

# 1 Introduction

## 1.1 Overview

JzRISC core is a high performance and low power implementation of RISC processor. In architecture level, JzRISC implements MIPS32 Release 2 instruction set architecture; in micro-architecture level, JzRISC deploys a new 8 stage pipelining structure; in user level, it is basically follow MIPS 4KE core user programming interface with some kinds of adjustment. This document summarizes JzRISC's difference/adjustment from MIPS 4KE core (marked with **red color**). MIPS document No. MD00103, "MIPS32™ 4KE™ Processor Cores Software User's Manual" is a supplement to this document.

## 1.2 Features

Table 1-1 JzRISC Core Features

Item	Features
JzRISC CPU	<ul style="list-style-type: none"><li>• MIPS32 ISA Release I and II.</li><li>• 32 32-bit general purpose registers, <b>no shadow GPR</b></li><li>• 8-stage pipeline</li><li>• Interlocked implementation</li><li>• Virtual address space: 4 G-Bytes</li></ul>
Multiply-Divide Unit (MDU)	<ul style="list-style-type: none"><li>• Maximum issue rate of one 32x16 multiply every clock</li><li>• Maximum issue rate of one 32x32 multiply every other clock</li><li>• Minimum 2 clock cycle, maximum 34 clock cycles for divide</li></ul>
Memory Manager Unit (MMU)	<ul style="list-style-type: none"><li>• 4 G-Bytes of address space</li><li>• 32/16 dual-entry full associative joint TLB plus 4 dual-entry ITLB and 4 dual-entry DTLB respectively</li><li>• 7 different page size from 4Kb to 16MB supported in any entry</li><li>• Support entry lock</li><li>• Space identifier ASID: 8 bits</li><li>• <b>Small (1K) page not implemented</b></li></ul>
Data Cache	<ul style="list-style-type: none"><li>• Virtually-indexed, physically-tagged</li><li>• 4 way, 8-word line, alterable size: 4K, 8K, 16K bytes</li><li>• LRU replacement algorithm</li><li>• Write-back, write-through</li><li>• 16-word depth write buffer</li><li>• <b>Cache line lock not implemented</b></li></ul>
Instruction Cache	<ul style="list-style-type: none"><li>• Physically-indexed, physically-tagged</li><li>• 4 way, 8-word line, alterable size: 4K, 8K, 16K bytes</li><li>• LRU replacement algorithm</li><li>• <b>Cache line lock not implemented</b></li></ul>
Debug&JTAG	<ul style="list-style-type: none"><li>• JTAG interface to host machine</li><li>• ACC mode to accelerate JTAG memory access</li><li>• Two instruction and one data breakpoint</li></ul>
Internal Timer	<ul style="list-style-type: none"><li>• <b>N/A</b></li></ul>
Branch Target Buffer (BTB)	<ul style="list-style-type: none"><li>• Virtally-tagged</li><li>• Up to 64 entry direct mapped</li><li>• 2-bit branch history maintained</li></ul>
Bus Interface	<ul style="list-style-type: none"><li>• compliance with AHB protocol</li></ul>

### 1.3 Block Diagram

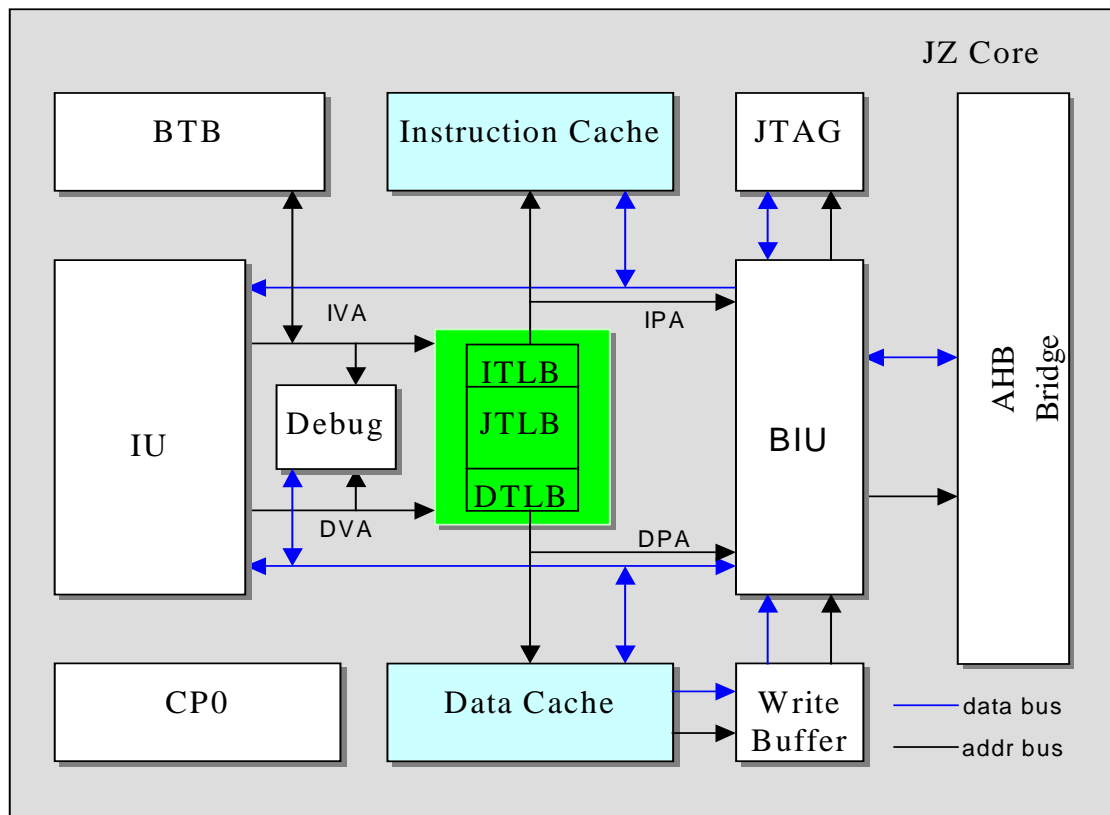
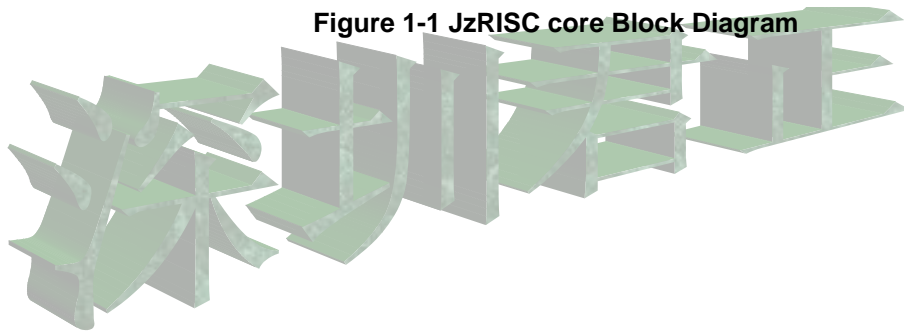


Figure 1-1 JzRISC core Block Diagram



## 2 Summary for Adjusted Contents

### 2.1 Cycles for JzRISC instruction

Most instructions in JzRISC are one cycle pass, that is, when the pipeline is fully filled, there is one instruction issued in each cycle. However, some particular instructions or execution sequences require extra execution cycles. Following table lists cycle consumption of all instructions.

Instruction	Cycles	Description
WAIT	-	WAIT instruction will be repeatedly executed until an interrupt arise
MTC0 CACHE(I) TLBW/R EXT/INS	4	3 extra interlock cycles
ROTR/ROTRV	3	2 extra interlock cycles
CACHE(D) SEB/SEH	2	1 extra interlock cycles
JALR/BCC	4/1	Zero penalty when BTB predict correctly and the branch is taken, otherwise, 3 cycles penalty
BCCL	5/4/2/1	Zero penalty when BTB taken and real taken, otherwise: 1. BTB miss, branch is taken, 3 penalties 2. BTB miss, branch is untaken, 1 penalty 3. BTB predict taken, branch is untaken, 4 cycles penalty 4. BTB predict untaken, branch is taken, 3 cycles penalty
MUL MULT/MULTU MADD/MADDU MSUB/MSUBU	2/1	No extra interlock cycle for 32x16 case 1 extra interlock cycle for 32x32 case
DIV/DIVU	2~34	Determined by characteristic of divider
Others	1	

### 2.2 Exception Priorities

Exception types	Exception Events	Exception Priorities
Reset	Reset or NMI	0 (highest)
Dfault	Data access fault	1
Dbrk	Debug data break	2
Trap/Ov	Trap or overflow	3
CpU/RI/Sys/ DBp/Brk	Co-processor unusable, reserved insn, syscall, break, sdbbp inssn	4
DSS	Debug single step	5
Ifault	Instruction Fetch Fault	6
lbrk/ Dint	Debug Insn Breakpoint or watchpoint Debug interrupt	7
INT	Hardware Interrupt or software interrupt	8 (lowest)

## 2.3 Branch Target Buffer

Dynamic branch prediction is employed in JzRISC Core to reduce branch penalties from 3 to 0. By buffering necessary target information for the branch instruction, BTB can help IU to switch the control immediately from the fall-through branch to the target branch. Taken history is also recorded in BTB to achieve highly accurate prediction.

## 2.4 CP0 Registers

Table 1-1 CP0 Registers

Register Number	Register Name	Function
0	Index	Index into the TLB array.
1	Random	Randomly generated index into the TLB array.
2	EntryLo0	Low-order portion of the TLB entry for even-numbered virtual pages.
3	EntryLo1	Low-order portion of the TLB entry for odd-numbered virtual pages.
4	Context	Pointer to page table entry in memory.
5	PageMask	Controls the variable page sizes in TLB entries.
6	Wired	Controls the number of fixed ("wired") TLB entries.
7	HWREna	N/A
8	BadVaddr	Reports the address for the most recent address related.
9	Count	N/A
10	EntrhHi	High-order portion of the TLB entry.
11	Compare	N/A
12	Status IntCtl	Processor status and control Controls expanded interrupted mode
12	SRSCtrl	N/A
12	SRSMap	N/A
13	Cause	Cause of last exception
14	EPC	Program counter at last exception
15	PRId	Processor identification and revision
15	EBase	CPUNum is set to zero
16	Config Config1 Config2 Config3 Config7	Configurations registers; config7 is added to configure BTB
17	LLAddr	Load linked address
18	WathLo	Watchpoint address (low order)
19	WatchHi	Watchpoint address (high order) and mask
20-22	Reserved	Reserved
23	Debug	Debug control and exception status
23	Trace ctl	N/A
23	Trace ctl2	N/A
23	UTD	N/A
23	TraceBPC	N/A
24	DDEPC	Program counter at last debug exception
26	ErrCtl	Controls access to data and SPRAM arrays for CACHE instruction
27	Reserved	Reserved
28	TagLo/ DataLo	Low-order portion of cache tag interface
29	Reserved	Reserverd
30	ErrorEPC	Program counter at last error
31	DESAVE	Debug hander scratchpad register

## 2.4.1 CP0 Registers Description

### 2.4.1.1 EntryLo0, EntryLo1 Register (CP0 Register 2, 3, Select 0)

Register number: 2, 3, select 0				
EntryLo0, EntryLo1 Register Format				
31	30	29	26	25
R	0	PFN		
6	5	3	2	1
C	D	V	G	0
Fields		Description	Read/Write	Reset state
Name	Bits			
PFN	25:6	Page Frame Number. PA[31:12]	R/W	Undefined
C	5:3	0~1: cacheable, noncoherent, write-through, write no allocate 2, 7: non-cacheable 3~6: cacheable, noncoherent, write-back, write allocate		
D	2	Dirty attribute of the page		
V	1	Valid attribute of the page		
G	0	Global attribute of the page		
R, 0	~	Reserved bits	0	0

### 2.4.1.2 PageMask Register (CP0 Register 5, Select 0)

Register number: 5 select 0				
PageMask Register Format				
31	25	24	13	12
0	Mask			0
Fields		Description	Read/Write	Reset state
Name	Bits			
Mask	24:13	Mask bits for varying page size 0000_0000_0000: 4KB 0000_0000_0011: 16KB 0000_0000_1111: 64KB 0000_0011_1111: 256KB 0000_1111_1111: 1MB 0011_1111_1111: 4MB 1111_1111_1111: 16MB 1K page size is not implemented	R/W	Undefined
Reserved	31:25 12:0	Assumed as 0.	0	0

#### 2.4.1.3 EntryHi Register (CP0 Register 10, Select 0)

Register number: 10 select 0				
EntryHi Register Format				
<div> <div>31</div> <div>13 12</div> <div>8 7</div> <div>0</div> </div>				
VPN2		0	ASID	
Fields		Description	Read/Write	Reset state
Name	Bits			
VPN2	31:13		R/W	Undefined
VPN2X		N/A		
ASID	7:0		R/W	Undefined
Reserved	12:8	Assumed as 0.	0	0

#### 2.4.1.4 Status Register (CP0 Register 12, Select 0)

Register Number: 12 select 0				
Status Register Format				
<div> <div>31</div> <div>28 27 26 25 24 23</div> <div>22 21 20 19 18 17 16 15</div> <div>8 7 5 4 3 2 1 0</div> </div>				
CU3-CU0	RP R RE 0	BEV TS SR NMI 0 0	IM7-IM0	R UM R ERL EXL IE
Fields		Description	Read/Write	Reset state
Name	Bits			
CU3-CU0	31-28	CU3~CU1 are not supported, always read as 0	R/W	Undefined
RP	27	Enables reduced power mode. The state of the RP bit is available on the core interface.	R/W	0
RE	25		R/W	Undefined
BEV	22		R/W	1
TS	21		R/W	0
SR	20		0	0
NMI	19		R/W	1 for NMI
IM[7:0]	15:8		R/W	Undefined
UM	4		R/W	Undefined
ERL	2		R/W	1
EXL	1		R/W	Undefined
IE	0		R/W	Undefined
Reserved	~		0	0

#### 2.4.1.5 IntCtl Register (CP0 Register 12, Select 1)

Register Number: 12 select 1				
<div> <div>31</div> <div>29 28</div> <div>26 25</div> <div>10 9</div> <div>5 4</div> <div>0</div> </div>				
IPTI		IPPCI	0	VS 0
Fields		Description	Read/Write	Reset state
Name	Bits			
IPTI	31:29	N/A	0	0
IPPCI	28:26	N/A	0	0
VS	9:5		R/W	0

#### 2.4.1.6 Cause Register (CP0 Register 13, Select 0)

Register Number: 13 select 0																								
Cause Register Format																								
31	30	29	28	27	24		23	22	21	16		15	10		9	8	7	6	5	4	3	2	1	0
BD	0	CE		0		IV	WP	0			IP[7:2]			IP[1:0]		0	Exc Code				0			
Fields					Description								Read/Write				Reset state							
Name		Bits																						
BD		31											R				Undefined							
CE		29:28											R				Undefined							
DC					N/A																			
PCI					N/A																			
IV		23											R/W				Undefined							
WP		22											R/W				Undefined							
IP[7:2]		15:10											R				Undefined							
IP[1:0]		9:8											R/W				Undefined							
Exc Code		6:2											R				Undefined							
Reserved		~											0				0							

