FARO LS SDK

VERSION 6.0

APRIL 2016

FARO Technologies Inc. Internal Control File Location:

X:\Control\Records\05manufa\PARTSPEC\7 Software\ E1566_FARO_LS_SDK_6.0_Manual_EN.pdf

Table of contents

1. Introduction

This manual describes the **FARO LS** assembly, which provides the **FARO Open** interface to read scan points, and the **FARO LS SDK** interface to control the scanner**.**

FARO Open provides access to the scan data by its interface iQLibIf. A FARO Laser Scan (.fls) can stand for itself, but normally SCENE is used to make it part of encompassing FARO Workspaces (.fws).

To access a scan in a workspace, first `load` the workspace; and then load or unload the single scans of this workspace by simply using the functions `loadScan` or `unloadScan`, and supplying the appropriate scan number. All the scans in the workspace are numbered from 0 to numScans-1, where numScans is the number of scans in the workspace.

To access an individual scan, `load` the scan without loading the workspace. In this case, you do not need to use `loadScan`, the scan will be accessible directly. Its index number is 0.

You can access the points in a scan in random order. The current implementation requires that the whole scan needs to be present in the memory.

See chapter 6 for more information on the FARO Open interface.

FARO LS SDK provides a compact and easy to use interface to control the scanner with its main functionality. All the basic operations like starting a scan, changing scan resolution or scan area can be controlled by this interface.

With FARO Photon scanners, the scan files will be stored on the local hard drive.

With FARO Focus^{3D} scanners, the scan files will be stored on the removable SD Card. They can be retrieved by ordinary file copy operations.

The FARO LS SDK and FARO Open examples in chapter 8.2 show how to use these interfaces within your application.

Developing applications with the FARO LS SDK and FARO Open interfaces is restricted by licenses. You need to obtain a license for your company from FARO. Then you can use and install as many instances of the interfaces as you like.

The interfaces might contain several classes and methods that are not described in this manual. Classes and methods not described in this manual are for experimental use only. Their proper functioning will not be guaranteed, they will not be supported and they might not exist in future versions of FARO LS anymore. We therefore recommend that you solely use the official classes and methods as described in this manual.

2. Prerequisites

FARO LS SDK and FARO Open is a compilation of COM interfaces and can be used in different programming languages, such as C++, C# or Visual Basic.

In general, COM interfaces can be used as registered COM interfaces, or with Side-by-Side. We recommend using the FARO LS not as a registered COM interface, but with Side-by-Side. See chapter 8.2 for more details about using the FARO LS assembly with Side-by-Side.

3. Compatibility

3.1 FARO Open

For compatibility reasons with older versions, the name iQLibIf and the signature of the functions of the interface have not changed.

Please note that the new license-protected FARO Open is not binary compatible with the old FARO Open which didn't have the license protection.

The interface can handle FARO workspace (.fws) FARO scan (.fls) and FARO scan project (.lsproj) files.

The interface supports scans with color, but it is currently only possible to load either the color information, or the original grey information. It is not possible to load and access both at the same time. With the default settings, the functions will load the scans with color information if it is available. You have to set the attribute before loading the scan. See chapter 6.1 for more information on this.

Please note:

Freestyle^{3D} scans are supported by the 64 bit SDK, but not with 32 bit SDK version.

The Focus ^{3D} X30 scanner does not support scanning with color information.

3.2 FARO LS SDK

This version of FARO LS SDK works with Focus^{3D} and Focus ^{3D} X scanners that have the scanner firmware version 5.0 and higher installed. Please make sure you have one of these version installed.

4. Installation

4.1 Installation for your development environment

As a developer, you have received a special developer setup which contains the recommended Side-by-Side[1] setup and in addition some useful tools.

When you run the developer setup, first the Side-by-Side setup and the additional tools will be installed to a folder. Then the Side-by-Side setup is run. Depending on your development environment, you may also want to register the DLLs manually, as this can be useful when developing C++ and C# applications. For more detailed information see chapter 8.2.

4.2 Installation with your application

For installing your application with FARO LS SDK and FARO Open only the Side-by-Side setup must be run and all needed files will be installed to the shared Side-By-Side folder. The Side-by-Side setup is then no longer needed and can be deleted. With Side-By-Side no registration is required, but your application must support Side-By-Side linking (see chapter 8.2).

[1] http://en.wikipedia.org/wiki/Side-by-side_assembly

5. Runtime Licensing

The FARO LS SDK and FARO Open interfaces are restricted by licenses and it is required to get a license key from FARO before using them. Then the class object must be created and before any other usage the license property must be set.

When creating the COM objects specify the CLSID of the object to create and the IID_IiQLicensedComponent as the interface to retrieve. Next, set the full license text to the license property. Once this is done, you may query for any interface on the object. Also, the license is propagated from object to object. Generally, it suffices to set the license once and use any objects/interfaces you require.

Visual C#.NET:

```
IiQLicensedInterfaceIf obj = new iQLibIfClass();
obj.License = /* FARO LS license code */
  "FARO Open Runtime License\n" +
  "Key:??????????????????????\n" +      //← replace ? with key!
  "\n" +
  "The software is the registered property of " +
  "FARO Scanner Production GmbH, Stuttgart, Germany.\n" +
  "All rights reserved.\n" +
  "This software may only be used with written permission " +
  "of FARO Scanner Production GmbH, Stuttgart, Germany.";
IiQLibIf libif = (IiQLibIf)obj
```

Please replace the question marks with your license key. It is important to keep the rest of the text.

Please refer to chapter 8.2 for examples of use.

6. Interface FARO Open

This chapter describes the functionality of the FARO Open interface iQLibIf.

6.1 General functions

`HRESULT` **`load`** `([in] BSTR fileName, [out, retval] int* result)`

Load a workspace (.fws) or a scan (.fls). If a scan is loaded, its load state will be `TRUE` and it will be number 0.

Return Values:

0, 11, 12, 13, 25

`HRESULT` **`save`** `([out, retval] int* result)`

Save a workspace (.fws) with the original name

Return Values:

0, 11, 14

`HRESULT` **`saveAs`** `([in] BSTR fileName, [out, retval] int* result)`

Save a workspace (.fws) with a new name

Return Values:

0, 11, 14

`HRESULT` **`getAttribute`** `([in] BSTR id, [out] BSTR* content,`
`[out, retval] int* result)`

Returns content of specified attribute

Return Values:

0, 21

`HRESULT` **`setAttribute`** `([in] BSTR id, [in] BSTR content,`
`[out, retval] int* result)`

Set specified attribute to defined content

Return Values:

0, 21

Example: Load grey information instead of color

```
setAttribute("#app/ScanLoadColor", "0")
```

`Property` **`scanReflectionMode`**

Determines the way reflection or color values are returned.

0. For grey scan points: Original reflection value of the scanner. In the case of a FARO Laser Scanner, this is in the range of 0 to 2047.
   For color: the lightness of the color in the range 0 to 4095.

1. For grey scan points: Mapped reflection value is in the range 0 (black) to 255 (white).
   For color: the lightness of the color in the range 0 to 255.
2. For grey scan points: Mapped reflection value is in the range 0 (black) to 255 (white) as RGB grey value.
   For color: RGB color, 8 bits per channel (bits 0-7: red, bits 8-15: green, bits 16-23: blue).

   The default reflection mode is 1.

`HRESULT` **`getNumScans`** `([out, retval] int* numScans)`

Get the number of scans in the current workspace. Scans are numbered starting with 0.

Return Values:

Number of scans, -1 in case of error

`HRESULT` **`getScanLoadState`** `([in] int scan,`

`[out, retval] BOOL* loadState)`

Get the load state of the given scan in the current workspace. The scan number can be in the range 0 to numScans-1.

Return Values:

True

False (also in case of error)

`HRESULT` **`loadScan`** `([in] int scan, [out, retval] int* result)`

Load the given scan in the current workspace. The scan number can be in the range 0 to numScans-1.

Return Values:

0, 11, 13, 16, 20, 25

`HRESULT` **`unloadScan`** `([in] int scan, [out, retval] int* result)`

Unload the given scan in the current workspace. The scan number can be in the range 0 to numScans-1.

Return Values:

0, 11, 12

`HRESULT` **`getScanNumRows`** `([in] int scan,`

`[out, retval] int* numRows)`

Get the number of rows in the given scan. The scan number can be in the range 0 to numScans-1.

Return Values:

Number of rows in this scan, -1 in case of error

`HRESULT` **`getScanNumCols`** `([in] int scan,`

`[out, retval] int* numCols)`

Get the number of columns in the given scan. The scan number can be in the range 0 to numScans-1.

