EAN-Command Example Code

2022-11-01

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

The example solution file in Visual Studio is designed to help software engineers develop code to interact with SightLine hardware platforms. The latest PC examples installer is available on SightLine Command and Control page.

The code examples provide a guide on generating and transmitting SightLine Command packets. This is important as it abstracts and simplifies creating message packets as defined by IDD. It also ensures that the packets will be correct and compatible with the Sightline hardware that will receive the commands.

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 located in 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 Software Downloads page.

1.2 Hardware Requirements

SightLine hardware with enabled functionality as described in one of the corresponding OEM startup guides: EAN-Startup Guide 1500-OEM, EAN-Startup Guide 3000-OEM, EAN-Startup Guide 4000-OEM.

1.3 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.

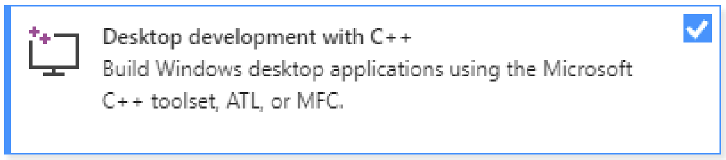
Table 1: Example Code Changes in SightLine Software

Software Version	Change
2.25.xx	Decode library was updated to use FFMPEG version 3.4.1. As part of this update ensure that linker settings in visual studio for SAFESH are set to No.
3.00.XX	Decode library was updated to use FFMPEG version 4.0.1. Hardware decoder option was added to the decoder. Requires Intel integrated graphics card.
3.01	Added option to have decoder allocate a group of buffers to be reused instead of requiring user to copy data out every frame. See Panel minus for an example of how to reuse buffers.

1.4 Third Party Tools - PC Examples 3.0.xx and Later

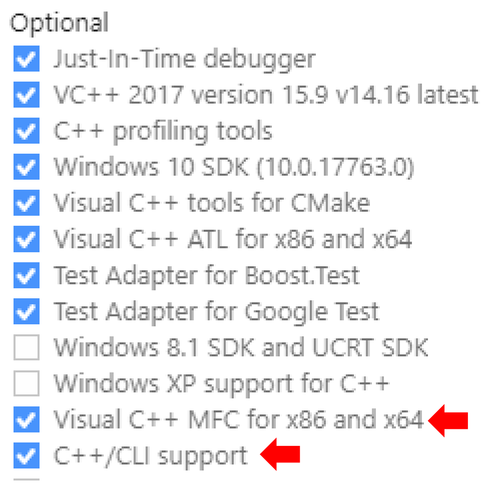
Microsoft Visual Studio Professional 2017 or similar.

1. Download and install Microsoft Visual Studio Professional 2017. During installation, in the *Workloads* tab, click on *Desktop development with C++*.



Desktop development with C++
Build Windows desktop applications using the Microsoft C++ toolset, ATL, or MFC.

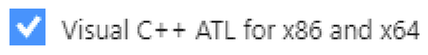
2. In the *Installation Details* » *Optional* section select *Visual C++ MFC for x86 and x64,* and *C++/CLI support*. Do not uncheck anything that is already checked.



Optional

- ✓ Just-In-Time debugger
- ✓ VC++ 2017 version 15.9 v14.16 latest v141 tools
- ✓ C++ profiling tools
- ✓ Windows 10 SDK (10.0.17763.0)
- ✓ Visual C++ tools for CMake
- ✓ Visual C++ ATL for x86 and x64
- ✓ Test Adapter for Boost.Test
- ✓ Test Adapter for Google Test
- ☐ Windows 8.1 SDK and UCRT SDK
- ☐ Windows XP support for C++
- ✓ Visual C++ MFC for x86 and x64
- ✓ C++/CLI support

3. Click the *Individual Components* tab. Scroll down to *SDKs, libraries, and frameworks* and select *Visual C++ ATL for x86 and x64* (this may already be checked by default)



✓ Visual C++ ATL for x86 and x64

4. Click *Install*.

Microsoft Visual Studio Installer Projects - Adds support for SightLine installer projects.