**Table1-2 Cause Register ExcCode Field Descriptions**

Exception Code Value	Mnemonic	Description
0	Int	Interrupt
1	Mod	TLB modification exception
2	TLBL	TLB exception (load or instruction fetch)
3	TLBS	TLB exception (store)
4	AdEL	Address error exception (load or instruction fetch)
5	AdES	Address error exception (store)
6	N/A	
7	N/A	
8	Sys	Syscall exception
9	Bp	Breakpoint exception
10	RI	Reserved instruction exception
11	CpU	Coprocessor unusable exception
12	Ov	Integer overflow exception
13	Tr	Trap exception
14-22	N/A	
23	WATCH	Reference to WatchHi/WatchLo address
24	Mcheck	Mchine Check
25-31	-	Reserved

#### 2.4.1.7 Processor Identification (CP0 Register 15, Select 0)

Register number: 15 select 0						
PRId Register Format						
31	24 23		16 15		8 7	0
R		Company ID		Processor ID		Revision
Fields		Description			Read/Write	Reset state
Name	Bits					
Company ID	23:16				R	0x2
Processor ID	15:8					0x80
Revision	7:0					0x11
R	~				0	0

#### 2.4.1.8 Ebase Register (CP0 Register 15, Select 1)

Register number: 15 select 1				
31	30	29	12	11
1	0	Exception Base		00
				CPU Num
				0
Fields		Description	Read/Write	Reset state
Name	Bits			
CPU Num	9:0		R	0
Exception Base	29:12		R/W	0
R	~		0	0

#### 2.4.1.9 Config Register (CP0 register 16, Select 0)

Register number: 16 select 0				
<i>Config</i> Register Format — Select 0				
31	30	28	27	25
24	21	20	19	18
17	16	15	14	13
12	10	9	7	6
3	2	0		
0				
M	K23	KU	R	MDU
R	MM	BM	BE	AT
AR	MT	0		
		K0		
Fields		Description	R/W	Reset state
Name	Bits			
M	31		R	1
MDU	20		R	0
MM	18:17		0	0
BM	16		R	0
BE	15		R	Externally set
AT	14:13		R	0
AR	12:10	(Release II)	R	1
MT	9:7		R	3'b001
K0	2:0	0~1: Cacheable, noncoherent, write-through, no write allocate 2, 7: Uncacheable 3~6: Cacheable, noncoherent, write-back, write allocate	R/W	2
K23, KU, R, 0	~		0	0



#### 2.4.1.10 Config1 Register (CP0 Register 16, Select 1)

Register number: 16 select 1																					
31 30		25 24		22 21		19 18		16 15		13 12		10 9		7 6		5 4		3 2		1 0	
M	MMU Size			IS	IL	IA	DS	DL	DA	C2	MD	PC	WR	CA	EP	FP					
Fields		Description														Read/Write		Reset state			
Name	Bits																				
M																R		1			
MMU size	30:25	0x1F: 32 entries JTLB 0x0F: 16 entries JTLB														R		Preset			
IS	24:22	0x0: 64      0x1: 128      0x7: 32 0x2 - 0x6: Reserved														R		preset			
IL	21:19	0x0: No I-cache present 0x3: 16 bytes; 0x4: 32 bytes 0x1, 0x2, 0x5 - 0x7: Reserved														R		0x4			
IA	18:16															R		0x3			
DS	15:13	0x0: 64      0x1: 128      0x7: 32 0x2 - 0x6: Reserved														R		preset			
DL	12:10	This field contains the data cache line size. If a data cache is present, it must contain a fixed line size. 0x0: No D-cache present 0x3: 16 bytes; 0x4: 32 bytes 0x1, 0x2, 0x5 - 0x7: Reserved														R		0x4			
DA	9:7															R		0x3			
PC	4															R		0			
WR	3															R		1			
CA	2															R		0			
EP	1															R		1			
FP	0															R		0			
R, 0	~															0		0			

#### 2.4.1.11 Config3 Register (CP0 Register 16, Select 3)

Register Number: 16 select 3																							
31		30														7	6	5	4	3	2	1	0
M	0 000 0000 0000 0000 0000 0000 0													VEIC	VInt	SP	0		SM	TL			
Fields		Description												Read/Write		Reset state							
Name	Bits																						
M	31													R		0							
30~7, 3~2	30:7, 3:2													0		0							
VEIC	6													R		0							
VInt	5													R		1							
SP	4													R		0							
SM	1													R		0							
TL	0													R		0							

#### 2.4.1.12 Config7 Register (CP0 Register 16, Select 7)

Register Number: 16 select 7				
<div> <div>31</div> <div>3</div> <div>2</div> <div>1</div> <div>0</div> </div> <div> <div></div> <div>ALLOC</div> <div>BTBV</div> <div>BTBE</div> </div>				
Fields		Description	Read/Write	Reset state
Name	Bits			
Reserved	31:3	Reserved bits	0	0
ALLOC	2	Allocate hint of PREF instruction 0: enabled (default); 1: disabled	R/W	0
BTBV	1	BTB invalid. Writing 1 to this bit to invalidates BTB;	W	0
BTBE	0	BTB enable. 0: enabled (default); 1: disabled	R/W	0

#### 2.4.1.13 Taglo Register (CP0 Register 28, Select 0)

Register number: 28 select 0.				
<div> <div>31</div> <div>10</div> <div>9</div> <div>4</div> <div>3</div> <div>1</div> <div>0</div> </div> <div> <div>PA</div> <div>R</div> <div>U</div> <div>L</div> <div>V</div> </div>				
Fields		Description	Read/Write	Reset state
Name	Bits			
PA	31:10	This field contains the physical address of the cache line being stored.	R/W	Undefined
U	3:2	Dirty bits of data cache	R	Undefined
L	1	lock bit for the cache tag.	R/W	Undefined
V	0	This field indicates whether the cache line is valid.	R/W	Undefined
R	9~4	Reserved	0	0

---

## 2.5 ACC Mode For EJTAG

ACC mode is used to accelerate processor access to dmseg. In the mode, another access protocol is adopted instead of standard EJTAG one to realize fast dmseg access.

### 2.5.1 ACC Mode Flag

AM: 1 – ACC mode; 0 – MIPS mode. Reset value is 0. It is invisible for EJTAG probe and software. The bit is set by expanded instruction EJTAGBOOTA, cleared by NORMALBOOT or TAP reset.

### 2.5.2 EJTAG Control Register in ACC mode (ECR\_A)

It is connected between TDI and TDO by instruction CONTROL in ACC mode. Probe polls this 2-bit register to service the processor access to EJTAG memory.

	1	0		
	PRW	PA		
Field	BITS	Description	Read/write	Reset value
PA	0	Processor Access (PA) 0: No pending processor access 1: Pending processor access	R	0
PRW	1	Processor access is read or write 0: read access 1: write access	R	Undefined

### 2.5.3 Processor Access Address Register in ACC mode (ADDRESS\_A)

It is connected between TDI and TDO by instruction ADDRESS or ALL in ACC mode. This register is used to provide the address of the processor access to EJTAG non-drseg region.

36	35	34	33	32	31	0
BST	R	SZ	PAA			

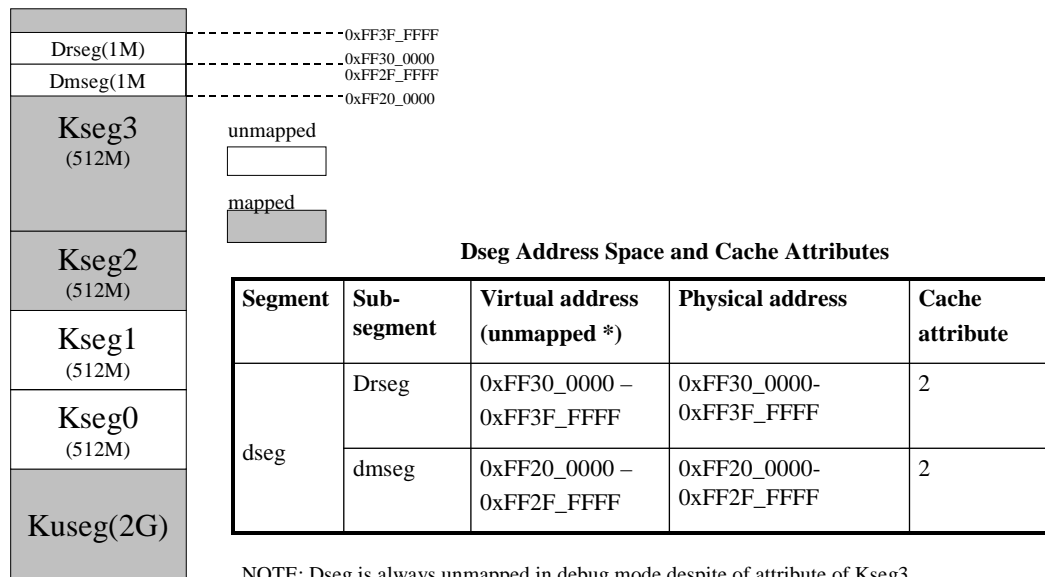
Field	BITS	Description	R/W	Reset value
PAA	31: 0	Processor Access address	R	Undefined
SZ	33:32	Processor Access size 00: byte 01: half word 10: word 11: reserved	R	Undefined
R	34	Reserved		
BST	36:35	Processor Access burst pattern 00: single 01: 4-beat wrapping burst 10: 8-beat wrapping burst 11: reserved  <i>Note: 4-beat wrapping burst will never occur in this implementation. And 8-beat wrapping burst may occur only for burst read access.</i>	R	Undefined

#### 2.5.4 Processor Access Data Register in ACC mode (DATA\_A)

It is connected between TDI and TDO by instruction DATA or ALL in ACC mode. This register is similar to PAD in normal MIPS mode, except it has one more RDY bit.

	32	31		0
	RDY	PAD		
Field	BITS	Description	R/W	Reset value
PAD	31:0	Processor Access data The register has the written value for a processor access write to the dmseg. And it is also used to provide the data value for load data or fetch instruction from the dmseg.	R/W	Undefined
RDY	1	Pipeline lock label. 1- processor can proceed due to processor access to dmseg done 0- processor should be locked due to unfinished processor access to dmseg	W	Undefined

#### 2.5.5 Address space in Debug mode (AM = 0)



Dseg space in MIPS mode

## 2.5.6 Address space in Debug mode (AM = 1)

Extended Dmseg (12M)	0xFFFF_FFFF
Drseg(1M)	0xFF40_0000-0xFF3F_FFFF
Dmseg(1M)	0xFF30_0000-0xFF2F_FFFF
Extended Dmseg (2M)	0xFF20_0000-0xFF1F_FFFF
Kseg3 (512M)	0xFF00_0000
Kseg2 (512M)	unmapped
Kseg1 (512M)	mapped
Kseg0 (512M)	
Kuseg (2G)	

**Dseg Address Space and Cache Attributes**

Segment	Sub segment	Virtual address	Physical address	Cache attribute
Dseg	extended Dmseg (2M)	0xFF00_0000-0xFF1F_FFFF	0xFF00_0000-0xFF1F_FFFF	0
	Dmseg (1M)	0xFF20_0000-0xFF2F_FFFF	0xFF20_0000-0xFF2F_FFFF	2
	Drseg (1M)	0xFF30_0000-0xFF3F_FFFF	0xFF30_0000-0xFF3F_FFFF	2
	extended Dmseg (12M)	0xFF40_0000-0xFFFF_FFFF	0xFF40_0000-0xFFFF_FFFF	0

NOTE: Dseg is always unmapped in debug mode despite of attribute of Kseg3.

## Extended Dseg space in ACC mode

## 2.5.7 Supported EJTAG Instructions

Value	Instruction	Function
0x01	IDCODE	Select Chip Identification data register
0x03	IMPCODE	Select implementation register
0x08	ADDRESS	ADDRESS register is selected in MIPS mode while ADDRESS_A register is selected in ACC mode.
0x09	DATA	DATA register is selected in MIPS mode while DATA_A register is selected in ACC mode.
0x0A	CONTROL	ECR register is selected in MIPS mode while ECR_A register is selected in ACC mode
0x0B	ALL	In MIPS mode, Selects the ADDRESS, DATA and ECR register. The scan sequence is TDI-> ADDRESS-> DATA->ECR->TDO. In ACC mode, Selects the DATA_A, ADDRESS_A, and ECR_A register. The scan sequence is TDI-> DATA_A ->ADDRESS_A ->ECR_A->TDO.
0x0C	EJTAGBOOT	Boot from probe host in MIPS mode by setting ECR.Ejtagbrk, ECR.ProbEn and ECR.ProbTrap when reset. Bypass register is selected.
0x0D	NORMALBOOT	Boot in normal way by clearing Ejtagbrk, ProbEn and ProbTrap when reset. Bypass register is selected.
0x1C	EJTAGBOOTA	Boot from probe host in ACC mode by setting ECR.Ejtagbrk, ECR.ProbEn, ECR.ProbTrap and AM when reset. Bypass register is selected.
0x1F	BYPASS	Select Bypass register

---

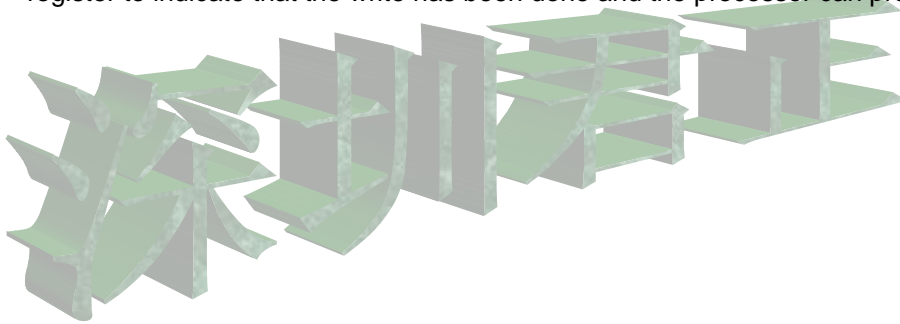
## 2.5.8 Fetch/Load and Store From/to the EJTAG Probe through dmseg in MIPS mode

### Fetch/load from EJTAG memory

1. The internal hardware latches the requested address into the Address register (ADDRESS)
2. The internal hardware sets the following bits in the EJTAG Control register:  
PrAcc = 1;  
PRnW = 0;  
Psz = Value depending on the transfer size
3. The EJTAG Probe selects the EJTAG Control register, shifts out its content and tests the PrAcc bit: when the PrAcc bit is found 1, it means that pending processor access need be serviced.
4. The EJTAG Probe checks the PRnW bit to determine the required access.
5. The EJTAG Probe selects the ADDRESS register and shifts out its content.
6. The EJTAG Probe selects the DATA register and shifts in the required instruction/data.
7. The EJTAG Probe selects the EJTAG Control register again, and shifts a PrAcc = 0 bit into this register to indicate that the instruction/data is available and the processor can proceed.

### Store to EJTAG memory

1. The internal hardware latches the requested address into the Address register (ADDRESS)
2. The internal hardware sets the following bits in the EJTAG Control register:  
PrAcc = 1;  
PRnW = 1;  
Psz = Value depending on the transfer size
3. The EJTAG Probe selects the EJTAG Control register, shifts out its content and tests the PrAcc bit: when the PrAcc bit is found 1, it means that pending processor access need be serviced.
4. The EJTAG Probe checks the PRnW bit to determine the required access.
5. The EJTAG Probe selects the ADDRESS register and shifts out its content.
6. The EJTAG Probe selects the DATA register and shifts out its content to location determined by ADDRESS.
7. The EJTAG Probe selects the EJTAG Control register again and shifts a PrAcc = 0 bit into this register to indicate that the write has been done and the processor can proceed.



