エラーになります。 $1.4 \times T_{Response}{}^{*3}$
Framing	各 Field、Data、および Checksum の Stop Bit の状態が Low レベルであることを検出したとき、エラーになります。 Frame 途中での Break Field + Synch Field を検出したときも、エラーになります。

*1 255 を超えた場合、キャリーオーバーします。

*2 公称ヘッダ長 $T_{Header} = 34 \times T_{BIT}{}^{*4}$

*3 公称レスポンス長 $T_{Response} = 10 \times (N + 1) \times T_{BIT}{}^{*4}$　(N：データ長)

*4 物理層で定義されている 1 ビットを送信するために必要な公称時間

Note

バス上にデータ長が異なるフレームがある場合、エラータイプに Checksum、Timeout、または Framing を選択すると、正しい位置を検索できない場合があります。

ビットレート / サンプルポイント / レビジョン / レベル / ヒステリシス

ビットレート (Bitrate)

LIN バス信号の転送レートを次の中から選択できます。

19200bps、9600bps、4800bps、2400bps、1200bps

User 設定を選択した場合は、1000bps ～ 20000bps の範囲 (設定分解能 10bps) で設定できます。

サンプルポイント (Sample Point)

バスレベルを判定するポイントを 18.8 ～ 90.6[%] の範囲 (設定分解能 3.1%) で設定できます。

本機器の LIN バス信号のトリガ回路では、入力された LIN バス信号を内部クロックでサンプリングして、レベルの変化点を検出しています。検出された変化点を 0% とし、変化点からビットタイム (設定したビットレートの逆数) が経過したところを 100% として、サンプルポイントを % で設定します。 3-17 ページの説明図をご覧ください。

レビジョン (Revision)

レビジョンを次の中から選択できます。検索の種類が Error モードで、エラータイプに「Checksum」を選択しているとき、Enhanced Checksum または Classic Checksum のどちらのエラーを検知するかを選択します。

LIN 2.0	保護 ID を含む Enhanced Checksum のエラーを検知します。 (ただし、ID=60(0x3c) ～ 63(0x3f) は Classic Checksum のエラーを検知します。)
LIN 1.3	Data Field だけの Classic Checksum のエラーを検知します。
Both	LIN 2.0 と LIN 1.3 の両方の Checksum エラーを検知したとき、エラーとします。どちらかだけの場合は、エラーになりません。

ソース (Source)

ソースを選択できます。

- DL6000 シリーズの場合、None[*1]、CH1 ～ CH4 または M1 ～ M4 から選択します。
- DLM6000 シリーズの場合、None[*1]、CH1 ～ CH4、M1 ～ M4、A0 ～ A7、B0 ～ B7、C0 ～ C7、および D0 ～ D7(16 ビットモデルは None[*1]、A0 ～ A7 と C0 ～ C7) から選択します。

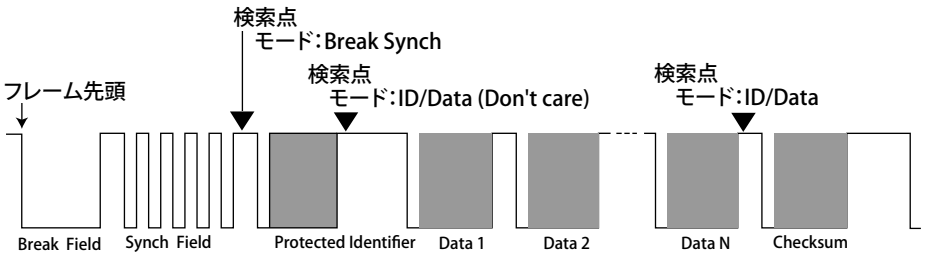
- **レベル (Level)**
  信号レベルが 0 か 1 かを判定するレベル * を設定します。
  設定範囲は垂直ポジションを中心に± 10div 分で、設定分解能は 0.01div です。 たとえば、2V/div のときの設定分解能は 0.02V です。
  * Logic(A0 ～ D7) のときは、ユーザーズマニュアル IM DLM6054-01JA の 5.2 節で設定したスレショルドレベルです。

- **ヒステリシス (Hys)**
  設定範囲は 0.0 ～ 4.0div で、設定分解能は 0.1div です。
  トリガのヒステリシスとの関係は 2-7 ページをご覧ください。

検索点

Break Synch/ID/Data モード

下記に例を示します。



Error モード

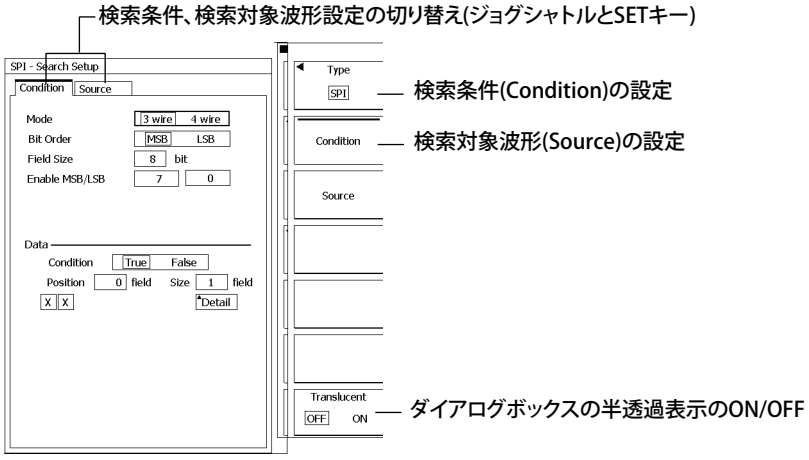
検索点の位置は、エラー発生点です。

5.5　SPI バス信号を検索する

操　作

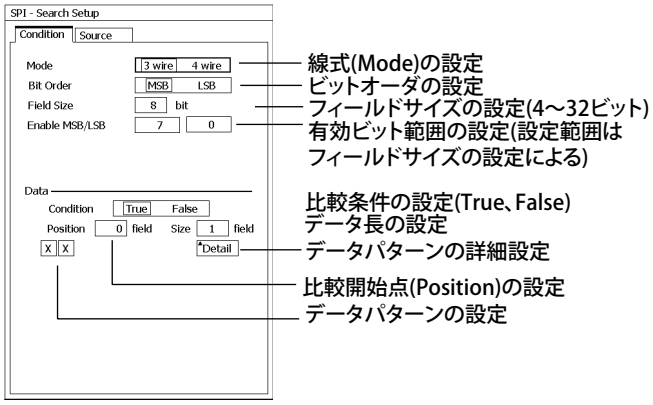
Setup メニュー

SEARCH キー > Setup のソフトキー > Type のソフトキー > Serial Bus のソフトキー > SPI のソフトキーを押します。次のメニューが表示されます。

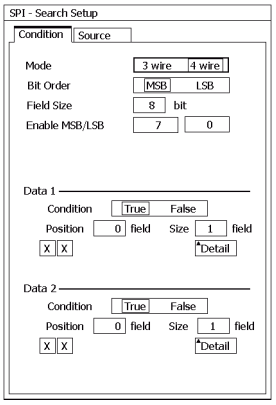


検索条件の設定 (Condition)

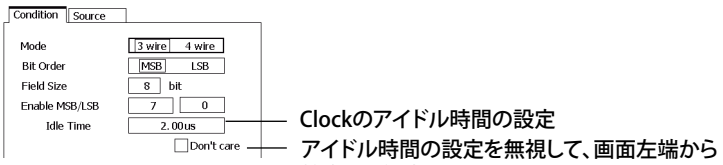
線式が3Wireのとき



線式が4Wireのとき



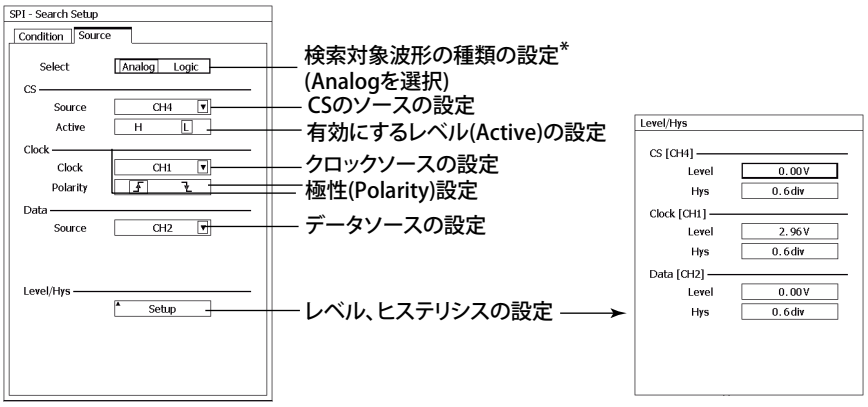
Source設定内のCSのソースがNoneのとき



検索対象波形の設定 (Source)

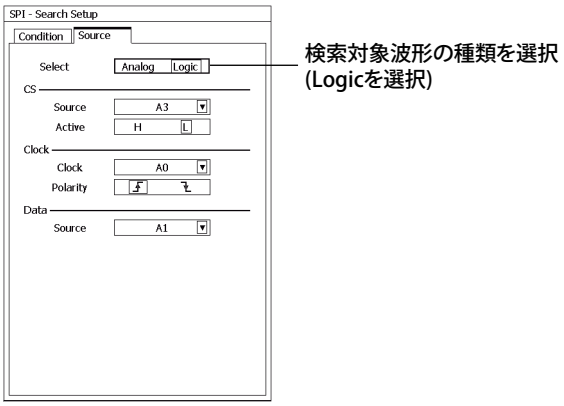
Source タブを選択するか、Source のソフトキーを押します。次のメニューが表示されます。

アナログ波形のとき



*：DL6000シリーズではAnalogに固定(設定無し)

ロジック波形のとき(DLM6000)



検索を実行する

5.1 節に従って、操作してください。

解　説

SPI バス信号を検索する機能です。SPI バス信号のタイムチャートについては、3.4 節の「解説」をご覧ください。

線式 (Mode)

次の中から選択できます。

3 wire	1 つの Data ラインのデータパターンを条件に検索します。
4 wire	Data 1 ラインと Data 2 ラインの 2 つのデータパターンを条件に検索します。Data 1 と Data 2 それぞれを単独で検索条件にすることもできます。

ビットオーダ (Bit Order)

データの信号の流れに合わせて、ビットオーダを選択できます。

- 2 進数でデータパターンを設定するときは、ビットオーダの設定に関係なく、流れるデータの順でパターンを設定します。
- 16 進数でデータパターンを設定するときは、ビットオーダの設定に従って、流れるデータの順に 4 ビットずつ区切ってパターンを設定します。

MSB	MSB からデータの信号が流れているとき
LSB	LSB からデータの信号が流れているとき

フィールドサイズ (Field Size)

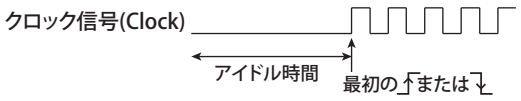
フィールドサイズを設定できます。
設定範囲は 4 ～ 32 ビットです。

有効ビットの範囲 (Enable MSB/LSB)

フィールドサイズの有効ビットの範囲を指定できます。左側の入力ボックスが MSB、右側の入力ボックスが LSB です。
有効ビット以外のデータは検索されません。

Clock のアイドル時間 (Idle Time)

CS の Source に None を選択した場合、設定したアイドル時間以上経過後の最初のクロックをデータ開始点とします。Don't care を選択した場合は、アイドル時間の設定値にかかわらず、画面左端から検索します。



データ (Data)

データパターンを検索条件として設定できます。

比較条件 (Condition)

設定したパターンと入力信号のパターンを比較して、選択した比較条件を満たしたとき、データの検索条件が成立したことになります。

True	パターンが一致したとき
False	パターンが一致しないとき

比較開始点 (Position)

比較開始点を設定します。たとえばデータ開始点の最初のデータから開始するときは、「0」を設定します。

設定範囲：0 ～ 9999 フィールド

データ長 (Size)

連続したデータを、何フィールド分比較するかを設定します。

設定範囲：1 ～ 4 フィールド

フィールドサイズを 32 ビットに設定し、データ長を 4 フィールドに設定すると、128 ビットのデータを比較できます。

データパターン

Size で設定した長さのデータに対して、データパターンを 16 進数または 2 進数で設定します。

- パターンに X を設定すると、対応するビットの状態にかかわらず条件を満たしていると見なされます。
- 2 進のパターンに 1 つでも X があると、対応する 16 進の表示は「$」になります。

CS/Clock/ データ

CS(チップセレクト)/ Clock(クロック)/ データのソースを選択できます。

- DL6000 シリーズの場合、None[*1]、CH1 ～ CH4 または M1 ～ M4 から選択します。
- DLM6000 シリーズの場合、None[*1]、CH1 ～ CH4、M1 ～ M4、A0 ～ A7、B0 ～ B7、C0 ～ C7、および D0 ～ D7(16 ビットモデルは None[*1]、A0 ～ A7 と C0 ～ C7) から選択します。

*1 None は CS にだけ選択できます。CS に None を選択した場合、Idle Time からデータ開始点を決定します。

CS

CS のレベルがどちらの状態のときに、データを有効 (Active) にするかを選択できます。

H	High レベルのとき
L	Low レベルのとき

Clock

Clock のどちらのエッジのタイミングで、データパターンを比較するかを選択できます。

↑	立ち上がりのとき
↓	立ち下がりのとき

レベル (Level)

CH1 ～ CH4 と M1 ～ M4 に対して、判定レベル *2 を設定できます。
設定範囲は垂直ポジションを中心に± 10div 分で、設定分解能は 0.01div です。 たとえば、2V/div のときの設定分解能は 0.02V です。
*2 Logic(A0 ～ D7) のときは、ユーザーズマニュアル IM DLM6054-01JA の 5.2 節で設定したスレショルドレベルです。

ヒステリシス (Hys)

CH1 ～ CH4 と M1 ～ M4 に対して、設定します。
設定範囲は 0.0 ～ 4.0div で、設定分解能は 0.1div です。
トリガのヒステリシスとの関係は 2-7 ページをご覧ください。

検索点

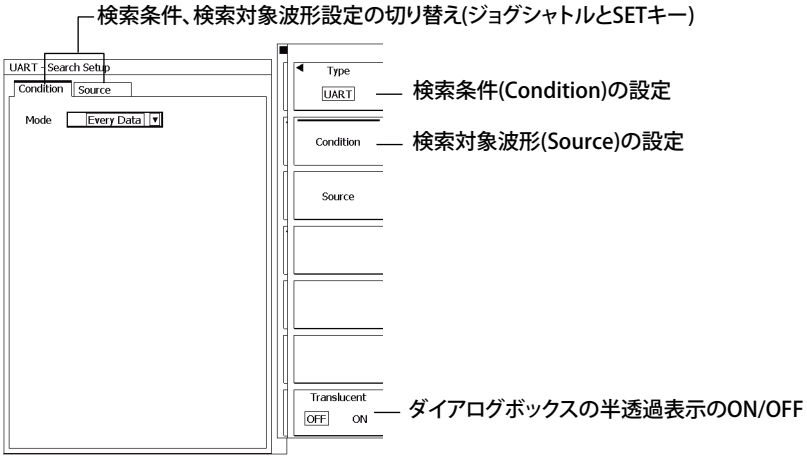
検索点の位置は、トリガ点と同じ位置です。トリガ点については、3.4 節の「解説」をご覧ください。

5.6 UART 信号を検索する

操　作

Setup メニュー

SEARCH キー > Setup のソフトキー > Type のソフトキー > Serial Bus のソフトキー > UART のソフトキーを押します。次のメニューが表示されます。

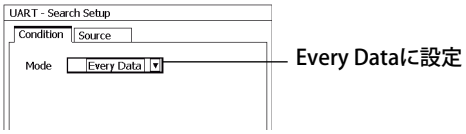


検索条件の設定 (Condition)

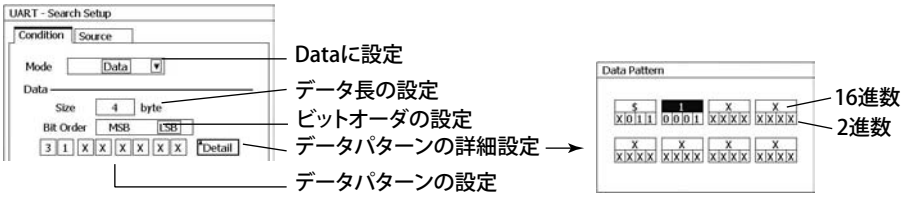
対象とするアドレスによって、3 つのモードがあります。

- Every Data
- Data
- Error

Every Data で検索する場合



Data で検索する場合

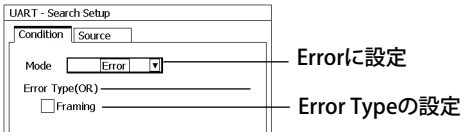


データパターンの詳細設定

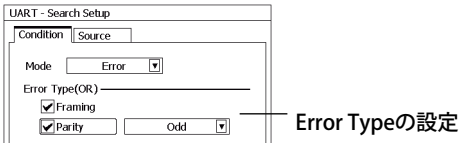
Detailを選択して表示されるダイアログボックスで、ジョグシャトル、SETキー、ソフトキーを使ってデータパターンを設定できます。

Error で検索する場合

Sourceの設定のFormatが
8bit(NonParity)のとき



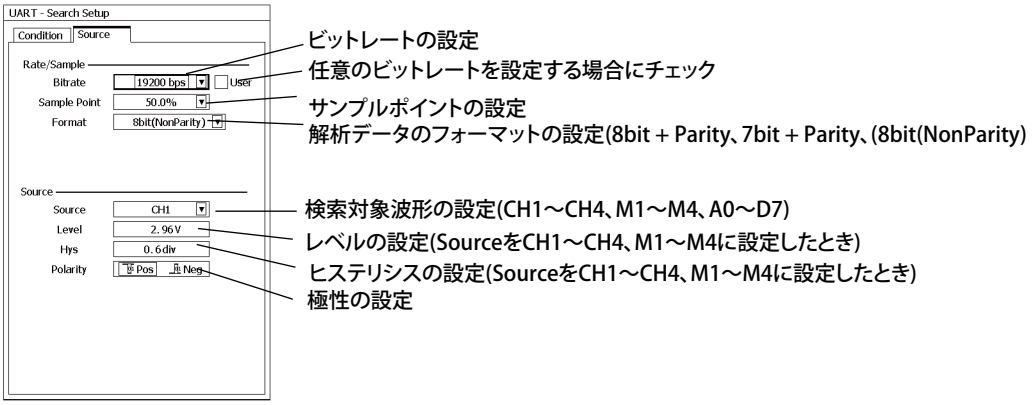
Sourceの設定のFormatが8bit +
Parityまたは7bit + Parityのとき



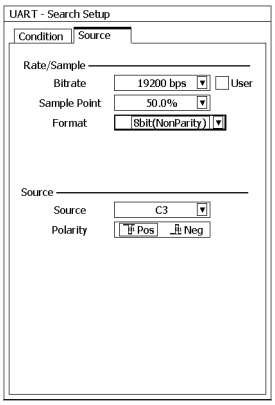
検索対象波形の設定 (Source)

Source タブを選択するか、Source のソフトキーを押します。次のメニューが表示されます。

アナログ波形のとき



ロジック波形のとき



検索を実行する

5.1 節に従って、操作してください。

解　説

UART 信号を検索する機能です。UART 信号のデータフォーマットについては、3.5 節の「解説」をご覧ください。

モード (Mode)

UART 検索の種類を、 Every Data、Data、および Error の 3 つのモードから選択します。

Every Data モード

すべてのデータの Stop Bit の位置を検索します。

Data モード

データパターンを検索します。

- データ長 (Size)
  連続した Data を、何バイト分比較するかを設定します。
  設定範囲：1 ～ 4 バイト
- ビットーダー (Bit Order)
  設定したデータパターンと入力信号のデータパターンを比較するときに、どちら側から読み込むかを選択します。

MSB	MSB 側からデータパターンを読み込みます。
LSB	LSB 側からデータパターンを読み込みます。

- データパターン
  Size で設定した長さのデータに対して、データパターンを 16 進数または 2 進数で設定します。
  - パターンに X を設定すると、対応するビットの状態にかかわらず条件を満たしていると見なされます。
  - 2 進のパターンに 1 つでも X があると、対応する 16 進の表示は「$」になります。

Error モード

エラーの発生点を検索します。
次の中から、検出するエラータイプを選択できます。

- 複数のエラータイプを選択できます。
- 選択したエラーすべてを検索します。

Framing	Stop Bit の論理値が「0」の位置を検索します。
Parity	受信したキャラクタの Parity エラーを検出したとき、Stop Bit の位置を検索します。 ・Odd、Even のどちらでチェックするかを選択できます。 ・Parity Bit 無しの設定にしているときは、エラーは発生しません。

ビットレート / サンプルポイント / フォーマット / レベル / ヒステリシス / 極性

ビットレート (Bitrate)

UART 信号の転送レートを次の中から選択できます。
115200bps、57600bps、38400bps、19200bps、9600bps、4800bps、2400bps、1200bps
User 設定を選択した場合は、1000bps ～ 200000bps の範囲 (設定分解能 100bps) で設定できます。

サンプルポイント (Sample Point)

信号レベルを判定するポイントを 18.8 ～ 90.6[%] の範囲 (設定分解能 3.1%) で設定できます。
本機器の UART 信号のトリガ回路では、入力された UART 信号を内部クロックでサンプリングして、レベルの変化点を検出しています。検出された変化点を 0% とし、変化点からビットタイム (設定したビットレートの逆数) が経過したところを 100% として、サンプルポイントを % で設定します。
3-17 ページの説明図をご覧ください。

フォーマット (Format)

フォーマットを次の中から選択できます。

8bit + Parity	8 ビットのデータ + Parity Bit
7bit + Parity	7 ビットのデータ + Parity Bit
8bit(NonParity)	8 ビットのデータ (Parity Bit 無し)

ソース (Source)

ソースを選択できます。

- DL6000 シリーズの場合、CH1 ～ CH4 または M1 ～ M4 から選択します。
- DLM6000 シリーズの場合、CH1 ～ CH4、M1 ～ M4、A0 ～ A7、B0 ～ B7、C0 ～ C7、および D0 ～ D7(16 ビットモデルは A0 ～ A7 と C0 ～ C7) から選択します。
- **レベル (Level)**
  信号レベルが 0 か 1 かを判定するレベル * を設定します。
  設定範囲は垂直ポジションを中心に± 10div 分で、設定分解能は 0.01div です。 たとえば、2V/div のときの設定分解能は 0.02V です。
  * Logic(A0 ～ D7) のときは、ユーザーズマニュアル IM DLM6054-01JA の 5.2 節で設定したスレショルドレベルです。
- **ヒステリシス (Hys)**
  設定範囲は 0.0 ～ 4.0div で、設定分解能は 0.1div です。
  トリガの Hysteresis との関係は 2-7 ページをご覧ください。
- **極性 (Polarity)**
  ビットのどちらの状態を論理 1 として認識するかを選択できます。

Pos	正論理
Neg	負論理

検索点

検索点の位置は、トリガ点と同じ位置です。3.5 節の「解説」をご覧ください。

6.1　メッセージ

ご使用中に、画面にメッセージが表示されることがあります。その意味と対処方法を説明します。ここでは、シリアルバス信号解析 / 検索機能に関するメッセージだけを記載しています。これら以外にも本体および通信関連のメッセージがあります。これらのメッセージについては、i ページに記載されている別冊のユーザーズマニュアルをご覧ください。

なお、メッセージ言語を選択できます。メッセージ言語の選択方法については、ユーザーズマニュアル IM DLM6054-01JA の 17.1 節をご覧ください。
サービスが必要なときは、お買い求め先まで修理をお申しつけください。

コード	日本語メッセージおよび対処方法	英語メッセージおよび対処方法	節
58	サーチを実行しましたが、条件と一致するパターンは見つかりませんでした。	Search execution is completed, but no record was found that matched the pattern.	5 章
69	シリアルバスの入力波形を認識することが出来ませんでした。	Any serial bus signal can not be detected.	2.1
70	シリアルバスの自動設定を中止しました。	Serial bus automatic setting was aborted.	2.1
73	入力電圧レベルと減衰比を確認してください。	Check the input voltage level and attenuation ratio.	2.1
506	セーブ対象となるデータがありません。 セーブデータの有無を確認してください。	Save data do not exist. Check the content to be saved.	4.1
670	該当するフィールドは見つかりませんでした。	The corresponding field was not found.	—
675	シリアルバス解析 / トリガの自動設定中です。 終了するまでお待ちください。	Serial bus automatic setting is in progress. Please wait.	—

7.1　コマンド一覧表

コマンド	機能	ページ
ANALysis グループ		
:ANALysis:LSBus<x>?	ロジックシリアルバス信号解析機能に関するすべての設定値を問い合わせます。	7-24
:ANALysis:LSBus<x>[:ANALyze]?	ロジックシリアルバス信号解析に関するすべての設定値を問い合わせます。	7-24
:ANALysis:LSBus<x>[:ANALyze]:I2CBus?	ロジック I^2C バス信号解析に関するすべての設定値を問い合わせます。	7-24
:ANALysis:LSBus<x>[:ANALyze]:I2CBus:CLOCk	ロジック I^2C バス信号解析のクロックチャネルを設定 / 問い合わせします。	7-24
:ANALysis:LSBus<x>[:ANALyze]:I2CBus:DTRace	ロジック I^2C バス信号解析のデータチャネルを設定 / 問い合わせします。	7-24
:ANALysis:LSBus<x>[:ANALyze]:LINBus?	ロジック LIN バス信号解析に関するすべての設定値を問い合わせます。	7-24
:ANALysis:LSBus<x>[:ANALyze]:LINBus:BRATe	ロジック LIN バス信号解析のビットレート (データ転送速度) を設定 / 問い合わせします。	7-24
:ANALysis:LSBus<x>[:ANALyze]:LINBus:FJUMp:BREak	ロジック LIN バス信号解析の結果を対象に Break Field へのフィールドジャンプを実行します。	7-25
:ANALysis:LSBus<x>[:ANALyze]:LINBus:FJUMp:CSUM	ロジック LIN バス信号解析の結果を対象に Checksum Field へのフィールドジャンプを実行します。	7-25
:ANALysis:LSBus<x>[:ANALyze]:LINBus:FJUMp:DATA	ロジック LIN バス信号解析の結果を対象に Data Field へのフィールドジャンプを実行します。	7-25
:ANALysis:LSBus<x>[:ANALyze]:LINBus:FJUMp:IDENtifier	ロジック LIN バス信号解析の結果を対象に Identifier Field へのフィールドジャンプを実行します。	7-25
:ANALysis:LSBus<x>[:ANALyze]:LINBus:FJUMp:SYNCh	ロジック LIN バス信号解析の結果を対象に Synch Field へのフィールドジャンプを実行します。	7-25
:ANALysis:LSBus<x>[:ANALyze]:LINBus:REVision	ロジック LIN バス信号解析のレビジョン (1.3 or 2.0 or Both) を設定 / 問い合わせします。	7-25
:ANALysis:LSBus<x>[:ANALyze]:LINBus:SPOint	ロジック LIN バス信号解析のサンプルポイントを設定 / 問い合わせします。	7-25
:ANALysis:LSBus<x>[:ANALyze]:LINBus:TRACe	ロジック LIN バス信号解析のトレースを設定 / 問い合わせします。	7-25
:ANALysis:LSBus<x>[:ANALyze]:LIST?	ロジックシリアルバス信号解析の解析結果リストに関するすべての設定値を問い合わせます。	7-26
:ANALysis:LSBus<x>[:ANALyze]:LIST:DISPlay	ロジックシリアルバス信号解析の解析結果リストの ON/OFF を設定 / 問い合わせします。	7-26
:ANALysis:LSBus<x>[:ANALyze]:LIST:ITEM?	ロジックシリアルバス信号解析の解析結果リストに表示される項目を問い合わせます。	7-26
:ANALysis:LSBus<x>[:ANALyze]:LIST:MODE	ロジックシリアルバス信号解析の解析結果リストのモードを設定 / 問い合わせします。	7-26
:ANALysis:LSBus<x>[:ANALyze]:LIST:SCRoll	ロジックシリアルバス信号解析の解析結果リストのスクロール方法を設定 / 問い合わせします。	7-26
:ANALysis:LSBus<x>[:ANALyze]:LIST:VALue?	ロジックシリアルバス信号解析の解析結果リストの指定した解析番号の自動測定値を問い合わせます。	7-26
:ANALysis:LSBus<x>[:ANALyze]:MODE	ロジックシリアルバス信号解析のモードを設定 / 問い合わせします。	7-26
:ANALysis:LSBus<x>[:ANALyze]:RPOint	ロジックシリアルバス信号解析のリファレンスポイント (解析基準点) を設定 / 問い合わせします。	7-27
:ANALysis:LSBus<x>[:ANALyze]:SPIBus?	ロジック SPI バス信号解析に関するすべての設定値を問い合わせます。	7-27
:ANALysis:LSBus<x>[:ANALyze]:SPIBus:CLOCk?	ロジック SPI バス信号解析のクロック信号のチャネルに関するすべての設定値を問い合わせます。	7-27
:ANALysis:LSBus<x>[:ANALyze]:SPIBus:CLOCk:POLarity	ロジック SPI バス信号解析のクロック信号のチャネルの極性を設定 / 問い合わせします。	7-27
:ANALysis:LSBus<x>[:ANALyze]:SPIBus:CLOCk:SOURce	ロジック SPI バス信号解析のクロック信号のチャネルを設定 / 問い合わせします。	7-27

コマンド	機能	ページ
:ANALysis:LSBus<x>[:ANALyze]:SPIBus:CS?	ロジック SPI バス信号解析のチップセレクト信号のチャネルに関するすべての設定値を問い合わせます。	7-27
:ANALysis:LSBus<x>[:ANALyze]:SPIBus:CS:ACTive	ロジック SPI バス信号解析のチップセレクト信号のチャネルのアクティブレベルを設定 / 問い合わせします。	7-28
:ANALysis:LSBus<x>[:ANALyze]:SPIBus:CS:TRACe	ロジック SPI バス信号解析のチップセレクト信号のチャネルを設定 / 問い合わせします。	7-28
:ANALysis:LSBus<x>[:ANALyze]:SPIBus:DATA<x>?	ロジック SPI バス信号解析の各データに関するすべての設定値を問い合わせます。	7-28
:ANALysis:LSBus<x>[:ANALyze]:SPIBus:DATA<x>:ACTive	ロジック SPI バス信号解析の各データのアクティブレベルを設定 / 問い合わせします。	7-28
:ANALysis:LSBus<x>[:ANALyze]:SPIBus:DATA<x>:TRACe	ロジック SPI バス信号解析の各データチャネルを設定 / 問い合わせします。	7-28
:ANALysis:LSBus<x>[:ANALyze]:SPIBus[:SETup]?	ロジック SPI バス信号解析のセットアップに関するすべての設定値を問い合わせます。	7-28
:ANALysis:LSBus<x>[:ANALyze]:SPIBus[:SETup]:BITorder	ロジック SPI バス信号解析のビットオーダを設定 / 問い合わせします。	7-29
:ANALysis:LSBus<x>[:ANALyze]:SPIBus[:SETup]:EMSBLSB	ロジック SPI バス信号解析のフィールドの有効範囲を設定 / 問い合わせします。	7-29
:ANALysis:LSBus<x>[:ANALyze]:SPIBus[:SETup]:FSIZe	ロジック SPI バス信号解析のフィールドサイズを設定 / 問い合わせします。	7-29
:ANALysis:LSBus<x>[:ANALyze]:SPIBus[:SETup]:ITIMe	ロジック SPI バス信号解析のアイドル時間を設定 / 問い合わせします。	7-29
:ANALysis:LSBus<x>[:ANALyze]:SPIBus[:SETup]:MODE	ロジック SPI バス信号解析の結線方式 (3 線式 /4 線式) を設定 / 問い合わせします。	7-29
ANALysis:LSBus<x>[:ANALyze]:UART?	ロジック UART 信号解析に関するすべての設定値を問い合わせます。	7-29
:ANALysis:LSBus<x>[:ANALyze]:UART:BITorder	ロジック UART 信号解析のビットオーダを設定 / 問い合わせします。	7-30
:ANALysis:LSBus<x>[:ANALyze]:UART:BRATe	ロジック UART 信号解析のビットレート (データ転送速度) を設定 / 問い合わせします。	7-30
:ANALysis:LSBus<x>[:ANALyze]:UART:BSPace	ロジック UART 信号解析のグルーピングのバイトスペースを設定 / 問い合わせします。	7-30
:ANALysis:LSBus<x>[:ANALyze]:UART:DFORmat	ロジック UART 信号解析のデコード文字表示形式を設定 / 問い合わせします。	7-30
:ANALysis:LSBus<x>[:ANALyze]:UART:FORMat	ロジック UART 信号解析のデータ形式を設定 / 問い合わせします。	7-30
:ANALysis:LSBus<x>[:ANALyze]:UART:GROuping	ロジック UART 信号解析のグルーピングの ON/OFF を設定 / 問い合わせします。	7-30
:ANALysis:LSBus<x>[:ANALyze]:UART:PMODe	ロジック UART 信号解析の Parity モードを設定 / 問い合わせします。	7-31
:ANALysis:LSBus<x>[:ANALyze]:UART:POLarity	ロジック UART 信号解析の極性を設定 / 問い合わせします。	7-31
:ANALysis:LSBus<x>[:ANALyze]:UART:SPOint	ロジック UART 信号解析のサンプルポイントを設定 / 問い合わせします。	7-31
:ANALysis:LSBus<x>[:ANALyze]:UART:TRACe	ロジック UART 信号解析のトレースを設定 / 問い合わせします。	7-31
:ANALysis:LSBus<x>:ZLINkage	ロジックシリアルバス信号解析のズームリンクを設定 / 問い合わせします。	7-31
:ANALysis:SBUS<x>?	シリアルバス信号解析機能に関するすべての設定値を問い合わせします。	7-31
:ANALysis:SBUS<x>:ANALyze?	シリアルバス信号解析に関するすべての設定値を問い合わせします。	7-31
:ANALysis:SBUS<x>[:ANALyze]:CANBus?	CAN バス信号解析に関するすべての設定値を問い合わせします。	7-31
:ANALysis:SBUS<x>[:ANALyze]:CANBus:BRATe	CAN バス信号解析のビットレート (データ転送速度) を設定 / 問い合わせします。	7-32
:ANALysis:SBUS<x>[:ANALyze]:CANBus:FJUMp:ACK	CAN バス信号解析の結果を対象に ACK Field へのフィールドジャンプを実行します。	7-32
:ANALysis:SBUS<x>[:ANALyze]:CANBus:FJUMp:CONTrol	CAN バス信号解析の結果を対象に Control Field へのフィールドジャンプを実行します。	7-32
:ANALysis:SBUS<x>[:ANALyze]:CANBus:FJUMp:CRC	CAN バス信号解析の結果を対象に CRC Field へのフィールドジャンプを実行します。	7-32
:ANALysis:SBUS<x>[:ANALyze]:CANBus:FJUMp:DATA	CAN バス信号解析の結果を対象に Data Field へのフィールドジャンプを実行します。	7-32

コマンド	機能	ページ
:ANALysis:SBUS<x>[:ANALyze]:CANBus:FJUMp:IDENtifier	CAN バス信号解析の結果を対象に Identifier Field へのフィールドジャンプを実行します。	7-32
:ANALysis:SBUS<x>[:ANALyze]:CANBus:FJUMp:SOF	CAN バス信号解析の結果を対象に SOF Field へのフィールドジャンプを実行します。	7-32
:ANALysis:SBUS<x>[:ANALyze]:CANBus:RECessive	CAN バス信号解析のリセッシブレベル (バスレベル) を設定 / 問い合わせします。	7-32
:ANALysis:SBUS<x>[:ANALyze]:CANBus:SPOint	CAN バス信号解析のサンプルポイントを設定 / 問い合わせします。	7-33
:ANALysis:SBUS<x>[:ANALyze]:CANBus:TRACe	CAN バス信号解析のトレースを設定 / 問い合わせします。	7-33
:ANALysis:SBUS<x>[:ANALyze]:DECode	シリアルバス信号解析のデコード表示の ON/OFF を設定 / 問い合わせします。	7-33
:ANALysis:SBUS<x>[:ANALyze]:I2CBus?	I^2C バス信号解析に関するすべての設定値を問い合わせします。	7-33
ANALysis:SBUS<x>[:ANALyze]:I2CBus:CLOCk	I^2C バス信号解析のクロックチャネルを設定 / 問い合わせします。	7-33
:ANALysis:SBUS<x>[:ANALyze]:I2CBus:DTRace	I^2C バス信号解析のデータチャネルを設定 / 問い合わせします。	7-33
:ANALysis:SBUS<x>[:ANALyze]:LINBus?	LIN バス信号解析に関するすべての設定値を問い合わせします。	7-33
:ANALysis:SBUS<x>[:ANALyze]:LINBus:BRATe	LIN バス信号解析のビットレート (データ転送速度) を設定 / 問い合わせします。	7-34
:ANALysis:SBUS<x>[:ANALyze]:LINBus:FJUMp:BREak	LIN バス信号解析の結果を対象に Break Field へのフィールドジャンプを実行します。	7-34
:ANALysis:SBUS<x>[:ANALyze]:LINBus:FJUMp:CSUM	LIN バス信号解析の結果を対象に Checksum Field へのフィールドジャンプを実行します。	7-34
:ANALysis:SBUS<x>[:ANALyze]:LINBus:FJUMp:DATA	LIN バス信号解析の結果を対象に Data Field へのフィールドジャンプを実行します。	7-34
:ANALysis:SBUS<x>[:ANALyze]:LINBus:FJUMp:IDENtifier	LIN バス信号解析の結果を対象に Identifier Field へのフィールドジャンプを実行します。	7-34
:ANALysis:SBUS<x>[:ANALyze]:LINBus:FJUMp:SYNCh	LIN バス信号解析の結果を対象に Synch Field へのフィールドジャンプを実行します。	7-34
:ANALysis:SBUS<x>[:ANALyze]:LINBus:REVision	LIN バス信号解析のレビジョン (1.3 or 2.0 or Both) を設定 / 問い合わせします。	7-34
:ANALysis:SBUS<x>[:ANALyze]:LINBus:SPOint	LIN バス信号解析のサンプルポイントを設定 / 問い合わせします。	7-34
:ANALysis:SBUS<x>[:ANALyze]:LINBus:TRACe	LIN バス信号解析のトレースを設定 / 問い合わせします。	7-35
:ANALysis:SBUS<x>[:ANALyze]:LIST?	シリアルバス信号解析の解析結果リストに関するすべての設定値を問い合わせします。	7-35
:ANALysis:SBUS<x>[:ANALyze]:LIST:DISPlay	シリアルバス信号解析の解析結果リストの ON/OFF を設定 / 問い合わせします。	7-35
:ANALysis:SBUS<x>[:ANALyze]:LIST:ITEM?	シリアルバス信号解析の解析結果リストに表示される項目を問い合わせします。	7-35
:ANALysis:SBUS<x>[:ANALyze]:LIST:MODE	シリアルバス信号解析の解析結果リストのモードを設定 / 問い合わせします。	7-35
:ANALysis:SBUS<x>[:ANALyze]:LIST:SCRoll	シリアルバス信号解析の解析結果リストのスクロール方法を設定 / 問い合わせします。	7-35
:ANALysis:SBUS<x>[:ANALyze]:LIST:VALue?	シリアルバス信号解析の解析結果リストの指定した解析番号の自動測定値を問い合わせします。	7-35
:ANALysis:SBUS<x>[:ANALyze]:MODE	シリアルバス信号解析のモードを設定 / 問い合わせします。	7-36
:ANALysis:SBUS<x>[:ANALyze]:RPOint	シリアルバス信号解析のリファレンスポイント (解析基準点) を設定 / 問い合わせします。	7-36
:ANALysis:SBUS<x>[:ANALyze]:SPIBus?	SPI バス信号解析に関するすべての設定値を問い合わせします。	7-36
:ANALysis:SBUS<x>[:ANALyze]:SPIBus:CLOCk?	SPI バス信号解析のクロック信号のチャネルに関するすべての設定値を問い合わせします。	7-36
:ANALysis:SBUS<x>[:ANALyze]:SPIBus:CLOCk:POLarity	SPI バス信号解析のクロック信号のチャネルの極性を設定 / 問い合わせします。	7-36
:ANALysis:SBUS<x>[:ANALyze]:SPIBus:CLOCk:SOURce	SPI バス信号解析のクロック信号のチャネルを設定 / 問い合わせします。	7-36
:ANALysis:SBUS<x>[:ANALyze]:SPIBus:CS?	SPI バス信号解析のチップセレクト信号のチャネルに関するすべての設定値を問い合わせします。	7-36
:ANALysis:SBUS<x>[:ANALyze]:SPIBus:CS:ACTive	SPI バス信号解析のチップセレクト信号のチャネルのアクティブレベルを設定 / 問い合わせします。	7-37
:ANALysis:SBUS<x>[:ANALyze]:SPIBus:CS:TRACe	SPI バス信号解析のチップセレクト信号のチャネルを設定 / 問い合わせします。	7-37
:ANALysis:SBUS<x>[:ANALyze]:SPIBus:DATA<x>?	SPI バス信号解析の各データに関するすべての設定値を問い合わせします。	7-37

コマンド	機能	ページ
:ANALysis:SBUS<x>[:ANALyze]:SPIBus:DATA<x>:ACTive	SPI バス信号解析の各データのアクティブレベルを設定 / 問い合わせします。	7-37
:ANALysis:SBUS<x>[:ANALyze]:SPIBus:DATA<x>:TRACe	SPI バス信号解析の各データチャネルを設定 / 問い合わせします。	7-37
:ANALysis:SBUS<x>[:ANALyze]:SPIBus:SETup?	SPI バス信号解析のセットアップに関するすべての設定値を問い合わせします。	7-37
:ANALysis:SBUS<x>[:ANALyze]:SPIBus[:SETup]:BITorder	SPI バス信号解析のビットオーダを設定 / 問い合わせします。	7-38
:ANALysis:SBUS<x>[:ANALyze]:SPIBus[:SETup]:EMSBLSB	SPI バス信号解析のフィールドの有効範囲を設定 / 問い合わせします。	7-38
:ANALysis:SBUS<x>[:ANALyze]:SPIBus[:SETup]:FSIZe	SPI バス信号解析のフィールドサイズを設定 / 問い合わせします。	7-38
:ANALysis:SBUS<x>[:ANALyze]:SPIBus[:SETup]:ITIMe	SPI バス信号解析のアイドル時間を設定 / 問い合わせします。	7-38
:ANALysis:SBUS<x>[:ANALyze]:SPIBus[:SETup]:MODE	SPI バス信号解析の結線方式 (3 線式 /4 線式) を設定 / 問い合わせします。	7-38
:ANALysis:SBUS<x>[:ANALyze]:TRACe<x>?:	シリアルバス信号解析の各ソースチャネルのしきい値 (Threshold) に関するすべての設定値を問い合わせします。	7-38
:ANALysis:SBUS<x>[:ANALyze]:TRACe<x>:HYSTeresis	シリアルバス信号解析の各ソースチャネルのしきい値 (Threshold) のヒステリシスを設定 / 問い合わせします。	7-39
:ANALysis:SBUS<x>[:ANALyze]:TRACe<x>:LEVel	シリアルバス信号解析の各ソースチャネルのしきい値 (Threshold) のレベルを設定 / 問い合わせします。	7-39
:ANALysis:SBUS<x>[:ANALyze]:UART?	UART 信号解析に関するすべての設定値を問い合わせします。	7-39
:ANALysis:SBUS<x>[:ANALyze]:UART:BITorder	UART 信号解析のビットオーダを設定 / 問い合わせします。	7-39
:ANALysis:SBUS<x>[:ANALyze]:UART:BRATe	UART 信号解析のビットレート (データ転送速度) を設定 / 問い合わせします。	7-39
:ANALysis:SBUS<x>[:ANALyze]:UART:BSPace	UART 信号解析のグルーピングのバイトスペースを設定 / 問い合わせします。	7-39
:ANALysis:SBUS<x>[:ANALyze]:UART:DFORmat	UART 信号解析のデコード文字表示形式を設定 / 問い合わせします。	7-40
:ANALysis:SBUS<x>[:ANALyze]:UART:FORMat	UART 信号解析のデータ形式を設定 / 問い合わせします。	7-40
:ANALysis:SBUS<x>[:ANALyze]:UART:GROuping	UART 信号解析のグルーピングの ON/OFF を設定 / 問い合わせします。	7-40
:ANALysis:SBUS<x>[:ANALyze]:UART:PMODe	UART 信号解析の Parity モードを設定 / 問い合わせします。	7-40
:ANALysis:SBUS<x>[:ANALyze]:UART:POLarity	UART 信号解析の極性を設定 / 問い合わせします。	7-40
:ANALysis:SBUS<x>[:ANALyze]:UART:SPOint	UART 信号解析のサンプルポイントを設定 / 問い合わせします。	7-40
:ANALysis:SBUS<x>[:ANALyze]:UART:TRACe	UART 信号解析のトレースを設定 / 問い合わせします。	7-41
:ANALysis:SBUS<x>:ZLINkage	シリアルバス信号解析のズームリンクを設定 / 問い合わせします。	7-41
:ANALysis:TYPE<x>	解析機能のタイプを設定 / 問い合わせします。	7-41

コマンド	機能	ページ
SEARch グループ		
SEARch<x>:CANBus?	CAN バス信号サーチに関するすべての設定値を問い合わせします。	7-42
:SEARch<x>:CANBus:SETup?	CAN バス信号サーチのセットアップに関するすべての設定値を問い合わせします。	7-42
:SEARch<x>:CANBus[:SETup]:ACK	CAN バス信号サーチの ACK 条件を設定 / 問い合わせします。	7-42
:SEARch<x>:CANBus[:SETup]:BRATe	CAN バス信号サーチのビットレート (データ転送速度) を設定 / 問い合わせします。	7-42
:SEARch<x>:CANBus[:SETup]:DATA?	CAN バス信号サーチのデータに関するすべての設定値を問い合わせします。	7-42
:SEARch<x>:CANBus[:SETup]:DATA:BORDer	CAN バス信号サーチのデータのバイトオーダを設定 / 問い合わせします。	7-42
:SEARch<x>:CANBus[:SETup]:DATA:CONDition	CAN バス信号サーチのデータ条件を設定 / 問い合わせします。	7-42
:SEARch<x>:CANBus[:SETup]:DATA:DATA<x>	CAN バス信号サーチのデータの比較データを設定 / 問い合わせします。	7-43
:SEARch<x>:CANBus[:SETup]:DATA:DLC	CAN バス信号サーチのデータの有効バイト数 (DLC) を設定 / 問い合わせします。	7-43
:SEARch<x>:CANBus[:SETup]:DATA:HEXA	CAN バス信号サーチのデータを HEXA で設定します。	7-43
:SEARch<x>:CANBus[:SETup]:DATA:MSBLsb	CAN バス信号サーチのデータの MSB/LSB のビットを設定 / 問い合わせします。	7-43
:SEARch<x>:CANBus[:SETup]:DATA:PATTern	CAN バス信号サーチのデータを BINARY で設定 / 問い合わせします。	7-43
:SEARch<x>:CANBus[:SETup]:DATA:SIGN	CAN バス信号サーチのデータの符号を設定 / 問い合わせします。	7-43
:SEARch<x>:CANBus[:SETup]:IDEXt?	CAN バス信号サーチの拡張フォーマットの ID に関するすべての設定値を問い合わせします。	7-44
:SEARch<x>:CANBus[:SETup]:IDEXt:HEXA	CAN バス信号サーチの拡張フォーマットの ID を HEXA で設定します。	7-44
:SEARch<x>:CANBus[:SETup]:IDEXt:PATTern	CAN バス信号サーチの拡張フォーマットの ID を BINARY で設定 / 問い合わせします。	7-44
:SEARch<x>:CANBus[:SETup]:IDSTd?	CAN バス信号サーチの標準フォーマットの ID に関するすべての設定値を問い合わせします。	7-44
:SEARch<x>:CANBus[:SETup]:IDSTd:HEXA	CAN バス信号サーチの標準フォーマットの ID を HEXA で設定します。	7-44
:SEARch<x>:CANBus[:SETup]:IDSTd:PATTern	CAN バス信号サーチの標準フォーマットの ID を BINARY で設定 / 問い合わせします。	7-44
:SEARch<x>:CANBus[:SETup]:MODE	CAN バス信号サーチのモードを設定 / 問い合わせします。	7-44
:SEARch<x>:CANBus[:SETup]:RECessive	CAN バス信号サーチのリセッシブレベル (バスレベル) を設定 / 問い合わせします。	7-44
:SEARch<x>:CANBus[:SETup]:RTR	CAN バス信号サーチの RTR を設定 / 問い合わせします。	7-44
:SEARch<x>:CANBus[:SETup]:SPOint	CAN バス信号サーチのサンプルポイントを設定 / 問い合わせします。	7-45
:SEARch<x>:CANBus[:SETup]:TRACe	CAN バス信号サーチのトレースを設定 / 問い合わせします。	7-45
SEARch<x>:I2CBus?	I^2C バス信号サーチに関するすべての設定値を問い合わせします。	7-45
:SEARch<x>:I2CBus:CLOCk?	I^2C バス信号サーチのクロックチャネルに関するすべての設定値を問い合わせします。	7-45
:SEARch<x>:I2CBus:CLOCk:SOURce	I^2C バス信号サーチのクロックチャネルを設定 / 問い合わせします。	7-45
:SEARch<x>:I2CBus:SETup?	I^2C バス信号サーチのセットアップに関するすべての設定値を問い合わせします。	7-45
:SEARch<x>:I2CBus[:SETup]:ADATa?	I^2C バス信号サーチのアドレスに関するすべての設定値を問い合わせします。	7-45
:SEARch<x>:I2CBus[:SETup]:ADATa:BIT10address?	I^2C バス信号サーチの 10bit アドレスに関するすべての設定値を問い合わせします。	7-45
:SEARch<x>:I2CBus[:SETup]:ADATa:BIT10address:HEXA	I^2C バス信号サーチの 10bit アドレスを HEXA で設定します。	7-45
:SEARch<x>:I2CBus[:SETup]:ADATa:BIT10address:PATTern	I^2C バス信号サーチの 10bit アドレスを BINARY で設定 / 問い合わせします。	7-45
:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7ADdress?	I^2C バス信号サーチの 7bit アドレスに関するすべての設定値を問い合わせします。	7-45

コマンド	機能	ページ
:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7ADdress:HEXA	I²C バス信号サーチの 7bit アドレスを HEXA で設定します。	7-46
:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7ADdress:PATTern	I²C バス信号サーチの 7bit アドレスを BINARY で設定 / 問い合わせします。	7-46
:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7APsub?	I²C バス信号サーチの 7bit+Sub アドレスに関するすべての設定値を問い合わせします。	7-46
:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7APsub:ADDRess?	I²C バス信号サーチの 7bit+Sub アドレスの 7bit アドレスに関するすべての設定値を問い合わせします。	7-46
:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7APsub:ADDRess:HEXA	I²C バス信号サーチの 7bit+Sub アドレスの 7bit アドレスを HEXA で設定します。	7-46
:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7APsub:ADDRess:PATTern	I²C バス信号サーチの 7bit+Sub アドレスの 7bit アドレスを BINARY で設定 / 問い合わせします。	7-46
:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7APsub:SADDress?	I²C バス信号サーチの 7bit+Sub アドレスの Sub アドレスに関するすべての設定値を問い合わせします。	7-46
:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7APsub:SADDress:HEXA	I²C バス信号サーチの 7bit+Sub アドレスの Sub アドレスを HEXA で設定します。	7-46
:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7APsub:SADDress:PATTern	I²C バス信号サーチの 7bit+Sub アドレスの Sub アドレスを BINARY で設定 / 問い合わせします。	7-47
:SEARch<x>:I2CBus[:SETup]:ADATa:TYPE	I²Cバス信号サーチのアドレスの種類を設定/問い合わせします。	7-47
:SEARch<x>:I2CBus[:SETup]:DATA?	I²C バス信号サーチのデータに関するすべての設定値を問い合わせします。	7-47
:SEARch<x>:I2CBus[:SETup]:DATA:BYTE	I²C バス信号サーチの設定データ数を設定 / 問い合わせします。	7-47
:SEARch<x>:I2CBus[:SETup]:DATA:CONDition	I²C バス信号サーチのデータの判定方法 (一致 / 不一致) を設定 / 問い合わせします。	7-47
:SEARch<x>:I2CBus[:SETup]:DATA:DPOSition	I²C バス信号サーチのデータのパターン比較する位置を設定 / 問い合わせします。	7-47
:SEARch<x>:I2CBus[:SETup]:DATA:HEXA<x>	I²C バス信号サーチのデータを HEXA で設定します。	7-47
:SEARch<x>:I2CBus[:SETup]:DATA:MODE	I²C バス信号サーチのデータ条件の有効 / 無効を設定 / 問い合わせします。	7-47
:SEARch<x>:I2CBus[:SETup]:DATA:PATTern<x>	I²C バス信号サーチのデータを BINARY で設定 / 問い合わせします。	7-48
:SEARch<x>:I2CBus[:SETup]:DATA:PMODe	I²C バス信号サーチのデータのパターン比較先頭位置モードを設定 / 問い合わせします。	7-48
:SEARch<x>:I2CBus[:SETup]:DATA:TRACe	I²C バス信号サーチのデータのトレースを設定 / 問い合わせします。	7-48
:SEARch<x>:I2CBus[:SETup]:GCALl?	I²C バス信号サーチのジェネラルコールに関するすべての設定値を問い合わせします。	7-48
:SEARch<x>:I2CBus[:SETup]:GCALl:BIT7maddress?	I²C バス信号サーチのジェネラルコールの 7bit マスタアドレスに関するすべての設定値を問い合わせします。	7-48
:SEARch<x>:I2CBus[:SETup]:GCALl:BIT7maddress:HEXA	I²C バス信号サーチのジェネラルコールの 7bit マスタアドレスを HEXA で設定します。	7-48
:SEARch<x>:I2CBus[:SETup]:GCALl:BIT7maddress:PATTern	I²C バス信号サーチのジェネラルコールの 7bit マスタアドレスを BINARY で設定 / 問い合わせします。	7-48
:SEARch<x>:I2CBus[:SETup]:GCALl:SBYTe (Second Byte)	I²C バス信号サーチのジェネラルコールのセカンドバイトのタイプを設定 / 問い合わせします。	7-49
:SEARch<x>:I2CBus[:SETup]:MODE	I²C バス信号サーチのサーチモードを設定 / 問い合わせします。	7-49
:SEARch<x>:I2CBus[:SETup]:NAIGnore?	I²C バス信号サーチの NON ACK 無視モードに関するすべての設定値を問い合わせします。	7-49
:SEARch<x>:I2CBus[:SETup]:NAIGnore:HSMode	I²C バス信号サーチのハイスピードモードで NON ACK を無視する / しないを設定 / 問い合わせします。	7-49
:SEARch<x>:I2CBus[:SETup]:NAIGnore:RACCess	I²C バス信号サーチのリードアクセスモードで NON ACK を無視する / しないを設定 / 問い合わせします。	7-49
:SEARch<x>:I2CBus[:SETup]:NAIGnore:SBYTe (Start Byte)	I²C バス信号サーチのスタートバイトで NON ACK を無視する / しないを設定 / 問い合わせします。	7-49
:SEARch<x>:I2CBus[:SETup]:SBHSmode?	I²C バス信号サーチのスタートバイト / ハイスピードモードに関するすべての設定値を問い合わせします。	7-49
:SEARch<x>:I2CBus[:SETup]:SBHSmode:TYPE	I²C バス信号サーチのスタートバイト / ハイスピードモードのタイプを設定 / 問い合わせします。	7-49
:SEARch<x>:LINBus?	LIN バス信号サーチに関するすべての設定値を問い合わせします。	7-50
:SEARch<x>:LINBus[:SETup]?	LIN バス信号サーチのセットアップに関するすべての設定値を問い合わせします。	7-50

コマンド	機能	ページ
:SEARch<x>:LINBus[:SETup]:BLENgth	LIN バス信号サーチの Break length を設定 / 問い合わせします。	7-50
:SEARch<x>:LINBus[:SETup]:BRATe	LIN バス信号サーチのビットレート (データ転送速度) を設定 / 問い合わせします。	7-50
:SEARch<x>:LINBus[:SETup]:DATA?	LIN バス信号サーチのデータに関するすべての設定値を問い合わせします。	7-50
:SEARch<x>:LINBus[:SETup]:DATA:BNUM	LIN バス信号サーチのデータのバイト数を設定 / 問い合わせします。	7-50
SEARch<x>:LINBus[:SETup]:DATA:BORDer	LIN バス信号サーチのデータのバイトオーダを設定 / 問い合わせします。	7-50
:SEARch<x>:LINBus[:SETup]:DATA:CONDition	LIN バス信号サーチのデータ条件を設定 / 問い合わせします。	7-50
:SEARch<x>:LINBus[:SETup]:DATA:DATA<x>	LIN バス信号サーチのデータの比較データを設定 / 問い合わせします。	7-51
:SEARch<x>:LINBus[:SETup]:DATA:HEXA	LIN バス信号サーチのデータを HEXA で設定します。	7-51
:SEARch<x>:LINBus[:SETup]:DATA:MSBLsb	LIN バス信号サーチのデータの MSB/LSB のビットを設定 / 問い合わせします。	7-51
:SEARch<x>:LINBus[:SETup]:DATA:PATTern	LIN バス信号サーチのデータを BINARY で設定 / 問い合わせします。	7-51
:SEARch<x>:LINBus[:SETup]:DATA:SIGN	LIN バス信号サーチのデータの符号を設定 / 問い合わせします。	7-51
:SEARch<x>:LINBus[:SETup]:ERRor?	LIN バス信号サーチの Error に関するすべてのを設定値を問い合わせします。	7-51
:SEARch<x>:LINBus[:SETup]:ERRor:CHECksum	LIN バス信号サーチの Checksum Error を設定 / 問い合わせします。	7-52
:SEARch<x>:LINBus[:SETup]:ERRor:FRAMing	LIN バス信号サーチの Framing Error を設定 / 問い合わせします。	7-52
:SEARch<x>:LINBus[:SETup]:ERRor:PARity	LIN バス信号サーチの Parity Error を設定 / 問い合わせします。	7-52
:SEARch<x>:LINBus[:SETup]:ERRor:SYNCh	LIN バス信号サーチの Synch Error を設定 / 問い合わせします。	7-52
:SEARch<x>:LINBus[:SETup]:ERRor:TOUT	LIN バス信号サーチの Timeout Error を設定 / 問い合わせします。	7-52
:SEARch<x>:LINBus[:SETup]:ID?	LIN バス信号サーチの ID に関するすべての設定値を問い合わせします。	7-52
:SEARch<x>:LINBus[:SETup]:ID:HEXA	LIN バス信号サーチの ID を HEXA で設定します。	7-52
:SEARch<x>:LINBus[:SETup]:ID:PATTern	LIN バス信号サーチの ID を BINARY で設定 / 問い合わせします。	7-52
:SEARch<x>:LINBus[:SETup]:MODE	LIN バス信号サーチのモードを設定 / 問い合わせします。	7-53
:SEARch<x>:LINBus[:SETup]:REVision	LIN バス信号サーチのレビジョン (1.3 or 2.0 or Both) を設定 / 問い合わせします。	7-53
:SEARch<x>:LINBus[:SETup]:SPOint	LIN バス信号サーチのサンプルポイントを設定 / 問い合わせします。	7-53
:SEARch<x>:LINBus[:SETup]:TRACe	LIN バス信号サーチのトレースを設定 / 問い合わせします。	7-53
:SEARch<x>:SLOGic:I2CBus?	ロジック I^2C バス信号サーチに関するすべての設定値を問い合わせます。	7-53
:SEARch<x>:SLOGic:I2CBus:CLOCk?	ロジック I^2C バス信号サーチのクロックチャネルに関するすべての設定値を問い合わせます。	7-53
:SEARch<x>:SLOGic:I2CBus:CLOCk:SOURce	ロジック I^2C バス信号サーチのクロックチャネルを設定 / 問い合わせします。	7-53
:SEARch<x>:SLOGic:I2CBus[:SETup]?	ロジック I^2C バス信号サーチのセットアップに関するすべての設定値を問い合わせます。	7-53
:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa?	ロジック I^2C バス信号サーチのアドレスに関するすべての設定値を問い合わせます。	7-53
:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:BIT10address?	ロジック I^2C バス信号サーチの 10bit アドレスに関するすべての設定値を問い合わせます。	7-53
:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:BIT10address:HEXA	ロジック I^2C バス信号サーチの 10bit アドレスを HEXA で設定します。	7-54
:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:BIT10address:PATTern	ロジック I^2C バス信号サーチの 10bit アドレスを BINARY で設定 / 問い合わせします。	7-54
:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:BIT7ADdress?	ロジック I^2C バス信号サーチの 7bit アドレスに関するすべての設定値を問い合わせます。	7-54
:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:BIT7ADdress:HEXA	ロジック I^2C バス信号サーチの 7bit アドレスを HEXA で設定します。	7-54
:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:BIT7ADdress:PATTern	ロジック I^2C バス信号サーチの 7bit アドレスを BINARY で設定 / 問い合わせします。	7-54
:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:BIT7APsub?	ロジック I^2C バス信号サーチの 7bit+Sub アドレスに関するすべての設定値を問い合わせます。	7-54
:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:BIT7APsub:ADDRess?	ロジック I^2C バス信号サーチの 7bit+Sub アドレスの 7bit アドレスに関するすべての設定値を問い合わせます。	7-54

コマンド	機能	ページ
:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:BIT7APsub:ADDRess:HEXA	ロジック I²C バス信号サーチの 7bit+Sub アドレスの 7bit アドレスを HEXA で設定します。	7-55
:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:BIT7APsub:ADDRess:PATTern	ロジック I²C バス信号サーチの 7bit+Sub アドレスの 7bit アドレスを BINARY で設定 / 問い合わせします。	7-55
:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:BIT7APsub:SADDress?	ロジック I²C バス信号サーチの 7bit+Sub アドレスの Sub アドレスに関するすべての設定値を問い合わせます。	7-55
:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:BIT7APsub:SADDress:HEXA	ロジック I²C バス信号サーチの 7bit+Sub アドレスの Sub アドレスを HEXA で設定します。	7-55
:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:BIT7APsub:SADDress:PATTern	ロジック I²C バス信号サーチの 7bit+Sub アドレスの Sub アドレスを BINARY で設定 / 問い合わせします。	7-55
:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:TYPE	ロジック I²C バス信号サーチのアドレスの種類を設定 / 問い合わせします。	7-55
:SEARch<x>:SLOGic:I2CBus[:SETup]:DATA?	ロジック I²C バス信号サーチのデータに関するすべての設定値を問い合わせます。	7-56
:SEARch<x>:SLOGic:I2CBus[:SETup]:DATA:BYTE	ロジック I²C バス信号サーチの設定データ数を設定 / 問い合わせします。	7-56
:SEARch<x>:SLOGic:I2CBus[:SETup]:DATA:CONDition	ロジック I²C バス信号サーチのデータの判定方法 (一致 / 不一致) を設定 / 問い合わせします。	7-56
:SEARch<x>:SLOGic:I2CBus[:SETup]:DATA:DPOSition	ロジック I²C バス信号サーチのデータのパターン比較する位置を設定 / 問い合わせします。	7-56
:SEARch<x>:SLOGic:I2CBus[:SETup]:DATA:HEXA<x>	ロジック I²C バス信号サーチのデータを HEXA で設定します。	7-56
:SEARch<x>:SLOGic:I2CBus[:SETup]:DATA:MODE	ロジック I²C バス信号サーチのデータ条件の有効 / 無効を設定 / 問い合わせします。	7-56
:SEARch<x>:SLOGic:I2CBus[:SETup]:DATA:PATTern<x>	ロジック I²C バス信号サーチのデータを BINARY で設定 / 問い合わせします。	7-57
:SEARch<x>:SLOGic:I2CBus[:SETup]:DATA:PMODe	ロジック I²C バス信号サーチのデータのパターン比較先頭位置モードを設定 / 問い合わせします。	7-57
:SEARch<x>:SLOGic:I2CBus[:SETup]:DATA:TRACe	ロジック I²C バス信号サーチのデータのトレースを設定 / 問い合わせします。	7-57
:SEARch<x>:SLOGic:I2CBus[:SETup]:GCALl?	ロジック I²C バス信号サーチのジェネラルコールに関するすべての設定値を問い合わせます。	7-57
:SEARch<x>:SLOGic:I2CBus[:SETup]:GCALl:BIT7maddress?	ロジック I²C バス信号サーチのジェネラルコールの 7bit マスタアドレスに関するすべての設定値を問い合わせます。	7-57
:SEARch<x>:SLOGic:I2CBus[:SETup]:GCALl:BIT7maddress:HEXA	ロジック I²C バス信号サーチのジェネラルコールの 7bit マスタアドレスを HEXA で設定します。	7-57
:SEARch<x>:SLOGic:I2CBus[:SETup]:GCALl:BIT7maddress:PATTern	ロジック I²C バス信号サーチのジェネラルコールの 7bit マスタアドレスを BINARY で設定 / 問い合わせします。	7-58
:SEARch<x>:SLOGic:I2CBus[:SETup]:GCALl:SBYTe (Second Byte)	ロジック I²C バス信号サーチのジェネラルコールのセカンドバイトのタイプを設定 / 問い合わせします。	7-58
:SEARch<x>:SLOGic:I2CBus[:SETup]:MODE	ロジック I²C バス信号サーチのサーチモードを設定 / 問い合わせします。	7-58
:SEARch<x>:SLOGic:I2CBus[:SETup]:NAIGnore?	ロジック I²C バス信号サーチの NON ACK 無視モードに関するすべての設定値を問い合わせます。	7-58
:SEARch<x>:SLOGic:I2CBus[:SETup]:NAIGnore:HSMode	ロジック I²C バス信号サーチのハイスピードモードで NON ACK を無視する / しないを設定 / 問い合わせします。	7-58
:SEARch<x>:SLOGic:I2CBus[:SETup]:NAIGnore:RACCess	ロジック I²C バス信号サーチのリードアクセスモードで NON ACK を無視する / しないを設定 / 問い合わせします。	7-58
:SEARch<x>:SLOGic:I2CBus[:SETup]:NAIGnore:SBYTe (Start Byte)	ロジック I²C バス信号サーチのスタートバイトで NON ACK を無視する / しないを設定 / 問い合わせします。	7-59
:SEARch<x>:SLOGic:I2CBus[:SETup]:SBHSmode?	ロジック I²C バス信号サーチのスタートバイト / ハイスピードモードに関するすべての設定値を問い合わせます。	7-59
:SEARch<x>:SLOGic:I2CBus[:SETup]:SBHSmode:TYPE	ロジック I²C バス信号サーチのスタートバイト / ハイスピードモードのタイプを設定 / 問い合わせします。	7-59
:SEARch<x>:SLOGic:LINBus?	ロジック LIN バス信号サーチに関するすべての設定値を問い合わせます。	7-59
:SEARch<x>:SLOGic:LINBus[:SETup]?	ロジック LIN バス信号サーチのセットアップに関するすべての設定値を問い合わせます。	7-59
SEARch<x>:SLOGic:LINBus[:SETup]:BLENgth	ロジック LIN バス信号サーチの Break length を設定 / 問い合わせします。	7-59
:SEARch<x>:SLOGic:LINBus[:SETup]:BRATe	ロジック LIN バス信号サーチのビットレート (データ転送速度) を設定 / 問い合わせします。	7-59

