

Further UI Software

Instructions

Issue: 1.0.0

Date: 2012.11.22

www.dokodemoterebi.com

About This Document

Purpose

This document describes the intended audience and document contents.

Intended Audience

This document is intended for the programmers who meet the following requirements:

- Be familiar with C/C++ programming language
- Be familiar with 32-bit Windows environment/ MFC

Organization

This document is organized as follows:

| Chapter | Content |
|------------------------|---|
| 1 Overview | Describes the three functions which are used to receive and process the return messages of network and AV commands. |
| 2 Function Invoking | Describes how to invoke the three functions to receive and process the return messages of network and AV commands. |
| 3 Function Description | Describes how to get the detailed return messages of network and AV commands, including the messages type, the structure and size of the additional data. |
| 4 Macro Definition | Describes the macro definition relating to the return messages of network and AV commands. |
| 5 Structure Definition | Describes the structure definition relating to the return additional data. |

1 Overview

This document mainly describes how to receive and process the return messages of network and AV commands.

UI uses the following three functions to process the return messages of network and AV commands:

```
static int WINAPI HandleCmdCallback(int Param1,int Param2,char *data,int dataSize);  
// Processes the return messages of network commands
```

```
static int WINAPI HandleAVofNetCallback(int Param1,int Param2,char *data,int dataSize);  
// Processes the AV return messages of network commands
```

```
static int WINAPI HandleAVCallback(int Param1,int Param2,char *data,int dataSize);  
// Processes the return messages of AV commands
```

2 Function Invoking

2.1 HandleCmdCallback function invoking

| Function invoking | Definition file | Example |
|---|------------------|---------------------------------------|
| <code>int AddUICallBack (PFCALLBACK Func);</code> | FG_NET_Manager.h | AddUICallBack (HandleCmdCallback); |

2.2 HandleAVofNetCallback function invoking

| Function invoking | Definition file | Example |
|---|------------------|---|
| <code>int AddUICallAVBack (PFCALLBACK Func);</code> | FG_NET_Manager.h | AddUICallAVBack (HandleAVofNetCallback); |

2.3 HandleAVCallback function invoking

| Function invoking | Definition file | Example |
|--|-----------------|----------------------------------|
| <code>int CallBackFunc (PFCALLBACK Func);</code> | FG_AV_Manager.h | CallBackFunc (HandleAVCallback); |

3 Function Description

3.1 HandleCmdCallback function

```
static int WINAPI HandleCmdCallback(int Param1,int Param2,char *data,int dataSize);
```

// Processes the return messages of network commands

Parameter:

| | |
|----------|---|
| Param1 | Return command ID, defined in the file Ft_Net_Global.h. |
| Param2 | Subcommand ID, defined in the file Ft_Net_Global.h. |
| Data | Return data of network commands |
| dataSize | Data size |

3.1.1 Login command

Param1: UI_LOGIN_CMD

| Param2 | Data | dataSize |
|----------------------|--------------------------------------|-------------------------------------|
| UI_LOGIN_CMD_SUCCESS | Messages of structure usermanager | Data size |
| UI_LOGIN_CMD_FAILED | NULL | The type of login failure, int type |

The type of login failure:

| | | |
|--|------|--|
| #define ERR0_NET_NULL | 0x00 | // Unkown login error |
| #define ERR1_NET_LOCAL_ERR | 0x01 | // Local network error |
| #define ERR2_NET_LOCAL_CONNECT_ERR | 0x02 | // Failed to connect to LAN |
| #define ERR3_NET_UPNP_SER_OUT | 0x03 | // Failed to connect to server |
| #define ERR4_NET_UPNP_BOX_OUT | 0x04 | // Failed to connect to Box |
| #define ERR5_NET_UPNP_FAIL | 0x05 | // UPnP connection failed |
| #define ERR6_NET_LOG_UPW | 0x06 | // User possword is wrong |
| #define ERR7_NET_LOG_NOR | 0x07 | // No response form login server |
| #define ERR8_NET_LOG_NOTSEND | 0x08 | // Failed to send login command |
| #define ERR9_NET_LOG_DN_ERR | 0x09 | // Domain name error |
| #define ERR10_NET_LOG_FULL_USER | 0x0A | // Online user currently full |
| #define ERR11_NET_LOG_DIF_CLIENT | 0x0B | // Different client can't login at the same time |
| #define ERR12_NET_LINK_RTSP_ERROR | 0x0C | // RTSP connection failed |
| #define ERR13_NET_LINK_USER_NO_LOG_REC | 0x0D | // Ordinary users can't login when recoding |

3.1.2 TV command

Includes scanning channel, getting channel table, uploading channel table and setting scanning country or region.

