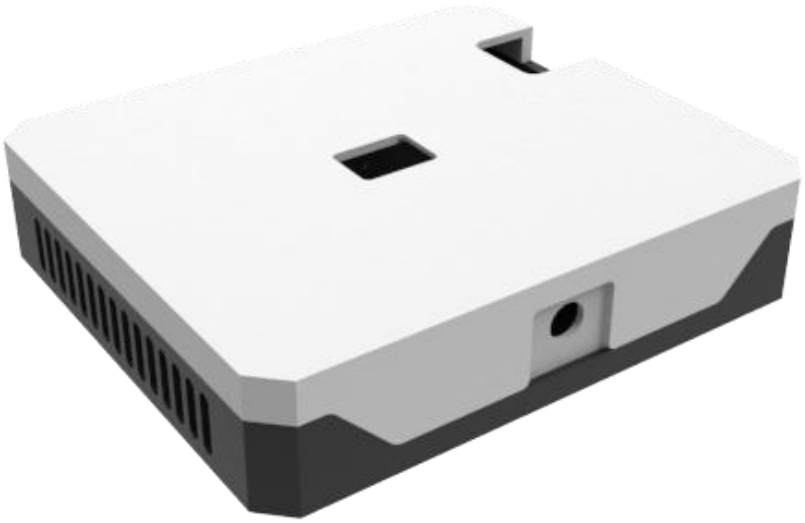
PAL-S 3D LiDAR

User Manual



UAE, Abu Dhabi
+ 971 5088 63660
support@sentiacu.com

Preface

This user manual contains the introduction, use and maintenance of PAL-S LiDAR. Please read this manual carefully before formal use, and strictly follow the steps described in the manual during use to avoid product damage, property loss, personal injury or/and violation of product warranty terms.

If you encounter problems that cannot be solved during use, please contact SentiAcu staff for assistance.

Contact Details

Official website: www.sentiacu.com

For technical questions, please contact: support@sentiacu.com

Copyright Notice

Disclaimer

The PAL-S product is constantly being improved, and its specifications and parameters will undergo iterative changes. Please refer to the official website for latest version.

Contents

1. Laser Safety Information

The LiDAR contains IR laser spots. IR laser: Wavelength 940nm;

Class 1 according to IEC 60825-1:2014, EN 60825-1:2014+A11:2021.

CAUTION !

Use of controls, adjustments or performance of procedures other than those specified herein may result in hazardous radiation exposure.

2. Installation and Maintenance

CAUTION !

This laser product is classified as Class 1 during operational procedures. When the ranging feature is activated, the laser emitter of the LiDAR module may emit laser radiation, therefore, the LiDAR should NOT be aimed at humans and animals to ensure safety.

This product is designed and calibrated for installation with exposed lenses. If a protective window needs to be added in front of the lens, it is necessary to ensure the use of materials with high transmission at 940nm wavelength and anti-reflective coating.

- Avoid the presence of smoke and fog in the detection field.
- Avoid condensation.
- Avoid direct exposure to moisture and water.
- Do not use rough fabric or dirty towels or aggressive products to clean the laser lenses.
- Do not use a supply voltage higher than the maximum required in the specifications to power the product.

- Clean the laser lenses with compressed air. When needed, wipe the laser lenses only with a soft, clean microfiber cloth.
- Make sure the sensor is securely mounted to prevent false readings or damage.
- Only trained and qualified personnel may install, setup and repair the LiDAR.

3. Product Overview

This chapter mainly introduces the measuring principle, technical specifications, structural description, field of view distribution of the PAL-S LiDAR.

3.1. Measuring principle

PAL-S is a ToF (time of flight) imaging sensor that integrates a 40x30 SPAD (single photon avalanche photo diode) pixel array, a front-end time-to-digital converter, and a depth processor. With a pulsed laser transmitter, the sensor can generate a 40x30 depth image. This user guide is intended to introduce setting development environment, the use of SDK and how to quickly obtain depth, point-cloud, etc. based on the SDK.

