JZ4725B

Mobile Application Processor

Programming Manual

Release Date: Dec. 14, 2010

北京君正集成电路股份有限公司
Ingenic Semiconductor Co.,Ltd.

JZ4725B Mobile Application Processor

Programming Manual

Disclaimer

Ingenic Semiconductor Co., Ltd.

Room 108, Building A, Information Center, Zhongguancun Software Park
8 Dongbeiwang West Road, Haidian District, Beijing, China,
Tel: 86-10-82826661
Fax: 86-10-82825845
Http: //www.ingenic.cn

CONTENTS

TABLES

FIGURES

1 Overview

JZ4725B is a cost effective SOC solution for multimedia rich and mobile devices like video MP3, MP4 and PMP like products.

At the heart of JZ4725B is XBurst CPU core. XBurst is an industry leading microprocessor core which delivers superior high performance and best-in-class low power consumption.

The SIMD instruction set implemented by XBurst core, in together with the video post processing unit, provides RMVB, MPEG-1/2/4 decoding capability up to D1 resolution.

The memory interface of JZ4725B supports a variety of memory types that allow flexible design requirements, including glueless connection to SLC NAND flash memory and 4-bit/8-bit/12-bit ECC MLC NAND flash memory for cost sensitive applications.

On-chip modules such as LCD controller, audio CODEC, SAR-ADC and I2S controller offer designers a rich suite of peripherals for multimedia application. Other peripherals such as I2C, UART, USB 2.0 device controller, MMC/SD/SDIO host controller and general system resources provide enough computing and connectivity capability for many applications.

1.1 Block Diagram

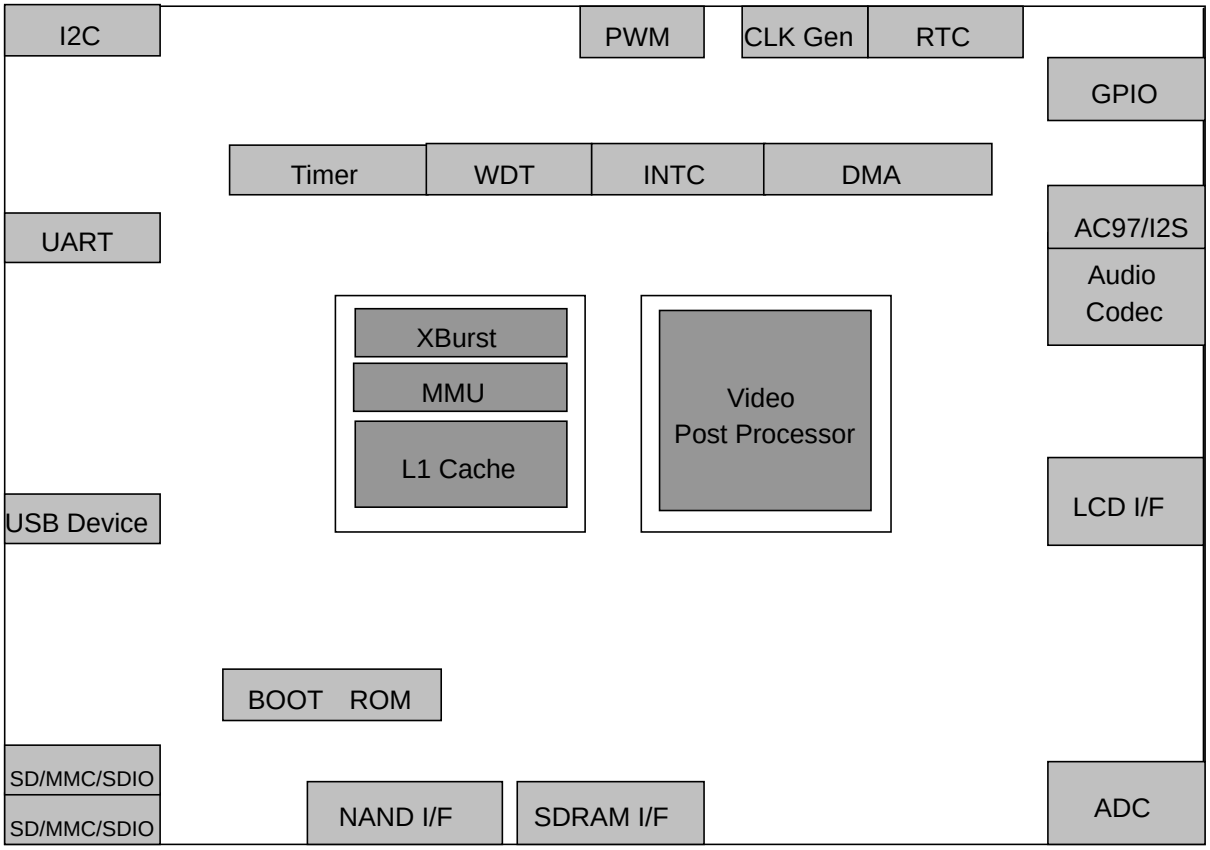


Figure 1-1 JZ4725B Diagram

1.2 Features

1.2.1 CPU Core

- XBurst CPU
  - XBurst® RISC instruction set to support Linux and WinCE
  - XBurst® SIMD instruction set to support multimedia acceleration
  - XBurst® 8-stage pipeline micro-architecture up to 360MHz
- MMU
  - 32-entry dual-pages joint-TLB
  - 4 entry Instruction TLB
  - 4 entry data TLB
- Cache
  - 16K instruction cache
  - 16K data cache
- Hardware debug support

1.2.2 Memory Sub-systems

- NAND flash interface
  - Support 4-bit/8-bit/12-bit MLC NAND as well as SLC NAND
  - Support all 8-bit/16-bit NAND Flash devices regardless of density and organization
  - Support automatic boot up from NAND Flash devices
- Synchronous DRAM interface
  - Standard SDRAM
  - 1 banks with programmable size and base address
  - 16-bit data bus width
  - Multiplexes row/column addresses according to SDRAM capacity
  - Two-bank or four-bank SDRAM is supported
  - Supports auto-refresh and self-refresh functions
  - Supports power-down mode to minimize the power consumption of SDRAM
  - Supports page mode
  - 1 Chip selects
- BCH controller
  - Implement data ECC encoding and decoding
- Direct memory access controller
  - Six independent DMA channels
  - Descriptor supported
  - Transfer data units: 8-bit, 16-bit, 32-bit, 16-byte or 32-byte
  - Transfer requests can be: auto-request within DMA; and on-chip peripheral module request
  - Interrupt on transfer completion or transfer error
  - Supports two transfer modes: single mode or block mode
  - External DMA supported

- The XBurst processor system supports little endian only

1.2.3 AHB Bus Arbiter

- Provide a fair chance for each AHB master to possess the AHB bus
- Fulfill the back-to-back feature of AHB protocol
- Divide two master groups with different privileges supports two arbitrating methods: Round-robin possession for masters in the same group, Preemptive possession for

1.2.4 masters with higher privileges System Devices

- Clock generation and power management
  - On-chip oscillator circuit for an 32768Hz clock and an 12MHz clock
  - On-chip phase-locked loops (PLL) with programmable multiple-ratio. Internal counter are used to ensure PLL stabilize time
  - PLL on/off is programmable by software
  - ICLK, PCLK, SCLK, MCLK and LCLK frequency can be changed separately for software by setting division ratio
  - Supports six low-power modes and function: NORMAL mode; DOZE mode; IDLE mode; SLEEP mode; HIBERNATE mode; and MODULE-STOP function.
- RTC (Real Time Clock)
  - 32-bit second counter
  - 1Hz from 32768hz
  - Alarm interrupt
  - Independent power
  - A 32-bits scratch register used to indicate whether power down happens for RTC power
- Interrupt controller
  - Total 32 maskable interrupt sources from on-chip peripherals and external request through GPIO ports
  - Interrupt source and pending registers for software handling
  - Unmasked interrupts can wake up the chip in sleep or standby mode
- Timer and counter unit with PWM output
  - Provide five separate channels
  - 16-bit A counter and 16-bit B counter with auto-reload function every channel
  - Support interrupt generation when the A counter underflows
  - Three clock sources: RTCLK (real time clock), EXCLK (external clock input), PCLK (APB Bus clock) selected with 1, 4, 16, 64, 256 and 1024 clock dividing selected
  - Six PWM outputs
- OS timer
  - One channel
  - 32-bit counter and 32-bit compare register
  - Support interrupt generation when the counter matches the compare register
  - Three clock sources: RTCLK (real time clock), EXCLK (external clock input), PCLK (APB

Bus clock) selected with 1, 4, 16, 64, 256 and 1024 clock dividing selected

- Watchdog timer
  - 16-bit counter in RTC clock with 1, 4, 16, 64, 256 and 1024 clock dividing selected
  - Generate power-on reset

1.2.5 Audio/Display/UI Interfaces

- LCD controller
  - Single-panel display in active mode, and single- or dual-panel displays in passive mode
  - 2, 4, 16 grayscales and up to 4096 colors in STN mode
  - 2, 4, 16, 256, 4K, 32K, 64K, 256K and 16M colors in TFT mode
  - 24-bit data bus
  - Support 1,2,4,8 pins STN panel, 16bit, 18bit TFT and 8bit I/F TFT
  - Display size up to 1280×1024 pixels
  - 256×16 bits internal palette RAM
  - Support ITU601/656 data format
  - Support smart LCD (SRAM-like interface LCD module)
  - Support delta RGB
  - One single color background and two foreground OSD
- Image post processor
  - Video frame resize
  - Color space conversion: 420/444/422 YUV to RGB convert
- On-chip audio CODEC
  - 24-bit DAC, SNR: 90dB
  - 24-bit ADC, SNR: 85dB
  - Sample rate: 8/9.6/11.025/12/16/22.05/24/32/44.1/48/96kHz
  - L/R channels line input
  - MIC input
  - L/R channels headphone output amplifier support up to 16ohm load
  - Capacitor-coupled
- AC97/I2S controller
  - Supports 8, 16, 18, 20 and 24 bit for sample for AC-link and I2S/MSB-Justified format
  - DMA transfer mode support
  - Support variable sample rate mode for AC-link format
  - Power down mode and two wake-up mode support for AC-link format
  - Programmable Interrupt function support
  - Support the on-chip CODEC
  - Support off-chip CODEC
- SADC
  - 12-bit, 2Mbps, SNR@500kHz is 61dB, THD@500kHz is -71dB
  - Battery voltage input
  - 1 generic input channel

1.2.6 On-chip Peripherals

- General-Purpose I/O ports
  - Total 84 GPIOs.
  - Each pin can be configured as general-purpose input or output or multiplexed with internal chip functions
  - Each pin can act as a interrupt source and has configurable rising/falling edge or high/low level detect manner, and can be masked independently
  - Each pin can be configured as open-drain when output
  - Each pin can be configured as internal resistor pull-up
- I2C bus interface
  - Only supports single master mode
  - Supports I2C standard-mode and F/S-mode up to 400 kHz
  - Double-buffered for receiver and transmitter
  - Supports general call address and START byte format after START condition
- USB 2.0 device interface
  - Compliant with USB protocol revision 2.0
  - High speed and full speed supported
  - Embedded USB 2.0 PHY
- Two MMC/SD/SDIO controllers (MSC0, MSC1)
  - Support automatic boot up from MSC0
  - 4-bit data bus on MSC0, 1-bit data bus on MSC1
  - Compliant with “The MultiMediaCard System Specification version 4.2”
  - Compliant with “SD Memory Card Specification version 2.0” and “SDIO Card Specification version 1.0” with 1 command channel and 4 data channels
  - Up to 320 Mbps data rate on MSC0, 80Mbps data rate on MSC1
  - Supports up to 10 cards (including one SD card)
  - Maskable hardware interrupt for SD I/O interrupt, internal status, and FIFO status
- UART
  - 5, 6, 7 or 8 data bit operation with 1 or 1.5 or 2 stop bits, programmable parity (even, odd, or none)
  - 32x8bit FIFO for transmit and 32x11bit FIFO for receive data
  - Interrupt support for transmit, receive (data ready or timeout), and line status
  - Supports DMA transfer mode
  - Provide complete serial port signal for modem control functions
  - Support slow infrared asynchronous interface (IrDA)
  - IrDA function up to 115200bps baudrate
  - UART function up to 3.7Mbps baudrate
  - Hardware flow control
  - Only the TxD (transmit data) pin is available

1.2.7 Bootrom

- 4kB Boot ROM memory

1.3 Characteristic

Item	Characteristic
Process Technology	0.16um CMOS
Power supply voltage	I/O: 3.3 ± 0.3V Core: 1.8 ± 0.2
Package	LQFP 128, 14mm x 14mm x 1.4mm, 0.4mm pitch
Operating frequency	360MHz

2 CPU Core

At the heart of the chip, there is a XBurst® CPU processor core. XBurst® CPU adopts a brand new micro-architecture which provides superior performance and power consumption than existent industry cores. Detailed description of XBurst® CPU core is specified in document titled "XBurst® Microprocessor Core User Manual".

Key features of the XBurst® CPU core implemented in this chip are as following:

Table 2-1 The XBurst® CPU Core Features

Item	Features
RISC ISA	• Industry standard Instruction set architecture • 32 32-bit general purpose registers
Ingenic Media ISA	• Implement 114 SIMD like instructions for multimedia acceleration • See document "JZ SIMD Instruction Set"
Ingenic Floating Point ISA	• Not implemented
Multiply-Divide Unit (MDU)	• Maximum issue rate of one 32x16 multiply every clock • Maximum issue rate of one 32x32 multiply every other clock • Minimum 2 clock cycle, maximum 34 clock cycles for division
Memory Manager Unit (MMU)	• 4 G-Bytes of address space • 32/16 dual-entry full associative joint TLB plus 4 dual-entry ITLB and 4 dual-entry DTLB respectively • 7 different page size from 4KB to 16MB supported in any entry • Support entry lock • Space identifier ASID: 8 bits
Data Cache	• Physically-indexed, physically-tagged • 4 way, 8-word line, alterable size: 4K, 8K, 16K bytes • LRU replacement algorithm • Write-back, write-through • 16-word depth write buffer
Instruction Cache	• Physically-indexed, physically-tagged • 4 way, 8-word line, alterable size: 4K, 8K, 16K bytes • LRU replacement algorithm
Debug&JTAG	• JTAG interface to host machine • ACC mode to accelerate JTAG memory access • Two instruction and one data breakpoints
Branch Target Buffer (BTB)	• Virtally-tagged • Up to 64 entry direct mapped • 2-bit branch history maintained

Bus Interface	● compliance with AHB protocol
Tightly coupled sharing memory (TCSM)	● Not include
Dedicated DDMA	● Not include

3 External Memory Controller

3.1 Overview

The External Memory Controller (EMC) divides the off-chip memory space and outputs control signals complying with specifications of various types of memory and bus interfaces. It enables the connection of static memory, NAND flash memory, synchronous DRAM, etc., to this processor.

- Static memory interface
  - Direct interface to ROM, Burst ROM, SRAM and NOR Flash
  - Support 4 external chip selection CS4~1#. Each bank can be configured separately
  - The size and base address of static memory banks are programmable
  - Output of control signals allowing direct connection of memory to each bank. Write strobe setup time and hold time periods can be inserted in an access cycle to enable connection to low-speed memory
  - Wait state insertion can be controlled by program
  - Wait insertion by WAIT pin
  - Automatic wait cycle insertion to prevent data bus collisions in case of consecutive memory accesses to different banks, or a read access followed by a write access to the same bank

- NAND flash interface
  - Support on CS4~CS1, sharing with static memory bank4~bank1
  - Support most types of NAND flashes, including 8-bit and 16-bit bus width, 512B and 2KB page size. For 512B page size, 3 and 4 address cycles are supported. For 2KB page size, 4 and 5 address cycles are supported
  - Support read/erase/program NAND flash memory
  - Support boot from NAND flash

- SDRAM Interface
  - Support 2 chip selection DCS0# and DCS1#
  - Support both 32-bit and 16-bit bus width
  - Support both two-bank and four-bank type SDRAM
  - Support burst operation
  - Support both auto-refresh and self-refresh functions
  - The size and base address of each bank is configurable
  - Multiplexes row/column addresses according to SDRAM capacity
  - Controls timing of SDRAM direct-connection control signals according to register setting
  - Supports power-down mode to minimize the power consumption of SDRAM
  - Support page mode

3.2 Pin Description

Following table list the EMC pins.

Table 3-1 EMC Pin Description

Pin Name	I/O	Signal	Description
Data Bus	I/O	D31 – D0	Data I/O.
Address bus	O	A25–A0	Address output.
Static chip select 4 ~ 1	O	CS4~1#	Chip select signal that indicates the static bank being accessed.
SDRAM chip select	O	DCS0#	Chip select signal that indicates the SDRAM bank being accessed.
SDRAM chip select	O	DCS1#	Chip select signal that indicates the SDRAM bank being accessed.
Read enable	O	RD# /	For Static memory read enable signal.
Write enable	O	WE# /	Static memory write enable signal.
Column address strobe	O	CAS#	SDRAM column address strobe signal.
Row address strobe	O	RAS#	SDRAM row address strobe signal.
Read/write	O	RD/WR#	Data bus direction designation signal. Also used as SDRAM write enable signal.
Byte enable 0	O	WE0# / BE0# / DQM0 /	For non-byte-control static memory, D7-0 write enable signal. For byte-control static memory, D7-0 selection signal. For SDRAM, D7–D0 selection signal.
Byte enable 1	O	WE1# / BE1# / DQM1/	For non-byte-control static memory, D15-8 write enable signal. For byte-control static memory, D15-8 selection signal. For SDRAM, D15–D8 selection signal.
Byte enable 2	O	WE2# / BE2# / DQM2 /	For non-byte-control static memory, D23-16 write enable signal. For byte-control static memory, D23-16 selection signal. For SDRAM, D23–D16 selection signal.
Byte enable 3	O	WE3# / BE3# / DQM3	For static memory, D31-24 write enable signal. For byte-control static memory, D31-24 selection signal. For SDRAM, D31–D24 selection signal.
SDRAM Clock enable	O	CKE	Enable the SDRAM clock.
SDRAM Clock	O	CKO	SDRAM clock.
Wait	I	Wait# /	External wait state request signal for memory-like devices.
NAND flash read enable	O	FRE#	NAND flash read enable signal.
NAND flash write enable	O	FWE#	NAND flash write enable signal.
NAND flash ready/busy	I	FRB#	Indicates NAND flash is ready or busy. (When Nand flash boot, GPC30 is used as FRB# of CS1#)

3.3 Physical Address Space Map

Both virtual spaces and physical spaces are 32-bit wide in this architecture. Virtual addresses are translated by MMU into physical address which is further divided into several partitions for static memory, SDRAM, and internal I/O.

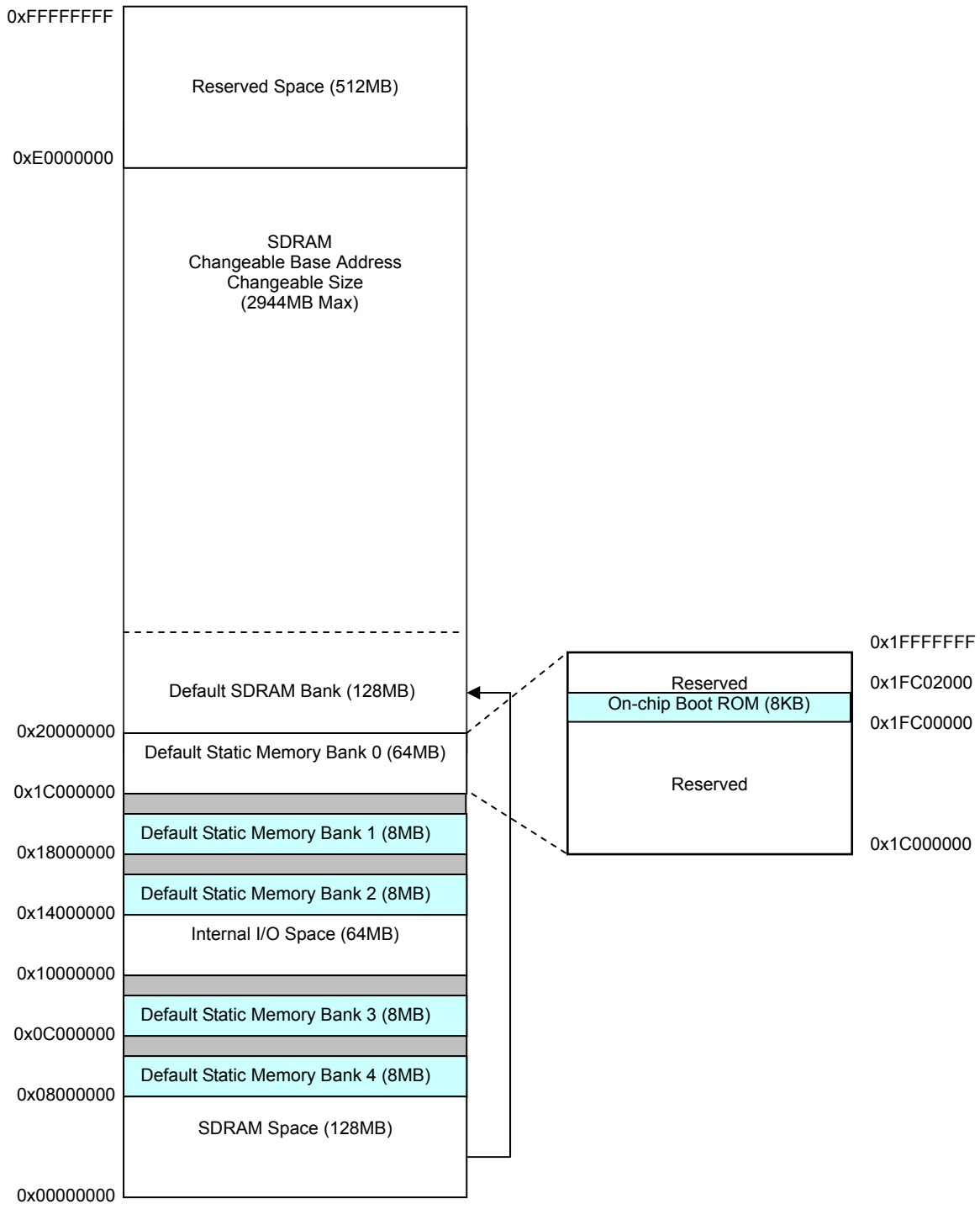


Figure 3-1 Physical Address Space Map

Table 3-2 Physical Address Space Map

Start Address	End Address	Connectable Memory	Capacity
H'0000 0000	H'07FF FFFF	SDRAM space	128 MB
H'0800 0000	H'0FFF FFFF	Static memory space	128 MB
H'1000 0000	H'13FF FFFF	Internal I/O space	64 MB
H'1400 0000	H'1BFF FFFF	Static memory space	128MB
H'1C00 0000	H'1FBF FFFF	Un-used	60MB
H'1FC0 0000	H'1FC0 1FFF	On-chip boot ROM	8KB
H'1FC0 1000	H'1FFF FFFF	Un-used	4095KB
H'2000 0000	H'BFFF FFFF	SDRAM space	2944 MB
H'D000 0000	H'FFFF FFFF	Reserved space	512 MB

The base address and size of each memory banks are configurable. Software can re-configure these memory banks according to the actual connected memories. Following table lists the default configuration after reset.

Table 3-3 Default Configuration of EMC Chip Select Signals

Chip-Select Signal	Connected Memory	Capacity	Memory Width *1	Start Address	End Address
CS1#	Static memory bank 1	8 MB	8, 16, 32	H'1800 0000	H'1BFF FFFF
CS2#	Static memory bank 2	8 MB	8, 16, 32	H'1400 0000	H'17FFFFFF
CS3#	Static memory bank 3	8 MB	8, 16, 32	H'0C00 0000	H'0FFF FFFF
CS4#	Static memory bank 4	8 MB	8, 16, 32	H'0800 0000	H'0BFF FFFF
DCS0#*3	SDRAM bank	128 MB	16, 32	H'2000 0000	H'27FF FFFF
DCS1#*3	SDRAM bank	128 MB	16, 32	H'2800 0000	H'2FFF FFFF

NOTES:

1 Data width of static memory banks can be configured to 8, 16 or 32 bits by software.

2 The 8KB address space from H'1FC00000 to H'1FC01FFF in bank 0 is mapped to on-chip boot ROM. The other memory spaces in bank 0 are not used.

3 To support large SDRAM space, EMC re-maps the physical address H'00000000-H07FFFFFF to H'20000000-H'27FFFFFF. Software must configure the SDRAM base address by the re-mapped address.

3.4 Static Memory Interface

The static memory controller provides a glueless interface to SRAM's, ROMs (PROMs/EPROMs/FLASH), dual port memory, IO devices, and many other peripherals devices. It can directly control up to 4 devices using four chip select lines. Additional devices may be supported through external decoding of the address bus. The Device Controller shares the data and address busses with the SDRAM controller. Thus, only one memory subsection (SDRAM, memory, or IO) can be active at any time.

Each chip select can directly access memory or IO devices that are 8-bits, 16-bits, or 32-bits wide. Each device connected to a chip select line has 2 associated registers that control its operation and the access timing to the external device. The Static Memory Control Register SMCRn specifies various configurations for the device. The Static Memory Address Configuration Register SACRn specifies the base address and size for each device, enabling any device to be located anywhere in the physical address range.

The static memory interface includes the following signals:

- Four chip selects, CS4~1#
- Twenty-six address signals, A25-A0
- One read enable, RD#
- One write enable, WE#
- Four byte enable, BE3~1#
- One wait pin, WAIT#

The SMT field in SMCRn registers specifies the type of memory and BW field specifies the bus width. BOOT_SEL[1:0] pin defines whether system boot from Nor or Nand flash and the page size when boot from Nand flash.

3.4.1 Register Description

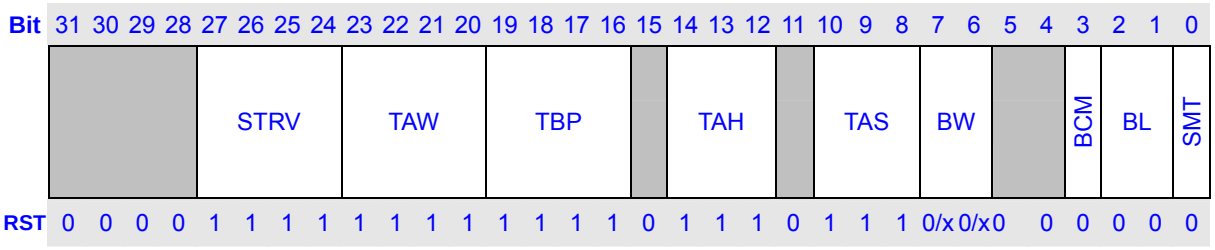
Table 3-4 Static Memory Interface Registers

Name	Description	RW	Reset Value	Address	Access Width
SMCR1	Static memory control register 1	RW	0x0FFF7700	0x13010014	32
SMCR2	Static memory control register 2	RW	0x0FFF7700	0x13010018	32
SMCR3	Static memory control register 3	RW	0x0FFF7700	0x1301001C	32
SMCR4	Static memory control register 4	RW	0x0FFF7700	0x13010020	32
SACR1	Static memory bank 1 address configuration register	RW	0x000018FC	0x13010034	32
SACR2	Static memory bank 2 address configuration register	RW	0x000016FE	0x13010038	32
SACR3	Static memory bank 3 address configuration register	RW	0x000014FE	0x1301003C	32
SACR4	Static memory bank 4 address configuration register	RW	0x00000CFC	0x13010040	32

3.4.1.1 Static Memory Control Register (SMCR1~4)

SMCR1~4 are 32-bit read/write registers that contain control bits for static memory. On reset, SMCR1~4 are initialized to 0x0FFF7700.

SMCR1 **0x13010014**
SMCR2 **0x13010018**
SMCR3 **0x1301001C**
SMCR4 **0x13010020**



Bits	Name	Description	RW
31:28	Reserved	Writes to these bits have no effect and always read as 0.	R
27:24	STRV	**Static Memory Recovery Time:** Its value is the number of idle cycles (0~15 cycles) inserted between bus cycles when switching from one bank to another bank or between a read access to a write access in the same bank. Its initial value is 0xF (15 cycles).	RW
23:20	TAW	**Access Wait Time:** For normal memory, these bits specify the number of wait cycles to be inserted in read strobe time. For burst ROM, these bits specify the number of wait cycles to be inserted in first data read strobe time.	RW

		TAW3~0 Wait cycle Wait# Pin 0000 0 cycle Ignored 0001 1 cycle Enabled 0010 2 cycles Enabled 0011 3 cycles Enabled 0100 4 cycles Enabled 0101 5 cycles Enabled 0110 6 cycles Enabled 0111 7 cycles Enabled 1000 8 cycles Enabled 1001 9 cycles Enabled 1010 10 cycles Enabled 1011 12 cycles Enabled 1100 15 cycles Enabled 1101 20 cycles Enabled 1110 25 cycles Enabled 1111 31 cycles Enabled (Initial Value)	
19:16	TBP	**Burst Pitch Time:** For burst ROM, these bits specify the number of wait cycles to be inserted in subsequent access. For normal memory, these bits specify the number of wait cycles to be inserted in write strobe time. **TBP3~0 Wait cycle Wait# Pin** 0000 0 cycle Ignored 0001 1 cycle Enabled 0010 2 cycles Enabled 0011 3 cycles Enabled 0100 4 cycles Enabled 0101 5 cycles Enabled 0110 6 cycles Enabled 0111 7 cycles Enabled 1000 8 cycles Enabled 1001 9 cycles Enabled 1010 10 cycles Enabled 1011 12 cycles Enabled 1100 15 cycles Enabled 1101 20 cycles Enabled 1110 25 cycles Enabled 1111 31 cycles Enabled (Initial Value)	RW
15	Reserved	Writes to these bits have no effect and always read as 0.	R
14:12	TAH	**Address Hold Time:** These bits specify the number of wait cycles to be inserted from negation of read/write strobe to address. **TAH2~0 Wait cycle** 000 0 cycle 001 1 cycle	RW

		010 2 cycles 011 3 cycles 100 4 cycles 101 5 cycles 110 6 cycles 111 7 cycles (Initial Value)	
11	Reserved	Writes to these bits have no effect and always read as 0.	R
10:8	TAS	**Address Setup Time:** These bits specify the number of wait cycles (0~7 cycles) to be inserted from address to assertion of read/write strobe. **TAS2~0 Wait cycle** 000 0 cycle 001 1 cycle 010 2 cycles 011 3 cycles 100 4 cycles 101 5 cycles 110 6 cycles 111 7 cycles (Initial Value)	RW
7:6	BW	**Bus Width :** These bits specify the bus width. this filed is writeable and are initialized to 0 by a reset. **BW1~0 Bus Width** 00 8 bits (Initial Value) 01 16 bits 10 32 bits 11 Reserved	RW
5:4	Reserved	Writes to these bits have no effect and always read as 0.	R
3	BCM	**SRAM Byte Control Mode (BCM):** When SRAM is connected; this bit specifies the type of SRAM. This bit is only valid when SMT is set to 0. **BCM Description** 0 SRAM is set to normal mode (Initial Value) 1 SRAM is set to byte control mode	RW
2:1	BL	**Burst Length (BL1, BL0):** When Burst ROM is connected; these bits specify the number of burst in an access. These bits are only valid when SMT is set to 1. **BL1~0 Burst Length** 00 4 consecutive accesses. Can be used with 8-, 16-, or 32-bit bus width (Initial Value) 01 8 consecutive accesses. Can be used with 8-, 16-, or 32-bit bus width 10 16 consecutive accesses. Can only be used with 8- or 16-bit bus width. Do not specify for 32-bit bus width 11 32 consecutive accesses. Can only be used with 8-bit bus width	

0	SMT	**Static Memory Type (SMT):** This bit specifies the type of static memory. **SMT** **Description** 0 Normal Memory (Initial Value) 1 Burst ROM	RW

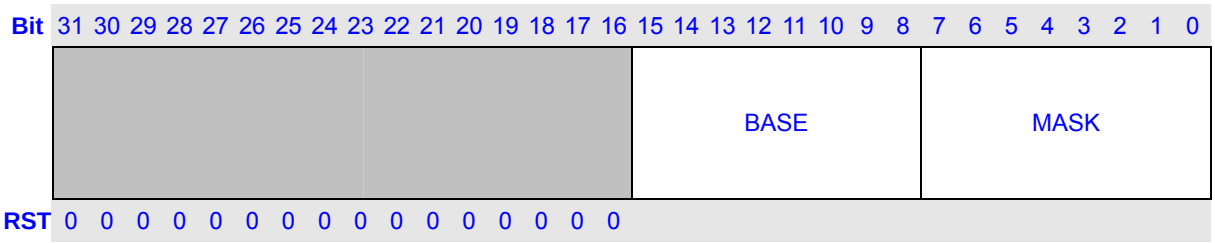
3.4.1.2 Static Bank Address Configuration Register (SACR1~4)

SACR1~4 defines the physical address for static memory bank 1 to 4, respectively. Each register contains a base address and a mask. When the following equation is met:

(*physical_address* [31:24] & **MASK$_n$**) == **BASE$_n$**

The bank n is active. The *physical_address* is address output on internal system bus. Static bank regions must be programmed so that each bank occupies a unique area of the physical address space. Bank 0 base address must be 0 because it's system boot address. Programming overlapping bank regions will result in unpredictable error. These registers are initialized by a reset.

SACR1 **0x13010034**
SACR2 **0x13010038**
SACR3 **0x1301003C0x13010040**
SACR4



Bits	Name	Description	RW
31:16	Reserved	Writes to these bits have no effect and read always as 0.	R
15:8	BASE	**Address Base:** Defines the base address of Static Bank n (n = 1 to 4). The initial values are: SACR1.BASE 0x18 SACR2.BASE 0x14 SACR3.BASE 0x0C SACR4.BASE 0x08	RW
23:20	MASK	**Address Mask**: Defines the mask of Static Bank n (n = 1 to 4). The initial values are: SACR1.MASK 0xFC SACR2.MASK 0xFC SACR3.MASK 0xFC SACR4.MASK 0xFC	RW

3.4.2 Example of Connection

Following figures shows examples of connection to 32-, 16- and 8-bit data width normal memory.

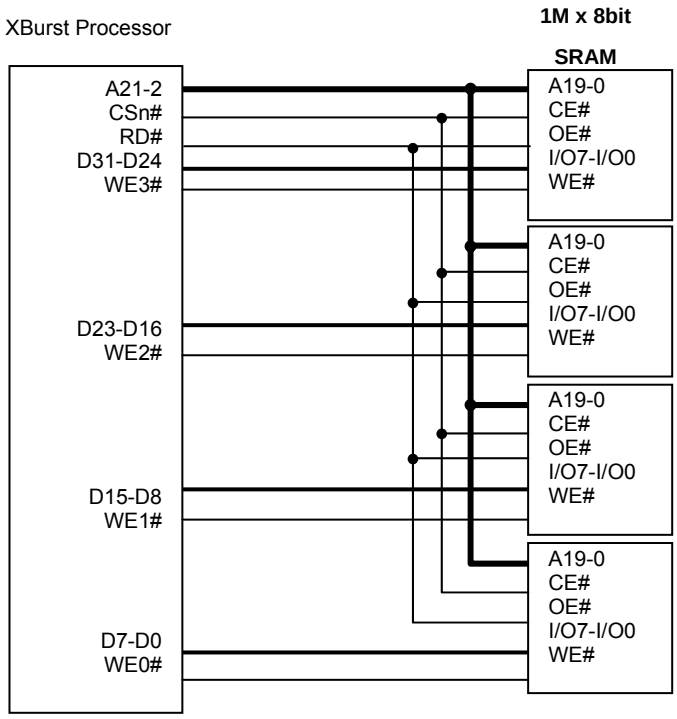


Figure 3-2 Example of 32-Bit Data Width SRAM Connection

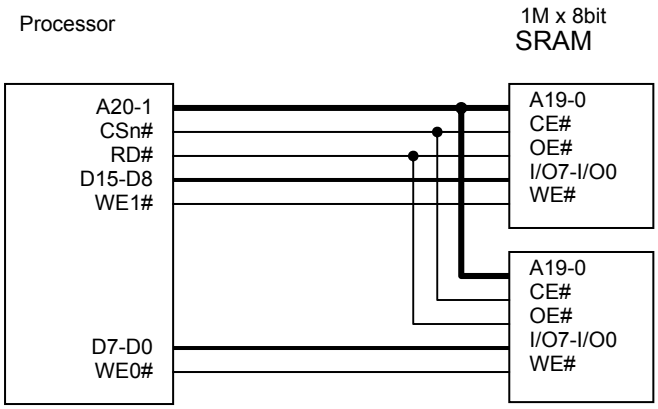


Figure 3-3 Example of 16-Bit Data Width SRAM Connection

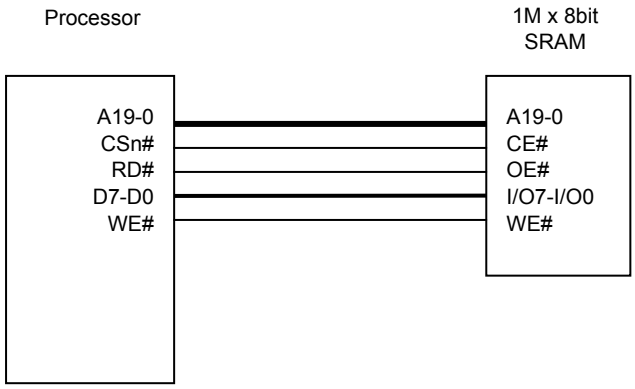


Figure 3-4 Example of 8-Bit Data Width SRAM Connection

3.4.3 Basic Interface

When SMT field in SMCRn (n = 1 to 4) is 0 and BCM field is 0, normal memory (non-burst ROM, Flash, normal SRAM or memory-like device) is connected to bank n. When bank n (n = 1 to 4) is accessed, CSn# is asserted as soon as address is output. In addition, the RD# signal, which can be used as OE#, and write control signals, WE0# to WE3#, are asserted.

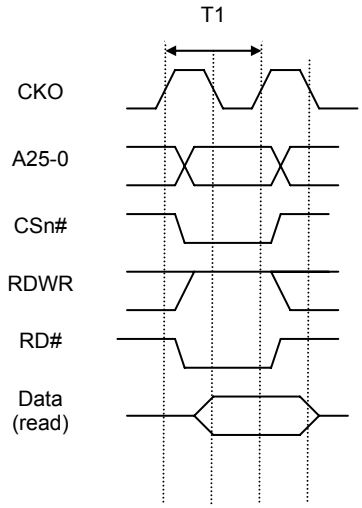
The TAS field in SMCRn is the latency from CSn# to read/write strobe. The TAW3 field is the delay time of RD# in read access. TBP3~0 field is the delay time of WE# and WEn# in write access. In addition, any number of waits can be inserted by means of the external pin (WAIT#). The TAH field is the latency from RD# and WEn# negation to CSn# negation, also the hold time to address and write data.

All kinds of normal memories (non-burst ROM, normal SRAM and Flash) have the same read and write timing. There are some requirements for writes to flash memory. Flash memory space must be un-cacheable and un-buffered. Writes must be exactly the width of the populated Flash devices on the data bus (no byte writes to a 32-bit bus or word writes to a 16-bit bus, and so on). Software is responsible for partitioning commands and data, and writing them out to Flash in the appropriate sequence.

Glossary

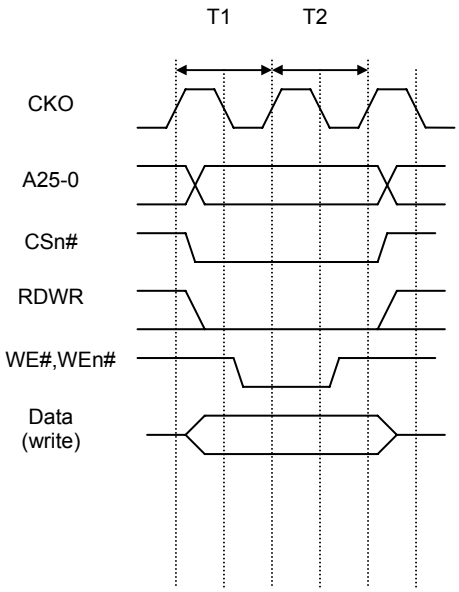
Th – hold cycle
Tw – wait cycle
Ts – setup cycle
T1 – read inherent cycle or first write inherent cycle
T2 – last write inherent cycle
Tb – burst read inherent cycle

Following figures show the timing of normal memory. A no-wait read access is completed in one cycle and a no-wait write access is completed in two cycles. Therefore, there is no negation period in case of access at minimum pitch.



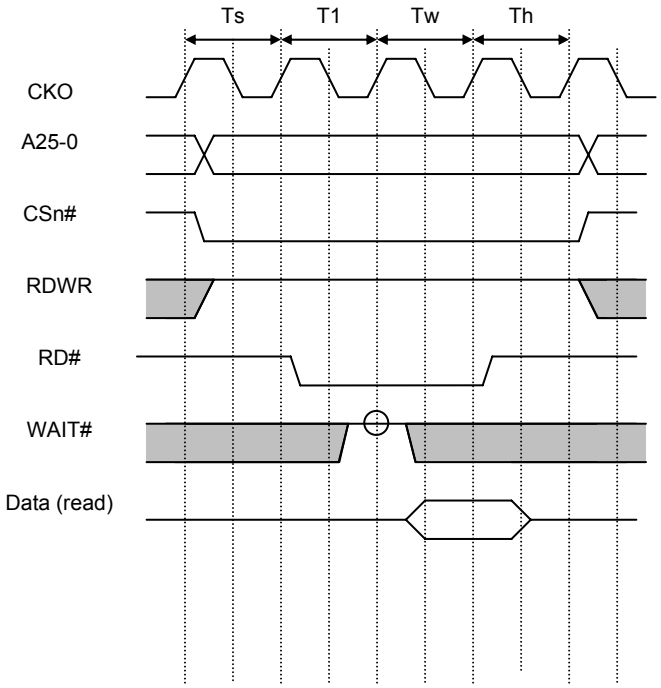
*In this example, SMCRn:MT = 0, BCM = 0,
TAS = 0, TAW = 0, TAH = 0

Figure 3-5 Basic Timing of Normal Memory Read



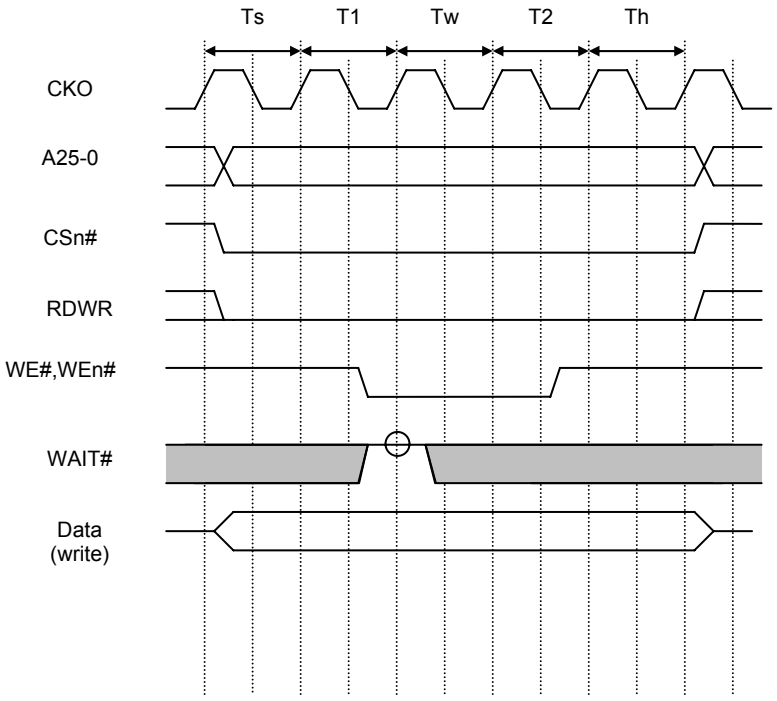
*In this example, SMCRn: SMT = 0, BCM = 0,
TAS = 0, TBP = 0, TAH = 0

Figure 3-6 Basic Timing of Normal Memory Write



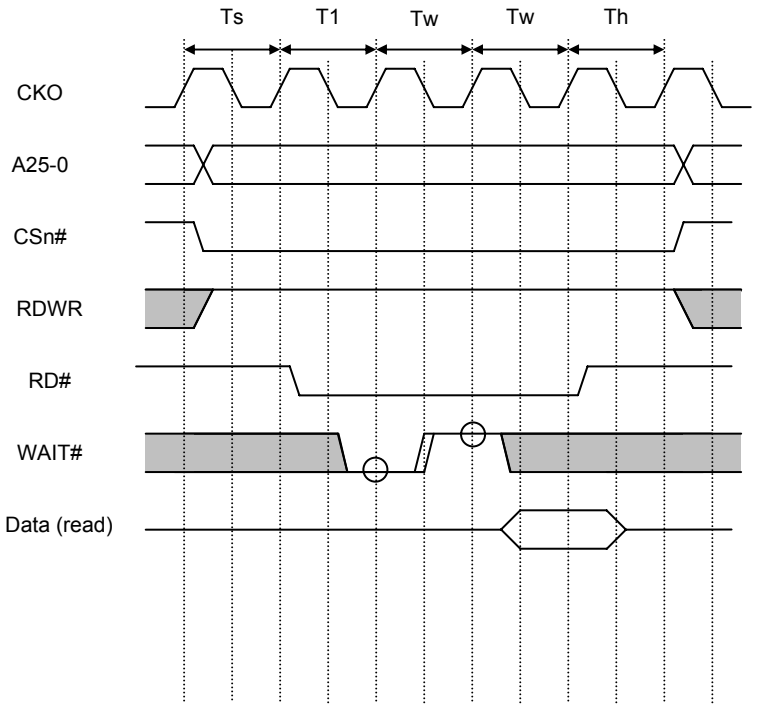
*In this example, SMCRn: SMT = 0, BCM = 0, TAS = 1, TAW = 1, TAH = 1

Figure 3-7 Normal Memory Read Timing With Wait (Software Wait Only)



*In this example, SMCRn: SMT = 0, BCM = 0, TAS = 1, TBP = 1, TAH = 1

Figure 3-8 Normal Memory Write Timing With Wait (Software Wait Only)



*In this example, SMCRn: SMT = 0, BCM = 0, TAS = 1, TAW = 1, TAH=1

Figure 3-9 Normal Memory Read Timing With Wait (Wait Cycle Insertion by WAIT# pin)

3.4.4 Byte Control

The byte control SRAM interface is a memory interface that outputs a byte select strobe WEn# in both read and write bus cycles. It has 16 bit data pins, and can be directly connected to SRAM which has an upper byte select strobe and lower byte select strobe function such as UB# and LB#.

In read/write access, RD#/WE# is used as read/write strobe signal and WEn# are used as byte select signals.

Following figure shows an example of byte control SRAM connection to the XBurst processor.

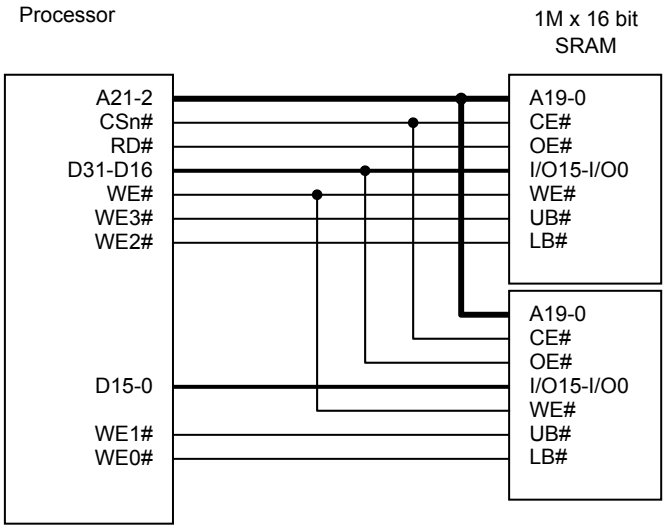
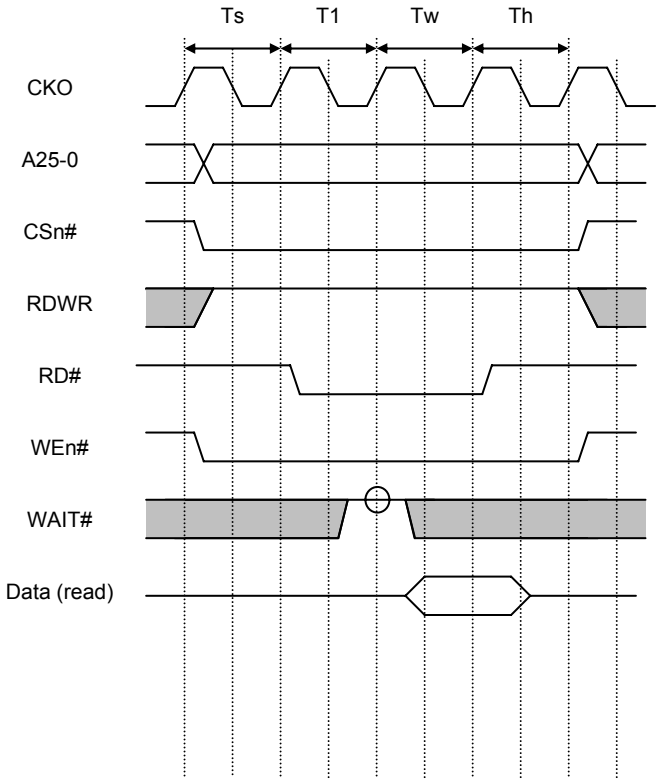


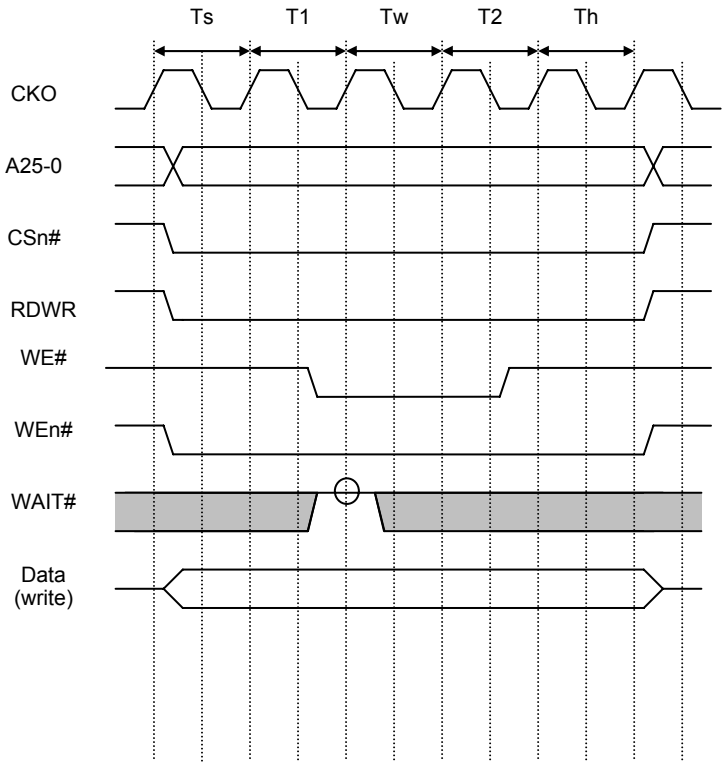
Figure 3-10 Example of 32-Bit Data Width Byte Control SRAM Connection

Following figures show examples of Byte Control SRAM timing.



*In this example, SMCRn: SMT = 0, BCM = 1, TAS = 1, TAW = 1, TAH = 1

Figure 3-11 Byte Control SRAM Read Timing



*In this example, SMCRn: SMT = 0, BCM = 1, TAS = 1, TBP = 1, TAH = 1

Figure 3-12 Byte Control SRAM Write Timing

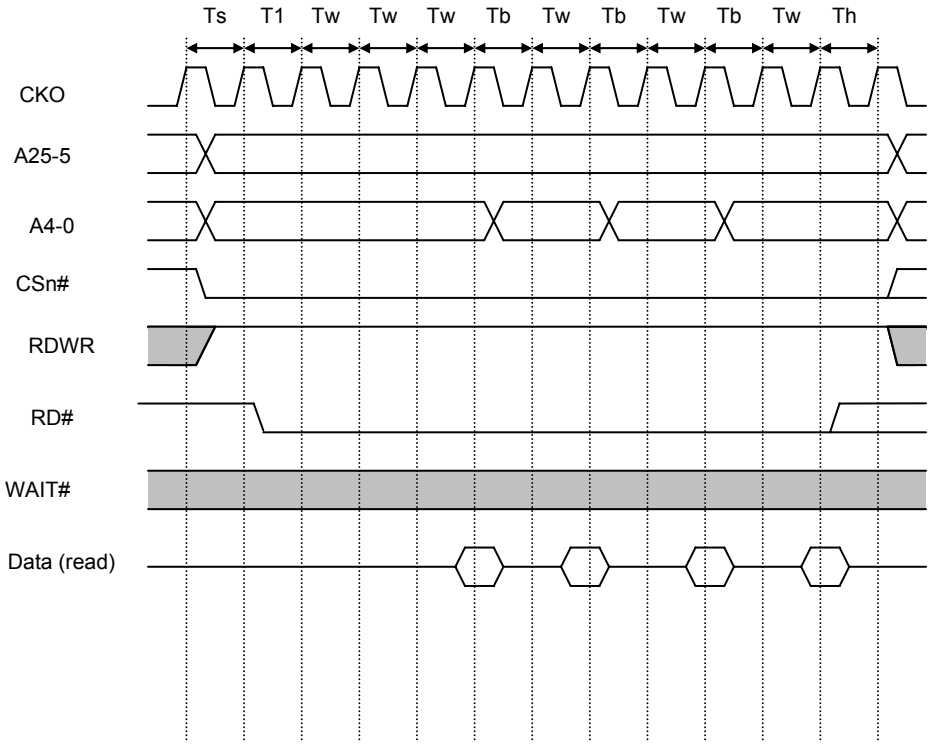
3.4.5 Burst ROM Interface

Setting SMT to 1 in SMCRn allows burst ROM to be connected to bank n (n = 1 to 4). The burst ROM interface provides high-speed access to ROM that has a nibble access function. Basically, access is performed in the same way as for normal memory, but when the first cycle ends, only the address is changed before the next access is executed. When 8-bit burst ROM is connected, the number of consecutive accesses can be set as 4, 8, 16, or 32 with bits BL1~0. When 16-bit ROM is connected, 4, 8, or 16 can be set in the same way. When 32-bit ROM is connected, 4 or 8 can be set.

For burst ROM read, TAW sets the delay time from read strobe to the first data, TBP sets the delay time from consecutive address to data. Burst ROM writes have the same timing as normal memory except TAW instead of TBP is used to set the delay time of write strobe.

WAIT# pin sampling is always performed when one or more wait states are set.

Following figures show the timing of burst ROM.



*In this example, SMT = 1, BL = 0, TAS = 1, TAW = 3,TBP = 1,TAH = 1

Figure 3-13 Burst ROM Read Timing (Software Wait Only)

3.5 NAND Flash Interface

NAND flash can be connected to static memory bank 4~ band 1. Both 8-bit and 16-bit NAND flashes are supported. A mechanism for booting from NAND flash is also supported.

3.5.1 Register Description

Table 3-5 NAND Flash Interface Registers

Name	Description	RW	Reset Value	Address	Access Width
NFCSR	NAND flash control/status register	RW	0x00000000	0x13010050	32

3.5.1.1 NAND Flash Control/Status Register (NFCSR)

NFCSR is a 32-bit read/write register that is used to configure NAND flash. It is initialized by any reset.

NFCSR **0x13010050**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																									NFCE4	NFE4	NFCE3	NFE3	NFCE2	NFE2	NFCE1	NFE1
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:16	Reserved	Writes to these bits have no effect and read always as 0.	R
1/3/5/7	FCEn (n=1,2,3,4)	**NAND Flash FCE# Assertion Control :** Controls the assertion of NAND Flash FCEn#. When set, FCEn# is always asserted until this bit is cleared. When the NAND flash require FCEn# to be asserted during read busy time, this bit should be set. **FCE** **Description** 0 FCEn# is asserted as normal static chip enable(Initial value) 1 FCEn# is always asserted	RW
0/2/4/6	NFEn (n=1,2,3,4)	**NAND Flash Enable:** Specifies if NAND flash is connected to static bank n. When system is configured to boot from NAND flash, this bit is initialized to 1. **NFE** **Description** 0 Static bank n is not used as NAND flash 1 Static bank n is used as NAND flash	RW

3.5.2 NAND Flash Boot Loader

To support boot from NAND flash, 8KB on-chip Boot ROM is implemented. Following figure illustrates the structure of NAND Flash Boot Loader.

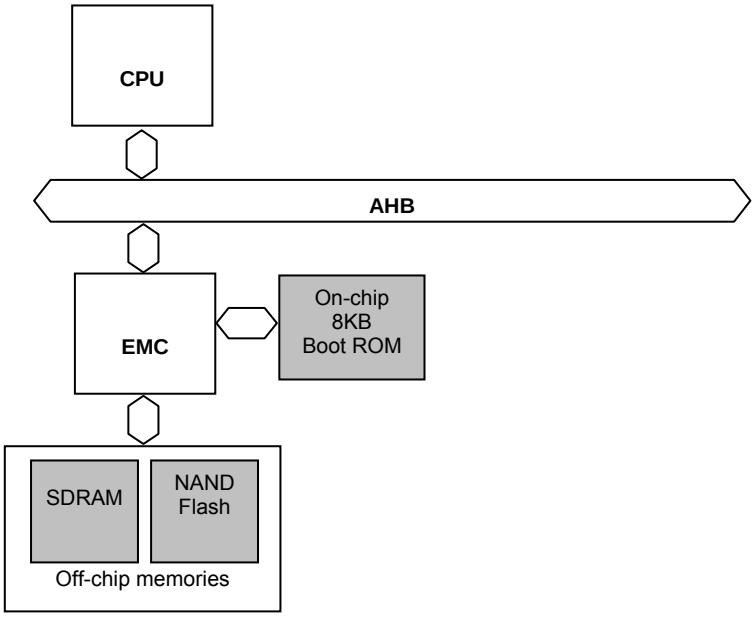


Figure 3-14 Structure of NAND Flash Boot Loader

When system is configured to boot from NAND flash, after reset, the program in Boot ROM is executed and the program will copy the first 8K bytes of NAND flash to CACHE for further initialization.

Generally, the boot code will copy more NAND flash content to SDRAM. Then the main program will be executed on SDRAM.

3.5.3 NAND Flash Operation

Set NFEn bit of NAND Flash Control/Status Register (NFCSR) will enable access to NAND flash. The partition of static bank n (n=1~4) is changed as following figure. Writes to any of address space will be translated to NAND flash address cycle. Writes to any of command space will be translated to NAND flash command cycle. Caution: don't read to address and command space, and these two partitions should be uncacheable. Reads and writes to any of data space will be translated to NAND flash data read/write cycle. DMA access to data space is supported to increase the speed of data read/write. The DMA access cannot exceed the page boundary (512 bytes or 2K bytes) of NAND flash.

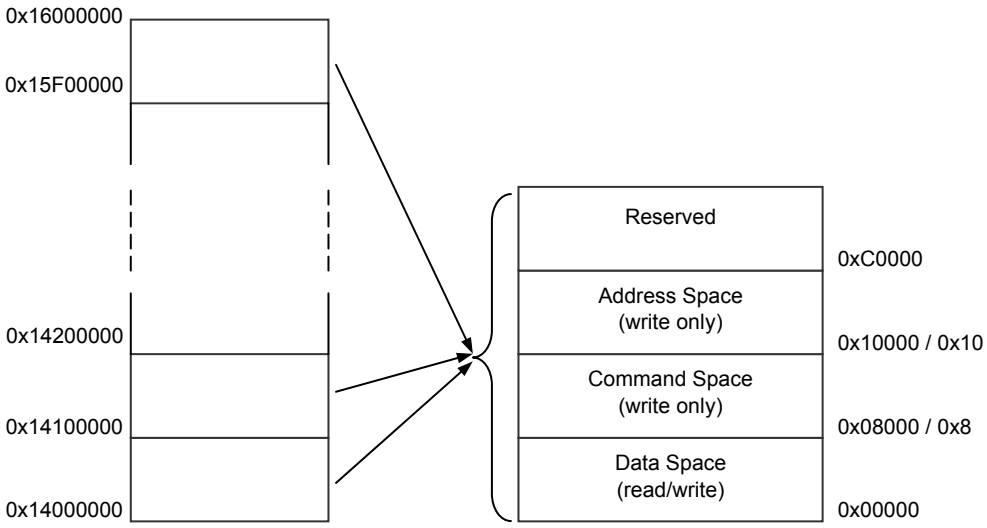


Figure 3-15 Static Bank 2 Partition When NAND Flash is Used (an example)

The timing of NAND flash access is configured by SMCRn and is same as normal static memory timing, except that CSn# is controlled by NFCE bit NFCSR. CSn# is always asserted when NFCE is 1. When NFCE is 0, CSn# is asserted as normal static memory access.

The control signals for direction connection of NAND flash are CSn#, FRE#, FWE#, FRB#(GPIO), A16 and A15. Following figure shows the connection between processor and NAND Flash.

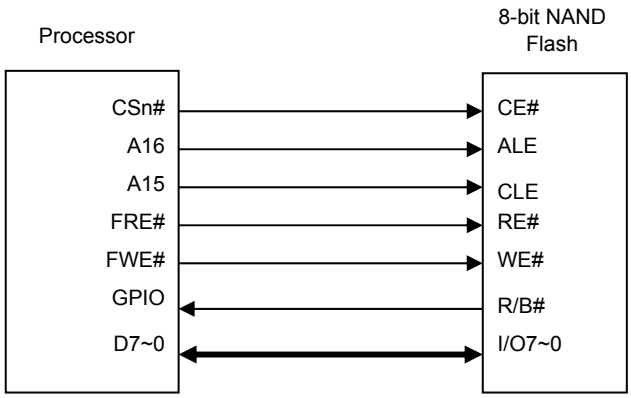


Figure 3-16 Example of 8-bit NAND Flash Connection

NOTES:

1 When BCR.BSR is 0, A16 is connected to ALE, A15 is connected to CLE, software should write 0x10000 for address space and 0x8000 for command space.

2 When BCR.BSR is 1, A4 is connected to ALE, A3 is connected to CLE, software should write 0x10 for address space and 0x8 for command space.

3.6 SDRAM Interface

The SDRAM controller provides a glueless interface to industry standard SDRAM chip. The SDRAM controller provides two chip selects DCS0~1# supporting 16-bit or 32-bit wide SDRAM.

Both 2-bank and 4-bank SDRAM modules are supported. The bank select signals are always output from the A13 pin and A14 pin of processor.

The SDRAM interface includes the following signals:

- Two chip selects, DCS0#, DCS1#
- Four byte mask signals, DQM3~0#
- 15 multiplexed bank/row/column address signals, A14-A0
- One write enable, RD/WR#
- One column-address strobe CAS#
- One row-address strobe RAS#
- One clock enable CKE
- One clock CKO

The processor performs auto-refresh (CBR) during normal operation and supports self-refreshing SDRAM during sleep, hibernate, and frequency-change modes. An SDRAM power-down mode bit (DMCR[PDM]]) can be set so that the CKO and the clock-enable signal CKE to SDRAM are automatically deasserted whenever none of the corresponding banks is being accessed.

3.6.1 Register Description

Table 3-6 SDRAM Registers

Name	Description	RW	Reset Value	Address	Access Width
DMCR	DRAM control register	RW	0x0000 0000	0x13010080	32
RTCSR	Refresh time control/status register	RW	0x0000	0x13010084	16
RTCNT	Refresh timer counter	RW	0x0000	0x13010088	16
RTCOR	Refresh time constant register	RW	0x0000	0x1301008C	16
DMAR1	SDRAM bank 0 address configuration register	RW	0x000020F8	0x13010090	32
DMAR2	SDRAM bank 1 address configuration register	RW	0x000028F8	0x13010094	32
SDMR	Mode register of SDRAM bank	W	--	0x1301-xxx (-: 4'b1xxx)	8

3.6.1.1 SDRAM Control Register (DMCR)

DMCR is a 32-bit read/write register that specifies the timing, address multiplexing and refresh control of SDRAM. This enables direct connection of SDRAM without external circuits.

The DMCR is initialized to 0x00000000 by any resets. SDRAM bank should not be accessed until initialization is completed.

DMCR **0x13010080**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	BW	Reserved		CA			RMODE	RFSH	Reserved		RA		BA	PDM	EPIN	MBSEL	TRAS			RCD		TPC			Reserved	TRWL		TRC			TCL	
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31	BW	Specifies the data bus width of SDRAM. **BW** **Description** 0 Data width is 32 bits (Initial value) 1 Data width is 16 bits	RW
30:29	Reserved	Writes to these bits have no effect and always read as 0.	R
28:26	CA	**Column Address Width**: Specify the column address width of connected SDRAM chip. **CA** **Description** 000 8 bits column address	RW

		001 9 bits column address 010 10 bits column address 011 11 bits column address 100 12 bits column address 101 Reserved 110 Reserved 111 Reserved	
25	RMODE	**Refresh Mode.** **RMODE Description** 0 Auto-refresh 1 Self-refresh	RW
24	RFSH	**Refresh Control.** **RFSH Description** 0 No refresh is performed (Initial value) 1 Refresh is performed	RW
23	MRSET	**Mode Register Set:** Set when a SDRAM mode register setting is used. When this bit is 0 and SDRAM mode register is written, a Pre-charge all banks command (PALL) is performed. When this bit is 1 and SDRAM mode register is written, a Mode Register Set command (MRS) is performed. **MRSET Description** 0 All-bank pre-charge (Initial value) 1 Mode register setting	RW
22	Reserved	Writes to these bits have no effect and always read as 0.	R
21:20	RA	**Row Address Width:** Specify the row address width of connected SDRAM. **RA Description** 00 11-bit row address (Initial value) 01 12-bit row address 10 13-bit row address 11 Reserved	RW
19	BA	**Bank Address Width:** Specify the number of bank select signals for one chip select. **BA Description** **0** 1-bit bank address is used (2 banks each chip select) (Initial value) **1** 2-bit bank address is used (4 banks each chip select)	RW
18	PDM	**Power Down Mode:** Set power-down mode. When power-down mode is set, SDRAM will be driven to power-down mode when it is not accessing and refreshing. Clock supply to SDRAM will be stopped also. **PDM Description** 0 Non-power-down mode (Initial value) 1 Power-down mode	RW

17	EPIN	**CKE Pin Control:** Controls the level of CKE pin. Clearing this bit by software causes a power-down command (if CKOEN of CPM is 1). Caution: after power-down command, all commands except power-down-exit are prohibited. Setting this bit by software causes a power-down-exit command. Setting EPIN is a part of initializes procedure for SDRAM. **EPIN Description** 0 CKE pin is deserted (Initial value) 1 CKE pin is asserted	RW
16	MBSEL	**Bank Select for Mode Register Load:** It is used to distinguish to load which bank Mode register. **MBSEL Description** 0 Bank 0 (Initial value) 1 Bank 1	RW
15:13	TRAS	**RAS Assertion Time:** When synchronous DRAM is connected, these bits set the minimum CKE negation time after self-refresh command is issued. **TRAS Description** 000 4 (Initial value) 001 5 010 6 011 7 100 8 101 9 110 10 111 11	RW
12:11	RCD	**RAS–CAS Delay**: Set the SDRAM bank active-read/write command delay time. **RCD Description** 00 1(Initial value) 01 2 10 3 11 4	RW
10:8	TPC	**RAS Precharge Time:** Specify the minimum number of cycles until the next bank active command is output after precharging. TPC Description 000 1 cycle (Initial value) 001 2 cycles 010 3 cycles 011 4 cycles 100 5 cycles 101 6 cycles 110 7 cycles	RW

		111 8 cycles	
7	Reserved	Writes to these bits have no effect and always read as 0.	R
6:5	TRWL	**Write Precharge Time:** Set the SDRAM write precharge delay time. In auto-precharge mode, they specify the time until the next bank active command is issued after a write cycle. After a write cycle, the next active command is not issued for a period of TRWL + TPC. **TRWL** **Description** 00 1 cycle (Initial value) 01 2 cycles 10 3 cycles 11 4 cycles	RW
4:2	TRC	**RAS Cycle Time:** For SDRAM, no bank active command is issued during the period TRC after an auto-refresh command. In self-refresh, these bits also specify the delay cycles to be inserted after CKE assertion. **TRC** **Description** 000 1 cycle (Initial value) 001 3 cycle 010 5 cycle 011 7 cycle 100 9 cycle 101 11 cycle 110 13 cycle 111 15 cycle	RW
1:0	TCL	**CAS Latency:** Specify the delay from read command to data becomes available at the outputs. **TCL** **Description** 00 Inhibit (Initial value) 01 2 cycles 10 3 cycles 11 Inhibit	RW

3.6.1.2 SDRAM Mode Register (SDMR)

SDMR is written to via the SDRAM address bus and is a 15-bit write-only register. It sets SDRAM mode for SDRAM bank. SDMR is undefined after a reset.

Write to the SDRAM mode register use the address bus rather than the data bus. If the value to be set is X and the SDMR address is Y, the value X is written in the SDRAM mode register by writing in address X + Y. Here Y is 0x8000, X is value for SDRAM configuration. For example X is 0x0022, random data is written to the address offset 0x8022, as a result, 0x0022 is written to the SDMR register. The range for value X is 0x0000 to 0x7FFF.

SDMR **0x1301-000**

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

The Mode Register is used to define the specific mode of operation of the SDRAM. This definition includes the section of a burst length, a burst type, a CAS latency, an operating mode and a write burst mode, as shown in following figure.

For Mobile SDR, Extended Mode Register is used to define low power mode.

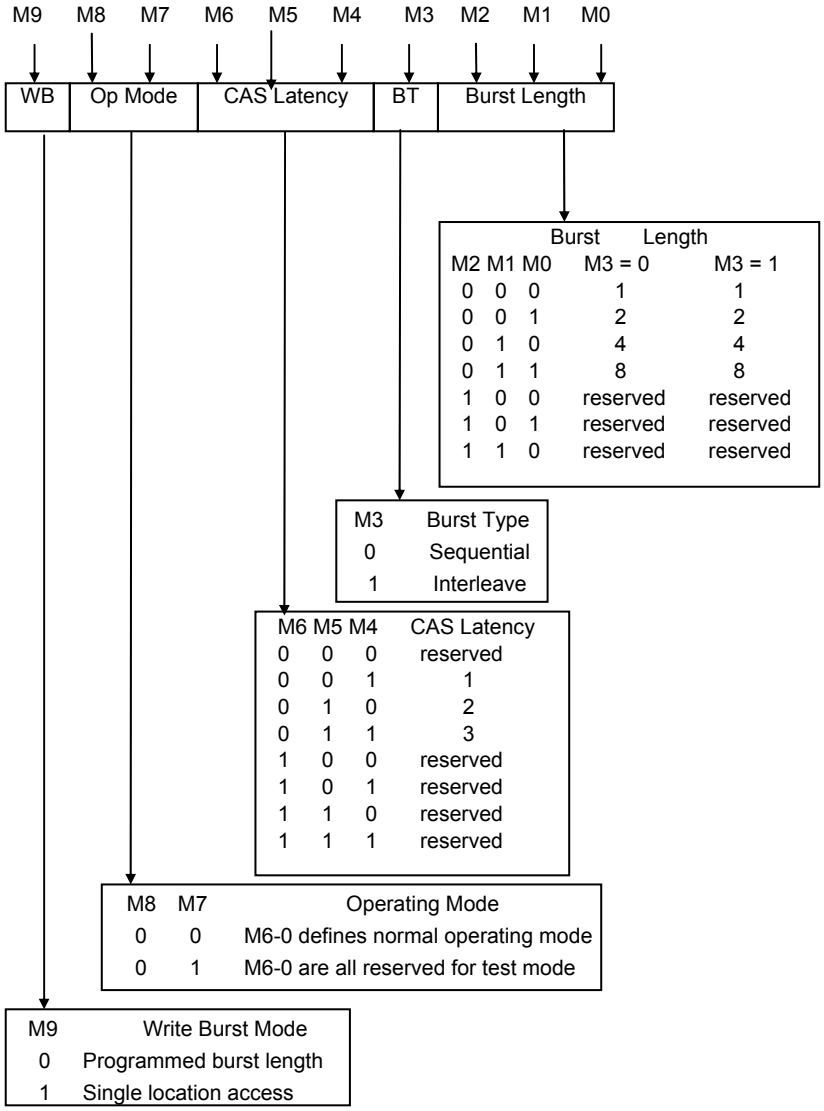


Figure 3-17 Synchronous DRAM Mode Register Configuration

3.6.1.3 Refresh Timer Control/Status Register (RTCSR)

RTCSR is a 16-bit readable/writable register that specifies the refresh cycle and the status of RTCNT.

RTCSR is initialized to 0x0000 by a reset.

RTSCR 0x13010084

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved							SRF	CMF	Reserved				CKS		
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15:9	Reserved	These bits always read 0. Data written to these bits are ignored.	R
8	SRF	**Self-refresh Flag (SRF):** Status flag that indicates EMC already enter self-refresh sequence. **SRF** **Description** 0 No self-refresh (Initial value) Clear condition: When 0 is written, write 1 is ignored 1 EMC already enter self-refresh sequence Set condition: when EMC enter self-refresh	RW
7	CMF	**Compare-Match Flag (CMF):** Status flag that indicates a match between the refresh timer counter (RTCNT) and refresh time constant register (RTCOR) values. Writes to 1 of this bit have no effect. **CMF** **Description** 0 RTCNT and RTCOR values do not match (Initial value) Clear condition: When 0 is written 1 RTCNT and RTCOR values match Set condition: When RTCNT = RTCOR	RW
2:0	CKS	**Refresh Clock Select Bits:** These bits select the clock input to RTCNT. The source clock is the external bus clock (CKO). The RTCNT count clock is CKO divided by the specified ratio. **CKS** **Description** 000 Disable clock input (Initial value) 001 Bus lock CKO/4 010 CKO/16 011 CKO/64 100 CKO/256 101 CKO/1024 110 CKO/2048 111 CKO/4096	RW

3.6.1.4 Refresh Timer Counter (RTCNT)

RTCNT is a 16-bit read/write register. RTCNT is a 16-bit counter that counts up with input clocks. The clock select bits (CKS2–CKS0) of RTCSR select the input clock. When the refresh bit (RFSH) of the memory control register (DMCR) is set to 1 and the refresh mode is set to auto-refresh, a memory refresh cycle starts when RTCNT matches RTCOR. RTCNT is initialized to 0x0000 by a reset.

RTCNT **0x13010088**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

3.6.2 Refresh Time Constant Register (RTCOR)

RTCOR is a 16-bit read/write register. The values of RTCOR and RTCNT (bottom 8 bits) are constantly compared. When the refresh bit (RFSH) of the memory control register (DMCR) is set to 1 and the refresh mode bit (RMODE) is set to auto-refresh, a memory refresh cycle starts when RTCNT matches RTCOR. RTCOR is initialized to 0x0000 by a reset.

RTCOR **0x1301008C**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

3.6.2.1 DRAM Bank Address Configuration Register (DMARn, n = 1, 2)

DMARn define the physical address for SDRAM bank0 or bank 1, respectively. Each register contains a base address and a mask. When the following equation is met:

(*physical_address* [31:24] & **MASK$_n$**) == **BASE$_n$**

The bank n is active. The *physical_address* is address output on internal system bus. DRAM bank regions must be programmed so that each bank occupies a unique area of the physical address space. Programming overlapping bank regions will result in unpredictable error.

These registers are initialized by a reset.

DMAR1 **0x13010090**

DMAR2 **0x13010094**

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

Bits	Name	Description	RW
31:16	Reserved	Writes to these bits have no effect and read always as 0.	R
15:8	BASEn	**Address Base:** Defines the base address of SDRAM Bank. The initial values are: DMAR.BASE1 0x20 DMAR.BASE2 0x28	RW
23:20	MASKn	**Address Mask**: Defines the mask of SDRAM Bank. The initial values are: DMAR.MASK 0xF8	RW

3.6.3 Example of Connection

Following figure shows an example of connection of 512K x 16-bit x 2-bank SDRAM.

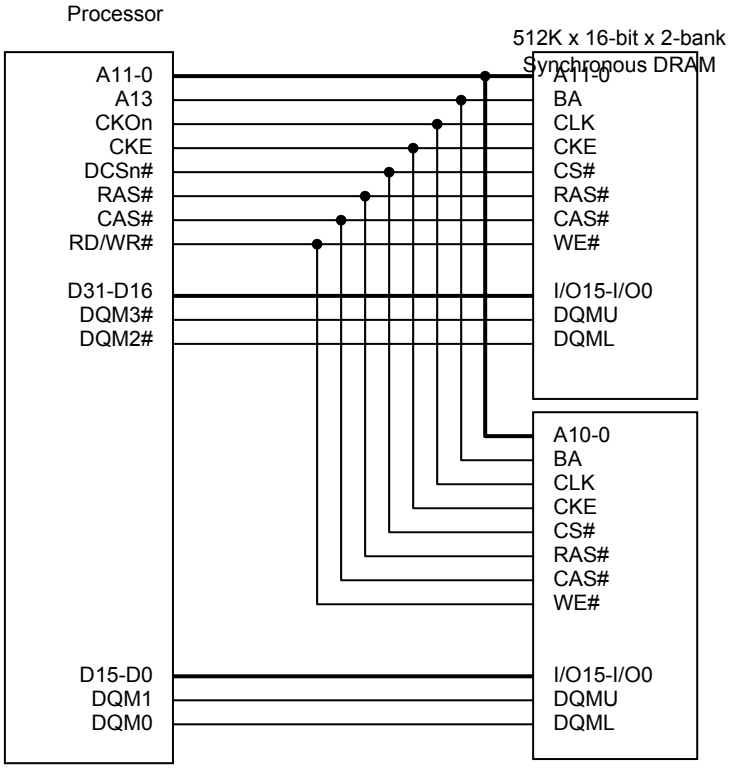


Figure 3-18 Example of Synchronous DRAM Chip Connection (1)

Following figure shows an example of connection of 1M x 16-bit x 4-bank synchronous DRAM.

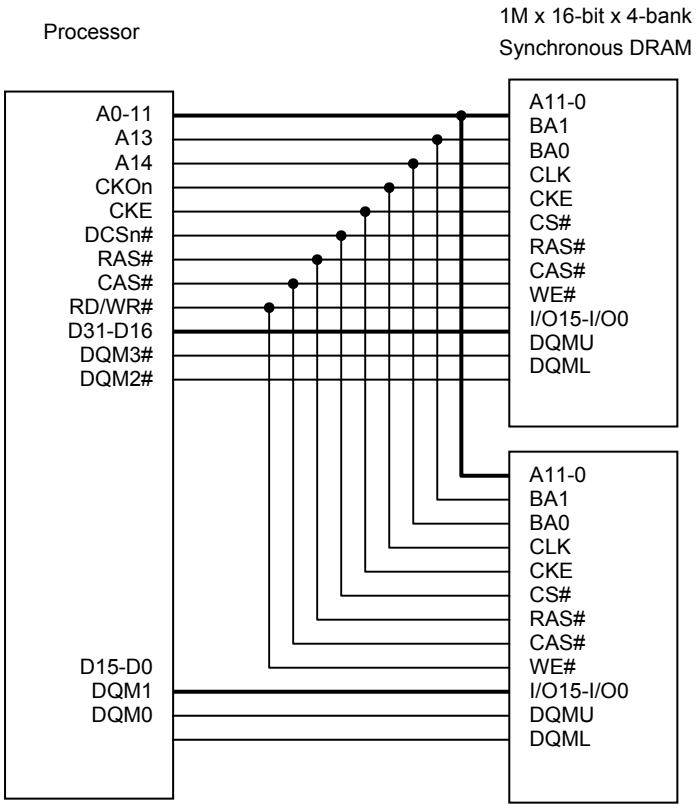


Figure 3-19 Example of Synchronous DRAM Chip Connection (2)

3.6.4 Address Multiplexing

SDRAM can be connected without external multiplexing circuitry in accordance the address multiplex specification bits CA2~0, RA1~0 and BA in DMCR. Table 3-7 shows the relationship between the address multiplex specification bits and the bits output at the address pins.

A14-0 is used as SDRAM address. The original values are always output at these pins.

Table 3-7 SDRAM Address Multiplexing (32-bit data width) *[4]

CA2~0	RA1~0	Output Timing	A0-A9, A10, A11, A12	A13	A14	Note
8 bits	11 bits	Column	A2-A11, L/H*[1], A12, A13	A21	A22	3, 4
		Row	A10-A22			
	12 bits	Column	A2-A11, L/H*[1], A12, A13	A22	A23	3, 4
		Row	A10-A22			
	13 bits	Column	A2-A11, L/H*[1], A12, A13	A23	A24	3, 4
		Row	A10-A22			
9 bits	11 bits	Column	A2-A11, L/H*[1], A12, A13	A22	A23	3, 4
		Row	A11-A23			
	12 bits	Column	A2-A11, L/H*[1], A12, A13	A23	A24	3, 4
		Row	A11-A23			
	13 bits	Column	A2-A11, L/H*[1], A12, A13	A24	A25	3, 4
		Row	A11-A23			
10 bits	11 bits	Column	A2-A11, L/H*[1], A12, A13	A23	A24	3, 4
		Row	A12-A24			
	12 bits	Column	A2-A11, L/H*[1], A12, A13	A24	A25	3, 4
		Row	A12-A24			
	13 bits	Column	A2-A11, L/H*[1], A12, A13	A25	A26	3, 4
		Row	A12-A24			
11 bits	11 bits	Column	A2-A11, L/H*[1], A12, A13	A24	A25	3, 4
		Row	A13-A25,			
	12 bits	Column	A2-A11, L/H*[1], A12, A13	A25	A26	3, 4
		Row	A13-A25,			
	13 bits	Column	A2-A11, L/H*[1], A12-A17	A26	A27	3, 4
		Row	A13-A25,			
12 bits	11 bits	Column	A2-A11, L/H*[1], A12, A13	A25	A26	3, 4
		Row	A14-A26			
	12 bits	Column	A2-A11, L/H*[1], A12, A13	A26	A27	3, 4
		Row	A14-A26			
	13 bits	Column	A2-A11, L/H*[1], A12, A13	A27	A28	3, 4
		Row	A14-A26			

NOTES:

1 L/H is a bit used in the command specification; it is fixed at L or H according to the Access mode.

2 Bank address specification.

3 If one bank select signal is used (BA = 0), take A13 as bank select signal. If two bank select signals are used (BA = 1), take A13 and A14 as bank select signals.

4 The A0 to A14 in table head are output pins. The A2 to A28 in table body are physical address.

3.6.5 SDRAM Command

Commands for SDRAM are specified by RAS#, CAS#, RD/WR and special address signals. The processor accesses SDRAM by using the following subset of standard interface commands.

- Mode Register Set (MRS)
- Bank Activate (ACTV)
- Read (READ)
- Write (WRIT)
- Burst Terminate
- Precharge All Banks (PALL)
- Auto-Refresh (CBR)
- Enter Self-Refresh (SLFRSH)
- No Operation (NOP)

Table 3-8 SDRAM Command Encoding (NOTES:1)

Command	Processor Pins							
	CS#	RAS#	CAS#	RD/WR#	DQM	A14-11, A9-0	A10	Note
INHIBIT	H	X	X	X	X	X	X	
NOP	L	H	H	H	X	X	X	
MRS	L	L	L	L	X	Op-Code		
ACTV	L	L	H	H	X	Bank, Row	X	2
READ	L	H	L	H	L/H	Bank, Col	L	3
WRIT	L	H	L	L	L/H	Bank, Col	L	3
Burst Terminate	L	H	H	L	X	X	X	
PRE	L	L	H	L	X	Bank	L	
PALL	L	L	H	L	X	X	H	
CBR/SLFRSH	L	L	L	H	X	X	X	4

NOTES:

1 CKE is HIGH for all commands shown except SLFRSH.

2 A0-A12 provides row address, and A13-A14 determines which bank is active.

3 A0-A9 provides column address, and A13-A14 determines which bank is being read from or written to.

4 This command is CBR if CKE is HIGH, SLFRSH if CKE is LOW.

3.6.6 SDRAM Timing

The SDRAM bank function is used to support high-speed accesses to the same row address. As SDRAM is internally divided into two or four banks, it is possible to activate one row address in each bank.

When a de-active bank is accessed, an access is performed by issuing an ACTV command following by READ or WRIT command.

When an active bank is accessed and just hit the open row, an access is performed by issuing READ or WRIT command immediately without issuing an ACTV command.

When an active bank is accessed but hit a closed row, a PRE command is first issued to precharge the bank, then the access is performed by issuing an ACTV command followed by a READ or WRIT command.

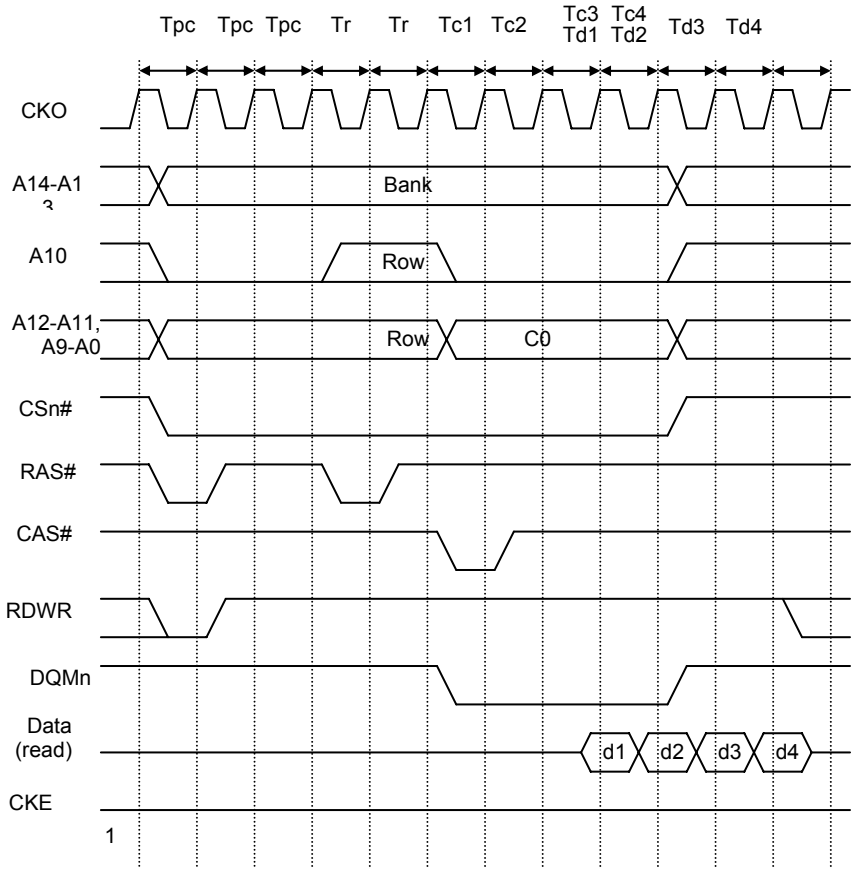
There is a limit on Tras, the time for placing each bank in the active state. If there is no guarantee that there will not be a cache hit and another row address will be accessed within the period in which this value is maintained by program execution, it is necessary to set auto-refresh and set the refresh cycle to no more than the maximum value of Tras. In this way, it is possible to observe the restrictions on the maximum active state time for each bank. If auto-refresh is not used, measures must be taken in the program to ensure that the banks do not remain active for longer than the prescribed time.

Glossary

Tr – row active cycle
Trw – row active wait cycle
Trwl – write latency cycle
Tpc – precharge cycle
TRr – refresh command cycle
Trc – RAS cycle
Trs1 – self refresh cycle 1
Trs2 – self refresh cycle 2
Trs3 – self refresh cycle 3
Trsw – self refresh wait cycle
Tc1 – command cycle 1
Tc2 – command cycle 2
Tc3 – command cycle 3
Tc4 – command cycle 4
Tc5 – command cycle 5
Tc6 – command cycle 6
Tc7 – command cycle 7
Tc8 – command cycle 8
Td1 – data cycle 1
Td2 – data cycle 2

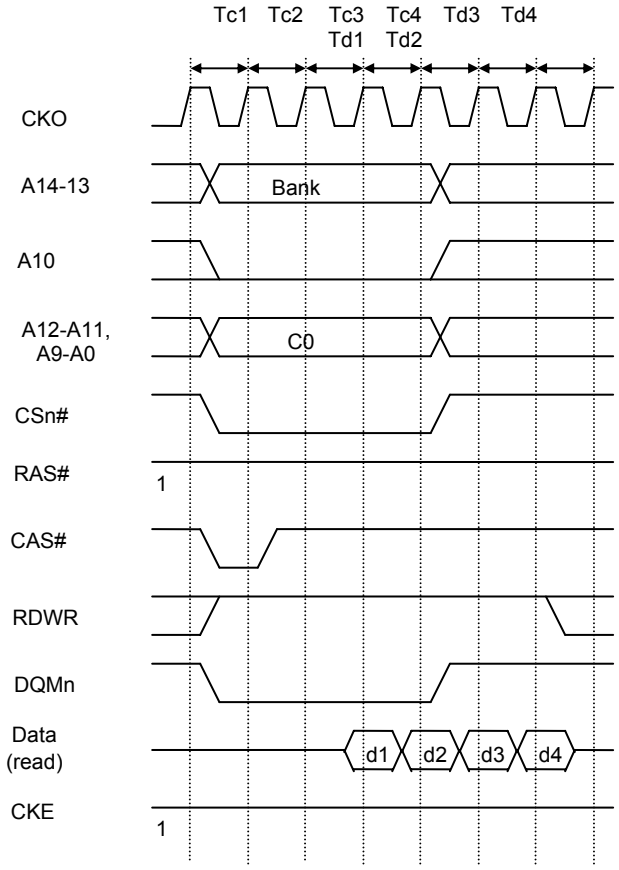
Td3 – data cycle 3
Td4 – data cycle 4
Td5 – data cycle 5
Td6 – data cycle 6
Td7 – data cycle 7
Td8 – data cycle 8
TRp1 – precharge-all cycle 1
TRp2 – precharge-all cycle 2
TRp3 – precharge-all cycle 3
TRp4 – precharge-all cycle 4
TMw1 – mode register set cycle 1
TMw2 – mode register set cycle 2
TMw3 – mode register set cycle 3
TMw4 – mode register set cycle 4

Following figures show the timing of 4-beat burst access, 8-beat burst access and single access.



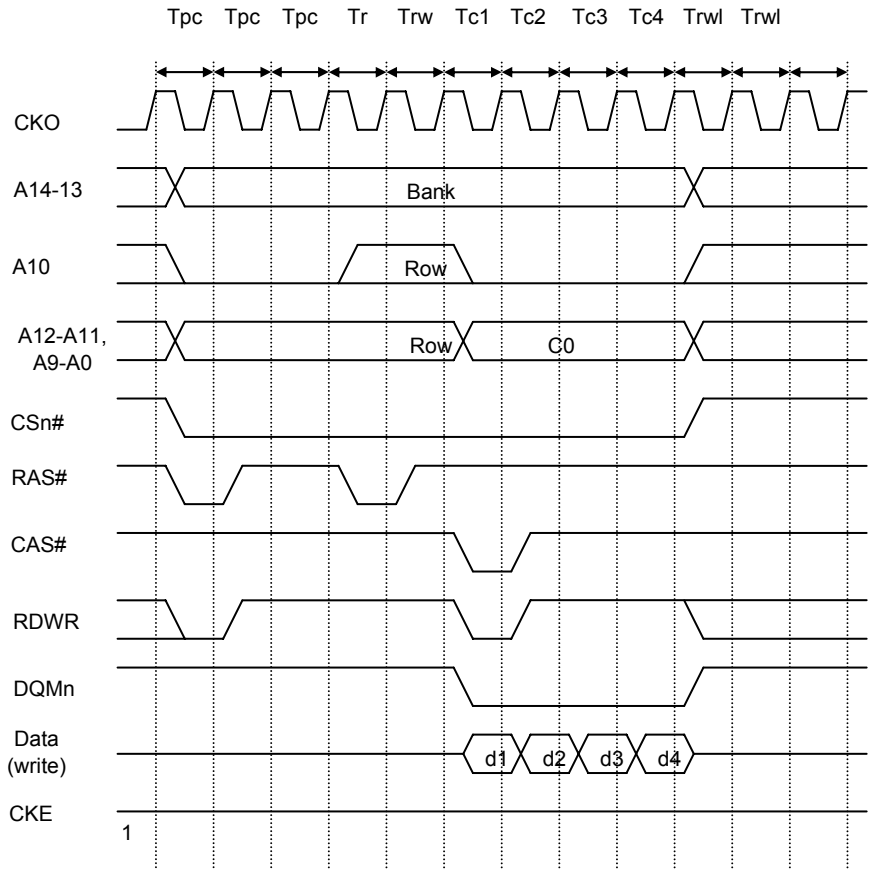
*DMCR: RCD = 1, TCL = 1, TPC = 2

Figure 3-20 Synchronous DRAM 4-beat Burst Read Timing (Different Row)



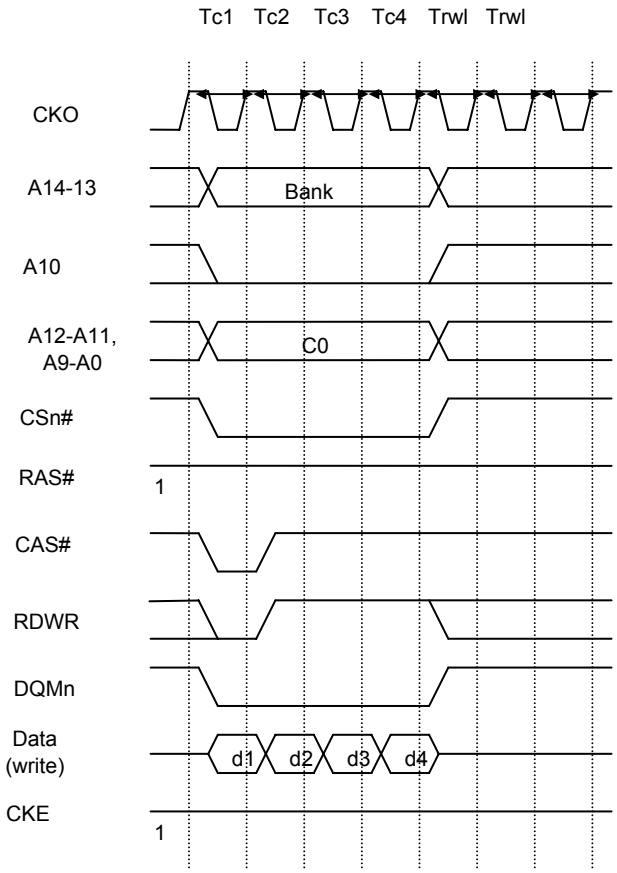
*DMCR: RCD = 1, TCL = 1, TPC = 2

Figure 3-21 Synchronous DRAM 4-beat Burst Read Timing (Same Row)



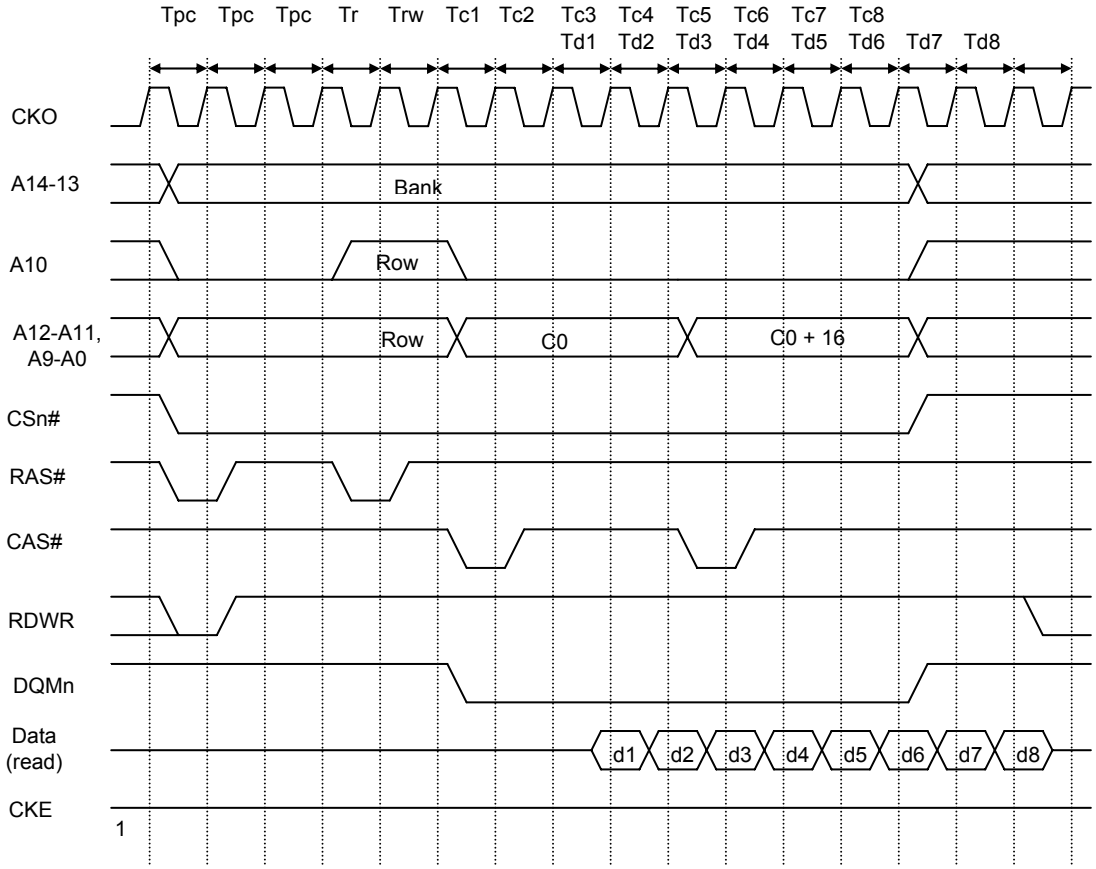
*DMCR: RCD = 1, TCL = 1, TPC = 2, TRWL = 1

Figure 3-22 Synchronous DRAM 4-beat Burst Write Timing (Different Row)



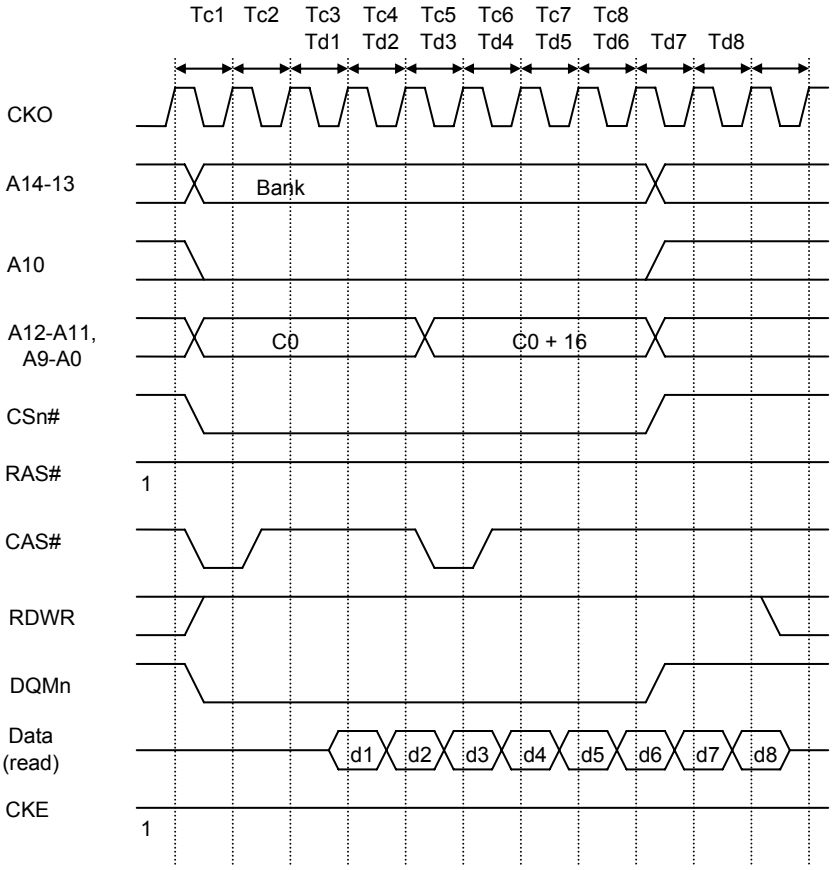
*DMCR: RCD = 1, TCL = 1, TPC = 2, TRWL = 1

Figure 3-23 Synchronous DRAM 4-beat Burst Write Timing (Same Row)



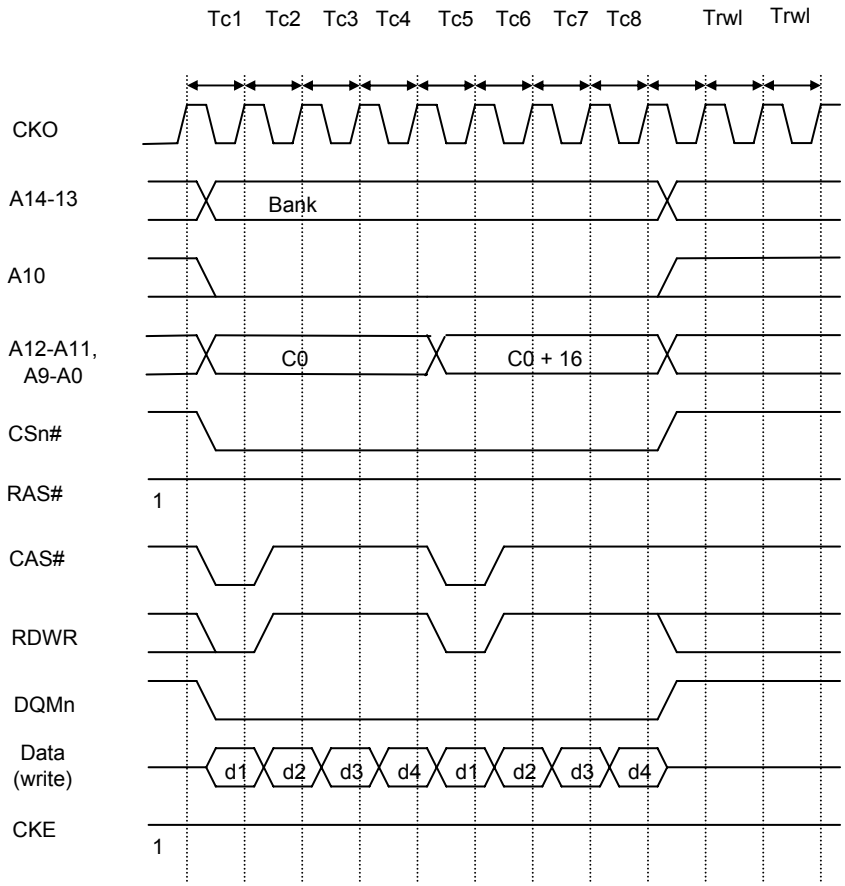
*DMCR: RCD = 1, TCL = 1, TPC = 2

Figure 3-24 Synchronous DRAM 8-beat Burst Read Timing (Different Row)



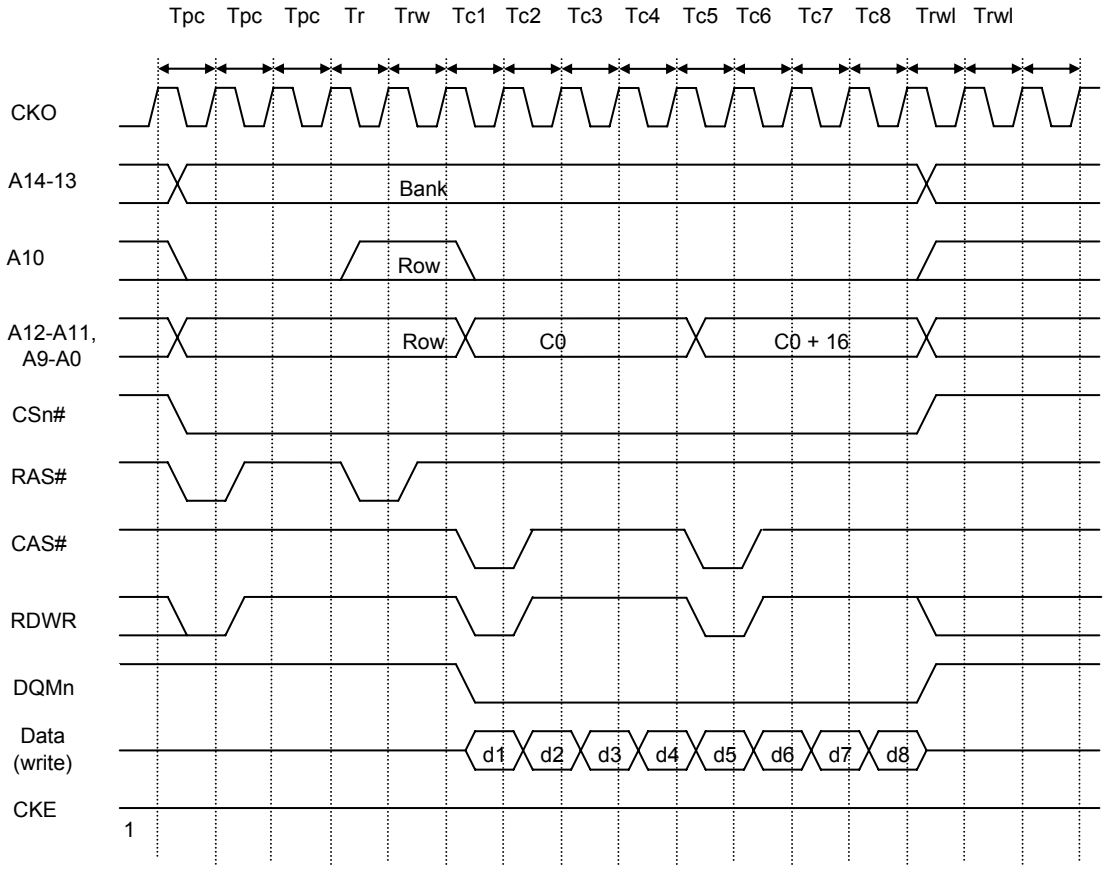
*DMCR: RCD = 1, TCL = 1, TPC = 2

Figure 3-25 Synchronous DRAM 8-beat Burst Read Timing (Same Row)



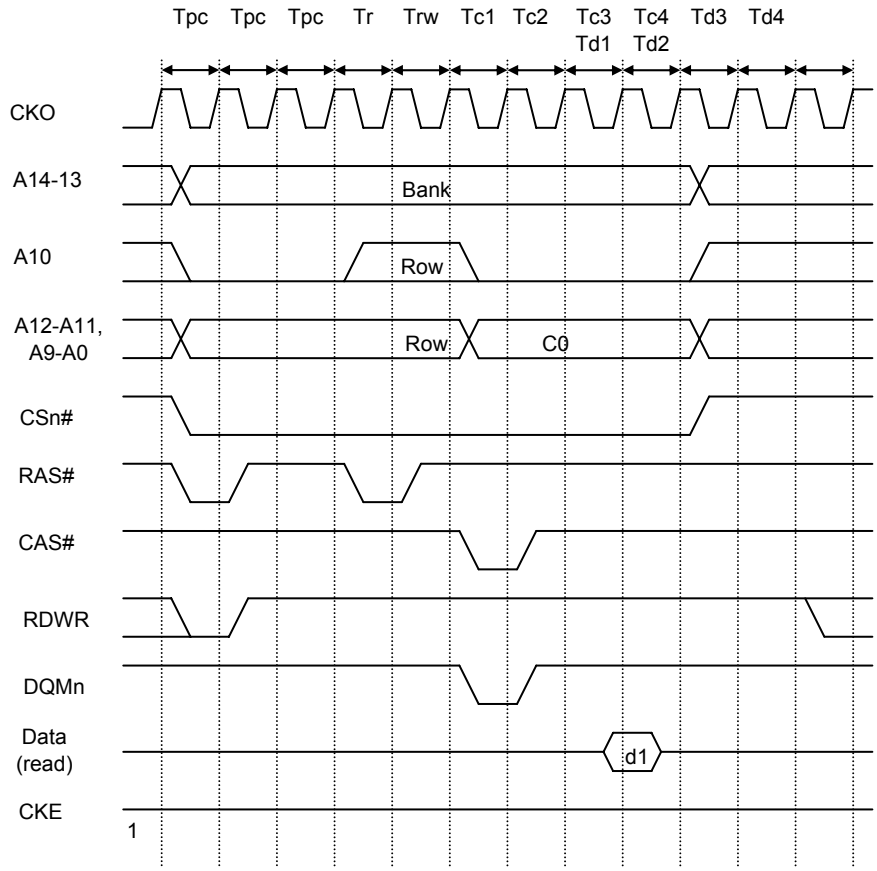
*DMCR: RCD = 1, TCL = 1, TPC = 2, TRWL = 1

Figure 3-26 Synchronous DRAM 8-beat Burst Write Timing (Same Row)



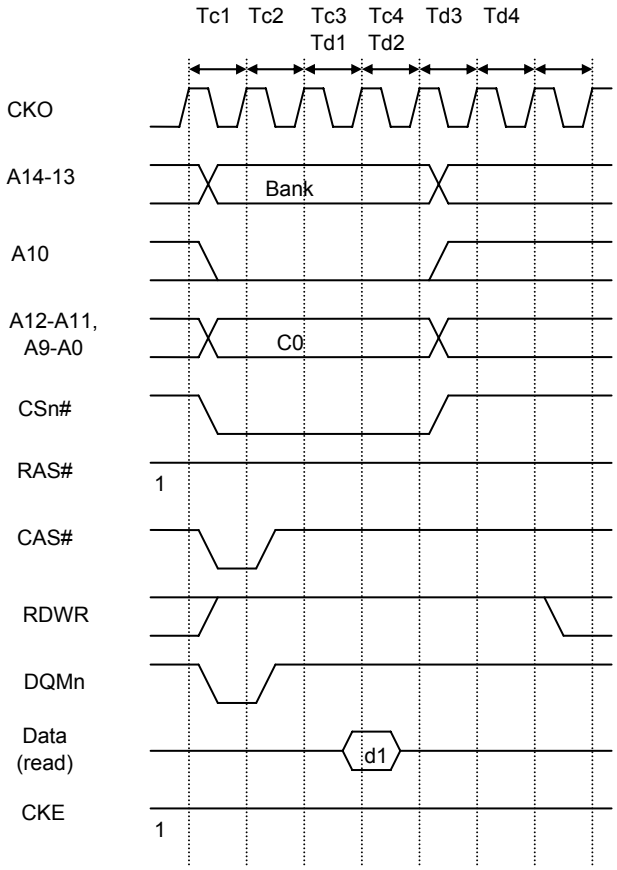
*DMCR: RCD = 1, TCL = 1, TPC = 2, TRWL = 1

Figure 3-27 Synchronous DRAM 8-beat Burst Write Timing (Different Row)



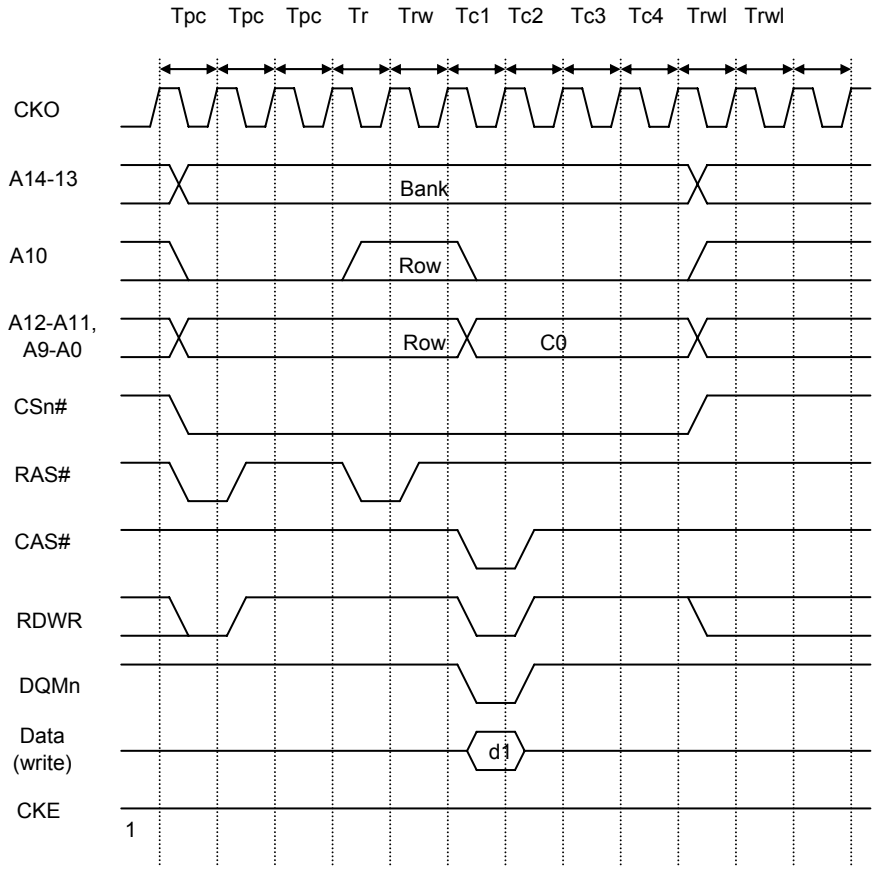
*DMCR: RCD = 1, TCL = 1, TPC = 2

Figure 3-28 Synchronous DRAM Single Read Timing (Different Row)



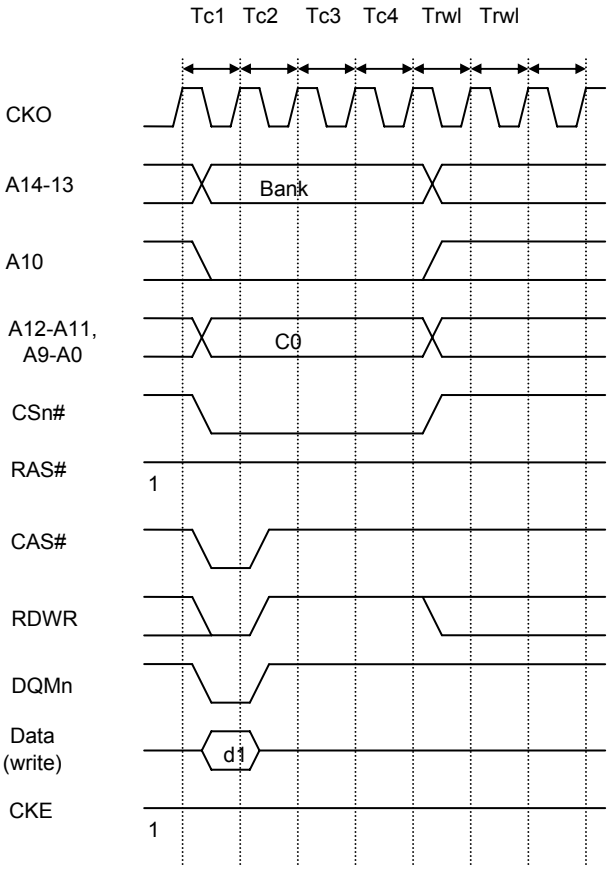
*DMCR: RCD = 1, TCL = 1, TPC = 2

Figure 3-29 Synchronous DRAM Single Read Timing (Same Row)



*DMCR: RCD = 1, TCL = 1, TPC = 2, TRWL = 1

Figure 3-30 Synchronous DRAM Single Write Timing (Different Row)



*DMCR: RCD = 1, TCL = 1, TPC = 2, TRWL = 1

Figure 3-31 Synchronous DRAM Single Write Timing (Same Row)

3.6.7 Power-Down Mode

The SDRAM power-down mode is supported to minimize the power consumption. CKE going to low level when SDRAM is idle/active state will drive SDRAM to precharge/active power-down mode. The clock supplies to SDRAM may be stopped also when CKE keep in low level more than two cycles. When a new access start or a refresh request, CKE is driven to high level and clock supplies is re-enabled. In power-down mode, clock of the accessed SDRAM bank pair is supplied. Clock of the other pair is stopped.

Following figures shows the timing of power-down mode and clock stopping.

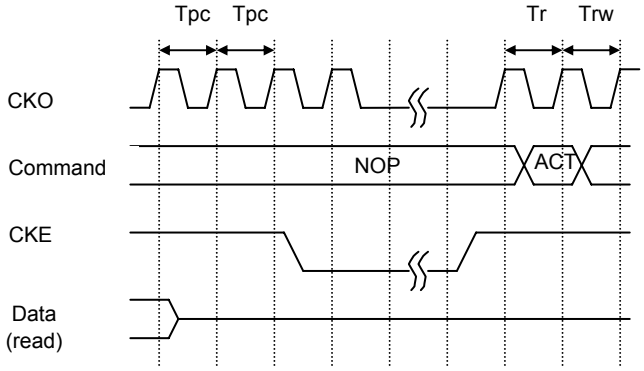


Figure 3-32 SDRAM Power-Down Mode Timing (CKO Stopped)

Following figure shows the power-down mode timing that CKE low level less than two cycles and clock is not stopped.

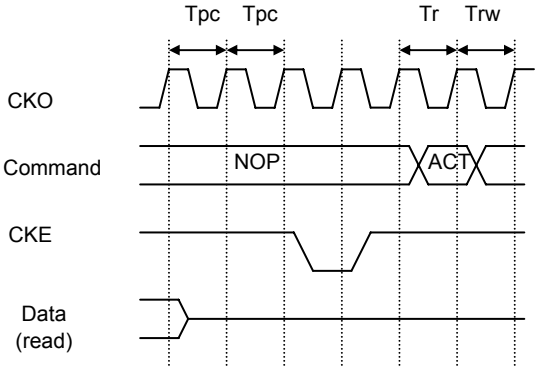


Figure 3-33 SDRAM Power-Down Mode Timing (Clock Supplied)

3.6.8 Refreshing

EMC provide a function for controlling the refresh of synchronous DRAM, Auto-refresh can be performed by clearing the RMODE bit to 0 and setting the RFSH bit to 1 in DMCR. If SDRAM is not accessed for a long period, self-refresh mode can be activated by set both the RMODE bit and the RFSH bit to 1.

3.6.8.1 AUTO-Refresh

Refreshing is performed at intervals determined by the input clock selected by bits CKS2-0 in RTCSR, and the value set in RTCOR. The value of bits CKS2-0 in RTCSR should be set so as to satisfy the refresh interval stipulation for the synchronous DRAM used. First make the settings for RTCOR, RTCNT, and the RMODE and RFSH bits in MCR, and then make the CKS2-CKS0 setting. When the clock is selected by CKS2-CKS0, RTCNT starts counting up from the value at that time. The RTCNT value is constantly compared with the RTCOR value, and if the two values are the same, a refresh request is generated and an auto-refresh is performed. At the same time, RTCNT is cleared to zero and the count-up is restarted. Figure 3-34 shows the auto-refresh cycle operation.

First, a REF command is issued in the TRr cycle. After the TRr cycle, new command output cannot be performed for the duration of the number of cycles specified by the TRC bits in DMCR. The TRC bits must be set so as to satisfy the synchronous DRAM refresh cycle time stipulation (active/active command delay time). Following figure shows the auto-refresh timing when TRC is set to 2.

Auto-refresh is performed in normal operation and sleep mode.

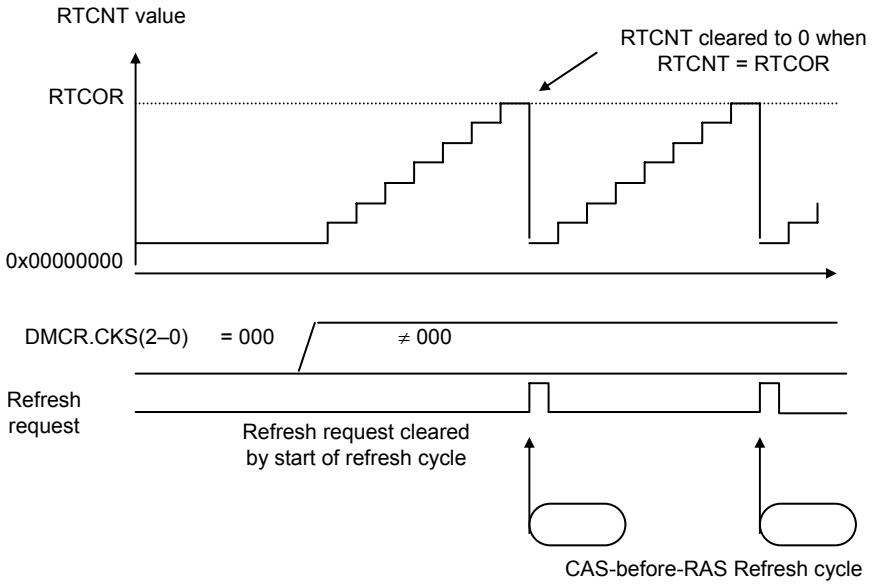


Figure 3-34 Synchronous DRAM Auto-Refresh Operation

A PALL command is issues firstly to precharge all banks. Then a REF command is issued in the TRr

cycle.

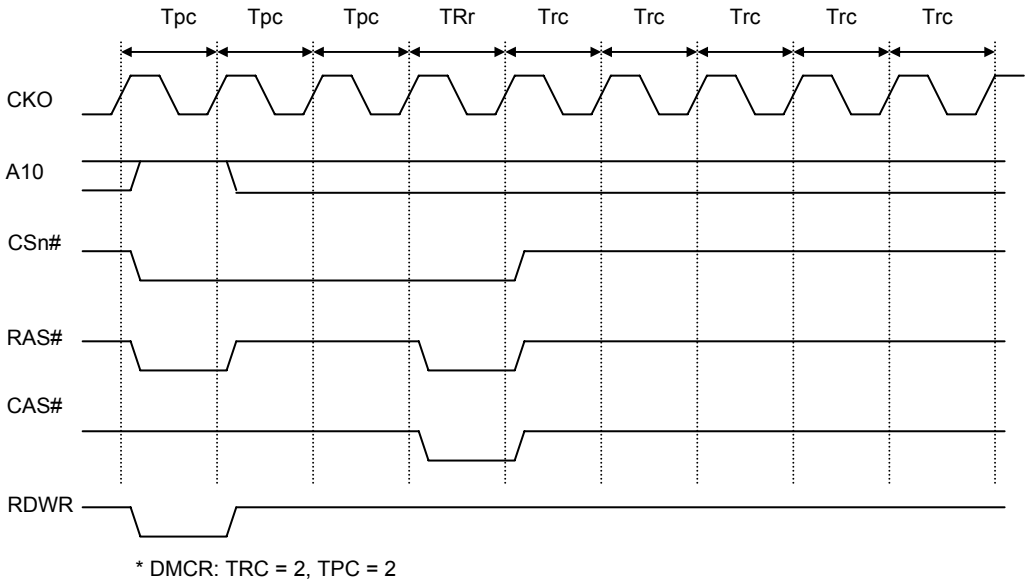


Figure 3-35 Synchronous DRAM Auto-Refresh Timing

3.6.8.2 SELF-Refresh

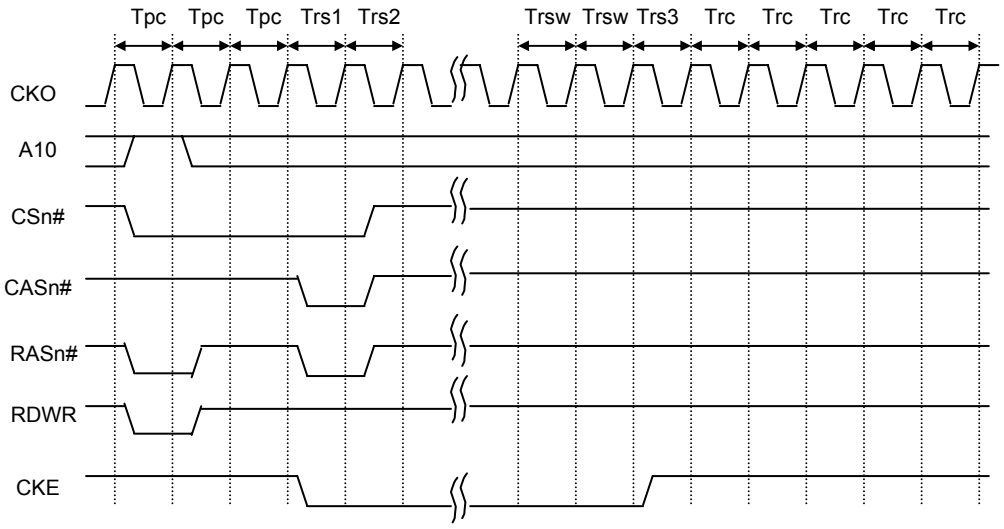
Self-refresh mode is a kind of sleep mode in which the refresh timing and refresh addresses are generated within the SDRAM. Self-refreshing is activated by setting both the RMODE bit and the RFSH bit to 1. The self-refresh state is maintained while the CKE signal is low. SDRAM cannot be accessed while in the self-refresh state. Self-refresh mode is cleared by clearing the RMODE bit to 0. After self-refresh mode has been cleared, command issuance is disabled for the number of cycles specified by the TRC bits in DMCR. Trsw cycles are inserted to meet the minimum CKE negation time specified by the TRAS bits in DMCR. Self-refresh timing is shown in following figure. Settings must be made so that self-refresh clearing and data retention are performed correctly, and auto-refresh is performed at the correct intervals. When self-refreshing is activated from the state in which auto-refreshing is set, or when exiting sleep mode other than through a reset, auto-refreshing is restarted if RFSH is set to 1 and RMODE is cleared to 0 when self-refresh mode is cleared. If the transition from clearing of self-refresh mode to the start of auto-refresh takes time, this time should be taken into consideration when setting the initial value of RTCNT. Making the RTCNT value 1 less than the RTCOR value will enable refreshing to be started immediately. After self-refreshing has been set, the self-refresh state continues even if the chip standby state is entered using the processor's sleep function, and is maintained even after recovery from sleep mode other than through a reset. In the case of a reset, the bus state controller's registers are initialized, and therefore the self-refresh state is cleared.

Self-refreshing is performed in normal operation, in idle mode and in sleep mode. In sleep mode, if RFSH bit in DMCR is 1, self-refresh is always performed in spite of RMODE field in DMCR until sleep mode is canceled.

Relationship between Refresh Requests and Bus Cycle Requests:

If a refresh request is generated during execution of a bus cycle, execution of the refresh is deferred until the bus cycle is completed. If a match between RTCNT and RTCOR occurs while a refresh is waiting to be executed, so that a new Refresh request is generated, the previous refresh request is eliminated. In order for refreshing to be performed normally, care must be taken to ensure that no bus cycle is longer than the refresh interval.

A PALL command is issued firstly to precharge all banks.



* DMCR: TRAS = 0, TRC = 2

Figure 3-36 Synchronous DRAM Self-Refresh Timing

3.6.9 Initialize Sequence

In order to use SDRAM, mode setting must first be performed after powering on. To perform SDRAM initialization correctly, the EMC registers must first be set, followed by a write to the SDRAM mode register.

In SDRAM mode register setting, the address signal value at that time is latched by MRS command. If the value to be set is X, the bus state controller provides for value X to be written to the synchronous DRAM mode register by performing a write to address offset 0x8000 + X for bank 0. In this operation the data is ignored, but the mode write is performed as a byte-size access. To set burst read/write, CAS latency 2 to 3, wrap type = sequential, and burst length 4 supported by the processor, arbitrary data is written in a byte-size access to the following addresses.

Table 3-9 SDRAM Mode Register Setting Address Example (32-bit)

	Bank 0	Bank 1		
CAS latency 2	8022	8022		
CAS latency 3	8032	8032		

Table 3-10 SDRAM Mode Register Setting Address Example (16-bit)

	Bank 0	Bank 1		
CAS latency 2	8011	8011		
CAS latency 3	8019	8011		

The value set in DMCR.MRSET is used to select whether a Pre-charge All Banks command (PALL) or a Mode Register Set command (MRS) is issued. DMCR.MBSEL is used to select Bank 0 or Bank 1 for Mode Register Set. The timing for the Pre-charge All Banks command is shown in Figure 3-37, and the timing for the Mode Register Set command in Figure 3-38.

Before mode register setting, a 200 μs idle time (depending on the memory manufacturer) must be guaranteed after powering on requested by the synchronous DRAM. If the reset signal pulse width is greater than this idle time, there is no problem in performing initialize sequence immediately.

First, a pre-CHARGE all bank (PALL) command must be issued by performing a write to address offset 0x8000 + X for bank 0, while DMCR.MRSET = 0, DMCR.MBSEL = 0.

Next the NUMBER of dummy auto-refresh cycles specified by the manufacturer (usually 8) or more must be executed. This is usually achieved automatically while various kinds of initialization are being performed after auto-refresh setting, but a way of carrying this out more dependably is to set a short refresh request generation interval just while these dummy cycles are being executed. With simple read or write access, the address counter in the synchronous DRAM used for auto-refreshing is not initialized, and so the cycle must always be an auto-refresh cycle.

After auto-REFRESH has been executed at least the prescribed number of times, a Mode Register Set command (MRS) is issued in the TMw1 cycle by setting DMCR.MRSET to 1 and DMCR.MBSEL to 0 for bank 0 or DMCR.MBSEL to 1 for bank 1 and performing a write to address offset 0x8000 + X.

An example of SDRAM operation flow is as the following:

1 Disable Bus release.
Write 0x00000000 to BCR.

2 Initialize RTCOR and RTCNT for auto-refresh cycle.
Before configure SDRAM SDMR, SDRAM needs to execute auto-refresh, the number of times depends on the type of SDRAM. It's better to set a short refresh request generation interval here. For example, set RTCOR to 0x0000000F, and set RTCNT 0x00000000.

3 Initialize DMCR for Precharge all bank and auto-refresh.
When DMCR.RMODE=0 and DMCR.RFSH=1, enter auto-refresh mode;
When DMCR.MRSET=0, DMCR.MBSEL=0 (bank 0) or 1 (bank 1), write SDMR generates Precharge all bank cycle.
DMCR.TPC must be defined for precharge.

4 Disable refresh counter clock.
Write 0x00000000 to RTCSR.

5 Execute Precharge all bank before auto-refresh.
Because DMCR.MRSET=0, DMCR.MBSEL=0 (bank 0) or 1 (bank 1), writing SDMR generates a Precharge all bank cycle, for example, write address (0x13018000).

6 Enable fast refresh counter clock for auto-refresh cycle.
For example, write 0x00000001 to RTCSR.

7 Wait for number of auto-refresh cycles. (defined by SDRAM chip)
When RTCSR.CMF=1, it indicates value of RTCOR and RTCNT match and an auto-refresh cycle occurs.

8 Configure DMCR for SDRAM MODE Register Set.
When DMCR.MRSET=1, DMCR.MBSEL=0 (bank 0) or 1 (bank 1), write SDMR generate MRSET cycle.
For example, write 0x059A5231 to DMCR, so that:
Bus-width: 32-bit; Column Address: 9-bit; Row Address: 12-bit; Auto-refresh mode; SDMR Set mode; 4-bank; etc..

9 SDRAM Mode Register Set.
Because DMCR.MRSET=1 and DMCR.MBSEL=0, for example, write address 0x13018022 to configure SDMR as:
Burst Length: 4 burst
Burst Type: Sequential
CAS Latency: 2

10 Set normal auto-refresh counter clock.
For example, write 0x00000005 to RTCSR.

11 Then Read/Write SDRAM can be executed.

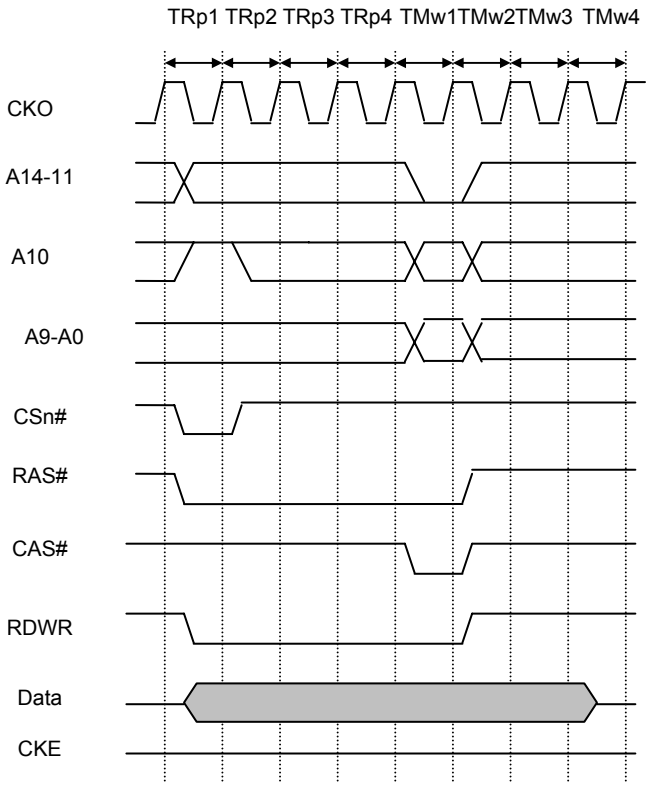


Figure 3-37 SDRAM Mode Register Write Timing 1 (Pre-charge All Banks)

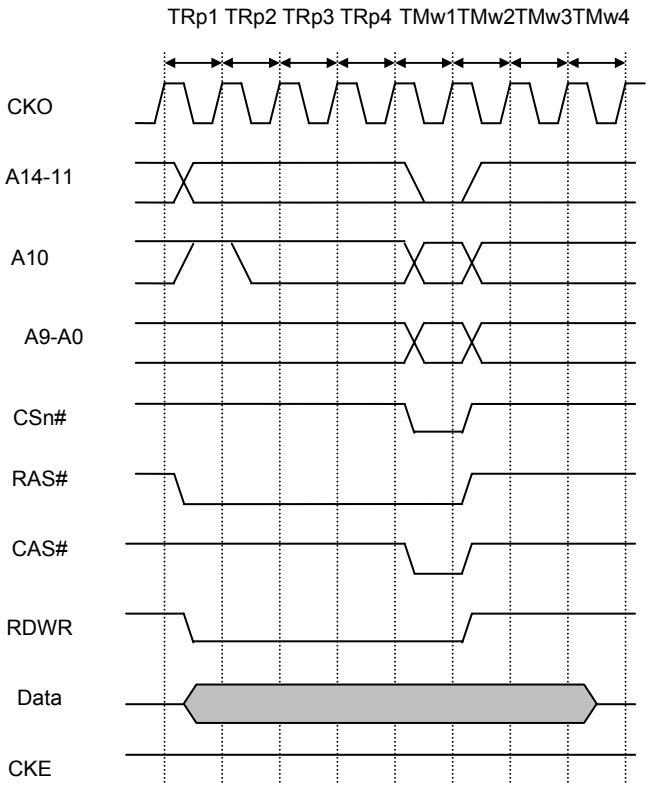


Figure 3-38 SDRAM Mode Register Write Timing 2 (Mode Register Set)

3.7 Bus Control Register (BCR)

BCR is used to specify the behavior of EMC on system bus. It is initialized to 0x00000001 by any reset.

Name	Description	RW	Reset Value	Address	Access Width
BCR	Bus Control Register	RW	0x?0000001	0x13010000	32

BCR **0x13010000**

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

Bits	Name	Description	RW
31:25	Reserved	Writes to these bits has no effect and always read as 0.	R
24	PK_SEL	**PKG Select.** 0: 32/16-bit data normal order; 1:16-bit data special order.	R
23:3	Reserved	Writes to these bits has no effect and always read as 0.	R
2	BSR	**Bus Share Select.** 0: Nand and SDRAM bus share; 1: Nand and SDRAM bus separate.	RW
1	BRE	**Bus Release Enable**: When clear, once a transaction to EMC begins on the system bus; it must be completed before another transaction starts. When set, the system bus may be released to allow other transaction before EMC prepare the read data or be able to receipt the write data. If slow memory devices are used in the system, setting this bit will improve the efficiency of the whole system. The efficiency of SDRAM access may be improved by setting this bit. But the power consumption is increased if this bit is set. **BRE** **Description** 0 The system bus can not be released during an access (Initial value) 1 The system bus can be released during an access	RW
0	Endian	**Endian:** Indicates the system is little-endian.	R

4 BCH Controller

4.1 Overview

The BCH Controller implements data ECC encoding and decoding.

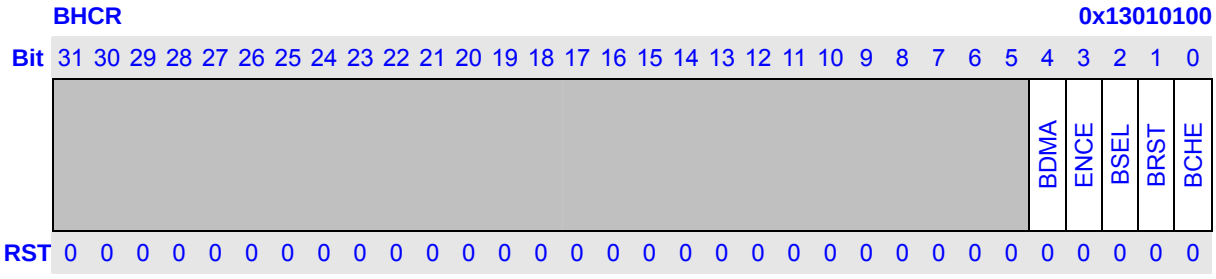
4.2 Register Description

Table 4-1 BCH Registers

Name	Description	RW	Reset Value	Address	Access Width
BHCR	BCH Control register	R	0x00000000	0x130D0000	32
BHCSR	BCH Control Set register	W	Undefined	0x130D0004	32
BHCCR	BCH Control Clear register	W	Undefined	0x130D0008	32
BHCNT	BCH ENC/DEC Count register	RW	0x00000000	0x130D000C	32/16
BHDR	BCH data register	W	Undefined	0x130D0010	8
BHPAR0	BCH Parity 0 register	RW	0x00000000	0x130D0014	32/16/8
BHPAR1	BCH Parity 1 register	RW	0x00000000	0x130D0018	32/16/8
BHPAR2	BCH Parity 2 register	RW	0x00000000	0x130D001C	32/16/8
BHPAR3	BCH Parity 3 register	RW	0x00000000	0x130D0020	32/16/8
BHINT	BCH Interrupt Status register	R	0x00000000	0x130D0024	32
BHERR0	BCH Error Report 0 register	R	0x00000000	0x130D0028	32/16
BHERR1	BCH Error Report 1 register	R	0x00000000	0x130D002C	32/16
BHERR2	BCH Error Report 2 register	R	0x00000000	0x130D0030	32/16
BHERR3	BCH Error Report 3 register	R	0x00000000	0x130D0034	32/16
BHINTE	BCH Interrupt Enable register	RW	0x00000000	0x130D0038	32
BHINTES	BCH Interrupt Set register	W	Undefined	0x130D003C	32
BHINTEC	BCH Interrupt Clear register	W	Undefined	0x130D0040	32

4.2.1 BCH Control Register (BHCR)

BHCR is a 32-bit read/write register that is used to configure BCH controller. It is initialized by any reset.



