Software Developer's Guide for CPiA Cameras

Release 1.0

2nd November 1998

1 Introduction

1.1 Overview

This document provides information on developing non Wintel software for CPiA based cameras. The documents is broken into two main sections. First a summary of the CPiA command set and then some notes of how to use the commands. It is recommended that this document is read in conjunction with the CPiA datasheet.

2 CPIA Command Set

2.1 Introduction

This document details the CPiA Command Set which is handled by VISION USB, VISION DUAL and VISION PPC2 Cameras. For cameras using the USB interface (VISION USB and VISION DUAL) the CPiA Command Set forms the Vendor Specific Command Set. These cameras will also respond to Standard USB Commands compliant with the USB Specification V1.00.

Each CPiA command is passed to the camera in the *bRequest* field of the 8 byte command packet (which adopts the USB SETUP packet definition). The three most significant bits of this byte field are used to identify the module which will ultimately process the command. The five least significant bits are used to identify the specific command.

2.2 CPiA Module Commands

The three most significant bits of *bRequest* are set to 000 to specify a CPiA Module Command. CPIA module commands are used primarily during the system initialisation and power up/down sequences. In addition this module provides the commands to retrieve the global status of the camera and check that commands have executed correctly without errors.

2.2.1 GetCPIAVersion

The host uses this command to request the current CPiA version. It returns the firmware version number (2 bytes) followed by the version of the Video Compressor (VC) device (2 bytes). The first byte returned specifies the main firmware version where 0 indicates pre-production firmware, 1 indicates the first production or release firmware, 2 indicates a significant release firmware change (ie. not a bug fix). The second byte returned specifies the revision within the main firmware version. Therefore, if the first byte returned was 0 and the second was 8, this would signify overall firmware version 0.8. The third byte specifies the main VC hardware version. The fourth byte specifies the revision within the main VC hardware version. Therefore, if the third byte returned was 1 and the fourth was 2, this would signify overall VC hardware version 1.

This is a READ or device to host transfer where one DATA IN transaction follows the SETUP transaction.

bmRequestType	**bRequest**	**wValue**		**wIndex**		**wLength**	
11000000	000 00001	0x00	0x00	0x00	0x00	0x04	0x00

Table 2.1

DATA IN byte	Data
1	Main firmware version
2	Revision within the main firmware version
3	Main Video Compressor (VC) hardware version
4	Revision within the main VC hardware version

Table 2.2

2.2.2 GetPnPID

The host uses this command to request the device Plug and Play (PnP) Device Version Information. The first byte pair returned represent the Vendor ID, or hex number associated with the device manufacturer (0x0553 for USB). The second byte pair returned represent the Product ID which is a hex number assigned by the manufacturer to represent the hardware device. The third byte pair returned represent the Device Revision ID which is a Binary Coded Decimal number assigned by the manufacturer to represent the revision of the hardware device

This is a READ or device to host transfer with one DATA IN transaction following the SETUP transaction.

bmRequestType	bRequest	wValue		wIndex		wLength	
11000000	000 00010	0x00	0x00	0x00	0x00	0x06	0x00

Table 2.3

DATA IN byte	Data
1	Vendor ID (least significant byte)
2	Vendor ID (most significant byte)
3	Product ID (least significant byte)
4	Product ID (most significant byte)
5	Device Revision ID (least significant byte)
6	Device Revision ID (most significant byte)

Table 2.4

2.2.3 GetCameraStatus

NOTE - not all fields of the camera status are currently updated by the firmware correctly.
The host uses this command to request the current status of the device. This is a READ or device to host

transfer with one DATA IN transaction following the SETUP transaction.

bmRequestType	bRequest	wValue		wIndex		wLength	
11000000	000 00011	0x00	0x00	0x00	0x00	0x08	0x00

Table 2.5

DATA IN byte	Data
1	System State
2	Grab State
3	Stream State
4	Fatal Error
5	Cmd Error
6	Debug Flags
7	VPStatus
8	ErrorCode

Table 2.6

2.2.3.1 SystemState

This byte shows the overall system state of the unit. eg UNITIALISED_STATE, LO_POWER_STATE, HI_POWER_STATE.

2.2.3.2 GrabState

NOTE - not yet implemented.

Shows the current state of the frame grabbing process. eg GRAB_IDLE, GRAB_WAITING, GRAB_ACTIVE, GRAB_FINISHED.

2.2.3.3 StreamState

This shows the state of the stream process. eg STREAM_NOT_READY, STREAM_READY, STREAM_OPEN, STREAM_PAUSED, STREAM_FINISHING.

2.2.3.4 FatalError

This byte contains a flag bit for each module that will be set if the module encounters an error that prevents it from processing any further commands reliably. If this state is ever reached it is likely that a hardware error has been detected.

The table below shows which bit is used for each module.

Module Name	Flag bit
CPIA	00000001

Table 2.7

Module Name	Flag bit
SYSTEM	00000010
INT_CTRL	00000100
PROCESS	00001000
USB_COM or PP_COM	00010000
VP_CTRL	00100000
CAPTURE	01000000
DEBUG	10000000

Table 2.7

2.2.3.5 CmdError

This byte contains a flag bit for each module that will be set if a command is not recognised or fails to execute correctly. In the case of a parallel port interface between host and camera, checking this byte is the only way the host can find out if a command it submitted to the camera actually executed sucessfully or not. In the case of a USB interface the host can determine if the command has failed to execute successfully by means of a STALL handshake packet on subsequent data phases and/or the status phase.

Definition of module bits is as the FatalError field above.

2.2.3.6 DebugFlag

This byte contains flag bits for each module that control whether that module will output debug information on the RS232 port as it executes.

Definition of module bits is as the FatalError field above.

2.2.3.7 VPStatus

This byte is used by the VP_CTRL module to report the status of the VP or commands relating to it to the host.

2.2.3.8 ErrorCode

If a module detects an error it may write a byte in this field to inform the host of what type of error has occured. (Exposure status documentation in here!!!!!).

2.2.4 GotoHiPower

The host uses this command to tell the device to go into Hi-Power mode. The operations listed below are performed:

- Power is applied to the sensor on the camera head, and the DRAM on the main board.
- The VP, video processor, is released from reset.
- The DRAM controller is enabled, thus starting the DRAM refresh process.
- The firmware initialises, and enables, the VP_CTRL and CAPTURE modules.

This is a WRITE or host to device transfer which has no DATA transactions following the SETUP transaction.

bmRequestType	bRequest	wValue		wIndex		wLength	
01000000	000 00100	0x00	0x00	0x00	0x00	0x00	0x00

Table 2.8

2.2.5 GotoLoPower

The host uses this command to tell the device to go into Lo-Power mode. The operations listed below are performed:

- The firmware deinitialises, the VP_CTRL and CAPTURE modules. NOTE - this causes these modules to lose all state information they contained.
- The VP, video processor, is put in reset.
- The DRAM controller is disabled.
- Power is removed from the sensor on the camera head, and the DRAM on the main board.

This is a WRITE or host to device transfer which has no DATA transactions following the SETUP transaction.

bmRequestType	bRequest	wValue		wIndex		wLength	
01000000	000 00101	0x00	0x00	0x00	0x00	0x00	0x00

Table 2.9

2.2.6 GotoSuspend

This command will force the camera into suspend state on parallel port interface cameras. The camera can be woken out of suspend state by taking the parallel port interface signal nSelectIn LO -> HI.
This is a WRITE or host to device transfer which has no DATA transactions following the SETUP transaction.

bmRequestType	bRequest	wValue		wIndex		wLength	
01000000	000 00111	0x00	0x00	0x00	0x00	0x00	0x00

Table 2.10

2.2.7 GotoPassThrough

This command will put the parallel port of the module into pass-through mode.
The camera can be taken out of pass-through mode by sending a VPP (Vision Packet Protocol) packet to the camera.
This is a WRITE or host to device transfer which has no DATA transactions following the SETUP transaction.

bmRequestType	bRequest	wValue		wIndex		wLength	
01000000	000 01000	0x00	0x00	0x00	0x00	0x00	0x00

Table 2.11

2.2.8 ModifyCameraStatus

This command is used to change the state of a single byte within the CAMERA_STATUS structure that exists on the camera. The *StatusByte* parameter is used to specify which byte within this structure is to be altered. This index into the CAMERA_STATUS structure is the same as that used to reference the data in the DATA_IN buffer returned by the **GetCameraStatus** command.
The operation performed on the byte is as follows:
New_Value = (Old_Value & AndMask) | OrMask
Note the the use of the AndMask and OrMask allows the arbitary setting and resetting of bits in the target byte, as well as being able to simply assign a value to that byte.
This is a WRITE or host to device command with no DATA transactions following the SETUP transaction.

bmRequestType	**bRequest**	**wValue**		**wIndex**		**wLength**	
01000000	000 01010	StatusByte	And Mask	Or Mask	0x00	0x00	0x00

Table 2.12

2.3 System Module Commands

The three most significant bits of *bRequest* are set to 001 to specify a System Module Command.

