TW2819

Multichannel H.264 Audio/Video Codec

FN8404
Rev.1.00
July 31, 2015

The TW2819 is a high performance and cost effective multi-channel H.264 CODEC solution designed for the security surveillance market. It is capable of receiving and encoding maximum 16 channels standard definition (SD) or 960H (WD1) video. It can also support up to 8-channel high definition (HD) 720p or 4-channel 1080p encode. It supports D1/960H/720p/1080p decode also up to 4 channels. Each video channel can be independently controlled in terms of encode/decode mode, frame rate, bit rate.

Applications

- HD-SDI Digital Video Recorder
- H.264 Network Video Encoder
- HD compression PC card

Features

Video Features

- Real time full-duplex video CODEC compliant to H.264 main profile standard
- Four BT.656 byte-interleaved video input port up to 144MHz
- Four BT.656/1120 video output port up to 74.25MHz
- Four dedicated BT.1120 video input ports at 148.5MHz
- Four BT.1120 bidirectional video input/output port at 148.5MHz
- Supports up to 16-channel real time SD encode
- Supports up to 16-channel 960H encode
- Supports up to 8-channel HD 720p encode
- Supports up to 4-channel HD 1080p encode
- Supports CIF JPEG or H.264 network stream encode
- Supports simple OSD function on the H-264 encoding path

Audio Features

- Real time 16-channel encode and 1-channel decode compliant to ADPCM standard.
- Four digital audio input I/F compliant to I2S standard
- One digital audio output I/F compliant to I2S standard
- Audio sample rate from 8kHz to 48kHz Peripheral I/F

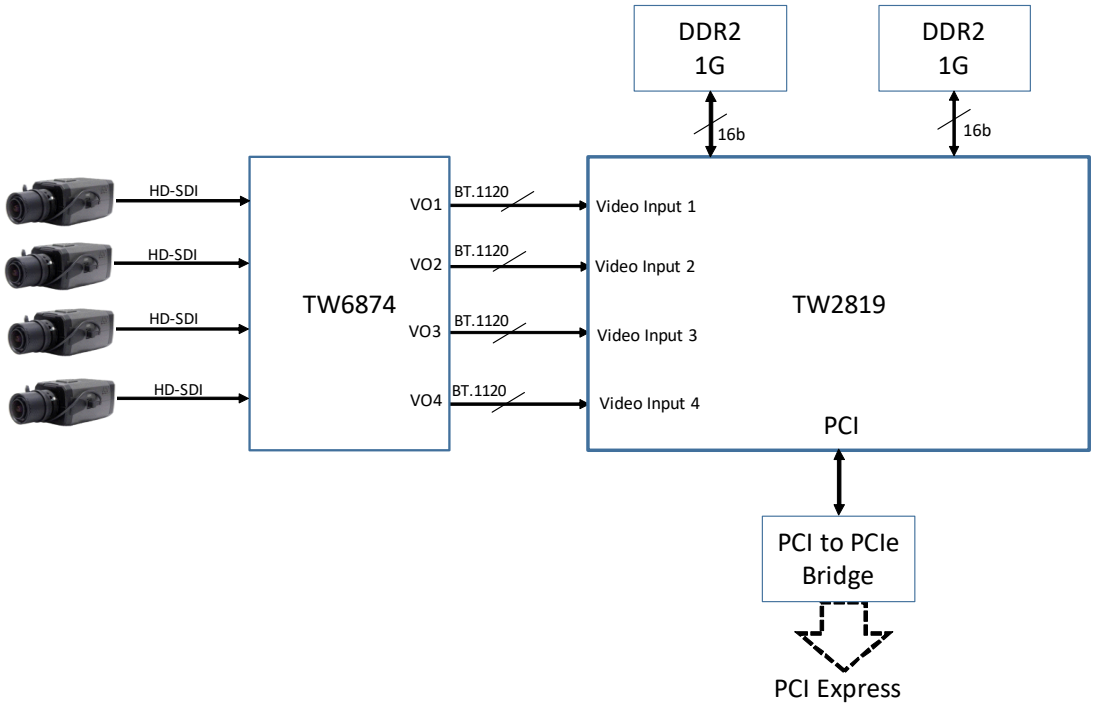


FIGURE 1. TYPICAL APPLICATION DIAGRAM

Peripheral

- Two external 16-bit DDR2 SDRAM at 432MHz
- 32-bit PCI target for host communications at 33/66MHz

Voltage and Power Consumption

- 1.2V for core
- 3.3V for I/O pad
- 1.8V for DDR2 DRAM I/O
- 4.7W typical case power consumption

Package

- TEPBGA-676, 35mmx35mm, 1.0mm lead pitch

Table of Contents

Ordering Information

PART NUMBER (Notes 1, 2)	PART MARKING	TEMP RANGE (°C)	PACKAGE (RoHS Compliant)	PKG. DWG. #
TW2819-BA2-CR	TW2819 PKBA2-CR	-40 to +85	676 Ld 35mmx35mm PBGA	N/A

NOTES:

1. Intersil Pb-free BGA packaged products employ special Pb-free material sets; molding compounds/die attach materials and SnAgCu - e1 solder ball terminals, which are RoHS compliant and compatible with both SnPb and Pb-free soldering operations. Intersil Pb-free BGA packaged products are MSL classified at Pb-free peak reflow temperatures that meet or exceed the Pb-free requirements of IPC/JEDEC J STD-020.
2. For more information on MSL, please see tech brief TB363.

Pin Configuration

TW2819
(676 LD PBGA)
TOP VIEW

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
A	N/C	VSS	VSS	csdi_data [6]	csdi_data [9]	VDE33	clk_csdi	VSS	ASYNR_3	ASYNR_0	VSS	mem_dqs [1]	SSTL_VDE1	mem_dq[5]	mem_dqs [0]	VSS	mem_clk
B	VSS	VSS	ACLKR_0	csdi_data [2]	csdi_data [7]	csdi_data [12]	csdi_data [13]	ASYNR_4	ASYNR_2	ASYNR_1	SSTL_VDE1	mem_dqs_n[1]	mem_dq[8]	mem_dq[6]	mem_dqs_n[0]	SSTL_VDE1	mem_clk_n
C	ADATR_2	ACLKR_6	ACLKR_3	csdi_data [0]	csdi_data [3]	csdi_data [10]	csdi_data [11]	ASYNR_6	ASYNR_5	VSS	mem_dq[13]	mem_dq[11]	mem_dq[9]	SSTL_VDE1	mem_dq[3]	mem_dq[1]	mem_cke
D	VDE33	ADATR_1	ACLKR_5	ACLKR_2	csdi_data [1]	csdi_data [8]	csdi_data [5]	csdi_data [15]	ASYNR_7	VSS	mem_dq[14]	mem_dq[12]	mem_dq[10]	mem_dq[7]	mem_dq[4]	mem_dq[2]	mem_dq[0]
E	MEM_PLL_AVD	ADATR_3	ADATR_0	ACLKR_4	ACLKR_1	csdi_data [4]	VDE33	csdi_data [14]	VDE33	VDD	mem_dq[15]	SSTL_VDE1	mem_dm [1]	mem_ref	SSTL_VDE1	mem_dm [0]	mem_doc dres
F	MEM_PLL_AVS	ADATR_5	ADATR_4	ACLKR_7	VSS												
G	VSS	ACLKP	ADATR_7	ADATR_6	VSS												
H	VIDEO_PLL_AVD	pci_ad[1]	ADATP	ASYNP	VDE33												
J	VIDEO_PLL_AVS	pci_ad[4]	pci_ad[2]	pci_ad[0]	VDE33												
K	VDE33	pci_ad[7]	pci_ad[5]	pci_ad[3]	VDD												
L	CORE_PLL_AVD	pci_ad[10]	pci_ad[8]	pci_ad[6]	VDD												
M	CORE_PLL_AVS	pci_ad[12]	pci_ad[11]	pci_ad[9]	VSS												
N	pci_ad[15]	VSS	pci_ad[14]	pci_ad[13]	VSS								SSTL_VDE1	VDD	VSS	VDD	SSTL_VDE1
P	VDE33	pci_ad[18]	pci_ad[17]	pci_ad[16]	VDE33								SSTL_VDE1	VDD	VSS	VDD	SSTL_VDE1
R	pci_ad[21]	VDE33	pci_ad[20]	pci_ad[19]	VDE33								VSS	VSS	VSS	VSS	VSS
T	VSS	VSS	pci_ad[23]	pci_ad[22]	VSS								VDE33	VDE33	VSS	VSS	VSS
U	VSS	pci_ad[26]	pci_ad[25]	pci_ad[24]	VDE33								VDD	VDD	VSS	VSS	VSS

FIGURE 2. UPPER LEFT OF BALL DIAGRAM

Pin Configuration (Continued)

TW2819
(676 LD PBGA)
TOP VIEW

18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	
VSS	mem_addr[12]	mem_ba[0]	mem_addr[7]	mem_cas_n	mem_addr[1]	mem_addr[0]	VSS	vi_data_7[3]	VDE33	clk_vi_7	VDE33	clk_vi_6	VSS	clk_vi_5	VSS	N/C	A
SSTL_VDE1	mem_addr[10]	mem_addr[8]	mem_addr[6]	mem_addr[5]	mem_cs_n	SSTL_VDE1	vi_data_7[0]	vi_data_7[4]	vi_data_7[7]	vi_data_6[1]	vi_data_6[5]	vi_data_6[7]	vi_data_5[2]	vi_data_5[4]	VSS	VSS	B
mem_addr[13]	mem_addr[11]	mem_ba[1]	SSTL_VDE1	mem_addr[3]	mem_odt	VSS	vi_data_7[1]	vi_data_7[5]	vi_data_6[0]	vi_data_6[3]	vi_data_6[6]	vi_data_5[1]	vi_data_5[6]	vi_data_5[7]	vi_data_4[2]	vi_data_4[3]	C
mem_we_n	mem_ba[2]	mem_addr[9]	mem_ras_n	mem_addr[4]	mem_addr[2]	VSS	vi_data_7[2]	vi_data_7[6]	vi_data_6[2]	vi_data_6[4]	vi_data_5[0]	vi_data_5[3]	vi_data_5[5]	vi_data_4[1]	vi_data_4[4]	VDE33	D
SSTL_VDE1	VSS	SSTL_VDE1	SSTL_VDE1	VSS	SSTL_VDE1	VSS	VDD	VSS	VSS	VSS	VSS	vi_data_4[7]	vi_data_4[0]	vi_data_4[5]	vi_data_3[0]	clk_test	E
												VDD	vi_data_4[6]	vi_data_3[3]	vi_data_3[2]	VDE33	F
												VSS	vi_data_3[1]	vi_data_3[4]	vi_data_3[5]	clk_vi_4	G
												vi_data_3[6]	vi_data_3[7]	vi_data_2[0]	vi_data_2[1]	VDE33	H
												VDD	vi_data_2[2]	vi_data_2[4]	vi_data_2[5]	clk_vi_3	J
												VDE33	vi_data_2[3]	vi_data_2[6]	vi_data_1[2]	VSS	K
												VSS	vi_data_2[7]	vi_data_1[0]	vi_data_1[3]	clk_vi_2	L
												VDE33	vi_data_1[1]	vi_data_1[4]	vi_data_1[6]	VDE33	M
VSS	SSTL_VDE1	VDD	VSS	SSTL_VDE1								VDD	vi_data_1[5]	vi_data_1[7]	vi_data_0[0]	clk_vi_1	N
VSS	SSTL_VDE1	VDD	VSS	SSTL_VDE1								VDE33	vi_data_0[1]	vi_data_0[2]	vi_data_0[3]	VSS	P
VSS	VSS	VSS	VSS	VSS								VSS	vi_data_0[4]	vi_data_0[5]	vi_data_0[6]	clk_vi_0	R
VSS	VSS	VSS	VDE33	VDE33								VDE33	vi_data_0[7]	vo_data_7[0]	vo_data_7[1]	VDE33	T
VSS	VSS	VSS	VDD	VDD								VDD	vo_data_7[2]	vo_data_7[3]	vo_data_7[4]	vo_data_7[5]	U

FIGURE 3. UPPER RIGHT OF BALL DIAGRAM

TW2819
(676 LD PBGA)
TOP VIEW

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
V	ARM_PLL_AVD	VDE33	VSS	pci_ad[27]	VDD								VDD	VDD	VSS	VSS	VSS
W	ARM_PLL_AVS	VDE33	pci_ad[28]	pci_ad[29]	VDD								VDE33	VDE33	VSS	VSS	VSS
Y	VSS	VSS	pci_ad[30]	pci_ad[31]	VDE33								VSS	VSS	VSS	VSS	VSS
AA	pci_cbe_n[0]	pci_cbe_n[1]	pci_cbe_n[2]	pci_cbe_n[3]	VDD								SSTL_VDE2	VDD	VSS	VDD	SSTL_VDE2
AB	VDE33	VDE33	pci_par	pci_frame_n	VDE33								SSTL_VDE2	VDD	VSS	VDD	SSTL_VDE2
AC	pci_clk_in	VDE33	pci_trdy_n	pci_irdy_n	VDD												
AD	VSS	pci_stop_n	pci_devsel_n	pci_perr_n	VSS												
AE	DDR0_PLL_AVD	pci_req_n	pci_inta_n	pci_idsel	VDD												
AF	DDR0_PLL	pci_serr_	pci_gnt_n	pci_rst_in	VSS												
AG	VDE33	i2c_scl	chip_test	sdo_chip	VDD												
AH	VSS	i2c_sda	jtag_tck	jtag_trstn	VSS												
AJ	ref_clk	VDE33	jtag_tdi	jtag_rtck	VDE33												
AK	VDE33	jtag_tms	gpio[28]	gpio[25]	VDD	VDE33	VSS	VDD	VSS	VDD	ddr_dq[15]	$SSTL_VDE2	ddr_dm[1]	ddr_ref	SSTL_VDE2	ddr_dm[0]	ddr_docdres
AL	jtag_tdo	gpio[27]	gpio[24]	uart1_tx	gpio[17]	csdo_data[14]	csdo_data[9]	csdo_data[6]	VDE33	VSS	ddr_dq[14]	ddr_dq[12]	ddr_dq[10]	ddr_dq[7]	ddr_dq[4]	ddr_dq[2]	ddr_dq[0]
AM	gpio[26]	gpio[22]	gpio[23]	uart0_rx	gpio[16]	csdo_data[11]	csdo_data[7]	csdo_data[3]	csdo_data[1]	VSS	ddr_dq[13]	ddr_dq[11]	ddr_dq[9]	SSTL_VDE2	ddr_dq[3]	ddr_dq[1]	ddr_cke
AN	VSS	VSS	uart1_rx	csdo_data[15]	csdo_data[12]	csdo_data[8]	csdo_data[4]	csdo_data[2]	csdo_data[0]	VSS	SSTL_VDE2	ddr_dqs_n[1]	ddr_dq[8]	ddr_dq[6]	ddr_dqs_n[0]	SSTL_VDE2	ddr_clk_n
AP	N/C	VSS	uart0_tx	csdo_data[13]	csdo_data[10]	csdo_data[5]	VDE33	VSS	clk_csdo	VDE33	VSS	ddr_dqs[1]	SSTL_VDE2	ddr_dq[5]	ddr_dqs[0]	VSS	ddr_clk

FIGURE 4. BOTTOM LEFT OF BALL DIAGRAM

Pin Configuration (Continued)

TW2819
(676 LD PBGA)
TOP VIEW

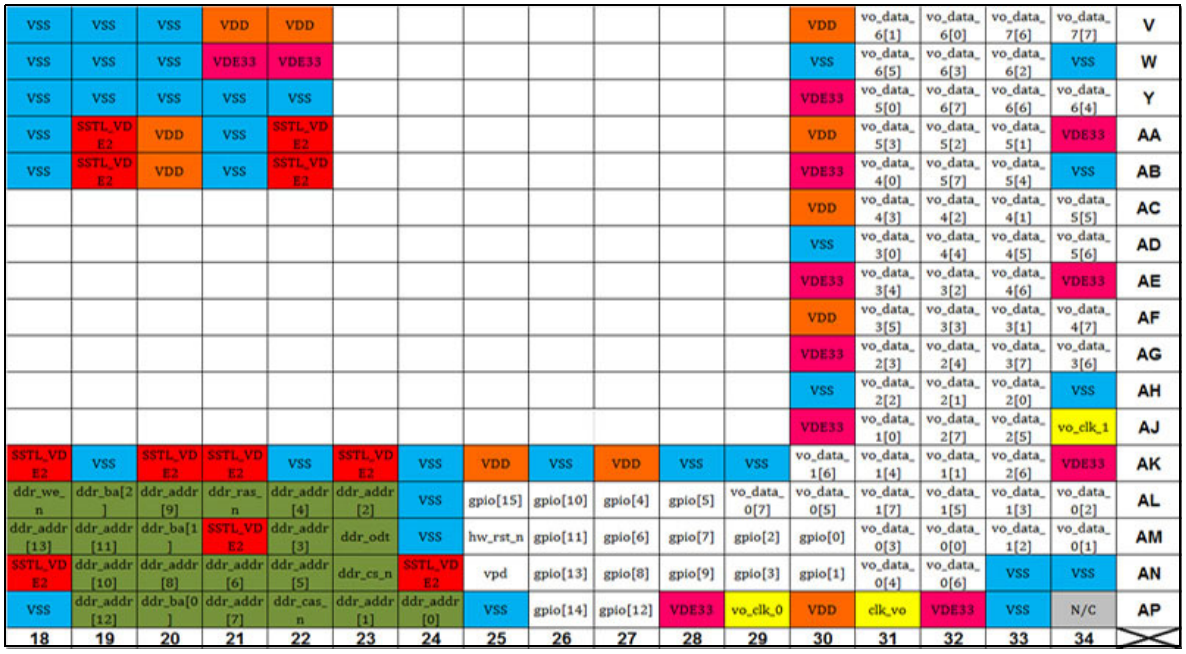


FIGURE 5. BOTTOM RIGHT OF BALL DIAGRAM

Pin Descriptions

BALL NUMBER	SYMBOL	ATTRIBUTE	DESCRIPTIONS	BLOCK
AJ1	ref_clk	Input	Chip reference clock input (54MHz or 27MHz)	
AM25	hw_rst_n	Input	Chip reset input	
AG3	chip_test	Input	Test pin (Tied to "0")	
E34	clk_test	I/O	Test clock	
AN25	vpd	Input	DDR Phy IP test pins (Tied to “0”)	
AP17	ddr_clk	Output	DDR differential clocks	DDR2 #0 Interface
AN17	ddr_clk_n			
AP24	ddr_addr[0]	Output	DDR address bus	
AP23	ddr_addr[1]			
AL23	ddr_addr[2]			
AM22	ddr_addr[3]			
AL22	ddr_addr[4]			
AN22	ddr_addr[5]			
AN21	ddr_addr[6]			
AP21	ddr_addr[7]			
AN20	ddr_addr[8]			
AL20	ddr_addr[9]			

Pin Descriptions (Continued)

BALL NUMBER	SYMBOL	ATTRIBUTE	DESCRIPTIONS	BLOCK
AN19	ddr_addr[10]	Output	DDR address bus	DDR2 #0 Interface
AM19	ddr_addr[11]			
AP19	ddr_addr[12]			
AM18	ddr_addr[13]			
AP20	ddr_ba[0]	Output	DDR control signals	
AM20	ddr_ba[1]			
AL19	ddr_ba[2]			
AM17	ddr_cke			
AN23	ddr_cs_n			
AL21	ddr_ras_n			
AP22	ddr_cas_n			
AL18	ddr_we_n			
AM23	ddr_odt			
AL17	ddr_dq[0]	I/O	DDR data bus	
AM16	ddr_dq[1]			
AL16	ddr_dq[2]			
AM15	ddr_dq[3]			
AL15	ddr_dq[4]			
AP14	ddr_dq[5]			
AN14	ddr_dq[6]			
AL14	ddr_dq[7]			
AN13	ddr_dq[8]			
AM13	ddr_dq[9]	I/O	DDR data bus	DDR2 #0 Interface
AL13	ddr_dq[10]			
AM12	ddr_dq[11]			
AL12	ddr_dq[12]			
AM11	ddr_dq[13]			
AL11	ddr_dq[14]			
AK11	ddr_dq[15]			
AP15	ddr_dqs[0]	I/O	DDR differential data strobe	
AN15	ddr_dqs_n[0]			
AP12	ddr_dqs[1]			
AN12	ddr_dqs_n[1]			
AK16	ddr_dm[0]	Output	DDR data write mask	
AK13	ddr_dm[1]			
AK17	ddr_docdres	Output	DDR SSTL_18 special purpose output signal	
AK14	ddr_ref	Input	DDR SSTL_18 reference voltage	

Pin Descriptions (Continued)

BALL NUMBER	SYMBOL	ATTRIBUTE	DESCRIPTIONS	BLOCK
A17	mem_clk	Output	DDR differential clocks	DDR2 #1 Interface
B17	mem_clk_n			
A24	mem_addr[0]	Output	DDR address bus	
A23	mem_addr[1]			
D23	mem_addr[2]			
C22	mem_addr[3]			
D22	mem_addr[4]			
B22	mem_addr[5]			
B21	mem_addr[6]			
A21	mem_addr[7]			
B20	mem_addr[8]			
D20	mem_addr[9]			
B19	mem_addr[10]			
C19	mem_addr[11]			
A19	mem_addr[12]			
C18	mem_addr[13]			
A20	mem_ba[0]	Output	DDR control signals	
C20	mem_ba[1]			
D19	mem_ba[2]			
C17	mem_cke			
B23	mem_cs_n			
D21	mem_ras_n			
A22	mem_cas_n			
D18	mem_we_n			
C23	mem_odt	Output	DDR control signals	DDR2 #1 Interface
D17	mem_dq[0]	I/O	DDR data bus	
C16	mem_dq[1]			
D16	mem_dq[2]			
C15	mem_dq[3]			
D15	mem_dq[4]			
A14	mem_dq[5]			
B14	mem_dq[6]			
D14	mem_dq[7]			
B13	mem_dq[8]			
C13	mem_dq[9]			
D13	mem_dq[10]			
C12	mem_dq[11]			
D12	mem_dq[12]			
C11	mem_dq[13]			

Pin Descriptions (Continued)

BALL NUMBER	SYMBOL	ATTRIBUTE	DESCRIPTIONS	BLOCK
D11	mem_dq[14]	I/O	DDR data bus	DDR2 #1 Interface
E11	mem_dq[15]			
A15	mem_dqs[0]	I/O	DDR differential data strobe	
B15	mem_dqs_n[0]			
A12	mem_dqs[1]			
B12	mem_dqs_n[1]			
E16	mem_dm[0]	Output	DDR data write mask	
E13	mem_dm[1]			
E17	mem_docdres	Output	DDR SSTL_18 special purpose output signal	
E14	mem_ref	Input	DDR SSTL_18 reference voltage	
AC1	pci_clk_in	Input	PCI input clock	PCI Interface
J4	pci_ad[0]	I/O	PCI address and data bus	
H2	pci_ad[1]			
J3	pci_ad[2]			
K4	pci_ad[3]			
J2	pci_ad[4]			
K3	pci_ad[5]			
L4	pci_ad[6]			
K2	pci_ad[7]			
L3	pci_ad[8]			
M4	pci_ad[9]			
L2	pci_ad[10]			
M3	pci_ad[11]			
M2	pci_ad[12]			
N4	pci_ad[13]	I/O	PCI address and data bus	PCI Interface
N3	pci_ad[14]			
N1	pci_ad[15]			
P4	pci_ad[16]			
P3	pci_ad[17]			
P2	pci_ad[18]			
R4	pci_ad[19]			
R3	pci_ad[20]			
R1	pci_ad[21]			
T4	pci_ad[22]			
T3	pci_ad[23]			
U4	pci_ad[24]			
U3	pci_ad[25]			
U2	pci_ad[26]			
V4	pci_ad[27]			

Pin Descriptions (Continued)

BALL NUMBER	SYMBOL	ATTRIBUTE	DESCRIPTIONS	BLOCK
W3	pci_ad[28]	I/O	PCI address and data bus	PCI Interface
W4	pci_ad[29]			
Y3	pci_ad[30]			
Y4	pci_ad[31]			
AA1	pci_cbe_n[0]	I/O	PCI bus command and byte enable	
AA2	pci_cbe_n[1]			
AA3	pci_cbe_n[2]			
AA4	pci_cbe_n[3]			
AF4	pci_rst_in_n	Input	PCI control signals	
AB3	pci_par	I/O		
AB4	pci_frame_n	I/O		
AC4	pci_irdy_n	I/O		
AC3	pci_trdy_n	I/O		
AD2	pci_stop_n	I/O		
AE4	pci_idsel	Input		
AD3	pci_devsel_n	I/O		
AE2	pci_req_n	I/O		
AF3	pci_gnt_n	Input		
AD4	pci_perr_n	I/O		
AF2	pci_serr_n	I/O		
AE3	pci_inta_n	Output		
AP31	clk_vo	Input	VO clock input to generate internal four video output ports clocks	Video output interface
AP29	vo_clk0	Output	VO clock output for video output ports 0 to 3	
AM32	vo_data_0[0]	I/O	VO SD video output port 0 or VO HD video output port 0[7:0] or VI HD video input port 1 [7:0] with clk_vi_1	
AM34	vo_data_0[1]			
AL34	vo_data_0[2]			
AM31	vo_data_0[3]			
AN31	vo_data_0[4]			
AL30	vo_data_0[5]			
AN32	vo_data_0[6]			
AL29	vo_data_0[7]			
AJ31	vo_data_1[0]		VO SD video output port 1 or VO HD video output port 0[15:8] or VI HD video input port 1 [15:8] with clk_vi_1	
AK32	vo_data_1[1]			
AM33	vo_data_1[2]			
AL33	vo_data_1[3]			
AK31	vo_data_1[4]			
AL32	vo_data_1[5]			
AK30	vo_data_1[6]			
AL31	vo_data_1[7]			

Pin Descriptions (Continued)

BALL NUMBER	SYMBOL	ATTRIBUTE	DESCRIPTIONS	BLOCK
AH33	vo_data_2[0]	I/O	VO HD video output port 1[7:0] or VI HD video input port 3 [7:0] with clk_vi_3	Video output interface
AH32	vo_data_2[1]			
AH31	vo_data_2[2]			
AG31	vo_data_2[3]			
AG32	vo_data_2[4]			
AJ33	vo_data_2[5]			
AK33	vo_data_2[6]			
AJ32	vo_data_2[7]			
AD31	vo_data_3[0]	I/O	VO HD video output port 1[15:8] or VI HD video input port 3[15:8] with clk_vi_3	
AF33	vo_data_3[1]			
AE32	vo_data_3[2]			
AF32	vo_data_3[3]			
AE31	vo_data_3[4]			
AF31	vo_data_3[5]			
AG34	vo_data_3[6]			
AG33	vo_data_3[7]			
AJ34	VO_CLK1	Output	VO clock output for video output ports 4 to 7	
AB31	vo_data_4[0]	I/O	VO SD video output port 2 or VO HD video output port 2[7:0] or VI HD video input port 5[7:0] with clk_vi_5	
AC33	vo_data_4[1]			
AC32	vo_data_4[2]			
AC31	vo_data_4[3]			
AD32	vo_data_4[4]			
AD33	vo_data_4[5]			
AE33	vo_data_4[6]		VO SD video output port 2 or VO HD video output port 2[7:0] or VI HD video input port 5[7:0] with clk_vi_5	
AF34	vo_data_4[7]			
Y31	vo_data_5[0]		VO SD video output port 3 or VO HD video output port 2[15:8] or VI HD video input port 5[15:8] with clk_vi_5	
AA33	vo_data_5[1]			
AA32	vo_data_5[2]			
AA31	vo_data_5[3]			
AB33	vo_data_5[4]			
AC34	vo_data_5[5]			
AD34	vo_data_5[6]			
AB32	vo_data_5[7]			
V32	vo_data_6[0]		VO HD video output port 3[7:0] or VI HD video input port 7[7:0] with clk_vi_7	
V31	vo_data_6[1]			
W33	vo_data_6[2]			
W32	vo_data_6[3]			
Y34	vo_data_6[4]			
W31	vo_data_6[5]			

Pin Descriptions (Continued)

BALL NUMBER	SYMBOL	ATTRIBUTE	DESCRIPTIONS	BLOCK
Y33	vo_data_6[6]	I/O	VO HD video output port 3[7:0] or VI HD video input port 7[7:0] with clk_vi_7	Video output interface
Y32	vo_data_6[7]			
T32	vo_data_7[0]		VO HD video output port 3[15:8] or VI HD video input port 7[15:8] with clk_vi_7	
T33	vo_data_7[1]			
U31	vo_data_7[2]			
U32	vo_data_7[3]			
U33	vo_data_7[4]			
U34	vo_data_7[5]			
V33	vo_data_7[6]			
V34	vo_data_7[7]			
R34	clk_vi_0	Input	VI video input port 0 clock input	Video input interface
N33	vi_data_0[0]	Input	VI SD video input port 0 or **VI SD video byte-interleave input port 0 with 108MHz** or VI HD video input port 0 [7:0]	
P31	vi_data_0[1]			
P32	vi_data_0[2]			
P33	vi_data_0[3]			
R31	vi_data_0[4]			
R32	vi_data_0[5]			
R33	vi_data_0[6]			
T31	vi_data_0[7]			
N34	clk_vi_1	Input	VI video input port 1 clock input	
L32	vi_data_1[0]		VI SD video input port 1 or **VI SD video byte-interleave input port 1 with 108MHz** or VI HD video input port 1 [15:08]	
M31	vi_data_1[1]		VI SD video input port 1 or **VI SD video byte-interleave input port 1 with 108MHz** or VI HD video input port 1 [15:08]	Video input interface
K33	vi_data_1[2]			
L33	vi_data_1[3]			
M32	vi_data_1[4]			
N31	vi_data_1[5]			
M33	vi_data_1[6]			
N32	vi_data_1[7]			
L34	clk_vi_2		VI video input port 2 clock input	
H32	vi_data_2[0]		VI SD video input port 2 or VI HD video input port 2 [7:0]	
H33	vi_data_2[1]			
J31	vi_data_2[2]			
K31	vi_data_2[3]			
J32	vi_data_2[4]			
J33	vi_data_2[5]			
K32	vi_data_2[6]			
L31	vi_data_2[7]			
J34	clk_vi_3		VI video input port 3 clock input	

Pin Descriptions (Continued)

BALL NUMBER	SYMBOL	ATTRIBUTE	DESCRIPTIONS	BLOCK
E33	vi_data_3[0]	input	VI SD video input port 3 or VI HD video input port 2 [15:08]	Video input interface
G31	vi_data_3[1]			
F33	vi_data_3[2]			
F32	vi_data_3[3]			
G32	vi_data_3[4]			
G33	vi_data_3[5]			
H30	vi_data_3[6]			
H31	vi_data_3[7]			
G34	clk_vi_4		VI video input port 0 clock input	
E31	vi_data_4[0]		VI SD video input port 4 or **VI SD video byte-interleave input port 2 with 108MHz** or VI HD video input port 4[7:0]	
D32	vi_data_4[1]			
C33	vi_data_4[2]			
C34	vi_data_4[3]			
D33	vi_data_4[4]			
E32	vi_data_4[5]			
F31	vi_data_4[6]			
E30	vi_data_4[7]			
A32	clk_vi_5	Input	VI video input port 1 clock input	
D29	vi_data_5[0]		VI SD video input port 5 or **VI SD video byte-interleave input port 3 with 108MHz** or VI HD video input port 4 [15:08]	
C30	vi_data_5[1]			
B31	vi_data_5[2]		VI SD video input port 5 or **VI SD video byte-interleave input port 3 with 108MHz** or VI HD video input port 4 [15:08]	Video input interface
D30	vi_data_5[3]			
B32	vi_data_5[4]			
D31	vi_data_5[5]			
C31	vi_data_5[6]			
C32	vi_data_5[7]			
A30	clk_vi_6		VI video input port 2 clock input	
C27	vi_data_6[0]		VI SD video input port 6 or VI HD video input port 6 [7:0]	
B28	vi_data_6[1]			
D27	vi_data_6[2]			
C28	vi_data_6[3]			
D28	vi_data_6[4]			
B29	vi_data_6[5]			
C29	vi_data_6[6]			
B30	vi_data_6[7]			
A28	clk_vi_7		VI video input port 3 clock input	
B25	vi_data_7[0]		VI SD video input port 7 or VI HD video input port 6 [15:08]	
C25	vi_data_7[1]			

Pin Descriptions (Continued)

BALL NUMBER	SYMBOL	ATTRIBUTE	DESCRIPTIONS	BLOCK
D25	vi_data_7[2]	Input	VI SD video input port 7 or VI HD video input port 6 [15:08]	Video input interface
A26	vi_data_7[3]			
B26	vi_data_7[4]			
C26	vi_data_7[5]			
D26	vi_data_7[6]			
B27	vi_data_7[7]			
G2	ACLKP	I/O	AUDIO playback port bit clock, word select, and serial data	Audio playback interface
H4	ASYNP			
H3	ADATP	Output		
B3	ACLKR_0	Input	AUDIO record port bit clock, word select, and serial data (8-channel)	Audio recording interface
E5	ACLKR_1			
D4	ACLKR_2			
C3	ACLKR_3			
E4	ACLKR_4			
D3	ACLKR_5			
C2	ACLKR_6			
F4	ACLKR_7			
A10	ASYNR_0			
B10	ASYNR_1			
B9	ASYNR_2			
A9	ASYNR_3			
B8	ASYNR_4	Input	AUDIO record port bit clock, word select, and serial data (8-channel)	Audio recording interface
C9	ASYNR_5			
C8	ASYNR_6			
D9	ASYNR_7			
E3	ADATAR_0			
D2	ADATAR_1			
C1	ADATAR_2			
E2	ADATAR_3			
F3	ADATAR_4			
F2	ADATAR_5			
G4	ADATAR_6			
G3	ADATAR_7			
A7	clk_csdi	Input	CASCADE VI Clock	Cascade input interface
C4	csdi_data[0]		CASCADE SD VI input[7:0] or CASCADE HD VI input[15:0]	
D5	csdi_data[1]			
B4	csdi_data[2]			
C5	csdi_data[3]			
E6	csdi_data[4]			

Pin Descriptions (Continued)

BALL NUMBER	SYMBOL	ATTRIBUTE	DESCRIPTIONS	BLOCK
D7	csdi_data[5]	Input	CASCADE SD VI input[7:0] or CASCADE HD VI input[15:0]	Cascade input interface
A4	csdi_data[6]			
B5	csdi_data[7]			
D6	csdi_data[8]			
A5	csdi_data[9]			
C6	csdi_data[10]			
C7	csdi_data[11]			
B6	csdi_data[12]			
B7	csdi_data[13]			
E8	csdi_data[14]			
D8	csdi_data[15]			
AP9	clk_csdo	Output	CASCADE VO Clock	Cascade output interface
AN9	csdo_data[0]	Output	CASCADE SD VO output[7:0] or CASCADE HD VO output[15:0]	
AM9	csdo_data[1]			
AN8	csdo_data[2]			
AM8	csdo_data[3]			
AN7	csdo_data[4]			
AP6	csdo_data[5]			
AL8	csdo_data[6]			
AM7	csdo_data[7]			
AN6	csdo_data[8]			
AL7	csdo_data[9]	Output	CASCADE SD VO output[7:0] or CASCADE HD VO output[15:0]	Cascade output interface
AP5	csdo_data[10]			
AM6	csdo_data[11]			
AN5	csdo_data[12]			
AP4	csdo_data[13]			
AL6	csdo_data[14]			
AN4	csdo_data[15]			
AH3	jtag_tck	Input	JTAG test pins. All 4 input pins should be tied to LOW for normal operation.	ARM JTAG Interface
AH4	jtag_trstn	Input		
AJ3	jtag_tdi	Input		
AK2	jtag_tms	Input		
AL1	jtag_tdo	Output		
AJ4	jtag_rtck	Output		
AG4	sdo_chip	Output		
AG2	i2c_scl	I/O	I^2C interface	Peripheral Interface
AH2	i2c_sda			
AP3	uart0_tx / gpio[18]	I/O	UART #0, while using as UART, the direction of the uart0_tx should be set to output, while uart0_rx should be set to input	
AM4	uart0_rx / gpio[19]			

Pin Descriptions (Continued)

BALL NUMBER	SYMBOL	ATTRIBUTE	DESCRIPTIONS	BLOCK
AL4	uart1_tx / gpio[20]	I/O	UART #1, while using as UART, the direction of the uart0_tx should be set to output, while uart0_rx should be set to input	Peripheral Interface
AN3	uart1_rx / gpio[21]			
AM30	gpio[0]		General Purpose Port [28:0]. GPIO[21:18] port share with UART#0/#1	
AN30	gpio[1]			
AM29	gpio[2]			
AN29	gpio[3]			
AL27	gpio[4]			
AL28	gpio[5]			
AM27	gpio[6]			
AM28	gpio[7]			
AN27	gpio[8]			
AN28	gpio[9]			
AL26	gpio[10]			
AM26	gpio[11]			
AP27	gpio[12]			
AN26	gpio[13]			
AP26	gpio[14]			
AL25	gpio[15]			
AM5	gpio[16]			
AL5	gpio[17]		General Purpose Port [28:0]. GPIO[21:18] port share with UART#0/#1	
AM2	gpio[22]			
AM3	gpio[23]			
AL3	gpio[24]			
AK4	gpio[25]			
AM1	gpio[26]			
AL2	gpio[27]			
AK3	gpio[28]			

Pin Descriptions (Continued)

BALL NUMBER	SYMBOL	ATTRIBUTE	DESCRIPTIONS	BLOCK
A1, A34, AP1, AP34	NC		No Ball Connection (N/C)	Power
E1, H1, L1, V1, AE1	AVDD	1.2V	Power of analog PLL	
F1, J1, M1, W1, AF1	AVSS	GND	Ground of analog PLL	
E10, E25, F30, J30, K5, L5, N14, N16, N20, N30, P14, P16, P20, U13, U14, U21, U22, U30, V5, V13, V14, V21, V22, V30, W5, AA5, AA14, AA16, AA20, AA30, AB14, AB16, AB20, AC5, AC30, AE5, AF30, AG5, AK5, AK8, AK10, AK25, AK27, AP30	VDD	1.2V	Power of core logic	
A13, B11, B16, B18, B24, C14, C21, E12, E15, E18, E20, E21, E23, N13, N17, N19, N22, P13, P17, P19, P22, AA13, AA17, AA19, AA22, AB13, AB17, AB19, AB22, AK12, AK15, AK18, AK20, AK21, AK23, AM14, AM21, AN11, AN16, AN18, AN24, AP13	SSTL_VDE SSTL_VDE1 SSTL_VDE2	1.8V	Power of the DDR I/O	
A6, A27, A29, D1, D34, E7, E9, F34, H5, H34, J5, K1, K30, M30, M34, P1, P5, P30, R2, R5, T13, T14, T21, T22, T30, T34, U5, V2, W2, W13, W14, W21, W22, Y5, Y30, AA34, AB1, AB2, AB5, AB30, AC2, AE30, AE34, AG1, AG30, AJ2, AJ5, AJ30, AK1, AK6, AK34, AL9, AP7, AP10, AP28, AP32	VDE33	3.3V	Power of the external I/O	
A2, A3, A8, A11, A16, A18, A25, A31, A33, B1, B2, B33, B34, C10, C24, D10, D24, E19, E22, E24, E26, E27, E28, E29, F5, G1, G5, G30, K34, L30, M5, N2, N5, N15, N18, N21, P15, P18, P21, P34, R13, R14, R15, R16, R17, R18, R19, R20, R21, R22, R30, T1, T2, T5, T15, T16, T17, T18, T19, T20, U1, U15, U16, U17, U18, U19, U20, V3, V15, V16, V17, V18, V19, V20, W15, W16, W17, W18, W19, W20, W30, W34, Y1, Y2, Y13, Y14, Y15, Y16, Y17, Y18, Y19, Y20, Y21, Y22, AA15, AA18, AA21, AB15, AB18, AB21, AB34, AD1, AD5, AD30, AF5, AH1, AH5, AH30, AH34, AK7, AK9, AK19, AK22, AK24, AK26, AK28, AK29, AL10, AL24, AM10, AM24, AN1, AN2, AN10, AN33, AN34, AP2, AP8, AP11, AP16, AP18, AP25, AP33	VSS	-	Ground	

Absolute Maximum Ratings

VDD 1.2V (DVDI) . . . . . -0.5V to 1.8V
PLL AVD (AVDD) . . . . . -0.5V to 1.8V
DDR AVD 1.8V (SSTL_VDE) . . . . . -0.5V to 2.5V
VDD 3.3V (DVDE) . . . . . -0.5V to 4.6V
Voltage on Any Digital Pin (see caution statement). . . . . 3.8V (Max)
Voltage on OSC Related Analog Pin. . . . . 3.8V (Max)

Thermal Information

Thermal Resistance (Typical)	θ_{JA} (°C/W)	θ_{JC} (°C/W)
676 Ld PBGA Package	13.5	3.5

Junction Temperature (Plastic Package). . . . . -40°C to +125°C
Storage Temperature Range. . . . . -55°C to +125°C
Pb-free Reflow Profile . . . . . see TB493

Recommended Operating Conditions

Ambient Temperature . . . . . 0°C to +70°C
VDD 1.2V (DVDI) . . . . . 1.15V to 1.25V
PLL AVD (AVDD) . . . . . 1.15V to 1.25V
DDR AVD (SSTL_VDE). . . . . 1.7V to 1.9V
VDE 3.3V (DVDE). . . . . 3V to 3.6V

CAUTION: Do not operate at or near the maximum ratings listed for extended periods of time. Exposure to such conditions may adversely impact product reliability and result in failures not covered by warranty.

