VISION CPiA Data Sheet (VV0670P001)

Colour Processor Interface ASIC

Revision History

REV	Change	Revised by	Date
1	Original Document	Ed Duncan	4/12/97
2	First Body Text Added	Peter Slawek	09/02/98
3	Amendments & added prelim VP section	Ed Duncan	12/02/98
4	Restructure and add DRAM timing	Ed Duncan	26/03/98
5	Substantiated - more sections, text and diagrams	Ed Duncan	01/03/98
6	Power management + compression + mech drawings	Ed Duncan	03/03/98
7	Add REM's parallel port section	Ed Duncan	10/04/98
8	Extended CP section	REM	13/04/98
9	Add USB section and proof read document	Linda Russell	02/07/98

VISION CPiA

Colour Processor Interface ASIC

DATASHEET

FEATURES

- Interfaces to Vision's VV6404 colour CMOS image sensor
- Directly interfaces to a host PC using USB or Parallel port interfaces
- 100% compliant with USB specification v1.0
- 100% compliant with IEEE1284 parallel port specifications
- Support for Windows 95/98 Plug and Play
- Image capture at CIF (352x288) and QCIF(176x144) resolutions
- Image compression providing up to 30fps CIF
- Automatic exposure control and colour balance
- Camera flicker control for 50 Hz and 60 Hz mains driven lighting
- Supports selective transfer of requested regions of interest to the host system
- Power management including watchdog reset
- Absolute minimum of support components
- Device pinout designed to facilitate layout and area of support PCB

GENERAL DESCRIPTION

The Colour Processor Interface ASIC (CPiA) is a digital video processor requiring only a single 4Mbit DRAM and minimum of passive support components to provide a complete PC peripheral video capture system.

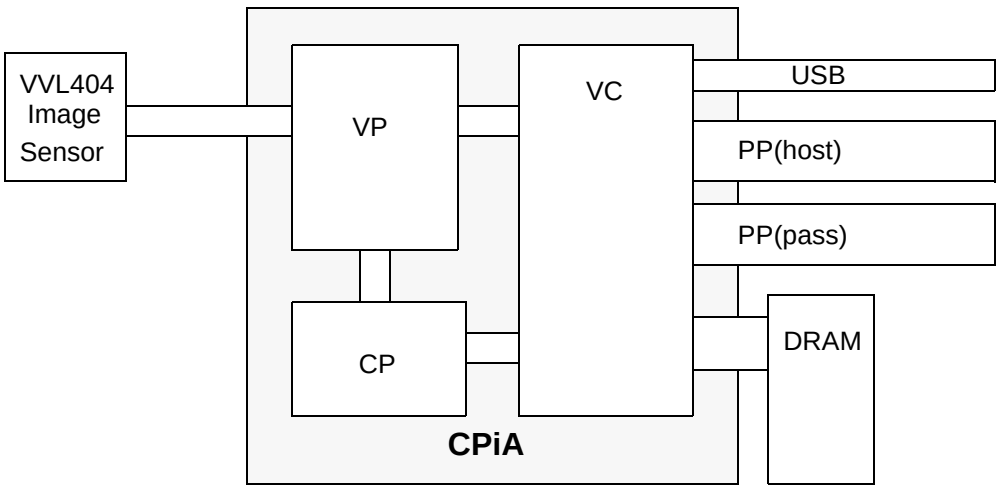
CPiA accepts raw digital video data from Vision's CIF format CMOS sensors and is capable of transferring the resulting YUV video data to a host PC over either USB or Parallel Port interfaces at rates up to 30 frames per second.

The CPiA architecture consists of three conceptually separate functional blocks: the Video Processor (VP), the Video Compressor (VC) and the Control Processor (CP).

The VP controls the VVL404 sensor and processes the raw pixel data into CIF or QCIF YUV images for compression and transfer to the Host by the VC.

The CP is responsible for coordinating system operation, responding to host requests and commands as well as performing automatic camera exposure control and colour balance.

CPiA SYSTEM BLOCK DIAGRAM

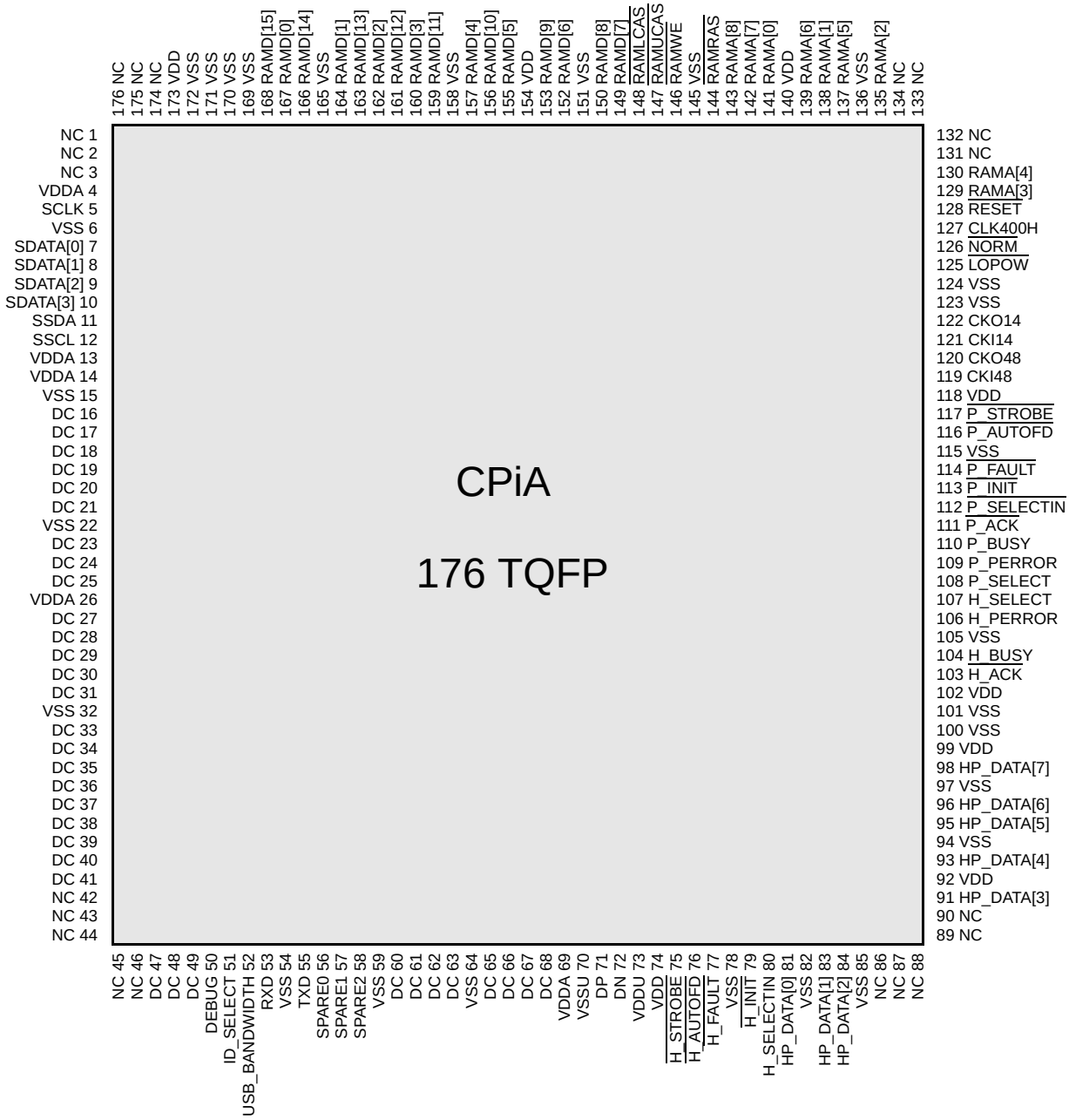


Commercial In Confidence

1.1 Representative Device Pinout

CPiA has been packaged in an industry-standard 176 TQFP. The pinout has been carefully developed to minimise the physical size of the support printed circuit board by facilitating placement of and electrical routing to peripheral support components such as DRAM and parallel port connectors.

Reference should also be made to drawings located at the end of this data sheet for physical dimensions and other, more detailed mechanical information.



1.2 CPiA Pin Description

1.2.1 Device I/O

Pin	Signal	Type	Description	drive
			Video Processor	
5	SCLK	O (4)	VVL6404 Sensor Clock (7.159MHz)	2 mA or 4mA
7	SDATA[0]	I (3)	VVL6404 Sensor Data Bus	-
8	SDATA[1]	I (3)	VVL6404 Sensor Data Bus	-
9	SDATA[2]	I (3)	VVL6404 Sensor Data Bus	-
10	SDATA[3]	I (3)	VVL6404 Sensor Data Bus	-
11	SSDA	I/O (2,3,6)	VVL6404 Sensor Serial Interface Data	2 mA
12	SSCL	I/O (2,3,6)	VVL6404 Sensor Serial Interface Clock	2 mA
			Control Processor	
50	$\overline{\text{DEBUG}}$	I	Control Processor Debug Output Select	-
51	$\overline{\text{ID\_SELECT}}$	I	DUAL Camera Device ID Select	-
52	$\overline{\text{USB\_BANDWIDTH}}$	I	Alternate USB Bandwith Settings	-
53	RXD	I	Control Processor Serial Communications	-
55	TXD	O(4,6)	Control Processor Serial Communications	2 mA
56	SPARE0	I	Control Processor spare (P1.1)	-
57	SPARE1	I	Control Processor spare (P3.4)	2 mA
58	SPARE2	I	Control Processor spare (P3.5)	2 mA
			USB	
71	DP	I/O (1,5,7)	USB Differential Output (+)	-
72	DN	I/O (1,5,7)	USB Differential Output (-)	-
			Parallel Port Interface	
75	$\overline{\text{H\_STROBE}}$	I (1,4,9)	Parallel port HOST STROBE control	-
76	$\overline{\text{H\_AUTOFD}}$	I (1,4,9)	Parallel port HOST AUTOFEED control	-
77	$\overline{\text{H\_FAULT}}$	O (5,9)	Parallel port HOST FAULT status	16 mA
79	$\overline{\text{H\_INIT}}$	I (1,4,9)	Parallel port HOST INIT control	-
80	$\overline{\text{H\_SELECTIN}}$	I (1,4,9)	Parallel port HOST SELECTIN control	-
81	HP_DATA[0]	I/O (1,4,5,9)	Parallel port Host/Pass data bus	16 mA
83	HP_DATA[1]	I/O (1,4,5,9)	Parallel port Host/Pass data bus	16 mA
84	HP_DATA[2]	I/O (1,4,5,9)	Parallel port Host/Pass data bus	16 mA
91	HP_DATA[3]	I/O (1,4,5,9)	Parallel port Host/Pass data bus	16 mA
93	HP_DATA[4]	I/O (1,4,5,9)	Parallel port Host/Pass data bus	16 mA
95	HP_DATA[5]	I/O (1,4,5,9)	Parallel port Host/Pass data bus	16 mA
96	HP_DATA[6]	I/O (1,4,5,9)	Parallel port Host/Pass data bus	16 mA
98	HP_DATA[7]	I/O (1,4,5,9)	Parallel port Host/Pass data bus	16 mA
103	$\overline{\text{H\_ACK}}$	O (5,9)	Parallel port HOST ACK status	16 mA
104	H_BUSY	O (5,9)	Parallel port HOST BUSY status	16 mA
106	H_PERROR	O (5,9)	Parallel port HOST PERROR status	16 mA
107	H_SELECT	O (5,9)	Parallel port HOST SELECT status	16 mA
108	P_SELECT	I (1,4,9)	Parallel port PASS-THRU SELECT status	-
109	P_PERROR	I (1,4,9)	Parallel port PASS-THRU PERROR status	-
110	P_BUSY	I (1,4,9)	Parallel port PASS-THRU BUSY status	-

Pin	Signal	Type	Description	drive
111	$\overline{\text{P\_ACK}}$	I (1,4,9)	Parallel port PASS-THRU ACK status	-
112	$\overline{\text{P\_SELECTIN}}$	O (5,9)	Parallel port PASS-THRU SELECTIN control	16 mA
113	$\overline{\text{P\_INIT}}$	O (5,9)	Parallel port PASS-THRU INIT control	16 mA
114	$\overline{\text{P\_FAULT}}$	I (1,4,9)	Parallel port PASS-THRU FAULT status	-
116	$\overline{\text{P\_AUTOFD}}$	O (1,5,9)	Parallel port PASS-THRU AUTOFEED control	16 mA
117	$\overline{\text{P\_STROBE}}$	O (1,5,9)	Parallel port PASS-THRU STROBE control	16 mA
			VC Master Clocks	
119	CKI48	I(8)	48.0MHz Oscillator Amplifier Input Pad	-
120	CKO48	O(7)	48.0MHz Oscillator Amplifier Output Pad	
121	CKI14	I	14.318MHz Oscillator Amplifier Input Pad	-
122	CKO14	O	14.318MHz Oscillator Amplifier Output Pad	
			Power Management Interface	
125	$\overline{\text{LOPOW}}$	O(7)	External device VCC control	2 mA
126	$\overline{\text{NORM}}$	O(7)	External device VCC control	2 mA
127	CLK400H	I (4)	Power management clock	-
128	$\overline{\text{RESET}}$	I (4)	Master Reset	-
			System Memory (DRAM) Interface	
129	RAMA[3]	O (10)	DRAM multiplexed address	2 mA
130	RAMA[4]	O (10)	DRAM multiplexed address	2 mA
135	RAMA[2]	O (10)	DRAM multiplexed address	2 mA
137	RAMA[5]	O (10)	DRAM multiplexed address	2 mA
138	RAMA[1]	O (10)	DRAM multiplexed address	2 mA
139	RAMA[6]	O (10)	DRAM multiplexed address	2 mA
141	RAMA[0]	O (10)	DRAM multiplexed address	2 mA
142	RAMA[7]	O (10)	DRAM multiplexed address	2 mA
143	RAMA[8]	O (10)	DRAM multiplexed address	2 mA
144	RAMRAS_n	O (10)	DRAM row address strobe	8 mA
146	RAMWE_n	O (10)	DRAM data write enable	2 mA
147	RAMUCAS_n	O (10)	DRAM upper column address strobe	8 mA
148	RAMLCAS_n	O (10)	DRAM lower column address strobe	8 mA
149	RAMD[7]	I/O (10)	DRAM data	2 mA
150	RAMD[8]	I/O (10)	DRAM data	2 mA
152	RAMD[6]	I/O (10)	DRAM data	2 mA
153	RAMD[9]	I/O (10)	DRAM data	2 mA
155	RAMD[5]	I/O (10)	DRAM data	2 mA
156	RAMD[10]	I/O (10)	DRAM data	2 mA
157	RAMD[4]	I/O (10)	DRAM data	2 mA
159	RAMD[11]	I/O (10)	DRAM data	2 mA
160	RAMD[3]	I/O (10)	DRAM data	2 mA
161	RAMD[12]	I/O (10)	DRAM data	2 mA
162	RAMD[2]	I/O (10)	DRAM data	2 mA
163	RAMD[13]	I/O (10)	DRAM data	2 mA
164	RAMD[1]	I/O (10)	DRAM data	2 mA
166	RAMD[14]	I/O (10)	DRAM data	2 mA
167	RAMD[0]	I/O (10)	DRAM data	2 mA

Pin	Signal	Type	Description	drive
168	RAMD[15]	I/O (10)	DRAM data	2 mA

If the target application requires that only one interface mode is required (eg. USB interface only), please refer to the above table and notes 7, 8 and 9 to establish how unused pins should be electrically connected on the PCB. Failure to ensure such pins are correctly connected may result in erroneous system behaviour.