2.3.1 ReadVCRegs

The host uses this command to read any 4 VC, Video Compressor, registers. The addresses of the four registers to be read are specified in wValue and wIndex as shown.
This is a READ or device to host transfer which has one DATA transaction following the SETUP transaction.

bmRequestType	**bRequest**	**wValue**		**wIndex**		**wLength**	
11000000	001 00001	Addr0	Addr1	Addr2	Addr3	0x04	0x00

Table 2.13

DATA IN byte	**Data**
1	Contents of Addr0
2	Contents of Addr1
3	Contents of Addr2
4	Contents of Addr3

Table 2.14

2.3.2 WriteVCReg

The host uses this command to write to a single VC register. It takes three parameters, the address of the VC register, an *AndMask* and an *OrMask*. The specified register is read and combined with the AndMask and OrMask as show below (in 'C' notation), before being written back to the specified register.

New_Value = (Old_Value & AndMask) | OrMask

Note the the use of the AndMask and OrMask allows the arbitary setting and resetting of bits in the target

register, as well as being able to simply write a value to the register.
This is a WRITE or host to device command with no DATA transactions following the SETUP transaction.

bmRequestType	**bRequest**	**wValue**		**wIndex**		**wLength**	
01000000	001 00010	Addr	And Mask	Or Mask	0x00	0x00	0x00

Table 2.15

2.3.3 ReadMCPorts

The host uses this command to read the four user I/O ports of the 8052 type microcontroller.
This is a READ or device to host transfer which has one DATA transaction following the SETUP transaction.

bmRequestType	**bRequest**	**wValue**		**wIndex**		**wLength**	
11000000	001 00011	0x00	0x00	0x00	0x00	0x04	0x00

Table 2.16

DATA IN byte	**Data**
1	Value of PORT 0
2	Value of PORT 1
3	Value of PORT 2
4	Value of PORT 3

Table 2.17

2.3.4 WriteMCPort

The host uses this command to change the state of a microcontroller user IO port register. It takes three parameters, the port number(0 to 3), an *AndMask* and an *OrMask*. The specified port register is read and combined with the AndMask and OrMask as show below (in 'C' notation), before being written back to the specified register.

New_Value = (Old_Value & AndMask) | OrMask

Note the the use of the AndMask and OrMask allows the arbitary setting and resetting of bits in the target register, as well as being able to simply write a value to the register.
This is a WRITE or host to device command with no DATA transactions following the SETUP transaction.

bmRequestType	**bRequest**	**wValue**		**wIndex**		**wLength**	
01000000	001 00100	Port	And Mask	Or Mask	0x00	0x00	0x00

Table 2.18

2.3.5 SetBaudRate

This command can be used to set the baudrate of the RS232 port used to output debug information.The baudrate is specified as a multiple of 1200 baud. eg 1 = 1200, 2 = 2400, 3 = 3600 ... 8 = 9600, 16 = 19200

etc.
This is a WRITE or host to device command with no DATA transactions following the SETUP transaction.

bmRequestType	bRequest	wValue		wIndex		wLength	
01000000	001 00101	BaudRate	0x00	0x00	0x00	0x00	0x00

Table 2.19

2.3.6 SetECPTiming

The host uses this command to instruct the device to use a slower, but safer, timing when streaming image data in ECP mode.
Setting SlowTiming to 1 causes a delay to be added between the camera setting up the data byte on the parallel port and taking nAutoFd LO.
Setting SlowTiming to 0 causes setting up of the data byte on the parallel port to be coincident with nAutoFd being taken LO.
This is a WRITE or host to device command with no DATA transactions following the SETUP transaction.

bmRequestType	bRequest	wValue		wIndex		wLength	
01000000	001 00110	SlowTiming	0x00	0x00	0x00	0x00	0x00

Table 2.20

2.3.7 ReadIDATA

The host uses this command to read any 4 bytes from 8052 IDATA space. The addresses of the four bytes to be read are specified in wValue and wIndex as shown.
This is a READ or device to host transfer which has one DATA transaction following the SETUP transaction.

bmRequestType	bRequest	wValue		wIndex		wLength	
11000000	001 00111	Addr0	Addr1	Addr2	Addr3	0x04	0x00

Table 2.21

DATA IN byte	Data
1	Contents of Addr0
2	Contents of Addr1
3	Contents of Addr2
4	Contents of Addr3

Table 2.22

2.3.8 WriteIDATA

The host uses this command to write to a byte in 8052 IDATA space. It takes three parameters, the

address of the byte, an *AndMask* and an *OrMask*. The specified register is read and combined with the AndMask and OrMask as show below (in 'C' notation), before being written back to the specified register.

New_Value = (Old_Value & AndMask) | OrMask

Note the the use of the AndMask and OrMask allows the arbitary setting and resetting of bits in the target register, as well as being able to simply write a value to the register.

This is a WRITE or host to device command with no DATA transactions following the SETUP transaction.

bmRequestType	**bRequest**	**wValue**		**wIndex**		**wLength**	
01000000	001 1000	Addr	And Mask	Or Mask	0x00	0x00	0x00

Table 2.23

2.3.9 GenericCall

The host may use this command to trigger a CALL to an arbitary subroutine in the 8052 CODE space.

NOTE - this command should only be used after careful studing of the CPIA.M51 file generated by the C51 compiler. Call to some functions may cause data corruption due to the C51 linkers data/idata overlay scheme.

This is a WRITE or host to device command with no DATA transactions following the SETUP transaction.

bmRequestType	**bRequest**	**wValue**		**wIndex**		**wLength**	
01000000	001 1001	AddrLo	AddrHi	0x00	0x00	0x00	0x00

Table 2.24

2.3.10 I2CStart

To be documented.

bmRequestType	**bRequest**	**wValue**		**wIndex**		**wLength**	
	001 1010						

Table 2.25

2.3.11 I2CStop

To be documented.

bmRequestType	**bRequest**	**wValue**		**wIndex**		**wLength**	
	001 1011						

Table 2.26

2.3.12 I2CWrite

To be documented.

bmRequestType	bRequest	wValue		wIndex		wLength	
	001 1100						

Table 2.27

2.3.13 I2CRead

To be documented.

bmRequestType	bRequest	wValue		wIndex		wLength	
	001 1101						

Table 2.28

2.4 Video Processor (VP) Control Module Commands

The three most significant bits of *bRequest* are set to 101 to specify a VP Control Module Command.

VP Control commands are used to control the colour video processing front end that takes raw pixel data from the camera head and colour processes it to YUV 4:2:2 format. These commands will be mainly used to adjust the colour quality of the image when the user requests it. Commands to adjust the camera framerate are also included and may be used if it is required to increase the sensitivity of the system under low-light conditions at the expense of framerate.

2.4.1 GetVPVersion

The host uses this command to request the VP hardware version. It returns the VP hardware version number (2 bytes) followed by the current camera head ID (2 bytes). The first byte specifies the main VP hardware version. The second byte specifies the revision within the main VP hardware version. Therefore, if the first byte returned was 1 and the second was 2, this would signify overall VP hardware version 1.2. The third and fourth bytes present a number which identifies the current camera head.

