CPiA Command Usage for QX3 Microscope

Rev: 1.4
Author: S. Gillies
Date: 01-03-00

1. Introduction

In order to support the QX3 microscope, a CPIA USB camera driver must provide an interface to the controls listed below. Firmware version 1.33 is required.

2. Controls List

This section lists the primary microscope controls and what camera settings are required for each. Most of these controls map directly to a CPIA command. Sample code is given in the next section for those which do not have a direct CPIA command implementation.

Startup

SetCompression(0, 0)	disable compression + disable decimation
SetBacklit: Enable(0)	don't use backlit exposure
SetFlicker: FlickerMode(0)	disable flicker control
SetExposure: ExpMode(2), Coarse(3), Fine(40), Gain(0)	enable AEC
SetExposure: ExpMode(1), Coarse(3), Fine(40), Gain(0)	set AEC values
ButtonStateSet: Button State (0)	enable button detect
SetSensorFPS(7.5 FPS)	use 7.5fps sensor speed
SetColourParams: Brightness(50) Contrast(60) Saturation(60)	

Brighness slider

SetColourParams: Brightness(0 - 100) Contrast(60) Saturation(60)

Lamps

Top or bottom lamp on:

SetGPIO	illuminate lamp
SetColourBalance: Mode(3), Red(0), Green(0), Blue(0)	disable ACB
SetColourBalance: Mode(1), Red(51), Green(47), Blue(3)	lamp specific colour balance
SetSensorMatrix: (0x61, 0xED, 0xF1, 0xEA, 0x5F, 0xF6, 0xE0, 0xE0, 0x7F)	lamp specific matrix
SetBaseComps (0xDC, 0xF3, 0xF3, 0x65)	these must be clipped in the range 210 - 240 before downloading if the banding filter is enabled
SetGainCeiling (0)	max gain = 1

Both lamps off:

SetGPIO	un-illuminate lamp(s)
SetColourBalance: Mode(3), Red(0), Green(0), Blue(0)	disable ACB
SetColourBalance: Mode(1), Red(20), Green(5), Blue(7)	normal colour balance
SetSensorMatrix: (0x60, 0xE2, 0xFE, 0xF7, 0x59, 0xF0, 0xFC, 0xE4, 0x60)	normal matrix
SetBaseComps (0xDC, 0xD6, 0xD6, 0xE6)	these must be clipped in the range 210 - 240 before downloading if the banding filter is enabled
SetGainCeiling (1)	max gain = 2

Cradle Detect

GetGPIO pins state	detect whether the unit is in the cradle or not

Push Button

GetGPIO pins	detect whether the button is held down or not – create a movie
GetPushButtonState	detects whether the button has been pushed or not – take a snapshot

Implementation Notes

- In order to detect the push button and cradle state with low latency, the GPIO pins are polled at a high frequency. As the Isochronous channel must be disabled for several milliseconds each time a READ command is issued, the driver should read the state of the GPIO between frames to avoid adversely affecting the framerate. The GPIO state returned to the application should be the value last read unless it was read more than 400ms ago, in which case a new value should be read from the camera.

3. Sample Source Code

The code listed below demonstrates how to use the CPIA command set to implement the QX3 features which cannot be implemented via simple CPIA commands, i.e.