Param1: UI_TV_CMD

| Param2 | Data | dataSize |
|-----------------------------|---|--|
| UI_SET_CHANNEL_SUCCESS | NULL | NULL |
| UI_SET_CHANNEL_FAILED | NULL | NULL |
| UI_SCANCHN_START | NULL | NULL |
| UI_GET_A_CHANNEL | Messages of structure fcp_tvop | Data size |
| UI_SCAN_FINISH | NULL | NULL |
| UI_GET_FRQ_TABLE_SUCCESS | The first 32 letters are channel table name, other letters are information of each channel, its structure is tv_table. The number of this type structures is the same as the number of channel. | The returned 32 bit Long type after using "MAKELONG (WORD wLow, WORD wHigh);", wHigh means the total number of channels. |
| UI_GET_FRQ_TABLE_FAILED | NULL | NULL |
| UI_SET_COUNTRY_SUCCESS | NULL | NULL |
| UI_SET_COUNTRY_FAILED | NULL | NULL |
| UI_UPLOAD_FRQ_TABLE_SUCCESS | NULL | NULL |
| UI_UPLOAD_FRQ_TABLE_FAILED | NULL | NULL |

3.1.3 IR remote control command

Includes IR learning, uploading IR table and downloading IR table.

Param1: UI_RM_CMD

| Param2 | Data | dataSize |
|--------------------------------|---|--|
| UI_RM_LEARN_ONE | NULL | 0 or 1 0: Failed to learn 1: Learn successfully |
| UI_MSG_RM_FAILED | NULL | 219: Operation failed, insufficient permissions! Other: Failed to send network command! |
| UI_RM_GET_ALL_COD E_SUCCESS | IR table number (512 byte, the first byte means IR table number) IR table name (512 byte) Key information (36 keys in total, taking 36*512 byte) IR table name (512 byte) Key information (36 keys in total, taking 36*512 byte) IR table name (512 byte) Key information (36 keys in total, taking 36*512 byte) ... | Data size |
| UI_RM_GET_ALL_COD E_FAILED | NULL | 219: Operation failed, insufficient permissions! Other: Failed to download IR table! |
| UI_PROFILE_TOBOX_ SUCCESS | NULL | NULL |
| UI_PROFILE_TOBOX_ FAILED | NULL | 219: Operation failed, insufficient permissions! |

3.1.4 Account management command

Includes modifying password.

Param1: UI_ACCOUNT_CMD

| Param2 | Data | dataSize |
|----------------------------|------|----------|
| UI_MODIFY_USER_PAW_SUCCESS | NULL | NULL |
| UI_MODIFY_USER_PAW_FAILED | NULL | NULL |

3.1.5 Schedule recording command

Param1: UI_RC_CMD

| Param2 | Data | dataSize |
|-------------------------|---|---|
| UI_MODIFY_TTZ_SUCCESS | Value of char[32] type time string | Data size |
| UI_MODIFY_TTZ_FAILED | NULL | NULL |
| UI_UPDATE_ALL_SUCCESS | Data of structure RecordDataBlock | Data size |
| UI_UPDATE_ALL_FAILED | NULL | NULL |
| UI_ADD_TASK_SUCCESS | NULL | Schedule recording task ID (unsigned int type) |
| UI_ADD_TASK_FAILED | NULL | NULL |
| UI_DEL_ALL_TASK_SUCCESS | NULL | NULL |
| UI_DEL_TASK_SUCCESS | NULL | Schedule recording task ID (unsigned int type) |
| UI_DEL_TASK_FAILED | NULL | NULL |
| UI_MOD_TASK_SUCCESS | NULL | NULL |
| UI_MOD_TASK_FAILED | NULL | NULL |
| UI_SCHEDELE_WHEN_ONLINE | Information of structure RecordEntry | Data size |