---

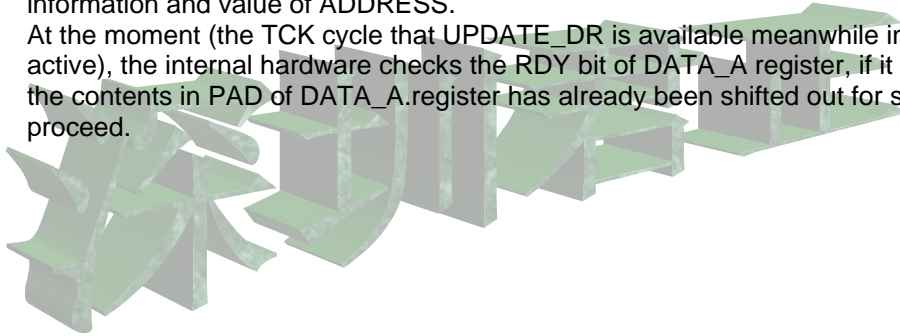
### 2.5.9 Fetch/Load and Store From/to the EJTAG Probe through dmseg in ACC mode

#### **Fetch/load from EJTAG memory**

1. The internal hardware latches the requested address into the ADDRESS\_A register
2. The internal hardware sets the following bits in the ECR\_A register:  
PA = 1;  
PRW = 0;
3. The EJTAG Probe selects the ECR\_A register, shifts out its content and tests the PA bit: when the PA bit is found 1, it means that pending processor access need be serviced.
4. The EJTAG Probe checks the PRW bit to determine the required access.
5. The EJTAG Probe selects the ADDRESS\_A register and shifts out the requested address and other access information such as burst type, access size.
6. The EJTAG Probe selects the DATA\_A register and shifts in the instruction/data according to access information and value of ADDRESS.
7. At the moment (the TCK cycle that UPDATE\_DR is available meanwhile instruction DATA\_A is active), the internal hardware checks the RDY bit of DATA\_A register, if it is found 1, which means the contents in PAD of DATA\_A. register is available for processor.

#### **Store to EJTAG memory**

1. The internal hardware latches the requested address into the ADDRESS\_A register
2. The internal hardware sets the following bits in the ECR\_A register:  
PA = 1;  
PRW = 1;
3. The EJTAG Probe selects the ECR\_A register, shifts out its content and tests the PA bit: when the PA bit is found 1, it means that pending processor access need be serviced.
4. The EJTAG Probe checks the PRW bit to determine the required access.
5. The EJTAG Probe selects the ADDRESS\_A register and shifts out the requested address and other access information such as burst type, access size.
6. The EJTAG Probe selects the DATA\_A register and shifts out its content according to access information and value of ADDRESS.
7. At the moment (the TCK cycle that UPDATE\_DR is available meanwhile instruction DATA\_A is active), the internal hardware checks the RDY bit of DATA\_A register, if it is found 1, which means the contents in PAD of DATA\_A. register has already been shifted out for store and processor can proceed.



# 1 Clock Reset and Power Controller

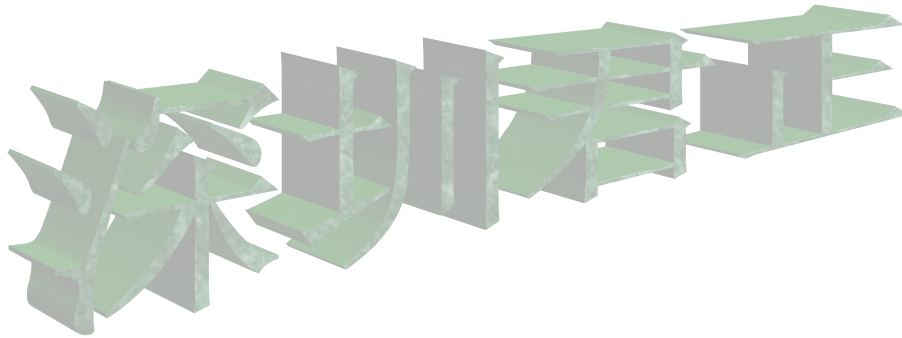
## 1.1 Overview

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

The Clock control logic in JZ4730 can generate the required clock signals including CCLK for CPU, HCLK for the AHB bus peripherals, and PCLK for the APB bus peripherals. The JZ4730 has one Phase Locked Loops (PLL): for CCLK, HCLK, and PCLK, USB block (48Mhz) ,LDCLK, 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, the JZ4730 has various power management schemes to keep optimal power consumption for a given task. The power management block in the JZ4730 can activate four modes: NORMAL mode , DOZE mode, IDLE mode, SLEEP mode, and HIBERNATE mode.

For reset control logic, the reset module controls or distributes all of the system reset signals used by the JZ47xx





## 1.2 Clock Generation Module

The JZ4730 processor contains one PLL driven by the 3.6864-MHz oscillator and a clock generator from which the following are derived:

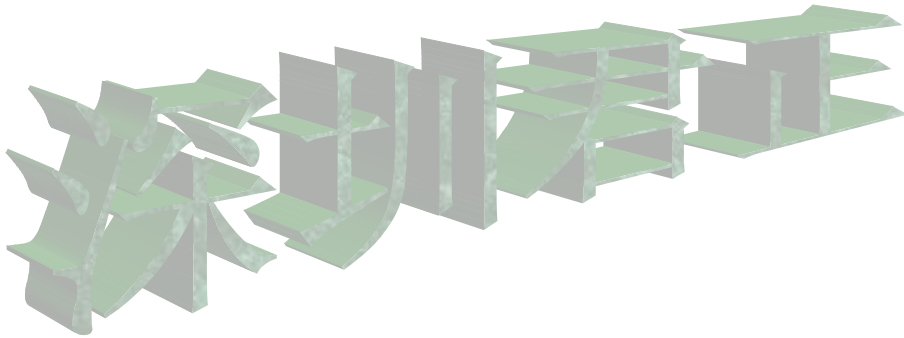
- CPU clock
- System bus clock
- Peripheral bus clock
- SDRAM bus clock
- Programmable clocks needed by certain peripherals

The output clocks are:

Signal	Description
CCLK	Fast clock used by JZ47xx 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 and Hibernate mode when all the criteria to enter a low power are met
PCLK	peripheral clock – APB BUS device clock
MCLK	Flash and Sram external memory clock
SCLKO	SDRAM Clock
LDCLK	Divided device clock output for the LCD module
LPCLK	Divided pixel clock output for the LCD module
I2SCLK	Divided clock output for AIC module
CLK48M	Divided to 48 MHz clock for the USB Module
SSICLK	Divided clock output for SSI module
MMCCLK	Divided clock output about 19M to MSC module (MMC card)
SDCLK	Divided clock output about 24M to MSC module (SD card)
3.6864M	Divided clock output for UART I2C SSI SCC and PWM

Feature:

- On-chip 3.6864MHz oscillator circuit
- One On-chip phase-locked loops (PLL) with programmable multiplier
- CCLK, PCLK, HCLK, MCLK , SCLKO and LDCLK LPCLK frequency can be changed separately for software by setting registers.
- CLK48M may output from PLL dividers.



### 1.2.1 CGM Block Diagram

Following figure illustrates a block diagram of CGM.

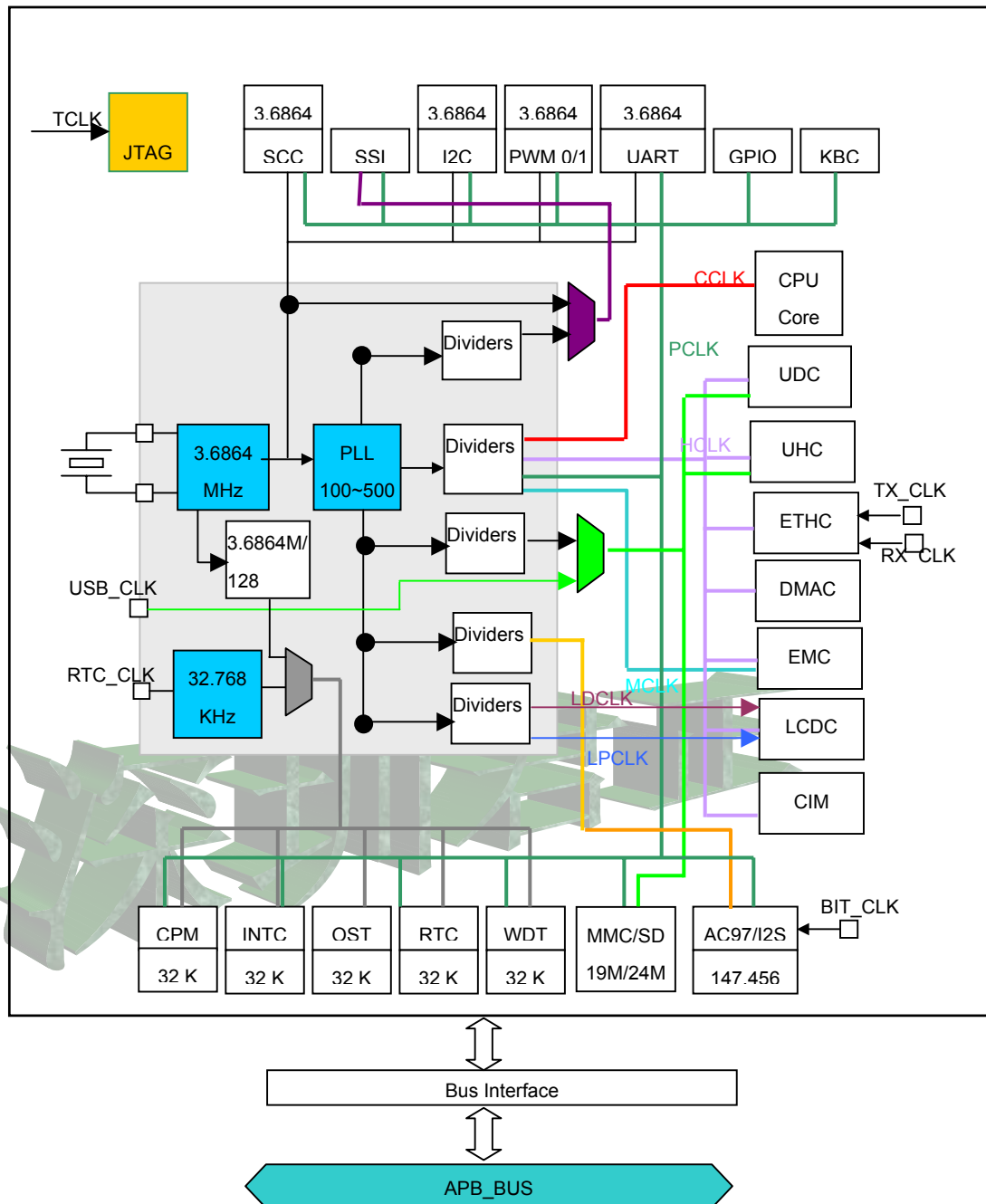


Figure 1-1 CGM Block Diagram

## 1.2.2 CGM Registers

All CGM register 32bit access address is physical address.

**Table 1-1 CGM Registers Configuration**

Name	description	RW	Reset Value	Address	Access Size
CPCCR0	Clock Control Register0	RW	0x00C00000	0x10000000	32
CPPCR	PLL Control Register	RW	0x28080011	0x10000010	32
CPOCR	Oscillator Control Register	RW	0x00150015	0x1000001C	32
CPCCR1	Clock Control Register1	RW	0x00000000	0x10000060	32

### 1.2.2.1 Clock Control Register0

The Clock Control Register0 (CPCCR0) is a 32-bit read/write register, which controls CCLK, HCLK, PCLK, MCLK and LDCLK division ratios. It is initialized to 0x00C00000 by poweron reset and WDT reset. Only word access can be used on CPCCR0.

CPCCR0																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							
	SCS		LCS		I2CS		UCS		UDIV				MCS		Reserved	SCLKOEN	Reserved	CE		MDIV				LDIV				PDIV				HDIV				CDIV			
RST	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							

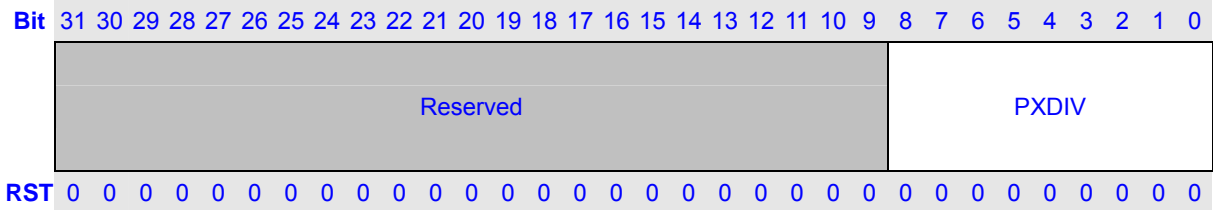
Bits	Name	Description	RW
31	SCS	SSI Clock Source Selection. Selects the SSI device clock source between 3.6864MHz input and USBCLK 0 : SSI clock source is 3.6864MHz input 1: SSI clock source is USBCLK	RW
30	LCS	LCD Pixel Clock Source Selection. Selects the LCD pixel clock source between divider and external clock input 0: LCD pixel clock source is divider output 1: LCD pixel clock source is LCD_PCLK pin	RW
29	I2CS	I2S Clock Source Selection. Selects the I2S clock source between PLL output and divider. 0: I2S clock source is PLL output 1: I2S clock source is PLL output divided by 2	RW
28	UCS	USB Clock Source Selection. Selects the USB clock source between PLL output and pin USB_CLK.  0: USB clock source is PLL output	RW

		1: USB clock source is pin USB_CLK																																																													
27:25	UDIV	Divider for USB Clock Frequency. When USB clock source is PLL (UCS bit is 0), this field specified the USB clock division ratio, which varies from 1 to 8 (division ratio = UDIV + 1).	RW																																																												
24	MCS	MSC Clock Selection 0: Support 19.1692 MHz clock to MSC 1: Support 24.576 MHz clock to MSC	RW																																																												
23	Reserved	Writes to these bits have no effect and always read as 0	R																																																												
22	SCLKOE N	SCLKO Output Enable. Controls the output of SCLKO 0: Disable SCLKO output. SCLKO is Hi-Z 1: Enable SCLKO output	RW																																																												
21	Reserved	Writes to these bits have no effect and always read as 0	R																																																												
20	CE	change enable. If CE is 1, writes on CDIV, HDIV, PDIV, MDIV, UDIV, PXDIV or LDIV will start a frequency changing sequence immediately. When CE is 0, writes on CDIV, HDIV, 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	RW																																																												
19:16	MDIV	Divider for Memory Clock Frequency. Specified the MCLK division ratio. <table><tr><th colspan="4">Bit 19~16: MDIV</th><th>Description</th></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>0</td><td>1</td><td>1</td><td>0</td><td>X1/12</td></tr><tr><td>0</td><td>1</td><td>1</td><td>1</td><td>X1/16</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>X1/24</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td><td>X1/32</td></tr><tr><td colspan="4">Other Value</td><td>Reserved</td></tr></table>	Bit 19~16: MDIV				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	0	1	1	0	X1/12	0	1	1	1	X1/16	1	0	0	0	X1/24	1	0	0	1	X1/32	Other Value				Reserved	RW
Bit 19~16: MDIV				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																																																											
0	1	1	0	X1/12																																																											
0	1	1	1	X1/16																																																											
1	0	0	0	X1/24																																																											
1	0	0	1	X1/32																																																											
Other Value				Reserved																																																											
15:12	LDIV	Divider for LCD Clock Frequency. Specified the LCLK division ratio, which varies from 1 to 16 (division ratio = LDIV + 1). The frequency of LCLK must be equal to or less than 150 MHz.	RW																																																												
11:8	PDIV	Divider for Peripheral Clock Frequency. Specified the PCLK division ratio. <table><tr><th colspan="4">Bit 17~8: PDIV</th><th>Description</th></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>	Bit 17~8: PDIV				Description	0	0	0	0	X1	0	0	0	1	X1/2	RW																																													
Bit 17~8: PDIV				Description																																																											
0	0	0	0	X1																																																											
0	0	0	1	X1/2																																																											