コマンド	機能	ページ
:SEARch<x>:SLOGic:LINBus[:SETup]:DATA?	ロジック LIN バス信号サーチのデータに関するすべての設定値を問い合わせます。	7-59
:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:BNUM	ロジック LIN バス信号サーチのデータのバイト数を設定 / 問い合わせします。	7-60
:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:BORDer	ロジック LIN バス信号サーチのデータのバイトオーダを設定 / 問い合わせします。	7-60
:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:CONDition	ロジック LIN バス信号サーチのデータ条件を設定 / 問い合わせします。	7-60
:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:DATA<x>	ロジック LIN バス信号サーチのデータの比較データを設定 / 問い合わせします。	7-60
:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:HEXA	ロジック LIN バス信号サーチのデータを HEXA で設定します。	7-60
:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:MSBLsb	ロジック LIN バス信号サーチのデータの MSB/LSB のビットを設定 / 問い合わせします。	7-61
:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:PATTern	ロジック LIN バス信号サーチのデータを BINARY で設定 / 問い合わせします。	7-61
:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:SIGN	ロジック LIN バス信号サーチのデータの符号を設定 / 問い合わせします。	7-61
:SEARch<x>:SLOGic:LINBus[:SETup]:ERRor?	ロジック LIN バス信号サーチの Error に関するすべてのを設定値を問い合わせします。	7-61
:SEARch<x>:SLOGic:LINBus[:SETup]:ERRor:CHECksum	ロジック LIN バス信号サーチの Checksum Error を設定 / 問い合わせします。	7-61
:SEARch<x>:SLOGic:LINBus[:SETup]:ERRor:FRAMing	ロジック LIN バス信号サーチの Framing Error を設定 / 問い合わせします。	7-61
:SEARch<x>:SLOGic:LINBus[:SETup]:ERRor:PARity	ロジック LIN バス信号サーチの Parity Error を設定 / 問い合わせします。	7-62
:SEARch<x>:SLOGic:LINBus[:SETup]:ERRor:SYNCh	ロジック LIN バス信号サーチの Synch Error を設定 / 問い合わせします。	7-62
:SEARch<x>:SLOGic:LINBus[:SETup]:ERRor:TOUT	ロジック LIN バス信号サーチの Timeout Error を設定 / 問い合わせします。	7-62
:SEARch<x>:SLOGic:LINBus[:SETup]:ID?	ロジック LIN バス信号サーチの ID に関するすべての設定値を問い合わせます。	7-62
:SEARch<x>:SLOGic:LINBus[:SETup]:ID:HEXA	ロジック LIN バス信号サーチの ID を HEXA で設定します。	7-62
:SEARch<x>:SLOGic:LINBus[:SETup]:ID:PATTern	ロジック LIN バス信号サーチの ID を BINARY で設定 / 問い合わせします。	7-62
:SEARch<x>:SLOGic:LINBus[:SETup]:MODE	ロジック LIN バス信号サーチのモードを設定 / 問い合わせします。	7-63
:SEARch<x>:SLOGic:LINBus[:SETup]:REVision	ロジック LIN バス信号サーチのレビジョン (1.3 or 2.0 or Both) を設定 / 問い合わせします。	7-63
:SEARch<x>:SLOGic:LINBus[:SETup]:SPOint	ロジック LIN バス信号サーチのサンプルポイントを設定 / 問い合わせします。	7-63
:SEARch<x>:SLOGic:LINBus[:SETup]:TRACe	ロジック LIN バス信号サーチのトレースを設定 / 問い合わせします。	7-63
:SEARch<x>:SLOGic:SPIBus?	ロジック SPI バス信号サーチに関するすべての設定値を問い合わせます。	7-63
:SEARch<x>:SLOGic:SPIBus:CLOCk?	ロジック SPI バス信号サーチのクロック信号のチャネルに関するすべての設定値を問い合わせます。	7-63
:SEARch<x>:SLOGic:SPIBus:CLOCk:POLarity	ロジック SPI バス信号サーチのクロック信号のチャネルの極性を設定 / 問い合わせします。	7-63
:SEARch<x>:SLOGic:SPIBus:CLOCk:SOURce	ロジック SPI バス信号サーチのクロック信号のチャネルを設定 / 問い合わせします。	7-64
:SEARch<x>:SLOGic:SPIBus:CS?	ロジック SPI バス信号サーチのチップセレクト信号のチャネルに関するすべての設定値を問い合わせます。	7-64
:SEARch<x>:SLOGic:SPIBus:CS:ACTive	ロジック SPI バス信号サーチのチップセレクト信号のチャネルのアクティブレベルを設定 / 問い合わせします。	7-64
:SEARch<x>:SLOGic:SPIBus:CS:TRACe	ロジック SPI バス信号サーチのチップセレクト信号のチャネルを設定 / 問い合わせします。	7-64
:SEARch<x>:SLOGic:SPIBus[:SETup]?	ロジック SPI バス信号サーチのセットアップに関するすべての設定値を問い合わせます。	7-64
:SEARch<x>:SLOGic:SPIBus[:SETup]:BITorder	ロジック SPI バス信号サーチのビットオーダを設定 / 問い合わせします。	7-64
:SEARch<x>:SLOGic:SPIBus[:SETup]:DATA<x>?	ロジック SPI バス信号サーチの各データに関するすべての設定値を問い合わせます。	7-64

コマンド	機能	ページ
:SEARch<x>:SLOGic:SPIBus[:SETup]:DATA<x>:BYTE	ロジック SPI バス信号サーチの各データのデータサイズ (バイト数) を設定 / 問い合わせします。	7-65
:SEARch<x>:SLOGic:SPIBus[:SETup]:DATA<x>:CONDition	ロジック SPI バス信号サーチの各データの判定方法 (一致 / 不一致) を設定 / 問い合わせします。	7-65
:SEARch<x>:SLOGic:SPIBus[:SETup]:DATA<x>:DPOSition	ロジック SPI バス信号サーチの各データのパターン比較先頭位置を設定 / 問い合わせします。	7-65
SEARch<x>:SLOGic:SPIBus[:SETup]:DATA<x>:DSIZe	ロジック SPI バス信号サーチの各データのフィールド数を設定 / 問い合わせします。	7-65
:SEARch<x>:SLOGic:SPIBus[:SETup]:DATA<x>:HEXA<x>	ロジック SPI バス信号サーチの各データを HEXA で設定します。	7-65
:SEARch<x>:SLOGic:SPIBus[:SETup]:DATA<x>:PATTern<x>	ロジック SPI バス信号サーチの各データを BINARY で設定 / 問い合わせします。	7-65
:SEARch<x>:SLOGic:SPIBus[:SETup]:DATA<x>:TRACe	ロジック SPI バス信号サーチの各データのソースチャネルを設定 / 問い合わせします。	7-66
:SEARch<x>:SLOGic:SPIBus[:SETup]:EMSBLSB	ロジック SPI バス信号サーチのフィールドの有効範囲を設定 / 問い合わせします。	7-66
:SEARch<x>:SLOGic:SPIBus[:SETup]:FSIZe	ロジック SPI バス信号サーチのフィールドサイズを設定 / 問い合わせします。	7-66
:SEARch<x>:SLOGic:SPIBus[:SETup]:ITIMe	ロジック SPI バス信号サーチのアイドル時間を設定 / 問い合わせします。	7-66
:SEARch<x>:SLOGic:SPIBus[:SETup]:MODE	ロジック SPI バス信号サーチの結線方式 (3 線式 /4 線式) を設定 / 問い合わせします。	7-66
:SEARch<x>:SLOGic:UART?	ロジック UART 信号サーチに関するすべて設定値を問い合わせします。	7-66
:SEARch<x>:SLOGic:UART:BRATe	ロジック UART 信号サーチのビットレート (データ転送速度) を設定 / 問い合わせします。	7-67
:SEARch<x>:SLOGic:UART:DATA?	ロジック UART 信号サーチのデータに関するすべての設定値を問い合わせします。	7-67
:SEARch<x>:SLOGic:UART:DATA:BITorder	ロジック UART 信号サーチのデータのビットオーダを設定 / 問い合わせします。	7-67
:SEARch<x>:SLOGic:UART:DATA:DSIZe	ロジック UART 信号サーチのデータのバイト数を設定 / 問い合わせします。	7-67
:SEARch<x>:SLOGic:UART:DATA:HEXA	ロジック UART 信号サーチのデータを HEXA で設定します。	7-67
:SEARch<x>:SLOGic:UART:DATA:PATTern	ロジック UART 信号サーチのデータを BINARY で設定 / 問い合わせします。	7-67
:SEARch<x>:SLOGic:UART:ERRor?	ロジック UART 信号サーチの Error に関するすべてのを設定値を問い合わせします。	7-67
:SEARch<x>:SLOGic:UART:ERRor:FRAMing	ロジック UART 信号サーチの Framing Error を設定 / 問い合わせします。	7-67
:SEARch<x>:SLOGic:UART:ERRor:PARity	ロジック UART 信号サーチの Parity Error を設定 / 問い合わせします。	7-68
::SEARch<x>:SLOGic:UART:ERRor:PMODe	ロジック UART 信号サーチの Parity モードを設定 / 問い合わせします。	7-68
:SEARch<x>:SLOGic:UART:FORMat	ロジック UART 信号サーチのフォーマットを設定 / 問い合わせします。	7-68
:SEARch<x>:SLOGic:UART:MODE	ロジック UART 信号サーチのモードを設定 / 問い合わせします。	7-68
:SEARch<x>:SLOGic:UART:POLarity	ロジック UART 信号サーチの極性を設定 / 問い合わせします。	7-68
:SEARch<x>:SLOGic:UART:SPOint	ロジック UART 信号サーチのサンプルポイントを設定 / 問い合わせします。	7-68
:SEARch<x>:SLOGic:UART:TRACe	ロジック UART 信号サーチのトレースを設定 / 問い合わせします。	7-68
:SEARch<x>:SPIBus?	SPI バス信号サーチに関するすべての設定値を問い合わせします。	7-69
:SEARch<x>:SPIBus:CLOCk	SPI バス信号サーチのクロック信号のチャネルに関するすべての設定値を問い合わせします。	7-69
:SEARch<x>:SPIBus:CLOCk:POLarity	SPI バス信号サーチのクロック信号のチャネルの極性を設定 / 問い合わせします。	7-69
:SEARch<x>:SPIBus:CLOCk:SOURce	SPI バス信号サーチのクロック信号のチャネルを設定 / 問い合わせします。	7-69
:SEARch<x>:SPIBus:CS?	SPI バス信号サーチのチップセレクト信号のチャネルに関するすべての設定値を問い合わせします。	7-69

コマンド	機能	ページ
:SEARch<x>:SPIBus:CS:ACTive	SPI バス信号サーチのチップセレクト信号のチャネルのアクティブレベルを設定 / 問い合わせします。	7-69
:SEARch<x>:SPIBus:CS:TRACe	SPI バス信号サーチのチップセレクト信号のチャネルを設定 / 問い合わせします。	7-69
:SEARch<x>:SPIBus:SETup?	SPI バス信号サーチのセットアップに関するすべての設定値を問い合わせします。	7-69
:SEARch<x>:SPIBus[:SETup]:BITorder	SPI バス信号サーチのビットオーダを設定 / 問い合わせします。	7-69
:SEARch<x>:SPIBus[:SETup]:DATA<x>?	SPI バス信号サーチの各データに関するすべての設定値を問い合わせします。	7-69
:SEARch<x>:SPIBus[:SETup]:DATA<x>:BYTE	SPI バス信号サーチの各データのデータサイズ (バイト数) を設定 / 問い合わせします。	7-69
:SEARch<x>:SPIBus[:SETup]:DATA<x>:CONDition	SPI バス信号サーチの各データの判定方法 (一致 / 不一致) を設定 / 問い合わせします。	7-70
:SEARch<x>:SPIBus[:SETup]:DATA<x>:DPOSition	SPI バス信号サーチの各データのパターン比較先頭位置を設定 / 問い合わせします。	7-70
:SEARch<x>:SPIBus[:SETup]:DATA<x>:DSIZe	SPI バス信号サーチの各データのフィールド数を設定 / 問い合わせします。	7-70
:SEARch<x>:SPIBus[:SETup]:DATA<x>:HEXA<x>	SPI バス信号サーチの各データを HEXA で設定します。	7-70
:SEARch<x>:SPIBus[:SETup]:DATA<x>:PATTern<x>	SPI バス信号サーチの各データを BINARY で設定 / 問い合わせします。	7-70
:SEARch<x>:SPIBus[:SETup]:DATA<x>:TRACe	SPI バス信号サーチの各データのソースチャネルを設定 / 問い合わせします。	7-70
:SEARch<x>:SPIBus[:SETup]:EMSBLSB	SPI バス信号サーチのフィールドの有効範囲を設定 / 問い合わせします。	7-71
:SEARch<x>:SPIBus[:SETup]:FSIZe	SPI バス信号サーチのフィールドサイズを設定 / 問い合わせします。	7-71
:SEARch<x>:SPIBus[:SETup]:ITIMe	SPI バス信号サーチのアイドル時間を設定 / 問い合わせします。	7-71
:SEARch<x>:SPIBus[:SETup]:MODE:	SPI バス信号サーチの結線方式 (3 線式 /4 線式) を設定 / 問い合わせします。	7-71
:SEARch<x>:TRACe<x>:LEVel	各ソースチャネルのしきい値 (Threshold) レベルを設定 / 問い合わせします。	7-71
:SEARch<x>:TYPE	サーチタイプを設定 / 問い合わせします。	7-71
:SEARch<x>:UART?	UART 信号サーチに関するすべて設定値を問い合わせします。	7-71
:SEARch<x>:UART:BRATe	UART 信号サーチのビットレート (データ転送速度) を設定 / 問い合わせします。	7-72
:SEARch<x>:UART:DATA?	UART 信号サーチのデータに関するすべての設定値を問い合わせします。	7-72
:SEARch<x>:UART:DATA:BITorder	UART 信号サーチのデータのビットオーダを設定 / 問い合わせします。	7-72
:SEARch<x>:UART:DATA:DSIZe	UART 信号サーチのデータのバイト数を設定 / 問い合わせします。	7-72
:SEARch<x>:UART:DATA:HEXA	UART 信号サーチのデータを HEXA で設定します。	7-72
:SEARch<x>:UART:DATA:PATTern	UART 信号サーチのデータを BINARY で設定 / 問い合わせします。	7-72
:SEARch<x>:UART:ERRor?	UART 信号サーチの Error に関するすべてのを設定値を問い合わせします。	7-72
:SEARch<x>:UART:ERRor:FRAMing	UART 信号サーチの Framing Error を設定 / 問い合わせします。	7-72
:SEARch<x>:UART:ERRor:PARity	UART 信号サーチの Parity Error を設定 / 問い合わせします。	7-72
:SEARch<x>:UART:ERRor:PMODe	UART 信号サーチの Parity モードを設定 / 問い合わせします。	7-72
:SEARch<x>:UART:FORMat	UART 信号サーチのフォーマットを設定 / 問い合わせします。	7-73
:SEARch<x>:UART:MODE	UART 信号サーチのモードを設定 / 問い合わせします。	7-73
:SEARch<x>:UART:POLarity	UART 信号サーチの極性を設定 / 問い合わせします。	7-73

コマンド	機能	ページ
:SEARch<x>:UART:SPOint	UART 信号サーチのサンプルポイントを設定 / 問い合わせします。	7-73
:SEARch<x>:UART:TRACe	UART 信号サーチのトレースを設定 / 問い合わせします。	7-73

SERialbus グループ

:SERialbus?	シリアルバスセットアップに関するすべての設定値を問い合わせます。	7-74
:SERialbus:SETup<x>?	シリアルバスセットアップの各セットアップに関するすべての設定値を問い合わせます。	7-74
:SERialbus:SETup<x>:ASETup:ABORt	シリアルバスセットアップのオートセットアップを中止します。	7-74
:SERialbus:SETup<x>:ASETup:EXECute	シリアルバスセットアップのオートセットアップを実行します。	7-74
:SERialbus:SETup<x>:ASETup:UNDO	シリアルバスセットアップの実行したオートセットアップを取り消します。	7-74
:SERialbus:SETup<x>:CANBus?	CAN バスセットアップに関するすべての設定値を問い合わせます。	7-74
:SERialbus:SETup<x>:CANBus:BRATe	CAN バスセットアップのビットレート (データ転送速度) を設定 / 問い合わせします。	7-74
:SERialbus:SETup<x>:CANBus:RECessive	CAN バスセットアップのリセッシブレベル (バスレベル) を設定 / 問い合わせします。	7-74
:SERialbus:SETup<x>:CANBus:SPOint	CAN バスセットアップのサンプルポイントを設定 / 問い合わせします。	7-74
:SERialbus:SETup<x>:CANBus:TRACe	CAN バスセットアップのトレースを設定 / 問い合わせします。	7-75
:SERialbus:SETup<x>:I2CBus?	I2C バスセットアップに関するすべての設定値を問い合わせます。	7-75
:SERialbus:SETup<x>:I2CBus:CLOCk	I2C バスセットアップのクロックチャネルを設定 / 問い合わせします。	7-75
:SERialbus:SETup<x>:I2CBus:DTRace	I2C バス信号解析のデータチャネルを設定 / 問い合わせします。	7-75
:SERialbus:SETup<x>:LINBus?	LIN バスセットアップに関するすべての設定値を問い合わせます。	7-75
:SERialbus:SETup<x>:LINBus:BRATe	LIN バスセットアップのビットレート (データ転送速度) を設定 / 問い合わせします。	7-75
:SERialbus:SETup<x>:LINBus:REVision	LIN バスセットアップのレビジョン (1.3 or 2.0 or Both) を設定 / 問い合わせします。	7-75
:SERialbus:SETup<x>:LINBus:SPOint	LIN バスセットアップのサンプルポイントを設定 / 問い合わせします。	7-75
:SERialbus:SETup<x>:LINBus:TRACe	LIN バスセットアップのトレースを設定 / 問い合わせします。	7-76
:SERialbus:SETup<x>:SPIBus?	SPI バスセットアップに関するすべての設定値を問い合わせます。	7-76
:SERialbus:SETup<x>:SPIBus:BITorder	SPI バスセットアップのビットオーダを設定 / 問い合わせします。	7-76
:SERialbus:SETup<x>:SPIBus:CLOCk?	SPI バスセットアップのクロック信号のチャネルに関するすべての設定値を問い合わせします。	7-76
SERialbus:SETup<x>:SPIBus:CLOCk:POLarity	SPI バスセットアップのクロック信号のチャネルの極性を設定 / 問い合わせします。	7-76
:SERialbus:SETup<x>:SPIBus:CLOCk:TRACe	SPI バスセットアップのクロック信号のチャネルを設定 / 問い合わせします。	7-76
:SERialbus:SETup<x>:SPIBus:CS?	SPI バスセットアップのチップセレクト信号のチャネルに関するすべての設定値を問い合わせします。	7-76
:SERialbus:SETup<x>:SPIBus:CS:ACTive	SPI バスセットアップのチップセレクト信号のチャネルのアクティブレベルを設定 / 問い合わせします。	7-76
:SERialbus:SETup<x>:SPIBus:CS:TRACe	SPI バスセットアップのチップセレクト信号のチャネルを設定 / 問い合わせします。	7-77
:SERialbus:SETup<x>:SPIBus:DATA<x>?	SPI バスセットアップの各データに関するすべての設定値を問い合わせします。	7-77
:SERialbus:SETup<x>:SPIBus:DATA<x>:ACTive	SPI バスセットアップの各データのアクティブレベルを設定 / 問い合わせします。	7-77
:SERialbus:SETup<x>:SPIBus:DATA<x>:TRACe	SPI バスセットアップの各データチャネルを設定 / 問い合わせします。	7-77
:SERialbus:SETup<x>:SPIBus:ITIMe	SPI バスセットアップのアイドル時間を設定 / 問い合わせします。	7-77
:SERialbus:SETup<x>:SPIBus:MODE	SPI バスセットアップの結線方式 (3 線式 /4 線式) を設定 / 問い合わせします。	7-77
:SERialbus:SETup<x>:TRACe<x>?	各トレースに関するすべての設定値を問い合わせします。	7-77

コマンド	機能	ページ
:SERialbus:SETup<x>:TRACe<x>:HYSTeresis	各トレースのしきい値 (Threshold) のヒステリシスを設定 / 問い合わせします。	7-78
:SERialbus:SETup<x>:TRACe<x>:LEVel	各トレースのしきい値 (Threshold) レベルを設定 / 問い合わせします。	7-78
:SERialbus:SETup<x>:TYPE	シリアルバスセットアップのタイプを設定 / 問い合わせします。	7-78
:SERialbus:SETup<x>:UART?	UART バスセットアップに関するすべての設定値を問い合わせます。	7-78
:SERialbus:SETup<x>:UART:BITorder	UART バスセットアップのビットオーダを設定 / 問い合わせします。	7-78
:SERialbus:SETup<x>:UART:BRATe	UART バスセットアップのビットレート (データ転送速度) を設定 / 問い合わせします。	7-78
:SERialbus:SETup<x>:UART:FORMat	UART バスセットアップのデータ形式を設定 / 問い合わせします。	7-78
:SERialbus:SETup<x>:UART:PMODe	UART バスセットアップの Parity モードを設定 / 問い合わせします。	7-78
:SERialbus:SETup<x>:UART:POLarity	UART バスセットアップの極性を設定 / 問い合わせします。	7-79
:SERialbus:SETup<x>:UART:SPOint	UART バスセットアップのサンプルポイントを設定 / 問い合わせします。	7-79
:SERialbus:SETup<x>:UART:TRACe	UART バスセットアップのソースを設定 / 問い合わせします。	7-79
:SERialbus:TLINk	シリアルバスセットアップのトリガリンクを設定 / 問い合わせします。	7-79

TRIGger グループ

:TRIGger:EINTerval:EVENt<x>:CANBus?	各イベントの CAN バス信号トリガに関するすべての設定値を問い合わせます。	7-80
:TRIGger:EINTerval:EVENt<x>:CANBus:ACK	CAN バス信号トリガの ACK 条件を設定 / 問い合わせします。	7-80
:TRIGger:EINTerval:EVENt<x>:CANBus:BRATe	CAN バス信号トリガのビットレート (データ転送速度) を設定 / 問い合わせします。	7-80
:TRIGger:EINTerval:EVENt<x>:CANBus:DATA?	CAN バス信号トリガのデータに関するすべての設定値を問い合わせます。	7-80
:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:BORDer	CAN バス信号トリガのデータのバイトオーダを設定 / 問い合わせします。	7-80
:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:CONDition	CAN バス信号トリガのデータ条件を設定 / 問い合わせします。	7-80
:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:DATA<x>	CAN バス信号トリガのデータの比較データを設定 / 問い合わせします。	7-81
:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:DLC	CAN バス信号トリガのデータの有効バイト数 (DLC) を設定 / 問い合わせします。	7-81
:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:HEXA	CAN バス信号トリガのデータを HEXA で設定します。	7-81
:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:MSBLsb	CAN バス信号トリガのデータの MSB/LSB のビットを設定 / 問い合わせします。	7-81
:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:PATTern	CAN バス信号トリガのデータを BINARY で設定 / 問い合わせします。	7-81
:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:SIGN	CAN バス信号トリガのデータの符号を設定 / 問い合わせします。	7-82
:TRIGger:EINTerval:EVENt<x>:CANBus:IDEXt?	CAN バス信号トリガの拡張フォーマットの ID に関するすべての設定値を問い合わせます。	7-82
:TRIGger:EINTerval:EVENt<x>:CANBus:IDEXt:HEXA	CAN バス信号トリガの拡張フォーマットの ID を HEXA で設定します。	7-82
:TRIGger:EINTerval:EVENt<x>:CANBus:IDEXt:PATTern	CAN バス信号トリガの拡張フォーマットの ID を BINARY で設定 / 問い合わせします。	7-82
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR?	CAN バス信号トリガの OR 条件に関するすべての設定値を問い合わせます。	7-82
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>?	CAN バス信号トリガの OR 条件の各 ID に関するすべての設定値を問い合わせます。	7-82
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:ACK	CAN バス信号トリガの OR 条件の各 ACK 条件を設定 / 問い合わせします。	7-82
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA?	CAN バス信号トリガの OR 条件の各データに関するすべての設定値を問い合わせます。	7-82
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:BORDer	CAN バス信号トリガの OR 条件の各データのバイトオーダを設定 / 問い合わせします。	7-83

コマンド	機能	ページ
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:CONDition	CAN バス信号トリガの OR 条件の各データ条件を設定 / 問い合わせします。	7-83
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:DATA<x>	CAN バス信号トリガの OR 条件の各データの比較データを設定 / 問い合わせします。	7-83
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:DLC	CAN バス信号トリガの OR 条件の各データの有効バイト数 (DLC) を設定 / 問い合わせします。	7-83
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:HEXA	CAN バス信号トリガの OR 条件の各データを HEXA で設定します。	7-84
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:MSBLsb	CAN バス信号トリガの OR 条件の各データの MSB/LSB のビットを設定 / 問い合わせします。	7-84
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:PATTern	CAN バス信号トリガの OR 条件の各データを BINARY で設定 / 問い合わせします。	7-84
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:SIGN	CAN バス信号トリガの OR 条件の各データの符号を設定 / 問い合わせします。	7-84
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:FORMat	CAN バス信号トリガの OR 条件の各メッセージフォーマット (標準 / 拡張) を設定 / 問い合わせします。	7-84
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:IDEXt?	CAN バス信号トリガの OR 条件の各拡張フォーマットの ID に関するすべての設定値を問い合わせます。	7-84
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:IDEXt:HEXA	CAN バス信号トリガの OR 条件の各拡張フォーマットの ID を HEXA で設定します。	7-84
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:IDEXt:PATTern	CAN バス信号トリガの OR 条件の各拡張フォーマットの ID を BINARY で設定 / 問い合わせします。	7-85
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:IDSTd?	CAN バス信号トリガの OR 条件の各標準フォーマットの ID に関するすべての設定値を問い合わせます。	7-85
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:IDSTd:HEXA	CAN バス信号トリガの OR 条件の各標準フォーマットの ID を HEXA で設定します。	7-85
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:IDSTd:PATTern	CAN バス信号トリガの OR 条件の各標準フォーマットの ID を BINARY で設定 / 問い合わせします。	7-85
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:MODE	CAN バス信号トリガの OR 条件の各条件の有効 (1)/ 無効 (0) を設定 / 問い合わせします。	7-85
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:RTR	CAN バス信号トリガの OR 条件の各 RTR を設定 / 問い合わせします。	7-85
:TRIGger:EINTerval:EVENt<x>:CANBus:IDSTd?	CAN バス信号トリガの標準フォーマットの ID に関するすべての設定値を問い合わせます。	7-86
:TRIGger:EINTerval:EVENt<x>:CANBus:IDSTd:HEXA	CAN バス信号トリガの標準フォーマットの ID を HEXA で設定します。	7-86
:TRIGger:EINTerval:EVENt<x>:CANBus:IDSTd:PATTern	CAN バス信号トリガの標準フォーマットの ID を BINARY で設定 / 問い合わせします。	7-86
:TRIGger:EINTerval:EVENt<x>:CANBus:MODE	CAN バス信号トリガのモードを設定 / 問い合わせします。	7-86
:TRIGger:EINTerval:EVENt<x>:CANBus:RECessive	CAN バス信号トリガのリセッシブレベル (バスレベル) を設定 / 問い合わせします。	7-86
:TRIGger:EINTerval:EVENt<x>:CANBus:RTR	CAN バス信号トリガの RTR を設定 / 問い合わせします。	7-86
:TRIGger:EINTerval:EVENt<x>:CANBus:SOURce	CAN バス信号トリガのトリガソースを設定 / 問い合わせします。	7-86
:TRIGger:EINTerval:EVENt<x>:CANBus:SPOint	CAN バス信号トリガのサンプルポイントを設定 / 問い合わせします。	7-87
:TRIGger:EINTerval:EVENt<x>:I2CBus?	各イベントの I^2C バストリガに関するすべての設定値を問い合わせます。	7-87
:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa?	I^2C バストリガのアドレスに関するすべての設定値を問い合わせます。	7-87
:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT10address?	I^2C バストリガの 10bit アドレスに関するすべての設定値を問い合わせます。	7-87
:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT10address:HEXA	I^2C バストリガの 10bit アドレスを HEXA で設定します。	7-87
:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT10address:PATTern	I^2C バストリガの 10bit アドレスを BINARY で設定 / 問い合わせします。	7-87
:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7ADdress?	I^2C バストリガの 7bit アドレスに関するすべての設定値を問い合わせます。	7-87
:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7ADdress:HEXA	I^2C バストリガの 7bit アドレスを HEXA で設定します。	7-88
:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7ADdress:PATTern	I^2C バストリガの 7bit アドレスを BINARY で設定 / 問い合わせします。	7-88
:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7APsub?	I^2C バストリガの 7bit + Sub アドレスに関するすべての設定値を問い合わせます。	7-88

コマンド	機能	ページ
:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7APsub:ADDRess?	I²C バストリガの 7bit ＋ Sub アドレスの 7bit アドレスに関するすべての設定値を問い合わせます。	7-88
:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7APsub:ADDRess:HEXA	I²C バストリガの 7bit ＋ Sub アドレスの 7bit アドレスを HEXA で設定します。	7-88
:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7APsub:ADDRess:PATTern	I²C バストリガの 7bit ＋ Sub アドレスの 7bit アドレスを BINARY で設定 / 問い合わせします。	7-88
:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7APsub:SADDress?	I²C バストリガの 7bit ＋ Sub アドレスの Sub アドレスに関するすべての設定値を問い合わせます。	7-88
:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7APsub:SADDress:HEXA	I²C バストリガの 7bit ＋ Sub アドレスの Sub アドレスを HEXA で設定します。	7-88
:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7APsub:SADDress:PATTern	I²C バストリガの 7bit ＋ Sub アドレスの Sub アドレスを BINARY で設定 / 問い合わせします。	7-89
:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:TYPE	I²C バストリガのアドレスの種類を設定 / 問い合わせします。	7-89
:TRIGger:EINTerval:EVENt<x>:I2CBus:CLOCk?	I²C バストリガのクロックに関するすべての設定値を問い合わせます。	7-89
:TRIGger:EINTerval:EVENt<x>:I2CBus:CLOCk:SOURce	I²C バストリガのクロックトレースを設定 / 問い合わせします。	7-89
:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA?	I²C バストリガのデータに関するすべての設定値を問い合わせます。	7-89
:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA:BYTE	I²C バストリガの設定データ数を設定 / 問い合わせします。	7-89
:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA:CONDition	I²C バストリガのデータの判定方法 (一致 / 不一致) を設定 / 問い合わせします。	7-89
:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA:DPOSition	I²C バストリガのデータのパターン比較する位置を設定 / 問い合わせします。	7-90
:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA:HEXA<x>	I²C バストリガのデータを HEXA で設定します。	7-90
:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA:MODE	I²C バストリガのデータ条件の有効 / 無効を設定 / 問い合わせします。	7-90
:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA:PATTern<x>	I²C バストリガのデータを BINARY で設定 / 問い合わせします。	7-90
:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA:PMODe	I²C バストリガのデータのパターン比較先頭位置モードを設定 / 問い合わせします。	7-90
:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA:SOURce	I²C バストリガのデータトレースを設定 / 問い合わせします。	7-90
:TRIGger:EINTerval:EVENt<x>:I2CBus:GCALl?	I²C バストリガのジェネラルコールに関するすべての設定値を問い合わせます。	7-90
:TRIGger:EINTerval:EVENt<x>:I2CBus:GCALl:BIT7maddress?	I²C バストリガのジェネラルコールの 7bit マスタアドレスに関するすべての設定値を問い合わせます。	7-91
:TRIGger:EINTerval:EVENt<x>:I2CBus:GCALl:BIT7maddress:HEXA	I²C バストリガのジェネラルコールの 7bit マスタアドレスを HEXA で設定します。	7-91
:TRIGger:EINTerval:EVENt<x>:I2CBus:GCALl:BIT7maddress:PATTern	I²C バストリガのジェネラルコールの 7bit マスタアドレスを BINARY で設定 / 問い合わせします。	7-91
:TRIGger:EINTerval:EVENt<x>:I2CBus:GCALl:SBYTe (Second Byte)	I²C バストリガのジェネラルコールのセカンドバイトのタイプを設定 / 問い合わせします。	7-91
:TRIGger:EINTerval:EVENt<x>:I2CBus:MODE	I²C バストリガのトリガモードを設定 / 問い合わせします。	7-91
:TRIGger:EINTerval:EVENt<x>:I2CBus:NAIGnore?	I²C バストリガの NON ACK 無視モードに関するすべての設定値を問い合わせます。	7-91
:TRIGger:EINTerval:EVENt<x>:I2CBus:NAIGnore:HSMode	I²C バストリガのハイスピードモードで NON ACK を無視する / しないを設定 / 問い合わせします。	7-91
:TRIGger:EINTerval:EVENt<x>:I2CBus:NAIGnore:RACCess	I²C バストリガのリードアクセスモードで NON ACK を無視する / しないを設定 / 問い合わせします。	7-91
:TRIGger:EINTerval:EVENt<x>:I2CBus:NAIGnore:SBYTe (Start Byte)	I²C バストリガのスタートバイトで NON ACK を無視する / しないを設定 / 問い合わせします。	7-92
:TRIGger:EINTerval:EVENt<x>:I2CBus:SBHSmode?	I²C バストリガのスタートバイト / ハイスピードモードに関するすべての設定値を問い合わせます。	7-92
:TRIGger:EINTerval:EVENt<x>:I2CBus:SBHSmode:TYPE	I²C バストリガのスタートバイト / ハイスピードモードのタイプを設定 / 問い合わせします。	7-92
:TRIGger:EINTerval:EVENt<x>:LINBus?	各イベントの LIN バス信号トリガに関するすべての設定値を問い合わせします。	7-92
:TRIGger:EINTerval:EVENt<x>:LINBus:BRATe	LIN バス信号トリガのビットレート (データ転送速度) を設定 / 問い合わせします。	7-92

コマンド	機能	ページ
:TRIGger:EINTerval:EVENt<x>:LINBus:SOURce	LIN バス信号トリガのトリガソースを設定 / 問い合わせします。	7-92
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus?	各イベントのロジック I^2C バストリガに関するすべての設定値を問い合わせます。	7-92
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa?	ロジック I^2C バストリガのアドレスに関するすべての設定値を問い合わせます。	7-92
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:BIT10address?	ロジック I^2C バストリガの 10bit アドレスに関するすべての設定値を問い合わせます。	7-92
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:BIT10address:HEXA	ロジック I^2C バストリガの 10bit アドレスを HEXA で設定します。	7-93
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:BIT10address:PATTern	ロジック I^2C バストリガの 10bit アドレスを BINARY で設定 / 問い合わせします。	7-93
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:BIT7ADdress?	ロジック I^2C バストリガの 7bit アドレスに関するすべての設定値を問い合わせます。	7-93
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:BIT7ADdress:HEXA	ロジック I^2C バストリガの 7bit アドレスを HEXA で設定します。	7-93
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:BIT7ADdress:PATTern	ロジック I^2C バストリガの 7bit アドレスを BINARY で設定 / 問い合わせします。	7-93
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:BIT7APsub?	ロジック I^2C バストリガの 7bit ＋ Sub アドレスに関するすべての設定値を問い合わせます。	7-93
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:BIT7APsub:ADDRess?	ロジック I^2C バストリガの 7bit ＋ Sub アドレスの 7bit アドレスに関するすべての設定値を問い合わせます。	7-93
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:BIT7APsub:ADDRess:HEXA	ロジック I^2C バストリガの 7bit ＋ Sub アドレスの 7bit アドレスを HEXA で設定します。	7-93
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:BIT7APsub:ADDRess:PATTern	ロジック I^2C バストリガの 7bit ＋ Sub アドレスの 7bit アドレスを BINARY で設定 / 問い合わせします。	7-94
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:BIT7APsub:SADDress?	ロジック I^2C バストリガの 7bit ＋ Sub アドレスの Sub アドレスに関するすべての設定値を問い合わせます。	7-94
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:BIT7APsub:SADDress:HEXA	ロジック I^2C バストリガの 7bit ＋ Sub アドレスの Sub アドレスを HEXA で設定します。	7-94
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:BIT7APsub:SADDress:PATTern	ロジック I^2C バストリガの 7bit ＋ Sub アドレスの Sub アドレスを BINARY で設定 / 問い合わせします。	7-94
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:TYPE	ロジック I^2C バストリガのアドレスの種類を設定 / 問い合わせします。	7-94
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:CLOCk?	ロジック I^2C バストリガのクロックに関するすべての設定値を問い合わせます。	7-94
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:CLOCk:SOURce	ロジック I^2C バストリガのクロックトレースを設定 / 問い合わせします。	7-95
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA?	ロジック I^2C バストリガのデータに関するすべての設定値を問い合わせます。	7-95
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:BYTE	ロジック I^2C バストリガの設定データ数を設定 / 問い合わせします。	7-95
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:CONDition	ロジック I^2C バストリガのデータの判定方法 (一致 / 不一致) を設定 / 問い合わせします。	7-95
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:DPOSition	ロジック I^2C バストリガのデータのパターン比較する位置を設定 / 問い合わせします。	7-95
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:HEXA<x>	ロジック I^2C バストリガのデータを HEXA で設定します。	7-95
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:MODE	ロジック I^2C バストリガのデータ条件の有効 / 無効を設定 / 問い合わせします。	7-96
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:PATTern<x>	ロジック I^2C バストリガのデータを BINARY で設定 / 問い合わせします。	7-96
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:PMODe	ロジック I^2C バストリガのデータのパターン比較先頭位置モードを設定 / 問い合わせします。	7-96
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:SOURce	ロジック I^2C バストリガのデータトレースを設定 / 問い合わせします。	7-96
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:GCALl?	ロジック I^2C バストリガのジェネラルコールに関するすべての設定値を問い合わせます。	7-96
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:GCALl:BIT7maddress?	ロジック I^2C バストリガのジェネラルコールの 7bit マスタアドレスに関するすべての設定値を問い合わせます。	7-96
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:GCALl:BIT7maddress:HEXA	ロジック I^2C バストリガのジェネラルコールの 7bit マスタアドレスを HEXA で設定します。	7-96

コマンド	機能	ページ
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:GCALl:BIT7maddress:PATTern	ロジック I²C バストリガのジェネラルコールの 7bit マスタアドレスを BINARY で設定 / 問い合わせします。	7-97
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:GCALl:SBYTe (Second Byte)	ロジック I²C バストリガのジェネラルコールのセカンドバイトのタイプを設定 / 問い合わせします。	7-97
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:MODE	ロジック I²C バストリガのトリガモードを設定 / 問い合わせします。	7-97
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:NAIGnore?	ロジック I²C バストリガの NON ACK 無視モードに関するすべての設定値を問い合わせます。	7-97
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:NAIGnore:HSMode	ロジック I²C バストリガのハイスピードモードで NON ACK を無視する / しないを設定 / 問い合わせします。	7-97
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:NAIGnore:RACCess	ロジック I²C バストリガのリードアクセスモードで NON ACK を無視する / しないを設定 / 問い合わせします。	7-97
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:NAIGnore:SBYTe (Start Byte)	ロジック I²C バストリガのスタートバイトで NON ACK を無視する / しないを設定 / 問い合わせします。	7-98
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:SBHSmode?	ロジック I²C バストリガのスタートバイト / ハイスピードモードに関するすべての設定値を問い合わせます。	7-98
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:SBHSmode:TYPE	ロジック I²C バストリガのスタートバイト / ハイスピードモードのタイプを設定 / 問い合わせします。	7-98
:TRIGger:EINTerval:EVENt<x>:LOGic:LINBus?	各イベントのロジック LIN バス信号トリガに関するすべての設定値を問い合わせします。	7-98
:TRIGger:EINTerval:EVENt<x>:LOGic:LINBus:BRATe	ロジック LIN バス信号トリガのビットレート (データ転送速度) を設定 / 問い合わせします。	7-98
:TRIGger:EINTerval:EVENt<x>:LOGic:LINBus:SOURce	ロジック LIN バス信号トリガのトリガソースを設定 / 問い合わせします。	7-98
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus?	各イベントのロジック SPI バストリガに関するすべての設定値を問い合わせます。	7-98
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:BITorder	ロジック SPI バストリガのビットオーダーを設定 / 問い合わせします。	7-99
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:CLOCk?	ロジック SPI バストリガのクロックに関するすべての設定値を問い合わせます。	7-99
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:CLOCk:POLarity	ロジック SPI バストリガのクロックトレースの極性を設定 / 問い合わせします。	7-99
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:CLOCk:SOURce	ロジック SPI バストリガのクロックトレースを設定 / 問い合わせします。	7-99
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:CS?	ロジック SPI バストリガのチップセレクトに関するすべての設定値を問い合わせます。	7-99
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:CS:ACTive	ロジック SPI バストリガのチップセレクトのアクティブレベルを設定 / 問い合わせします。	7-99
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:CS:SOURce	ロジック SPI バストリガのチップセレクトトレースを設定 / 問い合わせします。	7-100
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:DATA<x>?	ロジック SPI バストリガの各データに関するすべての設定値を問い合わせます。	7-100
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:DATA<x>:BYTE	ロジック SPI バストリガの各データの設定データ数を設定 / 問い合わせします。	7-100
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:DATA<x>:CONDition	ロジック SPI バストリガの各データの判定方法 (一致 / 不一致) を設定 / 問い合わせします。	7-100
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:DATA<x>:DPOSition	ロジック SPI バストリガの各データのパターン比較先頭位置を設定 / 問い合わせします。	7-100
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:DATA<x>:HEXA<x>	ロジック SPI バストリガの各データを HEXA で設定します。	7-100
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:DATA<x>:PATTern<x>	ロジック SPI バストリガの各データを BINARY で設定 / 問い合わせします。	7-101
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:DATA<x>:SOURce	ロジック SPI バストリガの各データのトレースを設定 / 問い合わせします。	7-101
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:MODE	ロジック SPI バストリガの結線方式 (3 線式 /4 線式) を設定 / 問い合わせします。	7-101
:TRIGger:EINTerval:EVENt<x>:SPIBus?	各イベントの SPI バストリガに関するすべての設定値を問い合わせます。	7-101
:TRIGger:EINTerval:EVENt<x>:SPIBus:BITorder	SPI バストリガのビットオーダーを設定 / 問い合わせします。	7-101
:TRIGger:EINTerval:EVENt<x>:SPIBus:CLOCk?	SPI バストリガのクロックに関するすべての設定値を問い合わせます。	7-101

コマンド	機能	ページ
:TRIGger:EINTerval:EVENt<x>:SPIBus:CLOCk:POLarity	SPI バストリガのクロックトレースの極性を設定 / 問い合わせします。	7-101
:TRIGger:EINTerval:EVENt<x>:SPIBus:CLOCk:SOURce	SPI バストリガのクロックトレースを設定 / 問い合わせします。	7-102
:TRIGger:EINTerval:EVENt<x>:SPIBus:CS?	SPI バストリガのチップセレクトに関するすべての設定値を問い合わせます。	7-102
:TRIGger:EINTerval:EVENt<x>:SPIBus:CS:ACTive	SPI バストリガのチップセレクトのアクティブレベルを設定 / 問い合わせします。	7-102
:TRIGger:EINTerval:EVENt<x>:SPIBus:CS:SOURce	SPI バストリガのチップセレクトトレースを設定 / 問い合わせします。	7-102
:TRIGger:EINTerval:EVENt<x>:SPIBus:DATA<x>?	SPI バストリガの各データに関するすべての設定値を問い合わせます。	7-102
:TRIGger:EINTerval:EVENt<x>:SPIBus:DATA<x>:BYTE	SPI バストリガの各データの設定データ数を設定 / 問い合わせします。	7-102
:TRIGger:EINTerval:EVENt<x>:SPIBus:DATA<x>:CONDition	SPI バストリガの各データの判定方法 (一致 / 不一致) を設定 / 問い合わせします。	7-102
:TRIGger:EINTerval:EVENt<x>:SPIBus:DATA<x>:DPOSition	SPI バストリガの各データのパターン比較先頭位置を設定 / 問い合わせします。	7-103
:TRIGger:EINTerval:EVENt<x>:SPIBus:DATA<x>:HEXA<x>	SPI バストリガの各データを HEXA で設定します。	7-103
:TRIGger:EINTerval:EVENt<x>:SPIBus:DATA<x>:PATTern<x>	SPI バストリガの各データを BINARY で設定 / 問い合わせします。	7-103
:TRIGger:EINTerval:EVENt<x>:SPIBus:DATA<x>:SOURce	SPI バストリガの各データのトレースを設定 / 問い合わせします。	7-103
:TRIGger:EINTerval:EVENt<x>:SPIBus:MODE	SPI バストリガの結線方式 (3 線式 /4 線式) を設定 / 問い合わせします。	7-103
:TRIGger:EINTerval:EVENt<x>:STATe:CHANnel<x>	各チャネルの成立条件を設定 / 問い合わせします。	7-103
:TRIGger:EINTerval:EVENt<x>:TYPE	各イベントのトリガの種類を設定 / 問い合わせします。	7-103
:TRIGger:ENHanced:CANBus?	CAN バス信号トリガに関するすべての設定値を問い合わせます。	7-104
:TRIGger:ENHanced:CANBus:ACK	CAN バス信号トリガの ACK 条件を設定 / 問い合わせします。	7-104
:TRIGger:ENHanced:CANBus:BRATe	CAN バス信号トリガのビットレート (データ転送速度) を設定 / 問い合わせします。	7-104
:TRIGger:ENHanced:CANBus:DATA?	CAN バス信号トリガのデータに関するすべての設定値を問い合わせます。	7-104
:TRIGger:ENHanced:CANBus:DATA:BORDer	CAN バス信号トリガのデータのバイトオーダを設定 / 問い合わせします。	7-104
:TRIGger:ENHanced:CANBus:DATA:CONDition	CAN バス信号トリガのデータ条件を設定 / 問い合わせします。	7-104
:TRIGger:ENHanced:CANBus:DATA:DATA<x>	CAN バス信号トリガのデータの比較データを設定 / 問い合わせします。	7-104
:TRIGger:ENHanced:CANBus:DATA:DLC	CAN バス信号トリガのデータの有効バイト数 (DLC) を設定 / 問い合わせします。	7-105
:TRIGger:ENHanced:CANBus:DATA:HEXA	CAN バス信号トリガのデータを HEXA で設定します。	7-105
:TRIGger:ENHanced:CANBus:DATA:MSBLsb	CAN バス信号トリガのデータの MSB/LSB のビットを設定 / 問い合わせします。	7-105
:TRIGger:ENHanced:CANBus:DATA:PATTern	CAN バス信号トリガのデータを BINARY で設定 / 問い合わせします。	7-105
:TRIGger:ENHanced:CANBus:DATA:SIGN	CAN バス信号トリガのデータの符号を設定 / 問い合わせします。	7-105
:TRIGger:ENHanced:CANBus:IDEXt?	CAN バス信号トリガの拡張フォーマットの ID に関するすべての設定値を問い合わせます。	7-105
:TRIGger:ENHanced:CANBus:IDEXt:HEXA	CAN バス信号トリガの拡張フォーマットの ID を HEXA で設定します。	7-105
:TRIGger:ENHanced:CANBus:IDEXt:PATTern	CAN バス信号トリガの拡張フォーマットの ID を BINARY で設定 / 問い合わせします。	7-105
:TRIGger:ENHanced:CANBus:IDOR?	CAN バス信号トリガの OR 条件に関するすべての設定値を問い合わせます。	7-105
:TRIGger:ENHanced:CANBus:IDOR:ID<x>?	CAN バス信号トリガの OR 条件の各 ID に関するすべての設定値を問い合わせます。	7-106
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:ACK	CAN バス信号トリガの OR 条件の各 ACK 条件を設定 / 問い合わせします。	7-106
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA?	CAN バス信号トリガの OR 条件の各データに関するすべての設定値を問い合わせます。	7-106

コマンド	機能	ページ
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:BORDer	CAN バス信号トリガの OR 条件の各データのバイトオーダを設定 / 問い合わせします。	7-106
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:CONDition	CAN バス信号トリガの OR 条件の各データ条件を設定 / 問い合わせします。	7-106
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:DATA<x>	CAN バス信号トリガの OR 条件の各データの比較データを設定 / 問い合わせします。	7-106
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:DLC	CAN バス信号トリガの OR 条件の各データの有効バイト数 (DLC) を設定 / 問い合わせします。	7-107
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:HEXA	CAN バス信号トリガの OR 条件の各データを HEXA で設定します。	7-107
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:MSBLsb	CAN バス信号トリガの OR 条件の各データの MSB/LSB のビットを設定 / 問い合わせします。	7-107
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:PATTern	CAN バス信号トリガの OR 条件の各データを BINARY で設定 / 問い合わせします。	7-107
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:SIGN	CAN バス信号トリガの OR 条件の各データの符号を設定 / 問い合わせします。	7-107
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:FORMat	CAN バス信号トリガの OR 条件の各メッセージフォーマット (標準 / 拡張) を設定 / 問い合わせします。	7-107
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:IDEXt?	CAN バス信号トリガの OR 条件の各拡張フォーマットの ID に関するすべての設定値を問い合わせます。	7-108
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:IDEXt:HEXA	CAN バス信号トリガの OR 条件の各拡張フォーマットの ID を HEXA で設定します。	7-108
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:IDEXt:PATTern	CAN バス信号トリガの OR 条件の各拡張フォーマットの ID を BINARY で設定 / 問い合わせします。	7-108
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:IDSTd?	CAN バス信号トリガの OR 条件の各標準フォーマットの ID に関するすべての設定値を問い合わせます。	7-108
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:IDSTd:HEXA	CAN バス信号トリガの OR 条件の各標準フォーマットの ID を HEXA で設定します。	7-108
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:IDSTd:PATTern	CAN バス信号トリガの OR 条件の各標準フォーマットの ID を BINARY で設定 / 問い合わせします。	7-108
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:MODE	CAN バス信号トリガの OR 条件の各条件の有効 (1)/ 無効 (0) を設定 / 問い合わせします。	7-109
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:RTR	CAN バス信号トリガの OR 条件の各 RTR を設定 / 問い合わせします。	7-109
:TRIGger:ENHanced:CANBus:IDSTd?	CAN バス信号トリガの標準フォーマットの ID に関するすべての設定値を問い合わせます。	7-109
:TRIGger:ENHanced:CANBus:IDSTd:HEXA	CAN バス信号トリガの標準フォーマットの ID を HEXA で設定します。	7-109
:TRIGger:ENHanced:CANBus:IDSTd:PATTern	CAN バス信号トリガの標準フォーマットの ID を BINARY で設定 / 問い合わせします。	7-109
:TRIGger:ENHanced:CANBus:MODE	CAN バス信号トリガのモードを設定 / 問い合わせします。	7-109
:TRIGger:ENHanced:CANBus:RECessive	CAN バス信号トリガのリセッシブレベル (バスレベル) を設定 / 問い合わせします。	7-109
:TRIGger:ENHanced:CANBus:RTR	CAN バス信号トリガの RTR を設定 / 問い合わせします。	7-109
:TRIGger:ENHanced:CANBus:SOURce	CAN バス信号トリガのトリガソースを設定 / 問い合わせします。	7-110
:TRIGger:ENHanced:CANBus:SPOint	CAN バス信号トリガのサンプルポイントを設定 / 問い合わせします。	7-110
:TRIGger:ENHanced:I2CBus?	I^2C バストリガに関するすべての設定値を問い合わせます。	7-110
:TRIGger:ENHanced:I2CBus:ADATa?	I^2C バストリガのアドレスに関するすべての設定値を問い合わせます。	7-110
:TRIGger:ENHanced:I2CBus:ADATa:BIT10address?	I^2C バストリガの 10bit アドレスに関するすべての設定値を問い合わせます。	7-110
:TRIGger:ENHanced:I2CBus:ADATa:BIT10address:HEXA	I^2C バストリガの 10bit アドレスを HEXA で設定します。	7-110
:TRIGger:ENHanced:I2CBus:ADATa:BIT10address:PATTern	I^2C バストリガの 10bit アドレスを BINARY で設定 / 問い合わせします。	7-110
:TRIGger:ENHanced:I2CBus:ADATa:BIT7ADdress?	I^2C バストリガの 7bit アドレスに関するすべての設定値を問い合わせます。	7-110
:TRIGger:ENHanced:I2CBus:ADATa:BIT7ADdress:HEXA	I^2C バストリガの 7bit アドレスを HEXA で設定します。	7-110
:TRIGger:ENHanced:I2CBus:ADATa:BIT7ADdress:PATTern	I^2C バストリガの 7bit アドレスを BINARY で設定 / 問い合わせします。	7-110

コマンド	機能	ページ
:TRIGger:ENHanced:I2CBus:ADATa:BIT7APsub?	I²C バストリガの 7bit + Sub アドレスに関するすべての設定値を問い合わせます。	7-111
:TRIGger:ENHanced:I2CBus:ADATa:BIT7APsub:ADDRess?	I²C バストリガの 7bit + Sub アドレスの 7bit アドレスに関するすべての設定値を問い合わせます。	7-111
:TRIGger:ENHanced:I2CBus:ADATa:BIT7APsub:ADDRess:HEXA	I²C バストリガの 7bit + Sub アドレスの 7bit アドレスを HEXA で設定します。	7-111
:TRIGger:ENHanced:I2CBus:ADATa:BIT7APsub:ADDRess:PATTern	I²C バストリガの 7bit + Sub アドレスの 7bit アドレスを BINARY で設定 / 問い合わせします。	7-111
:TRIGger:ENHanced:I2CBus:ADATa:BIT7APsub:SADDress?	I²C バストリガの 7bit + Sub アドレスの Sub アドレスに関するすべての設定値を問い合わせます。	7-111
:TRIGger:ENHanced:I2CBus:ADATa:BIT7APsub:SADDress:HEXA	I²C バストリガの 7bit + Sub アドレスの Sub アドレスを HEXA で設定します。	7-111
:TRIGger:ENHanced:I2CBus:ADATa:BIT7APsub:SADDress:PATTern	I²C バストリガの 7bit + Sub アドレスの Sub アドレスを BINARY で設定 / 問い合わせします。	7-111
:TRIGger:ENHanced:I2CBus:ADATa:TYPE	I²C バストリガのアドレスの種類を設定 / 問い合わせします。	7-111
:TRIGger:ENHanced:I2CBus:CLOCk?	I²C バストリガのクロックに関するすべての設定値を問い合わせます。	7-111
:TRIGger:ENHanced:I2CBus:CLOCk:SOURce	I²C バストリガのクロックトレースを設定 / 問い合わせします。	7-112
:TRIGger:ENHanced:I2CBus:DATA?	I²C バストリガのデータに関するすべての設定値を問い合わせます。	7-112
:TRIGger:ENHanced:I2CBus:DATA:BYTE	I²C バストリガの設定データ数を設定 / 問い合わせします。	7-112
:TRIGger:ENHanced:I2CBus:DATA:CONDition	I²C バストリガのデータの判定方法 (一致 / 不一致) を設定 / 問い合わせします。	7-112
:TRIGger:ENHanced:I2CBus:DATA:DPOSition	I²C バストリガのデータのパターン比較する位置を設定 / 問い合わせします。	7-112
:TRIGger:ENHanced:I2CBus:DATA:HEXA<x>	I²C バストリガのデータを HEXA で設定します。	7-112
:TRIGger:ENHanced:I2CBus:DATA:MODE	I²C バストリガのデータ条件の有効 / 無効を設定 / 問い合わせします。	7-112
:TRIGger:ENHanced:I2CBus:DATA:PATTern<x>	I²C バストリガのデータを BINARY で設定 / 問い合わせします。	7-112
:TRIGger:ENHanced:I2CBus:DATA:PMODe	I²C バストリガのデータのパターン比較先頭位置モードを設定 / 問い合わせします。	7-113
:TRIGger:ENHanced:I2CBus:DATA:SOURce	I²C バストリガのデータトレースを設定 / 問い合わせします。	7-113
:TRIGger:ENHanced:I2CBus:GCALl?	I²C バストリガのジェネラルコールに関するすべての設定値を問い合わせます。	7-113
:TRIGger:ENHanced:I2CBus:GCALl:BIT7maddress?	I²C バストリガのジェネラルコールの 7bit マスタアドレスに関するすべての設定値を問い合わせます。	7-113
:TRIGger:ENHanced:I2CBus:GCALl:BIT7maddress:HEXA	I²C バストリガのジェネラルコールの 7bit マスタアドレスを HEXA で設定します。	7-113
:TRIGger:ENHanced:I2CBus:GCALl:BIT7maddress:PATTern	I²C バストリガのジェネラルコールの 7bit マスタアドレスを BINARY で設定 / 問い合わせします。	7-113
:TRIGger:ENHanced:I2CBus:GCALl:SBYTe (Second Byte)	I²C バストリガのジェネラルコールのセカンドバイトのタイプを設定 / 問い合わせします。	7-113
:TRIGger:ENHanced:I2CBus:MODE	I²C バストリガのトリガモードを設定 / 問い合わせします。	7-113
:TRIGger:ENHanced:I2CBus:NAIGnore?	I²C バストリガの NON ACK 無視モードに関するすべての設定値を問い合わせます。	7-113
:TRIGger:ENHanced:I2CBus:NAIGnore:HSMode	I²C バストリガのハイスピードモードで NON ACK を無視する / しないを設定 / 問い合わせします。	7-114
:TRIGger:ENHanced:I2CBus:NAIGnore:RACCess	I²C バストリガのリードアクセスモードで NON ACK を無視する / しないを設定 / 問い合わせします。	7-114
:TRIGger:ENHanced:I2CBus:NAIGnore:SBYTe (Start Byte)	I²C バストリガのスタートバイトで NON ACK を無視する / しないを設定 / 問い合わせします。	7-114
:TRIGger:ENHanced:I2CBus:SBHSmode?	I²C バストリガのスタートバイト / ハイスピードモードに関するすべての設定値を問い合わせます。	7-114
:TRIGger:ENHanced:I2CBus:SBHSmode:TYPE	I²C バストリガのスタートバイト / ハイスピードモードのタイプを設定 / 問い合わせします。	7-114
:TRIGger:ENHanced:LINBus?	LIN バストリガに関するすべての設定値を問い合わせします。	7-114
:TRIGger:ENHanced:LINBus:BRATe	LIN バス信号トリガのビットレート (データ転送速度) を設定 / 問い合わせします。	7-114
:TRIGger:ENHanced:LINBus:SOURce	LIN バス信号トリガのトリガソースを設定 / 問い合わせします。	7-114
:TRIGger:ENHanced:SPIBus?	SPI バストリガに関するすべての設定値を問い合わせます。	7-114
:TRIGger:ENHanced:SPIBus:BITorder	SPI バストリガのビットオーダーを設定 / 問い合わせします。	7-115
:TRIGger:ENHanced:SPIBus:CLOCk?	SPI バストリガのクロックに関するすべての設定値を問い合わせます。	7-115

コマンド	機能	ページ
:TRIGger:ENHanced:SPIBus:CLOCk:POLarity	SPI バストリガのクロックトレースの極性を設定 / 問い合わせします。	7-115
:TRIGger:ENHanced:SPIBus:CLOCk:SOURce	SPI バストリガのクロックトレースを設定 / 問い合わせします。	7-115
:TRIGger:ENHanced:SPIBus:CS?	SPI バストリガのチップセレクトに関するすべての設定値を問い合わせます。	7-115
:TRIGger:ENHanced:SPIBus:CS:ACTive	SPI バストリガのチップセレクトのアクティブレベルを設定 / 問い合わせします。	7-115
:TRIGger:ENHanced:SPIBus:CS:SOURce	SPI バストリガのチップセレクトトレースを設定 / 問い合わせします。	7-115
:TRIGger:ENHanced:SPIBus:DATA<x>?	SPI バストリガの各データに関するすべての設定値を問い合わせます。	7-115
:TRIGger:ENHanced:SPIBus:DATA<x>:BYTE	SPI バストリガの各データの設定データ数を設定 / 問い合わせします。	7-115
:TRIGger:ENHanced:SPIBus:DATA<x>:CONDition	SPI バストリガの各データの判定方法 (一致 / 不一致) を設定 / 問い合わせします。	7-116
:TRIGger:ENHanced:SPIBus:DATA<x>:DPOSition	SPI バストリガの各データのパターン比較先頭位置を設定 / 問い合わせします。	7-116
:TRIGger:ENHanced:SPIBus:DATA<x>:HEXA<x>	SPI バストリガの各データを HEXA で設定します。	7-116
:TRIGger:ENHanced:SPIBus:DATA<x>:PATTern<x>	SPI バストリガの各データを BINARY で設定 / 問い合わせします。	7-116
:TRIGger:ENHanced:SPIBus:DATA<x>:SOURce	SPI バストリガの各データのトレースを設定 / 問い合わせします。	7-116
:TRIGger:ENHanced:SPIBus::MODE	SPI バストリガの結線方式 (3 線式 /4 線式) を設定 / 問い合わせします。	7-116
:TRIGger:ENHanced:UART?	UART 信号トリガに関するすべて設定値を問い合わせします。	7-116
:TRIGger:ENHanced:UART:BRATe	UART 信号トリガのビットレート (データ転送速度) を設定 / 問い合わせします。	7-117
:TRIGger:ENHanced:UART:FORMat	UART 信号トリガのフォーマットを設定 / 問い合わせします。	7-117
:TRIGger:ENHanced:UART:POLarity	UART 信号トリガの極性を設定 / 問い合わせします。	7-117
:TRIGger:ENHanced:UART:SOURce	UART 信号トリガのトリガソースを設定 / 問い合わせします。	7-117
:TRIGger:ENHanced:UART:SPOint	UART 信号トリガのサンプルポイントを設定 / 問い合わせします。	7-117
:TRIGger:LOGic:I2CBus?	ロジック I^2C バストリガに関するすべての設定値を問い合わせます。	7-117
:TRIGger:LOGic:I2CBus:ADATa?	ロジック I^2C バストリガのアドレスに関するすべての設定値を問い合わせます。	7-117
:TRIGger:LOGic:I2CBus:ADATa:BIT10address?	ロジック I^2C バストリガの 10bit アドレスに関するすべての設定値を問い合わせます。	7-117
:TRIGger:LOGic:I2CBus:ADATa:BIT10address:HEXA	ロジック I^2C バストリガの 10bit アドレスを HEXA で設定します。	7-117
:TRIGger:LOGic:I2CBus:ADATa:BIT10address:PATTern	ロジック I^2C バストリガの 10bit アドレスを BINARY で設定 / 問い合わせします。	7-117
:TRIGger:LOGic:I2CBus:ADATa:BIT7ADdress?	ロジック I^2C バストリガの 7bit アドレスに関するすべての設定値を問い合わせます。	7-118
:TRIGger:LOGic:I2CBus:ADATa:BIT7ADdress:HEXA	ロジック I^2C バストリガの 7bit アドレスを HEXA で設定します。	7-118
:TRIGger:LOGic:I2CBus:ADATa:BIT7ADdress:PATTern	ロジック I^2C バストリガの 7bit アドレスを BINARY で設定 / 問い合わせします。	7-118
:TRIGger:LOGic:I2CBus:ADATa:BIT7APsub?	ロジック I^2C バストリガの 7bit + Sub アドレスに関するすべての設定値を問い合わせます。	7-118
:TRIGger:LOGic:I2CBus:ADATa:BIT7APsub:ADDRess?	ロジック I^2C バストリガの 7bit + Sub アドレスの 7bit アドレスに関するすべての設定値を問い合わせます。	7-118
:TRIGger:LOGic:I2CBus:ADATa:BIT7APsub:ADDRess:HEXA	ロジック I^2C バストリガの 7bit + Sub アドレスの 7bit アドレスを HEXA で設定します。	7-118
:TRIGger:LOGic:I2CBus:ADATa:BIT7APsub:ADDRess:PATTern	ロジック I^2C バストリガの 7bit + Sub アドレスの 7bit アドレスを BINARY で設定 / 問い合わせします。	7-118
:TRIGger:LOGic:I2CBus:ADATa:BIT7APsub:SADDress?	ロジック I^2C バストリガの 7bit + Sub アドレスの Sub アドレスに関するすべての設定値を問い合わせます。	7-118
:TRIGger:LOGic:I2CBus:ADATa:BIT7APsub:SADDress:HEXA	ロジック I^2C バストリガの 7bit + Sub アドレスの Sub アドレスを HEXA で設定します。	7-118
:TRIGger:LOGic:I2CBus:ADATa:BIT7APsub:SADDress:PATTern	ロジック I^2C バストリガの 7bit + Sub アドレスの Sub アドレスを BINARY で設定 / 問い合わせします。	7-119

