

SDK-10000 for Windows Mobile
C Library Edition
V2.0

API Reference Guide



www.acti.com

Table of Contents

1	OVERVIEW	1-1
	INTRODUCTION	1-1
	INTRODUCTION	1-2
	FUNCTION SPECIFICATION.....	1-3
2	DATA STRUCTURE	2-1
	MEDIA_CONNECTION_CONFIG.....	2-1
	MEDIA_RENDER_INFO	2-4
	MEDIA_PTZ_PROTOCOL.....	2-5
3	API REFERENCE GUIDE	3-1
	INITIALIZATION	3-1
	M_KOpenInterace	3-2
	M_KCloseInterface	3-2
	CONNECTION.....	3-4
	M_KConnect	3-5
	M_KDisconnect	3-7
	M_KSetMediaConfig	3-9
	M_KSetNetworkLossCallback	3-11
	STREAM	3-13
	M_KSetAfterRenderCallback	3-14
	AUDIO	3-16
	M_KFreeAudioToken	3-17
	M_KGetAudioToken	3-19
	M_KSetMute	3-21
	M_KStartAudioTransfer	3-22
	M_KStopAudioTransfer	3-24
	RS-232/422/485 CONTROL	3-26
	M_KSendRS232Setting	3-27
	PTZ 3-29	
	M_KEnablePTZProtocol	3-30
	M_KPTZ_UP	3-31
	M_KPTZ_DOWN	3-32
	M_KPTZ_LEFT	3-33
	M_KPTZ_RIGHT	3-34
	M_KPTZ_UP_LEFT	3-35
	M_KPTZ_UP_RIGHT	3-36
	M_KPTZ_DOWN_LEFT	3-37

M_KPTZ_DOWN_RIGHT	3-38
M_KPTZ_STOP	3-39
M_KPTZ_GOTO_PRESET	3-40
M_KPTZ_SET_PRESET	3-41
M_KPTZ_ZOOM_IN	3-42
M_KPTZ_ZOOM_OUT	3-43
M_KPTZ_ZOOM_STOP	3-44
M_KPTZ_OSD_ON	3-45
M_KPTZ_OSD_OFF	3-46
M_KPTZ_OSD_UP	3-47
M_KPTZ_OSD_DOWN	3-48
M_KPTZ_OSD_LEFT	3-49
M_KPTZ_OSD_RIGHT	3-50
M_KPTZ_OSD_ENTER	3-51
M_KPTZ_OSD_LEAVE	3-52
DIGITAL I/O	3-53
KSendDO	3-54
QUAD	3-56
KQuadSetDisplayMode	3-57

4	SAMPLE CODES	4-1
	INITIALIZATION	4-1
	PREVIEW.....	4-2
	PTZ – PAN/TILT/ZOOM	4-3
	MOTION DETECTION	4-4
	DIGITAL I/O	4-6

1

OVERVIEW

Introduction

This SDK can help with application go beyond passive viewing to interact with the application developed by system integrator. This SDK provides a real time streaming to deliver live video and other surveillance functions controlling.



NOTE: This SDK Support Windows Mobile 5.0 only and suggest to develop with Visual Studio 2005

Introduction

Hardware Spec

1. Supported OS:
 - Windows Mobile 5.0
 - Windows Mobile 6.0
 - Pocket PC 2003
2. Supported CPU:
 - Intel Xscale Series (624 Mhz recommended)
 - Samsung Series
 - TI OMAP Series
3. Tested Devices
 - HP iPaq HX2790b (recommended)
 - Dopod CHT9100, P800W, D810
 - ETEN M500
 - Note that PTZ control is implemented with touch-screen control. If your mobile device does not support touch-screen control, then you cannot operate PTZ operation.
4. Supported Video Profile (benchmarked with HP iPaq HX2790b)
 - D1@6 FPS with 1.5 Mbps
 - CIF@15 FPS with 1.5 Mbps
 - CIF@30 FPS with 500 Kbps
 - Note that you need to adjust the video profile according to your mobile devices' performance

Function Specification

Function Spec

1. Add support to all ACTi Devices, including streaming type TCP 1.0 and TCP 2.0
2. Add support to all ACTi product lines, including LPL, EPL and MPL
3. Add support to Quad device. User may control 1, 2, 3, 4, Quad view
4. Add support to Speed Dome PTZ control
5. Add support to on-screen PTZ, 8-directional PT control + zoom in/out
6. Add support to set preset and goto preset PTZ operation
7. Add support to PTZ OSD operations
8. Add support to PTZ speed adjustment
9. Add support to 2-way audio support.
10. Add support to camera name display on screen
11. Add support to event display on screen, including motion detection event and digital input event
12. Add support trigger digital output

2

Data Structure

MEDIA_CONNECTION_CONFIG

The **MEDIA_CONNECTION_CONFIG** structure enable the media source information.

```
typedef struct structural_MEDIA_CONNECTION_CONFIG
{
    int             ContactType;
    int             Channel Number;
    char            Uni CastIP[16];
    char            Mul ti CastIP[16];
    char            Pl ayFi leName[256];
    char            UserID[64];
    char            Password[64];
    unsigned long   RegisterPort;
    unsigned long   StreamingPort;
    unsigned long   Control Port;
    unsigned long   Mul ti CastPort;
    unsigned long   SearchPortC2S;
    unsigned long   SearchPortS2C;
    unsigned long   HTTPPort;
    unsigned long   RTSPPort;
    unsigned long   Vi deoRTPOverMCastPort;
    unsigned long   Audi oRTPOverMCastPort;
    int             ConnectTi meOut;
}MEDIA_CONNECTION_CONFIG;
```

Members

ContactType

Contact Type	Descri ption
MOBILE_CONTACT_TYPE_UNICAST_PREVIEW	Uni -cast , usi ng ATCP10-M and ATCP20-M

ChannelNumber

Camera channel number for Mul ti -Channel video to use.

