



SightLine
APPLICATIONS

EAN-Command Example Code

PN: EAN-Command-Example-Code

11/29/2018

**Contact:**

Web: sightlineapplications.com

Sales: sales@sightlineapplications.com

Support: support@sightlineapplications.com

Phone: +1 (541) 716-5137

Export Controls

Exports of SightLine products are governed by the US Department of Commerce, Export Administration Regulations (EAR); classification is ECCN 4A994. The [export summary sheet](#) located on the support/documentation page of our website outlines customers responsibilities and applicable rules. SightLine Applications takes export controls seriously and works to stay compliant with all export rules.

Copyright and Use Agreement

© Copyright 2018, SightLine Applications, Inc. All Rights reserved. The SightLine Applications name and logo and all related product and service names, design marks and slogans are the trademarks, and service marks of SightLine Applications, Inc.

Before loading, downloading, installing, upgrading or using any Licensed Product of SightLine Applications, Inc., users must read and agree to the license terms and conditions outlined in the [End User License Agreement](#).


All data, specifications, and information contained in this publication are based on information that we believe is reliable at the time of printing. SightLine Applications, Inc. reserves the right to make changes without prior notice.

Alerts

The following notifications are used throughout the document to help identify important safety and setup information to the user:

 **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.*



Contents

1	Overview	1
1.1	Associated Documents	1
1.2	Hardware Requirements	1
1.3	SightLine Software Requirements	1
1.4	Third Party Tools	1
2	Getting Started.....	1
2.1	Projects Overview	2
2.2	Sample Video Decode Application	3
2.3	Enabling RTSP Client Support in SLAPanelMinus	4
3	Generating, Unpacking and Parsing SightLine Command Packets	4
3.1	Communication Protocol Packet Generation and Unpacking	4
3.1.1	Packet Generation	4
3.1.2	Packet Unpacking.....	5
3.2	Packet Parsing	6
3.2.1	Packet Read.....	6
3.2.2	Packet Parse.....	6
4	Troubleshooting.....	7
4.1	Questions and Additional Support.....	7

List of Figures

Figure 1:	QT Options Dialog	7
Figure 2:	Add New QT Version	7



1 Overview

The example solution file in Visual Studio is designed to help software engineers develop code to interact with the SLA libraries running on SightLine hardware (1500-OEM, 3000-OEM). The solution file is in the latest SLA-PC Examples file available on SightLine [Software Download](#) page.

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

1.1 Associated Documents

[EAN-Startup Guide 1500-OEM](#): Describes steps for connecting, configuring, and testing the 1500-OEM video processing board on the 1500-AB accessory board.

[EAN-Startup Guide 3000-OEM](#): Describes steps for connecting, configuring, and testing the 3000-OEM video processing board on the 3000-IO interface board.

[Interface Command and Control \(IDD\)](#): Describes the native communications protocol used by the SightLine Applications product line. The IDD is also available as a local download on the [Software Download](#) page.

Panel Plus User Guide: Provides descriptions of all the settings in the Panel Plus application. (Located in the Panel Plus application in the *Help* menu.)

1.2 Hardware Requirements

- SightLine hardware with enabled functionality (i.e., Developer's kit with all connection cables.)
- A video source and video monitor.

1.3 SightLine Software Requirements

ⓘ IMPORTANT: The Panel Plus software version should match the firmware version running on the board.

1.4 Third Party Tools

- Microsoft Visual Studio latest version (Visual Studio 2013 is recommended)
- Multibyte MFC Library for Visual Studio [VS2013 MBCS](#)
- [QT Visual Studio Tools](#) (download includes QT 5.6.0). Used for SLAPanelMinus application

2 Getting Started

1. Unzip and install the SLA-PC Examples file.
2. Go to *C:\SightLine Applications\SLAExamples X.XX.XX\CmdCtrlExamples*. (*X.XX.XX will change with each release version*.)
3. Open the *slexample.sln* example solution file in Microsoft Visual Studio. The project implements a subset of the entire SightLine functional protocol.

For a complete overview of the SLA library interface see the [Interface Command and Control \(IDD\)](#).