コマンド	機能	ページ
:TRIGger:LOGic:I2CBus:ADATa:TYPE	ロジック I²C バストリガのアドレスの種類を設定 / 問い合わせします。	7-119
:TRIGger:LOGic:I2CBus:CLOCk?	ロジック I2C バストリガのクロックに関するすべての設定値を問い合わせます。	7-119
:TRIGger:LOGic:I2CBus:CLOCk:SOURce	ロジック I²C バストリガのクロックトレースを設定 / 問い合わせします。	7-119
:TRIGger:LOGic:I2CBus:DATA?	ロジック I²C バストリガのデータに関するすべての設定値を問い合わせます。	7-119
:TRIGger:LOGic:I2CBus:DATA:BYTE	ロジック I²C バストリガの設定データ数を設定 / 問い合わせします。	7-119
:TRIGger:LOGic:I2CBus:DATA:CONDition	ロジック I²C バストリガのデータの判定方法 (一致 / 不一致) を設定 / 問い合わせします。	7-119
:TRIGger:LOGic:I2CBus:DATA:DPOSition	ロジック I²C バストリガのデータのパターン比較する位置を設定 / 問い合わせします。	7-119
:TRIGger:LOGic:I2CBus:DATA:HEXA<x>	ロジック I²C バストリガのデータを HEXA で設定します。	7-119
:TRIGger:LOGic:I2CBus:DATA:MODE	ロジック I²C バストリガのデータ条件の有効 / 無効を設定 / 問い合わせします。	7-120
:TRIGger:LOGic:I2CBus:DATA:PATTern<x>	ロジック I²C バストリガのデータを BINARY で設定 / 問い合わせします。	7-120
:TRIGger:LOGic:I2CBus:DATA:PMODe	ロジック I²C バストリガのデータのパターン比較先頭位置モードを設定 / 問い合わせします。	7-120
:TRIGger:LOGic:I2CBus:DATA:SOURce	ロジック I²C バストリガのデータトレースを設定 / 問い合わせします。	7-120
:TRIGger:LOGic:I2CBus:GCALl?	ロジック I²C バストリガのジェネラルコールに関するすべての設定値を問い合わせます。	7-120
:TRIGger:LOGic:I2CBus:GCALl:BIT7maddress?	ロジック I²C バストリガのジェネラルコールの 7bit マスタアドレスに関するすべての設定値を問い合わせます。	7-120
:TRIGger:LOGic:I2CBus:GCALl:BIT7maddress:HEXA	ロジック I²C バストリガのジェネラルコールの 7bit マスタアドレスを HEXA で設定します。	7-120
:TRIGger:LOGic:I2CBus:GCALl:BIT7maddress:PATTern	ロジック I²C バストリガのジェネラルコールの 7bit マスタアドレスを BINARY で設定 / 問い合わせします。	7-120
:TRIGger:LOGic:I2CBus:GCALl:SBYTe (Second Byte)	ロジック I²C バストリガのジェネラルコールのセカンドバイトのタイプを設定 / 問い合わせします。	7-120
:TRIGger:LOGic:I2CBus:MODE	ロジック I²C バストリガのトリガモードを設定 / 問い合わせします。	7-121
:TRIGger:LOGic:I2CBus:NAIGnore?	ロジック I²C バストリガの NON ACK 無視モードに関するすべての設定値を問い合わせます。	7-121
:TRIGger:LOGic:I2CBus:NAIGnore:HSMode	ロジック I²C バストリガのハイスピードモードで NON ACK を無視する / しないを設定 / 問い合わせします。	7-121
:TRIGger:LOGic:I2CBus:NAIGnore:RACCess	ロジック I²C バストリガのリードアクセスモードで NON ACK を無視する / しないを設定 / 問い合わせします。	7-121
:TRIGger:LOGic:I2CBus:NAIGnore:SBYTe (Start Byte)	ロジック I²C バストリガのスタートバイトで NON ACK を無視する / しないを設定 / 問い合わせします。	7-121
:TRIGger:LOGic:I2CBus:SBHSmode?	ロジック I²C バストリガのスタートバイト / ハイスピードモードに関するすべての設定値を問い合わせます。	7-121
:TRIGger:LOGic:I2CBus:SBHSmode:TYPE	ロジック I²C バストリガのスタートバイト / ハイスピードモードのタイプを設定 / 問い合わせします。	7-121
:TRIGger:LOGic:LINBus?	ロジック LIN バス信号トリガに関するすべての設定値を問い合わせします。	7-121
:TRIGger:LOGic:LINBus:BRATe	ロジック LIN バス信号トリガのビットレート (データ転送速度) を設定 / 問い合わせします。	7-121
:TRIGger:LOGic:LINBus:SOURce	ロジック LIN バス信号トリガのトリガソースを設定 / 問い合わせします。	7-122
:TRIGger:LOGic:SPIBus?	ロジック SPI バストリガに関するすべての設定値を問い合わせます。	7-122
:TRIGger:LOGic:SPIBus:BITorder	ロジック SPI バストリガのビットオーダーを設定 / 問い合わせします。	7-122
:TRIGger:LOGic:SPIBus:CLOCk?	ロジック SPI バストリガのクロックに関するすべての設定値を問い合わせます。	7-122
:TRIGger:LOGic:SPIBus:CLOCk:POLarity	ロジック SPI バストリガのクロックトレースの極性を設定 / 問い合わせします。	7-122
:TRIGger:LOGic:SPIBus:CLOCk:SOURce	ロジック SPI バストリガのクロックトレースを設定 / 問い合わせします。	7-122

コマンド	機能	ページ
:TRIGger:LOGic:SPIBus:CS?	ロジック SPI バストリガのチップセレクトに関するすべての設定値を問い合わせます。	7-122
:TRIGger:LOGic:SPIBus:CS:ACTive	ロジック SPI バストリガのチップセレクトのアクティブレベルを設定 / 問い合わせします。	7-122
:TRIGger:LOGic:SPIBus:CS:SOURce	ロジック SPI バストリガのチップセレクトトレースを設定 / 問い合わせします。	7-122
:TRIGger:LOGic:SPIBus:DATA<x>?	ロジック SPI バストリガの各データに関するすべての設定値を問い合わせます。	7-122
:TRIGger:LOGic:SPIBus:DATA<x>:BYTE	ロジック SPI バストリガの各データの設定データ数を設定 / 問い合わせします。	7-123
:TRIGger:LOGic:SPIBus:DATA<x>:CONDition	ロジック SPI バストリガの各データの判定方法 (一致 / 不一致) を設定 / 問い合わせします。	7-123
:TRIGger:LOGic:SPIBus:DATA<x>:DPOSition	ロジック SPI バストリガの各データのパターン比較先頭位置を設定 / 問い合わせします。	7-123
:TRIGger:LOGic:SPIBus:DATA<x>:HEXA<x>	ロジック SPI バストリガの各データを HEXA で設定します。	7-123
:TRIGger:LOGic:SPIBus:DATA<x>:PATTern<x>	ロジック SPI バストリガの各データを BINARY で設定 / 問い合わせします。	7-123
:TRIGger:LOGic:SPIBus:DATA<x>:SOURce	ロジック SPI バストリガの各データのトレースを設定 / 問い合わせします。	7-123
:TRIGger:LOGic:SPIBus:MODE	ロジック SPI バストリガの結線方式 (3 線式 /4 線式) を設定 / 問い合わせします。	7-123
:TRIGger:LOGic:UART?	ロジック UART 信号トリガに関するすべて設定値を問い合わせします。	7-123
:TRIGger:LOGic:UART:BRATe	ロジック UART 信号トリガのビットレート (データ転送速度) を設定 / 問い合わせします。	7-124
:TRIGger:LOGic:UART:FORMat	ロジック UART 信号トリガのフォーマットを設定 / 問い合わせします。	7-124
:TRIGger:LOGic:UART:POLarity	ロジック UART 信号トリガの極性を設定 / 問い合わせします。	7-124
:TRIGger:LOGic:UART:SOURce	ロジック UART 信号トリガのトリガソースを設定 / 問い合わせします。	7-124
:TRIGger:LOGic:UART:SPOint	ロジック UART 信号トリガのサンプルポイントを設定 / 問い合わせします。	7-124
:TRIGger:SOURce:CHANnel<x>:LEVel	各チャネルのトリガレベルを設定 / 問い合わせします。	7-124
:TRIGger:SOURce:CHANnel<x>:STATe	各チャネルの成立条件を設定 / 問い合わせします。	7-124
:TRIGger:TYPE	トリガの種類を設定 / 問い合わせします。	7-125

7.2 ANALysis グループ

:ANALysis:LSBus<x>?

機能　ロジックシリアルバス信号解析機能に関するすべての設定値を問い合わせます。

構文　:ANALysis:LSBus<x>?
<x> = 1、2

解説　このコマンドは、DLM6000 に適用できます。

:ANALysis:LSBus<x>[:ANALyze]?

機能　ロジックシリアルバス信号解析に関するすべての設定値を問い合わせます。

構文　:ANALysis:LSBus<x>[:ANALyze]?
<x> = 1、2

:ANALysis:LSBus<x>[:ANALyze]:I2CBus?

機能　ロジック I²C バス信号解析に関するすべての設定値を問い合わせます。

構文　:ANALysis:LSBus<x>[:ANALyze]:I2CBus?
<x> = 1、2

:ANALysis:LSBus<x>[:ANALyze]:I2CBus:CLOCk

機能　ロジック I²C バス信号解析のクロックチャネルを設定 / 問い合わせします。

構文　:ANALysis:LSBus<x>[:ANALyze]:I2CBus:CLOCk {A<y>|B<y>|C<y>|D<y>}
:ANALysis:LSBus<x>[:ANALyze]:I2CBus:CLOCk?
<x> = 1、2
<y> = 0 ～ 7

例　:ANALYSIS:LSBUS1:ANALYZE:I2CBUS:CLOCK A0
:ANALYSIS:LSBUS1:ANALYZE:I2CBUS:CLOCK? -> :ANALYSIS:LSBUS1:ANALYZE:I2CBUS:CLOCK A0

解説　DLM6000 の 16 ビットモデルでは {A<x>|C<x>} が有効です。

:ANALysis:LSBus<x>[:ANALyze]:I2CBus:DTRace

機能　ロジック I²C バス信号解析のデータチャネルを設定 / 問い合わせします。

構文　:ANALysis:LSBus<x>[:ANALyze]:I2CBus:DTRace {A<y>|B<y>|C<y>|D<y>}
:ANALysis:LSBus<x>[:ANALyze]:I2CBus:DTRace?
<x> = 1、2
<y> = 0 ～ 7

例　:ANALYSIS:LSBUS1:ANALYZE:I2CBUS:DTRACE A0
:ANALYSIS:LSBUS1:ANALYZE:I2CBUS:DTRACE? -> :ANALYSIS:LSBUS1:ANALYZE:I2CBUS:DTRACE A0

解説　DLM6000 の 16 ビットモデルでは {A<x>|C<x>} が有効です。

:ANALysis:LSBus<x>[:ANALyze]:LINBus?

機能　ロジック LIN バス信号解析に関するすべての設定値を問い合わせます。

構文　:ANALysis:LSBus<x>[:ANALyze]:LINBus?
<x> = 1、2

例　:ANALYSIS:LSBUS1:ANALYZE:LINBUS?
-> :ANALYSIS:LSBUS1:ANALYZE:LINBUS:BRATE 19200;REVISION LIN1_3;SPOINT 18.8E+00;TRACE A0

:ANALysis:LSBus<x>[:ANALyze]:LINBus:BRATe

機能　ロジック LIN バス信号解析のビットレート (データ転送速度) を設定 / 問い合わせします。

構文　:ANALysis:LSBus<x>[:ANALyze]:LINBus:BRATe {<NRf>|USER,<NRf>}
:ANALysis:LSBus<x>[:ANALyze]:LINBus:BRATe?
<x> = 1、2
<NRf> = 1200、2400、4800、9600、19200
USER の <NRf> = 4.4 節参照。

例　:ANALYSIS:LSBUS1:ANALYZE:LINBUS:BRATE 19200
:ANALYSIS:LSBUS1:ANALYZE:LINBUS:BRATE? -> :ANALYSIS:LSBUS1:ANALYZE:LINBUS:BRATE 19200

:ANALysis:LSBus<x>[:ANALyze]:LINBus:FJUMp:BREak

機能　ロジック LIN バス信号解析の結果を対象に Break Field へのフィールドジャンプを実行します。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:LINBus:
FJUMp:BREak
```
<x> = 1、2

例
```
:ANALYSIS:LSBUS1:ANALYZE:LINBUS:
FJUMP:BREAK
```

:ANALysis:LSBus<x>[:ANALyze]:LINBus:FJUMp:CSUM

機能　ロジック LIN バス信号解析の結果を対象に Checksum Field へのフィールドジャンプを実行します。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:LINBus:
FJUMp:CSUM
```
<x> = 1、2

例
```
:ANALYSIS:LSBUS1:ANALYZE:LINBUS:
FJUMP:CSUM
```

:ANALysis:LSBus<x>[:ANALyze]:LINBus:FJUMp:DATA

機能　ロジック LIN バス信号解析の結果を対象に Data Field へのフィールドジャンプを実行します。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:LINBus:
FJUMp:DATA
```
<x> = 1、2

例
```
:ANALYSIS:LSBUS1:ANALYZE:LINBUS:
FJUMP:DATA
```

:ANALysis:LSBus<x>[:ANALyze]:LINBus:FJUMp:IDENtifier

機能　ロジック LIN バス信号解析の結果を対象に Identifier Field へのフィールドジャンプを実行します。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:LINBus:
FJUMp:IDENtifier
```
<x> = 1、2

例
```
:ANALYSIS:LSBUS1:ANALYZE:LINBUS:
FJUMP:IDENTIFIER
```

:ANALysis:LSBus<x>[:ANALyze]:LINBus:FJUMp:SYNCh

機能　ロジック LIN バス信号解析の結果を対象に Synch Field へのフィールドジャンプを実行します。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:LINBus:
FJUMp:SYNCh
```
<x> = 1、2

例
```
:ANALYSIS:LSBUS1:ANALYZE:LINBUS:
FJUMP:SYNCH
```

:ANALysis:LSBus<x>[:ANALyze]:LINBus:REVision

機能　ロジック LIN バス信号解析のレビジョン (1.3 or 2.0 or Both) を設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:LINBus:
REVision {BOTH|LIN1_3|LIN2_0}
:ANALysis:LSBus<x>[:ANALyze]:LINBus:
REVision?
```
<x> = 1、2

例
```
:ANALYSIS:LSBUS1:ANALYZE:LINBUS:
REVISION LIN1_3
:ANALYSIS:LSBUS1:ANALYZE:LINBUS:
REVISION? -> :ANALYSIS:LSBUS1:
ANALYZE:LINBUS:REVISION LIN1_3
```

:ANALysis:LSBus<x>[:ANALyze]:LINBus:SPOint

機能　ロジック LIN バス信号解析のサンプルポイントを設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:LINBus:
SPOint {<NRf>}
:ANALysis:LSBus<x>[:ANALyze]:LINBus:
SPOint?
```
<x> = 1、2
<NRf> = 18.8 ～ 90.6(%)

例
```
:ANALYSIS:LSBUS1:ANALYZE:LINBUS:
SPOINT 18.8
:ANALYSIS:LSBUS1:ANALYZE:LINBUS:
SPOINT?
-> :ANALYSIS:LSBUS1:ANALYZE:LINBUS:
SPOINT 18.8E+00
```

:ANALysis:LSBus<x>[:ANALyze]:LINBus:TRACe

機能　ロジック LIN バス信号解析のトレースを設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:LINBus:
TRACe {A<y>|B<y>|C<y>|D<y>}
:ANALysis:LSBus<x>[:ANALyze]:LINBus:
TRACe?
```
<x> = 1、2
<y> = 0 ～ 7

例
```
:ANALYSIS:LSBUS1:ANALYZE:LINBUS:
TRACE A0
:ANALYSIS:LSBUS1:ANALYZE:LINBUS:
TRACE? -> :ANALYSIS:LSBUS1:ANALYZE:
LINBUS:TRACE A0
```

解説　DLM6000 の 16 ビットモデルでは {A<x>|C<x>} が有効です。

:ANALysis:LSBus<x>[:ANALyze]:LIST?

機能　ロジックシリアルバス信号解析の解析結果リストに関するすべての設定値を問い合わせます。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:LIST?
<x> = 1、2
```

:ANALysis:LSBus<x>[:ANALyze]:LIST:DISPlay

機能　ロジックシリアルバス信号解析の解析結果リストの ON/OFF を設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:LIST:
DISPlay {<Boolean>}
:ANALysis:LSBus<x>[:ANALyze]:LIST:
DISPlay?
<x> = 1、2
```

例
```
:ANALYSIS:LSBUS1:ANALYZE:LIST:
DISPLAY ON
:ANALYSIS:LSBUS1:ANALYZE:LIST:
DISPLAY? -> :ANALYSIS:LSBUS1:
ANALYZE:
LIST:DISPLAY 1
```

:ANALysis:LSBus<x>[:ANALyze]:LIST:ITEM?

機能　ロジックシリアルバス信号解析の解析結果リストに表示される項目を問い合わせます。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:LIST:
ITEM?
<x> = 1、2
```

例
```
:ANALYSIS:LSBUS1:ANALYZE:LIST:ITEM?
-> :ANALYSIS:LSBUS1:ANALYZE:LIST:
ITEM "No.,S/P,Hex,Form,R/W,ACK,"
```

:ANALysis:LSBus<x>[:ANALyze]:LIST:MODE

機能　ロジックシリアルバス信号解析の解析結果リストのモードを設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:LIST:
MODE {DETail|SIMPle}
:ANALysis:LSBus<x>[:ANALyze]:LIST:
MODE?
<x> = 1、2
```

例
```
:ANALYSIS:LSBUS1:ANALYZE:LIST:
MODE DETAIL
:ANALYSIS:LSBUS1:ANALYZE:LIST:
MODE? -> :ANALYSIS:LSBUS1:ANALYZE:
LIST:MODE DETAIL
```

:ANALysis:LSBus<x>[:ANALyze]:LIST:SCRoll

機能　ロジックシリアルバス信号解析の解析結果リストのスクロール方法を設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:LIST:
SCRoll {HORizontal|VERTical}
:ANALysis:LSBus<x>[:ANALyze]:LIST:
SCRoll?
<x> = 1、2
```

例
```
:ANALYSIS:LSBUS1:ANALYZE:LIST:
SCROLL HORIZONTAL
:ANALYSIS:LSBUS1:ANALYZE:LIST:
SCROLL? -> :ANALYSIS:LSBUS1:ANALYZE:
LIST:SCROLL HORIZONTAL
```

:ANALysis:LSBus<x>[:ANALyze]:LIST:VALue?

機能　ロジックシリアルバス信号解析の解析結果リストの指定した解析番号の自動測定値を問い合わせます。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:LIST:
VALue? {<NRf>|MAXimum|MINimum}
<x> = 1、2
<NRf> = -40000 ～ 40000
(「:ANALysis:SBUS<x>[:ANALyze]:MODE
CANBus」のときは、<NRf> = -2999 ～ 2999)
```

例
```
:ANALYSIS:LSBUS1:ANALYZE:LIST:
VALUE? 1
-> :ANALYSIS:LSBUS1:ANALYZE:LIST:
VALUE "1, P, 00, A, , 0,"
```

解説　データが「MAXimum」の場合は最大リスト表示番号、「MINimum」の場合は最小リスト表示番号が指定になります。

:ANALysis:LSBus<x>[:ANALyze]:MODE

機能　ロジックシリアルバス信号解析のモードを設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:
MODE {I2CBus|LINBus|SPIBus|UART}
:ANALysis:LSBus<x>[:ANALyze]:MODE?
<x> = 1、2
```

例
```
:ANALYSIS:LSBUS1:ANALYZE:MODE I2CBUS
:ANALYSIS:LSBUS1:ANALYZE:
MODE? -> :ANALYSIS:LSBUS1:ANALYZE:
MODE I2CBUS
```

:ANALysis:LSBus<x>[:ANALyze]:RPOint

機能　ロジックシリアルバス信号解析のリファレンスポイント (解析基準点) を設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:
RPOint {<NRf>,MANual|TRIGger}
:ANALysis:LSBus<x>[:ANALyze]:RPOint?
```
<x> = 1、2
<NRf> = −5 ～ 5 (div)

例
```
:ANALYSIS:LSBUS1:ANALYZE:RPOINT
MANUAL,1
:ANALYSIS:LSBUS1:ANALYZE:
RPOINT? -> :ANALYSIS:LSBUS1:ANALYZE:
RPOINT MANUAL, 1.00000E+00
```

:ANALysis:LSBus<x>[:ANALyze]:SPIBus?

機能　ロジック SPI バス信号解析に関するすべての設定値を問い合わせます。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:SPIBus?
```
<x> = 1、2

:ANALysis:LSBus<x>[:ANALyze]:SPIBus:CLOCk?

機能　ロジック SPI バス信号解析のクロック信号のチャネルに関するすべての設定値を問い合わせます。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:SPIBus:
CLOCk?
```
<x> = 1、2

:ANALysis:LSBus<x>[:ANALyze]:SPIBus:CLOCk:POLarity

機能　ロジック SPI バス信号解析のクロック信号のチャネルの極性を設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:SPIBus:
CLOCk:POLarity {FALL|RISE}
:ANALysis:LSBus<x>[:ANALyze]:SPIBus:
CLOCk:POLarity?
```
<x> = 1、2

例
```
:ANALYSIS:LSBUS1:ANALYZE:SPIBUS:
CLOCK:POLARITY FALL
:ANALYSIS:LSBUS1:ANALYZE:SPIBUS:
CLOCK:POLARITY? -> :ANALYSIS:LSBUS1:
ANALYZE:SPIBUS:CLOCK:POLARITY FALL
```

:ANALysis:LSBus<x>[:ANALyze]:SPIBus:CLOCk:SOURce

機能　ロジック SPI バス信号解析のクロック信号のチャネルを設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:SPIBus:
CLOCk:SOURce {A<y>|B<y>|C<y>|D<y>}
:ANALysis:LSBus<x>[:ANALyze]:SPIBus:
CLOCk:SOURce?
```
<x> = 1、2
<y> = 0 ～ 7

例
```
:ANALYSIS:LSBUS1:ANALYZE:SPIBUS:
CLOCK:SOURCE A0
:ANALYSIS:LSBUS1:ANALYZE:SPIBUS:
CLOCK:SOURCE? -> :ANALYSIS:LSBUS1:
ANALYZE:SPIBUS:CLOCK:SOURCE A0
```

解説　DLM6000 の 16 ビットモデルでは {A<x>|C<x>} が有効です。

:ANALysis:LSBus<x>[:ANALyze]:SPIBus:CS?

機能　ロジック SPI バス信号解析のチップセレクト信号のチャネルに関するすべての設定値を問い合わせます。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:SPIBus:
CS?
```
<x> = 1、2

:ANALysis:LSBus<x>[:ANALyze]:SPIBus:CS:ACTive

機能　ロジック SPI バス信号解析のチップセレクト信号のチャネルのアクティブレベルを設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:SPIBus:CS:ACTive {HIGH|LOW}
:ANALysis:LSBus<x>[:ANALyze]:SPIBus:CS:ACTive?
```
<x> = 1、2

例
```
:ANALYSIS:LSBUS1:ANALYZE:SPIBUS:CS:ACTIVE HIGH
:ANALYSIS:LSBUS1:ANALYZE:SPIBUS:CS:ACTIVE? -> :ANALYSIS:LSBUS1:ANALYZE:SPIBUS:CS:ACTIVE HIGH
```

:ANALysis:LSBus<x>[:ANALyze]:SPIBus:CS:TRACe

機能　ロジック SPI バス信号解析のチップセレクト信号のチャネルを設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:SPIBus:CS:TRACe {A<y>|B<y>|C<y>|D<y>|NONE}
:ANALysis:LSBus<x>[:ANALyze]:SPIBus:CS:TRACe?
```
<x> = 1、2
<y> = 0 ～ 7

例
```
:ANALYSIS:LSBUS1:ANALYZE:SPIBUS:CS:TRACE A0
:ANALYSIS:LSBUS1:ANALYZE:SPIBUS:CS:TRACE? -> :ANALYSIS:LSBUS1:ANALYZE:SPIBUS:CS:TRACE A0
```

解説　DLM6000 の 16 ビットモデルでは {A<x>|C<x>} が有効です。

:ANALysis:LSBus<x>[:ANALyze]:SPIBus:DATA<x>?

機能　ロジック SPI バス信号解析の各データに関するすべての設定値を問い合わせます。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:SPIBus:DATA<x>?
```
LSBus<x> の <x> = 1、2
DATA<x> の <x> = 1、2

:ANALysis:LSBus<x>[:ANALyze]:SPIBus:DATA<x>:ACTive

機能　ロジック SPI バス信号解析の各データのアクティブレベルを設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:SPIBus:DATA<x>:ACTive {HIGH|LOW}
:ANALysis:LSBus<x>[:ANALyze]:SPIBus:DATA<x>:ACTive?
```
LSBus<x> の <x> = 1、2
DATA<x> の <x> = 1、2

例
```
:ANALYSIS:LSBUS1:ANALYZE:SPIBUS:DATA1:ACTIVE HIGH
:ANALYSIS:LSBUS1:ANALYZE:SPIBUS:DATA1:ACTIVE? -> :ANALYSIS:LSBUS1:ANALYZE:SPIBUS:DATA1:ACTIVE HIGH
```

:ANALysis:LSBus<x>[:ANALyze]:SPIBus:DATA<x>:TRACe

機能　ロジック SPI バス信号解析の各データチャネルを設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:SPIBus:DATA<x>:TRACe {A<y>|B<y>|C<y>|D<y>}
:ANALysis:LSBus<x>[:ANALyze]:SPIBus:DATA<x>:TRACe?
```
LSBus<x> の <x> = 1、2
DATA<x> の <x> = 1、2
<y> = 0 ～ 7

例
```
:ANALYSIS:LSBUS1:ANALYZE:SPIBUS:DATA1:TRACE A0
:ANALYSIS:LSBUS1:ANALYZE:SPIBUS:DATA1:TRACE? -> :ANALYSIS:LSBUS1:ANALYZE:SPIBUS:DATA1:TRACE A0
```

解説　DLM6000 の 16 ビットモデルでは {A<x>|C<x>} が有効です。

:ANALysis:LSBus<x>[:ANALyze]:SPIBus[:SETup]?

機能　ロジック SPI バス信号解析のセットアップに関するすべての設定値を問い合わせます。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:SPIBus[:SETup]?
```
<x> = 1、2

:ANALysis:LSBus<x>[:ANALyze]:SPIBus[:SETup]:BITorder

機能　ロジック SPI バス信号解析のビットオーダを設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:
SPIBus[:SETup]:BITorder
{LSBFirst|MSBFirst}
:ANALysis:LSBus<x>[:ANALyze]:
SPIBus[:SETup]:BITorder?
```
<x> = 1、2

例
```
:ANALYSIS:LSBUS1:ANALYZE:SPIBUS:
SETUP:BITORDER LSBFIRST
:ANALYSIS:LSBUS1:ANALYZE:SPIBUS:
SETUP:BITORDER? -> :ANALYSIS:LSBUS1:
ANALYZE:SPIBUS:SETUP:
BITORDER LSBFIRST
```

:ANALysis:LSBus<x>[:ANALyze]:SPIBus[:SETup]:EMSBLSB

機能　ロジック SPI バス信号解析のフィールドの有効範囲を設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:
SPIBus[:SETup]:EMSBLSB {<NRf>,<NRf>}
:ANALysis:LSBus<x>[:ANALyze]:
SPIBus[:SETup]:EMSBLSB?
```
<x> = 1、2
<NRf> = 4.5 節参照。

例
```
:ANALYSIS:LSBUS1:ANALYZE:SPIBUS:
SETUP:EMSBLSB 1,7
:ANALYSIS:LSBUS1:ANALYZE:SPIBUS:
SETUP:EMSBLSB? -> :ANALYSIS:LSBUS1:
ANALYZE:SPIBUS:SETUP:EMSBLSB 1,7
```

:ANALysis:LSBus<x>[:ANALyze]:SPIBus[:SETup]:FSIZe

機能　ロジック SPI バス信号解析のフィールドサイズを設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:
SPIBus[:SETup]:FSIZe {<NRf>}
:ANALysis:LSBus<x>[:ANALyze]:
SPIBus[:SETup]:FSIZe?
```
<x> = 1、2
<NRf> = 4 ～ 32

例
```
:ANALYSIS:LSBUS1:ANALYZE:SPIBUS:
SETUP:FSIZE 4
:ANALYSIS:LSBUS1:ANALYZE:SPIBUS:SETUP:FSIZE?
-> :ANALYSIS:LSBUS1:ANALYZE:SPIBUS:SETUP:
FSIZE 4
```

:ANALysis:LSBus<x>[:ANALyze]:SPIBus[:SETup]:ITIMe

機能　ロジック SPI バス信号解析のアイドル時間を設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:
SPIBus[:SETup]:ITIMe {< 時間
>|DONTcare}
:ANALysis:LSBus<x>[:ANALyze]:
SPIBus[:SETup]:ITIMe?
```
<x> = 1、2
< 時間 > = 10ns ～ 1ms(10ns ステップ)

例
```
:ANALYSIS:LSBUS1:ANALYZE:SPIBUS:
SETUP:ITIME 10NS
:ANALYSIS:LSBUS1:ANALYZE:SPIBUS:
SETUP:ITIME? -> :ANALYSIS:LSBUS1:
ANALYZE:SPIBUS:SETUP:
ITIME 10.0000E-09
```

:ANALysis:LSBus<x>[:ANALyze]:SPIBus[:SETup]:MODE

機能　ロジック SPI バス信号解析の結線方式 (3 線式 /4 線式) を設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:
SPIBus[:SETup]:MODE {WIRe3|WIRe4}
:ANALysis:LSBus<x>[:ANALyze]:
SPIBus[:SETup]:MODE?
```
<x> = 1、2

例
```
:ANALYSIS:LSBUS1:ANALYZE:SPIBUS:
SETUP:MODE WIRE3
:ANALYSIS:LSBUS1:ANALYZE:SPIBUS:
SETUP:MODE? -> :ANALYSIS:LSBUS1:
ANALYZE:SPIBUS:SETUP:MODE WIRE3
```

ANALysis:LSBus<x>[:ANALyze]:UART?

機能　ロジック UART 信号解析に関するすべての設定値を問い合わせます。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:UART?
```
<x> = 1、2

:ANALysis:LSBus<x>[:ANALyze]:UART:BITorder

機能　ロジック UART 信号解析のビットオーダを設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:UART:
BITorder {LSBFirst|MSBFirst}
:ANALysis:LSBus<x>[:ANALyze]:UART:
BITorder?
```
<x> = 1、2

例
```
:ANALYSIS:LSBUS1:ANALYZE:UART:
BITORDER LSBFIRST
:ANALYSIS:LSBUS1:ANALYZE:UART:
BITORDER?
-> :ANALYSIS:LSBUS1:ANALYZE:UART:
BITORDER LSBFIRST
```

:ANALysis:LSBus<x>[:ANALyze]:UART:BRATe

機能　ロジック UART 信号解析のビットレート (データ転送速度) を設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:UART:
BRATe {<NRf>|USER,<NRf>}
:ANALysis:LSBus<x>[:ANALyze]:UART:
BRATe?
```
<x> = 1、2
<NRf> = 1200、2400、4800、9600、19200、38400、57600、115200
USER の <NRf> = 4.6 節参照。

例
```
:ANALYSIS:LSBUS1:ANALYZE:UART:BRATE
19200
:ANALYSIS:LSBUS1:ANALYZE:UART:BRATE?
-> :ANALYSIS:LSBUS1:ANALYZE:UART:
BRATE 19200
```

:ANALysis:LSBus<x>[:ANALyze]:UART:BSPace

機能　ロジック UART 信号解析のグルーピングのバイトスペースを設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:UART:
BSPace {< 時間 >}
:ANALysis:LSBus<x>[:ANALyze]:UART:
BSPace?
```
<x> = 1、2
< 時間 > = 4.6 節参照。

例
```
:ANALYSIS:LSBUS1:ANALYZE:UART:
BSPACE 10ms
:ANALYSIS:LSBUS1:ANALYZE:UART:
BSPACE?       -> :ANALYSIS:LSBUS1:
ANALYZE:UART:BSPACE 10.00E-03
```

:ANALysis:LSBus<x>[:ANALyze]:UART:DFORmat

機能　ロジック UART 信号解析のデコード文字表示形式を設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:UART:
DFORmat {ASCii|HEXA}
:ANALysis:LSBus<x>[:ANALyze]:UART:
DFORmat?
```
<x> = 1、2

例
```
:ANALYSIS:LSBUS1:ANALYZE:UART:
DFORMAT ASCII
:ANALYSIS:LSBUS1:ANALYZE:UART:
DFORMAT? -> :ANALYSIS:LSBUS1:
ANALYZE:UART:DFORMAT ASCII
```

:ANALysis:LSBus<x>[:ANALyze]:UART:FORMat

機能　ロジック UART 信号解析のデータ形式を設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:UART:
FORMat {BIT7parity|BIT8Noparity|BIT8
Parity}
:ANALysis:LSBus<x>[:ANALyze]:UART:
FORMat?
```
<x> = 1、2

例
```
:ANALYSIS:LSBUS1:ANALYZE:UART:
FORMAT BIT7PARITY
:ANALYSIS:LSBUS1:ANALYZE:UART:
FORMAT?
-> :ANALYSIS:LSBUS1:ANALYZE:UART:
FORMAT BIT7PARITY
```

:ANALysis:LSBus<x>[:ANALyze]:UART:GROuping

機能　ロジック UART 信号解析のグルーピングの ON/OFF を設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:UART:
GROuping {<Boolean>}
:ANALysis:LSBus<x>[:ANALyze]:UART:
GROuping?
```
<x> = 1、2

例
```
:ANALYSIS:LSBUS1:ANALYZE:UART:
GROUPING ON
:ANALYSIS:LSBUS1:ANALYZE:UART:
GROUPING? -> :ANALYSIS:LSBUS1:
ANALYZE:UART:GROUPING 1
```

:ANALysis:LSBus<x>[:ANALyze]:UART:PMODe

機能　ロジック UART 信号解析の Parity モードを設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:UART:
PMODe {EVEN|ODD}
:ANALysis:LSBus<x>[:ANALyze]:UART:
PMODe?
```
<x> = 1、2

例
```
:ANALYSIS:LSBUS1:ANALYZE:UART:PMODE
EVEN
:ANALYSIS:LSBUS1:ANALYZE:UART:PMODE?
-> :ANALYSIS:LSBUS1:ANALYZE:UART:
PMODE EVEN
```

:ANALysis:LSBus<x>[:ANALyze]:UART:POLarity

機能　ロジック UART 信号解析の極性を設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:UART:
POLarity {NEGative|POSitive}
:ANALysis:LSBus<x>[:ANALyze]:UART:
POLarity?
```
<x> = 1、2

例
```
:ANALYSIS:LSBUS1:ANALYZE:UART:
POLARITY NEGATIVE
:ANALYSIS:LSBUS1:ANALYZE:UART:
POLARITY?
-> :ANALYSIS:LSBUS1:ANALYZE:UART:
POLARITY NEGATIVE
```

:ANALysis:LSBus<x>[:ANALyze]:UART:SPOint

機能　ロジック UART 信号解析のサンプルポイントを設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:UART:
SPOint {<NRf>}
:ANALysis:LSBus<x>[:ANALyze]:UART:
SPOint?
```
<x> = 1、2
<NRf> = 18.8 ～ 90.6(%)

例
```
:ANALYSIS:LSBUS1:ANALYZE:UART:
SPOINT 18.8
:ANALYSIS:LSBUS1:ANALYZE:UART:
SPOINT?
-> :ANALYSIS:LSBUS1:ANALYZE:UART:
SPOINT 18.8E+00
```

:ANALysis:LSBus<x>[:ANALyze]:UART:TRACe

機能　ロジック UART 信号解析のトレースを設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>[:ANALyze]:UART:
TRACe {A<y>|B<y>|C<y>|D<y>}
:ANALysis:LSBus<x>[:ANALyze]:UART:
TRACe?
```
<x> = 1、2
<y> = 0 ～ 7

例
```
:ANALYSIS:LSBUS1:ANALYZE:UART:
TRACE A0
:ANALYSIS:LSBUS1:ANALYZE:UART:TRACE?
-> :ANALYSIS:LSBUS1:ANALYZE:UART:
TRACE A0
```

解説　DLM6000 の 16 ビットモデルでは {A<x>|C<x>} が有効です。

:ANALysis:LSBus<x>:ZLINkage

機能　ロジックシリアルバス信号解析のズームリンクを設定 / 問い合わせします。

構文
```
:ANALysis:LSBus<x>:ZLINkage
{OFF|Z1|Z2}
:ANALysis:LSBus<x>:ZLINkage?
```
<x> = 1、2

例
```
:ANALYSIS:LSBUS1:ZLINKAGE OFF
:ANALYSIS:LSBUS1:ZLINKAGE?
-> :ANALYSIS:LSBUS1:ZLINKAGE OFF
```

:ANALysis:SBUS<x>?

機能　シリアルバス信号解析機能に関するすべての設定値を問い合わせします。

構文
```
:ANALysis:SBUS<x>?
```
<x> = 1、2

:ANALysis:SBUS<x>:ANALyze?

機能　シリアルバス信号解析に関するすべての設定値を問い合わせします。

構文
```
:ANALysis:SBUS<x>:ANALyze?
```
<x> = 1、2

:ANALysis:SBUS<x>[:ANALyze]:CANBus?

機能　CAN バス信号解析に関するすべての設定値を問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:CANBus?
```
<x> = 1、2

:ANALysis:SBUS<x>[:ANALyze]:CANBus:BRATe

機能　CAN バス信号解析のビットレート (データ転送速度) を設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:CANBus:
BRATe {<NRf>|USER,<NRf>}
:ANALysis:SBUS<x>[:ANALyze]:CANBus:
BRATe?
```
<x> = 1、2
<NRf> = 33300、83300、125000、250000、500000、1000000
USER の <NRf> = 4.3 節参照。

例
```
:ANALYSIS:SBUS1:ANALYZE:CANBUS:
BRATE 83300
:ANALYSIS:SBUS1:ANALYZE:CANBUS:
BRATE?
-> :ANALYSIS:SBUS1:ANALYZE:CANBUS:
BRATE 83300
```

:ANALysis:SBUS<x>[:ANALyze]:CANBus:FJUMp:ACK

機能　CAN バス信号解析の結果を対象に ACK Field へのフィールドジャンプを実行します。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:CANBus:
FJUMp:ACK
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:CANBUS:
FJUMP:
ACK
```

:ANALysis:SBUS<x>[:ANALyze]:CANBus:FJUMp:CONTrol

機能　CAN バス信号解析の結果を対象に Control Field へのフィールドジャンプを実行します。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:CANBus:
FJUMp:CONTrol
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:CANBUS:
FJUMP:
CONTROL
```

:ANALysis:SBUS<x>[:ANALyze]:CANBus:FJUMp:CRC

機能　CAN バス信号解析の結果を対象に CRC Field へのフィールドジャンプを実行します。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:CANBus:
FJUMp:CRC
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:CANBUS:
FJUMP:
CRC
```

:ANALysis:SBUS<x>[:ANALyze]:CANBus:FJUMp:DATA

機能　CAN バス信号解析の結果を対象に Data Field へのフィールドジャンプを実行します。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:CANBus:
FJUMp:DATA
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:CANBUS:
FJUMP:
DATA
```

:ANALysis:SBUS<x>[:ANALyze]:CANBus:FJUMp:IDENtifier

機能　CAN バス信号解析の結果を対象に Identifier Field へのフィールドジャンプを実行します。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:CANBus:
FJUMp:IDENtifier
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:CANBUS:
FJUMP:
IDENTIFIER
```

:ANALysis:SBUS<x>[:ANALyze]:CANBus:FJUMp:SOF

機能　CAN バス信号解析の結果を対象に SOF Field へのフィールドジャンプを実行します。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:CANBus:
FJUMp:SOF
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:CANBUS:
FJUMP:
SOF
```

:ANALysis:SBUS<x>[:ANALyze]:CANBus:RECessive

機能　CAN バス信号解析のリセッシブレベル (バスレベル) を設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:CANBus:
RECessive {HIGH|LOW}
:ANALysis:SBUS<x>[:ANALyze]:CANBus:
RECessive?
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:CANBUS:
RECESSIVE HIGH
:ANALYSIS:SBUS1:ANALYZE:CANBUS:
RECESSIVE?
-> :ANALYSIS:SBUS1:ANALYZE:CANBUS:
RECESSIVE HIGH
```

:ANALysis:SBUS<x>[:ANALyze]:CANBus:SPOint

機能　CAN バス信号解析のサンプルポイントを設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:CANBus:SPOint {<NRf>}
:ANALysis:SBUS<x>[:ANALyze]:CANBus:SPOint?
```
<x> = 1、2
<NRf> = 18.8 ～ 90.6(%)

例
```
:ANALYSIS:SBUS1:ANALYZE:CANBUS:SPOINT 18.8
:ANALYSIS:SBUS1:ANALYZE:CANBUS:SPOINT? -> :ANALYSIS:SBUS1:ANALYZE:CANBUS:SPOINT 18.8E+00
```

:ANALysis:SBUS<x>[:ANALyze]:CANBus:TRACe

機能　CAN バス信号解析のトレースを設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:CANBus:TRACe {<NRf>}
:ANALysis:SBUS<x>[:ANALyze]:CANBus:TRACe?
```
<x> = 1、2
<NRf> = 1 ～ 8

例
```
:ANALYSIS:SBUS1:ANALYZE:CANBUS:TRACE 1
:ANALYSIS:SBUS1:ANALYZE:CANBUS:TRACE?
-> :ANALYSIS:SBUS1:ANALYZE:CANBUS:TRACE 1
```

:ANALysis:SBUS<x>[:ANALyze]:DECode

機能　シリアルバス信号解析のデコード表示の ON/OFF を設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:DECode {<Boolean>}
:ANALysis:SBUS<x>[:ANALyze]:DECode?
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:DECODE ON
:ANALYSIS:SBUS1:ANALYZE:DECODE?
-> :ANALYSIS:SBUS1:ANALYZE:DECODE 1
```

:ANALysis:SBUS<x>[:ANALyze]:I2CBus?

機能　I^2C バス信号解析に関するすべての設定値を問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:I2CBus?
```
<x> = 1、2

ANALysis:SBUS<x>[:ANALyze]:I2CBus:CLOCk

機能　I^2C バス信号解析のクロックチャネルを設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:I2CBus:CLOCk {<NRf>}
:ANALysis:SBUS<x>[:ANALyze]:I2CBus:CLOCk?
```
<x> = 1、2
<NRf> = 1 ～ 8

例
```
:ANALYSIS:SBUS1:ANALYZE:I2CBUS:CLOCK 1
:ANALYSIS:SBUS1:ANALYZE:I2CBUS:CLOCK?
-> :ANALYSIS:SBUS1:ANALYZE:I2CBUS:CLOCK 1
```

:ANALysis:SBUS<x>[:ANALyze]:I2CBus:DTRace

機能　I^2C バス信号解析のデータチャネルを設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:I2CBus:DTRace {<NRf>}
:ANALysis:SBUS<x>[:ANALyze]:I2CBus:DTRace?
```
<x> = 1、2
<NRf> = 1 ～ 8

例
```
:ANALYSIS:SBUS1:ANALYZE:I2CBUS:DTRACE 1
:ANALYSIS:SBUS1:ANALYZE:I2CBUS:DTRACE? -> :ANALYSIS:SBUS1:ANALYZE:I2CBUS:DTRACE 1
```

:ANALysis:SBUS<x>[:ANALyze]:LINBus?

機能　LIN バス信号解析に関するすべての設定値を問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:LINBus?
```
<x> = 1、2

:ANALysis:SBUS<x>[:ANALyze]:LINBus:BRATe

機能　LIN バス信号解析のビットレート (データ転送速度) を設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:LINBus:
BRATe {<NRf>|USER,<NRf>}
:ANALysis:SBUS<x>[:ANALyze]:LINBus:
BRATe?
```
<x> = 1、2
<NRf> = 1200、2400、4800、9600、19200
USER の <NRf> = 4.4 節参照。

例
```
:ANALYSIS:SBUS1:ANALYZE:LINBUS:
BRATE 19200
:ANALYSIS:SBUS1:ANALYZE:LINBUS:
BRATE?
-> :ANALYSIS:SBUS1:ANALYZE:LINBUS:
BRATE 19200
```

:ANALysis:SBUS<x>[:ANALyze]:LINBus:FJUMp:BREak

機能　LIN バス信号解析の結果を対象に Break Field へのフィールドジャンプを実行します。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:LINBus:
FJUMp:BREak
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:LINBUS:
FJUMP:BREAK
```

:ANALysis:SBUS<x>[:ANALyze]:LINBus:FJUMp:CSUM

機能　LIN バス信号解析の結果を対象に Checksum Field へのフィールドジャンプを実行します。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:LINBus:
FJUMp:CSUM
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:LINBUS:
FJUMP:CSUM
```

:ANALysis:SBUS<x>[:ANALyze]:LINBus:FJUMp:DATA

機能　LIN バス信号解析の結果を対象に Data Field へのフィールドジャンプを実行します。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:LINBus:
FJUMp:DATA
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:LINBUS:
FJUMP:DATA
```

:ANALysis:SBUS<x>[:ANALyze]:LINBus:FJUMp:IDENtifier

機能　LIN バス信号解析の結果を対象に Identifier Field へのフィールドジャンプを実行します。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:LINBus:
FJUMp:IDENtifier
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:LINBUS:
FJUMP:IDENTIFIER
```

:ANALysis:SBUS<x>[:ANALyze]:LINBus:FJUMp:SYNCh

機能　LIN バス信号解析の結果を対象に Synch Field へのフィールドジャンプを実行します。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:LINBus:
FJUMp:SYNCh
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:LINBUS:
FJUMP:SYNCH
```

:ANALysis:SBUS<x>[:ANALyze]:LINBus:REVision

機能　LIN バス信号解析のレビジョン (1.3 or 2.0 or Both) を設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:LINBus:
REVision {BOTH|LIN1_3|LIN2_0}
:ANALysis:SBUS<x>[:ANALyze]:LINBus:
REVision?
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:LINBUS:
REVISION LIN1_3
:ANALYSIS:SBUS1:ANALYZE:LINBUS:
REVISION?
-> :ANALYSIS:SBUS1:ANALYZE:LINBUS:
REVISION LIN1_3
```

:ANALysis:SBUS<x>[:ANALyze]:LINBus:SPOint

機能　LIN バス信号解析のサンプルポイントを設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:LINBus:
SPOint {<NRf>}
:ANALysis:SBUS<x>[:ANALyze]:LINBus:
SPOint?
```
<x> = 1、2
<NRf> = 18.8 ～ 90.6(%)

例
```
:ANALYSIS:SBUS1:ANALYZE:LINBUS:
SPOINT 18.8
:ANALYSIS:SBUS1:ANALYZE:LINBUS:
SPOINT?
-> :ANALYSIS:SBUS1:ANALYZE:LINBUS:
SPOINT 18.8E+00
```

:ANALysis:SBUS<x>[:ANALyze]:LINBus:TRACe

機能　LIN バス信号解析のトレースを設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:LINBus:TRACe {<NRf>}
:ANALysis:SBUS<x>[:ANALyze]:LINBus:TRACe?
```
<x> = 1、2
<NRf> = 1 ～ 8

例
```
:ANALYSIS:SBUS1:ANALYZE:LINBUS:TRACE 1
:ANALYSIS:SBUS1:ANALYZE:LINBUS:TRACE?
-> :ANALYSIS:SBUS1:ANALYZE:LINBUS:TRACE 1
```

:ANALysis:SBUS<x>[:ANALyze]:LIST?

機能　シリアルバス信号解析の解析結果リストに関するすべての設定値を問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:LIST?
```
<x> = 1、2

:ANALysis:SBUS<x>[:ANALyze]:LIST:DISPlay

機能　シリアルバス信号解析の解析結果リストの ON/OFF を設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:LIST:DISPlay {<Boolean>}
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:LIST:DISPLAY ON
:ANALYSIS:SBUS1:ANALYZE:LIST:DISPLAY?
-> :ANALYSIS:SBUS1:ANALYZE:LIST:DISPLAY 1
```

:ANALysis:SBUS<x>[:ANALyze]:LIST:ITEM?

機能　シリアルバス信号解析の解析結果リストに表示される項目を問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:LIST:ITEM?
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:LIST:ITEM?
-> :ANALYSIS:SBUS1:ANALYZE:LIST:ITEM "No.,S/P,Hex,Form,R/W,ACK,"
```

:ANALysis:SBUS<x>[:ANALyze]:LIST:MODE

機能　シリアルバス信号解析の解析結果リストのモードを設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:LIST:MODE {DETail|SIMPle}
:ANALysis:SBUS<x>[:ANALyze]:LIST:MODE?
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:LIST:MODE DETAIL
:ANALYSIS:SBUS1:ANALYZE:LIST:MODE?
-> :ANALYSIS:SBUS1:ANALYZE:LIST:MODE DETAIL
```

:ANALysis:SBUS<x>[:ANALyze]:LIST:SCRoll

機能　シリアルバス信号解析の解析結果リストのスクロール方法を設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:LIST:SCRoll {HORizontal|VERTical}
:ANALysis:SBUS<x>[:ANALyze]:LIST:SCRoll?
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:LIST:SCROLL HORIZONTAL
:ANALYSIS:SBUS1:ANALYZE:LIST:SCROLL?
-> :ANALYSIS:SBUS1:ANALYZE:LIST:SCROLL HORIZONTAL
```

:ANALysis:SBUS<x>[:ANALyze]:LIST:VALue?

機能　シリアルバス信号解析の解析結果リストの指定した解析番号の自動測定値を問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:LIST:VALue? {<NRf>|MAXimum|MINimum}
```
<x> = 1、2
<NRf> = − 40000 ～ 40000
(「:ANALysis:SBUS<x>[:ANALyze]:MODE CANBus」のときは、<NRf> = − 2999 ～ 2999)

例
```
:ANALYSIS:SBUS1:ANALYZE:LIST:VALUE? 1
-> :ANALYSIS:SBUS1:ANALYZE:LIST:VALUE "1, P, 00, A, , 0,"
```

解説　データが「MAXimum」の場合は最大リスト表示番号、「MINimum」の場合は最小リスト表示番号が指定になります。

:ANALysis:SBUS<x>[:ANALyze]:MODE

機能　シリアルバス信号解析のモードを設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:MODE
{CANBus|I2CBus|LINBus|SPIBus|UART}
:ANALysis:SBUS<x>[:ANALyze]:MODE?
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:MODE I2CBUS
:ANALYSIS:SBUS1:ANALYZE:MODE?
-> :ANALYSIS:SBUS1:ANALYZE:
MODE I2CBUS
```

:ANALysis:SBUS<x>[:ANALyze]:RPOint

機能　シリアルバス信号解析のリファレンスポイント (解析基準点) を設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:
RPOint {MANual,<NRf>|TRIGger}
:ANALysis:SBUS<x>[:ANALyze]:RPOint?
```
<x> = 1、2
<NRf> = − 5 ～ 5(div)

例
```
:ANALYSIS:SBUS1:ANALYZE:RPOINT
MANAUL,1
:ANALYSIS:SBUS1:ANALYZE:RPOINT?
-> :ANALYSIS:SBUS1:ANALYZE:
RPOINT MANUAL,1.00000E+00
```

:ANALysis:SBUS<x>[:ANALyze]:SPIBus?

機能　SPI バス信号解析に関するすべての設定値を問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:SPIBus?
```
<x> = 1、2

:ANALysis:SBUS<x>[:ANALyze]:SPIBus:CLOCk?

機能　SPI バス信号解析のクロック信号のチャネルに関するすべての設定値を問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:SPIBus:
CLOCk?
```
<x> = 1、2

:ANALysis:SBUS<x>[:ANALyze]:SPIBus:CLOCk:POLarity

機能　SPI バス信号解析のクロック信号のチャネルの極性を設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:SPIBus:
CLOCk:POLarity {FALL|RISE}
:ANALysis:SBUS<x>[:ANALyze]:SPIBus:
CLOCk:POLarity?
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:SPIBUS:
CLOCK:
POLARITY FALL
:ANALYSIS:SBUS1:ANALYZE:SPIBUS:
CLOCK:
POLARITY?
-> :ANALYSIS:SBUS1:ANALYZE:SPIBUS:
CLOCK:POLARITY FALL
```

:ANALysis:SBUS<x>[:ANALyze]:SPIBus:CLOCk:SOURce

機能　SPI バス信号解析のクロック信号のチャネルを設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:SPIBus:
CLOCk:SOURce {<NRf>}
:ANALysis:SBUS<x>[:ANALyze]:SPIBus:
CLOCk:SOURce?
```
<x> = 1、2
<NRf> = 1 ～ 8

例
```
:ANALYSIS:SBUS1:ANALYZE:SPIBUS:
CLOCK:
SOURCE 1
:ANALYSIS:SBUS1:ANALYZE:SPIBUS:
CLOCK:
SOURCE?
-> :ANALYSIS:SBUS1:ANALYZE:SPIBUS:
CLOCK:SOURCE 1
```

:ANALysis:SBUS<x>[:ANALyze]:SPIBus:CS?

機能　SPI バス信号解析のチップセレクト信号のチャネルに関するすべての設定値を問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:SPIBus:
CS?
```
<x> = 1、2

:ANALysis:SBUS<x>[:ANALyze]:SPIBus:CS:ACTive

機能　SPI バス信号解析のチップセレクト信号のチャネルのアクティブレベルを設定 / 問い合わせします。

構文
```
ANALysis:SBUS<x>[:ANALyze]:SPIBus:
CS:
ACTive {HIGH|LOW}
:ANALysis:SBUS<x>[:ANALyze]:SPIBus:
CS:ACTive?
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:SPIBUS:CS:
ACTIVE HIGH
:ANALYSIS:SBUS1:ANALYZE:SPIBUS:CS:
ACTIVE? -> :ANALYSIS:SBUS1:ANALYZE:
SPIBUS:CS:ACTIVE HIGH
```

:ANALysis:SBUS<x>[:ANALyze]:SPIBus:CS:TRACe

機能　SPI バス信号解析のチップセレクト信号のチャネルを設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:SPIBus:
CS:TRACe {<NRf>|NONE}
:ANALysis:SBUS<x>[:ANALyze]:SPIBus:
CS:TRACe?
```
<x> = 1、2
<NRf> = 1 ～ 8

例
```
:ANALYSIS:SBUS1:ANALYZE:SPIBUS:CS:
TRACE 1
:ANALYSIS:SBUS1:ANALYZE:SPIBUS:CS:
TRACE? -> :ANALYSIS:SBUS1:ANALYZE:
SPIBUS:CS:TRACE 1
```

:ANALysis:SBUS<x>[:ANALyze]:SPIBus:DATA<x>?

機能　SPI バス信号解析の各データに関するすべての設定値を問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:SPIBus:
DATA<x>?
```
SBUS<x> の <x> = 1、2
DATA<x> の <x> = 1、2

:ANALysis:SBUS<x>[:ANALyze]:SPIBus:DATA<x>:ACTive

機能　SPI バス信号解析の各データのアクティブレベルを設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:SPIBus:
DATA<x>:ACTive {HIGH|LOW}
:ANALysis:SBUS<x>[:ANALyze]:SPIBus:
DATA<x>:ACTive?
```
SBUS<x> の <x> = 1、2
DATA<x> の <x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:SPIBUS:
DATA1:
ACTIVE HIGH
:ANALYSIS:SBUS1:ANALYZE:SPIBUS:
DATA1:
ACTIVE? -> :ANALYSIS:SBUS1:ANALYZE:
SPIBUS:DATA1:ACTIVE HIGH
```

:ANALysis:SBUS<x>[:ANALyze]:SPIBus:DATA<x>:TRACe

機能　SPI バス信号解析の各データチャネルを設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:SPIBus:
DATA<x>:TRACe {<NRf>}
:ANALysis:SBUS<x>[:ANALyze]:SPIBus:
DATA<x>:TRACe?
```
SBUS<x> の <x> = 1、2
DATA<x> の <x> = 1、2
<NRf> = 1 ～ 8

例
```
:ANALYSIS:SBUS1:ANALYZE:SPIBUS:
DATA1:
TRACE 1
:ANALYSIS:SBUS1:ANALYZE:SPIBUS:
DATA1:
TRACE? -> :ANALYSIS:SBUS1:ANALYZE:
SPIBUS:DATA1:TRACE 1
```

:ANALysis:SBUS<x>[:ANALyze]:SPIBus:SETup?

機能　SPI バス信号解析のセットアップに関するすべての設定値を問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:SPIBus:
SETup?
```
<x> = 1、2

:ANALysis:SBUS<x>[:ANALyze]:SPIBus[:SETup]:BITorder

機能　SPI バス信号解析のビットオーダを設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:SPIBus[:SETup]:BITorder {LSBFirst|MSBFirst}
:ANALysis:SBUS<x>[:ANALyze]:SPIBus[:SETup]:BITorder?
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:SPIBUS:SETUP:BITORDER LSBFIRST
:ANALYSIS:SBUS1:ANALYZE:SPIBUS:SETUP:BITORDER? -> :ANALYSIS:SBUS1:ANALYZE:SPIBUS:SETUP:BITORDER LSBFIRST
```

:ANALysis:SBUS<x>[:ANALyze]:SPIBus[:SETup]:EMSBLSB

機能　SPI バス信号解析のフィールドの有効範囲を設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:SPIBus[:SETup]:EMSBLSB {<NRf>,<NRf>}
:ANALysis:SBUS<x>[:ANALyze]:SPIBus[:SETup]:EMSBLSB?
```
<x> = 1、2
<NRf> = 4.5 節参照。

例
```
:ANALYSIS:SBUS1:ANALYZE:SPIBUS:SETUP:EMSBLSB 1,7
:ANALYSIS:SBUS1:ANALYZE:SPIBUS:SETUP:EMSBLSB? -> :ANALYSIS:SBUS1:ANALYZE:SPIBUS:SETUP:EMSBLSB 1,7
```

:ANALysis:SBUS<x>[:ANALyze]:SPIBus[:SETup]:FSIZe

機能　SPI バス信号解析のフィールドサイズを設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:SPIBus[:SETup]:FSIZe {<NRf>}
:ANALysis:SBUS<x>[:ANALyze]:SPIBus[:SETup]:FSIZe?
```
<x> = 1、2
<NRf> = 4 ～ 32

例
```
:ANALYSIS:SBUS1:ANALYZE:SPIBUS:SETUP:FSIZE 4
:ANALYSIS:SBUS1:ANALYZE:SPIBUS:SETUP:FSIZE? -> :ANALYSIS:SBUS1:ANALYZE:SPIBUS:SETUP:FSIZE 4
```

:ANALysis:SBUS<x>[:ANALyze]:SPIBus[:SETup]:ITIMe

機能　SPI バス信号解析のアイドル時間を設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:SPIBus[:SETup]:ITIMe {< 時間 >|DONTcare}
:ANALysis:SBUS<x>[:ANALyze]:SPIBus[:SETup]:ITIMe?
```
<x> = 1、2
< 時間 > = 10ns ～ 1ms(10ns ステップ)

例
```
:ANALYSIS:SBUS1:ANALYZE:SPIBUS:SETUP:ITIME 10NS
:ANALYSIS:SBUS1:ANALYZE:SPIBUS:SETUP:ITIME? -> :ANALYSIS:SBUS1:ANALYZE:SPIBUS:SETUP:ITIME 10.0000E-09
```

:ANALysis:SBUS<x>[:ANALyze]:SPIBus[:SETup]:MODE

機能　SPI バス信号解析の結線方式 (3 線式 /4 線式) を設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:SPIBus[:SETup]:MODE {WIRE3|WIRE4}
:ANALysis:SBUS<x>[:ANALyze]:SPIBus[:SETup]:MODE?
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:SPIBUS:SETUP:MODE WIRE3
:ANALYSIS:SBUS1:ANALYZE:SPIBUS:SETUP:MODE? -> :ANALYSIS:SBUS1:ANALYZE:SPIBUS:SETUP:MODE WIRE3
```

:ANALysis:SBUS<x>[:ANALyze]:TRACe<x>?