Notes:

1. Internal pullups.
2. These pins shall go tri-state when $\overline{\text{LOPOW}}$ is high.
3. TTL level Schmitt input.
   $V_{IL(max)}$= 0.8V, $V_{IH\ (MIN)}$= 2.2V, schmitt trigger hysteresis=250mV (typ).
   Input leakage -10µA to +10µA, V_{IN} : 0 to DVDD
4. Output
   $V_{OL(max)}$ =0.4V @ I_{OL} =16mA
   $V_{OH(min)}$ =2.4V @ I_{OH} = -12mA
5. These signals to comply with USB Specification Version 1.0 section 7.
6. Open drain output.
   $V_{OL(max)}$ =0.4V @ I_{OL} =3mA
7. This pin does not require connection if CPiA is used for parallel port interfacing products only.
8. This pin should be tied low if CPiA is used for parallel port interfacing products only.
9. This pin does not require connection if CPiA is used for USB interfacing products only.
10. These pins shall go tri-state when $\overline{\text{NORM}}$ is high.

Where I/O pad type is not explicitly defined in the above notes assume CMOS.

1.2.2 Device Power, NC and DC pins

Pin	Signal	Description
6, 15, 22, 32, 54, 64, 78, 82, 85, 94, 100, 101, 105, 115, 124, 136, 145, 151, 158, 165, 97	VSS	Digital padring & logic ground.
59, 123, 169, 170, 171, 172	VSS	Reserved Signal Inputs (Tie Low)
4, 13, 14, 26, 69	VDDA	Digital padring & logic power for VP & CP
74, 92, 99, 102, 118, 140, 154, 173	VDD	Digital padring & logic power for VC
70	VSSU	Ground for the USB
73	VDDU	3.3V supply for the USB
16, 17, 18, 19, 20, 21, 23, 24, 25, 27, 28, 29 , 30, 31, 33, 34, 35,36, 37, 38, 39, 40, 41, 47, 48, 49	DC	Do NOT connect these pins
1, 2, 3, 42, 43, 44, 45, 46, 86, 87, 88, 89, 90, 131, 132, 133, 134, 174, 175, 176	NC	Not connected pins
60, 61, 62, 63, 65, 66, 67, 68	DC	Do NOT connect these pins

1.3 Absolute Max Ratings

Description	Range	Unit
Ambient Temperature	0 to 40	^{o}C
Storage Temperature	-50 to 150	^{o}C
Voltage on USB D+/D-, Vcc3V to Vss		V
Voltage on any other I/O to Vss		V

1.4 DC Characteristics

Parameter	Description	Min	Max	Units
VDD	Primary CPiA Power Supply			V
VDDA	Secondary CPiA Power Supply			V
VDDU	3.3V Power Supply for on-chip USB tranceiver			V
Isuspend	CPiA suspend mode current			uA
Ilowpo	CPiA low power modecurrent			mA
Iactive	CPiA active, high power mode current			mA
Vilu, Vihu	USB differential pad D+/D- Vil, Vih	See Section 1.2		V
Volu, Vohu	USB differential pad D+/D- Vol, Voh			V
Iilu, Iihu	USB differential pad D+/D- Iil, Ili			mA
Vilp, Vihp	Parallel Port pads Vil, Vih			V
Volp, Vohp	Parallel Port pads Vol, Voh			V
Iilp, Iihp	Parallel Port pads Iil, Ili			mA
Vild, Vihd	Dram Interface pads Vil, Vih			V
Vold, Vohd	Dram Interface pads Vol, Voh			V
Iild, Iihd	Dram Interface pads Iil, Ili			mA
Vilm, Vihm	Power Management pads Vil, Vih			V
Volm, Vohm	Power Management pads Vol, Voh			V
Iilm, Iihm	Power Management pads Iil, Ili			mA

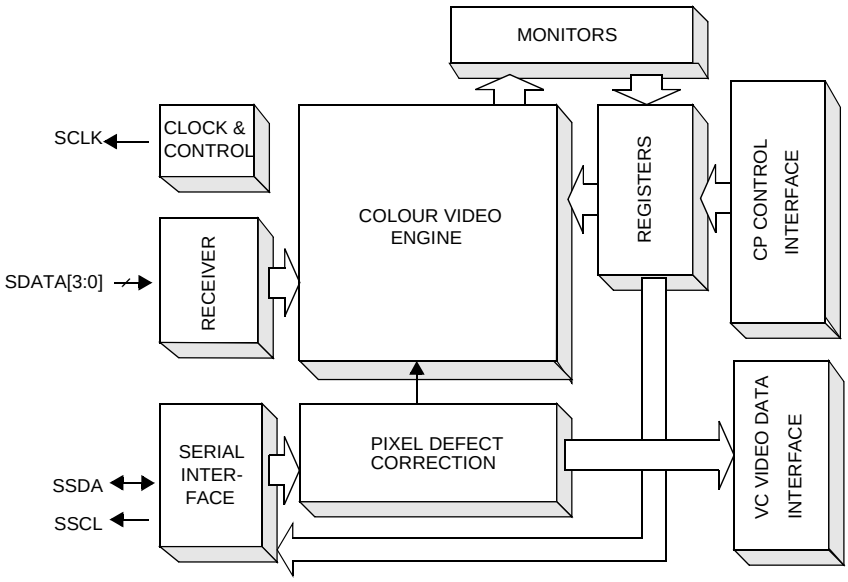
2 CPiA Functional Description

2.1 Video Processor Module

The CPiA video processor module provides CIF-format 4:2:2-sampled digital video to the VC module at frame rates up to 30 frames per second and also interfaces directly to the 356 x 292 pixel colour CMOS image sensor provided in Vision's VM6404 camera head module. Using a 9-wire cable and an absolute minimum of external components, the interface incorporates:

1. A 4-wire data bus SDATA[3:0] for receiving both video data and embedded timing references.
2. A 2-wire serial interface SSDA,SSCL for controlling the camera.
3. The clock for camera module SCLK.
4. 5V and 0V power lines. CPiA is not required to provide power directly to the camera; camera power is derived from the system power supply.

The simplified block diagram shown below highlights the key functional blocks within CPiA's VP module.



Block Diagram of CPiA Video Processor Module

CPiA provides a master clock SCLK to the camera module. Each 8-bit pixel value generated by the camera is transmitted across the 4 wire databus SDATA[3:0] as a pair of sequential 4-bit nibbles, most significant nibble first. Codes representing the start and end frames and the start and end of lines are embedded within the video pixel data stream to allow the receiver to synchronise with the video data which the camera module is generating.

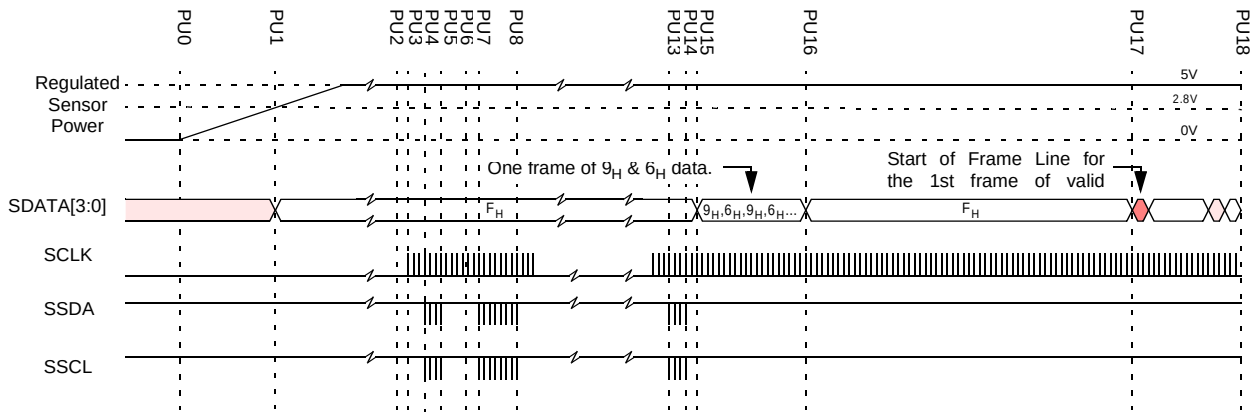
The video processing engine performs these basic functions on incoming data: full colour restoration at each pixel site from Bayer-patterned input data, matrixing/gain on each colour channel for colour purity, aperture correction for image clarity, gamma correction, and colour space conversion (including hue and saturation control) from RGB to YCbCr.

Image statistic monitors gather data required by the CP module for end-of-frame housekeeping tasks such as exposure control and colour balance.

The 2-wire camera serial interface provides complete control over how the sensor is setup and run.

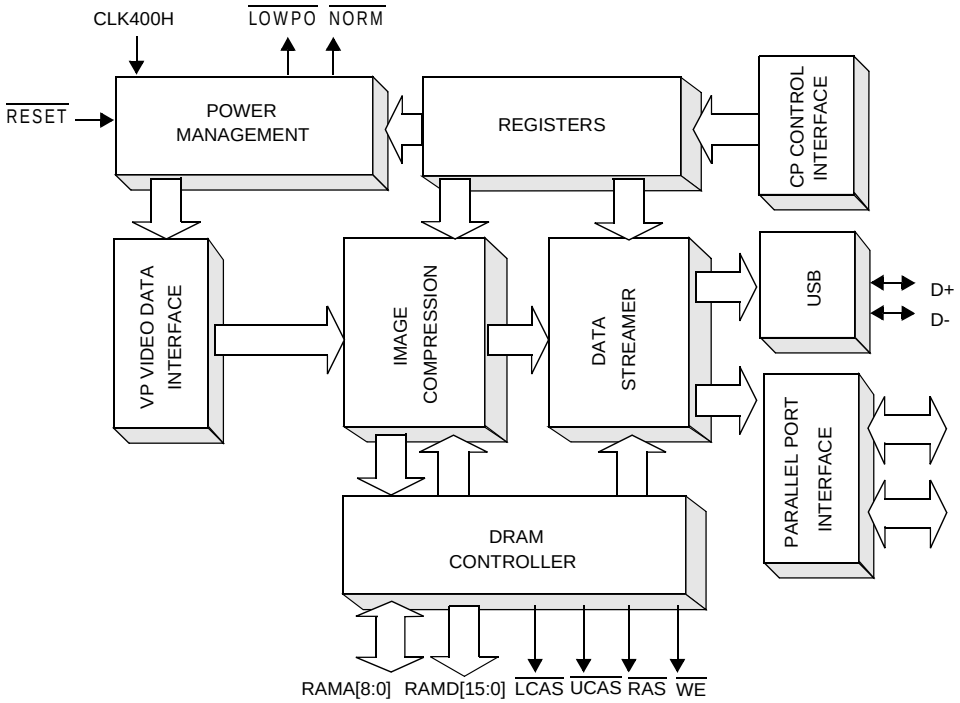
Camera exposure and gain values are programmed via this interface.
The following table and diagram illustrate the camera power up sequence.

PU0	System Power Up
PU1	Sensor Internal-on Reset Triggers, the sensor enters low power mode and SDATA[3:0] is set to F_H.
PU2	CPiA internally releases video processing modules from reset. NOTE - this event is under the control of the Control Processor, and does not occur until the host has requested video from the camera.
PU3	CPiA enables the sensor clock, SCLK.
PU4-PU5	At least 16 SCLK clock periods after SCLK has been enabled CPiA sends a “Soft-Reset” command to the sensor via the serial interface. This ensures that if a sensor is present then it is in low-power mode.
PU6	On detecting 32 consecutive F_H values on SDATA [3:0], CPiA detects the camera
PU7-PU8	If present, CPiA uploads the sensor defect map from camera head E^2PROM.
PU13-PU4	At least 16 SCLK clock periods after SCLK has been enabled, CPiA sends an “Exit Low-Power Mode” command to the sensor via the serial interface. This initiates the sensors 4 frame start sequence.
PU15-PU16	One frame of alternating 9_H & 6_H data on SDATA[3:0] for the CPiA to determine the best sampling phase for the nibble data SDATA[3:0].
PU17-PU18	4 Frames after the “Exit Low-Power Mode” serial comms, the sensor starts to output valid video data.