Qt 5.12.1 or QT 5.12.9 - Download and install QT. Used with the Panel Minus application. 3.2.xx and earlier use 5.12.1. 3.3.xx and later use 5.12.9.

QT Visual Studio Tools 2017 - Download and install after Qt 5.12.x above. Used for the Panel Minus application.

If upgrading from Microsoft Visual Studio 2013 with QT Visual Studio Tools 2013 see the Verify QT Options in Visual Studio section.

1.5 Third Party Tools - PC Examples 2.0.xx and earlier:

- Microsoft Visual Studio 2013 recommended.
- Microsoft Visual Studio 2013 Installer Projects - Necessary for building the *SLAPanelMinusInstaller* project.
- VS2013 MBCS - Multibyte MFC Library for Visual Studio
- Qt 5.6.0 - Download and install. Used with the Panel Minus application.
- QT Visual Studio Tools 2013 - Download and install after QT 5.6.0. above is installed. Used for the Panel Minus application.

2 Getting Started

1. Install the PC examples installer file from the download (e.g., *SLAExamplesInstaller.msi*).
2. Use Windows File Explorer and navigate to the install directory. By default this will be: *C:\SightLine Applications\SLAExamples X.XX.XX\CmdCtrlExamples.* This will change with each release version.
3. Open the *slexample.sln* example solution file in Microsoft Visual Studio. The included projects implement a subset of the entire SightLine command and control protocol.

For a complete overview of the SightLine command and control interface see the IDD.

IMPORTANT: If using Visual Studio 2017 Community Edition and building *slexamples.sln* in release build configuration, it is important to point to the necessary merge module (*Microsoft_VC141_CRT_x86.msm*) or find another way to redistribute Microsoft dependencies.

IMPORTANT: For each installation add the following directory to the system path: C:\SightLine Applications\SLAExamples X.XX.XX\bin (X.XX.XX is the installed version).

2.1 Projects Overview

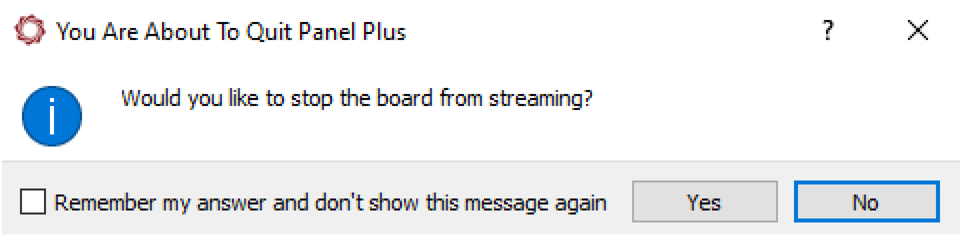
To debug/run a project, right click on the project in Visual Studio and select *Set as Startup Project*. The project is displayed in bold indicating it is the startup project, e.g., *SLAPanelMinus*.

Table 2: SightLine Projects

Folder	Included Files
bin	Contains Windows 32-bit DLLs for FFmpeg (avcodec, avdevice … dlls)
lib	Contains windows 32-bit libraries for FFmpeg
include	FFmpeg and SightLine header files
SLADecode	Defines the SLADecode class interface for decoding mjpeg, mpeg2-ts video and KLV metadata.
	SLATestGDI
	Sample application - decodes mpeg2-ts or rtp-mjpeg video from a network stream and displays in a GDI window.
	SLADecodePng
	Sample code utilizing **libpng** to decode PNG files and extract encoded metadata
CmdCtrlExamples	**simpleDiscover**
	This project uses an Ethernet connection to discover SightLine systems over an Ethernet network.
	slaCommand
	This project use either a Serial or Ethernet connection to demonstrates how to request and set parameters, and how to initiate and receive track telemetry.
	SLAPanelMinus
	This project is an example of how to include the SLADecoder into a Windows Qt application.
	SLAPanelMinusInstaller
	Example project for an installer to deploy the Panel Minus application.
	SLAVideoGrid
	This project is a Windows and QT application that displays video from up to 6 sources in a 2 row, 3 column grid. See the readme.txt file in the SLAVideoGrid project directory for details.
NucTableExample	Used to create or modify Non-Uniformity Correction (.nuc) or Dead Pixel Removal (.dead) tables for use with SightLine video processing systems.