3.1.6 Remote recorded files download command

Param1: UI_DL_CMD

| Param2 | Data | dataSize |
|---------------------------------|--|---|
| UI_UPDATE_REC_FILE_INFO_SUCCESS | "typedef vector<FILE_ALL> FILE_MANAGER;" defined in the file Ft_Net_Global.h | Data size |
| UI_UPDATE_REC_FILE_INFO_FAILED | NULL | Cause of login failure (int type): 1: There's no USB 2: There's no recorded file in the USB storage Other: Failed to download |
| UI_REC_FILE_DL_REQ_SUCCESS | NULL | File's size (int type) |
| UI_REC_FILE_DL_REQ_FAILED | NULL | Cause of login failure (int type): 1: Box is recording Other: Failed to download |
| UI_REC_FILE_DL_CANCEL_SUCCESS | NULL | NULL |
| UI_REC_FILE_DL_CANCEL_FAILED | NULL | NULL |
| UI_REC_FILE_DELETE_SUCCESS | NULL | NULL |
| UI_REC_FILE_DELETE_FAILED | NULL | NULL |

3.1.7 Recorded files playback command

Param1: UI_PB_CMD

| Param2 | Data | dataSize |
|---------------------------------|---|---|
| UI_REC_FILE_LIVE_REQ_SUCCESS | Information of structure FCP_RECLIVEREQ | Failure type: 0: Request failed 2: Request timed out |
| UI_REC_FILE_LIVE_REQ_FAILED | Type information of structure FCP_RECLIVEREQ | Failure type: 0: Request failed 2: Request timed out |
| UI_REC_FILE_PLAY_BEGIN_SUCCESS | NULL | Failure type: 0: Request failed 2: Request timed out |
| UI_REC_FILE_PLAY_BEGIN_FAILED | NULL | Failure type: 0: Request failed 2: Request timed out |
| UI_REC_FILE_PLAY_CANCEL_SUCCESS | NULL | Failure type: 0: Request failed 2: Request timed out |
| UI_REC_FILE_PLAY_CANCEL_FAILED | NULL | Failure type: 0: Request failed 2: Request timed out |
| UI_REC_FILE_PLAY_SEEK_SUCCESS | NULL | Request result: 1: successful 2: Timed out 3. Failed |
| UI_REC_FILE_PLAY_SEEK_FAILED | NULL | Request result: 1: successful 2: Timed out 3. Failed |

3.1.8 Ports setting command

Param1: UI_OTHER_CMD

| Param2 | Data | dataSize |
|--------------------------|------|----------|
| UI_SAVE_SET_PORT_SUCCESS | NULL | NULL |
| UI_SAVE_SET_PORT_FAILED | NULL | NULL |

3.1.9 Box network setting command

Param1: UI_OTHER_CMD

| Param2 | Data | dataSize |
|----------------------------|------|----------|
| UI_BOX_NETWORK_SET_SUCCESS | NULL | NULL |
| UI_BOX_NETWORK_SET_FAILED | NULL | NULL |

3.1.10 Visual Server update command

Param1: UI_OTHER_CMD

| Param2 | Data | dataSize |
|--------------------------|------|----------|
| UI_UPDATE_BOX_VS_SUCCESS | NULL | NULL |
| UI_UPDATE_BOX_VS_FAILED | NULL | NULL |

3.1.11 Video source switch command

Param1: UI_OTHER_CMD

| Param2 | Data | dataSize |
|--------------------------|------|---|
| UI_SOURCE_SWITCH_SUCCESS | NULL | NULL |
| UI_SOURCE_SWITCH_FAILED | NULL | Failure cause: 1: The destination source is recording. 2: The destination source is busy. |

3.1.12 Video parameter setting command

Param1: UI_OTHER_CMD

| Param2 | Data | dataSize |
|-----------------------------|------|----------|
| UI_SET_VIDEO_FORMAT_SUCCESS | NULL | NULL |
| UI_SET_VIDEO_FORMAT_FAILED | NULL | NULL |