Camera Head Interface Behaviour up to and including first valid video data

2.2 Video Compressor Module



Block Diagram of CPiA Video Compressor Module

2.2.1 Power Management

The power management block is primarily responsible for low-level control of system power, clocks and reset sequences to ensure full compliance with the USB specification and power saving modes of operation. This module also includes a watchdog reset ensuring, for example, the system always returns to a safe state if power-failure or any other event has caused CPiA to encounter an unknown and fatal error.

CPiA requires up to 3 independent power supplies. When CPiA is designed into a product providing full USB power consumption compliance, the system must provide CPiA with VDD=5V, a switched VDDA=5V and VDDU=3.3V (for internal USB differential pads). If CPiA is used in a target application where only the parallel port interface is required and slightly higher module power consumption is permitted, a single, constant 5V supply can be safely fed to VDD, VDDA and VDDU.

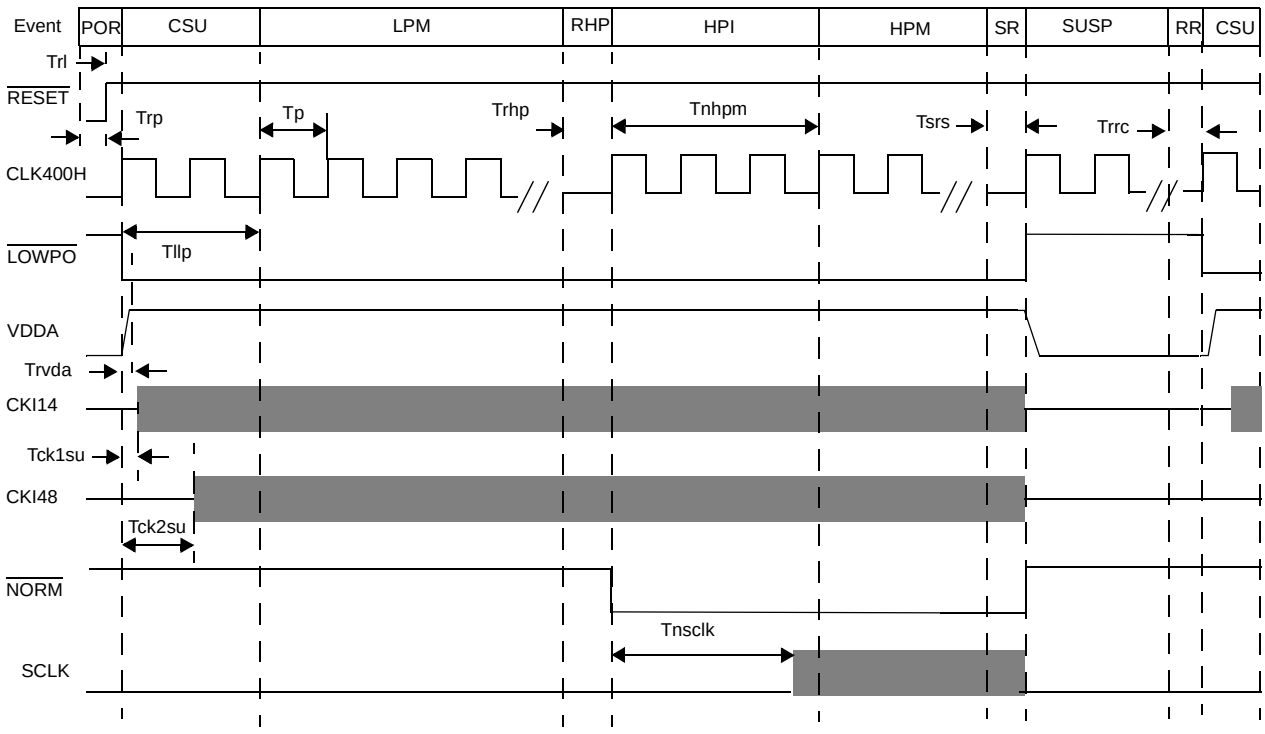
VDD and VDDU must always be permanently connected to system power supplies. The switched VDDA power line is typically supplied using a power FET external to CPiA. The power management module provides two outputs for driving the FET gates and hence is capable of enabling or disabling power to system-level components; LOPOW and NORM.

LOPOW is asserted when CPiA is ready for VDDA to be powered and once power has been applied puts CPiA into a state where commands received from the host PC can be processed. As well as VDDA becoming active, this also includes starting up of high-speed 14.318MHz and 48MHz clocks using external crystals, resetting and initialising all internal state machines and logic within VP, VC and CP modules. LOPOW is de-asserted at any time the host PC requests the module is either put into a USB SUSPEND condition or any other host-application-dependent mode when ultra low power consumption is required.

NORM is asserted when CPiA is ready to put the module into the highest power mode of operation which occurs when the camera becomes active and high speed image transfers via USB or parallel port interfaces commences. NORM is used internally to CPiA only but is provided as an optional output for driving customer-specific logic. For example, NORM can be used to illuminate an LED indicating the camera is active. NORM is also deasserted at any time the host requests USB SUSPEND mode.

Effect of LOPOW on CPiA-based module power consuption WITH EXTERNAL CONTROL OF VDDA using external FET. NORM is shown for reference only and is not required for power control using external FET.					
LOPOW	**NORM**	**Mode**	**Description**	**VDDA**	**Approx Module Current**
1	1	Suspend	CPiA, camera module and DRAM components are in lowest power mode. VDDA is withdrawn as external FET is off, all fast clocks are disabled and CPiA logic is in held reset	0V	<500uA
0	1	Low Power	CPiA is in low power mode. Fast clocks enabled and CPiA can process commands from host PC. Camera, video processing and DRAM controller modules are held in reset	5V	<50mA
0	0	High Power	All CPiA modules, camera and DRAM components are active and video data is being transferred to host PC.	5V	<250mA
If VDDA is not controlled using VDDA (VDD=VDDA=VDDU) in parallel port modes of operation, approx module power consumption will be <250mA at all times. In such cases, LOPOW and NORM can be considered redundant.					

CPiA's power management block requires two externally derived inputs: RESET provides a global reset to all CPiA logic and CLK400H provides a low frequency clock to CPiA's internal power control state machines and should be constrained in accordance with the timing diagrams presented below. The Events are also described in the table below. After event RR, the process returns to CSU.

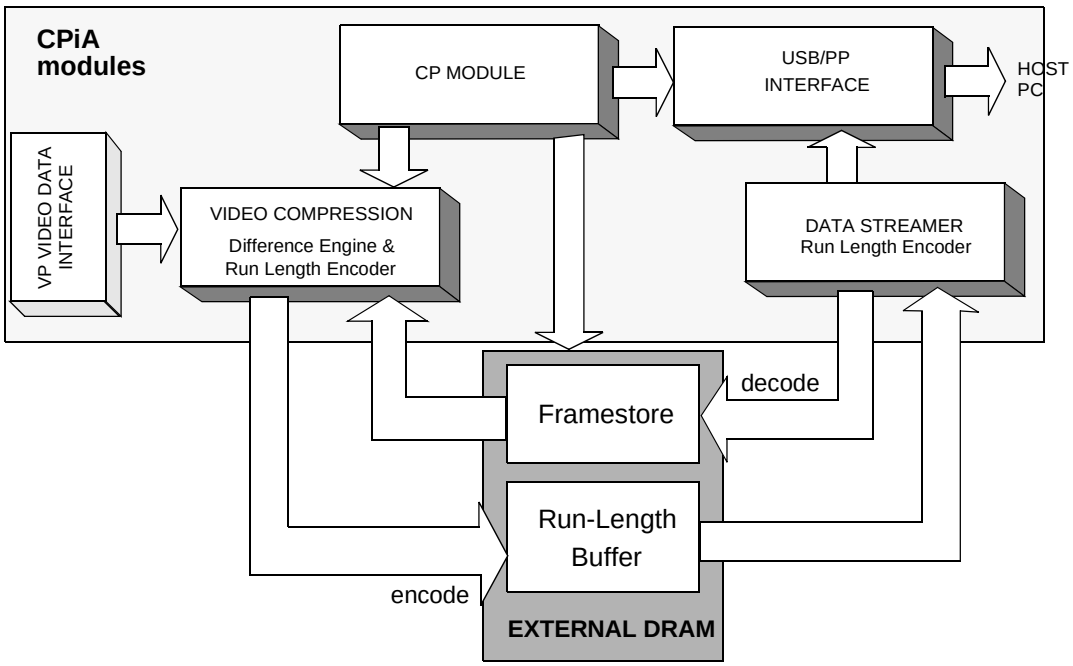


Timing Diagram Highlighting Power Management Events

Event	Description
POR	POWER ON RESET PROVIDED TO CPIA BY EXTERNAL HARDWARE
CSU	HIGH FREQUENCY CLOCK START UP
LPM	LOW POWER MODE OF CPIA OPERATION
RHP	HOST REQUEST FOR CPIA TO ENTER HIGH POWER STATE
HPI	INITIALISE HIGH POWER MODE OF OPERATION
HPM	HIGH POWER MODE OF CPIA OPERATION
SR	USB REQUEST POWER SAVING SUSPEND MODE OF OPERATION
SUSP	CPIA HELD IN USB SUSPEND MODE
RR	USB REQUEST RESUME TO RECOVER LOW POWER MODE OF OPERATION

Parameter	Description	min	max	Units
Trp	CPiA global $\overline{\text{RESET}}$ pulse width	0.1	-	ms
Tp	CPiA CLK400H period	2.25	2.75	ms
	CPiA CLK400 duty	20:80	80:20	-
Trl	Time for low power mode after $\overline{\text{RESET}}$ deasserted	0	Tp	ms
Tck1su	14.318MHz Clock start-up time	-	Tp	ms
Tck2su	48MHz Clock start-up time	-	(2*Tp)-1	ms
Trvda	VDDA Power Supply rise time from $\overline{\text{LOWPO}}$ asserted	-	Tp	ms
Tllp	Time from $\overline{\text{LOWPO}}$ asserted to low power mode		2*Tp	ms
Tnsclk	Time from $\overline{\text{NORM}}$ to camera interface active			ms
Tnhpm	Time from $\overline{\text{NORM}}$ to high power mode			ms
Trhp	Time to acknowledge host request for high power mode			ms
Tsrs	Time for CPiA to acknowledge and act upon USB suspend mode request			ms
Trrc	Time for CPiA to acknowledge and act upon USB resume low power mode request			ms

2.2.2 Video Compression



Video compression components and VC Module interfacing to External DRAM

In the discussion to follow the CPiA device is assumed to be interfaced to the the host PC via the USB, however the discussion equally well applies to the parallel port interface option that CPiA also provides.

The VP module provides the video data stream to VC. The pixel-based compression algorithm operates by considering the differences between consecutive frames and encoding those differences in a Run-Length Buffer (RLB) held in external DRAM. The result is a run-length encoded stream of the changes from frame-to-frame. The stream consists of run-lengths of regions where no change has occurred, coded YCbYCr (YUV422) values of pixel-pairs which have changed significantly, additional codes to mark the start/end/length of frame rows, a frame header detailing parameters associated with the capture settings of the encoded data and finally codes to mark the end of the encoded frame.

The VC module implements the compression algorithm conceptually by way of two processes. The first process is the Differencing Engine/Run-Length Encoder (DE/RLE):

- Each pixel-pair of the current video frame is compared in turn with the corresponding pixel-pair stored in the DRAM FrameStore (FS). If this comparison indicates that there exists a significant difference in value for the given pixel-pair then that pixel-pair's value is stored at the next locationin the RLB .The stored value is further coded to indicate its status as a pixel-pair value rather than a run-length . Conversely, if the comparison indicates that the difference is not significant, a run-length counter is incremented and stored at the current position in the RLB. The level of *significant* difference is dynamically calculated by the CP module from frame-to-frame and can be influenced by the host PC. When the DE/RLB finishes processing a frame the CP module is alerted indicating subsequent processes can commence.

The second process is the Data Streamer/Run-Length Decoder (DS/RLD):

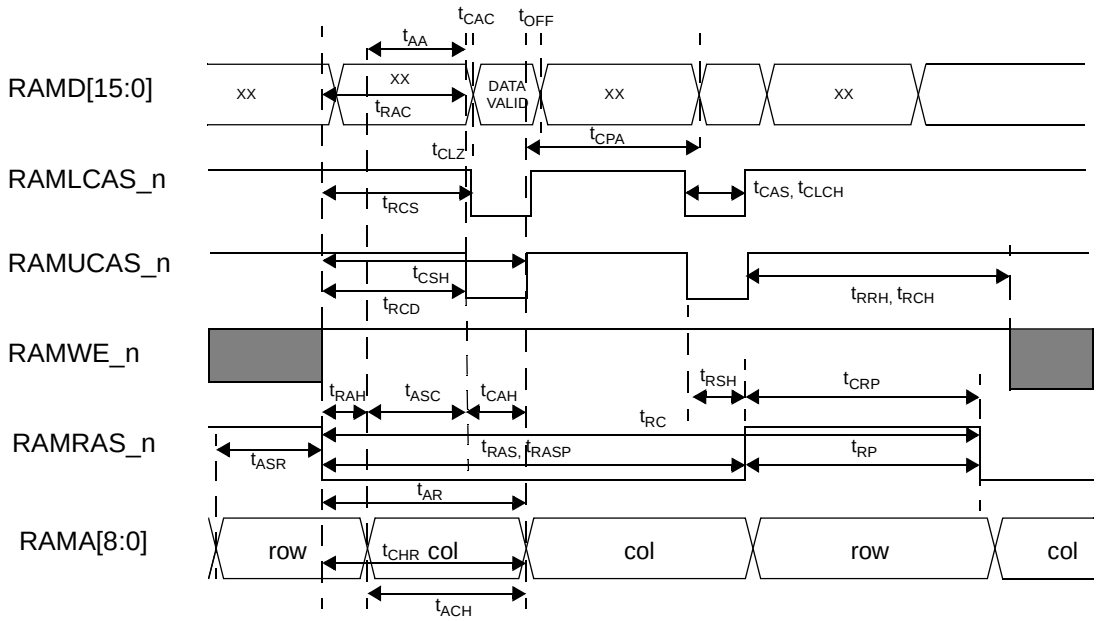
- In parallel with the DE/RLE but delayed with respect to the current position in the RLB, the Data Streamer (DS) streams data from the RLB to USB interface module for transfer to the host. In addition, the DS also decodes the RLB stream back into the FS thus updating it. The software driver also decodes the received RLB stream on the host PC. When the DS/RLB reaches the end of the data stream the CP module is again alerted and the process repeats.

