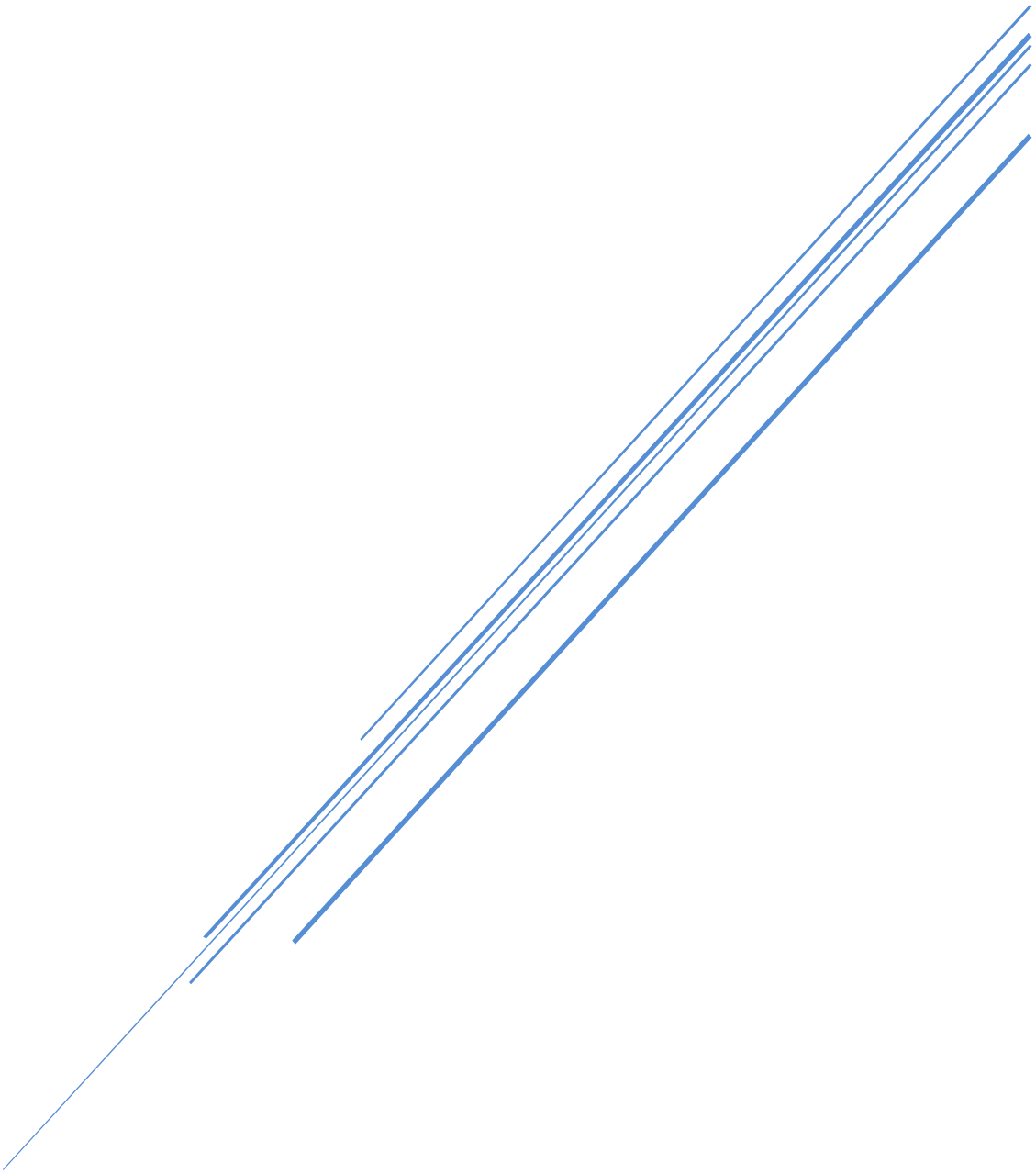
UTD2000CEX&UTD7000B 系列示波器

编程手册



优利德科技（中国）股份有限公司

版本：

版本	修改项
V1.0	正式版
V1.1	增加附录代码中各编码对应的表格定义； 增加典型案例描述

声明：

1、文档中的所有指令都必须通过 uci.dll 提供的 UCI 接口下发。UCI 接口说明见《UCI 帮助文档.pdf》
2、所有字符指令都不区分大小写；
3、直接使用 UCI 接口的设备地址是：
[C:DSO][D:DSO-X][T:USB][PID:0x5537][VID:0x4348][EI:0x82][EO:0x2][CFG:1][I:0]
或使用 UCI 的查询接口查询设备地址。
4、多个相同设备连接，请使用 UCI 查询接口查询唯一的设备地址。

指令集：

IDN?