3.1.13 Box version information getting command

Param1: UI_OTHER_CMD

| Param2 | Data | dataSize |
|-------------------------|---|----------|
| UI_GET_BOX_INFO_SUCCESS | Information of structure BOX_INFO_S, defined in the file Ft_Net_Global.h | NULL |
| UI_GET_BOX_INFO_FAILED | NULL | NULL |

3.1.14 Notify message of network

Param1: UI_NET_MSG

| Param2 | Data | dataSize |
|--------------------------------|---------------------------------|---|
| UI_NET_NOTIFY_UI_SRC_SET_OK | NULL | Video source (int type): 0:CV; 1:SV; 2:TV; 3:CAM |
| UI_NET_NOTIFY_UI_CHN_SET_OK | NULL | Channel ID (int type) |
| UI_NET_NOTIFY_BOX_LINK_TIMEOUT | NULL | NULL |
| UI_NET_NOTIFY_UI_BITRATE | NULL | The current bitrate (int type) |
| UI_NET_NOTIFY_UI_SET_CMD_PORT | NULL | Command port (UINT type) |
| UI_NET_NOTIFY_UI_SET_AV_PORT | NULL | AV port (UINT type) |
| UI_NET_NOTIFY_UI_SET_RTSP_PORT | NULL | RTSP port (UINT type) |
| UI_NET_NOTIFY_UI_SET_USER_MASK | NULL | User type (UINT type): 0: Manager, can use all functions 1: Ordinary user, only can watch and has no operating authority |
| UI_NET_NOTIFY_UI_RES_SET_OK | NULL | Resolution (UINT type): 1: CIF; 2: H-D1; 4: D1; |
| UI_NET_NOTIFY_UI_SET_USER_TYPE | NULL | User type (int type): 0: Manager, can use all functions 1: Ordinary user, only can watch and has no operating authority |
| UI_NET_NOTIFY_UI_FILE_DL_START | Name of the string type file | Data size |
| UI_NET_NOTIFY_UI_FILE_DL_DATA | NULL | Size of the received file |
| UI_NET_NOTIFY_UI_FILE_DL_END | File data | Data size |

3.2 HandleAVofNetCallback function

```
static int WINAPI HandleAVofNetCallback(int Param1,int Param2,char *data,int dataSize);  
// Processes the AV return messages of network commands
```

Parameter:

| | |
|----------|---|
| Param1 | Return command ID, defined in the file Ft_Net_Global.h. |
| Param2 | Subcommand ID, defined in the file Ft_Net_Global.h. |
| Data | Return data of network commands. |
| dataSize | Data size |

3.2.1 AV return messages of network commands

Param1: UI_AV_WRITERB

| Param2 | Data | dataSize |
|-----------------------|------------|-----------|
| UI_AV_AUDIO_WRITE_BUF | Audio data | Data size |
| UI_AV_VIDEO_WRITE_BUF | Video data | Data size |

3.3 HandleAVCallback function

```
static int WINAPI HandleAVCallback(int Param1,int Param2,char *data,int dataSize);  
// Processes the return messages of AV commands
```

Parameter:

| | |
|----------|---|
| Param1 | Return command ID, defined in the file Ft_Net_Global.h. |
| Param2 | Subcommand ID, defined in the file Ft_Net_Global.h. |
| Data | Return messages of AV commands. |
| dataSize | Data size |