UniCastIP

Camera IP address.

MultiCastIP

Camera Multicast IP address.

PlayFileName

File name for Playback.

UserID

User Login ID.

Password

User Login password.

RegisterPort

Register port number.

StreamingPort

Streaming port number.

ControlPort

Control port number.

MultiCastPort

Multicast port number.

SearchPortC2S

Search port number (Client to Server)

SearchPortS2C

Search port number (Server to Client).

HTTPPort

HTTP port number.

RTSPPort

RTSP port number

VideoRTPOverMCastPort

Video RTP over multicast port number.

AudioRTPOverMCastPort

Audio RTP over multicast port number.

ConnectTimeOut

Time out value for connect.

MEDIA_RENDER_INFO

The **MEDIA_RENDER_INFO** structure is used to set render information.

```
typedef struct structural_MEDIA_RENDER_INFO
{
    int        DrawerInterface;
    HWND       hWnd;
    RECT       rect;
} MEDIA_RENDER_INFO;
```

Members

DrawerInterface

Draw Interface	Description
DGDI (0)	use Windows GDI for draw
DXDRAW (1)	use Direct Draw for draw

hWnd

Handle of window.

rect

Area to draw.

MEDIA_PTZ_PROTOCOL

The **PTZinfo** structure is used to set PTZ information.

```
structural _MEDIA_PTZ_PROTOCOL
{
    int nSourceType;
    char szVender[32];
    char szProtocol[32];
    char szProtocolFilename[512];
    DWORD dwAddressID;
} MEDIA_PTZ_PROTOCOL ;
```

Members

szVender

Vender name.

szProtocol

Protocol .

szProtocolFilename

File name.

dwAddressID

Address ID.

3

API Reference Guide

Initialization

<i>Name</i>	<i>Description</i>
M_KOpenInterface	Open SDK Interface
M_KCloseInterface	Close SDK Interface

M_KOpenInterface

M_KCloseInterface

Description

KOpenInterface and KCloseInterface are used for open and close SDK's Interface.

User call `HANDLE h = M_KOpenInterface();` to get the ip camera's object handle.

Then user can use the handle to deal with the IP Camera.

When the user wants to end the process, just call `M_KCloseInterface(h);` to delete the object.

Syntax

```
HANDLE M_KOpenInterface (void);  
void M_KCloseInterface(HANDLE h);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle returned by M_KOpenInterface.

Returns

Valid handle returned if success otherwise NULL.

Remarks

Check available memory for instance to allocate.

Requirements

Header file: SDK10000-M.h

Import library: KMpeg4-M.lib

Runtime DLL: KMpeg4-M.dll

Example

```
HANDLE h = M_KOpenInterface();  
...  
M_KCloseInterface(h);
```


See Also

([Back To Initialization List](#))

Connection

<i>Name</i>	<i>Description</i>
<u>M_KConnect</u>	Create a connection connects to IP Camera Server and start preview.
<u>M_KDisconnect</u>	Stop preview and disconnect connection from IPCamera Server
<u>M_KSetMediaConfig</u>	Set media configuration setting.
<u>M_KSetNetworkLossCallback</u>	Set callback function for newwork loss.

M_KConnect

Description

Create a connection and connects to IPCamera Server and start preview.

Syntax

```
bool M_KConnect(HANDLE h);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle returned by M_KOpenI nterface.

Returns

If the function succeeds, then connect to video server.

If the function fails, fail to connect.

Remarks

Requirements

Header file: SDK10000-M. h

Import library: KMpeg4-M. l l b

Runtime DLL: KMpeg4-M. d l l & relate AVC adaptors

Example

```
MEDIA_CONNECTION_CONFIG mcc;
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.Uni CastIP, "172.16.1.105 \0");
mcc.ContactType = MOBILE_CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.Mul ti CastIP, "172.16.1.105\0");
mcc.Mul ti CastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admi n\0");
```

```

mcc.ConnectTimeOut = 3;

HANDLE h = M_KOpenInterface();
if(NULL != h)
{
    if(M_KSetMediaConfig(h, &mcc)
    {
        if(M_KConnect(h))
        {
            . . . . .
        }
    }
}
. . . . .
if(NULL != h)
{
    M_KDisconnect(h);
    M_KCloseInterface(h);
    h = NULL;
}

```

See Also

[M_KDisconnect](#), ([Back To Connection List](#))

M_KDisconnect

Description

Stop preview and disconnect connection from IPCamera Server

Syntax

```
void M_KDisconnect(HANDLE h);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle returned by M_KOpenInterface()

Returns

No return value.

Remarks

Requirements

Header file: SDK10000-M.h

Import library: KMpeg4-M.lib

Runtime DLL: KMpeg4-M.dll & relate AVC adaptors

Example

```
MEDIA_CONNECTION_CONFIG mcc;
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.105\0");
mcc.ContactType = MOBILE_CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.105\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
```

```

strcpy(mcc.UserID, "Admin\0");
mcc.ConnectTimeout = 3;