获取设备名。
数据格式　　：显示名%内部信息#SN 序列号
数据量是　　：　**50**Bytes

比如：
UTG2102CEX%**#SN005

通信协议版本：

查询系统信息版本号，用于检查协议接口是否支持。

命令名	命令参数	命令参数类型
CVer?;	无	无

示例：
“CVer?;”
接口：
使用 uci_Read 读取数据。
数据大小是 50 个字节。
数据：
1,BG, 100M,1GS,2CH

按键

命令名	命令参数	命令参数类型
KEY	键值	String ：缩写的键名

按键名	意义
AT	AUTO
RS	RUN/STOP
TM	*MENU(触发菜单)
FC	FORCE
HP	HELP
VM	*MENU(视窗设置菜单)
STZ	SET TO ZERO
MT	MATH
C1	CH1

C2	CH2
F1	F1
F2	F2
F3	F3
F4	F4
F5	F5
PS	PrScrn(截屏)
MEA	Measure
CSR	Cursor
ACQ	Acquire
DISP	Display
STG	Storage
UTIL	Utility
FKN	Function Knob
FKNL	Function Knob Left Rotation
FKNR	Function Knob Right Rotation
VKNL	Vertical Position Knob Left Rotation
VKNR	Vertical Position Knob Right Rotation
HKNL	Horizontal Position Knob Left Rotation
HKNR	Horizontal Position Knob Right Rotation
TGKNL	Trigger Position Knob Left Rotation
TGKNR	Trigger Position Knob Right Rotation
VBKNL	Voltage Base Knob Left Rotation
VBKNR	Voltage Base Knob Right Rotation
TBKNL	Time Base Knob Left Rotation
TBKNR	Time Base Knob Right Rotation

属性名	意义	IO	数据
Lock	锁定该按键	W	无数据
Unlock	解锁该按键	W	无数据
Lock?	查询该按键锁状态	R	Integer(4B): 0 – 未锁; 1 – 已锁

示例：

“KEY:AT;”

“KEY:C1;”

“KEY:MATH;”

“KEY:FKN;” -> 按下

“KEY:FKNL;” -> 功能旋钮增

“KEY:FKNR;” -> 功能旋钮减

“KEY:FKNL@lock;” -> 功能旋钮增-锁定

注意：

因为不同的设备对同一按键名会有不同，所以，采用缩写以规避这种差异！

键盘全锁和全解使用 local 命令（与 UPO2000cs 一样。）。

带问号的指令需要用 uci_Read 读取。

截图

命令名	命令参数	命令参数类型
PrtScn	图像数据格式	String: bmp {BMP 文件}

示例：

BMP 文件：　　“PrtScn:bmp;”

注意：

“PrtScn:bmp;” 使用 uci_Read 接口，缓冲区大小可设置为 387072，读取到的是 8 位的 BMP 文件 。

读写配置数据

命令名	命令参数	命令参数类型
dconfig	无	无

示例：

“dconfig;”

说明：

如果要保存到文件，请自行保存。 如果超时，请设置较大的超时时间。

接口：

用 uci_Read；读取数据，用 uci_Write 写数据。 读取数据量设置为 1024， 实际大小通过 uci_Read 的返回值得到。

运行状态

命令名	命令参数	数据
Proc	**设置**运行状态	Enum<string> : STOP/RUN/AUTO
Proc?	**查询**运行状态	Enum<string> : STOP/RUN/ARMD/READY/TRIGD/AUTO/SCAN/OVER/RESET { STOP/RUN/ARMED/READY/TRIGED/AUTO/SCAN/OVER/RESET }

示例：

查询：“Proc?;”

设置：“Proc:Stop;”　　“Proc:AUTO;”

