EAN-Lens DLL Development

2022-10-10

Exports: Export Summary Sheet

EULA: End User License Agreement

Web: sightlineapplications.com

Sales: sales@sightlineapplications.com

Support: support@sightlineapplications.com

Phone: +1 (541) 716-5137

CAUTION: Alerts to a potential hazard that may result in personal injury, or an unsafe practice that causes damage to the equipment if not avoided.

IMPORTANT: Identifies crucial information that is important to setup and configuration procedures.

Used to emphasize points or reminds the user of something. Supplementary information that aids in the use or understanding of the equipment or subject that is not critical to system use.

1 Overview

Customers can now implement their own custom lens command and control drivers. This document is an overview of building custom lens DLL. Familiarity with custom ARM development as outlined in EAN-ARM-Development-1500-3000-OEM or EAN-ARM-Development-4000-OEM is expected when reading this document.

This document covers the following topics:

- Compiling custom DLL based on *SLNopLens* (No Op) template.
- Deploying DLL onto target hardware.
- Using SightLine Command and Control (directly or via Panel Plus) to configure serial ports and load @DLL.
- Using SightLine Command and Control to communicate with the DLL to drive the lens such as zoom in and out and focus.

Fundamentals of lens control as well as specifics on using existing lens DLLs can be found in EAN-Lens-Focus-Control.

1.1 Additional Support Documentation

Additional Engineering Application Notes (EANs) can be found on the Documentation page of the SightLine Applications website.

The Panel Plus User Guide provides a complete overview of settings and dialog windows. It can be accessed from the Help menu of the Panel Plus application.

The Interface Command and Control (IDD) describes the native communications protocol used by the SightLine Applications product line. The IDD is also available as a PDF download on the Documentation page under Software Support Documentation.

1.2 SightLine Software Requirements

ⓘ **IMPORTANT:** The Panel Plus software version should match the firmware version running on the board. Firmware and Panel Plus software versions are available on the Software Download page.

1.3 Third Party Software

Tera Term (or PuTTY): SightLine recommends Tera Term for troubleshooting, debugging, and issuing commands on SightLine hardware.

2 Hardware Configuration

An example of a hardware configuration with the 4000-OEM is shown in Figure 1.

All OEM hardware platforms have a similar hardware and lens configuration.