Return Values:

Number of columns in this scan, -1 in case of error

```
HRESULT getScanPoint ([in] int scan, [in] int row,
            [in] int col, [out] double* x, [out] double* y,
            [out] double* z, [out] int* color,
            [out, retval] int* result)
```

Get the point at the given row/column in the given scan. The result is in the local cartesian coordinate system of the scanner. The returned color value depends on the selected reflection mode:

0. For grey scan points: Original reflection value of the scanner. In the case of an FARO Laser Scanner this is in the range of 0 to 2047.
   For color: the lightness of the color in the range 0 to 255.
1. For grey scan points: Mapped reflection value is in the range 0 (black) to 255 (white).
   For color: the lightness of the color in the range 0 to 255.
2. For grey scan points: Mapped reflection value is in the range 0 (black) to 255 (white) as RGB grey value.
   For color: RGB color, 8 bits per channel (bits 0-7: red, bits 8-15: green, bits 16-23: blue).

The default reflection mode is 1.

The scan number can be in the range 0 to numScans-1.

See chapter 8.4 for information about the calculation of the global position of scan points.

Attention! Due to COM overhead, this method is not the best way to read the complete scan. Use `getXYZScanPoints` or `getXYZScanPoints2` instead to read multiple scan points in a single call. For example, you can read the scan points column by column with these two methods.

Return Values:

0, 26

```
HRESULT getScanPosition ([in] int scan,
            [out] double* x, [out] double* y, [out] double* z,
            [out, retval] int* result)
```

Get the position of the given scan. The result is in the global cartesian coordinate system of the model. The scan number can be in the range 0 to numScans-1. See chapter 8.4 for information about the usage of the scan position for calculating the global position of scan points.

Return Values:

0, 26

```
HRESULT getScanOrientation ([in] int scan,
            [out] double* x, [out] double* y, [out] double* z,
            [out] double* angle, [out, retval] int* result)
```

Get the orientation of the given scan in axis-angle notation. The result is in the global cartesian coordinate system of the model. The scan number can be in the

range 0 to numScans-1. See chapter 8.4 for information about the usage of the scan orientation for calculating the global position of scan points.

Return Values:

0, 26

```
HRESULT getXYZScanPoints ([in] int scan,
        [in] int row, [in] int col, [in] int numRows,
        [out] double* pos, [out] int* color,
        [out, retval] int* result)
```

Get multiple points starting at the given row/column in the given scan. All points in the given column from row row up to row+numRows-1 are fetched. Each point stores 3 values (x, y, z) into the array pos and one value into the array refl. The result is in the local cartesian coordinate system of the scanner. The returned color value depends on the selected reflection mode:

0. For grey scan points: Original reflection value of the scanner. In the case of an FARO Laser Scanner this is in the range of 0 to 2047.
   For color: the lightness of the color in the range 0 to 4095.
1. For grey scan points: Mapped reflection value is in the range 0 (black) to 255 (white).
   For color: the lightness of the color in the range 0 to 255.
2. For grey scan points: Mapped reflection value is in the range 0 (black) to 255 (white) as RGB grey value.
   For color: RGB color, 8 bits per channel (bits 0-7: red, bits 8-15: green, bits 16-23: blue).

The default reflection mode is 1.

The scan number can be in the range 0 to numScans-1.

See chapter 8.4 for information about the calculation of the global position of scan points.

Attention! The arrays are not managed! Therefore this function can only be used in C++! `getXYZScanPoints2` offers a managed access to the scan points.

Return Values:

0, 26, 92

```
HRESULT getXYZScanPoints2 ([in] int scan,
        [in] int row, [in] int col, [in] int numRows,
        [out] SAFEARRAY(double)* pos,
        [out] SAFEARRAY(int)* color,
        [out, retval] int* result)
```

Same as **`getXYZScanPoints`**, but managed.

```
HRESULT getPolarScanPoints ([in] int scan,
        [in] int row, [in] int col, [in] int numRows,
        [out] double* pos, [out] int* color,
        [out, retval] int* result)
```

Get multiple points starting at the given row/column in the given scan. All points in the given column from `row` up to `row+numRows`-1 are fetched. Each point stores 3 values (r, ω, θ) into the array `pos` and one value into the array `refl`. The result is in the local polar coordinate system of the scanner. The returned color value depends on the selected reflection mode:

0. For grey scan points: Original reflection value of the scanner. In the case of an FARO Laser Scanner this is in the range of 0 to 2047.
   For color: the lightness of the color in the range 0 to 255.
1. For grey scan points: Mapped reflection value is in the range 0 (black) to 255 (white).
   For color: the lightness of the color in the range 0 to 255.
2. For grey scan points: Mapped reflection value is in the range 0 (black) to 255 (white) as RGB grey value.
   For color: RGB color, 8 bits per channel (bits 0-7: red, bits 8-15: green, bits 16-23: blue).

The default reflection mode is 1.

The scan number can be in the range 0 to numScans-1.

Attention! The arrays are not managed! Therefore this function can only be used in C++! `getPolarScanPoints2` offers a managed access to the scan points.

Return Values:

0, 26, 92

```
HRESULT getPolarScanPoints2 ([in] int scan,
        [in] int row, [in] int col, [in] int numRows,
        [out] SAFEARRAY(double)* pos,
        [out] SAFEARRAY(int)* color,
        [out, retval] int* result)
```

Same as getPolarScanPoints, but managed.

```
HRESULT getScanAnglesIf ([in] int scan,
        [out] FAROScanAnglesIf* scanAnglesIf,
        [out, retval] int* result)
```

Get an interface to the internal angle information of the scan (see chapter 6.2). The scan number can be in the range 0 to numScans-1.

Return Values:

0, 26

6.2 Functions for Automation Related Data

The functions described in this chapter are only for FARO Focus^{3D} scans that have been enhanced with automation data using the automation interface of the Focus^{3D}. For more information on the automation interface, please refer to the *Automation Interface manual*.

To get access to the integrated automation data stored in the scan data stream, the FARO Open interface offers access routines. With these access routines, every automation message and every automation trigger signal can be read out. The automation data and scan data access takes place after the scan operation has been finished.

The automation data is maintained in two lists, one for automation messages sent via CAN bus and one for trigger signals issued through the I/O interface. Each list is sorted chronologically.

To get a correspondence between automation data and scan data, each scan point and each automation message or trigger signal has an individual time stamp. For the time synchronization with a third party device, the absolute automation time can be set by the automation application with a CAN message.

In most cases you will use the SDK in the following ways:

1. Having an event (trigger signal or automation message), you want to identify the scan point that corresponds to this event. In this case, you first have to get the automation time of the event and then identify the row and the column of the scan point using the retrieved automation time.
2. Having a scan point at a known row and column, you want to identify its automation time in order to find correspondence to your external data. In this case you can retrieve automation time for a given scan point by the given row and column.

6.2.1. Access to automation messages

```
HRESULT getNumAutomationMessages ([in] int scan,
                [out,retval] int* numberOfMessages);
```

Get number of automation messages in scan file.

Return Values:

numberOfMessages: Number of messages or -1 on error

```
HRESULT getAutomationMessage ([in] int scan,
                [in] int messageNr,
                [out] unsigned long long * atime,
                [out] unsigned int * canId,
                [out] unsigned int * dataHigh,
                [out] unsigned int * dataLow,
                [out,retval] int* result);
```

Get automation message in scan file.

messageNr: Number of message (should be less than numberOfMessages provided by previous method)

atime: Automation time

canId: ID of CAN message

dataHigh: Data bytes D7 to D4 of CAN data

dataLow: Data bytes D3 to D0 of CAN data

Return Values:

`result`: 1 on success, 0 on error

```
HRESULT getPreviousAutomationMessage ([in] int scan,
                [in] unsigned long long atime,
                [out,retval] int* messageNr);
```

Get automation message previous to the provided automation time. Assuming that there is an automation message exactly at the provided automation time the message before will be returned.

atime: Automation time

Return Values:

`messageNr`: Number of message or -1 if not existent

```
HRESULT getNextAutomationMessage ([in] int scan,
              [in] unsigned long long atime,
              [out,retval] int* messageNr);
```

Get the automation message which comes after provided automation time. Assuming that there is an automation message exactly at the provided automation time the message after will be returned.

atime: Automation time

Return Values:

`messageNr`: Number of message or -1 if not existent

6.2.2. Access to automation trigger signals

```
HRESULT getNumAutomationTriggers ([in] int scan,
              [out,retval] int* numberOfTriggers);
```

Get number of automation trigger signals in scan file.