		<table><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>0</td><td>1</td><td>1</td><td>0</td><td>X1/12</td></tr><tr><td>0</td><td>1</td><td>1</td><td>1</td><td>X1/16</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>X1/24</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td><td>X1/32</td></tr><tr><td colspan="4">Other Value</td><td>Reserved</td></tr></table>	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	0	1	1	0	X1/12	0	1	1	1	X1/16	1	0	0	0	X1/24	1	0	0	1	X1/32	Other Value				Reserved											
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																																																											
0	1	1	0	X1/12																																																											
0	1	1	1	X1/16																																																											
1	0	0	0	X1/24																																																											
1	0	0	1	X1/32																																																											
Other Value				Reserved																																																											
7:4	HDIV	<div>Divider for System Clock Frequency. Specified the HCLK division ratio.</div> <table><tr><td colspan="4">Bit 17~8: HFR</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>0</td><td>1</td><td>1</td><td>0</td><td>X1/12</td></tr><tr><td>0</td><td>1</td><td>1</td><td>1</td><td>X1/16</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>X1/24</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td><td>X1/32</td></tr><tr><td colspan="4">Other Value</td><td>Reserved</td></tr></table>	Bit 17~8: HFR				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	0	1	1	0	X1/12	0	1	1	1	X1/16	1	0	0	0	X1/24	1	0	0	1	X1/32	Other Value				Reserved	
Bit 17~8: HFR				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																																																											
0	1	1	0	X1/12																																																											
0	1	1	1	X1/16																																																											
1	0	0	0	X1/24																																																											
1	0	0	1	X1/32																																																											
Other Value				Reserved																																																											
3:0	CDIV	<div>Divider for CPU Clock Frequency. Specifies the CCLK division ratio.</div> <table><tr><td>BIT 3~0</td><td>description</td><td>BIT 3~0</td><td>description</td></tr><tr><td>0 0 0 0 :</td><td>X1</td><td>0 0 0 1 :</td><td>X1/2</td></tr><tr><td>0 0 1 0 :</td><td>X1/3</td><td>0 0 1 1 :</td><td>X1/4</td></tr><tr><td>0 1 0 0 :</td><td>X1/6</td><td>0 1 0 1 :</td><td>X1/8</td></tr><tr><td>0 1 1 0 :</td><td>X1/12</td><td>0 1 1 1 :</td><td>X1/16</td></tr><tr><td>1 0 0 0 :</td><td>X1/24</td><td>1 0 0 1 :</td><td>X1/32</td></tr><tr><td colspan="2">Other Value</td><td colspan="2">Reserved</td></tr></table>	BIT 3~0	description	BIT 3~0	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	0 1 1 0 :	X1/12	0 1 1 1 :	X1/16	1 0 0 0 :	X1/24	1 0 0 1 :	X1/32	Other Value		Reserved																																		
BIT 3~0	description	BIT 3~0	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																																																												
0 1 1 0 :	X1/12	0 1 1 1 :	X1/16																																																												
1 0 0 0 :	X1/24	1 0 0 1 :	X1/32																																																												
Other Value		Reserved																																																													

### 1.2.2.2 Clock Control Register1

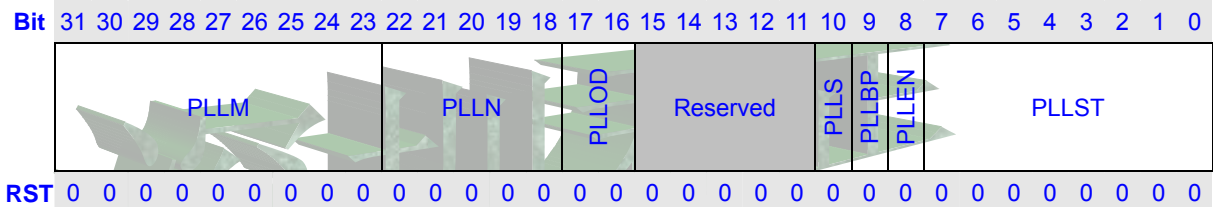
Clock Control Register1 (CPCCR1) is a 32-bit read/write register that specifies the divider of LCD pixel clock (LPCLK). This register is initialized to 0x00000000 only by poweron reset. Only word access can be used on CPCCR1

**CPCCR1****0x10000060**

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

### 1.2.2.3 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 poweron reset and WDT reset. Only word access can be used on CPPCR.

**CPPCR****0x10000010**

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 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 . 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 to 1 will bypass PLL. The PLL is still running background but the source of associated dividers is switched to	RW

		3.6864-M. If PLEN is 0, set this bit to 1 has no effect. If PLEN is 1, clear this bit to 0 will switch the source of associated dividers to PLL output. Follow table lists the PLL state and source of associated dividers	
8	PLEN	PLL Enable. When PLEN 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 PLEN is clear to 0, PLL is shut off and the source of associated dividers is switched to 3.6864-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

#### 1.2.2.4 3.6864M Oscillator Control Register

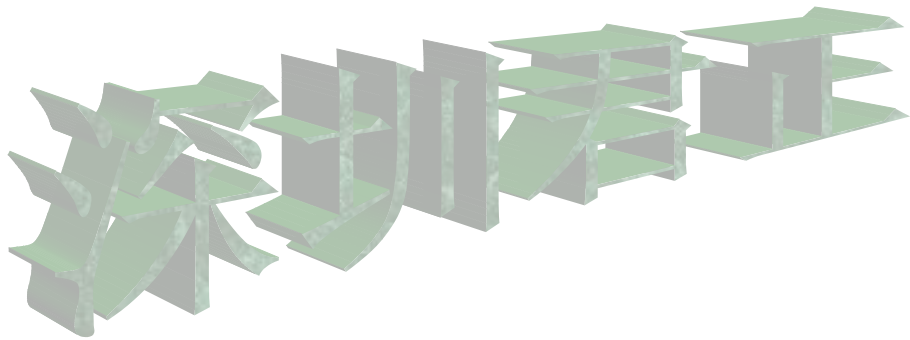
Oscillator Control Register (CPOCR) is a 32-bit read/write register that specifies stabilize time of 3.6864MHz oscillator. This register is initialized to 0x00150000 only by poweron reset. Only word access can be used on CPOCR.

CPOCR																0x1000001C																			
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								O1ST								Reserved								O2SE	SPEND1	SPEND0	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
31:24	Reserved	Writes to these bits have no effect and always read as 0	R
23:16	O1ST	3.6864MHz Oscillator Stabilize Time. This filed specifies the 3.6864Mhz oscillator stabilize time by unit of 16 RTCCLK periods (oscillator stable time $O1ST \times 16 / 32768$ ) cycles. It is initialized to H'15.	RW
15:9	Reserved	Writes to these bits have no effect and always read as 0	RW
8	O2SE	select 32.768KHz input clock or 3.6864M/128 clock. This filed controls the state of the 3.6864Mhz oscillator in sleep mode. If the 32.768KHz clock disable (O2SE bit is 0), this bit is ignored and the 3.6864MHz oscillator is always on. 0: Select 3.6864M/128 division ratio clock 1: Select 32.768Khz input clock	RW
7	SPEND1	force USB port 1 phy to enter suspend mode to reduce phy power consume 0: USB port 1 phy hasn't forced to entered SUSPEND mode 1: USB port 1 phy has forced to entered SUSPEND mode	RW



6	SPEND0	force USB port 0 phy to enter suspend mode to reduce phy power consume 0: USB port 0 phy hasn't forced to entered SUSPEND mode 1: USB port 0 phy has forced to entered SUSPEND mode	
5:0	Reserved	Writes to these bits have no effect and always read as 0	



### 1.2.3 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

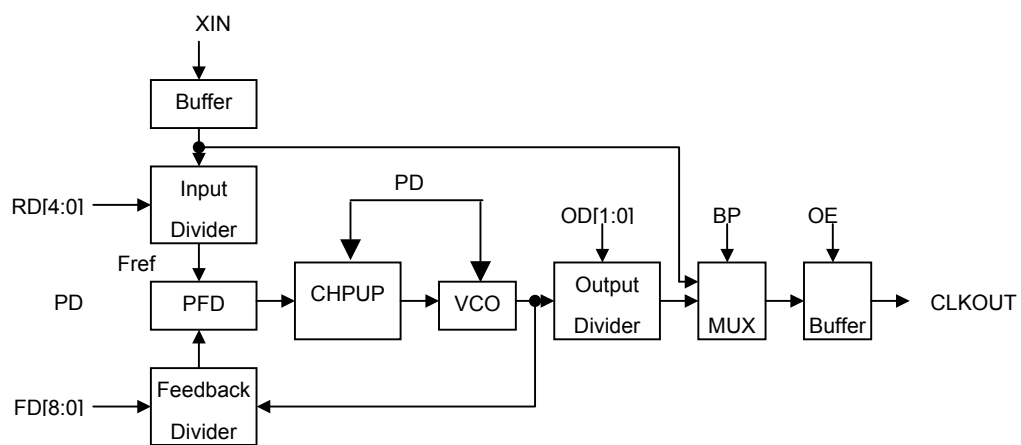


Figure 1-2 Block Diagram of PLL

#### 1.2.3.1 PLL Configuration

##### – PLL Divider Value Setting

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

##### – Input Divider Value N

$$N = \text{PLLN of CPPCR} + 2$$

##### – Feedback Divider Value M

$$M = \text{PLLM of CPPCR} + 2$$

##### – Output Divider Value NO

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

2	2
3	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:

$$\text{CLKOUT} = \text{XIN} \times (\text{M} / \text{N}) \times (1 / \text{NO})$$

$$\text{N} = \text{F0} * 1 + \text{F1} * 2 + \text{F2} * 4 + \text{F3} * 8 + \text{F4} * 16 + \text{F5} * 32 + \text{F6} * 64 + \text{F7} * 128 + \text{F8} * 256 + 2$$

$$\text{M} = \text{R0} * 1 + \text{R1} * 2 + \text{R2} * 4 + \text{R3} * 8 + \text{R4} * 16 + 2$$

$$\text{NO} = 2^{\text{od0} + \text{od1}}$$

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.  $1\text{MHz} \leq \text{XIN}/\text{N} \leq 15\text{MHz}$
2.  $100\text{MHz} \leq \text{CLK\_OUT} \times \text{NO} \leq 500\text{MHz}$

#### 1.2.4 Main Clock Division Change Sequence

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

- CCLK must be integral multiple of HCLK
- The frequency ratio of CCLK and HCLK can not be 24 and 32
- HCLK must be equal to MCLK or twice of MCLK
- HCLK and MCLK must be integral multiple of PCLK.

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

In normal mode, if CE bit of CPCCR0 is 1, changing CDIV, HDIV, PDIV or MDIV will start a Division Change Sequence immediately. If CE bit of CPCCR0 is 0, changing CDIV, HDIV, PDIV or MDIV will not start Division Change Sequence.

### 1.2.5 Change Other Clock Frequencies

The divider of LCD device clock (LDCLK), LCD pixel clock (LPCLK) and USB clock can be changed by programming LDIV, PXDIV and UDIV, respectively.

Change LDIV PXDIV and UDIV as following steps:

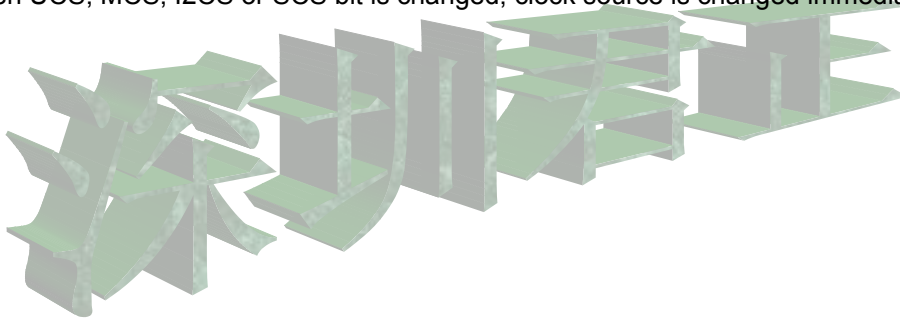
1. Stop related devices with clock-disable function. Clock supplies to the devices are stopped.
2. Change LDIV, PXDIV or UDIV. If CE is 1, clock frequencies are changed immediately. If CE is 0, clock frequencies are not changed until PLL Multiplier Change Sequence is started.
3. Cancel above clock-disable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

### 1.2.6 Change Clock Source Selection

USB, MSC, I2S and SSI device clocks can be selected from two or more sources. Before change clock source, corresponding devices should be stopped using clock-disable function.

- When USB clock source is changed (UCS bit of CPCCR0), UHC, UDC and MSC clock should be stopped. If SCS selects USB clock as source (SCS bit of CPCCR0 is 1), SSI clock should be stopped also.
- When MSC clock source is changed (MCS bit of CPCCR0), MSC should be stopped.
- When I2S clock source is changed (I2CS bit of CPCCR0), AIC should be stopped.
- When SSI clock source is changed (SCS bit of CPCCR0), SSI should be stopped.

When UCS, MCS, I2CS or SCS bit is changed, clock source is changed immediately.



## 1.3 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

### 1.3.1 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.

- HIBERNATE mode

In HIBERNATE mode, all modules except CPM are halted. PLL and 3.6864M oscillator are halt also. HIBERNATE mode is cancelled by programmable wakeup event. When HIBERNATE mode is cancelled, all modules except parts of CPM are reset.

- CLOCK DISABLE function

CLOCK DISABLE function is used to stop specified on-chip module when it is not used. Set specified CDR0~15 bits in CDR will enter specified CLK DISABLE function. CLOCK DISABLE function is canceled by reset or clearing specified CDR0~15 to 0.

### 1.3.2 Register Description

All PMC register 32bit access address is physical address.