This is a READ or device to host transfer where one DATA IN transaction follows the SETUP transaction..

bmRequestType	bRequest	wValue		wIndex		wLength	
11000000	101 00001	0x00	0x00	0x00	0x00	0x04	0x00

Table 2.29

DATA IN byte	Data
1	Main Video Processor (VP) hardware version
2	Revision within the main VP hardware version
3	Camera Head ID (least significant byte)

DATA IN byte	Data
4	Camera Head ID (most significant byte

Table 2.30

2.4.2 SetColourParams

The host uses this command to control the colour of the image in terms of its brightness, contrast and saturation. These parameters are defined to range from 0 to 100, with 50 being the nominal value. Brightness controls the target exposure of the AEC algorithm, contrast inversely controls the intensity of gamma correction applied, and saturation scales the entries in the 2nd and 3rd rows of the RGB->YUV matrix. Contrast values are quantised in steps of 8, limited by the number of available gamma steps. 1-02 firmware has a maximum contrast of 80 instead of 96 for later firmware versions.
This is a WRITE or host to device command with no DATA transactions following the SETUP transaction. .

bmRequestType	**bRequest**	**wValue**		**wIndex**		**wLength**	
01000000	101 00011	Brightness	Contrast	Saturation	0x00	0x00	0x00

Table 2.31

2.4.3 SetExposure

This command is used by the host to enable/disable automatic exposure and gain control.
If *GainMode* is set to 0 the gain ceiling remains unchanged, otherwise the gain ceiling is set to the value of (*GainMode* - 1). The gain ceiling may be set to {0,1,2,3} representing sensor gains of {1,2,4,8}.
If *ExpMode* is is 0 the exposure values in the following DATA transaction are ignored. If *ExpMode* is set to 1 the coarse, fine and gain settings in the following DATA transaction are downloaded to VP. Note that the fine exposure downloaded to the sensor is twice that of the FineExp parameter to enable an 8 bit parameter to control the sensor's full fine exposure range. If *ExpMode* is set to 2 then AEC is enabled and the settings in the following DATA transaction are not used. If *ExpMode* is set to 3 then AEC is disabled and the settings in the following DATA transaction are not used.
if *CompMode* is set to 0 the compensation gains in the following DATA transaction are ignored. If *CompMode* is set to 1 then compensation gains in the following DATA transaction are written.
If *CentreWeight* is set to 0 no change is made to the scene weighting used in the exposure calculation. If *CentreWeight* is set to 1 the centre and lower areas of the scene are given more precedence in the expsoure calculation. If *CentreWeight* is set to 2 the whole scene is given equal weighting in the exposure calculation.
Note that several bugs exist in 1-02 firmware. Setting *GainMode* to 1 sets the gain ceiling to 1, i.e. a sensor gain of 2, and all other values are ignored. When setting exposure values, CoarseExpHi has no effect and only FineExp parameters of 127 or less are used correctly.
This is a WRITE or host to device transfer which has one DATA transaction following the SETUP transaction.

bmRequestType	**bRequest**	**wValue**		**wIndex**		**wLength**	
01000000	101 00100	GainMode	ExpMode	CompMode	CentreWeight	0x08	0x00

Table 2.32

DATA OUT byte	Data
1	Gain
2	FineExp/2 (max 127 in 1-02)
3	CoarseExpLo
4	CoarseExpHi (max 0 in 1-02)
5	RedComp
6	Green1Comp
7	Green2Comp
8	BlueComp

Table 2.33

2.4.4 SetColourBalance

This command is used to either enable or disable automatic colour balance. If auto colour balance is disabled it may be used to set the colour channel gains.

Setting *BalanceMode* to 0 causes no change to colour balance.

SSetting *BalanceMode* to 1 allows the manual setting of the colour balance parameters. The accompanying Red/Green/BlueGain parameters are taken as the new values for the three channels. Values may be in the range 0-212 for each channel. Values above 212 cause the colour balance matrix to overflow.

Setting *BalanceMode* to 2 enables the auto-colour balance algorithm. The camera will automatically adjust the three colour channel gains to acheive optimum colour balance.

Setting *BalanceMode* to 3 disables the auto-colour balance algorithm. It will effectively freeze the current colour balance settings.

This is a WRITE or host to device command with no DATA transactions following the SETUP transaction..

bmRequestType	**bRequest**	**wValue**		**wIndex**		**wLength**	
01000000	101 00110	BalanceMode	RedGain	GreenGain	BlueGain	0x00	0x00

Table 2.34

2.4.5 SetSensorFPS

This command is used by the host to set the sensor framerate.
The *SensorBaseRate* parameter may take the following values 0 = 25 fps, 1 = 30 fps.
The *SensorClkDivisor* parameter is used set a clock divisor that slows the clock sent to the sensor. The following values are valid: 0 = div 1; 1 = div 2; 2 = div 4; 3 = div 8.
For example if *SensorBaseRate* = 1 (30fps) and *SensorClockDivisor* = 2, the resultant frame rate would be 30 / 4 = 7.5fps.

This is a WRITE or host to device command with no DATA transactions following the SETUP transaction. .

bmRequestType	**bRequest**	**wValue**		**wIndex**		**wLength**	
01000000	101 00111	SensorClkDivisor	SensorBaseRate	0x00	0x00	0x00	0x00

Table 2.35

2.4.6 SetVPDefaults

This command is used by the host to set the VP_CTRL module to its default colour processing parameters, as listed in the table below.

Parameter	**Default Applied**
Brightness	50
Contrast	50
Saturation	50
Gain	0
Flicker Control	Off
Coarse Exposure	185
Fine Exposure	0
Red Comp	220
Green1 Comp	214
Green2 Comp	214
Blue Comp	230
Gain Ceiling	2 (= max gain 4)
Exposure Weighting	Centre Weighted
VlOffset Table	24, 28 ,30, 30
ApCor Table	0x18, 0x16, 0x24, 0x34
Red Gain	32
Green Gain	06
Blue Gain	92

Table 2.36

This is a WRITE or host to device command with no DATA transactions following the SETUP transaction.

bmRequestType	bRequest	wValue		wIndex		wLength	
01000000	101 01000	0x00	0x00	0x00	0x00	0x00	0x00

Table 2.37

2.4.7 SetApcor

This command is used by the host to control the aperture correction parameters used by VP to sharpen the image. The apcor register is modified by the AEC algorithm when the sensor gain is changed. The parameters (*gain1*, *gain2*, *gain4*, *gain8*) reflect the aperture correction value used for each sensor gain. The aperture correction value is defined with the 4 most significant bits representing apcor threshold and the 4 least significant bits representing apcor intensity.
This is a WRITE or host to device command with no DATA transactions following the SETUP transaction.

bmRequestType	bRequest	wValue		wIndex		wLength	
01000000	101 01001	gain1	gain2	gain4	gain8	0x00	0x00

Table 2.38

2.4.8 SetFlickerCtrl

This command is used by the host to control the operation of the anti-flicker algorithm. Refer to the driver notes document for details of how to correctly enable flicker control.
If *FlickerMode* is set to 0 then flicker control is disabled. If *FlickerMode* is set to 1 then flicker control is enabled. Auto-exposure must be enabled for flicker control to run, and disabling auto-exposue will freeze the flicker control algorithm.
This is a WRITE or host to device command where there are no DATA transactions following the SETUP transaction.

bmRequestType	bRequest	wValue		wIndex		wLength	
01000000	101 01010	FlickerMode	CoarseJump	AllowableOverExposure	0x00	0x00	0x00

Table 2.39

2.4.9 SetVLOffset

The video level offset register is set by the AEC algorithm whenever the sensor gain is updated. The parameters (*gain1*, *gain2*, *gain4*, *gain8*) reflect the video level offset value used for each sensor gain. This command sets the offset which is appropriate for the current gain immediately.
This is a WRITE or host to device command with no DATA transactions following the SETUP transaction.

bmRequestType	bRequest	wValue		wIndex		wLength	
01000000	101 01011	gain1	gain2	gain4	gain8	0x00	0x00

Table 2.40

2.4.10 GetColourParameters

This command is used by the host to get the current colour parameters values.

This is a READ or device to host transfer which has one DATA transaction following the SETUP transaction.

bmRequestType	bRequest	wValue		wIndex		wLength	
11000000	101 10000	0x00	0x00	0x00	0x00	0x03	0x00

Table 2.41

DATA IN byte	Data
1	Brightness
2	Contrast
3	Saturation

Table 2.42

2.4.11 GetColourBalance

This command is used by the host to get the current colour correction gains from the camera. These are in the range 0 - 212.
This is a READ or device to host transfer which has one DATA transaction following the SETUP transaction.

bmRequestType	bRequest	wValue		wIndex		wLength	
11000000	101 10001	0x00	0x00	0x00	0x00	0x03	0x00

Table 2.43

DATA IN byte	Data
1	RedGain
2	GreenGain
3	BlueGain

Table 2.44

2.4.12 GetExposure

This command is used by the host to get the current exposure, sensor gain and channel compensation gain values from the camera.
This is a READ or device to host transfer which has one DATA transaction following the SETUP

transaction.

bmRequestType	bRequest	wValue		wIndex		wLength	
11000000	101 10010	0x00	0x00	0x00	0x00	0x08	0x00

Table 2.45

DATA IN byte	Data
1	Gain
2	FineExp
3	CoarseExpLo
4	CoarseExpHi
5	RedComp
6	Green1Comp
7	Green2Comp
8	BlueComp

Table 2.46

2.4.13 SetSensorMatrix

The basic colour correction matrix may be updated with this command to accommodate changes in the colour filters. The colour balance algorithm multiplies red, green and blue gains by this matrix to generate the colour correction matrix which is downloaded to VP. An immediate update is forced in firmware after the basic matrix has been updated.

