HIOKI

メモリハイロガ―LR8400 プロバイダ

Version 1.0.1

ユーザーズ ガイド

April 20, 2020

備考:

【改版履歴】

バージョン	日付	内容
1.0.0	2019-10-7	初版
1.0.1	2020-4-20	バグ修正　メモリリーク

【動作確認機器】

機種	注意事項
LR8400 シリーズ	TCP/IP, USB による仮想 COM(シリアル)接続に対応.

目次

1. はじめに

本書は，HIOKI 製メモリハイロガー(LR8400)に対し TCP/IP またはシリアル接続し，通信コマンドを用いてデータの送受信を行う CAO プロバイダのユーザーズガイドです．本書で扱う CAO プロバイダ(CaoProvHIOKILR8400.dll)を LR8400 プロバイダと呼びます．

第 2 章に LR8400 プロバイダの概要，変数やコマンドの詳細を記載しています．

LR8400 プロバイダで実装している通信コマンドの対応状況については，LR8400 のコマンド対応表を参照ください．通信コマンドについては LR8400 シリーズの取扱説明書の「通信コマンドについて」や LR8400 本体付属のロガーユーティリティ (CD-R)内の通信コマンド取扱説明を参照ください．

下記に PC と LR8400 の接続イメージを示します．USB 接続の場合，ドライバーのインストールが必要です．また LAN で接続する場合，IP を設定してください．

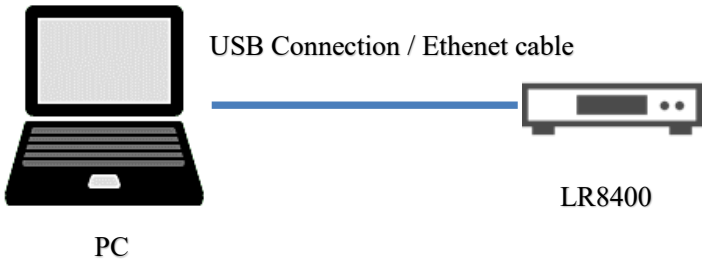

図 1-1 接続イメージ

2. プロバイダの概要

2.1. 概要

LR8400 プロバイダは, HIOKI 製メモリハイロガーに対して TCP/IP またはシリアル接続し, 通信コマンド[1]を用いてデータの送受信を行う CAO プロバイダです.

そのファイル形式は DLL(Dynamic Link Library)であり, CAO エンジンから使用時に動的ロードされます. LR8400 プロバイダを使用するにあたっては ORiN2SDK をインストールするか, 下表を参照して手作業でレジストリ登録を行う必要があります.

表 2-1 LR8400 プロバイダ

ファイル名	CaoProvHIOKILR8400.dll
ProgID	CaoProv.HIOKI.LR8400
レジストリ登録	regsvr32 CaoProvHIOKILR8400.dll
レジストリ登録の抹消	regsvr32 /u CaoProvHIOKILR8400.dll

[1] 通信コマンドの詳細は, LR8400 本体付属のロガーユーティリティ (CD-R)を参照してください. 通信コマンドの説明が HTML 形式で保存されています.

2.2. 制限事項

本プロバイダを使用する際は，下記に注意して使用してください．

- LR8400 プロバイダは，本体機器からの応答データの解析をヘッダあり前提で作成されています．ヘッダの設定[2]を ON 以外に設定しないで下さい．
- TCP 接続の場合，LR8400 本体機器の仕様で後勝ちとなります．本プロバイダで TCP 接続中に他の PC 等が LR8400 に対して TCP 接続をした場合は，本プロバイダの接続が切断されます．
- 本体機器の通信設定は，LR8400 プロバイダにて接続後には変更しないで下さい．
  もし，変更してしまった場合は，本体機器の再起動後に再度 LR8400 プロバイダにて接続し直して下さい．
- 通信コマンド[1]の仕様上，1 秒より短い記録間隔でリアルタイムデータを取得することはできません．ただし，1 秒より短い記録間隔でも測定停止後のデータは取得できます．

[2] ヘッダの設定は，本体機器のシステム > 通信設定 を参照ください．

2.3. メソッド・プロパティ

2.3.1. CaoWorkspace::AddController メソッド

LR8400 プロバイダは AddController 時に通信用の接続パラメータを設定し，通信の接続を行います.