HANDLE h = M_KOpenInterface();
if(NULL != h)
{
    if(M_KSetMediaConfig(h, &mcc))
    {
        if(M_KConnect(h))
        {
            . . . . .
        }
    }
    . . . . .
}
if(NULL != h)
{
    M_KDisconnect(h);
    M_KCloseInterface(h);
    h = NULL;
}

```

See Also

[M_KConnect](#), ([Back To Connection List](#))

M_KSetMediaConfig

Description

Set media configuration setting

Syntax

```
bool M_KSetMediaConfig(HANDLE h, MEDIA_CONNECTION_CONFIG* MediaConfig);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle returned by M_KOpenInterface.
<i>MediaConfig</i>	MEDIA_CONNECTION_CONFIG*	[in] Structure for connection setting.

Returns

If function succeeds, then media configuration set to SDK.

If function fails, call function KGetLastError to retrieve error code.

Remarks

Requirements

Header file: SDK10000-M.h

Import library: KMpeg4-M.lib

Runtime DLL: KMpeg4-M.dll & relate AVC adaptors

Example

```
MEDIA_CONNECTION_CONFIG mcc;
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.105\0");
mcc.ContactType = MOBILE_CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.105\0");
```

```

mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
mcc.ConnectTimeout = 3;

HANDLE h = M_KOpenInterface();
if(NULL != h)
{
    if(M_KSetMediaConfig(h, &mcc)
    {
        if(M_KConnect(h))
        {
            . . . . .
        }
    }
}
. . . . .
if(NULL != h)
{
    M_KDisconnect(h);
    M_KCloseInterface(h);
    h = NULL;
}

```

See Also

([Back To Connection List](#))

M_KSetNetworkLossCallback

Description

Set callback function for network loss.

Syntax

```
void M_KSetNetworkLossCallback(HANDLE h, DWORD UserParam,  
NETWORK_LOSS_CALLBACK fnCallback);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle returned by M_KOpenInterface()
<i>UserParam</i>	DWORD	[in] User parameter carry with callback.
<i>fnCallback</i>	NETWORK_LOSS_CALLBACK	[in] Pointer for callback function.

Returns

No return value.

Remarks

Requirements

Header file: SDK10000-M.h

Import library: KMpeg4-M.lib

Runtime DLL: KMpeg4-M.dll & relate AVC adaptors

Example

```
void CALLBACK NetWorkLossCB( DWORD UserParam )  
{  
    . . . . .  
}  
  
. . . . .  
HANDLE h = M_KOpenInterface();  
if(NULL != h)  
{
```

```
        M_KSetNetworkLossCallback(h, (DWORD)this, NetworkLossCB);  
        . . . . .  
    }
```

See Also

([Back To Connection List](#))

Stream

<i>Name</i>	<i>Description</i>
<u>M_KSetAfterRenderCallback</u>	Set the callback to get the handle after SDK paints the video on the window.

M_KSetAfterRenderCallback

Description

Set the callback to get the handle after SDK paints the video on the window

Syntax

```
void M_KSetAfterRenderCallback(HANDLE h, DWORD UserParam, AFTERRENDER_CALLBACK  
fnCallback);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle returned by M_KOpenInterface()
<i>UserParam</i>	DWORD	[in] Custom param for carry to callback function
<i>fnCallback</i>	AFTER_RENDER_CALLBACK	[in] The pointer to the callback function

Returns

No return value.

Remarks

Requirements

Header file: SDK10000-M.h

Import library: KMpeg4-M.lib

Runtime DLL: KMpeg4-M.dll & relate adaptors

Example

```
void CALLBACK AfterRenderCB( DWORD UserParam )  
{  
    . . . . .  
}  
  
. . . . .  
HANDLE h = KOpenInterface();  
if(NULL != h)
```

```
{  
    KSetAfterRenderCallback(h, (DWORD) this, AfterRenderCB);  
    . . . . .  
}
```

See Also

([Back To Stream List](#))

Audio

<i>Name</i>	<i>Description</i>
<u>M_KFreeAudioToken</u>	To release speak out session of audio
<u>M_KGetAudioToken</u>	To creat speak out session of audio to video server
<u>M_KSetMute</u>	Set to change mute status to video server
<u>M_KStartAudioTransfer</u>	Send audio to video server.
<u>M_KStopAudioTransfer</u>	Stop send audio to video server.

M_KFreeAudioToken

Description

To release speak out session of audio.

Syntax

```
void M_KFreeAudioToken(HANDLE h);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle returned by M_KOpenInterface()

Returns

No return value.

Remarks

Requirements

Header file: SDK10000-M.h

Import library: KMpeg4-M.lib

Runtime DLL: KMpeg4-M.dll & relate adaptors

Example

```
char holderip[16] = {0};
HANDLE h = M_KOpenInterface();
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.82\0");
mcc.ContactType = MOBILE_CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
```

```

strcpy(mcc.UserID, "Admin\0");
M_KSetMediaConfig(h, &mcc);
if( h )
{
    if( M_KGetAudioToken( h, holderip ) )
    {
        if( M_KStartAudioTransfer( h ) )
        {
        }
    }
}

. . . . .
if( h )
{
    M_KStopAudioTransfer( h );
    M_KFreeAudioToken( h );
}

```

See Also

[M_KGetAudioToken](#), ([Back To Audio List](#))

M_KGetAudioToken

Description

To creat speak out session of audio to video server.

Syntax

```
bool M_KGetAudioToken(HANDLE h, char* holder);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>H</i>	HANDLE	[in] The handle returned by M_KOpenInterface().
<i>holder</i>	char*	[out] Current user information.