Electrical Specifications

T_A = 0°C to +70°C. **Boldface limits apply across the operating temperature range, 0°C to +70°C.**

PARAMETER	SYMBOL	MIN (Note 3)	TYP	MAX (Note 3)	UNIT
SUPPLY CURRENT AND POWER DISSIPATION					
PLL Supply Current (1.2V nominal)	I_{DDP}		23		mA
SSTL 1.8V Supply Current (1.8V nominal)	I_{DDH}		391		mA
Digital Internal Supply Current (1.2V nominal)	I_{DDI}		3174		mA
Digital I/O Supply Current (3.3V nominal)	I_{DDO}		69		mA
Total Power Dissipation	P_D		4.7		W
DC CHARACTERISTICS					
Digital Inputs					
Input High Voltage (TTL)	V_{IH}	**2.0**		**V_{DVDE} + 0.3**	V
Input Low Voltage (TTL)	V_{IL}	**–0.3**		**0.8**	V
Input Leakage Current (at V_I = 3.3V or 0V)	I_L			**± 4**	µA
Input Capacitance	C_{IN}			**16**	pF
Digital Outputs					
Output High Voltage	V_{OH}	**V_{DDE} – 0.2**		**V_{DVDE}**	V
Output Low Voltage	V_{OL}	**0**		**0.2**	V
High Level Output Current (at V_{OH} = 2.8V)	I_{OH}	*1			mA
Low Level Output Current (at V_{OL} = 0.2V)	I_{OL}				mA
Tri-state Output Leakage Current (at V_O = 2.5V or 0V)	I_{OZ}			**± 4**	µA
Output Capacitance	C_O			**16**	pF

NOTE:
3. Compliance to datasheet limits is assured by one or more methods: production test, characterization and/or design.

Input/Output Capacitance

PARAMETER	SYMBOL	REQUIREMENTS	UNIT
Input Pin	CIN	Max 16	pF
Output Pin	COUT	Max 16	pF
I/O Pin	CI/O	Max 16	pF

General Description

TW2819 is a highly integrated multichannel H.264 codec targeting security surveillance market. The video codec is designed to work in full duplex mode so it is capable of doing digital video compressing and decompressing simultaneously. TW2819 uses a tightly pipelined hardware solution to guarantee its performance benchmark. The embedded ARM926 microprocessor enables TW2819 to tailor various customized sets of functions such as various bit rates, different frame rates, and different resolutions for each channel. Customers can change TW2819 configuration parameters to meet their own needs. These parameters include a video channel number, bit rate, frame rate, NTSC/PAL, progressive/interleave mode, encode/decode mode, etc.

Peripheral Interface

TW2819 implements a few peripheral interfaces: video capture, video display, audio I/F, memory I/F, and PCI I/F. These interfaces are described in more detail in this section.

VIDEO INPUT I/F

SD/Video Input I/F

There are 8 SD/ video input ports with their own clocks. The video input format is compliant to BT.656.

When receiving SD video, each port can take byte-interleaved video input at 27MHz, 54MHz, or 108MHz. When receiving WD1 video, each port can take byte-interleaved video input at 36MHz, 72MHz, or 144MHz.

In the maximum video input capacity, TW2819 video input ports can take a 16-channel SD or WD1 data. It is not possible for all 8 video input ports working at 108MHz or 144MHz.

HD Video Input I/F

There are 4 HD video input ports with their own clocks. The video input format is compliant to BT.1120. These ports pins are shared with SD/video input ports pins.

When receiving HD video, each port can take single channel or byte-interleaved video input at 74.25MHz or 148.5MHz.

VIDEO I/O (INPUT/OUTPUT) I/F

There are four video input/output bi-directional ports. These ports can be configured as video input or video output ports.

When these video I/F ports are used for video output, they can be used for either BT.656 or BT.1120 mode. In the maximum video output capacity, TW2819 can send out 4CH SD//HD video data.

When these video I/F ports are used for video input, they can only be in BT.1120 HD mode and receive up to 4CH HD data. They will not be used for SD/video input. Please note that total HD video input channels can only be 8 when both video input/output ports and video input ports are used at the same time.

In the maximum video input capacity, TW2819 can take 8CH channel HD data through its video inputs ports.

AUDIO I/F

The CODEC has two sets of digital audio I2S interface. One is the audio input interface operating as slave mode. The other is the audio output interface operating as master mode.

DDR I/F

The external DDR2 SDRAMs are used as memory for storing video and audio information during encoding and decoding processes. The codec supports two sets of independently controlled 1Gb DDR2 SDRAM with 16-bit data bus.

PCI I/F

TW2819 uses PCI bus to communicate with the host. The bitstream from encoder output is stored to the hard disk via the PCI bus. The encoded bitstream is transferred to TW2819 decoder via PCI bus in the playback mode. The PCI clock can be either 33MHz or 66MHz.

Top Level Block Diagram

Figure 6 on page 22 shows the top level codec block diagram. The ARM926 handles both A/V synchronization and slice-layer-above video processing. The major video codec task is done by the tightly coupled macroblock video pipeline. There is an internal 64-bit memory data bus connecting most of the codec modules with the external DDR2 memory. The ARM processor has AHB bus interface. The codec implements its own internal 32-bit control bus for global on-chip register access. There is an interface bridge to convert AHB protocol to the internal control bus. The external memory interface needs to operate at 432MHz in order to provide enough bandwidth.

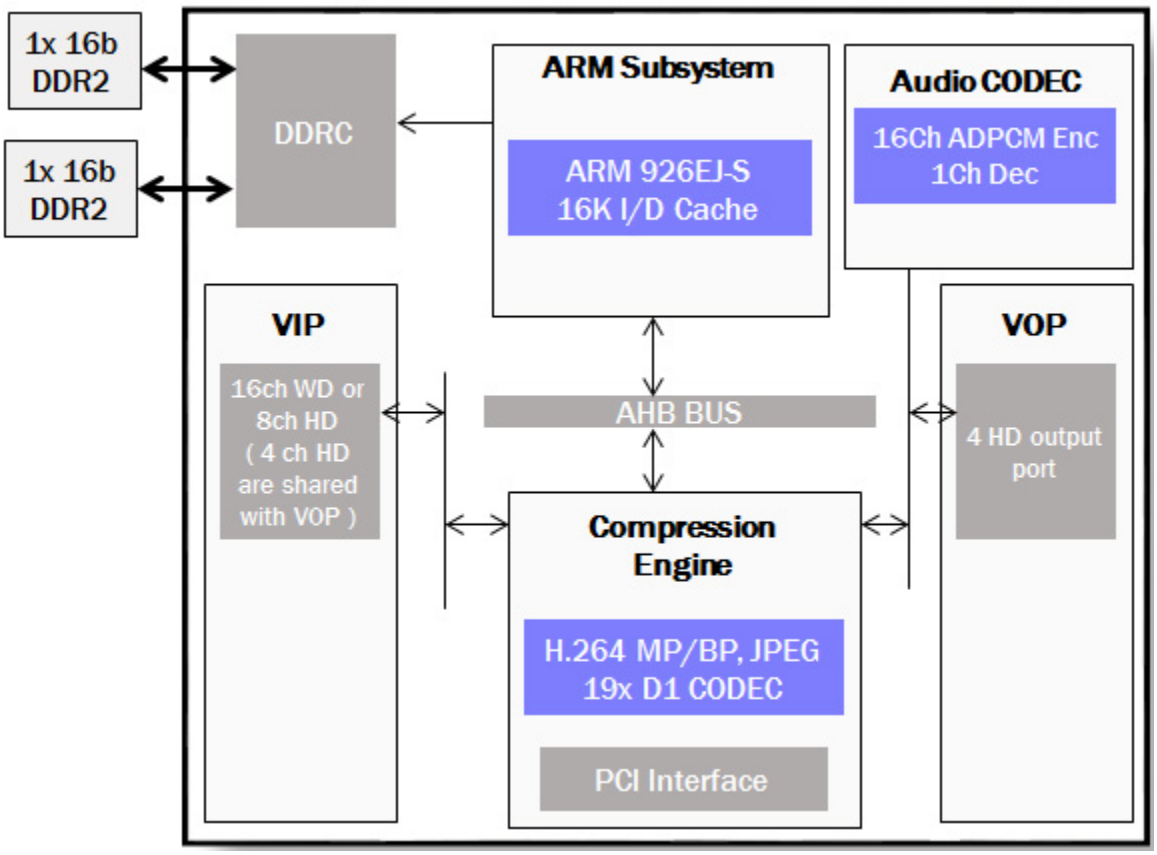


FIGURE 6. TW2819 BLOCK DIAGRAM

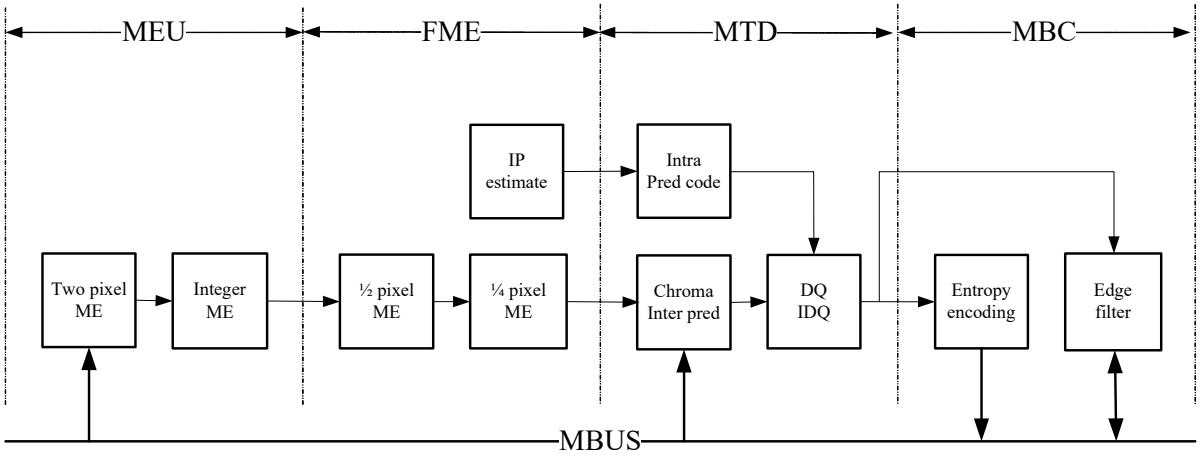


FIGURE 7. ENCODER HW PIPELINE

ENCODING BLOCK DIAGRAM

TW2819 hardwired video encoder is designed to compress a sequence of YUV 4:2:0 pictures to a single compressed video bitstream. It supports various resolutions from CIF (352x240) up to HD (1920X1080). The motion vector is in ¼ pixel accuracy. All intra-prediction modes are supported. The firmware in the embedded microprocessor is responsible for rate control scheme, such as CBR and VBR. In the CBR mode, the QP value can be adjusted at each macroblock. The visual subjective video quality is improved by implementing programmable in-loop filter.

The video encoding is divided into a series of processing steps for each 16x16 macroblock for each video frame. The initial step is to generate the prediction for the current macroblock. There are two kinds of predictions: intra prediction (spatial prediction using encoded current pix) and inter prediction (temporal prediction using previously encoded pictures). The residual difference between current block and the predicated block goes through transformation and then quantized. The results will be coded into H.264 bitstream using content adaptive variable length coding (CAVLC) method. Meanwhile, the reconstructed macroblock is calculated by applying inverse quantization and transformed to the quantized coefficients. An in-loop deblocking filter is applied to the reconstructed macroblock before it is stored in the reference frame buffer.

The H.264 encoding requires many of its operation in sequence, which imposes a computation challenge for ASIC design. In order to meet this challenge, the codec simplifies certain parts of sequence operation into parallel operation without introducing noticeable artifacts. A macroblock level pipeline is implemented in TW2819 to break down one macroblock processing it into smaller tasks and making it easier for hardware implementation. As shown in Figure 7, the macroblock pipeline consists of 4 stages: the Motion Estimation Unit (*MEU*), the Fractional Motion Estimation (*FME*), Macroblock Type Decision (*MTD*), and Macroblock Coding (*MBC*). Each stage has to complete all assigned tasks within one macroblock processing period.

Each macroblock pipeline can be further partitioned into several sub-bocks. During the encoding process, all four macroblock pipelines are concurrently processing 4 consecutive macroblock data in raster-scan order.

Figure 8 illustrates encoding macroblock pipeline schedule in the time domain.

For each of these processes, the ARM must set up parameters and monitor events communicated by interrupts. The microprocessor involvement is limited, in the normal operation mode, to the slice-layer-above processing with the exception of video bit rate control. The rate control adjusts the quantization value at macroblock level.

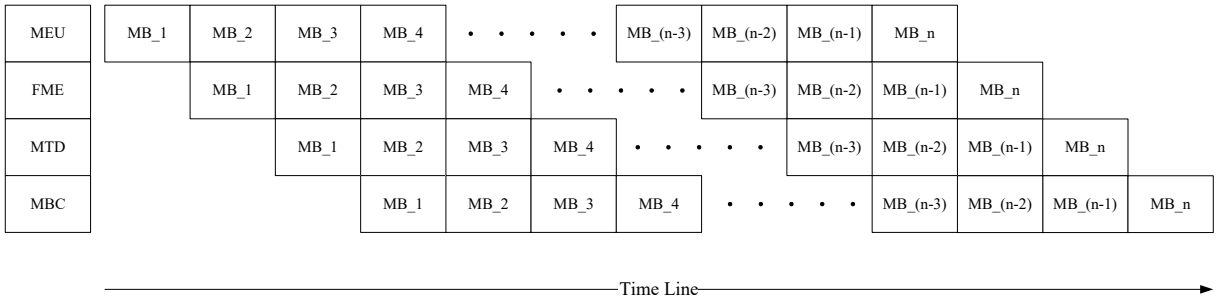


FIGURE 8. ENCODER MACROBLOCK PIPELINE SCHEDULE

DECODING BLOCK DIAGRAM

The major video decompression task is done by the tightly coupled macroblock pipeline that consists of Variable Length Decoding (***VLD***), Block Transform Unit for idct/iq (***BTU***), Intra/Inter Prediction (***IPD***), and Edge Filter (***EGF***).

The decoding macroblock pipeline is able to decode in its entirety a video bitstream from slice layer downwards. The ARM must decode the higher layers in order to extract the information needed for decoding and appropriately set up the registers.

The codec implements a start code detector that is a few slices in advance of macroblock decoding pipeline. This start code detector parses the bitstream and locates start codes corresponding to slice layer and above. When one of these start-codes is found, the start code detectors stops and raises an interrupt to the ARM. The ARM is then able to read the header data following the start code. The decoded header parameters will be programmed into hardware shared parameter data structure that will be discussed in detail later in this document.

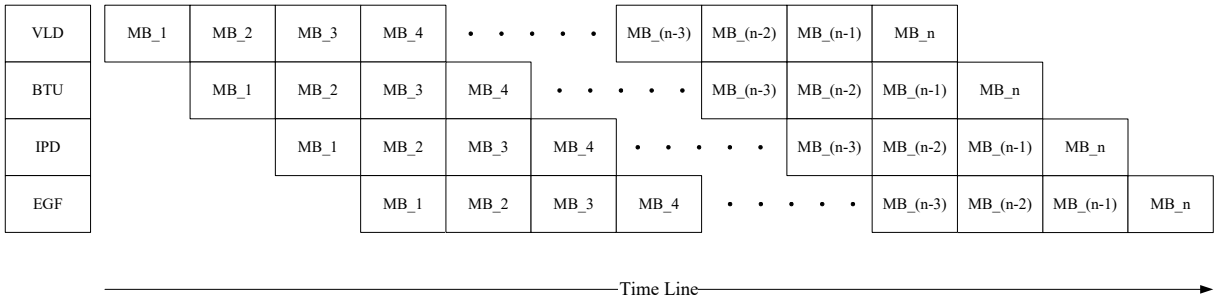


FIGURE 9. DECODER BLOCK DIAGRAM

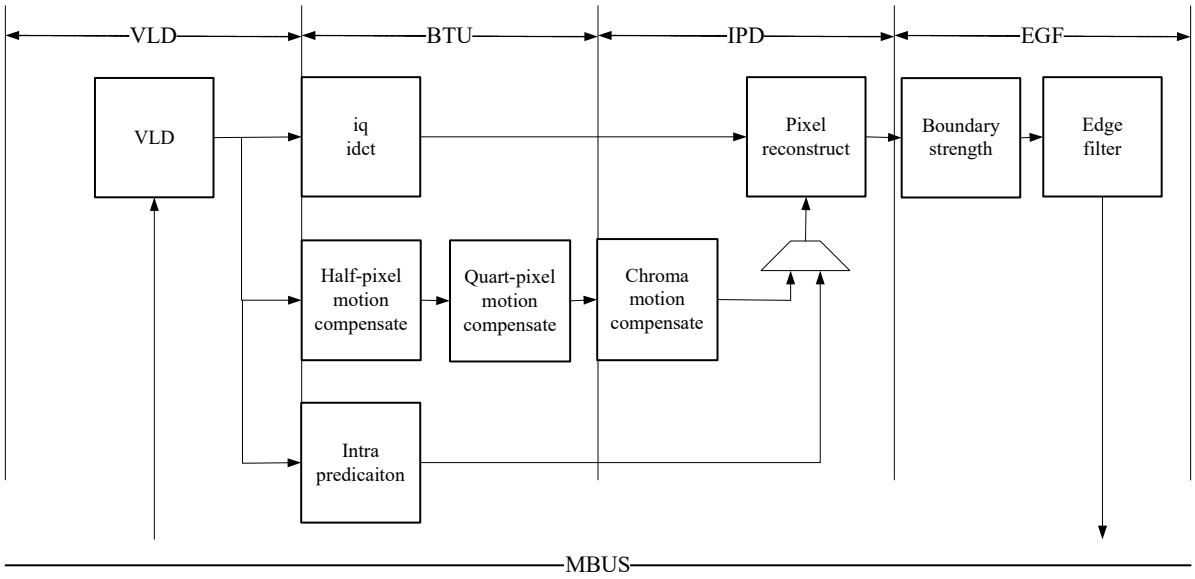


FIGURE 10. DECODER MACROBLOCK PIPELINE SCHEDULE

Video CODEC Configuration and Interfaces

TW2819 can be configured differently for its video codec operation. Customers can set resolution and frame rate differently for primary and secondary video channels. This section provides guidelines for the customer to configure TW2819 video correctly.

Primary Video Stream

There are some limitations for TW2819 primary video processing. Customers should be careful not to violate any of these limitations.

TW2819 MAXIMUM VIDEO COMPRESSION POWER

TW2819 compression capability depends on various factors, such as the input frame rate, the core clock frequency, the DDR clock frequency, and other on-chip features enabled, such as JPEG encoding, etc. The raw compression power of TW2819 is roughly equivalent to a 19-channel of full frame rate D1 encoding, with core clock running at maximum of 360MHz.

TW2819 MAXIMUM EXTERNAL FRAME BUFFER NUMBER

The user should be aware that external frame buffer size is decided by the resolution of the video channel. It has nothing to do with the frame rate. For example, the frame buffer size for two SD at 15fps is two times larger than one SD at 30fps, even though the computation power remains roughly the same. With 1Gb DDR as external memory, there is a limitation of maximum frame buffer number.

Network Secondary Video Stream

TW2819 supports network stream encoding. The compression format can be either H.264 or JPEG.

Audio Processing

The audio processing unit (APU) handles audio compression and decompression. For simple speech codec, there is no need to support multi-format audio compression; instead a single ADPCM format codec is implemented using hardware approach.

ADPCM is a variant of DPCM that varies the size of the quantization step, to allow further reduction of the required bandwidth for a given signal to noise ratio. It is used to map a series of 8-bit μ-law (or a-law) PCM samples into a series of 4-bit ADPCM samples (The ADPCM compression ratio is 4:1, but TW2819 will take each 8-bit sample and extend it to 16-bit before compression).

The audio compressed bitstream is packed into packets. The size of each packet is 188 Bytes with an embedded 28-bit packet header. The actual audio compressed bitstream size is 1476-bit or 369 audio samples.

I2S Protocol

The I2S bus is a serial bus consisting of three lines: serial clock, word selection and serial data. The digital audio input interface is working as slave mode. The digital audio output interface is working as master mode. The codec shall generate both serial clock and word select when it operates as master mode. It receives both serial clock and word select when it operates as slave mode.

The CODEC uses two set I2S pins to transmit or receive multichannel digital audio data.

Multichannel Audio Protocol

The serial clock (*sck*) depends on audio sample rate, audio sample bit width and how many audio channels are occupied for current configuration. Different audio channels may have different sample rate and sample bit width. In order to simplify the audio codec design, Equation 1 is used to calculate the audio serial clock.

$$Sck = 256 \times sample\_rate(Max) \quad \text{(EQ. 1)}$$

Audio Serial Clock

Where sample_rate(Max) is the maximum audio sample rate among all. For example, a 4-channel audio with maximum audio sample rate of 8kHz sample rate and maximum bit width of 16-bit, the serial clock should be 2,048kHz. In Equation 1, it is assumed that each word select cycle covers 64 bits. The codec audios are all mono channel. There is no need for the codec to support stereo audio. The audio channels are evenly divided to fit into left and right channel space.

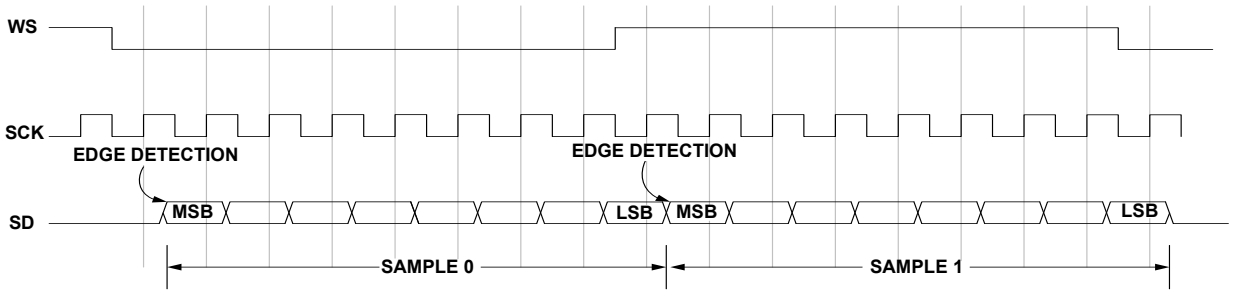


FIGURE 11. I2S PROTOCOL

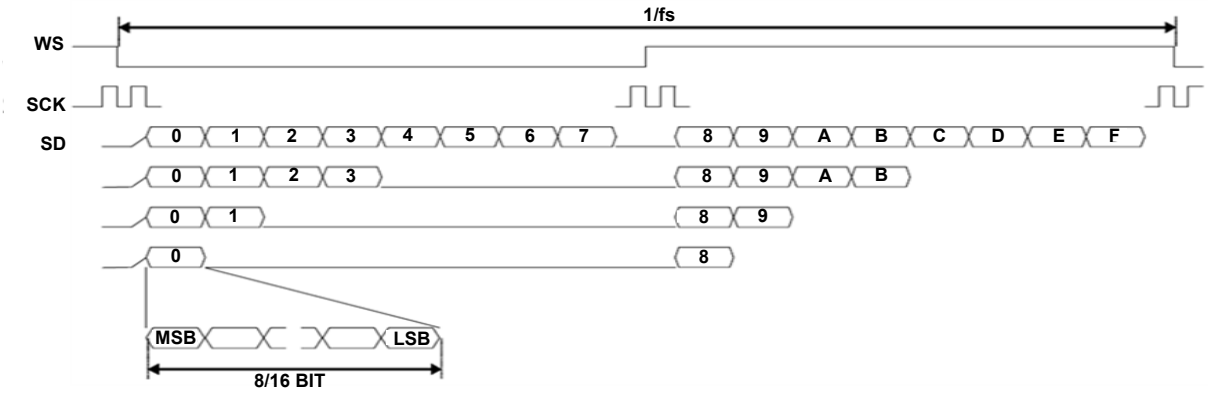


FIGURE 12. MULTICHANNEL DIGITAL AUDIO INTERFACE

Figure 12 shows an example of 4 channels digital audio input protocol. It is noted that only first 4 bits after word select transition are valid. The remaining bits in the word select are not used. ADPCM is a variant of DPCM that varies the size of the quantization step, to allow further reduction of the required bandwidth for a given signal to noise ratio. It is used to map a series of 8-bit μ-law (or a-law) PCM samples into a series of 4-bit ADPCM samples (The ADPCM compression ratio is 4:1, but TW2819 will take each 8-bit sample and extend it to 16-bit before compression).

The audio compressed bitstream is packed into packets. The size of each packet is 188 Bytes with an embedded 28-bit packet header. The actual audio compressed bitstream size is 1476-bit or 369 audio samples.

PLL Clock Scheme

TW2819 takes on-board oscillator as reference clock to all embedded PLL. The oscillator input clock can be 27MHz or 54MHz. However, internal PLL reference clock is fixed to 27MHz. If oscillator input clock is 54MHz, it has to be divided by half before it is fed to PLL.

There are two registers for PLL clock setup. There are a total of 5 embedded analog PLL: one PLL for video output, one PLL for ARM9, one PLL for core clock, two PLL for DDR. The following two registers are used for TW2819 PLL setup. These registers need to be programmed in the TW2819 boot-up phase.

TW2819 Register Overview

TW2819 Memory Map

TW2819 memory map is partitioned into two distinct segments: one for the external DDR memory and the other for the on-chip global control registers.

TABLE 1. SOC MEMORY MAP

SEGMENT	START_ADDRESS	END_ADDRESS
MBUS	32'h0000_0000	32'h3FFF_FFFF
CBUS	32'h8000_0000	32'h8000_1FFF

The external DDR access supports both single cycle and burst type. ARM926EJS burst length is always 32 bytes. The CBUS supports up to 16 clients with each client takes 256 Byte space. Table 2 shows current CBUS memory map.

TABLE 2. CBUS MEMORY MAP

MODULE NAME	START ADDRESS	END ADDRESS
DBG	32'h8000_0000	32'h8000_00FF
SPR	32'h8000_0100	32'h8000_01FF
VLD	32'h8000_0200	32'h8000_02FF
IPD	32'h8000_0300	32'h8000_03FF
IT	32'h8000_0400	32'h8000_04FF
EGF	32'h8000_0500	32'h8000_05FF
VIF	32'h8000_0600	32'h8000_06FF
TSM	32'h8000_0700	32'h8000_07FF
TME	32'h8000_0800	32'h8000_08FF
RTB	32'h8000_0900	32'h8000_09FF
HPM	32'h8000_0A00	32'h8000_0AFF
MBC	32'h8000_0B00	32'h8000_0BFF
VLC	32'h8000_0C00	32'h8000_0CFF
DCM	32'h8000_0D00	32'h8000_0DFF
HIF	32'h8000_0E00	32'h8000_0EFF
CTR	32'h8000_0F00	32'h8000_0FFF
GPIO	32'h8000_1C00	32'h8000_1CFF
UART0	32'h8000_1D00	32'h8000_1DFF
UART1	32'h8000_1E00	32'h8000_1EFF

Interrupt Scheme

The interrupt protocol is outlined in this section. By default, the first register of each hardware module should be the register that defines the interrupt enable and status for the module.

INTERRUPT REGISTER

Address: Module base address + 0x00

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
															status
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															enable

[31:17] **Reserved**

[16] **Status** – interrupt status. HW set this bit to HIGH to initiate interrupt. FW write HIGHT to clear interrupt

[15:1] **Reserved**

[0] **Enable** – interrupt enable. 0: interrupt disable. 1: interrupt enable. Default is 1.

INTERRUPT PROTOCOL

HW generates an interrupt signal to FW when the interrupt source exist and the interrupt is enabled by the FW. When FW detects the interrupt, it gets into interrupt service routine. When it is done, FW writes HIGH to the *status* bit to clear the interrupt. When HW detects that FW is writing 1 to *status* bit, it shall clear the interrupt source.

PCI Register Definitions

This section describes PCI module registers. Normally, any PCI core supports the following registers. TW2819 defines some value in the registers such as device ID, vendor ID, class code, and so on. If customers want to know more detail information, they can check the configuration space section in “PCI local Bus Specification” document.

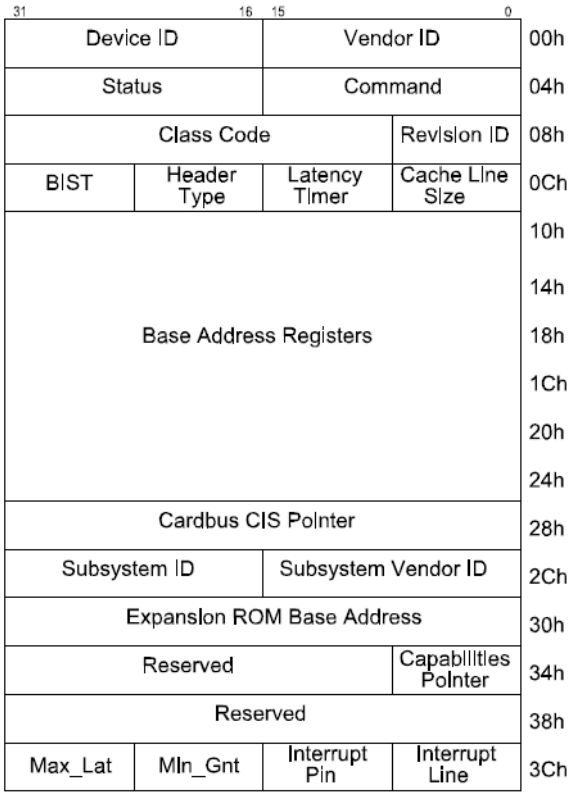


FIGURE 13. BASE ADDRESS REGISTERS

PC Register 00

Address: PCI Base address + 0x00
Type: Read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Device ID															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Vendor ID															

[31:16]: **Device ID** – It specifies TW2819 PCI device ID
Default value: 0x2819
[15:0]**: Vendor ID** – It specifies TW2819 PCI vendor ID
Default value: 0x1719

PCI Register 04

Address: PCI Base address + 0x04
Type: Read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Status															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Command															

[31:16]**: Status** – It specifies TW2819 PCI status: 66MHz capable, DEVSEL timing is medium
Default value: 0x02A0
[15:0]: **Command** – It specifies TW2819 PCI command: TW2819 uses memory space only, but no I/O space.
Default value: 0x0007

PCI Register 08

Address: PCI Base address + 0x08
Type: Read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Class Code															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Class Code								Revision ID							

[31:8]: **Class code** – It specifies TW2819 PCI class code
Default value: 0x048000
[7:0]: **Revision ID** – It specifies TW2819 PCI revision ID
0: TW2819A1/2

PCI Register 0c

Address: PCI Base address + 0x0c

Type: Read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BIST								Header type							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Latency timer								Cache line size							

[31:24]: **BIST** – It specifies TW2819 PCI BIST
Default value: 0x00
[23:16]: **Header type** – It specifies TW2819 PCI header type
Default value: 0x00
[15:8]: **Latency timer** – It specifies TW2819 PCI latency timer
Default value: 0x00
[7:0]: **Cache line size** – It specifies TW2819 PCI cache line size
Default value: 0x00

PCI Register 10-24

Address: PCI Base address + 0x10 – 0x24

Type: Read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Base address registers															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Register 10 (Bart #0) – TW2819 has 64k byte memory space
Register 14 (Bart #1) – Reserved
Register 18 (Bart #2) – Reserved
Register 1c (Bart #3) – Reserved
Register 20 (Bart #4) – Reserved
Register 24 (Bart #5) – Reserved

PCI Register 28

Address: PCI Base address + 0x28

Type: Read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Cardbus CIS Pointer															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

[31:0]: **Cardbus CIS Pointer** – It specifies TW2819 PCI Cardbus CIS pointer
Default value: 0x00

PCI Register 2c

Address: PCI Base address + 0x2c

Type: Read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Subsystem ID															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Subsystem Vendor ID															

[31:16]: **Subsystem ID** – It specifies TW2819 PCI subsystem ID

Default value: 0x2819

[15:0]: **Subsystem vendor ID** – It specifies TW2819 PCI subsystem vendor ID

Default value: 0x1719

PCI Register 30

Address: PCI Base address + 0x30

Type: Read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Expansion ROM Base Address															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

[31:0]: **Expansion ROM base address** – It specifies TW2819 PCI expansion ROM base address

Default value: 0x00

PCI Register 34

Address: PCI Base address + 0x34

Type: Read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Capabilities pointers							

[31:8]: Reserved

[7:0]: **Capabilities pointer** – It specifies TW2819 PCI capabilities pointer

Default value: 0x00

PCI Register 38

Address: PCI Base address + 0x38

Type: Read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

[31:0]: Reserved

PCI Register 3c

Address: PCI Base address + 0x3c

Type: Read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Max_lat								Min grant							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Interrupt pin								Interrupt line							

[31:24]: **Max_lat**– It specifies TW2819 PCI max_lat
Default value: 48
[23:16]: **Min_grant** – It specifies TW2819 PCI min_grant
Default value: 20
[15:8]: **Interrupt pin** – It specifies TW2819 PCI interrupt pin
Default value: 01
[7:0]: **Interrupt line** – It specifies TW2819 PCI interrupt line
Default value: 00

PCI Special Registers

PLL Setting Register 0

Address: PCI base address + 0x00

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PLL_M3								PLL_M2							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL_M1								PLL_M0							

[31:24]: PLL_M3 – It specifies PLL setting for ARM
[23:16]: PLL_M2 – It specifies PLL setting for MEM
[15:08]: PLL_M1 – It specifies PLL setting for CORE
[07:00]: PLL_M0 – It specifies PLL setting for DDR

Description:

PLL_M[7]: This is IDIVX field for PLL. In current TW2819 application, this field is always set to HIGH.