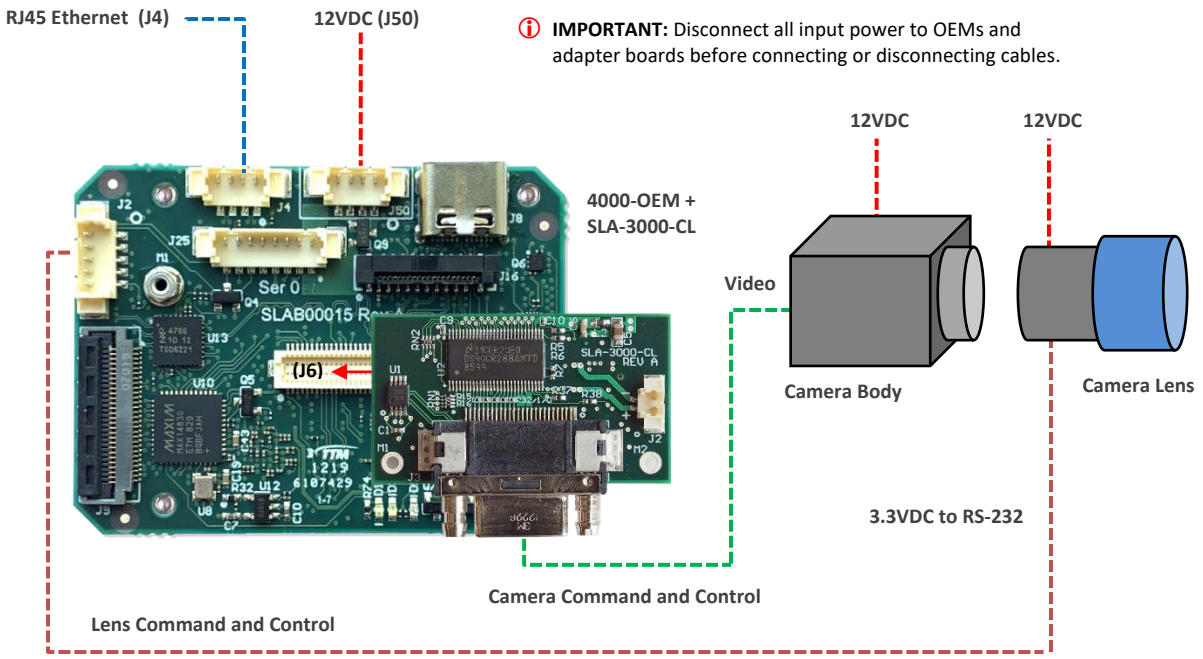


Figure 1: Example 4000-OEM with Camera and Lens Configuration

3 Lens Control Functionality

The interface that all lens DLLs must meet is defined in *sllens_public.h*. A summary of the key functions is shown in Table 1. For a complete description and examples see the source code provided with the examples.

Table 1: Key Functions

Function	Description
Initialize	Called when the lens DLL is loaded and can be used to start thread, initialize variables, etc.
Cleanup	Called when the lens DLL is unloaded. It can be used to clean up cleanly exit threads, close comm ports, etc.
ConfigureSerialPort	Called shortly after *Initialize* and can be used to configure comm ports. In most cases this can simply do the following: • If port is open close it • Create new SLARs232 object • Open port using SLARs232 object • Verify port opened successfully.

(Key Functions table continued)

ConfigureI2C	Currently not called, but it can be used to configure the I2C bus. The setup is similar to the *ConfigureSerialPort* function. This function is defined in the *sllens_public.h* header as a no-op. It does not need to be implemented by a custom DLL unless I2C is required for camera communications.
GetZoomRange	Gets the lens ranges for zoom to be returned in the LensRanges SVP packet. This must be implemented for the Panel Plus lens GUI slider bars to work correctly.
GetFocusRange	Gets the lens ranges for focus to be returned in the LensRanges SVP packet. This must be implemented for the Panel Plus lens GUI slider bars to work correctly.
ZoomStop	Sends the command to stop the lens zoom motor. This is called in response to an *SLALensCommand.*
ZoomWide	Sends the command to zoom out, with an optional argument for the motor speed. The speed argument is an unsigned 16-bit value, and the lens driver is responsible for ensuring it is within a valid range.
ZoomNarrow	Sends the command to zoom in with an optional argument for the motor speed. See notes on speed range under *ZoomWide*.
ZoomPos	This sends the command to set the zoom motor to a specific position. This is called in response to an *SLALensCommand.*
FocusStop	Sends the command to stop the lens focus motor. This is called in response to an *SLALensCommand.*
FocusNear	Sends the command to focus on a closer position, with an optional argument for the motor speed. The speed argument is an unsigned 16-bit value, and the lens driver is responsible for ensuring it is within a valid range.
FocusFar	Sends the command to focus or a further position with an optional argument for the motor speed. See notes on speed range under FocusNear.
FocusPos	Sends the command to set the focus motor to a specific position. This would be called in response to an *SLALensCommand.*
CmdPassthrough	Can be used to pass other commands through to the lens that are not part of the basic set of focus, zoom, commands. The passthrough command is called in response to *SLALensCommand*.
FocusAuto	If there is no built-in auto focus capability, do not implement it or return *SLA_NOP* and the default implementation of *SLA Auto Focus* will be used. This could also be used to implement a custom auto focus algorithm or enable lens built in auto focus, in this case also make sure SLA_SUCCESS is returned.
RequestStatus	Requests the zoom and focus position from the lens. This is called regularly by the built in SLA Auto Focus algorithm, but also on command. When the data is received it should be sent out using the *UpdateStatusCallback.*
Reset	Forces the lens to perform a reset. This would be called in response to an *SLALensCommand.*
NUC	Commands the camera to do a NUC if such a feature exists. Currently none of the SLA supported lenses have this feature. If this feature is not needed return *SLA_SUCCESS*.
CustomCommand	Intended for backwards compatibility with *SLLensMode lensMode* 128 - 255. SightLine recommends using *CmdPassthrough* instead. However, if using an interface that is sending older *lensMode* commands, implement this function to send this data to the lens.