注意：

关于“Proc:Stop”、 “Proc:Run” 和 按键“RUN/STOP”的区别：

与按键“RUN/STOP”的功能不同。这里“RUN”就是置 DSO 为 RUN 状态，无论目前处于何种状态。“STOP”类似。

通道控制

命令名	意义	IO	数据	备注
CHSel?	查询当前选中的通道	R	通道号分别是：0/1/2/3/4 {CH1/ CH2/ MATH/ REF-A/ REF-B }	设置通道选中请使用：“CH:0@SEL;”

基本（共有）属性：

命令名	命令参数	命令参数类型	备注
CH	通道号	以 0 开始的通道编码，整数值。0:CH1;1:CH2	**通道号**指明后续的指令针对的是哪个通道！

属性名	意义	IO	数据
EN	ON/OFF	WR	Enum(Integer): 0/1{ON/OFF}
SEL	Select(设置或者查询当前通道的选中状态)	WR	W：无{命令类型} R：Enum(Integer): 0/1{ON/OFF}
VP	通道垂直位置	WR	Integer：位置值，屏幕中间为 128，上增下减，一格为 25.
HP	预触发位置（水平调节）	WR	Integer：位置值，屏幕中间为 350，左减右增，一格为 50.
TB	Time Base（时基挡）	WR	W: Enum(String) : +/- {加一档/减一档}或电流值。 R : Double(8Bytes)，以 us 为单位的电流值。
VB	Voltage Base（幅格挡）	WR	W: Enum(String) : +/- {加一档/减一档}或电压值。 R : Double(8Bytes)，以 V 为单位的电源值。
STZ	VPOS and HPos Set to zero	W	无{命令类型}

支持的伏格挡挡位：

1mV	10mV	100mV	1V	10V
2mV	20mV	200mV	2V	20V
5mV	50mV	500mV	5V	--

--	10ns	100ns	1us	10us	100us	1ms	10ms	100ms	1s	10s
2ns	20ns	200ns	2us	20us	200us	2ms	20ms	200ms	2s	20s
5ns	50ns	500ns	5us	50us	500us	5ms	50ms	500ms	5s	50s

伏格档和实际档位命令直接写数值和单位即可，比如 100mv：“CH:0@VB:100MV”，“CH:0@TB:500US;”，单位不区分大小写。

示例：

通道号分别是：0/1/2/3/4{CH1/ CH2/ MATH/ REF-A/ REF-B }

设置：

“CH:0@EN:1@VP:128@HP:350@VB:100MV@TB:500US;”

“CH:2@EN:1;” --- 打开 MATH 通道

“CH:2@VP:128;” --- 调节 MATH 通道的垂直位置为 128

“CH:2@SEL;” --- 选中 MATH 通道

查询：

“CH:0@VP;”

“CH:0@TB;”

“CH:0@TBV;”

注意：

1、 关于基础命令和物理通道的命令协议：
基础命令和物理通道的命令名都使用“CH”，格式为“CH: channel id@Attribute:Value;”。基础命令是适用于物理通道、MATH、REF 的，通过 channel id 进行区分；

2、 CH 命令必须跟通道号作为命令参数，否则为错误的命令格式；

3、 执行“@SEL;”指令时，如果将要选中的通道未打开，则会返回 “通道未打开” 的错误；

4、 如果是物理通道（CH1/ CH2），“@VB” 在细调时（通过“@ VD”设置）只能通过“Enum(String) : +/- {加一档/减一档}”的方式设置，然后通过“@VB”指令获取实际的伏格挡信息。

物理通道：

命令名	命令参数	命令参数类型	备注
CH	通道号	以 0 开始的通道编码，整数值。	**通道号**指明后续的指令针对的是哪个通道！

属性名	意义	IO	数据
CP	Coupling(通道耦合)	WR	Enum(String): D/A/G{DC/AC/GND}
BW	BW limit(带宽限制)	WR	Enum(String): Enum(Integer): 0/1{ON/OFF}
VD	Volts/Div(幅度粗调和细调)	WR	Enum(String): C/F{Coarse/Fine}
Probe	Probe（探头倍率）	WR	Enum(Decimals): 1/10/100/1000
Invert	Invert（反相）	WR	Enum(Integer): 0/1{ON/OFF}

示例：

“CH:0@CP:D@BW:0@VD:C@Probe:1@invert:0;”

