DUO C++ Samples

Overview

This article is a review of the `C++` samples that ship with the DUO SDK. This will be updated as new features are added or specifications updated to the DUO API. In the `Developers/Samples` folder within the SDK download you can find the latest version of these samples with a `cmake` project to generate specific IDE projects (Visual Studio/Qt Creator).

Prerequisites

Before reading this guide it is recommended to review our API and SDK Docs to get an understanding of the design, common practices and usage. With DUO devices you will always receive a "frame" which contains all relevant sensor data. The following examples showcase the usage of the DUOFrame structure while using different device functions. Make sure you have the latest SDK download.

- DUOLib.lib & DUOLib dynamic library that match your compiler and architecture (x86/x64)
- CMake 2.8+

Methods/Structures

Here are some of the common methods used through-out the examples:

- `EnumerateResolutions` - Lists available resolutions.
    - `DUOResolutionInfo` - Structure used for resolution information.
- `DUOFrameCallback` - Callback on frame update.
- `OpenDUO` - Opens a new connection to the device.
- `StartDUO` - Initializes capture on the device.
- `StopDUO` - Un-Initializes capture on the device.
- `CloseDUO` - Closes the connection to the device.

And variables/structures:

- `PDUOFrame` - Structure containing device sensor data.
    - `pFrameData->timeStamp` - Frame time stamp in 100µs increments.

Include/Linkage

The including/linkage of the DUOLib library (which may vary dependant on compiler).

```
#include "DUOLib.h"
#pragma comment(lib, "DUOLib.lib")
```

Headers

Samples 1-4 header files are generally the same with platform dependent method for console input and including of the DUOLib header.

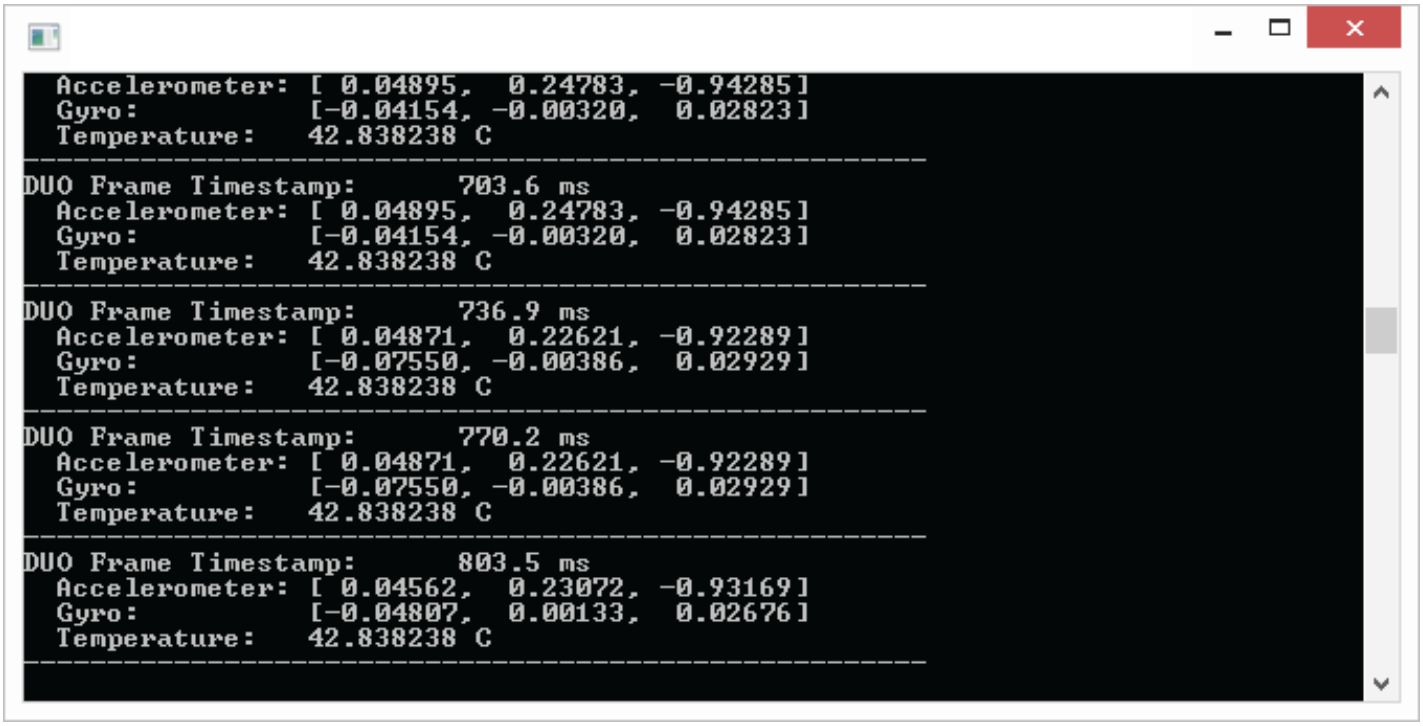
```
#ifndef SAMPLE_H
#define SAMPLE_H

// Include some generic header files
#if defined(WIN32)
    ...
#elif defined(__linux__) || defined(__APPLE__)
    ...
#endif

// Include DUO API header file
#include "DUOLib.h"

#endif // SAMPLE_H
```

