

# Standalone Filter Wheel Driver Interface

The ability to handle standalone filter wheels (filter wheels not controlled through the camera, but utilizing own communication and power lines and thus needing own drivers, independent on camera drivers) was added to SIPS version 4. The Moravian Instruments SFW-XL series of standalone filter wheels is an example of such devices.

This document describes the interface between SIPS and standalone filter wheel drivers. It can be used for both implementing of a new standalone filter wheel driver for SIPS and also for using of the driver DLLs to control Moravian Instruments SFW form other software packages.

Two driver DLLs are used to control all current Moravian Instruments SFW:

- **'sfw.dll'** handles the SFW connected directly to the host computer through USB line.
- **'sfweth.dll'** handles the SFW connected to the host computer over TCP/IP network. In such case the **Moravian Camera Ethernet Adapter** device is necessary. Up to four Gx/Cx cameras of arbitrary type and/or standalone filter wheels can be connected to this device using standard USB lines on one side and the standard 1 Gbps Ethernet interface (compatible with older 10/100 Mbps networks) can be attached on the other side. Because the TCP/IP protocol can be routed, the distance between camera and filter wheel and the host PC is virtually unlimited.

Please note SFW depends on the firmware version of the Camera Ethernet Adapter device. Refer to the Camera Ethernet Adapter User's Guide for procedures to check and/or update the device firmware.

Please note the USB connected SFW requires installation of the kernel driver on the particular computer. The USB kernel drivers are identified by USB Vendor and Product identifiers. The operating system takes care of these drivers and the user usually needs not to install them manually (refer to the **Installing and Using Drivers and Software** manual for kernel driver installation instructions, please). Of course, no kernel drivers are installed when the Moravian Camera Ethernet Adapter is used to communicate with filter wheels.

## Driver DLL Interface

SIPS drivers are Dynamic Link Libraries (DLLs) with specific set of exported functions. All exported functions use standard "C" calling conventions, so they can be called from arbitrary software package without problems with name decorations etc.

C calling conventions apply to 32-bit versions of supplied DLLs. There is only one universal calling convention in 64-bit mode, so specifications of calling conventions in the source code are ignored.

Some functions are mandatory and the DLL must implement them to be accepted as a SIPS driver. Other functions may be optional and the particular functionality are used only if the driver exports (implements) these functions.

## Driver API function description

Exported functions are grouped according to functionality. Individual function description follows this list of all exported functions. Functions marked \* are optional, SIPS does not require them to be implemented (exported from the driver DLL).

Driver life-cycle functions:

```
Enumerate
Initialize
Configure*
Release
RegisterNotifyHWND
```

Driver information functions:

```
GetBoolean
GetInteger
GetString
```

Filter wheel functions:

```
EnumerateFilters
SetFilter
ReinitFilterWheel*
```

Data types used in SFW driver Application Programming Interface:

```
typedef int          INTEGER;
typedef unsigned int  CARDINAL;
typedef unsigned char CHAR;
typedef unsigned char BOOLEAN;
typedef void *        ADDRESS;
```

```
struct CSFW;
```

### Driver API function description

```
void __cdecl Enumerate(
    void ( __cdecl *CallbackProc )( CARDINAL )
)
```

Mandatory function.

Enumerate allows discovering of all filter wheels currently connected to the host PC. Its argument is a pointer to callback function 'CallbackProc' with single unsigned integer argument. This callback function is called for each connected filter wheels and the filter wheel identifier is passed as an argument. Application can then offer the user a list of all connected filter wheels to choose the one with which the user wants to work etc.

If the application is designed to work with one filter wheel only or the filter wheel identifier is known (for instance the application scans some .INI file where the identifier is defined), 'Enumerate' needs not to be called at all.

Please note the callback function uses the “C” calling conventions, as opposed to “stdcall” conventions often used in the Windows OS.

```
CSFW* __cdecl Initialize(  
    CARDINAL Id  
)
```

Mandatory function.

User-space driver is designed to handle multiple filter wheels at once. Driver distinguishes individual cameras using handles to individual instances. Handle is defined as unsigned integer of the size corresponding to the size of the address (32 bits or 64 bits). This allows the driver implementation to allocate class/structure and return a pointer to it as a handle.

This handle is returned by the 'Initialize' function. It is necessary to pass the filter wheel identifier to the Initialize (see the 'Enumerate' function description). If the 'Initialize' returns 0xFFFFFFFF (INVALID\_HANDLE\_VALUE), the particular filter wheel cannot be used. It is either not connected or is already used by some other application.

```
BOOLEAN __cdecl Configure(  
    CSFW *PSFW,  
    ADDRESS ParentHWND  
)
```

Optional function.

If the driver requires configuration (for instance TCP/IP based filter wheel requires definition of IP address), it should implement this function. This function can open a dialog box (and use the ParentHWND passed from the user application).

This function can use called already enumerated Handle to re-configure existing instance of driver or the Handle can be -1, in which case the configuration affects all instances of the particular driver. User application should allow calling of 'Configure' function with Handle = -1 also in the case 'Enumerate' did not return any filter wheel.

