TEL: 408-329-5540
FAX: 408-329-5541
Email: sales@paradetech.com
Website: www.paradetech.com

Application Note

Version 0.4

Dec 30, 2019

Notice

All information provided in this document on “AS IS” basis without any guarantee or warranty. Information in this document is provided in relation to Parade products and is subject to change without notice. No intellectual rights or licenses are implied.

Contents

Revision History

Version 0.1	06/24/2019	Initial draft
Version 0.2	07/03/2019	update sample code for I2C master0 and SPI master0
Version 0.2a	07/09/2019	add SPI_CFG 1and SPI_INT_STATUS register in page 7
Version 0.3	09/19/2019	update CDI bert sequence and CDI PLL programming
Version 0.4	12/30/2019	update CDI PLL sequence ,add I2C master3 and SPI master1/2 register

1. Chip ID and Configuration ID

1.1 CHIP ID

Register name	Offset	Initial Value	Read Write	Description
CHIP_ID[7:0]	0x0ffc	0x00	R/W	Chip Revision, metal change revision.
CHIP_ID[15:8]	0x0ffd	0x0A	R/W	Chip Revision, full layer revision.
CHIP_ID[23:16]	0x0ffe	0x55	R/W	Chip ID “55”
CHIP_ID[31:24]	0x0fff	0x08	R/W	Chip ID “8”
UNIQ_CHIP_ID[7:0]	0x10bc	0x00	R/W	Programmed by parade
UNIQ_CHIP_ID[15:8]	0x10bd	0x00	R/W	
UNIQ_CHIP_ID[23:16]	0x10be	0x00	R/W	
UNIQ_CHIP_ID[31:24]	0x10bf	0x00	R/W	
UNIQ_CHIP_ID[39:32]	0x10c0	0x00	R/W	
UNIQ_CHIP_ID[47:40]	0x10c1	0x00	R/W	
UNIQ_CHIP_ID[55:48]	0x10c2	0x00	R/W	
UNIQ_CHIP_ID[63:56]	0x10c3	0x00	R/W	

1.2 HWCID

Register name	Offset	Initial Value	Read Write	Description
REG_PHY_OFFF8	0x0ff8	0x00	RO	[3:0] HWCID It is latched by 4 strapping pins

1.3 OTP Configuration ID

Register name	Offset	Initial Value	Read Write	Description
CFG_ID[7:0]	0x10C4	0x00	R/W	bit[7:0]:CFG7-CFG0
CFG_ID[15:8]	0x10C5	0x00	R/W	bit8: HW_LOAD_DIS bit9: PROD_STATUS bit15:10: Reserved (0x00)
CFG_ID[23:16]	0x10C6	0x00	R/W	bit23:16:Reserved(set as 0xff)

2. I2C Slave

2.1 Read Access

i2cslv_id=0x10
addr_h=Page number
addr_l=0ffest
Read855reg(i2cslv_id, addr_h, addr_l)

2.2 Write Access

i2cslv_id=0x10
addr_h=Page number
addr_l=0ffest
Write855reg(i2cslv_id, addr_h, addr_l,write_data)

3. SPI Slave

3.1 Read Access

SlaveAddr=0x00
SPIReadReg(slaveAddr,offset)

3.2 Write Access

slaveAddr=0x00
SPIWriteReg(slaveAddr, offset, data)

Register name	Offset	Initial Value	Read Write	Description
SPI_CFG0	0x0b00	0x2c	R/W	[5]:HIZ [4]:MULTI_IO_BIT_ODER [3]:RX_BIT_ORDER [2]:TX_BIT_ORDER [1]:CPHA [0]:CPOL
SPI_CFG1	0x0b04	0x2f	R/W	[7:0]Read_start
SPI_CFG2	0x0b08	0x00	R/W	[3:0]Device_addr
SPI_INT_STATUS	0x0b0c	0x00	RW1C	[7]buf_underflow [6]buf_overflow [5]CS_ERR [4]CRC_ERR [3:2]reserved [1]end_of_transaction [0]start_of_transaction
SPI_IN_EN	0x0b10	0x00	R/W	[7]buf_underflow_en_sys [6]buf_overflow_en_sys [5]cs_err_en_sys [4]crc_err_en_sys [3:2]reserved [1] end_of_transaction_en_sys [0] start_of_transaction_en_sys

4. I2C Master

4.1 I2C Slave/SPI Slave/AUX read/write to I2C Master sequence

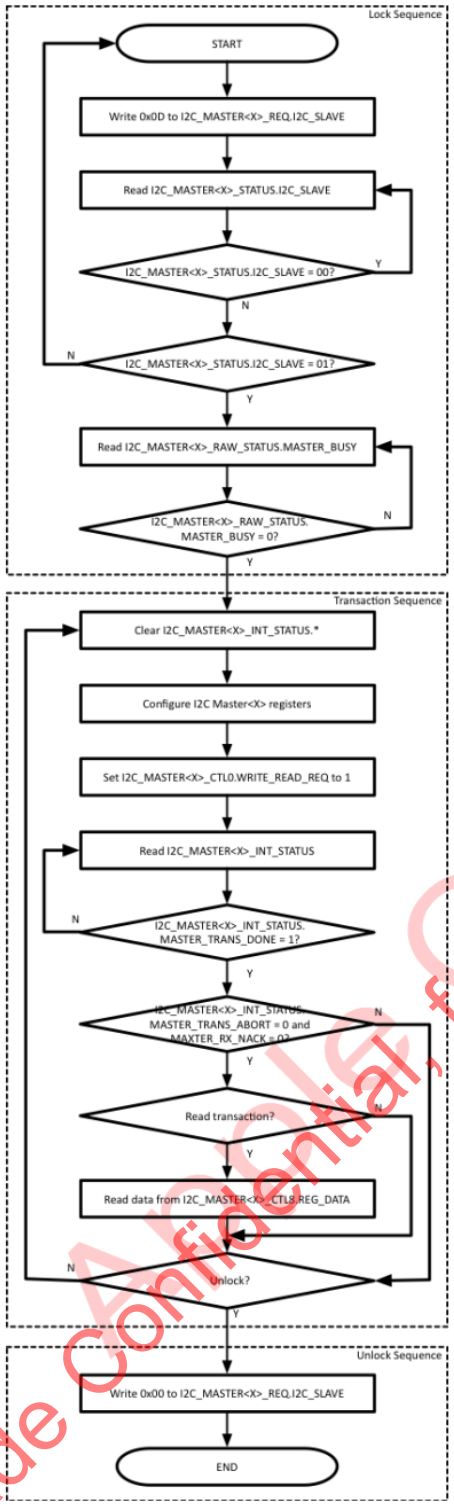


Figure 18-7: Flow for I2C Masters - I2C Slave, SPI Slave, AUX CH and MCU FW - MAX_BUS_HOLD_TIME timer is disabled

4.2 Register definition

Register name	Offset	Initial	Read	Description

		Value	Write	
MI2C0_REQ_PART0[7:0]	0x3710	0x00	R/W	I2C_MASTER_REQ0 //MCU
MI2C0_REQ_PART0[15:8]	0x3711	0x00	R/W	I2C_MASTER_REQ1 //Master Data Loader
MI2C0_REQ_PART0[23:16]	0x3712	0x00	R/W	I2C_MASTER_REQ2 //0x0c=SPI slave request lock to i2c master,0x00= clear LOCK on FIFO
MI2C0_REQ_PART0[31:24]	0x3713	0x00	R/W	I2C_MASTER_REQ3 /0x0d=I2C slave request lock to i2c master,0x00= clear LOCK on FIFO
MI2C0_REQ_PART1	0x3714	0x00	R/W	I2C_MASTER_REQ4 //0x0b=aux request lock to i2c master,0x00= clear LOCK on FIFO
MI2C0_STATUS_PART0[7:0]	0x3718	0x00	RO	I2C_MASTER_STATUS0 //MCU
MI2C0_STATUS_PART0[15:8]	0x3719	0x00	RO	I2C_MASTER_STATUS1 //Master Data Loader

MI2C0_STATUS_PART0[23:16]	0x371a	0x00	RO	I2C_MASTER_STATUS2
				//SPI slave
				0x00= I2C buffer is busy
				0x01 = I2C buffer is available
				0x10 = Reserved
				0x11 = IDLE or MAX_BUS_HOLD_TIME timeout

MI2C0_STATUS_PART0[31:24']	0x371b	0x00	RO	I2C_MASTER_STATUS3
				//I2C slave
				0x00= I2C buffer is busy
				0x01 = I2C buffer is available
				0x10 = Reserved
				0x11 = IDLE or MAX_BUS_HOLD_TIME timeout

MI2C0_STATUS_PART1	0x371c	0x00	RO	I2C_MASTER_STATUS4
				//AUX
				0x00= I2C buffer is busy
				0x01 = I2C buffer is available
				0x10 = Reserved
				0x11 = IDLE or MAX_BUS_HOLD_TIME timeout

MI2C0_CTL0[7:0]	0x3004	0x00	R/W	[0] WRITE_REQ_SEL 1:read, 0:write [2:1] ADDR_LENGTH [7:3] WRITE_LENGTH[4:0]
MI2C0_CTL0[15:8]	0x3005	0x00	R/W	[7:5]READ_LENGTH[2:0] [4:0] WRITE_LENGTH[9:5]
MI2C0_CTL0[23:16]	0x3006	0x00	R/W	[6:0]]READ_LENGTH[9:3]
MI2C0_CTL1[7:0]	0x3008	0x00	R/W	REG_NVM_ADDR[7:0]
MI2C0_CTL1[15:8]	0x3009	0x00	R/W	REG_NVM_ADDR[15:8]
MI2C0_CTL1[23:16]	0x300a	0x00	R/W	REG_NVM_ADDR[23:16]
MI2C0_CTL2	0x300c	0x00	R/W	[1]CLEAR_BUFFER [0]WRITE_READ_REQ
MI2C0_STATUS	0x3010	0x00	RO	[2]MASTER_BUSY
MI2C0_INT_STATUS	0x3100	0x00	RW1C	[2]MASTER_TRANS_ABORT [1]MASTER_RX_NACK [0]MASTER_TRANS_DONE
MI2C0_REG_DATA	0x3200 ~ 0x35fc	0x00	VR/W	I2C Master buffer(size:256x32)
MI2C3_CTL0	0x3804	0x04	R/W	[7:3] wr_len[4:0] [2:1]ADDR_LENGTH [0]WRITE_REQ_SEL
MI2C3_CTL0	0x3805	0x00	R/W	[4:0]wr_len[9:5] [7:5] READ_LENGTH[2:0]
MI2C3_CTL0	0x3806	0x00	R/W	[6:0] READ_LENGTH[9:3]
MI2C3_CTL1	0x3808	0x00	R/W	[7:0]SLAVE_ADDR //i2c device address
MI2C3_CTL1	0x3809	0x00	R/W	[7:0]REG_ADDR[7:0] //offset address
MI2C3_CTL1	0x380a	0x00	R/W	[7:0]REG_ADDR[15:8] //offset address
MI2C3_CTL2	0x380c	0x00	R/W	[1]CLEAR_BUFFER [0]WRITE_READ_REQ
MI2C3_STATUS	0x3810	0x00	R/W	[2]MASTER_BUSY [1]MI2C_AL [0]MI2C_BUSY
MI2C3_INT_STATUS	0x3900	0x00	R/W	[2]MASTER_TRANS_ABORT [1]MASTER_RX_NACK

				[0]MASTER_TRANS_DONE
MI2C3_REG_WDATA0[31:0]	0x3a00-0x3a03	0x00	R/W	MI2C3_REG_WDATA0[31:0]
MI2C3_REG_WDATA1[31:0]	0x3a04-0x3a07	0x00	R/W	MI2C3_REG_WDATA1[31:0]
MI2C3_REG_WDATA2[31:0]	0x3a08-0x3a0a	0x00	R/W	MI2C3_REG_WDATA2[31:0]
MI2C3_REG_WDATA3[31:0]	0x3a0-0x3a0f	0x00	R/W	MI2C3_REG_WDATA3[31:0]
MI2C3_REG_RDATA0 [31:0]	0x3a10-0x3a13	0x00	R/W	MI2C3_REG_RDATA0 [31:0]
MI2C3_REG_RDATA1 [31:0]	0x3a14-0x3a17	0x00	R/W	MI2C3_REG_RDATA1 [31:0]
MI2C3_REG_RDATA2 [31:0]	0x3a18-0x3a1b	0x00	R/W	MI2C3_REG_RDATA2 [31:0]
MI2C3_REG_RDATA3[31:0]	0x3a1c-0x3a1f	0x00	R/W	MI2C3_REG_RDATA3[31:0]
MI2C3_REQ_PART0	0x3b93	0x00	R/W	[31:24]I2C_MASTER_REQ3 //I2C slave
MI2C3_STATUS_PART0	0x3b9b	0x00	R/W	[31:24]I2C_MASTER_STATUS3 //I2C slave

4.3 *Sample code for I2C master0 write by I2C slave*

```
def mi2c0_wr_offset(offset):
    WriteReg(0x10, 0x30, 0x08, offset & 0xff)
    WriteReg(0x10, 0x30, 0x09, (offset>>8) & 0xff)
    WriteReg(0x10, 0x30, 0x0a, (offset>>16) & 0xff)

#addr_len 01:offset1byte;10:offset2bytes     wr_rd_sel 0:write;1:read
def mi2c0_wr_param(rd_len, wr_len, addr_len, wr_rd_sel):
```

```
    para_0 = ((wr_len<<3) | (addr_len<<1) | wr_rd_sel) & 0xff
    para_1 = ((rd_len<<5) | (wr_len>>5)) & 0xff
    para_2 = (rd_len>>3) & 0xff
    WriteReg(0x10, 0x30, 0x04, para_0)
    WriteReg(0x10, 0x30, 0x05, para_1)
    WriteReg(0x10, 0x30, 0x06, para_2)

##**********write sequence**************
req_id=0x0d
req_addr=0x13
lock_status_addr=0x1b
WriteReg(0x10, 0x37, req_addr, req_id)                    #lock req
while(1):
    LockStatus=ReadReg(0x10, 0x37, lock_status_addr)          #read lock status
    LockStatus &= 0x03
    print 'LockStatus:0x%02x' %LockStatus
    if LockStatus==1:
        break
while(1):
    BusyStatus=ReadReg(0x10, 0x30, 0x10)       #read i2c master busy status
    BusyStatus &= 0x04
    print 'BusyStatus:0x%02x' %BusyStatus
    if BusyStatus==0:
        break
WriteReg(0x10, 0x31, 0x00, 0x07)                  #clear i2c master interupt status
##****config parameter*****
offset=0
rd_len=0
wr_len=0xf #max of write_length is eeprom page size-1
addr_len=2
wr_rd_sel=0
mi2c0_wr_offset(offset)
mi2c0_wr_param(rd_len, wr_len, addr_len, wr_rd_sel)
##write data to the buffer in i2c master
for i in range(wr_len+1):
    data=i+1
    WriteReg(0x10, 0x32+(i>>8), i & 0xff, data)
WriteReg(0x10, 0x30, 0x0c, 0x01)                  #start
while(1):
    I2C_Master_Status=ReadReg(0x10, 0x31, 0x00) #read trans_done status
    Trans_Done_Status=I2C_Master_Status & 0x01
    if Trans_Done_Status==1:
```

```
            break
WriteReg(0x10, 0x37, req_addr, 0x00)

##function mi2c0_wr_data(f) is used to write data in .txt file to eeprom
def mi2c0_wr_data(f):
    file_object = file(f)
    lines = file_object.readlines()
    page_size = 64      #unit:byte
    i = 0               #line number
    j = 0               #page number
    for line in lines:
        line_temp = line.strip()
        data = int(line_temp, 16)
        a=page_size*j
        if i==a:
            j = j+1
            WriteReg(0x10, 0x37, 0x13, 0xd)#lock req
            while(1):
                LockStatus=ReadReg(0x10, 0x37, 0x1b)
                LockStatus &= 0x3
                print 'LockStatus:0x%02x' %LockStatus
                if LockStatus==1:
                    break
            offset=page_size*(j-1)
            mi2c0_wr_offset(offset)
            mi2c0_wr_param(0x0, page_size-1, 0x2, 0x0)
        addr=i-(j-1)*page_size
        WriteReg(0x10, 0x32, addr, data)
        b=page_size*j-1
        if i==b:
            WriteReg(0x10, 0x30, 0x0c, 0x1)#start
            while(1):
                BusyStatus=ReadReg(0x10, 0x30, 0x10)
                BusyStatus &= 0x04
                print 'BusyStatus:0x%02x' %BusyStatus
                if BusyStatus==0:
                    break
            WriteReg(0x10, 0x37, 0x13, 0x0)#unlock
        i = i+1
```

4.4 Sample code for I2C master0 read by I2C slave

```
def mi2c0_wr_offset(offset):
    WriteReg(0x10, 0x30, 0x08, offset & 0xff)
    WriteReg(0x10, 0x30, 0x09, (offset>>8) & 0xff)
    WriteReg(0x10, 0x30, 0x0a, (offset>>16) & 0xff)

#addr_len 01:offset1byte;10:offset2bytes    wr_rd_sel 0:write;1:read
def mi2c0_wr_param(rd_len, wr_len, addr_len, wr_rd_sel):
    para_0 = ((wr_len<<3) | (addr_len<<1) | wr_rd_sel) & 0xff
    para_1 = ((rd_len<<5) | (wr_len>>5)) & 0xff
    para_2 = (rd_len>>3) & 0xff
    WriteReg(0x10, 0x30, 0x04, para_0)
    WriteReg(0x10, 0x30, 0x05, para_1)
    WriteReg(0x10, 0x30, 0x06, para_2)
##**********read sequence**************
req_id=0x0d
req_addr=0x13
lock_status_addr=0x1b
WriteReg(0x10, 0x37, req_addr, req_id)                    #lock req
while(1):
    LockStatus=ReadReg(0x10, 0x37, lock_status_addr)          #read lock status
    LockStatus &= 0x03
    print 'LockStatus:0x%02x' %LockStatus
    if LockStatus==1:
        break
while(1):
    BusyStatus=ReadReg(0x10, 0x30, 0x10)          #read i2c master busy status
    BusyStatus &= 0x04
    print 'BusyStatus:0x%02x' %BusyStatus
    if BusyStatus==0:
        break
WriteReg(0x10, 0x31, 0x00, 0x7)                   #clear i2c master interupt status
##****config parameter*****
offset=0
rd_len=0xf #max of read_length is 0x3ff
wr_len=0
addr_len=2
wr_rd_sel=1
mi2c0_wr_offset(offset)
mi2c0_wr_param(rd_len, wr_len, addr_len, wr_rd_sel)
WriteReg(0x10, 0x30, 0x0c, 0x1)                   #start
```

```
while(1):
    I2C_Master_Status=ReadReg(0x10, 0x31, 0x00) #read trans_done status
    Trans_Done_Status=I2C_Master_Status & 0x01
    if Trans_Done_Status==1:
        break
for i in range(rd_len+1):
    data = ReadReg(0x10, 0x32+(i>>8), i & 0xff)
    print 'Readdata:0x%02x' %data
WriteReg(0x10, 0x37, req_addr, 0x0)                    #unlock
```

4.5 *Sample code for I2C master3 read/write by I2C slave*

```
def mi2c3_wr_addr(offset, slave_id):
    WriteReg(0x10, 0x38, 0x08, slave_id & 0xff)
    WriteReg(0x10, 0x38, 0x09, offset & 0xff)
    WriteReg(0x10, 0x38, 0x0a, (offset>>8) & 0xff)

#addr_len 01:offset1byte;10:offset2bytes    wr_rd_sel 0:write;1:read
def mi2c3_wr_param(rd_len, wr_len, addr_len, wr_rd_sel):
    para_0 = ((wr_len<<3) | (addr_len<<1) | wr_rd_sel) & 0xff
    para_1 = ((rd_len<<5) | (wr_len>>5)) & 0xff
    para_2 = (rd_len>>3) & 0xff
    WriteReg(0x10, 0x38, 0x04, para_0)
    WriteReg(0x10, 0x38, 0x05, para_1)
    WriteReg(0x10, 0x38, 0x06, para_2)


##**********write sequence**************
req_id=0x0d
req_addr=0x93
lock_status_addr=0x9b
WriteReg(0x10, 0x3b, req_addr, req_id)                 #lock req
while(1):
    LockStatus=ReadReg(0x10, 0x3b, lock_status_addr)        #read lock status
    LockStatus &= 0x03
    print 'LockStatus:0x%02x' %LockStatus
    if LockStatus==1:
        break
while(1):
    BusyStatus=ReadReg(0x10, 0x38, 0x10)        #read i2c master busy status
    BusyStatus &= 0x04
    print 'BusyStatus:0x%02x' %BusyStatus
```

```
    if BusyStatus==0:
        break
WriteReg(0x10, 0x39, 0x00, 0x7)                    #clear i2c master interrupt status
##****config parameter*****
slave_id=0xa0
offset=0
rd_len=0
wr_len=0xf
addr_len=2
wr_rd_sel=0
mi2c3_wr_addr(offset, slave_id)
mi2c3_wr_param(rd_len, wr_len, addr_len, wr_rd_sel)
##write data to the buffer in i2c master
for i in range(wr_len+1):
    data=i+1
    WriteReg(0x10, 0x3a, i & 0xff, data)          #write data buff address
0x3a00~0x3a0f
    print 'Writedata:0x%02x' %data
WriteReg(0x10, 0x38, 0x0c, 0x1)                    #start
while(1):
    I2C_Master_Status=ReadReg(0x10, 0x39, 0x00) #read trans_done status
    Trans_Done_Status=I2C_Master_Status & 0x01
    if Trans_Done_Status==1:
        break
WriteReg(0x10, 0x3b, req_addr, 0x0)                #unlock


##**********read sequence**************
req_id=0x0d
req_addr=0x93
lock_status_addr=0x9b
WriteReg(0x10, 0x3b, req_addr, req_id)             #lock req
while(1):
    LockStatus=ReadReg(0x10, 0x3b, lock_status_addr)      #read lock status
    LockStatus &= 0x03
    print 'LockStatus:0x%02x' %LockStatus
    if LockStatus==1:
        break
while(1):
    BusyStatus=ReadReg(0x10, 0x38, 0x10)       #read i2c master busy status
    BusyStatus &= 0x04
    print 'BusyStatus:0x%02x' %BusyStatus
```

```
    if BusyStatus==0:
        break
WriteReg(0x10, 0x39, 0x00, 0x7)                    #clear i2c master interrupt status
##****config parameter*****
slave_id=0xa0
offset=0
rd_len=0xf
wr_len=0
addr_len=2
wr_rd_sel=1
mi2c3_wr_addr(offset, slave_id)
mi2c3_wr_param(rd_len, wr_len, addr_len, wr_rd_sel)
WriteReg(0x10, 0x38, 0x0c, 0x1)                    #start
while(1):
    I2C_Master_Status=ReadReg(0x10, 0x39, 0x00) #read trans_done status
    Trans_Done_Status=I2C_Master_Status & 0x01
    if Trans_Done_Status==1:
        break
for i in range(rd_len+1):
    data = ReadReg(0x10, 0x3a, (0x10+i) & 0xff) #read data buff address
0x3a10~0x3a1f
    print 'Readdata:0x%02x' %data
WriteReg(0x10, 0x3b, req_addr, 0x0)                #unlock
```

5. SPI Master0

5.1 I2C Slave/SPI Slave/AUX CH read/write to SPI Master0 sequence

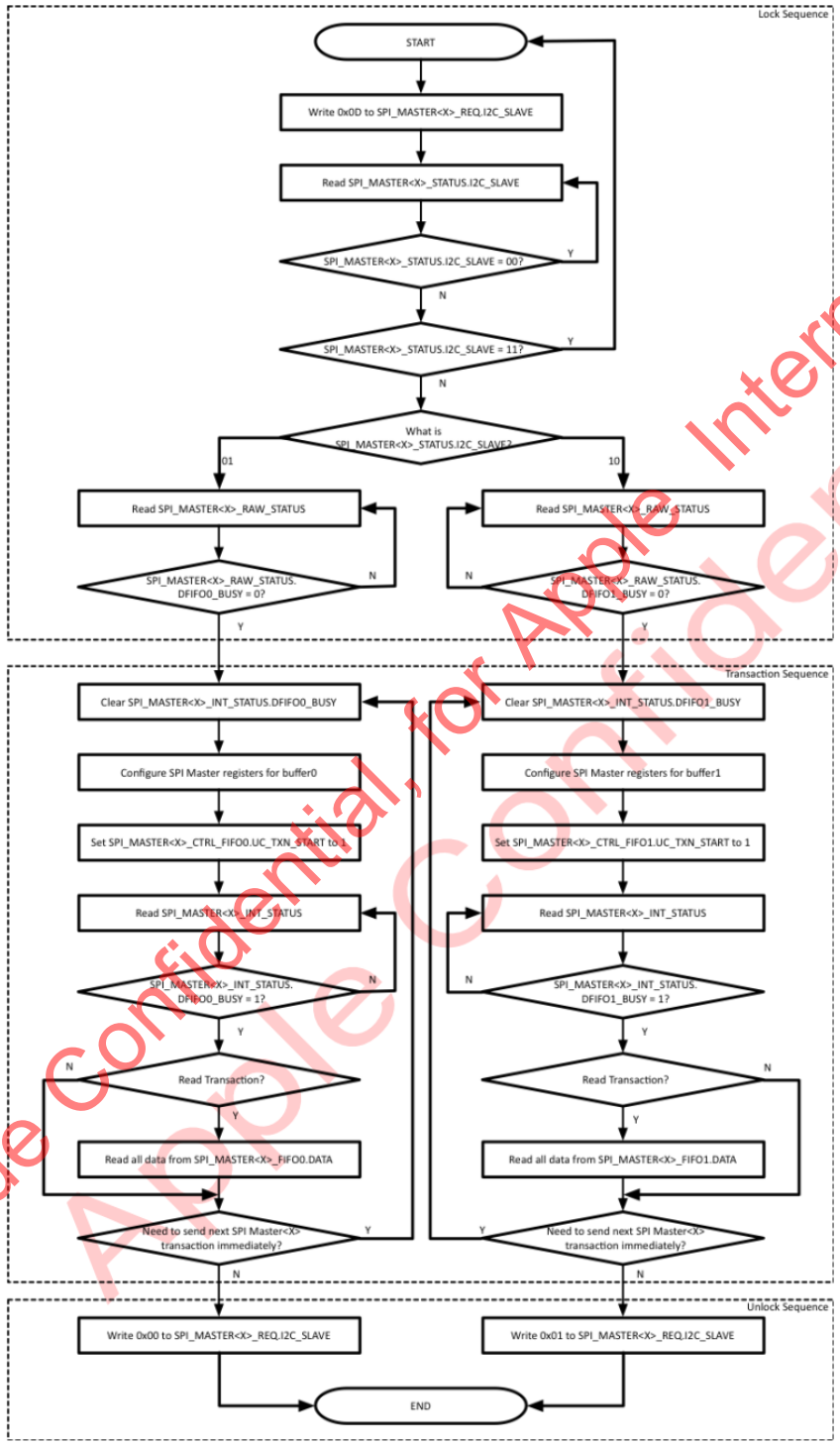


Figure 18-4: Flow for SPI Masters - I2C Slave, SPI Slave, AUX CH and MCU FW - MAX_BUS_HOLD_TIME timer is disabled

5.2 Register definition

Register name	Offset	Initial Value	Read Write	Description
SPI0_MASTER1_REQ[7:0]	0x3c04	0x00	R/W	[7:0]AHB_MASTER4_REQ // 0x0c:Request LOCK to SPI master for SPI slave 0x00 :Clear LOCK on FIFO0 0x01 :Clear LOCK on FIFO1
SPI0_MASTER1_REQ[15:8]	0x3c05	0x00	R/W	[15:8]AHB_MASTER5_REQ //0x0d:Request LOCK to SPI master for I2C slave 0x00 :Clear LOCK on FIFO0 0x01 :Clear LOCK on FIFO1
SPI0_MASTER1_REQ[23:16]	0x3c06	0x00	R/W	[15:8]AHB_MASTER6REQ //0x0b:Request LOCK to SPI master for Aux ch 0x00 :Clear LOCK on FIFO0 0x01 :Clear LOCK on FIFO1
SPI0_MASTER1_STATUS[7:0]	0x3c0c	0x00	RO	[7:0]SPI_MASTER4_STATUS // SPI Slave 0x00:Both buffer0 and buffer1 are busy 0x01:SPI buffer0 is available *0*x10:SPI buffer1 is available 0x11:IDLE or MAX_BUS_HOLD_TIME timeout
SPI0_MASTER1_STATUS[15:8]	0x3c0d	0x00	RO	[7:0]SPI_MASTER4_STATUS // I2C Slave 0x00:Both buffer0 and buffer1 are busy 0x01:SPI buffer0 is available *0*x10:SPI buffer1 is available *0x11:IDLE or MAX_BUS_HOLD_TIME timeout*
SPI0_MASTER1_STATUS[23:16]	0x3c0e	0x00	RO	[7:0]SPI_MASTER4_STATUS // AUX 0x00:Both buffer0 and buffer1 are busy 0x01:SPI buffer0 is available

				0x10:SPI buffer1 is available *0x11:IDLE or MAX_BUS_HOLD_TIME timeout*
SPI0_BUS_CFG_BUF0[7:0]	0x3c10	0x0c	R/W	[7:0]SPI_MASTER_CONFIG_BUF0 //Bit5: HI_Z Bit4: MULTI_IO_BIT_ORDER Bit3: RX_BIT_ORDER Bit2: TX_BIT_ORDER Bit1: CPHA Bit0: CPOL
SPI0_BUS_CFG_BUF0[15:8]	0x3c11	0x08	R/W	[15:8]MIN_FREE_TIME_BUF0 //Minimum SPI bus free time in units of HCLK cycles
SPI0_MASTER_FREQ_BUF0[7:0]	0x3c14	0x01	R/W	[7: 0]SPI_FREQ_BUF0 Bit7:0 - LO_CNT
SPI0_MASTER_FREQ_BUF0[15:8]	0x3c15	0x01	R/W	[15:8]SPI_FREQ_BUF0 Bit15:8 - HI_CNT
SPI0_CFG0_BUF0[7:0]	0x3c18	0x08	R/W	[7:0]SPI_DUMMY_ADDR_BUF0 single-bit mode:0x08 Dual IO mode:0x04 Dual Output mode:0x08
SPI0_CFG0_BUF0[15:8]	0x3c19	0x81	R/W	[15:8]SPI_DUMMY_DATA_BUF0 //Bit7 - CLKON Bit6:0 - DATA Dummy Cycle
SPI0_CFG1_BUF0	0x3c1c	0x00	R/W	[7:0]SPI_CMD0_BUF0 //SPI CMD Byte= single-bit mode:0x0b Dual IO mode:0xbb Dual Output mode:0x3b
MEGSPI0_CFG1_BUF0[15:8]	0x3c1d	0x01	R/W	SPI_CMD1_BUF0 Bit7 - ADDR_DUMMY_BYPASS Bit6 - ADDR_BYPASS Bit5 - Reserved Bit4 - CRC_EN Bit3 - DATA_POL Bit2:1 - MULTI_IO_MODE 0 = Single Bit Mode Transaction 1 = Dual Mode Transaction 2 = Dual IO Mode Transaction 3 = Dual Output Mode Transaction Bit0 - CMD_SEL

SPI0_CFG1_BUF0[23:16]	0x3c1e	0x03	R/W	SPI_CMD2_BUF0 Bit7 - Reserved Bit6 - MOT(Middle-of-Transaction) Bit5 - HEADER_SEL 0-CMD and ADDR bytes are from the CMD and ADDR registers fields 1-CMD and ADDR bytes are from the head of the DATA FIFO's Bit4 - TYPE 0 - Read 1 - write Bit3 - SPI_UNLOCK_SEL 0 - SPI master unlock itself after SPI write finished 1 - AHB Master unlock SPI Master after SPI write finished Bit2:0 - ADDR_SEL
SPI0_CFG1_BUF0[31:24]	0x3c1f	0xff	R/W	SPI_CRC_INIT_BUF0
SPI0_CFG2_BUF0[7:0]	0x3c20	0x00	R/W	Bit7:0 SPI_DATA_SEL_BUF0[7:0]
SPI0_CFG2_BUF0[15:8]	0x3c21	0x00	R/W	Bit1:0 SPI_DATA_SEL_BUF0[9:8]
SPI0_CFG2_BUF0[23:16]	0x3c22	0x00	R/W	SPI_DUMMY_VALUE_BUF0 single-bit mode:0x00 Dual IO mode:0xf0 Dual Output mode:0x00
SPI0_ADDR_BUF0[7:0]	0x3c24	0x00	R/W	[7:0] SPI Address Byte 0
SPI0_ADDR_BUF0[15:8]	0x3c25	0x00	R/W	[7:0] SPI Address Byte 1
SPI0_ADDR_BUF0[23:16]	0x3c26	0x00	R/W	[7:0] SPI Address Byte 2
SPI0_ADDR_BUF0[31:24]	0x3c27	0x00	R/W	[7:0] SPI Address Byte 3
SPI0_CTRL_BUF0	0x3c28	0x00	R/W	[1] BUF0_FIFO_RESET [0] BUF0_UC_TXN_START
SPI0_MASTER_RAW_STATUS_BUF0	0x4070	0x02	RO	[1]DFIFO0_EMPTY_RAW_STATUS [0] DFIFO0_BUSY_RAW_STATUS