注意：

1、“CH:0;” 这条是无效的命令，通道选中应该使用“CH:0@SEL;”

2、属性读取时，缓冲区至少 10 个字节，返回的是字符串，字符串的编码格式由编译器决定，即如果是 UNICODE，则返回的就是 UNCODE 编码的字符串。

频率计：

命令名	命令参数	命令参数类型	备注
cmeter	--	--	--

属性名	意义	IO	数据
EN	Enable(打开/关闭)	WR	Enum(Integer<2Bytes>): 0/1
Freq?	频率值	R	Double<8Bytes>， 以 Hz 为单位。

示例：

“cmeter@en:1;”

“cmeter@freq?;” -- 读取频率值

注意：

1、 频率计测量的是与触发源对应的通道频率。

2、 频率计是硬件测量，精度高于参数测量中的频率参数测量。

参数测量：

命令名	命令参数	命令参数类型	备注
Mea	见下表	字符串	

命令参数	意义	IO	数据
all	打包读取测量参数	R	数据见 MeasureParamPacket_UTD2000CEX，也可以使用下面的方式，单独读取测量值。
All?	打包读取测量参数	R	读取的是所有示波器统一的数据结构。数据见参数测量统一数据包：
freq	Freq	R	Double<8Bytes>
period	Period	R	Double<8Bytes>
rtime	Rise	R	Double<8Bytes>
ftime	Fall	R	Double<8Bytes>
pwidth	+Width	R	Double<8Bytes>
nwidth	-Width	R	Double<8Bytes>
oshoot	Overshoot	R	Double<8Bytes>
pshoot	Preshoot	R	Double<8Bytes>
pduty	+Duty	R	Double<8Bytes>
nduty	-Duty	R	Double<8Bytes>
avg	Average	R	Double<8Bytes>
vpp	Peak	R	Double<8Bytes>
rms	RMS	R	Double<8Bytes>
high	High	R	Double<8Bytes>
low	Low	R	Double<8Bytes>

mid	Middle	R	Double<8Bytes>
max	Max	R	Double<8Bytes>
min	Min	R	Double<8Bytes>
amp	Amplitude	R	Double<8Bytes>

属性名	意义	IO	数据
src	测量源	WR	Enum(Integer<2Bytes>): 0/1 {CH1/CH2}

示例：

“mea:freq;” --- 读取频率值；

“mea:all;” -- 打包读取所有参数值.

“mea:all?;” -- 打包读取所有参数值（通用）.

“mea@src:0;” --- 设置测量源为 CH1；

代码示例：

读取单独参数：

```
double dv = 0.0;
r = uci_ReadX(m_session, _T("mea:freq"), 1000, (byte*)&dv, sizeof(dv));
if (UCISUCCESS(r)) {
    printf("Freq = %f", dv);
}
```

打包读取所有参数：

```
namespace cb = comAPICommon;//comAPICommon alias

    void Test_MeasureParams_DSO() {
        comAPICommon::MeaValue params[50];

        auto r = uci_ReadX(m_session, _T("mea:all?;"), 2000, (byte*)params, sizeof(params));
        ASSERT(r >= 0);

        PrintMeasureParams(params);
    };

    void PrintMeasureParams(comAPICommon::MeaValue* _p) {
        printf("\n++++++++++++++++++++++++++++++++++++++++++++++++++\n");
        PrintMeaParam(_p[cb::MP_FREQ], "freq");
        PrintMeaParam(_p[cb::MP_PERIOD], "period");

        PrintMeaParam(_p[cb::MP_NDUTY], "NDUTY");
        PrintMeaParam(_p[cb::MP_PDUTY], "PDUTY");
```

```
        PrintMeaParam(_p[cb::MP_MAX], "MAX");
        PrintMeaParam(_p[cb::MP_MIN], "MIN");

        PrintMeaParam(_p[cb::MP_PKPK], "VPP");
        PrintMeaParam(_p[cb::MP_RMS], "RMS");

        PrintMeaParam(_p[cb::MP_OVERSHOOT], "OVERSHOOT");
        PrintMeaParam(_p[cb::MP_PRESHOOT], "PRESHOOT");

        PrintMeaParam(_p[cb::MP_AMP], "AMP");

        PrintMeaParam(_p[cb::MP_RISE_TIME], "RISE_TIME");
        PrintMeaParam(_p[cb::MP_FALL_TIME], "FALL_TIME");
        printf("\n----------------------------------------------------");
    }

void PrintMeaParam(const comAPICommon::MeaValue& _p, const char * _name) {
    printf("%s = ", _name);
    if (_p.IsExist) {
        if (_p.IsValid) {
            printf("%f ", _p.Value);
        } else {
            printf("--");
        }
    } else {
        printf("NotExit");
    }

    printf(" %s%s\n", unit::uci_UnitFindScaleName(_p.Unit.Scale),
        unit::uci_UnitFindTypeName(_p.Unit.Type));
}
```

