TILL Photonics Imaging Microscopy

Software Development Kit

Reference Manual

0.2.19

Generated by Doxygen 1.4.4

15 Mar 2006

TILL Photonics GmbH
Lochhamer Schlag 19
82166 Gräfelfing
Germany

Contents

1 TILL Photonics iMIC SDK

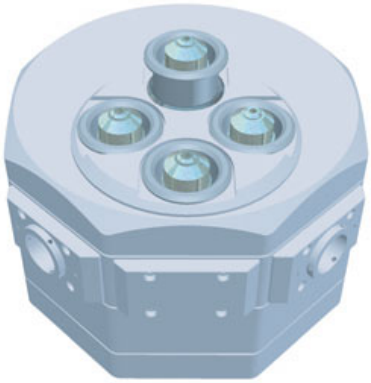

Figure 1.1: TILL Photonics Imaging Microscope – iMIC

The iMIC is a fully motorized imaging platform, ideally suited for software controlled, automated screening, laser scanning and PC-based microscopy.

The first iMIC configuration consists of a stage with moving directions X and Y, a Z-focus, as well as a fine-focus - the piezo drive. Additionally the iMIC has an objective turret and a filter changer. Future iMIC configurations may vary in the existance or quantity of the before mentioned moving axes.

This document describes the TILL Photonics Imaging Microscopy Software Development Kit (iMIC SDK). The SDK allows applications running on a host computer to control one or more iMIC devices.

This SDK deals with the mechanical properties of the iMIC as the XY stage, the Z focus, the objective turret, and the filter changers.

The iMIC SDK will provide bindings, Application Programming Interfaces (APIs), for several popular application programming languages and environments. This reference documentation describes the C iMIC API in detail.

2 Module Documentation

2.1 iMIC Application Programming Interface (API)

This module holds functions to control the TILL Photonics iMIC.

iMIC Management

The functions listed in this section are used to manage the connection between the application and the iMIC.

Attention:

It is important to initialize the iMIC device after connecting to it. Otherwise all other functions, except the iMIC management, stay disabled.

- int IMIC_OpenByRS232 (const char *port, void **handle)

  Connecting to a specific iMIC device.

- int IMIC_OpenByCallback (void *refCon, LockPortCb *lock, UnlockPortCb *unlock, WriteToPortCb *write, ReadFromPortCb *read, void **handle)

  Connecting to a specific iMIC device.

- int IMIC_Close (void *handle)

  Disconnecting from an iMIC device.

- int IMIC_Init (void *handle, InitProgressCb *ips)

  Initialize an iMIC device.

- int IMIC_InitFilterChanger (void *handle, int filterIndex)

  Reinitialize the filter changer.

- int IMIC_IsInit (void *handle, bool *init)

  Checks, if the selected iMIC device was initialized.

iMIC Axis Properties

The functions listed in this section are used to obtain information about the axes of the iMIC. The distance properties of the X, Y, and Z axes are given as floating point values measured in millimeters [mm]. The discrete objective turret and the filter changer properties are integers specifying the movement units.

The property functions make use of an enum of possible values:

Enum	Interpretation
Min	The minimum position of an axis.
Res	The smallest possible unit of movement.
Max	The maximum position of an axis.
not_a_property	This is a dummy end marker.

It is guaranteed for further changes that 'Min' will always stay the first and 'Max' always the second to the last value in this enum list. Thus it is possible to iterate over the enum if desired.

- int IMIC_GetXProp (void *handle, Property propIndex, double *propValue)

  Returns the current property of the **X** *axis.*

- int IMIC_GetYProp (void *handle, Property propIndex, double *propValue)

  Returns the current property of the **Y** *axis.*

- int IMIC_GetZProp (void *handle, int zAxisIndex, Property propIndex, double *propValue)

  Returns the current property of the **Z** *axis.*

- int IMIC_GetObjectiveTurretProp (void *handle, Property propIndex, int *propValue)

  Returns the current property of the **objective** *turret.*

- int IMIC_GetFilterChangerProp (void *handle, int filterIndex, Property propIndex, int *propValue)

  Returns the current property of the **filter**.

- int IMIC_GetNumberOfZAxes (void *handle, int *number)

  Returns the current number of **axes**.

- int IMIC_GetNumberOfFilterChangers (void *handle, int *number)

  Returns the current number of **filter changers**.

iMIC Axis Position

The functions listed in this section are used to obtain the position of a selected axis of the iMIC. The position of the X, Y, and Z axes are given as floating point values measured in millimeters [mm]. The discrete objective turret and the filter changer positions are integers specifying the movement units. In case of IMIC_GetXYPos a NULL pointer may be provided for each of the positions if the value is not needed.