SPI0_MASTER_INT_STATUS_BUF0	0x4074	0x00	RW1C	[1]DFIFO0_EMPTY_INT [0] DFIFO0_BUSY_INT
SPI0_BUS_CFG_BUF17[7:0]	0x407c	0x0c	R/W	[7:0]SPI_MASTER_CONFIG_BUF1 //Bit5: HI_Z Bit4: MULTI_IO_BIT_ORDER Bit3: RX_BIT_ORDER Bit2: TX_BIT_ORDER Bit1: CPHA Bit0: CPOL
SPI0_BUS_CFG_BUF1[15:8]	0x407d	0x08	R/W	[15:8]MIN_FREE_TIME_BUF1 //Minimum SPI bus free time in units of HCLK cycles
SPI0_MASTER_FREQ_BUF1[7:0]	0x4080	0x01	R/W	[7: 0]SPI_FREQ_BUF1 Bit7:0 - LO_CNT
SPI0_MASTER_FREQ_BUF1[15:8]	0x4081	0x01	R/W	[15:8]SPI_FREQ_BUF1 Bit15:8 - HI_CNT
SPI0_CFG0_BUF1[7:0]	0x4084	0x08	R/W	[7:0]SPI_DUMMY_ADDR_BUF1 single-bit mode:0x08 Dual IO mode:0x04 Dual Output mode:0x08
SPI0_CFG0_BUF1[15:8]	0x4085	0x81	R/W	[15:8]SPI_DUMMY_DATA_BUF1 //Bit7 - CLKON Bit6:0 - DATA Dummy Cycle
SPI0_CFG1_BUF1[7:0]	0x4088	0x00	R/W	[7:0]SPI_CMD0_BUF1 //SPI CMD Byte= single-bit mode:0x0b Dual IO mode:0xbb Dual Output mode:0x3b
SPI0_CFG1_BUF1[15:8]	0x4089	0x01	R/W	SPI_CMD1_BUF1 Bit7 - ADDR_DUMMY_BYPASS Bit6 - ADDR_BYPASS Bit5 - Reserved Bit4 - CRC_EN Bit3 - DATA_POL Bit2:1 - MULTI_IO_MODE 0 = Single Bit Mode Transaction 1 = Dual Mode Transaction 2 = Dual IO Mode Transaction 3 = Dual Output Mode Transaction Bit0 - CMD_SEL

SPI0_CFG1_BUF1[23:16]	0x408a	0x03	R/W	SPI_CMD2_BUF0 Bit7 - Reserved Bit6 - MOT(Middle-of-Transaction) Bit5 - HEADER_SEL 0-CMD and ADDR bytes are from the CMD and ADDR registers fields 1-CMD and ADDR bytes are from the head of the DATA FIFO's Bit4 - TYPE 0 - Read 1 - write Bit3 - SPI_UNLOCK_SEL 0 - SPI master unlock itself after SPI write finished 1 - AHB Master unlock SPI Master after SPI write finished Bit2:0 - ADDR_SEL
SPI0_CFG1_BUF1[31:24]	0x408b	0xff	R/W	SPI_CRC_INIT_BUF1
SPI0_CFG2_BUF1[7:0]	0x408c	0x00	R/W	Bit7:0 SPI_DATA_SEL_BUF1[7:0]
SPI0_CFG2_BUF1[15:8]	0x408d	0x00	R/W	Bit1:0 SPI_DATA_SEL_BUF1[9:8]
SPI0_CFG2_BUF1[23:16]	0x408e	0x00	R/W	SPI_DUMMY_VALUE_BUF1 single-bit mode:0x00 Dual IO mode:0xf0 Dual Output mode:0x00
SPI0_ADDR_BUF1[7:0]	0x4090	0x00	R/W	[7:0] SPI Address Byte 0
SPI0_ADDR_BUF1[15:8]	0x4091	0x00	R/W	[7:0] SPI Address Byte 1
SPI0_ADDR_BUF1[23:16]	0x4092	0x00	R/W	[7:0] SPI Address Byte 2
SPI0_ADDR_BUF1[31:24]	0x4093	0x00	R/W	[7:0] SPI Address Byte 3
SPI0_CTRL_BUF1	0x4094	0x00	R/W	[1] BUF1_FIFO_RESET [0] BUF1_UC_TXN_START
SPI0_MASTER_RAW_STATUS_BUF1	0x44d8	0x02	RO	[1]DFIFO1_EMPTY_RAW_STATUS [0] DFIFO1_BUSY_RAW_STATUS

SPI0_MASTER_INT_STATUS_BUF1	0x44dc	0x00	RW1C	[1]DFIFO1_EMPTY_INT [0] DFIFO1_BUSY_INT
SPI0_DATA_BUF0	0x3c30 ~ 0x406c	0x00000000	VR/W	SPI Master0 Buffer0(size:272x32)
SPI0_DATA_BUF1	0x4098 ~ 0x44d4	0x00000000	VR/W	SPI Master0 Buffer1(size:272x32)
SPI1_BUS_CFG_BUF0	0x4810	0x0c	R/W	[7:0]SPI_MASTER_CONFIG_BUF0 //Bit5: HI_Z Bit4: MULTI_IO_BIT_ORDER Bit3: RX_BIT_ORDER Bit2: TX_BIT_ORDER Bit1: CPHA Bit0: CPOL
SPI1_BUS_CFG_BUF0	0x4811	0x08	R/W	[7:0]]MIN_FREE_TIME_BUF0 //Minimum SPI bus free time in units of HCLK cycles
SPI1_MASTER_FREQ_BUF0[7:0]	0x4814	0x01	R/W	Bit7:0 - LO_CNT
SPI1_MASTER_FREQ_BUF0[15:8]	0x4815	0x01	R/W	Bit15:8 - HI_CNT
SPI1_CFG0_BUF0	0x4818	0x08	R/W	[7:0]SPI_DUMMY_ADDR_BUF0
SPI1_CFG0_BUF0	0x4819	0x81	R/W	[7:0]SPI_DUMMY_DATA_BUF0 //Bit7 - CLKON Bit6:0 - DATA Dummy Cycle
SPI1_CFG1_BUF0	0x481c	0x00	R/W	[7:0]SPI_CMD0_BUF0 //SPI CMD Byte
SPI1_CFG1_BUF0	0x481d	0x01	R/W	[7:0] SPI_CMD1_BUF0 Bit7 - ADDR_DUMMY_BYPASS Bit6 - ADDR_BYPASS Bit5 - CS# Number Bit4 - CRC_EN Bit3 - DATA_POL Bit2:1 - MULTI_IO_MODE 0 = Single Bit Mode Transaction 1 = Dual Mode Transaction 2 = Dual IO Mode Transaction 3 = Dual Output Mode Transaction

				Bit0 - CMD_SEL
SPI1_CFG1_BUF0	0x481e	0x03	R/W	[23:16] SPI_CMD2_BUF0 Bit7 - Reserved Bit6 - MOT(Middle-of-Transaction) Bit5 - HEADER_SEL 0-CMD and ADDR bytes are from registers fields (default) 1-CMD and ADDR bytes are from the head of the DATA FIFO's Bit4 - TYPE 0 - Read 1 - write Bit3 - SPI_UNLOCK_SEL 0 - SPI master unlock itself after SPI write finished 1 - AHB Master unlock SPI Master after SPI write finished Bit2:0 - ADDR_SEL
SPI1_CFG1_BUF0	0x481f	0xff	R/W	[31:24]SPI_CRC_INIT_BUF0
SPI1_CFG2_BUF0	0x4820	0x00	R/W	[7:0]SPI_DATA_SEL_BUF0[7:0] // length in bytes minus one of the DATA phase
SPI1_CFG2_BUF0	0x4821	0x00	R/W	[1:0] SPI_DATA_SEL_BUF0[9:8] [7:6] Reserved
SPI1_CFG2_BUF0[7:0]	0x4822	0x00	R/W	[7:0]SPI_DUMMY_VALUE_BUF0[7:0]
SPI1_CFG2_BUF0[15:8]	0x4823	0x00	R/W	[7:0]SPI_DUMMY_VALUE_BUF0[15:8]
SPI1_ADDR_BUF0	0x4824	0x00	R/W	[7:0] SPI Address Byte 0
SPI1_ADDR_BUF0	0x4825	0x00	R/W	[15:8] SPI Address Byte 1
SPI1_ADDR_BUF0	0x4826	0x00	R/W	[23:16] SPI Address Byte 2
SPI1_ADDR_BUF0	0x4827	0x00	R/W	[31:24] SPI Address Byte 3
SPI1_CTRL_BUF0	0x4828	0x00	SC	[1] BUF0_ FIFO_RESET [0] BUF0_UC_TXN_START
SPI1_DATA_BUF0	0x4830 ~ 0x4c6c	0x00	VR/W	SPI Master1 Buffer0(size:272x32) Read pops data, Write pushes data
SPI1_DATA_BUF1	0x4c98 ~ 0x50d4	0x00	VR/W	SPI Master1 Buffer1(size:272x32) Read pops data, Write pushes data
SPI1_MASTER_RAW	0x4c70	0x02	R/W	[1]DFIFO0_EMPTY_RAW_STATUS

_STATUS_BUF0				[0] DFIFO0_BUSY_RAW_STATUS

SPI1_MASTER_INT_STATUS_BUF0	0x4c74	0x00	RW1C	[7]DFIFO0_TIMEOUT_M5_INT
				//I2C Slave
				[6]DFIFO0_TIMEOUT_M4_INT
				//SPI Slave
				[5]DFIFO0_TIMEOUT_M3_INT
				//Slave Data Loader
				[4]DFIFO0_TIMEOUT_M2_INT
				//Master Data Loader
				[3]DFIFO0_TIMEOUT_M1_INT
				//MCU
				[2]DFIFO0_TIMEOUT_M0_INT
				//TCON Sync IF
				[1]DFIFO0_EMPTY_INT
				[0] DFIFO0_BUSY_INT

SPI1_MASTER_INT_CTL_BUF0	0x4c7b	0x00	R/W	[26] DFIFO0_EMPTY_POL [25] DFIFO0_SPI_BUSY_POL
SPI1_BUS_CFG_BUF1	0x4c7c	0x0c	R/W	[7:0]SPI_MASTER_CONFIG_BUF1 //Bit5: HI_Z Bit4: MULTI_IO_BIT_ORDER Bit3: RX_BIT_ORDER Bit2: TX_BIT_ORDER Bit1: CPHA Bit0: CPOL
SPI1_BUS_CFG_BUF1	0x4c7d	0x08	R/W	[7:0]MIN_FREE_TIME_BUF1
SPI1_MASTER_FREQ_BUF1	0x4c80	0x01	R/W	SPI_FREQ_BUF1// Bit7:0 - LO_CNT
SPI1_MASTER_FREQ_BUF1	0x4c81	0x01	R/W	SPI_FREQ_BUF1//Bit15:8 - HI_CNT
SPI1_CFG0_BUF1	0x4c84	0x08	R/W	[7:0]SPI_DUMMY_ADDR_BUF1
SPI1_CFG0_BUF1	0x4c85	0x81	R/W	[7:0]SPI_DUMMY_DATA_BUF1 //Bit7 - CLKON Bit6:0 - DATA Dummy Cycle
SPI1_CFG1_BUF1	0x4c88	0x00	R/W	[7:0]SPI_CMD0_BUF1 //SPI CMD Byte
SPI1_CFG1_BUF1	0x4c89	0x01	R/W	[7:0] SPI_CMD1_BUF1 Bit7 - ADDR_DUMMY_BYPASS Bit6 - ADDR_BYPASS Bit5 - CS# Number Bit4 - CRC_EN Bit3 - DATA_POL

				Bit2:1 - MULTI_IO_MODE 0 = Single Bit Mode Transaction 1 = Dual Mode Transaction 2 = Dual IO Mode Transaction 3 = Dual Output Mode Transaction Bit0 - CMD_SEL
SPI1_CFG1_BUF1	0x4c8a	0x03	R/W	SPI_CMD2_BUF1 Bit7 - Reserved Bit6 - MOT(Middle-of-Transaction) Bit5 - HEADER_SEL 0-CMD and ADDR bytes are from the registers fields (default) 1-CMD and ADDR bytes are from the head of the DATA FIFO's Bit4 - TYPE 0 - Read 1 - write Bit3 - SPI_UNLOCK_SEL 0 - SPI master unlock itself after SPI write finished 1 - AHB Master unlock SPI Master after SPI write finished Bit2:0 - ADDR_SEL
SPI1_CFG1_BUF1	0x4c8b	0xff	R/W	[31:24]SPI_CRC_INIT_BUF1
SPI1_CFG2_BUF1	0x4c8c	0x00	R/W	[7:0]SPI_DATA_SEL_BUF1[7:0] // length in bytes minus one of the DATA phase
SPI1_CFG2_BUF1	0x4c8d	0x00	R/W	[1:0] SPI_DATA_SEL_BUF1[9:8] [7:2] Reserved
SPI1_CFG2_BUF1	0x4c8e	0x00	R/W	[7:0]SPI_DUMMY_VALUE_BUF1
SPI1_ADDR_BUF1	0x4c90	0x00	R/W	[7:0] SPI Address Byte 0
SPI1_ADDR_BUF1	0x4c91	0x00	R/W	[15:8] SPI Address Byte 1
SPI1_ADDR_BUF1	0x4c92	0x00	R/W	[23:16] SPI Address Byte 2
SPI1_ADDR_BUF1	0x4c93	0x00	R/W	[31:24] SPI Address Byte 3
SPI1_CTRL_BUF1	0x4c94	0x00	R/W	[1] BUF1_FIFO_RESET [0] BUF1_UC_TXN_START
SPI1_MASTER1_REQ	0x4805	0x00	R/W	[7:0]AHB_MASTER5_REQ //I2C Slave

SPI1_MASTER1_STATUS	0x480d	0x00	R/W	[7:0]SPI_MASTER5_STATUS //I2C Slave
SPI1_MASTER_RAW_STATUS_BUF1	0x50d8	0x02	RO	[1]DFIFO1_EMPTY_RAW_STATUS [0] DFIFO1_BUSY_RAW_STATUS
SPI1_MASTER_INT_STATUS_BUF1	0x50dc	0x00	RW1C	[7]DFIFO1_TIMEOUT_M5_INT [6]DFIFO1_TIMEOUT_M4_INT [5]DFIFO1_TIMEOUT_M3_INT [4]DFIFO1_TIMEOUT_M2_INT [3]DFIFO1_TIMEOUT_M1_INT [2]DFIFO1_TIMEOUT_M0_INT [1]DFIFO1_EMPTY_INT [0] DFIFO1_BUSY_INT
SPI1_MASTER_INT_CTL_BUF1	0x50e3	0x00	R/W	[26] DFIFO1_EMPTY_POL [25] DFIFO1_SPI_BUSY_POL
SPI1_HOLD_TIME	0x50f4	0x00	R/W	[7:0]MAX_BUS_HOLD_TIME[7:0] ////in units of ms
SPI1_HOLD_TIME	0x50f5	0x00	R/W	[7:0] MAX_BUS_HOLD_TIME[15:8] ////in units of ms
SPI2_BUS_CFG_BUF0	0x5410	0x0c	R/W	[7:0]SP2_MASTER_CONFIG_BUF0 //Bit5: HI_Z Bit4: MULTI_IO_BIT_ORDER Bit3: RX_BIT_ORDER Bit2: TX_BIT_ORDER Bit1: CPHA Bit0: CPOL
SPI2_BUS_CFG_BUF0	0x5411	0x08	R/W	[7:0]]MIN_FREE_TIME_BUF0 //Minimum SPI bus free time in units of HCLK cycles
SPI2_MASTER_FREQ_BUF0[7:0]	0x5414	0x01	R/W	Bit7:0 - LO_CNT
SPI2_MASTER_FREQ_BUF0[15:8]	0x5415	0x01	R/W	Bit15:8 - HI_CNT
SPI2_CFG0_BUF0	0x5418	0x08	R/W	[7:0]SPI_DUMMY_ADDR_BUF0
SPI2_CFG0_BUF0	0x5419	0x81	R/W	[7:0]SPI_DUMMY_DATA_BUF0 //Bit7 - CLKON Bit6:0 - DATA Dummy Cycle
SPI2_CFG1_BUF0	0x541c	0x00	R/W	[7:0]SPI_CMD0_BUF0 //SPI CMD Byte
SPI2_CFG1_BUF0	0x541d	0x01	R/W	[7:0] SPI_CMD1_BUF0

				Bit7 - ADDR_DUMMY_BYPASS Bit6 - ADDR_BYPASS Bit5 - CS# Number Bit4 - CRC_EN Bit3 - DATA_POL Bit2:1 - MULTI_IO_MODE 0 = Single Bit Mode Transaction 1 = Dual Mode Transaction 2 = Dual IO Mode Transaction 3 = Dual Output Mode Transaction Bit0 - CMD_SEL
SPI2_CFG1_BUF0	0x541e	0x03	R/W	[23:16] SPI_CMD2_BUF0 Bit7 - Reserved Bit6 - MOT(Middle-of-Transaction) Bit5 - HEADER_SEL 0-CMD and ADDR bytes are from registers fields (default) 1-CMD and ADDR bytes are from the head of the DATA FIFO's Bit4 - TYPE 0 - Read 1 - write Bit3 - SPI_UNLOCK_SEL 0 - SPI master unlock itself after SPI write finished 1 - AHB Master unlock SPI Master after SPI write finished Bit2:0 - ADDR_SEL
SPI2_CFG1_BUF0	0x541f	0xff	R/W	[31:24]SPI_CRC_INIT_BUF0
SPI2_CFG2_BUF0	0x5420	0x00	R/W	[7:0]SPI_DATA_SEL_BUF0[7:0] // length in bytes minus one of the DATA phase
SPI2_CFG2_BUF0	0x5421	0x00	R/W	[1:0] SPI_DATA_SEL_BUF0[9:8] [7:6] Reserved
SPI2_CFG2_BUF0[7:0]	0x5422	0x00	R/W	[7:0]SPI_DUMMY_VALUE_BUF0[7:0]
SPI2_CFG2_BUF0[15:8]	0x5423	0x00	R/W	[7:0]SPI_DUMMY_VALUE_BUF0[15:8]
SPI2_ADDR_BUF0	0x5424	0x00	R/W	[7:0] SPI Address Byte 0
SPI2_ADDR_BUF0	0x5425	0x00	R/W	[15:8] SPI Address Byte 1
SPI2_ADDR_BUF0	0x5426	0x00	R/W	[23:16] SPI Address Byte 2
SPI2_ADDR_BUF0	0x5427	0x00	R/W	[31:24] SPI Address Byte 3

SPI2_CTRL_BUF0	0x5428	0x00	SC	[1] BUF0_ FIFO_RESET [0] BUF0_UC_TXN_START
SPI2_DATA_BUF0	0x5430 ~ 0x586c	0x00	VR/W	SPI Master2 Buffer0(size:272x32) Read pops data, Write pushes data
SPI2_DATA_BUF1	0x5898 ~ 0x5cd4	0x00	VR/W	SPI Master2 Buffer1(size:272x32) Read pops data, Write pushes data
SPI2_MASTER_RAW_STATUS_BUF0	0x5870	0x02	R/W	[1]DFIFO0_EMPTY_RAW_STATUS [0] DFIFO0_BUSY_RAW_STATUS

SPI2_MASTER_INT_STATUS_BUF0	0x5874	0x00	RW1C	[7]DFIFO0_TIMEOUT_M5_INT
				//I2C Slave
				[6]DFIFO0_TIMEOUT_M4_INT
				//SPI Slave
				[5]DFIFO0_TIMEOUT_M3_INT
				//Slave Data Loader
				[4]DFIFO0_TIMEOUT_M2_INT
				//Master Data Loader
				[3]DFIFO0_TIMEOUT_M1_INT
				//MCU
				[2]DFIFO0_TIMEOUT_M0_INT
				//TCON Sync IF
				[1]DFIFO0_EMPTY_INT
				[0] DFIFO0_BUSY_INT

SPI2_MASTER_INT_CTL_BUF0	0x587b	0x00	R/W	[26] DFIFO0_EMPTY_POL [25] DFIFO0_SPI_BUSY_POL
SPI2_BUS_CFG_BUF1	0x587c	0x0c	R/W	[7:0]SPI_MASTER_CONFIG_BUF1 //Bit5: HI_Z Bit4: MULTI_IO_BIT_ORDER Bit3: RX_BIT_ORDER Bit2: TX_BIT_ORDER Bit1: CPHA Bit0: CPOL
SPI2_BUS_CFG_BUF1	0x587d	0x08	R/W	[7:0]MIN_FREE_TIME_BUF1
SPI2_MASTER_FREQ_BUF1	0x5880	0x01	R/W	SPI_FREQ_BUF1// Bit7:0 - LO_CNT
SPI2_MASTER_FREQ_BUF1	0x5881	0x01	R/W	SPI_FREQ_BUF1//Bit15:8 - HI_CNT
SPI2_CFG0_BUF1	0x5884	0x08	R/W	[7:0]SPI_DUMMY_ADDR_BUF1
SPI2_CFG0_BUF1	0x5885	0x81	R/W	[7:0]SPI_DUMMY_DATA_BUF1 //Bit7 - CLKON

				Bit6:0 - DATA Dummy Cycle
SPI2_CFG1_BUF1	0x5888	0x00	R/W	[7:0]SPI_CMD0_BUF1 //SPI CMD Byte
SPI2_CFG1_BUF1	0x5889	0x01	R/W	[7:0] SPI_CMD1_BUF1 Bit7 - ADDR_DUMMY_BYPASS Bit6 - ADDR_BYPASS Bit5 - CS# Number Bit4 - CRC_EN Bit3 - DATA_POL Bit2:1 - MULTI_IO_MODE 0 = Single Bit Mode Transaction 1 = Dual Mode Transaction 2 = Dual IO Mode Transaction 3 = Dual Output Mode Transaction Bit0 - CMD_SEL
SPI2_CFG1_BUF1	0x588a	0x03	R/W	SPI_CMD2_BUF1 Bit7 - Reserved Bit6 - MOT(Middle-of-Transaction) Bit5 - HEADER_SEL 0-CMD and ADDR bytes are from the registers fields (default) 1-CMD and ADDR bytes are from the head of the DATA FIFO's Bit4 - TYPE 0 - Read 1 - write Bit3 - SPI_UNLOCK_SEL 0 - SPI master unlock itself after SPI write finished 1 - AHB Master unlock SPI Master after SPI write finished Bit2:0 - ADDR_SEL
SPI2_CFG1_BUF1	0x588b	0xff	R/W	[31:24]SPI_CRC_INIT_BUF1
SPI2_CFG2_BUF1	0x588c	0x00	R/W	[7:0]SPI_DATA_SEL_BUF1[7:0] // length in bytes minus one of the DATA phase
SPI2_CFG2_BUF1	0x588d	0x00	R/W	[1:0] SPI_DATA_SEL_BUF1[9:8] [7:2] Reserved
SPI2_CFG2_BUF1	0x588e	0x00	R/W	[7:0]SPI_DUMMY_VALUE_BUF1

SPI2_ADDR_BUF1	0x5890	0x00	R/W	[7:0] SPI Address Byte 0
SPI2_ADDR_BUF1	0x5891	0x00	R/W	[15:8] SPI Address Byte 1
SPI2_ADDR_BUF1	0x5892	0x00	R/W	[23:16] SPI Address Byte 2
SPI2_ADDR_BUF1	0x5893	0x00	R/W	[31:24] SPI Address Byte 3
SPI2_CTRL_BUF1	0x5894	0x00	R/W	[1] BUF1_FIFO_RESET [0] BUF1_UC_TXN_START
SPI2_MASTER1_REQ	0x5405	0x00	R/W	[7:0]AHB_MASTER5_REQ //I2C Slave
SPI2_MASTER1_STATUS	0x540d	0x00	R/W	[7:0]SPI_MASTER5_STATUS //I2C Slave
SPI2_MASTER_RAW_STATUS_BUF1	0x5cd8	0x02	RO	[1]DFIFO1_EMPTY_RAW_STATUS [0] DFIFO1_BUSY_RAW_STATUS
SPI2_MASTER_INT_STATUS_BUF1	0x5cdc	0x00	RW1C	[7]DFIFO1_TIMEOUT_M5_INT [6]DFIFO1_TIMEOUT_M4_INT [5]DFIFO1_TIMEOUT_M3_INT [4]DFIFO1_TIMEOUT_M2_INT [3]DFIFO1_TIMEOUT_M1_INT [2]DFIFO1_TIMEOUT_M0_INT [1]DFIFO1_EMPTY_INT [0] DFIFO1_BUSY_INT
SPI2_MASTER_INT_CTL_BUF1	0x5ce3	0x00	R/W	[26] DFIFO1_EMPTY_POL [25] DFIFO1_SPI_BUSY_POL
SPI2_HOLD_TIME	0x5cf4	0x00	R/W	[7:0]MAX_BUS_HOLD_TIME[7:0] ////in units of ms
SPI2_HOLD_TIME	0x5cf5	0x00	R/W	[7:0] MAX_BUS_HOLD_TIME[15:8] ////in units of ms

5.3 Sample code for SPI MASTER0 write by I2C slave

length = 1

spi_addr_init = 0x000000
erase_cmd = 0xd8 ##0x20:4KB erase for 3-byte addr, 0x52:32KB, 0xd8:64KB
wen_cmd = 0x06
rdsr_cmd = 0x05
Write_CMD = 0x02

##addr_byte3 = (spi_addr_init>>24) & 0xff

```
addr_byte2 = (spi_addr_init>>16) & 0xff
addr_byte1 = (spi_addr_init>>8) & 0xff
addr_byte0 = spi_addr_init & 0xff


def spi_fifo0_init():
    WriteReg(0x10,0x3c,0x10,0x2c)
    WriteReg(0x10,0x3c,0x11,0x08)
    WriteReg(0x10,0x3c,0x14,0x01)
    WriteReg(0x10,0x3c,0x15,0x01)

def spi_fifo1_init():
    WriteReg(0x10,0x40,0x7c,0x2c)
    WriteReg(0x10,0x40,0x7d,0x08)
    WriteReg(0x10,0x40,0x80,0x01)
    WriteReg(0x10,0x40,0x81,0x01)

def setctor_erase_fifo0():
    spi_fifo0_init()
    WriteReg(0x10,0x3c,0x18,0x00)
    WriteReg(0x10,0x3c,0x19,0x81)
    WriteReg(0x10,0x3c,0x1c,erase_cmd)
    WriteReg(0x10,0x3c,0x1d,0xc1)
    WriteReg(0x10,0x3c,0x1e,0x58)
    WriteReg(0x10,0x3c,0x20,0x02)
    WriteReg(0x10,0x3c,0x21,0x00)
    WriteReg(0x10,0x3c,0x30,addr_byte2)
    WriteReg(0x10,0x3c,0x31,addr_byte1)
    WriteReg(0x10,0x3c,0x32,addr_byte0)
    WriteReg(0x10,0x3c,0x28,0x01)
    print 'Subsetctor Erase is in progress'

def setctor_erase_fifo1():
    spi_fifo1_init()
    WriteReg(0x10,0x40,0x84,0x00)
    WriteReg(0x10,0x40,0x85,0x81)
    WriteReg(0x10,0x40,0x88,erase_cmd)
    WriteReg(0x10,0x40,0x89,0xc1)
    WriteReg(0x10,0x40,0x8a,0x58)
    WriteReg(0x10,0x40,0x8c,0x02)
    WriteReg(0x10,0x40,0x8d,0x00)
    WriteReg(0x10,0x40,0x98,addr_byte2)
```

```
    WriteReg(0x10,0x40,0x99,addr_byte1)
    WriteReg(0x10,0x40,0x9a,addr_byte0)
    WriteReg(0x10,0x40,0x94,0x01)
    print 'Subsetctor Erase is in progress'

def write_enable_fifo0():
    spi_fifo0_init()
    WriteReg(0x10,0x3c,0x18,0x00)
    WriteReg(0x10,0x3c,0x19,0x81)
    WriteReg(0x10,0x3c,0x1d,0xc0)
    WriteReg(0x10,0x3c,0x1e,0x58)
    WriteReg(0x10,0x3c,0x20,0x00)
    WriteReg(0x10,0x3c,0x21,0x00)
    WriteReg(0x10,0x3c,0x30,wen_cmd)
    WriteReg(0x10,0x3c,0x28,0x01)
    print 'Write Enable CMD'

def write_enable_fifo1():
    spi_fifo1_init()
    WriteReg(0x10,0x40,0x84,0x00)
    WriteReg(0x10,0x40,0x85,0x81)
    WriteReg(0x10,0x40,0x89,0xc0)
    WriteReg(0x10,0x40,0x8a,0x58)
    WriteReg(0x10,0x40,0x8c,0x00)
    WriteReg(0x10,0x40,0x8d,0x00)
    WriteReg(0x10,0x40,0x98,wen_cmd)
    WriteReg(0x10,0x40,0x94,0x01)
    print 'Write Enable CMD'

def rdsr_fifo0():
    spi_fifo0_init()
    global STATUS_REG
    WriteReg(0x10,0x3c,0x18,0x00)
    WriteReg(0x10,0x3c,0x19,0x81)
    WriteReg(0x10,0x3c,0x1c,rdsr_cmd)
    WriteReg(0x10,0x3c,0x1d,0xc1)
    WriteReg(0x10,0x3c,0x1e,0x40)
    WriteReg(0x10,0x3c,0x20,0x00)
    WriteReg(0x10,0x3c,0x21,0x00)
    WriteReg(0x10,0x3c,0x28,0x01)
    STATUS_REG = ReadReg(0x10,0x3c,0x30)
```

```
def rdsr_clr_fifo0():
    spi_fifo0_init()
    global STATUS_REG
    WriteReg(0x10,0x3c,0x18,0x00)
    WriteReg(0x10,0x3c,0x19,0x81)
    WriteReg(0x10,0x3c,0x1c,rdsr_cmd)
    WriteReg(0x10,0x3c,0x1d,0xc1)
    WriteReg(0x10,0x3c,0x1e,0x00)
    WriteReg(0x10,0x3c,0x20,0x00)
    WriteReg(0x10,0x3c,0x21,0x00)
    WriteReg(0x10,0x3c,0x28,0x01)
    STATUS_REG = ReadReg(0x10,0x3c,0x30)

def rdsr_fifo1():
    spi_fifo1_init()
    global STATUS_REG
    WriteReg(0x10,0x40,0x84,0x00)
    WriteReg(0x10,0x40,0x85,0x81)
    WriteReg(0x10,0x40,0x88,rdsr_cmd)
    WriteReg(0x10,0x40,0x89,0xc1)
    WriteReg(0x10,0x40,0x8a,0x40)
    WriteReg(0x10,0x40,0x8c,0x00)
    WriteReg(0x10,0x40,0x8d,0x00)
    WriteReg(0x10,0x40,0x94,0x01)
    STATUS_REG = ReadReg(0x10,0x40,0x98)

def rdsr_clr_fifo1():
    spi_fifo1_init()
    global STATUS_REG
    WriteReg(0x10,0x40,0x84,0x00)
    WriteReg(0x10,0x40,0x85,0x81)
    WriteReg(0x10,0x40,0x88,rdsr_cmd)
    WriteReg(0x10,0x40,0x89,0xc1)
    WriteReg(0x10,0x40,0x8a,0x00)
    WriteReg(0x10,0x40,0x8c,0x00)
    WriteReg(0x10,0x40,0x8d,0x00)
    WriteReg(0x10,0x40,0x94,0x01)
    STATUS_REG = ReadReg(0x10,0x40,0x98)

def spi_fifo0_write():
    spi_fifo0_init()
    f_write=open('SPIM0_FIFO0_DATA_WRITE.txt','w')
```

```
    for j in range(length*4):
        write_enable_fifo0()
        SPI_ADDR=j*256 + spi_addr_init
        for k in range(0,256):
            FIFO0_ADDR=k + 0x3c30
            data=int(random.random()*255)& 0xff
            WriteReg(0x10,FIFO0_ADDR>>8, FIFO0_ADDR&0xff,data)
            f_write.writelines(hex(data)[2:].zfill(2))
            f_write.writelines('\n')
        WriteReg(0x10,0x3c,0x1c,Write_CMD)
        WriteReg(0x10,0x3c,0x1d,0x81)
        WriteReg(0x10,0x3c,0x1e,0x5b)
        WriteReg(0x10,0x3c,0x20,0xff) ##one SPI write max length is 256
        WriteReg(0x10,0x3c,0x21,0x00)
        WriteReg(0x10,0x3c,0x24,SPI_ADDR&0xff)
        WriteReg(0x10,0x3c,0x25,(SPI_ADDR>>8)&0xff)
        WriteReg(0x10,0x3c,0x26,(SPI_ADDR>>16)&0xff)
        WriteReg(0x10,0x3c,0x28,0x01)
        while(1):
            SPI_STATUS_FIFO0 = ReadReg(0x10,0x40,0x70)
            SPI_BUSY_FLAG_FIFO0 = SPI_STATUS_FIFO0 & 0x01
            if(SPI_BUSY_FLAG_FIFO0 == 0x01):
                print "!!!SPI BUS is busy!!!"
            else:
                print '***SPI Write Round%d is done***' %j
                break
    f_write.close()

def spi_fifo1_write():
    spi_fifo1_init()
    f_write=open('SPIM0_FIFO1_DATA_WRITE.txt','w')
    for j in range(length):
        write_enable_fifo1()
        SPI_ADDR=j*256 + spi_addr_init
        for k in range(0,256):
            FIFO1_ADDR=k + 0x4098
            data=int(random.random()*255)& 0xff
            WriteReg(0x10,FIFO1_ADDR>>8, FIFO1_ADDR&0xff,data)
            f_write.writelines(hex(data)[2:].zfill(2))
            f_write.writelines('\n')
        WriteReg(0x10,0x40,0x88,Write_CMD)
        WriteReg(0x10,0x40,0x89,0x81)
```

```
            WriteReg(0x10,0x40,0x8a,0x5b)
            WriteReg(0x10,0x40,0x8c,0xff) ##one SPI write max length is 256
            WriteReg(0x10,0x40,0x8d,0x00)
            WriteReg(0x10,0x40,0x90,SPI_ADDR&0xff)
            WriteReg(0x10,0x40,0x91,(SPI_ADDR>>8)&0xff)
            WriteReg(0x10,0x40,0x92,(SPI_ADDR>>16)&0xff)
            WriteReg(0x10,0x40,0x94,0x01)
            while(1):
                SPI_STATUS_FIFO1 = ReadReg(0x10,0x44,0xd8)
                SPI_BUSY_FLAG_FIFO1 = SPI_STATUS_FIFO1 & 0x01
                if(SPI_BUSY_FLAG_FIFO1 == 0x01):
                    print "!!!SPI BUS is busy!!!"
                else:
                    print '***SPI Write Round%d is done***' %j
                    break
        f_write.close()



##*************** Lock Sequence Begin ******************
##**send request to SPI Master0**
WriteReg(0x10,0x3c, 0x05,0x0d) ##i2c_slave_id = 0x0d

##***polling the status of ping-pong buffers in SPI Master0***
while(1):
    SPI_MASTER_STATUS = ReadReg(0x10,0x3c,0x0d)
    if((SPI_MASTER_STATUS == 0x03) or (SPI_MASTER_STATUS == 0x00)):
        print "SPI busy"
    elif(SPI_MASTER_STATUS == 0x01):
        fifo_sel = 0
        print "SPI FIFO0 bundle is locked"
        break
    elif(SPI_MASTER_STATUS == 0x02):
        fifo_sel = 1
        print "SPI FIFO1 bundle is locked"
        break
    else:
        print" !!!SPI lock error!!! "
        KeyInt()

##***Transaction Sequence Begin***
##***************SPI NVMEM Erase************************
```

```
if(fifo_sel): ##SPI Master Buffer1 is used
    write_enable_fifo1()
    setctor_erase_fifo1()
    while(1):
        rdsr_fifo1()
        BUSY_STATUS = STATUS_REG & 0x01
        if(BUSY_STATUS):
            print "*** Erase is ongoing ***"
        else:
            print "*** Erase is finshed ***"
            break
    rdsr_clr_fifo1()
else:           ##SPI Master Buffer0 is used
    write_enable_fifo0()
    setctor_erase_fifo0()
    while(1):
        rdsr_fifo0()
        BUSY_STATUS = STATUS_REG & 0x01
        if(BUSY_STATUS):
            print "*** Erase is ongoing ***"
        else:
            print "*** Erase is finshed ***"
            break
    rdsr_clr_fifo0()

##************Write to SPI NVMEM************************
if(fifo_sel): ##SPI Master Buffer1 is used
    spi_fifo1_write()
    while(1):
        rdsr_fifo1()
        BUSY_STATUS = STATUS_REG & 0x01
        if(BUSY_STATUS):
            print "*** Program is ongoing ***"
        else:
            print "*** Program is finshed ***"
            break
    rdsr_clr_fifo1()
else:           ##SPI Master Buffer0 is used
    spi_fifo0_write()
    while(1):
        rdsr_fifo0()
        BUSY_STATUS = STATUS_REG & 0x01
```

```
        if(BUSY_STATUS):
            print "*** Program is ongoing ***"
        else:
            print "*** Program is finshed ***"
            break
    rdsr_clr_fifo0()


##***Unlock the SPI Master by I2C slave***
if(fifo_sel):
    WriteReg(0x10,0x3c,0x05,0x01) ##write 0x01 to release SPI buffer1
else:
    WriteReg(0x10,0x3c,0x05,0x00) ##write 0x00 to release SPI buffer0
```

