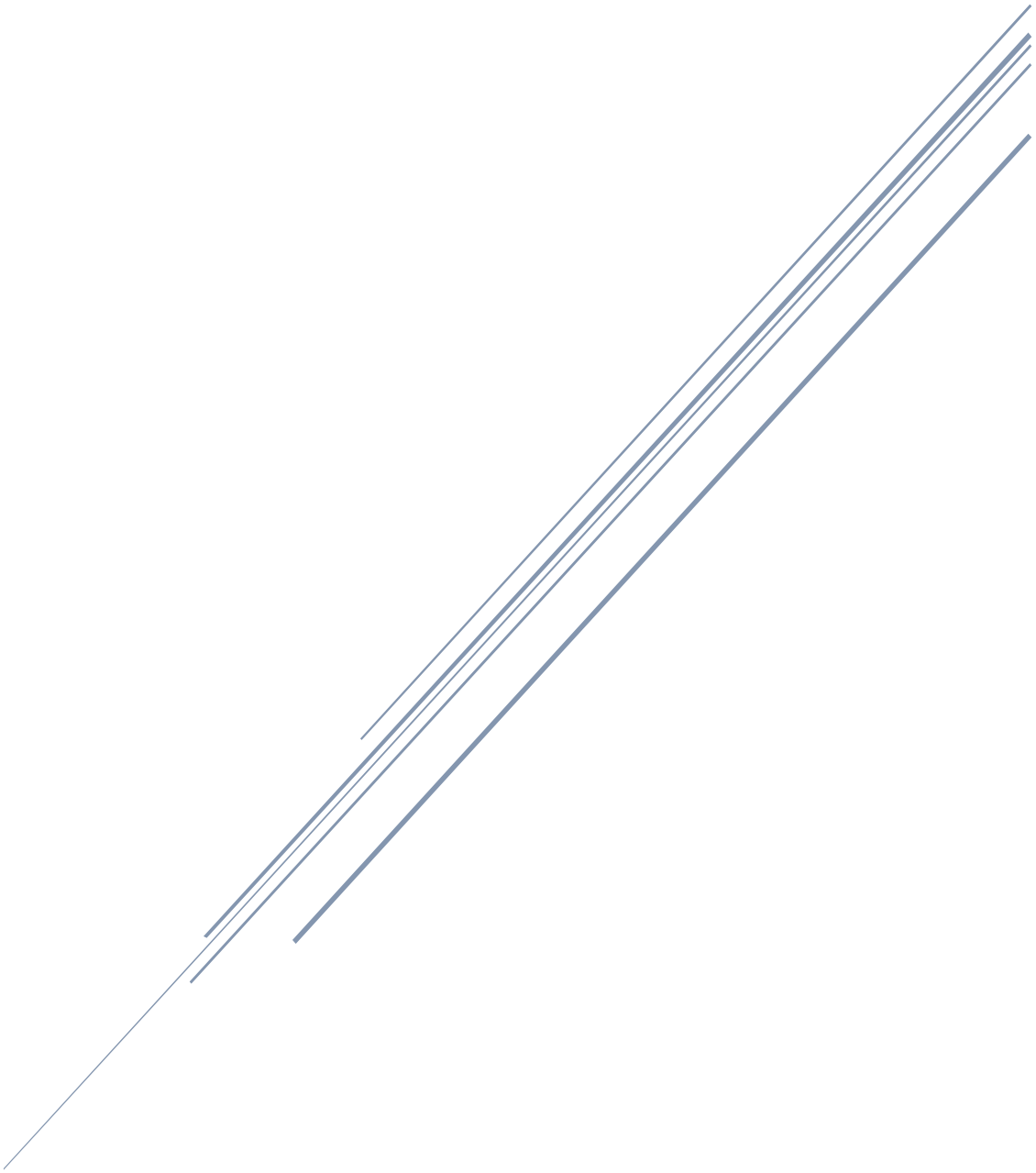
UTG2000A&UTG7000B Series Signal Source

# Programming Manual



## Version：

| Version | Modification Item |
| --- | --- |
| V1.0 | Official version |

## Introduction：

UCI interface, related problems see UCI Help Documentation. Detailed operating see example project.