PLL_M[6:0]: These are IDIVFB fields for PLL. It is the multiply factor used in all PLL.

PLL Setting Register 1

Address: PCI base address + 0x04

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		refclk		vi_clk_1to4_inv											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		ahb_sel		mem_sel		arm_sel				core_sel				ddr_sel	

[31:30]: **Reserved**

[29]: **refclk** - This bit is to adaptive for oscillator input frequency. It is set to HIGH for 27MHz input clock. It is set for LOW for 54MHz input clock.

[28]: **Reserved**

[27:24]: **vi_clk_1to4_inv** – It specifies if video input clocks need to be inverted or not.

[24]: It is set to LOW to take vi1_clk_in. It is set to HIGH to take inverted vi1_clk_in.

[25]: It is set to LOW to take vi2_clk_in. It is set to HIGH to take inverted vi2_clk_in.

[26]: It is set to LOW to take vi3_clk_in. It is set to HIGH to take inverted vi3_clk_in.

[27]: It is set to LOW to take vi4_clk_in. It is set to HIGH to take inverted vi4_clk_in.

[13:12]: **ahb_sel** - It specifies the ratio between AHB bus clock and ARM core clock

[13:12] = 2'b00: ahb_clk = arm_clk/2

[13:12] = 2'b01: ahb_clk = arm_clk

[13:12] = 2'b10: ahb_clk = arm_clk/3

[13:12] = 2'b11: ahb_clk = arm_clk/4

[11:10]: **mem_sel** – Selects the ddr1 clock division from the ddr1 clock PLL

[11:0] = 2'b01: ddr1_clk = ddr1_pll_clk/2

[11:10] = 2'b10: ddr1_clk = ddr1_pll_clk/4

[9:8]: **arm_sel** - Selects the arm clock division from the arm clock PLL

[9:8] = 2'b01:arm_clk = arm_pll_clk/2

[9:8] = 2'b10: arm_clk = arm_pll_clk/4

[5:4]: **core_sel** - Selects the core clock division from the core clock PLL

[5:4] = 2'b01: core clock = core_pll_clk/2

[5:4] = 2'b10: core clock = core_pll_clk/4

[1:0]: **ddr_sel** - Selects the ddr0 clock division from DDR0 PLL

[1:0] = 2'b01: the ddr0_clk = ddr0_pll_clk/2.

[1:0] = 2'b10: the ddr0_clk = ddr0_pll_clk/4.

VO Output Clock Control Register

Address: PCI base address + 0x08

Type: R/W

31	30	29	28	27	26	25	24
						vo2_clk_sel[2]	vo1_clk_sel[2]
23	22	21	20	19	18	17	16
vo2_out_delay_sel				vo1_out_delay_sel			
15	14	13	12	11	10	9	8
vo2_clk_sel[1:0]		vo1_clk_sel[1:0]		vo2_out_sclkd_sel		vo1_out_sclkd_sel	
7	6	5	4	3	2	1	0
	vo2_core_clk_sel			vo1_clk_inv	vo1_core_clk_sel		

[31:26]: **Reserved**

[23:20]: **vo2_out_delay_sel** – It specifies video output #2 clock delay control

[23:20] = 4'b0000: Bypass (vo2_out_clk)
[23:20] = 4'b0001: One buffer delay
[23:20] = 4'b0010: Two buffer delay
[23:20] = 4'b0011: Three buffer delay
[23:20] = 4'b0100: Four buffer delay
[23:20] = 4'b0101: Five buffer delay
[23:20] = 4'b0110: Six buffer delay
[23:20] = 4'b0111: Seven buffer delay
[23:20] = 4'b1000: Eight buffer delay
[23:20] = 4'b1001: Nine buffer delay
[23:20] = 4'b1010: Ten buffer delay
[23:20] = 4'b1011: Eleven buffer delay
[23:20] = 4'b1100: Twelve buffer delay
[23:20] = 4'b1101: Thirteen buffer delay
[23:20] = 4'b1110: Fourteen buffer delay
[23:20] = 4'b1111: Fifteen buffer delay

[19:16]: **vo1_out_delay_sel** – It specifies video output #1 clock delay control

[19:16] = 4'b0000: Bypass (vo1_out_clk)
[19:16] = 4'b0001: One buffer delay
[19:16] = 4'b0010: Two buffer delay
[19:16] = 4'b0011: Three buffer delay
[19:16] = 4'b0100: Four buffer delay
[19:16] = 4'b0101: Five buffer delay
[19:16] = 4'b0110: Six buffer delay
[19:16] = 4'b0111: Seven buffer delay
[19:16] = 4'b1000: Eight buffer delay
[19:16] = 4'b1001: Nine buffer delay
[19:16] = 4'b1010: Ten buffer delay
[19:16] = 4'b1011: Eleven buffer delay
[19:16] = 4'b1100: Twelve buffer delay
[19:16] = 4'b1101: Thirteen buffer delay
[19:16] = 4'b1110: Fourteen buffer delay
[19:16] = 4'b1111: Fifteen buffer delay

[25,15:14]: **vo2_out_clk_sel** – It specifies video output block #2 clock selection
[25,15:14] = 3'b000: vo_clk's pll/4
[25,15:14] = 3'b001: vo_clk's pll/8
[25,15:14] = 3'b010: vo_clk's pll/16
[25,15:14] = 3'b011: clk_vo_in
[25,15:14] = 3'b100: vi2_clk_in
[25,15:14] = 3'b101: vi4_clk_in
[25,15:14] = 3'b110: vi6_clk_in
[25,15:14] = 3'b111: vi8_clk_in

[24,13:12]: **vo1_out_clk_sel** – It specifies video output block #1 clock selection
[24,13:12] = 3'b000: vo_clk's pll/4
[24:13:12] = 3'b001: vo_clk's pll/8
[24:13:12] = 3'b010: vo_clk's pll/16
[24:13:12] = 3'b011: clk_vo_in
[24:13:12] = 3'b100: vi2_clk_in
[24:13:12] = 3'b101: vi4_clk_in
[24:13:12] = 3'b110: vi6_clk_in
[24:13:12] = 3'b111: vi8_clk_in

[09:08]: **vo2_out_clkd_sel** – It specifies video output #2 clock stable selection
[09:08] = 2'b00: vo2_out_clk_source generates after vo clock's PLL locking
[09:08] = 2'b01: vo2_out_clk_source generates after one clock delay from vo clock's PLL locking
[09:08] = 2'b10: vo2_out_clk_source generates after two clock delay from vo clock's PLL locking
[09:08] = 2'b11: vo2_out_clk_source generates after three clock delay from vo clock's PLL locking

[09:08]: **vo1_out_clkd_sel** – It specifies video output #1 clock stable selection
[09:08] = 2'b00: vo1_out_clk_source generates after vo clock's PLL locking
[09:08] = 2'b01: vo1_out_clk_source generates after one clock delay from vo clock's PLL locking
[09:08] = 2'b10: vo1_out_clk_source generates after two clock delay from vo clock's PLL locking
[09:08] = 2'b11: vo1_out_clk_source generates after three clock delay from vo clock's PLL locking

[06:04]: **vo2_core_clk_sel** – It specifies video output block #2 clock selection
[06:04] = 3'b000: vo_clk's pll/4
[06:04] = 3'b001: vo_clk's pll/8
[06:04] = 3'b010: vo_clk's pll/16
[06:04] = 3'b011: clk_vo_in
[06:04] = 3'b100: vi2_clk_in
[06:04] = 3'b101: vi4_clk_in
[06:04] = 3'b110: vi6_clk_in
[06:04] = 3'b111: vi8_clk_in

[02:00]: **vo1_core_clk_sel** – It specifies video output block #1 clock selection
[02:00] = 3'b000: vo_clk's pll/4
[02:00] = 3'b001: vo_clk's pll/8
[02:00] = 3'b010: vo_clk's pll/16
[02:00] = 3'b011: clk_vo_in
[02:00] = 3'b100: vi2_clk_in
[02:00] = 3'b101: vi4_clk_in
[02:00] = 3'b110: vi6_clk_in
[02:00] = 3'b111: vi8_clk_in

PLL Setting Register 2

Address:PCI base address + 0x20
Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								PLL_M4							

[31:08]: **Reserved**
[07:00]: **PLL_M4** - It specifies PLL setting for VO clock VO clock.

PCI Core Reset Control Register

Address: PCI base address + 0x24
Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															PCI_RST_EN

[31:08]: **Reserved**
[00:00]: **PCI_RST_EN** – It specifies PCI Core reset will be controlled by software reset after enable.

VO Output Data Select and Clock Control Register

Address: PCI base address + 0x2C

Type: R/W

31	30	29	28	27	26	25	24
vo_pad_in							
23	22	21	20	19	18	17	16
	csd_vo_clk_sel						
15	14	13	12	11	10	9	8
		vi_data_order_sel				votovi_en	
7	6	5	4	3	2	1	0
	vo_pad_ctrl						

[31:31]: **vo_pad_in** – It specifies all of vo output pad will be input pad.

[30:14]: **Reserved**

[13:11] **vi_data_order_sel** – It specifies vi data is remapping with vo data pad.

[13:11] = 3'b000: vo_data[63:00] = {vi_port1,vi_port3,vi_port5,vi_port7}

[13:11] = 3'b001: vo_data[63:00] = {vi_port3,vi_port1,vi_port7,vi_port5}

[13:11] = 3'b010: vo_data[63:00] = {vi_port5,vi_port7,vi_port1,vi_port3}

[13:11] = 3'b011: vo_data[63:00] = {vi_port7,vi_port5,vi_port3,vi_port1}

[10]: **Reserved**

[09:09]: **votovi_en** – It specifies vi data input using vo pad for HD input.

[08:07]: **Reserved**

[06:03] **vo_pad_ctrl** – It specifies vo data pad control

[06:03] = 4'b0000: All of vo data pad are output (default)

[06:03] = 4'b0001: vo data [07:00] pad are input and others are output.

[06:03] = 4'b0010: vo data [15:08] pad are input and others are output.

[06:03] = 4'b0011: vo data [23:16] pad are input and others are output.

[06:03] = 4'b0100: vo data [31:24] pad are input and others are output.

[06:03] = 4'b0101: vo data [39:32] pad are input and others are output.

[06:03] = 4'b0110: vo data [47:40] pad are input and others are output.

[06:03] = 4'b0111: vo data [55:48] pad are input and others are output.

[06:03] = 4'b1000: vo data [63:56] pad are input and others are output.

[06:03] = 4'b1001: vo data [15:00] pad are input and others are output.

[06:03] = 4'b1010: vo data [31:16] pad are input and others are output.

[06:03] = 4'b1011: vo data [47:32] pad are input and others are output.

[06:03] = 4'b1100: vo data [63:48] pad are input and others are output.

[06:03] = v4'b1101: vo data [31:00] pad are input and others are output.

[06:03] = 4'b1110: vo data [63:32] pad are input and others are output.

[02:00]: **Reserved**

VI Data and Clock Remapping Control Register

Address: PCI base address + 0x30

Type: R/W

31	30	29	28	27	26	25	24
23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0
vi_clk_data_remap				remap_vi68	remap_vi57	remap_vi24	remap_vi13

[31:08]: **Reserved**

[07:06]: **vi_clk_data_remap** – It specifies vi clock/data ports remapping.

[07:06] = 2'b00: default

[07:06] =2'b01: (remap_vi68,remap_vi57,remap_vi24,and remap_vi13 are "0")

vi_port1 = vi_port5

vi_port2 = vi_port6

vi_port3 = vi_port7

vi_port4 = vi_port8

vi_port5 = vi_port1

vi_port6 = vi_port2

vi_port7 = vi_port3

vi_port8 = vi_port4

[07:06] = 2'b10:

[07:06] = 2'b11:

[03:03]: **remap_vi68** – It specifies vi port 6 and 8 is remapping with other ports

[07:06] = 2'b00: (default)

[07:06] = 2'b01: vi port 6 and 8 port is reconnected with vi port 4 and port 2.

[07:06] = 2'b10: vi port 6 and 8 port is reconnected with vi port 8 and port 6.

[02:02] **remap_vi57** – It specifies vi port 5 and 7 is remapping with other ports

[07:06] = 2'b00: (default)

[07:06] = 2'b01: vi port 5 and 7 port is remapping with vi port 4 and port 2.

[07:06] = 2'b10: vi port 5 and 7 port is remapping with vi port 7 and port 5.

[01:01] **remap_vi24** – It specifies vi port 2 and 4 is remapping with other ports

[07:06] = 2'b00: (default)

[07:06] = 2'b01: vi port 2 and 4 port is remapping with vi port 8 and port 6.

[07:06] = 2'b10: vi port 2 and 4 port is remapping with vi port 4 and port 2.

[07:06] = 2'b11: vi port 3 and 4 port is remapping with vi port 7 and port 8.

[00:00] **remap_vi13** – It specifies vi port 1 and 3 is remapping with other ports

[07:06] = 2'b00: (default)

[07:06] = 2'b01: vi port 01 and 03 port is remapping with vi port 7 and port 5.

[07:06] = 2'b10: vi port 01 and 03 port is remapping with vi port 3 and port 1.

[07:06] = 2'b11: vi port 09 and 10 port is remapping with vi port 3 and port 4.

PCI Core Reset Control Register

Address: PCI base address + 0x3C

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vi_clk_disable												vi_clk_5to8_inv			

[15:08]: **vi_clk_disable** – It specifies input vi clock disable.

[15:15]: vi clock #8 disable

[14:14]: vi clock #7 disable

[13:13]: vi clock #6 disable

[12:12]: vi clock #5 disable

[11:11]: vi clock #4 disable

[10:10]: vi clock #3 disable

[09:09]: vi clock #2 disable

[08:08]: vi clock #1 disable

[03:00] **vi_clk_5to8_inv** – If video input clocks need to be inverted or not.

[03:03]: vi input clock #8 invert

[02:02]: vi input clock #7 invert

[01:01]: vi input clock #6 invert

[00:00] vi input clock #5 invert

Global Control (CTR) Registers

This section describes CTR module registers.

Normal interrupt

Address: CTR base address + 0x00
Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dcm	jpeg	gpio	uart1	uart0	i2c	dsm	pci	aud	tme	ndma	sdma	vlc	vld	egf	hpm
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
dcmk	jpegk	cbus	mii	0	drm	dsm_k	0	aud_k	0	ndma_k	sdma_k	vlc_k	vld_k	egf_k	hpm_k

[31]: **dcm** – It specifies the normal interrupt from dcm module
[30]: **jpeg** – It specifies the normal interrupt from jpeg module
[29]: **gpio** – It specifies the normal interrupt from gpio module
[28]: **uart1** – It specifies the normal interrupt from uart1 module
[27]: **uart0** – It specifies the normal interrupt from uart0 module
[26]: **Reserved**
[25]: **dsm** - It specifies the normal interrupt from dsm module
[24]: **pci** - It specifies the normal interrupt from pci module
[23]: **aud** - It specifies the normal interrupt from aud module
[22]: **tme** - It specifies the normal interrupt from tme module
[21]: **ndma** - It specifies the normal interrupt from dma module
[20]: **sdma** -It specifies the normal interrupt from dma module
[19]: **vlc** - It specifies the normal interrupt from vlc module
[18]: **vld** - It specifies the normal interrupt from vld module
[17]: **egf** - It specifies the normal interrupt from egf module
[16]: **hpm** - It specifies the normal interrupt from hpm module
[15]: **dcmk** - It specifies the normal interrupt from dcm module
[14]: **jpegk** - It specifies the normal interrupt from jpeg module
[13]: **cbus** - It specifies the normal interrupt from cbus_arbiter module
[12]: **mii** - It specifies the normal interrupt from mii module
[11]: **Reserved**
[10]: **drm** - It specifies the normal interrupt from dsm module
[09]: **dsm_k** - It specifies the normal interrupt from dsm module
[08]: **Reserved**
[07]: **aud_k** - It specifies the normal interrupt from aud module
[06]: **Reserved**
[05]: **ndma_k** - It specifies the normal interrupt from dma module
[04]: **sdma_k** - It specifies the normal interrupt from dma module
[03]: **vlc_k** - It specifies the normal interrupt from vlc module
[02]: **vld_k** - It specifies the normal interrupt from vld module
[01]: **egf_k** - It specifies the normal interrupt from egf module
[00]: **hpm_k** - It specifies the normal interrupt from hpm module

Description
DCM shall follow the interrupt protocol described in interrupt scheme section.

Codec Video Channel

Address: CTR base address + 0x04
Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
int_mux[3:0]											cur_dch[4:0]				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											cur_vch[4:0]				

[31:28]: **int_mux[3:0]** - It specifies int_mux
[20:16]: **cur_dch** - It specifies current decode video channel
[15:5]: **Reserved**
[4:0]: **cur_vch** - It specifies current encode video channel

Description
DCM shall follow the interrupt protocol described in interrupt scheme section.

Software Reset

Address: CTR base address + 0x08
Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	dcm	mii	vids	vods	AUDD		uart1	uart0	i2c	IT	VLD	HPM	TME	MBC	VLC
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JPG	EGF	SPR	TSM	RTB	IPD		pbds	AUD	DSM	VOF	VIF	PCI	HIF	DDR	MPU

[31]: **Reserved**
[30]**R_DCM (dcm)**: reset the DCM module
[29]**R_MII (mii)**: reset the MII module
[28]**R_vids (vids)**: reset the VIDS module
[27]**R_vods (vods)**: reset the VODS module
[26]**AUDD**: audio decoding reset
[25]: **Reserved**
[24]**R_UART1 (uart1)**: UART1 reset
[23]**R_UART0 (uart0)**: UART0 reset
[22]: **Reserved**
[21]**R_IT (IT)**: Reset the IT block for decoding
[20]**R_VLD (VLD)**: Reset the VLD block for decoding
[19]**R_HPM (HPM)**: Reset the HPM block for decoding
[18]**R_TME (TME)**: Reset the TME block for encoding
[17]**R_MBC (MBC)**: Reset the MBC block for encoding
[16]**R_VLC (VLC)**: Reset the VLC block for encoding
[15]**R_JPG (JPG)**: Reset JPEG
[14]**R_EGF (EGF)**: Reset the EGF block for encoding or decoding
[13]**R_SPR (SPR)**: Reset the SPR block for encoding or decoding
[12]**R_TSM (TSM)**: Reset the TSM block for encoding or decoding
[11]**R_RTB (RTB)**: Reset the RTB block for encoding or decoding
[10]**R_IPD (IPD)**: Reset the IPD block for encoding or decoding
[09]: **Reserved**
[08]: **Reserved**
[07:07]**R_AUDE (AUD)**: Reset the audio encoding block
[06:06]**R_DSM (DSM)**: Reset the sub sample block

[05:05]**R_VOF (VOF)**: Reset the VOF block
[04:04]**R_VIF (VIF)**: Reset the VIF block
[03:03]**R_PCI (PCI)**: Reset the PCI block
[02:02]**R_HIF (HIF)**: Reset the HIF block
[01:01]**R_DDR (DDR)**: Reset the DDR controller
[00:00]**R_MPU (MPU)**: Reset the ARM and AMBA

Description

This register is used to control the TW2819 reset. Host or FW write LOW to assert TW2819 blocks reset. Host or FW write HIGH to release the reset.

Timer Period Register

Address: CTR base address + 0x0C

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
timer3_period								timer2_period							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
timer1_period								timer0_period							

[31:24]**timer3_period** - It specifies timer 3 period
[23:16]: **timer2_period** - It specifies timer 2 period
[15:8]: **timer1_period** - It specifies timer 1 period
[7:0]: **timer0_period** - It specifies timer 0 period

Description

The timer period specifies the timer unit in terms of system cycle. If timer_period equals 0, then timer counter increments every cycle. If timer_period equals 1, then timer counter increments every 2 cycles.

Encode Mode Register

Address: CTR base address + 0x10

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
															enc_mode_1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															enc_mode_0

[31:17]: **Reserved**
[16]: **enc_mode_1** - It specifies the operation mode for the video pipeline_1.
[0]: **enc_mode_0** - It specifies the operation mode for the video pipeline_0.

Fast Interrupt

Address: CTR base address + 0x14

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
timer0	timer1	timer2	timer3											vout	vin
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
t0en	t1en	t2en	t3en											voen	vien

[31]: **timer0** - It specifies the fast interrupt from timer0
[30]: **timer1** - It specifies the fast interrupt from timer1
[29]: **timer2** - It specifies the fast interrupt from timer2
[28]: **timer3** - It specifies the fast interrupt from timer 3
[27:18]: **Reserved** - It specifies the fast interrupt source
[17]: **vout** - It specifies the fast interrupt from V_{OUT} module
[16]: **vin** - It specifies the fast interrupt from V_{IN} module
[12:2]: **Reserved** - It specifies the fast interrupt source
[15]: **t0en** - It specifies the fast interrupt enable for timer0
[14]: **t1en** - It specifies the fast interrupt enable for timer1
[13]: **t2en** - It specifies the fast interrupt enable for timer2
[12]: **t3en** - It specifies the fast interrupt enable for timer3
[1]: **voen** - It specifies the fast interrupt enable for V_{OUT} module
[0]: **vien** - It specifies the fast interrupt enable for V_{IN} module

Description

DCM shall follow the interrupt protocol described in interrupt scheme section.

Software Reset K

Address: CTR base address + 0x18

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	dcm	mii	vids	vods	AUDD		uart1	uart0	i2c	IT	VLD	HPM	TME	MBC	VLC
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JPG	EGF	SPR	TSM	RTB	IPD		pbds	AUD	DSM	VOF	VIF	PCI	HIF	DDR	MPU

[31]: **Reserved**
[30] **R_DCM (dcm)**: Reset the DCM module
[29] **R_MII (mii)**: Reset the MII module
[28] **R_vids (vids)**: Reset the VIDS module
[27]**R_vods (vods)**: Reset the VODS module
[26]**AUDD**: Audio decoding reset
[25]: **Reserved**
[24]**R_UART1 (uart1):** UART1 reset
[23]**R_UART0 (uart 0)**: UART0 reset
[22]: **Reserved**
[21]**R_IT (IT):** Reset the IT block for decoding
[20]**R_VLD (VLD):** Reset the VLD block for decoding
[19]**R_HPM (HPM):** Reset the HPM block for decoding
[18]**R_TME (TME):** Reset the TME block for encoding
[17]**R_MBC (MBC):** Reset the MBC block for encoding
[16]**R_VLC (VLC):** Reset the VLC block for encoding
[15]**R_JPG (JPG):** Reset JPEG
[14]**R_EGF (EGF):** Reset the EGF block for encoding or decoding
[13]**R_SPR (SPR):** Reset the SPR block for encoding or decoding
[12]**R_TSM (TSM):** Reset the TSM block for encoding or decoding
[11]**R_RTB (RTB):** Reset the RTB block for encoding or decoding
[10]**R_IPD (IPD)**: Reset the IPD block for encoding or decoding
[09]: **Reserved**
[08]: **Reserved**
[07:07]**R_AUDE (AUD):** Reset the audio encoding block
[06:06]**R_DSM (DSM):** Reset the sub sample block
[05:05]**R_VOF (VOF)** Reset the VOF block
[04:04]**R_VIF (VIF)** Reset the VIF block
[03:03]**R_PCI (PCI):** Reset the PCI block
[02:02]**R_HIF (HIF):** Reset the HIF block
[01:01]**R_DDR (DDR):** Reset the DDR controller
[00:00]**R_MPU (MPU):** Reset the ARM and AMBA

Description

This register is used to control the TW2819 reset. Host or FW write LOW to assert TW2819 blocks reset. Host or FW write HIGH to release the reset.

Normal interrupt Enable

Address: CTR base address + 0x1c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dcme	jpege	gpioe	uart1e	uart0e	i2ce	dsme	pcie	aude	tmee	ndmae	sdmae	vlce	vlde	egfe	hpme
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
dcmke	jpegke	cbuse	miie	0	drme	dsmke	0	audke	0	ndmake	sdmake	vlcke	vldke	egfke	hpmke

[31]: **dcme** - It enables the normal interrupt from dcm module
[30]: **jpege** - It enables the normal interrupt from jepeg module
[29]: **gpioe** - It enables the normal interrupt from gpio module
[28]: **uart1e** - It enables the normal interrupt from uart1 module
[27]: **uart0e** - It enables the normal interrupt from uart0 module
[26]: **Reserved**
[25]: **dsme** - It enables the normal interrupt from dsm module
[24]: **pcie** - It enables the normal interrupt from pci module
[23]: **aude** - It enables the normal interrupt from aud module
[22]: **tmee** - It enables the normal interrupt from tme module
[21]: **ndmae** - It enables the normal interrupt from dma module
[20]: **sdmae** - It enables the normal interrupt from dma module
[19]: **vlce** - It enables the normal interrupt from vlc module
[18]: **vlde** - It enables the normal interrupt from vld module
[17]: **egfe** - It enables the normal interrupt from egf module
[16]: **hpme** - It enables the normal interrupt from hpm module
[15]: **dcme** - It enables the normal interrupt from dcm module
[14]: **gpioe** - It enables the normal interrupt from gpio module
[13]: **cbuse** - It enables the normal interrupt from cbus_arbiter module
[12]: **miie** - It enables the normal interrupt from mii module
[11]: **Reserved**
[10]: **drme** – It enables the normal interrupt from dsm module
[09]: **dsmke** - It enables the normal interrupt from dsm module
[08]: **Reserved**
[07]: **audke** - It enables the normal interrupt from aud module
[06]: **Reserved**
[05]: **ndmake** - It enables the normal interrupt from dma module
[04]: **sdmake** - It enables the normal interrupt from dma module
[03]: **vlcke** - It enables the normal interrupt from vlc module
[02]: **vldke** - It enables the normal interrupt from vld module
[01]: **egfke** - It enables the normal interrupt from egf module
[00]: **hpmke** - It enables the normal interrupt from hpm module

Description

DCM shall follow the interrupt protocol described in interrupt scheme section.

Encoder Parameter Register 0

Address: CTR base address + 0x20
Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
					vertical_size										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					horizontal_size										

[31:27]: **Reserved**
[26:16]: **vertical_size** - It specifies video vertical size in pixel
[15:11]: **Reserved**
[10:0]: **horizontal_size** - It specifies video horizontal size in pixel

Encoding Parameter Register 1

Address: CTR base address + 0x24
Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
rc_type		bframe_num		prog	Initial_QP										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
frame_rate								IDRPeriod							

[31:30]: **rc_type** - video rate control type: 00 – CQP, 01-CBR, 10- VBR, 11- HBR
[29:28]: **bframe_num** - It specifies how many b frames between two reference frame
[27]: **prog** - It specifies if current video sequence is progressive video
1 - progressive video
0 - interleaved video
[26:16]: **Initial_QP** - It specifies initial QP for the video encoding
[15:8]: **frame_rate** - It specifies encoding frame rate
[7:0]: **IDRPeriod** - It specifies IDR distance in terms of frame count

Encoder Parameter Register 2

Address: CTR base address + 0x28
Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
video bit rate															

[31:16]: **Reserved**
[15:0]: **video_bit_rate** - It specifies current video sequence encoding bit rate in terms of Kbps

Encoder Parameter Register 3

Address: CTR base address + 0x2c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
lp_flag	lp_disable							Loopfilter_AlphaOffset							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								Loopfilter_BeltaOffset							

[31] **lp_flag**

[30]: **lp_disable** – It specifies loop filter disable

[29:24]: **Reserved**

[23:16]: **Loopfilter_AlphaOffset**

[15:8]: **Reserved**

[7:0]: **Loopfilter_BeltaOffset**

Timer 0 Count

Address:CTR base address + 0x30

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
timer_count															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

[31:0]: **timer_count** - It specifies timer value for Timer 0

Timer 1 Count

Address: CTR base address + 0x34

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
timer_count															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

[31:0]: **timer_count** – It specifies timer value for Timer 1

Timer 2 Count

Address: CTR base address + 0x38

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
timer_count															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

[31:0]: **timer_count** – It specifies timer value for Timer 2

Timer 3 Count

Address: CTR base address + 0x3c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
timer_count															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

[31:0]: **timer_count** - It specifies timer value for Timer 3

Encoder Parameter Register 4

Address: CTR base address + 0x40

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
lp_flag															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

[31:0]: **lp_flag**

Encoder Parameter Register 5

Address: CTR base address + 0x44

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
lp_flag															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

[31:0]: **lp_flag**

Encoder Parameter Register 6

Address: CTR base address + 0x48

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
lp_flag															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

[31:0]: **lp_flag**

Encoder Parameter Register 7

Address: CTR base address + 0x4c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
lp_flag															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

[31:0]:**lp_flag**

Encoder Parameter Register 8

Address: CTR base address + 0x50

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
lp_flag															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

[31:0]: **lp_flag**

Encoder Parameter Register 9

Address: CTR base address + 0x54

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
lp_flag															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

[31:0]: **lp_flag**

Encoder Parameter Register 10

Address: CTR base address + 0x58

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
lp_flag															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

[31:0]: **lp_flag**

Encoder Parameter Register 11

Address: CTR base address + 0x5c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
lp_flag															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

[31:0]: **lp_flag**

Encoder Parameter Register 12

Address: CTR base address + 0x60

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
lp_flag															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

[31:0]: **lp_flag**

DDR-Limit Control Register

Address: CTR base address + 0x70

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ddr_limit[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ddr_limit[15:0]															

[31:0]: **ddr_limit[31:0]** - it specifies ddr_limit[31:0]. Default value is 32'h00A0_0000.

UC-Limit Control Register

Address: CTR base address + 0x74

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UC_limit[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UC_limit[15:0]															

[31:0]: **UC_limit[31:0]** – it specifies UC_limit[31:0]. Default value is 32'h0000_FFF0.

Vi-sync Check And Vi-fake-DCM Control Register

Address: CTR base address + 0x90

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Vien8	Vien7	Vien6	Vien5	Vidis8	Vidis7	Vidis6	Vidis5	Vien4	Vien3	Vien2	Vien1	Vidis4	Vidis3	Vidis2	Vidis1

[31:16]: **Reserved**

[15]: **vi_chk_en8 (Vien8)** - It specifies vi-ch-8 sync-check is enabled.

[14]: **vi_chk_en7 (Vien7)** - It specifies vi-ch-7 sync-check is enabled.

[13]: **vi_chk_en6 (Vien6)** - It specifies vi-ch-6 sync-check is enabled.

[12]: **vi_chk_en5 (Vien5)** - It specifies vi-ch-5 sync-check is enabled.

[11]: **vi_dis8 (Vidis8)** - It specifies vi-ch-8 real_req is disabled, and is served by a fake dcm_wvld. Applied to the 2nd logic_core.

[10]: **vi_dis7 (Vidis7)** - It specifies vi-ch-7 real_req is disabled, and is served by a fake dcm_wvld. Applied to the 2nd logic_core.

[09]: **vi_dis6 (Vidis6)** - It specifies vi-ch-6 real_req is disabled, and is served by a fake dcm_wvld. Applied to the 2nd logic_core

[08]: **vi_dis5 (Vidis5)** - It specifies vi-ch-5 real_req is disabled, and is served by a fake dcm_wvld. Applied to the 2nd logic_core.

[07]: **vi_chk_en4 (Vien4)** - It specifies vi-ch-4 sync-check is enabled.

[06]: **vi_chk_en3 (Vien3)** - It specifies vi-ch-3 sync-check is enabled

[05]: **vi_chk_en2 (Vien2)** - It specifies vi-ch-2 sync-check is enabled

[04]: **vi_chk_en1 (Vien1)** - It specifies vi-ch-1 sync-check is enabled

[03]: **vi_dis4 (Vidis4)** - It specifies vi-ch-4 real_req is disabled, and is served by a fake dcm_wvld. Applied to the 1st logic core.

[02]: **vi_dis3 (Vidis3)** - It specifies vi-ch-3 real_req is disabled, and is served by a fake dcm_wvld. Applied to the 1st logic core

[01]: **vi_dis2 (Vidis2)** - It specifies vi-ch-2 real_req is disabled, and is served by a fake dcm_wvld. Applied to the 1st logic core.

[00]: **vi_dis1 (Vidis1)** - It specifies vi-ch-1 real_req is disabled, and is served by a fake dcm_wvld. Applied to the 1st logic core.

VO Fake-DCM Control Register – Not Used

Address: CTR base address + 0x94

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												Vodis4	Vodis3	Vodis2	Vodis1

[31:4]: **Reserved**

[03]: **vo_dis4 (Vodis4)** - it specifies vo-ch-4 real request is disabled, and is served by a fake dcm_wvld. Not used in TW2819.
[02]: **vo_dis3 (Vodis3) -** it specifies vo-ch-3 real request is disabled, and is served by a fake dcm_wvld. Not used in TW2819.
[01]: **vo_dis2 (Vodis2) -** it specifies vo-ch-2 real request is disabled, and is served by a fake dcm_wvld. Not used in TW2819.
[00]: **vo_dis1 (Vodis1) -** it specifies vo-ch-1 real request is disabled, and is served by a fake dcm_wvld. Not used in TW2819.

VI-sync Check Control Register

Address: CTR base address + 0x98

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Vien16	Vien15	Vien14	Vien13	Vidis12	Vidis11	Vidis10	Vidis9	Virst8	Virst7	Virst6	Virst5	Vrsts4	Virst3	Virst2	Virst1

[31:16]: **Reserved**
[15]: **vi_chk_en16 (Vien16)** - It specifies vi-ch-16 sync-check is enabled.
[14]: **vi_chk_en15 (Vien15)** - It specifies vi-ch-15 sync-check is enabled.
[13]: **vi_chk_en14 (Vien14)** - It specifies vi-ch-14 sync-check is enabled.
[12]: **vi_chk_en13 (Vien13)** - It specifies vi-ch-13 sync-check is enabled.
[11]: **vi_chk_en12 (Vien12)** - It specifies vi-ch-12 sync-check is enabled.
[10]: **vi_chk_en11 (Vien11)** - It specifies vi-ch-11 sync-check is enabled.
[09]: **vi_chk_en10 (Vien10)** - It specifies vi-ch-10 sync-check is enabled.
[08]: **vi_chk_en9 (Vien9)** - It specifies vi-ch-9 sync-check is enabled.
[07]: **vi_rst8 (Virst8)** - It specifies vi-port-8 reset, not used.
[06]: **vi_rst7 (Virst7)** - It specifies vi-port-7 reset, not used.
[05]: **vi_rst6 (Virst6)** - It specifies vi-port-6 reset, not used.
[04]: **vi_rst5 (Virst5)** - It specifies vi-port-5 reset, not used.
[03]: **vi_rst4 (Virst4)** - It specifies vi-port-4 reset, not used.
[02]: **vi_rst3 (Virst3)** - It specifies vi-port-3 reset, not used.
[01]: **vi_rst2 (Virst2)** - It specifies vi-port-2 reset, applied to the 2nd instance of vif_fiv_top.
[00]: **vi_rst1 (Virst1)** - It specifies vi-port-1 reset, applied to the 1st instance of vif_fiv_top.

VI-sync Check Control Register

Address: CTR base address + 0x9c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														vorst2	vorst1

[31:2]: **Reserved**

[01]: **vo_rst_n2 (vorst2)** - It specifies vo-port-2 reset, applied to the 2nd instance of vif_otop.

[00]: **vo_rst_n1 (vorst1)** - It specifies vo-port-1 reset, applied to the 1st instance of vif_otop.

VIF Registers

This section describes video interface module registers. Registers described in this section are controlled by firmware running on embedded processor and only provided for reference. It is not recommended to access these registers from PCI host.

VIF-VI Interrupt 0 (r0)

Address: VIF_Baseaddress + 0x00

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vi_clear	vi_ch_no[3:0]													vi_int	field _err
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vi_set_port[3:0]						vi_srv_no								vi_int_en	

[31]: **vi_clear**

[30:27]: **vi_ch_no** - Channel number 0-15.

[17]: **vi_int** - It specifies VIF video input interrupt.

