

The SDK (Software Developers Kit) provides a control method for the i-SPEED TR, 3, FS and DF monochrome and colour cameras. It allows the user to integrate C++ commands into their own software to have full control of all of the cameras functions and features.

Below shows the Sections within the SDK guide and camera features that can be controlled via 100+ commands

I-SPEED 3 SDK LIBRARY

DESCRIPTION

General Notes on Usage

RETURN VALUES

SETUP AND MONITORING FUNCTIONS

Initialize, Model, Serial number

Resolution

Speed

Shutter

Trigger

Sync mode

Manual economy resolution

Normal, ROC, BROCC buffer

STATUS FUNCTIONS

Trigger

Recording

Power Off

Record Sate

POST RECORDING FUNCTIONS

Number of frames

Buffer Information

Trigger Frame

IMAGE TRANSFER FUNCTIONS

Live Frames

Thumbnails

Multi Frames

ADDITIONAL CAMERA STATUS AND SETTINGS

Version, Battery Info, System Status

Permanent text

Optimise

White Balance

Dynamic range, Enhanced sensitivity

Configuration

iFocus

IRIG FUNCTIONS (FS CAMERA BUILD TYPES ONLY)

IRIG Signal State

IRIG Sync

IRIG recording

PIV FUNCTIONS (DF CAMERA BUILD TYPES ONLY)

PIV timing setting

BAYER CONVERSION FUNCTIONS

Bayer Gains

Convert Bayer Image

ERROR AND TIMEOUT HANDLING

Errors

Timeouts

EXAMPLES OF USAGE

Incorporating the SDK into a Project

Initializing a Camera and Starting a recording

Retrieving frames from a Previous Recording

Handling Images

Handling Bayer Images

Using the Multi frame Transfer function

Below is an example of the detail given per command

GETMODEL

*HRESULT GetModel (char * pcModel, short nLen)*

Get the model string of the camera associated with this instance of the SDK.

Parameters:

char * pcModel Character array which will receive the model string of the camera.

The array should be at least 64 characters in length.

short nLen Size of the buffer pointed to by **pcModel**

Below is an example of usage

RETRIEVING FRAMES FROM A PREVIOUS RECORDING

Below is a sample code fragment which queries the camera for information about the current recording stored in memory, and then retrieves a frame from the current recording. The code assumes that an instance of the **CISPEED3_SDK** class has been created with the variable name **m_iSPEED3**.

```
// get information about the last recording
// for readability, we have omitted error checking
IS3BufferInfo info;
m_iSPEED3.GetBufferInfo(&info);
// make sure there is a recording in memory
if ( info.IFrames > 0 )
{
    // allocate memory to receive the frame
    long lSize;
    IMAGE_BUFFER pFrame = AllocImageBuffer(&lSize);
    // get the first frame
    // frame numbers start from 0 up to the info.IFrames-1
    m_iSPEED3.GetFrame(0, &lSize, pFrame, CISPEED3_SDK::QUALITY_UNCOMPRESSED);
    // do processing on the frame or render the frame
    ...
    // free memory
```