

MultiDII Interface User Manual

Date: 2013-4-9

Version: V1.0

Record:

Revised date	Version	Contents	Writer	Approver
2013-4-9	V1.0	New	Hu Qihui	

Contents

Contents.....	3
1. DLL Introduction.....	4
2. Including header file.....	4
3. Library functions operation flow.....	4
4. Library Function interface description.....	5
4.1AS3992 (UART) library function interface description.....	5
4.1.1 Reader connection.....	5
4.1.3 Reader Setup.....	6
4.1.4 Reset reader.....	8
This function will make the reader's parameter recovery to default value, it's used when wrong-set parameter and refresh to default status.....	8
4.1.5 Reboot Reader.....	8
4.1.6 Inventory tags.....	9
4.1.7 Stop inventory.....	9
4.1.8 Select tag.....	10
4.1.9Read tag's memory block data.....	11
4.1.10 Write data into tag's memory area.....	11
4.1.11 Lock tag.....	12
4.1.12 Kill tag.....	13
4.2Indy library function interface description.....	13
4.2.1 connect reader.....	13
4.2.2 disconnect reader.....	13
4.2.3 inventory tags.....	14
4.2.4 select tag.....	14
4.2.5 read data of tag's memory area.....	14
4.2.6 write data into tag's memory area.....	15
4.2.7 lock tag's memory area.....	15
4.2.8 get serial number of module.....	16
4.2.9 Cancel current operation.....	16
4.2.10 reset module.....	16
4.2.11 abort module.....	17
4.2.12 kill tag.....	17
4.3AS3992 (Ethernet) library function interface description.....	17
4.3.1 set event notice callback function.....	17
4.3.2 Start Sever.....	18
4.3.2 Stop service.....	19
4.3.3 Connect Reader.....	19
4.3.4 Disconnect Reader.....	20
4.3.5Get version of the reader.....	20
4.3.6 Set Reader.....	20
4.3.7 Reset reader.....	22
4.3.8 Reboot reader.....	23
4.3.9 Inventory tags.....	23
4.3.10 Stop inventory.....	24
4.3.11 Select tag.....	24
4.3.12Get tag's memory area data.....	25
4.3.12Write data into memory area.....	26
4.3.13 Lock tag.....	27
4.3.14 Kill tag.....	27

1. DLL Introduction

MultiDLL is dll used of MultiUHF, including all kinds of interface function the UHF module, user only need to add this Library file and according Header files in programming projects, it's able to operate all the interface of the modules.

MultiDll use standard .dll interface technology, enable it can be invoked easily in any kinds of programming language, and no need to do transfer with the Third tools.

2. Including header file

Following are the header files included in necessary:

- Indy_COM_Interface.h
- AS3992_COM_Interface.h
- MultiDll.h
- rfid_structs.h
- rfid_types.h
- rfid_constants
- rfid_library_export.h
- rfid_platform_types.h

3. Library functions operation flow

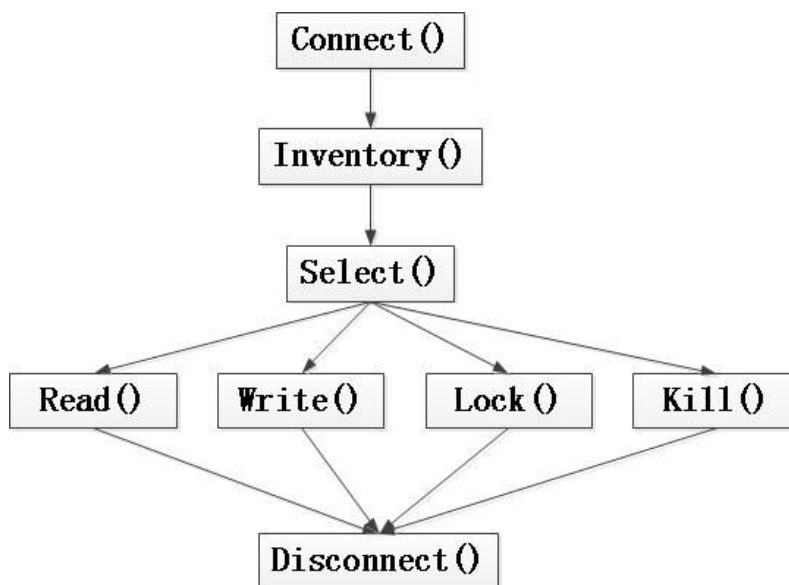


Figure 1 Function invoking flow

4. Library Function interface description

4.1 AS3992 (UART) library function interface description

4.1.1 Reader connection

Structure definition:

Structural body in file:MultiDll.h

```
/* Reader information type */
```

```
typedef struct
```

```
{
```

```
    unsigned char comPort;  
    unsigned long baudRate;  
    unsigned char hardwareVersion[40];  
    unsigned char softwareVersion[40];  
    unsigned char serialNumber[40];
```

```
}READER_INFO_TYPE;
```