[16]: **field_err** - It specifies top_fld_cnt != bottom_fld_cnt error.

[15:12]: **vi_set_port** – It specifies which vi_port (0 to 15) takes the set parameters.

[11:10]: **Reserved**

[9:6]: **vi_srv_no** – It specifies VIF video input interrupt source:

0: Module ready after system reset

1: End of top field

2: End of bottom field

3: vbi extraction is done at end of the vbi-line

4: vi input luma FIFO overflows, error condition

5: vi input chroma FIFO overflows, error condition

6: vi input top-field-no_video, error condition

7: vi input bottom-field-no_video, error condition

[1]: **vi_int_en** - It specifies video input module interrupt enable.

Description

VIF shall follow the interrupt protocol described in interrupt scheme section.

VIF-VI Interrupt 1 (r1)

Address: VIF_Baseaddress + 0x04

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hhr_ftr		set_vs_neg	dcm_glue_on									horizontal_offset			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	bottom_fld_ver_offset					top_fld_ver_offset						tp_vi	fld2rm_fw	frm2fld_fw	fld2fld_fw

[31]: **hhr_ftr1** - It specifies down-sample to horizontal half resolution for vi port-1

1: Down-sample scalar enabled

0: Down-sample scalar is not enabled

[29]: **set_vs_neg** - 1 = VSYNC at falling edge of vs, 0 = VSYNC at rising edge of fs

[28]: **dcm_glue_on** - 1 = Use vofifo to prolong vi_dcm_req latency tolerance

[19:15]: **horizontal_offset** - It specifies horizontal offset for video input.

[14:10]: **bottom_fld_ver_offset** - It specifies the bottom field vertical line offset for video input.

[9:5]: **top_fid_ver_offset** – It specifies the top field vertical line offset for video input.

[3]: **tp_vi** – It specifies simple bar video input test pattern is enabled.

[2]: **fld2frm_fw** - It specifies dram line-order is forced to frame-order.

[1]: **frm2fld_fw** - It specifies dram line-order is forced to field-order.

[0]: **fld2fld_fw** - It specifies dram line-order is forced to field-order.

VIF-VI Control Register 2 (r2)

Address: VIF_Baseaddress + 0x08

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vi_size_en	vi_cnt_en		pre_index			yc_os		pause	cdrop	ydrop	cif1		tff	sev	tqv
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ttv	ser	tqr	ttr	qcif	c_sht	s_sft	vbi_en	byte8	h_pel_vi						

[31]: **vi_sz_en -** It specifies arbitrary picture-width capture in vi

[30]: **vi_cnt_en -** It specifies arbitrary picture-width capture in vi

[28:26]: **pre_index** - It specifies prefilter selection

[25:24]: **yc_os** - It specifies u-y-v-y video input timing setting

[23]: **pause_PTS (pause)** - It specifies the video pause on 90kHz pts control.

[22]: **chroma_drop (cdrop)** - It specifies chroma horizontal filter option.

[21]: **luma_drop (ydrop)** - It specifies luma horizontal filter option.

[20]: **cif1** - It specifies video input resolution down-sample to cif for the set vi port.

[19]: **Reserved**

[18]: **tff** - It specifies video input top field first.

[17]: **sev** - It specifies video input resolution vertical down-sample to 7/8 for the set port.

[16]: **tqv** - It specifies video input resolution vertical down-sample to ¾ for the set port.

[15]: **ttv** - It specifies video input resolution vertical down-sample to 2/3 for the set port.

[14]: **ser** - It specifies video input resolution horizontal down-sample to 7/8 for the set port.

[13]: **tqr** - It specifies video input resolution horizontal down-sample to ¾ for the set port.

[12]: **ttr** - It specifies video input resolution horizontal down-sample to 2/3 for the set port.

[6:0]**h_pel_vi** - It specifies video input x16 for active pixel per line.

e.g., d1 (44 or 45), 4d1/hd_8d1 (88 or 90), hd (120)

VIF-VI Control Register 3 (r3)

Address: VIF_Baseaddress + 0x0c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
tw2880	tw2851	tw2835	tw2864		tw2868	sw_hd	patch	P81120	P71120	P61120	P51120	P41120	P31120	P21120	P11120
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
clk_div8		clk_div7		clk_div6		clk_div5		clk_div4		clk_div3		clk_div2		clk_div1	

[31]: **tw2880** - It specifies input is from TW2880.
[30]: **tw2851** - It specifies input is from TW2851.
[29]: **tw2835** - It specifies input is from TW2835.
[28]: **tw2815/64** - It specifies input is from TW2815 or TW2864, 4-bit chid extraction.
[26]: **tw2868** - It specifies TW2868's 8-bit ch_id extraction mode in vi.
[25]: **swap_hd_byte (sw_hd)** - It specifies video input HD format data is swapped.
[24]: **patch_last (patch)** - It specifies video input line will be patched to the 8Byte boundary.
[23]: **p8_bt1120 (P81120)** - It specifies video input HD format for vi-port-15.
[22]: **p7_bt1120 (P71120)** - It specifies video input HD format for vi-port-13.
[21]: **p6_bt1120 (P61120)** - It specifies video input HD format for vi-port-11.
[21]: **p5_bt1120 (P51120)** - It specifies video input HD format for vi-port-9.
[21]: **p4_bt1120 (P41120)** - It specifies video input HD format for vi-port-7.
[21]: **p3_bt1120 (P31120)** - It specifies video input HD format for vi-port-5.
[21]: **p2_bt1120 (P21120)** - It specifies video input HD format for vi-port-3.
[21]: **p1_bt1120 (P11120)** - It specifies video input HD format for vi-port-1.
[15:14]: **clk_div8** - It specifies port-8 clock divider.
[13:12]: **clk_div7** - It specifies port-7 clock divider.
[11:10]: **clk_div6** - It specifies port-6 clock divider.
[9:8]: **clk_div5** - It specifies port-5 clock divider.
[7:6]: **clk_div4** - It specifies port-4 clock divider.
[5:4]: **clk_div3** - It specifies port-3 clock divider.
[3:2]: **clk_div2** - It specifies port-2 clock divider.
[1:0]: **clk_div1** - It specifies port-1 clock divider.
Clk_div = "10" means 108-to-27 conversion, which can be applied to port-1/2/5/6.
Clk_div = "01" means 54-to-27 conversion, which can be applied to port-1/2/3/4/5/6/7/8.
Clk_div = "10" means 27-to-27 conversion, which can be applied to port-1/2/3/4/5/6/7/8.

VIF-VI Control Register 4 (r4)

Address: VIF_Baseaddress + 0x10

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	disable_vi_dcm	vbi_vi_line_start1								vbi_vi_line_end1					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vbi_vi_pixel_start1								vbi_vi_pixel_end1							

[31]: **Reserved**
[30]: **disable_vi_dcm**–It specifies vi_dcm_req is disabled, or no data-capture to dram.
[29:24]: **vbi_vi_line_start1**–It specifies generic vbi-vi line start number for vi port-1.
[21:16]: **vbi_vi_line_end1**–It specifies generic vbi-vi line end number for vi port-1.
[15:8]: **vbi_vi_pixel_start1**–It specifies generic vbi-vi pixel start number for vi port-1.
[7:0]: **vbi_vi_pixel_end1**–It specifies generic vbi-vi pixel end number for vi port-1.

VIF-VO Control Register 5 (r5)

Address: VIF_Baseaddress + 0x14

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
				P720_30	d1_delay		deblk_filter_sel			h_den					h_num
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
black	p2n	v_den						tp_vo	d2a	prog_c	d1w	h_dec	h_int	v_int	v_dec

[31:28]: **Reserved**

[27]:**P720_30** - It specifies 30Hz HD-720P output mode:

1: 30Hz HD-720P vo output

0: 60Hz HD-720P vo output

[26:25]: **d1_delay** - It specifies CCIR656 output pel/color tuning.

[24:22]: **deblk_filter_sel** - It specifies simple horizontal deblock filter selection, not used.

[21:17]: **h_den** - It specifies hor. Interpolation denominator.

[16]: **h_num** - It specifies hor. Interpolation numerator.

[15]: **black** - It specifies black out video output.

[14]: **p2n** - It specifies convert video output from PAL to NTSC.

[13:11]: **v_den** - It specifies vertical decimation line-drop ratio parameter.

[7]: **test_pattern_vo (tp_vo)** - It specifies vo's bar test pattern generation, used to test sdram.

[6]: **d2a** - It specifies special y-line increment to adopt to special DAC convention.

[5]: **prog_c** - It specifies chroma output from consecutive chroma line in the DDR.

[4]: **d1_width_vo (d1w)** - It specifies 0 = 704, 1 = 720 for 4d1 composition.

[3]: **hor_decim (h_dec)** - It specifies video output uses horizontal decimation, not used.

[2]: **hor_intrp (h_int)** - It specifies video output uses horizontal interpolation, not used.

[1]: **ver_intrp (v_int)** - It specifies video output uses vertical interpolation, not used.

[0]: **ver_decim (v_dec)** - It specifies video output uses veritical decimation, not used.

VIF Control Register 6 (r6)

Address: VIF_Baseaddress + 0x18

Type: r/w

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
														vb2[10:9]	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vb2[8:0]									vb1						

[31:18]: **Reserved**

[17:7]: **vb2[10:0]** - It specifies ITU656/BT1120 vb2, the bot-fld-v-blanking-start line number

[6:0]: **vb1[6:0]** - It specifies ITU656/BT1120 vb1, the top-fld-v-blanking-end line number

VIF Control Register 7 (r7)

Address: VIF_Baseaddress + 0x1c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
flip_vo_fld	v_os_f2					v_os_f1					fld_vo				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					odd2[10:0]										

[31]: **flip_vo_fld** - It specifies vo's sav/eav-bit-6 F-bit is reversed to display bottom-field content on top-field.
[30:26]: **v_os_f2** - It specifies video output vertical offset in the bottom field
[25:21]: **v_os_f1** - It specifies video output vertical offset in the top field
[20]: **fld_vo** - It specifies field ordered DDR frame buffer mapping
[19:11]: **Reserved**
[10:0]: **odd2[10:0]** - It specifies odd2, the bottom-fld-start line number in bt656/bt1120.

VIF-VO Control Register 8 (r8)

Address: VIF_Baseaddress + 0x20

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cif_vo	tff_vo	pal_vo	fld_pof	deblk_os			deblk_ftr	h_pixel_vo							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
v_in_fm_vo[11:0]											f_offset_vo				

[31]: **cif_vo** - It specifies video output convert cif to d1.
[30]: **tff_vo** - It specifies video output top-field-first flag.
[29]: **pal_vo** - It specifies 1 = using the pal format, 0 = ntsc format.
[28]: **fld_pof** - It specifies 1 = inter-field-post-filter option.
[27:25]: **deblk_os** - It specifies deblock offset.
[24]: **deblk_ftr** - It specifies turning on a simple vo deblocking filter.
[23:17]: **h_pixel_vo** - It specifies x16 for active pixel number in vo
[16:5]: **v_in_fm_vo[11:0]** - It specifies active line number for video output, It is 480 for ntsc, 576 for pal, and 2160 for 1080P-HD.
[4:0]: **h_offset_vo** - It specifies horizontal offset for video output.

VIF Control Register 9 (r9)

Address: VIF_Baseaddress + 0x24

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
t2880	t2851	t2835	t2815			p1_lp_vo	p0_lp_vo	p0_sd	p0_hd	p1_sd	p1_hd	p1_vo_format			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	prog2	prog1	p0_vo_format						diseco	bysw1	bysw0	yc_os		pic_struct	

[31] **TW2880_vo (t2880):** channel-id use 2880 format.
[30] **TW2851_vo (t2851):** channel-id use 2851 format.
[29] **TW2835_vo (t2835):** channel-id use 2835 format.
[28] **TW2815_vo (tw2815):** channel-id use 2815 format.
[25] **p1_loopbk_vo (p1_lp_vo):** otop-1 video loop-back for spot.
[24] **p0_loopbk_vo (p0_lp_vo)**: otop-0 video loop-back for spot.
[23] **p0_sd_port (p0_sd):** it controls port-0/1 vofifo_full flag.
[22] **p0_hd_rd (p0_hd):** it controls port-0/1 vofifo increment in hd-format.
[21] **p1_sd_port (p1_sd):** it controls port-2/3 vofifo_full flag.
[20] **p1_hd_rd (p1_hd) :** it controls port-2/3 vofifo increment in hd-format.
[19:15] **p1_vo_format:** specification is the same as p0_vo_format.
[14] **progressive2 (prog2):** vo port-2 is in progressive display output mode.
[13] **progressive1 (prog1):** vo port-1 is in progressive display output mode.
[12:8] **p0_vo_format:** 0 = d1-frame-to-d1 1=cif-field-to-d1 2 = cif-frame-to-d1.
3 = hhr-field-to-d1 4 = hhr-frame-to-d1 7=2-d1-field-to-d1 9 = 4-cif-frame-to-d1.
10 = 8-cif-field-to-d1 11 = 6cif-left-hhr-field-to-d1 12 = 6cif-right-hhr-to-d1
13 = 4hhr-field-to-d1 14 = 4d1-frame-to-4d1 15 = 6vga-frame-to-hd.
16 = hd-frame-to-hd 17 = 4d1-frame-to-hd 18 = 2-hhr-frame-to-d1
30 = 160x128-format for ate pattern 31 = vi-to-vo-loop-back
[7] **Reserved**
[6] **disable_vo_eco (diseco):** 1 = disable an eco option in vif_odcm.
0 = normal operation
[5] **bysw1:** vo hd-port-1 switches Y-C bytes (1) or not (0).
[4] **bysw0:** vo hd-port-0 switches Y-C bytes (1) or not (0).
[3:2] **yc_os:** It specifies ITU656 output pel-order line fine tune.
[1:0] **pic_struct**: It specifies vo picture structure (not used).

Encoding PTS Register r10 (Read Only)

Address: VIF_Baseaddress + 0x28

Type: Read-only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						PTS_extension									PTS_32

[31:10]: **Reserved**
[9:1]: **PTS_extension** – It specifies PTS-remainder value when 27MHz divided by 300, its range is 0 to 299.
[0]: **PTS_32** – It specifies MSB bit 32 of PTS value in 90kHz.
These values are assigned to vi port0 to port15 according to vi_ch_no[3:0] specified by hardware in r000[30:27].

Encoding PTS Register r11 (Read-only)

Address: VIF_Baseaddress + 0x2c

Type: Read-only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PTS															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

[31:0]: **PTS** - It specifies lsb 32 bits of PTS value in 90kHz, the value is assigned to port0 to port15 following the vi_ch_no[3:0] value specified by hardware in r000[30:27].

VIF VO Register r12

Address: VIF_Baseaddress + 0x30

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		setvs	n8v_vo	Sev_vo	tqv _vo	ttv _vo	qcif_d1			vbi_vo_location_bottom_feild					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					vbi_vo_location_top_field										

[29]: **Reserved**

[28]: **n8v_vo** - It specifies vo output interpolation from 8/15 to full-height format (288-to-540 height).

[27]: **sev_vo** - It specifies vo output interpolation from 7/8 to full-height format.

[26]: **tqv_vo** - It specifies vo output interpolation from ¾ to full-height format.

[25]: **ttv_vo** - It specifies vo output interpolation from 2/3-height to full-height format.

[24]: **qcif_d1** - It specifies vo output interpolation from qcif to d1 format.

[21:11]: **vbi_vo_location_bottom_field** - It specifies 11-bit vbi-location in bottom field.

[10:0]: **vbi_vo_location_top_field** - It specifies 11-bit vbi-location in top field.

VIF VI Control Register r13

Address: VIF_Baseaddress + 0x34

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Thd8	Thd7	Thd6	Thd5	Thd4	Thd3	Thd2	Thd1	Qhd8	Qhd7	Qhd6	Qhd5	Qhd4	Qhd3	Qhd2	Qhd1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vi_sync_limit															

[31:24]: **Thd8 to Thd1** - It specifies to use double vi-fifo for hd-input (1) or not (0)

[23:16]: **Qhd8 to Qhd1** - It specifies using the even-number ports to do 16-bit-word-interleaved hd-input (1) or not (0). E.g., if qhd1 = 1, port-2 handles hd-ch-2; if qhd2 = 1, port-4 handles hd-ch-4, etc.

[15:0]: **vi_sync_limit** - It specifies cycle counts after sav or eav before an error interrupt is sent to firmware. This is used to detect sync-loss in video input. The default value is set at 3960 video input clock cycles. Increasing this number decreases the sensitivity of error detection.

VIF VI control Register r14

Address: VIF_Baseaddress + 0x38

Type: R/W

[31:0]: **Reserved**

VIF VI Control Register r15 (Read Only for HW Debugging)

Address: VIF_Baseaddress + 0x3c

Type: Read-only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					vi_total_line[10:0]										

[31:11]: **Reserved**

[10:0]: **vi_total_line** - It specifies the total number of lines captured per field at interrupt time for the channel specified by r0[30:27].

VIF VI Control Register r16 (Read Only for HW Debugging)

Address: VIF_Baseaddress + 0x40

Type: Read-only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
enc_chid3[7:0]								enc_chid2[7:0]							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
enc_chid1[7:0]								enc_chid0[7:0]							

[31:24]: **enc_chid3[7:0]** - It specifies the region-3 channel_id extracted from tw2851/2880.

[23:16]: **enc_chid2[7:0]** - It specifies the region-2 channel_id extracted from tw2851/2880.

[15:8]: **enc_chid1[7:0]** - It specifies the region-1 channel_id extracted from tw2851/2880.

[7:0]: **enc_chid0[7:0]** - It specifies the region-0 channel_id extracted from tw2851/2880.

When r0[30] = vi_ch_no[3] = 0, this register contains the chid information extracted from vi-port-0.

When r0[30] = vi_ch_no[3] = 1, this register contains the chid information extracted from vi-port-1.

Each 8-bit field is defined as {field, size[1:0], ch_no[4:0]}, where field is defined as

Top-field: 1,

Bottom-field: 0; and

size[1:0] is defined as:

D1: 2'b00

HHR: 2'b01

CIF: 2'b10

Others: 2'b11.

VIF-VO Interrupt (r17)

Address: VIF_Baseaddress + 0x44

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo4_st	vo3_st	vo2_st	vo1_st					sel_vo_org	vo_set_port		vo_int_clr				vo_int
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
pal_err	int_err							vo_ch_no		vo_srv_no					Vo_int_en

[31]: **vo4_st** - It specifies the initial-start of vo-port-4.
[30]: **vo3_st** - It specifies the initial-start of vo-port-3.
[29]: **vo2_st** - It specifies the initial-start of vo-port-2.
[28]: **vo1_st** - It specifies the initial-start of vo-port-1.
[27:24]: **Reserved**
[23]: **sel_vo_org** - It specifies sel_vo_org, not used.
[22:21]: **vo_set_port** - It specifies the vo-port (0-to-3) to set vo parameters.
[20]: **vo_int_clr** - It specifies VIF vo interrupt is cleared.
[19:17]: **Reserved**
[16]: **vo_int** - It specifies VIF video output interrupt
[13:8]: **Reserved**
[7:6]: **vo_ch_no** - It specifies the vo-port number at vo_cpu_int time (0-to-3).
[5:2]: **vo_srv_no** - It specifies VIF video output interrupt source:
0: module ready after system reset
1: end of the top field
2: end of the bottom field
3: DCM request wait cycles exceed the FW set limit in r45[15:0].
[0]: **vo_int_en** - It specifies video output module interrupt enable.

VIF-VO Control Register 18 (r18)

Address: VIF_Baseaddress + 0x48

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	chidcp1	chidcp0	p0_vo_st	p1_vo_st						vo_en1	vo_en2	vo_en3	vo_en4		

[31:15]: **Reserved**
[14]: **vo_chid_copy_p1 (chidcp1)** - It specifies copying channel-id directly from vi for port-1
[13]: **vo_chid_copy_p0 (chidcp0)** - It specifies copying channel-id directly from vi for port-0
[12]: **p0_vo_start (p0_vo_st)** - It specifies vo start for port-0
[11]: **p1_vo_start (p1_vo_st)** - It specifies vo start for port-1
[5]: **vo_en1** - It specifies video output enable for ch-1
[4]: **vo_en2** - It specifies video output enable for ch-2
[3]: **vo_en3** - It specifies video output enable for ch-3
[2]: **vo_en4** - It specifies video output enable for ch-4

VIF-VI-VO Control Register 19 (r19)

Address: VIF_Baseaddress + 0x4c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vi_upper_limit								vi_lower_limit							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo_upper_limit								vo_lower_limit							

[31:24]: **vi_upper_limit** - Input clipping upper-limit [1-254] for all vi ports

[23:16]: **vi_lower_limit** - Input clipping lower-limit for all vi ports

[15:8]: **vo_upper_limit** - Output clipping upper-limit [1-254] for both vo ports

[7:0]: **vo_lower_limit** - Output clipping lower-limit for both vo ports

VIF-VI Control Register 20 (r20)

Address: VIF_Baseaddress + 0x50

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vi_en[15:0]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			fldonly		ctr0_top	Set_top	vi_start	initbot	fwset	fwfrm				fwmas	sepin

[31:16]: **vi_en[15:0]** - these 16 bits specify which vi port (0 to 15) is enabled. [16] is port0, [17] is port1, and [31] is port15, etc.

[12]: **vi_field_only (fildonly)** - vi only captures in field line-order for the specified channel in vi_ch_no[3:0].

[11]: **Reserved**

[10]: **ctr0_top** - this bit is used along with set_top bit below, to fix the tw2960 F-bit error when the video source is noisy.
When set_top = 1, a 1-bit field-counter is started. If ctr0_top = 1, vi set the “0” value counter as top-field (equivalent to F = 0). If ctr0_top = 0, vi set the “0” value counter as bottom-field (equivalent to F = 1). The counter will not stop after it is initiated, so it takes at most 2 guesses to guess the correct top-field when TW2960 error happens.

[9]: **set_top** - This bit is set to “1” when TW2960 F-bit error happens. Default is “0”.

[8]: **vi_start** - vi start

[7]: **init_bot_field_start (initbot)** - Special start signal when vi needs start at bottom-field for the specified port number in vi_ch_no[3:0] in r0.

[6]: **fw_extra_set (fwset)** - It specifies field or frame line-order to over-write input chid setting.
When this bit is set, firmware-specified line-order is used in sdram. (option-2)

[5]: **fw_master_d1_frame (fwfrm)** - when this value is “1”t specifies frame-line-order in sdram for the port set by vi_set_port[3:0] in r0; when this value is “0”, it specifies the field-line-order in sdram for the set port.

[4:2]: **Reserved**

[1]: **fw_master (fwmas)** - It specifies field or frame line-order to over-write input chid setting
When this bit is set, firmware-specified line-order is used in sdram. (option-1)

[0]: **separate_input (sepin)** - It specifies each channel is separately allocated in sdram (not used by firmware now).

VIF-VI Control Register 21 (r21)

Address: VIF_Baseaddress + 0x54

Type: Read-only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vif_debug_port[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vif_debug_port[15:0]															

[31:0]: **vif_debug_port** - This is the vif_debug_port whose information is specified by vif_dbg_ad[7:0].

When vif_dbg_ad = 1, this contains the d1_state[3:0] information in sync extractor.

In normal operation, d1_state changes fast and firmware cannot catch it meaningfully. This is useful only when a port gets stuck, and firmware can specify the port by vi_set_port[3:0] to read back the port's d1_state stuck value.

When vif_dbg_ad = 2, this contains the p0_vo_status.

When vif_dbg_ad = 3, this contains the p1_vo_status.

VIF-VI Control Register 22 (r22)

Address: VIF_Baseaddress + 0x58

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vif_fifo_status		P1_vo_status[29:16]													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P1_vo_status[15:0]															

[31:30]: **vif_fifo_status** - {luf_full, chf_full} for the port specified in vi_set_port[3:0].

[29:0]: **P1_vo_status** - vo-port-1 status for debugging

VIF-VI Information Register 23 (r23)

Address: VIF_Baseaddress + 0x5c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Vo0_fail	vo1_fail	P0_vo_status[29:16]													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P0_vo_status[15:0]															

[31]: **vo0_fail** - vo-port-0 failed

[30]: **vo1_fail** - vo-port-1 failed

[29:0]: **p0_vo_status** - vo-port-0 status for debugging

VIF-VI Information Register 24 (r24)

Address: VIF_Baseaddress + 0x60

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vi_sync_det_en[15:0]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vi_sync_lost_en[15:0]															

[31:16]: **vi_sync_det_en[15:0]** - Bit 0 enables port-0, Bit 1 enables port-1, etc.

[15:0]: **vi1_sync_lost_en[15:0]** - Bit 0 enables port-0, Bit 1 enables port-1, etc.

VIF-VI Information Register 25 (Read-only)

Address: VIF_Baseaddress + 0x64

Type: Read-only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo_frame_count[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo_frame_count[15:0]															

[31:0]: **vo_frame_count** - Output video's frame-count

VIF-VI Information Register 26 (Read-only)

Address: VIF_Baseaddress + 0x68

Type: Read-only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo_first_yout[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo_first_youtt[15:0]															

[31:0]: **vo_first_yout** - In vo debug mode, the first 4 bytes of active video data contains the vo_frame_count in the previous register, and is reported back in r26. If this number doesn't match the firmware read value of r25, then hardware reports vo failure.

VIF-VI Information Register 27 (Read-only)

Address: VIF_Baseaddress + 0x68

Type: Read-only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											od1_err	od2_err	Cnt_err	pal_err	int_err

[31:5]: **Reserved**

[4]: **od1_err – odd1_err**

[3]: **od2_err – odd2_err**

[2]: **cnt_err – field_cnt_err_vo**

[1]: **pal_err – pal_err**

[0]: **int_err – vo_cpu_int_err**

VIF-VI Information Register 28 (r28)

Address: VIF_Baseaddress + 0x70

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vi_sync_det[15:0]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vi_sync_lost[15:0]															

[31:16]: **vi_sync_det[15:0]** - Bit0 = 1 means port-0 has sync, Bit 1 = 1 means port-1 has sync, etc. This specifies normal video input condition.

[15:0]: **vi_sync_lost[15:0]** - Bit 0 = 1 means port-0 loses sync, Bit 1 = 1 means port-1 loses sync, etc This specifies bad video input condition.

VIF-VI Information Register 29 (r29)

Address: VIF_Baseaddress + 0x74

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
						resize_width_in2[9:0]									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						resize_width_in1[9:0]									

[31:26]: **Reserved**

[25:16]: **Resize_width_in2 [9:0]** – This controls vo's active-line period for vo-port-1, the value needs to be set when horizontal-interpolation resize is used. The width is set at required-width/4+6. e.g., in hd-1080, the value is 486, in hd-720p, the value is 326.

[15:10]: **Reserved**

[9:0]: **Resize_width_in1[9:0]** – This controls vo's active-line period for vo-port-0, the value needs to be set when horizontal-interpolation resize is used. The width is set at required-width/4+6. e.g., in hd-1080, the value is 486, in hd-720p, the value is 326.

VIF-VI Information Register 30 (r30)

Address: VIF_Baseaddress + 0x78

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
width1_set				width1_vo[11:0]											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
width2_set				width2_vo[11:0]											

[31]: **width1_set** - 1 = set vo-port-0's line-width in ccir656 or bt1120 for various formats.

[27:16]: **width1_vo [11:0]** – This field sets the line-width for various formats for vo-port-0.

[15:10]: **width2_set** - 1 = set vo-port-1's line-width in ccir656 or bt1120 for various formats

[9:0]: **width2_vo[11:0]** – This field sets the line-width for various formats for vo-port-1.

VIF-VO Control Register 31 (r31)

Address: VIF_Baseaddress + 0x7c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		dec_frame_no3								dec_frame_no2					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		dec_frame_no1								dec_frame_no0					

[29:24]: **dec_frame_no3** - vo's top-field ch-3frame_id for both port1 and port2

[21:16]: **dec_frame_no2** - vo's top-field ch-2frame_id for both port1 and port2

[13:8]: **dec_frame_no1** - vo's top-field ch-1frame_id for both port1 and port2

[5:0]: **dec_frame_no0** - vo's top-field ch-0frame_id for both port1 and port2

VIF-VO Control Register 32 (r32)

Address: VIF_Baseaddress + 0x80

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dec_size[15:0]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				dec_ch[11:0]											

[31:16]: **dec_size[15:0] (unused)**

[11:0]: **dec_ch[11:0]** – vo's 6-window composition channel selection, e.g., if the upper-left window is activated, bits[11][5] are set to "1"; if the lower-right window is activated, bits[0][6] are set to 1; etc.

VIF-VO Control Register 33 (r33)

Address: VIF_Baseaddress + 0x84

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dec_auto_id_top[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
dec_auto_id_top[15:0]															

[31:0]: **dec_auto_id_top[31:0]** – vo's decoding channel's top-field auto_id for port-1 or port-2, set from firmware for ccir656 output from vo. These 32 bits are also assigned as the generic dec_mv_chid1[159:128] in TW2819.

VIF-VO Control Register 34 (r34)

Address: VIF_Baseaddress + 0x88

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dec_detection_id_top[63:48]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
dec_detection_id_top[47:32]															

[31:0]: **dec_detection_id_top[63:32]** - vo's decoding channel's top-field detection_id for port-1 or port-2, set from firmware for ccir656 output from vo. These 32 bits are assigned as the generic dec_mv_chid1[127:96] in TW2819.

VIF-VO Control Register 35 (r35)

Address: VIF_Baseaddress + 0x8c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dec_detection_id_top[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
dec_detection_id_top[15:0]															

[31:0]: **dec_detection_id_top[31:0]** - vo's decoding channel's top-field detection_id for port-1 or port-2, set from firmware for ccir656 output from vo. These 32 bits are assigned as the generic dec_mv_chid1[95:64] in TW2819.

VIF-VO Control Register 36 (r36)

Address: VIF_Baseaddress + 0x90

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dec_user_id_top[63:48]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
dec_user_id_top[47:32]															

[31:0]: **dec_user_id_top[63:32]** – vo's decoding channel's top-field user_id for port-1 or port-2, set from firmware for ccir656 output from vo. These 32 bits are assigned as the generic dec_mv_chid1[63:32] in TW2819.

VIF-VO Control Register 37 (r37)

Address: VIF_Baseaddress + 0x94

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dec_user_id_top[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
dec_user_id_top[15:0]															

[31:0]: **dec_user_id_top[31:0]** – vo's decoding channel's top-field user_id for port-1 or port-2, set from firmware for ccir656 output from vo. These 32 bits are assigned as the generic dec_mv_chid1[31:0] in TW2819.

VIF-VO Control Register 38 (r38)

Address: VIF_Baseaddress + 0x98

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		dec_frame_no7								dec_frame_no6					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		dec_frame_no5								dec_frame_no4					

[29:24]: **dec_frame_no7** - vo's bot- field ch-7frame_id for both port1 and port2

[21:16]: **dec_frame_no6** - vo's bot- field ch-6frame_id for both port1 and port2

[13:8]: **dec_frame_no5** - vo's top- field ch-5frame_id for both port1 and port2

[5:0]: **dec_frame_no4** - vo's top- field ch-4frame_id for both port1 and port2

VIF-VO Control Register 39(r39)

Address: VIF_Baseaddress + 0x9c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		dec_frame_no11								dec_frame_no10					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		dec_frame_no9								dec_frame_no8					

[29:24]: **dec_frame_no11** - vo's bot-field ch-11 frame_id for both port1 and port2.

[21:16]: **dec_frame_no10** - vo's bot-field ch-10 frame_id for both port1 and port2.

[13:8]: **dec_frame_no9** - vo's bot-field ch-9 frame_id for both port1 and port2.

[5:0]: **dec_frame_no8** - vo's bot-field ch-8 frame_id for both port1 and port2.

VIF-VO Control Register 40 (r40)

Address: VIF_Baseaddress + 0xa0

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dec_auto_id_bot[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
dec_auto_id_bot[15:0]															

[31:0]: **dec_auto_id_bot[31:0]** – vo's decoding channel's bot-field auto_id for port-1 or port-2, set from firmware for ccir656 output from vo. These 32 bits are also assigned as the generic dec_mv_chid2[159:128] in TW2819.

VIF-VO Control Register 41 (r41)

Address: VIF_Baseaddress + 0xa4

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dec_detection_id_bot[63:48]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
dec_detection_id_bot[47:32]															

[31:0]: **dec_detection_id_bot[63:32]** - vo's decoding channel's bot-field detection_id for port-1 or port-2, set from firmware for ccir656 output from vo. These 32 bits are also assigned as the generic dec_mv_chid2[127:96] in TW2819.

VIF-VO Control Register 42 (r42)

Address: VIF_Baseaddress + 0xa8

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dec_detection_id_bot[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
dec_detection_id_bot[15:0]															

[31:0]:**dec_detection_id_bot[31:0]** – vo's decoding channel's bot-field detection_id for port-1 or port-2, set from firmware for ccir656 output from vo. These 32 bits are also assigned as the generic dec_mv_chid2[95:64] in TW2819.

VIF-VO Control Register 43 (r43)

Address: VIF_Baseaddress + 0xac

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dec_user_id_bot[63:48]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
dec_user_id_bot[47:32]															

[31:0]: **dec_user_id_bot[63:32]** – vo's decoding channel's bot-field user_id for port-1 or port-2, set from firmware for ccir656 output from vo. These 32 bits are also assigned as the generic dec_mv_chid2[63;32] in TW2819.

VIF-VO Control Register 44 (r44)

Address: VIF_Baseaddress + 0xb0

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dec_user_id_bot[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
dec_user_id_bot[15:0]															

[31:0]: **dec_user_id_bot[31:0]** – vo's decoding channel's bot_field user_id for port-1 or port-2, set from firmware for ccir656 output from vo. These 32 bits are also assigned as the generic dec_mv_chid2[31:0] in TW2819.

VIF-VI-VO Control Register 45 (r45)

Address: VIF_Baseaddress + 0xb4

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								vi_exceptin_en							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo_dcm_wait_limit															

[23:16]: **vi_exception_en** – Special vi setting to enable fifo-full cpu interrupts.

[15:0]: **vo_dcm_wait_limit** – Used to control a vo counter that enables cpu interrupt when the vo_dcm_req waiting period exceeds this limit.

VIF-VI-VO Control Register 46 (r46)

Address: VIF_Baseaddress + 0xb8

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								vif_debug_addr							

[7:0]: **vif_debug_addr[7:0]**– specify which vif debug register firmware wants to read

VIF-VI Information Register 47 (r47 Read-only)

Address: VIF_Baseaddress + 0xbc

Type: Read-only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
auto_id/ch_id															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
auto_id/ch_id															

[31:0]: **auto_id/ch_id[31:0]** - extracted auto_id or chid for firmware.
In TW2819, firmware has to read vi_ch_no16[3] to know whether the auto_id is extracted from either port-1(1) or port-0 (0).

VIF-VO Information Register 48

Address: VIF_Baseaddress + 0xc0

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo4_hhr_offset															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo4_hhr_offset															

[31:0]: **vo4_hhr_offset** - vo4 output frame-line offset for compositions that include hhr.

VIF-VO Information Register 49 (r49)

Address: VIF_Baseaddress + 0xc4

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo4_dcm_cfg0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo4_dcm_cfg0															

[31:0]: **vo4_dcm_cfg0** - dcm_cfg0 register for vo4. See r128 description.

VIF-VO Information Register 50 (r50)

Address: VIF_Baseaddress + 0xc8

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo4_dcm_cfg1															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo4_dcm_cfg1															

[31:0]: **vo4_dcm_cfg1** - vo4_dcm_cfg1 register. See r128 description.

VIF-VO Information Register 51 (r51)

Address: VIF_Baseaddress + 0xcc

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo4_dcm_cfg2															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo4_dcm_cfg2															

[31:0]: **vo4_dcm_cfg2** - vo4_dcm_cfg2 register. See r128 description.

VIF-VO Information Register 52 (r52)

Address: VIF_Baseaddress + 0xd0

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo4_dcm_cfg3															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo4_dcm_cfg3															

[31:0]: **vo4_dcm_cfg3** - vo4_dcm_cfg3 register. See r128 description.

VIF-VO Information Register 53 (r53)

Address: VIF_Baseaddress + 0xd4

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo4_dcm_cfg4															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo4_dcm_cfg4															

[31:0]: **vo4_dcm_cfg4** - vo4_dcm_cfg4 register. See r128 description.

VIF-VO Information Register 54 (r54)

Address: VIF_Baseaddress + 0xd8

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo4_dcm_cfg5															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo4_dcm_cfg5															

[31:0]: **vo4_dcm_cfg5** - vo4_dcm_cfg5 register. See r128 description.

VIF-VO Information Register 55 (r55)

Address: VIF_Baseaddress + 0xdc

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo4_dcm_cfg6															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo4_dcm_cfg6															

[31:0]: **vo4_dcm_cfg6** - vo4_dcm_cfg6 register, which contains field_line_number[10:0] in its lower 11 bits.

VIF-VO Information Register 56 (r56)

Address: VIF_Baseaddress + 0xe0

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo4_dcm_cfg7															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo4_dcm_cfg7															

[31:0]: **vo4_dcm_cfg7** - vo4_dcm_cfg7 register. This register is used to force luma and chroma data into vo display.
Forced_luma[7:0] = cfg7[7:0] (top-field) or cfg7[23:16] (bot-field).
Forced_chroma[7:0] = cfg7[15:8] (top-field) or cfg7[31;24] (bot-field).