## Reference File：

1、UTG2025Def.h ： the basic definition of this series
2、UCI relevant documentation： see UCI Help Documentation

## Basic Format of Command String

The name of command string 1: command parameter@ attribute 1: attribute value@ attribute 2: attribute value … @ attribute n: attribute value;
The name of command string 2: command parameter@ attribute 1: attribute value@ attribute 2: attribute value … @ attribute n: attribute value;
The name of command string n: command parameter@ attribute 1: attribute value@ attribute 2: attribute value … @ attribute n: attribute value;

### Description：

1、It is not case-sensitive;
2、Numerical value support the format of hexadecimal, octal and decimalism;
3、Support multiple statements (depends on model), if multiple statements and attributes on failure, try to using single statement and attribute;
4、Every statement must end with ';'；
5、Spaces are supported between names, values, and tags;

### Example：

“wp@ch:0@addr:10@v:10;” “key: c1;”

### Term： SG - short name of Signal Source

## General Command

| Name of command | Meaning | IO | Data | Note |
|---|---|---|---|---|
| Local | Lock Pad | W | Enum:0/1{remote/local status} | Keyboard locked in remote status |
| Local? | Query keyboard is locked or not | R | Enum:0/1{unlocked/locked} | |
| Lock? | Query the lock status of keyboard | R | 8Bytes，64 signed integer, flag bit | |

## Write Parameter

| Name of command | Command parameter | Type of command parameter |
|---|---|---|
| wp | None | None |

| Name of attribute | Meaning | IO | Data |
|---|---|---|---|
| CH | Number of channel | W | Enum(Integer) : 0/1{ CH1/ CH2 } |
| addr | Parameter address | W | Enum(ParamNo): view the definition of parameter address |
| v | Parameter value | W | Value usually measured by the smallest unit |

## Example :

"wp@CH:0@addr:9@v:1000;" - set the frequency of CH1 is 1mHz;

## Note :

UCI corresponding interface: uci_Write

## Read Parameter

| Name of command | Command parameter | Type of command parameter |
|---|---|---|
| rp | None | None |

| Name of attribute | Meaning | IO | Data |
|---|---|---|---|
| CH | Number of channel | W | Enum(Integer) : 0/1{ CH1/ CH2 } |
| addr | Parameter address | W | Enum(ParamNo): view the definition of parameter address |

### Example：

“rp@CH:0@addr:9;” - read the frequency of CH1;

### Note：

UCI corresponding interface: uci_Read, corresponding data size are 8 bytes, double types!

## Key

| Name of command | Command parameter | Type of command parameter |
|---|---|---|
| KEY | Key value | See key code as below |

| KEY | Character encoding | KEY | Character encoding |
|---|---|---|---|
| Bottom function key 1 | AF1 | 0 | 0 |
| Bottom function key 2 | AF2 | 1 | 1 |
| Bottom function key 3 | AF3 | 2 | 2 |
| Bottom function key 4 | AF4 | 3 | 3 |
| Bottom function key 5 | AF5 | 4 | 4 |

| | | | |
|---|---|---|---|
| Bottom function key 6 | AF6 | 5 | 5 |
| F1 | F1 | 6 | 6 |
| F2 | F2 | 7 | 7 |
| F3 | F3 | 8 | 8 |
| F4 | F4 | 9 | 9 |
| Menu | MENU | . | . |
| Knob Left | FKNL | +/- | SIGN |
| Knob Right | FKNR | Trigger | TG |
| Knob Click | FKN | Utility | UTIL |
| Left | L | CH1 | C1 |
| Right | R | CH2 | C2 |

| Name of attribute | Meaning | IO | Data |
|---|---|---|---|
| Lock | Lock the keyboard | W | No data |
| Unlock | Unlock the keyboard | W | No data |
| Lock? | Query the lock status of keyboard | R | Integer<4Bytes>： 0 – unlocked； 1 –locked |

## Example：

“KEY:c1;” -- CH1

“KEY:c2;” -- CH2

“KEY:c2@lock;” -- CH2 locked the key

“KEY:c2@unlock;” -- CH2 unlocked the key

“KEY:c2@lock?;” -- Query keyboard locked or not

## Note：

Command with question mark read in uci_Read . The status gets from the interface return value.

### Read and Write Configuration File

| Name of command | Command parameter | Type of command parameter |
|---|---|---|
| dconfig | None | None |

**Example：**

“dconfig;”

**Note：**

Use the interface uci_Read to read，buffer area size can set as 1024Bytes，the actual of effective data is defined by the interface return value. Use the interface *uci_WriteFromFile* to write configuration data, which does not recognize the file suffix，it can only recognize command "dconfig;"，write configuration timeout at least 6s.

## Capture Screen

| Name of command | Command parameter | Type of command parameter |
|---|---|---|
| PrtScn | Picture format | Enum(String):null/zip/bmp<br>{unpacked pixel data/packed pixel data /BMP file data |

**Example:**

“PrtScn:bmp;” --- screenshot saved as bmp file data；

“PrtScn;” --- screenshot saved as pixel file data；

“PrtScn:zip;” --- screenshot saved as packed pixel file data；

**Note：**

Use uci_Read to read data，the command has no saved data file, it should return to the specified buffer area of uci_Read. If you want to buffer a local file, please save it by yourself.