Returns

If the function succeeds, then Aduio Token is get by current user.

If the function fails, holder holds the information of current user.

Remarks

Requirements

Header file: SDK10000-M.h

Import library: KMpeg4-M.lib

Runtime DLL: KMpeg4-M.dll & relate adaptors

Example

```
char holderip[16] = {0};
HANDLE h = M_KOpenInterface();
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.82\0");
mcc.ContactType = MOBILE_CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
```

```

mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
M_KSetMediaConfig(h, &mcc);
if( h )
{
    if( M_KGetAudioToken( h, holderip ) )
    {
        if( M_KStartAudioTransfer( h ) )
        {
        }
    }
}

. . . . .
if( h )
{
    M_KStopAudioTransfer( h );
    M_KFreeAudioToken( h );
}

```

See Also

[M_KFreeAudioToken](#), ([Back To Audio List](#))

M_KSetMute

Description

Set to change mute status to video server.

Syntax

```
void M_KSetMute(HANDLE h, bool bMute);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle returned by M_KOpenInterface()
<i>bMute</i>	bool	[in] true for set to mute and false not

Returns

No return value.

Remarks

Requirements

Header file: SDK10000-M.h

Import library: KMpeg4-M.lib

Runtime DLL: KMpeg4-M.dll & relate adaptors

Example

```
HANDLE h = M_KOpenInterface();
if(NULL != h)
{
    M_KSetMute(h, true);
    . . . . .
}
```

See Also

([Back To Audio List](#))

M_KStartAudioTransfer

Description

Send audio data to video server.

Syntax

```
bool M_KStartAudioTransfer(HANDLE h);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>H</i>	HANDLE	[in] The handle returned by M_KOpenInterface()

Returns

If the function succeeds, then Audio data is sent.

If the function fails, then no Audio data been send.

Remarks

M_KGetAudioToken() must called before this function. To stop audio transfer, function M_KStopAudioTransfer must called.

Requirements

Header file: SDK10000-M.h

Import library: KMpeg4-M.lib

Runtime DLL: KMpeg4-M.dll & relate adaptors

Example

```
char holderIp[16] = {0};
HANDLE h = M_KOpenInterface();
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.82\0");
mcc.ContactType = MOBILE_CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
```

```

mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
M_KSetMediaConfig(h, &mcc);
if( h )
{
    if( M_KGetAudioToken( h, holderip ) )
    {
        if( M_KStartAudioTransfer( h ) )
        {
        }
    }
}

. . . . .
if( h )
{
    M_KStopAudioTransfer( h );
    M_KFreeAudioToken( h );
}

```

See Also

[M_KStopAudioTransfer](#), [M_KGetAudioToken](#), [M_KFreeAudioToken](#),
([Back To Audio List](#))

M_KStopAudioTransfer

Description

Stop send audio data to video server.

Syntax

```
void M_KStopAudioTransfer(HANDLE h);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>H</i>	HANDLE	[in] The handle returned by M_KOpenInterface()

Returns

No return value.

Remarks

M_KFreeAudioToken() should call if the token is no longer use by the user.

Requirements

Header file: SDK10000-M.h

Import library: KMpeg4-M.lib

Runtime DLL: KMpeg4-M.dll & relate adaptors

Example

```
char holderip[16] = {0};
HANDLE h = M_KOpenInterface();
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.82\0");
mcc.ContactType = MOBILE_CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
mcc.MultiCastPort = 5000;
```

```

strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
M_KSetMediaConfig(h, &mcc);
if( h )
{
    if( M_KGetAudioToken( h, holderip ) )
    {
        if( M_KStartAudioTransfer( h ) )
        {
        }
    }
}

. . . . .
if( h )
{
    M_KStopAudioTransfer( h );
    M_KFreeAudioToken( h );
}

```

See Also

[M_KStartAudioTransfer](#), [M_KGetAudioToken](#), [M_KFreeAudioToken](#),
([Back To Audio List](#))

RS-232/422/485 Control

<i>Name</i>	<i>Description</i>
KSendRS232Setting	Setup the Server's RS232 X81 and BaudRate

M_KSendRS232Setting

Description

Setup the Server's RS232 X81 and BaudRate

Syntax

```
void M_KsendRs232Setting(HANDLE h, BYTE c81, BYTE dwBaudRate, int nComNumber);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle returned by M_KOpenInterface()
<i>c81</i>	BYTE	[in] The None, Even, Odd Parity.
<i>dwBaudRate</i>	BYTE	[in] The Baudrate
<i>nComNumber</i>	int	[in] Com port number

Returns

No return value.