It is important to note that the two processes can run concurrently, that is the DE/RLE can be encoding the current frame into the RLB while at the same time the DE/RLE can be streaming out the previously encoded frame to the USB.

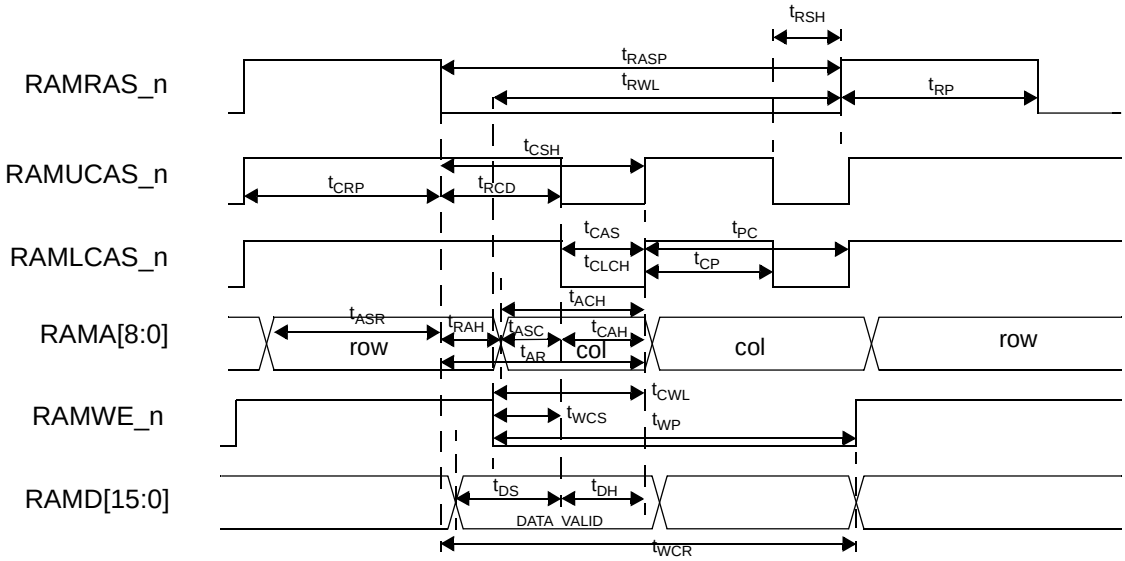
2.2.3 DRAM Interface

In order to perform video capture and compression, CPiA depends upon an a frame store provided by a single, low cost DRAM. The integral DRAM controller is designed to support standard 256K x 16 EDO devices, with access times better than or equal to 60ns. Timing diagrams showing bus read and write cycles are highlighted below. As seen in Section 2.2.2, the compression algorithm uses two main data structures within the DRAM:

- The Framestore holds a copy of the previous image uploaded to the host. The DE uses this information to determine significant differences between the FS image and the next image read from the VP.
- The Run Length Buffer stores the current frame in a run-length encoded form, ready for upload by the DS/USB to the host.



DRAM Read Cycle Timing



DRAM Write Cycle Timing

CPIA DRAM Interface Timing Parameters							
Parameter	DRAM requirement (ns) Note 1	CPiA DRAM Timing (ns) Note 2		Parameter	DRAM requirement (ns) Note 1	CPiA DRAM Timing (ns) Note 2	
		min	max			min	max
t_{AA}	>30	43	58	t_{CAS}	>10	28	40
t_{ACH}	>15	52	58	t_{CAH}	>10	28	40
t_{AR}	>45	68	72	t_{CHR}	>10	28	40
t_{ASC}	>0	17	21	t_{CLCH}	>10	28	40
t_{ASR}	>0	68	72	t_{CLZ}	0	0	0
t_{CAC}	>17	19	26	t_{CP}	>8	28	40
t_{CRP}	>5	68	72	t_{CPA}	>35	62	65
t_{CSR}	>5	28	40	t_{CSH}	>45	68	72
t_{DH}	>10	28	40	t_{CWL}	>10	52	57
t_{DS}	>0	28	40	t_{OFF}	$0<t_{OFF<}15$	5.5	6
t_{RAC}	>60	63	67	t_{PC}	>25	68	72
t_{RAS}	>60	68	72	t_{RAH}	>10	11	21
t_{RRH}	>0	68	72	t_{RASP}	>60	68	72
t_{RCD}	$14<t_{RCD<}45$	28	40	t_{RC}	>105	136	144
t_{RCS}	>0	28	40	t_{RCH}	>0	68	72
t_{RP}	>40	68	72	t_{REF}	<8ms	-	16us
t_{RWL}	>15	52	57	t_{RPC}	>5	28	40
t_{WCH}	>10	28	40	t_T	<2	0.5	2
t_{WCS}	>0	17	20	t_{WCR}	>45	68	72
t_{WP}	>5	52	57	t_{RSH}	>15	28	40

Note 1 - DRAM timing parameters extracted from DRAM manufacturer ‘s -6 worst-case spec data tables. Due to some variation in manufacturer’s figures it is recommended required values presented in this table are closely checked against the data tables from specific target DRAM manufacturers.

Note 2 - Min and Max timing derived from worst and best case design simulation wih respect to process parameter derating with supply voltage and Tj. All timing parameters listed have been verified and validated through device characterisation with Ta@25C, VDD=VDDA=5V.

Framestore and Run Length Buffer DRAM partitioning		
Data Structure	Size (bytes)	Size (Kbytes)
Framestore (FS)	202752	198
Run-Length Buffer (RLB)	205124	~201
Total (FS + RLB)	407876	~399

2.2.4 USB Interface

The USB interface is fully compliant with USB Specification Version 1.0. The USB interface is designed such that the camera attaches to the PC via a 4-wire USB cable, either directly at the PC motherboard USB connector (root hub) or via a USB hub (in turn connected to the PC motherboard root hub). The USB hub may be either a stand-alone USB hub device or a compound USB hub device incorporated, for example, in a USB monitor or printer.

USB transfers both signal and power over the 4-wire cable such that the camera is powered from the same cable/connector over which it communicates with the PC. The 4-wire cable carries VBus (nominally +5V at the source) and GND wires to deliver power to devices and differential data lines, D+ and D-. The clock is transmitted encoded along with the differential data. The clock encoding scheme is NRZI with bit stuffing to ensure adequate transitions. A SYNC field precedes each packet to allow the receiver(s) to synchronise their bit recovery clocks. CPiA provides fully compliant USB differential pads on-chip so there is no need to use an external, discrete bus tranceiver chip such as Philips' PDIUSBP11. In accordance with the USB Specification Version 1.0, the pads have been characterised for correct operation up to the maximum peripheral cable length of 5m.

2.2.4.1 CPiA-Based USB Camera

The CPiA-based USB camera is a high-power, bus-powered device. When it is first attached to a USB hub port it operates in its low-power mode consuming no more than 100mA. Once the camera has been successfully enumerated it may operate in its high-power mode in which it consumes no more than 400mA. Since the camera is a high-power, bus-powered device it will only enumerate successfully if connected to a self-powered USB hub. (A bus-powered USB hub is only capable of supplying 100mA at each of its downstream ports).

The CPiA-based USB camera responds to suspend signalling issued by the USB (more than 3ms of bus IDLE time). In its suspend mode of operation the camera draws no more than 500uA. The camera does not support remote wakeup.

The CPiA USB camera is a single Configuration, single Interface USB peripheral device. Its single Interface (Interface 1) supports Control Endpoint 0 and Isochronous IN Endpoint 1.

All devices must support Control Endpoint 0 as this is the endpoint with which the PC communicates over the default pipe at device enumeration time. CPiA utilises Control Endpoint 0 to respond to standard USB commands (such as those used at device enumeration time) and to Vendor specific CPiA commands which are used to control the operation of the camera and to request status information from the camera. Note that the camera does not support a USB Class Specification and so only Vendor commands are used to control the operation of the camera.

Isochronous IN Endpoint 1 is used to transfer video data from the camera to the PC. Using an isochronous endpoint for this task guarantees bandwidth and latency for the transfer of video data. The bandwidth requested for video data transfer can be varied by means of the four defined Alternate Settings of Interface 1.

2.2.4.2 USB Control Transfers

Control transfers are used to pass control information to the CPiA-based USB camera from the host and to request state information back from the camera. For example, control transfers are used at device enumeration time when the host PC requests information about the USB device which has been dynamically attached to the bus to enable it to load the appropriate driver(s). (USB Device Enumeration is discussed in detail in section **2.2.4.3**). CPiA uses Control Endpoint 0 for all control transfers. Therefore Endpoint 0 handles both standard USB commands and Vendor specific CPiA commands.

A control transfer consists of three phases: a SETUP phase, a DATA phase (should one exist) and a STATUS phase.

The SETUP phase contains details of the command being sent to the camera including, the specific command request, whether the command is a WRITE (host to camera) or READ (camera to host) command and the number of bytes expected in the DATA phase. (For further information on the SETUP phase please refer to USB Specification Version 1.0, Section 9.3).

The DATA phase is used to transfer a number of bytes of information associated with the command to the camera if this is a WRITE command or from the camera if this is a READ command. The DATA phase can consist of a number of discrete DATA transactions on the bus. With CPiA-based USB cameras, data is transfered in 8-byte packets with the remaining data bytes transfered in the last data transaction being less than 8 bytes if the total number of bytes to be transfered is not a multiple of 8. This is because the maximum packet size of Control Endpoint 0 in CPiA is 8 bytes.

The STATUS phase is used to indicate whether or not all was well with the SETUP and DATA phases and consequently whether or not the control transfer has completed successfully. If at any time the camera receives an unrecognised command (for example due to data corruption on the USB) or it cannot process a valid command for some reason it returns a STALL handshake on the bus to indicate to the host that something is wrong.

2.2.4.3 USB Device Enumeration

The USB hub detects that a high speed USB device has been dynamically attached to one of its downstream ports by the presence of a 1.5KOhm pull-up resistor on the D+ differential data line on the upstream port of the device, as is the case with a CPiA-based USB camera. Before the port to which the camera has been connected can be enabled the hub ensures that the device is reset. It does this by issuing SE0 reset signalling on the USB differential data lines for at least 10ms and then enables the port. This causes the camera to ‘listen’ to USB traffic at bus address 0 until such time as it is assigned its own unique USB bus address. The camera is ready to respond to host access within 10ms of this reset signalling being de-asserted. Note that the hub waits a specified delay time after device attach before issuing the reset signalling to ensure the device’s internal power supply has stabilised.

Now that the camera is listening at bus address 0 the host begins to communicate with the camera. The initial communications comprise the device enumeration traffic sequence. The host first issues a *GetDescriptor* (DEVICE) command (SETUP Packet 80 06 00 01 00 00 40 00) to Control Endpoint 0 (which all USB devices must support) at bus address 0. The host only receives the first 8 data bytes of the DATA phase of this control transfer (that is, the first DATA transaction within the DATA phase) before issuing the STATUS phase.

The host then issues a *SetAddress* command (SETUP Packet 00 05 yz wx 00 00 00 00 where wxyz is the USB device address) to assign a unique USB bus address to the camera. Following the STATUS phase of a successful *SetAddress* command the camera then ‘listens’ and responds to USB traffic at this unique address only.

The host continues by issuing a *GetDescriptor* (DEVICE) command (SETUP Packet 80 06 00 01 00 00 12 00). In response to the *GetDescriptor* (DEVICE) command the camera returns its DEVICE descriptor which comprises 18 bytes of information. The DEVICE descriptor describes general information about the USB device. The table below shows the DEVICE descriptor information returned by a CPiA-based USB camera in response to the *GetDescriptor* (DEVICE) command. This information can be checked at device enumeration time using a USB bus analyser such as the CATC Inspector when debugging a CPiA-based USB camera design.

Descriptor Field	Size (bytes)	Value	Description
bLength	1	0x12	Size of this descriptor in bytes.
bDescriptorType	1	0x01	DEVICE Descriptor Type.

Descriptor Field	Size (bytes)	Value	Description
bcdUSB	2	0x0100	USB Specification Release Number identifying the release of the USB Specification that the device and its descriptors are compliant with. (Release 1.00)
bDeviceClass	1	0x00	Class Code (assigned by USB). This field is set to 0x00 meaning that the class information is specified in the Interface descriptor.
bDeviceSubClass	1	0x00	Subclass Code (assigned by USB).
bDeviceProtocol	1	0x00	Protocol Code (assigned by USB). This field is set to 0x00, meaning that the device does not use class specific protocols on a device basis, however may use class specific protocols on an interface basis.
bMaxPacketSize0	1	0x08	Maximum packet size for Endpoint 0.
idVendor	2	0x0553	Vendor ID (assigned by USB).
idProduct	2	0x0002 (1)	Product ID (assigned by the manufacturer).
bcdDevice	2	0x0100	Device release number in binary-coded decimal.
iManufacturer	1	0x00 (2)	Index of string descriptor describing manufacturer.
iProduct	1	0x00 (2)	Index of string descriptor describing product.
iSerialNumber	1	0x00 (2)	Index of string descriptor describing the device's serial number.
bNumConfigurations	1	0x01	Number of possible configurations

Notes:

(1) The CPiA device returns a Product ID as part of its DEVICE descriptor. This Product ID is used along with the Vendor ID to load the approriate device drivers for the CPiA-based USB camera. The device can return two possible Product ID numbers depending on the state of the ID_SELECT pin. This pin has an internal pull-up resistor which, when not connected, provides a Product ID of 0x0002 for a USB-only CPiA-based camera module. When the ID_SELECT pin is pulled low externally, the Product ID will be 0x0001, which is used for a combined parallel port and USB module.