Return Values:

`numberOfTriggers`: Number of trigger signals or -1 on error

```
HRESULT getAutomationTriggerTime ([in] int scan,
              [in] int triggerNr,
              [out] unsigned long long * atime,
              [out,retval] int* result);
```

Get automation time of trigger signal in scan file.

triggerNr: Number of the trigger signal

atime: Automation time

Return Values:

`result`: 1 on success, 0 on error

```
HRESULT getPreviousAutomationTrigger ([in] int scan,
              [in] unsigned long long atime,
              [out,retval] int* triggerNr);
```

Get the automation trigger signal previous to the provided automation time. Assuming that there is a trigger signal exactly at the provided automation time the trigger signal before will be returned.

atime: Automation time

Return Values:

`triggerNr`: Number of trigger signal or -1 if not existent

```
HRESULT getNextAutomationTrigger ([in] int scan,
              [in] unsigned long long atime,
              [out,retval] int* triggerNr);
```

Get the automation trigger signal which comes right after the provided automation time. Assuming that there is a trigger signal exactly at the provided automation time the trigger signal after will be returned.

atime: Automation time

Return Values:

`triggerNr`: Number of trigger signal or -1 if not existent

6.2.3. Access to automation time of scan points and reverse lookup

```
HRESULT getNearestScanPointFromAutomationTime ([in] int scan,
              [in] unsigned long long atime,
              [out] int * row, [out] int * col,
              [out,retval] int* result);
```

Get the nearest scan point before or after the provided automation time.

atime: Automation time

row: Row of scan point

col: Column of scan point

Return Values:

`result`: 1 on success, 0 on error

```
HRESULT getAutomationTimeOfScanPoint ([in] int scan,
              [in] int row, [in] int col,
              [out] unsigned long long * atime,
              [out,retval] int* result);
```

Get approximated automation time of a scan point

row: Row of scan point

col: Column of scan point

atime: Automation time

Return Values:

`result`: 1 on success, 0 on error

```
HRESULT getAutomationTimeOfSyncPulse ([in] int scan,
              [in] int col,
              [out] unsigned long long * atime,
              [out,retval] int* result);
```

Get the automation time of the sync pulse for the given column. As one rotation consists of two columns, there will be two columns with the same automation time for the sync pulse. See the *Automation Interface manual* for more information.

col: Column of scan point

atime: Automation time

Return Values:

`result`: 1 on success, 0 on error

6.3 Interface FAROScanAnglesIf

This interface provides access to some internal angle information. You will most probably never need it, as all this information is of course already used when retrieving the scan points with the iQLibIf.

```
HRESULT getHorizontalAngle ([in] int row, [in] int col,
            [out] double* phi,
            [out, retval] VARIANT_BOOL* retVal)
```

Returns the uncompensated horizontal angle of a scan point at a given row and column. Uncompensated means that module misalignments are not corrected.

Return Values:

0

```
HRESULT getVerticalAngles [in] int col,
            [out] double* start, [out] double* delta,
            [out, retval] VARIANT_BOOL* retVal)
```

Returns the vertical start angle of a column and the angular delta within the column. The start angle is given for the top row. No compensations of module misalignments are applied.

Return Values:

0

```
Property ColEndFrontSight
```

Contains the last column which was scanned in front sight. The following columns have been scanned in back sight. The value -1 indicates that no column was scanned in front sight (Read only).

7. Interface FARO LS SDK

This chapter describes the functionality of the FARO LS SDK interface IScanCtrlSDK.

7.1 Connecting to the scanner

`Property BSTR` **`ScannerIP`** `(Read/Write)`

The IP address of the scanner.

`HRESULT` **`connect`** `([out, retval] int* result)`

Tries to connect to the scanner with the given **ScannerIP**. Changing **ScannerIP** doesn't automatically start the connection, so always first set **ScannerIP**, then call connect.

Return Values are (see chapter 8.1):

0, 2, 4

`Property bool` **`Connected`** `(Read only)`

It returns, whether FARO LS SDK is connected to a scanner or not.

7.2 Scanner parameters

Two different laser scanner operation modes are supported, the **spherical** and the **helical** scan mode.

7.2.1. Spherical scan mode

The **spherical scan mode** is the default scan mode and corresponds to normal 3D scan activity, the laser mirror and the laser scanner itself are rotating to acquire scan data.

7.2.2. Helical scan mode

In the **helical scan mode**, an optional operation mode, the scanner operates in 2D profiling-mode, where the rotating mirror axis is used, while the horizontal rotation axis stays locked. The scanner captures a continuous stream of scan point data and is typically moved along a track or road. By synchronizing the captured scan point data with positioning information a longitudinal profile is generated in form of a screw line (Helix).

The helical mode has two sub modes, the helical TTL and the helical CAN mode. These two sub modes differ from one another in the control and communication signals.

The helical sub modes have to be enabled first before recording a helical scan. Enabling one of the two helical sub modes as well as setting the scanning parameters can be done via the scanner's user interface or with the FARO LS SDK.

When a helical mode is enabled, the horizontal rotation axis stays locked and the scanner will not turn horizontally. When switching the scanner off in helical mode, it will keep knowledge about this mode and the horizontal rotation axis is still locked when the scanner is switched on again.

In both helical modes, capturing of colored scans is not supported as well as setting the horizontal scan angles. In addition to the standard scanning parameters resolution and quality, the number of the scan lines to be recorded can be set. The scanner will stop scanning when this number has been reached. It is also possible to select that the scan

data stream is automatically split into several files when a certain amount of scan lines has been reached.

When a helical mode has been enabled and the parameters have been set, the helical scan operation has to be initiated before starting a helical scan. This can be done

- via the scanner's user interface by pressing the **Start** button,
- by pressing the **Start/Stop** button on the scanner,
- by executing the **Start Scan** function of the FARO LS SDK or
- by sending an automation CAN message (in both, the TTL and the CAN mode).

Please refer to the *Automation Interface manual* for further details.

7.2.3. Scan Mode Properties

`Property ScanMode` **`ScanMode`** `(Read/Write)`

`ScanMode` can have four values:

- *`StationaryGrey`*
  Set *`StationaryGrey`* to take a normal spherical grey scale scan.
- *`StationaryColor`*
  Set *`StationaryColor`* to record spherical scans with color. This feature is only available with FARO Focus^{3D} X330 and X130 scanners. It will not work with FARO Photon, FARO LS scanners, or the Focus^{3D} X30.
- *`HelicalGrey`*
  Set *`HelicalGrey`* to take a scan in helical mode using the TTL interface.
- *`HelicalCANGrey`*
  Set *`HelicalCANGrey`* to take a scan in helical mode using CAN communication. **Please note** that this feature is only available for FARO Focus^{3D} scanners and will not work with FARO Photon or FARO LS scanners.

In the *`HelicalGrey`* and *`HelicalCANGrey`* modes, there are some differences in the behavior of the SDK compared to the *StationaryGrey/StationaryColor* modes:

- `NumCols` must be set by the user and will not automatically be adjusted when changing other parameters like `Resolution`.
- `HorizontalAngleMin` and `HorizontalAngleMax` will always be set to 0°. The old values of `HorizontalAngleMin` and `HorizontalAngleMax` will be lost and must be set again when returning to *StationaryGrey* or *StationaryColor* mode.
- At calling `startScan` the mirror will rotate, but the recording of the scan points will not start until a trigger signal was sent through the scanner's automation interface (see the *Automation Interface manual* for more information).

Attention: In C++, the property is named **`_ScanMode`** (note the underscore)

`Property int` **`SplitAfterLines`** `(Read/Write)`

Split the scan data stream into several files when the amount of lines or columns defined by this parameter has been reached. This is mostly relevant for scans recorded in helical mode.

`Property double` **`VerticalAngleMin`** `(Read/Write)`

Reads / Writes the vertical start angle for the scan area in degrees.

`Property double` **`VerticalAngleMax`** `(Read/Write)`

Reads / Writes the vertical end angle for the scan area in degrees.

`Property double` **`HorizontalAngleMin`** `(Read/Write)`

Reads / Writes the horizontal start angle for the scan area in degrees.

In **`ScanMode`** *HelicalGrey* and *HelicalCANGrey* this parameter will automatically be set to 0° and cannot be changed.

`Property double` **`HorizontalAngleMax`** `(Read/Write)`

Reads / Writes the horizontal end angle for the scan area in degrees.

In **`ScanMode`** *HelicalGrey* and *HelicalCANGrey* this parameter will automatically be set to 0° and cannot be changed.

Please note:

`HorizontalAngleMax` always has to be higher than **`HorizontalAngleMin`**, and that both values are within 0 and 360°. Otherwise, undefined behavior could arise.

`Property int` **`Resolution`** `(Read/Write)`

Reads / Writes the scan resolution.

Only the following values are permitted: 1, 2, 4, 5, 8, 10, 16, 20, 32

Example: Resolution 10 means that the number of scan columns and rows are 1/10 of the scan at resolution 1.

Resolution 16, 20 and 32 are not supported on some older FARO LS880 and LS 420 scanners. Invalid resolutions will be corrected and automatically set to the next higher resolution. E.g. if you set the resolution to 16, it will be corrected to 10.

Please note: This resolution is the recorded resolution; the resulting resolution of the scan might differ depending on the chosen **`NoiseCompression`**. For more information, see the descriptions below and chapter 7.3.

`Property int` **`MeasurementRate`** `(Read/Write)`

The measurement rate or speed used to record the scans.

Possible values: 1, 2, 4, 8.

1 = 122.000 points per second

2 = 244.000 points per second

4 = 488.000 points per second

8 = 976.000 points per second

`Property int` **`NoiseCompression`** `(Read/Write)`

Apply noise compression to the recorded scans. This reduces noise in the scans and thus the resulting resolution.