Remarks

c81	Description
RS232_SET_N81 (0x00)	RS232 Setting N81
RS232_SET_081 (0x08)	RS232 Setting 081
RS232_SET_E81 (0x18)	RS232 Setting E81

BaudRate	Description
BAUD_RATE_1200BPS (0)	1200 BPS
BAUD_RATE_2400BPS (1)	2400 BPS
BAUD_RATE_4800BPS (2)	4800 BPS
BAUD_RATE_9600BPS (3)	9600 BPS
BAUD_RATE_19200BPS (4)	19200 BPS
BAUD_RATE_38400BPS (5)	38400 BPS

BAUD_RATE_57600BPS	(6)	57600 BPS
BAUD_RATE_115200BPS	(7)	115200 BPS
BAUD_RATE_230400BPS	(8)	230400 BPS

Requirements

Header file: SDK10000-M.h

Import library: KMpeg4-M.lib

Runtime DLL: KMpeg4-M.dll & relate adaptors

Example

```

HANDLE h = M_KOpenInterface();
memset(&mcc, 0x00, sizeof(MEDIA_CONNECTION_CONFIG));
strcpy(mcc.UniCastIP, "172.16.1.82\0");
mcc.ContactType = MOBILE_CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
mcc.MultiCastPort = 5000;
strcpy(mcc.Password, "123456\0");
strcpy(mcc.UserID, "Admin\0");
strcpy(mcc.PlayFileName, "c:\\rec.raw\0");
if(NULL != h)
{
    if(M_KSetMediaConfig(h, &mcc))
    {
        M_KConnect(h)
    }
}
. . . . .
M_KSendRS232Setting(h, RS232_SET_N81, BAUD_RATE_9600BPS, 0);
. . . . .
if(NULL != h)
{
    M_KDisconnect(h);
    M_KCloseInterface(h);
    h = NULL;
}

```

See Also

([Back To RS-232/422/485 Control List](#))

PTZ

<i>Name</i>	<i>Description</i>
<u>M_KEnablePTZProtocol</u>	Enable PTZ function with PTZ Protocol.
<u>M_KPTZ_UP</u>	Move Speed Dome up.
<u>M_KPTZ_DOWN</u>	Move Speed Dome down.
<u>M_KPTZ_LEFT</u>	Move Speed Dome left.
<u>M_KPTZ_RIGHT</u>	Move Speed Dome right.
<u>M_KPTZ_STOP</u>	Stop moving Speed Dome.
<u>M_KPTZ_GOTO_PRESET</u>	Goto preset position.
<u>M_KPTZ_SET_PRESET</u>	Set preset position.
<u>M_KPTZ_UP_RIGHT</u>	Move Speed Dome up and right.
<u>M_KPTZ_DOWN_RIGHT</u>	Move Speed Dome down and right.
<u>M_KPTZ_UP_LEFT</u>	Move Speed Dome up and left.
<u>M_KPTZ_DOWN_LEFT</u>	Move Speed Dome down and left.
<u>M_KPTZ_ZOOM_IN</u>	Zoom in Speed Dome
<u>M_KPTZ_ZOOM_OUT</u>	Zoom out Speed Dome
<u>M_KPTZ_ZOOM_STOP</u>	Stop Zoom in or Zoom out.
<u>M_KPTZ_OSD_ON</u>	Turn OSD on.
<u>M_KPTZ_OSD_OFF</u>	Turn OSD off.
<u>M_KPTZ_OSD_UP</u>	Move cursor in OSD up.
<u>M_KPTZ_OSD_DOWN</u>	Move cursor in OSD down.
<u>M_KPTZ_OSD_LEFT</u>	Move cursor in OSD left.
<u>M_KPTZ_OSD_RIGHT</u>	Move cursor in OSD right.
<u>M_KPTZ_OSD_ENTER</u>	Enter OSD options.
<u>M_KPTZ_OSD_LEAVE</u>	Leave OSD options.

M_KEnablePTZProtocol

Description

M_KEnablePTZProtocol is used to enable PTZ function.

Syntax

```
bool M_KEnablePTZProtocol (HANDLE h, bool bEnable, MEDIA_PTZ_PROTOCOL *pMPP );
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle returned by M_KOpenInterface.
<i>bEnable</i>	Bool	[in] Enable or disable PTZ Protocol
<i>pMPP</i>	MEDIA_PTZ_PROTOCOL*	[in] Structure for PTZ Protocol.

Returns

Remarks

MEDIA_PTZ_PROTOCOL is a data structure that is included the information of PTZ.

```
structural _MEDIA_PTZ_PROTOCOL
{
    int nSourceType;
    char szVender[32];
    char szProtocol[32];
    char szProtocolFilename[512];
    DWORD dwAddressID;
} MEDIA_PTZ_PROTOCOL ;
```

Requirements

Example

See Also

([Back To PTZ List](#))

M_KPTZ_UP

Description

M_KPTZ_UP is used to move Speed Dome up.

Syntax

```
BOOL M_KPTZ_UP(HANDLE h, int nSpeed);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle return from function M_KOpenInterface().
<i>nSpeed</i>	Int	[in] The int set the moving speed of Speed Dome nSpeed is 1 ~ 5.

Returns

Remarks

Requirements

Example

See Also

([Back To PTZ List](#))

M_KPTZ_DOWN

Description

M_KPTZ_DOWN is used to move Speed Dome down.

Syntax

```
BOOL M_KPTZ_DOWN(HANDLE h, int nSpeed);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle return from function M_KOpenInterface().
<i>nSpeed</i>	Int	[in] The int set the moving speed of Speed Dome nSpeed is 1 ~ 5.

Returns

Remarks

Requirements

Example

See Also

([Back To PTZ List](#))

M_KPTZ_LEFT

Description

M_KPTZ_LEFT is used to move Speed Dome left.

Syntax

```
BOOL M_KPTZ_LEFT(HANDLE h, int nSpeed);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle return from function M_KOpenInterface().
<i>nSpeed</i>	Int	[in] The int set the moving speed of Speed Dome nSpeed is 1 ~ 5.