3.3.1 Not enough recording space message

Param1: ID_EVENT_NO_ENOUGH_SPACE

| Param2 | Data | dataSize |
|--------|------|----------|
| 0 | NULL | 0 |

3.3.2 Graph reconnection message

Param1: ID_EVENT_RECONNECTFILTERS

| Param2 | Data | dataSize |
|--------|------|----------|
| 0 | NULL | 0 |

3.3.3 Recorded files playback finished message

Param1: ID_EVENT_FILEFINISH

| Param2 | Data | dataSize |
|--------|------|----------|
| 0 | NULL | 0 |

3.3.4 Remote recorded files playback finished message

Param1: FG_UI_MSG_FILE_LIVE_DECODE_OVER

| Param2 | Data | dataSize |
|--------|------|----------|
| 0 | NULL | 0 |

4 Macro Definition

```
#define UI_LOGIN_CMD          0x01000000    // Login command and its return messages
//----- Return messages of login command -----
#define UI_LOGIN_CMD_SUCCESS      UI_LOGIN_CMD | 0X00000001 // Successful
#define UI_LOGIN_CMD_FAILED      UI_LOGIN_CMD | 0X00000002 // Failed

#define UI_TV_CMD          0x02000000    // TV command
//----- Return messages of TV command -----
#define UI_SET_CHANNEL_SUCCESS      UI_TV_CMD | 0X00000001 // Successful
#define UI_SET_CHANNEL_FAILED      UI_TV_CMD | 0X00000002 // Failed
#define UI_SCANCHN_START          UI_TV_CMD | 0X00000005
#define UI_GET_A_CHANNEL          UI_TV_CMD | 0X00000006
#define UI_SCAN_FINISH            UI_TV_CMD | 0X00000007

#define UI_GET_FRQ_TABLE_SUCCESS    UI_TV_CMD | 0X00000008 // Successful
#define UI_GET_FRQ_TABLE_FAILED    UI_TV_CMD | 0X00000009 // Failed

#define UI_SET_COUNTRY_SUCCESS      UI_TV_CMD | 0X0000000A // Successful
#define UI_SET_COUNTRY_FAILED      UI_TV_CMD | 0X0000000B // Failed

#define UI_UPLOAD_FRQ_TABLE_SUCCESS UI_TV_CMD | 0X0000000C // Successful
#define UI_UPLOAD_FRQ_TABLE_FAILED UI_TV_CMD | 0X0000000D // Failed

#define UI_RM_CMD          0x03000000    // IR learning command
//----- Return messages of IR learning command -----
#define UI_RM_LEARN_ONE          UI_RM_CMD | 0X00000001 // Successful
#define UI_MSG_RM_FAILED          UI_RM_CMD | 0X00000002 // Failed

#define UI_RM_GET_ALL_CODE_SUCCESS UI_RM_CMD | 0X00000003 // Successful
#define UI_RM_GET_ALL_CODE_FAILED UI_RM_CMD | 0X00000004 // Failed

#define UI_PROFILE_TOBOX_SUCCESS    UI_RM_CMD | 0X00000005 // Successful
#define UI_PROFILE_TOBOX_FAILED    UI_RM_CMD | 0X00000006 // Failed

#define UI_ACCOUNT_CMD          0x04000000    // User management command
//----- Return messages of user management command -----
#define UI_MODIFY_USER_PAW_SUCCESS    UI_ACCOUNT_CMD | 0X00000001 // Successful
#define UI_MODIFY_USER_PAW_FAILED    UI_ACCOUNT_CMD | 0X00000002 // Failed
```

```

#define UI_RC_CMD      0x05000000    // Recording command
//----- Return messages of recording command -----
#define UI_MODIFY_TTZ_SUCCESS          UI_RC_CMD | 0X00000001 // Successful
#define UI_MODIFY_TTZ_FAILED           UI_RC_CMD | 0X00000002 // Failed

#define UI_UPDATE_ALL_SUCCESS          UI_RC_CMD | 0X00000003 // Successful
#define UI_UPDATE_ALL_FAILED           UI_RC_CMD | 0X00000004 // Failed

#define UI_ADD_TASK_SUCCESS             UI_RC_CMD | 0X00000005 // Successful
#define UI_ADD_TASK_FAILED              UI_RC_CMD | 0X00000006 // Failed

#define UI_DEL_ALL_TASK_SUCCESS         UI_RC_CMD | 0X00000007 // Successful
#define UI_DEL_TASK_SUCCESS             UI_RC_CMD | 0X00000008 // Successful
#define UI_DEL_TASK_FAILED              UI_RC_CMD | 0X00000009 // Failed

#define UI_MOD_TASK_SUCCESS             UI_RC_CMD | 0X0000000A // Successful
#define UI_MOD_TASK_FAILED              UI_RC_CMD | 0X0000000B // Failed
#define UI_SCHDELE_WHEN_ONLINE          UI_RC_CMD | 0X0000000C

#define UI_DL_CMD      0x06000000    // Recorded file download command
//----- Return messages of download command -----
#define UI_UPDATE_REC_FILE_INFO_SUCCESS UI_DL_CMD | 0X00000001 // Successful
#define UI_UPDATE_REC_FILE_INFO_FAILED  UI_DL_CMD | 0X00000002 // Failed

#define UI_REC_FILE_DL_REQ_SUCCESS       UI_DL_CMD | 0X00000003 // Successful
#define UI_REC_FILE_DL_REQ_FAILED        UI_DL_CMD | 0X00000004 // Failed

#define UI_REC_FILE_DL_CANCEL_SUCCESS    UI_DL_CMD | 0X00000005 // Successful
#define UI_REC_FILE_DL_CANCEL_FAILED     UI_DL_CMD | 0X00000006 // Failed

#define UI_REC_FILE_DELETE_SUCCESS       UI_DL_CMD | 0X00000007 // Successful
#define UI_REC_FILE_DELETE_FAILED        UI_DL_CMD | 0X00000008 // Failed

#define UI_PB_CMD      0x07000000    // Playback command
//----- Return messages of playback command -----
#define UI_REC_FILE_LIVE_REQ_SUCCESS     UI_PB_CMD | 0X00000001
#define UI_REC_FILE_LIVE_REQ_FAILED      UI_PB_CMD | 0X00000002

#define UI_REC_FILE_PLAY_BEGIN_SUCCESS   UI_PB_CMD | 0X00000003
#define UI_REC_FILE_PLAY_BEGIN_FAILED    UI_PB_CMD | 0X00000004

#define UI_REC_FILE_PLAY_CANCEL_SUCCESS  UI_PB_CMD | 0X00000005 // Successful
#define UI_REC_FILE_PLAY_CANCEL_FAILED  UI_PB_CMD | 0X00000006 // Successful