Name	description	R/W	Initial Value	Address	Access Size
LCR	Low Power Control Register	R/W	0x001F0000	0x10000004	32
CDR	CLOCK DISABLE Register	R/W	0x00000000	0x10000020	32
HCR	Hibernate Control Register	R/W	0x00000000	0x10000024	32
HRER	Wakeup Event Rising-Edge Detect Enable Register in Hibernate mode	R/W	0x00000003	0x10000028	32
HFER	Wakeup Event Falling-Edge Detect Enable Register in Hibernate mode	R/W	0x00000003	0x1000002C	32
HER	Wakeup Event Enable Register in Hibernate mode	R/W	0x00000003	0x10000030	32
HSR	Wakeup Event Status Register in Hibernate mode	R/W	0x00000000	0x10000034	32
HGR0	GPIO Hibernate State Register 0	R/W	0x00000000	0x10000038	32
HGR1	GPIO Hibernate State Register 1	R/W	0x00000000	0x1000003C	32
HGR2	GPIO Hibernate State Register 2	R/W	0x00000000	0x10000040	32
HSPR	Hibernate Scratch Pad Register	R/W	0x00000000	0x10000044	32
HGR3	GPIO Hibernate State Register 3	R/W	0x00000000	0x10000048	32

Table 1-2 Power/Reset Management Controller Registers Configuration

#### 1.3.2.1 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 poweron and WDT reset.

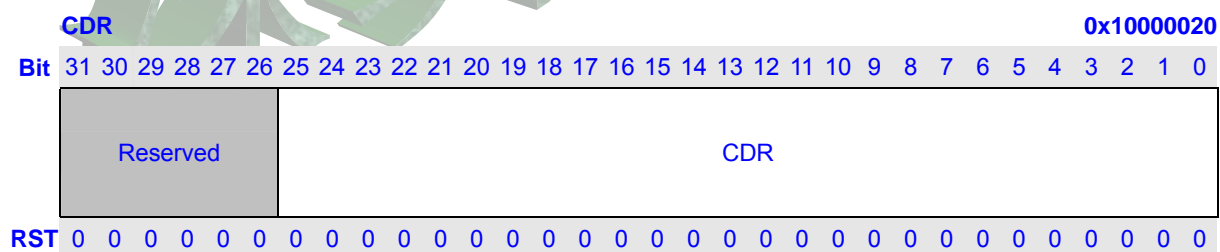
LCR																								0x10000004								
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																								DUTY				DOZE	LPM		
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Bits	Name	Description	RW
31:8	Reserved	Writes to these bits have no effect and always read as 0	R

7:3	DUTY	<p>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.</p> <p>00000 = 0/31 duty-cycle  00001 = 1/31 duty-cycle  00010 = 2/31 duty-cycle  ...  11111 = 31/31 duty-cycle</p>	RW
2	DOZE	<p>Doze Mode. Control the doze mode. When doze mode is canceled, this bit is cleared to 0 automatically</p> <p>0: Doze mode is off  1: Doze mode is on</p>	RW
1:0	LPM	<p>Low Power Mode. Specifies which low-power mode will be entered when SLEEP instruction is executed</p> <p>Bit 1~0:</p> <p><b>00</b>: IDLE mode will be entered when SLEEP instruction is executed  <b>01</b>: SLEEP mode will be entered when SLEEP instruction is executed  <b>10</b>: HIBERNATE mode will be entered when SLEEP instruction is executed  <b>11</b>: Reserved</p>	RW

### 1.3.2.2 Clock Disable Register

The Clock Disable Register (CDR) is a 32-bit read/write register that controls the CLOCK ENABLE function of peripherals. It is initialized to 0x00000000 by poweron and WDT reset.



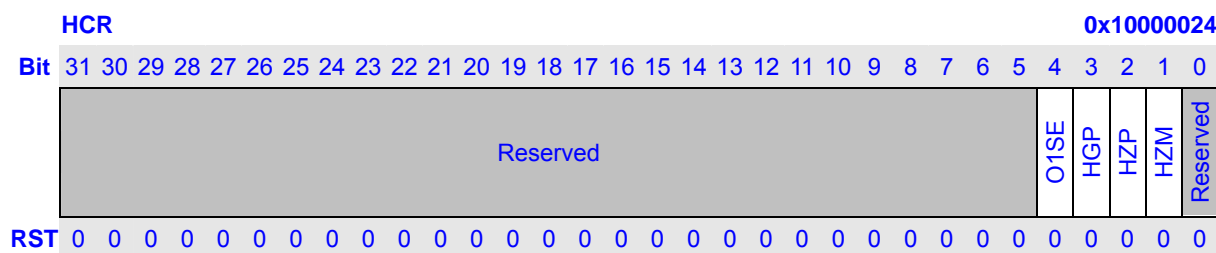
Bits	Name	Description	RW
31:26	Reserved	Writes to these bits have no effect and always read as 0	R
25:0	CDR	Clock Control Bits. Controls the clock supplies to some peripherals. If set, clock supplies to associated devices are stopped, and registers of the device cannot be accessed also	RW

Bit	Module	Description
25	UPRT	
24	UDC	
23	CIM	
22	KBC	
21	EMAC	
20	UART3	
19	<b>Reserved</b>	Only read
18	AIC	BITCLK
17	<b>Reserved</b>	Only read
16	<b>Reserved</b>	Only read
15	<b>Reserved</b>	Only read
14	SCC	
13	MSC	
12	SSI	
11	PWM1	
10	PWM0	
9	AIC	PCLK
8	I2C	
7	LCD	
6	UHC	
5	DMAC	
4	<b>Reserved</b>	Only read
3	OST	If RTCCLK or 3.6864-M is selected as count clock, the counter is still running
2	UART2	
1	UART1	
0	UART0	

### 1.3.2.3 Hibernate Control Register (HCR)

The Hibernate Control Register is a 32-bit read/write register that specifies some special controls of HIBERNATE mode. It is initialized to 0x00001500 by poweron and WDT reset.



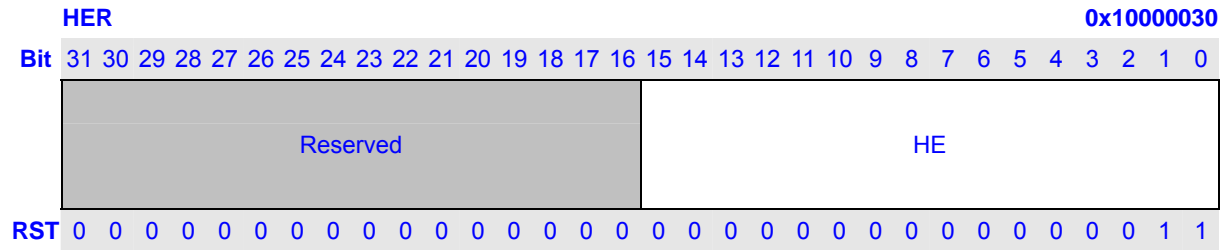


Bits	Name	Description	R/W
31:5	Reserved	Writes to these bits have no effect and always read as 0	R
4	O1SE	3.6864MHz Oscillator HIBERNATE Mode Enable. This field controls the state of the 3.6864Mhz oscillator in HIBERNATE mode. If the 32.768KHz crystal is disabled (O2SE bit is 0), this bit is ignored and the 3.6864MHz oscillator is always on 0: 3.6864MHz oscillator is disabled in HIBERNATE mode 1: 3.6864MHz oscillator is enabled in HIBERNATE mode	RW
3	HGP	Hold GPIO Control. Specifies the configuration of GPIO pins. This bit is set by poweron when HIBERNATE mode starts, must be cleared by software 0: GPIO pins are configured according to their GPIO configuration 1: GPIO pins are being held in their HIBERNATE mode state	RW
2	HZIP	Hi-Z PCMCIA Control. Specifies the state of PCMCIA pins in HIBERNATE mode 0: PCMCIA pins are not floated in HIBERNATE mode. 1: The PCMCIA pins: POE#, PIOW#, PIOR# and PCE#[2:1] are floated in HIBERNATE mode. PSKTSEL# and PREG# are derived from the address signals and assume the state of the address bus in HIBERNATE mode	RW
1	H2M	Hi-Z Memory Control. Specifies the state of static chip select pins CS1~5 in HIBERNATE mode 0: CS1~5 are driven to the state of the associated HSR register bits in HIBERNATE mode. CS0, WE and OE are driven high 1: CS0~5, WE and OE are floated in HIBERNATE mode	RW
0	Reserved	Writes to these bits have no effect and always read as 0	R

### 1.3.2.4 HIBERNATE mode Wakeup Event Enable Register (HER)

The HIBERNATE mode Wakeup Event Enable Register (HER) is a 32-bit read/write register that specifies all HIBERNATE mode wakeup sources. If a GPIO is used as a wakeup source of HIBERNATE mode, it must be configured as an input in the GPIO Controller and either rising edge or falling edge must be selected in the HRER or HFER. GPIO16~23, and GPIO96~103 can be used

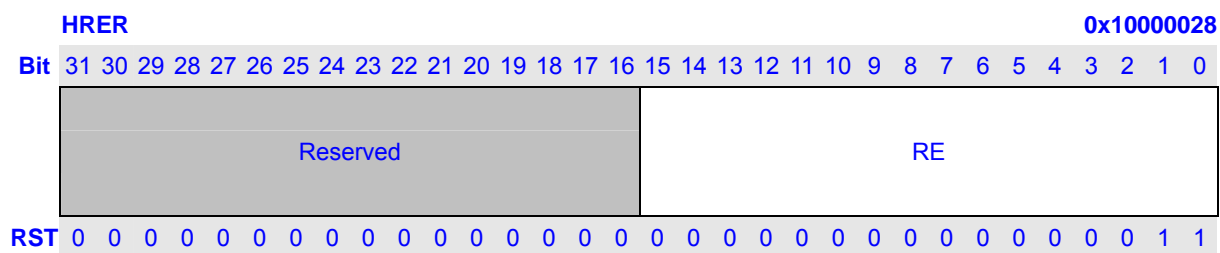
to wakeup HIBERNATE mode. The WER is initialized to 0x00000003 by poweron and WDT reset.



Bits	Name	Description	RW																																		
31:16	Reserved	Writes to these bits have no effect and always read as 0	R																																		
15:0	HE	<div>HIBERNATE mode Wakeup Enable bits</div> <div>0: Wakeup due to GPx edge detect is disabled</div> <div>1: Wakeup due to GPx edge detect is enabled</div> <table><thead><tr><th>Blt</th><th>GPIO PIN</th></tr></thead><tbody><tr><td>HE0/RE0/FE0/HS0</td><td>GPIO[96]</td></tr><tr><td>HE1/RE1/FE1/HS1</td><td>GPIO[97]</td></tr><tr><td>HE2/RE2/FE2/HS2</td><td>GPIO[98]</td></tr><tr><td>HE3/RE3/FE3/HS3</td><td>GPIO[99]</td></tr><tr><td>HE4/RE4/FE4/HS4</td><td>GPIO[100]</td></tr><tr><td>HE5/RE5/FE5/HS5</td><td>GPIO[101]</td></tr><tr><td>HE6/RE6/FE6/HS6</td><td>GPIO[102]</td></tr><tr><td>HE7/RE7/FE7/HS7</td><td>GPIO[103]</td></tr><tr><td>HE8/RE8/FE8/HS8</td><td>GPIO[16]</td></tr><tr><td>HE9/RE9/FE9/HS9</td><td>GPIO[17]</td></tr><tr><td>Reserved</td><td></td></tr><tr><td>Reserved</td><td></td></tr><tr><td>Reserved</td><td></td></tr><tr><td>HE13/RE13/FE13/HS13</td><td>GPIO[21]</td></tr><tr><td>Reserved</td><td></td></tr><tr><td>HE15/RE15/FE15/HS15</td><td>GPIO[23]</td></tr></tbody></table>	Blt	GPIO PIN	HE0/RE0/FE0/HS0	GPIO[96]	HE1/RE1/FE1/HS1	GPIO[97]	HE2/RE2/FE2/HS2	GPIO[98]	HE3/RE3/FE3/HS3	GPIO[99]	HE4/RE4/FE4/HS4	GPIO[100]	HE5/RE5/FE5/HS5	GPIO[101]	HE6/RE6/FE6/HS6	GPIO[102]	HE7/RE7/FE7/HS7	GPIO[103]	HE8/RE8/FE8/HS8	GPIO[16]	HE9/RE9/FE9/HS9	GPIO[17]	Reserved		Reserved		Reserved		HE13/RE13/FE13/HS13	GPIO[21]	Reserved		HE15/RE15/FE15/HS15	GPIO[23]	RW
Blt	GPIO PIN																																				
HE0/RE0/FE0/HS0	GPIO[96]																																				
HE1/RE1/FE1/HS1	GPIO[97]																																				
HE2/RE2/FE2/HS2	GPIO[98]																																				
HE3/RE3/FE3/HS3	GPIO[99]																																				
HE4/RE4/FE4/HS4	GPIO[100]																																				
HE5/RE5/FE5/HS5	GPIO[101]																																				
HE6/RE6/FE6/HS6	GPIO[102]																																				
HE7/RE7/FE7/HS7	GPIO[103]																																				
HE8/RE8/FE8/HS8	GPIO[16]																																				
HE9/RE9/FE9/HS9	GPIO[17]																																				
Reserved																																					
Reserved																																					
Reserved																																					
HE13/RE13/FE13/HS13	GPIO[21]																																				
Reserved																																					
HE15/RE15/FE15/HS15	GPIO[23]																																				

### 1.3.2.5 Hibernate Wakeup Event Rising-Edge Detect Enable Register (HRER)

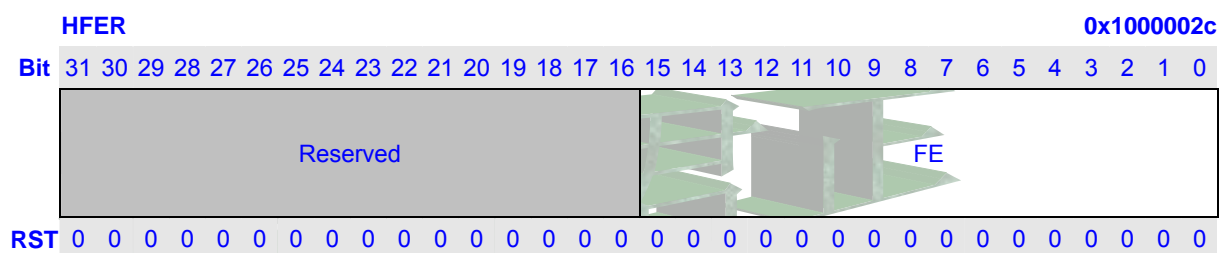
The Hibernate Wakeup Event Rising-Edge Detect Enable Register is a 32-bit read/write register that conjunctly with the Hibernate Wakeup Enable Register (HER), specifies whether a rising edge of a GPIO pin can wakeup HIBERNATE mode. The HRER is initialized to 0x00000003 by poweron and WDT reset



Bits	Name	Description	RW
31:16	Reserved	Writes to these bits have no effect and always read as 0	R
15:0	RE	HIBERNATE Wakeup Rising-Edge Enable bits 0: Wakeup due to GPx rising-edge detect is disabled 1: Wakeup due to GPx rising-edge detect is enabled	RW

### 1.3.2.6 HIBERNATE Wakeup Event Falling-Edge Detect Enable Register (HFER)

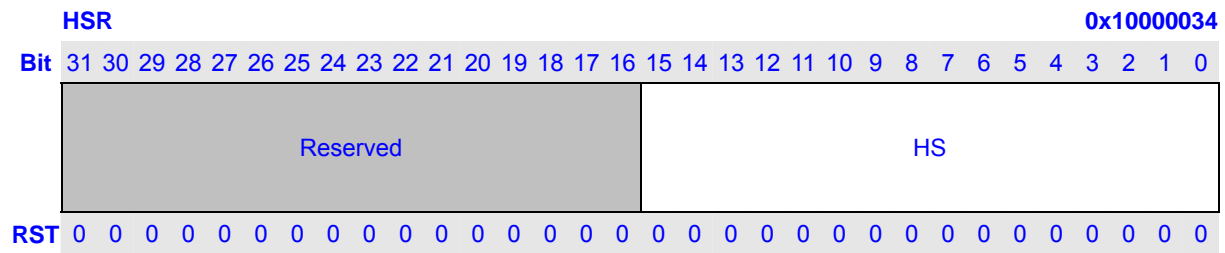
The HIBERNATE Wakeup Event Falling-Edge Detect Enable Register is a 32-bit read/write register that conjunctly with the HIBERNATE Wakeup Enable Register (HER), specifies whether a falling edge of a GPIO pin can wakeup HIBERNATE mode. The HFER is initialized to 0x00000003 by poweron and WDT reset.