5.4 Sample code for SPI MASTER0 read by I2C slave

```
length       = 1023 ##the range is 0~1023(1-1K)
length_low   = length & 0xff
length_high = (length>>8) & 0x03

spi_addr_init = 0x000000
multi_io_mode = 0      ## for Boot ROM, multi_io_mode should always be 0

if(multi_io_mode == 0):      ##single-bit mode
    Read_CMD        = 0x0b
    addr_dummy_num  = 0x08
    addr_dummy_value = 0x00
elif(multi_io_mode == 2):   ##Dual IO mode
    Read_CMD        = 0xbb
    addr_dummy_num  = 0x04
    addr_dummy_value = 0xf0
elif(multi_io_mode == 3):   ##Dual Output mode
    Read_CMD        = 0x3b
    addr_dummy_num  = 0x08
    addr_dummy_value = 0x00
else:
    print "!!!Dual Mode(2-2-2) is not supported!!!"
    sys.exit(0)
```

```
##addr_byte3 = (spi_addr_init>>24) & 0xff
addr_byte2 = (spi_addr_init>>16) & 0xff
addr_byte1 = (spi_addr_init>>8) & 0xff
addr_byte0 = spi_addr_init & 0xff

def spi_fifo0_init():
    WriteReg(0x10,0x3c,0x10,0x2c)
    WriteReg(0x10,0x3c,0x11,0x08)
    WriteReg(0x10,0x3c,0x14,0x01)
    WriteReg(0x10,0x3c,0x15,0x01)

def spi_fifo1_init():
    WriteReg(0x10,0x40,0x7c,0x2c)
    WriteReg(0x10,0x40,0x7d,0x08)
    WriteReg(0x10,0x40,0x80,0x01)
    WriteReg(0x10,0x40,0x81,0x01)

def spi_fifo0_read():
    spi_fifo0_init()
    WriteReg(0x10,0x3c,0x18,addr_dummy_num)
    WriteReg(0x10,0x3c,0x19,0x81)
    WriteReg(0x10,0x3c,0x1c,Read_CMD)
    WriteReg(0x10,0x3c,0x1d,0x01)
    WriteReg(0x10,0x3c,0x1e,0x03)
    WriteReg(0x10,0x3c,0x20,length_low)
    WriteReg(0x10,0x3c,0x21,length_high)
    WriteReg(0x10,0x3c,0x22,addr_dummy_value)
    WriteReg(0x10,0x3c,0x24,addr_byte0)
    WriteReg(0x10,0x3c,0x25,addr_byte1)
    WriteReg(0x10,0x3c,0x26,addr_byte2)
    WriteReg(0x10,0x3c,0x28,0x01)
    while(1):
        SPI_STATUS_FIFO0 = ReadReg(0x10,0x40,0x70)
        SPI_BUSY_FLAG_FIFO0 = SPI_STATUS_FIFO0 & 0x01
        if(SPI_BUSY_FLAG_FIFO0 == 0x01):
            print "!!!SPI BUS is busy!!!"
        else:
            print "SPI transaction is done"
            break
    file_read = 'SPIM0_FIFO0_DATA_READ.txt'
    fp = open(file_read, 'w')
```

```
    for i in range(length+1):
        FIFO0_ADDR = 0x3c30 + i ##0x3c30 is SRAM0 base addr
        FIFO0_DATA =
ReadReg(0x10,(FIFO0_ADDR>>8)&0xff,FIFO0_ADDR&0xff)
        fp.writelines(hex(FIFO0_DATA)[2:].zfill(2))
        fp.writelines('\n')
    fp.close()

def spi_fifo1_read():
    spi_fifo1_init()
    WriteReg(0x10,0x40,0x84,addr_dummy_num)
    WriteReg(0x10,0x40,0x85,0x81)
    WriteReg(0x10,0x40,0x88,Read_CMD)
    WriteReg(0x10,0x40,0x89,0x01)
    WriteReg(0x10,0x40,0x8a,0x03)
    WriteReg(0x10,0x40,0x8c,length_low)
    WriteReg(0x10,0x40,0x8d,length_high)
    WriteReg(0x10,0x40,0x8e,addr_dummy_value)
    WriteReg(0x10,0x40,0x90,addr_byte0)
    WriteReg(0x10,0x40,0x91,addr_byte1)
    WriteReg(0x10,0x40,0x92,addr_byte2)
    WriteReg(0x10,0x40,0x94,0x01)
    while(1):
        SPI_STATUS_FIFO1 = ReadReg(0x10,0x44,0xd8)
        SPI_BUSY_FLAG_FIFO1 = SPI_STATUS_FIFO1 & 0x01
        if(SPI_BUSY_FLAG_FIFO1 == 0x01):
            print "!!!SPI BUS is busy!!!"
        else:
            print "SPI transaction is done"
            break
    file_read = 'SPIM0_FIFO1_DATA_READ.txt'
    fp = open(file_read, 'w')
    for i in range(length+1):
        FIFO1_ADDR = 0x4098 + i ##0x4098 is SRAM1 base addr
        FIFO1_DATA =
ReadReg(0x10,(FIFO1_ADDR>>8)&0xff,FIFO1_ADDR&0xff)
        fp.writelines(hex(FIFO1_DATA)[2:].zfill(2))
        fp.writelines('\n')
    fp.close()
```

```
##***************send request to SPI Master0******************
WriteReg(0x10,0x3c, 0x05,0x0d) ##i2c_slave_id = 0x0d

##***polling the status of ping-pong buffers in SPI Master0***
while(1):
    SPI_MASTER_STATUS = ReadReg(0x10,0x3c,0x0d)
    if((SPI_MASTER_STATUS == 0x03) or (SPI_MASTER_STATUS == 0x00)):
        print "SPI busy"
    else:
        print "SPI_MASTER_STATUS=0x%02x" %SPI_MASTER_STATUS
        break

##***SPI Master0 is locked by MCU***
if(SPI_MASTER_STATUS == 0x01):
    fifo_sel = 0
    print "SPI FIFO0 is locked"
elif(SPI_MASTER_STATUS == 0x02):
    print "SPI FIFO1 is locked"
    fifo_sel = 1
else:
    print" !!!SPI lock error!!! "
    KeyInt()

##************Read from SPI NVMEM************************
if(fifo_sel): ##SPI Master Buffer1 is used
    spi_fifo1_init()
    spi_fifo1_read()
else:           ##SPI Master Buffer0 is used
    spi_fifo0_init()
    spi_fifo0_read()


##***Unlock the SPI Master by I2C slave***
if(fifo_sel):
    WriteReg(0x10,0x3c,0x05,0x01) ##write 0x01 to release SPI buffer1
else:
    WriteReg(0x10,0x3c,0x05,0x00) ##write 0x00 to release SPI buffer0
```

6. Indirect register access over aux

In order to access the internal register over aux , the following sequence should be followed :

a. An unlock sequence of 11 bytes is written to DPCD00490

1. 11 single byte writes(0x50,0x41,0x52,0x41,0x41,0x55,0x58,0x02, 0x52,0x45,0x47) to DPCD00490
2. Read DPCD00490
3. If the read value is 0x00, repeat #1

b. The internal register address is set to DPCD00491 and DPCD00492

c. A single byte read or write to DPCD00493 accesses the internal register addressed by the page and offset set previously

d. After the access is complete, 0x00 should be written to DPCD00490 to re-lock the access

DPCD Address	Field Name	Initial Value	Read Write	Description
DPCD00490h	AUX_REGACC_EN	0x00	R/W	Internal register access enable via aux
DPCD00491h	AUX_REGACC_OFS_MSB	0x00	R/W	Internal register page selection
DPCD00492h	AUX_REGACC_OFS_LSB	0x00	R/W	Internal register offset
DPCD00493h	AUX_REGACC_DATA	0x00	R/W	Internal register value

7. FW Command Interface

Register name	Offset	Initial Value	Read Write	Description
REG_DPCD00410h	0x0370	0x00	R/W	Data 0 for FW interface
REG_DPCD00411h	0x0371	0x00	R/W	Data 1 for FW interface
REG_DPCD00412h	0x0372	0x00	R/W	Data 2 for FW interface
REG_DPCD00413h	0x0373	0x00	R/W	Data 3 for FW interface
REG_DPCD00414h	0x0374	0x00	R/W	Data 4 for FW interface
REG_DPCD00415h	0x0375	0x00	R/W	Data 5 for FW interface
REG_DPCD00416h	0x0376	0x00	R/W	Data 6 for FW interface
REG_DPCD00417h	0x0377	0x00	R/W	Data 7 for FW interface
REG_DPCD00418h	0x0488	0x00	R/W	Data 8 for FW interface
REG_DPCD00419h	0x0489	0x00	R/W	Data 9 for FW interface
REG_DPCD0041ah	0x048a	0x00	R/W	Data 10 for FW interface
REG_DPCD0041bh	0x048b	0x00	R/W	Data 11 for FW interface
REG_DPCD0041ch	0x048c	0x00	R/W	Data 12 for FW interface
REG_DPCD0041dh	0x048d	0x00	R/W	Data 13 for FW interface
REG_DPCD0041eh	0x048e	0x00	R/W	Data 14 for FW interface
REG_DPCD0041fh	0x048f	0x00	R/W	Data 15 for FW interface
REG_DPCD00424h	0x03d4	0x00	R/W	NVM load status
REG_DPCD004f8h	0x0490	0x00	R/W	REQ_ARG2
REG_DPCD004f9h	0x0491	0x00	R/W	REQ_ARG1
REG_DPCD004fah	0x0492	0x00	R/W	REQ_ARG0
REG_DPCD004fbh	0x0493	0x00	R/W	REQ_FUNC 0x01:read internal register 0x02:write internal register 0x42:read spi nvm(16 bytes) 0x43:write nvm(16 bytes) 0x44:erase spi nvm(4k bytes)

REG_DPCD004fch	0x0494	0x00	R/W	Bit7:REQ_TRIGGER Bit 6:REQ_STATUS BIT5:REQ_ERROR BIT4:REQ_COMP
REG_DPCD004fdh	0x0495	0x00	R/W	FW_COUNT
REG_DPCD004feh	0x0496	0x00	R/W	FW Version[15:8]
REG_DPCD004ffh	0x0497	0x00	R/W	FW Version[7:0]

7.1 Internal Register Read/Write Sequencewo

Read internal registers

Input:

DPCD004fch: 0x80

DPCD004fbh: 0x01

DPCD004fah : Page number

DPCD004f9h: Offset

DPCD004f8h: Number of consecutive register to be read – 1

DPCD00410h~DPCD0041fh : Ignore

Return:

DPCD004fch: Status

DPCD00410h~DPCD0041fh : Values read back from registers

Write internal registers

Input:

DPCD004fch: 0x80

DPCD004fbh: 0x02

DPCD004fah : Page number

DPCD004f9h: Offset

DPCD004f8h: Number of consecutive register to be written – 1

DPCD00410h~DPCD0041fh : Values to be written to registers

Return:

DPCD004fch: Status

7.2 SPI NVM Read/Write/Erase SequenceRead SPI NVRAM (16 bytes)

Read SPI NVRAM (16 bytes)

Input:

DPCD004fch: 0x80

DPCD004fbh: 0x42

DPCD004fah : SPI NVMEM address[27:20]

DPCD004f9h: SPI NVMEM address[19:12]

DPCD004f8h: SPI NVMEM address[11:4]

DPCD00410h~DPCD0041fh : Ignore

Return:

DPCD004fch: Status

DPCD00410h~DPCD0041fh : Data read back from SPI NVMEM

Write SPI NVRAM (16 bytes)

Input:

DPCD004fch: 0x80

DPCD004fbh: 0x43

DPCD004fah : SPI NVMEM address[27:20]

DPCD004f9h: SPI NVMEM address[19:12]

DPCD004f8h: SPI NVMEM address[11:4]

DPCD00410h~DPCD0041fh : 16 byte data to write

Return:

DPCD004fch: Status

Erase SPI NVRAM (4K bytes)

Input:

DPCD004fch: 0x80

DPCD004fbh: 0x44

DPCD004fah : SPI NVMEM address[27:20]

DPCD004f9h: SPI NVMEM address[19:12]

DPCD004f8h: SPI NVMEM address[11:4]

DPCD00410h~DPCD0041fh : Ignore

Return:

DPCD004fch: Status

8. BIST Pattern Generation

8.1 BIST Pattern Selection

Register name	Offset	Initial Value	Read Write	Description
REG_BIST_MODE0[7:0]	0x0624	0x40	R/W	[5:0] REG_BIST_PAT0 [7:6] REG_BIST_PAT1[1:0]
REG_BIST_MODE0[15:8]	0x0625	0x20	R/W	[11:8] REG_BIST_PAT1[5:2] [15:12] REG_BIST_PAT2[3:0]
REG_BIST_MODE0[23:16]	0x0626	0x0c	R/W	[17:16] REG_BIST_PAT2[5:4] [23:18] REG_BIST_PAT3[5:0]
REG_BIST_MODE0[31;24]	0x0627	0x04	R/W	[29:24] REG_BIST_PAT4
REG_BIST_MODE1[7:0]	0x0628	0x85	R/W	[5:0] REG_BIST_PAT5 [7:6] REG_BIST_PAT6[1:0]
REG_BIST_MODE1[15:8]	0x0629	0x71	R/W	[11:8] REG_BIST_PAT6[5:2] [15:12] REG_BIST_PAT7[3:0]
REG_BIST_MODE1[23:16]	0x062a	0x20	R/W	[17:16] REG_BIST_PAT7[5:4] [23:18] REG_BIST_PAT8[5:0]
REG_BIST_MODE1[32:24]	0x062b	0x09	R/W	[29:24] REG_BIST_PAT9
REG_BIST_MODE2[7:0]	0x062c	0xca	R/W	[5:0] REG_BIST_PAT10 [7:6] REG_BIST_PAT11[1:0]
REG_BIST_MODE2[15:8]	0x062d	0xc2	R/W	[11:8] REG_BIST_PAT11[5:2] [15:12] REG_BIST_PAT12[3:0]
REG_BIST_MODE2[23:16]	0x062e	0x34	R/W	[17:16] REG_BIST_PAT12[5:4] [23:18] REG_BIST_PAT13[5:0]
REG_BIST_MODE2[32:24]	0x062f	0x0e	R/W	[29:24] REG_BIST_PAT14
REG_BIST_MODE3[	0x0630	0x0f	R/W	[5:0] REG_BIST_PAT15

7:0]				[7:6] REG_BIST_PAT16[1:0]
REG_BIST_MODE3[15:8]	0x0631	0x14	R/W	[11:8] REG_BIST_PAT16[5:2] [15:12] REG_BIST_PAT17[3:0]
REG_BIST_MODE3[23:16]	0x0632	0x49	R/W	[17:16] REG_BIST_PAT17[5:4] [23:18] REG_BIST_PAT18[5:0]
REG_BIST_MODE3[32:24]	0x0633	0x13	R/W	[29:24] REG_BIST_PAT19
REG_BIST_MODE4[7:0]	0x0634	0x54	R/W	[5:0] REG_BIST_PAT20 [7:6] REG_BIST_PAT21[1:0]
REG_BIST_MODE4[15:8]	0x0635	0x65	R/W	[11:8] REG_BIST_PAT21[5:2] [15:12] REG_BIST_PAT22[3:0]
REG_BIST_MODE4[23:16]	0x0636	0x5d	R/W	[17:16] REG_BIST_PAT22[5:4] [23:18] REG_BIST_PAT23[5:0]
REG_BIST_MODE4[32:24]	0x0637	0x18	R/W	[29:24] REG_BIST_PAT24
REG_BIST_MODE5[7:0]	0x0638	0x99	R/W	[5:0] REG_BIST_PAT25 [7:6] REG_BIST_PAT26[1:0]
REG_BIST_MODE5[15:8]	0x0639	0xB	R/W	[11:8] REG_BIST_PAT26[5:2] [15:12] REG_BIST_PAT27[3:0]
REG_BIST_MODE5[23:16]	0x063a	0x71	R/W	[17:16] REG_BIST_PAT27[5:4] [23:18] REG_BIST_PAT28[5:0]
REG_BIST_MODE5[32:24]	0x063b	0x1d	R/W	[29:24] REG_BIST_PAT29
REG_BIST_MODE6[7:0]	0x063c	0xDE	R/W	[5:0] REG_BIST_PAT30 [7:6] REG_BIST_PAT31[1:0]
REG_BIST_MODE6[15:8]	0x063d	0x07	R/W	[11:8] REG_BIST_PAT31[5:2] [15:12] REG_BIST_PAT32[3:0]
REG_BIST_MODE6[23:16]	0x063e	0x86	R/W	[17:16] REG_BIST_PAT32[5:4] [23:18] REG_BIST_PAT33[5:0]
REG_BIST_MODE6[32:24]	0x063f	0x22	R/W	[29:24] REG_BIST_PAT34

REG_BIST_MODE7[7:0]	0x0640	0x23	R/W	[5:0] REG_BIST_PAT35 [7:6] REG_BIST_PAT36[1:0]
REG_BIST_MODE7[15:8]	0x0641	0x59	R/W	[11:8] REG_BIST_PAT36[5:2] [15:12] REG_BIST_PAT37[3:0]
REG_BIST_MODE7[23:16]	0x0642	0x9A	R/W	[17:16] REG_BIST_PAT37[5:4] [23:18] REG_BIST_PAT38[5:0]
REG_BIST_MODE7[32:24]	0x0643	0x27	R/W	[29:24] REG_BIST_PAT39
REG_BIST_CTRL0[7:0]	0x0648	0x0f	R/W	[7:0] REG_BIST_Length
REG_BIST_CTRL0[15:8]	0x0649	0x77	R/W	[7:0] REG_BIST_cycle
REG_BIST_CTRL0[23:16]	0x064a	0x77	R/W	[7:0] REG_CVCNT
REG_BIST_CTRL0[31:24]	0x064b	0x2f	R/W	[7:0] REG_CHCNT
REG_BIST_CTRL1[7:0]	0x064c	0x00	R/W	[7:0] REG_VIDEO_MCTL
REG_BIST_CTRL1[15:8]	0x064d	0x00	R/W	[7:0] REG_TS_CTRL <7> DCHG_OVERRIDE_EN <6> DCHG_REQ
REG_BIST_CTRL1[23:16]	0x064e	0x00	R/W	[7:0] REG_CRC_CTRL
REG_BIST_CTRL1[31:24]	0x064f	0x00	R/W	[7:0] REG_ABV_INT_SET_EN
REG_FR_STMODE	0x0573	0x01	R/W	[5:0] STMODE 6'd0 : White 6'd1 : Black 6'd2 : Red 6'd3 : Green 6'd4 : Blue 6'd5 : Cyan 6'd6 : Magenta 6'd7 : Yellow 6'd8 : H gray ramp 6'd9 : V gray ramp

				6'd10 : H RGBW ramp 6'd11 : V RGBW ramp 6'd12 : Full screen check pattern 16*10 6'd13 : Full screen dot check pattern 1*1 6'd14 : Full screen dot moire pattern 6'd15 : Full screen v line moire pattern 6'd16 : Color(programmable r, g, b) 6'd17 : 1 line on, 1 line off 6'd18 : Pixel stripe(programmable r, g, b) 6'd19 : Sub pixel stripe(127) 6'd20 : Horizontal bars(1 line on, 2 line off) 6'd21 : H x-talk-1 G127 6'd22 : H x-talk-2 G127 6'd23 : H x-talk-3 G127 6'd24 : V x-talk-1 G127 6'd25 : V x-talk-2 G127 6'd26 : V x-talk-3 G127 6'd27 : HV x-talk-1 G127 6'd28 : Repeating Hramp 6'd29 : Repeating Vramp 6'd30 : Repeating tile ramp 6'd31 : on/off dot 6'd32 : pixel 2 on/2 off vertical 6'd33 : pixel 2 on/2 off horizontal 6'd34 : 4*4 pixel(12*4 subpixel) programmable 6'd35 : 3 top and bottom lines 6'd36 : Horizontal 2 colors 6'd37 : Vertical 2 colors 6'd38 : 2-dot H stripe 6'd39 : Color(programmable r, g, b) 6'd40 : Color(programmable r, g, b)

8.2 BIST Pattern Timing Generation

Register name	Offset	Initial Value	Read Write	Description
REG_ALPM_CTRL0[7:0]	0x05b0	0x01	R/W	[7:0] REG_VBCNT_INIT[7:0]
REG_ALPM_CTRL0[15:8]	0x05b1	0x00	R/W	[15:8] REG_VBCNT_INIT[15:8]
REG_ALPM_CTRL0[23:16]	0x05b2	0x00	R/W	[23:16] REG_N6_IFP
REG_ALPM_CTRL0[31:24]	0x05b3	0x00	R/W	[27] REG_VDE_DUMMY_MASK [26] REG_ALPM_CMDSEL 1:sleep 0:standby [25]REG_ALPM_IFP_EN [24] REG_ALPM_VB_EN
REG_ALPM_CTRL1[7:0]	0x05b4	0x01	R/W	[7:0] REG_N5A_VB
REG_ALPM_CTRL1[15:8]	0x05b5	0x00	R/W	[15:8] REG_N5B_VB
REG_ALPM_CTRL1[23:16]	0x05b6	0x00	R/W	[23:16] REG_N1_VB
REG_ALPM_CTRL1[31:24]	0x05b7	0x00	R/W	[31:24] REG_N7_VB
REG_ALPM_CTRL2[7:0]	0x05b8	0x01	R/W	[7:0] REG_N5A_ IFP
REG_ALPM_CTRL2[15:8]	0x05b9	0x00	R/W	[15:8] REG_N5B_ IFP
REG_ALPM_CTRL2[23:16]	0x05ba	0x00	R/W	[23:16] REG_N1_ IFP
REG_ALPM_CTRL2[31:24]	0x05bb	0x00	R/W	[31:24] REG_N7_IFP
REG_ALPM_HCNT[7:0]	0x55bc	0x10	R/W	[7:0] REG_ALPM_HCNT0[7:0]
REG_ALPM_HCNT[15:8]	0x55bd	0x00	R/W	[7:0] REG_ALPM_HCNT0[15:8]

REG_ALPM_HCNT[23:16]	0x55be	0x00	R/W	[7:0] REG_ALPM_HCNT1[7:0]
REG_ALPM_HCNT[31:24]	0x55bf	0x00	R/W	[7:0] REG_ALPM_HCNT1[15:8]
REG_BIST_RR0[7:0]	0x0584	0x00	R/W	REG_BIST_RR0[7:0]
REG_BIST_RR0[15:8]	0x0585	0x00	R/W	REG_BIST_RR0[15:8]
REG_BIST_RR0[23:16]	0x0586	0x00	R/W	REG_BIST_RR0[23:16]
REG_BIST_RR0[31:24]	0x0587	0x00	R/W	REG_BIST_RR0[31:24]
REG_BIST_RR1[7:0]	0x0588	0x00	R/W	REG_BIST_RR1[7:0]
REG_BIST_RR1[15:8]	0x0589	0x00	R/W	REG_BIST_RR1[15:8]
REG_BIST_RR1[23:16]	0x058a	0x00	R/W	REG_BIST_RR1[23:16]
REG_BIST_RR1[31:24]	0x058b	0x00	R/W	REG_BIST_RR1[31:24]
REG_BIST_RR2[7:0]	0x058c	0x00	R/W	REG_BIST_RR2[7:0]
REG_BIST_RR2[15:8]	0x058d	0x00	R/W	REG_BIST_RR2[15:8]
REG_BIST_RR2[23:16]	0x058e	0x00	R/W	REG_BIST_RR2[23:16]
REG_BIST_RR2[31:24]	0x058f	0x00	R/W	REG_BIST_RR2[31:24]
REG_BIST_RR3[7:0]	0x0590	0x00	R/W	REG_BIST_RR3[7:0]
REG_BIST_RR3[15:8]	0x0591	0x00	R/W	REG_BIST_RR3[15:8]
REG_BIST_RR3[23:16]	0x0592	0x00	R/W	REG_BIST_RR3[23:16]
REG_BIST_RR3[31:24]	0x0593	0x00	R/W	REG_BIST_RR3[31:24]
REG_BIST_RR4[7:0]	0x0594	0x00	R/W	REG_BIST_RR4[7:0]
REG_BIST_RR4[15:8	0x0595	0x00	R/W	REG_BIST_RR4[15:8]

]				
REG_BIST_RR4[23:16]	0x0596	0x00	R/W	REG_BIST_RR4[23:16]
REG_BIST_RR4[31:24]	0x0597	0x00	R/W	REG_BIST_RR4[31:24]
REG_BIST_RR5[7:0]	0x0598	0x00	R/W	REG_BIST_RR5[7:0]
REG_BIST_RR5[15:8]	0x0599	0x00	R/W	REG_BIST_RR5[15:8]
REG_BIST_RR5[23:16]	0x059a	0x00	R/W	REG_BIST_RR5[23:16]
REG_BIST_RR5[31:24]	0x059b	0x00	R/W	REG_BIST_RR5[31:24]
REG_BIST_RR6[7:0]	0x059c	0x00	R/W	REG_BIST_RR6[7:0]
REG_BIST_RR6[15:8]	0x059d	0x00	R/W	REG_BIST_RR6[15:8]
REG_BIST_RR6[23:16]	0x059e	0x00	R/W	REG_BIST_RR6[23:16]
REG_BIST_RR6[31:24]	0x059f	0x00	R/W	REG_BIST_RR6[31:24]
REG_BIST_RR7[7:0]	0x05a0	0x00	R/W	REG_BIST_RR7[7:0]
REG_BIST_RR7[15:8]	0x05a1	0x00	R/W	REG_BIST_RR7[15:8]
REG_BIST_RR7[23:16]	0x05a2	0x00	R/W	REG_BIST_RR7[23:16]
REG_BIST_RR7[31:24]	0x05a3	0x00	R/W	REG_BIST_RR7[31:24]
REG_BIST_RR_FRAMES0	0x05a4	0x00	R/W	REG_BIST_RR_FRAMES0
REG_BIST_RR_FRAMES1	0x05a5	0x00	R/W	REG_BIST_RR_FRAMES1
REG_BIST_RR_FRAMES2	0x05a6	0x00	R/W	REG_BIST_RR_FRAMES2
REG_BIST_RR_FRAMES3	0x05a7	0x00	R/W	REG_BIST_RR_FRAMES3

REG_BIST_RR_FRAMES4	0x05a8	0x00	R/W	REG_BIST_RR_FRAMES4
REG_BIST_RR_FRAMES5	0x05a9	0x00	R/W	REG_BIST_RR_FRAMES5
REG_BIST_RR_FRAMES6	0x05aa	0x00	R/W	REG_BIST_RR_FRAMES6
REG_BIST_RR_FRAMES7	0x05ab	0x00	R/W	REG_BIST_RR_FRAMES7
REG_BIST_RR_CTRL	0x05ac	0x00	R/W	[7:0] REG_BIST_RR_MASK
REG_BIST_RR_CTRL	0x05ad	0x00	R/W	[8] REG_BIST_RR_EN
REG_IFP_CTRL0	0x05c0	0x10	R/W	[6] REG_VCNT_RST_MASK [5:4] REG_IFP_ROLL_OFFSET [3:0] REG_05
REG_IFP_CTRL1[7:0]	0x05c4	0x00	R/W	[7:0] REG_IFP_START0[7:0]
REG_IFP_CTRL1[15:8]	0x05c5	0x00	R/W	[11:8] REG_IFP_START0[11:8] [15:12] REG_IFP_START1[3:0]
REG_IFP_CTRL1[23:16]	0x05c6	0x00	R/W	[23:16] REG_IFP_START1[11:4]
REG_IFP_CTRL2[7:0]	0x05c8	0x00	R/W	[7:0] REG_IFP_START2[7:0]
REG_IFP_CTRL2[15:8]	0x05c9	0x00	R/W	[11:8] REG_IFP_START2[11:8] [15:12] REG_IFP_START3[3:0]
REG_IFP_CTRL2[23:16]	0x05ca	0x00	R/W	[23:16] REG_IFP_START3[11:4]
REG_IFP_CTRL3[7:0]	0x05cc	0x00	R/W	[7:0] REG_IFP_START4[7:0]
REG_IFP_CTRL3[15:8]	0x05cd	0x00	R/W	[11:8] REG_IFP_START4[11:8] [15:12] REG_IFP_START5[3:0]
REG_IFP_CTRL3[23:16]	0x05ce	0x00	R/W	[23:16] REG_IFP_START5[11:4]
REG_IFP_CTRL4[7:0]	0x05d0	0x00	R/W	[7:0] REG_IFP_START6[7:0]

