Remote Control

In order to avoid any potential issues, robust and stable connection is preferred. WAVESLINE UVS device supports the connection via USB or LAN. In general, USB3 is preferred for better transfer speed while LAN is preferred for longer distance control.

The USB device driver is integrated in Windows 10, thus no extra driver is required. WAVESLINE provides C++ based DLL, which can be imported into most popular software development environments such as C# and simplifies the operation when controlling UVS device.

WAVESLINE also provides a Windows and .Net Framework based software, with which user may easily setup and control the UVS device.

With regard to the LAN connection, only Gigabit connection is supported. The UVS device works as a TCP Server while host PC works as a TCP Client. End-User may complain that equipment can be accidently controlled by other host PC in network by mistake, in this case, user ATE test may be interrupted. To avoid this, WAVESLINE UVS device filters host IP and accepts only one authorized IP as host to respond.

Command List

To control the UVS device, user may send commands and read responses to/from UVS, the command systems are listed as below.

Note:

1, Command terminates with a "\n" character.

2, Commands are NOT case sensitive.

3, Commands are in string format.

4, Each Command returns a response.

5, Device Accepts only one command each time.

FREQ<space><value>

Function: Set Output Function

Value: frequency in Hz

Returns: value

Eg: "freq 3500000000\n" returns "3500000000"

FREQ?

Function: Query Frequency

Returns: frequency in Hz

Eg: “freq?\n”

POWR<space><value>

Function: Set Output Power

Value: output power in dBm

Returns: value

Eg: “powr -10.5\n” returns “-10.5”

POWR?

Function: Query Output Power

Returns: output power in dBm

Eg: “powr?\n”

OUTP<space><value>

Function: Set Output ON/OFF

Value: 0, OFF; 1 ON

Returns: value

Eg: “outp 1\n” returns “1”

OUTP?

Function: Query Output ON/OFF

Returns: value of Output ON/OFF

Eg: “outp?\n”

MOD<space><value>

Value: 0, OFF; 1 ON

Returns: value

Eg: “mod 1\n” returns “1”

MOD?

Function: Query MOD ON/OFF

Returns: value of Mod On/Off

Eg: “mod?\n”

BBG<space><value>

Value: 0, OFF; 1 ON

Returns: value

Eg: “bbg 1\n” returns “1”

BBG?

Function: Query Baseband ON/OFF

Returns: value of Baseband On/Off

Eg: “bbg?\n”

EXREF<space><value>

Value: 0, Use Internal REF; 1 Use External REF

Returns: value

Eg: “exref 1\n” returns “1”

EXREF?

Function: Query External or Internal REF?

Returns: value of REF Selected

Eg: “exref?\n”

LO2<space><value>

Value: 0, OFF; 1 ON

Returns: value

Eg: “lo2 1\n” returns “1”

WFMCONT?

Function: Query The Number of WFM Files Stored in UVS

Returns: number of WFM files

Eg: “wfmcont?\n”

WFMNAME<space><value>

Function: Retrieve WFM Name by Index

Value: WFM Index number, which starts from “0”

Returns: Retrieve WFM Name

Eg: “wfmname 0\n”

REFRESHWFM

Function: Refresh WFM Information

Returns: number of WFM files. Use this command when WFM file is added or deleted.

Eg: “refreshwfm\n”

IQQMC<space><value1><“,”><value2><“,”><value>

Function: Set QMC Values.

value1: Gain I

value2: Gain Q

value3: Phase

Returns: Phase value

Eg: “iqqmc 1920,1925,125\n” returns “125”

IQOFS<space><value1><“,”><value2>

Function: Set QMC Offset Values

value1: Offset I

value2: Offset Q

Returns: Offset Q value

Eg: “iqofs -68,56\n” returns “56”

Note: When BBG is turned on, UVS applies default QMC and OFS values, user values can be set after BBG is turned on.

GAINI?

Function: Query Default Gain I value

Returns: Gain I value

Eg: “gaini?\n”

GAINQ?

Function: Query Default Gain Q value

Returns: Gain Q value

Eg: “gainq?\n”

IQPHS?

Function: Query Default I,Q phase value

Returns: phase value

Eg: “iqphs?\n”

OFSI?

Function: Query Default OFS I value

Returns: OFS I value

Eg: “ofsi?\n”

OFSQ?

Function: Query Default OFS Q value

Returns: OFS Q value

Eg: “ofsq?\n”

Note: The default values should work for most applications at room temperature, values changes when RF frequency varies.

*IDN?