Bits	Name	Description	RW
31:5	Reserved	Writes to these bits have no effect and read always as 0.	R
4	BDMA	**BCH DMA Enable**: It is used to enable or disable dma transfer during correction. **BDMA** **Description** 0 DMA transfer is disabled (Initial value) 1 DMA transfer is enabled	RW
3	ENCE	**BCH Encoding/Decoding Select**: It is used to define whether in encoding or in decoding phase when BCH is used. **ENCE** **Description**	RW

		0 Decoding (Initial value) 1 Encoding	
2	BSEL	**4/8 Bit BCH Select**: It is used to select the correction algorithm between 4-bit and 8-bit BCH. **BSE** **Description** 0 4-bit correction (Initial value) 1 8-bit correction	RW
1	BRST	**BCH Reset:** It is used to reset BCH controller. This bit is cleared automatically by hardware and always read as 0. **BRST** **Description** 0 BCH controller is not reset (Initial value) 1 BCH controller is reset	W
0	BCHE	**BCH Enable:** BCH correction is enable/disable. **BCHE** **Description** 0 BCH is disabled (initial value) 1 BCH is enabled	RW

4.2.2 BCH Control Set Register (BHCSR)

BHCSR is a 32-bit write-only register that is used to set BCH controller to 1.

When write 1 to BHCSR, the corresponding bit in BHCR register is set to 1. Write 0 to BHCSR is ignored.

BHCSR **0x13010100**

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

Bits	Name	Description	RW
31:5	Reserved	Writes to these bits have no effect and read always as 0.	R
4	BDMAS	**BCH DMA Enable Set**: It is used to set BHCR.BDMA to 1.	W
3	ENCES	**BCH Encoding/Decoding Select Set**: It is used to set BHCR.ENCE to 1.	W
2	BSELS	**4/8 Bit BCH Select Set**: It is used to set BHCR.BSEL to 1.	W
1	BRSTS	**BCH Reset Set:** It is used to set BHCR.BRST to 1.	W
0	BCHES	**BCH Enable Set:** It is used to set BHCR.BCHE to 1.	W

4.2.3 BCH Control Clear Register (BHCCR)

BHCCR is a 32-bit write-only register that is used to clear BCH controller to 0.

When write 1 to BHCCR, the corresponding bit in BHCR register is cleared to 0. Write 0 to BHCCR is ignored.

BHCCR **0x13010100**

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

Bits	Name	Description	RW
31:5	Reserved	Writes to these bits have no effect and read always as 0.	R
4	BDMAC	**BCH DMA Enable Clear**: It is used to clear BHCR.BDMA to 0.	W
3	ENCEC	**BCH Encoding/Decoding Select Clear**: It is used to clear BHCR.ENCE to 0.	W
2	BSELC	**4/8 Bit BCH Select Clear**: It is used to clear BHCR.BSEL to 0.	W
1	Reserved	Writes to this bit have no effect and read always as 0.	R
0	BCHEC	**BCH Enable Clear:** It is used to clear BHCR.BCHE to 0.	W

4.2.4 BCH ENC/DEC Count Register (BHCNT)

BHCNT is a 32-bit read/write register that is used to indicate the total number of bytes during encoding or decoding. It is initialized by any reset.

BHCNT **0x13010100**

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

Bits	Name	Description	RW
31:26	Reserved	Writes to these bits have no effect and read always as 0.	R
25:16	DEC Count	**DEC Count**: It is used to indicate total byte count in BCH decoding which includes data bytes + parity bytes.	RW
15:10	Reserved	Writes to these bits have no effect and read always as 0.	R
9:0	ENC Count	**ENC Count**: It is used to indicate total byte count in BCH encoding which just includes data bytes and should be less and equal to 1010 bytes when 8-bit BCH is selected and 1016 bytes when 4-bit BCH is selected.	RW

4.2.5 BCH Data Register (BHDR)

BHDR is an 8-bit write-only register that is used to transfer ecc data to BCH.

BHDR **0x130100104**

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

4.2.6 BH Parity Register (BHPARn, n=0,1,2,3)

BHPAR0, BHPAR1, BHPAR2 and BHPAR3 are all 32-bit read/write register that contains the encoding parity data during BCH correction. It is initialized by any reset and BRST of BHCR.

When 8-bit BCH is selected, the four parity register, BHPAR0, BHPAR1, BHPAR2 and BHPAR3 together consist of the 104 bits of parity data and bit 0 of BHPAR0 is the 104th bit of parity data and bit 7 of BHPAR3 is the 1st bit of parity data.

Similarly, when 4-bit BCH is selected, the two parity register, BHPAR0 and BHPAR1 together consist of the 52 bits of parity data and bit 0 of BHPAR0 is the 52th bit of parity data and bit 19 of BHPAR1 is the 1st bit of parity data.

BHPAR0 **0x13010108**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																									BHPAR0							
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BHPAR1 **0x1301010C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																									BHPAR1							
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BHPAR2 **0x1301010C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	BHPAR2																															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BHPAR3 **0x13010110**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								BHPAR3																							
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

4.2.7 BCH Interrupt Status Register (BHINT)

BHINT is a 32-bit read-only register that contains the interrupt flag and error count information during BCH correction. It is initialized by any reset. Software write 1 to clear the corresponding bit except ERRC.

BHINT **0x13010114**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ERRC				Reserved																						ALL_0	ALL_f	DECF	ENCF	UNCOR	ERR
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:28	ERRC	**Error Count:** It indicates the number of errors in the data block and these bits are also reset by BHCR.BRST bit. **ERRC** **Description** 0 No errors or uncorrection error occurs (Initial value) 1 One error in the data block 2 Two errors in the data block 3 Three errors 4 Four errors 5 Five errors 6 Six errors 7 Seven errors 8 Eight errors	R
27:6	Reserved	Writes to these bits have no effect and read always as 0.	R

5	ALL_0	**ALL_0:** It indicates that all data received during decoding are 0x0. **ALL_0 Description** 0 Not all data (data + parity bytes) are 0x0 (Initial value) 1 All data (data + parity bytes) are 0x0	R
4	ALL_f	**ALL_f:** It indicates that all data received during decoding are 0xf. When receiving all 0xf data, BCH doesn't correct the data and no error occurs. **ALL_f Description** 0 Not all data (data + parity bytes) are 0xf (Initial value) 1 All data (data + parity bytes) are 0xf	R
3	DECF	**Decoding Finish:** It indicates that hardware finish BCH decoding. **DECF Description** 0 Decoding not Finish (Initial value) 1 Decoding Finish	R
2	ENCF	**Encoding Finish:** It indicates that hardware finish BCH encoding. **ENCF Description** 0 Encoding not Finish (Initial value) 1 Encoding Finish	R
1	UNCOR	**Uncorrection Error:** It indicates that hardware finish BCH encoding. **UNCOR Description** 0 No uncorrectable error (Initial value) 1 Uncorrectable error occur	R
0	ERR	**Error:** It indicates that hardware detects error bits in data in the data block during BCH decoding. **ERR Description** 0 No error (Initial value) 1 Error occur	R

4.2.8 BCH Interrupt Enable Set Register (BHINTES)

BHINTES is a 32-bit write-only register that is used to set BHINTE register. Writing 1 to BHINTES will set the corresponding bit in BHINTE to 1. Writing 0 to BHINTES is ignored.

BHINTES **0x13010118**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																										ALL_0ES	ALL_FES	DECFES	ENCFES	UNCORES	ERRES
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

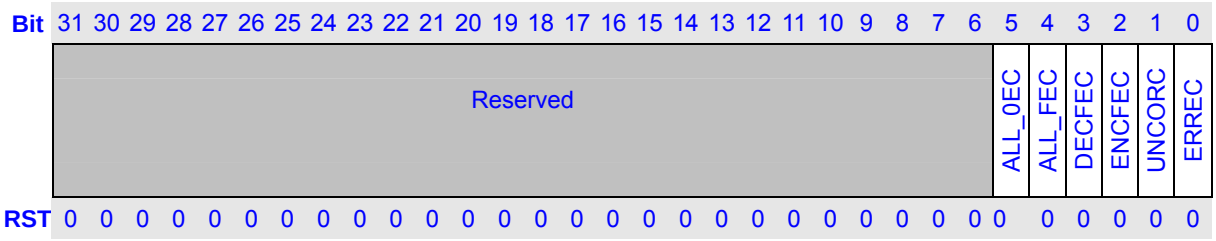
Bits	Name	Description	RW
31:6	Reserved	Writes to these bits have no effect and read always as 0.	R
5	ALL_0ES	**ALL_0 Interrupt Enable Set:** It is used to set BHINTE.ALL_0E to 1.	W

4	ALL_FES	**ALL_F Interrupt Enable Set:** It is used to set BHINTE.ALL_FE to 1.	W
3	DECFES	**Decoding Finish Interrupt Enable Set:** It is used to set BHINTE.DECFE to 1.	W
2	ENCFES	**Encoding Finish Interrupt Enable Set:** It is used to set BHINTE.ENCFE to 1.	W
1	UNCORES	**Uncorrection Error Interrupt Enable Set:** It is used to set BHINTE.ENCFE to 1.	W
0	ERRES	**Error Interrupt Enable Set:** It is used to set BHINTE.ERRE to 1.	W

4.2.9 BCH Interrupt Enable Clear Register (BHINTEC)

BHINTEC is a 32-bit write-only register that is used to clear BHINTE register. Writing 1 to BHINTEC will clear the corresponding bit in BHINTE to 0. Writing 0 to BHINTEC is ignored.

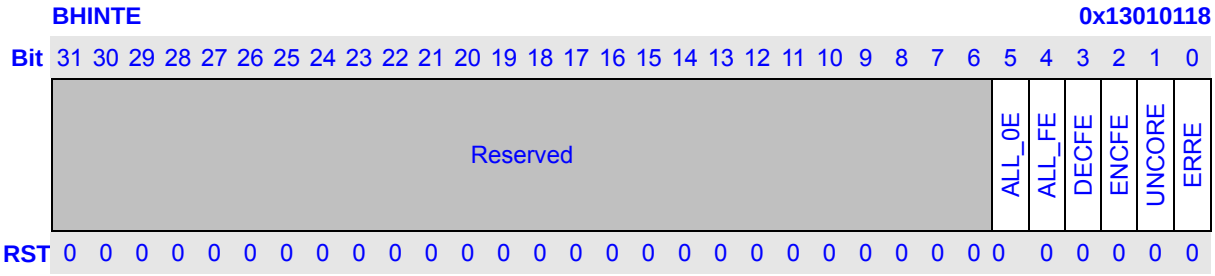
BHINTEC **0x13010118**



Bits	Name	Description	RW
31:6	Reserved	Writes to these bits have no effect and read always as 0.	R
5	ALL_0EC	**ALL_0 Interrupt Enable Clear:** It is used to clear BHINTE.ALL_0E to 0.	W
4	ALL_FEC	**ALL_F Interrupt Enable Clear:** It is used to clear BHINTE.ALL_FE to 0.	W
3	DECFEC	**Decoding Finish Interrupt Enable Clear:** It is used to clear BHINTE.DECFE to 0.	W
2	ENCFEC	**Encoding Finish Interrupt Enable Clear:** It is used to clear BHINTE.ENCFE to 0.	W
1	UNCOREC	**Uncorrection Error Interrupt Enable Clear:** It is used to clear BHINTE.ENCFE to 0.	W
0	ERREC	**Error Interrupt Enable Clear:** It is used to set BHINTE.ERRE to 0.	W

4.2.10 BCH Interrupt Enable Register (BHINTE)

BHINTE is a 32-bit read/write register that is used to enable/disable interrupts during BCH correction. It is initialized by any reset.



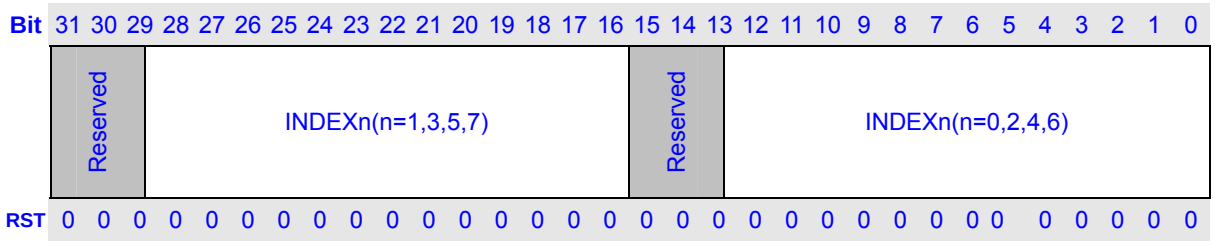
Bits	Name	Description	RW
31:6	Reserved	Writes to these bits have no effect and read always as 0.	R
5	ALL_0E	**ALL_0 Interrupt Enable:** It is used to enable or disable all_0 data interrupt. **ALL_0E Description** 0 Disable ALL_0 data interrupt (Initial value) 1 Enable ALL_0 data interrupt	RW
4	ALL_FE	**ALL_F Interrupt Enable:** It is used enable or disable all_f data interrupt. **ALL_FE Description** 0 Disable ALL_F data interrupt (Initial value) 1 Enable ALL_F data interrupt	RW
3	DECFE	**Decoding Finish Interrupt Enable:** It is used to enable or disable decoding finish interrupt. **DECFE Description** 0 Disable Decoding Finish Interrupt (Initial value) 1 Enable Decoding Finish Interrupt	RW
2	ENCFE	**Encoding Finish Interrupt Enable:** It is used to enable or disable encoding finish interrupt. **ENCFE Description** 0 Disable Encoding Finish Interrupt (Initial value) 1 Enable Encoding Finish Interrupt	RW
1	UNCORE	**Uncorrection Error Interrupt Enable:** It is used to enable or disable uncorrection error interrupt. **UNCORE Description** 0 Disable Uncorrectable Error interrupt (Initial value) 1 Enable Uncorrectable Error Interrupt	RW
0	ERRE	**Error Interrupt Enable:** It is used to enable or disable error interrupt. **ERRE Description** 0 Disable Error interrupt (Initial value) 1 Enable Error interrupt	RW

4.2.11 BCH Error Report Register (BHERRn, n=0,1,2,3)

BHERRn is 32-bit read/write register that contains the index for each error after BCH decoding. It is initialized by any reset and BRST of BHCR.

BHERR0 contains INDEX0 and INDEX1.
BHERR1 contains INDEX2 and INDEX3.
BHERR2 contains INDEX4 and INDEX5.
BHERR3 contains INDEX6 and INDEX7.

BHERR0 **0x1301011C**
BHERR1 **0x13010120**
BHERR2 **0x13010124**
BHERR3 **0x13010128**



Bits	Name	Description	RW
31:29	Reserved	Writes to these bits have no effect and read always as 0.	R
28:16	INDEXn	**Error Bit Index:** It is used to indicate the location of the error bit. For example, INDEX=2, it means the second bit is an error bit.	R
15:13	Reserved	Writes to these bits have no effect and read always as 0.	R
12:0	INDEXn	**Error Bit Index:** It is used to indicate the location of the error bit. For example, INDEX=1, it means the first bit is an error bit.	R

4.3 BCH Operation

BCH controller uses BCH(n, k) codes. Here n is less and equal to 8191-bit and k is less and equal to 8087-bit in 8-bit correction and 8139-bit in 4-bit correction. During encoding, hardware will generate 104-bit parity data in 8-bit correction or 52-bit parity data in 4-bit correction. Parity data can be read out by cpu or dma. During decoding, if there are error bits in data block, after decoding BHERRn registers will hold the error bit location that can be read by cpu or dma.

4.3.1 Encoding Sequence

BCH encoding can be operated by cpu or dma.

4.3.1.1 CPU

1 Set BHCR.BCHE to 1 to enable BCH controller.
2 Select 4-bit or 8-bit correction by setting BHCR.BSEL.
3 Set BHCR.ENCE to 1 to enable encoding.
4 Set BHCR.BRST to 1 to reset BCH controller.
5 Set BHCNT.ENC_COUNT to data block size in bytes.
6 Byte-write all data block to BHDR.
7 Check BHINTS.ENCF bit or by enabling encoding finish interrupt.
8 When encoding finishes, read out the parity data in BHPARn.

4.3.1.2 DMA

1 Set BHCR.BCHE to 1 to enable BCH controller.
2 Select 4-bit or 8-bit correction by setting BHCR.BSEL.
3 Set BHCR.ENCE to 1 to enable encoding.
4 Set BHCR.BRST to 1 to reset BCH controller.
5 Set BHCNT.ENC_COUNT to data block size in bytes.
6 Set BHCR.BDMA to 1 to select DMA transfer.
7 Start DMA transfer after configuring DMA channel.
8 DMA read data block from system memory and write to BCH controller automatically.
9 DMA will wait BCH encoding request when finishes writing data block.
10 BCH controller will issue encoding request to DMA when encoding ends.
11 DMA start to read out parity data.
12 After parity data is read out, BCH automatically reset itself and clear BHINT.ENCF.

NOTES:

1 When DMA is enabled, software should guarantee not to enable encoding finish interrupt.

4.3.2 Decoding Sequence

BCH decoding can be operated by cpu or dma.

4.3.2.1 CPU

1 Set BHCR.BCHE to 1 to enable BCH controller.
2 Select 4-bit or 8-bit correction by setting BHCR.BSEL.
3 Clear BHCR.ENCE to 0 to enable decoding.
4 Set BHCR.BRST to 1 to reset BCH controller.
5 Set BHCNT.DEC_COUNT to data block size in bytes.
6 Byte-write all data block to BHDR.
7 Check BHINTS.DECF bit or by enabling decoding finish interrupt.
8 When decoding finishes, read out the status in BHINT and error report in BHERRn.

4.3.2.2 Decoding Sequence

1 Set BHCR.BCHE to 1 to enable BCH controller.
2 Select 4-bit or 8-bit correction by setting BHCR.BSEL.
3 Clear BHCR.ENCE to 0 to enable decoding.
4 Set BHCR.BRST to 1 to reset BCH controller.
5 Set BHCNT.DEC_COUNT to data block size in bytes.
6 Set BHCR.BDMA to 1 to select DMA transfer.
7 Start DMA transfer after configuring DMA channel.
8 DMA read data block from system memory and write to BCH controller automatically.
9 DMA will wait BCH decoding request when finishes writing data block.
10 BCH controller will issue decoding request to DMA when decoding ends.
11 DMA start to read out bch int status and error report data and write to memory.
12 If using descriptor DMA, if the data block needs error correction, the current data block syndrome generation and last data block error correction can be executed in pipeline automatically by DMA .
13 After status and error report data is read out, BCH automatically reset itself and clear BHINT.DECF and Error status in BHINT.

NOTES:

1 If the data block is all 0xf, BCH will set All_f bit in BHINT and doesn't do error correction.

5 DMA Controller

DMA controller (DMAC) is dedicated to transfer data between on-chip peripherals (MSC, AIC, UART, etc.), external memories, and memory-mapped external devices.

5.1 Features

- Support up to 6 independent DMA channels
- Descriptor or No-Descriptor Transfer
- Transfer data units: byte, 2-byte (half word), 4-byte (word), 16-byte or 32-byte
- Transfer number of data unit: 1 ~ 2^{24}
- Independent source and target port width: 8-bit, 16-bit, 32-bit
- Two channel priority modes: fixed, round robin

5.2 Register Descriptions

Table 5-1 DMAC Registers

Name	Description	RW	Reset Value	Address	Access Size (bit)
DSA0	DMA Source Address 0	RW	0x0	0x13020000	32
DTA0	DMA Target Address 0	RW	0x0	0x13020004	32
DTC0	DMA Transfer Count 0	RW	0x0	0x13020008	32
DRT0	DMA Request Source 0	RW	0x0	0x1302000C	32
DCS0	DMA Channel Control/Status 0	RW	0x0	0x13020010	32
DCM0	DMA Command 0	RW	0x0	0x13020014	32
DDA0	DMA Descriptor Address 0	RW	0x0	0x13020018	32
DSA1	DMA Source Address 1	RW	0x0	0x13020020	32
DTA1	DMA Target Address 1	RW	0x0	0x13020024	32
DTC1	DMA Transfer Count 1	RW	0x0	0x13020028	32
DRT1	DMA Request Source 1	RW	0x0	0x1302002C	32
DCS1	DMA Channel Control/Status 1	RW	0x0	0x13020030	32
DCM1	DMA Command 1	RW	0x0	0x13020034	32
DDA1	DMA Descriptor Address 1	RW	0x0	0x13020038	32
DSA2	DMA Source Address 2	RW	0x0	0x13020040	32
DTA2	DMA Target Address 2	RW	0x0	0x13020044	32
DTC2	DMA Transfer Count 2	RW	0x0	0x13020048	32
DRT2	DMA Request Source 2	RW	0x0	0x1302004C	32
DCS2	DMA Channel Control/Status 2	RW	0x0	0x13020050	32
DCM2	DMA Command 2	RW	0x0	0x13020054	32
DDA2	DMA Descriptor Address 2	RW	0x0	0x13020058	32
DSA3	DMA Source Address 3	RW	0x0	0x13020060	32
DTA3	DMA Target Address 3	RW	0x0	0x13020064	32
DTC3	DMA Transfer Count 3	RW	0x0	0x13020068	32
DRT3	DMA Request Source 3	RW	0x0	0x1302006C	32
DCS3	DMA Channel Control/Status 3	RW	0x0	0x13020070	32
DCM3	DMA Command 3	RW	0x0	0x13020074	32
DDA3	DMA Descriptor Address 3	RW	0x0	0x13020078	32
DSA4	DMA Source Address 4	RW	0x0	0x13020080	32
DTA4	DMA Target Address 4	RW	0x0	0x13020084	32
DTC4	DMA Transfer Count 4	RW	0x0	0x13020088	32
DRT4	DMA Request Source 4	RW	0x0	0x1302008C	32
DCS4	DMA Channel Control/Status 4	RW	0x0	0x13020090	32
DCM4	DMA Command 4	RW	0x0	0x13020094	32
DDA4	DMA Descriptor Address 4	RW	0x0	0x13020098	32
DSA5	DMA Source Address 5	RW	0x0	0x130200A0	32

DDA5	DMA Target Address 5	RW	0x0	0x130200A4	32
DTC5	DMA Transfer Count 5	RW	0x0	0x130200A8	32
DRT5	DMA Request Source 5	RW	0x0	0x130200AC	32
DCS5	DMA Channel Control/Status 5	R/W	0x0	0x130200B0	32
DCM5	DMA Command 5	RW	0x0	0x130200B4	32
DDA5	DMA Descriptor Address 5	RW	0x0	0x130200B8	32
DSD0	DMA Stride Address 0	RW	0x0	0x130200C0	32
DSD1	DMA Stride Address 1	RW	0x0	0x130200C4	32
DSD2	DMA Stride Address 2	RW	0x0	0x130200C8	32
DSD3	DMA Stride Address 3	RW	0x0	0x130200CC	32
DSD4	DMA Stride Address 4	RW	0x0	0x130200D0	32
DSD5	DMA Stride Address 5	RW	0x0	0x130200D4	32

DMAC1	DMA Control 1 Register	R/W	0x0	0x13020300	32
DIRQP1	DMA Interrupt Pending 1	R	0x0	0x13020304	32
DDR1	DMA Doorbell 1 Register	RW	0x0	0x13020308	32
DDRS1	DMA Doorbell Set 1 Register	W	0x0	0x1302030C	32
DCKE1	DMA Clock Enable 1 Register	W	0x0	0x13020310	32

5.2.1 DMA Source Address (DSAn, n = 0 ~ 5)

DSA0, DSA1, DSA2,
DSA3, DSA4, DSA5

0x13020000, 0x13020020, 0x13020040,
0x13020060, 0x13020080, 0x130200a0

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

Bits	Name	Description	RW
31:0	SA	Source physical address.	RW

5.2.2 DMA Target Address (DTAn, n = 0 ~ 5)

DTA0, DTA1, DTA2,
DTA3, DTA4, DTA5

0x13020004, 0x13020024, 0x13020044,
0x13020064, 0x13020084, 0x130200a4

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

Bits	Name	Description	RW
31:0	TA	Target physical address.	RW

5.2.3 DMA Transfer Count (DTCn, n = 0 ~ 5)

DTC0, DTC1, DTC2,
DTC3, DTC4, DTC5

0x13020008, 0x13020028, 0x13020048,
0x13020068, 0x13020088, 0x130200a8

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

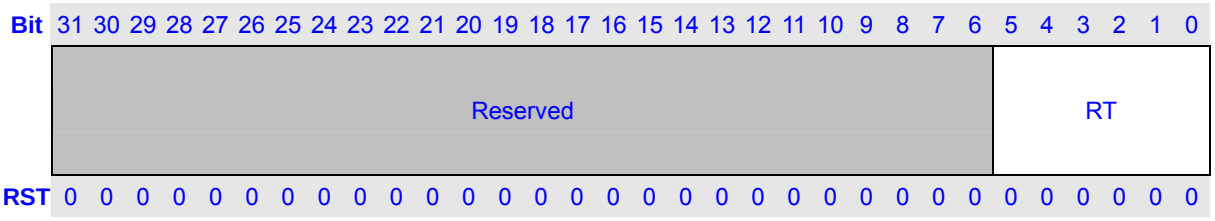
Bits	Name	Description	RW
31:24	Reserved	Write has no effect, read as zero.	R

23:0	TC	When Stride address transfer is disabled: TC hold the number of data unit to transfer and it counts down to 0 at the end; When Stride address transfer is enabled: TC composes of two parts: The lower 16 bits: the number of data unit for sub-block transfer The higher 8 bits: the number of sub-block And both the two parts count down to 0 at the end.	RW

5.2.4 DMA Request Types (DRTn, n = 0 ~ 5)

DRT0, DRT1, DRT2,
DRT3, DRT4, DRT5

0x1302000c, 0x1302002c, 0x1302004c,
0x1302006c, 0x1302008c, 0x130200ac



Bits	Name	Description	R/W
31:6	Reserved	Write has no effect, read as zero.	R
5:0	RT	Transfer request type.	RW

Table 5-2 Transfer Request Types

RT5-0	Description
000000	External request with DREQn. (external address ←→ external device with DACKn)
000001	NAND DMA request. (external address → external address)
000010	BCH Encoding DMA request.
000011	BCH Decoding DMA request.
000100	Reserved.
000101	Reserved.
000110	Reserved.
000111	Reserved.
001000	Auto-request. (ignore RDIL3-0, external address → external address)
001001	TSSI receive-fifo-full transfer request. (TS fifo → external address)
001010	Reserved.
001011	Reserved.
001100	Reserved.
001101	Reserved.

001110	UART3 transmit-fifo-empty transfer request. (external address → UTHR)
001111	UART3 receive-fifo-full transfer request.(URBR → external address)
010000	UART2 transmit-fifo-empty transfer request. (external address → UTHR)
010001	UART2 receive-fifo-full transfer request. (URBR → external address)
010010	UART1 transmit-fifo-empty transfer request. (external address → UTHR)
010011	UART1 receive-fifo-full transfer request. (URBR → external address)
010100	UART0 transmit-fifo-empty transfer request. (external address → UTHR)
010101	UART0 receive-fifo-full transfer request. (URBR → external address)
010110	SSI transmit-fifo-empty transfer request.
010111	SSI receive-fifo-full transfer request.
011000	AIC transmit-fifo-empty transfer request.
011001	AIC receive-fifo-full transfer request.
011010	MSC transmit-fifo-empty transfer request.
011011	MSC receive-fifo-full transfer request.
011100	TCU channel n. (overflow interrupt, external address→external address space)
011101	SADC transfer request. (SADC → external address)
011110	MSC1 transmit-fifo-empty transfer request.
011111	MSC1 receive-fifo-full transfer request.
100000	SSI1 transmit-fifo-empty transfer request.
100001	SSI1 receive-fifo-full transfer request.
100010	PM transmit-fifo-empty transfer request.
100011	PM receive-fifo-full transfer request.
Other	Reserved.

NOTES:

1 Only auto request can be concurrently selected in all channels with different source and target address.

2 For on-chip device DMA request except TCU, the corresponding source or target address that map to on-chip device must be set as fixed.

5.2.5 DMA Channel Control/Status (DCSn, n = 0 ~ 5)

DCS0, DCS1, DCS2, **0x13020010, 0x13020030, 0x13020050,**
DCS3, DCS4, DCS5 **0x13020070, 0x13020090, 0x130200b0**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NDES	DES8	Reserved						CDOA								Reserved								BERR	INV	Reserved	AR	TT	HLT	CT	CTE
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31	NDES	Descriptor or No-Descriptor Transfer Select. 0: Descriptor Transfer; 1: No-descriptor Transfer.	RW
30	DES8	Descriptor 8 Word. 0: 4-word descriptor; 1: 8-word descriptor.	RW
29:24	Reserved	Write has no effect, read as zero.	R
23:16	CDOA	Copy of offset address of last completed descriptor from that in DMA command register. Software could know which descriptor is just completed combining with count terminate interrupt resulted by DCSn.CT. (Ignored in No-Descriptor Transfer)	RW
15:8	Reserved	Write has no effect, read as zero.	R
7	BERR	BCH error. 0: no BCH error; 1: BCH error within this transfer. (Only channel 0 has this bit for BCH transfer)	RW
6	INV	Descriptor Invalid error. 0: no invalid error; 1: descriptor invalid, DCMn.V bit is loaded as 0. (Ignored in No-Descriptor Transfer)	RW
5	Reserved	Write has no effect, read as zero.	R
4	AR	Address Error. 0: no address error; 1: address error.	RW
3	TT	Transfer Terminate. 0: No-Link Descriptor or No-Descriptor DMA transfer does not end 1: No-Link Descriptor or No-Descriptor DMA transfer end	RW
2	HLT	DMA halt. 0: DMA transfer is in progress; 1; DMA halt.	RW
1	CT	Count Terminate. 0: Link DMA transfer does not end; 1: Link DMA transfer end. (Ignored in No-Descriptor Transfer)	RW
0	CTE	Channel transfer enable. 0: disable; 1: enable.	RW

5.2.6 DMA Channel Command (DCMn, n = 0 ~ 5)

DCM0, DCM1, DCM2, DCM3, DCM4, DCM5

0x13020014, 0x13020034, 0x13020054, 0x13020074, 0x13020094, 0x130200b4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EACKS	EACKM	ERDM	Reserved			BLAST	Reserved	SAI	DAI	Reserved		RDIL				SP		DP		Reserved		TSZ		Reserved		STDE	V	VM	VIE	TIE	LINK
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31	EACKS	External DACK Output Level Select. 0: active high; 1: active low. (Only channel 2 support external transfer)	RW
30	EACKM	External DACK Output Mode Select. 0: output in read cycle; 1: output in write cycle. (Only channel 2 support external transfer)	RW
29:28	ERDM	External DREQ Detection Mode Select. 00: Low level detection 01: Falling edge detection 10: High level detection 11:Rising edge detection (Only channel 2 support external transfer)	RW
27:26	Reserved	Write has no effect, read as zero.	R
25	BLAST	BCH/NAND last. 0: non-last data block for BCH/NAND; 1: last data block for BCH/NAND. (Only channel 0 support BCH transfer; all channel support Nand transfer, when it is used for nand, it means the last data block transfer for one nand dma request detection)	RW
24	Reserved	Write has no effect, read as zero.	R
23	SAI	Source Address Increment. 0: no increment; 1: increment.	RW
22	DAI	Target Address Increment. 0: no increment; 1: increment.	RW
19:16	RDIL	Request Detection Interval Length. Set the number of transfer unit between two requests detection in single mode. Please refer to following Table 5-3.	RW
15:14	SP	Source port width. 00: 32-bit; 01: 8-bit; 10: 16-bit; 11: reserved.	RW
13:12	DP	Target port width. 00: 32-bit; 01: 8-bit; 10: 16-bit; 11: reserved. (**NOTE:** for bch transfer encoding, DP only can be 32-bit or 8-bit; for bch decoding, DP only can be 32-bit)	RW
11	Reserved	Write has no effect, read as zero.	R
10:8	TSZ	Transfer Data Size of a data unit. 000: 32-bit; 001: 8-bit; 010: 16-bit; 011: 16-byte; 100: 32-byte; others: reserved.	RW
7:6	Reserved	Write has no effect, read as zero.	R
5	STDE	Stride Disable/Enable. 0: address stride disable; 1: address stride enable.	RW
4	V	Descriptor Valid flag. 0: Descriptor Invalid; 1: Descriptor Valid for transfer.	R

		(Ignored in No-Descriptor Transfer and in BCH decoding transfer and in Descriptor Transfer with VM=0)	
3	VM	Descriptor Valid Mode. 0: V bit is ignored; 1: Support V bit. (Ignored in No-Descriptor and in BCH decoding transfer)	RW
2	VIE	DMA Valid Error Interrupt Enable. 0: disable; 1: enable. (Ignored in No-Descriptor Transfer)	RW
1	TIE	Transfer Interrupt Enable (TIE). 0: disable interrupt; 1: enable interrupt when TT is set to 1.	RW
0	LINK	Descriptor Link Enable. 0: disable; 1: enable. (Ignored in No-Descriptor Transfer)	RW

Table 5-3 Detection Interval Length

RDIL	Description
0	Interval length is 0
1	Interval length is 2 transfer unit
2	Interval length is 4 transfer unit
3	Interval length is 8 transfer unit
4	Interval length is 12 transfer unit
5	Interval length is 16 transfer unit
6	Interval length is 20 transfer unit
7	Interval length is 24 transfer unit
8	Interval length is 28 transfer unit
9	Interval length is 32 transfer unit
10	Interval length is 48 transfer unit
11	Interval length is 60 transfer unit
12	Interval length is 64 transfer unit
13	Interval length is 124 transfer unit
14	Interval length is 128 transfer unit
15	Interval length is 200 transfer unit

5.2.7 DMA Descriptor Address (DDAn, n = 0 ~ 5)

This register is ignored in No-Descriptor Transfer.

DDA0, DDA1, DDA2,
DDA3, DDA4, DDA5

0x13020018, 0x13020038, 0x13020058,
0x13020078, 0x13020098, 0x130200b8

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

Bits	Name	Description	RW
31:12	DBA	Descriptor Base Address.	RW
11:4	DOA	Descriptor Offset Address.	RW
3:0	Reserved	Write has no effect, read as zero.	R

5.2.8 DMA Stride Address (DSDn, n = 0 ~ 5)

This register is ignored in No-Descriptor Transfer.

When address stride transfer is enabled in Descriptor mode, after a sub-block defined in DTCRn is finished transferring, the source or target stride address will be added up to the corresponding source or target address and the transfer will keep going until the transfer ends which means TC in DTCRn reach 0.

DSD0, DSD1, DSD2,
DSD3, DSD4, DSD5

0x130200C0, 0x130200C4, 0x130200C8,
0x130200CC, 0x130200D0, 0x130200D4

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

Bits	Name	Description	RW
31:16	TSD	Target Stride Address.	RW
15:0	SSD	Source Stride Address.	RW

5.2.9 DMA Control

DMAC1 controls channel 0~5 .

DMAC1 **0x13020300**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FMSC	FSSI	FTSSI	FUART	FAIC	Reserved																	PM		Reserved				HLT	AR	Reserved	DMAE
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31	FMSC	MSC Fast DMA mode. 0: normal DMA transfer; 1: fast DMA transfer.	RW
30	FSSI	SSI Fast DMA mode. 0: normal DMA transfer; 1: fast DMA transfer.	RW
29	FTSSI	TSSI Fast DMA mode. 0: normal DMA transfer; 1: fast DMA transfer.	RW
28	FUART	UART Fast DMA mode. 0: normal DMA transfer; 1: fast DMA transfer.	RW
27	FAIC	AIC Fast DMA mode. 0: normal DMA transfer; 1: fast DMA transfer.	RW
26:10	Reserved	Write has no effect, read as zero.	R
9:8	PM	Channel priority mode. 00: CH0, CH1 > CH2, CH3, CH4, CH5 01: CH1, CH2 > CH0, CH3, CH4, CH5 10: CH2, CH3 > CH0, CH1, CH4, CH5 11: CH3, CH4 > CH0, CH1, CH2, CH5 For example, when PM == 2'b00, it means set1 includes ch0 and ch1 and set2 includes ch2~ch5, set 1 has the higher priority than set 2, within one set, channel priority is round robin, that is: ch0→ch1→ch2→ch0→ch1→ch3→ch0→ch1→ch4→ch0→ch1→ch5.	RW
7:4	Reserve	Write has no effect, read as zero.	R
3	HLT	Global halt status, halt occurs in any channel, the bit should set to 1. 0: no halt 1: halt occurred	RW
2	AR	Global address error status, address error occurs in any channel, the bit should be set to 1. 0: no address error 1: address error occurred	RW
1	Reserved	Write has no effect, read as zero.	R
0	DMAE	Global DMA transfer enable.	RW

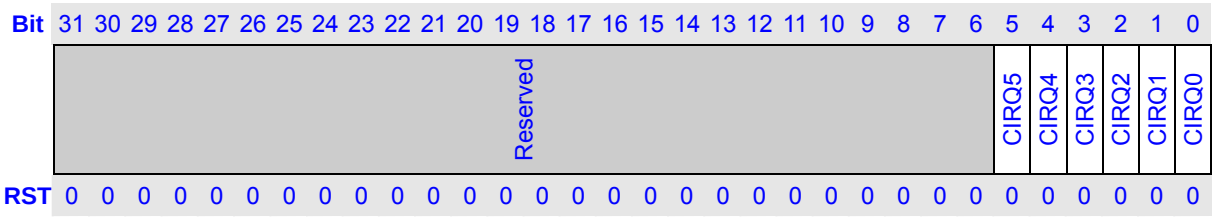
		0: disable DMA channel transfer 1: enable DMA channel transfer	

NOTE: FMSC/FSSI/FTSSI/FUART/FAIC bit either in DMAC1 is set, the corresponding dma transfer for MSC(MSC1), SSI(SSI1), UART0~3, AIC is in fast dma mode.

5.2.10 DMA Interrupt Pending (DIRQP)

DMAC supports total 6 pending interrupt.

DIRQP **0x13020304**

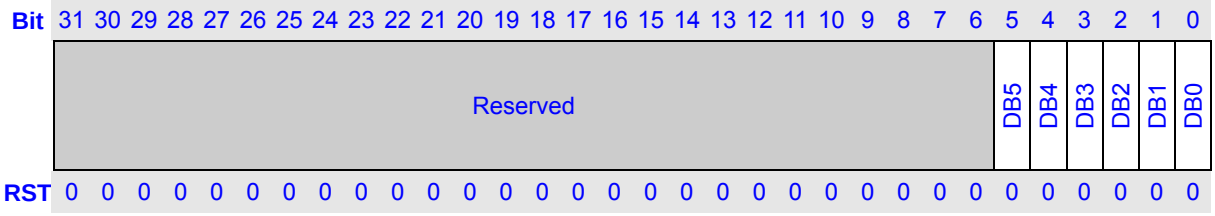


Bits	Name	Description	RW
31:6	Reserved	Write has no effect, read as zero.	R
5:0	CIRQn	CIRQn (n=0~5) denotes pending status for corresponding channel. 0: no abnormal situation or normal DMA transfer is in progress 1: abnormal situation occurred or normal DMA transfer done	RW

5.2.11 DMA Doorbell (DDR)

DDR supports channel 0~5 .

DDR **0x13020308**



Bits	Name	Description	RW
31:8	Reserved	Write has no effect, read as zero.	R
7:0	DBn	DMA Doorbell for each channel, n=0~5, for example DB0 is for DMA channel 0. Software set it to 1 and hardware clears it to 0. 0: disable DMA controller to fetch the first descriptor or DMA controller clears it to 0 as soon as it starts to fetch the descriptor	R

		1: Write 1 to DDS will set the corresponding DBn bit to 1 and enable DMA controller to fetch the first descriptor For example, write 0x00000001 to DDS, DB0 bit is set to 1 and enable DMA channel 0 to fetch the first descriptor. Write 0 to DDS, no meaning.	

5.2.12 DMA Doorbell Set (DDRS)

DDRS supports channel 0~5.

DDRS **0x1302030c**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																										DBS5	DBS4	DBS3	DBS2	DBS1	DBS0
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:8	Reserved	Write has no effect, read as zero.	R
7:0	DBSn	DMA Doorbell Set for each channel. 0: ignore 1: Set the corresponding DBn bit to 1	W

5.2.13 DMA Clock Enable (DCKE)

DCKE supports channel 0~5 .

DCKE **0x13020310**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																										DCKE5	DCKE4	DCKE3	DCKE2	DCKE1	DCKE0
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:8	Reserved	Write has no effect, read as zero.	R
7:0	DCKEn	DMA Clock Enable for each channel. 0: ignore 1: Set the corresponding DCKEn bit to 1	W

5.3 DMA manipulation

5.3.1 Descriptor Transfer

5.3.1.1 Normal Transfer

To do proper Descriptor DMA transfer, do as following steps:

1 First of all, open channel clock by setting DCKEn register for corresponding channel.

2 Check whether the status of DMA controller are available, that is, for global control (DMAC), ensure that DMAC.AR=0 and DMAC.HLT=0; while for expected channels, ensure that DCSn.AR=0, DCSn.HLT=0, DCSn.TT=0, DTCn=0 and DCSn.INV=0.

3 Select 4 word or 8 word descriptor by DCSn.DES8.

4 For Descriptor transfer, guarantee DCSn.NDES=0.

5 Initiate channel request register DRSRn.

6 Build descriptor in memory. Write the first descriptor address in DDAn and the address must be 16Bytes aligned in 4word descriptor and 32Bytes aligned in 8word descriptor. The descriptor address includes two parts: Base and Offset address. If the descriptor is linked, the 32-bit address of next descriptor is composed of 20-bit Base address in DDAn and 8-bit Offset address in DES3.DOA and the four LSB is 0x0. See Table 5-4 for the detailed 4-word descriptor structure.

NOTE: if stride address transfer is enabled, the address must be 32Bytes aligned because DES4 needs to read out.

7 Set 1 to the corresponding bit in DDR to initiate descriptor fetch.

8 Set DMAC.DMAE=1 and expected DCSn.CTE=1 to launch DAM transfer.

9 Hardware clears the corresponding bit in DDR as soon as it starts to fetch the descriptor.

10 If DES0.V =0 and DES0.VM=1, DMAC stops and set DCSn.INV=1. Otherwise, it waits for dma request from peripherals to start dma transfer.

11 After DMAC completes the current descriptor dma transfer, if DES0.VM=1, it clears DES0.V to 0 and writes back to memory. If DES0.Link=1, it sets DCSn.CT to 1, otherwise it sets DCSn.TT to 1. If the interrupt enabled, it will generates the corresponding interrupts.

12 If DES0.LINK=1, after DMAC completes the current descriptor dma transfer and return to fetch the next descriptor and continues dma transfer until completes the descriptor dma transfer which DES0.LINK=0.

13 When transfer end, clr DCSn.CTE to 0 to close the channel, and then clear DCSn.TT and DCSn.CT bits.

Table 5-4 Descriptor Structure

Word	Bit	Name	Function
1st (DES0)	31	EACKS	External DMA DACKn output polarity select
	30	EACKM	External DMA DACKn output Mode select
	29-28	ERDM	External DMA request detection Mode
	27	EOPM	External DMA End of process mode
	26	Reserved	
	25	BLAST	BCH Last (Only for BCH and Nand transfer)
	24	Reserved	
	23	SAI	Source Address Increment
	22	DAI	Target Address Increment
	21-20	Reserved	
	19-16	RDIL	Request Detection Interval Length
	15-14	SP	Source port width
	13-12	DP	Target port width
	11	Reserved	
	10-8	TSZ	Transfer Data Size
	7	TM	Transfer Mode
	6	Reserved	
	5	STDE	Stride transfer enable
	4	V	Descriptor Valid
	3	VM	Descriptor Valid Mode
	2	VIE	Descriptor Invalid Interrupt Enable
	1	TIE	Transfer Interrupt Enable
	0	LINK	Descriptor Link Enable
2nd (DES1)	31-0	DSA	Source Address
3rd (DES2)	31-0	DTA	Target Address
4th (DES3)	31-24	DOA	Descriptor Offset address
	23-0	DTC	Transfer Counter
5th (DES4)	31-16	TSD	Target Stride Address
	15-0	SSD	Source Stride Address
6th(DES5)	5-0	DRT	DMA Request Type
	31-6	Reserved	
7th(DES6)	31-0	Reserved	
8th(DES7)	31-0	Reserved	

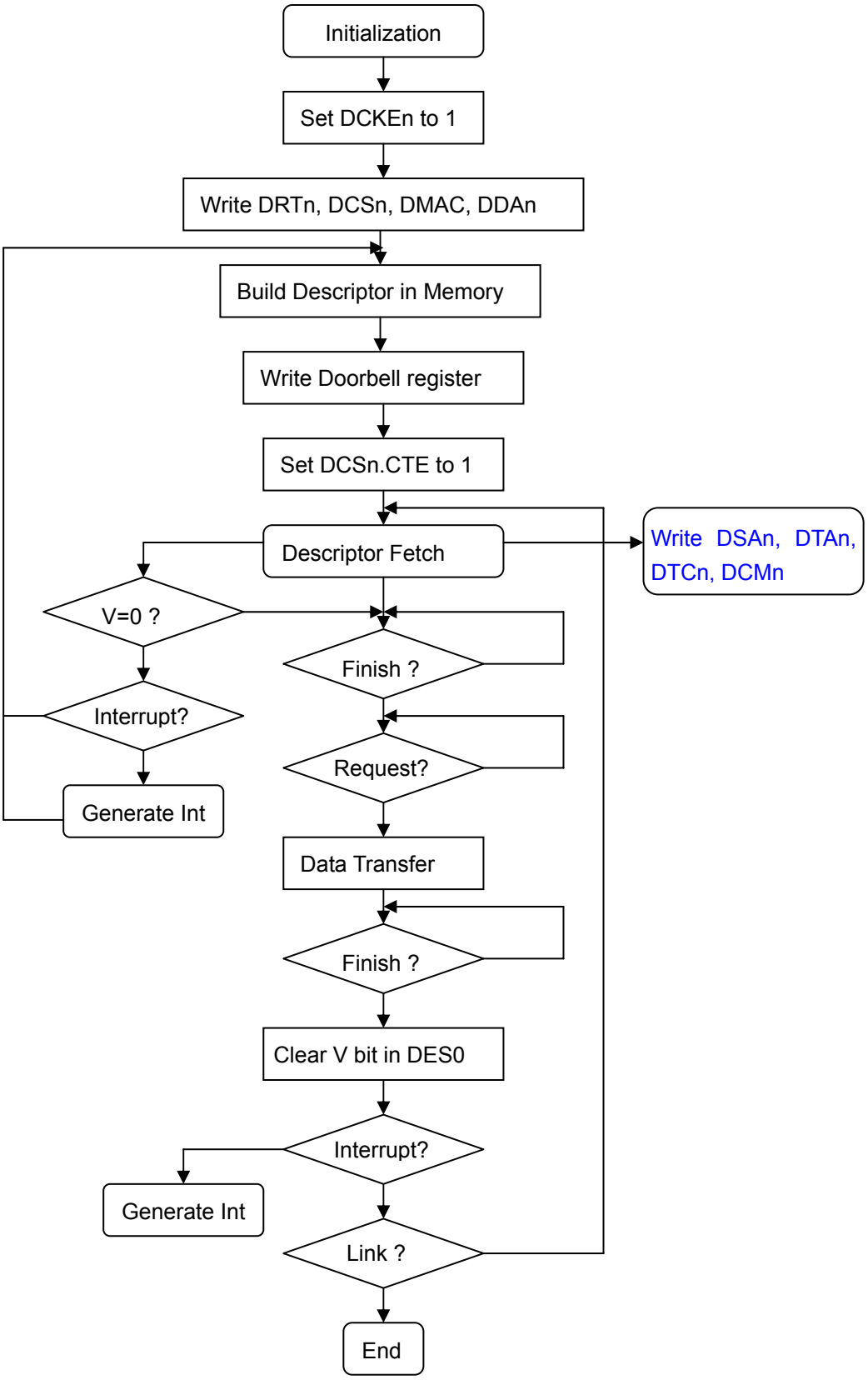


Figure 5-1 Descriptor Transfer Flow

5.3.1.2 Stride Address Transfer

During transfer, source or target address can be not continuous and the source and target stride offset address are showed in DSDn registers.

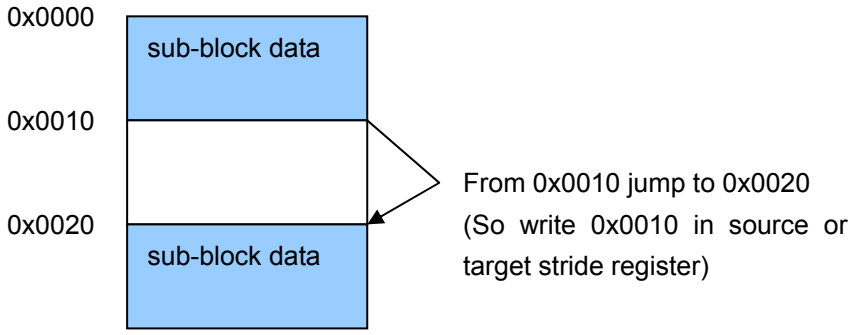


Figure 5-2 Example for Stride Address Transfer

5.3.1.3 BCH DMA Transfer

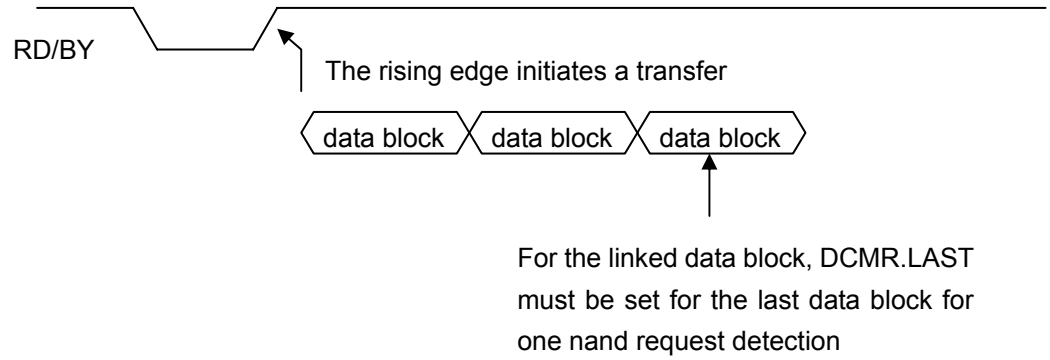
Channel 0 supports BCH DMA transfer.

During BCH encoding, DMA read data from memory pointed by DSAR0 and write to BCH data register BHDR, after BCH encoding finishes, DMA write BHINT and BCH parity data BHPAR0~3 (8-bit BCH) or BHPAR0~1 (4-bit BCH) respectively to memory pointed by DTAR0, and then DMA clear BHINT and set BCH reset to BCH automatically.

During BCH decoding, DMA read data from memory pointed by DSAR0 and write to BCH data register BHDR, after BCH decoding finishes, if there is error in the data block, DMA will write BHINT, BHERR0~3 (8-bit BCH) or BHERR0~1 (4-bit BCH) to memory pointed by DTAR0 or if there is no error in the data block, DMA will only write BHINT to memory, and then DMA clear BHINT and set BCH reset to BCH. If multiple data block are linked to wait for BCH decoding, data transfer and decoding can be executed in pipeline, that is when the first data block is being decoding, and second data can be transfer to BCH for syndrome generation.

Here one data block means, for encoding, the entire data bytes need encoding, for decoding, the entire data bytes and parity bytes need decoding. DCM.BLAST must be used in descriptor BCH transfer. When one data block is in a continuous memory space, BLAST must be set to 1 for this data block; when one data block is linked in multiple data space, BLAST must be set to 1 for the last data space.

5.3.1.4 Nand Transfer



5.3.2 No-Descriptor Transfer

To do proper DMA transfer, do as following steps:

1. First of all, check whether the status of DMA controller are available, that is, for global control (DMAC), ensure that DMAC.AR=0 and DMAC.HLT=0; while for expected channels, ensure that DCSn.AR=0, DCSn.HLT=0 and DCSn.TT=0 and DTCn=0.
2. For each channel n, initialize DSAn, DTAn, DTCn, DRTn, DCSn, DCMn properly.
3. Set DMAC.DMAE=1 and expected DCSn.CTE=1 and DCSn.NDES=1 to launch DAM transfer.

For a channel with auto-request (DRTn.RT=0x8), the transfer begins automatically when the DCSn.CTE bit and DMAC.DMAE bit are set to 1. While for a channel with other request types, the transfer does not start until a transfer request is issued and detected.

For any channel n, The DTCn value is decremented by 1 for each successful transaction of a data unit. When the specified number of transfer data unit has been completed (DTCn = 0), the transfer ends normally. Meanwhile corresponding bit of DIRQP is set to 1. If DCMn.TIE bit is set to 1, an interrupt request is sent to the CPU. However, during the transfer, if a DMA address error occurs, the transfer is suspended, both DCSn.AR and DMAC.AR are set to 1 as well as corresponding bit of DIRQP. Then an interrupt request is sent to the CPU despite of DCMn.TIE.

Sometimes, for example, an UART parity error occurs for a channel that is transferring data between such UART and another terminal. In the case, both DCSn.HLT and DMAC.HLT are set to 1 and the transfer is suspended. Software should identify halt status by checking such two bits and re-configure DMA to let DMA rerun properly later.

For non-descriptor BCH transfer, there is no pipeline execution for BCH decoding. DCM.BLAST doesn't need to be set in non-descriptor BCH transfer.

5.4 DMA Requests

DMA transfer requests are normally generated from either the data transfer source or target, but also they can be issued by on-chip peripherals that are neither the source nor the target. There are two DMA transfer request types: auto-request, and on-chip peripheral request. For any channel n, its transfer request type is determined through DRTn.

5.4.1 Auto Request

When there is no explicit transfer request signal available, for example, memory-to-memory transfer or memory to some on-chip peripherals like GPIO, the auto-request mode allows the DMA to automatically generate a transfer request signal internally. Therefore, when DMA initialization done, once the DMAC.DMAE and DCSn.CTE are set to 1, the transfer begins immediately in channel n which DRTn=0x8.

5.4.2 On-Chip Peripheral Request

In the mode, transfer request signals come from on-chip peripherals. All request types except 0x8 (value of DRT) belong to the mode. Note: the transfer byte number for one request detection according to DCMn.RDIL must be equal or less than the byte number according to receive or transmit trigger value of source or target devices.

5.5 Channel Priorities

There are two dma cores, each one supports 6 channels dma transfer. The two cores have the same priority.
In each core, there are two sets: set 1 has the higher priority than set 2, within each set priority is round robin.

Table 5-5 Relationship among DMA Transfer connection, request Mode & transfer Mode

Transfer Connection	Request Mode	Transfer Mode	Data Size (bits)	Channel
External memory or memory-mapped external device and on-chip peripheral module	Auto on-chip	Single	8/16/32 16-byte/32-byte	0~5

5.6 Examples

5.6.1 Memory-to-memory auto request No-Descriptor Transfer

Suppose you want to do memory move between two different memory regions through channel 3, for example, moving 1KB data from address 0x20001000 to 0x20011000, do as following steps:

1 Check if (DMAC.AR==0 && DMAC.HLT==0 && DCS3.AR==0 && DCS3.HLT==0 && DCS3.CT==0 && DCS3.NDES=1 && DTC3==0).
2 If above condition is true, set value 0 to DCS3.CTE to disable the channel 3 temporarily.
3 Set source address 0x20001000 to DSA3 and target address 0x20011000 to DTA3.
4 Suppose the data unit is word, set transfer count number 256 (1024/4) to DTC3.
5 Set auto-request (0x8) to DRT3.
6 Up to now, only the most important channel control register DCM3 is left, set it carefully:
   - Set value 1 to SAI and DAI[*1].
   - Ignore RDIL because in the case there is no explicit request signal can be detected.
   - Set word size (0) to SP and DP[*2].
   - Set value 1 to TIE to let CPU do some post process after the transfer done.
7 Set value 1 to DCS3.CTE and DMAC.DMAE to launch the transfer in channels 3.
8 When the transfer terminates normally (DTC3==0 && DCS3.TT==1), DIRQP.CIRQ3 will automatically be set value 1 and an interrupt request will be sent to CPU.
9 When CPU grants the interrupt request, in the corresponding IRQ handler, software must clear the DCS3.CT to value 0, and the behavior will automatically clear DIRQP.CIRQ3.

NOTES:

1 [*1]: Either source or target is a FIFO, must not enable corresponding address increment.
2 [*2]: When either source or target need be accessed through EMC (external memory controller), the real port with of the device is encapsulated by EMC, so you can set any favorite port with for it despite of the real one.

6 AHB Bus Arbiter

6.1 Overview

AHB bus arbiter is responsible for AHB bus transactions' arbitrating to provide a fair chance for each AHB master to possess the AHB bus. The refined arbiter in this processor adopts a new arbitrating technique to fulfill the back-to-back feature of AHB protocol. Moreover, dividing two master groups with different privileges supports two arbitrating methods:

1 Round-robin possession for masters in the same group
2 Preemptive possession for masters with higher privileges

There are two AHB buses in this processor, AHB0 and AHB1. AHB0 is responsible for interconnecting 8 main AHB masters including main CPU core, LCD, IPU, CIM, DMA, USB, the bridge for AHB1 and the bridge for MC. While the AHB1 is responsible for interconnecting 8 AHB masters belonging to video processor including auxiliary CPU core, MC, ME, IDCT, DBLK, DDMA0 for TCSM0, DDMA1 for TCSM1 and the bridge for main CPU core.

6.2 Register Descriptions

The base address for memory-mapped registers in the AHB0 bus' arbiter is 0x13000000. The AHB1 bus' arbiter has the same base address as the AHB0, but its memory-mapped registers can only be accessed by auxiliary CPU core.

Table 6-1 AHB Bus Arbiter Registers List

Register Name	Offset	Size	R/W	Reset Value	Description
RPIOR	0x00	32	R/W	0x00000000	Master group priority order
CTRL	0x04	32	R/W	0x00000000	AHB monitor control *1
CLKL	0x08	32	R/W	0x00000000	AHB clock counter low *1
EVENT0L	0x0C	32	R/W	0x00000000	AHB bus event 0 counter low *1
EVENT1L	0x10	32	R/W	0x00000000	AHB bus event 1 counter low *1
EVENTH	0x14	32	R/W	0x00000000	AHB bus event & clock counter high *1

NOTES:

1 *1 denotes the register is not implemented in AHB1.

6.2.1 Priority Order Register

HARB_PRIOR offset 0

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

Bits	Name	Description	R/W
31:2	Reserved	Write is ignored, read as zero.	R
1:0	PRI	Priority order for AHB0. (first 3 masters belong to high privilege group) 0: {lcd, ipu, cim}, {dma, localbridge, mcbridge, cpu, usb} 1: {lcd, ipu, cpu}, {dma, cim, localbridge, mcbridge, usb} 2: {cim, ipu, cpu}, {lcd, dma, usb, localbridge, mcbridge} 3: {cim, ipu, dma}, {cpu, lcd, usb, localbridge, mcbridge} Priority order for AHB1. (first 3 masters belong to high privilege group) 0: {dblk, mc, aux}, {me, ddma1, idct, bridge, ddma0} 1: {dblk, mc, bridge}, {me, aux, ddma1, idct, ddma0} 2: {aux, mc, bridge}, {dblk, me, ddma0, ddma1, idct} 3: {aux, mc, me}, {bridge, dblk, ddma0, ddma1, idct}	RW

6.2.2 Monitor Control Register

HARB_MC offset 4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								M1				M0				Reserved	EV1			Reserved	EV0			Reserved					EV1E	EV0E	CLKE
Rst	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	0	0	0

Bits	Name	Description	R/W
31:24	Reserved	Write is ignored, read as zero.	R
23:20	M1	Monitored Master ID in monitor channel 1[*1].	RW
19:16	M0	Monitored Master ID in monitor channel 0[*1].	RW
15	Reserved	Write is ignored, read as zero.	R
14:12	EV1	AHB bus event encoding for monitor channel 1[*2].	RW
11	Reserved	Write is ignored, read as zero.	R
10:8	EV0	AHB bus event encoding for monitor channel 0[*2].	RW
7:3	Reserved	Write is ignored, read as zero.	R
2	EV1E	Enable monitor channel 1. 0: disable; 1: enable.	RW
1	EV0E	Enable monitor channel 0. 0: disable; 1: enable.	RW
0	CLKE	AHB clock counting enable. 0: disable; 1: enable.	RW

NOTES:

1 [*1] denotes the masterID encoding are described in the Table 6-3.

2 [*2] denotes the event encoding are described in the Table 7-2 AHB Bus Monitor Events.

Table 6-2 AHB Bus Monitor Events

Events	Full Name	Comment
0	bus transaction cycles	exclude idle cycles.
1	bus transaction times	total NONSEQ times.
2	grant latency[*3]	total pending request cycles for occurred transactions.
3	critical grant latency trigger[*4]	Once the grant latency for one time of bus transaction exceeds the critical value preset in the counter low register, the associative counter high register will accumulate 1.
4	single beat transaction times	BURST type is SINGLE.
5	fixed length burst transaction times	BURST type is INCR4/8/16 or WRAP4/8/16.
6	INCR bust transaction times	BURST type is INCR.

7	critical transaction cycles trigger[*5]	Once the active transaction cycles for one time of bus transaction exceeds the critical value preset in the counter low register, the associative counter high register will accumulate 1.

NOTES:

1 [*3, *4, *5] denotes that such events are undefined when masterID is ALL.

Table 6-3 AHB0 Master-ID

Masters	Full Name
0	CPU
1	CIM
2	LCD
3	DMA
4	IPU
5	USB
6	LocalBridge
7	MCBridge
8~14	Reserved
15	ALL (events triggered by any master should be monitored)

6.2.3 AHB Clock Counter Low Register

HARB_CLKL offset 8

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

Bits	Name	Description	R/W
31:0	CLKL	Record the low 32 bits of AHB clock counter.	RW

6.2.4 Event0 Low Register

HARB_EVENT0L offset 12

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EVENT0L																															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
31:0	EVENT0L	Record the low 32 bits of event 0 counter.	RW

6.2.5 Event1 Low Register

HARB_EVENT1L offset 16

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EVENT1L																															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
31:0	EVENT1L	Record the low 32 bits of event 1 counter.	RW

6.2.6 Event High Register

HARB_EVENTH offset 20

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CLKH																EVENT1H								EVENT0H							
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
31:16	CLKH	Record the high 16 bits of AHB clock counter.	RW
15:8	EVENT1H	Record the high 8 bits of event 1 counter.	RW
7:0	EVENT0H	Record the high 8 bits of event 0 counter.	RW

Note that fields of EVENTH register will not overflow automatically. For example, when EVENT1H reaches 0xFF during monitoring, it remains the value until software modifies it.

7 Clock Reset and Power Controller

7.1 Overview

The Clock & Power management block consists of three parts: Clock control, PLL control, and Power control, Reset control.

The Clock control logic can generate the required clock signals including CCLK for CPU, HCLK for the AHB bus peripherals, IPU_CLK for the IPU, and PCLK for the APB bus peripherals. The chip has one Phase Locked Loops (PLL): for CCLK, HCLK, IPU_CLK and PCLK, MSCLK, SSICLK, LPCLK. The clock control logic can make slow clocks without PLL and connect/disconnect the clock to each peripheral block by software, which will reduce the power consumption.

For the power control logic, there are various power management schemes to keep optimal power consumption for a given task. The power management block can activate four modes: NORMAL mode , DOZE mode, IDLE mode, SLEEP mode.

For reset control logic, the hardware reset and hibernate reset is extended to more 40ms . It controls or distributes all of the system reset signals.

7.2 Clock Generation UNIT

The clock generation unit (CGU) contains one PLL driven by an external oscillator and the clock generation circuit from which the following clocks are derived:

Signal	Description
CCLK	Fast clock for internal operations such as executing instructions from the cache. It can be gated during doze and idle mode when all the criteria to enter a low power are met.
HCLK	System clock—This signal appears as the HCLK input to the CPU and the HCLK to the system. This is a continuous clock (when the system is not in sleep mode) It can be gated during Sleep mode when all the criteria to enter a low power are met.
IPU_CLK	IPU clock for IPU module.
PCLK	Peripheral clock – APB BUS device clock.
MCLK	Clock for EMC controller.
CKO	SDRAM Clock.
LPCLK	LCD pixel clock.
CIM_MCLK	Clock output from CIM module.
CIM_PCLK	Clock input to CIM module.
I2SCLK	I2S codec clock.
BITCLK	AC97 bit clock.
MSC0CLK	MSC0 clock.
MSC1CLK	MSC1 clock.
SSICLK	SSI0 clock.
EXCLK	12M clock output for UART I2C TCU USB2.0-PHY AUDIO CODEC.

Feature:

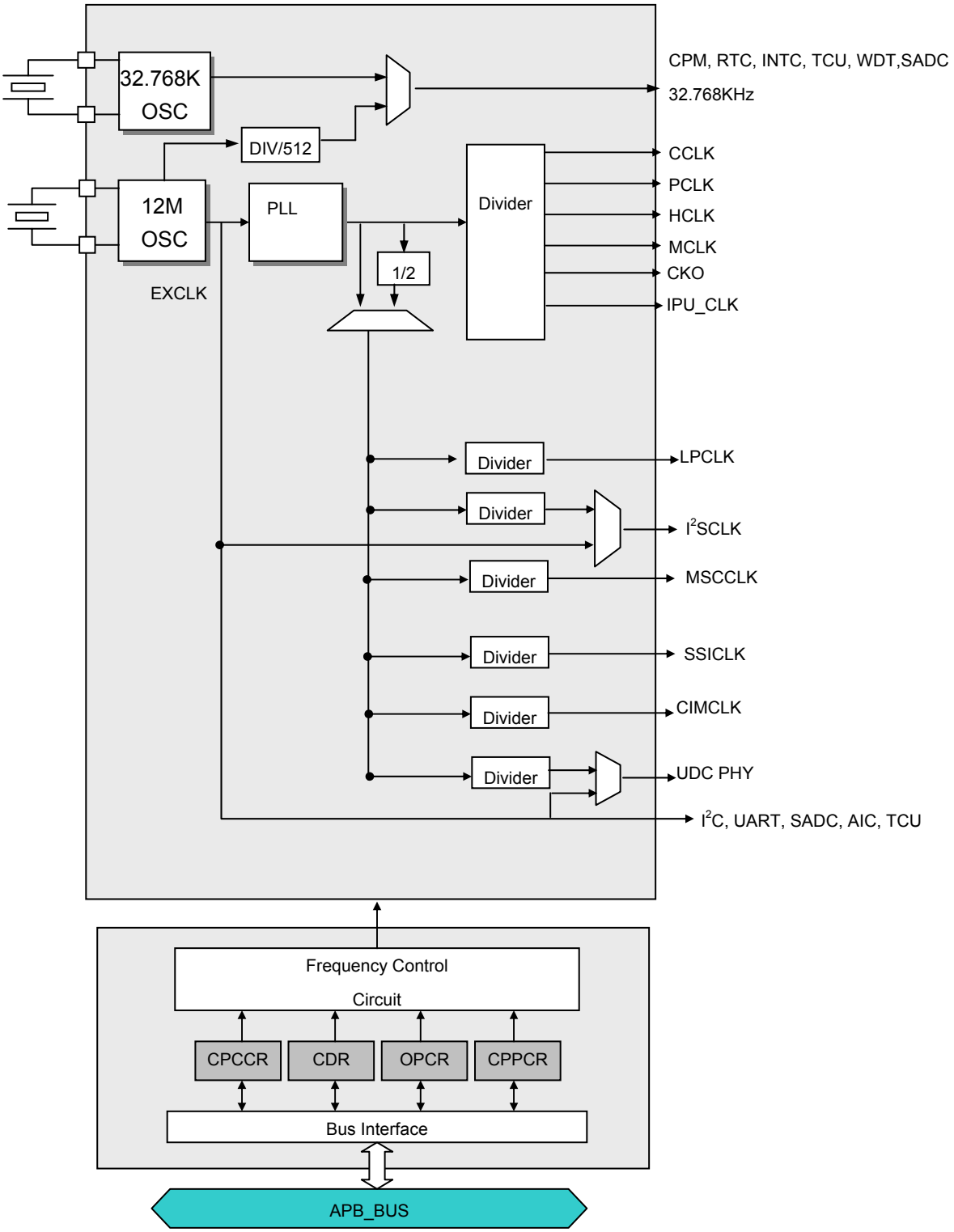
- On-chip 2MHz~27MHZ oscillator circuit
- On-chip 32.768KHZ oscillator circuit
- One On-chip phase-locked loops (PLL) with programmable multiplier
- CCLK, PCLK, HCLK, MCLK , CKO and LPCLK, I2SCLK, MSC0CLK , MSC1CLK ,SSICLK frequency can be changed separately for software by setting registers
- SSI clock supports 50M clock
- MSC clock supports 50M clock
- Functional-unit clock gating

7.2.1 Pin Description

Name	I/O	Description
RTCLK_XI	Input	32.768KHZ Oscillator input signal
RTCLK_XO	Output	32.768KHZ Oscillator output signal
EXCLK	Input	Oscillator input signal
EXCLKO	Output	Oscillator output signal
CIM_MCLK	Output	Clock output from CIM module signal
CIM_PCLK	Input	Clock input to CIM module signal
LPCLK	Output	LCD pix clock signal
CKO	Output	SDRAM clock signal
BITCLK	Inout	I2S/AC97 bit clock
MSC0_CLK	Output	Clock output For MMC0/SD0 Card signal
MSC1_CLK	Output	Clock output For MMC1/SD1 Card signal
SSI_CLK	Output	Clock output from SSI module signal

7.2.2 CGU Block Diagram

Following figure illustrates a block diagram of CGU.



7.2.3 Clock Overview

There is an internal PLL . PLL input clock is an external input clock EXCLK. Theoretically, EXCLK can be 2MHz ~ 27MHz.

CCLK is CPU clock. It is usually the fastest clock in the chip. This clock represents the chip speed.

HCLK is for on chip high speed peripherals connected to AHB bus.

PCLK is for on chip slow speed peripherals connected to APB bus.

MCLK is external memory bus clock. MCLK represents the SDRAM speed.

CCLK, HCLK, PCLK and MCLK are synchronous clocks that may have different frequencies. They are from the same clock source, the on chip PLL output clock in most cases. HCLK frequency can be equal to CCLK or divided CCLK by an integer. PCLK frequency can be equal to HCLK or divided HCLK by an integer. MCLK frequency can be equal to or half of HCLK.

AC97 in AIC module needs a 12.288MHz BIT clock. It is input from the external AC97 CODEC chip or other clock source.

Besides PLL input, EXCLK also provides device clock or one of device clocks for many peripherals, such as, UART, I2C, TCU, SSI, SADC and WDT.

Device clock of MSC (MMC/SD) is taken from software divided PLL output clock.
Device clock of SSI is taken from software divided PLL output clock.

LCD's pixel clock are generated from PLL output clock, which are divided by two independent dividers.

The slowest clock is RTCLK, which is usually 12M/512 or 32768Hz.

7.2.4 CGU Registers

All CGU register 32bit access address is physical address.

Table 7-1 CGU Registers Configuration

Name	description	RW	Reset Value	Address	Access Size
CPCCR	Clock Control Register	RW	0x42040000	0x10000000	32
CPPCR	PLL Control Register	RW	0x28080011	0x10000010	32
CPPSR	PLL switch and status register	RW	0x80000000	0x10000014	32
I2SCDR	I2S device clock divider Register	RW	0x00000004	0x10000060	32
LPCDR	LCD pix clock divider Register	RW	0x00000004	0x10000064	32
MSCCDR	MSC clock divider Register	RW	0x00000000	0x10000068	32
SSICDR	SSI clock divider Register	RW	0x00000000	0x10000074	32
CIMCDR	CIM MCLK clock divider Register		0x00000004	0x1000007C	32

7.2.4.1 Clock Control Register

The Clock Control Register (CPCCR) is a 32-bit read/write register, which controls CCLK, HCLK, PCLK, MCLK and LDCLK division ratios. It is initialized to 0x42000000 by any reset. Only word access can be used on CPCCR.

CPCCR **0x10000000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	I2CS	Reserved	UCS	UDIV						CE	PCS	Reserved	IPU_DIV				MDIV				PDIV				HDIV				CDIV			
RST	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31	I2CS	I2S Clock Source Selection. Selects the I2S clock source between PLL output and pin EXCLK. 0: I2S clock source is EXCLK 1: I2S clock source is PLL output divided by I2SDIV If EXCLK is 12M, please don't change the bit.	RW
30	Reserved		RW
29	UCS	UDC PHY Clock Source Selection. Selects the UDC PHY clock source between PLL output and pin EXCLK. 0: UDC clock source is pin EXCLK 1: UDC clock source is PLL output If EXCLK is 12M, please don't change the bit.	RW

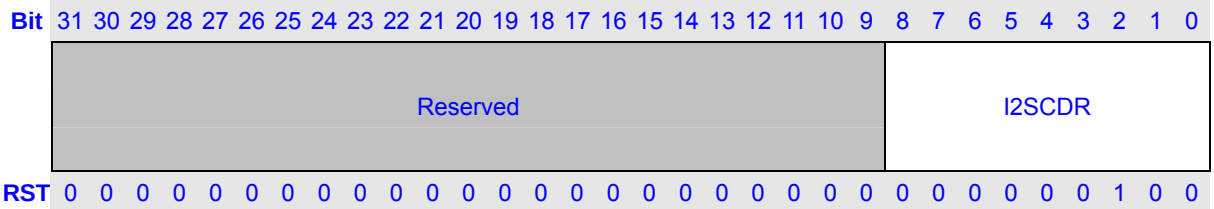
<table>
<tr><td>28:23</td><td>UDIV</td><td>Divider for UDC PHY Clock Frequency. When UDC PHY clock source is PLL (UCS bit is 1), this field specified the UDC PHY clock division ratio, which varies from 1 to 64 (division ratio = UDIV + 1).</td><td>RW</td></tr>
<tr><td>22</td><td>CE</td><td>change enable. If CE is 1, writes on CDIV, HDIV, IPU_DIV, PDIV, MDIV, UDIV, PXDIV or LDIV will start a frequency changing sequence immediately. When CE is 0, writes on CDIV, HDIV, IPU_DIV, PDIV, MDIV, UDIV, PXDIV and LDIV will not start a frequency changing sequence immediately. The division ratio is actually updated in PLL multiple ratio changing sequence or PLL Disable Sequence.
0: Division ratios are updated in PLL multiple ratio changing sequence or PLL Disable Sequence
1: Division ratios are updated immediately</td><td>RW</td></tr>
<tr><td>21</td><td>PCS</td><td>PLL out clock source clock selection. It supplies source clock for MSC I2S LCD UDC SSI.
0: divider clock source is PLL output divided by 2
1: divider clock source is PLL output
Software should set the bit according to various needs.</td><td>RW</td></tr>
<tr><td>20</td><td>Reserved</td><td></td><td>R</td></tr>
<tr><td>19:16</td><td>IPU_DIV</td><td>Divider for IPU Clock Frequency. Specified the IPU_CLK division ratio.
<table>
<tr><td colspan="4">Bit 19~16: HDIV</td><td>Description</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>X1</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>X1/2</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>X1/3</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>X1/4</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>X1/6</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>X1/8</td></tr>
<tr><td colspan="4">Other Value</td><td>Reserved</td></tr>
</table></td><td>RW</td></tr>
<tr><td>15:12</td><td>MDIV</td><td>Divider for Memory Clock Frequency. Specified the MCLK division ratio.
<table>
<tr><td colspan="4">Bit 15~12: MDIV</td><td>Description</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>X1</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>X1/2</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>X1/3</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>X1/4</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>X1/6</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>X1/8</td></tr>
<tr><td colspan="4">Other Value</td><td>Reserved</td></tr>
</table></td><td>RW</td></tr>
<tr><td>11:8</td><td>PDIV</td><td>Divider for Peripheral Clock Frequency. Specified the PCLK division ratio.
<table>
<tr><td colspan="4">Bit 11~8: PDIV</td><td>Description</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>X1</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>X1/2</td></tr>
</table></td><td>RW</td></tr>
</table>

			0	0	1	0	X1/3	
			0	0	1	1	X1/4	
			0	1	0	0	X1/6	
			0	1	0	1	X1/8	
			Other Value				Reserved	
7:4	HDIV	Divider for AHB Clock Frequency. Specified the HCLK division ratio.						RW
			Bit 7~4: HDIV				Description	
			0	0	0	0	X1	
			0	0	0	1	X1/2	
			0	0	1	0	X1/3	
			0	0	1	1	X1/4	
			0	1	0	0	X1/6	
			0	1	0	1	X1/8	
			Other Value				Reserved	
3:0	CDIV	Divider for CPU Clock Frequency. Specifies the CCLK division ratio.						RW
			Bit 3~0: HDIV				Description	
			0	0	0	0	X1	
			0	0	0	1	X1/2	
			0	0	1	0	X1/3	
			0	0	1	1	X1/4	
			0	1	0	0	X1/6	
			0	1	0	1	X1/8	
			Other Value				Reserved	

7.2.4.2 I2S device clock divider Register

I2S device clock divider Register (I2SCDR) is a 32-bit read/write register that specifies the divider of I2S device clock . This register is initialized to 0x00000004 only by any reset. Only word access can be used on I2SCDR.

I2SCDR **0x10000060**



Bits	Name	Description	RW
31:9	Reserved	Writes to these bits have no effect and always read as 0.	R
8:0	I2SCDR	Divider for I2S Frequency. Specified the I2S device clock division ratio, which varies from 1 to 512 (division ratio = I2SCDR + 1).	RW

		When EXCLK is 24M, don't care the bit.	

7.2.4.3 LCD pix clock divider Register

LCD pix clock divider Register (LPCDR) is a 32-bit read/write register that specifies the divider of LCD pixel clock (LPCLK). This register is initialized to 0x00000004 only by any reset. Only word access can be used on LPCDR.

LPCDR **0x10000064**

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

Bits	Name	Description	RW
31:11	Reserved	Writes to these bits have no effect and always read as 0.	R
10:0	LPCDR	Divider for Pixel Frequency. Specified the LCD pixel clock (LPCLK) division ratio, which varies from 1 to 2048 (division ratio = LPCDR + 1).	RW

7.2.4.4 MSC device clock divider Register

MSC device clock divider Register (MSCCDR) is a 32-bit read/write register that specifies the divider of MSC device clock . This register is initialized to 0x00000000 only by any reset. Only word access can be used on MSCCDR.

MSCCDR **0x10000068**

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

Bits	Name	Description	RW
31:5	Reserved	Writes to these bits have no effect and always read as 0.	R
4:0	MSCCDR	Divider for MSC Frequency. Specified the MSC device clock division ratio, which varies from 1 to 32 (division ratio = MSCCDR + 1).	RW

7.2.4.5 SSI device clock divider Register

SSI device clock divider Register (SSICDR) is a 32-bit read/write register that specifies the divider of SSI device clock . This register is initialized to 0x00000000 only by any reset. Only word access can

be used on SSICDR.

SSICDR																															0x10000074
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																												SSICDR			
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:4	Reserved	Writes to these bits have no effect and always read as 0.	R
3:0	SSICDR	Divider for SSI Frequency. Specified the SSI device clock division ratio, which varies from 1 to 16 (division ratio = SSICDR + 1).	RW

7.2.4.6 CIM MCLK clock divider Register

CIM mclk clock divider Register (CIMCDR) is a 32-bit read/write register that specifies the divider of CIM mclk clock (CIM_MCLK). This register is initialized to 0x00000004 only by any reset. Only word access can be used on CMCDR.

CIMCDR																															0x1000007C
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																								CIMCDR							
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Bits	Name	Description	RW
31:8	Reserved	Writes to these bits have no effect and always read as 0.	R
7:0	CIMCDR	Divider for CIM MCLK Frequency. Specified the CIM MCLK clock (CIM_MCLK) division ratio, which varies from 1 to 256 (division ratio = CIMCDR + 1).	RW

7.2.4.7 PLL Control Register

The PLL Control Register (CPPCR) is a 32-bit read/write register, which controls PLL multiplier, on/off state and stabilize time. It is initialized to 0x28080011 only by any reset. Only word access can be used on CPPCR.

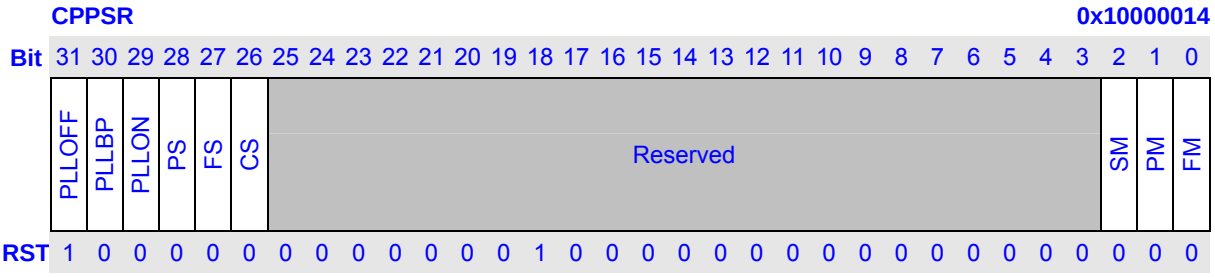
CPPCR **0x10000010**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PLLM									PLLN					PLLOD		Reserved					PLLS	PLLBP	PLLEN	PLLST							
RST	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Bits	Name	Description	RW
31:23	PLLM	the PLL feedback 9-bit divider.	RW
22:18	PLLN	the PLL input 5-bit divider.	RW
17:16	PLLOD	00: divide by 1 01: divide by 2 10: divide by 2 11: divide by 4	RW
15:11	Reserved	Writes to these bits have no effect and always read as 0.	R
10	PLLS	PLL Stabilize Flag. 0: PLL is off or not stable 1: PLL is on and stable	R
9	PLLBP	PLL Bypass. If PLLEN is 1, set this bit to1 will bypass PLL. The PLL is still running background but the source of associated dividers is switched to 12-M. If PLLEN is 0, set this bit to 1 has no effect. If PLLEN is 1, clear this bit to 0 will switch the source of associated dividers to PLL output.	RW
8	PLLEN	PLL Enable. When PLLEN is set to 1, PLL starts to lock phase. After PLL stabilizes, PLLS bit is set. If PLLBP is 0, the source of associated dividers, is switched to PLL output. When PLLEN is clear to 0, PLL is shut off and the source of associated dividers is switched to 12-MHz in spite of PLLBP bit.	RW
7:0	PLLST	PLL Stabilize Time. Specifies the PLL stabilize time by unit of RTCCLK (approximate 32kHz) cycles. It is used when change PLL multiplier or change PLL from off to on. It is initialized to H'11.	RW

7.2.4.8 PLL Switch and Status Register

The PLL Switch and Status Register (CPPSR) is a 32-bit read/write register, which controls the clock switch ,frequency change mode and reflect the PLL and clock switch Status .It is initialized to 0x80000000 by any reset. Only word access can be used on CPPSR.



Bits	Name	Description	RW
31	PLLOFF	0: PLL doesn't enter shut off state 1: PLL is in shut off state	R
30	PLLBP	0: PLL doesn't enter by pass state 1: PLL is in by pass state	R
29	PLLON	0: PLL doesn't enter on state 1: PLL is in on state	R
28	PS	0: disable PLL or no change PLL parameters 1: enable PLL or change PLL parameters have finished. The bit is asserted to 1 auto by hardware . when software concerns this bit, at first software write 0 to the bit, then read the status bit until to 1.	RW
27	FS	Indicate the change frequency has finished . the bit only reflect CDIV, HDIV, MDIV, PDIV change. 0 : no change CDIV, HDIV, MDIV, PDIV 1: change clock parameters have finished when software concerns this bit, at first software write 0 to the bit, then read the status bit until to 1.	RW
26	CS	Indicate the clock switch has finished, the bit reflects when PLL switch to EXCLK or EXCLK to PLL. 0: no clock switch 1: clock switch has finished when software concerns this bit, at first software write 0 to the bit, then read the status bit until to 1.	RW
25:3	Reserved		R
2	SM	When cdiv hdiv mdiv pdiv change, whether cclk hclk ipu_clk mclk pclk are all stopped. 0: hardware control 1: when frequency changes, above clocks are all stopped	RW
1	PM	Clock switch mode. When PLL switch to EXCLK or EXCLK switch to PLL. 0: slow mode	RW

		1: fast mode	
0	FM	Clock frequency change mode. Only to CDIV MDIV HDIV PDIV. 0: slow mode 1: fast mode	RW

7.2.5 PLL Operation

The PLL developed as a macro cell for clock generator. It can generate a stable high-speed clock from a slower clock signal. The output frequency is adjustable and can be up to 500MHz. The PLL integrates a phase frequency detector (PFD), a low pass filter (LPF), a voltage controlled oscillator (VCO) and other associated support circuitry. All fundamental building blocks as well as fully programmable dividers are integrated on the core. It is useful for clock multiplication of stable crystal oscillator sources and for de-skew clock signals.

The PLL block diagram is shown in following figure:

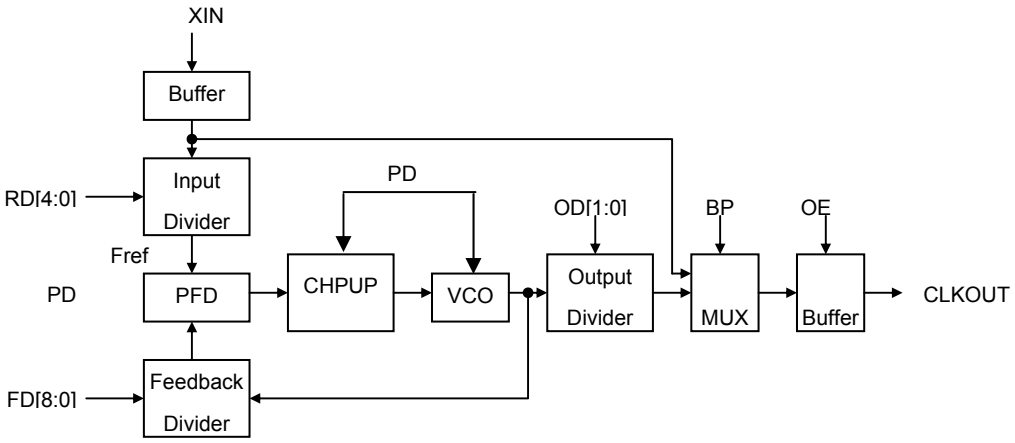


Figure 7-1 Block Diagram of PLL

7.2.5.1 PLL Configuration

PLL Divider Value Setting

There are 3 divider values (N, M and NO) to set the PLL output clock frequency CLKOUT:

1 Input Divider Value N.

N = PLLN of CPPCR + 2

2 Feedback Divider Value M.

M = PLLM of CPPCR + 2

3 Output Divider Value NO.

Output Divider Setting (OD)	Output Divider Value (NO)
0	1
1	2
2	2
3	4

4 The PLL output frequency, CLK_OUT, is determined by the ratio set between the value set in the input divider and the feedback divider. PLL output frequency CLK_OUT is calculated from the following equations:

CLKOUT = XIN x (M / N) x (1/ NO)

M = F0 *1 + F1*2 + F2*4 + F3*8 + F4*16 + F5*32 + F6*64 + F7*128 + F8*256 + 2
N = R0*1 + R1*2 + R2*4 + R3*8 + R4*16 + 2
NO = $2^{od0+od1}$

Where:

CLK_OUT represents the output frequency
XIN represents PLL input frequency
N represents input divider value
M represents feedback divider value
NO represents output divider value

< Attention >

1) 1MHZ ≤ XIN/N ≤ 15MHZ
2) 100MHZ ≤ CLK_OUT x NO ≤ 500MHZ

7.2.5.2 PLL out clock frequency selection

PLL-freq = PLL-freq-raw / NO, where NO = 1, 2, 4.
PLL-freq-raw = EXCLK * M / N, where M = integer of 2 ~ 513, N = integer of 2 ~ 33.

So, to generate a specified PLL-freq, there are many valid sets of NO, M and N value.

Smaller PLL-freq-raw is better since it consumes less power. Reduce PLL-freq-raw from 200MHz to 100MHz saving a few milliwatts. Please beware not put PLL-freq-raw less than 100MHz.

If EXCLK is in small jitter, like a crystal-generated clock, a smaller N is better.

7.2.6 Main Clock Division Change Sequence

Main clock (CCLK, HCLK, IPU_CLK, PCLK and MCLK) frequencies can be changed separately or simultaneously by changing division ratio. Following conditions must be obeyed:

1 CCLK must be integral multiple of HCLK, IPU_CLK.
2 The FREQUENCY RATIO OF CCLK AND HCLK CAN NOT BE 24 AND 32.
3 HCLK MUST BE EQUAL TO MCLK OR TWICE OF MCLK.
4 HCLK AND MCLK MUST BE INTEGRAL MULTIPLE OF PCLK.
5 HCLK must be equal to IPU_CLK or 2/3 of IPU_CLK.

Don't violate this limitation, otherwise unpredictable error may occur.

In normal mode, if CE bit of CPCCR is 1, changing CDIV, HDIV, IPU_DIV, PDIV or MDIV will start a Division Change Sequence immediately. If CE bit of CPCCR is 0, changing CDIV, HDIV, IPU_DIV, PDIV or MDIV will not start Division Change Sequence.

7.2.7 Change Other Clock Frequencies

The divider of LCD pixel clock (LPCLK), I2S device clock, SSI device clock, MSC device clock and USB clock can be changed by programming LPCDR , I2SCDR, SSICDR, MSCCDR and UDIV, respectively.

Change LPCDR I2SCDR SSICDR MSCCDR and UDIV as following steps:

1 Stop related devices with clock-gate function. Clock supplies to the devices are stopped.
2 Change LPCDR, I2SCDR, SSICDR, MSCCDR or UDIV. If CE is 1, clock frequencies are changed immediately. If CE is 0, clock frequencies are not changed until PLL Multiplier Change Sequence is started.
3 Cancel above clock-gate function.

7.2.8 Change Clock Source Selection

USB, I2S device clocks and LCD pix clock can be selected from two sources. Before change clock source, corresponding devices should be stopped using clock-gate function.

1 When USB clock source is changed (UCS bit of CPCCR), USB clock should be stopped.
2 When I2S clock source is changed (I2CS bit of CPCCR), AIC should be stopped.
3 When LCD pix clock source is changed (LSCS LTCS bit of LPCDR), LCD should be stopped.

When UCS, I2CS , LSCS, LTCS bit is changed, clock source is changed immediately.

When PCS of CPPCR is changed, the LCD AIC MSC SSI clock should be stopped.
When ECS of CPCCR is changed, the UART SADC I2C clock should be stopped.

7.2.9 EXCLK Oscillator

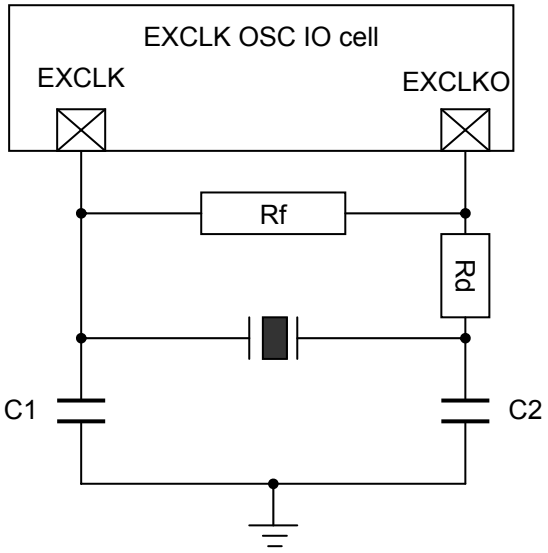


Figure 7-2 Oscillating circuit for fundamental mode

To turn on the oscillator, the oscillating circuit must provide the negative resistance (-Re) at least five times the equivalent series resistance (ESR) of the crystal sample. For larger -Re value, faster turn on the crystal. Higher gm provides larger -Re therefore can start-up the crystal with higher ESR for the same load capacitance (CL). However, it's required higher power consumption.

There are two key parameters to turn on oscillator. Which are CL and the maximum ESR at the target frequency. By reducing the CL, the -Re can be increased thus; shorter turn on time can be achieved. However, if CL is too small, the deviation from the target frequency will increase because of the capacitance variation. So, a trade-off relationship between short turn on time and small frequency deviation in deciding CL value. The smaller ESR of the crystal sample will reduce turn on time but the price is higher. The typical CL and ESR values for difference target frequencies are listed in Table 7-2.

Table 7-2 Typical CL and the corresponding maximum ESR

Target Frequency (Hz)	2M ~ 3M	3M ~ 6M	6M ~ 10M	10M ~ 20M
CL (pf)	25	20	16	12
Maximum ESR (ohm)	1K	400	100	80

Figure 7-2 shows the oscillating circuit is connected with the oscillator I/O cell. Components feedback resistor (Rf), damping resistor (Rd), C1 and C2 are used to adjust the turn on time, keep stability and accurate of the oscillator.

Rf is used to bias the inverter in the high gain region. It cannot be too low or the loop may not oscillate. For mega Hertz range applications, Rf of 1Mohm is applied.

Rd is used to increase stability, low power consumption, suppress the gain in high frequency region and also reduce -Re of the oscillator. Thus, proper Rd cannot be too large to cease the loop oscillating.

C1 and C2 are deciding regard to the crystal or resonator CL specification. In the steady state of oscillating, CL is defined as (C1*C2)/(C1+C2). Actually, the I/O ports, bond pad, and package pin all contribute the parasitic capacitance to C1 and C2. Thus, CL can be rewrite to (C1'*C2')/(C1'+C2'), where C1'=(C1+Cin,stray) and C2'=(C2+Cout,stray). In this case, the required C1 and C2 will be reduced.

Notice, this oscillating circuit is for parallel resonate but not series resonate. Because C1, C2, Rd and Rf are varying with the crystal specifications; therefore there is no single magic number of all the applications.

7.3 Power Manager

In the Low-Power mode, part or whole processor is halted. This will reduce power consumption. The Power Management Controller contains low-power mode control and reset sequence control.

7.3.1 Low-Power Modes and Function

The processor supports six low-power modes and function:

- NORMAL mode
  In Normal mode, all peripherals and the basic blocks including power management block, the CPU core, the bus controller, the memory controller, the interrupt controller, DMA, and the external master may operate completely. But, the clock to each peripheral, except the basic blocks, can be stopped selectively by software to reduce the power consumption.

- DOZE mode
  DOZE mode is entered by setting DOZE bit of LCR to 1. In DOZE mode, clock is burst to CPU core and the clock duty is set by DUTY field of LCR. DOZE mode is canceled by reset, interrupt or clearing DOZE bit to 0. Continuous clock is supplied immediately after DOZE mode is canceled. The other Clocks except CCLK run continuously in DOZE mode.

- IDLE mode
  In IDLE mode, the clock to the CPU core is stopped except the bus controller, the memory controller, the interrupt controller, and the power management block. To exit the IDLE mode, the any interrupts should be activated.

- SLEEP mode
  In SLEEP mode, all clocks except RTC clock are disabled. PLL is disabled also. SLEEP mode is canceled by reset or interrupt. When SLEEP mode is canceled, PLL is restarted , the PLL needs clock stabilization time (PLL lock time). This PLL stabilization time is automatically inserted by the internal logic with lock time count register. and all clocks start operating after PLL stability time.

- CLOCK GATE function
  CLOCK GATE function is used to gate specified on-chip module when it is not used. Set specified CLKG0~15 bits in CLKGR will enter specified CLK gate function. CLOCK gate function is canceled by reset or clearing specified CLKGR0~15 to 0.

7.3.2 Register Description

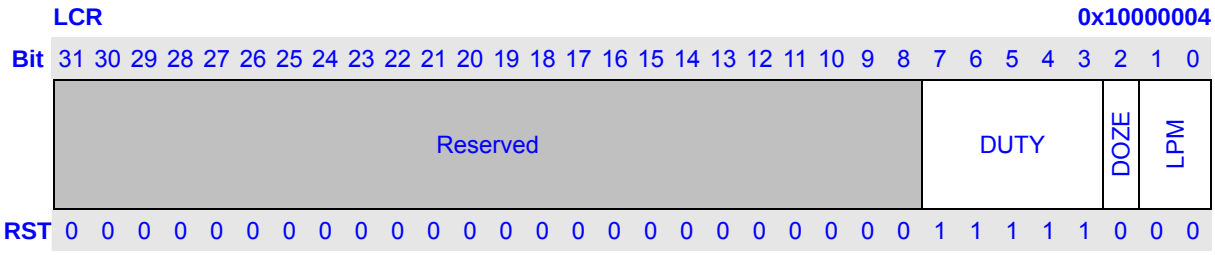
All PMC register 32bit access address is physical address.

Table 7-3 Power/Reset Management Controller Registers Configuration

Name	description	RW	Initial Value	Address	Access Size
LCR	Low Power Control Register	RW	0x000000F8	0x10000004	32
CLKGR	Clock Gate Register	RW	0x00000000	0x10000020	32
OPCR	Oscillator and Power Control Register	RW	0x00001500	0x10000024	32

7.3.2.1 Low Power Control Register

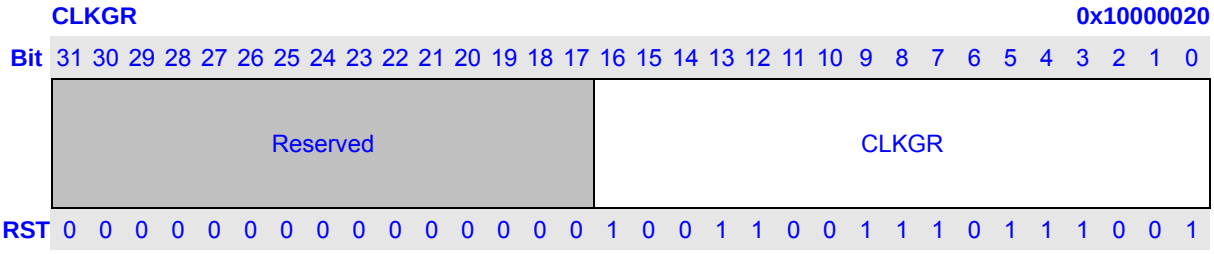
The Low Power Control Register (LCR) is a 32-bit read/write register that controls low-power mode status. It is initialized to 0x000000F8 by any reset.



Bits	Name	Description	RW
31:8	Reserved	Writes to these bits have no effect and always read as 0.	R
7:3	DUTY	CPU Clock Duty. Control the CPU clock duty in doze mode. When the DUTY field is 0x1F, the clock is always on and when it is zero, the clock is always off. Set the DUTY field to 0 when the CPU will be disabled for an extended amount of time. 00000: 0/31 duty-cycle 00001: 1/31 duty-cycle 00010: 2/31 duty-cycle … 11111: 31/31 duty-cycle	RW
2	DOZE	Doze Mode. Control the doze mode. When doze mode is canceled, this bit is cleared to 0 automatically. 0: Doze mode is off 1: Doze mode is on	RW
1:0	LPM	Low Power Mode. Specifies which low-power mode will be entered when SLEEP instruction is executed. Bit 1~0: **00**: IDLE mode will be entered when SLEEP instruction is executed **01**: SLEEP mode will be entered when SLEEP instruction is executed **10**: Reserved **11**: Reserved	RW

7.3.2.2 Clock Gate Register

The Clock Gate Register (CLKGR) is a 32-bit read/write register that controls the CLOCK GATE function of peripherals. It is reset to 0x000133B9.

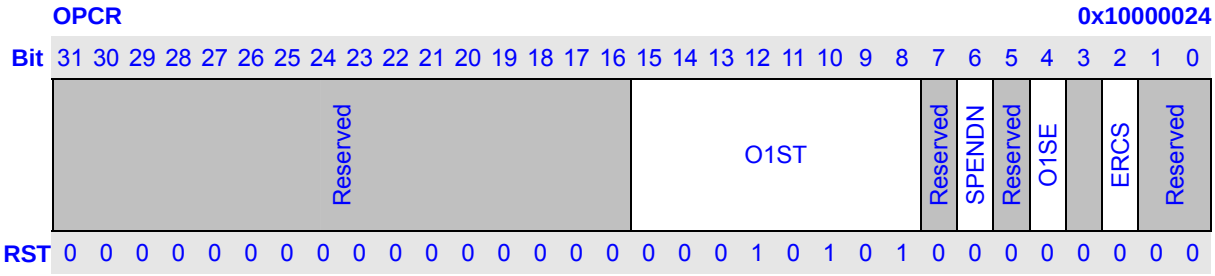


Bits	Name	Description	RW
31:29	Reserved	Writes to these bits have no effect and always read as 0.	R
28:0	CLKGR	Clock gate Bits. Controls the clock supplies to some peripherals. If set, clock supplies to associated devices are stopped, and registers of the device cannot be accessed also.	RW

Bit	Module	Description
31:17	Reserved	
16	MSC1	After reset period, the clock is stopped.
15	Reserved	
14	Reserved	
13	IPU	After reset period, the clock is stopped.
12	DMAC	After reset period, the clock is stopped.
11	BCH	
10	UDC	0: udc_hclk always running, don't stop 1: Only udc enters suspend mode, udc_hclk has been stopped . if the bit is 1 and udc doesn't enter suspend mode, udc_hclk always runs.
9	LCD	After reset period, the clock is stopped.
8	CIM	After reset period, the clock is stopped.
7	SADC	After reset period, the clock is stopped.
6	MSC0	
5	AIC	After reset period, the clock is stopped.
4	SSI	After reset period, the clock is stopped.
3	I2C	After reset period, the clock is stopped.
2	RTC	
1	TCU	
0	UART0	After reset period, the clock is stopped.

7.3.2.3 Oscillator and Power Control Register (OPCR)

The Oscillator and Power Control Register is a 32-bit read/write register that specifies some special controls to oscillator and analog block. It is initialized to 0x00001500 by reset.



Bits	Name	Description	RW
31:16	Reserved	Writes to these bits have no effect and always read as 0.	R
15:8	O1ST	EXCLK Oscillator Stabilize Time. This filed specifies the EXCLKoscillator stabilize time by unit of 16 RTCCLK periods (oscillator stable time O1ST × 16 / 32768) cycles. It is initialized to H'15.	RW
7	Reserved		R
6	SPENDN	force UDC phy to enter suspend mode. 0: UDC phy has forced to entered SUSPEND mode 1: UDC phy hasn't forced to entered SUSPEND mode	RW
5	Reserved		RW
4	O1SE	EXCLK Oscillator Sleep Mode Enable. This filed controls the state of the EXCLK oscillator in Sleep mode. 0: EXCLK oscillator is disabled in Sleep mode 1: EXCLK oscillator is enabled in Sleep mode	RW
3	Reserved		R
2	ERCS	EXCLK/512 clock and RTCLK clock selection. 0: select EXCLK/512 division ration clock 1: select RTCLK clock the clock only output to CPM INTC SSI TCU etc.	RW
1:0	Reserved		R

7.3.3 Doze Mode

Firstly, software should set the DUTY bits of LCR. Then set DOZE bit of LCR to 1 to enter doze mode. When slot controller of PMC indicates that the CPU clock's time-slot has expired, CPU is halted but its register contents are retained. During doze mode, program can modify clock duty-cycle according to core resource requirement. Clock control is in increments of approximately 3% (1/31).

Doze is exited by software, interrupt, reset or SLEEP instruction.

7.3.4 IDLE Mode

In normal or mode, when LPM bits in LCR are 0 and SLEEP instruction is executed, the processor enters idle mode. CPU is halted but its register contents are retained All critical application must be finished and peripherals must be configured to generate interrupts when they need CPU attention.

The procedure of entering sleep mode is shown blow:

1 Set LPM bits in LCR to 0.
2 Executes SLEEP instruction.
3 When current operation of CPU core has finished and CPU core is idle, CCLK supply to CPU core is stopped.

IDLE mode is exited by an interrupt (IRQ or on-chip devices) or a reset.

7.3.5 SLEEP Mode

In normal mode, when LPM bits in LCR is 1 and SLEEP instruction is executed, the processor enter SLEEP mode. CPU and on-chip devices are halted, except some wakeup-logic. PLL is shut off. Clock output from CKO pin is also stopped. SDRAM content is preserved by driving into self-refresh state. CPU registers and on-chip devices registers contents are retained.

Before enter SLEEP mode, software should ensure that all peripherals are not running. The procedure of entering SLEEP mode is shown blow:

1 Set LPM bit in LCR to 1.
2 Execute a SLEEP instruction.
3 When current access on system bus complete, the arbiter will not grant any following request. EMC will drive SDRAM from auto-refresh mode to self-refresh mode.
4 When system bus is idle state and SDRAM is self-refresh mode, internal clock supplies are stopped.

SLEEP mode can be exited by an interrupt (IRQ or on-chip devices), WDT reset or a poweron reset via the RESETP pin.

7.4 Reset Control Module

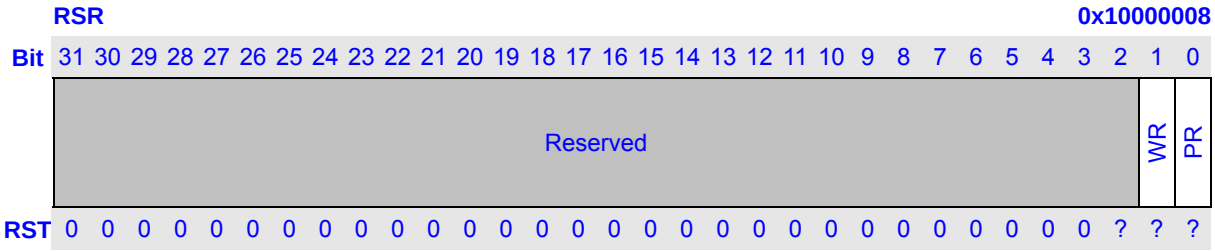
7.4.1 Register Description

All RCM register 32bit access address is physical address.

Name	description	RW	Initial Value	Address	Access Size
RSR	Reset Status Register	RW	0x????????	0x10000008	32

7.4.1.1 Reset Status Register (RSR)

The Reset Status Register (RSR) is a 32-bit read/write register which records last cause of reset. Each RSR bit is set by a different source of reset. Please refer to Reset Sequence Control for reset sources description.



Bits	Name	Description	RW
31:2	Reserved	Writes to these bits have no effect and always read as 0.	R
1	WR	WDT Reset. When a WDT reset is detected, WR is set and remains set until software clears it or another reset occurs. This bit can only be written with 0. Write with 1 will be ignored. 0: WDT reset has not occurred since the last time the software clears this bit 1: WDT reset has occurred since the last time the software clears this bit	RW
0	PR	Power On Reset. When a poweron reset via PRESET pin is detected, PR is set and remains set until software clears it or another reset occurs. This bit can only be written with 0. Write with 1 is ignored. 0: Power on reset has not occurred since the last time the software clears this bit 1: Power on reset has occurred since the last time the software clears this bit	RW

7.4.2 Power On Reset

Power on reset is generated when PRESET pin is driven to low. Internal reset is asserted immediately. All pins return to their reset states. The Power on reset is extended to 40MS.

PRESET pin must be held low until power stabilizes and the EXCLK oscillator stabilize. CPU and peripherals are clocked by EXCLK oscillator output directly. PLL is reset to off state. All internal modules are initialized to their predefined reset states.

7.4.3 WDT Reset

WDT reset is generated when WDT overflow. Internal reset is asserted within two RTCCLK cycles. All pins return to their reset states.

Then WDT reset source is cleared because of internal reset. The internal reset is asserted for about 10 milliseconds. CPU and peripherals are clocked by EXCLK oscillator output directly. PLL is reset to off state.

8 Real-Time Clock (RTC)

8.1 Overview

The Real-Time Clock (RTC) unit can be operated in either chip main power is on or the main power is down but the RTC power is still on. In this case, the RTC power domain consumes only a few micro watts power.

The RTC contains a 32768Hz oscillator, the real time and alarm logic, and the power down and wakeup control logic.

8.1.1 Features

RTC module has following features:

- Embedded 32768Hz oscillator for 32k clock generation with an external 32k crystal
- RTCLK selectable from the oscillator or from the divided clock of EXCLK, so that 32k crystal can be absent if the hibernating mode is not needed
- 32-bits second counter
- Programmable and adjustable counter to generate accurate 1 Hz clock
- Alarm interrupt, 1Hz interrupt
- Stand alone power supply, work in hibernating mode
- Power down controller
- Alarm wakeup
- External pin wakeup with up to 2s glitch filter

8.1.2 Signal Descriptions

RTC has 5 signal IO pins and 1 power pin. They are listed and described in.

Pin Names	Pin Loc	IO	IO Cell Char.	Pin Description	Power
RTCLK		AI	32768Hz	RTCLK: 32768 clock input or OSC input	VDD_{RTC}
RTCLKO		AO		RTCLKO: OSC output	VDD_{RTC}
PWRON_		AO	~2mA, Open-Draw	PWRON_: Power on/off control of main power	VDD_{RTC}
WKUP_		AI	Schmitt	WKUP_: Wake signal after main power down	VDD_{RTC}
PPRST_		AI	Schmitt	PPRST_: RTC power on reset and RESET-KEY reset input	VDD_{RTC}
VDDRTC		P		VDDRTC: 3.3V power for RTC and hibernating mode controlling that never power down	-

RTCLK/RTCLKO pins. We have an embedded oscillator for 32768Hz crystal. These two pins are the crystal XTALI and XTALO connection pins. If an input clock is used instead, please input it to RTCLKO pin.

If do not use any clock, hibernate mode will be NOT available any more, and the time will lose if power down.

PWRON_ pin: this pin is used to control the main power on/off. Output low voltage means on and high-Z means off.

WKUP_ pin: hibernating mode wakeup input.

PPRST_ pin: This pin should be set to low voltage only in two cases.

- When RTC power is turned on (so that whole chip is power on)
- A RESET-KEY is pressed

Don't set this pin to low voltage when wakeup from hibernating mode. When entering/exiting to/from hibernating mode (in another word, in main power up/down procedure), please avoid putting both WKUP_ and PPRST_ in low voltage. Because the RTC registers, for instance, the second counter and others may be changed.

8.2 Register Description

Table 8-1 Registers for real time clock

Name	Description	RW	Reset Value	Address	Access Size
RTCCR	RTC Control Register	RW	0x00000081[1][2]	0x10003000	32
RTCSR	RTC Second Register	RW	0x????????	0x10003004	32
RTCSAR	RTC Second Alarm Register	RW	0x????????	0x10003008	32
RTCGR	RTC Regulator Register	RW	0x0???????	0x1000300C	32

NOTES:

1 Unless otherwise stated, the reset value is for PPRST_ and Hibernating wakeup reset. WDT reset doesn't change the value.

2 The reset value can be either of 0x00000081, 0x00000091, 0x00000089, 0x00000099.

Table 8-2 Registers for hibernating mode

Name	Description	RW	Reset Value	Address	Access Size
HCR	Hibernate Control Register	RW	0x00000000[1]	0x10003020	32
HWFCR	Wakeup filter counter Register in Hibernate mode	RW	0x0000???0	0x10003024	32
HRCR	Hibernate reset counter Register in Hibernate mode	RW	0x00000??0	0x10003028	32
HWCR	Wakeup control Register in Hibernate mode	RW	0x00000000[1]	0x1000302C	32
HWRSR	Wakeup Status Register in Hibernate mode	RW	0x00000000[1]	0x10003030	32
HSPR	Scratch pattern register	RW	0x????????	0x10003034	32

NOTES:

1 Unless otherwise stated, the reset value is for PPRST_ and Hibernating wakeup reset. WDT reset doesn't change the value.

All these registers, include those for real time clock and for hibernating mode control, except otherwise stated, are implemented in RTCLK clock domain. When write to these registers, it needs about 1 ~ 2 RTCLK cycles to actually change the register's value and needs another RTCLK cycle to allow the next write access. A bit RTCCR.WRDY is used to indicate it. When RCR.WRDY is 1, it means the previous write is finished, a right value can be read from the target register, and a new write access can be issued. So before any write access, please make sure RCR.WRDY = 1.

8.2.1 RTC Control Register (RTCCR)

RTCCR contains bits to configure the real time clock features. Unless otherwise stated, the reset value is for PPRST_ and Hibernating wakeup reset. WDT reset doesn't change the value.

RTCCR **0x10003000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																								WRDY	1HZ	1HZIE	AF	AIE	AE	SELEXC	RTCE
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1[1]	0	0[1]	?	?	0	0[1]	1

NOTES:

1 These bits are reset in all resets: PPRST_ input pin reset, hibernating reset and WDT reset.

Bits	Name	Description	RW
31:7	Reserved	Writes to these bits have no effect and always read as 0.	R
7	WRDY	Write ready flag. It is 0 when a write is currently processing and the value has not been written to the writing target register. No write to any RTC registers can be issued in this case, or the result is undefined. The read value from the target register is also undefined. The reading is meaningful and another write can be issued when it is 1. Please reference to descriptions in 0 for some more details. This bit is read only and write to it is ignored.	R
6	1HZ	1Hz flag. This bit is set by hardware once every 1 second through the 1Hz pulse if the real time clock is enabled (RTCCR.RTCE = 1). This bit can be cleared by software. Write 1 to this bit is ignored. Writing to this bit takes effect immediately without delay.	RW
5	1HZIE	1Hz interrupt enable. Writing to this bit takes effect immediately without delay. **1HZIE** / **Description** 0 / 1Hz interrupt is disabled. 1 / 1Hz interrupt is enabled. RTC issues interrupt when 1HZ bit is set.	RW
4	AF	Alarm flag. This bit is set by hardware when alarm match (RTCSR = RTCSAR) is found and alarm is enabled (RTCCR.AE = 1) and the real time clock is enabled (RTCCR.RTCE = 1). This bit can be cleared by software. Write 1 to this bit is ignored. Writing to this bit takes effect immediately.	RW
3	AIE	Alarm interrupt enable. **AIE** / **Description** 0 / Alarm interrupt is disabled. 1 / Alarm interrupt is enabled. RTC issues interrupt	RW

		when AF is set.	
2	AE	Alarm enable. **AE** / **Description** 0 – Alarm function is disabled. 1 – Alarm function is enabled.	RW
1	SELEXC	The divided EXCLK is selected as RTCLK in rtc-hiber module. **SELEXC** / **Description** 0 – OSC32K or RTCLK input clock is selected as RTCLK in rtc-hiber module. 1 – The divided EXCLK is selected as RTCLK in rtc-hiber module. **NOTE:** If do not use any 32Khz clock (either input clock or using crystal), hibernate mode will be NOT available any more, and the time will lose if power down. CPM.OPCR.ERCS must be 0, when using SELEXC = 1. When the main chip power down, SELEXC will be 0 in internal circuit, in this time, RTCLK will use OSC32K clock.	RW
0	RTCE	Real time clock enable. **RTCE** / **Description** 0 – Real time clock function is disabled. 1 – Real time clock function is enabled.	RW

8.2.2 RTC Second Register (RTCSR)

RTCSR is a 32-bit width second counter. It can be read and write by software. It is increased by 1 at every 1Hz pulse if the real time clock is enabled (RTCCR.RTCE = 1). When read, it should be read continued more than once and take the value if the adjacent results are the same. RTCSR is not initialized by any reset.

RTCSR **0x10003004**

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

8.2.3 RTC Second Alarm Register (RTCSAR)

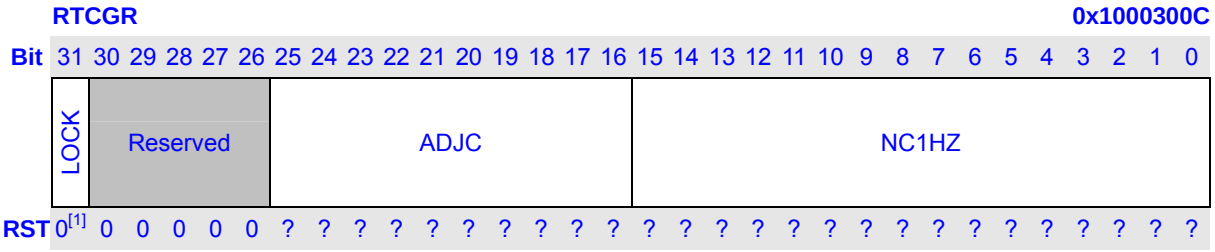
RTCSAR serves as a second alarm register. Alarm flag (RTCCR.AF) is set to 1 when the RTCSR equals the RTCSAR in the condition of alarm is enabled (RTCCR.AE = 1) and the real time clock is enabled (RTCCR.RTCE = 1). RTCSAR can be read and write by software and is not initialized by any reset.

RTCSAR **0x10003008**

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

8.2.4 RTC Regulator Register (RTCGR)

RTCGR is serves as the real time clock regulator, which is used to adjust the interval of the 1Hz pulse.



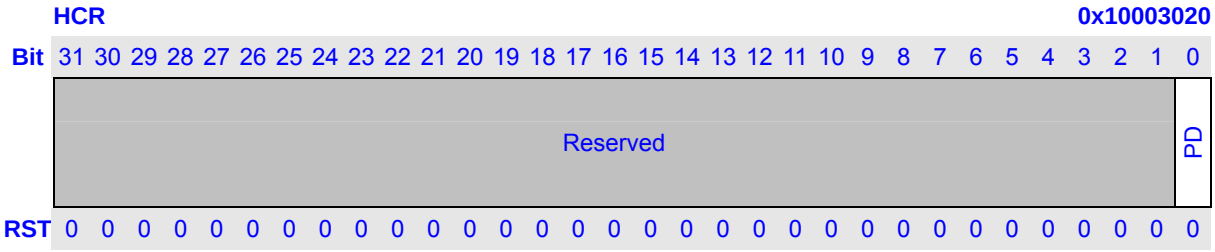
NOTES:

1 This bit is reset in all resets: PPRST_ input pin reset, hibernating reset and WDT reset.

Bits	Name	Description	RW
31	LOCK	Lock bit. This bit is used to safeguard the validity of the data written into the RTCGR register. Once it is set, write to RTCGR is ignored. This bit can only be set by software and cleared by (any type of) resets. LOCK: 0 – Write to RTCGR is allowed. LOCK: 1 – Write to RTCGR is forbidden.	RW
30:26	Reserved	Writes to these bits have no effect and always read as 0.	R
25:16	ADJC	This field specifies how many times it needs to add one 32kHz cycle for the 1Hz pulse interval in every 1024 1Hz pulses. In other word, among every 1024 1Hz pulses, ADJC number of them are trigged in every (NC1HZ + 2) 32kHz clock cycles, (1024 – ADJC) number of them are trigged in every (NC1HZ + 1) 32kHz clock cycles.	RW
15:0	NC1HZ	This field specifies the number plus 1 of the working 32kHz clock cycles are contained in the 1Hz pulse interval. In other word, 1Hz pulse is trigged every (NC1HZ + 1) 32kHz clock cycles, if RTCGR.ADJC = 0.	RW

8.2.5 Hibernate Control Register (HCR)

HCR contains the bit to control the main chip power on/off.



Bits	Name	Description	RW
31:1	Reserved	Writes to these bits have no effect and always read as 0.	R
0	PD	Power down or power on bit. Besides writing by CPU, this bit will be set to 1 if an unknown reason main power supply off is detected. This bit controls the PWRON_ pin level. When co-working with some external components, this bit is used for power management of this chip. It is supposed when 1 is written to this bit, the main power supply of the chip, except RTC power, will be shut down immediately. After this bit is set to 1, all registers in RTC module, except RTCCR.1HZ and RTCCR.1HZIE, cannot be changed by write access. This bit is cleared by reset pin reset and hibernating reset. The later one is asserted by wakeup procedure.	RW

PD	PWRON_	Description
0	0 V	No power down, keep power on.
1	VDDRTC	Power down enable, turn power off.

8.2.6 HIBERNATE mode Wakeup Filter Counter Register (HWFCR)

The HIBERNATE mode Wakeup Filter Counter Register (HWFCR) is a 32-bit read/write register .It filters the glitch generated by a dedicated wakeup pin. The HRCR is initialized by PPRST_ and WDT reset.

HWFCR **0x10003024**

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

Bits	Name	Description	RW
31:16	Reserved	Writes to these bits have no effect and always read as 0.	R
15:5	HWFCR	Wakeup pin effective minimum time in number of 32 RTCLK cycles, used as glitch filter logic. Maximum of 2 seconds if the RTCLK is 32768Hz.	RW
4:0	Reserved	Writes to these bits have no effect and always read as 0.	R

8.2.7 Hibernate Reset Counter Register (HRCR)

The Hibernate Reset Counter Register is a 32-bit read/write register that specifies hibernate reset assertion time. The HRCR is initialized by PPRST_ and WDT reset.

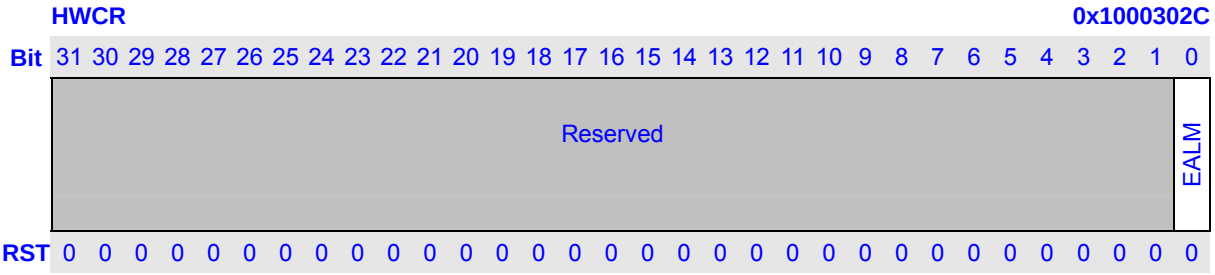
HRCR **0x10003028**

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

Bits	Name	Description	RW
31:12	Reserved	Writes to these bits have no effect and always read as 0.	R
11:5	HRCR	HIBERNATE Reset waiting time. Number of 32 RTCLK cycles. Maximum 125 ms if the RTCLK is 32768Hz.	RW
4:0	Reserved	Writes to these bits have no effect and always read as 0.	R

8.2.8 HIBERNATE Wakeup Control Register (HWCR)

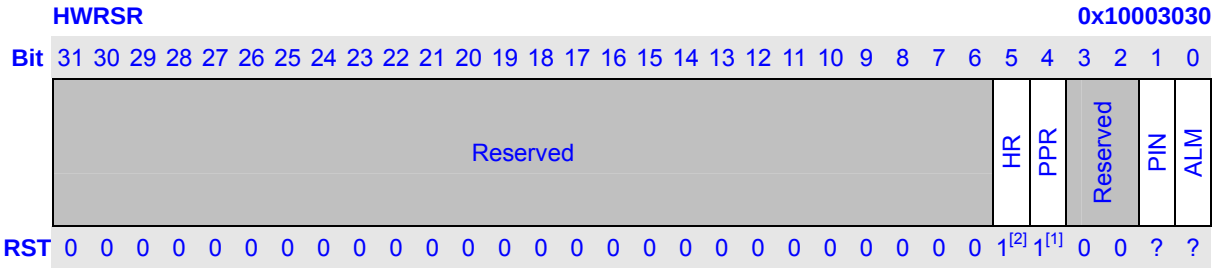
The HIBERNATE Wakeup Control Register is a 32-bit read/write register that controls real time clock alarm wake up enable. The reset value is for PPRST_ and Hibernating wakeup reset. WDT reset doesn't change the value.



Bits	Name	Description	RW
31:1	Reserved	Writes to these bits have no effect and always read as 0.	R
0	EALM	RTC Alarm wakeup enable. 0: disable 1: enable	RW

8.2.9 HIBERNATE Wakeup Status Register (HWRSR)

The HIBERNATE Wakeup Status Register is a 32-bit read/write register that reflects wakeup status bits.



NOTES:

1 This reset value only for PPRST_. It is undefined in case of other resets.

2 This reset value only for HRST_. It is undefined in case of other resets.

Bits	Name	Description	RW
31:6	Reserved	Writes to these bits have no effect and always read as 0.	R
5	HR	Hibernate Reset. When a Hibernate reset detected, HR is set and remains set until software clears it or another reset occurs. This bit can only be written with 0. Write with 1 is ignored. **HR** / **Description** 0 / Hibernate reset has not occurred since the last time the software clears this bit. 1 / Hibernate reset has occurred since the last time the software clears this bit.	RW
4	PPR	PAD PIN Reset. When a PPRST_ is detected, PPR is set and remains set until software clears it or another reset occurs. This bit can only be written with 0. Write with 1 is ignored. **PPR** / **Description** 0 / PPRST_ reset has not occurred since last time the software clears this bit. 1 / PPRST_ reset has occurred since last time the software clears this bit.	RW
3:2	Reserved	Writes to these bits have no effect and always read as 0.	R
1	PIN	Wakeup Pin Status bit. The bit is cleared when chip enters hibernating mode. It is set when exit the hibernating mode by wakeup pin. This bit can only be written with 0. Write with 1 is ignored.	RW
0	ALM	RTC Alarm Status bit. The bit is cleared when chip enters hibernating mode. It is set when exit the hibernating mode by alarm. This bit can only be written with 0. Write with 1 is ignored.	RW

8.2.10 Hibernate Scratch Pattern Register (HSPR)

This is a scratch register used to hold a pattern. The software can check the pattern is kept to know whether RTC power has ever been down and whether it is needed to setup the real time clock.

HSPR 0x10003034

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

Bits	Name	Description	RW
31:0	PAT	The pattern.	RW

8.3 Time Regulation

Because of the inherent inaccuracy of crystal and other variables, the time counter may be inaccurate. This requires a slight adjustment. The application processor, through the RTCGR, lets you adjust the 1Hz time base to an error of less than 1ppm. Such that if the Hz clock were set to be 1Hz, there would be an error of less than 5 seconds per month.

To determine the value programmed into the RTCGR, you must first measure the output frequency at the oscillator multiplex (approximately 32 kHz) using an accurate time base, such as a frequency counter. This clock is externally visible by selecting the alternate function of GPIO[?]
To gain access to the clock, program this pin as an output and then switch to the alternate function. To trim the clock, divide the output of the oscillator by an integer value and fractional adjust it by periodically deleting clocks from the stream driving this integer divider.

After the true frequency of the oscillator is known, it must be split into integer and fractional portions. The integer portion of the value (minus one) is loaded into the DIV field of the RTCGR.

The fractional part of the adjustment is done by periodically deleting clocks from the clock stream driving the Hz divider. The trim interval period is hardwired to be 1024 1Hz clock cycles (approximately 17 minutes). The number of clocks (represented by ADC field of RTCGR) are deleted from the input clock stream per trim interval. If ADC is programmed to be zero, then no trim operations occur and the RTC is clocked with the raw 32 kHz clock. The relationship between the Hz clock frequency and the nominal 32 kHz clock (f1 and f32K, respectively) is shown in the following equation.

$$f1 = \frac{2^{10} \times (DIV + 1)}{2^{10} \times (DIV + 1) + ADC} \times \frac{f32k}{DIV + 1}$$

f1 = actual frequency of 1Hz clock
f32k = frequency of either 32.768KHz crystal output or 3.6864MHz crystal output further divided down to 32.914KHz

8.3.1 HIBERNATE Mode

First make sure RTCCR.SELEXC is 0.

When Software writes 1 to PD bit of HCR, the system at once enters HIBERNATE mode. The powers of CORE and IO are disconnected by PWRON_ pin, no power consumption to core and IO. When a wakeup event occurs, the core enters through a hibernate reset. Only CPM wake up logic and RTC is operating in HIBERNATE mode.

8.3.1.1 Procedure to Enter HIBERNATE mode

Before enter HIBERNATE mode, software must complete following steps:

1 Finish the current operation and preserve all data to flash.
2 Configure the wake-up sources properly by configure HWCSR.
3 Set HIBERNATE MODE. (Set PD bit in HCR to 1.)

8.3.1.2 Procedure to Wake-up from HIBERNATE mode

1 The internal hibernate reset signal will be asserted if one of the wake-up sources is issued.
2 Check RSR to determine what caused the reset.
3 Check PIN/ALM bits of HWCSR in order to know whether or not the power-up is caused by which wake-up from HIBERNATE mode.
4 Configure the SDRAM memory controller.
5 Recover the data from flash.

NOTES:

8.4 Clock select

There could be two clock input to RTC internal clock called rtclk. One is OSC32k clock; the other is EXCLK/512.

The software MUST make sure the RTC run in valid clock configuration.

Table 8-3 Clock select registers

RTCCR.SELEXC	CPM.ERCS	Description	Valid
0	0	RTC use OSC32K clock	OK
0	1		OK
1	0	RTC use EXCLK/512 clock	OK
1	1	RTC will lost clock (Not Valid)	NO

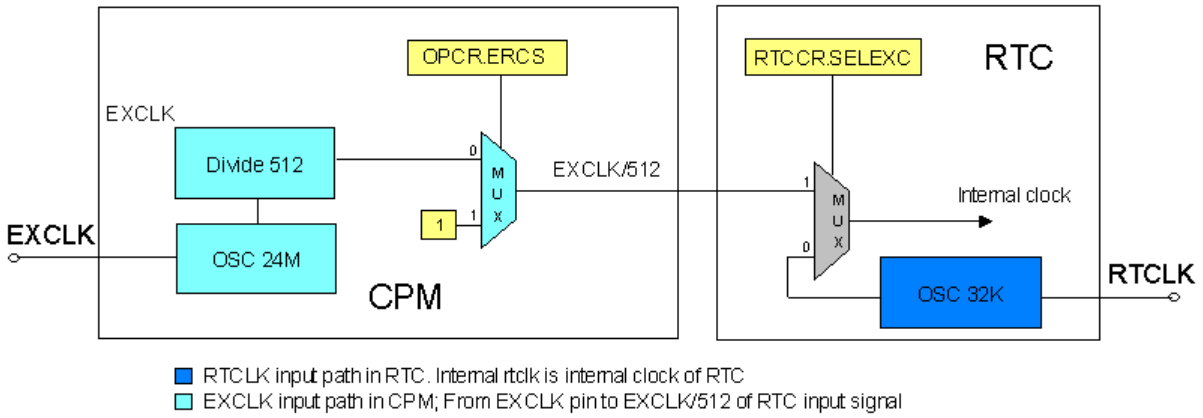


Figure 8-1 RTC clock selection path

Changing RTCLK sequence:

1 There are both 32KHz crystal and 24Mhz EXCLK crystal connected, so RTCLK input path has 32Khz clock.

In this case, there is no need to change internal clock, so do NOT change SELEXC all the time.

2 There is no 32KHz crystal connected but only 24Mhz EXCLK crystal connected, so RTCLK input path has no clock.

In this case, should flow the sequence below to change internal clock:

a Set OPCR.ERCS of CPM to 1; close EXCLK/512 to RTC.

b Set CLKGR.RTC of CPM to 1; close PCLK to RTC.

c Set RTCCR.SELEXC to 1; change internal clock to EXCLK/512.

d Wait two clock period of clock.

e Clear OPCR.ERCS of CPM to 0; open EXCLK/512 to RTC.

f Clear CLKGR.RTC of CPM to 0; open PCLK to RTC.

g Configure all RTC registers but RTCCR.SELEXC.

h Check RTCCR.SELEXC == 1.

i IF YES, finish this sequence; IF NO, do step (1) again.

NOTES:

1 If using HIBERNATE mode, MUST have both 32KHz crystal (or input 32Khz clock) and 24Mhz EXCLK crystal connected, or RTC time will be insignificant.

9 Interrupt Controller

9.1 Overview

This chapter describes the interrupt controller included in the XBurst Processor, explains its modes of operation, and defines its registers. The interrupt controller controls the interrupt sources available to the processor and contains the location of the interrupt source to allow software to determine source of all interrupts. It also determines whether the interrupts cause an IRQ to occur and masks the interrupts.

Features:

- Total 32 interrupt sources
- Each interrupt source can be independently enabled
- Priority mechanism to indicate highest priority interrupt
- All the registers are accessed by CPU
- Unmasked interrupts can wake up the chip in sleep mode

9.2 Register Description

Table 9-1 INTC Register lists the registers of Interrupt Controller. All of these registers are 32bit, and each bit of the register represents or controls one interrupt source that list in Table 9-1 INTC Register.

All INTC register 32bit access address is physical address.

Table 9-1 INTC Register

Name	Description	RW	Reset Value	Address	Access Size
ICSR	Interrupt controller Source Register	R	0x00000000	0x10001000	32
ICMR	Interrupt controller Mask Register	RW	0xFFFFFFFF	0x10001004	32
ICMSR	Interrupt controller Mask Set Register	W	0x????????	0x10001008	32
ICMCR	Interrupt controller Mask Clear Register	W	0x????????	0x1000100C	32
ICPR	Interrupt controller Pending Register	R	0x00000000	0x10001010	32
ICSSR	Interrupt controller Source Set Register	W	0x00000001	0x13016000	32

9.2.1 Interrupt Controller Source Register (ICSR)

This register contains all the interrupts' status. A "1" indicates that the corresponding interrupt is pending. A "0" indicates that the interrupt is not pending now. The register is read only.

ICSR **0x10001000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	LCD	IPU	DMA0	Reserved	UDC	SSI	MSC0	MSC1	TCU0	TCU1	TCU2	Reserved	CIM	SADC	BCH	GPIO0	GPIO1	GPIO2	GPIO3	Reserved		AIC	UART0	Reserved		RTC	I2C	SFT	Reserved			
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits Of ICSR	Description
0	The corresponding interrupt source is not pending.
1	The corresponding interrupt source is pending.

9.2.2 Interrupt Controller Source Set Register (ICSSR)

Software can write this bit to trigger / clear an interrupt. This register can be read or write
Please notice that the interrupt will continue until you set this bit to 1.

ICSSR 0x13016000

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

Bits Of ICSSR	Description
0	Set software interrupt.
1(reset value)	Clear software interrupt.

9.2.3 Interrupt Controller Mask Register (ICMR)

This register is used to mask the interrupt input sources and defines which active sources are allowed to generate interrupt requests to the processor. Its value can be changed either by writing ICMSR and ICMCR or by writing itself. The masked interrupts are invisible to the processor.

ICMR 0x10001004

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	LCD	IPU	DMA0	Reserved	UDC	SSI	MSC0	MSC1	TCU0	TCU1	TCU2	Reserved	CIM	SADC	BCH	GPIO0	GPIO1	GPIO2	GPIO3	Reserved		AIC	UART0	Reserved		RTC	I2C	SFT	Reserved			
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits Of ICMR	Description
0	The corresponding interrupt is not masked.
1	The corresponding interrupt is masked.

9.2.4 Interrupt Controller Mask Set Register (ICMSR)

This register is used to set bits in the interrupt mask register. This register is write only.

ICMSR 0x10001008

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	LCD	IPU	DMA0	Reserved	UDC	SSI	MSC0	MSC1	TCU0	TCU1	TCU2	Reserved	CIM	SADC	BCH	GPIO0	GPIO1	GPIO2	GPIO3	Reserved		AIC	UART0	Reserved		RTC	I2C	SFT	Reserved			
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits Of ICMSR	Description
0	Ignore.
1	Will set the corresponding interrupt mask bit.

9.2.5 Interrupt Controller Mask Clear Register (ICMCR)

This register is used to clear bits in the interrupt mask register. This register is write only.

ICMCR **0x1000100C**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	LCD	IPU	DMA0	Reserved	UDC	SSI	MSC0	MSC1	TCU0	TCU1	TCU2	Reserved	CIM	SADC	BCH	GPIO0	GPIO1	GPIO2	GPIO3	Reserved		AIC	UART0	Reserved		RTC	I2C	SFT	Reserved			
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits Of ICMCR	Description
0	Ignore.
1	Will clear the corresponding interrupt mask bit.

9.2.6 Interrupt Controller Pending Register (ICPR)

This register contains the status of the interrupt sources after masking. This register is read only.

ICPR **0x10001010**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	LCD	IPU	DMA0	Reserved	UDC	SSI	MSC0	MSC1	TCU0	TCU1	TCU2	Reserved	CIM	SADC	BCH	GPIO0	GPIO1	GPIO2	GPIO3	Reserved		AIC	UART0	Reserved		RTC	I2C	SFT	Reserved			
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits Of ICPR	Description
0	The corresponding interrupt is not active or is masked.
1	The corresponding interrupt is active and is not masked to the processor.

NOTE: Reserved bits in ICMR, ICMSR and ICMCR are normal bits to be written into and read out. Reserved bits in ICSR and ICPR are read-only and always 0.

9.3 Software Considerations

The interrupt controller is reflecting the status of interrupts sources in the peripheral.

Software should perform the task - determine the interrupt source from in ICPR. In this chip, pending interrupts have two levels in structure. Interrupting module in the system that contains more than one interrupt sources need software to determine how to service it by reading interrupt status registers within it.

In the interrupt handler, the serviced interrupt source needs to be cleared in the interrupting device. In order to make certain the cleared source request status has been reflected at the corresponding ICPR bit, software should wait enough time before exiting interrupt state.

The procedure is described following:

1 Interrupt generated.
2 CPU query interrupt sources, saves the current environment and then goes to interrupt common service routine.
3 Get ICPR.
4 Find the highest priority interrupt and vector it. (The software decides which one has the highest priority)
5 Mask the chosen interrupt by writing the register ICMSR.
6 Enable the system interrupt to allow the interrupt nesting.(software decided)
7 Execute the interrupt handler and unmask it by writing the register ICMCR when exit the handler.
8 CPU restores the saved environment and exit the interrupt state.

NOTE: If you want to use software interrupt, you need to set the SFT bit of the corresponding.

10 Timer/Counter Unit

10.1 Overview

The TCU (Timer/Counter with PWM output) contains 6 channels of 16-bit programmable timers (timers 0 to 5). They can be used as Timer or PWM.

TCU has the following features:

- There are two modes of TCU for the six channels
  - TCU1: Channel 0, 1,2, 3,and 4
  - TCU2: Channel 5
- Six independent channels, each consisting of
  - Counter
  - Data register (FULL and HALF)
  - Control register
- Independent clock for each counter, selectable by software
  - PCLK, EXTAL and RTCCLK can be used as the clock for counter
  - The division ratio of the clock can be set to 1, 4, 16, 64, 256 and 1024 by software
- FULL interrupt and HALF interrupt can be generated for each channel using the compare data registers
  - Timer 0-5 can be used as PWM (Set the initial signal level)
  - Timer 0 has separated interrupt
  - Timer 1-5 has one interrupt in common
  - OST uses interrupt 0, Timer 0 uses interrupt 1, and Timer 1-5 uses interrupt 2
- The difference between TCU1 and TCU2
  - TCU1: It cannot work in sleep mode, but operated easily
  - TCU2: It can work in sleep mode, but operated more complicated than TCU1

10.2 Pin Description

Table 10-1 PWM Pins Description

Name	I/O	Description
PWM [5:0]	Output	PWM channel output signals.

10.3 Register Description

In this section, we will describe the registers in timer. Following table lists all the registers definition. All timer register's 32bit address is physical address. And detailed function of each register will be described below.