Sample 01

Capturing Motion Data

If your DUO features an accelerometer/gyroscope sensors you can access the data (which is passed alongside with frame data) by accessing the `pFrameData` returned to on callback. The key variables in this example are:

- `pFrameData->accelerometerPresent` - Flag for check if a MPU sensor is available.
- `pFrameData->accelData[0,1,2]` - Array containing the accelerometer data (x,y,z)
- `pFrameData->gyroData[0,1,2]` - Array containing the gyroscope data (x,y,z)
- `pFrameData->tempData` - Float with the temperature data (0.0C)




```
#include "Sample.h"
void CALLBACK DUOCallback(const PDUOFrame pFrameData, void *pUserData)
{
    printf("DUO Frame Timestamp: %10.1f ms\n", pFrameData->timeStamp/10.0f);
    printf("  Accelerometer: [%8.5f, %8.5f, %8.5f]\n", pFrameData->accelData[0],
                                                       pFrameData->accelData[1],
                                                       pFrameData->accelData[2]);
    printf("  Gyro:          [%8.5f, %8.5f, %8.5f]\n", pFrameData->gyroData[0],
                                                       pFrameData->gyroData[1],
                                                       pFrameData->gyroData[2]);
    printf("  Temperature:   %8.6f C\n", pFrameData->tempData);
    printf("-----------------------------------------------------\n");
}
```