VIF-VO Information Register 57 (r57)

Address: VIF_Baseaddress + 0xe4

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo4_offset_0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo4_offset_0															

[31:0]: **vo4_offset_0** - vo4_offset_0 register. See r146 description.

VIF-VO Information Register 58 (r58)

Address: VIF_Baseaddress + 0xe8

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo4_offset_1															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo4_offset_1															

[31:0]: **vo4_offset_1** - vo4_offset_1 register. See r146 description.

VIF-VO Information Register 59 (r59)

Address: VIF_Baseaddress + 0xec

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo4_offset_2															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo4_offset_2															

[31:0]: **vo4_offset_2** - vo4_offset_2 register. See r146 description.

VIF-VI Information Register 60 (r60)

Address: VIF_Baseaddress + 0xf0

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		buf_vo4_id							vbien4	vo4_vbi_size					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					vo4_v_rgn										

[29:23]: **buf_vo4_id** – vo4 buf_id register.
[22]: **vbi_en4** – vo4 vbi_en register.
[21:11]: **vo4_vbi_size** – vo4 vbi_size register.
[10:0]: **vo4_v_rgn** – vo4 v_region register.

VIF-VO Information Register 61 (r61)

Address: VIF_Baseaddress + 0xf4

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo4_boundary_strip															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo4_boundary_strip															

[31:0]: **vo4_boundary_strip[31:0]**– vo4_boundary_strip register. See r152 description.

VIF-VO Information Register 62 (r62)

Address: VIF_Baseaddress + 0xf8

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo4_boundary_value															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo4_boundary_value															

[31:0]: **vo4_boundary_value** - vo4_boundary_value register. See r154 description.

VIF-VO Information Register 63 (r63)

Address: VIF_Baseaddress + 0xfc

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
r1z4					vo4_rgn1_pel										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
r0z4					vo4_rgn0_pel										

[31]: **r1z4** - vo4_rgn1_zero
[26:16]: **vo4_rgn1_pel** - vo4_rgn1_pel[10:0]
[15]: **r0z4** - vo4_rgn0_zero
[10:0]: **vo4_rgn0_pel** - vo4_rgn0_pel[10:0]

VIF-VO Information Register 64 (r64)

Address: VIF_Baseaddress + 0x100

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
r3z4					vo4_rgn3_pel										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
r2z4					vo4_rgn2_pel										

[31]: **r3z4** - vo4_rgn3_zero
[26:16]: **vo4_rgn3_pel** - vo4_rgn3_pel[10:0]
[15]: **r2z4** - vo4_rgn2_zero
[10:0]: **vo4_rgn2_pel** - vo4_rgn2_pel[10:0]

VIF-VO Information Register 65 (r65)

Address: VIF_Baseaddress + 0x104

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
r5z4					vo4_rgn5_pel										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
r4z4					vo4_rgn4_pel										

[31]: **r5z4** - vo4_rgn5_zero
[26:16]: **vo4_rgn5_pel** - vo4_rgn5_pel[10:0]
[15]: **r4z4** - vo4_rgn4_zero
[10:0]: **vo4_rgn4_pel** - vo4_rgn4_pel[10:0]

VIF-VO information register 66 (r66)

Address: VIF_Baseaddress + 0x108

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
r7z4					vo4_rgn7_pel										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
r6z4					vo4_rgn6_pel										

[31]: **r7z4** – vo4_rgn7_zero
[26:16]: **vo4_rgn7_pel** – vo4_rgn7_pel[10:0]
[15]: **r6z4** – vo4_rgn6_zero
[10:0]: **vo4_rgn6_pel** – vo4_rgn6_pel[10:0].

VIF-VI Information Register 67 (r67)

Address: VIF_Baseaddress + 0x10c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	enc_fm_no_bot[5:0]						enc_fm_no_top[5:0]						buf_id[6:4]		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
buf_id[3:0]				v_ln_fm_vi[11:0]											

[31]: **Reserved**

[30:25]: **enc_fm_no_bot[5:0]** - This is the raw-data buf_id for the bottom-field

[31:0]: **enc_fm_no_top[5:0]** - This is the raw-data buf_id for the top-field.

[31:0]: **buf_id[6:0]** - This is the VBI-data buf_id for both top/bottom-fields.

[31:0]: **v_ln_fm_vi[11:0]** - This is the number-of-lines-per-frame for the specified channel.

VIF-VI Information Register 69 (r69)

Address: VIF_Baseaddress + 0x114

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vi_sync_detction_clear[15:0]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vi_sync_lost_clear[15:0]															

[31:16]: **vi_sync_detection_clear[15:0]** – These 16 bits clears the vi_sync_det flags for ch-15 to ch-0 respectively.

[15:0]: **vi_sync_lost_clear[15:0]** – These.16 tis clears the vi_sync_lost flags for ch-15 to ch-0 respectively.

VIF-VI Information Register 70 (r70)

Address: VIF_Baseaddress + 0x118

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

[31:0]: **Reserved**

VIF-VI Information Register 71 (r71)

Address: VIF_Baseaddress + 0x11c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
									h_act_start[11:5]						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
h_act_start[4:0]					vb3[10:0]										

[31:22]: **Reserved**

[22:11]: **h_act_start[11:0]** - It specifies the horizontal-blanking period in vo, e.g., 276. Increased to 12 bits to cover 720p-25's 12'd2680 range.

[10:0]: **vb3[10:0]** – It specifies the vb3, bottom-field blanking-interval-end line number in vo ccir656 or bt1120 format.

VIF-VI Information Register 72 (r72)

Address: VIF_Baseaddress + 0x120

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

[31:0]: **Reserved**

VIF-VI Information Register 73 (r73)

Address: VIF_Baseaddress + 0x124

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

[31:0]: **Reserved**

VIF-VI Information Register 74 (r70)

Address: VIF_Baseaddress + 0x128

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

[31:0]: **Reserved**

VIF-VI Information Register 75 (r75)

Address: VIF_Baseaddress + 0x12c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

[31:0]: **Reserved**

VIF-VI information register 76 (r76)

Address: VIF_Baseaddress + 0x130

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

[31:0]: **Reserved**

VIF-VO Information Register 77 (r77 Read-only)

Address: VIF_Baseaddress + 0x134

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															pts_ctr_vo[32]

[31:0]: **pts_ctr_vo[32]** - vo's PTS counter, bit-32.

VIF-VI Information Register 78 (r78 Read-only)

Address: VIF_Baseaddress + 0x138

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
pts_ctr_vo[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
pts_ctr_vo[15:0]															

[31:0]: **pts_ctr_vo[31:0]** - vo's PTS counter, bit[31:0] for firmware.

VIF-VI Control Register 120 to 127 (r120 to r127, Write-only)

Address: VIF_Baseaddress + 0x1e0-to-0x1fc

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ds_cfg_00_07[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ds_cfg_00_07[15:0]															

[31:0]: **ds_cfg_00_to_07[31:0]** – This is the dual-stream configuration register. It is used to generate the ½, or ¼, or 1/8 sub-sampled dual-stream directly from the main-stream's dcm request's command and data. This saves a full-size read bandwidth for dcm.

Each 32-bit ds_cfg register contains 4 8-bit registers. All together there are 32 8-bit registers defined as follows to control up to 32-frame-id's dual-stream operations.

[7]: **Reserved**

[6]: ds_enable - 1 = enable dual-stream; 0 = disable dual-stream operation.

[5:4]: **line_mode[1:0]** – 1 = div-by-2, 2 = div-by-4, 3 = div-by-8, this controls the sub-sampled dram logical y-address in the y-start field in dcm_cmd.

[3:2]: **x_scale[1:0]** – 1 = div-by-2, 2 = div-by-4, 3 = div-by-8, this controls the sub-sampled data value in the horizontal direction, viz., which data pixel to be retained from the main-stream write-data.

[1:0]: **y_scale[1:0]** – 1 = div-by-2, 2 = div-by-4, 3 = div-by-8, this controls the sub-sampled data value in the vertical direction, viz., which data line to be retained from the original main-stream write data.

VO Port 0 Configuration Register 0 (r128)

Address: VIF_Baseaddress + 0x200

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
valid	luma_vertical_ofst				field_type			chroma_vertical_ofst			horizontal_size				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
burst_size			cvom	y_vertical_size											

[31]: **valid** - It specifies if video output region 1 is valid or not

[30:27]: **luma_vertical_ofst** - It specifies the luma picture offset value

[26:24]: **field_type** - It specifies current field type

0: Top recon-type

1: Bottom recon-type

2: Top raw-type

3: Bottom raw-type

Others: reserved

[23:21]: **chroma_vertical_ofst** - It specifies the chroma picture offset value.

[20:16]: **horizontal_size** - It specifies current picture horizontal size in terms of burst_size.

[15:13]: **burst_size** - It specifies DCM service size:

0: 40 (for HD 1920 = 40x8x6)

1: 44 (for cif/hhr 352 = 44x8x1)

2: 45 (for d1 720 = 45x8x2)

4: 10 (for small 160x128 160 = 10x8x2)

Others: reserved

[12]: **cvom** - It specifies the chroma picture offset value MSB

[11:0]: **y_vertical_size** - It specifies current picture height in pix

Description:

It defines for the region 0 of port 0.

VO Port 0 DCM Configuration Register 1(r129)

Address: VIF_Baseaddress + 0x204

Type: R/W

Description:

The field definitions are the same as r128. It defines for the region 1 of port 0.

DCM Port 0 Configuration Register 2(r130)

Address: VIF_Baseaddress + 0x208

Type: R/W

Description:

The field definitions are the same as r128. It defines for the region 2 of port 0.

DCM Port 0 Configuration Register 3(r131)

Address: VIF_Baseaddress + 0x20C

Type: R/W

Description:

The field definitions are the same as r128. It defines for the region 3 of port 0.

DCM Port 0 Configuration Register 4(r132)

Address: VIF_Baseaddress + 0x210
Type: R/W

Description
The field definitions are the same as r128. It defines for the region 4 of port 0.

DCM Port 0 Configuration Register 5(r133)

Address: VIF_Baseaddress + 0x214
Type: R/W

Description
The field definitions are the same as r128. It defines for the region 5 of port 0.

VO Port 1 DCM Configuration Register 0(r134)

Address: VIF_Baseaddress + 0x218
Type: R/W

Description
The field definitions are the same as r128. It defines for the region 0 of port 1.

VO Port 1 DCM Configuration Register 1(r135)

Address: VIF_Baseaddress + 0x21c
Type: R/W

Description
The field definitions are the same as r128. It defines for the region 1 of port 1.

DCM Port 1 Configuration Register 2(r136)

Address: VIF_Baseaddress + 0x220
Type: R/W

Description
The field definitions are the same as r128. It defines for the region 2 of port 1.

DCM Port 1 Configuration Register 3(r137)

Address: VIF_Baseaddress + 0x224
Type: R/W

Description
The field definitions are the same as r128/. It defines for the region 3 of port 1.

DCM Port 1 Configuration Register 4(r138)

Address: VIF_Baseaddress + 0x229
Type: R/W

Description:
The field definitions are the same as r128. It defines for the region 4 of port 1.

DCM Port 1 Configuration Register 5(r139)

Address: VIF_Baseaddress + 0x22c
Type: R/W

Description:
The field definitions are the same as r128. It defines for the region 5 of port 1.

VO Port Configuration Status Register (r140)

AddressVIF_Baseaddress + 0x230
Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo1_cfg_start	vo3_cfg_start														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo2_cfg_start	vo4_cfg_start														

[31]: **vo1_cfg_start** -1 = Start vo1 operation following vo1_dcm_cfg0-to-7 register settings.
[30]: **vo3_cfg_start** -1 = Start vo3 operation following vo3_dcm_cfg0-to-7 register settings.
[15]: **vo2_cfg_start** -1 = Start vo2 operation following vo2_dcm_cfg0-to-7 register settings.
[14]: **vo4_cfg_start** -1 = Start vo4 operation following vo4_dcm_cfg0-to-7 register settings.

VO Port 0 Active Video Line Register (r141)

Address: VIF_Baseaddress + 0x234
Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					video_active_line_num										

[30:11]: **Reserved**
[10:0]: **video_active_line_num** -It specifies how many vo active video lines for the current picture to control vo_dcm_req commands.

VO Port 0 Active Video Line Register (r142)

Address: VIF_Baseaddress + 0x238
Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo1_forced_chroma_top								vo1_forced_luma_top							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo1_forced_chroma_bot								vo1_forced_luma_bot							

[31:24]: **vo1_forced_chroma_top[7:0]** - It specifies the forced vo-chroma value for top-field.
[23:16]: **vo1_forced_luma_top[7:0]** - It specifies the forced vo-luma value for top-field.
[15:8]: **vo1_forced_chroma_bot[7:0]** - It specifies the forced vo-chroma value for bottom-field.
[7:0]: **vo1_forced_luma_bot[7:0]** - It specifies the forced vo-luma value for bottom-field.

VO Port 0 Active Video Line Register (r143)

Address: VIF_Baseaddress + 0x23c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo1_forced_en	vo2_forced_en														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

[31]: **vo1_forced_en** - It specifies applying the forced vo value to vo port-1.

[30]: **vo2_forced_en** - It specifies applying the forced vo value to vo-port-2.

[29:0]: **Reserved**

VO Port 1 Active Video Line Register (r144)

Address: VIF_Baseaddress + 0x240

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					video_active_line_num										

[30:11]: **Reserved**

[10:0]: **video_active_line_num** - It specifies how many active video lines for the current picture for vo port-2.

VO Port 0 Active Video Line Register (r145)

Address: VIF_Baseaddress + 0x244

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo2_forced_chroma_top								vo2_forced_luma_top							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo2_forced_chroma_bot								vo2_forced_luma_bot							

[31:24]: **vo2_forced_chroma_top[7:0]** –It specifies the forced vo-chroma value for top-field.

[23:16]: **vo2_forced_luma_top[7:0]** –It specifies the forced vo-luma value for top-field.

[15:8]: **vo2_forced_chroma_bot[7:0]** –It specifies the forced vo-chroma value for bottom-field.

[7:0]: **vo2_forced_luma_bot[7:0]** –It specifies the forced vo-luma value for bottom-field.

VO Port 1 Picture Offset Register 0 (r146)

Address: VIF_Baseaddress + 0x248

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
chroma_ofst_rgn1								luma_ofst_rgn1							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
chroma_ofst_rgn0								luma_ofst_rgn0							

[31:24]: **chroma_ofst_rgn1** - It specifies picture chroma offset for region 1

[23:16]: **luma_ofst_rgn1** - It specifies picture luma offset for region 1

[15:8]: **chroma_ofst_rgn0** - It specifies picture chroma offset for region 0

[7:0]: **luma_ofst_rgn0** - It specifies picture luma offset for region 0

VO Port 1 Picture Offset Register 1(r147)

Address: VIF_Baseaddress + 0x24c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
chroma_ofst_rgn3								luma_ofst_rgn3							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
chroma_ofst_rgn2								luma_ofst_rgn2							

[31:24]: **chroma_ofst_rgn3** - It specifies picture chroma offset for region 3.
[23:16]: **luma_ofst_rgn3** - It specifies picture luma offset for region 3.
[15:8]: **chroma_ofst_rgn2** - It specifies picture chroma offset for region 2.
[7:0]: **luma_ofst_rgn2** - It specifies picture luma offset for region 2.

VO Port 1 Picture Offset Register 2(r148)

Address: VIF_Baseaddress + 0x250

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
chroma_ofst_rgn5								luma_ofst_rgn5							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
chroma_ofst_rgn4								luma_ofst_rgn4							

[31:24]: **chroma_ofst_rgn5** - It specifies picture chroma offset for region 5.
[23:16]: **luma_ofst_rgn5** - It specifies picture luma offset for region 5.
[15:8]: **chroma_ofst_rgn4** - It specifies picture chroma offset for region 4.
[7:0]: **luma_ofst_rgn4** - It specifies picture luma offset for region 4.

VO Port 2 Picture Offset Register 0(r149)

Address: VIF_Baseaddress + 0x254

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
chroma_ofst_rgn1								luma_ofst_rgn1							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
chroma_ofst_rgn0								luma_ofst_rgn0							

[31:24]: **chroma_ofst_rgn1** – It specifies picture chroma offset for region 1.
[23:16]: **luma_ofst_rgn1** – It specifies picture luma offset for region 1.
[15:8]: **chroma_ofst_rgn0** – It specifies picture chroma offset for region 0.
[7:0]: **luma_ofst_rgn0** – It specifies picture luma offset for region 0.

VO Port 2 Picture Offset Register 1(r150)

Address: VIF_Baseaddress + 0x258

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
chroma_ofst_rgn3								luma_ofst_rgn3							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
chroma_ofst_rgn2								luma_ofst_rgn2							

[31:24]: **chroma_ofst_rgn3** - It specifies picture chroma offset for region 3.
[23:16]: **luma_ofst_rgn3** - It specifies picture luma offset for region 3.
[15:8]: **chroma_ofst_rgn2** - It specifies picture chroma offset for region 2.
[7:0]: **luma_ofst_rgn2** - It specifies picture luma offset for region 2.

VO Port 2 Picture Offset Register 2(r151)

Address: VIF_Baseaddress + 0x25c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
chroma_ofst_rgn5								luma_ofst_rgn5							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
chroma_ofst_rgn4								luma_ofst_rgn4							

[31:24]: **chroma_ofst_rgn5** - It specifies picture chroma offset for region 5.
[23:16]: **luma_ofst_rgn5** - It specifies picture luma offset for region 5.
[15:8]: **chroma_ofst_rgn4** - It specifies picture chroma offset for region 4.
[7:0]: **luma_ofst_rgn4** - It specifies picture luma offset for region 4.

VO Port 1 Boundary Strip (r152)

Address: VIF_Baseaddress + 0x260

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								strip_left_5		strip_right_5		strip_left_4		strip_right_4	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
strip_left_3		strip_right_3		strip_left_2		strip_right_2		strip_left_1		strip_right_1		strip_left_0		strip_right_0	

[31:24]: **Reserved**
[23:22]: **strip_left_5** - It specifies vertical strips (in terms of 8 byte) to be added to the left of region 5.
[21:20]: **strip_right_5** - It specifies vertical strips (in terms of 8 byte) to be added to the right of region 5.
[19:18]: **strip_left_4** - It specifies vertical strips (in terms of 8 byte) to be added to the left of region 4.
[17:16]: **strip_right_4** - It specifies vertical strips (in terms of 8 byte) to be added to the right of region 4.
[15:14]: **strip_left_3** - It specifies vertical strips (in terms of 8 byte) to be added to the left of region 3.
[13:12]: **strip_right_3** - It specifies vertical strips (in terms of 8 byte) to be added to the right of region 3.
[11:10]: **strip_left_2** - It specifies vertical strips (in terms of 8 byte) to be added to the left of region 2.
[09:08]: **strip_right_2** - It specifies vertical strips (in terms of 8 byte) to be added to the right of region 2.
[07:06]: **strip_left_1** - It specifies vertical strips (in terms of 8 byte) to be added to the left of region 1.
[05:04]: **strip_right_1** - It specifies vertical strips (in terms of 8 byte) to be added to the right of region 1.
[03:02]: **strip_left_0** - It specifies vertical strips (in terms of 8 byte) to be added to the left of region 0.
[01:00]: **strip_right_0** - It specifies vertical strips (in terms of 8 byte) to be added to the right of region 0.

VO Port 2 Boundary Strip (r153)

Address: VIF_Baseaddress + 0x264

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								strip_left_5		strip_right_5		strip_left_4		strip_right_4	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
strip_left_3		strip_right_3		strip_left_2		strip_right_2		strip_left_1		strip_right_1		strip_left_0		strip_right_0	

[31:24]: **Reserved**

[23:22]: **strip_left_5** - It specifies vertical strips (in terms of 8 byte) to be added to the left of region 5.

[21:20]: **strip_right_5** - It specifies vertical strips (in terms of 8 byte) to be added to the right of region 5.

[19:18]: **strip_left_4** - It specifies vertical strips (in terms of 8 byte) to be added to the left of region 4.

[17:16]: **strip_right_4** - It specifies vertical strips (in terms of 8 byte) to be added to the right of region 4.

[15:14]: **strip_left_3** - It specifies vertical strips (in terms of 8 byte) to be added to the left of region 3.

[13:12]: **strip_right_3** - It specifies vertical strips (in terms of 8 byte) to be added to the right of region 3.

[11:10]: **strip_left_2** - It specifies vertical strips (in terms of 8 byte) to be added to the left of region 2.

[09:08]: **strip_right_2** - It specifies vertical strips (in terms of 8 byte) to be added to the right of region 2.

[07:06]: **strip_left_1** - It specifies vertical strips (in terms of 8 byte) to be added to the left of region 1.

[05:04]: **strip_right_1** - It specifies vertical strips (in terms of 8 byte) to be added to the right of region 1.

[03:02]: **strip_left_0** - It specifies vertical strips (in terms of 8 byte) to be added to the left of region 0.

[01:00]: **strip_right_0** - It specifies vertical strips (in terms of 8 byte) to be added to the right of region 0.

VO Port 1 Boundary Strip Value (r154)

Address: VIF_Baseaddress + 0x268

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								strip_y_value							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
strip_u_value								strip_v_value							

[31:24]: **Reserved**

[23:16]: **strip_y_value** - It specifies luma pix value for vertical strip.

[15:8]: **strip_u_value** - It specifies chroma u component pix value for vertical strip.

[7:0]: **strip_v_value** - It specifies chroma v component pix value for vertical strip.

VO Port 2 Boundary Strip Value (r155)

Address: VIF_Baseaddress + 0x268

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								strip_y_value							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
strip_u_value								strip_v_value							

[31:24]: **Reserved**

[23:16]: **strip_y_value**–It specifies luma pix value for vertical strip.

[15:8]: **strip_u_value**–It specifies chroma u component pix value for vertical strip.

[7:0]: **strip_v_value**–It specifies chroma v component pix value for vertical strip.

VIF-VO Information Register 158 (r158 Write-only)

Address: VIF_Baseaddress + 0x278

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo3_hhr_offset															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo3_hhr_offset															

[31:0]: **vo3_hhr_offset** - vo4 output frame-line offset for compositions that include hhr.

VIF-VO Information Register 159 (r159)

Address: VIF_Baseaddress + 0x27c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo3_dcm_cfg0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo3_dcm_cfg0															

[31:0]: **vo3_dcm_cfg0** - dcm_cfg0 register for vo3. See r128 description.

VIF-VO Information Register 160 (r160)

Address: VIF_Baseaddress + 0x280

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo3_dcm_cfg1															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo3_dcm_cfg1															

[31:0]: **vo3_dcm_cfg1** - vo3_dcm_cfg1 register.See r128 description.

VIF-VO information register 161 (r161)

Address: VIF_Baseaddress + 0x284

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo3_dcm_cfg2															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo3_dcm_cfg2															

[31:0]: **vo3_dcm_cfg2** - vo3_dcm_cfg2 register. See r128 description.

VIF-VO information register 162 (r162)

Address: VIF_Baseaddress + 0x288

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo3_dcm_cfg3															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo3_dcm_cfg3															

[31:0]: **vo3_dcm_cfg3** - vo3_dcm_cfg3 register. See r128 description.

VIF-VO Information Register 163 (r163)

Address: VIF_Baseaddress + 0x28c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo3_dcm_cfg4															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo3_dcm_cfg4															

[31:0]: **vo3_dcm_cfg4** - vo3_dcm_cfg4 register. See r128 description.

VIF-VO Information Register 164 (r164)

Address: VIF_Baseaddress + 0x290

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo3_dcm_cfg5															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo3_dcm_cfg5															

[31:0]: **vo3_dcm_cfg5** - vo3_dcm_cfg5 register. See r128 description.

VIF-VO Information Register 165 (r165)

Address: VIF_Baseaddress + 0x294

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo3_dcm_cfg6															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo3_dcm_cfg6															

[31:0]: **vo3_dcm_cfg6** - vo3_dcm_cfg6 register. See r128 description.

VIF-VO Information Register 166 (r166)

Address: VIF_Baseaddress + 0x298

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo3_dcm_cfg7															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo3_dcm_cfg7															

[31:0]: **vo3_dcm_cfg7** - vo3_dcm_cfg7 register. See r128 description.

VIF-VO Information Register 167 (r167)

Address: VIF_Baseaddress + 0x29c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo3_offset_0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo3_offset_0															

[31:0]: **vo3_offset_0** - vo4_offset_0 register. See r146 description.

VIF-VO Information Register 168 (r168)

Address: VIF_Baseaddress + 0x2a0

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo3_offset_1															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo3_offset_1															

[31:0]: **vo3_offset_1** - vo3_offset_1 register. See r147 description.

VIF-VO Information Register 169 (r169)

Address: VIF_Baseaddress + 0x2a4

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo4_offset_2															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo4_offset_2															

[31:0]: **vo4_offset_2** - vo4_offset_2 register. See r148 description.

VIF-VO Information Register 170 (r170)

Address: VIF_Baseaddress + 0x2a8

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		buf_vo3_id							vbi_en3	vo3_vbi_size					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					vo3_v_rgn										

[29:23]: **buf_vo3_id –** vo3 buf_id register
[22]: **vbi_en3 –** vo3 vbi_en register
[21:11]: **vo3_vbi_size –** vo3 vbi_size register
[10:0]: **vo3_v_rgn –** vo3 v_region register

VIF-VO Information Register 171 (r171)

Address: VIF_Baseaddress + 0x2ac

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo3_boundary_strip															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo3_boundary_strip															

[31:0]: **vo3_boundary_strip[31:0]** - vo3_boundary_strip register. See r152 description.

VIF-VO information register 172 (r172)

Address: VIF_Baseaddress + 0x2b0

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo3_boundary_value															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo3_boundary_value															

[31:0]: **vo3_boundary_value** - vo3_boundary_value register. See r154 description.

VIF-VO information register 173 (r173)

Address: VIF_Baseaddress + 0x2b4

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
r1z3					vo3_rgn1_pel										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
r0z3					vo3_rgn0_pel										

[31]: **r1z3 -** vo3_rgn1_zero
[26:16]: **vo3_rgn1_pel** - vo3_rgn1_pel[10:0]
[15]: **r0z3** - vo3_rgn0_zero
[10:0]: **vo3_rgn0_pel** - vo3_rgn0_pel[10:0]

VIF-VO information register 174 (r174)

Address: VIF_Baseaddress + 0x2b8

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
r3z3					vo3_rgn3_pel										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
r2z3					vo3_rgn2_pel										

[31]: **r3z3** - vo3_rgn3_zero
[26:16]: **vo3_rgn3_pel** - vo3_rgn3_pel[10:0]
[15]: **r2z3** - vo3_rgn2_zero
[10:0]: **vo3_rgn2_pel** - vo3_rgn2_pel[10:0]

VIF-VO information register 175 (r175)

Address: VIF_Baseaddress + 0x2bc

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
r5z3					vo3_rgn5_pel										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
r4z3					vo3_rgn4_pel										

[31]: **r5z3** – vo3_rgn5_zero
[26:16]: **vo3_rgn5_pel** – vo3_rgn5_pel[10:0]
[15]: **r4z3** – vo3_rgn4_zero
[10:0]: **vo3_rgn4_pel** – vo3_rgn4_pel[10:0]

VIF-VO Information Register 176 (r176)

Address: VIF_Baseaddress + 0x2c0

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
r7z3					vo3_rgn7_pel										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
r6z3					vo3_rgn6_pel										

[31]: **r7z3** - vo3_rgn7_zero
[26:16]: **vo3_rgn7_pel** - vo3_rgn7_pel[10:0]
[15]: **r6z3** - vo3_rgn6_zero
[10:0]: **vo3_rgn6_pel** - vo3_rgn6_pel[10:0]

VIF VO Special Register for vbi_vo_en1/2 (r185)

Address: VIF_Baseaddress + 0x2e4

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														vbi_vo_en2	vbi_vo_en1

[31:2]: **Reserved**
[1]: **vbi_vo_en2** - It specifies generic vbi-vo enable for vo port-1.
[0]: **vbi_vo_en1** - It specifies generic vbi-vo enable for vo port-2.

VIF VO Special Register for vbi_vo_en1 (r186)

Address: VIF_Baseaddress + 0x2e8

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		vbi_vo_line_start1								vbi_vo_line_end1					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vbi_vo_pixel_start1								vbi_vo_pixel_end1							

[31:30]: **Reserved**
[29:24]: **vbi_vo_line_start1**–It specifies generic vbi-vo line start number for vo port-1.
[21:16]: **vbi_vo_line_end1**–It specifies generic vbi-vo line end number for vo port-1.
[15:8]: **vbi_vo_pixel_start1**–It specifies generic vbi-vo pixel start number for vo port-1.
[7:0]: **vbi_vo_pixel_end1**–It specifies generic vbi-vo pixel end number for vo port-1.

VIF VO Special Register for vbi_vo_en2 (r187)

Address: VIF_Baseaddress + 0x2ec

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		vbi_vo_line_start2								vbi_vo_line_end2					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vbi_vo_pixel_start2								vbi_vo_pixel_end2							

[31:30]: **Reserved**

[29:24]: **vbi_vo_line_start2** - It specifies generic vbi-vo line start number for vo port-2.

[21:16]: **vbi_vo_line_end2** - It specifies generic vbi-vo line end number for vo port-2.

[15:8]: **vbi_vo_pixel_start2** - It specifies generic vbi-vo pixel start number for vo port-2.

[7:0]: **vbi_vo_pixel_end2** - It specifies generic vbi-vo pixel end number for vo port-2.

VIF VO Special Register for vbi Data Read from SDRAM (r201)

Address: VIF_Baseaddress + 0x324

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo1_vbi_en									buf_vo1_id						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					vo1_vbi_size										

[31]: **vo1_vbi_en** - It specifies vbi data read is enabled for vo port-1.

[22:16]: **buf_vo1_id** - It specifies vbi data buf_id for vo port-1.

[10:0]: **vo1_vbi_size** - It specifies vbi data size for vo port-1.

VIF VO Special Register for vbi Data Read from SDRAM (r202)

Address: VIF_Baseaddress + 0x328

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo2_vbi_en									buf_vo2_id						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					vo2_vbi_size										

[31]: **vo2_vbi_en** - It specifies vbi data read is enabled for vo port-2.

[22:16]: **buf_vo2_id** - It specifies vbi data buf_id for vo port-2.

[10:0]: **vo2_vbi_size** - It specifies vbi data size for vo port-2.

VIF VO Special Register for Display-window Selection (r203)

Address: VIF_Baseaddress + 0x32c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		Vo_line_M21	vo_line_M22												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					vo1_v_rgn										

[31:28]: **Reserved**

[29]: **vo_line_M21** - vo-port-1's line-width is a multiple-of-2 (not 4 in most cases).

[28]: **vo_line_M22** - vo-port-2's line-width is a multiple-of-2 (not 4 in most cases).

[10:0]: **vo1_v_rgn** - It specifies generic vo port2 interrupt line number.

VIF VO Special Register for Display-window Selection (r204)

Address: VIF_Baseaddress + 0x330

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo1_rgn1_zero					vo1_rgn1_pel										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo1_rgn0_zero					vo1_rgn0_pel										

[31]: **vo1_rgn1_zero** - It specifies vo1 region1 is blank(1) or not(0).

[26:16]: **vo1_rgn1_pel**–It specifies vo1 region1 zero-out pel number.

[15]: **vo1_rgn0_zero** - It specifies vo1 region0 is blank(1) or not(0).

[10:0]: **vo1_rgn0_pel** - It specifies vo1 region0 zero-out pel number.

VIF VO Special Register for Display-window Selection (r205)

Address: VIF_Baseaddress + 0x334

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo1_rgn3_zero					vo1_rgn3_pel										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo1_rgn2_zero					vo1_rgn2_pel										

[31]: **vo1_rgn3_zero** - It specifies vo1 region3 is blank(1) or not(0).

[26:16]: **vo1_rgn3_pel** - It specifies vo1 region3 zero-out pel number.

[15]: **vo1_rgn2_zero** - It specifies vo1 region2 is blank(1) or not(0).

[10:0]: **vo1_rgn2_pel** - It specifies vo1 region2 zero-out pel number.

VIF VO Special Register for Display-window Selection (r206)

Address: VIF_Baseaddress + 0x338

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo1_rgn5_zero					vo1_rgn5_pel										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo1_rgn4_zero					vo1_rgn4_pel										

[31]: **vo1_rgn5_zero** - It specifies vo1 region5 is blank(1) or not(0).

[26:16]: **vo1_rgn5_pel** - It specifies vo1 region5 zero-out pel number.

[15]: **vo1_rgn4_zero** - It specifies vo1 region4 is blank(1) or not(0).

[10:0]: **vo1_rgn4_pel** - It specifies vo1 region4 zero-out pel number.

VIF VO Special Register for Display-window Selection (r207)

Address: VIF_Baseaddress + 0x33c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo2_rgn1_zero					vo2_rgn1_pel										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo2_rgn0_zero					vo2_rgn0_pel										

[31]: **vo2_rgn1_zero** - It specifies vo2 region1 is blank(1) or not(0).

[26:16]: **vo2_rgn1_pel** - It specifies vo2 region1 zero-out pel number.

[15]: **vo2_rgn0_zero** - It specifies vo2 region0 is blank(1) or not(0).

[10:0]: **vo2_rgn0_pel** - It specifies vo2 region0 zero-out pel number.

VIF VO Special Register for Display-window Selection (r208)

Address: VIF_Baseaddress + 0x340

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo2_rgn3_zero					vo2_rgn3_pel										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo2_rgn2_zero					vo2_rgn2_pel										

[31]: **vo2_rgn3_zero** - It specifies vo2 region3 is blank(1) or not(0)

[26:16]: **vo2_rgn3_pel** - It specifies vo2 region3 zero-out pel number.

[15]: **vo2_rgn2_zero** - It specifies vo2 region2 is blank(1) or not(0)

[10:0]: **vo2_rgn2_pel** - It specifies vo2 region2 zero-out pel number.

VIF VO Special Register for Display-window Selection (r209)

Address: VIF_Baseaddress + 0x344

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo2_rgn5_zero					vo2_rgn5_pel										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo2_rgn4_zero					vo2_rgn4_pel										

[31]: **vo2_rgn5_zero** - It specifies vo2 region5 is blank(1) or not(0)

[26:16]: **vo2_rgn5_pel** - It specifies vo2 region5 zero-out pel number.

[15]: **vo2_rgn4_zero** - It specifies vo2 region4 is blank(1) or not(0)

[10:0]: **vo2_rgn4_pel** - It specifies vo2 region4 zero-out pel number.

VIF VO Special Register for Display-window Selection (r210)

Address: VIF_Baseaddress + 0x348

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo1_rgn7_zero					vo1_rgn7_pel										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo1_rgn6_zero					vo1_rgn6_pel										

[31]: **vo1_rgn7_zero** - It specifies vo1 region7 is blank(1) or not(0).

[26:16]: **vo1_rgn7_pel** - It specifies vo1 region7 zero-out pel number.

[15]: **vo1_rgn6_zero** - It specifies vo1 region6 is blank(1) or not(0).

[10:0]: **vo1_rgn6_pel** - It specifies vo1 region6 zero-out pel number.

VIF VO Special Register for Display-window Selection (r211)

Address: VIF_Baseaddress + 0x34c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
vo2_rgn7_zero					vo2_rgn7_pel										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vo2_rgn6_zero					vo2_rgn6_pel										

[31]: **vo2_rgn7_zero** - It specifies vo2 region7 is blank(1) or not(0).

[26:16]: **vo2_rgn7_pel** - It specifies vo2 region7 zero-out pel number.

[15]: **vo2_rgn6_zero** - It specifies vo2 region6 is blank(1) or not(0).

[10:0]: **vo2_rgn6_pel** - It specifies vo2 region6 zero-out pel number.

VIF VO Special Register for Display-window Selection (r212)

Address: VIF_Baseaddress + 0x350

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
					vo1_right_hhr_offset										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					vo1_left_hhr_offset										

[26:16]: **vo1_right_hhr_offset** - It specifies vo1 right-hhr ch-4offset line number.

[10:0]: **vo1_left_hhr_offset** - It specifies vo1 left-hhr ch-3 offset line number.

VIF VO Special Register for Display-window Selection (r213)

Address: VIF_Baseaddress + 0x354

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
					vo2_right_hhr_offset										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					vo2_left_hhr_offset										

[26:16]: **vo2_right_hhr_offset** - It specifies vo2 right-hhr ch-4offset line number.

[10:0]: **vo2_left_hhr_offset** - It specifies vo2 left-hhr ch-3 offset line number.

VIF VO Special Register for Display-window Selection (r214)

Address: VIF_Baseaddress + 0x358

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					vo2_v_rgn										

[31:11]: **Reserved**

[10:0]: **vo2_v_rgn** - It specifies generic vo port2 interrupt line number.

HIF Registers

This section describes the host interface module registers. The HIF register space is partitioned into three sections:

1. Normal HIF register space
2. PDMA slave register space
3. PDMA master register space

The PDMA register space is defined in the PDMA Register Map section beginning on page 98.