Name	Description	RW	Reset Value	Address	Access Size
TSTR	Timer Status Register	R	0x00000000	0x100020F0	32
TSTSR	Timer Status Set Register	W	0x????????	0x100020F4	32
TSTCR	Timer Status Clear Register	W	0x????????	0x100020F8	32
TSR	Timer STOP Register	R	0x00000000	0x1000201C	32
TSSR	Timer STOP Set Register	W	0x00000000	0x1000202C	32
TSCR	Timer STOP Clear Register	W	0x0000	0x1000203C	32
TER	Timer Counter Enable Register	R	0x0000	0x10002010	16
TESR	Timer Counter Enable Set Register	W	0x????	0x10002014	16
TECR	Timer Counter Enable Clear Register	W	0x????	0x10002018	16
TFR	Timer Flag Register	R	0x003F003F	0x10002020	32
TFSR	Timer Flag Set Register	W	0x????????	0x10002024	32
TFCR	Timer Flag Clear Register	W	0x????????	0x10002028	32
TMR	Timer Mask Register	R	0x00000000	0x10002030	32
TMSR	Timer Mask Set Register	W	0x????????	0x10002034	32
TMCR	Timer Mask Clear Register	W	0x????????	0x10002038	32
TDFR0	Timer Data FULL Register 0	RW	0x????	0x10002040	16
TDHR0	Timer Data HALF Register 0	RW	0x????	0x10002044	16
TCNT0	Timer Counter 0	RW	0x????	0x10002048	16
TCSR0	Timer Control Register 0	RW	0x0000	0x1000204C	16
TDFR1	Timer Data FULL Register 1	RW	0x????	0x10002050	16
TDHR1	Timer Data HALF Register 1	RW	0x????	0x10002054	16
TCNT1	Timer Counter 1	RW	0x????	0x10002058	16
TCSR1	Timer Control Register 1	RW	0x0000	0x1000205C	16
TDFR2	Timer Data FULL Register 2	RW	0x????	0x10002060	16
TDHR2	Timer Data HALF Register 2	RW	0x????	0x10002064	16
TCNT2	Timer Counter 2	RW	0x????	0x10002068	16
TCSR2	Timer Control Register 2	RW	0x0000	0x1000206C	16
TDFR3	Timer Data FULL Register 3	RW	0x????	0x10002070	16
TDHR3	Timer Data HALF Register 3	RW	0x????	0x10002074	16
TCNT3	Timer Counter 3	RW	0x????	0x10002078	16
TCSR3	Timer Control Register 3	RW	0x0000	0x1000207C	16
TDFR4	Timer Data FULL Register 4	RW	0x????	0x10002080	16
TDHR4	Timer Data HALF Register 4	RW	0x????	0x10002084	16
TCNT4	Timer Counter 4	RW	0x????	0x10002088	16

TCSR4	Timer Control Register 4	RW	0x0000	0x1000208C	16
TDFR5	Timer Data FULL Register 5	RW	0x????	0x10002090	16
TDHR5	Timer Data HALF Register 5	RW	0x????	0x10002094	16
TCNT5	Timer Counter 5	RW	0x????	0x10002098	16
TCSR5	Timer Control Register 5	RW	0x0000	0x1000209C	16

10.3.1 Timer Control Register (TCSR)

The TCSR is a 16-bit read/write register. It contains the control bits for each channel. It is initialized to 0x00 by any reset.

TCSR0, TCSR1, TCSR2, TCSR3, TCSR4, TCSR5 — **0x1000204C, 0x1000205C, 0x1000206C, 0x1000207C, 0x1000208C, 0x1000209C**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved					CLRZ	SD	INITL	PWM_EN	Reserved	PRESCALE			EXT_EN	RTC_EN	PCK_EN
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15:11	Reserved	These bits always read 0, and written are ignored.	R
10	CLRZ	Clear counter to 0. It is only used in TCU2 mode. Writing 1 to this bit will clear the counter to 0. When the counter is finished setting to 0, it will be cleared by hardware. Writing 0 to this bit will be ignored.	RW
9	SD	Shut Down (SD) the PWM output. It is only used in TCU1 mode. 0: Graceful shutdown 1: Abrupt shutdown Graceful shutdown: The output level for PWM output will keep the level after the comparison match of FULL. Abrupt shutdown: The output level for PWM output will keep the level.	RW
8	INITL	Selects an initial output level for PWM output. 1: High 0: Low	RW
7	PWM_EN	PWM output pin control bit. 1: PWM pin output enable 0: PWM pin output disable, and the PWM pin will be set to the initial level according to INITL	RW
6	Reserved	These bits always read 0, and written are ignored.	R
5:3	PRESCALE	These bits select the TCNT count clock frequency. Don't change this field when the channel is running.	RW

		Bit 2	Bit1	Bit 0	Description	
		0	0	0	Internal clock: CLK/1	
		0	0	1	Internal clock: CLK/4	
		0	1	0	Internal clock: CLK/16	
		0	1	1	Internal clock: CLK/64	
		1	0	0	Internal clock: CLK/256	
		1	0	1	Internal clock: CLK/1024	
		110~111			Reserved	
2	EXT_EN	Select EXTAL as the timer clock input. 1: Enable 0: Disable				RW
1	RTC_EN	Select RTCCLK as the timer clock input. 1: Enable 0: Disable				RW
0	PCK_EN	Select PCLK as the timer clock input. 1: Enable 0: Disable				RW

NOTE: The input clock of timer and the PCLK should keep to the rules as follows:

Input clock of timer: IN_CLK	**Clock generated from the frequency divider (PRESCALE): DIV_CLK**
PCK_EN == 0, RTC_EN == 1 and EXT_EN == 0 (IN_CLK = RTCCLK)	$f_{DIV\_CLK} < ½ f_{PCLK}$
PCK_EN == 0, RTC_EN == 0 and EXT_EN == 1 (IN_CLK = EXTAL)	$f_{DIV\_CLK} < ½ f_{PCLK}$
PCK_EN == 1, RTC_EN == 0 and EXT_EN == 0 (IN_CLK = PCLK)	ANY

10.3.2 Timer Data FULL Register (TDFR)

The comparison data FULL registers TDFR is used to store the data to be compared with the content of the up-counter TCNT. This register can be directly read and written. (Default: indeterminate) But it is not suggested changing when counter is working in TCU2 mode.

TDFR0, TDFR1, TDFR2, TDFR3, TDFR4, TDFR5 — **0x10002040, 0x10002050, 0x10002060, 0x10002070, 0x10002080, 0x10002090**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TDFR															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

10.3.3 Timer Data HALF Register (TDHR)

The comparison data HALF registers TDHR is used to store the data to be compared with the content of the up-counter TCNT. This register can be directly read and written. (Default: indeterminate) But it is not suggested changing when counter is working in TCU2 mode.

TDHR0, TDHR1, TDHR2, TDHR3, TDHR4, TDHR5																0x10002044, 0x10002054, 0x10002064, 0x10002074, 0x10002084, 0x10002094
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TDHR															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

10.3.4 Timer Counter (TCNT)

TCNT is a 16-bit read/write register. The up-counter TCNT can be reset to 0 by software and counts up using the prescaler output clock. When TCNT count up to equal to TDFR, it will reset to 0 and continue to count up.

TCU1: The counter data can be read out at any time. The data can be written at any time. This makes it possible to change the interrupt and/or clock output cycles temporarily. (Default: indeterminate)

TCU2: The counter data can be read out at any time, but you should read TSTR.REALn to check whether the data is real data or not. The data can only be written before counter is started, and the counter clock is pclk. But it can be cleared to 0 by setting TCSR.CLRZ to 1, and if the counter is really cleared, TCSR.CLRZ will be set to 0 by hardware.

TCNT0, TCNT1, TCNT2, TCNT3, TCNT4, TCNT5																0x10002048, 0x10002058, 0x10002068, 0x10002078, 0x10002088, 0x10002098
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TCNT															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

10.3.5 Timer Counter Enable Register (TER)

The TER is a 16-bit read-only register. It contains the counter enable control bits for each channel. It is initialized to 0x0000 by any reset. It can only be set by register TESR and TECR. Since the timer enable control bits are located in the same addresses, two or more timers can be started at the same time.

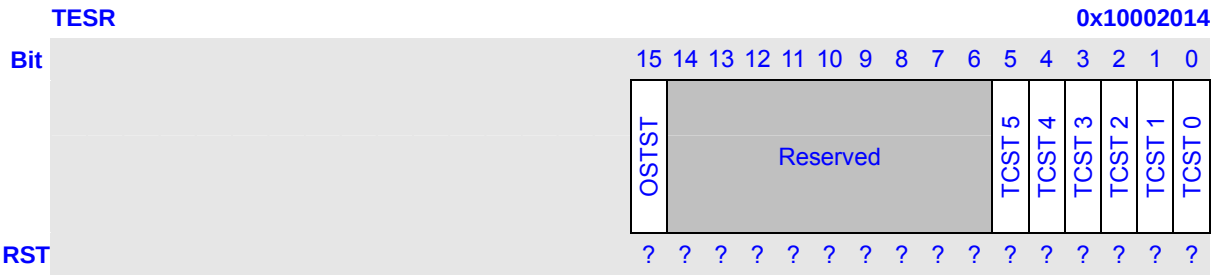
TER 0x10002010

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	OSTEN	Reserved									TCEN 5	TCEN 4	TCEN 3	TCEN 2	TCEN 1	TCEN 0
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15	OSTEN	Enable the counter in OST. 1: Begin counting up 0: Stop counting up	
14:6	Reserved	These bits always read 0, and written are ignored.	R
5	TCEN 5	Enable the counter in timer 5. 1: Begin counting up 0: Stop counting up	R
4	TCEN 4	Enable the counter in timer 4. 1: Begin counting up 0: Stop counting up	R
3	TCEN 3	Enable the counter in timer 3. 1: Begin counting up 0: Stop counting up	R
2	TCEN 2	Enable the counter in timer 2. 1: Begin counting up 0: Stop counting up	R
1	TCEN 1	Enable the counter in timer 1. 1: Begin counting up 0: Stop counting up	R
0	TCEN 0	Enable the counter in timer 0. 1: Begin counting up 0: Stop counting up	R

10.3.6 Timer Counter Enable Set Register (TESR)

The TCCSR is a 32-bit write-only register. It contains the counter enable set bits for each channel. Since the timer enable control set bits are located in the same addresses, two or more timers can be started at the same time.



Bits	Name	Description	RW
15	OSTST	Set OSTEN bit of TER. 1: Set OSTEN bit to 1 0: Ignore	W
14:6	Reserved	These bits always read 0, and written are ignored.	W
5	TCST 5	Set TCEN 5 bit of TER. 1: Set TCEN 5 bit to 1 0: Ignore	W
4	TCST 4	Set TCEN 4 bit of TER. 1: Set TCEN 4 bit to 1 0: Ignore	W
3	TCST 3	Set TCEN 3 bit of TER. 1: Set TCEN 3 bit to 1 0: Ignore	W
2	TCST 2	Set TCEN 2 bit of TER. 1: Set TCEN 2 bit to 1 0: Ignore	W
1	TCST 1	Set TCEN 1 bit of TER. 1: Set TCEN 1 bit to 1 0: Ignore	W
0	TCST 0	Set TCEN 0 bit of TER. 1: Set TCEN 0 bit to 1 0: Ignore	W

10.3.7 Timer Counter Enable Clear Register (TECR)

The TECR is a 32-bit write-only register. It contains the counter enable clear bits for each channel. Since the timer enable clear bits are located in the same addresses, two or more timers can be stop at the same time.

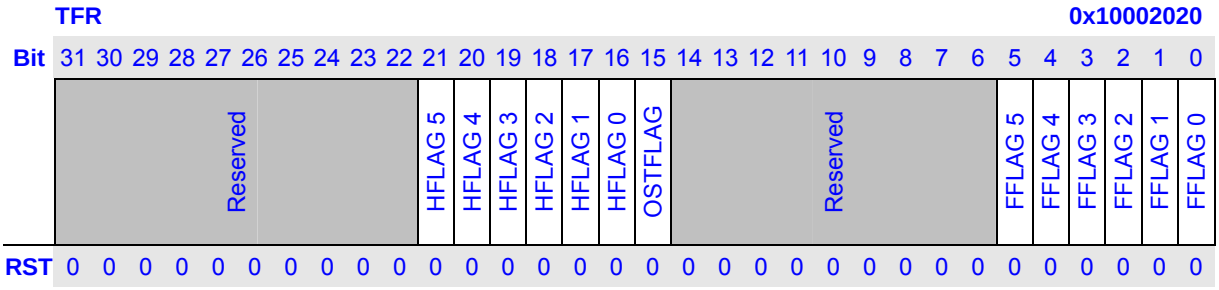
TECR **0x10002018**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	OSTCL	Reserved									TCCL 5	TCCL 4	TCCL 3	TCCL 2	TCCL 1	TCCL 0
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
15	OSTCL	Set OSTEN bit of TER. 1: Set OSTEN 5 bit to 0 0: Ignore	W
14:6	Reserved	These bits always read 0, and written are ignored.	W
5	TCCL 5	Set TCEN 5 bit of TER. 1: Set TCEN 5 bit to 0 0: Ignore	W
4	TCCL 4	Set TCEN 4 bit of TER. 1: Set TCEN 4 bit to 0 0: Ignore	W
3	TCCL 3	Set TCEN 3 bit of TER. 1: Set TCEN 3 bit to 0 0: Ignore	W
2	TCCL 2	Set TCEN 2 bit of TER. 1: Set TCEN 2 bit to 0 0: Ignore	W
1	TCCL 1	Set TCEN 1 bit of TER. 1: Set TCEN 1 bit to 0 0: Ignore	W
0	TCCL 0	Set TCEN 0 bit of TER. 1: Set TCEN 0 bit to 0 0: Ignore	W

10.3.8 Timer Flag Register (TFR)

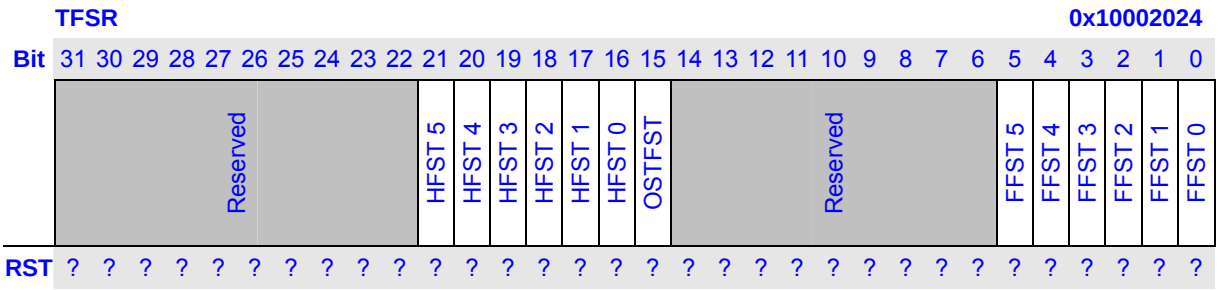
The TFR is a 32-bit read-only register. It contains the comparison match flag bits for all the channels. It can also be set by register TFSR and TFCR. It is initialized to 0x00000000 by any reset.



Bits	Name	Description	RW
31:22	Reserved	These bits always read 0, and written are ignored.	R
21:16	HFLAG 5~0	HALF comparison match flag. (TCNT = TDHR) 1: Comparison match 0: Comparison not match	R
15	OSTFLAG	OST comparison match flag. (OSTCNT = OSTDR) 1: Comparison match 0: Comparison not match	R
14:6	Reserved	These bits always read 0, and written are ignored.	R
5:0	FFLAG 5~0	FULL comparison match flag. (TCNT = TDFR) 1: Comparison match 0: Comparison not match	R

10.3.9 Timer Flag Set Register (TFSR)

The TFSR is a 32-bit write-only register. It contains the comparison match flag set bits for all the channels.



Bits	Name	Description	RW
31:22	Reserved		-
21:16	HFST 5~0	Set HFLAG n bit of TFR. 1: Set HFLAG n bit to 1	W

		0: Ignore	
15	OSTFST	Set OSTFLAG n bit of TFR. 1: Set OSTFLAG n bit to 1 0: Ignore	W
14:6	Reserved		-
5:0	FFST 5~0	Set FFLAG n bit of TFR. 1: Set FFLAG n bit to 1 0: Ignore	W

10.3.10 Timer Flag Clear Register (TFCR)

The TFCR is a 32-bit write-only register. It contains the comparison match flag clear bits for all the channels.

TFCR 0x10002028

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

Bits	Name	Description	RW
31:22	Reserved		-
21:16	HFCL 5~0	Set HFLAG n bit of TFR. 1: Set FFLAG n bit to 0 0: Ignore	W
15	OSTFCL	Set OSTFLAG n bit of TFR. 1: Set OSTFLAG n bit to 0 0: Ignore	W
14:6	Reserved		-
5:0	FFCL 5~0	Set FFLAG n bit of TFR. 1: Set FFLAG n bit to 0 0: Ignore	W

10.3.11 Timer Mask Register (TMR)

The TMR is a 32-bit read-only register. It contains the comparison match flag bits for all the channels. It is initialized to 0x003F003F by any reset. It can only be set by register TMSR and TMCR.

TMR 0x10002030

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

Bits	Name	Description	RW
31:22	Reserved	These bits always read 0, and written are ignored.	R
21:16	HMASK 5~0	HALF comparison match interrupt mask. 1: Comparison match interrupt mask 0: Comparison match interrupt not mask	R
15	OSTMASK	OST comparison match interrupt mask. 1: Comparison match interrupt mask 0: Comparison match interrupt not mask	R
14:6	Reserved	These bits always read 0, and written are ignored.	R
5:0	FMASK 5~0	FULL comparison match interrupt mask. 1: Comparison match interrupt mask 0: Comparison match interrupt not mask	R

10.3.12 Timer Mask Set Register (TMSR)

The TMSR is a 32-bit write-only register. It contains the comparison match flag set bits for all the channels.

TMSR 0x10002034

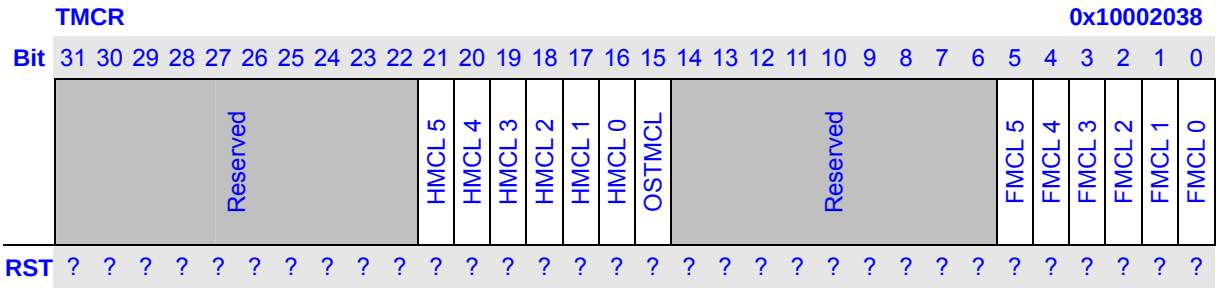
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved										HMST 5	HMST 4	HMST 3	HMST 2	HMST 1	HMST 0	OSTMST	Reserved									FMST 5	FMST 4	FMST 3	FMST 2	FMST 1	FMST 0
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
31:22	Reserved		-
21:16	HMST 5~0	Set HMASK n bit of TMR. 1: Set HMASK n bit to 1	W

		0: Ignore	
15	OSTMST	Set OSTMASK n bit of TMR. 1: Set OSTMASK n bit to 1 0: Ignore	W
14:6	Reserved		-
5:0	FMST 5~0	Set FMASK n bit of TMR. 1: Set FMASK n bit to 1 0: Ignore	W

10.3.13 Timer Mask Clear Register (TMCR)

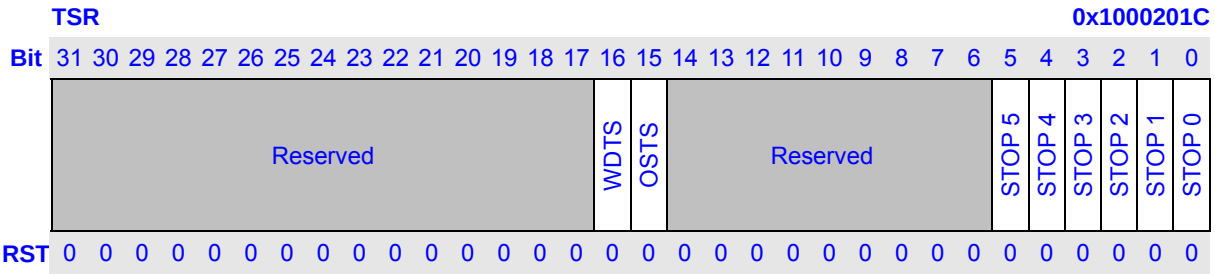
The TMCR is a 32-bit write-only register. It contains the comparison match flag clear bits for all the channels.



Bits	Name	Description	RW
31:22	Reserved		-
21:16	HMCL 5~0	Set HMASK n bit of TMR. 1: Set HMASK n bit to 0 0: Ignore	W
15	OSTMCL	Set OSTMASK n bit of TMR. 1: Set OSTMASK n bit to 0 0: Ignore	W
14:6	Reserved		-
5:0	FMCL 5~0	Set FMASK n bit of TMR. 1: Set FMASK n bit to 0 0: Ignore	W

10.3.14 Timer Stop Register (TSR)

The TSR is a 32-bit read-only register. It contains the timer stop control bits for each channel, WDT and OST. It is initialized to 0x00000000 by any reset. It can only be set by register TSSR and TSCR. If set, clock supplies to timer n / WDT / OST is stopped, and registers of the timer / WDT / OST cannot be accessed also.

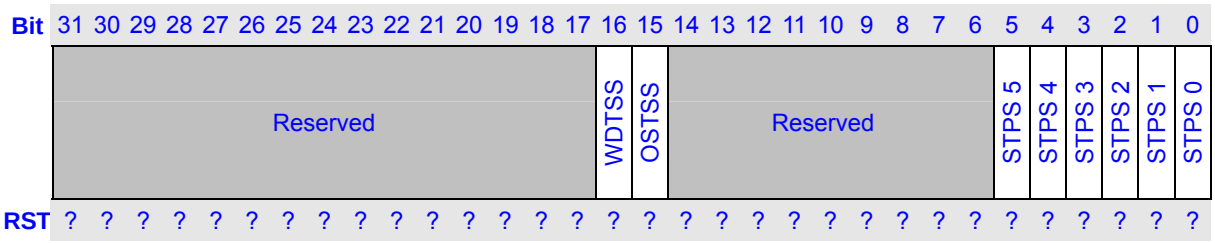


Bits	Name	Description	RW
31:17	Reserved	These bits always read 0, and written are ignored.	R
16	WDTS	1: The clock supplies to WDT is stopped 0: The clock supplies to WDT is supplied	R
15	OSTS	1: The clock supplies to OST is stopped 0: The clock supplies to OST is supplied	R
14:6	Reserved	These bits always read 0, and written are ignored.	R
5	STOP 5	1: The clock supplies to timer 5 is stopped 0: The clock supplies to timer 5 is supplied	R
4	STOP 4	1: The clock supplies to timer 4 is stopped 0: The clock supplies to timer 4 is supplied	R
3	STOP 3	1: The clock supplies to timer 3 is stopped 0: The clock supplies to timer 3 is supplied	R
2	STOP 2	1: The clock supplies to timer 2 is stopped 0: The clock supplies to timer 2 is supplied	R
1	STOP 1	1: The clock supplies to timer 1 is stopped 0: The clock supplies to timer 1 is supplied	R
0	STOP 0	1: The clock supplies to timer 0 is stopped 0: The clock supplies to timer 0 is supplied	R

10.3.15 Timer Stop Set Register (TSSR)

The TCSR is a 32-bit write-only register. It contains the timer stop set bits for each channel, WDT and OST. Since the timer stop control set bits are located in the same addresses, two or more timers can be started at the same time.

TSSR 0x1000202C



Bits	Name	Description	RW
31:17	Reserved		-
16	WDTSS	Set WDTS bit of TSR. 1: Set WDTS bit to 1 0: Ignore	W
15	OSTSS	Set OSTS bit of TSR. 1: Set OSTS bit to 1 0: Ignore	W
14:6	Reserved		-
5	STPS 5	Set STOP 5 bit of TSR. 1: Set STOP 5 bit to 1 0: Ignore	W
4	STPS 4	Set STOP 4 bit of TSR. 1: Set STOP 4 bit to 1 0: Ignore	W
3	STPS 3	Set STOP 3 bit of TSR. 1: Set STOP 3 bit to 1 0: Ignore	W
2	STPS 2	Set STOP 2 bit of TSR. 1: Set STOP 2 bit to 1 0: Ignore	W
1	STPS 1	Set STOP 1 bit of SR. 1: Set STOP 1 bit to 1 0: Ignore	W
0	STPS 0	Set STOP 0 bit of TSR. 1: Set STOP 0 bit to 1 0: Ignore	W

10.3.16 Timer Stop Clear Register (TSCR)

The TSCR is a 32-bit write-only register. It contains the timer stop clear bits for each channel, WDT and OST. Since the timer stop clear bits are located in the same addresses, two or more timers can be stop at the same time.

TSCR 0x1000203C

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

Bits	Name	Description	RW
31:17	Reserved		-
16	WDTSC	Set WDTS bit of TSR. 1: Set WDTS bit to 0 0: Ignore	W
15	OSTSC	Set OSTS bit of TSR. 1: Set OSTS bit to 0 0: Ignore	W
14:6	Reserved		-
5	STPC 5	Set STOP 5 bit of TSR. 1: Set STOP 5 bit to 0 0: Ignore	W
4	STPC 4	Set STOP 4 bit of TSR. 1: Set STOP 4 bit to 0 0: Ignore	W
3	STPC 3	Set STOP 3 bit of TSR. 1: Set STOP 3 bit to 0 0: Ignore	W
2	STPC 2	Set STOP 2 bit of TSR. 1: Set STOP 2 bit to 0 0: Ignore	W
1	STPC 1	Set STOP 1 bit of TSR. 1: Set STOP 1 bit to 0 0: Ignore	W
0	STPC 0	Set STOP 0 bit of TSR. 1: Set STOP 0 bit to 0 0: Ignore	W

10.3.17 Timer Status Register (TSTR)

The TSTR is a 32-bit read-only register. It contains the status of channel in TCU2 mode. The register can be written by setting register TSTSR and TSTCR.

TSR 0x100020F0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved										REAL5	Reserved															BUSY 5	Reserved				
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:22	Reserved	These bits always read 0, and written are ignored.	R
21	REAL 5	1: The value read from counter 5 is a real value 0: The value read from counter 5 is a false value	R
20:6	Reserved	These bits always read 0, and written are ignored.	R
5	BUSY 5	1: The counter 5 is busy now 0: The counter 5 is ready now	R
4:0	Reserved	These bits always read 0, and written are ignored.	R

10.3.18 Timer Status Set Register (TSTSR)

The TSTSR is a 32-bit write-only register. It contains the timer status set bits for each channel.

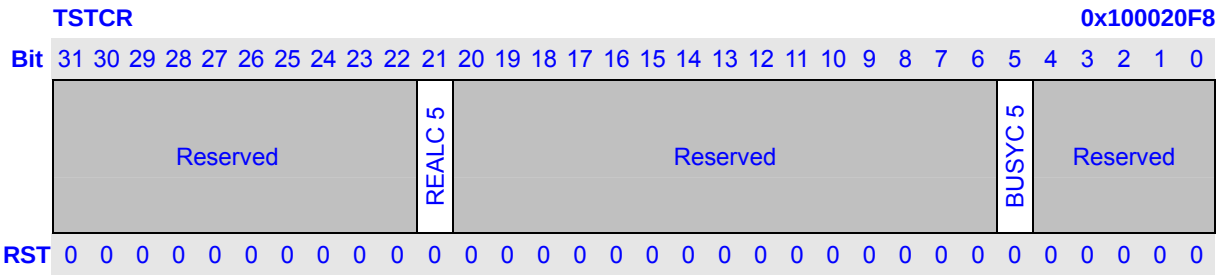
TSTSR 0x100020F4

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

Bits	Name	Description	RW
31:22	Reserved	These bits always read 0, and written are ignored.	R
21	REALS 5	Set REAL 5 bit of TSTR. 1: Set REAL 5 bit to 1 0: Ignore	R
20:6	Reserved	These bits always read 0, and written are ignored.	R
5	BUSYS 5	Set BUSY 5 bit of TSTR. 1: Set BUSY 5 bit to 1 0: Ignore	R
4:0	Reserved	These bits always read 0, and written are ignored.	R

10.3.19 Timer Status Clear Register (TSTCR)

The TSTCR is a 32-bit write-only register. It contains the timer status clear bits for each channel.



Bits	Name	Description	RW
31:22	Reserved	These bits always read 0, and written are ignored.	R
21	REALC 5	Clear REAL 5 bit of TSTR. 1: Clear REAL 5 bit to 1 0: Ignore	R
20:6	Reserved	These bits always read 0, and written are ignored.	R
5	BUSYC 5	Clear BUSY 5 bit of TSTR. 1: Clear BUSY 5 bit to 1 0: Ignore	R
4:0	Reserved	These bits always read 0, and written are ignored.	R

10.4 Operation

10.4.1 Basic Operation in TCU1 Mode

The value of TDFR should be bigger than TDHR, and the minimum settings are TDHR = 0 and TDFR = 1. In this case, the timer output clock cycle is the input clock × 1/2. If TDHR > TDFR, no comparison TFHR signal is generated.

Before the timer counter begins to count up, we need to do as follows:

If you want to use PWM you should set TCSR.PWM_EN to be 0 before you initial TCU.

1 Initial the configuration.
   a Writing TCSR.INITL to initialize PWM output level.
   b Writing TCSR.SD to setting the shutdown mode (Abrupt shutdown or Graceful shutdown).
   c Writing TCSR.PRESCALE to set TCNT count clock frequency.
   d Setting TCNT, TDHR and TDFR.

2 Enable the clock.
   a Writing TCSR.PWM_EN to set whether enable PWM or disable PWM.
   b Writing TCSR.EXT_EN, TCSR.RTC_EN or TCSR.PCK_EN to 1 to select the input clock and enable the input clock. Only one of TCSR.EXT_EN, TCSR.RTC_EN and TCSR.PCK_EN can be set to 1.

After initialize the register of timer, we should start the counter as follows:

3 Enable the counter.
   Setting the TESR.TCST bit to 1 to enable the TCNT.

NOTE: The input clock and PCLK should follow the rules advanced before.

10.4.2 Disable and Shutdown Operation in TCU1 Mode

1 Setting the TECR.TCCL bit to 1 to disable the TCNT.

10.4.3 Basic Operation in TCU2 Mode

The value of TDFR should be bigger than TDHR, and the minimum settings are TDHR = 0 and TDFR = 1. In this case, the timer output clock cycle is the input clock × 1/2. If TDHR > TDFR, no comparison TFHR signal is generated.

Initial state is that TCSR.PRESCALE=0, TCSR.PWM_EN=0 and TCENR=0.

1 Reset the TCU.
   a Writing TCSR.PCK_EN to 1 to select pclk as the input clock.
   b Set TCSR.CLRZ to 1 to clear TCNT or set TCNT to an initial value.
   c Writing TCSR.PCK_EN to 0 to close the input clock.

2 Initial the configuration.
   a Setting TDHR and TDFR.
   b Writing TCSR.INITL to initialize PWM output level (if used PWM).
   c Writing TCSR.PRESCALE to set TCNT count clock frequency.
   d Writing TCSR.EXT_EN, TCSR.RTC_EN or TCSR.PCK_EN to 1 to select the input clock and enable the input clock. Only one of TCSR.EXT_EN, TCSR.RTC_EN and TCSR.PCK_EN can be set to 1.
   e Writing TCSR.PWM_EN to set whether enable PWM or disable PWM.

After initialize the register of timer, we should start the counter as follows:

3 Setting the TESR.TCST bit to 1 to enable the TCNT.

NOTE:

You can clear the counter when counter is working.

1 Set TCSR.CLRZ to 1 to clear TCNT.
2 Wait till TSTR.BUSY = 0, that is the counter have been cleared.

You can enable PWM or disable PWM the counter when counter is working.

1 Set TCSR.PWM_EN to 1 to enable PWM.
2 Set TCSR.PWM_EN to 0 to disable PWM.

10.4.4 Disable and Shutdown Operation in TCU2 Mode

1 Writing TCSR.PWM_EN to 0 to disable PWM.
2 Setting the TECR.TCCL bit to 1 to disable the TCNT.
3 Wait till TSTR.BUSY = 0, that is the reset of counter is finished.

10.4.5 Read Counter in TCU2 Mode

If you want to read the data from register TCNT when the TCU is working, you can check TSTR.REAL whether it is good or not. It is suggested that:

1 If TSTR.REAL==1, the data read is available.
2 If TSTR.REAL==0, reread the counter till TSTR.REAL==1, the data read is available.
3 If TSTR.REAL is always 0, you can read some data, and lose some data that is quick different from the others. Then choose a data from them as the available data.

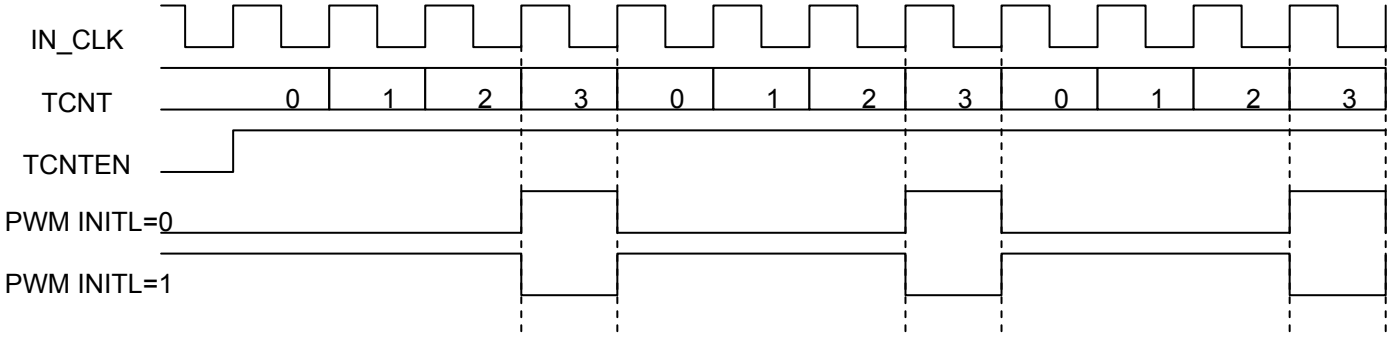
NOTES:

1 It suggested that (1), (2) is often used when the counter clock is very slow.
2 It suggested that (3) is often used when the counter clock is very fast.

10.4.6 Pulse Width Modulator (PWM)

Timer 0~5 can be used as Pulse Width Modulator (PWM). The PWM can be used to control the back light inverter or adjust bright or contrast of LCD panel.

FULL comparison match signal and HALF comparison match signal can determine an attribute of the PWM_OUT waveform. FULL comparison match signal specifies the clock cycle for the PWM module clock. HALF comparison match signal specifies the duty ratio for the PWM module clock.



11 Operating System Timer

11.1 Overview

The OST (Operating System Timer) contains one 32-bit programmable timer. It can be used as operating system timer.

OST has the following features:

- OST includes
  - 32-bit Counter
  - 32-bit Compare Data Register
  - Control Register
- Independent clock for each counter, selectable by software
  - PCLK, EXTAL and RTCCLK can be used as the clock for counter
  - The division ratio of the clock can be set to 1, 4, 16, 64, 256 and 1024 by software
- Match interrupt can be generated for OST using the compare data registers
  - Interrupt flag and interrupt mask is same with TCU in TCU spec

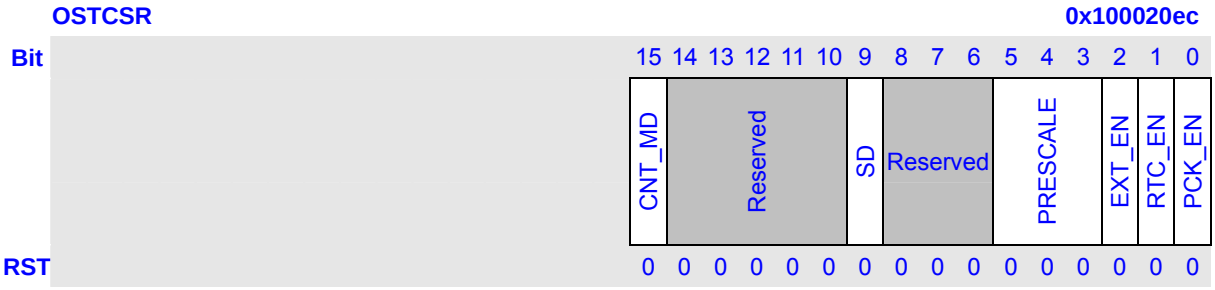
11.2 Register Description

In this section, we will describe the registers in OST. Following table lists all the registers definition. All OST register's 32bit address is physical address. And detailed function of each register will be described below.

Name	Description	RW	Reset Value	Address	Access Size
OSTDR	Operating System Timer Data Register	RW	0x????????	0x100020e0	32
OSTCNT	Operating System Timer Counter	RW	0x????????	0x100020e8	32
OSTCSR	Operating System Timer Control Register	RW	0x0000	0x100020ec	16

11.2.1 Operating System Control Register (OSTCSR)

The TCSR is a 16-bit read/write register. It contains the control bits for OST. It is initialized to 0x00 by any reset.



Bits	Name	Description	RW
15	CNT_MD	Counter mode choose bit. 0: When the value counter is equal to compare value, the counter will be cleared, and increase from 0 1: When the value counter is equal to compare value, the counter will go on increasing till overflow, and then increase from 0	
14:6	Reserved	These bits always read 0, and written are ignored.	R
9	SD	Shut Down (SD) the PWM output. It is only used in TCU1 mode. 0: Graceful shutdown (only used when CNT_MD = 0) 1: Abrupt shutdown	RW
5:3	PRESCALE	These bits select the TCNT count clock frequency.	RW

Bit 2	Bit1	Bit 0	Description
0	0	0	Internal clock: CLK/1
0	0	1	Internal clock: CLK/4
0	1	0	Internal clock: CLK/16
0	1	1	Internal clock: CLK/64

		1	0	0	Internal clock: CLK/256	
		1	0	1	Internal clock: CLK/1024	
		110~111			Reserved	
2	EXT_EN	Select EXTAL as the timer clock input. 1: Enable 0: Disable				RW
1	RTC_EN	Select RTCCLK as the timer clock input. 1: Enable 0: Disable				RW
0	PCK_EN	Select PCLK as the timer clock input. 1: Enable 0: Disable				RW

NOTES:

1 The input clock of timer and the PCLK should keep to the rules as follows:

Input clock of timer: IN_CLK	**Clock generated from the frequency divider (PRESCALE): DIV_CLK**
PCK_EN == 0, RTC_EN == 1 and EXT_EN == 0 (IN_CLK = RTCCLK)	$f_{DIV\_CLK} < ½ f_{PCLK}$
PCK_EN == 0, RTC_EN == 0 and EXT_EN == 1 (IN_CLK = EXTAL)	$f_{DIV\_CLK} < ½ f_{PCLK}$
PCK_EN == 1, RTC_EN == 0 and EXT_EN == 0 (IN_CLK = PCLK)	ANY

11.2.2 Operating System Timer Data Register (OSTDR)

The operating system timer data register OSTDR is used to store the data to be compared with the content of the operating system timer up-counter OSTCNT. This register can be directly read and written. (Default: indeterminate)

OSTDR **0x100020e0**

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

11.2.3 Operating System Timer Counter (OSTCNT)

The operating system timer counter (OSTCNT) is a 32-bit read/write counter. The up-counter OSTCNT can be set by software and counts up using the prescaler output clock. The data can be read out at any time. The counter data can be written at any time. (Default: indeterminate)

OSTCNT **0x100020e8**

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

11.3 Operation

11.3.1 Basic Operation

Before the timer counter begins to count up, we need to do as follows:

1 Initial the configuration.
   a Writing TCSR.SD to setting the shutdown mode (Abrupt shutdown or Graceful shutdown).
   b Writing OSTCSR.PRESCALE to set OSTCNT count clock frequency.
   c Setting OSTCNT and OSTDR.

2 Enable the clock.
   Writing OSTCSR.EXT_EN, OSTCSR.RTC_EN or OSTCSR.PCK_EN to 1 to select the input clock and enable the input clock. Only one of OSTCSR.EXT_EN, OSTCSR.RTC_EN and OSTCSR.PCK_EN can be set to 1.

After initialize the register of timer, we should start the counter as follows:

3 Enable the counter.
   Setting the TESR.OSTCST bit to 1 to enable the OSTCNT.

NOTES:

1 The input clock and PCLK should follow the rules advanced before.

11.3.2 Disable and Shutdown Operation

1 Setting the TECR.OSTCCL bit to 1 to disable the OSTCNT.

12 Watchdog Timer

12.1 Overview

The watchdog timer is used to resume the processor whenever it is disturbed by malfunctions such as noise and system errors. The watchdog timer can generate the reset signal.

Features:

- Generates WDT reset
- A 16-bit Data register and a 16-bit counter
- Counter clock uses the input clock selected by software
  - PCLK, EXTAL and RTCCLK can be used as the clock for counter
  - The division ratio of the clock can be set to 1, 4, 16, 64, 256 and 1024 by software

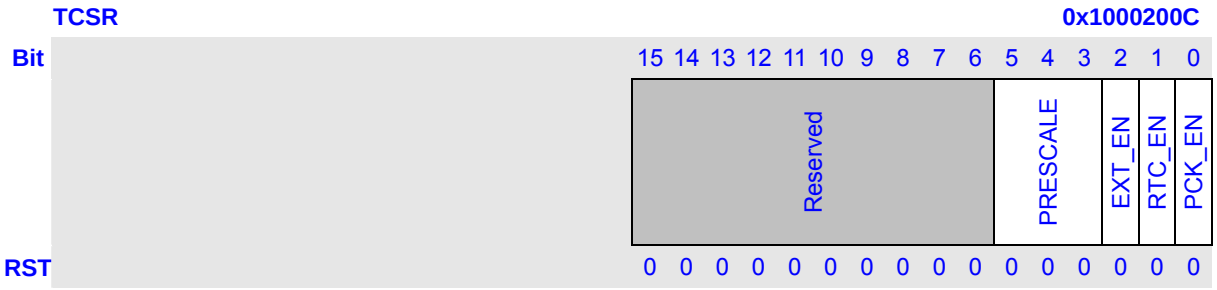
12.2 Register Description

In this section, we will describe the registers in WDT. Following table lists all the registers definition. All WDT register's 32bit address is physical address. And detailed function of each register will be described below.

Name	Description	RW	Reset Value	Address	Access Size
TDR	Watchdog Timer Data Register	RW	0x????	0x10002000	16
TCER	Watchdog Counter Enable Register	RW	0x00	0x10002004	8
TCNT	Watchdog Timer Counter	RW	0x????	0x10002008	16
TCSR	Watchdog Timer Control Register	RW	0x0000	0x1000200C	16

12.2.1 Watchdog Control Register (TCSR)

The TCSR is a 16-bit read/write register. It contains the control bits for WDT. It is initialized to 0x00 by any reset.



Bits	Name	Description	RW
15:6	Reserved	These bits always read 0, and written are ignored.	R
5:3	PRESCALE	These bits select the TCNT count clock frequency.	RW
2	EXT_EN	Select EXTAL as the timer clock input. 1: Enable 0: Disable	RW
1	RTC_EN	Select RTCCLK as the timer clock input. 1: Enable	RW

PRESCALE:

Bit 2	Bit1	Bit 0	Description
0	0	0	Internal clock: CLK/1
0	0	1	Internal clock: CLK/4
0	1	0	Internal clock: CLK/16
0	1	1	Internal clock: CLK/64
1	0	0	Internal clock: CLK/256
1	0	1	Internal clock: CLK/1024
110~111			Reserved

		0: Disable	
0	PCK_EN	Select PCLK as the timer clock input. 1: Enable 0: Disable	RW

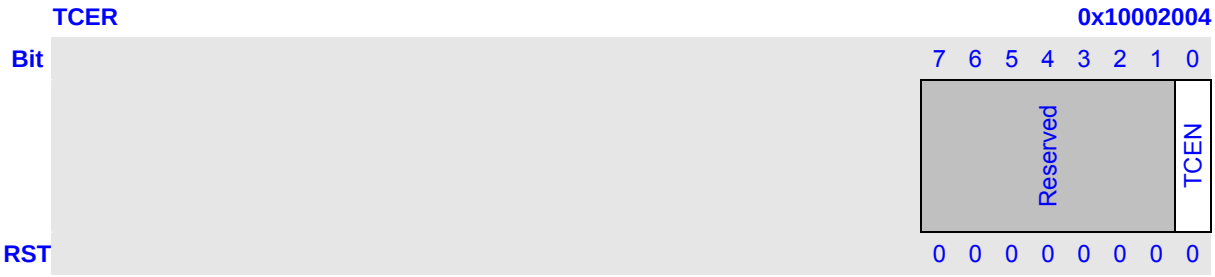
NOTES:

1 The input clock of timer and the PCLK should keep to the rules as follows:

Input clock of timer: IN_CLK	**Clock generated from the frequency divider (PRESCALE): DIV_CLK**
PCK_EN == 0, RTC_EN == 1 and EXT_EN == 0 (IN_CLK = RTCCLK)	$f_{DIV\_CLK} < ½ f_{PCLK}$
PCK_EN == 0, RTC_EN == 0 and EXT_EN == 1 (IN_CLK = EXTAL)	$f_{DIV\_CLK} < ½ f_{PCLK}$
PCK_EN == 1, RTC_EN == 0 and EXT_EN == 0 (IN_CLK = PCLK)	ANY

12.2.2 Watchdog Enable Register (TCER)

The TCER is an 8-bit read/write register. It contains the counter enable control bits for watchdog. It is initialized to 0x00 by any reset.



Bits	Name	Description	RW
7:1	Reserved	These bits always read 0, and written are ignored.	R
0	TCEN	Counter enable control. 0: Timer stop 1: Timer running	RW

12.2.3 Watchdog Timer Data Register (TDR)

The watchdog timer data register TDR is used to store the data to be compared with the content of the watchdog timer up-counter TCNT. This register can be directly read and written. (Default: indeterminate)

TDR																0x10002000
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TDR															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

12.2.4 Watchdog Timer Counter (TCNT)

The watchdog timer counter (TCNT) is a 16-bit read/write counter. The up-counter TCNT can be reset to 0 by software and counts up using the prescaler output clock. When TCNT count up to equal to TDR, the comparison match signal will be generated and a WDT reset is generated. The data can be read out at any time. The counter data can be written at any time. (Default: indeterminate)

TCNT																0x10002008
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TCNT															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

12.3 Watchdog Timer Function

The following describes steps of using WDT:

1 Setting the PRESCALE of input clock in register TCSR.
2 Set register TDR and TCNT.
3 Select the input clock and enable the input clock in register TCSR.

After initialize the register of timer, we should start the counter as follows:

4 Set TCEN bit in TCER to 1. The counter TCNT begins to count.
5 If TCNT = TDR, a WDT reset will be generated.

NOTES:

1 The input clock and PCLK should follow the rules advanced before.
2 The clock of WDT can be stopped by setting register TSR, and register TSR can only be set by register TSSR or TSCR. The content of register TSR, TSSR and TSCR can be found in TCU spec.

13 LCD Controller

13.1 Overview

The JZ integrated LCD controller has the capabilities to driving the latest industry standard STN and TFT LCD panels. It also supports some special TFT panels used in consuming electronic products. The controller performs the basic memory based frame buffer and palette buffer to LCD panel data transfer through use of a dedicated DMA controller. Temporal dithering (frame rate modulation) is supported for STN LCD panels. And OSD is also supported for LCD controller.

Features:

- Basic Features
  - Support CCIR601/656 data format
  - Single and Dual panel displays in STN mode
  - Single panel displays in TFT mode
  - Display size up to 800x600
  - Internal palette RAM 256x16 bits

- Colors Supports
  - Encoded pixel data of 1, 2, 4, 8 or 16 BPP in STN mode
  - Support 2, 4, 16 grayscales and up to 4096 colors in STN mode
  - Encoded pixel data of 1, 2, 4, 8, 16, 18 or 24 BPP in TFT mode
  - Support 65,536(65K), 262,144(260K) and up to 16,777,216 (16M) colors in TFT mode

- Panel Supports
  - Support single STN panel and dual STN panel with 1, 2, 4, 8 data output pins
  - Support 16-bit parallel TFT panel
  - Support 18-bit parallel TFT panel
  - Support 24-bit serial TFT panel with 8 data output pins
  - Support 24-bit parallel TFT panel (22 pin correspond to RED[7:1], GREEN[7:0], BLUE[7:1])
  - Support Delta RGB panel

- OSD Supports
  - Supports one single color background
  - Supports two foregrounds, and every size can be set for each foreground
  - Supports one transparency for the whole graphic
  - Supports one transparency for each pixel in one graphic
  - Supports color key and mask color key

13.2 Pin Description

Table 13-1 LCD Controller Pins Description

Name	I/O	Description
Lcd_pclk	Input/Output	Display device pixel clock
Lcd_vsync	Input/Output	Display device vertical synchronize pulse
Lcd_hsync	Input/Output	Display device horizontal synchronize pulse
Lcd_de	Output	Display device is STN: AC BIAS Pin Display device is NOT STN: data enable Pin
Lcd_d[17:0]	Output	Display device data pins
lcd_lo6_o[5:0]	Output	Display device data pins use in 24 bit parallel mode.
Lcd_spl*[1]	Output	Programmable special pin for generating control signals
Lcd_cls*[1]	Output	Programmable special pin for generating control signals
Lcd_ps*[1]	Output	Programmable special pin for generating control signals
Lcd_rev*[1]	Output	Programmable special pin for generating control signals

NOTE: The mode and timing of special pin Lcd_spl, Lcd_cls, Lcd_ps and Lcd_rev can be seen in **part 1.7 LCD Controller Pin Mapping.**

13.3 Block Diagram

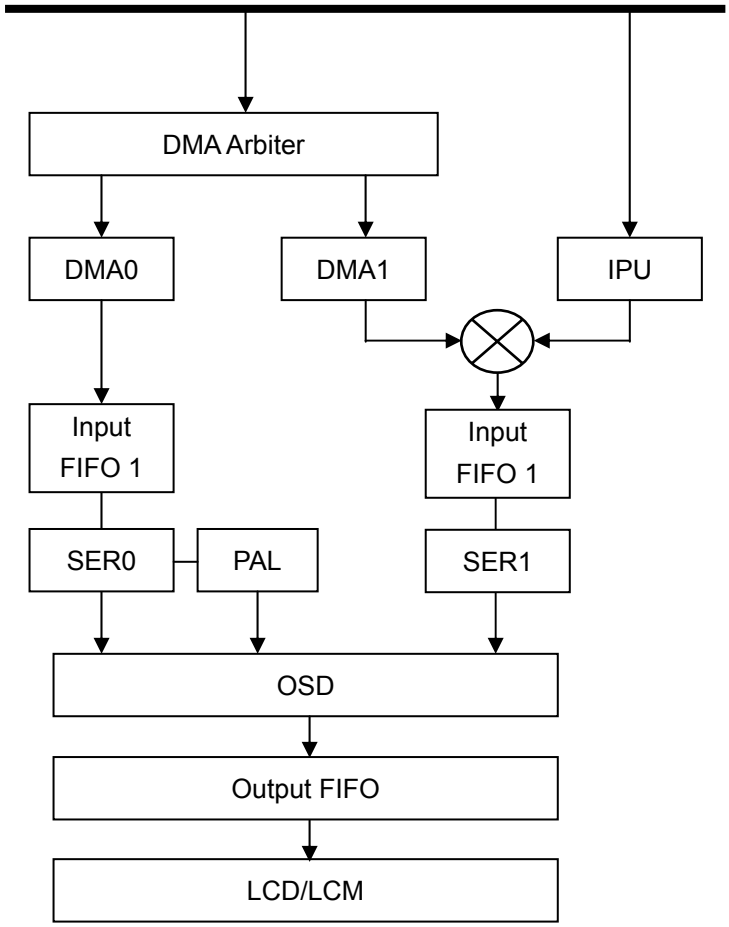


Figure 13-1 Block Diagram when use OSD mode

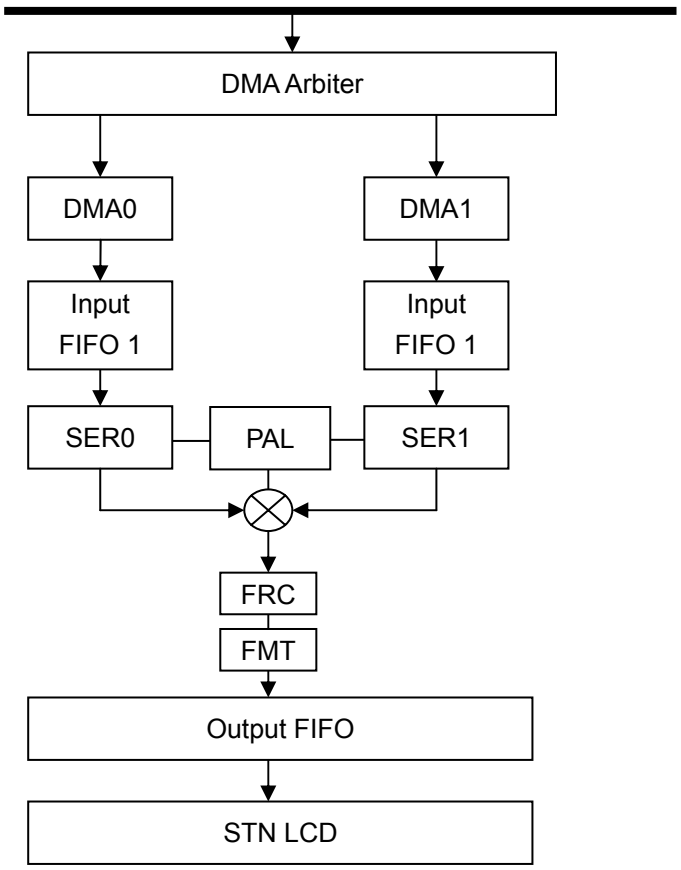


Figure 13-2 Block Diagram of STN mode (not use OSD)

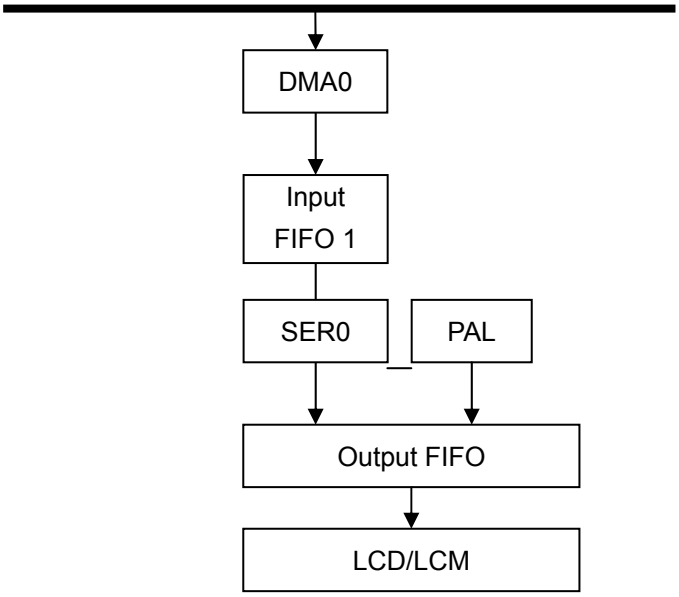


Figure 13-3 Block Diagram of TFT mode (not use OSD)

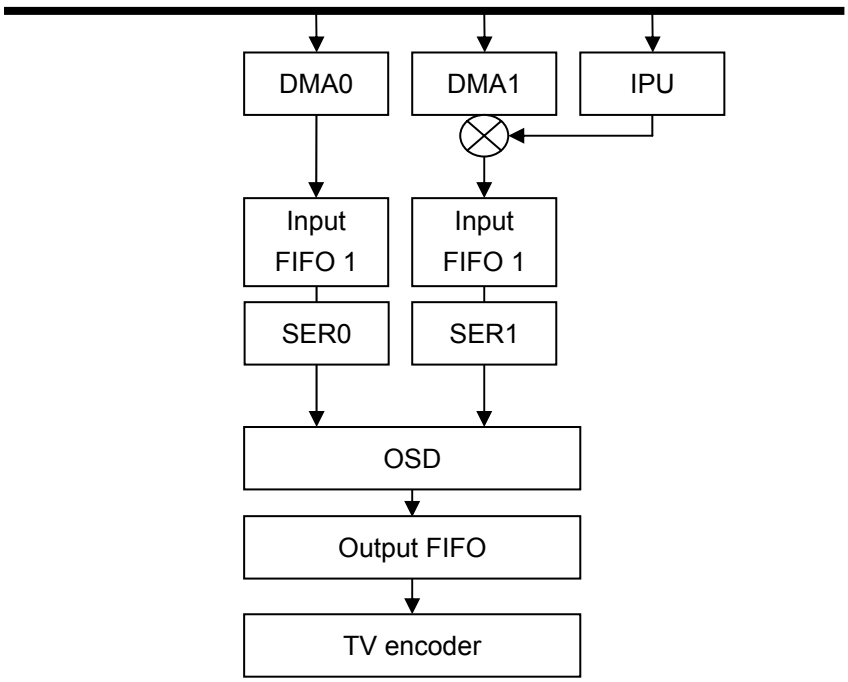


Figure 13-4 Block Diagram of TV interface

13.4 LCD Display Timing

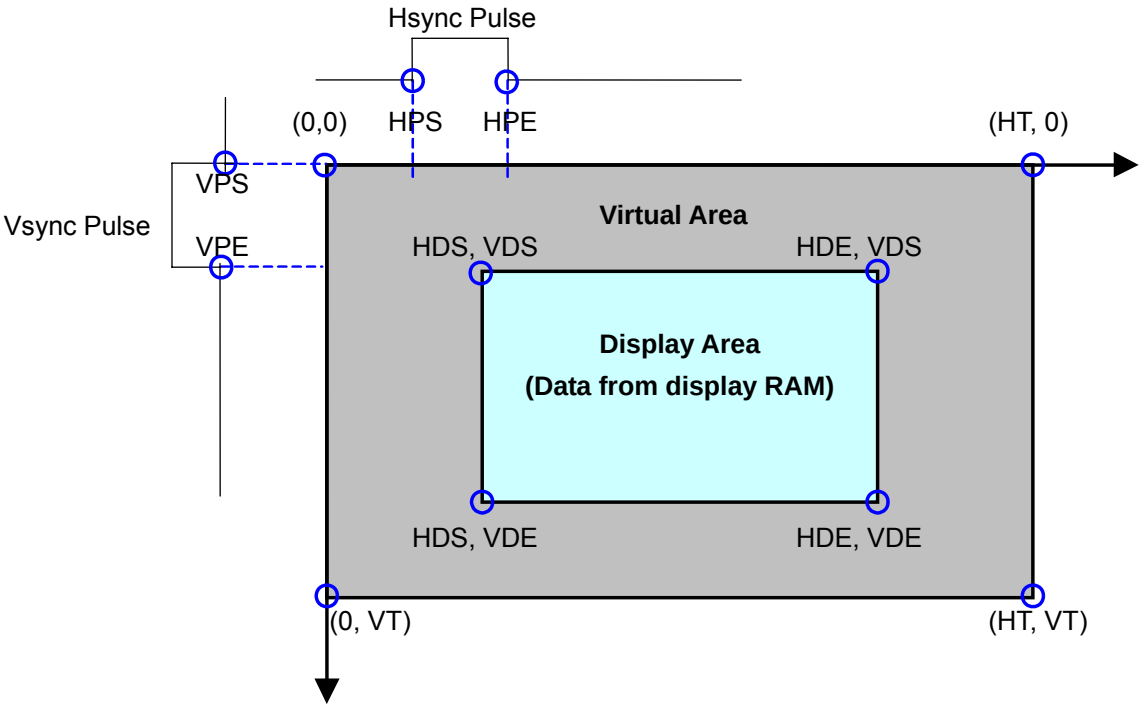


Figure 13-5 Display Parameters

NOTES:

1 VPS === 0
VSYNC pulse always start at point (0,0)

2 H: Horizontal V: Vertical T: Total
D: Display Area P: Pulse
S: Start point E: End point

In the (H, V) Coordinates:

1 The gray rectangle (0, 0) to (HT, VT) is “Virtual Area”.
2 The blue rectangle (HDS, VDS) to (HDE, VDE) is “Display Area”.
3 VPS, VPE defines the VSYNC signal timing. (VPS always be zero)
4 HPS, HPE defines the HSYNC signal timing.

All timing parameters start with “H” is measured in lcd_pclk ticks.
All timing parameters start with “V” is measured in lcd_hsync ticks.

This diagram describes the general LCD panel parameters, these can be set via the registers that describes in next section.

13.5 OSD Graphic

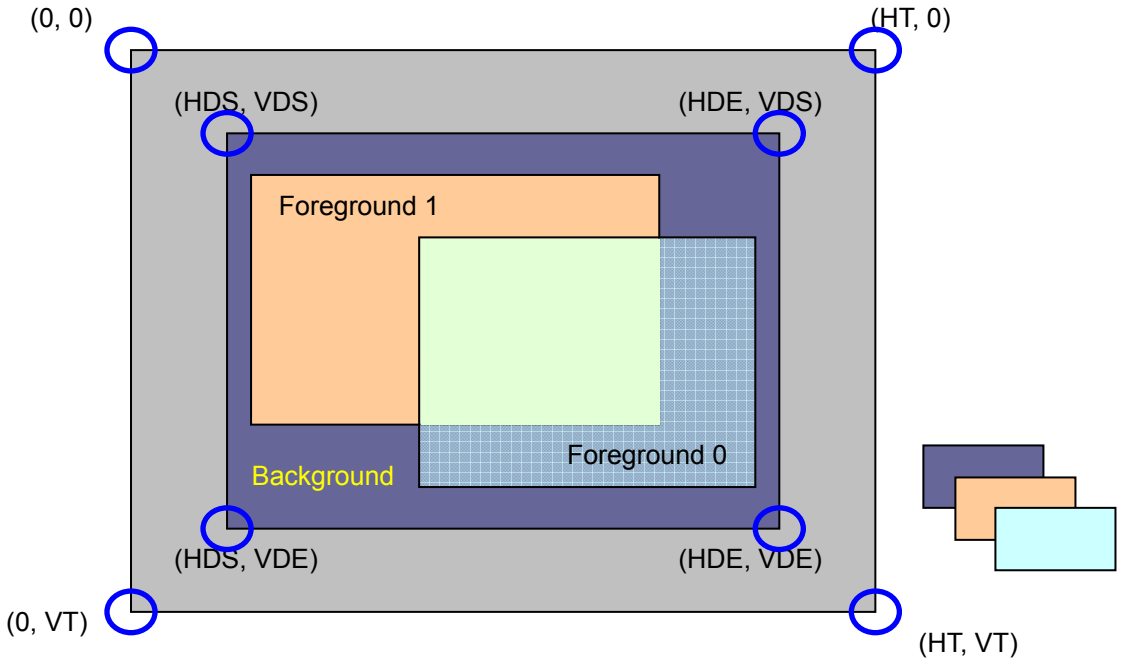


Figure 13-6 OSD Graphic

NOTES:

1 Background is one single color and the size is the full screen.

2 The size of foregrounds can be every size smaller than background.

3 The order of the graphic is as follows:

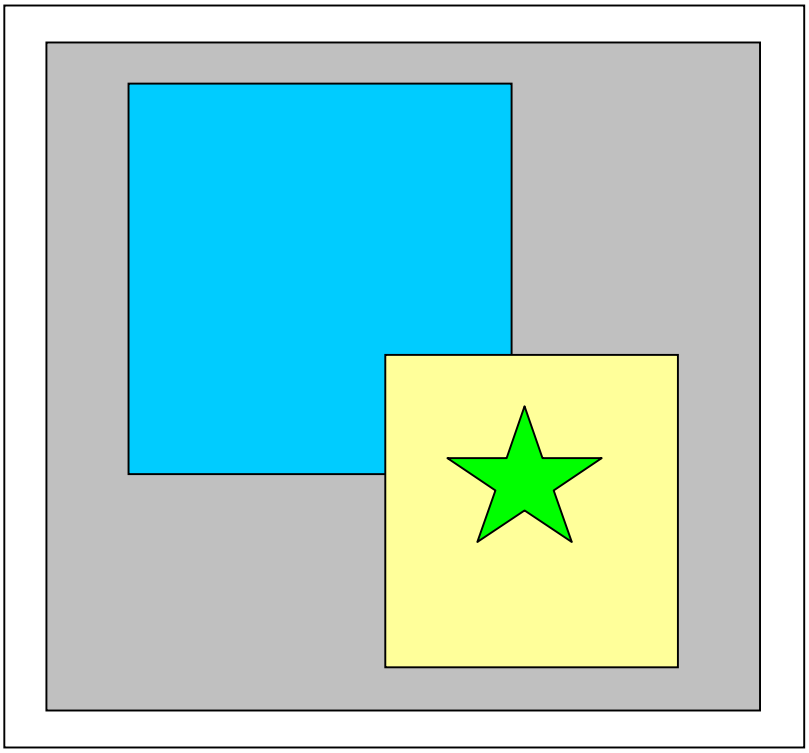
- a Top layer: Foreground 0
- b Middle layer: Foreground 1
- c Bottom layer: Background

13.5.1 Color Key

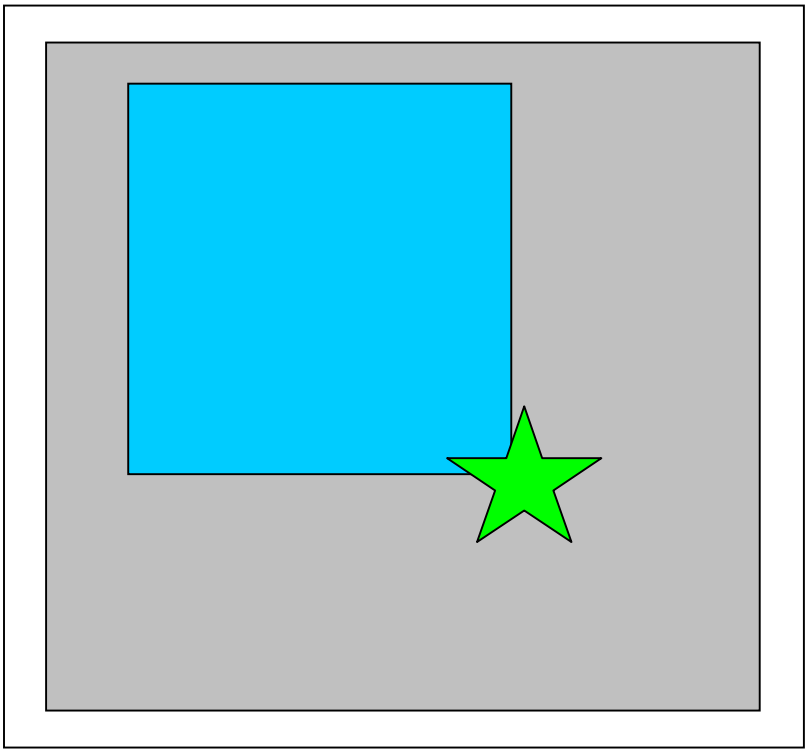
This function gives user a method to implement irregular display window. User can make foreground 0 and foreground 1 to different shape. The color key has two implements mode that called color key and mask color key.

Color Key mode is mean to mask a chosen color and show others.

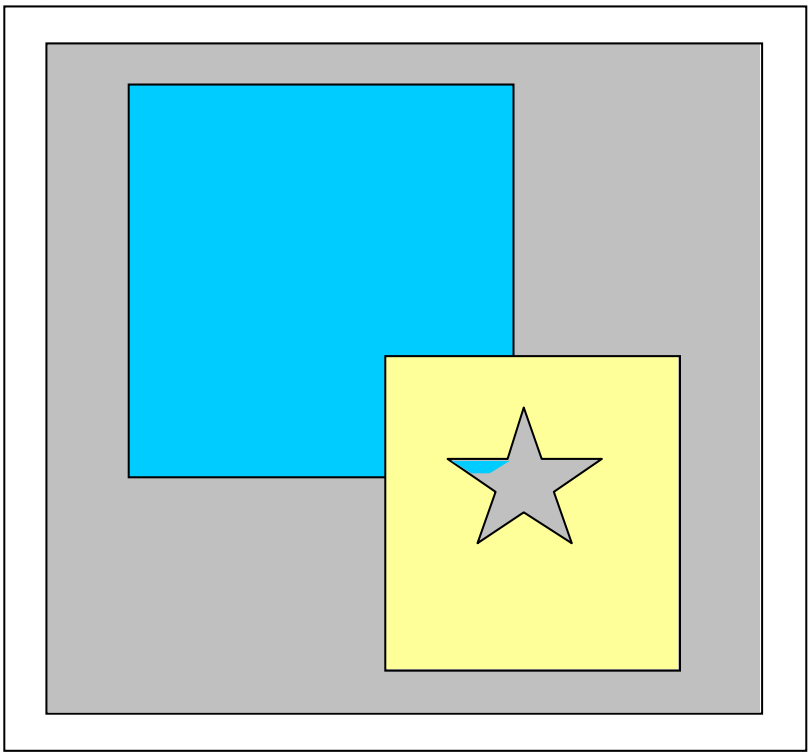
Mask Color Key mode is mean to only show a chosen color and mask others.



Not use color key function



Color key mode



Mask color key mode

13.6 Register Description

Table 13-2 LCD Controller Registers Description

Name	RW	Reset Value	Address	Access Size
LCDCFG	RW	0x00000000	0x13050000	32
LCDCTRL	RW	0x00000000	0x13050030	32
LCDSTATE	RW	0x00000000	0x13050034	32
LCDOSDC	RW	0x0000	0x13050100	16
LCDOSDCTRL	RW	0x0000	0x13050104	16
LCDOSDS	RW	0x0000	0x13050108	16
LCDBGC	RW	0x00000000	0x1305010C	32
LCDKEY0	RW	0x00000000	0x13050110	32
LCDKEY1	RW	0x00000000	0x13050114	32
LCDALPHA	RW	0x00	0x13050118	8
LCDIPUR	RW	0x00000000	0x1305011C	32
LCDRGBC	RW	0x0000	0x13050090	16
LCDVAT	RW	0x00000000	0x1305000C	32
LCDDAH	RW	0x00000000	0x13050010	32
LCDDAV	RW	0x00000000	0x13050014	32
LCDXYP0	RW	0x00000000	0x13050120	32
LCDXYP1	RW	0x00000000	0x13050124	32
LCDSIZE0	RW	0x00000000	0x13050128	32
LCDSIZE1	RW	0x00000000	0x1305012C	32
LCDVSYNC	RW	0x00000000	0x13050004	32
LCDHSYNC	RW	0x00000000	0x13050008	32
LCDPS[*1]	RW	0x00000000	0x13050018	32
LCDCLS[*1]	RW	0x00000000	0x1305001C	32
LCDSPL[*1]	RW	0x00000000	0x13050020	32
LCDREV[*1]	RW	0x00000000	0x13050024	32
LCDIID	R	0x00000000	0x13050038	32
LCDDA0	RW	0x00000000	0x13050040	32
LCDSA0	R	0x00000000	0x13050044	32
LCDFID0	R	0x00000000	0x13050048	32
LCDCMD0	R	0x00000000	0x1305004C	32
LCDOFFS0	R	0x00000000	0x13050060	32
LCDPW0	R	0x00000000	0x13050064	32
LCDCNUM0	R	0x00000000	0x13050068	32
LCDDESSIZE0	R	0x00000000	0x1305006C	32
LCDDA1[*2]	RW	0x00000000	0x13050050	32
LCDSA1[*2]	R	0x00000000	0x13050054	32

LCDFID1[*2]	R	0x00000000	0x13050058	32
LCDCMD1[*2]	R	0x00000000	0x1305005C	32
LCDOFFS1[*2]	R	0x00000000	0x13050070	32
LCDPW1[*2]	R	0x00000000	0x13050074	32
LCDCNUM1[*2]	R	0x00000000	0x13050078	32
LCDDESSIZE1[*2]	R	0x00000000	0x1305007C	32
LCDXYP0_PART2	RW	0x00000000	0x130501F0	32
LCDSIZE0_PART2	RW	0x00000000	0x130501F4	32
LCDDA0_PART2	RW	0x00000000	0x130501C0	32
LCDSA0_PART2	R	0x00000000	0x130501C4	32
LCDFID0_PART2	R	0x00000000	0x130501C8	32
LCDCMD0_PART2	R	0x00000000	0x130501CC	32
LCDOFFS0_PART2	R	0x00000000	0x130501E0	32
LCDPW0_PART2	R	0x00000000	0x130501E4	32
LCDCNUM0_PART2[*3]	R	0x00000000	0x130501E8	32
LCDDESSIZE0_PART2	R	0x00000000	0x130501EC	32

NOTES:

1 [*1]: These registers are only used for SPECIAL TFT panels.

2 [*2]: These registers are only used for Dual Panel STN panels and use DMA channel 1 in OSD mode for TFT panels.

3 [*3]: Not use in these version, set to 0.

13.6.1 Configure Register (LCDCFG)

LCDCFG **0x13050000**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	LCDPIN			NEWDES	PALBP		RECOVER	DITHER	PSM	CLSM	SPLM	REVM	HSYNM	PCLKM	INVDAT	SYNDIR	PSP	CLSP	SPLP	REVP	HSP	PCP	DEP	VSP	18/16	24	PDW		MODE			
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31	LCDPIN[*1]	LCD PIN Select bit. It is used to choose the function of LCD PINS or SLCD PINS. The function of pins is as follows: **LCDPIN** / **PIN SELECT** 0 / LCD PIN 1 / SLCD PIN	RW
30		**KEEP THIS BIT TO 0.**	RW
29		**KEEP THIS BIT TO 0.**	RW

28	NEWDES	indicate use new 8 words descriptor or not. 0: use old 4 words descriptor 1: use new 8 words descriptor (add LCDOFFSx, LCDPWx, LCDCUNMx, LCDDESSIZEx) OSD mode use 8 word descriptor.	RW
27	PALBP	Indicate bypass pal in BPP8, and in OSD mode, set this bit to 1 is also bypass data format and alpha blending. 0: use PAL; 1: not use PAL.	RW
26		**KEEP THIS BIT TO 0.**	RW
25	RECOVER	Auto recover when output FIFO under run. 0: disable, 1: enable.	RW
24	DITHER	Dither function. (use when 24bpp data output to a 18/16bit panel) 0: disable, 1: enable. Dither function use to make the picture misty, when you show a static picture with few color, strongly recommend you not use it. When you use this function both static and dynamic picture, strongly recommend you to set the static picture with 16/18BPP color.	RW
23	PSM	PS signal mode bit. 0: enabled; 1:disabled.	RW
22	CLSM	CLS signal mode bit. 0: enabled; 1: disabled.	RW
21	SPLM	SPL signal mode bit. 0: enabled; 1: disabled.	RW
20	REVM	REV signal mode bit. 0: enabled; 1: disabled.	RW
19	HSYNM	H-Sync signal polarity choice function. 0: enabled; 1: disabled.	RW
18	PCLKM	Dot clock signal polarity choice function. 0: enabled; 1: disabled.	RW
17	INVDAT	Inverse output data. 0: normal; 1: inverse.	RW
16	SYNDIR	V-Sync and H-Sync direction. 0: output; 1: input.	RW
15	PSP	PS pin reset state.	RW
14	CLSP	CLS pin reset state.	RW
13	SPLP	SPL pin reset state.	RW
12	REVP	REV pin reset state.	RW
11	HSP	H-Sync polarity. 0: active high; 1: active low.	RW
10	PCP	Pix-clock polarity. 0: data translations at rising edge 1: data translations at falling edge	RW
9	DEP	Data Enable polarity. 0: active high; 1: active low.	RW
8	VSP	V-Sync polarity. 0: leading edge is rising edge 1: leading edge is falling edge	RW
7	18/16	18-bit TFT Panel or 16-bit TFT Panel. This bit will be available when MODE [3:2] is equal to 0 and 24[6] is equal to 0. 0: 16-bit TFT Panel 1: 18-bit TFT Panel	RW
6	24	Set this bit to 1 for 24-bit TFT Panel.	RW

5:4	PDW	STN pins utilization. **Signal Panel** 00 Lcd_d[0] 01 Lcd_d[0:1] 10 Lcd_d[0:3] 11 Lcd_d[0:7] **Dual-Monochrome Panel** 00 Reserved 01 Reserved 10 Upper panel: lcd_d[3:0], lower panel: lcd_d[11:8] 11 Upper panel: lcd_d[7:0], lower panel: lcd_d[15:8]	RW
3:0	MODE	Display Device Mode Select/Output mode. **LCD Panel** 0000 Generic 16-bit/18-bit Parallel TFT Panel 0001 Special TFT Panel Mode1 0010 Special TFT Panel Mode2 0011 Special TFT Panel Mode3 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000 Single-Color STN Panel 1001 Single-Monochrome STN Panel 1010 Dual-Color STN Panel 1011 Dual-Monochrome STN Panel 1100 8-bit Serial TFT 1101 LCM 1110 Reserved 1111 Reserved	RW

NOTES:

***1:**

LCDPIN	PIN25	PIN24	PIN23	PIN22	PIN21	PIN20	PIN19	PIN18	PIN17-0
0	LCD PCLK	LCD VSYNC	LCD HSYNC	LCD DE	LCD REV	LCD PS	LCD CLS	LCD SPL	LCD D [17:0]
1	SLCD CLK	SLCD CS	SLCD RS						SLCD D [17:0]

1 The direction of PIN25 is set by register LPCDR.LCS in CPM SPEC.

2 The direction of PIN23 and PIN23 are set by register LCDCFG.SYNDIR.

13.6.2 Control Register (LCDCTRL)

LCDCTRL 0x13050030

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PINMD	Reserved	BST		RGB	OFUP	FRC		PDD										EOFM	SOFM	OFUM	IFUM0	IFUM1	LDDM	QDM	BEDN	PEDN	DIS	ENA	BPP		
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31	PINMD	This register set Pin distribution in 16-bit parallel mode. 0: 16-bit data correspond with LCD_D[15:0] 1: 16-bit data correspond with LCD_D[17:10], LCD_D[8:1]	RW
30	Reserved		RW
29:28	BST	Burst Length Selection. 00: 4 word 01: 8 word 10: 16 word 11: reserved	RW
27	RGB	Bpp16 RGB mode. 0: RGB565; 1: RGB555. In OSD mode, this bit configure the foreground 0. If use parallel 18 bit，set this bit to 0.	RW
26	OFUP	Output FIFO under run protection. 0: disable; 1: enable.	RW
25:24	FRC	STN FRC Algorithm Selection. 00: 16 grayscale 01: 4 grayscale 10: 2 grayscale 11: Reserved	RW
23:16	PDD	Load Palette Delay Counter.	RW
15		**KEEP THIS BIT TO 0.**	
14		**KEEP THIS BIT TO 0.**	R
13	EOFM	Mask end of frame interrupt. 0: INT-disabled; 1:INT-enabled.	RW
12	SOFM	Mask start of frame interrupt. 0: INT-disabled; 1: INT-enabled.	RW
11	OFUM	Mask out FIFO under run interrupt. 0: INT-disabled; 1: INT-enabled.	RW
10	IFUM0	Mask in FIFO 0 under run interrupt. 0: INT-disabled; 1: INT-enabled.	RW
9	IFUM1	Mask in FIFO 1 under run interrupt. 0: INT-disabled; 1: INT-enabled.	RW
8	LDDM	Mask LCD disable done interrupt. 0: INT-disabled; 1: INT-enabled.	RW
7	QDM	Mask LCD quick disable done interrupt. 0: INT-disabled; 1: INT-enabled.	RW

6	BEDN	Endian selection. 0: same as system Endian; 1: reverse endian format.	RW
5	PEDN	Endian in byte. 0: msb first; 1: lsb first.	RW
4	DIS	Disable controller indicate bit. 0: enable; 1: in disabling or disabled.	RW
3	ENA	Enable controller. 0: disable; 1: enable.	W
2:0	BPP	Bits Per Pixel. In OSD mode, those bits configure the foreground 0.	RW

	Bits Per Pixel
000	1bpp
001	2bpp
010	4bpp
011	8bpp
100	15/16bpp
101	18bpp/24bpp
110	24bpp compressed
111	Reserved

13.6.3 Status Register (LCDSTATE)

LCDSTATE **0x13050034**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																									QD	Reserved	EOF	SOF	OUF	IFU0	IFU1	LDD
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7	QD	LCD Quick disable. 0: not been quick disabled; 1: quick disabled done.	RW
6	Reserved	These bits always read 0, and written are ignored.	R
5	EOF	End of Frame indicate bit.	RW
4	SOF	Start of Frame indicate bit.	RW
3	OUF	Out FIFO under run.	RW
2	IFU0	In FIFO 0 under run.	RW
1	IFU1	In FIFO 1 under run.	RW
0	LDD	LCD disable. 0: not been normal disabled; 1: been normal disabled.	RW

13.6.4 OSD Configure Register (LCDOSDC)

LCDOSDC **0x13050100**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SOFM1	EOFM1	Reserved		SOFM0	EOFM0	ENDM	F0DIVMD	F0P1EN	F0P2MD	F0P2EN	F1EN	F0EN	ALPHAEN	ALPHAMD	OSDEN
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15	SOFM1	Start of frame interrupt mask for foreground 1.	RW
14	EOFM1	End of frame interrupt mask for foreground 1.	RW
13:12	Reserved	These bits always read 0, and written are ignored.	R
11	SOFM0	Start of frame interrupt mask for foreground 0.	RW
10	EOFM0	End of frame interrupt mask for foreground 0.	RW
9	ENDM	End of frame interrupt mask for panel.	RW
8	F0DIVMD	1: Divide Foreground 0 into 2 parts 0: Foreground 0 only has one part	RW
7	F0P1EN	1: Foreground 0 PART1 is enabled 0: Foreground 0 PART1 is disabled	RW
6	F0P2MD	1: Foreground 0 PART1 PART2 with only one descriptor	RW
5	F0P2EN	1: Foreground 0 PART2 is enabled 0: Foreground 0 PART2 is disabled	RW
4	F1EN	1: Foreground 1 is enabled 0: Foreground 1 is disabled	RW
3	F0EN	1: Foreground 0 is enabled 0: Foreground 0 is disabled.*When use slcd, F0EN must set 1	RW
2	ALPHAEN	1: Alpha blending is enabled 0: Alpha blending is disabled	RW
1	ALPHAMD	Alpha blending mode. 0: One transparency for the whole graphic, and the LCDALPHA register is used for transparency 1: One transparency for each pixel in one graphic, and the alpha value is coming from each pixel data	RW
0	OSDEN	OSD mod enable. 1: enabled. And you can use F0 F1 0: disabled	RW

13.6.5 OSD Control Register (LCDOSDCTRL)

LCDOSDCTRL **0x13050104**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	IPU	Reserved										RGB	CHANGES	OSDBPP		
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15	IPU	Indicate use IPU or DMA channel 1 to transport data to FIFO 1. This bit is only use in OSD mode. 0: use DMA channel 1 1: use IPU	RW
14:5	Reserved	These bits always read 0, and written are ignored.	
4	OSDRGB	Bpp16 RGB mode. 0: RGB565; 1: RGB555. This bit only use in OSD mode to configure foreground 1.	RW
3	CHANGES	Change configure flag, when software need change the foreground0 and foreground1's enable/position/size, it need set this bit to 1. When hardware finishes the change, It will clear this bit to 0. DO NOT set this bit when you needed change size or position. AND make sure the reconfigure value is different to the old one. Only one of these (F0EN, F1EN, F0's position, F1's position, F0's size, F1's size, F0P1EN, F0P2EN, F0 part2's position, F0 part2's size) could be change in one time. Refer to 1.12.6.	RW
2:0	OSDBPP	Bits Per Pixel of OSD channel 1(this channel cannot use palette). Those bits only use in OSD mode to configure display window 1.	RW

	Bits Per Pixel
000	Reserved
001	Reserved
010	Reserved
011	Reserved
100	15/16bpp
101	18bpp/24bpp
110	24bpp compressed
111	Reserved

13.6.6 OSD State Register (LCDOSDS)

LCDOSDS **0x13050108**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SOF1	EOF1	Reserved		SOF0	EOF0	Reserved	DEND	Reserved							READY
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15	SOF1	Start of frame flag for foreground 1.	RW
14	EOF1	End of frame flag for foreground 1.	RW
13:12	Reserved	These bits always read 0, and written are ignored.	R
11	SOF0	Start of frame flag for foreground 0.	RW
10	EOF0	End of frame flag for foreground 0.	RW
9	Reserved	These bits always read 0, and written are ignored.	R
8	DEND	Display end flag. mean a whole frame data are send to panel.	RW
7:1	Reserved	These bits always read 0, and written are ignored.	R
0	READY	Ready for accept the change. When this bit set 1, the software can change the descriptor's LCDDESSIZE0, 1 to change the foreground size. This bit will clear by hardware when the change is finished.	R

13.6.7 Background Color Register (LCDBGC)

LCDBGC **0x1305010C**

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

Bits	Name	Description	RW
31:27	Reserved	These bits always read 0, and written are ignored.	R
23:16	Red	Red part or Y part of background.	RW
15:8	Green	Green part or Cb part of background.	RW
7:0	Blue	Blue part or Cr part of background.	RW

13.6.8 Foreground Color Key Register 0 (LCDKEY0)

LCDKEY0 **0x13050110**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	KEYEN	KEYMD	Reserved						Red [7:0]								Green [7:0]								Blue [7:0]							
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31	KEYEN	The enable bit of color key for foreground 0.	RW
30	KEYMD	Color key mod. 0: color key; 1: mask color key.	RW
29:27	Reserved	These bits always read 0, and written are ignored.	R
23:16	Red	Red part of color key for foreground 0.	RW
15:8	Green	Green part of color key for foreground 0.	RW
7:0	Blue	Blue part of color key for foreground 0.	RW

13.6.9 Foreground Color Key Register 1 (LCDKEY1)

LCDKEY1 **0x13050114**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	KEYEN	KEYMD	Reserved						Red [7:0]								Green [7:0]								Blue [7:0]							
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31	KEYEN	The enable bit of color key for foreground 1.	RW
30	KEYMD	Color key mod. 0: color key; 1: mask color key.	RW
29:27	Reserved	These bits always read 0, and written are ignored.	R
23:16	Red	Red part of color key for foreground 1.	RW
15:8	Green	Green part of color key for foreground 1.	RW
7:0	Blue	Blue part of color key for foreground 1.	RW

13.6.10 ALPHA Register (LCDALPHA)

LCDALPHA **0x13050118**

Bit	7	6	5	4	3	2	1	0
	ALPHA [7:0]							
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7:0	ALPHA	The alpha value for one graphic with one transparency.	RW

The formula of alpha blending is as follows:

$$NewPixel = \frac{[(256 - Alpha) * (Foreground1\_or\_background) + Alpha * Froeground0 + 128]}{256}$$

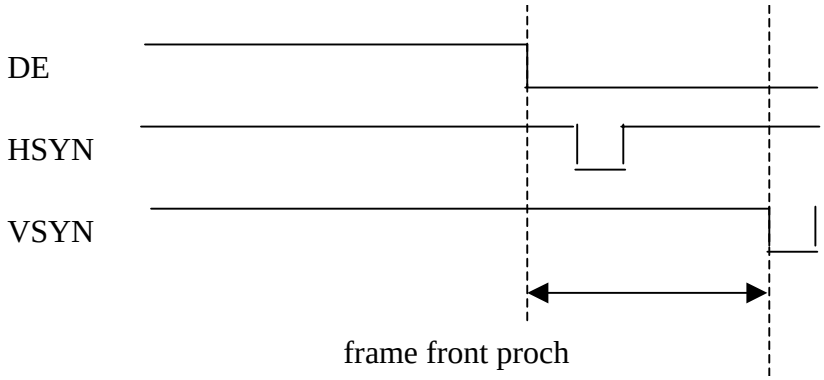
Note that foreground 1 must be overlay background.

13.6.11 IPU Restart (LCDIPUR)

LCDIPUR **0x1305011C**

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

Bits	Name	Description	RW
31	IPUREN	IPU restart function enable. 0:disable; 1:enable.	RW
30:24	Reserved	These bits always read 0, and written are ignored.	RW
23:0	IPUR	This register is indicating when one frame is end, how long the panel can wait for the next frame data from IPU. In common, set this number larger than frame front porch and near to ((HT-0) X (VPE-VPS))/3. This signal only use when foreground1 work in IPU mode. Trigger IPU transfer the last frame again to avoid output FIFO under run.	RW



13.6.12 RGB Control (LCDRGBC)

LCDRGBC **0x13050090**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RGBDM	DMM	Reserved					YCC	Reserved	OddRGB			Reserved	EvenRGB		
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15	RGBDM	RGB with dummy data enable. Only useful for RGB serial mode. If this bit set to 1, the one pixel include 4 clock periods, that Red, Green, Blue and Dummy data. Dummy is equal to 0. 0: Disable; 1: Enable.	RW
14	DMM	RGB dummy mode. 0: R-G-B-Dummy 1: Dummy-R-G-B	
13:9	Reserved	These bits always read 0, and written are ignored.	RW
8	YCC	Change RGB to YCbCr. 0: not change; 1: change to YUV. This bit only use in OSD mode. Change RGB data to YCbYCr and sent to TV encoder. Please notice that the data will be translated as 16 bits parallel. And only half of it will be transfer. (YCb or YCr in one pixel). If you not use OSD mode and TV encoder, please set this bit to 0. When use this function with IPU transfer data to an interlaced TV, please set IPU output as RGB 888, and OSDBPP to 24. or IPU output data as PACKAGE(YCbYCr) and OSDBPP to 16.	RW
7	Reserved	These bits always read 0, and written are ignored.	RW
6:4	OddRGB	Odd line serial RGB data arrangement, useful for RGB serial mode	RW

<table>
<tr><td></td><td></td><td>only. *Please notice that you must set 000 when use 16/18parallel mode.
<table><tr><th></th><th>RGB mode</th></tr><tr><td>000</td><td>RGB</td></tr><tr><td>001</td><td>RBG</td></tr><tr><td>010</td><td>GRB</td></tr><tr><td>011</td><td>GBR</td></tr><tr><td>100</td><td>BRG</td></tr><tr><td>101</td><td>BGR</td></tr><tr><td>110</td><td>Reserved</td></tr><tr><td>111</td><td>Reserved</td></tr></table></td><td></td></tr>
<tr><td>3</td><td>Reserved</td><td>These bits always read 0, and written are ignored.</td><td>RW</td></tr>
<tr><td>2:0</td><td>EvenRGB</td><td>Even line serial RGB data arrangement, useful for RGB serial mode only. *Please notice that you must set 000 when use 16/18parallel mode.
<table><tr><th></th><th>RGB mode</th></tr><tr><td>000</td><td>RGB</td></tr><tr><td>001</td><td>RBG</td></tr><tr><td>010</td><td>GRB</td></tr><tr><td>011</td><td>GBR</td></tr><tr><td>100</td><td>BRG</td></tr><tr><td>101</td><td>BGR</td></tr><tr><td>110</td><td>Reserved</td></tr><tr><td>111</td><td>Reserved</td></tr></table></td><td>RW</td></tr>
</table>

13.6.13 Virtual Area Setting (LCDVAT)

LCDVAT **0x1305000C**

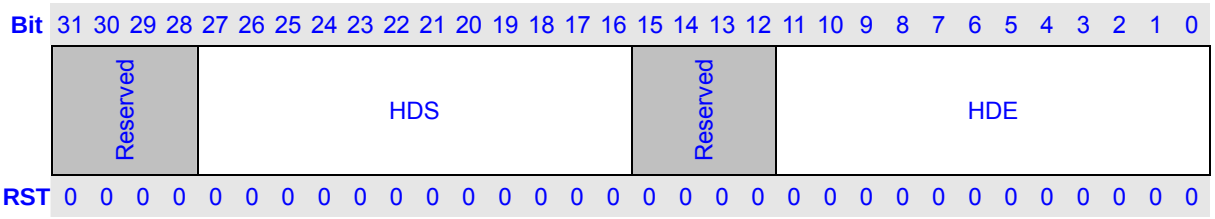
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved				HT												Reserved				VT											
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:28	Reserved	These bits always read 0, and written are ignored.	R
27:16	HT	Horizontal Total size. (in dot clock, sum of display area and blank space)	RW
15:12	Reserved	These bits always read 0, and written are ignored.	R

11:0	VT	Vertical Total size. (in line clock, sum of display area and blank space)	RW

13.6.14 Display Area Horizontal Start/End Point (LCDDAH)

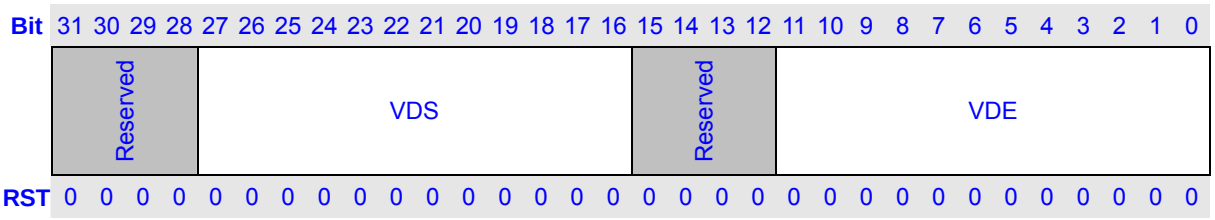
LCDDAH **0x13050010**



Bits	Name	Description	RW
31:28	Reserved	These bits always read 0, and written are ignored.	R
27:16	HDS	Horizontal display area start. (in dot clock)	RW
15:12	Reserved	These bits always read 0, and written are ignored.	R
11:0	HDE	Horizontal display area end. (in dot clock)	RW

13.6.15 Display Area Vertical Start/End Point (LCDDAV)

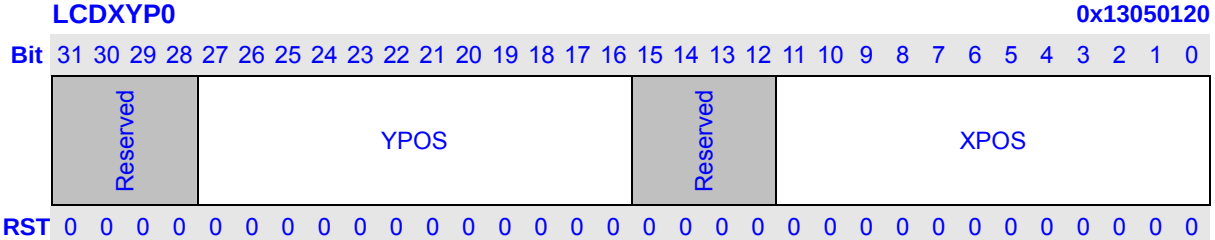
LCDDAV **0x13050014**



Bits	Name	Description	RW
31:28	Reserved	These bits always read 0, and written are ignored.	R
27:16	VDS	Vertical display area start position. (in line clock)	RW
15:12	Reserved	These bits always read 0, and written are ignored.	R
11:0	VDE	Vertical display area end position. (in line clock)	RW

13.6.16 Foreground 0 XY Position Register (LCDXYP0)

LCDXYP0 **0x13050120**



Bits	Name	Description	RW
31:28	Reserved	These bits always read 0, and written are ignored.	R
27:16	YPOS	The Y position of top-left part for foreground 0.	RW
15:12	Reserved	These bits always read 0, and written are ignored.	R
11:0	XPOS	The X position of top-left part for foreground 0.	RW

13.6.17 Foreground 0 PART2 XY Position Register (LCDXYP0_PART2)

LCDXYP0_PART2 **0x130501F0**

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

Bits	Name	Description	RW
31:28	Reserved	These bits always read 0, and written are ignored.	R
27:16	YPOS	The Y position of top-left part for foreground 0 PART2.	RW
15:12	Reserved	These bits always read 0, and written are ignored.	R
11:0	XPOS	The X position of top-left part for foreground 0 PART2.	RW

13.6.18 Foreground 1 XY Position Register (LCDXYP1)

LCDXYP1 **0x13050124**

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

Bits	Name	Description	RW
31:28	Reserved	These bits always read 0, and written are ignored.	R
27:16	YPOS	The Y position of top-left part for foreground 1.	RW
15:12	Reserved	These bits always read 0, and written are ignored.	R
11:0	XPOS	The X position of top-left part for foreground 1.	RW

13.6.19 Foreground 0 Size Register (LCDSIZE0)

LCDSIZE0 **0x13050128**

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

Bits	Name	Description	RW
31:28	Reserved	These bits always read 0, and written are ignored.	R
27:16	Height	The height of foreground 0.	RW
15:12	Reserved	These bits always read 0, and written are ignored.	R
11:0	Width	The width of foreground 0.	RW

When use TVE interlaced mode, please set the area of F0 and F1 aligned with BST.

13.6.20 Foreground 0 PART2 Size Register (LCDSIZE0_PART2)

LCDSIZE0_PART2 **0x130501F4**

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

Bits	Name	Description	RW
31:28	Reserved	These bits always read 0, and written are ignored.	R
27:16	Height	The height of foreground 0 PART2.	RW
15:12	Reserved	These bits always read 0, and written are ignored.	R
11:0	Width	The width of foreground 0 PART2.	RW

13.6.21 Foreground 1 Size Register (LCDSIZE1)

LCDSIZE1 **0x1305012C**

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

Bits	Name	Description	RW
31:28	Reserved	These bits always read 0, and written are ignored.	R
27:16	Height	The height of foreground 1.	RW
15:12	Reserved	These bits always read 0, and written are ignored.	R
11:0	Width	The width of foreground 1.	RW

13.6.22 Vertical Synchronize Register (LCDVSYNC)

LCDVSYNC **0x13050004**

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

Bits	Name	Description	RW
31:28	Reserved	These bits always read 0, and written are ignored.	R
27:16	VPS	V-Sync Pulse start position, fixed to 0. (in line clock)	R
15:12	Reserved	These bits always read 0, and written are ignored.	R
11:0	VPE	V-Sync Pulse end position. (in line clock)	RW

13.6.23 Horizontal Synchronize Register (LCDHSYNC)

LCDHSYNC **0x13050008**

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

Bits	Name	Description	RW
31:28	Reserved	These bits always read 0, and written are ignored.	R
27:16	HPS	H-Sync pulse start position. (in dot clock)	RW
15:12	Reserved	These bits always read 0, and written are ignored.	R
11:0	HPE	H-Sync pulse end position. (in dot clock)	RW

13.6.24 PS Signal Setting (LCDPS)

LCDPS **0x13050018**

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

Bits	Name	Description	RW
31:28	Reserved	These bits always read 0, and written are ignored.	R
27:16	PSS	PS signal start position. (in dot clock) In STN mode, PS signal is ignored. But this register is used to define the AC BIAS signal. AC BIAS signal will toggle very N lines per frame. PSS defines the Toggle position.	RW
15:12	Reserved	These bits always read 0, and written are ignored.	R
11:0	PSE	PS signal end position. (in dot clock) In STN mode, PSE defines N, which described in PSS.	RW

13.6.25 CLS Signal Setting (LCDCLS)

LCDCLS **0x1305001C**

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

Bits	Name	Description	RW
31:28	Reserved	These bits always read 0, and written are ignored.	R
27:16	CLSS	CLS signal start position. (in dot clock)	RW
15:12	Reserved	These bits always read 0, and written are ignored.	R
11:0	CLSE	CLS signal end position. (in dot clock)	RW

13.6.26 SPL Signal Setting (LCDSPL)

LCDSPL **0x13050020**

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

Bits	Name	Description	RW
31:28	Reserved	These bits always read 0, and written are ignored.	R
27:16	SPLS	SPL signal start position. (in dot clock)	RW
15:12	Reserved	These bits always read 0, and written are ignored.	R
11:0	SPLE	SPL signal end position. (in dot clock)	RW

In test mode this register use to keep TV encoder module's output data: comp_luma([25:16]) and chroma([9:0]).

13.6.27 REV Signal Setting (LCDREV)

LCDREV **0x13050024**

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

Bits	Name	Description	RW
31:28	Reserved	These bits always read 0, and written are ignored.	R
27:16	REVS	REV signal start position. (in dot clock)	RW
15:0	Reserved	These bits always read 0, and written are ignored.	R

13.6.28 Interrupt ID Register (LCDIID)

LCDIID is a read-only register that contains a copy of the Frame ID register (LCDFID) from the descriptor currently being processed when a start of frame (SOF) or end of frame (EOF) interrupt is generated. LCDIID is written to only when an unmasked interrupt of the above type is signaled and there are no other unmasked interrupts in the LCD controller pending. As such, the register is considered to be sticky and will be overwritten only when the signaled interrupt is cleared by writing the LCD controller status register. For dual-panel displays, LCDIID is written only when both channels have reached a given state.

LCDIID is written with the last channel to reach that state. (i.e. LCDFID of the last channel to reach SOF would be written in LCDIID if SOF interrupts are enabled). Reserved bits must be written with zeros and reads from them must be ignored.

LCDIID **0x13050038**

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

Bits	Name	Description	RW
31:0	IID	A copy of Frame ID register, which transferred from Descriptor.	RW

13.6.29 Descriptor Address Register0, 1 (LCDDA0, LCDDA1, LCDDA0_PART2)

A frame descriptor is a 4-word block, aligned on 4-word (16-byte) boundary, in external memory:

WORD [0] contains the physical address for next LCDDAx.

WORD [1] contains the physical address for LCDSAx.

WORD [2] contains the value for LCDFIDx.

WORD [3] contains the value for LCDCMDx.

Software must write the physical address of the first descriptor to LCDDAx before enabling the LCD Controller. Once the LCD Controller is enabled, the first descriptor is read, and all 4 registers are written by the DMAC. The next frame descriptor pointed to by LCDDAx is loaded into the registers for the associated DMA channel after all data for the current descriptor has been transferred.

NOTE: If only one frame buffer is used in external memory, the LCDDAx field (word [0] of the frame descriptor) must point back to itself. That is to say, the value of LCDDAx is the physical address of itself.

Read/write registers LCDDA0 and LCDDA1, corresponding to DMA channels 0 and 1, contain the physical address of the next descriptor in external memory. The DMAC fetches the descriptor at this location after finishing the current descriptor. On reset, the bits in this register are zero. The target address must be aligned to 16-byte boundary. Bits [3:0] of the address must be zero.

LCDDA0, LCDDA1, LCDDA0_PART2 **0x13050040, 0x13050050, 0x130501C0**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DA0, 1,0_PART2																															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:0	DA0, 1, DA0_PART2	Next descriptor physical address. And descriptor structure as following: WORD [0]: next descriptor physical address WORD [1]: the buffer physical address WORD [2]: the buffer ID value (Only for debug) WORD [3]: the buffer property. The value is same as LCDCMD	RW

13.6.30 Source Address Register0, 1 (LCDSA0, LCDSA1, LCDSA0_PART2)

Registers LCDSA0 and LCDSA1, corresponding to DMA channels 0 and 1, contain the physical address of frame buffer or palette buffer in external memory. The address must be aligned on a 4, 8, or 16 word boundary according to register LCDCTRL.BST. If this descriptor is for palette data, LCDSA0 points to the memory location of the palette buffer. If this descriptor is for frame data, LCDSAx points to the memory location of the frame buffer. This address is incremented by hardware as the DMAC fetches data from memory. If desired, the Frame ID Register can be used to hold the initial frame source address.

LCDSA0, LCDSA1, LCDSA0_PART2 **0x13050044, 0x13050054, 0x130501C4**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SA0 / SA1 / SA0_PART2																															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:0	SA0, SA1, SA0_PART2	Buffer start address. (Only for driver debug)	R

13.6.31 Frame ID Register0 (LCDFID0, LCDFID1, LCDFID0_PART2)

Registers LCDFID0 and LCDFID1, corresponding to DMA channels 0 and 1, contain an ID field that describes the current frame. The particular use of this field is up to the software. This ID register is copied to the LCD Controller Interrupt ID Register when an interrupt occurs.

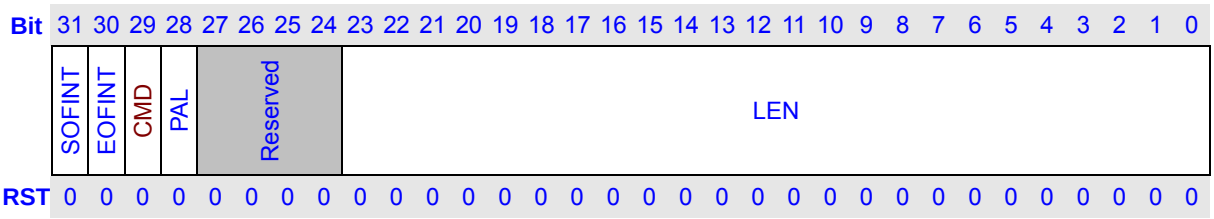
LCDFID0, LCDFID1, LCDFID0_PART2 **0x13050048, 0x13050058, 0x130501C8**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FID0 / FID1 / FID0_PART2																															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:0	FID0, FID1, FID0_PART2	Frame ID. (Only for debug)	R

13.6.32 DMA Command Register (LCDCMDx, LCDCMD0_PART2)

LCDCMD0, LCDCMD1, LCDCMD0_PART2 0x1305004C, 0x1305005C, 0x130501CC



Bits	Name	Description	RW
31	SOFINT	Enable start of frame interrupt. When SOFINT =1, the DMAC sets the start of frame bit (LCDSTATE.SOF) when starting a new frame. The SOF bit is set after a new descriptor is loaded from memory and before the palette/frame data is fetched. In dual-panel mode, LCDSTATE.SOF is set only when both channels reach the start of frame and both frame descriptors have SOFINT set. SOFINT must not be set for palette descriptors in dual-panel mode, since only one channel is ever used to load the palette descriptor.	R
30	EOFINT	Enable end of frame interrupt. When EOFINT =1, the DMAC sets the end of frame bit (LCDSTATE.EOF) after fetching the last word in the frame buffer. In dual-panel mode, LCDSTATE.EOF is set only when both channels reach the end of frame and both frame descriptors have EOFINT set. EOFINT must not be set for palette descriptors in dual-panel mode, since only one channel is ever used to load the palette descriptor.	R
29	CMD	It is used to distinguish command and data in lcm mode. And it is only loaded via DMA channel 0. 1: The data is command 0: The data is data	R
28	PAL	The descriptor contains a palette buffer. PAL indicates that data being fetched will be loaded into the palette RAM. If PAL =1, the palette RAM data is loaded via DMA channel 0 as follows: In bpp1, 2, 4, 8 mode, software must load the palette at least once after enabling the LCD. In bpp16 mode, PAL must be 0.	R
27:24	Reserved	These bits always read 0, and written are ignored.	R

23:0	LEN	The buffer length value. (in WORD) The LEN bit field determines the number of bytes of the buffer size pointed by LCDSAx. LEN = 0 is not valid. DMAC fetch data according to LEN. Each time one or more word(s) been fetched, LEN is decreased automatically. Software can read LEN.	R

13.6.33 DMA OFFSIZE Register (LCDOFFSx, LCDOFFS0_PART2)

LCDOFFS0, LCDOFFS1, LCDOFFS0_PART2 **0x13050060, 0x13050070, 0x130501E0**

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

Bits	Name	Description	RW
23:0	OFFSIZE0, 1 OFFSIZE0_PART2	OFFSIZE value for DMA 0,1. Indicate the offset in word. *please notice that when you need OFFSIZE function, to set this reg to an un-zero value and also need to set LCDPW0, 1 to indicate how much word in one line of this frame.	R

13.6.34 DMA Page Width Register (LCDPWx, LCDPW0_PART2)

LCDPW0, LCDPW1, LCDPW_PART2 **0x13050064, 0x13050074, 0x130501E4**

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

Bits	Name	Description	RW
23:0	PAGEWIDTH0, 1 PAGEWIDTH0_PART2	Page width for DMA 0,1. * When you set LCDOFFS.OFFSIZE0/1 to 0, you need not care the PAGEWIDTH0/1.	R

13.6.35 DMA Commend Counter Register0, 1 (LCDCNUM0,1)

When LCDCMD.CMD = 1, **0x13050068, 0x13050078** is use as LCDCNUM0, 1 LCDCNUM0_PART2 are not used now, set it to 0.

LCDCNUM0, LCDCNUM1, LCDCNUM0_PART2 **0x13050068, 0x13050078, 0x130501E8**

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

Bits	Name	Description	RW
7:0	CNUM0,1	Commands' number in this frame transfer by DMA. (only use in Smart LCD mode).	R

13.6.36 Foreground x Size in Descriptor (LCDDESSIZEx, LCDDESSIZE0_PART2)

When LCDCMD.CMD = 0, **0x1305006C, 0x1305007C** is use as LCDDESSIZE0, 1, to indicator the next frame foreground0, 1's size.

LCDDESSIZE0, 1, LCDDESSIZE0_PART2 **0x1305006C, 0x1305007C, 0x130501EC**

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

Bits	Name	Description	RW
31:28	Reserved	These bits always read 0, and written are ignored.	R
27:16	Height	The height of foreground 0.	R
15:12	Reserved	These bits always read 0, and written are ignored.	R
11:0	Width	The width of foreground 0.	R

13.7 LCD Controller Pin Mapping

There are several mapping schemes for different LCD panels.

13.7.1 TFT and CCIR Pin Mapping

Pin	Generic 8-bit Serial TFT	Generic 18/16-bit Parallel TFT		Special TFT 1 18/16-bit Parallel		Special TFT 2 18/16-bit Parallel		Special TFT 3 18/16-bit Parallel		CCIR656 8-bit	CCIR601 16-bit
Lcd_pclk/ Slcd_clk	CLK	CLK		DCLK		CLK		HCLK		-	CLK
Lcd_vsync/ Slcd_cs	VSYNC	VSYNC		SPS		GSRT		STV		-	VSYNC
Lcd_hsync/ Slcd_rs	HSYNC	HSYNC		LP		GPCK		STH		-	HSYNC
Lcd_de	DE	DE		-		-		-		-	-
Lcd_ps	-	-		Pulse		Toggle		Toggle		-	-
Lcd_cls	-	-		Pulse		Pulse		Pulse		-	-
Lcd_rev	-	-		Toggle		Toggle		Toggle		-	-
Lcd_spl	-	-		Pulse		Pulse		**Toggle**		-	-
Lcd_dat17	-	R5	-	R5	-	R5	-	R5	-	-	-
Lcd_dat16	-	R4	-	R4	-	R4	-	R4	-	-	-
Lcd_dat15	-	R3	R5	R3	R5	R3	R5	R3	R5	-	D15
Lcd_dat14	-	R2	R4	R2	R4	R2	R4	R2	R4	-	D14
Lcd_dat13	-	R1	R3	R1	R3	R1	R3	R1	R3	-	D13
Lcd_dat12	-	R0	R2	R0	R2	R0	R2	R0	R2	-	D12
Lcd_dat11	-	G5	R1	G5	R1	G5	R1	G5	R1	-	D11
Lcd_dat10	-	G4	G5	G4	G5	G4	G5	G4	G5	-	D10
Lcd_dat9	-	G3	G4	G3	G4	G3	G4	G3	G4	-	D9
Lcd_dat8	-	G2	G3	G2	G3	G2	G3	G2	G3	-	D8
Lcd_dat7	R7/G7/B7	G1	G2	G1	G2	G1	G2	G1	G2	D7	D7
Lcd_dat6	R6/G6/B6	G0	G1	G0	G1	G0	G1	G0	G1	D6	D6
Lcd_dat5	R5/G5/B5	B5	G0	B5	G0	B5	G0	B5	G0	D5	D5
Lcd_dat4	R4/G4/B4	B4	B5	B4	B5	B4	B5	B4	B5	D4	D4
Lcd_dat3	R3/G3/B3	B3	B4	B3	B4	B3	B4	B3	B4	D3	D3
Lcd_dat2	R2/G2/B2	B2	B3	B2	B3	B2	B3	B2	B3	D2	D2
Lcd_dat1	R1/G1/B1	B1	B2	B1	B2	B1	B2	B1	B2	D1	D1
Lcd_dat0	R0/G0/B0	B0	B1	B0	B1	B0	B1	B0	B1	D0	D0

TFT 24 bit parallel mode

Pin	16 bit Parallel mode2	24 bit Parallel
Lcd_pclk/ Slcd_clk	CLK	CLK
Lcd_vsync/Slcd_cs	VSYNC	VSYNC
Lcd_hsync/Slcd_rs	HSYNC	HSYNC
Lcd_de	DE	DE
Lcd_ps	-	-
Lcd_cls	-	-
Lcd_rev	-	-
Lcd_spl	-	-
Lcd_dat17	R7	R7
Lcd_dat16	R6	R6
Lcd_dat15	R5	R5
Lcd_dat14	R4	R4
Lcd_dat13	R3	R3
Lcd_dat12	G7	R2
Lcd_dat11	G6	G7
Lcd_dat10	G5	G6
Lcd_dat9	0 (NC for panel)	G5
Lcd_dat8	G4	G4
Lcd_dat7	G3	G3
Lcd_dat6	G2	G2
Lcd_dat5	B7	B7
Lcd_dat4	B6	B6
Lcd_dat3	B5	B5
Lcd_dat2	B4	B4
Lcd_dat1	B3	B3
Lcd_dat0	0 (NC for panel)	B2
Lcd_lo6_o[5]	0	R1
Lcd_lo6_o[4]	0	R0
Lcd_lo6_o[3]	0	G1
Lcd_lo6_o[2]	0	G0
Lcd_lo6_o[1]	0	B1
Lcd_lo6_o[0]	0	B0

13.7.2 Single Panel STN Pin Mapping

Pin	Color STN	Mono STN			
	PDW=3	PDW=0	PDW=1	PDW=2	PDW=3
Lcd_pclk	CLK	CLK	CLK	CLK	CLK
Lcd_vsync	VSYNC	VSYNC	VSYNC	VSYNC	VSYNC
Lcd_hsync	HSYNC	HSYNC	HSYNC	HSYNC	HSYNC
Lcd_de	BIAS	BIAS	BIAS	BIAS	BIAS
Lcd_ps	-	-	-	-	-
Lcd_cls	-	-	-	-	-
Lcd_rev	-	-	-	-	-
Lcd_spl	-	-	-	-	-
Lcd_dat17	-	-	-	-	-
Lcd_dat16	-	-	-	-	-
Lcd_dat15	-	-	-	-	-
Lcd_dat14	-	-	-	-	-
Lcd_dat13	-	-	-	-	-
Lcd_dat12	-	-	-	-	-
Lcd_dat11	-	-	-	-	-
Lcd_dat10	-	-	-	-	-
Lcd_dat9	-	-	-	-	-
Lcd_dat8	-	-	-	-	-
Lcd_dat7	D7	-	-	-	D7
Lcd_dat6	D6	-	-	-	D6
Lcd_dat5	D5	-	-	-	D5
Lcd_dat4	D4	-	-	-	D4
Lcd_dat3	D3	-	-	D3	D3
Lcd_dat2	D2	-	-	D2	D2
Lcd_dat1	D1	-	D1	D1	D1
Lcd_dat0	D0	D0	D0	D0	D0

13.7.3 Dual Panel STN Pin Mapping

Pin	Color STN	Mono STN			
	PDW=3	PDW=0	PDW=1	PDW=2	PDW=3
Lcd_pclk	CLK	-	-	CLK	CLK
Lcd_vsync	VSYNC	-	-	VSYNC	VSYNC
Lcd_hsync	HSYNC	-	-	HSYNC	HSYNC
Lcd_de	BIAS	-	-	BIAS	BIAS
Lcd_ps	-	-	-	-	-
Lcd_cls	-	-	-	-	-
Lcd_rev	-	-	-	-	-
Lcd_spl	-	-	-	-	-
Lcd_dat17	-	-	-	-	-
Lcd_dat16	-	-	-	-	-
Lcd_dat15	UD7	-	-	-	UD7
Lcd_dat14	UD6	-	-	-	UD6
Lcd_dat13	UD5	-	-	-	UD5
Lcd_dat12	UD4	-	-	-	UD4
Lcd_dat11	UD3	-	-	UD3	UD3
Lcd_dat10	UD2	-	-	UD2	UD2
Lcd_dat9	UD1	-	-	UD1	UD1
Lcd_dat8	UD0	-	-	UD0	UD0
Lcd_dat7	LD7	-	-	-	LD7
Lcd_dat6	LD6	-	-	-	LD6
Lcd_dat5	LD5	-	-	-	LD5
Lcd_dat4	LD4	-	-	-	LD4
Lcd_dat3	LD3	-	-	LD3	LD3
Lcd_dat2	LD2	-	-	LD2	LD2
Lcd_dat1	LD1	-	-	LD1	LD1
Lcd_dat0	LD0	-	-	LD0	LD0

13.8 Display Timing

13.8.1 General 16-bit and 18-bit TFT Timing

This section shows the general 16-bit and 18-bit TFT LCD timing diagram, the polarity of signal "Vsync", "Hsync", and "PCLK" can be programmed correspond to the LCD panel specification.

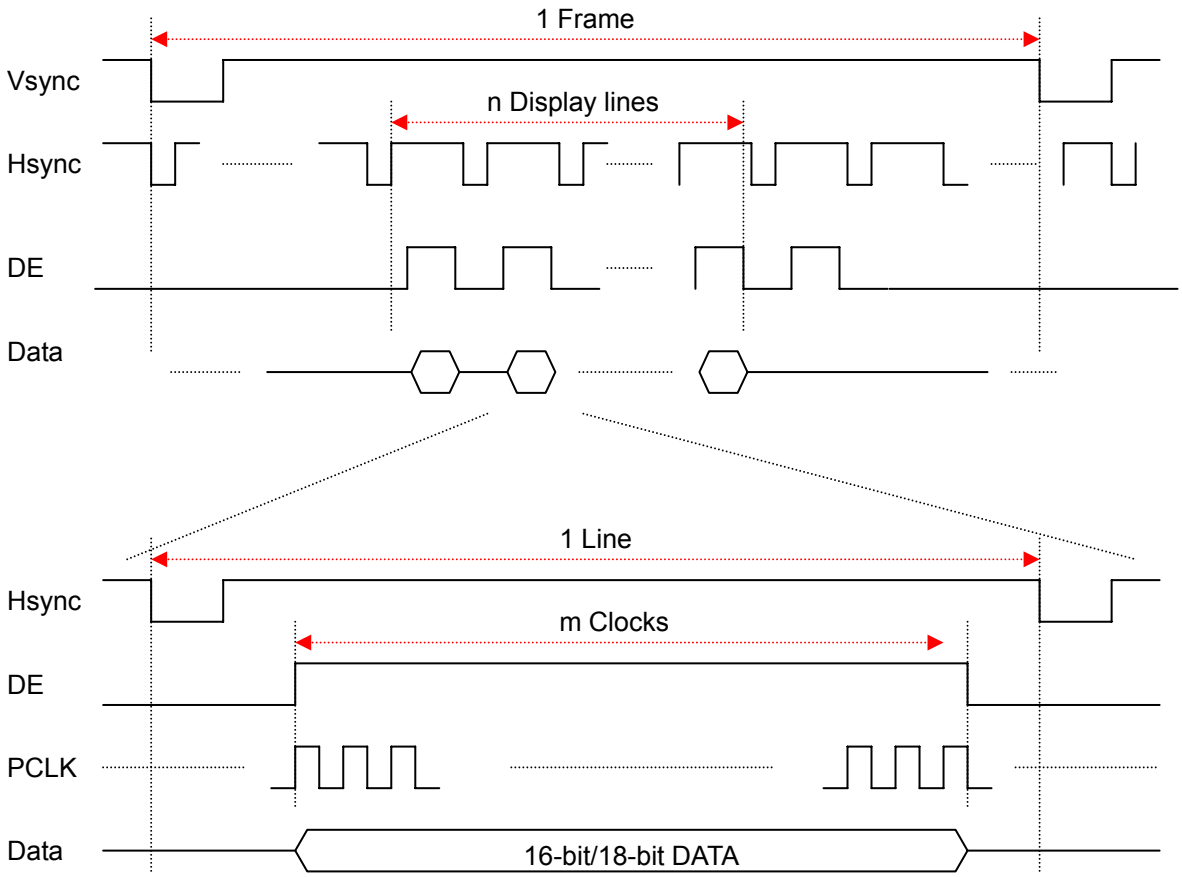


Figure 13-7 General 16-bit and 18-bit TFT LCD Timing

13.8.2 8-bit Serial TFT Timing

This section shows the 8-bit serial TFT LCD timing diagram, the polarity of signal “Vsync”, “Hsync”, and “PCLK” can be programmed correspond to the LCD panel specification.

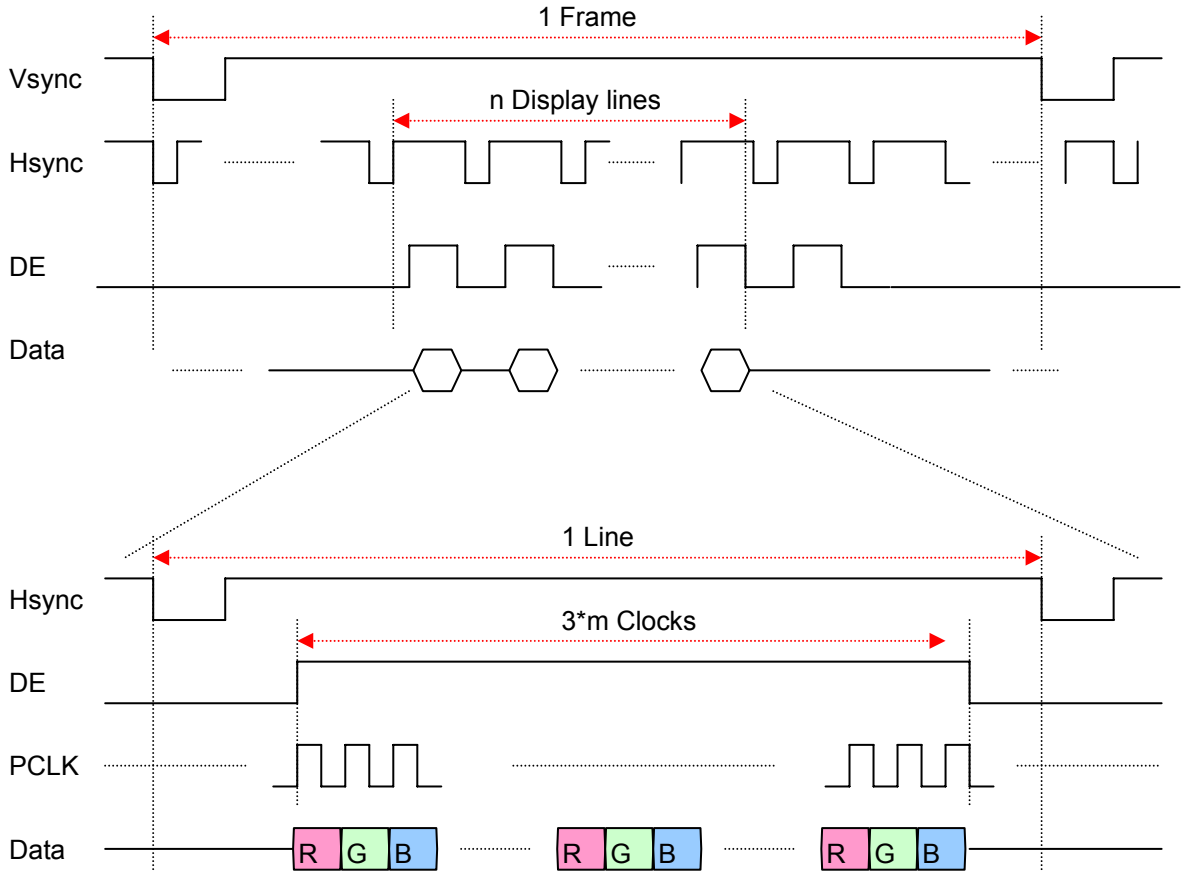


Figure 13-8 8-bit serial TFT LCD Timing (24bpp)

13.8.3 Special TFT Timing

Based on the general TFT LCD support, this controller also provides 4 special signals that can be programmed to general some special timing used for some panel. All 4 signals are worked in two modes: pulse mode and toggle mode. Signal “CLS” is fixed in pulse mode, and “REV” in toggle mode. The work mode of signals “SPL” and “PS” are defined in the special TFT LCD mode 1 to mode 3, either pulse mode or toggle mode. The position and polarity of these 4 signals can be programmed via registers. The Figures show the two modes as follows: (The toggle mode of signal “SPL” is different with the others signal. “SPL” does toggle after display line.)

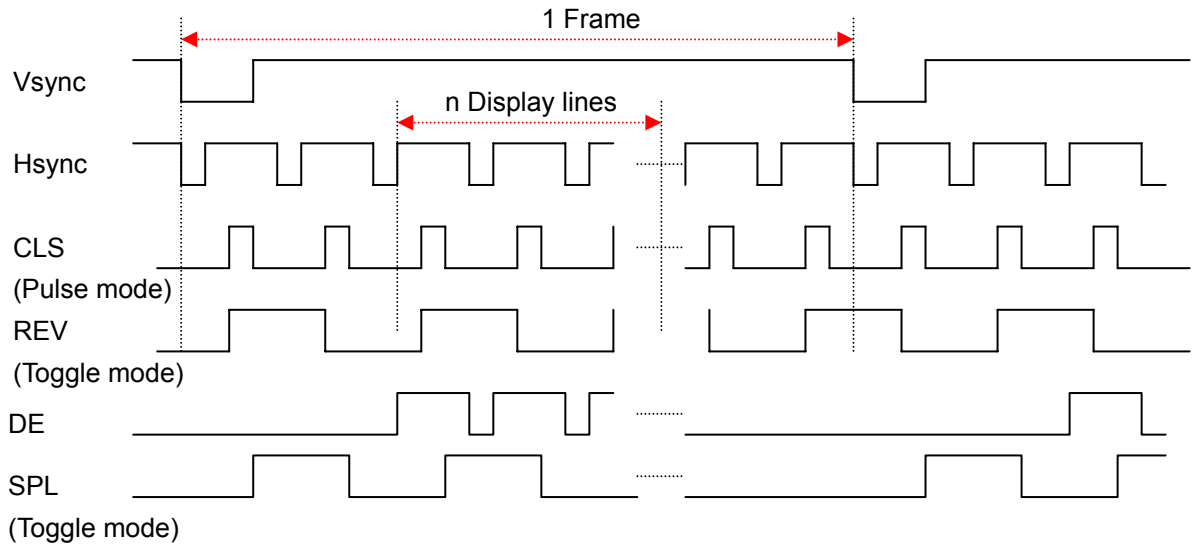


Figure 13-9 Special TFT LCD Timing 1

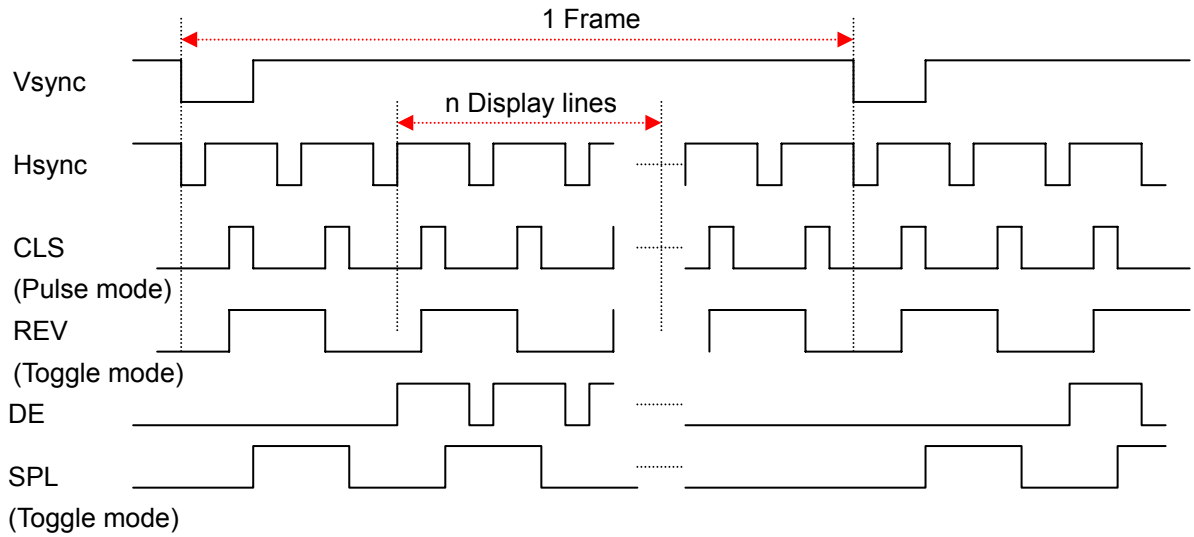


Figure 13-10 Special TFT LCD Timing 2

These two Figures show the timing of pulse mode and toggle mode, the pulse mode timing is same

and the toggle mode timing is different. Timing 1 shows the condition when the total lines in 1 frame is odd (the number of display is even and the number of blank is odd), so the phase of REV inverse at the first line of each frame and the phase of SPL dose not inverse at the first line of each frame. Timing 2 shows the condition when the total lines in 1 frame is even (the number of display is even and the number of blank is even), so the phase of REV and SPL dose not inverse at the first line of each frame.

When LCDC is enabled ,there will be a null line to be add before transferring data to LCD panel. So the toggle mode exept SPL signal of special 3 TFT mode is when reset level is high,the first valid edge will be rising edge. SPL signal of special 3 TFT mode is when reset level is high,the first valid edge will be falling edge.

13.8.4 Delta RGB panel timing

This section shows the Delta RGB timing diagram, the polarity of signal “Vsync”, “Hsync”, and “PCLK” can be programmed. And the odd/even line RGB order also can be programmed correspond to the LCD panel specification.

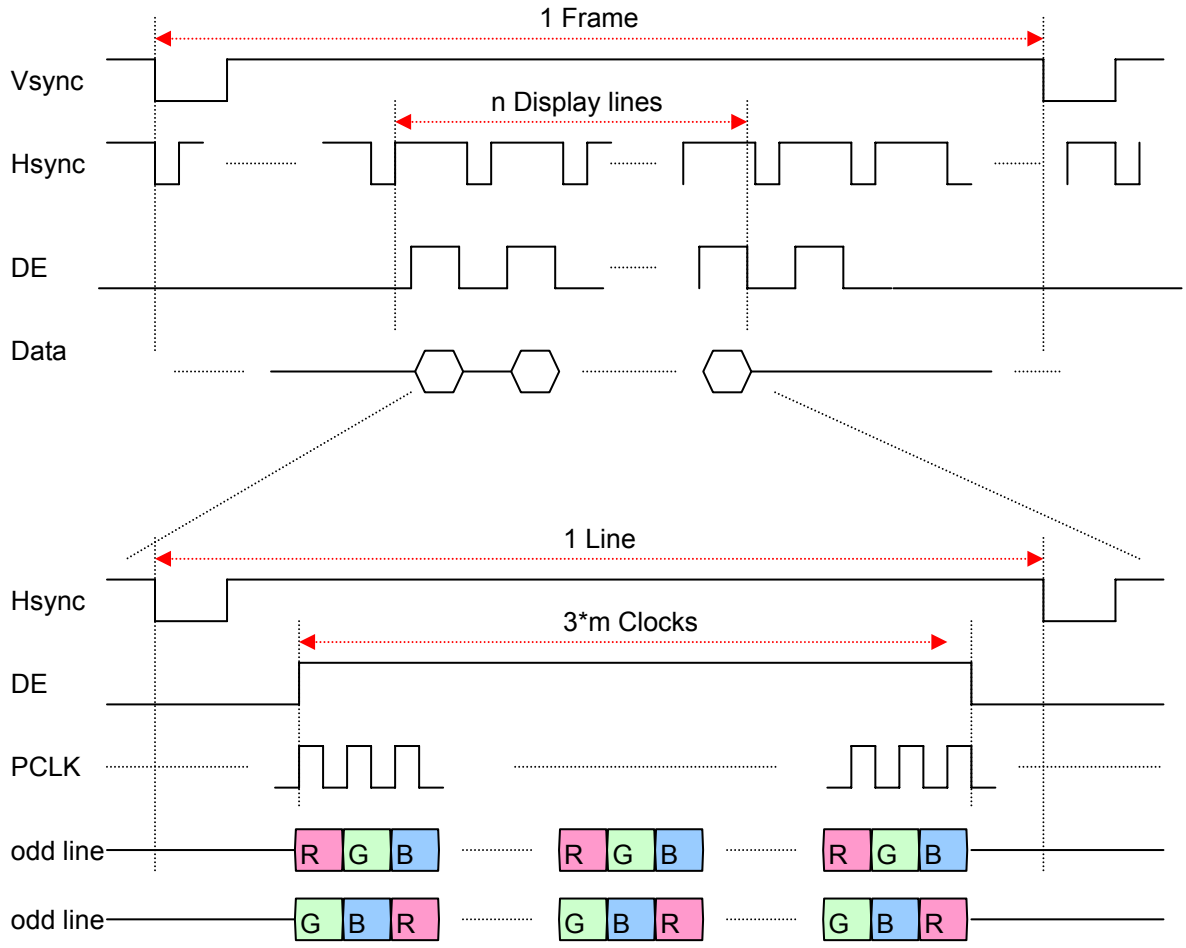
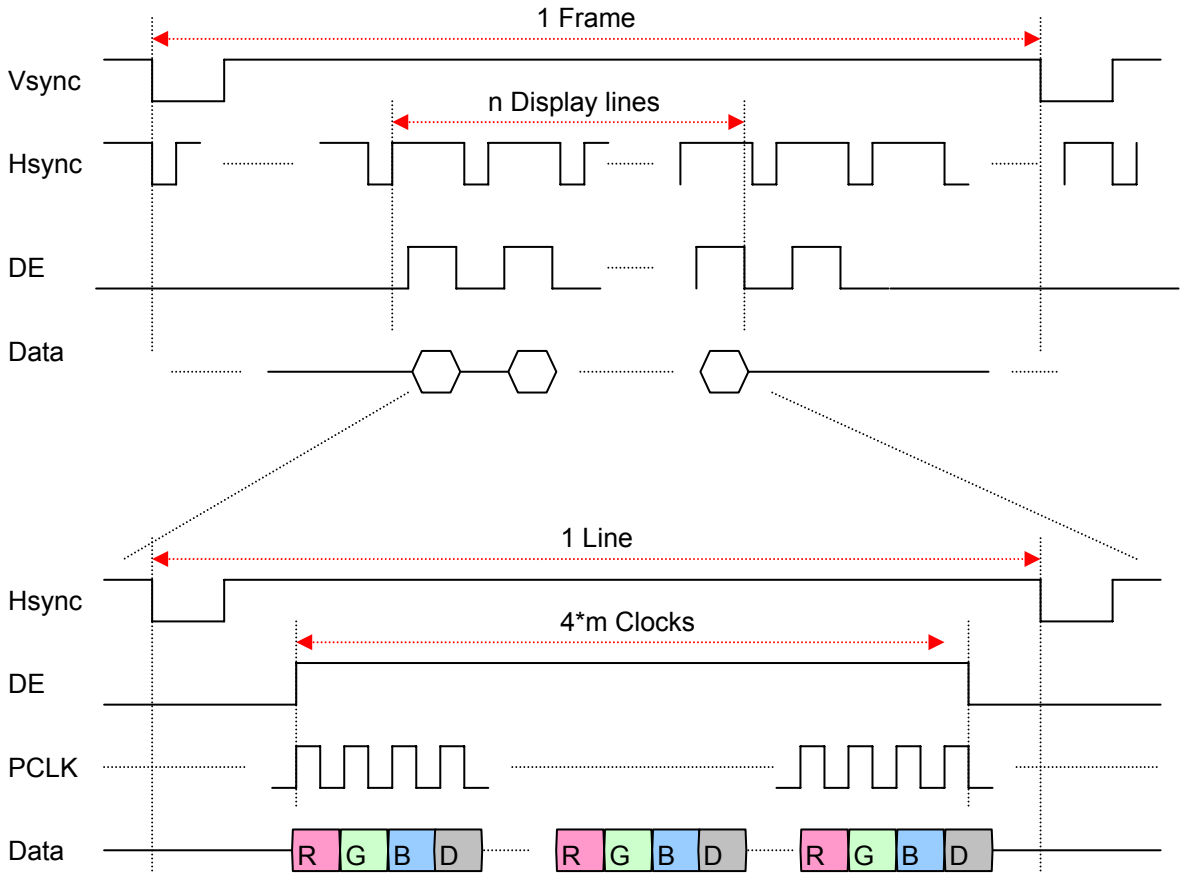


Figure 13-11 Delta RGB timing

13.8.5 RGB Dummy mode timing

This section shows the RGB Dummy diagram, the polarity of signal “Vsync”, “Hsync”, and “PCLK” can be programmed.



***Dummy = 0**

Figure 13-12 RGB Dummy timing

13.9 Format of Palette

This LCD controller contains a palette RAM with 256-entry x 16-bit used only for BPP8, BPP4, BPP2 and BPP1. Palette RAM data is loaded directly from the external memory palette buffer by DMAC channel 0. Each word of palette buffer contains 2 palette entries.

- In 8-bpp modes, palette buffer size is128 words
- In 4-bpp modes, palette buffer size is 8 words
- In 2-bpp modes, palette buffer size is 2 words
- In 1-bpp modes, palette buffer size is 1 word
- In 16/18/24-bpp modes, has no palette buffer

Palette buffer base address	Bit: 31 . . . 16	Bit: 15 . . . 0
Palette entry	Entry-1 bit: 15 . . . 0	Entry-0 bit: 15 . . . 0
Palette buffer base address + 4	Bit: 31 . . . 16	Bit: 15 . . . 0
Palette entry	Entry-3 bit: 15 . . . 0	Entry-2 bit: 15 . . . 0
Palette buffer base address + 8	Bit: 31 . . . 16	Bit: 15 . . . 0
Palette entry	Entry-5 bit: 15 . . . 0	Entry-4 bit: 15 . . . 0

13.9.1 STN

For STN Panel, 16-bpp pixel data is encoded with RGB 565 or RGB 555.
Please refer to register LCDCTRL.RGB.

BPP 16, RGB 565, pixel encoding for STN Panel:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0

BPP 16, RGB 555, pixel encoding for STN Panel:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	R4	R3	R2	R1	R0	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0

13.9.2 TFT

BPP 16, RGB 565, pixel encoding for TFT Panel:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0

NOTE: For BPP 16, 18, 24, palette is bypass.

13.10 Format of Frame Buffer

13.10.1 16bpp

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0

13.10.2 18bpp

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	R5	R4	R3	R2	R1	R0	0	0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
G5	G4	G3	G2	G1	G0	0	0	B5	B4	B3	B2	B1	B0	0	0

13.10.3 24bpp

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	R7	R6	R5	R4	R3	R2	R1	R0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0

13.10.4 16bpp with alpha

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
A7	A6	A5	A4	A3	A2	A1	A0	0	0	0	0	0	0	0	0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
R5	R4	R3	R2	R1	G5	G4	G3	G2	G1	G0	B5	B4	B3	B2	B1

13.10.5 18bpp with alpha

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
A7	A6	A5	A4	A3	A2	A1	A0	R5	R4	R3	R2	R1	R0	0	0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
G5	G4	G3	G2	G1	G0	0	0	B5	B4	B3	B2	B1	B0	0	0

13.10.6 24bpp with alpha

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
A7	A6	A5	A4	A3	A2	A1	A0	R7	R6	R5	R4	R3	R2	R1	R0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0

13.10.7 24bpp compressed

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RED 3 [7:0]								GLEEN 3 [7:0]							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BLUE3 [7:0]								RED2 [7:0]							

13.11 Format of Data Pin Utilization

13.11.1 Mono STN

In Mono STN mode, data pin pixel ordering of one LCD screen row. Column 0 is the first pixel of a screen row.