Bits	Name	Description	RW
31:16	Reserved	Writes to these bits have no effect and always read as 0	R
15:0	FE	HIBERNATE Wakeup Falling-Edge Enable bits 0: Wakeup due to GPx falling-edge detect is disabled 1: Wakeup due to GPx falling-edge detect is enabled	RW

### 1.3.2.7 HIBERNATE Wakeup Event Status Register (HSR)

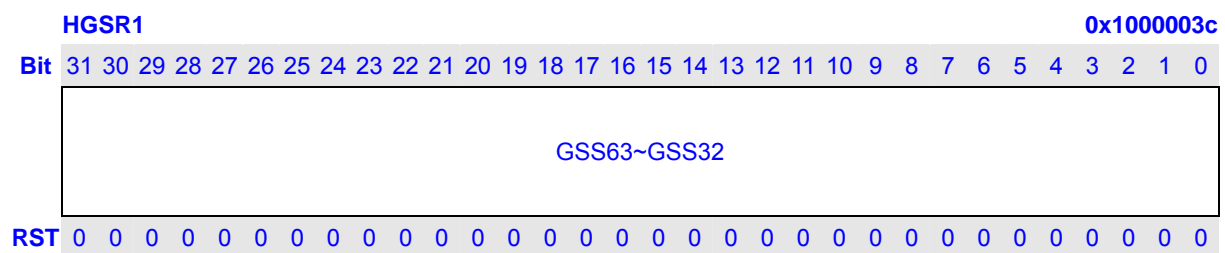
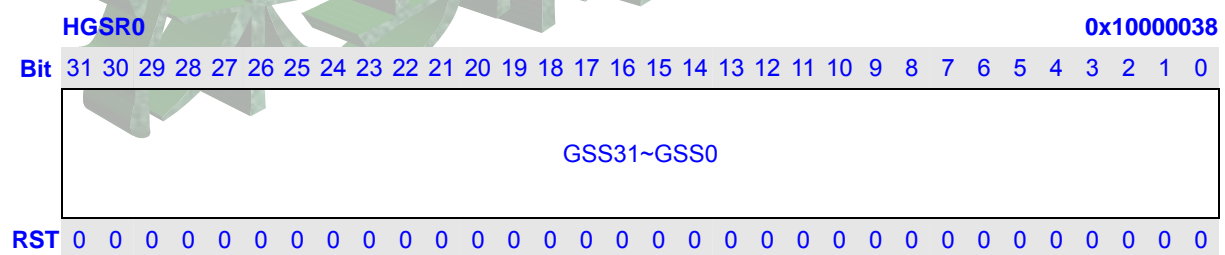
The HIBERNATE Wakeup Event Status Register (HSR) is a 32-bit read/write register that specifies which of GPIO pins wakeup HIBERNATE mode. The register is initialized to 0x00000000 by poweron and WDT reset.

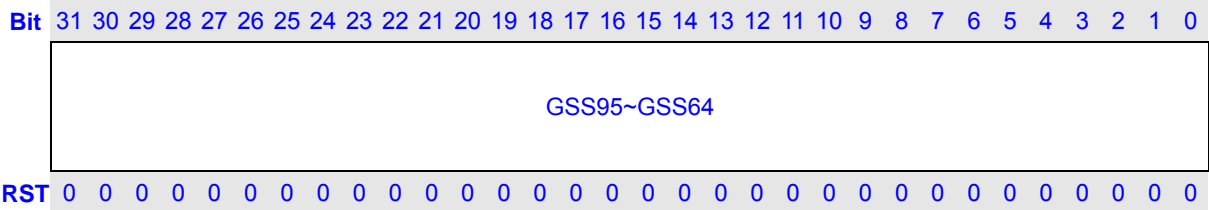
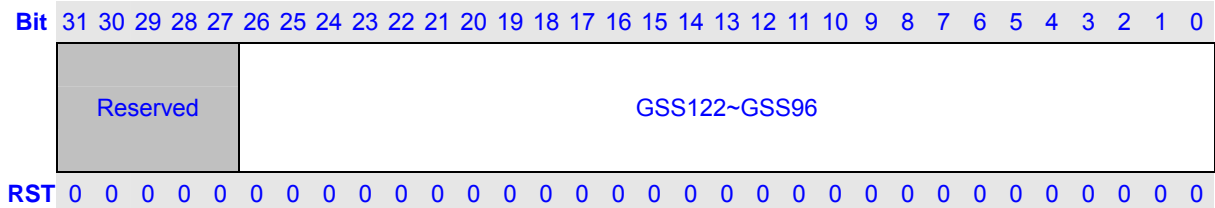


Bits	Name	Description	RW
31:16	Reserved	Writes to these bits have no effect and always read as 0.	R
15:0	HS	<p>HIBERNATE Wakeup Status of GPIO[x] pin. The bit is set when associated REx bit of HER is set and rising edge is detect and HREx bit of HRER is set, or falling edge is detect and FEx bit of HFER is set</p> <p>These bits can only be cleared by software. Set 1 to these bits has no effect</p> <p>0: HIBERNATE mode is not waken up due to edge detect on GPIO[x] pin</p> <p>1: HIBERNATE mode is waken up due to edge detect on GPIO[x] pin</p>	RW

### 1.3.2.8 GPIO HIBERNATE State Registers (HGSR0~3)

The GPIO HIBERNATE State Registers (HGSR0~3) are 32-bit read/write registers that specify the states of output GPIO pins in HIBERNATE mode. When HIBERNATE mode is entered, the contents of HGSR0~3 are loaded into the GPIO output data registers. Only pins, which are configured as output pins are affected. When HIBERNATE mode is waking up, these GPIO pins retain the HIBERNATE states until software clears the HGP bit of HCR. It is initialized to 0x00000000 by poweron and WDT reset.

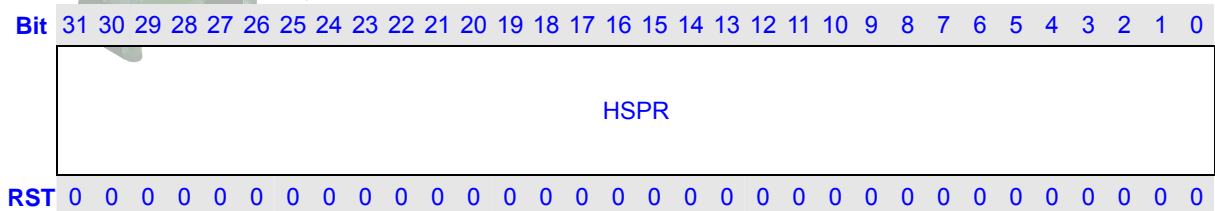


**HGSR2****0x10000040****HGSR3****0x10000048**

Bits	Name	Description	RW
31:27	Reserved	<b>HGSR3</b> : Reserved bits. Writes to these bits have no effect and always read as 0	R
122:0	GSS	HIBERNATE State of GPIO[n]. Specifies the state of GPIO[n] during HIBERNATE mode, if GPIO[n] is configured as output pin 0: GPIO[n] is driven to low during HIBERNATE mode 1: GPIO[n] is driven to high during HIBERNATE mode	RW

**1.3.2.9 Hibernate Scratch Pad Register (HSPR)**

The Scratch Pad Register is a 32-bit read/write register that allows software to preserve some critical data during HIBERNATE mode. The HSPR is hold during HIBERNATE mode. It is initialized to 0x00000000 by poweron and WDT reset

**HSPR****0x10000044**

### 1.3.3 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.

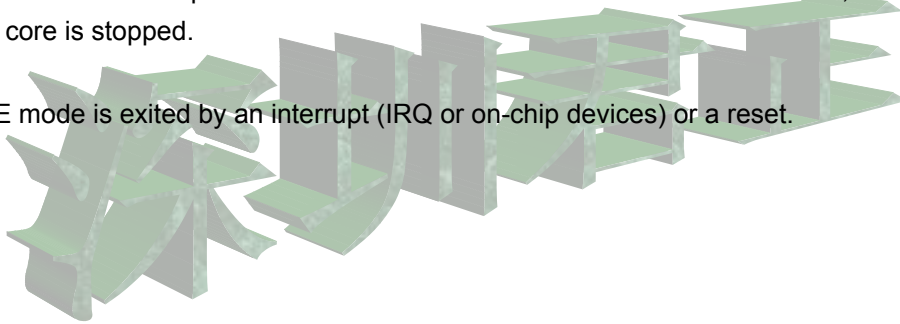
### 1.3.4 IDLE Mode

In normal or `idle` 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 below:

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.



### 1.3.5 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 CPM. PLL is shut off. Clock output from SCLKO 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.

### 1.3.6 HIBERNATE Mode

Executing SLEEP instruction while LPM bit of LCR is 2 can enter HIBERNATE mode. All GPIO output pins are switched to their predefined state. SDRAM content is preserved by driving into self-refresh state. When a wakeup event occurs, the core enters through a reset. Only CPM is operating in HIBERNATE mode.

#### 1.3.6.1 the Procedure to Enter HIBERNATE mode

Before enter HIBERNATE mode, software must complete following steps:

- Set the GPIO configuration adequate for HIBERNATE mode. Configured the GPIO HIBERNATE State registers (HGSR0~3) and Hi-Z Memory Control bit (HZM) and Hi-Z PCMCIA bit (HZP)
- Mask all interrupts
- Configure the wake-up sources properly . configured Wakeup Event Enable register (HER ) Wakeup Event Rising-Edge Detect Enable and GPIO Falling-edge Detect Enable registers (HFER HRER)
- Set USB pads as suspend mode.
- Stop LCD
- Set HIBERNATE MODE (Set LPM bits in LCR to 2.)

### 1.3.6.2 the Procedure to Wake-up from HIBERNATE mode

- The internal reset signal and RESET0 will be asserted if one of the wake-up sources is issued.
- Check HSR in order to know whether or not the power-up is caused by the wake-up from HIBERNATE mode.
- Clear Hold GPIO bit (HGP) of HCR
- check RSR to determine what caused the reset
- If the HSPR was used to preserve some data, recover the data
- Configure the SDRAM memory controller.

## 1.4 Reset Control Module

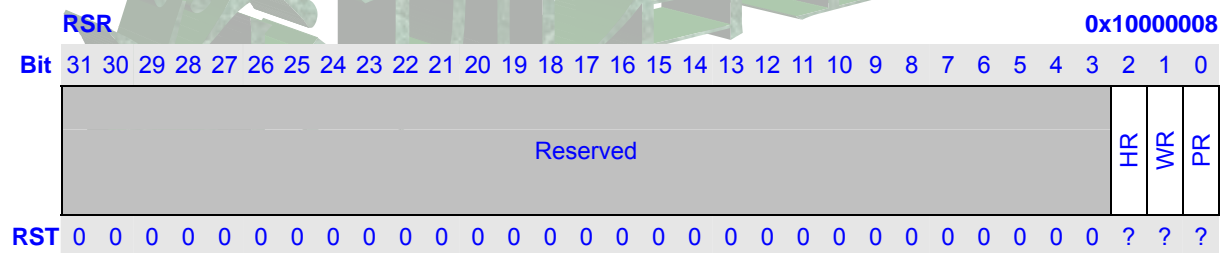
### 1.4.1 Register Description

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

#### 1.4.1.1 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
31:3	Reserved	Writes to these bits have no effect and always read as 0	R
2	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 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



1	WR	<p>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</p> <p>0: WDT reset has not occurred since the last time the software clears this bit</p> <p>1: WDT reset has occurred since the last time the software clears this bit</p>	RW
0	PR	<p>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</p> <p>0: Power on reset has not occurred since the last time the software clears this bit</p> <p>1: Power on reset has occurred since the last time the software clears this bit</p>	RW

### 1.4.2 Reset Sources and Pins

Any qualified global reset signal resets the JZ47xx and all related peripherals to their default state. After the internal reset is deasserted, the JZ47xx processor begins fetching code from the internal bootstrap ROM or CS0 space. The memory location of the fetch depends on the configuration of the BOOT pins.

Signal	Direction	Description
PRESET	IN	<b>Power On Reset</b> —An active low signal that resets the JZ47xx. This reset is propagated to the RESETO pin.
WRESET	internal	<b>Watchdog Timer Reset</b> —An active low signal generated by the watchdog timer when a time-out period has expired. Resets the same modules as HRSET. This reset is propagated to the RESETO pin.
HRESET	internal	<b>Hibernate Reset</b> —An active low signal generated by HIBERNATE mode This reset is propagated to the RESETO pin.
RESETO	OUT	An active low signal generated by JZ47xx to reset external devices

### 1.4.3 Power On Reset

Power on reset is generated when PRESET pin is driven to low. Internal reset and RESETO are asserted immediately. All pins return to their reset states.

PRESET pin must be held low until power stabilizes and the 3.6864MHz oscillator stabilize. CPU

---

and peripherals are clocked by 3.6864MHz oscillator output directly. PLL is reset to off state. All internal modules are initialized to their predefined reset states.

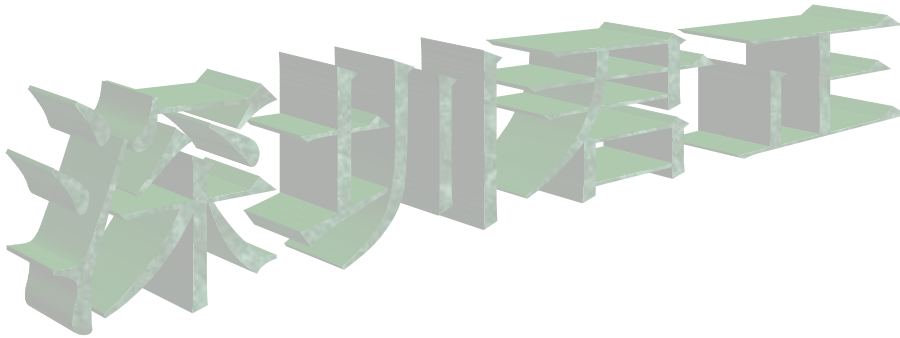
#### 1.4.4 WDT Reset

WDT reset is generated when WDT overflow. Internal reset and RESETO are 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 and RESETO are asserted for about 10 milliseconds. CPU and peripherals are clocked by 3.6864MHz oscillator output directly. PLL is reset to off state.

#### 1.4.5 Hibernate Reset

Hibernate Reset is generated by CPU has entered JZ47xx low power mode. This reset is propagated to the RESETO pin. Software should check RSR to determine which reset.



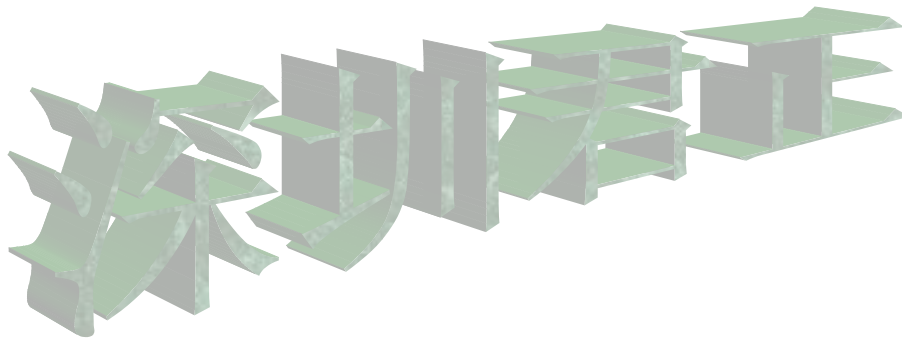
# 1 Interrupt Controller

## 1.1 Overview

This chapter describes the interrupt controller included in the JZ4730 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.



## 1.2 Register Description

Table 1-1 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 1-1 INTC Register.

All INTC register 32bit access address is physical address.