- comPort : Reader using port;
- baudRate: Reader's port Baudrate;
- hardwareVersion: Reader Hardware version ;
- softwareVersion : Reader software version ;
- serialNumber: Reader's serial number ;

```
/******
```

```
** Function name:      as3992_connect
```

```
** Descriptions:      This function use to connect the reader
```

```
** input parameters:   readerInfo      : Point to the reader information struct
```

```
** output parameters:  none
```

```
** Returned value:     TRUE  : Connect success
```

```
                     FALSE   : Connect failed
```

```
** Created by:         Wanwu
```

```
** Created date:       2013-3-22
```

```
**
```

```
** Modified by:
```

```
** Modified date:
```

```
**
```

```
*****/
```

```
extern "C" BOOL _stdcall as3992_connect(void *readerInfo);
```

Return TRUE when success, and enclose Reader information into structure boy of readerInfo

```
/******
```

```
** Function name:      as3992_disConnect
```

```
** Descriptions:      This function use to disconnect the reader
```

```
** input parameters:   readerInfo      : Point to the reader information struct
```

```
** output parameters:  none
```

```
** Returned value:     TRUE  : Disconnect success
```

```
                     FALSE   : Disconnect failed
```

```

** Created by:      Wanwu
** Created date:    2013-3-22
** -----
** Modified by:
** Modified date:
** -----
*****/
extern "C" BOOL _stdcall as3992_disconnect(void);

```

Return TRUE when disconnect success;
Return FALSE when disconnect failed ;

4.1.3 Reader Setup

Structure definition:

Structural body in file: MultiDII.h

```

/* Config type define */
typedef enum
{
    SET_FREQUENCY      = 0,
    SET_LINK_FREQ,
    SET_CODING,
    SET_SESSION,
    SET_TREXT,
    SET_Q_VALUE,
    SET_SENSITIVITY,
    SET_RF_POWER,
    SET_GEN2,
    SET_MODE,
}UHF_CONFIG_TYPE;
● SET_FREQUENCY:    set frequency points
● SET_LINK_FREQ:    set link frequency
● SET_CODING:        set coding way
● SET_SESSION:       set session mode
● SET_TREXT:         set leading adjust
● SET_Q_VALUE:       set Q value
● SET_SENSITIVITY:   set sensitivity
● SET_RF_POWER:      set RF power
● SET_GEN2:          set GEN2 parameters
● SET_MODE:          set Reader mode
●

```

Structural body in file: MultiDII.h

```

/* UHF configure type */
typedef struct
{
    unsigned char    mode;           /* Reader mode */
    unsigned int     startFreq;      /* Start frequency */
    unsigned int     endFreq;        /* End frequency */
    unsigned char    rssi_thresh;    /* RSSI threshold */
    unsigned int     freqIncrement;  /* Frequency increment */
    unsigned char    linkFreq;       /* Link frequency */
    unsigned char    coding;         /* coding type */
    unsigned char    session;        /* session */
    unsigned char    trext;          /* trext */
}

```

```

        unsigned char    q_Value;        /* Q value */
        unsigned char    sensitivity;    /* Sensitivity */
        unsigned char    rf_power;       /* rf power */
        unsigned char    last_error;     /* last error number */
    }UHF_Configure;

```

Structure body definition :

```

mode          : Reader working mode,0x00 is standard mode,0x01 is auto-adjust mode
startFreq     : start Frequency, for example 922875 stands for 922.875 MHz
endFreq       : end frequency
freqIncrement  : frequency increment
rssi_thresh   :RSSI signal threshold value(-83 ~ 0)
linkFreq: link frequency rate
              0x00  40 kHz
              0x06  160 kHz
              0x08  213 kHz
              0x09  256 kHz
              0x0c  320 kHz
              0x0f  640 kHz
Coding        :coding method
              0x00  FM0 coding for rx
              0x01  MILLER2 coding for rx
              0x02  MILLER4 coding for rx
              0x03  MILLER8 coding for rx
Session       : session option
              0x00  Inventoried (S0)
              0x01  Inventoried (S1)
              0x02  Inventoried (S2)
              0x03  Inventoried (S3)

Trex          : preamble pre-pilot tone option, default 0x01
              0x00  preamble no pilot tone
              0x01  preamble adding pilot tone

gen2qbegin    : begin value choice of inventory tag's Q value, range from 0 ~15, default 4

Sensitivity    : Reader receiving sensitivity set value as negative of type of signed char(this
              is with sign character) , the best sensitivity is -84, this is 0xAC, default 0xAC

rf_power       : set RF output power, range from 0~19, corresponding to 27~8dbm, 0 is
              correspond to maximum power, 19 corresponds to minimum power

last_error     : stands the last error when setting parameters

```

```

/*****
** Function name:      as3992_config
** Descriptions:      This function use to config the reader

** input parameters:  config : Point to the reader configure struct
                      type  : Config type, see UHF_CONFIG_TYPE definition
** output parameters: none
** Returned value:    TRUE  : Success
                      FALSE : Failed

** Created by:        Wanwu
** Created date:      2013-3-22
** -----
** Modified by:
** Modified date:
** -----

```

```

*****/
extern "C" BOOL _stdcall as3992_config(UHF_Configure *config, UHF_CONFIG_TYPE type);
Return TRUE when setting success ;
Return FALSE when setting failed ;

```

4.1.4 Reset reader

This function will make the reader's parameter recovery to default value, it's used when wrong-set parameter and refresh to default status

```

/***** Function
name:      as3992_Reset
** Descriptions:  This function use to reset the reader to the default parameters

** input parameters:  none
** output parameters:  none
** Returned value:    TRUE : Success
                     FALSE : Failed

** Created by:      Wanwu
** Created date:    2013-3-22
** -----
** Modified by:
** Modified date:
** -----
*****/
extern "C" BOOL _stdcall as3992_Reset(void);

