UNI-T

UCI Help Document

Versions：

Versions	Revised item
V1.0	Official version
V1.1	Provide functions of query, open, read&write data， all simplified version will be marked with an X. More details on uci.h
V1.2	Add *uci_SetAttribute* and *uci_SetNotify* interface introduction.

Introduction：

UCI interface support uniform communication of multiple devices and interfaces. Every communication protocol of a specified device has corresponding file. This file only introduces the use of UCI interface.

You can also refer to demonstration case in SDK

uci.h is the finial reference

Note：

1. This SDK provide UCI depot of UNICODE and ASCII coding；please select the suitable version. For example, LABVIEW use ASCII. VC default use UNICODE
2. All interfaces has Timeout parameter，which means：if the task not finished within the specified time(ms), it will return to the original interface

Reference：

1. ucidef.h
   UCI depot basic definition
2. uci.h
   UCI depot interface definition
3. uci.lib
   UCI dynamic depot link document
4. uci.dll
   UCI dynamic depot document

Interface introduction：

1. uci_QueryNodes

Query specified communication node for obtaining all connectable devices. You can use simplified interface uci_QueryNodesX.

Syntax：

```
u_status _UCIAPI uci_QueryNodes(_in const QParams* _params,
                                _out Node* _outBuf,
                                _in_out u_size* _nodeCnt,
                                _in u_size _timeOut);
```

Parameters ：

const QParams **_params**	Specify query parameters
Node **_outBuf**	Save node data
u_size **_nodeCnt**	Enter parameter: required node quantity , exit parameter: actual obtained node quantity
u_size **_timeOut**	Query timeout , unit： ms.

Return Value：

<0: represent error, error code refer to ：UCI error code

Remarks ：

Required node quantity is decided by the operator. Introduce the corresponding **_outBuf** and **_nodeCnt**

eg：

```
Node nodes[10];
ZeroMemory(nodes, sizeof(nodes));
u_size notesCNT = sizeof(nodes) / sizeof(nodes[0]);
int r = uci_QueryNodes(&qp, nodes, &notesCNT, 2000);
if (r < 0) {
    TRACE(_T("\r\n Error to query nodes, err code = 0x%x"), r);
    return;
}
if (notesCNT == 0){
    AfxMessageBox(_T("no result！ "));
```

```
        return ;
    }
```

Creat qp：QParams

2. *uci_QueryNodesX*

Query specified communication node, obtain all connectable devices. You can use simplified interface uci_QueryNodesX.

Syntax：

```
u_status _UCIAPI uci_QueryNodesX(u_cstring _msg, Node* _nodes,
                                 u_size _node_count, u_size _timeout);
```

Parameters ：

u_cstring ***_msg***	Query character string， format： "LAN:4162,5000;USB:0x1234&0x5345,0x7777&0x5345;"
*Node** ***_nodes***	Save obtained node data, see Node.
u_size ***_node_count***	Required node quantity
u_size ***_ timeout***	Query timeout，unit： ms.

Return Value：

<0 :represent error, error code refer to ：UCI error code

Remarks ：

Required node quantity is decided by the operator. Introduce the corresponding **_outBuf** and **_nodeCnt**

eg：

```
Node nodes[10];
ZeroMemory(nodes, sizeof(nodes));
u_size notesCNT = sizeof(nodes) / sizeof(nodes[0]);
int r = uci_QueryNodesX(_T("LAN:4162,5000;USB:0x1234&0x5345,0x7777&0x5345;"), nodes, notesCNT, 2000);
if (r < 0) {
    TRACE(_T("\r\n Error to query nodes, err code = 0x%x"), r);
    return;
}
if (notesCNT == 0){
    AfxMessageBox(_T("no result！ "));
```

```
    return ;
}
```

3. *uci_Open*

Connect with the device. You can use the simplified version uci_OpenX.