Not all combinations with **`MeasurementRate`** will be accepted by the scanner. See chapter 7.3 for the supported combinations.

Possible values: 1, 2, 4.

1 = no noise compression.

2 = reduces number of points by 4.

4 = reduces number of points by 16.

`Property int` **`NumCols`** `(Read/Write)`

Reads / Writes the number of columns of the scan.

In **`ScanMode`** *StationaryGrey* `NumCols` will automatically be adjusted every time, when `Resolution`, `HorizontalAngleMin`, or `HorizontalAngleMax` has changed. So this parameter should be changed at last.

In **`ScanMode`** *HelicalGrey* changes on `Resolution` or the scan area will have no effect on `NumCols`.

`Property int` **`NumRows`** `(Read Only)`

It returns the number of rows of the scan.

`Property int` **`ScanFileNumber`** `(Read/Write)`

Reads / Writes the scan file number of the last scan. It will automatically be increased before every new scan.

`Property BSTR` **`ScanBaseName`** `(Read/Write)`

The full scan file name will be "ScanBaseName" + ScanFileNumber + ".fls". The scan will be saved locally in the scans folder of the scanner hard drive.

`Property StorageMode` **`StorageMode`** `(Read/Write)`

This mode defines where to save the recorded scans.

Possible values:

`SMLocal`: Store scans on scanner (on the removable SD card of the Focus^{3D} or the internal hard disk of the Laser Scanner Photon).

`SMRemote`: Store scans on the remote computer (the computer the SDK is running on).

`SMAuto`: If connected to the scanner, the scans will be stored to the remote computer. If the remote computer is not connected, scans will be stored locally on the scanner.

`SMUndefined`: Storage mode is not defined yet (e.g. not yet connected to scanner).

`Property BSTR` **`RemoteScanStoragePath`** `(Read/Write)`

This defines the folder on the remote computer to save the scans to (relevant if **`StorageMode`** is set to *`SMRemote`*). E.g.

`HRESULT` **`syncParam`** `([out, retval] int *result)`

All scanner parameters are stored locally by the FARO LS SDK. By calling syncParam they get synchronized with the scanner

7.3 Combinations of Resolution, MeasurementRate and NoiseCompression

The following table shows all supported combinations of **`Resolution`**, **`MeasurementRate`** and **`NoiseCompression`** as well as the corresponding settings (resolution and quality) in the scanner's user interface:

SCANNER USER INTERFACE		SDK PARAMETERS		
Resolution	**Quality**	**Resolution**	**MeasurementRate**	**NoiseCompression**
1/1	1x	1	8	1
1/1	2x	1	4	1
1/1	3x	1	2	1
1/1	4x	1	1	1
1/2	1x	2	8	1
1/2	2x	2	4	1
1/2	3x	2	2	1
1/2	4x	2	1	1
1/2	6x	1	1	2
1/4	1x	4	8	1
1/4	2x	4	4	1
1/4	3x	4	2	1
1/4	4x	4	1	1
1/4	6x	2	1	2
1/4	8x	1	1	4
1/5	2x	5	4	1
1/5	3x	5	2	1
1/5	4x	5	1	1
1/8	2x	8	4	1
1/8	3x	8	2	1
1/8	4x	8	1	1
1/8	6x	4	1	2
1/8	8x	2	1	4
1/10	3x	10	2	1
1/10	4x	10	1	1
1/10	6x	5	1	2
1/10	8x	5	1	2
1/16	3x	16	2	1
1/16	4x	16	1	1
1/16	6x	8	1	2
1/16	8x	4	1	4
1/20	4x	20	1	1
1/20	6x	10	1	2
1/20	8x	5	1	4
1/32	4x	32	1	1

1/32	6x	16	1	2
1/32	8x	8	1	4

7.4 Scanning

`HRESULT` **`startScan`** `([out, retval] int* result)`

In the **spherical scan mode**, this function starts the scan.

In the **helical scan mode**, this initiates the scan. The mirror will start rotating but scan data will not be recorded until a trigger signal was sent through the scanner's helical TTL interface if the helical TTL mode is used.

In the **helical CAN mode**, the corresponding CAN message or **`recordScan`** (see chapter 7.4.1) will start scan data recording. In case of HelicalCANGrey, `recordScan` and `pauseScan` can be used.

For more information on the helical modes, please see the *Automation Interface manual*.

Be sure to be connected and have run **syncParam** to transfer the parameters to the scanner.

Return Values (see chapter 8.1):

0, 1, 3, 4

startScan is asynchronous, which means that even if it returns the value 0 (OK), the scanner might still not start scanning for some reasons (e.g. because the SD card is full). In this case you will get the related error code of the scanner by Error Handling (see chapter 7.5 for more information).

`HRESULT` **`stopScan`** `([out, retval] int* result)`

This stops a scan, if running.

Return Values ((see chapter 8.1):

0, 1, 3

`Property int` **`ScanProgress`**

It returns the current scan progress in percent.

`HRESULT` **`shutdown`** `([out, retval] int* result)`

This shuts down the scanner. Before disconnecting the scanner from the power supply always shut down the scanner (by calling this function or pressing button on scanner).

Return Values ((see chapter 8.1):

0, 3

7.4.1. Special Functions for the Helical CAN Mode

Please note: These functions only work with HelicalCANGrey, and not with HelicalGrey.

`HRESULT` **`recordScan`** `([out,retval] int* result)`

Starts scan data recording when scanning in helical CAN mode.

Return Values (see chapter 8.1):

0, 3

`HRESULT` **`pauseScan`** `([out, retval] int* result)`

Pauses scan data recording when scanning on helical CAN mode. Restart scan data recording with **`recordScan.`**

Return Values (see chapter 8.1):

0, 3

`HRESULT` **`inquireRecordingStatus`** `([out] HelicalRecordingStatus* status, [out,retval] int* result)`

Retrieves the recording status from a helical CAN scan. Possible values are:

`HRSUnknown` - the SDK does not know the status yet.

`HRSPaused` - the scan is not being recorded

`HRSRecording` - the scan is being recorded

Return Values (see chapter 8.1):

0, 2, 3, 4

7.5 Error Handling

In case the one of the interface functions of FARO.LS.SDK returns an error code, you may retrieve a more detailed exception code from the scanner, which will give you more information why the interface function failed.

`Property int` **`NumberExceptions`**

Returns the current number of exceptions.

`HRESULT` **`clearExceptions`** `([out, retval] int* result)`

Clears all exceptions on the scanner. This function is automatically called at the beginning of scanStart.

Return Values (see chapter 8.1):

0, 1, 3

7.6 Connection Point _IScanCtrlSDKEvents

`HRESULT` **`scanCompleted(void)`**

This function will be called, when a scan has been completed. This callback works only, if your program is running a Windows message loop (as in every Windows application). For console applications you must run the Windows message loop manually by polling the syncParam function. An example can be found in chapter 8.3.1.4.

7.7 Scanner Turning

The scanner can be turned to arbitrary horizontal angles without actually scanning. The turning will be handled asynchronously, so your application has to check in regular intervals whether the requested position has been reached by the scanner.

HRESULT **moveToHorizontalAngle** ([in] double angle,

[out, retval] int* result)

This function starts the turning, the angle is given in degrees.

Return Values (see chapter 8.1):

0, 1, 3, 4

HRESULT **requestScannerAngles** ([out, retval] int* result)

This sends a poll request to the scanner to retrieve the current angles. The poll will be handled asynchronously, as soon as the answer is back, the property ReceivedScannerAngles is set true.

Return Values (see chapter 8.1):

0, 3

Property bool **ReceivedScannerAngles**

It Returns the status of the last poll for scanner angles. It is true if the last poll has been answered.

Property double **HorizontalAngle**

It returns the horizontal angle as it was acquired by the most recent request that has been answered in the meantime. The value is in degrees.

Please note: If you are using the Focus^{3D} with FARO's Helical Adapter, you have to turn the scanner to 89° or -91° in order to lock the scanner with the adapter's fixation pins. Please see the *Automation Interface manual* of the Focus^{3D} for more information.

7.8 Remote Access to Scans (only Focus^{3D})

To access the scans on the Focus^{3D} (the scans stored on the inserted removable SD card), remote access must be enabled. After enabling remote access all common methods to access a network drive are allowed.

Property RemoteScanAccessStatus **RemoteScanAccess** (Read/Write)

RemoteScanAccess can have three different values:

RSAEnabled, *RSADisabled* or *RSAUnknown*

By getting this value you can determine the current state of the remote access. *RSAUnknown* will be returned if the SDK doesn't know the current state.

Setting this value to *RSAEnable* or *RSADisable* will enable or disable the remote access. Setting this to *RSAUnknown* will not change the current state.

Setting a state will wait until the scanner has the correct state.

Attention: In C++, the property is named **_RemoteScanAccess** (note the underscore).