書式　AddController (<bstrCtrlName:BSTR>,<bstrProvName:BSTR>,
<bstrPcName:BSTR>, [<bstrOption:BSTR>])

<bstrCtrlName>	:	[in] コントローラ名
<bstrProvName>	:	[in] プロバイダ名．固定値 =" CaoProv.HIOKI.LR8400"
<bstrPcName>	:	[in] プロバイダの実行マシン名（未使用）
<bstrOption>	:	[in] オプション文字列

使用例

```
// エンジンの生成
var eng = new CCaoEngine();
// ワークスペースの生成
var ws = eng.AddWorkspace("SampleWorkSpace","");

// TCP 接続の場合
var option =
"Conn=TCP:192.168.1.1:8802,Delimiter=1,ConnTimeout=5000,TimeOut=4000";
// シリアル接続の場合
// var option =
"Conn=COM:3:9600:N:8:1,Delimiter=1,ConnTimeout=5000,TimeOut=4000";

// LR8400 への接続
this.ctrl = ws.AddController("Sample","CaoProv.HIOKI.LR8400","",option);
```

以下にオプション文字列に指定するリストを示します.

表 2-2 CaoWorkspace::AddController のオプション文字列

オプション	意味
Conn=<接続パラメータ>	必須. 通信形態とその接続パラメータを設定します. (参照 2.3.1.1)
Delimiter[=<デリミタ番号>]	コマンド通信時のデリミタ[3]を指定します. (デフォルト:1) 0 : LF 1 : CR+LF
ConnTimeout[=<タイムアウト時間>]	接続時のタイムアウト時間(ミリ秒)を指定します. (デフォルト:3000)
Timeout[=<タイムアウト時間>]	応答コマンドのタイムアウト時間(ミリ秒)を指定します. (デフォルト:3000)

[3] デリリミタの設定は, 本体機器のシステム＞通信設定を参照してください

2.3.1.1. Conn オプション

以下に Conn オプションの接続パラメータ文字列を示します. ここで角括弧("[ ]")内は省略可能を示します. また, 各パラメータの解説中の下線部はオプションを指定しなかったときのデフォルト値を示します.

【シリアルデバイス】

"Conn=com:<COM Port>"

<COM Port> : COM ポート番号. '1'-COM1, '2'-COM2, …

【Ethernet デバイス】

TCP/IP

"Conn=TCP:<Dest IP Address>:<Dest Port No>[:<Src IP Address>:<Src Port No>]"

<Dest IP Address> : 接続先の IP アドレス.

<Dest Port No> : 接続先のポート番号.

※本体機器の仕様で, 下 1 桁は 2 を指定してください.

例) 880X ⇒ 8802

<Src IP Address> : 自 IP アドレス. 127.0.0.1

<Src Port No> : 自ポート番号. 5001

2.3.2. CaoController::GetVariableNames プロパティ

2.4 に示しているシステム変数名の一覧を取得します.

2.3.3. CaoController::AddVariable メソッド

LR8400 に対しデータの読出しを行うための変数オブジェクトを作成します.

書式　AddVariable (<bstrVariableName:BSTR>, [<bstrOption:BSTR>])

<bstrVariableName> : [in] 変数名
<bstrOption> : [in] オプション文字列

使用例

```
var result = this.ctrl .AddVariable("@MAKER_NAME","");
```

2.3.4. CaoController::Execute メソッド

CaoController クラスの Execute メソッドは, コマンドを実行するためのメソッドです. 各コマンドの詳細はコマンドリファレンスを参照してください.

書式　Execute (<bstrCommandName:VT_BSTR>,[<vntParam : VT_VARIANT>])

<bstrCommandName> : [in] コマンド名
<vntParam> : [in] パラメータ

使用例

```
var result = this.ctrl.Execute("Send", ":START");
```

2.3.5. CaoVariable::get_Value プロパティ

引数で渡された値をオプション指定に従い, アクセス対象のデータに対し読出しコマンドを送出します.

2.4. 変数一覧

2.4.1. CaoController クラス

2.4.1.1. システム変数

表 2-3 CaoController クラス システム変数一覧

変数名	データ型	説明	属性	
			get	put
@MAKER_NAME	VT_BSTR	メーカー名=HIOKI を返す.	○	-
@VERSION	VT_BSTR	プロバイダバージョンを返す.	○	-
@TITLE_COMMENT	VT_BSTR	LR8400 のタイトルコメントを返す. (タイトルコメントの問い合わせコマンドを送信し, その応答内容を返します.)	○	-

2.4.1.2. ユーザ変数

表 2-4 CaoController クラス ユーザ変数一覧

全ての変数で get 操作のみ可能です.