```



```

#define UI_REC_FILE_PLAY_SEEK_SUCCESS      UI_PB_CMD | 0X00000007 // Failed
#define UI_REC_FILE_PLAY_SEEK_FAILED      UI_PB_CMD | 0X00000008 // Failed

#define UI_OTHER_CMD      0x08000000      // Other commands
//----- Return messages of other commands -----
#define UI_SAVE_SET_PORT_SUCCESS          UI_OTHER_CMD | 0X00000001
#define UI_SAVE_SET_PORT_FAILED          UI_OTHER_CMD | 0X00000002

#define UI_BOX_NETWORK_SET_SUCCESS        UI_OTHER_CMD | 0X00000003
#define UI_BOX_NETWORK_SET_FAILED        UI_OTHER_CMD | 0X00000004

#define UI_UPDATE_BOX_VS_SUCCESS          UI_OTHER_CMD | 0X00000005 // Successful
#define UI_UPDATE_BOX_VS_FAILED          UI_OTHER_CMD | 0X00000006 // Successful

#define UI_SOURCE_SWITCH_SUCCESS          UI_OTHER_CMD | 0X00000007 // Failed
#define UI_SOURCE_SWITCH_FAILED          UI_OTHER_CMD | 0X00000008 // Failed

#define UI_SET_VIDEO_FORMAT_SUCCESS       UI_OTHER_CMD | 0X00000009 // Successful
#define UI_SET_VIDEO_FORMAT_FAILED       UI_OTHER_CMD | 0X0000000A // Successful

#define UI_GET_BOX_INFO_SUCCESS           UI_OTHER_CMD | 0X0000000B // Failed
#define UI_GET_BOX_INFO_FAILED           UI_OTHER_CMD | 0X0000000C // Failed

#define UI_NET_MSG      0x09000000      // Network messages
#define UI_NET_NOTIFY_UI_SRC_SET_OK      UI_NET_MSG | 0X00000001
#define UI_NET_NOTIFY_UI_CHN_SET_OK      UI_NET_MSG | 0X00000002
#define UI_NET_NOTIFY_BOX_LINK_TIMEOUT  UI_NET_MSG | 0X00000003
#define UI_NET_NOTIFY_UI_BITRATE         UI_NET_MSG | 0X00000004
#define UI_NET_NOTIFY_UI_SET_CMD_PORT     UI_NET_MSG | 0X00000005
#define UI_NET_NOTIFY_UI_SET_AV_PORT      UI_NET_MSG | 0X00000007
#define UI_NET_NOTIFY_UI_SET_RTSP_PORT    UI_NET_MSG | 0X00000008
#define UI_NET_NOTIFY_UI_SET_USER_MASK    UI_NET_MSG | 0X00000009
#define UI_NET_NOTIFY_UI_RES_SET_OK       UI_NET_MSG | 0X0000000A
#define UI_NET_NOTIFY_UI_SET_USER_TYPE    UI_NET_MSG | 0X0000000B
#define UI_NET_NOTIFY_UI_FILE_DL_START    UI_NET_MSG | 0X0000000C // Starts to download
#define UI_NET_NOTIFY_UI_FILE_DL_DATA     UI_NET_MSG | 0X0000000D // Writes data
#define UI_NET_NOTIFY_UI_FILE_DL_END      UI_NET_MSG | 0X0000000E // Download complete
#define UI_NET_NOTIFY_UI_SET_WANIP        UI_NET_MSG | 0X00000010
#define UI_NET_NOTIFY_UI_UPDATE_FILELEN    UI_NET_MSG | 0X00000020

#define UI_AV_WRITERB      0x0A000000      // AV messages
#define UI_AV_AUDIO_WRITE_BUF      UI_AV_WRITERB | 0X00000001
#define UI_AV_VIDEO_WRITE_BUF      UI_AV_WRITERB | 0X00000002