3.2. Technical specifications

Table. 1: Technical specifications

Performance Parameters	
Detection range	9.0m @90% ref. Indoor 7.5 m @10% ref. indoor
Blind zone	< 1 m
Accuracy①	±1% @＞1 m
Repeatability①	＜3 cm
Distance resolution	1 mm
Default frame rate②	17Hz, 60fps max.
Ambient light resistance	30Klux,performance will degrade under strong sunlight
Optical Parameters	
Light source	VCSEL
Central wavelength	940 nm
FoV	TYP. 58.5°×45.6°
Resolution	30×40 pixels
Eye safety	Class 1 [EN60825]

Mechanical Electrical	
Average power consumption	to be tested
Peak current	to be tested
Power supply	DC 5V
Data output interface	USB Type C
Operating temperature	-20℃ ~ +85°C
Dimensions	Evaluation kit: TYP. 70×80×17 mm^3 Future product: around 30×30×15 mm^3

NOTE:

1. Measured with 90% reflectivity white target board, indoor environment;

2. Current sample's default frame rate is 17Hz, 60Hz will be available in later versions.

3.3. Appearance

The current appearance of the LiDAR is as shown in the figure below:

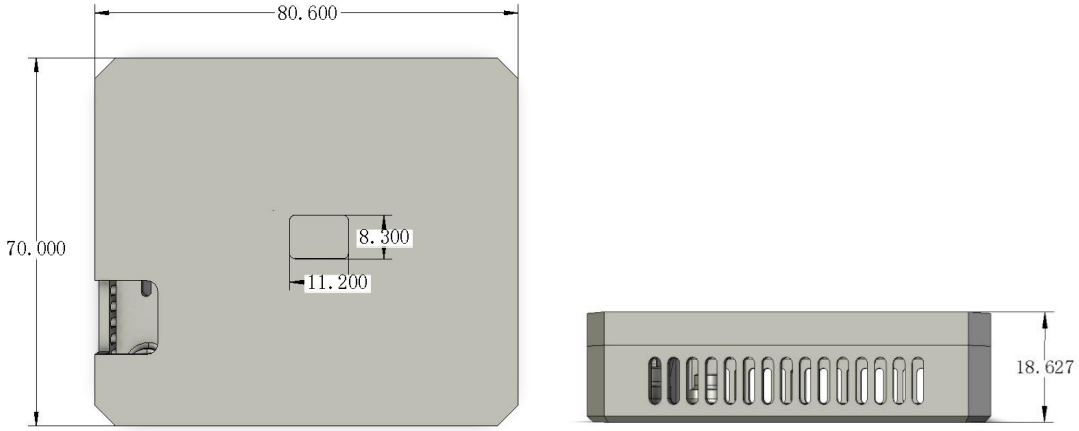


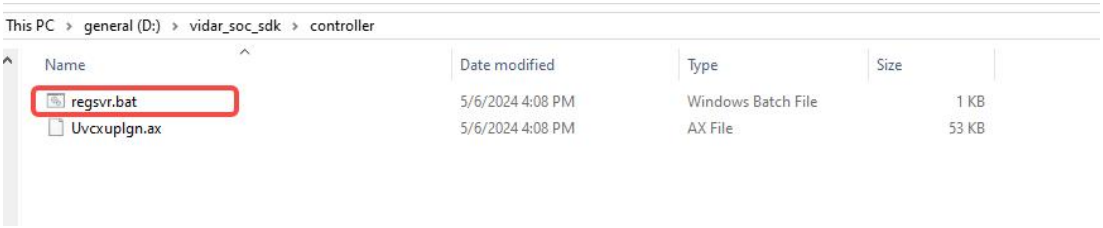
Figure. 1: PAL-S current Dimensions

4. SDK User Guide

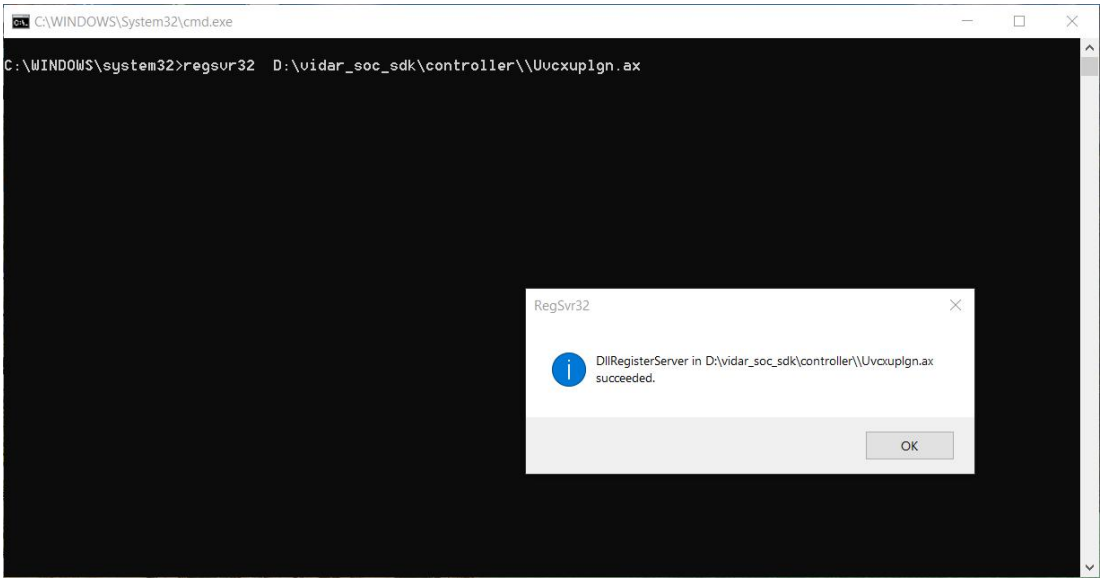
4.1. Registering Controller

Under Windows platform, you need to use the extension unit plug-in to complete the UVC XU function. Before running the SDK, you need to

register this Uvcxuplgn plug-in, otherwise the SDK interface will not work properly. Install regsvr.bat in the controller path as an administrator.



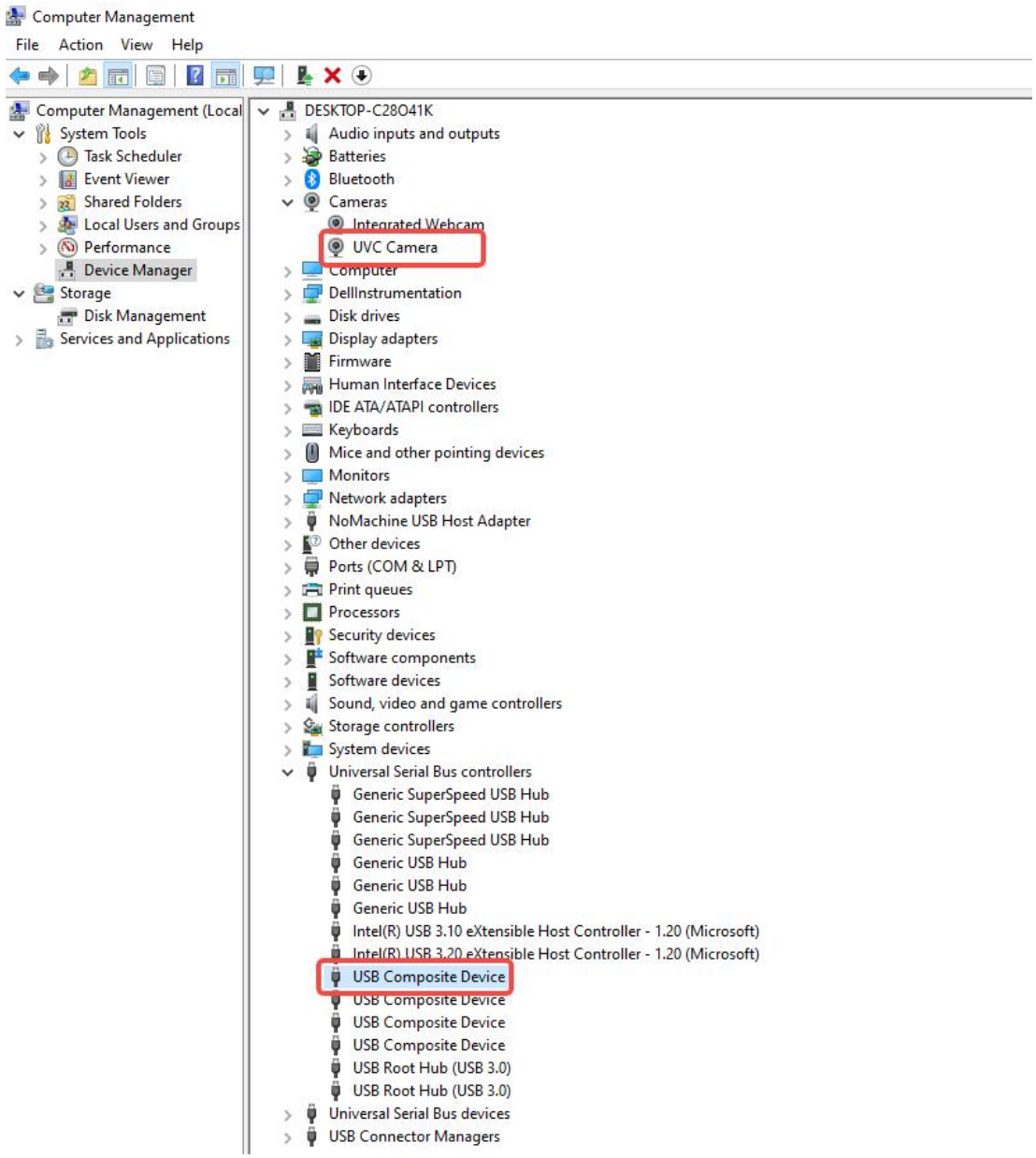
If the bat file is run successfully, you should see the following message in the terminal:



NOTE: please keep the path of controller as English, there should be no non-English character. Otherwise, the registration of controller might experience failure.

4.2.Interfacing with Sensor

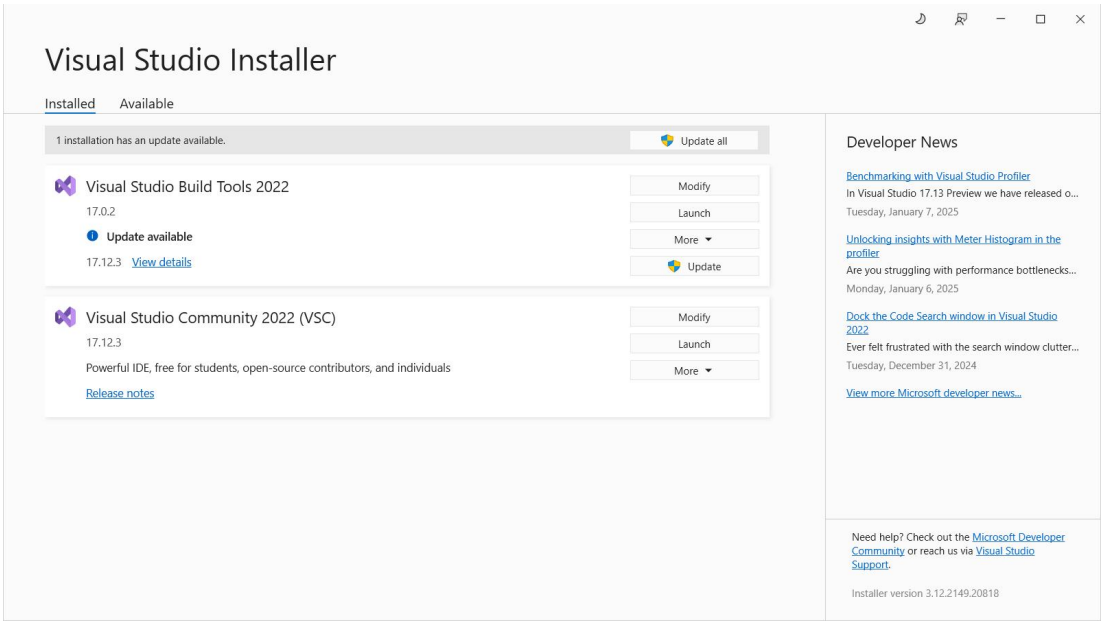
One end of the Type-c USB cable needs to be connected to the module (sensor), and the other end should be connected to the PC. Under Windows system, after the connection is successful, the USB serial device will appear in the device manager as shown below:



In the Cameras section, you should see UVC Camera. A new device USB Composite Device will also appear in Universal Serial Bus controllers. In my case, I have multiple USB Composite Devices, but when you connect the module, you should see the new device appeared.