1、 If use command： “PrtScn;”，buffer area size must be >=391680(480* 272* 3)，readout is 24bits of pixel data;

2、 If use command：“PrtScn:bmp;”，buffer area size must be 480* 272* 3 + 54 = 391734,that is the size of picture.

3、 If use command： “PrtScn:zip;”， buffer area size can set >=391680 (480* 272* 3 ) （maximum data volume），readout is packed pixel data. And then use the interface： alg_UnCompressPixels_25 to unzipping data.
Note: uci_Read return value is packed data volume.

4、Use the interface uci_ReadToFile to add the command “prtscn:bmp;”, it can save bitmap to disk file.

## Write Random Wave File

| Name of command | Command parameter | Type of command parameter |
|---|---|---|
| WARB | None | None |

| Name of attribute | Meaning | IO | Data |
|---|---|---|---|
| CH | Number of channel | W | Enum(Integer) : 0/1{ CH1/ CH2 } |
| Mode | Loading mode | W | Enum(Integer): 0/1 {Carrier/Mod} |

**Example：**

“WARB@CH:0@MODE:0;”
Loading wave file as carrier wave form into to CH1

**Note：**

use the interface uci_WriteFromFileto to write random wave file, timeout set to 1000.

# Appendix

## 1、Parameter address

Numerical value unit：

Frequency unit： μHz， 1mHz° set value 1*1000；
Voltage unit： μV
Time unit：ns（ns、μs、ms、s）, for example 1s return value is 1*1000^3
Degree unit：0.01°，phase set to 90°，return value is 90 * 100
Percent unit 0.01%，for example duty cycle, return value= input value *100

```
//@brief : UTG2025A and UTG2062A signal source parameter coding
typedef enum _ParamNo
    { Invalid_Cmd = -1,
    //@brief : the current working mode
    CurMode = 6,
    //@brief : random wave play mode
    ArbPlayMode = 7,
    //@brief : waveform types
    WaveType = 8,
    //@brief : frequency unit: μHz°, 1mHz°set to 1*1000
    Freq = 9,
    //@brief : period
    Period = 10,
    //@brief : amplitude-peak value, set value unit is mV
    AmpVpp = 11,
    //@brief : amplitude –effective value unit: μV
    AmpVrms = 12,
    //@brief : amplitude -power
    AmpDbm = 13,
    //@brief : DC offset
    DCOffset = 14,
    //@brief : high level
    High = 15,
    //@brief : low level
    Low = 16,
    //@brief : phase, set 0.01 as basic value, like 1° set to 100
    Phase = 17,
    //@brief : duty cycle unit: 0.01%, 1% set to 1*100
    DutyCycle = 18,
    //@brief : symmetry unit: 0.01°, 1°set to 1*100
```

```
 Symmetry = 19,
 //@brief :
 PulseWidth = 20,
 //@brief : impulse wave - rising time
 PulseRisingTime = 21,
 //@brief : impulse wave - falling time
 PulseFallingTime = 22,
 //@brief :
 ArbSampleRate = 23,
 //@brief : ARB file name
 ArbName = 24,
 //@brief :
 ArbLength = 25,
//@brief : modulation types
 ModulateType = 40,
//@brief : modulation frequency
 ModulateFreq = 28,
//@brief : modulation source
 ModulateSource = 29,
//@brief : modulation wave
 ModulateShape = 30,
 //@brief :
 ModulateOnOff = 31,
//@brief : modulation depth
 ModulateAMDeepth = 32,
//@brief : frequency offset
 ModulateFMFreqDev = 33,
//@brief : phase offset
 ModulatePMPhaseDev = 34,
//@brief : hopping frequency
 ModulateFSK_HopFreq = 35,
//@brief : speed rate
 ModulateFSKRate = 36,
 //@brief : phase shift keying - phase
 ModulatePSKPhase = 37,
 //@brief : frequency-shift keying - carrier frequency
 ModulateFSK_CarrierFreq = 38,
 //@brief : pulse width modulation-duty cycle skew
 ModulatePWM_DutyDev = 39,
 //@brief : sweep frequency - start frequency
 SweepStartFreq = 47,
 //@brief : sweep frequency-stop frequency
 SweepStopFreq = 48,
```

```
 //@brief : sweep frequency - sweep time
 SweepTime = 49,
 //@brief : sweep frequency - sweep type
 SweepType = 50,
 //@brief : switch frequency setting - frequency or period
 Wave_Freq_Or_Period = 51,
 //@brief : switch amplitude setting - amplitude or high/low level
 Wave_Amp_Or_HighLow = 52,
 //@brief : switch different amplitude unit
 Wave_Amp_Unit = 53,

 //@brief : pulse string – start phase
 BurstStartPhase = 57,
 //@brief : pulse string - burst cycle
 BurstPeriod = 58,
 //@brief : pulse string – cycle count
 BurstCycleCount = 59,
 //@brief : pulse string - gating-polarity
 BurstPolarity = 60,
 //@brief : pulse string – burst type
 BurstType = 61,

 //@brief : sweep frequency - trigger
 TriggerSource = 68,
 SyncType = 69,
 //@brief : sweep frequency – trigger edge 0: rising edge 1:falling edge
 TriggerEdge = 70,
 //@brief : sweep frequency-trigger output switch
 TriggerOnOff = 71,

 //@brief : CH1 channel function
 OutputEnable = 77,

 //@brief : CH2 channel function
 OutputEnable2 = 78,
 //@brief : inverted
 Invert = 79,
//@brief : output resistance
 OutputExtRes = 80,
//@brief : amplitude limit
 OutputLimit = 81,
//@brief : amplitude upper limit
 OutputLimitHigh = 82,
//@brief : amplitude lower limit
```

```
    OutputLimitLow = 83,
    //@brief : sync output
    SyncOut = 84,
    //@brief :
    Couple = 85,
    //@brief : IP address
    type
    IPAddrType = 101,
    //@brief : IP address
    IPAddr = 102,
    //@brief : subnet mask
    IPSubnetMask = 103,
    //@brief : gateway
    IPGateway = 104,
    //@brief : DNS
    IPDns = 105,
    //@brief : MAC address
    Mac = 106,
    //@brief : language
    Language = 107,
    //@brief : clock source
    ClkSource = 108,
    //@brief :
    PowerOnParam = 109,
    //@brief : buzzer switch
    BeepOnOff = 110,
    //@brief : numeric separator
    CharDivideInt = 111,
    //@brief : load setting
    LoadSettting = 112,
    //@brief : save setting
    SaveSetting = 113,
   //@brief : adjust backlight
    Backlight = 114,
    //@brief : about
    About = 115,
    //@brief : clock output
    ClkOut = 116
}ParamNo
```