```

5 Structure Definition

```
typedef struct usermanager
```

```
{
    char user_id;
    char user_level;
    char user_name[33];
    char password[33];    //
    char net_type;        // Network type, intranet 0, internet 1
    char video_source;    // Video source, CV:0, SV:1, TV 2
    char client_type;      // 0:Web  1:AP  2:phone
    char video_size;       // P:0 - 4   N:5- 9
    unsigned int country;  // Country
    int tv_channel;        // Channel number
    int tv_fre;            // Frequency
    char cur_user_num;     // Online users number
    char audio_br;         // Audio bitrate
    char rand_port;        // 0:not rand port  // 1: rand port
    char cancel_rec;       // 0:first, 1:stop rec and login in
    int upnp_vs;           // 1:UPnP, 2:vs
    int rerserver0;        // Reservation, user_mask
    int rerserver1;        // Number of remote users
    int rerserver2;        // Number of local users
    int rerserver3;        // Number of recording users
    int rerserver4;        // 0:no signal;1:has signal
}USER_MANAGER;
```

```
/*-----*/
```

```
// Tuner operating structure
```

```
// Scanning command:          uWay,uCountry,uType,uBeginFrq,uEndFrq
```

```
// --Reply:                   ScanResFlag,uScanResChnNo,uScanResChnFreq
```

```
// Stopping scanning command: uConutry
```

```
// --Reply:                   With no structure
```

```
// Channel switch command: uSetFreq, uScanResChnNo
```

```
// --Reply:                   With no structure
```

```
typedef struct fcp_tvop
```

```
{
    unsigned int uSetFreq;    // Fuequency used by scanning;
    unsigned int uWay;        // 'C': by country channel (cable or air); 'F': by frequency;
    unsigned int uCountry;    // Country code name, used by scanning and stopping scanning
commands;
    unsigned int uType;       // When uWay=='C', 'C' means cable (by channel), 'A' means air;
    unsigned int uBeginFrq;   // When uWay=='F', it means the start frequency of scanning;
    unsigned int uEndFrq;     // When uWay=='F', it means the end frequency of scanning;
```

```

    unsigned int uScanResFlag;    // Return type parameter of scanning command
    // 0x01: Starting to scan response
    // 0x02: Searching a channel response
    // 0x03: Keeping scanning response
    // 0x04: Stopping scanning response
    // 0x05: Failed to start to scan response
    unsigned int uScanResChnNo;    // Return channel number when scanning
    unsigned int uScanResChnFreq; // Return frequency when scanning
    fcp_tvop()
    {
        memset(this, 0, sizeof(fcp_tvop));
    }
}FCP_TVOP;

