MIGHTEX
Simply Brighter

(In Canada)
111 Railside Road
Suite 201
Toronto, ON M3A 1B2
CANADA
Tel: 1-416-840 4991
Fax: 1-416-840 6541

(In US)
1241 Quarry Lane
Suite 105
Pleasanton, CA 94566
USA
Tel: 1-925-218 1885
Email: sales@mightex.com

Mightex S-Series USB Camera USB Protocol

Version 1.0.1

Jan. 2, 2019

Relevant Products

Part Numbers
SCN-BG04-U, SCE-BG04-U, SCN-CG04-U, SCE-CG04-U, SCN-B013-U, SCE-B013-U, SCN-C013-U, SCE-C013-U, SCN-C030-U, SCE-C030-U

Revision History

Revision	**Date**	**Author**	**Description**
1.0.0	Aug. 12, 2010	JT Zheng	Initial Revision
1.0.1	Jan. 2,2019	JT Zheng	New Mightex Logo

Mightex USB 2.0 SCX camera is designed for low cost machine vision and other applications, With USB 2.0 high speed interface and powerful PC camera engine, the camera delivers CMOS image at high frame rate. The GUI demonstration application and SDK are provided for user's Windows application developments, for users who want to use the camera in Non-Windows based system, the USB2.0 based command protocol is described below.

Mightex USB 2.0 S Series camera is using Cypress CY68013A as its on board USB device controller, please refer to the Cypress documents for the details of the chip's USB protocol supporting.

Descriptors

Device Descriptor:

bcdUSB:	0x0200
bDeviceClass:	0x00
bDeviceSubClass:	0x00
bDeviceProtocol:	0x00
bMaxPacketSize0:	0x40 (64)
idVendor:	0x04B4 (For evaluation samples)
idProduct:	0x0228
bcdDevice:	0x0000
iManufacturer:	0x01
	0x0409: "Mightex"
iProduct:	0x02
	0x0409: "USB-CLASSIC-2"
iSerialNumber:	0x00
bNumConfigurations:	0x01

Configuration Descriptor:

wTotalLength:	0x0027
bNumInterfaces:	0x01
bConfigurationValue:	0x01
iConfiguration:	0x00
bmAttributes:	0x80 (Bus Powered)
MaxPower:	0x96 (300 mA)

Interface Descriptor:

bInterfaceNumber:	0x00
bAlternateSetting:	0x00
bNumEndpoints:	0x04
bInterfaceClass:	0xFF
bInterfaceSubClass:	0x00
bInterfaceProtocol:	0x00
iInterface:	0x00

Endpoint Descriptor:

bEndpointAddress:	0x01 **OUT**
Transfer Type:	Bulk
wMaxPacketSize:	0x0200 (512)
bInterval:	0x00
bEndpointAddress:	0x81 **IN**
Transfer Type:	Bulk
wMaxPacketSize:	0x0200 (512)
bInterval:	0x00
bEndpointAddress:	0x82 **IN**
Transfer Type:	Bulk
wMaxPacketSize:	0x0200 (512)
bInterval:	0x00

bEndpointAddress:	0x86 **IN**
Transfer Type:	Bulk
wMaxPacketSize:	0x0200 (512)
bInterval:	0x00

IMPORTANT:

1). End point 0 is NOT listed above, as it's used for standard USB enumeration and configuration, for details of EP0 protocols, please refer to the latest USB specification and Cypress CY68013A manual.

2). End point 1 (for IN and OUT) , 2 and 6 (for IN only) are used for camera specific commands, this documents is focused on the data communications occurred on these End points.

3). For SCX camera, as there's **NO** frame buffer on camera, it's very important for the USB Host to have proper resources (CPU cycles and Memory) to fetch the image data from the camera (via USB2.0)on time, the resources involved might be related to MCU, RAM and USB Host controller on user's embedded system. The image quality will be affected if the resources are limited and data are not fetched from camera on time. There're two USB commands to allow user to customize the timing of sensor's data outputting, "***Set Sensor Main Clock Frequency***" and "***Set Sensor HBlanking***" which are described below.

The protocol is extremely simplified for quick understanding and development.

End points 1 (IN & OUT)

End Point 1(OUT) and 0x81(IN) are used for command control, it's always the Host initiates a command to Camera on EP1(OUT), and according to the command, Camera may prepare Response in its buffer and waiting for Host to fetch. The command and response has the following generic format:

EP(0x01)	→	CommandID	Length	Data
EP(0x81)	←	Result	Length	Data

For **Command**:
CommandID : This is a ONE byte field, represent the command itself, it tells device what to do.
Length: The length of data in byte.
Data: Depends on the command ID.