(2) The USB camera does not support String Descriptors. Any request by the host for a STRING descriptor will cause the camera to STALL the default pipe.

Finally the host issues a *GetDescriptor* (CONFIGURATION) command (SETUP Packet 80 06 xx 02 00 00 FF 00). CPiA has only one Configuration (index xx = 00). If xx is anything other than 00 CPiA will STALL the default pipe. In response to the *GetDescriptor* (CONFIGURATION) command the camera returns its Configuration, Interface and Endpoint descriptors which comprise 73 bytes of information.

The tables below show the descriptor information returned by a CPiA-based USB camera in response to the *GetDescriptor* (CONFIGURATION) command. This information can be checked at device enumeration time using a USB bus analyser such as the CATC Inspector when debugging a CPiA-based USB camera design.

Descriptor Field	Size (bytes)	Value	Description
bLength	1	0x09	Size of this descriptor in bytes.
bDescriptorType	1	0x02	CONFIGURATION Descriptor Type.

Descriptor Field	Size (bytes)	Value	Description
wTotalLength	2	0x0049	Total length of data returned for this configuration. Includes the combined length of all descriptors (configuration, interface, endpoint, and class or vendor specific) returned for this configuration.
bNumInterfaces	1	0x01	Number of interfaces supported by this configuration.
bConfigurationValue	1	0x01	Value to use as an argument to *SetConfiguration* to select this configuration.
iConfiguration	1	0x00 (1)	Index of string descriptor describing this configuration.
bmAttributes	1	0x80	Configuration characteristics: Bus Powered and no Remote Wakeup.
MaxPower	1	0xC8	Maximum power consumption of USB device from the bus in this specific configuration when the device is fully operational. Expressed in 2mA units (0xC8 = 400mA).

Descriptor Field	Size (bytes)	Value	Description
bLength	1	0x09	Size of this descriptor in bytes.
bDescriptorType	1	0x04	INTERFACE Descriptor Type.
bInterfaceNumber	1	0x01	Number of interface.
bAlternateSetting	1	0x00	Value used to select alternate setting for the interface identified in the prior field.
bNumEndpoints	1	0x01	Number of endpoints used by this interface (excluding Endpoint 0).
bInterfaceClass	1	0xFF	Class code (assigned by USB). A value of 0xFF means that the interface class is vendor specific.
bInterfaceSubClass	1	0x00	Subclass code (assigned by USB).
bInterfaceProtocol	1	0xFF	Protocol code (assigned by USB). A value of 0xFF means that the device uses a vendor specific protocol for this interface.
iInterface	1	0x00 (1)	Index of string descriptor describing this interface.

Descriptor Field	Size (bytes)	Value	Description
bLength	1	0x07	Size of this descriptor in bytes.
bDescriptorType	1	0x05	ENDPOINT Descriptor Type.

Descriptor Field	Size (bytes)	Value	Description
bEndpointAddress	1	0x81	The address of the endpoint on the USB device described by this descriptor: (IN endpoint, number 1)
bmAttributes	1	0x01	Field describing the endpoint's attributes: Isochronous transfer type.
wMaxPacketSize	2	0x0000/ 0x0000 (2)	Maximum packet size this endpoint is capable of sending: 0 bytes or 0 bytes (2).
bInterval	1	0x01	Interval for polling endpoint for data transfers (in ms). For isochronous endpoints this field must be set to 1 (ie. every USB frame).

Descriptor Field	Size (bytes)	Value	Description
bLength	1	0x09	Size of this descriptor in bytes.
bDescriptorType	1	0x04	INTERFACE Descriptor Type.
bInterfaceNumber	1	0x01	Number of interface.
bAlternateSetting	1	0x01	Value used to select alternate setting for the interface identified in the prior field.
bNumEndpoints	1	0x01	Number of endpoints used by this interface (excluding Endpoint 0).
bInterfaceClass	1	0xFF	Class code (assigned by USB). A value of 0xFF means that the interface class is vendor specific.
bInterfaceSubClass	1	0x00	Subclass code (assigned by USB).
bInterfaceProtocol	1	0xFF	Protocol code (assigned by USB). A value of 0xFF means that the device uses a vendor specific protocol for this interface.
iInterface	1	0x00 (1)	Index of string descriptor describing this interface.

Descriptor Field	Size (bytes)	Value	Description
bLength	1	0x07	Size of this descriptor in bytes.
bDescriptorType	1	0x05	ENDPOINT Descriptor Type.
bEndpointAddress	1	0x81	The address of the endpoint on the USB device described by this descriptor: (IN endpoint, number 1)
bmAttributes	1	0x01	Field describing the endpoint's attributes: Isochronous transfer type.

Descriptor Field	Size (bytes)	Value	Description
wMaxPacketSize	2	0x01C0/ 0x0140 (2)	Maximum packet size this endpoint is capable of sending: 448 bytes or 320 bytes (2).
bInterval	1	0x01	Interval for polling endpoint for data transfers (in ms). For isochronous endpoints this field must be set to 1 (ie. every USB frame).

Descriptor Field	Size (bytes)	Value	Description
bLength	1	0x09	Size of this descriptor in bytes.
bDescriptorType	1	0x04	INTERFACE Descriptor Type.
bInterfaceNumber	1	0x01	Number of interface.
bAlternateSetting	1	0x02	Value used to select alternate setting for the interface identified in the prior field.
bNumEndpoints	1	0x01	Number of endpoints used by this interface (excluding Endpoint 0).
bInterfaceClass	1	0xFF	Class code (assigned by USB). A value of 0xFF means that the interface class is vendor specific.
bInterfaceSubClass	1	0x00	Subclass code (assigned by USB).
bInterfaceProtocol	1	0xFF	Protocol code (assigned by USB). A value of 0xFF means that the device uses a vendor specific protocol for this interface.
iInterface	1	0x00 (1)	Index of string descriptor describing this interface.

Descriptor Field	Size (bytes)	Value	Description
bLength	1	0x07	Size of this descriptor in bytes.
bDescriptorType	1	0x05	ENDPOINT Descriptor Type.
bEndpointAddress	1	0x81	The address of the endpoint on the USB device described by this descriptor: (IN endpoint, number 1)
bmAttributes	1	0x01	Field describing the endpoint's attributes: Isochronous transfer type.
wMaxPacketSize	2	0x02C0/ 0x0268 (2)	Maximum packet size this endpoint is capable of sending: 704 bytes or 616 bytes (2).
bInterval	1	0x01	Interval for polling endpoint for data transfers (in ms). For isochronous endpoints this field must be set to 1 (ie. every USB frame).

Descriptor Field	Size (bytes)	Value	Description
bLength	1	0x09	Size of this descriptor in bytes.
bDescriptorType	1	0x04	INTERFACE Descriptor Type.
bInterfaceNumber	1	0x01	Number of interface.
bAlternateSetting	1	0x03	Value used to select alternate setting for the interface identified in the prior field.
bNumEndpoints	1	0x01	Number of endpoints used by this interface (excluding Endpoint 0).
bInterfaceClass	1	0xFF	Class code (assigned by USB). A value of 0xFF means that the interface class is vendor specific.
bInterfaceSubClass	1	0x00	Subclass code (assigned by USB).
bInterfaceProtocol	1	0xFF	Protocol code (assigned by USB). A value of 0xFF means that the device uses a vendor specific protocol for this interface.
iInterface	1	0x00 (1)	Index of string descriptor describing this interface.

Descriptor Field	Size (bytes)	Value	Description
bLength	1	0x07	Size of this descriptor in bytes.
bDescriptorType	1	0x05	ENDPOINT Descriptor Type.
bEndpointAddress	1	0x81	The address of the endpoint on the USB device described by this descriptor: (IN endpoint, number 1)
bmAttributes	1	0x01	Field describing the endpoint's attributes: Isochronous transfer type.
wMaxPacketSize	2	0x03C0/ 0x0380 (2)	Maximum packet size this endpoint is capable of sending: 960 bytes or 896 bytes (2).
bInterval	1	0x01	Interval for polling endpoint for data transfers (in ms). For isochronous endpoints this field must be set to 1 (ie. every USB frame).

Notes:

(1) The USB camera does not support String Descriptors. Any request by the host for a STRING descriptor will cause the camera to STALL the default pipe.

(2) The first number shows maximum packet size values returned when $\overline{\text{USB\_BANDWIDTH}}$ is left unconnected (internally pulled up), as is the case in normal operation. If USB_BANDWIDTH is pulled LO externally the second number is returned as the maximum packet size for the isochronous endpoint. For further information see section **2.2.4.4**.

On successful completion of these control transfers the host has gathered all relevant information about the

camera and is able to load the appropriate drivers for the camera. Once this has been done the camera is said to be enumerated.

2.2.4.4 USB Isochronous Transfers

As stated in section **2.2.4.1**, Isochronous IN Endpoint 1 is used to transfer video data from the camera to the host. Interface 1 has four Alternate Settings defined allowing different USB bandwidths to be requested for this transfer of video data. Furthermore, it is possible to use the USB_BANDWIDTH input to select a second set of Alternate Setting isochronous bandwidths. In normal operation USB_BANDWIDTH is left unconnected (internally pulled up). If it is pulled LO externally a second set of Alternate Setting isochronous bandwidths are used. The table below shows the Alternate Setting isochronous bandwidths of Endpoint 1 for both USB_BANDWIDTH HI and LO.

Alternate Setting	Isochronous bandwidth per ms USB_BANDWIDTH=1 (default)	Isochronous bandwidth per ms USB_BANDWIDTH=0
0	0 bytes	0 bytes
1	448 bytes	320 bytes
2	704 bytes	616 bytes
3	960 bytes	896 bytes

Note that the second set of Alternate Setting bandwidths available by pulling USB_BANDWIDTH LO allow a slightly lower set of bandwidth settings to be used.

The isochronous bandwidth is reported as the number of bytes per USB frame. The USB is framed into 1ms time slots.

Alternate Setting 0 is always 0 bytes/ms. This ensures that we can select an Alternate Setting for the camera where no bandwidth is reserved. Alternate Setting 0 is the default Alternate Setting selected when the CPiA USB driver is loaded, such that when the camera is enumerated but not in operation (no application is using the camera hardware to transfer video data) then no USB bandwidth is requested. This is essential for a USB-friendly peripheral such that other devices may use the full potential of the bus at this time.

When an application wishes to use the camera the CPiA USB driver will request isochronous bandwidth for the transfer of video data. It will start by requesting the largest bandwidth supported by CPiA, that is Alternate Setting 3, to ensure the best possible framerate by default. If the USB bus has many isochronous devices attached and Alternate Setting 3 cannot be supported the CPiA USB driver will then request Alternate Setting 2 and if necessary Alternate Setting 1 to guarantee bandwidth for the transfer of video data. If even the bandwidth requirement of Alternate Setting 1 cannot be supported by the bus at that time the user will have to stop using another isochronous device to free-up isochronous bus time to allow the USB camera to operate.

Additionally, the camera user can select the USB bandwidth they wish to use by means of the Multimedia Control Panel Settings Dialogue for the camera. Here they can select High (Alternate Setting 3), Medium (Alternate Setting 2) or Low (Alternate Setting 1) USB bandwidth. Thus if the CPiA camera is currently using High bandwidth and they wish to connect and simultaneously use another isochronous USB device they can manually force the CPiA camera to use less bandwidth by selecting Medium or Low bandwidth settings. Obviously this may result in lower framerate video (depending on the current compression setting).

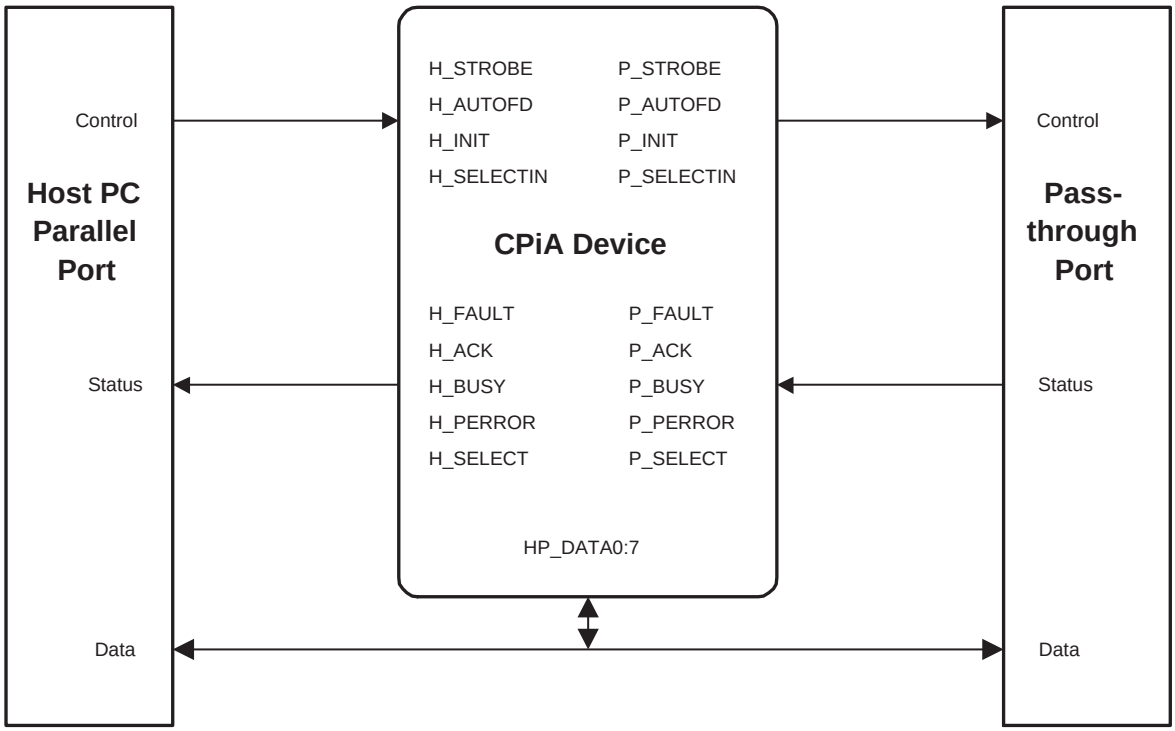
Once an Alternate Setting has been selected and is supported by the bus then the CPiA camera has guaranteed bandwidth and latency for the transfer of video data. This means that in every 1ms USB time slot an IN token will be issued to the CPiA Endpoint 1 for the transfer of up to *MaxPacketSize* bytes of video data, where *MaxPacketSize* is the reserved bandwidth per ms. The CPiA camera will transfer as much data as it currently has available up to *MaxPacketSize*. When no compression is selected then *MaxPacketSize* bytes of video data will be transfered in every USB frame. If compression is selected then less than *MaxPacketSize* bytes of video data may be transfered depending on the level of compression requested. When the device is

sending a compressed image stream to the host the amount of data to be sent for each image frame can vary, depending on the compression ratio achieved. To enable the host to detect the end of valid image data the device inserts four *0xFF* values into the image stream to mark the end of image frame. As a sequence of four 0xFF values will never occur within a valid frame of image data the host can use this to simply detect the end of the image data, without having to decompress the image. The host driver checks the intergrity of the image data stream before it is decoded. If any corruption is detected the frame is discarded and another uploaded.