```

4.1.5 Reboot Reader

This command can reboot reader module

```

/*****
** Function name:      as3992_Reboot
** Descriptions:  This function use to reboot the reader

** input parameters:  none
** output parameters:  none
** Returned value:    TRUE : Success
                     FALSE : Failed

** Created by:      Wanwu
** Created date:    2013-3-22
** -----
** Modified by:
** Modified date:
** -----
*****/
extern "C" BOOL _stdcall as3992_Reboot(void);

```

4.1.6 Inventory tags

```
/* Tag data struct type */
typedef struct
{
    unsigned int    tagIndex;           /* record for the row of tag table */
    unsigned char   rssi;               /* receive rssi */
    unsigned int    freq;               /* receive frequency */
    unsigned short  reflectPower;       /* reflected power */
    unsigned char   data[TAG_DATA_SIZE]; /* tag data area */
    unsigned char   recvLen;            /* tag data length */
    unsigned int    recvCount;          /* tag received count */
    unsigned int    readerAddr;         /* reader addr */
}TagDataStructType;
```

- tagIndex : use to record tags amount
- rssi : tags received signal strength
- freq : tags receiving frequency points
- reflectPower : tags reflect power
- data : data of tags' PC and EPC
- recvLen : tags' receiving data length
- recvCount : tags' receiving counting
- readerAddr : reader address receiving tag

```
/******
** Function name:      as3992_inventory
** Descriptions:      Inventory tag.
** input parameters:  ptagData      : Point to DataStructType
                      TagDataStructType : Callback function
** output parameters: none
** Returned value:    none.

** Created by:        Herren
** Created date:      2013/01/08
** -----
** -----
******/
extern "C" void _stdcall as3992_inventory(TagDataStructType *ptagData, void
(*callBack)(TagDataStructType *tag, unsigned short tagCount));
```

4.1.7 Stop inventory

```
/******
** Function name:      as3992_stop_inventory
** Descriptions:      Stop inventory tag.
** input parameters:   none
** output parameters:  none
** Returned value:     none

** Created by:         Wanwu
** Created date:       2013/03/23
** -----
** Modified by:
** Modified date:

** -----
******/
extern "C" BOOL _stdcall as3992_stop_inventory(void);
```

4.1.8 Select tag

Structure body:

From header file: rfid_structs.h

```
/******  
 * Name: RFID_18K6C_SELECT_MASK - The select mask for partitioning a tag  
 * population  
 *****/  
enum  
{  
    RFID_18K6C_SELECT_MASK_BYTE_LEN = 32  
};  
typedef struct  
{  
    /* The memory bank to match against */  
    RFID_18K6C_MEMORY_BANK bank;  
    /* The offset of the first bit to match */  
    INT32U offset;  
    /* The number of bits in the mask */  
    INT32U count;  
    /* The bit pattern to match. */  
    INT8U mask[RFID_18K6C_SELECT_MASK_BYTE_LEN];  
} RFID_18K6C_SELECT_MASK;  
  
/******  
 * Name: RFID_18K6C_SELECT_ACTION - The matching and non-matching action to  
 * take when a selection mask matches/doesn't match  
 *****/  
typedef struct  
{  
    /* What will be affected by the action */  
    RFID_18K6C_TARGET target;  
    /* The actions to be performed on the tag populations (i.e., matching and */  
    /* non-matching. */  
    RFID_18K6C_ACTION action;  
    /* Should the EPC be truncated when the tag is singulated? A non-zero */  
    /* value requestes that the EPC is truncated. A zero value requests the */  
    /* entire EPC. */  
    BOOL32 enableTruncate;  
} RFID_18K6C_SELECT_ACTION;  
  
/******  
 * Name: RFID_18K6C_SELECT_CRITERION - A single selection criterion - i.e.,  
 * combination of selection mask and action.  
 *****/  
typedef struct {  
    /* The selection mask to test for */  
    RFID_18K6C_SELECT_MASK mask;  
    /* The actions to perform */  
    RFID_18K6C_SELECT_ACTION action;  
} RFID_18K6C_SELECT_CRITERION;
```

When user using, just need to write 12 bytes EPC data into element of mask in RFID_18K6C_SELECT_MASK, then invoke this function to realize function of selecting tag

```
/******  
** Function name:      as3992_select  
** Descriptions:      select the tag  
*****
```

```

** input parameters:    select : select parameter
** output parameters:  none
** Returned value:     none

** Created by:         Heiren
** Created date:       2013/01/08
** -----
** Modified by:
** Modified date:
** -----
*****/
extern "C" BOOL _stdcall as3992_select(RFID_18K6C_SELECT_CRITERION select);