Function: Query UVS Device Information

Returns: UVS Device Information

Eg: “*idn?\n”

BBVER?

Function: Query UVS Baseband Firmware Information

Returns: UVS Baseband Firmware Information

Eg: “bbver?\n”

TEMP?

Function: Query UVS Internal Core Processor Temperature

Returns: Temperature in ºC

Eg: “temp?\n”

LOCALIP<space><value>

Function: Set UVS Local IPV4 Address

Value: a 32-bit integer value for IPV4 address, each IPV4 section occupies one byte wide.

Returns: IPV4 address integer

Eg: “localip 3232238270\n” //IPV4 address: 192.168.10.190

LOCALIP?

Function: Query UVS Local IPV4 Address

Returns: IPV4 address integer

Eg: “localip?\n”

LOCALPORT<space><value>

Function: Set UVS Local LAN Port

Value: Port value.

Returns: Port value

Eg: “localport 5025\n”

LOCALPORT?

Function: Query UVS Local LAN Port

Returns: Port value

Eg: “localport?\n”

HOSTIP<space><value1>

Function: Set UVS Host IPV4 Address

Value1: a 32-bit integer value for IPV4 address, each IPV4 section occupies one byte wide.

Returns: IPV4 address integer

Eg: “hostip 3232238279\n” //IPV4 address: 192.168.10.199

HOSTIP?

Function: Query UVS Host IPV4 Address

Returns: IPV4 address integer

Eg: “hostip?\n”

User DLL

The user dll is a C++ based dynamic library that wraps useful functions to minimize user efforts when trying to control the UVS device via USB interface. The integrated functions are listed as below.

```
int GetConnectedDevices();
        Scan devices and retrieve the number of active devices.


int GetType(int index);
        Get device type by index.


HANDLE GetHandel(int index);
        Get device handle(pointer) by index.
```

```
bool Open(HANDLE _handel);
```

Open device.

```
bool Close(HANDLE _handel);
```

Close device.

```
bool WfmClear(HANDLE _handel);
```

Clear all WFM files stored in UVS device.

```
int GetWfmCount(HANDLE _handel);
```

Get WFM file count stored in UVS device.

```
bool WfmAppend(HANDLE _handel, char* filename, char* dispname);
```

Append a new WFM file to UVS device memory.

filename: the WFM path to be added

dispname: the name of WFM file to be recognized by UVS device.

```
bool WfmDelLast(HANDLE _handel);
```

Delete the last WFM file in UVS memory.

```
bool Send(HANDLE _handel, char* cmd);
```

Send a command to UVS device.

cmd: command string, terminates with a “\n” character.

```
char* Read(HANDLE _handel);
```

Read response from UVS device.

```
bool LoadWfm(HANDLE _handel, char* filename);
```

Load an existing WFM file from PC.

```
        filename: the WFM path to be loaded.


bool LoadIQ(HANDLE _handel, double* data, int datcnt, UInt32 iqrate);

        Load I/Q data from double array.

        data: a double array contains I and Q data. The array length should be 2
              times of I/Q samples; data[0] ~ data[N-1] are I sample values, while
              data[N] ~ data[2N-1] are Q sample values.

        datcnt: I/Q sample count. The data array length should be 2 times of I/Q
              sample count. Due to the UVS hardware structure, I/Q sample count
              should be Integer times of 128.

        iqrate: I/Q sample rate.
```