変数名[4]	データ型	説明	属性	
			get	put
UI<??>	VT_BSTR	指定したチャネルの入力種類の設定値を取得します. (参照:2.4.1.2.1)	○	-
UR<??>	VT_R8	指定したチャネルの測定レンジの設定値を取得します. (参照:2.4.1.2.2)	○	-
CCH<??>	VT_BSTR	指定したチャネルのコメント文字の設定を取得します. (参照:2.4.1.2.3)	○	-
MVREAL<??>	VT_R8	指定したチャネルのリアルタイムデータ出力(電圧, 温度, 湿度, 抵抗, 積算, 回転数)の値を電圧値に変換した値を取得します. (参照:2.4.1.2.4)	○	-
MTVREAL<??>	VT_ARRAY | VT_R8	指定したユニットのリアルタイムデータ出力(測定 ON のチャネルをすべて)の値を電圧値に変換した値を取得します. (参照:2.4.1.2.5)	○	-

[4] <??>には, 任意の文字列を指定してください.

変数名[4]	データ型	説明	属性	
			get	put
MTVRCH<??>	VT_ARRAY ｜VT_BSTR	指定したユニットの測定 ON のチャネル名を全て取得します.（参照:0）	○	－

2.4.1.2.1. UI <??>

指定したチャネルの入力種類を取得する変数です.

表 2-5 に変数作成時の必須オプションを, 表 2-6 には取得値のインデックスとデータ内容について示します.

表 2-5 UI 必須オプション

オプション	値範囲	説明
CH	CH1_1 ～ CH4_15	入力種類を取得するチャネル 例）CH = CH1_1

表 2-6 UI 取得値

説明
入力種類 VOLTAGE : 電圧 TC : 熱電対 RTD : 測温抵抗体 HUMIDITY : 湿度 RESIST : 抵抗

使用例

```
var userVar= this.ctrl.AddVariable("UI1","CH=CH1_1");
Debug.WriteLine(userVar.Value.ToString());

// →表示結果例:VOLTAGE
```

2.4.1.2.2. UR <??>

指定したチャネルの測定レンジの種類を取得する変数です.

表 2-7 に変数作成時の必須オプションを, 表 2-8 には取得値のインデックスとデータ内容について列挙します.

表 2-7 UR 必須オプション

オプション	値範囲	説明
CH	CH1_1 ～ CH4_15	測定レンジの種類を取得するチャネル 例）CH = CH1_1

表 2-8 UR 取得値

説明
指定したチャネルの測定レンジを浮動小数点に変換した値. ※本体機器にて, 1-5V を指定すると 15 の値が設定されます.

使用例

```
var userVar= this.ctrl.AddVariable("UR1","CH=CH1_1");
Debug.WriteLine(userVar.Value.ToString());

// 本体機器設定が 10mV の場合      →表示結果例: 0.01
// 本体機器設定が 20mV の場合      →表示結果例: 0.02
// 本体機器設定が 100mV の場合     →表示結果例: 0.1
// 本体機器設定が 200mV の場合     →表示結果例: 0.2
// 本体機器設定が 1V の場合        →表示結果例: 1
// 本体機器設定が 2V の場合        →表示結果例: 2
// 本体機器設定が 10V の場合       →表示結果例: 10
// 本体機器設定が 20V の場合       →表示結果例: 20
// 本体機器設定が 100V の場合      →表示結果例: 100
// 本体機器設定が 1-5V の場合      →表示結果例: 15
```

2.4.1.2.3. CCH <??>

指定したチャネルのコメント文字列を取得する変数です.

表 2-9 に変数作成時の必須オプションを, 表 2-10 には取得値のインデックスとデータ内容について列挙します.

表 2-9 CCH 必須オプション

オプション	値範囲	説明
CH	CH1_1 ～ CH4_15 PLS1 ～ PLS8 W1 ～ W30	コメント文字列を取得するチャネル 例） CH = PLS1

表 2-10 CCH 取得値

説明
コメント文字の前後をダブルコーテーションで囲んだ形式の文字列 コメント文字(最大全角 20 文字, 半角 40 文字)として扱える半角文字は LR8400 本体で入力できる文字ですが, 特殊文字の入力は下記のように表示されます. ^2 : 2 乗 ^3 : 3 乗 ~c : ° ~e : ε ~u : μ ~o : Ω ^^ : ^ ~~ : ~ ~, : ' ~; : " 上記以外は, スペースに置き換えます.

