



# **Yuan's QCAP SDK SC510 Easy Programming Guide**

*1.1.0.127.8, 2014.04.18*

# 1. Introduction

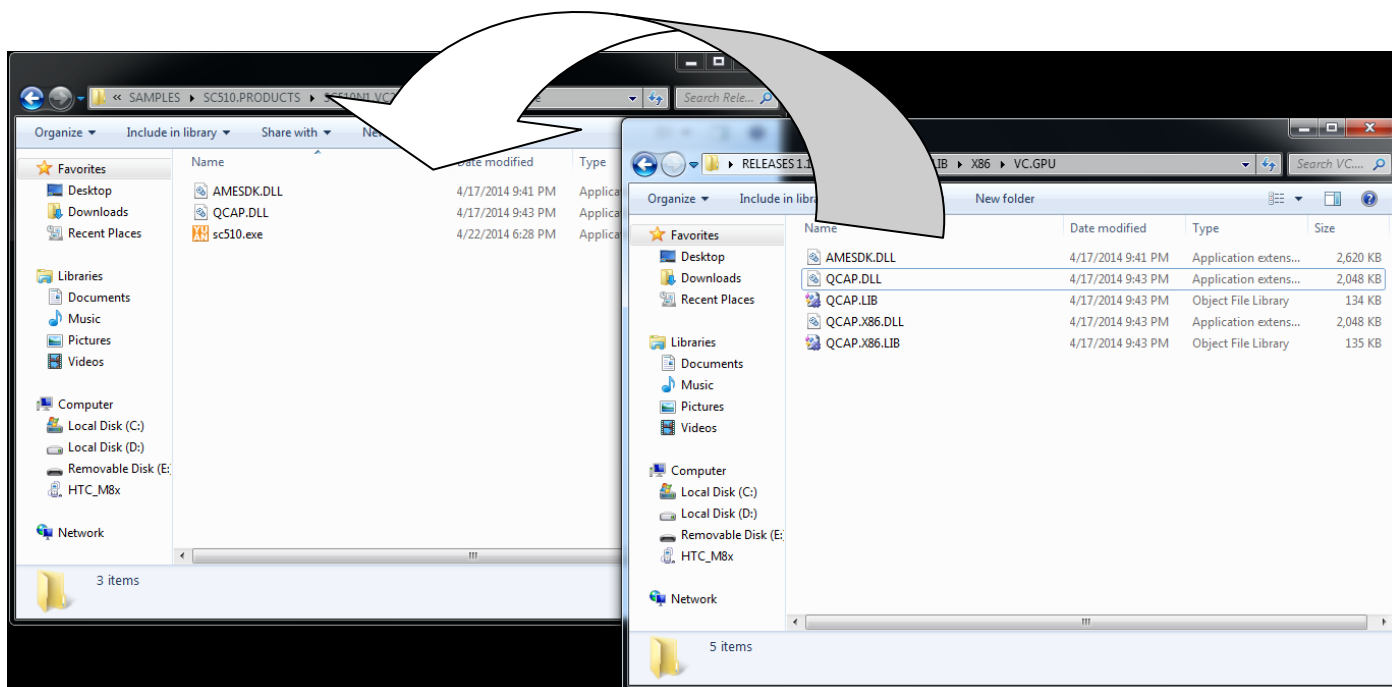
## Overview

In this document, we will guide and teach users how step by step to use QCAP SDK functions that implement SC510 capture card.

## Setting DLL File

VC, LabView	X86	QCAP.DLL, AMESDK.DLL
	X64	QCAP.X64.DLL, AMESDK.X64.DLL
C#, VB, Delphi	X86	QCAP.DLL, QCAP.NET.DLL, AMESDK.DLL
	X64	QCAP.X64.DLL, QCAP.NET.X64.DLL, AMESDK.X64.DLL

Developer should copy QCAP.DLL and AMESDK.DLL with .EXE file as is shown in below figure.