REG_IFP_CTRL4[15:8]	0x05d1	0x00	R/W	[11:8] REG_IFP_START6[11:8] [15:12] REG_IFP_START7[3:0]
REG_IFP_CTRL4[23:16]	0x05d2	0x00	R/W	[23:16] REG_IFP_START7[11:4]
REG_IFP_CTRL5[7:0]	0x05d4	0x00	R/W	[7:0] REG_IFP_START8[7:0]
REG_IFP_CTRL5[15:8]	0x05d5	0x00	R/W	[11:8] REG_IFP_START8[11:8] [15:12] REG_IFP_START9[3:0]
REG_IFP_CTRL5[23:16]	0x05d6	0x00	R/W	[23:16] REG_IFP_START9[11:4]
REG_IFP_CTRL6[7:0]	0x05d8	0x00	R/W	[7:0] REG_IFP_START10[7:0]
REG_IFP_CTRL6[15:8]	0x05d9	0x00	R/W	[11:8] REG_IFP_START10[11:8] [15:12] REG_IFP_START11[3:0]
REG_IFP_CTRL6[23:16]	0x05da	0x00	R/W	[23:16] REG_IFP_START11[11:4]
REG_IFP_CTRL7[7:0]	0x05dc	0x00	R/W	[7:0] REG_IFP_LEN0 [7:0]
REG_IFP_CTRL7[15:8]	0x05dd	0x00	R/W	[11:8] REG_IFP_LEN0 [11:8] [15:12] REG_IFP_LEN1 [3:0]
REG_IFP_CTRL7[23:16]	0x05de	0x00	R/W	[23:16] REG_IFP_LEN1 [11:4]
REG_IFP_CTRL8[7:0]	0x05e0	0x00	R/W	[7:0] REG_IFP_LEN2 [7:0]
REG_IFP_CTRL8[15:8]	0x05e1	0x00	R/W	[11:8] REG_IFP_LEN2 [11:8] [15:12] REG_IFP_LEN3 [3:0]
REG_IFP_CTRL8[23:16]	0x05e2	0x00	R/W	[23:16] REG_IFP_LEN3 [11:4]
REG_IFP_CTR9[7:0]	0x05e4	0x00	R/W	[7:0] REG_IFP_LEN4 [7:0]
REG_IFP_CTRL9[15:8]	0x05e5	0x00	R/W	[11:8] REG_IFP_LEN4 [11:8] [15:12] REG_IFP_LEN5 [3:0]
REG_IFP_CTRL9[23:16]	0x05e6	0x00	R/W	[23:16] REG_IFP_LEN5 [11:4]
REG_IFP_CTRL10[7:0]	0x05e8	0x00	R/W	[7:0] REG_IFP_LEN06[7:0]

REG_IFP_CTR10[15:8]	0x05e9	0x00	R/W	[11:8] REG_IFP_LEN6 [11:8] [15:12] REG_IFP_LEN7[3:0]
REG_IFP_CTR10[23:16]	0x05ea	0x00	R/W	[23:16] REG_IFP_LEN7[11:4]
REG_IFP_CTRL11[7:0]	0x05ec	0x00	R/W	[7:0] REG_IFP_LEN08[7:0]
REG_IFP_CTR11[15:8]	0x05ed	0x00	R/W	[11:8] REG_IFP_LEN8 [11:8] [15:12] REG_IFP_LEN9[3:0]
REG_IFP_CTR11[23:16]	0x05ef	0x00	R/W	[23:16] REG_IFP_LEN9[11:4]
REG_IFP_CTRL12[7:0]	0x05f0	0x00	R/W	[7:0] REG_IFP_LEN10[7:0]
REG_IFP_CTR12[15:8]	0x05f1	0x00	R/W	[11:8] REG_IFP_LEN10[11:8] [15:12] REG_IFP_LEN11[3:0]
REG_IFP_CTR12[23:16]	0x05f2	0x00	R/W	[23:16] REG_IFP_LEN11[11:4]
REG_IFP_CTRL13[7:0]	0x05f4	0x00	R/W	[7:0] REG_IFP_MASK[7:0]
REG_IFP_CTR13[15:8]	0x05f5	0x00	R/W	[11:8] REG_IFP_MASK [11:8] [15:12] REG_IFP_ROLL_FRAMES [3:0]
REG_IFP_CTR13[23:16]	0x05f6	0x00	R/W	[23:16] REG_IFP_ROLL_FRAMES [11:4]
REG_IFP_CTR13[31:24]	0x05f7	0x00	R/W	[27:24] REG_IFP_ROLL_FRAMES [15:12]
REG_IFP_CTRL14[7:0]	0x05f8	0x00	R/W	[7:0] REG_IFP_ROLL_LINES [7:0]
REG_IFP_CTR14[15:8]	0x05f9	0x00	R/W	[11:8] REG_IFP_ROLL_LINES [11:8] [15:12] REG_IFP_ROLL_MAX[3:0]
REG_IFP_CTR14[23:16]	0x05fa	0x00	R/W	[23:16] REG_IFP_ROLL_MAX [11:4]
REG_IFP_CTR14[31:24]	0x05fb	0x00	R/W	[27:24] REG_IFP_ROLL_MAX [15:12]
REG_FR_HTOTALHSTAT[7:0]	0x05fc	0x50	R/W	[7:0] REG_FR_HTOTAL[7:0]
REG_FR_HTOTALH	0x05fd	0x0c	R/W	[15:8] REG_FR_HTOTAL[15:8]

STAT[15:8]				
REG_FR_HTOTALHSTAT[23:16]	0x05fe	0x48	R/W	[23:1 6] REG_FR_HSTAT[7:0]
REG_FR_HTOTALHSTAT[31:24]	0x05ff	0x00	R/W	[31:24]] REG_FR_HSTAT[15:8]
REG_FR_HWIDTHHSPHSW [7:0]	0x0600	0x00	R/W	[7:0] REG_FR_HWIDTH [7:0]
REG_FR_HWIDTHHSPHSW [15:8]	0x0601	0x0c	R/W	[15:8] REG_FR_HWIDTH [15:8]
REG_FR_HWIDTHHSPHSW [23:16]	0x0602	0x00	R/W	[23:1 6] REG_FR_HSPHSW [7:0]
REG_FR_HWIDTHHSPHSW [31:24]	0x0603	0x00	R/W	[31:24]] REG_FR_HSPHSW [15:8]
REG_FR_DUMMYVTOTAL[7:0]	0x0604	0x00	R/W	[7:0] REG_FR_FROND_DUMMY
REG_FR_DUMMYVTOTAL[15:8]	0x0605	0x00	R/W	[15:8] REG_FR_END_DUMMY
REG_FR_DUMMYVTOTAL[23:16]	0x0606	0xf1	R/W	[23:16] REG_FR_VTOTAL0[7:0]
REG_FR_DUMMYVTOTAL [31:24]	0x0607	0x07	R/W	[30:24] REG_FR_VTOTAL0[14:8] [31]:FR_VTRDY
REG_FR_VSTARTVHEIGHT[7:0]	0x0608	0x0e	R/W	[7:0] REG_FR_VSTART[7:0]
REG_FR_VSTARTVHEIGHT[15:8]	0x0609	0x00	R/W	[15:8] REG_FR_VSTART[15:8]
REG_FR_VSTARTVHEIGHT[23:16]	0x060a	0x80	R/W	[23:16] REG_FR_VHEIGHT[23:16]
REG_FR_VSTARTVHEIGHT[31:24]	0x060b	0x07	R/W	[31:24] REG_FR_VHEIGHT[31:24]
REG_FR_VSPVSWHDIV3 [7:0]	0x060c	0x00	R/W	[7:0] REG_FR_VSPVSW [7:0]
REG_FR_VSPVSWHDIV3 [15:8]	0x060d	0x00	R/W	[15:8] REG_FR_VSPVSW [15:8]
REG_FR_VSPVSWHDIV3 [23:16]	0x060e	0x0a	R/W	[23:16] REG_HWIDTH_DIV3 [23:16]

REG_FR_VSPVSWH DIV3 [31:24]	0x060f	0x00	R/W	[31:24] REG_HWIDTH_DIV3 [31:24]
REG_STRIP_VDIV3 [7:0]	0x0610	0x0f	R/W	[7:0] REG_STRIP_HEIGHT0
REG_STRIP_VDIV3 [15:8]	0x0611	0x1f	R/W	[15:8] REG_STRIP_HEIGHT1
REG_STRIP_VDIV3 [23:16]	0x0612	0x00	R/W	[23:16] REG_FR_VTOTAL0[7:0]
REG_STRIP_VDIV3 [31:24]	0x0613	0x00	R/W	[31:24] REG_VHEIGHT_DIV3
REG_STRIP_GAP [7:0]	0x0614	0x0f	R/W	[7:0] REG_STRIP_WIDTH
REG_STRIP_GAP [15:8]	0x0615	0x07	R/W	[15:8] REG_GAP_WIDTH
REG_STRIP_GAP [23:16]	0x0616	0xf0	R/W	[23:16] REG_STRIP_HSTART
REG_STRIP_GAP [31:24]	0x0617	0x00	R/W	[31:24] REG_STRIP_HSTART
REG_STEP [7:0]	0x0618	0x44	R/W	[7:0] REG_RAMP_STEP
REG_STEP [15:8]	0x0619	0x00	R/W	[15:8] REG_HRAMP_STEP
REG_STEP [23:16]	0x061a	0x00	R/W	[23:16] REG_VRAMP_STEP
REG_TRAMP_VSYN C[7:0]	0x061c	0x00	R/W	[7:0] REG_TRAMP_STEP[7:0]
REG_TRAMP_VSYN C[15:8]	0x061d	0x00	R/W	[15:8] REG_TRAMP_STEP[15:8]
REG_TRAMP_VSYN C[23:16]	0x061e			[23:16] REG_VSYNC_SEL[7:0]
REG_FR_DOT [7:0]	0x0620	0x01	R/W	[7:0] REG_FR_MISC
REG_FR_DOT [15:8]	0x0621	0x25	R/W	[15:8] REG_FR_DOT0
REG_FR_DOT [23:16]	0x0622	0x12	R/W	[23:16] REG_FR_DOT1
REG_FR_DOT [31:24]	0x0623	0x00	R/W	[31:24] REG_HCNT_RST_VALUE

9. TCON Timing Control

9.1 TCON Signal Generation (TPIO)

includes several configurable TCON Programmable I/O ports. In general, these ports will be used for special functions required for the panel Gate Driver interface – either used directly for COG Gate Drivers, or via a Programmable Level Shifter (PLS) for Integrated Row Driver (IRD) panels. The TPIOs can also be used for various other external devices, which require signals that are in sync with the panel line timing.

uses a programmable scheme to generate TCON control signals. Each control signal has several dedicated programming registers to control the start and the end of the signal, as well as the pulse polarity and toggle.

The following table illustrates the control signal mapping and their definitions.

Pin name	GDIC	PLSI	APLS	Alternate Function
TPIO0	LS_INIT	GSP	LS_SOL	
TPIO1	LS_CLK	GSC	LS_CLK	
TPIO2	LS_EN	GOE	LS_EN	
TPIO3	LS_VBE	GOH	LS_VBE	
TPIO4	LS_TT	GOE/VGL_EN	LS_TT	SWCLK
TPIO5		GOH/IFP_GDIC		SWDIO
TPIO6	FLK			UART_TXD
TPIO7	DCHG_N_IN			UART_RXD
TPIO8	TSI_FRAME_POL(IN/OUT)			
TPIO9	LSYNC_OUT/TSI_VDE(IN/OUT)			

9.2 Register Definitions

The TPIO signals are programmed by following registers.

Register name	Offset	Initial Value	Read Write	Description
RGTSIG0_HS1[7:0]	0x1400	0x00	R/W	HS1[7:0]
RGTSIG0_HS1[15:8]	0x1401	0x00	R/W	HS1[15:8]
RGTSIG0_HE1[7:0]	0x1402	0x00	R/W	HE1[7:0]
RGTSIG0_HE1[15:8]	0x1403	0x00	R/W	HE1[15:8]
RGTSIG0_HS2[7:0]	0x1404	0x00	R/W	HS2[7:0]
RGTSIG0_HS2[15:8]	0x1405	0x00	R/W	HS2[15:8]
RGTSIG0_HE2[7:0]	0x1406	0x00	R/W	HE2[7:0]

RGTSIG0_HE2[15:8]	0x1407	0x00	R/W	HE2[15:8]
RGTSIG0_VS[7:0]	0x1408	0x00	R/W	VS[7:0]
RGTSIG0_VS[15:8]	0x1409	0x00	R/W	VS[15:8]
RGTSIG0_VS[23:16]	0x140a	0x00	R/W	VS[23:16]
RGTSIG0_VE[7:0]	0x140b	0x00	R/W	VE[7:0]
RGTSIG_VE[15:8]	0x140c	0x00	R/W	VE[15:8]
RGTSIG_VE[23:16]	0x140d	0x00	R/W	VE[23:16]
RGTSIG0_VLVH	0x140e	0x00	R/W	Bit3:0 VL Bit7:4 VH
RGTSIG0_CTRL[7:0]	0x140f	0x00	R/W	Bit7 RGMASK_EN: 1= masking output during BIST and fail-safe mode Bit6 SINV(signal inversion) Bit5 FIELD_EN Bit4 Reserved Bit3 RGVDE_DUMMY_MASK_EN Bit2 RGJT_EN 1=jitter injection enable (HDE from Jitter) Bit1 RGIFP_DUMMY_MASK_EN 1= no toggle during IS_DUMMY Bit0 RGIFP_MASK_EN 1= no toggle during IFP
RGTSIG0_CTRL[15:8]	0x1410	0x00	R/W	Bit 7: RGLB4ODD When pulse width is different between odd and even line, the bit is a option for odd/even line boundary definition. 1=HDE rising edge is line boundary 0=HDE falling edge is line boundary Bit 6: VEN_SYNC_DIS 1= V_EN sync with HSP. need set to 1 to avoid glitch. when signal is not cross line boundary Bit 5: RGFR60HZ_EN(no use) Bit 4: RGVB_VALUE (the tcon value

				during VCNT > VE and VCNT < VS) Bit 3: RGVWIDTH_EN 1: toggle once per frame, high from VS to VE Bit 2: RGODD_DIFF_EN The width of odd line and even line is different. The width of even line is related with HS2/HE2. Bit 1: RGHS2HE2_EN 1= need set 1 when two pulse per line Bit 0: RGWIDTH_EN 1= HS + WIDTH scheme enable, RGHE is used as RGHWIDTH
RGTSIG0_MASK_FNUM<7:0>	0x1411	0x00	R/W	Bit3:0= MASK_FNUM_fail Fail mode Bit7:4= MASK_FNUM_NORM
RGTSIG1_HS1[7:0]	0x1412	0x00	R/W	HS1[7:0]
RGTSIG1_HS1[15:8]	0x1413	0x00	R/W	HS1[15:8]
RGTSIG1_HE1[7:0]	0x1414	0x00	R/W	HE1[7:0]
RGTSIG1_HE1[15:8]	0x1415	0x00	R/W	HE1[15:8]
RGTSIG1_HS2[7:0]	0x1416	0x00	R/W	HS2[7:0]
RGTSIG1_HS2[15:8]	0x1417	0x00	R/W	HS2[15:8]
RGTSIG1_HE2[7:0]	0x1418	0x00	R/W	HE2[7:0]
RGTSIG1_HE2[15:8]	0x1419	0x00	R/W	HE2[15:8]
RGTSIG1_VS[7:0]	0x141a	0x00	R/W	VS[7:0]
RGTSIG1_VS[15:8]	0x141b	0x00	R/W	VS[15:8]
RGTSIG1_VS[23:16]	0x141c	0x00	R/W	VS[23:16]
RGTSIG1_VE[7:0]	0x141d	0x00	R/W	VE[7:0]
RGTSIG1_VE[15:8]	0x141e	0x00	R/W	VE[15:8]
RGTSIG1_VE[23:16]	0x141f	0x00	R/W	VE[23:16]
RGTSIG1_VLVH	0x1420	0x00	R/W	Bit3:0 VL Bit7:4 VH
RGTSIG1_CTRL[7:0]	0x1421	0x00	R/W	Bit7 RGMASK_EN: 1= masking output during BIST and fail-safe mode Bit6 SINV(signal inversion)

				Bit5 FIELD_EN Bit4 Reserved Bit3 RGVDE_DUMMY_MASK_EN Bit2 RGJT_EN 1=jitter injection enable (HDE from Jitter) Bit1 RGIFP_DUMMY_MASK_EN 1= no toggle during IS_DUMMY Bit0 RGIFP_MASK_EN 1= no toggle during IFP
RGTSIG1_CTRL[15:8]	0x1422	0x00	R/W	Bit 7: RGLB4ODD When pulse width is different between odd and even line, the bit is a option for odd/even line boundary definition. 1=HDE rising edge is line boundary 0=HDE falling edge is line boundary Bit 6: VEN_SYNC_DIS 1= V_EN sync with HSP. need set to 1 to avoid glitch. when signal is not cross line boundary Bit 5: RGFR60HZ_EN(no use) Bit 4: RGVB_VALUE (the tcon value during VCNT > VE and VCNT < VS) Bit 3: RGVWIDTH_EN 1: toggle once per frame, high from VS to VE Bit 2: RGODD_DIFF_EN The width of odd line and even line is different. The width of even line is related with HS2/HE2. Bit 1: RGHS2HE2_EN 1= need set 1 when two pulse per

				line Bit 0: RGWIDTH_EN 1= HS + WIDTH scheme enable, RGHE is used as RGHWIDTH
RGTSIG1_MASK_FNUM<7:0>	0x1423	0x00	R/W	Bit3:0= MASK_FNUM_fail Fail mode Bit7:4= MASK_FNUM_NORM
RGTSIG2_HS1[7:0]	0x1424	0x00	R/W	HS1[7:0]
RGTSIG2_HS1[15:8]	0x1425	0x00	R/W	HS1[15:8]
RGTSIG2_HE1[7:0]	0x1426	0x00	R/W	HE1[7:0]
RGTSIG2_HE1[15:8]	0x1427	0x00	R/W	HE1[15:8]
RGTSIG2_HS2[7:0]	0x1428	0x00	R/W	HS2[7:0]
RGTSIG2_HS2[15:8]	0x1429	0x00	R/W	HS2[15:8]
RGTSIG2_HE2[7:0]	0x142a	0x00	R/W	HE2[7:0]
RGTSIG2_HE2[15:8]	0x142b	0x00	R/W	HE2[15:8]
RGTSIG2_VS[7:0]	0x1434	0x00	R/W	VS[7:0]
RGTSIG2_VS[15:8]	0x1435	0x00	R/W	VS[15:8]
RGTSIG2_VS[23:16]	0x1436	0x00	R/W	VS[23:16]
RGTSIG2_VE[7:0]	0x1437	0x00	R/W	VE[7:0]
RGTSIG2_VE[15:8]	0x1438	0x00	R/W	VE[15:8]
RGTSIG2_VE[23:16]	0x1439	0x00	R/W	VE[23:16]
RGTSIG2_VLVH	0x143a	0x00	R/W	Bit3:0 VL Bit7:4 VH
RGTSIG2_CTRL[7:0]	0x143b	0x00	R/W	Bit7 RGMASK_EN: 1= masking output during BIST and fail-safe mode Bit6 SINV(signal inversion) Bit5 FIELD_EN Bit4 Reserved Bit3 RGVDE_DUMMY_MASK_EN Bit2 RGJT_EN 1=jitter injection enable (HDE from Jitter) Bit1 RGIFP_DUMMY_MASK_EN 1= no toggle during IS_DUMMY Bit0 RGIFP_MASK_EN 1= no toggle during IFP
RGTSIG2_CTRL[15:8]	0x143c	0x00	R/W	Bit 7: RGLB4ODD When pulse width is different

				between odd and even line, the bit is a option for odd/even line boundary definition. 1=HDE rising edge is line boundary 0=HDE falling edge is line boundary Bit 6: VEN_SYNC_DIS 1= V_EN sync with HSP. need set to 1 to avoid glitch. when signal is not cross line boundary Bit 5: RGFR60HZ_EN(no use) Bit 4: RGVB_VALUE (the tcon value during VCNT > VE and VCNT < VS) Bit 3: RGVWIDTH_EN 1: toggle once per frame, high from VS to VE Bit 2: RGODD_DIFF_EN The width of odd line and even line is different. The width of even line is related with HS2/HE2. Bit 1: RGHS2HE2_EN 1= need set 1 when two pulse per line Bit 0: RGWIDTH_EN 1= HS + WIDTH scheme enable, RGHE is used as RGHWIDTH
RGTSIG2_MASK_FNUM<7:0>	0x143d	0x00	R/W	Bit3:0= MASK_FNUM_fail Fail mode Bit7:4= MASK_FNUM_NORM
RGTSIG3_HS1[7:0]	0x143e	0x00	R/W	HS1[7:0]
RGTSIG3_HS1[15:8]	0x143f	0x00	R/W	HS1[15:8]
RGTSIG3_HE1[7:0]	0x1440	0x00	R/W	HE1[7:0]
RGTSIG3_HE1[15:8]	0x1441	0x00	R/W	HE1[15:8]
RGTSIG3_HS2[7:0]	0x1442	0x00	R/W	HS2[7:0]

RGTSIG3_HS2[15:8]	0x1443	0x00	R/W	HS2[15:8]
RGTSIG3_HE2[7:0]	0x1444	0x00	R/W	HE2[7:0]
RGTSIG3_HE2[15:8]	0x1445	0x00	R/W	HE2[15:8]
RGTSIG3_VS[7:0]	0x1446	0x00	R/W	VS[7:0]
RGTSIG3_VS[15:8]	0x1447	0x00	R/W	VS[15:8]
RGTSIG3_VS[23:16]	0x1448	0x00	R/W	VS[23:16]
RGTSIG3_VE[7:0]	0x1449	0x00	R/W	VE[7:0]
RGTSIG3_VE[15:8]	0x144a	0x00	R/W	VE[15:8]
RGTSIG3_VE[23:16]	0x144b	0x00	R/W	VE[23:16]
RGTSIG3_VLVH	0x144c	0x00	R/W	Bit3:0 VL Bit7:4 VH
RGTSIG3_CTRL[7:0]	0x144d	0x00	R/W	Bit7 RGMASK_EN: 1= masking output during BIST and fail-safe mode Bit6 SINV(signal inversion) Bit5 FIELD_EN Bit4 Reserved Bit3 RGVDE_DUMMY_MASK_EN Bit2 RGJT_EN 1=jitter injection enable (HDE from Jitter) Bit1 RGIFP_DUMMY_MASK_EN 1= no toggle during IS_DUMMY Bit0 RGIFP_MASK_EN 1= no toggle during IFP
RGTSIG3_CTRL[15:8]	0x144e	0x00	R/W	Bit 7: RGLB4ODD When pulse width is different between odd and even line, the bit is a option for odd/even line boundary definition. 1=HDE rising edge is line boundary 0=HDE falling edge is line boundary Bit 6: VEN_SYNC_DIS 1= V_EN sync with HSP. need set to 1 to avoid glitch. when signal is not cross line boundary Bit 5: RGFR60HZ_EN(no use)

				Bit 4: RGVB_VALUE (the tcon value during VCNT > VE and VCNT < VS) Bit 3: RGVWIDTH_EN 1: toggle once per frame, high from VS to VE Bit 2: RGODD_DIFF_EN The width of odd line and even line is different. The width of even line is related with HS2/HE2. Bit 1: RGHS2HE2_EN 1= need set 1 when two pulse per line Bit 0: RGWIDTH_EN 1= HS + WIDTH scheme enable, RGHE is used as RGHWIDTH
RGTSIG3_MASK_FNUM <7:0>	0x144f	0x00	R/W	Bit3:0= MASK_FNUM_fail Fail mode Bit7:4= MASK_FNUM_NORM
RGTSIG4_HS1[7:0]	0x1450	0x00	R/W	HS1[7:0]
RGTSIG4_HS1[15:8]	0x1451	0x00	R/W	HS1[15:8]
RGTSIG4_HE1[7:0]	0x1452	0x00	R/W	HE1[7:0]
RGTSIG4_HE1[15:8]	0x1453	0x00	R/W	HE1[15:8]
RGTSIG4_HS2[7:0]	0x1454	0x00	R/W	HS2[7:0]
RGTSIG4_HS2[15:8]	0x1455	0x00	R/W	HS2[15:8]
RGTSIG4_HE2[7:0]	0x1456	0x00	R/W	HE2[7:0]
RGTSIG4_HE2[15:8]	0x1457	0x00	R/W	HE2[15:8]
RGTSIG4_VS[7:0]	0x1458	0x00	R/W	VS[7:0]
RGTSIG4_VS[15:8]	0x1459	0x00	R/W	VS[15:8]
RGTSIG4_VS[23:16]	0x145a	0x00	R/W	VS[23:16]
RGTSIG4_VE[7:0]	0x145b	0x00	R/W	VE[7:0]
RGTSIG4_VE[15:8]	0x145c	0x00	R/W	VE[15:8]
RGTSIG4_VE[23:16]	0x145d	0x00	R/W	VE[23:16]
RGTSIG4_VLVH	0x145e	0x00	R/W	Bit3:0 VL Bit7:4 VH
RGTSIG4_CTRL[7:0]	0x145f	0x00	R/W	Bit7 RGMASK_EN: 1= masking output during BIST and

				fail-safe mode Bit6 SINV(signal inversion) Bit5 FIELD_EN Bit4 Reserved Bit3 RGVDE_DUMMY_MASK_EN Bit2 RGJT_EN 1=jitter injection enable (HDE from Jitter) Bit1 RGIFP_DUMMY_MASK_EN 1= no toggle during IS_DUMMY Bit0 RGIFP_MASK_EN 1= no toggle during IFP
RGTSIG4_CTRL[15:8]	0x1460	0x00	R/W	Bit 7: RGLB4ODD When pulse width is different between odd and even line, the bit is a option for odd/even line boundary definition. 1=HDE rising edge is line boundary 0=HDE falling edge is line boundary Bit 6: VEN_SYNC_DIS 1= V_EN sync with HSP. need set to 1 to avoid glitch. when signal is not cross line boundary Bit 5: RGFR60HZ_EN(no use) Bit 4: RGVB_VALUE (the tcon value during VCNT > VE and VCNT < VS) Bit 3: RGVWIDTH_EN 1: toggle once per frame, high from VS to VE Bit 2: RGODD_DIFF_EN The width of odd line and even line is different. The width of even line is related with HS2/HE2. Bit 1:

				RGHS2HE2_EN 1= need set 1 when two pulse per line Bit 0: RGWIDTH_EN 1= HS + WIDTH scheme enable, RGHE is used as RGHWIDTH
RGTSIG4_MASK_FNUM<7:0>	0x1461	0x00	R/W	Bit3:0= MASK_FNUM_fail Fail mode Bit7:4= MASK_FNUM_NORM
RGTSIG5_HS1[7:0]	0x1462	0x00	R/W	HS1[7:0]
RGTSIG5_HS1[15:8]	0x1463	0x00	R/W	HS1[15:8]
RGTSIG5_HE1[7:0]	0x1464	0x00	R/W	HE1[7:0]
RGTSIG5_HE1[15:8]	0x1465	0x00	R/W	HE1[15:8]
RGTSIG5_HS2[7:0]	0x1466	0x00	R/W	HS2[7:0]
RGTSIG5_HS2[15:8]	0x1467	0x00	R/W	HS2[15:8]
RGTSIG5_HE2[7:0]	0x1468	0x00	R/W	HE2[7:0]
RGTSIG5_HE2[15:8]	0x1469	0x00	R/W	HE2[15:8]
RGTSIG5_VS[7:0]	0x146a	0x00	R/W	VS[7:0]
RGTSIG5_VS[15:8]	0x146b	0x00	R/W	VS[15:8]
RGTSIG5_VS[23:16]	0x146c	0x00	R/W	VS[23:16]
RGTSIG5_VE[7:0]	0x146d	0x00	R/W	VE[7:0]
RGTSIG5_VE[15:8]	0x146e	0x00	R/W	VE[15:8]
RGTSIG5_VE[23:16]	0x146f	0x00	R/W	VE[23:16]
RGTSIG5_VLVH	0x1470	0x00	R/W	Bit3:0 VL Bit7:4 VH
RGTSIG5_CTRL[7:0]	0x1471	0x00	R/W	Bit7 RGMASK_EN: 1= masking output during BIST and fail-safe mode Bit6 SINV(signal inversion) Bit5 FIELD_EN Bit4 Reserved Bit3 RGVDE_DUMMY_MASK_EN Bit2 RGJT_EN 1=jitter injection enable (HDE from Jitter) Bit1 RGIFP_DUMMY_MASK_EN 1= no toggle during IS_DUMMY Bit0 RGIFP_MASK_EN 1= no toggle during IFP
RGTSIG5_CTRL[15:8]	0x1472	0x00	R/W	Bit 7:

				RGLB4ODD When pulse width is different between odd and even line, the bit is a option for odd/even line boundary definition. 1=HDE rising edge is line boundary 0=HDE falling edge is line boundary Bit 6: VEN_SYNC_DIS 1= V_EN sync with HSP. need set to 1 to avoid glitch. when signal is not cross line boundary Bit 5: RGFR60HZ_EN(no use) Bit 4: RGVB_VALUE (the tcon value during VCNT > VE and VCNT < VS) Bit 3: RGVWIDTH_EN 1: toggle once per frame, high from VS to VE Bit 2: RGODD_DIFF_EN The width of odd line and even line is different. The width of even line is related with HS2/HE2. Bit 1: RGHS2HE2_EN 1= need set 1 when two pulse per line Bit 0: RGWIDTH_EN 1= HS + WIDTH scheme enable, RGHE is used as RGHWIDTH
RGTSIG5_MASK_FNUM<7:0>	0x1473	0x00	R/W	Bit3:0= MASK_FNUM_fail Fail mode Bit7:4= MASK_FNUM_NORM
RGTSIG6_HS1[7:0]	0x1474	0x00	R/W	HS1[7:0]
RGTSIG6_HS1[15:8]	0x1475	0x00	R/W	HS1[15:8]
RGTSIG6_HE1[7:0]	0x1476	0x00	R/W	HE1[7:0]

RGTSIG6_HE1[15:8]	0x1477	0x00	R/W	HE1[15:8]
RGTSIG6_HS2[7:0]	0x1478	0x00	R/W	HS2[7:0]
RGTSIG6_HS2[15:8]	0x1479	0x00	R/W	HS2[15:8]
RGTSIG6_HE2[7:0]	0x147a	0x00	R/W	HE2[7:0]
RGTSIG6_HE2[15:8]	0x147b	0x00	R/W	HE2[15:8]
RGTSIG6_VS[7:0]	0x147c	0x00	R/W	VS[7:0]
RGTSIG6_VS[15:8]	0x147d	0x00	R/W	VS[15:8]
RGTSIG6_VS[23:16]	0x147e	0x00	R/W	VS[23:16]
RGTSIG6_VE[7:0]	0x147f	0x00	R/W	VE[7:0]
RGTSIG6_VE[15:8]	0x1480	0x00	R/W	VE[15:8]
RGTSIG6_VE[23:16]	0x1481	0x00	R/W	VE[23:16]
RGTSIG6_VLVH	0x1482	0x00	R/W	Bit3:0 VL Bit7:4 VH
RGTSIG6_CTRL[7:0]	0x1483	0x00	R/W	Bit7 RGMASK_EN: 1= masking output during BIST and fail-safe mode Bit6 SINV(signal inversion) Bit5 FIELD_EN Bit4 Reserved Bit3 RGVDE_DUMMY_MASK_EN Bit2 RGJT_EN 1=jitter injection enable (HDE from Jitter) Bit1 RGIFP_DUMMY_MASK_EN 1= no toggle during IS_DUMMY Bit0 RGIFP_MASK_EN 1= no toggle during IFP
RGTSIG6_CTRL[15:8]	0x1484	0x00	R/W	Bit 7: RGLB4ODD When pulse width is different between odd and even line, the bit is a option for odd/even line boundary definition. 1=HDE rising edge is line boundary 0=HDE falling edge is line boundary Bit 6: VEN_SYNC_DIS 1= V_EN sync with HSP. need set to 1 to avoid glitch. when signal is not cross line boundary