機能 :　シリアルバス信号解析の各ソースチャネルのしきい値 (Threshold) に関するすべての設定値を問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:TRACe<x>?
```
SBUS<x> の <x> = 1、2
TRACe<x> の <x> = 1 ～ 8

:ANALysis:SBUS<x>[:ANALyze]:TRACe<x>:HYSTeresis

機能　シリアルバス信号解析の各ソースチャネルのしきい値 (Threshold) のヒステリシスを設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:
TRACe<x>:
HYSTeresis {<NRf>}
:ANALysis:SBUS<x>[:ANALyze]:
TRACe<x>:
HYSTeresis?
```
SBUS<x> の <x> = 1、2
TRACe<x> の <x> = 1 ～ 8
<NRf> = 0 ～ 4(div)

例
```
:ANALYSIS:SBUS1:ANALYZE:TRACE1:
HYSTERESIS 1
:ANALYSIS:SBUS1:ANALYZE:TRACE1:
HYSTERESIS? -> :ANALYSIS:SBUS1:
ANALYZE:TRACE1:HYSTERESIS 1.000E+00
```

:ANALysis:SBUS<x>[:ANALyze]:TRACe<x>:LEVel

機能　シリアルバス信号解析の各ソースチャネルのしきい値 (Threshold) のレベルを設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:
TRACe<x>:
LEVel {<NRf>|< 電圧 >|< 電流 >}
:ANALysis:SBUS<x>[:ANALyze]:
TRACe<x>:
LEVel?
```
SBUS<x> の <x> = 1、2
TRACe<x> の <x> = 1 ～ 8
<NRf>、< 電圧 >、< 電流 > = 4.2 ～ 4.6 節参照。

例
```
:ANALYSIS:SBUS1:ANALYZE:TRACE1:
LEVEL 1V
:ANALYSIS:SBUS1:ANALYZE:TRACE1:
LEVEL? -> :ANALYSIS:SBUS1:ANALYZE:
TRACE1:LEVEL 1.000E+00
```

:ANALysis:SBUS<x>[:ANALyze]:UART?

機能　UART 信号解析に関するすべての設定値を問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:UART?
```
<x> = 1、2

:ANALysis:SBUS<x>[:ANALyze]:UART:BITorder

機能　UART 信号解析のビットオーダを設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:UART:
BITorder {LSBFirst|MSBFirst}
:ANALysis:SBUS<x>[:ANALyze]:UART:
BITorder?
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:UART:
BITORDER LSBFIRST
:ANALYSIS:SBUS1:ANALYZE:UART:
BITORDER?
-> :ANALYSIS:SBUS1:ANALYZE:UART:
BITORDER LSBFIRST
```

:ANALysis:SBUS<x>[:ANALyze]:UART:BRATe

機能　UART 信号解析のビットレート (データ転送速度) を設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:UART:
BRATe {<NRf>|USER,<NRf>}
      :ANALysis:SBUS<x>[:ANALyze]:
UART:BRATe?
```
<x> = 1、2
<NRf> = 1200、2400、4800、9600、19200、38400、57600、115200
USER の <NRf> = 4.6 節参照。

例
```
:ANALYSIS:SBUS1:ANALYZE:UART:
BRATE 19200
:ANALYSIS:SBUS1:ANALYZE:UART:BRATE?
-> :ANALYSIS:SBUS1:ANALYZE:UART:
BRATE 19200
```

:ANALysis:SBUS<x>[:ANALyze]:UART:BSPace

機能　UART 信号解析のグルーピングのバイトスペースを設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:UART:
BSPace {< 時間 >}
:ANALysis:SBUS<x>[:ANALyze]:UART:
BSPace?
```
<x> = 1、2
< 時間 > = 4.6 節参照。

例
```
:ANALYSIS:SBUS1:ANALYZE:UART:
BSPACE 10ms
:ANALYSIS:SBUS1:ANALYZE:UART:BSPACE?
-> :ANALYSIS:SBUS1:ANALYZE:UART:
BSPACE 10.00E-03
```

:ANALysis:SBUS<x>[:ANALyze]:UART:DFORmat

機能　UART 信号解析のデコード文字表示形式を設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:UART:
DFORmat {ASCii|HEXA}
:ANALysis:SBUS<x>[:ANALyze]:UART:
DFORmat?
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:UART:
DFORMAT ASCII
:ANALYSIS:SBUS1:ANALYZE:UART:
DFORMAT? -> :ANALYSIS:SBUS1:ANALYZE:
UART:DFORMAT ASCII
```

:ANALysis:SBUS<x>[:ANALyze]:UART:FORMat

機能　UART 信号解析のデータ形式を設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:UART:
FORMat {BIT7parity|BIT8Noparity|BIT8
Parity}
:ANALysis:SBUS<x>[:ANALyze]:UART:
FORMat?
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:UART:
FORMAT BIT7PARITY
:ANALYSIS:SBUS1:ANALYZE:UART:FORMAT?
-> :ANALYSIS:SBUS1:ANALYZE:UART:
FORMAT BIT7PARITY
```

:ANALysis:SBUS<x>[:ANALyze]:UART:GROuping

機能　UART 信号解析のグルーピングの ON/OFF を設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:UART:
GROuping {<Boolean>}
:ANALysis:SBUS<x>[:ANALyze]:UART:
GROuping?
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:UART:
GROUPING ON
:ANALYSIS:SBUS1:ANALYZE:UART:
GROUPING? -> :ANALYSIS:SBUS1:
ANALYZE:UART:GROUPING 1
```

:ANALysis:SBUS<x>[:ANALyze]:UART:PMODe

機能　UART 信号解析の Parity モードを設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:UART:
PMODe {EVEN|ODD}
:ANALysis:SBUS<x>[:ANALyze]:UART:
PMODe?
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:UART:
PMODE EVEN
:ANALYSIS:SBUS1:ANALYZE:UART:PMODE?
-> :ANALYSIS:SBUS1:ANALYZE:UART:
PMODE EVEN
```

:ANALysis:SBUS<x>[:ANALyze]:UART:POLarity

機能　UART 信号解析の極性を設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:UART:
POLarity {NEGative|POSitive}
:ANALysis:SBUS<x>[:ANALyze]:UART:
POLarity?
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ANALYZE:UART:
POLARITY NEGATIVE
:ANALYSIS:SBUS1:ANALYZE:UART:
POLARITY?
-> :ANALYSIS:SBUS1:ANALYZE:UART:
POLARITY NEGATIVE
```

:ANALysis:SBUS<x>[:ANALyze]:UART:SPOint

機能　UART 信号解析のサンプルポイントを設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:UART:
SPOint {<NRf>}
:ANALysis:SBUS<x>[:ANALyze]:UART:
SPOint?
```
<x> = 1、2
<NRf> = 18.8 ～ 90.6(%)

例
```
:ANALYSIS:SBUS1:ANALYZE:UART:
SPOINT 18.8
:ANALYSIS:SBUS1:ANALYZE:UART:SPOINT?
-> :ANALYSIS:SBUS1:ANALYZE:UART:
SPOINT 18.8E+00
```

:ANALysis:SBUS<x>[:ANALyze]:UART:TRACe

機能　UART 信号解析のトレースを設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>[:ANALyze]:UART:TRACe {<NRf>}
:ANALysis:SBUS<x>[:ANALyze]:UART:TRACe?
```
<x> = 1、2
<NRf> = 1 ～ 8

例
```
:ANALYSIS:SBUS1:ANALYZE:UART:TRACE 1
:ANALYSIS:SBUS1:ANALYZE:UART:TRACE?
-> :ANALYSIS:SBUS1:ANALYZE:UART:TRACE 1
```

:ANALysis:SBUS<x>:ZLINkage

機能　シリアルバス信号解析のズームリンクを設定 / 問い合わせします。

構文
```
:ANALysis:SBUS<x>:ZLINkage {OFF|Z1|Z2}
:ANALysis:SBUS<x>:ZLINkage?
```
<x> = 1、2

例
```
:ANALYSIS:SBUS1:ZLINKAGE OFF
:ANALYSIS:SBUS1:ZLINKAGE?
-> :ANALYSIS:SBUS1:ZLINKAGE OFF
```

:ANALysis:TYPE<x>

機能　解析機能のタイプを設定 / 問い合わせします。

構文
```
:ANALysis:TYPE<x> {AHIStogram|CANBus|FFT|HARMonics|I2CBus|LINBus|SPIBus|UART|WPARameter|XY|OFF}
:ANALysis:TYPE<x>?
```
<x> = 1、2

例
```
:ANALYSIS:TYPE1 AHISTOGRAM
:ANALYSIS:TYPE1? -> :ANALYSIS:TYPE1 AHISTOGRAM
```

解説　{HARMonics|LSBus} は、DLM6000 に適用できます。

7.3 SEARch グループ

SEARch<x>:CANBus?

機能　CAN バス信号サーチに関するすべての設定値を問い合わせします。

構文　:SEARch<x>:CANBus?
<x> = 1、2

:SEARch<x>:CANBus:SETup?

機能　CAN バス信号サーチのセットアップに関するすべての設定値を問い合わせします。

構文　:SEARch<x>:CANBus:SETup?
<x> = 1、2

:SEARch<x>:CANBus[:SETup]:ACK

機能　CAN バス信号サーチの ACK 条件を設定 / 問い合わせします。

構文　:SEARch<x>:CANBus[:SETup]:
ACK {ACK|ACKBoth|DONTcare|NONack}
:SEARch<x>:CANBus[:SETup]:ACK?
<x> = 1、2

例　:SEARCH1:CANBUS:SETUP:ACK ACK
:SEARCH1:CANBUS:SETUP:ACK?
-> :SEARCH1:CANBUS:SETUP:ACK ACK

:SEARch<x>:CANBus[:SETup]:BRATe

機能　CAN バス信号サーチのビットレート (データ転送速度) を設定 / 問い合わせします。

構文　:SEARch<x>:CANBus[:SETup]:
BRATe {<NRf>|USER,<NRf>}
:SEARch<x>:CANBus[:SETup]:BRATe?
<x> = 1、2
<NRf> = 33300、83300、125000、250000、500000、1000000
USER の <NRf>=5.3 節参照。

例　:SEARCH1:CANBUS:SETUP:BRATE 83300
:SEARCH1:CANBUS:SETUP:BRATE?
-> :SEARCH1:CANBUS:SETUP:BRATE 83300

:SEARch<x>:CANBus[:SETup]:DATA?

機能　CAN バス信号サーチのデータに関するすべての設定値を問い合わせします。

構文　:SEARch<x>:CANBus[:SETup]:DATA?
<x> = 1、2

:SEARch<x>:CANBus[:SETup]:DATA:BORDer

機能　CAN バス信号サーチのデータのバイトオーダを設定 / 問い合わせします。

構文　:SEARch<x>:CANBus[:SETup]:DATA:
BORDer {BIG|LITTle}
:SEARch<x>:CANBus[:SETup]:DATA:
BORDer?
<x> = 1、2

例　:SEARCH1:CANBUS:SETUP:DATA:BORDER BIG
:SEARCH1:CANBUS:SETUP:DATA:BORDER?
-> :SEARCH1:CANBUS:SETUP:DATA:BORDER BIG

:SEARch<x>:CANBus[:SETup]:DATA:CONDition

機能　CAN バス信号サーチのデータ条件を設定 / 問い合わせします。

構文　:SEARch<x>:CANBus[:SETup]:DATA:
CONDition {BETWeen|DONTcare|FALSe|GTHan|LTHan|ORANge|TRUE}
:SEARch<x>:CANBus[:SETup]:DATA:
CONDition?
<x> = 1、2

例　:SEARCH1:CANBUS:SETUP:DATA:CONDITION BETWEEN
:SEARCH1:CANBUS:SETUP:DATA:CONDITION? -> :SEARCH1:CANBUS:SETUP:DATA:CONDITION BETWEEN

:SEARch<x>:CANBus[:SETup]:DATA:DATA<x>

機能　CAN バス信号サーチのデータの比較データを設定 / 問い合わせします。

構文
```
:SEARch<x>:CANBus[:SETup]:DATA:DATA<x> {<NRf>}
:SEARch<x>:CANBus[:SETup]:DATA:DATA<x>?
```
SEARch<x> の <x> = 1、2
DATA<x> の <x> = 1、2
<NRf>=5.3 節参照。

例
```
:SEARCH1:CANBUS:SETUP:DATA:DATA1 1
:SEARCH1:CANBUS:SETUP:DATA:DATA1?
-> :SEARCH1:CANBUS:SETUP:DATA:DATA1 1.0000000E+00
```

解説
- 「:SEARch<x>:CANBus[:SETup]:DATA:CONDition GTHan」のときは「:SEARch<x>:CANBus[:SETup]:DATA:DATA1」で設定します。
- 「:SEARch<x>:CANBus[:SETup]:DATA:CONDition LTHan」のときは「:SEARch<x>:CANBus[:SETup]:DATA:DATA2」で設定します。
- 「:SEARch<x>:CANBus[:SETup]:DATA:CONDition BETWeen|ORANge」のときは、小さい値を「:SEARch<x>:CANBus[:SETup]:DATA:DATA1」、大きい値を「:SEARch<x>:CANBus[:SETup]:DATA:DATA2」で設定します。

:SEARch<x>:CANBus[:SETup]:DATA:DLC

機能　CAN バス信号サーチのデータの有効バイト数 (DLC) を設定 / 問い合わせします。

構文
```
:SEARch<x>:CANBus[:SETup]:DATA:DLC {<NRf>}
:SEARch<x>:CANBus[:SETup]:DATA:DLC?
```
<x> = 1、2
<NRf> = 0 ～ 8

例
```
:SEARCH1:CANBUS:SETUP:DATA:DLC 0
:SEARCH1:CANBUS:SETUP:DATA:DLC?
-> :SEARCH1:CANBUS:SETUP:DATA:DLC 0
```

:SEARch<x>:CANBus[:SETup]:DATA:HEXA

機能　CAN バス信号サーチのデータを HEXA で設定します。

構文
```
:SEARch<x>:CANBus[:SETup]:DATA:HEXA {< 文字列 >}
```
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 16 文字以内 (1 バイト単位)

例
```
:SEARCH1:CANBUS:SETUP:DATA:HEXA "A9"
```

:SEARch<x>:CANBus[:SETup]:DATA:MSBLsb

機能　CAN バス信号サーチのデータの MSB/LSB のビットを設定 / 問い合わせします。

構文
```
:SEARch<x>:CANBus[:SETup]:DATA:MSBLsb {<NRf>,<NRf>}
:SEARch<x>:CANBus[:SETup]:DATA:MSBLsb?
```
<x> = 1、2
<NRf> = 5.3 節参照。

例
```
:SEARCH1:CANBUS:SETUP:DATA:MSBLSB 1,0
:SEARCH1:CANBUS:SETUP:DATA:MSBLSB?
-> :SEARCH1:CANBUS:SETUP:DATA:MSBLSB 1,0
```

:SEARch<x>:CANBus[:SETup]:DATA:PATTern

機能　CAN バス信号サーチのデータを BINARY で設定 / 問い合わせします。

構文
```
:SEARch<x>:CANBus[:SETup]:DATA:PATTern {< 文字列 >}
:SEARch<x>:CANBus[:SETup]:DATA:PATTern?
```
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 64 文字以内 (1 バイト単位)

例
```
:SEARCH1:CANBUS:SETUP:DATA:PATTERN "11011111"
:SEARCH1:CANBUS:SETUP:DATA:PATTERN?
-> :SEARCH1:CANBUS:SETUP:DATA:PATTERN "11011111"
```

:SEARch<x>:CANBus[:SETup]:DATA:SIGN

機能　CAN バス信号サーチのデータの符号を設定 / 問い合わせします。

構文
```
:SEARch<x>:CANBus[:SETup]:DATA:SIGN {SIGN|UNSign}
:SEARch<x>:CANBus[:SETup]:DATA:SIGN?
```
<x> = 1、2

例
```
:SEARCH1:CANBUS:SETUP:DATA:SIGN SIGN
:SEARCH1:CANBUS:SETUP:DATA:SIGN?
-> :SEARCH1:CANBUS:SETUP:DATA:SIGN SIGN
```

:SEARch<x>:CANBus[:SETup]:IDEXt?
機能　CAN バス信号サーチの拡張フォーマットの ID に関するすべての設定値を問い合わせします。
構文　:SEARch<x>:CANBus[:SETup]:IDEXt?
<x> = 1、2

:SEARch<x>:CANBus[:SETup]:IDEXt:HEXA
機能　CAN バス信号サーチの拡張フォーマットの ID を HEXA で設定します。
構文　:SEARch<x>:CANBus[:SETup]:IDEXt:HEXA {< 文字列 >}
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 8 文字
例　:SEARCH1:CANBUS:SETUP:IDEXT:HEXA "1AEF5906"

:SEARch<x>:CANBus[:SETup]:IDEXt:PATTern
機能　CAN バス信号サーチの拡張フォーマットの ID を BINARY で設定 / 問い合わせします。
構文　:SEARch<x>:CANBus[:SETup]:IDEXt:PATTern {< 文字列 >}
:SEARch<x>:CANBus[:SETup]:IDEXt:PATTern?
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 29 文字
例　:SEARCH1:CANBUS:SETUP:IDEXT:PATTERN "11001011011100001110111011111"
:SEARCH1:CANBUS:SETUP:IDEXT:PATTERN?
-> :SEARCH1:CANBUS:SETUP:IDEXT:PATTERN "11001011011100001110111011111"

:SEARch<x>:CANBus[:SETup]:IDSTd?
機能　CAN バス信号サーチの標準フォーマットの ID に関するすべての設定値を問い合わせします。
構文　:SEARch<x>:CANBus[:SETup]:IDSTd?
<x> = 1、2

:SEARch<x>:CANBus[:SETup]:IDSTd:HEXA
機能　CAN バス信号サーチの標準フォーマットの ID を HEXA で設定します。
構文　:SEARch<x>:CANBus[:SETup]:IDSTd:HEXA {< 文字列 >}
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 3 文字
例　:SEARCH1:CANBUS:SETUP:IDSTD:HEXA "5DF"

:SEARch<x>:CANBus[:SETup]:IDSTd:PATTern
機能　CAN バス信号サーチの標準フォーマットの ID を BINARY で設定 / 問い合わせします。
構文　:SEARch<x>:CANBus[:SETup]:IDSTd:PATTern {< 文字列 >}
:SEARch<x>:CANBus[:SETup]:IDSTd:PATTern?
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 11 文字
例　:SEARCH1:CANBUS:SETUP:IDSTD:PATTERN "10111011111"
:SEARCH1:CANBUS:SETUP:IDSTD:PATTERN?
-> :SEARCH1:CANBUS:SETUP:IDSTD:PATTERN "10111011111"

:SEARch<x>:CANBus[:SETup]:MODE
機能　CAN バス信号サーチのモードを設定 / 問い合わせします。
構文　:SEARch<x>:CANBus[:SETup]:MODE {EFRame|IDEXt|IDSTd|SOF}
:SEARch<x>:CANBus[:SETup]:MODE?
<x> = 1、2
例　:SEARCH1:CANBUS:SETUP:MODE EFRAME
:SEARCH1:CANBUS:SETUP:MODE?
-> :SEARCH1:CANBUS:SETUP:MODE EFRAME

:SEARch<x>:CANBus[:SETup]:RECessive
機能　CAN バス信号サーチのリセッシブレベル (バスレベル) を設定 / 問い合わせします。
構文　:SEARch<x>:CANBus[:SETup]:RECessive {HIGH|LOW}
:SEARch<x>:CANBus[:SETup]:RECessive?
<x> = 1、2
例　:SEARCH1:CANBUS:SETUP:RECESSIVE HIGH
:SEARCH1:CANBUS:SETUP:RECESSIVE?
-> :SEARCH1:CANBUS:SETUP:RECESSIVE HIGH

:SEARch<x>:CANBus[:SETup]:RTR
機能　CAN バス信号サーチの RTR を設定 / 問い合わせします。
構文　:SEARch<x>:CANBus[:SETup]:RTR {DATA|DONTcare|REMote}
:SEARch<x>:CANBus[:SETup]:RTR?
<x> = 1、2
例　:SEARCH1:CANBUS:SETUP:RTR DATA
:SEARCH1:CANBUS:SETUP:RTR?
-> :SEARCH1:CANBUS:SETUP:RTR DATA

:SEARch<x>:CANBus[:SETup]:SPOint

機能　CAN バス信号サーチのサンプルポイントを設定 / 問い合わせします。

構文
```
:SEARch<x>:CANBus[:SETup]:SPOint {<NRf>}
:SEARch<x>:CANBus[:SETup]:SPOint?
```
<x> = 1、2
<NRf>=18.8 ～ 90.6(%)

例
```
:SEARCH1:CANBUS:SETUP:SPOINT 18.8
:SEARCH1:CANBUS:SETUP:SPOINT?
-> :SEARCH1:CANBUS:SETUP:SPOINT 18.8E+00
```

:SEARch<x>:CANBus[:SETup]:TRACe

機能　CAN バス信号サーチのトレースを設定 / 問い合わせします。

構文
```
:SEARch<x>:CANBus[:SETup]:TRACe {<NRf>}
:SEARch<x>:CANBus[:SETup]:TRACe?
```
<x> = 1、2
<NRf> = 1 ～ 8

例
```
:SEARCH1:CANBUS:SETUP:TRACE 1
:SEARCH1:CANBUS:SETUP:TRACE?
-> :SEARCH1:CANBUS:SETUP:TRACE 1
```

SEARch<x>:I2CBus?

機能　I²C バス信号サーチに関するすべての設定値を問い合わせします。

構文
```
:SEARch<x>:I2CBus?
```
<x> = 1、2

:SEARch<x>:I2CBus:CLOCk?

機能　I²C バス信号サーチのクロックチャネルに関するすべての設定値を問い合わせします。

構文
```
:SEARch<x>:I2CBus:CLOCk?
```
<x> = 1、2

:SEARch<x>:I2CBus:CLOCk:SOURce

機能　I²C バス信号サーチのクロックチャネルを設定 / 問い合わせします。

構文
```
:SEARch<x>:I2CBus:CLOCk:SOURce {<NRf>}
:SEARch<x>:I2CBus:CLOCk:SOURce?
```
<x> = 1、2
<NRf> = 1 ～ 8

例
```
:SEARCH1:I2CBUS:CLOCK:SOURCE 1
:SEARCH1:I2CBUS:CLOCK:SOURCE?
-> :SEARCH1:I2CBUS:CLOCK:SOURCE 1
```

:SEARch<x>:I2CBus:SETup?

機能　I²C バス信号サーチのセットアップに関するすべての設定値を問い合わせします。

構文
```
:SEARch<x>:I2CBus:SETup?
```
<x> = 1、2

:SEARch<x>:I2CBus[:SETup]:ADATa?

機能　I²C バス信号サーチのアドレスに関するすべての設定値を問い合わせします。

構文
```
:SEARch<x>:I2CBus[:SETup]:ADATa?
```
<x> = 1、2

:SEARch<x>:I2CBus[:SETup]:ADATa:BIT10address?

機能　I²C バス信号サーチの 10bit アドレスに関するすべての設定値を問い合わせします。

構文
```
:SEARch<x>:I2CBus[:SETup]:ADATa:BIT10address?
```
<x> = 1、2

:SEARch<x>:I2CBus[:SETup]:ADATa:BIT10address:HEXA

機能　I²C バス信号サーチの 10bit アドレスを HEXA で設定します。

構文
```
:SEARch<x>:I2CBus[:SETup]:ADATa:BIT10address:HEXA {< 文字列 >}
```
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 3 文字 (ビット 8 は、 R/$\overline{W}$ ビット)

例
```
:SEARCH1:I2CBUS:SETUP:ADATA:BIT10ADDRESS:HEXA "5DF"
```

:SEARch<x>:I2CBus[:SETup]:ADATa:BIT10address:PATTern

機能　I²C バス信号サーチの 10bit アドレスを BINARY で設定 / 問い合わせします。

構文
```
:SEARch<x>:I2CBus[:SETup]:ADATa:BIT10address:PATTern {< 文字列 >}
:SEARch<x>:I2CBus[:SETup]:ADATa:BIT10address:PATTern?
```
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 11 文字 (ビット 8 は、 R/$\overline{W}$ ビット)

例
```
:SEARCH1:I2CBUS:SETUP:ADATA:BIT10ADDRESS:PATTERN "10111011111"
:SEARCH1:I2CBUS:SETUP:ADATA:BIT10ADDRESS:PATTERN?
-> :SEARCH1:I2CBUS:SETUP:ADATA:BIT10ADDRESS:PATTERN "10111011111"
```

:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7ADdress?

機能　I²C バス信号サーチの 7bit アドレスに関するすべての設定値を問い合わせします。

構文
```
:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7ADdress?
```
<x> = 1、2

:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7ADdress:HEXA

機能　I²C バス信号サーチの 7bit アドレスを HEXA で設定します。

構文　:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7ADdress:HEXA {< 文字列 >}

<x> = 1、2

< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字 (ビット 0 は、R/$\overline{W}$ ビット)

例　:SEARCH1:I2CBUS:SETUP:ADATA:BIT7ADDRESS:HEXA "DE"

:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7ADdress:PATTern

機能　I²C バス信号サーチの 7bit アドレスを BINARY で設定 / 問い合わせします。

構文　:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7ADdress:PATTern {< 文字列 >}

:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7ADdress:PATTern?

<x> = 1、2

< 文字列 > = '0'、'1'、'X' の組み合わせ 8 文字 (ビット 0 は、R/$\overline{W}$ ビット)

例　:SEARCH1:I2CBUS:SETUP:ADATA:BIT7ADDRESS:PATTERN "11011110"

:SEARCH1:I2CBUS:SETUP:ADATA:BIT7ADDRESS:PATTERN?

-> :SEARCH1:I2CBUS:SETUP:ADATA:BIT7ADDRESS:PATTERN "11011110"

:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7APsub?

機能　I²C バス信号サーチの 7bit+Sub アドレスに関するすべての設定値を問い合わせします。

構文　:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7APsub?

<x> = 1、2

:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7APsub:ADDRess?

機能　I²C バス信号サーチの 7bit+Sub アドレスの 7bit アドレスに関するすべての設定値を問い合わせします。

構文　:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7APsub:ADDRess?

<x> = 1、2

:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7APsub:ADDRess:HEXA

機能　I²C バス信号サーチの 7bit+Sub アドレスの 7bit アドレスを HEXA で設定します。

構文　:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7APsub:ADDRess:HEXA {< 文字列 >}

<x> = 1、2

< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字 (ビット 0 は、R/$\overline{W}$ ビット)

例　:SEARCH1:I2CBUS:SETUP:ADATA:BIT7APSUB:ADDRESS:HEXA "CD"

:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7APsub:ADDRess:PATTern

機能　I²C バス信号サーチの 7bit+Sub アドレスの 7bit アドレスを BINARY で設定 / 問い合わせします。

構文　:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7APsub:ADDRess:PATTern {< 文字列 >}

:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7APsub:ADDRess:PATTern?

<x> = 1、2

< 文字列 > = '0'、'1'、'X' の組み合わせ 8 文字 (ビット 0 は、R/$\overline{W}$ ビット)

例　:SEARCH1:I2CBUS:SETUP:ADATA:BIT7APSUB:ADDRESS:PATTERN "11001101"

:SEARCH1:I2CBUS:SETUP:ADATA:BIT7APSUB:ADDRESS:PATTERN?

-> :SEARCH1:I2CBUS:SETUP:ADATA:BIT7APSUB:ADDRESS:PATTERN "11001101"

:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7APsub:SADDress?

機能　I²C バス信号サーチの 7bit+Sub アドレスの Sub アドレスに関するすべての設定値を問い合わせします。

構文　:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7APsub:SADDress?

<x> = 1、2

:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7APsub:SADDress:HEXA

機能　I²C バス信号サーチの 7bit+Sub アドレスの Sub アドレスを HEXA で設定します。

構文　:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7APsub:SADDress:HEXA {< 文字列 >}

<x> = 1、2

< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字

例　:SEARCH1:I2CBUS:SETUP:ADATA:BIT7APSUB:SADDRESS:HEXA "EF"

:SEARch<x>:I2CBus[:SETup]:ADATa:BIT7APsub:SADDress:PATTern

機能　I²C バス信号サーチの 7bit+Sub アドレスの Sub アドレスを BINARY で設定 / 問い合わせします。

構文
```
:SEARch<x>:I2CBus[:SETup]:ADATa:
BIT7APsub:SADDress:PATTern {< 文字列 >}
:SEARch<x>:I2CBus[:SETup]:ADATa:
BIT7APsub:SADDress:PATTern?
```
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 8 文字

例
```
:SEARCH1:I2CBUS:SETUP:ADATA:
BIT7APSUB:SADDRESS:PATTERN
"11101111"
:SEARCH1:I2CBUS:SETUP:ADATA:
BIT7APSUB:SADDRESS:PATTERN?
-> :SEARCH1:I2CBUS:SETUP:ADATA:
BIT7APSUB:SADDRESS:PATTERN
"11101111"
```

:SEARch<x>:I2CBus[:SETup]:ADATa:TYPE

機能　I²C バス信号サーチのアドレスの種類を設定 / 問い合わせします。

構文
```
:SEARch<x>:I2CBus[:SETup]:ADATa:
TYPE {BIT10address|BIT7ADdress|
BIT7APsub}
:SEARch<x>:I2CBus[:SETup]:ADATa:
TYPE?
```
<x> = 1、2

例
```
:SEARCH1:I2CBUS:SETUP:ADATA:
TYPE BIT10ADDRESS
:SEARCH1:I2CBUS:SETUP:ADATA:TYPE?
-> :SEARCH1:I2CBUS:SETUP:ADATA:
TYPE BIT10ADDRESS
```

:SEARch<x>:I2CBus[:SETup]:DATA?

機能　I²C バス信号サーチのデータに関するすべての設定値を問い合わせします。

構文
```
:SEARch<x>:I2CBus[:SETup]:DATA?
```
<x> = 1、2

:SEARch<x>:I2CBus[:SETup]:DATA:BYTE

機能　I²C バス信号サーチの設定データ数を設定 / 問い合わせします。

構文
```
:SEARch<x>:I2CBus[:SETup]:DATA:
BYTE {<NRf>}
:SEARch<x>:I2CBus[:SETup]:DATA:BYTE?
```
<x> = 1、2
<NRf> = 1 ～ 4

例
```
:SEARCH1:I2CBUS:SETUP:DATA:BYTE 1
:SEARCH1:I2CBUS:SETUP:DATA:BYTE?
-> :SEARCH1:I2CBUS:SETUP:DATA:BYTE 1
```

:SEARch<x>:I2CBus[:SETup]:DATA:CONDition

機能　I²C バス信号サーチのデータの判定方法 (一致 / 不一致) を設定 / 問い合わせします。

構文
```
:SEARch<x>:I2CBus[:SETup]:DATA:
CONDition {FALSe|TRUE}
:SEARch<x>:I2CBus[:SETup]:DATA:
CONDition?
```
<x> = 1、2

例
```
:SEARCH1:I2CBUS:SETUP:DATA:
CONDITION TRUE
:SEARCH1:I2CBUS:SETUP:DATA:
CONDITION?
-> :SEARCH1:I2CBUS:SETUP:DATA:
CONDITION TRUE
```

:SEARch<x>:I2CBus[:SETup]:DATA:DPOSition

機能　I²C バス信号サーチのデータのパターン比較する位置を設定 / 問い合わせします。

構文
```
:SEARch<x>:I2CBus[:SETup]:DATA:
DPOSition {<NRf>}
:SEARch<x>:I2CBus[:SETup]:DATA:
DPOSition?
```
<x> = 1、2
<NRf> = 0 ～ 9999

例
```
:SEARCH1:I2CBUS:SETUP:DATA:
DPOSITION 1
:SEARCH1:I2CBUS:SETUP:DATA:
DPOSITION?
-> :SEARCH1:I2CBUS:SETUP:DATA:
DPOSITION 1
```

:SEARch<x>:I2CBus[:SETup]:DATA:HEXA<x>

機能　I²C バス信号サーチのデータを HEXA で設定します。

構文
```
:SEARch<x>:I2CBus[:SETup]:DATA:
HEXA<x> {< 文字列 >}
```
SEARch<x> の <x> = 1、2
HEXA<x> の <x> = 1 ～ 4
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字

例
```
:SEARCH1:I2CBUS:SETUP:DATA:HEXA1
"AB"
```

:SEARch<x>:I2CBus[:SETup]:DATA:MODE

機能　I²C バス信号サーチのデータ条件の有効 / 無効を設定 / 問い合わせします。

構文
```
:SEARch<x>:I2CBus[:SETup]:DATA:
MODE {<Boolean>}
:SEARch<x>:I2CBus[:SETup]:DATA:MODE?
```
<x> = 1、2

例
```
:SEARCH1:I2CBUS:SETUP:DATA:MODE ON
:SEARCH1:I2CBUS:SETUP:DATA:MODE?
-> :SEARCH1:I2CBUS:SETUP:DATA:MODE 1
```

:SEARch<x>:I2CBus[:SETup]:DATA:PATTern<x>

機能　I²C バス信号サーチのデータを BINARY で設定 / 問い合わせします。

構文　:SEARch<x>:I2CBus[:SETup]:DATA:PATTern<x> {< 文字列 >}
:SEARch<x>:I2CBus[:SETup]:DATA:PATTern<x>?
SEARch<x> の <x> = 1、2
PATTern<x> の <x> = 1 ～ 4
< 文字列 > = '0'、'1'、'X' の組み合わせ 8 文字

例　:SEARCH1:I2CBUS:SETUP:DATA:PATTERN1 "10101011"
:SEARCH1:I2CBUS:SETUP:DATA:PATTERN1?
-> :SEARCH1:I2CBUS:SETUP:DATA:PATTERN1 "10101011"

:SEARch<x>:I2CBus[:SETup]:DATA:PMODe

機能　I²C バス信号サーチのデータのパターン比較先頭位置モードを設定 / 問い合わせします。

構文　:SEARch<x>:I2CBus[:SETup]:DATA:PMODe {DONTcare|SELect}
:SEARch<x>:I2CBus[:SETup]:DATA:PMODe?
<x> = 1、2

例　:SEARCH1:I2CBUS:SETUP:DATA:PMODE DONTCARE
:SEARCH1:I2CBUS:SETUP:DATA:PMODE?
-> :SEARCH1:I2CBUS:SETUP:DATA:PMODE DONTCARE

:SEARch<x>:I2CBus[:SETup]:DATA:TRACe

機能　I²C バス信号サーチのデータのトレースを設定 / 問い合わせします。

構文　:SEARch<x>:I2CBus[:SETup]:DATA:TRACe {<NRf>}
:SEARch<x>:I2CBus[:SETup]:DATA:TRACe?
<x> = 1、2

例　:SEARCH1:I2CBUS:SETUP:DATA:TRACE 1
:SEARCH1:I2CBUS:SETUP:DATA:TRACE?
-> :SEARCH1:I2CBUS:SETUP:DATA:TRACE 1

:SEARch<x>:I2CBus[:SETup]:GCALl?

機能　I²C バス信号サーチのジェネラルコールに関するすべての設定値を問い合わせします。

構文　:SEARch<x>:I2CBus[:SETup]:GCALl?
<x> = 1、2

:SEARch<x>:I2CBus[:SETup]:GCALl:BIT7maddress?

機能　I²C バス信号サーチのジェネラルコールの 7bit マスタアドレスに関するすべての設定値を問い合わせします。

構文　:SEARch<x>:I2CBus[:SETup]:GCALl:BIT7maddress?
<x> = 1、2

:SEARch<x>:I2CBus[:SETup]:GCALl:BIT7maddress:HEXA

機能　I²C バス信号サーチのジェネラルコールの 7bit マスタアドレスを HEXA で設定します。

構文　:SEARch<x>:I2CBus[:SETup]:GCALl:BIT7maddress:HEXA {< 文字列 >}
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字 (ビット 0 は、'1' に固定)

例　:SEARCH1:I2CBUS:SETUP:GCALL:BIT7MADDRESS:HEXA "BA"

:SEARch<x>:I2CBus[:SETup]:GCALl:BIT7maddress:PATTern

機能　I²C バス信号サーチのジェネラルコールの 7bit マスタアドレスを BINARY で設定 / 問い合わせします。

構文　:SEARch<x>:I2CBus[:SETup]:GCALl:BIT7maddress:PATTern {< 文字列 >}
:SEARch<x>:I2CBus[:SETup]:GCALl:BIT7maddress:PATTern?
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 7 文字

例　:SEARCH1:I2CBUS:SETUP:GCALL:BIT7MADDRESS:PATTERN "1010101"
:SEARCH1:I2CBUS:SETUP:GCALL:BIT7MADDRESS:PATTERN? -> :SEARCH1:I2CBUS:SETUP:GCALL:BIT7MADDRESS:PATTERN "1010101"

:SEARch<x>:I2CBus[:SETup]:GCALl:SBYTe (Second Byte)

機能　I²C バス信号サーチのジェネラルコールのセカンドバイトのタイプを設定 / 問い合わせします。

構文
```
:SEARch<x>:I2CBus[:SETup]:GCALl:
SBYTe {BIT7maddress|DONTcare|H04|H06
}
:SEARch<x>:I2CBus[:SETup]:GCALl:
SBYTe?
```
<x> = 1、2

例
```
:SEARCH1:I2CBUS:SETUP:GCALL:
SBYTE BIT7MADDRESS
:SEARCH1:I2CBUS:SETUP:GCALL:SBYTE?
-> :SEARCH1:I2CBUS:SETUP:GCALL:
SBYTE BIT7MADDRESS
```

:SEARch<x>:I2CBus[:SETup]:MODE

機能　I²C バス信号サーチのサーチモードを設定 / 問い合わせします。

構文
```
:SEARch<x>:I2CBus[:SETup]:
MODE {ADATa|ESTart|GCALl|NAIGnore|
SBHSmode}
:SEARch<x>:I2CBus[:SETup]:MODE?
```
<x> = 1、2

例
```
:SEARCH1:I2CBUS:SETUP:MODE ADATA
:SEARCH1:I2CBUS:SETUP:MODE?
-> :SEARCH1:I2CBUS:SETUP:MODE ADATA
```

:SEARch<x>:I2CBus[:SETup]:NAIGnore?

機能　I²C バス信号サーチの NON ACK 無視モードに関するすべての設定値を問い合わせします。

構文
```
:SEARch<x>:I2CBus[:SETup]:NAIGnore?
```
<x> = 1、2

:SEARch<x>:I2CBus[:SETup]:NAIGnore:HSMode

機能　I²C バス信号サーチのハイスピードモードで NON ACK を無視する / しないを設定 / 問い合わせします。

構文
```
:SEARch<x>:I2CBus[:SETup]:NAIGnore:
HSMode {<Boolean>}
:SEARch<x>:I2CBus[:SETup]:NAIGnore:
HSMode?
```
<x> = 1、2

例
```
:SEARCH1:I2CBUS:SETUP:NAIGNORE:
HSMODE ON
:SEARCH1:I2CBUS:SETUP:NAIGNORE:
HSMODE? -> :SEARCH1:I2CBUS:SETUP:
NAIGNORE:HSMODE 1
```

:SEARch<x>:I2CBus[:SETup]:NAIGnore:RACCess

機能　I²C バス信号サーチのリードアクセスモードで NON ACK を無視する / しないを設定 / 問い合わせします。

構文
```
:SEARch<x>:I2CBus[:SETup]:NAIGnore:
RACCess {<Boolean>}
:SEARch<x>:I2CBus[:SETup]:NAIGnore:
RACCess?
```
<x> = 1、2

例
```
:SEARCH1:I2CBUS:SETUP:NAIGNORE:
RACCESS ON
:SEARCH1:I2CBUS:SETUP:NAIGNORE:
RACCESS? -> :SEARCH1:I2CBUS:SETUP:
NAIGNORE:RACCESS 1
```

:SEARch<x>:I2CBus[:SETup]:NAIGnore:SBYTe (Start Byte)

機能　I²C バス信号サーチのスタートバイトで NON ACK を無視する / しないを設定 / 問い合わせします。

構文
```
:SEARch<x>:I2CBus[:SETup]:NAIGnore:
SBYTe {<Boolean>}
:SEARch<x>:I2CBus[:SETup]:NAIGnore:
SBYTe?
```
<x> = 1、2

例
```
:SEARCH1:I2CBUS:SETUP:NAIGNORE:
SBYTE ON
:SEARCH1:I2CBUS:SETUP:NAIGNORE:
SBYTE?
-> :SEARCH1:I2CBUS:SETUP:NAIGNORE:
SBYTE 1
```

:SEARch<x>:I2CBus[:SETup]:SBHSmode?

機能　I²C バス信号サーチのスタートバイト / ハイスピードモードに関するすべての設定値を問い合わせします。

構文
```
:SEARch<x>:I2CBus[:SETup]:SBHSmode?
```
<x> = 1、2

:SEARch<x>:I2CBus[:SETup]:SBHSmode:TYPE

機能　I²C バス信号サーチのスタートバイト / ハイスピードモードのタイプを設定 / 問い合わせします。

構文
```
:SEARch<x>:I2CBus[:SETup]:SBHSmode:
TYPE {HSMode|SBYTe}
:SEARch<x>:I2CBus[:SETup]:SBHSmode:
TYPE?
```
<x> = 1、2

例
```
:SEARCH1:I2CBUS:SETUP:SBHSMODE:
TYPE HSMODE
:SEARCH1:I2CBUS:SETUP:SBHSMODE:TYPE?
-> :SEARCH1:I2CBUS:SETUP:SBHSMODE:
TYPE HSMODE
```

:SEARch<x>:LINBus?

機能　LIN バス信号サーチに関するすべての設定値を問い合わせします。

構文　:SEARch<x>:LINBus?
<x> = 1、2

:SEARch<x>:LINBus[:SETup]?

機能　LIN バス信号サーチのセットアップに関するすべての設定値を問い合わせします。

構文　:SEARch<x>:LINBus[:SETup]?
<x> = 1、2

:SEARch<x>:LINBus[:SETup]:BLENgth

機能　LIN バス信号サーチの Break length を設定 / 問い合わせします。

構文　:SEARch<x>:LINBus[:SETup]:
BLENgth {<NRf>}
:SEARch<x>:LINBus[:SETup]:BLENgth?
<x> = 1、2
<NRf> = 10 ～ 13

例　:SEARCH1:LINBUS:SETUP:BLENGTH 10
:SEARCH1:LINBUS:SETUP:BLENGTH?
-> :SEARCH1:LINBUS:SETUP:BLENGTH 10

:SEARch<x>:LINBus[:SETup]:BRATe

機能　LIN バス信号サーチのビットレート (データ転送速度) を設定 / 問い合わせします。

構文　:SEARch<x>:LINBus[:SETup]:
BRATe {<NRf>|USER,<NRf>}
:SEARch<x>:LINBus[:SETup]:BRATe?
<x> = 1、2
<NRf> = 1200、2400、4800、9600、19200
USER の <NRf> = 5.4 節参照。

例　:SEARCH1:LINBUS:SETUP:BRATE 19200
:SEARCH1:LINBUS:SETUP:BRATE?
-> :SEARCH1:LINBUS:SETUP:BRATE 19200

:SEARch<x>:LINBus[:SETup]:DATA?

機能　LIN バス信号サーチのデータに関するすべての設定値を問い合わせします。

構文　:SEARch<x>:LINBus[:SETup]:DATA?
<x> = 1、2

:SEARch<x>:LINBus[:SETup]:DATA:BNUM

機能　LIN バス信号サーチのデータのバイト数を設定 / 問い合わせします。

構文　:SEARch<x>:LINBus[:SETup]:DATA:BNUM
{<NRf>}
:SEARch<x>:LINBus[:SETup]:DATA:BNUM?
<x> = 1、2
<NRf> = 1 ～ 8

例　:SEARCH1:LINBUS:SETUP:DATA:BNUM 1
:SEARCH1:LINBUS:SETUP:DATA:BNUM?
-> :SEARCH1:LINBUS:SETUP:DATA:BNUM 1

SEARch<x>:LINBus[:SETup]:DATA:BORDer

機能　LIN バス信号サーチのデータのバイトオーダを設定 / 問い合わせします。

構文　:SEARch<x>:LINBus[:SETup]:DATA:
BORDer {BIG|LITTle}
:SEARch<x>:LINBus[:SETup]:DATA:
BORDer?
<x> = 1、2

例　:SEARCH1:LINBUS:SETUP:DATA:BORDER
BIG
:SEARCH1:LINBUS:SETUP:DATA:BORDER?
-> :SEARCH1:LINBUS:SETUP:DATA:
BORDER BIG

:SEARch<x>:LINBus[:SETup]:DATA:CONDition

機能　LIN バス信号サーチのデータ条件を設定 / 問い合わせします。

構文　:SEARch<x>:LINBus[:SETup]:DATA:
CONDition {BETWeen|DONTcare|FALSe|
GTHan|LTHan|ORANge|TRUE}
:SEARch<x>:LINBus[:SETup]:DATA:
CONDition?
<x> = 1、2

例　:SEARCH1:LINBUS:SETUP:DATA:
CONDITION BETWEEN
:SEARCH1:LINBUS:SETUP:DATA:
CONDITION?
-> :SEARCH1:LINBUS:SETUP:DATA:
CONDITION BETWEEN

:SEARch<x>:LINBus[:SETup]:DATA:DATA<x>

機能　LIN バス信号サーチのデータの比較データを設定 / 問い合わせします。

構文
```
:SEARch<x>:LINBus[:SETup]:DATA:
DATA<x> {<NRf>}
:SEARch<x>:LINBus[:SETup]:DATA:
DATA<x>?
```
SEARch<x> の <x> = 1、2
DATA<x> の <x> = 1、2
<NRf> = 5.4 節参照。

例
```
:SEARCH1:LINBUS:SETUP:DATA:DATA1 1
:SEARCH1:LINBUS:SETUP:DATA:DATA1?
-> :SEARCH1:LINBUS:SETUP:DATA:
DATA1 1.0000000E+00
```

解説
- 「:SEARch<x>:LINBus[:SETup]:DATA:CONDition GTHan」のときは「:SEARch<x>:LINBus[:SETup]:DATA:DATA1」で設定します。
- 「:SEARch<x>:LINBus[:SETup]:DATA:CONDition LTHan」のときは「:SEARch<x>:LINBus[:SETup]:DATA:DATA2」で設定します。
- 「:SEARch<x>:LINBus[:SETup]:DATA:CONDition BETWeen|ORANge」のときは、小さい値を「:SEARch<x>:LINBus[:SETup]:DATA:DATA1」、大きい値を「:SEARch<x>:LINBus[:SETup]:DATA:DATA2」で設定します。

:SEARch<x>:LINBus[:SETup]:DATA:HEXA

機能　LIN バス信号サーチのデータを HEXA で設定します。

構文
```
:SEARch<x>:LINBus[:SETup]:DATA:HEXA
{< 文字列 >}
```
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 16 文字以内（BNUM の設定値で変化します）

例
```
:SEARCH1:LINBUS:SETUP:DATA:HEXA "3B"
```

:SEARch<x>:LINBus[:SETup]:DATA:MSBLsb

機能　LIN バス信号サーチのデータの MSB/LSB のビットを設定 / 問い合わせします。

構文
```
:SEARch<x>:LINBus[:SETup]:DATA:
MSBLsb {<NRf>,<NRf>}
:SEARch<x>:LINBus[:SETup]:DATA:
MSBLsb?
```
<x> = 1、2
<NRf> = 5.4 節参照。

例
```
:SEARCH1:LINBUS:SETUP:DATA:MSBLSB
1,0
:SEARCH1:LINBUS:SETUP:DATA:MSBLSB?
-> :SEARCH1:LINBUS:SETUP:DATA:MSBLSB
1,0
```

:SEARch<x>:LINBus[:SETup]:DATA:PATTern

機能　LIN バス信号サーチのデータを BINARY で設定 / 問い合わせします。

構文
```
:SEARch<x>:LINBus[:SETup]:DATA:
PATTern {< 文字列 >}
:SEARch<x>:LINBus[:SETup]:DATA:
PATTern?
```
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 64 文字以内（BNUM の設定値で変化します）

例
```
:SEARCH1:LINBUS:SETUP:DATA:
PATTERN "11011111"
:SEARCH1:LINBUS:SETUP:DATA:PATTERN?
-> :SEARCH1:LINBUS:SETUP:DATA:
PATTERN "11011111"
```

:SEARch<x>:LINBus[:SETup]:DATA:SIGN

機能　LIN バス信号サーチのデータの符号を設定 / 問い合わせします。

構文
```
:SEARch<x>:LINBus[:SETup]:DATA:
SIGN {SIGN|UNSign}
:SEARch<x>:LINBus[:SETup]:DATA:SIGN?
```
<x> = 1、2

例
```
:SEARCH1:LINBUS:SETUP:DATA:SIGN SIGN
:SEARCH1:LINBUS:SETUP:DATA:SIGN?
-> :SEARCH1:LINBUS:SETUP:DATA:SIGN
SIGN
```

:SEARch<x>:LINBus[:SETup]:ERRor?

機能　LIN バス信号サーチの Error に関するすべてのを設定値を問い合わせします。

構文
```
:SEARch<x>:LINBus[:SETup]:ERRor?
```
<x> = 1、2

:SEARch<x>:LINBus[:SETup]:ERRor:CHECksum

機能　LIN バス信号サーチの Checksum Error を設定 / 問い合わせします。

構文
```
:SEARch<x>:LINBus[:SETup]:ERRor:
CHECksum {<Boolean>}
:SEARch<x>:LINBus[:SETup]:ERRor:
CHECksum?
```
<x> = 1、2

例
```
:SEARCH1:LINBUS:SETUP:ERROR:CHECKSUM
ON
:SEARCH1:LINBUS:SETUP:ERROR:
CHECKSUM?
-> :SEARCH1:LINBUS:SETUP:ERROR:
CHECKSUM 1
```

:SEARch<x>:LINBus[:SETup]:ERRor:FRAMing

機能　LIN バス信号サーチの Framing Error を設定 / 問い合わせします。

構文
```
:SEARch<x>:LINBus[:SETup]:ERRor:
FRAMing {<Boolean>}
:SEARch<x>:LINBus[:SETup]:ERRor:
FRAMing?
```
<x> = 1、2

例
```
:SEARCH1:LINBUS:SETUP:ERROR:
FRAMING ON
:SEARCH1:LINBUS:SETUP:ERROR:FRAMING?
-> :SEARCH1:LINBUS:SETUP:ERROR:
FRAMING 1
```

:SEARch<x>:LINBus[:SETup]:ERRor:PARity

機能　LIN バス信号サーチの Parity Error を設定 / 問い合わせします。

構文
```
:SEARch<x>:LINBus[:SETup]:ERRor:
PARity {<Boolean>}
:SEARch<x>:LINBus[:SETup]:ERRor:
PARity?
```
<x> = 1、2

例
```
:SEARCH1:LINBUS:SETUP:ERROR:
PARITY ON
:SEARCH1:LINBUS:SETUP:ERROR:PARITY?
-> :SEARCH1:LINBUS:SETUP:ERROR:
PARITY 1
```

:SEARch<x>:LINBus[:SETup]:ERRor:SYNCh

機能　LIN バス信号サーチの Synch Error を設定 / 問い合わせします。

構文
```
:SEARch<x>:LINBus[:SETup]:ERRor:
SYNCh {<Boolean>}
:SEARch<x>:LINBus[:SETup]:ERRor:
SYNCh?
```
<x> = 1、2

例
```
:SEARCH1:LINBUS:SETUP:ERROR:SYNCH ON
:SEARCH1:LINBUS:SETUP:ERROR:SYNCH?
-> :SEARCH1:LINBUS:SETUP:ERROR:
SYNCH 1
```

:SEARch<x>:LINBus[:SETup]:ERRor:TOUT

機能　LIN バス信号サーチの Timeout Error を設定 / 問い合わせします。

構文
```
:SEARch<x>:LINBus[:SETup]:ERRor:
TOUT {<Boolean>}
:SEARch<x>:LINBus[:SETup]:ERRor:
TOUT?
```
<x> = 1、2

例
```
:SEARCH1:LINBUS:SETUP:ERROR:TOUT ON
:SEARCH1:LINBUS:SETUP:ERROR:TOUT?
-> :SEARCH1:LINBUS:SETUP:ERROR:TOUT
1
```

:SEARch<x>:LINBus[:SETup]:ID?

機能　LIN バス信号サーチの ID に関するすべての設定値を問い合わせします。

構文
```
:SEARch<x>:LINBus[:SETup]:ID?
```
<x> = 1、2

:SEARch<x>:LINBus[:SETup]:ID:HEXA

機能　LIN バス信号サーチの ID を HEXA で設定します。

構文
```
:SEARch<x>:LINBus[:SETup]:ID:HEXA
{< 文字列 >}
```
<x> = 1、2

< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字

例
```
:SEARCH1:LINBUS:SETUP:ID:HEXA "2A"
```

:SEARch<x>:LINBus[:SETup]:ID:PATTern

機能　LIN バス信号サーチの ID を BINARY で設定 / 問い合わせします。

構文
```
:SEARch<x>:LINBus[:SETup]:ID:PATTern
{< 文字列 >}
:SEARch<x>:LINBus[:SETup]:ID:
PATTern?
```
<x> = 1、2

< 文字列 > = '0'、'1'、'X' の組み合わせ 6 文字

例
```
:SEARCH1:LINBUS:SETUP:ID:
PATTERN "101111"
-:SEARCH1:LINBUS:SETUP:ID:PATTERN?
-> :SEARCH1:LINBUS:SETUP:ID:
PATTERN "101111"
```

:SEARch<x>:LINBus[:SETup]:MODE
機能　LIN バス信号サーチのモードを設定 / 問い合わせします。
構文　:SEARch<x>:LINBus[:SETup]:MODE {IDData|SYNCh}
:SEARch<x>:LINBus[:SETup]:MODE?
<x> = 1、2
例　:SEARCH1:LINBUS:SETUP:MODE IDDATA
:SEARCH1:LINBUS:SETUP:MODE?
-> :SEARCH1:LINBUS:SETUP:MODE IDDATA

:SEARch<x>:LINBus[:SETup]:REVision
機能　LIN バス信号サーチのレビジョン (1.3 or 2.0 or Both) を設定 / 問い合わせします。
構文　:SEARch<x>:LINBus[:SETup]:REVision {BOTH|LIN1_3|LIN2_0}
:SEARch<x>:LINBus[:SETup]:REVision?
<x> = 1、2
例　:SEARCH1:LINBUS:SETUP:REVISION LIN1_3
:SEARCH1:LINBUS:SETUP:REVISION?
-> :SEARCH1:LINBUS:SETUP:REVISION LIN1_3

:SEARch<x>:LINBus[:SETup]:SPOint
機能　LIN バス信号サーチのサンプルポイントを設定 / 問い合わせします。
構文　:SEARch<x>:LINBus[:SETup]:SPOint {<NRf>}
:SEARch<x>:LINBus[:SETup]:SPOint?
<x> = 1、2
<NRf> = 18.8 ～ 90.6(%)
例　:SEARCH1:LINBUS:SETUP:SPOINT 18.8
:SEARCH1:LINBUS:SETUP:SPOINT?
-> :SEARCH1:LINBUS:SETUP:SPOINT 18.8E+00

:SEARch<x>:LINBus[:SETup]:TRACe
機能　LIN バス信号サーチのトレースを設定 / 問い合わせします。
構文　:SEARch<x>:LINBus[:SETup]:TRACe {<NRf>}
:SEARch<x>:LINBus[:SETup]:TRACe?
<x> = 1、2
<NRf> = 1 ～ 8
例　:SEARCH1:LINBUS:SETUP:TRACE 1
:SEARCH1:LINBUS:SETUP:TRACE?
-> :SEARCH1:LINBUS:SETUP:TRACE 1

:SEARch<x>:SLOGic:I2CBus?
機能　ロジック I²C バス信号サーチに関するすべての設定値を問い合わせます。
構文　:SEARch<x>:SLOGic:I2CBus?
<x> = 1、2

:SEARch<x>:SLOGic:I2CBus:CLOCk?
機能　ロジック I²C バス信号サーチのクロックチャネルに関するすべての設定値を問い合わせます。
構文　:SEARch<x>:SLOGic:I2CBus:CLOCk?
<x> = 1、2
例　:SEARCH1:SLOGIC:I2CBUS:CLOCK? -> :SEARCH1:SLOGIC:I2CBUS:CLOCK:SOURCE A0

:SEARch<x>:SLOGic:I2CBus:CLOCk:SOURce
機能　ロジック I²C バス信号サーチのクロックチャネルを設定 / 問い合わせします。
構文　:SEARch<x>:SLOGic:I2CBus:CLOCk:SOURce {A<y>|B<y>|C<y>|D<y>}
:SEARch<x>:SLOGic:I2CBus:CLOCk:SOURce?
<x> = 1、2
<y> = 0 ～ 7
例　:SEARCH1:SLOGIC:I2CBUS:CLOCK:SOURCE A0
:SEARCH1:SLOGIC:I2CBUS:CLOCK:SOURCE? -> :SEARCH1:SLOGIC:I2CBUS:CLOCK:SOURCE A0
解説　DLM6000 の 16 ビットモデルでは {A<x>|C<x>} が有効です。

:SEARch<x>:SLOGic:I2CBus[:SETup]?
機能　ロジック I²C バス信号サーチのセットアップに関するすべての設定値を問い合わせます。
構文　:SEARch<x>:SLOGic:I2CBus[:SETup]?
<x> = 1、2

:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa?
機能　ロジック I²C バス信号サーチのアドレスに関するすべての設定値を問い合わせます。
構文　:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa?
<x> = 1、2

:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:BIT10address?
機能　ロジック I²C バス信号サーチの 10bit アドレスに関するすべての設定値を問い合わせます。
構文　:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:BIT10address?
<x> = 1、2

:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:BIT10address:HEXA

機能　ロジック I²C バス信号サーチの 10bit アドレスを HEXA で設定します。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
ADATa:BIT10address:HEXA {< 文字列 >}
```
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 3 文字 (ビット 8 は、R/$\overline{W}$ ビット)

例
```
:SEARCH1:SLOGIC:I2CBUS:SETUP:ADATA:
BIT10ADDRESS:HEXA "5DF"。
```

:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:BIT10address:PATTern

機能　ロジック I²C バス信号サーチの 10bit アドレスを BINARY で設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
ADATa:BIT10address:PATTern {< 文字列 >}
:SEARch<x>:SLOGic:I2CBus[:SETup]:
ADATa:BIT10address:PATTern?
```
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 11 文字 (ビット 8 は、R/$\overline{W}$ ビット)

例
```
:SEARCH1:SLOGIC:I2CBUS:SETUP:ADATA:
BIT10ADDRESS:PATTERN "10111011111"
:SEARCH1:SLOGIC:I2CBUS:SETUP:ADATA:
BIT10ADDRESS:PATTERN? -> :SEARCH1:
SLOGIC:I2CBUS:SETUP:ADATA:
BIT10ADDRESS:PATTERN "10111011111"
```

:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:BIT7ADdress?

機能　ロジック I²C バス信号サーチの 7bit アドレスに関するすべての設定値を問い合わせます。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
ADATa:BIT7ADdress?
```
<x> = 1、2

:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:BIT7ADdress:HEXA

機能　ロジック I²C バス信号サーチの 7bit アドレスを HEXA で設定します。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
ADATa:BIT7ADdress:HEXA {< 文字列 >}
```
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字 (ビット 0 は、R/$\overline{W}$ ビット)

例
```
:SEARCH1:SLOGIC:I2CBUS:SETUP:ADATA:
BIT7ADDRESS:HEXA "DE"
```

:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:BIT7ADdress:PATTern

機能　ロジック I²C バス信号サーチの 7bit アドレスを BINARY で設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
ADATa:BIT7ADdress:PATTern {< 文字列 >}
:SEARch<x>:SLOGic:I2CBus[:SETup]:
ADATa:BIT7ADdress:PATTern?
```
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 8 文字 (ビット 0 は、R/$\overline{W}$ ビット)

例
```
:SEARCH1:SLOGIC:I2CBUS:SETUP:ADATA:
BIT7ADDRESS:PATTERN "11011110"
:SEARCH1:SLOGIC:I2CBUS:SETUP:ADATA:
BIT7ADDRESS:PATTERN? -> :SEARCH1:
SLOGIC:I2CBUS:SETUP:ADATA:
BIT7ADDRESS:PATTERN "11011110"。
```

:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:BIT7APsub?

機能　ロジック I²C バス信号サーチの 7bit+Sub アドレスに関するすべての設定値を問い合わせます。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
ADATa:
BIT7APsub?
```
<x> = 1、2

:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:BIT7APsub:ADDRess?

機能　ロジック I²C バス信号サーチの 7bit+Sub アドレスの 7bit アドレスに関するすべての設定値を問い合わせます。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
ADATa:BIT7APsub:ADDRess?
```
<x> = 1、2

:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:BIT7APsub:ADDRess:HEXA

機能　ロジック I²C バス信号サーチの 7bit+Sub アドレスの 7bit アドレスを HEXA で設定します。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
ADATa:BIT7APsub:ADDRess:
HEXA {< 文字列 >}
```
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字 (ビット 0 は、R/$\overline{W}$ ビット)

例
```
:SEARCH1:SLOGIC:I2CBUS:SETUP:ADATA:
BIT7APSUB:ADDRESS:HEXA "CD"
```

:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:BIT7APsub:ADDRess:PATTern

機能　ロジック I²C バス信号サーチの 7bit+Sub アドレスの 7bit アドレスを BINARY で設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
ADATa:BIT7APsub:ADDRess:
PATTern {< 文字列 >}
:SEARch<x>:SLOGic:I2CBus[:SETup]:
ADATa:BIT7APsub:ADDRess:PATTern?
```
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 8 文字 (ビット 0 は、R/$\overline{W}$ ビット)

例
```
:SEARCH1:SLOGIC:I2CBUS:SETUP:ADATA:
BIT7APSUB:ADDRESS:PATTERN "11001101"
:SEARCH1:SLOGIC:I2CBUS:SETUP:ADATA:
BIT7APSUB:ADDRESS:PATTERN?
-> :SEARCH1:SLOGIC:I2CBUS:SETUP:
ADATA:BIT7APSUB:ADDRESS:
PATTERN "11001101"
```

:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:BIT7APsub:SADDress?

機能　ロジック I²C バス信号サーチの 7bit+Sub アドレスの Sub アドレスに関するすべての設定値を問い合わせます。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
ADATa:BIT7APsub:SADDress?
```
<x> = 1、2

:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:BIT7APsub:SADDress:HEXA

機能　ロジック I²C バス信号サーチの 7bit+Sub アドレスの Sub アドレスを HEXA で設定します。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
ADATa:BIT7APsub:SADDress:
HEXA {< 文字列 >}
```
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字

例
```
:SEARCH1:SLOGIC:I2CBUS:SETUP:ADATA:
BIT7APSUB:SADDRESS:HEXA "EF"
```

:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:BIT7APsub:SADDress:PATTern

機能　ロジック I²C バス信号サーチの 7bit+Sub アドレスの Sub アドレスを BINARY で設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
ADATa:BIT7APsub:SADDress:
PATTern {< 文字列 >}
:SEARch<x>:SLOGic:I2CBus[:SETup]:
ADATa:
BIT7APsub:SADDress:PATTern?
```
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 8 文字

例
```
:SEARCH1:SLOGIC:I2CBUS:SETUP:ADATA:
BIT7APSUB:SADDRESS:PATTERN
"11101111"
:SEARCH1:SLOGIC:I2CBUS:SETUP:ADATA:
BIT7APSUB:SADDRESS:PATTERN?
-> :SEARCH1:SLOGIC:I2CBUS:SETUP:
ADATA:BIT7APSUB:SADDRESS:
PATTERN "11101111"
```

:SEARch<x>:SLOGic:I2CBus[:SETup]:ADATa:TYPE

機能　ロジック I²C バス信号サーチのアドレスの種類を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
ADATa:TYPE {BIT10address|BIT7ADdress
|BIT7APsub}
:SEARch<x>:SLOGic:I2CBus[:SETup]:
ADATa:TYPE?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:I2CBUS:SETUP:ADATA:
TYPE BIT10ADDRESS
:SEARCH1:SLOGIC:I2CBUS:SETUP:ADATA:
TYPE? -> :SEARCH1:SLOGIC:I2CBUS:
SETUP:ADATA:TYPE BIT10ADDRESS
```

:SEARch<x>:SLOGic:I2CBus[:SETup]:DATA?

機能　ロジック I²C バス信号サーチのデータに関するすべての設定値を問い合わせます。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
DATA?
```
<x> = 1、2

:SEARch<x>:SLOGic:I2CBus[:SETup]:DATA:BYTE

機能　ロジック I²C バス信号サーチの設定データ数を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
DATA:BYTE {<NRf>}
:SEARch<x>:SLOGic:I2CBus[:SETup]:
DATA:BYTE?
```
<x> = 1、2
<NRf> = 1 ～ 4

例
```
:SEARCH1:SLOGIC:I2CBUS:SETUP:DATA:
BYTE 1
:SEARCH1:SLOGIC:I2CBUS:SETUP:DATA:
BYTE? -> :SEARCH1:SLOGIC:I2CBUS:
SETUP:DATA:BYTE 1
```

:SEARch<x>:SLOGic:I2CBus[:SETup]:DATA:CONDition

機能　ロジック I²C バス信号サーチのデータの判定方法 (一致 / 不一致) を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
DATA:CONDition {FALSe|TRUE}
:SEARch<x>:SLOGic:I2CBus[:SETup]:
DATA:CONDition?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:I2CBUS:SETUP:DATA:
CONDITION FALSE
:SEARCH1:SLOGIC:I2CBUS:SETUP:DATA:
CONDITION? -> :SEARCH1:SLOGIC:
I2CBUS:
SETUP:DATA:CONDITION FALSE
```

:SEARch<x>:SLOGic:I2CBus[:SETup]:DATA:DPOSition

機能　ロジック I²C バス信号サーチのデータのパターン比較する位置を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
DATA:DPOSition {<NRf>}
:SEARch<x>:SLOGic:I2CBus[:SETup]:
DATA:DPOSition?
```
<x> = 1、2
<NRf> = 0 ～ 9999

例
```
:SEARCH1:SLOGIC:I2CBUS:SETUP:DATA:
DPOSITION 1
:SEARCH1:SLOGIC:I2CBUS:SETUP:DATA:
DPOSITION? -> :SEARCH1:SLOGIC:
I2CBUS:
SETUP:DATA:DPOSITION 1
```

:SEARch<x>:SLOGic:I2CBus[:SETup]:DATA:HEXA<x>

機能　ロジック I²C バス信号サーチのデータを HEXA で設定します。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
DATA:
HEXA<x> {< 文字列 >}
```
SEARch<x> の <x> = 1、2
HEXA<x> の <x> = 1 ～ 4
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字

例
```
:SEARCH1:SLOGIC:I2CBUS:SETUP:DATA:
HEXA1 "AB"
```

:SEARch<x>:SLOGic:I2CBus[:SETup]:DATA:MODE

機能　ロジック I²C バス信号サーチのデータ条件の有効 / 無効を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
DATA:MODE {<Boolean>}
:SEARch<x>:SLOGic:I2CBus[:SETup]:
DATA:MODE?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:I2CBUS:SETUP:DATA:
MODE ON
:SEARCH1:SLOGIC:I2CBUS:SETUP:DATA:
MODE? -> :SEARCH1:SLOGIC:I2CBUS:
SETUP:DATA:MODE 1
```

:SEARch<x>:SLOGic:I2CBus[:SETup]:DATA:PATTern<x>

機能　ロジック I²C バス信号サーチのデータを BINARY で設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
DATA:PATTern<x> {< 文字列 >}
:SEARch<x>:SLOGic:I2CBus[:SETup]:
DATA:PATTern<x>?
```
SEARch<x> の <x> = 1、2
PATTern<x> の <x> = 1 ～ 4
< 文字列 > = '0'、'1'、'X' の組み合わせ 8 文字

例
```
:SEARCH1:SLOGIC:I2CBUS:SETUP:DATA:
PATTERN1 "10101011"
:SEARCH1:SLOGIC:I2CBUS:SETUP:DATA:
PATTERN1? -> :SEARCH1:SLOGIC:I2CBUS:
SETUP:DATA:PATTERN1 "10101011"
```

:SEARch<x>:SLOGic:I2CBus[:SETup]:DATA:PMODe

機能　ロジック I²C バス信号サーチのデータのパターン比較先頭位置モードを設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
DATA:PMODe {DONTcare|SELect}
:SEARch<x>:SLOGic:I2CBus[:SETup]:
DATA:PMODe?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:I2CBUS:SETUP:DATA:
PMODE DONTCARE
:SEARCH1:SLOGIC:I2CBUS:SETUP:DATA:
PMODE? -> :SEARCH1:SLOGIC:I2CBUS:
SETUP:DATA:PMODE DONTCARE
```

:SEARch<x>:SLOGic:I2CBus[:SETup]:DATA:TRACe

機能　ロジック I²C バス信号サーチのデータのトレースを設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
DATA:TRACe {A<y>|B<y>|C<y>|D<y>}
:SEARch<x>:SLOGic:I2CBus[:SETup]:
DATA:TRACe?
```
<x> = 1、2
<y> = 0 ～ 7

例
```
:SEARCH1:SLOGIC:I2CBUS:SETUP:DATA:
TRACE A0
:SEARCH1:SLOGIC:I2CBUS:SETUP:DATA:
TRACE? -> :SEARCH1:SLOGIC:I2CBUS:
SETUP:DATA:TRACE A0
```

解説　DLM6000 の 16 ビットモデルでは {A<x>|C<x>} が有効です。

:SEARch<x>:SLOGic:I2CBus[:SETup]:GCALl?