8. Appendix

8.1 Error Numbers

0	Ok
1	Busy
2	Time out
3	Not connected
4	Failed
11	No workspace
13	No scan
14	Cannot open for reading
15	Cannot open for writing
16	Cannot find scan file
17	Cannot find scan data
21	Unknown scan version
22	Unknown key
26	Out of memory
27	Data missing
81	Scan still active
83	Scanner Operation Failure
94	Out of boundary
116	Scanner Busy
169	File not found

8.2 Using FARO LS with Side-By-Side

When using functionality from a COM object, there are two different version numbers which should be distinguished.

First, there is the version number of the interface. A different version number indicates that the interface changed in some way, for example in the number of functions it offers, or in the number of parameters. Your compiled application is very strongly depending on this version number, as in most cases it will no longer be able to use the COM object correctly if the interface has changed. At least a recompilation would be required, in some cases you even have to change your source code.

Second, there is the version number of the implementation. For example to indicate a bug fix in the implementation, it will have a different implementation version number, but the interface is still the same.

The version number of the interface and the version number of the implementation in combination describe the version of the FARO LS assembly.

In general your application requires a specific interface, but nevertheless it would like to use the implementation with the highest version number.

Although there is some support for this behavior in the registration mechanism of COM interfaces, the best support is given by Microsoft's *Side-By-Side*. It is recommended to use FARO LS only with *Side-By-Side*. The usage of *Side-By-Side* is a little bit tricky, so a brief tutorial will be provided in this chapter.

After installing FARO LS there will a folder created in "*C:\Windows\WinSxS* named "*x86_FARO.LS_1d23f5635ba800ab…*". This folder contains all DLLs needed for FARO LS. The *WinSxS* folder also contains a folder named *Manifest.* Here you'll find an assembly manifest file named "*x86_FARO.LS_1d23f5635ba800ab….manifest".* It is just a XML file containing some information, e. g. the version number, of the FARO LS assembly.

Now different versions of FARO LS are installed to different folders in *WinSxS*, so there is no problem to install several versions parallel. In many situations the new version has the same interface as the old version. It is fully compatible with the old version, it contains just some bug fixes or, in case of FARO Open, it just can open the scans of a new SCENE version. In this case an application should use the new FARO LS version. Therefore there is the "*Policy"* folder in *WinSxS* containing some redirections to take a newer version. This redirection works automatically and there is nothing you must do about it.

All you must do is writing an application manifest file for your executable. An application manifest is a XML file connecting your executable with the assembly manifest in the *WinSxS* folder. If the name of your executable is "myApplication.exe", the manifest name must be "myApplication.exe.manifest" and it must be in the same folder as the executable. It should look like this:

```
<?xml version="1.0" encoding="utf-8"?>
<assembly xmlns='urn:schemas-microsoft-com:asm.v1' manifestVersion='1.0'>
  <dependency>
    <dependentAssembly>
      <assemblyIdentity type='win32' name='FARO.LS' version='1.1.0.0'
      processorArchitecture='x86' publicKeyToken="1d23f5635ba800ab"/>
    </dependentAssembly>
  </dependency>
</assembly>
```

There is an entry *version='1.1.0.0'* which must match the version number of the FARO LS assembly manifest. The version number of the manifest is connected to the version number of the interface. You'll find this manifest with the correct version number in the file *Example.FARO.LS.exe.manifest* of your FARO LS folder.

8.2.1. C++ with Visual Studio

If you use C++ in Visual Studio 2005, then Visual Studio can write the manifest file for you. Open the properties dialog of the project and change in *Linker - Manifest File* the option *Generate Manifest* to yes, *Manifest File* should be correct by default. Then add the dependencies for FARO LS to *Additional Manifest Dependencies* (see image). Insert the version number of FARO LS assembly manifest. Then set *Allow Isolation* to yes.

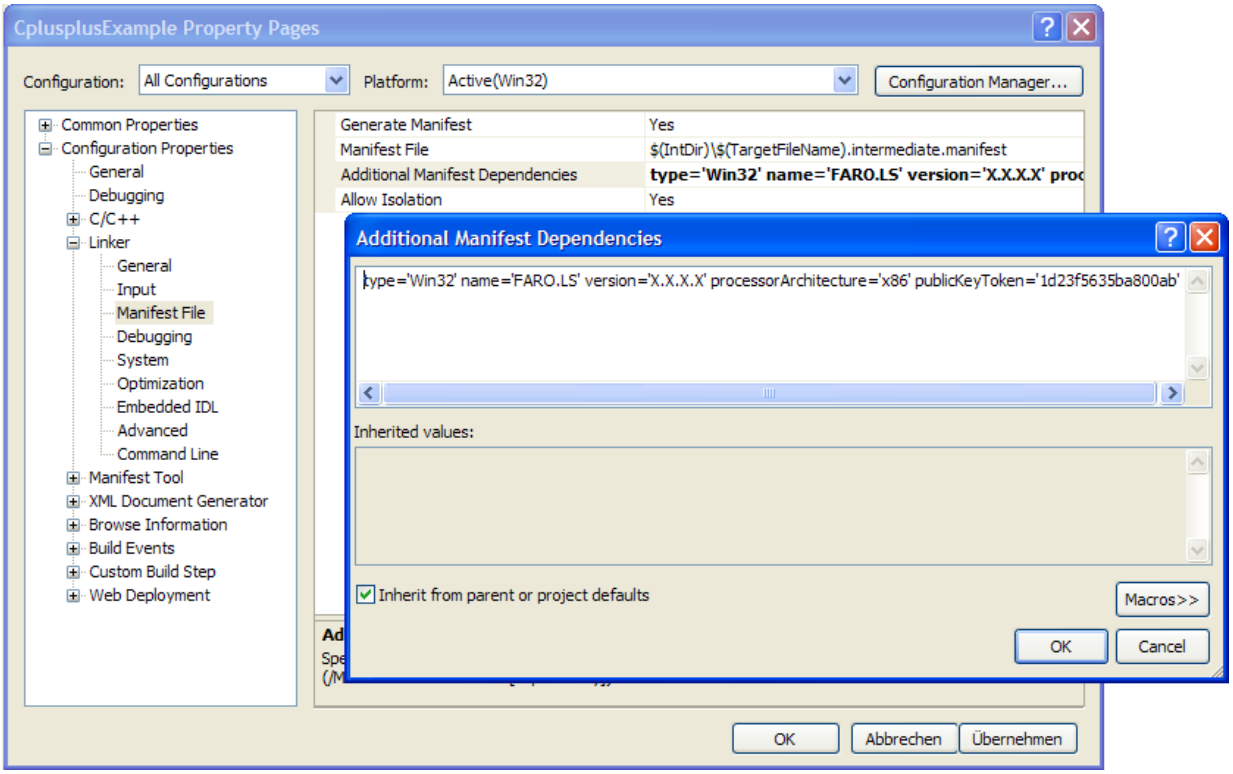

Figure 8-1: Manifest Dependencies

As an alternative to the *Additional Manifest Dependencies* option you can set the dependency by this pragma command directly into the source code:

```
#pragma comment(linker, "\"/manifestdependency:type='win32' name='FARO.LS'
version='1.1.0.0' processorArchitecture='x86'
publicKeyToken='1d23f5635ba800ab'\"")
```

(Please edit the version string - you'll find the correct version number in the file *Example.FARO.LS.exe.manifest* of your FARO LS folder)

The manifest file can be embedded to the resources of the executable, although it works as good with an external manifest file. But it's convenient to have all in the executable. The manifest gets embedded by the *mt.exe* tool with some parameters, but in Visual Studio this all can be done by a couple of clicks: Open the *Manifest Tool - Input and Output* property and set *Embed Manifest* to Yes (see image). The other properties should be correct by default. After setting all these properties you should clean the solution and rebuild, otherwise Visual Studio may not update the manifest file correctly. To be sure check the generated manifest file.

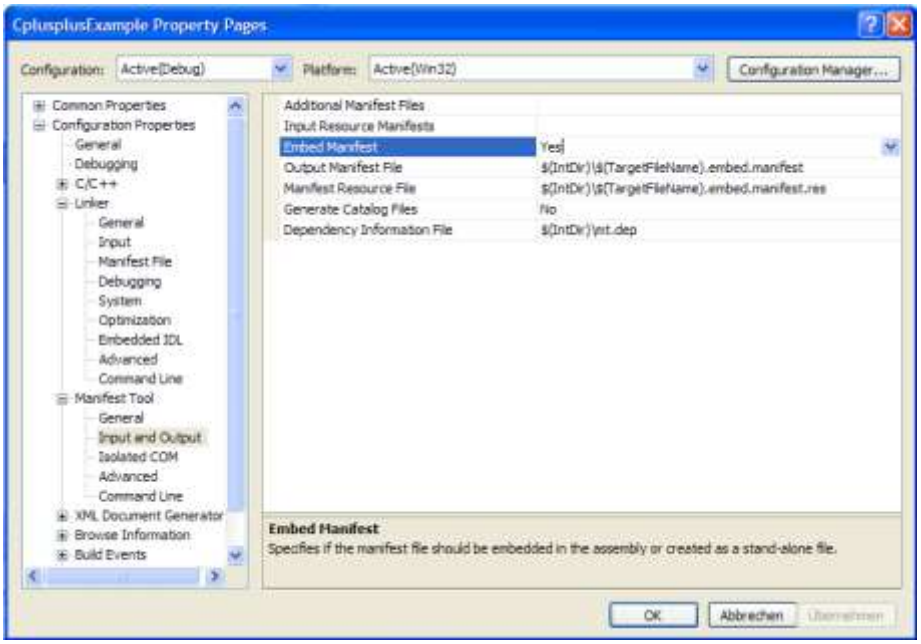

Figure 8-2: Properties Dialog of the C++ Project

Now your application should start without any registration.