关于 MeaValue 的结构和描述请参看：参数测量统一数据包

抓取波形数据

命令名	命令参数	命令参数类型
Capture wave	波形格式	String:.bin /.csv/.sav {二进制数据/CSV 文件/仪器内置波形文件}

属性名	意义	IO	数据
CH	Channel(导出的通道 ID)	W	Enum(Integer) : 0/1/2/3{ CH1/ CH2/CH3/CH4}
DT	Data Type(数据格式)	W	Enum(String) : ad/vol{ ADC 原始数据/电压值}

示例：

“capture wave:.bin@CH:0@DT:AD;” -- 读取 CH1 的波形数据，AD 值，存放在内存缓冲区中。

“capture wave:.bin@CH:0@DT:vol;” -- 读取 CH1 的波形数据，电压值，存放在内存缓冲区中。

“capture wave:.csv@CH:0@DT:vol;” -- 读取 CH1 的波形数据，电压值，存放在 CSV 文件中。

说明：

1、 “.bin”和”.csv“文件必须包含：@CH:0”和“@DT:”属性，而”.sav”文件必须包含@CH:0 属性；

2、 这里的波形数据是原始数据，不是显示数据（只有几百个点），可以用于波形分析任务；

3、 可以使用读参数接口{uci_Read 或者 uci_ReadX}也可以使用读文件接口{uci_ReadToFile 或者 uci_ReadToFileX}，前者将数据存放在内存缓冲区中，后者将数据存放在硬盘文件中。

4、 波形数据格式：

a） AD 值 ： 是以 16 位 short 类型，为一个波形点的波形数据；

b） VOL 值，即电压值，是以通道基线为零点的电压值。

触发系统：

命令名	命令参数	命令参数类型	备注
trig	--	--	--

属性名	意义	IO	数据
t	类型	W	Enum(String) : E/V /P{ EDGE/VIDEO/PLUSE WIDTH }
src	触发源	W	Enum(String) : c1/c2 /ext/ac/alt{ CH1/CH2/EXT/AC LINE/ALTER }
mode	触发模式	W	Enum(String) : A/ N /S{ AUTO/NORMAL/SINGLE }
cp	触发耦合	W	Enum(String) : D/A/H/L { DCl / AC /H Restrain /L H Restrain }
pos	触发位置	W	Ingeger<2Bytes> ： 以通道基线为-点，上正下负，每格 25；
st	边沿触发斜率类型	W	Enum(String) : F / R /A{ Fall / Rise/Rise and Fall }

示例：

“trig@t:e;” -- 设置触发类型为边沿触发；

“trig@pos:25;” -- 设置触发电平为以基线为准向上一格，如果伏格挡位是 1V，那么触发电压就是 1V；

【基础】 写参数

该指令中的参数地址都是数字编码，需要查看 comApiDef.h 文件中的命令编码定义。
该指令是为了兼容以前老的协议而存在。

命令名	命令参数	命令参数类型	备注
wp	--	--	--

属性名	意义	IO	数据
CH	通道号	W	Enum(Integer<2Bytes>): 0/1 {CH1/CH2}
addr	参数地址（命令编号）	W	见中 comApiDef.h 中的 CONTROL_CMD 定义

示例：

“WP@CH:0@ADDR:950:;” -- 设置 DISPLAY 参数。

注意：

“@CH”和“@addr”必须同时使用。

【基础】 读参数

该指令中的参数地址都是数字编码，需要查看 comApiDef.h 文件中的命令编码定义。
该指令是为了兼容以前老的协议而存在。

命令名	命令参数	命令参数类型	备注
rp	--	--	--

属性名	意义	IO	数据
CH	通道号	R	Enum(Integer<2Bytes>): 0/1 {CH1/CH2}
addr	参数地址（命令编号）	R	见中 comApiDef.h 中的 CONTROL_CMD 定义