機能　ロジック I²C バス信号サーチのジェネラルコールに関するすべての設定値を問い合わせます。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
GCALl?
```
<x> = 1、2。

:SEARch<x>:SLOGic:I2CBus[:SETup]:GCALl:BIT7maddress?

機能　ロジック I²C バス信号サーチのジェネラルコールの 7bit マスタアドレスに関するすべての設定値を問い合わせます。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
GCALl:BIT7maddress?
```
<x> = 1、2

:SEARch<x>:SLOGic:I2CBus[:SETup]:GCALl:BIT7maddress:HEXA

機能　ロジック I²C バス信号サーチのジェネラルコールの 7bit マスタアドレスを HEXA で設定します。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
GCALl:BIT7maddress:HEXA {< 文字列 >}
```
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字 (ビット 0 は、'1' に固定)

例
```
:SEARCH1:SLOGIC:I2CBUS:SETUP:GCALL:
BIT7MADDRESS:HEXA "BA"
```

:SEARch<x>:SLOGic:I2CBus[:SETup]:GCALl:BIT7maddress:PATTern

機能　ロジック I²C バス信号サーチのジェネラルコールの 7bit マスタアドレスを BINARY で設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
GCALl:BIT7maddress:PATTern {<文字列>}
:SEARch<x>:SLOGic:I2CBus[:SETup]:
GCALl:BIT7maddress:PATTern?
```
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 7 文字

例
```
:SEARCH1:SLOGIC:I2CBUS:SETUP:GCALL:
BIT7MADDRESS:PATTERN "1010101"
:SEARCH1:SLOGIC:I2CBUS:SETUP:GCALL:
BIT7MADDRESS:PATTERN? -> :SEARCH1:
SLOGIC:I2CBUS:SETUP:GCALL:
BIT7MADDRESS:PATTERN "1010101"
```

:SEARch<x>:SLOGic:I2CBus[:SETup]:GCALl:SBYTe (Second Byte)

機能　ロジック I²C バス信号サーチのジェネラルコールのセカンドバイトのタイプを設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
GCALl:SBYTe {BIT7maddress|DONTcare|
H04|H06}
:SEARch<x>:SLOGic:I2CBus[:SETup]:
GCALl:SBYTe?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:I2CBUS:SETUP:GCALL:
SBYTE BIT7MADDRESS
:SEARCH1:SLOGIC:I2CBUS:SETUP:GCALL:
SBYTE? -> :SEARCH1:SLOGIC:I2CBUS:
SETUP:GCALL:SBYTE BIT7MADDRESS
```

:SEARch<x>:SLOGic:I2CBus[:SETup]:MODE

機能　ロジック I²C バス信号サーチのサーチモードを設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
MODE {ADATa|ESTart|GCALl|NAIGnore|
SBHSmode}
:SEARch<x>:SLOGic:I2CBus[:SETup]:
MODE?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:I2CBUS:SETUP:MODE
ADATA
:SEARCH1:SLOGIC:I2CBUS:SETUP:
MODE? -> :SEARCH1:SLOGIC:I2CBUS:
SETUP:MODE ADATA
```

:SEARch<x>:SLOGic:I2CBus[:SETup]:NAIGnore?

機能　ロジック I²C バス信号サーチの NON ACK 無視モードに関するすべての設定値を問い合わせます。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
NAIGnore?
```
<x> = 1、2。

:SEARch<x>:SLOGic:I2CBus[:SETup]:NAIGnore:HSMode

機能　ロジック I²C バス信号サーチのハイスピードモードで NON ACK を無視する / しないを設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
NAIGnore:HSMode {<Boolean>}
:SEARch<x>:SLOGic:I2CBus[:SETup]:
NAIGnore:HSMode?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:I2CBUS:SETUP:
NAIGNORE:HSMODE ON
:SEARCH1:SLOGIC:I2CBUS:SETUP:
NAIGNORE:HSMODE? -> :SEARCH1:SLOGIC:
I2CBUS:SETUP:NAIGNORE:HSMODE 1
```

:SEARch<x>:SLOGic:I2CBus[:SETup]:NAIGnore:RACCess

機能　ロジック I²C バス信号サーチのリードアクセスモードで NON ACK を無視する / しないを設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
NAIGnore:RACCess {<Boolean>}
:SEARch<x>:SLOGic:I2CBus[:SETup]:
NAIGnore:RACCess?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:I2CBUS:SETUP:
NAIGNORE:RACCESS ON
:SEARCH1:SLOGIC:I2CBUS:SETUP:
NAIGNORE:RACCESS? -> :SEARCH1:
SLOGIC:I2CBUS:SETUP:NAIGNORE:RACCESS
1
```

:SEARch<x>:SLOGic:I2CBus[:SETup]:NAIGnore:SBYTe (Start Byte)

機能　ロジック I²C バス信号サーチのスタートバイトで NON ACK を無視する / しないを設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
NAIGnore:SBYTe {<Boolean>}
:SEARch<x>:SLOGic:I2CBus[:SETup]:
NAIGnore:SBYTe?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:I2CBUS:SETUP:
NAIGNORE:SBYTE ON
:SEARCH1:SLOGIC:I2CBUS:SETUP:
NAIGNORE:SBYTE? -> :SEARCH1:SLOGIC:
I2CBUS:SETUP:NAIGNORE:SBYTE 1
```

:SEARch<x>:SLOGic:I2CBus[:SETup]:SBHSmode?

機能　ロジック I²C バス信号サーチのスタートバイト / ハイスピードモードに関するすべての設定値を問い合わせます。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
SBHSmode?
```
<x> = 1、2

:SEARch<x>:SLOGic:I2CBus[:SETup]:SBHSmode:TYPE

機能　ロジック I²C バス信号サーチのスタートバイト / ハイスピードモードのタイプを設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:I2CBus[:SETup]:
SBHSmode:TYPE {HSMode|SBYTe}
:SEARch<x>:SLOGic:I2CBus[:SETup]:
SBHSmode:TYPE?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:I2CBUS:SETUP:
SBHSMODE:TYPE HSMODE
:SEARCH1:SLOGIC:I2CBUS:SETUP:
SBHSMODE:TYPE? -> :SEARCH1:SLOGIC:
I2CBUS:SETUP:SBHSMODE:TYPE HSMODE
```

:SEARch<x>:SLOGic:LINBus?

機能　ロジック LIN バス信号サーチに関するすべての設定値を問い合わせます。

構文
```
:SEARch<x>:SLOGic:LINBus?
```
<x> = 1、2

:SEARch<x>:SLOGic:LINBus[:SETup]?

機能　ロジック LIN バス信号サーチのセットアップに関するすべての設定値を問い合わせます。

構文
```
:SEARch<x>:SLOGic:LINBus[:SETup]?
```
<x> = 1、2

SEARch<x>:SLOGic:LINBus[:SETup]:BLENgth

機能　ロジック LIN バス信号サーチの Break length を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:LINBus[:SETup]:
BLENgth {<NRf>}
:SEARch<x>:SLOGic:LINBus[:SETup]:
BLENgth?
```
<x> = 1、2
<NRf> = 10 ～ 13

例
```
:SEARCH1:SLOGIC:LINBUS:SETUP:
BLENGTH 10
:SEARCH1:SLOGIC:LINBUS:SETUP:
BLENGTH?
 -> :SEARCH1:SLOGIC:LINBUS:SETUP:
BLENGTH 10
```

:SEARch<x>:SLOGic:LINBus[:SETup]:BRATe

機能　ロジック LIN バス信号サーチのビットレート (データ転送速度) を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:LINBus[:SETup]:
BRATe {<NRf>|USER,<NRf>}
:SEARch<x>:SLOGic:LINBus[:SETup]:
BRATe?
```
<x> = 1、2
<NRf> = 1200、2400、4800、9600、19200
USER の <NRf> = 5.4 節参照。

例
```
:SEARCH1:SLOGIC:LINBUS:SETUP:
BRATE 19200
:SEARCH1:SLOGIC:LINBUS:SETUP:
BRATE? -> :SEARCH1:SLOGIC:LINBUS:
SETUP:BRATE 19200
```

:SEARch<x>:SLOGic:LINBus[:SETup]:DATA?

機能　ロジック LIN バス信号サーチのデータに関するすべての設定値を問い合わせます。

構文
```
:SEARch<x>:SLOGic:LINBus[:SETup]:
DATA?
```
<x> = 1、2

:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:BNUM

機能　ロジック LIN バス信号サーチのデータのバイト数を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:LINBus[:SETup]:
DATA:BNUM {<NRf>}
:SEARch<x>:SLOGic:LINBus[:SETup]:
DATA:BNUM?
```
<x> = 1、2
<NRf> = 1 ～ 8

例
```
:SEARCH1:SLOGIC:LINBUS:SETUP:DATA:
BNUM 1
:SEARCH1:SLOGIC:LINBUS:SETUP:DATA:
BNUM? -> :SEARCH1:SLOGIC:LINBUS:
SETUP:DATA:BNUM 1
```

:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:BORDer

機能　ロジック LIN バス信号サーチのデータのバイトオーダを設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:LINBus[:SETup]:
DATA:BORDer {BIG|LITTle}
:SEARch<x>:SLOGic:LINBus[:SETup]:
DATA:BORDer?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:LINBUS:SETUP:DATA:
BORDER BIG
:SEARCH1:SLOGIC:LINBUS:SETUP:DATA:
BORDER? -> :SEARCH1:SLOGIC:LINBUS:
SETUP:DATA:BORDER BIG
```

:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:CONDition

機能　ロジック LIN バス信号サーチのデータ条件を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:LINBus[:SETup]:
DATA:CONDition {BETWeen|DONTcare|
FALSe|GTHan|LTHan|ORANge|TRUE}
:SEARch<x>:SLOGic:LINBus[:SETup]:
DATA:CONDition?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:LINBUS:SETUP:DATA:
CONDITION BETWEEN
:SEARCH1:SLOGIC:LINBUS:SETUP:DATA:
CONDITION? -> :SEARCH1:SLOGIC:
LINBUS:SETUP:DATA:CONDITION BETWEEN
```

:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:DATA<x>

機能　ロジック LIN バス信号サーチのデータの比較データを設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:LINBus[:SETup]:
DATA:DATA<x> {<NRf>}
:SEARch<x>:SLOGic:LINBus[:SETup]:
DATA:DATA<x>?
```
SEARch<x> の <x> = 1、2
DATA<x> の <x> = 1、2
<NRf> = 5.4 節参照。

例
```
:SEARCH1:SLOGIC:LINBUS:SETUP:DATA:
DATA1 1
:SEARCH1:SLOGIC:LINBUS:SETUP:DATA:
DATA1? -> :SEARCH1:SLOGIC:LINBUS:
SETUP:DATA:DATA1 1.0000000E+00
```

解説
- 「:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:CONDition GTHan」のときは「:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:DATA1」で設定します。
- 「:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:CONDition LTHan」のときは「:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:DATA2」で設定します。
- 「:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:CONDition BETWeen|ORANge」のときは、小さい値を「:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:DATA1」、大きい値を「:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:DATA2」で設定します。

:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:HEXA

機能　ロジック LIN バス信号サーチのデータを HEXA で設定します。

構文
```
:SEARch<x>:SLOGic:LINBus[:SETup]:
DATA:HEXA {< 文字列 >}
```
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 16 文字以内 (BNUM の設定値で変化します)

例
```
:SEARCH1:SLOGIC:LINBUS:SETUP:DATA:
HEXA "3B"
```

:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:MSBLsb

機能　ロジック LIN バス信号サーチのデータの MSB/LSB のビットを設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:MSBLsb {<NRf>,<NRf>}
:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:MSBLsb?
```
<x> = 1、2
<NRf> = 5.4 節参照。

例
```
:SEARCH1:SLOGIC:LINBUS:SETUP:DATA:MSBLSB 1,0
:SEARCH1:SLOGIC:LINBUS:SETUP:DATA:MSBLSB? -> :SEARCH1:SLOGIC:LINBUS:SETUP:DATA:MSBLSB 1,0
```

:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:PATTern

機能　ロジック LIN バス信号サーチのデータを BINARY で設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:PATTern {< 文字列 >}
:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:PATTern?
```
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 64 文字以内 (BNUM の設定値で変化します)

例
```
:SEARCH1:SLOGIC:LINBUS:SETUP:DATA:PATTERN "11011111"
:SEARCH1:SLOGIC:LINBUS:SETUP:DATA:PATTERN? -> :SEARCH1:SLOGIC:LINBUS:SETUP:DATA:PATTERN "11011111"
```

:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:SIGN

機能　ロジック LIN バス信号サーチのデータの符号を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:SIGN {SIGN|UNSign}
:SEARch<x>:SLOGic:LINBus[:SETup]:DATA:SIGN?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:LINBUS:SETUP:DATA:SIGN SIGN
:SEARCH1:SLOGIC:LINBUS:SETUP:DATA:SIGN? -> :SEARCH1:SLOGIC:LINBUS:SETUP:DATA:SIGN SIGN
```

:SEARch<x>:SLOGic:LINBus[:SETup]:ERRor?

機能　ロジック LIN バス信号サーチの Error に関するすべてのを設定値を問い合わせします。

構文
```
:SEARch<x>:SLOGic:LINBus[:SETup]:ERRor?
```
<x> = 1、2

:SEARch<x>:SLOGic:LINBus[:SETup]:ERRor:CHECksum

機能　ロジック LIN バス信号サーチの Checksum Error を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:LINBus[:SETup]:ERRor:CHECksum {<Boolean>}
:SEARch<x>:SLOGic:LINBus[:SETup]:ERRor:CHECksum?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:LINBUS:SETUP:ERROR:CHECKSUM ON
:SEARCH1:SLOGIC:LINBUS:SETUP:ERROR:CHECKSUM? -> :SEARCH1:SLOGIC:LINBUS:SETUP:ERROR:CHECKSUM 1
```

:SEARch<x>:SLOGic:LINBus[:SETup]:ERRor:FRAMing

機能　ロジック LIN バス信号サーチの Framing Error を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:LINBus[:SETup]:ERRor:FRAMing {<Boolean>}
:SEARch<x>:SLOGic:LINBus[:SETup]:ERRor:FRAMing?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:LINBUS:SETUP:ERROR:FRAMING ON
:SEARCH1:SLOGIC:LINBUS:SETUP:ERROR:FRAMING? -> :SEARCH1:SLOGIC:LINBUS:SETUP:ERROR:FRAMING 1
```

:SEARch<x>:SLOGic:LINBus[:SETup]:ERRor:PARity

機能　ロジック LIN バス信号サーチの Parity Error を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:LINBus[:SETup]:
ERRor:PARity {<Boolean>}
:SEARch<x>:SLOGic:LINBus[:SETup]:
ERRor:PARity?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:LINBUS:SETUP:ERROR:
PARITY ON
:SEARCH1:SLOGIC:LINBUS:SETUP:ERROR:
PARITY? -> :SEARCH1:SLOGIC:LINBUS:
SETUP:ERROR:PARITY 1
```

:SEARch<x>:SLOGic:LINBus[:SETup]:ERRor:SYNCh

機能　ロジック LIN バス信号サーチの Synch Error を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:LINBus[:SETup]:
ERRor:SYNCh {<Boolean>}
:SEARch<x>:SLOGic:LINBus[:SETup]:
ERRor:SYNCh?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:LINBUS:SETUP:ERROR:
SYNCH ON
:SEARCH1:SLOGIC:LINBUS:SETUP:ERROR:
SYNCH? -> :SEARCH1:SLOGIC:LINBUS:
SETUP:ERROR:SYNCH 1
```

:SEARch<x>:SLOGic:LINBus[:SETup]:ERRor:TOUT

機能　ロジック LIN バス信号サーチの Timeout Error を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:LINBus[:SETup]:
ERRor:TOUT {<Boolean>}
:SEARch<x>:SLOGic:LINBus[:SETup]:
ERRor:TOUT?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:LINBUS:SETUP:ERROR:
TOUT ON
:SEARCH1:SLOGIC:LINBUS:SETUP:ERROR:
TOUT? -> :SEARCH1:SLOGIC:LINBUS:
SETUP:ERROR:TOUT 1
```

:SEARch<x>:SLOGic:LINBus[:SETup]:ID?

機能　ロジック LIN バス信号サーチの ID に関するすべての設定値を問い合わせます。

構文
```
:SEARch<x>:SLOGic:LINBus[:SETup]:ID?
```
<x> = 1、2

:SEARch<x>:SLOGic:LINBus[:SETup]:ID:HEXA

機能　ロジック LIN バス信号サーチの ID を HEXA で設定します。

構文
```
:SEARch<x>:SLOGic:LINBus[:SETup]:ID:
HEXA {< 文字列 >}
```
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字

例
```
:SEARCH1:SLOGIC:LINBUS:SETUP:ID:
HEXA "2A"
```

:SEARch<x>:SLOGic:LINBus[:SETup]:ID:PATTern

機能　ロジック LIN バス信号サーチの ID を BINARY で設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:LINBus[:SETup]:ID:
PATTern {< 文字列 >}
:SEARch<x>:SLOGic:LINBus[:SETup]:ID:
PATTern?
```
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 6 文字

例
```
:SEARCH1:SLOGIC:LINBUS:SETUP:ID:
PATTERN "101111"
:SEARCH1:SLOGIC:LINBUS:SETUP:ID:
PATTERN? -> :SEARCH1:SLOGIC:LINBUS:
SETUP:ID:PATTERN "101111"
```

:SEARch<x>:SLOGic:LINBus[:SETup]:MODE

機能　ロジック LIN バス信号サーチのモードを設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:LINBus[:SETup]:
MODE {IDData|SYNCh}
:SEARch<x>:SLOGic:LINBus[:SETup]:
MODE?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:LINBUS:SETUP:MODE
IDDATA
:SEARCH1:SLOGIC:LINBUS:SETUP:
MODE? -> :SEARCH1:SLOGIC:LINBUS:
SETUP:MODE IDDATA
```

:SEARch<x>:SLOGic:LINBus[:SETup]:REVision

機能　ロジック LIN バス信号サーチのレビジョン (1.3 or 2.0 or Both) を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:LINBus[:SETup]:
REVision {BOTH|LIN1_3|LIN2_0}
:SEARch<x>:SLOGic:LINBus[:SETup]:
REVision?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:LINBUS:SETUP:
REVISION LIN1_3
:SEARCH1:SLOGIC:LINBUS:SETUP:
REVISION? -> :SEARCH1:SLOGIC:LINBUS:
SETUP:REVISION LIN1_3
```

:SEARch<x>:SLOGic:LINBus[:SETup]:SPOint

機能　ロジック LIN バス信号サーチのサンプルポイントを設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:LINBus[:SETup]:
SPOint {<NRf>}
:SEARch<x>:SLOGic:LINBus[:SETup]:
SPOint?
```
<x> = 1、2
<NRf> = 18.8 ～ 90.6(%)

例
```
:SEARCH1:SLOGIC:LINBUS:SETUP:
SPOINT 18.8
:SEARCH1:SLOGIC:LINBUS:SETUP:SPOINT?
-> :SEARCH1:SLOGIC:LINBUS:SETUP:
SPOINT 18.8E+00
```

:SEARch<x>:SLOGic:LINBus[:SETup]:TRACe

機能　ロジック LIN バス信号サーチのトレースを設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:LINBus[:SETup]:
TRACe {A<y>|B<y>|C<y>|D<y>}
:SEARch<x>:SLOGic:LINBus[:SETup]:
TRACe?
```
<x> = 1、2
<y> = 0 ～ 7

例
```
:SEARCH1:SLOGIC:LINBUS:SETUP:TRACE
A0
:SEARCH1:SLOGIC:LINBUS:SETUP:
TRACE? -> :SEARCH1:SLOGIC:LINBUS:
SETUP:TRACE A0
```

解説　DLM6000 の 16 ビットモデルでは {A<x>|C<x>} が有効です。

:SEARch<x>:SLOGic:SPIBus?

機能　ロジック SPI バス信号サーチに関するすべての設定値を問い合わせます。

構文
```
:SEARch<x>:SLOGic:SPIBus?
```
<x> = 1、2

:SEARch<x>:SLOGic:SPIBus:CLOCk?

機能　ロジック SPI バス信号サーチのクロック信号のチャネルに関するすべての設定値を問い合わせます。

構文
```
:SEARch<x>:SLOGic:SPIBus:CLOCk?
```
<x> = 1、2

:SEARch<x>:SLOGic:SPIBus:CLOCk:POLarity

機能　ロジック SPI バス信号サーチのクロック信号のチャネルの極性を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:SPIBus:CLOCk:
POLarity {FALL|RISE}
:SEARch<x>:SLOGic:SPIBus:CLOCk:
POLarity?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:SPIBUS:CLOCK:
POLARITY FALL
:SEARCH1:SLOGIC:SPIBUS:CLOCK:
POLARITY? -> :SEARCH1:SLOGIC:SPIBUS:
CLOCK:POLARITY FALL
```

:SEARch<x>:SLOGic:SPIBus:CLOCk:SOURce

機能　ロジック SPI バス信号サーチのクロック信号のチャネルを設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:SPIBus:CLOCk:
SOURce {A<y>|B<y>|C<y>|D<y>}
:SEARch<x>:SLOGic:SPIBus:CLOCk:
SOURce?
```
<x> = 1、2
<y> = 0 ～ 7

例
```
:SEARCH1:SLOGIC:SPIBUS:CLOCK:
SOURCE A0
:SEARCH1:SLOGIC:SPIBUS:CLOCK:
SOURCE? -> :SEARCH1:SLOGIC:SPIBUS:
CLOCK:SOURCE A0
```

解説　DLM6000 の 16 ビットモデルでは {A<x>|C<x>} が有効です。

:SEARch<x>:SLOGic:SPIBus:CS?

機能　ロジック SPI バス信号サーチのチップセレクト信号のチャネルに関するすべての設定値を問い合わせます。

構文
```
:SEARch<x>:SLOGic:SPIBus:CS?
```
<x> = 1、2

:SEARch<x>:SLOGic:SPIBus:CS:ACTive

機能　ロジック SPI バス信号サーチのチップセレクト信号のチャネルのアクティブレベルを設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:SPIBus:CS:
ACTive {HIGH|LOW}
:SEARch<x>:SLOGic:SPIBus:CS:ACTive?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:SPIBUS:CS:ACTIVE
HIGH
:SEARCH1:SLOGIC:SPIBUS:CS:
ACTIVE? -> :SEARCH1:SLOGIC:SPIBUS:
CS:
ACTIVE HIGH
```

:SEARch<x>:SLOGic:SPIBus:CS:TRACe

機能　ロジック SPI バス信号サーチのチップセレクト信号のチャネルを設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:SPIBus:CS:TRACe
{A<y>|B<y>|C<y>|D<y>|NONE}
:SEARch<x>:SLOGic:SPIBus:CS:TRACe?
```
<x> = 1、2
<y> = 0 ～ 7

例
```
:SEARCH1:SLOGIC:SPIBUS:CS:TRACE A0
:SEARCH1:SLOGIC:SPIBUS:CS:
TRACE? -> :SEARCH1:SLOGIC:SPIBUS:CS:
TRACE A0
```

解説　DLM6000 の 16 ビットモデルでは {A<x>|C<x>} が有効です。

:SEARch<x>:SLOGic:SPIBus[:SETup]?

機能　ロジック SPI バス信号サーチのセットアップに関するすべての設定値を問い合わせます。

構文
```
:SEARch<x>:SLOGic:SPIBus[:SETup]?
```
<x> = 1、2

:SEARch<x>:SLOGic:SPIBus[:SETup]:BITorder

機能　ロジック SPI バス信号サーチのビットオーダを設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:SPIBus[:SETup]:
BITorder {LSBFirst|MSBFirst}
:SEARch<x>:SLOGic:SPIBus[:SETup]:
BITorder?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:SPIBUS:SETUP:
BITORDER LSBFIRST
:SEARCH1:SLOGIC:SPIBUS:SETUP:
BITORDER? -> :SEARCH1:SLOGIC:SPIBUS:
SETUP:BITORDER LSBFIRST
```

:SEARch<x>:SLOGic:SPIBus[:SETup]:DATA<x>?

機能　ロジック SPI バス信号サーチの各データに関するすべての設定値を問い合わせます。

構文
```
:SEARch<x>:SLOGic:SPIBus[:SETup]:
DATA<x>?
```
SEARch<x> の <x> = 1、2
DATA<x> の <x> = 1、2

解説　DATA2 は「:SEARch<x>:SLOGic:SPIBus [:SETup]:MODE WIRE4」のときに有効です。

:SEARch<x>:SLOGic:SPIBus[:SETup]:DATA<x>:BYTE

機能　ロジック SPI バス信号サーチの各データのデータサイズ (バイト数) を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:SPIBus[:SETup]:
DATA<x>:BYTE {<NRf>}
:SEARch<x>:SLOGic:SPIBus[:SETup]:
DATA<x>:BYTE?
```
SEARch<x> の <x> = 1、2
DATA<x> の <x> = 1、2
<NRf> = 1 ～ 4

例
```
:SEARCH1:SLOGIC:SPIBUS:SETUP:DATA1:
BYTE 1
:SEARCH1:SLOGIC:SPIBUS:SETUP:DATA1:
BYTE? -> :SEARCH1:SLOGIC:SPIBUS:
SETUP:DATA1:BYTE 1
```

:SEARch<x>:SLOGic:SPIBus[:SETup]:DATA<x>:CONDition

機能　ロジック SPI バス信号サーチの各データの判定方法 (一致 / 不一致) を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:SPIBus[:SETup]:
DATA<x>:CONDition {FALSe|TRUE}
:SEARch<x>:SLOGic:SPIBus[:SETup]:
DATA<x>:CONDition?
```
SEARch<x> の <x> = 1、2
DATA<x> の <x> = 1、2

例
```
:SEARCH1:SLOGIC:SPIBUS:SETUP:DATA1:
CONDITION FALSE
:SEARCH1:SLOGIC:SPIBUS:SETUP:DATA1:
CONDITION? -> :SEARCH1:SLOGIC:
SPIBUS:
SETUP:DATA1:CONDITION FALSE
```

:SEARch<x>:SLOGic:SPIBus[:SETup]:DATA<x>:DPOSition

機能　ロジック SPI バス信号サーチの各データのパターン比較先頭位置を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:SPIBus[:SETup]:
DATA<x>:DPOSition {<NRf>}
:SEARch<x>:SLOGic:SPIBus[:SETup]:
DATA<x>:DPOSition?
```
SEARch<x> の <x> = 1、2
DATA<x> の <x> = 1、2
<NRf> = 0 ～ 9999

例
```
:SEARCH1:SLOGIC:SPIBUS:SETUP:DATA1:
DPOSITION 1
:SEARCH1:SLOGIC:SPIBUS:SETUP:DATA1:
DPOSITION? -> :SEARCH1:SLOGIC:
SPIBUS:
SETUP:DATA1:DPOSITION 1
```

:SEARch<x>:SLOGic:SPIBus[:SETup]:DATA<x>:DSIZe

機能　ロジック SPI バス信号サーチの各データのフィールド数を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:SPIBus[:SETup]:
DATA<x>:DSIZe {<NRf>}
:SEARch<x>:SLOGic:SPIBus[:SETup]:
DATA<x>:DSIZe?
```
SEARch<x> の <x> = 1、2
DATA<x> の <x> = 1、2
<NRf> = 1 ～ 4

例
```
:SEARCH1:SLOGIC:SPIBUS:SETUP:DATA1:
DSIZE 1
:SEARCH1:SLOGIC:SPIBUS:SETUP:DATA1:
DSIZE? -> :SEARCH1:SLOGIC:SPIBUS:
SETUP:DATA1:DSIZE 1
```

:SEARch<x>:SLOGic:SPIBus[:SETup]:DATA<x>:HEXA<x>

機能　ロジック SPI バス信号サーチの各データを HEXA で設定します。

構文
```
:SEARch<x>:SLOGic:SPIBus[:SETup]:
DATA<x>:HEXA<x> {< 文字列 >}
```
SEARch<x> の <x> = 1、2
DATA<x> の <x> = 1、2
HEXA<x> の <x> = 1 ～ 4
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 8 文字以内

例
```
:SEARCH1:SLOGIC:SPIBUS:SETUP:DATA1:
HEXA1 "EF"
```

:SEARch<x>:SLOGic:SPIBus[:SETup]:DATA<x>:PATTern<x>

機能　ロジック SPI バス信号サーチの各データを BINARY で設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:SPIBus[:SETup]:
DATA<x>:PATTern<x> {< 文字列 >}
:SEARch<x>:SLOGic:SPIBus[:SETup]:
DATA<x>:PATTern<x>?
```
SEARch<x> の <x> = 1、2
DATA<x> の <x> = 1、2
PATTern<x> の <x> = 1 ～ 4
< 文字列 > = '0'、'1'、'X' の組み合わせ 32 文字以内

例
```
:SEARCH1:SLOGIC:SPIBUS:SETUP:DATA1:
PATTERN1 "11101111"
:SEARCH1:SLOGIC:SPIBUS:SETUP:DATA1:
PATTERN1? -> :SEARCH1:SLOGIC:SPIBUS:
SETUP:DATA1:PATTERN1 "11101111"
```

:SEARch<x>:SLOGic:SPIBus[:SETup]:DATA<x>:TRACe

機能　ロジック SPI バス信号サーチの各データのソースチャネルを設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:SPIBus[:SETup]:
DATA<x>:TRACe {A<y>|B<y>|C<y>|D<y>}
:SEARch<x>:SLOGic:SPIBus[:SETup]:
DATA<x>:TRACe?
```
SEARch<x> の <x> = 1、2
DATA<x> の <x> = 1、2
<y> = 0 ～ 7

例
```
:SEARCH1:SLOGIC:SPIBUS:SETUP:DATA1:
TRACE A0
:SEARCH1:SLOGIC:SPIBUS:SETUP:DATA1:
TRACE? -> :SEARCH1:SLOGIC:SPIBUS:
SETUP:DATA1:TRACE A0
```

解説　DLM6000 の 16 ビットモデルでは {A<x>|C<x>} が有効です。

:SEARch<x>:SLOGic:SPIBus[:SETup]:EMSBLSB

機能　ロジック SPI バス信号サーチのフィールドの有効範囲を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:SPIBus[:SETup]:
EMSBLSB {<NRf>,<NRf>}
:SEARch<x>:SLOGic:SPIBus[:SETup]:
EMSBLSB?
```
<x> = 1、2
<NRf> = 5.5 節参照。

例
```
:SEARCH1:SLOGIC:SPIBUS:SETUP:
EMSBLSB 1,7
```
:SEARCH1:SLOGIC:SPIBUS:SETUP:EMSBLSB? -> :SEARCH1:SLOGIC:SPIBUS:SETUP:EMSBLSB 1,7

:SEARch<x>:SLOGic:SPIBus[:SETup]:FSIZe

機能　ロジック SPI バス信号サーチのフィールドサイズを設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:SPIBus[:SETup]:
FSIZe {<NRf>}
:SEARch<x>:SLOGic:SPIBus[:SETup]:
FSIZe?
```
<x> = 1、2
<NRf> = 4 ～ 32

例
```
:SEARCH1:SLOGIC:SPIBUS:SETUP:FSIZE 4
:SEARCH1:SLOGIC:SPIBUS:SETUP:FSIZE?
-> :SEARCH1:SLOGIC:SPIBUS:SETUP:
FSIZE 4
```

:SEARch<x>:SLOGic:SPIBus[:SETup]:ITIMe

機能　ロジック SPI バス信号サーチのアイドル時間を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:SPIBus[:SETup]:
ITIMe {< 時間 >|DONTcare}
:SEARch<x>:SLOGic:SPIBus[:SETup]:
ITIMe?
```
<x> = 1、2
< 時間 > = 10ns ～ 1ms(10ns ステップ)

例
```
:SEARCH1:SLOGIC:SPIBUS:SETUP:
ITIME 10NS
:SEARCH1:SLOGIC:SPIBUS:SETUP:ITIME?
-> :SEARCH1:SLOGIC:SPIBUS:SETUP:
ITIME 10.0000E-09
```

:SEARch<x>:SLOGic:SPIBus[:SETup]:MODE

機能　ロジック SPI バス信号サーチの結線方式 (3 線式 /4 線式) を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:SPIBus[:SETup]:
MODE {WIRE3|WIRE4}
:SEARch<x>:SLOGic:SPIBus[:SETup]:
MODE?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:SPIBUS:SETUP:MODE
WIRE3
:SEARCH1:SLOGIC:SPIBUS:SETUP:
MODE? -> :SEARCH1:SLOGIC:SPIBUS:
SETUP:MODE WIRE3
```

:SEARch<x>:SLOGic:UART?

機能　ロジック UART 信号サーチに関するすべて設定値を問い合わせします。

構文
```
:SEARch<x>:SLOGic:UART?
```
<x> = 1、2

:SEARch<x>:SLOGic:UART:BRATe

機能　ロジック UART 信号サーチのビットレート (データ転送速度) を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:UART:
BRATe {<NRf>|USER,<NRf>}
:SEARch<x>:SLOGic:UART:BRATe?
```
<x> = 1、2
<NRf> = 1200、2400、4800、9600、19200、38400、57600、115200
USER の <NRf> = 5.6 節参照。

例
```
:SEARCH1:SLOGIC:UART:BRATE 19200
:SEARCH1:SLOGIC:UART:BRATE?
-> :SEARCH1:SLOGIC:UART:BRATE 19200
```

:SEARch<x>:SLOGic:UART:DATA?

機能　ロジック UART 信号サーチのデータに関するすべての設定値を問い合わせします。

構文
```
:SEARch<x>:SLOGic:UART:DATA?
```
<x> = 1、2

:SEARch<x>:SLOGic:UART:DATA:BITorder

機能　ロジック UART 信号サーチのデータのビットオーダを設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:UART:DATA:
BITorder {LSBFirst|MSBFirst}
:SEARch<x>:SLOGic:UART:DATA:
BITorder?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:UART:DATA:
BITORDER LSBFIRST
:SEARCH1:SLOGIC:UART:DATA:BITORDER?
-> :SEARCH1:SLOGIC:UART:DATA:
BITORDER LSBFIRST
```

:SEARch<x>:SLOGic:UART:DATA:DSIZe

機能　ロジック UART 信号サーチのデータのバイト数を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:UART:DATA:
DSIZe {<NRf>}
:SEARch<x>:SLOGic:UART:DATA:DSIZe?
```
<x> = 1、2
<NRf> = 1 ～ 4

例
```
:SEARCH1:SLOGIC:UART:DATA:DSIZE 1
:SEARCH1:SLOGIC:UART:DATA:DSIZE?
-> :SEARCH1:SLOGIC:UART:DATA:DSIZE 1
```

:SEARch<x>:SLOGic:UART:DATA:HEXA

機能　ロジック UART 信号サーチのデータを HEXA で設定します。

構文
```
:SEARch<x>:SLOGic:UART:DATA:
HEXA {< 文字列 >}
```
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 8 文字以内 (1 バイト単位)

例
```
:SEARCH1:SLOGIC:UART:DATA:HEXA "A9"
```

:SEARch<x>:SLOGic:UART:DATA:PATTern

機能　ロジック UART 信号サーチのデータを BINARY で設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:UART:DATA:
PATTern {< 文字列 >}
:SEARch<x>:SLOGic:UART:DATA:PATTern?
```
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 32 文字以内 (1 バイト単位)

例
```
:SEARCH1:SLOGIC:UART:DATA:
PATTERN "11011111"
:SEARCH1:SLOGIC:UART:DATA:PATTERN?
-> :SEARCH1:SLOGIC:UART:DATA:PATTERN
"11011111"
```

:SEARch<x>:SLOGic:UART:ERRor?

機能　ロジック UART 信号サーチの Error に関するすべてのを設定値を問い合わせします。

構文
```
:SEARch<x>:SLOGic:UART:ERRor?
```
<x> = 1、2

:SEARch<x>:SLOGic:UART:ERRor:FRAMing

機能　ロジック UART 信号サーチの Framing Error を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:UART:ERRor:
FRAMing {<Boolean>}
:SEARch<x>:SLOGic:UART:ERRor:
FRAMing?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:UART:ERROR:FRAMING
ON
:SEARCH1:SLOGIC:UART:ERROR:FRAMING?
-> :SEARCH1:SLOGIC:UART:ERROR:
FRAMING 1
```

:SEARch<x>:SLOGic:UART:ERRor:PARity

機能　ロジック UART 信号サーチの Parity Error を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:UART:ERRor:
PARity {<Boolean>}
:SEARch<x>:SLOGic:UART:ERRor:PARity?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:UART:ERROR:PARITY ON
:SEARCH1:SLOGIC:UART:ERROR:PARITY?
-> :SEARCH1:SLOGIC:UART:ERROR:
PARITY 1
```

::SEARch<x>:SLOGic:UART:ERRor:PMODe

機能　ロジック UART 信号サーチの Parity モードを設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:UART:ERRor:
PMODe {EVEN|ODD}
:SEARch<x>:SLOGic:UART:ERRor:PMODe?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:UART:ERROR:PMODE
EVEN
:SEARCH1:SLOGIC:UART:ERROR:PMODE?
-> :SEARCH1:SLOGIC:UART:ERROR:
PMODE EVEN
```

:SEARch<x>:SLOGic:UART:FORMat

機能　ロジック UART 信号サーチのフォーマットを設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:UART:
FORMat {BIT7parity|BIT8Noparity|
BIT8Parity}
:SEARch<x>:SLOGic:UART:FORMat?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:UART:
FORMAT BIT7PARITY
:SEARCH1:SLOGIC:UART:FORMAT?
-> :SEARCH1:SLOGIC:UART:
FORMAT BIT7PARITY
```

:SEARch<x>:SLOGic:UART:MODE

機能　ロジック UART 信号サーチのモードを設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:UART:MODE
{DATA|ERRor}
:SEARch<x>:SLOGic:UART:MODE?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:UART:MODE DATA
:SEARCH1:SLOGIC:UART:MODE?
-> :SEARCH1:SLOGIC:UART:MODE DATA
```

:SEARch<x>:SLOGic:UART:POLarity

機能　ロジック UART 信号サーチの極性を設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:UART:
POLarity {NEGative|POSitive}
:SEARch<x>:SLOGic:UART:POLarity?
```
<x> = 1、2

例
```
:SEARCH1:SLOGIC:UART:
POLARITY NEGATIVE
:SEARCH1:SLOGIC:UART:POLARITY?
-> :SEARCH1:SLOGIC:UART:
POLARITY NEGATIVE
```