8.2.2. C# with Visual Studio

The usage of Side-By-Side in C# is different to the use with C++: iQOpen.dll and FARO.LS.SDK.dll must be registered (see chapter 8.2.4) for the entire application development. Now you can add the COM references of iQOpen and FARO LS SDK to your project.

In Visual Studio 2012 you can change the Embed Interop Types in properties of reference. Make sure that it is set to false, otherwise your application won't run.

Then you must create a manifest file for the FARO LS dependency of your executable (simply use Example.FARO.LS.exe.manifest or merge it with additional dependencies). In Visual Studio 2008 add the manifest file to the project of your executable and open the properties of the project. In the **Application** tab select the manifest file to use and to embed to your executable.

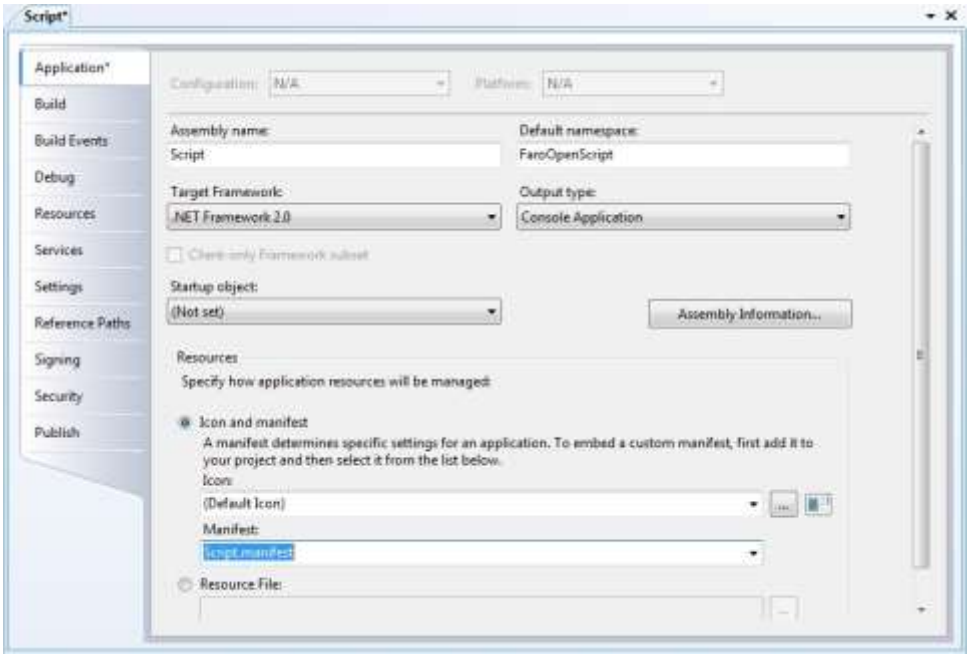

Figure 8-3: Properties dialog of the C# project

After compilation you can deregister the dlls and your application will still work.

Attention: for compiling the application it is necessary to have the COM interface registered, just for the execution the registration isn't needed any more.

8.2.3. Visual Basic with Excel, VBS and WSH scripts

Using Visual Basic scripts, e. g. in Excel, doesn't work with Side-By-Side. So you must register iQOpen.dll and FARO.LS.SDK.dll (see chapter 8.2.4) to use it with Visual Basic.

Generally, there is no way to add a manifest dependency to VBS or WSH scripts. As a workaround you can however register FARO.LS.SDK.dll and iQOpen.dll as described in section 8.2.4. Keep in mind that only the last recently registered versions will then be used by your scripts. We recommend deregistering older versions before registering a new version.

8.2.4. Registering FARO LS SDK and FARO Open

Although Side-By-Side should make registration redundant, in some cases, especially during development, the COM interfaces must be registered. Therefore open a command line window and change to the *WinSxS* folder of your *Windows* folder. You'll find a subfolder named "*x86_FARO.LS_1d23f5635ba800…"* for 32 bit versions or "*amd64_FARO.LS_1d23f5635ba800…" for 64 bit versions,* change to this subfolder and type:

```
regsvr32 .\iQOpen.dll
```

```
regsvr32 .\FARO.LS.SDK.dll
```

Make sure you get a positive feedback in a dialog window.

Now the FARO Open and FARO LS SDK interfaces are registered.

Please note that the registration with Windows Vista won't work, if the command line, you are using, doesn't have Administration rights.

8.2.5. Using FARO LS SDK and FARO Open within C++ Plug-in DLLs

The objects of FARO LS SDK and FARO Open are created using registration-free COM. In order for the COM layers to resolve object ids to the DLLs providing the implementation Manifest information is required. When using the FARO components from within an executable a manifest dependency as described in *C++ with Visual Studio* suffices. This approach however requires the FARO components being installed on the target machine. Because of the static dependency the application fails to load when the components are missing.

When developing a plug-in for an application the plug-in DLL should depend on the FARO components instead of the application executable. In this case some additional work has to be done. Under the hood, Windows processes manifests and turns the information into an activation context. Upon loading the plug-in DLL the activation context including the dependency on FARO LS is created and made active for the duration of the call to DllMain for initialization. This implies, when the FARO components are created from within DllMain of your plug-in DLL there is nothing special to do.

When creating objects from within a function exported by the DLL however, the proper activation context needs to be made active, so that the loader can correctly resolve COM object creations. To accomplish this we suggest storing the activation context in DllMain for later activation by the exported function. The example in section 8.3.2.3 illustrates this approach. Other alternatives are outlined in MSDN. We encourage the interested reader to lookup "*Isolated Applications and Side-by-side Assemblies*" in "*Win32 and COM Development/Administration and Management/Application Installation and Servicing*".

Attention: When testing your compiled plug-in for proper activation context management you must make sure that neither FARO.LS.SDK.dll nor iQOpen.dll is registered on

your system. Otherwise, you will not notice any load failures from incorrect activation contexts.

8.3 Examples

8.3.1. C#

For developing with FARO LS SDK and C# the libraries iQOpen.dll and FARO.LS.SDK.dll must be registered during the whole development time. After that, if you use Side-By-Side, you can run your application without registration. For more details see chapter 8.2.4. As Visual Studio is a 32 bit application, you must have the 32 bit SDK registered for compiling. For execution, the same SDK platform as your application is built for must be registered. In particular, if using Side by Side your manifest must have the same platform definition as your application.

8.3.1.1. Faro Open

```
using IQOPENLib;


...
// FARO LS Licensing
string licenseCode = ... /* FARO LS license Code */;
IiQLicensedInterfaceIf licLibIf = new iQLibIfClass();
licLibIf.License = licenseCode;
IiQLibIf libRef = (IiQLibIf)licLibIf;

libRef.load(@"C:\temp\demo.fls");
double x,y,z, angle;
libRef.getScanPosition(0, out x, out y, out z);
libRef.getScanOrientation(0, out x, out y, out z, out angle);
...
```

8.3.1.2. FARO LS SDK

```
using LSSDKLib;
using IQOPENLib;

...
// FARO LS Licensing
string licenseCode = ... /* FARO LS license Code */;
IiQLicensedInterfaceIf licSDKIf = new ScanCtrlSDKClass();
licSDKIf.License = licenseCode;
IScanCtrlSDK scanCtrl = (IScanCtrlSDK)licSDKIf;

scanCtrl.ScannerIP = "132.154.24.13";
scanCtrl.connect();
// wait for connection.
Thread.Sleep(5000);
if (!scanCtrl.Connected)
  return;

// set parameters.
scanCtrl.ScanMode = ScanMode.StationaryGrey;
scanCtrl.HorizontalAngleMin = 0;
scanCtrl.HorizontalAngleMax = 360;
scanCtrl.VerticalAngleMin = -65;
scanCtrl.VerticalAngleMax = 90;
scanCtrl.Resolution = 10;
scanCtrl.ScanBaseName = "myScanFile";
scanCtrl.ScanFileNumber = 0;
scanCtrl.syncParam();

// wait a little bit for synchronizing scanner parameters.
Thread.Sleep(2000);
scanCtrl.startScan();
...
```