Upper panel								
Panel data width	**Col0**	**Col1**	**Col2**	**Col3**	**Col4**	**Col5**	**Col6**	**Col7**
1 bit	D0	D0	D0	D0	D0	D0	D0	D0
2 bit	D1	D0	D1	D0	D1	D0	D1	D0
4 bit	D3	D2	D1	D0	D3	D2	D1	D0
8 bit	D7	D6	D5	D4	D3	D2	D1	D0
Lower panel (dual-panel mode)								
4 bit	D11	D10	D9	D8	D11	D10	D9	D8
8 bit	D15	D14	D13	D12	D11	D10	D9	D8

13.11.2 Color STN

In Color STN mode, data pin pixel ordering of one LCD screen row. Column 0 is the first pixel of a screen row.

Upper panel							
Col0 (R)	**Col0 (G)**	**Col0 (B)**	**Col1 (R)**	**Col1 (G)**	**Col1 (B)**	**Col2 (R)**	**Col2 (G)**
D7	D6	D5	D4	D3	D2	D1	D0
Lower panel (dual-panel mode)							
D15	D14	D13	D12	D11	D10	D9	D8

13.11.3 18-bit Parallel TFT

Col0 (RGB)																	
D17	D16	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

13.11.4 16-bit Parallel TFT

Col0 (RGB)															
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

13.11.5 8-bit Serial TFT (24bpp)

Col0 (R)							
D7	D6	D5	D4	D3	D2	D1	D0
Col0 (G)							
D7	D6	D5	D4	D3	D2	D1	D0
Col0 (B)							
D7	D6	D5	D4	D3	D2	D1	D0

13.12 LCD Controller Operation

13.12.1 Set LCD Controller Device Clock and Pixel Clock

The LCD Controller has 2 clock input: device clock and pixel clock. The both clocks are generated by CPM (Clock and Power Manager). The frequency of the 2 clocks can be set by CPM registers. CPM registers CPCCR.LDIV and CPCCR.PCS set LCD device clock division ratio, and LPCDR set LCD pixel clock division ratio. Please refer to CPM spec for detail.

LCD device clock is the LCD controller's internal clock while LCD pixel clock is output to drive LCD panel. There have 2 rules for LCD clocks:

1 For TFT Panel, the frequency of LCD device clock must be at least 1.5 times of LCD pixel clock.
2 For STN Panel, the frequency of LCD device clock must be at least 3 times of LCD pixel clock.

LCD panel determines the frequency of LCD pixel clock.

13.12.2 Enabling the Controller

If the LCD controller is being enabled for the first time after system reset or sleep reset, all of the LCD registers must be programmed as follows:

1 Write the frame descriptors and, if needed, the palette descriptor to memory.
2 Program the entire LCD configuration registers except the Frame Descriptor Address Registers (LCDDAx) and the LCD Controller enable bit (LCDCTRL.ENA).
3 Program LCDDAx with the memory address of the palette/frame descriptor.
4 Enable the LCD controller by writing to LCDCTRL.ENA.

If the LCD controller is being re-enabled, there has not been a reset since the last programming; only the registers LCDDAx and LCDCTRL.ENA need to be reprogrammed. The LCD Controller Status Register (LCDSTATE) must also be written to clear any old status flags before re-enabling the LCD controller.

Once the LCD controller has been enabled, do not write new values to LCD registers except LCDCTRL.ENA or DIS or LCDDA0/1 or LCDOSDC.F0/1EN .

13.12.3 Disabling the Controller

The LCD controller can be disabled in two ways: regular and quick.

1 Regular disabling.
Regular disabling is accomplished by setting the disable bit, LCDCTRL.DIS. The other bits in LCDCTRL must not be changed — read the register, set the DIS bit, and rewrite the register. This method causes the LCD controller to stop cleanly at the end of a frame. The LCD Disable Done bit, LCDSTATE.LDD, is set when the LCD controller finishes displaying the last

frame, and the enable bit, LCDCTRL.ENA, is cleared automatically by hardware. LCDCTRL.DIS must be set zero when enabling the controller.

2 Quick disabling.

Quick disabling is accomplished by clearing the enable bit, LCDCTRL.ENA. The LCD controller will finish any current DMA transfer, stop driving the panel, setting the LCD Quick Disable bit (LCDSTATE.QD) and shut down immediately. This method is intended for situations such as a battery fault, where system bus traffic has to be minimized immediately so the processor can have enough time to store critical data to memory before the loss of power. The LCD controller must not be re-enabled until the QD bit is set, indicating that the quick shutdown is complete. Do not set the DIS bit when a quick disabling command has been issued.

NOTE: It is strongly recommended that software set the "LCD Module Stop Bit" in PMC to shut down LCDC clock supply to save power consumption after disable LCDC. Please refer to PMC for detailed information.

13.12.4 Resetting the Controller

At reset, the LCD Controller is disabled. All LCD Controller Registers are reset to the conditions shown in the register descriptions.

13.12.5 Frame Buffer & Palette Buffer

The starting address of frame buffer stored in external memory must be aligned to 4, 8 or 16 words boundary according to register LCDCTRL.BST. The length of buffer must be multiple of word (32-bit).

If LCDCTRL .BST = 0, align frame and palette buffer to 16 word boundary.
If LCDCTRL .BST = 1, align frame and palette buffer to 8 word boundary.
If LCDCTRL .BST = 2, align frame and palette buffer to 4 word boundary.

One frame buffer contains encoded pixel data of multiple of screen lines; each line of encoded pixel data must be aligned to word boundary. If the length of a line is not the multiple of word, extra bits must be applied to reach a word boundary. It is suggested that the extra bits to be set zero.

13.12.6 CCIR601/CCIR656

CCIR601: just as 16bit-parallel output.
CCIR656: need external encoder, or software designer need give digital blanking data and timing reference signal in data buffer.

13.12.7 OSD Operation

1 Normal process.

a Configuration.

* LCDCFG and LCDCTRL
* LCDOSDC and LCDOSDCTRL
* LCDRGBC and LCDIPUR

b Set Color.

* LCDBGC, LCDKEY0, LCDKEY1, LCDALHPA

c Set Display.

* LCDVAT, LCDDAH, LCDDAV
* LCDXYP0, LCDXYP1, LCDSIZE0, LCDSIZE1
* LCDVSYNC, LCDHSYNC

d Set DMAC.

* LCDIID
* LCDDA0, LCDSA0, LCDFID0, LCDCMD0, LCDOFFS0, LCDPW0, LCDCNUM0, LCDDESSIZE0
* LCDDA1, LCDSA1, LCDFID1, LCDCMD1, LCDOFFS1, LCDPW1, LCDCNUM1, LCDDESSIZE1

e Enable LCDC.

f Check the state from register LCDSTATE and LCDOSDS.

2 Reconfigure OSD.

If foreground0 and foreground1 (enable, position, size)need to reconfigure during display process, there has two methods.

Method1:(recommend in TFT and SLCD):

a Reconfigure the relate Register after disable LCDC.

b In TFT mode, use normal disable to avoid lcd panel flicker.

c In SLCD mode, use quick disable (smart LCD could keep the frame by its inner buffer).

d After disable LCDC, you can reconfigure any register/descriptor, but please make sure this process is quick enough in TFT mode (less than the interval between two frames).

Method2:

Dynamic reconfigure the register:

a When use TFT panel. During the display process, you can re-configure the LCDOSDC.F0EN, LCDCOSDC.F1EN; (You can not change them when use SLCD or TVE) but the new configuration will recognized by LCDC module after finished a complete frame.

- Set LCDOSDCTRL.CHANGE = 1.
- Wait LCDOSDS.READY=1.
- Reconfigure F0/1EN.
- Wait CHANGE = 0.

b If you need to re-configure LCDOSDCTRL.IPU to select IPU or DMA channel 1, you need to follow the process below:

- Quick or Normal disable LCDC. (SLCD only can use Quick disable)
- Configure the LCDOSDCTRL to set IPUEN, and then enable LCD.

To change IPU to DMA1 you can :

- Quick or Normal disable LCDC. (SLCD only can use Quick disable)
- Configure the LCDOSDCTRL to set IPUEN = 0, and then enable LCD.

3 During the display process, while foreground 1 use IPU, to change size of foreground 1 you need follow the step shown bellow.

a Quick or Normal disable LCDC. (SLCD only can use Quick disable)

b Configure the IPU, and LCDSIZE1.

c Run IPU and enable LCDC.

4 You **CAN NOT** change BPP or OSDBPP during the display process. if you want to change them first you should disable LCDC, change the BPP or OSDBPP and then enable LCDC.
If you need not use Foreground0 during the whole display process. set BPP to 5.

5 You can change LCDSIZE0/1, during display process without disable LCD controller.

method 1:

a Disable LCDCOSDC.F0/1EN.

b Re-configure LCDSIZE0/1 (and the relate DMA0/1 descriptor), then set LCDOSDCTRL.CHANGE = 1.

c Wait until CHANGE = 0 and then set LCDOSDC.F0/1EN = 1.

method 2:

a Set LCDOSDCTRL.CHANGE = 1.

b Wait until LCDOSDS. READY = 1.

c Change relate DMA0/1 channel descriptor.

d Wait until LCDOSDCTRL.CHANGE = 0.

*Please notice that in TVE (not include VGA)and SLCD only use method 2.

6 You can change LCDXY0/1 during display process without disable LCD controller.

Method 1:

a Disable LCDOSDC.F0/1EN.

b Change LCDXYPOS0/1 and then set LCDOSDCTRL.CHANGE = 1.

c Wait until LCDOSDCTRL.CHANGE = 0.

Method 2:

a Change LCDXYPOS0/1.

b Set LCDOSDCTRL.CHANGE = 1.

c Wait until LCDOSDCTRL.CHANGE = 0.

*Please notice that in TVE (not include VGA) and SLCD only use method 2.

*Please notice that if you do not change foreground 0/1's size and position, keep LCDOSDCTRL.CHANGE = 0. And you can only change one of them in one time.

7 How to "close/open" foreground0 and foreground1?

Method 1:

Direct change LCDOSDC.F0/1EN, but you must follow the rule above.

Method 2:

Change foreground0/1 size to 0 Without change LCDOSDC.F0/1EN.

Method 3: (recommend)

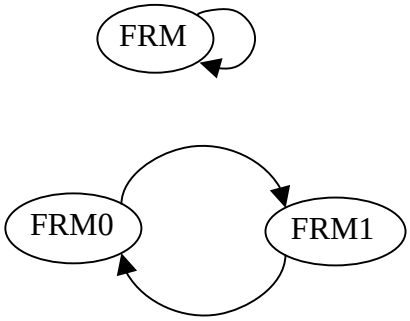
Normal disable LCDC, and change LCDOSDC.F0/1EN. Use normal disable need to wait LCDSTATE.LDD, and set relate register soon, to make sure the LCD panel are not flicker.

*Please notice that in TVE (not include VGA) and SLCD only use method 2,3. And strongly suggest that DO NOT close both foreground0 and 1 or set both foreground0 and 1 's size to 0.

13.12.8 Descriptor Operation

1 TFT panel

Not use palette: you can use only one descriptor or several connected descriptor. As which shown below.



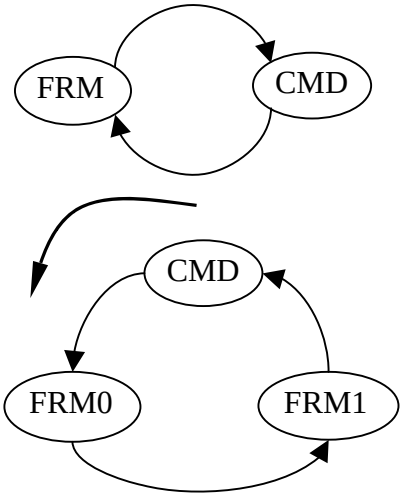
Use palette: add one PAL descriptor at the beginning of descriptor chain.

When you need to change palette during the display you need follow the steps shown below.

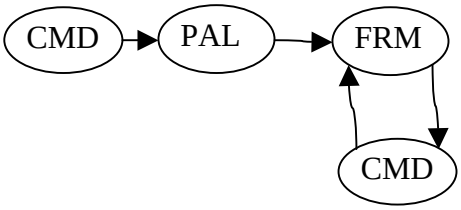
*Please notice that you **cannot** disable foreground 0 during the whole process. and also You can not change PAL when Foreground 0' s area == 0 or not enable LCDOSDC.F0EN

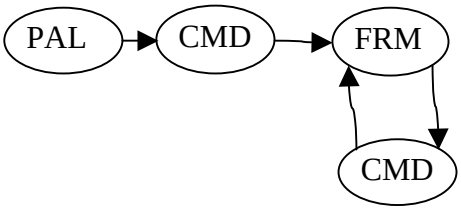
2 SLCD

Not use palette.

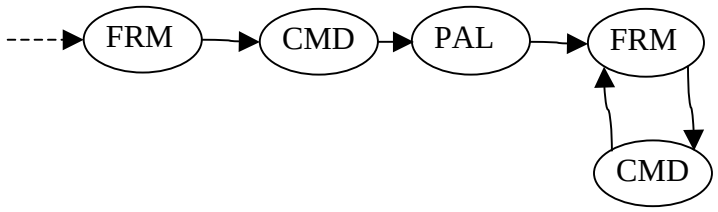


Use palette.





Change palette.



You can not change PAL when Foreground 0' s area == 0. Or not enable LCDOSDC.F0EN and during you change PAL, you can not change F0 or F1's size.

13.12.9 IPU direct connect mode

When you use IPU direct connect mode, you need to:

1 Open IPU early than LCDC.
2 Use normal disable in TFT mode, and use quick disable in SLCD/TVE mode.
3 When you use normal disable you need to wait IPU frame end flag.
4 When you use quick disable you must not wait IPU frame end flag, and must reset IPU before restart LCDC and IPU.
5 In SLCD mode, you can first wait IPU frame end flag, then quick stop LCDC. Then you need not reset IPU before restart LCDC and IPU.

* "IPU frame end flag" please refer to IPU spec.

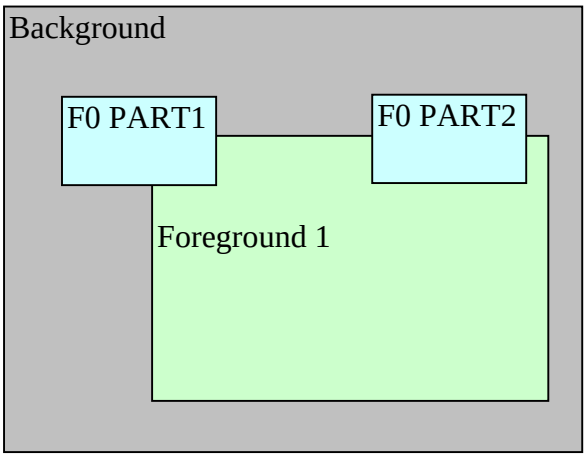
13.12.10 Foreground 0 divide mode

In divide mode the original register of foreground 0 position and size are correspond to F0 PART1, the additional (named with "_part2") registers correspond to F0 PART2.

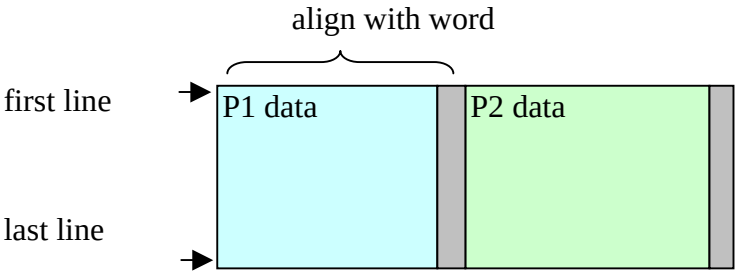
LCDOSDC.F0EN correspond the total foreground 0 （part1 and part2）and each part has a F0PxEN to enable.

F0EN, F0P2EN, F0P1EN, and part2's position/size can be reconfigure during display process. (refer to 1.12.6)

MODE 1: LCDOSDC.F0DIVMD = 0. F0P2MD = 1.

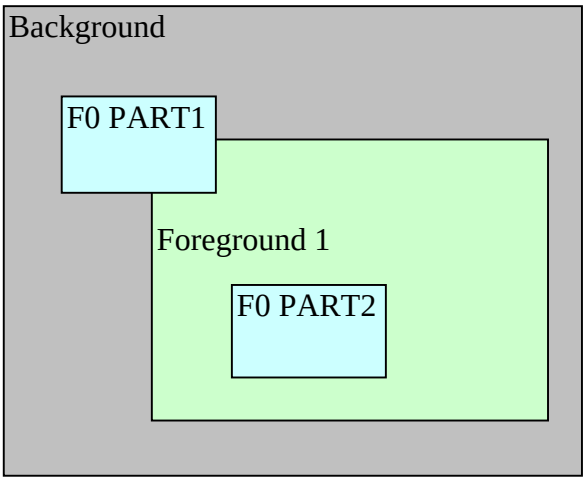


Foreground 0 divided into 2 parts, and PART1, PART2 must begin with same line and has the same height. They can have different width but cannot overlay each other.
They can use only one descriptor (connect to it self).
The two parts data must “combination” in one data buffer as follow:



*Please notice that in this mode, you need to disable LCDC before reconfigure foreground0/1’s Register.

MODE 2: LCDOSDC.F0DIVMD = 1. F0P2MD = 0. F0P1EN = 1. F0P2EN = 1.



Foreground 0 divided into 2 parts, and PART1, PART2 can have different width and height but cannot overlay each other. PART2 must below PART1 they also cannot have any superposition in vertical.

PART1 and PART2 use independent descriptor refer to descriptor register with “part2”.

14 Smart LCD Controller

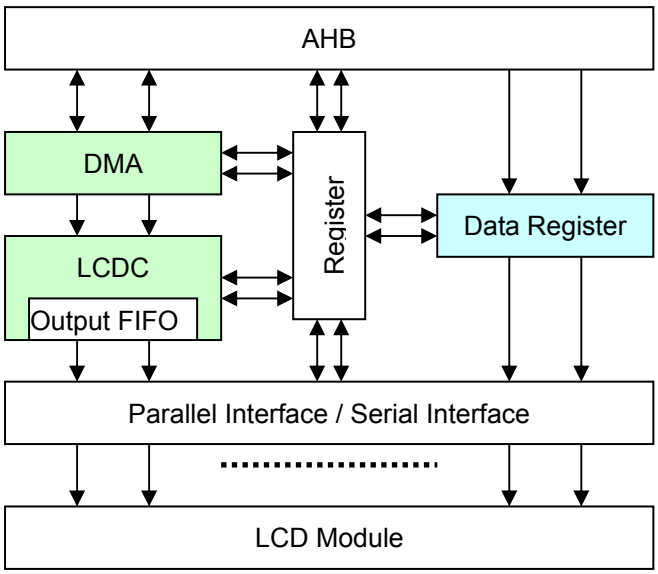
14.1 Overview

The Smart LCD Controller affords an interface to transfer data from the LCD controller to the LCD Module. It supports DMA operation and register operation.

Features:

- Supports a large variety of LCD Module from different vendors
- Supports parallel and serial interfaces
- Supports different size of display panel
- Supports different width of pixel data
- Supports internal DMA operation and register operation
- Supports Write Operation. Read Operation is not supported

14.2 Structure



*Please notice that the command only can transfer by DMA channel 0. No matter the DMA channel 1 or IPU are use or not.

14.3 Pin Description

Table 14-1 SLCD Pins Description

Name	I/O	Description	Interface
SLCD_RS	O	Command/Data Select Signal. The polarity of the signal can be programmable.	Serial: RS Parallel: RS
SLCD_CS	O	Data Sample Signal. The polarity of the signal can be programmable.	Serial: CS Parallel: Sample Data with the edge of CS
SLCD_CLK	O	The clock of SLCD. The polarity of the clock can be programmable.	Serial or not used
SLCD_DAT*1 [17:0]	O	The data of SLCD.	Serial: SLCD_DAT [15] Parallel: SLCD_DAT [17:0] SLCD_DAT [15:0] SLCD_DAT [7:0]
LCD_LO6_O	O	24 bit parallel SLCD RGB (or 24 bit command) low bit ([17:16],[9:8],[1:0]) output.	detail in GPIO spec

NOTE:

***1:** SLCD_DAT [15] is also use as data pin for serial. The SLCD pins are shared with LCDC. You can see the set of register LCDCFG.LCDPIN in LCDC spec.

14.4 Register Description

In this section, we will describe the registers in Smart LCD controller. Following table lists all the registers definition. All register's 32bit address is physical address. And detailed function of each register will be described below.

Name	Description	RW	Reset Value	Address	Access Size
MCFG	SLCD Configure Register	RW	0x0000	0x130500A0	32
MCTRL	SLCD Control Register	RW	0x00	0x130500A4	8
MSTATE	SLCD Status Register	RW	0x00	0x130500A8	8
MDATA	SLCD Data Register	RW	0x00000000	0x130500AC	32

14.4.1 SLCD Configure Register (MCFG)

The register MCFG is used to configure SLCD.

MCFG 0x130500A0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved			DWIDTH			CWIDTH		Reserved			CSPLY	RSPLY	Reserved	CLKPLY	TYPE
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15:13	Reserved	These bits always read 0, and written are ignored.	R
12:10	DWIDTH*1	Data Width.	RW

DWIDTH	Data Width
000	18-bit once Parallel/Serial
001	16-bit once Parallel/Serial
010	8-bit third time Parallel
011	8-bit twice Parallel
100	8-bit once Parallel/Serial
101	24-bit once Parallel
111	9-bit twice Parallel
110	Reserved

*Please notice that you can only use 24-bit parallel command when use 24-bit parallel data. (The command may not 24-bit but need put them as 24-bit in memory(one command use one word))

<table>
<tr><td>9:8</td><td>CWIDTH*1</td><td>Command Width.<table><tr><th>CWIDTH</th><th>Command Width</th></tr><tr><td>00</td><td>16-bit once / 9bit once</td></tr><tr><td>01</td><td>8-bit once</td></tr><tr><td>10</td><td>18-bit once</td></tr><tr><td>11</td><td>24-bit once</td></tr></table>*Please notice that you can only use 24-bit parallel command when use 24-bit parallel data. (The command may not 24-bit but need put them as 24-bit in memory (one command use one word))</td><td>RW</td></tr>
<tr><td>7:5</td><td>Reserved</td><td>These bits always read 0, and written are ignored.</td><td>R</td></tr>
<tr><td>4</td><td>CSPLY</td><td>CS Polarity. (CS initial level will be different from CS Polarity)
0: Active Level is Low
1: Active Level is High</td><td>RW</td></tr>
<tr><td>3</td><td>RSPLY</td><td>RS Polarity.
0: Command RS = 0, Data RS = 1
1: Command RS = 1, Data RS = 0</td><td>RW</td></tr>
<tr><td>2</td><td>Reserved</td><td>These bits always read 0, and written are ignored.</td><td>R</td></tr>
<tr><td>1</td><td>CLKPLY</td><td>LCD_CLK Polarity.
0: Active edge is Falling
1: Active edge is Rising</td><td>RW</td></tr>
<tr><td>0</td><td>TTYPE</td><td>Transfer Type.
0: Parallel
1: Serial</td><td>RW</td></tr>
</table>

NOTE:

*1: The set of DWIDTH and CWIDTH should keep to the rules as follows:

Interface Mode	Command Width	Data Width	Color
Parallel	18-bit	18-bit once	R6G6B6
	16-bit	16-bit once	R5G6B5
		9-bit twice	
	9-bit	9-bit twice	
	8-bit	8-bit once	
		8-bit twice	
		8-bit third times	
Serial	18-bit	18-bit once	
	16-bit	16-bit once	
	9-bit	9bit twice	
	8-bit	8-bit once	
		8-bit twice	
		8-bit third times	

14.4.2 SLCD Control Register (MCTRL)

MCTRL is SLCD Control Register.

MCTRL 0x130500A4

Bit	7	6	5	4	3	2	1	0
	Reserved					DMASTART	DMAMODE	DMATXEN
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7:3	Reserved	These bits always read 0, and written are ignored.	R
2	DMAMODE	SLCD descriptor DMA mode select. 0: DMA will continually transfer data follow descriptor chain 1: DMA will stop when one descriptor finished	
1	DMASTART	Only use when DMAMODE = 1, set 1 to restart DMA transfer.	
0	DMATXEN	SLCD DMA Transfer Enable. This bit is only used for DMA automatic transfer: – This bit starts the automatic transfer of image data from system memory to LCDM. – When DMAC finishes transferring the data, and the MSTATE.BUSY bit is 0, you can clear DMATXEN bit to stop DMA mode.	RW

* Please notice that you can set DMAMODE = 1, use DMASTART to control the data flow. For example you can stop data transfer before you disable LCDC. Wait a whole frame time (16ms in 60HZ) to make sure the whole frame are send to SLCD panel.

14.4.3 SLCD Status Register (MSTATE)

The register of MSTATE is SLCD status register.

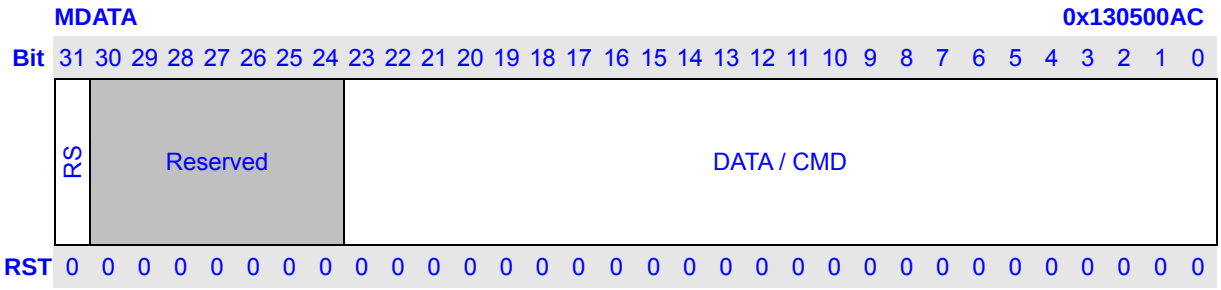
MSTATE 0x130500A8

Bit	7	6	5	4	3	2	1	0
	Reserved							BUSY
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7:1	Reserved	These bits always read 0, and written are ignored.	R
0	BUSY	Transfer is working or not. This bit will be set to 1 when transfer is working. It will be cleared by hardware when transfer is finished. 0: not busy 1: busy	RW

14.4.4 SLCD Data Register (MDATA)

The register MDATA is used to send command or data to LCM. When RS=0, the low 24-bit is used as command. When RS=1, the low 24-bit is used as data.



Bits	Name	Description	RW
31	RS	The RS bit of data register is used to decide the meanings of the low 24-bit. 0: data 1: command	RW
30:24	Reserved	These bits always read 0, and written are ignored.	R
23:0	DATA/CMD	Data or Command Register.	RW

14.5 System Memory Format

14.5.1 Data format

you can configure these registers according to LCDC module.

14.5.2 Command Format

1 18-bit command

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
X	X	X	X	X	X	X	X	X	X	X	X	X	X	C17	C16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
C15	C14	C13	C12	C11	C10	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0

2 16-bit command

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C15	C14	C13	C12	C11	C10	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
C15	C14	C13	C12	C11	C10	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0

3 9-bit command once

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
X	X	X	X	X	X	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
X	X	X	X	X	X	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0

4 8-bit command once

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C7	C6	C5	C4	C3	C2	C1	C0	C7	C6	C5	C4	C3	C2	C1	C0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
C7	C6	C5	C4	C3	C2	C1	C0	C7	C6	C5	C4	C3	C2	C1	C0

5 8-bit command twice (Command = command part + data part)

*Please notice that when you use this kind command, set CWIDTH as 8bit once and set the LCDCNUM.CNUM as doubled the real command number.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
D7	D6	D5	D4	D3	D2	D1	D0	C7	C6	C5	C4	C3	C2	C1	C0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
D7	D6	D5	D4	D3	D2	D1	D0	C7	C6	C5	C4	C3	C2	C1	C0

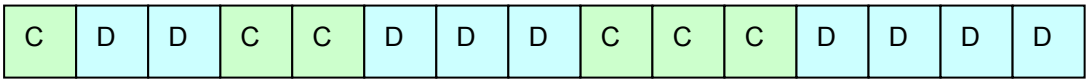
14.6 Transfer Mode

Two transfer modes can be used: DMA/IPU Transfer Mode and Data Register Transfer Mode. In DMA/IPU mode, always transfer commands by DMA 0.

14.6.1 DMA Transfer Mode

Command and data can be recognized by RS bit coming from memory. The format of DMA transfer can be as follows:

1 Command and Data



*Please notice that the command only can insert between two complete frame and the number of command is 0~255.

2 Only Data

*You can also not use command but you still need to use a command descriptor and set the CNUM = 0.

Because DMA transfer mode only can work in OSD mode, you need to configure the panel according OSD mode:

1 Configuration.
* LCDCFG and LCDCTRL
* LCDOSDC and LCDOSDCTRL
* LCDRGBC and LCDIPUR

2 Set Color.
* LCDBGC, LCDKEY0, LCDKEY1, LCDALHPA

3 Set Display.
* LCDVAT, LCDDAH, LCDDAV
* LCDXYP0, LCDXYP1, LCDSIZE0, LCDSIZE1
* LCDVSYNC, LCDHSYNC

4 Set DMAC.
* LCDIID

* LCDDA0, LCDSA0, LCDFID0, LCDCMD0, LCDOFFS0, LCDPW0, LCDCNUM0, LCDDESSIZE0
* LCDDA1, LCDSA1, LCDFID1, LCDCMD1, LCDOFFS1, LCDPW1, LCDCNUM1, LCDDESSIZE1

5 Enable slcd DMA.

6 Enable LCDC.

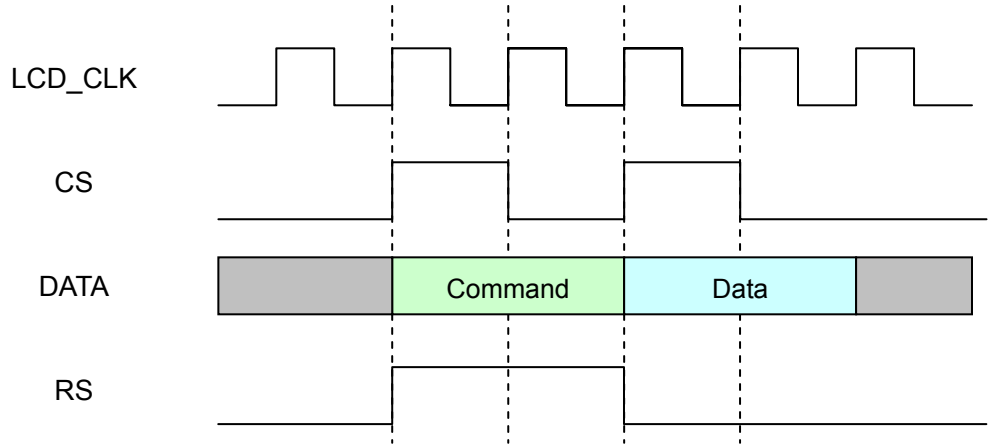
14.6.2 Register Transfer Mode

Each time you can write a command or a data to the register, then it will transfer the RS signal and data or command to LCM. Command and data can be recognized by RS bit coming from data register. The format of data register transfer can be as follows:

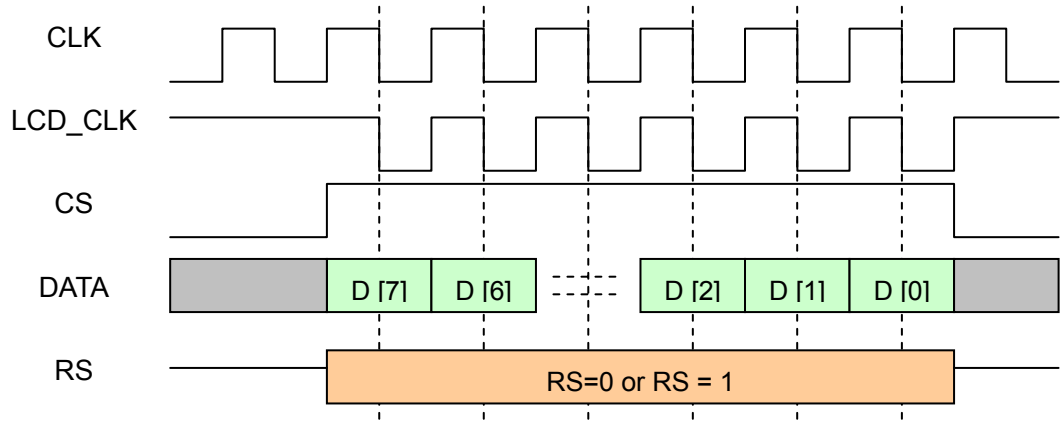
Command	RS [31] = 1	XXX [30:n+1]	Command [n:0]
Data	RS [31] = 0	XXX [30:24]	Data [23:0]
	RS [31] = 0	XXX [30:16]	Data [15:0]
	RS [31] = 0	XXX [30:9]	Data [8:0]
	RS [31] = 0	XXX [30:8]	Data [7:0]

14.7 Timing

14.7.1 Parallel Timing



14.7.2 Serial Timing



14.8 Operation Guide

14.8.1 DMA Operation

1 Start DMA transfer.

a Set LCDCFG.MODE to 1101 to choose LCM.

b Set LCDCTRL.BST to choose burst length for transferring.

c Set register LCDIID0, LCDDA0, LCDSA0, LCDFID0, LCDCMD0, LCDOFFS0, LCDPW0, LCDCNUM0, LCDDESSIZE0 to initial internal DMA.

d Also set register LCDIID1, LCDDA1, LCDSA1, LCDFID1, LCDCMD1, LCDOFFS1, LCDPW1, LCDCNUM1, LCDDESSIZE1 when use DMA channel 1 in OSD mode.

e Set MCFG to configure SLCDC.

f Before starting DMA, Wait for MSTATE.BUSY == 0.

g Set MCTRL.DMATXEN to 1 to prepare DMA transfer.

Note that if you don't want to stop DMA transfer, you need not to check MSTATE.BUSY.

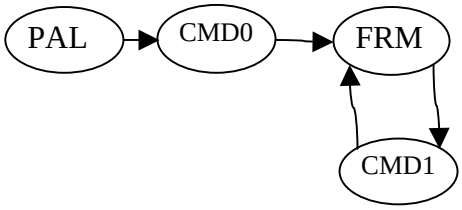
h Set LCDCCTRL.ENA to 1 to start LCDC internal DMA.

i The LCDC internal DMA will transfer data to SLCDC, and SLCDC transfer data to LCM.

Repeat this step till you want to close the SLCDC to transfer data to LCM Panel.

*please notice that use and only use DMA0 to transfer command no matter use DMA0 to transfer frame data or not.

One recommend descriptor chain (CMD0 with CNUM>0 and CMD1 with CNUM=0):



2 Stop DMA transfer.

a Set LCDCCTRL.ENA to 0 to stop LCDC internal DMA at once.

b Wait till MSTATE.BUSY is set to 0 by hardware.

MSTATE.BUSY == 1: there is data in the FIFO waited for transferring to LCM.

MSTATE.BUSY == 0: all data in the FIFO have finished transferring to LCM.

c Set MCTRL.DMATXEN to 0 to stop DMA transfer.

3 Restart DMA transfer.

When MCTRL.DMATXEN is set to 0, and then you want to restart DMA transfer at once, you should ensure that MCTRL.DMATXEN must keep 0 at least three cycles of PIXCLK.

14.8.2 Register Operation

1 Set MCFG to configure SLCD.

2 Wait for MSTATE.BUSY == 0.
3 Set MDATA register.
4 Wait for MSTATE.BUSY == 0.
5 Set MDATA register.
6 Wait for MSTATE.BUSY == 0.
7 … …

15 Image Process Unit

15.1 Overview

IPU (Image process unit) contains Resize and CSC (color space conversion), which is used for image post processing.

15.1.1 Feature

- Location
  - AHB bus
- Input format
  - Separate frame: YUV /YCbCr (4:2:0, 4:2:2, 4:4:4, 4:1:1), RGB
  - Packaged data: YUV422
- Output data format
  - RGB (565, 555, 888)
  - Packaged data YUV422
- Color convention coefficient
  - configurable (CSC enable)
- Minimum input image size (pixel)
  - 2x2
- Maximum input image size (pixel)
  - 4095x4095
- Maximum output image size (pixel)
  - Width: up to 4095 (with no vertical resizing)
    up to 800 (with vertical resizing)
  - Height: up to 4095
- Image resizing
  - Up scaling ratios up to 1:2 in fractional steps with 1/32 accuracy
  - Down scaling ratios up to 32:1 in fractional steps with 1/32 accuracy

**For more details, refer to Special Instruction.*

15.1.2 Block

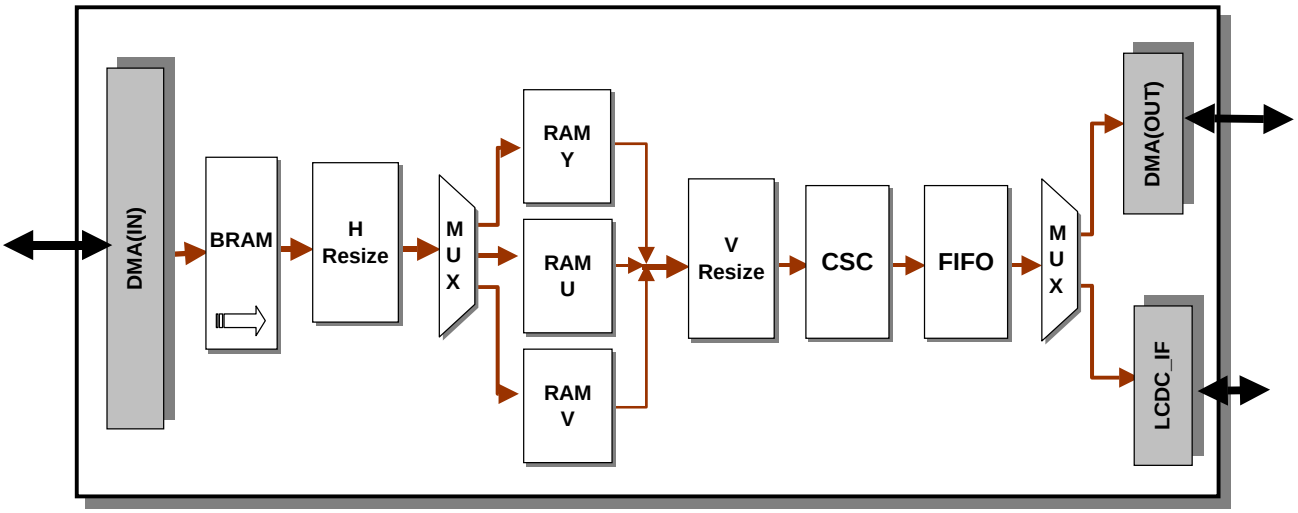


Figure 15-1 The Block about the IPU

15.2 Data flow

15.2.1 Input data

- Separated YUV (or YcbCr/RGB; the following use YUV for convenience) Frame case: Y, U, V data would be fetched from external memory by DMA burst read operation.
- Packaged YUV422 case: Packaged YUV data would be fetched from external memory by DMA burst read operation.

15.2.2 Output data

- DMA output to external memory case: The output data format could be RGB (565, 555, 888) or YUV (package), and the data would be stored to the external memory by DMA burst write operation.
- Flow into LCDC case: The output data format can be RGB or YUV (package), and the transfer would not cross AHB BUS.

15.2.3 Resize Coefficients LUT

The resize coefficients look up table is preset by software according to specific format (see *15.3.23*, *15.3.24*, *15.3.24.1* for detail). There are 2 tables support independent horizontal and vertical scaling. Each table has 32 entries that can accommodate up to 32 coefficients.

15.3 Registers Descriptions

The physical address base for the address-mapped registers of IPU is **0x13080000.**

15.3.1 IPU Control Register

IPU_CONTROL **0x0**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															DFIX_SEL	FIELD_SEL	FIELD_CONF	DISP_SEL	DPAGE_MAP	SPAGE_MAP	LCDC_SEL	SPKG_SEL	V_SCALE	H_SCALE		IPU_RST	FM_IRQ_EN	CSC_EN	VRSZ_EN	HRSZ_EN	IPU_RUN	CHIP_EN
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
31:18	Reserved	Writing has no effect, read as zero.	R
17	DFIX_ SEL	Fixed destination address choose. (valid when LCDC_SEL == 0) 0: not use the fixed address 1: use the fixed address	RW
16	FIELD_SEL *1	Destination field choose. (valid when FIELD_CONF_EN == 1) 0: top field 1: bottom field	RW
15	FIELD_CONF_EN *1	Destination field display configure enable. 0: do not change IPU field display 1: re-configure field read as zero.	W
14	DISP_ SEL	Destination display choose. 0: frame display mode 1: field display mode	RW
13	DPAGE_MAP	Destination address page mapping choose. 0: not use the page mapping 1: use the page mapping	RW
12	SPAGE_MAP	Source address page mapping choose. 0: not use the page mapping 1: use the page mapping	RW
11	LCDC_SEL	Output data destination choose. 0: output to external memory 1: output to LCDC FIFO	RW
10	SPKG_SEL	Input data case choose. 0: Separated YUV Frame	RW

		1: Packaged YUV422	
9	V_SCALE	Vertical direction scale flag. 0: down scaling; 1: up scaling.	RW
8	H_SCALE	Horizontal direction scale flag. 0: down scaling 1: up scaling	RW
7	Reserved	Writing has no effect, read as zero.	R
6	IPU_RST *2	Reset IPU. Writing 1: reset IPU; 0: no effect. Read as zero.	W
5	FM_IRQ_EN	Frame process finish interrupt enable. 1: enable; 0: disable.	RW
4	CSC_EN	CSC enable. 1: enable; 0: disable.	RW
3	VRSZ_EN	Vertical Resize enable. 1: enable; 0: disable.	RW
2	HRSZ_EN	Horizontal Resize enable. 1: enable; 0: disable.	RW
1	IPU_RUN	Run the IPU. 1: run.	RW
0	CHIP_EN	IPU chip enable. 1: enable; 0: disable.	RW

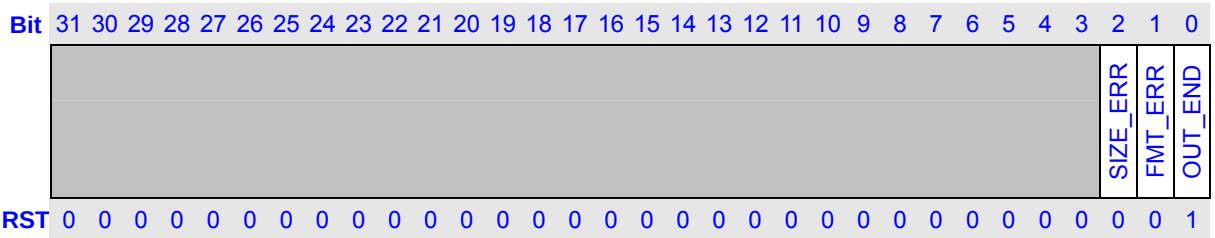
NOTES:

1 *1: The FIELD_SEL will work when the DISP_ SEL is 1, which indicates the IPU is under the field display mode. And the IPU will output the picture from the initial field (top or bottom) to the next field (bottom or top) automatically. The initial field can be configured by setting the FIELD_SEL to 0 or 1 with FIELD_CONF_EN is 1. The FIELD_CONF_EN is just the trigger that controls the FIELD_SEL valuation.

2 *2: Setting 1 to IPU_RST will reset all software visible IPU registers except the CHIP_EN immediately.

15.3.2 IPU Status Register

IPU_STATUS **0x4**



Bits	Name	Description	R/W
31:2	Reserved	Writing has no effect, read as zero.	R
2	SIZE_ERR	The size error flag. 1: size error; 0: size ok.	R
1	FMT_ERR	IPU format error flag. 1: format error; 0: format OK.	R
0	OUT_END	Output termination flag. 1: finished; 0: not finished.	R/W

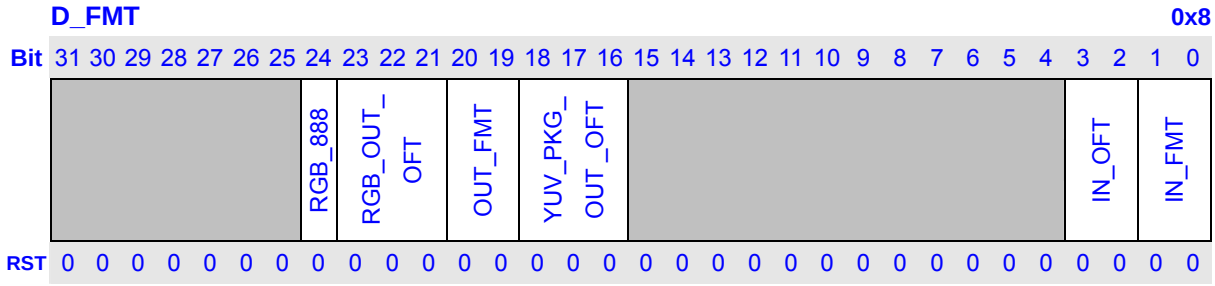
NOTES:

1 If IPU_CONTROL.FM_IRQ_EN has been set 1, once OUT_END is set value 1 which denotes a frame's post process done, an low level active interrupt request will be issued until corresponding software handler read IPU_STATUS and clean end flag.

2 When the IPU_CONTROL.FM_LCDC_SEL has been set 1, and the IPU has finished one transfer, the LCDC and CPU need to occupy the IPU control. The IPU will **monitor** the request signal from **LCDC** (hardware signal) and the read signal from the **CPU** (polling end flag), then it will determine whether re-configure itself by the CPU if the CPU read first or output the same frame to LCDC again if the LCDC get the control. Once the LCDC has occupied the IPU, the OUT_END will **turn to 0** and IPU will **restart again**, automatically. And if the CPU has occupied the IPU, the OUT_END will **not** turn to 0 except the CPU clean it and IPU will **not** restart again except the CPU run it.

3 When the IPU_CONTROL.FM_LCDC_SEL has been set 1, the IPU will output the result data to LCDC directly. Under this condition, the user must be careful when change the IPU parameters, as the LCDC will be under run easily. When turn on the mode that IPU output data to LCDC directly, user **must** run the IPU **first**, and **then** turn on the LCDC. And when need to turn off this mode, user **must** turn off the data channel between IPU and LCDC by turn off the parameter in **LCDC first** and **then polling** the end flag of IPU to occupy the control to IPU.

15.3.3 Data Format Register



Bits	Name	Description	R/W
31:25	Reserved	Writing has no effect, read as zero.	R
24	RGB_888_OUT_FMT	RGB888 output format indicator. (only used in RGB888 out) 0: the low 24 bits will be the pixel in a word 1: the high 24 bits will be the pixel in a word	RW
23:21	RGB_OUT_OFT	Output data packaged offset. (only used in RGB out) 000: RGB 001: RBG 010: GBR 011: GRB 100: BRG 101: BGR	RW

		Others: reserved	
20:19	OUT_FMT	Indicates the destination data format. 00: RGB555 01: RGB565 10: RGB888 11: YUV422 package	RW
18:16	YUV_PKG_ OUT_OFT	Output data packaged offset. (only used in CSC disable case and in the YUV422 packaged case) 000: Y1UY0V 001: Y1VY0U 010: UY1VY0 011: VY1UY0 100: Y0UY1V 101: Y0VY1U 110: UY0VY1 111: VY0UY1	RW
15: 4	Reserved	Writing has no effect, read as zero.	R
3:2	IN_OFT	Input data packaged offset. (only used in YUV422 packaged case) 00: Y1UY0V 01: Y1VY0U 10: UY1VY0 11: VY1UY0	RW
1:0	IN_FMT	Indicates the source data format. 00: YUV 4:2:0 01: YUV 4:2:2 10: YUV 4:4:4 11: YUV 4:1:1	RW

15.3.4 Input Y Data Address Register

Y_ADDR **0xc**

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

Bits	Name	Description	R/W
31:0	Y_ADDR *1	In separated Frame case, it indicates the source Y data buffer's start address.	RW

		In YUV422 package case, it indicates the start address of the packaged Frame.	

NOTES:

1 When the IPU_CONTROL.SPAGE_MAP == 1, the Y_ADDR should be the **low 12** bits of the start virtual address.

2 Y_ADDR should be word align.

15.3.5 Input U Data Address Register

U_ADDR **0x10**

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

Bits	Name	Description	R/W
31:0	U_ADDR *1	The source U data buffer's start address of separated frame case.	RW

NOTES:

1 When the IPU_CONTROL.SPAGE_MAP == 1, the U_ADDR should be the **low 12** bits of the start virtual address.

2 U_ADDR should be word align.

15.3.6 Input V Data Address Register

V_ADDR **0x14**

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

Bits	Name	Description	R/W
31:0	V_ADDR	The source V data buffer's start address of separated Frame case.	RW

NOTES:

1 When the IPU_CONTROL.SPAGE_MAP == 1, the V_ADDR should be the **low 12** bits of the start virtual address.

2 V_ADDR should be word align.

15.3.7 Input Y physics table address

Y_PHY_T_ADDR **0x54**

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

Bits	Name	Description	R/W
31:0	Y_PHY_T_ ADDR	The start address of the physics-mapping table about the Y data. (This register will act when the **IPU_CONTROL.PAGE_MAP is valid)**	RW

15.3.8 Input U physics table address

U_PHY_T_ADDR **0x58**

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

Bits	Name	Description	R/W
31:0	U_PHY_T_ ADDR	The start address of the physics-mapping table about the U data. (This register will work when the **IPU_CONTROL.PAGE_MAP is valid)**	RW

15.3.9 Input V physics table address

V_PHY_T_ADDR **0x5c**

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

Bits	Name	Description	R/W
31:0	V_PHY_T_ ADDR	The start address of the physics mapping table about the V data. (This register will work when the **IPU_CONTROL.PAGE_MAP is valid**)	RW

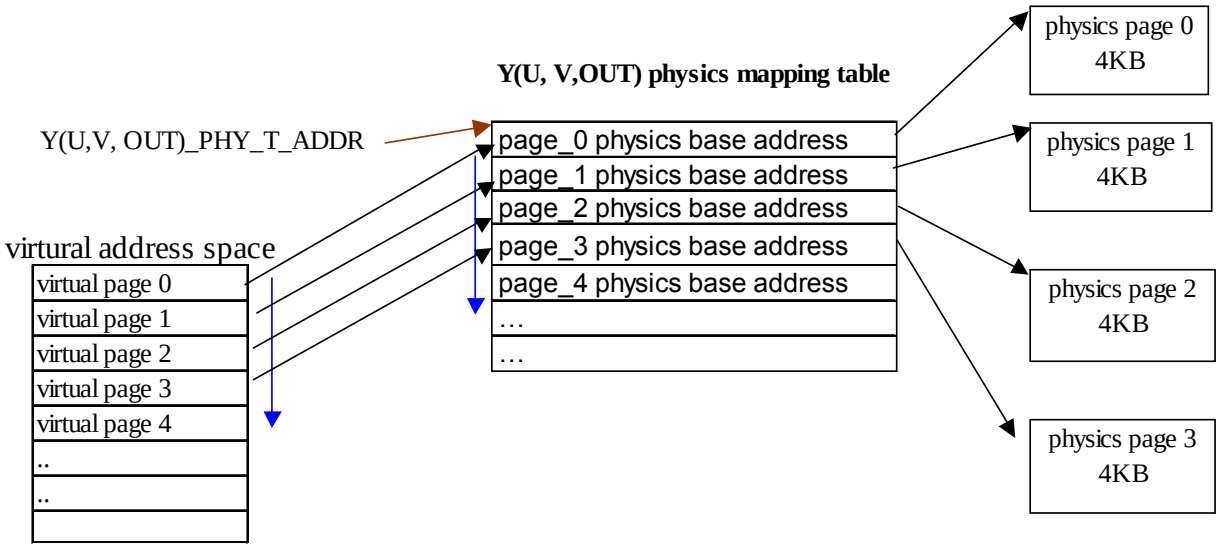
15.3.10 OUT physics table address

OUT_PHY_T_ADDR **0x60**

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

Bits	Name	Description	R/W
31:0	OUT_PHY_T_ ADDR	The start address of the physics mapping table about the data which will be DMA out. (This register will work when the **IPU_CONTROL.PAGE_MAP is valid)**	RW

15.3.10.1 ADDRESS Mapping



The Y (U, V, OUT)_PHY_T_ADDR should store the **base address** of the Y (U, V, OUT) physics-mapping table. In the Y (U, V, OUT) physics-mapping table, it should contain different physics page base address, and the physics page must be 4KB align.

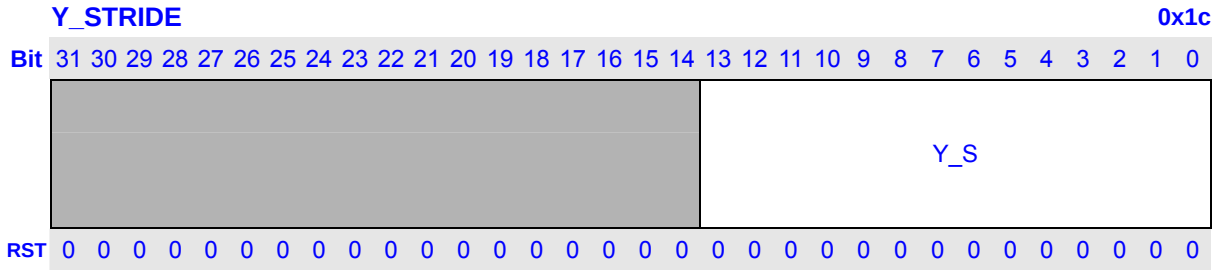
15.3.11 Input Geometric Size Register

IN_FM_GS **0x18**

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

Bits	Name	Description	R/W
31:28	Reserved	Writing has no effect, read as zero.	R
27:16	IN_FM_W	The width of the input frame (unit: byte). Y data width is the same as this value while U/V or Cb/Cr data width should do relatively zoom in according to the source data format. And **in the package pattern**, this value should be the Y data width also.	RW
15:12	Reserved	Writing has no effect, read as zero.	R
11:0	IN_FM_H	The height of the input frame (unit: byte). Y data width is same as this value while U/V or Cb/Cr data width should do relatively zoom in according to the source data format.	RW

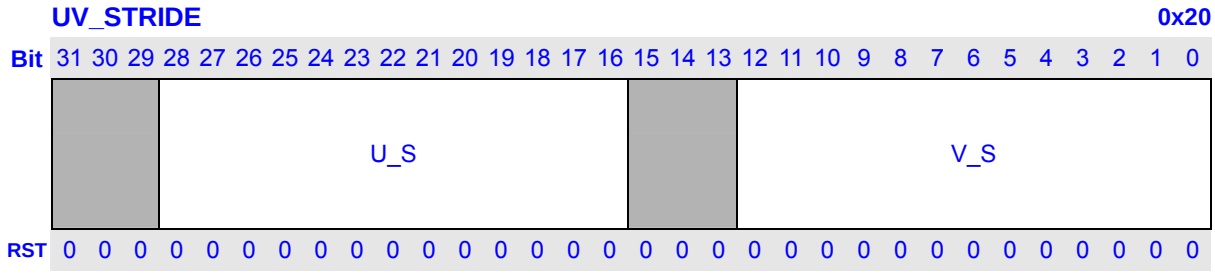
15.3.12 Input Y Data Line Stride Register



Bits	Name	Description	R/W
31:14	Reserved	Writing has no effect, read as zero.	R
13:0	Y_S	The line stride of the source Y data in the external memory of separated Frame case or packaged YUV Frame stride. (Unit: byte)	RW

NOTE: Y_S should be world align.

15.3.13 Input UV Data Line Stride Register



Bits	Name	Description	R/W
31:29	Reserved	Writing has no effect, read as zero.	R
28:16	U_S	The line stride of the source U data in the external memory. (Unit: byte)	RW
15:13	Reserved	Writing has no effect, read as zero.	R
12:0	V_S	The line stride of the source V data in the external memory. (Unit: byte)	RW

NOTE: U_S and V_S should be word align.

15.3.14 Output Frame Start Address Register

OUT_ ADDR **0x24**

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

Bits	Name	Description	R/W
31:0	OUT_ ADDR *1	The output buffer's start address.	RW

NOTES:

1 *1: When the IPU_CONTROL.DPAGE_MAP == 1, the OUT_ADDR should be the low 12 bits of the start virtual address.

2 it should be word align.

15.3.15 Output Geometric Size Register

OUT_GS **0x28**

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

Bits	Name	Description	R/W
31	Reserved	Writing has no effect, read as zero.	R
30:16	OUT_FM_W	The width of the output destination frame (unit: byte).	RW
15:13	Reserved	Writing has no effect, read as zero.	R
12:0	OUT_FM_H	The height of the output destination frame (unit: byte).	RW

NOTES:

1 In the **package** out pattern, the OUT_FM_W should be the **pixel number** in a line.

2 In the RGB out pattern, the OUT_FM_W should be the data space width in the RAM.

3 In the out package pattern, the OUT_FM_W should better be even number, else IPU will fill the last Y pixel result with the last second Y pixel automatically.

For example:

when the OUT_FM_W is an odd number (A), and the result will be like that: Y0, U, Y0, V

And when the OUT_FM_W is an even number (A+1), and the result is Y1, U, Y0, V

15.3.16 Output Data Line Stride Register

OUT_STRIDE **0x2c**

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

Bits	Name	Description	R/W
31:16	Reserved	Writing has no effect, read as zero.	R
15:0	OUT_S	The line stride of the destination data buffer in the external memory. (Unit: byte)	RW

15.3.17 Resize Coefficients Table Index Register

RSZ_COEF_INDEX **0x30**

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

Bits	Name	Description	R/W
31:21	Reserved	Writing has no effect, read as zero.	R
20:16	HE_IDX *1	Indicates the end address of the horizontal resize look up table.	RW
15:5	Reserved	Writing has no effect, read as zero.	R
4:0	VE_IDX *1	Indicates the end address of the vertical resize look up table.	RW

NOTE: The HE_IDX (VE_IDX) should be the depth of the horizontal (vertical) resize look up table **minus 1**.

15.3.18 CSC C0 Coefficient Register

CSC_C0_COEF **0x34**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																					C0_COEF										
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	C0_COEF	The C0 coefficient of the YUV/YCbCr to RGB conversion. C0_COEF = [C0 * 1024 + 0.5]	RW
NOTE: R = C0*(Y –LUMA_OF) + C1*(Cr-CHROM_OF) G = C0*(Y – LUMA_OF) – C2*(Cb-CHROM_OF) – C3*(Cr-CHROM_OF) B = C0*(Y – LUMA_OF) + C4*(Cb-CHROM_OF)			

15.3.19 CSC C1 Coefficient Register

CSC_C1_COEF **0x38**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																					C1_COEF										
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	C1_COEF	The C1 coefficient of the YUV/YCbCr to RGB conversion. C1_COEF = [C1 * 1024 + 0.5]	RW
NOTE: R = C0*(Y – LUMA_OF) + C1*(Cr-CHROM_OF) G = C0*(Y – LUMA_OF) – C2*(Cb-CHROM_OF) – C3*(Cr-CHROM_OF) B = C0*(Y – LUMA_OF) + C4*(Cb-CHROM_OF)			

15.3.20 CSC C2 Coefficient Register

CSC_C2_COEF 0x3C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																					C2_COEF											
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	C2_COEF	The C2 coefficient of the YUV/YCbCr to RGB conversion. C2_COEF = [C2 * 1024 + 0.5]	RW

NOTE:

R = C0*(Y – LUMA_OF) + C1*(Cr-CHROM_OF)

G = C0*(Y – LUMA_OF) – C2*(Cb-CHROM_OF) – C3*(Cr-CHROM_OF)

B = C0*(Y – LUMA_OF) + C4*(Cb-CHROM_OF)

15.3.21 CSC C3 Coefficient Register

CSC_C3_COEF 0x40

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																					C3_COEF											
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	C3_COEF	The C3 coefficient of the YUV/YCbCr to RGB conversion. C3_COEF = [C3 * 1024 + 0.5]	RW

NOTE:

R = C0*(Y – LUMA_OF) + C1*(Cr-CHROM_OF)

G = C0*(Y – LUMA_OF) – C2*(Cb-CHROM_OF) – C3*(Cr-CHROM_OF)

B = C0*(Y – LUMA_OF) + C4*(Cb-CHROM_OF)

15.3.22 CSC C4 Coefficient Register

CSC_C4_COEF 0x44

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																					C4_COEF										
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
31:12	Reserved	Writing has no effect, read as zero.	R
11:0	C4_COEF	The C4 coefficient of the YUV/YCbCr to RGB conversion. C4_COEF = [C4 * 1024 + 0.5]	RW
NOTE: R = C0*(Y – LUMA_OF) + C1*(Cr-CHROM_OF) G = C0*(Y – LUMA_OF) – C2*(Cb-CHROM_OF) – C3*(Cr-CHROM_OF) B = C0*(Y – LUMA_OF) + C4*(Cb-CHROM_OF)			

15.3.23 Horizontal Resize Coefficients Look Up Table Register group

HRSZ_COEF_LUT 0x48

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

Bits	Name	Description	R/W
31:13	Reserved	Writing has no effect, read as zero.	R
12	START	This bit will indicate the horizontal coefficient writing start. The IPU will reset the entire horizontal coefficient and waiting the new coefficient writing.	W

11:2	W_COEF	Weighting coefficients, 10 bits length, that is to say the precision is 1/512.
		For up-scaling,
		W_k = 1 – (k*n/m – [k*n/m]), k = 0, 1, … m-1.
		For down-scaling,
		for (t=0, k=0; k < n; k++) {
		If ([(t*n+1)/m] – k >=1) { W_k = 0;}
		else if ((t*n+1)/m – k == 0) { W_k = 1; t++;}
			W

		else { W_k = 1 – ((t*n+1)/m – [t*n/m]); t++;} } W_COEF_k = [512 * W_k] (stands for get the rounding integer, [20.33] = 20 while [20.66] = 21) Here n stands for original pixel points, m stands for pixel points after resize. For example down-scaling 5:3, n = 5, m = 3. Moreover, m and n are prime, that is, for example 8:2 should be converted to 4:1. When IPU_CONTROL.RSZ_EN set as 1 and m:n = 1:1, all coefficients should be calculated as up-scale case.	

1	IN_EN	Flag for whether new pixel would be used.
		IN_EN = 0, no new pixel
		IN_EN = 1, one new pixel
		In down scale case, IN_EN always equals 1.
		In up scale case,
		For (i=0, k=0; k < m; k++) {
		If(i<= k*n/m) { IN_EN_k = 1; i++;}
		else { IN_EN_k =0;}
		}
			RW

0	OUT_EN	Flag for whether current interpolation would be output.
		OUT_EN = 0, current interpolation would not be output
		OUT_EN = 1, current interpolation would be output
		In up scale case, OUT_EN always equals 1.
		In down scale case,
		For (t=0, k=0; k < n; k++) {
		If([(t*n+1)/m] – k >=1)
		OUT_EN_k = 0;
		else {OUT_EN_k =1; t++;}
		}
			RW

NOTE:

The coefficient number equals to max (m, n). HLUT (horizontal look up table) and VLUT (vertical look up table) are independent, so the two tables may have different coefficient number. Therefore,

RSZ_COEF_INDEX.VIDX = The coefficient number of VLUT – 1

RSZ_COEF_INDEX.HIDX = The coefficient number of HLUT – 1

Moreover, when m=1 for down-scaling, discard above formula and use following rules:

a W_COEF_0 = 256 (W_0 = 0.5), and $W\_COEF_{1 \sim n-1}$ = 0

b IN_EN always equals 1

c OUT_EN_0 = 1, and $OUT\_EN_{1 \sim n-1}$ = 0

Following are two examples of setting LUT:

a Resize coefficients for 7:3

W	W_COEF	IN_EN	OUT_EN	Pixel 1	Pixel 2	OUT
2/3	341	1	1	P [0]	P [1]	P [0] * 2/3 + P [1] * 1/3
0	0	1	0	P [1]	P [2]	No new pixel out
1/3	171	1	1	P [2]	P [3]	P [2] * 1/3 + P [3] * 2/3
0	0	1	0	P [3]	P [4]	No new pixel out
0	0	1	0	P [4]	P [5]	No new pixel out
1	512	1	1	P [5]	P [6]	P [5] * 1 + P [6] * 0
0	0	1	0	P [6]	P [7]	No new pixel out

b Resize coefficients for 3:5

W	W_COEF	IN_EN	OUT_EN	Pixel 1	Pixel 2	OUT
1	512	1	1	P [0]	P [1]	P [0] * 1 + P [1] * 0
2/5	205	0	1	P [0]	P [1]	P [0] * 2/5 + P [1] * 3/5
4/5	410	1	1	P [1]	P [2]	P [1] * 4/5 + P [2] * 1/5
1/5	102	0	1	P [1]	P [2]	P [1] * 1/5 + P [2] * 4/5
3/5	307	1	1	P [2]	P [3]	P [2] * 3/5 + P [3] * 2/5

15.3.24 Vertical Resize Coefficients Look Up Table Register group

VRSZ_COEF_LUT **0x4C**

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

*Function descriptions are same as horizontal LUT.

15.3.24.1 Calculation for Resized width and height

For software, to preset correct value for register OUT_GS, please refer to following formula.
Set IW stand for original input frame width, IH stand for original input frame height, OW stand for new frame width after resize, OH stand for new frame height after resize.

In Up-scale case (n < m):

If [(IW - 1) * (m/n)] * (n/m) ==(IW-1) then

OW = [(IW - 1) * (m/n)] + 1;

Else OW = [(IW - 1) * (m/n)] + 2;
If [(IH - 1) * (m/n)] * (n/m) == (IH-1) then
OH = [(IH - 1) * (m/n)] + 1;
Else OH = [(IH - 1) * (m/n)] + 2;

In Down-scale case (n>m):

If [(IW - 1) * (m/n)] * (n/m) ==(IW-1) then
OW = [(IW - 1) * (m/n)];
Else OW = [(IW - 1) * (m/n)] + 1;
If [(IH - 1) * (m/n)] * (n/m) ==(IH-1) then
OH = [(IH - 1) * (m/n)];
Else OH = [(IH - 1) * (m/n)] + 1;

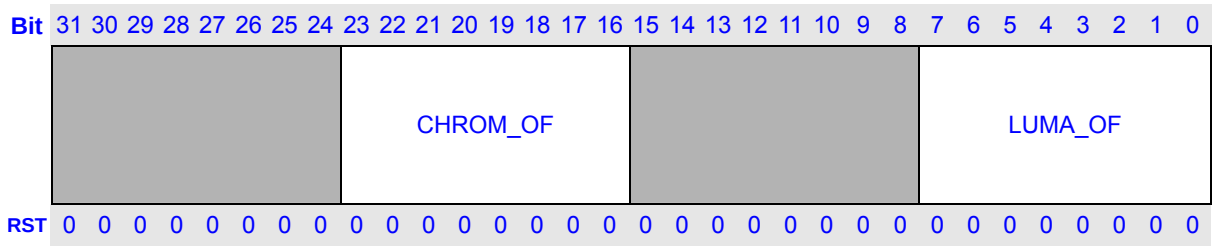
For example:

A 36x46 frame with the horizontal resize ratio of 4:5 (up-scale) and vertical resize ratio of 7:6 (down-scale), by the expressions above we get its new size after resize from the following process.

For Width: [(36 - 1) * (5/4)] * (4/5) = 34.4 ≠ (36-1)
So OW = [(36 - 1) * (5/4)] + 2 = 45
For Height: [(46 - 1) * (6/7)] * (7/6) = 44.33 ≠ (46 - 1)
So OH = [(46 - 1) * (6/7)] + 1 = 39

15.3.25 CSC Offset Parameter Register

CSC_OFSET_PARA **0x50**



Bits	Name	Description	R/W
31:24	Reserved	Writing has no effect, read as zero.	R
23:16	CHROM_OF	Chroma offset value.	RW
15:8	Reserved	Writing has no effect, read as zero.	R
7:0	LUMA_OF	Luma offset value.	RW

NOTE:

R = C0*(Y – LUMA_OF) + C1*(Cr-CHROM_OF)
G = C0*(Y – LUMA_OF) – C2*(Cb-CHROM_OF)– C3*(Cr-CHROM_OF)
B = C0*(Y – LUMA_OF) + C4*(Cb-CHROM_OF)

15.4 IPU Operation Flow

15.4.1 CONTROL SET

Step No.	Action	Register	Note
Base	Enable IPU Chip (IPU_CONTROL)	– CHIP_EN	
0	Read the END flag and do the next steps (IPU_STATUS)	– Wait (END == 1)	**Here you can also clean the END flag, but for** low power **consumer, we do not commend this.**
1	Set IPU primary control (IPU_CONTROL)	– VRSZ_EN – HRSZ_EN – CSC_EN – IRQ_EN	For saving power, when there is no vertical or horizontal resize, disable it.
2	Set IPU source data control (IPU_CONTROL)	– Source data package SPKG_SEL – Source address mapping SPAGE_MAP	
3	Set IPU destination data control (IPU_CONTROL)	– Output data destination LCDC_SEL – Destination address mapping DPAGE_MAP – Destination display mode DISP_ SEL – Initial field select FIELD_SEL	
4	Set scale flag (IPU_CONTROL)	– V_SCALE – H_SCALE	

15.4.2 FORMAT SET

Step No.	Action	Register	Note
5	Set IPU frame format (D_FMT)	– Source data format IN_FMT	It just works when the source is not package. (IPU_CONTROL.SPKG_SEL =0)

		– Input data packaged offset IN_OFT	It just works when the source is package. (IPU_CONTROL.SPKG_SEL =1)
		– Output data packaged offset YUV_PKG_OUT_OFT	Only used in CSC disable case and in the packaged case.
		– Destination data format OUT_FMT	
		– Output data packaged offset RGB_OUT_OFT	Only used in RGB output case.
		– RGB888 out format RGB_888_OUT_FMT	Only used in RGB 888 output case.

15.4.3 INPUT FRAME INFORMATION SET

Step No.	Action	Register	Note
6	Set input frame size (IN_FM_GS)	– Input frame width IN_FM_W	**In the package pattern, this value should be the Y data width also.**
		– Input frame height IN_FM_H	
7	Set input frame stride	– Y frame stride Y_STRIDE.Y_S	
		– U frame stride UV_STRIDE.U_S	
		– V frame stride UV_STRIDE.V_S	
8	Set input data start address	– Y frame start address Y_ADDR	**In the address mapping mode, the address should be the low 12 bits of the first virtual base address.**
		– U frame start address U_ADDR	
		– V frame start address V_ADDR	

15.4.4 OUTPUT FRAME INFORMATION SET

Step No.	Action	Register	Note
9	Set output frame size (OUT_GS)	– Output frame width OUT_FM_W	In the out package pattern, the OUT_FM_W should better be even number.
		– Output frame height OUT_FM_H	
10	Set output frame stride	– Output frame stride OUT_STRIDE	
11	Set output data start address	– Output frame start address OUT_ ADDR	**In the address mapping mode, the address should be the low 12 bits of the virtual base address.**

15.4.5 ADDRESS MAPPING SET

Step No.	Action	Register	Note
12	Set the start address of the address mapping table for source address	– Y_PHY_T_ADDR – U_PHY_T_ADDR – V_PHY_T_ADDR	It just works when the source address mapping.
13	Set the start address of the address mapping table for destination address	– OUT_PHY_T_ADDR	It just works when the output address mapping.

15.4.6 CSC SET

Step No.	Action	Register	Note
14	Set CSC coefficients	- CSC_C0_COEF - CSC_C1_COEF - CSC_C2_COEF - CSC_C3_COEF - CSC_C4_COEF	It just works when CSC is enable.
15	Set CSC offset (CSC_OFSET_PARA)	- LUMA_OF - CHROM_OF	It just works when CSC is enable.

15.4.7 RESIZE TABLE SET

Step No.	Action	Register	Note
16	Set resize coefficients table index register RSZ_COEF_INDEX	- Vertical LUT VE_IDX - Horizontal LUT HE_IDX	The HE_IDX (VE_IDX) should be the depth of the horizontal (vertical) resize look up table **minus 1.**
17	Start vertical direction look-up set VRSZ_COEF_LUT	- START =1	**Before initial the VRST LUT, this step is** necessary.
18	Set Vertical direction Look-Up Table: VRSZ_COEF_LUT	- VRSZ_COEF_LUT	
19 *	Start horizontal direction look-up set HRSZ_COEF_LUT	- START =1	**Before initial the HRST LUT, this step is** necessary.
20	Set Horizontal direction Look-Up Table: HRSZ_COEF_LUT	- HRSZ_COEF_LUT	

15.4.8 RUN IPU && WAIT END

21	Clean_end_flag and RUN IPU	Clean_end_flag(); Start_ipu();	
Next picture operation again			

15.5 Operation example

Table 15-1 no mapping mode

Step	Action
Base	Chip_enable();
0	**Do { } while {!polling_end_flag}**
1	set_primary_ctrl(VRSZ_ENABLE, HRSZ_ENABLE, CSC_EN, irq_en); //
2	set_source_ctrl(source_pkg_sel, SPAGE_SEL);
3	set_out_ctrl(lcdc_sel, DPAGE_SEL, DISP_SEL, FIELD_SEL, FIELD_CONF_EN);
4	set_scale_ctrl(V_SCALE, H_SCALE);
5	set_ipu_fmt(RGB888_OUT_FMT, OUT_OFT_RGB, OUT_FMT, OUT_Y1UY0V, IN_OF_YUYV, IN_FM_YUV444);
6	set_inframe_gsize(FIN_W, FIN_H, FIN_Y_STRIDE, FIN_U_STRIDE, FIN_V_STRIDE);

7	set_y_addr((unsigned int)fin_y & 0x1FFFFFFF);
	set_u_addr((unsigned int)fin_y & 0x1FFFFFFF);
	set_v_addr((unsigned int)fin_y & 0x1FFFFFFF);

8	set_outframe_gsize(FOUT_W, FOUT_H , FOUT_STRIDE);
9	set_out_addr((unsigned int)fout & 0x00000FFF);

10	set_csc_c0(YUV_CSC_C0);
	set_csc_c1(YUV_CSC_C1);
	set_csc_c2(YUV_CSC_C2);
	set_csc_c3(YUV_CSC_C3);
	set_csc_c4(YUV_CSC_C4);

11	set_csc_ofset_para (128, 0);
12	set_rsz_lut_end(H_MAX_LUT-1, V_MAX_LUT-1);
13	start_hlut_coef_write(); **NOTE:** This step is necessary before write new LUT.

14	for (i=0;i<H_MAX_LUT;i++) {
	set_hrsz_lut_coef(h_lut[i].coef, h_lut[i].in_n, h_lut[i].out_n);
	}

15	start_vlut_coef_write(); **NOTE:** This step is necessary before write new LUT.

16	for (i=0;i<V_MAX_LUT;i++) {
	set_vrsz_lut_coef(v_lut[i].coef, v_lut[i].in_n, v_lut[i].out_n);
	}

17	Clean_end_flag(); run_ipu();

Table 15-2 mapping mode

Step	Action

Prepare	y_phy_table[0] = ((unsigned int)fin_y & 0x0FFFF000) | 0x20000000;
	u_phy_table[0] = ((unsigned int)fin_u & 0x0FFFF000) | 0x20000000;
	v_phy_table[0] = ((unsigned int)fin_v & 0x0FFFF000) | 0x20000000;
	out_phy_table[0] = ((unsigned int)fout & 0x0FFFF000) | 0x20000000;
	for (i =1; i<100; i++){
	y_phy_table[i] = y_phy_table[i-1] + 4096;
	u_phy_table[i] = u_phy_table[i-1] + 4096;
	v_phy_table[i] = v_phy_table[i-1] + 4096;
	out_phy_table[i] = out_phy_table[i-1] + 4096;
	}

Base	Chip_enable();
0	**Do { } while {!polling_end_flag}**
1	set_primary_ctrl(VRSZ_ENABLE, HRSZ_ENABLE, CSC_EN, irq_en); //
2	set_source_ctrl(source_pkg_sel, SPAGE_SEL);
3	set_out_ctrl(lcdc_sel, DPAGE_SEL, DISP_SEL, FIELD_SEL, FIELD_CONF_EN);
4	set_scale_ctrl(V_SCALE, H_SCALE);
5	set_ipu_fmt(RGB888_OUT_FMT, OUT_OFT_RGB, OUT_FMT, OUT_Y1UY0V , IN_OF_YUYV, IN_FM_YUV444);
6	set_inframe_gsize(FIN_W, FIN_H, FIN_Y_STRIDE, FIN_U_STRIDE, FIN_V_STRIDE);

7	set_y_addr((unsigned int)fin_y & 0xFFF);
	set_u_addr((unsigned int)fin_y & 0xFFF);
	set_v_addr((unsigned int)fin_y & 0xFFF);

8	set_outframe_gsize(FOUT_W, FOUT_H , FOUT_STRIDE);
9	set_out_addr((unsigned int)fout & 0x00000FFF);

10	set_y_phy_t_addr((unsigned int)y_phy_table & 0x1FFFFFFF);
	set_u_phy_t_addr((unsigned int)u_phy_table & 0x1FFFFFFF);
	set_v_phy_t_addr((unsigned int)v_phy_table & 0x1FFFFFFF);
	set_out_phy_t_addr((unsigned int)out_phy_table & 0x1FFFFFFF);

11	set_csc_c0(YUV_CSC_C0);
	set_csc_c1(YUV_CSC_C1);
	set_csc_c2(YUV_CSC_C2);
	set_csc_c3(YUV_CSC_C3);
	set_csc_c4(YUV_CSC_C4);

12	set_csc_ofset_para (128, 0);
13	set_rsz_lut_end(H_MAX_LUT-1, V_MAX_LUT-1);
14	start_hlut_coef_write(); **NOTE:** This step is necessary before write new LUT.

15	for (i=0;i<H_MAX_LUT;i++) {
	set_hrsz_lut_coef(h_lut[i].coef, h_lut[i].in_n, h_lut[i].out_n);
	}

16	start_vlut_coef_write(); **NOTE:** This step is necessary before write new LUT.

17	for (i=0;i<V_MAX_LUT;i++) {
	set_vrsz_lut_coef(v_lut[i].coef, v_lut[i].in_n, v_lut[i].out_n);
	}

18	Clean_end_flag(); run_ipu();

15.6 Special Instruction

A1. Resizing size feature

Input size (W x H)		Output size (W x H)	
Min	2x2	Disable vertical scale	Min: 2x2
			Max: 4095x4095
Max	4095x4095	Enable vertical scale	Min: 2x2
			Max: 800x4095

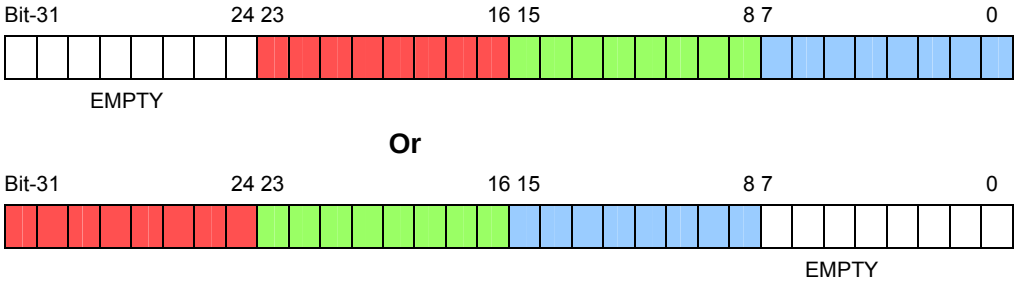
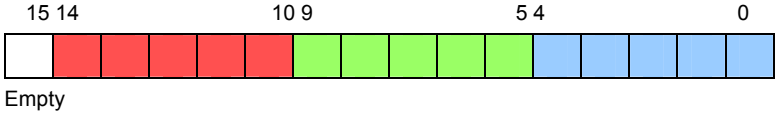
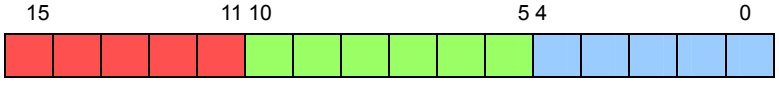
A2. Color convention feature

Source format	Output format	Parameter configure (necessary)
RGB	RGB	IPU_CONTROL.CSC_EN =0 IPU_CONTROL. SPKG_SEL = 0
		D_FMT. IN_FMT = 2'b10 (YUV 4:4:4)
		D_FMT.OUT_FMT = 2'b00, 2'b01, 2'b10
YUV	RGB	IPU_CONTROL.CSC_EN =1
		IPU_CONTROL. SPKG_SEL
		D_FMT. IN_FMT
		D_FMT. IN_OFT ***(IPU_CONTROL. SPKG_SEL == 1)***
		D_FMT.OUT_FMT = 2'b00, 2'b01, 2'b10
		D_FMT.RGB_OUT_OFT
		CSC_C0 (1,2,3,4)_COEF, CSC_OFSET_PARA
YUV	YUV (package)	IPU_CONTROL.CSC_EN =0
		IPU_CONTROL. SPKG_SEL
		D_FMT. IN_FMT
		D_FMT. IN_OFT ***(IPU_CONTROL. SPKG_SEL == 1)***
		D_FMT.OUT_FMT = 2'b11

A3. YUV/YCbCr to RGB CSC Equations

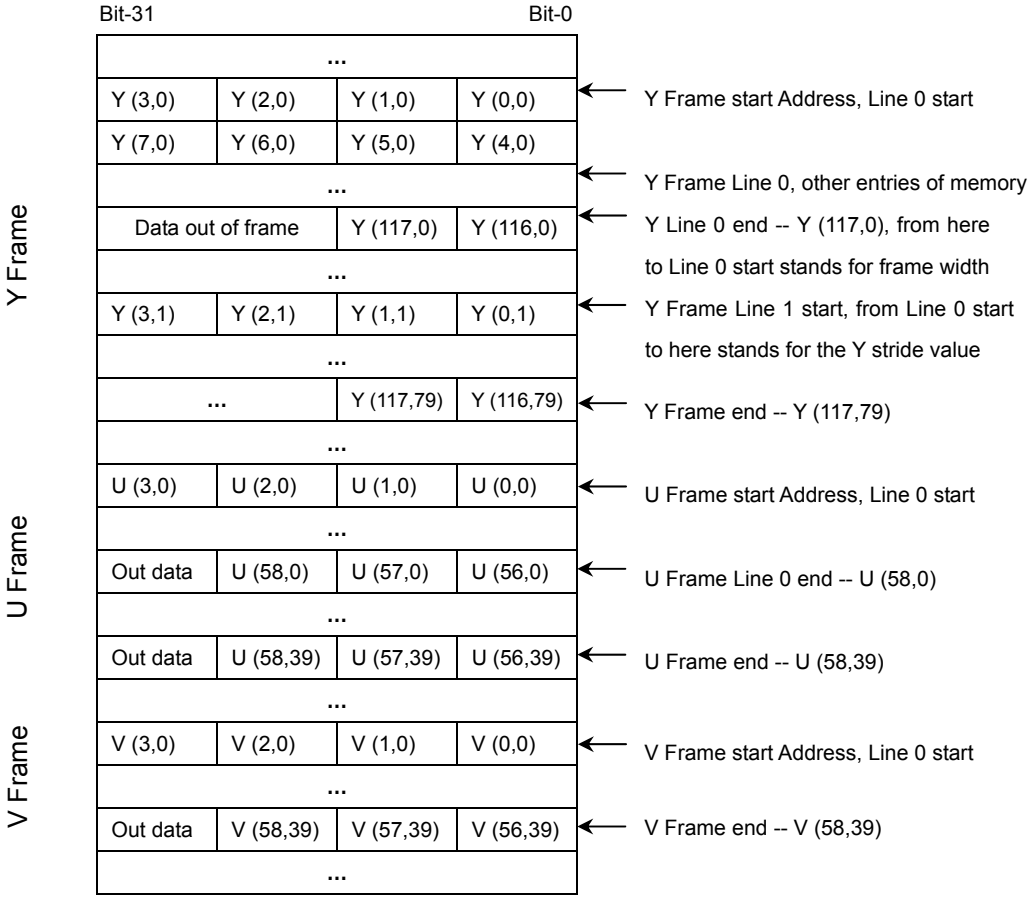
Input data	Matrix	CSC_COEF
YUV	R = C0*(Y – X0) + C1*(V-128) G = C`0*(Y – X0) – C2*(U-128) – C3*(V-128) B = C0*(Y – X0) + C4*(U-128) X0: 0 C0: 1 C1: 1.4026 C2: 0.3444 C3: 0.7144 C4: 1.7730	CSC_C0_COEF = 0x400 **CSC_C1_COEF= 0x59C** CSC_C2_COEF = 0x161 CSC_C3_COEF = 0x2DC CSC_C4_COEF = 0x718
YCbCr	R = C0*(Y – X0) + C1*(Cr-128) G = C0*(Y – X0) – C2*(Cb-128) – C3*(Cr-128) B = C0*(Y – X0) + C4*(Cb-128) X0: 16 C0: 1.164 C1: 1.596 C2: 0.391 C3: 0.813 C4: 2.018	CSC_C0_COEF = 0x4A8 CSC_C1_COEF = 0x662 CSC_C2_COEF = 0x191 CSC_C3_COEF = 0x341 CSC_C4_COEF = 0x811

A4. Output data package format (RGB order)

Format	Package
RGB888	
RGB555	
RGB565	
NOTE: All R/G/B data are little-endian type; all the empty bits in the above figure are filled with 0.	

A5. Source Data storing format in external memory (separated YUV Frame)

Example: YUV420 118x80 frame



NOTES:

1 Every line's start address should be word aligned.

2 All pixel data should be stored as little-endian format.

3 Destination data (RGB) storing format in external memory is similar with above figure, but RGB555 and RGB565 frame's every line start address can be half-word aligned (RGB888 frame still need word aligned).

16 Internal CODEC Interface

16.1 Overview

This chapter describes the embedded audio CODEC in the processor and related software interface.

This embedded CODEC is an I2S audio CODEC. AIC module is an interface to this CODEC in audio data replaying and recording. Several memory mapped registers are used to access this embedded CODEC, and write/read these registers could access the CODEC's internal control and configure registers that is using 12Mhz clock.

16.1.1 Features

The following are internal CODEC features:

- 24 bits ADC and DAC
- Headphone load up to 16 Ohm
- Sample frequency supported: 8k, 9.6k, 12k, 11.025k, 12k, 16k, 22.05k, 24k, 32k, 44.1k, 48k, 96k
- Two MIC input, 85db SNR
- Stereo line input
- Separate power-down modes for ADC and DAC path with several shutdown modes
- Reduction of audible glitches systems: Pop Reduction system, Soft Mute mode
- Support capacitor-less headphone connection

OPT = pins or functions may not available in some specify chip
TBD = parameter or document section to be defined later on
TBC = parameter or document section subject to change
TO BE COMPLETED = section to be filled or subject to change

16.1.2 Signal Descriptions

CODEC has max 13 analog signal IO pins and 4 power pins on chip. They are listed and described in Table 16-1.

Table 16-1 CODEC signal IO pin description

Pin Names	IO	Note	Pin Description	Power
AOHPL	AO		Left headphone out.	AVDHP
AOHPR	AO		Right headphone out.	AVDHP
AOHPM	AO	OPT	Headphone common mode output.	AVDHP
AOHPMS	AI	OPT	Headphone common mode sense input. (connect to AOHPM)	AVDHP
MICP1	AI		Microphone mono differential analog input 1 (MIC1), positive pin.	AVDCDC
MICN1	AI	OPT	Microphone mono differential analog input 1 (MIC1), negative pin.	AVDCDC
MICP2	AI	OPT	Microphone mono differential analog input 2 (MIC2), positive pin.	AVDCDC
MICN2	AI	OPT	Microphone mono differential analog input 2 (MIC2), negative pin.	AVDCDC
MICBIAS	AO	OPT	Microphone bias.	AVDCDC
AIL	AI		Left line input. Also named LLINEIN in some place.	AVDCDC
AIR	AI		Right line input. Also named RLINEIN in some place.	AVDCDC
VCOM	AO		Voltage Reference Output. An electrolytic capacitor more than 10μF in parallel with a 0.1μF ceramic capacitor attached from this pin to AVSCDC eliminates the effects of high frequency noise.	AVDCDC
AVDHP	P		Headphone amplifier power, 3.3V.	-
AVSHP	P		Headphone amplifier ground.	-
AVDCDC	P		CODEC analog power, 3.3V, inter signal VREFP.	-
AVSCDC	P		CODEC analog ground, inter signal VREFN.	-
HPSENSE	AI	OPT	Headphone jack sense.	AVDHP

NOTES:

1 AVDHP = 3.3v (typ).
2 AVDCDC= 3.3v (typ).
3 Inter signal VREFP is connected to AVDCDC, inter signal VREFN is connected to AVSCDC.
4 Please refer to data sheet of the chip for details.
5 In target chip package, NOT all pins are available. Please refer to the chip specification.

16.1.3 Block Diagram

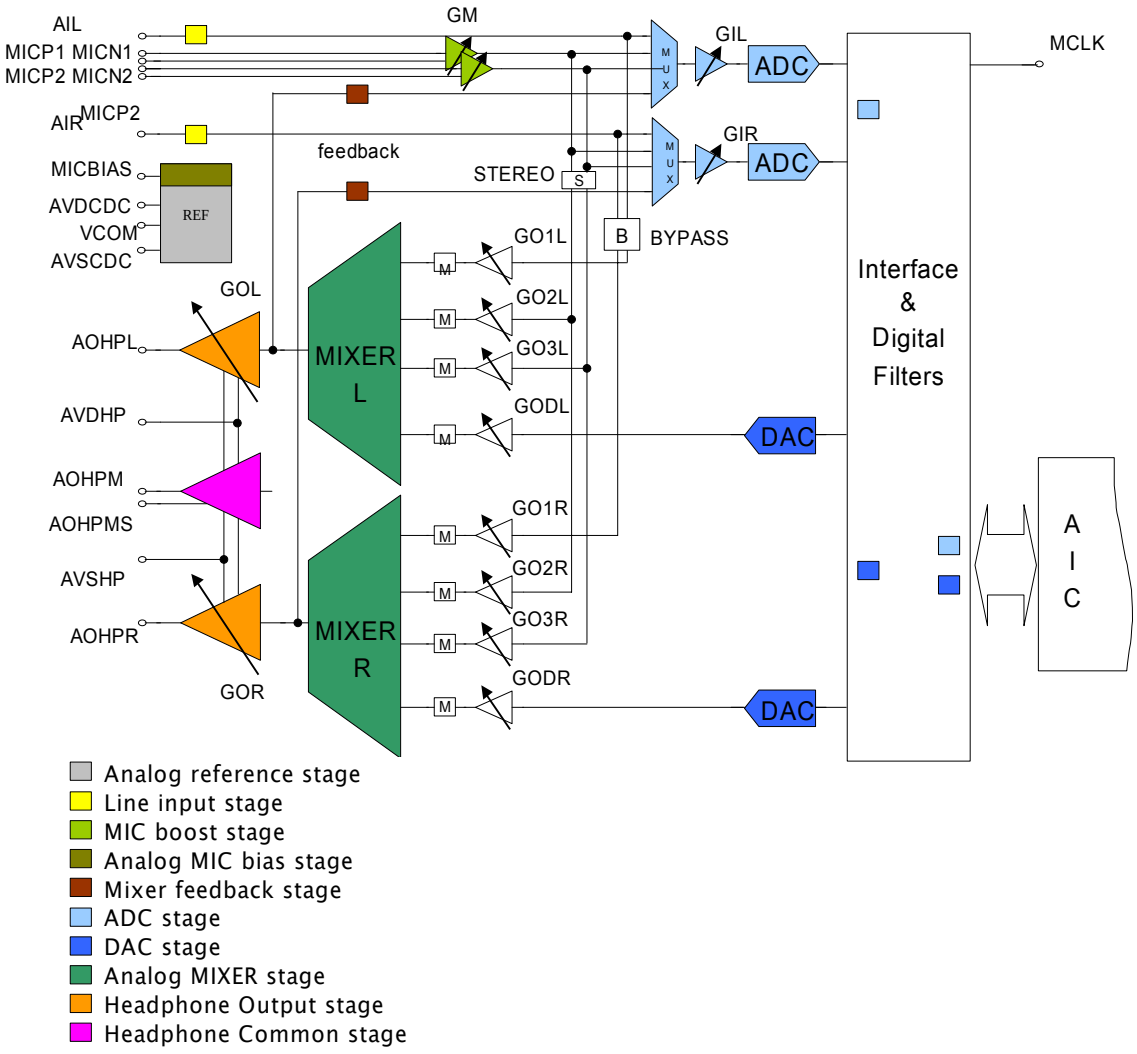


Figure 16-1 CODEC block diagram

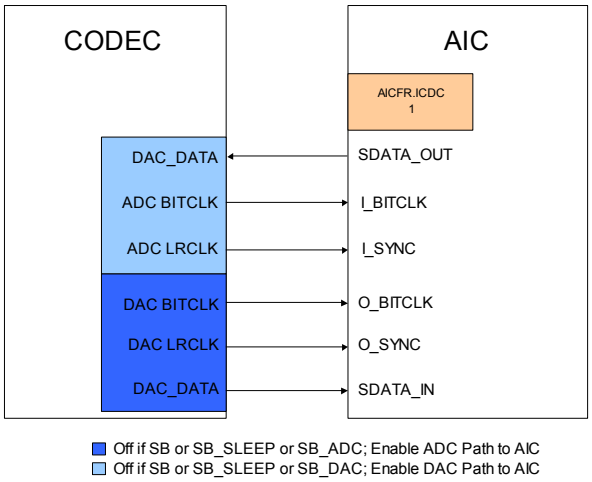


Figure 16-2 Internal CODEC works with AIC

16.2 Mapped Register Descriptions

The internal CODEC software interface includes 2 registers. They are mapped in IO memory address space of AIC module so that program can access them to control the operations of the CODEC.

Table 16-2 Internal CODEC Mapped Registers Description (AIC Registers)

Name	Description	RW	Reset value	Address	Size
RGADW	Address, data in and write command for accessing to internal registers of internal embedded CODEC	RW	0x00000000	0x100200A4	32
RGDATA	The read out data and interrupt request status of Internal registers data in the internal embedded CODEC.	R	0x00000000	0x100200A8	32

NOTES:

1 All these registers are AIC Registers, because they are mapped in AIC IO memory address.

2 RGADW contains data, address and write command to the internal registers of the internal CODEC.

3 RGDATA returns the internal register value of the internal CODEC and interrupt request status.

16.2.1 CODEC internal register access control (RGADW)

RGADW contains address, data and write command to the internal registers of the internal embedded CODEC.

RGADW 0x100200A4

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

Bits	Name	Description	RW
31:17	Reserved	Writes to these bits have no effect and always read as 0.	R
16	RGWR	Write 1 to this bit issues writing to CODEC's internal register process. This bit keeps value 1 until the current writing process is finished. A register read or a new register writing process cannot be issued before the previous writing process finished. In another word, it should not write to RGADW before RGADW.RGWR becomes 0. A writing process takes max of 0.17us plus 1 PCLK cycle. Write 0 to this bit is ignored.	RW

15	Reserved	Writes to these bits have no effect and always read as 0.	R
14:8	RGADDR	When it issues a writing to CODEC's internal register command, i.e. RGWR=1, this field specifies the register's address. In addition, this field also decides the address of the register's data out at any time.	RW
7:0	RGDIN	When it issues a writing to CODEC's internal register command, i.e. RGWR=1, this field contains the data to be written to the register.	RW

NOTES:

1 It is strong suggesting verifying the data (using read RGDATA below) after writing it to internal register of CODEC. When RGDATA returns the right data which writing to the address, the writing process is finish.

2 Please notice that AIC needs SYS_CLK (refers to AIC spec), when write new value to or read from CODEC internal registers.

16.2.2 CODEC internal register data output (RGDATA)

RGDATA returns the internal register value of the internal embedded CODEC and interrupt request status.

RGDATA **0x100200A8**

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

Bits	Name	Description	RW
31:9	Reserved	Writes to these bits have no effect and always read as 0.	R
8	IRQ	This field returns the internal embedded CODEC's interrupt request. **IRQ** / **Description** 0 / **No CODEC's interrupt request found** 1 / **CODEC's interrupt request is pending**	R
7:0	RGDOUT	This field returns the value of the internal register in internal embedded CODEC. As the RGADW.RGADDR field specifies the register's address.	R

Please notice that AIC needs SYS_CLK (refers to AIC spec), when write new value to or read from CODEC internal registers.

16.3 Operation

The internal embedded CODEC is controlled its internal registers. These registers can be accessed by through memory-mapped registers, RGADW and RGDATA, just like L3 bus or I2C bus for an external CODEC. AIC's BITCLK and SYNC are from/to the CODEC and is controlled by CKCFG.SELAD register. The audio data transferring, i.e. audio replaying and recording, is down by AIC. AIC still takes the role of I2S controller. We will refer to many AIC operations and registers in the following audio operation descriptions, please reference to AIC Spec for the details.
This is a guide for software.

16.3.1 Access to internal registers of the embedded CODEC

The embedded CODEC is controlled through its internal registers. RGADW and RGDATA are used to write to and read from these registers. Here are some examples.

Example 1. Write to a CODEC internal register.

Step 1: RGADW.RGWR == 0.
Step 2: If not, go to step 1.
Step 3: Write to RGADW and make it.
RGADW.RGDIN = <data to be written to the register>.
RGADW.RGADDR = <the register's address >.
Step 4: Write to RGADW to commit the writing operation.
RGADW.RGWR = 1.

Example 2. Read from a CODEC internal register.

Step 1: RGADW.RGWR == 0.
Step 2: If not, go to step 1.
Step 3: write to RGADW and make it.
RGADW.RGWR = 0.
RGADW.RGDIN = <don't care>.
RGADW.RGADDR = <the register's address>.
Step 4: read RGDATA.DOUT, which returns the register's content.

16.3.2 CODEC controlling and typical operations

This section is some typical operations. We are assumed the power supply of CODEC is on, and CODEC is in STANDBY mode, CRR is configured for audio Ramping system, clear PMR2.SB_MC to 0 in capacitor-less connection mode (refers to capacitor-less headphone connection).

Before using any of these operations, make sure AIC is configured properly as list below:

1 Make AIC to use internal CODEC mode:
AICFR.ICDC = 1; Use internal CODEC.
AICFR.AUSEL = 1; Use I2S mode.
AICFR.BCKD = 0; CODEC input BIT_CLK to AIC.

AICFR.SYNCD = 0; CODEC input SYNC to AIC.
I2SCR.AMSL = 1; Use I2S operation mode.
I2SCR.ESCLK = 1; Open SYS_CLK to internal CODEC.(if using PLL Clock)

2 Make sure AICCR.FLUSH = 0; AICFR.RST = 0; AICCR.ENLBF = 0.
3 Clear AICSR.ROR, AICSR.TUR, AICSR.RFS, AICSR.TFS = 0 to 0.
4 Set proper value to AICCR.M2S; AICCR.ENDSW; AICCR.ASVTSU.
5 Set AICFR.ENB to 1; Open AIC.

When using DMA mode, configure AICFR.RFTH, AICCR.RDMS or AICFR.TFTH, AICCR.TDMS.
Configure TX-FIFO and interrupt means setting proper value to AICFR.TFTH, clear AICCR.ETFS to 0, and clear AICCR.ETUR to 0.
Configure RX-FIFO and interrupt means setting proper value to AICFR.RFTH, clear AICCR.ERFS to 0 and clear AICCR.EROR to 0.
When configure interrupt, software must handle all the interrupt. So all interrupt is recommended disabled as shown above.
CODEC shares the interrupt with AIC module.

The register or register bit of CODEC will use the same form as the Mapped registers, but software should use the method in this section to access this registers.

All the REF parts of the working part diagrams are working. More details are listed in the CODEC guide.

16.3.3 Power saving

There are many power modes in CODEC. In every working mode, it should close stages (parts) of CODEC for saving power.
The power diagram is shown in Figure 2-7 CODEC Power Diagram; please refer to CODEC Operating modes.

16.3.4 Pop noise and the reduction of it

Please refre to Anti-pop operation sequences.

16.3.4.1 Reference open step

1 Init play.
Step 0: Open DMA and two AIC modules Clocks in CPM.CLKGR.
Step 1: Configure AIC as slave and using inter CODEC mode.
AICFR.ICDC = 1; Use internal CODEC.
AICFR.AUSEL = 1; Use I2S mode.
AICFR.BCKD = 0; CODEC input BIT_CLK to AIC.

AICFR.SYNCD = 0; CODEC input SYNC to AIC.

I2SCR.AMSL = 1; Use I2S operation mode.

I2SCR.ESCLK = 1; Open SYS_CLK to internal CODEC.

Step 2: Configure DMA as slave mode using internal CODEC.

2 Open.

Step 0: Enable DMA Channel Clock.

Step 1: Configure AIC sample size and sample rate. Configure AIC Output FIFO Threshold.

Step 2: Configure DMA.

Step 3: Configure CODEC.

3 Write.

Step 0: Enable DMA Channel Clock.

Step 1: Configure AIC.

Step 2: Configure DMA.

Step 3: Configure CODEC.

4 Read.

Step 0: Enable DMA Channel Clock.

Step 1: Configure AIC.

Step 2: Configure DMA.

Step 3: Configure CODEC.

5 Close.

6 End.

16.4 Timing parameters

1 Tsbyu: Reference wake-up time after complete power down.
When Cext = 10uF/100nF +/-20%, Typical value of Tsbyu is 250ms, Max is 500ms.

2 Tshd_adc: ADC wake-up time after SLEEP mode.
When Cext = 10uF/100nF +/-20%, Typical value is 200ms(TBC).

3 Tshd_dac: DAC wake-up time after SLEEP mode.
When Cext = 10uF/100nF +/-20% Typical value is to be defined later on.
The Cext is the two decoupling capacitors between the VREF and VREFN (AVSCDC). Refer to Avoid quiet ground common currents.

16.5 AC & DC parameters

Votages:

AVSHP and AVSCDC are connected to analog ground.

AVDHP = 3.3v (typ).

AVDCDCP= 3.3v (typ).

Currents:

Mode	Currents
1 Complete down (Static)	$I_{AVDCDC} + I_{AVDHP} < 5\mu A$
2 SLEEP mode (Static)	TBD
3 SLEEP mode with MCLK(Static)	TBD
4 Playback to AOHPR/AOHPL(Silence)	$2\ mA < I_{AVDCDC} + I_{AVDHP} < 8\ mA$
5 Record from AIL/AIR(Silence)	$1.5\ mA < I_{AVDCDC} + I_{AVDHP} < 6\ mA$
6 Playback with Record (4 + 5 Silence)	$3\ mA < I_{AVDCDC} + I_{AVDHP} < 10\ mA$
7 Playback to AOHPR/AOHPL(Digital Full Scale)	TBD
8 Record from AIL/AIR(2.8Vpp)	TBD
9 Playback with Record (7 + 8 Full Scale)	TBD

Current value is at AVDCDC = AVDHP = 3.3 V.

Chip pin Name	MAX Current across I/O @ AVDCDC = AVDHP = 3.3 V
AVDCDCP	< 20 mA in normal working mode
AVSCDCP	< 20 mA in normal working mode
AVDHP	< 160 mA in normal working mode
	< 1400 mA in case of short circuit
AVSHP	< 160 mA in normal working mode
	< 1400 mA in case of short circuit
VCOM	< 2 mA in normal working mode
MICP	< 2 mA in normal working mode
MICBIAS	< 5 mA in normal working mode
AIL, AIR	< 2 mA in normal working mode
AOHPL	< 80 mA in normal working mode
	< 1200 mA in case of short circuit
AOHPR	< 80 mA in normal working mode
	< 1200 mA in case of short circuit
AOHPM	< 160 mA in normal working mode
	< 1400 mA in case of short circuit
AOHPMS	< 1 mA in normal working mode

Please refer to Chip Datasheet for more details.

16.6 CODEC Configuration guide

16.6.1 CODEC internal Registers

Register Name	Function	Address	Reset value	Software Write
AICR	Audio Interface Control	00000 / 00 / 00	0C	0F
CR1	Control Register 1	00001 / 01/ 01	AA	
CR2	Control Register 2	00010 / 02 / 02	78	
CCR1	Control Clock Register 1	00011 / 03 / 03	00	00
CCR2	Control Clock Register 2	00100 / 04 / 04	00	
PMR1	Power Mode Register 1	00101 / 05 / 05	FF	
PMR2	Power Mode Register 2	00110 / 06 / 06	03	
CRR	Control Ramp Register	00111 / 07 / 07	51	51
ICR	Interrupt Control Register	01000 / 08 / 08	3F	A0 [1]
IFR	Interrupt Flag Register	01001 / 09 / 09	00	(IFR)
CGR1	Control Gain Register 1	01010 / 10 / 0A	00	
CGR2	Control Gain Register 2	01011 / 11 / 0B	04	
CGR3	Control Gain Register 3	01100 / 12 / 0C	04	
CGR4	Control Gain Register 4	01101 / 13 / 0D	04	
CGR5	Control Gain Register 5	01110 / 14 / 0E	04	
CGR6	Control Gain Register 6	01111 / 15 / 0F	04	
CGR7	Control Gain Register 7	10000 / 16 / 10	04	
CGR8	Control Gain Register 8	10001 / 17 / 11	0A	
CGR9	Control Gain Register 9	10010 / 18 / 12	0A	
CGR10	Control Gain Register 10	10011 / 19 / 13	00	
CR3	Control Register 3	10110 / 22 / 16	C0	C0 [2]
AGC1	Automatic Gain Control 1	10111 / 23 / 17	34	
AGC2	Automatic Gain Control 2	11000 / 24 / 18	07	
AGC3	Automatic Gain Control 3	11001 / 25 / 19	44	
AGC4	Automatic Gain Control 4	11010 / 26 / 1A	1F	
AGC5	Automatic Gain Control 5	11011 / 27 / 1B	00	

NOTES:

1 After write AFR by reading AFR value for clear AFR, Must set ICR to A0.

2 This register should keep the reset value 11000000(C0) in REPLAY mode.

3 Before configuration the CODEC make sure the CONFIG* field configured properly first.

16.6.2 CODEC internal registers

16.6.2.1 AICR: Audio Interface Control Register

Register Name: AICR Register Address: 0x00

bit7-RW-0	bit6-RW-0	bit5-RW-0	bit4-RW-0	bit3-RW-1	bit2-RW-1	bit1-RW-0	bit0-RW-0
Reserved				CONFIG1			

Bits	Field	Description
7:4	Reserved	These bits are not used, when read is 0000.
3:0	CONFIG1	These bits must be set to 1111.

NOTES:

1 This register should keep the value 00001111 by software for proper configuration status.

2 Must set a value to every CONFIGn(n=1 to 8) field before use this CODEC, here is CONFIG1.

16.6.2.2 CR1: Control Register 1

Register Name: CR1 Register Address: 0x01

bit7-RW-1	bit6-RW-0	Bit5-RW-1	bit4-RW-0	bit3-RW-1	bit2-RW-0	bit1-RW-1	bit0-RW-0
SB_MICBIAS	MONO	DAC_MUTE	HP_DIS	DACSEL	BYPASS	Reserved	

Bits	Field	Description
7	SB_MICBIAS	Microphone biasing buffer power-down. 0: active 1: power-down
6	MONO	Stereo-to-mono conversion for DAC path. 0: stereo 1: mono
5	DAC_MUTE	DAC soft mute mode. 0: mute inactive, digital input signal transmitted to the DAC 1: puts the DAC in soft mute mode
4	HP_DIS	HeadPhone output signal disabled. 0: Signal applied to headphone outputs 1: no signal on headphone outputs, acts as a mute signal
3	DACSEL	Mixer input selection. 0: DAC output ignored in input of the mixer 1: DAC output selected as an input of the mixer
2	BYPASS	Mixer input selection (line). 0: Bypass path ignored in input of the mixer

		1: Bypass path selected as an input of the mixer
1:0	Reserved	These bits are not used, when read is 00.

16.6.2.3 CR2: Control Register 2

Register Name: CR2 Register Address: 0x02

bit7-RW-0	bit6-RW-1	Bit5-RW-1	Bit4-RW-1	bit3-RW-1	bit2-RW-0	bit1-RW-0	bit0-RW-0
DAC_DEEMP	DAC_ADWL		ADC_ADWL		ADC_HPF	Reserved	

Bits	Field	Description
7	DAC_DEEMP	DAC De-emphasize filter enable. 0: inactive 1: enables the de-emphasis filter
6:5	DAC_ADWL	Audio Data Word Length of DAC path. 00: 16-bit word length data 01: 18-bit word length data 10: 20-bit word length data 11: 24-bit word length data
4:3	ADC_ADWL	Audio Data Word Length of ADC path. 00: 16-bit word length data 01: 18-bit word length data 10: 20-bit word length data 11: 24-bit word length data
2	ADC_HPF	ADC High Pass Filter enable. 0: inactive 1: enables the ADC High Pass Filter
1:0	Reserved	These bits are not used, when read is 00.

16.6.2.4 CR3: Control Register 3

Register Name: CR3 Register Address: 0x16

bit7-RW-1	bit6-RW-1	bit5-RW-0	Bit4-RW-0	bit3-RW-0	bit2-RW-0	bit1-RW-0	bit0-RW-0
SB_MIC1	SB_MIC2	SIDETONE1	SIDETONE2	MICDIFF	MICSTEREO	INSEL	

Bits	Field	Description
7	SB_MIC1	Analog Microphone 1 (MIC1) Input conditioning circuitry power-down mode. 0: active 1: power-down
6	SB_MIC2	Analog Microphone 2 (MIC2) Input conditioning circuitry power-down

<table>
<tr><td></td><td></td><td>mode.
0: active
1: power-down</td></tr>
<tr><td>5</td><td>SIDETONE1</td><td>Select Microphone 1 (MIC1) as an input of Mixer.
0: Sidetone1 path ignored in input of the mixer
1: Sidetone1 path selected as an input of the mixer
NOTE: It must configure in RECORD with Direct Playback mode. This signal is also affected by MICSTEREO. Refer to its description.</td></tr>
<tr><td>4</td><td>SIDETONE2</td><td>Select Microphone 2 (MIC2) as an input of Mixer.
0: Sidetone1 path ignored in input of the mixer
1: Sidetone1 path selected as an input of the mixer
NOTE: It must configure in RECORD with Direct Playback mode. This signal is also affected by MICSTEREO. Refer to its description.</td></tr>
<tr><td>3</td><td>MICDIFF</td><td>Microphone input mode selection.
0: Microphone single-ended inputs
1: Microphone differential inputs</td></tr>
<tr><td>2</td><td>MICSTEREO</td><td>Microphone input mode selection.
0: Microphone mono inputs
1: Microphone stereo inputs
<table>
<tr><th>SIDETONE1</th><th>SIDETONE2</th><th>MICSTEREO</th><th>Left channel input of Mixer</th><th>Right channel input of Mixer</th></tr>
<tr><td>0</td><td>0</td><td>X</td><td>None</td><td>None</td></tr>
<tr><td rowspan="2">1</td><td rowspan="2">0</td><td>0</td><td>MIC1</td><td>MIC1</td></tr>
<tr><td>1</td><td>MIC2</td><td>MIC1</td></tr>
<tr><td rowspan="2">0</td><td rowspan="2">1</td><td>0</td><td>MIC2</td><td>MIC2</td></tr>
<tr><td>1</td><td>MIC1</td><td>MIC2</td></tr>
</table></td></tr>
<tr><td>1:0</td><td>INSEL</td><td>Selection of the input signal converted by the ADC.
00: Microphone 1 (MIC1) input to left and right channels
01: Microphone 2 (MIC2) input to left and right channels
10: Line input
11: Mixer output</td></tr>
</table>

NOTES:

1 This register should keep the reset value 11000000 in REPLAY mode.

2 Please refer to section CODEC Operating modes.

16.6.2.5 CCR1: Control Clock Register 1

Register Name: CCR1 Register Address: 0x03

bit7-RW-0	bit6-RW-0	Bit5-RW-0	bit4-RW-0	bit3-RW-0	bit2-RW-0	bit1-RW-0	bit0-RW-0
Reserved				CONFIG4			

Bits	Field	Description
7:4	Reserved	This bits are not used.
3:0	CONFIG4	These bits must be clear to 0000.

NOTES:

1 This register should keep the reset value 00000000.

2 The CONFIG4 value 0000 means CODEC use the inter 12Mhz clock.

16.6.2.6 CCR2: Control Clock Register 2

Register Name: CCR2 Register Address: 0x04

bit7-RW-0	bit6-RW-0	Bit5-RW-0	bit4-RW-0	bit3-RW-0	bit2-RW-0	bit1-RW-0	bit0-RW-0
DFREQ				AFREQ			

Bits	Field	Description
7:4	DFREQ	Selection of the DAC sampling rate (Fs). NOTE: The sampling frequency value is given in the FREQ[3:0] table.
3:0	AFREQ	Selection of the ADC sampling rate (Fs). NOTE: The sampling frequency value is given in the FREQ[3:0] table.

NOTE: Please refer to section Sample frequency: FREQ.

16.6.2.7 PMR1: Power Mode Register 1

Register Name: PMR1 Register Address: 0x05

bit7-RW-1	bit6-RW-1	Bit5-RW-1	bit4-RW-1	bit3-RW-1	bit2-RW-1	bit1-RW-1	bit0-RW-1
SB_DAC	SB_OUT	SB_MIX	SB_ADC	SB_LIN	Reserved		SB_IND

Bits	Field	Description
7	SB_DAC	DAC power-down mode. 0: active 1: power-down
6	SB_OUT	Output Stage power-down mode. 0: active

		1: power-down
5	SB_MIX	Mixer and line output stage power-down. 0: active 1: power-down
4	SB_ADC	ADC power-down mode. 0: active 1: power-down
3	SB_LIN	Analog line Input (Bypass) conditioning circuitry power-down mode. 0: active 1: power-down
2:1	Reserved	These bits are not used, when read is 11.
0	SB_IND	Mixer to ADC circuitry power-down mode. 0: active 1: power-down

NOTE: Please refer to section CODEC Operating modes.

16.6.2.8 PMR2: Power Mode Register 2

Register Name: PMR2 Register Address: 0x06

bit7-RW-0	bit6-RW-0	bit5-RW-0	bit4-RW-0	bit3-RW-0	bit2-RW-0	bit1-RW-1	bit0-RW-1
LRGI	RLGI	LRGOD	RLGOD	GIM	SB_MC	SB	SB_SLEEP

Bits	Field	Description
7:6	LRGI, RLGI	PGATM input gain coupling. 00: Left and right channels gains are independent, respectively given by GIL and GIR 10: Left and right channels gain is given by GIR 01: Left and right channels gain is given by GIL 11: Left and right channels gain is given by GIR
5:4	LRGOD, RLGOD	DAC mixing gain coupling. 00: Left and right channels gains are independent, respectively given by GODL and GODR 10: Left and right channels gain is given by GODR 01: Left and right channels gain is given by GODL 11: Left and right channels gain is given by GODR
3	GIM	Microphone (MIC1) amplifier gain control. 0: 0 dB gain 1: 20 dB gain
2	SB_MC	Output Stage common mode buffer power-down. 0: active (capacitor less headphone output configuration)

		1: power-down (line output configuration)
1	SB	Complete power-down mode. 0: normal mode (active) 1: complete power-down
0	SB_SLEEP	SLEEP mode. 0: normal mode (active) 1: SLEEP mode

NOTES:

1 Please refer to section CODEC Operating modes, Programmable boost gain: GIM.

2 Please refer to section Capacitor-coupled headphone connection, Capacitor-less headphone connection for SB_MC setting.

16.6.2.9 CRR: Control Ramp Register

Register Name: CRR Register Address: 0x07

bit7-RW-0	bit6-RW-1	bit5-RW-0	bit4-RW-1	bit3-RW-0	bit2-RW-0	bit1-RW-0	bit0-RW-1
Reserved	RATIO		KFAST			TRESH	

Bits	Field	Description
7	Reserved	This bit are not used, when read is 0.
6:5	RATIO	Ratio between fast and slow steps. 00: Ratio is 1 01: Ratio is 2 10: Ratio is 4 (default) 11: Ratio is 8
4:2	KFAST	Factor for step time in fast slope part. 000: KFast is 1 001: KFast is 2 010: KFast is 4 011: KFast is 8 100: KFast is 16 (default) 101: Kfast is 32
1:0	TRESH	Threshold between fast and slow slope parts. 00: Threshold is 0 01: Threshold is 32 (default) 10: Threshold is 64 11: Threshold is 128

NOTES:

1 This register should keep the reset value 01010001(51) for reduce pop-noise.

2 Please refer to section Ramping system guide for details.

16.6.2.10 ICR: Interrupt Control Register

Register Name: ICR Register Address: 0x08

Bit7-RW-0	bit6-RW-0	bit5-RW-1	bit4-RW-1	bit3-RW-1	bit2-RW-1	bit1-RW-1	bit0-RW-1
INT_FORM		JACK_MASK	CCMC_MASK	RUD_MASK	RDD_MASK	GUD_MASK	GDD_MASK

Bits	Field	Description
7:6	INT_FORM	Waveform and polarity of the IRQ signal. 00: The generated IRQ is a high level 01: The generated IRQ is a low level 10: The generated IRQ is a high level pulse with an 8 SYS_CLK cycles duration. 11: The generated IRQ is a low level pulse with an 8 SYS_CLK cycles duration.
5	JACK_MASK	Mask for the JACK_EVENT flag. 0: interrupt enabled 1: interrupt masked (no IRQ generation)
4	CCMC_MASK	Mask for the CCMC flag. 0: interrupt enabled 1: interrupt masked (no IRQ generation)
3	RUD_MASK	Mask for the RAMP_UP_DONE flag. 0: interrupt enabled 1: interrupt masked (no IRQ generation)
2	RDD_MASK	Mask for the RAMP_DOWN_DONE flag. 0: interrupt enabled 1: interrupt masked (no IRQ generation)
1	GUD_MASK	Mask for the GAIN_UP_DONE flag. 0: interrupt enabled 1: interrupt masked (no IRQ generation)
0	GDD_MASK	Mask for the GAIN_DOWN_DONE flag. 0: interrupt enabled 1: interrupt masked (no IRQ generation)

NOTES:

1 When an interrupt is masked, the event do not generates any change on the IRQ signal, but the corresponding flag value is set to '1' in the IFR register.

2 When the IRQ signal is active on level, the IRQ signal is set to the inactive level while no IRQ occurs, which is unmasked.

3 When the IRQ signal is a pulse, the IRQ signal is set to the inactive state until a new non-masked event occurs in IFR[5:0] or until a masked event is unmasked.

4 When using CODEC Interrupt, it must set AIC.I2SCR.ESCLK to 1 and AIC.AICFR.ENB to 1.

5 CODEC Interrupt is sharing with AIC Interrupt.

6 That Writing IFR by reading current IFR value will clear all interrupt flag values.

16.6.2.11 IFR: Interrupt Flag Register

Register Name: IFR Register Address: 0x09

bit7-RW-0	Bit6-R-0	bit5-RW-0	bit4-R-0	bit3-RW-0	bit2-RW-0	bit1-RW-0	bit0-RW-0
Reserved	JACK	JACK_EVENT	CCMC	RAMP_UP _ DONE	RAMP_DOW N_ DONE	GAIN_UP _DONE	GAIN_DOWN _DONE

Bits	Field	Description
7	Reserved	These bits are not used, when read is 000.
6	JACK	Output Jack plug detection status. 0: no jack 1: output jack present
5	JACK_EVENT	Event on output Jack plug detection status. 0: no event 1: event detected (due to JACK flag change to '0' or '1') Write 1 to Reset of the flag.
4	CCMC	Output short circuit detection status – Reserved for future use Read. 0: inactive 1: indicates that a short circuit has been detected by the output stage The conditions that reset the flag are only in the capacitor-less headphone-ended outputs mode. AOHPL and AOHPR are driven through Programmable Gain Amplifiers/Attenuators. Write 1 to Update of the flag.
3	RAMP_UP_DONE	End of output stage ramp up flag. 0: no event 1: the ramp-up sequence is completed; Output Stage is active Write 1 to Reset of the flag.
2	RAMP_DOWN_DONE	End of output stage ramp down flag. 0: no event 1: the ramp-down sequence is completed; Output Stage in stand-by mode Write 1 to Reset of the flag.
1	GAIN_UP_DONE	End of mute gain up sequence flag. 0: no event 1: the mute sequence is completed; the DAC input signal is transmitted to the DAC path Write 1 to Reset of the flag.
0	GAIN_DOWN_DONE	End of mute gain down sequence flag. 0: no event

		1: the mute sequence is completed; a 0 DC signal is transmitted to the DAC path Write 1 to Reset of the flag.

NOTES:

1 The flags RAMP_UP_DONE, RAMP_DOWN_DONE, GAIN_UP_DONE and GAIN_DOWN_DONE can be reset after 4 cycles of SYS_CLK.

2 Interpretation of any unspecified point is absolutely up to the designer of analog part, so it is need to pay a attention to using this flags in section Operation sequences.

16.6.2.12 CGR1: Control Gain Register 1

Register Name: CGR1 Register Address: 0x0A

bit7-RW-0	bit6-RW-0	Bit5-RW-0	bit4-RW-0	bit3-RW-0	bit2-RW-0	Bit1-RW-0	bit0-RW-0
GODL				GODR			

Bits	Field	Description
7:4	GODL	DAC mixing left channel gain programming value.
3:0	GODR	DAC mixing right channel gain programming value.

NOTE: Please refer to section Programmable attenuation GOD for more details.

16.6.2.13 CGR2: Control Gain Register 2

Register Name: CGR2 Register Address: 0x0B

bit7-RW-0	bit6-RW-0	Bit5-RW-0	bit4-RW-0	bit3-RW-0	bit2-RW-1	Bit1-RW-0	bit0-RW-0
LRGO1	RLGO1	Reserved	GO1R				

Bits	Field	Description
7:6	RLGO1, LRGO1	Line 1 mixing gain coupling. 00: Left and right channels gains are independent, respectively given by GO1L and GO1R 10: Left and right channels gain is given by GO1R 01: Left and right channels gain is given by GO1L 11: Left and right channels gain is given by GO1R
5	Reserved	This bit are not used, when read is 0.
4:0	GO1R	Line 1 mixing right channel gain programming value.

NOTE: Please refer to section Programmable attenuation GOi for more details.

16.6.2.14 CGR3: Control Gain Register 3

Register Name: CGR3 Register Address: 0x0C

bit7-RW-0	bit6-RW-0	Bit5-RW-0	bit4-RW-0	bit3-RW-0	bit2-RW-1	bit1-RW-0	bit0-RW-0
Reserved			GO1L				

Bits	Field	Description
7:5	Reserved	These bits are not used, when read is 000.
4:0	GO1L	Line 1 mixing left channel gain programming value.

NOTE: Please refer to section Programmable attenuation GOi for more details.

16.6.2.15 CGR4: Control Gain Register 4

Register Name: CGR4 Register Address: 0x0D

bit7-RW-0	bit6-RW-0	Bit5-RW-0	Bit4-RW-0	bit3-RW-0	bit2-RW-1	bit1-RW-0	bit0-RW-0
RLGO2	LRGO2	Reserved	GO2R				

Bits	Field	Description
7:6	RLGO2, LRGO2	Microphone 1 mixing gain coupling. 00: Left and right channels gains are independent, respectively given by GO2L and GO2R 10: Left and right channels gain is given by GO2L 01: Left and right channels gain is given by GO2R 11: Left and right channels gain is given by GO2L
5	Reserved	This bit are not used, when read is 0.
4:0	GO2R	Microphone 1 mixing right channel gain programming value.

NOTE: Please refer to section Programmable attenuation GOi for more details.

16.6.2.16 CGR5: Control Gain Register 5

Register Name: CGR5 Register Address: 0x0E

bit7-RW-0	bit6-RW-0	bit5-RW-0	Bit4-RW-0	bit3-RW-0	bit2-RW-1	bit1-RW-0	bit0-RW-0
Reserved			GO2L				

Bits	Field	Description
7:5	Reserved	These bits are not used, when read is 000.
4:0	GO2L	Microphone 1 mixing left channel gain programming value.

NOTE: Please refer to section Programmable attenuation GOi for more details.

16.6.2.17 CGR6: Control Gain Register 6

Register Name: CGR6 Register Address: 0x0F

bit7-RW-0	bit6-RW-0	bit5-RW-0	bit4-RW-0	bit3-RW-0	bit2-RW-1	Bit1-RW-0	bit0-RW-0
RLGO3	LRGO3	Reserved	GO3R				

Bits	Field	Description
7:6	RLGO3, LRGO3	Microphone 2 mixing gain coupling. 00: Left and right channels gains are independent, respectively given by GO3L and GO3R 10: Left and right channels gain is given by GO3R 01: Left and right channels gain is given by GO3L 11: Left and right channels gain is given by GO3R
5	Reserved	This bit are not used, when read is 0.
4:0	GO3R	Microphone 2 mixing right channel gain programming value.

NOTE: Please refer to section Programmable attenuation GOi for more details.

16.6.2.18 CGR7: Control Gain Register 7

Register Name: CGR7 Register Address: 0x10

bit7-RW-0	bit6-RW-0	bit5-RW-0	bit4-RW-0	bit3-RW-0	bit2-RW-1	bit1-RW-0	bit0-RW-0
Reserved			GO3L				

Bits	Field	Description
7:5	Reserved	These bits are not used, when read is 000.
4:0	GO3L	Microphone 2 mixing left channel gain programming value.

NOTE: Please refer to section Programmable attenuation GOi for more details.

16.6.2.19 CGR8: Control Gain Register 8

Register Name: CGR8 Register Address: 0x11

bit7-RW-0	bit6-RW-0	Bit5-RW-0	bit4-RW-0	bit3-RW-1	bit2-RW-0	Bit1-RW-1	bit0-RW-0
RLGO	LRGO	Reserved	GOR				

Bits	Field	Description
7:6	RLGO, LRGO	Output stages gain coupling. 00: Left and right channels gains are independent, respectively given by GOL and GOR 10: Left and right channels gain is given by GOR 01: Left and right channels gain is given by GOL 11: Left and right channels gain is given by GOR
5	Reserved	This bit are not used, when read is 0.
4:0	GOR	Output stage right channel gain programming value.

NOTE: Please refer to section Programmable output amplifer: PGAT for more details.

16.6.2.20 CGR9: Control Gain Register 9

Register Name: CGR9 Register Address: 0x12

bit7-RW-0	bit6-RW-0	bit5-RW-0	bit4-RW-0	bit3-RW-1	bit2-RW-0	bit1-RW-1	bit0-RW-0
Reserved				GOL			

Bits	Field	Description
7:5	Reserved	These bits are not used, when read is 000.
4:0	GOL	Output stage left channel gain programming value.

NOTE: Please refer to section Programmable output amplifer: PGAT for more details.

16.6.2.21 CGR10: Control Gain Register 10

Register Name: CGR10 Register Address: 0x13

bit7-RW-0	bit6-RW-0	bit5-RW-0	bit4-RW-0	bit3-RW-0	bit2-RW-0	Bit1-RW-0	bit0-RW-0
GIR				GIL			

Bits	Field	Description
7:4	GIR	ADC right channel PGATM input gain programming value.
3:0	GIL	ADC left channel PGATM input gain programming value.

NOTE: Please refer to section Programmable input attenuation amplifer: PGATM for more details.

16.6.2.22 AGC1: Automatic Gain Control Register 1

Register Name: AGC1 Register Address: 0x17

bit7-RW-0	bit6-RW-0	bit5-RW-1	bit4-RW-1	bit3-RW-0	bit2-RW-1	bit1-RW-0	bit0-RW-0
AGC_EN	Reserved	TARGET				Reserved	

Bits	Field	Description
7	AGC_EN	Selection of the AGC system. 0: inactive 1: enables the automatic level control
6	Reserved	This bit are not used, when read is 0.
5:2	TARGET	Target output level of the ADC. 0000: -6dB 0001: -7.5dB … by step of 1.5 dB 1111: - 28.5dB
1:0	Reserved	These bits are not used, when read is 00.

NOTE: Please refer to section AGC system guide for more details.

16.6.2.23 AGC2: Automatic Gain Control Register 2

Register Name: AGC2 Register Address: 0x18

bit7-RW-0	bit6-RW-0	bit5-RW-0	bit4-RW-0	bit3-RW-0	bit2-RW-1	bit1-RW-1	bit0-RW-1
NG_EN	NG_THR			HOLD			

Bits	Field	Description
7	NG_EN	Selection of the Noise Gate system. 0: inactive 1: enables the noise gate system
6:4	NG_THR	Noise Gate Threshold value. Input level (dB) < Noise Gate Level (dB). 000: -72 dB 001: -66 dB … by step of 6dB 111: -30 dB
3:0	HOLD	Hold time before starting AGC adjustment to the TARGET value. 0000: 0ms 0001: 2 ms 0010: 4 ms … Time Step x2

		1111: 32.768s

NOTE: Please refer to section AGC system guide for more details.

16.6.2.24 AGC3: Automatic Gain Control Register 3

Register Name: AGC3 Register Address: 0x19

bit7-RW-0	bit6-RW-1	bit5-RW-0	bit4-RW-0	bit3-RW-0	bit2-RW-1	bit1-RW-0	bit0-RW-0
ATK				DCY			

Bits	Field	Description
7:4	ATK	Attack Time - Gain Ramp Down. 0000: 32 ms 0001: 64 ms … by step of 32 ms 1111 : 512 ms
3:0	DCY	Decay Time - Gain Ramp up. 0000: 32 ms 0001: 64 ms … by step of 32 ms 1111: 512 ms

NOTES:

1 DCY and ATK registers values are delays between each step of gain.

2 Please refer to section AGC system guide for more details.

16.6.2.25 AGC4: Automatic Gain Control Register 4

Register Name: AGC4 Register Address: 0x1A

bit7-RW-0	bit6-RW-0	bit5-RW-0	bit4-RW-1	bit3-RW-1	bit2-RW-1	bit1-RW-1	bit0-RW-1
Reserved			AGC_MAX				

Bits	Field	Description
7	Reserved	These bits are not used, when read is 000.
4:0	AGC_MAX	Maximum Gain Value to apply to the ADC path. 00000: 0 dB 00001: 1.5dB … by step of 1.5dB 01111: 22.5dB 10000: 23 dB 10001: 23 dB

		10010: 23 dB 10011: 24.5dB … by step of 1.5dB 11111: 42.5dB

NOTE: Please refer to section AGC system guide for more details.

16.6.2.26 AGC5: Automatic Gain Control Register 5

Register Name: AGC5 Register Address: 0x1B

bit7-RW-0	bit6-RW-0	bit5-RW-0	bit4-RW-0	bit3-RW-0	bit2-RW-0	bit1-RW-0	bit0-RW-0
Reserved			AGC_MIN				

Bits	Field	Description
7:5	Reserved	These bits are not used, when read is 000.
4:0	AGC_MIN	Maximum Gain Value to apply to the ADC path. 00000: 0 dB 00001: 1.5dB … by step of 1.5dB 01111: 22.5dB 10000: 23 dB 10001: 23 dB 10010: 23 dB 10011: 24.5dB … by step of 1.5dB 11111: 42.5dB

NOTE: Please refer to section AGC system guide for more details.

16.6.2.27 TR1: Test Register 1 (internal used only)

16.6.2.28 TR2: Test Register 2 (internal used only)

16.6.3 Programmable gains

This section helps you to configure the programmable gain amplifier in the CODEC.
Internal signal VREFP is connected to AVDCDC Pin and internal signal VREFN is connected to AVSCDC Pin.
In this section, VREF equals to (VREFP – VREFN).

16.6.3.1 Programmable boost gain: GIM

In the same way, the following table gives the relation between the gain and the input level for the microphone input amplifier when GI = 0000.

GIM	Gain value (dB)	Maximum input amplitude
0	0	0.85*VREF
1	20	0.085*VREF

NOTES:

1 Maximum analog input amplitude value is given in Vpp differential.

2 Maximum analog input amplitude is referenced as Full Scale (FS). After conversion, the corresponding digital code of the output value varies from 0x7FFF down to 0x8000 for a 16-bit word. When the analog input amplitude is greater than FS, the dynamic characteristics are not guaranteed.

16.6.3.2 Programmable input attenuation amplifier: PGATM

The gain of PGATM may be programmed through GI[3:0]. The value of the gain is programmable from 0 to 22.5dB with a pitch of 1.5dB.

The gain and input levels are obtained according to the following table:

GI[3:0]	Decimal	Gain (dB)	Maximum input amplitude
0 0 0 0	0	0	0.85*VREF
0 0 0 1	1	1.5	0.715*VREF
0 0 1 0	2	3	0.602*VREF
0 0 1 1	3	4.5	0.506*VREF
0 1 0 0	4	6.0	0.426*VREF
0 1 0 1	5	7.5	0.358*VREF
0 1 1 0	6	9.0	0.302*VREF
0 1 1 1	7	10.5	0.254*VREF
1 0 0 0	8	12.0	0.214*VREF
1 0 0 1	9	13.5	0.180*VREF
1 0 1 0	10	15.0	0.151*VREF
1 0 1 1	11	16.5	0.127*VREF
1 1 0 0	12	18.0	0.107*VREF
1 1 0 1	13	19.5	0.090*VREF
1 1 1 0	14	21.0	0.076*VREF
1 1 1 1	15	22.5	0.064*VREF

NOTE: The last column of the table gives the maximum analog input to be applied on the MICi inputs. The value is given in Vpp differential. These values refer to the external voltage reference VREF equals to (VREFP – VREFN). The voltage levels depend on the VREF voltage.

16.6.3.3 Programmable attenuation: GOi

The attenuation of analog bypass path may be programmed independently for each channel through GO1L[4:0], GO1R[4:0], GO2L[4:0], GO2R[4:0], GO3L[4:0], GO3R[4:0]. The value of the gain is programmable from +6 to –22.5dB with a constant pitch as below.

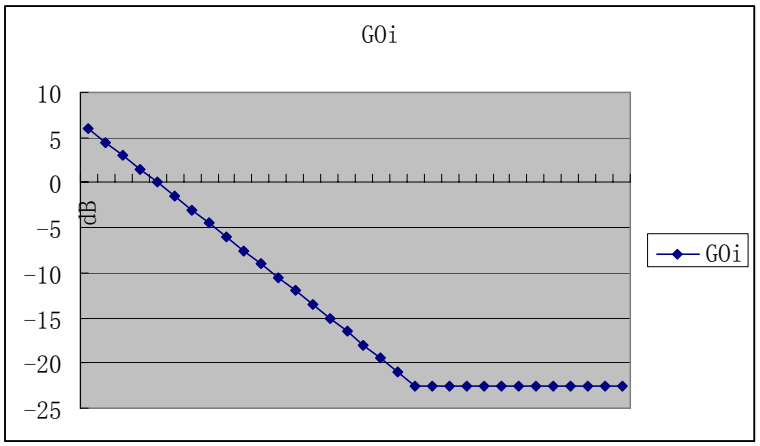


Figure 16-3 GOi values

The gain and output levels are obtained according to the following table:

Goi[4:0]	Decimal Value	Gain (dB)	Maximal input amplitude	Maximal output amplitude
0 0 0 0 0	0	+6.0	0.425*VREF	0.71*VREF
0 0 0 0 1	1	+4.5	0.506*VREF	0.71*VREF
0 0 0 1 0	2	+3.0	0.602*VREF	0.71*VREF
0 0 0 1 1	3	+1.5	0.715*VREF	0.71*VREF
0 0 1 0 0	4	+0	0.85*VREF	0.71*VREF
0 0 1 0 1	5	-1.5	0.85*VREF	0.597*VREF
0 0 1 1 0	6	-3.0	0.85*VREF	0.503*VREF
0 0 1 1 1	7	-4.5	0.85*VREF	0.423*VREF
0 1 0 0 0	8	-6.0	0.85*VREF	0.356*VREF
0 1 0 0 1	9	-7.5	0.85*VREF	0.299*VREF
0 1 0 1 0	10	-9.0	0.85*VREF	0.252*VREF
0 1 0 1 1	11	-10.5	0.85*VREF	0.212*VREF
0 1 1 0 0	12	-12.0	0.85*VREF	0.178*VREF
0 1 1 0 1	13	-13.5	0.85*VREF	0.150*VREF
0 1 1 1 0	14	-15.0	0.85*VREF	0.126*VREF

0 1 1 1 1	15	-16.5	0.85*VREF	0.106*VREF
1 0 0 0 0	16	-18.0	0.85*VREF	0.089*VREF
1 0 0 0 1	17	-19.5	0.85*VREF	0.075*VREF
1 0 0 1 0	18	-21.0	0.85*VREF	0.063*VREF
1 0 0 1 1	19	-22.5	0.85*VREF	0.053*VREF
	…	-22.5	0.85*VREF	0.053*VREF
1 1 1 1 1	31	-22.5	0.85*VREF	0.053*VREF

NOTES:

1 Maximal input amplitude and output amplitude value is Vpp single-ended.

2 Maximal input amplitude is the maximal value at input of Mixer on one of analog bypass or sidetone paths with no signal on DAC path.

3 Maximal output amplitude is the maximal value at output of Headphone, with no signal on DAC path.

16.6.3.4 Programmable attenuation: GOD

The attenuation of DAC path may be programmed independently for both channels through the pins GODL[3:0] and GODR[3:0]. The value of the gain is programmable from 0 to –22.5dB with a constant pitch as below.

GOD[3:0]	Decimal	Gain (dB)	Output amplitude [*2]
0 0 0 0	0	0	0.71*VREF
0 0 0 1	1	-1.5	0.597*VREF
0 0 1 0	2	-3.0	0.502 * VREF
0 0 1 1	3	-4.5	0.423*VREF
0 1 0 0	4	-6.0	0.356*VREF
0 1 0 1	5	-7.5	0.299*VREF
0 1 1 0	6	-9.0	0.252*VREF
0 1 1 1	7	-10.5	0.212*VREF
1 0 0 0	8	-12.0	0.178*VREF
1 0 0 1	9	-13.5	0.150*VREF
1 0 1 0	10	-15.0	0.126*VREF
1 0 1 1	11	-16.5	0.106*VREF
1 1 0 0	12	-18.0	0.089*VREF
1 1 0 1	13	-19.5	0.075*VREF
1 1 1 0	14	-21.0	0.063*VREF
1 1 1 1	15	-22.5	0.053*VREF

NOTES:

1 Output amplitude value is Vpp single-ended.

2 Output amplitude value is the value at output of headphone, for maximal amplitude 0.85*VREF (Vpp single-ended) at input of Mixer on DAC path, and no signal on Bypass path.