Syntax：

```
u_status _UCIAPI uci_Open(_in u_cstring _addr,
                          u_session* _session,
                          u_uint32 _timeOut = 6000);
```

Parameters ：

u_cstring ***_addr***	Device connecting character string, end up with'\0'
*u_session** ***_session***	Return to created dialogue ID which will be used in following interface
u_size ***_timeOut***	Connecting timeout, unit： ms.

Return Value：

<0 :represent error, error code refer to ：UCI error code

Remarks ：

The simplest method to confirm ***_addr*** 的 is to obtain in specified model's document or use uci_QueryNodes to query connecting devices, and obtain character string address through Node.***UCIAddr***

4. *uci_OpenX*

Connect with the device. You can use the simplified version uci_Open

Syntax：

```
u_status _UCIAPI uci_OpenX(_in u_cstring _addr, u_uint32 _timeOut);
```

Parameters ：

u_cstring ***_addr***	Device connecting character string, end up with'\0'
u_size ***_timeOut***	Connecting timeout, unit： ms.

Return Value：

<0: represent error, error code refer to ：UCI error code

>0: represent dialogue ID

Remarks ：

The simplest method to confirm ***_addr*** 的 is to obtain in specified model's document or use uci_QueryNodes to query connecting devices, and obtain character string address through Node.***UCIAddr***

example：

```
static u_session g_session = INVALID_SESSION;

//....

u_status r =
       uci_OpenX("[C:DSO][D:UPO2000CS][T:USB][PID:0x1234][VID:0x5345][EI:0x81][EO:0x3][CFG:3]", 2000);
if (UCISUCCESS(r))
    g_session = (u_session)r;
```

5. uci_Close

close dialogue

Syntax：

u_status _UCIAPI uci_Close(u_session ***_session****)*

Parameters ： null

Return Value：

<0: represent error, error code refer to ：UCI error code

Remarks ：

uci_Close and uci_Open occur at the same time. After creat connection, you should close it when exit.

6. *uci_Write*

write data, you can use simplified verstion uci_WriteX.

Syntax：

```
u_status _UCIAPI uci_Write(u_session _session,
                           PWParams _params,
                           const u_byte* _data = nullptr,
                           u_size _len = 0);
```

Parameters ：

u_session ***_session***	Dialogue ID. Obtained through uci_Open
PWParams ***_params***	Write operating parameter，see *WParams*.
*const u_byte** ***_data***	Data buffer area address， can be NULL
u_size ***_len***	Data buffer area length， If _buf == NULL, then_len will be ignored and be arbitrary value

Return Value：

<0: represent error, error code refer to ：UCI error code

Remarks ：

Common built-in data type，can format the data to character string instead of using***_data*** and***_len*** to write data.

```
int UCIWrite(const TCHAR* _cmd, UINT _timeOut) {
    WParams wp;
    ZeroMemory(&wp, sizeof(wp));
    return uci_Write(g_curSession, uci_CreateWParams(wp, _cmd, _timeOut));
}
```

g_curSession is obtained by uci_Open

if the data cannot be formatted into character string. You can use***_data*** and ***_len*** to write data.

7. *uci_WriteX*

write data, you can use simplified verstion uci_WriteX

.

Syntax：

u_status _UCIAPI uci_WriteX(*u_session* ***_session***,
u_cstring ***_msg***,
u_uint32 ***_timeout***,
*const u_byte** ***_data***,
u_size ***_len***);

Parameters ：

u_session ***_session***	Dialogue ID. Obtained through uci_Open
u_cstring ***_msg***	Write command character string
u_uint32 ***_timeout***	Write timeout
*const u_byte** ***_data***	Data buffer area address， can be NULL
u_size ***_len***	Data buffer area length， If _buf == NULL, then_len will be ignored and be arbitrary value

Return Value：

<0: represent error, error code refer to ：UCI error code

Remarks ：

Common built-in data type，can format the data to character string instead of using***_data*** and***_len*** to write data.

if the data cannot be formatted into character string. You can use***_data*** and ***_len*** to write data.