- int IMIC_GetXPos (void *handle, double *xPosition)

  Returns the current position of the **X** *axis.*

- int IMIC_GetXYPos (void *handle, double *xPosition, double *yPosition)

  Returns the current positions of the **X** *and* **Y** *axes.*

- int IMIC_GetYPos (void *handle, double *yPosition)

  Returns the current position of the **Y** *axis.*

- int IMIC_GetZPos (void *handle, int zAxisIndex, double *position)

  Returns the current position of the **Z** *axis.*

- int IMIC_GetObjectiveTurretPos (void *handle, int *position)

  Returns the current position of the **objective** *turret.*

- int IMIC_GetFilterChangerPos (void *handle, int filterIndex, int *position)

  Returns the current position of the selected **filter** *axis.*

iMIC Positioning Operations

The functions listed in this section are used to control the movement of the iMIC. The position of the X, Y, and Z axes are given as floating point values measured in millimeters [mm]. The objective turret and the filter changer function receive integers specifying the distinct positions.

- int IMIC_SetXYPosAbs (void *handle, double positionX, double positionY)

  Moves the iMIC **stage** *absolute to the origin.*

- int IMIC_SetXYPosRel (void *handle, double positionX, double positionY)

  Moves the iMIC **stage** *relative to the last* **X** *and* **Y** *position.*

- int IMIC_SetXPosAbs (void *handle, double position)

  Moves the iMIC **stage** *absolute to the origin.*

- int IMIC_SetXPosRel (void *handle, double position)

  Moves the iMIC **stage** *relative to the last* **X** *position.*

- int IMIC_SetYPosAbs (void *handle, double position)

  Moves the iMIC **stage** *absolute to the origin.*

- int IMIC_SetYPosRel (void *handle, double position)

  Moves the iMIC **stage** *relative to the last* **Y** *position.*

- int IMIC_SetZPosAbs (void *handle, int zAxisIndex, double position)

  Moves the iMIC **focus** *(Z) absolute to the origin.*

- int IMIC_SetZPosRel (void *handle, int zAxisIndex, double position)

  Moves the iMIC **focus** *(Z) relative to the last position.*

- int IMIC_SetObjectiveTurretPosAbs (void *handle, int position)

  Moves the iMIC **objective** *turret absolute to the origin.*

- int IMIC_SetObjectiveTurretPosRel (void *handle, int position)

  Moves the iMIC **turret** *relative to the last position.*

- int IMIC_SetFilterChangerPosAbs (void *handle, int filterIndex, int position)

  Moves the iMIC **filters** *absolute to the origin.*

- int IMIC_SetFilterChangerPosRel (void *handle, int filterIndex, int position)

  Moves the iMIC **filters** *relative to the last position.*

- int IMIC_SetFilterChangerServicePos (void *handle, int filterIndex)

  Moves the iMIC **filter changer** *to its service position.*

2.1.1 Detailed Description

This module holds functions to control the TILL Photonics iMIC.

2.1.2 Error Codes

All functions return an integer error code that can be interpreted as follows:

Error Code	Enumeration	Interpretation
0	IMIC_ErrorNone	no error
1	IMIC_ErrorMissingAxis	missing axis
2	IMIC_ErrorMissingZAxis	missing Z axis
3	IMIC_ErrorMissingFilter	missing filter
4	IMIC_ErrorNotInitialized	iMIC not initialized
5	IMIC_ErrorNoProperty	no property
6	IMIC_ErrorWrongHandle	wrong handle
7	IMIC_ErrorFinishProtocol	protocol not finished
8	IMIC_ErrorIllegalUser-Parameter	user parameter to long (>32 bits)
9	IMIC_ErrorWrongUser-Parameter	no user parameter handle
10	IMIC_ErrorOpenComPort-Failed	failed to open com port
11	IMIC_ErrorInitFailed	failed to initialize
12	IMIC_ErrorUnknown	unknown error
13	IMIC_ErrorOutOfRange	out of range
14	IMIC_ErrorReturn-ArgumentIsNull	return argument not initialized (==NULL)

2.1.3 Function Documentation

2.1.3.1 int IMIC_OpenByRS232 (const char * *port*, void ** *handle*)

Connecting to a specific iMIC device.

The iMIC is selected by the com port. On successful connection a valid iMIC handle is assigned by the function.

Remarks:

For LabView users it is adequate to take a long integer variable instead of the void-pointer-pointer (void**).