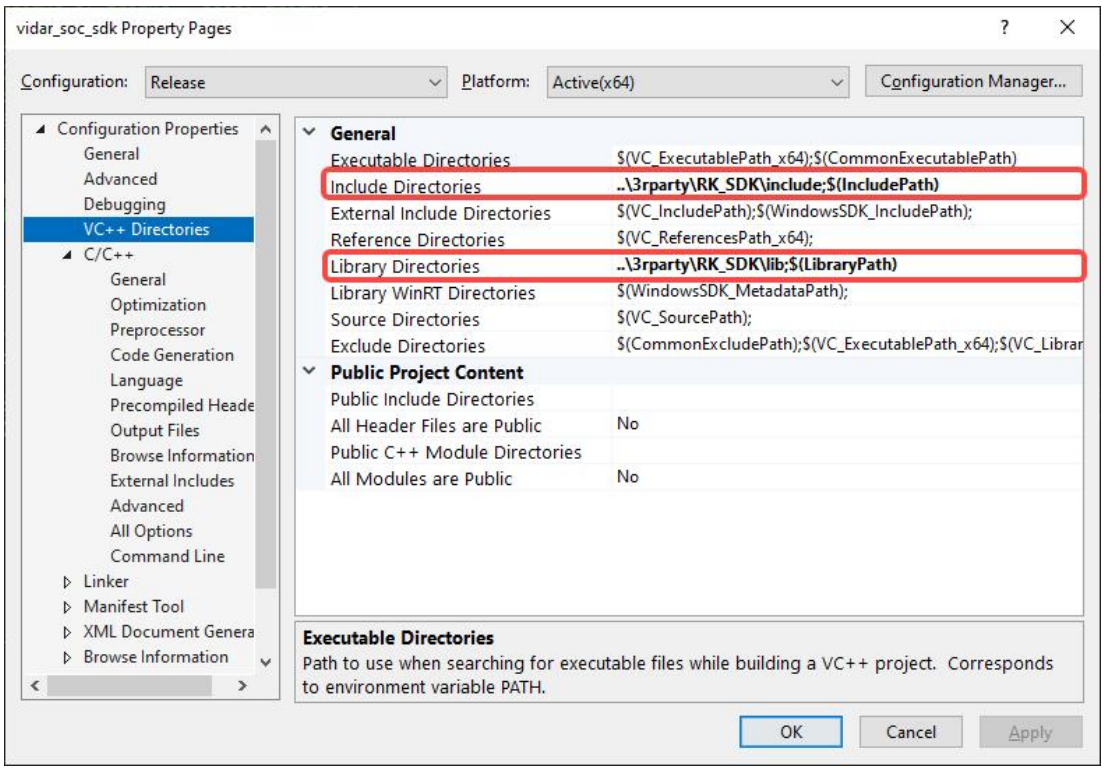
4.3.Compiling SDK

In order to compile the SDK, you need to have install Visual Studio 2022 (IDE and Build Tools). The following image shows which tools and IDE need to be installed for building the SDK.