8. uci_WriteSimple

write data. This is the simplified version of uci_Write, cancel the write data interface of binary system

Syntax：

u_status _UCIAPI uci_WriteSimple(*u_session* ***_session***, *u_cstring* ***_msg***, *u_uint32* ***_timeout***);

Parameters ：

u_session ***_session***	Dialogue ID. Obtained through uci_Open
u_cstring ***_msg***	Write command character string
u_uint32 ***_timeout***	Write timeout

Return Value：

<0: represent error, error code refer to ：UCI error code

Remarks ：

Common built-in data type，can format the data to character string instead of using ***_data*** and ***_len*** to write data.

if the data cannot be formatted into character string. You can use ***_data*** and ***_len*** to write data.

9. uci_FormatWrite

write data in the form of formatting character string

Syntax：

```
u_status _UCIAPI uci_FormatWrite( u_session _sesn,
                                  u_uint32 _timeOut,
                                  const u_char *format, ...);
```

Parameters ：

u_session ***_sesn***	Dialogue ID. Obtained through uci_Open
u_size ***_timeOut***	Write command character string
const u_char ****format****, ...*	Write timeout

Return Value：

<0: represent error, error code refer to ：UCI error code

Remarks ：

This interface uses uci_Write interface

example：

```
void SendKey(int _key) {
#if 0
    CString s;
    s.Format(_T("KEY:%d;"), _key);
    UCIWrite(s, 1000);
#else
    uci_FormatWrite(g_curSession, 1000, _T("KEY:%d;"), _key);
#endif
}
```

UCIWrite see uci_Write introduction

10. uci_Read

read data, you can also use simplified version：uci_ReadX

Syntax：

u_status _UCIAPI uci_Read(*u_session* ***_session***,
PRParams ***_params***,
*u_byte** ***_data***,
u_size ***_dataLen***);

Parameters ：

u_session ***_session***	Dialogue ID. Obtained through uci_Open
PRParams ***_params***	Read command parameters see RParams.
*u_byte** ***_data***	Receive buffer area of data
u_size ***_dataLen***	Size of buffer area for data receiving，size is decide by protocol

Return Value：

<0: represent error, error code refer to ：UCI error code

Remarks ：

11. uci_ReadX

read data, simplified version of uci_Read

Syntax：

u_status _UCIAPI uci_ReadX(*u_session* ***_session***, *u_cstring* ***_msg***, *u_uint32* ***_timeout***,
*u_byte** ***_data***, *u_size* ***_dataLen***);

Parameters ：

u_session ***_session***	Dialogue ID. Obtained through uci_Open
u_cstring ***_msg***	Read command character string
u_uint32 ***_timeout***	Read command timeout
*u_byte** ***_data***	Receive buffer area of data
u_size ***_dataLen***	Size of buffer area for data receiving，size is decide by protocol

Return Value：

<0: represent error, error code refer to ：UCI error code

Remarks ：

Only execute one read command one time

12. uci_WriteFromFile

write file, you can also use simplified version：uci_WriteFromFileX.

Syntax：

*u_status _UCIAPI uci_WriteFromFile(u_session **_session**, WFileParams* **_info**);*

Parameters ：

u_session ***_session***	Dialogue ID. Obtained through uci_Open
*WFileParams** ***_info***	Write file parameters（WFileParams）

Return Value：

<0: represent error, error code refer to ：UCI error code

Remarks ：

13. uci_WriteFromFileX

write file, is the simplified version of uci_WriteFromFile

Syntax：

*u_status _UCIAPI uci_WriteFromFileX(u_session **_session**,*
*u_cstring **_msg**,*
*u_cstring **_filePath**,*
*u_uint32 **_timeout**);*

Parameters ：

u_session ***_session***	Dialogue ID. Obtained through uci_Open
u_cstring ***_msg***	Command character string, end up with'\0'
u_cstring ***_filePath***	File path (file name)

u_uint32 ***_timeout***	Write timeout(ms)

Return Value：

<0: represent error, error code refer to ：UCI error code

Remarks ：

14. uci_ReadToFile

read data to file, you can also use the simplified version：uci_ReadToFileX.