## 2. Initialize Device APIs

```
QCAP_SET_SYSTEM_CONFIGURATION( ... );

QCAP_CREATE( "SA7160 PCI", 0, ... );

QCAP_REGISTER_FORMAT_CHANGED_CALLBACK( pDevice, on_format_changed_callback, ... );

QCAP_REGISTER_NO_SIGNAL_DETECTED_CALLBACK( m_pDevice, on_no_signal_detected_callback, ... );

QCAP_REGISTER_SIGNAL_REMOVED_CALLBACK( m_pDevice, on_no_signal_removed_callback, ... );

QCAP_REGISTER_VIDEO_PREVIEW_CALLBACK( m_pDevice, on_video_preview_callback, ... );

QCAP_REGISTER_AUDIO_PREVIEW_CALLBACK( m_pDevice, on_audio_preview_callback, ... );

QCAP_SET_VIDEO_DEINTERLACE_TYPE( m_pDevice, QCAP_SOFTWARE_DEINTERLACE_TYPE_BLENDING );

QCAP_SET_VIDEO_DEINTERLACE( m_pDevice, TRUE );

QCAP_SET_VIDEO_INPUT( m_pDevice, QCAP_INPUT_TYPE_AUTO );

QCAP_SET_AUDIO_INPUT( m_pDevice, QCAP_INPUT_TYPE_EMBEDDED_AUDIO );

QCAP_SET_AUDIO_VOLUME( m_pDevice, 100 );

QCAP_RUN( m_pDevice );
```

## 3. Uninitialize Device APIs

```
QCAP_STOP( m_pDevice );

QCAP_DESTROY( m_pDevice );
```



## 4. Start Channel Record APIs

```
QCAP_SET_VIDEO_RECORD_PROPERTY( m_pDevice, 0, ... );  
QCAP_SET_AUDIO_RECORD_PROPERTY( m_pDevice, 0, ... );  
QCAP_START_RECORD( m_pDevice, 0, "CHANNEL01.MP4" );
```

## 5. Stop Channel Record APIs

```
QCAP_STOP_RECORD( m_pDevice, 0 );
```



## 6. Start Share Record APIs

```
QCAP_SET_VIDEO_SHARE_RECORD_PROPERTY( 0, ... );  
QCAP_SET_AUDIO_SHARE_RECORD_PROPERTY( 0, ... );  
QCAP_START_SHARE_RECORD( 0, "SHARE01.MP4" ", dwFlags );
```

*Note!! For compression data user, the QCAP\_RECORD\_FLAG\_ENCODE flag should be cleared from the parameters, dwFlags, in QCAP\_START\_SHARE\_RECORD API to disable the software encoder's resource.*

## 7. Set Share Record Data APIs

```
QRETURN on_video_preview_callback( ..., BYTE * pFrameBuffer, ULONG nFrameBufferLen, ... )  
{  
    ...  
    if( g_n_share_record_state > 0 ) {  
        QCAP_SET_VIDEO_SHARE_RECORD_UNCOMPRESSION_BUFFER( ..., pFrameBuffer, nFrameBufferLen, ... );  
    }  
    ...  
}  
  
QRETURN on_audio_preview_callback( ..., BYTE * pFrameBuffer, ULONG nFrameBufferLen, ... )  
{  
    ...  
    if( g_n_share_record_state > 0 ) {  
        QCAP_SET_AUDIO_SHARE_RECORD_UNCOMPRESSION_BUFFER( ..., pFrameBuffer, nFrameBufferLen, ... );  
    }  
    ...  
}
```

