



Camera Specific Options

This section outlines the different camera specific options available to the Apx60.

The following features can be set using the Camera Specific Options:

1. Exposure Speed
2. Gain / Offset
3. Fast Mode

Note: The Artemis{Get/Set}Gain functions will not work with the Apx60.

Camera Specific Options Functions

In order to check / update the settings, you will need the following functions (described below):

1. ArtemisHasCameraSpecificOption
2. ArtemisCameraSpecificOptionGetData
3. ArtemisCameraSpecificOptionSetData

All functions require a handle to the camera (which is returned from the ArtemisConnect method) as well as an ID which defines the option you are using. The IDs are as follows:

- ID_GOPresetMode = 1
- ID_GOPresetLow = 2
- ID_GOPresetMed = 3
- ID_GOPresetHigh = 4
- ID_GOCustomGain = 5
- ID_GOCustomOffset = 6
- ID_ExposureSpeed = 14
- ID_FX3Version = 200
- ID_FPGAVersion = 201

ArtemisHasCameraSpecificOption

```
bool ArtemisHasCameraSpecificOption(ArtemisHandle handle, unsigned short id);
```

This function is used to check whether the camera has the requested option.

The function will return false if:

1. The option isn't available
2. The camera is no longer available

ArtemisCameraSpecificOptionGetData

```
int ArtemisCameraSpecificOptionGetData(ArtemisHandle handle, unsigned short id, unsigned char * data, int dataLength, int * actualLength);
```

This function is used to get the current value of the given option. The function will return 'ARTEMIS_INVALID_PARAM' if the option isn't available on the camera or the 'dataLength' provided is too small. Otherwise it will return 'ARTEMIS_OK'.

To call this method you will need to provide a buffer via the 'data' parameter to store the result of the function call. You will also need to pass the length of the buffer in bytes via the 'dataLength' param to ensure a buffer overflow does not occur. Below you will find listed the buffer size required for each ID and the corresponding data that is returned.

These two settings require a 6 byte buffer:

- ID_GOCustomGain = 5
- ID_GOCustomOffset = 6

The buffer will be populated with 3 unsigned shorts:

Byte	0-1	2-3	4-5
Value	Minimum Value	Maximum Value	Current Value

These three settings require a 5 byte buffer:

- ID_GOPresetLow = 2
- ID_GOPresetMed = 3
- ID_GOPresetHigh = 4

The buffer will be populated with 1 byte and 2 unsigned shorts

Byte	0	1-2	3-4
Value	If the camera has this preset	Preset Gain Value	Preset Offset Value

This settings requires a 2 byte buffer :

- ID_GOPresetMode = 1

The buffer will be populated with 2 bytes:

Byte	0-1
Value	Current Gain/Offset mode setting. <ul style="list-style-type: none">• Custom = 0• Low = 1• Medium = 2• High = 3

This setting requires a 2 byte buffer:

- ID_ExposureSpeed = 14

Byte	0-1
Value	Current Exposure Speed. <ul style="list-style-type: none">• Power Save = 0• Normal = 1• Fast = 2

These settings require a 6 byte buffer:

- ID_FX3Version = 200
- ID_FPGAVersion = 201

The buffer will be populated with 3 unsigned shorts:

Byte	0-1	2-3	4-5
Value	Year	Month	Date concatenated w/ build number that day

```
unsigned char data[6];
int actualLength;
ArtemisCameraSpecificOptionGetData(handle, ID_FPGAVersion, data, 6, &actualLength);
unsigned short year;
unsigned short month;
unsigned short dayPatch;
unsigned short* ptr = (unsigned short*)data;
year = ptr[0];
month = ptr[1];
dayPatch = ptr[2];
```

Figure 1 Example of getting the FPGA firmware version

ArtemisCameraSpecificOptionSetData

```
int ArtemisCameraSpecificOptionSetData(ArtemisHandle handle, unsigned
short id, unsigned char * data, int dataLength);
```

This function is used to set the current value of a given option. The function will return 'ARTEMIS_INVALID_PARAM' if the option isn't available on the camera. Otherwise it will return 'ARTEMIS_OK'.

You will need to pass the value in via the 'data' param and the data length in bytes via the 'dataLength' param.