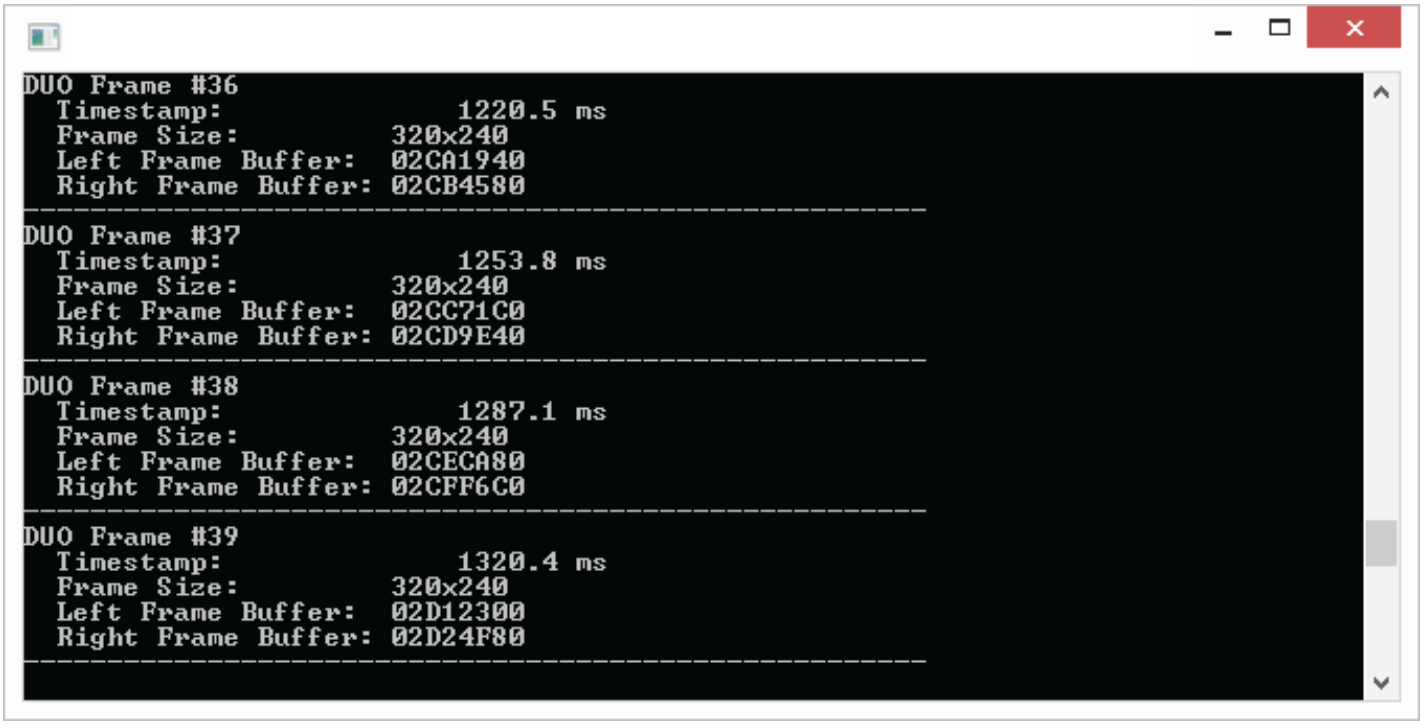
```
int main(int argc, char* argv[])
{
    printf("DUOLib Version:       v%s\n", GetLibVersion());
    DUOResolutionInfo ri;
    // Select 320x240 resolution with 2x2 binning capturing at 10FPS
    if(EnumerateResolutions(&ri, 1, 320, 240, DUO_BIN_HORIZONTAL2+DUO_BIN_VERTICAL2, 30))
    {
        DUOInstance duo;
        // Open DUO
        if(OpenDUO(&duo))
        {
            char tmp[260];
            // Get some DUO parameter values
            GetDUODeviceName(duo, tmp);
            printf("DUO Device Name:      '%s'\n", tmp);
            GetDUODeviceName(duo, tmp);
            printf("DUO Serial Number:    %s\n", tmp);
            GetDUOFirmwareVersion(duo, tmp);
            printf("DUO Firmware Version: v%s\n", tmp);
            GetDUOFirmwareBuild(duo, tmp);
            printf("DUO Firmware Build:   %s\n", tmp);

            printf("\nHit any key to start capturing");
            _getch();

            // Set selected resolution
            SetDUOResolutionInfo(duo, ri);
            // Start capture and pass DUOCallback function that will be called on every frame
            if(StartDUO(duo, DUOCallback, NULL))
            {
                // Wait for any key
                _getch();
                // Stop capture
                StopDUO(duo);
                // Close DUO
                CloseDUO(duo);
            }
        }
    }
    return 0;
}
```

Sample 02

Capturing Image Data

All DUO are capable of returning image frame data, this happens on callback which is passed the `pFrameData` from the device. In this structure you can access all sensor data, specifically in this example we focus on these variables:

- `pFrameData->width` - Integer of the current total frame width.
- `pFrameData->height` - Integer of the current total frame height.
- `pFrameData->leftData` - Contains the image data from the left sensor.
- `pFrameData->rightData` - Contains the image data from the right sensor.




```
#include "Sample.h"
int duoFrameNum = 0;
void CALLBACK DUOCallback(const PDUOFrame pFrameData, void *pUserData)
{
    printf("DUO Frame #%d\n", duoFrameNum++);
    printf("  Timestamp:          %10.1f ms\n", pFrameData->timeStamp/10.0f);
    printf("  Frame Size:         %dx%d\n", pFrameData->width, pFrameData->height);
    printf("  Left Frame Buffer:  %p\n", pFrameData->leftData);
    printf("  Right Frame Buffer: %p\n", pFrameData->rightData);
    printf("-----------------------------------------------------\n");
}
```

```
int main(int argc, char* argv[])
{
    printf("DUOLib Version:       v%s\n", GetLibVersion());

    DUOResolutionInfo ri;
    // Select 320x240 resolution with 2x2 binning capturing at 10FPS
    if(EnumerateResolutions(&ri, 1, 320, 240, DUO_BIN_HORIZONTAL2+DUO_BIN_VERTICAL2, 30))
    {
        DUOInstance duo;
        // Open DUO
        if(OpenDUO(&duo))
        {
            char tmp[260];
            // Get some DUO parameter values
            GetDUODeviceName(duo, tmp);
            printf("DUO Device Name:      '%s'\n", tmp);
            GetDUODeviceName(duo, tmp);
            printf("DUO Serial Number:    %s\n", tmp);
            GetDUOFirmwareVersion(duo, tmp);
            printf("DUO Firmware Version: v%s\n", tmp);
            GetDUOFirmwareBuild(duo, tmp);
            printf("DUO Firmware Build:   %s\n", tmp);

            printf("\nHit any key to start capturing");
            _getch();

            // Set selected resolution
            SetDUOResolutionInfo(duo, ri);
            // Start capture and pass DUOCallback function that will be called on every frame
            if(StartDUO(duo, DUOCallback, NULL))
            {
                // Wait for any key
                _getch();
                // Stop capture
                StopDUO(duo);
                // Close DUO
                CloseDUO(duo);
            }
        }
    }
    return 0;
}
```