HIF Interrupt

Address: HIF_Baseaddress + 0x00

Type: Read Only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								tx_f	mx_f	tr_e	tx_e	mr_e	mx_e	c_busy	int
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															int_enable

[31:24]: **Reserved**

[23]: **tx_f** – It specifies target transmit buffer fullness.
1: the buffer is full, 0: The buffer is not full

[22]: **mx_f** – It specifies master transmit buffer fullness.
1: The buffer is full, 0: The buffer is not full

[21]: **tr_e** – It specifies target receving buffer emptyness.
1: The buffer is empty, 0: The buffer is not empty

[20]: **tx_e** – It specifies target transmit buffer emptyness.
1: The buffer is empty, 0: The buffer is not empty

[19]: **mr_e** – It specifies master receving buffer emptyness.
1: The buffer is empty, 0: The buffer is not empty

[18]: **mx_e** – It specifies master transmit buffer emptyness.
1: The buffer is empty, 0: The buffer is not empty

[17]: **c_busy** – CBUS masters can only read this bit. If this bit is HIGH, CBUS master cannot issue next serial flash command.

[16]: **int** – It specifies HIF interrupt.

[15:1]: **Reserved**

[0]: **int_enable** – It specifies HIF interrupt enable.
0: Interrupt disable, 1: Interrupt enable, default is 1.

Description:

HIF shall follow the interrupt protocol described in interrupt scheme section.

Device ID

Address: HIF_Baseaddress + 0x04

Type: Read Only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
device_id															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vendor_id															

[31:16]: **device_id** - It specifies device ID. Default is 0x2819.

[15:0]: **vendor_id** - It specifies vendor ID. Default is 0x1797.

PCI Class Code

Address: HIF_Baseaddress + 0x08

Type: Read Only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
class_code															

[31:24]: **Reserved**

[23:0]: **class code** - It specifies PCI class code. The default value is 0x48000.

PCI Sub-system ID

Address: HIF_Baseaddress + 0x0c

Type: Read Only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Subsys_id															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Subsys_vendor_id															

[31:16]: **subsys_id** - It specifies PCI subsystem ID. Default is 0x2819.

[15:0]: **subsys_vendor_id** - It specifies PCI subsystem vendor ID. Default is 0x1797.

PCI Header Info

Address: HIF_Baseaddress + 0x10

Type: Read Only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
header								rev_id							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
max_lat								min_gnt							

[31:24]: **header** - It specifies PCI header.

[23:16]: **rev_id** - It specifies revision ID.

[15:8]: **max_lat** - It specifies PCI max latency.

[23:16]: **min_gnt** - It specifies PCI min grant timing.

DDR Mode Register

Address: HIF base address + 0x14

Type: r/w

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR			PD	WR			DLL	TM	CL			BT	BL		

[31:16]: **Reserved**

[15:14]: **MR** - It specifies mode register definition.

[13]: **Reserved**

[12]: **PD** - It specifies PD mode. 0: Fast exit, 1: Slow exit

[11:9]: **WR** - It specifies the write recovery.

[8]: **DLL** - It specifies the DLL reset. 0: No, 1: Yes

[7]: **TM** - It specifies test mode. 0: Normal operation, 1: Test

[6:4]: **CL** - It specifies the CAS latency.

[3]**BT** - It specifies burst type. 0: Sequenential, 1: Interleaved

[2:0]**BL** - It specifies the burst length.

MR	MODE REGISTER DEFINITION
0 0	Mode register (MR)
0 1	Extended mode register (EMR)
1 0	Extended mode register (EMR2)
1 1	Extended mode register (EMR3)

MR	WRITE RECOVERY
0 0 0	Reserved
0 0 1	2
0 1 0	3
0 1 1	4
1 0 0	5
1 0 1	6
1 1 0	Reserved
1 1 1	Reserved

CL	CAS LATENCY
0 0 0	Reserved
0 0 1	Reserved
0 1 0	Reserved
0 1 1	3
1 0 0	4
1 0 1	5
1 1 0	6
1 1 1	Reserved

BL	BURST LENGTH
0 0 0	Reserved
0 0 1	Reserved
0 1 0	4
0 1 1	8
1 0 0	Reserved
1 0 1	Reserved
1 1 0	Reserved
1 1 1	Reserved

DDR Timing Control Register 0

Address: HIF base address + 0x18

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
wtr		wl				rfc						rc			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rp				rcd				ras				rrd		half_cl	

Description

FW programs DDR timing control register based on DDR speed grade. All timing parameters need to be converted into cycle count in the current core frequency.

DDR Timing Control Register 1

Address: HIF base address + 0x1c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
									ds			mrd			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
power_up_delay															

Description

FW programs DDR timing control register based on DDR speed grade. All timing parameters need to be converted into cycle count in the current core frequency.

FW PDMA Control Register

Address: HIF base address + 0x24

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INT_EN	-														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-													T_TYPE	R/W	GO

[31]: **INT_EN** – Interrupt enable bit. When F/W want to finish PDMA access and GO bit is disabled, Interrupt is assert to ARM after the last transfer.

[30:03]: **Reserved**

[02:02]: **TRANSFER TYPE (T_TYPE)** – Power PC interface type.

"00": Interrupt is assert to Power PC when PDMA's FIFO is prepared data.

"01": Interrupt is only assert to Power PC when PDMA prepare the first 64 words.

[01:01]: **R/W** - Read or Write select bit. When this bit is set, Data in External memory is transferred to the Power PC.

[00:00]: **GO** - DMA request. User should enable the bit after PDMA Command register is written. When this bit is disabled, PDMA transfer is disabled.

FW PDMA Command Register

Address: HIF base address + 0x28

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										BUF_ID					

[31:06]: **Reserved**

[05:00]: **BUF_ID** – Linear buffer id (up to 64 linear buffer).

PDMA Register Map

This section describes PDMA control module registers. PDMA has three kind of operation and is explained seperately in this chapter.

Master Mode

PDMA INTERRUPT STATUS REGISTER FOR FW

Address: HIF base address + 0x48

Type: R/W

31	30	29	28	27	26	25	24
DEBUG_EN						SMBUS_ ERR_INT_ST	MMBUS_ ERR_INT_ST
23	22	21	20	19	18	17	16
	SLV_ INT_ST	PERR_ INT_ST	FATAL_ INT_ST	RX_ INT_ST	TX_ INT_ST	CFG_ INT_ST	COM_ INT_ST
15	14	13	12	11	10	9	8
-						SMBUS_ ERR_INT_EN	MMBUS_ ERR_INT_EN
7	6	5	4	3	2	1	0
	SLV_ INT_EN	PERR_ INT_EN	FATAL_ INT_EN	M_RX_ INT_EN	M_TX_ INT_EN	CFG_ INT_EN	COM_ INT_EN

[31:31]: **DEBUG_EN** - It specifies the debugging enable.

[30:26]: **Reserved**

[25:25]: **SMBUS_ERR_INT_ST** - It specifies the error of mbus for slave mode.

[24:24]: **MMBUS_ERR_INT_ST** - It specifies the error of mbus for master mode.

[23:23]: **Reserved**

[22:22]: **SLV_INT_ST** - It specifies the information of slave interrupt.

[21:21]: **PERR_INT_ST** - It specifies the information of PCI parity error interrupt.

[20:20]: **FATAL_INT_ST** - It specifies the information of PCI fatal error interrupt.

[19:19]: **RX_INT_ST** - It specifies the information of end of RX tranfer interrupt.

[18:18]: **TX_INT_ST** - It specifies the information of end of TX transfer interrupt.

[17:17]: **CFG_INT_ST** - It specifies the information of configuration interrupt.

[16:16]: **COM_INT_ST** - It specifies the information of communcation interrupt.

[15:10]: **Reserved**

[09:09]: **SMBUS_ERR_INT_EN** - It specifies the mbus error interrupt enable for slave mode

[08:08]: **MMBUS_ERR_INT_EN** - It specifies the mbus error interrupt enable for master mode

[07:07]: **Reserved**

[06:06]: **SLV_INT_EN** - It specifies the slave interrupt enable.

[05:05]: **PERR_INT_EN** – It specifies parity error interrupt enable.

[04:04]: **FATAL_INT_EN** - It specifies fatal error interrupt enable.

[03:03]: **M_RX_INT_EN** - It specifies the end of RX transfer interrupt enable.

[02:02]: **M_TX_INT_EN** - It specifies the end of TX transfer interrupt enable.

[01:01]: **CFG_INT_EN -** It specifies the configuration interrupt enable.

[00:00]: COM_INT_EN - It specifies the command interrupt enable.

Description

PDMA supports to manage the interrupt separately between FW and host driver. The interrupt register is only used by FW, not host driver. Even if these interrupts are enabled and host driver interrupts, which address “0x4c” are disable in the same bit, PCI interrupt doesn’t generate to host driver, but it is sent to FW.

For example, FW enables TX transfer. This signifies the end of TX transfer interrupt to be sent into FW, not host driver.

In initial time, FW should enable PCI interrupt enable bits. If PCI interrupt enabled bits are disabled, all of interrupt except COM_INT_EN don’t generate the interrupt. Otherwise, PDMA can keep the interrupt when PDMA generates the interrupt during COM_IN_EN to be disabled. Then PDMA will send the interrupt after COM_IN_EN to be enabled.

PDMA MASTER TX AND RX INTERRUPT STATUS REGISTER

Address: HIF base address + 0x60

Type: Read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														RX END	TX END

[31:02]: **Reserved**

[01:01]: **RX_END** – It specifies end of RX transfer

[00:00]: **TX_END** – It specifies end of TX transfer

Description:

This register isn't used the normal operation. But sometimes FW or host driver doesn't receive PDMA interrupt after finishing the reading and writing data. In that time, FW or host driver could just check the end of transfer using polling.

PDMA MASTER TX AND RX ENDIAN CONTROL REGISTER

Address: HIF Base address + 0x58

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	M_R_ENDIAN [02:00]				M_W_ENDIAN [02:00]				S_R_ENDIAN [02:00]				S_W_ENDIAN [02:00]		

[31:15]: **Reserved**

[14:12]: **M_R_ENDIAN** – It specifies the endian control of read data for master mode.

[11:11]: **Reserved**

[10:08]: **M_W_ENDIAN** – It specifies the endian control of write data for master mode.

[07:07]: **Reserved**

[06:04]: **S_R_ENDIAN** – It specifies the endian control of read data for slave mode.

[03:03]: **Reserved**

[02:00]: **S_W_ENDIAN** – It specifies the endian control of write data for slave mode.

Description:

The Intel processor uses "little ENDIAN", but TW2819 uses "big ENDIAN". In the case host driver wants to change little ENDIAN to big ENDIAN or big ENDIAN to little ENDIAN. So our PDMA is able to support it.

Table 3 bases on the 8 byte, which is the bus width.

TABLE 3.

M_ENDIAN OR S_ENDIAN	DATA FORMAT
0	Data[63:00]
1	Data[39:32], Data[47:40], Data[55:48], Data[63:56] Data[07:00], Data[15:08], Data[23:16], Data[31:24]
2	Data[47:40], Data[39:32], Data[63:56], Data[55:48] Data[15:08], Data[07:00], Data[31:24], Data[23:16]
3	Data[31:24], Data[23:16], Data[15:08], Data[07:00] Data[63:56], Data[55:48], Data[47:40], Data[39:32]
4	Data[55:48], Data[63:56], Data[39:32], Data[47:40] Data[23:16], Data[31:24], Data[07:00], Data[15:08]
5	Data[07:00], Data[15:08], Data[23:16], Data[31:24] Data[39:32], Data[47:40], Data[55:48], Data[63:56]
6	Data[15:08], Data[07:00], Data[31:24], Data[23:16] Data[47:40], Data[39:32], Data[63:56], Data[55:48]
7	Data[23:16], Data[31:24], Data[07:00], Data[15:08] Data[55:48], Data[63:56], Data[39:32], Data[47:40]

PDMA MASTER TX CONTROL REGISTER

Address: HIF Base address + 0x64

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					TX2D_FORMAT		2D_EN								TX_STR

[31:11]: **Reserved**

[10:09]: **TX2D_FORMAT** – It specifies the data format while TX 2D access.

TW2819 only supports RAW data format, or tx2d_format = 2'b01.

[08]: **2D_EN** – It specifies TX 2D access enable.

[07:01]: **Reserved**

[00:00]: **TX_STR** – It specifies PCI DMA TX start request for master mode.

Description:

Host driver or FW should enable "M_TX_START" bit after writing PDMA parameter registers such as TX start address, total length and TX buffer ID. Also "M_TX_START" bit is disabled automatically when TX transfer is finished. However, TX transfer means FW transfers some data into PCI host or host driver reads some data from our external memory.

PDMA MASTER TX BUFFER ID REGISTER

Address: HIF base address + 0x68

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M_TX_FRAME_ID[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M_TX_FRAME_ID[15:00]															

[31:00]: **M_TX_FRAME_ID** – It specifies the buffer ID which FW defines in our frame memory.

Description

The "M_TX_BUF_ID", which is indicated a position in our external memory, is decided by FW at initial time, not host driver. If a host driver wants to know a value of this register, the host driver should get the value from FW using communication protocol.

PDMA MASTER TX TARGET START ADDRESS REGISTER

Address: HIF base address + 0x6C

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M_TX_TAR_ADDR[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M_TX_TAR_ADDR[15:00]															

[31:00]: **M_X_TAR_ADDR** – It specifies the target address which is in host driver.

Description:

When FW transfers some data or the host driver receives data, FW should know the target address, which is the internal buffer in host driver. FW can know the target address using the communication protocol with the host driver.

PDMA MASTER TX TOTAL LENGTH REGISTER

Address: HIF base address + 0x70

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M_TX_TOTAL_LENGTH[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M_TX_TOTAL_LENGTH[15:00]															

[31:00]: **M_TX_TOTAL_LENGTH** – it specifies the TX total length. The value is in terms of 4 bytes. To transfer one SD frame (720x480) using 2D mode, the value is 2*720*480/4 = 172,800. It is actually two times of picture resolution. The reason behind this is that DMA needs to fetch Y, Cr, and Cb.

PDMA MASTER RX CONTROL REGISTER

Address: HIF Base address + 0x78

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					2d_en	rx_lmt	ds_en								rx_start

[31:11]: **Reserved**

[10]: **2d_en** – It is asserted to enable RX 2D transfer

TW2819 does not support 2D master RX mode. This bit should always set to zero.

[09]: **rx_lmt**– It specifies DS limit

[08]: **ds_en** – It is asserted to enable down sample mode.

[00]: **rx_start** – It specifies RX DMA start request.

Description

Host driver or FW can enable the bit, "M_RX_START", after setting PDMA RX parameter registers such as TX start address and total length. Also "M_RX_START" bit is disabled automatically by PDMA when TX transfer is finished. However, RX transfer means host driver writes some data into our external memory.

PDMA MASTER RX BUFFER ID REGISTER

Address: HIF base address + 0x7C

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M_RX_FRAME_ID[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M_RX_FRAME_ID[15:00]															

[31:00]: **M_RX_FRAME_ID** – It specifies the RX FRAME ID which FW defines in our frame memory.

Description

The "M_RX_BUF_ID", which is indicated a buffer in our external memory, is decided by FW, not host driver at initial time. If a host driver wants to know the value of "M_RX_BUF_ID", the host driver should get the value from FW using commulation protocol.

PDMA MASTER RX SOURCE START ADDRESS REGISTER

Address: HIF base address + 0x80

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M_RX_S_ADDR[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M_RX_S_ADDR[15:00]															

[31:00]: **M_RX_S_ADDR** – It specifies the source address which is the start address of internal buffer in host driver.

Description

If host driver wants to send some data into our buffer or FW wants to read some data from host driver's buffer, FW or host driver setup "M_RX_S_ADDR" which is in host driver's start buffer address. So FW should get the information from the host driver using communication register.

PDMA MASTER RX TOTAL LENGTH REGISTER

Address: HIF base address + 0x84

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M_RX_TOTAL_LENGTH[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M_RX_TOTAL_LENGTH[15:00]															

[31:00]: **M_RX_TOTAL_LENGTH** – it specifies the RX total length. The value is in terms of 4 Byte.

PDMA MASTER TX 2D XY START REGISTER

Address: HIF base address + 0xA0

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								M_TX_2D_X_SIZE[07:00]							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					M_TX_2D_Y_SIZE[10:00]										

[31:24]: **Reserved**

[23:16]: **M_TX_2D_X_SIZE**– It specifies the TX 2D horizontal transfer size in terms of 8 Byte data. For example of horizontal size of 720, the value is 720/8 = 90.

[15:11]: **Reserved**

[10:00]: **M_TX_2D_Y_SIZE**– It specifies the TX 2D vertical transfer size in terms of lines. For the example of vertical size of 480, the value is 480.

Slave Mode

HOST PCI driver should control registers for reading and writing.

PDMA SLAVE TX CONTROL REGISTER

Address: HIF Base address + 0x24

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											S_TX_ STRN				S_TX START

[31:05]: **Reserved**

[04:04]: **S_TX_STRN** – It specifies the single transfer enable.

[03:01]: **Reserved**

[00:00]: **S_TX_START** – It specifies DMA start request.

Description

Currently, our host driver only supports the single transfer, not burst transfer for slave mode.

If the host driver uses the single transfer, PCI should enable “S_TX_STRN” bit.

Host driver or FW enables “S_TX_START” bit after finishing to write other TX setting registers such as TX start address and length.

In addition, the performance is so slowly. Because PCI host driver is waiting to write data until receiving PCI interrupt after each 128 word transfer. This means PDMA internal buffer to be limited, 128 word.

PDMA SLAVE TX BUFFER ID REGISTER

Address: HIF base address + 0x28

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										S_TX_BUF_ID[05:00]					

[31:06]: **Reserved**

[05:00]: **S_TX_BUF_ID** – It specifies TX buffer ID

Description:

The S_TX_BUF_ID, which is indicated a buffer ID in our external memory, is decided by FW, not PCI host driver. So the PCI host driver should get the buffer ID from FW using commulation protocol.

PDMA SLAVE TX TOTAL LENGTH REGISTER

AddressHIF base address + 0x2c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
S_TX_TOTAL_LENGTH[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S_TX_TOTAL_LENGTH[15:00]															

[31:00]: **S_TX_TOTAL_LENGTH** – It specifies the TX total length

Description:

When the PCI driver transfers some data into our external memory, the PCI driver writes the total length, which is word count, into the register.

PDMA SLAVE RX CONTROL REGISTER

Address: HIF Base address + 0x34

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													rx_lmt	ds_en	rx_start

[31:03]: **Reserved**

[02]: **rx_lmt**– It specifies DS limit

[01]: **ds_en** – It is asserted to enable down sample mode.

[00]: **rx_start** – It specifies RX DMA start request.

Description:

PCI host driver or FW enables “S_RX_START” bit after finishing to write other RX setting registers such as RX start address and length.

If PCI host driver write some data using 2D access, PCI host driver decides the written format. The information is decides by FW.

PDMA SLAVE RX BUFFER ID REGISTER

Address: HIF base address + 0x38

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										S_R_BUF_ID[05:00]					

[31:06]: **Reserved**

[05:00]: **S_R_BUF_ID** - It specifies the external RX buffer ID

Description:

The “S_R_BUF_ID”, which is one of buffer ID in our external memory, is decided by FW, not PCI host driver. So the PCI host driver should get the “S_R_BUF_ID” from FW using commulation protocol.

PDMA SLAVE RX TOTAL LENGTH REGISTER

Address: HIF base address + 0x3c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
S_RX_TOTAL_LENGTH[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S_RX_TOTAL_LENGTH[15:00]															

[31:00]: S_RX_TOTAL_LENGTH - It specifies the RX total length

Description:

When PCI driver transfers some data into our external memory or FW receives some data from the internal buffer of PCI host driver, PCI driver writes the total length, which is word count, into the register.

PDMA SLAVE INTERRUPT STATUS REGISTER

Address: HIF base address + 0x44

Type: Read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												RX END	RX EMT	TX END	TX FULL

[31:04]: **Reserved**

[03:03]: **RX_END** - It specifies the end of RX transfer.

[02:02]: **RX_EMT** - It specifies the empty of RX FIFO.

[01:01]: **TX_END** - It specifies the end of TX transfer.

[00:00]: **TX_FULL** - It specifies the full of TX FIFO.

Description:

Host driver or FW has to check PCI interrupt status register when the host driver or FW receives PCI interrupt. If “SLV_ST” bit in PCI interrupt status is asserted, host driver or FW should check data in this register. Then PCI host or FW can know what kind of the interrupt for slave mode.

Communication between FW and Host

These registers are used to communicate between FW and host driver.

In the following, the FW and Host driver will be decided some commands for proper processing.

THE COMMUNICATION COMMAND REGISTER FROM HOST DRIVER

Address: HIF base address + 0x8C

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMD_FROM_HOST[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMD_FROM_HOST[15:00]															

[31:00]: **CMD_FROM_HOST**– It specifies a communication command which host driver sends into FW.

Description:

PDMA generates the interrupt to FW when the host driver writes a command in this register. FW will operate the proper task after reading data in the PCI interrupt service routine of FW.

For example, the host driver sends a command which is ready the decoding bitstream after sending the decoding bitstream into a external VLD buffer of TW2819. In that time, host driver uses this register.

THE FIRST EXTRA COMMUNICATION REGISTER FROM HOST DRIVER

Address: HIF base address + 0x54

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMD_EXTRA1_FROM_HOST[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMD_EXTRA1_FROM_HOST[15:00]															

[31:00]: **CMD_EXTRA1_FROM_HOST** – It specifies the first extra communication data.

Description:

Sometimes, the host driver needs to have the first extra data with a command. In that time, the host driver uses the register. The host driver has to write the register before writing the communication register, “Communication Register from Host”.

For example, the host driver wants to send the command, which is the bitstream size, after sending the bitstream for decoding. In that time, the host driver writes the bitstream size into this register. Then FW can know the bitstream size after reading this register.

THE SECOND EXTRA COMMUNICATION REGISTER FROM HOST DRIVER

Address: HIF base address + 0x58

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMD_EXTRA2_FROM_HOST[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMD_EXTRA2_FROM_HOST[15:00]															

[31:00]: **CMD_EXTRA2_FROM_HOST** - It specifies the second extra communication data.

THE COMMUNICATION COMMAND REGISTER FROM FW

Address: HIF base address + 0x90

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMD_FROM_FW[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMD_FROM_FW[15:00]															

[31:00]: **CMD_FROM_FW** – It specifies a communication command which FW sends into host.

Description:

PDMAs able to generate the interrupt to host driver when FW writes this register. After that, host driver will operate the proper task after reading the command.

For example, FW wants to receive a bitstream of slice for decoding. In that time, FW writes a command into the register and the host driver receives the command. Then, the host driver will download the bitstream into external VLD buffer of TW2819. In the further, FW and host driver will define the value of command.

THE FIRST EXTRA COMMUNICATION REGISTER FROM FW

Address: HIF base address + 0x94

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTRA1_COM_FROM_FW[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTRA1_COM_FROM_FW[15:00]															

[31:00]: **EXTRA1_COM_ FROM_FW** - It specifies the first extra communication data.

Description

Sometime, FW needs to have extra data with a command. In that time, FW uses current register. FW has to write the register before writing the communication register, “Communication Register from FW”.

For example, FW sends the command, which is the request bitstream, to the host driver. In that time, FW also send a buffer ID which is defined the external VLD buffer. Then FW writes the buffer ID into the register. Therefore the host driver can know buffer ID after reading this register.

THE SECOND EXTRA COMMUNICATION REGISTER FROM FW

Address: HIF base address + 0x98

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTRA2_COM_FROM_FW[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTRA2_COM_FROM_FW[15:00]															

[31:00]: **EXTRA2_COM_FROM_FW** – It specifies the second extra communication data.

Description

Sometimes, FW needs to have extra data with a communication command. In that time, FW uses the register. FW has to write the register before writing the communication register, “Communication Command Register from FW”.

For example, FW wants to send the command, which is the end of encoding, to the host driver. In that time, FW also send the buffer ID, which is defined in external VLC buffer, and a total length which is the bitstream size of current slice. Then FW uses extra #1 and #2 register. FW write the buffer ID into the register, “The First Communication Extra Register from FW”. Also current register is written the total length.

Interrupt status for PCI Host Driver

PCI Host should read this registers when it receives PCI interrupt from TW2819.

PDMA INTERRUPT STATUS REGISTER FOR PCI CHANNEL

Address: HIF base address + 0x4C

Type: R/W

31	30	29	28	27	26	25	24
INT_ EDGE_EN						SMBUS ERR_INT_ST	MMBUS ERR_INT_ST
23	22	21	20	19	18	17	16
WDOG INT_ST	SLV INT_ST	PERR INT_ST	FATAL INT_ST	RX INT_ST	TX INT_ST	CFG INT_ST	COM INT_ST
15	14	13	12	11	10	9	8
						SMBUS INT_EN	MMBUS INT_EN
7	6	5	4	3	2	1	0
WDOG INT_EN	SLV INT_EN	PERR INT_EN	FATAL INT_EN	RX INT_EN	TX INT_EN	CFG INT_EN	COM INT_EN

[31:31]: **INT_EDGE_EN** - It specifies the edge trigger interrupt.
[30:26]: **Reserved**
[25:25]: **SMBUS_ERR_INT_ST** - It specifies the error of mbus for slave mode.
[24:24]: **MMBUS_ERR_INT_ST** - It specifies the error of mbus for master mode.
[23:23]: **WDOG_INT_ST** - It specifies the information of watch dog.
[22:22]: **SLV_INT_ST** - It specifies the interrupt for slave mode (debug mode).
[21:21]: **PERR_INT_ST** - It specifies the information of parity error interrupt.
[20:20]: **FATAL_INT_ST** - It specifies the information of fatal error interrupt.
[19:19]: **RX_INT_ST** - It specifies the information of end of RX transfer.
[18:18]: **TX_INT_ST** - It specifies the information of end of TX transfer.
[17:17]: **CFG_INT_ST** - It specifies the information of config interrupt.
[16:16]: **COM_INT_ST** - It specifies the information of command interrupt.
[15:08]: **Reserved**
[09:09]: **SMBUS_ERR_INT_EN** - It specifies the mbus error interrupt enable for slave mode.
[08:08]: **MMBUS_ERR_INT_EN** - It specifies the mbus error interrupt enable for master mode.
[07:07]: **WDOG_INT_EN** - It specifies watch dog interrupt enable.
[06:06]: **SLV_INT_EN** - It specifies slave mode interrupt enable.
[05:05]: **PERR_INT_EN** - It specifies PCI parity error interrupt enable.
[04:04]: **FATAL_INT_E**N - It specifies PCI fatal error interrupt enable.
[03:03]: **RX_INT_EN** - It specifies RX END interrupt enable.
[02:02]: **TX_INT_EN** - It specifies TX END interrupt enable.
[01:01]: **CFG_INT_EN** - It specifies Configuration interrupt enable.
[00:00]: **COM_INT_EN** - It specifies Command interrupt enable.

Description: Our PCI Interrupt supports the level trigger or edge trigger. Default is the level trigger interrupt. So if the host driver wants to change the edge trigger, host driver enables “INT_EDGE_EN” bit.

UART Registers

UART Interrupt Enable Register

AddressUART_Baseaddress + 0x00

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															UART interrupt enable

[00]: **UART Interrupt Enable**

Description

UART Interrupt can be enabled by writing bit 0 “1”.

UART Interrupt Status Register

Address: UART_Baseaddress + 0x04

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															UART Interrupt Status

[00]: **UART Interrupt Status**

Description

UART Interrupt Status bit can be cleared by writing bit 0 “1”.

UART Debug Select Register

Address: UART_Baseaddress + 0x08

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								UART Debug Select							

[07: 00]: UART Debug Select

Description

Select different bus for UART debugging.

UART Line Control Register (LCR)

Address: UART_Baseaddress + 0x2C

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8

7	6	5	4	3	2	1	0
Divisor Latch Access bit	Break Control bit	Stick Parity bit.	Even Parity select	Parity Enable	Specify the number of generated stop bits	Select number of bits in each character	

[07]: **Divisor Latch Access bit**.

'1' – The divisor latches can be accessed

'0' – The normal registers are accessed

[06]: **Break Control bit**

'1' – the serial out is forced into logic '0' (break state).

'0' – break is disabled

[05]: **Stick Parity bit**.

'0' – Stick Parity disabled

'1' – If bits 3 and 4 are logic '1', the parity bit is transmitted and checked as logic '0'. If bit 3 is '1' and bit 4 is '0' then the parity bit is transmitted and checked as '1'.

[04]: **Even Parity select**

'0' – Odd number of '1' is transmitted and checked in each word (data and parity combined). In other words, if the data has an even number of '1' in it, then the parity bit is '1'.

'1' – Even number of '1' is transmitted in each word.

[03]: **Parity Enable**

'0' – No parity

'1' – Parity bit is generated on each outgoing character and is checked on each incoming one.

[02]: **Specify the number of generated stop bits**

'0' – 1 stop bit

'1' – 1.5 stop bits when 5-bit character length selected and 2 bits otherwise

Note that the receiver always checks the first stop bit only.

[01:00]: Select number of bits in each character

'00' – 5 bits

'01' – 6 bits

'10' – 7 bits

'11' – 8 bits

Description: The line control register allows the specification of the format of the asynchronous data communication used. A bit in the register also allows access to the divisor latches, which define the baud rate. Reading from the register is allowed to check the current settings of the communication.

Reset Value: 00000011b

UART Divisor Latch Byte 1 Register (LSB)

Address: UART_Baseaddress + 0x20

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								The LSB of the divisor latch							

UART Divisor Latch Byte 2 Register (MSB)

Address: UART_Baseaddress + 0x24

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								The MSB of the divisor latch							

Description: There are 2 clock divisor registers that together form one 16-bit. The registers can be accessed when the 7th (DLAB) bit of the Line Control Register is set to '1'. At this time the registers at addresses 0 to 1 cannot be accessed.

The divisor latches can be accessed by setting the 7th bit of LCR to '1'. You should restore this bit to '0' after setting the divisor latches in order to restore access to the other registers that occupy the same addresses. The 2 bytes form one 16-bit register, which is internally accessed as a single number. You should therefore set all 2 bytes of the register to ensure normal operation. The register is set to the default value of 0 on reset, which disables all serial I/O operations in order to ensure explicit setup of the register in the software. The value set should be equal to (system clock speed) / (16 x desired baud rate).

The internal counter starts to work when the LSB of DL is written, so when setting the divisor, write the MSB first and the LSB last.

UART Internal Interrupt Enable Register (IER)

AddressUART_Baseaddress + 0x24

Type R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8

7	6	5	4	3	2	1	0
Reserved. Should be logic '0'.				Modem Status Interrupt	Receiver Line Status Interrupt	Transmitter Holding Register empty interrupt	Received Data available interrupt

[07: 04]: **Reserved. Should be logic '0'**.

[03]: **Modem Status Interrupt**
'0' – disabled
'1' – enabled

[02]: **Receiver Line Status Interrupt**
'0' – disabled
'1' – enabled

[01]: **Transmitter Holding Register empty interrupt**
'0' – disabled
'1' – enabled

[00]: **Received Data available interrupt**
'0' – disabled
'1' – enabled

Description: Reset Value: 00h

UART Interrupt Identification IIR)

Address: UART_Baseaddress + 0x28

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8

7	6	5	4	3	2	1	0
				Bit 3	Bit 2	Bit 1	Bit 0

Description: The IIR enables the programmer to retrieve what is the current highest priority pending interrupt.
Bit 0 indicates that an interrupt is pending when it's logic '0'. When it's '1' – no interrupt is pending.

Table 4 displays the list of possible interrupts along with the bits they enable, priority, and their source and reset control.

TABLE 4.

BIT 3	BIT 2	BIT 1	PRIORITY	INTERRUPT TYPE	INTERRUPT SOURCE	INTERRUPT RESET
0	1	1	1st	Receiver Line Status	Parity, overrun or framing errors or Break Interrupt	Reading the line status register
0	1	0	2nd	Receiver Data Available	FIFO trigger level reached	FIFO drops below trigger level
1	1	0	2nd	Timeout Indication	There's at least 1 character in the FIFO but no character has been input to the FIFO or read from it for the last 4 Char times.	Reading from the FIFO (receiver buffer register)
0	0	1	3rd	Transmitter Holding Register Empty	Transmitter holding register empty	Writing to the transmitter holding register or reading IIR
0	0	0	4th	Modem Status	CTS, DSR, RI or DCD.	Reading the modem status register

Bits 4 and 5: Logic '0'.

Bits 6 and 7: Logic '1' for compatibility reason.

Reset Value: C1h

UART FIFO Control Register (FCR)

Address: UART_Baseaddress + 0x28

Type: Write only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8

7	6	5	4	3	2	1	0
Define the Receiver FIFO Interrupt trigger level '00' – 1 byte '01' – 4 bytes '10' – 8 bytes '11' – 14 bytes		Ignored			Writing a '1' to bit 2 clears the Transmitter FIFO and resets its logic. The shift register is not cleared, i.e. transmitting of the current character continues.	Writing a '1' to bit 1 clears the Receiver FIFO and resets its logic. But it doesn't clear the shift register, i.e. receiving of the current character continues.	Ignored (Used to enable FIFOs in NS16550D). Since this UART only supports FIFO mode, this bit is ignored.

Description: Write

[7:6]: **Define the Receiver FIFO Interrupt trigger level**

'00' – 1 byte

'01' – 4 bytes

'10' – 8 bytes

'11' – 14 bytes

[5:3]: **Ignored**

[2]: Writing a '1' to bit 2 clears the Transmitter FIFO and resets its logic. The shift register is not cleared, i.e. transmitting of the current character continues.

[1]: Writing a '1' to bit 1 clears the Receiver FIFO and resets its logic. But it doesn't clear the shift register, i.e. receiving of the current character continues.

[0]: **Ignored** (Used to enable FIFOs in NS16550D). Since this UART only supports FIFO mode, this bit is ignored.

Reset Value: 11000000b

UART Modem Control Register (MCR)

Address: UART_Baseaddress + 0x30

Type: Write only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8

7	6	5	4	3	2	1	0
Ignored			Loopback mode '0' – Normal operation '1' – Loopback mode. When in loopback mode, the Serial Output Signal (STX_PAD_O) is set to logic '1'. The signal of the transmitter shift register is internally connected to the input of the receiver shift register. The following connections are made: DTR to DSR RTS to CTS Out1 to RI Out2 to DCD	Out2. In loopback mode, connected to Data Carrier Detect (DCD) input.	Out1. In loopback mode, connected Ring Indicator (RI) signal input	Request To Send (RTS) signal control '0' – RTS is '1' '1' – RTS is '0'	Data Terminal Ready (DTR) signal control '0' – DTR is '1' '1' – DTR is '0'

Description: Reset Value: 0

UART Line Status Register (LSR)

Address: UART_Baseaddress + 0x34

Type: read

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8

7	6	5	4	3	2	1	0
'1' – At least one parity error, framing error or break indications have been received and are inside the FIFO. The bit is cleared upon reading from the register. '0' – Otherwise	Transmitter Empty indicator	Transmit FIFO is empty	Break Interrupt (BI) indicator	Framing Error (FE) indicator	Parity Error (PE) indicator	Overrun Error (OE) indicator	Data Ready (DR) indicator.

Description:

[7]: '1' – At least one parity error, framing error or break indications have been received and are inside the FIFO. The bit is cleared upon reading from the register.

'0' – Otherwise

[6]: Transmitter Empty indicator.

'1' – Both the transmitter FIFO and transmitter shift register are empty. The bit is cleared when data is being been written to the transmitter FIFO.

'0' – Otherwise

[5]: **Transmit FIFO is empty**.

'1' – The transmitter FIFO is empty. Generates transmitter holding register empty interrupt. The bit is cleared when data is being been written to the transmitter FIFO.

'0' – Otherwise

[4]: **Break Interrupt (BI) indicator**

'1' –A break condition has been reached in the current character. The break occurs when the line is held in logic 0 for a time of one character (start bit + data + parity + stop bit). In that case, one zero character enters the FIFO and the UART waits for a valid start bit to receive next character. The bit is cleared upon reading from the register. Generates Receiver Line Status interrupt.

'0' – No break condition in the current character

[3]: **Framing Error (FE) indicator**

'1' – The received character at the top of the FIFO did not have a valid stop bit. Of course, generally, it might be that all the following data is corrupt. The bit is cleared upon reading from the register. Generates Receiver Line Status interrupt.

'0' – No framing error in the current character

[2]: **Parity Error (PE) indicator**

'1' – The character that is currently at the top of the FIFO has been received with parity error. The bit is cleared upon reading from the register. Generates Receiver Line Status interrupt.

'0' – No parity error in the current character

[1]: **Overrun Error (OE) indicator**

'1' – If the FIFO is full and another character has been received in the receiver shift register. If another character is starting to arrive, it will overwrite the data in the shift register but the FIFO will remain intact. The bit is cleared upon reading from the register. Generates Receiver Line Status interrupt.

'0' – No overrun state

[0]: **Data Ready (DR) indicator**.