For **Response**:
Result: 0x01 – Means OK, 0x00 – means Error
Length: The length of response data in byte. (When Result is OK)
Data: Depends on the command ID.
Note only some commands will have Response returned back from camera.

Note that in the following command description, the "→" means data flow on EP1 OUT, "←" means data flow on EP1 IN.

****. Query Firmware Version (CommandID = 0x01)***

→ 0x01	1	0x01
← 0x01	3	Major Minor Revision

Host can use this command to query version of the firmware from device, device will response the current firmware version.

****. Query Device Information (CommandID = 0x21)***

→0x21	1	0x00
←0x01	sizeof(tDeviceInfo)	DeviceInfo

For this command, the data field (0x00) is not used. Upon receiving this command, device will prepare device information and put in the EP1(IN) buffer, for host to fetch. The tDeviceInfo has the following definition:

```
#define STRING_LENGTH   14
typedef struct
{
    BYTE    ConfigRevision;
    BYTE    ModuleNo[STRING_LENGTH];
    BYTE    SerialNo[STRING_LENGTH];
    BYTE    ManuafactureDate[STRING_LENGTH];
} tDeviceInfo;
```

it's actually a 43 bytes data structure which contains the ModuleNo[] and SerialNo[], these two fields are important if user wants to support multiple devices in the host software, the serialNo[] might be the only information for user to distinct the devices of the same type. For ModuleNo[], it contains the Module type (B013, C030…etc.) information, while there're different types of cameras connected.

. User Eeprom Write (commandID = 0x25)

→ 0x25	34	Addr_MSB Addr_LSB Data…(32 bytes)
← 0x01/0x00	1	0

There's 1K eeprom space reserved for user's data, it's from 0x3C00 to 0x3FFF, user might use this command to write his own data (e.g. calibration data) into this area. It will return 0x01 if the write operation successes…otherwise it will return 0x00.
Note: When host sends more than one command 0x25 to camera, host should wait at least 5ms (10ms recommended) between them. And the camera expects the Address is always aligned with 32byte boundary.

. User Eeprom Read (commandID = 0x26)

→ 0x26	3	Addr_MSB Addr_LSB Size
← 0x01	Size	Data….

User can read the data back from the user Eeprom area, the Size has to be between 1 to 32, and if the reading operation successes, it will return the data….otherwise it returns 0x00.

. Camera Work Mode Setting (commandID = 0x30)

→ 0x30	1	MODE (0 or 1)
		#define NORMAL_MODE 0x00
		#define TRIGGER_MODE 0x01

This command is used for setting the current mode of the device, there's no response for this command, for the details of NORMAL and TRIGGER mode, please refer to USB Camera's user manual and SDK manual.

. Set Sensor Main Clock Frequency (commandID = 0x32)

→ 0x32	1	0/1/2

By default, the camera is set to maximum Sensor clock (Fast Clock for B013/C030 camera, and Normal clock for BG04/CG04 camera) to have max frame rate, however, for some embedded systems with limited resources, user might set slower clock to the sensor.

0 – Slow Clock (Not recommended)
1 – Normal Clock
2 – Fast Clock

Note: After sending this command to camera, host should wait for enough time (>200ms) for the next USB command (any command via EP1/2/6), that allows the camera to reset the sensor and do initializations. For BG04/CG04 cameras, when it's set to "Fast Clock", its frame rate can go ~65fps, with degraded SNR.

. Get current frame property (commandID = 0x33)

→0x33	1	0x00
←0x01	18	RowSize(2 bytes) ColumnSize(2 bytes) Bin ExposureTime(2 bytes) RGain GGain BGain XStart(2 bytes) YStart(2 bytes) **FrameInvalid** Reserved TimeStamp(2 bytes)

After host gets a frame from camera (command 0x34 and get frame data via EP2/6), user should immediately get the frame property of the last frame with this command. Thus the camera expects the following command sequence for each frame:

```
→ 0x35 1  0
  ← 0x01 6 TriggerState RowSize(2 bytes) ColumnSize (2 bytes) Bin
→ 0x34  1  1
  ← Frame Data to EP2/6
→ 0x33  1  0
       ← 0x01 18 Frame Property…
```

The returned frame property includes all the parameters the camera used for the last fetched frame. Note that for all parameters of 2 bytes, it's always the MSB and LSB order.
An Example: For a camera setting of 1280x1024, 1:2 decimation mode, Exposure Time = 10ms, XStart = YStart = 0, RGain = GGain = BGain = 12, we have:

← 0x01 18 05 00 04 00 01 00 0xC8 0C 0C 0C 00 00 00 00 **00** xx 12 34

Here, the RowSize is 0x0500, ColumnSize is 0x0400, Bin is 1 (means "1:2" mode, "0" means normal mode), Exposure time is 0x00C8 (200, as the Exposure Time is in "50us" unit, 200x50us = 10000us = 10ms). R, G, B Gains are all 12, XStart = YStart = 0, FrameInvalid is "0", "xx" is reserved (xx means "reserved"). TimeStamp is a number between 0 – 65535, it's from an internal timer of the camera, in "ms" unit.
Note: The FrameInvalid flag might be set to "1" for BG04/CG04 cameras (for B013/C030 camera, it's always "0"), this is sourced from the sensor itself of the BG04/CG04 cameras, when host gets a frame with this flag set to "1", it's **IMPORTANT** for host to immediately grab a new frame as following:

```
→ 0x34  1  1
  ← Frame Data to EP2/6 (A new frame is grabbed)
→ 0x33  1  0
       ← 0x01 18 Frame Property…
```

There must be **no** other commands (e.g. camera parameters) are allowed during, after getting a frame, host should check the FrameInvalid flag again…and if it's still set ("1"), host should do the above frame grabbing again until get a valid frame.

. Get image data (commandID = 0x34)

→0x34	1	1

This is a very important command, Host send this command on EP1(OUT), and there's no response on EP1(IN), However, **ONE frame data** will be ready on EP2(IN) and EP6(IN) for host to fetch. So Host should prepare proper USB buffer and fetch from EP2/EP6(IN) after this command is sent.

For the details of Frame Data on EP2/EP6, please refer to the following "End Point2(IN)/End Point6(IN)".
Note: In the device driver, it's recommended that user should prepare the USB receiving on EP2/EP6 (e.g. sending USB IRQ to USB Bus driver) prior to sending command 0x34 to camera.
Important: When host sends this command to camera, the camera will start the grabbing of one frame, the camera hardware will clock the sensor (the clock can be set by command 0x32) to output image data, if the host hardware is with limited resource, host might use command 0x36 to extend the HBlanking of each row OR slow down the sensor clock with command 0x32.

. Get camera trigger state (commandID = 0x35)

→0x35 1 1

← 0x01 6 TriggerState RowSize(2 bytes) ColumnSize(2 bytes) Bin

This command should be used before the command 0x34 (getting image data), while the camera is in **NORMAL** mode, the TriggerState can be ignored by host, but host must check the returned RowSize, ColumnSize and Bin, make sure they're the same (correct parameters) as the parameters host sets to the camera, otherwise host should not use command 0x34 to get image data. So host should use this command to make sure the returned RowSize/ColumnSize/Bin are matched with the resolution setting by host.

In **TRIGGER** mode, host should additionally check the TriggerState, only if it's "1", user can use command 0x34 to get one frame data.

Note: For BG04/CG04, if the last frame is invalid (FrameInvalid flag is set), host should direct use command 0x34 (and following 0x33 for property) to get a frame, there's no need to use this command (0x35) to check the parameters or TriggerState.

. Set Sensor HBlanking (commandID = 0x36)

→ 0x36 1 0/1/2

In addition to the above "Set Sensor Main Clock Frequency", another command to allow user to Change the sensor output timing is this "Set Sensor HBlanking", by default, the camera is set to short HBlanking to have maximum frame rate, however, for embedded system with limited resources, user might want to set longer HBlanking to slow down the output of the image data.

0 – Short HBlanking Time

1 – Long HBlanking Time

2 – Longest HBlanking Time

There's no need to have extra wait (see the >200ms above "Set Sensor Main Clock Frequency" command) for this command.

. GPIO Chip Register Write (commandID = 0x40)

→ 0x40 2 Register Value

. GPIO Chip Register Read (commandID = 0x41)

→0x41	1	Register
←0x01	1	Value

The device has a on board Philips PCA9536 chip which provide 4 GPIO pins, for user wants to use it, user can use the above two commands for writing and reading register values. For details of the register and their definitions, please refer to the specification of PCA9536.