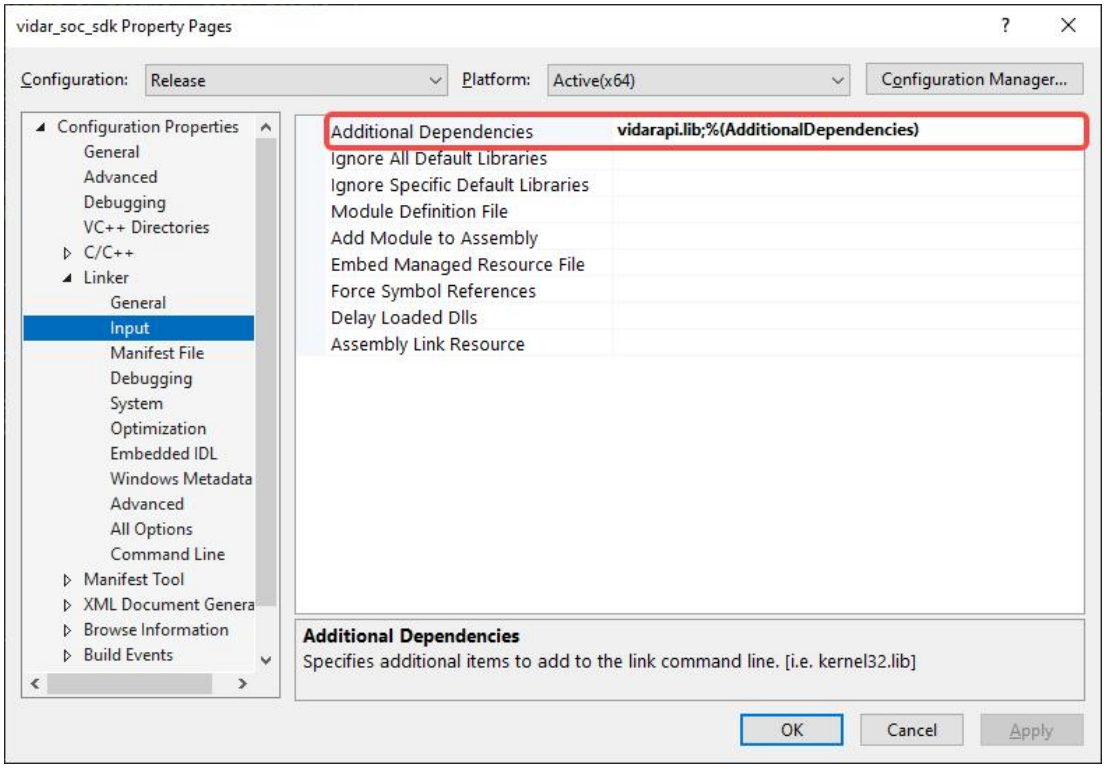
The major version of Build Tool and IDE should be the same. It is fine to have minor version as different.

Once you have installed the required build-tools and IDE, it's time to compile the SDK. Open vidar_soc_sdk.sln file in your Visual Studio IDE. To properly build the SDK, we need to make some configurations in our development environment setup (if they are not set by default). Right click on vidar_soc_sdk in the Solution Explorer of IDE and go to Properties option. You should see the following window:



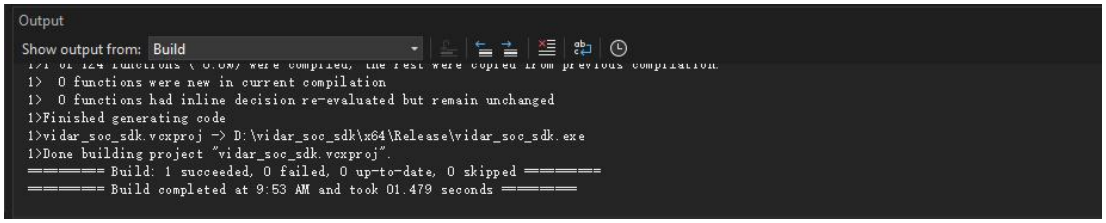
In VC++ Directories option, make sure that Include Directories and Library Directories are set according to the above image.

Next, in the same window, click on Linker option and then Input. Additional Dependencies need to be set as shown below:



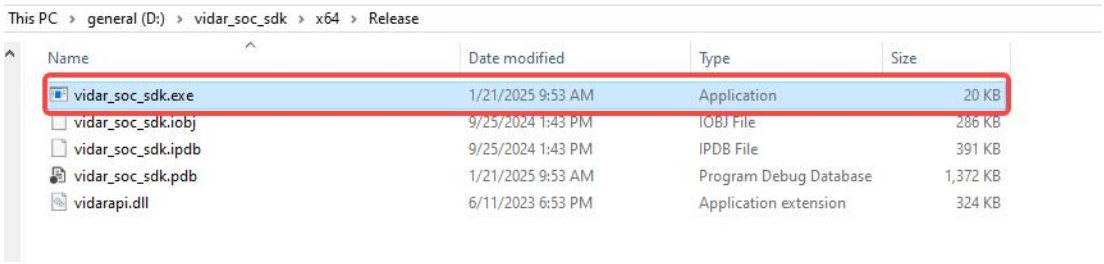
If everything is set, it's to compile the SDK. Click on play-button (Start Without Debugging) or press ctrl+F5:

If the SDK is compiled successfully, you should see the following message in Output window.