				Bit 5: RGFR60HZ_EN(no use) Bit 4: RGVB_VALUE (the tcon value during VCNT > VE and VCNT < VS) Bit 3: RGVWIDTH_EN 1: toggle once per frame, high from VS to VE Bit 2: RGODD_DIFF_EN The width of odd line and even line is different. The width of even line is related with HS2/HE2. Bit 1: RGHS2HE2_EN 1= need set 1 when two pulse per line Bit 0: RGWIDTH_EN 1= HS + WIDTH scheme enable, RGHE is used as RGHWIDTH
RGTSIG6_MASK_FNUM<7:0>	0x1485	0x00	R/W	Bit3:0= MASK_FNUM_fail Fail mode Bit7:4= MASK_FNUM_NORM
RGTSIG7_HS1[7:0]	0x1486	0x00	R/W	HS1[7:0]
RGTSIG7_HS1[15:8]	0x1487	0x00	R/W	HS1[15:8]
RGTSIG7_HE1[7:0]	0x1488	0x00	R/W	HE1[7:0]
RGTSIG7_HE1[15:8]	0x1489	0x00	R/W	HE1[15:8]
RGTSIG7_HS2[7:0]	0x148a	0x00	R/W	HS2[7:0]
RGTSIG7_HS2[15:8]	0x148b	0x00	R/W	HS2[15:8]
RGTSIG7_HE2[7:0]	0x148c	0x00	R/W	HE2[7:0]
RGTSIG7_HE2[15:8]	0x148d	0x00	R/W	HE2[15:8]
RGTSIG7_VS[7:0]	0x148e	0x00	R/W	VS[7:0]
RGTSIG7_VS[15:8]	0x148f	0x00	R/W	VS[15:8]
RGTSIG7_VS[23:16]	0x1490	0x00	R/W	VS[23:16]
RGTSIG7_VE[7:0]	0x1491	0x00	R/W	VE[7:0]
RGTSIG7_VE[15:8]	0x1492	0x00	R/W	VE[15:8]
RGTSIG7_VE[23:16]	0x1493	0x00	R/W	VE[23:16]
RGTSIG7_VLVH	0x1494	0x00	R/W	Bit3:0 VL Bit7:4 VH

RGTSIG7_CTRL[7:0]	0x1495	0x00	R/W	Bit7 RGMASK_EN: 1= masking output during BIST and fail-safe mode Bit6 SINV(signal inversion) Bit5 FIELD_EN Bit4 Reserved Bit3 RGVDE_DUMMY_MASK_EN Bit2 RGJT_EN 1=jitter injection enable (HDE from Jitter) Bit1 RGIFP_DUMMY_MASK_EN 1= no toggle during IS_DUMMY Bit0 RGIFP_MASK_EN 1= no toggle during IFP
RGTSIG7_CTRL[15:8]	0x1496	0x00	R/W	Bit 7: RGLB4ODD When pulse width is different between odd and even line, the bit is a option for odd/even line boundary definition. 1=HDE rising edge is line boundary 0=HDE falling edge is line boundary Bit 6: VEN_SYNC_DIS 1= V_EN sync with HSP. need set to 1 to avoid glitch. when signal is not cross line boundary Bit 5: RGFR60HZ_EN(no use) Bit 4: RGVB_VALUE (the tcon value during VCNT > VE and VCNT < VS) Bit 3: RGVWIDTH_EN 1: toggle once per frame, high from VS to VE Bit 2: RGODD_DIFF_EN The width of odd line and even line is different. The width of even

				line is related with HS2/HE2. Bit 1: RGHS2HE2_EN 1= need set 1 when two pulse per line Bit 0: RGWIDTH_EN 1= HS + WIDTH scheme enable, RGHE is used as RGHWIDTH
RGTSIG7_MASK_FNUM<7:0>	0x1497	0x00	R/W	Bit3:0= MASK_FNUM_fail Fail mode Bit7:4= MASK_FNUM_NORM
RGTSIG8_HS1[7:0]	0x1498	0x00	R/W	HS1[7:0]
RGTSIG8_HS1[15:8]	0x1499	0x00	R/W	HS1[15:8]
RGTSIG8_HE1[7:0]	0x149a	0x00	R/W	HE1[7:0]
RGTSIG8_HE1[15:8]	0x149b	0x00	R/W	HE1[15:8]
RGTSIG8_HS2[7:0]	0x149c	0x00	R/W	HS2[7:0]
RGTSIG8_HS2[15:8]	0x149d	0x00	R/W	HS2[15:8]
RGTSIG8_HE2[7:0]	0x149e	0x00	R/W	HE2[7:0]
RGTSIG8_HE2[15:8]	0x149f	0x00	R/W	HE2[15:8]
RGTSIG8_VS[7:0]	0x14a0	0x00	R/W	VS[7:0]
RGTSIG8_VS[15:8]	0x14a1	0x00	R/W	VS[15:8]
RGTSIG8_VS[23:16]	0x14a2	0x00	R/W	VS[23:16]
RGTSIG8_VE[7:0]	0x14a3	0x00	R/W	VE[7:0]
RGTSIG8_VE[15:8]	0x14a4	0x00	R/W	VE[15:8]
RGTSIG8_VE[23:16]	0x14a5	0x00	R/W	VE[23:16]
RGTSIG8_VLVH	0x14a6	0x00	R/W	Bit3:0 VL Bit7:4 VH
RGTSIG8_CTRL[7:0]	0x14a7	0x00	R/W	Bit7 RGMASK_EN: 1= masking output during BIST and fail-safe mode Bit6 SINV(signal inversion) Bit5 FIELD_EN Bit4 Reserved Bit3 RGVDE_DUMMY_MASK_EN Bit2 RGJT_EN 1=jitter injection enable (HDE from Jitter) Bit1 RGIFP_DUMMY_MASK_EN 1= no toggle during IS_DUMMY Bit0 RGIFP_MASK_EN

				1= no toggle during IFP
RGTSIG8_CTRL[15:8]	0x14a8	0x00	R/W	Bit 7: RGLB4ODD When pulse width is different between odd and even line, the bit is a option for odd/even line boundary definition. 1=HDE rising edge is line boundary 0=HDE falling edge is line boundary Bit 6: VEN_SYNC_DIS 1= V_EN sync with HSP. need set to 1 to avoid glitch. when signal is not cross line boundary Bit 5: RGFR60HZ_EN(no use) Bit 4: RGVB_VALUE (the tcon value during VCNT > VE and VCNT < VS) Bit 3: RGVWIDTH_EN 1: toggle once per frame, high from VS to VE Bit 2: RGODD_DIFF_EN The width of odd line and even line is different. The width of even line is related with HS2/HE2. Bit 1: RGHS2HE2_EN 1= need set 1 when two pulse per line Bit 0: RGWIDTH_EN 1= HS + WIDTH scheme enable, RGHE is used as RGHWIDTH
RGTSIG8_MASK_FNUM<7:0>	0x14a9	0x00	R/W	Bit3:0= MASK_FNUM_fail Fail mode Bit7:4= MASK_FNUM_NORM
RGTSIG9_HS1[7:0]	0x14aa	0x00	R/W	HS1[7:0]

RGTSIG9_HS1[15:8]	0x14ab	0x00	R/W	HS1[15:8]
RGTSIG9_HE1[7:0]	0x14ac	0x00	R/W	HE1[7:0]
RGTSIG9_HE1[15:8]	0x14ad	0x00	R/W	HE1[15:8]
RGTSIG9_HS2[7:0]	0x14ae	0x00	R/W	HS2[7:0]
RGTSIG9_HS2[15:8]	0x14af	0x00	R/W	HS2[15:8]
RGTSIG9_HE2[7:0]	0x14b0	0x00	R/W	HE2[7:0]
RGTSIG9_HE2[15:8]	0x14b1	0x00	R/W	HE2[15:8]
RGTSIG9_VS[7:0]	0x14b2	0x00	R/W	VS[7:0]
RGTSIG9_VS[15:8]	0x14b3	0x00	R/W	VS[15:8]
RGTSIG9_VS[23:16]	0x14b4	0x00	R/W	VS[23:16]
RGTSIG9_VE[7:0]	0x14b5	0x00	R/W	VE[7:0]
RGTSIG9_VE[15:8]	0x14b6	0x00	R/W	VE[15:8]
RGTSIG9_VE[23:16]	0x14b7	0x00	R/W	VE[23:16]
RGTSIG9_VLVH	0x14b8	0x00	R/W	Bit3:0 VL Bit7:4 VH
RGTSIG9_CTRL[7:0]	0x14b9	0x00	R/W	Bit7 RGMASK_EN: 1= masking output during BIST and fail-safe mode Bit6 SINV(signal inversion) Bit5 FIELD_EN Bit4 Reserved Bit3 RGVDE_DUMMY_MASK_EN Bit2 RGJT_EN 1=jitter injection enable (HDE from Jitter) Bit1 RGIFP_DUMMY_MASK_EN 1= no toggle during IS_DUMMY Bit0 RGIFP_MASK_EN 1= no toggle during IFP
RGTSIG9_CTRL[15:8]	0x14ba	0x00	R/W	Bit 7: RGLB4ODD When pulse width is different between odd and even line, the bit is a option for odd/even line boundary definition. 1=HDE rising edge is line boundary 0=HDE falling edge is line boundary Bit 6: VEN_SYNC_DIS 1= V_EN sync with HSP. need set

				to 1 to avoid glitch. when signal is not cross line boundary Bit 5: RGFR60HZ_EN(no use) Bit 4: RGVB_VALUE (the tcon value during VCNT > VE and VCNT < VS) Bit 3: RGVWIDTH_EN 1: toggle once per frame, high from VS to VE Bit 2: RGODD_DIFF_EN The width of odd line and even line is different. The width of even line is related with HS2/HE2. Bit 1: RGHS2HE2_EN 1= need set 1 when two pulse per line Bit 0: RGWIDTH_EN 1= HS + WIDTH scheme enable, RGHE is used as RGHWIDTH
RGTSIG9_MASK_FNUM<7:0>	0x14bb	0x00	R/W	Bit3:0= MASK_FNUM_fail Fail mode Bit7:4= MASK_FNUM_NORM
RGTSIG10_HS1[7:0]	0x14bc	0x00	R/W	HS1[7:0]
RGTSIG10_HS1[15:8]	0x14bd	0x00	R/W	HS1[15:8]
RGTSIG10_HE1[7:0]	0x14be	0x00	R/W	HE1[7:0]
RGTSIG10_HE1[15:8]	0x14bf	0x00	R/W	HE1[15:8]
RGTSIG10_HS2[7:0]	0x14c0	0x00	R/W	HS2[7:0]
RGTSIG10_HS2[15:8]	0x14c1	0x00	R/W	HS2[15:8]
RGTSIG10_HE2[7:0]	0x14c2	0x00	R/W	HE2[7:0]
RGTSIG10_HE2[15:8]	0x14c3	0x00	R/W	HE2[15:8]
RGTSIG10_VS[7:0]	0x14c4	0x00	R/W	VS[7:0]
RGTSIG10_VS[15:8]	0x14c5	0x00	R/W	VS[15:8]
RGTSIG10_VS[23:16]	0x14c6	0x00	R/W	VS[23:16]
RGTSIG10_VE[7:0]	0x14c7	0x00	R/W	VE[7:0]
RGTSIG10_VE[15:8]	0x14c8	0x00	R/W	VE[15:8]
RGTSIG10_VE[23:16]	0x14c9	0x00	R/W	VE[23:16]

RGTSIG10_VLVH	0x14ca	0x00	R/W	Bit3:0 VL Bit7:4 VH
RGTSIG10_CTRL[7:0]	0x14cb	0x00	R/W	Bit7 RGMASK_EN: 1= masking output during BIST and fail-safe mode Bit6 SINV(signal inversion) Bit5 FIELD_EN Bit4 Reserved Bit3 RGVDE_DUMMY_MASK_EN Bit2 RGJT_EN 1=jitter injection enable (HDE from Jitter) Bit1 RGIFP_DUMMY_MASK_EN 1= no toggle during IS_DUMMY Bit0 RGIFP_MASK_EN 1= no toggle during IFP
RGTSIG10_CTRL[15:8]	0x14cc	0x00	R/W	Bit 7: RGLB4ODD When pulse width is different between odd and even line, the bit is a option for odd/even line boundary definition. 1=HDE rising edge is line boundary 0=HDE falling edge is line boundary Bit 6: VEN_SYNC_DIS 1= V_EN sync with HSP. need set to 1 to avoid glitch. when signal is not cross line boundary Bit 5: RGFR60HZ_EN(no use) Bit 4: RGVB_VALUE (the tcon value during VCNT > VE and VCNT < VS) Bit 3: RGVWIDTH_EN 1: toggle once per frame, high from VS to VE Bit 2: RGODD_DIFF_EN

				The width of odd line and even line is different. The width of even line is related with HS2/HE2. Bit 1: RGHS2HE2_EN 1= need set 1 when two pulse per line Bit 0: RGWIDTH_EN 1= HS + WIDTH scheme enable. RGHE is used as RGHWIDTH
RGTSIG10_MASK_FNUM<7:0>	0x14cd	0x00	R/W	Bit3:0= MASK_FNUM_fail Fail mode Bit7:4= MASK_FNUM_NORM
RGTSIG11_HS1[7:0]	0x14ce	0x00	R/W	HS1[7:0]
RGTSIG11_HS1[15:8]	0x14cf	0x00	R/W	HS1[15:8]
RGTSIG11_HE1[7:0]	0x14d0	0x00	R/W	HE1[7:0]
RGTSIG11_HE1[15:8]	0x14d1	0x00	R/W	HE1[15:8]
RGTSIG11_HS2[7:0]	0x14d2	0x00	R/W	HS2[7:0]
RGTSIG11_HS2[15:8]	0x14d3	0x00	R/W	HS2[15:8]
RGTSIG11_HE2[7:0]	0x14d4	0x00	R/W	HE2[7:0]
RGTSIG11_HE2[15:8]	0x14d5	0x00	R/W	HE2[15:8]
RGTSIG11_VS[7:0]	0x14d6	0x00	R/W	VS[7:0]
RGTSIG11_VS[15:8]	0x14d7	0x00	R/W	VS[15:8]
RGTSIG11_VS[23:16]	0x14d8	0x00	R/W	VS[23:16]
RGTSIG11_VE[7:0]	0x14d9	0x00	R/W	VE[7:0]
RGTSIG11_VE[15:8]	0x14da	0x00	R/W	VE[15:8]
RGTSIG11_VE[23:16]	0x14db	0x00	R/W	VE[23:16]
RGTSIG11_VLVH	0x14dc	0x00	R/W	Bit3:0 VL Bit7:4 VH
RGTSIG11_CTRL[7:0]	0x14dd	0x00	R/W	Bit7 RGMASK_EN: 1= masking output during BIST and fail-safe mode Bit6 SINV(signal inversion) Bit5 FIELD_EN Bit4 Reserved Bit3 RGVDE_DUMMY_MASK_EN Bit2 RGJT_EN 1=jitter injection enable (HDE from Jitter) Bit1 RGIFP_DUMMY_MASK_EN

				1= no toggle during IS_DUMMY Bit0 RGIFP_MASK_EN 1= no toggle during IFP
RGTSIG11_CTRL[15:8]	0x14de	0x00	R/W	Bit 7: RGLB4ODD When pulse width is different between odd and even line, the bit is a option for odd/even line boundary definition. 1=HDE rising edge is line boundary 0=HDE falling edge is line boundary Bit 6: VEN_SYNC_DIS 1= V_EN sync with HSP. need set to 1 to avoid glitch. when signal is not cross line boundary Bit 5: RGFR60HZ_EN(no use) Bit 4: RGVB_VALUE (the tcon value during VCNT > VE and VCNT < VS) Bit 3: RGVWIDTH_EN 1: toggle once per frame, high from VS to VE Bit 2: RGODD_DIFF_EN The width of odd line and even line is different. The width of even line is related with HS2/HE2. Bit 1: RGHS2HE2_EN 1= need set 1 when two pulse per line Bit 0: RGWIDTH_EN 1= HS + WIDTH scheme enable, RGHE is used as RGHWIDTH
RGTSIG11_MASK_FNUM<7:0>	0x14df	0x00	R/W	Bit3:0= MASK_FNUM_fail Fail mode

				Bit7:4= MASK_FNUM_NORM
RGTSIG12_HS1[7:0]	0x14e0	0x00	R/W	HS1[7:0]
RGTSIG12_HS1[15:8]	0x14e1	0x00	R/W	HS1[15:8]
RGTSIG12_HE1[7:0]	0x14e2	0x00	R/W	HE1[7:0]
RGTSIG12_HE1[15:8]	0x14e3	0x00	R/W	HE1[15:8]
RGTSIG12_HS2[7:0]	0x14e4	0x00	R/W	HS2[7:0]
RGTSIG12_HS2[15:8]	0x14e5	0x00	R/W	HS2[15:8]
RGTSIG12_HE2[7:0]	0x14e6	0x00	R/W	HE2[7:0]
RGTSIG12_HE2[15:8]	0x14e7	0x00	R/W	HE2[15:8]
RGTSIG12_VS[7:0]	0x14e8	0x00	R/W	VS[7:0]
RGTSIG12_VS[15:8]	0x14e9	0x00	R/W	VS[15:8]
RGTSIG12_VS[23:16]	0x14ea	0x00	R/W	VS[23:16]
RGTSIG12_VE[7:0]	0x14eb	0x00	R/W	VE[7:0]
RGTSIG12_VE[15:8]	0x14ec	0x00	R/W	VE[15:8]
RGTSIG12_VE[23:16]	0x14ed	0x00	R/W	VE[23:16]
RGTSIG12_VLVH	0x14ee	0x00	R/W	Bit3:0 VL Bit7:4 VH
RGTSIG12_CTRL[7:0]	0x14ef	0x00	R/W	Bit7 RGMASK_EN: 1= masking output during BIST and fail-safe mode Bit6 SINV(signal inversion) Bit5 FIELD_EN Bit4 Reserved Bit3 RGVDE_DUMMY_MASK_EN Bit2 RGJT_EN 1=jitter injection enable (HDE from Jitter) Bit1 RGIFP_DUMMY_MASK_EN 1= no toggle during IS_DUMMY Bit0 RGIFP_MASK_EN 1= no toggle during IFP
RGTSIG12_CTRL[15:8]	0x14f0	0x00	R/W	Bit 7: RGLB4ODD When pulse width is different between odd and even line, the bit is a option for odd/even line boundary definition. 1=HDE rising edge is line boundary 0=HDE falling edge is line boundary

				Bit 6: VEN_SYNC_DIS 1= V_EN sync with HSP. need set to 1 to avoid glitch. when signal is not cross line boundary Bit 5: RGFR60HZ_EN(no use) Bit 4: RGVB_VALUE (the tcon value during VCNT > VE and VCNT < VS) Bit 3: RGVWIDTH_EN 1: toggle once per frame, high from VS to VE Bit 2: RGODD_DIFF_EN The width of odd line and even line is different. The width of even line is related with HS2/HE2. Bit 1: RGHS2HE2_EN 1= need set 1 when two pulse per line Bit 0: RGWIDTH_EN 1= HS + WIDTH scheme enable, RGHE is used as RGHWIDTH
RGTSIG12_MASK_FNUM<7:0>	0x14f1	0x00	R/W	Bit3:0= MASK_FNUM_fail Fail mode Bit7:4= MASK_FNUM_NORM
RGTSIG13_HS1[7:0]	0x1500	0x00	R/W	HS1[7:0]
RGTSIG13_HS1[15:8]	0x1501	0x00	R/W	HS1[15:8]
RGTSIG13_HE1[7:0]	0x1502	0x00	R/W	HE1[7:0]
RGTSIG13_HE1[15:8]	0x1503	0x00	R/W	HE1[15:8]
RGTSIG13_HS2[7:0]	0x1504	0x00	R/W	HS2[7:0]
RGTSIG13_HS2[15:8]	0x1505	0x00	R/W	HS2[15:8]
RGTSIG13_HE2[7:0]	0x1506	0x00	R/W	HE2[7:0]
RGTSIG13_HE2[15:8]	0x1507	0x00	R/W	HE2[15:8]
RGTSIG13_VS[7:0]	0x1508	0x00	R/W	VS[7:0]
RGTSIG13_VS[15:8]	0x1509	0x00	R/W	VS[15:8]
RGTSIG13_VS[23:16]	0x150a	0x00	R/W	VS[23:16]
RGTSIG13_VE[7:0]	0x150b	0x00	R/W	VE[7:0]

RGTSIG13_VE[15:8]	0x150c	0x00	R/W	VE[15:8]
RGTSIG13_VE[23:16]	0x150d	0x00	R/W	VE[23:16]
RGTSIG13_VLVH	0x150e	0x00	R/W	Bit3:0 VL Bit7:4 VH
RGTSIG13_CTRL[7:0]	0x150f	0x00	R/W	Bit7 RGMASK_EN: 1= masking output during BIST and fail-safe mode Bit6 SINV(signal inversion) Bit5 FIELD_EN Bit4 Reserved Bit3 RGVDE_DUMMY_MASK_EN Bit2 RGJT_EN 1=jitter injection enable (HDE from Jitter) Bit1 RGIFP_DUMMY_MASK_EN 1= no toggle during IS_DUMMY Bit0 RGIFP_MASK_EN 1= no toggle during IFP
RGTSIG13_CTRL[15:8]	0x1510	0x00	R/W	Bit 7: RGLB4ODD When pulse width is different between odd and even line, the bit is a option for odd/even line boundary definition. 1=HDE rising edge is line boundary 0=HDE falling edge is line boundary Bit 6: VEN_SYNC_DIS 1= V_EN sync with HSP. need set to 1 to avoid glitch. when signal is not cross line boundary Bit 5: RGFR60HZ_EN(no use) Bit 4: RGVB_VALUE (the tcon value during VCNT > VE and VCNT < VS) Bit 3: RGVWIDTH_EN 1: toggle once per frame, high from VS to VE

				Bit 2: RGODD_DIFF_EN The width of odd line and even line is different. The width of even line is related with HS2/HE2. Bit 1: RGHS2HE2_EN 1= need set 1 when two pulse per line Bit 0: RGWIDTH_EN 1= HS + WIDTH scheme enable, RGHE is used as RGHWIDTH
RGTSIG13_MASK_FNUM<7:0>	0x1511	0x00	R/W	Bit3:0= MASK_FNUM_fail Fail mode Bit7:4= MASK_FNUM_NORM
RGTSIG14_HS1[7:0]	0x1512	0x00	R/W	HS1[7:0]
RGTSIG14_HS1[15:8]	0x1513	0x00	R/W	HS1[15:8]
RGTSIG14_HE1[7:0]	0x1514	0x00	R/W	HE1[7:0]
RGTSIG14_HE1[15:8]	0x1515	0x00	R/W	HE1[15:8]
RGTSIG14_HS2[7:0]	0x1516	0x00	R/W	HS2[7:0]
RGTSIG14_HS2[15:8]	0x1517	0x00	R/W	HS2[15:8]
RGTSIG14_HE2[7:0]	0x1518	0x00	R/W	HE2[7:0]
RGTSIG14_HE2[15:8]	0x1519	0x00	R/W	HE2[15:8]
RGTSIG14_VS[7:0]	0x151a	0x00	R/W	VS[7:0]
RGTSIG14_VS[15:8]	0x151b	0x00	R/W	VS[15:8]
RGTSIG14_VS[23:16]	0x151c	0x00	R/W	VS[23:16]
RGTSIG14_VE[7:0]	0x151d	0x00	R/W	VE[7:0]
RGTSIG14_VE[15:8]	0x151e	0x00	R/W	VE[15:8]
RGTSIG14_VE[23:16]	0x151f	0x00	R/W	VE[23:16]
RGTSIG14_VLVH	0x1520	0x00	R/W	Bit3:0 VL Bit7:4 VH
RGTSIG14_CTRL[7:0]	0x1521	0x00	R/W	Bit7 RGMASK_EN: 1= masking output during BIST and fail-safe mode Bit6 SINV(signal inversion) Bit5 FIELD_EN Bit4 Reserved Bit3 RGVDE_DUMMY_MASK_EN Bit2 RGJT_EN 1=jitter injection enable (HDE from

				Jitter) Bit1 RGIFP_DUMMY_MASK_EN 1= no toggle during IS_DUMMY Bit0 RGIFP_MASK_EN 1= no toggle during IFP
RGTSIG14_CTRL[15:8]	0x1522	0x00	R/W	Bit 7: RGLB4ODD When pulse width is different between odd and even line, the bit is a option for odd/even line boundary definition. 1=HDE rising edge is line boundary 0=HDE falling edge is line boundary Bit 6: VEN_SYNC_DIS 1= V_EN sync with HSP. need set to 1 to avoid glitch. when signal is not cross line boundary Bit 5: RGFR60HZ_EN(no use) Bit 4: RGVB_VALUE (the tcon value during VCNT > VE and VCNT < VS) Bit 3: RGVWIDTH_EN 1: toggle once per frame, high from VS to VE Bit 2: RGODD_DIFF_EN The width of odd line and even line is different. The width of even line is related with HS2/HE2. Bit 1: RGHS2HE2_EN 1= need set 1 when two pulse per line Bit 0: RGWIDTH_EN 1= HS + WIDTH scheme enable, RGHE is used as RGHWIDTH

RGTSIG14_MASK_FNUM<7:0>	0x1523	0x00	R/W	Bit3:0= MASK_FNUM_fail Fail mode Bit7:4= MASK_FNUM_NORM
RGTSIG15_HS1[7:0]	0x1524	0x00	R/W	HS1[7:0]
RGTSIG15_HS1[15:8]	0x1525	0x00	R/W	HS1[15:8]
RGTSIG15_HE1[7:0]	0x1526	0x00	R/W	HE1[7:0]
RGTSIG15_HE1[15:8]	0x1527	0x00	R/W	HE1[15:8]
RGTSIG15_HS2[7:0]	0x1528	0x00	R/W	HS2[7:0]
RGTSIG15_HS2[15:8]	0x1529	0x00	R/W	HS2[15:8]
RGTSIG15_HE2[7:0]	0x152a	0x00	R/W	HE2[7:0]
RGTSIG15_HE2[15:8]	0x152b	0x00	R/W	HE2[15:8]
RGTSIG15_VS[7:0]	0x152c	0x00	R/W	VS[7:0]
RGTSIG15_VS[15:8]	0x152d	0x00	R/W	VS[15:8]
RGTSIG15_VS[23:16]	0x152e	0x00	R/W	VS[23:16]
RGTSIG15_VE[7:0]	0x152f	0x00	R/W	VE[7:0]
RGTSIG15_VE[15:8]	0x1530	0x00	R/W	VE[15:8]
RGTSIG15_VE[23:16]	0x1531	0x00	R/W	VE[23:16]
RGTSIG15_VLVH	0x1532	0x00	R/W	Bit3:0 VL Bit7:4 VH
RGTSIG15_CTRL[7:0]	0x1533	0x00	R/W	Bit7 RGMASK_EN: 1= masking output during BIST and fail-safe mode Bit6 SINV(signal inversion) Bit5 FIELD_EN Bit4 Reserved Bit3 RGVDE_DUMMY_MASK_EN Bit2 RGJT_EN 1=jitter injection enable (HDE from Jitter) Bit1 RGIFP_DUMMY_MASK_EN 1= no toggle during IS_DUMMY Bit0 RGIFP_MASK_EN 1= no toggle during IFP
RGTSIG15_CTRL[15:8]	0x1534	0x00	R/W	Bit 7: RGLB4ODD When pulse width is different between odd and even line, the bit is a option for odd/even line boundary definition. 1=HDE rising edge is line boundary

				0=HDE falling edge is line boundary Bit 6: VEN_SYNC_DIS 1= V_EN sync with HSP. need set to 1 to avoid glitch. when signal is not cross line boundary Bit 5: RGFR60HZ_EN(no use) Bit 4: RGVB_VALUE (the tcon value during VCNT > VE and VCNT < VS) Bit 3: RGVWIDTH_EN 1: toggle once per frame, high from VS to VE Bit 2: RGODD_DIFF_EN The width of odd line and even line is different. The width of even line is related with HS2/HE2. Bit 1: RGHS2HE2_EN 1= need set 1 when two pulse per line Bit 0: RGWIDTH_EN 1= HS + WIDTH scheme enable, RGHE is used as RGHWIDTH
RGTSIG15_MASK_FNUM<7:0>	0x1535	0x00	R/W	Bit3:0= MASK_FNUM_fail Fail mode Bit7:4= MASK_FNUM_NORM

9.3 PLS Signal Generation

Register name	Offset	Initial Value	Read Write	Description
RGLS_EN_HCNT [7:0]	0x1881	0x00	R/W	LS_EN_HCNT [7:0]
RGLS_EN_HCNT [15:8]	0x1882	0x00	R/W	LS_EN_HCNT [15:8]
RGLS_EN_VCNT[7:0]	0x1883	0x00	R/W	LS_EN_VCNT<7:0>

RGLS_EN_VCN[15:8]	0x1884	0x00	R/W	LS_EN_VCNT<15:8]
RGLS_EN_FCNT	0x1885	0x00	R/W	LS_EN_FCNT
RGLS_EN_CTL	0x1886	0x00	R/W	LS_EN_CTL
RGLS_CLK_START_LINE<7:0>	0x1887	0x00	R/W	LS_CLK_START_LINE<7:0>
RGLS_CLK_START_LINE[15:8]	0x1888	0x00	R/W	LS_CLK_START_LINE[15:8]
RGLS_CLK_START_LINE[23:16]	0x1889	0x00	R/W	LS_CLK_START_LINE[23:16]
RGLS_TOP_CTL[7:0]	0x188a	0x00	R/W	<7> RGLS_EN_MASK_EN <6> RGLS_INIT_MASK_EN <5> RGLS_VBE_MASK_EN <4> RGLS_TT_MASK_EN <3> RGLS_CLK_MASK_EN <2> RGTT_IFP_IGN_BS <1> RGTT_VB_IGN_BS <0> RGINIT_IFP_IGN_BS
RGLS_TOP_CTL[15:8]	0x188b	0x00	R/W	<15:13> Reserved <12> RGOUTPUT_DIS_EN <11> RGLSCLK_SWRST <10> RGLS_VBE_SYNC_DIS <9> RGLS_TT_SYNC_DIS <8> RGLS_CLK_SYNC_DIS
RGLS_TOP_CTL<23:16>	0x188c	0x00	R/W	<23:21> Reserved <20> RGLS_CLK_INIT <19> RGLS_TT_INIT <18> RGLS_INIT_INIT <17> RGLS_VBE_INIT <16> RGLS_EN_INIT
RGLS_CLK_HS[7:0]	0x188d	0x00	R/W	LS_CLK_HS[7:0]
RGLS_CLK_HS[15:8]	0x188e	0x00	R/W	LS_CLK_HS[15:8]
RGLS_CLK_HS[23:16]	0x188f	0x00	R/W	LS_CLK_HS[23:16]
RGLS_CLK_H[7:0]	0x1890	0x00	R/w	LS_CLK_H[7:0]
RGLS_CLK_H[15:8]	0x1891	0x00	R/W	LS_CLK_H[15:8]
RGLS_CLK_H[23:16]	0x1892	0x00	R/W	LS_CLK_H[23:16]
RGLS_CLK_L<7:0>	0x1893	0x00	R/W	LS_CLK_L[7:0]
RGLS_CLK_L[15:8]	0x1894	0x00	R/W	LS_CLK_L[15:8]
RGLS_CLK_L[23:16]	0x1895	0x00	R/W	LS_CLK_L[23:16]