使用例

```
var userVar= this.ctrl.AddVariable("CCH1","CH=CH1_1");
Debug.WriteLine(userVar.Value.ToString());

// →表示結果例:"test コメント"
```

2.4.1.2.4. MVREAL <??>

指定したチャネルのリアルタイムデータ出力(電圧, 温度, 湿度, 抵抗, 積算, 回転数)の値を電圧値に変換した値を取得する変数です.

表 2-11 に変数作成時の必須オプションを, 表 2-12 には取得値のデータ内容について列挙します.

表 2-11 MVREAL 必須オプション

オプション	値範囲	説明
CH	CH1_1 ～ CH4_15 PLS1 ～ PLS8 LOG ALARM W1 ～ W30	リアルタイムデータ出力するチャネルやパルス番号などを指定する. 指定する値で出力されるデータが異なる. 指定する値と出力するデータの組み合わせは下記 CH1_1 ～ CH4_15 : 電圧, 温度, 湿度, 抵抗のアナログデータ PLS1 ～ PLS8 : 積算, 回転数のパルスデータ LOG : 0～255のディジタルインデータ ALARM : 0～15のアラームアウトデータ W1～W30 : 波形演算の波形演算データ

表 2-12 MVREAL 取得値

説明
必須オプション CH で指定したチャネルやパルス番号などに入力されている値の電圧値.

使用例

```
var userVar= this.ctrl.AddVariable("MVREAL1","CH=CH1_1");
Debug.WriteLine(userVar.Value.ToString());

// →表示結果例: 0.02328
```

2.4.1.2.5. MTVREAL <??>

指定したユニットのリアルタイムデータ出力(測定 ON のチャネルをすべて)の値を電圧値に変換した値を取得する変数です.

表 2-13 に変数作成時の必須オプションを, 表 2-14 には取得値のインデックスとデータ内容について列挙します.

表 2-13 MTVREAL 必須オプション

オプション	値範囲	説明
UNIT	UNIT1 UNIT2 UNIT3 UNIT4 PLS&ALM CALC1 CALC2	リアルタイムデータ出力するユニットを指定 各チャネルによって出力されるデータが異なる. 各チャネルと出力するデータの組み合わせは下記 CH1_1 ～ CH4_15 : 電圧, 温度, 湿度, 抵抗のアナログデータ PLS1 ～ PLS8 : 積算, 回転数のパルスデータ LOG : 0～255のディジタルインデータ ALARM : 0～15のアラームアウトデータ W1 ～ W30 : 波形演算の波形演算データ

表 2-14 MTVREAL 取得値

インデックス[5]	説明
0～14	必須オプション UNIT で指定したチャネルやパルス番号などに入力されている値の電圧値

[5] 測定 ON のチャネルのみ出力されるため, インデックスの範囲は可変です.
例えば, LR8400 の本体にて CH1_2 を測定対象から外した場合に,
オプションにて UNIT1 を指定した際に CH1_1～CH1_15 の CH1_2 が除外されるため, 出力されるインデックスは 0～13 となります.

使用例

```
var userVar= this.ctrl.AddVariable("MTVREAL1","UNIT=UNIT1");
foreach(var item in userVal as Array)
{
    Debug.WriteLine(item.Value.ToString());
}

// →表示結果例:0.066065
                0.023062
                -3276.8
                0.0025615
                0.0028485
                0.0022975
                0.0031365
                0.003715
                0.0020615
                0.001176
                0.0001085
                0.0001085
                -0.0004455
                -0.0008855
                -0.0017835
```

2.4.1.2.6. MTVRCH<??>

指定したユニットの測定 ON のチャネル名を全て取得する変数です.

表 2-15 に変数作成時の必須オプションを, 表 2-16 には取得値のインデックスとデータ内容について列挙します.