Name	Description	RW	Reset Value	Address	Access Size
ICSR	Interrupt controller Source Register	R	0x00000000	0x10001000	32
ICMR	Interrupt controller Mask Register	RW	0x00000000	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

**Table 1-1 INTC Register**

### 1.2.1 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
	Reserved	LCD	Reserved	GPIO0	GPIO1	GPIO2	GPIO3	OST0	OST1	OST2	DMAC	AIC	ETH	CIM	SSI	Reserved	RTC	MSC	UHC	UDC	SCC0	SCC1	UART0	UART1	UART2	UART3	Reserved	Reserved	UPRT	KBC	I2C	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

### 1.2.2 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
	Reserved	LCD	Reserved	GPIO0	GPIO1	GPIO2	GPIO3	OST0	OST1	OST2	DMAC	AIC	ETH	CIM	SSI	Reserved	RTC	MSC	UHC	UDC	SCC0	SCC1	UART0	UART1	UART2	UART3	Reserved	Reserved	UPRT	KBC	I2C	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

### 1.2.3 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
	Reserved	LCD	Reserved	GPIO0	GPIO1	GPIO2	GPIO3	OST0	OST1	OST2	DMAC	AIC	ETH	CIM	SSI	Reserved	RTC	MSC	UHC	UDC	SCC0	SCC1	UART0	UART1	UART2	UART3	Reserved	Reserved	UPRT	KBC	I2C	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

### 1.2.4 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
	Reserved	LCD	Reserved	GPIO0	GPIO1	GPIO2	GPIO3	OST0	OST1	OST2	DMAC	AIC	ETH	CIM	SSI	Reserved	RTC	MSC	UHC	UDC	SCC0	SCC1	UART0	UART1	UART2	UART3	Reserved	Reserved	UPRT	KBC	I2C	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

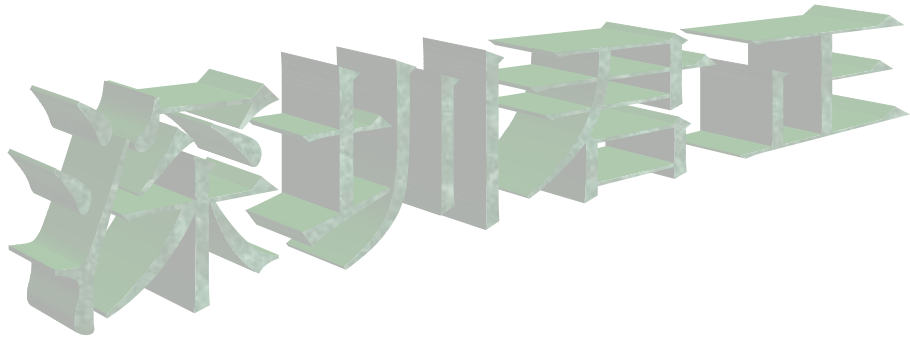
1.2.5 Interrupt Controller Pending Register (ICPR)

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
	Reserved	LCD	Reserved	GPIO0	GPIO1	GPIO2	GPIO3	OST0	OST1	OST2	DMAC	AIC	ETH	CIM	SSI	Reserved	RTC	MSC	UHC	UDC	SCC0	SCC1	UART0	UART1	UART2	UART3	Reserved	Reserved	UPRT	KBC	I2C	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.



### 1.3 Software Considerations

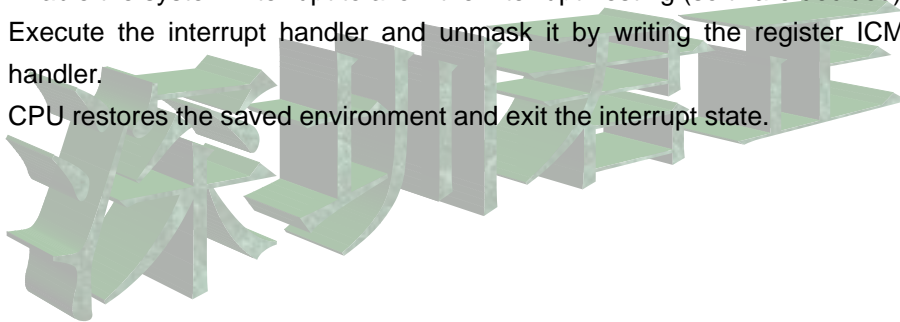
The interrupt controller is reflecting the status of interrupts sources in the peripheral .

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.



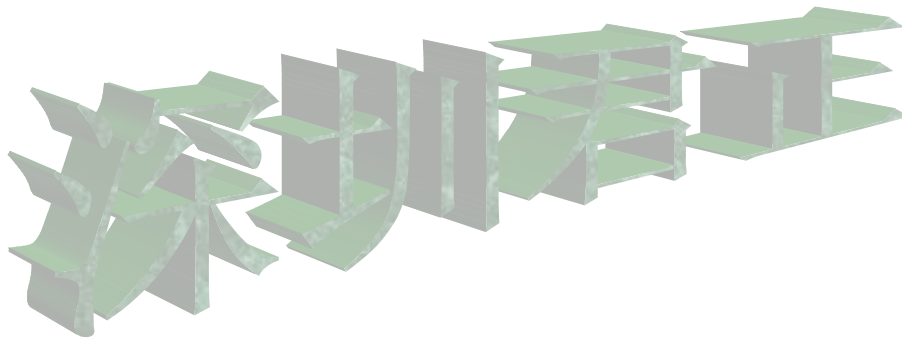
# 1 Operating System Timer

## 1.1 Overview

The operating-system timers block provides a set of timer channels that allows software to generate timed interrupts (or wake-up events).

Features:

- Three independent channels, each consisting of
  - Counter
  - Data register
  - Control register
- Independent clock for each counter, selectable by software
  - 32.768 kHz or 3.6864M/128 clock for low power
  - 3.6864-MHz clock for high accuracy
  - PCLK/4, PCLK/16, PCLK/64 and PCLK/256 for selectable by software
- One interrupt is generated for all channels when the down counter underflows
- Auto-reload function is provided for each channel.





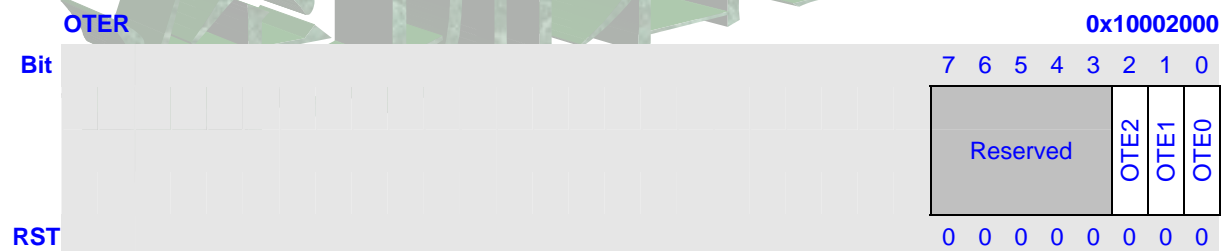
## 1.2 Register Description

All OST registers are 32bit physical address .

Name	Description	RW	Reset Value	Address	Access Size
OTER	OS Timer Enable Register	RW	0x00	0x10002000	8
OTDR0	OS Timer Data Register 0	RW	0xFFFFFFFF	0x10002010	32
OTCNT0	OS Timer Counter 0	RW	0xFFFFFFFF	0x10002014	32
OTCSR0	OS Timer Control Register 0	RW	0x0000	0x10002018	16
OTCRD0	OS Timer Counter Read Data 0	R	0x????????	0x1000201C	32
OTDR1	OS Timer Data Register 1	RW	0xFFFFFFFF	0x10002030	32
OTCNT1	OS Timer Counter 1	RW	0xFFFFFFFF	0x10002034	32
OTCSR1	OS Timer Control Register 1	RW	0x0000	0x10002038	16
OTCRD1	OS Timer Counter Read Data 1	R	0x????????	0x1000203C	32
OTDR2	OS Timer Data Register 2	RW	0xFFFFFFFF	0x10002050	32
OTCNT2	OS Timer Counter 2	RW	0xFFFFFFFF	0x10002054	32
OTCSR2	OS Timer Control Register 2	RW	0x0000	0x10002058	16
OTCRD2	OS Timer Counter Read Data 2	R	0x????????	0x1000205C	32

### 1.2.1 OS Timer Enable Register (OTER)

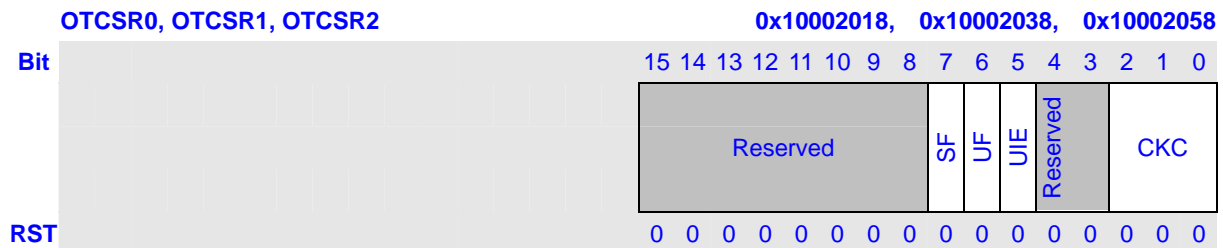
OTER is an 8-bit read/write register that selects whether start or halt the timer counters (OTCNT) for channels 0–2. OTER is initialized to 0x00 by any reset.



Bits	Name	Description	RW
7:3	Reserved	Only read	R
2	OTE2	0: Halts channel 2 count 1: Starts channel 2 count	RW
1	OTE1	0: Halts channel 1 count 1: Starts channel 1 count	RW
0	OTE0	0: Halts channel 0 count 1: Starts channel 0 count	RW

### 1.2.2 OS Timer Control Register (OTCSR)

The Timer Control Register is a 16-bit read/write register that controls clock selection and interrupt generated when the counter underflow flag has been set to 1. Each channel has a control register. The Timer Control Registers is initialized to 0x0000 by any reset

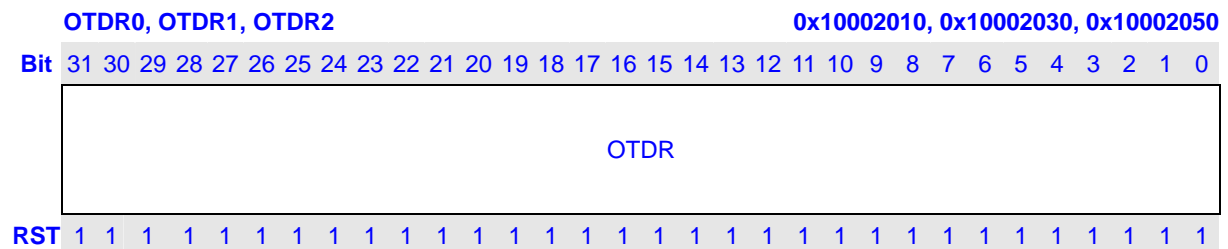


Bits	Name	Description	RW																
15:8	Reserved	These bits always read 0. Data written to these bits are ignored	R																
7	SF	Switch flag. This bit is set by hardware when the channel starts to switch clock or synchronize OTCNT read/write operation. And it is cleared by hardware after clock switching or synchronization complete	R																
6	UF	Underflow Flag. Indicates the status of OTCNT underflow. Contents do not change when 1 is written to UF 0: OTCNT underflow has not occurred. 1: OTCNT underflow has occurred.	RW																
5	UIE	Underflow Interrupt Enable. 0: Underflow interrupt is disabled 1: Underflow interrupt is enabled	RW																
4:3	Reserved	These bits always read 0. Data written to these bits are ignored	R																
2:0	CKC	OST Clock Control. These bits select the OTCNT count clock. Don't change this field when the channel is running <table border="1" style="margin-top: 5px;"> <tr> <th>Bit 2</th><th>Bit1</th><th>Bit0</th><th>Description</th></tr> <tr> <td>0</td><td>0</td><td>0</td><td>Internal clock: PCLK/4</td></tr> <tr> <td>0</td><td>0</td><td>1</td><td>Internal clock: PCLK/16</td></tr> <tr> <td>0</td><td>1</td><td>0</td><td>Internal clock: PCLK/64</td></tr> </table>	Bit 2	Bit1	Bit0	Description	0	0	0	Internal clock: PCLK/4	0	0	1	Internal clock: PCLK/16	0	1	0	Internal clock: PCLK/64	RW
Bit 2	Bit1	Bit0	Description																
0	0	0	Internal clock: PCLK/4																
0	0	1	Internal clock: PCLK/16																
0	1	0	Internal clock: PCLK/64																

		0	1	1	Internal clock: PCLK/256	
		1	0	0	RTCCLK	
		1	0	1	3.6864M	
		1	1	0	Reserved	
		1	1	1	Reserved	

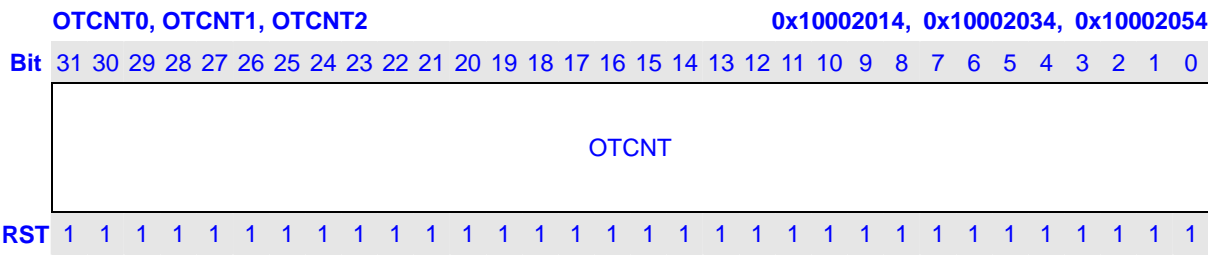
### 1.2.3 OS Timer Data Register (OTDR)

OTDR is a 32-bit read/write register used to restore the data for reload function. The counter (OTCNT) decreases from its initial value, and when results in an underflow, the value of OTDR is set in OTCNT and continues decreasing from that value, this is called reload function. Each channel has a Timer Data Register (OTDR). They have the same configuration. OTDR is initialized to 0xFFFFFFFF by any reset.



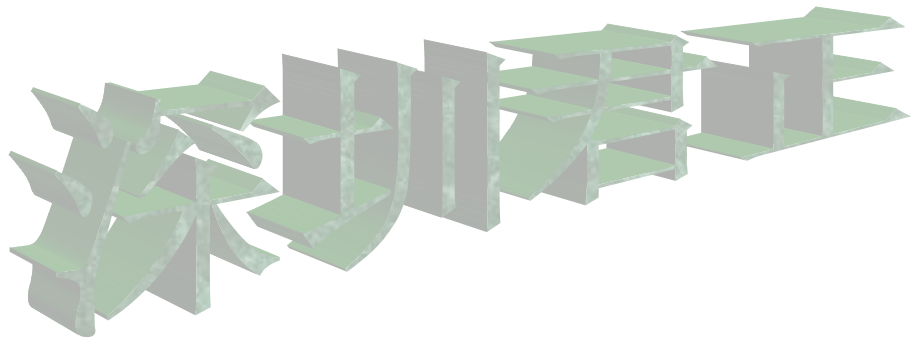
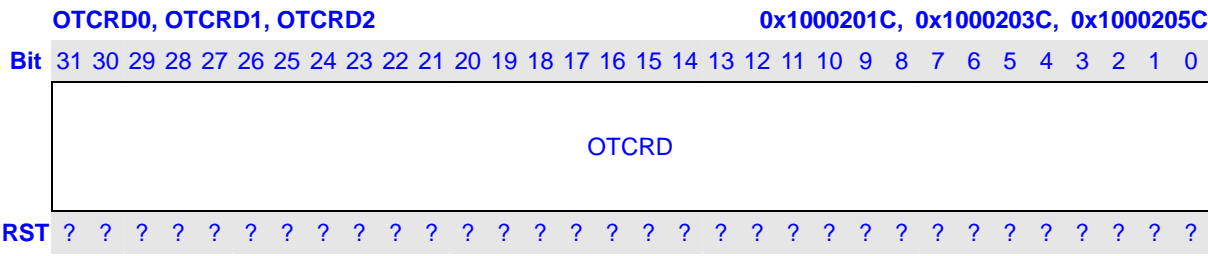
### 1.2.4 OS Timer Counter (OTCNT)

OTCNT is a 32-bit read/write register. When an OTCNT results in an underflow (0x00000000 → 0xFFFFFFFF), the underflow flag (UF) in the Timer Control Register (OTCSR) of the relevant channel set. The OTDR value is simultaneously set in OTCNT, and the countdown of OTCNT continues from that value. Each channel has a Timer Counter (OTCNT). They have the same configuration. OTCNT is initialized to 0xFFFFFFFF by any reset.