RGLS_CLK_NUM_PULSE[7:0]	0x1896	0x00	R/W	LS_CLK_NUM_PULSE[7:0]
RGLS_CLK_NUM_PULSE15:8]	0x1897	0x00	R/W	LS_CLK_NUM_PULSE15:8]
RGLS_CLK_NUM_PULSE [23:16]	0x1898	0x00	R/W	LS_CLK_NUM_PULSE [23:16]
RGLS_CLK_NUM_LINE[7:0]	0x1899	0x00	R/W	[7:0]:LS_CLK_NUM_LINE_IFP[7:0]
RGLS_CLK_NUM_LINE[15:8]	0x189a	0x00	R/W	[11:8]:LS_CLK_NUM_LINE_IFP[11:8] [15:12] LS_CLK_NUM_LINE_VB[3:0]
RGLS_CLK_NUM_LINE[23:16]	0x189b	0x00	R/W	[23:16] LS_CLK_NUM_LINE_VB[11:4]
RGVBE_START_LINE[7:0]	0x189f	0x00	R/W	VBE_START_LINE[7:0]
RGVBE_START_LINE[15:8]	0x18a0	0x00	R/W	VBE_START_LINE[15:8]
RGVBE_START_LINE[23:16]	0x18a1	0x00	R/W	VBE_START_LINE[23:16]
RGVBE_START_TPC [7:0]	0x18a2	0x00	R/W	VBE_START_TPC [7:0]
RGVBE_START_TPC [15:8]	0x18a3	0x00	R/W	VBE_START_TPC [15:8]
RGVBE_START_TPC [23:16]	0x18a4	0x00	R/W	VBE_START_TPC<23:16>
RGVBE_WIDTH_TP [7:0]	0x18a5	0x00	R/W	VBE_WIDTH_TP [7:0]
RGVBE_WIDTH_TP15:8]	0x18a6	0x00	R/W	VBE_WIDTH_TP15:8]
RGVBE_WIDTH_TP [23:16]	0x18a7	0x00	R/W	VBE_WIDTH_TP [23:16]
RGINIT_VS [7:0]	0x18a8	0x00	R/W	INIT_VS [7:0]
RGINIT_VS15:8]	0x18a9	0x00	R/W	INIT_VS15:8]
RGINIT_VS [23:16]	0x18aa	0x00	R/W	INIT_VS [23:16]
RGINIT_HS [7:0]	0x18ab	0x00	R/W	INIT_HS [7:0]
RGINIT_HS15:8]	0x18ac	0x00	R/W	INIT_HS15:8]
RGINIT_HS [23:16]	0x18ad	0x00	R/W	INIT_HS [23:16]
RGINIT_WIDTH [7:0]	0x18ae	0x00	R/W	INIT_WIDTH [7:0]
RGINIT_WIDTH[15:8]	0x18af	0x00	R/W	INIT_WIDTH[15:8]
RGINIT_WIDTH [23:16]	0x18b0	0x00	R/W	INIT_WIDTH [23:16]
RGTT_START_LINE [7:0]	0x18b1	0x00	R/W	TT_START_LINE[7:0]

RGTT_START_LINE[15:8]	0x18b2	0x00	R/W	TT_START_LINE[15:8]
RGTT_START_LINE [23:16]	0x18b3	0x00	R/W	TT_START_LINE [23:16]
RGTT_START_TPC [7:0]	0x18b4	0x00	R/W	TT_START_TPC [7:0]
RGTT_START_TPC [15:8]	0x18b5	0x00	R/W	TT_START_TPC [15:8]
RGTT_START_TPC [23:16]	0x18b6	0x00	R/W	TT_START_TPC [23:16]
RGTT_END_LINE[7:0]	0x18b7	0x00	R/W	TT_END_LINE[7:0]
RGTT_END_LINE[15:8]	0x18b8	0x00	R/W	TT_END_LINE[15:8]
RGTT_END_LINE[23:16]	0x18b9	0x00	R/W	TT_END_LINE[23:16]
RGTT_BSON_TPC[7:0]	0x18ba	0x00	R/W	TT_BSON_TPC[7:0]
RGTT_BSON_TPC[15:8]	0x18bb	0x00	R/W	TT_BSON_TPC[15:8]
RGTT_BSON_TPC[23:16]	0x18bc	0x00	R/W	TT_BSON_TPC[23:16]
RGTT_BSOFF_TPC[7:0]	0x18bd	0x00	R/W	TT_BSOFF_TPC[7:0]
RGTT_BSOFF_TPC[15:8]	0x18be	0x00	R/W	TT_BSOFF_TPC[15:8]
RGTT_BSOFF_TPC[23:16]	0x18bf	0x00	R/W	TT_BSOFF_TPC[23:16]
RGLS_FLT_EN	0x18c0	0x00	R/W	LS_FLT_EN
RGLS_FLT_IRQ_EN	0x18c1	0x00	R/W	LS_FLT_IRQ_EN
RGLS_FLT_CLK_CTL	0x18c2	0x00	R/W	LS_FLT_CLK_CTL
RGVBE_INIT_MIN_TH	0x18c3	0x00	R/W	VBE_INIT_MIN_TH
RGVBE_INIT_MAX_TH	0x18c4	0x00	R/W	VBE_INIT_MAX_TH
RGVBE_VBE_MIN_TH	0x18c5	0x00	R/W	VBE_VBE_MIN_TH
RGVBE_VBE_MAX_TH	0x18c6	0x00	R/W	VBE_VBE_MAX_TH
RGINIT_INIT_MIN_EN	0x18c7	0x00	R/W	INIT_INIT_MIN_EN
RGINIT_INIT_MAX_EN	0x18c8	0x00	R/W	INIT_INIT_MAX_EN
RGLS_FLT_STATUS_W1C	0x18c9	0x00	R/W	LS_FLT_STATUS_W1C
RGLS_FLT_SUPRESS_EN RGLS_FLT_CTL	0x18ca	0x00	R/W	[3:0] LS_FLT_SUPRESS_EN [5:4] LS_FLT_CTL
RGINIT_NUM_MAX	0x18cb	0x00	R/W	[3:0] INIT_NUM_MAX
RGLSEN_STEP	0x18f3	0x00	R/W	LSEN_STEP_NUM[7:0]
RGLSEN_SW	0x18f4	0x00	R/W	[1:0]:LSEN_STEP_NUM[9:8] [2] LSEN_SW_EN [3] LSEN_SW

10. CDI Control

10.1 Port Pin Mapping and Swapping Control

provide flexible COG port pin swap options for TCON PCB design. The following registers are used for this function.

Register name	Offset	Initial Value	Read Write	Description

RGPORT_SWP[7:0]	0x192b	0x10	R/W	[7:4] PORT2_SWP
				case(RGPORT2_SWP)
				4'h0: PORT2_OUT = PORT1_IN;
				4'h1: PORT2_OUT = PORT2_IN;
				4'h2: PORT2_OUT = PORT3_IN;
				4'h3: PORT2_OUT = PORT4_IN;
				4'h4: PORT2_OUT = PORT5_IN;
				4'h5: PORT2_OUT = PORT6_IN;
				4'h6: PORT2_OUT = PORT7_IN;
				4'h7: PORT2_OUT = PORT8_IN;
				default : PORT2_OUT = PORT2_IN;
				endcase
				[3:0] PORT1_SWP
				case(RGPORT1_SWP)
				4'h0: PORT1_OUT = PORT1_IN;
				4'h1: PORT1_OUT = PORT2_IN;
				4'h2: PORT1_OUT = PORT3_IN;
				4'h3: PORT1_OUT = PORT4_IN;
				4'h4: PORT1_OUT = PORT5_IN;
				4'h5: PORT1_OUT = PORT6_IN;
				4'h6: PORT1_OUT = PORT7_IN;
				4'h7: PORT1_OUT = PORT8_IN;
				default : PORT1_OUT = PORT1_IN;
				endcase

RGPORT_SWP[15:8]	0x192c	0x32	R/W	[7:4] PORT4_SWP case(RGPORT4_SWP) 4'h0: PORT4_OUT = PORT1_IN; 4'h1: PORT4_OUT = PORT2_IN; 4'h2: PORT4_OUT = PORT3_IN;

				4'h3: PORT4_OUT = PORT4_IN;
				4'h4: PORT4_OUT = PORT5_IN;
				4'h5: PORT4_OUT = PORT6_IN;
				4'h6: PORT4_OUT = PORT7_IN;
				4'h7: PORT4_OUT = PORT8_IN;
				default : PORT4_OUT = PORT4_IN;
				endcase
				[3:0] PORT3_SWP
				case(RGPORT3_SWP)
				4'h0: PORT3_OUT = PORT1_IN;
				4'h1: PORT3_OUT = PORT2_IN;
				4'h2: PORT3_OUT = PORT3_IN;
				4'h3: PORT3_OUT = PORT4_IN;
				4'h4: PORT3_OUT = PORT5_IN;
				4'h5: PORT3_OUT = PORT6_IN;
				4'h6: PORT3_OUT = PORT7_IN;
				4'h7: PORT3_OUT = PORT8_IN;
				default : PORT3_OUT = PORT3_IN;
				endcase

RGPORT_SWP[23:16]	0x192d	0x54	R/W	[7:4] PORT6_SWP case(RGPORT6_SWP) 4'h0: PORT6_OUT = PORT1_IN; 4'h1: PORT6_OUT = PORT2_IN; 4'h2: PORT6_OUT = PORT3_IN; 4'h3: PORT6_OUT = PORT4_IN; 4'h4: PORT6_OUT = PORT5_IN; 4'h5: PORT6_OUT = PORT6_IN; 4'h6: PORT6_OUT = PORT7_IN; 4'h7: PORT6_OUT = PORT8_IN; default : PORT6_OUT = PORT6_IN; endcase [3:0] PORT5_SWP case(RGPORT5_SWP) 4'h0: PORT5_OUT = PORT1_IN; 4'h1: PORT5_OUT = PORT2_IN; 4'h2: PORT5_OUT = PORT3_IN; 4'h3: PORT5_OUT = PORT4_IN; 4'h4: PORT5_OUT = PORT5_IN; 4'h5: PORT5_OUT = PORT6_IN;

				4'h6: PORT5_OUT = PORT7_IN; 4'h7: PORT5_OUT = PORT8_IN; default : PORT5_OUT = PORT5_IN; endcase

RGPORT_SWP[31:24]	0x192e	0x76	R/W	[7:4] PORT8_SWP
				case(RGPORT8_SWP)
				4'h0: PORT8_OUT = PORT1_IN;
				4'h1: PORT8_OUT = PORT2_IN;
				4'h2: PORT8_OUT = PORT3_IN;
				4'h3: PORT8_OUT = PORT4_IN;
				4'h4: PORT8_OUT = PORT5_IN;
				4'h5: PORT8_OUT = PORT6_IN;
				4'h6: PORT8_OUT = PORT7_IN;
				4'h7: PORT8_OUT = PORT8_IN;
				default : PORT8_OUT = PORT8_IN;
				endcase
				[3:0] PORT7_SWP
				case(RGPORT7_SWP)
				4'h0: PORT7_OUT = PORT1_IN;
				4'h1: PORT7_OUT = PORT2_IN;
				4'h2: PORT7_OUT = PORT3_IN;
				4'h3: PORT7_OUT = PORT4_IN;
				4'h4: PORT7_OUT = PORT5_IN;
				4'h5: PORT7_OUT = PORT6_IN;
				4'h6: PORT7_OUT = PORT7_IN;
				4'h7: PORT7_OUT = PORT8_IN;
				default : PORT7_OUT = PORT7_IN;
				endcase

RGPN_SWP[7:0]	0x192f	0x00	R/W	[7] pn swap for port 4 lane 1 [6] pn swap for port 4 lane 0 [5] pn swap for port 3 lane 1 [4] pn swap for port 3 lane 0 [3] pn swap for port 2 lane 1 [2] pn swap for port 2 lane 0 [1] pn swap for port 1 lane 1 [0] pn swap for port 1 lane 0
RGPN_SWP[15:8]	0x1930	0x00	R/W	[7] pn swap for port 8 lane 1 [6] pn swap for port 8 lane 0

				[5] pn swap for port 7 lane 1 [4] pn swap for port 7 lane 0 [3] pn swap for port 6 lane 1 [2] pn swap for port 6 lane 0 [1] pn swap for port 5 lane 1 [0] pn swap for port 5 lane 0
	0x1931	0x00	R/W	Reserved
RGLANE_SWP[7:0]	0x1932	0x00	R/W	[7] lane7 swap for port 8 [6] lane6 swap for port 7 [5] lane5 swap for port 6 [4] lane4 swap for port 5 [3] lane3 swap for port 4 [2] lane2 swap for port 3 [1] lane1 swap for port 2 [0] lane0 swap for port 1
RGPRE_SWP[7:0]	0x1933	0x00	R/W	[7] RGVD_EN 1: pixel data from RGVD [6:4] RGRGB_ORDER case(RGRGB_ORDER) 3'b000 : VD={R,G,B}; 3'b001 : VD={R,B,G}; 3'b010 : VD={G,R,B}; 3'b011 : VD={G,B,R}; 3'b100 : VD={B,R,G}; 3'b101 : VD={B,G,R}; default : VD={R,G,B}; endcase [3] RGDATA_INV 1: invert data [2] RG01_SWP 1: swap between video data 0 and 1. [1] RG0123_SWP 1: swap between video data 0/1 from PM and 2/3. [0] RGBIT_ORDER_SWP 1: bit order swap (LSB <-> MSB) for pixel data.
RGPRE_SWP[9:8]	0x1934	0x00	R/W	[7:2] Reserved [1] RGPOL_SWP_EN 1: enable [0] RGSWP_POL

10.2 Driver Control

The following table listed the registers for CDI driver control.

Register name	Offset	Initial Value	Read Write	Description
REG_RPIO[7:0]	0x0f00	0x40	R/W	[7] Reserved [6:5] Current driver source current control 00: 1.3mA 01: 2.6mA 10: 3.9mA 11: 5.2mA [4] Current driver mode select: 0: normal mode; 1: Output impedance test enable; [3] Reserved [2:1]COG current driver output state select: 00: L state 01:H state 10:HiZ state 11:HiZ state [0] COG current driver DC test enable. 0 : disable 1 : enable
REG_RPIO[15:8]	0x0f01	0x45	R/W	[7] Turn on additional figure of output impedance when 1 [10:8] Current driver differential output impedance control 000 : 180 Ohm 001 : 150 Ohm 010 : 128 Ohm 011 : 111 Ohm 100 : 100Ohm 101 : 90 Ohm 110 : 82 Ohm 111 : 75 Ohm [6] Current driver output slope control enable;

				0 : disable 1 : enable [5] Current driver output slope control delay tune; 0: 4 taps delay slope control 1: 3 taps delay slope control [4:3]Reserved [2:0] Current driver differential output impedance control (when REG_PIO[567] = 0) 000 : 222 Ohm 001 : 180 Ohm 010 : 150 Ohm 011 : 128 Ohm 100 : 111Ohm 101 : 100 Ohm 110 : 90 Ohm 111 : 82 Ohm
REG_RPIO[23:16]	0x0f02	0x00	R/W	[7:4] driver bias current control (port1) [3:0] driver bias current control (port0) 0000 : 50u 1000 : 75u 0001 : 47u 1001 : 72u 0010 : 44u 1010 : 69u 0011 : 41u 1011 : 66u 0100 : 38u 1100 : 63u 0101 : 34u 1101 : 60u 0110 : 31u 1110 : 56u 0111 : 28u 1111 : 53u
REG_RPIO[31:24]	0x0f03	0x00	R/W	[7:4] driver bias current control (port3) [3:0] driver bias current control (port2)
REG_RPIO[39:32]	0x0f04	0x00	R/W	[7:4] driver bias current control (port5) [3:0] driver bias current control (port4)
REG_RPIO[47:40]	0x0f05	0x00	R/W	[7:4] driver bias current control (port7) [3:0] driver bias current control (port6)
REG_RPIO[79:72]	0x0f09	0x00	R/W	REG_RPIO[80]+[7:6] COG1 data0 delay selection 000 : 29ps 100 : 152ps

				001 : 60ps 101 : 183ps 010 : 90ps 110 : 213ps 011 : 121ps 111 : 246ps [5:3]COG0 data1 delay selection 000 : 29ps 100 : 152ps 001 : 60ps 101 : 183ps 010 : 90ps 110 : 213ps 011 : 121ps 111 : 246ps [2:0] COG0 data0 delay selection 000 : 29ps 100 : 152ps 001 : 60ps 101 : 183ps 010 : 90ps 110 : 213ps 011 : 121ps 111 : 246ps
REG_RPIO[87:80]	0x0f0a	0x00	R/W	[7] See REG_RPIO[89:88] [6:4]COG2 data0 delay selection 000 : 29ps 100 : 152ps 001 : 60ps 101 : 183ps 010 : 90ps 110 : 213ps 011 : 121ps 111 : 246ps [3:1] COG1 data1 delay selection 000 : 29ps 100 : 152ps 001 : 60ps 101 : 183ps 010 : 90ps 110 : 213ps 011 : 121ps 111 : 246ps [0] See REG_RPIO[79:78]
REG_RPIO[95:88]	0x0f0b	0x00	R/W	[7:5] COG3 data1 delay selection 000 : 29ps 100 : 152ps 001 : 60ps 101 : 183ps 010 : 90ps 110 : 213ps 011 : 121ps 111 : 246ps [4:2] COG3 data0 delay selection 000 : 29ps 100 : 152ps 001 : 60ps 101 : 183ps 010 : 90ps 110 : 213ps 011 : 121ps 111 : 246ps

				[1:0]+REG_RPIO[87] COG2 data1 delay selection 000 : 29ps 100 : 152ps 001 : 60ps 101 : 183ps 010 : 90ps 110 : 213ps 011 : 121ps 111 : 246ps
REG_RPIO[103:96]	0x0f0c	0x00	R/W	REG_RPIO[104]+[7:6] COG5 data0 delay selection 000 : 29ps 100 : 152ps 001 : 60ps 101 : 183ps 010 : 90ps 110 : 213ps 011 : 121ps 111 : 246ps [5:3]COG4 data1 delay selection 000 : 29ps 100 : 152ps 001 : 60ps 101 : 183ps 010 : 90ps 110 : 213ps 011 : 121ps 111 : 246ps [2:0] COG4 data0 delay selection 000 : 29ps 100 : 152ps 001 : 60ps 101 : 183ps 010 : 90ps 110 : 213ps 011 : 121ps 111 : 246ps
REG_RPIO[111:104]	0x0f0d	0x00	R/W	[7] See REG_RPIO[113:112] [6:4]COG6 data0 delay selection 000 : 29ps 100 : 152ps 001 : 60ps 101 : 183ps 010 : 90ps 110 : 213ps 011 : 121ps 111 : 246ps [3:1] COG5 data1 delay selection 000 : 29ps 100 : 152ps 001 : 60ps 101 : 183ps 010 : 90ps 110 : 213ps 011 : 121ps 111 : 246ps [0] See REG_RPIO[103:102]

REG_RPIO[119:112]	0x0f0e	0x00	R/W	[7:5] COG7 data1 delay selection 000 : 29ps 100 : 152ps 001 : 60ps 101 : 183ps 010 : 90ps 110 : 213ps 011 : 121ps 111 : 246ps [4:2] COG7 data0 delay selection 000 : 29ps 100 : 152ps 001 : 60ps 101 : 183ps 010 : 90ps 110 : 213ps 011 : 121ps 111 : 246ps [1:0]+REG_RPIO[111] COG6 data1 delay selection 000 : 29ps 100 : 152ps 001 : 60ps 101 : 183ps 010 : 90ps 110 : 213ps 011 : 121ps 111 : 246ps
REG_RPIO[127:120]	0x0f0f	0x90	R/W	[7:4]Current driver output common mode voltage control If REG_COGTX<127>=0 current from res: 0000 : 0.7v 1000 : 0.5v 0001 : 0.725v 1001 : 0.525v 0010 : 0.65v 1010 :0.45v 0011 : 0.675v 1011 : 0.475v 0100 : 0.6v 1100 : 0.4v 0101 : 0.625v 1101 : 0.425v 0110 : 0.55v 1110 : 0.35v 0111 : 0.575v 1111 : 0.375v If REG_COGTX<127>=1 current mirror: [7:4] 0000 : 0.7v 1000 : 0.5v 0001 : 0.725v 1001 : 0.525v 0010 : 0.65v 1010 :0.45v 0011 : 0.675v 1011 : 0.475v 0100 : 0.6v 1100 : 0.4v 0101 : 0.625v 1101 : 0.425v 0110 : 0.55v 1110 : 0.35v 0111 : 0.575v 1111 : 0.375v

				[3] Current driver VCM reference 0: from variable series resistor 1: from variable current [2] Current driver 50U bias current select 0 :total from EXT 1 : 1/2 from INT and 1/2 from EXT [1:0] reserved
REG_RPIO[151:144])	0x0f12	0x40	R/W	[7] Register setting can enable SDDRS(CHG_FLG) to gate SSC during frequency change. 0: enable CHG_FLG(SDDRS) gating 1: CHG_FLG or gating SSC [6] The register to enable gating RVDS/Video SS when DP source SS is enabled. 0 – Using DP source SS to gate RVDS and Video SS. 1 – DP source SS don't gate the RVDS/VISEO SS. [5:4] Spread spectrum frequency select signal. 00 – 30KHz 01 – 10KHz 10 – 20KHz 11 – 40KHz [3] Spread spectrum pattern generator parameters set by register enable signal. 0 – automatic calculation of the parameters 1 – register set parameters [2:0] Spread spectrum down depth select signal 000 – 0.5% 001 – 0.25%

				010 – 0.75% 011 – 1% 100 – 1.25% 101 – 1.5% 110 – 1.75% 111 – 2%

10.3 EPI Packet Setting for Normal Mode

Register name	Offset	Initial Value	Read Write	Description
RGIF_CTRL	0x191a	0x00	R/W	[7:4]Reserved [4] RG2LANE 1: 2lane 0: 1lane [3] RG8BIT 1: 8bit 0: 10bit [2] Reserved [1:0] RGIF_SEL 00: epi 01: ervds 10: reserved 11: apdi
RGEPICFG_PORT 0[7:0]	0x1a00	0x15	R/W	[0] CTR_2 (CTR_START) [1] CTR_3 (CTR_START) [2] CTR_4 (CTR_START) [3] CTR_5 (CTR_START) [4] CTR_6 (CTR_START) [5] CTR_7 (CTR_START) [6] CTR_8 (PGMA_EN) [7] CTR_9 (PGMA_EN)
RGEPICFG_PORT 0[15:8]	0x1a01	0x00	R/W	[0] CTR_10 (PGMA_EN) [1] CTR_11 (Reserved) [2] CTR_12 (D_SCR_EN) [3] CTR_13 (D_SCR_RST) [4] CTR_14 (IMPC1) [5] CTR_15 (IMPC1) [6] CTR_16 (IMPC1) [7] CTR_17 (IMPC2)

RGEPICFG_PORT 0[23:16]	0x1a02	0x00	R/W	[0]	CTR_18 (IMPC2)
				[1]	CTR_19 (IMPC2)
				[2]	CTR_20 (ERR_DISP)
				[3]	CTR_21 (ERR_DISP)
				[4]	CTR_22 (ERR_DISP)
				[5]	CTR_23 (CNT_SEL)
				[6]	CTR_24 (CNT_SEL)
				[7]	CTR_25 (CNT_SEL)
RGEPICFG_ODD_ PORT0 [31:24]	0x1a03	0x00	R/W	[0]	CTR1_ODD_2 (SOE_START[0])
				[1]	CTR1_ODD_3 (SOE_START[1])
				[2]	CTR1_ODD_4 (SOE_START[2])
				[3]	CTR1_ODD_5 (SOE_START[3])
				[4]	CTR1_ODD_6 (SOE_START[4])
				[5]	CTR1_ODD_7 (SOE_START[5])
				[6]	CTR1_ODD_8 (PAIR_PKT)
				[7]	CTR1_ODD_9 (PAIR_PKT)
RGEPICFG_ODD_ PORT0 [39:32]	0x1a04	0x00	R/W	[0]	CTR1_ODD_10 (PAIR_PKT)
				[1]	CTR1_ODD_11 (SOE_END [0])
				[2]	CTR1_ODD_12 (SOE_ END [1])
				[3]	CTR1_ODD_13 (SOE_ END [2])
				[4]	CTR1_ODD_14 (SOE_ END [3])
				[5]	CTR1_ODD_15 (SOE_ END [4])
				[6]	CTR1_ODD_16 (SOE_ END [5])
				[7]	CTR1_ODD_17 (RXC)
RGEPICFG_ODD_ PORT0 [47:40]	0x1a05	0x00	R/W	[0]	CTR1_ODD_18 (GPWRC1)
				[1]	CTR1_ODD_19 (GPWRC2)
				[2]	CTR1_ODD_20 (CRC_EN)
				[3]	CTR1_ODD_21 (CRC_EN)
				[4]	CTR1_ODD_22 (CRC_EN)
				[5]	CTR1_ODD_23 (CRC_ERRRST)
				[6]	CTR1_ODD_24 (CRC_ERRRST)
				[7]	CTR1_ODD_25 (CRC_ERRRST)
RGEPICFG_PORT 0[55:48]	0x1a06	0x00	R/W	[0]	Reserved
				[1]	CTR2_3 (MODE)
				[2]	CTR2_4 (H2DOT)
				[3]	CTR2_5 (LTD1)
				[4]	CTR2_6 (LTD2)
				[5]	CTR2_7 (OBOF)
				[6]	CTR2_8 (GBOC1)
				[7]	CTR2_9 (GBOC2)
RGEPICFG_PORT 0[63:56]	0x1a07	0x00	R/W	[0]	CTR2_10 (Reserved)
				[1]	CTR2_11 (no use, GSP by HW)

				[2] CTR2_12 (CSC) [3] CTR2_13 (GMAEN1) [4] CTR2_14 (GMAEN2) [5] CTR2_15 (GMAEN3) [6] CTR2_16 (POLC) [7] CTR2_17 (PWRC1)
RGEPICFG_PORT0[71:64]	0x1a08	0x00	R/W	[0] CTR2_18 (PWRC2) [1] CTR2_19 (PWRC3) [2] CTR2_20 (D_SCR_EN) [3] CTR2_21 (D_SCR_EN) [4] CTR2_22 (D_SCR_RST) [5] CTR2_23 (D_SCR_RST) [6] CTR2_24 (ACS_EN) [7] CTR2_25 (BURST)
RGEPICFG_PORT0[79:72]	0x1a09	0x2a	R/W	[0] CTR2_2 (DATA_START) [1] CTR2_3 (DATA_START) [2] CTR2_4 (DATA_START) [3] CTR2_5 (DATA_START) [4] CTR2_6 (DATA_START) [5] CTR2_7 (DATA_START) [6] CTR2_8 (BIAS_CTRL) [7] CTR2_9 (BIAS_CTRL)
RGEPICFG_PORT0[87:80]	0x1a0a	0x00	R/W	[0] CTR2_10 (BIAS_CTRL) [1] CTR2_11 (VB_PD) [2] CTR2_12 (VB_PD) [3] CTR2_13 (VB_PD) [4] CTR2_14 (EQ1) [5] CTR2_15 (EQ1) [6] CTR2_16 (EQ1) [7] CTR2_17 (EQ2)
RGEPICFG_PORT0[95:88]	0x1a0b	0x00	R/W	[0] CTR2_18 (EQ2) [1] CTR2_19 (EQ2) [2] CTR2_20 (ADDR0) [3] CTR2_21 (ADDR1) [4] CTR2_22 (ADDR2) [5] CTR2_23 (ADDR3) [6] CTR2_24 (ADDR4) [7] CTR2_25 (ADDR5)
EPICFG_PORT1[95:0]	0x1a10~0x1a1b	0x00	R/W	Same as PORT0
EPICFG_PORT2[95:0]	0x1a20~0x1a2b	0x00	R/W	Same as PORT0

EPICFG_PORT3[9 5:0]	0x1a30~ 0x1a3b	0x00	R/W	Same as PORT0
EPICFG_PORT4[9 5:0]	0x1a40~ 0x1a4b	0x00	R/W	Same as PORT0
EPICFG_PORT5[9 5:0]	0x1a50~ 0x1a5b	0x00	R/W	Same as PORT0
EPICFG_PORT6[9 5:0]	0x1a60~ 0x1a6b	0x00	R/W	Same as PORT0
EPICFG_PORT7[9 5:0]	0x1a70~ 0x1a7b	0x00	R/W	Same as PORT0

10.4 EPI Packet Setting for bert Mode

Register name	Offset	Initial Value	Read Write	Description
RGEPICFG_BERT[7:0]	0x1a90	0x00	R/W	[0] CTR_START_2 (BERT) [1] CTR_START_3 (BERT) [2] CTR_START_4 (BERT) [3] CTR_START_5 (BERT) [4] CTR_START_6 (BERT) [5] CTR_START_7 (BERT) [6] CTR_START_8 (DSRST) [7] CTR_START_9 (DSRST)
RGEPICFG_BERT[15:8]	0x1a91	0x00	R/W	[0] CTR_START_10 (DSRST) [1] CTR_START_11 (DSEN) [2] CTR_START_12 (DSEN) [3] CTR_START_13 (DSEN) [4] CTR_START_14 (IMPC1) [5] CTR_START_15 (IMPC1) [6] CTR_START_16 (IMPC1) [7] CTR_START_17 (IMPC2)
RGEPICFG_BERT[23:16]	0x1a92	0x00	R/W	[0] CTR_START_18 (IMPC2) [1] CTR_START_19 (IMPC2) [2] CTR_START_20 (ERR_DISP) [3] CTR_START_21 (ERR_DISP) [4] CTR_START_22 (ERR_DISP)

				[5] CTR_START_23 (CNT_SEL) [6] CTR_START_24 (CNT_SEL) [7] CTR_START_25 (CNT_SEL)
RGEPICFG_BERT[31:24]	0x1a93	0x00	R/W	[0] CTR1_2 (Reserved) [1] CTR1_3 (Reserved) [2] CTR1_4 (Reserved) [3] CTR1_5 (Reserved) [4] CTR1_6 (Reserved) [5] CTR1_7 (Reserved) [6] CTR1_8 (PAIR_PKT) [7] CTR1_9 (PAIR_PKT)
RGEPICFG_BERT[39:32]	0x1a94	0x00	R/W	[0] CTR1_10 (PAIR_PKT) [1] CTR1_11 (SOE_WIDTH[3]) [2] CTR1_12 (SOE_WIDTH[4]) [3] CTR1_13 (SOE_WIDTH[5]) [4] CTR1_14 (SOE_WIDTH[6]) [5] CTR1_15 (SOE_WIDTH[7]) [6] CTR1_16 (Reserved) [7] CTR1_17 (Reserved)
RGEPICFG_BERT[47:40]	0x1a95	0x00	R/W	[0] CTR1_18 (Reserved) [1] CTR1_19 (Reserved) [2] CTR1_20 (CRC_EN) [3] CTR1_21 (CRC_EN) [4] CTR1_22 (CRC_EN) [5] CTR1_23 (CRC_ERRRST) [6] CTR1_24 (CRC_ERRRST) [7] CTR1_25 (CRC_ERRRST)
RGEPICFG_BERT[55:48]	0x1a96	0x00	R/W	[0] CTR2_2 (Reserved) [1] CTR2_3 (Reserved) [2] CTR2_4 (Reserved) [3] CTR2_5 (Reserved)

				[4] CTR2_6 (Reserved) [5] CTR2_7 (Reserved) [6] CTR2_8 (Reserved) [7] CTR2_9 (Reserved)
RGEPICFG_BERT[63:56]	0x1a97	0x00	R/W	[0] CTR2_10 (Reserved) [1] CTR2_11 (Reserved) [2] CTR2_12 (Reserved) [3] CTR2_13 (Reserved) [4] CTR2_14 (Reserved) [5] CTR2_15 (Reserved) [6] CTR2_16 (Reserved) [7] CTR2_17 (Reserved)
RGEPICFG_BERT[71:64]	0x1a98	0x00	R/W	[0] CTR2_18 (Reserved) [1] CTR2_19 (Reserved) [2] CTR2_20 (Reserved) [3] CTR2_21 (Reserved) [4] CTR2_22 (Reserved) [5] CTR2_23 (Reserved) [6] CTR2_24 (ACS_EN) [7] CTR2_25 (BURST)
RGEPICFG_BERT[79:72]	0x1a99	0x38	R/W	[0] DATA_START_2 (BERT) [1] DATA_START_3 (BERT) [2] DATA_START_4 (BERT) [3] DATA_START_5 (BERT) [4] DATA_START_6 (BERT) [5] DATA_START_7 (BERT) [6] DATA_START_8 (POL) [7] DATA_START_9 (POL)
RGEPICFG_BERT[87:80]	0x1a9a	0x00	R/W	[0] DATA_START_10 (POL) [1] DATA_START_11 (CLR/HLDb) [2] DATA_START_12 (CLR/HLDb) [3] DATA_START_13 (CLR/HLDb) [4] DATA_START_14 (EQ1) [5] DATA_START_15 (EQ1) [6] DATA_START_16 (EQ1) [7] DATA_START_17 (EQ2)
RGEPICFG_BERT[95:88]	0x1a9b	0x00	R/W	[0] DATA_START_18 (EQ2) [1] DATA_START_19 (EQ2) [2] DATA_START_20

