EFW filter wheel
Software Development Kit

Revision:1.8.4

2025.11.18

Overview

EFW Filter Wheel, designed for effortless operation, requires no additional drivers and offers plug-and-play functionality for straightforward handling. Its low-power design allows direct power supply via computer, ASIAIR, or camera, ensuring flexible and hassle-free use. Available in multiple sizes—including standard and mini versions—to accommodate diverse needs. Highly compatible with mainstream filters, it seamlessly integrates with ASCOM-enabled third-party software, making it an ideal choice for astrophotography and observation.

This document is intended to guide users in setting up an EFW device environment using the SDK, explaining the process of implementing EFW communication through the SDK.

Table of Contents

1 SDK Usage Guide

Change History

Change date	revision	comment
2025.11.4	1.8.3	Add API EFWCheck
2017.2.14	2.1	Add API EFWCalibrate
2016.11.15	2.0	Add EFW_ERROR_CODE： EFW_ERROR_CLOSED Add EFWGetProductIDs EFWOpen: change argument to ID

This Software Development Kit (SDK) describes a set of functions that can be used to operate EFW filter wheel devices. These functions are accessible through development tools such as C, C++, and C#, and are compatible with Windows, Linux, and macOS operating systems on x86 or x64 architectures, as well as Linux on ARM platforms.

Header file: EFW_filter.h

Import library and dynamic library for Windows: EFW_filter.lib, EFW_filter.dll

Dynamic library and static library for Linux: libEFWfilter.so, libEFWfilter.a

Dynamic library and static library for macOS: libEFWfilter.dylib, libEFWfilter.a

Installation method:

On Windows, download and extract the ZIP file to any directory. Add the DLL path to your system environment variables. You may need to log out and log back in. Alternatively, place the DLL in the folder containing your application executable.

Preparation:

USB:

1. An EFW filter wheel device
2. PC running Windows
3. USB cable

2 Definition of Enum-type and Struct

2.1 typedef struct _EFW_INFO

```
{
    int ID;
    char Name[64];
    int slotNum;
} EFW_INFO;
```

Structure containing EFW device ID, device name, and filter wheel count information

2.2 typedef enum _EFW_ERROR_CODE{

```
    EFW_SUCCESS = 0,// Operation successful
    EFW_ERROR_INVALID_INDEX,// Invalid index
    EFW_ERROR_INVALID_ID,// Invalid ID value
    EFW_ERROR_INVALID_VALUE,// Invalid parameter
    EFW_ERROR_REMOVED, // Device not found, detected removal
    EFW_ERROR_MOVING,// Filter wheel currently in motion
    EFW_ERROR_ERROR_STATE,// Filter wheel state error
    EFW_ERROR_GENERAL_ERROR,// Other error
    EFW_ERROR_NOT_SUPPORTED,// Device not supported
    EFW_ERROR_INVALID_LENGTH,// Data length not supported
    EFW_ERROR_CLOSED,// Device closed
    EFW_ERROR_END = -1
}EFW_ERROR_CODE;
```

EFW Device Error Codes

2.3 typedef struct _EFW_ID

```
{
    unsigned char id[8];
}EFW_ID;
```

ID value encapsulation structure

3 Function Declaration

3.1 EFWGetNum

Syntax: int EFWGetNum()

Purpose: Retrieves the number of inserted EFW devices

Description:

The int return value indicates the number of EFW devices detected. A return value of 1 signifies that one EFW device is connected.

Example:

```
int idnum = EFWGetNum();
if (idnum <= 0) {
    std::cout << "Don't find EFW Device ! " << std::endl;
    return -1;
}
```

Note:

This is the first function that must be called when using the EFW SDK.

3.2 EFWGetProductIDs

Syntax: int EFWGetProductIDs(int* pPIDs)

Purpose: Retrieves EFW product IDs.

Description:

Retrieves the product ID for each filter wheel. When using this function, first set pPIDs to 0 to obtain the array length, then allocate a buffer of sufficient size to load all PIDs.

Note:

This API will be deprecated. Please use EFWCheck instead.

3.3 EFW_API int EFWCheck

Syntax: int EFWCheck(int iVID, int iPID)

Purpose: Checks whether the device is an EFW.

Description:

int iVID: The device's VID. For EFW, this value is 0x03C3.

int iPID: The device's PID.