Sample 03

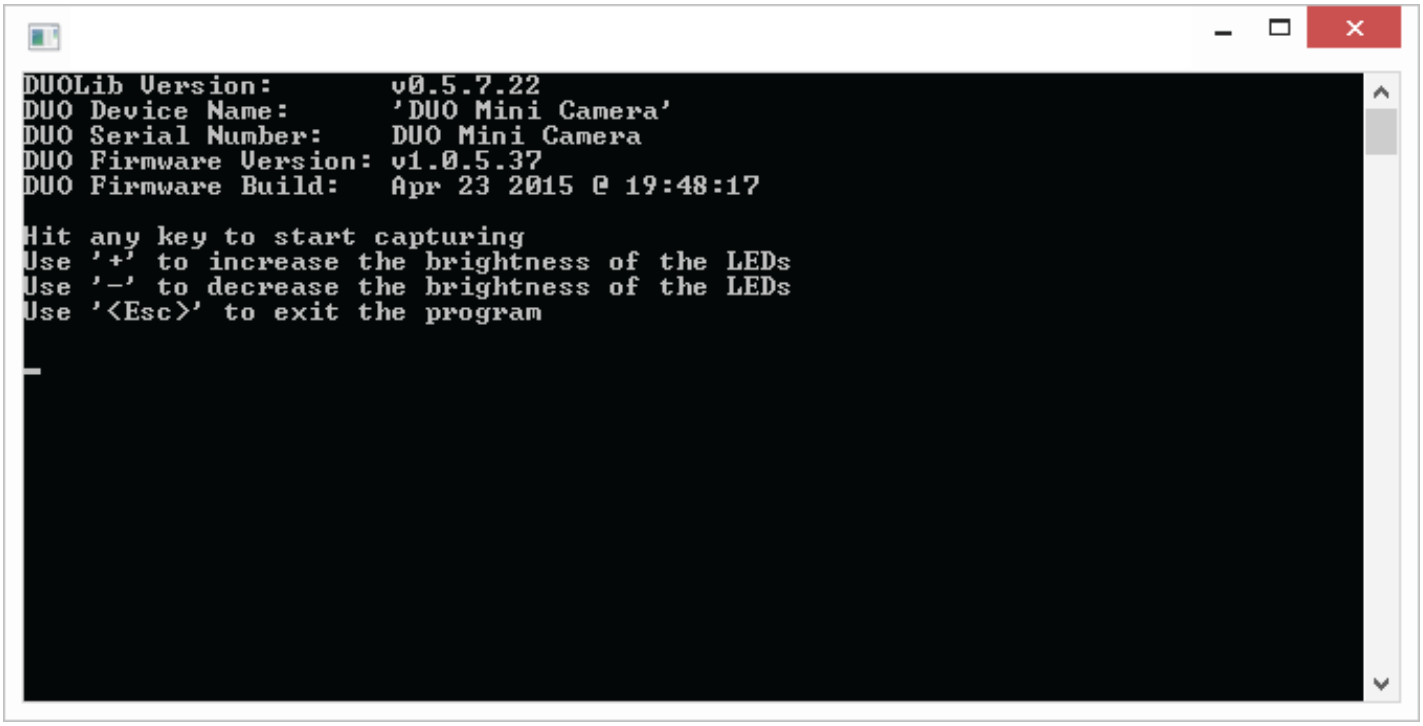
Configuring Parameters

In this example we show how to use the exposed methods to update the device configuration in real-time. Keep in mind that this code could be easy change to update different parameters such as Gain, Exposure and more. If your device features a programmable LED array you can change the brightness of the LEDs programmatically with the following code. Highlighting this specific variable from the following parameters:

- `SetDUOLedPWM` - Float value for pulse width management.