				(ADDR0) [3] DATA_START_20 (ADDR1) [4] DATA_START_20 (ADDR2) [5] DATA_START_20 (ADDR3) [6] DATA_START_20 (ADDR4) [7] DATA_START_20 (ADDR5)
RGEPI_DSTART_BERT[7:0]	0x1a9c	0x00	R/W	[0] EPI_BERT DS_RST_FRAME 1: Enable EPI BERT mode DS_RST per frame 0: Disable [1] EPI DS reset for new scramble 0: Enable EPI new scramble in BERT mode 1: Disable [7:2] Reserved
RGCFG_DUP	0x1afe	0x00	R/W	[7:1] Reserved [0] RGCFG_DUP 1: all ports configuration from port1

10.5 ERVDS Packet Setting

Register name	Offset	Initial Value	Read Write	Description
RGIF_CTRL	0x191a	0x00	R/W	[7:4]Reserved [4] RG2LANE 1: 2lane 0: 1lane [3] RG8BIT 1: 8bit 0: 10bit [2] Reserved [1:0] RGIF_SEL 00: epi

				01: ervds 10: reserved 11: apdi
RGERVDSCFG_PORT0[7:0]	0x1a00	0x15	R/W	eRVDS PORT0 CONFIG[7:0] [7:6] Reserved [5:3] Reserved [2:0] Logic0
RGERVDSCFG_PORT0[15:8]	0x1a01	0x00	R/W	eRVDS PORT0 CONFIG[15:8] [7] EQ[0] [6:4] Reserved [3:1] Reserved [0] Reserved
RGERVDSCFG_PORT0[23:16]	0x1a02	0x00	R/W	eRVDS PORT0 CONFIG[23:16] [7:5] EQ[2] [4:2] EQ[1] [1:0] EQ[0]
RGERVDSCFG_PORT0[31:24]	0x1a03	0x00	R/W	eRVDS PORT0 CONFIG[31:24] [7:6] TCC[0] [5:3] BIAS[1] [2:0] BIAS[0]
RGERVDSCFG_PORT0[39:32]	0x1a04	0x00	R/W	eRVDS PORT0 CONFIG[39:32] [7] Reserved [6:4] SHL [3:1] TCC[1] [0] TCC[0]
RGERVDSCFG_PORT0[47:40]	0x1a05	0x00	R/W	eRVDS PORT0 CONFIG[7:0] [7:5] Reserved [4:2] Reserved [1] CHOP[1] [0] CHOP[0]
RGERVDSCFG_PORT0[55:48]	0x1a06	0x00	R/W	eRVDS PORT0 CONFIG[55:48] [7:6] SEL[2] [5:3] SEL[1] [2:0] SEL[0]
RGERVDSCFG_PORT0[63:56]	0x1a07	0x00	R/W	eRVDS PORT0 CONFIG[63:56] [7] Reserved [6:4] LP_MODE[1] [3:1] LP_MODE[0] [0] SEL[2]
RGERVDSCFG_PORT0[71:64]	0x1a08	0x00	R/W	eRVDS PORT0 CONFIG[71:64] [7:5] Reserved [4:2] Reserved [1:0] Reserved
RGERVDSCFG_PORT0[79:72]	0x1a09	0x2a	R/W	eRVDS PORT0 CONFIG[79:72] [7:6] LP_EN [5:3] D_CTRL[1] [2:0] D_CTRL[0]
RGERVDSCFG_PORT0[87:80]	0x1a0a	0x00	R/W	eRVDS PORT0 CONFIG[87:80] [7] TEST_CFR[1] [6:4] TEST_CFR[0] [3:1] Reserved [0] LP_EN
RGERVDSCFG_PORT0[95:88]	0x1a0b	0x00	R/W	eRVDS PORT0 CONFIG[95:88] [7:5] ERRCNT_RST [4:2] Reserved [1:0] TEST_CFR[1]

RGERVDSCFG_PORT0[103:96]	0x1a0c	0x00	R/W	eRVDS PORT0 CONFIG[103:96] [7] LP_THRD[4] [6] LP_THRD[3] [5:3] Reserved [2:0] Reserved
RGERVDSCFG_PORT0[110:104]	0x1a0d	0x00	R/W	eRVDS PORT0 CONFIG[110:104] [7] LP_THRD[6] [6:4] FSR[1] [3:1] FSR[0] [0] LP_THRD[5]
RGERVDSCFG_PORT0[119:112]	0x1a0e	0x00	R/W	eRVDS PORT0 CONFIG[119:112] [7] LP_THRD[2] [6] LP_THRD[1] [5] LP_THRD[0] [4:2] SKEW_RD [1] LP_BIAS[1] [0] LP_BIAS[0]
RGERVDSCFG_PORT0[127:120]	0x1a0f	0x00	R/W	eRVDS PORT0 CONFIG[127:120] [7:6] DST_CONFIG[0] [5:3] BANK0 DPS_CONFIG[1] [2:0] BANK0 DPS_CONFIG[0]
RGERVDSCFG_PORT0[135:128]	0x1a10	0x00	R/W	eRVDS PORT0 CONFIG[135:128] [7] LOAD[0] [6:4] Reserved [3:1] DST_CONFIG[1] [0] DST_CONFIG[0]
RGERVDSCFG_PORT0[143:136]	0x1a11	0x00	R/W	eRVDS PORT0 CONFIG[143:136] [7:5] LOAD[2] [4:2] LOAD[1] [1:0] LOAD[0]
RGERVDSCFG_PORT0[151:144]	0x1a12	0x00	R/W	eRVDS PORT0 CONFIG[151:144] [7:6] BIAS[2] [5] CLK1S[5] [4] CLK1S[4] [3] CLK1S[3] [2] CLK1S[2] [1] CLK1S[1] [0] CLK1S[0]
RGERVDSCFG_PORT0[159:152]	0x1a13	0x00	R/W	eRVDS PORT0 CONFIG[159:152] [7] IGEN [6:4] DATAPOL [3:1] TCC[2] [0] BIAS[2]
RGERVDSCFG_PORT0[167:160]	0x1a14	0x00	R/W	eRVDS PORT0 CONFIG[167:160] [7:5] PORb [4:2] tMBC [1:0] IGEN
RGERVDSCFG_PORT0[177:168]	0x1a15	0x00	R/W	eRVDS PORT0 CONFIG[177:168] [7:6] MBCCHOP[1] [5:3] MBCCHOP[0] [2:0] VERR_CTR
RGERVDSCFG_PORT0[183:176]	0x1a16	0x00	R/W	eRVDS PORT0 CONFIG[183:176] [7] Reserved [6:4] Reserved [3:1] MBCCHOP[2]

				[0] MBCCHOP[1]
RGERVDSCFG_PORT0[191:184]	0x1a17	0x00	R/W	eRVDS PORT0 CONFIG[191:184] [7:5] VCRC [4:2] Reserved [1:0] Reserved
RGERVDSCFG_PORT0[199:192]	0x1a18	0x00	R/W	eRVDS PORT0 CONFIG[199:192] [7:6] ERRCNT_MD [5:3] VERR_NUM[1] [2:0] VERR_NUM[0]
RGERVDSCFG_PORT0[207:200]	0x1a19	0x00	R/W	eRVDS PORT0 CONFIG[207:200] [7] Reserved [6:4] Reserved [3:1] Reserved [0] ERRCNT_MD
RGERVDSCFG_PORT0[215:208]	0x1a1a	0x00	R/W	eRVDS PORT0 CONFIG[215:208] [7:5] Reserved [4:2] Reserved [1:0] Reserved
RGERVDSCFG_PORT0[223:216]	0x1a1b	0x00	R/W	eRVDS PORT0 CONFIG[223:216] [7:6] Reserved [5:3] Reserved [2:0] Reserved
RGERVDSCFG_PORT0[231:224]	0x1a1c	0x00	R/W	eRVDS PORT0 CONFIG[231:224] [7] Reserved [6:4] Reserved [3:1] Reserved [0] Reserved
RGERVDSCFG_PORT0[239:232]	0x1a1d	0x00	R/W	eRVDS PORT0 CONFIG[239:232] [7:5] BANK0 DPS [4:2] VBLANK_CNT [1:0] Reserved
RGERVDSCFG_PORT1[239:0]	0x1a20~0x1a3d	0x00	R/W	Same as PORT0
RGERVDSCFG_PORT2[239:0]	0x1a40~0x1a5d	0x00	R/W	Same as PORT0
RGERVDSCFG_PORT3[239:0]	0x1a60~0x1a7d	0x00	R/W	Same as PORT0
RGERVDSCFG_PORT4[239:0]	0x1a80~0x1a9d	0x1b99 = 0x38 Others = 0x00	R/W	Same as PORT0
RGERVDSCFG_PORT5[239:0]	0x1aa0~0x1abd	0x00	R/W	Same as PORT0
RGERVDSCFG_PORT6[239:0]	0x1ac0~0x1add	0x00	R/W	Same as PORT0
RGERVDSCFG_PORT7[239:0]	0x1ae0~0x1afd	0x00	R/W	Same as PORT0
RGCFG_DUP	0x1afe	0x00	R/W	[0] RGCFG_DUP

				1: all ports packet configuration is same as port0 0: Each port packet configuration is programmed independently [7:1] Reserved
RGERVDSCFG_GAMMA0[7:0]	0x1b00	0x00	R/W	[7:4] BANK0_IGAM1[3:0] ERVDS gamma bank0 IGAM1[3:0] [3:0] Reserved
RGERVDSCFG_GAMMA0[15:8]	0x1b01	0x00	R/W	[7:2] BANK0_IGMA2[5:0] ERVDS gamma bank0 IGMA2[5:0] [1:0] BANK0_IGAM1[5:4] ERVDS gamma bank0 IGAM1[5:4]
RGERVDSCFG_GAMMA0[23:16]	0x1b02	0x00	R/W	[7:6] BANK0_IGAM4[1:0] ERVDS gamma bank0 IGAM4[1:0] [5:0] BANK0_IGAM3[5:0] ERVDS gamma bank0 IGAM3[5:0]
RGERVDSCFG_GAMMA0[31:24]	0x1b03	0x00	R/W	[7:4] BANK0_IGAM5[3:0] ERVDS gamma bank0 IGAM5[3:0] [3:0] BANK0_IGAM4[5:2] ERVDS gamma bank0 IGAM4[5:2]
RGERVDSCFG_GAMMA0[39:32]	0x1b04	0x00	R/W	[7:2] BANK0_IGAM6[5:0] ERVDS gamma bank0 IGAM6[5:0] [1:0] BANK0_IGAM5[5:4] ERVDS gamma bank0 IGAM5[5:4]
RGERVDSCFG_GAMMA0[47:40]	0x1b05	0x00	R/W	[7:6] BANK0_IGAM8[1:0] ERVDS gamma bank0 IGAM8[1:0] [5:0] BANK0_IGAM7[5:0] ERVDS gamma bank0 IGAM7[5:0]

RGERVDSCFG_GAMMA0[55:48]	0x1b06	0x00	R/W	[7:4] BANK0_IGAM9[3:0] ERVDS gamma bank0 IGAM9[3:0] [3:0] BANK0_IGAM8[5:2] ERVDS gamma bank0 IGAM8[5:2]
RGERVDSCFG_GAMMA0[63:56]	0x1b07	0x00	R/W	[7:2] BANK0_IGAM9[5:0] ERVDS gamma bank0 IGAM9[5:0] [1:0] BANK0_IGAM9[5:4] ERVDS gamma bank0 IGAM9[5:4]
RGERVDSCFG_GAMMA0[71:64]	0x1b08	0x00	R/W	[7:0] reserved
RGERVDSCFG_GAMMA0[79:72]	0x1b09	0x00	R/W	[7:0] reserved
RGERVDSCFG_GAMMA1[79:0]	0x1b0a~ 0x1b13	0x00	R/W	BANK1_IGAM ERVDS gamma bank1
RGERVDSCFG_GAMMA2[79:0]	0x1b14~ 0x1b1d	0x00	R/W	BANK2_IGAM ERVDS gamma bank2
RGERVDSCFG_GAMMA3[79:0]	0x1b1e~ 0x1b27	0x00	R/W	BANK3_IGAM ERVDS gamma bank3
RGCFG_DUP	0x1afe	0x00	R/W	[7:1] Reserved [0] RGCFG_DUP 1: all ports configuration from port1

10.6 APDI Packet Setting

Register name	Offset	Initial Value	Read Write	Description
RGIF_CTRL	0x1a00	0x15	0x1a00	[7:4]Reserved [4] RG2LANE 1: 2lane 0: 1lane [3] RG8BIT 1: 8bit 0: 10bit [2] Reserved [1:0] RGIF_SEL 00: epi 01: ervds

				10: reserved 11: apdi
RGAPDI_FCFG1_I[7:0]	0x1a00	0x15	R/W	[7] BIAS<1> [6]BIAS<2> [5]EQ<0> [4]EQ<1> [3] EQ<2> [2:0]FCFG<2:0>
RGAPDI_FCFG1_I[15:8]	0x1a01	0x00	R/W	[7]CDR_CTR<2> [6]CDR_CTR<3> [5]CDR_CTR<4> [4]CDR_CTR<5> [3]TCC<0> [2]TCC<1> [1]TCC<2> [0]BIAS<0>
RGAPDI_FCFG1_I[23:16]	0x1a02	0x00	R/W	[7]DST_CONFIG<0> [6]DST_CONFIG<1> [5]CH_SEL<0> [4]CH_SEL<1> [3]CH_SEL<2> [2]SHL [1]CDR_CTR<0> [0]CDR_CTR<1>
RGAPDI_FCFG1_I[31:24]	0x1a03	0x00	R/W	[7]MB_CHOP<2> [6]MB_BIAS<0> [5]MB_BIAS<1> [4]MB_BIAS<2> [3]GB_CHOP<0> [2]GB_CHOP<1> [1]GB_BIAS<0> [0]GB_BIAS<1>
RGAPDI_FCFG1_I[39:32]	0x1a04	0x00	R/W	[7]OUTPUT_DELAY0<3> [6]OUTPUT_DELAY0<4> [5]OUTPUT_DELAY0<5> [4]SKEW_RD [3]SLEW_CTRL<0> [2]SLEW_CTRL<1> [1]MB_CHOP<0> [0]MB_CHOP<1>
RGAPDI_FCFG1_I[47:40]	0x1a05	0x00	R/W	[7]OUTPUT_DELAY1<1> [6]OUTPUT_DELAY1<2>

				[5]OUTPUT_DELAY1<3> [4]OUTPUT_DELAY1<4> [3]OUTPUT_DELAY1<5> [2]OUTPUT_DELAY0<0> [1]OUTPUT_DELAY0<1> [0]OUTPUT_DELAY0<2>
RGAPDI_FCFG1_I[55:48]	0x1a06	0x00	R/W	[7]Reserved [6]Reserved [5]Reserved [4]CBBC_BIAS<0> [3]CBBC_BIAS<1> [2]CBBC_MODE [1]CBBC_EN [0] OUTPUT_DELAY1<0>
RGAPDI_FCFG1_I[63:56]	0x1a07	0x00	R/W	[7] tMBC [6] Reserved [5] Reserved [4] Reserved [3] Reserved [2] Reserved [1] Reserved [0] Reserved
RGAPDI_FCFG1_I[71:64]	0x1a08	0x00	R/W	[7]VERR_NUM<0> [6]VERR_NUM<1> [5]ERRCNT_RST [4]Reserved [3]VERR_COLOR [2]Reserved [1]CRC_MIS_CNT<0> [0]CRC_MIS_CNT<1>
RGAPDI_FCFG1_I[79:72]	0x1a09	0x2a	R/W	[7]CBAOD_TIME_BASE<1> [6]CBAOD_TIME_BASE<2> [5]CBAOD_TIME_BASE<3> [4]CBAOD_AMP<0> [3]CBAOD_AMP<1> [2]CBAOD_EN [1]PWDN_POL_CNT_EN [0]CNT_MD
RGAPDI_FCFG1_I[87:80]	0x1a0a	0x00	R/W	[7] TOEO<2> [6] TOEO<3> [5] TOEO<4>

				[4] TOEO<5> [3] TOEO<6> [2] TOEO<7> [1] IGEN [0] CBAOD_TIME_BASE<0>
RGAPDI_FCFG1_I[95:88]	0x1a0b	0x00	R/W	[7] TGR<4> [6] TGR<5> [5] TGR<6> [4] TGR<7> [3] TGR<8> [2] TGR<9> [1] TOEO<0> [0] TOEO<1>
RGAPDI_FCFG1_I[103:96]	0x1a0c	0x00	R/W	[7] GCHRG_BASE<2> [6] GCHRG_BASE<3> [5] HSEL<0> [4] HSEL<1> [3] TGR<0> [2] TGR<1> [1] TGR<2> [0] TGR<3>
RGAPDI_FCFG1_I[105:104]	0x1a0d	0x00	R/W	[7] Reserved [6] Reserved [5] Reserved [4] Reserved [3] Reserved [2] Reserved [1] GCHRG_BASE<0> [0] GCHRG_BASE<1>
RGAPDI_FCFG2_I[105:0]	0x1a10~0x1a1d	0x00	R/W	RGAPDI_FCFG2_I<105:0>
RGAPDI_FCFG3_I[105:0]	0x1a20~0x1a2	0x00	R/W	RGAPDI_FCFG3_I[105:0]
RGAPDI_FCFG4_I[105:0]	0x1a30~0x1a3d	0x00	R/W	RGAPDI_FCFG4_I[105:0]
RGAPDI_FCFG5_I[105:0]	0x1a40~0x1a4d	0x00	R/W	RGAPDI_FCFG5_I[105:0]
RGAPDI_FCFG6_I[105:0]	0x1a50~0x1a5d	0x00	R/W	RGAPDI_FCFG6_I[105:0]
RGAPDI_FCFG7_I[105:0]	0x1a60~0x1a6d	0x00	R/W	RGAPDI_FCFG7_I[105:0]

RGAPDI_FCFG8_I[105:0]	0x1a70~0x1a7d	0x00	R/W	RGAPDI_FCFG8_I[105:0]
RGAPDI_FCFG_S[7:0]	0x1a80	0x00	R/W	[7] VCOM_REF<4> [6] VCOM_REF<5> [5] VCOM_REF<6> [4] VCOM_REF<7> [3] VCOM_REF<8> [2] VCOM_REF<9> [1] VCOM_REF<10> [0] VCOM_REF<11>
RGAPDI_FCFG_S[15:8]	0x1a81	0x00	R/W	[7] STA_EST_A0<6> [6] STA_EST_A0<7> [5] STA_EST_A0<8> [4] STA_EST_A0<9> [3] VCOM_REF<0> [2] VCOM_REF<1> [1] VCOM_REF<2> [0] VCOM_REF<3>
RGAPDI_FCFG_S[23:16]	0x1a82	0x00	R/W	[7] STA_EST_A1<8> [6] STA_EST_A1<9> [5] STA_EST_A0<0> [4] STA_EST_A0<1> [3] STA_EST_A0<2> [2] STA_EST_A0<3> [1] STA_EST_A0<4> [0] STA_EST_A0<5>
RGAPDI_FCFG_S[31:24]	0x1a83	0x00	R/W	[7] STA_EST_A1<0> [6] STA_EST_A1<1> [5] STA_EST_A1<2> [4] STA_EST_A1<3> [3] STA_EST_A1<4> [2] STA_EST_A1<5> [1] STA_EST_A1<6> [0] STA_EST_A1<7>
RGAPDI_FCFG_S[39:32]	0x1a84	0x00	R/W	[7] STA_EST_A2<2> [6] STA_EST_A2<3> [5] STA_EST_A2<4> [4] STA_EST_A2<5> [3] STA_EST_A2<6> [2] STA_EST_A2<7> [1] STA_EST_A2<8> [0] STA_EST_A2<9>
RGAPDI_FCFG_S[47:40]	0x1a85	0x00	R/W	[7] STA_EST_A3<4> [6] STA_EST_A3<5> [5] STA_EST_A3<6> [4] STA_EST_A3<7> [3] STA_EST_A3<8> [2] STA_EST_A3<9> [1] STA_EST_A2<0> [0] STA_EST_A2<1>
RGAPDI_FCFG_S[51:48]	0x1a86	0x00	R/W	[7] Reserved [6] Reserved [5] Reserved [4] Reserved [3] STA_EST_A3<0> [2] STA_EST_A3<1> [1] STA_EST_A3<2> [0] STA_EST_A3<3>
RGAPDI_LCFG1_L0[7:0]	0x1ab8	0x00	R/W	[7]GD_AD<8> [6]GD_AD<9> [5]DPS_CONFIG_bank0<0>

				[4]DPS_CONFIG_bank0<1> [3]DPS_bank0 [2]Reserved [1]Reserved [0]Reserved
RGAPDI_LCFG1_L0[15:8]	0x1ab9	0x00	R/W	[7]VCOM_OFFSET<4> [6]VCOM_OFFSET<5> [5]VCOM_GAIN<0> [4]VCOM_GAIN<1> [3] GD_AD<4> [2] GD_AD<5> [1] GD_AD<6> [0] GD_AD<7>
RGAPDI_LCFG1_L0[23:16]	0x1aba	0x00	R/W	[7]Reserved [6]Reserved [5]Reserved [4]Reserved [3]CBAOD_TIME_OFFSET<0> [2]CBAOD_TIME_OFFSET<1> [1]VCOM_OFFSET<2> [0]VCOM_OFFSET<3>
RGAPDI_LCFG1_L1[7:0]	0x1abb	0x00	R/W	[7]GD_AD<2> [6]GD_AD<3> [5]GCHRG_OFFSET<0> [4]GCHRG_OFFSET<1> [3]GCHRG_OFFSET<2> [2]GCHRG_OFFSET<3> [1]GCHRG_OFFSET<4> [0]GCHRG_OFFSET<5>
RGAPDI_LCFG1_L1[15:8]	0x1abc	0x00	R/W	[7]CBAOD_TIME_OFFSET<4> [6]CBAOD_TIME_OFFSET<5> [5]VCOM_OFFSET<0> [4]VCOM_OFFSET<1> [3]VCOM_GAIN<2> [2]VCOM_GAIN<3> [1]GD_AD<0> [0]GD_AD<1>
RGAPDI_LCFG1_L1[23:16]	0x1abd	0x00	R/W	[7]Reserved [6]Reserved [5]Reserved [4]Reserved [3]Reserved [2]Reserved [1]CBAOD_TIME_OFFSET<2> [0]CBAOD_TIME_OFFSET<3> >
RGAPDI_LCFG2_L0[23:0]	0x1abe~0x1ac0	0x00	R/W	RGAPDI_LCFG2_L0<23:0>
RGAPDI_LCFG2_L1[23:0]	0x1ac1~0x1ac3	0x00	R/W	RGAPDI_LCFG2_L1<23:0>
RGAPDI_LCFG3_L0[23:0]	0x1ac4~0x1ac6	0x00	R/W	RGAPDI_LCFG3_L0<23:0>
RGAPDI_LCFG3_L1[23:0]	0x1ac7~0x	0x00	R/W	RGAPDI_LCFG3_L1<23:0>

	1ac9			
RGAPDI_LCFG4_L0[23:0]	0x1aca~0x1acc	0x00	R/W	RGAPDI_LCFG4_L0<23:0>
RGAPDI_LCFG4_L1[23:0]	0x1acd~0x1acf	0x00	R/W	RGAPDI_LCFG4_L1<23:0>
RGAPDI_LCFG5_L0[23:0]	0x1ad0~0x1ad2	0x00	R/W	RGAPDI_LCFG5_L0<23:0>
RGAPDI_LCFG5_L1[23:0]	0x1ad3~0x1ad5	0x00	R/W	RGAPDI_LCFG5_L1<23:0>
RGAPDI_LCFG6_L0[23:0]	0x1ad6~0x1ad8	0x00	R/W	RGAPDI_LCFG6_L0<23:0>
RGAPDI_LCFG6_L1[23:0]	0x1ad9~0x1adb	0x00	R/W	RGAPDI_LCFG6_L1<23:0>
RGAPDI_LCFG7_L0[23:0]	0x1adc~0x1ade	0x00	R/W	RGAPDI_LCFG7_L0<23:0>
RGAPDI_LCFG7_L1[23:0]	0x1adf~0x1ae1	0x00	R/W	RGAPDI_LCFG7_L1<23:0>
RGAPDI_LCFG8_L0[23:0]	0x1ae2~0x1ae4	0x00	R/W	RGAPDI_LCFG8_L0<23:0>
RGAPDI_LCFG8_L1[23:0]	0x1ae5~0x1ae7	0x00	R/W	[7] Reserved [6:5] PORT2 DPS_CONFIG<1:0> bank1 [4] PORT2 DPS bank1 [3] Reserved [2:1] PORT1 DPS_CONFIG<1:0> bank1 [0] PORT1 DPS bank1
RGAPDI_PS12_BANK1	0x1af0	0x00	R/W	[7] Reserved [6:5] PORT4 DPS_CONFIG<1:0> bank1 [4] PORT4 DPS bank1 [3] Reserved [2:1] PORT3 DPS_CONFIG<1:0> bank1 [0] PORT3 DPS bank1
RGAPDI_PS34_BANK1	0x1af1	0x00	R/W	[7] Reserved [6:5] PORT6 DPS_CONFIG<1:0> bank1 [4] PORT6 DPS bank1 [3] Reserved [2:1] PORT5 DPS_CONFIG,<1:0> bank1

				[0] PORT5 DPS bank1
RGAPDI_PS56_BANK1	0x1af2	0x00	R/W	[7] Reserved [6:5] PORT8 DPS_CONFIG<1:0> bank1 [4] PORT8 DPS bank1 [3] Reserved [2:1] PORT7 DPS_CONFIG<1:0> bank1 [0] PORT7 DPS bank1
RGAPDI_PS78_BANK1	0x1af3	0x00	R/W	[7] Reserved [6:5] PORT2 DPS_CONFIG<1:0> bank1 [4] PORT2 DPS bank1 [3] Reserved [2:1] PORT1 DPS_CONFIG<1:0> bank1 [0] PORT1 DPS bank1
RGCFG_DUP	0x1afe	0x00	R/W	[7:1] Reserved [0] RGCFG_DUP 1: all ports configuration from port1

Register name	Offset	Initial Value	Read Write	Description
RGREAD_BIT_T[7:0]	0x1968	0x00	R/W	One bit time for reading MBC. lcd clock sycle as unit
RGREAD_BIT_T[15:8]	0x1969	0x00	R/W	
RGREAD_SAMPLE [7:0]	0x196a	0x00	R/W	For MBC read, sample point for one bit, in unit of LCD_CLK.
RGREAD_SAMPLE [15:8]	0x196b	0x00	R/W	

10.7 MBC Read Back Timing Control and Sequence

RGRDBYTE_NUM	0x196c	0x00	R/W	RGBYTE_NUM
RGRD_CTRL[7:0]	0x196d	0x00	R/W	[7] Reserved [6] RGPWDN_RST_EN [5] RGCDVBLANK_RST_EN [4] RGRD_TO_RST_EN 1: reset read block when time out happen [3] RGRDBYTE_NUM_SW read byte num sw en [2] RGRD_TO_TH_SEL [1] RGMBC_READ_SEL [0] RGRD_SWRST 1: reset read block
REG_DPCD00460h	0x0390	0x00	R/W	[7:2]RBS_COM 0x00=read,0x01=read and clear,0x02=write
REG_DPCD00461h	0x0391	0x00	R/W	RBS_ADDR
REG_DPCD00462h	0x0392	0x00	R/W	RBS_LEN
REG_DPCD00463h	0x0393	0x00	R/W	[7]command valid [6]RBS data valid [5:4]rsvd [3:0]cd_sel
REG_DPCD00464h	0x0394	0x00	R/W	Debug mode
REG_DPCD00465h	0x0395	0x00	R/W	WDATA
REG_DPCD00468h	0x0398	0x00	R/W	Rbs read back data
REG_DPCD00469h	0x0399	0x00	R/W	Rbs read back data
REG_DPCD0046ah	0x039a	0x00	R/W	Rbs read back data
REG_DPCD0046bh	0x039b	0x00	R/W	Rbs read back data
REG_DPCD0046ch	0x039c	0x00	R/W	Rbs read back data
REG_DPCD0046dh	0x039d	0x00	R/W	Rbs read back data
REG_DPCD0046eh	0x039e	0x00	R/W	Rbs read back data
REG_DPCD0046fh	0x039f	0x00	R/W	Rbs read back data
REG_DPCD00440h	0x03f5	0x00	R/W	[4]: BERT hold for EPI BERT [3:0]: 0x0e=BERT enable, others=BERT disable
REG_DPCD00441h	0x03f6	0x00	R/W	MBC

Sample code for MBC read back

```
def read_n_byte(addr,len,cd_num):
    global data_valid
    print "cd%x addr= 0x%x len = %x " %(cd_num,addr,len)
```

```
    WriteReg(0x03, 0x90, 0x00);  ####[7:2] 0x00, read;0x01, read and clear; 0x02, write, [1:0] RBS_ADDR[9:8]
    WriteReg(0x03, 0x91, addr);  ####[7:0] RBS_ADDR[7:0]
    WriteReg(0x03, 0x92, len);   ####[7:0] RBS_LEN
    WriteReg(0x03, 0x93, 0x80+cd_num); ####[7] command valid [6] RBS data valid [3:0] CD_SEL

    sleep (1)

    REG_93 = ReadReg(0x03, 0x93);
    data_valid = ((REG_93 & 0x40) == 0x40);
    if(data_valid):
        REG_98 = ReadReg(0x03, 0x98);
        REG_99 = ReadReg(0x03, 0x99);
        REG_9a = ReadReg(0x03, 0x9a);
        REG_9b = ReadReg(0x03, 0x9b);
        REG_9c = ReadReg(0x03, 0x9c);
        REG_9d = ReadReg(0x03, 0x9d);
        REG_9e = ReadReg(0x03, 0x9e);
        REG_9f = ReadReg(0x03, 0x9f);
        print "1st byte = 0x%02x" %REG_98
        print "2nd byte = 0x%02x" %REG_99
        print "3rd byte = 0x%02x" %REG_9a
        print "4th byte = 0x%02x" %REG_9b
        print "5th byte = 0x%02x" %REG_9c
        print "6th byte = 0x%02x" %REG_9d
        print "7th byte = 0x%02x" %REG_9e
        print "8th byte = 0x%02x" %REG_9f



cd_num = 0
len = 2
RBS_ADDR = 0

while(1):
    read_n_byte(RBS_ADDR,len,cd_num)
    if(data_valid):
        print "read sequence success"
    else:
        print "read sequence fail"
        break
```

10.8 CDI BERT Sequence

1. Enable CDI Vbert

WriteReg(0x03, 0xf5, 0x0e); ## enable VBERT

2. Set CDI read back parameters

WriteReg(0x19, 0x68, 0xe0); ## RGREAD_BIT_T[7:0] [1]# for epi, set 0x70

WriteReg(0x19, 0x69, 0x00); ## RGREAD_BIT_T[15:8]

WriteReg(0x19, 0x6a, 0xc0); ## RGREAD_SAMPLE[7:0] [2]# for epi, set 0x38

WriteReg(0x19, 0x6b, 0x00); ## RGREAD_SAMPLE[15:8]

3. Disable MBC monitoring

WriteReg(0x19, 0x6d, 0x45);

4. Read back data

addr=0x00

Len=0x02

CD num=0-7

WriteReg(0x03, 0x90, 0x00); ####[7:2] 0x00, read;0x01, read and clear; 0x02, write, [1:0] RBS_ADDR[9:8]

WriteReg(0x03, 0x91, addr); ####[7:0] RBS_ADDR[7:0]

WriteReg(0x03, 0x92, len); ####[7:0] RBS_LEN

WriteReg(0x03, 0x93, 0x80+cd_num);####[7] command valid [6] RBS data valid [3:0] CD_SEL

If data_valid=ReadReg(0x03,0x93)&0x40==0x40

DPCD00468h = ReadReg(0x03, 0x98);

DPCD00469h = ReadReg(0x03, 0x99);

DPCD0046ah = ReadReg(0x03, 0x9a);

DPCD0046bh = ReadReg(0x03, 0x9b);

DPCD0046ch = ReadReg(0x03, 0x9c);

DPCD0046dh = ReadReg(0x03, 0x9d);

DPCD0046eh = ReadReg(0x03, 0x9e);

DPCD0046fh = ReadReg(0x03, 0x9f);

10.9 MBC Lock Timeout

Register name	Offset	Initial Value	Read Write	Description
RGLOCK_CTRL[7:0]	0x1910	0x00	R/W	[3] LOCK_TOIRQ_EN 1: enable LOCK interrupt [2] LOCK_TIMER_EN 1:enable LOCK timer
RGLOCK_TH[7:0]	0x1911	0x00	R/W	LOCK_TH<7:0> LOCK interrupt thresh hold
RGLOCK_TH[15:8]	0x1917	0x00	R/W	LOCK_TH<15:8> LOCK interrupt thresh hold
RGLOCK_TH[23:16]	0x1918	0x00	R/W	LOCK_TH<23:16> LOCK interrupt thresh hold
RGENLCD_RGIRQ_CTL	0x196f	0x00	R/W	[5] LOL_IRQ0_EN MBC LOL interrupt to IRQ0 enable [4] TO_IRQ0_EN MBC read time out interrupt to IRQ0 enable [3] RGMBC_LOL_IRQ_EN 1:enable MBC LOL IRQ

11. DisplayPort Control

11.1 Receiver Control

Register name	Offset	Initial Value	Read Write	Description
REG_AUX_OFFE0	0x02e0	0x00	R/W	[1:0] debug select for state memory 00 : STAOUT = {CR_LVL, CR_ADRQ, MSTATE, EQSTATE, CRSTATE}
REG_AUX_OFFE2	0x02e2	0x00	RO	TRAIN_DEBR1 0000 : {1'b0, EQ_BAND, MSTATE}

11.2 Statues Registers

Register name	Offset	Initial Value	Read Write	Description
REG_LINK_OFF108	0x0108	0x00	RO	RX_VTOTAL[7:0]
REG_LINK_OFF109	0x0109	0x00	RO	RX_VTOTAL[15:8]
REG_LINK_OFF10a	0x010a	0x00	RO	RX_VHEIGHT[7:0]
REG_LINK_OFF10b	0x010b	0x00	RO	RX_VHEIGHT[15:8]
REG_LINK_OFF10c	0x010c	0x00	RO	RX_Htotal[7:0]
REG_LINK_OFF10d	0x010d	0x00	RO	RX_Htotal [15:8]
REG_LINK_OFF10e	0x010e	0x00	RO	RX_HWIDTH[7:0]
REG_LINK_OFF10f	0x010f	0x00	RO	RX_HWIDTH[15:8]
REG_LINK_OFF110	0x0110	0x00	RO	RX_N4[7:0]
REG_LINK_OFF111	0x0111	0x00	RO	RX_N4[15:8]
REG_LINK_OFF112	0x0112	0x00	RO	RX_N5b[7:0]
REG_LINK_OFF113	0x0113	0x00	RO	RX_N5b[15:8]
REG_LINK_OFF114	0x0114	0x00	RO	RX_N5A[7:0]
REG_LINK_OFF115	0x0115	0x00	RO	RX_N5A[15:8]
REG_LINK_OFF116	0x0116	0x00	RO	RX_N7[7:0]
REG_LINK_OFF117	0x0117	0x00	RO	RX_N7[15:8]

11.3 Internal Statues Registers

Register name	Offset	Initial Value	Read Write	Description
REG_rx_ml_int1_status	0x0121	0x00	R/W1C	[0] RX_ML_SLEEP [1] RX_ML_STANDBY [2] RX_AUX_PHY_WAKE [3 RX_WAKE_FCHANGE [4] RX_VHEIGHT_ERROR [5] RX_HWIDTH_ERROR [6] RX_N7_ERROR
REG_rx_aux_int_status	0x0158	0x00	R/W1C	[0] DPCD_TRAIN_IRQ_STATUS [1] DPCD_SRC_IRQ_STATUS [2] AUXERR_IRQ_STATUS [3] AUXCMD_IRQ_STATUS [4] CDR_LOL_IRQ_STATUS [5] TRAIN_FINISH_IRQ_STATUS [6] CP_IRQ_STATUS [7] INV_DPCDADDR_IRQ_STATUS

11.4 Internal Jitter Injection and EQ Check Sequence

Register name	Offset	Initial Value	Read Write	Description
Jitter_mag	0x0f88	0	R/W	

Sample code :

```
void dpcd_102_written_interrupt_handler()
  {
    if ((pre_102_value = 2 || pre_102_value = 3 || pre_102_value = 4) &&
          cur_102_value == 0)) {
            if (P15.0x88[6:4] != Jitter_Max)
                P15.0x88[6:4]++;
      }
      pre_102_value = cur_102_value;
  }

   void init_chip()
   {
```

```
        ......
        if (dp_jitter_en)
            P15.0x88[6:4] = Jitter_Min;
    }
```

11.5 I2C over aux

eDP source uses I2C-over-aux for EDID/DisplayID accesses and I2C Master0/Master3

1. *EDID at 0xa0/0xa1*
2. *DisplayID at 0xa4/a5*
3. *I2c master0 and I2C master3*

The TCON has either EDID or DisplayID in the local memory, but it does not distinguish EDID and DisplayID. Whenever the TCON receives the read request at 0xa0/0xa1or 0xa4/0xa5, the TCON reads the contents in the local memory and send back to the source.