Syntax：

u_status _UCIAPI uci_ReadToFile(u_session ***_session****, RFileParams** ***_params****);*

Parameters ：

u_session ***_session***	Dialogue ID. Obtained through uci_Open
*RFileParams** ***_params***	Read file data parameter. see：RFileParams

Return Value：

<0: represent error, error code refer to ：UCI error code

Remarks ：

15. uci_ReadToFileX

read data to file, is the simplified version of uci_ReadToFile.

Syntax：

u_status _UCIAPI uci_ReadToFileX(u_session ***_session****, u_cstring* ***_msg****, u_cstring* ***_filePath****,*
u_uint32 ***_timeout****, u_tchar** ***_filePathFinal****, u_int32* ***_filePathFinalLength****);*

Parameters ：

u_session ***_session***	Dialogue ID. Obtained through uci_Open
u_cstring ***_msg***	Command character string, end up with'\0'
u_cstring ***_filePath***	File path to buffer data to local disk. (including file name and suffix),

	canbe absolute path or relative path. If it is relative, you can obtain absolute through_filePathFinal
u_uint32 ***_timeout***	Timeout(ms)
u_tchar ****_filePathFinal***	Absolute file path
u_int32 ***_filePathFinalLength***	Length of filePathFinal，count by character

Return Value：

<0: represent error, error code refer to ：UCI error code

Remarks ：

16. uci_SendCommand

send control command

Syntax：

u_status _UCIAPI uci_SendCommand(u_session ***_session****, PCommandParams* ***_params****);*

Parameters ：

u_session ***_session***	Dialogue ID. Obtained through uci_Open
PCommandParams ***_params***	Command parameter, see *CommandParams*.

Return Value：

<0: represent error, error code refer to ：UCI error code

Remarks ：

17. uci_GetLastError

get the last error information

Syntax：

u_cstring _UCIAPI uci_GetLastError();

Parameters ：

Return Value：

Return character string of error information

Remarks ：

19、 uci_SetAttribute

set up attribute

Syntax：

*u_status _UCIAPI uci_SetAttribute(u_session **_session**, u_cstring **_msg**, const u_object* **_obj**, u_size **_objSize**);*

Parameters ：

*u_session **_session***	Dialogue ID. Obtained through uci_Open
*u_cstring **_msg***	Attribute character string, see Remarks section.
const u_object **_obj***	Data buffer area address
*u_size **_objSize***	Data buffer area size

Return Value：

<0: represent error, error code refer to ：UCI error code

Remarks ：

Current support attributes:

1. set up UCT interface error information language Chinese ： “lang:zh-hans;” English ： ”lang:en-Us;”
2. subscription notification of device connecting or removal : “devchange:1”； you can use this task to deal with the logic of offline, online, reconnection.

Example：

uci_SetAttribute(INVALID_SESSION, _T("lang:zh-Hans;devchange:1;"), nullptr, 0);

Note： ***_session*** must be INVALID_SESSION.

20、 uci_SetNotify

add UCI event subscription

Syntax：

*u_status _UCIAPI uci_SetNotify(UCIMSGProc **_pNotify**)*

typedef int(__stdcall *UCIMSGProc)(*UCIMSG* **_msg**);

Parameters ：

UCIMSGProc ***_pNotify***	Response the subscription of callback function

Return Value：

<0: represent error, error code refer to ：UCI error code

Remarks ：

You can use this interface to add response interface if you need to add device removal and event.