16.6.3.5 Programmable output amplifier: PGAT

The attenuation of PGAT may be programmed independently for the both channels through the registers bits GOL[4:0] and GOR[4:0]. The value of the gain is programmable from +4.5 to –33.5dB with a variable pitch as below:

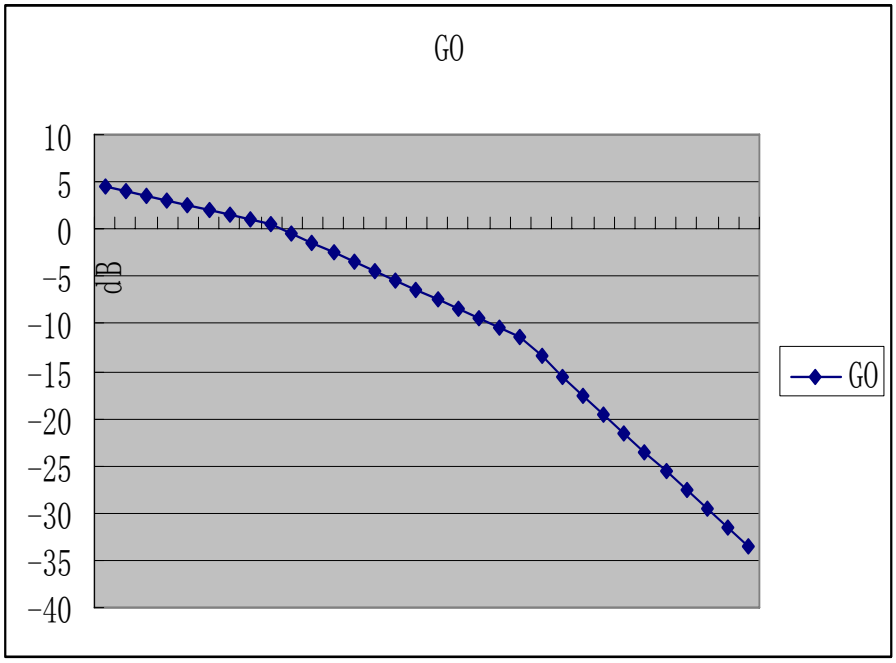


Figure 16-4 GO values

The gain and output levels are obtained according to the following table:

GO[4:0]	Decimal	Gain (dB)	Maximal PGAT input amplitude [*1]	Maximal PGAT output amplitude(*1)
0 0 0 0 0	0	+4.5	0.425*VREF	0.71*VREF
0 0 0 0 1	1	+4.0	0.451*VREF	0.71*VREF
0 0 0 1 0	2	+3.5	0.478*VREF	0.71*VREF
0 0 0 1 1	3	+3.0	0.506*VREF	0.71*VREF
0 0 1 0 0	4	+2.5	0.536*VREF	0.71*VREF
0 0 1 0 1	5	+2.0	0.568*VREF	0.71*VREF
0 0 1 1 0	6	+1.5	0.602*VREF	0.71*VREF
0 0 1 1 1	7	+1.0	0.637*VREF	0.71*VREF
0 1 0 0 0	8	+0.5	0.675*VREF	0.71*VREF
0 1 0 0 1	9	-0.5	0.757*VREF	0.71*VREF

0 1 0 10	10	-1.5	0.85*VREF	…
0 1 0 1 1	11	-2.5	0.85*VREF	…
0 1 1 0 0	12	-3.5	0.85*VREF	…
0 1 1 0 1	13	-4.5	0.85*VREF	…
0 1 1 1 0	14	-5.5	0.85*VREF	…
0 1 1 1 1	15	-6.5	0.85*VREF	…
1 0 0 0 0	16	-7.5	0.85*VREF	…
1 0 0 0 1	17	-8.5	0.85*VREF	…
1 0 0 1 0	18	-9.5	0.85*VREF	…
1 0 0 1 1	19	-10.5	0.85*VREF	0.251*VREF
1 0 1 0 0	20	-11.5	0.85*VREF	0.225*VREF
1 0 1 0 1	21	-13.5	0.85*VREF	0.178*VREF
1 0 1 1 0	22	-15.5	0.85*VREF	…
1 0 1 1 1	23	-17.5	0.85*VREF	…
1 1 0 0 0	24	-19.5	0.85*VREF	…
1 1 0 0 1	25	-21.5	0.85*VREF	…
1 1 0 1 0	26	-23.5	0.85*VREF	…
1 1 0 1 1	27	-25.5	0.85*VREF	…
1 1 1 0 0	28	-27.5	0.85*VREF	…
1 1 1 0 1	29	-29.5	0.85*VREF	…
1 1 1 1 0	30	-31.5	0.85*VREF	0.023*VREF
1 1 1 1 1	31	-33.5	0.85*VREF	0.017*VREF

NOTES:

1 Maximal PGAT input amplitude and output amplitude value is Vpp single-ended.

2 If the gain set to the value which is in the table cells in gray background, it may generated slight POP noise.

When the values of GO inputs are changed, the analog output amplitude is stabilized after about 1ms. The last column of the table gives the analog output voltage delivered on the AOHPL, AOHPR outputs and corresponding to a digital input at FS (Full Scale). The value is given in Vpp single-ended. These values refer to the external voltage reference VREF equals to (VREFP – VREFN). The voltage levels depend on the VREF voltage.

16.6.4 Sampling frequency: FREQ

The sampling frequency value is given in the FREQ[3:0] table below.

FREQ [3:0]	Sampling Rate (Fs)
0000	96kHz
0001	48kHz

0010	44.1kHz
0011	32kHz
0100	24kHz
0101	22.05kHz
0110	16kHz
0111	12kHz
1000	11.025kHz
1001	9.6kHz
1010	8kHz
1011	8kHz
1100	8kHz
1101	8kHz
1110	8kHz
1111	8kHz

NOTE: The sampling rate settings are the same as 8khz from 1010 to 1111, so the setting of FREQ from 1011 to 1111 could be ignored.

16.6.5 Programmable data word length

The Data Word Length block (DWL) allows selecting the length of the input data and of the output data between 24-/20-/18-/16-bit thanks to CR2.DAC_ADWL and CR2.ADC_ADWL (respectively for the DAC and ADC paths) in accordance with the following table:

*ADWL[1:0]	Word length
0 0	16-bit word length data
0 1	18-bit word length data
1 0	20-bit word length data
1 1	24-bit word length data

The size of the buses is always 24 bits, but the input/output data only use the number of MSB programmed with ADWL. The LSB are considered as ‘0’ in input and set to ‘0’ in output.
The capability to use a data word length of 16 bits is kept for compatibility with standard applications.

16.6.6 Ramping system guide

An internal mechanism is used to reduce output glitches when the headphone stage enters or leaves the power-down mode.

When the SB_OUT is set to ‘1’, the headphone output voltages (AOHPL, AOHPR, and AOHPM) are slowly decreased in the same time from AVDHP/2 down to 0. The output ramp waveform is programmable thanks to the CRR register.

When the SB_OUT is set to '0', the headphone output voltages (AOHPL, AOHPR, and AOHPM) are slowly increased in the same time from 0 to AVDHP/2.

An interrupt request is sent when the ramp completes.

Do not change the level of SB_OUT as long as the sequence due to the previous change is not complete or working not guaranteed.

In order to prevent audible glitch, it is required to power-down the output stage (SB_OUT=1) before changing the type of output load (capacitor less load or capacitor coupled load) with SB_MC.

The ramp time depends on the RATIO, KFAST and TRESH that are located in CRR register. CRR.RATIO set the Ratio; CRR.KFAST means the fast ratio and set the K_{fast}; CRR.TRESH means the threshold and set the TH.

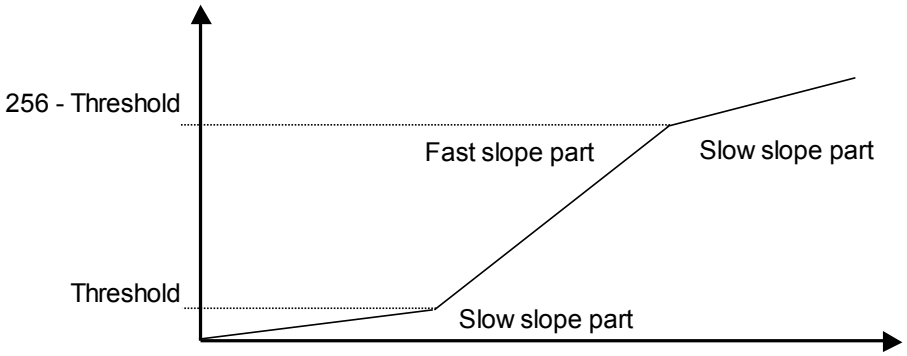


Figure 16-5 Ramp up

when CRR.TRESH = 11, the counter stays 2*Tslowstep on 127 at the middle of the falling and 2*Tslowstep on 128 at the middle of the rising.

The step count unit clock cycle length is called T_{step}, T_{step} = 39.16us at SYS_CLK = 12MHz.
If using the CRR reset value, the default Ramp Time duration is 224ms, and should keep this value.

The time parameters is calculated by the following formulas and finally get the total ramp time T_{ramp}.

Step time on fast part = $K_{fast} * T_{step}$
Step time on slow part = Ratio * $K_{fast} * T_{step}$
$T_{faststep} = K_{fast} * T_{step}$
$T_{slowstep}$ = Ratio * $K_{fast} * T_{step}$
$T_{ramp} = K_{fast} * T_{step} * (256 – 2 * TH) + R * K_{fast} * T_{step} * 2 * TH$

16.6.7 AGC system guide

For the microphone input to ADC path, an Automatic Gain Control (AGC) system allows to optimize

the signal swing at the input of the ADC.

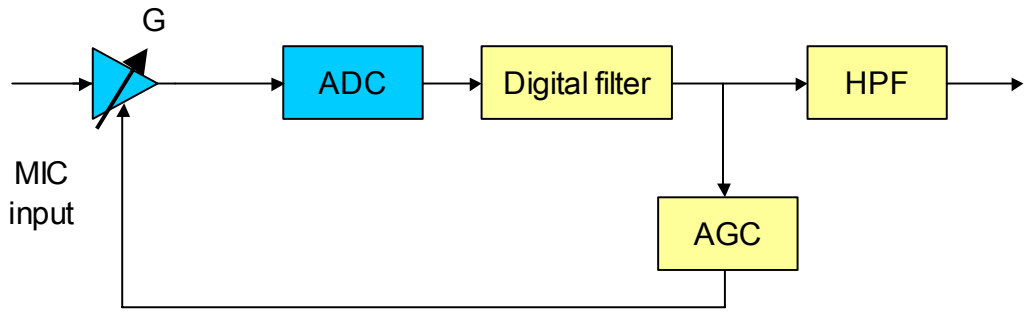


Figure 16-6 AGC Function Block Diagram

The AGC circuit compares the output of the ADC to a level and increases or decreases the gain of the microphone preamplifier to compensate. The full dynamic range of the ADC can be used automatically if the audio from the microphone is to be output digitally through the ADC.

The AGC_EN register bit enables the AGC system. If using the AGC system, CR3.INSEL must be clear to 00.
If not using AGC system, AGC1.AGC_EN must be clear 0.

The AGC system is used at the MIC input, and the Cut Frequency of HPF filter is 300 Hz.

If the AGC system is enabled, the system of gain control will directly assign the values of the gains GIL, GIR (shown as G in Figure 2-4 AGC Function Block Diagram) of the PGATM and GIM of the MIC boost stage.

16.6.7.1 AGC operating mode

The AGC system adapts the gain stages (PGATM and MIC boost stage) in order to best reach a setting target that AGC1.TARGET sets the desired ADC output range level. The limits of the gain variation are set by AGC4.AGC_MAX and AGC5.AGC_MIN.

The AGC system should not alter the dynamic content of the signal, so AGC system is continuously adapting the gain to fit the target level. The hold time between two consecutives gain adjustments is modifiable by the AGC2.HOLD register value.
After this hold time, there are two conditions:

- If the output level is lower than AGC1.TARGET, the gain is increased step by step in accordance to the AGC3.DCY register value.
- If the output level is higher than AGC1.TARGET, the gain is decreased step by step in accordance to the AGC3.ATK register value.

The following figure illustrates the behavior of AGC system:

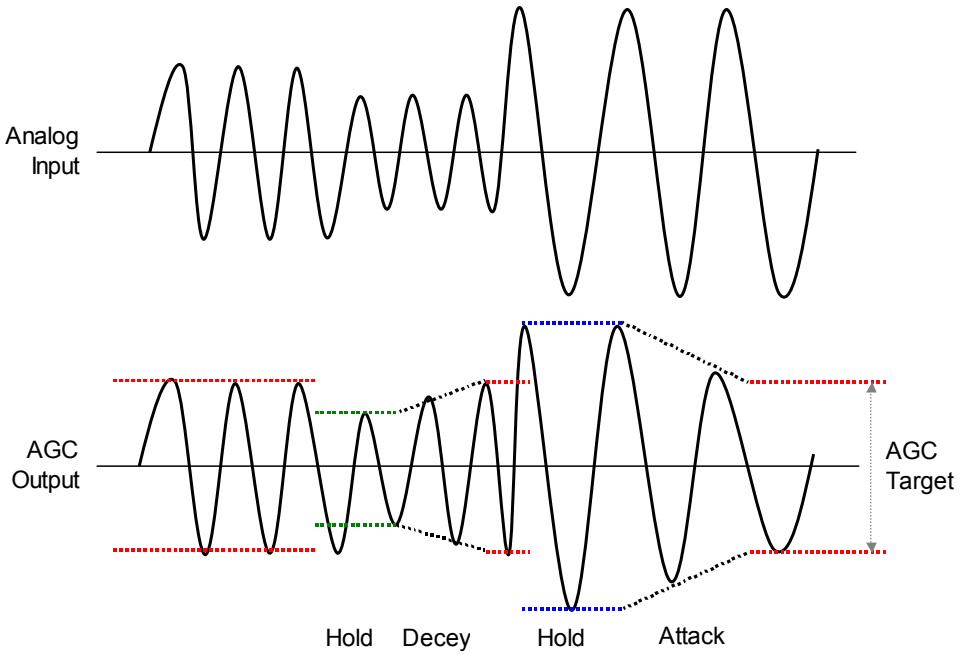


Figure 16-7 AGC adjusting waves

AGC system has a noise-gating feature to prevent gain increasing when no signal or small signal is present at the input, which is enabled using the AGC2.NG_EN register bit. And the noise gate threshold is set by the AGC2.NG_THR register value.

The following graph summarizes the operations and shows more details.

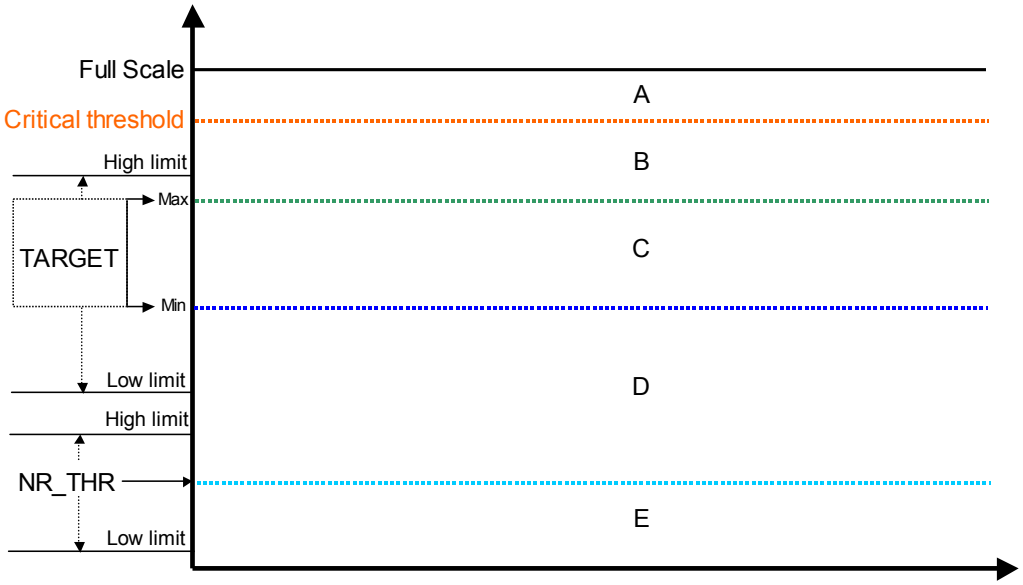


Figure 16-8 AGC adjust areas

The areas from A to E is deferent working area of AGC system, which is listing below:

A: If the signal level is in this critical area: the AGC system decreases quickly the gain at the input of the ADC until the signal goes under the critical threshold.

B: If the signal level remains in this area after the HOLD delay: the AGC system decreases the gain at the input of the ADC until the signal reaches the target area with a slope defined by AGC3.ATK register value.

C: If the signal level is in this area: the AGC system does not perform gain adjustment.

D: If the signal level remains in this area after the HOLD delay: the AGC system increases gain at the input of the ADC until the signal reach the target area with a slope defined by AGC3.DCY register value.

E: If the signal level is in this range: the AGC system considers the signal as noise and does not perform gain adjustment.

16.6.8 CODEC Operating modes

Different operating modes are available:

- Power-up mode: During power on time, CODEC is in this mode.
- Reset mode: When NRST is low, CODEC is in this mode.
- Soft mute mode: When CR1.DAC_MUTE is 1, CODEC is in this mode.
- Complete Power-down mode: After RESET, CODEC is in this mode.
- SLEEP modes: When PMR2.SB_SLEEP is 1, CODEC is in this mode.
- Normal mode: When CODEC is not in above mode, it is in this mode. This mode has three modes: RECORD mode, REPLAY mode, RECORD_REPLAY mode.

The power diagram is shown below.

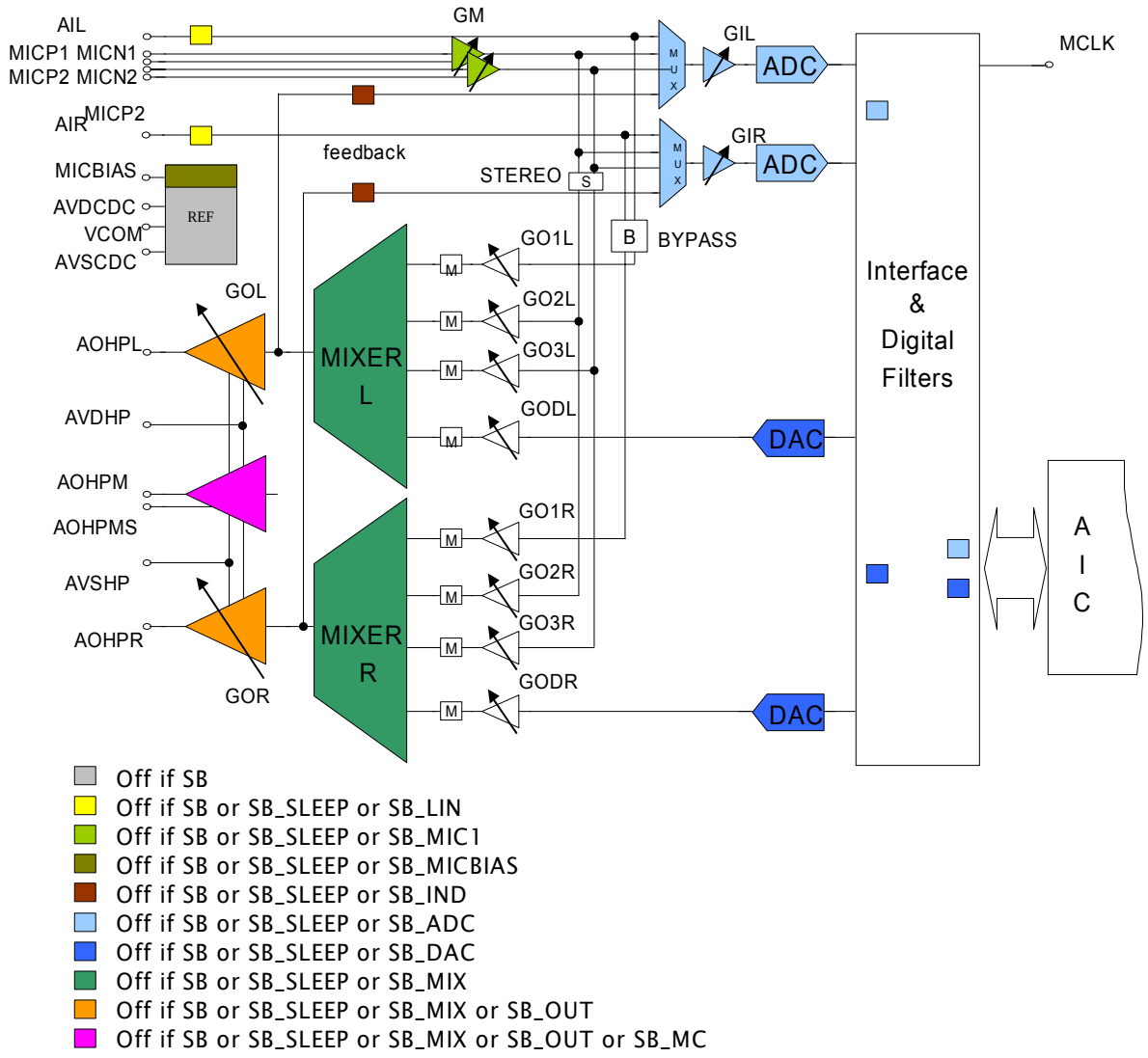


Figure 16-9 CODEC Power Diagram

There are many power parts of CODEC. Any part could be powered down independently.

16.6.8.1 Power-On mode and Power-Off mode

When the power supply ramps up, hiCODIv-9001-2G enters the power-on mode. During the reset, the CODEC is put in stand-by in order to reduce audible pops.

The CODEC doesn't handle the power supply ramp down on itself. The software has to turn the CODEC in complete stand-by mode before the power supply starts to ramp down.

16.6.8.2 RESET mode

The reset input signal is asynchronous; the reset minimum duration is one SYS_CLK cycle.
During the power-up mode and system reset, the CODEC goes into Reset mode.
After system reset the CODEC will exit Reset mode and go to STANDBY mode.

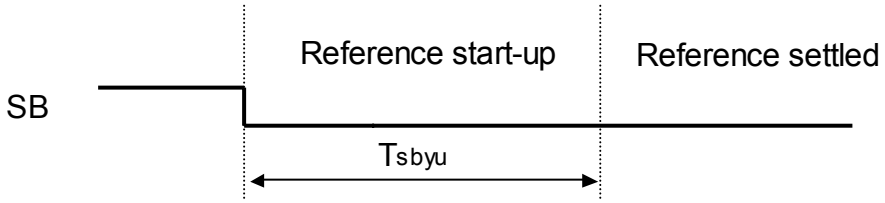
NOTES:

1 Except during the power-up mode, do NOT perform any reset in order to avoid audible pops.
2 Resetting the CODEC during normal operating mode will turn instantaneously the CODEC in STANDBY mode. This will lead to generate a large audible pop.

16.6.8.3 STANDBY mode

CODEC goes to STANDBY mode when the SB register bit equals '1', and all functions including ADC path, DAC path and analog references will stop and whole CODEC is shutdown for saving power. CODEC is complete down in this mode.

During the STANDBY mode, the power consumption is reduced to a minimum, so it is also called Complete Power-Down mode. When SB is set to '0', CODEC leaves the STANDBY mode. It is necessary to wait some time before the CODEC references settle. This time is called Tsbyu.



The typical value of T_{sbyu} is 250ms, maximizes 500ms(TBC).

16.6.8.4 Soft Mute mode

Soft Mute mode is used in order to reduce audible parasites when before the DAC enters or after leaves the Normal mode. Set the CR1.DAC_MUTE register bit to 1, it will go to Soft Mute mode.

Set CR1.DAC_MUTE to 1 puts the DAC in Soft Mute mode. The CODEC decreases progressively the digital gain from 0dB to -∞. When the gain down sequence is completed, the signal of the DAC is equal to 0 whatever the value of the digital input data is. Then CODEC generates an interrupt and if ICR.GDD_MASK is 0, and set IFR.GAIN_DOWN_DONE register bit to 1.

During Soft Mute mode, the DAC is still converting but the output final voltages (AOL, AOR) are equal to VREF/2, so the differential of the Headphone voltage is zero that cause no sound output.

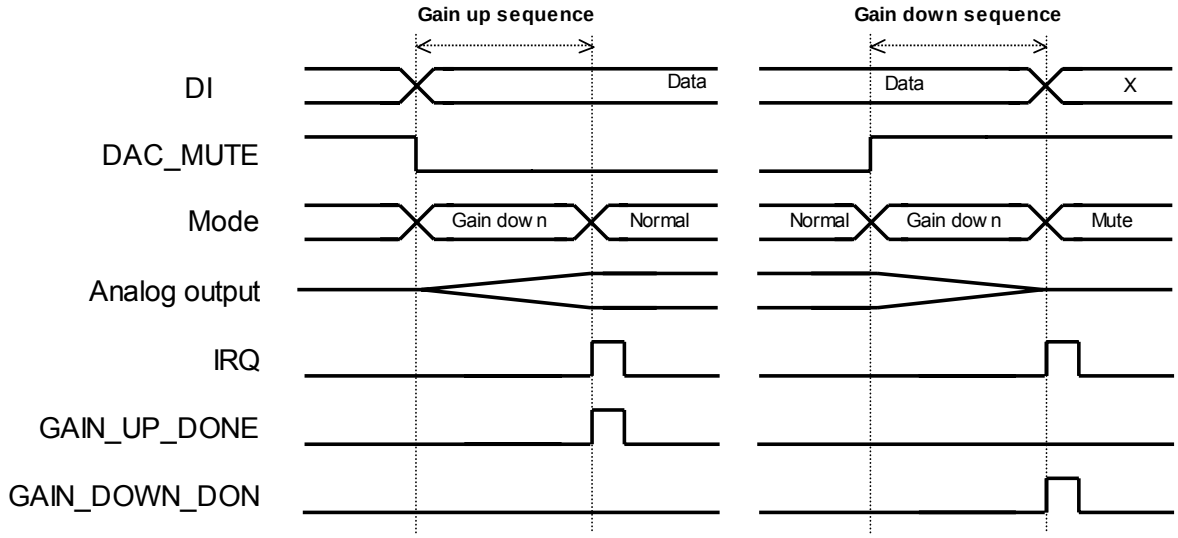


Figure 16-10 Gain up and gain down sequence

In the opposite, when CR1.DAC_MUTE is set to 0, the DAC leaves the Soft Mute mode by increasing progressively the digital gain from -∞ to 0dB. When the gain up sequence is completed, the DAC returns in Normal mode. The CODEC then generates an interrupt and if ICR.GDD_MASK is 0, and set IFR.GAIN_UP_DONE register bit to 1.

After exiting Soft Mute mode, the DAC output will flow the DAC input data, and there is sound in the Headphone.

The duration of gain down and gain up sequences are nearly independent of Fs as shown below:

Fs(kHz)	Time(ms)	Fs(kHz)	Time(ms)	Fs(kHz)	Time(ms)
96	17.72	24	17.25	11.025	17.73
48	17.72	22.05	17.73	9.6	17.98
44.1	17.73	16	17.25	8	17.25
32	17.96	12	17.25		

NOTES:

1. **Do NOT change the value of DAC_MUTE while the effect of the previous change is not reached, or the working is not guaranteed.**
2. **Do NOT enter in stand-by mode while the gain sequence is not completed, or the working is not guaranteed.**

16.6.8.5 Power-Down mode and SLEEP mode

Twelve stand-by inputs allow putting independently the different parts of CODEC into Power-Down mode.

16.6.8.6 Working modes summary

Different working modes are sum-up in the following table (non exhaustive table):

Working Mode		SB	SB_SLEEP	SB_DAC	SB_MIX	SB_OUT	SB_MC	SB_ADC	SB_MICBIAS	SB_LIN	SB_MIC1	SB IND	INSEL[1:0]	DACSEL	BYPASS	SIDETONE1	MICSTEREO	HP_DIS	DAC_MUTE
0. Reset / Power-On / Power_Off (After)		1	1	1	1	1	0	1	1	1	1	1	00	1	0	0	0	0	1
1. STANDBY		1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
2. SLEEP		0	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
3. RECORD	Mono mic1 input	0	0	-	-	-	-	0	-	-	0	-	00	-	-	-	0	-	-
	Line input	0	0	-	-	-	-	0	-	0	-	-	10	-	-	-	-	-	-
4. REPLAY	DAC to headphone	0	0	0	0	0	0	-	-	-	-	-	-	1	0	0	-	0	0
	Line to headphone	0	0	-	0	0	0	-	-	0	-	-	-	0	1	0	-	0	-
	Mic1 to headphone	0	0	-	0	0	0	-	-	-	0	-	-	0	0	1	0	0	-
	All inputs mix to headphone	0	0	0	0	0	0	-	0	0	0	-	-	1	1	1	-	0	0
5.RECORD_ REPLAY	Playback with Record from Mic1	0	0	0	0	0	0	-	-	-	-	-	-	1	0	0	-	0	0
	Playback with Record from Line	0	0	0	0	0	0	-	-	-	-	-	-	1	0	0	-	0	0
	Playback with Record from Mic1, Mixer	0	0	0	0	0	0	-	-	-	-	-	-	1	0	0	-	0	0
	Playback with Record from Line, Mixer	0	0	0	0	0	0	-	-	-	-	-	-	1	0	0	-	0	0
	Playback with Record from Mic1 with playback	0	0	0	0	0	0	-	-	-	-	-	-	1	0	0	-	0	0
Mixer output		0	0	-	0	-	-	0	-	-	-	0	11	-	-	-	-	-	-
6. “Default mode” (for test)		0	0	0	0	0	0	0	1	0	1	1	00	1	0	0	0	0	0

NOTE: The ‘-‘ means don’t care this bit, but most of them should be set to 1 for reduce power.

16.6.8.7 SYS_CLK turn-off and turn-on

The main clock of CODEC is called SYS_CLK, which is generated in CPM module and called cpm_i2s_sysclk.

During the SLEEP mode and the complete power-down mode, the main clock SYS_CLK may be

stopped to reduce the power consumption to the leakage currents only. In other modes, the main clock SYS_CLK must not be stopped.

The main clock SYS_CLK must not be stopped until CODEC has reached the complete power-down mode and must be restarted before leaving the power-down mode.

16.6.8.8 Requirements on mixer and PGATM inputs selection and power-down modes

The following rules must be respected in order not to damage performances and to keep the functionality:

- If SB_LIN is set to 1, BYPASS must be equal to 0.
- If SB_MIC1 or SB_MIC2 is set to 1, SIDETONE1 and MICSTEREO must be equal to 0.
- If SB_DAC is set to 1, DACSEL must be equal to 0.

16.6.8.9 Anti-pop operation sequences

The main idea of this section is to describe the sequences to perform to minimize the audible pop to the minimum for the headphone output.

Due to the large number of stand-by combinations and to be the most flexible, the handling of the sequence from one working mode to another is left to the software. So for helping the software designer in this task, some specific sequences are automatically performed by CODEC and an interrupt mechanism (IRQ signal and associated registers) warns the application when these sequences end.

16.6.8.9.1 Initialization and configuration

To use the embedded CODEC with AIC, several AIC registers should be set up the below register of AIC before start the CODEC:

AICFR.ICDC = 1
AICFR.AUSEL = 1
AICFR.BCKD = 0
AICFR.SYNCD = 0
I2SCR.AMSL = 0
I2SCR.ESCLK = 1

16.6.8.9.2 Start up sequence

This sequence is from Power-on mode to CODEC REPLAY mode.
The intent of the following sequence is to prevent for large audible glitches due to the system start-up with the CODEC.
Before this sequence, setup the AIC properly.

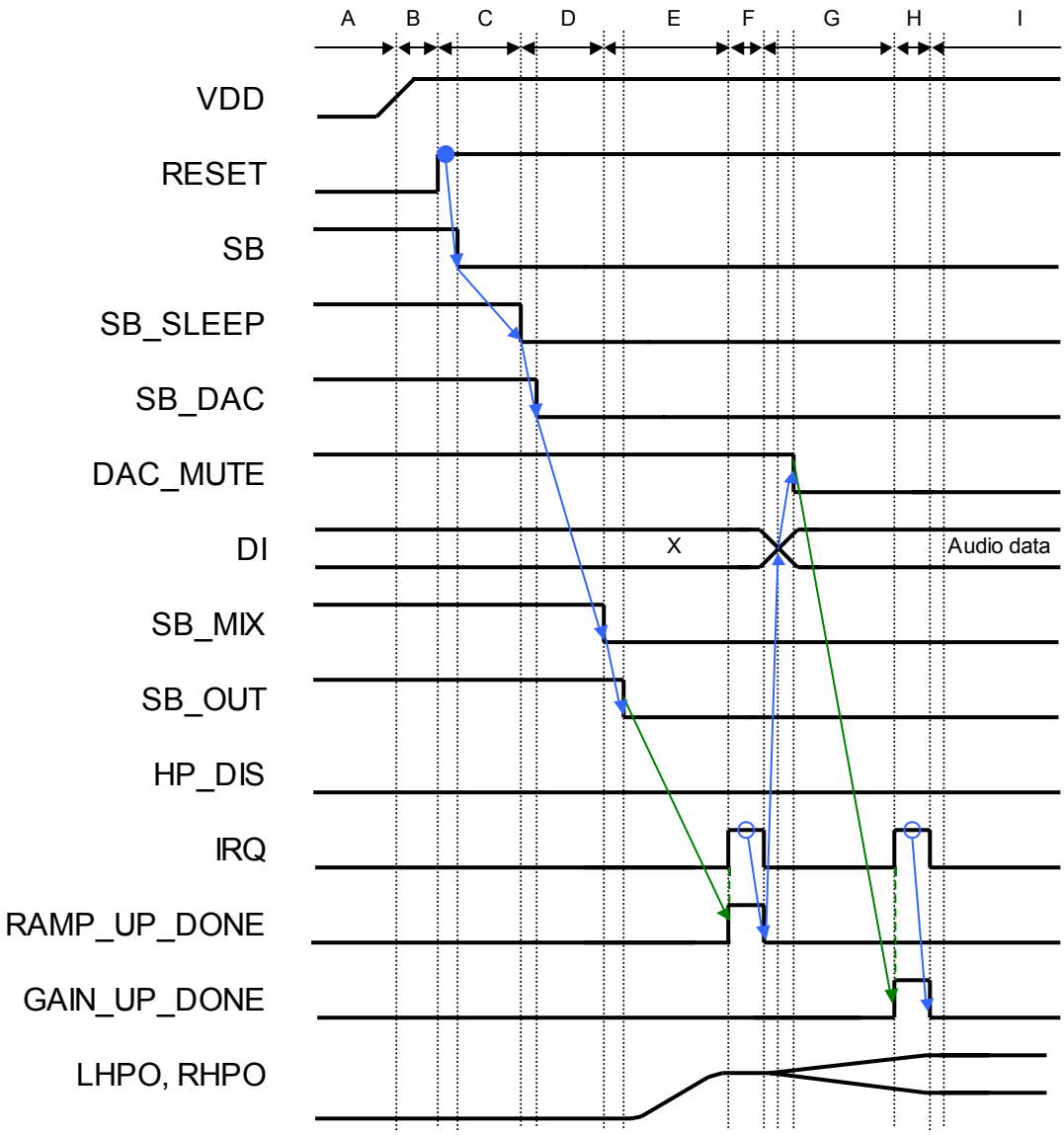


Figure 16-11 Start up sequence

NOTES:

1 The sequences in blue are manually handled by the software.

2 The sequences in black are automatically handled by the CODEC.

SEQUENCE:

A: Initial state.

The power supply is off.

B: Power supply ramp up.

The RESET of CODEC is '0' during system reset or other form reset.

C: Starting of CODEC reference.

The software turns the CODEC on SLEEP mode by clearing SB register bit to 0.

D: Turns on the DAC.
After waiting the Tsbyu duration (for example, on event generated by a timer at the software level), the application turns on the DAC by clearing SB_SLEEP and SB_DAC register bits to 0.

E: Ramp up cycle.
After waiting 1ms (TBC), the software turns on the mixer and the headphone output stages by clearing SB_MIX, SB_OUT to 0.

F: IRQ generation.
Once the ramp up cycle completes, the CODEC sets the RAMP_UP_DONE flag to 1 and generates an interrupt.

G: IRQ handling and gain up cycle.
The software handles the interrupt and resets the RAMP_UP_DONE flag and releases the mute of the DAC by clearing DAC_MUTE register bit to 0.
In the same time, the software sends valid audio data to the DAC.

H: IRQ generation.
Once the gain up cycle completes, the CODEC sets the GAIN_UP_DONE flag to 1 and generates an interrupt.

I: IRQ handling and active mode.
The software handles the interrupt and resets the GAIN_UP_DONE flag.
The DAC is now fully activated.

The sequence from C to I can be used to switch from the Stand-by mode to the active mode such as REPLAY mode.
The sequence from D to I can be used to switch from the SLEEP mode to the active mode such as REPLAY mode.

16.6.8.9.3 Shutdown sequence

This sequence is from CODEC REPLAY mode to STANDBY mode.
The intent of the following sequence is to prevent for large audible glitches due to the system shutdown with the CODEC.

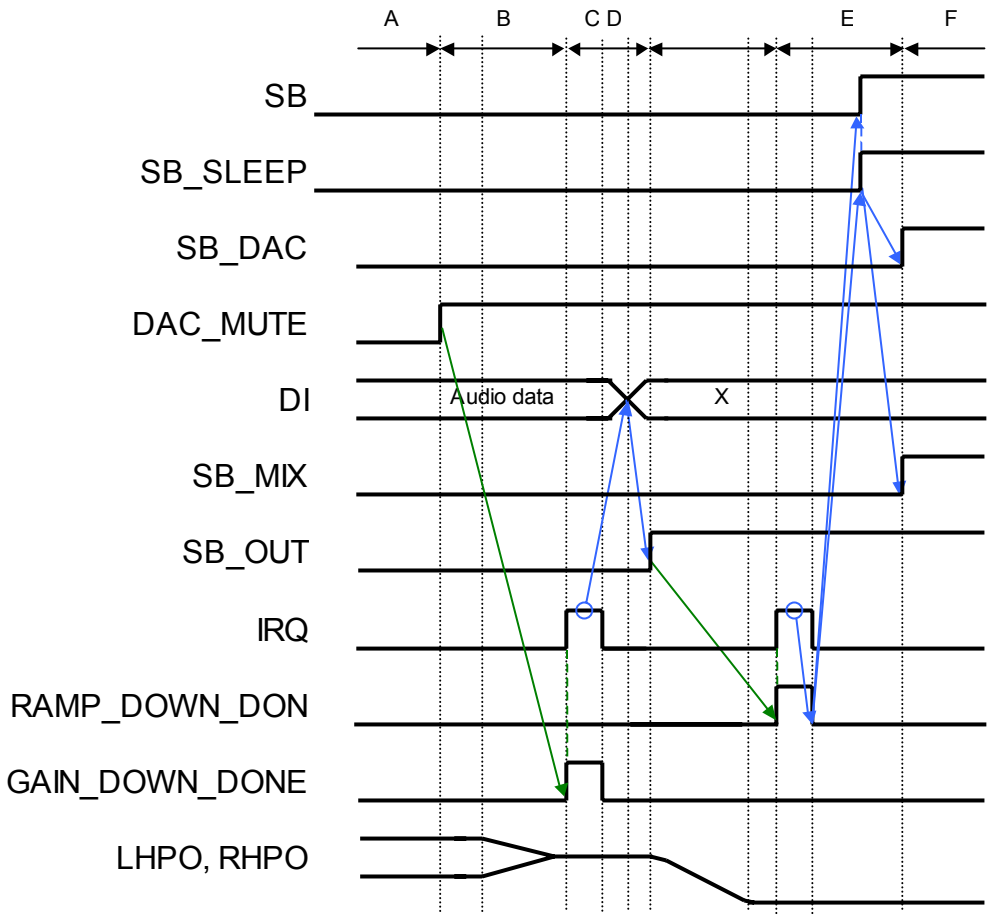


Figure 16-12 Shutdown sequence

NOTES:

1 The sequences in blue are handled by the software.

2 The sequences in black are automatically handled by the CODEC.

SEQUENCE:

A: Initial state.

It's a long time after the power supply on; CODEC is in REPLAY mode and DAC is activated.

B: Gain down cycle.

The software activates the mute of the DAC by setting DAC_MUTE register bit to 1.

Once the gain down cycle completes, the CODEC sets the GAIN_DOWN_DONE flag to '1' and generates an interrupt.

C: IRQ handling and ramp down cycle.

The software handles the interrupt and resets the GAIN_DOWN_DONE flag.

The software then turns off DAC output stage by setting SB_OUT register bit to 1.

D: IRQ generation.

Once the ramp down cycle completes, the CODEC sets the RAMP_DOWN_DONE flag to '1' and generates an interrupt.

E: IRQ handling.

The software handles the interrupt and resets the RAMP_DOWN_DONE flag.
The software turns off the DAC by setting SB_MIX, SB_DAC register bits to 1.
The software turns off the CODEC by setting the SB_SLEEP, SB register bits to 1.

F: Ideal.
Now, the CODEC is in STANDBY Mode.

16.6.9 Circuits design suggestions

This section lists a few PCB design suggestions with difference using mode.

16.6.9.1 Avoid quiet ground common currents

16.6.9.1.1 References pins

To work properly, CODEC requires few additional external components.

CODEC includes an internal voltage reference based on a resistive potential divider connected between AVDCDC and AVSCDC. For a correct working, it is required to connect two decoupling capacitor (10µF tantalum and 100nF ceramic) called Cext between the pins VREF and AVSCDC.

16.6.9.1.2 Power supply pins

CODEC analog power supplies require external decoupling capacitors. For each power supply, one 100nF ceramic has to be used. The ceramic capacitor has to be kept as close as possible to IC package (closer than 0.2 inch). One tantalum has to be used to decouple the analog power supply provided to the CODEC. Its value depends on the power supply generator; its typical value is between 1uF and 10uF. Ideally use separate ground planes for analog and digital parts.

Connect all ground pins with thick traces to power plane in order to ensure lowest impedance connections.

16.6.9.2 Capacitor-less headphone connection

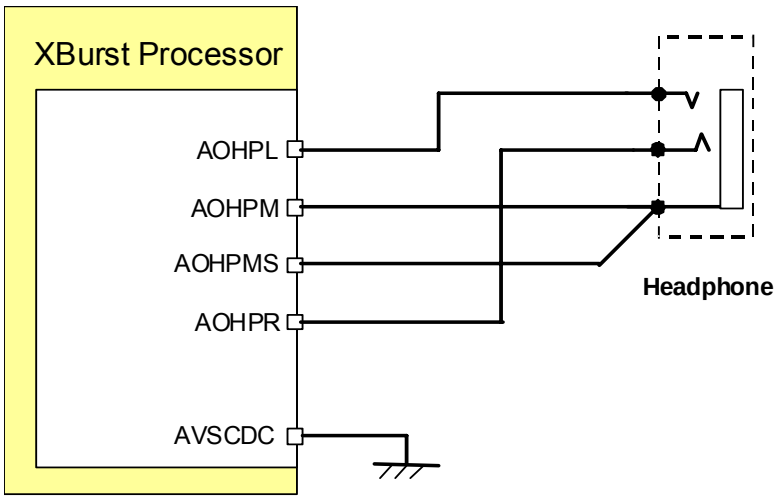


Figure 16-13 Capacitor-less connection

The AOHPL and AOHPR pins are applied directly to the loads. The ground of the headphone is connected to AOHPM.

The DC value of the signal AOHPL or AOHPR equals to AVDCDC/2.
AOHPM and AOHPMS have to be connected together as close as possible of the headphone connector.

16.6.9.3 Capacitor-coupled headphone connection

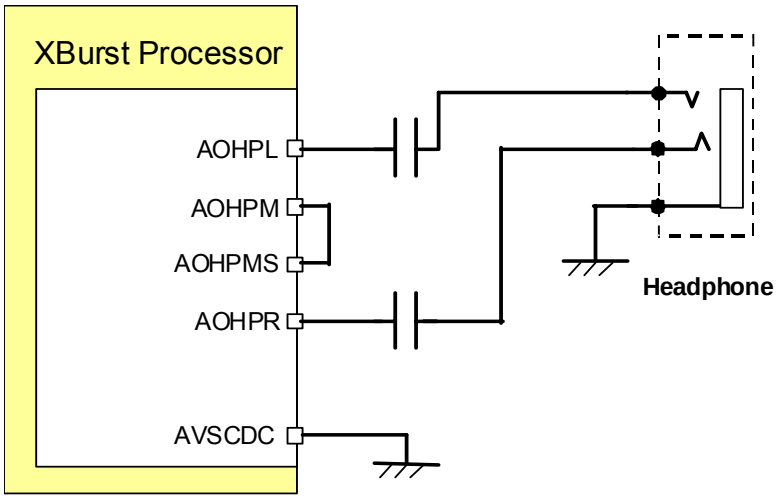


Figure 16-14 Capacitor-coupled connection

The AOHPL and AOHPR pins are connected to the headphone through an external bypass capacitor

which is a DC blocking capacitors.

This capacitor is called C_L. When the headphone resistance R_L is 16 Ohm, C_L equals 200 uF to 1 uF.

The DC value of the signal AOHPL or AOHPR equals to AVDCDC/2.

The ground of the headphone is connected to AVSCDC.

16.6.9.4 Microphone connection

This section is talking about single-ended microphone connection with single-ended microphone input.

Specific value of resistor (R, commonly from 2.2 kOhm to 4.7 kOhm) and Vmicbias (usually from 1 to 2V or more) depends on the selected EC (Electret Condenser) microphone.

The 1nf decoupling capacitance removes high frequency noise of the chip.

Setting SB_MIC1/SB_MIC2 to 1 will close microphone input path for saving power, also setting SB_MICBIAS to 1 will close MICBIAS stage and the MICBIAS output voltage will be zero.

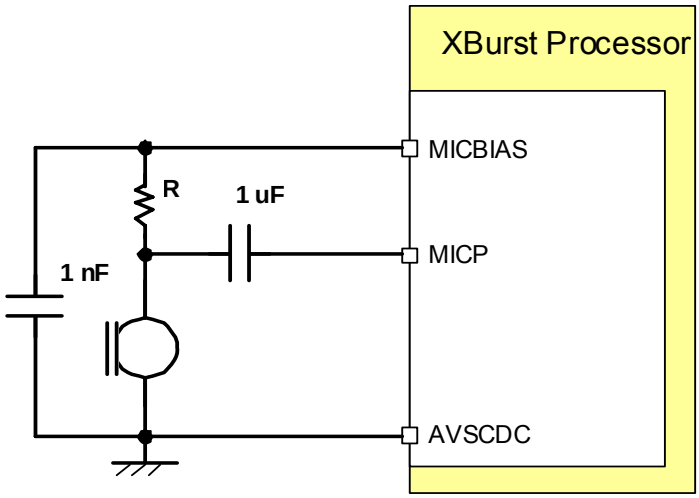


Figure 16-15 MIC connection with MICBIAS

MICBIAS output voltage scales with AVDCDC, equals to 5/6*AVDCDC (typical 2.75v).
MICBIAS output current is 4mA max.
MICBIAS output noise is 40uVrms max.

Of cause, If there is more accurate $V_{MICBIAS}$ off the chip, should use the circuit shown below:

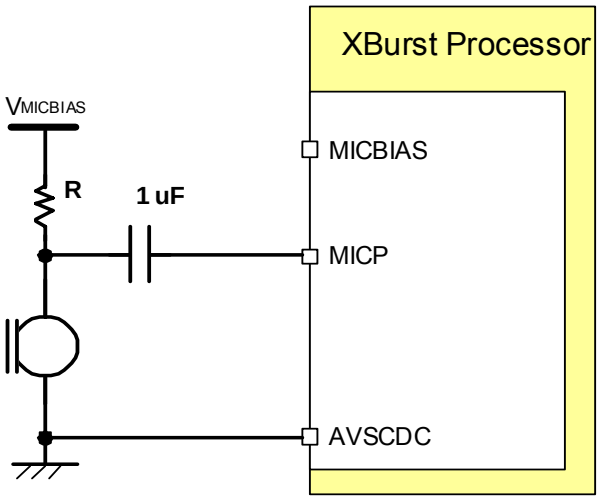
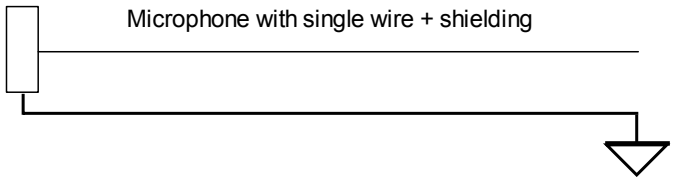


Figure 16-16 MIC connection with external $V_{MICBIAS}$

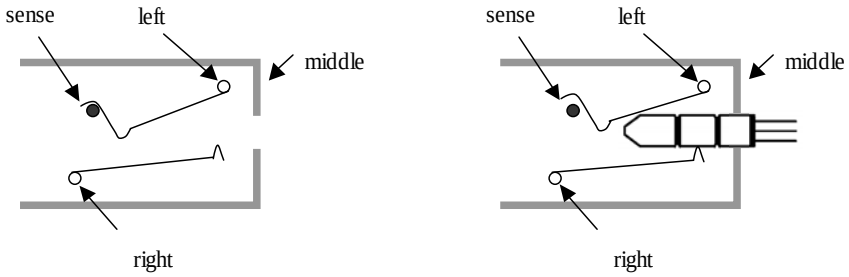
This configuration is better suited for microphone with single wire + shielding.



The AVSCDC Pin is connected the analog quiet reference ground in the chip (refers to Grounds and analog signal references). So the ground of MIC must be connected to AVSCDC using a star connection.

16.6.9.5 Description of the connections to the jack

When the jack is inserted, “sense” and “left” are disconnected. The “left” pin is connected to AOHPL, “right” pin is connected to AOHPR, and the “sense” pin is connected to HPSENSE.



16.6.9.6 Grounds and analog signal references

In order to limit the parasitic disturbances from the AVSHP output power supplies to inter VREFN quiet ground(which is using AVSCDC pin), should use the following principle to distribute the grounds.

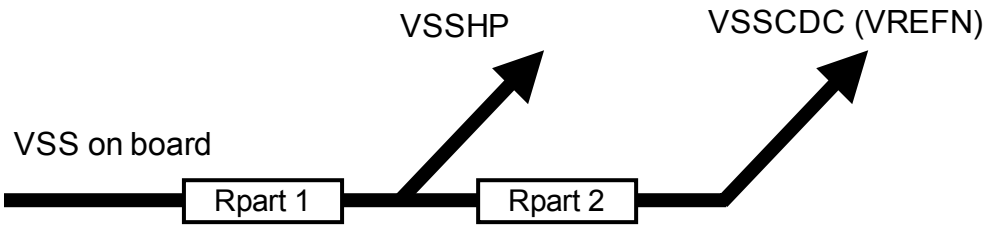


Figure 16-17 Ground distributing

Minimize the values of the connections parasitic resistance Rpar1, Rpar2.

Take a special care for Rpar1 in order to limit the disturbance from the output stages (AVSHP) to the signal reference (VREFN).

The reference of the input signals must be connected to VREFN (quiet ground which using the AVSCDC pin) using a star connection.

In the chip, The AVSHP and AVSCDC pin is very close, so could connect together with out this reference, please refer to PCB considerations.

16.6.9.7 PCB considerations

The reference PCB design is shown below:

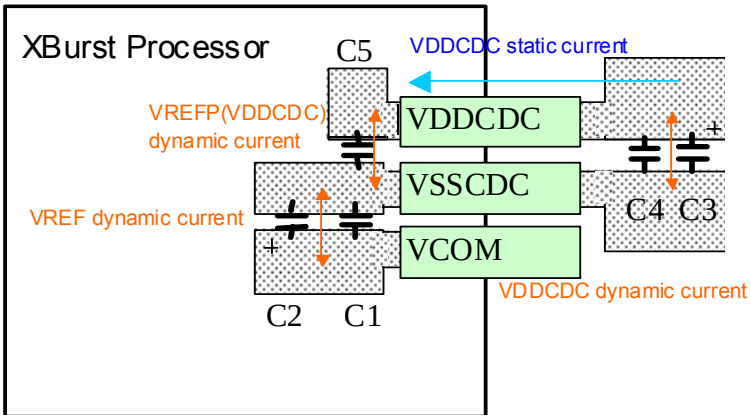


Figure 16-18 the bottom corner of chip PCB Layer

C1, C2, C3, C4 and C5 are defined in section Required external components.
This is just an example reference. You should change and select the PCB layer and route with your design constraints.

16.6.9.8 Required external components

The following table summarizes the external components required for a proper working of CODEC, except those used for the analog input and output signals.

Name	Description	Typical Value	Unit
C1	Ceramic reference decoupling capacitor. Cext	100	nF
C2	Tantalum reference decoupling capacitor. Cext	4.7	uF
C3	Tantalum analog power supply decoupling capacitor	1 to 10	uF
C4	Ceramic AVDCDC decoupling capacitor.	100	nF
C5	Ceramic inter signal VREFP decoupling capacitor (1)	100	nF
C6	Ceramic AVDHP decoupling capacitor. Not Used in PCB considerations.	100	nF
C7	MICBIAS decoupling capacitor, Refer to Microphone connection.	1	uF

17 AC97/I2S Controller

17.1 Overview

This chapter describes the AIC (AC'97 and I^2S Controller) included in this processor.

The AIC supports the Audio Codec '97 Component Specification 2.3 for AC-link format and I2S or IIS (for inter-IC sound), a protocol defined by Philips Semiconductor. Both normal I2S and the MSB-justified I2S formats are supported by AIC.

AIC consists of buffers, status registers, control registers, serializers, and counters for transferring digitized audio between the processor's system memory and an internal I2S CODEC, an external AC97 or I2S CODEC. AIC can record digitized audio by storing the samples in system memory. For playback of digitized audio or production of synthesized audio, the AIC retrieves digitized audio samples from system memory and sends them to a CODEC through the serial connection with AC-link or I2S formats. The internal or external digital-to-analog converter in the CODEC then converts the audio samples into an analog audio waveform. The audio sample data can be stored to and retrieved from system memory either by the DMA controller or by programmed I/O.

The AC-link is a synchronous, fixed-rate serial bus interface for transferring CODEC register control and status information in addition to digital audio. Where both normal I2S and MSB-justified-I2S work with a variety of clock rates, which can be obtained either by dividing the PLL clock by two programmable dividers or from an external clock source.

For I2S systems that support the L3 control bus protocol, additional pins are required to control the external CODEC. CODECs that use an L3 control bus require 3 signals: L3_CLK, L3_DATA, and L3_MODE for writing bytes into the L3 bus register. The AIC supports the L3 bus protocol via software control of the general-purpose I/O (GPIO) pins. The AIC does not provide hardware control for the L3 bus protocol.

To control the internal CODEC, internal CODEC Spec can be referenced.

This chapter describes the programming model for the AIC. The information in this chapter requires an understanding of the AC'97 specification, Revision 2.3.

17.1.1 Block Diagram

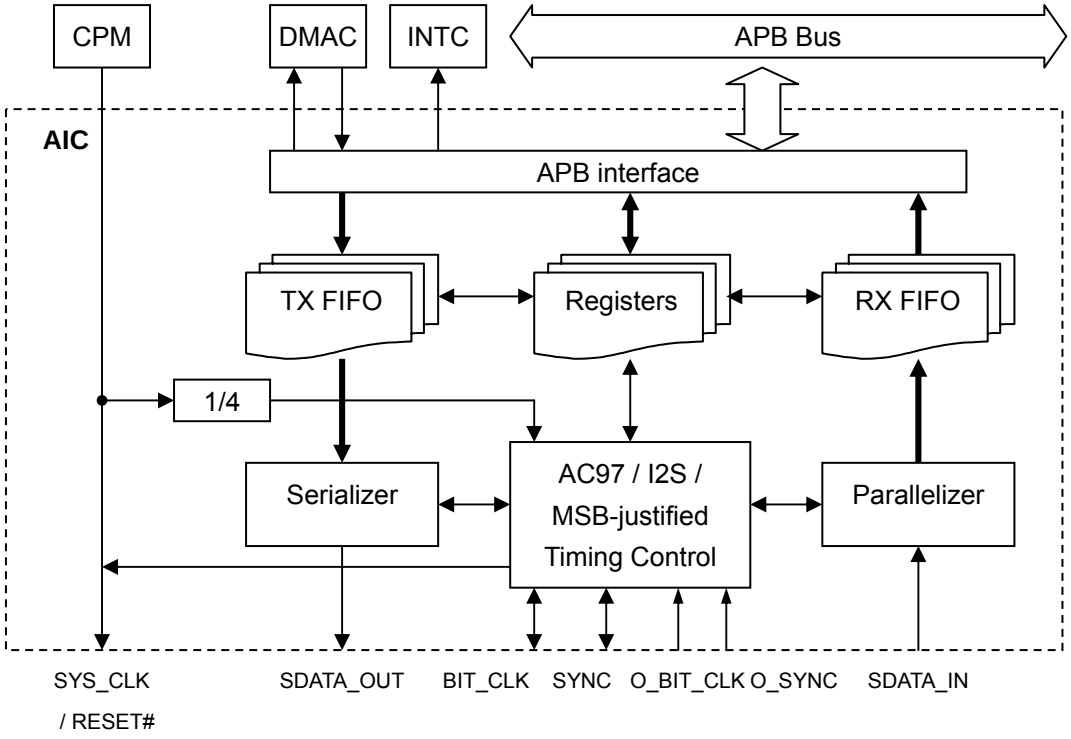


Figure 17-1 AIC Block Diagram

The O_BIT_CLK and O_SYNC ports are only used by inter CODEC.

17.1.2 Features

AIC support following AC97/I2S features:

- 8, 16, 18, 20 and 24 bit audio sample data sizes supported
- DMA transfer mode supported
- Stop serial clock supported
- Programmable Interrupt function supported
- Support mono PCM data to stereo PCM data expansion on audio play back
- Support endian switch on 16-bits audio samples play back
- Support variable sample rate in AC-link format
- Multiple channel output and double rated supported for AC-link format
- Power Down Mode and two Wake-Up modes Supported for AC-link format
- Internal programmable or external serial clock and optional system clock supported for I2S or MSB-Justified format
- Internal I2S CODEC supported
- Two FIFOs for transmit and receive respectively with 32 samples capacity in every direction

17.1.3 Interface Diagram

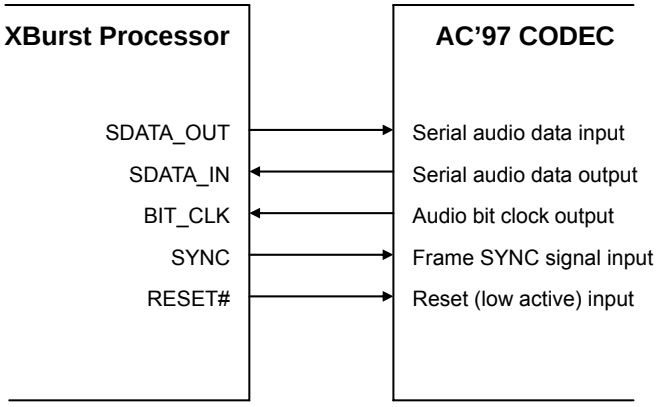


Figure 17-2 Interface to an External AC'97 CODEC Diagram

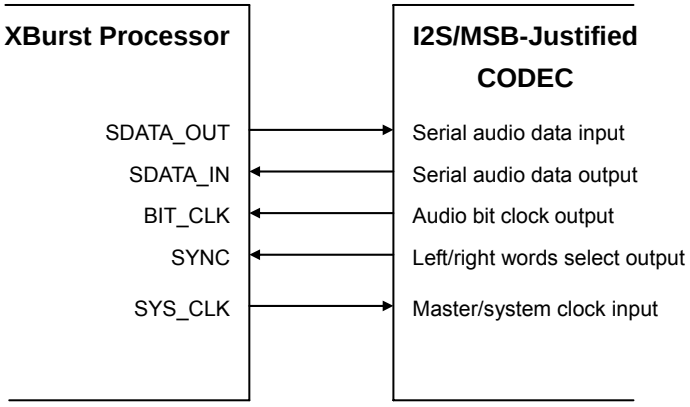


Figure 17-3 Interface to an External Master Mode I2S/MSB-Justified CODEC Diagram

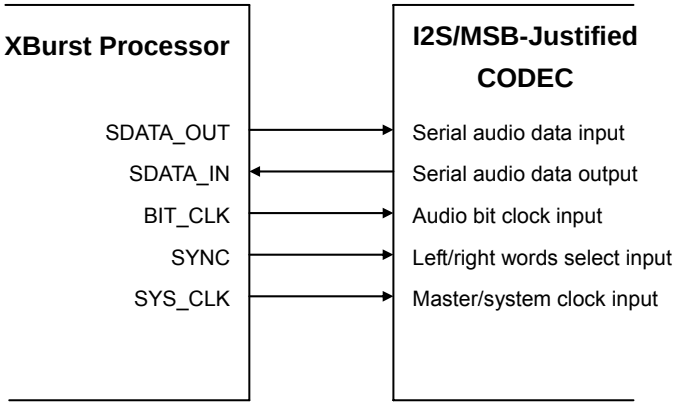


Figure 17-4 Interface to an External Slave Mode I2S/MSB-Justified CODEC Diagram

Please refer to the related CODEC specification for AIC Interface to the Internal CODEC Diagram.

17.1.4 Signal Descriptions

There are all 5 pins used to connect between AIC and an external audio CODEC device. If an internal CODEC is used, these pins are not needed. Please refer to Chip Spec. They are listed and described in Table 17-1.

Table 17-1 AIC Pins Description

Name	I/O	Description
RESET# **SYS_CLK**	**O**	**RESET#: AC-link format, active-low CODEC reset.** **SYS_CLK: I2S/MSB-Justified formats, supply system clock to CODEC.**
BIT_CLK	I I/O	**12.288 MHz bit-rate clock input for AC-link, and sample rate dependent bit-rate clock input/output for I2S/MSB-Jistified.**
SYNC	O	48-kHz frame indicator and synchronizer for AC-link format.
	I/O	**Indicates the left- or right-channel for I2S/MSB-Justified format.**
SDATA_OUT	**O**	**Serial audio output data to CODEC.**
SDATA_IN	**I**	**Serial audio input data from CODEC.**

The O_BIT_CLK and O_SYNC signals are not connected to any pin for only using by internal CODEC.

17.1.5 RESET# / SYS_CLK Pin

RESET# is AC97 active-low CODEC reset, which outputs to CODEC. The CODEC's registers are reset when this RESET# is asserted. This pin is useful only in AC-link format. If AIC is disabled, it retains the high.

SYS_CLK outputs the system clock to CODEC. This pin is useful only in I2S/MSB-justified format. It generates a frequency between approximately 2.048 MHz and 24.576 MHz by dividing down the PLL clock with a programmable divisor. This frequency can be 256, 384, 512 and etc. times of the audio sampling frequency. Or it can be set to a wanted frequency. If AIC is disabled, it retains the high.

17.1.6 BIT_CLK Pin

BIT_CLK is the serial data bit rate clock, at which AC97/I2S data moves between the CODEC and the processor. One bit of the serial data is transmitted or received each BIT_CLK period. It is fixed to 12.288 MHz in AC-link format, which inputs from the CODEC. In I2S and MSB-justified format it inputs from the CODEC in slave mode and outputs to CODEC in master mode. In the master mode, the clock is generated internally that is 64 times the sampling frequency. Table 17-7 lists the available sampling frequencies based on an internal clock source. If AIC is disabled, AICFR.AUSEL and AICFR.BCKD determine the direction. And it retains the low if it is output and the state is undefined if it is input.

17.1.7 SYNC Pin

In AC-link format, SYNC provides frame synchronization, fixed to 48kHz, by specifying beginning of an audio sample frame and outputs to CODEC. In I2S/MSB-Justified formats, SYNC is used to indicate left- or right-channel sample data and toggled in sample rate frequency. It outputs to CODEC in master mode and inputs from CODEC in slave mode. If AIC is disabled, AICFR.AUSEL and AICFR.BCKD determine the direction. And it retains the low if it is output and the state is undefined if it is input.

17.1.8 SDATA_OUT Pin

SDATA_OUT is AIC output data pin, which outputs serial audio data or data of AC97 CODEC register control to an external audio CODEC device. If AIC is disabled, it retains the low.

17.1.9 SDATA_IN Pin

SDATA_IN is AIC inputs data pin, which inputs serial audio data or data of AC97 CODEC register status from an external audio CODEC device. If AIC is disabled, its state is undefined.

17.2 Register Descriptions

AIC software interface includes 13 registers and 1 FIFO data port. They are mapped in IO memory address space so that program can access them to control the operation of AIC and the outside CODEC.

Table 17-2 AIC Registers Description

Name	Description	RW	Reset value	Address	Size
AICFR	AIC Configuration Register	RW	0x00007800	0x10020000	32
AICCR	AIC Common Control Register	RW	0x00000000	0x10020004	32
ACCR1	AIC AC-link Control Register 1	RW	0x00000000	0x10020008	32
ACCR2	AIC AC-link Control Register 2	RW	0x00000000	0x1002000C	32
I2SCR	AIC I2S/MSB-justified Control Register	RW	0x00000000	0x10020010	32
AICSR	AIC FIFO Status Register	RW	0x00000008	0x10020014	32
ACSR	AIC AC-link Status Register	RW	0x00000000	0x10020018	32
I2SSR	AIC I2S/MSB-justified Status Register	RW	0x00000000	0x1002001C	32
ACCAR	AIC AC97 CODEC Command Address Register	RW	0x00000000	0x10020020	32
ACCDR	AIC AC97 CODEC Command Data Register	RW	0x00000000	0x10020024	32
ACSAR	AIC AC97 CODEC Status Address Register	R	0x00000000	0x10020028	32
ACSDR	AIC AC97 CODEC Status Data Register	R	0x00000000	0x1002002C	32
I2SDIV	AIC I2S/MSB-justified Clock Divider Register	RW	0x00000003	0x10020030	32
AICDR	AIC FIFO Data Port Register	RW	0x????????	0x10020034	32
CKCFG	Clock Configure for the embedded CODEC to AIC	RW	0x00000000 0x00000002	0x100200A0	32
RGADW	Address, data in and write command for accessing to internal registers of embedded CODEC	RW	0x00000000	0x100200A4	32
RGDATA	The read out data and interrupt request status of Internal registers data in the embedded CODEC.	R	0x00000000	0x100200A8	32

1 AICFR is used to control FIFO threshold, AC-link or I2S/MSB-justified selection, AIC reset, master/slave selection, and AIC enable.

2 AICCR is used to control DMA mode, FIFO flush, interrupt enable, internal loop-back, play back and recording enable. It also controls sample size and signed/unsigned data transfer.

3 ACCR1 is used to reflect/control valid incoming/outgoing slots of AC97.

4 ACCR2 is used to control interrupt enable, output/input sample size, and alternative control of RESET#, SYNC and SDATA_OUT pins in AC-link.
5 I2SCR is used to control BIT_CLK stop, audio sample size, I2S or MSB-justified selection in I2S/MSB-justified.
6 AICSR is used to reflect FIFOs status.
7 ACSR is used to reflect the status of the connected external CODEC in AC-link.
8 I2SSR is used to reflect AIC status in I2S/MSB-justified.
9 ACCAR and ACCDR are used to hold address and data for AC-link CODEC register read/write.
10 ACSAR and ACSDR are used to receive AC-link CODEC registers address and data.
11 I2SDIV is used to set clock divider for BIT_CLK generating in I2S/MSB-justified format.
12 AICDR is act as data input/output port to/from transmit/receive FIFO when write/read.
13 CKCFG, RGADW and RGDATA are used to access internal CODEC, please refer to CODEC Spec.

17.2.1 AIC Configuration Register (AICFR)

AICFR contains bits to control FIFO threshold, AC-link or I2S/MSB-justified selection, AIC reset, master/slave selection, and AIC enable.

AICFR 0x10020000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																RFTH				TFTH				Reserved	LSMP	ICDC	AUSEL	RST	BCKD	SYNCD	ENB
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:16	Reserved	Writes to these bits have no effect and always read as 0.	R
15:12	RFTH	Receive FIFO threshold for interrupt or DMA request. The RFTH valid value is 0 ~ 15. This value represents a threshold value of (RFTH + 1) * 2. When the sample number in receive FIFO, indicated by AICSR.RFL, is great than or equal to the threshold value, AICSR.RFS is set. Larger RFTH value provides lower DMA/interrupt request frequency but have more risk to involve receive FIFO overflow. The optimum value is system dependent.	RW
11:8	TFTH	Transmit FIFO threshold for interrupt or DMA request. The TFTH valid value 0 ~ 15. This value represents a threshold value of TFTH * 2. When the sample number in transmit FIFO, indicated by AICSR.TFL, is less than or equal to the threshold value, AICSR.TFS is set. Smaller TFTH value provides lower DMA/interrupt request frequency but have more risk to involve transmit FIFO underflow. The optimum value is system dependent.	RW
7	Reserved	Writes to these bits have no effect and always read as 0.	R
6	LSMP	Select between play last sample or play ZERO sample in TX FIFO underflow. ZERO sample means sample value is zero. This bit is better be changed while audio replay is stopped. **LSMP** / **CODEC used** 0 / **Play ZERO sample when TX FIFO underflow.** 1 / **Play last sample when TX FIFO underflow.**	RW
5	ICDC	Internal CODEC used. Select between internal or external CODEC. **ICDC** / **CODEC used** 0 / **External CODEC.** 1 / **Internal CODEC.**	RW
4	AUSEL	Audio Unit Select. Select between AC-link and I2S/MSB-justified. Change this bit in case of BIT_CLK is stopped (I2SCR.STPBK = 1). **AUSEL** / **Selected** 0 / **Select AC-link format.**	RW

		1 **Select I2S/MSB-justified format.**	
3	RST	Reset AIC. Write 1 to this bit reset AIC registers and FIFOs except AICFR and I2SDIV register. Writing 0 to this bit has no effect and this bit is always reading 0.	W
2	BCKD	BIT_CLK Direction. This bit specifies input/output direction of BIT_CLK. It is only valid in I2S/MSB-justified format. When AC-link format is selected, BIT_CLK is always input and this bit is ignored. Change this bit in case of BIT_CLK is stopped (I2SCR.STPBK = 1). **BCKD** | **BIT_CLK Direction** 0 | BIT_CLK is input from an external source. 1 | BIT_CLK is generated internally and driven out to the CODEC.	RW
1	SYNCD	SYNC Direction. This bit specifies input/output direction of SYNC in I2S/MSB-justified format. When AC-link format is selected, SYNC is always output and this bit is ignored. Change this bit in case of BIT_CLK is stopped (I2SCR.STPBK = 1). **SYNCD** | **SYNC Direction** 0 | SYNC is input from an external source. 1 | SYNC is generated internally and driven out to the CODEC.	RW
0	ENB	Enable AIC function. This bit is used to enable or disable the AIC function. **ENB** | **Description** 0 | **Disable AIC Controller.** 1 | **Enable AIC Controller.**	RW

The BCKD bit (bit 2) and SYNCD bit (bit 1) configure the mode of I2S/MSB-justified interface. This is compliant with I2S specification.

BCKD	SYNCD	Description
0 (input)	**0 (input)**	**AIC roles the slave of I2S/MSB-justified interface.**
	1 (output)	**AIC roles the master with external serial clock source of I2S/MSB-justified interface.**
1 (output)	**0 (input)**	**Reserved.**
	1 (output)	**AIC roles the master of I2S/MSB-justified interface.**

17.2.2 AIC Common Control Register (AICCR)

AICCR contains bits to control DMA mode, FIFO flush, interrupt enable, internal loop-back, play back and recording enable. It also controls sample size and signed/unsigned data transfer.

AICCR 0x10020004

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved										OSS			ISS			RDMS	TDMS	Reserved	Reserved	M2S	ENDSW	ASVTSU	TFLUSH	RFLUSH	EROR	ETUR	ERFS	ETFS	ENLBF	ERPL	EREC
RST	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:22	Reserved	Writes to these bits have no effect and always read as 0.	R
21:19	OSS	Output Sample Size. These bits reflect output sample data size from memory or register. The data sizes supported are: 8, 16, 18, 20 and 24 bits. The sample data is LSB-justified in memory/register. **OSS** / **Sample Size** 0x0 / 8 bit. 0x1 / 16 bit. 0x2 / 18 bit. 0x3 / 20 bit. 0x4 / 24 bit. 0x5~0x7 / Reserved.	RW
18:16	ISS	Input Sample Size. These bits reflect input sample data size to memory or register. The data sizes supported are: 8, 16, 18, 20 and 24 bits. The sample data is LSB-justified in memory/register. **ISS** / **Sample Size** 0x0 / 8 bit. 0x1 / 16 bit. 0x2 / 18 bit. 0x3 / 20 bit. 0x4 / 24 bit. 0x5~0x7 / Reserved.	RW
15	RDMS	Receive DMA enable. This bit is used to enable or disable the DMA during receiving audio data. **RDMS** / **Receive DMA** 0 / **Disabled.** 1 / **Enabled.**	RW
14	TDMS	Transmit DMA enable. This bit is used to enable or disable the DMA during transmit audio data. **TDMS** / **Transmit DMA**	RW

			0	**Disabled.**
			1	**Enabled.**
13:12	Reserved	Writes to these bits have no effect and always read as 0.		R
11	M2S	Mono To Stereo. This bit control whether to do mono to stereo sample expansion in play back. When this bit is set, every outgoing sample data in the steam plays in both left and right channels. This bit should only be set in 2 channels configuration. It takes effective immediately when the bit is changed. Change this before replay started.		RW
		M2S	**Description**	
		0	**No mono to stereo expansion**	
		1	**Do mono to stereo expansion**	
10	ENDSW	Endian Switch. This bit control endian change on outgoing 16-bits size audio sample by swapping high and low bytes in the sample data.		RW
		ENDSW	**Description**	
		0	**No change on outgoing sample data.**	
		1	**Swap high and low byte for outgoing 16-bits size sample data.**	
9	ASVTSU	Audio Sample Value Transfer between Signed and Unsigned data format. This bit is used to control the signed ←→unsigned data transfer. If it is 1, the incoming and outgoing audio sample data will be transferred by toggle its most significant bit.		RW
		ASVTSU	**Description**	
		0	**No audio sample value signed ←→unsigned transfer.**	
		1	**Do audio sample value signed ←→unsigned transfer.**	
8	TFLUSH	Transmit FIFO Flush. Write 1 to this bit flush transmit FIFOs to empty. Writing 0 to this bit has no effect and this bit is always reading 0.		W
7	RFLUSH	Receive FIFO Flush. Write 1 to this bit flush receive FIFOs to empty. Writing 0 to this bit has no effect and this bit is always reading 0.		W
6	EROR	Enable ROR Interrupt. This bit is used to control the ROR interrupt enable or disable.		RW
		EROR	**ROR Interrupt**	
		0	**Disabled.**	
		1	**Enabled.**	
5	ETUR	Enable TUR Interrupt. This bit is used to control the TUR interrupt enable or disable.		RW
		ETUR	**TUR Interrupt**	
		0	**Disabled.**	

			1	**Enabled.**	
4	ERFS	Enable RFS Interrupt. This bit is used to control the RFS interrupt enable or disable.			RW
			ERFS	**RFS Interrupt**	
			0	**Disabled.**	
			1	**Enabled.**	
3	ETFS	Enable TFS Interrupt. This bit is used to control the TFS interrupt enable or disable.			RW
			ETFS	**TFS Interrupt**	
			0	**Disabled.**	
			1	**Enabled.**	
2	ENLBF	Enable AIC Loop Back Function. This bit is used to enable or disable the internal loop back function of AIC, which is used for test only. When the AIC loop back function is enabled, normal audio replay/record functions are disabled.			RW
			ENLBF	**Description**	
			0	AIC Loop Back Function is Disabled.	
			1	AIC Loop Back Function is Enabled.	
1	ERPL	Enable Playing Back function. This bit is used to disable or enable the audio sample data transmitting.			RW
			ERPL	**Description**	
			0	**AIC Playing Back Function is Disabled.**	
			1	**AIC Playing Back Function is Enabled.**	
0	EREC	Enable Recording Function. This bit is used to disable or enable the audio sample data receiving.			RW
			EREC	**Description**	
			0	**AIC Recording Function is Disabled.**	
			1	**AIC Recording Function is Enabled.**	

17.2.3 AIC AC-link Control Register 1 (ACCR1)

ACCR1 contains bits to reflect/control valid incoming/outgoing slots of AC97. It is used only in AC-link format.

ACCR1 **0x10020008**

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

Bits	Name	Description	RW
31:26	Reserved	Writes to these bits have no effect and always read as 0.	R
25:16	RS	Receive Valid Slots. These bits are used to indicate which incoming slots are valid. Slot 3 is mapped to bit 16 or RS[0], slot 4 to bit 17 or RS[1] and so on. When write to this field, a bit 1 means we expect a PCM data in the corresponding slot, a bit 0 means the corresponding slot PCM data will be discarded. When read from this field, a bit 1 means we receive an expected valid PCM data in the corresponding slot. This field should be written before record started.	RW

RS[n] Value	Description
0	**Slot n+3 is invalid.**
1	**Slot n+3 has valid PCM data.**

15:10	Reserved	Writes to these bits have no effect and always read as 0.	R
9:0	XS	Transmit Valid Slots. These bits making up slots map to the valid bits in the AC'97 tag (slot 0 on SDATA_OUT) and indicate which outgoing slots have valid PCM data. Bit 0 or XS[0] maps to slot 3, bit 1 or XS[1] to slot 4 and so on. Setting the corresponding bit indicates to AIC to take an audio sample from transmit FIFO to fill the respective slot. And it indicates to the CODEC that valid PCM data will be in the respective slot. The number of valid bits will designate how many words will be pulled out of the FIFO per audio frame. This field should be written before record and replay started.	RW

XS[n] Value	Description
0	**Slot n+3 is invalid.**
1	**Slot n+3 has valid PCM data.**

17.2.4 AIC AC-link Control Register 2 (ACCR2)

ACCR2 contains bits to control interrupt enable, output/input sample size, and alternative control of RESET#, SYNC and SDATA_OUT pins in AC-link. It is valid only in AC-link format.

ACCR2 0x1002000C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved													ERSTO	ESADR	ECADT	Reserved												SO	SR	SS	SA
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:19	Reserved	Writes to these bits have no effect and always read as 0.	R
18	ERSTO	Enable RSTO Interrupt. This bit is used to control the RSTO interrupt enable or disable. **ERSTO** / **RSTO Interrupt** 0 / **Disabled.** 1 / **Enabled.**	RW
17	ESADR	Enable SADR Interrupt. This bit is used to control the SADR interrupt enable or disable. **ESADR** / **SADR Interrupt** 0 / **Disabled.** 1 / **Enabled.**	RW
16	ECADT	Enable CADT Interrupt. This bit is used to control the CADT interrupt enable or disable. **ECADT** / **CADT Interrupt** 0 / **Disabled.** 1 / **Enabled.**	RW
15:4	Reserved	Writes to these bits have no effect and always read as 0.	R
3	SO	SDATA_OUT output value. When SA is 1, this bit controls SDATA_OUT pin voltage level, 0 for low, 1 for high; otherwise, it is ignored.	RW
2	SR	RESET# pin level. When AC-link is selected, this bit is used to drive the RESET# pin. **SR** / **RESET# Pin Voltage Level** 0 / High. 1 / Low.	RW
1	SS	SYNC value. When this bit is read, it returns the actual value of SYNC. When SA is 1, write value controls SYNC pin value. When SA is 0, write to it is ignored.	RW
0	SA	SYNC and SDATA_OUT Alternation. This bit is used to determine the driven signal of SYNC and SDATA_OUT. When SA is 0, SYNC and SDATA_OUT being driven AIC function logic; otherwise, SYNC is	RW

controlled by the SS and SDATA_OUT is controlled by the SO. The true table of SYNC is described in following.

SA	SS	Description
0	0	When read, indicated SYNC is 0.
		When write, not effect.
	1	When read, indicated SYNC is 1.
		When write, not effect.
1	0	When read, indicated SYNC is 0.
		When write, SYNC is driven to 0.
	1	When read, indicated SYNC is 1.
		When write, SYNC is driven to 1.

17.2.5 AIC I2S/MSB-justified Control Register (I2SCR)

I2SCR contains bits to control BIT_CLK stop, audio sample size, I2S or MSB-justified selection in I2S/MSB-justified. It is valid only in I2S/MSB-justified format.

I2SCR 0x10020010

Bit	31:13	12	11:5	4	3:1	0
	Reserved	STPBK	Reserved	ESCLK	Reserved	AMSL
RST	0	0	0	0	0	0

Bits	Name	Description	RW
31:13	Reserved	Writes to these bits have no effect and always read as 0.	R
12	STPBK	Stop BIT_CLK. It is used to stop the BIT_CLK in I2S/MSB-justified format. When AC-link is selected, all of its operations are ignored. **STPBK** / **Description** 0 / **BIT_CLK is not stopped.** 1 / **BIT_CLK is stopped.** Please set this bit to 1 to stop BIT_CLK when change AICFR.AUSEL and AICFR.BCKD.	RW
11:5	Reserved	Writes to these bits have no effect and always read as 0.	R
4	ESCLK	Enable SYSCLK output. When this bit is 1, the SYSCLK outputs to chip outside is enabled. Else, the clock is disabled.	RW
0	AMSL	Specify Alternate Mode (I2S or MSB-Justified) Operation. **AMSL** / **Description** 0 / Select I2S Operation Mode. 1 / Select MSB-Justified Operation Mode.	RW

17.2.6 AIC Controller FIFO Status Register (AICSR)

AICSR contains bits to reflect FIFOs status. Most of the bits are read-only except two, which can be written a 0.

AICSR **0x10020014**

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

Bits	Name	Description	RW
31:30	Reserved	Writes to these bits have no effect and always read as 0.	R
29:24	RFL	Receive FIFO Level. The bits indicate the amount of valid PCM data in Receive FIFO. **RFL Value** / **Description** 0x00 ~ 0x20 — RFL valid PCM data in receive FIFO. 0x21 ~ 0x3F — Reserved.	R
23:14	Reserved	Writes to these bits have no effect and always read as 0.	R
13:8	TFL	Transmit FIFO Level. The bits indicate the amount of valid PCM data in Transmit FIFO. **TFL Value** / **Description** 0x00 ~ 0x20 — TFL valid PCM data in transmit FIFO. 0x21 ~ 0x3F — Reserved.	R
7	Reserved	Writes to these bits have no effect and always read as 0.	R
6	ROR	Receive FIFO Over Run. This bit indicates that receive FIFO has or has not experienced an overrun. **ROR** / **Description** 0 — When read, indicates over-run has not been found. 0 — When write, clear itself. 1 — When read, indicates data has even been written to full receive FIFO. 1 — When write, not effects.	RW
5	TUR	Transmit FIFO Under Run. This bit indicates that transmit FIFO has or has not experienced an under-run. **TUR** / **Description** 0 — When read, indicates under-run has not been found. 0 — When write, clear itself. 1 — When read, indicates data has even been read from empty transmit FIFO. 1 — When write, not effects.	RW
4	RFS	Receive FIFO Service Request. This bit indicates that receive FIFO level	R

<table>
<tr><td></td><td></td><td>is or not below receive FIFO threshold, which is controlled by AICFR.RFTH. When RFS is 1, it may trigger interrupt or DMA request depends on the interrupt enable and DMA setting.
<table><tr><th>RFS</th><th>Description</th></tr><tr><td>0</td><td>Receive FIFO level below RFL threshold.</td></tr><tr><td>1</td><td>Receive FIFO level at or above RFL threshold.</td></tr></table></td><td></td></tr>
<tr><td>3</td><td>TFS</td><td>Transmit FIFO Service Request. This bit indicates that transmit FIFO level is below Transmit FIFO threshold, which is controlled by AICFR.TFTH. When TFS is 1, it may trigger interrupt or DMA request depends on the interrupt enable and DMA setting.
<table><tr><th>TFS</th><th>Description</th></tr><tr><td>0</td><td>Transmit FIFO level exceeds TFL threshold.</td></tr><tr><td>1</td><td>Transmit FIFO level at or below TFL threshold.</td></tr></table></td><td>R</td></tr>
<tr><td>2:0</td><td>Reserved</td><td>Writes to these bits have no effect and always read as 0.</td><td>R</td></tr>
</table>

17.2.7 AIC AC-link Status Register (ACSR)

ACSR contains bits to reflect the status of the connected external CODEC in AC-link format. Bits in this register are read-only in general, except some of them can be written a 0.

ACSR **0x10020018**

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

Bits	Name	Description	RW
31:22	Reserved	Writes to these bits have no effect and always read as 0.	R
21	SLTERR	Hardware detects a Slot Error. This bit indicates an error in SLOTREQ bits on incoming data from external CODEC is detected. The error can be: (1) find 1 in a SLOTREQ bit, which corresponding to an inactive slot; (2) all active slots should be request in the same time by SLOTREQ, but an exception is found. All errors are accumulated to ACSR.SLTERR by hardware until software clears it. Software writes 0 clear this bit and write 1 has no effect.	RW
20	CRDY	External CODEC Ready. This bit is derived from the CODEC Ready bit of Slot 0 in SDATA_IN, and it indicates the external AC97 CODEC is ready or not. **CRDY** / **Description** 0 / CODEC is not ready. 1 / CODEC is ready.	R
19	CLPM	External CODEC Low Power Mode. This bit indicates the external CODEC is switched to low power mode or BIT_CLK is active from CODEC after wake up. **CLPM** / **Description** 0 / BIT_CLK is active. 1 / CODEC is switched to low power mode.	R
18	RSTO	External CODEC Registers Read Status Time Out. This bit indicates that the read status time out is detected or not. It is set to 1 if the data not return in 4 frames after a CODEC registers read command issued. **RSTO** / **Description** 0 / When read, indicates time out has not occurred. 1 / When read, indicates read status time out found. Write 0 clear this bit and write 1 is ignored. When RSTO is 1, it may trigger an interrupt depends on the interrupt enable setting.	RW
17	SADR	External CODEC Registers Status Address and Data Received. This bit indicates that address and data of an external AC '97 CODEC register	RW

<table>
<tr><td></td><td></td><td>has or has not been received.
<table><tr><th>SADR</th><th>Description</th></tr><tr><td>0</td><td>When read, indicates no register address/data received.</td></tr><tr><td>1</td><td>When read, indicates address/data received.</td></tr></table>Write 0 clear this bit and write 1 is ignored. When SADR is 1, it may trigger an interrupt depends on the interrupt enable setting.</td><td></td></tr>
<tr><td>16</td><td>CADT</td><td>Command Address and Data Transmitted. This bit indicates that a CODEC register reading/writing command transmission has completed or not.
<table><tr><th>CADT</th><th>Description</th></tr><tr><td>0</td><td>When read, indicates the command has not done.</td></tr><tr><td>1</td><td>When read, indicates the command has done.</td></tr></table>Write 0 clear this bit and write 1 is ignored. When CADT is 1, it may trigger an interrupt depends on the interrupt enable setting.</td><td>RW</td></tr>
<tr><td>15:0</td><td>Reserved</td><td>Writes to these bits have no effect and always read as 0.</td><td>R</td></tr>
</table>

17.2.8 AIC I2S/MSB-justified Status Register (I2SSR)

I2SSR is used to reflect AIC status in I2S/MSB-justified. It is a read-only register.

I2SSR **0x1002001c**

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

Bits	Name	Description	RW
31:3	Reserved	Writes to these bits have no effect and always read as 0.	R
2	BSY	AIC busy in I2S/MSB-justified format. **BSY** / **Description** 0 — AIC controller is idle or disabled. 1 — AIC controller currently is transmitting or receiving a frame.	R
1:0	Reserved	Writes to these bits have no effect and always read as 0.	R

17.2.9 AIC AC97 CODEC Command Address & Data Register (ACCAR, ACCDR)

ACCAR and ACCDR are used to hold register address and data for external AC-link CODEC register read/write operation through SDATA_OUT. The format of ACCAR.CAR and ACCDR.CDR is compliant with AC'97 Component Specification 2.3 where ACCAR.CAR[19] of "1" specifies CODEC register read operation, of "0" specifies CODEC register write operation. The write access to ACCAR and ACCDR signals AIC to issue this operation. Please reference to 0 for software flow. These registers are valid only in AC-link. It is ignored in I2S/MSB-justified format.

ACCAR **0x10020020**

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

Bits	Name	Description	RW
31:20	Reserved	Writes to these bits have no effect and always read as 0.	R
19:0	CAR	Command Address Register. This is used to hold 20-bit AC '97 CODEC register address transmitted in SDATA_OUT slot 1. After this field is write, it should not be write again until the operation is finished.	RW

ACCDR **0x10020024**

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

Bits	Name	Description	RW
31:20	Reserved	Writes to these bits have no effect and always read as 0.	R
19:0	CDR	Command Data Register. This is used to hold 20-bit AC'97 CODEC register data transmitted in SDATA_OUT slot 2. After this field is write, it should not be write again until the operation is finished.	RW

17.2.10 AIC AC97 CODEC Status Address & Data Register (ACSAR, ACSDR)

ACSAR and ACSDR are used to receive the external AC-link CODEC registers address and data from SDATA_IN. When AIC receives CODEC register status from SDATA_IN, it set ACSR.SADR bit and put the address and data to ACSAR.SAR and ACSDR.SDR. Please reference to 0 for software flow. These registers are valid only in AC-link format and are ignored in I2S/MSB-justified format.

ACSAR **0x10020028**

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

Bits	Name	Description	RW
31:20	Reserved	Writes to these bits have no effect and always read as 0.	R
19:0	SAR	CODEC Status Address Register. This is used to receive 20-bit AC ’97 CODEC status address from SDATA_IN slot 1. Which reflect the register index for which data is being returned. The write operation is ignored.	R

ACSDR **0x1002002C**

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

Bits	Name	Description	RW
31:20	Reserved	Writes to these bits have no effect and always read as 0.	R
19:0	SDR	CODEC Status Data Register. This is used to receive 20-bit AC ’97 CODEC status data from SDATA_IN slot 2. The register data of external CODEC is returned. The write operation is ignored.	R

17.2.11 AIC I2S/MSB-justified Clock Divider Register (I2SDIV)

I2SDIV is used to set clock divider to generated BIT_CLK from SYS_CLK in I2S/MSB-justified format.

I2SDIV **0x10020030**

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

Bits	Name	Description	RW
31:4	Reserved	Writes to these bits have no effect and always read as 0.	R
3:0	DV	Audio BIT_CLK clock divider value minus 1. I2SDIV.DV is used to control the generating of BIT_CLK from dividing SYS_CLK. The dividing value should be even and I2SDIV.DV should be set to the dividing value minus 1. So I2SDIV.DV bit0 is fixed to 1. BIT_CLK frequency is fixed to 64 f_S in AIC, where f_S is the audio sample frequency. I2SDIV.DV depends on SYS_CLK frequency f_{SYS_CLK}, which is selected according to external CODEC's requirement and internal PLL frequency. Please reference to 0 "Serial Audio Clocks and Sampling Frequencies" for further description.	RW

17.2.12 AIC FIFO Data Port Register (AICDR)

AICDR is act as data input port to transmit FIFO when write and data output port from receive FIFO when read, one audio sample every time. The FIFO width is 24 bits. Audio sample with size N that is less than 24 is located in LSB N-bits. The sample size is specified by ACCR2.OASS and ACCR2.IASS in AC-link, and by I2SCR.WL in I2S/MSB-justified. The sample order is specified by ACCR1.XS and ACCR1.RS in AC-link. In I2S/MSB-justified, the left channel sample is prior to the right channel sample.

Care should be taken to monitor the status register to insure that there is room for data in the FIFO when executing a program read or write transaction. This is taken care automatically in DMA.

AICDR **0x10020034**

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

Bits	Name	Description	RW
31:24	Reserved	Writes to these bits have no effect and always read as 0.	R
23:0	DATA	FIFO port. When write to it, data is push to the transmit FIFO. When read from it, data is pop from the receiving FIFO.	RW

17.3 Serial Interface Protocol

17.3.1 AC-link serial data format

Following figures are AC-link serial data format. Audio data is MSB adjusted, regardless of 8, 16, 18, 20, 24 bits sample size. When a 24-bits sample is transmitted, the LSB 4-bits are truncated. When try to record 24-bits sample, 4-bits of 0 are appended in LSB. Please reference to “AC ’97 Component Specification Revision 2.3, 2002”, provided by Intel Corporation, for details of AC ’97 architecture and AC-link specification.

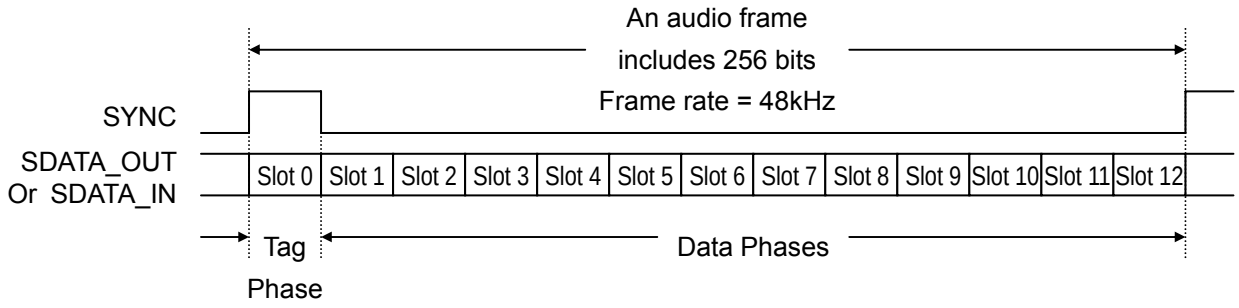


Figure 17-5 AC-link audio frame format

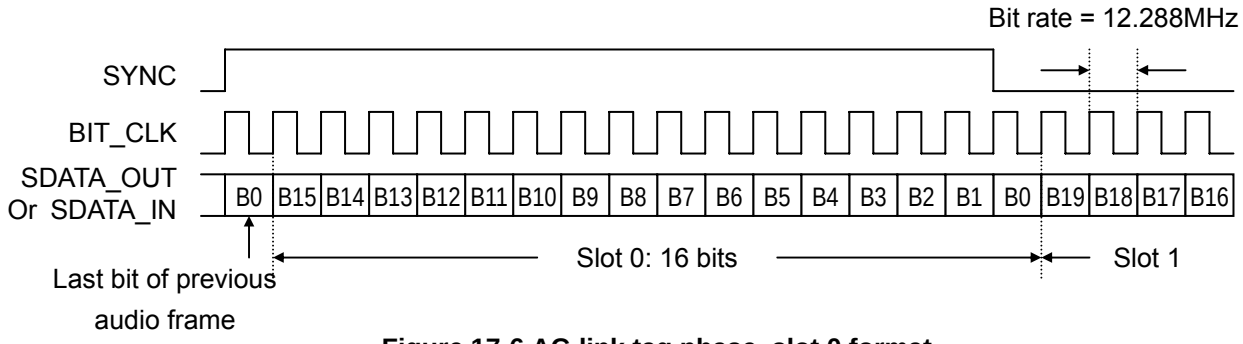


Figure 17-6 AC-link tag phase, slot 0 format

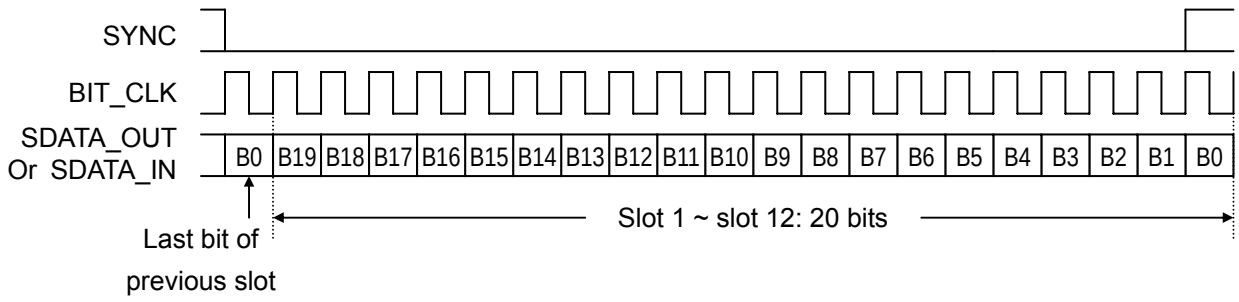


Figure 17-7 AC-link data phases, slot 1 ~ slot 12 format

17.3.2 I2S and MSB-justified serial audio format

Normal I2S and MSB-justified are similar protocols for digitized stereo audio transmitted over a serial path.

The BIT_CLK supplies the serial audio bit rate, the basis for the external CODEC bit-sampling logic. Its frequency is 64 times the audio sampling frequency. Divided by 64, the resulting 8 kHz to 48 kHz or even higher signal signifies timing for left and right serial data samples passing on the serial data paths. This left/right signal is sent to the CODEC on the SYNC pin. Each phase of the left/right signal is accompanied by one serial audio data sample on the data pins SDATA_IN and SDATA_OUT.

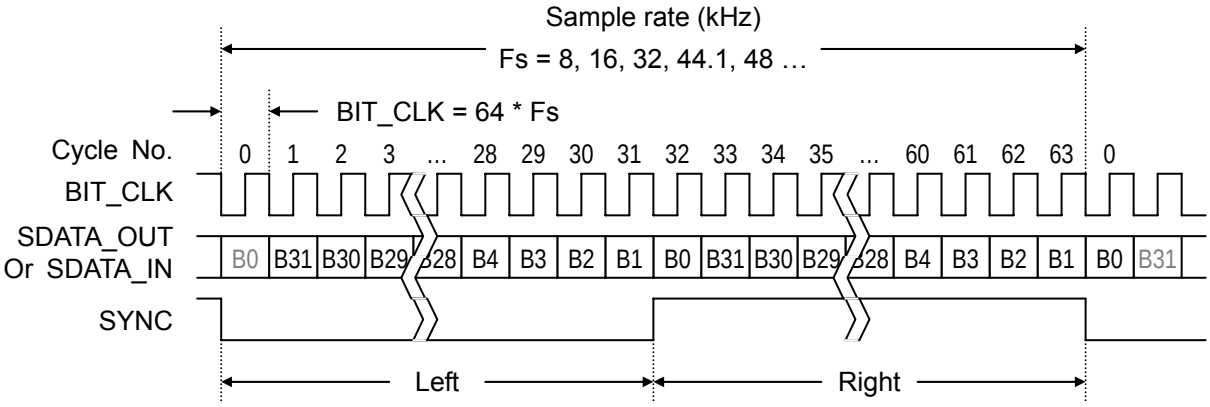


Figure 17-8 I2S data format

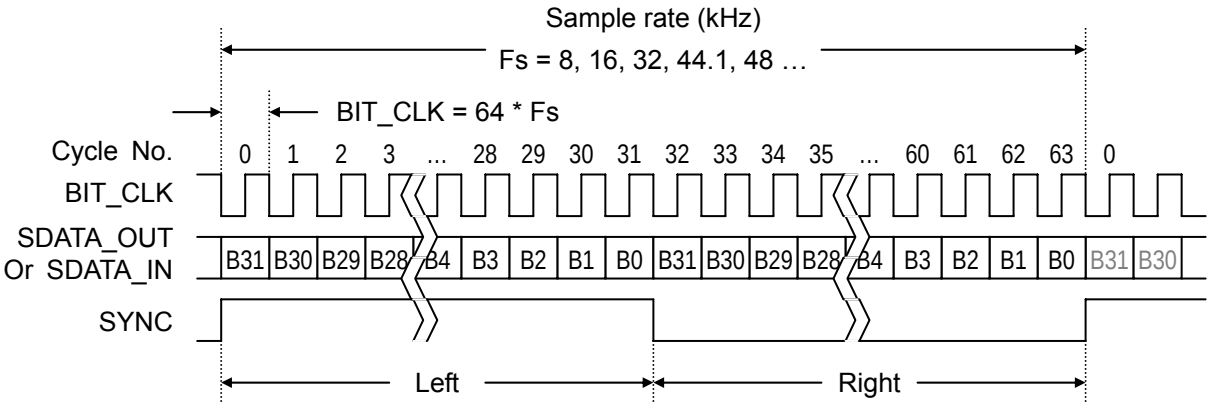


Figure 17-9 MSB-justified data format

Figure 17-8 and Figure 17-9 provide timing diagrams that show formats for the normal I2S and MSB-justified modes of operations. Data is sampled on the rising edge of the BIT_CLK and data is sent out on the falling edge of the BIT_CLK.

Data is transmitted and received in frames of 64 BIT_CLK cycles. Each frame consists of a left sample

and a right sample. Each sample holds 8, 16, 18, 20 or 24 bits of valid data. The LSB other bits of each sample is padded with zeroes.

In the normal I2S mode, the SYNC is low for the left sample and high for the right sample. Also, the MSB of each data sample lags behind the SYNC edges by one BIT_CLK cycle.
In the MSB-justified mode, the SYNC is high for the left sample and low for the right sample. Also, the MSB of each data sample is aligned with the SYNC edges.

When use with the internal CODEC, the BIT_CLK and SYNC signals also with O_BIT_CLK and O_SYNC signals are provided by the internal CODEC from the SYSCLK, which is enabled by I2SCR.ESCLK and configured to 12MHz clock using CPM.

17.3.3 Audio sample data placement in SDATA_IN/SDATA_OUT

The placement of audio sample in incoming/outgoing serial data stream for all formats support in AIC is MSB (Most Significant Bit) justified. Suppose n bit sample composed by

S_{n-1}	S_{n-2}	…	S_2	S_1	S_0

Table 17-3 described the how sample data bits are transferred.

Table 17-3 Sample data bit relate to SDATA_IN/SDATA_OUT bit

AC-link Format						I2S/MSB-Justified Format					
SDATA IN/OUT	Audio Sample Size (bit)					SDATA IN/OUT					
	8	16	18	20	24		8	16	18	20	24
B19	S7	S15	S17	S19	S23	B31	S7	S15	S17	S19	S23
B18	S6	S14	S16	S18	S22	B30	S6	S14	S16	S18	S22
B17	S5	S13	S15	S17	S21	B29	S5	S13	S15	S17	S21
B16	S4	S12	S14	S16	S20	B28	S4	S12	S14	S16	S20
B15	S3	S11	S13	S15	S19	B27	S3	S11	S13	S15	S19
B14	S2	S10	S12	S14	S18	B26	S2	S10	S12	S14	S18
B13	S1	S9	S11	S13	S17	B25	S1	S9	S11	S13	S17
B12	S0	S8	S10	S12	S16	B24	S0	S8	S10	S12	S16
B11	0	S7	S9	S11	S15	B23	0	S7	S9	S11	S15
B10	0	S6	S8	S10	S14	B22	0	S6	S8	S10	S14
B9	0	S5	S7	S9	S13	B21	0	S5	S7	S9	S13
B8	0	S4	S6	S8	S12	B20	0	S4	S6	S8	S12
B7	0	S3	S5	S7	S11	B19	0	S3	S5	S7	S11
B6	0	S2	S4	S6	S10	B18	0	S2	S4	S6	S10
B5	0	S1	S3	S5	S9	B17	0	S1	S3	S5	S9
B4	0	S0	S2	S4	S8	B16	0	S0	S2	S4	S8
B3	0	0	S1	S3	S7	B15	0	0	S1	S3	S7
B2	0	0	S0	S2	S6	B14	0	0	S0	S2	S6
B1	0	0	0	S1	S5	B13	0	0	0	S1	S5
B0	0	0	0	S0	S4	B12	0	0	0	S0	S4
						B11	0	0	0	0	S3
						B10	0	0	0	0	S2
						B9	0	0	0	0	S1
						B8	0	0	0	0	S0
						B7~ B0	0	0	0	0	0

17.4 Operation

The AIC can be accessed either by the processor using programmed I/O instructions or by the DMA controller. The processor uses programmed I/O instructions to access the AIC and can access the following types of data.

The AIC memory mapped registers data—All registers are 32 bits wide and are aligned to word boundaries.
AIC controller FIFO data—An entry is placed into the transmit FIFO by writing to the I2S controller's Serial Audio Data register (AICDR). Writing to AICDR updates a transmit FIFO entry. Reading AICDR flushes out a receive FIFO entry.
The external CODEC registers for I2S CODEC—CODEC registers can be accessed through the L3 bus. The L3 bus operation is emulated by software controlling three GPIO pins.
The external CODEC registers for AC97 CODEC—An AC97 audio CODEC can contain up to sixty-four 16-bit registers. A CODEC uses a 16-bit address boundary for registers. The AIC supplies access to the CODEC registers through several registers.
The internal CODEC registers can be accessed via memory mapped registers in the CODEC.

The DMA controller can only access the FIFOs. Accesses are made through the data registers, as explained in the previous paragraph. The DMA controller responds to the following DMA requests made by the I2S controller:

The transmit FIFO request is based on the transmit trigger-threshold (AICFR.TFTH) setting. See 0 for further details regarding AICFR.TFTH.
The receive FIFO request is based on the receive trigger-threshold (AICFR.RFTH) setting. See 0 for further details regarding AICFR.RFTH.

Before operation to AIC, you may need to set proper PIN function selection from GPIO using if the pin is shared with GPIO.

Please also reference to "AC '97 Component Specification Revision 2.3, 2002" when deal with AIC AC-link operations.

17.4.1 Initialization

At power-on or other hardware reset (WDT and etc), AIC is disabled. Software must initiate AIC and the internal or external CODEC after power-on or reset. If errors found in data transferring, or in other places, software must initial AIC and optional, the internal or external CODEC. Here is the initial flow.

1 Select internal or external CODEC (AICFR.ICDC).
2 If external CODEC is selected, select AC-link or I2S/MSB-Justified (AICFR.AUSEL). If internal CODEC is used, select I2S/MSB-Justified format (AICFR.AUSEL=1). If the resettlement without involving link format and architecture changing, this step can be skip.
3 If I2S/MSB-Justified is selected, select between I2S and MSB-Justified (I2SCR.AMSL), decide BIT_CLK direction (AICFR.BCKD) and SYNC direction (AICFR.SYNCD). If BIT_CLK is configured as output, BIT_CLK divider I2SDIV.DV must be set to what correspond with the values as shown in Table 17-7. And the clock selection and the divider between PLL clock out and AIC also must be set (CFCR.I2S and I2SCDR in CPM). If internal CODEC is used, select 12MHz clock input (via set proper value in CFCR.I2S and I2SCDR), I2S format (I2SCR.AMSL=0), input BIT_CLK (AICFR.BCKD=0), input SYNC (AICFR.SYNCD=0).
4 Enable AIC by write 1 to AICFR.ENB.
5 If it needs to reset AIC registers and flush FIFOs, write 1 to AICFR.RST. If it need only flush FIFOs, write 1 to AICCR.FLUSH. BIT_CLK must exist here and after.
6 In AC-link format, issue a warm or cold CODEC reset.
7 In AC-link format, configure AC ’97 CODEC via ACCAR and ACCDR registers. If the resettlement doesn’t involving AC’97 CODEC registers changing, this step can be skip.
8 In case of external CODEC with I2S/MSB-Justified format, configure I2S/MSB-justified CODEC via the control bus connected to the CODEC, for instance I2C or L3, depends on CODEC. In case of internal CODEC, configure CODEC via CODEC’s memory mapped registers. If the resettlement without involving I2S/MSB-justified CODEC or ADC/DAC function changing, this step can be skip.

17.4.2 AC '97 CODEC Power Down

AC '97 CODEC can be placed in a low power mode. When the CODEC's power-down register (26h), is programmed to the appropriate value, the CODEC will be put in a low power mode and both BIT_CLK and SDATA_IN will be brought to and held at a logic low voltage level.

Once powered down, re-activation of the AC-link via re-assertion of the SYNC signal must not occur for a minimum of four audio frame times following the frame in which the power down was triggered. When AC-link powers up it indicates readiness via the CODEC Ready bit (input slot 0, bit 15).

17.4.3 Cold and Warm AC '97 CODEC Reset

AC-link reset operations occur when the system is initially powered up, when resuming from a lower powered sleep state, and in response to critical subsystem failures that can only be recovered from with a reset.

17.4.3.1 Cold AC '97 CODEC Reset

A cold reset is achieved by asserting RESET# for the minimum specified time. By driving RESET# low, BIT_CLK, and SDATA_IN will be activated, or re-activated as the case may be, and all AC '97 CODEC registers will be initialized to their default power on reset values.

RESET# is an asynchronous AC '97 CODEC input.

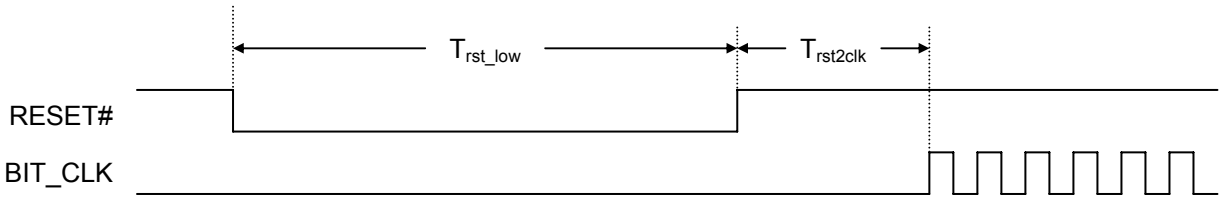


Figure 17-10 Cold AC '97 CODEC Reset Timing

Table 17-4 Cold AC '97 CODEC Reset Timing parameters

Parameter	Symbol	Min	Type	Max	Units
RESET# active low pulse width	T_{rst_low}	1.0	-	-	μs
RESET# inactive to BIT_CLK startup delay	$T_{rst2clk}$	162.8	-	-	ns

17.4.3.2 Warm AC '97 CODEC Reset

A warm AC'97 reset will re-activate the AC-link without altering the current AC'97 register values. Driving SYNC high for a minimum of 1 μs in the absence of BIT_CLK signals a warm reset.

Within normal audio frames SYNC is a synchronous AC '97 CODEC input. However, in the absence of BIT_CLK, SYNC is treated as an asynchronous input used in the generation of a warm reset to AC '97 CODEC.

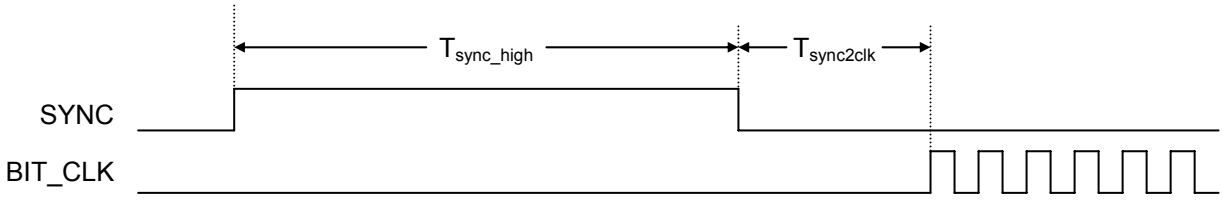


Figure 17-11 Warm AC '97 CODEC Reset Timing

Table 17-5 Warm AC '97 CODEC Reset Timing Parameters

Parameter	Symbol	Min	Type	Max	Units
SYNC active high pulse width	T_{sync_high}	1.0	-	-	Ms
SYNC inactive to BIT_CLK startup delay	$T_{sync2clk}$	162.8	-	-	Ns

17.4.4 External CODEC Registers Access Operation

The external audio CODEC can be configured/controlled by its internal registers. To access these registers, an I2S/MSB-justified CODEC usually employs L3 bus, SPI bus, I2C bus or other control bus. The L3 bus operation can be emulated by software by using 3 GPIO pins of the chip. For AC '97, "AC '97 Component Specification" defines the CODEC register access protocol. Several registers are provided in AIC to accomplish this task.

The ACCAR and ACCDR are used to send a register accessing request command to external AC'97 CODEC. The ACSAR and ACSDR are used to receive a register's content from external AC'97 CODEC. The register accessing request and the register's content returning is asynchronous.

The AC'97 CODEC register accessing request flow:

1 If ACSR.CADT is 0, wait for 25.4μs. If no previous accessing request, this step can be skip.
2 Clear ACSR.CADT.
3 If read access, write read-command and register address to ACCAR, if write access, write write-command and register address to ACCAR and write data to ACCDR. Any order of write ACCAR and ACCDR is OK.
4 Polling for ACSR.CADT changing to 1, which means the request has been send to CODEC via AC-link.

The AC'97 CODEC register content receiving flow by polling:

1 Polling for ACSR.SADR changing to 1.
2 Read the CODEC register's address from ACSAR and content from ACSDR.
3 Clear ACSR.SADR.

The AC'97 CODEC register content receiving flow by interrupt:

1 Before accessing request, clear ACSR.SADR and set ACCR2.ESADR.
2 Waiting for the interrupt. When the interrupt is found, check if ACSR.SADR is 1, if not, repeat this step again.
3 Read the CODEC register's address from ACSAR and content from ACSDR.
4 Clear ACSR.SADR.

17.4.5 Audio Replay

Outgoing audio sample data (from AIC to CODEC) is written to AIC transmit FIFO from processor via store instruction or from memory via DMA. AIC then takes the data from the FIFO, serializes it, and sends it over the serial wire SDATA_OUT to an external CODEC or over an internal wire to an internal CODEC.

The audio transmission is enabled automatically when the AIC is enabled by set AICFR.ENB. But all replay data is zero at this time except both of the following conditions are true:

1 AICCR.ERPL must be 1. If AICCR.ERPL is 0, value of zero is send to CODEC even if there are samples in transmit FIFO.
2 At least one audio sample data in the transmit FIFO. If the transmit FIFO is empty, value of zero or last sample depends on AICFR.LSMP, is send to CODEC even if AICCR.ERPL is 1.

Here is the audio replay flow:

1 Configure the CODEC as needed.
2 Configure sample size by AICCR.OSS.
3 Configure sample rate by clock dividers (for I2S/MSB-Justified format with BIT_CLK is provided internally) or by CODEC registers (for AC-link or BIT_CLK provided by external CODEC) or by accessing CODEC internal registers (for internal CODEC).
4 For AC-link, configure replay channels by ACCR1.XS.
5 Some other configurations: mono to stereo, endian switch, signed/unsigned data transfer, transmit FIFO configuration, play ZERO or last sample when TX FIFO under-run, and etc.
6 Write 1 to AICCR.ERPL.
  It is suggested that at least a frame of PCM data is pre-filled in the transmit FIFO to prevent FIFO under-run flag (AICSR.TUR).
  But when using internal CODEC, write first frame of PCM data to transmit FIFO till TX FIFO under-run (AICSR.TUR is set to 1), otherwise left/right channel may be switched.
7 Fill sample data to the transmit FIFO. Repeat this till finish all sample data. In this procedure, please control the FIFO to make sure no FIFO under-run and other errors happen. When the transmit FIFO under-run, noise or pause may be heard in the audio replay, AICSR.TUR is 1, and if AICCR.ETUR is 1, AIC issues an interrupt. Please reference to 0 for detail description on FIFO.
8 Waiting for AICSR.TFL change to 0. So that all samples in the transmit FIFO has been replayed, then we can have a clean start up next time.
9 Write 0 to AICCR.ERPL.

NOTE: Before replaying Open ADC BITCLK and close it to generating Record internal circuit reset when using internal CODEC.

17.4.6 Audio Record

Incoming audio sample data (from CODEC to AIC) is received from SDATA_IN (for an external CODEC) or an internal wire (for an internal CODEC) serially and converted to parallel word and stored in AIC receive FIFO. Then the data can be taken from the FIFO to processor via load instruction or to memory via DMA.

The audio recording is enabled automatically when the AIC is enabled by set AICFR.ENB. But all received data is discarded at this time except both of the following conditions are true:

1 AICCR.EREC must be 1. If AICCR.EREC is 0, the received data is discarded even if there are rooms in the receive FIFO.
2 At least one room left in the receive FIFO. If the receive FIFO is full, the received data is discarded even if AICCR.EREC is 1.

Here is the audio record flow:

1 Configure the CODEC as needed.
2 Configure sample size by AICCR.ISS.
3 Configure sample rate by clock dividers (for I2S/MSB-Justified format with BIT_CLK is provided internally) or by CODEC registers (for AC-link or BIT_CLK provided by external CODEC) or by CODEC memory mapped registers (for internal CODEC).
4 Some other configurations: signed/unsigned data transfer, receive FIFO configuration, and etc.
5 Write 1 to AICCR.EREC. Make sure there are rooms available in the receive FIFO before set AICCR.EREC. Usually, it should empty the receive FIFO by fetch data from it before set AICCR.EREC.
6 Take sample data form the receive FIFO. Repeat this till the audio finished. In this procedure, please control the FIFO to make sure no FIFO over-run and other errors happen. When the receive FIFO over-run, same recorded audio samples will be lost, AICSR.ROR is 1, and if AICCR.EROR is 1, AIC issues an interrupt. Please reference to 0 for detail description on FIFO. For AC-link, ACCR1.RS tells which channels are recorded.
   When using internal CODEC, the first data should be ignored.
7 Write 0 to AICCR.EREC.
8 Take sample data from the receive FIFO until AICSR.RFL change to 0. So that all samples in the receive FIFO has been taken away, then we can have a clean start up next time. When the receive FIFO is empty, read from it returns zero.

17.4.7 FIFOs operation

AIC has two FIFOs, one for transmit audio sample and one for receive. All AIC played/recorded audio sample data is taken from/send to transmit/receive FIFOs. The FIFOs are in 24 bits width and 32 entries depth, one entry for keep one audio sample regardless of the sample size. AICDR.DATA provides the access point for processor/DMA to write to transmit FIFO and read from receive FIFO. One time access to AICDR.DATA process one sample. The sample data should be put in LSB (Least Significant Bit) in memory or processor registers. For transmitting, bits exceed sample are discarded. For receiving, these bits are set to 0. Figure 17-12 illustrates the FIFOs access.

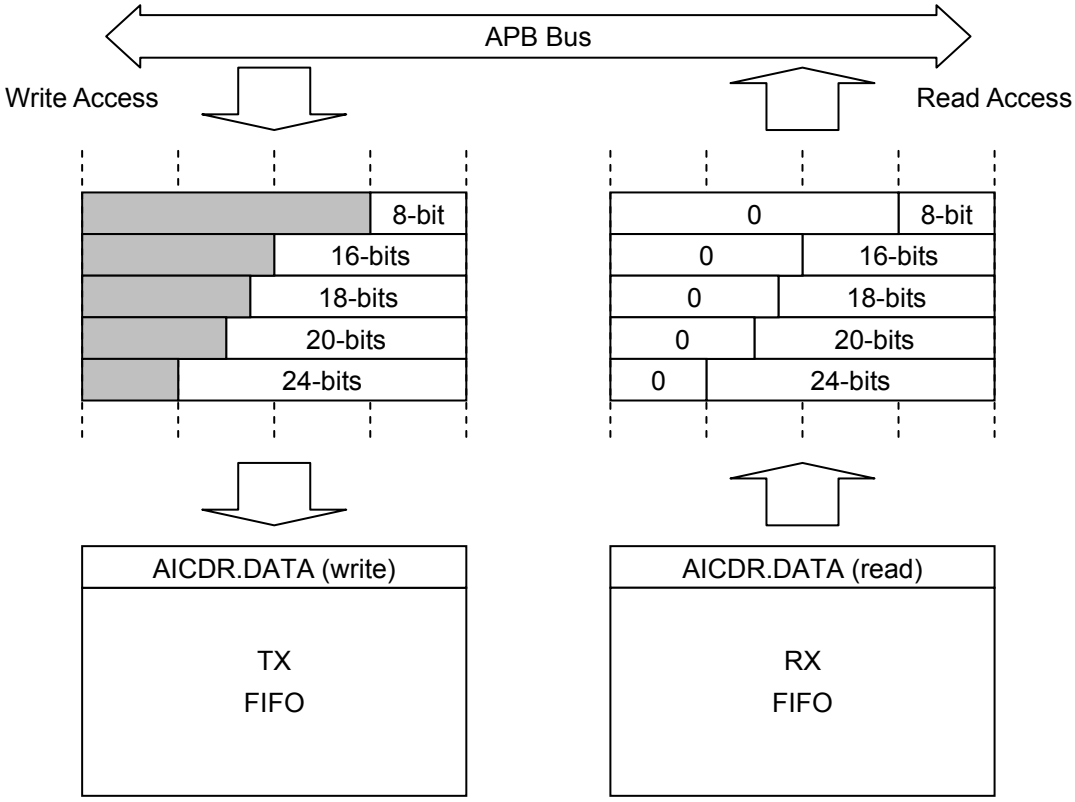


Figure 17-12 Transmitting/Receiving FIFO access via APB Bus

The software and bus initiator must guarantee the right sample placement at the bus.

In case of DMA bus initiator, one 24, 20, 18 bits audio sample must occupies one 32-bits word in memory, so 32-bits width DMA must be used. One 16 bits sample occupies one 16-bits half word in memory, so 16-bits width DMA must be used. One 8-bits sample occupies one byte in memory, and use 8-bits width DMA.

In case of processor bus initiator, any type of the audio sample must occupies one CPU general-purpose register at LSB, and read/write from/to AICDR.DATA with 32-bits load/store instruction. When process small sample size, 16-bits or 8-bits, software may need to do the data

pack/unpack.

The AICFR.TFTH and AICFR.RFTH are used to set the FIFO level thresholds, which are the trig levels of DMA request and/or FIFO service interrupt. The AICFR.TFTH and AICFR.RFTH should be set to proper values, too small or too big are not good. When it is too small, the DMA burst length or the number of sample can be processed by processor is too small, which harms the bus or processor efficiency. When it is too big, the bus or the interrupt latency left for under-run/over-run is too small, which may causes replay/record errors.

AICSR.TUR is set to 1 during transmit under-run conditions. If AICCR.ETUR is 1, this can trigger an interrupt. During transmit under-run conditions, zero or last sample is continuously sent out across the serial link. Transmit under-run can occur under the following conditions:

1 Valid transmit data is still available in memory, but the DMA controller/processor starves the transmit FIFO, as it is busy servicing other higher-priority tasks.
2 The DMA controller/processor has transferred all valid data from memory to the transmit FIFO.

AICSR.ROR is set to 1 during receive over-run conditions. If AICCR.EROR is 1, this can trigger an interrupt. During receive over-run conditions, data sent by the CODEC is lost and is not recorded.

When replay/record two channels data, the left channel is always the first data in FIFOs and in the serial link. If multiple channels in AC-link are used, the channel sample order is follows the slot order.

17.4.8 Data Flow Control

There are three approaches provided to control/synchronize the audio incoming/outgoing data flow.

17.4.8.1 Polling and Processor Access

AICSR.RFL and AICSR.TFL reflect how many samples exist in receiving and transmitting FIFOs. Through read these register fields, processor can detect when there are samples in receiving FIFO in audio record and then load them from the RX-FIFO, and when there are rooms in transmitting FIFO in audio replay and then store samples to the TX-FIFO.

Polling approach is in very low efficiency and is not recommended.

17.4.8.2 Interrupt and Processor Access

Set proper values to AICFR.TFTH and AICFR.RFTH, the FIFO interrupts trig thresholds. Set AICCR.ETFS and/or AICCR.ERFS to 1 to enable transmitting and/or receiving FIFO level trigger interrupts. When the interrupt found, it means there are rooms or samples in the TX or RX FIFO, and processor can store or load samples to or from the FIFO.

Interrupt approach is more efficient than polling approach.

17.4.8.3 DMA Access

Audio data is real time stream, though it is in low data bandwidth, usually less than 1.2Mbps. DMA approach is the most efficient and is the recommended approach.

To enable DMA operation, set AICCR.TDMS and AICCR.RDMS to 1 for transmit and receive respectively. It also needs to allocate two channels in DMA controller for data transmitting and receiving respectively. Please reference to the processor's DMA controller spec for the details.

The AICFR.TFTH and AICFR.RFTH are used to set the transmitting and receiving FIFO level thresholds, which determine the issuing of DMA request to DMA controller. To respond the request, DMAC initiator and controls the data movement between memory and TX/RX FIFO.

17.4.9 Serial Audio Clocks and Sampling Frequencies

For internal CODEC, CODEC module containing the audio CODEC circuit/logic and corresponding controlling registers. CODEC needs a 12MHz clock from CPM called SYS_CLK and provides I_BITCLK, O_BITCLK and I_SYNC, O_SYNC (left-right clock which is the sample rate as ADC or DAC) to AIC for outgoing and incoming audio respectively. These clocks change when change the sample rate in CODEC controlling registers. When using internal CODEC, must configure SYNC and BIT_CLK as input, more details refers to CODEC Spec.

For AC-link, the bit clock is input from chip external and is fixed to 12.288MHz. The sample frequency of 48kHz is supported in nature. Variable Sample Rate feature is supported in this AIC. If the CODEC supports this feature, sample rate other than 48kHz audio data can be replay directly. Otherwise, software has to do the rate transfer to replay other sample rate audio data. Double rate, 96kHz or even 88.2kHz audio is also supported with proper CODEC.

Following are for BIT_CLK/SYS_CLK configuration in I2S/MSB-Justified format with external CODEC.

The BIT_CLK is the rate at which audio data bits enter or leave the AIC. BIT_CLK can be supplied either by the CODEC or an internally PLL. If it is supplied internally, BIT_CLK is configured as output pins, and is supplied out to the CODEC. If BIT_CLK is supplied by the CODEC, then it is configured as an input pin. Register bit AICFR.BCKD is used to select BIT_CLK direction.

The audio sampling frequency is the frequency of the SYNC signal, which must be 1/64 of BIT_CLK, $f_{BIT\_CLK} = 64\ f_S$. But SYNC signal frequency is not fixed when using internal CODEC.

SYS_CLK is only for CODEC. It usually takes one of the two roles, as CODEC master clock input or as CODEC over-sampling clock input. If SYS_CLK roles as CODEC master clock input, it usually should be set to a fixed frequency according to CODEC requirement but independent to audio sample rate. In this case, usually there is a PLL in the CODEC and CODEC roles master mode. See Figure 17-3 for the interface diagram. This is the recommended AIC CODEC system configuration.

If SYS_CLK roles as CODEC over-sampling clock, its frequency is usually 4, 6, 8 or 12 times of BIT_CLK frequency, which are 256, 384, 512 and 768 times of audio sample rates. Table 17-6 lists the relation between sample rate, BIT_CLK and SYS_CLK frequencies.

Table 17-6 Audio Sampling rate, BIT_CLK and SYS_CLK frequencies

Sample Rate f_S (kHz)	BIT_CLK (MHz) $f_{BIT\_CLK} = 64\ f_S$	SYS_CLK (MHz)			
		256 f_S	384 f_S	512 f_S	768 f_S
48	3.072	12.288	18.432	24.576	36.864
44.1	2.8224	11.2896	16.9344	22.5792	33.8688
32	2.048	8.192	12.288	16.384	24.576
24	1.536	6.144	9.216	12.288	18.432
22.05	1.4112	5.6448	8.4672	11.2896	16.9344

16	1.024	4.096	6.144	8.192	12.288
11.025	0.7056	2.8224	4.2336	5.6448	8.4672
8	0.512	2.048	3.072	4.096	6.144

In this processor, SYS_CLK can be selected from EXCLK or generated by dividing the PLL output clock in a CPM divider controlled by I2SCDR. If BIT_CLK is chosen as an output, another divider in AIC is used to divide SYS_CLK for it.

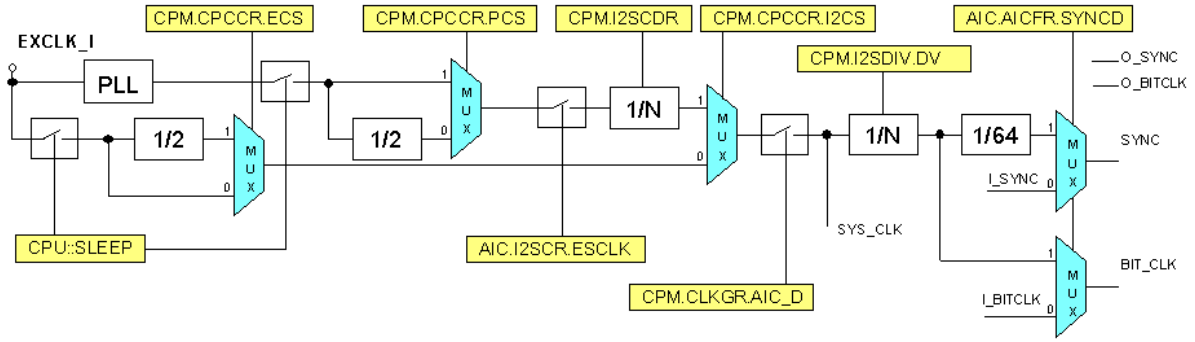


Figure 17-13 SYS_CLK, BIT_CLK and SYNC generation scheme

The setting of I2SDIV.DV is shown in Table 17-7.

Table 17-7 BIT_CLK divider setting

I2SDIV.DV	f_{SYS_CLK}	f_{BIT_CLK}	f_{SYS_CLK} / f_{BIT_CLK}
0x1	128 f_S	64 f_S	2
0x2	196 f_S	64 f_S	3
0x3	256 f_S	64 f_S	4
0x5	384 f_S	64 f_S	6
0x7	512 f_S	64 f_S	8
0xB	768 f_S	64 f_S	12

As we observe in Table 17-6, if SYS_CLK is taken as over-sampling clock by CODEC, the common multiple of all SYS_CLK frequencies is much bigger than the PLL output clock frequency. To generate all different SYS_CLK frequencies, one approach is change PLL frequency according to sample rate. This is not realistic, since frequently change PLL frequency during normal operation is not recommended.

Another approach is to found some approximate common multiples of all SYS_CLK frequencies according to the fact that there tolerance in audio sample rate. Take f_{SYS_CLK} = 256 f_S, Table 17-8 list most frequencies, which are less than 400MHz, with relatively small sample rate errors. It is suggested to set PLL frequency as close to the frequencies listed as possible, then use clock dividers to generate different SYS_CLK/BIT_CLK for different sample rate.

Table 17-8 Approximate common multiple of SYS_CLK for all sample rates

Approximate Common Frequency (MHz)	Max Error Caused in Audio Sample Rate (%)
123.53	0.53
147.11	0.24
170.68	0.79
235.5	0.87
247.06	0.53
270.64	0.11
280.56	0.73
294.22	0.24
305.14	0.67
317.79	0.53
329.57	0.66
341.35	0.79
347	0.85
353.13	0.90
358.79	0.69
370.59	0.53
382.96	0.54
394.17	0.24

Take PLL = 270.64 MHz as an example, Table 17-9 lists the divider settings for various sample rates.

Table 17-9 CPM/AIC clock divider setting for various sampling rate if PLL = 270.64MHz

Sample Rate (kHz)	I2SCDR	I2SDIV.DV	Sample Rate Error (%)
48	1	11	0.11
44.1	1	12	-0.11
32	0	33	0.11
24	1	22	0.11
22.05	1	24	-0.11
16	1	33	0.11
12	1	44	0.11
11.025	1	48	-0.11
8	1	66	0.11

For an EXCLK clock frequency, try to generate PLL frequencies as close to the frequencies listed in Table 17-8 as possible. Table 17-10 lists the PLL parameters and audio sample errors at different PLL frequencies for EXCLK at 12MHz.

Table 17-10 PLL parameters and audio sample errors for EXCLK=12MHz

PLL			Max Sample Rate Error
M	N	Freq (MHz)	
103	10	123.6	0.59%
49	4	147	0.31%
128	9	170.67	0.79%
157	8	235.5	0.87%
103	5	247.2	0.59%
65	3	260	0.82%
45	2	270	0.35%
203	9	270.67	0.12%
113	5	271.2	0.32%
187	8	280.5	0.75%
237	10	284.4	0.81%
49	2	294	0.31%
178	7	305.14	0.67%
53	2	318	0.60%
302	11	329.45	0.70%
256	9	341.33	0.79%
318	11	346.91	0.88%
206	7	353.14	0.90%
299	10	358.8	0.69%
247	8	370.5	0.55%
351	11	382.91	0.55%
230	7	394.29	0.27%

The BIT_CLK should be stopped temporary when change the divider settings, or when change BIT_CLK source (from internal or external), to prevent clock glitch. Register I2SCR.STPBK is provided to assist the task. When I2SCR.STPBK = 1, BIT_CLK is disabled no matter whether it is generated internally or inputted from the external source. The operation flow is described in following.

1 Stop all replay/record by clear AICCR.ERPL and AICCR.EREC.
2 Polling I2SSR.BSY till it is 0.
3 Stop the BIT_CLK by write 1 to I2SCR.STPBK.
4 Operations concerning BIT_CLK.
5 Resume the BIT_CLK by write 0 to I2SCR.STPBK.

17.4.10 Interrupts

The following status bits, if enabled, interrupt the processor:

- Receive FIFO Service (AICSR.RFS). It's also DMA Request.
- Transmit FIFO Service (AICSR.TFS). It's also DMA Request.
- Transmit Under-Run (AICSR.TUR).
- Receive Over-Run (AICSR.ROR).
- Command Address and Data Transmitted, AC-link only (ACSR.CADT).
- External CODEC Registers Status Address and Data Received, AC-link only (ACSR.SADR).
- External CODEC Registers Read Status Time Out, AC-link only (ACSR.RSTO).

For further details, see the corresponding register description sections.

18 SAR A/D Controller

18.1 Overview

The A/D embedded in this processor is a CMOS low-power dissipation 12bit SAR analog to digital converter. It operates with 3.3/1.8V power supply. Circuits needed by touch screen function and battery voltage measurement are also included.

The SAR A/D controller is dedicated to control A/D to work at three different modes: Touch Screen (measure pen position and pen down pressure), Battery (check the battery power), and SADCIN (external ADC input). Touch Screen can transfer the data to memory through the DMA or CPU. Battery and SADCIN can transfer the data to memory through CPU.

Features:

- 6 Channels
- Resolution: 12-bit
- Integral nonlinearity: ±0.5 LSB
- Differential nonlinearity: ±0.4 LSB
- Resolution/speed: up to12bit 187.5ksps
- Max Frequency: 8.0MHz
- Power-down current: 1uA
- Support touch screen measurement (Through pin XP, XN, YP, YN)
- Support voltage measurement (Through pin PBAT)
- Support external SAR-ADC input (Through pin SADCIN)
- Separate Channel Conversion Mode
- Single-end and Differential Conversion Mode
- Auto X/Y, X/Y/Z and X/Y/Z1/Z2 position measurement

18.2 Pin Description

Table 18-1 SADC Pin Description

Name	I/O	Description
XN	AI	Touch screen analog differential X- position input
YN	AI	Touch screen analog differential Y- position input
XP	AI	Touch screen analog differential X- position input
YP	AI	Touch screen analog differential Y- position input
ADIN0 (PBAT)	AI	Analog input for VBAT measurement
ADIN1 (SADCIN)	AI	External SAR-ADC input

18.3 Register Description

In this section, we will describe the registers in SAR A/D controller. Following table lists all the register definitions. All registers' 32bit addresses are physical addresses. And detailed function of each register will be described below.

Table 18-2 SADC Register Description

Name	Description	RW	Reset Value	Address	Access Size
ADENA	ADC Enable Register	RW	0x00	0x10070000	8
ADCFG	ADC Configure Register	RW	0x0002000C	0x10070004	32
ADCTRL	ADC Control Register	RW	0x3F	0x10070008	8
ADSTATE	ADC Status Register	RW	0x00	0x1007000C	8
ADSAME	ADC Same Point Time Register	RW	0x0000	0x10070010	16
ADWAIT	ADC Wait Time Register	RW	0x0000	0x10070014	16
ADTCH	ADC Touch Screen Data Register	RW	0x00000000	0x10070018	32
ADBDAT	ADC PBAT Data Register	RW	0x0000	0x1007001C	16
ADSDAT	ADC SADCIN Data Register	RW	0x0000	0x10070020	16
ADFLT	ADC Filter Register	RW	0x0000	0x10070024	16
ADCLK	ADC Clock Divide Register	RW	0x00000000	0x10070028	32

18.3.1 ADC Enable Register (ADENA)

The register ADENA is used to trigger A/D to work.

ADENA **0x10070000**

Bit	7	6	5	4	3	2	1	0
	Reserved	ENTR_SLP	EXIT_SLP	Reserved		TCHEN	PBATEN	SADCINEN
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7	Reserved	Only read and can't write.	R
6	ENTR_SLP	Enter SLEEP Mode Control. Set this bit to 1 to initiate a process of entering the SLEEP mode. When the Touch Screen is ready to enter the SLEEP mode. ENTR_SLP will be cleared by hardware auto.	RW

5	EXIT_SLP	Exit SLEEP Mode Control. Set this bit to 1 to initiate a process of exiting the SLEEP mode. After the Touch Screen has exited from the SLEEP mode. EXIT_SLP will be cleared by hardware auto.	RW
4:3	Reserved	These bits always read 0, and written are ignored.	R
2	TCHEN*1	Touch Screen Enable Control. 0: disable 1: enable	RW
1	PBATEN*1	PBAT Enable Control. Sample the voltage of battery, PBATEN can be set to 1 no matter TCHEN is disable or enable, and when the voltage of battery is ready. PBATEN will be cleared by hardware auto.	RW
0	SADCINEN*1	SADCIN Enable Control. Sample SADCIN, SADCINEN can be set to 1 no matter TCHEN is disable or enable, and when SADCIN is ready, SADCINEN will be cleared by hardware auto.	RW

NOTES:

1 *1. ENTR_SLP, TCHEN, PBATEN and SADCINEN can be set to 1 at the same time. The priority of the three mode is SADCIN > PBAT > ENTR_SLP > TCH.

18.3.2 ADC Configure Register (ADCFG)

The register ADCFG is used to configure the A/D.

ADCFG **0x10007004**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SPZZ	EX_IN	Reserved												DNUM		DMA_EN	XYZ		SNUM			Reserved					BAT_MD	Reserved			
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

Bits	Name	Description	RW
31	SPZZ*1	The $X_dY_dZ_mZ_n$ of different point measure can be different. But the $X_dY_dZ_mZ_n$ of the same point measure can be same or different. 0: The $X_dY_dZ_mZ_n$ of the same point measure is all the same. (X_dY_dZ1Z2, X_dY_dZ1Z2, X_dY_dZ1Z2, X_dY_dZ1Z2 … X_dY_dZ1Z2) 1: The $X_dY_dZ_mZ_n$ of the same point measure maybe different. (X_dY_dZ1Z2, X_dY_dZ3Z4, X_dY_dZ3Z4, X_dY_dZ1Z2 … X_dY_dZ1Z2)	RW
30	EX_IN	Choose external driver or internal driver. 0: X_sY_s or X_sY_sZ	RW

<table>
<tr><td></td><td></td><td>1: X_dY_d or X_dY_dZ
It is no use for $X_dY_dZ_mZ_n$.
It is no use when ADCFG.XYZ = 10.
It is useful when ADCFG.XYZ = 00/01.</td><td></td></tr>
<tr><td>29:19</td><td>Reserved</td><td>These bits always read 0, and written are ignored.</td><td>R</td></tr>
<tr><td>18:16</td><td>DNUM</td><td>This will set which is the sampled data is the virtual value.
Default: = 3’b010.
<table>
<tr><th>DNUM</th><th>Number</th></tr>
<tr><td>3’b000</td><td>Reserved</td></tr>
<tr><td>3’b001</td><td>The virtual value is the 2nd sampled data</td></tr>
<tr><td>3’b010</td><td>The virtual value is the 3rd sampled data</td></tr>
<tr><td>3’b011</td><td>The virtual value is the 4th sampled data</td></tr>
<tr><td>3’b100</td><td>The virtual value is the 5th sampled data</td></tr>
<tr><td>3’b101</td><td>The virtual value is the 6th sampled data</td></tr>
<tr><td>3’b110</td><td>The virtual value is the 7th sampled data</td></tr>
<tr><td>3’b111</td><td>The virtual value is the 8th sampled data</td></tr>
</table></td><td>RW</td></tr>
<tr><td>15</td><td>DMA_EN</td><td>When A/D is used as Touch Screen (CMD=1100), DMA_EN is used as follows.
0: The sample data is read by CPU
1: The sample data is read by DMA</td><td>RW</td></tr>
<tr><td>14:13</td><td>XYZ</td><td>When A/D is used in Touch Screen mode (CMD=1100), XYZ is used as follows.
<table>
<tr><td>XYZ</td><td>Measure (EX_IN = 1)</td><td>Measure (EX_IN = 0)</td></tr>
<tr><td>00</td><td>$X_d \rightarrow Y_d$</td><td>$X_s \rightarrow Y_s$</td></tr>
<tr><td>01</td><td>$X_d \rightarrow Y_d \rightarrow Z_s$</td><td>$X_s \rightarrow Y_s \rightarrow Z_s$</td></tr>
<tr><td>10</td><td>$X_d \rightarrow Y_d \rightarrow Z1_d \rightarrow Z2_d$
or
$X_d \rightarrow Y_d \rightarrow Z3_d \rightarrow Z4_d$</td><td>$X_d \rightarrow Y_d \rightarrow Z1_d \rightarrow Z2_d$
or
$X_d \rightarrow Y_d \rightarrow Z3_d \rightarrow Z4_d$</td></tr>
<tr><td>11</td><td>Reserved</td><td>Reserved</td></tr>
</table></td><td>RW</td></tr>
<tr><td>12:10</td><td>SNUM</td><td>The number of repeated sampling one point. When A/D is used as Touch Screen (CMD=1100), SNUM is used as follows.
<table>
<tr><th>SNUM</th><th>Number</th></tr>
<tr><td>000</td><td>1</td></tr>
<tr><td>001</td><td>2</td></tr>
<tr><td>010</td><td>3</td></tr>
<tr><td>011</td><td>4</td></tr>
<tr><td>100</td><td>5</td></tr>
<tr><td>101</td><td>6</td></tr>
<tr><td>110</td><td>8</td></tr>
<tr><td>111</td><td>9</td></tr>
</table></td><td>RW</td></tr>
<tr><td>9:5</td><td>Reserved</td><td>These bits always read 0, and written are ignored.</td><td>R</td></tr>
<tr><td>4</td><td>BAT_MD</td><td>When AD is used as PBAT measure the following channel mode can be</td><td>RW</td></tr>
</table>

		chose to measure the battery power. 0: PBAT (full battery voltage>=2.5V) 1: PBAT (full battery voltage<2.5V)	
3:0	Reserved	Only read and can't write.	R

NOTES:

1 [*1]: X_s, Y_s, Z_s means the reference mode of X, Y, Z is single-end mode.