Returns:

An integer error code.

Parameters:

port The com port number of an existing iMIC device.

handle The iMIC device handle for the given com port.

2.1.3.2 int IMIC_GetXProp (void * *handle*, Property *propIndex*, double * *propValue*)

Returns the current property of the **X** axis.

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.

propIndex The index of the selected property.

propValue The current property of the **X** axis.

2.1.3.3 int IMIC_GetXPos (void * *handle*, double * *xPosition*)

Returns the current position of the **X** axis.

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.

xPosition The current position of the **X** axis.

2.1.3.4 int IMIC_SetXYPosAbs (void * *handle*, double *positionX*, double *positionY*)

Moves the iMIC **stage** absolute to the origin.

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.

positionX The new **X** position absolute to the origin.

positionY The new **Y** position absolute to the origin.

2.1.3.5 int IMIC_OpenByCallback (void * *refCon*, LockPortCb * *lock*, UnlockPortCb * *unlock*, WriteToPortCb * *write*, ReadFromPortCb * *read*, void ** *handle*)

Connecting to a specific iMIC device.

The connection to the iMIC is managed by a set of callback functions provided by the caller. On successful connection a valid iMIC handle is assigned by the function.

Returns:
An integer error code.

Parameters:
refCon Connection to an external reference handle.

lock Callback function for unlocking the communication.

unlock Callback function for locking the communication.

write Callback function allowing for writing through the communication.

read Callback function allowing for reading through the communication.

handle The iMIC device handle for the given index.

2.1.3.6 int IMIC_Close (void * *handle*)

Disconnecting from an iMIC device.

On disconnection the iMIC handle will be invalidated and all previous made settings are lost. It is recommended to disconnect from an iMIC by using this function. Thus the device has the chance to prepare an idle position before powering down the system.

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.

2.1.3.7 int IMIC_Init (void * *handle*, InitProgressCb * *ips*)

Initialize an iMIC device.

After connecting to an iMIC it is essential to initialize the device first. Without initialization all other function, except to the iMIC management functions, are disabled. The initialization takes care of finding the axes boundaries and preparing the initial starting position for all given axes.

Attention:
The iMIC will be reinitialized on a second call of the function!!!

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.

ips An init function-pointer to a prograss callback function.

2.1.3.8 int IMIC_InitFilterChanger (void * *handle*, int *filterIndex*)

Reinitialize the filter changer.

After setting the filter changer to its service position it is essential to reinitialize the axis again. Without initialization all other filter changer functions are disabled. The initialization takes care of finding the axes boundaries and preparing the initial starting position for the filter changer.

Attention:
The iMIC will be reinitialized on a second call of the function!!!

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.

filterIndex The index of the currently selected filter changer. This index is 0-based.

2.1.3.9 int IMIC_IsInit (void * *handle*, bool * *init*)

Checks, if the selected iMIC device was initialized.

This function directly corresponds to the initialization of the iMIC. Without initialization all other function, except to the iMIC management functions, are disabled.

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.

init TRUE, if the selected iMIC was initialized, else FALSE.

2.1.3.10 int IMIC_GetYProp (void * *handle*, Property *propIndex*, double * *propValue*)

Returns the current property of the **Y** axis.

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.
propIndex The index of the selected property.
propValue The current property of the **Y** axis.

2.1.3.11 int IMIC_GetZProp (void * *handle*, int *zAxisIndex*, Property *propIndex*, double * *propValue*)

Returns the current property of the **Z** axis.

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.
zAxisIndex The index of the selected **Z** axis. This index is 0-based.
propIndex The index of the selected property.
propValue The current property of the **Z** axis.

2.1.3.12 int IMIC_GetObjectiveTurretProp (void * *handle*, Property *propIndex*, int * *propValue*)

Returns the current property of the **objective** turret.

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.
propIndex The index of the selected property.
propValue The current property of the **objective** turret.

2.1.3.13 int IMIC_GetFilterChangerProp (void * *handle*, int *filterIndex*, Property *propIndex*, int * *propValue*)

Returns the current property of the **filter**.

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.
filterIndex The index of the selected filter changer. This index is 0-based.
propIndex The index of the selected property.
propValue The current property of the **filter**.

2.1.3.14 int IMIC_GetNumberOfZAxes (void * *handle*, int * *number*)

Returns the current number of **axes**.

The indices are 0-based. Thus, the last legal index is `number-1`.

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.
number The quantity of given **axes**.

2.1.3.15 int IMIC_GetNumberOfFilterChangers (void * *handle*, int * *number*)

Returns the current number of **filter changers**.