```

```

typedef struct tv_table //tv table;
{
    unsigned int    TVChNo;
    unsigned int    freq;
    unsigned char    chn_name[MAX_CHN_NAME];
}TV_TABLE;

```

```

typedef struct DiskEntry
{
    unsigned char id;           // Disk ID
    unsigned char name[16];     // Disk name
    unsigned int size;          // Disk space
}DISKENTRY;

```

```

typedef struct DiskDescript
{
    unsigned char disk_num;      // Disk number
    DISKENTRY disks[DISK_MAX_NUM]; // Disk information
}DISKDESCRIPT;

```

```

typedef struct RecordEntry
{
    unsigned char id;           // Task ID
    unsigned char start_time[20]; // Start time
    unsigned char end_time[20];  // End time
    unsigned char disk_id;       // Disk ID
    unsigned char record_file[33]; // -- jieM AP 090304 --
    unsigned char video_format;  // Resolution

```

```

    unsigned char  video_source;    // Video source
    unsigned short channel;         // Channel
    unsigned int   uFrq;            // Frequency
    unsigned char  mark;            // Task period mark
    unsigned char  is_cover;        // Repeat: 0: Close; 1: Enabled
    unsigned char  day_week;        // Repeat mode: once:1; daily:2; weekly:3;
}RECORDENTRY;

```

```

typedef struct RecordSchedule
{
    unsigned char task_num;          // Task number
    RECORDENTRY tasks[TASK_MAX_NUM]; // Task information
}RECORDSCHEDULE;

```

```

typedef struct RecordDataBlock
{
    unsigned char block_header;      // Data header
    unsigned char time_str[20];      // Box's time
    unsigned char time_zone[20];     // Box's time zone
    unsigned char timezone_id;       // Time zone ID
    unsigned char server_name[32];   // Server name
    DISKDESCRIPT disk_desc;          // Disk
    RECORDSCHEDULE schedules;        // Task
}RECORDDATABLOCK;

```

```

typedef struct file_all
{
    unsigned short int file_id;       // 0.1....
    unsigned char    disk_id;         // Disk ID of Box
    unsigned char    file_name[32];   // file
    FILECUT_MANAGER  filecut;
}FILE_ALL;                          // A complete file structure

```

// Remote recorded files' playback request information

```

typedef struct fcp_reclivereq
{
    FCP_BOXDLOP fileInfo;
    FCP_MP4INFO mp4Info;
    int seekTime;
    int recState; // 0: Normal 1: Box is recording 2: User cancels recording and sends command again
}FCP_RECLIVEREQ;

```

```

/*****/
// Box version information
// #define          FCP_OTHER_GET_BOX_INFO          FCP_OTHRE_BASE | 0x00070000
| PA | PN | PG
typedef struct box_info
{
    char szDevNICName[32];    //
    char szDevNICIPAddr[16]; //
    char szDevNICMaskAddr[16]; //
    char szDevNICMacAddr[32]; //
    char szDevNICGwAddr[16]; //
    char szDevNICDNSAddr1[16]; // DNS 1
    char szDevNICDNSAddr2[16]; // DNS 2
    int  uDevNICMode;        // net work mode ,0: PPPoE, 1: static ip, 2: DHCP

    char szProductVer[80];    // product version
    char szRootBoxVer[80];    // root box version
    char szLinuxVer[80];      // kernel version
    char szMspVer[80];        // msp  version
    char szAlgVer[80];        // codec version
    char szChipVer[80];       // chip version
    char szProVer[80];        // project version
    char szProTime[80];       // project build time
    char szKoVer[80];         // kernel driver version
    char szKoTime[80];        // kernel driver build time
    box_info()
    {
        memset(this, 0, sizeof(box_info));
    }
}BOX_INFO_S;

```