This is a WRITE or host to device transfer which has one DATA transaction following the SETUP transaction.

bmRequestType	bRequest	wValue		wIndex		wLength	
	101 10011	m00	m01	m02	m10	0x05	

Table 2.47

DATA OUT byte	Data
1	m11
2	m12
3	m20
4	m21

DATA OUT byte	Data
5	m22

Table 2.48

2.4.14 ColourBars

To be documented.

bmRequestType	bRequest	wValue		wIndex		wLength	
	101 11101						

Table 2.49

2.4.15 ReadVPRegs

The host uses this command to read any 4 VP Registers. The addresses of the four registers to be read are specified in wValue and wIndex as shown.
This is a READ or device to host transfer which has one DATA transaction following the SETUP transaction.

bmRequestType	bRequest	wValue		wIndex		wLength	
11000000	101 11110	Addr0	Addr1	Addr2	Addr3	0x04	0x00

Table 2.50

DATA IN byte	Data
1	Contents of Addr0
2	Contents of Addr1
3	Contents of Addr2
4	Contents of Addr3

Table 2.51

2.4.16 WriteVPReg

The host uses this command to write to a single VP Register. It takes three parameters, the address of the VP register, an *AndMask* and an *OrMask*. The specified register is read and combined with the AndMask and OrMask as show below (in 'C' notation), before being written back to the specified register.

New_Value = (Old_Value & AndMask) | OrMask

Note the the use of the AndMask and OrMask allows the arbitary setting and resetting of bits in the target register, as well as being able to simply write a value to the register.

This is a WRITE or host to device command with no DATA transactions following the SETUP transaction.

bmRequestType	**bRequest**	**wValue**		**wIndex**		**wLength**	
01000000	101 11111	Addr	And Mask	Or Mask	0x00	0x00	0x00

Table 2.52

2.5 Capture Control Commands

The three most significant bits of *bRequest* are set to 110 to specify a Capture Control Command.

These commands are used to control the image capture / upload process. Commands allow the selection of image sizes/formats together with the compression techniques to apply before upload.

2.5.1 GrabFrame

This command requests that the camera grab the next video frame.

The *StreamStartLine* parameter specifies a line number, divided by 2, at which point the image stream to the host will be enabled. This parameter is provided so that the host may choose to upload the data of a highly compressed image in a single burst by delaying the time when the upload is started until later in the frame when more data should be available to upload which will prevent the upload from pausing. On the PP camera an interrupt will be sent to the host via the nAck line at the time at which the stream is enabled.

The operation of this command depends on whether *SetGrabMode* command has been used to enable *ContinuousGrab* mode or not. When *ContinuousGrab* is disabled a *GrabFrame* command will cause a single grab of the next available video frame to be performed. When *ContinuousGrab* is enabled a *GrabFrame* command does not initiate a fresh grab, but will cause the frame currently being grabbed to be uploaded.
If the *StreamStartLine* parameter is set in this case, and if a full frame of data is already available, then the interrupt to the host will be sent immediately after this command is executed.
Note: If the camera is in stream capture mode, this command will be ignored. Command is only processed when camera is in frame capture mode.

This is a WRITE or host to device command with no DATA transactions following the SETUP transaction.

bmRequestType	**bRequest**	**wValue**		**wIndex**		**wLength**	
01000000	110 00001	0x00	StreamStartLine	0x00	0x00	0x00	0x00

Table 2.53

2.5.2 UploadFrame

This command is used to enable the DataStream process on the camera that provides image data to the USB Isochronous Channel (or PP interface). Only a single frame of video will be streamed, after which time the DataStream process will be disabled.
When the USB interface is used this command causes the StreamState field of the Camera Status to be set to STREAM_OPEN. When used with the PP interface it causes the StreamState field of the Camera Status to be set to STREAM_READY (it becomes STREAM_OPEN during the time that the parallel port is in ECP or Nibble Upload mode).
When the PP interface is used this command starts only the DataStream process, it does not enable the transfer of data, via the PP Auto Mode hardware. To do this the host must negoitiate into the appropriate, ECP or Nibble, upload mode to start the actual transfer.

The *ForceUpload* parameter can be used to force the immediate upload of the current image in the camera's DRAM, rather than waiting for the data of the next frame to be available. Using this parameter it is possible to repeatedly upload the same image from the camera.
Note: If the camera is in stream capture mode, this command will be ignored. Command is only processed when camera is in frame capture mode.
This is a WRITE or host to device command with no DATA transactions following the SETUP transaction.

bmRequestType	**bRequest**	**wValue**		**wIndex**		**wLength**	
01000000	110 00010	ForceUpload	0x00	0x00	0x00	0x00	0x00

Table 2.54

2.5.3 SetGrabMode

The command is used to enable/disable the continuous grab mode of operation.

When *ContinuousGrab* is set to 1 the Grab process will operate continuously, thus making available to the host the most recent frame of video data when a *GrabFrame* or *UploadFrame* request is issued. When *ContinuousGrab* is set tp 0 then a sinle Grab will be initiated when a *GrabFrame* request is issued.
This is a WRITE or host to device command with no DATA transactions following the SETUP transaction.

bmRequestType	**bRequest**	**wValue**		**wIndex**		**wLength**	
01000000	110 00011	ContinuousGrab	0x00	0x00	0x00	0x00	0x00

Table 2.55

2.5.4 InitStreamCap

Used to initialised the stream capture process. This command puts the capture module into stream capture mode, however the actual capture and uploading of images will not start until a *StartStreamCap* command is received.

The *SkipFrames* parameter is used to request the number of sensor frames that are to be skipped between each upload, i.e. setting SkipFrames to 2 would request the upload of every 3rd frame. If the upload process of the last frame takes longer than the requested number of frames to skip the next upload will take place immediately the last finishes.
The *StreamStartLine* parameter specifies a line number, divided by 2, at which point the image stream to the host will be enabled. This parameter is provided so that the host may choose to upload the data of a highly compressed image in a single burst by delaying the time when the upload is started until later in the frame when more data should be available to upload which will prevent the upload from pausing. On the PP camera an interrupt will be sent to the host via the nAck line at the time at which the stream is enabled.

NOTE - Sending this command puts the CAPTURE module into stream capture mode. Once in this mode the capture module will refuse to perform either *GrabFrame* or *UploadFrame* commands until a *FiniStreamCap* command has been sent, to put it back into frame mode.

This is a WRITE or host to device command with no DATA transactions following the SETUP transaction.

bmRequestType	**bRequest**	**wValue**		**wIndex**		**wLength**	
01000000	110 00100	SkipFrames	StreamStartLine	0x00	0x00	0x00	0x00

Table 2.56

2.5.5 FiniStreamCap

Puts the CAPTURE module back into frame mode, thus allowing the processing of *GrabFrame* and *UploadFrame* commands.

bmRequestType	**bRequest**	**wValue**		**wIndex**		**wLength**	
01000000	110 00101	0x00	0x00	0x00	0x00	0x00	0x00

Table 2.57

2.5.6 StartStreamCap

Starts the autonomous stream capture / upload of images to the host. In this mode a frame will be captured and uploaded to the host at the requested framerate without the host having to explictly send a request for each frame.
This is a WRITE or host to device command with no DATA transactions following the SETUP transaction.

bmRequestType	**bRequest**	**wValue**		**wIndex**		**wLength**	
01000000	110 00110	0x00	0x00	0x00	0x00	0x00	0x00

Table 2.58

2.5.7 EndStreamCap

Stops the autonomous stream capture / upload of images. NOTE - a further *FiniStreamCap* command must be sent to the camera before it will process further *GrabFrame* and *UploadFrame* commands.

bmRequestType	**bRequest**	**wValue**		**wIndex**		**wLength**	
01000000	110 00111	0x00	0x00	0x00	0x00	0x00	0x00

Table 2.59

2.5.8 SetFormat

This command is used to select the video format that the camera is to capture.

The *VideoSize* parameter selects between 0 = QCIF(176x144) and 1 = CIF(352x288) image sizes.

The *SubSample* parameter selects between 0 = 4:2:0 and 1 = 4:2:2.

The *YUVOrder* parameter selects between 0 = YUYV and 1 = UYVY byte order.