2.2.5 Parallel Port Interface

The parallel port interface is designed to attach directly to the parallel printer port of a PC. As such it contains 8 bi-directional data lines, 4 control inputs and 5 status outputs for connection to the host parallel port. In addition, a further 4 outputs, 5 inputs are provided to support pass-through parallel port designs. These two ports are referred to as the *Host Port* and *Pass-through Port.*

Figure 2.5 : Parallel Port Interface Signals



This following sections give a guide to the low level protocols used to transfer command and image data over the parallel port interface. They should be read in conjunction with the document "IEEE Std 1284 - Standard Signaling method for a Bidirectional Parallel Peripheral Interface for Personal Computers". It is intended to give users enough information to design and debug boards utilising the CPiA device.

2.2.5.6 Parallel Port Protocol

There are three classes of transfer that take place over the parallel port.

- IEEE Std 1284 Device ID Transfers
- Command / Status Transfers
- Image Stream Transfers

1284 Device ID transfers are used by the Microsoft Windows 95/98 operating systems to detect the presence of the CPiA-based camera at boot time and load the approriate drivers.

Command and Status transfers are used to pass control information to the CPiA-based camera from the host as well as returning state information back to the host.

Image Stream Transfers are used to transfer image data from CPiA to the host. This data can be in a compressed or uncompressed format depending on how the CPiA has been set up via the associated host PC software driver.

2.2.5.7 1284 Device ID Transfers

The CPiA device can return a Device ID string, as defined in IEEE Std 1284, using Nibble Mode Reverse Channel Transfer. This Device ID is requested by the Microsoft Windows 95/98 operating systems during their PnP boot processes. It is used to load the approriate device drivers for the hardware attached to the port.

The device can return two possible Device ID strings dependent on the state of the $\overline{\text{ID\_SELECT}}$ pin. This pin has an internal pull-up resistor which, when not connected, provides a Device ID for a parallel port only CPiA-based camera module. When pulled low externally, the Device ID will be that which is used for a combined parallel port and USB module.

The complete Device ID's returned in these two situations are shown in the table below.

$\overline{\text{ID\_SELECT}}$	Returned Device ID
1(default)	MFG:VLSI Vision Ltd;MDL:PPC2 Camera;CMD:CPIA_1-20;CLS:MEDIA;DES:Parallel Port Camera;
0	MFG:VLSI Vision Ltd;MDL:DUAL-B Camera;CMD:CPIA_1-20;CLS:MEDIA;DES:USB / Parallel Port Camera;

2.2.5.8 Command / Status Transfers

All command and status transfers are preceeded by the negotiation and setup phases as defined in the Std 1284 for the ECP and NIbble modes.

Forward data transfers always use the ECP transfer protocol.

Reverse data transfers use either ECP, or Nibble mode, transfer protocols, depending or whether the host has a bi-directional port or not.

Channel addressing and RLE as defined in the ECP spec are not used by CPiA.

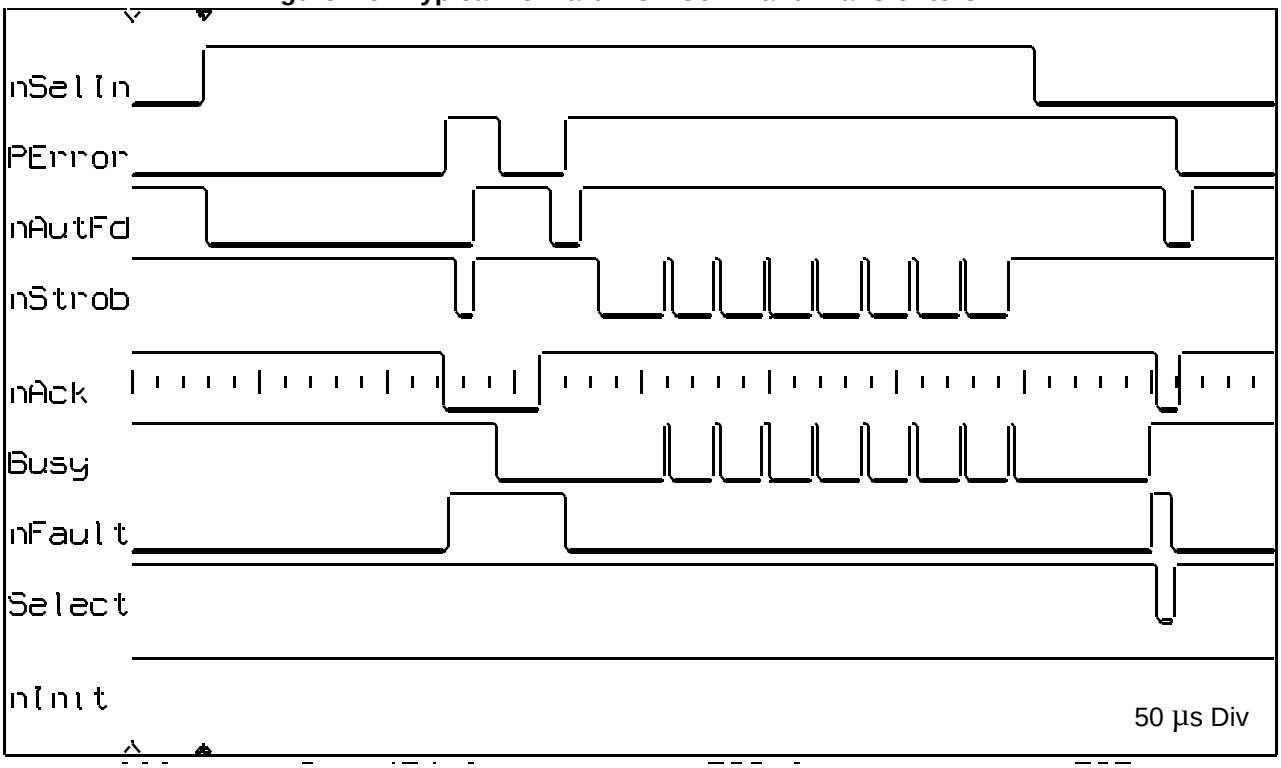
Data is transfered in 8 byte packets, each packet being preceeded by the 1284 negotiation and setup sequences, and terminated with a 1284 termination sequence.

Commands are passed to CPiA in a single 8 byte packet, termed a *command transfer.* This can optionally be followed by the transfer of a d*ata transfer.* Bytes within the command packet are used to specify whether a data packet is to follow, and its direction.

Most commands are not accompanied by data packets.

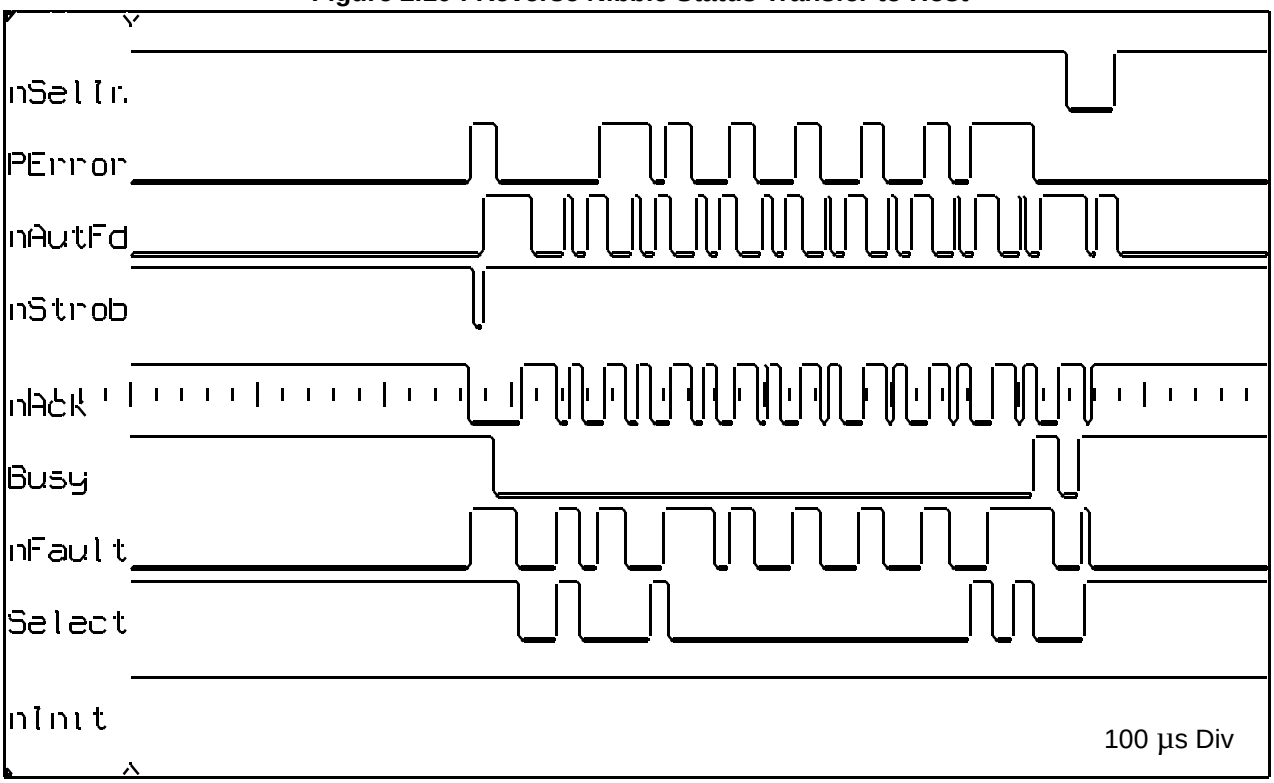
The following diagram, Figure X.Y, shows the activity that occurs on the parallel port control and status lines during a command transfer to CPiA, via a Forward ECP Mode transfer. NOTE - The actual data is carried on the 8 data lines, which are not shown on the diagram.

Figure 2.9 : Typical Forward ECP Command Transfer to CPiA



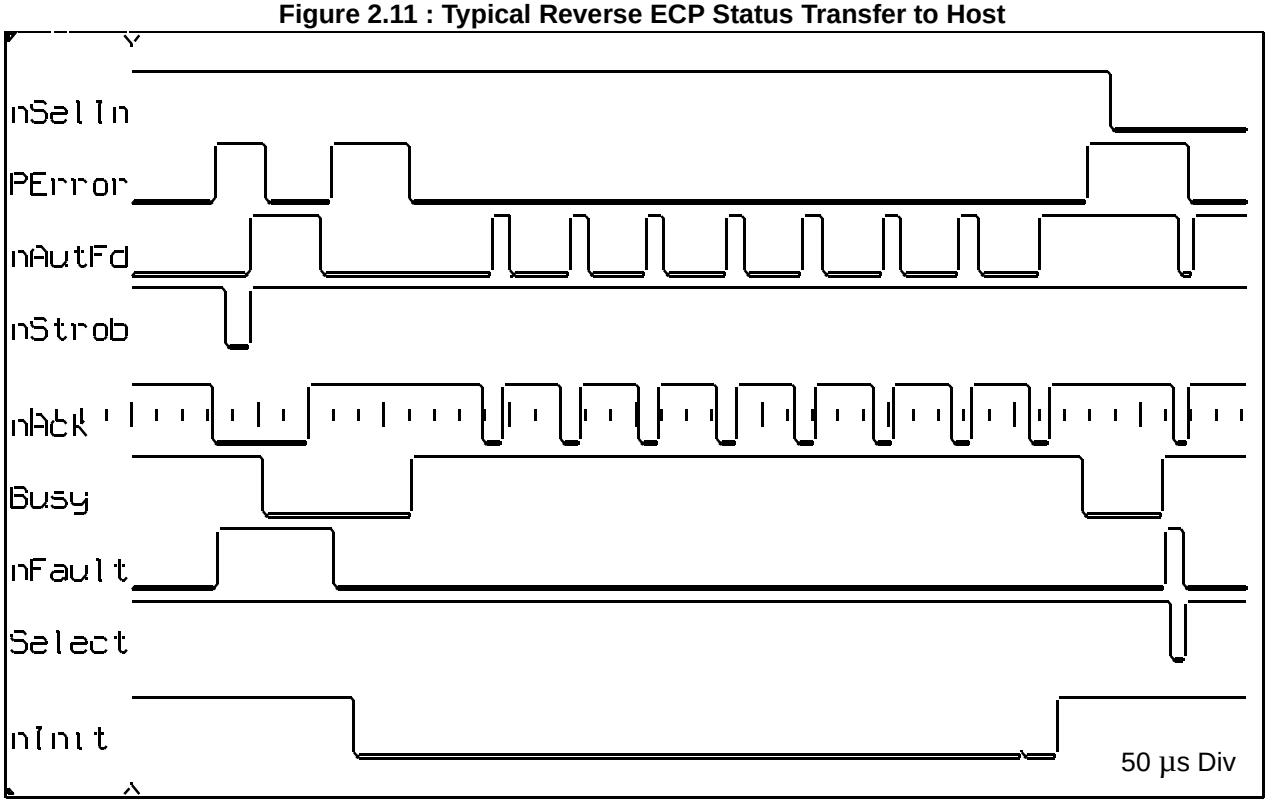
The diagram below, Figure X.Y, shows the activity that occurs on parallel port control and status lines during the return of status information to the host via a Nibble Mode transfer. In this transfer mode the actual data is carried on the PError, nAutoFd, nAck, Select lines.