## 8. Stop Share Record APIs

```
QCAP_STOP_SHARE_RECORD( 0 );
```



## 9. Share Record with Multi-Threading

```
DWORD WINAPI on_video_preview_callback_ex( LPVOID params )
{
    ...

    HANDLE events[ 2 ] = { g_h_share_record_thread_stop_events[ 0 ],
                          g_h_share_record_buffer_ready_events[ 0 ] };

    while( TRUE ) {

        DWORD returns = WaitForMultipleObjects( 2, events, FALSE, INFINITE );

        if( returns == (WAIT_OBJECT_0) ) { break ; }

        if( returns == (WAIT_OBJECT_0 + 1) ) {

            EnterCriticalSection( g_h_share_record_access_critical_sections[ 0 ] );

            BYTE * po = g_p_share_record_buffers[ 0 ];

            ULONG sz = g_n_share_record_buffer_lengths[ 0 ];

            if( g_n_share_record_state > 0 ) {

                LeaveCriticalSection( g_h_share_record_access_critical_sections[ 0 ] );

                QCAP_SET_VIDEO_SHARE_RECORD_UNCOMPRESSION_BUFFER( ..., po, sz, ... );

            }
            else {

                LeaveCriticalSection( g_h_share_record_access_critical_sections[ 0 ] );

            }

        }

    }

    ...
}

QRETURN on_video_preview_callback( ..., BYTE * pFrameBuffer, ULONG nFrameBufferLen, ... )
{
    ...

    EnterCriticalSection( g_h_share_record_access_critical_sections[ 0 ] );

    if( g_n_share_record_state > 0 ) {

        g_p_share_record_buffers[ 0 ] = pFrameBuffer;


        g_n_share_record_buffer_lengths[ 0 ] = nFrameBufferLen;

        SetEvent( g_h_share_record_buffer_ready_events[ 0 ] );

    }
    LeaveCriticalSection( g_h_share_record_access_critical_sections[ 0 ] );

    ...

}
```



```

DWORD WINAPI on_audio_preview_callback_ex( LPVOID params )
{
    ...

    HANDLE events[ 2 ] = { g_h_share_record_thread_stop_events[ 1 ],
                           g_h_share_record_buffer_ready_events[ 1 ] };

    while( TRUE ) {

        DWORD returns = WaitForMultipleObjects( 2, events, FALSE, INFINITE );

        if( returns == (WAIT_OBJECT_0) ) { break ; }

        if( returns == (WAIT_OBJECT_0 + 1) ) {

            EnterCriticalSection( g_h_share_record_access_critical_sections[ 1 ] );

            BYTE * po = g_p_share_record_buffers[ 1 ];

            ULONG sz = g_n_share_record_buffer_lengths[ 1 ];

            if( g_n_share_record_state > 0 ) {

                LeaveCriticalSection( g_h_share_record_access_critical_sections[ 1 ] );

                QCAP_SET_AUDIO_SHARE_RECORD_UNCOMPRESSION_BUFFER( ..., po, sz, ... );

            }
            else {

                LeaveCriticalSection( g_h_share_record_access_critical_sections[ 1 ] );

            }

        }

    }

    ...
}

QRETURN on_audio_preview_callback( ..., BYTE * pFrameBuffer, ULONG nFrameBufferLen, ... )
{
    ...

    EnterCriticalSection( g_h_share_record_access_critical_sections[ 1 ] );

    if( g_n_share_record_state > 0 ) {

        g_p_share_record_buffers[ 1 ] = pFrameBuffer;

        g_n_share_record_buffer_lengths[ 1 ] = nFrameBufferLen;

        SetEvent( g_h_share_record_buffer_ready_events[ 1 ] );

    }
    LeaveCriticalSection( g_h_share_record_access_critical_sections[ 1 ] );

    ...
}

```



## 10. Start Broadcast APIs

```
QCAP_CREATE_BROADCAST_RTSP_SERVER( 0, ..., &pServer, ... );  
QCAP_SET_VIDEO_BROADCAST_SERVER_PROPERTY( pServer, ..., dwFlags );  
QCAP_SET_AUDIO_BROADCAST_SERVER_PROPERTY( pServer, ... );  
QCAP_START_BROADCAST_SERVER( pServer );
```

*Note!! For compression data user, the QCAP\_BROADCAST\_FLAG\_ENCODE flag should be cleared from the parameter, dwFlags, in QCAP\_START\_SHARE\_RECORD API to disable the software encoder's resource.*