The selection of YUYV or UYVY byte order may be useful if the host has a graphics adaptor that can perform hardware YUV -> RGB conversion.
This is a WRITE or host to device command with no DATA transactions following the SETUP transaction.

bmRequestType	**bRequest**	**wValue**		**wIndex**		**wLength**	
01000000	110 01000	VideoSize	SubSample	YUVOrder	0x00	0x00	0x00

Table 2.60

2.5.9 SetROI

Allows the setting of a ROI, within the full frame VideoSize (CIF or QCIF), that will be used during capture and upload operations.

The *ColStart* and *ColEnd* are in terms of 8 pixel blocks.ie ColStart = 2 means start at pixel column 16.

The *RowStart* and *RowEnd* parameters are in terms of 4 lines.

When CIF format video is being captured 0 < ColEnd <= 44, 0 < RowEnd <= 72.

When QCIF format video is being captured 0 < ColEnd <= 22, 0 < RowEnd <= 36.
This is a WRITE or host to device command with no DATA transactions following the SETUP transaction.

bmRequestType	**bRequest**	**wValue**		**wIndex**		**wLength**	
01000000	110 01001	ColStart	ColEnd	RowStart	RowEnd	0x00	0x00

Table 2.61

2.5.10 SetCompression

Selects the compression mode and options the camera is to use before upload of the image to the host.
CompMode can take the following values:

0 = Compression disabled, no compression will be used
1 = Auto Compression, camera will automatically adjust Decimation and YUV thresholds.
2 = Manual Compression, host will set Decimation and YUV thresholds.

Decimation can take the following values:

0 = No decimation
1 = Decimate image, vertical decimation is preformed which results in only half the normal number of lines being sent to the host.

During AutoCompression the *Decimation* parameter controls whether the dynamic compression control algorithm can switch in Decimation or not.
This is a WRITE or host to device command with no DATA transactions following the SETUP transaction.

bmRequestType	**bRequest**	**wValue**		**wIndex**		**wLength**	
01000000	110 01010	CompMode	Decimation	0x00	0x00	0x00	0x00

Table 2.62

2.5.11 SetCompressionTarget

When Auto Compression mode is selected, via the *SetCompression* command, this command is used to control the performance metric and limits of the algorithm that dynamically optimises the compression process. Two modes of automatic compression are supported, targeting a desired framerate and targeting a quality level. The mode is selected by the parameter *FRTargeting*:

0 = target quality
1 = target framerate

The *TargetFR* and *TargetQ* parameters are used to determine the actual target level/rate for a targeted parameter and the minimum acceptable level/rate for the untargeted parameter. For example, when targeting quality, the *TargetQ* parameter is set to the quality desired, while the *TargetFR* parameter is set to the minimum framerate required. Importantly, the minimum framerate is maintained, if possible, regardless of quality achieved. Similarly, for framerate targeting the *TargetQ* value becomes the minimum acceptable quality regardless of framerate achieved. Stated simply, the task of achieving the minimum

value overides, the targeted level/rate.

This is a WRITE or host to device command with no DATA transactions following the SETUP transaction.

bmRequestType	**bRequest**	**wValue**		**wIndex**		**wLength**	
01000000	110 01011	FRTargeting	TargetFR	TargetQ	0x00	0x00	0x00

Table 2.63

2.5.12 SetYUVThresh

Used to set the Y and UV thresholds that control the inter-frame differencing process.

These threshold values are used to determine which pixels have changed significantly, and thus must be uploaded to the host. They can take the following range of values, 0 <= *YThreshold* < 32, 0<= *UVThreshold* < 32.
This is a WRITE or host to device command with no DATA transactions following the SETUP transaction.

bmRequestType	**bRequest**	**wValue**		**wIndex**		**wLength**	
01000000	110 01100	YThreshold	UVThreshold	0x00	0x00	0x00	0x00

Table 2.64

2.5.13 SetCompressionParams

This command can be used to alter the dynamic control of the CPIA compression algorithm. Unless fully familiar with the dynamic compression control algorithm it is advised that this command is left unused.
Each parameter's effect on the dynamic compression algorithm is as follows:
Hysteresis - specifies the width of a dead-band around the targetted framerate/quality for which the current compression thresholds are left unmodified.
ThreshMax - specifies the maximum value to which the compression thresholds can be set.
SmallStep - specifies the size of a small change that the algorithm can make to the compression thresholds in order to achieve the desired target performance.
LargeStep - specifies a large change in compression thresholds that the algorithm can make in order to achieve the desired target performance.
DecimationHysteresis - specifies the width of a band measured from the maximum setting of the threshold values, for which decimation is left switched-on.
FRDiffStepThresh - specifies the magnitude of the difference between current framerate and desired framerate, which in turn is used to select between SmallStep or LargeStep for the dynamic control algorithm.
QDiffStepThresh - specifies the magnitude of the difference between current quality and desired quailty, which in turn is used to select between SmallStep or LargeStep for the dynamic control algorithm.
DecimationThreshMod - specifies a positive increase in the compression thresholds to apply immediately in the case of Decimation being switched-off.

This is a WRITE or host to device transfer which has one DATA transaction following the SETUP

transaction.

bmRequestType	bRequest	wValue		wIndex		wLength	
01000000	110 01101	0x00	0x00	0x00	0x00	0x08	0x00

Table 2.65

DATA OUT byte	Data
1	Hysteresis
2	ThreshMax
3	SmallStep
4	LargeStep
5	DecimationHysteresis
6	FRDiffStepThresh
7	QDiffStepThresh
8	DecimationThreshMod

Table 2.66

2.5.14 DiscardFrame

This command is used to discard the currently captured frame - such that a grab must occur before a new frame will be available for upload.

bmRequestType	bRequest	wValue		wIndex		wLength	
01000000	110 01110	0x00	0x00	0x00	0x00	0x00	0x00

Table 2.67

2.6 Debug Module Commands

The three most significant bits of *bRequest* are set to 111 to specify a Debug Module Command.

These commands can be used to cause the execution of a variety of debug related routines in each module.

2.6.1 OutputRS232

Outputs a string to the RS232 port. The first seven bytes that make up the WRITE packet are output to the RS232 port.

This is a WRITE or host to device transfer which has one WRITE transactions following the SETUP

transaction.

bmRequestType	bRequest	wValue		wIndex		wLength	
01000000	111 00001	0x00	0x00	0x00	0x00	0x08	0x00

Table 2.68

2.6.2 AbortProcess

The host uses this command to abort the command processing loop. On receiving this command the device will return control to the top level interactive debug menu. This menu is used with via the RS232 port of the device.
This is a WRITE or host to device transfer which has no DATA transactions following the SETUP transaction.

bmRequestType	bRequest	wValue		wIndex		wLength	
01000000	111 00100	0x00	0x00	0x00	0x00	0x00	0x00

Table 2.69

2.6.3 SetDRAMPage

Used to set the DRAM area to upload when the StartDRAMUpload command is called
This is a WRITE or host to device transfer which has no DATA transactions following the SETUP transaction.

bmRequestType	bRequest	wValue		wIndex		wLength	
01000000	111 00101	StartLo	StartHi	EndLo	EndHi	0x00	0x00

Table 2.70

2.6.4 StartDRAMUpload

Used to start an upload of the DRAM via the parallel port interface.
This is a WRITE or host to device transfer which has no DATA transactions following the SETUP transaction.

bmRequestType	bRequest	wValue		wIndex		wLength	
01000000	111 00110	0x00	0x00	0x00	0x00	0x00	0x00

Table 2.71

2.6.5 StartDummyStream

Switches in the DS_TEST_DATA register into the data stream and sets it to provide an incrementing data stream. This can can be used by both USB and PP interfaces to provide a dummy data source.
This is a WRITE or host to device transfer which has no DATA transactions following the SETUP

transaction.

bmRequestType	bRequest	wValue		wIndex		wLength	
01000000	111 01000	0x00	0x00	0x00	0x00	0x00	0x00

Table 2.72

2.6.6 AbortStream

Used to terminate a Dummy Stream upload and stop the stream channel.
This is a WRITE or host to device transfer which has no DATA transactions following the SETUP transaction.

bmRequestType	bRequest	wValue		wIndex		wLength	
01000000	111 01001	0x00	0x00	0x00	0x00	0x00	0x00

Table 2.73

2.6.7 DownloadDRAM

Used to download data into the camera's DRAM. It can be used to download up to 32KBytes to the DRAM. The data received is written into DRAM at the start address specified by the last SetDRAMPage command (the EndAddress fields in that command being ignored). The number of data bytes to follow is specified in the last two bytes of the SETUP transaction. This number should be a multiple of 8.
This is a WRITE or host to device transfer which has a number of DATA transactions following the SETUP transaction.

bmRequestType	bRequest	wValue		wIndex		wLength	
01000000	111 01010	0x00	0x00	0x00	0x00	DataBytesLo	DataBytesHi

Table 2.74

2.6.8 NullCmd

To be defined and implemented.
This command will provide a universal data source or sink functionality to test the passing of data to and from the device during command transactions.