'0' – No characters in the FIFO

'1' – At least one character has been received and is in the FIFO.

UART Modem Status Register (MSR)

Address: UART_Baseaddress + 0x38

Type: Read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8

7	6	5	4	3	2	1	0
Complement of the DCD input or equals to Out2 in loopback mode.	Complement of the RI input or equals to Out1 in loopback mode.	Complement of the DSR input or equals to DTR in loopback mode.	Complement of the CTS input or equals to RTS in loopback mode.	Delta Data Carrier Detect (DDCD) indicator '1' – The DCD line has changed its state.	Trailing Edge of Ring Indicator (TERI) detector. The RI line has changed its state from low to high state.	Delta Data Set Ready (DDSR) indicator '1' – The DSR line has changed its state.	Delta Clear To Send (DCTS) indicator '1' – The CTS line has changed its state.

Description: The register displays the current state of the modem control lines. Also, four bits also provide an indication in the state of one of the modem status lines. These bits are set to '1' when a change in corresponding line has been detected and they are reset when the register is being read.

[7]: Complement of the DCD input or equals to Out2 in loopback mode.
[6]: Complement of the RI input or equals to Out1 in loopback mode.
[5]: Complement of the DSR input or equals to DTR in loopback mode.
[4]: Complement of the CTS input or equals to RTS in loopback mode.
[3]: Delta Data Carrier Detect (DDCD) indicator
'1' – The DCD line has changed its state.
[2]: Trailing Edge of Ring Indicator (TERI) detector. The RI line has changed its state from low to high state.
[1]: Delta Data Set Ready (DDSR) indicator
'1' – The DSR line has changed its state.
[0]: Delta Clear To Send (DCTS) indicator
'1' – The CTS line has changed its state.

UART Receiver Buffer

Address: UART_Baseaddress + 0x20

Type: Read only

Description: Receiver FIFO output

UART Transmitter Holding Register

Address: UART_Baseaddress + 0x20

Type: Write only

Description: Transmits FIFO input

GPIO Registers

GPIO Interrupt Status Register

Address: GPIO_Baseaddress + 0x00

Type: Read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
															GPIO Interrupt clear
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															GPIO interrupt Enable

[16]: **GPIO Interrupt Clear** – Write a 1'b1 to clear the GPIO interrupt. This bit takes effect every time a 1'b1 is written. Does not need to write 1'b0 in between.

[00]: **GPIO Interrupt Enable** – Write a 1'b1 to Enable GPIO Interrupt

Description: GPIO interrupt status bit can be cleared by CPU writing interrupt clear bit.

GPIO Line Driving Register0

Address: GPIO_Baseaddress + 0x30

Type: Write only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO_LINE_DRVR[13:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIO_LINE_DRVR[15:0]															

[31: 00]: **GPIO Line Driver 31**: 0

Bit i control the output of GPIO[i].

1: GPIO[i] output is 1.

0: GPIO[i] output is 0.

GPIO Line Driving Register1

Address: GPIO_Baseaddress + 0x34

Type: Write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO_LINE_DRVR[63:48]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIO_LINE_DRVR[47:31]															

[31: 00]: **GPIO Line Driver 63:32**

Bit i control the output of GPIO[i+32].

1: GPIO[i+32] output is 1.

0: GPIO[i+32] output is 0.

GPIO Line Control Register0

Address: GPIO_Baseaddress + 0x40

Type: Write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO_LINE_CTL[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIO_LINE_CTL[15:0]															

[31: 00]:: **GPIO Control Register0: Input/Output Control 31:0**

Bit i control the direction of GPIO[i].

1: GPIO[i] is output

0: GPIO[i] is input

GPIO Line Control Register1

Address: GPIO_Baseaddress + 0x44

Type: Write only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO_LINE_CTL[63:48]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIO_LINE_CTL[47:32]															

[31: 00]: **GPIO Control Register1: Input/Output Control 63:32**

Bit i control the direction of GPIO[i+32].

1: GPIO[i+32] is output

0: GPIO[i+32] is input

GPIO Line Load Register0

Address: GPIO_Baseaddress + 0x20

Type: Read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO_LINE_LOAD[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIO_LINE_LOAD15:0]															

[31: 00]: **GPIO Line Load 31:0**

Bit i reflect the input of GPIO[i].

1: GPIO[i] input is 1.

0: GPIO[i] input is 0.

GPIO Line Load Register1

Address: GPIO_Baseaddress + 0x24

Type: Read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPIO_LINE_LOAD[63:48]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIO_LINE_LOAD[47:32]															

[31: 00]: **GPIO Line Load 63:32**

Bit i reflect the input of GPIO[i+32].

1: GPIO[i+32] input is 1.

0: GPIO[i+32] input is 0.

JPEG Register Definitions

This section describes header processing module registers.

JPEG Interrupt

Address: JPEG_Baseaddress + 0x00

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
															int
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															int_enable

[31:17]: **Reserved**

[16]: **int** – It specifies JPEG interrupt

[15:1]: **Reserved**

[0]: **int_enable** – It specifies JPEG interrupt enable.

Description: JPEG shall follow the interrupt protocol described in Interrupt scheme section.

JPEG Buffer id Register

Address: JPEG_Baseaddress + 0x04

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
pic_st															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									buf_id						

[31]: **pic_st** – FW asserts this bit to indicate JPEG encoding start.

[30:7]:**Reserved**

[6:0]:**buf_id** – It specifies buffer id.

JPEG Picture Resolution

Address: JPEG_Baseaddress + 0x08

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
						vertical_size									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						horizontal_size									

[31:26]: **Reserved**

[25:16]: **vertical_size** – It specifies JPEG vertical size in pix.

[15:10]: **Reserved**

[9:0]: **horizontal_size** – It specifies JPEG horizontal size in pix.

JPEG Stream Start Address

Address: JPEG_BaseAddress + 0x10

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JPEG bitstream buffer start address															

[31:25]: **Reserved**

[24:5]: **JPEG_bitstream_buffer_start_address** – The circular input bistream buffer start address.

[5:0]: **Reserved** –This is the LSB 5-bit of the address. It is always 0 since DRAM buffer base address is always at 256 Byte boundary.

Description: FW programs this register after power-up, or at the beginning of video channel switching.

JPEG Stream Buffer End Address

Address: JPEG_BaseAddress + 0x14

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JPEG bitstream buffer end address															

[31:25]: **Reserved**

[24:5]: **JPEG_bitstream_buffer_end_address** – The circular input bistream buffer end address.

[5:0]: **Reserved** –This is the LSB 5-bit of the address. It is always 0 since DRAM buffer base address is always at 256 Byte boundary.

Description: FW programs this register after power-up.

JPEG Stream Data Port

Address: JPEG_BaseAddress + 0x18

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
jpeg_bistream_data															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

[31:0]: **jpeg_bitstream_data** – JPEG video bitstream.

Description: This register contains JPEG circular input bistream data. FW write JPEG bitstream data to this port.

JPEG Data Port Status

Address: JPEG_BaseAddress + 0x1c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
done															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
full											data_length				

[31]: **Done** – FW header encoding done

1 – FW complets header encoding and HW JPEG core shall take over encoding

0 – FW has not done with header encoding.

[30:16]: **Reserved**

[15]: **Full** – FW needs to pull this bit before write data port.

1: The JPEG buffer is full. FW has to wait

0: The JPEG buffer is not full. FW is allowed to write to data port register

[14:5]: **Reserved**

[4:0]: **Data_length** – Data port valid bit count.

1: jpeg_bitstream_data [31] is valid

2: jpeg_bitstream_data[31:30] is valid

:

31: jpeg_bitstream_data[31:1] is valid

0: jpeg_bitstream_data[31:0] is valid

JPEG Qscale Register

Address: JPEG_Baseaddress + 0x20

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
qscale															

[31:16]: **Reserved**

[15:0]: **qscale** - It specifies qscale factor for JPEG quantizer operation.

JPEG Control Register

Address: JPEG_Baseaddress + 0x24

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															d601

[0]: d601 – Enable scaling from 601 data range to 0 to 255 range.

JPEG Status Register

Address: JPEG_Baseaddress + 0x28

Type: Read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
										core_idle	mbus_idle	jpx_ostb	Jpx_ordy	jout_ostb	jout_ordy
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
mb_cnt[15:0]															

Description: JPEG debug status registers, for hardware debug, not used by firmware.

21: core_idle

20: mbus_idle

19: jpx_ostb

18: jpx_ordy

17: jout_ostb

16: jout_ordy

[15:0]: mb_cnt[15:0]

JPEG Bitstream Length Register

Address: JPEG_Baseaddress + 0x2c

Type: Read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JPEG_BITSTREAM_LENGTH[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JPEG_BITSTREAM_LENGTH[15:0]															

Description: JPEG bit stream bit length

Audio Codec Register Definitions

AUD Interrupt

Address: AUD_Baseaddress + 0x00

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
												dec_idle	aud_cmp_int	aud_enc_int	aud_dec_int
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													cmp_int_clr	enc_int_clr	dec_int_clr

[19]: **dec_idle** – Audio decocde is idle (status).

[18]: **aud_cmp_int** - Audio encoder compression Interrupt flag (status).

[17]: **aud_enc_int** - Audio encoder raw data Interrupt flag (status).

[16]: **aud_dec_int** - Audio decoder Interrupt flag (status).

[2]: **cmp_int_clr** - Audio encoder compression interrupt clear bit.

[1]: **enc_int_clr** - Audio encoder raw data interrupt clear bit.

[0]: **dec_int_clr** - Audio decoder interrupt clear bit.

Description: Audio module interrupt control.

AUD Encoder Time Code

Address: AUD_Baseaddress + 0x04

Type: Read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
encoder_time_code [31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
encoder_time_code [15:0]															

[31:0]: **encoder_time_code** – audio encoder time code

Description: Audio module encoder time code.

AUD Control Registers

Address: AUD_Baseaddress + 0x08

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
enc_time code[32]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					cmp_int_en	enc_int_en	dec_int_en						cmp_go	enc_go	dec_go

[31]: encoder_time_code[32] - Audio encoder time code bit [32].

[10]: cmp_int_en - Audio encoder compression interrupt enable bit.

[9]: enc_int_en - Audio encoder raw data interrupt enable bit.

[8]: dec_int_en - Audio decoder interrupt enable bit.

[2]: cmp_go - Audio encoder compression start bit.

[1]: enc_go - Audio encoder raw data start bit.

[0]: dec_go - Audio decoder start bit.

Description: Audio module control registers

AUD Decoder Buf ID

Address: AUD_Baseaddress + 0x0c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									dec_buf_id						

[31:24]: **Reserved** - Audio reserved (test only)

[6:0]: **dec_buf_id** - Audio decoder linear buffer ID

Description: Audio module decoder linear buffer ID

AUD Decoder Control

Address: AUD_Baseaddress + 0x10

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
												decRawMode			decMute
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			dec_BlockSamples												

[19]: **decRawMode** - Audio decocde raw data mode (0: adpcm decode mode).

[16]: **decMute** - Audio decoder mute bit.

[12:0]: **dec_BlockSamples** - Audio decoder block sample size.

Description: Audio module decoder control.

AUD Decoder Control

Address: AUD_Baseaddress + 0x14

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dec_byte_cnt															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			decChValid												dec_I2S_n

[31:16]: dec_byte_cnt – Audio decocde byte count for decoder interrupt.

[12]: decChValid – Audio decoder channel valid bit (always set to “1”).

[0]: dec_I2S_n – Audio decoder I2S mode (always set to “0”).

Description: Audio module decoder control.

AUD Encoder Raw Data Buffer IDs

Address: AUD_Baseaddress + 0x18

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	enc_ch3_buf_id								enc_ch2_buf_id						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	enc_ch1_buf_id								enc_ch0_buf_id						

[30:24]: **enc_ch3_buf_id** - Audio encoder channel 3 raw data buffer ID.
[22:16]: **enc_ch2_buf_id** - Audio encoder channel 2 raw data buffer ID.
[14:8]: **enc_ch1_buf_id** - Audio encoder channel 1 raw data buffer ID.
[6:0]: **enc_ch0_buf_id** - Audio encoder channel 0 raw data buffer ID.

Description: Audio module encoder raw data buffer IDs

AUD Encoder Raw Data Buffer IDs

Address: AUD_Baseaddress + 0x1c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	enc_ch7_buf_id								enc_ch6_buf_id						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	enc_ch5_buf_id								enc_ch4_buf_id						

[30:24]: **enc_ch7_buf_id** - Audio encoder channel 7 raw data buffer ID.
[22:16]: **enc_ch6_buf_id** - Audio encoder channel 6 raw data buffer ID.
[14:8]: **enc_ch5_buf_id** - Audio encoder channel 5 raw data buffer ID.
[6:0]: **enc_ch4_buf_id** - Audio encoder channel 4 raw data buffer ID.
Description: Audio module encoder raw data buffer IDs.

AUD Encoder Raw Data Buffer IDs

Address: AUD_Baseaddress + 0x20

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	enc_chb_buf_id								enc_cha_buf_id						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	enc_ch9_buf_id								enc_ch8_buf_id						

[30:24]: **enc_chb_buf_id** – Audio encoder channel b raw data buffer ID.
[22:16]: **enc_cha_buf_id** – Audio encoder channel a raw data buffer ID.
[14:8]: **enc_ch9_buf_id** – Audio encoder channel 9 raw data buffer ID.
[6:0]: **enc_ch8_buf_id** – Audio encoder channel 8 raw data buffer ID.
Description: Audio module encoder raw data buffer IDs

AUD Encoder Raw Data Buffer IDs

Address: AUD_Baseaddress + 0x24

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	enc_chf_buf_id								enc_che_buf_id						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	enc_chd_buf_id								enc_chc_buf_id						

[30:24]: **enc_chf_buf_id** - Audio encoder channel f raw data buffer ID

[22:16]: **enc_che_buf_id** - Audio encoder channel e raw data buffer ID

[14:8]: **enc_chd_buf_id** - Audio encoder channel d raw data buffer ID

[6:0]: **enc_chc_buf_id** - Audio encoder channel f raw data buffer ID

Description: Audio module encoder raw data buffer IDs

AUD Encoder Raw Data Control

Address: AUD_Baseaddress + 0x28

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
enc_chvalid															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
enc_bytecnt															

[31:16]: **enc_chvalid** - Audio encoder channel valid bit;

[31]: channel 0; [30] channel 1; [29] channel 2;....[16] channel f;

[15:0]: **enc_bytecnt** - Audio encoder raw data byte count for encoder interrupt

Description: Audio module encoder raw data control

AUD Encoder Raw Data Control

Address: AUD_Baseaddress + 0x2c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
											encChNumber				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	enc2pinEn			encSmp8bitMode			encRawMode								enc_I2S_n

[20:16]: **enChNumber** - Audio encoder raw data channel number (1: 1 channels; 2: channels).

[14]: **enc2pinEn** - Test purpose (always set to “0”).

[11]: **encSmp8bitMode** - Audio encoder raw data 8 bits mode.

[8]: **encRawMode** - Audio_encoder raw data mode (always set to “1”).

[0]: **enc_I2S_n** - Audio encoder I2S mode (always set to “0”).

Description: Audio module encoder raw data control.

AUD Encoder Compression Linear Buffer IDs

Address: AUD_Baseaddress + 0x30

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	ch3_compbuf_id								ch2_compbuf_id						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ch1_compbuf_id								ch0_compbuf_id						

[30:24]: **ch3_compbuf_id** - Audio encoder compression channel 3 buf ID
[22:16]: **ch2_compbuf_id** - Audio encoder compression channel 2 buf ID
[14:8]: **ch1_compbuf_id** - Audio encoder compression channel 1 buf ID.
[6:0]: **ch0_compbuf_id** - Audio encoder compression channel 0 buf ID
Description: Audio module encoder compression buffer IDs

AUD Encoder Compression Linear Buffer IDs

Address: AUD_Baseaddress + 0x34

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	ch7_compbuf_id								ch6_compbuf_id						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ch5_compbuf_id								ch4_compbuf_id						

[30:24]: **ch7_compbuf_id** - Audio encoder compression channel 7 buf ID
[22:16]: **ch6_compbuf_id** - Audio encoder compression channel 6 buf ID
[14:8]: **ch5_compbuf_id** - Audio encoder compression channel 5 buf ID
[6:0]: **ch4_compbuf_id** - Audio encoder compression channel 4 buf ID
Description: Audio module encoder compression buffer IDs

AUD Encoder Compression Linear Buffer IDs

Address: AUD_Baseaddress + 0x38

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	chb_compbuf_id								cha_compbuf_id						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ch9_compbuf_id								ch8_compbuf_id						

[30:24]: **chb_compbuf_id** – Audio encoder compression channel b buf ID
[22:16]: **cha_compbuf_id** – Audio encoder compression channel a buf ID
[14:8]: **ch9_compbuf_id** – Audio encoder compression channel 9 buf ID
[6:0]: **ch8_compbuf_id** – Audio encoder compression channel 8 buf ID
Description: Audio module encoder compression buffer IDs

AUD Encoder Compression Linear Buffer IDs

Address: AUD_Baseaddress + 0x3c

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	chf_compbuf_id								che_compbuf_id						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	chd_compbuf_id								chc_compbuf_id						

[30:24]: **chf_compbuf_id** – Audio encoder compression channel f buf ID

[22:16]: **che_compbuf_id** – Audio encoder compression channel e buf ID

[14:8]: **chd_compbuf_id** – Audio encoder compression channel d buf ID

[6:0]: **chc_compbuf_id** – Audio encoder compression channel c buf ID

Description: Audio module encoder compression buffer IDs

AUD Encoder Compression Control

Address: AUD_Baseaddress + 0x40

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
rawChannelReady															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
channelNoforComp															

[31:16]: **rawChannelReady** - Audio raw data channel ready bit

[31] channel f; [30] channel e; [16] channel 0;

[15:0]: **channelNoforComp** - Audio encoder compression channel number (1: 1ch for compression; 2: 2 chs for compression)

Description: Audio module encoder compression control bit

AUD Encoder Compression Control

Address: AUD_Baseaddress + 0x44

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
							compBlockNoForChangeCh								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			comp_BlockSamples												

[24:20]: **compBlockNoForChangeCh** - Block number for change channel (multi channels use)

[12:0]: **comp_BlockSamples** - Audio encoder compression block sample size

Description: Audio module encoder compression control

Multichannel Audmux Control

Address: AUD_Baseaddress + 0x50

Type: R/W

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ch_swap															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
mux_mode_3				mux_mode_2				mux_mode_1				mux_mode_0			

[31:30]: **ch_swap** – Swap channel 1, 2, or 3 with channel 0.

0: Internal audio channel unchanged

1: Internal swap audio channel 0 with channel 1

2: Internal swap audio channel 0 with channel 2

3: Internal swap audio channel 0 with channel 3

[15:12]: **mux_mode_3** – Audio mux mode for audio channel 3

[11:08]: **mux_mode_2** – Audio mux mode for audio channel 2

[07:04]: **mux_mode_1** – Audio mux mode for audio channel 1

[03:00]: **mux_mode_0** – Audio mux mode for audio channel 0

Description: Audio mux mode mux_mode[3:0] definition:

mux_enable = mux_mode[3]: set to HIGH to mux the specified audio channel to the mixed 16-channel audio.

mux_audio_clk = mux_mode[2]: set to HIGH to use falling edge audio clock to capture audio sample. Set to LOW to use raising edge audio clock to capture audio sample.

mux_audio_rate = mux_mode[1:0]: It specifies the incoming audio data down sample rate.

mux_audio_rate = 0: Capture one incoming audio data every 4 audio samples.

mux_audio_rate = 1: Capture one incoming audio data every 2 audio samples.

mux_audio_rate = 2: Capture every incomping audio samples.

Peripheral Timing

This section describes timing restrictions for TW2819 peripheral interface ports such as video interface audio interface. Timing for other industry standard peripherals such as PCI and DDR2 are not covered in this datasheet. Customers can refer to related documents for details on timing information.

Video Interface

TW2819 supports 4 video input ports and four video output ports. Their timing information is shown in Figure 9 and Table 5.

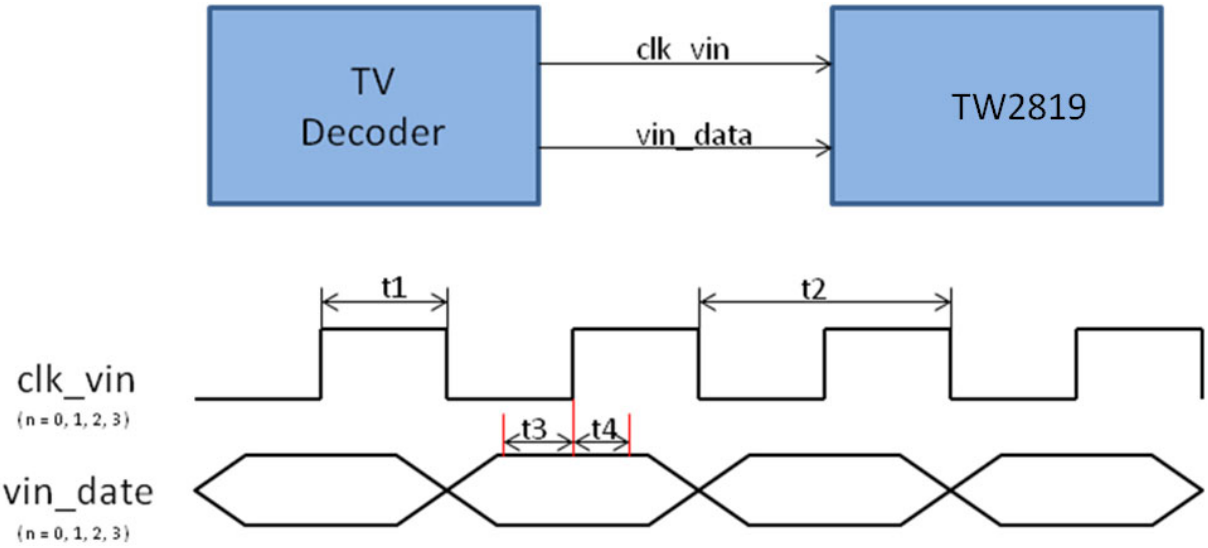


FIGURE 14. VIDEO INPUT PORT TIMING DIAGRAM

TABLE 5. VIDEO INPUT PORT AC TIMING

PARAMETER	SYMBOL	MIN	TYP	MAX	UNIT
Input Clock Half Period	t1	4.39	4.62 (108MHz)	19.45	ns
			6.76 (74MHz)		ns
			9.26 (54MHz)		ns
			18.52 (27MHz)		ns
Input Clock Period	t2	8.78	9.25 (108MHz)	38.8	ns
			13.51 (74MHz)		ns
			18.51 (54MHz)		ns
			37.03 (27MHz)		ns
Input Data Setup Time	t3	3			ns
Input Data Hold Time	t4	1.5			ns

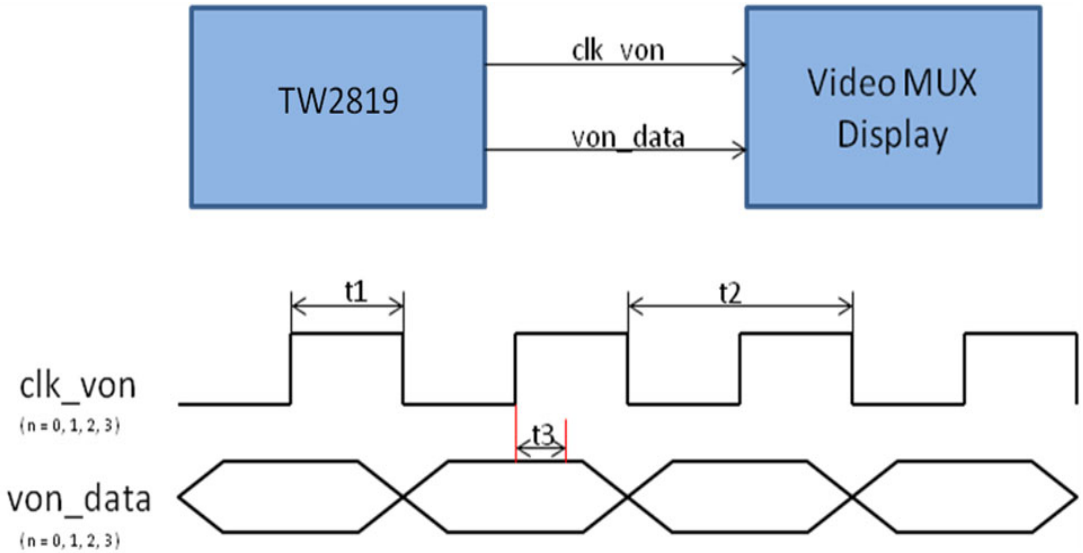


FIGURE 15. VIDEO OUTPUT PORT TIMING DIAGRAM

TABLE 6. VIDEO OUTPUT PORT AC TIMING

PARAMETER	SYMBOL	MIN	TYP	MAX	UNIT
Output Clock Half Period	t1	4.39	4.62 (108MHz)	19.45	ns
			6.76 (74MHz)		ns
			9.26 (54MHz)		ns
			18.52 (27MHz)		ns
Output Clock Period	t2	8.78	9.25 (108MHz)	38.8	ns
			13.51 (74MHz)		ns
			18.51 (54MHz)		ns
			37.03 (27MHz)		ns
Output Data Delay	t3	2.6		5.2	ns

Audio Interface

This section describes audio interface port timing. TW2819 audio interfaces follow industry standard I2S protocol with modifications to support multichannel audio.

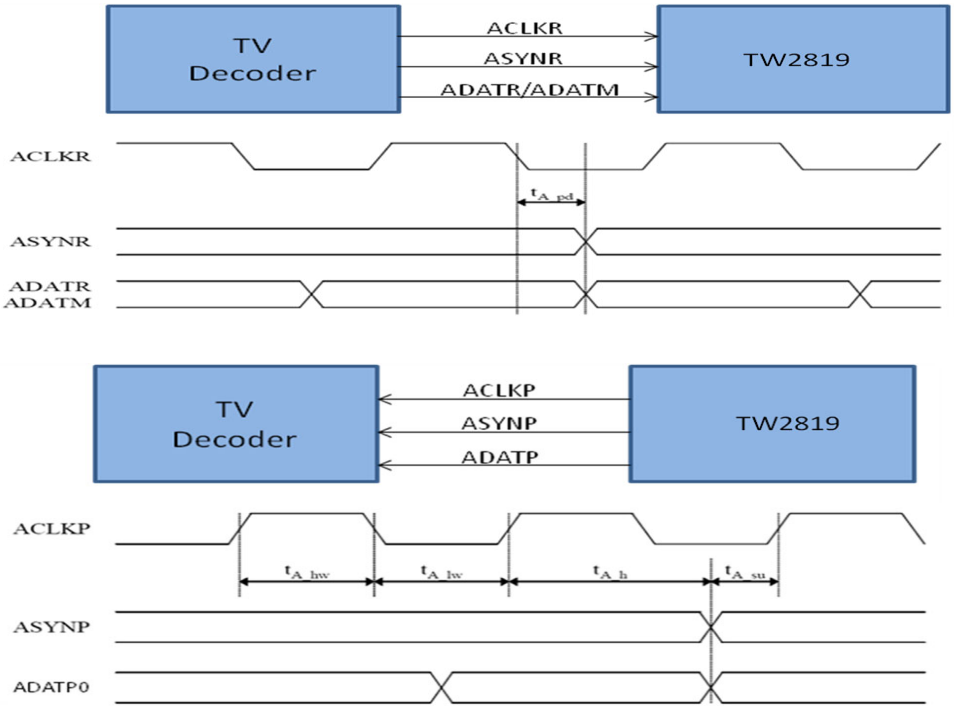


FIGURE 16. AUDIO INPUT/OUTPUT PORT TIMING DIAGRAM

TABLE 7. AUDIO INPUT PORT AC TIMING

PARAMETER	SYMBOL	MIN	TYP	MAX	UNIT
Input Clock Half Period	1	8333.2	10416.5 (48kHz)	19.45	ns
		12500	15625 (32kHz)	19.45	ns
Input Clock Period	2		20833 (48kHz)	38.8	ns
			31250 (32kHz)		ns
Input Data Setup Time	3	8			ns
Input Data Hold Time	4	3			ns

PCI Interface

This section describes PCI interface timing for TW2819.

The PCI of the TW2819 is designed to be satisfied with the protocol and electrical features of the PCI Local Bus Specification, Revision 2.2 32bit/33MHz (66MHz). The PCI interface requires a minimum of 47 pins for a target-only device to handle data and addressing, interface control, arbitration and system functions.

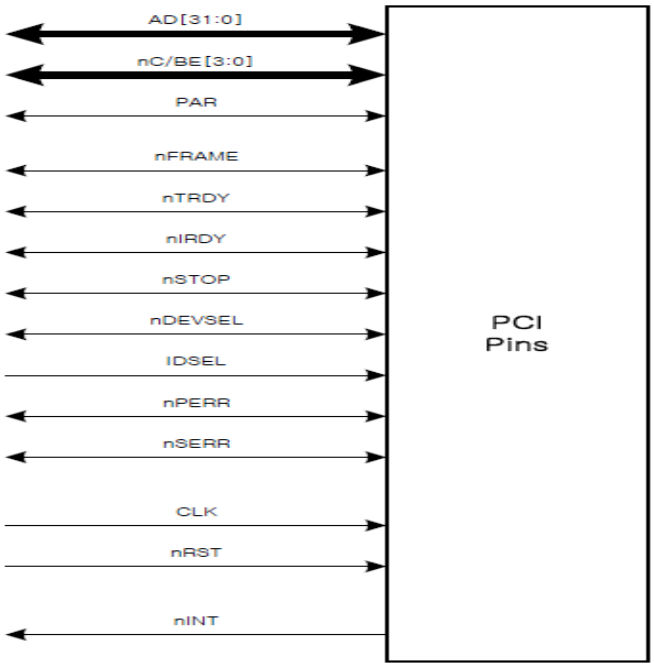


FIGURE 17. PCI PINS INTERFACE

in	*Input* is a standard input-only signal.
out	*Totem Pole Output* is a standard active driver.
t/s	*Tri-State* is a bi-directional, tri-state input/output pin.
s/t/s	*Sustained Tri-State* is an active low tri-state signal owned and driven by one and only one agent at a time. The agent that drives an s/t/s pin low must drive it high for at least one clock before letting it float. A new agent cannot start driving a s/t/s signal any sooner than one clock after the previous owner tri-states it. A pullup is required to sustain the inactive state until another agent drives it and must be provided by the central resource.
o/d	*Open Drain* allows multiple devices to share as a wire-OR. A pull-up is required to sustain the inactive state until another agent drives it and must be provided by the central resource.

TABLE 8. PCI PIN LIST

PIN NAME	FUNCTION	DESCRIPTION
SYSTEM PINS		
CLK	in	*Clock* provides timing for all transactions on PCI and is an input to every PCI device. All other PCI signals, except **nRST**, and **nINTD**, are sampled on the rising edge of **CLK** and all other timing parameters are defined with respect to this edge. PCI operates up to 33MHz and, in general, the minimum frequency is DC (0Hz).
nRST	in	*Reset* is used to bring PCI-specific registers, sequencers and signals to a consistent state. What effect **nRST** has on a device beyond the PCI sequencer is beyond the scope of this specification, except for reset states of required PCI configuration registers. A device that can wake the system while in a powered down bus state has additional requirements related to **nRST**. Refer to the *PCI Power Management Interface Specification* for details. Any time **nRST** is asserted, all PCI output signals must be driven to their benign state. In general, this means they must be asynchronously tri-stated. **nREQ** and **nGNT** must both be tri-stated (they cannot be driven low or high during reset). To prevent **AD**, **nC/BE** and **PAR** signals from floating during reset, the central resource may drive these lines during reset (bus parking) but only to a logic low level; they may not be driven high. **nRST** may be asynchronous to **CLK** when asserted or deasserted. Although asynchronous, deassertion is guaranteed to be a clean, bounce-free edge. Except for configuration accesses, only devices that are required to boot the system will respond after reset.
ADDRESS AND DATA PINS		
AD[31:0]	t/s	*Address* and *Data* are multiplexed on the same PCI pins. A bus transaction consists of an address2 phase followed by one or more data phases. PCI supports both read and write bursts. The address phase is the first clock cycle in which **nFRAME** is asserted. During the address phase, **AD[31:0]** contain a physical address (32 bits). For I/O, this is a byte address; for configuration and memory, it is a DWORD address. During data phases, **AD[07:0]** contain the least significant byte (LSB) and **AD[31:24]** contain the most significant byte (MSB). Write data is stable and valid when **nIRDY** is asserted; read data is stable and valid when **nTRDY** is asserted. Data is transferred during those clocks where both **nIRDY** and **nTRDY** are asserted.
nC/BE[3:0]	t/s	*Bus Command* and *Byte Enables* are multiplexed on the same PCI pins. During the address phase of a transaction, **nC/BE[3:0]** define the bus command. During the data phase, **nC/BE[3:0]** are used as Byte Enables. The Byte Enables are valid for the entire data phase and determine which byte lanes carry meaningful data. **nC/BE[0]** applies to byte 0 (LSB) and **nC/BE[3]** applies to byte 3 (MSB).
PAR	t/s	*Parity* is even3 parity across **AD[31:0]** and **nC/BE[3:0]**. Parity generation is required by all PCI agents. **PAR** is stable and valid one clock after each address phase. For data phases, **PAR** is stable and valid one clock after either **nIRDY** is asserted on a write transaction or **nTRDY** is asserted on a read transaction. Once **PAR** is valid, it remains valid until one clock after the completion of the current data phase. (**PAR** has the same timing as **AD[31:0]**, but it is delayed by one clock.) The master drives **PAR** for address and write data phases; the target drives **PAR** for read data phases.
INTERFACE CONTROL PINS		
nFRAME	s/t/s	*Cycle Frame* is driven by the current master to indicate the beginning and duration of an access. **nFRAME** is asserted to indicate a bus transaction is beginning. While **nFRAME** is asserted, data transfers continue. When **nFRAME** is deasserted, the transaction is in the final data phase or has completed.
nIRDY	s/t/s	*Cycle Frame* is driven by the current master to indicate the beginning and duration of an access. **nFRAME** is asserted to indicate a bus transaction is beginning. While **nFRAME** is asserted, data transfers continue. When **nFRAME** is deasserted, the transaction is in the final data phase or has completed.
nTRDY	s/t/s	*Target Ready* indicates the target agent's (selected device's) ability to complete the current data phase of the transaction. **nTRDY** is used in conjunction with **nIRDY**. A data phase is completed on any clock both **nTRDY** and **nIRDY** are asserted. During a read, **nTRDY** indicates that valid data is present on **AD[31:0]**. During a write, it indicates the target is prepared to accept data. Wait cycles are inserted until both **nIRDY** and **nTRDY** are asserted together.
nSTOP	s/t/s	*Stop* indicates the current target is requesting the master to stop the current transaction.
nLOCK	s/t/s	*Lock* indicates an atomic operation to a bridge that may require multiple transactions to complete. When **nLOCK** is asserted, nonexclusive transactions may proceed to a bridge that is not currently locked. A grant to start a transaction on PCI does not guarantee control of **nLOCK**. Control of **nLOCK** is obtained under its own protocol in conjunction with **nGNT**. It is possible for different agents to use PCI while a single master retains ownership of **nLOCK**. Locked transactions may be initiated only by host bridges, PCI-to- PCI bridges, and expansion bus bridges.
IDSEL	in	*Initialization Device Select* is used as a chip select during configuration read and write transactions.

TABLE 8. PCI PIN LIST (Continued)

PIN NAME	FUNCTION	DESCRIPTION
nDEVSEL	s/t/s	*Device Select*, when actively driven, indicates the driving device has decoded its address as the target of the current access. As an input, **nDEVSEL** indicates whether any device on the bus has been selected.
ARBITRATION PINS (BUS MASTERS ONLY)		
nREQ	t/s	*Request* indicates to the arbiter that this agent desires use of the bus. This is a point-to-point signal. Every master has its own **nREQ** which must be tri-stated while **nRST** is asserted.
nGNT	t/s	*Grant* indicates to the agent that access to the bus has been granted. This is a point-to-point signal. Every master has its own **nGNT** which must be ignored while **nRST** is asserted.
ERROR REPORTING PINS		
nPERR	s/t/s	*Parity Error* is only for the reporting of data parity errors during all PCI transactions except a Special Cycle. The **nPERR** pin is sustained tri-state and must be driven active by the agent receiving data (when enabled) two clocks following the data when a data parity error is detected. The minimum duration of **nPERR** is one clock for each data phase that a data parity error is detected. (If sequential data phases each have a data parity error, the **nPERR** signal will be asserted for more than a single clock.) **nPERR** must be driven high for one clock before being tri-stated as with all sustained tri-state signals.
nSERR	o/d	*System Error* is for reporting address parity errors, data parity errors on the Special Cycle command, or any other system error where the result will be catastrophic. If an agent does not want a non-maskable interrupt (NMI) to be generated, a different reporting mechanism is required. **nSERR** is pure open-drain and is actively driven for a single PCI clock by the agent reporting the error. The assertion of **nSERR** is synchronous to the clock and meets the setup and hold times of all bused signals. However, the restoring of **nSERR** to the deasserted state is accomplished by a weak pull-up (same value as used for s/t/s) which is provided by the central resource not by the signaling agent. This pull-up may take two to three clock periods to fully restore **nSERR**. The agent that reports **nSERR** to the operating system does so anytime **nSERR** is asserted.
INTERRUPT PINS (OPTIONAL)		
nINT	o/d	*Interrupt* is used to request an interrupt.