This setting requires 2 bytes(unsigned short) of data:

- ID_ExposureSpeed = 14

Byte	0-1
Value	Exposure speed to set <ul style="list-style-type: none">• Power Save = 0• Normal = 1• Fast = 2

```
unsigned char exposureSpeed[] = { 1, 0 };
ArtemisCameraSpecificOptionSetData(handle, ID_ExposureSpeed, exposureSpeed, 2);
```

Figure 2 Example of setting the exposure speed to Normal

This setting requires a two byte buffer:

- ID_GOPresetMode = 1

Byte	0-1
Value	Gain/Offset setting to set <ul style="list-style-type: none">• Custom = 0• Low = 1• Medium = 2• High = 3

These two settings require 2 bytes(unsigned short) of data:

- ID_GOCustomGain = 5
- ID_GOCustomOffset = 6

For these settings to take effect you must set ID_GOPresetMode to Custom(0). The argument passed should be between the minimum and maximum value returned by ArtemisCameraSpecificOptionGetData. If the argument passed is outside of the min/max value, it will be clamped to a valid one (See figure 3 for a partial example of how to set the gain and offset to a custom value).

```

// Set the camera into custom gain/offset mode
unsigned char gainOffsetPreset[] = { 0, 0 }; // custom
result = ArtemisCameraSpecificOptionSetData(handle, ID_GOPresetMode, gainOffsetPreset, 2);
// Error handling is left as an exercise to the user
if (result != ARTEMIS_OK)
    return;

// Get the bounds of the custom gain/offset
unsigned short minG, maxG, currentG;
unsigned short minO, maxO, currentO;
unsigned char gData[6];
unsigned char oData[6];
int actualLength;

result = ArtemisCameraSpecificOptionGetData(handle, ID_GOCustomGain, gData, 6, &actualLength);
result = ArtemisCameraSpecificOptionGetData(handle, ID_GOCustomOffset, oData, 6, &actualLength);

// A couple of examples of how to get the data back from the above buffers
unsigned short* ptr = (unsigned short*)gData;
minG = *ptr;
maxG = ptr[1];
currentG = ptr[2];

minO = oData[0] | (oData[1] << 8);
maxO = oData[2] | (oData[3] << 8);
currentO = oData[4] | (oData[5] << 8);

unsigned char customGain[] = { 4, 0 };
unsigned short customOffset = 1000;

// Check that the gain/offset we want to set are within the bounds of the available values
if (customGain[0] > maxG || customGain[0] < minG)
    return;
if (customOffset > maxO || customOffset < minO)
    return;

// Set the custom gain/offset
result = ArtemisCameraSpecificOptionSetData(handle, ID_GOCustomGain, customGain, 2);
result = ArtemisCameraSpecificOptionSetData(handle, ID_GOCustomOffset,
                                             (unsigned char*)&customOffset, 2);

// Error handling is left as an exercise to the user
if (result != ARTEMIS_OK)
    return;

```

Figure 3 Example of setting the custom gain/offset.

Fast Mode

When ID_ExposureSpeed is set to fast(2) the fast mode of the camera is activated. Images will not be returned in the usual manner via a call to ArtemisImageBuffer. You will need to provide a function pointer of type:

```
typedef void(*FastModeCallback)(ArtemisHandle handle, int x, int y, int w,
int h, int binx, int biny, void * imageBuffer);
```

to ArtemisSetFastCallback. The function you provide will be called every time an image is ready. **Your function should copy the image buffer into your application and return as quickly as possible as you will be blocking the download of the next image.**

To start capturing images in fast mode a call to ArtemisStartFastExposure must be made. Once this call is made changes to the exposure settings (exposure length, binning, sub frame, gain/offset) will

not update **as the camera locks its settings whilst in fast mode**. If you want to change the settings you must stop imaging by using ArtemisStopExposure change the settings you wish to and then call ArtemisStartFastExposure again.

Unused Functionality

As we use a C API which covers all of our cameras, many functions are not applicable to the Apx60. They are listed below:

- ArtemisSetSubSample
- All continuous exposure functionality
- ArtemisSetPreview
- ArtemisAutoAdjustBlackLevel
- ArtemisPrechargeMode
- ArtemisEightBitMode
- ArtemisGetEightBitMode
- ArtemisStartOverlappedExposure
- ArtemisOverlappedExposureValid
- ArtemisSetOverlappedExposureTime
- ArtemisGetProcessing
- ArtemisSetProcessing
- ArtemisClearVReg
- ArtemisGetAmplifierSwitched
- ArtemisSetAmplifierSwitched
- All column repair functionality
- All eeprom functionality
- ArtemisGetGain
- ArtemisSetGain
- All GPIO functionality
- All guiding functionality
- All lens functionality
- All shutter functionality