3 Notes on use of the CPiA Command Set

3.1 Introduction

This document gives guidance about the use of the CPIA Command Set to implement a video capture driver. It is not intended to be a comprehensive guide, but rather give a rough outline of the way that the command set was intended to be used.

It is assumed the reader is familiar with general Video for Windows (VfW) terminology, and the operation of the VfW Capture Driver public interface in particular.

3.2 Terms

3.2.1 VP

Video Processor. This is the front end video processing stage of the CPIA chipset. It takes raw bayer colourised data and outputs colour processed YUV 4:2:2 data.

Compression Engine

The compression engine works in conjunction with the DataStream engine to perform the capture and image compression functions. The compression engine is responsible for the image capture process.

3.2.2 Data Stream Engine

The DataStream Engine is responsible to for taking data out of DRAM and passing it to either the USB or parallel port interfaces.

3.2.3 PP Interface

Parallel port interface, consists of both the host port, connect to the host PC, and the pass-through port which may be used to attach other PP peripherals.

3.2.4 VC

Video Compressor. This includes all the functionality of the of the Compression Engine, Data Stream Engine, USB interface, PP interface together with other logic used to "glue" these major modules together.

3.2.5 Micro-controller

A 8052 type micro-controller is used to perform all the high level functionality required of the design. It is responsible for processing host commands and supervising the operation of the VP, Compression and Data Stream processes

3.3 Power Management

The camera can be in one of three possible states:

3.3.1 Suspend State < 500uA

In suspend state the whole camera, except for a small power management module, is shutdown to reduce power consumption to an absolute minimum.

In a USB camera this state is entered on a USB Suspend condition and left automatically when the USB Suspend state is removed by the host.

In a Parallel Port camera the *GotoSuspend* command is used to enter suspend state and a parallel port control line, nSelectIn, is used to trigger an exit from this state.

Once the device has entered suspend mode all state information is lost, with the exception of the USB Core Configuration and Device Address. When suspend state is exited the micro-controller, and almost all of the VC device is reset. The power management block of VC is not reset however, and from a register in this block the micro-controller can detect the fact that it is being restarted after being in a suspend state. When it detects this condition it modifies its usual start up sequence to prevent the usual selftest diagnostics from being run.

3.3.2 Lo-Power State < 100mA

Lo-Power state is the default state of the system after initial power up. In this state the micro-controller is active, together with the USB and parallel port interfaces. When in this state the camera head, VP, Compression and DRAM controller are all disabled.

In Lo-Power state USB cameras will process all Standard commands, to allow device enumeration, and a subset of the Vendor Specific commands.

In PP cameras the parallel port interface is set to pass-through mode after power-up and the device will not respond to host commands until a Daisy Chain CPP selection sequence has been received (this not yet implemented). Once selected the camera can execute a subset of the CPIA Command Set.

3.3.3 Hi-Power State < 500mA

Hi-Power state is entered from Lo-Power state on receipt of the *GotoHiPower* command from the host. The camera head, VP, Compression and DRAM are then powered up and after a short initialisation phase the camera will be in state whereby it can deliver video data to the host. The *GotoHiPower* command should only be issued by the driver once a VfW application has been launched and requires video data.

When the host driver determines that no further applications are requiring video data it should place the camera back in Lo-Power mode by sending a *GotoLoPower* command.

3.4 Initialisation

For PP cameras the driver should first establish communications with the camera by switching the camera out of pass-through mode, via a Daisy Chain CPP.

For USB cameras the driver should, once enumeration is complete select Config #1, Interface #1, AlterNate Setting #0. This assumes that following device enumeration USB system software has reported that a high power bus powered device can be supported. If this is not true, then on VfW application launch the driver should pass this information to the application so that the User can be informed via a Message Dialogue that a high-power bus-power device cannot be supported on this branch of the USB.

Get the version information from the camera via the *GetCPIAVersion* and *GetPnPID* commands.

Check the status of the camera via the *GetCameraStatus* command. In particular the FatalError field should be checked to ensure that no fatal errors have been detected during the selftest process.

Once the VfW Application has been launched and requires video data the driver must switch the camera into Hi-Power state via the *GotoHiPower* command.

Check the status of the camera again via the *GetCameraStatus* command. Much of the functionality of the camera can only be tested once in HiPower state and so the FatalError field must be checked following the first Lo->Hi-Power transition.

For USB cameras the driver, must now select Interface #1, with the highest possible AlterNate setting. If none of AlterNate settings #3, #2 and #1 can be supported by the USB system then the driver must pass this information to the VfW application. The application can then inform the USer (via a Message Dialogue) that the USB does not currently have sufficient resource to stream video from the VISION USB camera and the User must disconnect another USB device which they do not require to free resource. Furthermore, in the case where only Alternate setting #0 can be supported by USB the driver must not schedule isochronous transfers to Endpoint 1 because the isochronous pipe has a 0 byte data payload.

The camera can now be regarded as fully operational. The driver may now set the desired video configuration and start the capture and upload of video frames to the host.

Note that the driver must select Alternate setting #0 when no VfW application is using the device so that the VISION USB camera does not'claim' USB bandwidth which is not being used.

3.5 Configuration

The host may set various options of the configuration: (NOTE - the actual commands to set this configuration information will be added once they are finalised)

3.5.1 Setting the Video Format Required

The *SetFormat* command is used by the driver to select the format of the video it requires. The VP device can deliver data in either CIF(353x288) or QCIF(176x144) formats. Also either 4:2:2 (UV subsampled by a factor of 2 horizontally) or 4:2:0(UV subsampled by a factor of 2 horizontally and vertically) YUV data can be delivered.

3.5.2 ROI of Capture

The host uses the *SetROI* command to instruct the camera to only capture and upload a Region Of Interest of the total image. This may be used to deliver SIF(320x240) or QSIF(160x120) images to the host.

3.5.3 Colour Parameters

The host uses the *SetColourParams* command to control the brightness/contrast/saturation of the image. It is desirable that the users preferred setting of these values is reloaded each time the driver starts. The host may read these values back from the camera, via the *GetColourParams* command, when it is exiting and store them on the host.

In addition the colour balance gains should also be read from the camera when the driver exits an stored on the host. While these values are not user configurable, the colour balance algorithm has a relatively long time constant so it is useful to "prime" the algorithm with the values it previously calculated for the scene. The *Get/SetColourBalance* commands are used to retrieve and set these values.

3.5.4 Video Dialogue Box

To enable the user to control image quality a dialogue box should be provided to control the brightness, contrast, saturation and compression quality required. This dialogue box is termed the

VIDEO_EXTERNALIN dialogue in VfW terminology, but is referred to as the *Video In* dialogue in the VidCap application.

3.5.5 Format Dialogue Box

To enable the user to select the appropriate capture image size and format a dialogue box should be provided. This dialogue box is termed the *VIDEO_IN* dialogue in VfW terminology, but is referred to as the *Video Format* dialogue in the VidCap application.

3.6 Frame Capture

Frame capture mode is the simplest way of capturing and transferring images to the host. It is the default mode of operation for VfW drivers and used in “Preview” mode of VidCap.

This mode of operation is controlled by the host on a frame by frame basis, with the host requesting each frame capture operation explicitly.

An image capture operation is requested by the sending of a *GrabFrame* command. When the camera receives this it will cause the next frame received from the VP to be captured to DRAM. Once the capture process has started, and data is available in DRAM, the image stream to the host, via PP or USB, will be enabled.

In the case of a USB camera, this “enabling of the image stream” actually causes the data to start being sent over the isochronous channel to the host.

In the case of a PP camera the “enabling of the image stream” does not in itself cause data to start being transferred to the host. All it does is to allow the host to request an upload mode without error. On PP cameras, after the host has sent a *GrabFrame* command it should either poll the camera, via the *GetCameraStatus* command, until it detects that StreamState == STREAM_READY, or wait for the camera to generate a host interrupt, before it attempts to open the upload channel.

3.7 Stream Capture

In addition to frame capture mode the camera can operate in stream capture mode, whereby it will autonomously stream image data to the host at a pre-requested framerate.