表 2-15 MTVRCH 必須オプション

オプション	値範囲	説明
UNIT	UNIT1 UNIT2 UNIT3 UNIT4 PLS&ALM CALC1 CALC2	確認したいユニット

表 2-16 MTVRCH 取得値

インデックス[5]	説明
0 ～ 14	チャネル名

使用例

```
var userVar= this.ctrl .AddVariable("MTVRCH","UNIT=UNIT1");
foreach(var item in userVal as Array)
{
    Debug.WriteLine(item.Value.ToString());
}

    // →表示結果例: CH1_1
                    CH1_2
                    CH1_3
                    CH1_4
                    CH1_5
                    CH1_6
                    CH1_7
                    CH1_8
                    CH1_9
                    CH1_10
                    CH1_11
                    CH1_12
                    CH1_13
                    CH1_14
                    CH1_15
```

3. コマンドリファレンス

3.1. CaoController クラス

表 3-1 CaoController クラス コマンド一覧

コマンド	機能	ページ
汎用コマンド送信		
Send	指定したコマンド文字列を送信します. コマンド文字列に？が含まれている場合は，送信後にデータの受信を行います.	25
実行処理コマンド送信		
Start	測定（波形取り込み動作)の開始をします.	25
Status	ストレージの状態を取得します.	26
Stop	測定（波形取り込み動作）が完了した時点で，測定を停止します.	26
Abort	測定（波形取り込み動作）を強制終了します. 測定が完了しなくても，測定を停止します.	27
Error	本体で発生したエラーまたはワーニングの番号を取得します.	27

3.1.1. CaoController::Execute (″Send″) コマンド

LR8400 に引数に指定したコマンド文字列[1]を送信し, コマンド文字列に？が含まれている場合は, 応答内容の受信も行います.

書式　Send(<command>)

command	:	[in] 送信するコマンド文字列. (VT_BSTR)
戻り値	:	[out] 送信するコマンド文字列に？が含まれていない場合は, なし. (VT_EMPTY) 送信するコマンド文字列に？が含まれている場合は, LR8400 の応答内容. (VT_BSTR) ※送信するコマンド文字列が間違っていた場合などで, 応答内容が取得できない場合は, タイムアウトエラーが発生します.

使用例

```
this.ctrl.Execute("Send", ":START");
var result = this.ctrl.Execute("Send", ":STATUS?");
```

3.1.2. CaoController::Execute (″Start ″) コマンド

LR8400 にスタート処理コマンドを送信し, 測定(波形取り込み動作)の開始をします.

書式　Start()

戻り値	:	なし

使用例

```
this.ctrl.Execute("Start", null);
```

3.1.3. CaoController::Execute (″Status″) コマンド

LR8400 にストレージ状態の問い合わせコマンドを送信し, ストレージの状態を取得します.

書式　Status()

戻り値　：　[out] ストレージの状態を, 数値で返します. (VT_UI1)
応答が 3 場合は,
スタート中, およびストレージ中であることを意味します.
各 bit 値の意味は下記です.

bit0	：	スタート
bit1	：	ストレージ中
bit2	：	トリガ待ち
bit3	：	プリトリガ待ち
bit4	：	未使用
bit5	：	セーブ中

使用例

```
var result = this.ctrl.Execute("Status",null);
```

3.1.4. CaoController::Execute (″Stop ″) コマンド

LR8400 にストップ処理コマンドを送信します. 測定(波形取り込み動作)が完了した時点で, 終了します.
連続記録が OFF の場合, 本コマンドを 1 回実行すると, 記録時間分測定したあと停止します.
連続記録が ON, OFF の場合, 本コマンドを 2 回実行すると即座に測定を停止します.

書式　Stop()

戻り値　：　なし

使用例

```
this.ctrl.Execute("Stop",null);
```

3.1.5. CaoController::Execute (″Abort″) コマンド

LR8400 にアボート処理コマンドを送信し，測定（波形取り込み動作）を強制終了します．測定が完了しなくても，測定を停止します．

書式　Abort()

戻り値　：　なし

使用例

```
this.ctrl.Execute("Abort",null);
```

3.1.6. CaoController::Execute (″Error″) コマンド

LR8400 に本体エラー番号の問い合わせコマンドを送信し，本体で発生したエラーまたはワーニングの番号を取得します．

書式　Error()

戻り値　：　[out] 本体で発生したエラーまたはワーニングの番号を返します．(VT_UI2)
※エラーまたはワーニングについては，本体の取扱説明書を参照してください．

使用例

```
var result = this.ctr.Execute("Error",null);
```

4. サンプルプログラム

以下に各種 Variable 変数を取得後，Controller の Execute コマンド各種を実行する C#のサンプルを示します.