. Set Camera Resolution (commandID = 0x60)

→ 0x60 7 RowSize(MSB,LSB) ColumnSize(MSB,LSB) BinMode

Host can set ROI (Region Of Interesting), 1:2 Desimation mode

Note:

1). RowSize and ColumnSize should be set to camera according to the camera type, e.g. for 3M camera it should be less than 2048x1536, for 1.3M camera, should be less than 1280x1024 should be set. Note that the Row Size and Column Size should be the multiple of 4, otherwise the firmware will correct it to the 4 boundary, e.g. while user sets row size to 1025, it actually is 1024. The minimum of row size is 32, and the minimum of column size is 4.

2). BinMode can be 0 and 1, 0 means "None decimation", 1 means "1:2 decimation".

For example, set camera to 1024x768, 1:2 decimation, we have:

→ 0x60 7 4 0 3 0 1
Here, (4 0) means 0x400, which is 1024 decimal, (3 0) means 0x300, which is 768. 1 means 1:2 mode.

. Set Camera (X,Y) Start Points (commandID = 0x61)

→ 0x61 4 XStartMSB XStartLSB YStartMSB YStartLSB

Host can set Start position of the ROI at any resolution other than the maximum one, On the other hand, there's a valid range for Xstart and Ystart at a certain ROI, the device will evaluate the Xstart and Ystart value set by this command, and if they're out of the valid range, device check and set proper value.

. Set Camera Gains (commandID = 0x62)

→ 0x62 3 Rgain Ggain Bgain

Host can set Gains for Red, Green and Blue pixels on sensor, there's no response for it. The Gain value should be in 1 – 64 (inclusive) for C030/B013 cameras, and 8 – 32 for BG04/CG04 cameras. Device will check the validity of the input gain value and set proper gain if the data in command is out of the range. (E.g. if gain value is more than 64, device will set the gain to 64)
Note: Gain value is the analog gain multiples of sensor's internal amplifier, it's from 0.25x –8x. For C030/B013 cameras, the actual multiple is (GainValue /8). For BG04/CG04 cameras, it's 1x – 4x.

. Set Camera Exposure Time (commandID = 0x63)

→ 0x63 2 MSB LSB (of Exposure Time)

Host can set Exposure Time of the device with this command, there's no response for it. The two bytes Data are the MSB and LSB of the new exposure time (16 bit word), note that the unit of the exposure time is 0.05ms, for example, if setting exposure time to 5ms, the value will be 100 (100x0.05ms = 5ms), so the command is:

→0x63 2 0x00 0x64

Note: the Exposure time range is 0.05ms – 750ms (1 – 15000 for the set value)

. Soft trigger command (commandID = 0x65)

→ 0x65 1 1

Host can simulate an external trigger assertion when the camera is in TRIGGER mode, after sending this command (Soft Trigger) to camera, host might use the command 0x35 to camera, the returned TriggerState must be "1" in this case, thus host can get ONE frame data back (with command 0x34) from the camera.

End point 2/End point 6 (IN)

End point2 and End point6 are used for fetching image data only, before reading data from it, please make sure Command 0x34 is sent successfully (please refer to the command 0x34 description, and before command 0x34, command 0x35 is usually to be issued).
When proper buffer is arranged and command 0x34 is sent, host can read data from EP2/EP6(IN), device will return the data in the following format for each frame on each End Point (EP2/EP6):

```
Typedef struct
{
tUINT8 PixelData[RowNumber/2][ColumnNumber];
} tImageFrame;
```

Basically, Row 0, 2, 4, …RowNumber-2 will be returned from EP2, Row 1, 3, 5, …., RowNumber -1 will be returned from EP6. Thus Host has to combine the data from EP2 and EP6 to have a full frame data.
For example, if the resolution is 1280x1024, no decimation, there're 1024 rows as following:

```
Row 0 …………………………………………….   ( On EP2)
Row 1 …………………………………………….   ( On EP6)
Row 2…………………………………………….   ( On EP2)
Row 3…………………………………………….   ( On EP6)
……
……
Row 1022 …………………………………………….   ( On EP2)
Row 1023 …………………………………………….   ( On EP6)
```

Host can get (1024*1280/2) = 655360 bytes from EP2 and EP6.

The Row and Column is depending on the resolution (and decimation value). Please refer to the resolution command for Row and Column value for each resolution index setting, note that if "1:2 decimation" is enabled for a resolution, the Row and Column value is only half of the standard value.

Note:
For returned frame data, it's CMOS type dependant, for Monochrome sensor, they're ADC value for each pixel. For Color sensor, note that there's a bayer filter on sensor and the format for rows are:
1st Row → G R G R G R….
2nd Row → B G B G B G….
….
For details, user might refer to sensor's specification.