Example:

```
int* pPIDs = new int[numPIDs];
EFW_ERROR_CODE err = EFWGetID(idnum - 1, pPIDs);
ret = EFWCheck(0x03C3, pPIDs[idnum - 1]);
if (ret == 1){
    std::cout << "Confirm that it is an EFW device!" << std::endl;
}else{
    std::cout << "Device is not EFW!" << std::endl;
}
```

3.4 EFWGetID

Syntax: EFW_ERROR_CODE EFWGetID(int index, int* ID)

Purpose: Retrieves the ID value of a connected EFW device.

Description:

int index: The index value of the filter wheel, ranging from 0 to N - 1, where N is returned by the GetNum() function.

int* ID: Pointer to the filter wheel ID. This ID is a unique integer ranging from 0 to EFW_ID_MAX - 1. After the device is opened, all subsequent operations rely on this ID, which remains unchanged during use.

Return Values:

EFW_ERROR_INVALID_INDEX: Indicates an invalid index value was passed.

EFW_SUCCESS: Indicates the operation succeeded.

Note:

This ID value will be used in nearly all control and information retrieval APIs.

3.5 EFWOpen

Syntax: EFW_ERROR_CODE EFWOpen(int ID)

Purpose: Opens the EFW device with the specified ID. After executing this step, you will gain control over the EFW device with the specified ID value.

3.6 EFWGetProperty

Syntax: EFW_ERROR_CODE EFWGetProperty(int ID, EFW_INFO *pInfo)

Purpose: Retrieves information about the EFW device with the specified ID value.

Description:

EFW_INFO *pInfo: Pointer to the EFW information structure.

int ID: The ID value of the EFW device being controlled.

Note:

This function must be used before controlling the EFW device, i.e., before calling EFWSetPosition, because EFWSetPosition requires the number of filter wheels obtained via EFWGetProperty.

3.7 EFWGetPosition

Syntax: EFW_ERROR_CODE EFWGetPosition(int ID, int *pPosition)

Purpose: Retrieves the current filter wheel position used by the EFW device.

Description:

int ID: The ID value of the EFW device being controlled.

int *pPosition: Pointer to the filter slot position. This value ranges from 0 to M - 1, where M is the total number of slots. When the filter wheel is rotating, this value is -1.

3.8 EFWSetPosition

Syntax: EFW_ERROR_CODE EFWSetPosition(int ID, int Position)

Purpose: Retrieves the current filter wheel position used by the EFW device.

Description:

int ID: The filter wheel ID.

int Position: The slot position, ranging from 0 to M - 1, where M is the total number of slots.

3.9 EFWSetDirection

Syntax: EFW_ERROR_CODE EFWSetDirection(int ID, bool bUnidirectional)
Purpose: Sets the rotation direction of the EFW filter wheel with the specified ID value.
Description:
int ID: The ID of the filter wheel.
bool bUnidirectional: If set to true, the filter wheel will always rotate in the same direction.

3.10 EFWGetDirection

Syntax: EFW_ERROR_CODE EFWGetDirection(int ID, bool *bUnidirectional)
Purpose: Retrieves the rotation direction of the EFW filter wheel with the specified ID value.

3.11 EFWCalibrate

Syntax: EFW_ERROR_CODE EFWCalibrate(int ID)
Purpose: Enters the EFW device into calibration mode.

3.12 EFWClose

Syntax: EFW_ERROR_CODE EFWClose(int ID)
Purpose: Closes the EFW device with the specified ID to release its resources. This should be the last function called.

3.13 EFWGetSDKVersion

Syntax: char* EFWGetSDKVersion()
Purpose: Retrieves the SDK version string.

3.14 EFWGetHWErrorCode

Syntax: EFW_ERROR_CODE EFWGetHWErrorCode(int ID, int *pErrCode)
Purpose: Retrieves the firmware error code for the EFW filter wheel with the specified ID value.

3.15 EFWGetFirmwareVersion

Syntax: EFW_ERROR_CODE EFWGetFirmwareVersion(int ID, unsigned char *major, unsigned char *minor, unsigned char *build)
Purpose: Retrieves the firmware version information for the EFW filter wheel with the specified ID value.