2.2 SLATestGDI - Sample Video Decode Application

1. Open Panel Plus and connect to the SightLine on-board video processing system.
2. From the *Connect* tab set the *Video Output* to *Network* for the 1500 (*Network 1* or *0* for 3000).
3. From the *Compress* tab:
   a. Set *MPEG2-S* to *H.264*
   b. Click *Use My IP - Unicast.*
   c. Click *Send*.
4. Close the Panel Plus application. At the prompt to stop the board from streaming, select *No*.


You Are About To Quit Panel Plus

Would you like to stop the board from streaming?

☐ Remember my answer and don't show this message again

5. Browse to: *C:\SightLine Applications\SLAExamples X.XX.XX\SLADecode*.
6. Open *SLATestGDI.sln* with Visual Studio.
7. Build the solution, set *SLATestGDI* as StartUp Project, and then run *SLATestGDI.cpp*
8. From the main menu select *File* » *Change to Address udp://@:15004.*

To decode from a different address or port, open SLATestGDI.cpp and change the address passed to the SLADecode, defined as char ADDR[].

Examples include:

- MPEG2-TS default port, unicast to pc: *char ADDR[] = udp://@:15004*
- RTP-MJPEG default port, unicast to pc:
  - *char ADDR[] = udp://@:5004*
  - // Unicast to this PC (rtp-mjpeg default port)
- MPEG2-TS default multicast address and port: *char ADDR[] = udp://@224.10.10.10:15004*
- RTP-MJPEG default multicast address and port: *char ADDR[] = udp://@224.10.10.10:15004*
- RTSP client (release 2.24.7 and later): rtsp://192.168.1.102/clientPort=14560
- Decode a recorded file: *char ADDR[] = video1.ts*

video1.ts must be saved to C:\SightLine Applications\SLAExamples x.xx.xx\SLADecode\SLATestGDI.

If video is not displayed:

- Allow access to the network port used.
- Disable the windows firewall.
- Disable multiple network cards.
- Verify that other programs using the network video are closed (e.g., Panel Plus or VLC).

2.3 SLADecodePng

SightLine products can record 16-bit capture images with no loss to PNG files along with a custom EXIF-like header with metadata such as latitude and longitude. This sample code utilizes *libpng* to decode PNG files and extract the encoded metadata. See EAN-File-Recording for information about recording PNG Snap Shots.

2.4 Simple Discover Example (simpleDiscover)

This is a Windows console project that uses an Ethernet connection to discover SightLine systems over an Ethernet network. This SLADiscover is usually the first step in establishing network communication with a system for command and control. The SLADiscover protocol allows each system to inform the client regarding IP Address, hardware type, and features information. The SightLine Discover Protocol is described in the IDD.

2.5 SightLine Command and Control Example (slaCommand)

This is a Windows console project that uses either a Serial or Ethernet connection to a SightLine system. The code contains instructions on how to modify each connection type. After the connection is established, the application demonstrates how to request and set parameters and wait for a response. Reading information from the hardware and parsing is done using the message unpacking routines in slfip.cpp in a separate thread. Some command and control features will only be available with the corresponding Application Bits (appbits) enabled on the SightLine hardware. For questions regarding which features have been enabled, please contact Sales.