List 4-1

```
…(略)…
using ORiN2.ManagedCAO;

namespace HIOKI_LR8400_Sample
{
    public partial class Sample : Form
    {
        private CCaoEngine eng;
        private CCaoWorkspace ws;
        private CCaoWorkspaces wss;
        private CCaoControllers ctrls;
        private CCaoController ctrl;

        public Sample()
        {
            InitializeComponent();
        }

        private void Sample_Load(object sender, EventArgs e)
        {
            // Caoエンジンの生成
            this.eng = new CCaoEngine();
            this.wss = this.eng.Workspaces;
            this.ws = this.wss[0];
            this.ctrls = this.ws.Controllers;

            // TCP接続の場合
            var option =
            "Conn=TCP:192.168.2.43:8802,Delimiter=1,ConnTimeout=5000,TimeOut=4000";

            // シリアル接続の場合は以下のようにする
            // var option = "Conn=COM:3,ConnTimeout=5000,TimeOut=4000";

            // LR8400への接続
            this.ctrl = this.ws.AddController("Sample",
                                              "CaoProv.HIOKI.LR8400",
                                              ""
                                              ,
                                              option);
        }

        private void btnGetVariable_Click(object sender, EventArgs e)
        {
            // 各種Variableの追加
            // システム変数
            var tempVal1 = this.ctrl.AddVariable("@MAKER_NAME", null);
            var tempVal2 = this.ctrl.AddVariable("@VERSION", null);
            var tempVal3 = this.ctrl.AddVariable("@TITLE_COMMENT", null);
            // ユーザ変数
            var tempVal4 = this.ctrl.AddVariable("UI_1", "CH=CH1_1");
            var tempVal5 = this.ctrl.AddVariable("UR_1", "CH=CH1_1");
            var tempVal6 = this.ctrl.AddVariable("CCH_1", "CH=CH1_1");
            var tempVal7 = this.ctrl.AddVariable("MVREAL_CH1_1", "CH=CH1_1");
            var tempVal8 = this.ctrl.AddVariable("MTVREAL_1", "UNIT=UNIT1");
            var tempVal9 = this.ctrl.AddVariable("MTVRCH_1", "UNIT=UNIT1");
```

```
            // 各種VariableのGetValue
            // システム変数
            var val1 = tempVal1.Value;
            var val2 = tempVal2.Value;
            var val3 = tempVal3.Value;
            // ユーザ変数
            var val4 = tempVal4.Value;;
            var val5 = tempVal5.Value;;
            var val6 = tempVal6.Value;;
            var val7 = tempVal7.Value;;
            var val8 = tempVal8.Value;;
            var val9 = tempVal9.Value;;
        }

        private void btnExecute_Click(object sender, EventArgs e)
        {
            // 各種Executeを実行
            var tempResult1 = this.ctrl.Execute("Send", ":START");
            var tempResult2 = this.ctrl.Execute("Send", ":STATUS?");
            var tempResult3 = this.ctrl.Execute("Start", null);
            var tempResult4 = this.ctrl.Execute("Status", null);
            var tempResult5 = this.ctrl.Execute("Stop", null);
            var tempResult6 = this.ctrl.Execute("Abort", null);
            var tempResult7 = this.ctrl.Execute("Error", null);
        }

        private void Sample_FormClosed(object sender, FormClosedEventArgs e)
        {
            // Cao関連オブジェクトの開放
            if (this.eng != null)
            {
                this.eng.Dispose();
                this.eng = null;
            }
        }
    }
}
```

5. LR8400 のコマンド対応表

変数名	get_Value	set_Value
@TITLE_COMMENT	:COMMent:TITLe?	-
UI<??>	:UNIT:INMOde? ch$	-
UR<??>	:UNIT:RANGe? ch$	-
CCH<??>	:COMMent:CH? ch$	-
MVREAL<??>	:MEMory:VREAl? ch$	-
MTVREAL<??>	:MEMory:TVREAl? unit$	-
MTVRCH<??>	:MEMory:TVRCH? unit$	-

Execute のメソッド名	コマンド名
Start	:STARt
Status	:STATUS?
Stop	:STOP
Abort	:ABORT
Error	:ERRor?