:SEARch<x>:SLOGic:UART:SPOint

機能　ロジック UART 信号サーチのサンプルポイントを設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:UART:SPOint
{<NRf>}
:SEARch<x>:SLOGic:UART:SPOint?
```
<x> = 1、2
<NRf> = 18.8 ～ 90.6(%)

例
```
:SEARCH1:SLOGIC:UART:SPOINT 18.8
:SEARCH1:SLOGIC:UART:SPOINT? ->
:SEARCH1:SLOGIC:UART:SPOINT 18.8E+00
```

:SEARch<x>:SLOGic:UART:TRACe

機能　ロジック UART 信号サーチのトレースを設定 / 問い合わせします。

構文
```
:SEARch<x>:SLOGic:UART:TRACe
{A<y>|B<y>|C<y>|D<y>}
:SEARch<x>:SLOGic:UART:TRACe?
```
<x> = 1、2
<y> = 0 ～ 7

例
```
:SEARCH1:SLOGIC:UART:TRACE A0
:SEARCH1:SLOGIC:UART:TRACE?
-> :SEARCH1:SLOGIC:UART:TRACE A0
```

解説　DLM6000 の 16 ビットモデルでは {A<x>|C<x>} が有効です。

:SEARch<x>:SPIBus?

機能　SPI バス信号サーチに関するすべての設定値を問い合わせします。

構文
```
:SEARch<x>:SPIBus?
```
<x> = 1、2

:SEARch<x>:SPIBus:CLOCk

機能　SPI バス信号サーチのクロック信号のチャネルに関するすべての設定値を問い合わせします。

構文
```
:SEARch<x>:SPIBus:CLOCk?
```
<x> = 1、2

:SEARch<x>:SPIBus:CLOCk:POLarity

機能　SPI バス信号サーチのクロック信号のチャネルの極性を設定 / 問い合わせします。

構文
```
:SEARch<x>:SPIBus:CLOCk:
POLarity {FALL|RISE}
:SEARch<x>:SPIBus:CLOCk:POLarity?
```
<x> = 1、2

例
```
:SEARCH1:SPIBUS:CLOCK:POLARITY FALL
:SEARCH1:SPIBUS:CLOCK:POLARITY?
-> :SEARCH1:SPIBUS:CLOCK:
POLARITY FALL
```

:SEARch<x>:SPIBus:CLOCk:SOURce

機能　SPI バス信号サーチのクロック信号のチャネルを設定 / 問い合わせします。

構文
```
:SEARch<x>:SPIBus:CLOCk:SOURce
{<NRf>}
:SEARch<x>:SPIBus:CLOCk:SOURce?
```
<x> = 1、2
<NRf> = 1 ～ 8

例
```
:SEARCH1:SPIBUS:CLOCK:SOURCE 1
:SEARCH1:SPIBUS:CLOCK:SOURCE?
-> :SEARCH1:SPIBUS:CLOCK:SOURCE 1
```

:SEARch<x>:SPIBus:CS?

機能　SPI バス信号サーチのチップセレクト信号のチャネルに関するすべての設定値を問い合わせします。

構文
```
:SEARch<x>:SPIBus:CS?
```
<x> = 1、2

:SEARch<x>:SPIBus:CS:ACTive

機能　SPI バス信号サーチのチップセレクト信号のチャネルのアクティブレベルを設定 / 問い合わせします。

構文
```
:SEARch<x>:SPIBus:CS:ACTive
{HIGH|LOW}
:SEARch<x>:SPIBus:CS:ACTive?
```
<x> = 1、2

例
```
:SEARCH1:SPIBUS:CS:ACTIVE HIGH
:SEARCH1:SPIBUS:CS:ACTIVE?
-> :SEARCH1:SPIBUS:CS:ACTIVE HIGH
```

:SEARch<x>:SPIBus:CS:TRACe

機能　SPI バス信号サーチのチップセレクト信号のチャネルを設定 / 問い合わせします。

構文
```
:SEARch<x>:SPIBus:CS:
TRACe {<NRf>|NONE}
:SEARch<x>:SPIBus:CS:TRACe?
```
<x> = 1、2
<NRf> = 1 ～ 8

例
```
:SEARCH1:SPIBUS:CS:TRACE 1
:SEARCH1:SPIBUS:CS:TRACE?
-> :SEARCH1:SPIBUS:CS:TRACE 1
```

:SEARch<x>:SPIBus:SETup?

機能　SPI バス信号サーチのセットアップに関するすべての設定値を問い合わせします。

構文
```
:SEARch<x>:SPIBus:SETup?
```
<x> = 1、2

:SEARch<x>:SPIBus[:SETup]:BITorder

機能　SPI バス信号サーチのビットオーダを設定 / 問い合わせします。

構文
```
:SEARch<x>:SPIBus[:SETup]:
BITorder {LSBFirst|MSBFirst}
:SEARch<x>:SPIBus[:SETup]:BITorder?
```
<x> = 1、2

例
```
:SEARCH1:SPIBUS:SETUP:BITORDER
LSBFIRST
:SEARCH1:SPIBUS:SETUP:BITORDER?
-> :SEARCH1:SPIBUS:SETUP:
BITORDER LSBFIRST
```

:SEARch<x>:SPIBus[:SETup]:DATA<x>?

機能　SPI バス信号サーチの各データに関するすべての設定値を問い合わせします。

構文
```
:SEARch<x>:SPIBus[:SETup]:DATA<x>?
```
SEARch<x> の <x> = 1、2
DATA<x> の <x> = 1、2

解説　DATA2 は「:SEARch<x>:SPIBus[:SETup]:MODE WIRE4」のときに有効です。

:SEARch<x>:SPIBus[:SETup]:DATA<x>:BYTE

機能　SPI バス信号サーチの各データのデータサイズ (バイト数) を設定 / 問い合わせします。

構文
```
:SEARch<x>:SPIBus[:SETup]:DATA<x>:
BYTE {<NRf>}
:SEARch<x>:SPIBus[:SETup]:DATA<x>:
BYTE?
```
SEARch<x> の <x> = 1、2
DATA<x> の <x> = 1、2
<NRf> = 1 ～ 4

例
```
:SEARCH1:SPIBUS:SETUP:DATA1:BYTE 1
:SEARCH1:SPIBUS:SETUP:DATA1:BYTE?
-> :SEARCH1:SPIBUS:SETUP:DATA1:BYTE
1
```

:SEARch<x>:SPIBus[:SETup]:DATA<x>:CONDition

機能　SPI バス信号サーチの各データの判定方法 (一致 / 不一致) を設定 / 問い合わせします。

構文
```
:SEARch<x>:SPIBus[:SETup]:DATA<x>:
CONDition {FALSe|TRUE}
:SEARch<x>:SPIBus[:SETup]:DATA<x>:
CONDition?
```
SEARch<x> の <x> = 1、2
DATA<x> の <x> = 1、2

例
```
:SEARCH1:SPIBUS:SETUP:DATA1:
CONDITION TRUE
:SEARCH1:SPIBUS:SETUP:DATA1:
CONDITION?
-> :SEARCH1:SPIBUS:SETUP:DATA1:
CONDITION TRUE
```

:SEARch<x>:SPIBus[:SETup]:DATA<x>:DPOSition

機能　SPI バス信号サーチの各データのパターン比較先頭位置を設定 / 問い合わせします。

構文
```
:SEARch<x>:SPIBus[:SETup]:DATA<x>:
DPOSition {<NRf>}
:SEARch<x>:SPIBus[:SETup]:DATA<x>:
DPOSition?
```
SEARch<x> の <x> = 1、2
DATA<x> の <x> = 1、2
<NRf> = 0 ～ 9999

例
```
:SEARCH1:SPIBUS:SETUP:DATA1:
DPOSITION 1
:SEARCH1:SPIBUS:SETUP:DATA1:
DPOSITION?
-> :SEARCH1:SPIBUS:SETUP:DATA1:
DPOSITION 1
```

:SEARch<x>:SPIBus[:SETup]:DATA<x>:DSIZe

機能　SPI バス信号サーチの各データのフィールド数を設定 / 問い合わせします。

構文
```
:SEARch<x>:SPIBus[:SETup]:DATA<x>:
DSIZe {<NRf>}
:SEARch<x>:SPIBus[:SETup]:DATA<x>:
DSIZe?
```
SEARch<x> の <x> = 1、2
DATA<x> の <x> = 1、2
<NRf> = 1 ～ 4

例
```
:SEARCH1:SPIBUS:SETUP:DATA1:DSIZE 1
:SEARCH1:SPIBUS:SETUP:DATA1:DSIZE?
-> :SEARCH1:SPIBUS:SETUP:DATA1:
DSIZE 1
```

:SEARch<x>:SPIBus[:SETup]:DATA<x>:HEXA<x>

機能　SPI バス信号サーチの各データを HEXA で設定します。

構文
```
:SEARch<x>:SPIBus[:SETup]:DATA<x>:
HEXA<x> {< 文字列 >}
```
SEARch<x> の <x> = 1、2
DATA<x> の <x> = 1、2
HEXA<x> の <x> = 1 ～ 4
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 8 文字以内

例
```
:SEARCH1:SPIBUS:SETUP:DATA1:
HEXA1 "EF"
```

:SEARch<x>:SPIBus[:SETup]:DATA<x>:PATTern<x>

機能　SPI バス信号サーチの各データを BINARY で設定 / 問い合わせします。

構文
```
:SEARch<x>:SPIBus[:SETup]:DATA<x>:
PATTern<x> {< 文字列 >}
:SEARch<x>:SPIBus[:SETup]:DATA<x>:
PATTern<x>?
```
SEARch<x> の <x> = 1、2
DATA<x> の <x> = 1、2
PATTern<x> の <x> = 1 ～ 4
< 文字列 > = '0'、'1'、'X' の組み合わせ 32 文字以内

例
```
:SEARCH1:SPIBUS:SETUP:DATA1:
PATTERN1 "11101111"
:SEARCH1:SPIBUS:SETUP:DATA1:
PATTERN1?
-> :SEARCH1:SPIBUS:SETUP:DATA1:
PATTERN1 "11101111"
```

:SEARch<x>:SPIBus[:SETup]:DATA<x>:TRACe

機能　SPI バス信号サーチの各データのソースチャネルを設定 / 問い合わせします。

構文
```
:SEARch<x>:SPIBus[:SETup]:DATA<x>:
TRACe {<NRf>}
:SEARch<x>:SPIBus[:SETup]:DATA<x>:
TRACe?
```
SEARch<x> の <x> = 1、2
DATA<x> の <x> = 1、2
<NRf> = 1 ～ 8

例
```
:SEARCH1:SPIBUS:SETUP:DATA1:TRACE 1
:SEARCH1:SPIBUS:SETUP:DATA1:TRACE?
-> :SEARCH1:SPIBUS:SETUP:DATA1:
TRACE 1
```

:SEARch<x>:SPIBus[:SETup]:EMSBLSB

機能　SPI バス信号サーチのフィールドの有効範囲を設定 / 問い合わせします。

構文
```
:SEARch<x>:SPIBus[:SETup]:
EMSBLSB {<NRf>,<NRf>}
:SEARch<x>:SPIBus[:SETup]:EMSBLSB?
```
<x> = 1、2
<NRf> = 5.5 節参照。

例
```
:SEARCH1:SPIBUS:SETUP:EMSBLSB 1,7
:SEARCH1:SPIBUS:SETUP:EMSBLSB?
-> :SEARCH1:SPIBUS:SETUP:EMSBLSB 1,7
```

:SEARch<x>:SPIBus[:SETup]:FSIZe

機能　SPI バス信号サーチのフィールドサイズを設定 / 問い合わせします。

構文
```
:SEARch<x>:SPIBus[:SETup]:
FSIZe {<NRf>}
:SEARch<x>:SPIBus[:SETup]:FSIZe?
```
<x> = 1、2
<NRf> = 4 ～ 32

例
```
:SEARCH1:SPIBUS:SETUP:FSIZE 4
:SEARCH1:SPIBUS:SETUP:FSIZE?
-> :SEARCH1:SPIBUS:SETUP:FSIZE 4
```

:SEARch<x>:SPIBus[:SETup]:ITIMe

機能　SPI バス信号サーチのアイドル時間を設定 / 問い合わせします。

構文
```
:SEARch<x>:SPIBus[:SETup]:
ITIMe {< 時間 >|DONTcare}
:SEARch<x>:SPIBus[:SETup]:ITIMe?
```
<x> = 1、2
< 時間 > = 10ns ～ 1ms(10ns ステップ)

例
```
:SEARCH1:SPIBUS:SETUP:ITIME 10NS
:SEARCH1:SPIBUS:SETUP:ITIME?
-> :SEARCH1:SPIBUS:SETUP:ITIME
10.0000E-09
```

:SEARch<x>:SPIBus[:SETup]:MODE

機能 :　SPI バス信号サーチの結線方式 (3 線式 /4 線式) を設定 / 問い合わせします。

構文
```
:SEARch<x>:SPIBus[:SETup]:
MODE {WIRE3|WIRE4}
:SEARch<x>:SPIBus[:SETup]:MODE?
```
<x> = 1、2

例
```
:SEARCH1:SPIBUS:SETUP:MODE WIRE3
:SEARCH1:SPIBUS:SETUP:MODE?
-> :SEARCH1:SPIBUS:SETUP:MODE WIRE3
```

:SEARch<x>:TRACe<x>:LEVel

機能　各ソースチャネルのしきい値 (Threshold) レベルを設定 / 問い合わせします。

構文
```
:SEARch<x>:TRACe<x>:LEVel {<NRf>|
< 電圧 >|< 電流 >}
:SEARch<x>:TRACe<x>:LEVel?
```
SEARch<x> の <x> = 1、2
TRACe<x> の <x> = 1 ～ 8
<NRf>、< 電圧 >、< 電流 > = 5.2 ～ 5.6 節参照。

例
```
:SEARCH1:TRACE1:LEVEL 0
:SEARCH1:TRACE1:LEVEL?
-> :SEARCH1:TRACE1:LEVEL 0.000E+00
```

解説　下記コマンドで設定したソースに対応するチャネルが対象です。
- 「:SEARch<x>:I2CBus:CLOCk:SOURce」
- 「:SEARch<x>:STRace」
- 「:SEARch<x>:SPIBus:CLOCk:SOURce」
- 「:SEARch<x>:SPIBus:CS:TRACe」
- 「:SEARch<x>:SPIBus:DATA[1-2]:TRACe」

:SEARch<x>:TYPE

機能　サーチタイプを設定 / 問い合わせします。

構文
```
:SEARch<x>:TYPE {CANBus|EDGE|
EQUalify|I2CBus|LEDGe|LI2Cbus|LINBus
|LLINbus|LQUalify|LSPAttern|LSPIbus|
LSTate|LUARt|LWIDth|SPATtern|SPIBus|
STATe|UART|WIDTh}
:SEARch<x>:TYPE?
```
<x> = 1、2

例
```
:SEARCH1:TYPE CANBUS
:SEARCH1:TYPE?
-> :SEARCH1:TYPE CANBUS
```

解説　{LEDGe|LI2Cbus|LLINbus|LQUalify|LSPAttern|LSPIbus|LSTate|LUARt|LWIDth} は、DLM6000 に適用できます。

:SEARch<x>:UART?

機能　UART 信号サーチに関するすべて設定値を問い合わせします。

構文
```
:SEARch<x>:UART?
```
<x> = 1、2

:SEARch<x>:UART:BRATe

機能　UART 信号サーチのビットレート (データ転送速度) を設定 / 問い合わせします。

構文
```
:SEARch<x>:UART:BRATe
{<NRf>|USER,<NRf>}
:SEARch<x>:UART:BRATe?
```
<x> = 1、2
<NRf> = 1200、2400、4800、9600、19200、38400、57600、115200
USER の <NRf> = 5.6 節参照。

例
```
:SEARCH1:UART:BRATE 19200
:SEARCH1:UART:BRATE?
-> :SEARCH1:UART:BRATE 19200
```

:SEARch<x>:UART:DATA?

機能　UART 信号サーチのデータに関するすべての設定値を問い合わせします。

構文
```
:SEARch<x>:UART:DATA?
```
<x> = 1、2

:SEARch<x>:UART:DATA:BITorder

機能　UART 信号サーチのデータのビットオーダを設定 / 問い合わせします。

構文
```
:SEARch<x>:UART:DATA:
BITorder {LSBFirst|MSBFirst}
:SEARch<x>:UART:DATA:BITorder?
```
<x> = 1、2

例
```
:SEARCH1:UART:DATA:BITORDER LSBFIRST
:SEARCH1:UART:DATA:BITORDER? ->
:SEARCH1:UART:DATA:BITORDER LSBFIRST
```

:SEARch<x>:UART:DATA:DSIZe

機能　UART 信号サーチのデータのバイト数を設定 / 問い合わせします。

構文
```
:SEARch<x>:UART:DATA:DSIZe {<NRf>}
:SEARch<x>:UART:DATA:DSIZe?
```
<x> = 1、2
<NRf> = 1 ～ 4

例
```
:SEARCH1:UART:DATA:DSIZE 1
:SEARCH1:UART:DATA:DSIZE?
-> :SEARCH1:UART:DATA:DSIZE 1
```

:SEARch<x>:UART:DATA:HEXA

機能　UART 信号サーチのデータを HEXA で設定します。

構文
```
:SEARch<x>:UART:DATA:HEXA {< 文字列 >}
```
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 8 文字以内 (1 バイト単位)

例
```
:SEARCH1:UART:DATA:HEXA "A9"
```

:SEARch<x>:UART:DATA:PATTern

機能　UART 信号サーチのデータを BINARY で設定 / 問い合わせします。

構文
```
:SEARch<x>:UART:DATA:
PATTern {< 文字列 >}
:SEARch<x>:UART:DATA:PATTern?
```
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 32 文字以内 (1 バイト単位)

例
```
:SEARCH1:UART:DATA:PATTERN
"11011111"
:SEARCH1:UART:DATA:PATTERN?
-> :SEARCH1:UART:DATA:
PATTERN "11011111"
```

:SEARch<x>:UART:ERRor?

機能　UART 信号サーチの Error に関するすべてのを設定値を問い合わせします。

構文
```
:SEARch<x>:UART:ERRor?
```
<x> = 1、2

:SEARch<x>:UART:ERRor:FRAMing

機能　UART 信号サーチの Framing Error を設定 / 問い合わせします。

構文
```
:SEARch<x>:UART:ERRor:
FRAMing {<Boolean>}
:SEARch<x>:UART:ERRor:FRAMing?
```
<x> = 1、2

例
```
:SEARCH1:UART:ERROR:FRAMING ON
:SEARCH1:UART:ERROR:FRAMING?
-> :SEARCH1:UART:ERROR:FRAMING 1
```

:SEARch<x>:UART:ERRor:PARity

機能　UART 信号サーチの Parity Error を設定 / 問い合わせします。

構文
```
:SEARch<x>:UART:ERRor:
PARity {<Boolean>}
:SEARch<x>:UART:ERRor:PARity?
```
<x> = 1、2

例
```
:SEARCH1:UART:ERROR:PARITY ON
:SEARCH1:UART:ERROR:PARITY?
-> :SEARCH1:UART:ERROR:PARITY 1
```

:SEARch<x>:UART:ERRor:PMODe

機能　UART 信号サーチの Parity モードを設定 / 問い合わせします。

構文
```
:SEARch<x>:UART:ERRor:
PMODe {EVEN|ODD}
:SEARch<x>:UART:ERRor:PMODe?
```
<x> = 1、2

例
```
:SEARCH1:UART:ERROR:PMODE EVEN
:SEARCH1:UART:ERROR:PMODE?
-> :SEARCH1:UART:ERROR:PMODE EVEN
```

:SEARch<x>:UART:FORMat

機能　UART 信号サーチのフォーマットを設定 / 問い合わせします。

構文
```
:SEARch<x>:UART:
FORMat {BIT7parity|BIT8Noparity|
BIT8Parity}
:SEARch<x>:UART:FORMat?
```
<x> = 1、2

例
```
:SEARCH1:UART:FORMAT BIT7PARITY
:SEARCH1:UART:FORMAT?
-> :SEARCH1:UART:FORMAT BIT7PARITY
```

:SEARch<x>:UART:MODE

機能　UART 信号サーチのモードを設定 / 問い合わせします。

構文
```
:SEARch<x>:UART:MODE {DATA|ERRor}
:SEARch<x>:UART:MODE?
```
<x> = 1、2

例
```
:SEARCH1:UART:MODE DATA
:SEARCH1:UART:MODE?
-> :SEARCH1:UART:MODE DATA
```

:SEARch<x>:UART:POLarity

機能　UART 信号サーチの極性を設定 / 問い合わせします。

構文
```
:SEARch<x>:UART:
POLarity {NEGative|POSitive}
:SEARch<x>:UART:POLarity?
```
<x> = 1、2

例
```
:SEARCH1:UART:POLARITY NEGATIVE
:SEARCH1:UART:POLARITY?
-> :SEARCH1:UART:POLARITY NEGATIVE
```

:SEARch<x>:UART:SPOint

機能　UART 信号サーチのサンプルポイントを設定 / 問い合わせします。

構文
```
:SEARch<x>:UART:SPOint {<NRf>}
:SEARch<x>:UART:SPOint?
```
<x> = 1、2
<NRf> = 18.8 ～ 90.6(%)

例
```
:SEARCH1:UART:SPOINT 18.8
:SEARCH1:UART:SPOINT?
-> :SEARCH1:UART:SPOINT 18.8E+00
```

:SEARch<x>:UART:TRACe

機能　UART 信号サーチのトレースを設定 / 問い合わせします。

構文
```
:SEARch<x>:UART:TRACe {<NRf>}
:SEARch<x>:UART:TRACe?
```
<x> = 1、2
<NRf> = 1 ～ 8

例
```
:SEARCH1:UART:TRACE 1
:SEARCH1:UART:TRACE?
-> :SEARCH1:UART:TRACE 1
```

7.5 SERialbus グループ

:SERialbus?

機能　シリアルバスセットアップに関するすべての設定値を問い合わせます。

構文　:SERialbus?

:SERialbus:SETup<x>?

機能　シリアルバスセットアップの各セットアップに関するすべての設定値を問い合わせます。

構文　:SERialbus:SETup<x>?

<x> = 1、2

:SERialbus:SETup<x>:ASETup:ABORt

機能　シリアルバスセットアップのオートセットアップを中止します。

構文　:SERialbus:SETup<x>:ASETup:ABORt

<x> = 1、2

例　:SERIALBUS:SETUP1:ASETUP:ABORT

:SERialbus:SETup<x>:ASETup:EXECute

機能　シリアルバスセットアップのオートセットアップを実行します。

構文　:SERialbus:SETup<x>:ASETup:EXECute

<x> = 1、2

例　:SERIALBUS:SETUP1:ASETUP:EXECUTE

:SERialbus:SETup<x>:ASETup:UNDO

機能　シリアルバスセットアップの実行したオートセットアップを取り消します。

構文　:SERialbus:SETup<x>:ASETup:UNDO

<x> = 1、2

例　:SERIALBUS:SETUP1:ASETUP:UNDO

:SERialbus:SETup<x>:CANBus?

機能　CAN バスセットアップに関するすべての設定値を問い合わせます。

構文　:SERialbus:SETup<x>:CANBus?

<x> = 1、2

:SERialbus:SETup<x>:CANBus:BRATe

機能　CAN バスセットアップのビットレート (データ転送速度) を設定 / 問い合わせします。

構文　:SERialbus:SETup<x>:CANBus:
BRATe {<NRf>|USER,<NRf>}
:SERialbus:SETup<x>:CANBus:BRATe?

<x> = 1、2

<NRf> = 33300、83300、125000、250000、500000、1000000

USER の <NRf> = 5.3 節参照。

例　:SERIALBUS:SETUP1:CANBUS:BRATE 83300
:SERIALBUS:SETUP1:CANBUS:BRATE? ->
:SERIALBUS:SETUP1:CANBUS:BRATE 83300

:SERialbus:SETup<x>:CANBus:RECessive

機能　CAN バスセットアップのリセッシブレベル (バスレベル) を設定 / 問い合わせします。

構文　:SERialbus:SETup<x>:CANBus:
RECessive {HIGH|LOW}
:SERialbus:SETup<x>:CANBus:
RECessive?

<x> = 1、2

例　:SERIALBUS:SETUP1:CANBUS:
RECESSIVE HIGH
:SERIALBUS:SETUP1:CANBUS:RECESSIVE?
-> :SERIALBUS:SETUP1:CANBUS:
RECESSIVE HIGH

:SERialbus:SETup<x>:CANBus:SPOint

機能　CAN バスセットアップのサンプルポイントを設定 / 問い合わせします。

構文　:SERialbus:SETup<x>:CANBus:
SPOint {<NRf>}
:SERialbus:SETup<x>:CANBus:SPOint?

<x> = 1、2

<NRf> = 18.8 ～ 90.6(%)

例　:SERIALBUS:SETUP1:CANBUS:SPOINT 18.8
:SERIALBUS:SETUP1:CANBUS:SPOINT?
-> :SERIALBUS:SETUP1:CANBUS:
SPOINT 18.8E+00

:SERialbus:SETup<x>:CANBus:TRACe
機能　CAN バスセットアップのトレースを設定 / 問い合わせします。
構文　:SERialbus:SETup<x>:CANBus:
TRACe {<NRf>}
:SERialbus:SETup<x>:CANBus:TRACe?
<x> = 1、2
<NRf> = 1 ～ 8
例　:SERIALBUS:SETUP1:CANBUS:TRACE 1
:SERIALBUS:SETUP1:CANBUS:TRACE?
-> :SERIALBUS:SETUP1:CANBUS:TRACE 1

:SERialbus:SETup<x>:I2CBus?
機能　I2C バスセットアップに関するすべての設定値を問い合わせます。
構文　:SERialbus:SETup<x>:I2CBus?
<x> = 1、2

:SERialbus:SETup<x>:I2CBus:CLOCk
機能　I2C バスセットアップのクロックチャネルを設定 / 問い合わせします。
構文　:SERialbus:SETup<x>:I2CBus:
CLOCk {<NRf>|A<y>|B<y>|C<y>|D<y>}
:SERialbus:SETup<x>:I2CBus:CLOCk?
<x> = 1、2
<NRf> = 1 ～ 8
<y> = 0 ～ 7
例　:SERIALBUS:SETUP1:I2CBUS:CLOCK 1
:SERIALBUS:SETUP1:I2CBUS:CLOCK?
-> :SERIALBUS:SETUP1:I2CBUS:CLOCK 1
解説　DLM6000 の 16 ビットモデルでは {A<x>|C<x>} が有効です。。

:SERialbus:SETup<x>:I2CBus:DTRace
機能　I2C バス信号解析のデータチャネルを設定 / 問い合わせします。
構文　:SERialbus:SETup<x>:I2CBus:
DTRace {<NRf>|A<y>|B<y>|C<y>|D<y>}
:SERialbus:SETup<x>:I2CBus:DTRace?
<x> = 1、2
<NRf> = 1 ～ 8
<y> = 0 ～ 7
例　:SERIALBUS:SETUP1:I2CBUS:DTRACE 1
:SERIALBUS:SETUP1:I2CBUS:DTRACE?
-> :SERIALBUS:SETUP1:I2CBUS:DTRACE 1
解説　DLM6000 の 16 ビットモデルでは {A<x>|C<x>} が有効です。

:SERialbus:SETup<x>:LINBus?
機能　LIN バスセットアップに関するすべての設定値を問い合わせます。
構文　:SERialbus:SETup<x>:LINBus?
<x> = 1、2

:SERialbus:SETup<x>:LINBus:BRATe
機能　LIN バスセットアップのビットレート (データ転送速度) を設定 / 問い合わせします。
構文　:SERialbus:SETup<x>:LINBus:
BRATe {<NRf>|USER,<NRf>}
:SERialbus:SETup<x>:LINBus:BRATe?
<x> = 1、2
<NRf> = 1200、2400、4800、9600、19200
USER の <NRf> = 5.4 節参照。
例　:SERIALBUS:SETUP1:LINBUS:BRATE 19200
:SERIALBUS:SETUP1:LINBUS:BRATE? ->
:SERIALBUS:SETUP1:LINBUS:BRATE 19200

:SERialbus:SETup<x>:LINBus:REVision
機能　LIN バスセットアップのレビジョン (1.3 or 2.0 or Both) を設定 / 問い合わせします。
構文　:SERialbus:SETup<x>:LINBus:
REVision {BOTH|LIN1_3|LIN2_0}
:SERialbus:SETup<x>:LINBus:REVision?
<x> = 1、2
例　:SERIALBUS:SETUP1:LINBUS:
REVISION LIN1_3
:SERIALBUS:SETUP1:LINBUS:REVISION?
-> :SERIALBUS:SETUP1:LINBUS:
REVISION LIN1_3

:SERialbus:SETup<x>:LINBus:SPOint
機能　LIN バスセットアップのサンプルポイントを設定 / 問い合わせします。
構文　:SERialbus:SETup<x>:LINBus:
SPOint {<NRf>}
:SERialbus:SETup<x>:LINBus:SPOint?
<x> = 1、2
<NRf> = 18.8 ～ 90.6(%)
例　:SERIALBUS:SETUP1:LINBUS:SPOINT 18.8
:SERIALBUS:SETUP1:LINBUS:SPOINT?
-> :SERIALBUS:SETUP1:LINBUS:SPOINT
18.8E+00

:SERialbus:SETup<x>:LINBus:TRACe

機能　LIN バスセットアップのトレースを設定 / 問い合わせします。

構文
```
:SERialbus:SETup<x>:LINBus:
TRACe {<NRf>|A<y>|B<y>|C<y>|D<y>}
:SERialbus:SETup<x>:LINBus:TRACe?
```
<x> = 1、2
<NRf> = 1 ～ 8
<y> = 0 ～ 7

例
```
:SERIALBUS:SETUP1:LINBUS:TRACE 1
:SERIALBUS:SETUP1:LINBUS:TRACE?
-> :SERIALBUS:SETUP1:LINBUS:TRACE 1
```

解説　DLM6000 の 16 ビットモデルでは {A<x>|C<x>} が有効です。

:SERialbus:SETup<x>:SPIBus?

機能　SPI バスセットアップに関するすべての設定値を問い合わせます。

構文
```
:SERialbus:SETup<x>:SPIBus?
```
<x> = 1、2

:SERialbus:SETup<x>:SPIBus:BITorder

機能　SPI バスセットアップのビットオーダを設定 / 問い合わせします。

構文
```
:SERialbus:SETup<x>:SPIBus:
BITorder {LSBFirst|MSBFirst}
:SERialbus:SETup<x>:SPIBus:BITorder?
```
<x> = 1、2

例
```
:SERIALBUS:SETUP1:SPIBUS:
BITORDER LSBFIRST
:SERIALBUS:SETUP1:SPIBUS:BITORDER?
-> :SERIALBUS:SETUP1:SPIBUS:BITORDER
LSBFIRST
```

:SERialbus:SETup<x>:SPIBus:CLOCk?

機能　SPI バスセットアップのクロック信号のチャネルに関するすべての設定値を問い合わせします。

構文
```
:SERialbus:SETup<x>:SPIBus:CLOCk?
```
<x> = 1、2

SERialbus:SETup<x>:SPIBus:CLOCk:POLarity

機能　SPI バスセットアップのクロック信号のチャネルの極性を設定 / 問い合わせします。

構文
```
:SERialbus:SETup<x>:SPIBus:CLOCk:
POLarity {FALL|RISE}
:SERialbus:SETup<x>:SPIBus:CLOCk:
POLarity?
```
<x> = 1、2

例
```
:SERIALBUS:SETUP1:SPIBUS:CLOCK:
POLARITY FALL
:SERIALBUS:SETUP1:SPIBUS:CLOCK:
POLARITY? -> :SERIALBUS:SETUP1:
SPIBUS:CLOCK:POLARITY FALL
```

:SERialbus:SETup<x>:SPIBus:CLOCk:TRACe

機能　SPI バスセットアップのクロック信号のチャネルを設定 / 問い合わせします。

構文
```
:SERialbus:SETup<x>:SPIBus:CLOCk:
TRACe {<NRf>|A<y>|B<y>|C<y>|D<y>}
:SERialbus:SETup<x>:SPIBus:CLOCk:
TRACe?
```
<x> = 1、2
<NRf> = 1 ～ 8
<y> = 0 ～ 7

例
```
:SERIALBUS:SETUP1:SPIBUS:CLOCK:
TRACE 1
:SERIALBUS:SETUP1:SPIBUS:CLOCK:
TRACE?
-> :SERIALBUS:SETUP1:SPIBUS:CLOCK:
TRACE 1
```

解説　DLM6000 の 16 ビットモデルでは {A<x>|C<x>} が有効です。

:SERialbus:SETup<x>:SPIBus:CS?

機能　SPI バスセットアップのチップセレクト信号のチャネルに関するすべての設定値を問い合わせします。

構文
```
:SERialbus:SETup<x>:SPIBus:CS?
```
<x> = 1、2

:SERialbus:SETup<x>:SPIBus:CS:ACTive

機能　SPI バスセットアップのチップセレクト信号のチャネルのアクティブレベルを設定 / 問い合わせします。

構文
```
:SERialbus:SETup<x>:SPIBus:CS:
ACTive {HIGH|LOW}
:SERialbus:SETup<x>:SPIBus:CS:
ACTive?
```
<x> = 1、2

例
```
:SERIALBUS:SETUP1:SPIBUS:CS:
ACTIVE HIGH
:SERIALBUS:SETUP1:SPIBUS:CS:ACTIVE?
-> :SERIALBUS:SETUP1:SPIBUS:CS:
ACTIVE HIGH
```

:SERialbus:SETup<x>:SPIBus:CS:TRACe

機能　SPI バスセットアップのチップセレクト信号のチャネルを設定 / 問い合わせします。

構文
```
:SERialbus:SETup<x>:SPIBus:CS:TRACe {<NRf>|A<y>|B<y>|C<y>|D<y>|ANONe|LNONe}
:SERialbus:SETup<x>:SPIBus:CS:TRACe?
```
<x> = 1、2
<NRf> = 1 ～ 8
<y> = 0 ～ 7

例
```
:SERIALBUS:SETUP1:SPIBUS:CS:TRACE 1
:SERIALBUS:SETUP1:SPIBUS:CS:TRACE?
->
:SERIALBUS:SETUP1:SPIBUS:CS:TRACE 1
```

解説
- {A<y>|B<y>|C<y>|D<y>|ANONe|LNONe} は DLM6054-L32 と DLM6104-L32 に、{A<y>|C<y>|ANONe|LNONe} は DLM6054-L16 と DLM6104-L16 にそれぞれ適用できます。
- ANONe は、アナログ SPI バスのデータ開始点をアイドル時間でコントロールするときに設定します。
- LNONe は、ロジック SPI バスのデータ開始点をアイドル時間でコントロールするときに設定します。

:SERialbus:SETup<x>:SPIBus:DATA<x>?

機能　SPI バスセットアップの各データに関するすべての設定値を問い合わせします。

構文
```
:SERialbus:SETup<x>:SPIBus:DATA<x>?
```
SETup<x> の <x> = 1、2
DATA<x> の <x> = 1、2

:SERialbus:SETup<x>:SPIBus:DATA<x>:ACTive

機能　SPI バスセットアップの各データのアクティブレベルを設定 / 問い合わせします。

構文
```
:SERialbus:SETup<x>:SPIBus:DATA<x>:ACTive {HIGH|LOW}
:SERialbus:SETup<x>:SPIBus:DATA<x>:ACTive?
```
SETup<x> の <x> = 1、2
DATA<x> の <x> = 1、2

例
```
:SERIALBUS:SETUP1:SPIBUS:DATA1:ACTIVE HIGH
:SERIALBUS:SETUP1:SPIBUS:DATA1:ACTIVE? -> :SERIALBUS:SETUP1:SPIBUS:DATA1:ACTIVE HIGH
```

:SERialbus:SETup<x>:SPIBus:DATA<x>:TRACe

機能　SPI バスセットアップの各データチャネルを設定 / 問い合わせします。

構文
```
:SERialbus:SETup<x>:SPIBus:DATA<x>:TRACe {<NRf>|A<y>|B<y>|C<y>|D<y>}
:SERialbus:SETup<x>:SPIBus:DATA<x>:TRACe?
```
SETup<x> の <x> = 1、2
DATA<x> の <x> = 1、2
<NRf> = 1 ～ 8
<y> = 0 ～ 7

例
```
:SERIALBUS:SETUP1:SPIBUS:DATA1:TRACE 1
:SERIALBUS:SETUP1:SPIBUS:DATA1:TRACE?
-> :SERIALBUS:SETUP1:SPIBUS:DATA1:TRACE 1
```

解説　DLM6000 の 16 ビットモデルでは {A<x>|C<x>} が有効です。

:SERialbus:SETup<x>:SPIBus:ITIMe

機能　SPI バスセットアップのアイドル時間を設定 / 問い合わせします。

構文
```
:SERialbus:SETup<x>:SPIBus:ITIMe {< 時間 >}
:SERialbus:SETup<x>:SPIBus:ITIMe?
```
<x> = 1、2
< 時間 > = 10ns ～ 1ms(10ns ステップ)

例
```
:SERIALBUS:SETUP1:SPIBUS:ITIME 10NS
:SERIALBUS:SETUP1:SPIBUS:ITIME?
-> :SERIALBUS:SETUP1:SPIBUS:ITIME 10.0000E-09
```

:SERialbus:SETup<x>:SPIBus:MODE

機能　SPI バスセットアップの結線方式 (3 線式 /4 線式) を設定 / 問い合わせします。

構文
```
:SERialbus:SETup<x>:SPIBus:MODE {WIRE3|WIRE4}
:SERialbus:SETup<x>:SPIBus:MODE?
```
<x> = 1、2

例
```
:SERIALBUS:SETUP1:SPIBUS:MODE WIRE3
:SERIALBUS:SETUP1:SPIBUS:MODE?
-> :SERIALBUS:SETUP1:SPIBUS:MODE WIRE3
```

:SERialbus:SETup<x>:TRACe<x>?

機能　各トレースに関するすべての設定値を問い合わせします。

構文
```
:SERialbus:SETup<x>:TRACe<x>?
```
SETup<x> の <x> = 1、2
TRACe<x> の <x> = 1 ～ 8

:SERialbus:SETup<x>:TRACe<x>:HYSTeresis

機能　各トレースのしきい値 (Threshold) のヒステリシスを設定 / 問い合わせします。

構文
```
:SERialbus:SETup<x>:TRACe<x>:
HYSTeresis {<NRf>}
:SERialbus:SETup<x>:TRACe<x>:
HYSTeresis?
```
SETup<x> の <x> = 1、2
TRACe<x> の <x> = 1 ～ 8
<NRf> = 0 ～ 4(div、0.1div ステップ)

例
```
:SERIALBUS:SETUP1:TRACE1:
HYSTERESIS 1
:SERIALBUS:SETUP1:TRACE1:HYSTERESIS?
-> :SERIALBUS:SETUP1:TRACE1:
HYSTERESIS 1.000E+00
```

:SERialbus:SETup<x>:TRACe<x>:LEVel

機能　各トレースのしきい値 (Threshold) レベルを設定 / 問い合わせします。

構文
```
:SERialbus:SETup<x>:TRACe<x>:
LEVel {<NRf>|< 電圧 >|< 電流 >}
:SERialbus:SETup<x>:TRACe<x>:LEVel?
```
SETup<x> の <x> = 1、2
TRACe<x> の <x> = 1 ～ 8
<NRf>、< 電圧 >、< 電流 > = 5.2 ～ 5.6 節参照。

例
```
:SERIALBUS:SETUP1:TRACE1:LEVEL 0
:SERIALBUS:SETUP1:TRACE1:LEVEL?
-> :SERIALBUS:SETUP1:TRACE1:
LEVEL 0.000E+00
```

:SERialbus:SETup<x>:TYPE

機能　シリアルバスセットアップのタイプを設定 / 問い合わせします。

構文
```
:SERialbus:SETup<x>:TYPE {CANBus|
I2CBus|LINBus|SPIBus|UART}
:SERialbus:SETup<x>:TYPE?
```
<x> = 1、2

例
```
:SERIALBUS:SETUP1:TYPE CANBUS
:SERIALBUS:SETUP1:TYPE?
-> :SERIALBUS:SETUP1:TYPE CANBUS
```

:SERialbus:SETup<x>:UART?

機能　UART バスセットアップに関するすべての設定値を問い合わせます。

構文
```
:SERialbus:SETup<x>:UART?
```
<x> = 1、2

:SERialbus:SETup<x>:UART:BITorder

機能　UART バスセットアップのビットオーダを設定 / 問い合わせします。

構文
```
:SERialbus:SETup<x>:UART:
BITorder {LSBFirst|MSBFirst}
:SERialbus:SETup<x>:UART:BITorder?
```
<x> = 1、2

例
```
:SERIALBUS:SETUP1:UART:
BITORDER LSBFIRST
:SERIALBUS:SETUP1:UART:BITORDER?
-> :SERIALBUS:SETUP1:UART:BITORDER
LSBFIRST
```

:SERialbus:SETup<x>:UART:BRATe

機能　UART バスセットアップのビットレート (データ転送速度) を設定 / 問い合わせします。

構文
```
:SERialbus:SETup<x>:UART:
BRATe {<NRf>|USER,<NRf>}
:SERialbus:SETup<x>:UART:BRATe?
```
<x> = 1、2
<NRf> = 1200、2400、4800、9600、19200、38400、57600、115200

例
```
:SERIALBUS:SETUP1:UART:BRATE 19200
:SERIALBUS:SETUP1:UART:BRATE?
-> :SERIALBUS:SETUP1:UART:BRATE
19200
```

:SERialbus:SETup<x>:UART:FORMat

機能　UART バスセットアップのデータ形式を設定 / 問い合わせします。

構文
```
:SERialbus:SETup<x>:UART:
FORMat {BIT7parity|BIT8Noparity|
BIT8Parity}
:SERialbus:SETup<x>:UART:FORMat?
```
<x> = 1、2

例
```
:SERIALBUS:SETUP1:UART:
FORMAT BIT7PARITY
:SERIALBUS:SETUP1:UART:FORMAT?
-> :SERIALBUS:SETUP1:UART:
FORMAT BIT7PARITY
```

:SERialbus:SETup<x>:UART:PMODe

機能　UART バスセットアップの Parity モードを設定 / 問い合わせします。

構文
```
:SERialbus:SETup<x>:UART:
PMODe {EVEN|ODD}
:SERialbus:SETup<x>:UART:PMODe?
```
<x> = 1、2

例
```
:SERIALBUS:SETUP1:UART:PMODE EVEN
:SERIALBUS:SETUP1:UART:PMODE?
-> :SERIALBUS:SETUP1:UART:PMODE EVEN
```

:SERialbus:SETup<x>:UART:POLarity

機能　UART バスセットアップの極性を設定 / 問い合わせします。

構文
```
:SERialbus:SETup<x>:UART:
POLarity {NEGative|POSitive}
:SERialbus:SETup<x>:UART:POLarity?
```
<x> = 1、2

例
```
:SERIALBUS:SETUP1:UART:
POLARITY NEGATIVE
:SERIALBUS:SETUP1:UART:POLARITY?
-> :SERIALBUS:SETUP1:UART:
POLARITY NEGATIVE
```

:SERialbus:SETup<x>:UART:SPOint

機能　UART バスセットアップのサンプルポイントを設定 / 問い合わせします。

構文
```
:SERialbus:SETup<x>:UART:
SPOint {<NRf>}
:SERialbus:SETup<x>:UART:SPOint?
```
<x> = 1、2
<NRf> = 18.8 ～ 90.6(%)

例
```
:SERIALBUS:SETUP1:UART:SPOINT 18.8
:SERIALBUS:SETUP1:UART:SPOINT?
-> :SERIALBUS:SETUP1:UART:
SPOINT 18.8E+00
```

:SERialbus:SETup<x>:UART:TRACe

機能　UART バスセットアップのソースを設定 / 問い合わせします。

構文
```
:SERialbus:SETup<x>:UART:
TRACe {<NRf>|A<y>|B<y>|C<y>|D<y>}
:SERialbus:SETup<x>:UART:TRACe?
```
<x> = 1、2
<NRf> = 1 ～ 8
<y> = 0 ～ 7

例
```
:SERIALBUS:SETUP1:UART:TRACE 1
:SERIALBUS:SETUP1:UART:TRACE?
-> :SERIALBUS:SETUP1:UART:TRACE 1
```

解説　DLM6000 の 16 ビットモデルでは {A<x>|C<x>} が有効です。

:SERialbus:TLINk

機能　シリアルバスセットアップのトリガリンクを設定 / 問い合わせします。

構文
```
:SERialbus:TLINk {OFF|SETUP1|SETUP2}
:SERialbus:TLINk?
```

例
```
:SERIALBUS:TLINK OFF
:SERIALBUS:TLINK?
-> :SERIALBUS:TLINK OFF
```

7.6 TRIGger グループ

:TRIGger:EINTerval:EVENt<x>:CANBus?

機能　各イベントの CAN バス信号トリガに関するすべての設定値を問い合わせます。

構文　:TRIGger:EINTerval:EVENt<x>:CANBus?
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:CANBus:ACK

機能　CAN バス信号トリガの ACK 条件を設定 / 問い合わせします。

構文　:TRIGger:EINTerval:EVENt<x>:CANBus:ACK {ACK|ACKBoth|DONTcare|NONack}
:TRIGger:EINTerval:EVENt<x>:CANBus:ACK?
<x> = 1、2

例　:TRIGGER:EINTERVAL:EVENT1:CANBUS:ACK ACK
:TRIGGER:EINTERVAL:EVENT1:CANBUS:ACK? -> :TRIGGER:EINTERVAL:EVENT1:CANBUS:ACK ACK

:TRIGger:EINTerval:EVENt<x>:CANBus:BRATe

機能　CAN バス信号トリガのビットレート (データ転送速度) を設定 / 問い合わせします。

構文　:TRIGger:EINTerval:EVENt<x>:CANBus:BRATe {<NRf>|USER,<NRf>}
:TRIGger:EINTerval:EVENt<x>:CANBus:BRATe?
<x> = 1、2
<NRf> = 33300、83300、125000、250000、500000、1000000
USER の <NRf> =本体ユーザーズマニュアル参照。

例　:TRIGGER:EINTERVAL:EVENT1:CANBUS:BRATE 83300
:TRIGGER:EINTERVAL:EVENT1:CANBUS:BRATE? -> :TRIGGER:EINTERVAL:EVENT1:CANBUS:BRATE 83300

:TRIGger:EINTerval:EVENt<x>:CANBus:DATA?

機能　CAN バス信号トリガのデータに関するすべての設定値を問い合わせます。

構文　:TRIGger:EINTerval:EVENt<x>:CANBus:DATA?
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:BORDer

機能　CAN バス信号トリガのデータのバイトオーダを設定 / 問い合わせします。

構文　:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:BORDer {BIG|LITTle}
:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:BORDer?
<x> = 1、2

例　:TRIGGER:EINTERVAL:EVENT1:CANBUS:DATA:BORDER BIG:TRIGGER:EINTERVAL:EVENT1:CANBUS:DATA:BORDER?
-> :TRIGGER:EINTERVAL:EVENT1:CANBUS:DATA:BORDER BIG

:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:CONDition

機能　CAN バス信号トリガのデータ条件を設定 / 問い合わせします。

構文　:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:CONDition {BETWeen|DONTcare|FALSe|GTHan|LTHan|ORANge|TRUE}
:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:CONDition?
<x> = 1、2

例　:TRIGGER:EINTERVAL:EVENT1:CANBUS:DATA:CONDITION BETWEEN
:TRIGGER:EINTERVAL:EVENT1:CANBUS:DATA:CONDITION? -> :TRIGGER:EINTERVAL:EVENT1:CANBUS:DATA:CONDITION BETWEEN

:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:DATA<x>

機能　CAN バス信号トリガのデータの比較データを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:DATA<x> {<NRf>}
:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:DATA<x>?
```
EVENt<x> の <x> = 1、2
DATA<x> の <x> = 1、2
<NRf> =本体ユーザーズマニュアル参照。

例
```
:TRIGGER:EINTERVAL:EVENT1:CANBUS:DATA:DATA1 1
:TRIGGER:EINTERVAL:EVENT1:CANBUS:DATA:DATA1? -> :TRIGGER:EINTERVAL:EVENT1:CANBUS:DATA:DATA1 1.0000000E+00
```

解説
- 「:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:CONDition GTHan」のときは「:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:DATA1」で設定します。「:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:CONDition LTHan」のときは「:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:DATA2」で設定します。
- 「:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:CONDition BETWeen|ORANge」のときは、小さい値を「:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:DATA1」、大きい値を「:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:DATA2」で設定します。

:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:DLC

機能　CAN バス信号トリガのデータの有効バイト数 (DLC) を設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:DLC {<NRf>}
:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:DLC?
```
<x> = 1、2
<NRf> = 0 ～ 8

例
```
:TRIGGER:EINTERVAL:EVENT1:CANBUS:DATA:DLC 0:TRIGGER:EINTERVAL:EVENT1:CANBUS:DATA:DLC? -> :TRIGGER:EINTERVAL:EVENT1:CANBUS:DATA:DLC 0
```

:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:HEXA

機能　CAN バス信号トリガのデータを HEXA で設定します。

構文
```
:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:HEXA {< 文字列 >}
```
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 16 文字以内 (1 バイト単位)

例
```
:TRIGGER:EINTERVAL:EVENT1:CANBUS:DATA:HEXA "A9"
```

:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:MSBLsb

機能　CAN バス信号トリガのデータの MSB/LSB のビットを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:MSBLsb {<NRf>,<NRf>}
:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:MSBLsb?
```
<x> = 1、2
<NRf> =本体ユーザーズマニュアル参照。

例
```
:TRIGGER:EINTERVAL:EVENT1:CANBUS:DATA:MSBLSB 1,0
:TRIGGER:EINTERVAL:EVENT1:CANBUS:DATA:MSBLSB? -> :TRIGGER:EINTERVAL:EVENT1:CANBUS:DATA:MSBLSB 1,0
```

:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:PATTern

機能　CAN バス信号トリガのデータを BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:PATTern {< 文字列 >}
:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:PATTern?
```
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 64 文字以内 (1 バイト単ハ)

例
```
:TRIGGER:EINTERVAL:EVENT1:CANBUS:DATA:PATTERN "11011111"
:TRIGGER:EINTERVAL:EVENT1:CANBUS:DATA:PATTERN? -> :TRIGGER:EINTERVAL:EVENT1:CANBUS:DATA:PATTERN "11011111"
```

:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:SIGN

機能　CAN バス信号トリガのデータの符号を設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:SIGN {SIGN|UNSign}
:TRIGger:EINTerval:EVENt<x>:CANBus:DATA:SIGN?
```
<x> = 1、2

例
```
:TRIGGER:EINTERVAL:EVENT1:CANBUS:DATA:SIGN SIGN
:TRIGGER:EINTERVAL:EVENT1:CANBUS:DATA:SIGN? -> :TRIGGER:EINTERVAL:EVENT1:CANBUS:DATA:SIGN SIGN
```

:TRIGger:EINTerval:EVENt<x>:CANBus:IDEXt?

機能　CAN バス信号トリガの拡張フォーマットの ID に関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:CANBus:IDEXt?
```
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:CANBus:IDEXt:HEXA

機能　CAN バス信号トリガの拡張フォーマットの ID を HEXA で設定します。

構文
```
:TRIGger:EINTerval:EVENt<x>:CANBus:IDEXt:HEXA {< 文字列 >}
```
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 8 文字

例
```
:TRIGGER:EINTERVAL:EVENT1:CANBUS:IDEXT:HEXA "1AEF5906"
```

:TRIGger:EINTerval:EVENt<x>:CANBus:IDEXt:PATTern

機能　CAN バス信号トリガの拡張フォーマットの ID を BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:CANBus:IDEXt:PATTern {< 文字列 >}
:TRIGger:EINTerval:EVENt<x>:CANBus:IDEXt:PATTern?
```
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 29 文字

例
```
:TRIGGER:EINTERVAL:EVENT1:CANBUS:IDEXT:PATTERN
"11001011011100001110111011111"
:TRIGGER:EINTERVAL:EVENT1:CANBUS:IDEXT:PATTERN? -> :TRIGGER:EINTERVAL:EVENT1:CANBUS:IDEXT:PATTERN
"11001011
01110000111011011111"
```

:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR?

機能　CAN バス信号トリガの OR 条件に関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR?
```
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>?

機能　CAN バス信号トリガの OR 条件の各 ID に関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>?
```
EVENt<x> の <x> = 1、2
ID<x> の <x> = 1 ～ 4

:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:ACK

機能　CAN バス信号トリガの OR 条件の各 ACK 条件を設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:ACK {ACK|ACKBoth|DONTcare|NONack}
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:ACK?
```
EVENt<x> の <x> = 1、2
ID<x> の <x> = 1 ～ 4

例
```
:TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:ACK ACK
:TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:ACK? -> :TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:ACK ACK
```

:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA?

機能　CAN バス信号トリガの OR 条件の各データに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA?
```
EVENt<x> の <x> = 1、2
ID<x> の <x> = 1 ～ 4

:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:BORDer

機能　CAN バス信号トリガの OR 条件の各データのバイトオーダを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:BORDer {BIG|LITTle}
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:BORDer?
```
EVENt<x> の <x> = 1、2
ID<x> の <x> = 1 ～ 4

例
```
:TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:DATA:BORDER BIG
:TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:DATA:BORDER? -> :TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:DATA:BORDER BIG
```

:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:CONDition

機能　CAN バス信号トリガの OR 条件の各データ条件を設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:CONDition {BETWeen|DONTcare|FALSe|GTHan|LTHan|ORANge|TRUE}
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:CONDition?
```
EVENt<x> の <x> = 1、2
ID<x> の <x> = 1 ～ 4

例
```
:TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:DATA:CONDITION BETWEEN
:TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:DATA:CONDITION? -> :TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:DATA:CONDITION BETWEEN
```

:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:DATA<x>

機能　CAN バス信号トリガの OR 条件の各データの比較データを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:DATA<x> {<NRf>}
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:DATA<x>?
```
EVENt<x> の <x> = 1、2
ID<x> の <x> = 1 ～ 4
DATA<x> の <x> = 1、2
<NRf> =本体ユーザーズマニュアル参照。

例
```
:TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:DATA:DATA1 1
:TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:DATA:DATA1? -> :TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:DATA:DATA1 1.0000000E+00
```

解説
- 「:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:CONDition GTHan」のときは「:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:DATA1」で設定します。
- 「:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:CONDition LTHan」のときは「:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:DATA2」で設定します。
- 「:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:CONDition BETWeen|ORANge」のときは、小さい値を「:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:DATA1」、大きい値を「:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:DATA2」で設定します。

:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:DLC

機能　CAN バス信号トリガの OR 条件の各データの有効バイト数 (DLC) を設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:DLC {<NRf>}
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:DLC?
```
EVENt<x> の <x> = 1、2
ID<x> の <x> = 1 ～ 4
<NRf> = 0 ～ 8

例
```
:TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:DATA:DLC 0
:TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:DATA:DLC? -> :TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:DATA:DLC 0
```

:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:HEXA

機能　CAN バス信号トリガの OR 条件の各データを HEXA で設定します。

構文　:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:HEXA {< 文字列 >}
EVENt<x> の <x> = 1、2
ID<x> の <x> = 1 ～ 4
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 16 文字以内 (1 バイト単位)

例　:TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:DATA:HEXA "A9"

:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:MSBLsb

機能　CAN バス信号トリガの OR 条件の各データの MSB/LSB のビットを設定 / 問い合わせします。

構文　:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:MSBLsb {<NRf>,<NRf>}
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:MSBLsb?
EVENt<x> の <x> = 1、2
ID<x> の <x> = 1 ～ 4
<NRf> =本体ユーザーズマニュアル参照。

例　:TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:DATA:MSBLSB 1,0
:TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:DATA:MSBLSB? -> :TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:DATA:MSBLSB 1,0

:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:PATTern

機能　CAN バス信号トリガの OR 条件の各データを BINARY で設定 / 問い合わせします。

構文　:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:PATTern {< 文字列 >}
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:PATTern?
EVENt<x> の <x> = 1、2
ID<x> の <x> = 1 ～ 4
< 文字列 > = '0'、'1'、'X' の組み合わせ 64 文字以内 (1 バイト単位)

例　:TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:DATA:PATTERN "11011111"
:TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:DATA:PATTERN? -> :TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:DATA:PATTERN "11011111"

:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:SIGN

機能　CAN バス信号トリガの OR 条件の各データの符号を設定 / 問い合わせします。

構文　:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:SIGN {SIGN|UNSign}
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:DATA:SIGN?
EVENt<x> の <x> = 1、2
ID<x> の <x> = 1 ～ 4

例　:TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:DATA:SIGN SIGN
:TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:DATA:SIGN? -> :TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:DATA:SIGN SIGN

:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:FORMat

機能　CAN バス信号トリガの OR 条件の各メッセージフォーマット (標準 / 拡張) を設定 / 問い合わせします。

構文　:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:FORMat {STD|EXT}
:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:FORMat?
EVENt<x> の <x> = 1、2
ID<x> の <x> = 1 ～ 4

例　:TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:FORMAT STD
:TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:FORMAT? -> :TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:FORMAT STD

:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:IDEXt?

機能　CAN バス信号トリガの OR 条件の各拡張フォーマットの ID に関するすべての設定値を問い合わせます。

構文　:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:IDEXt?
EVENt<x> の <x> = 1、2
ID<x> の <x> = 1 ～ 4

:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:IDEXt:HEXA

機能　CAN バス信号トリガの OR 条件の各拡張フォーマットの ID を HEXA で設定します。

構文　:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:IDEXt:HEXA {< 文字列 >}
EVENt<x> の <x> = 1、2
ID<x> の <x> = 1 ～ 4
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 8 文字

例　:TRIGGER:EINTERVAL:EVENT1:CANBUS:IDOR:ID1:IDEXT:HEXA "1AEF5906"

:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:IDEXt:PATTern

機能　CAN バス信号トリガの OR 条件の各拡張フォーマットの ID を BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:CANBus:
IDOR:ID<x>:IDEXt:PATTern {< 文字列 >}
:TRIGger:EINTerval:EVENt<x>:CANBus:
IDOR:ID<x>:IDEXt:PATTern?
```
EVENt<x> の <x> = 1、2
ID<x> の <x> = 1 ～ 4
< 文字列 > = '0'、'1'、'X' の組み合わせ 29 文字

例
```
:TRIGGER:EINTERVAL:EVENT1:CANBUS:
IDOR:ID1:IDEXT:PATTERN
"1100101101110
0001110111011111"
:TRIGGER:EINTERVAL:EVENT1:CANBUS:
IDOR:ID1:IDEXT:PATTERN? -> :TRIGGER:
EINTERVAL:EVENT1:CANBUS:IDOR:ID1:
IDEXT:PATTERN
"11001011011100000111011
1011111"
```

:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:IDSTd?

機能　CAN バス信号トリガの OR 条件の各標準フォーマットの ID に関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:CANBus:
IDOR:ID<x>:IDSTd?
```
EVENt<x> の <x> = 1、2
ID<x> の <x> = 1 ～ 4

:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:IDSTd:HEXA

機能　CAN バス信号トリガの OR 条件の各標準フォーマットの ID を HEXA で設定します。

構文
```
:TRIGger:EINTerval:EVENt<x>:CANBus:
IDOR:ID<x>:IDSTd:HEXA {< 文字列 >}
```
EVENt<x> の <x> = 1、2
ID<x> の <x> = 1 ～ 4
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 3 文字

例
```
:TRIGGER:EINTERVAL:EVENT1:CANBUS:
IDOR:ID1:IDSTD:HEXA "5DF"
```

:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:IDSTd:PATTern

機能　CAN バス信号トリガの OR 条件の各標準フォーマットの ID を BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:CANBus:
IDOR:ID<x>:IDSTd:PATTern {< 文字列 >}
:TRIGger:EINTerval:EVENt<x>:CANBus:
IDOR:ID<x>:IDSTd:PATTern?
```
EVENt<x> の <x> = 1、2
ID<x> の <x> = 1 ～ 4
< 文字列 > = '0'、'1'、'X' の組み合わせ 11 文字

例
```
:TRIGGER:EINTERVAL:EVENT1:CANBUS:
IDOR:ID1:IDSTD:PATTERN "10111011111"
:TRIGGER:EINTERVAL:EVENT1:CANBUS:
IDOR:ID1:IDSTD:PATTERN? -> :TRIGGER:
EINTERVAL:EVENT1:CANBUS:IDOR:ID1:
IDSTD:PATTERN "10111011111"
```

:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:MODE

機能　CAN バス信号トリガの OR 条件の各条件の有効 (1)/ 無効 (0) を設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:CANBus:
IDOR:ID<x>:MODE {<Boolean>}
:TRIGger:EINTerval:EVENt<x>:CANBus:
IDOR:ID<x>:MODE?
```
EVENt<x> の <x> = 1、2
ID<x> の <x> = 1 ～ 4

例
```
:TRIGGER:EINTERVAL:EVENT1:CANBUS:
IDOR:ID1:MODE ON
:TRIGGER:EINTERVAL:EVENT1:CANBUS:
IDOR:ID1:MODE? -> :TRIGGER:EINTERVAL:
EVENT1:CANBUS:IDOR:ID1:MODE 1
```

:TRIGger:EINTerval:EVENt<x>:CANBus:IDOR:ID<x>:RTR

機能　CAN バス信号トリガの OR 条件の各 RTR を設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:CANBus:
IDOR:ID<x>:RTR
{DATA|DONTcare|REMote}
:TRIGger:EINTerval:EVENt<x>:CANBus:
IDOR:ID<x>:RTR?
```
EVENt<x> の <x> = 1、2
ID<x> の <x> = 1 ～ 4

例
```
:TRIGGER:EINTERVAL:EVENT1:CANBUS:
IDOR:ID1:RTR DATA
:TRIGGER:EINTERVAL:EVENT1:CANBUS:
IDOR:ID1:RTR? -> :TRIGGER:EINTERVAL:
EVENT1:CANBUS:IDOR:ID1:RTR DATA
```

:TRIGger:EINTerval:EVENt<x>:CANBus:IDSTd?

機能　CAN バス信号トリガの標準フォーマットの ID に関するすべての設定値を問い合わせます。

構文　:TRIGger:EINTerval:EVENt<x>:CANBus:IDSTd?
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:CANBus:IDSTd:HEXA

機能　CAN バス信号トリガの標準フォーマットの ID を HEXA で設定します。

構文　:TRIGger:EINTerval:EVENt<x>:CANBus:IDSTd:HEXA {< 文字列 >}
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 3 文字

例　:TRIGGER:EINTERVAL:EVENT1:CANBUS:IDSTD:HEXA "5DF"

:TRIGger:EINTerval:EVENt<x>:CANBus:IDSTd:PATTern

機能　CAN バス信号トリガの標準フォーマットの ID を BINARY で設定 / 問い合わせします。

構文　:TRIGger:EINTerval:EVENt<x>:CANBus:IDSTd:PATTern {< 文字列 >}
:TRIGger:EINTerval:EVENt<x>:CANBus:IDSTd:PATTern?
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 11 文字

例　:TRIGGER:EINTERVAL:EVENT1:CANBUS:IDSTD:PATTERN "10111011111"
:TRIGGER:EINTERVAL:EVENT1:CANBUS:IDSTD:PATTERN? -> :TRIGGER:EINTERVAL:EVENT1:CANBUS:IDSTD:PATTERN "10111011111"

:TRIGger:EINTerval:EVENt<x>:CANBus:MODE

機能　CAN バス信号トリガのモードを設定 / 問い合わせします。

構文　:TRIGger:EINTerval:EVENt<x>:CANBus:MODE {EFRame|IDEXt|IDOR|IDSTd|SOF}
:TRIGger:EINTerval:EVENt<x>:CANBus:MODE?
<x> = 1、2

例　:TRIGGER:EINTERVAL:EVENT1:CANBUS:MODE EFRAME
:TRIGGER:EINTERVAL:EVENT1:CANBUS:MODE? -> :TRIGGER:EINTERVAL:EVENT1:CANBUS:MODE EFRAME

:TRIGger:EINTerval:EVENt<x>:CANBus:RECessive

機能　CAN バス信号トリガのリセッシブレベル (バスレベル) を設定 / 問い合わせします。

構文　:TRIGger:EINTerval:EVENt<x>:CANBus:RECessive {HIGH|LOW}
:TRIGger:EINTerval:EVENt<x>:CANBus:RECessive?
<x> = 1、2

例　:TRIGGER:EINTERVAL:EVENT1:CANBUS:RECESSIVE HIGH
:TRIGGER:EINTERVAL:EVENT1:CANBUS:RECESSIVE? -> :TRIGGER:EINTERVAL:EVENT1:CANBUS:RECESSIVE HIGH

:TRIGger:EINTerval:EVENt<x>:CANBus:RTR

機能　CAN バス信号トリガの RTR を設定 / 問い合わせします。

構文　:TRIGger:EINTerval:EVENt<x>:CANBus:RTR {DATA|DONTcare|REMote}
:TRIGger:EINTerval:EVENt<x>:CANBus:RTR?
<x> = 1、2

例　:TRIGGER:EINTERVAL:EVENT1:CANBUS:RTR DATA
:TRIGGER:EINTERVAL:EVENT1:CANBUS:RTR? -> :TRIGGER:EINTERVAL:EVENT1:CANBUS:RTR DATA

:TRIGger:EINTerval:EVENt<x>:CANBus:SOURce

機能　CAN バス信号トリガのトリガソースを設定 / 問い合わせします。

構文　:TRIGger:EINTerval:EVENt<x>:CANBus:SOURce {<NRf>}
:TRIGger:EINTerval:EVENt<x>:CANBus:SOURce?
<x> = 1、2
<NRf> = 1 ～ 4

例　:TRIGGER:EINTERVAL:EVENT1:CANBUS:SOURCE 1
:TRIGGER:EINTERVAL:EVENT1:CANBUS:SOURCE? -> :TRIGGER:EINTERVAL:EVENT1:CANBUS:SOURCE 1

:TRIGger:EINTerval:EVENt<x>:CANBus:SPOint

機能　CAN バス信号トリガのサンプルポイントを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:CANBus:SPOint {<NRf>}
:TRIGger:EINTerval:EVENt<x>:CANBus:SPOint?
```
<x> = 1、2
<NRf> = 18.8 ～ 90.6(%)

例
```
:TRIGGER:EINTERVAL:EVENT1:CANBUS:SPOINT 18.8
:TRIGGER:EINTERVAL:EVENT1:CANBUS:SPOINT? -> :TRIGGER:EINTERVAL:EVENT1:CANBUS:SPOINT 18.8E+00
```

:TRIGger:EINTerval:EVENt<x>:I2CBus?

機能　各イベントの I²C バストリガに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:I2CBus?
```
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa?

機能　I²C バストリガのアドレスに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa?
```
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT10address?

機能　I²C バストリガの 10bit アドレスに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT10address?
```
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT10address:HEXA

機能　I²C バストリガの 10bit アドレスを HEXA で設定します。

構文
```
:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT10address:HEXA {< 文字列 >}
```
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 3 文字 (ビット 8 は、R/$\overline{W}$ ビット)

例
```
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:ADATA:BIT10ADDRESS:HEXA "7AB"
```

:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT10address:PATTern

機能　I²C バストリガの 10bit アドレスを BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT10address:PATTern {< 文字列 >}
:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT10address:PATTern?
```
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 11 文字 (ビット 8 は、R/$\overline{W}$ ビット)

例
```
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:ADATA:BIT10ADDRESS:PATTERN "10111011111"
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:ADATA:BIT10ADDRESS:PATTERN?
-> :TRIGGER:EINTERVAL:EVENT1:I2CBUS:ADATA:BIT10ADDRESS:PATTERN "10111011111"
```

:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7ADdress?

機能　I²C バストリガの 7bit アドレスに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7ADdress?
```
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7ADdress:HEXA

機能　I²C バストリガの 7bit アドレスを HEXA で設定します。

構文　:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7ADdress:HEXA {< 文字列 >}
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字 (ビット 0 は、R/$\overline{W}$ ビット)

例　:TRIGGER:EINTERVAL:EVENT1:I2CBUS:ADATA:BIT7ADDRESS:HEXA "DE"

:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7ADdress:PATTern

機能　I²C バストリガの 7bit アドレスを BINARY で設定 / 問い合わせします。

構文　:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7ADdress:PATTern {< 文字列 >}
:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7ADdress:PATTern?
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 8 文字 (ビット 0 は、R/$\overline{W}$ ビット)

例　:TRIGGER:EINTERVAL:EVENT1:I2CBUS:ADATA:BIT7ADDRESS:PATTERN "11011110"
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:ADATA:BIT7ADDRESS:PATTERN?
-> :TRIGGER:EINTERVAL:EVENT1:I2CBUS:ADATA:BIT7ADDRESS:PATTERN "11011110"

:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7APsub?

機能　I²C バストリガの 7bit + Sub アドレスに関するすべての設定値を問い合わせます。

構文　:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7APsub?
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7APsub:ADDRess?

機能　I²C バストリガの 7bit + Sub アドレスの 7bit アドレスに関するすべての設定値を問い合わせます。

構文　:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7APsub:ADDRess?
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7APsub:ADDRess:HEXA

機能　I²C バストリガの 7bit + Sub アドレスの 7bit アドレスを HEXA で設定します。

構文　:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7APsub:ADDRess:HEXA {< 文字列 >}
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字 (ビット 0 は、R/$\overline{W}$ ビット)

例　:TRIGGER:EINTERVAL:EVENT1:I2CBUS:ADATA:BIT7APSUB:ADDRESS:HEXA "AB"

:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7APsub:ADDRess:PATTern

機能　I²C バストリガの 7bit + Sub アドレスの 7bit アドレスを BINARY で設定 / 問い合わせします。

構文　:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7APsub:ADDRess:PATTern {< 文字列 >}
:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7APsub:ADDRess:PATTern?
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 8 文字 (ビット 0 は、R/$\overline{W}$ ビット)

例　:TRIGGER:EINTERVAL:EVENT1:I2CBUS:ADATA:BIT7APSUB:ADDRESS:PATTERN "10101011"
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:ADATA:BIT7APSUB:ADDRESS:PATTERN?
-> :TRIGGER:EINTERVAL:EVENT1:I2CBUS:ADATA:BIT7APSUB:ADDRESS:PATTERN "10101011"

:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7APsub:SADDress?

機能　I²C バストリガの 7bit + Sub アドレスの Sub アドレスに関するすべての設定値を問い合わせます。

構文　:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7APsub:SADDress?
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7APsub:SADDress:HEXA

機能　I²C バストリガの 7bit + Sub アドレスの Sub アドレスを HEXA で設定します。

構文　:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7APsub:SADDress:HEXA {< 文字列 >}
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字

例　:TRIGGER:EINTERVAL:EVENT1:I2CBUS:ADATA:BIT7APSUB:SADDRESS:HEXA "EF"

:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:BIT7APsub:SADDress:PATTern

機能　I²C バストリガの 7bit + Sub アドレスの Sub アドレスを BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:I2CBus:
ADATa:BIT7APsub:SADDress:
PATTern {< 文字列 >}
:TRIGger:EINTerval:EVENt<x>:I2CBus:
ADATa:BIT7APsub:SADDress:PATTern?
```
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 8 文字

例
```
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:
ADATA:BIT7APSUB:SADDRESS:
PATTERN "10101011"
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:
ADATA:BIT7APSUB:SADDRESS:PATTERN?
-> :TRIGGER:EINTERVAL:EVENT1:I2CBUS:
ADATA:BIT7APSUB:SADDRESS:
PATTERN "10101011"
```

:TRIGger:EINTerval:EVENt<x>:I2CBus:ADATa:TYPE

機能　I²C バストリガのアドレスの種類を設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:I2CBus:
ADATa:TYPE {BIT10address|BIT7ADdress
|
BIT7APsub}
:TRIGger:EINTerval:EVENt<x>:I2CBus:
ADATa:TYPE?
```
<x> = 1、2

例
```
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:
ADATA:TYPE BIT10ADDRESS
:TRIGGER:EINTERVAL:EVENT1:
I2CBUS:ADATA:TYPE? -> :TRIGGER:
EINTERVAL:EVENT1:I2CBUS:ADATA:TYPE
BIT10ADDRESS
```

:TRIGger:EINTerval:EVENt<x>:I2CBus:CLOCk?

機能　I²C バストリガのクロックに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:I2CBus:
CLOCk?
```
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:I2CBus:CLOCk:SOURce

機能　I²C バストリガのクロックトレースを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:I2CBus:
CLOCk:SOURce {<NRf>}
:TRIGger:EINTerval:EVENt<x>:I2CBus:
CLOCk:SOURce?
```
<x> = 1、2
<NRf> = 1 ～ 4

例
```
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:
CLOCK:SOURCE 1
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:
CLOCK:SOURCE? -> :TRIGGER:EINTERVAL:
EVENT1:I2CBUS:CLOCK:SOURCE 1
```

:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA?

機能　I²C バストリガのデータに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:I2CBus:
DATA?
```
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA:BYTE

機能　I²C バストリガの設定データ数を設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:I2CBus:
DATA:BYTE {<NRf>}
:TRIGger:EINTerval:EVENt<x>:I2CBus:
DATA:BYTE?
```
<x> = 1、2
<NRf> = 1 ～ 4

例
```
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:
DATA:BYTE 1
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:
DATA:BYTE? -> :TRIGGER:EINTERVAL:
EVENT1:I2CBUS:DATA:BYTE 1
```

:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA:CONDition

機能　I²C バストリガのデータの判定方法 (一致 / 不一致) を設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:I2CBus:
DATA:CONDition {FALSe|TRUE}
:TRIGger:EINTerval:EVENt<x>:I2CBus:
DATA:CONDition?
```
<x> = 1、2

例
```
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:
DATA:CONDITION TRUE
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:
DATA:CONDITION? -> :TRIGGER:
EINTERVAL:EVENT1:I2CBUS:DATA:
CONDITION TRUE
```

:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA:DPOSition

機能　I²C バストリガのデータのパターン比較する位置を設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA:DPOSition {<NRf>}
:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA:DPOSition?
```
<x> = 1、2
<NRf> = 0 ～ 9999

例
```
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:DATA:DPOSITION 1
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:DATA:DPOSITION? -> :TRIGGER:EINTERVAL:EVENT1:I2CBUS:DATA:DPOSITION 1
```

:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA:HEXA<x>

機能　I²C バストリガのデータを HEXA で設定します。

構文
```
:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA:HEXA<x> {< 文字列 >}
```
EVENt<x> の <x> = 1、2
HEXA<x> の <x> = 1 ～ 4
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字

例
```
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:DATA:HEXA1 "AB"
```

:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA:MODE

機能　I²C バストリガのデータ条件の有効 / 無効を設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA:MODE {<Boolean>}
:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA:MODE?
```
<x> = 1、2

例
```
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:DATA:MODE ON
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:DATA:MODE? -> :TRIGGER:EINTERVAL:EVENT1:I2CBUS:DATA:MODE 1
```

:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA:PATTern<x>

機能　I²C バストリガのデータを BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA:PATTern<x> {< 文字列 >}
:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA:PATTern<x>?
```
EVENt<x> の <x> = 1、2
PATTern<x> の <x> = 1 ～ 4
< 文字列 > = '0'、'1'、'X' の組み合わせ 8 文字

例
```
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:DATA:PATTERN1 "10101011"
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:DATA:PATTERN1? -> :TRIGGER:EINTERVAL:EVENT1:I2CBUS:DATA:PATTERN1 "10101011"
```

:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA:PMODe

機能　I²C バストリガのデータのパターン比較先頭位置モードを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA:PMODe {DONTcare|SELect}
:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA:PMODe?
```
<x> = 1、2

例
```
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:DATA:PMODE SELECT
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:DATA:PMODE? -> :TRIGGER:EINTERVAL:EVENT1:I2CBUS:DATA:PMODE SELECT
```

:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA:SOURce

機能　I²C バストリガのデータトレースを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA:SOURce {<NRf>}
:TRIGger:EINTerval:EVENt<x>:I2CBus:DATA:SOURce?
```
<x> = 1、2
<NRf> = 1 ～ 4

例
```
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:DATA:SOURCE 1
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:DATA:SOURCE? -> :TRIGGER:EINTERVAL:EVENT1:I2CBUS:DATA:SOURCE 1
```

:TRIGger:EINTerval:EVENt<x>:I2CBus:GCALl?

機能　I²C バストリガのジェネラルコールに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:I2CBus:GCALl?
```
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:I2CBus:GCALl:BIT7maddress?

機能　I²C バストリガのジェネラルコールの 7bit マスタアドレスに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:I2CBus:
GCALl:BIT7maddress?
```
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:I2CBus:GCALl:BIT7maddress:HEXA

機能　I²C バストリガのジェネラルコールの 7bit マスタアドレスを HEXA で設定します。

構文
```
:TRIGger:EINTerval:EVENt<x>:I2CBus:
GCALl:BIT7maddress:HEXA {< 文字列 >}
```
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字 (ビット 0 は、'1' に固定)

例
```
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:
GCALL:BIT7MADDRESS:HEXA "AB"
```

:TRIGger:EINTerval:EVENt<x>:I2CBus:GCALl:BIT7maddress:PATTern

機能　I²C バストリガのジェネラルコールの 7bit マスタアドレスを BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:I2CBus:
GCALl:BIT7maddress:PATTern {< 文字列 >}
:TRIGger:EINTerval:EVENt<x>:I2CBus:
GCALl:BIT7maddress:PATTern?
```
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 7 文字

例
```
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:
GCALL:BIT7MADDRESS:PATTERN "1010101"
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:
GCALL:BIT7MADDRESS:PATTERN?
-> :TRIGGER:EINTERVAL:EVENT1:I2CBUS:
GCALL:BIT7MADDRESS:PATTERN "1010101"
```

:TRIGger:EINTerval:EVENt<x>:I2CBus:GCALl:SBYTe (Second Byte)

機能　I²C バストリガのジェネラルコールのセカンドバイトのタイプを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:I2CBus:
GCALl:SBYTe {BIT7maddress|DONTcare|
H04|H06}
:TRIGger:EINTerval:EVENt<x>:I2CBus:
GCALl:SBYTe?
```
<x> = 1、2

例
```
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:
GCALL:SBYTE BIT7MADDRESS
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:
GCALL:SBYTE? -> :TRIGGER:EINTERVAL:
EVENT1:I2CBUS:GCALL:
SBYTE BIT7MADDRESS
```

:TRIGger:EINTerval:EVENt<x>:I2CBus:MODE

機能　I²C バストリガのトリガモードを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:I2CBus:
MODE {ADATa|ESTart|GCALl|NAIGnore|
SBHSmode}
:TRIGger:EINTerval:EVENt<x>:I2CBus:
MODE?
```
<x> = 1、2

例
```
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:
MODE ADATA
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:
MODE? -> :TRIGGER:EINTERVAL:EVENT1:
I2CBUS:MODE ADATA
```

:TRIGger:EINTerval:EVENt<x>:I2CBus:NAIGnore?

機能　I²C バストリガの NON ACK 無視モードに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:I2CBus:
NAIGnore?
```
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:I2CBus:NAIGnore:HSMode

機能　I²C バストリガのハイスピードモードで NON ACK を無視する / しないを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:I2CBus:
NAIGnore:HSMode {<Boolean>}
:TRIGger:EINTerval:EVENt<x>:I2CBus:
NAIGnore:HSMode?
```
<x> = 1、2

例
```
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:
NAIGNORE:HSMODE ON
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:
NAIGNORE:HSMODE? -> :TRIGGER:
EINTERVAL:EVENT1:I2CBUS:NAIGNORE:
HSMODE 1
```

:TRIGger:EINTerval:EVENt<x>:I2CBus:NAIGnore:RACCess

機能　I²C バストリガのリードアクセスモードで NON ACK を無視する / しないを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:I2CBus:
NAIGnore:RACCess {<Boolean>}
:TRIGger:EINTerval:EVENt<x>:I2CBus:
NAIGnore:RACCess?
```
<x> = 1、2

例
```
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:
NAIGNORE:RACCESS ON
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:
NAIGNORE:RACCESS? -> :TRIGGER:
EINTERVAL:EVENT1:I2CBUS:
NAIGNORE:RACCESS 1
```

:TRIGger:EINTerval:EVENt<x>:I2CBus:NAIGnore:SBYTe (Start Byte)

機能　I²C バストリガのスタートバイトで NON ACK を無視する / しないを設定 / 問い合わせします。

構文　:TRIGger:EINTerval:EVENt<x>:I2CBus:NAIGnore:SBYTe {<Boolean>}
:TRIGger:EINTerval:EVENt<x>:I2CBus:NAIGnore:SBYTe?
<x> = 1、2

例　:TRIGGER:EINTERVAL:EVENT1:I2CBUS:NAIGNORE:SBYTE ON
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:NAIGNORE:SBYTE? -> :TRIGGER:EINTERVAL:EVENT1:I2CBUS:NAIGNORE:SBYTE 1

:TRIGger:EINTerval:EVENt<x>:I2CBus:SBHSmode?

機能　I²C バストリガのスタートバイト / ハイスピードモードに関するすべての設定値を問い合わせます。

構文　:TRIGger:EINTerval:EVENt<x>:I2CBus:SBHSmode?
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:I2CBus:SBHSmode:TYPE

機能　I²C バストリガのスタートバイト / ハイスピードモードのタイプを設定 / 問い合わせします。

構文　:TRIGger:EINTerval:EVENt<x>:I2CBus:SBHSmode:TYPE {HSMode|SBYTe}
:TRIGger:EINTerval:EVENt<x>:I2CBus:SBHSmode:TYPE?
<x> = 1、2

例　:TRIGGER:EINTERVAL:EVENT1:I2CBUS:SBHSMODE:TYPE HSMODE
:TRIGGER:EINTERVAL:EVENT1:I2CBUS:SBHSMODE:TYPE? -> :TRIGGER:EINTERVAL:EVENT1:I2CBUS:SBHSMODE:TYPE HSMODE

:TRIGger:EINTerval:EVENt<x>:LINBus?

機能　各イベントの LIN バス信号トリガに関するすべての設定値を問い合わせします。

構文　:TRIGger:EINTerval:EVENt<x>:LINBus?
<x> = 1,2

:TRIGger:EINTerval:EVENt<x>:LINBus:BRATe

機能　LIN バス信号トリガのビットレート (データ転送速度) を設定 / 問い合わせします。

構文　:TRIGger:EINTerval:EVENt<x>:LINBus:BRATe {<NRf>|USER,<NRf>}
:TRIGger:EINTerval:EVENt<x>:LINBus:BRATe?
<x> = 1,2
<NRf> = 1200、2400、4800、9600、19200
USER の <NRf> =本体ユーザーズマニュアル参照。

例　:TRIGGER:EINTERVAL:EVENT1:LINBUS:BRATE 19200
:TRIGGER:EINTERVAL:EVENT1:LINBUS:BRATE? -> :TRIGGER:EINTERVAL:EVENT1:LINBUS:BRATE 19200

:TRIGger:EINTerval:EVENt<x>:LINBus:SOURce

機能　LIN バス信号トリガのトリガソースを設定 / 問い合わせします。

構文　:TRIGger:EINTerval:EVENt<x>:LINBus:SOURce {<NRf>}
:TRIGger:EINTerval:EVENt<x>:LINBus:SOURce?
<x> = 1,2
<NRf> = 1 ～ 4

例　:TRIGGER:EINTERVAL:EVENT1:LINBUS:SOURCE 1
:TRIGGER:EINTERVAL:EVENT1:LINBUS:SOURCE? -> :TRIGGER:EINTERVAL:EVENT1:LINBUS:SOURCE 1

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus?

機能　各イベントのロジック I²C バストリガに関するすべての設定値を問い合わせます。

構文　:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus?
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa?

機能　ロジック I²C バストリガのアドレスに関するすべての設定値を問い合わせます。

構文　:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa?
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:BIT10address?

機能　ロジック I²C バストリガの 10bit アドレスに関するすべての設定値を問い合わせます。

構文　:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:BIT10address?
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:BIT10address:HEXA

機能　ロジック I²C バストリガの 10bit アドレスを HEXA で設定します。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:ADATa:BIT10address:HEXA {< 文字列 >}
```
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 3 文字 (ビット 8 は、R/$\overline{W}$ ビット)

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:ADATA:BIT10ADDRESS:HEXA "7AB"
```

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:BIT10address:PATTern

機能　ロジック I²C バストリガの 10bit アドレスを BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:ADATa:BIT10address:
PATTern {< 文字列 >}
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:ADATa:BIT10address:PATTern?
```
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 11 文字 (ビット 8 は、R/$\overline{W}$ ビット)

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:ADATA:BIT10ADDRESS:
PATTERN "10111011111"
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:ADATA:BIT10ADDRESS:PATTERN?
-> :TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:ADATA:BIT10ADDRESS:
PATTERN "10111011111"
```

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:BIT7ADdress?

機能　ロジック I²C バストリガの 7bit アドレスに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:ADATa:BIT7ADdress?
```
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:BIT7ADdress:HEXA

機能　ロジック I²C バストリガの 7bit アドレスを HEXA で設定します。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:ADATa:BIT7ADdress:
HEXA {< 文字列 >}
```
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字 (ビット 0 は、R/$\overline{W}$ ビット)

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:ADATA:BIT7ADDRESS:HEXA "DE"
```

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:BIT7ADdress:PATTern

機能　ロジック I²C バストリガの 7bit アドレスを BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:ADATa:BIT7ADdress:
PATTern {< 文字列 >}
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:ADATa:BIT7ADdress:PATTern?
```
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 8 文字 (ビット 0 は、R/$\overline{W}$ ビット)

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:ADATA:BIT7ADDRESS:
PATTERN "11011110"
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:ADATA:BIT7ADDRESS:PATTERN?
-> :TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:ADATA:BIT7ADDRESS:
PATTERN "11011110"
```

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:BIT7APsub?

機能　ロジック I²C バストリガの 7bit + Sub アドレスに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:ADATa:BIT7APsub?
```
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:BIT7APsub:ADDRess?

機能　ロジック I²C バストリガの 7bit + Sub アドレスの 7bit アドレスに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:ADATa:BIT7APsub:ADDRess?
```
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:BIT7APsub:ADDRess:HEXA

機能　ロジック I²C バストリガの 7bit + Sub アドレスの 7bit アドレスを HEXA で設定します。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:ADATa:BIT7APsub:ADDRess:
HEXA {< 文字列 >}
```
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字 (ビット 0 は、R/$\overline{W}$ ビット)

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:ADATA:BIT7APSUB:ADDRESS:
HEXA "AB"
```

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:BIT7APsub:ADDRess:PATTern

機能　ロジック I²C バストリガの 7bit + Sub アドレスの 7bit アドレスを BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:ADATa:BIT7APsub:ADDRess:
PATTern {< 文字列 >}
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:ADATa:BIT7APsub:ADDRess:
PATTern?
```
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 8 文字 (ビット 0 は、R/$\overline{W}$ ビット)

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:ADATA:BIT7APSUB:ADDRESS:
PATTERN "10101011"
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:ADATA:BIT7APSUB:ADDRESS:
PATTERN? -> :TRIGGER:EINTERVAL:
EVENT1:LOGIC:I2CBUS:ADATA:BIT7APSUB:
ADDRESS:PATTERN "10101011"
```

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:BIT7APsub:SADDress?