Others methods that could be used in similar manner:

- `SetDUOExposure` - Float value for camera exposure (0-100).
- `SetDUOGain` - Float value for camera gain (0-100).
- `SetDUOHFlip` - Boolean which determines if horizontal flip is enabled.
- `SetDUOVFlip` - Boolean which determines if vertical flip is enabled.
- `SetDUOCameraSwap` - Boolean to determine if camera positions are swapped.




```
int main(int argc, char* argv[])
{
    printf("DUOLib Version:       v%s\n", GetLibVersion());
    DUOResolutionInfo ri;
    // Select 320x240 resolution with 2x2 binning capturing at 10FPS
    if(EnumerateResolutions(&ri, 1, 320, 240, DUO_BIN_HORIZONTAL2+DUO_BIN_VERTICAL2, 30))
```

```
    {
        DUOInstance duo;
        // Open DUO
        if(OpenDUO(&duo))
        {
            char tmp[260];
            // Get some DUO parameter values
            GetDUODeviceName(duo, tmp);
            printf("DUO Device Name:      '%s'\n", tmp);
            GetDUODeviceName(duo, tmp);
            printf("DUO Serial Number:    %s\n", tmp);
            GetDUOFirmwareVersion(duo, tmp);
            printf("DUO Firmware Version: v%s\n", tmp);
            GetDUOFirmwareBuild(duo, tmp);
            printf("DUO Firmware Build:   %s\n", tmp);
            printf("\nHit any key to start capturing");
            _getch();
            printf("\n");

            // Set selected resolution
            SetDUOResolutionInfo(duo, ri);

            float ledPwm = 30;
            // Set the LED brightness value in %
            SetDUOLedPWM(duo, ledPwm);
            // Start capture (no callback function)
            if(StartDUO(duo, NULL, NULL))
            {
                printf("Use '+' to increase the brightness of the LEDs\n");
                printf("Use '-' to decrease the brightness of the LEDs\n");
                printf("Use '' to exit the program\n\n");
                int ch;
                do
                {
                    ch = _getch();
                    if(ch == '-') ledPwm > 0 ? ledPwm-- : 0;
                    else if(ch == '+') ledPwm < 100 ? ledPwm++ : 0;
                    printf("LED: %3d%%\r", (int)ledPwm);
                    SetDUOLedPWM(duo,ledPwm);
                }while(ch != 27);
                // Stop capture
                StopDUO(duo);
                // Close DUO
                CloseDUO(duo);
            }
        }
    }
    return 0;
}
```