There is a two stage initialisation process to set up a stream capture, similar to that used in VfW. Firstly the required parameters of the capture are sent to the camera via a *InitStreamCap* command. Once the configuration is set the stream capture can be started by send a *StartStreamCap* command.

Once started the camera will attempt to send frames to the host at the requested framerate. It may not be able to sustain this framerate if the PP or USB channels do not transfer the data fast enough, in which case it will send frames as fast as it can, i.e. it will gradually degrade the framerate.

The frame header that prepends each frame will contain information that the host will use to monitor the operation of the camera. This header contains: the time the frame was captured, the compression options used to compress it, a copy of the CameraStatus structure (as would be returned by *GetCameraStatus*).

When the host requires to finish stream capture mode it must send a *StopStreamCap* and *FiniStreamCap* commands before requesting any more frame captures.

3.8 Image Reformating / Colour Space Conversion

The image data stream delivered by the camera to the host consists of YUV image data together with additional bytes to allow the host to delimit video frames and lines. Each video frame of data is prepended with a 64 byte header that contains information about the image format, size, compression method, etc. Additional fields contain a time stamp and a copy of the camera status information.

The basic image data delivered by the camera is YUV 4:2:2 format, with a YUYV bytes order. Optionally 4:2:0 data can be delivered.

The image compression technique used is that of performing an inter frame difference comparison between the new incoming frame and the last one capture, and only sending to the host pixels that have changed significantly since the last frame. Pixels that have changed are sent directly to the host while horizontal runs of unchanged pixels are counted and sent as single bytes. The least significant bit of each byte within the image data stream is used to denote whether it contains pixel data or an "unchanged pixel count".

It should be noted that this compression method is very dependant of the amount of change in the scene from frame to frame and so the actual framerate delivered by the camera will vary based on the amount of movement in the scene.

The driver will be responsible for performing the decompression of the compressed image data together with the conversion from YUV 4:2:2 (or 4:2:0) format into the format required by the application.

Common image formats that must be supported are RGB 32, 24 and 16 bit, RGB 8 bit palettised, I420, YV12(4:2:0 planar), UYVY(4:2:2) and YUY2(4:2:2).

As the colour space conversion process from YUV to RGB is a relatively expensive process it is desirable that the decompression process also incorporates the colour space conversion stage. Thus at each frame only those pixels that have changed will be converted from YUV to RGB. This will minimise the processor overhead in situations where there is very little movement in the scene.

If the host PC has a graphics adaptor that can perform hardware YUV -> RGB conversion then this should be utilised. In this case the driver would decompress the data into a YUV format and then pass it to the graphics adaptor for conversion.

3.9 Image Stream Format

This section describes the image data stream from the CPIA device. Each frame of data consists of three parts, the Frame Header, the lines of video data and the End of Frame code.

3.9.1 Frame Header

Each frame of video data is preceded by a 64 byte header. This header contains information about the image format and compression of the video data together with status information about the camera. This allows the host to monitor the health/performance of the camera without having to continuously send GetCameraStatus commands.

The format of the frame header is detailed below.

Address	Name	Description
0	HEADER0	0x19 - Magic number
1	HEADER1	0x68 - Magic number
2	TIMESTAMP_0	Time at which image was captured. Format to be decided. (Not related to the USB SOF timestamp).....
3	TIMESTAMP_1	
...		

Table 4

Address	Name	Description
6	TIMESTAMP_2	Time at which image was captured. Format to be decided. (Not related to the USB SOF timestamp).....
7	TIMESTAMP_3	
...		
16	VIDEO_SIZE	0 = QCIF, 1 = CIF size video
17	SUB_SAMPLE	1 = 4:2:2, 0 = 4:2:0 colour subsampling
18	YUV_ORDER	0 = YUYV, 1 = UYVY byte order
....		
24	COL_START	Capture ROI left edge
25	COL_END	Capture ROI right edge
26	ROW_START	Capture ROI top edge
27	ROW_END	Capture ROI bottom edge
28	COMP_ENABLE	0 = Non compressed image, 1 = Compressed image
29	DECIMATION	0 = No decimation, 1 = Decimation enabled.
30	Y_THRESH	Y Threshold used during compression.
31	UV_THRESH	UV Threshold used during compression
32	SYSTEM_STATE	Copy of field in CameraStatus
33	GRAB_STATE	Copy of field in CameraStatus
34	STREAM_STATE	Copy of field in CameraStatus
35	FATAL_ERROR	Copy of field in CameraStatus
36	CMD_ERROR	Copy of field in CameraStatus
37	DEBUG_FLAGS	Copy of field in CameraStatus
38	CAMERA_STATE_7	Copy of field in CameraStatus
39	CAMERA_STATE_8	Copy of field in CameraStatus
40	CR_ACHIEVED	Compression ration achieved for frame
41	FR_ACHIEVED	Framerate achieved between consecutive upload requests
63	LAST_BYTE	

Table 4

3.9.2 Video Line Format

Each line has the following format.

3.9.2.1 Linelength

2 bytes, low byte first, that contain the number of bytes following this on the line. This number is the number of data bytes plus the EOL code.

3.9.2.2 Data Bytes

The actual video data samples. The number and format of the these vary with the video mode and compression options selected.

In YUV 4:2:2 mode there is 4 bytes for every 2 image pixels, in order YUYV.

In YUV 4:2:0 mode even lines have both Y and UV data as in 4:2:2, while odd lines have only Y data, i.e. there are 2 bytes per 2 pixels.

3.9.2.3 End Of Line

Each line is finished by a single 0xFD byte.

3.9.3 End of Frame

The value 0xFF is used as an End Of Frame code. Four contiguous 0xFF characters are sent at the end of the frame before the isochronous pipe is disabled. In the parallel port case after the four 0xFFs have been sent the image stream is replaced by a continuous stream of 0xFFs until the current transfer is finished.

See also notes on the detection of the EOF conditions in the **USB Image Stream** and **PP Image Stream** sections below.

3.10 USB Image Stream

The USB camera has an single Interface (Interface #1) which has a control endpoint, Endpoint 0 and an isochronous endpoint, Endpoint 1. There are four alternate settings of Interface #1 to allow the data payload of the isochronous endpoint to be varied. The data payload sizes are 960, 704, 448 and 0 bytes for Alternate setting #3 to Alternate setting #0 respectively. (An alternate set of Alternate settings will also be provided via a hardware modification as a backup position. These data payload sizes are 896, 616, 320 and 0 bytes for Alternate setting #3 to Alternate setting #0 respectively).

A short data packet and/or a zero byte data packet are sent at the end of the video frame but can also be sent mid-way through a video frame in the case where the image is highly compressed and the Data Streamer catches up with the Compression Engine. In order to allow the USB minidriver to recognise the end of video frames without decompressing the image data stream the following approach is proposed:

The USB minidriver queues a URB to the isochronous pipe by specifying the number of USB frames (uframes) to which the URB relates and passing a pointer to an image buffer large enough to hold the data (MaxPacketSize * uframes). The USB_SHORT_TRANSFER_OK transfer flag parameter of the URB is not set. When a short or zero byte data packet is received then this is flagged in the status parameter of the URB. After the specified number of uframes the URB is returned with status information provided relating to each uframe packet indicating its offset from the beginning of the image buffer, the number of bytes transferred and any error status. Therefore the driver can determine in which uframe a short or zero byte transfer occurred. The driver can then check the short data packet to determine if it ended with four 0xFF characters or in the case of no short data packets being returned the driver could go to the last MaxPacketSize packet preceding the first zero byte data packet to determine if that transaction ended with four 0xFF characters. If this is the case then the driver has determined the location of the end of the video frame, otherwise the driver has not determined the end of the video frame but has simply found a pause in the image data stream.

3.11 PP Image Data Stream

3.11.1 Description of ECP / Nibble Upload Modes.

In the parallel port case after the four 0xFFs have been sent the image stream is replaced by a continuous stream of 0xFFs until the current transfer is finished.

3.12 Flicker Control

Flicker Control is an AEC algorithm which avoids the exposure values that result in flicker being seen on the video. Digital channel compensation gains are used to adjust the brightness of the image between allowed exposure values. Some work is required on the part of the driver to enable flicker control correctly, and guidelines for this are detailed below. Version 1-02 of the firmware has a very basic implementation of the AntiFlicker algorithm and so requires considerably more work on the part of the driver to achieve a result which is still inferior to the 1-03 implementation.