Figure 2.10 : Reverse Nibble Status Transfer to Host



The diagram below, Figure X.Y, shows the activity that occurs on parallel port control and status lines during the return of status information to the host via a Reverse ECP Mode transfer. NOTE - The actual data is carried on the 8 data lines, which are not shown on the diagram.

Figure 2.11 : Typical Reverse ECP Status Transfer to Host



2.2.5.12 Image Stream Transfers

Transfers of image stream data are distinguished from those of the command and status data by the use of a slightly different negotiation phase preceeding the transfer. Image stream transfers operates only in the reverse direction i.e. CPiA to host.

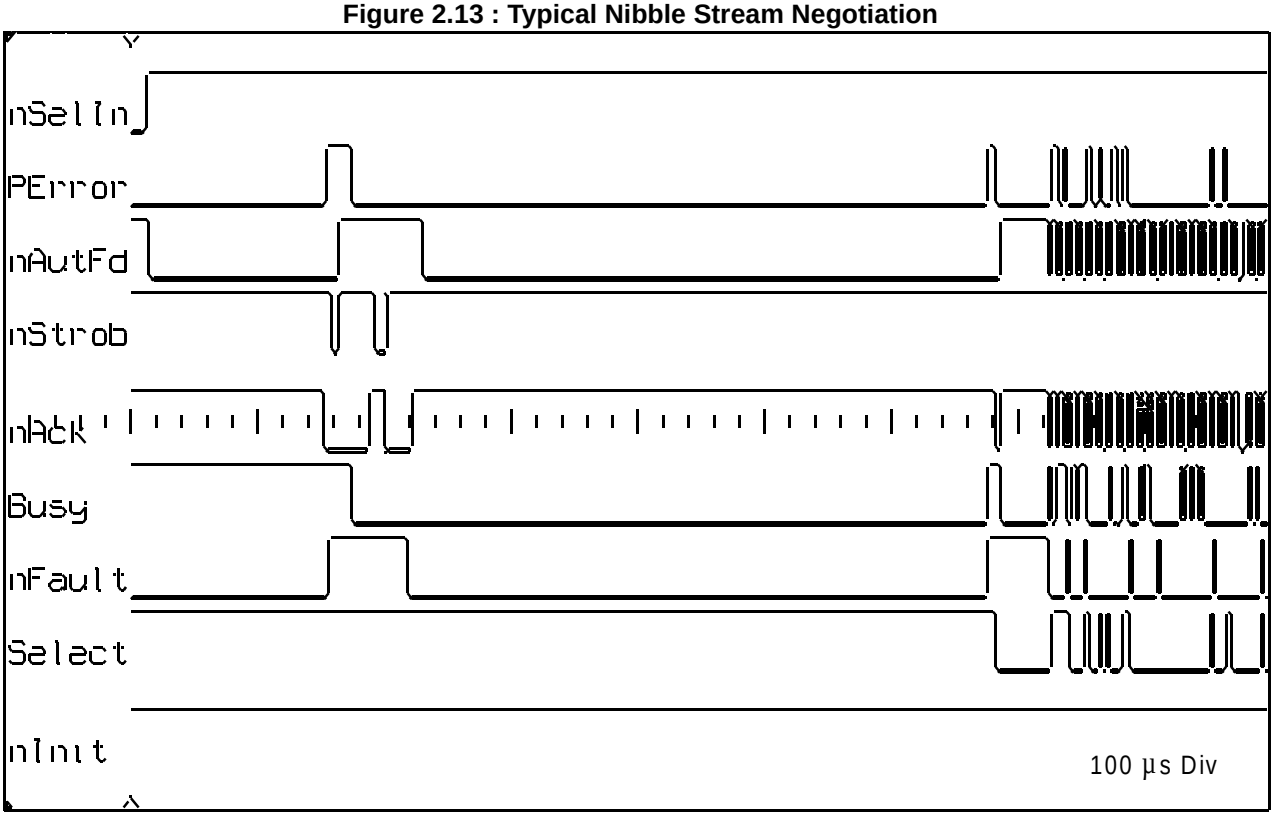
All image stream transfers are preceeded by a 1284 negotiation sequence which sets the "extensiblity link request" bit within the first Extensibility Request Value. The subsequent Extensiblity Request Value then determines which of two possible transfer modes will be used to transfered the data. Image Stream transfers are terminated with the standard 1284 termination sequence.

The protocol used to transfer the data depends on the capabilites of the host. If it has an ECP parallel port then the standard ECP byte level protocol is used. However, if it has a bi-directional, or 4 bit port, then a modified form of the 1284 Nibble mode transfer is used. This modified form, refered to hereafter as a *Nibble Stream mode* , is used in preference to the standard nibble mode to improve the maximum transfer rate. Nibble Stream mode differs from standard Nibble mode by the fact that both the rising and falling edges of the nAutoFd signal are used to clock data.

The diagram below shows the negotiation sequence that proceeds a Nibble Stream Mode transfer, and the start of of the actual image data transfer.

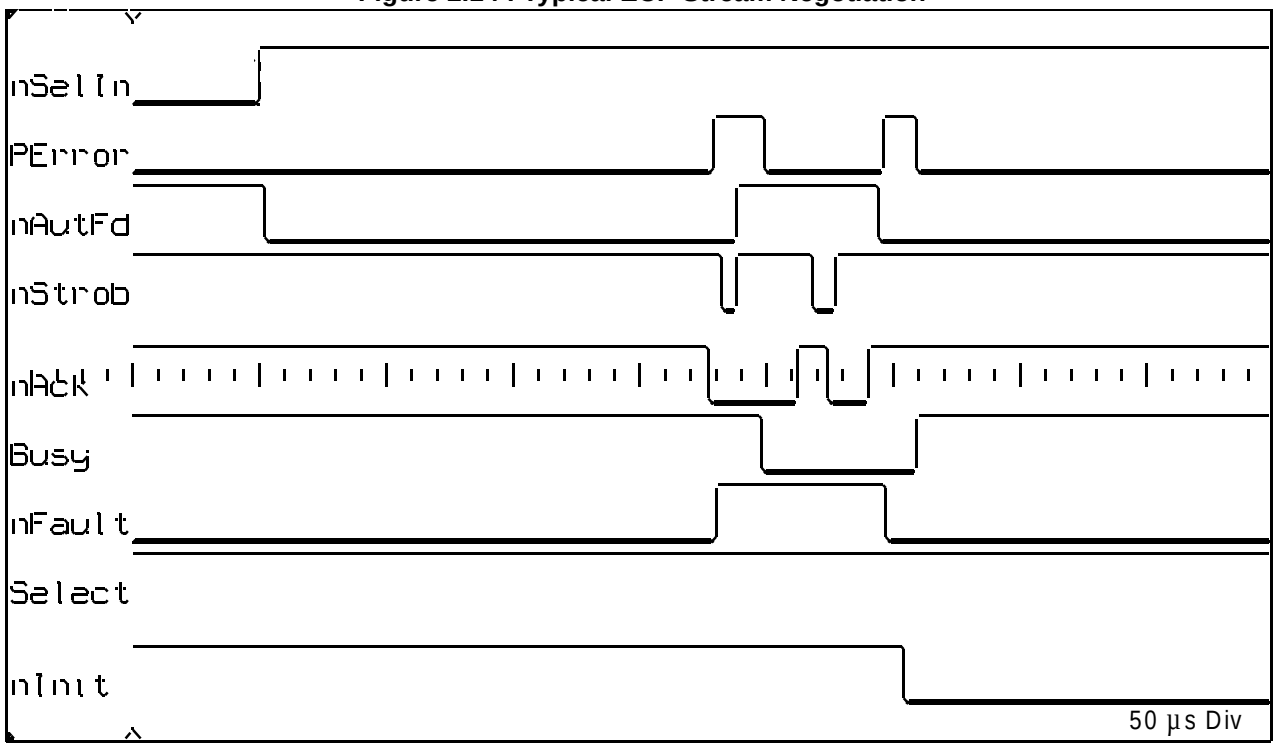
NOTE - Requests by the host for Nibble Stream or ECP Stream mode transfers can be easily disguished from standard Nibble Mode and ECP negotiations by the double nStrobe low pulses that occur during the negotiation phase.

Figure 2.13 : Typical Nibble Stream Negotiation



The diagram below shows the negotiation sequence that proceeds a ECP Stream Mode transfer, no actual bytes are shown being transfered.

Figure 2.14 : Typical ECP Stream Negotiation



Notes on Image Stream Transfers

CPiA produces a low going pulse, of approximately 200 µs, on the nAck line when image stream data is available to upload. If the host driver can use this pulse to trigger a parallel port interrupt service routine to start the the image transfer process.

When the device is sending compressed image stream to the host the amount of data to be sent for each frame can vary, depending on the compression ratio achieved. So that the host can detect the end of valid data the device inserts a stream of *0xFF* values into the image stream after the host reads past the end the valid image data. As a sequence of four 0xFF values will never occur within a valid frame of image data the host can use this to simply detect the end of the image data, without having to decompress the image.

The host driver checks the intergrity of the image data stream before it is decoded, if any corruption is detected the frame is discarded and another uploaded.

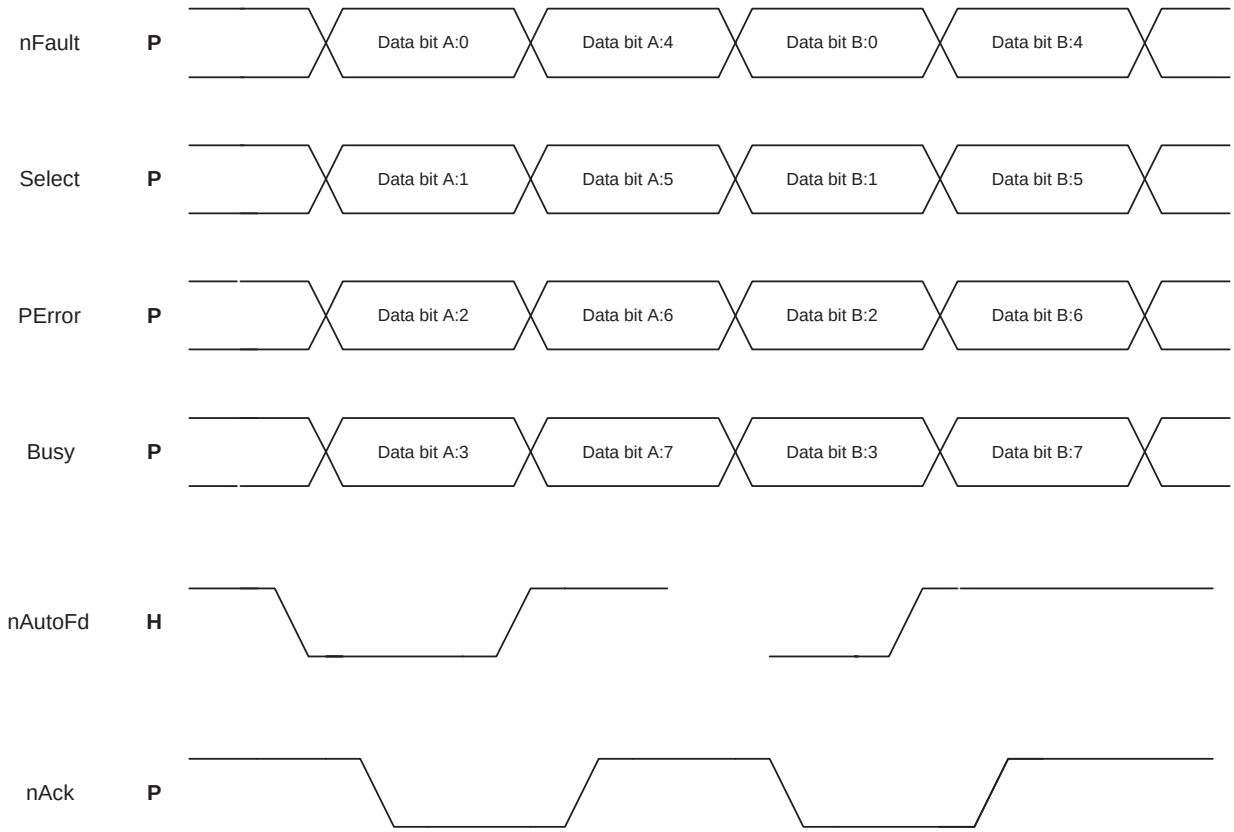
If, while uploading uncompressed images, the transfer of data is monitored via an osciliscope, or logic analyser, it will be seen there are regular short pauses in the transfer, in the order of 200µs to 2ms each. These are caused by the host driver suspending the transfer while it copies data out of it's temporary buffer into the target frame buffer.

Details of Nibble Stream Mode Protocol

The figure below shows the details of the Nibble Stream Protocol. It shows two succesive bytes, A and B, being transfered.

The host requests a byte bysetting nAutoFd to LO. CPiA will then present the lower nibble of the byte on the 4 status lines. It then sets nAck LO to signify that the data is ready. nAutoFd going HI causes the upper nibble to be placed on the status lines and nAck is then taken HI to signify the data is ready for the host to read.