2.6 Panel Minus - Sample Graphical User Interface

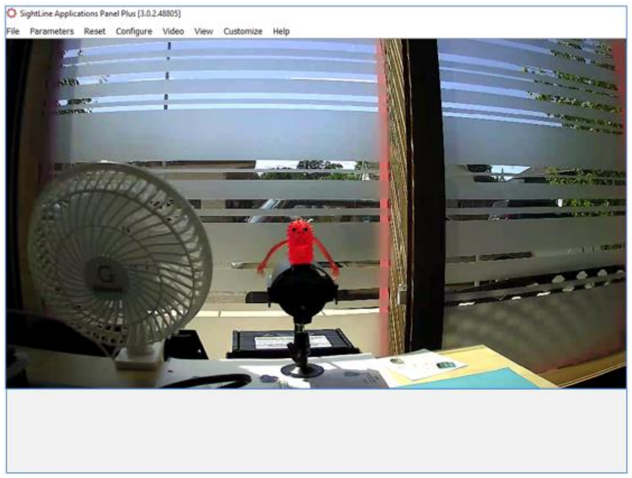
This project is an example of how to include the SLADecoder into a Windows Qt application. Installation of the Qt resources are required (see 1.4 Third Party Tools). For additional instructions see the Qt webpage downloads section for more information.

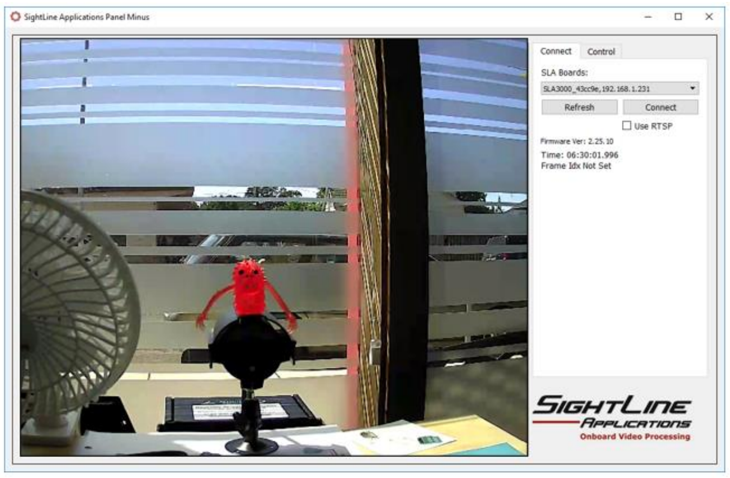
2.6.1 Panel Minus Notes

- Before using Panel Minus close all instances of Panel Plus. Closing Panel Plus will prevent it from consuming messages from SightLine hardware instead of Panel Minus.
- The default display area of SLADecode is approximately 640x480. If the camera input is larger (e.g., 1280x720), then only a 640x480 sub area of the screen is displayed. No scroll bars or other UI artifacts have been implemented to allow the user to see other parts of the image in this example.
- Clicking on *Start Track* sends three distinct track messages to the system via a call to SendStartTrack at the following pixel locations: (100,200) (200,300) (300,400)

Depending on the input resolution of the video this could be in an area of the screen not displayed in Panel Minus.

Figure 1 shows a full image displayed in Panel Plus. Figure 2 shows Panel Minus with a 640x480 center view of a full image.


SightLine Applications Panel Plus [3.0.2.48805]
File Parameters Reset Configure Video View Customize Help



SLA Boards: SLA3000_43cc9e,192.168.1.231
Firmware Ver: 2.25.10
Time: 06:30:01.996
Frame Idx Not Set

Figure 1: Panel Plus Full Image

Figure 2: Panel Minus 640x480 Full Image

2.6.2 Enabling RTSP Client Support

The latest version of SLA Decode (2.24.07 or latest) has built in support for a sample RTSP client. Run the *SLATestGDI.cpp* and enter the RTSP URL in the open URL drop-down menu to test the feature, e.g., *rtsp://192.168.1.102/clientPort=14560*.