The EDID address 0xa0/0xa1conflicts with the I2C base address of external NVM. In order to access both EDID and external NVM from I2C master0, the TCON supports a remap register, I2C_NVM_REMAP_CFG,for the external NVM base address, if the address received by the TCON matches the I2C_NVM_REMAP_CFG.NVM_REMAP_ADDR,the TCON remaps the address and sends either 0xa0/0xa1 or 0xa8/0xa9 on I2C master0.

Register name	Offset	Initial Value	Read Write	Description
I2C_NVM_REMAP_CFG	0x0f88	0x33	R/W	Bit 0: remap_en 1=remap enabled 0=remap disabled Bit 7-0:remap_addr 7bit I2C slave address to be remapped. I2C-over-aux transactions to the slave address in this filed are remapped to the I2C slave address(either 0x50 or 0x54) on the tcon I2C master
I2C_OVER_AUX_CFG	0x1208	0x02	R/W	Bit 0: 0=I2C-over-aux enabled

				1=I2C-over-aux disabled

12. GPIO

Register name	Offset	Initial Value	Read Write	Description
GPIOOEB0	0x1238	0xff	R/W	GPIO output enable for GPIO pin7~ GPIO pin0
GPIOOEB1	0x1239	0xff	R/W	GPIO output enable for GPIO pin15 ~ GPIO pin8
GPIOOEB2	0x123a	0xff	R/W	GPIO output enable for GPIO pin23 ~ GPIO pin16
GPIOOEB3	0x123b	0xff	R/W	GPIO output enable for GPIO pin31 ~ GPIO pin24
GPIOOEB4	0x123c	0xff	R/W	GPIO output enable for GPIO pin39 ~ GPIO pin32
GPIOOEB5	0x123d	0xff	R/W	GPIO output enable for GPIO pin47~ GPIO pin40
GPIOOEB6	0x123e	0xff	R/W	GPIO output enable for GPIO pin55 ~ GPIO pin48
GPIO_OUT0	0x1240	0x00	R/W	GPIO output of GPIO pin7 ~ GPIO pin0
GPIO_OUT1	0x1241	0x00	R/W	GPIO output of GPIO pin15~ GPIO pin8
GPIO_OUT2	0x1242	0x00	R/W	GPIO output of GPIO pin23~ GPIO pin16
GPIO_OUT3	0x1243	0x00	R/W	GPIO output of GPIO pin31 ~ GPIO pin24
GPIO_OUT4	0x1244	0x00	R/W	GPIO output of GPIO pin39 ~ GPIO pin32
GPIO_OUT5	0x1245	0x00	R/W	GPIO output of GPIO pin47 ~ GPIO pin40
GPIO_OUT6	0x1246	0x00	R/W	GPIO output of GPIO pin55 ~ GPIO pin48
GPIO_IN0	0x1248	0x00	R/O	GPIO input of GPIO pin7 ~ GPIO pin0
GPIO_IN1	0x1249	0x00	R/O	GPIO input of GPIO pin15 ~ GPIO pin8
GPIO_IN2	0x124a	0x00	R/O	GPIO input of GPIO pin23 ~ GPIO pin16
GPIO_IN3	0x124b	0x00	R/O	GPIO input of GPIO pin31 ~ GPIO pin24
GPIO_IN4	0x124c	0x00	R/O	GPIO input of GPIO pin39 ~ GPIO pin32
GPIO_IN5	0x124d	0x00	R/O	GPIO input of GPIO pin47 ~ GPIO pin40
GPIO_IN6	0x124e	0x00	R/O	GPIO input of GPIO pin55 ~ GPIO pin48

12.1 GPIO Control and Statues Registers

Register name	Offset	Initial Value	Read Write	Description
IOCFG[7:0]	0x12a0	0x00	R/W	[1:0]gpio0_cfg [3:2]gpio1_cfg [5:4]gpio2_cfg [7:6] gpio3_cfg
IOCFG[15:8]	0x12a1	0x00	R/W	[10:8]gpio4_cfg [12:11]gpio5_cfg [14:13]gpio6_cfg [15]gpio7_cf<0>
IO_CFG[23:16]	0x12a2	0x00	R/W	[17:16]gpio7_cfg<2:1> [20:18]gpio8_cfg [23:21]gpio9_cfg
IO_CFG[31:24]	0x12a3	0x00	R/W	[25:24]gpio10_cfg [26]gpio11_cfg [27]dchg_n_in_sel [29:28]gpio12_cfg [31:30]gpio13_cfg<1:0>
IO_CFG[39:32]	0x12a4	0x00	R/W	[32]gpio13_cfg<2> [35:33]gpio14_cfg [37:36]gpio15_cfg [39:38]gpio16_cfg
IO_CFG[47:40]	0x12a5	0x00	R/W	[41:40]gpio17_cfg [44:42]gpio18_cfg [47:45]gpio19_cfg
IO_CFG[55:48]	0x12a6	0x00	R/W	[50:48]gpio20_cfg [53:51]gpio21_cfg [55:54]gpio22_cfg
IO_CFG[63:56]	0x12a7	0x00	R/W	[57:56]gpio23_cfg [58]gpio24_cfg [60:59]gpio25_cfg [62:61]gpio26_cfg [63]gpio27_cfg<0>

12.2 Function Pin Selection

IO_CFG[71:64]	0x12a8	0x00	R/W	[64]gpio27_cfg<1> [65]gpio30_cfg [66]gpio31_cfg [67]gpio32_cfg [68]gpio33_cfg [70:69]gpio34_cfg [71]gpio35_cfg
IO_CFG[79:72]	0x12a9	0x00	R/W	[73:72]gpio36_cfg [74]gpio37_cfg [75]gpio38_cfg [76]gpio39_cfg [77]gpio40_cfg [78]gpio41_cfg [79]gpio42_cfg
IO_CFG[87:80]	0x12aa	0x00	R/W	[80]gpio43_cfg [81]gpio44_cfg [82]gpio45_cfg [83]gpio46_cfg<0> [85]gpio48_cfg [87:86]gpio49_cfg
IO_CFG[95:88]	0x12ab	0x00	R/W	[88]gpio46_cfg<1>

12.3 GPIO and functional pin mapping table

IO	Function 1(default)	Function 2	Function 3	Function 4	Function 5	Function 6
TPIO0	TPIO0 input (0x12a0[1:0]=2'b10)	TPIO0 output (0x12a0[1:0]=2'b00)	GPIO0 (0x12a0[1:0]=2'b11)			
TPIO1	TPIO1 output (0x12a0[3:2]=2'b00)	TPIO1 input (0x12a0[3:2]=2'b10)	GPIO1 (0x12a0[3:2]=2'b11)			
TPIO2	TPIO2 output (0x12a0[5:4]=2'b00)	TPIO2 input (0x12a0[5:4]=2'b10)	GPIO2 (0x12a0[5:4]=2'b11)			
TPIO3	TPIO3 output (0x12a0[7:6]=2'b00)	TPIO3 input (0x12a0[7:6]=2'b10)	GPIO3 (0x12a0[7:6]=2'b11)			
TPIO4	TPIO4 input (0x12a1[2:0]=3	TPIO4 output (0x12a1[2:0]=	GPIO4 (0x12a1[2:0]	SWCLK input	FRAME_START	

	'b100)	3'b000)	=3'b111)	(0x12a1[2:0] =3'b110)	(0x12a1[2:0] =3'b101)	
TPIO5	TPIO5 input (0x12a1[4:3]=2 'b01)	TPIO5 output (0x12a1[4:3]= 2'b00)	GPIO5 (0x12a1[4:3] =2'b11)	SWDIO (0x12a1[4:3] =2'b10)		
TPIO6	TPIO6 input (0x12a1[6:5]=2 'b01)	TPIO6 output (0x12a1[6:5]= 2'b00)	GPIO6 (0x12a1[6:5] =2'b11)	UART_TXD output (0x12a1[6:5] =2'b10)		
TPIO7	TPIO7 input (0x12a2[1:0]=2 'b10, 0x12a1[7]=1'b 0)	TPIO7 output (0x12a2[1:0]= 2'b00, 0x12a1[7]=1'b 0)	GPIO7 (0x12a2[1:0] =2'b11, 0x12a1[7]=1' b1)	UART_RXD input (0x12a2[1:0] =2'b11, 0x12a1[7]=1' b0)	DCHG_N_IN input (0x12a2[1:0] =2'b10, 0x12a1[7]=1' b1)	
TPIO8	TPIO8 input (0x12a2[4:2]=3 'b011)	TPIO8 output (0x12a2[4:2]= 3'b000)	GPIO8 (0x12a2[4:2] =3'b111)	TSI_FRAME _POL input (0x12a2[4:2] =3'b100)	TSI_FRAME _POL output (0x12a2[4:2] =3'b110)	
TPIO9	TPIO9 input (0x12a2[7:5]=3 'b010)	TPIO9 output (0x12a2[7:5]= 3'b000)	GPIO9 (0x12a2[7:5] =3'b111)	TSI_VDE input (0x12a2[7:5] =3'b110)	TSI_VDE output (0x12a2[7:5] =3'b101)	LSYNC_ OUT output (0x12a2[7 :5]=3'b10 0)
VSYN C_OUT	TSI_HSYNC input (0x12a3[1:0]=2 'b00)	TSI_HSYNC output (0x12a3[1:0]= 2'b01)	GPIO10 (0x12a3[1:0] =2'b11)	VSYNC_OU T output (0x12a3[1:0] =2'b10)		
BSYN C0_IN	BSYNC0_IN input/ HPD_IN input/ BL_PWM_DIM input (0x12a3[2]=1'b 0)		GPIO11 (0x12a3[2]=1 'b1)			
BSYN C1_IN	BSYNC1_IN input/ LSYNC_IN input/ DCHG_N_IN		GPIO12 (0x12a3[5:4] =2'b11)			

	input (0x12a3[5:4]=2'b00)					
BSYNC0	TSI_VDE_VALID input (0x12a4[0]=1'b0, 0x12a3[7:6]=2'b00)	TSI_VDE_VALID output (0x12a4[0]=1'b0, 0x12a3[7:6]=2'b11)	GPIO13 (0x12a4[0]=1'b1, 0x12a3[7:6]=2'b11)	BSYNC0 output (0x12a4[0]=1'b1, 0x12a3[7:6]=2'b10)	VSYNC output (0x12a4[0]=1'b1, 0x12a3[7:6]=2'b01)	DIV_PCLK output (0x12a4[0]=1'b1, 0x12a3[7:6]=2'b00)
BSYNC1	TSI_RX_VALID input (0x12a4[3:1]=3'b000)	TSI_RX_VALID output (0x12a4[3:1]=3'b011)	GPIO14 (0x12a4[3:1]=3'b111)	BSYNC1 output (0x12a4[3:1]=3'b110)	PWMO_B output (0x12a4[3:1]=3'b100)	
BSYNC0BIS	TSI_ABV_ON_REQ input (0x12a4[5:4]=2'b00)	TSI_ABV_ON_REQ output (0x12a4[5:4]=2'b01)	GPIO15 (0x12a4[5:4]=2'b11)	BSYNC0BIS output (0x12a4[5:4]=2'b10)		
BSYNC1BIS	TSI_LIVE_ON_REQ input (0x12a4[7:6]=2'b00)	TSI_LIVE_ON_REQ output (0x12a4[7:6]=2'b01)	GPIO16 (0x12a4[7:6]=2'b11)	BSYNC1BIS output (0x12a4[7:6]=2'b10)		
VRR_CTL0	VRR_CTL0 output (0x12a5[1:0]=2'b10)	SPIM2_SCLK output (0x12a5[1:0]=2'b00)	GPIO17 (0x12a5[1:0]=2'b11)	DEBUG0 output (0x12a5[1:0]=2'b01)		
VRR_CTL1	VRR_CTL1 output (0x12a5[4:2]=3'b110)	SPIM2_MOSI (0x12a5[4:2]=3'b101)	GPIO18 (0x12a5[4:2]=3'b111)	DEBUG1 output (0x12a5[4:2]=3'b100)	BL_EN output (0x12a5[4:2]=3'b000)	
VRR_CTL2	VRR_CTL2 output (0x12a5[7:5]=3'b110)	SPIM2_CS0 output (0x12a5[7:5]=3'b101)	GPIO19 (0x12a5[7:5]=3'b111)	DEBUG2 output (0x12a5[7:5]=3'b100)	PWMO_A output (0x12a5[7:5]=3'b000)	
VRR_CTL3	GPIO20 (0x12a6[2:0]=3'b111 ,default input)	VRR_CTRL3 output (0x12a6[2:0]=3'b110)	LSYNC output (0x12a6[2:0]=3'b101)	DEBUG3 output (0x12a6[2:0]=3'b100)	SPIM2_MISO (0x12a6[2:0]=3'b000)	
TR_CLK	TR_CLK output (0x12a6[5:3]=3	SPIM2_CS1 output (0x12a6[5:3]=	GPIO21 (0x12a6[5:3]=3'b111)	DEBUG4 output (0x12a6[5:3]	VFB_CLK output (0x12a6[5:3]	

	'b101)	3'b000)		=3'b100)	=3'b110)	
PM_VBLANK_N	PM_VBLANK_N output/ HPD_OUT output (0x12a6[7:6]=2'b10, PCFG_HPD_MAS_SEL)	PM_VBLANK_N output (0x12a6[7:6]=2'b01)	GPIO22 (0x12a6[7:6]=2'b11)	HPD_OUT output (0x12a6[7:6]=2'b00)		
PM_FLT_N	PM_FLT_N input (0x12a7[1:0]=2'b10)	IRQ_OUT output (0x12a7[1:0]=2'b00)	GPIO23 (0x12a7[1:0]=2'b11)			
VCS_FLT_N	VCS_FLT_N input/ VSYNC_IN input/ INT_N_IN input (0x12a7[2]=1'b0)		GPIO24 (0x12a7[2]=1'b1)			
MSYNC	MSYNC output (0x12a7[4:3]=2'b10)	DCHG_N_OUT output (0x12a7[4:3]=2'b01)	GPIO25 (0x12a7[4:3]=2'b11)	REFCLK_OUT output (0x12a7[4:3]=2'b00)		
VCOM_SEL0	GPIO26 (0x12a7[6:5]=2'b11, default input)	VCOM_SEL0 output (0x12a7[6:5]=2'b00)		DEBUG6 output (0x12a7[6:5]=2'b10)		
VCOM_SEL1	GPIO27 (0x12a8[0]=1'b1, 0x12a7[7]=1'b1, default input)	VCOM_SEL1 output (0x12a8[0]=1'b0, 0x12a7[7]=1'b0)	FRAME_START output (0x12a8[0]=1'b0, 0x12a7[7]=1'b1)	DEBUG7 output (0x12a8[0]=1'b1, 0x12a7[7]=1'b0)		
CSCL0	CSCL0		GPIO28 input			
CSDA0	CSDA0		GPIO29 input			

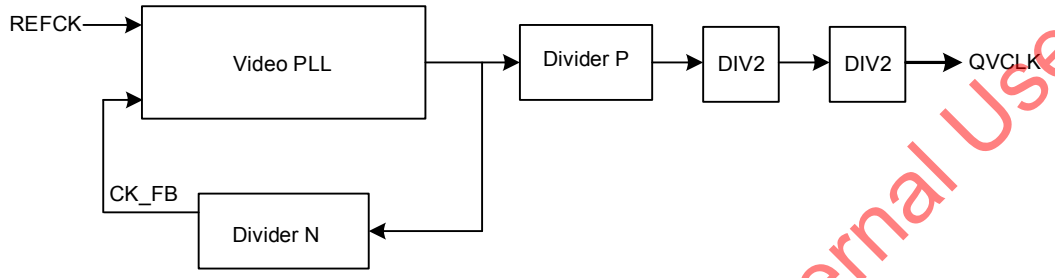
SPIS1_SCLK	SPIS1_SCLK input (0x12a8[1]=1'b0)		GPIO30 (0x12a8[1]=1'b1)			
SPIS1_CS	SPIS1_CS input (0x12a8[2]=1'b0)		GPIO31 (0x12a8[2]=1'b1)			
SPIS1_MOSI	SPIS1_MOSI (0x12a8[3]=1'b0)		GPIO32 (0x12a8[3]=1'b1)			
SPIS1_MISO	SPIS1_MISO (0x12a8[4]=1'b0)		GPIO33 (0x12a8[4]=1'b1)			
SPIM0_SCLK	SPIM0_SCLK output/ I2CM0_SCL output (0x12a8[6:5]=2'b10, PCFG_MAS0_SEL)	SPIM0_SCLK output (0x12a8[6:5]=2'b00)	GPIO34 (0x12a8[6:5]=2'b11)	I2CM0_SCL output (0x12a8[6:5]=2'b01)		
SPIM0_CS	SPIM0_CS output (0x12a8[7]=1'b0)		GPIO35 (0x12a8[7]=1'b1)			
SPIM0_MOSI	SPIM0_MOSI/ I2CM0_SDA (0x12a9[1:0]=2'b10, PCFG_MAS0_SEL)	SPIM0_MOSI (0x12a9[1:0]=2'b00, PCFG_MAS0_SEL)	GPIO36 (0x12a9[1:0]=2'b11)	I2CM0_SDA (0x12a9[1:0]=2'b01)		
SPIM0_MISO	SPIM0_MISO (0x12a9[2]=1'b0)		GPIO37 (0x12a9[2]=1'b1)			
SPIM1_SCLK	SPIM1_SCLK output (0x12a9[3]=1'b0)		GPIO38 (0x12a9[3]=1'b1)			
SPIM1	SPIM1_CS0		GPIO39			

_CS0	output (0x12a9[4]=1'b0)		(0x12a9[4]=1'b1)			
SPIM1_CS1	SPIM1_CS1 output (0x12a9[5]=1'b0)		GPIO40 (0x12a9[5]=1'b1)			
SPIM1_MOSI	SPIM1_MOSI (0x12a9[6]=1'b0)		GPIO41 (0x12a9[6]=1'b1)			
SPIM1_MISO	SPIM1_MISO (0x12a9[7]=1'b0)		GPIO42 (0x12a9[7]=1'b1)			
MSCL3	MSCL3 (0x12aa[0]=1'b0)		GPIO43 (0x12aa[0]=1'b1)			
MSDA3	MSDA3 (0x12aa[1]=1'b0)		GPIO44 (0x12aa[1]=1'b1)			
NVM_WP	NVM_WP output (0x12aa[2]=1'b0)		GPIO45 (0x12aa[2]=1'b1)			
CD_VBLANK	GPIO46 (0x12ab[0]=1'b1, 0x12aa[3]=1'b1, default input)	CD_VBLANK output (0x12ab[0]=1'b0, 0x12aa[3]=1'b0)		DEBUG5 output (0x12ab[0]=1'b1, 0x12aa[3]=1'b0)		
DP_HPD	HPD output (0x12aa[5]=1'b0)		GPIO48 (0x12aa[5]=1'b1)			
BISTEN	BISTEN input (0x12aa[7:6]=2'b01)	BISTEN output (0x12aa[7:6]=2'b10)	GPIO49 (0x12aa[7:6]=2'b11)			

13. Clock Control

13.1 Video PLL Control

The following diagram illustrate the detailed Video PLL structure.



Video PLL can be configured by the following registers.

Register name	Offset	Initial Value	Read Write	Description
REG_PLL_CTRL0	0x0e31	0x00	R/W	[7:0]=VDPLL NF[7:0]
REG_PLL_CTRL0	0x0e32	0x00	R/W	[4:0]=VDPLL NF[12:8] [7:5]=VDPLL NI[2:0]
REG_PLL_CTRL0	0x0e33	0x00	R/W	[4:0]=VDPLL NI[7:3] [7:6]=VDPLL BAND[1:0]
REG_PLL_CTRL0	0x0e30	0x00	R/W	[0] REG_VPLL_START [1] REG_VPLL_START_SEL [2] REG_VPLL_LOAD [3] REG_VPLL_LOAD_SEL [7:4]Reserved

The following equation can be used to calculate the value of VPLL_BAND:

$$VPLL\_DivN = \frac{QVCLK \times 4 \times P \times 8192}{27 \times 10^6}$$

$$N[20:0] = Dec\ to\ Hex[2b'00, (VPLL\_DivN)]$$

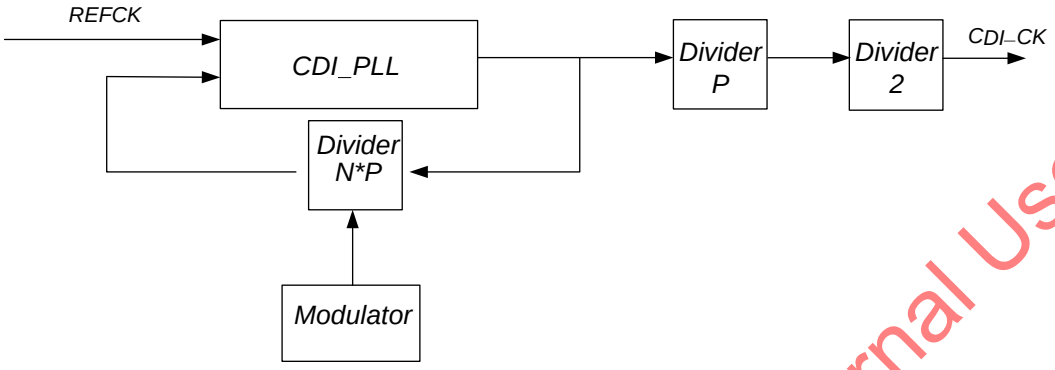
N [12:0] = REG_VPLL_NF [12:0] is fraction part and N [20:13] = REG_VPLL_NI [7:0] is integer part.

The VPLL_BAND (P) is defined in the following table:

QVCLK (MHz)	VPLL_BAND (Divider P)
28.6875M ~ 57.375M	8 (00)
57.375M ~ 114.75M	4 (01)
114.75M ~ 229.5M	2 (10)
229.5M ~ 459M	1 (11)

13.2 CDI PLL Control

The following diagram illustrate the detailed CDI PLL structure.



CDI PLL can be configured by the following registers.

Register name	Offset	Initial Value	Read Write	Description
REG_DPCD_OFF61	0x0361	0x00	R/W	0:CDI agile clocking disabled
REG_ALPM_CTRL0 [31:24]	0x05b3	0x00	R/W	[1:0]REG_ALPM_EN<1:0> [2]REG_ALPM_CMDSEL [3]REG_VDE_DUMMY_MASK [4]REG_ALPM_EN<2>
REG_PLL_CTRL4	0x0e41	0xff	R/W	[15:8] REG_COGIF_EN
REG_PLL_CTRL4	0x0e43	0xff	R/W	[5] cogpll_ckin_sel
REG_PLL_CTRL5	0x0e44	0xff	R/W	[1] cogpll_ssc_en [0] REG_VPLL_SS_EN
REG_PLL_CTRL4	0x0e40	0xff	R/W	[0] REG_CDPLL_START [1] REG_CDPLL_START_SEL [2] REG_CDPLL_LOAD [3] REG_CDPLL_LOAD_SEL
REG_PLL_CTRL13	0x0e65	0x00	R/W	[7:0] CDPLL_DIV_N[7:0]
REG_PLL_CTRL13	0x0e66	0x00	R/W	[7:0] CDPLL_DIV_N[15:8]
REG_PLL_CTRL13	0x0e67	0x00	R/W	[1:0]CDPLL_DIV_N[17:16] [3:2] CDIPLL_BAND_P 855 use LCO, ignore to set this bit [6:4] CDIPLL_POS_SEL 000:sec1 001:sec2….110:sec7 111:sec8
REG_PLL_CTRL6	0x0e48	0x00	R/W	[7:0] CALI0_VAL
REG_PLL_CTRL6	0x0e49	0x00	R/W	[15:8] CALI0_VAL
REG_PLL_CTRL6	0x0e4a	0x00	R/W	[7:0] CALI1_VAL

REG_PLL_CTRL6	0x0e4b	0x00	R/W	[15:8] CALI1_VAL
REG_PLL_CTRL7	0x0e4c	0x00	R/W	[7:0] CALI2_VAL
REG_PLL_CTRL7	0x0e4d	0x00	R/W	[15:8] CALI2_VAL
REG_PLL_CTRL7	0x0e4e	0x00	R/W	[7:0] CALI3_VAL
REG_PLL_CTRL7	0x0e4f	0x00	R/W	[15:8] CALI3_VAL
REG_PLL_CTRL8	0x0e50	0x00	R/W	[7:0] CALI4_VAL
REG_PLL_CTRL8	0x0e51	0x00	R/W	[15:8] CALI4_VAL
REG_PLL_CTRL8	0x0e52	0x00	R/W	[7:0] CALI5_VAL
REG_PLL_CTRL8	0x0e53	0x00	R/W	[15:8] CALI5_VAL
REG_PLL_CTRL9	0x0e54	0x00	R/W	[7:0] CALI6_VAL
REG_PLL_CTRL9	0x0e55	0x00	R/W	[15:8] CALI6_VAL
REG_PLL_CTRL9	0x0e56	0x00	R/W	[7:0] CALI7_VAL
REG_PLL_CTRL9	0x0e57	0x00	R/W	[15:8] CALI7_VAL
REG_PLL_CTRL10	0x0e58	0x00	R/W	[3:0] REG_CDI_CALI0<19:16> [7:4] REG_CDI_CALI1<19:16>
REG_PLL_CTRL10	0x0e59	0x00	R/W	[3:0] REG_CDI_CALI2<19:16> [7:4] REG_CDI_CALI3<19:16>
REG_PLL_CTRL10	0x0e5a	0x00	R/W	[3:0] REG_CDI_CALI4<19:16> [7:4] REG_CDI_CALI5<19:16>
REG_PLL_CTRL10	0x0e5b	0x00	R/W	[3:0] REG_CDI_CALI6<19:16> [7:4] REG_CDI_CALI7<19:16>
REG_RPIO[223:216]	0x0f1b	0x40	R/W	[7] COGPLL calib enable [6]COGPLL calib_en select signal 0: CALIB_EN = REG_CALIB_EN. 1: CALIB_EN= PLL_START.
REG_RPIO[215:208]	0x0f1a	0x00	R/W	[7] COGPLL AGILE enable. 0: LCO calib one time. 1: LCO calib eight times. [6] COGPLL calibration timeout wait time set 0 : 128* T(osck) 1 : 1024*T(osck) [5:4] wait time for LCVCO to settle at steady state before frequency measurement during 00 : 5us 01 : 10us 10 : 25us 11 : 2.5us

The following equation can be used to calculate the value of N

$$N = \frac{CDI\_CK \times 2 \times 1024}{27 \times 10^6}$$

$$REG\_CDIPLL\_N[17:0] = Dec\ to\ Hex\ [N]$$

REG_CDIPLL_N [9:0] is fraction part of N and REG_CDIPLL_N [17:10] is integer part of N.

The following sequence is used to manually set the CDI PLL in LCO mode:

1. Disable CDI Agile clocking :write 0x0361=0x00 ,0x05b3=0x04, and 0x0e3a=0xb0
2. Disable HW CDI PLL start: write 0x0e40=0x02
3. Set CDI PLL calibration target frequency: N=0x0e58[3:0]+0x0e49[7:0]+0x0e48[7:0]
4. Set CDI PLL value: N= 0x0e67[1:0]+0x0e66[7:0]+0x0e65[7:0], and 0x0e67[7]=1
5. Set CDIPLL calibration EN: write 0x0f1a=0x10 and 0x0f1b=0x40
6. Set SW CDIPLL start: write 0x0e40=0x1b
7. Set SW CDIPLL load: write 0x0e40=0x1f

Sample code for 900Mbps CDI data rate:

```
WriteReg(0x03,0x61,0x00)
WriteReg(0x05,0xb3,0x04)
WriteReg(0x0e,0x3a,0xb0)
WriteReg(0x0e,0x40,0x02)

WriteReg(0x0e,0x48,0x55)        #N=900*1024/27=68266=0x8555
WriteReg(0x0e,0x49,0x85)
WriteReg(0x0e,0x58,0x00)

WriteReg(0x0e,0x65,0x55)        #N=900*1024/27=68266=0x8555
WriteReg(0x0e,0x66,0x85)
WriteReg(0x0e,0x67,0x80)

WriteReg(0x0f,0x1a,0x10)
WriteReg(0x0f,0x1b,0x40)
WriteReg(0x0e,0x40,0x1b)
WriteReg(0x0e,0x40,0x1f)
```