Returns

Remarks

Requirements

Example

See Also

([Back To PTZ List](#))

M_KPTZ_RIGHT

Description

M_KPTZ_RIGHT is used to move Speed Dome right.

Syntax

```
BOOL M_KPTZ_RIGHT(HANDLE h, int nSpeed);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle return from function M_KOpenInterface().
<i>nSpeed</i>	Int	[in] The int set the moving speed of Speed Dome nSpeed is 1 ~ 5.

Returns

Remarks

Requirements

Example

See Also

([Back To PTZ List](#))

M_KPTZ_UP_LEFT

Description

M_KPTZ_UP_LEFT is used to move Speed Dome up and left.

Syntax

```
BOOL M_KPTZ_UP_LEFT(HANDLE h, int nSpeed);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle return from function M_KOpenInterface().
<i>nSpeed</i>	Int	[in] The int set the moving speed of Speed Dome nSpeed is 1 ~ 5.

Returns

Remarks

Requirements

Example

See Also

([Back To PTZ List](#))

M_KPTZ_UP_RIGHT

Description

M_KPTZ_UP_RIGHT is used to move Speed Dome up and right.

Syntax

```
BOOL M_KPTZ_UP_RIGHT(HANDLE h, int nSpeed);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle return from function M_KOpenInterface().
<i>nSpeed</i>	Int	[in] The int set the moving speed of Speed Dome nSpeed is 1 ~ 5.

Returns

Remarks

Requirements

Example

See Also

([Back To PTZ List](#))

M_KPTZ_DOWN_LEFT

Description

M_KPTZ_DOWN_LEFT is used to move Speed Dome down and left.

Syntax

```
BOOL M_KPTZ_DOWN_LEFT(HANDLE h, int nSpeed);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle return from function M_KOpenInterface().
<i>nSpeed</i>	Int	[in] The int set the moving speed of Speed Dome nSpeed is 1 ~ 5.

Returns

Remarks

Requirements

Example

See Also

([Back To PTZ List](#))

M_KPTZ_DOWN_RIGHT

Description

M_KPTZ_DOWN_RIGHT is used to move Speed Dome down and right.

Syntax

```
BOOL M_KPTZ_DOWN_RIGHT(HANDLE h, int nSpeed);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle return from function M_KOpenInterface().
<i>nSpeed</i>	Int	[in] The int set the moving speed of Speed Dome nSpeed is 1 ~ 5.

Returns

Remarks

Requirements

Example

See Also

([Back To PTZ List](#))

M_KPTZ_STOP

Description

M_KPTZ_STOP is used to stop moving Speed Dome.

Syntax

```
BOOL M_KPTZ_STOP(HANDLE h, int nDirect);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle return from function M_KOpenInterface().
<i>nDirect</i>	Int	[in] The int indicate the moving direction. nDirect = 0 when up_left, down_left, up_right, down_right nDirect = 1 when up nDirect = 2 when down nDirect = 3 when left nDirect = 4 when right

Returns

Remarks

Requirements

Example

See Also

([Back To PTZ List](#))

M_KPTZ_GOTO_PRESET

Description

M_KPTZ_GOTO_PRESET is used to move Speed Dome to preset position.

Syntax

```
BOOL M_KPTZ_GOTO_PRESET(HANDLE h, int nPresetPos);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle return from function M_KOpenInterface().
<i>nPresetPos</i>	Int	[in] nPresetPos is 1 ~ 8.

Returns

Remarks

Requirements

Example

See Also

([Back To PTZ List](#))

M_KPTZ_SET_PRESET

Description

M_KPTZ_SET_PRESET is used to set current position to preset position.

Syntax

```
BOOL M_KPTZ_SET_PRESET(HANDLE h, int nPresetPos);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle return from function M_KOpenInterface().
<i>nPresetPos</i>	Int	[in] nPresetPos is 1 ~ 8.

Returns

Remarks

Requirements

Example

See Also

([Back To PTZ List](#))

M_KPTZ_ZOOM_IN

Description

M_KPTZ_ZOOM_IN is used to zoom in.

Syntax

```
BOOL M_KPTZ_ZOOM_IN(HANDLE h);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle return from function M_KOpenInterface().

Returns

Remarks

Requirements

Example

See Also

([Back To PTZ List](#))

M_KPTZ_ZOOM_OUT

Description

M_KPTZ_ZOOM_OUT is used to zoom out.

Syntax

```
BOOL M_KPTZ_ZOOM_OUT(HANDLE h);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle return from function M_KOpenInterface().

Returns

Remarks

Requirements

Example

See Also

([Back To PTZ List](#))

M_KPTZ_ZOOM_STOP

Description

M_KPTZ_ZOOM_STOP is used to stop zoom.

Syntax

```
BOOL M_KPTZ_ZOOM_STOP(HANDLE h);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle return from function M_KOpenInterface().

Returns

Remarks

Requirements

Example

See Also

([Back To PTZ List](#))

M_KPTZ_OSD_ON

Description

M_KPTZ_OSD_ON is used to turn OSD screen on.

Syntax

```
BOOL M_KPTZ_OSD_ON(HANDLE h);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle return from function M_KOpenInterface().

Returns

Remarks

Requirements

Example

See Also

([Back To PTZ List](#))

M_KPTZ_OSD_OFF

Description

M_KPTZ_OSD_OFF is used to turn OSD screen off.