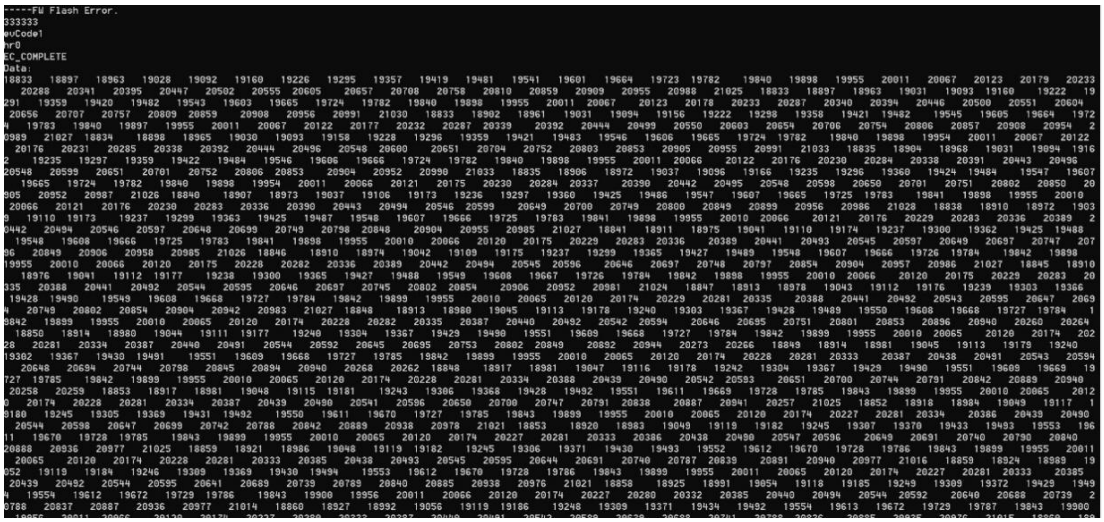


The compiled contents are saved in x64 folder under the main folder of SDK.

While the LiDAR module is connected, we need to run the file vidar_soc_sdk.exe as shown below:



When we execute the EXE file, the output can be observed in the terminal. For example, I printed Y data of a single frame.



You can play around with main.cpp file to print XYZ, depth-data and other parameters as well.

4.4. Useful API Definitions

```
/**
* @brief Gets the device list
* @return List of devices
*/
static std::vector<DeviceInfo *> *GetDeviceList();
//-------------------------------------------------------------------------------------
/**
* @brief Open the device
* @param[out] deviceinfo
* @param[out] status SUCCEEDED >= 0, FAILED < 0
* @return Device
* Window >> HRESULT
*/
static Device *OpenDevice(DeviceInfo *deviceinfo, int &status);
//-------------------------------------------------------------------------------------
/**
* @brief Create streaming of data from the device
* @return Stream
*/
virtual Stream *CreateStream() = 0;
//-------------------------------------------------------------------------------------
/**
* @brief Start the device. This starts the streaming of data from the device.
* @return 0 Success , <0 Fail
*/
virtual int Start() = 0;
//-------------------------------------------------------------------------------------
/**
* @brief Get the frame from the device
* @return Frame
*/
virtual Frame *GetFrame() = 0;
//-------------------------------------------------------------------------------------
/**
* @brief Stop the device. This makes the device to stop streaming.
*/
virtual int Stop() = 0;
//-------------------------------------------------------------------------------------
/**
* @brief Close the device
* @return 0 Success , <0 Fail
*/
virtual int Close() = 0;
//-------------------------------------------------------------------------------------
```