X_d, Y_d, $Z1_d$, $Z2_d$, $Z3_d$, Z4d means the reference mode of X, Y, Z1, Z2, Z3, Z4 is differential mode.

When you measure Xs you need to make sure that X-plate is driven by external DC power.

When you measure Ys you need to make sure that Y-plate is driven by external DC power.

18.3.3 ADC Control Register (ADCTRL)

The register ADCTRL is used to control A/D to work.

ADCTRL **0x10070008**

Bit	7	6	5	4	3	2	1	0
	Reserved		SLPENDM	PENDM	PENUM	DTCHM	DRDYM	SRDYM
RST	0	0	1	1	1	1	1	1

Bits	Name	Description	RW
7:6	Reserved	These bits always read 0, and written are ignored.	R

5	SLPENDM	In SLEEP mode pen down interrupt mask. 0: enabled 1: masked	RW
4	PENDM	Pen down interrupt mask. 0: enabled 1: masked	RW
3	PENUM	Pen up interrupt mask. 0: enabled 1: masked	RW
2	DTCHM	Touch Screen Data Ready interrupt mask. 0: enabled 1: masked	RW
1	DRDYM	PBAT data ready interrupt mask. 0: enabled 1: masked	RW
0	SRDYM	SADCIN Data Ready interrupt mask. 0: enabled 1: masked	RW

18.3.4 ADC Status Register (ADSTATE)

The register ADSTATE is used to keep the status of A/D.

ADSTATE **0x1007000C**

Bit	7	6	5	4	3	2	1	0
	Reserved		SLPEND	PEND	PENU	DTCH	DRDY	SRDY
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7:6	Reserved	These bits always read 0, and written are ignored.	R
5	SLPEND	In SLEEP mode pen down interrupt flag. Write 1 to this bit, the bit will clear this bit. 1: active 0: not active	RW
4	PEND	Pen down interrupt flag. Write 1 to this bit, the bit will clear this bit. 1: active 0: not active	RW
3	PENU	Pen up interrupt flag. Write 1 to this bit, the bit will clear this bit. 1: active 0: not active	RW

2	DTCH	Touch screen data ready interrupt flag. Write 1 to this bit, the bit will clear this bit. 1: active 0: not active	RW
1	DRDY	PBAT data ready interrupt flag. Write 1 to this bit, the bit will clear this bit. 1: active 0: not active	RW
0	SRDY	SADCIN Data ready interrupt flag. Write 1 to this bit, the bit will clear this bit. 1: active 0: not active	RW

18.3.5 ADC Same Point Time Register (ADSAME)

The register ADSAME is used to store the interval time between repeated sampling the same point. The clock frequency of the counter is about 1/10us.

ADSAME **0x10070010**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SCNT															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

18.3.6 ADC Wait Pen Down Time Register (ADWAIT)

The register ADWAIT is used to store the interval time of wait pen down. And the register can be used as the interval time among the different point. The clock frequency of the counter is about 1/10us.

ADWAIT **0x10070014**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	WCNT															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

18.3.7 ADC Touch Screen Data Register (ADTCH)

The read-only ADTCH is corresponded to 2x32 bit FIFO, it keep the sample data for touch screen. 0~11 bits are data, 15 bit is data type. 16~27 bits are data, 31 bit is data type. When write to the register, DATA will be clear to 0.

ADTCH **0x10070018**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TYPE1	Reserved			TDATA1												TYPE0	Reserved			TDATA0											
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31	TYPE1	Type of the Touch Screen Data1. When A/D is used as Touch Screen, ADCFG.XYZ=10. TYPE1=1: X_d→Y_d→Z1→Z2 TYPE1=0: X_d→Y_d→Z3→Z4 When A/D is used as Touch Screen, ADCFG.XYZ=00 or XYZ=01, TYPE1=0.	RW
30:28	Reserved	These bits always read 0, and written are ignored.	R
27:16	TDATA1	The concert data of touch screen A/D.	RW
15	TYPE0	Type of the Touch Screen Data2. When A/D is used as Touch Screen, ADCFG.XYZ=10. TYPE0=1: X_d→Y_d→Z1→Z2 TYPE0=0: X_d→Y_d→Z3→Z4 When A/D is used as Touch Screen, ADCFG.XYZ=00 or XYZ=01, TYPE0=0.	RW
14:12	Reserved	These bits always read 0, and written are ignored.	R
11:0	TDATA0	The concert data of touch screen A/D.	RW

NOTES:

1 When A/D is used as Touch Screen, EX_IN=0 and ADCFG.XYZ=00.
The format of touch screen data is as follows:

Type1	Reserved	Data1	Type0	Reserved	Data0
0	000	Y_s	0	000	X_s

2 When A/D is used as Touch Screen, EX_IN=1 and ADCFG.XYZ=00.
The format of touch screen data is as follows:

Type1	Reserved	Data1	Type0	Reserved	Data0
0	000	Y_d	0	000	X_d

3 When A/D is used as Touch Screen, EX_IN=0 and ADCFG.XYZ=01.
The format of touch screen data is as follows:

Type1	Reserved	Data1	Type0	Reserved	Data0
0	000	Y_s	0	000	X_s
0	000	000000000000	0	000	Z_s

Users need to read twice to get the whole data. The first time reading gets the data Y_s and X_s. The second time reading gets the data Z_s. The relation between "touch pressure" and "Z_s" are inverse ratio.

4 When A/D is used as Touch Screen, EX_IN=1 and ADCFG.XYZ=01.
The format of touch screen data is as follows:

Type1	Reserved	Data1	Type0	Reserved	Data0
0	000	Y_d	0	000	X_d
0	000	000000000000	0	000	Z_s

Users need to read twice to get the whole data. The first time reading gets the data Y_d and X_d. The second time reading gets the data Z_s. The relation between "touch pressure" and "Z_s" are inverse ratio.

5 When A/D is used as Touch Screen, ADCFG.XYZ=11,TYPE=1.
The format of touch screen data is as follows:

Type1	Reserved	Data1	Type0	Reserved	Data0
1	000	Y_d	1	000	X_d
1	000	$Z2_d$	1	000	$Z1_d$

Users need to read twice to get the whole data. The first time reading gets the data Y_d and X_d. The second time reading gets the data $Z2_d$ and $Z1_d$.

The touch pressure measurement formula is as follows: (You can use formula 1 or formula 2.)

$$R_{TOUCH} = R_{X\text{-}Plate} \bullet \frac{X\text{-}Position}{4096}\left(\frac{Z_2}{Z_1} - 1\right) \qquad (1)^{*1}$$

$$R_{TOUCH} = \frac{R_{X\text{-}Plate} \bullet X\text{-}Position}{4096}\left(\frac{4096}{Z_1} - 1\right) - R_{Y\text{-}Plate} \bullet \left(1 - \frac{Y\text{-}Position}{4096}\right) \qquad (2)^{*1}$$

6 When A/D is used as Touch Screen, ADCFG.XYZ=11,TYPE=0.
The format of touch screen data is as follows:

Type1	Reserved	Data1	Type0	Reserved	Data0
0	000	Y_d	0	000	X_d
0	000	$Z4_d$	0	000	$Z3_d$

Users need to read twice to get the whole data. The first time reading gets the data Y_d and X_d. The second time reading gets the data $Z4_d$ and $Z3_d$.
The touch pressure measurement formula is as follows: (You can use formula 3 or formula 4.)

$$R_{TOUCH} = R_{Y\text{-}Plate} \bullet \frac{Y\text{-}Position}{4096}\left(\frac{Z_4}{Z_3} - 1\right) \quad (3)^{*1}$$

$$R_{TOUCH} = \frac{R_{Y\text{-}Plate} \bullet Y\text{-}Position}{4096}\left(\frac{4096}{Z_3} - 1\right) - R_{X\text{-}Plate} \bullet \left(1 - \frac{X\text{-}Position}{4096}\right) \quad (4)^{*1}$$

NOTES:

1 *1: To determine pen or finger touch, the pressure of the touch needs to be determined. Generally, it is not necessary to have very high performance for this test; therefore, the 8-bit resolution mode is recommended (however, calculations will be shown here are in 12-bit resolution mode).

$R_{X\text{-}plate}$: Total X-axis resistor value (about 200Ω~ 600Ω)

$R_{Y\text{-}plate}$: Total Y-axis resistor value (about 200Ω~ 600Ω)

X-Position: X-axis voltage sample value

Y-Position: Y-axis voltage sample value

Z1, Z2: Z1, Z2 voltage sample value

Z3, Z4: Z3, Z4 voltage sample value

18.3.8 ADC PBAT Data Register (ADBDAT)

The read-only ADBDAT is a 16-bit register, it keep the sample data of both “PBAT mode”.
0~11 bits are data.

ADBDAT **0x1007001C**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved				BDATA											
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15:12	Reserved	These bits always read 0, and written are ignored.	R
11:0	BDATA	Data of PBAT A/D convert. When write to the register, DATA will be clear to 0.	RW

When ADCCFG.BAT_MD = 0 (full battery voltage>=2.5V), the measured voltage V_{BAT} is as follows:

$$V_{BAT} = \frac{BDATA}{4096} \bullet 7.5V$$

When ADCCFG.BAT_MD = 1 (full battery voltage<2.5V), the measured voltage V_{BAT} is as follows:

$$V_{BAT} = \frac{BDATA}{4096} \bullet 2.5V$$

It is recommended to connect a capacitance of about 0.1uF near to pin ADIN0 to have a more stable battery measurement and better ESD protection.

18.3.9 ADC SADCIN Data Register (ADSDAT)

The read-only ADSDAT is a 16-bit register, it keep the sample data. 0~11 bits are data.

ADSDAT **0x10070020**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved				SDATA											
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15:12	Reserved	These bits always read 0, and written are ignored.	R
11:0	SDATA	Data of SADCIN A/D convert. When write to the register, DATA will be clear to 0.	RW

The measured voltage V_{SADICN} is as follows:

$$V_{SADCIN} = \frac{SDATA}{4096} \bullet 3.3V$$

18.3.10 ADC Filter Register (ADFLT)

ADC Filter Register ADFLT is used for filter out the no valid point for Touch Screen control.

ADFLT **0x10070024**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FLT_EN	Reserved			FLT_D											
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15	FLT_EN	Filter enable bit. 1: Filter function enable 0: Filter function disable	RW
14:12	Reserved	These bits always read 0, and written are ignored.	R
11:0	FLT_D	Filter Data.	RW

NOTE:

When ADFLT.FLT_EN is set to 1,

If (|Z2-Z1|> ADFLT.FLT_D), X=Y=Z1=Z2=0;

If (|Z4-Z3|> ADFLT.FLT_D), X=Y=Z3=Z4=0;

18.3.11 ADC Clock Divide Register (ADCLK)

The register ADCLK is used to set the A/D's clock dividing number.

ADCLK **0x10070028**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved									CLKDIV_10							Reserved										CLKDIV					
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:23	Reserved	These bits always read 0, and written are ignored.	R
22:16	CLKDIV_10	Dividing number to get 10us clock from ADC clock. CLKDIV_10 = adc_clk / 100K – 1 0 ≤CLKDIV_10 ≤127	RW
15:6	Reserved	These bits always read 0, and written are ignored.	R
5:0	CLKDIV	Dividing number to get ADC clock from device clock. The A/D works at the frequency between 500KHz and 8MHz. If CLKDIV = N, Then the freq of adc_clk = dev_clk / (N+1). 0 ≤ N ≤ 63	RW

18.4 SAR A/D Controller Guide

The following describes steps of using SAR-ADC.

18.4.1 Single Operation (internal used only)

(only used as a test mode to check the channel function)

18.4.2 A Sample Touch Screen Operation

(Pen Down → Sample some data of several points → Pen Up)

1 Set ADCTRL to 0x1f to mask all the interrupt of SADC.
2 Set DMA_EN to choose whether to use DMA to read the sample data out or to use CPU to read the sample data out.
3 Set ADCFG.SPZZ, ADCFG.EX_IN and ADCFG.XYZ to choose sample mode.
   a X_s→Y_s (Single-end X→Single-end Y).
   b X_d→Y_d (Differential X→Differential Y).
   c X_s→Y_s→Z_s (Single-end X→Single-end Y→Single-end Z).
   d X_d→Y_d→Z_s (Differential X→Differential Y→Single-end Z).
   e X_d→Y_d→$Z1_d$→$Z2_d$ or X_d→Y_d→$Z3_d$→$Z4_d$ (Reference register ADCFG.SPZZ).
   (Differential X→Differential Y→Differential Z1→Differential Z2 or
   Differential X→Differential Y→Differential Z3→Differential Z4)
4 Set ADCLK.CLKDIV and ADCLK.CLKDIV_10 to set A/D clock frequency.
5 Set ADWAIT to decide the wait time of pen down and the interval time between sampling different points. This time delay is necessary because when pen is put down or pen position change, there should be some time to wait the pen down signal to become stable.
6 Set ADSAME to decide the interval time between repeated sampling the same point. User can repeat sampling one point to get the most accurate data.
7 Set ADCTRL.PENDM to 0 to enable the pen down interrupt of touch panel.
8 Set ADENA.TCHEN to 1 to start touch panel.
9 When pen down interrupt happened, you should set ADCTRL.PENDM to 1 and clear ADSTATE.PEND to close pen down interrupt. Then you should clear ADSTATE.PENDU and set ADCTRL.PENUM to 0 to enable pen up interrupt.
10 When pen down interrupt happened, the SAR ADC is sampling data. When ADSTATE.DTCH to 1, user must read the sample data from ADTCH. The SAR ADC will not sample the next point until the whole data of the one point are read (no matter by CPU or DMA). If ADCFG.XYZ is mode one and mode two, user only needs to read once to get the whole data. In other modes, user needs to read twice to get the whole data.
11 Repeat 10 till pen up interrupt happened.
12 When pen up interrupt happened, you should set ADCTRL.PENUM to 1 and clear ADSTATE.PENU. Then you should clear ADSTATE.PEND and set ADCTRL.PENDM to 0 to enable pen down interrupt.
13 Wait pen down interrupt and repeat from 9.
14 When you want to shut down the touch screen, user can set the ADENA.TCHEN to 0. If the

last point is not sampled completely, user can abandon it.

18.4.3 SLEEP mode Sample Operation

1 Set ADCLK.CLKDIV and ADCLK.CLKDIV_10 to set A/D clock frequency.
2 Then you can set ADENA.ENTR_SLP to 1. When the Touch Screen is ready to enter the SLEEP mode, ADENA.ENTR_SLP will be cleared by hardware auto.
3 After that you should clear ADSTATE.SLPEND and set ADCTRL.SLPENDM to 0 to enable “in SLEEP mode pen down interrupt” and mask all other interrupts. Then you can execute the SLEEP instruction to enter the SLEEP mode.
4 When “in SLEEP mode pen down interrupt” happened, it will switch from the SLEEP mode to NORMAL. Then, you should set ADCTRL.SLPENDM to 1 and clear ADSTATE.SLPEND to close “in SLEEP mode pen down interrupt”. And you should set ADENA.EXIT_SLP to 1. When the Touch Screen has exited from the SLEEP mode, EXIT_SLP will be cleared by hardware auto.
5 Then you can do any other operations.

18.4.4 PBAT Sample Operation

1 Set ADCLK.CLKDIV and ADCLK.CLKDIV_10 to set A/D clock frequency.
2 Set ADCFG.CH_MD to choose PBAT test mode channel.
3 Set ADENA.PBATEN to 1 to enable the channel.
4 When ADSTATE.DRDY = 1, you can read the sample data from ADBDAT. And the PBATEN will be set to 0 auto.

18.4.5 SADCIN Sample Operation

1 Set ADCLK.CLKDIV and ADCLK.CLKDIV_10 to set A/D clock frequency.
2 Set ADENA.SADCINEN to 1 to enable the channel.
3 When ADSTATE. SRDY = 1, you can read the sample data from ADSDAT. And the SADCINEN will be set to 0 auto.

NOTE:

Touch Screen mode can be interrupted by the PBAT and SADCIN mode and “In SLEEP mode pen down”. And the priority is SADCIN > PBAT > ENTR_SLP > TOUCH. If SADCINEN or PBATEN or ENTR_SLP is set to 1 before or at the same time with TCHEN, SAR ADC will first work in SADCIN mode then in PBAT mode, then enter SLEEP mode and at last in touch screen mode (after exit SLEEP). If SADCINEN, PBATEN and ENTR_SLP are set to 1 after the TCHEN, the SAR ADC will work in touch screen mode first and finish sampling the same point completely then turn to the SADCIN, PBAT or SLEEP mode. And return to touch screen mode.

18.4.6 Use TSC to support keypad

SADC TSC function can apply to a keypad, if touch screen is not used. Suppose the keypad is a NxM matrix, where X direction has N key columns and Y direction has M key rows. Kij is used to indicate the key in ith column from left to right and jth row from bottom to top, where i=0~(N-1) and j=0~(M-1). Figure 18-1 is a 6x5 keypad circuit. The blue color is for X direction network and pink color is for Y. The networks are composed by resistors and metal line. These two networks should be connected to SADC 4 pins: XP/XN/YP/YN as illustrated in the figure. The gray circle is the key. When no key pressing, X network and Y network is open circuit. When a key is pressed, the X network and Y network is shorted under the key position.

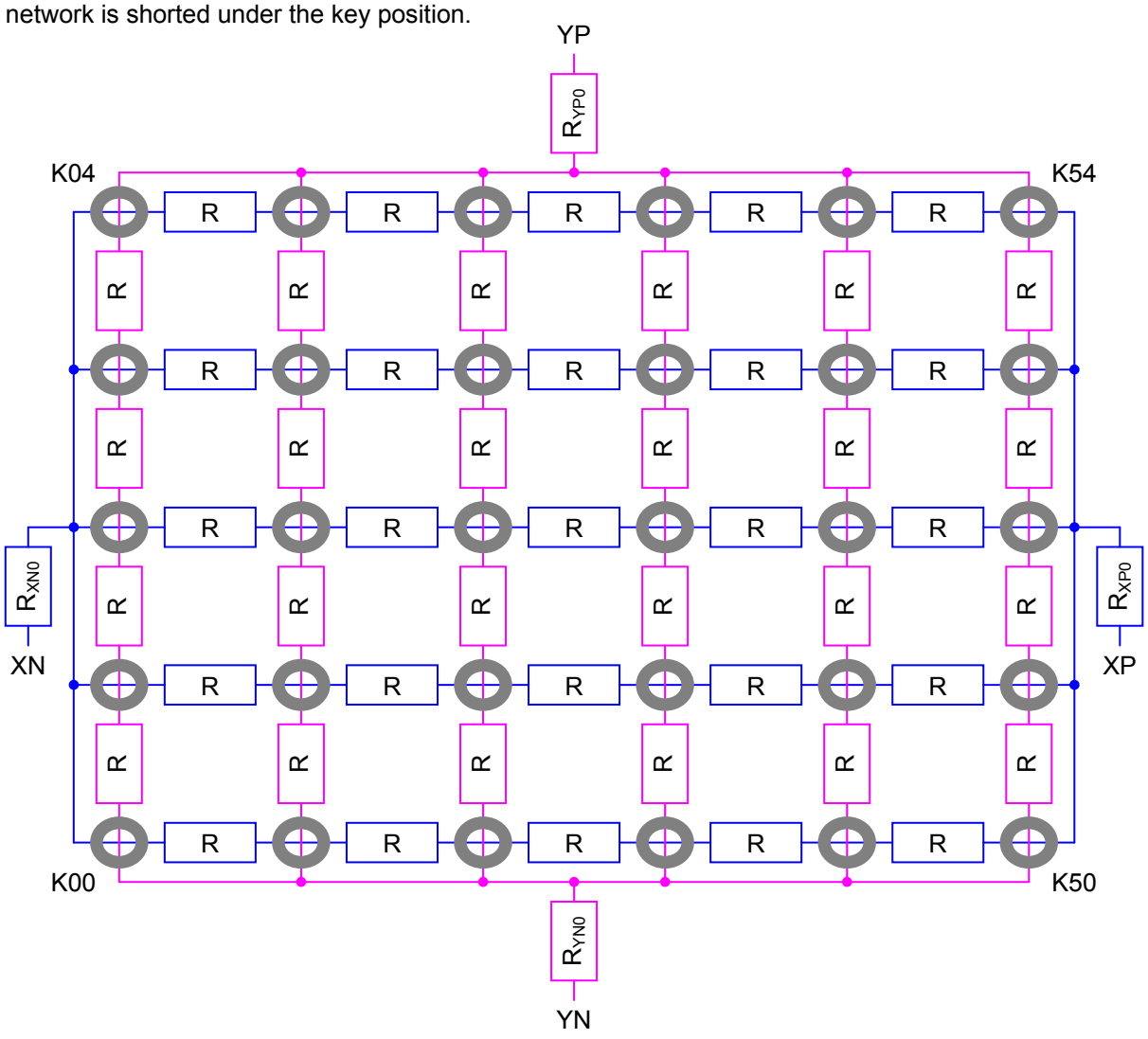


Figure 18-1 6x5 keypad circuit

When SADC is in waiting for pen-down status (C=1100), the equivalent circuit is show in Figure 18-2. When the key is not pressed, XP is open and the PEN is pulled to VDDADC, which is logic 1. When the key Kij is pressed, the circuit is: VDDADC→(10kΩ resistor)→R_{XP}→R_{YN}→VSSADC.

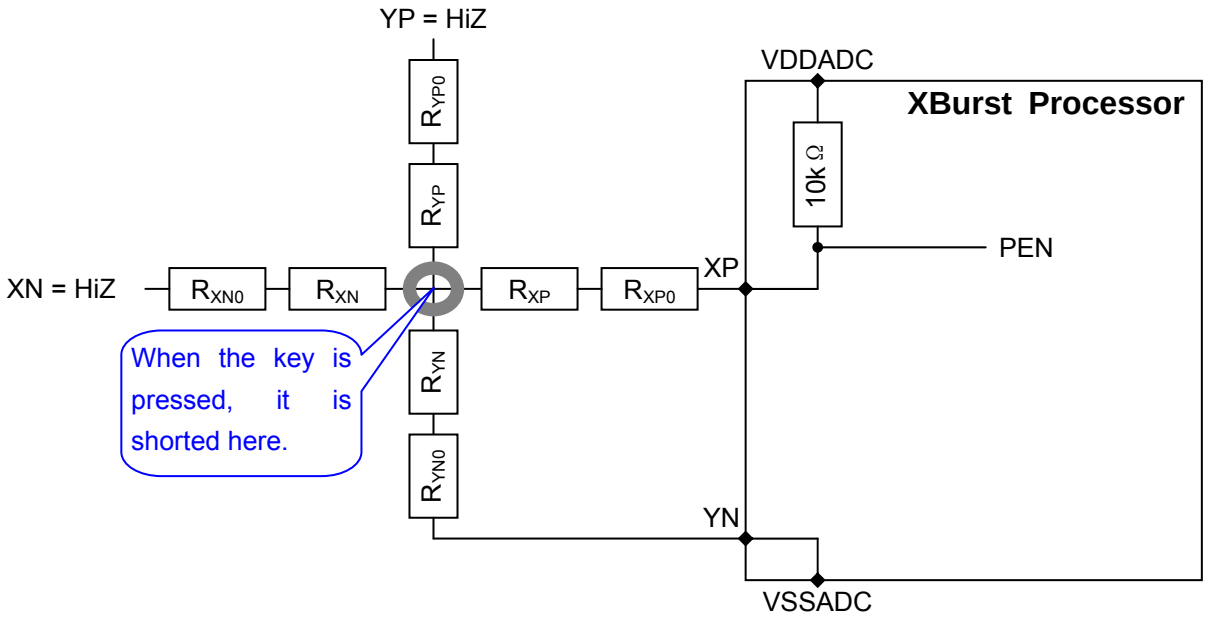


Figure 18-2 Wait for pen-down (C=1100) circuit

Where

$$R_{XP} = \frac{(N-1)^2 - i^2}{M \times (N-1-i) + 2i} \times R$$

$$R_{YN} = \frac{j \times (2M-2-j)}{N \times j + 2M - 2 - 2j} \times R$$

To ensure logic 0 at PEN in this case, following formula should be obeyed.

$$R_{XP} + R_{YN} + R_{XP0} + R_{YN0} \le 3k\Omega \quad (1)$$

It is suggested the value of N and M is as close to each other as possible. For N=2~20, M=2~20 and M=(N-1, N or N+1), we found

$$R_{XP} + R_{YN} < 2.7 \times R \quad (2)$$

After key pressing is found, the key Kij location, columns and row, should be measured by using C=0010 and C=0011 respectively. The equivalent circuits are show in Figure 18-3 and Figure 18-4, where

$$R_{X0} = \frac{N-1}{M-1} \times R$$

$$R_{Y0} = \frac{M-1}{N-1} \times R$$

$$R_{XNi} = i \times R$$

$$R_{XPi} = (N-1-i) \times R$$

$$R_{YNj} = j \times R$$

$$R_{YPj} = (M-1-j) \times R$$

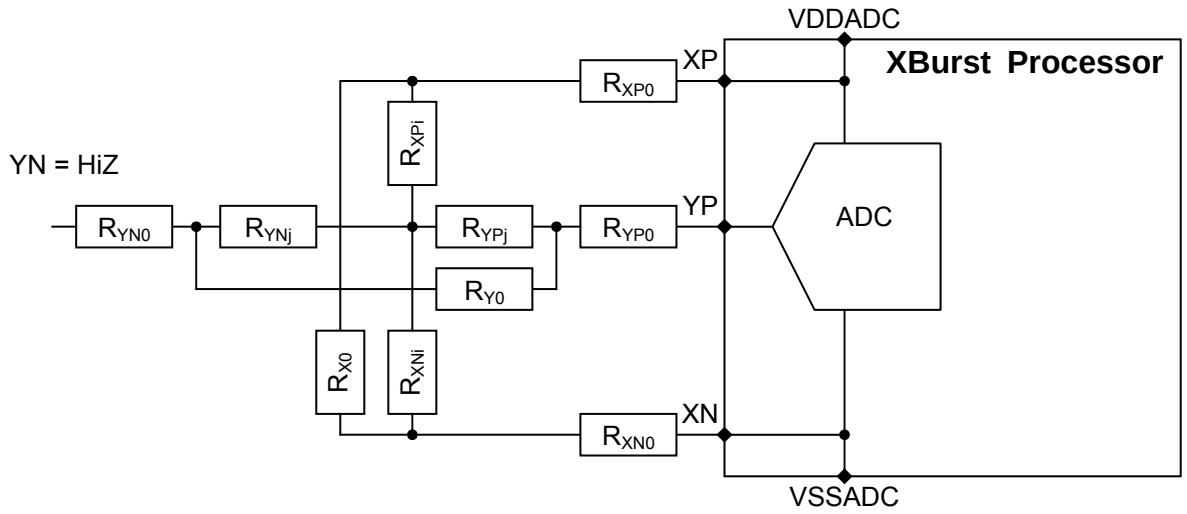


Figure 18-3 Measure X-position (C=0010) circuit

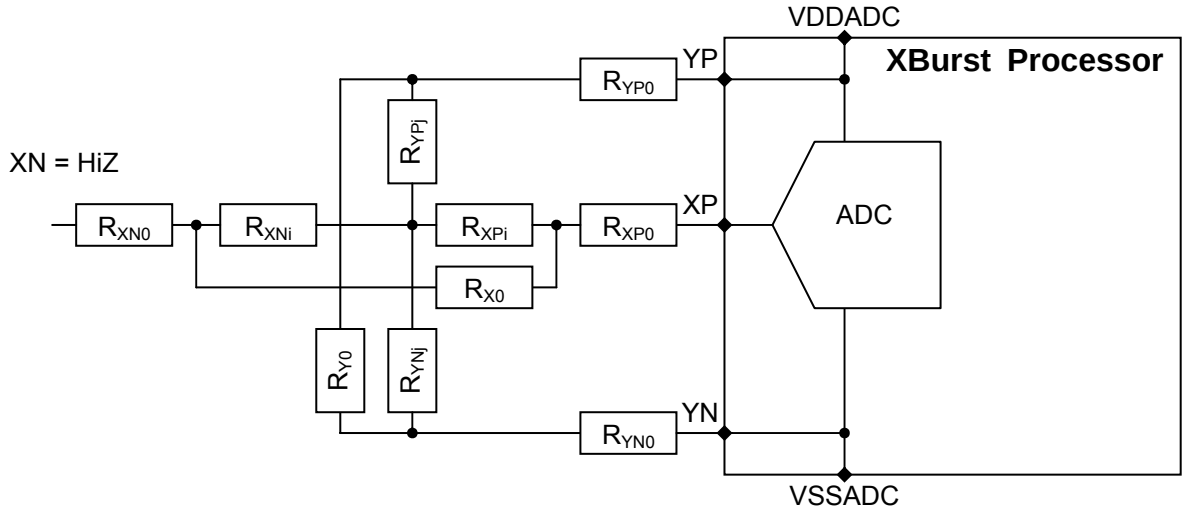


Figure 18-4 Measure Y-position (C=0011) circuit

So for Kij pressing, we should get ADC converted number Ni and Nj for i and j respectively.

$$Ni = \frac{R_{XN0} + \frac{i}{M}R}{R_{XN0} + \frac{N-1}{M}R + R_{XP0}} \times 4096$$

$$Nj = \frac{R_{YN0} + \frac{j}{N}R}{R_{YN0} + \frac{M-1}{N}R + R_{YP0}} \times 4096$$

It is required the resistor between XP and XN in case of C=0010, between YP and YN in case of C=0011, must be $\geqslant 200\,\Omega$ and it better be $\geqslant 500\,\Omega$. Also consider the requirement in formula (1) and

(2) above, we suggest to put $R_{XP0} = R_{XN0} = R_{YP0} = R_{YN0} = 50\Omega$ or 100Ω, put $R = 500\Omega$ ~ $1k\Omega$.

To use the keypad, the software should set:

ADENA.TCHEN = 1

ADCFG.EX_IN = 1

ADCFG.XYZ = 00

The operation is similar to touch screen.

19 General-Purpose I/O Ports

19.1 Overview

General Purpose I/O Ports (GPIO) is used in generating and capturing application-specific input and output signals. Each port can be programmed as an output, an input or function port that serves certain peripheral. As input, pull up/down can be enabled/disabled for the port and the port also can be configured as level or edge tripped interrupt source.

Features:

- Each port can be configured as an input, an output or an alternate function port
- Each port can be configured as an interrupt source of low/high level or rising/falling edge triggering. Every interrupt source can be masked independently
- Each port has an internal pull-up or pull-down resistor connected. The pull-up/down resistor can be disabled

The 105 GPIO ports, named PA00~31, PB00~31, PC00~31 and PD00~31 in JZ4750L are divided into 4 GPIO groups with maximum of 32 GPIO in each group. Group A includes PA00~PA31 (except PA16~PA31); Group B includes PB00~31 (except PB15, PB22, PB23, PB26~PE29); Group C includes PC00~31; Group D includes PD00~PD31. GPIO output 4 interrupts, 1 for every group, to INTC.

For every group, 23 memory-mapped 32-bit registers can be used to operate the GPIO ports:

PAPIN, PBPIN, PCPIN, PDPIN, PEPIN, PFPIN - PIN Level Register
PADAT, PBDAT, PCDAT, PDDAT, PEDAT, PFDAT - Data Register
PADATS, PBDATS, PCDATS, PDDATS, PEDATS, PFDATS - Data Set Register
PADATC, PBDATC, PCDATC, PDDATC, PEDATC, PFDATC - Data Clear Register
PAIM, PBIM, PCIM, PDIM, PEIM, PFIM - Interrupt Mask Register
PAIMS, PBIMS, PCIMS, PDIMS, PEIMS, PFIMS - Interrupt Mask Set Register
PAIMC, PBIMC, PCIMC, PDIMC, PEIMC, PFIMC - Interrupt Mask Clear Register
PAPE, PBPE, PCPE, PDPE, PEPE, PFPE - PULL Disable Register
PAPES, PBPES, PCPES, PDPES, PEPES, PFPES - PULL Disable Set Register
PAPEC, PBPEC, PCPEC, PDPEC, PEPEC, PFPEC - PULL Disable Clear Register
PAFUN, PBFUN, PCFUN, PDFUN, PEFUN, PFFUN - Function Register
PAFUNS, PBFUNS, PCFUNS, PDFUNS, PEFUNS, PFFUNS - Function Set Register
PAFUNC, PBFUNC, PBFUNC, PDFUNC, PEFUNC, PFFUNC - Function Clear Register
PASEL, PBSEL, PCSEL, PDSEL, PESEL, PFSEL - Select Register
PASELS, PBSELS, PCSELS, PDSELS, PESELS, PFSELS - Select Set Register
PASELC, PBSELC, PCSELC, PDSELC, PESELC, PFSELC - Select Clear Register
PADIR, PBDIR, PCDIR, PDDIR, PEDIR, PFDIR - Direction Register
PADIRS, PBDIRS, PCDIRS, PDDIRS, PEDIRS, PFDIRS - Direction Set Register
PADIRC, PBDIRC, PCDIRC, PDDIRC, PEDIRC, PFDIRC - Direction Clear Register

PATRG, PBTRG, PCTRG, PDTRG, PETRG, PFTRG - Trigger Mode Register
PATRGS, PBTRGS, PCTRGS, PDTRGS, PETRGS, PFTRGS - Trigger Mode Set Register
PATRGC, PBTRGC, PCTRGC, PDTRGC, PETRGC, PFTRGC- Trigger Mode Clear Register
PAFLG, PBFLG, PCFLG, PDFLG, PEFLG, PFFLG - FLAG Register

The following table summarize pull resistor, direction and shared function ports for all GPIO.

Table 19-1 GPIO Port A summary

Bit N	PA N	Pull (U/D)	Shared Function Port Selected by			
			Bypass Mode	PFUN = 1 PTRG=0 PSEL = 0	PFUN = 1 PTRG=0 PSEL = 1	Note
0	00	U	-	D0 (io)	-	-
1	01	U	-	D1 (io)	-	-
2	02	U	-	D2 (io)	-	-
3	03	U	-	D3 (io)	-	-
4	04	U	-	D4 (io)	-	-
5	05	U	-	D5 (io)	-	-
6	06	U	-	D6 (io)	-	-
7	07	U	-	D7 (io)	-	-
8	08	U	-	D8 (io)	-	-
9	09	U	-	D9 (io)	-	-
10	10	U	-	D10 (io)	-	-
11	11	U	-	D11 (io)	-	-
12	12	U	-	D12 (io)	-	-
13	13	U	-	D13 (io)	-	-
14	14	U	-	D14 (io)	-	-
15	15	U	-	D15 (io)	-	-

Table 19-2 GPIO Port B summary

Bit N	PB N	Pull (U/D)	Shared Function Port Selected by				
			Bypass Mode	PFUN = 1 PTRG = 0 PSEL = 0	PFUN = 1 PTRG = 0 PSEL = 1	PFUN = 1 PTRG = 1 PSEL = 0	Note
0	00	U	-	A0 (out)	-	-	-
1	01	U	-	A1 (out)	-	-	-
2	02	U	-	A2 (out)	-	-	-
3	03	U	-	A3 (out)	-	-	-
4	04	U	-	A4 (out)	-	-	-
5	05	U	-	A5 (out)	-	-	-
6	06	U	-	A6 (out)	-	-	-
7	07	U	-	A7 (out)	-	-	-
8	08	U	-	A8 (out)	-	-	-
9	09	U	-	A9 (out)	-	-	-
10	10	U	-	A10 (out)	-	-	-
11	11	U	-	A11 (out)	-	-	-
12	12	U	-	A12 (out)	-	-	-
13	13	U	-	A13 (out)	-	-	-
14	14	U	-	A14 (out)	-	-	-
16	16	U	-	DCS0_ (out)	-	-	-
17	17	U	-	RAS_ (out)	-	-	-
18	18	U	-	CAS_ (out)	-	-	-
19	19	U	-	SDWE_ & BUFD_ (out)	-	-	-
20	20	U	-	WE0_ (out)	-	-	-
21	21	U	-	WE1_ (out)	-	-	-
24	24	U	-	CKO (out)	-	-	1
25	25	U	-	CKE (out)	-	-	-
30	30	U				-	2
31	31	U		WKUP		-	3

Table 19-3 GPIO Port C summary

Bit N	PC N	Pull (U/D)	Shared Function Port Selected by				
			Bypass Mode	PFUN = 1 PTRG = 0 PSEL = 0	PFUN = 1 PTRG = 0 PSEL = 1	PFUN = 1 PTRG = 1 PSEL = 0	Note
0	00	U	-	A17 (out)	CIM_D0 (in)	-	-
1	01	U	-	A18 (out)	CIM_D1 (in)	-	-
2	02	U	-	A19 (out)	CIM_D2 (in)	-	-
3	03	U	-	A20 (out)	CIM_D3 (in)	-	-
4	04	U	-	A21 (out)	CIM_D4 (in)	-	-
5	05	U	-	A22 (out)	CIM_D5 (in)	-	-
6	06	U	-	A23 (out)	CIM_D6 (in)	DREQ (in)	-
7	07	U	-	A24 (out)	CIM_D7 (in)	DACK (out)	-
8	08	U		A15 (out)/CL (out) used for share nandflash	MSC0_CLK (out)	-	-
9	09	U		A16 (out)/AL (out) used for share nandflash,	MSC0_CMD (io)	-	-
10	10	U		PWM0 (out)	I2C_SDA (io)	-	-
11	11	U		PWM1 (out)	I2C_SCK (io)	-	-
12	12	U		PWM2 (out)	UART_TXD (out)	-	-
13	13	U		PWM3 (out)	UART_RXD (in)	-	-
14	14	U		PWM4 (out)	-	-	-
15	15	U		PWM5 (out)	-	-	-
16	16	U		SSI_CLK (out)			-
17	17	U		SSI_DT (out)		-	-
18	18	U		SSI_DR (in)		-	-
19	19	U		SSI_CE0_ (out)		-	-
20	20	U		WAIT_ (in)	-	-	-
21	21	U		CS1_ (out)	-	-	-
22	22	U		CS2_ (out)	MSC0_D3 (io)	-	-
23	23	U		CS3_ (out)	-	-	-
24	24	U		CS4_ (out)	-	-	-
25	25	U		RD_ (out)	-	-	-
26	26	U		WR_ (out)	-	-	-
27	27	U	-	MSC0_D2 (io)	-	-	4
28	28	U	-	FRE_ (out)	MSC0_D0 (io)	-	-
29	29	U	-	FWE_ (out)	MSC0_D1 (io)	-	-
30	30	U	-	-	-	-	5,7
31	31	U	-	-	-	--	6,7

Table 19-4 GPIO Port D summary

Bit N	PD N	Pull (U/D)	Shared Function Port Selected by				
			Bypass Mode	PFUN = 1 PTRG = 0 PSEL = 0	PFUN = 1 PTRG = 0 PSEL = 1	PFUN = 1 PTRG = 1 PSEL = 0	Note
0	00	U		LCD_B2 (out)	-		-
1	01	U		LCD_B3 (out)	-		-
2	02	U		LCD_B4 (out)	-		-
3	03	U		LCD_B5 (out)	-		-
4	04	U		LCD_B6 (out)	-		-
5	05	U		LCD_B7 (out)	-		-
6	06	U		LCD_G2 (out)	-		-
7	07	U		LCD_G3 (out)	-		-
8	08	U		LCD_G4 (out)	-		-
9	09	U		LCD_G5 (out)	-		-
10	10	U		LCD_G6 (out)	-		-
11	11	U		LCD_G7 (out)	-		-
12	12	U		LCD_R2 (out)	-		-
13	13	U		LCD_R3 (out)	-		-
14	14	U		LCD_R4 (out)	-		-
15	15	U		LCD_R5 (out)	-		-
16	16	U		LCD_R6 (out)	-		-
17	17	U		LCD_R7 (out)	-		-
18	18	U		LCD_PCLK (io)	-		-
19	19	U		LCD_HSYNC (io)	-		-
20	20	U		LCD_VSYNC (io)	-		-
21	21	U		LCD_DE (out)	-		-
22	22	U		LCD_CLS (out)	LCD_R1 (out)	CIM_MCLK (out)	-
23	23	U		LCD_SPL (out)	LCD_G0 (out)	CIM_HSYNC (in)	-
24	24	U		LCD_PS (out)	LCD_G1 (out)	CIM_PCLK (in)	-
25	25	U		LCD_REV (out)	LCD_B1 (out)	CIM_VSYNC (in)	-
26	26	U		MSC1_CLK (out)	-		-
27	27	U		MSC1_CMD (io)	-		-
28	28	U		MSC1_D0 (io)	-		-
29	29	U		MSC1_D1 (io)	-		-
30	30	U		MSC1_D2 (io)	-		-
31	31	U		MSC1_D3 (io)	-		-

NOTES:

1 PB24: GPIO group B bit 24. It is CKO function when chip is reset.

2 PB30: GPIO group B bit 30. No corresponding pin exists for this GPIO. It is only used to select the function between UART and JTAG, which share the same set of pins, by using register PBSEL [30].

When PBSEL [30]=0, select JTAG function.

When PBSEL [30]=1, select UART function.

3 PB31: GPIO group B bit 31 can only be used as input and interrupt, no pull-up and pull-down.

4 PC27: GPIO group C bit 27. If NAND flash is used, it should connect to NAND FRB. (NAND flash ready/busy) If NAND flash is not used, it is used as general GPIO.

5 PC30: GPIO group C bit 30 is used as BOOT_SEL0 input during boot.

6 PC31: GPIO group C bit 31 is used as BOOT_SEL1 input during boot.

7 BOOT_SEL1, BOOT_SEL0 are used to select boot source and function during the processor boot.

19.2 Register Description

Table 19-5 summarized all memory-mapped registers, which can be programmed to operate GPIO port and alternate function port sharing configuration.

All registers are in 32-bits width. Usually, 1 bit in the register affects a corresponding GPIO port and every GPIO port can be operated independently.

Table 19-5 GPIO Registers

Name	Description	RW	Reset Value	Address	Size
GPIO PORT A					
PAPIN	PORT A PIN Level Register	R	0x00000000	0x10010000	32
PADAT	PORT A Data Register	R	0x00000000	0x10010010	32
PADATS	PORT A Data Set Register	W	0x????????	0x10010014	32
PADATC	PORT A Data Clear Register	W	0x????????	0x10010018	32
PAIM	PORT A Interrupt Mask Register	R	0xFFFFFFFF	0x10010020	32
PAIMS	PORT A Interrupt Mask Set Register	W	0x????????	0x10010024	32
PAIMC	PORT A Interrupt Mask Clear Register	W	0x????????	0x10010028	32
PAPE	PORT A PULL Disable Register	R	0x00000000	0x10010030	32
PAPES	PORT A PULL Disable Set Register	W	0x????????	0x10010034	32
PAPEC	PORT A PULL Disable Clear Register	W	0x????????	0x10010038	32
PAFUN	PORT A Function Register	R	0x00000000	0x10010040	32
PAFUNS	PORT A Function Set Register	W	0x????????	0x10010044	32
PAFUNC	PORT A Function Clear Register	W	0x????????	0x10010048	32
PASEL	PORT A Select Register	R	0x00000000	0x10010050	32
PASELS	PORT A Select Set Register	W	0x????????	0x10010054	32
PASELC	PORT A Select Clear Register	W	0x????????	0x10010058	32
PADIR	PORT A Direction Register	R	0x00000000	0x10010060	32
PADIRS	PORT A Direction Set Register	W	0x????????	0x10010064	32
PADIRC	PORT A Direction Clear Register	W	0x????????	0x10010068	32
PATRG	PORT A Trigger Register	R	0x00000000	0x10010070	32
PATRGS	PORT A Trigger Set Register	W	0x????????	0x10010074	32
PATRGC	PORT A Trigger Clear Register	W	0x????????	0x10010078	32
PAFLG	PORT A FLAG Register	R	0x00000000	0x10010080	32
PAFLGC	PORT A FLAG Clear Register	W	0x????????	0x10010014	32
GPIO PORT B					
PBPIN	PORT B PIN Level Register	R	0x00000000	0x10010100	32
PBDAT	PORT B Data Register	R	0x00000000	0x10010110	32
PBDATS	PORT B Data Set Register	W	0x????????	0x10010114	32
PBDATC	PORT B Data Clear Register	W	0x????????	0x10010118	32
PBIM	PORT B Interrupt Mask Register	R	0xFFFFFFFF	0x10010120	32
PBIMS	PORT B Interrupt Mask Set Register	W	0x????????	0x10010124	32

PBIMC	PORT B Interrupt Mask Clear Register	W	0x????????	0x10010128	32
PBPE	PORT B PULL Enable Register	R	0x00000000	0x10010130	32
PBPES	PORT B PULL Enable Set Register	W	0x????????	0x10010134	32
PBPEC	PORT B PULL Enable Clear Register	W	0x????????	0x10010138	32
PBFUN	PORT B Function Register	R	0x00000000	0x10010140	32
PBFUNS	PORT B Function Set Register	W	0x????????	0x10010144	32
PBFUNC	PORT B Function Clear Register	W	0x????????	0x10010148	32
PBSEL	PORT B Select Register	R	0x00000000	0x10010150	32
PBSELS	PORT B Select Set Register	W	0x????????	0x10010154	32
PBSELC	PORT B Select Clear Register	W	0x????????	0x10010158	32
PBDIR	PORT B Direction Register	R	0x00000000	0x10010160	32
PBDIRS	PORT B Direction Set Register	W	0x????????	0x10010164	32
PBDIRC	PORT B Direction Clear Register	W	0x????????	0x10010168	32
PBTRG	PORT B Trigger Register	R	0x00000000	0x10010170	32
PBTRGS	PORT B Trigger Set Register	W	0x????????	0x10010174	32
PBTRGC	PORT B Trigger Clear Register	W	0x????????	0x10010178	32
PBFLG	PORT B FLAG Register	R	0x00000000	0x10010180	32
PBFLGC	PORT B FLAG Clear Register	W	0x????????	0x10010114	32
GPIO PORT C					
PCPIN	PORT C PIN Level Register	R	0x00000000	0x10010200	32
PCDAT	PORT C Data Register	R	0x00000000	0x10010210	32
PCDATS	PORT C Data Set Register	W	0x????????	0x10010214	32
PCDATC	PORT C Data Clear Register	W	0x????????	0x10010218	32
PCIM	PORT C Interrupt Mask Register	R	0xFFFFFFFF	0x10010220	32
PCIMS	PORT C Interrupt Mask Set Register	W	0x????????	0x10010224	32
PCIMC	PORT C Interrupt Mask Clear Register	W	0x????????	0x10010228	32
PCPE	PORT C PULL Enable Register	R	0x00000000	0x10010230	32
PCPES	PORT C PULL Enable Set Register	W	0x????????	0x10010234	32
PCPEC	PORT C PULL Enable Clear Register	W	0x????????	0x10010238	32
PCFUN	PORT C Function Register	R	0x00000000	0x10010240	32
PCFUNS	PORT C Function Set Register	W	0x????????	0x10010244	32
PCFUNC	PORT C Function Clear Register	W	0x????????	0x10010248	32
PCSEL	PORT C Select Register	R	0x00000000	0x10010250	32
PCSELS	PORT C Select Set Register	W	0x????????	0x10010254	32
PCSELC	PORT C Select Clear Register	W	0x????????	0x10010258	32
PCDIR	PORT C Direction Register	R	0x00000000	0x10010260	32
PCDIRS	PORT C Direction Set Register	W	0x????????	0x10010264	32
PCDIRC	PORT C Direction Clear Register	W	0x????????	0x10010268	32
PCTRG	PORT C Trigger Register	R	0x00000000	0x10010270	32
PCTRGS	PORT C Trigger Set Register	W	0x????????	0x10010274	32
PCTRGC	PORT C Trigger Clear Register	W	0x????????	0x10010278	32

PCFLG	PORT C FLAG Register	R	0x00000000	0x10010280	32
PCFLGC	PORT C FLAG Clear Register	W	0x????????	0x10010214	32
GPIO PORT D					
PDPIN	PORT D PIN Level Register	R	0x00000000	0x10010300	32
PDDAT	PORT D Data Register	R	0x00000000	0x10010310	32
PDDATS	PORT D Data Set Register	W	0x????????	0x10010314	32
PDDATC	PORT D Data Clear Register	W	0x????????	0x10010318	32
PDIM	PORT D Interrupt Mask Register	R	0xFFFFFFFF	0x10010320	32
PDIMS	PORT D Interrupt Mask Set Register	W	0x????????	0x10010324	32
PDIMC	PORT D Interrupt Mask Clear Register	W	0x????????	0x10010328	32
PDPE	PORT D PULL Enable Register	R	0x00000000	0x10010330	32
PDPES	PORT D PULL Enable Set Register	W	0x????????	0x10010334	32
PDPEC	PORT D PULL Enable Clear Register	W	0x????????	0x10010338	32
PDFUN	PORT D Function Register	R	0x00000000	0x10010340	32
PDFUNS	PORT D Function Set Register	W	0x????????	0x10010344	32
PDFUNC	PORT D Function Clear Register	W	0x????????	0x10010348	32
PDSEL	PORT D Select Register	R	0x00000000	0x10010350	32
PDSELS	PORT D Select Set Register	W	0x????????	0x10010354	32
PDSELC	PORT D Select Clear Register	W	0x????????	0x10010358	32
P\DDIR	PORT D Direction Register	R	0x00000000	0x10010360	32
PDDIRS	PORT D Direction Set Register	W	0x????????	0x10010364	32
PDDIRC	PORT D Direction Clear Register	W	0x????????	0x10010368	32
PDTRG	PORT D Trigger Register	R	0x00000000	0x10010370	32
PDTRGS	PORT D Trigger Set Register	W	0x????????	0x10010374	32
PDTRGC	PORT D Trigger Clear Register	W	0x????????	0x10010378	32
PDFLG	PORT D FLAG Register	R	0x00000000	0x10010380	32
PDFLGC	PORT D FLAG Clear Register	W	0x????????	0x10010314	32

NOTE: PX**** in the description of register as follows means PA****, PB****, PC****, PD****.

19.2.1 PORT PIN Level Register (PAPIN, PBPIN, PCPIN, PDPIN)

PAPIN, PBPIN, PCPIN and PDPIN are four 32-bit PORT PIN level registers. They are read-only registers.

PAPIN, PBPIN, PCPIN, PDPIN **0x10010000, 0x10010100, 0x10010200, 0x10010300**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PINL31	PINL30	PINL29	PINL28	PINL27	PINL26	PINL25	PINL24	PINL23	PINL22	PINL21	PINL20	PINL19	PINL18	PINL17	PINL16	PINL15	PINL14	PINL13	PINL12	PINL11	PINL10	PINL09	PINL08	PINL07	PINL06	PINL05	PINL04	PINL03	PINL02	PINL01	PINL00
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
n	PINL n	Where n = 0 ~ 31 and PINL n = PINL0 ~ PINL31. The PORT PIN level can be read by reading PINL n bit in register PXPIN.	R

PAPIN bits 31-0 correspond to PA31-0; PBPIN to PB31-0; PCPIN to PC31-0 and PDPIN to PD 31-0.

19.2.2 PORT Data Register (PADAT, PBDAT, PCDAT, PDDAT)

PADAT, PBDAT, PCDAT and PDDAT are four 32-bit PORT DATA registers. They are read-only registers.

PADAT, PBDAT, PCDAT, PDDAT **0x10010010, 0x10010110, 0x10010210, 0x10010310**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATA31	DATA30	DATA29	DATA28	DATA27	DATA26	DATA25	DATA24	DATA23	DATA22	DATA21	DATA20	DATA19	DATA18	DATA17	DATA16	DATA15	DATA14	DATA13	DATA12	DATA11	DATA10	DATA09	DATA08	DATA07	DATA06	DATA05	DATA04	DATA03	DATA02	DATA01	DATA00
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
n	DATA n	Where n = 0 ~ 31 and DATA n = DATA0 ~ DATA31. The register is used as GPIO data register. When GPIO is used as interrupt the register is no used.	R

PADAT bits 31-0 correspond to PA31-0; PBDAT to PB31-0; PCDAT to PC31-0 and PDDAT to PD 31-0.

19.2.3 PORT Data Set Register (PADATS, PBDATS, PCDATS, PDDATS)

PADATS, PBDATS, PCDATS and PDDATA are four 32-bit PORT DATA set registers. They are write-only registers.

PADATS, PBDATS, PCDATS, PDDATS **0x10010014, 0x10010114, 0x10010214, 0x10010314**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATAS31	DATAS30	DATAS29	DATAS28	DATAS27	DATAS26	DATAS25	DATAS24	DATAS23	DATAS22	DATAS21	DATAS20	DATAS19	DATAS18	DATAS17	DATAS16	DATAS15	DATAS14	DATAS13	DATAS12	DATAS11	DATAS10	DATAS09	DATAS08	DATAS07	DATAS06	DATAS05	DATAS04	DATAS03	DATAS02	DATAS01	DATAS00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	DATAS n	Writing 1 to DATAS n will set DATA n to 1 in register PXDAT. Writing 0 to DATAS n will no use.	W

PADATS bits 31-0 correspond to PA31-0; PBDATS to PB31-0; PCDATS to PC31-0 and PDDATS to PD 31-0.

19.2.4 PORT Data Clear Register (PADATC, PBDATC, PCDATC, PDDATC)

PADATC, PBDATC, PCDATC and PDDATC are four 32-bit PORT DATA clear registers. They are write-only registers.

PADATC, PBDATC, PCDATC, PDDATC **0x10010018, 0x10010118, 0x10010218, 0x10010318**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATAC31	DATAC30	DATAC29	DATAC28	DATAC27	DATAC26	DATAC25	DATAC24	DATAC23	DATAC22	DATAC21	DATAC20	DATAC19	DATAC18	DATAC17	DATAC16	DATAC15	DATAC14	DATAC13	DATAC12	DATAC11	DATAC10	DATAC09	DATAC08	DATAC07	DATAC06	DATAC05	DATAC04	DATAC03	DATAC02	DATAC01	DATAC00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	DATAC n	Writing 1 to DATAC n will set DATA n to 0 in register PXDAT. Writing 0 to DATAC n will no use.	W

PADATC bits 31-0 correspond to PA31-0; PBDATC to PB31-0; PCDATC to PC31-0 and PDDATC to PD 31-0.

19.2.5 PORT Mask Register (PAIM, PBIM, PCIM, PDIM)

PAIM, PBIM, PCIM and PDIM are four 32-bit PORT MASK registers. They are read-only registers.

PAIM, PBIM, PCIM, PDIM **0x10010020, 0x10010120, 0x10010220, 0x10010320**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MASK31	MASK30	MASK29	MASK28	MASK27	MASK26	MASK25	MASK24	MASK23	MASK22	MASK21	MASK20	MASK19	MASK18	MASK17	MASK16	MASK15	MASK14	MASK13	MASK12	MASK11	MASK10	MASK09	MASK08	MASK07	MASK06	MASK05	MASK04	MASK03	MASK02	MASK01	MASK00
RST	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description	R/W
n	MASK n	Where n = 0 ~ 31 and MASK n = MASK0 ~ MASK31. MASK n is used for mask the interrupt of GPIO n. 0: Enable the pin as an interrupt source 1: Disable the pin as an interrupt source	R

PAIM bits 31-0 correspond to PA31-0; PBIM to PB31-0; PCIM to PC31-0 and PDIM to PD 31-0.

19.2.6 PORT Mask Set Register (PAIMS, PBIMS, PCIMS, PDIMS)

PAIMS, PBIMS, PCIMS and PIMS are four 32-bit PORT MASK set registers. They are write-only registers.

PAIMS, PBIMS, PCIMS, PDIMS **0x10010024, 0x10010124, 0x10010224, 0x10010324**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MASKS31	MASKS30	MASKS29	MASKS28	MASKS27	MASKS26	MASKS25	MASKS24	MASKS23	MASKS22	MASKS21	MASKS20	MASKS19	MASKS18	MASKS17	MASKS16	MASKS15	MASKS14	MASKS13	MASKS12	MASKS11	MASKS10	MASKS09	MASKS08	MASKS07	MASKS06	MASKS05	MASKS04	MASKS03	MASKS02	MASKS01	MASKS00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	MASKS n	Writing 1 to MASKS n will set MASK n to 1 in register PXIM. Writing 0 to MASKS n will no use.	W

PAIMS bits 31-0 correspond to PA31-0; PBIMS to PB31-0; PCIMS to PC31-0 and PDIMS to PD 31-0.

19.2.7 PORT Mask Clear Register (PAIMC, GBPIMC, PCIMC, PDIMC)

PAIMC, PBIMC, PCIMC and PDIMC are four 32-bit PORT MASK clear registers. They are write-only registers.

PAIMS, PBIMC, PCIMC, PDIMC **0x10010028, 0x10010128, 0x10010228, 0x10010328**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MASKC31	MASKC30	MASKC29	MASKC28	MASKC27	MASKC26	MASKC25	MASKC24	MASKC23	MASKC22	MASKC21	MASKC20	MASKC19	MASKC18	MASKC17	MASKC16	MASKC15	MASKC14	MASKC13	MASKC12	MASKC11	MASKC10	MASKC09	MASKC08	MASKC07	MASKC06	MASKC05	MASKC04	MASKC03	MASKC02	MASKC01	MASKC00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	MASKC n	Writing 1 to MASKC n will set MASK n to 0 in register PXIM. Writing 0 to MASKC n will no use.	W

PAIMC bits 31-0 correspond to PA31-0; PBIMC to PB31-0; PCIMC to PC31-0 and PDIMC to PD 31-0.

19.2.8 PORT PULL Disable Register (PAPE, PBPE, PCPE, PDPE)

PAPE, PBPE, PCPE and PDPE are four 32-bit PORT PULL disable registers. They are read-only registers.

PAPE, PBPE, PCPE, PDPE **0x10010030, 0x10010130, 0x10010230, 0x10010330**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PULL31	PULL30	PULL29	PULL28	PULL27	PULL26	PULL25	PULL24	PULL23	PULL22	PULL21	PULL20	PULL19	PULL18	PULL17	PULL16	PULL15	PULL14	PULL13	PULL12	PULL11	PULL10	PULL09	PULL08	PULL07	PULL06	PULL05	PULL04	PULL03	PULL02	PULL01	PULL00
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
n	PULL n	Where n = 0 ~ 31 and PULL n = PULL0 ~ PULL31. PULL n is used for setting the port to be PULL UP or PULL DOWN enable. 1: No pull up or pull down resistor connects to the port 0: An internal pull up or pull down resistor connects to the port. Up or down is pin dependence. Please reference to 1 ~ 4 for it	R

PAPE bits 31-0 correspond to PA31-0; PBPE to PB31-0; PCPE to PC31-0 and PDPE to PD 31-0.

19.2.9 PORT PULL Set Register (PAPES, PBPES, PCPES, PDPES)

PAPES, PBPES, PCPES and PDPES are four 32-bit PORT PULL set registers. They are write-only registers.

PAPES, PBPES, PCPES, PDPES **0x10010034, 0x10010134, 0x10010234, 0x10010334**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PULLS31	PULLS30	PULLS29	PULLS28	PULLS27	PULLS26	PULLS25	PULLS24	PULLS23	PULLS22	PULLS21	PULLS20	PULLS19	PULLS18	PULLS17	PULLS16	PULLS15	PULLS14	PULLS13	PULLS12	PULLS11	PULLS10	PULLS09	PULLS08	PULLS07	PULLS06	PULLS05	PULLS04	PULLS03	PULLS02	PULLS01	PULLS00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	PULLS n	Writing 1 to PULLS n will set PULL n to 1 in register PXPE. Writing 0 to PULLS n will no use.	W

PAPES bits 31-0 correspond to PA31-0; PBPES to PB31-0; PCPES to PC31-0 and PDPES to PD 31-0.

19.2.10 PORT PULL Clear Register (PAPEC, PBPEC, PCPEC, PDPEC)

PAPEC, PBPEC, PCPEC and PDPEC are four 32-bit PORT PULL clear registers. They are write-only registers.

PAPES, PBPEC, PCPEC, PDPEC **0x10010038, 0x10010138, 0x10010238, 0x10010338**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PULLC31	PULLC30	PULLC29	PULLC28	PULLC27	PULLC26	PULLC25	PULLC24	PULLC23	PULLC22	PULLC21	PULLC20	PULLC19	PULLC18	PULLC17	PULLC16	PULLC15	PULLC14	PULLC13	PULLC12	PULLC11	PULLC10	PULLC09	PULLC08	PULLC07	PULLC06	PULLC05	PULLC04	PULLC03	PULLC02	PULLC01	PULLC00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	PULLC n	Writing 1 to PULLC n will set PULL n to 0 in register PXPE. Writing 0 to PULLC n will no use.	W

PAPEC bits 31-0 correspond to PA31-0; PBPEC to PB31-0; PCPEC to PC31-0 and PDPEC to PD 31-0.

19.2.11 PORT Function Register (PAFUN, PBFUN, PCFUN, PDFUN)

PAFUN, PBFUN, PCFUN and PDFUN are four 32-bit PORT function registers. They are read-only registers.

PAFUN, PBFUN, PCFUN, PDFUN **0x10010040, 0x10010140, 0x10010240, 0x10010340**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FUN31	FUN30	FUN29	FUN28	FUN27	FUN26	FUN25	FUN24	FUN23	FUN22	FUN21	FUN20	FUN19	FUN18	FUN17	FUN16	FUN15	FUN14	FUN13	FUN12	FUN11	FUN10	FUN09	FUN08	FUN07	FUN06	FUN05	FUN04	FUN03	FUN02	FUN01	FUN00
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
n	FUN n	Where n = 0 ~ 31 and FUN n = FUN0 ~ FUN31. In most cases, port is shared with one or more peripheral functions. FUN n controls the owner of the port n. 0: GPIO or Interrupt 1: Alternate Function (Function 0 *1 or Function 1*1) **NOTE:** Please reference to 1 ~ 4 for the details.	R

PAFUN bits 31-0 correspond to PA31-0; PBFUN to PB31-0; PCFUN to PC31-0 and PDFUN to PD 31-0.

19.2.12 PORT Function Set Register (PAFUNS, PBFUNS, PCFUNS, PDFUNS)

PAFUNS, PBFUNS, PCFUNS and PDFUNS are four 32-bit PORT function set registers. They are write-only registers.

PAFUNS, PBFUNS, PCFUNS, PDFUNS **0x10010044, 0x10010144, 0x10010244, 0x10010344**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FUNS31	FUNS30	FUNS29	FUNS28	FUNS27	FUNS26	FUNS25	FUNS24	FUNS23	FUNS22	FUNS21	FUNS20	FUNS19	FUNS18	FUNS17	FUNS16	FUNS15	FUNS14	FUNS13	FUNS12	FUNS11	FUNS10	FUNS09	FUNS08	FUNS07	FUNS06	FUNS05	FUNS04	FUNS03	FUNS02	FUNS01	FUNS00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	FUNS n	Writing 1 to FUNS n will set FUN n to 1 in register PXFUN. Writing 0 to FUNS n will no use.	W

PAFUNS bits 31-0 correspond to PA31-0; PBFUNS to PB31-0; PCFUNS to PC31-0 and PDFUNS to PD 31-0.

19.2.13 PORT Function Clear Register (PAFUNC, PBFUNC, PCFUNC, PDFUNC)

PAFUNC, PBFUNC, PCFUNC and PDFUNC are four 32-bit PORT function clear registers. They are write-only registers.

PAFUNC, PBFUNC, PCFUNC, PDFUNC **0x10010048, 0x10010148, 0x10010248, 0x10010348**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FUNC31	FUNC30	FUNC29	FUNC28	FUNC27	FUNC26	FUNC25	FUNC24	FUNC23	FUNC22	FUNC21	FUNC20	FUNC19	FUNC18	FUNC17	FUNC16	FUNC15	FUNC14	FUNC13	FUNC12	FUNC11	FUNC10	FUNC09	FUNC08	FUNC07	FUNC06	FUNC05	FUNC04	FUNC03	FUNC02	FUNC01	FUNC00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	FUNC n	Writing 1 to FUNC n will set FUN n to 0 in register PXFUN. Writing 0 to FUNC n will no use.	W

PAFUNC bits 31-0 correspond to PA31-0; PBFUNC to PB31-0; PCFUNC to PC31-0 and PDFUNC to PD 31-0.

19.2.14 PORT Select Register (PASEL, PBSEL, PCFSEL, PDSEL)

PASEL, PBSEL, PCSEL and PDSEL are four 32-bit PORT select registers. They are read-only registers.

PASEL, PBSEL, PCSEL, PDSEL **0x10010050, 0x10010150, 0x10010250, 0x10010350**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SEL31	SEL30	SEL29	SEL28	SEL27	SEL26	SEL25	SEL24	SEL23	SEL22	SEL21	SEL20	SEL19	SEL18	SEL17	SEL16	SEL15	SEL14	SEL13	SEL12	SEL11	SEL10	SEL09	SEL08	SEL07	SEL06	SEL05	SEL04	SEL03	SEL02	SEL01	SEL00
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
n	SEL n	Where n = 0 ~ 31 and SEL n = SEL0 ~ SEL31. SEL n is used for selecting the function of GPIO. When PXFUN = 0: 0: GPIO 1: Interrupt When PXFUN = 1: 0: Alternate Function 0[*1] 1: Alternate Function 1[*1] **NOTE:** Please reference to 1 ~ 4 for the details.	R

PASEL bits 31-0 correspond to PA31-0; PBSEL to PB31-0; PCSEL to PC31-0 and PDSEL to PD 31-0.

19.2.15 PORT Select Set Register (PASELS, PBSELS, PCSELS, PDSELS)

PASELS, PBSELS, PCSELS and PDSELS are four 32-bit PORT select set registers. They are write-only registers.

PASELS, PBSELS, PCSELS, PFDSELS **0x10010054, 0x10010154, 0x10010254, 0x10010354**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SELS31	SELS30	SELS29	SELS28	SELS27	SELS26	SELS25	SELS24	SELS23	SELS22	SELS21	SELS20	SELS19	SELS18	SELS17	SELS16	SELS15	SELS14	SELS13	SELS12	SELS11	SELS10	SELS09	SELS08	SELS07	SELS06	SELS05	SELS04	SELS03	SELS02	SELS01	SELS00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	SELS n	Writing 1 to SELS n will set SEL n to 1 in register PXSEL. Writing 0 to SELS n will no use.	W

PASELS bits 31-0 correspond to PA31-0; PBSELS to PB31-0; PCSELS to PC31-0 and PDSELS to PD 31-0.

19.2.16 PORT Select Clear Register (PASELC, PBSELC, PCSELC, PDSELC)

PASELC, PBSELC, PCSELC and PDSELC are four 32-bit PORT select clear registers. They are write-only registers.

PASELC, PBSELC, PCSELC, PDSELC **0x10010058, 0x10010158, 0x10010258, 0x10010358**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SELC31	SELC30	SELC29	SELC28	SELC27	SELC26	SELC25	SELC24	SELC23	SELC22	SELC21	SELC20	SELC19	SELC18	SELC17	SELC16	SELC15	SELC14	SELC13	SELC12	SELC11	SELC10	SELC09	SELC08	SELC07	SELC06	SELC05	SELC04	SELC03	SELC02	SELC01	SELC00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	SELC n	Writing 1 to SELC n will set SEL n to 0 in register PXSEL. Writing 0 to SELC n will no use.	W

PASELC bits 31-0 correspond to PA31-0; PBSELC to PB31-0; PCSELC to PC31-0 and PDSELC to PD 31-0.

19.2.17 PORT Direction Register (PADIR, PBDIR, PCDIR, PDDIR)

PADIR, PBDIR, PCDIR and PDDIR are four 32-bit PORT direction registers. They are read-only registers.

PADIR, PBDIR, PCDIR, PDDIR **0x10010060, 0x10010160, 0x10010260, 0x10010360**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DIR31	DIR30	DIR29	DIR28	DIR27	DIR26	DIR25	DIR24	DIR23	DIR22	DIR21	DIR20	DIR19	DIR18	DIR17	DIR16	DIR15	DIR14	DIR13	DIR12	DIR11	DIR10	DIR09	DIR08	DIR07	DIR06	DIR05	DIR04	DIR03	DIR02	DIR01	DIR00
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
n	DIR n	Where n = 0 ~ 31 and DIR n = DIR0 ~ DIR31. DIR n is used for setting the direction of port or setting the trigger direction of interrupt trigger. GPIO Direction: (GPIO Function) 0: INPUT 1: OUTPUT Interrupt Trigger Direction: (Interrupt Function) PXTRG = 0: 0: Low Level Trigger 1: High Level Trigger PXTRG =1: 0: Falling Edge Trigger 1: Rising Edge Trigger	R

PADIR bits 31-0 correspond to PA31-0; PBDIR to PB31-0; PCDIR to PC31-0 and PDDIR to PD 31-0.

19.2.18 PORT Direction Set Register (PADIRS, PBDIRS, PCDIRS, PDDIRS)

PADIRS, PBDIRS, PCDIRS and PDDIRS are four 32-bit PORT direction set registers. They are write-only registers.

PADIRS, PBDIRS, PCDIRS, PDDIRS **0x10010064, 0x10010164, 0x10010264, 0x10010364**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DIRS31	DIRS30	DIRS29	DIRS28	DIRS27	DIRS26	DIRS25	DIRS24	DIRS23	DIRS22	DIRS21	DIRS20	DIRS19	DIRS18	DIRS17	DIRS16	DIRS15	DIRS14	DIRS13	DIRS12	DIRS11	DIRS10	DIRS09	DIRS08	DIRS07	DIRS06	DIRS05	DIRS04	DIRS03	DIRS02	DIRS01	DIRS00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	DIRS n	Writing 1 to DIRS n will set DIR n to 1 in register PXDIR.	W

		Writing 0 to DIRS n will no use.	

PADIRS bits 31-0 correspond to PA31-0; PBDIRS to PB31-0; PCDIRS to PC31-0 and PDDIRS to PD 31-0.

19.2.19 PORT Direction Clear Register (PADIRC, PBDIRC, PCDIRC, PDDIRC)

GPDIRC0, GPDIRC1, GPDIRC2 and GPDIRC3 are four 32-bit PORT direction clear registers. They are write-only registers.

PADIRS, PBDIRC, PCDIRC, PDDIRC **0x10010068, 0x10010168, 0x10010268, 0x10010368**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DIRC31	DIRC30	DIRC29	DIRC28	DIRC27	DIRC26	DIRC25	DIRC24	DIRC23	DIRC22	DIRC21	DIRC20	DIRC19	DIRC18	DIRC17	DIRC16	DIRC15	DIRC14	DIRC13	DIRC12	DIRC11	DIRC10	DIRC09	DIRC08	DIRC07	DIRC06	DIRC05	DIRC04	DIRC03	DIRC02	DIRC01	DIRC00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	DIRC n	Writing 1 to DIRC n will set DIR n to 0 in register PXDIR. Writing 0 to DIRC n will no use.	W

PADIRC bits 31-0 correspond to PA31-0; PBDIRC to PB31-0; PCDIRC to PC31-0 and PDDIRC to PD 31-0.

19.2.20 PORT Trigger Register 0, 1, 2 and 3 (PATRG, PBTRG, PCTRG, PDTRG)

PATRG, PBTRG, PCTRG and PDTRG are four 32-bit PORT trigger registers. They are read-only registers.

PATRG, PBTRG, PCTRG, PDTRG **0x10010070, 0x10010170, 0x10010270, 0x10010370**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TRIG31	TRIG30	TRIG29	TRIG28	TRIG27	TRIG26	TRIG25	TRIG24	TRIG23	TRIG22	TRIG21	TRIG20	TRIG19	TRIG18	TRIG17	TRIG16	TRIG15	TRIG14	TRIG13	TRIG12	TRIG11	TRIG10	TRIG09	TRIG08	TRIG07	TRIG06	TRIG05	TRIG04	TRIG03	TRIG02	TRIG01	TRIG00
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
n	TRIG n	Where n = 0 ~ 31 and TRIG n = TRIG00 ~ TRIG31. TRIG n is used for setting the trigger mode for interrupt. When GPIO is used as interrupt function: 0: Level Trigger Interrupt 1: Edge Trigger Interrupt	R

		When GPIO is used as alternate function: 0: Alternate Function Group 0 1: Alternate Function Group 1	

PATRG bits 31-0 correspond to PA31-0; PBTRG to PB31-0; PCTRG to PC31-0 and PDTRG to PD 31-0.

19.2.21 PORT Trigger Set Register (PATRGS, PBTRGS, PCTRGS, PDTRGS)

PATRGS, PBTRGS, PCTRGS and PDTRGS are four 32-bit PORT trigger set registers. They are write-only registers.

PATRGS, PBTRGS, PCTRGS, PDTRGS **0x10010074, 0x10010174, 0x10010274, 0x10010374**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TRIGS31	TRIGS30	TRIGS29	TRIGS28	TRIGS27	TRIGS26	TRIGS25	TRIGS24	TRIGS23	TRIGS22	TRIGS21	TRIGS20	TRIGS19	TRIGS18	TRIGS17	TRIGS16	TRIGS15	TRIGS14	TRIGS13	TRIGS12	TRIGS11	TRIGS10	TRIGS09	TRIGS08	TRIGS07	TRIGS06	TRIGS05	TRIGS04	TRIGS03	TRIGS02	TRIGS01	TRIGS00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	TRIGS n	Writing 1 to TRIGS n will set TRIG n to 1 in register PXTRG. Writing 0 to TRIGS n will no use.	W

PATRGS bits 31-0 correspond to PA31-0; PBTRGS to PB31-0; PCTRGS to PC31-0 and PDTRGS to PD 31-0.

19.2.22 PORT Trigger Clear Register (PATRGC, PBTRGC, PCTRGC, PDTRGC)

PATRGC, PBTRGC, PCTRGC and PDTRGC are four 32-bit PORT trigger clear registers. They are write-only registers.

PATRGC, PBTRGC, PCTRGC, PDTRGC **0x10010078, 0x10010178, 0x10010278, 0x10010378**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TRIGC31	TRIGC30	TRIGC29	TRIGC28	TRIGC27	TRIGC26	TRIGC25	TRIGC24	TRIGC23	TRIGC22	TRIGC21	TRIGC20	TRIGC19	TRIGC18	TRIGC17	TRIGC16	TRIGC15	TRIGC14	TRIGC13	TRIGC12	TRIGC11	TRIGC10	TRIGC09	TRIGC08	TRIGC07	TRIGC06	TRIGC05	TRIGC04	TRIGC03	TRIGC02	TRIGC01	TRIGC00
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	R/W
n	TRIGC n	Writing 1 to TRIGC n will set TRIG n to 0 in register PXTRG. Writing 0 to TRIGC n will no use.	W

PATRGC bits 31-0 correspond to PA31-0; PBTRGC to PB31-0; PCTRGC to PC31-0 and PDTRGC to PD 31-0.

19.2.23 PORT FLAG Register (PAFLG, PBFLG, PCFLG, PDFLG)

PAFLG, PBFLG, PCFLG and PDFLG are four 32-bit GPIO FLAG registers. They are read-only registers.

PAFLG, PBFLG, PCFLG, PDFLG **0x10010080, 0x10010180, 0x10010280, 0x10010380**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FLAG31	FLAG30	FLAG29	FLAG28	FLAG27	FLAG26	FLAG25	FLAG24	FLAG23	FLAG22	FLAG21	FLAG20	FLAG19	FLAG18	FLAG17	FLAG16	FLAG15	FLAG14	FLAG13	FLAG12	FLAG11	FLAG10	FLAG09	FLAG08	FLAG07	FLAG06	FLAG05	FLAG04	FLAG03	FLAG02	FLAG01	FLAG00
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
n	FLAG n	Where n = 0 ~ 31 and FLAG n = FLAG00 ~ FLAG31. FLAG n is interrupt flag bit for checking the interrupt whether to happen. When GPIO is used as interrupt function and the interrupt happened, the FLAG n in PXFLG will be set to 1.	R

PAFLG bits 31-0 correspond to PA31-0; PBFLG to PB31-0; PCFLG to PC31-0 and PDFLG to PD 31-0.

19.2.24 PORT FLAG Clear Register (PAFLGC, PBFLGC, PCFLGC, PDFLGC)

PAFLGC, PBFLGC, PCFLGC and PDFLGC are four 32-bit GPIO FLAG Clear registers. They are read-only registers.

PAFLGC, PBFLGC, PCFLGC, PDFLGC **0x10010014, 0x10010114, 0x10010214, 0x10010314**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FLAGC31	FLAGC30	FLAGC29	FLAGC28	FLAGC27	FLAGC26	FLAGC25	FLAGC24	FLAGC23	FLAGC22	FLAGC21	FLAGC20	FLAGC19	FLAGC18	FLAGC17	FLAGC16	FLAGC15	FLAGC14	FLAGC13	FLAGC12	FLAGC11	FLAGC10	FLAGC09	FLAGC08	FLAGC07	FLAGC06	FLAGC05	FLAGC04	FLAGC03	FLAGC02	FLAGC01	FLAGC00
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
n	FLAGC n	When GPIO is used as interrupt function and when write 1 to the bit, the bit FLAG n in PXFLG will be cleared.	R

PAFLGC bits 31-0 correspond to PA31-0; PBFLGC to PB31-0; PCFLGC to PC31-0 and PDFLGC to PD 31-0.

19.3 Program Guide

19.3.1 GPIO Function Guide

1 Set PXFUN to choose the function of GPIO / Interrupt by writing 1 to register PXFUNC.
2 Set PXSEL to choose the function of GPIO by writing 1 to register PXSELC.
3 Set PXDIR to choose the direction of GPIO by writing 1 to register PXDIRS or PXDIRC.
4 Others.
  a You can read the PORT PIN level by reading register PXPIN.
  b You can use register PXDAT as normal data register. The register can be set by register PXDATS and PXDATC.
  c You can set PXPE by writing 1 to register PXPES or PXPE to use Internal pull-up/down resistor or not.

19.3.2 Alternate Function Guide

1 Set PXFUN to 0 by writing 1 to register PXFUNC. (Ready state)
2 Set PXTRG to choose the alternate function group 0 by writing 1 to register PXTRGC.
  Set PXTRG to choose the alternate function group 1 by writing 1 to register PXTRGS.
3 Set PXSEL to choose the alternate function 0 by writing 1 to register PXSELC.
  Set PXSEL to choose the alternate function 1 by writing 1 to register PXSELS.
4 Set PXFUN to choose the function of alternate function by writing 1 to register PXFUNS.

19.3.3 Interrupt Function Guide

First you should keep GPIO status.

1 Set PXIM by writing 1 to register PXIMS.
2 Set PXTRG to choose the interrupt trigger mode by writing 1 to register PXTRGS or PXTRGC.
3 Set PXFUN to choose the function of GPIO / Interrupt by writing 1 to register or PXFUNC.
4 Set PXSEL to choose the Interrupt function by writing 1 to register PXSELS.
5 Set PXDIR to choose the direction of interrupt trigger by writing 1 to register PXDIRS or PXDIRC.
6 Set the PXFLGC register to clear the interrupt flag.
7 Clear PXIM by writing 1 to register PXIMC to enable the GPIO interrupt.
8 Others.
  You should check the level interrupt whether to happen as follows:
  a When the PIN level read from register PXPIN is the same with what you have set in register PXTRG and PXDIR, then the level interrupt happened.
  b When the PIN level read from register PXPIN is different from what you have set in register PXTRG and PXDIR, then the level interrupt did not happen.

19.3.4 Disable Interrupt Function Guide

1 Set PXIM by writing 1 to register PXIMS.
2 Set PXTRG to 0 by writing 1 to register PXTRGC.
3 Set PXDIR to 0 by writing 1 to register PXDIRC.
4 Set PXFUN to 0 by writing 1 to register or PXFUNC.
5 Set PXSEL to 0 by writing 1 to register PXSELC.

20 I2C Bus Interface

20.1 Overview

The I2C bus was created by the Phillips Corporation and is a serial bus with a two-pin interface. The SDA data pin is used for input and output functions and the SCL clock pin is used to control and reference the I2C bus. The I2C unit allows the processor to serve as a master and slave device that resides on the I2C bus. The I2C unit enables the processor to communicate with I2C peripherals and microcontrollers for system management functions. The I2C bus requires a minimum amount of hardware to relay status and reliability information concerning the processor subsystem to an external device. The I2C unit is a peripheral device that resides on the processor internal bus. Data is transmitted to and received from the I2C bus via a buffered interface. Control and status information is relayed through a set of memory-mapped registers. Refer to ***The I2C-Bus Specification*** for complete details on I2C bus operation.

The I2C has the following features:

- Supports only single master mode
- Supports I2C standard-mode and F/S-mode up to 400 kHz
- I2C receiver and transmitter are double-buffered
- Supports burst reading or writing of data
- Supports random writing access of data
- Supports general call address and START byte format after START condition
- Independent, programmable serial clock generator
- Supports slave coping with fast master during data transfers by holding the SCL line on a bit level
- The number of devices that you can connect to the same I2C-bus is limited only by the maximum bus capacitance of 400pF

20.2 Pin Description

Table 20-1 Smart Card Controller Pins Description

Name	I/O	Description
SDA	Input/Output	I2C Serial Clock Line signal.
SCL	Input/Output	I2C Serial Data/Address signal.

20.3 Register Description

Table 20-2 I2C Registers Description

Name	RW	Reset Value	Address	Access Size
I2CDR	RW	0x??	0x10042000	8
I2CCR	RW	0x00	0x10042004	8
I2CSR	RW	0x04	0x10042008	8
I2CGR	RW	0x0000	0x1004200C	16

20.3.1 Data Register (I2CDR)

I2CDR 0x10042000

Bit	7	6	5	4	3	2	1	0
	DR							
RST	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
7:0	DR	Data port of HW FIFO.	RW

20.3.2 Control Register (I2CCCR)

I2CCR 0x10042004

Bit	7	6	5	4	3	2	1	0
	Reserved			IEN	STA*	STO*	AC	I2CE
RST	0	0	0	0	0	0	0	0

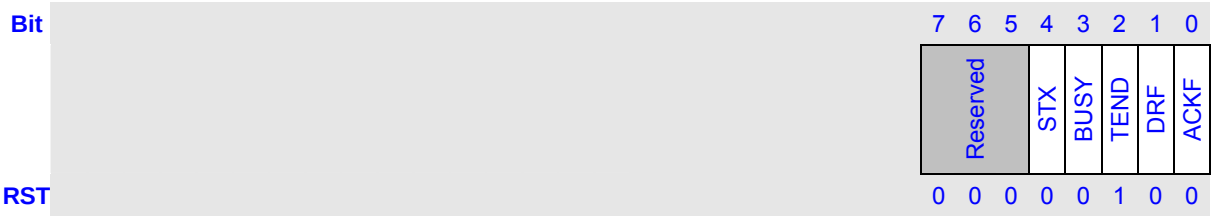
NOTE: STA and STO can only be written with 1.

Bits	Name	Description	RW
7:5	Reserved	These bits always read as 0. Write data to these bits are ignored.	R
4	IEN	I2C interrupt bit. 0: Disable I2C interrupt; 1: Enable I2C interrupt.	RW
3	STA	I2C START bit. 0: START condition will not be sent to I^2C bus; 1: START condition will be sent to I^2C bus.	RW
2	STO	I2C STOP bit. 0: STOP condition won't be sent to I^2C bus; 1: STOP condition will be sent to I^2C bus.	RW

1	AC	I2C Acknowledge Control Bit. 0: will be sent to I^{2}C bus as LOW level acknowledge signal; 1: will be sent to I^{2}C bus as HIGH level acknowledge signal.	RW
0	I2CE	Enable of I2C. 0: I2C module is disabled; 1: I2C module is enabled.	RW

20.3.3 Status Register (I2CSR)

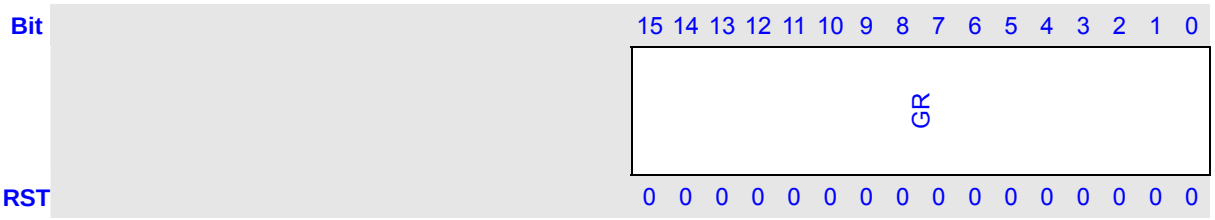
I2CSR 0x10042008



Bits	Name	Description	RW
7:5	Reserved	These bits always read as 0. Write data to these bits are ignored.	R
4	STX	STA/STO Command is On. 0: STA/STO FIFO buffer is empty; 1: STA/STO FIFO buffer is not empty.	R
3	BUSY	I2C Bus Busy. 0: I2C bus is free; 1: I2C bus is busy.	R
2	TEND	Transmission End Flag. 0: Byte transmission or acknowledge bit for that byte has not completed; 1: The I2C is in transmission idle state.	R
1	DRF	Data Register Valid Flag. 0: Data in I2CDR is invalid; 1: Data in I2CDR is valid.	RW
0	ACKF	Acknowledge Level Flag. 0: The acknowledge signal from I^{2}C-bus is “0”; 1: The acknowledge signal from I^{2}C-bus is “1”.	R

20.3.4 Clock Generator Register (I2CGR)

I2CGR 0x1004200C



Bits	Name	Description	RW
15:01	GR	Sets the frequency of serial clock. The serial clocks frequency is calculated as follows: [Value of I2CGR] = [Frequency of Device_clock] / (16 * [SCL clock rate]) – 1	RW

NOTE: To make the I2C operate normally, frequency of PCLK (APB-bus clock) should not lower than transfer 2 * [byte rate].

20.4 I²C-Bus Protocol

20.4.1 Bit Transfer

Due to the variety of different technology devices (CMOS, NMOS, bipolar) which can be connected to the I²C-bus, the levels of the logical '0' (LOW) and '1' (HIGH) are not fixed and depend on the associated level of VDD. One clock pulse is generated for each data bit transferred.

20.4.2 Data Validity

The data on the SDA line must be stable during the HIGH period of the clock. The HIGH or LOW states of the data line can only change when the clock signal on the SCL line is LOW.

20.4.3 START and STOP Conditions

A HIGH to LOW transition on the SDA line while SCL is HIGH indicates a START condition. A LOW to HIGH transition on the SDA line while SCL is HIGH defines a STOP condition.

20.4.4 Byte Format

1 Every byte put on the SDA line must be 8-bits width.
2 The number of bytes that can be transmitted/received per transfer is unrestricted.
3 Each byte has to be followed by an acknowledge (ack/nack) bit.
4 Data is transferred with the most significant bit (MSB) first.
5 Data transfer with an acknowledge signal (acknowledge or not-acknowledge) is obligatory.
6 The acknowledge_ related clock pulse is generated by the master.
7 The transmitter releases the SDA line (HIGH) during the acknowledge clock pulse. The receiver must pull down the SDA line during the acknowledge clock pulse so that it remains stable LOW during the HIGH period of this clock pulse.
8 Slave can hold the SCL line LOW during the SCL in LOW level at any bit to force the master to proceed a lower speed of transfer.

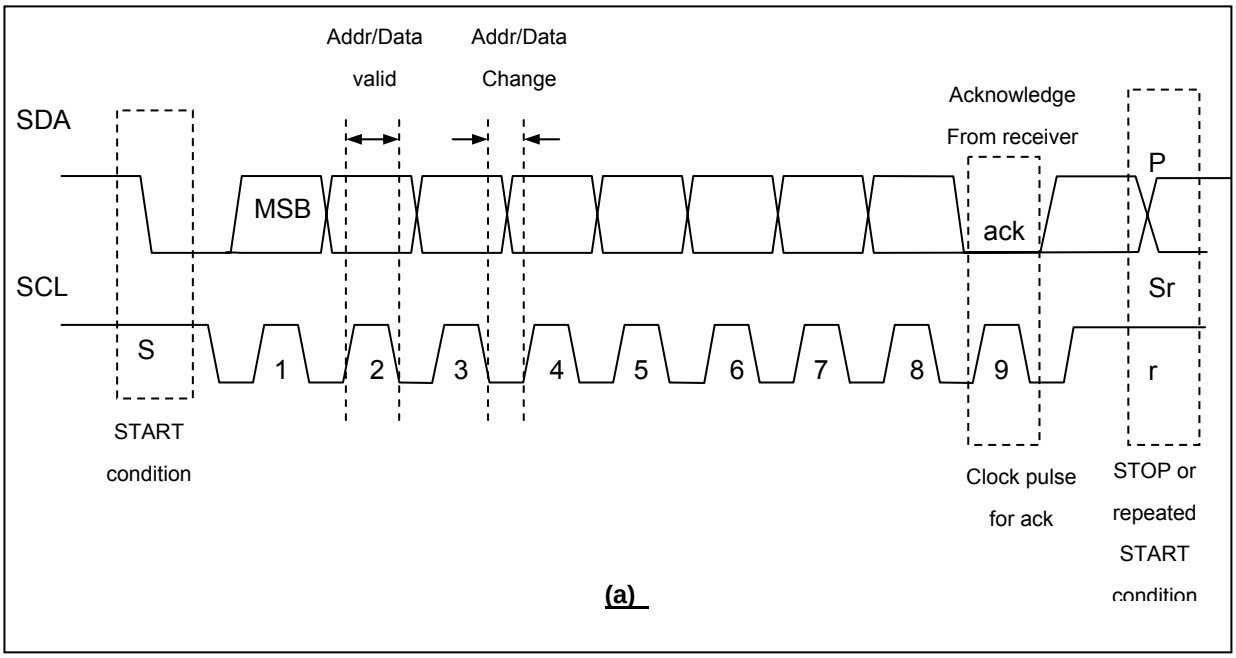


Figure 20-1 I2C-bus Protocol

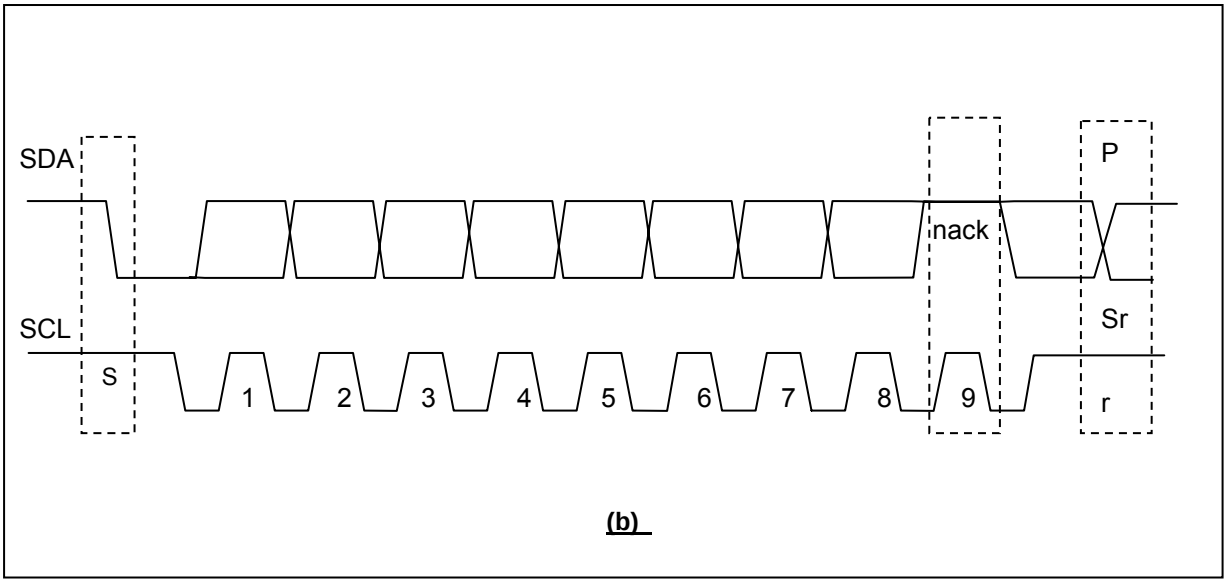


Figure 20-2 I^{2}C-bus Protocol (cont.)

NOTES:

1 Sr means repeated START condition. P means STOP condition.

2 In Fig (a), if the master does not generate Sr or P, the next data byte follows the ack.

3 In Fig (b), nack is received, the master generates Sr or P and the transfer terminates.

20.4.5 Data Transfer Format

20.4.5.1 First Byte

The first byte is a term indicates the address byte after START condition.

1 Normal 7-bit Address.

After the START condition, the addressing procedure for the I^2C-bus is such that the first byte usually determines which slave will been selected by the master.

The first seven bits of the first byte make up the slave address. The eighth bit is the LSB (least significant bit). It determines the direction of the message. A 'zero' in the least significant position of the first byte means that the master will write information to a selected slave. A 'one' in this position means that the master will read information from the slave.

When an address is sent, each device in a system compares the first seven bits after the START condition with its address. If they match, the device considers itself addressed by the master as a slave-receiver or slave-transmitter, depending on the R/W bit.

A slave address can be made-up of a fixed and a programmable part. Since it's likely that there will be several identical devices in a system, the programmable part of the slave address enables the maximum possible number of such devices to be connected to the I^2C-bus. The number of programmable address bits of a device depends on the number of pins available. For example, if a device has 4 fixed and 3 programmable address bits, a total of 8 identical devices can be connected to the same bus.

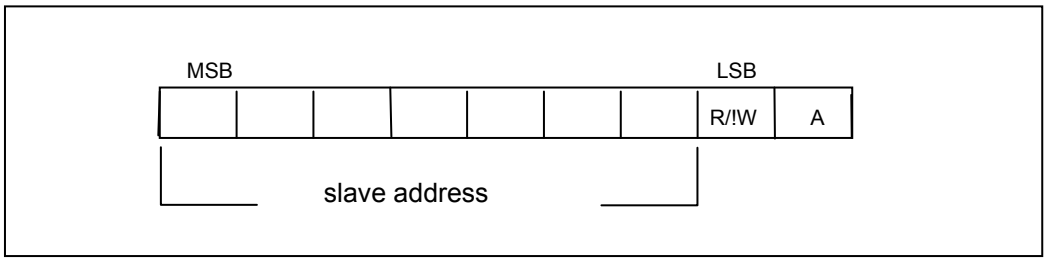


Figure 20-3 Normal 7 Bit Address after START Condition

2 General Call Address.

Address byte with all bits are "0" is defined as "general call address". When this address is used, all devices should, in theory, respond with an acknowledge. However, if a device doesn't need any of the data supplied within the general call structure, it can ignore this address by not issuing an acknowledgment. If a device does require data from a general call address, it will acknowledge this address and behave as a slave- receiver. The second and following bytes will be acknowledged by every slave-receiver capable of handling this data. A slave that cannot process

one of these bytes must ignore it by not-acknowledging.

The second byte of the general call address then defines the action to be taken.

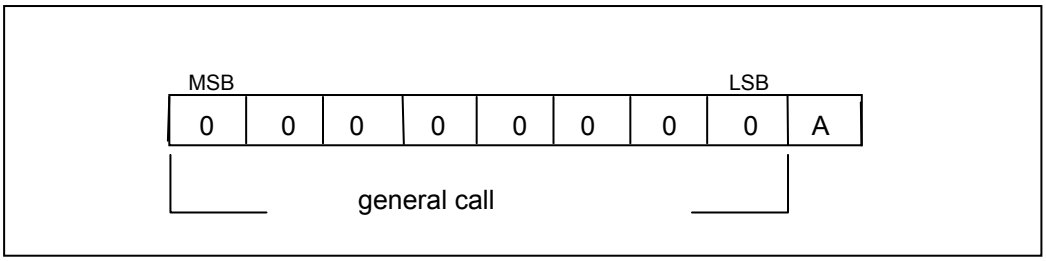


Figure 20-4 General Call Address after START Condition

3 START Byte Address.

START Byte:

After the START condition S has been transmitted by the master, data transfer can be preceded by a start procedure which is much longer than normal. The start procedure consists of:

a A START condition (S)
b A START byte (00000001)
c An acknowledge clock pulse (ACK)*
d A repeated START condition (Sr)

NOTE: An acknowledge-related clock pulse is generated after the START byte. This is present only to conform to the byte handling format used on the bus. No device is allowed to acknowledge the START byte.

When the START byte (00000001) is transmitted, another microcontroller (the slave) can therefore sample the SDA line at a low sampling rate (also determined by the I2CGR) until one of the seven zeros in the START byte is detected. After detection of this LOW level on the SDA line, the microcontroller can switch to a higher sampling rate to find the repeated START condition Sr which is then used for synchronization.

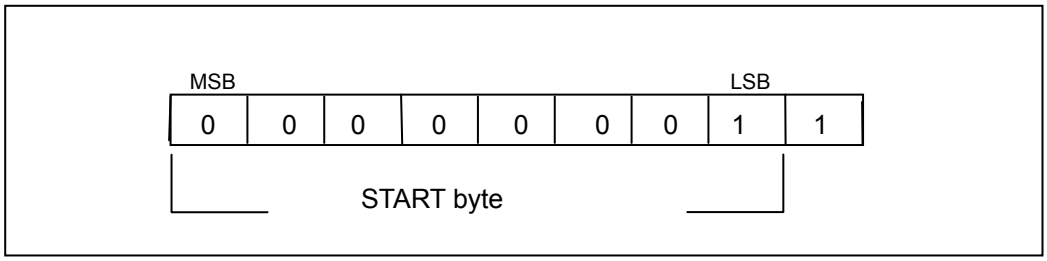


Figure 20-5 START Byte after START Condition

20.4.5.2 Transfer Format

A data transfer is always terminated by a STOP condition (P) generated by the master. However, if a master still wishes to communicate on the bus, it can generate a repeated START condition (Sr) and address another slave without first generating a STOP condition. Various combinations of read/write formats are then possible within such a transfer.

Possible data transfer formats are:

1 Master-transmitter transmits to slave-receiver. The transfer direction is not changed.
2 Master reads slave immediately after first byte. At the moment of the first acknowledge, the master-transmitter becomes a master- receiver and the slave-receiver becomes a slave-transmitter.
3 This first acknowledge is still generated by the slave. The STOP condition is generated by the master, which has previously sent a not-acknowledge.

NOTES:

1 Combined formats can be used, for example, to control a serial memory. During the first data byte, the internal memory location has to be written. After the START condition and slave address is repeated, data can be transferred.
2 All decisions on auto-increment or decrement of previously accessed memory locations etc. are taken by the designer of the device.
3 Each byte is followed by an acknowledgment bit as indicated by the ‘A ‘or ‘!A ‘ blocks in the sequence.

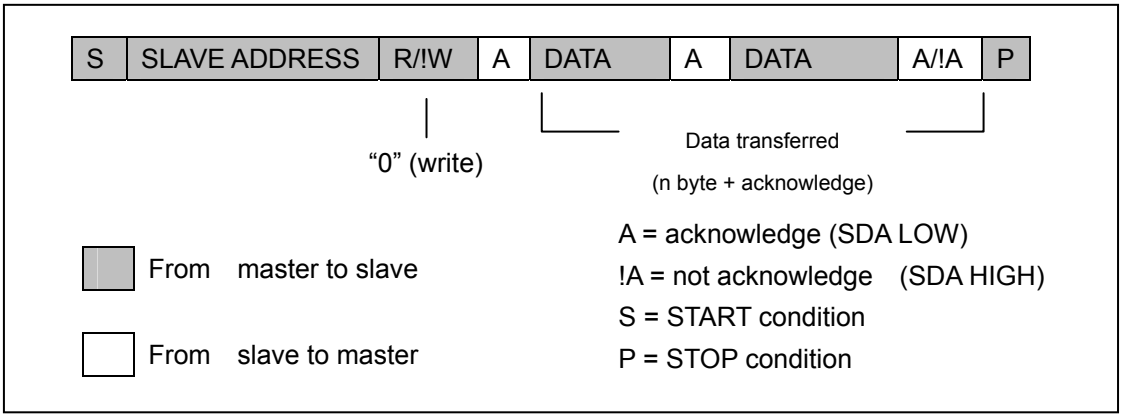


Figure 20-6 A Master-Transmitter Addresses a Slave Receiver with a 7-Bit Address

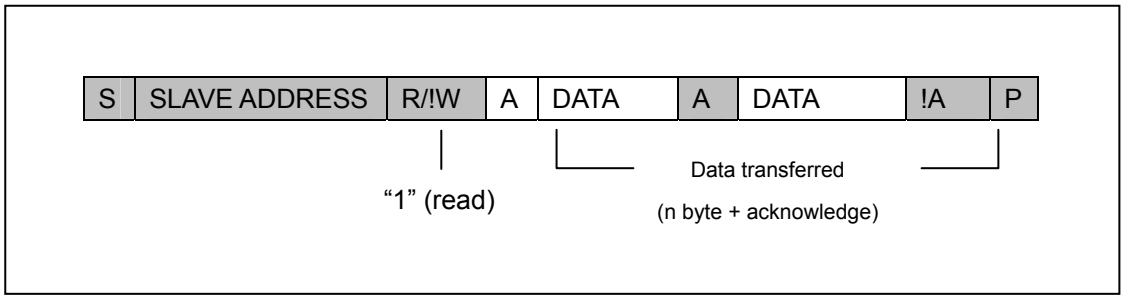


Figure 20-7 A Master Reads the Slave Immediately after the First Byte (Master-Receiver)

20.5 I2C Operation

20.5.1 I2C Initialization

Before transmitting and receiving data, set the I2CE bit in I2CCR to 1 to enable I2C operation and set I2CGR for proper serial clock frequency. Set the I2CE bit to 0 after transmitting or receiving data for low power dissipation.

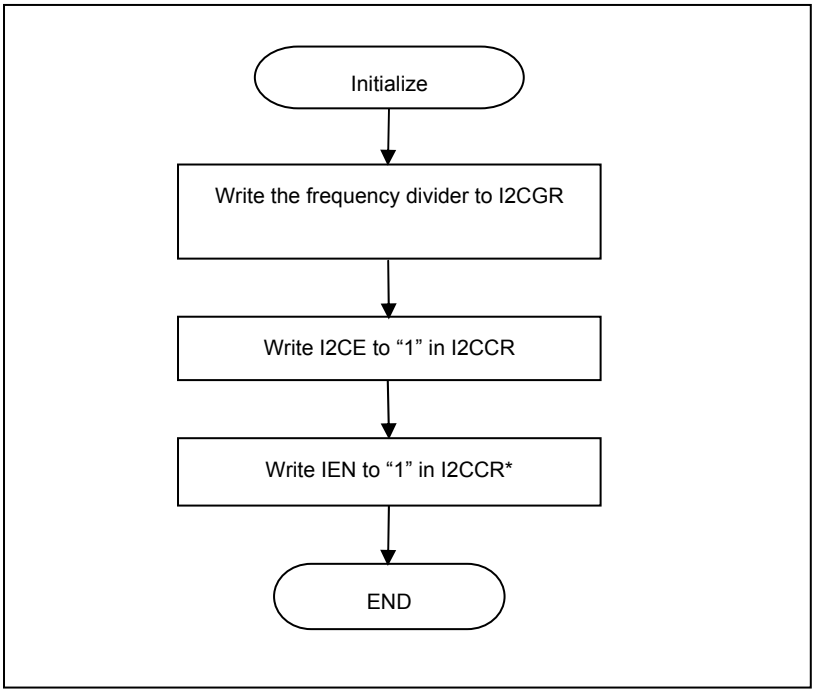


Figure 20-8 I2C Initialization

NOTE: This step is selectable.

20.5.2 Write Operation

Following figure illustrates the flow of a write operation.

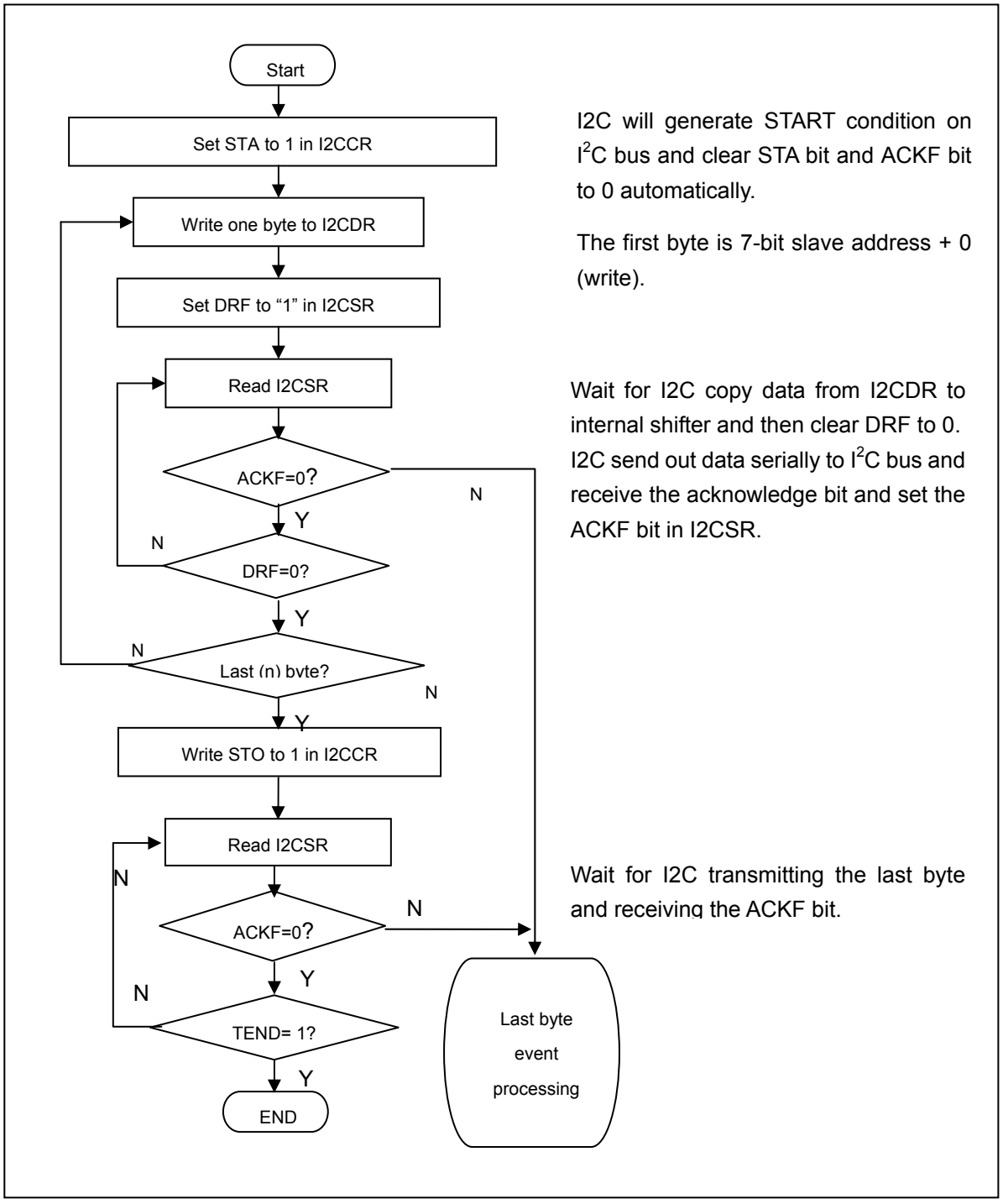


Figure 20-9 I2C Write Operation Flowchart

20.5.3 Read Operation

Following figure illustrates the flow of read operation.

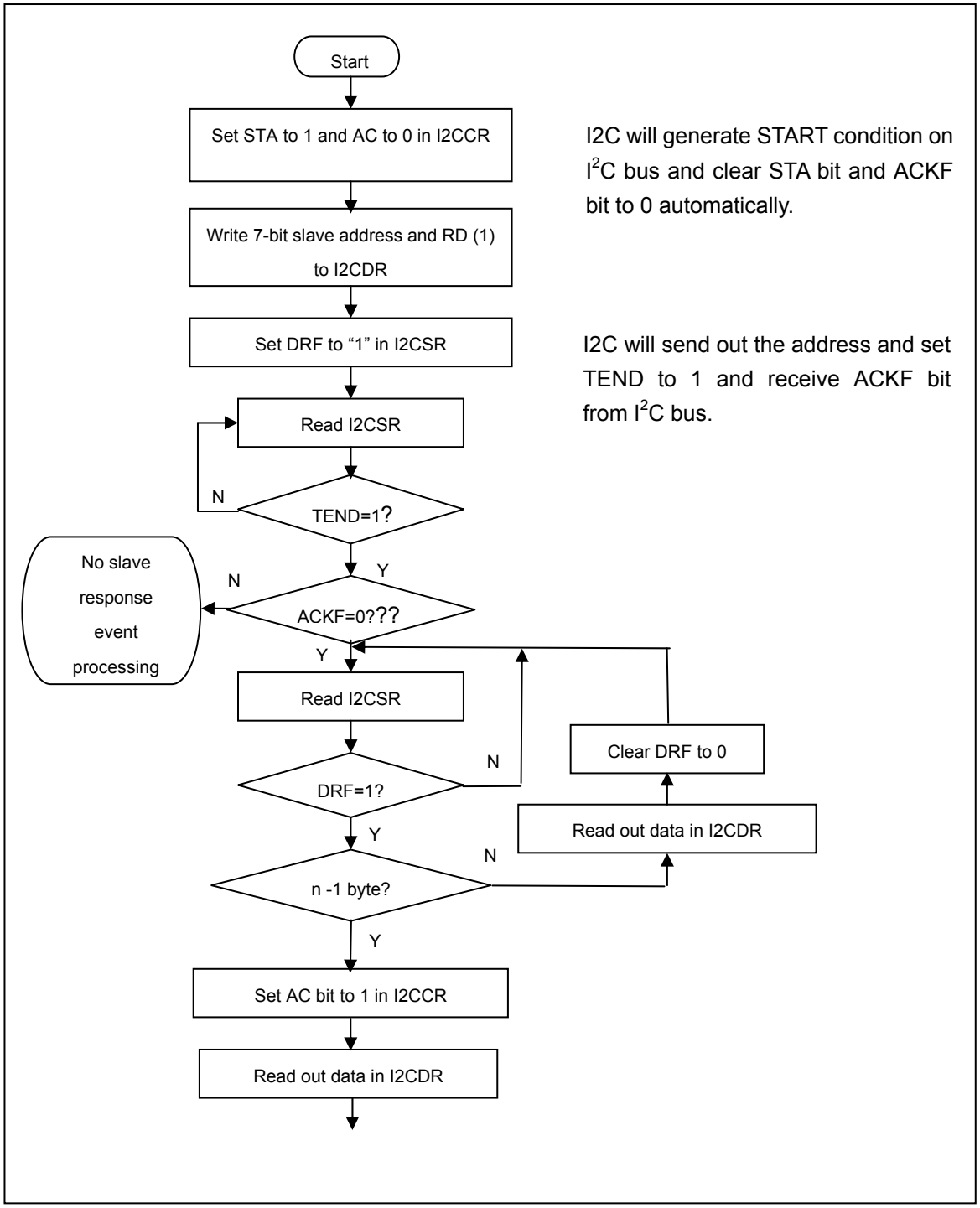


Figure 20-10 I2C Read Operation Flowchart

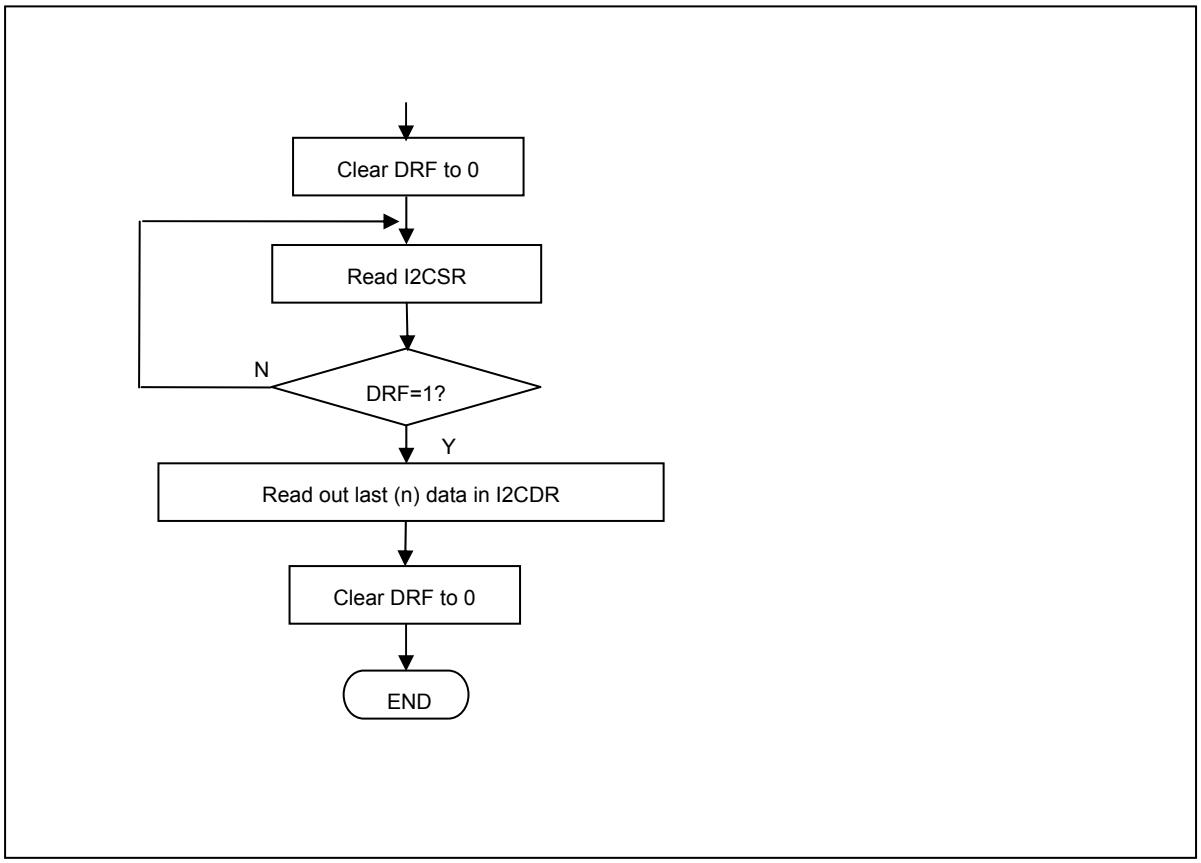


Figure 20-11 Read Operation Flowchart (cont.)

21 USB 2.0 Device Controller

21.1 Overview

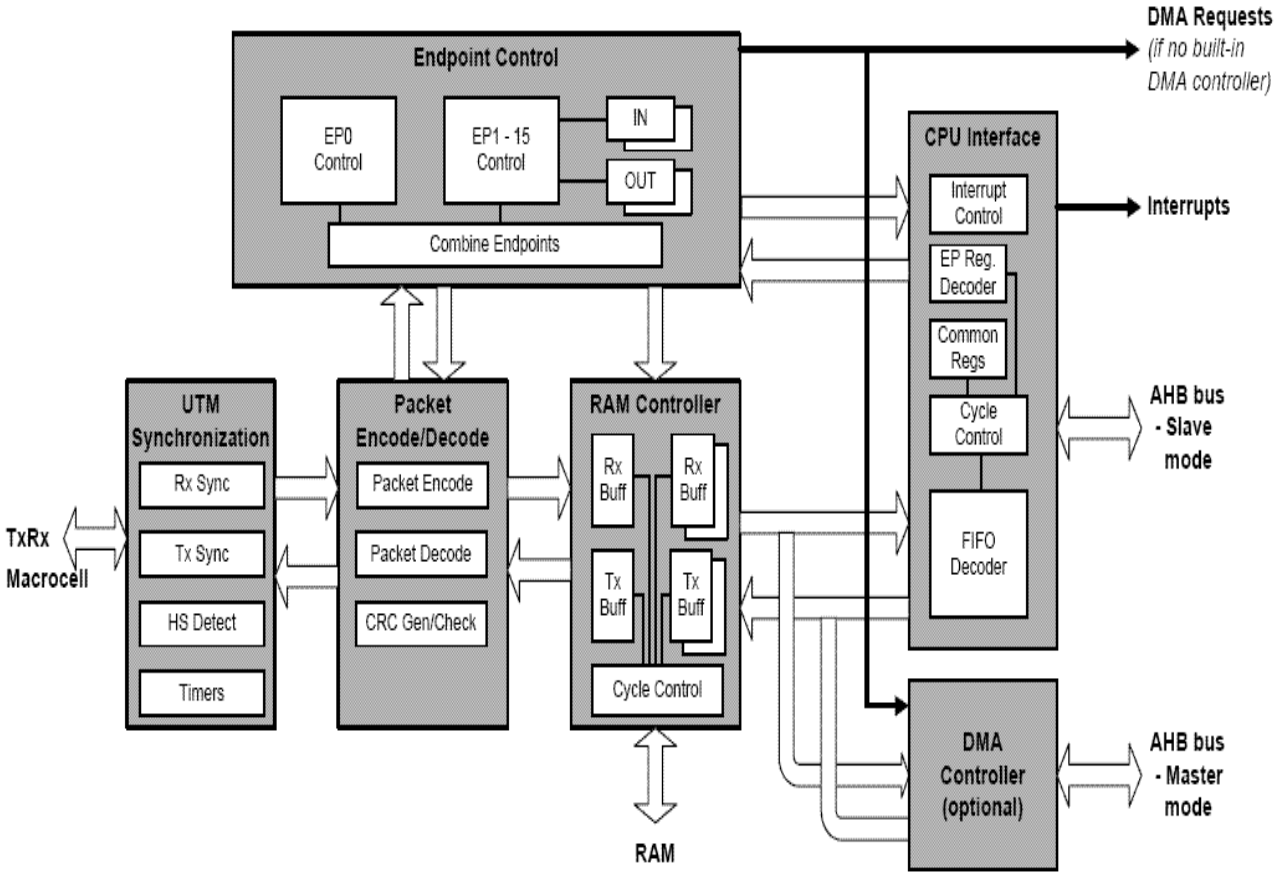
The USB 2.0 device controller core provides a USB function controller that has been certified compliant with the USB 2.0 specification for high/full-speed (480/12 Mbps) functions. The core has up to 3 IN endpoints and/or up to 2 OUT endpoints in addition to Endpoint 0.

21.2 Feature

- Support USB Full Speed (12Mb/sec) and High Speed (480Mb/sec)
- Support Control, BULK and Interrupt transfer type
- 3 IN Endpoint Number -- (including EP0, Control, BULKIN, Interrupt)
- 2 OUT Endpoint Number – (including EP0, Control, BULKOUT)
- Support DMA Engine
  - move data between system memory and USB without CPU intervene
- On-chip USB2.0 PHY
- Support Soft Connect/Disconnect
- Support Suspend and Resume Operation

21.3 Functional Description

21.3.1 Block Diagram



21.3.2 Block Description

The Block of USB2.0 device provides a USB 2.0 Transceiver Macrocell Interface to connect to an 8-bit high/full-speed transceiver. The design is also offered with a choice of high-level CPU interfaces. In one implementation, access to the FIFOs and the internal control/status registers is via a 16/32-bit through a 32-bit AHB-compatible interface.

The Block provides all the USB packet encoding, decoding, checking and handshaking – interrupting the CPU only when endpoint data has been successfully transferred.

21.3.2.1 UMTI SYNCHRONIZATION

The role of the UTM Synchronization block is to resynchronize between the transceiver macrocell 60MHz clock domain and the function controller's system clock HCLK, which drives the remainder of the core up to and including the CPU interface. This allows the rest of the Block to run without requiring any further synchronization. The block also performs the High-speed detection handshaking.

21.3.2.2 PACKET ENCODING/DECODING

The Packet Encode/Decode block generates headers for packets to be transmitted and decodes the headers on received packets. It also generates the CRC for packets to be transmitted and checks the CRC on received packets.

21.3.2.3 ENDPOINT CONTROLLERS

Two controller state machines are used: one for control transfers over Endpoint 0, and one for Bulk/Interrupt/Isochronous transactions over Endpoints 1 to 15.

21.3.2.4 CPUINTERFACE

The CPU Interface allows access to the control/status registers and the FIFOs for each endpoint. It also generates interrupts to the CPU when packets are successfully transmitted or received, and when the core enters Suspend mode or resumes from Suspend mode.

21.3.2.5 DMA CONTROLLER

The DMA Controller offer an AHB interface may be configured to include a multi-channel DMA controller. This DMA controller is configurable for up to 8 channels and is intended to promote efficient loading/unloading of the endpoint FIFOs. The DMA controller has its own block of control registers and its own interrupt controller. It supports two modes of operation and each channel can be independently programmed for operating mode.

21.3.2.6 RAM CONTROLLER

The RAM controller provides an interface to a single block of synchronous single-port RAM, which is used to buffer packets between the CPU and USB. It takes the FIFO pointers from the endpoint controllers, converts them to address pointers within the RAM block and generates the RAM access control signals.

21.3.2.7 BIT/ BYTE ORDERING

The Block is intrinsically little-endian, both in bit ordering within a byte and in byte ordering within words.

21.4 Register Description

21.4.1 Register Map

1 Common USB registers (addresses 00h to 0Fh).

These registers provide control and status for the entire function controller.

2 Indexed registers (addresses 10h to 1Fh).

These registers provide control and status for one endpoint. There is an InMaxP and InCSR register for each IN endpoint and an OutMaxP, OutCSR, and OutCount for each OUT endpoint (except for Endpoint 0 which has a reduced registered set: see below).

Only the registers for one IN endpoint and one OUT endpoint appear in the register map at any one time. The endpoints are selected by writing the endpoint number to the Index register. Therefore to access the registers for IN Endpoint 1 and OUT Endpoint 1, 1 must first be written to the Index register and then the control and status registers appear in the memory map.

3 FIFOs (addresses 20h to 3F/5Fh).

The FIFOs for each IN endpoint appear as a single 16-bit word (if a 16-bit CPU bus is configured) or as a 32-bit double word (if a 32-bit CPU bus is configured) consecutively in the memory map starting at address 20h. The FIFOs for each OUT endpoint also appear consecutively at the same set of addresses. A write to address 22h (24h if a 32-bit CPU bus is configured) results in the word being loaded into the FIFO for IN Endpoint 1. A read of address 22h (24h if a 32-bit CPU bus is configured) results in a word being unloaded from the FIFO for OUT Endpoint 1.

4 Additional Configuration registers (70h–7Fh).

Registers in this area of the memory map provide additional device status information.

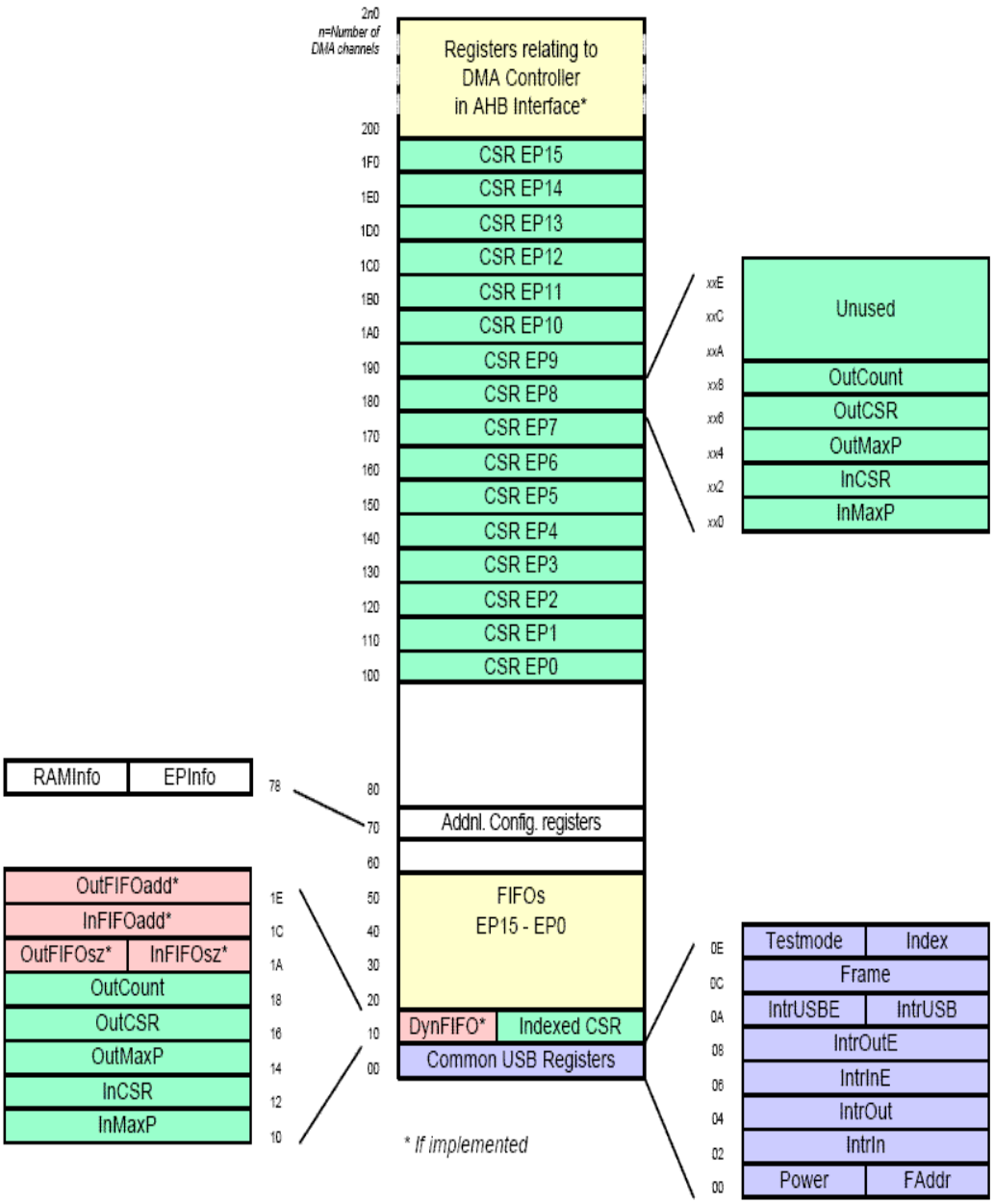
5 Non-Indexed Endpoint Control/Status registers (100h and above).

The registers available at 10h–1Fh, accessible independently of the setting of the Index register.
100h–10Fh EP0 registers;
110h–11Fh EP1 registers; 120h–12Fh EP2; and so on.

6 DMA Control Registers (200h and above).

These registers available at 200h and above.

21.4.2 Memory Map



21.4.3 Registers Summary

Common USB Registers				
Name	**RW**	**Reset Value**	**Address**	**Access Size**
FAddr	RW	0x00	0x13040000	8
Power	RW	0x20	0x13040001	8
IntrIn	R	0x0000	0x13040002	16
IntrOut	R	0x0000	0x13040004	16
IntrInE	RW	0xFFFF	0x13040006	16
IntrOutE	RW	0xFFFE	0x13040008	16
IntrUSB	R	0x0	0x1304000A	8
IntrUSBE	RW	0x6	0x1304000B	8
Frame	R	0x0000	0x1304000C	16
Index	RW	0x0	0x1304000E	8
Testmode	RW	0x00	0x1304000F	8

Indexed Registers				
Name	**RW**	**Reset Value**	**Address**	**Access Size**
InMaxP	RW	11/13/0x0000	0x13040010	16
CSR0	RW	0x00	0x13040012	8
InCSR	RW	0x0000	0x13040012	16
OutMaxP	RW	11/13/0x0000	0x13040014	16
OutCSR	RW	0x0000	0x13040016	16
Count0	R	0x00	0x13040018	8
OutCount	R	0x0000	0x13040018	16

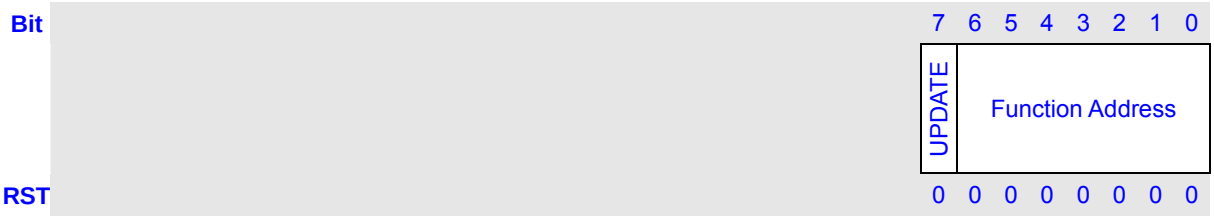
FIFOs				
Name	**RW**	**Reset Value**	**Address**	**Access Size**
FIFOx	RW	0x????????	0x130400(20 –5F)	32

Additional Configuration Registers				
Name	**RW**	**Reset Value**	**Address**	**Access Size**
EPInfo	R	0x??	0x13040078	8
RAMInfo	R	0x??	0x13040079	8

21.4.3.1 FADDR

FAddr is an 8-bit register that should be written with the function's 7-bit address (received through a SET_ADDRESS descriptor). It is then used for decoding the function address in subsequent token packets.

FADDR **0x13040000**



Bits	Name	Description	RW	
			CPU	USB
7	UPDATE	Set when FAddr is written. Cleared when the new address takes effect (at the end of the current transfer).	R	RC
6:0	Func Addr	The function address.	RW	R

This register should be written with the address value contained in the SET_ADDRESS standard device request (see Universal Serial Bus Specification Revision 2.0, Chapter 9), when it is received on Endpoint 0. The new address will not take effect immediately as the host will still be using the old address for the Status stage of the device request. The Block will continue to use the old address for decoding packets until the device request has completed. The status of the device request can be determined by reading bit 7 of this register. When a new address is written to this register, bit 7 will be automatically set. It will remain high until the device request has completed and will be cleared when the new address takes effect.

NOTE: While the firmware may write the new address to the FADDR register immediately it is received, it is recommended to leave this operation to the Status phase of the operation in case the host aborts the command. Otherwise confusion may arise.

21.4.3.2 POWER

Power is an 8-bit register that is used for controlling Suspend and Resume signaling, and high-speed operation.

POWER **0x13040001**

Bit	7	6	5	4	3	2	1	0
	ISO UP	VER	HS Enab	HS Mode	Reset	Resume	SUSPEND	ENA SUSP
RST	0	0	0	0	0	0	1	0

Bits	Name	Description	RW	
			CPU	USB
7	ISO UPDATE	When set by the CPU, the block will wait for an SOF token from the time InPktRdy is set before sending the packet. If an IN token is received before an SOF token, then a zero length data packet will be sent. **NOTE:** This bit only affects endpoints performing Isochronous transfers.	RW	R
6	VERSION	Version specific.	RW	R
5	HS ENab	When set by the CPU, the block will negotiate for high-speed mode when the device is reset by the hub. If not set, the device will only operate in Full-speed mode.	RW	R
4	HS Mode	This read-only bit is set when the block has successfully negotiated for High-speed mode.	R	RW
3	Reset	This read-only bit is set when Reset signaling has been detected on the bus (after 2.5μs of SE0). It is cleared when either HS negotiation has completed successfully or after 2.1ms of Reset signaling if HS negotiation fails.	R	RW
2	Resume	Set by the CPU to generate Resume signaling when the function is in Suspend mode. The CPU should clear this bit after 10 ms (a maximum of 15 ms) to end Resume signaling.	RW	R
1	Suspend Mode	This read-only bit is set when Suspend mode is entered. It is cleared when the CPU reads the interrupt register, or sets the Resume bit of this register.	R	SET
0	Enable SuspendM	Set by the CPU to enable the SUSPENDM signal.	RW	R

The **ISO Update** bit affects all IN Isochronous endpoints in the CORE. It is normally used as a method of ensuring a “clean” start-up of an IN Isochronous pipe. See the section on IN Isochronous Endpoints (Section 9) for more details on starting up IN Isochronous pipes.

The **HS Enab** bit can be used to disable high-speed operation. Normally the CORE will automatically

negotiate for highspeed operation, when it is reset, by sending a “chirp” to the hub. However if this bit is cleared then the block will not send any “chirps” to the hub so the function will remain in Full-speed mode, even when connected to a high-speed-capable USB.

The **HS Mode** bit can be used to determine whether the block is in High-speed mode or Full-speed mode. It will go high when the function has successfully negotiated for high-speed operation during a USB reset.

The **Reset** bit can be used to determine when reset signaling is present on the USB. It is taken high when Reset signaling is detected and remains high until the bus reverts to an idle state.
The Resume bit is used to force the block to generate Resume signaling on the USB to perform remote wake-up from Suspend mode. Once set high, it should be left high for approximately 10 ms (at least 1 ms and no more than 15 ms), then cleared.

The **Suspend Mode** bit is set by the core when Suspend mode is entered. It will be cleared when the IntrUSB register is read (as a result of receiving a Suspend interrupt). It will also be cleared if Suspend mode is left by setting the Resume bit to initiate a remote wake-up.

The **Enable SuspendM** bit is set to enable the SUSPENDM signal to put the UTM (and any other hardware which uses the SUSPENDM signal) into Suspend mode. If this bit is not set, Suspend mode will be detected as normal but the SUSPENDM signal will remain high so that the UTM does not go into its low-power mode.

21.4.3.3 INTRIN

IntrIn is a 16-bit read-only register that indicates which of the interrupts for IN Endpoints 1 – 15 are currently active. It also indicates whether the Endpoint 0 interrupt is currently active. **NOTE:** Bits relating to endpoints that have not been configured will always return 0. Note also that all active interrupts are cleared when this register is read.

INTRIN **0x13040002**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EP15IN	EP14IN	EP13IN	EP12IN	EP11IN	EP10IN	EP9IN	EP8N	EP7IN	EP6IN	EP5IN	EP4IN	EP3IN	EP2IN	EP1IN	EP0
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW	
			CPU	USB
15	EP15 IN	IN Endpoint 15 interrupt.	R	SET
14	EP14 IN	IN Endpoint 14 interrupt.	R	SET
13	EP13 IN	IN Endpoint 13 interrupt.	R	SET
12	EP12 IN	IN Endpoint 12 interrupt.	R	SET

11	EP11 IN	IN Endpoint 11 interrupt.	R	SET
10	EP10 IN	IN Endpoint 10 interrupt.	R	SET
9	EP9 IN	IN Endpoint 9 interrupt.	R	SET
8	EP8 IN	IN Endpoint 8 interrupt.	R	SET
7	EP7 IN	IN Endpoint 7 interrupt.	R	SET
6	EP7 IN	IN Endpoint 6 interrupt.	R	SET
5	EP5 IN	IN Endpoint 5 interrupt.	R	SET
4	EP4 IN	IN Endpoint 4 interrupt.	R	SET
3	EP3 IN	IN Endpoint 3 interrupt.	R	SET
2	EP2 IN	IN Endpoint 2 interrupt.	R	SET
1	EP1 IN	IN Endpoint 1 interrupt.	R	SET
0	EP0	Endpoint 0 interrupt.	R	SET

21.4.3.4 INTROUT

IntrOut is a 16-bit read-only register that indicates which of the interrupts for OUT Endpoints 1 – 15 are currently active. (Endpoint 0 uses a single interrupt, included in the IntrIn register.) **NOTE:** Bits relating to endpoints that have not been configured will always return 0. Note also that all active interrupts are cleared when this register is read.

INTROUT **0x13040004**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EP15OUT	EP14OUT	EP13OUT	EP12OUT	EP11OUT	EP10OUT	EP9OUT	EP8OUT	EP7OUT	EP6OUT	EP5OUT	EP4OUT	EP3OUT	EP2OUT	EP1OUT	Reserved
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW	
			CPU	USB
15	EP15 OUT	OUT Endpoint 15 interrupt.	R	SET
14	EP14 OUT	OUT Endpoint 14 interrupt.	R	SET
13	EP13 OUT	OUT Endpoint 13 interrupt.	R	SET
12	EP12 OUT	OUT Endpoint 12 interrupt.	R	SET
11	EP11 OUT	OUT Endpoint 11 interrupt.	R	SET
10	EP10 OUT	OUT Endpoint 10 interrupt.	R	SET
9	EP9 OUT	OUT Endpoint 9 interrupt.	R	SET
8	EP8 OUT	OUT Endpoint 8 interrupt.	R	SET
7	EP7 OUT	OUT Endpoint 7 interrupt.	R	SET
6	EP7 OUT	OUT Endpoint 6 interrupt.	R	SET
5	EP5 OUT	OUT Endpoint 5 interrupt.	R	SET
4	EP4 OUT	OUT Endpoint 4 interrupt.	R	SET
3	EP3 OUT	OUT Endpoint 3 interrupt.	R	SET

2	EP2 OUT	OUT Endpoint 2 interrupt.	R	SET
1	EP1 OUT	OUT Endpoint 1 interrupt.	R	SET
0	Reserved	Always returns 0.	R	R

21.4.3.5 INTRINE

IntrInE is a 16-bit register that provides interrupt enable bits for each of the interrupts in IntrIn. Where a bit is set to 1, MC_NINT will be asserted on the corresponding interrupt in the IntrIn register becoming set. Where a bit is set to 0, the interrupt in IntrIn is still set but MC_NINT is not asserted. On reset, D0 – D*n* are set to 1 where *n* is the number of IN Endpoints (in addition to Endpoint 0) that are included in the design, while the remaining bits are set to 0. ***NOTE:*** Bits relating to endpoints that have not been configured will always return 0.