Look in *SLARtspClient.cpp* for reference implementation in cases where the RTSP client needs modified. Call the *SLARtspOpenURL* with the correct URL path. All socket creation is done internally.

This happens in Panel Plus when streaming using RTSP. Look in *SLADecodeFFMpeg.cpp* in *ffmpegTask* for lines below. Do the same for the software.

The client uses one of the available ports in the system for RTSP. To configure a specific port, pass in the RTP port as a third parameter in the *SLARtspOpenUR*L (the client will try to use it). Another option is to use c*lientPort=* in the URL. The port does not need to be configured externally, since the decoder is aware that the port has been created.

2.6.3 Panel Minus Installer

Building the Panel Minus Installer provides a way to distribute a custom application without requiring users to install Visual Studio or Qt.

An alternate option to allow for seamless distribution is to install Panel Plus on a PC. Once installed copy the .dlls from the Panel Plus install location to the Panel Minus application location where panelminus.exe is running.

Before building the installer add all required .dlls from the c:\SightLine Applications\SLAExamples X.XX.XX\bin folder to the installer.

3 System Configuration Using Panel Plus and SightLine Commands

SightLine recommends using a combination of Panel Plus and SightLine commands to configure a system to start in a known state.

Use the following guidelines to help facilitate this process:

- Use Panel Plus to configure settings that are expected to remain constant for the application. Examples may include:
    - Acquisition settings
    - Serial communication settings
    - Network settings
    - Display settings (in certain applications)
- Use Panel Plus to configure the default settings for video encoding if applicable.

In most applications, the way the video is encoded (e.g., H.264) will not change because of the potential problems it can cause on the client receiver.

- Use Panel Plus to configure the default settings for other image processing functions if applicable, e.g., registration limits and stabilization rates.
- Save the default settings to the SightLine hardware. This creates a parameter file on the system that is loaded on every reboot cycle ensuring the system will start up in a known state.
- Look for the VersionNumber (0x40) message. To query version information, use the GetVersionNumber (0x00) command.
    - The VersionNumber (0x40) message is sent after the system starts up to indicate it is running and ready to receive commands.
    - The system may also begin streaming the telemetry packet TrackingPosition (0x51)depending on configuration and features enabled.
    - Check the major, minor, and release numbers that come back. If it is possible to upgrade the SightLine hardware asynchronous to the microcontroller firmware, then it is possible for the supported command and control protocol to be out of sync. This can cause some commands to no longer work with the system.
- Query the state of the system for information that may change over time, e.g., querying the system prior to changing the false color mode with SetDisplayParameters (0x16) command.

If the microcontroller starts up in sync with the SightLine system, this becomes less necessary.

4 Generating, Unpacking, and Parsing SightLine Command Packets

The example projects rely on packet generation and parsing provided by the source code in the following sections.

4.1 Communication Protocol Packet Generation and Unpacking

Packet generation and unpacking are implemented in *slfip.cpp, slfip.h*.

4.1.1 Packet Generation

Packet generation provides functions to generate most outgoing packets. These functions take parameters and generate a buffer of data to send out the communication port. It also implements correct data packing when generating outbound packets.

An example of generating and sending data is shown below. The calls that begin with *My* indicate customer generated calls.

```
#include "slfip.h"
u8 buffer[MAX_SLFIP_PACKET] = {0}
u8 dataLen = 0;
MyOpenSerialPort(COM1, 57600, 8, 1, NO_HW_HANDSHAKE);
dataLen = SLFIPStartTracking(buffer, 320, 240, 1);
MySerialPortWrite(buffer, dataLen);
```