Sample 04

Configuring LED Sequences

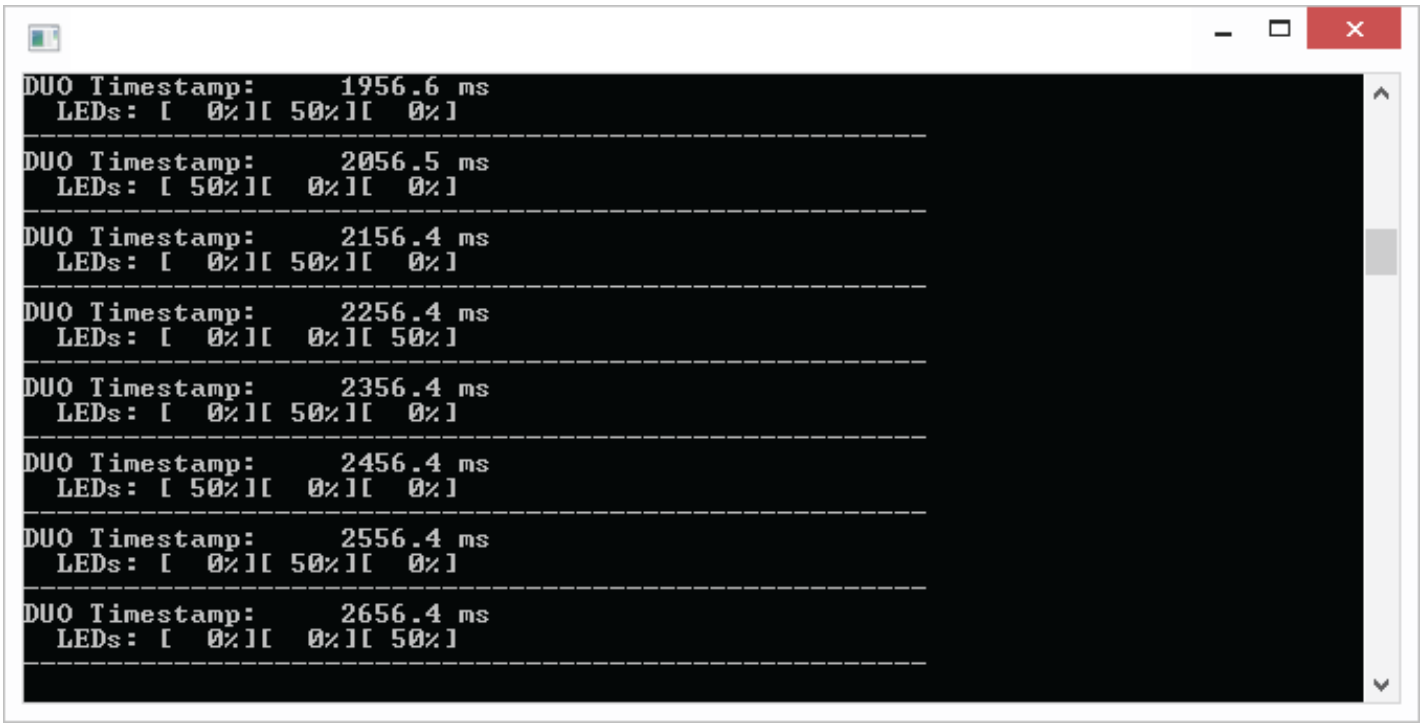
In this example we cover how expand on the control of the LED Array by passing it a specific sequence to follow, you can think of this as a pattern which the LEDs will fire. With the following code we create a sequence with four steps that first blinks the left LED and then middle and the right and finally back to middle.

Highlighting this specific parameters/structures used:

- `DUOLEDSeq` - The structure for passing sequences to PWM. Each LED having a unique position as shown:

```
DUOLEDSeq ledSequence[] = {{ LEFT,  MIDDLE,  RIGHT, ... }, ...};
```

- `LED_PWM_SEQ` - Number of LED sequence steps (max 64)
- `pFrameData->ledSeqTag` - Current status of the LEDs.
    - `ledPwmValue[0,1,2]` - Current LED values.




```
#include "Sample.h"
DUOLEDSeq ledSequence[] = {
    { 50,  0,  0, },
    {  0, 50,  0, },
    {  0,  0, 50, },
    {  0, 50,  0, },
```