- snapshot button detect
- cradle detect
- lamp control
- set colour balance matrix

```
//
//      camera.c
//
//      This file contains example code which could be used as a basis for
//      implementing the additonal driver functions required by the QX3 microscope
//

#define GPIO_TIMEOUT 400

static BOOL     interFrameGetGPIO       = FALSE;
static DWORD    lastTimeGotGPIO         = 0;
static BOOL     buttonCheck             = TRUE;

//
//      InterFrameGPIO
//
//      Use this function between frame uploads for rapid checking of the state of the GPIO
//      ports and push button.
//
//
void InterFrameGPIOCheck(       BOOL *pButtonIsPushed           ,
                                BOOL *pButtonHasBeenPushed      )
{
        BYTE    port0   ,
                port1   ,
                port2   ,
                port3   ;

        if ( buttonCheck || interFrameGetGPIO )
        {
                // read data from the camera
                CPIA_ReadMCPorts(       &port0  ,
                                        &port1  ,
                                        &port2  ,
                                        &port3  ) );
```

```
                if ( buttonCheck )
                {
                        if ( ( port1 & 0x02 ) ? FALSE : TRUE )
                        {

                                // button has been pressed since the last time we checked (and
                                // may still be pressed)
                                *pButtonIsPushed        = TRUE;
                                *pButtonHasBeenPushed   = TRUE;

                                // reset push button latch
                                CPIA_WriteMCPort( 3, 0xDF, 0xDF );
                                CPIA_WriteMCPort( 3, 0xFF, 0xFF );
                        }
                        else
                        {
                                // button has not been pressed since the last time we checked
                                *pButtonIsPushed                = FALSE;
                        }
                }


                // note the time the GPIO ports were read
                lastTimeGotGPIO = GetTime();
        }
}

//
//      GetGPIO
//
//      Reads the current state of the input pins.
//      The state is only read from the camera if the inter frame get GPIO check has
//      not been made or the GPIO state was read more than GPIO_TIMEOUTms ago
//
//      QX3 GPIO uses:
//              port2 bit6 = Cradle Detect. A '0' indicates that the microscope is in the cradle.
//

void GetGPIO( DWORD *pGpio )
{
        BOOL    result  = TRUE;
        BYTE    port0   ,
                port1   ,
                port2   ,
                port3   ;

        if (    !interFrameGetGPIO ||
                (GetTime( ) > (lastTimeGotGPIO + GPIO_TIMEOUT) ) )
        {
                // read data from the camera
                CPIA_ReadMCPorts(       &port0,
                                        &port1,
                                        &port2,
                                        &port3  ) );

                interFrameGetGPIO = TRUE;

                lastTimeGotGPIO = GetTime( );
        }
```

```
        *pGpio =        port0           |
                        (port1 << 8)    |
                        (port2 << 16)   |
                        (port3 << 24);

}

//
//      SetGPIO
//
//      Writes to available output pins
//
//      QX3 GPIO uses:
//              port2 bit3 = lamp                       Set to '0' to illuminate the lamp.
//              port2 bit1 = lamp                       Set to '0' to illuminate the lamp.
//
void SetGPIO( BYTE gpio )
{
        BYTE port;

        // take port out of passthrough mode
        CPIA_WriteVCReg( 0x90, 0x8F, 0x50);

        // Mask port 2 and write 1s for all the inputs, i.e. 111xxxxx
        port = gpio & 0x1F;
        port = port | 0xE0;

        // Write port value
        CPIA_WriteMCPort( 2, 0, (BYTE)port);
}


//
//      SetMatrix
//
//      Download an updated colour balance matrix
//
//      Notes:
//              The matrix currently being used by the camera cannot be uploaded
//
void SetMatrix(   DWORD Row1   ,
                  DWORD Row2   ,
                  DWORD Row3 )
{
        char matrix[9];

        matrix[0] = (Row1 & 0x00FF0000) >> 16;
        matrix[1] = (Row1 & 0x0000FF00) >> 8;
        matrix[2] = (Row1 & 0x000000FF);
        matrix[3] = (Row2 & 0x00FF0000) >> 16;
        matrix[4] = (Row2 & 0x0000FF00) >> 8;
        matrix[5] = (Row2 & 0x000000FF);
        matrix[6] = (Row3 & 0x00FF0000) >> 16;
        matrix[7] = (Row3 & 0x0000FF00) >> 8;
        matrix[8] = (Row3 & 0x000000FF);

        CPIA_SetSensorMatrix( matrix );

        return result;
}
```