Data structure：

1、QParams（query parameters）

```
typedef struct _QParams {
        //@brief : communication node type
        //@remarks : get value from enum NodeType, remark the save node type value
        //@eg ：  NodeType::USB | NodeType::LAN
        u_int32     Type;
        //@brief : port count
        u_int32     PortCount;
        //@brief : port set
        u_int32*    Ports;
        //@brief : PVID count
        //@remarks :
        u_int32     PVIDCount;
        //@brief : PID and VID set
        //@remarks : use MakePVID to creat， GetPID and GetVID to obtain
        u_int32*    PVID;
    }QParams, *PQParams;//@brief : configured parameter when query the device
```

introduce file： ucidef.h

you can use interface UCI_CreateQParam to creat this structure data：

eg：

```
QParams qp;
    int pvids[10] = { MakePVID(0x1234, 0x5345) };
```

only query USB port：

```
UCI_CreateQParam(qp, NodeType::USB, NULL, 0, pvids, 1);
```

Query USB and LAN ports：

```
int ports[10] = { 4162, 8000 };
UCI_CreateQParam(qp, NodeType::USB | NodeType::LAN, ports, 2, pvids, 1);
```

2、WParams（write data package parameters）

```
//@brief : write parameters
//@remarks : you can use interface uci_CreateWParams to create
typedef struct _WParams {
    //@brief : command character string, end up with '\0'.
    u_cstring CMDString;
    //@brief : return value size
    u_uint32    RetCount;
    //@brief : read timeout
      _uint32    Timeout;
}WParams, *PWParams;
```

CMDString format is different between devices

3、RParams（read data package parameters）

```
//@brief : read parameters
//@remarks : use uci_CreateRParams interface to create ordinary read parameters command
typedef struct _RParams {
    //@brief : command character string end up with'\0'
    u_cstring CMDString;
    //@brief : return value size
    u_uint32    RetCount;
    //@brief : read timeout
    u_uint32    Timeout;
    //@remarks : do not use it normally, can be null. It is used when query devices
    //data type is QParams，other types refer to files
    u_buf        ExtraData; //reserve
```

```
        //@brief : extra data length
        u_size      ExtraDataLen;
    }RParams, *PRParams;
```

This structure is used in uci_Query, uci_QueryNodes and uci_Read
ExtraData and ExtraDataLen are only used in special occasions, they can be null in most cases

4、WFileParams（write file parameters）

```
    //write file parameters
    typedef struct _WFileParams {
        //@brief : command character string, end up with '\0'
        u_cstring   CMDString;
        //file path（including file name）
        u_cstring   FilePath;
        //write timeout(ms)
        u_uint32    TimeOut;
    }WFileParams, *PWFileParams;
```

CMDString: for details of command character string, please refer to protocol files

FilePath：full path, including file name and suffix

5、RFileParams（read file name）

```
    //read file parameter
    typedef struct _RFileParams {
        //@brief : command character string, end up with'\0'
        u_cstring    CMDString;
        //file path（including file name）
        u_cstring    FilePath;
        //read timeout(ms)
        u_uint32     TimeOut;
    }RFileParams, *PRFileParams;
```

CMDString : command character details, please refer to protocol files
FilePath ： including file name and suffix

6、PCommandParams （command parameters）

```
    typedef struct _CommandParams {
```

```
        //@brief : command character string, end up with'\0'
        u_cstring CMDString;
        //@brief : command parameter1
        u_uint32    Param1;
        //@brief : command prameter2
        u_uint32    Param2;
        //@brief : timeout
        u_uint32    Timeout;
    }CommandParams, *PCommandParams;
```