## 11. Set Broadcast Data APIs

```
QRETURN on_video_preview_callback( ..., BYTE * pFrameBuffer, ULONG nFrameBufferLen, ... )  
{  
    ...  
    if( g_n_broadcast_server_state > 0 ) {  
        QCAP_SET_VIDEO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER( ..., pFrameBuffer, nFrameBufferLen, ... );  
    }  
    ...  
}  
  
QRETURN on_audio_preview_callback( ..., BYTE * pFrameBuffer, ULONG nFrameBufferLen, ... )  
{  
    ...  
    if( g_n_broadcast_server_state > 0 ) {  
        QCAP_SET_AUDIO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER( ..., pFrameBuffer, nFrameBufferLen, ... );  
    }  
    ...  
}
```

## 12. Stop Broadcast APIs

```
QCAP_STOP_BROADCAST_SERVER( pServer );
```





## 13. Broadcast with Multi-Threading

```
DWORD WINAPI on_video_preview_callback_ex( LPVOID params )
{
    ...

    HANDLE events[ 2 ] = { g_h_broadcast_server_thread_stop_events[ 0 ],
                          g_h_broadcast_server_buffer_ready_events[ 0 ] };

    while( TRUE ) {

        DWORD returns = WaitForMultipleObjects( 2, events, FALSE, INFINITE );

        if( returns == (WAIT_OBJECT_0) ) { break ; }

        if( returns == (WAIT_OBJECT_0 + 1) ) {

            EnterCriticalSection( g_h_broadcast_server_access_critical_sections[ 0 ] );

            BYTE * po = g_p_broadcast_server_buffers[ 0 ];

            ULONG sz = g_n_broadcast_server_buffer_lengths[ 0 ];

            if( g_n_broadcast_server_state > 0 ) {

                LeaveCriticalSection( g_h_broadcast_server_access_critical_sections[ 0 ] );

                QCAP_SET_VIDEO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER( ..., po, sz, ... );

            }
            else {

                LeaveCriticalSection( g_h_broadcast_server_access_critical_sections[ 0 ] );

            }

        }

    }

    ...
}

QRETURN on_video_preview_callback( ..., BYTE * pFrameBuffer, ULONG nFrameBufferLen, ... )
{
    ...

    EnterCriticalSection( g_h_broadcast_server_access_critical_sections[ 0 ] );

    if( g_n_broadcast_server_state > 0 ) {

        g_p_broadcast_server_buffers[ 0 ] = pFrameBuffer;

        g_n_broadcast_server_buffer_lengths[ 0 ] = nFrameBufferLen;

        SetEvent( g_h_broadcast_server_buffer_ready_events[ 0 ] );