Import Functions to C#

C# is widely used in automatic testing program developing. Since pointer is abandoned by C#, importing C++ dll have to make some data type mapping during the process. Below is an example showing the procedure, importing dll to other language environments are similar.

```
[DllImport(@"UvsLib.dll", CallingConvention = CallingConvention.Cdecl, SetLastError = true)]
extern static int GetConnectedDevices();

[DllImport(@"UvsLib.dll", CallingConvention = CallingConvention.Cdecl, SetLastError = true)]
extern static int GetType(int index);

[DllImport(@"UvsLib.dll", CallingConvention = CallingConvention.Cdecl, SetLastError = true)]
extern static IntPtr GetHandel(int index);

[DllImport(@"UvsLib.dll", CallingConvention = CallingConvention.Cdecl, SetLastError = true)]
extern static bool Open(IntPtr _handel);
```

```
[DllImport(@"UvsLib.dll", CallingConvention = CallingConvention.Cdecl, SetLastError = true)]
extern static bool Close(IntPtr _handel);

[DllImport(@"UvsLib.dll", CallingConvention = CallingConvention.Cdecl, SetLastError = true)]
extern static bool Send(IntPtr ftHandle, string cmd);

[DllImport(@"UvsLib.dll", CallingConvention = CallingConvention.Cdecl, SetLastError = true)]
extern static IntPtr Read(IntPtr ftHandle);

[DllImport(@"UvsLib.dll", CallingConvention = CallingConvention.Cdecl, SetLastError = true)]
extern static bool WfmClear(IntPtr ftHandle);

[DllImport(@"UvsLib.dll", CallingConvention = CallingConvention.Cdecl, SetLastError = true)]
extern static int GetWfmCount(IntPtr ftHandle);

[DllImport(@"UvsLib.dll", CallingConvention = CallingConvention.Cdecl, SetLastError = true)]
extern static bool WfmAppend(IntPtr ftHandle, string filename, string dispname);

[DllImport(@"UvsLib.dll", CallingConvention = CallingConvention.Cdecl, SetLastError = true)]
extern static bool WfmDelLast(IntPtr ftHandle);

[DllImport(@"UvsLib.dll", CallingConvention = CallingConvention.Cdecl, SetLastError = true)]
extern static bool LoadWfm(IntPtr _handel, string filename);

[DllImport(@"UvsLib.dll", CallingConvention = CallingConvention.Cdecl, SetLastError = true)]
extern static bool LoadIQ(IntPtr _handel, double[] data, int datcnt, UInt32 iqrate);
```

Note: Marshal can be used in C# to convert between IntPtr and data array.

Load IQ Data

An IQ sample is defined as a 16-bit I and a 16-bit Q data.

User may write arbitration data to UVS baseband generator that can be modulated to RF band for output. There are two approaches available for writing IQ data to UVS device. Call 'LoadIQ' function in user DLL or use 'LOADIQ' command.

When 'LoadIQ' function is used, the IQ data array must in below format

doubles	doubles
I[0] ~ I[N-1]	Q[0] ~ Q[N-1]

[Example]: write 1024 IQ samples

(this example uses function)

```
double[] data = new double[2048];    // length = 2048, 1024 x I + 1024 x Q
…       //fill iq data
int datacnt = 1024;      //IQ sample count = 1024
UInt32 iqrate = 300000000;   //IQ sample replay rate = 300MSPS
bool success = LoadIQ(_handel, data, datcnt, iqrate);
```

When 'LOADIQ' command is used, follow the procedure:

1, send command

LOADIQ<space><byte length>

byte length = 4 x IQ sample count

Read response, it returns the number of pages to be transferred.

2, send data pages

Each 128 samples(512 bytes) composes a 'page', the IQ samples are sent by pages.

The IQ data array must in below format, each I or Q is a 16-bit integer.

IQ Sample[0]		IQ Sample[1]		IQ Sample[2]		…
16bit I	16bit Q	16bit I	16bit Q	16bit I	16bit Q	…

The array length of IQ samples needs to be N(integer) times of 128 (512 bytes).

Send the 1st page of data, 512 bytes, read response, it returns the number of pages to be transferred.

Send the 2nd page of data, 512 bytes, read response, it returns the number of pages to be transferred.

…

[Example]: write 1024 IQ samples

(this example uses command)

Write command: "loadiq 4096\n" // 1024 samples x 4 = 4096 bytes

Read response: "8"

Write the 1st page of data(512 bytes), Read response: "7"

Write the 2nd page of data(512 bytes), Read response: "6"

…

Write the last page of data(512 bytes), Read response: "0"

Write command: "iqrate 300000000\n" // IQ sample replay rate = 300MSPS

Load Waveform File

A Waveform File (*.wfm) contains all necessary information and data for UVS baseband operation. This is usually the operation of loading LTE or 5G NR waveform. User can directly load a waveform file to UVS device by calling DLL function 'LoadWfm'.

Write Waveform

A waveform can be written to UVS so that user can load it from UVS directly, with no need to write IQ data from host PC. Call '`WfmAppend`' function in user DLL simplifies the process.

When '`WfmAppend`' function is used, simply use a string variable to specify the waveform file location.

[Example]: write 5G NR waveform, this example uses function

string file = "c://5g_tm31a.wfm";

bool success = WfmAppend(ftHandle, file, "5G NR TM31a");

LAN Connection

The host PC acts as a TCP Client.

User may develop a TCP client software or make use Wavesline TCP Client Tool.

[Example]: Create a LAN Connection Setup.

Device Address: 192.168.10.190, Socket Port: 5025

Host PC Address: 192.168.10.199

When any 3rd-party VISA32/IEEE488 software or libraries are used, the setup procedure is similar.