```

4.1.9 Read tag's memory block data

This function is used to read data of tag's memory block

```

Bank          : tag's memory area selection
                0 stands for Reserved area(reserved area/password area)
                1 stands for EPC area
                2 stands for TID area
                3 stands for USER area
Offset        : offset in selected area, range from 0 to its max address
Count         : counts to be read, please note here take Byte (2 bytes) as a unit,
when it's 0, it's the max byte data to be read
Pwd           : read password(default as 0)
pTagData      : point to return structure body of tag data
/*****
** Function name:    as3992_read
** Descriptions:    read tag data.

** input parameters: bank    : access bank register
                    offset : access pointer register, offset
                    count  : access count register
                    pwd     : access passwaord, 0 if not password protected
                    pTagData : pointer to tag data stuct

** output parameters: none
** Returned value:    Return read length

** Created by:       HeiRen
** Created date:     2013/01/08
** -----
** Modified by:
** Modified date:
** -----
*****/
extern "C" BYTE _stdcall as3992_read(UINT bank, UINT offset, UINT count, UINT pwd,
                                     TagDataStructType *pTagData);
Return length read of this operation

```

4.1.10 Write data into tag's memory area

This function is to write data into tag's memory area

```

Bank          : tag's memory area selection
                0 stands for Reserved area(reserved area/password area)
                1 stands for EPC area
                2 stands for TID area(TID area cannot be written)
                3 stands for USER area

```

```

Offset      : offset in selected area, range from 0 to its max address
Count       : counts to be written, please note here take byte(2 bytes) as a unit
Pwd         : write password(default 0)
pTagData    : point to return structure body of tag's data
/*****
** Function name:      indy_write
** Descriptions:      This function writes data to the specified bank.

** input parameters:  bank      : access bank register
                      offset   : access pointer register, offset
                      count    : access count register
                      data     : data to write
                      pwd      : access password, 0 if not password protected

** output parameters: none
** Returned value:    none

** Created by:        HeiRen
** Created date:      2013/01/08
** -----
** Modified by:
** Modified date:
** -----
*****/
extern "C" BYTE _stdcall as3992_write(BYTE bank, BYTE offset, BYTE count,
                                     UINT16 *data, UINT pwd);

```

4.1.11 Lock tag

This function is used to lock designated memory area of tag

```

action      :lock action
Mask        :lock mask
pwd         :lock password

/*****
** Function name:      as3992_lock
** Descriptions:      This function locks a specified memory location.
** input parameters:  action  : lock action
                      mask    : lock mask
                      pwd     : access password

** output parameters: none
** Returned value:    none

** Created by:        HeiRen
** Created date:      2013/01/08
** -----
** Modified by:
** Modified date:
** -----
*****/
extern "C" int _stdcall as3992_lock(INT32U action, INT32U mask, INT32U pwd);

```

4.1.12 Kill tag

This function is used to kill tag, kill password is needed

```
/******  
** Function name:      as3992_kill  
** Descriptions:      This function kill the tag.  
  
** input parameters:   pwd      : kill password  
  
** output parameters:  none  
** Returned value:     TRUE   : Success  
                     FALSE   : Failed  
  
** Created by:        HeiRen  
** Created date:      2013/01/08  
** -----  
** Modified by:  
** Modified date:  
** -----  
*****/  
extern "C" int _stdcall as3992_kill(INT32U pwd);
```

4.2 Indy library function interface description

4.2.1 connect reader

```
/******  
** Function name:      indy_connect  
** Descriptions:      This function use to connect the reader  
  
** input parameters:   readerInfo : Point to the reader information struct  
** output parameters:  none  
** Returned value:     none  
  
** Created by:        Wangwu  
** Created date:      2013-3-22  
** -----  
** Modified by:  
** Modified date:  
** -----  
*****/  
extern "C" BOOL _stdcall indy_connect(void *readerInfo);
```

4.2.2 disconnect reader

```
/******  
** Function name:      indy_disConnect  
** Descriptions:      This function use to disconnect the reader  
  
** input parameters:   none  
** output parameters:  none  
** Returned value:     none  
  
** Created by:        Wangwu  
** Created date:      2013-3-22  
** -----
```

```

** Modified by:
** Modified date:
** -----
*****/
extern "C" BOOL _stdcall indy_disconnect(void);

```

4.2.3 inventory tags

```

/*****
** Function name:      indy_inventory
** Descriptions:      inventory tag.

** input parameters:   ptagData      : the pointer to store tags
                      callBack      : call back function
** output parameters:  none
** Returned value:     none

** Created by:        Wangwu
** Created date:      2013/01/08
** -----
** Modified by:
** Modified date:
** -----
*****/
extern "C" void _stdcall indy_inventory(TagDataStructType *ptagData, void
(*callBack)(TagDataStructType *tag, unsigned short tagCount));

```

4.2.4 select tag

```

/*****
** Function name:      indy_select
** Descriptions:      select the tag

** input parameters:   select : select parameter
** output parameters:  none
** Returned value:     none

** Created by:        Wangwu
** Created date:      2013/01/08
** -----
** Modified by:
** Modified date:
** -----
*****/
extern "C" BOOL _stdcall indy_select(RFID_18K6C_SELECT_CRITERION select);

```

4.2.5 read data of tag's memory area

```

/*****
** Function name:      indy_read
** Descriptions:      read tag data.

** input parameters:   bank      : access bank register
                      offset    : access pointer register, offset
                      count    : access count register

