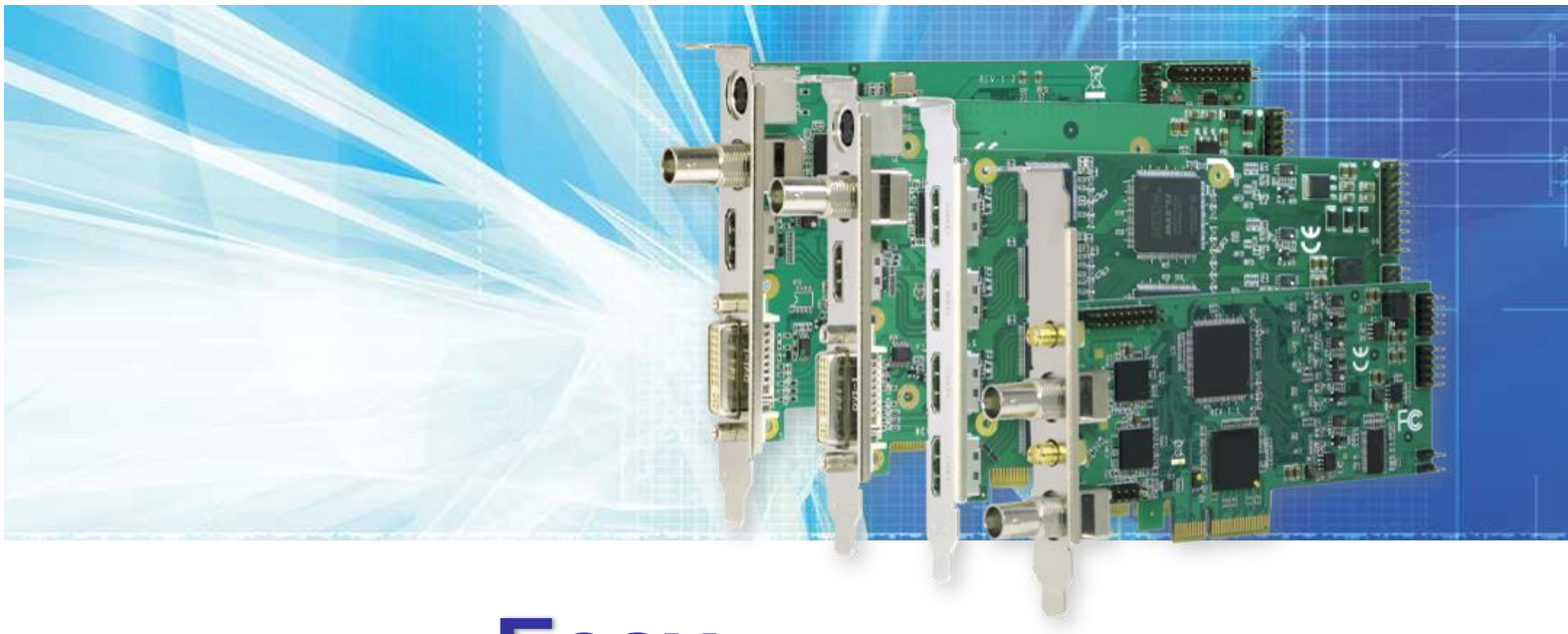
UFG-12 Family

Frame Grabbers



Easy Programming Guide

Windows

UNIGRAF

Copyright

Notice

The information given in this manual is verified in the correctness on the date of issue. The authors reserve the rights to make any changes to this product and to revise the information about the products contained in this manual without an obligation to notify any persons about such revisions or changes.

Edition

UFG-12 Easy Programming Guide, Rev 1.1.0.129.8

Document identifier:

Date: 02 October 2014

Trademarks

Company information

Unigraf Oy

Piispantilankuja 4
FI-02240 ESPOO
Finland

info@unigraf.fi

http://www.unigraf.fi

Table of Contents

1. INTRODUCTION

Overview

This document will provide step-by-step quidance of the use of basic UFG-12 QCAP SDK functions needed to control and capture video using the UFG-12 models without on-board hardware compression in Windows operating system. Please refer to *UFG-12 MC Easy Programming Guide* for use of the *UFG-12 MC* model with hardware compression and *UFG-12 Linux Software Programming Guide* for the use of the non compression models in Linux operating system.

Setting DLL File

VC	X86	QCAP.DLL, AMESDK.DLL
	X64	QCAP.X64.DLL, AMESDK.X64.DLL

Note	Please make sure that QCAP.DLL and AMESDK.DLL are included in the same directory with executable (*.exe) file.

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 );
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 ],
                    _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 ] );
    …
}

        Cntd…
```

```
        Cntd…

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 ] );
    …
}
```

9. START BROADCAST APIS

```
QCAP_CREATE_BROADCAST_RTSP_SERVER( 0, …, &pServer, … );
CAP_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.

10. 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,
… );
    }
    …
}
```

11. STOP BROADCAST APIS

```
QCAP_STOP_BROADCAST_SERVER( pServer );
```

12. 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 ] );
    …
}

        Cntd…
```

```
        Cntd…

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 ] );
    …
}
```

13. CUSTOM PROPERTY FOR UFG-12

Example for 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; }
```