8.3.1.3. Garbage Collector

In C# we don’t have to care about destruction of objects, this all will be done by garbage collector. Unfortunately with this we cannot determine the exact time, an object will be deleted, but sometimes this is necessary, for example if we have loaded a workspace with FARO LS Open and must set it free to delete the file outside of FARO LS Open. Therefore we can call the garbage collector manually to delete all unused objects. It must be used in the following way:

```
...
iQLibIf libRef = new iQLibIfClass();
...
// delete reference
libRef = null;

// run garbage collector
GC.Collect();
GC.WaitForPendingFinalizers();
```

8.3.1.4. Connection Point Usage

```
using IQOPENLib;
using LSSDKLib;
...
public class ScanTestSink : _IScanCtrlSDKEvents
{
  int cookie;
  IConnectionPoint icp;
  public void startScan()
  {
    string licenseCode = ... /* FARO LS license Code */;
    IiQLicensedInterfaceIf licSDKIf = new ScanCtrlSDKClass();
    licSDKIf.License = licenseCode;
    IScanCtrlSDK scanCtrl = (IScanCtrlSDK)licSDKIf;

    scanCtrl.ScannerIP = "132.154.24.13";
    scanCtrl.connect();
    scanCtrl.Resolution = 10;
    scanCtrl.syncParam();
    IConnectionPointContainer icpc =
        (IConnectionPointContainer)(scanCtrl);
    Guid IID_IScanCtrlSDKEvents =
         typeof(_IScanCtrlSDKEvents).GUID;
    icpc.FindConnectionPoint(ref IID_IScanCtrlSDKEvents,
                             out icp);
    icp.Advise(this, out cookie);
    ScanCtrl.startScan();
  }

  #region _IScanCtrlSDKEvents Members
  void _IScanCtrlSDKEvents.scanCompleted()
  {
    MessageBox.Show("Scan completed");
    icp.Unadvise(cookie);
  }
  #endregion
}
```

8.3.2. C++

It is recommended to have iQOpen.dll and FARO.LS.SDK.dll registered during development time. The import directive of FARO.LS.SDK.dll doesn't work without registration. After that, if you use Side-By-Side, you can run your application without registration. For more details see chapter 8.2.4.

Please note that the following examples use smart interface pointers. With these pointers, you don't have to call Release at the end of usage. Instead, you have to assign NULL to the smart interface pointer.

8.3.2.1. FARO Open

```
#import "C:\...\WinSxS\...\iQOpen.dll" no_namespace

...
CoInitialize(NULL);
// FARO LS Licensing
BSTR licenseCode = ... /* FARO LS license code */;
IiQLicensedInterfaceIfPtr liPtr(__uuidof(iQLibIf));
liPtr->License = licenseCode;
IiQLibIfPtr libRef = static_cast<IiQLibIfPtr>(liPtr);

libRef->load("c:\\temp\\demo.fls");
double x, y, z, angle;
libRef->getScanPosition(0, &x, &y, &z);
libRef->getScanOrientation(0, &x, &y, &z, &angle);
int numRows = libRef->getScanNumRows(0);
int numCOls = libRef->getScanNumCols(0);

// Access all points points by point
for (int col=0; col<numCols; col++)
  for (int row=0; row<numRows; row++) {
    double x, y, z;
    int refl;
    result = libRef->getScanPoint(0,row,col,&x,&y,&z,&refl);
    // ...
  }
```

```
// Access all points column per column in polar coordinates
double* positions = new double[numRows*3];
int* reflections = new int[numRows];
for (int col=0; col<numCols; col++) {
result = libRef->getPolarScanPoints(0, 0, col, numRows,
positions, reflections);
  for (int row=0 ; row<numRows ; row++) {
    double r, phi, theta;
    int refl;
    r = positions[3*row+0];
    phi = position[3*row+1];
    theta = positions[3*row+2];
    refl = reflections[row];
    // ...
  }
}

delete[] positions;
delete[] reflections;

libRef = NULL;
liPtr = NULL;
CoUninitialize();
...
```

8.3.2.2. FARO LS SDK

```
#import "C:\...\WinSxS\...\iQOpen.dll" no_namespace
#import "C:\...\WinSxS\...\FARO.LS.SDK.dll" no_namespace

...
CoInitialize(NULL);
// FARO LS Licensing
BSTR licenseCode = ... /* FARO LS license code */;
IiQLicensedInterfaceIfPtr liPtr(__uuidof(ScanCtrlSDK));
liPtr->License = licenseCode;
IScanCtrlSDKPtr scanCtrl = static_cast<IScanCtrlSDKPtr>(liPtr);

scanCtrl->ScannerIP = L"132.154.24.13";
scanCtrl->connect();

if (!scanCtrl->Connected)
  return MY_FALSE;

// set parameters.
scanCtrl->_ScanMode = StationaryGrey;
scanCtrl->HorizontalAngleMin = 0;
scanCtrl->HorizontalAngleMax = 360;
scanCtrl->VerticalAngleMin = -65;
scanCtrl->VerticalAngleMax = 90;
scanCtrl->Resolution = 10;
scanCtrl->ScanBaseName = L"myCPPScanFile";
scanCtrl->ScanFileNumber = 67;
scanCtrl->syncParam();

// wait a little bit for synchronizing scanner parameters.
Sleep(2000);
scanCtrl->startScan();

scanCtrl = NULL;
liPtr = NULL;
CoUninitialize();
...
```

It is important to import iQOpen.dll first and then FARO.LS.SDK.dll, otherwise it won't work. The import directives must contain the exact path to iQOpen.dll and FARO.LS.SDK.dll.

8.3.2.3. C++ Plug-in DLL Example

The following example code may serve as a skeleton for a plug-in DLL working with FARO LS SDK and FARO Open. Notice, you must edit the license key, FARO LS assembly version and the full dll paths to make the example work:

```
// DynLibrary.cpp
#define _WIN32_WINNT 0x0501          // target Windows XP
#define WIN32_LEAN_AND_MEAN          // exclude rarely-used stuff
#define ISOLATION_AWARE_ENABLED 1    // auto. context management
#include <windows.h>
#include <cassert>
// advice linker to add manifest entry
#pragma comment(linker, "\"/manifestdependency:type='win32' name='FARO.LS'
version='1.1.0.0' processorArchitecture='x86'
publicKeyToken='1d23f5635ba800ab'\"") // ← Edit version!
// import type library to create proxies
#import "C:\...\WinSxS\...\iQOpen.dll" no_namespace
#import "C:\...\WinSxS\...\FARO.LS.SDK.dll" no_namespace

HANDLE hActCtx = INVALID_HANDLE_VALUE;
class ActivationContext {
  ULONG_PTR cookie;
  BOOL okay;
public:
  explicit ActivationContext(HANDLE hActCtx = ::hActCtx) :
    cookie(0u),
    okay(ActivateActCtx(hActCtx, &cookie))
  {}
  ~ActivationContext() {
    if(okay)
      okay = DeactivateActCtx(0, cookie);
  }
  operator BOOL() const {
    return okay;
  }
};

extern "C" LPCTSTR __declspec(dllexport) DynProc()
{
  ActivationContext actctx;
  assert(actctx);
```

```
  const BSTR licenseCode = /* FARO LS license code */
    L"FARO Open Runtime License\n"
    L"Key:??????????????????????????\n"       // ← Edit key!
    L"\n"
    L"The software is the registered property of "
    L"FARO Scanner Production GmbH, Stuttgart, Germany.\n"
    L"All rights reserved.\n"
    L"This software may only be used with written permission "
    L"of FARO Scanner Production GmbH, Stuttgart, Germany.";

  IiQLicensedInterfaceIfPtr liPtr(__uuidof(ScanCtrlSDK));
  liPtr->License = licenseCode;
  IScanCtrlSDKPtr scanCtrl=static_cast<IScanCtrlSDKPtr>(liPtr);

  return scanCtrl->Connected ?
    L"We are connected." :
    L"We are not connected.";
}

BOOL APIENTRY DllMain(HMODULE hModule,
                      DWORD ul_reason_for_call,
                      LPVOID lpReserved)
{
  switch(ul_reason_for_call) {
  case DLL_PROCESS_ATTACH:
    if(!SUCCEEDED(CoInitialize(0)))
      return FALSE;
    if(!GetCurrentActCtx(&hActCtx)) {
      CoUninitialize();
      return FALSE;
    }
    break;
  case DLL_PROCESS_DETACH:
    if(hActCtx != INVALID_HANDLE_VALUE)
      ReleaseActCtx(hActCtx);
    CoUninitialize();
    break;
  }
  return TRUE;
}
```

Consuming the plug-in DLL does not require any special preparations. It suffices to load the plug-in DLL by means of LoadLibrary() and calling the exported functions retrieved via GetProcAddress():

```
// Application.cpp
#include <tchar.h>
#include <windows.h>
#include <iostream>

int _tmain(int argc, _TCHAR* argv[])
{
  const HMODULE hDLL = LoadLibrary(L"DynLibrary.dll");
  typedef LPCTSTR DynProcT();
  DynProcT* const DynProc = reinterpret_cast<DynProcT*>(
                              GetProcAddress(hDLL, "DynProc"));
  std::wcout << DynProc() << std::endl;
  FreeLibrary(hDLL);
  return 0;
}
```