Syntax

```
BOOL M_KPTZ_OSD_OFF(HANDLE h);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle return from function M_KOpenInterface().

Returns

Remarks

Requirements

Example

See Also

([Back To PTZ List](#))

M_KPTZ_OSD_UP

Description

M_KPTZ_OSD_UP is used to move cursor in OSD up.

Syntax

```
BOOL M_KPTZ_OSD_UP(HANDLE h);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle return from function M_KOpenInterface().

Returns

Remarks

Requirements

Example

See Also

([Back To PTZ List](#))

M_KPTZ_OSD_DOWN

Description

M_KPTZ_OSD_DOWN is used to move cursor in OSD down.

Syntax

```
BOOL M_KPTZ_OSD_DOWN(HANDLE h);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle return from function M_KOpenInterface().

Returns

Remarks

Requirements

Example

See Also

([Back To PTZ List](#))

M_KPTZ_OSD_LEFT

Description

M_KPTZ_OSD_LEFT is used to move cursor in OSD left.

Syntax

```
BOOL M_KPTZ_OSD_LEFT(HANDLE h);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle return from function M_KOpenInterface().

Returns

Remarks

Requirements

Example

See Also

([Back To PTZ List](#))

M_KPTZ_OSD_RIGHT

Description

M_KPTZ_OSD_RIGHT is used to move cursor in OSD right.

Syntax

```
BOOL M_KPTZ_OSD_RIGHT(HANDLE h);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle return from function M_KOpenInterface().

Returns

Remarks

Requirements

Example

See Also

([Back To PTZ List](#))

M_KPTZ_OSD_ENTER

Description

M_KPTZ_OSD_ENTER is used to move cursor in OSD enter current option.

Syntax

```
BOOL M_KPTZ_OSD_ENTER(HANDLE h);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle return from function M_KOpenInterface().

Returns

Remarks

Requirements

Example

See Also

([Back To PTZ List](#))

M_KPTZ_OSD_LEAVE

Description

M_KPTZ_OSD_LEAVE is used to move cursor in OSD leave current option.

Syntax

```
BOOL M_KPTZ_OSD_LEAVE(HANDLE h);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle return from function M_KOpenInterface().

Returns

Remarks

Requirements

Example

See Also

([Back To PTZ List](#))

Digital I/O

<i>Name</i>	<i>Description</i>
KSendDO	Send DO to video server.

KSendDO

Description

Send DO to video server.

Syntax

```
void KSendDO (HANDLE h, BYTE bDOData);
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle returned by KOpenInterface()
<i>bDOData</i>	BYTE	[in] the DO Event

Returns

No return value.

Remarks

DO value	Description
DO_OUTPUT_CLEAN (0X00)	Clean DO.
DO_OUTPUT_1 (0X01)	DO 1
DO_OUTPUT_2 (0X02)	DO 2
DO_OUTPUT_BOTH (0X03)	DO 1 & 2

Requirements

Header file: SDK10000-M.h

Import library: KMpeg4-M.lib

Runtime DLL: KMpeg4-M.dll & relate AVC adaptors

Example

```
HANDLE h = KOpenInterface();  
.  
.  
.  
if(NULL != h)  
{
```

```

    if(M_KSetMediaConfig(h, &mcc))
    {
        if(M_KConnect(h))
        {
            if(M_KStartStream(h))
            {
                M_KSendD0(h, DO_OUTPUT_1);
            }
        }
    }
    . . . . .
    if(NULL != h)
    {
        M_KStop(h);
        M_KStopStream(h);
        M_KDisconnect(h);
        M_KCloseInterface(h);
        h = NULL;
    }

```

See Also

([Back To Digital I/O List](#))

QUAD

<i>Name</i>	<i>Description</i>
M_ KQuadSetDisplayMode	Set Quad display mode.

KQuadSetDisplayMode

Description

Set Quad display mode.

Syntax

```
bool M_KQuadSetDisplayMode(HANDLE h, int nMode)
```

Parameters

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>h</i>	HANDLE	[in] The handle returned by KOpenInterface()
<i>nMode</i>	int	[in] Display mode.

Returns

If function return succeeds, then new display has set to the Quad video server.

If function return fails, then display remain the same.

Remarks

Value for Display mode.

0 – Quad display.

1 – Display channel one.

2 – Display channel two.

3 – Display channel three.

4 – Display channel four.

Requirements

Header file: **SDK10000-M.h**

Import library: **KMpeg4-M.lib**

Runtime DLL: **KMpeg4-M.dll & relate AVC adaptors**

Example

```
HANDLE h = M_KOpenInterface();  
. . . . .
```

```

if(NULL != h)
{
    if(M_KSetMediaConfig(h, &mcc))
    {
        if(M_KConnect(h))
        {
            if(M_KStartStream(h))
            {
                M_KQuadSetDisplayMode(h, nMode);
            }
        }
    }
}
. . . . .
if(NULL != h)
{
    M_KStop(h);
    M_KStopStream(h);
    M_KDisconnect(h);
    M_KCloseInterface(h);
    h = NULL;
}

```

See Also

[KQuadGetDisplayMode](#), ([Back To QUAD List](#))