## 2、Keyboard Locked Status Mark Bits

```
//@brief : query UTG2025A and UTG2062A keyboard in locked status, save it in 64bits
integer register
typedef enum _UTG2025Key_bit {
    IKEY_Unknown = -1,
    //@brief   : bottom functional key 1
    IKEY_AF1 = 0,
    //@brief   : bottom functional key 2
    IKEY_AF2,
    //@brief   : bottom functional key 3
    IKEY_AF3,
    //@brief   : bottom functional key 4
    IKEY_AF4,
    //@brief   : bottom functional key 5
    IKEY_AF5,
    //@brief   : bottom functional key 6
    IKEY_AF6,
    //@brief   : functional key 1
    IKEY_F1,
    //@brief   : functional key 2
    IKEY_F2,
    //@brief   : functional key 3
    IKEY_F3,
    //@brief   : functional key 4
    IKEY_F4,
    //@brief   : Menu key
    IKEY_MENU,
    //@brief   : numeric keypad 0
    IKEY_NUM_0 = 14,
    //@brief   : numeric keypad 1
    IKEY_NUM_1,
    //@brief   : numeric keypad 2
    IKEY_NUM_2,
    //@brief   : numeric keypad 3
    IKEY_NUM_3,
    //@brief   : numeric keypad 4
    IKEY_NUM_4,
    //@brief   : numeric keypad 5
    IKEY_NUM_5,
    //@brief   : numeric keypad 6
```

```
    IKEY_NUM_6,
    //@brief   : numeric keypad 7
    IKEY_NUM_7,
    //@brief   : numeric keypad 8
    IKEY_NUM_8,
    //@brief   : numeric keypad 9
    IKEY_NUM_9,
    //@brief   : numeric keypad - symbol（+/-）key
    IKEY_SIGN,
    //@brief   : numeric keypad “.” key
    IKEY_DOT,
    //@brief   : the right knob
    IKEY_PLUS,
    //@brief   : the left knob
    IKEY_MINUS,
    //@brief   : direction key→
    IKEY_RIGHT,
    //@brief   : direction key←
    IKEY_LEFT,
    //@brief   : Trigger key
    IKEY_TRIGGER,
    //@brief   : CH1 key
    IKEY_CH1,
    //@brief   : CH2 key
    IKEY_CH2,
    //@brief   : UTILITY key
    IKEY_LOCAL =34,
    //@brief   : push functional knob (Enter/OK)
    IKEY_OK,
}UTG2025KeyBit;
```