示例：

“RP@CH:0@ADDR:951:;” -- 读取 DISPLAY 参数。

注意：

“@CH”和“@addr”必须同时使用。

案例

1、测量信号频率和幅度等参数

方法一： 使用参数测量功能(Measure)测量信号的各种参数，包括频率、幅度、周期等参数，具体请参考仪器 Measure 功能。

参数测量

方法二：使用 UTILITY->频率计 功能测量信源频率：

频率计

区别：

1、 参数测量(MEASURE)是软件测量，频率计是硬件测量，频率计测量精度较高；

2、 频率计测量的信号源是触发源对应的信号，参数测量测量的信号源是只有在参数测量中可以设置的源；

2、触发通道触发后，抓取待测通道波形数据，做深度分析

比如：CH1 作为信号测量通道，连接测量源。CH2 作为触发通道，连接触发信号，这个触发信号通常由用户自定义生成，目的是找到触发时机，捕捉对应时间的 CH1 的波形区块，提取数据，然后做数据分析。

通常的模型是：

1、 设置触发方式为单次触发，指令：“trig@mode:s;”(使用写参数接口);

2、 设置运行状态为 RUN，指令：“proc:run;” (使用写参数接口);

3、 获取运行状态，检查运行状态是否为 STOP，为 STOP 则已经被触发，指令：“proc？”(使用读参数接口)

4、 如果已经被触发，则提取波形数据，指令：“capture wave:.bin@CH:0@DT:vol;”

循环 2-4 步骤，完成本模型。

附录：

参数测量数据包：

```
typedef struct _measure_param {
    float value;
    int unit;
}MeasureParam;

typedef struct
{
    MeasureParam freq;
    MeasureParam period;
    MeasureParam risetime;
    MeasureParam falltime;
    MeasureParam pwidth;
    MeasureParam nwidth;
    MeasureParam overshoot;
    MeasureParam preshoot;
    MeasureParam pduty;
    MeasureParam nduty;
    MeasureParam vmean;
    MeasureParam vpp;
    MeasureParam vrms;
    MeasureParam vtop;
    MeasureParam vbase;
    MeasureParam vmid;
    MeasureParam vmax;
    MeasureParam vmin;
    MeasureParam vamp;

    //
    char        Reserve[24];
}MeasureParamPacket_UTD2000CEX;
```

参数测量统一数据包：

代码定义：

数据包为 MeaValue mp[50] （400Bytes），即固定50个测量参数的一维序列。各参数在序列中的位置由EMeaParam定义。

```
struct UnitParam {
    char Type;      //单位类型，比如Time、Freq等，由EType定义。
    char Scale;     //量级，比如k、n，p，M等，由EScale定义。
};

//@brief : 物理量数值
//@remark: 4Byte align -> 8Bytes
struct MeaValue {
    float      Value;
    UnitParam  Unit;
    char       IsValid;   //是否有效。  0 表示无效；  1表示有效。
    char       IsExist;   //是否存在。  0 表示存在；  1表示不存在。
};
```

内存表定义：

每个参数占用 8 个字节，一共 50 个参数空间，共 400 个字节，每个参数的内存模型是：

字段	Value	Unit.Type	Unit.Scale	IsValid	IsExist
占用	4Bytes	1Byte	1Byte	1Byte	1Byte
意义	物理值（小数）	单位类型编码	单位量级编码	是否有效	是否存在
数据		取值	取值	0:无效;1:有效	0:不存在；1:存在

物理单位编码

代码定义：

```
enum EScale : char {
    SCALE_p = -4,
    SCALE_n,
    SCALE_μ,
    SCALE_m,
```

```
        SCALE_STD = 0,
        SCALE_K,
        SCALE_M,
        SCALE_G,
        SCALE_T,
    };

    enum EType : char {
        TYPE_INVALID = -1,
        TYPE_FREQ,
        TYPE_TIME,
        TYPE_AREA,     //面积(Vs)
        TYPE_SAMPLERATE,//采样率（Sa/s）
        TYPE_POINT,    //点数（Sa）
        TYPE_VPP,      //峰峰值
        TYPE_VOLTAGE,//电压
        TYPE_CURRENT,//电流
        TYPE_DB,       //DB
        TYPE_VV,       //
        TYPE_PERCENT,//百分比
        TYPE_DEGREE, //度数
        TYPE_WATT,     //瓦特，功率
        TYPE_UNKNOWN,// 未知单位
    };
```