INTRINE **0x13040006**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EP15INE	EP14INE	EP13INE	EP12INE	EP11INE	EP10INE	EP9INE	EP8NE	EP7INE	EP6INE	EP5INE	EP4INE	EP3INE	EP2INE	EP1INE	EP0E
RST	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description	RW	
			CPU	USB
15	EP15 INE	IN Endpoint 15 interrupt enable.	RW	R
14	EP14 INE	IN Endpoint 14 interrupt enable.	RW	R
13	EP13 INE	IN Endpoint 13 interrupt enable.	RW	R
12	EP12 INE	IN Endpoint 12 interrupt enable.	RW	R
11	EP11 INE	IN Endpoint 11 interrupt enable.	RW	R
10	EP10 INE	IN Endpoint 10 interrupt enable.	RW	R
9	EP9 INE	IN Endpoint 9 interrupt enable.	RW	R
8	EP8 INE	IN Endpoint 8 interrupt enable.	RW	R
7	EP7 INE	IN Endpoint 7 interrupt enable.	RW	R
6	EP7 INE	IN Endpoint 6 interrupt enable.	RW	R
5	EP5 INE	IN Endpoint 5 interrupt enable.	RW	R
4	EP4 INE	IN Endpoint 4 interrupt enable.	RW	R
3	EP3 INE	IN Endpoint 3 interrupt enable.	RW	R
2	EP2 INE	IN Endpoint 2 interrupt enable.	RW	R
1	EP1 INE	IN Endpoint 1 interrupt enable.	RW	R
0	EP0 E	Endpoint 0 interrupt enable.	RW	R

21.4.3.6 INTROUTE

IntrOutE is a 16-bit register that provides interrupt enable bits for each of the interrupts in IntrOut. Where a bit is set to 1, MC_NINT will be asserted on the corresponding interrupt in the IntrOut register becoming set. Where a bit is set to 0, the interrupt in IntrOut is still set but MC_NINT is not asserted. On reset, D1 – D*m* are set to 1 where *m* is the number of OUT Endpoints (in addition to Endpoint 0) that are included in the design, while the remaining bits are set to 0. ***NOTE:*** Bits relating to endpoints that have not been configured will always return 0.

INTROUTE **0x13040008**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EP15OUTE	EP14OUTE	EP13OUTE	EP12OUTE	EP11OUTE	EP10OUTE	EP9OUTE	EP8OUTE	EP7OUTE	EP6OUTE	EP5OUTE	EP4OUTE	EP3OUTE	EP2OUTE	EP1OUTE	Reserved
RST	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Bits	Name	Description	RW	
			CPU	USB
15	EP15 OUTE	OUT Endpoint 15 interrupt enable.	RW	R
14	EP14 OUTE	OUT Endpoint 14 interrupt enable.	RW	R
13	EP13 OUTE	OUT Endpoint 13 interrupt enable.	RW	R
12	EP12 OUTE	OUT Endpoint 12 interrupt enable.	RW	R
11	EP11 OUTE	OUT Endpoint 11 interrupt enable.	RW	R
10	EP10 OUTE	OUT Endpoint 10 interrupt enable.	RW	R
9	EP9 OUTE	OUT Endpoint 9 interrupt enable.	RW	R
8	EP8 OUTE	OUT Endpoint 8 interrupt enable.	RW	R
7	EP7 OUTE	OUT Endpoint 7 interrupt enable.	RW	R
6	EP7 OUTE	OUT Endpoint 6 interrupt enable.	RW	R
5	EP5 OUTE	OUT Endpoint 5 interrupt enable.	RW	R
4	EP4 OUTE	OUT Endpoint 4 interrupt enable.	RW	R
3	EP3 OUTE	OUT Endpoint 3 interrupt enable.	RW	R
2	EP2 OUTE	OUT Endpoint 2 interrupt enable.	RW	R
1	EP1 OUTE	OUT Endpoint 1 interrupt enable.	RW	R
0	Reserved	Always returns 0.	RW	R

21.4.3.7 INTRUSB

IntrUSB is a 4-bit read-only register that indicates which USB interrupts are currently active. ***NOTE:*** All active interrupts are cleared when this register is read.

INTRUSB					0x1304000A
Bit					3 2 1 0
					SOF Reset Resume SUSPEND
RST					0 0 0 0

Bits	Name	Description	RW	
			CPU	USB
3	SOF	Set at the start of each frame.	R	SET
2	Reset	Set when reset signaling is detected on the bus.	R	SET
1	Resume	Set when resume signaling is detected on the bus while the CORE is in Suspend mode.	R	SET
0	Suspend	Set when suspend signaling is detected on the bus.	R	SET

21.4.3.8 INTRUSBE

IntrUSBE is a 4-bit register that provides interrupt enable bits for each of the interrupts in IntrUSB.

INTRUSBE					0x1304000B
Bit					3 2 1 0
					SOF Reset Resume SUSPEND
RST					0 0 0 0

Bits	Name	Description	RW	
			CPU	USB
3	SOF	At the start of each frame interrupt enable.	RW	R
2	Reset	Reset signaling is detected on the bus interrupt enable.	RW	R
1	Resume	Resume signaling is detected on the bus while the core is in Suspend mode interrupt enable.	RW	R
0	Suspend	Suspend signaling is detected on the bus interrupt enable.	RW	R

21.4.3.9 FRAME

Frame is a 16-bit read-only register that holds the last received frame number.

FRAME 0x1304000C

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved					FRAME										
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW	
			CPU	USB
15:10	Reserved	Always returns 0.	R	W
10:0	Reset	FRAME.	R	W

21.4.3.10 INDEX

Index is a 4-bit register that determines which endpoint control/status registers are accessed at addresses 10h to 19h.

INDEX 0x1304000E

Bit	3	2	1	0
	INDEX			
RST	0	0	0	0

Bits	Name	Description	RW	
			CPU	USB
3:0	INDEX	Selected Endpoint.	RW	R

Each IN endpoint and each OUT endpoint have their own set of control/status registers. Only one set of IN control/status and one set of OUT control/status registers appear in the memory map at any one time. Before accessing an endpoint's control/status registers, the endpoint number should be written to the Index register to ensure that the correct control/status registers appear in the memory map.

21.4.3.11 TESTMODE

Testmode is a 6-bit register that is primarily used to put the CORE into one of the four test modes described in the USB 2.0 specification. It is not used in normal operation.

TESTMODE **0x1304000F**

Bit	7	6	5	4	3	2	1	0
	Reserved		FORCE_FS	FORCE_HS	TEST_Packet	TEST_K	TEST_J	TEST_SE0_NAK
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW	
			CPU	USB
5	FORECE_FS	The CPU sets this bit to force the CORE into Full-speed mode when it receives a USB reset.	RW	R
4	FORCE_HS	The CPU sets this bit to force the CORE into High-speed mode when it receives a USB reset.	RW	R
3	TEST_PACKET	The CPU sets this bit to enter the Test_Packet test mode. In this mode, the core – in highspeed mode – repetitively transmits on the bus a 53-byte test packet, the form of which is defined in Section 21.11.4. **NOTE:** The 53-byte test packet must be loaded into the Endpoint 0 FIFO before the test mode is entered.	RW	R
2	TEST_K	The CPU sets this bit to enter the Test_K test mode. In this mode, the CORE – in high-speed mode – transmits a continuous K on the bus.	RW	R
1	TEST_J	The CPU sets this bit to enter the Test_J test mode. In this mode, the CORE – in high-speed mode – transmits a continuous J on the bus.	RW	R
0	TEST_SE0_NAK	The CPU sets this bit to enter the Test_SE0_NAK test mode. In this mode, the CORE remains in high-speed mode and responds to any valid IN token with a NAK.	RW	R

21.4.3.12 CSR0

CSR0 is an 8-bit register that provides control and status bits for Endpoint 0. **NOTE:** Users should be aware that the value returned when the register is read reflects the status attained e.g. as a result of writing to the register.

CSR0 **0x13040012**

Bit	7	6	5	4	3	2	1	0
	ServicedSetupEnd	ServicedoutPktRdy	SendStall	SetupEnd	DataEnd	SentStall	InPktRdy	OutPktRdy
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW	
			CPU	USB
7	ServicedSetupEnd	The CPU writes a 1 to this bit to clear the SetupEnd bit. It is cleared automatically.	SET	R
6	ServicedOutPktRdy	The CPU writes a 1 to this bit to clear the OutPktRdy bit. It is cleared automatically.	SET	R
5	SendStall	The CPU writes a 1 to this bit to terminate the current transaction. The STALL handshake will be transmitted and then this bit will be cleared automatically. **NOTE:** This behavior differs from that of the SendStall bits associated with additional IN/OUT endpoints which need to be cleared by the CPU.	SET	R
4	SetupEnd	This bit will be set when a control transaction ends before the DataEnd bit has been set. An interrupt will be generated and the FIFO flushed at this time. The bit is cleared by the CPU writing a 1 to the ServicedSetupEnd bit.	R	SET
3	DataEnd	The CPU sets this bit: 1 When setting InPktRdy for the last data packet. 2 When clearing OutPktRdy after unloading the last data packet. 3 When setting InPktRdy for a zero length data packet. It is cleared automatically.	SET	SET

2	SentStall	This bit is set when a STALL handshake is transmitted. The CPU should clear this bit.	RC	R
1	InPktRdy	The CPU sets this bit after loading a data packet into the FIFO. It is cleared automatically when the data packet has been transmitted. An interrupt is generated when the bit is cleared.	RS	R
0	OutPktRdy	This bit is set when a data packet has been received. An interrupt is generated when this bit is set. The CPU clears this bit by setting the ServicedOutPktRdy bit.	R	SET

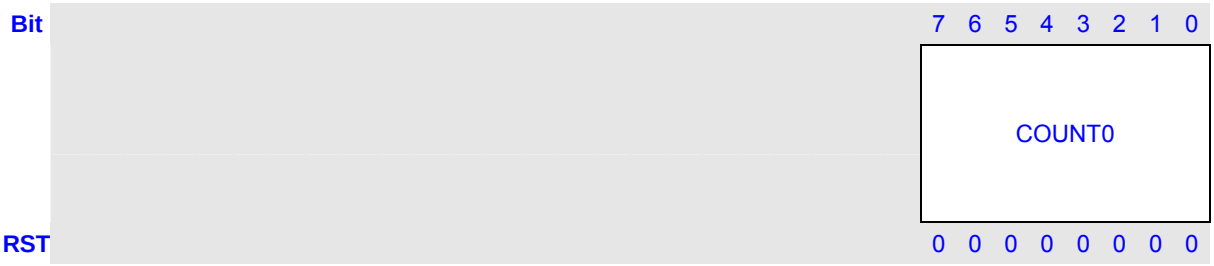
CSR0 appears in the memory map at address 12h when the Index register is set to 0. It is used for all control/status of Endpoint 0. For details of how to service device requests to Endpoint 0, see Section 6: 'Endpoint 0 Handling'.

21.4.3.13 COUNT0

Count0 is a 7-bit read-only register that indicates the number of received data bytes in the Endpoint 0 FIFO.

NOTE: The value returned changes as the contents of the FIFO change and is only valid while OutPktRdy (CSR0.D0) is set.

COUNT0 **0x13040018**



Bits	Name	Description	RW	
			CPU	USB
7:0	COUNT0	Endpoint 0 OUT Count.	R	W

21.4.3.14 INMAXP

The InMaxP register defines the maximum amount of data that can be transferred through the selected IN endpoint in a single operation. There is an InMaxP register for each IN endpoint (except Endpoint 0).

INMAXP																0x13040010
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	m-1					Maximum Payload/transaction										
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits 10:0 define (in bytes) the maximum payload transmitted in a single transaction. The value set can be up to 1024 bytes but is subject to the constraints placed by the USB Specification on packet sizes for Bulk, Interrupt and Isochronous transfers in Fullspeed and High-speed operations.

Where the option of High-bandwidth Isochronous endpoints or of packet splitting on Bulk endpoints has been taken when the core is configured, the register includes either 2 or 5 further bits that define a multiplier *m* which is equal to one more than the value recorded.

In the case of Bulk endpoints with the packet splitting option enabled, the multiplier *m* can be up to 32 and defines the maximum number of 'USB' packets (i.e. packets for transmission over the USB) of the specified payload into which a single data packet placed in the FIFO should be split, prior to transfer. (If the packet splitting option is not enabled, D15–D13 is not implemented and D12–D11(if included) is ignored.) **NOTE:** The data packet is required to be an exact multiple of the payload specified by bits 10:0, which is itself required to be either 8, 16, 32, 64 or (in the case of High Speed transfers) 512 bytes.

For Isochronous endpoints operating in High-Speed mode and with the High-bandwidth option enabled, *m* may only be either 2 or 3 (corresponding to bit 11 set or bit 12 set, respectively) and it specifies the maximum number of such transactions that can take place in a single microframe. If either bit 11 or bit 12 is non-zero, the CORE will automatically split any data packet written to the FIFO into up to 2 or 3 'USB' packets, each containing the specified payload (or less). The maximum payload for each transaction is 1024 bytes, so this allows up to 3072 bytes to be transmitted in each microframe. (For Isochronous transfers in Fullspeed mode or if High-bandwidth is not enabled, bits 11 and 12 are ignored.)

The value written to bits 10:0 (multiplied by *m* in the case of high-bandwidth Isochronous transfers) must match the value given in the *wMaxPacketSize* field of the Standard Endpoint Descriptor for the associated endpoint (see *USB Specification* Revision 2.0, Chapter 9). A mismatch could cause unexpected results.

The total amount of data represented by the value written to this register (specified payload × *m*)

must not exceed the FIFO size for the IN endpoint, and should not exceed half the FIFO size if double-buffering is required.

If this register is changed after packets have been sent from the endpoint, the IN endpoint FIFO should be completely flushed (using the FlushFIFO bit in InCSR) after writing the new value to this register.

21.4.3.15 INCSR

InCSR is a 16-bit register that provides control and status bits for IN transactions through the currently-selected endpoint. There is an InCSR register for each IN endpoint (not including Endpoint 0).

NOTE: Users should be aware that the value returned when the register is read reflects the status attained e.g. as a result of writing to the register.

INCSR **0x13040012**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	AUTOSET	ISO	Mode	DMAReqEnab	FrcDataTog	DMAREQMode	Reserved		IncompTx	ClrDataTog	SentStall	Sendstall	FlushFifo	underrun	FIFONempty	InPktRdy
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW	
			CPU	USB
15	AutoSet	If the CPU sets this bit, InPktRdy will be automatically set when data of the maximum packet size (value in InMaxP) is loaded into the IN FIFO. If a packet of less than the maximum packet size is loaded, InPktRdy will have to be set manually. **NOTE:** Should not be set for high-bandwidth Isochronous endpoints.	RW	R
14	ISO	The CPU sets this bit to enable the IN endpoint for Isochronous transfers (ISO mode), and clears it to enable the IN endpoint for Bulk/Interrupt transfers.	RW	R
13	Mode	The CPU sets this bit to enable the endpoint direction as IN, and clears it to enable the endpoint direction as OUT. **NOTE:** Only valid where the endpoint FIFO is used for both IN and OUT transactions, otherwise ignored.	RW	R
12	DMAReqEnab	The CPU sets this bit to enable the DMA request for the IN endpoint.	RW	R
11	FrcDataTog	The CPU sets this bit to force the endpoint's IN data	RW	R

		toggle to switch after each data packet is sent regardless of whether an ACK was received. This can be used by Interrupt IN endpoints that are used to communicate rate feedback for Isochronous endpoints.		
10	DMAReqMode	The CPU sets this bit to select DMA Request Mode 1 and clears it to select DMA Request Mode 0. **NOTE:** This bit must not be cleared either before or in the same cycle as the above DMAReqEnab bit is cleared.	RW	R
9:8	–	Unused, always return 0.	R	R
7	IncompTx	When the endpoint is being used for high-bandwidth Isochronous transfers, this bit is set to indicate where a large packet has been split into 2 or 3 packets for transmission but insufficient IN tokens have been received to send all the parts. The remainder of the current packet is then flushed from the FIFO (but any second packet in the FIFO will remain). **NOTE:** In anything other than a high-bandwidth Isochronous transfer, this bit will always return 0.	RC	SET
6	ClrDataTog	The CPU writes a 1 to this bit to reset the endpoint IN data toggle to 0.	SET	RC
5	SentStall	This bit is set when a STALL handshake is transmitted. The FIFO is flushed and the InPktRdy bit is cleared (see below). The CPU should clear this bit.	RC	SET
4	SendStall	The CPU writes a 1 to this bit to issue a STALL handshake to an IN token. The CPU clears this bit to terminate the stall condition. **NOTE:** This bit has no effect where the endpoint is being used for Isochronous transfers.	RW	R
3	FlushFIFO	The CPU writes a 1 to this bit to flush the next packet to be transmitted from the endpoint IN FIFO. The FIFO pointer is reset and the InPktRdy bit (below) is cleared. May be set simultaneously with InPktRdy to abort the packet that has just been loaded into the FIFO. **NOTE:** (i) FlushFIFO should only be set when InPktRdy is set (at other times, it may cause data corruption). (ii) If the FIFO contains two packets, FlushFIFO will need to be set twice to completely clear the FIFO.	SET	R
2	**UnderRun**	In ISO mode, this bit is set when a zero length data packet is sent after receiving an IN token with the InPktRdy bit not set. In Bulk/Interrupt mode, this bit is set when a NAK is returned in response to an IN token.	RC	SET

		The CPU should clear this bit.		
1	**FIFONotEmpty**	This bit is set when there is at least 1 packet in the IN FIFO.	RC	SET
0	**InPktRdy**	The CPU sets this bit after loading a data packet into the FIFO. It is cleared automatically when a data packet has been transmitted. If the FIFO is double-buffered, it is also automatically cleared when there is space for a second packet in the FIFO (see Section 8.1.2). An interrupt is generated (if enabled) when the bit is cleared (suppressed by the built-in DMA controller in DMA Mode 1).	RS	CLEAR

21.4.3.16 OUTMAXP

The OutMaxP register defines the maximum amount of data that can be transferred through the selected OUT endpoint in a single operation. There is an OutMaxP register for each OUT endpoint (except Endpoint 0).

OUTMAXP **0x13040014**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	m-1					Maximum Payload/transaction										
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits 10:0 define (in bytes) the maximum payload transmitted in a single transaction. The value set can be up to 1024 bytes but is subject to the constraints placed by the USB Specification on packet sizes for Bulk, Interrupt and Isochronous transfers in Fullspeed and High-speed operations.

Where the option of High-bandwidth Isochronous endpoints or of combining Bulk packets has been taken when the core is configured, the register includes either 2 or 5 further bits that define a multiplier *m* which is equal to one more than the value recorded.

For Bulk endpoints with the packet combining option enabled, the multiplier *m* can be up to 32 and defines the number of USB packets of the specified payload which are to be combined into a single data packet within the FIFO. (If the packet splitting option is not enabled, D15–D13 is not implemented and D12–D11 (if included) is ignored.)

For Isochronous endpoints operating in High-Speed mode and with the High-bandwidth option enabled, *m* may only be either 2 or 3 (corresponding to bit 11 set or bit 12 set, respectively) and it specifies the maximum number of such transactions that can take place in a single microframe. If either bit 11 or bit 12 is non-zero, the CORE will automatically combine the separate USB packets received in any microframe into a single packet within the OUT FIFO. The maximum payload for each

transaction is 1024 bytes, so this allows up to 3072 bytes to be received in each microframe. (For Isochronous transfers in Full-speed mode or if High-bandwidth is not enabled, bits 11 and 12 are ignored.)
The value written to bits 10:0 (multiplied by *m* in the case of high-bandwidth Isochronous transfers) must match the value given in the *wMaxPacketSize* field of the Standard Endpoint Descriptor for the associated endpoint (see *USB Specification* Revision 2.0, Chapter 9). A mismatch could cause unexpected results.

The total amount of data represented by the value written to this register (specified payload $\times$ *m*) must not exceed the FIFO size for the OUT endpoint, and should not exceed half the FIFO size if double-buffering is required.

21.4.3.17 OUTCSR

OutCSR is a 16-bit register that provides control and status bits for OUT transactions through the currently-selected endpoint. It is reset to 0. ***NOTE:*** Users should be aware that the value returned when the register is read reflects the status attained e.g. as a result of writing to the register.

OUTCSR **0x13040016**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	AUTOClear	ISO	DMAReqEnab	DisNyet/PID ERR	DMAReqMode			IncompRx	ClrDataTog	SentStall	SendStall	FlushFIFO	DataError	OverRun	FIFOFull	OutPktRdy
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW	
			CPU	USB
15	AutoClear	If the CPU sets this bit then the OutPktRdy bit will be automatically cleared when a packet of OutMaxP bytes has been unloaded from the OUT FIFO. When packets of less than the maximum packet size are unloaded, OutPktRdy will have to be cleared manually. **NOTE:** Should not be set for high-bandwidth Isochronous endpoints.	RW	R
14	ISO	The CPU sets this bit to enable the OUT endpoint for Isochronous transfers, and clears it to enable the OUT endpoint for Bulk/Interrupt transfers.	RW	R
13	DMAReqEnab	The CPU sets this bit to enable the DMA request for the OUT endpoint.	RW	R
12	DisNyet PID Error	**Bulk/Interrupt Transactions: The CPU sets this bit to disable the sending of NYET handshakes. When set,**	RW	RW

		all successfully received OUT packets are ACK'd including at the point at which the FIFO becomes full. NOTE: This bit only has any effect in High-speed mode, in which mode it should be set for all Interrupt endpoints. ISO Transactions: The core sets this bit to indicate a PID error in the received packet.		
11	DMAReqMode	Two modes of DMA Request operation are supported: DMA Request Mode 0 in which a DMA request is generated for all received packets, together with an interrupt (if enabled); and DMA Request Mode 1 in which a DMA request (but no interrupt) is generated for OUT packets of size OutMaxP bytes and an interrupt (but no DMA request) is generated for OUT packets of any other size. The CPU sets this bit to select DMA Request Mode 1 and clears this bit to select DMA Request Mode 0.	RW	R
10:9	–	Unused, always return 0.	R	R
8	IncompRx	This bit is set in a high-bandwidth Isochronous transfer if the packet in the OUT FIFO is incomplete because parts of the data were not received. It is cleared when OutPktRdy is cleared. **NOTE:** In anything other than a high-bandwidth Isochronous transfer, this bit will always return 0.	RC	SET
7	ClrDataTog	The CPU writes a 1 to this bit to reset the endpoint data toggle to 0.	SET	RC
6	SentStall	This bit is set when a STALL handshake is transmitted. The CPU should clear this bit.	RC	SET
5	SendStall	The CPU writes a 1 to this bit to issue a STALL handshake to a DATA packet. The CPU clears this bit to terminate the stall condition. **NOTE:** This bit has no effect where the endpoint is being used for Isochronous transfers.	RW	R
4	FlushFIFO	The CPU writes a 1 to this bit to flush the next packet to be read from the endpoint OUT FIFO. The FIFO pointer is reset and the OutPktRdy bit (below) is cleared. **NOTE:** FlushFIFO should only be used when OutPktRdy is set. At other times, it may cause data to be corrupted. Note also that, if the FIFO is double-buffered, FlushFIFO may need to be set twice to completely clear the FIFO.	SET	R
3	DataError	This bit is set at the same time that OutPktRdy is set if the data packet has a CRC error. It is cleared when	R	SET

		OutPktRdy is cleared. **NOTE:** This bit is only valid when the endpoint is operating in ISO mode. In Bulk mode, it always returns zero.		
2	OverRun	This bit is set if an OUT packet arrives while FIFOFull is set i.e. the OUT packet cannot be loaded into the OUT FIFO. The CPU should clear this bit. **NOTE:** This bit is only valid when the endpoint is operating in ISO mode. In Bulk mode, it always returns zero.	RC	SET
1	FIFOFull	This bit is set when no more packets can be loaded into the OUT FIFO.	R	SET
0	OutPktRdy	This bit is set when a data packet has been received. The CPU should clear this bit when the packet has been unloaded from the OUT FIFO. An interrupt is generated (if enabled) when the bit is set.	RC	SET

21.4.3.18 OUTCOUNT

OutCount is a 13-bit read-only register that holds the number of received data bytes in the packet in the OUT FIFO. ***NOTE:*** The value returned changes as the contents of the FIFO change and is only valid while OutPktRdy (OutCSR.D0) is set.

OUTCOUNT **0x13040018**

Bit	12	11	10	9	8	7	6	5	4	3	2	1	0
	OUTCOUNT												
RST	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW	
			CPU	USB
12:0	OUTCOUNT	Endpoint OUT Count.	R	W

21.4.3.19 FIFOx (Addresses 20h – XXh)

This address range provides 16 addresses for CPU access to the FIFOs for each endpoint. Writing to these addresses loads data into the IN FIFO for the corresponding endpoint. Reading from these addresses unloads data from the OUT FIFO for the corresponding endpoint.

If the CPU bus is 16-bit, the address range is 20h – 3Fh and the FIFOs are located on 16-bit word boundaries (Endpoint 0 at 20h, Endpoint 1 at 22h ... Endpoint 15 at 3Eh). If the CPU bus is 32-bit, the address range is 20h – 5Fh and the FIFOs are located on 32-bit double-word boundaries (Endpoint 0

at 20h, Endpoint 1 at 24h ... Endpoint 15 at 5Ch).

NOTE: Transfers to and from FIFOs may be 8-bit, 16-bit, 24-bit or 32-bit as required, and any combination of access is allowed provided the data accessed is contiguous. However, all the transfers associated with one packet must be of the same width so that the data is consistently byte-, word- or double-word-aligned. The last transfer may however contain fewer bytes than the previous transfers in order to complete an odd-byte or odd-word transfer.

21.4.3.20 EPINFO

This 8-bit read-only register allows read-back of the number of IN and OUT endpoints included in the design.

EPINFO **0x13040078**

Bit	7	6	5	4	3	2	1	0
	OutEndPoint				InEndPoint			
RST	?	?	?	?	?	?	?	?

Bits	Name	Description	RW	
			CPU	USB
7:4	OutEndPoint	The number of OUT endpoints implemented in the design.	R	R
3:0	InEndPoint	The number of IN endpoints implemented in the design.	R	R

21.4.3.21 RAMINFO

This 8-bit read-only register provides information about the width of the RAM and the number of DMA channels associated with the built-in DMA controller (where implemented).

RAMINFO **0x13040079**

Bit	7	6	5	4	3	2	1	0
	DMAChans				RAMBITS			
RST	?	?	?	?	?	?	?	?

Bits	Name	Description	RW	
			CPU	USB
7:4	DMAChans	The number of DMA channels implemented in the design.	R	R
3:0	RAMBITS	The width of the RAM address bus – 1.	R	R

21.5 Programming Scheme

This and the following sections look at the actions that the device controlling the CORE core will need to perform and at the aspects of the operation of the core that affect this.

Throughout this discussion, the controlling device is assumed to be a microcontroller running some firmware but it could be a customized hard-wired logic block.

21.5.1 SOFT CONNECT/DISCONNECT

The core can be configured to allow the connection of the CORE to the USB to be controlled by software.

When the Soft Connect/Disconnect option is selected, the UTMIcompliant PHY used alongside the core can be switched between normal mode and non-driving mode by setting/clearing bit 6 of the Power register (which is then identified as the Soft Conn bit).

When the Soft Conn bit is set to 1, the PHY is placed in its normal mode and the D+/D- lines of the USB bus are enabled. At the same time, the core is placed in 'Powered' state, in which it will not respond to any USB signaling except a USB reset.

When this feature is enabled and the Soft Conn bit is zero, the PHY is put into non-driving mode, D+ and D- are tri-stated and the core appears to the host CPU as if it has been disconnected.

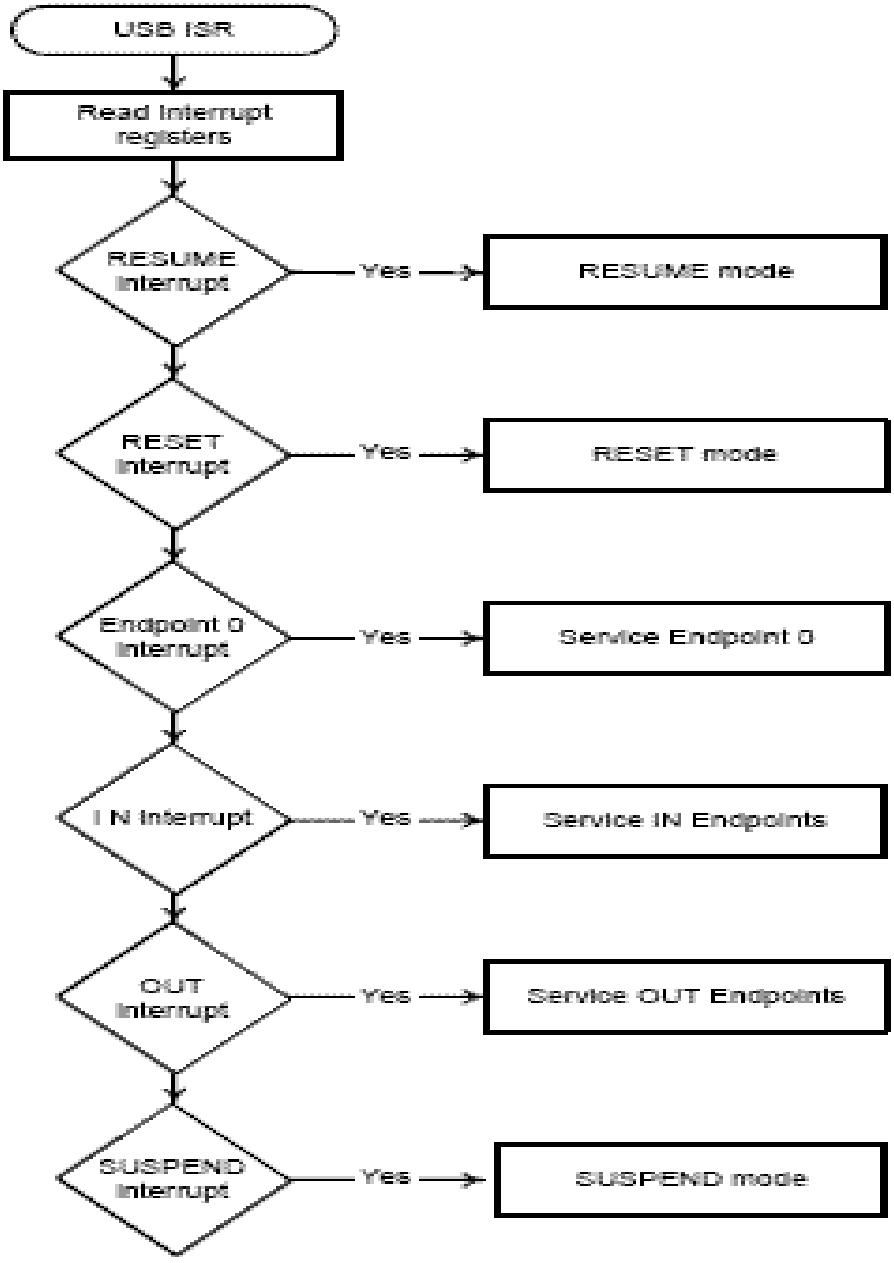
After a hardware reset (NRST = 0), Soft Conn is cleared to 0. The core will therefore appear disconnected until the software has set Soft Conn to 1. The application software can then choose when to set the PHY into its normal mode. Systems with a lengthy initialization procedure may use this to ensure that initialization is complete and the system is ready to perform enumeration before connecting to the USB.

Once the Soft Conn bit has been set to 1, the software can also simulate a disconnect by clearing this bit to 0.

21.5.2 USB INTERRUPT HANDLING

When the CPU is interrupted with a USB interrupt, it needs to read the interrupt status register to determine which endpoint(s) have caused the interrupt and jump to the appropriate routine. If multiple endpoints have caused the interrupt, Endpoint 0 should be serviced first, followed by the other endpoints. The Suspend interrupt should be serviced last.

A flowchart for the USB Interrupt Service Routine is given in as follows:



21.6 USB RESET

When a reset condition is detected on the USB, the CORE performs the following actions:

1 Sets FAddr to 0.
2 Sets Index to 0.
3 Flushes all endpoint FIFOs.
4 Clears all control/status registers.
5 Enables all endpoint interrupts.
6 Generates a Reset interrupt.

If the HS Enab bit in the Power register (D5) was set, the CORE also tries to negotiate for high-speed operation. Whether high-speed operation is selected is indicated by HS Mode bit (Power.D4).

When the software receives a Reset interrupt, it should close any open pipes and wait for bus enumeration to begin.

21.7 SUSPEND/RESUME

When the CORE has no activity on the USB for 3 ms, it will go into Suspend mode. It will also generate a Suspend interrupt (if enabled). At this point, the CORE can then be left active or the application may arrange to disable the USBHSFC by stopping its clock.

The USB may exit Suspend mode by sending Resume signaling on the bus. Alternatively software may perform “Remote wakeup”. How the CORE will respond depends on whether it has been left active or inactive during the suspend.

21.7.1 ACTIVE DURING SUSPEND

When the CORE goes into Suspend mode, the UTM will also be put into Suspend mode by the SUSPENDM line if the Enable SuspendM bit in the Power register (D0) is set. When the CORE remains active, however, it can detect when Resume signaling occurs on the USB. It will then bring the UTM out of Suspend mode and generate a Resume interrupt.

21.7.2 INACTIVE DURING SUSPEND

When the Suspend interrupt described above is received, the software may disable the CORE stopping its clock (this must be done by some external means). However, the CORE will I not then be able to detect Resume signaling on the USB.

As a result, external hardware will be needed to detect Resume signaling (by monitoring the LINESTATE lines from the UTM), o that the clock to the CORE can be restarted when this occurs. Appropriate gates could be added to the system design, or example, by specifying that an active high, asynchronous wake-up event is generated when the transceiver is in Suspend mode (SUSPENDM low) and either a K state (linestate == 2'b10 (resume)) or an SE0 (linestate == 2'b00 (reset)) is detected.

21.7.3 REMOTE WAKEUP

If the CORE is in Suspend mode and the software wants to initiate a remote wakeup, it should write to the Power register to set the Resume bit (D2) to 1. (If the clock to the CORE has been stopped, it will need to be restarted before this write can occur.)

The software should leave this bit set for approximately 10 ms (minimum of 2 ms, a maximum of 15 ms) then reset it to 0. By this time the hub should have taken over driving Resume signaling on the USB.

NOTE: No Resume interrupt will be generated when the software initiates a remote wakeup.

21.8 ENDPOINT 0 HANDLING

Endpoint 0 is the main control endpoint of the core. As such, the routines required to service Endpoint 0 are more complicated than those required to service other endpoints.

The software is required to handle all the Standard Device Requests that may be received via Endpoint 0. These are described in Universal Serial Bus Specification, Revision 2.0, Chapter 9. The protocol for these device requests involves different numbers and types of transaction per transfer. To accommodate this, the CPU needs to take a state machine approach to command decoding and handling.

The Standard Device Requests can be divided into three categories: Zero Data Requests (in which all the information is included in the command), Write Requests (in which the command will be followed by additional data), and Read Requests (in which the device is required to send data back to the host).

This section looks at the sequence of events that the software must perform to process the different types of device request.

NOTE: The Setup packet associated with any Standard Device Request should include an 8-byte command. Any Setup packet containing a command field of anything other than 8 bytes will be automatically rejected by the core.

21.8.1 ZERO DATA REQUESTS

Zero data requests have all their information included in the 8-byte command and require no additional data to be transferred. Examples of zero data Standard Device Requests are: SET_FEATURE, CLEAR_FEATURE, SET_ADDRESS, SET_CONFIGURATION, SET_INTERFACE.

The sequence of events will begin, as with all requests, when the software receives an Endpoint 0 interrupt. The OutPktRdy bit (CSR0.D0) will also have been set. The 8-byte command should then be read from the Endpoint 0 FIFO, decoded and the appropriate action taken. For example if the command is SET_ADDRESS, the 7-bit address value contained in the command should be written to the FAddr register.

The CSR0 register should then be written to set the ServicedOutPktRdy bit (D6) (indicating that the command has been read from the FIFO) and to set the DataEnd bit (D3) (indicating that no further data is expected for this request).

When the host moves to the status stage of the request, a second Endpoint 0 interrupt will be generated to indicate that the request has completed. No further action is required from the software: the second interrupt is just a confirmation that the request completed successfully.

If the command is an unrecognized command, or for some other reason cannot be executed, then when it has been decoded, the CSR0 register should be written to set the ServicedOutPktRdy bit (D6)

and to set the SendStall bit (D5). When the host moves to the status stage of the request, the CORE will send a STALL to tell the host that the request was not executed. A second Endpoint 0 interrupt will be generated and the SentStall bit (CSR0.D2) will be set.

If the host sends more data after the DataEnd bit has been set, then the CORE will send a STALL. An Endpoint 0 interrupt will be generated and the SentStall bit (CSR0.D2) will be set.

21.8.2 WRITE REQUESTS

Write requests involve an additional packet (or packets) of data being sent from the host after the 8-byte command. An example f a write Standard Device Request is: SET_DESCRIPTOR.

The sequence of events will begin, as with all requests, when the software receives an Endpoint 0 interrupt. The OutPktRdy bit CSR0.D0) will also have been set. The 8-byte command should then be read from the Endpoint 0 FIFO and decoded.

As with a zero data request, the CSR0 register should then be written to set the ServicedOutPktRdy bit (D6) (indicating that the command has been read from the FIFO) but in this case the DataEnd bit (D3) should not be set (indicating that more data is expected).

When a second Endpoint 0 interrupt is received, the CSR0 register should be read to check the endpoint status. The OutPktRdy bit (CSR0:D0) should be set to indicate that a data packet has been received. The COUNT0 register should then be read to determine the size of this data packet. The data packet can then be read from the Endpoint 0 FIFO.

If the length of the data associated with the request (indicated by the *wLength* field in the command) is greater than the maximum packet size for Endpoint 0, further data packets will be sent. In this case, CSR0 should be written to set the ServicedOutPktRdy bit, but the DataEnd bit should not be set.

When all the expected data packets have been received, the CSR0 register should be written to set the ServicedOutPktRdy bit and to set the DataEnd bit (indicating that no more data is expected).

When the host moves to the status stage of the request, another Endpoint 0 interrupt will be generated to indicate that the request has completed. No further action is required from the software, the interrupt is just a confirmation that the request completed successfully.

If the command is an unrecognized command, or for some other reason cannot be executed, then when it has been decoded, the CSR0 register should be written to set the ServicedOutPktRdy bit (D6) and to set the SendStall bit (D5). When the host sends more data, the CORE will send a STALL to tell the host that the request was not executed. An Endpoint 0 interrupt will be generated and the SentStall bit (CSR0.D2) will be set.

If the host sends more data after the DataEnd has been set, then the CORE will send a STALL. An

Endpoint 0 interrupt will be generated and the SentStall bit (CSR0.D2) will be set.

21.8.3 READ REQUESTS

Read requests have a packet (or packets) of data sent from the function to the host after the 8-byte command. Examples of read Standard Device Requests are: GET_CONFIGURATION, GET_INTERFACE, GET_DESCRIPTOR, GET_STATUS, SYNCH_FRAME.

The sequence of events will begin, as with all requests, when the software receives an Endpoint 0 interrupt. The OutPktRdy bit (CSR0.D0) will also have been set. The 8-byte command should then be read from the Endpoint 0 FIFO and decoded. The CSR0 register should then be written to set the ServicedOutPktRdy bit (D6) (indicating that the command has read from the FIFO).

The data to be sent to the host should then be written to the Endpoint 0 FIFO. (If required, this data may be transferred using the DMA controller in the AHB bridge in its DMA Mode 0.) If the data to be sent is greater than the maximum packet size for Endpoint 0, only the maximum packet size should be written to the FIFO. The CSR0 register should then be written to set the InPktRdy bit (D1) (indicating that there is a packet in the FIFO to be sent). When the packet has been sent to the host, another Endpoint 0 interrupt will be generated and the next data packet can be written to the FIFO.

When the last data packet has been written to the FIFO, the CSR0 register should be written to set the InPktRdy bit and to set the DataEnd bit (D3) (indicating that there is no more data after this packet). When the host moves to the status stage of the request, another Endpoint 0 interrupt will be generated to indicate that the request has completed. No further action is required from the software: the interrupt is just a confirmation that the request completed successfully.

If the command is an unrecognized command, or for some other reason cannot be executed, then when it has been decoded, the CSR0 register should be written to set the ServicedOutPktRdy bit (D6) and to set the SendStall bit (D5). When the host requests data, the CORE will send a STALL to tell the host that the request was not executed. An Endpoint 0 interrupt will be generated and the SentStall bit (CSR0.D2) will be set.

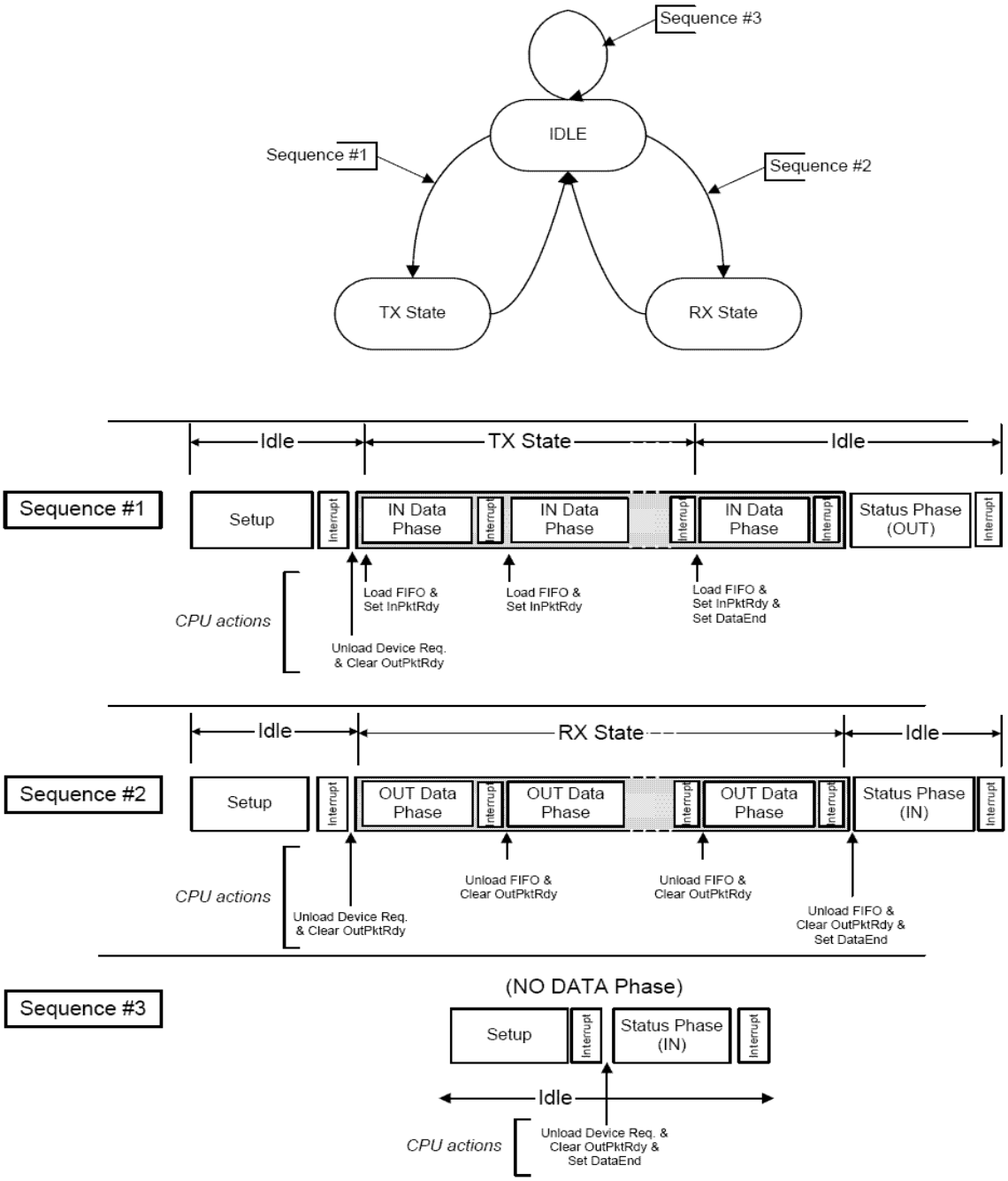
If the host requests more data after the DataEnd has been set, then the CORE will send a STALL. An Endpoint 0 interrupt will be generated and the SentStall bit (CSR0.D2) will be set.

21.8.4 END POINT0 STATES

The Endpoint 0 control needs three modes – IDLE, TX and RX – corresponding to the different phases of the control transfer and the states Endpoint 0 enters for the different phases of the transfer (see Figure 7-1 below).

The default mode on power-up or reset should be IDLE.

OutPktRdy (CSR0.D0) becoming set when Endpoint 0 is in IDLE state indicates a new device request. Once the device request is unloaded from the FIFO, the CORE decodes the descriptor to find whether there is a Data phase and, if so, the direction of the Data phase for the control transfer (in order to set the FIFO direction).



Depending on the direction of the Data phase, Endpoint 0 goes into either TX state or RX state. If there is no Data phase, Endpoint 0 remains in IDLE state to accept the next device request.

The actions that the CPU needs to take at the different phases of the possible transfers (e.g. Loading the FIFO, Setting InPktRdy) are indicated in the diagram on the following page.

Note that the CORE changes the FIFO direction depending on the direction of the Data phase independently of the CPU.

21.8.5 END POINT0 SERVICER OUTINE

An Endpoint 0 interrupt is generated:

- When the core sets the OutPktRdy bit (CSR0.D0) after a valid token has been received and data has been written to the FIFO.
- When the core clears the InPktRdy bit (CSR0.D1) after the packet of data in the FIFO has been successfully transmitted to the host.
- When the core sets the SentStall bit (CSR0.D2) after a control transaction is ended due to a protocol violation.
- When the core sets the SetupEnd bit (CSR0.D4) because a control transfer has ended before DataEnd (CSR0.D3) is set.

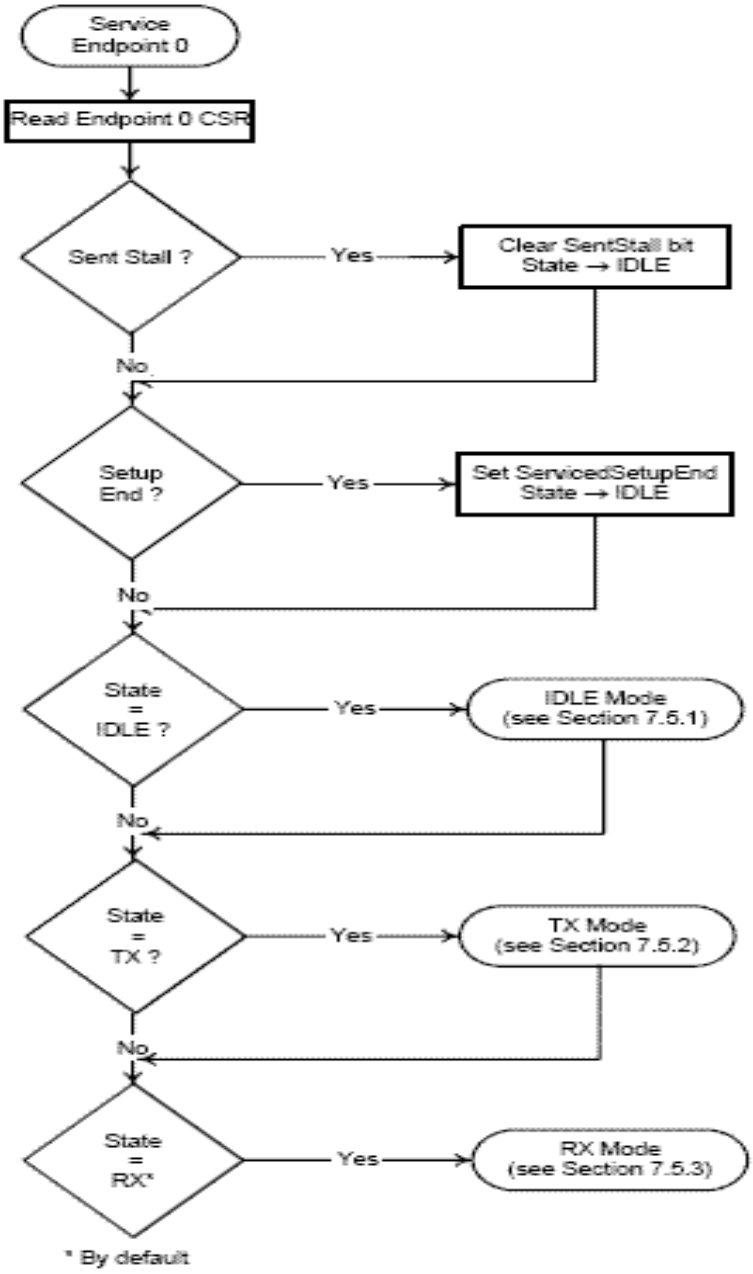
Whenever the Endpoint 0 service routine is entered, the firmware must first check to see if the current control transfer has been ended due to either a STALL condition or a premature end of control transfer. If the control transfer ends due to a STALL condition, the SentStall bit would be set. If the control transfer ends due to a premature end of control transfer, the SetupEnd bit would be set. In either case, the firmware should abort processing the current control transfer and set the state to IDLE.

Once the firmware has determined that the interrupt was not generated by an illegal bus state, the next action taken depends on the Endpoint state.

If Endpoint 0 is in IDLE state, the only valid reason an interrupt can be generated is as a result of the core receiving data from the USB bus. The service routine must check for this by testing the OutPktRdy bit. If this bit is set, then the core has received a SETUP packet. This must be unloaded from the FIFO and decoded to determine the action the core must take. Depending on the command contained within the SETUP packet, Endpoint 0 will enter one of three states:

- If the command is a single packet transaction (SET_ADDRESS, SET_INTERFACE etc) without any data phase, the endpoint will remain in IDLE state.
- If the command has an OUT data phase (SET_DESCRIPTOR etc) the endpoint will enter RX state.
- If the command has an IN data phase (GET_DESCRIPTOR etc) the endpoint will enter TX state.

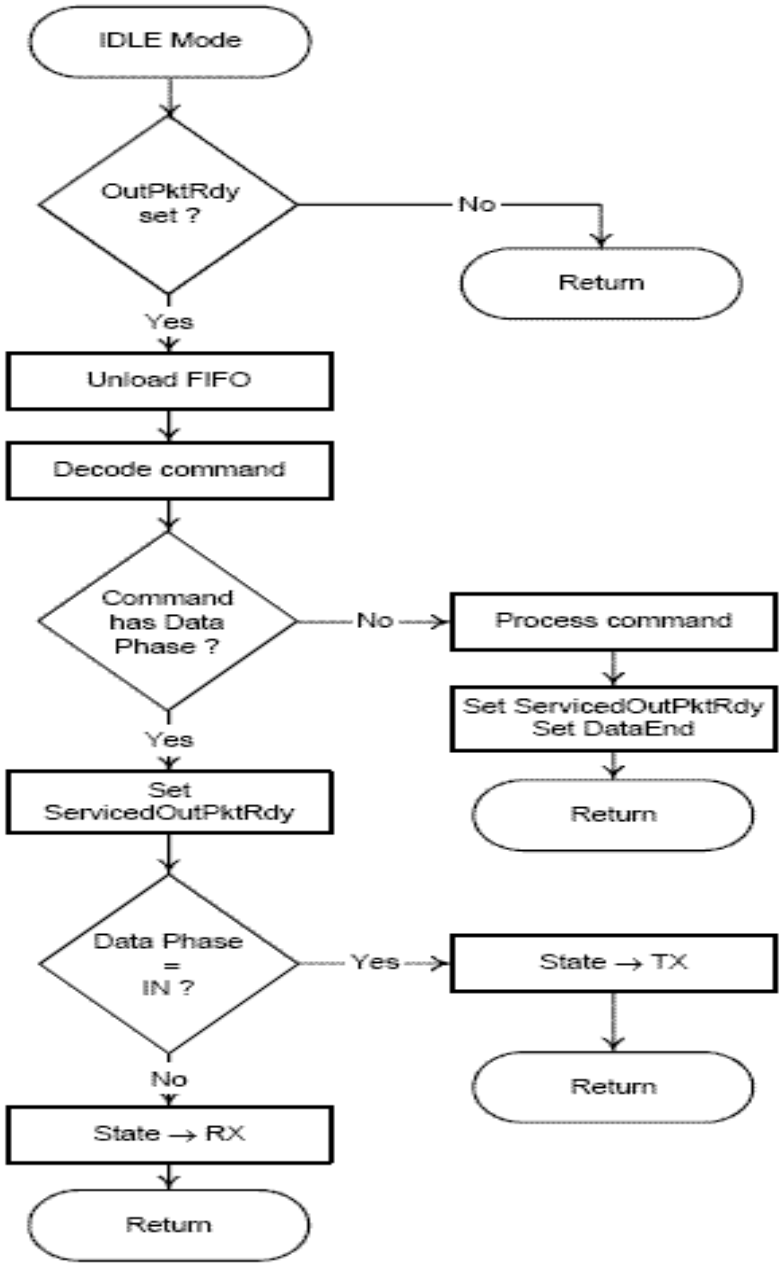
If the endpoint is in TX state, the interrupt indicates that the core has received an IN token and data from the FIFO has been sent. The firmware must respond to this either by placing more data in the FIFO if the host is still expecting more data2 or by setting he DataEnd bit to indicate that the data phase is complete. Once the data phase of the transaction has been completed, endpoint 0 should be returned to IDLE state to await the next control transaction.



If the endpoint is in RX state, the interrupt indicates that a data packet has been received. The firmware must respond by unloading he received data from the FIFO. The firmware must then determine whether it has received all of the expected data2. If it has, the firmware should set the DataEnd bit and return Endpoint 0 to IDLE state. If more data is expected, the firmware should set the ServicedOutPktRdy bit (CSR0.D6) to indicate that it has read the data in the FIFO and leave the endpoint in RX state.

21.8.6 IDLE MODE

IDLE mode is the mode the Endpoint 0 control needs to select at power-on or reset and is the mode to which the Endpoint 0 control should return when the RX and TX modes are terminated.



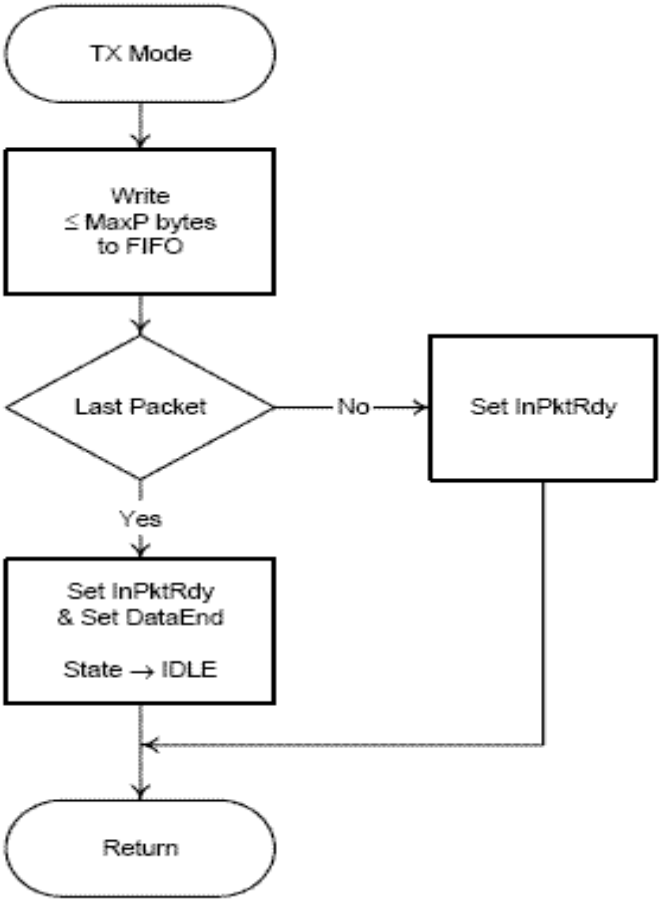
21.8.7 TX MODE

When the endpoint is in TX state, all arriving IN tokens need to be treated as part of a Data phase until the required amount of data has been sent to the host. If either a SETUP or an OUT token is received whilst the endpoint is in the TX state, this will cause a SetupEnd condition to occur as the core expects only IN tokens.

Three events can cause TX mode to be terminated before the expected amount of data has been sent:

- The host sends an invalid token causing a SetupEnd condition (CSR0.D4 set).
- The firmware sends a packet containing less than the maximum packet size for Endpoint 0 (MaxP).
- The firmware sends an empty data packet.

Until the transaction is terminated, the firmware simply needs to load the FIFO when it receives an interrupt which indicates that packet has been sent from the FIFO. (An interrupt is generated when InPktRdy is cleared.)



When the firmware forces the termination of a transfer (by sending a short or empty data packet), it should set the DataEnd bit CSR0.D3) to indicate to the core that the Data phase is complete and that the core should next receive an acknowledge packet.

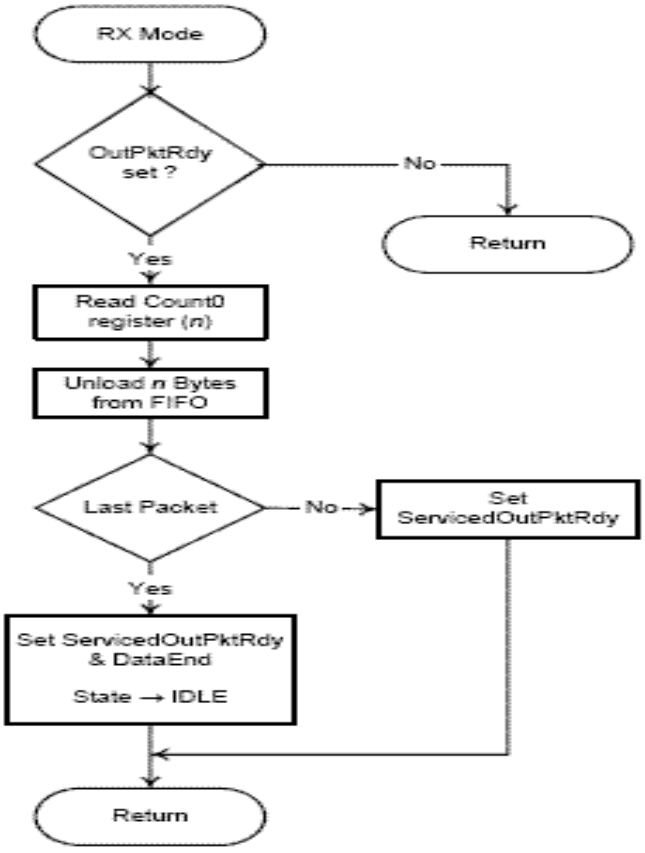
21.8.8 RX MODE

In RX mode, all arriving data should be treated as part of a Data phase until the expected amount of data has been received. If ither a SETUP or an IN token is received while the endpoint is in RX state, this will cause a SetupEnd condition to occur as the ore expects only OUT tokens.

Three events can cause RX mode to be terminated before the expected amount of data has been received:

- The host sends an invalid token causing a SetupEnd condition (CSR0.D4 set).
- The host sends a packet which contains less than the maximum packet size for Endpoint 0.
- The host sends an empty data packet.

Until the transaction is terminated, the firmware simply needs to unload the FIFO when it receives an interrupt which indicates hat new data has arrived (OutPktRdy (CSR0.D0) set) and to clear OutPktRdy by setting the ServicedOutPktRdy bit (CSR0.D6).



When the firmware detects the termination of a transfer (by receiving either the expected amount of data or an empty data packet), it should set the DataEnd bit (CSR0.D3) to indicate to the core that the Data phase is complete and that the core should receive an acknowledge packet next.

21.8.9 ERROR HANDLING

A control transfer may be aborted due to a protocol error on the USB, the host prematurely ending the transfer, or if the function controller software wishes to abort the transfer (e.g. because it cannot process the command).

The CORE will automatically detect protocol errors and send a STALL packet to the host under the following conditions:

1 *The host sends more data during the OUT Data phase of a write request than was specified in the command.* This condition is detected when the host sends an OUT token after the

DataEnd bit (CSR0.D3) has been set.

2 *The host request more data during the IN Data phase of a read request than was specified in the command.* This condition is detected when the host sends an IN token after the DataEnd bit in the CSR0 register has been set.

3 The host sends more than MaxP data bytes in an OUT data packet.

4 The host sends a non-zero length DATA1 packet during the STATUS phase of a read request.

When the CORE has sent the STALL packet, it sets the SentStall bit (CSR0.D2) and generates an interrupt. When the software receives an Endpoint 0 interrupt with the SentStall bit set, it should abort the current transfer, clear the SentStall bit, and return to the IDLE state.

If the host prematurely ends a transfer by entering the STATUS phase before all the data for the request has been transferred, or by sending a new SETUP packet before completing the current transfer, then the SetupEnd bit (CSR0.D4) will be set and an Endpoint 0 interrupt generated. When the software receives an Endpoint 0 interrupt with the SetupEnd bit set, it should abort the current transfer, set the ServicedSetupEnd bit (CSR0.D7), and return to the IDLE state. If the OutPktRdy bit (CSR0.D0) is set this indicates that the host has sent another SETUP packet and the software should then process this command.

If the software wants to abort the current transfer, because it cannot process the command or has some other internal error, then it should set the SendStall bit (CSR0.D5). The CORE will then send a STALL packet to the host, set the SentStall bit (CSR0.D2) and generate an Endpoint 0 interrupt.

21.9 BULK TRANSACTIONS

21.9.1 BULK IN ENDPOINT

A Bulk IN endpoint is used to transfer non-periodic data from the function controller to the host. Four optional features are available for use with a Bulk IN endpoint:

- **Double packet buffering**
  Except where dynamic FIFO sizing is being used, when the value written to the InMaxP register is less than, or equal to, half the size of the FIFO allocated to the endpoint, double packet buffering will be automatically enabled. When enabled, up to two packets can be stored in the FIFO awaiting transmission to the host.

- **DMA**
  If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint is able to accept another packet in its FIFO. This feature can be used to allow an external DMA controller (such as the one included in the supplied AHB bridge) to load packets into the FIFO without processor intervention. See Section 8.3.

- **AutoSet**
  When the AutoSet feature is enabled, the InPktRdy bit (InCSR.D0) will be automatically set when a packet of InMaxP bytes has been loaded into the FIFO. This is particularly useful when DMA is used to load the FIFO as it avoids the need for any processor intervention when loading individual packets during a large Bulk transfer.

- **Automatic Packet Splitting**
  For some system designs, it may be convenient for the application software to write larger amounts of data to an endpoint in a single operation than can be transferred in a single USB operation. A particular case in point is where the same endpoint is used for high-speed transfers of 512 bytes under certain circumstances but for full-speed transfers under other circumstances. When operating at full-speed, the maximum amount of data transferred in a single operation is then just 64 bytes. To cater for such circumstances, the CORE includes a configuration option which, if selected, allows larger data packets to be written to Bulk endpoints which are then split into packets of an appropriate (specified) size for transfer across the USB bus. The necessary packet size information is set via the InMaxP register.

21.9.1.1 SETUP

Before using a Bulk IN endpoint the InMaxP register must be written with the maximum packet size (in bytes) for the endpoint. This value should be the same as the *wMaxPacketSize* field of the Standard Endpoint Descriptor for the endpoint. In addition, the relevant interrupt enable bit in the IntrInE register should be set to 1 (if an interrupt is required for this endpoint), and the high byte of the InCSR register should be set as shown below (Bits D9 – D8 are unused).

When a Bulk IN endpoint is first configured, following a SET_CONFIGURATION or SET_INTERFACE command on Endpoint 0, then the lower byte of InCSR should be written to set the ClrDataTog bit (D6). This will ensure that the data toggle (which is handled automatically by the CORE) starts in the correct state. Also if there are any data packets in the FIFO (indicated by the FIFONotEmpty bit (InCSR.D1) being set), they should be flushed by setting the FlushFIFO bit (InCSR.D3).

NOTE: It may be necessary to set this bit twice in succession if double buffering is enabled.

D15	AutoSet	0/1	Set to 1 if the AutoSet feature is required.
D14	ISO	0	Set to 0 to enable Bulk protocol.
D13	Mode	1	Set to 1 to ensure the FIFO is enabled (only necessary if the FIFO is shared with an OUT endpoint).
D12	DMAReqEnab	0/1	Set to 1 if a DMA request is required for this endpoint. *Note:* If set to 1, will also need to select the chosen DMAReqMode (InCSR.D10).
D11	FrcDataTog	0	Set to 0 to allow normal data toggle operation.

21.9.1.2 OPERATION

When data is to be transferred over a Bulk IN pipe, a data packet is loaded into the FIFO and the InCSR register written to set the InPktRdy bit (D0). When the packet has been sent, the InPktRdy bit will be cleared by the CORE and an interrupt generated so that the next packet can be loaded into the FIFO. If double packet buffering is enabled (i.e. if the size of the FIFO is at least twice the maximum packet size set in the InMaxP register), then after the first packet has been loaded and the InPktRdy bit set, InPktRdy will be immediately cleared by the CORE and an interrupt generated so that a second packet can be loaded into the FIFO. This means the software can operate the same way, loading a packet when it receives an interrupt, regardless of whether double packet buffering is enabled or not.

In general, the packet size must not exceed the payload specified in the InMaxP register. This defines the maximum packet size (MaxP) for a single transfer over the USB and, for bulk transfers, is required by the USB Specification to be either 8, 16, 32, 64 (Full-Speed or High-Speed) or 512 bytes (High-Speed only). If more than this amount of data is to be transferred, this needs to be sent as multiple USB packets which should all carry the full payload, except for the last packet which holds the residue.

The exception to this rule applies where the automatic Bulk packet splitting option has been selected when the core was configured. Where this option has been selected, packets up to 32 times MaxP can be written to the FIFO (assuming that the FIFO is big enough to accept these larger packets) which are then split by the core into packets of the appropriate size for transfer over the USB. The size of the packets written to the FIFO is given by $m \times payload$ where InMaxP[D15:D11] = m – 1. All the application software needs to do to take advantage of this feature is to set the appropriate values in the InMaxP register (and ensure that the value written to bits 10:0 matches the value given in the *wMaxPacketSize* field of the Standard Endpoint Descriptor for the associated endpoint). As far as the

application software is concerned, the process of transferring these larger packets is no different from that used to transfer a standard-sized Bulk packet.

The host may determine that all the data for a transfer has been sent by knowing the total size of the data block. Alternatively it may infer that all the data have been sent when it receives a packet which is less than the payload in size. In the latter case, if the total size of the data block is an exact multiple of the payload, it will be necessary for the function to send a null packet after all the data has been sent. This is done by setting InPktRdy when the next interrupt is received, without loading any data into the FIFO.

If large blocks of data are being transferred, then the overhead of calling an interrupt service routine to load each packet can be avoided by using DMA.

21.9.1.3 ERROR HANDLING

If the software wants to shut down the Bulk IN pipe, it should set the SendStall bit (InCSR.D4). When the CORE receives the next IN token, it will send a STALL to the host, set the SentStall bit (InCSR.D5) and generate an interrupt.

When the software receives an interrupt with the SentStall bit (InCSR.D5) set, it should clear the SentStall bit. It should leave the SendStall bit (InCSR.D4) set until it is ready to re-enable the Bulk IN pipe. **NOTE:** If the host failed to receive the STALL packet for some reason, it will send another IN token, so it is advisable to leave the SendStall bit set until the software is ready to re-enable the Bulk IN pipe.

When a pipe is re-enabled, the data toggle sequence should be restarted by setting the ClrDataTog bit in the InCSR register (D6).

21.9.2 BULK OUT ENDPOINT

A Bulk OUT endpoint is used to transfer non-periodic data from the host to the function controller. Four optional features are available for use with a Bulk OUT endpoint:

- **Double packet buffering**
  Except where dynamic FIFO sizing is being used, when the value written to the OutMaxP register is less than, or equal to, half the size of the FIFO allocated to the endpoint, double packet buffering will be automatically enabled. When enabled, up to two packets can be stored in the FIFO.

- **DMA**
  If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint has a packet in its FIFO. This feature can be used to allow an external DMA controller (such as the one included in the supplied AHB bridge) to unload packets from the FIFO without processor

intervention.

- **AutoClear**
  When the AutoClear feature is enabled, the OutPktRdy bit (OutCSR.D0) will be automatically cleared when a packet of OutMaxP bytes has been unloaded from the FIFO. This is particularly useful when DMA is used to unload the FIFO as it avoids the need for any processor intervention when unloading individual packets during a large Bulk transfer.

- **Automatic Packet Combining**
  For some system designs, it may be convenient for the application software to read larger amounts of data from an endpoint in a single operation than can be transferred in a single USB operation. A particular case in point is where the same endpoint is used for high-speed transfers of 512 bytes under certain circumstances but for full-speed transfers under other circumstances. When operating at full-speed, the maximum amount of data transferred in a single operation is then just 64 bytes. To cater for such circumstances, the CORE includes a configuration option which, if selected, causes the CORE to combine the packets received across the USB bus into larger data packets prior to being read by the application software. The necessary packet.

21.9.2.1 SET UP

Before using a Bulk OUT endpoint, the OutMaxP register must be written with the maximum packet size (in bytes) for the endpoint. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the endpoint. In addition, the relevant interrupt enable bit in the IntrOutE register should be set to 1 (if an interrupt is required for this endpoint) and the high byte of the OutCSR register should be set as shown below: (Bits D10 – D8 are unused/Read-only.)

D15	AutoClear	0/1	Set to 1 if the AutoClear feature is required.
D14	ISO	0	Set to 0 to enable Bulk protocol.
D13	DMAReqEnab	0/1	Set to 1 if a DMA request is required for this endpoint. *Note:* If set to 1, will also need to select the chosen DMAReqMode (OutCSR.D11).
D12	DisNyet	0	Set to 0 to allow normal PING flow control.

When a Bulk OUT endpoint is first configured, following a SET_CONFIGURATION or SET_INTERFACE command on Endpoint 0, the lower byte of OutCSR should be written to set the ClrDataTog bit (D7). This will ensure that the data toggle (which is handled automatically by the CORE) starts in the correct state. Also if there are any data packets in the FIFO (indicated by the OutPktRdy bit (OutCSR.D0) being set), they should be flushed by setting the FlushFIFO bit (OutCSR.D4).

NOTE: It may be necessary to set this bit twice in succession if double buffering is enabled.

21.9.2.2 OPERATION

When a data packet is received by a Bulk OUT endpoint, the OutPktRdy bit (OutCSR.D0) is set and an interrupt is generated. The software should read the OutCount register for the endpoint to determine the size of the data packet. The data packet should be read from the FIFO, then the OutPktRdy bit should be cleared.

The packets received should not exceed the size specified in the OutMaxP register (because this should match the value set in the *wMaxPacketSize* field of the endpoint descriptor sent to the host). When a block of data larger than *wMaxPacketSize* needs to be sent to the function, it will be sent as multiple packets. All the packets will be *wMaxPacketSize* in size, except the last packet which will contain the residue. The software may use an application specific method of determining the total size of the block and hence when the last packet has been received. Alternatively it may infer that the entire block has been received when it receives a packet which is less than *wMaxPacketSize* in size. (If the total size of the data block is a multiple of *wMaxPacketSize*, a null data packet will be sent after the data to signify that the transfer is complete.)

In general, the application software will need to read each packet from the FIFO individually. The exception to this rule applies where the option for automatic combining of Bulk packets has been selected when the core was configured. Where this option has been selected, the core can receive up to 32 packets at a time and combine them into a single packet within the FIFO (assuming that the FIFO is big enough to accept these larger packets). The size of the packets written to the FIFO is given by $m \times$ *wMaxPacketSize* where OutMaxP[D15:D11] = $m - 1$. All the application software needs to do to take advantage of this feature is set the appropriate values in the OutMaxP register (and ensure that the value written to bits 10:0 matches the value given in the *wMaxPacketSize* field of the endpoint descriptor). As far as the application software is concerned, the process of transferring these larger packets is no different from that used to transfer a standard-sized Bulk packet.

If large blocks of data are being transferred, the overhead of calling an interrupt service routine to unload each packet can be avoided by using DMA.

21.9.2.3 ERROR HANDLING

If the software wants to shut down the Bulk OUT pipe, it should set the SendStall bit (OutCSR.D5). When the CORE receives the next packet it will send a STALL to the host, set the SentStall bit (OutCSR.D6) and generate an interrupt.

When the software receives an interrupt with the SentStall bit (OutCSR.D6) set, it should clear the SentStall bit. It should leave the SendStall bit (OutCSR.D5) set until it is ready to re-enable the Bulk OUT pipe. ***NOTE:*** If the host failed to receive the STALL packet for some reason, it will send another packet, so it is advisable to leave the SendStall bit set until the software is ready to re-enable the Bulk OUT pipe. When a Bulk OUT pipe is re-enabled, the data toggle sequence should be restarted by setting the ClrDataTog bit in the OutCSR register (D7).

21.9.2.4 EMPLOYINGDMA

The advantage of employing DMA is that it improves bus and processor utilization when loading or unloading the FIFOs. It is particularly useful when large blocks of data are to be transferred through a Bulk endpoint. The USB protocol requires that large data blocks are transferred by sending a series of packets of the maximum packet size for the endpoint (512 bytes for high speed, 64 bytes for full speed). Only the last packet in the series may be less than the maximum packet size. Indeed, the receiver may use the reception of this 'short' packet to signal the end of the transfer (a null packet may be sent at the end of the series if the size of the data block is an exact multiple of the maximum packet size).

The DMA facilities of the CORE may be used, with a suitably programmed DMA controller, to avoid the overhead of having to interrupt the processor after each individual packet, interrupting the processor only after the transfer has completed.

Versions of the core that use the AHB Interface optionally include a DMA controller, built into the AHB interface. Where the core is configured with the VCI interface, this DMA controller needs to be added by the user. This should be connected to the CPU interface such that DMA accesses appear like normal CPU reads and writes to the CORE.

21.9.2.5 USING DMA WITH BULKINENDPOINTS

For IN endpoints, the DMA request line will go high when the endpoint FIFO is able to accept a data packet. It will either go low when InMaxP bytes have been loaded into the FIFO (or, if the 'Early DMA De-assert' option is selected, while the last 16 bytes are being loaded) – Alternatively, the request line will go low when the InPktRdy bit in InCSR is set.

To use DMA to send a large block of data to the USB host over a Bulk IN endpoint, the DMA controller and CORE should be set up as follows.

The DMA controller should be programmed to perform a burst read of the maximum packet size for the endpoint (512 bytes for high speed, 64 bytes for full speed) when the DMA request line for the endpoint transitions from low to high. Details of the settings to make in the case of the built-in DMA controller are given in Section 12 of the CORE Product Specification. The controller should keep performing these burst reads on each DMA request until the entire data block has been transferred. (The last burst may however be less than the maximum packet size.) It should then interrupt the processor.

The CORE should be programmed to enable AutoSet and DMA Request Mode 1 by setting the AutoSet, DMAReqEnab and DMAReqMode bits in the InCSR register (bits D15, D12 and D10 respectively).

Programmed like this, the CORE will take the DMA request line high whenever there is space in its FIFO to accept a packet. Further, the InPktRdy bit will be automatically set after the DMA controller has

loaded the FIFO with a packet of the maximum packet size. The packet is then ready to be sent to the host. When the last packet has been loaded by the DMA controller, the controller should interrupt the processor. (The built-in controller does this by asserting DMA_NINT.) If the last packet loaded was less than the maximum packet size, the InPktRdy bit will not have been set and will therefore need to be set manually (i.e. by the CPU) to allow the last packet to be sent. The InPktRdy bit will also need to be set manually if the last packet was of the maximum packet size and a null packet is to be sent to indicate the end of the transfer.

21.9.2.6 USING DMA WITHBULKOUT ENDPOINTS

The behavior of the DMA request line for an OUT Endpoint depends on the DMA Request Mode selected through the OutCSR register (D11). In DMA Request Mode 0, the OUT DMA request line goes high when a data packet is available in the endpoint FIFO and goes low either when the last byte of the data packet has been read (or, if the 'Early DMA De-assert' option is selected, just before the last 16 bytes are read from the FIFO) – or when the OutPktRdy bit in OutCSR is cleared. In DMA Request Mode 1, the DMA request line only goes high when the packet received is of the maximum packet size (as set in the OutMaxP register). If the packet received is of some other size, the DMA request line stays low with the result that the packet remains in the FIFO with outPktRdy set. This causes an OUT Endpoint interrupt to be generated (if enabled).

The DMA Request Modes are primarily designed to be used where large packets of data are transferred to a Bulk endpoint. The USB protocol requires such packets to be split into a series of packets of maximum packet size (512 bytes for high speed, 64 bytes for full speed). The last packet in the series may be less than the maximum packet size (or a null packet if the total size of the transfer is an exact multiple of the maximum packet size) and the receiver may interpret this 'short' packet as signaling the end of the transfer. DMA Request Mode 1 can be used, with a suitably programmed DMA controller, to avoid the overhead of having to interrupt the processor after each individual packet – instead just interrupting the processor after the transfer has completed.

NOTE: If the Request Mode is switched from Request Mode 1 to Request Mode 0, the request line will be asserted if there is a packet in the FIFO in order to allow this 'pre-received' packet to be downloaded.

21.9.3 INTERRUPT TRANSACTIONS

21.9.3.1 INTERRUPT INENDPOINT

An Interrupt IN endpoint is used to transfer periodic data from the function controller to the host.

An Interrupt IN endpoint uses the same protocol as a Bulk IN endpoint and can be used the same way. However, though DMA can be used, it offers little benefit as Interrupt endpoints are usually expected to transfer all their data in a single packet.

Interrupt IN endpoints also support one feature that Bulk IN endpoints do not, in that they support continuous toggle of the data toggle bit. This feature is enabled by setting the FrcDataTog bit in the InCSR register (D11). When this bit is set to 1, the CORE will consider the packet as having been successfully sent and toggle the data bit for the endpoint, regardless of whether an ACK was received from the host.

21.9.3.2 INTERRUPT OUT END POINT

An Interrupt OUT endpoint is used to transfer periodic data from the host to a function controller.

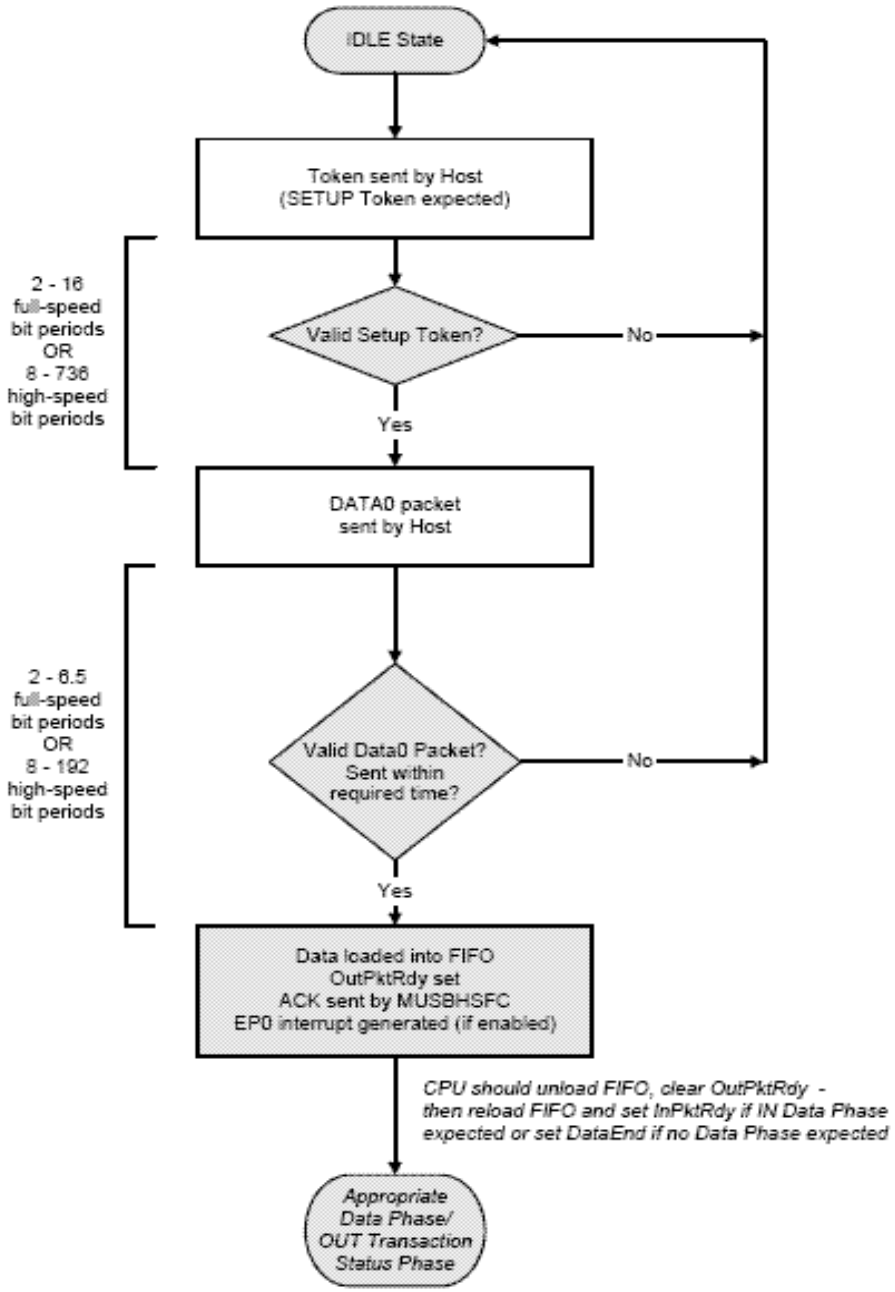
An Interrupt OUT endpoint uses almost the same protocol as a Bulk OUT endpoint and can be used the same way. The one difference is that Interrupt endpoints do not support PING flow control. This means that the CORE should never respond with a NYET handshake, only ACK/NAK/STALL. To ensure this, the DisNyet bit in the OutCSR register (D12) should be set to 1 to disable the transmission of NYET handshakes in High-speed mode.

Though DMA can be used with an Interrupt OUT endpoint, it generally offers little benefit as Interrupt endpoints are usually expected to transfer all their data in a single packet.

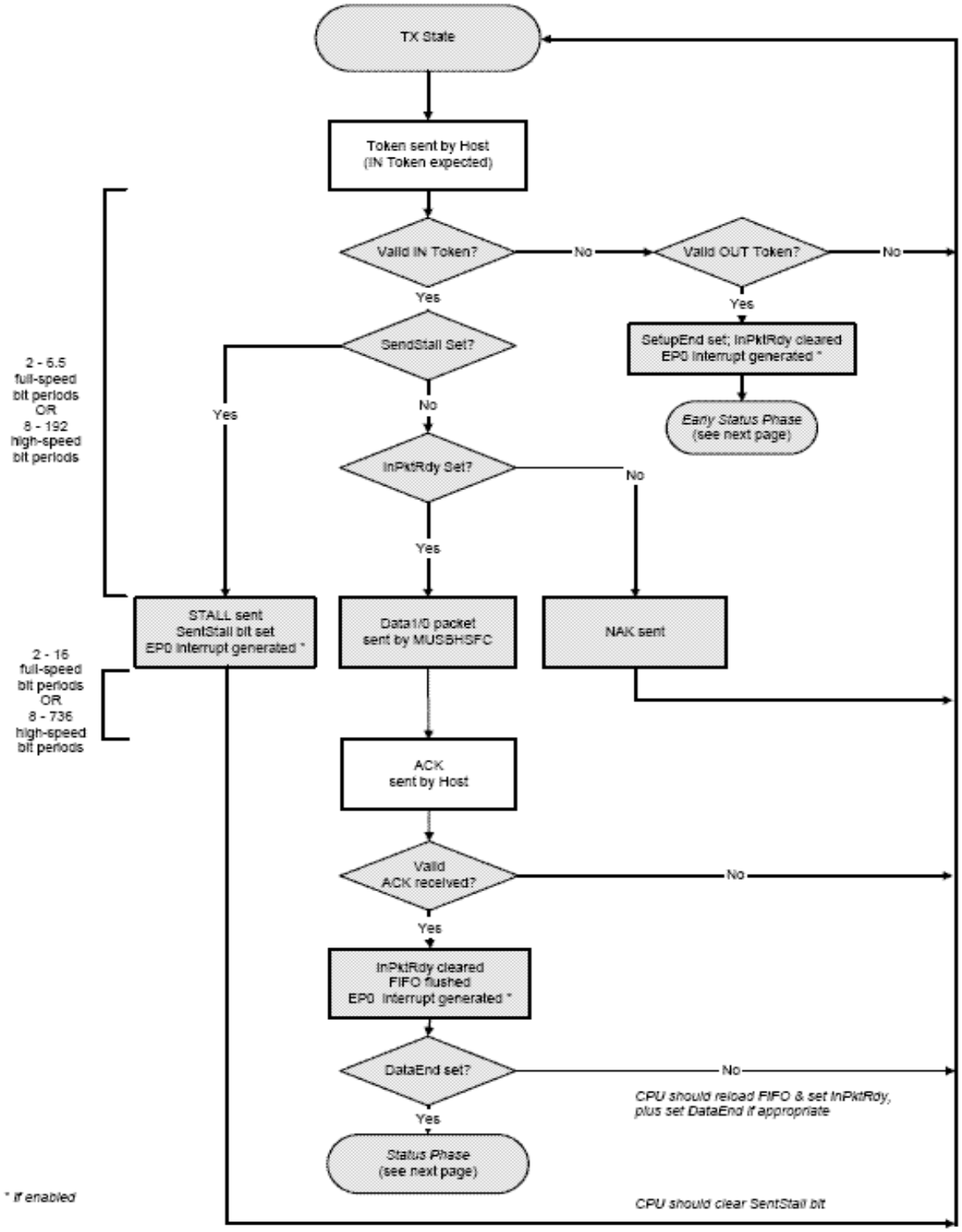
21.10 TRANSACTION FLOWS

21.10.1 CONTROL TRANSACTIONS

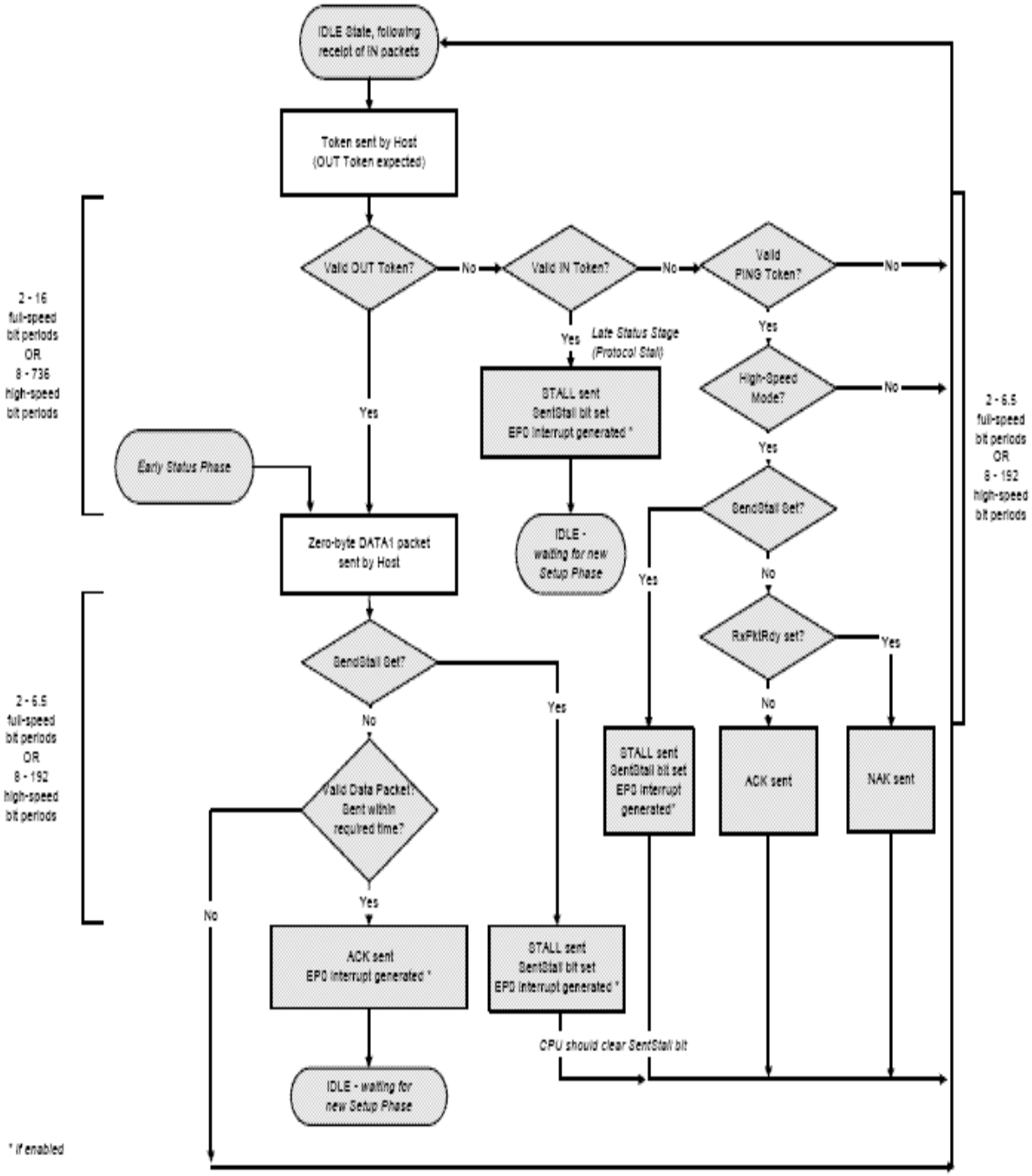
21.10.1.1 SET UP PHASE



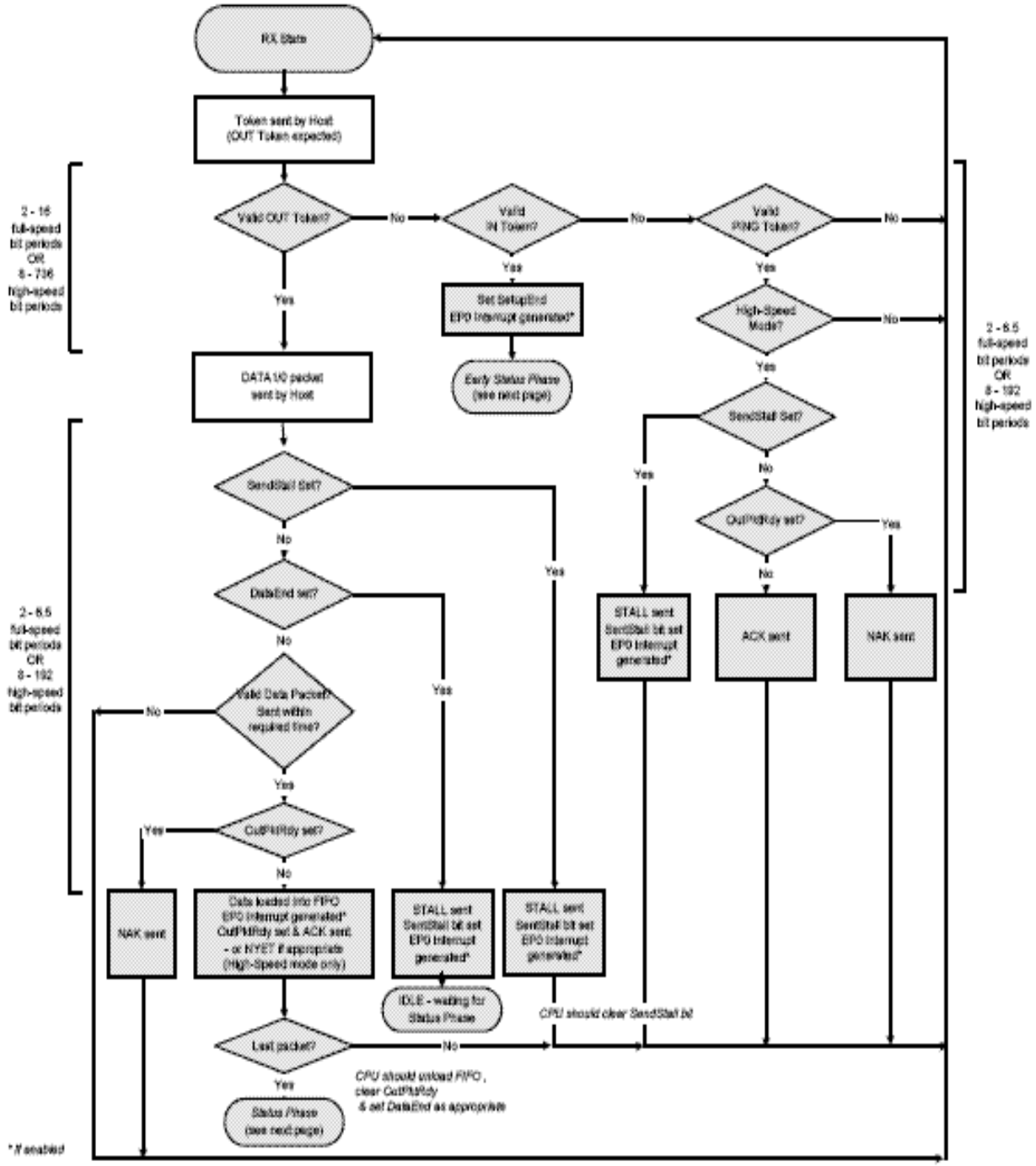
21.10.1.2 IN DATA PHASE



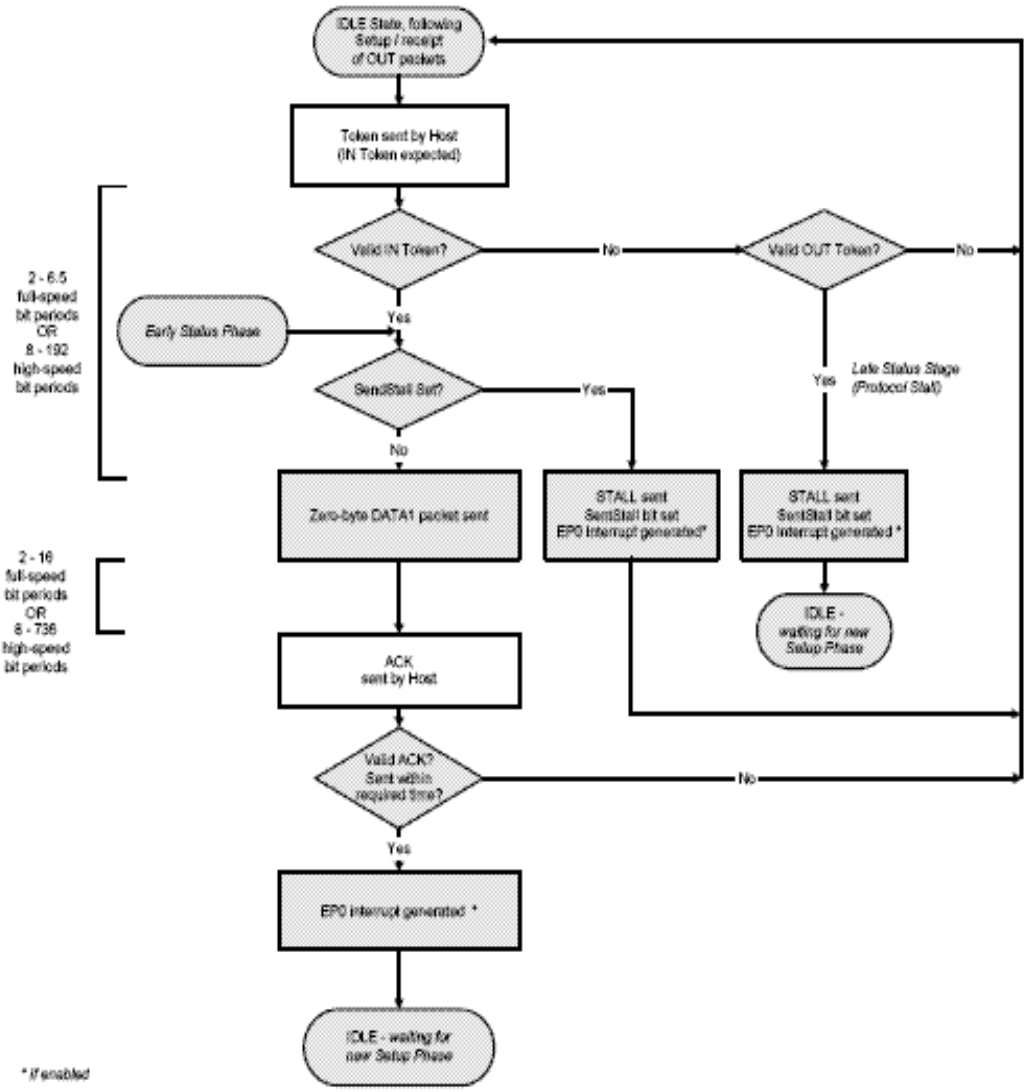
21.10.1.3 AND FOLLOWING STATUS PHASE



21.10.1.4 OUT DATA PHASE

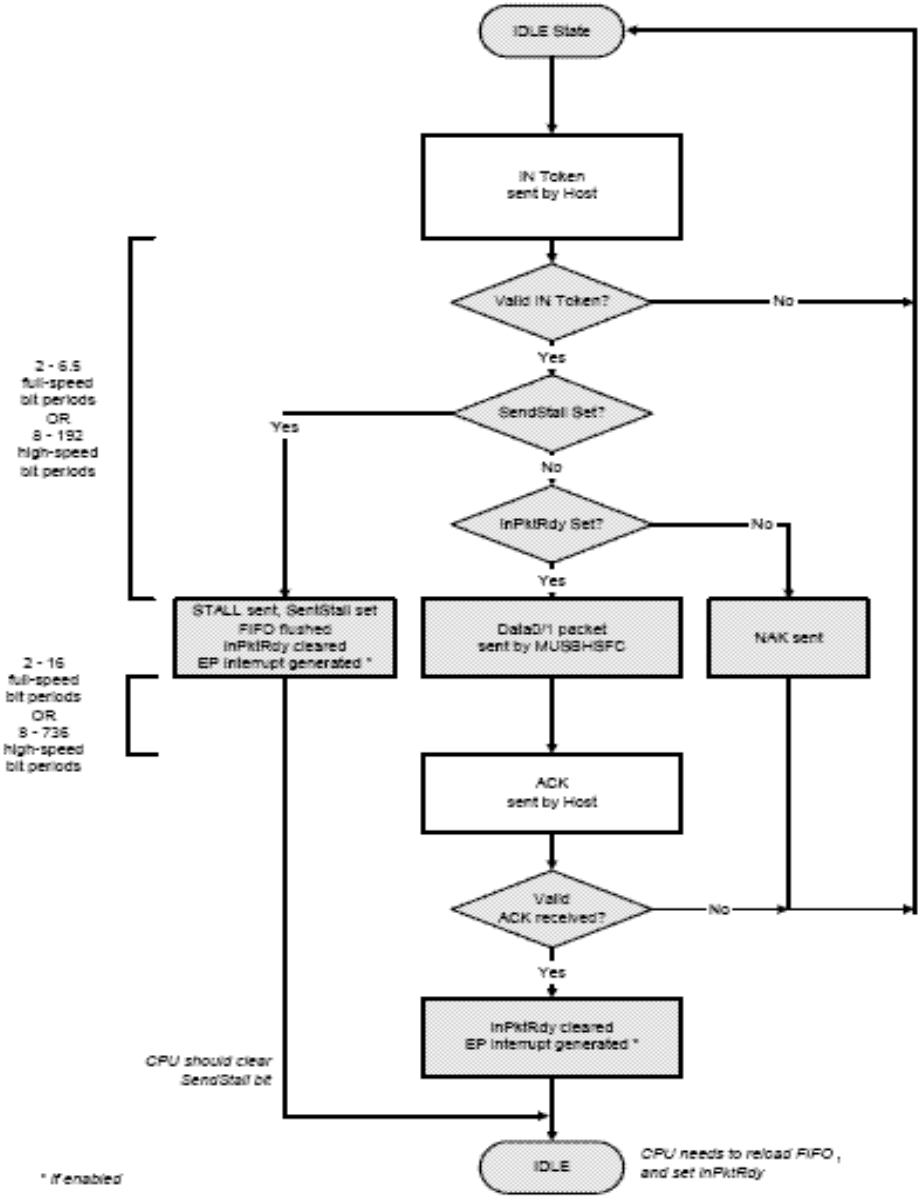


21.10.1.5 FOLLOWING STATUS PHASE

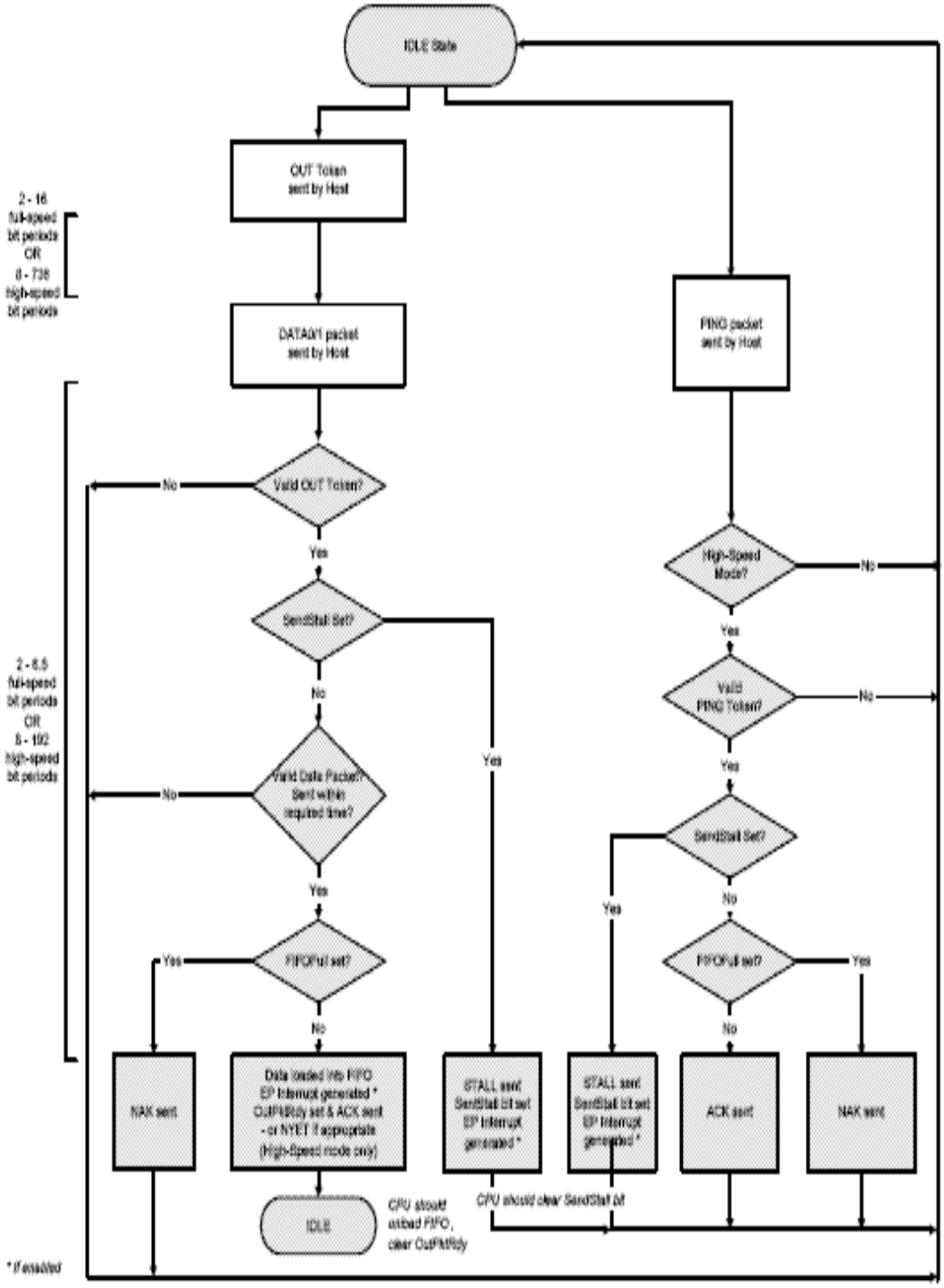


21.10.2 BULK/INTERRUPT TRANSACTIONS

21.10.2.1 INTRANSACTION

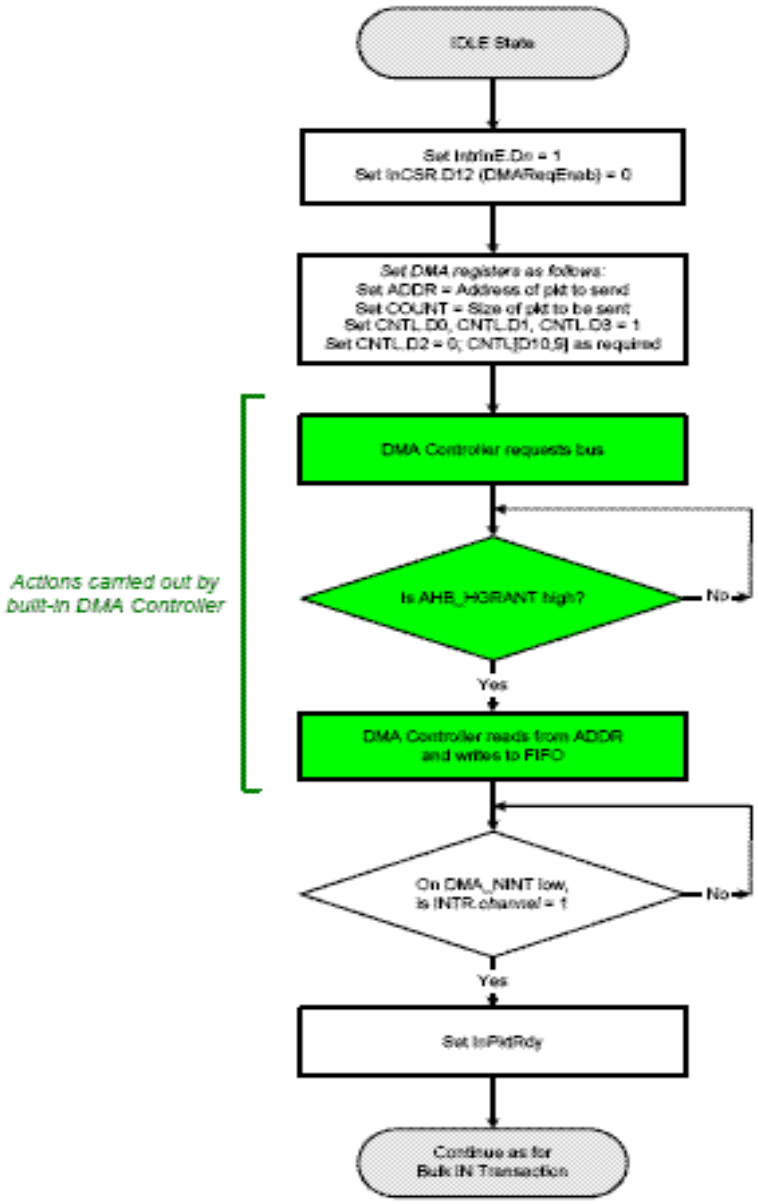


21.10.2.2 OUT TRANSACT ION

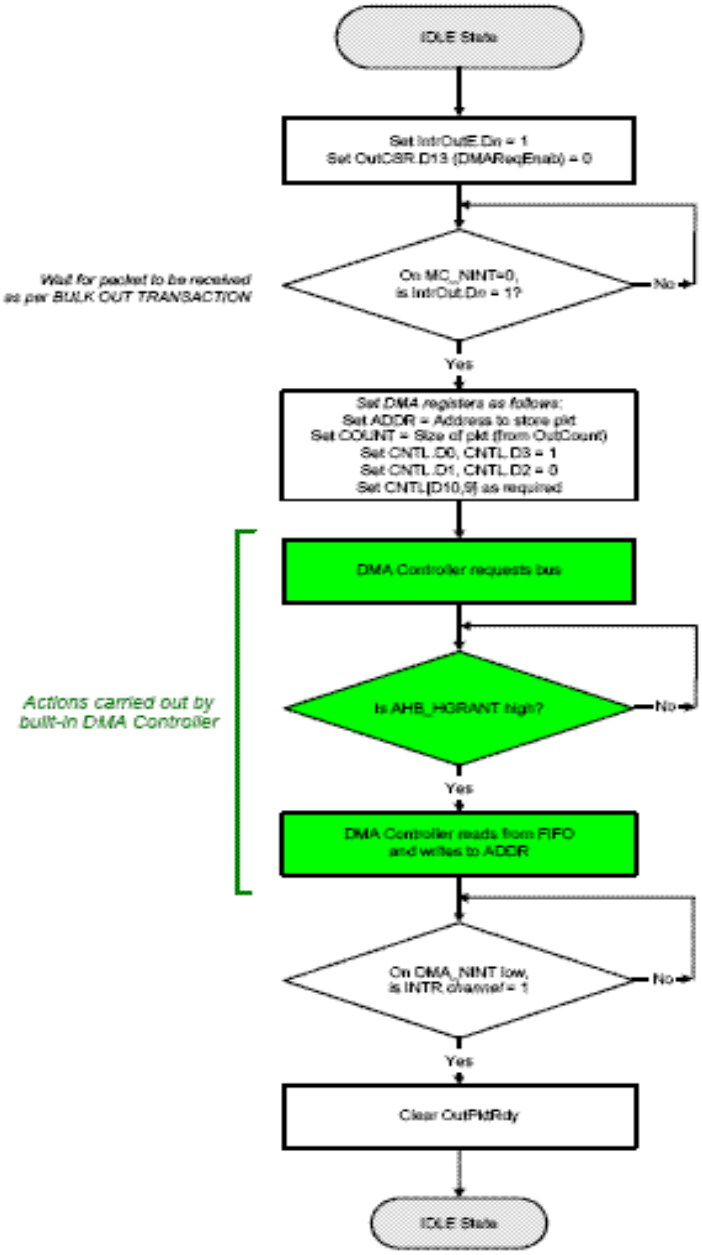


21.10.3 DMA OPERATIONS (WITH BUI LT- IN DMA CONTROLLE)

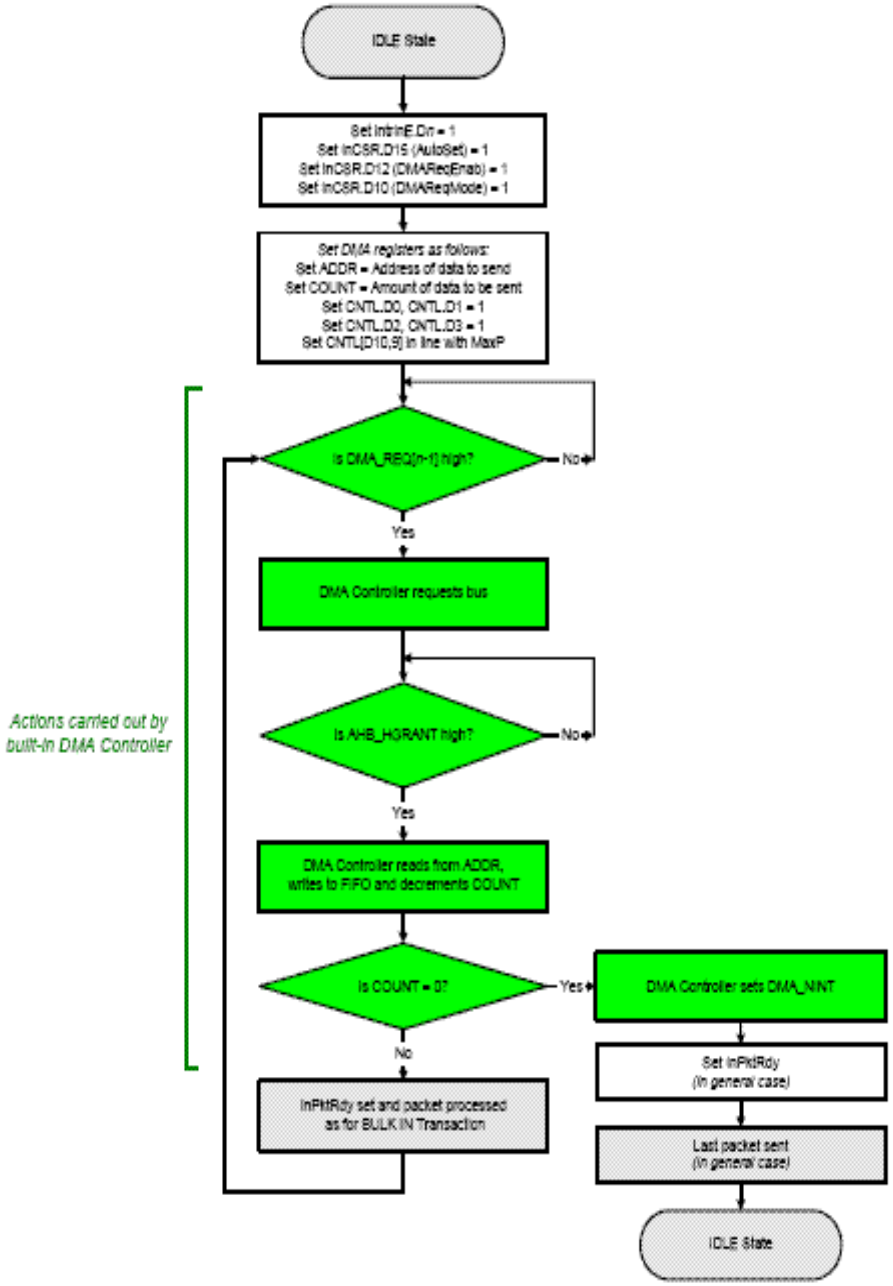
21.10.3.1 SINGLE IN PACKET



21.10.3.2 SINGLE OUT PACKET

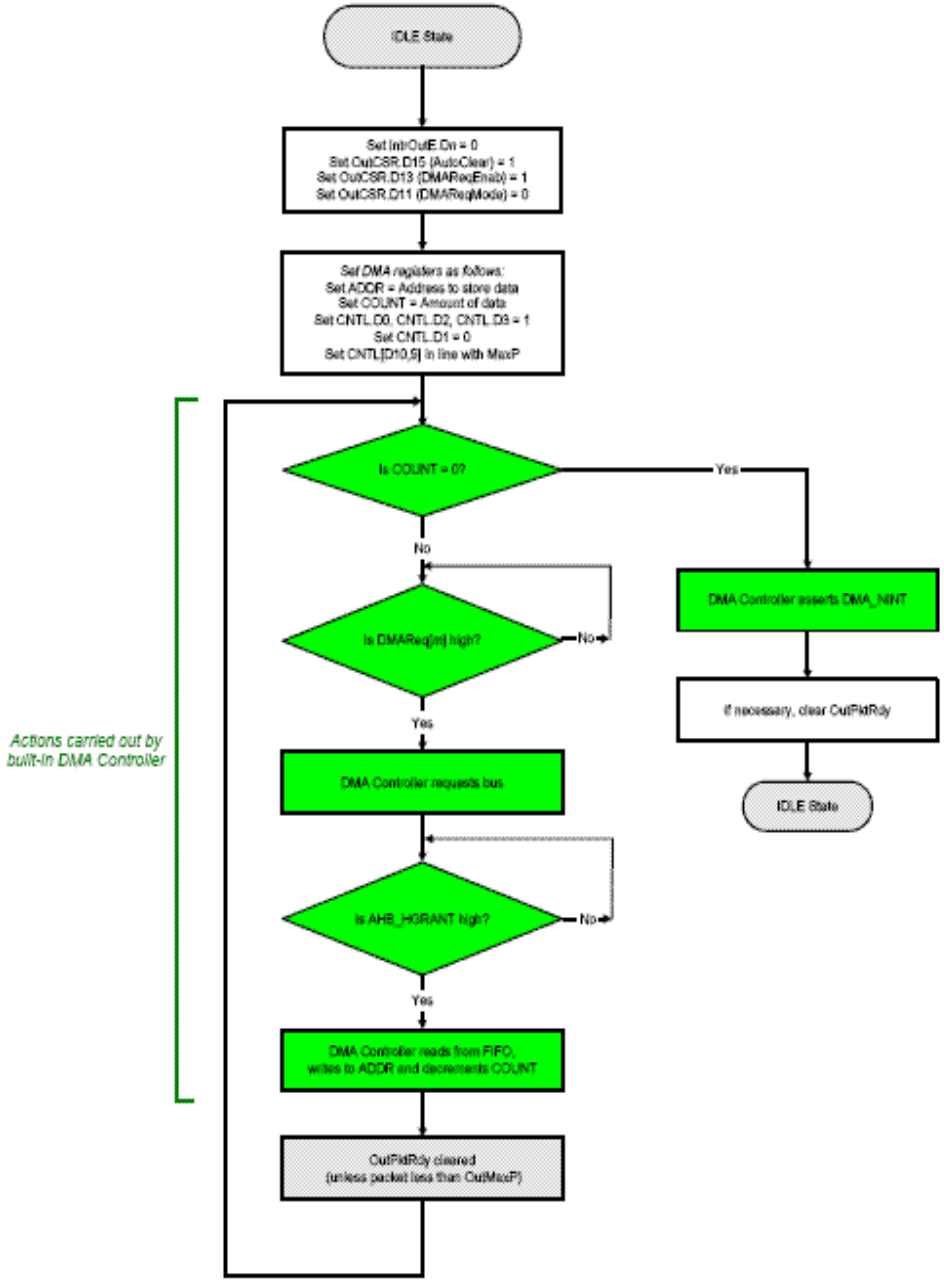


21.10.3.3 MULTIPLE IN PAC KETS

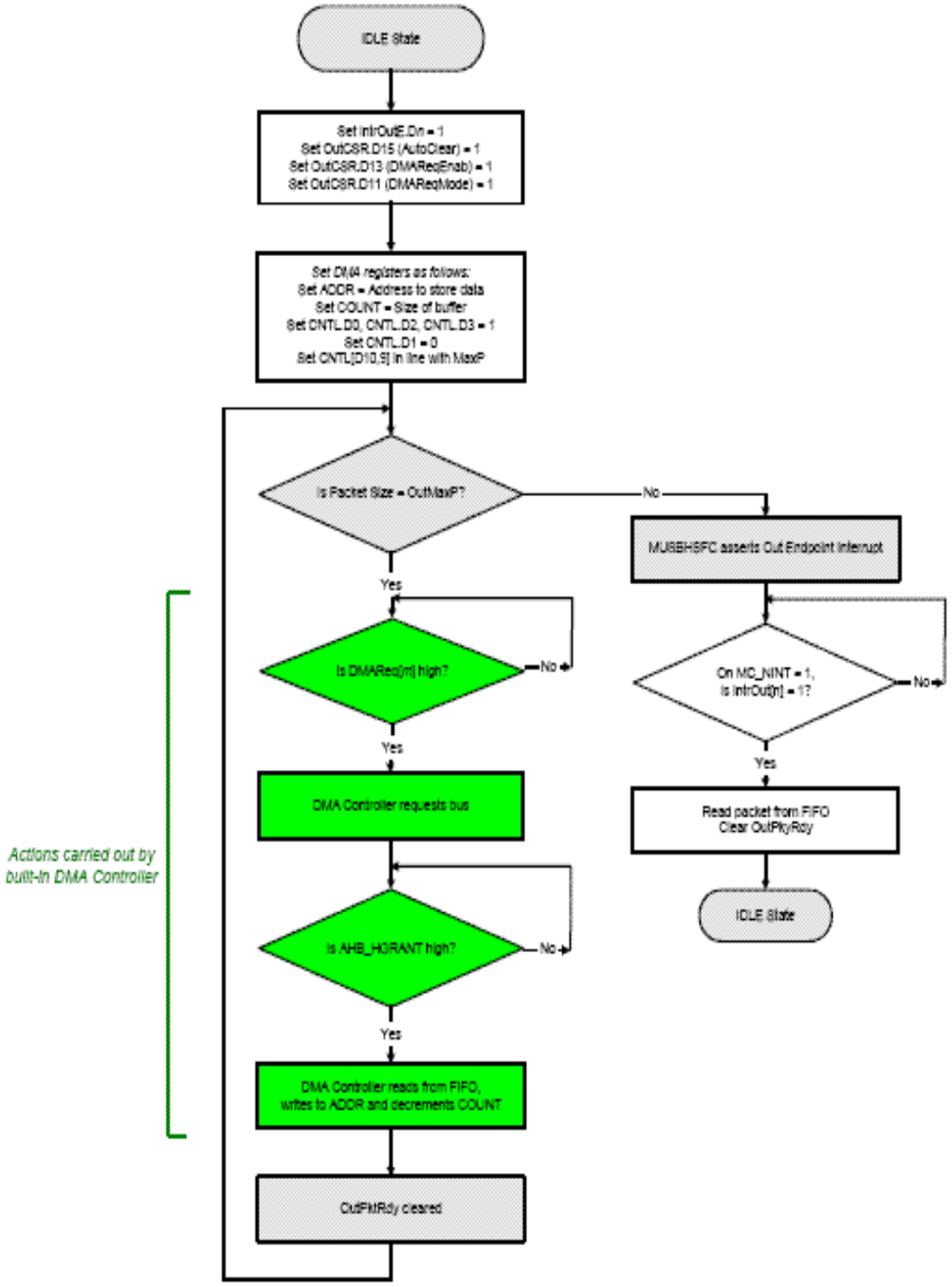


21.10.3.4 MULTIPLE OUT PACKETS

If Size of Data Block Known:



If Size of Data Block not Known:



21.11 TESTMODES

The CORE supports the four USB 2.0 test modes defined for High-speed functions. The test modes are entered by writing to the TestMode register (address 0Fh). A test mode is usually requested by the host sending a SET_FEATURE request to Endpoint 0. When the software receives the request, it should wait until the Endpoint 0 transfer has completed (when it receives the Endpoint 0 interrupt indicating that the status phase has completed) then write to the TestMode register.

NOTE: These test modes have no purpose in normal operation.

21.11.1 TESTMODETEST_SE0_ NAK

To enter the Test_SE0_NAK test mode, the software should set the Test_SE0_NAK bit by writing 6'h01 to the TestMode register. The CORE will then go into a mode in which it responds to any valid IN token with a NAK.

21.11.2 TESTMODETEST_J

To enter the Test_J test mode, the software should set the Test_J bit by writing 6'h02 to the TestMode register. The CORE will then go into a mode in which it transmits a continuous J on the bus.

21.11.3 TESTMODETEST_K

To enter the Test_K test mode, the software should set the Test_K bit by writing 6'h04 to the TestMode register. The CORE will then go into a mode in which it transmits a continuous K on the bus.

21.11.4 TESTMODETEST _ PACKET

To execute the Test_Packet test, the software should first write the standard test packet (shown below) to the Endpoint 0 FIFO and set the InPktRdy bit in the CSR0 register (D1). It should then write 6'h08 to the TestMode register to enter Test_Packet test mode.

The 53 byte test packet to load is as follows (all bytes in hex). The test packet only has to be loaded once, the CORE will keep re-sending the test packet without any further intervention from the software.

00 00 00 00 00 00 00 00
00 AA AA AA AA AA AA AA
AA EE EE EE EE EE EE EE
EE FE FF FF FF FF FF FF
FF FF FF FF FF 7F BF DF
EF F7 FB FD FC 7E BF DF
EF F7 FB FD 7E

This data sequence is defined in *Universal Serial Bus Specification* Revision 2.0.

22 MMC/SD CE-ATA Controller

22.1 Overview

The MultiMediaCard (MMC) is a universal low cost data storage and communication media that is designed to cover a wide area of applications such as electronic toys, organizers, PDAs, smart phones, and so on.

The Secure Digital (SD) card is an evolution of MMC, It is specifically designed to meet the security, capacity, performance, and environmental requirements inherent in newly emerging audio and video consumer electronic devices. The physical form factor, pin assignment, and data transfer protocol are forward compatible with the MultiMediaCard with some additions. An SD card can be categorized as SD memory or SD I/O card, commonly known as SDIO. A memory card invokes a copyright protection mechanism that complies with the security of the SDMI standard and is faster and capable of higher memory capacity. The SDIO card provides high-speed data I/O with low-power consumption for mobile electronic devices.

For CE-ATA detail protocol , please referred to WWW.CE-ATA.ORG.

Features of the MSC Controller include the following:

- Fully compatible with the *MMC System Specification version 4.2*
- Fully compatible with the *SD Memory Card Specification 2.0* and *SD I/O Specification 1.0* with 1 command channel and 4 data channels
- Consumer Electronics Advanced Transport Architecture (CE-ATA – version 1.1)
- 20-80 Mbps maximum data rate
- Support MMC data width 1bit ,4bit and 8bit
- Built-in programmable frequency divider for MMC/SD bus
- Maskable hardware interrupt for SDIO interrupt, internal status and FIFO status
- 32-entry x 32-bit built-in data FIFO
- Multi-SD function support including multiple I/O and combined I/O and memory
- IRQ supported enable card to interrupt MMC/SD controller
- Single or multi block access to the card including erase operation
- Stream access to the MMC card
- Supports SDIO read wait, interrupt detection during 1-bit or 4-bit access
- Supports CE-ATA digital protocol commands
- Support Command Completion Signal and interrupt to CPU
- Command Completion Signal disable feature
- The maximum block length is 4096bytes

22.2 Block Diagram

MSC Controller Block Diagram

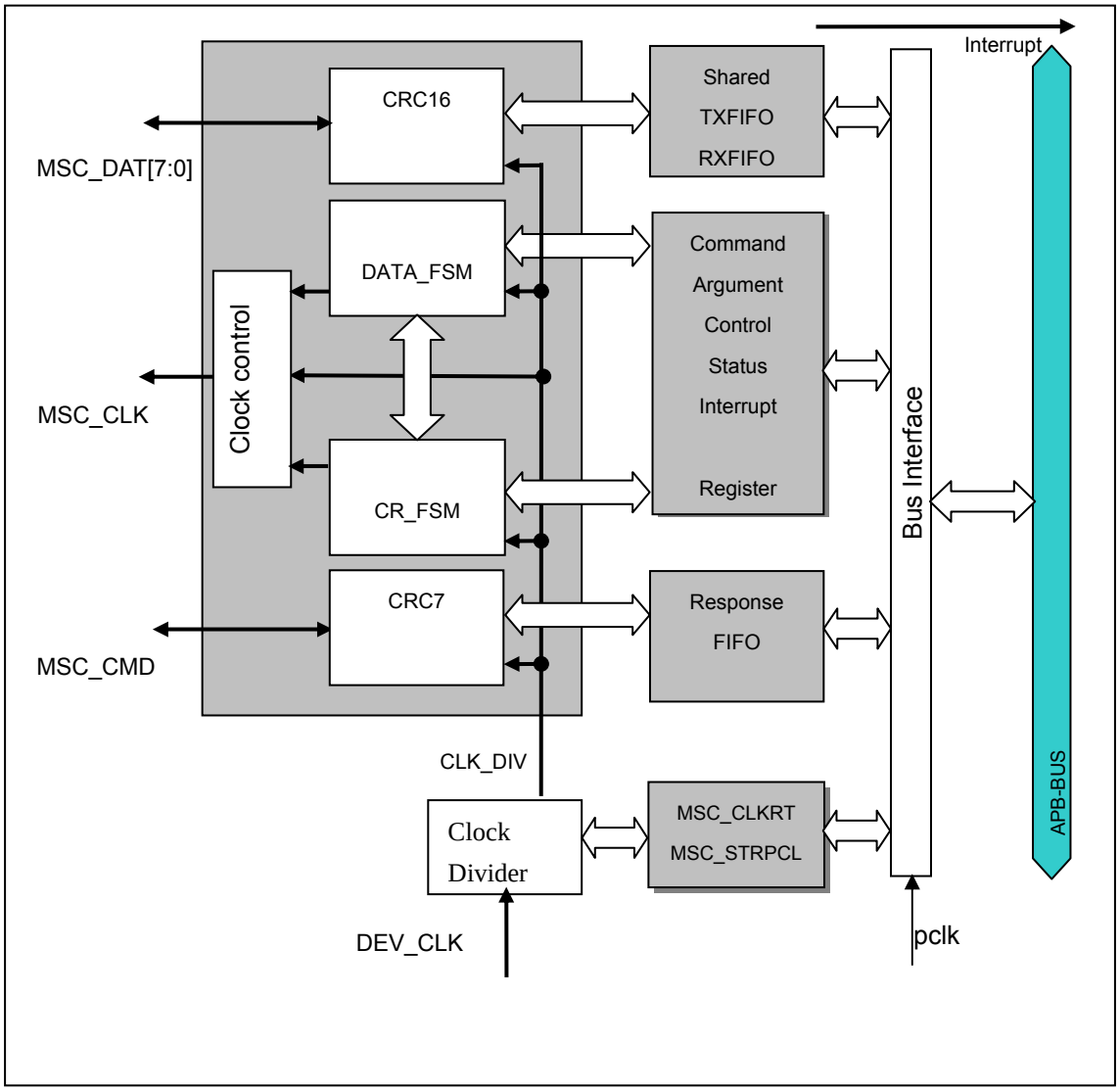


Figure 22-1 MMC/SD CE-ATA Controller Block Diagram

22.3 MMC/SD Controller Signal I/O Description

MSC and the card communication over the CMD and DATA line is base on command and data bit streams which are initiated by a start bit and terminated by a stop bit.

Command: a command is a token, which starts an operation. A command is sent from MSC either to a single card (addressed command) or to all connected cards (broadcast command). A command is transferred serially on the CMD line. Each command token is preceded by a start bit (‘0’) and succeeded by an end bit (‘1’). The total length is 48 bits and protected by CRC bits.

Table 22-1 Command Token Format

Bit position	**47**	**46**	**[45 : 40]**	**[39 : 8]**	**[7 : 1]**	**0**
Width (bits)	1	1	6	32	7	1
Value	0	1	X	X	x	1
Description	Start bit	Transmission bit	Command index	argument	CRC7	End bit

Response: a response is a token which is sent from an addressed card, or (synchronously) from all connected cards, to MSC as an answer to a previously received command. A response is transferred serially on the CMD line. Response tokens have varies coding schemes depending on their content.

Data: data can be transferred from the card to MSC or vice versa. Data is transferred via the data line. Data transfers to/from the SD Memory Card are done in blocks. Data blocks always succeeded by CRC bits. Single and multiple block operations are defined. Note that the Multiple Block operation mode is better for faster write operation. A multiple block transmission is terminated when a stop command follows on the CMD line. Data transfer can be configured by the MSC to use single or multiple data lines.

Table 22-2 MMC/SD Data Token Format

Description	**Start bit**	**Data**	**CRC16**	**End bit**
Stream Data	0	X	no CRC	1
Block Data	0	X	X	1

22.4 Register Description

The MMC-SD-CE_ATA controller is controlled by a set of registers that the application configures before every operation. The Table 22-3 lists all the MSC registers.

Table 22-3 MMC/SD Controller Registers Description

Name	RW	Reset Value	Address	Access Size
MSC_CTRL0	W	0x0000	0x10021000	16
MSC_STAT0	R	0x00000040	0x10021004	32
MSC_CLKRT0	RW	0x0000	0x10021008	16
MSC_CMDAT0	RW	0x00000000	0x1002100C	32
MSC_RESTO0	RW	0x40	0x10021010	16
MSC_RDTO0	RW	0xFFFF	0x10021014	32
MSC_BLKLEN0	RW	0x0000	0x10021018	16
MSC_NOB0	RW	0x0000	0x1002101C	16
MSC_SNOB0	R	0x????	0x10021020	16
MSC_IMASK0	RW	0x00FF	0x10021024	32
MSC_IREG0	RW	0x0000	0x10021028	16
MSC_CMD0	RW	0x00	0x1002102C	8
MSC_ARG0	RW	0x00000000	0x10021030	32
MSC_RES0	R	0x????	0x10021034	16
MSC_RXFIFO0	R	0x????????	0x10021038	32
MSC_TXFIFO0	W	0x????????	0x1002103C	32
MSC_LPM0	RW	0x00000000	0x10021040	32
MSC_CTRL1	W	0x0000	0x10022000	16
MSC_STAT1	R	0x00000040	0x10022004	32
MSC_CLKRT1	RW	0x0000	0x10022008	16
MSC_CMDAT1	RW	0x00000000	0x1002200C	32
MSC_RESTO1	RW	0x40	0x10022010	16
MSC_RDTO1	RW	0xFFFF	0x10022014	32
MSC_BLKLEN1	RW	0x0000	0x10022018	16
MSC_NOB1	RW	0x0000	0x1002201C	16
MSC_SNOB1	R	0x????	0x10022020	16
MSC_IMASK1	RW	0x00FF	0x10022024	32
MSC_IREG1	RW	0x0000	0x10022028	16
MSC_CMD1	RW	0x00	0x1002202C	8
MSC_ARG1	RW	0x00000000	0x10022030	32
MSC_RES1	R	0x????	0x10022034	16
MSC_RXFIFO1	R	0x????????	0x10022038	32
MSC_TXFIFO1	W	0x????????	0x1002203C	32
MSC_LPM1	RW	0x00000000	0x10022040	32

22.4.1 MMC/SD Control Register (MSC_CTRL)

MSC_CTRL0 **0x10021000**

MSC_CTRL1 **0x10022000**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								SEND_CCSD	SEND_AD_CCSD	Reserved		EXIT_MULTIPLE	EXIT_TRANSFER	START_READWAIT	STOP_READWAIT	RESET	START_OP	STRPCL	
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15	SEND_CCSD	0: clear bit 1: Send Command Completion Signal Disable (CCSD) to CE_ATA device when set, host sends CCSD to CE_ATA device. Software set the bit only if current command is expecting CCS and interrupts are enabled in CE_ATA devices. Once the CCSD pattern is sent to device, host automatically clears the SEND_CCSD bit.	W
14	SEND_AS_CCSD	0: clear bit 1: send internally generated stop after sending CCSD to CE_ATA device When set, host automatically sends internally-generated STOP command(CMD12) to CE_ATA device. After sending CMD12, Auto Command Done (ACD) is set and generates interrupt to CPU. After sending the CCSD, controller automatically clears the SEND_AS_CCSD bit.	W
13:8	Reserved		R
7	EXIT_MULTIPLE	If CMD12 or CMD52 (I/O abort) is to be sent to terminate multiple block read/write in advance, set this bit to 1. 0: No effect 1: Exit from multiple block read/write	W
6	EXIT_TRANSFER	**Only used for SDIO suspend/resume and MMC stream read.** For SDIO, after suspend is accepted, set this bit with 1. For MMC, after the expected number of data are received, set this bit with 1. 0: No effect 1: Exit from multiple block read/write after suspend is accepted, or exit from stream read	W
5	START_READWAIT	Only used for SDIO ReadWait. Start the ReadWait cycle. 0: No effect	W

		1: Start ReadWait	
4	STOP_READWAIT	Only used for SDIO ReadWait. Stop the ReadWait cycle. 0: No effect 1: Start ReadWait	W
3	RESET	Resets the MMC/SD controller. 0: No effect 1: Reset the MMC/SD controller	W
2	START_OP	This bit is used to start the new operation. When starting the clock, this bit can be 1. When stopping the clock, this bit can only be 0. 0: Do nothing 1: Start the new operation	W
1:0	CLOCK_CONTROL	These bits are used to start or stop clock. 00: Do nothing 01: Stop MMC/SD clock 10: Start MMC/SD clock 11: Reserved	W

22.4.2 MSC Status Register (MSC_STAT)

MSC_STAT0 **0x10021004**

MSC_STAT1 **0x10022004**

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	AUTO_CMD_DONE	RESERVED															IS_RESETTING	SDIO_INT_ACTIVE	PRG_DONE	DATA_TRAN_DONE	END_CMD_RES	DATA_FIFO_AFULL	IS_READWAIT	CLK_EN	DATA_FIFO_FULL	DATA_FIFO_EMPTY	CRC_RES_ERR	CRC_READ_ERROR	CRC_WRITE_ERROR	TIME_OUT_RES	TIME_OUTREAD	
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

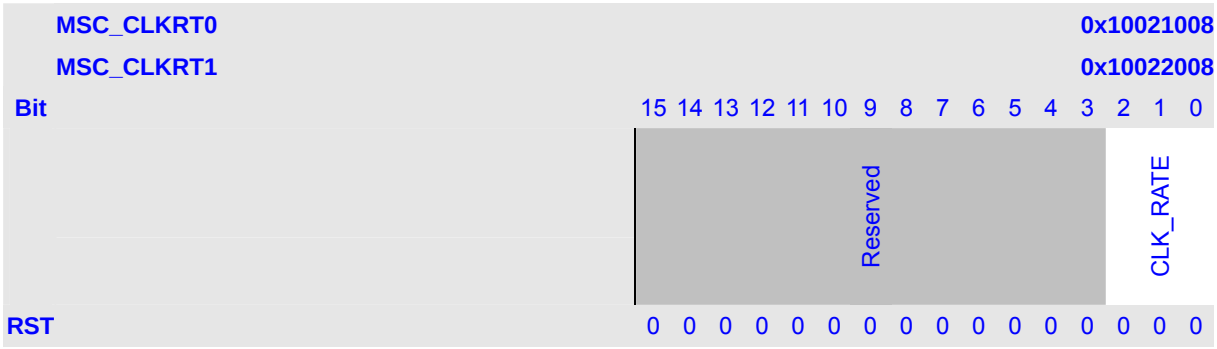
Bits	Name	Description	RW
31	AUTO_CMD_DONE	Indicate that the stop command (CMD12) that is internally generated by controller has finished.	R
30:16	Reserved		R
15	IS_RESETTING	MSC is resetting after power up or MSC_STRPCL[RESET] is written with 1. 0: Reset has been finished 1: Reset has not been finished	R
14	SDIO_INT_ACTIVE	Indicates whether an interrupt is detected at the SD I/O	R

		card. A separate acknowledge command to the card is required to clear this interrupt. 0: No interrupt detected 1: The interrupt from SDIO is detected	
13	PRG_DONE	Indicates whether card has finished programming. 0: Card has not finished programming and is busy 1: Card has finished programming and is not busy	R
12	DATA_TRAN_DONE	Indicates whether data transmission to card has completed. 0: Data transmission to card has not completed 1: Data transmission to card has completed	R
11	END_CMD_RES	End command-response sequence or command sequence. 0: Command and response/no-response sequence has not completed 1: Command and response/no-response sequence has completed	R
10	DATA_FIFO_AFULL	Indicates whether data FIFO is almost full (The number of words >= 15). For reading data from card, use this bit. 0: Data FIFO is not full 1: Data FIFO is full	R
9	IS_READWAIT	Indicates whether SDIO card has entered ReadWait State. 0: Card has not entered ReadWait 1: Card has entered ReadWait	R
8	CLK_EN	Clock enabled. 0: Clock is off 1: Clock is on	R
7	DATA_FIFO_FULL	Indicates whether data FIFO is full. For reading data from card, do not use this bit, because it almost keeps to be 0. 0: Data FIFO is not full 1: Data FIFO is full	R
6	DATA_FIFO_EMPTY	Indicates whether data FIFO is empty. 0: Data FIFO is not empty 1: Data FIFO is empty	R
5	CRC_RES_ERR	Response CRC error. 0: No error on the response CRC 1: CRC error occurred on the response	R
4	CRC_READ_ERROR	CRC read error. 0: No error on received data 1: CRC error occurred on received data	R
3:2	CRC_WRITE_ERROR	CRC write error. 00: No error on transmission of data	R

		01: Card observed erroneous transmission of data 10: No CRC status is sent back 11: Reserved	
1	TIME_OUT_RES	Response time out. 0: Card response has not timed out 1: Card response has time out	R
0	TIME_OUT_READ	Read time out. 0: Card read data has not timed out 1: Card read data has timed out	R

22.4.3 MSC Clock Rate Register (MSC_CLKRT)

The MSC_CLKRT register specifies the frequency division of the MMC/SD bus clock. The software is responsible for setting this register.