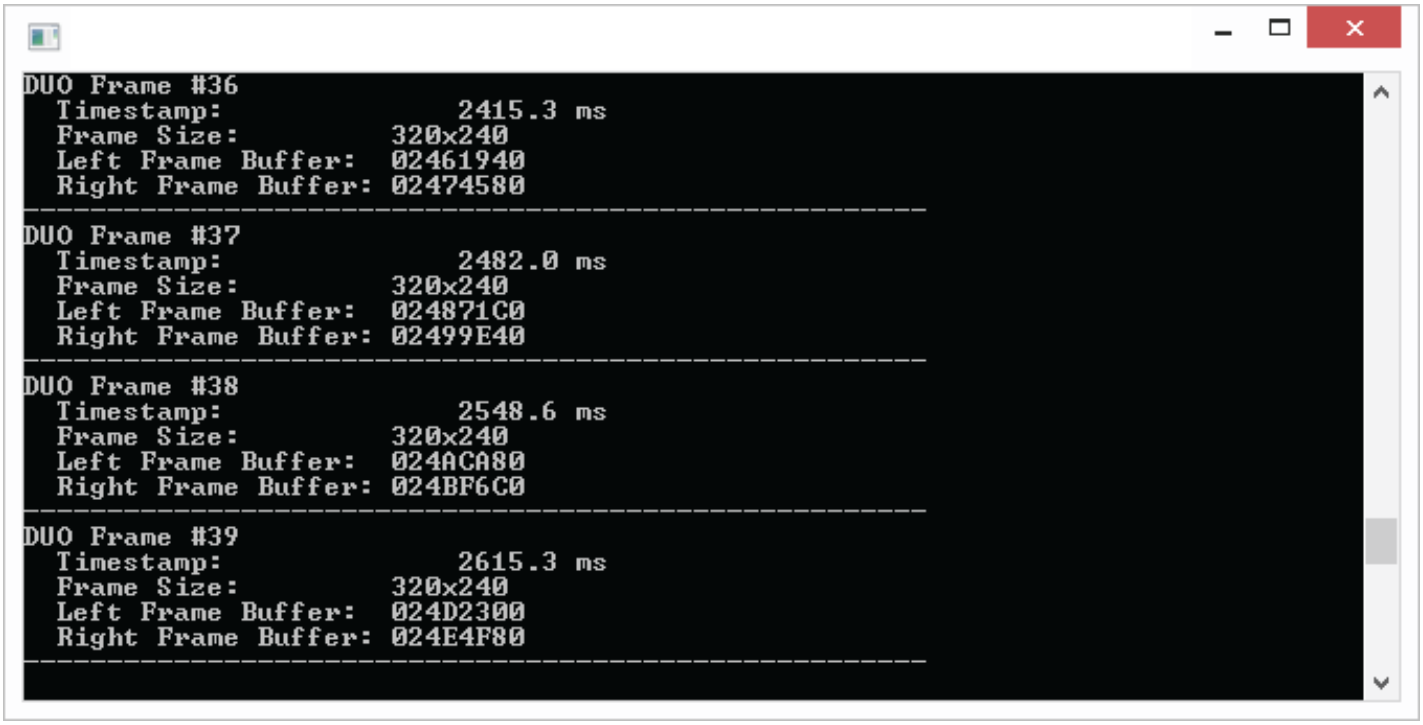
```
};
void CALLBACK DUOCallback(const PDUOFrame pFrameData, void *pUserData)
{
    printf("DUO Timestamp: %10.1f ms\n", pFrameData->timeStamp/10.0f);
    printf("  LEDs: [%3d%%][%3d%%][%3d%%]\n",
            ledSequence[pFrameData->ledSeqTag].ledPwmValue[0],
            ledSequence[pFrameData->ledSeqTag].ledPwmValue[1],
            ledSequence[pFrameData->ledSeqTag].ledPwmValue[2]);
    printf("-----------------------------------------------------\n");
}
int main(int argc, char* argv[])
{
    printf("DUOLib Version:       v%s\n", GetLibVersion());
    DUOResolutionInfo ri;
    // Select 320x240 resolution with 2x2 binning capturing at 10FPS
    if(EnumerateResolutions(&ri, 1, 320, 240, DUO_BIN_HORIZONTAL2+DUO_BIN_VERTICAL2, 10))
    {
        DUOInstance duo;
        // Open DUO
        if(OpenDUO(&duo))
        {
            char tmp[260];
            // Get some DUO parameter values
            GetDUODeviceName(duo, tmp);
            printf("DUO Device Name: '%s'\n", tmp);
            GetDUODeviceName(duo, tmp);
            printf("DUO Serial Number: %s\n", tmp);
            GetDUOFirmwareVersion(duo, tmp);
            printf("DUO Firmware Version: v%s\n", tmp);
            GetDUOFirmwareBuild(duo, tmp);
            printf("DUO Firmware Build:   %s\n", tmp);

            printf("\nHit any key to start capturing");
            _getch();
            printf("\n");

            // Set selected resolution
            SetDUOResolutionInfo(duo, ri);
            // Set the LED sequence
            SetDUOLedPWMSeq(duo, ledSequence, sizeof(ledSequence)/sizeof(DUOLEDSeq));
            // Start capture (no callback function)
            if(StartDUO(duo, DUOCallback, NULL))
            {
                _getch();
                // Stop capture
                StopDUO(duo);
                // Close DUO
                CloseDUO(duo);
            }
        }
    }
    return 0;
}
```

Sample 05

Capture frames using polling mechanism

Demonstrates polling mechanism for capturing frames.