SIPS always offer an empty line marked “unconfigured” for the drive exporting this function. After the configuration finished, SIPS calls 'Enumerate' to get a fresh list of available cameras (e.g. from new Moravian Camera Ethernet Adapter device).

```
void __cdecl Release(
    CSFW *PSFW
)
```

Mandatory function.

When the filter wheel is no longer used, the handle must be released by the 'Release' call. No other function (with the exception of 'Enumerate' and 'Initialize') may be called after the 'Release' call.

```
void __cdecl RegisterNotifyHWND(
    CSFW *PSFW,
    ADDRESS NotifyHWND
)
```

Mandatory function.

The driver can notify the application the the filter wheel was plugged or unplugged. Notifications are sent as Windows messages to the windows, which HWND was passed as an argument to the 'RegisterNotifyHWND' function. Notification messages are:

```
#define WM_FW_CONNECT      1039
#define WM_FW_DISCONNECT   1040
```

When the application is no longer interested in this notification, it can call 'RegisterNotifyHWND' with NULL as a handle.

Calling Release automatically calls 'RegisterNotifyHWND' with NotifyHWND = NULL.

```
BOOLEAN __cdecl GetBoolean(
    CSFW *PSFW,
    CARDINAL Index,
    BOOLEAN *Boolean
)
```

Mandatory function.

Function 'GetBoolean' returns a boolean value depending on the Index parameter. If the function does not "understand" passed Index, it returns FALSE.

gbConnected	= 0	TRUE if filter wheel currently connected
gbInitialized	= 1	TRUE if filter wheel initialized (zero filter position found)
gbMicrometerFilterOffsets	= 2	TRUE if filter focusing offsets are expressed in micrometers
gbpConfigured	= 127	TRUE if camera is configured

```

BOOLEAN __cdecl GetInteger(
    CSFW      *PSFW,
    CARDINAL  Index,
    INTEGER   *Num
)

```

Mandatory function.

Function 'GetInteger' returns integer value depending on the Index parameter. If the function does not “understand” passed Index, it returns FALSE.

giVersion1	= 0	Filter wheel drive major version
giVersion2	= 1	Filter wheel drive minor version
giVersion3	= 2	Filter wheel drive build version
giVersion4	= 3	Filter wheel drive release version
giFilterWheelId	= 4	Unique identifier of the standalone filter wheel
giFilters	= 5	Number of filters available

```

BOOLEAN __cdecl GetString(
    CSFW      *PSFW,
    CARDINAL  Index,
    CARDINAL  String_HIGH,
    CHAR      *String
)

```

Mandatory function.

Function 'GetString' returns string value depending on the Index parameter. If the function does not “understand” passed Index, it returns FALSE.

The string is copied to the buffer pointed by the String parameter. The function checks the maximum buffer size not to cause buffer overrun. The highest character index (index starts with 0) of the buffer must be passed as String\_HIGH parameter. If the buffer is longer than the passed string, terminating zero character is also copied.

gsFWDescription	= 0	Filter wheel description
gsManufacturer	= 1	Manufacturer name
gsSerialNumber	= 2	Filter wheel serial number

```

BOOLEAN __cdecl EnumerateFilters(
    CSFW      *PSFW,
    CARDINAL  Index,
    CARDINAL  Description_HIGH,
    CHAR      *Description,
    CARDINAL  *Color,
    INTEGER   *Offset
) ;

```

Mandatory function.

Enumerates all filters provided by the filter wheel. This enumeration does not use any callback, but the caller passes index beginning with 0 and repeats the call with incremented index until the call returns FALSE. Description string is passed similarly to GetString call. Color parameter hints the Windows color, which can be used to draw the filter name in the application. The 'Offset' parameter indicates the focuser shift when the particular filter is selected.

Units of the 'Offset' can be micrometers or arbitrary focuser specific units (steps). If the units used are micrometers, driver should return TRUE from GetBoolean( gbMicrometerFilterOffsets, ...) call.

```
BOOLEAN __cdecl SetFilter(  
    CSFW      *PSFW,  
    CARDINAL  FilterIndex  
);
```

Mandatory function.

Sets the required filter.

```
BOOLEAN __cdecl SetFilter2(  
    CSFW      *PSFW,  
    CARDINAL  FilterIndex1,  
    CARDINAL  FilterIndex2  
);
```

Optional function.

This function works with dual wheel SFW series only, it fails if used with IFW or EFW wheel. The dual wheel normally always sets the first (clear) position of each wheel if the filter from another wheel is used. However, this function allows to set any combination of positions in both wheels. So, this function can be useful for some special applications, requiring combination of two filters.

```
BOOLEAN __cdecl ReinitFilterWheel(  
    CSFW *PSFW  
);
```

Optional function.

The filter wheel performs the initialization, during which the zero filter position is found and set.

## Document updates

2024-07-09: Version 1.0 of driver DLLs released.

2025-06-25: Version 1.1 of driver DLLs released.

- The **SetFilter2** API function, intended to control dual wheel SFW series only, was added.