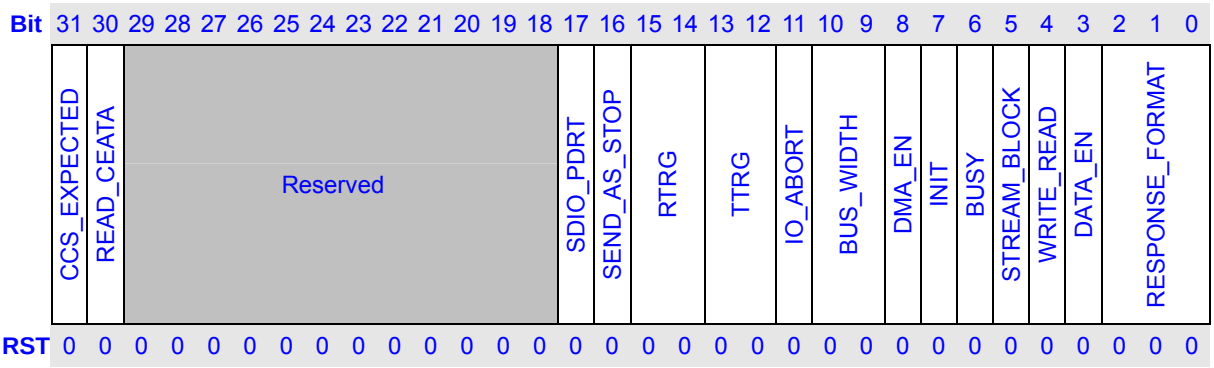


Bits	Name	Description	RW
15:3	Reserved		R
2:0	CLK_RATE	Clock rate. 000: CLK_SRC 001: 1/2 of CLK_SRC 010: 1/4 of CLK_SRC 011: 1/8 of CLK_SRC 100: 1/16 of CLK_SRC 101: 1/32 of CLK_SRC 110: 1/64 of CLK_SRC 111: 1/128 of CLK_SRC	WR

22.4.4 MMC/SD Command and Data Control Register (MSC_CMDAT)

MSC_CMDAT0 **0x1002100C**

MSC_CMDAT1 **0x1002200C**



Bits	Name	Description	RW
31	CCS_EXPECTED	0: interrupts are not enabled in CE-ATA device, or commands does not expect CCS from device 1: interrupts are enabled in CE_ATA device, or RW_BLK command expects command completion signal from device If the command expects Command Completion Signal (CCS) from the device, the software should set the control bit. It is auto cleared 0 by hardware.	RW
30	READ_CEATA	0: host is not performing read access (RW_BLK or RW_REG) towards CE_ATA device 1: host id performing read access (RW_BLK or RW_REG) towards CE_ATA device Software should set the bit to indicate that CE_ATA device is being accessed for read transfer. The bit is used to disable read data timeout indication while performing CE_ATA read transfers. It is auto cleared 0 by hardware.	RW
29:18	Reserved		R
17	SDIO_PRDT	Determine whether SDIO interrupt is 2 cycle or extend more cycle when data block last is transferred. 0: more cycle (like single block) 1: exact 2 cycle	RW
16	SEND_AS_STOP	0: no stop command sent at end of data transfer 1: send stop command at end of data transfer when stop command has finished, it is auto cleared 0 by hardware.	RW
15:14	RTRG	These bits set the receive FIFO half-empty threshold value, when the number of transmit FIFO >= threshold	RW

		value , RXFIFO_RD_REQ will be set to 1. 00 : more than or equal to 8 01: more than or equal to 16 10: more than or equal to 24 11: reserved	
13:12	TTRG	These bits set the transmit FIFO half-empty threshold value, when the number of transmit FIFO < threshold value , TXFIFO_WR_REQ will be set to 1. 00 : less than 8 01: less than 16 10: less than 24 11: reserved	RW
11	STOP_ABORT	Specifies the current command is used to abort data transfer. 0: Nothing 1: The current command is used to abort transfer it is auto cleared 0 by hardware.	WR
10:9	BUS_WIDTH	Specifies the width of the data bus. 00: 1-bit 01: Reserved 10: 4-bit 11: 8bit	WR
8	DMA_EN	DMA mode enables. When DMA mode is used, this bit is also a mask on RXFIFO_RD_REQ and TXFIFO_WR_REQ interrupts. 0: Program I/O 1: DMA mode	WR
7	INIT	80 initialization clocks. 0: Do not precede command sequence with 80 clocks 1: Precede command sequence with 80 clocks	W
6	BUSY	Specifies whether a busy signal is expected after the current command. This bit is for no data command/response transactions only. 0: Not expect a busy signal 1: Expects a busy signal. If the response is R1b, then set it	WR
5	STREAM_BLOCK	Stream mode. 0: Data transfer of the current command sequence is not in stream mode 1: Data transfer of the current command sequence is in stream mode	WR
4	WRITE_READ	Specifies that the data transfer of the current command is a read or write operation. 0: Specifies that the data transfer of the current command	WR

		is a read operation 1: Specifies that the data transfer of the current command is a write operation	
3	DATA_EN	Specifies whether the current command includes a data transfer. It is also used to reset RX_FIFO and TX_FIFO. 0: No data transfer with current command 1: Has data transfer with current command. It is also used to reset RX_FIFO and TX_FIFO	WR
2:0	RESPONSE_FORMAT	These bit specify the response format for the current command. 000: No response 001: Format R1 and R1b 010: Format R2 011: Format R3 100: Format R4 101: Format R5 110: Format R6 111: Format R7	WR

22.4.5 MMC/SD Response Time Out Register (MSC_RESTO)

MSC_RESTO0 **0x10021010**

MSC_RESTO1 **0x10022010**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								RES_TO							
RST	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Bits	Name	Description	RW
15:8	Reserved		R
7:0	RES_TO	Specifies the number of MSC_CLK clock counts between the command and when the MMC/SD controller turns on the time-out error for the received response. The default value is 64.	WR

22.4.6 MMC/SD Read Time Out Register (MSC_RDTO)

MSC_RDTO0 **0x10021014**
MSC_RDTO1 **0x10022014**

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

Bits	Name	Description	RW
31:0	READ_TO	Specifies the number of clocks between the command and when the MMC/SD host controller turns on the time-out error for the received data. The unit is MSC_CLK.	WR

22.4.7 MMC/SD Block Length Register (MSC_BLKLEN)

MSC_BLKLEN0 **0x10021018**
MSC_BLKLEN1 **0x10022018**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	BLK_LEN															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15:0	BLK_LEN	Specifies the number of bytes in a block, and is normally set to 0x200 for MMC/SD data transactions. The value Specified in the cards CSD.	WR

22.4.8 MSC/SD Number of Block Register (MSC_NOB)

MSC_NOB0 **0x1002101C**
MSC_NOB1 **0x1002201C**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NOB															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15:0	NOB	Specifies the number of blocks in a data transfer. One block is a possibility.	WR

22.4.9 MMC/SD Number of Successfully-transferred Blocks Register (MSC_SNOB)

In block mode, the MSC_SNOB register records the number of successfully transferred blocks. If the last block has CRC error, this register also summaries it. It is used to query blocks for multiple block transfer.

MSC_SNOB0 **0x10021020**

MSC_SNOB1 **0x10022020**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MSC_SNOB															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
15:0	MSC_SNOB	Specify the number of successfully transferred blocks for a multiple block transfer.	R

22.4.10 MMC/SD Interrupt Mask Register (MSC_IMASK)

MSC_IMASK0 **0x100210240**

MSC_IMASK1 **0x10022024**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved	AUTO_CMD_DONE	DATA_FIFO_FULL	DATA_FIFO_EMP	CRC_RES_ERR	CRC_READ_ERR	CRC_WRITE_ERR	TIME_OUT_RES	TIME_OUT_READ	SDIO	TXFIFO_WR_REQ	RXFIFO_RD_REQ	Reserved	END_CMD_RES	PRG_DONE	DATA_TRAN_DONE
RST	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	Name	Description	RW
31:16	Reserved		R
15	AUTO_CMD_DONE	Mask the interrupt Auto Cmd Done (ACD). 0: Not masked 1: Masked	RW
14	DATA_FIFO_FULL	0: Not masked 1: Masked	RW

13	DATA_FIFO_EMP	0: Not masked 1: Masked	RW
12	CRC_RES_ERR	0: Not masked 1: Masked	RW
11	CRC_READ_ERR	0: Not masked 1: Masked	RW
10	CRC_WRITE_ERR	0: Not masked 1: Masked	RW
9	TIME_OUT_RES	0: Not masked 1: Masked	RW
8	TIME_OUT_READ	0: Not masked 1: Masked	RW
7	SDIO	Mask the interrupt from the SD I/O card. 0: Not masked 1: Masked	WR
6	TXFIFO_WR_REQ	Mask the Transmit FIFO write request interrupt. 0: Not masked 1: Masked	WR
5	RXFIFO_RD_REQ	Mask the Receive FIFO read request interrupt. 0: Not masked 1: Masked	WR
4:3	Reserved		R
2	END_CMD_RES	Mask the End command response interrupt. 0: Not masked 1: Masked	WR
1	PRG_DONE	Mask the Programming done interrupt. 0: Not masked 1: Masked	WR
0	DATA_TRAN_DONE	Mask the Data transfer done interrupt. 0: Not masked 1: Masked	WR

22.4.11 MMC/SD Interrupt Register (MSC_IREG)

The MSC_IREG register shows the currently requested interrupt. The FIFO request interrupts, TXFIFO_WR_REQ, and RXFIFO_RD_REQ are masked off with the DMA_EN bit in the MSC_CMDAT register. The software is responsible for monitoring these bit in program I/O mode.

MSC_IREG0 **0x10021028**

MSC_IREG1 **0x10022028**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved	AUTO_CMD_DONE	DATA_FIFO_FULL	DATA_FIFO_EMP	CRC_RES_ERR	CRC_READ_ERR	CRC_WRITE_ERR	TIME_OUT_RES	TIME_OUT_READ	SDIO	TXFIFO_WR_REQ	RXFIFO_RD_REQ	Reserved	END_CMD_RES	PRG_DONE	DATA_TRAN_DONE
RST		0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15	AUTO_CMD_DONE	indicate Auto Cmd Done (ACD) interrupt. 0: the interrupt is not detected 1: the interrupt is detected	RW
14	DATA_FIFO_FULL	Indicate data FIFO is full interrupt. 0: the interrupt is not detected 1: the interrupt is detected	R
13	DATA_FIFO_EMP	Indicate data FIFO is empty interrupt. 0: the interrupt is not detected 1: the interrupt is detected	R
12	CRC_RES_ERR	Indicate response CRC error interrupt. 0: the interrupt is not detected 1: the interrupt is detected	RW
11	CRC_READ_ERR	Indicate CRC read error interrupt. 0: the interrupt is not detected 1: the interrupt is detected	RW
10	CRC_WRITE_ERR	Indicate CRC write error interrupt. 0: the interrupt is not detected 1: the interrupt is detected	RW
9	TIME_OUT_RES	Indicate response time out interrupt. 0: the interrupt is not detected 1: the interrupt is detected	RW
8	TIME_OUT_READ	Indicate read time out interrupt. 0: the interrupt is not detected 1: the interrupt is detected	RW
7	SDIO	Indicates whether the interrupt from SDIO is detected.	R

		0: The interrupt from SDIO is not detected 1: The interrupt from SDIO is detected	
6	TXFIFO_WR_REQ	Transmit FIFO write request. Set if data FIFO becomes half empty. (the number of words is < 8) 0: No Request for data Write to MSC_TXFIFO 1: Request for data write to MSC_TXFIFO	R
5	RXFIFO_RD_REQ	Receive FIFO read request. Set if data FIFO becomes half full (the number of words is >= 8) or the entries in data FIFO are the last read data. 0: No Request for data read from MSC_RXFIFO 1: Request for data read from MSC_RXFIFO	R
4:3	Reserved		R
2	END_CMD_RES	Indicates whether the command/response sequence has been finished. 0: The command/response sequence has not been finished 1: The command/response sequence has been finished Write 1 to clear.	WR
1	PRG_DONE	Indicates whether card has finished programming. 0: Card has not finished programming and is busy 1: Card has finished programming and is no longer busy Write 1 to clear.	WR
0	DATA_TRAN_DONE	Indicates whether data transfer is done. Note that for stream read/write, only when CMD12 (STOP_TRANS) has been sent, is this bit set. 0: Data transfer is not complete 1: Data transfer has completed or an error has occurred Write 1 to clear.	WR

22.4.12 MMC/SD Command Index Register (MSC_CMD)

MSC_CMD0 **0x1002102C**
MSC_CMD1 **0x1002202C**

Bit	7	6	5	4	3	2	1	0
			Reserved	CMD_INDEX				
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7:6	Reserved		R
5:0	CMD_INDEX	Specifies the command index to be executed.	WR

22.4.13 MMC/SD Command Argument Register (MSC_ARG)

MSC_ARG0																														0x10021030	
MSC_ARG1																														0x10022030	
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ARG																															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:0	ARG	Specifies the argument for the current command.	WR

22.4.14 MMC/SD Response FIFO Register (MSC_RES)

The read-only MMC/SD Response FIFO register (RES_FIFO) holds the response sent back to the MMC/SD controller after every command. The size of this FIFO is 8 x 16-bit. The RES FIFO does not contain the 7-bit CRC for the response. The Status for CRC checking and response time-out status is in the status register, MSC_STAT.

The first halt-word read from the response FIFO is the most significant halt-word of the received response.

MSC_RES0																														0x10021034	
MSC_RES1																														0x10022034	
Bit																	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																	DATA															
RST																	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
15:0	DATA	Contains the response to every command that is sent by the MMC/SD controller. The size of this FIFO register is 8 x 16-bit.	R

22.4.15 MMC/SD Receive Data FIFO Register (MSC_RXFIFO)

The MSC_RXFIFO is used to read the data from a card. It is read-only to the software, and is read on 32-bit boundary. The size of this FIFO is 16 x 32-bit.

MSC_RXFIFO0																														0x10021038	
MSC_RXFIFO1																														0x10022038	
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATA																															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
31:0	DATA	One word of read data. The size of this FIFO is 16 x 32-bit.	R

22.4.16 MMC/SD Transmit Data FIFO Register (MSC_TXFIFO)

The MSC_TXFIFO is used to write the data to a card. It is write-only to the software, and is written on 32-bit boundary. The size of this FIFO is 16 x 32-bit.

MSC_TXFIFO0																														0x1002103C	
MSC_TXFIFO1																														0x1002203C	
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATA																															
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
31:0	DATA	One word of write data. The size of this FIFO is 16 x 32-bit.	W

22.4.17 MMC/SD Low Power Mode Register (MSC_LPM)

The MSC_LPM is used to control whether MSC controller enters Low-Power Mode.

MSC_LPM0																														0x10021040	
MSC_LPM1																														0x10022040	
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																															LPM
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:1	Reserved		R
0	LPM	0 : Non –Low Power Mode 1: Low-Power Mode. Stop clock when card in idle (should be normally set to only MMC and SD cards. For SDIO cards, if interrupts must be detected, clock should not be stopped) When software sets the bit, MSC clock can auto be stopped. **NOTE:** when set the bit, the start_clock and stop clock can be not use.	RW

22.5 MMC/SD Functional Description

All communication between system and cards is controlled by the MSC. The MSC sends commands of two type: broadcast and addressed (point-to-point) commands.

Broadcast commands are intended for all cards, command like “Go_Idle_State”, “Send_Op_Cond”, “All_send_CID” and “Set_relative_Addr” are using way of broadcasting. During Broadcast mode, all cards are in open-drain mode, to avoid bus contention.

After Broadcast commands “Set_relative_Addr” issue, cards are enter standby mode, and Addressed command will be used from now on, in this mode, CMD/DAT will return to push-pull mode, to have maximum driving for maximum operation frequency.

The MMC and the SD are similar product. Besides the 4x bandwidth and the built-in encryption, they are being programmed similarly.

The MMC/SD controller (MSC) is the interface between the software and the MMC/SD bus. It is responsible for the timing and protocol between the software and the MMC/SD bus. It consists of control and status registers, a 16-bit response FIFO that is 8 entries deep, and one 32-bit receive/transmit data FIFOs that are 16 entries deep. The registers and FIFOs are accessible by the software.

MSC also enable minimal data latency by buffering data and generating and checking CRCs.

22.5.1 MSC Reset

The MMC/SD controller (MSC) can be reset by a hardware reset or software reset. All registers and FIFO controls are set to their default values after any reset.

22.5.2 MSC Card Reset

The command Go_Idle_State, CMD0 is the software reset command for MMC and SD Memory Card, and sets each card into Idle State regardless of the current card state; while in SDIO card, CMD52 is used to write IO reset in CCCR. The cards are initialized with a default relative card address (RCA=0x0000) and with a default driver stage register setting (lowest speed, highest driving current capability).

22.5.3 Voltage Validation

All cards shall be able to establish communication with the host using any operation voltage in the maximal allowed voltage range specified in this standard. However, the support minimum and maximum values for Vdd are defined in Operation Conditions register (OCR) and many not cover the whole range. Cards that store the CID and CSD data in the payload memory would be able to communicate these information only under data transfer Vdd conditions. That means if host and card

have non compatible Vdd ranges, the card will not be able to complete the identification cycle, nor to send CSD data.

Therefore, a special command Send_Op_cont (CMD1 for MMC), SD_Send_Op_Cont (CMD41 for SD Memory) and IO_Send_Op_Cont (CMD5 for SDIO) are designed to provide a mechanism to identify and reject cards which do not match the Vdd range desired by the host. This is accomplished by the host sending the required Vdd voltage window as the operand of this command. Cards which can not perform data transfer in the specified range must discard themselves from further bus operations and go into Inactive State. By omitting the voltage range in the command, the host can query each card and determine the common voltage range before sending out-of-range cards into the Inactive State. This query should be used if the host is able to select a common voltage range or if a notification to the application of non usable cards in the stack is desired.

22.5.4 Card Registry

Card registry on MCC and SD card are different.

For SD card, Identification process start at clock rate Fod, while CMD line output drives are push-pull drivers instead of open-drain. After the bus is activated the host will request the cards to send their valid operation conditions. The response to ACMD41 is the operation condition register of the card. The same command shall be send to all of the new cards in the system. Incompatible cards are sent into Inactive State. The host then issue the command All_Send_CID (CMD2) to each card and get its unique card identification (CID) number. Card that is unidentified, that is, which is in Ready State, send its CID number as the response. After the CID was sent by the card it goes into Identification State. Thereafter, the host issues Send_Relative_Addr (CMD3) asks the card to publish a new relative card address (RCA), which is shorter that CID and which will be used to address the card in the future data transfer mode. Once the RCA is received the card state changes to the Stand-by State. At this point, if the host wants that the card will have another RCA number, it may ask the card to publish a new number by sending another Send_Relative_Addr command to the card. The last published RCA is the actual RCA of the card. The host repeats the identification process, that is, the cycles with CMD2 and CMD3 for each card in the system.

In MMC, the host starts the card identification process in open-drain mode with the identification clock rate Fod. The open drain driver stages on the CMD line allow parallel card operation during card identification. After the bus is active the host will request the cards to send their valid operation conditions (CMD1). The response to CMD1 is the ‘wired or’ operation on the condition restrictions of all cards in the system. Incompatible cards are sent into Inactive State. The host then issues the broadcast command All_Send_CID (CMD2), asking all cards for their unique card identification (CID) number. All unidentified cards, that is, those which are in Ready State, simultaneously start sending their CID numbers serially, while bit-wise monitoring their outgoing bitstream. Those cards, whose outgoing CID bits do not match the corresponding bits on the command line in any one of the bit periods stop sending their CID immediately and must wait for the next identification cycle. Since CID is unique for each card, only one card can be successfully send its full CID to the host. This card then

goes into Identification State. Thereafter, the host issues Set_Relative_Addr (CMD3) to assign to this card a relative card address (RCA). Once the RCA is received the card state changes to the Stand-by State, and the card does not react to further identification cycles, and its output switches from open-drain to push-pull. The host repeat the process, which is CM2 and CMD3, until the host receive time-out condition to recognize completion of the identification process.

22.5.5 Card Access

22.5.5.1 Block Access, Block Write and Block Read

During block write (CMD24-27) one or more blocks of data are transferred from the host to the card with a CRC appended to the end of each block by the host. A card supporting block write shall always be able to accept a block of data defined by WRITE_BL_LEN. If the CRC fails, the card shall indicate the failure on the DAT line; the transferred data will be discarded and not written, and all further transmitted blocks (in multiple block write mode) will be ignored.

Programming of the CID and CSD registers does not require a previous block length setting. The transferred data is also CRC protected. If a part of the CSD or CID register is stored in ROM, then this unchangeable part must match the corresponding part of the receive buffer. If this match fails, then the card will report an error and not change any register contents. Some cards may require long and unpredictable times to write a block of data. After receiving a block of data and completing the CRC check, the card will begin writing and hold the DAT line low if its write buffer is full and unable to accept new data from a new WRITE_BLOCK command. The host may poll the status of the card with a SEND_STATUS command (CMD13) at any time, and the card will respond with its status. The status bit READY_FOR_DATA indicates whether the card can accept new data or whether the write process is still in progress). The host may deselect the card by issuing CMD7 (to select a different card) which will displace the card into the Disconnect State and release the DAT line without interrupting the write operation. When reselecting the card, it will reactivate busy indication by pulling DAT to low if programming is still in progress and the write buffer is unavailable.

Block read is similar to stream read, except the basic unit of data transfer is a block whose maximizes is defined in the CSD (READ_BL_LEN). If READ_BL_PARTIAL is set, smaller blocks whose starting and ending address are entirely contained within one physical block (as defined by READ_BL_LEN) may also be transmitted. Unlike stream read, a CRC is appended to the end of each block ensuring data transfer integrity. CMD17 (READ_SINGLE_BLOCK) initiates a block read and after completing the transfer, the card returns to the Transfer state. CMD18 (READ_MULTIPLE_BLOCK) starts a transfer of several consecutive blocks. Blocks will be continuously transferred until a stop command is issued. If the host uses partial blocks whose accumulated length is not block aligned and block misalignment is not allowed, the card shall detect a block misalignment at the beginning of the first mis-aligned block, set the ADDRESS_ERROR error bit in the status register, abort transmission and wait in the Data State for a stop command.

22.5.5.2 Stream Access, Stream Write and Stream Read (MMC Only)

Stream write (CMD20) starts the data transfer from the host to the card beginning from the starting address until the host issues a stop command. Since the amount of data to be transferred is not determined in advance, CRC can not be used. If the end of the memory range is reached while sending data and no stop command has been sent by the host, all further transferred data is discarded.

There is a stream oriented data transfer controlled by READ_DAT_UNTIL_STOP (CMD11). This command instructs the card to send its payload, starting at a specified address, until the host sends a STOP_TRANSMISSION command (CMD12). The stop command has execution delay due to the serial command transmission. The data transfer stops after the end bit of the stop command. If the end of the memory range is reached while sending data and no stop command has been sent yet by the host, the contents of the further transferred payload is undefined.

22.5.5.3 Erase, Group Erase and Sector Erase (MMC Only)

It is desirable to erase many sectors simultaneously in order to enhance the data throughput. Identification of these sectors is accomplished with the TAG_* commands. Either an arbitrary set of sectors within a single erase group, or an arbitrary selection of erase groups may be erase at one time, but not both together. That is, the unit of measure for determining an erase is either a sector or an erase group. If a set of sectors must be erased, all selected sectors must lie within the same erase group. To facilitate selection, a first command with the starting address is followed by a second command with the final address, and all sectors (or groups) within this range will be selected for erase.

22.5.5.4 Wide Bus Selection/Deselection

Wide Bus (4 bit bus width) operation mode may be selected / deselected using ACMD6. The default bus width after power up or GO_IDLE (CMD0) is 1 bit bus width. ACMD6 command is valid in ‘trans state’ only. That means the bus width may be changed only after a card was selected (CMD7).

22.5.6 Protection Management

Three write protect methods are supported in the host for Cards, Card internal write protect (Card’s responsibility), Mechanical write protect switch (Host responsibility only) and Password protection card lock operation.

22.5.6.1 Card Internal Write Protection

Card data may be protected against either erase or write. The entire card may be permanently write protected by the manufacturer or content provider by setting the permanent or temporary write protect bits in the CSD. For cards which support write protection of groups of sectors by setting the WP_GRP_SIZE sectors as specified in the CSD), and the write protection may be changed by the application. The SET_WRITE_PROT command sets the write protection of the addressed

write-protect group, and the CLR_WRITE_PROT command clears the write protection of the addressed write-protect group.

The SEND_WRITE_PROT command is similar to a single block read command. The card shall send a data block containing 32 write protection bits (representing 32 write protect groups starting at the specified address) followed by 16 CRC bits. The address field in the write protect commands is a group address in byte units. The card will ignore all LSB's below the group size.

22.5.6.2 Mechanical write protect switch

A mechanical sliding tablet on the side of the card will be used by the user to indicate that a given card is write protected or not. If the sliding tablet is positioned in such a way that the window is open that means the card is write protected. If the window is close the card is not write protected.

A proper, matched, switch on the socket side will indicated to the host that the card is write protected or not. It is the responsibility of the host to protect the card. The position of the write protect switch is un-known to the internal circuitry of the card.

22.5.6.3 Password Protect

The password protection feature enables the host to lock a card while providing a password, which later will be used for unlocking the card. The password and its size is kept in an 128-bit PWD and 8-bit PWD_LEN registers, respectively. These registers are non-volatile so that a power cycle will not erase them.

Locked cards respond to (and execute) all commands in the basic command class (class 0) and "lock card" command class. Thus the host is allowed to reset, initialize, select, query for status, etc., but not to access data on the card. If the password was previously set (the value of PWD_LEN is not 0) will be locked automatically after power on. Similar to the existing CSD and CID register write commands the lock/unlock command is available in "trans_state" only. This means that it does not include an address argument and the card must be selected before using it. The card lock/unlock command has the structure and bus transaction type of a regular single block write command. The transferred data block includes all the required information of the command (password setting mode, PWD itself, card lock/unlock etc.). The following table describes the structure of the command data block.

Table 22-4 Command Data Block Structure

Byte #	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Rsv	Rsv	Rsv	Rsv	ERASE	LOCK_UNLOCK	CLR_PWD	SET_PWD
1	PWDS_LEN							
2	Password Data							
…								

PWDS_LEN + 1	

ERASE – 1 Defines Forced Erase Operation (all other bits shall be 0) and only the command byte is sent.

LOCK/UNLOCK – 1=Locks the card. 0=Unlock the card (note that it is valid to set this bit together with SET_PWD but it is not allowed to set it together with CLR_PWD).

CLR_PWD – 1=Clears PWD.

SET_PWD – 1=Set new password to PWD.

PWD_LEN – Defines the following password length (in bytes).

PWD – The password (new or currently used depending on the command).

The data block size shall be defined by the host before it send the card lock/unlock command. This will allow different password sizes.

The following paragraphs define the various lock/unlock command sequences:

Lock command sequences:

1 Setting the Password.

a Select a card (CMD7), if not previously selected already.

b Define the block length (CMD16), given by the 8bit card lock/unlock mode, the 8 bits password size (in bytes), and the number of bytes of the new password. In case that a password replacement is done, then the block size shall consider that both passwords, the old and the new one, are sent with the command.

c Send Card Lock/Unlock command with the appropriate data block size on the data line including 16-bit CRC. The data block shall indicate the mode (SET_PWD), the length (PWD_LEN) and the password itself. In case that a password replacement is done, then the length value (PWD_LEN) shall include both passwords, the old and the new one, and the PWD field shall include the old password (currently used) followed by the new password.

d In case that the sent old password is not correct (not equal in size and content) then LOCK_UNLOCK_FAILED error bit will be set in the status register and the old password does not change. In case that PWD matches the sent old password then the given new password and its size will be saved in the PWD and PWD_LEN fields, respectively.

NOTE:

the password length register (PWD_LEN) indicates if a password is currently set. When it equals 0 there is no password set. If the value of PWD_LEN is not equal to zero the card will lock itself after power up. It is possible to lock the card immediately in the current power session by setting the LOCK/UNLOCK bit (while setting the password) or sending additional command for card lock.

2 Reset the password.

a Select a card (CMD7), if not previously selected already.

b Define the block length (CMD16), given by the 8-bit card lock/unlock mode, the 8-bit password size (in bytes), and the number of bytes of the currently used password.

c Send the card lock/unlock command with the appropriate data block size on the data line including 16-bit CRC. The data block shall indicate the mode CLR_PWD, the length (PWD_LEN) and the password (PWD) itself (LOCK/UNLOCK bit is don't care). If the PWD and PWD_LEN is set to 0. If the password is not correct then the LOCK_UNLOCK_FAILED error bit will be set in the status register.

3 Locking a card.

a Select a card (CMD7), if not previously selected already.

b Define the block length (CMD16), given by the 8-bit card lock/unlock mode, the 8-bit password size (in bytes), and the number of bytes of currently used password.

c Send the card lock/unlock command with the appropriate data block size on the data line including 16-bit CRC. The data block shall indicate the mode LOCK, the length (PWD_LEN) and the password (PWD) itself.

If the PWD content equals to the sent password then the card will be locked and the card-locked status bit will be set in the status register. If the password is not correct then LOCK_UNLOCK_FAILED error bit will be set in the status register.

NOTE:

it is possible to set the password and to lock the card in the same sequence. In such case the host shall perform all the required steps for setting the password (as described above) including the bit LOCK set while the new password command is sent. If the password was previously set (PWD_LEN is not 0), then the card will be locked automatically after power on reset. An attempt to lock a locked card or to lock a card that does not have a password will fail and the LOCK_UNLOCK_FAILED error bit will be set in the status register.

Unlock command sequences:

1 Unlocking the card.

a Select a card (CMD7), if not previously selected already.

b Define the block length (CMD16), given by the 8-bit card lock/unlock mode, the 8-bit password size (in bytes), and the number of bytes of the currently used password.

c Send the card lock/unlock command with the appropriate data block size on the data line including 16-bit CRC. The data block shall indicate the mode UNLOCK, the length (PWD_LEN) and the password (PWD) itself.

If the PWD content equals to the sent password then the card will be unlocked and the card-locked status bit will be cleared in the status register. If the password is not correct then the LOCK_UNLOCK_FAILED error bit will be set in the status register.

NOTE:

the unlocking is done only for the current power session. As long as the PWD is not

cleared the card will be locked automatically on the next power up. The only way to unlock the card is by clearing the password. An attempt to unlock an unlocked card will fail and LOCK_UNLOCK_FAILED error bit will be set in the status register.

2 Forcing Erase.

In case that the user forgot the password (the PWD content) it is possible to erase all the card data content along with the PWD content. This operation is called Forced Erase.

a Select a card (CMD7), if not previously selected already.

b Define the block length (CMD16) to 1 byte (8bit card lock/unlock command). Send the card lock/unlock command with the appropriate data block of one byte on the data line including 16-bit CRC. The data block shall indicate the mode ERASE (the ERASE bit shall be the only bit set).

If the ERASE bit is not the only bit in the data field then the LCOK_UNLOCK_FAILED error bit will be set in the status register and the erase request is rejected. If the command was accepted then ALL THE CARD CONTENT WILL BE ERASED including the PWD and PWD_LEN register content and the locked card will get unlocked.

An attempt to force erase on an unlocked card will fail and LOCK_UNLOCK_FAILED error bit will be set in the status register.

22.5.7 Card Status

The response format R1 contains a 32-bit field named card status. This field is intended to transmit the card's status information (which may be stored in a local status register) to the host. If not specified otherwise, the status entries are always related to the previous issued command.

Table below defines the different entries of the status. The type and clear condition fields in the table are abbreviate as follows:

Type:

E: Error bit.

S: Status bit..

R: Detected and set for the actual command response.

X: Detected and set during command execution. The host must poll the card by issuing the status command in order to read these bits.

Clear Condition:

A: According to the card current state.

B: Always related to the previous command. Reception of a valid command will clear it (with a delay of one command).

C: Clear by read.

Table 22-5 Card Status Description

Bits	Identifier	Type	Description	Clear Condition
31	OUT_OF_RAGE	E R	The command's argument was out of the allowed range for this card. 0: No Error 1: Error	C
30	ADDRESS_ERROR	E R X	A misaligned address which did not match the block length was used in the command. 0: No Error 1: Error	C
29	BLOCK_LEN_ERROR	E R	The transferred block length is not allowed for this, or the number of transferred bytes does not match the block length. 0: No Error 1: Error	C
28	ERASE_SEQ_ERROR	E R	An error in the sequence of erase commands occurred. 0: No Error 1: Error	C
27	ERASE_PARAM	E X	An invalid selection of sectors or groups for erase occurred. 0: No Error 1: Error	C
26	WP_VIOLATION	E R X	Attempt to program a write protected block. 0: No Protected 1: Protected	C
25	CARD_IS_LOCKED	S X	When set, signals that the card is locked by the host. 0: Card unlocked 1: Card locked	A
24	LOCK_UNLOCK_FAILED	E R X	Set when a sequence or password error has been detected in lock/unlock card command or if there was an attempt to access a locked card. 0: No Error 1: Error	C

23	COM_CRC_ERROR	E R	The CRC check of the previous command failed. 0: No Error 1: Error	B
22	ILLEGAL_COMMAND	E R	Command not legal for the card state. 0: No Error 1: Error	B
21	CARD_ECC_FAILED	E X	Card internal ECC was applied but failed to correct the data. 0: normal 1: failure	C
20	CC_ERROR	E R X	Internal card controller error. 0: No Error 1: Error	C
19	ERROR	E R X	A general or an unknown error occurred during the operation. 0: No Error 1: Error	C
18	UNDERRUN	E X	The card could not sustain data transfer in stream read mode. 0: No Error 1: Error	C
17	OVERRUN	E X	The card could not sustain data programming in stream write mode. 0: No Error 1: Error	C
16	CID/CSD_OVERWRITE	E R X	Can be either one of the following errors. 0: No Error 1: Error	C
15	WP_ERASE_SKIP	S X	Only partial address space was erased due to existing write protected blocks. 0: No Protected 1 : Protected	C
14	CARD_ECC_DISABLED	S X	The command has been executed without using the internal ECC. 0: enabled 1: disabled	A

13	ERASE_RESET	S R	An erase sequence was cleared before executing because an out of erase sequence command was received. 0: normal 1: set	C
12:9	CURRENT_STATE	S X	The state of the card when receiving the command. If the command execution causes a state change, it will be visible to the host in the response to the next command. The four bits are interpreted as binary coded number between 0 and 15. 0: idle 1: ready 2: ident 3: stby 4: tran 5: data 6: rcv 7: prg 8 : dis (9 – 15) : rsv	B
8	READY_FOR_DATA	S X	Corresponds to buffer empty signaling on the bus. 0: No Ready 1: Ready	A
7:6	Reserved	-	-	-
5	APP_CMD	S R	The card will expect ACMD, or indication that the command has been interpreted as ACMD. 0: Disable 1: Enable	C
4:0	Reserved	-	-	-

22.5.8 SD Status

The SD status contains status bits that are related to the SD card proprietary features and may be used for future application specific usage. The size of the SD status is one data block of 512bit. The content of this register is transmitted to the Host over the DAT bus along with 16-bit CRC. The SD status is sent to the host over the DAT bus if ACMD13 is sent (CMD55 followed with CMD13). ACMD13 can be sent to a card only in 'tran_state' (card is selected). SD status structure is described in below.

The same abbreviation for *type* and *clear condition* were used as for the Card Status above.

Table 22-6 SD Status Structure

Bits	Identifier	Type	Description	Clear Condition
511:510	DAT_BUS_WIDTH	S R	Shows the currently defined data bus width that was defined by SET_BUS_WIDTH command. 00: 1 (default) 01: Reserved 10: 4 bit width 11: Reserved	A
509	SECURED_MODE	S R	Card is in Secured Mode of operation. 0: Not in the Mode 10: In the mode	A
508:496	Reserved.			
495:480	SD_CARD_TYPE	S R	All 0, is SD Memory cards.	A
479:448	SIZE_OF_PROTECTED_AREA	S R	Size of protected area.	A
447:312	Reserved.			
311:0	Reserved for manufacturer.			

22.5.9 SDIO

I/O access differs from memory in that the registers can be written and read individually and directly without a FAT file structure or the concept of blocks (although block access is supported). These registers allow access to the IO data, control of the IO function, and report on status or transfer I/O data to and from the host.

Each SDIO card may have from 1 to 7 functions plus one memory function built into it. A function is a self contained I/O device. I/O functions may be identical or completely different from each other. All I/O functions are organized as a collection of registers, and there is a maximum of 131,072 registers possible for each I/O function.

22.5.9.1 SDIO Interrupts

In order to allow the SDIO card to interrupt the host, and interrupt function is added to a pin on the SD interface. Pin number 8 which is used as DAT[1] when operating in the 4 bit SD mode is used to signal the card's interrupt to the host. The use of interrupt is optional for each card or function within a card. The SDIO interrupt is "level sensitive", that is, the interrupt line must be held active (low) until it is either recognized and acted upon by the host or de-asserted due to the end of the Interrupt Period. Once the host has serviced the interrupt, it is cleared via an IO write to the appropriate bit in the CCCR.

The interrupt output of all SDIO cards is active low. This host controller provides pull-up resistors on all data lines DAT[3:0].

As Pin 8 of the card is shared between the IRQ and DAT[1] use in the 4 bit SD mode, and interrupt shall only be sent by the card and recognized by the host during a specific time. The time that a low on Pin 8 will be recognized as an interrupt is defined as the Interrupt Period.

The host here will only sample the level of Pin 8 (DAT[1]/IRQ) into the interrupt detector during the Interrupt Period. At all other times, the host will ignore the level on Pin 8. Note that the Interrupt Period is applicable for both memory and IO operations. The definition of the Interrupt Period is different for operations with single block and multiple block data transfer.

22.5.9.2 SDIO Suspend/Resume

Within a multi-function SDIO or a Combo (Mix IO and Memory) card, there are multiple devices (I/O and memory) that must share access to the SD bus. In order to allow the sharing of access to the host among multiple devices, SDIO and combo cards can implement the optional concept of suspend/resume. In a card supports suspend/resume, the host may temporarily halt a data transfer operation to one function or memory (suspend) in order to free the bus for a higher priority transfer to a different function of memory. Once this higher-priority transfer is complete, the original transfer is re-started where it left off (resume). The host controller here is supported by all IO functions except zero, and the memory of a combo card, and can suspend multiple transactions and resume them in any order desired. IO function zero does not support suspend/resume.

The procedure used to perform the Suspend/Resume operation on the SD bus is:

- The host determines which function currently used the DAT[] line(s).
- The host requests the lower priority or slower transaction to suspend.
- The host checks for the transaction suspension to complete.
- The host begins the higher priority transaction.
- The host waits for the completion of the higher priority transaction.
- The host restores the suspended transaction.

22.5.9.3 SDIO Read Wait

The optional Read Wait (RW) operation is defined only for the SD 1-bit and 4-bit modes. The read wait operation allows a host to signal a card that it is doing a read multiple (CMD53) operation to temporarily stall the data transfer while allowing the host to send commands to any function within the SDIO device. To determine if a card supports the Read Wait protocol, the host must test capability bits in CCCR. The timing for Read Wait is base on the Interrupt Period.

22.5.10 Clock Control

The software should guarantee that the card identification process starts in open-drain mode with the

clock rate fod (0 ~ 400khz). In addition, the software should also make the card into interrupt mode with fod (only for MMC). The commands that require fod are CMD0, CMD1, CMD2, CMD3, CMD5, CMD40 and ACMD41. In data transfer mode, the MSC controller can operate card with clock rate fpp (0 ~ 25Mhz).

22.5.11 Application Specified Command Handling

The MultiMediaCard/SD system is designed to provide a standard interface for a variety applications types. In this environment it is anticipate that there will be a need for specific customers/applications features. To enable a common way of implementing these features, two types of generic commands are defined in the standard: Application Specific Command, ACMD, and General Command, GEN_CMD.

GEN_CMD, this command, when received by the card, will cause the card to interpret the following command as an application specific command, ACMD. The ACMD has the same structure as of regular MultiMediaCard standard commands and it may have the same CMD number. The card will recognize it as ACMD by the fact that it appears after APP_CMD.

The only effect of the APP_CMD is that if the command index of the, immediately, following command has an ACMD overloading, the none standard version will used. If, as an example, a card has a definition for ACMD13 but not for ACMD7 then, if received immediately after APP_CMD command, Command 13 will be interpreted as the non standard ACMD13 but, command 7 as the standard CMD7.

In order to use one of the manufacturer specific ACMD's the host will:

1 Send APP_CMD. The response will have the APP_CMD bit (new status bit) set signaling to the host that ACMD is now expected.
2 Send the required ACMD. The response will have the APP_CMD bit set, indicating that the accepted command was interpreted as ACMD. If a non-ACMD is sent then it will be respected by the card as normal MultiMediaCard command and the APP_CMD bit in the Card Status stays clear.

If a non valid command is sent (neither ACMD nor CMD) then it will be handled as a standard MultiMediaCard illegal command error.

The bus transaction of the GEN_CMD is the same as the single block read or write commands (CMD24 or CMD17). The difference is that the argument denotes the direction of the data transfer (rather than the address) and the data block is not a memory payload data but has a vendor specific format and meaning.

The card shall be selected ('tran_state') before sending CMD56. The data block size is the BLOCK_LEN that was defined with CMD16. The response to CMD56 will be R1b (card status + busy indication).

22.6 MMC/SD Controller Operation

22.6.1 Data FIFOs

The controller FIFOs for the response tokens, received data, and transmitted data are MSC_RES, MSC_RXFIFO, and MSC_TXFIFO, respectively. These FIFOs are accessible by the software and are described in the following paragraphs.

22.6.1.1 Response FIFO (MSC_RES)

The response FIFO, MSC_RES, contains the response received from an MMC/SD card after a command is sent from the controller. MSC_RES is a read-only, 16-bit, and 8-entry deep FIFO.

The FIFO will hold all possible response lengths. Responses that are only one byte long are located on the LSB of the 16-bit entry in the FIFO. The first half-word read from the response FIFO is the most significant half-word of the received response. For example, if the response format is R1, then the response read from RES_FIFO is bit [47:32], bit[31:16], bit[15:0] and in the third half-word only the low 8-bit is effective response [15:8] and the high 8-bit is ignored. If the response format is R2, then the response read from MSC_RES is bit [135:8] and needs reading 8 times.

The FIFO does not contain the response CRC. The status of the CRC check is in the status register, MSC_STAT.

22.6.1.2 Receive/Transmit Data FIFO (MSC_RXFIFO/MSC_TXFIFO)

The receive data FIFO and transmit data FIFO share one 16-entry x 32-bit FIFO, because at one time data are only received or are only transmitted. If it is used to receive data, it is called MSC_RXFIFO and read-only. If it is used to transmit data, it is called MSC_TXFIFO and write-only.

Data FIFO and its controls are cleared to a starting state after a system reset or at the beginning of the operations which include data transfer. (MSC_CMDAT[DATA_EN] == 1)

If at any time MSC_RXFIFO becomes full and the data transmission is not complete, the controller turns the MSC_CLK off to prevent any overflows. When the clock is off, data transmission from the card stops until the clock is turned back on. After MSC_RXFIFO is not full, the controller turns the clock on to continue data transmission. The full status of the FIFO is registered in the MSC_STAT [DATA_FIFO_FULL] bit.
If at any time MSC_TXFIFO becomes empty and the data transmission is not complete, the controller turns the MSC_CLK off to prevent any underrun. When the clock is off, data transmission to the card stops until the clock is turned back on. When MSC_TXFIFO is no longer empty, the controller automatically restarts the clock. The empty status of the FIFO is registered in the MSC_STAT [DATA_FIFO_EMPTY] bit.

The FIFO is readable on word (32-bit) boundaries. The max read/written number is 16 words. The

controller can correctly process big-endian and little-endian data.

Because at the beginning of the operation which include data transfer (MSC_CMDAT [DATA_EN] == 1), Data FIFO and its controls are cleared, software should guarantee data in FIFO have been read/written before beginning a new command.

22.6.2 DMA and Program I/O

Software may communicate to the MMC controller via the DMA or program I/O.

To access MSC_RXFIFO/MSC_TXFIFO with the DMA, the software must program the DMA to read or write the FIFO with source port width 32-bit, destination port width 32-bit, transfer data size 32-byte, transfer mode single. For example, to write 64 bytes of data to the MSC_TXFIFO, the software must program the DMA as follows:

```
DMA_DCTRn = 2                // Write 2 32-bytes (64 bytes)
DMA_DCCRn[SWDH] = 0          // source port width is 32-bit
DMA_DCCRn[DWDH] = 0          // destination port width is 32-bit
DMA_DCCRn[DS] = 4            // transfer data size is 32-byte
DMA_DCCRn[TM] = 4            // transfer mode is single
DMA_DCCRn[RDIL] = 0          // request detection interval length is 0
```

The number of 32-bytes should be calculated from the number of transferred bytes as follows:

The number of words = (The number of bytes + 31) / 32

If the number of transferred bytes is not the multiple of 4, the controller can correctly process endian.

The DMA trigger level is 8 words, that is to say, the DMA read trigger is when data words in MSC_RXFIFO is >= 8 and the DMA write trigger is when data words in MSC_TXFIFO is < 8. Software can also configure DMA registers based on requirements, but the above 32-byte transfer data size is most efficient.

With program I/O, the software waits for the MSC_IREG [RXFIFO_RD_REQ] or MSC_IREG [TXFIFO_WR_REQ] interrupts before reading or writing the respective FIFO.

NOTES:

1. The MSC_CMDAT [DMA_EN] bit must be set to a 1 to enable communication with the DMA and it must be set to a 0 to enable program I/O.
2. DMA can be enabled only after MSC_CMDAT is written, because MSC_CMDAT [DATA_EN] is used to reset TX/RXFIFO.

22.6.3 Start and Stop clock

The software stops the clock as follows:

1 Write MSC_STRPCL with 0x01 to stop the MMC/SD bus clock.
2 Wait until MSC_STAT[CLK_EN] becomes zero.

To start the clock the software writes MSC_STRPCL with 0x02.

22.6.4 Software Reset

Reset includes the MSC reset and the card reset.

The MSC reset is through MSC_STRPCL [RESET] bit.

The card reset is to make the card into idle state. CMD0 (GO_IDLE_STATE) sets the MMC and SD memory cards into idle state. CMD52 (IO_RW_DIRECT, with argument 0x88000C08) reset the SD I/O card. The MMC/SD card are initialized with a default relative card address (RCA = 0x0001 for MMC and RCA = 0x0000 for SD) and with a default driver stage register setting (lowest speed, highest driving current capability).

The following registers must be set before the clock is started:

Step 1. Stop the clock.
Step 2. Set MSC_STRPCL register to 0x08 to reset MSC.
Step 3. Wait while MSC_STAT [IS_RESETTING] is 1.
Step 4. Set MSC_CMD with CMD0.
Step 5. Update the MSC_CMDAT register as follows:
  a Write 0x0000 to MSC_CMDAT [RESPONSE_FORMAT].
  b Clear the MSC_CMDAT [DATA_EN] bit.
  c Clear the MSC_CMDAT [BUSY] bit.
  d Clear the MSC_CMDAT [INIT] bit.
Step 6. Start the clock.
Step 7. Start the operation. (write MSC_STRPCL with 0x04)
Step 8. Wait for the END_CMD_RES interrupt.
Step 9. Set MSC_CMD with CMD52.
Step 10. Set MSC_ARG with 0x88000C08.
Step 11. Update the MSC_CMDAT register as follows:
  a Write 0x005 to MSC_CMDAT [RESPONSE_FORMAT].
  b Clear the MSC_CMDAT [DATA_EN] bit.
  c Clear the MSC_CMDAT [BUSY] bit.
  d Clear the MSC_CMDAT [INIT] bit.
Step 12. Start the operation.
Step 13. Wait for the END_CMD_RES interrupt.

22.6.5 Voltage Validation and Card Registry

At most 10 MMC and 1 SD (either SDMEM or SDIO) can be inserted MMC/SD bus at the same time, and their voltage validation and card registry steps are different, so the software should be

programmed as follows:

Step 1. Check whether SDIO card is inserted.

Step 2. Check whether SDMEM card is inserted.

Step 3. Check whether MMC cards are inserted.

22.6.5.1 Check SDIO

The commands are sent as follows:

Step 1. (Optional) Send CMD52 (IO_RW_DIRECT) with argument 0x88000C08 to reset SDIO card.

Step 2. Send CMD5 (IO_SEND_OP_CMD) to validate voltage.

Step 3. If the response is correct and the number of IO functions > 0, then continue, else go to check SDMEM.

Step 4. If C-bit in the response is ready (the initialization has finished), go to 6.

Step 5. Send CMD5 (IO_SEND_OP_CMD) to validate voltage, then go to 4.

Step 6. If memory-present-bit in the response is true, then it is a combo card (SDIO + Memory), else it is only a SDIO card.

Step 7. If it is a combo card, go to check SDMEM to initialize the memory part.

Step 8. Send CMD3 (SET_RELATIVE_ADDR) to let the card publish a RCA. The RCA is returned from the response.

Step 9. If do not accept the new RCA, go to 8, else record the new RCA.

Step 10. Go to check MMC, because we can assure that there is no SDMEM card.

22.6.5.2 Check SDMEM

If there is no SDIO card or there is a combo card, continue to check SDMEM.

The commands are sent as follows:

Step 1. (Optional) Send CMD0 (GO_IDLE_STATE) to reset MMC and SDMEM card. This command has no response.

Step 2. Send CMD55. Here the default RCA 0x0000 is used for CMD55.

Step 3. If the response is correct (CMD55 has response), then continue, else go to check MMC.

Step 4. Send ACMD41 (SD_SEND_OP_CMD) to validate voltage (the general OCR value is 0x00FF8000).

Step 5. If the initialization has finished, go to 7. (The response is the OCR register and it includes a status information bit (bit [31]). This status bit is set if the card power up procedure has been finished. As long as the card is busy, the corresponding bit[31] is set to LOW.)

Step 6. Send CMD55 and ACMD41 to validate voltage, and then go to 5.

Step 7. Send CMD2 (ALL_SEND_CID) to get the card CID.

Step 8. Send CMD3 (SET_RELATIVE_ADDR) to let card publish a RCA. The RCA is returned from the response.

Step 9. If do not accept the new RCA, go to 8, else record the new RCA.

Step 10. Go to check MMC.

22.6.5.3 Check MMC

Because there may be several MMC card, so some steps (5 ~ 8) should be repeated several times.

The commands are sent as follows:

Step 1. Send CMD1 (SEND_OP_CMD) to validate voltage (the general OCR value is 0x00FF88000).
Step 2. If the response is correct, then continue, else goto 9.
Step 3. If the initialization has finished, go to 5. (The response is the OCR register and it includes a status information bit (bit [31]). This status bit is set if the card power up procedure has been finished. As long as the card is busy, the corresponding bit[31] is set to LOW.)
Step 4. Send CMD1 (SEND_OP_CMD) to validate voltage, and then go to 3.
Step 5. Send CMD2 (ALL_SEND_CID) to get the card CID.
Step 6. If the response timeout occurs, goto 9.
Step 7. Send CMD3 (SET_RELATIVE_ADDR) to assign the card a RCA.
Step 8. If there are other MMC cards, then go to 5.
Step 9. Finish.

22.6.6 Single Data Block Write

In a single block write command, the following registers must be set before the operation is started:

Step 1. Set MSC_NOB register to 0x0001.
Step 2. Set MSC_BLKLEN to the number of bytes per block.
Step 3. Update the MSC_CMDAT register as follows:
  a Write 0x001 to MSC_CMDAT [RESPONSE_FORMAT].
  b Write 0x2 to MSC_CMDAT [BUS_WIDTH] if the card is SD, else clear it.
  c Set the MSC_CMDAT [DATA_EN] bit.
  d Set the MSC_CMDAT [WRITE_READ] bit.
  e Clear the MSC_CMDAT [STREAM_BLOCK] bit.
  f Clear the MSC_CMDAT [BUSY] bit.
  g Clear the MSC_CMDAT [INIT] bit.
Step 4. Start the operation.
Step 5. Write MSC_IMASK with some value to unmask the expected interrupts.

Then the software must perform the following steps:

Step 1. Wait for the MSC_IREG [END_CMD_RES] interrupt.
Step 2. Wait for the MSC_IREG [DATA_TRAN_DONE] interrupt.
At the same time write data to the MSC_TXFIFO and continue until all of the data have been written to the FIFO.
Step 3. Wait for MSC_IREG [PROG_DONE] interrupt. This interrupt indicates that the card has finished programming. Certainly software may start another command sequence on a different card.
Step 4. Read the MSC_STAT register to verify the status of the transaction (i.e. CRC error status).

To address a different card, the software sends a select command to that card by sending a basic no data command and response transaction. To address the same card, the software must wait for MSC_IREG [PROG_DONE] interrupt. This ensures that the card is not in the busy state.

In addition, CMD26 (PROGRAM_CID), CMD27 (PROGRAM_CSD), CMD42 (LOCK/UNLOCK), CMD56 (GEN_CMD: write) and CMD53 (single_block_write) operations are similar to single block write.

22.6.7 Single Block Read

In a single block read command, the following registers must be set before the operation is started:

Step 1. Set MSC_NOB register to 0x0001.
Step 2. Set MSC_BLKLEN register to the number of bytes per block.
Step 3. Update the following bits in the MSC_CMDAT register:
  a Write 0x001 to MSC_CMDAT [RESPONSE_FORMAT].
  b Write 0x2 to MSC_CMDAT [BUS_WIDTH] if the card is SD, else clear it.
  c Set the MSC_CMDAT [DATA_EN] bit.
  d Clear the MSC_CMDAT [WRITE_READ] bit.
  e Clear the MSC_CMDAT [STREAM_BLOCK] bit.
  f Clear the MSC_CMDAT [BUSY] bit.
  g Clear the MSC_CMDAT [INIT] bit.
Step 4. Start the operation.
Step 5. Write MSC_IMASK with some value to unmask the expected interrupts.

Then the software must perform the following steps:

Step 1. Wait for the MSC_IREG [END_CMD_RES] interrupt.
Step 2. Wait for the MSC_IREG [DATA_TRAN_DONE] interrupt.
  At the same time read data from the MSC_RXFIFO as data becomes available in the FIFO, and continue reading until all data is read from the FIFO.
Step 3. Read the MSC_STAT register to verify the status of the transaction (i.e. CRC error status).

In addition, CMD30 (SEND_WRITE_PROT), ACMD13 (SD_STATUS), CMD56 (GEN_CMD-read), ACMD51 (SEND_SCR) and CMD53 (single_block_read) are similar to single block read.

22.6.8 Multiple Block Write

The multiple block write mode is similar to the single block write mode, except that multiple blocks of data are transferred. Each block is the same length. All the registers are set as they are for the single block write, except that the MSC_NOB register is set to the number of blocks to be written.

The multiple block write mode also requires a stop transmission command, CMD12, after the data is transferred to the card. After the MSC_IREG [DATA_TRAN_DONE] interrupt occurs, the software must

program the controller register to send a stop data transmission command.

If multiple block write with pre-defined block count (refer to MMC spec v-3.3) is used, CMD12 should not be sent.

For SDIO card, CMD53 (multiple_block_write) is also similar, but when IO abort (CMD52) is sent, MSC_CMDAT [IO_ABORT] should be 1.

Table 22-7 How to stop multiple block write

Operation	Stop condition	Software processing
Open-ended or SDIO infinite	After write MSC_NOB blocks	1 Wait for DATA_TARN_DONE interrupt. 2 Send CMD12 or CMD52. (IO abort) 3 Wait for END_CMD_RES and PRG_DONE interrupt.
Open-ended or SDIO infinite	Stop writing in advance (not write MSC_NOB blocks)	1 Set MSC_STRPCL [EXIT_MULTIPLE]. 2 Wait for DATA_TRAN_DONE interrupt. 3 Send CMD12 or CMD52. (IO abort) 4 Wait for END_CMD_RES and PRG_DONE interrupt.
Predefined block or SDIO finite	After writing MSC_NOB blocks	1 Wait for DATA_TRAN_DONE interrupt.
Predefined block or SDIO finite	Stop writing in advance (not write MSC_NOB blocks)	1 Set MSC_STRPCL [EXIT_MULTIPLE]. 2 Wait for DATA_TRAN_DONE interrupt. 3 Send CMD12 or CMD52. (IO abort) 4 Wait for END_CMD_RES and PRG_DONE interrupt.

22.6.9 Multiple Block Read

The multiple blocks read mode is similar to the single block read mode, except that multiple blocks of data are transferred. Each block is the same length. All the registers are set as they are for the single block read, except that the MSC_NOB register is set to the number of blocks to be read.

The multiple blocks read mode requires a stop transmission command, CMD12, after the data from the card is received. After the MSC_IREG [DATA_TRAN_DONE] interrupt has occurred, the software must program the controller registers to send a stop data transmission command.

If multiple block read with pre-defined block count (refer to MMC spec v-3.3) is used, CMD12 should not be sent.

For SDIO card, CMD53 (multiple_block_read) is also similar, but when IO abort (CMD52) is sent, MSC_CMDAT [IO_ABORT] should be 1.

Table 22-8 How to stop multiple block read

Operation	Stop condition	Software processing
Open-ended or SDIO infinite	After reading MSC_NOB blocks	1 Wait for DATA_TRAN_DONE interrupt. 2 Send CMD12 or CMD52. (IO abort) 3 Wait for END_CMD_RES interrupt.
Open-ended or SDIO infinite	Stop reading in advance (not write MSC_NOB blocks)	1 Set MSC_STRPCL [EXIT_MULTIPLE]. 2 Wait for DATA_TRAN_DONE interrupt. 3 Send CMD12 or CMD52. (IO abort) 4 Wait for END_CMD_RES interrupt.
Predefined block or SDIO finite	After reading MSC_NOB blocks	1 Wait for DATA_TRAN_DONE interrupt.
Predefined block or SDIO finite	Stop reading in advance (not write MSC_NOB blocks)	1 Set MSC_STRPCL [EXIT_MULTIPLE]. 2 Wait for DATA_TRAN_DONE interrupt. 3 Send CMD12 or CMD52. (IO abort) 4 Wait for END_CMD_RES interrupt.

22.6.10 Stream Write (MMC)

In a stream write command, the following registers must be set before the operation is started:

1 Update MSC_CMDAT register as follows:
   a Write 0x001 to the MSC_CMDAT [RESPONSE_FORMAT].
   b Clear the MSC_CMDAT [BUS_WIDTH] because only MMC support stream write.
   c Set the MSC_CMDAT [DATA_EN] bit.
   d Set the MSC_CMDAT [WRITE_READ] bit.
   e Set the MSC_CMDAT [STREAM_BLOCK] bit.
   f Clear the MSC_CMDAT [BUSY] bit.
   g Clear the MSC_CMDAT [INIT] bit.
2 Start the operation.
3 Write MSC_IMASK with some value to unmask the expected interrupts.

Then the software must perform the following steps:

1 Wait for the MSC_IREG [END_CMD_RES] interrupt.
2 Write data to the MSC_TXFIFO and continue until all of the data is written to the Data FIFO.
3 Stop clock. Wait until MSC_STAT[CLK_EN] becomes 0. The clock must be stopped.
4 Set the command registers for a stop transaction command (CMD12) and other registers.
5 Start the clock and start the operation.
6 Wait for the MSC_IREG [END_CMD_ERS] interrupt.
7 Wait for the MSC_IREG [DATA_TRAN_DONE] interrupt.
8 Wait for the MSC_IREG [PRG_DONE] interrupt. This interrupt indicates that the card has finished programming. Certainly software may start another command sequence on a different card.

9 Read the MSC_STAT register to verify the status of the transaction.

To address a different card, the software must send a select command to that card by sending a basic no data command and response transaction. To address the same card, the software must wait for MSC_IREG [PRG_DONE] interrupt. This ensures that the card is not in the busy state.

If partial blocks are allowed (if CSD parameter WRITE_BL_PARTIAL is set) the data stream can start and stop at any address within the card address space, otherwise it shall start and stop only at block boundaries. If WRITE_BL_PARTIAL is not set, 16 more stuff bytes need to be written after the useful written data, otherwise only write the useful written data.

22.6.11 Stream Read (MMC)

In a stream read command, the following registers must be set before the operation is turned on:

1 Update the MSC_CMDAT register as follows:
   a Write 0x01 to the MSC_CMDAT [RESPONSE_FORMAT].
   b Clear the MSC_CMDAT [BUS_WIDTH] because only MMC support stream read.
   c Clear the MSC_CMDAT [WRITE_READ] bit.
   d Set the MSC_CMDAT [STREAM_BLOCK] bit.
   e Clear the MSC_CMDAT [BUSY] bit.
   f Clear the MSC_CMDAT [INIT] bit.
2 Start the operation.
3 Write MSC_IMASK with some value to unmask the expected interrupts.

Then the software must perform the following steps:

1 Wait for the MSC_IREG [END_CMD_RES] interrupt.
2 Read data from the MSC_RXFIFO and continue until all of the expected data has been read from the FIFO.
3 Write MSC_STRPCL [EXIT_TRANSER] with 1. If MSC_STAT[DATA_FIFO_FULL] is 1, then read MSC_RXFIFO to make it not full. Because if data FIFO is full, MSC_CLK is stopped. Here, the data FIFO contains useless data.
4 Set the command registers for a stop transaction command (CMD12) and send it. There is no need to stop the clock.
5 Wait for the MSC_IREG [END_CMD_RES].
6 Wait for the MSC_IREG [DATA_TRAN_DONE] interrupt.
7 Read the MSC_STAT register to verify the status of the transaction.

22.6.12 Erase, Select/Deselect and Stop

For CMD7 (SELECT/DESELECT_CARD), CMD12 (STOP_TRANSMISSION) and CMD38 (ERASE), the following registers must be set before the operation is started:

1 Update the MSC_CMDAT register as follows:
   a Write 0x01 to the MSC_CMDAT [RESPONSE_FORMAT].
   b Clear the MSC_CMDAT [DATA_EN] bit.

c Clear the MSC_CMDAT [WRITE_READ] bit.
d Clear the MSC_CMDAT [STREAM_BLOCK] bit.
e Set the MSC_CMDAT [BUSY] bit.
f Clear the MSC_CMDAT [INIT] bit.

2 Start the operation.
3 Write MSC_IMASK with some value to unmask the expected interrupts.

Then the software must perform the following steps:

1 Wait for the MSC_IREG [END_CMD_RES] interrupt.
2 Wait for the MSC_IREG [PRG_DONE] interrupt. If CMD12 is sent to terminate data read operation, then there is no need to wait for MSC_IREG [PRG_DONE] interrupt. This interrupt indicates that the card has finished programming. Certainly software may start another command sequence on a different card.

22.6.13 SDIO Suspend/Resume

The actual suspend/resume steps are as follows:

1 During data transfer, send CMD52 to require suspend. BR and RAW flag should be 1.
2 If BS flag in the response is 0, then suspend has been accepted and goto 4.
3 Send CMD52 to query card status. R flag should be 1. Go to 2.
4 Write MSC_STRPCL [EXIT_TRANSFER] with 1.
5 Wait for the MSC_IREG [DATA_TRAN_DONE] interrupt.
6 Read MSC_NOB, MSC_SNOB and etc, save them into variables.
7 Set registers for high priority transfer and start it.
8 Wait until high priority transfer is finished.
9 Restore registers from variables, but MSC_NOB should be (MSC_NOB – MSC_SNOB).
10 Send CMD52 to require resume. FSx should be resumed function number.

22.6.14 SDIO ReadWait

The actual ReadWait steps are as follows:

1 During multiple block read, read MSC_SNOB. If MSC_SNOB is nearby or equal to MSC_NOB, no need to use ReadWait.
2 Write MSC_STRPCL [START_READWAIT] with 1.
3 Wait until MSC_STAT [IS_READWAIT] becomes 1.
4 Send CMD52 to query card status.
5 Write MSC_STRPCL [STOP_READWAIT] with 1.

22.6.15 Operation and Interrupt

The software can use polling-status method to operate the MMC/SD card, but this is not the proposed method, because its performance is very low. The proposed method is to use interrupt. Generally there are fixed necessary steps to finish each command. The steps are as follows:

1 (Optional) Stop clock. Poll CLK_EN.
2 Fill the registers (MSC_CMD, MSC_CMDAT, MSC_ARG, MSC_CLKRT, and etc).
3 (Optional) Start clock.
4 Start the operation. Wait for the MSC_IREG [END_CMD_RES] interrupt.
5 Wait for the MSC_IREG [DATA_TRAN_DONE] interrupt.
6 Send STOP_TRANS (CMD12) or I/O abort (CMD52). Wait for the MSC_IREG [END_CMD_ERS] interrupt.
7 Wait for the MSC_IREG [DATA_TRAN_DONE] interrupt.
8 Wait for the MSC_IREG [PRG_DONE] interrupt.

Table 22-9 The mapping between Commands and Steps

Index	Abbreviation	1	2	3	4	5	6	7	8	Comments
CMD0	GO_IDLE_STATE	Y	Y	Y	Y					
CMD1	SEND_OP_COND	Y	Y	Y	Y					
CMD2	ALL_SEND_CID	Y	Y	Y	Y					
CMD3	SET_RELATIVE_ADDR	Y	Y	Y	Y					
CMD4	SET_DSR	Y	Y	Y	Y					
CMD7	SELECT/DSELECT_CARD	Y	Y	Y	Y				Y	
CMD9	SEND_CID	Y	Y	Y	Y					
CMD10	SEND_CSD	Y	Y	Y	Y					
CMD11	READ_DAT_UNTIL_STOP	Y	Y	Y	Y		Y	Y		
CMD12	STOP_TRANSMISSION	Y	Y	Y	Y				Y	
CMD13	SEND_STATUS	Y	Y	Y	Y					
CMD15	GO_INACTIVE_STATE	Y	Y	Y	Y					
CMD16	SET_BLOCKLEN	Y	Y	Y	Y					
CMD17	READ_SINGLE_BLOCK	Y	Y	Y	Y	Y				
CMD18	READ_MULTIPLE_BLOCK	Y	Y	Y	Y	Y	Y			Open-ended
CMD18	READ_MULTIPLE_BLOCK	Y	Y	Y	Y	Y				Predefine blocks
CMD20	WRITE_DAT_UNTIL_STOP	Y	Y	Y	Y		Y	Y	Y	
CMD23	SET_BLOCK_COUNT	Y	Y	Y	Y					
CMD24	WRITE_SINGLE_BLOCK	Y	Y	Y	Y	Y			Y	
CMD25	WRITE_MULTIPLE_BLOCK	Y	Y	Y	Y	Y	Y		Y	Open-ended
CMD25	WRITE_MULTIPLE_BLOCK	Y	Y	Y	Y	Y			Y	Predefine blocks
CMD26	PROGRAM_CID	Y	Y	Y	Y	Y			Y	
CMD27	PROGRAM_CSD	Y	Y	Y	Y	Y			Y	
CMD28	SET_WRITE_PROT	Y	Y	Y	Y				Y	
CMD29	CLR_WRITE_PROT	Y	Y	Y	Y				Y	
CMD30	SEND_WRITE_PROT	Y	Y	Y	Y	Y				
CMD32	ERASE_WR_BLOCK_START	Y	Y	Y	Y					
CMD33	ERASE_WR_BLOCK_END	Y	Y	Y	Y					
CMD35	ERASE_GROUP_START	Y	Y	Y	Y					

CMD36	ERASE_GROUP_END	Y	Y	Y	Y					
CMD38	ERASE	Y	Y	Y	Y				Y	
CMD39	FAST_IO	Y	Y	Y	Y					
CMD40	GO_IRQ_STATE	Y	Y	Y	Y					
CMD42	LOCK/UNLOCK	Y	Y	Y	Y	Y			Y	
CMD55	APP_CMD	Y	Y	Y	Y					
CMD56	GEN_CMD	Y	Y	Y	Y	Y				Read
CMD56	GEN_CMD	Y	Y	Y	Y	Y			Y	Write
ACMD6	SET_BUS_WIDTH	Y	Y	Y	Y					
ACMD13	SD_STATUS	Y	Y	Y	Y	Y				
ACMD22	SEND_NUM_WR_BLOCKS	Y	Y	Y	Y					
ACMD23	SET_WR_BLOCK_COUNT	Y	Y	Y	Y					
ACMD41	SD_SEND_OP_COND	Y	Y	Y	Y					
ACMD42	SET_CLR_CARD_DETECT	Y	Y	Y	Y					
ACMD51	SEND_SCR	Y	Y	Y	Y	Y				

NOTES:

1 For stream read/write, STOP_CMD is sent after finishing data transfer. For write, STOP_CMD is with the last six bytes. For read, STOP_CMD is sent after receiving data and card sends some data which MSC ignores.

23 UART Interface

23.1 Overview

This chapter describes the universal asynchronous receiver/transmitter (UART) serial ports. There are three UARTs: All UARTs use the same programming model. Each of the serial ports can operate in interrupt based mode or DMA-based mode.

The Universal asynchronous receiver/transmitter (UART) is compatible with the 16550-industry standard and can be used as slow infrared asynchronous interface that conforms to the Infrared Data Association (IrDA) serial infrared specification 1.1.

23.1.1 Features

- Full-duplex operation
- 5-, 6-, 7- or 8-bit characters with optional no parity or even or odd parity and with 1, 1½, or 2 stop bits
- 32x8 bit transmit FIFO and 32x11bit receive FIFO
- Independently controlled transmit, receive (data ready or timeout), line status interrupts
- Internal diagnostic capability Loopback control and break, parity, overrun and framing-error is provided
- Separate DMA requests for transmit and receive data services in FIFO mode
- Supports modem flow control by software or hardware
- Slow infrared asynchronous interface that conforms to IrDA specification

23.1.2 Pin Description

Table 23-1 UART Pins Description

Name	Type	Description
RxD	Input	Receive data input
TxD	Output	Transmit data output
CTS_	Input	Clear to Send — Modem Transmission enabled
RTS_	Output	Request to Send — UART Transmission request

NOTES:

1 UART2, UART0 support RxD, TxD, RTS_, CTS_, UART1 supports only RxD, TxD.

23.2 Register Descriptions

All UART register 32-bit access address is physical address. When ULCR.DLAB is 0, URBR, UTHR and UIER can be accessed; When ULCR.DLAB is 1, UDLLR and UDLHR can be accessed.

Table 23-2 UART Registers Description

Name	Description	RW	Reset Value	Address	Access Size
URBR0	UART Receive Buffer Register 0	R	0x??	0x10030000	8
UTHR0	UART Transmit Hold Register 0	W	0x??	0x10030000	8
UDLLR0	UART Divisor Latch Low Register 0	RW	0x00	0x10030000	8
UDLHR0	UART Divisor Latch High Register 0	RW	0x00	0x10030004	8
UIER0	UART Interrupt Enable Register 0	RW	0x00	0x10030004	8
UIIR0	UART Interrupt Identification Register 0	R	0x01	0x10030008	8
UFCR0	UART FIFO Control Register 0	W	0x00	0x10030008	8
ULCR0	UART Line Control Register 0	RW	0x00	0x1003000C	8
UMCR0	UART Modem Control Register 0	RW	0x00	0x10030010	8
ULSR0	UART Line Status Register 0	R	0x00	0x10030014	8
UMSR0	UART Modem Status Register 0	R	0x00	0x10030018	8
USPR0	UART Scratchpad Register 0	RW	0x00	0x1003001C	8
ISR0	Infrared Selection Register 0	RW	0x00	0x10030020	8
UMR0	UART M Register 0	RW	0x00	0x10030024	8
UACR0	UART Add Cycle Register 0	RW	0x00	0x10030028	16
URBR1	UART Receive Buffer Register 1	R	0x??	0x10031000	8
UTHR1	UART Transmit Hold Register 1	W	0x??	0x10031000	8
UDLLR1	UART Divisor Latch Low Register 1	RW	0x00	0x10031000	8
UDLHR1	UART Divisor Latch High Register 1	RW	0x00	0x10031004	8
UIER1	UART Interrupt Enable Register 1	RW	0x00	0x10031004	8
UIIR1	UART Interrupt Identification Register 1	R	0x01	0x10031008	8
UFCR1	UART FIFO Control Register 1	W	0x00	0x10031008	8
ULCR1	UART Line Control Register 1	RW	0x00	0x1003100C	8
UMCR1	UART Modem Control Register 1	RW	0x00	0x10031010	8
ULSR1	UART Line Status Register 1	R	0x00	0x10031014	8
UMSR1	UART Modem Status Register 1	R	0x00	0x10031018	8
USPR1	UART Scratchpad Register 1	RW	0x00	0x1003101C	8
ISR1	Infrared Selection Register 1	RW	0x00	0x10031020	8
UMR1	UART M Register 1	RW	0x00	0x10031024	8
UACR1	UART Add Cycle Register 1	RW	0x00	0x10031028	16
URBR2	UART Receive Buffer Register 2	R	0x??	0x10032000	8
UTHR2	UART Transmit Hold Register 2	W	0x??	0x10032000	8
UDLLR2	UART Divisor Latch Low Register 2	RW	0x00	0x10032000	8
UDLHR2	UART Divisor Latch High Register 2	RW	0x00	0x10032004	8

UIER2	UART Interrupt Enable Register 2	RW	0x00	0x10032004	8
UIIR2	UART Interrupt Identification Register 2	R	0x01	0x10032008	8
UFCR2	UART FIFO Control Register 2	W	0x00	0x10032008	8
ULCR2	UART Line Control Register 2	RW	0x00	0x1003200C	8
UMCR2	UART Modem Control Register 2	RW	0x00	0x10032010	8
ULSR2	UART Line Status Register 2	R	0x00	0x10032014	8
UMSR2	UART Modem Status Register 2	R	0x00	0x10032018	8
USPR2	UART Scratchpad Register 2	RW	0x00	0x1003201C	8
ISR2	Infrared Selection Register 2	RW	0x00	0x10032020	8
UMR2	UART M Register 2	RW	0x00	0x10032024	8
UACR2	UART Add Cycle Register 2	RW	0x00	0x10032028	16

23.2.1 UART Receive Buffer Register (URBR)

The read-only URBR is corresponded to one level 11bit buffer in non-FIFO mode and a 32x11bit FIFO that holds the character(s) received by the UART. Bits in URBR are right justified when being configured to use fewer than eight bits, and the rest of most significant data bits are zeroed and the most significant three bits of each buffer are the status for the character in the buffer. If ULSR.DRY is 0, don't read URBR, otherwise wrong operation may occur.

URBR0, URBR1, URBR2 — **0x10030000, 0x10031000, 0x10032000 (DLAB = 0)**

Bit	7	6	5	4	3	2	1	0
	URBR							
RST	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
7:0	URBR	8-bit UART receive read data.	R

23.2.2 UART Transmit Hold Register (UTHR)

The write-only UTHR is corresponded to one leve 8 bit buffer in non-FIFO mode and a 32x8bit FIFO in FIFO mode that holds the data byte(s) to be transmitted next.

UTHR0, UTHR1, UTHR2								0x10030000, 0x10031000, 0x10032000 (DLAB = 0)
Bit	7	6	5	4	3	2	1	0
	UTHR							
RST	?	?	?	?	?	?	?	?

Bits	Name	Description	RW
7:0	UTHR	8-bit UART transmit write hold data.	W

23.2.3 UART Divisor Latch Low/High Register (UDLLR / UDLHR)

UART Divisor Latch registers, UDLLR/UDLHR together compose the divisor for the programmable baud rate generator that can take the UART device clock and divide it by 1 to (2^{16} – 1).

The UART device source clock is EXCLK or EXCLK/2 that is determined by CPCCR.ECS. UDLHR/UDLLR stores the high/low 8-bit of the divisor respectively. Load these divisor latches during initialization to ensure that the baud rate generator operates properly. If both Divisor Latch registers are 0, the 16X clock stops.

If you don't set UMR and UACR, UART will work at normal mode with the specified frequency. The relationship between baud rate and the value of Divisor is shown by the formula when UMR and UACR are not set:

Baud Rate = (UART device clock) / (16 * **Divisor**)

UDLLR0, UDLLR1, UDLLR2								0x10030000, 0x10031000, 0x10032000 (DLAB = 1)
Bit	7	6	5	4	3	2	1	0
	Divisor Latch Low 8-bit							
RST	0	0	0	0	0	0	0	0

UDLHR0, UDLHR1, UDLHR2									0x10030004, 0x10031004, 0x10032004 (DLAB = 1)
Bit	7	6	5	4	3	2	1	0	
	Divisor Latch High 8-bit								
RST	0	0	0	0	0	0	0	0	

23.2.4 UART Interrupt Enable Register (UIER)

The UART Interrupt Enable Register (UIER) contains the interrupt enable bits for the five types of interrupts (receive data ready, timeout, line status, and transmit data request, and modem status) that set a value in UIIR.

UIER0, UIER1, UIER2									0x10030004, 0x100301004, 0x10032004 (DLAB = 0)
Bit	7	6	5	4	3	2	1	0	
	Reserved			RTOIE	MSIE	RLSIE	TDRIE	RDRIE	
RST	0	0	0	0	0	0	0	0	

Bits	Name	Description	RW
7:5	Reserved	Always read 0, write is ignored.	R
4	RTOIE	**Receive Timeout Interrupt Enable.** 0: Disable the receive timeout interrupt 1: Enable the receive timeout interrupt Timeout means the URDR (FIFO mode) is not empty but no character has received for a period of time T: T (bits) = 4 X Word length + 12.	RW
3	MSIE	**Modem Status Interrupt Enable.** 0: Disable the modem status interrupt 1: Enable the modem status interrupt	RW
2	RLSIE	**Receive Line Status Interrupt Enable.** 0: Disable receive line status interrupt 1: Enable receive line status interrupt	RW
1	TDRIE	**Transmit Data Request Interrupt Enable.** 0: Disable the transmit data request interrupt 1: Enable the transmit data request interrupt	RW
0	RDRIE	**Receive Data Ready Interrupt Enable.** 0: Disable the receive data ready interrupt 1: Enable the receive data ready interrupt	RW

23.2.5 UART Interrupt Identification Register (UIIR)

The read-only UART Interrupt Identification Register (UIIR) records the prioritized pending interrupt source information. Its initial value after power-on reset is 0x01.

UIIR0, UIIR1, UIIR2 **0x10030008, 0x10031008, 0x10032008**

Bit	7	6	5	4	3	2	1	0
	FFMSEL		Reserved		INID			INPEND
RST	0	0	0	0	0	0	0	1

Bits	Name	Description	RW
7:6	FFMSEL	**FIFO Mode Select.** 0b00: Non-FIFO mode 0b01: Reserved 0b10: Reserved 0b11: FIFO mode	R
5:4	Reserved	Always read 0, write is ignored.	R
3:1	INID	**Interrupt Identifier.** These bits identify the current highest priority pending interrupt. INID: Description 0b000: Modem Status 0b001: Transmit Data Request 0b010: Receive Data Ready 0b011: Receive Line Status 0b100: Reserved 0b101: Reserved 0b110: Receive Time Out 0b111: Reserved See Table 23-3 for details.	R
0	INPEND	**Interrupt Pending.** 0: interrupt is pending 1: No interrupt pending	R

Table 23-3 UART Interrupt Identification Register Description

UIIR.INID	Interrupt Set/Clear Cause			
	Priority	Type	Source	Clear Condition
0b0001	—	None	No pending interrupt	—
0b0110	1st Highest	Receive Line Status	Overrun, Parity, Frame Error, Break Interrupt, and FIFO Error (DMA mode only)	Reading ULSR or empty all the error characters in DMA mode
0b0100	2nd Highest	Receive Data Ready	FIFO mode: Trigger threshold was reached Non-FIFO mode: URBR full	FIFO mode: Reading URBR till below trigger threshold. Non-FIFO mode: Empty URBR
0b1100	2nd Highest	Receive Timeout	FIFO mode only: URBR not empty but no data read in for a period of time	Reset receive buffer by setting UFCR.RFRT to 1 or Reading URBR
0b0010	3rd Highest	Transmit Data Request	FIFO mode: Empty location in UTHR equal to half or more than half Non-FIFO mode: UTHR empty	FIFO mode: Data number in UTHR more than half Non-FIFO mode: Writing UTHR
0b0000	4th Highest	Modem Status	Modem CTS_ pin status change	Reading UMSR

23.2.6 UART FIFO Control Register (UFCR)

The write-only register UFCR contains the control bits for receive and transmit FIFO.

UFCR0, UFCR1, UFCR2 **0x10030008, 0x10031008, 0x10032008**

Bit	7	6	5	4	3	2	1	0
	RDTR		Reserved	UME	DME	TFRT	RFRT	FME
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7:6	RDTR	**Receive Buffer Data Number Trigger.** These bits are used to select the trigger level for the receive data ready interrupt in FIFO mode. 0b00: 1 0b01: 8 0b10: 16 0b11: 24	W

5	Reserved	Always read 0, write is ignored.	R
4	UME	**UART Module Enable.** 0: Disable UART 1: Enable UART	W
3	DME	**DMA Mode Enable.** 0: Disable DMA mode 1: Enable DMA mode	W
2	TFRT	**Transmit Holding Register Reset.** 0: Not reset 1: Reset transmit FIFO	W
1	RFRT	**Receive Buffer Reset.** 0: Not reset 1: Reset receive FIFO	W
0	FME	**FIFO Mode Enable.** Set this bit before the trigger levels. 0: non-FIFO mode 1: FIFO mode	W

23.2.7 UART Line Control Register (ULCR)

The ULCR defines the format for UART data transmission.

ULCR0. ULCR1, ULCR2, ULCR3 **0x1003000C, 0x1003100C, 0x1003200C**

Bit	7	6	5	4	3	2	1	0
	DLAB	SBK	STPAR	PARM	PARE	SBLS	WLS	
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7	DLAB	**Divisor Latch Access Bit.** 0: Enable to access URBR, UTHR or UIER 1: Enable to access UDLLR or UDLHR	W
6	SBK	**Set Break.** Causes a break condition (at least one 0x00 data) to be transmitted to the receiving UART. Acts only on the TXD pin and has no effect on the transmit logic. 0: No effect on TXD output 1: Forces TXD output to 0	W
5	STPAR	**Sticky Parity.** Setting this bit forces parity location to be opposite of PARM bit when PARE is 1 (it is ignored when PARE is 0).	W

		0: Disable Sticky parity 1: Enable Sticky parity (opposite o f PARM bit)	
4	PARM	**Parity Odd/Even Mode Select.** If PARE = 0, PARM is ignored. 0: Odd parity 1: Even parity	W
3	PARE	**Parity Enable.** Enables a parity bit to be generated on transmission or checked on reception. 0: No parity 1: Parity	W
2	SBLS	**Stop Bit Length Select.** Specifies the number of stop bits transmitted and received in each character. When receiving, the receiver checks only the first stop bit. 0: 1 stop bit 1: 2 stop bits, except for 5-bit character then 1-1/2 bits	W
1:0	WLS	**Word Length Select.** 0b00: 5-bit character 0b01: 6-bit character 0b10: 7-bit character 0b11: 8-bit character	W

23.2.8 UART Line Status Register (ULSR)

The read-only ULSR indicates status information during the data transfer. Receive error information in ULSR[4:1] remains set until software reads ULSR and it must be read before the error character is read.

ULSR0, ULSR1, ULSR2 **0x10030014, 0x10031014, 0x10032014**

Bit	7	6	5	4	3	2	1	0
	FIFOE	TEMP	TDRQ	BI	FMER	PARER	OVER	DRY
RST	0	1	1	0	0	0	0	0

Bits	Name	Description	RW
7	FIFOE	**FIFO Error Status. (FIFO mode only)** FIFOE is set when there is at least one kind of receive error (parity, frame, overrun, break) for any of the characters in receive buffer. FIFOE is reset when all error characters are read out of the buffer. During DMA transfer, the error interrupt generates when FIFOE is 1, and	R

		no receive DMA request generates even when data in receive buffer reaches the trigger threshold until all the error characters are read out. In non-DMA mode, FIFOE set does not generate error interrupt. 0: No error data in receive buffer or non-FIFO mode 1: One or more error character in receive buffer	
6	TEMP	**Transmit Holding Register Empty.** Set when both UTHR and shift register are empty. It is cleared when either the UTHR or the shift register contains a data character. 0: There is data in the transmit shifter and UTHR 1: All the data in the transmit shifter and UTHR has been shifted out	R
5	TDRQ	**Transmit Data Request.** Set when UTHR has half or more empty location (FIFO mode) or empty (non-FIFO mode). When both UIER.TDRIE and TDRQ are 1, transmit data request interrupt generates or during DMA transfer, DMA request to the DMA controller generates when UIER.TDRIE is 0 and TDRQ is 1. 0: There is one (non-FIFO mode) or more than half data (FIFO mode) in UTHR 1: None data (non-FIFO mode) or half or less than half data (FIFO mode) in UTHR	R
4	BI	**Break Interrupt.** BI is set when the received data input is held low for longer than a full-word transmission time (the total time of start bit + data bits + parity bit + stop bits). BI is cleared when the processor reads the ULSR. In FIFO mode, only one character equal to 0x00 is loaded into the FIFO regardless of the length of the break condition. BI shows the break condition for the character at the front of the FIFO, not the most recently received character. 0: No break signal has been received 1: Break signal received	R
3	FMER	**Framing Error.** Set when the bit following the last data bit or parity bit is detected to be 0. If the ULCR had been set for two or one and half stop bits, the other stop bits are not checked except the first one. In FIFO mode, FMER shows a framing error for the character at the front of the receive buffer, not for the most recently received character. Cleared when the processor reads the ULSR. 0: No framing error 1: Invalid stop bit has been detected	R
2	PARER	**Parity Error.**	R

		Indicates that the received data character does not have the correct even or odd parity, as selected by the even parity select bit. PARER is set upon detection if a parity error and is cleared when the processor reads the ULSR. In FIFO mode, PARER shows a parity error for the character at the front of the FIFO, not the most recently received character. 0: No parity error 1: Parity error has occurred	
1	OVER	**Overrun Error.** Set when both receive buffer and shifter are full and new data is received which will be lost. Cleared when the processor reads the ULSR. 0: No data has been lost 1: Receive data has been lost	R
0	DRY	**Data Ready.** Set when a complete incoming character has been received into the Receive Buffer registers. DRY is cleared when the receive buffer is read (non-FIFO mode) or when the buffer is empty or when the buffer is reset by setting UFCR.RFRT to 1. 0: No data has been received 1: Data is available in URBR	R

23.2.9 UART Modem Control Register (UMCR)

The UMCR uses the modem control pins RTS_ and CTS_ to control the interface with a modem or data set. UMCR also controls the loopback mode. Loopback mode must be enabled before the UART is enabled.

UMCR0, UMCR1, UMCR2 **0x10030010, 0x10031010, 0x10032010**

Bit	7	6	5	4	3	2	1	0
	MDCE	FCM	Reserved	LOOP	Reserved	Reserved	RTS	Reserved
RST	0	0	0	0	0	0	0	0

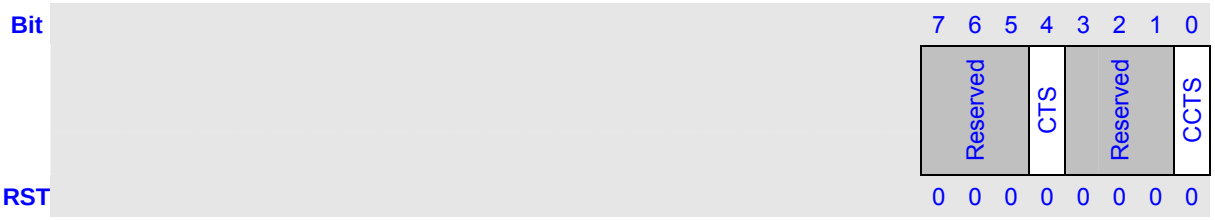
Bits	Name	Description	RW
7	MDCE	**Modem Control Enable.** 0: Modem function is disabled 1: Modem function is enabled	W
6	FCM	**Flow Control Mode.** 0: Flow control by software 1: Flow control by hardware	
5	Reserved	Always read 0, write is ignored.	R

4	LOOP	**Loop Back.** This bit is used for diagnostic testing of the UART. When LOOP is 1, TXD output pin is set to a logic 1 state, RXD is disconnected from the pin, and the output of the transmitter shifter register is looped back into the receiver shift register input internally, similar to CTS_ and RTS_ pins and the RTS bit of the UMCR is connected to CTS bit of UMSR respectively. Loopback mode must be selected before the UART is enabled. 0: Normal operation mode 1: Loopback-mode UART operation	W
3	Reserved	Always read 0, write is ignored.	R
2	Reserved	Always read 0, write is ignored.	R
1	RTS	**Request To Send.** This bit can control the RTS_ output state. 0: RTS_ force to high 1: RTS_ force to low	W
0	Reserved	Always read 0, write is ignored.	R

23.2.10 UART Modem Status Register (UMSR)

The read-only UMSR provides the current state of the control lines from the modem to the processor. They are cleared when the processor reads UMSR.

UMSR0, UMSR1, UMSR2 **0x10030018, 0x10031018, 0x10032018**



Bits	Name	Description	RW
7	Reserved	Always read 0, write is ignored.	R
6	Reserved	Always read 0, write is ignored.	R
5	Reserved	Always read 0, write is ignored.	R
4	CTS	**Status of Clear To Send.** When MDCE bit is 1, this bit is the complement of CTS_ input. If Loop bit of UMCR is 1, this bit is equivalent to RTS bit of UMCR. 0: CTS_ pin is 1 1: CTS_ pin is 0	R
3	Reserved	Always read 0, write is ignored.	R
2	Reserved	Always read 0, write is ignored.	R
1	Reserved	Always read 0, write is ignored.	R

0	CCTS	**Change status of CTS_.** When MDCE bit is 1, this bit indicates the state change on CTS_ pin. 0: No state change on CTS_ pin since last read of UMSR 1: A change occurs on the state of CTS_ pin	R

23.2.11 UART Scratchpad Register

This Scratchpad register is used as a scratch register for the programmer and has no effect on the UART.

USPR **0x1003001C**

Bit	7	6	5	4	3	2	1	0
	Scratch Data							
RST	?	?	?	?	?	?	?	?

23.2.12 Infrared Selection Register (ISR)

The ISR is used to configure the slow-infrared (SIR) interface that is provided in each UART to support two-way wireless communication using infrared transmission that conforms to the IrDA serial infrared specification 1.1. The maximum frequency is up to 115.2kbps.

ISR **0x10030020**

Bit	7	6	5	4	3	2	1	0
	Reserved			RDPL	TDPL	XMODE	RCVEIR	XMITIR
RST	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
7:5	Reserved	Always read 0, write is ignored.	R
4	RDPL	**Receive Data Polarity.** 0: Slow-infrared (SIR) interface decoder takes positive pulses as zeros 1: SIR decoder takes negative pulses as zeros	W
3	TDPL	**Transmit Data Polarity.** 0: SIR encoder generates a positive pulse for a data bit of zero 1: SIR encoder generates a negative pulse for a data bit of zero	W
2	XMODE	**Transmit Pulse Width Mode.** Set when the transmit encoder needs to generate 1.6us pulses (that are 3/16 of a bit-time at 115.2 kbps). Cleared when the transmit encoder needs to generate 3/16 of a bit-time	W

		wide according to current baud rate. 0: Transmit pulse width is 3/16 of a bit-time wide 1: Transmit pulse width is 1.6 us	
1	RCVEIR	**Receiver SIR Enable.** This bit is used to select the signal from the RXD pin is processed by the IrDA decoder before it is fed to the UART (RCVEIR = 1) or bypass IrDA decoder and is fed directly to the UART (RCVEIR = 0). 0: Receiver is in UART mode 1: Receiver is in SIR mode	W
0	XMITIR	**Transmitter SIR Enable.** This bit is used to select TXD output pin is processed by the IrDA encoder before it is fed to the device pin (XMITIR = 1) or bypass IrDA encoder and is fed directly to the device pin (XMITIR = 0). **NOTE:** disable infrared LED before XMITIR is set, otherwise a false start bit may occur. 0: Transmitter is in UART mode 1: Transmitter is in SIR mode	W

23.2.13 UART M Register (UMR)

UMSR0, UMSR1, UMSR2 **0x10030024, 0x10031024, 0x10032024**

Bit	7	6	5	4	3	2	1	0
	Reserved			M				
RST	0	0	0	0	0	0	0	0

M is the value of UMR register.
It will take UART at least M cycles to transmit one bit and receiver to receive one bit.
It will take UART at most M+1 cycles for transmitter to transmit one bit and receiver to receive one bit.

23.2.14 UART Add Cycle Register (UACR)

UACR0, UACR1, UACR2 **0x10030028, 0x10031028, 0x10032028**

Bit	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved				AC							
RST	0	0	0	0	0	0	0	0	0	0	0	0

If nth bit of the register is 1, it will take UART M+1 cycles to transmit or receive the bit of date for transmit or receive.
If the register is 12'h0, UART will receive or transmit a bit by M cycle.
If the register is 12'hfff, UART will receive or transmit a bit by M+1 cycle.
For the detail to see For any frequency clock to use the Uart.

23.3 Operation

The following sections describe the UART operations that include flow of configuration, data transmission, data reception, and Infrared mode.

23.3.1 UART Configuration

Before UART starts to transfer data or changing transfer format, configuration must be done to define the transfer format. The sample flow is as the following:

In FIFO mode, set FME bit of UFCR to 1, reset receive and transmit FIFO, then initialize the UART as described below:

1 Clear UFCR.UME to 0.
2 Set value in UDLL/UDHR to generate the baud rate clock.
3 Set data format in ULCR.
4 If it is in FIFO MODE, set FME bit and other FIFO control in UFCR, reset receive and transmit FIFO, otherwise skip item 4.
5 Set each interrupt enable bit in UIER in interrupt-based transfer or set UFCR.DME in DMA-based transfer (DMA transfer is FIFO mode only), then set UFCR.UME.

23.3.2 Data Transmission

After configuration, UART is ready for data transfer. For data transmission, refer to the following procedure:

1 Read ULSR.TDRQ (interrupt disable) or wait for transmit data request interrupt (interrupt enable), if TDRQ = 1 or transmit data request interrupt generates, that means there is enough empty location in UTHR for new data.
2 If ULSR.TDRQ is 1 or get the transmit data request interrupt, write transmit data to UTHR to start transmission.
3 Do item 1 and item 2 if there are more data waiting for transmit.
4 After all necessary data are written to UTHR, wait ULSR.TEMP = 1, that means all data completely transmitted.
5 If it is necessary to send break, set ULCR.SBK and at least wait for 1-bit interval time to send a valid break, then clear ULCR.SBK.
6 Clear UME bit to finish UART transmission.

23.3.3 Data Reception

After configuration, UART is ready for data transfer. For data reception, refer to the following sample procedure:

1 Read ULSR.DRY (interrupt disable) or wait for receive data request interrupt (interrupt enable), if ULSR.DRY =1 or receive data request interrupt generates, that means URBR has one data (non-FIFO mode) or data in URBR reaches the trigger value. (FIFO mode)
2 If ULSR.DRY = 1 or receive data request interrupt generates, then read ULSR.FIFOE or see if

there is error interrupt, if FIFOE = 1, it means received data has receive error, then go to error handler, other wise go to item 3.

3 Read one received data in URBR (non-FIFO mode) or data equal to trigger value in URBR. (FIFO mode)

4 Check whether all data received: check whether ULSR.DRY = 0, in FIFO mode and interrupt is enabled, timeout interrupt may generate, when timeout interrupt generates, read URBR till ULSR.DRY = 0.

5 Clear UFCR.UME to end data reception when all data are received and ULSR.DRY = 0.

23.3.4 Receive Error Handling

A sample error handling flow is as the following:

1 If ULSR.FIFOE = 1, it means there is receive error in received data, then check what error it is.

2 If ULSR.OVER = 1, go to OVER error handling.

3 If ULSR.BI = 1, go to Break handling.

4 If ULSR.FMER = 1, go to Frame error handling.

5 If PARER = 1, go to PARER error handling.

23.3.5 Modem Transfer

When UMCR.MDCE = 1, modem control is enabled. Transfer flow can be stopped and restarted by software through RTS_ and CTS_ pin. When UART transmitter detects low level on CTS_ pin, it stops transmission and TxD pin goes to mark state after finishing transmitting the current character until it detects CTS_ pin goes back to high level. RTS_ pin is output to receiving UART and its state can be controlled by setting UMCR.RTS bit, that is, setting UMCR.RTS to 1, RTS_ pin is low level output that means UART is ready to receive data, on the contrary, it means UART currently can't receive more data.

23.3.6 DMA Transfer

UART can operate in DMA-based (UFCR.DME = 1, FIFO mode only), that is, dma request initiated by UART takes the place of interrupt request and transmission/reception is carried out using DMA instead of CPU. Be sure that software guarantee to disable transmit and receive interrupt except timeout and error interrupts.

During DMA transfer, if an interrupt occurs, software must first read the ULSR to see if an error interrupt exists, then check the UIIR for the source of the interrupt and if DMA channel is already halt because of the error indicator from UART, then disable DMA channel and read out all the error data from receive FIFO. Software re-set and re-enable DMA and data transfer by DMA will re-start.

23.3.7 Slow IrDA Asynchronous Interface

Each UART supports slow infra-red (SIR) transmission and reception by setting ISR.XMITIR and ISR.RCVEIR to 1 (make sure the two bits are not set to 1 at the same time because SIR can't operate full-duplex). According to the IrDA 1.1, data rate is limited at a maximum value of 115.2Kbps.

In SIR transmit mode, the transmit pulse comes out at a rate of 3/16 (when the transmit data bit is zero); in SIR receive mode, the receiver must detect the 3/16 pulsed period to recognize a zero value (an active high or low pulse is demodulation to 0, and no pulse is demodulation to 1).

Compared to normal UART, there are some limitations to SIR, that is, each character is fixed to 8-bit data width, no parity and 1 stop bit and modem function is ignored. The IrDA 1.1 specifies a minimum 10ms latency after an optical node ceases transmitting before its receiver recovers its receiving function and software must guarantee this delay.

In the IrDA 1.1 specification, communication must start up at the rate of 9600bps, but then allows the link to negotiate higher (or lower) data rates if supported by both ends. However, the communication rate will not automatically change. Change, if necessary, is performed by software.

23.3.8 For any frequency clock to use the UART

NOTE: if you don't set M register and UACR the UART work at normal mode with the specified frequencies. To use other frequency you should to set M register and UACR to right value.

1 The Improving

Following changes are made:

a One bit is composed by M CLK_{BR} cycles, which can be 4~1024.

b Some extra CLK_{BR} cycles can be inserted in some bits in one frame, so that like M has fraction.

For instance:

$CLK_{BR} = CLK_{DEV} / N$ N = 1, 2, …

$CLK_{BR} = CLK_{DEV}$ = 4MHz

Band rate = 460800

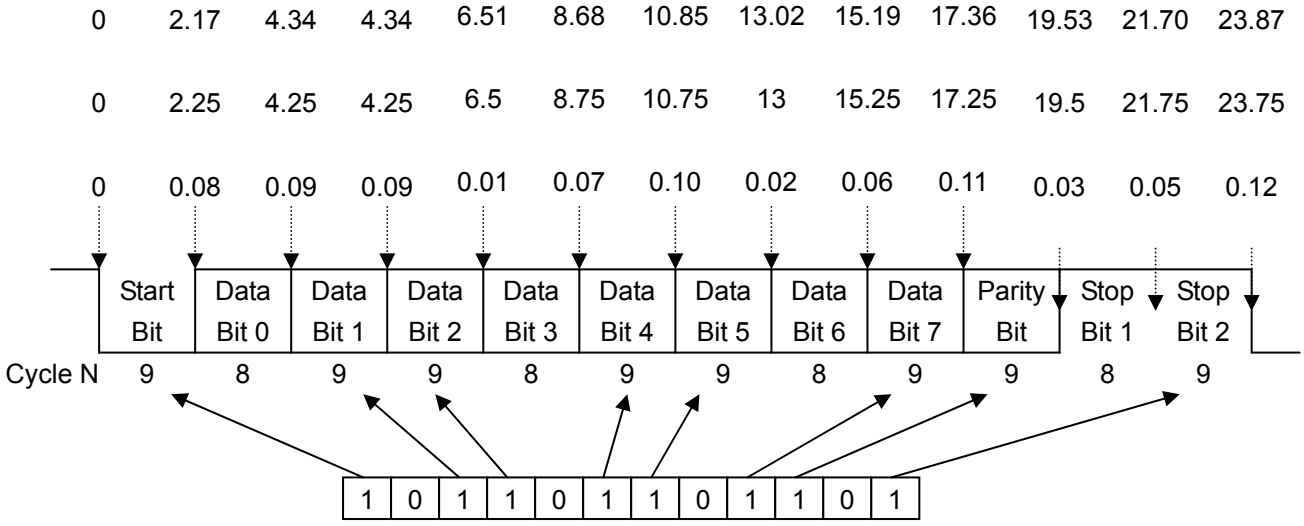
In accurate

$M_a = 8.681$

We take

M = 8, with 8 extra cycles in every frame

A 12-bit register is used to indicate where to insert the extra cycles.



For transmission, in theory, the biggest error is half of CLK_{BR} cycle, which is 0.125us here.

2 To set UMR register

$CLK_{BR} = CLK_{DEV} / N$

$M_a = CLK_{BR}$/band rate

M is modem of M_a.

Write M to Mregister.

Considering the power and the robust quality, for M form 6 to 32 is you better select by set the UDLR.

The max error

$$\frac{0.5 / CLK_{BR}}{M_a / CLK_{BR}} = 0.5/M_a < 0.5/M$$

M	4	8	16	32	64
error/W_{bit}	12.5%	6.25%	3.125%	1.56%	0.78%

3 To set UACR value

For each bit of it means:

0: means not to add additional cycle to the bit that UART is prepare to transmit or receive, in another word, you will to use M cycles to transmit or receive the bit

1: means to add additional cycle to the bit that UART is prepare to transmit or receive, in another word, you will to use M+1 cycles to transmit or receive the bit

To set UACR value you must ensure that the max error of each bit should be less than $0.5P_{BR}$.

For example：M_a-M =0.15; M+1-M_a=0.85;

Write UMR 8

Write UMR 408

cycle/bit	：	M，	M，	M，	M+1，	M，	M，	M，	M，	M，	M，	M+1，	M
UACR	：	0	0	0	1	0	0	0	0	0	0	1	0

24 XBurst Boot ROM Specification

JZ4725B contains an internal 8KB boot ROM. The CPU boots from the boot ROM after reset.

24.1 Boot Select

The boot sequence of JZ4725B is controlled by boot_sel[1:0]. The configuration of boot_sel[1:0] is shown as follow:

Table 24-1 Boot Configuration of JZ4725B

boot_sel[1:0]	
00	Boot from MMC/SD card at MSC0
01	Boot from USB device
10	Reserved
11	Boot from NAND flash at CS1

24.2 Boot Sequence

After reset, the boot program on the internal boot ROM executes as follows:

1. Read boot_sel[1:0] and branch to proper programs according to it.
2. If it is boot from NAND flash (“11”), first 12 bytes are read from the NAND to know whether the bus of NAND is shared with SDRAM or not, and to know the NAND information including bus width(8 or 16 bits), page cycle(2 or 3 cycles) and its page size(512, 2KB or 4KB bytes). Then EMC is set according to the information and 8KB are read out from NAND to internal SRAM, if the 8KB reading failed, the adjacent 8KB backup in NAND will be read. Then branch to SRAM at 12 bytes offset.
3. If it is boot from MMC/SD card at MSC0, its function pins MSC0_D0, MSC0_CLK, MSC0_CMD are initialized, the boot program loads the 8KB data from MMC/SD card to internal SRAM and jump to it. Only one data bus which is MSC0_D0 is used. The clock EXTCLK/128 is used initially. When reading data, the clock EXTCLK/2 is used.
4. If it is boot from USB (“01”), a block of data will be received through USB cable connected with host PC and be stored in internal SRAM. Then branch to this area in SRAM.

NOTE: The JZ4725B internal SRAM is 16KB, its address is from 0x80000000 to 0x80004000.

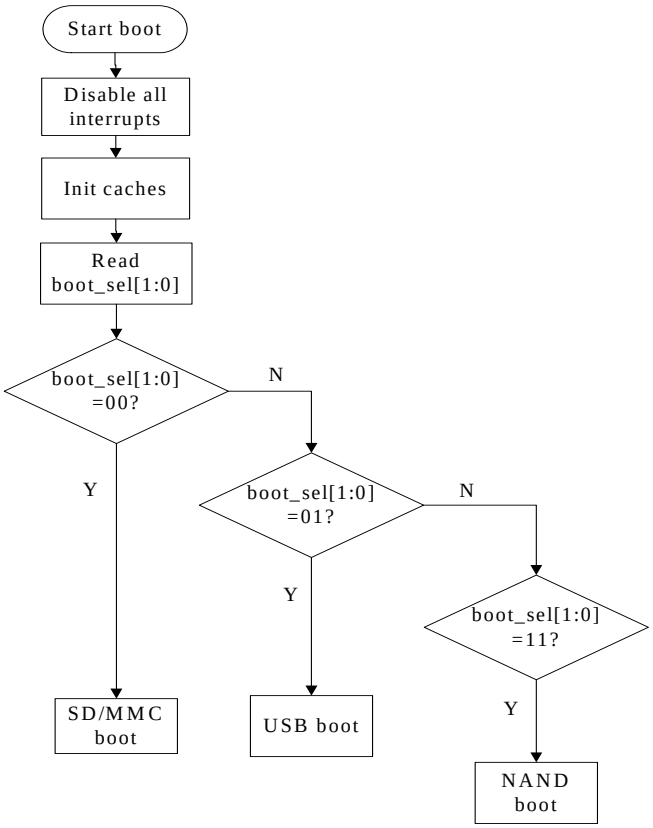


Figure 24-1 Boot sequence diagram of JZ4725B

24.3 NAND Boot Specification

If CPU boots from NAND flash (CS1), the boot ROM will read bit PK_SEL in EMC's bus control register EMC_BCR to know that the data bus is in normal order or special order, and then it will read the first 12 bytes from NAND flash to know whether the bus of NAND is shared with SDRAM or not, and to know the NAND information including bus width(8 or 16 bits), page cycle(2 or 3 cycles) and its page size(512, 2KB or 4KB bytes).

The content and definition of the first 12 bytes are shown as follow:

Table 24-2 The content of the first 12 bytes in NAND flash

Content	Bus width	0x55	0x55	0x55	0x55	0x55	0x55	0x55	Row address cycles	Page size0	Page size1	Reserved
Index	1	2	3	4	5	6	7	8	9	10	11	12

Table 24-3 The definition of the first 12 bytes in NAND flash

Byte index	Name	Value	Description
1	Bus width	0xff/0x00	Bus width. 0xff: 8bit bus width. 0x00: 16bit bus width.
2-8	Check code	0x55	Check code.
9	Row address cycles	0xff/0x00	The number of row cycles. 0xff: 3 row cycles. 0x00: 2 row cycles.
10	Page size0	0xff/0x00	Page size. 0xff: 2KB or 4KB. 0x00: 512B.
11	Page size1	0xff/0x00	Page size. 0xff: 2KB. 0x00: 4KB.
12	Reserved	XX	Reserved, not used.

Table 24-4 NAND Spare Area Definition

Spare Area Offset	Description
0 – 2	Reserved.
3 – 15	Stored 13-byte ECC of data0 – data511.
16 - 28	Stored 13-byte ECC of data512 – data1023 (2/4KB page NAND only).
29 – 41	Stored 13-byte ECC of data1024 – data1535 (2/4KB page NAND only).
42 – 54	Stored 13-byte ECC of data1536 – data2047 (2/4KB page NAND only).
55 – 67	Stored 13-byte ECC of data2048 – data2559 (4KB page NAND only).

68 - 80	Stored 13-byte ECC of data2560 – data3071 (4KB page NAND only).
81 – 93	Stored 13-byte ECC of data3072 – data3583 (4KB page NAND only).
94 – 106	Stored 13-byte ECC of data3584 – data4095 (4KB page NAND only).

At the beginnig of reading the 12 bytes, the check codes 0x55s from 2nd to 8th bytes will be read firstly to detect the bus share mode of EMC (share or unshare between NAND and SDRAM) and preliminarily the page size of NAND. Then 1st byte will be read to get the bus width of NAND, 9th byte to get the number of row cycles, 10th and 11th bytes to get the page size of NAND finally.

EMC and GPIO pins will be configured according to the information get above, then 8KB from NAND is loaded up to internal SRAM and branch to internal SRAM at 12 bytes offset.

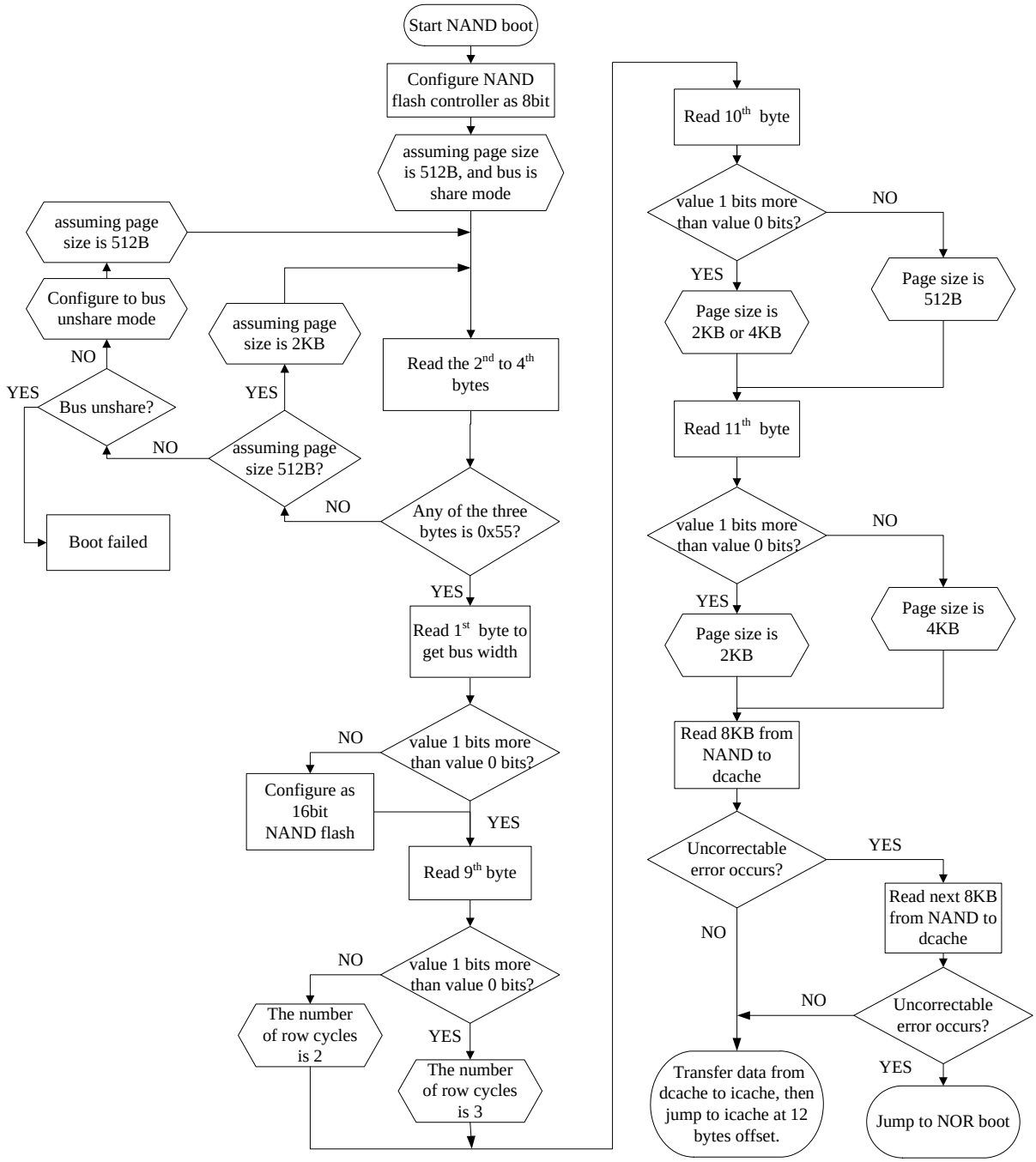


Figure 24-2 JZ4725B NAND Boot Sequence

The boot program can load two areas of data from NAND flash to internal SRAM, one is the normal area up to 8KB starting from NAND flash address 0, the other is the backup area up to 8KB starting from NAND flash address 0x2000. After reset, the boot program will firstly read the normal area data from NAND flash using hardware 8-bit BCH ECC. If no ECC error is detected or ECC error is correctable(number of error bits <= 8), the boot program then branches to internal SRAM at 12 bytes offset. If an uncorrectable ECC error is detected, the 8KB backup area of data will be read out from NAND flash using hardware 8-bit BCH ECC. If no ECC error is detected or ECC error is correctable,

the boot program will then branch to internal SRAM at 12 bytes offset. If it detects an uncorrectable ECC error again, it will continue to start booting from NOR flash at CS4.

The boot program enables 8-bit hardware BCH ECC when reading NAND flash data. When a 512-byte data is read, it will check the calculated ECC with stored ECC. The calculated and stored ECC are both 13 bytes per 512-byte data. And the 13-byte stored ECC is starting from the 4th byte of the spare area of each page.

The NAND spare area definition is shown as Table 24-4.

The procedure of the JZ4725B NAND boot is shown as Figure 24-2.

24.4 USB Boot Specification

When boot_sel[1:0] is selected as USB boot, the internal boot ROM downloads user program from the USB port to internal SRAM and branches to the internal SRAM to execute the program.

A 12MHz clock is required for the USB device controller to work normally, so the USB boot code needs to decide the main crystal frequency first. After it has decided the main crystal frequency, it will set the PLL and get a 12MHz clock for the USB device controller.

The boot program supports both high-speed (480MHz) and full-speed (12MHz) transfer modes. The boot program uses the following two transfer types.

Table 24-5 Transfer Types Used by the Boot Program

Transfer Type	Description
Control Transfer	Used for transmitting standard requests and vendor requests.
Bulk Transfer	Used for responding to vendor requests and transmitting a user program.

The following figure shows an overview of the USB communication flow.

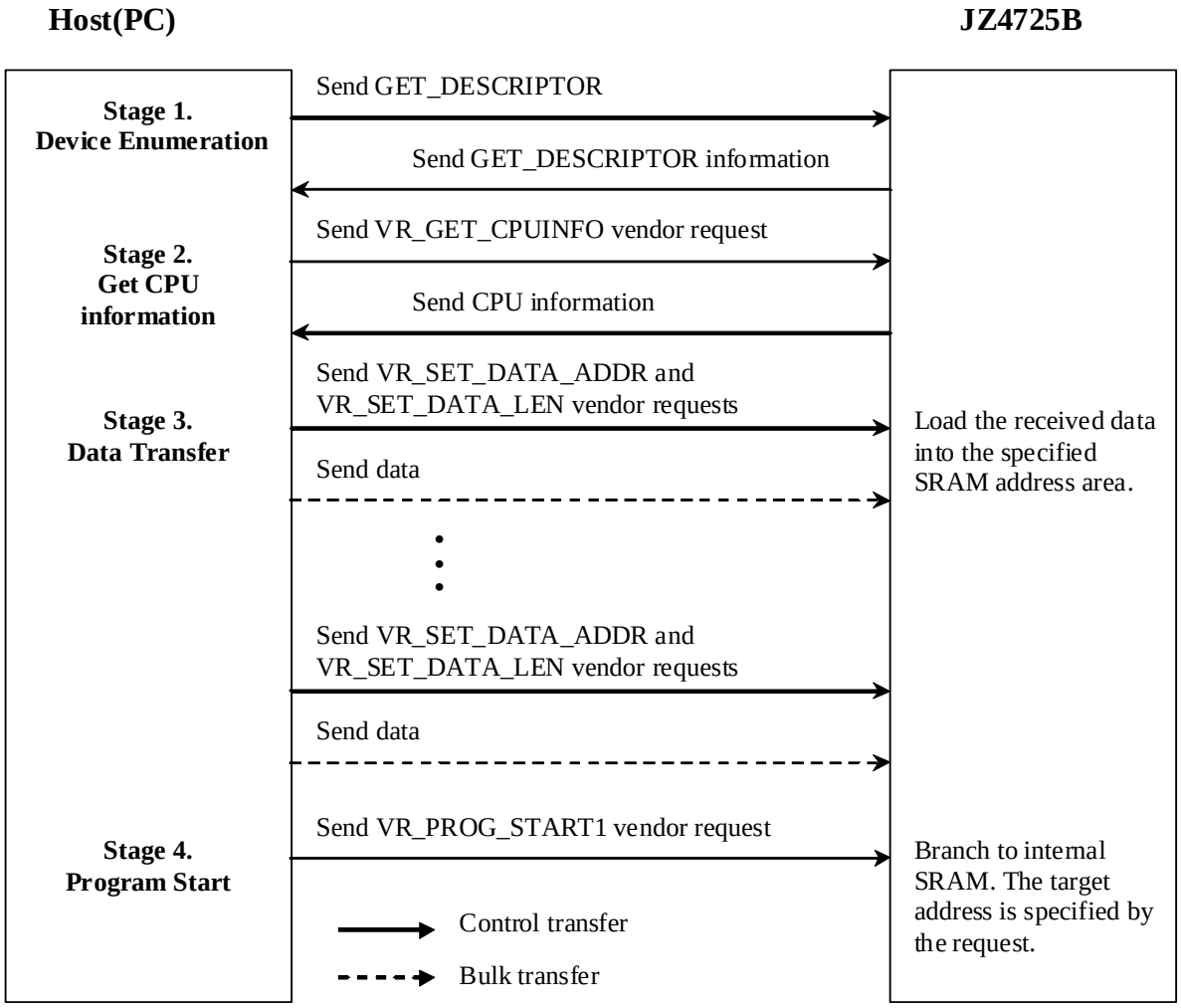


Figure 24-3 USB Communication Flow

The vendor ID and product ID for the USB boot device are 0x601A and 0x4750 respectively. The Configuration for USB is for Control Endpoint 0 with Max Packet Size equals 64 bytes, Bulk IN at Endpoint 1 with Max Packet Size equals 512 bytes in high-speed and 64 bytes in full-speed, Bulk OUT at Endpoint 1 with Max Packet Size equals 512 bytes in high-speed and 64 bytes in full-speed.

The USB boot program provides six vendor requests through control endpoint for user to download/upload data to/from device, and to branch to a target address to execute user program. The six vendor requests are VR_GET_CPU_INFO (0x00), VR_SET_DATA_ADDRESS (0x01), VR_SET_DATA_LENGTH (0x02), VR_FLUSH_CACHES (0x03), VR_PROGRAM_START1 (0x04) and VR_PROGRAM_START2 (0x05). User program is transferred through Bulk IN or Bulk OUT endpoint.

When JZ4725B is reset with boot_sel[1:0] equals 11b, the internal boot ROM will switch to USB boot

mode and wait for USB requests from host. After connecting the USB device port to host, host will recognize the connection of a USB device, and start device enumeration. After finishing the device enumeration, user can send VR_GET_CPU_INFO (0x00) to query the device CPU information. If user wants to download/upload a program to/from device, two vendor requests VR_SET_DATA_ADDRESS (0x01) and VR_SET_DATA_LENGTH (0x02) should be sent first to tell the device the address and length in byte of the subsequent transferring data. Then data can be transferred through bulk-out/bulk-in endpoint. After this first stage program has been transferred to device, user can send vendor request VR_PROGRAM_START1 (0x04) to let the CPU to execute the program. This first stage program must not greater than 16KB and is normally used to init GPIO and SDRAM of the target board. At the end of the first stage program, it can return back to the internal boot ROM by jumping to ra ($31) register. Thus user can download a new program to the SDRAM of the target board like the first stage, and send vendor request VR_FLUSH_CACHES (0x03) and VR_PROGRAM_START2 (0x05) to let the CPU to execute the new program. Next figure is the typical procedure of USB boot.

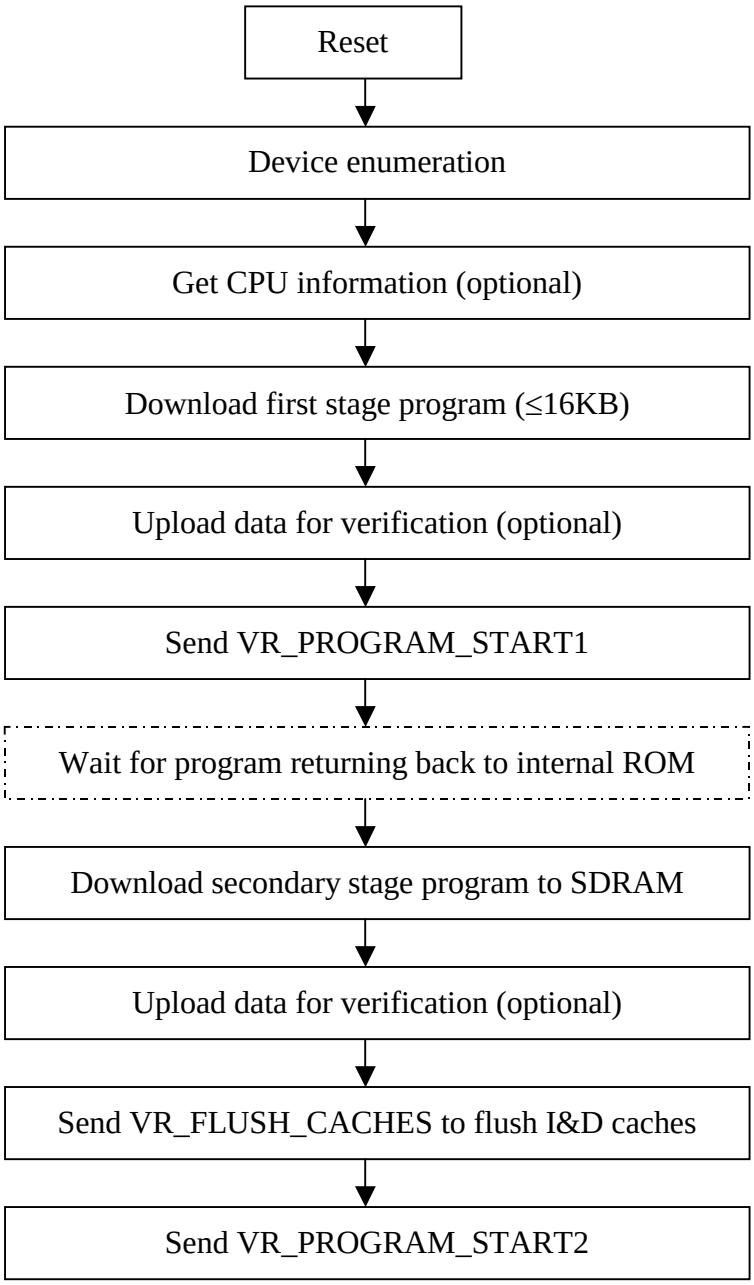


Figure 24-4 Typical Procedure of USB Boot

Following tables list all the vendor requests that USB boot program supports:

Table 24-6 Vendor Request 0 Setup Command Data Structure

Offset	Field	Size	Value	Description
0	bmRequestType	1	40H	D7 0: Host to Device. D6-D5 2: Vendor. D4-D0 0: Device.
1	bRequest	1	00H	VR_GET_CPU_INFO: get CPU information.
2	wValue	2	0000H	Not in used.
4	wIndex	2	0000H	Not in used.
6	wLength	2	0008H	8 bytes.

Table 24-7 Vendor Request 1 Setup Command Data Structure

Offset	Field	Size	Value	Description
0	bmRequestType	1	40H	D7 0: Host to Device. D6-D5 2: Vendor. D4-D0 0: Device.
1	bRequest	1	01H	VR_SET_DATA_ADDRESS: set address for next bulk-in/bulk-out transfer.
2	wValue	2	xxxxH	MSB (bit[31:16]) of the data address.
4	wIndex	2	xxxxH	LSB (bit[15:0]) of the data address.
6	wLength	2	0000H	Not in used.

Table 24-8 Vendor Request 2 Setup Command Data Structure

Offset	Field	Size	Value	Description
0	bmRequestType	1	40H	D7 0: Host to Device. D6-D5 2: Vendor. D4-D0 0: Device.
1	bRequest	1	02H	VR_SET_DATA_LENGTH: set length in byte for next bulk-in/bulk-out transfer.
2	wValue	2	xxxxH	MSB (bit[31:16]) of the data length.
4	wIndex	2	xxxxH	LSB (bit[15:0]) of the data length.
6	wLength	2	0000H	Not in used.

Table 24-9 Vendor Request 3 Setup Command Data Structure

Offset	Field	Size	Value	Description
0	bmRequestType	1	40H	D7 0: Host to Device. D6-D5 2: Vendor. D4-D0 0: Device.

1	bRequest	1	03H	VR_FLUSH_CACHES: flush I-Cache and D-Cache.
2	wValue	2	0000H	Not in used.
4	wIndex	2	0000H	Not in used.
6	wLength	2	0000H	Not in used.

Table 24-10 Vendor Request 4 Setup Command Data Structure

Offset	Field	Size	Value	Description
0	bmRequestType	1	40H	D7 0: Host to Device. D6-D5 2: Vendor. D4-D0 0: Device.
1	bRequest	1	04H	VR_PROGRAM_START1: transfer data from D-Cache to I-Cache and branch to address in I-Cache. **NOTE:** After downloading program from host to device for the first time, you can only use this request to start the program. Since the USB boot program will download data to D-Cache after reset. This request will transfer data from D-Cache to I-Cache and execute the program in I-Cache.
2	wValue	2	xxxxH	MSB (bit[31:16]) of the program entry point.
4	wIndex	2	xxxxH	LSB (bit[15:0]) of the program entry point.
6	wLength	2	0000H	Not in used.

Table 24-11 Vendor Request 5 Setup Command Data Structure

Offset	Field	Size	Value	Description
0	bmRequestType	1	40H	D7 0: Host to Device. D6-D5 2: Vendor. D4-D0 0: Device.
1	bRequest	1	05H	VR_PROGRAM_START2: branch to target address directly.
2	wValue	2	xxxxH	MSB (bit[31:16]) of the program entry point.
4	WIndex	2	xxxxH	LSB (bit[15:0]) of the program entry point.
6	WLength	2	0000H	Not in used.

24.5 MMC/SD Boot Specification

If CPU boots from MSC0, the boot program will load 8KB data starting at sector 0 from MMC/SD card to internal SRAM. First the boot program initializes MSC0_D0, MSC0_CLK, MSC0_CMD as function pins. Only one data pin MSC0_D0 is used. Then the boot program sends CMD55 and CMD41 to test if it's SD or MMC card and initializes the card. At last it loads 8KB data from the card to SRAM and branches to execute the code in SRAM.

When initializing the card, the clock of EXTCLK/128 is used. And when reading data, the clock of EXTCLK/2 is used.

The procedure of the JZ4725B MMC/SD boot is shown as follow:

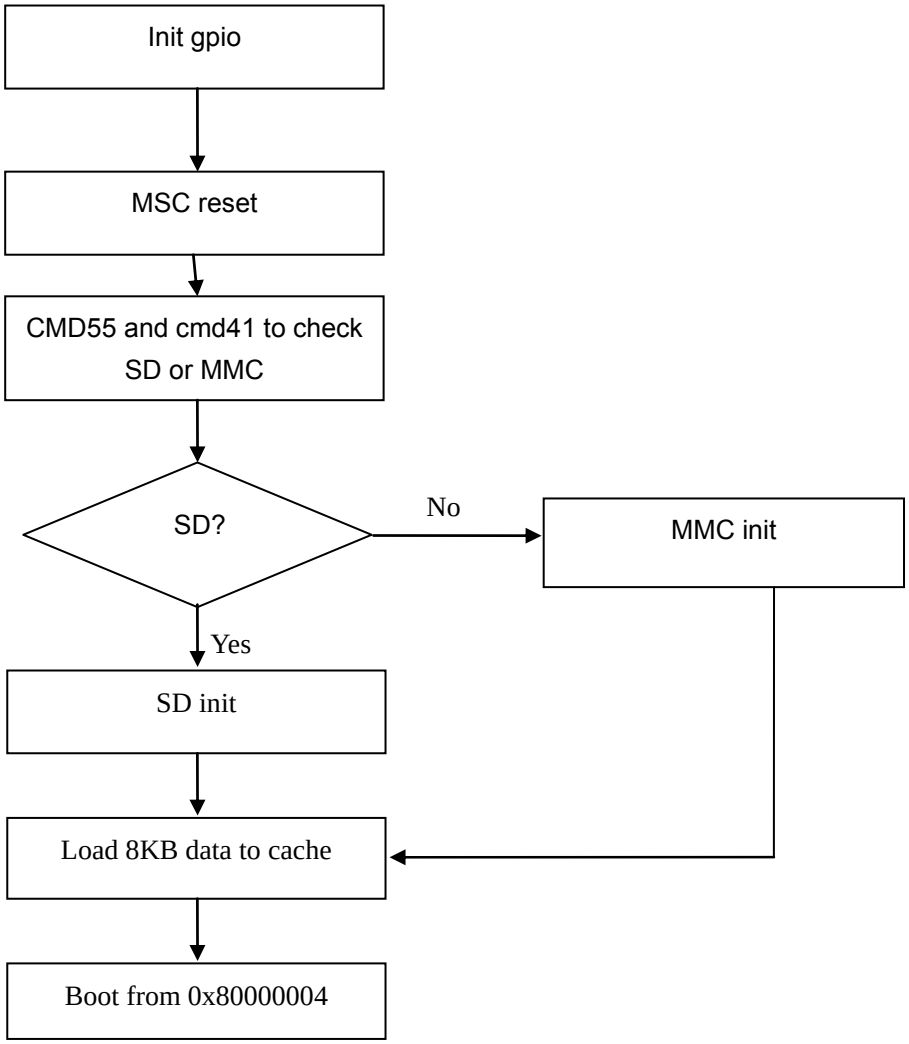


Figure 24-5 JZ4725B MMC/SD Boot Sequence

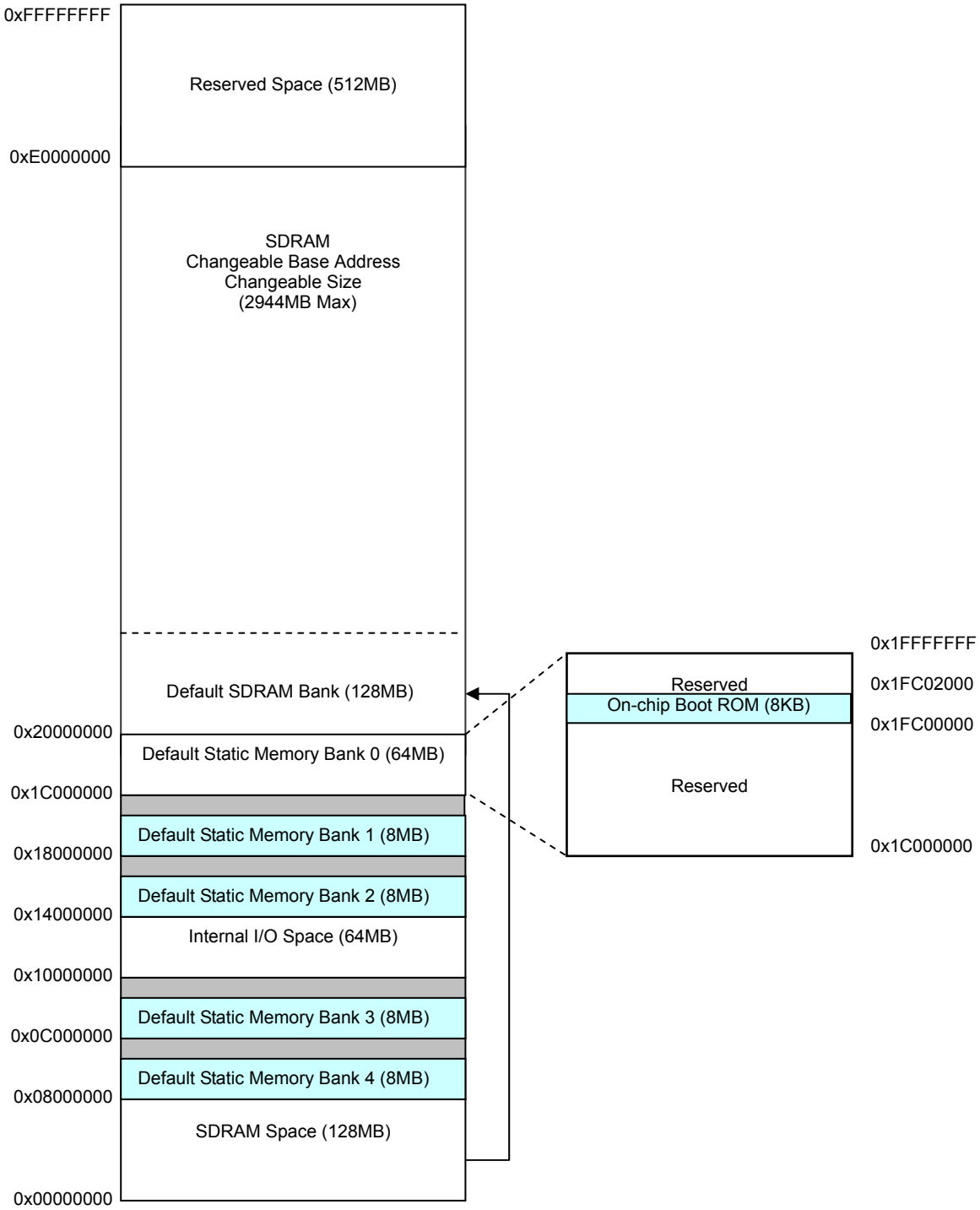
25 Memory Map and Registers

25.1 Physical Address Space Allocation

This chapter describes the physical address map, memory-mapped regions for every block in the JZ4725B processor. Both logical space and physical space of the JZ4725B are 32 bits wide. The 4Gbyte physical space is divided into several partitions for external memory, PCMCIA and internal I/O devices. Table 25-1 shows the basic physical memory map:

Table 25-1 JZ4725B Processor Physical Memory Map

Start Address	End Address	Size (MB)	Function
0x00000000	0x07FFFFFF	128	SDRAM Memory
0x08000000	0x087FFFFF	8	Static Memory, CS4#
0x08800000	0x0BFFFFFF	56	Reserved
0x0C000000	0x0C7FFFFF	8	Static Memory, CS3#
0x0C800000	0x0FFFFFFF	56	Reserved
0x10000000	0x10FFFFFF	16	I/O Devices on APB Bus
0x11000000	0x12FFFFFF	32	Reserved
0x13000000	0x13FFFFFF	16	I/O Devices on AHB Bus
0x14000000	0x147FFFFF	8	Static Memory, CS2#
0x14800000	0x17FFFFFF	56	Reserved
0x18000000	0x187FFFFF	8	Static Memory, CS1#
0x18800000	0x1BFFFFFF	56	Reserved
0x1C000000	0x1FBFFFFF	60	Reserved
0x1FC00000	0x1FC01FFF	0.008	On-chip Boot ROM (8kB)
0x1FC02000	0x1FFFFFFF	3.992	Reserved
0x20000000	0xDFFFFFFF	3072	SDRAM Memory
0xE0000000	0xFFFFFFFF	512	Reserved



The JZ4725B processor AHB bus devices are mapped at the addresses based at 0x13000000, and each device is allocated for 64KB space. Table 25-2 lists the complete addresses:

Table 25-2 AHB Bus Devices Physical Memory Map

Module	Start Address	End Address	Size (KB)	Description
HARB	0x13000000	0x1300FFFF	64	AHB Bus Arbiter
EMC	0x13010000	0x1301FFFF	64	External Memory Controller
DMAC	0x13020000	0x1302FFFF	64	DMA Controller
	0x13030000	0x1303FFFF	64	Reserved
UDC	0x13040000	0x1304FFFF	64	USB 2.0 Device Controller
LCDC	0x13050000	0x1305FFFF	64	LCD Controller
	0x13060000	0x1306FFFF	64	Reserved
	0x13070000	0x1307FFFF	64	Reserved
IPU	0x13080000	0x1308FFFF	64	Image Process Unit
	0x13090000	0x1309FFFF	64	Reserved
	0x130A0000	0x130AFFFF	64	Reserved
	0x130B0000	0x130BFFFF	64	Reserved
	0x130C0000	0x130CFFFF	64	Reserved
BCH	0x130D0000	0x130DFFFF	64	BCH Controller
	0x130E0000	0x130EFFFF	64	Reserved
	0x130F0000	0x130FFFFF	64	Reserved
	0x13100000	0x1310FFFF	64	Reserved
	0x13110000	0x138FFFFF	8128	Reserved
	0x13900000	0x1390FFFF	64	Reserved
	0x13910000	0x139FFFFF	960	Reserved
	0x13A00000	0x13A0FFFF	64	Reserved
	0x13A10000	0x13F0FFFF	5120	Reserved
	0x13F10000	0x13FFFFFF	960	Reserved

The JZ4725B processor APB bus devices are based at 0x10000000, and each device is allocated for 4KB space. Table 25-3 lists the complete addresses:

Table 25-3 APB Bus Devices Physical Memory Map

Module	Start Address	End Address	Size (KB)	Description
CPM	0x10000000	0x10000FFF	4	Clocks and Power Manager
INTC	0x10001000	0x10001FFF	4	Interrupt Controller
TCU OST WDT	0x10002000	0x10002FFF	4	Timer/Counter Unit Operating System Timer Watchdog Timer
RTC	0x10003000	0x10003FFF	4	Real-Time Clock
	0x10004000	0x1000FFFF	48	Reserved
GPIO	0x10010000	0x10010FFF	4	General-Purpose I/O
	0x10011000	0x1001FFFF	60	Reserved
AIC CODEC	0x10020000	0x10020FFF	4	AC97/I2S Controller Embedded CODEC
MSC	0x10021000	0x10021FFF	4	MMC/SD Controller
MSC1	0x10022000	0x10022FFF	4	MMC/SD 1 Controller
	0x10023000	0x1002FFFF	52	Reserved
UART0	0x10030000	0x10030FFF	4	UART 0
UART1	0x10031000	0x10031FFF	4	UART 1
UART2	0x10032000	0x10032FFF	4	UART 2
	0x10033000	0x10033FFF	4	Reserved
	0x10034000	0x1003FFFF	48	Reserved
	0x10040000	0x10040FFF	4	Reserved
	0x10041000	0x10041FFF	4	Reserved
I2C	0x10042000	0x10042FFF	4	I2C Bus Interface
	0x10043000	0x10043FFF	4	Reserved
	0x10044000	0x10044FFF	4	Reserved
	0x10045000	0x10045FFF	4	Reserved
	0x10050000	0x10050FFF	4	Reserved
	0x10051000	0x10051FFF	4	Reserved
	0x10052000	0x1005FFFF	56	Reserved
	0x10060000	0x10060FFF	4	Reserved
	0x10061000	0x10061FFF	4	Reserved
	0x10062000	0x10062FFF	4	Reserved
	0x10063000	0x1006FFFF	52	Reserved
SADC	0x10070000	0x10070FFF	4	SAR A/D Controller
	0x10071000	0x10071FFF	4	Reserved
	0x10072000	0x10072FFF	4	Reserved

	0x10073000	0x10073FFF	4	Reserved
	0x10074000	0x10FFFFFF	15920	Reserved