8.3.3. Visual Basic

8.3.3.1. FARO Open

```
Dim libIf As iQLibIf
Set iQLibIf = CreateObject("iQvolution.iQLibIf")
licenseCode = "FARO Open Runtime License" & vbLf & _
              "Key:??????????????????????????" & vbLf & _
              vbLf & _
              "The software is the registered property " & _
              "of FARO Scanner Production GmbH, " & _
              "Stuttgart, Germany." & vbLf & _
              "All rights reserved." & vbLf & _
              "This software may only be used with " & _
              "written permission of FARO Scanner " & _
              "Production GmbH, Stuttgart, Germany."

iQLibIf.License = licenseCode

iQLibIf.Load ("c:\\temp\\demo.fls")
numRows = iQLibIf.GetScanNumRows(0)
```

8.3.3.2. FARO LS SDK

```
...
Dim ScanCtrl As ScanCtrlSDK
Set ScanCtrl = CreateObject("FARO.ScanCtrlSDK")
licenseCode = ...
ScanCtrl.License = licenseCode

ScanCtrl.ScannerIP = "132.154.24.13"
ScanCtrl.Connect

ScanCtrl.HorizontalAngleMin = 0
ScanCtrl.HorizontalAngleMax = 360
ScanCtrl.VerticalAngleMin = -65
ScanCtrl.VerticalAngleMax = 90
ScanCtrl.Resolution = 10
ScanCtrl.ScanBaseName = "myCPPScanFile"
ScanCtrl.ScanFileNumber = 67
ScanCtrl.SyncParam
ScanCtrl.StartScan
```

8.4 Converting from local to global coordinates

The easiest way to convert the local coordinates of scan points into global ones is to use matrix computations. If you want to calculate the global coordinates p_g of a scan point, you have to know its local coordinates p_l, the rotation matrix R, and the translation vector t:

$$p_g = R \cdot p_l + t$$

The translation vector t is given by `getScanPosition`.

The rotation matrix R can be computed from the values of `getScanOrientation`. `getScanOrientation` provides the rotation in so called axis-angle notation: the rotation vector $r = (r_x, r_y, r_z)^T$ and the rotation angle α.

$$R = \begin{pmatrix} r_x \cdot r_x \cdot (1-\cos -\alpha) + \cos -\alpha & r_y \cdot r_x \cdot (1-\cos -\alpha) - r_z \cdot \sin -\alpha & r_z \cdot r_x \cdot (1-\cos -\alpha) + r_y \cdot \sin -\alpha \\ r_x \cdot r_y \cdot (1-\cos -\alpha) + r_z \cdot \sin -\alpha & r_y \cdot r_y \cdot (1-\cos -\alpha) + \cos -\alpha & r_z \cdot r_y \cdot (1-\cos -\alpha) - r_x \cdot \sin -\alpha \\ r_x \cdot r_z \cdot (1-\cos -\alpha) - r_y \cdot \sin -\alpha & r_y \cdot r_z \cdot (1-\cos -\alpha) + r_x \cdot \sin -\alpha & r_z \cdot r_z \cdot (1-\cos -\alpha) + \cos -\alpha \end{pmatrix}$$

8.5 Known Problems

8.5.1. FARO Open

The performance to retrieve the scan points of a scan with `getScanPoint` depends on the number of scans in the workspace. The more scans the workspace has, the slower the access will be.

The arrays in `getXYZScanPoints` and `getPolarScanPoints` are not managed! Therefore these functions can only be used in C++!

8.5.2. FARO LS SDK

There are no known problems.

8.6 Help

- Compiler complains:
  `#import referenced a type from a missing type library`
  when importing FARO.LS.SDK.dll
  Solution: register iQopen.dll (see chapter 8.2.4)

License

FARO Technologies, Inc. grants you a non-exclusive, royalty-free license to copy, modify, and distribute the software solely for the purpose of facilitating interoperability between your software and the FARO Technologies Laser Scanner.

THIS CODE AND INFORMATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT. WE DISCLAIM ALL LIABILITY FOR ANY LOSS OR DAMAGE SUFFERED BY YOUR OR ANY THIRD PARTY AS A RESULT OF USING OR DISTRIBUTING THIS SOFTWARE, INCLUDING DIRECT, INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES (INCLUDING LOST SAVINGS, LOST PROFIT OR ATTORNEY FEES), HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, EVEN IF FARO HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Implementation Notes

paintlib Library

FARO LS SDK contains paintlib code. paintlib is copyright (c) 1996-2000 Ulrich von Zadow.

Libtiff Library

Copyright (c) 1988-1997 Sam Leffler

Copyright (c) 1991-1997 Silicon Graphics, Inc.

JPEG Library

This software is based in part on the work of the Independent JPEG Group.

KissFFT Library

Copyright (c) 2003,4 Mark Borgerding

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the author nor the names of any contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Open Source Computer Vision Library

Copyright (C) 2000, 2001, Intel Corporation, all rights reserved.

Third party copyrights are property of their respective owners.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistribution's of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistribution's in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- The name of Intel Corporation may not be used to endorse or promote products derived from this software without specific prior written permission.

This software is provided by the copyright holders and contributors "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the Intel Corporation or contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

GPL (GNU General Public License)

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps:

(1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the

public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommer-

cially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the

Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder

fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a

patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the

Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

Written Offer

If you, the owner of the scanner, like to get a copy of the source code of GPL covered parts of the scanner’s firmware, please contact our support team support@faroeurope.com. This offer is valid for three years and valid for as long as FARO offers spare parts or customer support for this product model.

LGPL (GNU Lesser General License)

The FARO LS SDK includes the following applications and libraries, which are covered by the LGPL:

libusb

Linux API

DirectFB

GNU LESSER GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

0. Additional Definitions.

As used herein, "this License" refers to version 3 of the GNU Lesser General Public License, and the "GNU GPL" refers to version 3 of the GNU General Public License.

"The Library" refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

An "Application" is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

A "Combined Work" is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the "Linked Version".

The "Minimal Corresponding Source" for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The "Corresponding Application Code" for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work.

1. Exception to Section 3 of the GNU GPL.

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL.

2. Conveying Modified Versions.

If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or

b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

3. Object Code Incorporating Material from Library Header Files.

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License.

b) Accompany the object code with a copy of the GNU GPL and this license document.

4. Combined Works.

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License.

b) Accompany the Combined Work with a copy of the GNU GPL and this license document.

c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document.

d) Do one of the following:

0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.

1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version

of the Library that is interface-compatible with the Linked Version.

e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the

Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

5. Combined Libraries.

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License.

b) Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

6. Revised Versions of the GNU Lesser General Public License.

The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License "or

any later version" applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy's public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.

Written Offer

If you, the owner of the scanner, like to get a copy of the source code of LGPL covered parts of the scanner's firmware, please contact our support team support@faroeurope.com. This offer is valid for three years and valid for as long as FARO offers spare parts or customer support for this product model.

GEOTRANS

The product was developed using GEOTRANS, a product of the National Geospatial-Intelligence Agency (NGA) and U.S. Army Engineering Research and Development Center.

Trademarks

FARO and FARO Laser Scanner Focus^{3D} are registered trademarks or trademarks of FARO Technologies Inc. All other brand and product names are trademarks or registered trademarks of their respective companies.

Microsoft, Visual Studio, Visual Basic, Excel, Windows, Windows 7 and Windows 8 are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

FARO Technologies, Inc.
250 Technology Park
Lake Mary, FL 32746
Tel. (800)-736-2771 U.S. / +1 407-333-3182 Worldwide
E-Mail: support@faro.com

FARO Europe GmbH & Co. KG
Lingwiesenstrasse 11/2
D-70825 Korntal-Münchingen, Germany
Tel: +49 7150/9797-400 (FREECALL +800 3276 7378)
Fax: +49 7150/9797-9400 (FREEFAX +800 3276 1737)
E-Mail: support@faroeurope.com

FARO Singapore Pte. Ltd.
No. 03 Changi South Street 2
#01-01 Xilin Districentre Building B
SINGAPORE 486548
TEL: +65 6511.1350
E-Mail: supportap@faro.com

FARO Japan, Inc.
716 Kumada, Nagakute-city,
Aichi, 480-1144, Japan
Tel: 0120-922-927, 0561-63-1411
FAX:0561-63-1412
E-Mail: supportjapan@faro.com

FARO (Shanghai) Co., Ltd.
1/F, Building No. 2,
Juxin Information Technology Park
188 Pingfu Road, Xuhui District
Shanghai 200231, China
Tel.: 400.677.6826
Email: supportchina@faro.com

FARO Business Technologies India Pvt. Ltd.
E-12, B-1 Extension,
Mohan Cooperative Industrial Estate,
New Delhi-110044
India
Tel.: 1800.1028456
Email: supportindia@faro.com