```

```

        pwd          : access password, 0 if not password protected
        pTagData      : pointer to tag data struct

** output parameters:   none
** Returned value:      none

** Created by:          Wangwu
** Created date:        2013/01/08
** -----
** Modified by:
** Modified date:
** -----
*****/
extern "C" unsigned char _stdcall indy_read(unsigned short bank, unsigned short offset,
        unsigned short count, unsigned int pwd, TagDataStructType
        *pTagData);

```

4.2.6 write data into tag's memory area

```

/*****
** Function name:      indy_write
** Descriptions:       This function writes data to the specified bank.

** input parameters:   bank    : access bank register
                        offset  : access pointer register, offset
                        count   : access count register
                        data    : data to write
                        Pwd     : access password, 0 if not password protected

** output parameters:   none
** Returned value:      none

** Created by:          Wangwu
** Created date:        2013/01/08
** -----
** Modified by:
** Modified date:
** -----
*****/
extern "C" unsigned char _stdcall indy_write(unsigned short bank, unsigned short offset,
        unsigned short count, unsigned short *data, unsigned int pwd);

```

4.2.7 lock tag's memory area

```

/*****
** Function name:      indy_lock
** Descriptions:       This function locks a specified memory location.

** input parameters:   action   : lock action
                        mask    : lock mask
                        pwd      : access password

** output parameters:   none
** Returned value:      none

** Created by:          Wangwu
** Created date:        2013/01/08
** -----
** Modified by:

```

```

** Modified date:
** -----
*****/
extern "C" int _stdcall indy_lock(INT32U action, INT32U mask, INT32U pwd);

```

4.2.8 get serial number of module

```

/*****
** Function name:      getSerialNumber
** Descriptions:      This function synchronizes the host with the Indy reader by
                      attempting to find a valid serial number from the Indy reader.
** input parameters:  serNum : pointer to buffer that will hold the Indy serial number

** output parameters:  none
** Returned value:     Length of serial number, if 0 = no serial number found

** Created by:        Wangwu
** Created date:      2013/01/08
** -----
** Modified by:
** Modified date:
** -----
*****/
extern "C" unsigned short _stdcall getSerialNumber(unsigned char *serNum);

```

4.2.9 Cancel current operation

```

/*****
** Function name:      indy_cancel
** Descriptions:      Cancel current operation.

** input parameters:  none
** output parameters:  none
** Returned value:     none

** Created by:        Wangwu
** Created date:      2013/01/08
** -----
** Modified by:
** Modified date:
** -----
*****/
extern "C" void _stdcall indy_cancel();

```

4.2.10 reset module

```

/*****
** Function name:      indy_reset
** Descriptions:      Reset indy chip.

** input parameters:  none
** output parameters:  none
** Returned value:     none

** Created by:        Wangwu
** Created date:      2013/01/08
** -----
** Modified by:
** Modified date:

```

```

** -----
*****/
extern "C" void _stdcall indy_reset();

```

4.2.11 abort module

```

/*****
** Function name:      indy_abort
** Descriptions:      Reset indy chip.

** input parameters:   none
** output parameters:  none
** Returned value:     none

** Created by:        Wangwu
** Created date:      2013/01/08
** -----
** Modified by:
** Modified date:
** -----
*****/
extern "C" void _stdcall indy_abort();

```

4.2.12 kill tag

```

/*****
** Function name:      indy_kill
** Descriptions:      This function kill the tag.

** input parameters:   pwd          : kill password

** output parameters:  none
** Returned value:     none

** Created by:        Wangwu
** Created date:      2013/01/08
** -----
** Modified by:
** Modified date:
** -----
*****/
extern "C" int _stdcall indy_kill(INT32U pwd);

```

4.3 AS3992 (Ethernet) library function interface description

Note: there are two ways to use Ethernet port communicate with module, one is Server mode, another is Client mode. If there is no special remind in following functions, that means this function's using are the same on both modes.

4.3.1 set event notice callback function

Note : before starting server or act as Client side connecting to server, to use this function in advance, when there is any event happen, it will notice main function response in time

Callback function definition:

```
typedef void (*NotifyCallBack)(READER_NET_CALLBACK_INFO);
```

Structure body definition

Structural body in file:MultiDll.h

typedef struct

```
{
    Int work_mode;           \\mode:
                             \\0: Sever mode
                             \\1: Client mode

    int client_status;       \\ when working in Client mode, it stands for the status of
    Local Client side
                             \\1: already connected to remote server
                             \\0: already disconnected with remote server

    int server_status;       \\ when working in Server mode, it stands for the status of
                             remote Client side
                             \\0: having remote Client connect to local server
                             \\1: remote Client side disconnect with local server

    char client_to_server_ip[20]; \\ when working in Server mode, it stands for remote
    Client's IP

    unsigned short client_to_server_port; \\ when working in Server mode, it stands for port
    number of remote Client port
}
```

}READER_NET_CALLBACK_INFO;

/******

```
** Function name:      as3992Tcp_SetNotifyCallBack
** Descriptions:      This function use to set the notify callback function address
** input parameters:  NONE
** output parameters: NONE
** Returned value:    NONE
```

```
** Created by:      HeiRen
** Created date:    2013/07/01
```

```
** -----
** Modified by:
** Modified date:
** -----
```

*****/

```
extern "C" void __stdcall as3992Tcp_SetNotifyCallBack(NotifyCallBack func)
```

4.3.2 Start Sever

Note: This function is only used in Server mode

Structure body:

Structural body in file:MultiDll.h

typedef struct

```
{
    char *ip;           local IP address
    unsigned short port; Local Server monitor port
}
```

}READER_NET_INFO_TYPE;

/******

```
** Function name:      as3992Tcp_StartService
** Descriptions:      This function use to start service to listen the client (only in the service
                       mode)
** input parameters:  readerInfo: include the ip and port info
** output parameters: NONE
```

```

** Returned value:      NONE
** Created by:          HeiRen
** Created date:        2013/07/01
** -----
** Modified by:
** Modified date:
** -----
*****/
extern "C" BOOL __stdcall as3992Tcp_StartService(void *readerInfo)

```

4.3.2 Stop service

Note: this function is only used in Server mode

```

/*****
** Function name:      as3992Tcp_StopService()
** Descriptions:      This function use to stop service (only in the service mode)
** input parameters:   NONE
** output parameters:  NONE
** Returned value:     NONE
** Created by:        HeiRen
** Created date:      2013/07/01
** -----
** Modified by:
** Modified date:
** -----
*****/
extern "C" void __stdcall as3992Tcp_StopService()

```

4.3.3 Connect Reader

Note: this function only used in Client mode

Structure body:

Structural body in file:MultiDll.h

typedef struct

```

{
    char *ip;           target Server IP address
    unsigned short port; target Server monitor port
}READER_NET_INFO_TYPE;

```

```

/*****
** Function name:      as3992Tcp_connect
** Descriptions:      This function use to connect the reader (only in tcp client mode)
** input parameters:   readerInfo      : the type is READER_NET_INFO_TYPE

** output parameters:  none
** Returned value:     0 : fail
                     1: success
** Created by:        HeiRen
** Created date:      2013 07/01
** -----
** Modified by:
** Modified date:
** -----
*****/
extern "C" BOOL __stdcall as3992Tcp_connect(void *readerInfo)

```

4.3.4 Disconnect Reader

Note: this function is used only in Client mode

```
/******  
** Function name:      as3992Tcp_disConnect  
** Descriptions:      This function use to disconnect the reader (only in tcp client mode)  
** input parameters:   none  
** output parameters:  none  
** Returned value:     none  
  
** Created by:         HeiRen  
** Created date:       2013-7-1  
** -----  
** Modified by:  
** Modified date:  
** -----  
*****/  
extern "C" BOOL __stdcall as3992Tcp_disConnect(void)
```

4.3.5 Get version of the reader

```
/******  
** Function name:      as3992Tcp_Getversion  
** Descriptions:      This function use to get the reader version  
  
** input parameters:   *hardwareVersion , *softwareVersion  
** output parameters:  *hardwareVersion , *softwareVersion  
** Returned value:     TRUE : Success  
                      FALSE : Failed  
  
** Created by:         HeiRen  
** Created date:       2013 7-1  
** -----  
** Modified by:  
** Modified date:  
** -----  
*****/  
extern "C" BOOL __stdcall as3992Tcp_Getversion(unsigned char * hardwareVersion ,  
unsigned char * *softwareVersion)  
After function invoked successfully , then store software and hardware version information in  
array, the capacity of the array is over 30 bytes.
```

4.3.6 Set Reader

Structure definition:

Structural body in file:MultiDll.h

```
/* Config type define */  
typedef enum  
{  
    SET_FREQUENCY      = 0,  
    SET_LINK_FREQ,  
    SET_CODING,  
    SET_SESSION,  
    SET_TREXT,  
    SET_Q_VALUE,  
    SET_SENSITIVITY,
```

```

        SET_RF_POWER,
        SET_GEN2,
        SET_MODE,
    }UHF_CONFIG_TYPE;
    ● SET_FREQUENCY: set frequency point
    ● SET_LINK_FREQ: set link frequency
    ● SET_CODING: set coding way
    ● SET_SESSION: set session mode
    ● SET_TREXT: set precursor adjustment
    ● SET_Q_VALUE: set Q value
    ● SET_SENSITIVITY: set sensitivity
    ● SET_RF_POWER: set RF power
    ● SET_GEN2: set GEN2 parameter
    ● SET_MODE: set reader mode

```

Structural body in file:MultiDll.h

```

/* UHF configure type */
typedef struct
{
    unsigned char    mode;           /* Reader mode */
    unsigned int     startFreq;      /* Start frequency */
    unsigned int     endFreq;        /* End frequency */
    unsigned char    rssi_thresh;    /* RSSI threshold */
    unsigned int     freqIncrement;  /* Frequency increment */
    unsigned char    linkFreq;       /* Link frequency */
    unsigned char    coding;         /* coding type */
    unsigned char    session;        /* session */
    unsigned char    trext;          /* trext */
    unsigned char    q_Value;        /* Q value */
    unsigned char    sensitivity;     /* Sensitivity */
    unsigned char    rf_power;       /* rf power */
    unsigned char    last_error;     /* last error number */
}UHF_Configure;