機能　ロジック I²C バストリガの 7bit + Sub アドレスの Sub アドレスに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:ADATa:BIT7APsub:SADDress?
```
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:BIT7APsub:SADDress:HEXA

機能　ロジック I²C バストリガの 7bit + Sub アドレスの Sub アドレスを HEXA で設定します。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:ADATa:BIT7APsub:SADDress:
HEXA {< 文字列 >}
```
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:ADATA:BIT7APSUB:SADDRESS:
HEXA "EF"
```

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:BIT7APsub:SADDress:PATTern

機能　ロジック I²C バストリガの 7bit + Sub アドレスの Sub アドレスを BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:ADATa:BIT7APsub:SADDress:
PATTern {< 文字列 >}
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:ADATa:BIT7APsub:SADDress:
PATTern?
```
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 8 文字

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:ADATA:BIT7APSUB:SADDRESS:
PATTERN "10101011"
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:ADATA:BIT7APSUB:SADDRESS:
PATTERN? -> :TRIGGER:EINTERVAL:
EVENT1:LOGIC:I2CBUS:ADATA:BIT7APSUB:
SADDRESS:PATTERN "10101011"
```

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:ADATa:TYPE

機能　ロジック I²C バストリガのアドレスの種類を設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:ADATa:TYPE {BIT10address|
BIT7ADdress|BIT7APsub}
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:ADATa:TYPE?
```
<x> = 1、2

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:ADATA:TYPE BIT10ADDRESS
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:ADATA:TYPE? -> :TRIGGER:
EINTERVAL:EVENT1:LOGIC:I2CBUS:ADATA:
TYPE BIT10ADDRESS
```

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:CLOCk?

機能　ロジック I²C バストリガのクロックに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:CLOCk?
```
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:CLOCk:SOURce

機能　ロジック I²C バストリガのクロックトレースを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:CLOCk:SOURce {A<y>|B<y>|C<y>|D<y>}
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:CLOCk:SOURce?
```
<x> = 1、2
<y> = 0 ～ 7

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:I2CBUS:CLOCK:SOURCE A0
:TRIGGER:EINTERVAL:EVENT1:LOGIC:I2CBUS:CLOCK:SOURCE? -> :TRIGGER:EINTERVAL:EVENT1:LOGIC:I2CBUS:CLOCK:SOURCE A0
```

解説　DLM6000 の 16 ビットモデルでは {A<y>|C<y>} が有効です。

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA?

機能　ロジック I²C バストリガのデータに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA?
```
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:BYTE

機能　ロジック I²C バストリガの設定データ数を設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:BYTE {<NRf>}
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:BYTE?
```
<x> = 1、2
<NRf> = 1 ～ 4

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:I2CBUS:DATA:BYTE 1
:TRIGGER:EINTERVAL:EVENT1:LOGIC:I2CBUS:DATA:BYTE? -> :TRIGGER:EINTERVAL:EVENT1:LOGIC:I2CBUS:DATA:BYTE 1
```

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:CONDition

機能　ロジック I²C バストリガのデータの判定方法 (一致 / 不一致) を設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:CONDition {FALSe|TRUE}
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:CONDition?
```
<x> = 1、2

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:I2CBUS:DATA:CONDITION FALSE
:TRIGGER:EINTERVAL:EVENT1:LOGIC:I2CBUS:DATA:CONDITION? -> :TRIGGER:EINTERVAL:EVENT1:LOGIC:I2CBUS:DATA:CONDITION FALSE
```

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:DPOSition

機能　ロジック I²C バストリガのデータのパターン比較する位置を設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:DPOSition {<NRf>}
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:DPOSition?
```
<x> = 1、2
<NRf> = 0 ～ 9999

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:I2CBUS:DATA:DPOSITION 1
:TRIGGER:EINTERVAL:EVENT1:LOGIC:I2CBUS:DATA:DPOSITION? -> :TRIGGER:EINTERVAL:EVENT1:LOGIC:I2CBUS:DATA:DPOSITION 1
```

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:HEXA<x>

機能　ロジック I²C バストリガのデータを HEXA で設定します。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:HEXA<x> {< 文字列 >}
```
EVENt<x> の <x> = 1、2
HEXA<x> の <x> = 1 ～ 4
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:I2CBUS:DATA:HEXA1 "AB"
```

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:MODE

機能　ロジック I²C バストリガのデータ条件の有効 / 無効を設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:MODE {<Boolean>}
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:MODE?
```
<x> = 1、2

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:I2CBUS:DATA:MODE ON
:TRIGGER:EINTERVAL:EVENT1:LOGIC:I2CBUS:DATA:MODE? -> :TRIGGER:EINTERVAL:EVENT1:LOGIC:I2CBUS:DATA:MODE 1
```

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:PATTern<x>

機能　ロジック I²C バストリガのデータを BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:PATTern<x> {< 文字列 >}
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:PATTern<x>?
```
EVENt<x> の <x> = 1、2
PATTern<x> の <x> = 1 ～ 4
< 文字列 > = '0'、'1'、'X' の組み合わせ 8 文字

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:I2CBUS:DATA:PATTERN1 "10101011"
:TRIGGER:EINTERVAL:EVENT1:LOGIC:I2CBUS:DATA:PATTERN1? -> :TRIGGER:EINTERVAL:EVENT1:LOGIC:I2CBUS:DATA:PATTERN1 "10101011"
```

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:PMODe

機能　ロジック I²C バストリガのデータのパターン比較先頭位置モードを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:PMODe {DONTcare|SELect}
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:PMODe?
```
<x> = 1、2

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:I2CBUS:DATA:PMODE DONTCARE
:TRIGGER:EINTERVAL:EVENT1:LOGIC:I2CBUS:DATA:PMODE? -> :TRIGGER:EINTERVAL:EVENT1:LOGIC:I2CBUS:DATA:PMODE DONTCARE
```

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:SOURce

機能　ロジック I²C バストリガのデータトレースを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:SOURce {A<y>|B<y>|C<y>|D<y>}
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:DATA:SOURce?
```
<x> = 1、2
<y> = 0 ～ 7

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:I2CBUS:DATA:SOURCE A0
:TRIGGER:EINTERVAL:EVENT1:LOGIC:I2CBUS:DATA:SOURCE? -> :TRIGGER:EINTERVAL:EVENT1:LOGIC:I2CBUS:DATA:SOURCE A0
```

解説　DLM6000 の 16 ビットモデルでは {A<y>|C<y>} が有効です。

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:GCALl?

機能　ロジック I²C バストリガのジェネラルコールに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:GCALl?
```
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:GCALl:BIT7maddress?

機能　ロジック I²C バストリガのジェネラルコールの 7bit マスタアドレスに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:GCALl:BIT7maddress?
```
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:GCALl:BIT7maddress:HEXA

機能　ロジック I²C バストリガのジェネラルコールの 7bit マスタアドレスを HEXA で設定します。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:GCALl:BIT7maddress:HEXA {< 文字列 >}
```
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字 (ビット 0 は、'1' に固定)

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:I2CBUS:GCALL:BIT7MADDRESS:HEXA "AB"
```

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:GCALl:BIT7maddress:PATTern

機能　ロジック I²C バストリガのジェネラルコールの 7bit マスタアドレスを BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:GCALl:BIT7maddress:
PATTern {< 文字列 >}
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:GCALl:BIT7maddress:PATTern?
```
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 7 文字

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:GCALL:BIT7MADDRESS:
PATTERN "1010101"
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:GCALL:BIT7MADDRESS:PATTERN?
-> :TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:GCALL:BIT7MADDRESS:
PATTERN "1010101"
```

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:GCALl:SBYTe (Second Byte)

機能　ロジック I²C バストリガのジェネラルコールのセカンドバイトのタイプを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:GCALl:SBYTe {BIT7maddress|
DONTcare|H04|H06}
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:GCALl:SBYTe?
```
<x> = 1、2

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:GCALL:SBYTE BIT7MADDRESS
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:GCALL:SBYTE? -> :TRIGGER:
EINTERVAL:EVENT1:LOGIC:I2CBUS:GCALL:
SBYTE BIT7MADDRESS
```

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:MODE

機能　ロジック I²C バストリガのトリガモードを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:MODE {ADATa|ESTart|GCALl|
NAIGnore|SBHSmode}
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:MODE?
```
<x> = 1、2

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:MODE ADATA
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:MODE? -> :TRIGGER:EINTERVAL:
EVENT1:LOGIC:I2CBUS:MODE ADATA
```

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:NAIGnore?

機能　ロジック I²C バストリガの NON ACK 無視モードに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:NAIGnore?
```
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:NAIGnore:HSMode

機能　ロジック I²C バストリガのハイスピードモードで NON ACK を無視する / しないを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:NAIGnore:HSMode {<Boolean>}
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:NAIGnore:HSMode?
```
<x> = 1、2

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:NAIGNORE:HSMODE ON
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:NAIGNORE:HSMODE? -> :TRIGGER:
EINTERVAL:EVENT1:LOGIC:I2CBUS:
NAIGNORE:HSMODE 1
```

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:NAIGnore:RACCess

機能　ロジック I²C バストリガのリードアクセスモードで NON ACK を無視する / しないを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:NAIGnore:RACCess {<Boolean>}
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:NAIGnore:RACCess?
```
<x> = 1、2

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:NAIGNORE:RACCESS ON
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:NAIGNORE:RACCESS? -> :TRIGGER:
EINTERVAL:EVENT1:LOGIC:I2CBUS:
NAIGNORE:RACCESS 1
```

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:NAIGnore:SBYTe (Start Byte)

機能　ロジック I²C バストリガのスタートバイトで NON ACK を無視する / しないを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:NAIGnore:SBYTe {<Boolean>}
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:NAIGnore:SBYTe?
```
<x> = 1、2

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:NAIGNORE:SBYTE ON
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:NAIGNORE:SBYTE? -> :TRIGGER:
EINTERVAL:EVENT1:LOGIC:I2CBUS:
NAIGNORE:SBYTE 1
```

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:SBHSmode?

機能　ロジック I²C バストリガのスタートバイト / ハイスピードモードに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:SBHSmode?
```
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:LOGic:I2CBus:SBHSmode:TYPE

機能　ロジック I²C バストリガのスタートバイト / ハイスピードモードのタイプを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:SBHSmode:TYPE {HSMode|SBYTe}
:TRIGger:EINTerval:EVENt<x>:LOGic:
I2CBus:SBHSmode:TYPE?
```
<x> = 1、2

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:SBHSMODE:TYPE HSMODE
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
I2CBUS:SBHSMODE:TYPE? -> :TRIGGER:
EINTERVAL:EVENT1:LOGIC:I2CBUS:
SBHSMODE:TYPE HSMODE
```

:TRIGger:EINTerval:EVENt<x>:LOGic:LINBus?

機能　各イベントのロジック LIN バス信号トリガに関するすべての設定値を問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
LINBus?
```
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:LOGic:LINBus:BRATe

機能　ロジック LIN バス信号トリガのビットレート (データ転送速度) を設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
LINBus:BRATe {<NRf>|USER,<NRf>}
:TRIGger:EINTerval:EVENt<x>:LOGic:
LINBus:BRATe?
```
<x> = 1、2
<NRf> = 1200、2400、4800、9600、19200
USER の <NRf> =本体ユーザーズマニュアル参照。

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
LINBUS:BRATE 19200
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
LINBUS:BRATE? -> :TRIGGER:EINTERVAL:
EVENT1:LOGIC:LINBUS:BRATE 19200
```

:TRIGger:EINTerval:EVENt<x>:LOGic:LINBus:SOURce

機能　ロジック LIN バス信号トリガのトリガソースを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
LINBus:SOURce {A<y>|B<y>|C<y>|D<y>}
:TRIGger:EINTerval:EVENt<x>:LOGic:
LINBus:SOURce?
```
<x> = 1、2
<y> = 0 ～ 7

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
LINBUS:SOURCE A0
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
LINBUS:SOURCE? -> :TRIGGER:EINTERVAL:
EVENT1:LOGIC:LINBUS:SOURCE A0
```

解説　DLM6000 の 16 ビットモデルでは {A<y>|C<y>} が有効です。

:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus?

機能　各イベントのロジック SPI バストリガに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
SPIBus?
```
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:BITorder

機能　ロジック SPI バストリガのビットオーダーを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
SPIBus:BITorder {LSBFirst|MSBFirst}
:TRIGger:EINTerval:EVENt<x>:LOGic:
SPIBus:BITorder?
```
<x> = 1、2

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
SPIBUS:BITORDER LSBFIRST
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
SPIBUS:BITORDER? -> :TRIGGER:
EINTERVAL:EVENT1:LOGIC:SPIBUS:
BITORDER LSBFIRST
```

:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:CLOCk?

機能　ロジック SPI バストリガのクロックに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
SPIBus:CLOCk?
```
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:CLOCk:POLarity

機能　ロジック SPI バストリガのクロックトレースの極性を設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
SPIBus:CLOCk:POLarity {FALL|RISE}
:TRIGger:EINTerval:EVENt<x>:LOGic:
SPIBus:CLOCk:POLarity?
```
<x> = 1、2

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
SPIBUS:CLOCK:POLARITY FALL
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
SPIBUS:CLOCK:POLARITY? -> :TRIGGER:
EINTERVAL:EVENT1:LOGIC:SPIBUS:CLOCK:
POLARITY FALL
```

:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:CLOCk:SOURce

機能　ロジック SPI バストリガのクロックトレースを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
SPIBus:CLOCk:SOURce {A<y>|B<y>|C<y>|
D<y>}
:TRIGger:EINTerval:EVENt<x>:LOGic:
SPIBus:CLOCk:SOURce?
```
<x> = 1、2
<y> = 0 ～ 7

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
SPIBUS:
CLOCK:SOURCE A0
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
SPIBUS:CLOCK:SOURCE? -> :TRIGGER:
EINTERVAL:EVENT1:LOGIC:SPIBUS:CLOCK:
SOURCE A0
```

解説　DLM6000 の 16 ビットモデルでは {A<y>|C<y>} が有効です。

:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:CS?

機能　ロジック SPI バストリガのチップセレクトに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
SPIBus:CS?
```
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:CS:ACTive

機能　ロジック SPI バストリガのチップセレクトのアクティブレベルを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:
SPIBus:CS:ACTive {HIGH|LOW}
:TRIGger:EINTerval:EVENt<x>:LOGic:
SPIBus:CS:ACTive?
```
<x> = 1、2

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
SPIBUS:CS:ACTIVE HIGH
:TRIGGER:EINTERVAL:EVENT1:LOGIC:
SPIBUS:CS:ACTIVE? -> :TRIGGER:
EINTERVAL:EVENT1:LOGIC:SPIBUS:CS:
ACTIVE HIGH
```

:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:CS:SOURce

機能　ロジック SPI バストリガのチップセレクトトレースを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:CS:SOURce {A<y>|B<y>|C<y>|D<y>}
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:CS:SOURce?
```
<x> = 1、2
<y> = 0 ～ 7

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:SPIBUS:CS:SOURCE A0
:TRIGGER:EINTERVAL:EVENT1:LOGIC:SPIBUS:CS:SOURCE? -> :TRIGGER:EINTERVAL:EVENT1:LOGIC:SPIBUS:CS:SOURCE A0
```

解説　DLM6000 の 16 ビットモデルでは {A<y>|C<y>} が有効です。

:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:DATA<x>?

機能　ロジック SPI バストリガの各データに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:DATA<x>?
```
EVENt<x> の <x> = 1、2
DATA<x> の <x> = 1、2

解説　DATA2 は、「:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:MODE WIRE4」のときに有効です。

:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:DATA<x>:BYTE

機能　ロジック SPI バストリガの各データの設定データ数を設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:DATA<x>:BYTE {<NRf>}
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:DATA<x>:BYTE?
```
EVENt<x> の <x> = 1、2
DATA<x> の <x> = 1、2
<NRf> = 1 ～ 4

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:SPIBUS:DATA1:BYTE 1
:TRIGGER:EINTERVAL:EVENT1:LOGIC:SPIBUS:DATA1:BYTE? -> :TRIGGER:EINTERVAL:EVENT1:LOGIC:SPIBUS:DATA1:BYTE 1
```

:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:DATA<x>:CONDition

機能　ロジック SPI バストリガの各データの判定方法 (一致 / 不一致) を設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:DATA<x>:CONDition {FALSe|TRUE}
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:DATA<x>:CONDition?
```
EVENt<x> の <x> = 1、2
DATA<x> の <x> = 1、2

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:SPIBUS:DATA1:CONDITION FALSE
:TRIGGER:EINTERVAL:EVENT1:LOGIC:SPIBUS:DATA1:CONDITION? -> :TRIGGER:EINTERVAL:EVENT1:LOGIC:SPIBUS:DATA1:CONDITION FALSE
```

:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:DATA<x>:DPOSition

機能　ロジック SPI バストリガの各データのパターン比較先頭位置を設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:DATA<x>:DPOSition {<NRf>}
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:DATA<x>:DPOSition?
```
EVENt<x> の <x> = 1、2
DATA<x> の <x> = 1、2
<NRf> = 0 ～ 9999

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:SPIBUS:DATA1:DPOSITION 1
:TRIGGER:EINTERVAL:EVENT1:LOGIC:SPIBUS:DATA1:DPOSITION? -> :TRIGGER:EINTERVAL:EVENT1:LOGIC:SPIBUS:DATA1:DPOSITION 1
```

:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:DATA<x>:HEXA<x>

機能　ロジック SPI バストリガの各データを HEXA で設定します。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:DATA<x>:HEXA<x> {< 文字列 >}
```
EVENt<x> の <x> = 1、2
DATA<x> の <x> = 1、2
HEXA<x> の <x> = 1 ～ 4
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字。

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:SPIBUS:DATA1:HEXA1 "AB"
```

:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:DATA<x>:PATTern<x>

機能　ロジック SPI バストリガの各データを BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:DATA<x>:PATTern<x> {< 文字列 >}
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:DATA<x>:PATTern<x>?
```
EVENt<x> の <x> = 1、2
DATA<x> の <x> = 1、2
PATTern<x> の <x> = 1 ～ 4
< 文字列 > = '0'、'1'、'X' の組み合わせ 8 文字。

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:SPIBUS:DATA1:PATTERN1 "10101011"
:TRIGGER:EINTERVAL:EVENT1:LOGIC:SPIBUS:DATA1:PATTERN1? -> :TRIGGER:EINTERVAL:EVENT1:LOGIC:SPIBUS:DATA1:PATTERN1 "10101011"
```

:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:DATA<x>:SOURce

機能　ロジック SPI バストリガの各データのトレースを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:DATA<x>:SOURce {A<y>|B<y>|C<y>|D<y>}
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:DATA<x>:SOURce?
```
EVENt<x> の <x> = 1、2
DATA<x> の <x> = 1、2
<y> = 0 ～ 7

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:SPIBUS:DATA1:SOURCE A0
:TRIGGER:EINTERVAL:EVENT1:LOGIC:SPIBUS:DATA1:SOURCE? -> :TRIGGER:EINTERVAL:EVENT1:LOGIC:SPIBUS:DATA1:SOURCE A0
```

解説　DLM6000 の 16 ビットモデルでは {A<y>|C<y>} が有効です。

:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:MODE

機能　ロジック SPI バストリガの結線方式 (3 線式 /4 線式) を設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:MODE {WIRE3|WIRE4}
:TRIGger:EINTerval:EVENt<x>:LOGic:SPIBus:MODE?
```
<x> = 1、2

例
```
:TRIGGER:EINTERVAL:EVENT1:LOGIC:SPIBUS:MODE WIRE3
:TRIGGER:EINTERVAL:EVENT1:LOGIC:SPIBUS:MODE? -> :TRIGGER:EINTERVAL:EVENT1:LOGIC:SPIBUS:MODE WIRE3
```

:TRIGger:EINTerval:EVENt<x>:SPIBus?

機能　各イベントの SPI バストリガに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:SPIBus?
```
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:SPIBus:BITorder

機能　SPI バストリガのビットオーダーを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:SPIBus:BITorder {LSBFirst|MSBFirst}
:TRIGger:EINTerval:EVENt<x>:SPIBus:BITorder?
```
<x> = 1、2

例
```
:TRIGGER:EINTERVAL:EVENT1:SPIBUS:BITORDER LSBFIRST
:TRIGGER:EINTERVAL:EVENT1:SPIBUS:BITORDER? -> :TRIGGER:EINTERVAL:EVENT1:SPIBUS:BITORDER LSBFIRST
```

:TRIGger:EINTerval:EVENt<x>:SPIBus:CLOCk?

機能　SPI バストリガのクロックに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:EINTerval:EVENt<x>:SPIBus:CLOCk?
```
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:SPIBus:CLOCk:POLarity

機能　SPI バストリガのクロックトレースの極性を設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:SPIBus:CLOCk:POLarity {FALL|RISE}
:TRIGger:EINTerval:EVENt<x>:SPIBus:CLOCk:POLarity?
```
<x> = 1、2

例
```
:TRIGGER:EINTERVAL:EVENT1:SPIBUS:CLOCK:POLARITY FALL
:TRIGGER:EINTERVAL:EVENT1:SPIBUS:CLOCK:POLARITY? -> :TRIGGER:EINTERVAL:EVENT1:SPIBUS:CLOCK:POLARITY FALL
```

:TRIGger:EINTerval:EVENt<x>:SPIBus:CLOCk:SOURce

機能　SPI バストリガのクロックトレースを設定 / 問い合わせします。

構文　:TRIGger:EINTerval:EVENt<x>:SPIBus:CLOCk:SOURce {<NRf>}
:TRIGger:EINTerval:EVENt<x>:SPIBus:CLOCk:SOURce?
<x> = 1、2
<NRf> = 1 ～ 4

例　:TRIGGER:EINTERVAL:EVENT1:SPIBUS:CLOCK:SOURCE 1
:TRIGGER:EINTERVAL:EVENT1:SPIBUS:CLOCK:SOURCE? -> :TRIGGER:EINTERVAL:EVENT1:SPIBUS:CLOCK:SOURCE 1

:TRIGger:EINTerval:EVENt<x>:SPIBus:CS?

機能　SPI バストリガのチップセレクトに関するすべての設定値を問い合わせます。

構文　:TRIGger:EINTerval:EVENt<x>:SPIBus:CS?
<x> = 1、2

:TRIGger:EINTerval:EVENt<x>:SPIBus:CS:ACTive

機能　SPI バストリガのチップセレクトのアクティブレベルを設定 / 問い合わせします。

構文　:TRIGger:EINTerval:EVENt<x>:SPIBus:CS:ACTive {HIGH|LOW}
:TRIGger:EINTerval:EVENt<x>:SPIBus:CS:ACTive?
<x> = 1、2

例　:TRIGGER:EINTERVAL:EVENT1:SPIBUS:CS:ACTIVE HIGH
:TRIGGER:EINTERVAL:EVENT1:SPIBUS:CS:ACTIVE? -> :TRIGGER:EINTERVAL:EVENT1:
SPIBUS:CS:ACTIVE HIGH

:TRIGger:EINTerval:EVENt<x>:SPIBus:CS:SOURce

機能　SPI バストリガのチップセレクトトレースを設定 / 問い合わせします。

構文　:TRIGger:EINTerval:EVENt<x>:SPIBus:CS:SOURce {<NRf>}
:TRIGger:EINTerval:EVENt<x>:SPIBus:CS:SOURce?
<x> = 1、2
<NRf> = 1 ～ 4

例　:TRIGGER:EINTERVAL:EVENT1:SPIBUS:CS:SOURCE 1
:TRIGGER:EINTERVAL:EVENT1:SPIBUS:CS:SOURCE? -> :TRIGGER:EINTERVAL:EVENT1:
SPIBUS:CS:SOURCE 1

:TRIGger:EINTerval:EVENt<x>:SPIBus:DATA<x>?

機能　SPI バストリガの各データに関するすべての設定値を問い合わせます。

構文　:TRIGger:EINTerval:EVENt<x>:SPIBus:DATA<x>?
EVENt<x> の <x> = 1、2
DATA<x> の <x> = 1、2

解説　DATA2 は、「:TRIGger:EINTerval:EVENt<x>:SPIBus:MODE WIRE4」のときに有効です。

:TRIGger:EINTerval:EVENt<x>:SPIBus:DATA<x>:BYTE

機能　SPI バストリガの各データの設定データ数を設定 / 問い合わせします。

構文　:TRIGger:EINTerval:EVENt<x>:SPIBus:DATA<x>:BYTE {<NRf>}
:TRIGger:EINTerval:EVENt<x>:SPIBus:DATA<x>:BYTE?
EVENt<x> の <x> = 1、2
DATA<x> の <x> = 1、2
<NRf> = 1 ～ 4

例　:TRIGGER:EINTERVAL:EVENT1:SPIBUS:DATA1:BYTE 1
:TRIGGER:EINTERVAL:EVENT1:SPIBUS:DATA1:BYTE? -> :TRIGGER:EINTERVAL:EVENT1:SPIBUS:DATA1:BYTE 1

:TRIGger:EINTerval:EVENt<x>:SPIBus:DATA<x>:CONDition

機能　SPI バストリガの各データの判定方法 (一致 / 不一致) を設定 / 問い合わせします。

構文　:TRIGger:EINTerval:EVENt<x>:SPIBus:DATA<x>:CONDition {FALSe|TRUE}
:TRIGger:EINTerval:EVENt<x>:SPIBus:DATA<x>:CONDition?
EVENt<x> の <x> = 1、2
DATA<x> の <x> = 1、2

例　:TRIGGER:EINTERVAL:EVENT1:SPIBUS:DATA1:CONDITION TRUE
:TRIGGER:EINTERVAL:EVENT1:SPIBUS:DATA1:CONDITION? -> :TRIGGER:EINTERVAL:EVENT1:SPIBUS:DATA1:CONDITION TRUE

:TRIGger:EINTerval:EVENt<x>:SPIBus:DATA<x>:DPOSition

機能　SPI バストリガの各データのパターン比較先頭位置を設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:SPIBus:
DATA<x>:DPOSition {<NRf>}
:TRIGger:EINTerval:EVENt<x>:SPIBus:
DATA<x>:DPOSition?
```
EVENt<x> の <x> = 1、2
DATA<x> の <x> = 1、2
<NRf> = 0 ～ 9999

例
```
:TRIGGER:EINTERVAL:EVENT1:SPIBUS:
DATA1:DPOSITION 1
:TRIGGER:EINTERVAL:EVENT1:SPIBUS:
DATA1:DPOSITION? -> :TRIGGER:
EINTERVAL:EVENT1:SPIBUS:DATA1:
DPOSITION 1
```

:TRIGger:EINTerval:EVENt<x>:SPIBus:DATA<x>:HEXA<x>

機能　SPI バストリガの各データを HEXA で設定します。

構文
```
:TRIGger:EINTerval:EVENt<x>:SPIBus:
DATA<x>:HEXA<x> {< 文字列 >}
```
EVENt<x> の <x> = 1、2
DATA<x> の <x> = 1、2
HEXA<x> の <x> = 1 ～ 4
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字。

例
```
:TRIGGER:EINTERVAL:EVENT1:SPIBUS:
DATA1:HEXA1 "AB"
```

:TRIGger:EINTerval:EVENt<x>:SPIBus:DATA<x>:PATTern<x>

機能　SPI バストリガの各データを BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:SPIBus:
DATA<x>:PATTern<x> {< 文字列 >}
:TRIGger:EINTerval:EVENt<x>:SPIBus:
DATA<x>:PATTern<x>?
```
EVENt<x> の <x> = 1、2
DATA<x> の <x> = 1、2
PATTern<x> の <x> = 1 ～ 4
< 文字列 > = '0'、'1'、'X' の組み合わせ 8 文字。

例
```
:TRIGGER:EINTERVAL:EVENT1:SPIBUS:
DATA1:PATTERN1 "10101011"
:TRIGGER:EINTERVAL:EVENT1:SPIBUS:
DATA1:PATTERN1? -> :TRIGGER:
EINTERVAL:EVENT1:SPIBUS:DATA1:
PATTERN1 "10101011"
```

:TRIGger:EINTerval:EVENt<x>:SPIBus:DATA<x>:SOURce

機能　SPI バストリガの各データのトレースを設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:SPIBus:
DATA<x>:SOURce {<NRf>}
:TRIGger:EINTerval:EVENt<x>:SPIBus:
DATA<x>:SOURce?
```
EVENt<x> の <x> = 1、2
DATA<x> の <x> = 1、2
<NRf> = 1 ～ 4

例
```
:TRIGGER:EINTERVAL:EVENT1:SPIBUS:
DATA1:SOURCE 1
:TRIGGER:EINTERVAL:EVENT1:SPIBUS:
DATA1:SOURCE? -> :TRIGGER:EINTERVAL:
EVENT1:SPIBUS:DATA1:SOURCE 1
```

:TRIGger:EINTerval:EVENt<x>:SPIBus:MODE

機能　SPI バストリガの結線方式 (3 線式 /4 線式) を設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:SPIBus:
MODE {WIRE3|WIRE4}
:TRIGger:EINTerval:EVENt<x>:SPIBus:
MODE?
```
<x> = 1、2

例
```
:TRIGGER:EINTERVAL:EVENT1:SPIBUS:
MODE WIRE3
:TRIGGER:EINTERVAL:EVENT1:SPIBUS:
MODE? -> :TRIGGER:EINTERVAL:EVENT1:
SPIBUS:MODE WIRE3
```

:TRIGger:EINTerval:EVENt<x>:STATe:CHANnel<x>

機能　各チャネルの成立条件を設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:STATe:
CHANnel<x> {DONTcare|HIGH|LOW}
:TRIGger:EINTerval:EVENt<x>:STATe:
CHANnel<x>?
```
EVENt<x> の <x> = 1、2
CHANnel<x> の <x> = 1 ～ 4

例
```
:TRIGGER:EINTERVAL:EVENT1:STATE:
CHANNEL1 HIGH
:TRIGGER:EINTERVAL:EVENT1:STATE:
CHANNEL1? -> :TRIGGER:EINTERVAL:
EVENT1:STATE:CHANNEL1 HIGH
```

解説　「:TRIGger:EINTerval:EVENt<x>:TYPE I2CBus」のときに有効です。

:TRIGger:EINTerval:EVENt<x>:TYPE

機能　各イベントのトリガの種類を設定 / 問い合わせします。

構文
```
:TRIGger:EINTerval:EVENt<x>:
TYPE {CANBus|EDGE|EQUalify|I2CBus|
LEDGe|LINBus|LI2Cbus|LLINbus|
LSPAttern|LSPIbus|LPSTate|LPULse|
```

LQUalify|LSTate|PQUalify|PSTate|PULSe|SPATtern|SPIBus|STATe}
:TRIGger:EINTerval:EVENt<x>:TYPE?
<x> = 1、2

例 :TRIGGER:EINTERVAL:EVENT1:TYPE CANBUS
:TRIGGER:EINTERVAL:EVENT1:TYPE?
-> :TRIGGER:EINTERVAL:EVENT1:TYPE CANBUS

解説 {LEDGe|LI2Cbus|LLINbus|LSPAttern|LSPIbus|LPSTate|LPULse|LQUalify|LSTate} は、DLM6000 に適用できます。

:TRIGger:ENHanced:CANBus?

機能 CAN バス信号トリガに関するすべての設定値を問い合わせます。

構文 :TRIGger:ENHanced:CANBus?

:TRIGger:ENHanced:CANBus:ACK

機能 CAN バス信号トリガの ACK 条件を設定 / 問い合わせします。

構文 :TRIGger:ENHanced:CANBus:ACK {ACK|ACKBoth|DONTcare|NONack}
:TRIGger:ENHanced:CANBus:ACK?

例 :TRIGGER:ENHANCED:CANBUS:ACK ACK
:TRIGGER:ENHANCED:CANBUS:ACK?
-> :TRIGGER:ENHANCED:CANBUS:ACK ACK

:TRIGger:ENHanced:CANBus:BRATe

機能 CAN バス信号トリガのビットレート (データ転送速度) を設定 / 問い合わせします。

構文 :TRIGger:ENHanced:CANBus:BRATe {<NRf>|USER,<NRf>}
:TRIGger:ENHanced:CANBus:BRATe?
<NRf> = 33300、83300、125000、250000、500000、1000000
USER の <NRf> =本体ユーザーズマニュアル参照。

例 :TRIGGER:ENHANCED:CANBUS:BRATE 83300
:TRIGGER:ENHANCED:CANBUS:BRATE?
-> :TRIGGER:ENHANCED:CANBUS:BRATE 83300

:TRIGger:ENHanced:CANBus:DATA?

機能 CAN バス信号トリガのデータに関するすべての設定値を問い合わせます。

構文 :TRIGger:ENHanced:CANBus:DATA?

:TRIGger:ENHanced:CANBus:DATA:BORDer

機能 CAN バス信号トリガのデータのバイトオーダを設定 / 問い合わせします。

構文 :TRIGger:ENHanced:CANBus:DATA:BORDer {BIG|LITTle}
:TRIGger:ENHanced:CANBus:DATA:BORDer?

例 :TRIGGER:ENHANCED:CANBUS:DATA:BORDER BIG:TRIGGER:ENHANCED:CANBUS:DATA:BORDER? -> :TRIGGER:ENHANCED:CANBUS:DATA:BORDER BIG

:TRIGger:ENHanced:CANBus:DATA:CONDition

機能 CAN バス信号トリガのデータ条件を設定 / 問い合わせします。

構文 :TRIGger:ENHanced:CANBus:DATA:CONDition {BETWeen|DONTcare|FALSe|GTHan|LTHan|ORANge|TRUE}
:TRIGger:ENHanced:CANBus:DATA:CONDition?

例 :TRIGGER:ENHANCED:CANBUS:DATA:CONDITION BETWEEN
:TRIGGER:ENHANCED:CANBUS:DATA:CONDITION? -> :TRIGGER:ENHANCED:CANBUS:DATA:CONDITION BETWEEN

:TRIGger:ENHanced:CANBus:DATA:DATA<x>

機能 CAN バス信号トリガのデータの比較データを設定 / 問い合わせします。

構文 :TRIGger:ENHanced:CANBus:DATA:DATA<x> {<NRf>}
:TRIGger:ENHanced:CANBus:DATA:DATA<x>?
<x> = 1、2
<NRf> =本体ユーザーズマニュアル参照。

例 :TRIGGER:ENHANCED:CANBUS:DATA:DATA11:TRIGGER:ENHANCED:CANBUS:DATA:DATA1? -> :TRIGGER:ENHANCED:CANBUS:DATA:DATA1 1.0000000E+00

解説
- 「:TRIGger:ENHANCED:CANBus:DATA:CONDition GTHan」のときは「:TRIGger:ENHANCED:CANBus:DATA:DATA1」で設定します。
- 「:TRIGger:ENHANCED:CANBus:DATA:CONDition LTHan」のときは「:TRIGger:ENHANCED:CANBus:DATA:DATA2」で設定します。
- 「:TRIGger:ENHANCED:CANBus:DATA:CONDition BETWeen|ORANge」のときは、小さい値を「:TRIGger:ENHANCED:CANBus:DATA:DATA1」、大きい値を「:TRIGger:ENHANCED:CANBus:DATA:DATA2」で設定します。

:TRIGger:ENHanced:CANBus:DATA:DLC

機能　CAN バス信号トリガのデータの有効バイト数 (DLC) を設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:CANBus:DATA:DLC
{<NRf>}
:TRIGger:ENHanced:CANBus:DATA:DLC?
```
<NRf> = 0 ～ 8

例
```
:TRIGGER:ENHANCED:CANBUS:DATA:DLC 0
:TRIGGER:ENHANCED:CANBUS:DATA:DLC?
-> :TRIGGER:ENHANCED:CANBUS:DATA:
DLC 0
```

:TRIGger:ENHanced:CANBus:DATA:HEXA

機能　CAN バス信号トリガのデータを HEXA で設定します。

構文
```
:TRIGger:ENHanced:CANBus:DATA:
HEXA {< 文字列 >}
```
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 16 文字以内 (1 バイト単位)

例
```
:TRIGGER:ENHANCED:CANBUS:DATA:
HEXA "A9"
```

:TRIGger:ENHanced:CANBus:DATA:MSBLsb

機能　CAN バス信号トリガのデータの MSB/LSB のビットを設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:CANBus:DATA:MSBLsb
{<NRf>,<NRf>}
:TRIGger:ENHanced:CANBus:DATA:
MSBLsb?
```
<NRf> =本体ユーザーズマニュアル参照。

例
```
:TRIGGER:ENHANCED:CANBUS:DATA:
MSBLSB 1,0
:TRIGGER:ENHANCED:CANBUS:DATA:
MSBLSB? -> :TRIGGER:ENHANCED:CANBUS:
DATA:MSBLSB 1,0
```

:TRIGger:ENHanced:CANBus:DATA:PATTern

機能　CAN バス信号トリガのデータを BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:CANBus:DATA:
PATTern {< 文字列 >}
:TRIGger:ENHanced:CANBus:DATA:
PATTern?
```
< 文字列 > = '0'、'1'、'X' の組み合わせ 64 文字以内 (1 バイト単位)

例
```
:TRIGGER:ENHANCED:CANBUS:DATA:
PATTERN "11011111"
:TRIGGER:ENHANCED:CANBUS:DATA:
PATTERN? -> :TRIGGER:ENHANCED:
CANBUS:
DATA:PATTERN "11011111"
```

:TRIGger:ENHanced:CANBus:DATA:SIGN

機能　CAN バス信号トリガのデータの符号を設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:CANBus:DATA:
SIGN {SIGN|UNSign}
:TRIGger:ENHanced:CANBus:DATA:SIGN?
```

例
```
:TRIGGER:ENHANCED:CANBUS:DATA:SIGN
SIGN
:TRIGGER:ENHANCED:CANBUS:DATA:
SIGN? -> :TRIGGER:ENHANCED:CANBUS:
DATA:SIGN SIGN
```

:TRIGger:ENHanced:CANBus:IDEXt?

機能　CAN バス信号トリガの拡張フォーマットの ID に関するすべての設定値を問い合わせます。

構文
```
:TRIGger:ENHanced:CANBus:IDEXt?
```

:TRIGger:ENHanced:CANBus:IDEXt:HEXA

機能　CAN バス信号トリガの拡張フォーマットの ID を HEXA で設定します。

構文
```
:TRIGger:ENHanced:CANBus:IDEXt:
HEXA {< 文字列 >}
```
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 8 文字

例
```
:TRIGGER:ENHANCED:CANBUS:IDEXT:
HEXA "1AEF5906"
```

:TRIGger:ENHanced:CANBus:IDEXt:PATTern

機能　CAN バス信号トリガの拡張フォーマットの ID を BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:CANBus:IDEXt:
PATTern {< 文字列 >}
:TRIGger:ENHanced:CANBus:IDEXt:
PATTern?
```
< 文字列 > = '0'、'1'、'X' の組み合わせ 29 文字

例
```
:TRIGGER:ENHANCED:CANBUS:IDEXT:
PATTERN "11001011011100001110111011
11"
:TRIGGER:ENHANCED:CANBUS:IDEXT:
PATTERN? -> :TRIGGER:ENHANCED:
CANBUS:
IDEXT:PATTERN
"11001011011100001110111
1011111"
```

:TRIGger:ENHanced:CANBus:IDOR?

機能　CAN バス信号トリガの OR 条件に関するすべての設定値を問い合わせます。

構文
```
:TRIGger:ENHanced:CANBus:IDOR?
```

:TRIGger:ENHanced:CANBus:IDOR:ID<x>?

機能　CAN バス信号トリガの OR 条件の各 ID に関するすべての設定値を問い合わせます。

構文　:TRIGger:ENHanced:CANBus:IDOR:ID<x>?
<x> = 1 ～ 4

:TRIGger:ENHanced:CANBus:IDOR:ID<x>:ACK

機能　CAN バス信号トリガの OR 条件の各 ACK 条件を設定 / 問い合わせします。

構文　:TRIGger:ENHanced:CANBus:IDOR:ID<x>:ACK {ACK|ACKBoth|DONTcare|NONack}
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:ACK?
<x> = 1 ～ 4

例　:TRIGGER:ENHANCED:CANBUS:IDOR:ID1:ACK ACK
:TRIGGER:ENHANCED:CANBUS:IDOR:ID1:ACK? -> :TRIGGER:ENHANCED:CANBUS:IDOR:ID1:ACK ACK

:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA?

機能　CAN バス信号トリガの OR 条件の各データに関するすべての設定値を問い合わせます。

構文　:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA?
<x> = 1 ～ 4

:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:BORDer

機能　CAN バス信号トリガの OR 条件の各データのバイトオーダを設定 / 問い合わせします。

構文　:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:BORDer {BIG|LITTle}
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:BORDer?
<x> = 1 ～ 4

例　:TRIGGER:ENHANCED:CANBUS:IDOR:ID1:DATA:BORDER BIG
:TRIGGER:ENHANCED:CANBUS:IDOR:ID1:DATA:BORDER? -> :TRIGGER:ENHANCED:CANBUS:IDOR:ID1:DATA:BORDER BIG

:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:CONDition

機能　CAN バス信号トリガの OR 条件の各データ条件を設定 / 問い合わせします。

構文　:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:CONDition {BETWeen|DONTcare|FALSe|GTHan|LTHan|ORANge|TRUE}
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:CONDition?
<x> = 1 ～ 4

例　:TRIGGER:ENHANCED:CANBUS:IDOR:ID1:DATA:CONDITION BETWEEN
:TRIGGER:ENHANCED:CANBUS:IDOR:ID1:DATA:CONDITION? -> :TRIGGER:ENHANCED:CANBUS:IDOR:ID1:DATA:CONDITION BETWEEN

:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:DATA<x>

機能　CAN バス信号トリガの OR 条件の各データの比較データを設定 / 問い合わせします。

構文　:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:DATA<x> {<NRf>}
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:DATA<x>?
ID<x> の <x> = 1 ～ 4
DATA<x> の <x> = 1、2
<NRf> =本体ユーザーズマニュアル参照。

例　:TRIGGER:ENHANCED:CANBUS:IDOR:ID1:DATA:DATA1 1
:TRIGGER:ENHANCED:CANBUS:IDOR:ID1:DATA:DATA1? -> :TRIGGER:ENHANCED:CANBUS:IDOR:ID1:DATA:DATA1 1.0000000E+00

解説
- 「:TRIGger:ENHANCED:CANBus:IDOR:ID<x>:DATA:CONDition GTHan」のときは「:TRIGger:ENHANCED:CANBus:IDOR:ID<x>:DATA:DATA1」で設定します。
- 「:TRIGger:ENHANCED:CANBus:IDOR:ID<x>:DATA:CONDition LTHan」のときは「:TRIGger:ENHANCED:CANBus:IDOR:ID<x>:DATA:DATA2」で設定します。
- 「:TRIGger:ENHANCED:CANBus:IDOR:ID<x>:DATA:CONDition BETWeen|ORANge」のときは、小さい値を「:TRIGger:ENHANCED:CANBus:IDOR:ID<x>:DATA:DATA1」、大きい値を「:TRIGger:ENHANCED:CANBus:IDOR:ID<x>:DATA:DATA2」で設定します。

:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:DLC

機能　CAN バス信号トリガの OR 条件の各データの有効バイト数 (DLC) を設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:DLC {<NRf>}
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:DLC?
```
<x> = 1 ～ 4
<NRf> = 0 ～ 8

例
```
:TRIGGER:ENHANCED:CANBUS:IDOR:ID1:DATA:DLC 0
:TRIGGER:ENHANCED:CANBUS:IDOR:ID1:DATA:DLC? -> :TRIGGER:ENHANCED:CANBUS:IDOR:ID1:DATA:DLC 0
```

:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:HEXA

機能　CAN バス信号トリガの OR 条件の各データを HEXA で設定します。

構文
```
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:HEXA {< 文字列 >}
```
<x> = 1 ～ 4
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 16 文字以内 (1 バイト単位)

例
```
:TRIGGER:ENHANCED:CANBUS:IDOR:ID1:DATA:HEXA "A9"
```

:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:MSBLsb

機能　CAN バス信号トリガの OR 条件の各データの MSB/LSB のビットを設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:MSBLsb {<NRf>,<NRf>}
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:MSBLsb?
```
<x> = 1 ～ 4
<NRf> =本体ユーザーズマニュアル参照。

例
```
:TRIGGER:ENHANCED:CANBUS:IDOR:ID1:DATA:MSBLSB 1,0
:TRIGGER:ENHANCED:CANBUS:IDOR:ID1:DATA:MSBLSB? -> :TRIGGER:ENHANCED:CANBUS:IDOR:ID1:DATA:MSBLSB 1,0
```

:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:PATTern

機能　CAN バス信号トリガの OR 条件の各データを BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:PATTern {< 文字列 >}
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:PATTern?
```
<x> = 1 ～ 4
< 文字列 > = '0'、'1'、'X' の組み合わせ 64 文字以内 (1 バイト単位)

例
```
:TRIGGER:ENHANCED:CANBUS:IDOR:ID1:DATA:PATTERN "11011111"
:TRIGGER:ENHANCED:CANBUS:IDOR:ID1:DATA:PATTERN? -> :TRIGGER:ENHANCED:CANBUS:IDOR:ID1:DATA:PATTERN "11011111"
```

:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:SIGN

機能　CAN バス信号トリガの OR 条件の各データの符号を設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:SIGN {SIGN|UNSign}
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:DATA:SIGN?
```
<x> = 1 ～ 4

例
```
:TRIGGER:ENHANCED:CANBUS:IDOR:ID1:DATA:SIGN SIGN
:TRIGGER:ENHANCED:CANBUS:IDOR:ID1:DATA:SIGN? -> :TRIGGER:ENHANCED:CANBUS:IDOR:ID1:DATA:SIGN SIGN
```

:TRIGger:ENHanced:CANBus:IDOR:ID<x>:FORMat

機能　CAN バス信号トリガの OR 条件の各メッセージフォーマット (標準 / 拡張) を設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:FORMat {STD|EXT}
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:FORMat?
```
<x> = 1 ～ 4

例
```
:TRIGGER:ENHANCED:CANBUS:IDOR:ID1:FORMAT STD
:TRIGGER:ENHANCED:CANBUS:IDOR:ID1:FORMAT? -> :TRIGGER:ENHANCED:CANBUS:IDOR:ID1:FORMAT STD
```

:TRIGger:ENHanced:CANBus:IDOR:ID<x>:IDEXt?

機能　CAN バス信号トリガの OR 条件の各拡張フォーマットの ID に関するすべての設定値を問い合わせます。

構文　:TRIGger:ENHanced:CANBus:IDOR:ID<x>:IDEXt?
<x> = 1 ～ 4

:TRIGger:ENHanced:CANBus:IDOR:ID<x>:IDEXt:HEXA

機能　CAN バス信号トリガの OR 条件の各拡張フォーマットの ID を HEXA で設定します。

構文　:TRIGger:ENHanced:CANBus:IDOR:ID<x>:IDEXt:HEXA {< 文字列 >}
<x> = 1 ～ 4
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 8 文字

例　:TRIGGER:ENHANCED:CANBUS:IDOR:ID1:IDEXT:HEXA "1AEF5906"

:TRIGger:ENHanced:CANBus:IDOR:ID<x>:IDEXt:PATTern

機能　CAN バス信号トリガの OR 条件の各拡張フォーマットの ID を BINARY で設定/問い合わせします。

構文　:TRIGger:ENHanced:CANBus:IDOR:ID<x>:IDEXt:PATTern {< 文字列 >}
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:IDEXt:PATTern?
<x> = 1 ～ 4
< 文字列 > = '0'、'1'、'X' の組み合わせ 29 文字

例　:TRIGGER:ENHANCED:CANBUS:IDOR:ID1:IDEXT:PATTERN "11001011011100001110111011111":TRIGGER:ENHANCED:CANBUS:IDOR:ID1:IDEXT:PATTERN? -> :TRIGGER:ENHANCED:CANBUS:IDOR:ID1:IDEXT:PATTERN "11001011011100001110111011111"

:TRIGger:ENHanced:CANBus:IDOR:ID<x>:IDSTd?

機能　CAN バス信号トリガの OR 条件の各標準フォーマットの ID に関するすべての設定値を問い合わせます。

構文　:TRIGger:ENHanced:CANBus:IDOR:ID<x>:IDSTd?
<x> = 1 ～ 4

:TRIGger:ENHanced:CANBus:IDOR:ID<x>:IDSTd:HEXA

機能　CAN バス信号トリガの OR 条件の各標準フォーマットの ID を HEXA で設定します。

構文　:TRIGger:ENHanced:CANBus:IDOR:ID<x>:IDSTd:HEXA {< 文字列 >}
<x> = 1 ～ 4
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 3 文字

例　:TRIGGER:ENHANCED:CANBUS:IDOR:ID1:IDSTD:HEXA "5DF"

:TRIGger:ENHanced:CANBus:IDOR:ID<x>:IDSTd:PATTern

機能　CAN バス信号トリガの OR 条件の各標準フォーマットの ID を BINARY で設定/問い合わせします。

構文　:TRIGger:ENHanced:CANBus:IDOR:ID<x>:IDSTd:PATTern {< 文字列 >}
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:IDSTd:PATTern?
<x> = 1 ～ 4
< 文字列 > = '0'、'1'、'X' の組み合わせ 11 文字

例　:TRIGGER:ENHANCED:CANBUS:IDOR:ID1:IDSTD:PATTERN "10111011111"
:TRIGGER:ENHANCED:CANBUS:IDOR:ID1:IDSTD:PATTERN? -> :TRIGGER:ENHANCED:CANBUS:IDOR:ID1:IDSTD:PATTERN "10111011111"

:TRIGger:ENHanced:CANBus:IDOR:ID<x>:MODE

機能　CAN バス信号トリガの OR 条件の各条件の有効 (1)/ 無効 (0) を設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:MODE {<Boolean>}
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:MODE?
```
<x> = 1 ～ 4

例
```
:TRIGGER:ENHANCED:CANBUS:IDOR:ID1:MODE ON
:TRIGGER:ENHANCED:CANBUS:IDOR:ID1:MODE? -> :TRIGGER:ENHANCED:CANBUS:IDOR:ID1:MODE 1
```

:TRIGger:ENHanced:CANBus:IDOR:ID<x>:RTR

機能　CAN バス信号トリガの OR 条件の各 RTR を設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:RTR {DATA|DONTcare|REMote}
:TRIGger:ENHanced:CANBus:IDOR:ID<x>:RTR?
```
<x> = 1 ～ 4

例
```
:TRIGGER:ENHANCED:CANBUS:IDOR:ID1:RTR DATA
:TRIGGER:ENHANCED:CANBUS:IDOR:ID1:RTR? -> :TRIGGER:ENHANCED:CANBUS:IDOR:ID1:RTR DATA
```

:TRIGger:ENHanced:CANBus:IDSTd?

機能　CAN バス信号トリガの標準フォーマットの ID に関するすべての設定値を問い合わせます。

構文
```
:TRIGger:ENHanced:CANBus:IDSTd?
```

:TRIGger:ENHanced:CANBus:IDSTd:HEXA

機能　CAN バス信号トリガの標準フォーマットの ID を HEXA で設定します。

構文
```
:TRIGger:ENHanced:CANBus:IDSTd:HEXA {< 文字列 >}
```
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 3 文字

例
```
:TRIGGER:ENHANCED:CANBUS:IDSTD:HEXA "5DF"
```

:TRIGger:ENHanced:CANBus:IDSTd:PATTern

機能　CAN バス信号トリガの標準フォーマットの ID を BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:CANBus:IDSTd:PATTern {< 文字列 >}
:TRIGger:ENHanced:CANBus:IDSTd:PATTern?
```
< 文字列 > = '0'、'1'、'X' の組み合わせ 11 文字

例
```
:TRIGGER:ENHANCED:CANBUS:IDSTD:PATTERN "10111011111"
:TRIGGER:ENHANCED:CANBUS:IDSTD:PATTERN? -> :TRIGGER:ENHANCED:CANBUS:IDSTD:PATTERN "10111011111"
```

:TRIGger:ENHanced:CANBus:MODE

機能　CAN バス信号トリガのモードを設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:CANBus:MODE {EFRame|IDEXt|IDOR|IDSTd|SOF}
:TRIGger:ENHanced:CANBus:MODE?
```

例
```
:TRIGGER:ENHANCED:CANBUS:MODE EFRAME
:TRIGGER:ENHANCED:CANBUS:MODE? -> :TRIGGER:ENHANCED:CANBUS:MODE EFRAME
```

:TRIGger:ENHanced:CANBus:RECessive

機能　CAN バス信号トリガのリセッシブレベル (バスレベル) を設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:CANBus:RECessive {HIGH|LOW}
:TRIGger:ENHanced:CANBus:RECessive?
```

例
```
:TRIGGER:ENHANCED:CANBUS:RECESSIVE HIGH
:TRIGGER:ENHANCED:CANBUS:RECESSIVE? -> :TRIGGER:ENHANCED:CANBUS:RECESSIVE HIGH
```

:TRIGger:ENHanced:CANBus:RTR

機能　CAN バス信号トリガの RTR を設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:CANBus:RTR {DATA|DONTcare|REMote}
:TRIGger:ENHanced:CANBus:RTR?
```

例
```
:TRIGGER:ENHANCED:CANBUS:RTR DATA
:TRIGGER:ENHANCED:CANBUS:RTR? -> :TRIGGER:ENHANCED:CANBUS:RTR DATA
```

:TRIGger:ENHanced:CANBus:SOURce
機能　CAN バス信号トリガのトリガソースを設定 / 問い合わせします。
構文　:TRIGger:ENHanced:CANBus:SOURce {<NRf>}
:TRIGger:ENHanced:CANBus:SOURce?
<NRf> = 1 ～ 4
例　:TRIGGER:ENHANCED:CANBUS:SOURCE 1
:TRIGGER:ENHANCED:CANBUS:SOURCE?
-> :TRIGGER:ENHANCED:CANBUS:SOURCE 1

:TRIGger:ENHanced:CANBus:SPOint
機能　CAN バス信号トリガのサンプルポイントを設定 / 問い合わせします。
構文　:TRIGger:ENHanced:CANBus:SPOint {<NRf>}
:TRIGger:ENHanced:CANBus:SPOint?
<NRf> = 18.8 ～ 90.6(%)
例　:TRIGGER:ENHANCED:CANBUS:SPOINT 18.8
:TRIGGER:ENHANCED:CANBUS:SPOINT? -> :TRIGGER:ENHANCED:CANBUS:SPOINT 18.8E+00

:TRIGger:ENHanced:I2CBus?
機能　I^2C バストリガに関するすべての設定値を問い合わせます。
構文　:TRIGger:ENHanced:I2CBus?

:TRIGger:ENHanced:I2CBus:ADATa?
機能　I^2C バストリガのアドレスに関するすべての設定値を問い合わせます。
構文　:TRIGger:ENHanced:I2CBus:ADATa?

:TRIGger:ENHanced:I2CBus:ADATa:BIT10address?
機能　I^2C バストリガの 10bit アドレスに関するすべての設定値を問い合わせます。
構文　:TRIGger:ENHanced:I2CBus:ADATa:BIT10address?

:TRIGger:ENHanced:I2CBus:ADATa:BIT10address:HEXA
機能　I^2C バストリガの 10bit アドレスを HEXA で設定します。
構文　:TRIGger:ENHanced:I2CBus:ADATa:BIT10address:HEXA {< 文字列 >}
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 3 文字 (ビット 8 は、R/$\overline{W}$ ビット)
例　:TRIGGER:ENHANCED:I2CBUS:ADATA:BIT10ADDRESS:HEXA "7AB"

:TRIGger:ENHanced:I2CBus:ADATa:BIT10address:PATTern
機能　I^2C バストリガの 10bit アドレスを BINARY で設定 / 問い合わせします。
構文　:TRIGger:ENHanced:I2CBus:ADATa:BIT10address:PATTern {< 文字列 >}
:TRIGger:ENHanced:I2CBus:ADATa:BIT10address:PATTern?
< 文字列 > = '0'、'1'、'X' の組み合わせ 11 文字 (ビット 8 は、R/$\overline{W}$ ビット)
例　:TRIGGER:ENHANCED:I2CBUS:ADATA:BIT10ADDRESS:PATTERN "10111011111"
:TRIGGER:ENHANCED:I2CBUS:ADATA:BIT10ADDRESS:PATTERN? -> :TRIGGER:ENHANCED:I2CBUS:ADATA:BIT10ADDRESS:PATTERN "10111011111"

:TRIGger:ENHanced:I2CBus:ADATa:BIT7ADdress?
機能　I^2C バストリガの 7bit アドレスに関するすべての設定値を問い合わせます。
構文　:TRIGger:ENHanced:I2CBus:ADATa:BIT7ADdress?

:TRIGger:ENHanced:I2CBus:ADATa:BIT7ADdress:HEXA
機能　I^2C バストリガの 7bit アドレスを HEXA で設定します。
構文　:TRIGger:ENHanced:I2CBus:ADATa:BIT7ADdress:HEXA {< 文字列 >}
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字 (ビット 0 は、R/$\overline{W}$ ビット)
例　:TRIGGER:ENHANCED:I2CBUS:ADATA:BIT7ADDRESS:HEXA "DE"

:TRIGger:ENHanced:I2CBus:ADATa:BIT7ADdress:PATTern
機能　I^2C バストリガの 7bit アドレスを BINARY で設定 / 問い合わせします。
構文　:TRIGger:ENHanced:I2CBus:ADATa:BIT7ADdress:PATTern {< 文字列 >}
:TRIGger:ENHanced:I2CBus:ADATa:BIT7ADdress:PATTern?
< 文字列 > = '0'、'1'、'X' の組み合わせ 8 文字 (ビット 0 は、R/$\overline{W}$ ビット)
例　:TRIGGER:ENHANCED:I2CBUS:ADATA:BIT7ADDRESS:PATTERN "11011110"
:TRIGGER:ENHANCED:I2CBUS:ADATA:BIT7ADDRESS:PATTERN? -> :TRIGGER:ENHANCED:I2CBUS:ADATA:BIT7ADDRESS:PATTERN "11011110"

:TRIGger:ENHanced:I2CBus:ADATa:BIT7APsub?

機能　I²C バストリガの 7bit + Sub アドレスに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:ENHanced:I2CBus:ADATa:BIT7APsub?
```

:TRIGger:ENHanced:I2CBus:ADATa:BIT7APsub:ADDRess?

機能　I²C バストリガの 7bit + Sub アドレスの 7bit アドレスに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:ENHanced:I2CBus:ADATa:BIT7APsub:ADDRess?
```

:TRIGger:ENHanced:I2CBus:ADATa:BIT7APsub:ADDRess:HEXA

機能　I²C バストリガの 7bit + Sub アドレスの 7bit アドレスを HEXA で設定します。

構文
```
:TRIGger:ENHanced:I2CBus:ADATa:BIT7APsub:ADDRess:HEXA {< 文字列 >}
```
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字 (ビット 0 は、R/$\overline{W}$ ビット)

例
```
:TRIGGER:ENHANCED:I2CBUS:ADATA:BIT7APSUB:ADDRESS:HEXA "AB"
```

:TRIGger:ENHanced:I2CBus:ADATa:BIT7APsub:ADDRess:PATTern

機能　I²C バストリガの 7bit + Sub アドレスの 7bit アドレスを BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:I2CBus:ADATa:BIT7APsub:ADDRess:PATTern {< 文字列 >}
:TRIGger:ENHanced:I2CBus:ADATa:BIT7APsub:ADDRess:PATTern?
```
< 文字列 > = '0'、'1'、'X' の組み合わせ 8 文字 (ビット 0 は、R/$\overline{W}$ ビット)

例
```
:TRIGGER:ENHANCED:I2CBUS:ADATA:BIT7APSUB:ADDRESS:PATTERN "10101011"
:TRIGGER:ENHANCED:I2CBUS:ADATA:BIT7APSUB:ADDRESS:PATTERN?
-> :TRIGGER:ENHANCED:I2CBUS:ADATA:BIT7APSUB:ADDRESS:PATTERN "10101011"
```

:TRIGger:ENHanced:I2CBus:ADATa:BIT7APsub:SADDress?

機能　I²C バストリガの 7bit + Sub アドレスの Sub アドレスに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:ENHanced:I2CBus:ADATa:BIT7APsub:SADDress?
```

:TRIGger:ENHanced:I2CBus:ADATa:BIT7APsub:SADDress:HEXA

機能　I²C バストリガの 7bit + Sub アドレスの Sub アドレスを HEXA で設定します。

構文
```
:TRIGger:ENHanced:I2CBus:ADATa:BIT7APsub:SADDress:HEXA {< 文字列 >}
```
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字

例
```
:TRIGGER:ENHANCED:I2CBUS:ADATA:BIT7APSUB:SADDRESS:HEXA "EF"
```

:TRIGger:ENHanced:I2CBus:ADATa:BIT7APsub:SADDress:PATTern

機能　I²C バストリガの 7bit + Sub アドレスの Sub アドレスを BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:I2CBus:ADATa:BIT7APsub:SADDress:PATTern {< 文字列 >}
:TRIGger:ENHanced:I2CBus:ADATa:BIT7APsub:SADDress:PATTern?
```
< 文字列 > = '0'、'1'、'X' の組み合わせ 8 文字

例
```
:TRIGGER:ENHANCED:I2CBUS:ADATA:BIT7APSUB:SADDRESS:PATTERN "10101011"
:TRIGGER:ENHANCED:I2CBUS:ADATA:BIT7APSUB:SADDRESS:PATTERN?
-> :TRIGGER:ENHANCED:I2CBUS:ADATA:BIT7APSUB:SADDRESS:PATTERN "10101011"
```

:TRIGger:ENHanced:I2CBus:ADATa:TYPE

機能　I²C バストリガのアドレスの種類を設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:I2CBus:ADATa:TYPE {BIT10address|BIT7ADdress|BIT7APsub}
:TRIGger:ENHanced:I2CBus:ADATa:TYPE?
```

例
```
:TRIGGER:ENHANCED:I2CBUS:ADATA:TYPE BIT10ADDRESS
:TRIGGER:ENHANCED:I2CBUS:ADATA:TYPE?
-> :TRIGGER:ENHANCED:I2CBUS:ADATA:TYPE BIT10ADDRESS
```

:TRIGger:ENHanced:I2CBus:CLOCk?

機能　I²C バストリガのクロックに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:ENHanced:I2CBus:CLOCk?
```

:TRIGger:ENHanced:I2CBus:CLOCk:SOURce

機能　I²C バストリガのクロックトレースを設定 / 問い合わせします。

構文　:TRIGger:ENHanced:I2CBus:CLOCk:SOURce {<NRf>}
:TRIGger:ENHanced:I2CBus:CLOCk:SOURce?
<NRf> = 1 ～ 4

例　:TRIGGER:ENHANCED:I2CBUS:CLOCK:SOURCE 1
:TRIGGER:ENHANCED:I2CBUS:CLOCK:SOURCE? -> :TRIGGER:ENHANCED:I2CBUS:CLOCK:SOURCE 1

:TRIGger:ENHanced:I2CBus:DATA?

機能　I²C バストリガのデータに関するすべての設定値を問い合わせます。

構文　:TRIGger:ENHanced:I2CBus:DATA?
<x> = 1、2

:TRIGger:ENHanced:I2CBus:DATA:BYTE

機能　I²C バストリガの設定データ数を設定 / 問い合わせします。

構文　:TRIGger:ENHanced:I2CBus:DATA:BYTE {<NRf>}
:TRIGger:ENHanced:I2CBus:DATA:BYTE?
<NRf> = 1 ～ 4

例　:TRIGGER:ENHANCED:I2CBUS:DATA:BYTE 1
:TRIGGER:ENHANCED:I2CBUS:DATA:BYTE? -> :TRIGGER:ENHANCED:I2CBUS:DATA:BYTE 1

:TRIGger:ENHanced:I2CBus:DATA:CONDition

機能　I²C バストリガのデータの判定方法 (一致 / 不一致) を設定 / 問い合わせします。

構文　:TRIGger:ENHanced:I2CBus:DATA:CONDition {FALSe|TRUE}
:TRIGger:ENHanced:I2CBus:DATA:CONDition?

例　:TRIGGER:ENHANCED:I2CBUS:DATA:CONDITION TRUE
:TRIGGER:ENHANCED:I2CBUS:DATA:CONDITION? -> :TRIGGER:ENHANCED:I2CBUS:DATA:CONDITION TRUE

:TRIGger:ENHanced:I2CBus:DATA:DPOSition

機能　I²C バストリガのデータのパターン比較する位置を設定 / 問い合わせします。

構文　:TRIGger:ENHanced:I2CBus:DATA:DPOSition {<NRf>}
:TRIGger:ENHanced:I2CBus:DATA:DPOSition?
<NRf> = 0 ～ 9999

例　:TRIGGER:ENHANCED:I2CBUS:DATA:DPOSITION 1
:TRIGGER:ENHANCED:I2CBUS:DATA:DPOSITION? -> :TRIGGER:ENHANCED:I2CBUS:DATA:DPOSITION 1

:TRIGger:ENHanced:I2CBus:DATA:HEXA<x>

機能　I²C バストリガのデータを HEXA で設定します。

構文　:TRIGger:ENHanced:I2CBus:DATA:HEXA<x> {< 文字列 >}
<x> = 1 ～ 4
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字

例　:TRIGGER:ENHANCED:I2CBUS:DATA:HEXA1 "AB"

:TRIGger:ENHanced:I2CBus:DATA:MODE

機能　I²C バストリガのデータ条件の有効 / 無効を設定 / 問い合わせします。

構文　:TRIGger:ENHanced:I2CBus:DATA:MODE {<Boolean>}
:TRIGger:ENHanced:I2CBus:DATA:MODE?

例　:TRIGGER:ENHANCED:I2CBUS:DATA:MODE ON
:TRIGGER:ENHANCED:I2CBUS:DATA:MODE? -> :TRIGGER:ENHANCED:I2CBUS:DATA:MODE 1

:TRIGger:ENHanced:I2CBus:DATA:PATTern<x>

機能　I²C バストリガのデータを BINARY で設定 / 問い合わせします。

構文　:TRIGger:ENHanced:I2CBus:DATA:PATTern<x> {< 文字列 >}
:TRIGger:ENHanced:I2CBus:DATA:PATTern<x>?
<x> = 1 ～ 4
< 文字列 > = '0'、'1'、'X' の組み合わせ 8 文字

例　:TRIGGER:ENHANCED:I2CBUS:DATA:PATTERN1 "10101011"
:TRIGGER:ENHANCED:I2CBUS:DATA:PATTERN1? -> :TRIGGER:ENHANCED:I2CBUS:DATA:PATTERN1 "10101011"

:TRIGger:ENHanced:I2CBus:DATA:PMODe

機能　I²C バストリガのデータのパターン比較先頭位置モードを設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:I2CBus:DATA:
PMODe {DONTcare|SELect}
:TRIGger:ENHanced:I2CBus:DATA:PMODe?
```

例
```
:TRIGGER:ENHANCED:I2CBUS:DATA:
PMODE SELECT
:TRIGGER:ENHANCED:I2CBUS:DATA:PMODE?
-> :TRIGGER:ENHANCED:I2CBUS:DATA:
PMODE SELECT
```

:TRIGger:ENHanced:I2CBus:DATA:SOURce

機能　I²C バストリガのデータトレースを設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:I2CBus:DATA:
SOURce {<NRf>}
:TRIGger:ENHanced:I2CBus:DATA:
SOURce?
```
<NRf> = 1 ～ 4

例
```
:TRIGGER:ENHANCED:I2CBUS:DATA:
SOURCE 1
:TRIGGER:ENHANCED:I2CBUS:DATA:
SOURCE?
-> :TRIGGER:ENHANCED:I2CBUS:DATA:
SOURCE 1
```

:TRIGger:ENHanced:I2CBus:GCALl?

機能　I²C バストリガのジェネラルコールに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:ENHanced:I2CBus:GCALl?
```
<x> = 1、2

:TRIGger:ENHanced:I2CBus:GCALl:BIT7maddress?

