ULS24 Solution Kit SDK Readme

Release date: May, 12, 2017

# Introduction

This is an SDK for ULS24 CMOS sensor device from Anitoa Systems. The SDK is provided in source form and can be built with Microsoft Visual Studio 2010 edition or later.

The SDK is provided as a Bare-bone test with a single dialog box as the interface. The project name is TestBB (BB stands for “bare bone”). The user can customize this demo program to produce a useable application, or extract the relevant files from this demo to construct a new application.

# List of files

Below is a list of the files and their main function:

**TestBB (.h, .cpp):** This is the windows-based application frame work. There is not much content in there.

**TestBBDlg(.h, .cpp):** This is the dialog box that pops up when the application starts. It is primarily the graphic user interface (GUI) of the demo. The drawing method is implemented in this module. Users can customize the drawing program in this module if so wish. Users can also add other GUI features such buttons in this dialog box to customize the demo.

**HidMgr(.h, .cpp):** This module manages the USB communication with the ULS24 development board. These files will be managed by Anitoa should not be touched by the user.

**TrimReader (.h, .cpp):** This module manages the loading of Trim.dat file (a file provided by Anitoa with every ULS24 development kit). It also manages the communication with ULS24 development board. These files will be managed by Anitoa should not be touched by the user.

**InterfaceObject (.h, cpp):** The InterfaceObject is provided for the user to add his/her custom implementations. The user can expand the function of this class to fulfill the application needs, or create a new class that is called by this class to implement additional features.

# Instructions for customize the application.

Most of the customizable application code is in InterfaceObject.cpp. A set of interface APIs is implemented within the CInterfaceObject class.

## Routines to perform basic ULS24 chip operations

First of all, the user application can call the member functions below to perform basic operation of the ULS24 device:

void CaptureFrame12(); // Capture a 12X12 image
void CaptureFrame24(); // Capture a 24X24 image

The data of the captured image will be stored in member variable.

int frame\_data[MAX\_IMAGE\_SIZE][MAX\_IMAGE\_SIZE];

In addition, the user can perform additional actions to display the captured image or raw data in the GUI.

void DisplayImage(int contrast);         // Display image in GUI

 void DisplayFrameText();                         // Display frame data in GUI text window

void InsertText(String s);                       // Insert a string in GUI text window

## Routines to adjust operation parameters of ULS24

Below are routines that provide interfaces for application to adjust variables operating parameters (trim data) of the sensor chip. Please see ULS24 Solution Sheet Datasheet for further explanation of these parameters.

 void SetGainMode(int); // Set the gain mode of the ULS24 device: 0: high gain mode; 1: low gain mode.

 void SetTXbin(BYTE txbin); // Set the TX binning pattern of the ULS24 device: Tx Binning pattern: 0x0 to 0xf.

 void SetIntTime(float); // Set the integration time of the ULS24 device in ms. Legal values are 1ms to 66000 ms.

 void SetV15(BYTE v15); // Set the V15 parameters of the ULS24 device. This is normally not changed by the user but load from the trim data file.

 void SetV20(BYTE v20); // Set the V20 parameters of the ULS24 device. This is normally not changed by the user but load from the trim data file.

 void SetRangeTrim(BYTE range); // Set the Range trim parameters of the ULS24 device. This is normally not changed by the user but load from the trim data file.

 void SetRampgen(BYTE rampgen); // Set the Rampgen parameters of the ULS24 device. This is normally not changed by the user but load from the trim data file.

## Routines to insert user customization code

Although there is not a “main” function in this windows-based program, which is event driven. To insert custom code for a user application, considering the following:

a. Create an event to trigger a procedure. We have included 3 buttons A, B and C. Pressing these buttons will trigger functions:

    void OnButtonA();                    // Automatically called when button A is pressed
    void OnButtonB();                    // Automatically called when button A is pressed
    void OnButtonC();                    // Automatically called when button A is pressed

You can add your code to these functions and have them triggered by button press.

b. If you want your code to execute on power up or camera insertion, you can add your code to

    void Initialize();                    // Automatically called when power up or HW inserted

c. You can also create a timer to trigger event and direct that to a function where you insert your code. This is supported by windows (examples not shown).

d. You can check the status of whether the device is detected or get the name of the chip.

    int IsDeviceDetected();                // 0: Device not detected; 1: device detected.

 CString  GetChipName();                 // Get the name of the chip embedded in trim.dat file

# Compile and run the SDK demo code

The user can load the demo code in Visual Studio and build the project in either “Debug” or “Release” mode. The resultant .exe file will appear in the /Release and /Debug folder.

The demo can be executed directly from the Visual Studio environment. Please remember to include the trim.dat file that comes with your device in the /Trim directory of the SDK.

To distribute the executable code (\*.exe file), please remember to always include trim.dat in the /Trim directory along with the compiled executable file.