2.1 Projects Overview


Projects		
bin	Contains windows 32-bit DLLs for FFmpeg (avcodec, avdevice ... dlls)	IMPORTANT: Add the following directory to the system path for all projects: C:\SightLine Applications\SLAExamples X.XX.XX\bin (where X.XX.XX is the version installed)
lib	Contains windows 32-bit libraries for FFmpeg	
include	FFmpeg and SLA header files	
SLADecode	Defines the SLADecode class interface for decoding mjpeg, mpeg2-ts video and KLV metadata.	See the Sample Video Decode Application section.
	SLATestGDI	
	Sample application - decodes mpeg2-ts or rtp-mjpeg video from a network stream and displays in a GDI window. <i>Software release 2.24.7 supports sample RTSP client.</i>	
CmdCtrlExamples	SLADecodePng	
	Sample code utilizing libpng to decode PNG files and extract encoded metadata	
	simpleDiscover	
	This project uses an Ethernet connection to discover any number of SLA boards over an Ethernet network. This is a Windows console project.	
	slaCommand	
	This project opens either a Serial or Ethernet connection to an SLA board (See the code for instructions on how to modify connection type). After the connection is established, communication to the board demonstrates how to request and set parameters, and how to initiate and receive track telemetry. This is a Windows console project.	It is important to use pre-processor definitions and define the IP and COM parameters accordingly if using a serial or Ethernet connection.
	SLAPanelMinus	
	This project is an example of how to include the SLADecoder into a Windows Qt application. Installation of the Qt resources are required (see section 1.4 above). See the Qt webpage downloads section for more information. See also Enabling RTSP Client Support in SLAPanelMinus	QT5.6 must be installed so that the directory C:\Qt\Qt5.6.0\5.6\msvc2013 exists After installing the Qt Visual Studio plug V 1.2.5, restart Visual Studio, and go to the new menu <i>QT5</i> and select <i>QT Options</i> Add in a new Qt Version named 5.6 and value C:\Qt\Qt5.6.0\5.6\msvc2013 It may also be necessary to add a windows environment variable <i>QTDIR</i> C:\Qt\Qt5.6.0\5.6\msvc2013 It may also be necessary to add C:\Qt\Qt5.6.0\5.6\msvc2013\bin to the system path.
	SLAPanelMinusInstaller	
	Example Installer project for installing the PanelMinus application on an PC without Visual Studio or Qt installed. Another option is to install SLAPanelPlus on a PC, and then copy the .dlls from the SLAPanelPlus install location to the PanelMinus application location. Before building the installer add all required .dlls from the c:\SightLine Applications\SLAExamples X.XX.XX\bin folder to the installer.	
	SLAVideoGrid	
	This project is a Windows, and QT application that display video from up to 6 sources in a 2 row by 3 column grid. See the readme.txt file in the SLAVideoGrid project directory for details.	

IMPORTANT: The *IP_ADDRESS* must be changed to the PC's address in SLACommManager.h



2.2 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 *Compression* tab:
 - a. Set *MPEG2-S* to *H.264*
 - b. Click *Use My IP - Unicast*.
 - c. Click *Apply*.
4. Close the Panel Plus application.
5. Browse to: *C:\SightLine Applications\SLAExamples X.XX.XX\SLADecode*.
6. Open *SLATestGDIDbg2013.sln* with Visual Studio.
7. Build the solution 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*

Tips:

If video is not displayed:

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



2.3 Enabling RTSP Client Support in SLAPanelMinus

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 *SLARtspOpenURL* (the client will try to use it). Another option is to use *clientPort=* in the URL. The port does not need to be configured externally, since the decoder is aware that the port has been created.

3 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.

3.1 Communication Protocol Packet Generation and Unpacking

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

3.1.1 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);
```



3.1.2 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);
}
```




3.2 Packet Parsing

Packet read, and parse are supported.

3.2.1 Packet Read

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

```
s32 FIPReadPacket(u8 *data, SLPort *port, s32 timeoutMs, bool
fipEx, u8 *extraHeaderByte)
void SLAReceiveThread::receive()
{
    // just read until we're 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 )
        {
            ...
        }
    }
}
```

3.2.2 Packet Parse

The `SLParsePackets()` example is included in `simplestfipport.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)
```



4 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 system's 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.

Verify QT Options in Visual Studio:

1. In Visual Studio, go to **QT VS TOOLS » Qt Options**.
2. Verify that the entry matches the entry shown in [Figure 1](#).
3. If the entry does not match, delete the entry and click **Add**. Browse to the path where the new version of QT is installed and click **OK** ([Figure 2](#)).

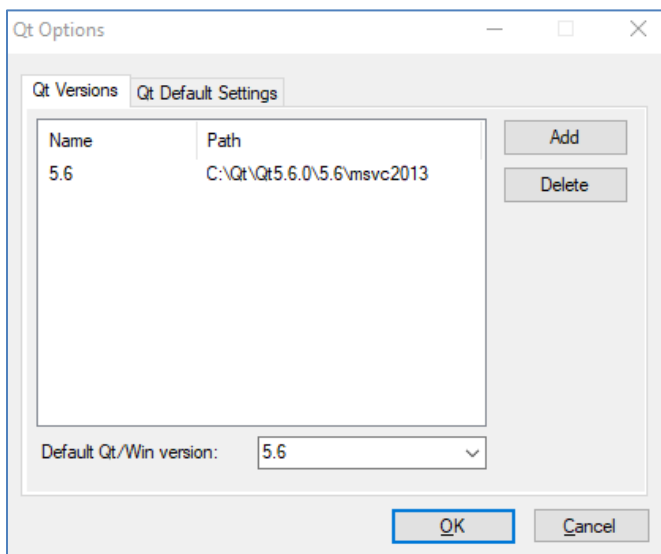


Figure 1: QT Options Dialog

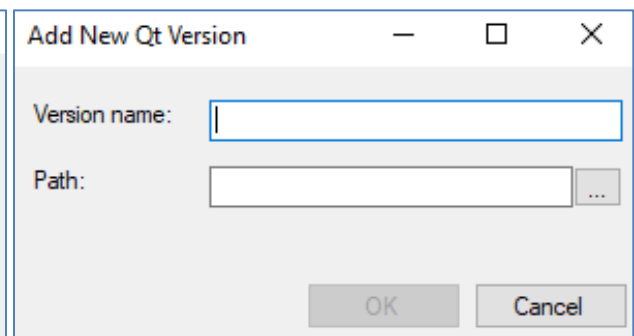


Figure 2: Add New QT Version

4.1 Questions and Additional Support

For questions and additional support, please contact [Technical Support](#). Additional support documentation and Engineering Application Notes (EANs) can be found on the Support pages of the SightLine Applications [website](#).