3.16 EFWGetSerialNumber

Syntax: EFW_ERROR_CODE EFWGetSerialNumber(int ID, EFW_SN* pSN)
Purpose: Retrieves the serial number of the EFW filter wheel with the specified ID value.
Description:
EFW_SN* pSN: Pointer to the serial number structure.

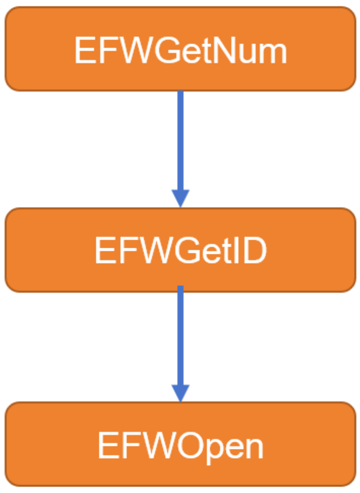
3.17 EFWSetID

Syntax: EFW_ERROR_CODE EFWSetID(int ID, EFW_ID alias)

Purpose: Sets an alias for the EFW device with the specified ID.

4 Recommended Call Sequence

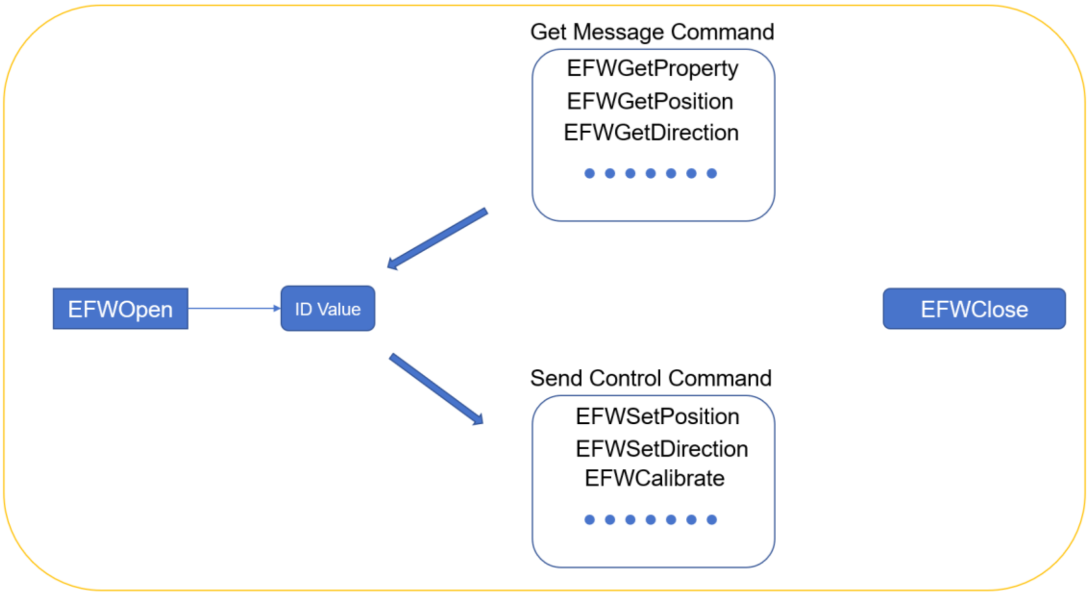
4.1 Establishing a connection with the EFW device



Connect the EFW device to a PC via USB. The USB driver on the PC will poll the EFW device and generate a device node on the computer based on the report descriptor.
Retrieve the number of EFW devices:
EFW_API   int EFWGetNum();
    We first call this interface, which is the first API to be invoked. It checks the number of connected EFW devices and returns the count of EFW devices.
Retrieve the device ID to be controlled
* EFW_API   EFW_ERROR_CODE EFWGetID(int index, int* ID);
    This function is used to obtain the ID value of the connected EFW device.
    Index: The EFW number obtained from EFWGetNum(). The input parameter is EFWGetNum() - 1. If you have connected multiple EFW devices, you can use a for loop to obtain the ID value for each EFW device. The ID value is crucial, as we will use it in nearly all APIs (USB communication mode).

Open Device:
* EFW_API   EFW_ERROR_CODE EFWOpen(int ID);
This open function initiates communication with the selected EFW device. Only after calling this function and receiving EFW_SUCCESS can the EFW device be controlled. Its ID parameter is the value obtained from the GetID function.
After completing the above steps, you can communicate with the EFW device using other command interfaces.