```

Structure element definition:

```

mode           : reader working mode, 0x00 standard mode, 0x01 auto-adjust mode
startFreq      : start frequency, for example 922875 stands 922.875 MHz
endFreq        : end frequency
freqIncrement   : frequency increment
rssi_thresh    :RSSI signal threshold value (range from -83 ~ 0)
linkFreq: link frequency
               0x00  40 kHz
               0x06  160 kHz
               0x08  213 kHz
               0x09  256 kHz
               0x0c  320 kHz
               0x0f  640 kHz
Coding         :coding way
               0x00  FM0 coding for rx
               0x01  MILLER2 coding for rx
               0x02  MILLER4 coding for rx
               0x03  MILLER8 coding for rx
Session        :session selection
               0x00  Inventoried (S0)
               0x01  Inventoried (S1)
               0x02  Inventoried (S2)
               0x03  Inventoried (S3)

Trest          : preamble pre-pilot tone selection, default 0x01

```

0x00 preamble without pilot tone
 0x01 preamble adding pilot tone

gen2qbegins : inventory tag's Q value selection, range from 0~15, default 4

Sensitivity :reader receive sensitivity set negative value as type of signed char (with sign character), the best receive sensitivity is -84, that is 0xAC, default 0xAC.

rf_power : set rf power, range from 0~19, in correspond to 27~8dbm, 0 correspond to max output power, 19 correspond to min power

last_error : last error caused when setting parameter

```

/*****
** Function name:      as3992Tcp_config
** Descriptions:      This function use to config the reader

** input parameters:  config : Point to the reader configure struct
                      type  : Config type, see UHF_CONFIG_TYPE definition
** output parameters: none
** Returned value:    TRUE  : Success
                      FALSE  : Failed

** Created by:        HeiRen
** Created date:      2013/07/01
** -----
** Modified by:
** Modified date:
** -----
*****/
extern "C" BOOL _stdcall as3992Tcp_config(UHF_Configure *config, UHF_CONFIG_TYPE
type);
Return TRUE when setup success ;
Return FALSE when setup failed ;

```

4.3.7 Reset reader

This function is to reset the reader parameters to default value, it's used when user wrong-set parameter and refresh to default status

```

/***** Function
name:      as3992Tcp_Reset
** Descriptions: This function use to reset the reader to the default parameters

** input parameters:  none
** output parameters: none
** Returned value:    TRUE  : Success
                      FALSE  : Failed

** Created by:        HeiRen
** Created date:      2013/07/01
** -----
** Modified by:
** Modified date:
** -----
*****/
extern "C" BOOL _stdcall as3992Tcp_Reset(void);

```

4.3.8 Reboot reader

This function can be used to reboot reader module

```
/******  
** Function name:      as3992Tcp_Reboot  
** Descriptions:      This function use to reboot the reader  
  
** input parameters:   none  
** output parameters:  none  
** Returned value:     TRUE  : Success  
                      FALSE   : Failed  
  
** Created by:        HeiRen  
** Created date:      2013/07/01  
** -----  
** Modified by:  
** Modified date:  
** -----  
*****/  
extern "C" BOOL _stdcall as3992Tcp_Reboot(void);
```

4.3.9 Inventory tags

```
/* Tag data struct type */  
typedef struct  
{  
    unsigned int    tagIndex;           /* record for the row of tag table */  
    unsigned char   rssi;               /* receive rssi */  
    unsigned int    freq;               /* receive frequency */  
    unsigned short  reflectPower;       /* reflected power */  
    unsigned char   data[TAG_DATA_SIZE]; /* tag data area */  
    unsigned char   recvLen;            /* tag data length */  
    unsigned int    recvCount;          /* tag received count */  
    unsigned int    readerAddr;         /* reader addr */  
}TagDataStructType;
```

- tagIndex : to record tags counting
- rssi : tags received signal strength
- freq : tag's received frequency point
- reflectPower : tag reflect power
- data : tags PC and EPC data
- recvLen : tag received data length
- recvCount : tag received count
- readerAddr : reader address receive this tag

```
/******  
** Function name:      as3992Tcp_inventory  
** Descriptions:      Inventory tag.  
** input parameters:   ptagData : Point to DataStructType  
                      TagDataStructType : Callback function  
** output parameters:  none  
** Returned value:     none.  
  
** Created by:        HeiRen  
** Created date:      2013/07/01  
** -----  
*****/  
extern "C" void _stdcall as3992Tcp_inventory(TagDataStructType *ptagData, void
```

```
(*callBack)(TagDataStructType *tag, unsigned short tagCount));
```

4.3.10 Stop inventory

```
/******  
** Function name:      as3992Tcp_stop_inventory  
** Descriptions:      Stop inventory tag.  
** input parameters:   none  
** output parameters:  none  
** Returned value:     none  
  
** Created by:        HeiRen  
** Created date:       2013/07/01  
** -----  
** Modified by:  
** Modified date:  
  
** -----  
*****/  
extern "C" BOOL      _stdcall as3992Tcp_stop_inventory(void);
```

4.3.11 Select tag

Structure body:

From header file: rfid_structs.h

```
/******  
* Name: RFID_18K6C_SELECT_MASK - The select mask for partitioning a tag  
* population  
*****/  
enum  
{  
    RFID_18K6C_SELECT_MASK_BYTE_LEN = 32  
};  
typedef struct  
{  
    /* The memory bank to match against */  
    RFID_18K6C_MEMORY_BANK bank;  
    /* The offset of the first bit to match */  
    INT32U offset;  
    /* The number of bits in the mask */  
    INT32U count;  
    /* The bit pattern to match. */  
    INT8U mask[RFID_18K6C_SELECT_MASK_BYTE_LEN];  
} RFID_18K6C_SELECT_MASK;  
  
/******  
* Name: RFID_18K6C_SELECT_ACTION - The matching and non-matching action to  
* take when a selection mask matches/doesn't match  
*****/  
typedef struct  
{  
    /* What will be affected by the action */  
    RFID_18K6C_TARGET target;  
    /* The actions to be performed on the tag populations (i.e., matching and */  
    /* non-matching. */  
    RFID_18K6C_ACTION action;  
    /* Should the EPC be truncated when the tag is singulated? A non-zero */  
    /* value requestes that the EPC is truncated. A zero value requestes the */
```

```

    /* entire EPC.                                     */
    BOOL32      enableTruncate;
} RFID_18K6C_SELECT_ACTION;

/*****
 * Name: RFID_18K6C_SELECT_CRITERION - A single selection criterion - i.e.,
 *       combination of selection mask and action.
 *****/
typedef struct {
    /* The selection mask to test for                                     */
    RFID_18K6C_SELECT_MASK  mask;
    /* The actions to perform                                           */
    RFID_18K6C_SELECT_ACTION action;
} RFID_18K6C_SELECT_CRITERION;

```

When using it, user just need to write 12 bytes EPC into mask of RFID_18K6C_SELECT_MASK, then invoke this function to realize Select tags

```

/*****
** Function name:      as3992Tcp_select
** Descriptions:      select the tag

** input parameters:   select : select parameter
** output parameters:  none
** Returned value:     none

** Created by:        HeiRen
** Created date:      2013/07/01
** -----
** Modified by:
** Modified date:
** -----
 *****/
extern "C" BOOL _stdcall as3992Tcp_select(RFID_18K6C_SELECT_CRITERION select);

```

4.3.12 Get tag's memory area data

This function is to get the data of memory area of tags

```

Bank          : tag's memory area selection
                0 stands Reserved area(reserved area/password area)
                1 stands EPC area
                2 stands TID area
                3 stands USER area

Offset        : offset in selected area, range from 0 to its max address
Count         : words counts to be read, please note here take word(2 bytes) as a
unit, when it is 0, it's to get the max byte data in this area
Pwd           : get password (default as 0)
pTagData      : point to returned structure body of tags data

```

```

/*****
** Function name:      as3992Tcp_read
** Descriptions:      read tag data.

** input parameters:   bank      : access bank register
                        offset    : access pointer register, offset
                        count     : access count register
                        pwd       : access password, 0 if not password protected
                        pTagData  : pointer to tag data struct
 *****/

```

```

** output parameters:    none
** Returned value:      Return read length

** Created by:          HeiRen
** Created date:        2013/07/01
** -----
** Modified by:
** Modified date:
** -----
*****/
extern "C" BYTE _stdcall as3992Tcp_read(UINT bank, UINT offset, UINT count, UINT pwd,
                                         TagDataStructType *pTagData);
Reading range of this operation

```

4.3.12 Write data into memory area

This function is to write data into tag's memory area

```

Bank          : tag's memory area selection
                0 stands Reserved area(reserved area/password are)
                1 stands EPC area
                2 stands TID area (TID area cannot be written)
                3 stands USER area
Offset        : offset of selected area, range from 0 to its max address
Count         : byte counts, please note here take byte (2 bytes) as a unit
Pwd           : password to be written(default as 0)
pTagData      : point to returned structure of tag's data
/*****
** Function name:    as3992Tcp_write
** Descriptions:    This function writes data to the specified bank.

** input parameters: bank    : access bank register
                    offset : access pointer register, offset
                    count  : access count register
                    data   : data to write
                    pwd     : access passwaord, 0 if not password protected

** output parameters: none
** Returned value:    none

** Created by:      HeiRen
** Created date:    2013/07/01
** -----
** Modified by:
** Modified date:
** -----
*****/
extern "C" BYTE _stdcall as3992Tcp_write(BYTE bank, BYTE offset, BYTE count,
                                         UINT16 *data, UINT pwd);

```

4.3.13 Lock tag

This function is to lock assigned memory area of tag

action : Lock action

Mask : lock mask
pwd : lock password

```

/*****
** Function name:      as3992Tcp_lock
** Descriptions:      This function locks a specified memory location.
** input parameters:  action : lock action
                      mask  : lock mask
                      pwd    : access passwaord

** output parameters: none
** Returned value:    none

** Created by:        HeiRen
** Created date:      2013/07/01
** -----
** Modified by:
** Modified date:
** -----
*****/
extern "C" int _stdcall as3992Tcp_lock(INT32U action, INT32U mask, INT32U pwd);

```

4.3.14 Kill tag

This function is to kill tags, kill password of this tag is needed

```

/*****
** Function name:      as3992Tcp_kill
** Descriptions:      This function kill the tag.

** input parameters:  pwd      : kill passwaord

** output parameters: none
** Returned value:    TRUE  : Success
                      FALSE : Failed

** Created by:        HeiRen
** Created date:      2013/07/01
** -----
** Modified by:
** Modified date:
** -----
*****/
extern "C" int _stdcall as3992Tcp_kill(INT32U pwd);

```

All rights reserved and the content may be updated without notice!