The indices are 0-based. Thus, the last legal index is `number-1`.

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.
number The quantity of given **filter changers**.

2.1.3.16 int IMIC_GetXYPos (void * *handle*, double * *xPosition*, double * *yPosition*)

Returns the current positions of the **X** and **Y** axes.

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.

xPosition The current position of the **X** axis.

yPosition The current position of the **Y** axis.

2.1.3.17 int IMIC_GetYPos (void * *handle*, double * *yPosition*)

Returns the current position of the **Y** axis.

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.

yPosition The current position of the **Y** axis.

2.1.3.18 int IMIC_GetZPos (void * *handle*, int *zAxisIndex*, double * *position*)

Returns the current position of the **Z** axis.

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.

zAxisIndex The index of the selected **Z** axis. This index is 0-based.

position The current position of the **Z** axis.

2.1.3.19 int IMIC_GetObjectiveTurretPos (void * *handle*, int * *position*)

Returns the current position of the **objective** turret.

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.

position The current position of the **objective** turret.

2.1.3.20 int IMIC_GetFilterChangerPos (void * *handle*, int *filterIndex*, int * *position*)

Returns the current position of the selected **filter** axis.

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.

filterIndex The index of the currently selected filter changer. This index is 0-based.

position The current position of the selected **filter** axis.

2.1.3.21 int IMIC_SetXYPosRel (void * *handle*, double *positionX*, double *positionY*)

Moves the iMIC **stage** relative to the last **X** and **Y** position.

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.

positionX The new **X** position relative to the last position.

positionY The new **Y** position relative to the last position.

2.1.3.22 int IMIC_SetXPosAbs (void * *handle*, double *position*)

Moves the iMIC **stage** absolute to the origin.

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.

position The new **X** position absolute to the origin.

2.1.3.23 int IMIC_SetXPosRel (void * *handle*, double *position*)

Moves the iMIC **stage** relative to the last **X** position.

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.

position The new **X** position relative to the last position.

2.1.3.24 int IMIC_SetYPosAbs (void * *handle*, double *position*)

Moves the iMIC **stage** absolute to the origin.

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.

position The new **Y** position absolute to the origin.

2.1.3.25 int IMIC_SetYPosRel (void * *handle*, double *position*)

Moves the iMIC **stage** relative to the last **Y** position.

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.

position The new **Y** position relative to the last position.

2.1.3.26 int IMIC_SetZPosAbs (void * *handle*, int *zAxisIndex*, double *position*)

Moves the iMIC **focus** (Z) absolute to the origin.

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.

zAxisIndex The index of the selected **Z** axis. This index is 0-based.

position The new **focus** (Z) position absolute to the origin.

2.1.3.27 int IMIC_SetZPosRel (void * *handle*, int *zAxisIndex*, double *position*)

Moves the iMIC **focus** (Z) relative to the last position.

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.

zAxisIndex The index of the selected **Z** axis. This index is 0-based.

position The new **focus** (Z) position relative to the last position.

2.1.3.28 int IMIC_SetObjectiveTurretPosAbs (void * *handle*, int *position*)

Moves the iMIC **objective** turret absolute to the origin.

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.

position The new **objective** turret position absolute to the origin.

2.1.3.29 int IMIC_SetObjectiveTurretPosRel (void * *handle*, int *position*)

Moves the iMIC **turret** relative to the last position.

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.

position The new **turret** position relative to the last position.

2.1.3.30 int IMIC_SetFilterChangerPosAbs (void * *handle*, int *filterIndex*, int *position*)

Moves the iMIC **filters** absolute to the origin.

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.

filterIndex The index of the currently selected filter changer. This index is 0-based.

position The new **filters** position absolute to the origin.

2.1.3.31 int IMIC_SetFilterChangerPosRel (void * *handle*, int *filterIndex*, int *position*)

Moves the iMIC **filters** relative to the last position.

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.

filterIndex The index of the currently selected filter changer. This index is 0-based.

position The new **filters** position relative to the last position.

2.1.3.32 int IMIC_SetFilterChangerServicePos (void * *handle*, int *filterIndex*)

Moves the iMIC **filter changer** to its service position.

From here, filters can easily be removed or exchanged.

Attention:
Before invoking this command it must be ensured that both filter changer doors are opened to avoid any damage to the hardware. Afterwards make sure to reinitialize the filter changer with IMIC_InitFilterChanger.

Returns:
An integer error code.

Parameters:
handle The iMIC device handle.

filterIndex The index of the currently selected filter changer. This index is 0-based.