6

Sample Codes

Initialization

```
HANDLE m_hSDK = M_KOpenInterface();
M_KSetStreamingType( m_hSDK, 2 ); // 1 for 1.0, 2 for 2.0
MEDIA_CONNECTION_CONFIG mcc;
strcpy(mcc.UniCastIP, "172.16.1.100\0");
mcc.ContactType = MOBILE_CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
strcpy(mcc.UserID, "Admin\0");
strcpy(mcc.Password, "123456\0");

if( NULL != hSDK )
{
    if( M_KSetMediaConfig( m_hSDK, &mcc ); )
    {
        MEDIA_RENDER_INFO mri;
        mri.hWnd = m_hWnd; // HWND of Display target window
        if( M_KConnect( m_hSDK ) )
        {
            ...
        }
    }
}
```

Preview

```
HANDLE m_hSDK = M_KOpenInterface();
M_KSetStreamingType( m_hSDK, 2 ); // 1 for 1.0, 2 for 2.0
MEDIA_CONNECTION_CONFIG mcc;
strcpy(mcc.UniCastIP, "172.16.1.100\0");
mcc.ContactType = MOBILE_CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
strcpy(mcc.UserID, "Admin\0");
strcpy(mcc.Password, "123456\0");

if( NULL != hSDK )
{
    if( M_KSetMediaConfig( m_hSDK, &mcc ); )
    {
        MEDIA_RENDER_INFO mri;
        mri.hWnd = m_hWnd; // HWND of Display target window
        if( M_KConnect( m_hSDK ) )
        {
            ...
            // Preview will start
        }
    }
}

// To Stop Preview
if(NULL != h)
{
    M_KDisconnect( m_hSDK );
    M_KCloseInterface( m_hSDK );
    m_hSDK = NULL;
}
```

PTZ – Pan/Tilt/Zoom

```
// To enable PTZ
MEDIA_PTZ_PROTOCOL mpp;
mpp.dwAddressID = 1;
sInfo.dwAddrID = 1;
mpp.nSourceType = 0;
strcpy(mpp.szProtocolFilename, "\\62-Pelco-P.ptz");
M_KEnablePTZProtocol( m_hSDK, true, &mpp );
// Specify the Rs232 setting for video server
M_KSendRS232Setting( m_hSDK, NET_N81, NET_9600BPS );
. . . . .
// After connect, we can send the PTZ command with the setting
M_KPTZ_UP( m_hSDK, nSpeed );
// To call stop when we want PTZ stop
M_KPTZ_STOP( m_hSDK, PTZ_STOP_MOVE_UP );


// zoom in
M_KPTZ_ZOOM_IN( m_hSDK );
// To stop zoom in
M_KPTZ_ZOOM_STOP( m_hSDK );
```

Motion Detection

```
void CALLBACK MotionDetectionCB( DWORD UserParam, bool bMotion1,
bool bMotion2, bool bMotion3, , bool bMotion4 )
{
    if(bMotion1)
    {
        printf("Motion 1\n");
    }
    if(bMotion2)
    {
        printf("Motion 2\n");
    }
    if(bMotion3)
    {
        printf("Motion 3\n");
    }
    if(bMotion4)
    {
        // Qaud only
        printf("Motion 4\n");
    }
}

HANDLE m_hSDK = M_KOpenInterface();
M_KSetStreamingType( m_hSDK, 2 ); // 1 for 1.0, 2 for 2.0
MEDIA_CONNECTION_CONFIG mcc;
strcpy(mcc.UniCastIP, "172.16.1.100\0");
mcc.ContactType = MOBILE_CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
strcpy(mcc.UserID, "Admin\0");
strcpy(mcc.Password, "123456\0");

if( NULL != hSDK )
{
    if( M_KSetMediaConfig( m_hSDK, &mcc ); )
    {
        M_KSetMotionDetectionCallback(m_hSDK,
MotionDetectionCB );
(DWORD) this,
```



```
MEDIA_RENDER_INFO mri;  
mri.hWnd = m_hWnd; // HWND of Display target window  
if( M_KConnect( m_hSDK ) )  
{  
    ...  
}
```

Digital I/O

```
void CALLBACK DICB( DWORD UserParam, bool bDI1, bool bDI2 )
{
    if(bDI1)
    {
        printf("DI 1\n");
    }
    if(bDI2)
    {
        printf("DI 2\n");
    }
}

HANDLE m_hSDK = M_KOpenInterface();
M_KSetStreamingType( m_hSDK, 2 ); // 1 for 1.0, 2 for 2.0
MEDIA_CONNECTION_CONFIG mcc;
strcpy(mcc.UniCastIP, "172.16.1.100\0");
mcc.ContactType = MOBILE_CONTACT_TYPE_UNICAST_PREVIEW;
mcc.HTTPPort = 80;
mcc.RegisterPort = 6000;
mcc.ControlPort = 6001;
mcc.StreamingPort = 6002;
mcc.ChannelNumber = 0;
strcpy(mcc.MultiCastIP, "172.16.1.82\0");
strcpy(mcc.UserID, "Admin\0");
strcpy(mcc.Password, "123456\0");

if( NULL != hSDK )
{
    if( M_KSetMediaConfig( m_hSDK, &mcc ); )
    {
        M_KSetCallback( m_hSDK, (DWORD)this, DICB );
        MEDIA_RENDER_INFO mri;
        mri.hWnd = m_hWnd; // HWND of Display target window
        if( M_KConnect( m_hSDK ) )
        {
            // 0x01 - D0 1, 0x02 - D02, 0x03 - D01 and D02, 0x00 - Stop D0
            M_KSendD0( m_hSDK, 0x01 );
            ...
        }
    }
}
```