Exactly how the IDSEL pin is driven is left to the discretion of the host/memory bridge or system designer. This signal has been designed to allow its connection to one of the upper 21 address lines, which are not otherwise used in a configuration access. However, there is no specified way of determining IDSEL from the upper 21 address bits. Therefore, the IDSEL pin must be supported by all targets. Devices must not make an internal connection between an AD line and an internal IDSEL signal in order to save a pin. The only exception is the host bridge, since it defines how IDSELs are mapped. IDSEL generation behind a PCI-to-PCI bridge is specified in the PCI-to-PCI Bridge Architecture Specification. How a system generates IDSEL is system specific; however, if no other mapping is required, the following example may be used. The IDSEL signal associated with Device Number 0 is connected to AD[16], IDSEL of Device Number 1 is connected to AD[17], and so forth until IDSEL of Device Number 15 is connected to AD[31]. For Device Number 17-31, the host bridge should execute the transaction but not assert any of the AD[31:16] lines but allow the access to be terminated with Master-Abort.

Twenty-one different devices can be uniquely selected for configuration accesses by connecting a different address line to each device and asserting one of the AD[31:11] lines at a time. The issue with connecting one of the upper 21 AD lines to IDSEL to the appropriate AD line. This does, however, create a very slow slew rate on IDSEL, causing it to be in an invalid logic state most of the time, as shown in Figure 18 on page 136 with “XXXX” marks.

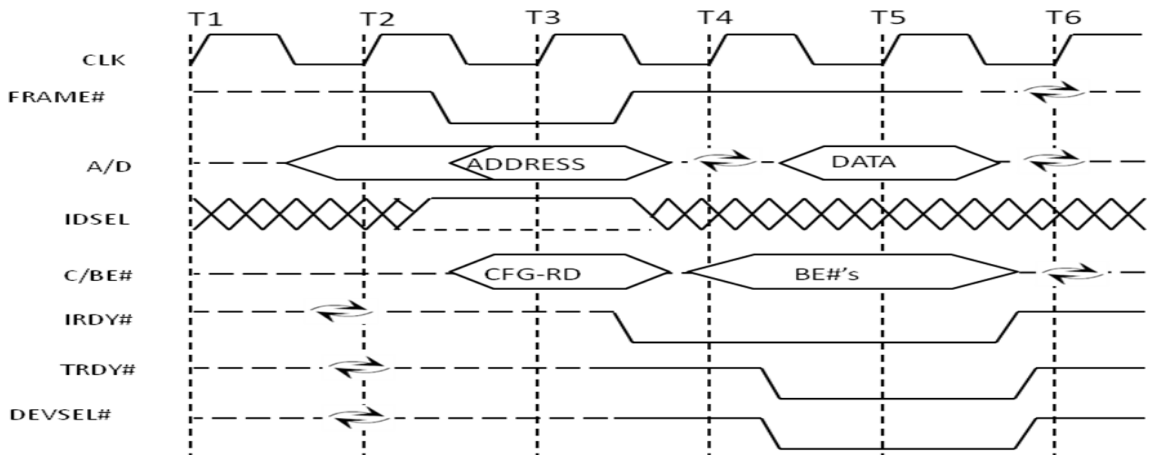


FIGURE 18. CONFIGURATION READ

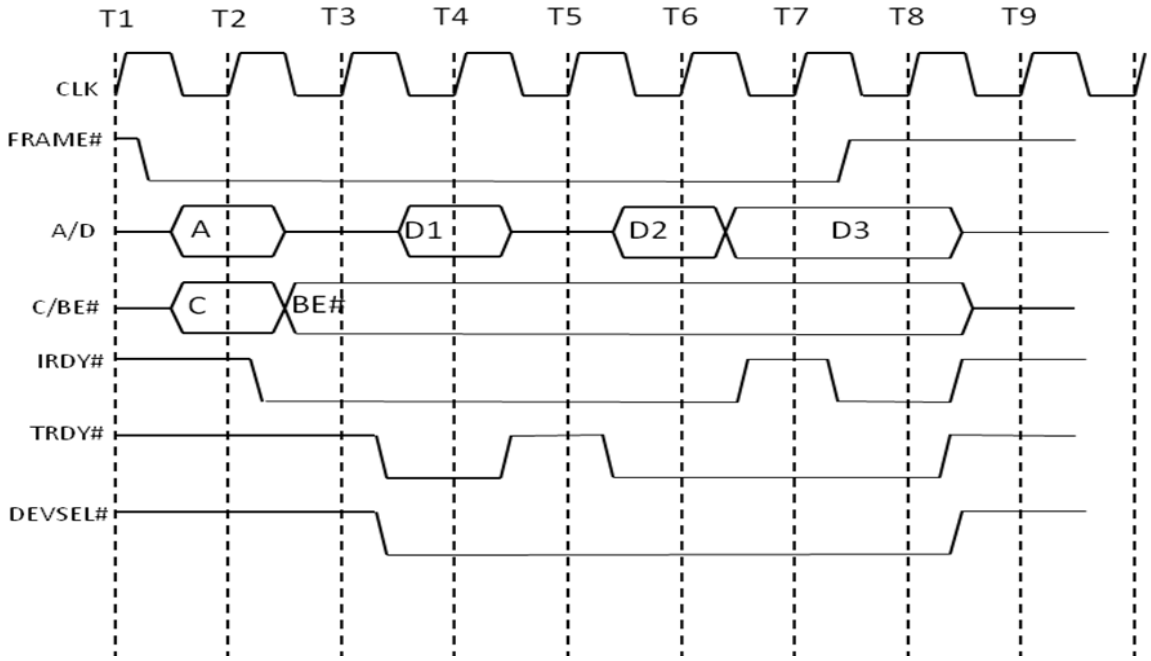


FIGURE 19. BASIC READ OPERATION

Figure 19 illustrates a read transaction and starts with an address phase which occurs when nFRAME is asserted for the first time and occurs on clock2. During the address phase, AD[31:0] contain a valid address and nC/BE[3:0] contain a valid bus command.

The first clock of the first data phase is clock3. During the data phase, nC/BE indicates which byte lanes are involved in the current data phase. A data phase may consist of wait cycles and a data transfer. The nC/BE output buffers must remain enabled (for both read and writes) from the first clock of the data phase through the end of the transaction. This ensures nC/BE are not left floating for long intervals. The nC/BE lines contain valid byte enable information during the entire data phase independent of the state of nIRDY. The nC/BE lines contain the byte enable information for data phase N+1 on the clock following the completion of the data phase N. This is not shown in Figure 19 because a burst read transaction typically has all byte enables asserted; however, it is shown in Figure 20. Notice on clock 5, the master inserted a wait state by deasserting nIRDY. However, the byte enables for data phase 3 are valid on clock 5 and remain valid until the data phase completes on clock 8.

The first data phase on a read transaction requires a turnaround-cycle (enforced by the target via nTRDY). In this case, the address is valid on clock2 and then the master stops driving AD. The earliest the target can provide valid data is clock 4. The target must drive the AD lines following the turnaround cycle when nDEVSEL is asserted. Once enabled, the output buffers must stay enabled through the end of the transaction. (This ensures that the AD lines are not left floating for long intervals.)

One way for a data phase to complete is when data is transferred, which occurs when both nIRDY and nTRDY are asserted on the same rising clock edge. There are other conditions that complete a data phase. (nTRDY cannot be driven until nDEVSEL is asserted.) When either nIRDY or nTRDY is deasserted, a wait cycle is inserted and no data is transferred. As noted in Figure 19, data is successfully transferred on clocks 4, 6, and 8 and wait cycles are inserted on clock 3, 5, and 7. The first data phase completes in the minimum time for a read transaction. The second data phase is extended on clock 5 because nTRDY is deasserted. The last data phase is extended because nIRDY was deasserted on clock 7.

The master knows at clock 7 that the next data phase is the last. However, because the master is not ready to complete the last transfer (nIRDY is deasserted on clock 7), nFRAME stays asserted. Only when nIRDY is asserted can nFRAME be deasserted as occurs on clock 8, indicating to the target that this is the last data phase of the transaction.

Figure 20 illustrates a write transaction. The transaction starts when nFRAME is asserted for the first time which occurs on clock 2. A write transaction is similar to a read transaction except no turnaround cycle is required following the address phase because the master provides both address and data. Data phases work the same for both read and write transactions.

In Figure 20, the first and second data phases complete with zero wait cycles. However, the third data phase has three wait cycles inserted by the target. Notice both agents insert a wait cycle on clock 5. nIRDY must be asserted when nFRAME is deasserted indicating the last data phase.

The date transfer was delayed by the master on clock 5 because nIRDY was deasserted. The last data phase is signaled by the master on clock 6, but it does not complete until clock 8.

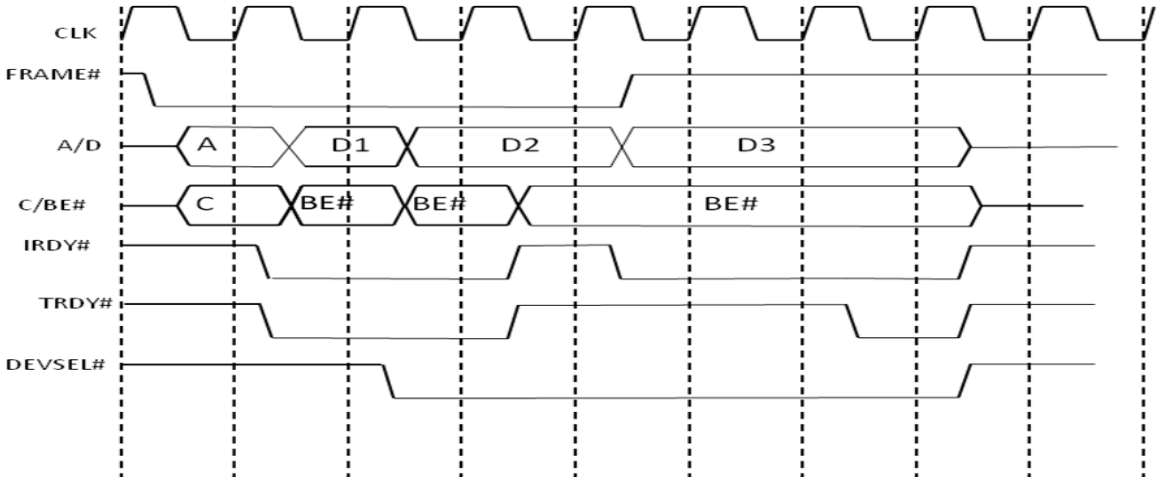


FIGURE 20. BASIC WRITE OPERATION

Table 9 shows PCI configuration registers, which is listed up with initial values. Device ID, Vendor ID, Subsys ID, Subsys Vendor ID, Class Code, Rev ID, Header Type, Max Lat, Min Gnt and Int Pin of PCI configuration registers can be updated by only PCI configuration EEPROM. Command, Status, Latency Timer, Cache Line Size, Memory Base Address, Interrupt Line, Retry Timeout and TRDY Timeout of the PCI configuration registers can be updated by PCI configuration write as shown in Figure 20 on page 137. In this case, PCI command needs to be changed as PCI configuration write instead of PCI configuration read. In addition, PCI configuration read is used to read PCI configuration registers for checking initial value and updated value.

TABLE 9. 8 PCI CONFIGURATION REGISTERS

31	16	15	0	OFFSET	INITIAL VALUE
Device ID		Vendor ID		00h	2819_1979h
Status		Command		04h	02a0_0000h
Class Code			Revision ID	08h	048000_00h
BIST	Header Type	Latency Timer	Cache line Size	0Ch	00_00_00_00h
Memory Base Address				10h	00_00000_8h
Reserved				14h	
Reserved				18h	
Reserved				1Ch	
Reserved				20h	
Reserved				24h	
Reserved				28h	
Subsystem ID		Subsystem Vendor ID		2Ch	2819_1719
Reserved				30h	
Reserved				34h	
Reserved				38h	
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3Ch	48_20_01_00h
Reserved		Retry Timeout	TRDY Timeout	40h	00_00_80_80h

TABLE 10. CONTENTS OF PCI CONFIGURATION EEPROM

ADDRESS	CONTENTS	
00h	Vendor ID	
04h	Device ID	
58h	Subsystem Vendor ID	
5Ch	Subsystem ID	
78h	Interrupt Pin	
7Ch	Max_Lat	Min_Gnt

UART Interface

The UART serial communication parameters comply with EIA-RS-232-C Interface Standard. The frame properties are specified in Figure 21. Every frame starts with the **start bit** (which is always **0**), followed by the least significant data bit (indicated with D0). Then the next data bits are succeeding, ending with the most significant bit (indicated with D7).

The parity bit is inserted after the data bits, before the **stop bit** (which is always **1**). When a complete frame is transmitted, it can be directly followed by a new frame, or the communication line can be set to an idle (high) state. Each **frame** corresponds to a **single byte of information**.

Each command or query can consist of one or several successive **RxD frames**. The lamp driver will answer accordingly with one or several successive **TxD frames**. Figure 22 shows the communication flow and defines the timing limit for receiving and transmitting **TMAX Command** and **TMAX Response**.

Following timing limits in the communication flow have to be kept:

- if after the time **TMAX Command = 15ms** a whole command is not transmitted completely to the lamp driver, the command queue is cleared (this is necessary to synchronize the command decoding algorithm with the projector electronics) and the error byte ABh is returned; the response will start earliest after having received the first frame of a command and latest after having received the last frame of a command plus a processing time, which can vary between zero and 5ms. The whole response time is guaranteed to be finished after **TMAX Response = 10ms**.

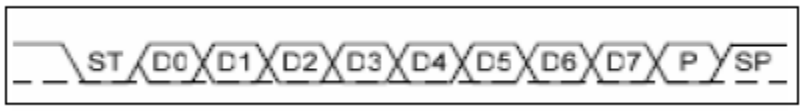


FIGURE 21. STRUCTURE (ST = START BIT, Dn = DATA BITS, P = PARITY BIT, SP = STOP BIT)

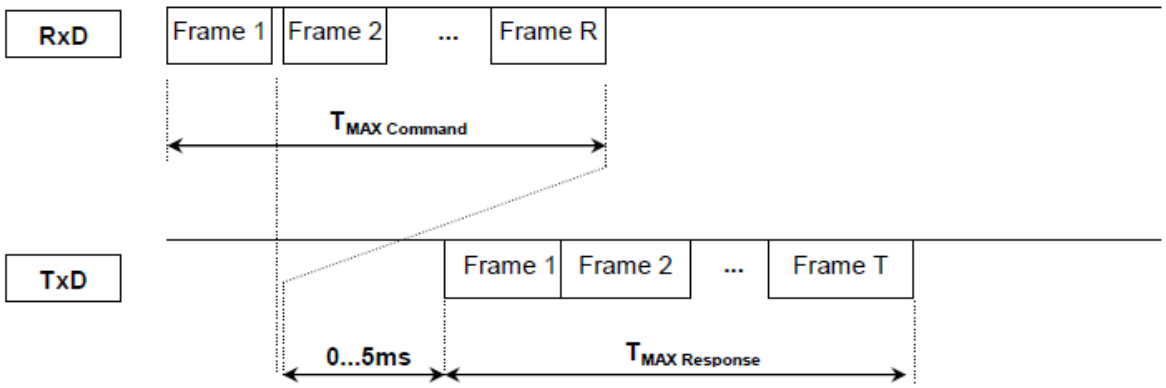


FIGURE 22. COMMUNICATION FLOW CHART

TABLE 11. 8 UART PARAMETER

FRAME PARAMETER	VALUE	TOLERANCE
Baud Rate	128000 (max.)	±1%
Number of Start Bits	1	N/A
Number of Data Bits	8	N/A
Number of Stop Bits	1/2	N/A
Parity Bit	Odd/Even/None	N/A

DDR2 Timing

TABLE 12. DDR2 TIMING TABLE

SPECIFICATION AME		SYMBOL	RELATED DRAM SPEC	VALUE [PS]	REMARKS (Note 4)
CMD-CK	Skew (maximum)	CMD_CLK_skew_max	tIS	680	1 (Figure 23)
	Skew (minimum)	CMD_CLK_skew_min	tIH	200	1 (Figure 23)
TxDQS-CK	Skew (maximum)	TxDQS-CK_skew_max	tDQSS(max), tDSS	280	2, 3 (Figures 23 and 24)
	Skew (minimum)	TxDQS-CK_skew_min	tDQSS(min), tDSH	170	2, 3 (Figures 23 and 24)
TxDQ-DQS	Skew (maximum)	TxDQ-DQS_skew_max	tDS	390	4 (Figure 25)
	Skew (minimum)	TxDQ-DQS_skew_min	tDH	380	4 (Figure 25)
RxDQ-DQS	Setup	RxDQ-DQS_setup	tDQSQ	-220	5 (Figure 26)
	Hold	RxDQ-DQS_hold	tQH	805	5 (Figure 26)
Round Trip Time	Delay (maximum)	RTT_delay_max	tDQSCK, tLZ	1000	6 (Figure 27)
	Delay (minimum)	RTT_delay_min	tDQSCK, tLZ	0	6 (Figure 27)

NOTE:

4. Refer to timing and circuit diagrams in Figures 23 through 27.

Timing and Circuit Diagrams

*1: CMD-CK timing and circuit diagram

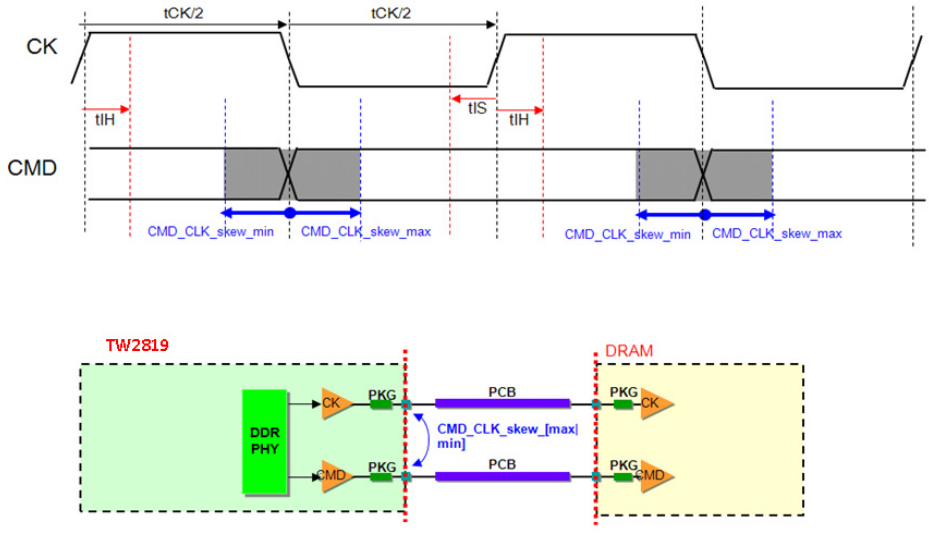


FIGURE 23. CMD-CK TIMING

Timing and Circuit Diagrams (Continued)

*2: TxDQS-CK timing and circuit diagram for tDQSS spec

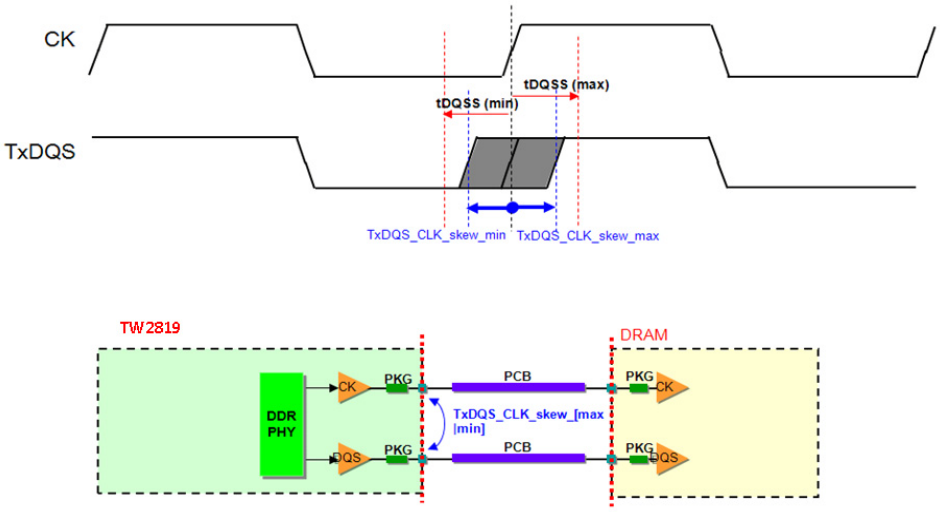


FIGURE 24. TXDQS-CK TIMNG FOR TDQSS

*3: TxDQS-CK timing for tDSS/tDSH. The circuitry is the same as the one for tDQSS.

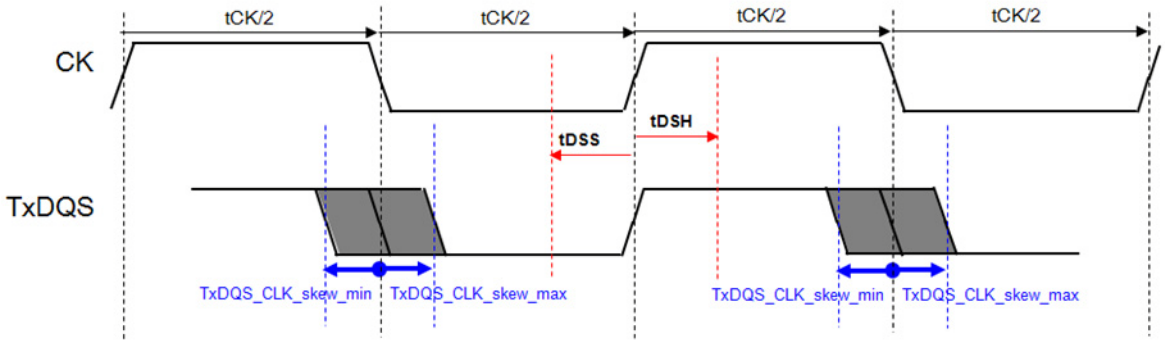


FIGURE 25. TXDQS-CK TIMING FOR TDSS/TDSH

Timing and Circuit Diagrams (Continued)

*4: TxDQ-DQS timing and circuit diagram

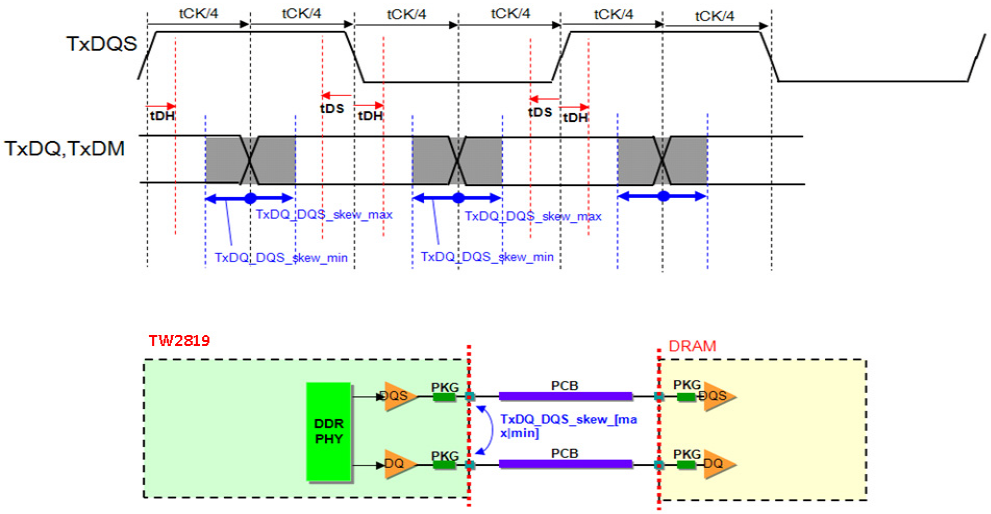


FIGURE 26. TXDQ-DQS TIMING

*5: RxDQ-DQS timing and circuit diagram. Please note that rightward arrow of **RxDQ_DQS_setup** is caused by DLL inside DDR PHY which shifts RxDQS by approximately 90 degree independently of PTV variation. Please make sure that your transmission line skew plus tDQSQ is less RxDQ_DQS_setup number.

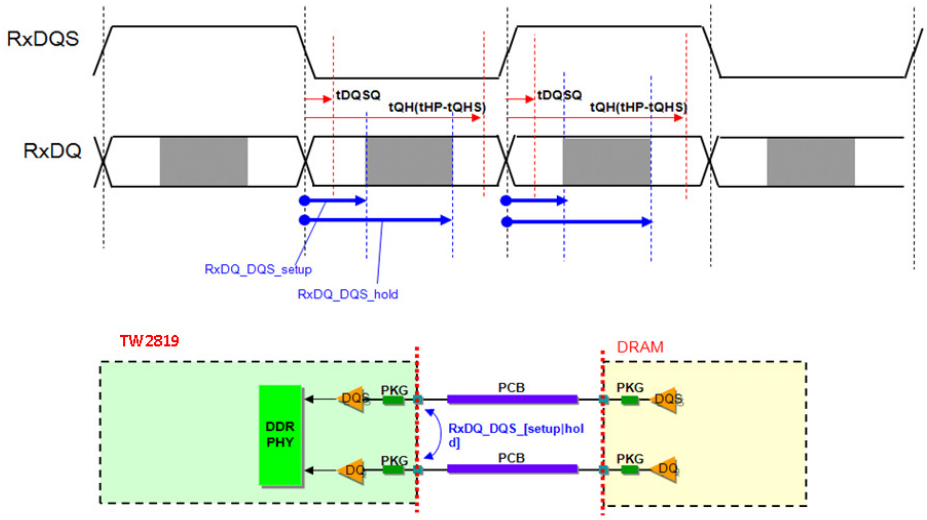


FIGURE 27. RXDQ-DQS TIMING

Timing and Circuit Diagrams (Continued)

*6: Round trip path timing and circuit diagram. Please note that the round trip time must contain worst value of tDQSCK or tLZ Specification, that is, minimum number for "minimum round trip time" case, and maximum number for "maximum round trip time" case.

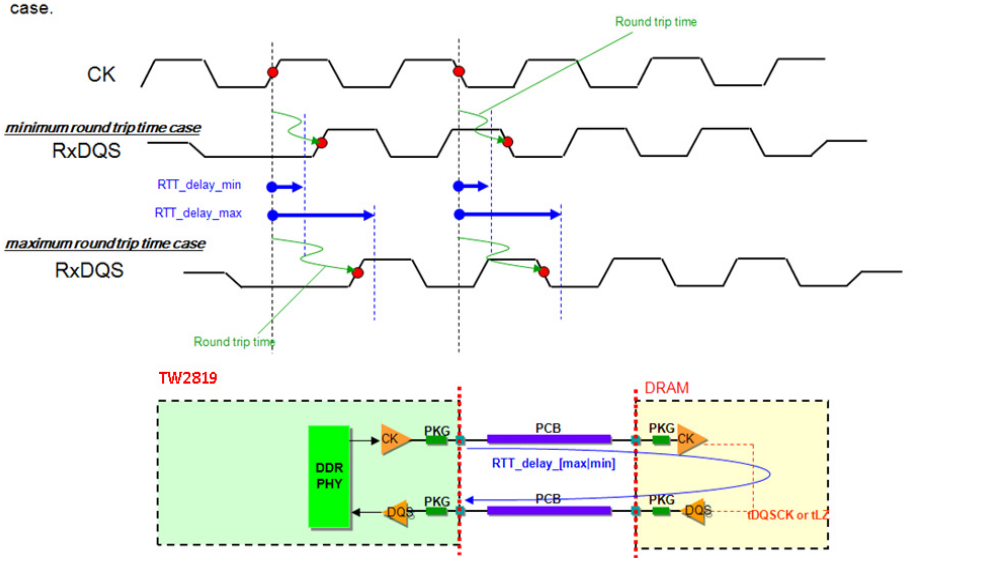


FIGURE 28. ROUND TRIP PATH TIMING

Power-up Sequence

The TW2819 must follow certain power-up sequences to guarantee it is working. This is shown in Figure 29.

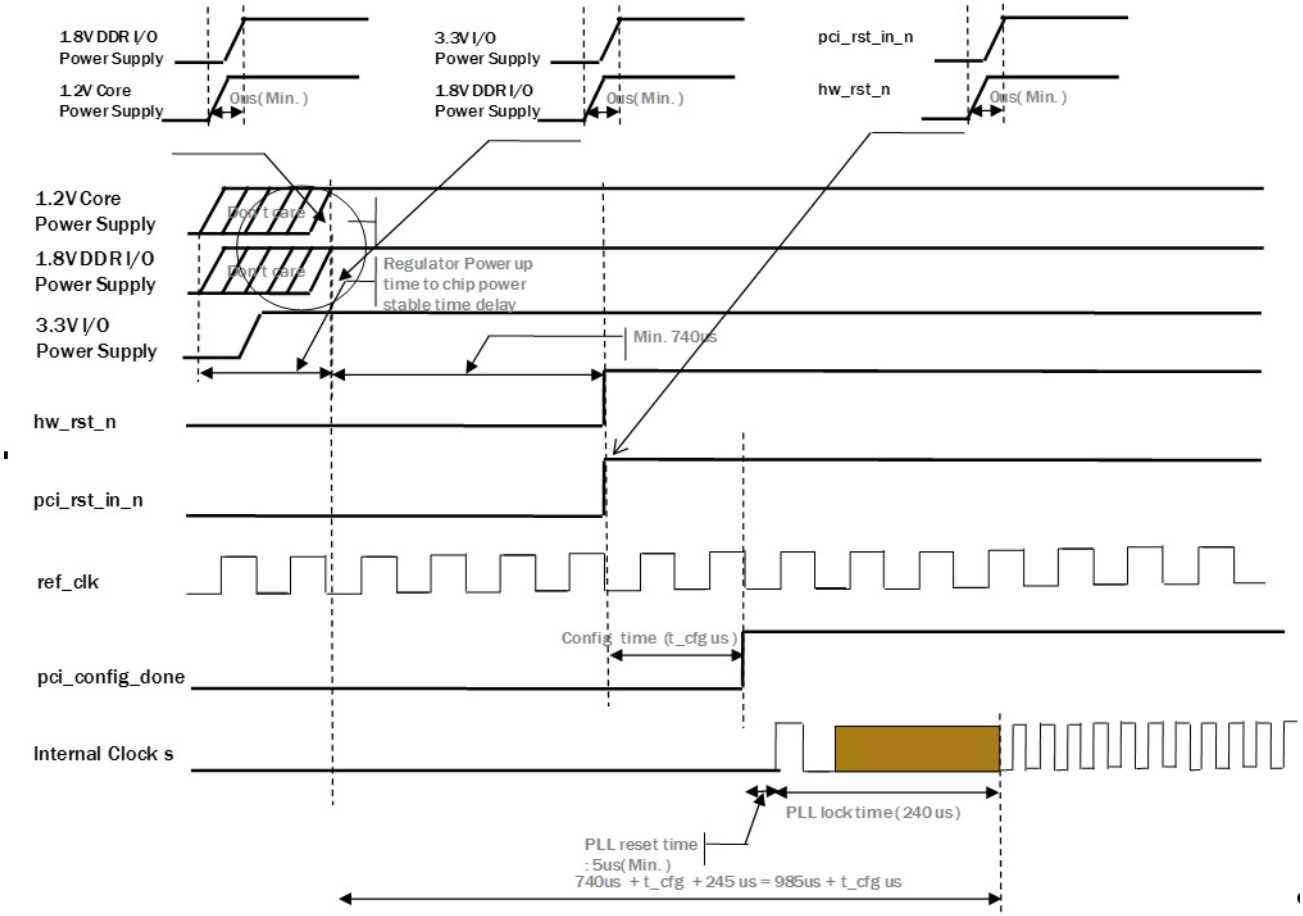


FIGURE 29. TW2819 POWER-UP SEQUENCE

Power-off Sequence

The TW2819 must follow certain power off sequences to guarantee it is working as shown in Figure 30.

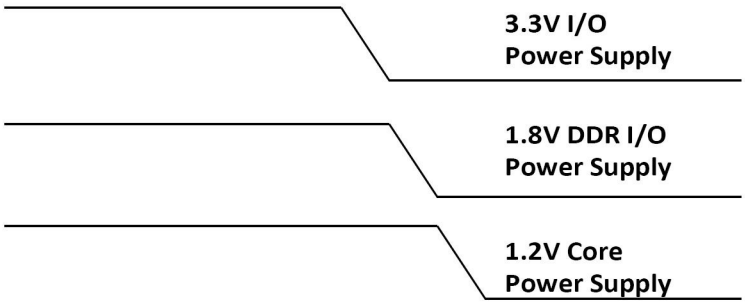


FIGURE 30. POWER OFF SEQUENCE

Revision History

The revision history provided is for informational purposes only and is believed to be accurate, but not warranted. Please go to the web to make sure that you have the latest revision.

DATE	REVISION	CHANGE
July 31, 2015	FN8404.1	Datasheet modified to Intersil standards and template. Modified feature list on page 1. Added Applications on page 1 Added Typical Application Diagram on page 1 Updated Block Diagram on page 22 Modified DDR clock frequency. Removed MUX display. Removed registers that are not open to PCI host. Removed I2C as it should only be used for debugging purposes.
June 21, 2013	FN8404.0	Initial Release

About Intersil

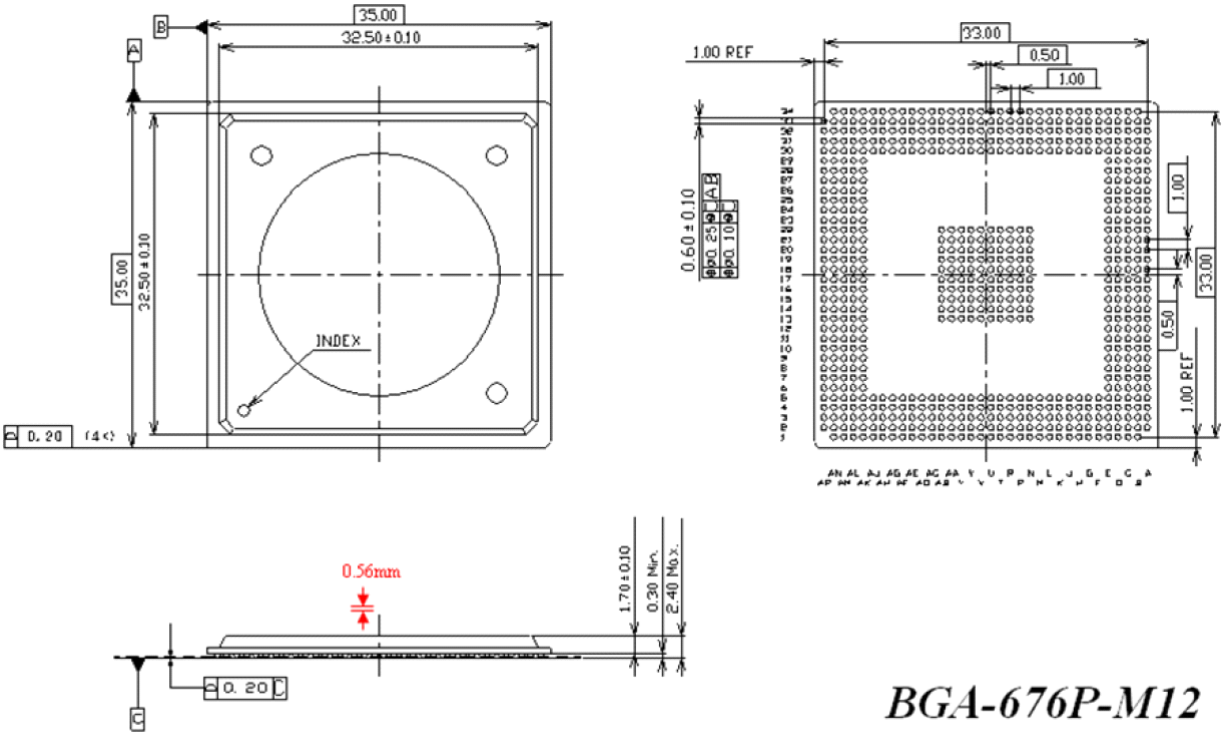
Intersil Corporation is a leading provider of innovative power management and precision analog solutions. The company's products address some of the largest markets within the industrial and infrastructure, mobile computing and high-end consumer markets.

For the most updated datasheet, application notes, related documentation and related parts, please see the respective product information page found at www.intersil.com.

You may report errors or suggestions for improving this datasheet by visiting www.intersil.com/ask.

Reliability reports are also available from our website at www.intersil.com/support

Mechanical Specifications



BGA-676P-M12

Dimensions in millimeters