Utilizing all highlighted methods/structures from Samples 1-4.



Sample.h

```
#ifndef SAMPLE_H
#define SAMPLE_H

// Platform Specific
...

// Include DUO API header file
#include "DUOLib.h"

// Some global variables
static DUOInstance _duo = NULL;
static PDUOFrame _pFrameData = NULL;

// Platform Specific
...
// One and only duo callback function
```

```
// It sets the current frame data and signals that the new frame data is ready
static void CALLBACK DUOCallback(const PDUOFrame pFrameData, void *pUserData)
{
    _pFrameData = pFrameData;
    SetEvent(_evFrame);
}
// Opens, sets current image format and fps and starts capturing
static bool OpenDUOCamera(int width, int height, float fps)
{
    if(_duo != NULL)
    {
        // Stop capture
        StopDUO(_duo);
        // Close DUO
        CloseDUO(_duo);
        _duo = NULL;
    }
    // Find optimal binning parameters for given (width, height)
    // This maximizes sensor imaging area for given resolution
    int binning = DUO_BIN_NONE;
    if(width <= 752/2)
        binning += DUO_BIN_HORIZONTAL2;
    if(height <= 480/4)
        binning += DUO_BIN_VERTICAL4;
    else if(height <= 480/2)
        binning += DUO_BIN_VERTICAL2;

    // Check if we support given resolution (width, height, binning, fps)
    DUOResolutionInfo ri;
    if(!EnumerateResolutions(&ri, 1, width, height, binning, fps))
        return 0;

    if(!OpenDUO(&_duo))
        return 0;

    char tmp[260];
    // Get and print some DUO parameter values
    GetDUODeviceName(_duo,tmp);
    printf("DUO Device Name:      '%s'\n", tmp);
    GetDUOSerialNumber(_duo, tmp);
    printf("DUO Serial Number:    %s\n", tmp);
    GetDUOFirmwareVersion(_duo, tmp);
    printf("DUO Firmware Version: v%s\n", tmp);
    GetDUOFirmwareBuild(_duo, tmp);
    printf("DUO Firmware Build:   %s\n", tmp);

    // Set selected resolution
    SetDUOResolutionInfo(_duo, ri);

    // Start capture
    if(!StartDUO(_duo, DUOCallback, NULL))
        return 0;
    return true;
}
```

```
// Waits until the new DUO frame is ready and returns it
static PDUOFrame GetDUOFrame()
{
    if(_duo == NULL)
        return 0;
    if(WaitForSingleObject(_evFrame, 1000) == WAIT_OBJECT_0)
        return _pFrameData;
    else
        return NULL;
}

// Stops capture and closes the camera
static void CloseDUOCamera()
{
    if(_duo == NULL)
        return;
    // Stop capture
    StopDUO(_duo);
    // Close DUO
    CloseDUO(_duo);
    _duo = NULL;
}
#endif // SAMPLE_H
```

Sample.cpp

```
#include "Sample.h"
int main(int argc, char* argv[])
{
    printf("DUOLib Version:       v%s\n", GetLibVersion());

    // Open DUO camera and start capturing
    if(!OpenDUOCamera(320, 240, 15))
    {
        printf("Could not open DUO camera\n");
        return 0;
    }

    int duoFrameNum = 0;

    // Run capture loop until  key is pressed
    while(!_kbhit() || _getch() != 27)
    {
        // Capture DUO frame
        PDUOFrame pFrameData = GetDUOFrame();

        // Process DUO frame
        if(pFrameData != NULL)
        {
            printf("DUO Frame #%d\n", duoFrameNum++);
            printf("  Timestamp:          %10.1f ms\n", pFrameData->timeStamp/10.0f);
            printf("  Frame Size:         %dx%d\n", pFrameData->width, pFrameData->height);
            printf("  Left Frame Buffer:  %p\n", pFrameData->leftData);
            printf("  Right Frame Buffer: %p\n", pFrameData->rightData);
            printf("--------------------------------------------------\n");
        }
    }

    // Close DUO camera
    CloseDUOCamera();
    return 0;
}
```

Resources

Related

- DUO API
- DUO SDK
- DUO Developers
- DUO Devices
- DUO Downloads