3.12.1 Using the FlickerControl Command

The Flicker Control command takes an exposure jump parameter, which is set to a value dependent on the sensor framerate and the local mains frequency. These are listed in the table below.

Mains Frequency (Hz)	Sensor Rate (fps)	Exp Jump (coarse)
50	30	92
50	25	76
50	15	46
50	12.5	38
50	7.5	23
50	6.25	19
60	30	76
60	25	64
60	15	38
60	12.5	32
60	7.5	19
60	6.25	16

Table 5

3.12.2 Setting Exposure

Exposure must be set correctly with the SetExposure command immediately after enabling Flicker Control. *NewExposure* is calculated as follows:

$$NewExposure = (ExpJump \times n) - 1$$

where

$$n = 1, 2, \ldots, K$$

and

$$(ExpJump - 1) \leq NewExposure \leq CurrentExposure$$

If however the *CurrentExposure* is lower than *(ExpJump-1)* then *NewExposure* should be set to *(ExpJump-1)*. This may cause temporary overexposure of the image but ensures flicker control is initialised correctly. The current exposure can be found either with the GET_EXPOSURE command or by multiplying the ErrorCode byte of the frame header (returned when an UploadFrame command is sent) by 2.

3.12.3 Colour steps

Set the VL_OFFSET to 20 (or any of 16 to 20) to avoid colour steps when exposure jumps are encountered. This is believed to be due to the black level of the sensor drifting with exposure. Changing the black level has an effect on the colour of the image.

3.12.4 Compensation Gains

Channel comp gains can be increased to compensate for any reduction of exposure. These are calculated as follows:

$$CompGain = \left((BaseComp - 128) \times \frac{CurrentExposure}{NewExposure}\right) + 128$$

Where BaseComps are listed below:

Channel	BaseComp
Red	(BYTE)(92 + 128) = 220

Table 6

Channel	BaseComp
Green 1	(BYTE)(86 + 128) = 214
Green 2	(BYTE)(86 + 128) = 214
Blue	(BYTE)(102 + 128) = 230

Table 6

This mechanism should only be used when the new exposure is lower than the old exposure however, and the comps should be left intact or rewritten at their base values when the new exposure is higher than the old one. Under no circumstances should the comp gains be written with values which represent lower gains than the Base Comps listed in the table above, i.e. the value of (BYTE)(NewComp-128) should be greater than (BYTE)(BaseComp-128).

3.12.5 Disabling Flicker Control

Comp gains should be set back to their base values listed in Section 3.12.4, Flicker Control.

3.12.6 Outstanding Limitations

Switching flicker control on or off can cause temporary changes in the brightness of the image before flicker control settles. This is a minor issue as flicker control is intended to be left either on or off for the whole time video is displayed.

3.13 Flicker Control for Firmware Version 1-02

There are several shortcomings in the flicker control in version 1.02 of the firmware. This section covers extensions to the driver to overcome these problems. If you only expect to deal with firmware version later than 1.02, it is not necessary to implement the algorithms described in this section.

3.13.1 Flicker Control Auto-Disable

The Flicker Control AEC algorithm cannot deal with very bright scenes because the minimum allowable exposure is large. To ensure these scenes do not become over-exposed, Flicker control in version 1-02 of the CPIA firmware disables itself when it has reached the minimum allowable exposure and comp. gain values, and the exposure accumulator is higher than the upper'dead band' limit plus the value of the allowable over-exposure parameter. The host should monitor exposure to determine when Flicker Control has disabled itself, which is evident when the exposure is not equal to one of the allowed values for the current mains frequency and sensor framerate (see calculation for new exposure in Section 3.12.1). Flicker Control can be re-enabled by the host when the exposure used by the regular AEC algorithm becomes higher than the minimum exposure allowed by Flicker Control.

3.13.2 Compensation Gains

Unlike with later versions of the firmware, the compensation gains must be left untouched. The algorithm uses a'min_comp' variable to decide when exposure steps should be made, and this variable is not set when new comps are downloaded, resulting in illegal comp values being used in some situations.

3.13.2.1Colour Jumps (not implemented)

The AntiFlicker algorithm moves all comp. gains equally in version 1-02 of the CPIA firmware and so the relative strengths of these gains are not maintained, causing change in the colour of the image when the'exposure' changes. Where brightness moves slowly, this is compensated for by the colour balance

algorithm, but jumps become obvious when an exposure step is encountered and the gains are shifted by a large amount.

The host can compensate for this by adjusting the colour balance when an exposure step is detected. The host already monitors current exposure to determine when Flicker Control has been disabled, and this value can also be used to detect when an exposure jump has been made. To compensate, find the current colour balance gains with the GetColourBalance command, and calculate the new colour balance gains as below.

$$NewRGBgain = MinRGBgain + (RGBgain - MinRGBgain) \times \frac{OldExposure}{NewExposure}$$

3.13.2.2Outstanding limitations

Oscillation between the lowest 2 exposure values occurs at 7.5fps. As a result flicker control also fails to disable itself in bright scenes at 7.5fps and should not be used.

An exposure'flash' can sometimes be seen as flicker control disables itself when the scene is too bright.

3.14 Optimising Performance of CPIA-Based Cameras

3.14.1 Compression Control

Use function SetCompressionTarget as an interface to the CPIA command SetCompressionTarget. This function implements a two mode control of compression targets. One for video-conferencing apps and one for local display apps. In both cases the "user" can alter the operation of delivered compression vis one slider-type control - this in effect gives a range of compression performance from high framerate to high quality.

3.14.2 CPIA Compression Algorithm Control Parameters

The Compression Algorithm implemented within CPIA has a number of critical constants, these constants can be altered by use of the CPIA command SetCompressionParams. Do not mess with this. All constants have default values however some of these have been improved upon. I suggest the following call just after camera initialisation:

```
CPIA_SetCompressionParams(3,11,1,3,2,5,3,2);
```

The only parameter altered from the defaults here is Hysteresis - the first parameter.

3.14.3 Improving (reducing) CPU Utilisation

CPU utilisation can be reduced by careful use of the StreamStartLine parameter of CPIA commands Grab and InitStreamCap. I suggest the following values for StreamStartLine:

CIF 120

QCIF 60

The function takes the parameter as half the index of the line as the point to start (or indicate the start) streaming of a frame. So the above values mean that CIF will start to stream at line 240 (out of 288) and QCIF will stream at 120 (out of 144). When these values were chosen to replace the default value of line 0, cpu utilisation was reduced by ~15% (for PPC2). The reason for this, is that this reduces potential upload stall time when streaming. Upload has less chance of being stalled by the grab because upload is not kicked

off until a large proportion of the grab has completed. When stalled PPC2 upload will still continue to eat cpu. So reduction in likelihood to stall reduces likelihood of cpu waste.

3.14.4 Image Sharpness

Image sharpness (and noise) is affected by values of Apcor and Apthresh. These values can be set using the CPIA command SetApcor. I suggest the following just after camera initialisation:

```
static apcorthresh[4] = {0x1C,0x1A,0x2D,0x2A};
        CPIA_SetApcor(apcorthresh[0],apcorthresh[1],apcorthresh[2],apcorthresh[3]);
```

This sets apcor and apthresh to the following:

	0	1	2	3
apcor	12	10	14	10
apthresh	1	1	2	2

3.15 Miscellaneous Considerations

3.15.1 PPC2 FatalError on Power-up

On the PPC2, parallel port only, product the 48Mhz crystal is not populated on the PCB.

On power-up, or after a watchdog reset, the firmware detects this and sets two flag bits in the FatalError field of the camera status structure.

Host software/drivers for the PPC2 product should therefore manually clear down these flag bits as part of the initialisation procedure. The ModifyCameraStatus command

should be used as follows to perform this.

ModifyCameraStatus(2, ~(XXX_COM_FLAG | CPIA_FLAG), 0);

How does the host software/driver know a PPC2 product is plugged in as opposed to a DUAL camera ? --- Well if exactly the two above noted flag bits are set after power-up it is fairly certain that the product is a PPC2, although it could also be a DUAL with a faulty 48 MHz clock.....

NOTE -Ver 1-20 of the firmware corrects this problem.

3.15.2 StreamState not reset on Hi->Lo->Hi sequence

In Ver 1-02 firmware the StreamState system variable is not reset correctly on a Lo->Hi power transition. Therefore if an error occured while streaming data, a Hi->Lo->Hi power sequence will not allow the stream to be restarted.

Solution is to manually reset the StreamState variable to STREAM_NOT_READY before the transition to Hi Power.

NOTE - This problem fixed in Ver 1-20 firmware.