Figure 2.15 : Nibble Stream Mode Protocol



2.2.5.16 Pass-Through Mode

- The CPiA device includes support for a parallel port pass-through mode of operation to allow another parallel port device, e.g. a printer, to be daisy chained from a CPiA-based camera.
- The Windows 95/98 device driver supplied by VLSI Vision Ltd does not yet support his feature and so only a brief description of the operation of this mode will be given here.
- Please contact Vision for more information.

Description of Pass-Through Mode Operation

The host can instruct the CPiA to enter pass-through mode by sending it a *GotoPassThrough* command. All host control signals will then be echoed to the pass-through port and status signals from the pass-through port will be echoed back to the host. In this mode the host has unrestricted access the pass-through device and may use it as if the CPiA-based camera was not attached. While in this mode the CPiA will not attempt to initiate communications with the host, or in anyway interfere with the host to pass-through device communication that may be taking place.

When the host requires to communicate with CPiA, it must switch CPiA out of pass-through mode by sending it a VPP(Vision Passthrough Packet) , which has a similar structure and signaling method as that defined in the IEEE 1284.3 Draft Specification. This VPP packet is sent using only the 8 data lines without any strobing or acknowledging on the control or status lines, this ensures that the pass-through device will not respond to it.

When CPiA switches from pass-through into manual mode it locks the state of the pass-through port control signals and prevents the pass-through status signals from being echoed back to the host. The host can now communicate as it wishes with CPiA without the pass-through device being aware of any activity on the parallel port.

2.2.5.17 IEEE 1284 Mode Handshake Values

The following table shows the timing values of the parallel port interface in terms of the signal transitions defined in the IEEE Std 1284 figures 3 through 23.

Figure 2.18 : IEEE 1284 Mode Handshake Values

Time	Minimum	Maximum	Description
T_H	0	0.5 s(note 1)	Host response time
T	0		Infinite response time
T_L	70 ns	10 ms (note 2)	Peripheral response time
T_R	70 ns		Peripheral response time (ECP Mode only)
T_S	(note 3)		Host recovery time (ECP Mode only)
T_P	0.5 µs		Minimum setup or pulse width
T_D	0		Minimum data setup time (ECP Mode only)

Notes:

1 - A delay of more than this period during a Command/Status transfer can cause the watchdog reset circuitry on CPiA to time out and generate a system reset.

2 - Normally 10ms, however, certain commands being issued to CPiA can caused this to increase to 40ms

3 - As an ECP Forward Channel stall condition will never be generated by CPiA, the Host Transfer Recovery handshake is not supported.

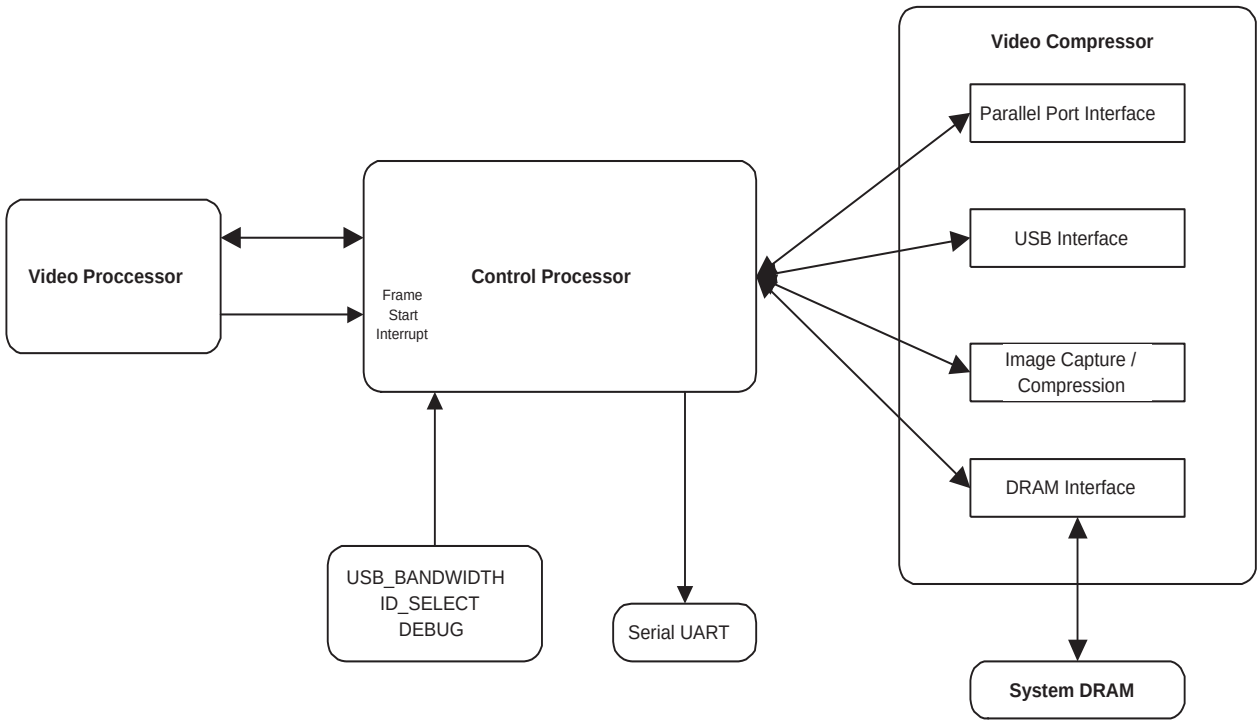
2.2.6 Control Processor

2.2.6.19 Control Processor Interfaces

As shown in figure 2.16 below, the Control Processor uses five main interfaces to control the operation of the CPiA device:

- VP Control Registers
- VP Frame Start Interrupt
- VC Control Registers i.e. PP, USB interfaces, Compression Control and DRAM Interface
- USB_BANDWIDTH, ID_SELECT and DEBUG input pins
- UART serial comms (debug output only)

Figure 2.20 : Control Processor Interfaces



The VP Control Registers are used to control the colour processing of the raw sensor data. Colour channel gains, colour saturation and contrast values are all applied by the VP module. Also, communications to the sensor, for control of exposure time and gain, are done via the VP module.

The VP Frame Start interrupt is used to trigger Exposure Control and Auto White Balance algorithms that run on the Control Processor.

The VC Registers are used to control the host interfaces (both Parallel Port and USB) and the image capture / compression / upload processes. In addition access to the DRAM, used to store the frameheader, and during self-test mode, is via the VC regisiters. Control of transitions between the system power states of the camera is preformed via VC registers that access the Power Managment module.

The USB_BANDWIDTH, ID_SELECT and DEBUG inputs go directly to port pins on the embedded Control Processor core and modify the several aspects of the camera behaviour. In normal operation these three inputs should be left unconnected (internally pulled up).

The USB_BANDWIDTH input can be used to select an alternate set of isochronous bandwidths for the USB interface. See the USB Interface section of this document for more details.

The ID_SELECT input is used to select a Device ID (Parallel Port interface) or Product ID (USB interface) that is used for combined USB and Parallel Port cameras. If held LO it causes a different 1284 Device ID to be returned on the parallel port interface, and a different USB Product ID on the USB Interface.

The DEBUG input if pulled LO will cause the Control Processor to send diagnostic output to the serial UART.

This output consists mainly of a trace of all command and status data that has been received/transmitted over the host parallel port or USB interfaces.

The serial UART Tx signal is used to output diagnostic information. The Rx input is not used to receive serial data, but the Control Processor does check it's state after power up, or reset. If it is LO at this time the Control Processor enters a continuous Self-test Mode which it will remain in until the next system reset.

2.2.6.21 Control Processor Tasks

The Control Processor is responsible for a variety of functions necessary for the operation of the CPIA device. In brief these functions are:

- Initialisation of VP and VC modules
- Host communications via Parallel Port IEEE 1284 protocol
- Supervision of host USB interface
- Host command execution
- Image Capture / Compression Control
- Camera Auto Exposure
- Camera Auto White Balance
- Production system selftest diagnostics
- Diagnostic debug output

2.2.6.22 Selftest Diagnostics

If the Serial UART input Rx is pulled LO when the system is released from reset the Control Processor will enter Self-test Diagnostics Mode. It will continuously run a set of system level tests and output the result on the Serial UART Tx output. The serial format used is 19200 baud, 8-bit data, no parity, 1 stop bit, no flow control.

Below is typical output from the Selftest Mode.

```
====== CPIA Ver 1-20 ======

Continuous Selftest ...

1 - VP_EN           --------
2 - VC_INT          --------
3 - Host port       --------
4 - Pass-thru port  Fail
5 - USB core        --------

6 - VC register     --------
7 - DRAM access     --------
8 - VP register     --------
9 - Frame grab      --------
10 - VP data        --------
```

It should be noted that several of these self-tests where designed to test system level connectivity between the VP, VC and CP modules during the device development phase, and are now effectively obseleted with the full intergrated CPiA device.

These self-tests do not attempt to test in anyway the connectivity, or correct operation, of the interface to the sensor.

VP_EN Test

Tests an internal CPiA control signal - should never fail.

VC_INT Test

Tests an internal CPiA control signal - should never fail.

Host Port

Tests for short circuits between signals on the Host Parallel port. Will fail if the camera is plugged in a host.

Pass-thru Port

Tests for short circuits and open circuits on the Pass-Through Parallel port. Requires a special loop back connector to do the test, and so will fail if this is not present.

USB Core

Tests that the USB core has been succesfully configured. Does not check actual communications with host. If the 48MHz Xtal is faulty, or starts up too slowly, this is test will fail. Because of this parallel port only designs will always show this test as failing.

VC Register Test

Tests internal CPiA control signals - should never fail.

DRAM Access Test

Checks for correct CPiA access to DRAM. It does this by writing a test pattern to DRAM, and then verifying it.

VP Register Test

Tests internal CPiA control signals - should never fail.

Frame Grab Test

Tests internal CPiA control signals - should never fail.

VP Data Test

Puts VP module into Colour Bar Mode, and captures a test image to DRAM. Then verifys the correct data has been captured. If DRAM Access test fails this test will also fail.

2.2.6.23 Debug output

If the DEBUG input is pulled LO, the Control Processor will output diagnostic information on the Serial UART TX line. This output consists primarily of a trace of all the command and status data transactions that occur over the host parallel port or USB interfaces. It was primarily designed to be of use for developers working on the host driver software.

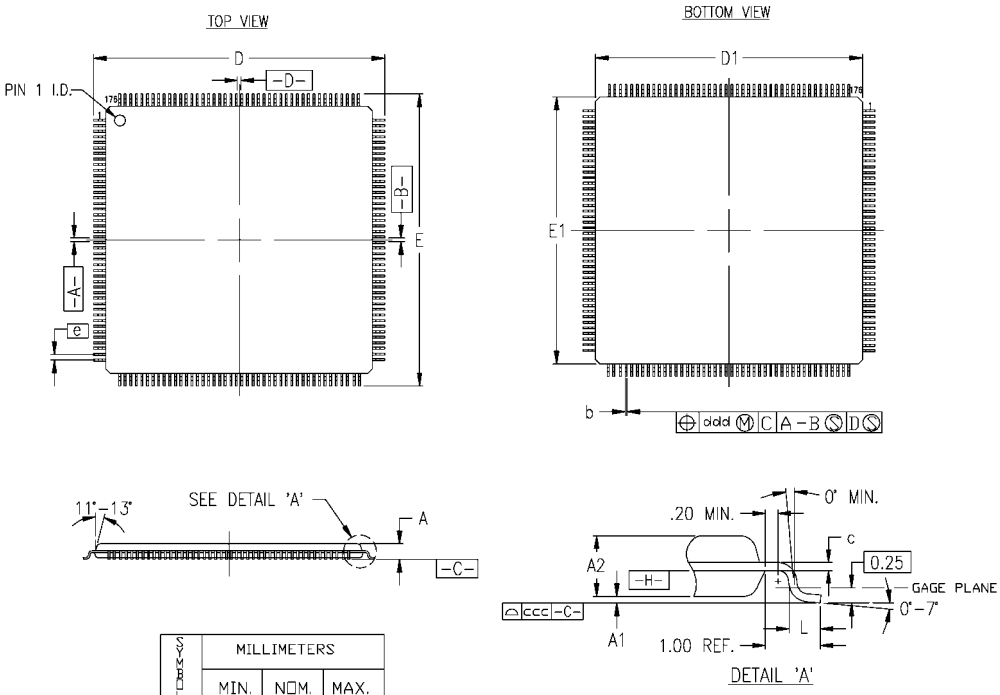
The serial format used is 19200 baud, 8-bit data, no parity, 1 stop bit, no flow control.

The following shows typical output following a system reset and then subsequent commands being received from the host by the camera.

```
====== CPIA Ver 1-20 ======

Process loop ...
Detected PP i/f

Command = 01    CmdData = 00 00 00 00    DataIn = 01 14 01 00 A2 88 88 85

Command = 03    CmdData = 00 00 00 00    DataIn = 02 00 00 00 00 00 10 00

Command = 04    CmdData = 00 00 00 00

Command = A6    CmdData = 02 00 00 00
```

NOTE - When the debug output is enabled it significantly slows the handling of commands by the camera, and thus the framerate acheived on the host will be much reduced.

3 Mechanical Information



SYMBOL	MILLIMETERS		
	MIN.	NOM.	MAX.
A			1.60
A_1	0.05	0.10	0.15
A_2	1.35	1.40	1.45
D/E	26.00 BSC		
D_1/E_1	24.00 BSC		
L	0.45	0.60	0.75
e	0.50 BSC		
b	0.17	0.22	0.27
c	0.09		0.20
ccc			0.08
ddd			0.08

NOTES:

1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ANSI Y14.5M-1982.
2. DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE MOLD PROTRUSION SHALL NOT EXCEED 0.25mm PER SIDE.
3. THE TOP OF PACKAGE MAY BE SMALLER THAN THE BOTTOM OF PACKAGE BY 0.15mm.
4. THIS OUTLINE CONFORMS TO JEDEC PUBLICATION 95 REGISTRATION MS-026-BGA

176-PIN TQFP (TQ176)