4.2 EFW Device Communication Interaction:



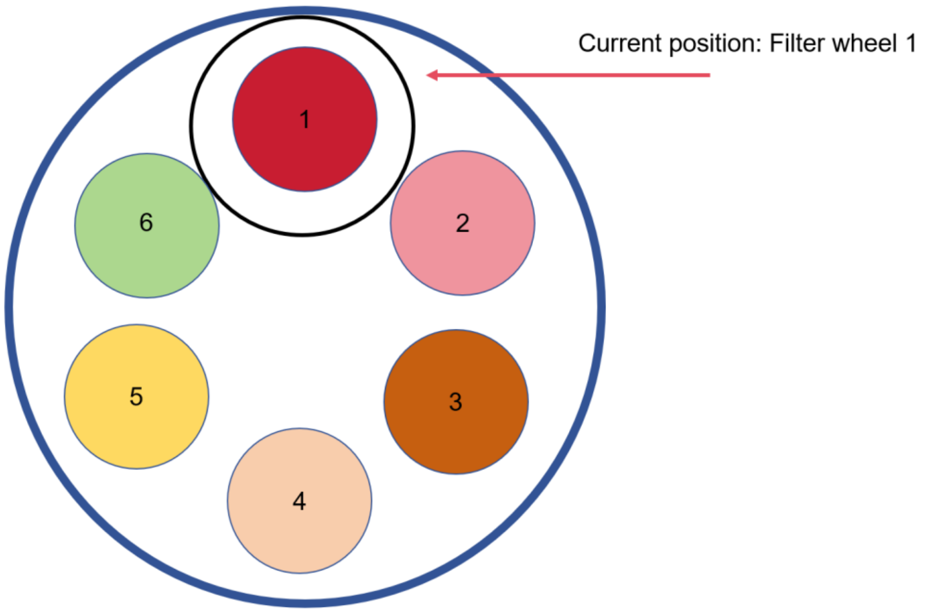
The EFW SDK supports the following interactive commands:

Acquire data		Control device	
Retrieve ID	EFWGetID	Set Movement Position	EFWSetPosition
Retrieve device data	EFWGetProperty	Set Rotation Direction	EFWSetDirection
Retrieve current location	EFWGetPosition	Position Verification	EFWCalibrate
Retrieve orientation	EFWGetDirection	Set ID Value	EFWSetID
Retrieve SDK version	EFWGetSDKVersion		
Retrieve firmware version	EFWGetFirmwareVersion		
Retrieve device serial number	EFWGetSerialNumber		

When performing rotation operations using the SetPosition command, EFW must first call the EFWGetProperty interface to obtain information such as the number of filter wheels before proceeding with control operations.

Obtaining position information:

EFW_API EFW_ERROR_CODE EFWGetPosition(int ID, int *pPosition);
This function is used to obtain the current position of the EFW filter wheel. The input parameter ID is the value obtained from the GetID function. The result will be written to the memory location pointed to by the passed pointer pPosition.



Control EFW:

Use the following functions to control the rotation and focusing of the EFW device.
EFW_API EFW_ERROR_CODE EFWSetPosition(int ID, int Position);
This function controls the operation of the EFW device. The input parameter ID is the ID value obtained from the GetID function. The input parameter Position specifies which filter to select. Note that the input Position target must not exceed the number of filter wheels.

4.3 Shut down the device

We no longer use the EFW device with this ID. To deactivate the EFW device, call the following function:
* EFW_API EFW_ERROR_CODE EFWClose(int ID);
This function deactivates the EFW device. The input parameter ID is the value obtained from the GetID function. After executing this command, the device with this ID can no longer be controlled via the API.

**For additional interfaces, refer to the EFW_filter.h header file. **

5 Special Note

If you encounter a situation where the SDK (in a Linux environment) detects the device but cannot open it, resolve it using the following method.
Solution:
The efw.rules file is located in the lib/ directory. Move this file to the /etc/udev/rules.d/ directory using the command:

```
sudo cp efw.rules /etc/udev/rules.d/
```

Reload the rules:

```
sudo udevadm control --reload-rules
```

Finally, unplug and replug the USB device to resolve the issue.

6 More Information

ZWO Official Website：https://www.zwoastro.cn/