    }
    LeaveCriticalSection( g_h_broadcast_server_access_critical_sections[ 0 ] );

    ...
}
```



```

DWORD WINAPI on_audio_preview_callback_ex( LPVOID params )
{
    ...

    HANDLE events[ 2 ] = { g_h_broadcast_server_thread_stop_events[ 1 ],
                           g_h_broadcast_server_buffer_ready_events[ 1 ] };

    while( TRUE ) {

        DWORD returns = WaitForMultipleObjects( 2, events, FALSE, INFINITE );

        if( returns == (WAIT_OBJECT_0) ) { break ; }

        if( returns == (WAIT_OBJECT_0 + 1) ) {

            EnterCriticalSection( g_h_broadcast_server_access_critical_sections[ 1 ] );

            BYTE * po = g_p_broadcast_server_buffers[ 1 ];

            ULONG sz = g_n_broadcast_server_buffer_lengths[ 1 ];

            if( g_n_broadcast_server_state > 0 ) {

                LeaveCriticalSection( g_h_broadcast_server_access_critical_sections[ 1 ] );

                QCAP_SET_AUDIO_BROADCAST_SERVER_UNCOMPRESSION_BUFFER( ..., po, sz, ... );

            }
            else {

                LeaveCriticalSection( g_h_broadcast_server_access_critical_sections[ 1 ] );

            }

        }

    }

    ...
}

QRETURN on_audio_preview_callback( ..., BYTE * pFrameBuffer, ULONG nFrameBufferLen, ... )
{
    ...

    EnterCriticalSection( g_h_broadcast_server_access_critical_sections[ 1 ] );

    if( g_n_broadcast_server_state > 0 ) {

        g_p_broadcast_server_buffers[ 1 ] = pFrameBuffer;

        g_n_broadcast_server_buffer_lengths[ 1 ] = nFrameBufferLength;

        SetEvent( g_h_broadcast_server_buffer_ready_events[ 1 ] );

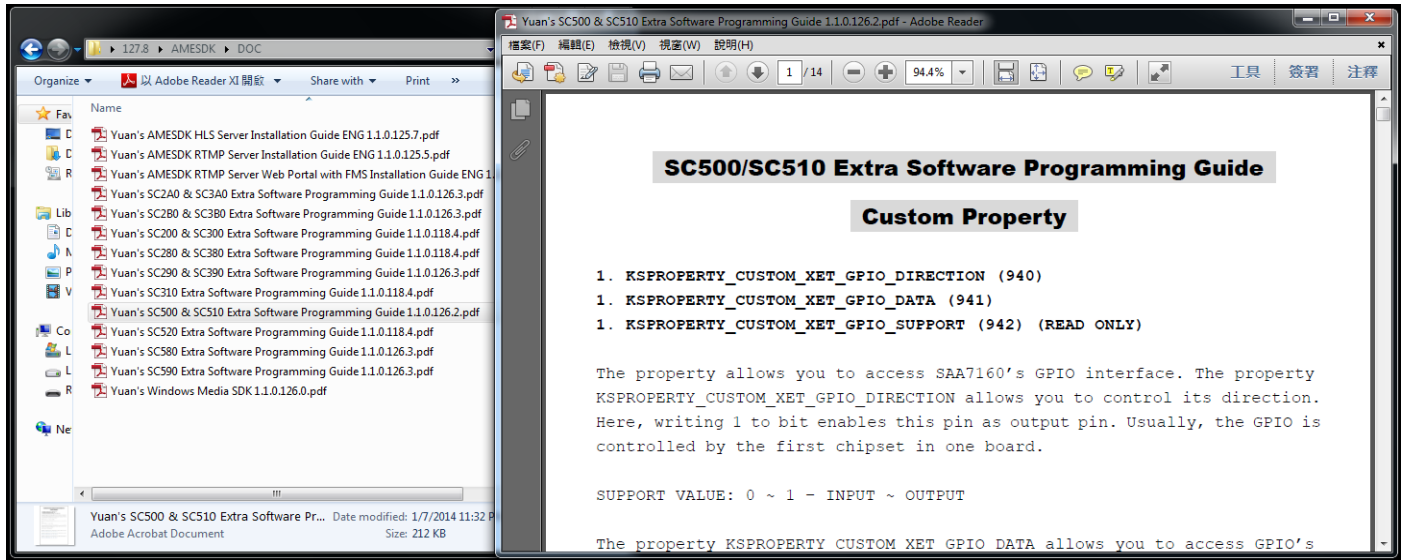
    }
    LeaveCriticalSection( g_h_broadcast_server_access_critical_sections[ 1 ] );

    ...
}

```

# 14. Custom Preoperty for SC510

For more custom device property programming, please reference product's Extra Prgoramming Guide in SDK packet. User can find Extra Prgoramming Guide as shown below.





Example for 510 video input's media content owns HDCP or MarcoVision protection.

```
QCAP_GET_DEVICE_CUSTOM_PROPERTY( pDevice, 202, &HDCP );  
  
IF( HDCP == 1 ) { RECORD_FUNCTION = DISABLE; }  
  
IF( HDCP == 0 ) { RECORD_FUNCTION = ENABLE; }
```