4 Developing Lens DLL

The next sections cover the following processes:

- Downloading and installing the example code.
- Moving files to the target hardware.
- Connecting to the target hardware and building the DLL.
- Configuring VideoTrack to use the new DLL.

4.1 Installing Example Code

Download the *ARM Processor Code Examples* package from the Example Code page on the SightLine website. Launch the installer and follow the installation prompts.

An overview of relevant files is shown in Table 2.

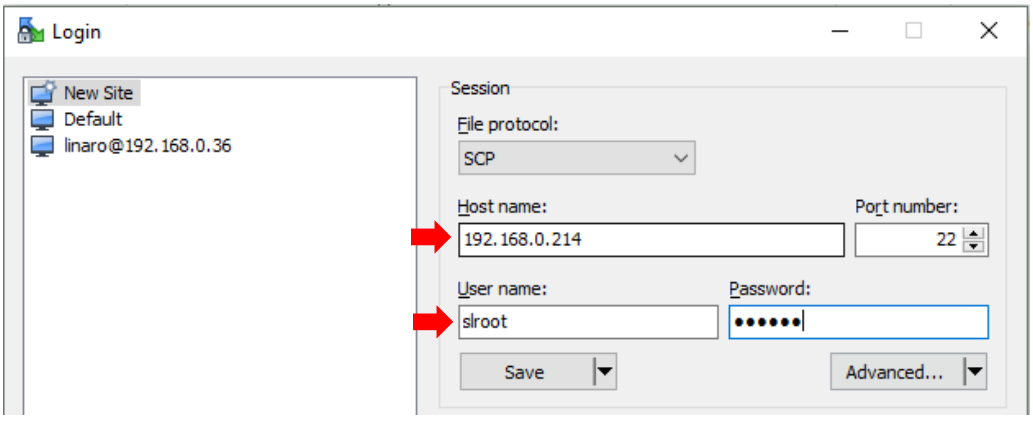
Table 2: Lens DLL Development Files

File	Description
lens_dll/sllens_public.h	Public interface for lens DLL development. Any custom lens drivers must meet this interface.
lens_dll/slNopLensCtrl.h lens_dll/slNopLensCtrl.cpp	Source code for *SLNopLens* driver. This is primarily *SLATrace* statements instead of real implementations.
lens_dll/slHitachiLens.h lens_dll/slHitachLens.cpp	Included as an example build. This is the same source code used by the SightLine Hitachi lens DLL and not built as part of the examples.
4000/SLAArmExamples.tgz	Compressed file to be used for development on the 4000. Includes *SLNopLens* driver and build scripts.
workspace/SLALensDLL1500	Example workspace used to build *SLNopLens* driver for 1500.
workspace/SLALensDLL3000	Example workspace used to build *SLNopLens* driver for 3000.

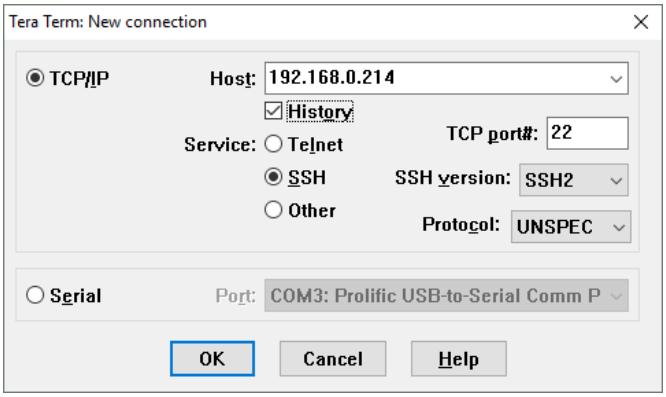
4.2 Build DLL - 4000-OEM

Copy examples to 4000-OEM using WinSCP:

1. Open WinSCP and login to the board.



2. Navigate to the folder where the ARM example is installed: *C:\SightLine Applications\SLA-Examples-Arm ***\4000*
3. Copy the file *SLAArmExamples.tgz* to /home/slroot/ on the 4000-OEM.
4. Use Tera Term to establish an SSH session to the 4000-OEM. Use the username and password: *slroot*.