CMDString : refer to protocol files

7、Node （node/device interface description）

```
        //@brief : node parameters
        //@remarks :
        typedef struct _Node {
            //@brief : communication interface type
            NodeType Type;
            //@brief : device name(model)
            //@remarks : export from this UCI depot, the name is the same as UCI depot protocol
        //                  UCI depot will automatch the name of the device if there is not setting after leaving the factory.
            u_tchar     Name[50];
            //@brief : device type
            //@remarks : signal source = SG, oscilloscopes = DSO;
            u_tchar     DevType[10];
            //@brief : LAN port parameter
            LANDescriptor LAN;
            //@brief : USB port parameter
            USBDescriptor USB;
            //@brief : connecting character string
            u_tchar     UCIAddr[256];
            //@brief : serial number
            u_tchar     SN[50];
            //@brief : device status
            u_status    Status;
            //@brief : actual display name
            u_tchar     IDN[20];
        }Node, *PNode;

        //@brief : node type
        //@remarks : In QParams, bitand， in Node, enum value
        typedef enum _NodeType{
```

```
    LAN = 0x0001,
    USB = 0x0010,
}NodeType;

//@brief : USB descriptor
//@remarks :
typedef struct _USBDescriptor {
    //@brief : PID
    u_ushort    PID;
    //@brief : VID
    u_ushort    VID;
    //@brief : address
    u_ushort    Addr;
}USBDescriptor;

//@brief : IP address
//@remarks : padding sequence f1(192).f2(168).f3(1).f4(253) – little end mode
typedef union _IPAddr {
    struct { u_byte f1, f2, f3, f4; } Field;
    u_int32 Addr;
}u_IPAddr;

//@brief : LAN port descriptor
//@remarks :
typedef struct _LANDescriptor {
    //@brief : IP address（character string type）
    u_tchar      IP[16];
    //@brief : IP address
    u_IPAddr     Addr;
    //@brief : network port
    //@remarks : TCPIP port number
    u_ushort     Port;
}LANDescriptor;
```

General Statement：

1、UCI error code

All return values of UCI interface share the same meaning, that is <0 indicates error，error code definition see ucidef.h（always comply to this files）.

Corresponding error information can be obtained through uci_GetLastError interface

You can use macro definition to judge interface return value：UCISUCCESS and UCIERR

You can switch the language： Chinese：“lang:zh-hans;” English：”lang:en-Us;”

uci_SetAttribute(INVALID_SESSION, _T("lang:zh-Hans;devchange:1;"), nullptr, 0);

Error code	Description
0	Succeed！
-5	The requested source is being used.
-116	Request time out
-1020	Resource initialization error
-1019	Invalid dialogue ID
-1018	Operation time out
-1017	Fail
-1016	Unsupported operation
-1015	Insufficient memory resources
-1014	System busy
-1013	Communication anomaly, irreversible！
-1012	Channel not open！
-1041	Unknown error
-1040	Error in connective character string address format
-1039	Connection has not been created
-1038	disconnected
-1037	Unsupported communication type
-1060	No specified device was found
-1059	Unsupported device
-1058	Please query device first
-1057	Device mismatch
-1056	Fail to query network device
-1055	The USB device address only works after the query device is executed
-1054	No USB is found.
-1053	The key has been locked！
-1080	Error in command string format
-1079	Support only single command
-1078	A command only supports one attribute
-1077	unsupported command
-1076	Send command failure
-1075	Protocol data format error

-1074	Device fails to write file to flash!
-1073	No valid response data were found
-1072	Command error, please check relevant protocol!
-1071	The expression is invalid!
-1120	Digital overflow!
-1119	Out of scope!
-1118	The data is not all read!
-1117	Data validation failed!
-1116	Invalid data!
-1115	Compression error!
-1114	Decompression error!
-1113	Data transmission error!
-1112	Data transmission interrupt!
-1111	No data to read!
-1140	No access!
-1139	An unspecified error occurred!
-1138	File not found!
-1137	Invalid path ！
-1136	Exceeds the number of open files
-1135	Invalid file handle
-1134	The current working directory cannot be removed
-1133	Disk full
-1132	Setting file pointer error
-1131	Hardware error occurred
-1130	SHARE.EXE, unloaded, or shared area lock!
-1129	Try to lock the locked area
-1128	Full disk
-1127	Reached the end of the file
-1126	Fail to write file to disk