编码表：

单位量级编码：

单位	编码
p	-4
n	-3
μ	-2
m	-1
标准单位	0
K	1
M	2
G	3
T	4

单位类型编码：

单位	编码
无效的类型	-1
频率（Hz）	0
时间（s）	1
面积（Vs）	2
采样率（Sa/s）	3
点数（Sa , pkts）	4
峰峰值（VPP）	5
电压（V）	6
电流（A）	7
功率（DB）	8
VV	9
百分比（%）	10
度数（° ）	11
瓦特（W）	12
未知单位（U）	13

参数编码

代码定义：

```
//@brief : 参数测量数据包的通用定义。
//@remark:
enum EMeaParam {
    MP_MAX = 0, //最大值
    MP_MIN,     //最小值
    MP_HIGH,    //High(Top)-高电平(顶端值)
    MP_MIDDLE,  //Middle-中间值
    MP_LOW,     //Low(Bottom) - 低电平(底端值)

    MP_PKPK,    //VPP-峰峰值
    MP_AMP,     //幅度
    MP_MEAN,    //平均值
    MP_CYCMEAN, //
    MP_RMS,     //均方根

    MP_CYCRMS,   //周期均方值
```

```
    MP_AREA,     //面积
    MP_CYCAREA,  //周期面积
    MP_OVERSHOOT,//过冲
    MP_PRESHOOT, //预冲

    MP_PERIOD,   //周期
    MP_FREQ,     //频率
    MP_RISE_TIME,//上升时间
    MP_FALL_TIME,//下降时间
    MP_PWIDTH,   //正脉宽

    MP_NWIDTH,   //负脉宽
    MP_PDUTY,    //正占空比
    MP_NDUTY,    //负占空比
    MP_RISEDELAY,//上升延时
    MP_FALLDELAY,//下升延时

    MP_PHASE,    //相位
    MP_FRR,      //
    MP_FRF,
    MP_FFR,
    MP_FFF,

    MP_LRF,
    MP_LRR,
    MP_LFR,
    MP_LFF,

    MP_BURST_WIDTH, //突发脉冲

    //
    //reserve section
    //
    //固定位50个参数
    MP_MAX_COUNT = 50,
};
```

编码表：

参数	编码
最大值（MAX）	0
最小值（MIN）	1
高电平/顶端值(High/Top)	2

中间值（Middle）	3
低电平/底端值（Low）	4
峰峰值（PKPK）	5
幅度（AMP）	6
平均值（MEAN）	7
周期平均（Cycmean）	8
均方根（RMS）	9
周期均方根（Cycrms）	10
面积（AREA）	11
周期面积（Cycarea）	12
过冲（Overshoot）	13
预冲（Preshoot）	14
周期（Period）	15
频率（Freq）	16
上升时间（Rise Time）	17
下降时间（Fall Time）	18
正脉宽（PWidth）	19
负脉宽（NWidth）	20
正占空比（PDuty）	21
负占空比（NDuty）	22
上升延时（Rise Delay）	23
下降延时（Fall Delay）	24
相位（Phase）	25
FRR	26
FRF	27
FFR	28
FFF	29
LRF	30
LRR	31
LFR	32
LFF	33
突发脉冲（Burst Width）	34
预留	35~39

注意：

1、 本机型只支持上述表格中的部分参数；

2、 C#接口中的定义名称有所不同，放在 `ucics.unit` 和 `ucics.mea` 两个命名空间中。

数据单位表：

单位	编码
空	0
ps	1
ns	2
μs	3
ms	4
ks	5
nVs	7
μVs	8
mVs	9
μV	11
mV	12
V	13
kV	14
pHz	18
nHz	19
μHz	20
mHz	21
Hz	22
KHz	23
MHz	24
GHz	25
mVV	52
VV	53
KVV	54
mDB	80
DB	81
KDB	82