When the SightLine commands are updated to add more parameters, there will be additional functions that correspond with the new parameter set. For example, if the *Set Registration Parameters* command was recently updated to include the camera index. The following commands are now implemented. (Both calls will work since SightLine firmware is backwards compatible.)

```
s32 SLFIPSetRegistrationParameters(SLPacketType buffer, u16 maxTranslation,
u8 maxRotation, u8 zoomRange=0, u8 lft=0, u8 rgt=0, u8 top=0, u8 bot=0);
s32 SLFIPSetRegistrationParameters(SLPacketType buffer, u16 maxTranslation,
u8 maxRotation, u8 zoomRange, u8 lft, u8 rgt, u8 top, u8 bot, u8 cameraIdx);
```

4.1.2 Packet Unpacking

Packet unpacking provides functions to unpack common incoming packets. These functions take a buffer of data and then fill in a structure containing the parameters. These functions should only be called in a callback routine for processing the matching command. For example, the SLFIPUnpackTrackingPosition() unpacks a buffer of data into the SLTelemetryData structure defined in *slfip.h*. The source code is in *slfip.cpp*. These are also good examples of how to unpack all commands. There are a limited number of these functions.

The following calls are currently implemented:

- (2.25.06) SLFIPUnpackTrackingPosition
- SLFIPUnpackTrackingPositions
- SLFIPUnpackTrackingPositionsExtended
- SLFIPUnpackTrackingPixelStats
- SLFIPUnpackFocusStats
- SLFIPUnpackLandingAid
- SLFIPUnpackLandingPosition

```
typedef struct {
  s16 trackCol;
  s16 trackRow;
  f32 sceneCol;
  f32 sceneRow;
  s16 displayCol;
  s16 displayRow;
  u8 trackingConfidence;
  u8 sceneConfidence;
  u16 displayAngle7;
  u8 idx;
  u8 reserved;
  s16 sceneAngle7;
  u16 sceneScale8;
} SLTelemetryData;

void SLACommManager::cbCurrentTrackingPosition(u8* data)
{
  SLTelemetryData trackPosition;
  u32 frameIdx = 0xFFFFFFFF;
  u64 timeStamp = 0;
  SLFIPUnpackTrackingPosition(&trackPosition, data, &timeStamp, 
&frameIdx);
  MyProcessTrackingPosition(&trackPosition);
}
```

4.2 Packet Parsing

Packet read and packet parse are supported.

4.2.1 Packet Read

The FIPReadPacket() reads and returns a complete packet from a port. This is included in *slfipport.cpp* and manages extended length bytes. An example of using a thread along with the FIPReadPacket() can be found in the *SLAPanelMinus* project in *SLAReceiveThread.spp.*

```
s32 FIPReadPacket(u8 *data, SLPort *port, s32 timeoutMs, bool fipEx, u8 
*extraHeaderByte)
void SLAReceiveThread::receive()
{
  // just read until we are told to quit:
  while( !m_quit )
  {
    packetLength = FIPReadPacket(&buffer[0], m_ReadingPort, timeout, 
fipEx);
    //if we got a whole packet, check the checksum:
    if( packetLength >0 && packetLength < 0xffffffff )
    {
…
```

4.2.2 Packet Parse

The SLParsePackets() example is included in *simpleslfipport.cpp.* This function takes a packet that has been read and calls the appropriate callback function.

```
u32 SLParsePackets(const u8 *data, u32 len, handlerCallback *callback,
u32 firstType, u32 nTypes)
```

5 Troubleshooting

Troubleshooting issues related to Visual Studio and third-party tools:

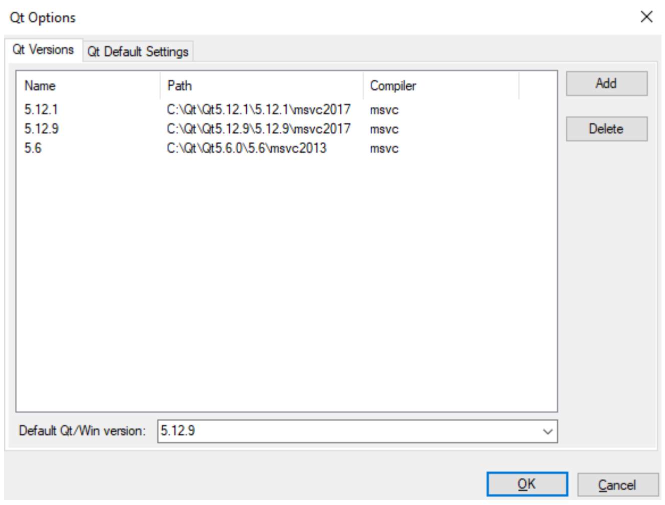
- To build and run projects, the QT directory and version must be specified. Check the environment variables for an entry for QTDIR. Set this to where the QT is installed, e.g., *C:\Qt\Qt5.6.0\5.6\msvc2013.*
- Delete *user file if edited manually.
- There can be problems building Qt projects such as SLAPanelMinus because it cannot find tools or dlls.
- Check the .user file and verify that QTDIR is defined and correct.
- Try manually adding QTDIR to *.user file* and restarting Visual Studio.

5.1 Verify QT Options in Visual Studio

Table 3: Visual Studio Software Compatibility

SightLine Software	Microsoft Visual Studio	QT Visual Studio Tools	QT Software Version
3.3.xx and above	VS 2017	QT VS Tools 2017	QT5.12.9
3.0.xx and above	VS 2017	QT VS Tools 2017	QT5.12.1
2.5.xx and below	VS 2013	QT VS Tools 2013	QT5.6.0

1. In Visual Studio, go to *QT VS Tools » Qt Options*.
2. Verify that the QT version that is installed (Figure 3) matches the software version in Table 3.
3. If the QT version is not correct, delete the entry and click *Add*. Browse to the path where the recent version of QT is installed and click *OK* (Figure 4).



Name	Path	Compiler
5.12.1	C:\Qt\Qt5.12.1\5.12.1\msvc2017	msvc
5.12.9	C:\Qt\Qt5.12.9\5.12.9\msvc2017	msvc
5.6	C:\Qt\Qt5.6.0\5.6\msvc2013	msvc

Default Qt/Win version: 5.12.9

Figure 3: QT Options Dialog

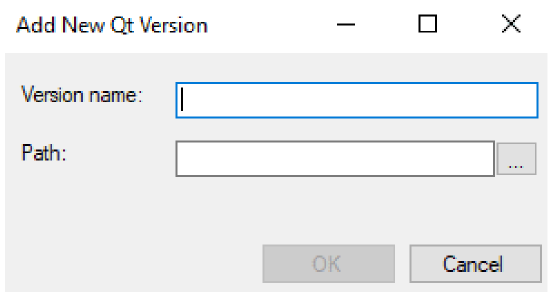


Figure 4: Add New QT Version

5.2 Building on Linux

The source code is provided for all the example code starting in version 3.0 of the installer. Additionally, *SLAHalLinux.cpp* is provided which can be used instead of *SLAHalPc.cpp* to build examples like slaCommand on a Linux machine. Makefiles or other build scripts are left up to the user.

5.3 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.

Appendix A - Managing Parameter File - ARM or PC Application Development

When developing an ARM or PC application to send SLA commands to the OEM hardware, commands sent by external applications can affect the state of the system when saving the parameter file. To alleviate these issues SightLine recommends the following guidelines when managing the parameter file:

- Disable ARM or PC applications when configuring and saving parameters to OEM hardware.
- Configure a single system with parameters and test the configuration.
- If the configuration test passes, use the SightLine upgrade utility application to retrieve the parameter file from the OEM hardware and save it to a separate location as a known good system configuration file.
- The upgrade utility can then be used to upload the known good system parameter file to OEM hardware.

See the EAN-Firmware Upgrade Utility for information on how to use the SightLine upgrade utility to manage the parameter file.