5. Extract the samples by running the following command:

```
SD> tar -xvf SLAArmExamples.tgz
```

6. Navigate to the lens_dll directory *SD> cd SLAArmExamples/lens_dll*.
7. Build the example driver using the build script *SD> ./buildLensDll.sh*.
8. The output of this step will be an example lens DLL *slCustomLens.so.*
9. Copy the custom lens DLL to the lens directory SD> cp *slCustomLens*.so /home/slroot/sl/bin/lens/.

4.3 Build DLL - 1500-OEM and 3000-OEM

Building the DLL on the 1500-OEM and 3000-OEM can be done from Windows using Code Composer Studio.

The following procedure assumes the customer has read EAN-ARM-Development-1500-3000-OEM, and has set up a system as outlined in the Preparation section.

1. Open the Code Composer Studio application.
2. Select Workspace *C:\SightLine Applications\SLA-Examples-Arm ***\workspace*.
3. Select *View* » *Project Explorer*.
4. Select *File* » *Import.*
5. Expand *General*. Select *Existing Projects into Workspace* and click *Next*.
6. In the *Import Projects* dialog click *Browse* and navigate to (*C:\SightLine Applications\SLA-Examples-Arm ***\workspace*).
7. Select the desired project directory (*SLALensDLL1500* or *SLALensDLL3000*) and click *Finish*.
8. The project can now be built in debug or release using *Project* » *Build All* menu option.

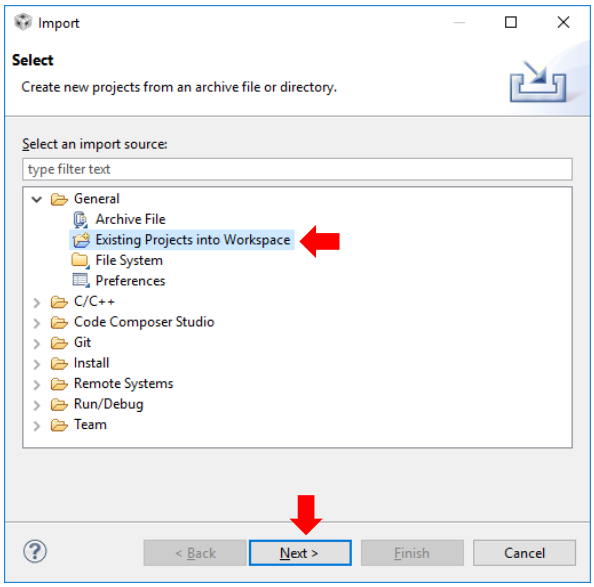


Figure 2: Importing a Project

Build notes:

- *A predefined CCS 5.x preferences file (ccs_eclipse_preference.epf) is in the workspace folder. Import this file via the Import dialog. Expand General and select Preferences.*
- *When importing preferences ensure that Import all is selected, and no other options are checked.*
- *Change between debug and release builds by right clicking on the project and selecting Build Configurations » Set Active.*
- *The DLL slCustomLens.so or slCustomLensDebug.so should automatically be copied to the board by the PostBuild**00.bat file if the SL_**00_IP is set up according to EAN-ARM-Development-1500-3000-OEM. It will be placed in the correct lens folder automatically.*

4.4 File Naming Syntax

All lens DLLs should use the same naming scheme <<filename>>.so with the following guidelines:

- No spaces in the file name.
- Filename including extension should be less than 128 characters.
- File should be placed in the “lens” folder, which is in the same directory as the VideoTrack application.
- Paths are not required when specifying the file name in the *SLALensParameters* message:
  - **Correct**: *MyLens02_03_04.so*
  - Incorrect: *MyLens02 03 04.so*
  - Incorrect: *./lens/MyLens02_03_04.so*

4.5 Configure and Load DLL

Once the lens DLL has been transferred to the hardware it can be configured and loaded as described in section 3 of EAN-Lens-Focus-Control.

5 Troubleshooting

Issue	Recommendation
Lens DLL not listed	• Verify the DLL has been copied to the Target. DLLs must be placed in the lens folder • Verify Lens DLL file name syntax

5.1 Questions and Additional Support

For questions and additional support, please contact SightLine Support. Additional support documentation and Engineering Application Notes (EANs) can be found on the Documentation page of the SightLine Applications website.