Application Note About the ShutterBase Feature

Integration Time for IEEE 1394 Cameras

Basler IEEE 1394 cameras are controlled by a register set defined in the "IIDC 1394-based Digital Camera Specification". This specification includes a "Shutter Control and Status Register" (CSR) that is used to change the effective integration time the camera uses when capturing images. Twelve of the bits in this register define the raw value of the shutter. The effective integration (exposure) time of the camera is then calculated by multiplying the raw value with a shutter time base.

$$T_{\exp} = (raw\,value) \times (time\,base)$$

The factory default time base for most Basler 1394 cameras is 20 µs. So a raw value of '30' results in the camera capturing images with an effective integration time of 600 µs. The maximum integration time that can be reached this way is normally 4095 x 20 µs = 81.9 ms.

Shutter_Base Control Register

On cameras with certain sensors, the time base for the integration time can be changed. Control of the time base has been implemented as a Smart Feature (see the "Smart Features and the Smart Features Framework" chapter of your cameras user's manual for an introduction to Smart Features). The following table shows the layout of the Control and Status Register for this new smart feature.

Name	**Shutter_Base**		
Address	See "Determining the Address of Smart Features CSRs" in the User's Manual.		
CSR GUID	648be1da-a416-11d8-9b47-00105a5bae55		
Position	**Field**	**Bit**	**Description**
0	Presence_Inq (Read only)	[0]	Presence of this feature 0: Not Available 1: Available
	---	[1 … 30]	Reserved
	Enable (Read / write)	[31]	Enable / Disable this feature 0: Disable 1: Enable
4	Value Increment (Read only)	[0 … 31]	Increment by which the time base value can be adjusted (float value in seconds).
8	Value Minimum (Read only)	[0 …31]	Minimum value for the time base (float value in seconds).
12	Value Maximum (Read only)	[0 … 31]	Maximum value for the time base (float value in seconds).
16	Value (Read / write)	[0 … 31]	Value to use as time base (float value in seconds). If the feature is disabled, this field shows the factory default time base (usually 20 µs). If the feature is enabled, this value is used as the time base for calculating the integration time.

The first quadlet of the register contains information about the availability of the register. The shutter base feature can only be used if Bit 0 is set to '1'. Bit 31 can be used to enable / disable the feature. If the feature is disabled, the factory default time base (usually 20 µs) is used. If the feature is enabled, the value in the *Value* field of the register will be used as the time base.

The *Value Increment*, *Value Minimum* and *Value Maximum* fields are used to describe the range of possible values that can be used as a valid time base. All values are coded in IEEE/REAL*4 floating point format (which directly maps to the "float" datatype of C/C++). All values are absolute values, the unit is seconds. The *Minimum Value* field describes the minimum value that can be written into the *Value* field. If a smaller number is written, it will be automatically set to the minimum value. Similarly, if a value larger than the *Maximum Value* is written, it is automatically reduced to the maximum value. The value in the *Increment* field describes the granularity by which the *Value* field can be adjusted. Values written to the *Value* field will be automatically rounded so that they fit the granularity.

Changing the Time Base for Integration Time

To change the time base for integration time, the following steps must be executed:

1. Determine the base address of the CSR (as described in the User's Manual).
2. Enable the Shutter Base feature by setting Bit 31 of the first quadlet in the control register to '1'.
3. Readout the minimum and maximum values and the increment to determine the possible range of values for the time base.
4. Choose a time base to use in seconds (e.g., 100e-6 for 100 µs)
5. Write the value into the *Value* quadlet of the control register

The C++ code fragment on the next page uses the BCAM-API and SFF-Headers to adjust the time base and shutter raw value to achieve an effective integration time of 2 seconds. Note that the user should always make sure the Shutter_Base CSR address can be successfully retreived and that the presence flag is set to '1'. Since the register access functions of the BCAM-API (ReadRegister / WriteRegister) only allow access to the register with QUADLET datatypes, the float values are cast to their QUADLET representation by a reinterpret_cast.

[Shutter_Base_Example.cpp]

```
#include <math.h>
#include <initguid.h>
#include "bcamextension.h"

// Define the GUID for the time base feature
DEFINE_GUID(GUID_SF_ADV_SHUTTER_BASE,
            0x648be1da,0xa416,0x11d8,0x9b,0x47,0x00,0x10,0x5a,0x5b,0xae,0x55);

int main() {

  // Open first camera on the IEEE 1394 bus
  CBcamExtension Bcam;
  Bcam.Open(*CBcam::DeviceNames().begin());

  // Get the base address and check for presence
  LONGLONG Shutter_Base = Bcam.GetSmartFeatureAddress(GUID_SF_ADV_SHUTTER_BASE);
  ASSERT(Shutter_Base != 0);
  QUADLET q = Bcam.ReadRegister(Shutter_Base);
  ASSERT(q & 0x80000000);

  // Enable the feature
  Bcam.WriteRegister(Shutter_Base, 0x80000001);

  // Inquire the value range
  q = Bcam.ReadRegister(Shutter_Base+4); // Read the increment
  float val_inc = *reinterpret_cast<float *>(&q); // Cast to a float value
  q = Bcam.ReadRegister(Shutter_Base+8); // Read the minimum
  float val_min = *reinterpret_cast<float *>(&q); // Cast to a float value
  q = Bcam.ReadRegister(Shutter_Base+12); // Read the maximum
  float val_max = *reinterpret_cast<float *>(&q); // Cast to a float value

  // Set the time base value to 1 microsecond
  float val = 0.001f; // 1 microsecond
  ASSERT(val > val_min && val < val_max);
  q = *reinterpret_cast<QUADLET *>(&val); // Cast to a quadlet value
  Bcam.WriteRegister(Shutter_Base+16, q); // Write value to register

  // Read back to see if the camera rounded the value
  q = Bcam.ReadRegister(Shutter_Base+16); // Read back value
  float val_effective = *reinterpret_cast<float *>(&q); // Cast to a float value

  // Set the shutter raw so that the resulting integration time will be 2 seconds
  unsigned long shutter_raw =
      static_cast<unsigned long>(floor(2.0f / val_effective + 0.5f));
  ASSERT(shutter_raw > Bcam.Shutter.Raw.Min() &&
         shutter_raw < Bcam.Shutter.Raw.Max());
  Bcam.Shutter.Raw = shutter_raw;

  // Close Bcam device and exit
  Bcam.Close();
  return 0;
}
```