機能　I²C バストリガのジェネラルコールの 7bit マスタアドレスに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:ENHanced:I2CBus:GCALl:
BIT7maddress?
```
<x> = 1、2

:TRIGger:ENHanced:I2CBus:GCALl:BIT7maddress:HEXA

機能　I²C バストリガのジェネラルコールの 7bit マスタアドレスを HEXA で設定します。

構文
```
:TRIGger:ENHanced:I2CBus:GCALl:
BIT7maddress:HEXA {< 文字列 >}
```
<x> = 1、2
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字 (ビット 0 は、'1' に固定)

例
```
:TRIGGER:ENHANCED:I2CBUS:GCALL:
BIT7MADDRESS:HEXA "AB"
```

:TRIGger:ENHanced:I2CBus:GCALl:BIT7maddress:PATTern

機能　I²C バストリガのジェネラルコールの 7bit マスタアドレスを BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:I2CBus:GCALl:
BIT7maddress:PATTern {< 文字列 >}
:TRIGger:ENHanced:I2CBus:GCALl:
BIT7maddress:PATTern?
```
<x> = 1、2
< 文字列 > = '0'、'1'、'X' の組み合わせ 7 文字

例
```
:TRIGGER:ENHANCED:I2CBUS:GCALL:
BIT7MADDRESS:PATTERN "1010101"
:TRIGGER:ENHANCED:I2CBUS:GCALL:
BIT7MADDRESS:PATTERN? -> :TRIGGER:
ENHANCED:I2CBUS:GCALL:BIT7MADDRESS:
PATTERN "1010101"
```

:TRIGger:ENHanced:I2CBus:GCALl:SBYTe (Second Byte)

機能　I²C バストリガのジェネラルコールのセカンドバイトのタイプを設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:I2CBus:GCALl:
SBYTe {BIT7maddress|DONTcare|H04|H06
}
:TRIGger:ENHanced:I2CBus:GCALl:
SBYTe?
```

例
```
:TRIGGER:ENHANCED:I2CBUS:GCALL:
SBYTE BIT7MADDRESS
:TRIGGER:ENHANCED:I2CBUS:GCALL:
SBYTE?
-> :TRIGGER:ENHANCED:I2CBUS:GCALL:
SBYTE BIT7MADDRESS
```

:TRIGger:ENHanced:I2CBus:MODE

機能　I²C バストリガのトリガモードを設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:I2CBus:MODE
{ADATa|
ESTart|GCALl|NAIGnore|SBHSmode}
:TRIGger:ENHanced:I2CBus:MODE?
```

例
```
:TRIGGER:ENHANCED:I2CBUS:MODE ADATA
:TRIGGER:ENHANCED:I2CBUS:MODE? -> :
TRIGGER:ENHANCED:I2CBUS:MODE ADATA
```

:TRIGger:ENHanced:I2CBus:NAIGnore?

機能　I²C バストリガの NON ACK 無視モードに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:ENHanced:I2CBus:NAIGnore?
```

:TRIGger:ENHanced:I2CBus:NAIGnore:HSMode

機能　I²C バストリガのハイスピードモードで NON ACK を無視する / しないを設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:I2CBus:NAIGnore:HSMode {<Boolean>}
:TRIGger:ENHanced:I2CBus:NAIGnore:HSMode?
```

例
```
:TRIGGER:ENHANCED:I2CBUS:NAIGNORE:HSMODE ON
:TRIGGER:ENHANCED:I2CBUS:NAIGNORE:HSMODE? -> :TRIGGER:ENHANCED:I2CBUS:NAIGNORE:HSMODE 1
```

:TRIGger:ENHanced:I2CBus:NAIGnore:RACCess

機能　I²C バストリガのリードアクセスモードで NON ACK を無視する / しないを設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:I2CBus:NAIGnore:RACCess {<Boolean>}
:TRIGger:ENHanced:I2CBus:NAIGnore:RACCess?
```

例
```
:TRIGGER:ENHANCED:I2CBUS:NAIGNORE:RACCESS ON
:TRIGGER:ENHANCED:I2CBUS:NAIGNORE:RACCESS? -> :TRIGGER:ENHANCED:I2CBUS:
NAIGNORE:RACCESS 1
```

:TRIGger:ENHanced:I2CBus:NAIGnore:SBYTe (Start Byte)

機能　I²C バストリガのスタートバイトで NON ACK を無視する / しないを設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:I2CBus:NAIGnore:SBYTe {<Boolean>}
:TRIGger:ENHanced:I2CBus:NAIGnore:SBYTe?
```

例
```
:TRIGGER:ENHANCED:I2CBUS:NAIGNORE:SBYTE ON
:TRIGGER:ENHANCED:I2CBUS:NAIGNORE:SBYTE? -> :TRIGGER:ENHANCED:I2CBUS:NAIGNORE:SBYTE 1
```

:TRIGger:ENHanced:I2CBus:SBHSmode?

機能　I²C バストリガのスタートバイト / ハイスピードモードに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:ENHanced:I2CBus:SBHSmode?
```

:TRIGger:ENHanced:I2CBus:SBHSmode:TYPE

機能　I²C バストリガのスタートバイト / ハイスピードモードのタイプを設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:I2CBus:SBHSmode:TYPE {HSMode|SBYTe}
:TRIGger:ENHanced:I2CBus:SBHSmode:TYPE?
```

例
```
:TRIGGER:ENHANCED:I2CBUS:SBHSMODE:TYPE HSMODE
:TRIGGER:ENHANCED:I2CBUS:SBHSMODE:TYPE? -> :TRIGGER:ENHANCED:I2CBUS:SBHSMODE:TYPE HSMODE
```

:TRIGger:ENHanced:LINBus?

機能　LIN バストリガに関するすべての設定値を問い合わせします。

構文
```
:TRIGger:ENHanced:LINBus?
```

:TRIGger:ENHanced:LINBus:BRATe

機能　LIN バス信号トリガのビットレート (データ転送速度) を設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:LINBus:BRATe {<NRf>|USER,<NRf>}
:TRIGger:ENHanced:LINBus:BRATe?
```
<NRf> ＝ 1200、2400、4800、9600、19200
USER の <NRf> ＝本体ユーザーズマニュアル参照。

例
```
:TRIGGER:ENHANCED:LINBUS:BRATE 19200
:TRIGGER:ENHANCED:LINBUS:BRATE? -> :TRIGGER:ENHANCED:LINBUS:BRATE 19200
```

:TRIGger:ENHanced:LINBus:SOURce

機能　LIN バス信号トリガのトリガソースを設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:LINBus:SOURce {<NRf>}
:TRIGger:ENHanced:LINBus:SOURce?
```
<NRf> ＝ 1 ～ 4

例
```
:TRIGGER:ENHANCED:LINBUS:SOURCE 1
:TRIGGER:ENHANCED:LINBUS:SOURCE? -> :TRIGGER:ENHANCED:LINBUS:SOURCE 1
```

:TRIGger:ENHanced:SPIBus?

機能　SPI バストリガに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:ENHanced:SPIBus?
```

:TRIGger:ENHanced:SPIBus:BITorder

機能　SPI バストリガのビットオーダーを設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:SPIBus:
BITorder {LSBFirst|MSBFirst}
:TRIGger:ENHanced:SPIBus:BITorder?
```

例
```
:TRIGGER:ENHANCED:SPIBUS:
BITORDER LSBFIRST
:TRIGGER:ENHANCED:SPIBUS:BITORDER?
-> :TRIGGER:ENHANCED:SPIBUS:
BITORDER LSBFIRST
```

:TRIGger:ENHanced:SPIBus:CLOCk?

機能　SPI バストリガのクロックに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:ENHanced:SPIBus:CLOCk?
```

:TRIGger:ENHanced:SPIBus:CLOCk:POLarity

機能　SPI バストリガのクロックトレースの極性を設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:SPIBus:CLOCk:
POLarity {FALL|RISE}
:TRIGger:ENHanced:SPIBus:CLOCk:
POLarity?
```

例
```
:TRIGGER:ENHANCED:SPIBUS:CLOCK:
POLARITY FALL
:TRIGGER:ENHANCED:SPIBUS:CLOCK:
POLARITY? -> :TRIGGER:ENHANCED:
SPIBUS:CLOCK:POLARITY FALL
```

:TRIGger:ENHanced:SPIBus:CLOCk:SOURce

機能　SPI バストリガのクロックトレースを設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:SPIBus:CLOCk:
SOURce {<NRf>}
:TRIGger:ENHanced:SPIBus:CLOCk:
SOURce?
```
<NRf> = 1 ～ 4

例
```
:TRIGGER:ENHANCED:SPIBUS:CLOCK:
SOURCE 1
:TRIGGER:ENHANCED:SPIBUS:CLOCK:
SOURCE? -> :TRIGGER:ENHANCED:SPIBUS:
CLOCK:SOURCE 1
```

:TRIGger:ENHanced:SPIBus:CS?

機能　SPI バストリガのチップセレクトに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:ENHanced:SPIBus:CS?
```

:TRIGger:ENHanced:SPIBus:CS:ACTive

機能　SPI バストリガのチップセレクトのアクティブレベルを設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:SPIBus:CS:
ACTive {HIGH|LOW}
:TRIGger:ENHanced:SPIBus:CS:ACTive?
```

例
```
:TRIGGER:ENHANCED:SPIBUS:CS:ACTIVE
HIGH
:TRIGGER:ENHANCED:SPIBUS:CS:ACTIVE?
-> :TRIGGER:ENHANCED:SPIBUS:CS:
ACTIVE HIGH
```

:TRIGger:ENHanced:SPIBus:CS:SOURce

機能　SPI バストリガのチップセレクトトレースを設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:SPIBus:CS:
SOURce {<NRf>}
:TRIGger:ENHanced:SPIBus:CS:SOURce?
```
<NRf> = 1 ～ 4

例
```
:TRIGGER:ENHANCED:SPIBUS:CS:SOURCE 1
:TRIGGER:ENHANCED:SPIBUS:CS:SOURCE?
-> :TRIGGER:ENHANCED:SPIBUS:CS:
SOURCE 1
```

:TRIGger:ENHanced:SPIBus:DATA<x>?

機能　SPI バストリガの各データに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:ENHanced:SPIBus:DATA<x>?
```
<x> = 1、2

解説　DATA2 は、「`:TRIGger:ENHanced:SPIBus:MODE WIRE4`」のときに有効です。

:TRIGger:ENHanced:SPIBus:DATA<x>:BYTE

機能　SPI バストリガの各データの設定データ数を設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:SPIBus:DATA<x>:
BYTE {<NRf>}
:TRIGger:ENHanced:SPIBus:DATA<x>:
BYTE?
```
<x> = 1、2
<NRf> = 1 ～ 4

例
```
:TRIGGER:ENHANCED:SPIBUS:DATA1:BYTE
1
:TRIGGER:ENHANCED:SPIBUS:DATA1:BYTE?
-> :TRIGGER:ENHANCED:SPIBUS:DATA1:
BYTE 1
```

:TRIGger:ENHanced:SPIBus:DATA<x>:CONDition

機能　SPI バストリガの各データの判定方法 (一致 / 不一致) を設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:SPIBus:DATA<x>:
CONDition {FALSe|TRUE}
:TRIGger:ENHanced:SPIBus:DATA<x>:
CONDition?
```
<x> = 1、2

例
```
:TRIGGER:ENHANCED:SPIBUS:DATA1:
CONDITION TRUE
:TRIGGER:ENHANCED:SPIBUS:DATA1:
CONDITION? -> :TRIGGER:ENHANCED:
SPIBUS:DATA1:CONDITION TRUE
```

:TRIGger:ENHanced:SPIBus:DATA<x>:DPOSition

機能　SPI バストリガの各データのパターン比較先頭位置を設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:SPIBus:DATA<x>:
DPOSition {<NRf>}
:TRIGger:ENHanced:SPIBus:DATA<x>:
DPOSition?
```
<x> = 1、2
<NRf> = 0 ～ 9999

例
```
:TRIGGER:ENHANCED:SPIBUS:DATA1:
DPOSITION 1
:TRIGGER:ENHANCED:SPIBUS:DATA1:
DPOSITION? -> :TRIGGER:ENHANCED:
SPIBUS:DATA1:DPOSITION 1
```

:TRIGger:ENHanced:SPIBus:DATA<x>:HEXA<x>

機能　SPI バストリガの各データを HEXA で設定します。

構文
```
:TRIGger:ENHanced:SPIBus:DATA<x>:
HEXA<x> {< 文字列 >}
```
DATA<x> の <x> = 1、2
HEXA<x> の <x> = 1 ～ 4
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字。

例
```
:TRIGGER:ENHANCED:SPIBUS:DATA1:
HEXA1 "AB"
```

:TRIGger:ENHanced:SPIBus:DATA<x>:PATTern<x>

機能　SPI バストリガの各データを BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:SPIBus:DATA<x>:
PATTern<x> {< 文字列 >}
:TRIGger:ENHanced:SPIBus:DATA<x>:
PATTern<x>?
```
DATA<x> の <x> = 1、2
PATTern<x> の <x> = 1 ～ 4
< 文字列 > = '0'、'1'、'X' の組み合わせ 8 文字

例
```
:TRIGGER:ENHANCED:SPIBUS:DATA1:
PATTERN1 "10101011"
:TRIGGER:ENHANCED:SPIBUS:DATA1:
PATTERN1? -> :TRIGGER:ENHANCED:
SPIBUS:DATA1:PATTERN1 "10101011"
```

:TRIGger:ENHanced:SPIBus:DATA<x>:SOURce

機能　SPI バストリガの各データのトレースを設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:SPIBus:DATA<x>:
SOURce {<NRf>}
:TRIGger:ENHanced:SPIBus:DATA<x>:
SOURce?
```
<x> = 1、2
<NRf> = 1 ～ 4

例
```
:TRIGGER:ENHANCED:SPIBUS:DATA1:
SOURCE 1
:TRIGGER:ENHANCED:SPIBUS:DATA1:
SOURCE? -> :TRIGGER:ENHANCED:SPIBUS:
DATA1:SOURCE 1
```

:TRIGger:ENHanced:SPIBus::MODE

機能　SPI バストリガの結線方式 (3 線式 /4 線式) を設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:SPIBus:MODE
{WIRE3|
WIRE4}
:TRIGger:ENHanced:SPIBus:MODE?
```

例
```
:TRIGGER:ENHANCED:SPIBUS:MODE WIRE3
:TRIGGER:ENHANCED:SPIBUS:MODE?
-> :TRIGGER:ENHANCED:SPIBUS:
MODE WIRE3
```

:TRIGger:ENHanced:UART?

機能　UART 信号トリガに関するすべて設定値を問い合わせします。

構文
```
:TRIGger:ENHanced:UART?
```

:TRIGger:ENHanced:UART:BRATe

機能　UART 信号トリガのビットレート (データ転送速度) を設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:UART:
BRATe {<NRf>|USER,<NRf>}
:TRIGger:ENHanced:UART:BRATe?
```
<NRf> = 1200、2400、4800、9600、19200、38400、57600、115200
USER の <NRf> =本体ユーザーズマニュアル参照。

例
```
:TRIGGER:ENHANCED:UART:BRATE 19200
:TRIGGER:ENHANCED:UART:BRATE?
-> :TRIGGER:ENHANCED:UART:BRATE
19200
```

:TRIGger:ENHanced:UART:FORMat

機能　UART 信号トリガのフォーマットを設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:UART:
FORMat {BIT7parity|BIT8Noparity|
BIT8Parity}
:TRIGger:ENHanced:UART:FORMat?
```

例
```
:TRIGGER:ENHANCED:UART:
FORMAT BIT7PARITY
:TRIGGER:ENHANCED:UART:FORMAT?
-> :TRIGGER:ENHANCED:UART:F
ORMAT BIT7PARITY
```

:TRIGger:ENHanced:UART:POLarity

機能　UART 信号トリガの極性を設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:UART:
POLarity {NEGative|POSitive}
:TRIGger:ENHanced:UART:POLarity?
```

例
```
:TRIGGER:ENHANCED:UART:
POLARITY NEGATIVE
:TRIGGER:ENHANCED:UART:POLARITY?
-> :TRIGGER:ENHANCED:UART:
POLARITY NEGATIVE
```

:TRIGger:ENHanced:UART:SOURce

機能　UART 信号トリガのトリガソースを設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:UART:SOURce
{<NRf>}
:TRIGger:ENHanced:UART:SOURce?
```
<NRf> = 1 ～ 4

例
```
:TRIGGER:ENHANCED:UART:SOURCE 1
:TRIGGER:ENHANCED:UART:SOURCE?
-> :TRIGGER:ENHANCED:UART:SOURCE 1
```

:TRIGger:ENHanced:UART:SPOint

機能　UART 信号トリガのサンプルポイントを設定 / 問い合わせします。

構文
```
:TRIGger:ENHanced:UART:SPOint
{<NRf>}
:TRIGger:ENHanced:UART:SPOint?
```
<NRf> = 18.8 ～ 90.6(%)

例
```
:TRIGGER:ENHANCED:UART:SPOINT 18.8
:TRIGGER:ENHANCED:UART:SPOINT?
-> :TRIGGER:ENHANCED:UART:
SPOINT 18.8E+00
```

:TRIGger:LOGic:I2CBus?

機能　ロジック I^2C バストリガに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:LOGic:I2CBus?
```

:TRIGger:LOGic:I2CBus:ADATa?

機能　ロジック I^2C バストリガのアドレスに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:LOGic:I2CBus:ADATa?
```

:TRIGger:LOGic:I2CBus:ADATa:BIT10address?

機能　ロジック I^2C バストリガの 10bit アドレスに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:LOGic:I2CBus:ADATa:
BIT10address?
```

:TRIGger:LOGic:I2CBus:ADATa:BIT10address:HEXA

機能　ロジック I^2C バストリガの 10bit アドレスを HEXA で設定します。

構文
```
:TRIGger:LOGic:I2CBus:ADATa:
BIT10address:HEXA {< 文字列 >}
```
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 3 文字 (ビット 8 は、R/$\overline{W}$ ビット)

例
```
:TRIGGER:LOGIC:I2CBUS:ADATA:
BIT10ADDRESS:HEXA "7AB"
```

:TRIGger:LOGic:I2CBus:ADATa:BIT10address:PATTern

機能　ロジック I^2C バストリガの 10bit アドレスを BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:I2CBus:ADATa:
BIT10address:PATTern {< 文字列 >}
:TRIGger:LOGic:I2CBus:ADATa:
BIT10address:PATTern?
```
< 文字列 > = '0'、'1'、'X' の組み合わせ 11 文字 (ビット 8 は、R/$\overline{W}$ ビット)

例
```
:TRIGGER:LOGIC:I2CBUS:ADATA:
BIT10ADDRESS:PATTERN "10111011111"
:TRIGGER:LOGIC:I2CBUS:ADATA:
BIT10ADDRESS:PATTERN? -> :TRIGGER:
LOGIC:I2CBUS:ADATA:BIT10ADDRESS:
PATTERN "10111011111"
```

:TRIGger:LOGic:I2CBus:ADATa:BIT7ADdress?

機能　ロジック I²C バストリガの 7bit アドレスに関するすべての設定値を問い合わせます。

構文　:TRIGger:LOGic:I2CBus:ADATa:BIT7ADdress?

:TRIGger:LOGic:I2CBus:ADATa:BIT7ADdress:HEXA

機能　ロジック I²C バストリガの 7bit アドレスを HEXA で設定します。

構文　:TRIGger:LOGic:I2CBus:ADATa:BIT7ADdress:HEXA {< 文字列 >}

< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字 (ビット 0 は、R/$\overline{W}$ ビット)

例　:TRIGGER:LOGIC:I2CBUS:ADATA:BIT7ADDRESS:HEXA "DE"

:TRIGger:LOGic:I2CBus:ADATa:BIT7ADdress:PATTern

機能　ロジック I²C バストリガの 7bit アドレスを BINARY で設定 / 問い合わせします。

構文　:TRIGger:LOGic:I2CBus:ADATa:BIT7ADdress:PATTern {< 文字列 >}

:TRIGger:LOGic:I2CBus:ADATa:BIT7ADdress:PATTern?

< 文字列 > = '0'、'1'、'X' の組み合わせ 8 文字 (ビット 0 は、R/$\overline{W}$ ビット)

例　:TRIGGER:LOGIC:I2CBUS:ADATA:BIT7ADDRESS:PATTERN "11011110"

:TRIGGER:LOGIC:I2CBUS:ADATA:BIT7ADDRESS:PATTERN? -> :TRIGGER:LOGIC:I2CBUS:ADATA:BIT7ADDRESS:PATTERN "11011110"

:TRIGger:LOGic:I2CBus:ADATa:BIT7APsub?

機能　ロジック I²C バストリガの 7bit + Sub アドレスに関するすべての設定値を問い合わせます。

構文　:TRIGger:LOGic:I2CBus:ADATa:BIT7APsub?

:TRIGger:LOGic:I2CBus:ADATa:BIT7APsub:ADDRess?

機能　ロジック I²C バストリガの 7bit + Sub アドレスの 7bit アドレスに関するすべての設定値を問い合わせます。

構文　:TRIGger:LOGic:I2CBus:ADATa:BIT7APsub:ADDRess?

:TRIGger:LOGic:I2CBus:ADATa:BIT7APsub:ADDRess:HEXA

機能　ロジック I²C バストリガの 7bit + Sub アドレスの 7bit アドレスを HEXA で設定します。

構文　:TRIGger:LOGic:I2CBus:ADATa:BIT7APsub:ADDRess:HEXA {< 文字列 >}

< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字 (ビット 0 は、R/$\overline{W}$ ビット)

例　:TRIGGER:LOGIC:I2CBUS:ADATA:BIT7APSUB:ADDRESS:HEXA "AB"

:TRIGger:LOGic:I2CBus:ADATa:BIT7APsub:ADDRess:PATTern

機能　ロジック I²C バストリガの 7bit + Sub アドレスの 7bit アドレスを BINARY で設定 / 問い合わせします。

構文　:TRIGger:LOGic:I2CBus:ADATa:BIT7APsub:ADDRess:PATTern {< 文字列 >}

:TRIGger:LOGic:I2CBus:ADATa:BIT7APsub:ADDRess:PATTern?

< 文字列 > = '0'、'1'、'X' の組み合わせ 8 文字 (ビット 0 は、R/$\overline{W}$ ビット)

例　:TRIGGER:LOGIC:I2CBUS:ADATA:BIT7APSUB:ADDRESS:PATTERN "10101011"

:TRIGGER:LOGIC:I2CBUS:ADATA:BIT7APSUB:ADDRESS:PATTERN?

-> :TRIGGER:LOGIC:I2CBUS:ADATA:BIT7APSUB:ADDRESS:PATTERN "10101011"

:TRIGger:LOGic:I2CBus:ADATa:BIT7APsub:SADDress?

機能　ロジック I²C バストリガの 7bit + Sub アドレスの Sub アドレスに関するすべての設定値を問い合わせます。

構文　:TRIGger:LOGic:I2CBus:ADATa:BIT7APsub:SADDress?

:TRIGger:LOGic:I2CBus:ADATa:BIT7APsub:SADDress:HEXA

機能　ロジック I²C バストリガの 7bit + Sub アドレスの Sub アドレスを HEXA で設定します。

構文　:TRIGger:LOGic:I2CBus:ADATa:BIT7APsub:SADDress:HEXA {< 文字列 >}

< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字

例　:TRIGGER:LOGIC:I2CBUS:ADATA:BIT7APSUB:SADDRESS:HEXA "EF"

:TRIGger:LOGic:I2CBus:ADATa:BIT7APsub:SADDress:PATTern

機能　ロジック I²C バストリガの 7bit + Sub アドレスの Sub アドレスを BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:I2CBus:ADATa:
BIT7APsub:SADDress:PATTern {<文字列>}
:TRIGger:LOGic:I2CBus:ADATa:
BIT7APsub:SADDress:PATTern?
```
<文字列> = '0'、'1'、'X' の組み合わせ 8 文字

例
```
:TRIGGER:LOGIC:I2CBUS:ADATA:
BIT7APSUB:SADDRESS:PATTERN
"10101011"
:TRIGGER:LOGIC:I2CBUS:ADATA:
BIT7APSUB:SADDRESS:PATTERN?
-> :TRIGGER:LOGIC:I2CBUS:ADATA:
BIT7APSUB:SADDRESS:PATTERN
"10101011"
```

:TRIGger:LOGic:I2CBus:ADATa:TYPE

機能　ロジック I²C バストリガのアドレスの種類を設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:I2CBus:ADATa:TYPE
{BIT10address|BIT7ADdress|BIT7APsub}
:TRIGger:LOGic:I2CBus:ADATa:TYPE?
```

例
```
:TRIGGER:LOGIC:I2CBUS:ADATA:
TYPE BIT10ADDRESS
:TRIGGER:LOGIC:I2CBUS:ADATA:TYPE?
-> :TRIGGER:LOGIC:I2CBUS:ADATA:
TYPE BIT10ADDRESS
```

:TRIGger:LOGic:I2CBus:CLOCk?

機能　ロジック I2C バストリガのクロックに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:LOGic:I2CBus:CLOCk?
```

:TRIGger:LOGic:I2CBus:CLOCk:SOURce

機能　ロジック I²C バストリガのクロックトレースを設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:I2CBus:CLOCk:SOURce
{A<x>|B<x>|C<x>|D<x>}
:TRIGger:LOGic:I2CBus:CLOCk:SOURce?
```
<x> = 0 ～ 7

例
```
:TRIGGER:LOGIC:I2CBUS:CLOCK:SOURCE
A0
:TRIGGER:LOGIC:I2CBUS:CLOCK:SOURCE?
-> :TRIGGER:LOGIC:I2CBUS:CLOCK:
SOURCE A0
```

解説　DLM6000 の 16 ビットモデルでは {A<x>|C<x>} が有効です。

:TRIGger:LOGic:I2CBus:DATA?

機能　ロジック I²C バストリガのデータに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:LOGic:I2CBus:DATA?
```

:TRIGger:LOGic:I2CBus:DATA:BYTE

機能　ロジック I²C バストリガの設定データ数を設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:I2CBus:DATA:BYTE
{<NRf>}
:TRIGger:LOGic:I2CBus:DATA:BYTE?
```
<NRf> = 1 ～ 4

例
```
:TRIGGER:LOGIC:I2CBUS:DATA:BYTE 1
:TRIGGER:LOGIC:I2CBUS:DATA:BYTE?
-> :TRIGGER:LOGIC:I2CBUS:DATA:BYTE 1
```

:TRIGger:LOGic:I2CBus:DATA:CONDition

機能　ロジック I²C バストリガのデータの判定方法 (一致 / 不一致) を設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:I2CBus:DATA:CONDition
{FALSe|TRUE}
:TRIGger:LOGic:I2CBus:DATA:
CONDition?
```

例
```
:TRIGGER:LOGIC:I2CBUS:DATA:
CONDITION FALSE
:TRIGGER:LOGIC:I2CBUS:DATA:
CONDITION?
-> :TRIGGER:LOGIC:I2CBUS:DATA:
CONDITION FALSE
```

:TRIGger:LOGic:I2CBus:DATA:DPOSition

機能　ロジック I²C バストリガのデータのパターン比較する位置を設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:I2CBus:DATA:DPOSition
{<NRf>}
:TRIGger:LOGic:I2CBus:DATA:
DPOSition?
```
<NRf> = 0 ～ 9999

例
```
:TRIGGER:LOGIC:I2CBUS:DATA:
DPOSITION 1
:TRIGGER:LOGIC:I2CBUS:DATA:
DPOSITION?
-> :TRIGGER:LOGIC:I2CBUS:DATA:
DPOSITION 1
```

:TRIGger:LOGic:I2CBus:DATA:HEXA<x>

機能　ロジック I²C バストリガのデータを HEXA で設定します。

構文
```
:TRIGger:LOGic:I2CBus:DATA:HEXA<x>
{<文字列>}
```
<x> = 1 ～ 4
<文字列> = '0' ～ 'F'、'X' の組み合わせ 2 文字

例
```
:TRIGGER:LOGIC:I2CBUS:DATA:HEXA1
"AB"
```

:TRIGger:LOGic:I2CBus:DATA:MODE

機能　ロジック I²C バストリガのデータ条件の有効 / 無効を設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:I2CBus:DATA:MODE {<Boolean>}
:TRIGger:LOGic:I2CBus:DATA:MODE?
```

例
```
:TRIGGER:LOGIC:I2CBUS:DATA:MODE ON
:TRIGGER:LOGIC:I2CBUS:DATA:MODE?
-> :TRIGGER:LOGIC:I2CBUS:DATA:MODE 1
```

:TRIGger:LOGic:I2CBus:DATA:PATTern<x>

機能　ロジック I²C バストリガのデータを BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:I2CBus:DATA:PATTern<x> {< 文字列 >}
:TRIGger:LOGic:I2CBus:DATA:PATTern<x>?
```
<x> = 1 ～ 4
< 文字列 > = '0'、'1'、'X' の組み合わせ 8 文字

例
```
:TRIGGER:LOGIC:I2CBUS:DATA:PATTERN1 "10101011"
:TRIGGER:LOGIC:I2CBUS:DATA:PATTERN1?
-> :TRIGGER:LOGIC:I2CBUS:DATA:PATTERN1 "10101011"
```

:TRIGger:LOGic:I2CBus:DATA:PMODe

機能　ロジック I²C バストリガのデータのパターン比較先頭位置モードを設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:I2CBus:DATA:PMODe {DONTcare|SELect}
:TRIGger:LOGic:I2CBus:DATA:PMODe?
```

例
```
:TRIGGER:LOGIC:I2CBUS:DATA:PMODE DONTCARE
:TRIGGER:LOGIC:I2CBUS:DATA:PMODE?
-> :TRIGGER:LOGIC:I2CBUS:DATA:PMODE DONTCARE
```

:TRIGger:LOGic:I2CBus:DATA:SOURce

機能　ロジック I²C バストリガのデータトレースを設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:I2CBus:DATA:SOURce {A<x>|B<x>|C<x>|D<x>}
:TRIGger:LOGic:I2CBus:DATA:SOURce?
```
<x> = 0 ～ 7

例
```
:TRIGGER:LOGIC:I2CBUS:DATA:SOURCE A0
:TRIGGER:LOGIC:I2CBUS:DATA:SOURCE?
-> :TRIGGER:LOGIC:I2CBUS:DATA:SOURCE A0
```

解説　DLM6000 の 16 ビットモデルでは {A<x>|C<x>} が有効です。

:TRIGger:LOGic:I2CBus:GCALl?

機能　ロジック I²C バストリガのジェネラルコールに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:LOGic:I2CBus:GCALl?
```

:TRIGger:LOGic:I2CBus:GCALl:BIT7maddress?

機能　ロジック I²C バストリガのジェネラルコールの 7bit マスタアドレスに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:LOGic:I2CBus:GCALl:BIT7maddress?
```

:TRIGger:LOGic:I2CBus:GCALl:BIT7maddress:HEXA

機能　ロジック I²C バストリガのジェネラルコールの 7bit マスタアドレスを HEXA で設定します。

構文
```
:TRIGger:LOGic:I2CBus:GCALl:BIT7maddress:HEXA {< 文字列 >}
```
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字 (ビット 0 は、'1' に固定)

例
```
:TRIGGER:LOGIC:I2CBUS:GCALL:BIT7MADDRESS:HEXA "AB"
```

:TRIGger:LOGic:I2CBus:GCALl:BIT7maddress:PATTern

機能　ロジック I²C バストリガのジェネラルコールの 7bit マスタアドレスを BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:I2CBus:GCALl:BIT7maddress:PATTern {< 文字列 >}
:TRIGger:LOGic:I2CBus:GCALl:BIT7maddress:PATTern?
```
< 文字列 > = '0'、'1'、'X' の組み合わせ 7 文字

例
```
:TRIGGER:LOGIC:I2CBUS:GCALL:BIT7MADDRESS:PATTERN "1010101"
:TRIGGER:LOGIC:I2CBUS:GCALL:BIT7MADDRESS:PATTERN? -> :TRIGGER:LOGIC:I2CBUS:GCALL:BIT7MADDRESS:PATTERN "1010101"
```

:TRIGger:LOGic:I2CBus:GCALl:SBYTe (Second Byte)

機能　ロジック I²C バストリガのジェネラルコールのセカンドバイトのタイプを設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:I2CBus:GCALl:SBYTe {BIT7maddress|DONTcare|H04|H06}
:TRIGger:LOGic:I2CBus:GCALl:SBYTe?
```

例
```
:TRIGGER:LOGIC:I2CBUS:GCALL:SBYTE BIT7MADDRESS
:TRIGGER:LOGIC:I2CBUS:GCALL:SBYTE?
-> :TRIGGER:LOGIC:I2CBUS:GCALL:SBYTE BIT7MADDRESS
```

:TRIGger:LOGic:I2CBus:MODE

機能　ロジック I²C バストリガのトリガモードを設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:I2CBus:MODE {ADATa|ESTart|GCALl|NAIGnore|SBHSmode}
:TRIGger:LOGic:I2CBus:MODE?
```

例
```
:TRIGGER:LOGIC:I2CBUS:MODE ADATA
:TRIGGER:LOGIC:I2CBUS:MODE?
-> :TRIGGER:LOGIC:I2CBUS:MODE ADATA
```

:TRIGger:LOGic:I2CBus:NAIGnore?

機能　ロジック I²C バストリガの NON ACK 無視モードに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:LOGic:I2CBus:NAIGnore?
```

:TRIGger:LOGic:I2CBus:NAIGnore:HSMode

機能　ロジック I²C バストリガのハイスピードモードで NON ACK を無視する / しないを設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:I2CBus:NAIGnore:HSMode {<Boolean>}
:TRIGger:LOGic:I2CBus:NAIGnore:HSMode?
```

例
```
:TRIGGER:LOGIC:I2CBUS:NAIGNORE:HSMODE ON
:TRIGGER:LOGIC:I2CBUS:NAIGNORE:HSMODE? -> :TRIGGER:LOGIC:I2CBUS:NAIGNORE:HSMODE 1
```

:TRIGger:LOGic:I2CBus:NAIGnore:RACCess

機能　ロジック I²C バストリガのリードアクセスモードで NON ACK を無視する / しないを設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:I2CBus:NAIGnore:RACCess {<Boolean>}
:TRIGger:LOGic:I2CBus:NAIGnore:RACCess?
```

例
```
:TRIGGER:LOGIC:I2CBUS:NAIGNORE:RACCESS ON
:TRIGGER:LOGIC:I2CBUS:NAIGNORE:RACCESS? -> :TRIGGER:LOGIC:I2CBUS:NAIGNORE:RACCESS 1
```

:TRIGger:LOGic:I2CBus:NAIGnore:SBYTe (Start Byte)

機能　ロジック I²C バストリガのスタートバイトで NON ACK を無視する / しないを設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:I2CBus:NAIGnore:SBYTe {<Boolean>}
:TRIGger:LOGic:I2CBus:NAIGnore:SBYTe?
```

例
```
:TRIGGER:LOGIC:I2CBUS:NAIGNORE:SBYTE ON
:TRIGGER:LOGIC:I2CBUS:NAIGNORE:SBYTE?
-> :TRIGGER:LOGIC:I2CBUS:NAIGNORE:SBYTE 1
```

:TRIGger:LOGic:I2CBus:SBHSmode?

機能　ロジック I²C バストリガのスタートバイト / ハイスピードモードに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:LOGic:I2CBus:SBHSmode?
```

:TRIGger:LOGic:I2CBus:SBHSmode:TYPE

機能　ロジック I²C バストリガのスタートバイト / ハイスピードモードのタイプを設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:I2CBus:SBHSmode:TYPE {HSMode|SBYTe}
:TRIGger:LOGic:I2CBus:SBHSmode:TYPE?
```

例
```
:TRIGGER:LOGIC:I2CBUS:SBHSMODE:TYPE HSMODE
:TRIGGER:LOGIC:I2CBUS:SBHSMODE:TYPE?
-> :TRIGGER:LOGIC:I2CBUS:SBHSMODE:TYPE HSMODE
```

:TRIGger:LOGic:LINBus?

機能　ロジック LIN バス信号トリガに関するすべての設定値を問い合わせします。

構文
```
:TRIGger:LOGic:LINBus?
```

:TRIGger:LOGic:LINBus:BRATe

機能　ロジック LIN バス信号トリガのビットレート (データ転送速度) を設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:LINBus:BRATe {<NRf>|USER,<NRf>}
:TRIGger:LOGic:LINBus:BRATe?
```
<NRf> = 1200、2400、4800、9600、19200
USER の <NRf> =本体ユーザーズマニュアル参照。

例
```
:TRIGGER:LOGIC:LINBUS:BRATE 19200
:TRIGGER:LOGIC:LINBUS:BRATE?
-> :TRIGGER:LOGIC:LINBUS:BRATE 19200
```

:TRIGger:LOGic:LINBus:SOURce

機能　ロジック LIN バス信号トリガのトリガソースを設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:LINBus:SOURce {A<x>|B<x>|C<x>|D<x>}
:TRIGger:LOGic:LINBus:SOURce?
```
<x> = 0 ～ 7

例
```
:TRIGGER:LOGIC:LINBUS:SOURCE A0
:TRIGGER:LOGIC:LINBUS:SOURCE?
-> :TRIGGER:LOGIC:LINBUS:SOURCE A0
```

解説　DLM6000 の 16 ビットモデルでは {A<x>|C<x>} が有効です。

:TRIGger:LOGic:SPIBus?

機能　ロジック SPI バストリガに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:LOGic:SPIBus?
```

:TRIGger:LOGic:SPIBus:BITorder

機能　ロジック SPI バストリガのビットオーダーを設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:SPIBus:BITorder {LSBFirst|MSBFirst}
:TRIGger:LOGic:SPIBus:BITorder?
```

例
```
:TRIGGER:LOGIC:SPIBUS:BITORDER LSBFIRST
:TRIGGER:LOGIC:SPIBUS:BITORDER?
-> :TRIGGER:LOGIC:SPIBUS:BITORDER LSBFIRST
```

:TRIGger:LOGic:SPIBus:CLOCk?

機能　ロジック SPI バストリガのクロックに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:LOGic:SPIBus:CLOCk?
```

:TRIGger:LOGic:SPIBus:CLOCk:POLarity

機能　ロジック SPI バストリガのクロックトレースの極性を設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:SPIBus:CLOCk:POLarity {FALL|RISE}
:TRIGger:LOGic:SPIBus:CLOCk:POLarity?
```

例
```
:TRIGGER:LOGIC:SPIBUS:CLOCK:POLARITY FALL
:TRIGGER:LOGIC:SPIBUS:CLOCK:POLARITY?
-> :TRIGGER:LOGIC:SPIBUS:CLOCK:POLARITY FALL
```

:TRIGger:LOGic:SPIBus:CLOCk:SOURce

機能　ロジック SPI バストリガのクロックトレースを設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:SPIBus:CLOCk:SOURce {A<x>|B<x>|C<x>|D<x>}
:TRIGger:LOGic:SPIBus:CLOCk:SOURce?
```
<x> = 0 ～ 7

例
```
:TRIGGER:LOGIC:SPIBUS:CLOCK:SOURCE A0
:TRIGGER:LOGIC:SPIBUS:CLOCK:SOURCE?
-> :TRIGGER:LOGIC:SPIBUS:CLOCK:SOURCE A0
```

解説　DLM6000 の 16 ビットモデルでは {A<x>|C<x>} が有効です。

:TRIGger:LOGic:SPIBus:CS?

機能　ロジック SPI バストリガのチップセレクトに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:LOGic:SPIBus:CS?
```

:TRIGger:LOGic:SPIBus:CS:ACTive

機能　ロジック SPI バストリガのチップセレクトのアクティブレベルを設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:SPIBus:CS:ACTive {HIGH|LOW}
:TRIGger:LOGic:SPIBus:CS:ACTive?
```

例
```
:TRIGGER:LOGIC:SPIBUS:CS:ACTIVE HIGH
:TRIGGER:LOGIC:SPIBUS:CS:ACTIVE?
-> :TRIGGER:LOGIC:SPIBUS:CS:ACTIVE HIGH
```

:TRIGger:LOGic:SPIBus:CS:SOURce

機能　ロジック SPI バストリガのチップセレクトトレースを設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:SPIBus:CS:SOURce {A<x>|B<x>|C<x>|D<x>}
:TRIGger:LOGic:SPIBus:CS:SOURce?
```
<x> = 0 ～ 7

例
```
:TRIGGER:LOGIC:SPIBUS:CS:SOURCE A0
:TRIGGER:LOGIC:SPIBUS:CS:SOURCE?
-> :TRIGGER:LOGIC:SPIBUS:CS:SOURCE A0
```

解説　DLM6000 の 16 ビットモデルでは {A<x>|C<x>} が有効です。

:TRIGger:LOGic:SPIBus:DATA<x>?

機能　ロジック SPI バストリガの各データに関するすべての設定値を問い合わせます。

構文
```
:TRIGger:LOGic:SPIBus:DATA<x>?
```
<x> = 1、2

:TRIGger:LOGic:SPIBus:DATA<x>:BYTE

機能　ロジック SPI バストリガの各データの設定データ数を設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:SPIBus:DATA<x>:
BYTE {<NRf>}
:TRIGger:LOGic:SPIBus:DATA<x>:BYTE?
```
<x> = 1、2
<NRf> = 1 ～ 4

例
```
:TRIGGER:LOGIC:SPIBUS:DATA1:BYTE 1
:TRIGGER:LOGIC:SPIBUS:DATA1:BYTE?
-> :TRIGGER:LOGIC:SPIBUS:DATA1:BYTE
1
```

:TRIGger:LOGic:SPIBus:DATA<x>:CONDition

機能　ロジック SPI バストリガの各データの判定方法 (一致 / 不一致) を設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:SPIBus:DATA<x>:
CONDition {FALSe|TRUE}
:TRIGger:LOGic:SPIBus:DATA<x>:
CONDition?
```
<x> = 1、2

例
```
:TRIGGER:LOGIC:SPIBUS:DATA1:
CONDITION FALSE
:TRIGGER:LOGIC:SPIBUS:DATA1:
CONDITION? -> :TRIGGER:LOGIC:SPIBUS:
DATA1:CONDITION FALSE
```

:TRIGger:LOGic:SPIBus:DATA<x>:DPOSition

機能　ロジック SPI バストリガの各データのパターン比較先頭位置を設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:SPIBus:DATA<x>:
DPOSition {<NRf>}
:TRIGger:LOGic:SPIBus:DATA<x>:
DPOSition?
```
<x> = 1、2
<NRf> = 0 ～ 9999

例
```
:TRIGGER:LOGIC:SPIBUS:DATA1:
DPOSITION 1
:TRIGGER:LOGIC:SPIBUS:DATA1:
DPOSITION? -> :TRIGGER:LOGIC:SPIBUS:
DATA1:DPOSITION 1
```

:TRIGger:LOGic:SPIBus:DATA<x>:HEXA<x>

機能　ロジック SPI バストリガの各データを HEXA で設定します。

構文
```
:TRIGger:LOGic:SPIBus:DATA<x>:
HEXA<x> {< 文字列 >}
```
DATA<x> の <x> = 1、2
HEXA<x> の <x> = 1 ～ 4
< 文字列 > = '0' ～ 'F'、'X' の組み合わせ 2 文字。

例
```
:TRIGGER:LOGIC:SPIBUS:DATA1:
HEXA1 "AB"
```

:TRIGger:LOGic:SPIBus:DATA<x>:PATTern<x>

機能　ロジック SPI バストリガの各データを BINARY で設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:SPIBus:DATA<x>:
PATTern<x> {< 文字列 >}
:TRIGger:LOGic:SPIBus:DATA<x>:
PATTern<x>?
```
DATA<x> の <x> = 1、2
PATTern<x> の <x> = 1 ～ 4
< 文字列 > = '0'、'1'、'X' の組み合わせ 8 文字。

例
```
:TRIGGER:LOGIC:SPIBUS:DATA1:
PATTERN1 "10101011"
:TRIGGER:LOGIC:SPIBUS:DATA1:
PATTERN1?
-> :TRIGGER:LOGIC:SPIBUS:DATA1:
PATTERN1 "10101011"
```

:TRIGger:LOGic:SPIBus:DATA<x>:SOURce

機能　ロジック SPI バストリガの各データのトレースを設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:SPIBus:DATA<x>:SOURce
{A<y>|B<y>|C<y>|D<y>}
:TRIGger:LOGic:SPIBus:DATA<x>:
SOURce?
```
<x> = 1、2
<y> = 0 ～ 7

例
```
:TRIGGER:LOGIC:SPIBUS:DATA1:SOURCE
A0
:TRIGGER:LOGIC:SPIBUS:DATA1:SOURCE?
-> :TRIGGER:LOGIC:SPIBUS:DATA1:
SOURCE A0
```

解説　DLM6000 の 16 ビットモデルでは {A<y>|C<y>} が有効です。

:TRIGger:LOGic:SPIBus:MODE

機能　ロジック SPI バストリガの結線方式 (3 線式 /4 線式) を設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:SPIBus:MODE {WIRE3|
WIRE4}
:TRIGger:LOGic:SPIBus:MODE?
```

例
```
:TRIGGER:LOGIC:SPIBUS:MODE WIRE3
:TRIGGER:LOGIC:SPIBUS:MODE?
-> :TRIGGER:LOGIC:SPIBUS:MODE WIRE3
```

:TRIGger:LOGic:UART?

機能　ロジック UART 信号トリガに関するすべて設定値を問い合わせします。

構文
```
:TRIGger:LOGic:UART?
```

:TRIGger:LOGic:UART:BRATe

機能　ロジック UART 信号トリガのビットレート (データ転送速度) を設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:UART:
BRATe {<NRf>|USER,<NRf>}
:TRIGger:LOGic:UART:BRATe?
```
<NRf> = 1200、2400、4800、9600、19200、38400、57600、115200
USER の <NRf> =本体ユーザーズマニュアル参照。

例
```
:TRIGGER:LOGIC:UART:BRATE 19200
:TRIGGER:LOGIC:UART:BRATE?
-> :TRIGGER:LOGIC:UART:BRATE 19200
```

:TRIGger:LOGic:UART:FORMat

機能　ロジック UART 信号トリガのフォーマットを設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:UART:
FORMat {BIT7parity|BIT8Noparity|
BIT8Parity}
:TRIGger:LOGic:UART:FORMat?
```

例
```
:TRIGGER:LOGIC:UART:FORMAT
BIT7PARITY
:TRIGGER:LOGIC:UART:FORMAT?
-> :TRIGGER:LOGIC:UART:
FORMAT BIT7PARITY
```

:TRIGger:LOGic:UART:POLarity

機能　ロジック UART 信号トリガの極性を設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:UART:
POLarity {NEGative|POSitive}
:TRIGger:LOGic:UART:POLarity?
```

例
```
:TRIGGER:LOGIC:UART:POLARITY
NEGATIVE
:TRIGGER:LOGIC:UART:POLARITY?
-> :TRIGGER:LOGIC:UART:
POLARITY NEGATIVE
```

:TRIGger:LOGic:UART:SOURce

機能　ロジック UART 信号トリガのトリガソースを設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:UART:SOURce
{A<x>|B<x>|C<x>|D<x>}
:TRIGger:LOGic:UART:SOURce?
```
<x> = 0 ～ 7

例
```
:TRIGGER:LOGIC:UART:SOURCE A0
:TRIGGER:LOGIC:UART:SOURCE?
-> :TRIGGER:LOGIC:UART:SOURCE A0
```

解説　DLM6000 の 16 ビットモデルでは {A<x>|C<x>} が有効です。

:TRIGger:LOGic:UART:SPOint

機能　ロジック UART 信号トリガのサンプルポイントを設定 / 問い合わせします。

構文
```
:TRIGger:LOGic:UART:SPOint {<NRf>}
:TRIGger:LOGic:UART:SPOint?
```
<NRf> = 18.8 ～ 90.6(%)

例
```
:TRIGGER:LOGIC:UART:SPOINT 18.8
:TRIGGER:LOGIC:UART:SPOINT?
-> :TRIGGER:LOGIC:UART:
SPOINT 18.8E+00
```

:TRIGger:SOURce:CHANnel<x>:LEVel

機能　各チャネルのトリガレベルを設定 / 問い合わせします。

構文
```
:TRIGger:SOURce:CHANnel<x>:
LEVel {<電圧>|<電流>}
:TRIGger:SOURce:CHANnel<x>:LEVel?
```
<x> = 1 ～ 4
<電圧>、<電流> = 3.1 または 3.4 節参照。

例
```
:TRIGGER:SOURCE:CHANNEL1:LEVEL 1V
:TRIGGER:SOURCE:CHANNEL1:LEVEL?
-> :TRIGGER:SOURCE:CHANNEL1:
LEVEL 1.000E+00
```

解説　下記コマンドで設定したソースに対応するチャネルが対象です。
- 「:TRIGger:ENHanced:I2CBus:CLOCk:SOURce」
- 「:TRIGger:ENHanced:I2CBus:DATA:SOURce」
- 「:TRIGger:ENHanced:SPIBus:CLOCk:SOURce」
- 「:TRIGger:ENHanced:SPIBus:CS:SOURce」
- 「:TRIGger:ENHanced:SPIBus:DATA[1-2]:SOURce」

:TRIGger:SOURce:CHANnel<x>:STATe

機能　各チャネルの成立条件を設定 / 問い合わせします。

構文
```
:TRIGger:SOURce:CHANnel<x>:
STATe {DONTcare|HIGH|LOW}
:TRIGger:SOURce:CHANnel<x>:STATe?
```
<x> = 1 ～ 4

例
```
:TRIGGER:SOURCE:CHANNEL1:STATE HIGH
:TRIGGER:SOURCE:CHANNEL1:STATE?
-> :TRIGGER:SOURCE:CHANNEL1:
STATE HIGH
```

解説　「:TRIGger:TYPE I2CBus」のときに有効です。

:TRIGger:TYPE

機能　トリガの種類を設定 / 問い合わせします。

構文
```
:TRIGger:TYPE {CANBus|EDGE|EICYcle|
EIDelay|EISequence|EOR|EQUalify|
I2CBus|LEDGe|LINBus|LI2Cbus|
LLINbus|LSPAttern|LSPIbus|LPSTate|
LPULse|LQUalify|LSTate|LUARt|
PQUalify|PSTAte|PULSe|SPATtern|
SPIBus|STATe|TV|UART}
:TRIGger:TYPE?
```

例
```
:TRIGGER:TYPE CANBUS
:TRIGGER:TYPE?
-> :TRIGGER:TYPE CANBUS
```

解説　{LEDGe|LI2Cbus|LLINbus|LSPAttern|LSPIbus|LPSTate|LPULse|LQUalify|LSTate|LUARt} は、DLM6000 に適用できます。

8.1　I²C バス信号解析

項目	仕様
バス転送レート	最大 3.4Mbit/s
アドレスモード	7bits/10bits
トリガ	
トリガソース	DL6000 シリーズの場合、CH1 ～ CH4 から選択。 DLM6000 シリーズの場合、CH1 ～ CH4、A0 ～ A7、B0 ～ B7、C0 ～ C7、および D0 ～ D7(DLM6054-L16 と DLM6104-L16 は A0 ～ A7 と C0 ～ C7) から選択。
モード	Every Start：　スタートコンディションを検出したときにトリガ ADR&DATA：　設定されたアドレスおよびデータとの比較でトリガ アドレスタイプ ・7bit Address ・7bit + Sub Address ・10bit Address NON ACK：　Nack を検出したときにトリガ General Call：　ジェネラルコールアドレスのセカンドバイトパターンとの比較でトリガ Start Byte/HS Mode：　スタートバイトまたは HS モードのマスターアドレスでトリガ
解析	
信号	DL6000 シリーズの場合、CH1 ～ CH4 または M1 ～ M4 から選択。 DLM6000 シリーズの場合、CH1 ～ CH4、M1 ～ M4、A0 ～ A7、B0 ～ B7、C0 ～ C7、および D0 ～ D7(DLM6054-L16 と DLM6104-L16 は A0 ～ A7 と C0 ～ C7) から選択。
解析可能データ数	最大 40000 バイト分 (解析基準点前後 20000 バイト)
解析結果表示	Simple(簡易表示) 解析番号 (No.)、スタートコンディション / ストップコンディション (S/P)、データの 16 進数表示 (Hex)、アドレス / データ (Form)、Read/Write 信号 (R/W)、Acknowledge ビット (ACK) Detail(詳細表示) 解析番号 (No.)、スタートコンディション / ストップコンディション (S/P)、トリガポジションからの時間 (Time(ms))、データの 2 進数表示 (Binary)、データの 16 進数表示 (Hex)、アドレス / データ (Form)、Read/Write 信号 (R/W)、Acknowledge ビット (ACK)、データ情報 (Info)
ズームリンク	解析結果リストでハイライト表示されたバイトにズーム位置 (ズームボックスの中心) を移動可能。ズーム位置を変更すると、解析結果リストのハイライト表示も連動して移動。
解析結果リストのデータ保存	解析結果リストの簡易表示および詳細表示のデータを CSV 形式で保存可能 (拡張子：.csv)。
データサーチ	アドレスパターン / データパターン /Acknowledge ビットの状態を設定して波形をサーチ可能。条件と一致する波形がみつかると、ズームボックスがそのポイントに移動して指定した波形をズーム波形エリアに拡大表示。

8.2 CAN バス信号解析

項目	仕様
バージョン	CAN Version 2.0B
ビットレート	1M、500k、250k、125k、83.3k、33.3k[bps]、または 10k ～ 1M[bps] の任意のビットレートを設定可能 (設定分解能は 0.1kbps)。 High speed CAN(ISO11898)、Low speed CAN(ISO11519-2) に対応。
トリガ	
トリガソース	CH1 ～ CH4 から選択。
モード	SOF： SOF(Start of Frame) でトリガ Error Frame： エラーフレームでトリガ ID Std/Data： データフレーム / リモートフレーム (ID: 標準フォーマット) でトリガ ID Ext/Data： データフレーム / リモートフレーム (ID: 拡張フォーマット) でトリガ ID/Data OR： 4 種類のデータフレーム / リモートフレームの OR 条件でトリガ ID ごとに標準フォーマット / 拡張フォーマットを選択可能
解析	
信号	CH1 ～ CH4 と M1 ～ M4 から選択。
解析可能フレーム数	最大 3000 フレーム分 (解析基準点前後 1500 フレーム)
解析対象フレーム	データフレーム、リモートフレーム、エラーフレーム、オーバーロードフレーム
解析結果表示	Simple(簡易表示) 解析番号 (No.)、フレームの種類 (Frame)、ID の 16 進数表示 (ID)、Data の 16 進数表示 (Data)、ACK スロットの状態 (Ack) Detail(詳細表示) 解析番号 (No.)、フレームの種類 (Frame)、トリガポジションからの時間 (Time(ms))、ID の 16 進数表示 (ID)、DLC の 16 進数表示 (DLC)、Data の 2 進数表示 (Data(Bin))、Data の 16 進数表示 (Data)、CRC シーケンスの 16 進数表示 (CRC)、ACK スロットの状態 (Ack)
ズームリンク	ズームリンク 解析結果リストでハイライト表示されたフレームの先頭にズーム位置 (ズームボックスの中心) を移動可能。ズーム位置を変更すると、解析結果リストのハイライト表示も連動して移動。 フィールドジャンプ ズームリンクが有効な場合、解析結果リストでハイライト表示されたフレームの指定したフィールドの先頭にズーム位置をジャンプ可能。フィールドは、SOF、ID、Control Field、Data Field、CRC、ACK の中から選択。
解析結果リストのデータ保存	解析結果リストの簡易表示および詳細表示のデータを CSV 形式で保存可能 (拡張子：.csv)。
データサーチ	フィールドやフレームの条件を指定して波形をサーチ可能。条件と一致する波形がみつかると、ズームボックスがそのポイントに移動して指定した波形をズーム波形エリアに拡大表示。

8.3 LIN バス信号解析

項目	仕様
レビジョン	LIN1.3 または LIN2.0
ビットレート	1200bps、2400bps、4800bps、9600bps、19200bps、または 1k ～ 20k[bps] の任意のビットレートを設定可能 (設定分解能は 10bps)。
トリガ	
トリガソース	DL6000 シリーズの場合、CH1 ～ CH4 から選択。 DLM6000 シリーズの場合、CH1 ～ CH4、A0 ～ A7、B0 ～ B7、C0 ～ C7、および D0 ～ D7(DLM6054-L16 と DLM6104-L16 は A0 ～ A7 と C0 ～ C7) から選択。
モード	Break： Break delimiter でトリガ
解析	
信号	DL6000 シリーズの場合、CH1 ～ CH4 または M1 ～ M4 から選択。 DLM6000 シリーズの場合、CH1 ～ CH4、M1 ～ M4、A0 ～ A7、B0 ～ B7、C0 ～ C7、および D0 ～ D7(DLM6054-L16 と DLM6104-L16 は A0 ～ A7 と C0 ～ C7) から選択。
解析可能フレーム数	最大 3000 フレーム (解析基準点前後 1500 フレーム)
解析対象フィールド	Break、Synch、ID、Data、Checksum
解析レビジョン	LIN 2.0、LIN1.3、または Both(両レビジョン同時)
解析結果表示	Simple(簡易表示) 解析番号 (No.)、ID の 16 進数表示 (ID)、Data の 16 進数表示 (Data)、Checksum の 16 進数表示 (Checksum) Detail(詳細表示) 解析番号 (No.)、トリガポジションからの時間 (Time(ms))、ID の 16 進数表示 (ID)、ID-Field の 16 進数表示 (ID-Field)、Data の 2 進数表示 (Data(Bin))、Data の 16 進数表示 (Data)、Checksum の 16 進数表示 (Checksum)、付加情報 (Information)
ズームリンク	ズームリンク 解析結果リストでハイライト表示されたフレームの先頭にズーム位置 (ズームボックスの中心) を移動可能。ズーム位置を変更すると、解析結果リストのハイライト表示も連動して移動。 フィールドジャンプ ズームリンクが有効な場合、解析結果リストでハイライト表示されたフレームの指定したフィールドの先頭にズーム位置をジャンプ可能。フィールドは、Break、Synch、ID、Data,Checksum の中から選択。
解析結果リストのデータ保存	解析結果リストの簡易表示および詳細表示のデータを CSV 形式で保存可能 (拡張子：.csv)。
データサーチ	フィールドやフレームの条件を指定して波形をサーチ可能。条件と一致する波形がみつかると、ズームボックスがそのポイントに移動して指定した波形をズーム波形エリアに拡大表示。

8.4 SPI バス信号解析

項目	仕様
トリガ	
トリガソース	DL6000 シリーズの場合、CH1 ～ CH4 から選択。 DLM6000 シリーズの場合、CH1 ～ CH4、A0 ～ A7、B0 ～ B7、C0 ～ C7、および D0 ～ D7(DLM6054-L16 と DLM6104-L16 は A0 ～ A7 と C0 ～ C7) から選択。
モード	3 線式、4 線式 CS がアサートされてから任意のバイトカウントからのデータを比較してトリガ。比較するデータ長は 1 ～ 4 バイトから選択可能。
解析	
信号	DL6000 シリーズの場合、CH1 ～ CH4 または M1 ～ M4 から選択。 DLM6000 シリーズの場合、CH1 ～ CH4、M1 ～ M4、A0 ～ A7、B0 ～ B7、C0 ～ C7、および D0 ～ D7(DLM6054-L16 と DLM6104-L16 は A0 ～ A7 と C0 ～ C7) から選択。
解析可能データ数	最大 40000 バイト分 (解析基準点前後 20000 バイト)
解析対象フレーム	Data
解析結果表示	Simple(簡易表示) 解析番号 (No.)、Data1 の 16 進数表示 (Data1(H))、Data2 の 16 進数表示 (Data2(H))、CS のステータス (CS) Detail(詳細表示) 解析番号 (No.)、Trigger Position から各バイトの先頭のビットまでの時間 (Time(ms))、Data1 の 2 進数表示 (Data1(B))、Data1 の 16 進数表示 (Data1(H))、Data2 の 2 進数表示 (Data2(B))、Data2 の 16 進数表示 (Data2(H))、CS のステータス (CS)、アクティブ期間の開始位置 / 終了位置 (S/P)
ズームリンク	解析結果リストでハイライト表示されたバイトにズーム位置 (ズームボックスの中心) を移動可能。ズーム位置を変更すると、解析結果リストのハイライト表示も連動して移動。
解析結果リストのデータ保存	解析結果リストの簡易表示および詳細表示のデータを CSV 形式で保存可能 (拡張子：.csv)。
データサーチ	データのパターンを指定して波形をサーチ可能。パターンと一致する波形がみつかると、ズームボックスがそのポイントに移動して指定した波形をズーム波形エリアに拡大表示。

8.5 UART 信号解析

項目	仕様
ビットレート	1200bps、2400bps、4800bps、9600bps、19200bps、38400bps、57600bps、115200bps、または 1k ～ 200k[bps] の任意のビットレートを設定可能 (設定分解能は 100bps)。
データ形式	8bit Data (Parity bit 無し) 7bit Data + Parity bit(エラートリガ時だけ選択可) 8bit Data + Parity bit(エラートリガ時だけ選択可)
トリガ	
トリガソース	DL6000 シリーズの場合、CH1 ～ CH4 から選択。 DLM6000 シリーズの場合、CH1 ～ CH4、A0 ～ A7、B0 ～ B7、C0 ～ C7、および D0 ～ D7(DLM6054-L16 と DLM6104-L16 は A0 ～ A7 と C0 ～ C7) から選択。
モード	Every Data： すべてのデータの Stop Bit の位置でトリガ
解析	
信号	DL6000 シリーズの場合、CH1 ～ CH4 または M1 ～ M4 から選択。 DLM6000 シリーズの場合、CH1 ～ CH4、M1 ～ M4、A0 ～ A7、B0 ～ B7、C0 ～ C7、および D0 ～ D7(DLM6054-L16 と DLM6104-L16 は A0 ～ A7 と C0 ～ C7) から選択。
解析可能フレーム数	最大 3000 バイト (解析基準点前後 1500 バイト)
解析対象フィールド	Data
解析結果表示	Simple(簡易表示) 解析番号 (No.)、Data の 16 進数表示 (Data)*、付加情報 (Information) Detail(詳細表示) 解析番号 (No.)、トリガポジションからの時間 (Time(ms))、Data の 2 進数表示 (Data(Bin))*、Data の 16 進数表示 (Data)*、付加情報 (Information) 表示モード 16 進数の ASCII 表示、データのグループ化
ズームリンク	解析結果リストでハイライト表示されたフレームの先頭にズーム位置 (ズームボックスの中心) を移動可能。ズーム位置を変更すると、解析結果リストのハイライト表示も連動して移動。
解析結果リストのデータ保存	解析結果リストの簡易表示および詳細表示のデータを CSV 形式で保存可能 (拡張子：.csv)。
データサーチ	フィールドやフレームの条件を指定して波形をサーチ可能。条件と一致する波形がみつかると、ズームボックスがそのポイントに移動して指定した波形をズーム波形エリアに拡大表示。

* 表示モードを「ASCII」にすると、 ASCII 表示になります。

索引