1.2.5 OS Timer Counter Read Data Register(OTCRD)

OS Timer Counter Read Data is 32-bit read-only register that used to latch OTCNT read data. When RTCCLK or 3.6864M is selected as count clock, reading to OTCNT may return an incorrect value because of asynchronous clock domain. But reading to OTCNT will start an internal read sequence and SF bit is set automatically. After correct value is locked into OTCRD, SF bit is cleared automatically. Software should monitor the SF bit and then read out correct data from OTCRD



## 1.3 Operation

### 1.3.1 Basic Functions

Counter Operation: set the enable bit in the OS Timer Enable Register (OTER) to start the corresponding OS timer counter (OTCNT) decreasing from its current value. When a OTCNT underflows ( $0x00000000 \rightarrow 0xFFFFFFFF$ ), its corresponding UF flag of the OS Timer Control Register (OTCSR) set, and if the UIE bit in OTCSR is 1, an interrupt is sent to the INTC. At this time, OTCNT reload the value OTDR as its new initial value, and continues down counting from the value.

The count operation is set as follows:

1. clear OTE in OTER .
2. Select the CKC2–CKC0 bits in the Timer Control Register (OTCSR).
3. the UIE bit in OTCSR decided to set whether to generate an interrupt when OTCNT underflows.
4. write a value OTDR ,and set the initial value OTCNT.
5. Wait for SF bit to be cleared.
6. Set the OTE in OTER to 1 to start operation.

### 1.3.2 OTCNT Reading Sequence

If PCLK/4, PCLK/16, PCLK/64 or PCLK/256 is selected as count clock, OTCNT can be read directly and correct value is always returned. If RTCCLK or 3.6864M is selected, incorrect value may be returned because asynchronous clock domain. Software should read OTCNT firstly, then monitor the SF bit of OTCSR until it is cleared, finally read correct value from OTCRD.

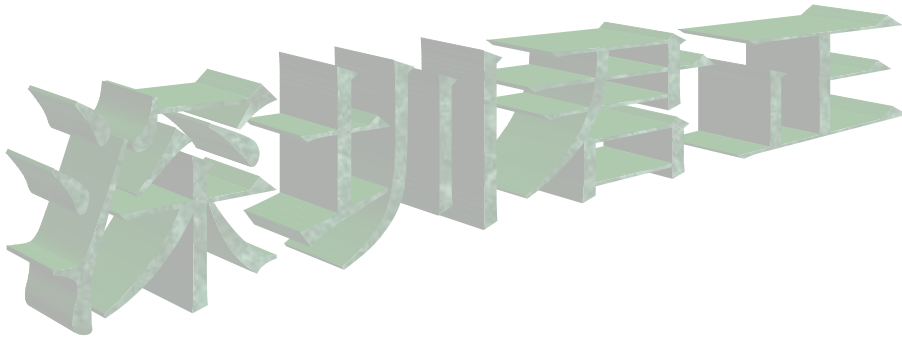
# 1 Real-Time Clock (RTC)

## 1.1 Overview

The Real-Time Clock (RTC) unit can be operated only when the system power is on, the backup battery now isn't supported. The RTC unit works with an external 32.768 kHz input clock and also can perform the alarm function.

Features:

- 32-bits second counter
- Programmable divider and regulator to generate accurate 1 Hz clock
- The 1Hz clock is generated by dividing two sources: one is 32.768KHz oscillator, another is 3.6864MHz oscillator further divided by 112 to approximately 32.914KHz.
- Alarm interrupt, 1Hz interrupt
- Run in Hibernate mode



## 1.2 Register Description

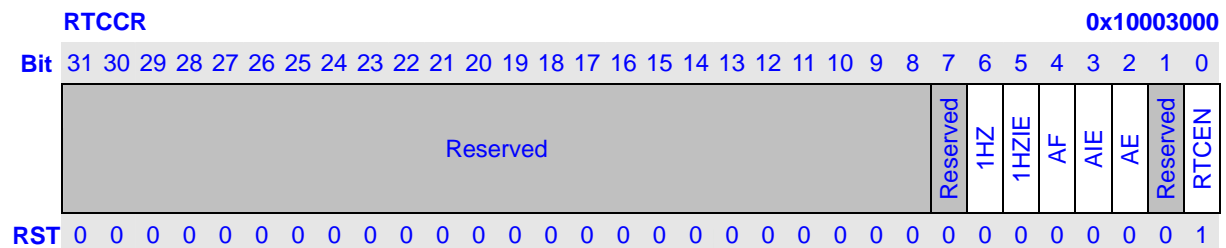
All RTC register 32bit access address is physical address.

**Table 1-1 RTC Registers**

Name	Description	RW	Reset Value	Address	Access Size
RTCCR	RTC Control Register	RW	0x00000001	0x10003000	32
RTCSR	RTC Second Register	RW	0x????????	0x10003004	32
RTCSAR	RTC Second Alarm Register	RW	0x????????	0x1000300C	32
RTCGR	RTC Regulator Register	RW	0x????????	0x10003018	32

### 1.2.1 RTC Control Register (RTCCR)

RTCCR is a 32-bit RW register that effects RTC count start/stop and interrupt. RTCCR is initialized to 0x00000001 by power on and WDT reset.



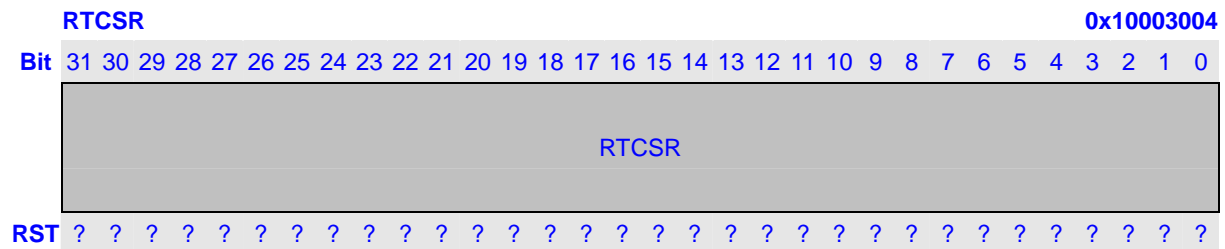
Bits	Name	Description	RW
31:7	Reserved	Only read	R
6	1HZ	1Hz Flag. Only 0 can be written to this bit. 0: 1Hz flag has not been generated. 1: 1Hz flag has been generated	RW
5	1HZIE	1Hz Interrupt Enable. 0: 1Hz interrupt is disabled 1: 1Hz interrupt is enabled	RW
4	AF	Alarm Flag. Only 0 can be written to this bit. 0: RTCSAR value doesn't matched RTCSR value 1: RTCSAR == RTCSR RTCSAR value matches the RTCSR value	RW
3	AIE	Alarm Interrupt Enable. 0: An alarm interrupt disabled 1: An alarm interrupt enabled	RW
2	AE	Alarm Enable. 0: Alarm function is disabled. 1: Alarm function is enabled.	RW

1	Reserved	Only Read	R
0	RTCEN	Enable Bit. Enable the second counter 0: Second counter disabled. 1: Second counter works normally.	RW

### 1.2.2 RTC Second Register (RTCSR)

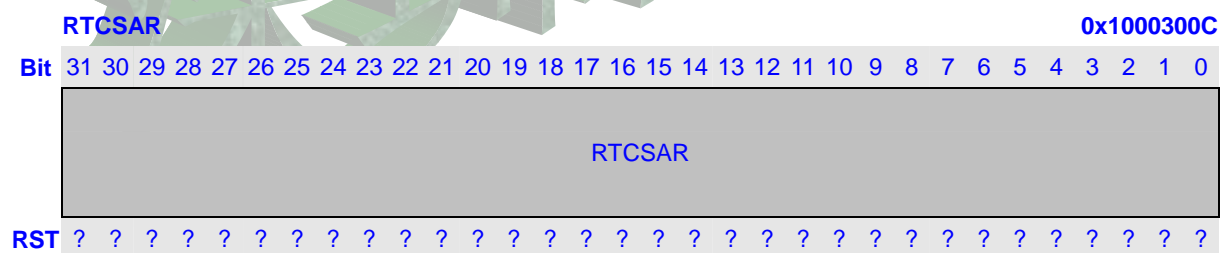
The RTC seconds register (RTCSR) is a 32-bit RW register that counts the number of elapsed seconds. Because 1Hz clock is asynchronous to system clock, writes to RTCSR is delayed actually by approximately two 32KHz clock cycles. Software may read RTCSR to ensure that desired value is written into RTCSR actually.

RTCSR is not initialized by any reset. It still counts in Hibernate mode, if RTCEN bit is set.



### 1.2.3 RTC Second Alarm Register (RTCSAR)

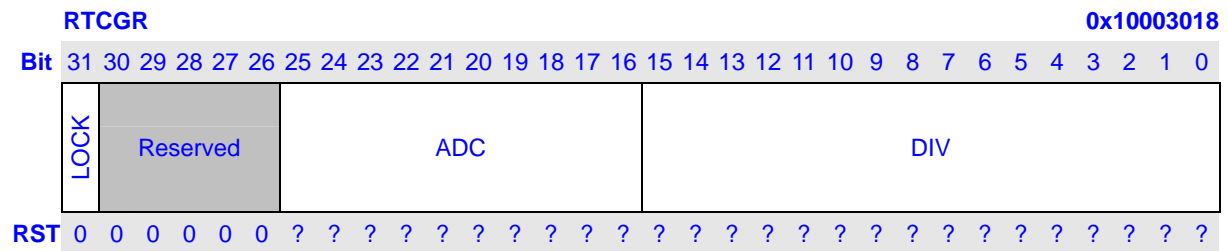
The RTC second alarm register (RTCSAR) is a 32-bit RW register that to be compared to the RTCSR value. The AE bit of RTCCR must be cleared before change the value of RTCSAR. This register is not initialized by any reset.





### 1.2.4 RTC Regulator Register (RTCGR)

The RTC Regulator register (RTCGR) is a 32-bit RW register that is used to adjust the frequency of 1Hz clock. To safeguard the validity of the data written into RTCGR, the bit 31 of RTCGR is used as Lock bit. Once Lock bit is set, write to RTCGR is ignored and the bit can only be cleared by any reset.



Bits	Name	Description	RW
31	LOCK	Lock flag. This bit can only be set by software and cleared by hardware and WDT reset. 0: write to RTCGR is allowed 1: write to RTCGR is forbidden	RW
30:26	Reserved	Only Read	RW
25:16	ADC	Adjust cycles. Specifies the number of 32KHz clock cycles to be masked every 1024 1Hz clock cycles.	RW
15:0	DIV	1Hz Divider Value. 1Hz clock is generated by dividing the adjusted 32KHz clock by the value of DIV plus 1.	RW

### 1.3 Time Regulation

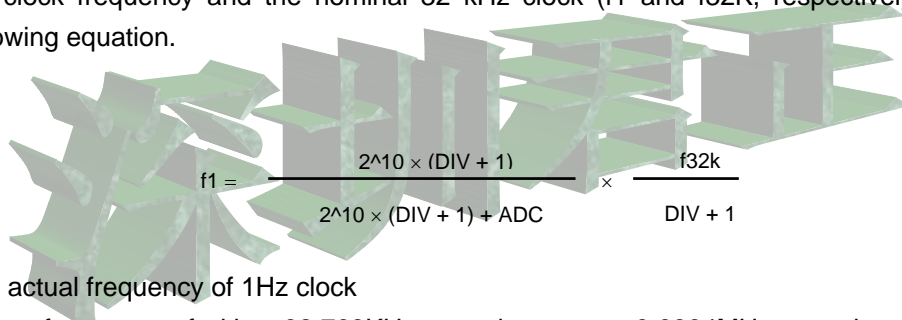
Because of the inherent inaccuracy of crystal and other variables, the time counter may be inaccurate. This requires a slight adjustment. The application processor, through the RTCGR, lets you adjust the 1HZ time base to an error of less than 1ppm. Such that if the HZ clock were set to be 1Hz, there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

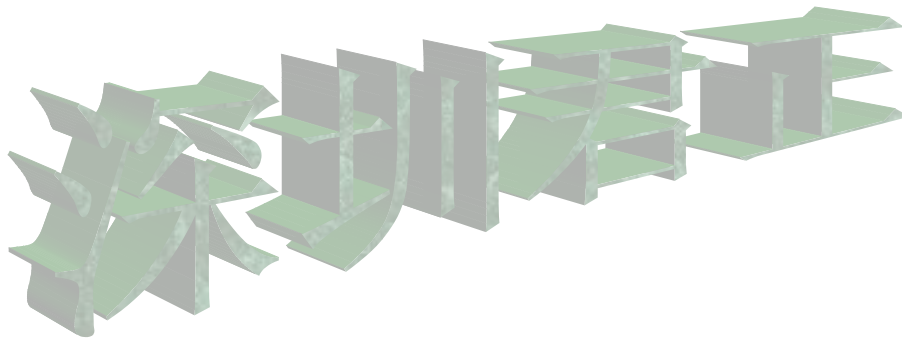
# 1 Watchdog Timer

## 1.1 Overview

The JZ47xx watchdog timer is used to resume the controller operation whenever it is disturbed by malfunctions such as noise and system errors. It can be used as a normal 32-bit interval timer to request interrupt service. The watchdog timer generates the reset signal.

Features:

- Generates wdt reset.
- 32-bit counter
- Counter clock uses the RTC clock (32.768 KHz or 3.6864M/128)



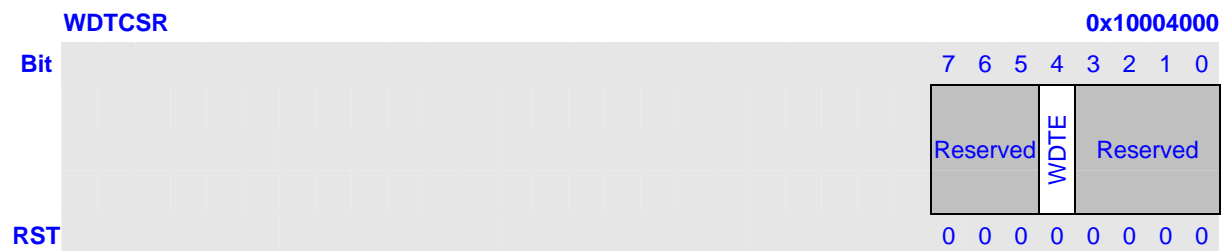
## 1.2 Register Description

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
WDTCNT	Watchdog Timer Counter	RW	0x00000000	0x10004004	32
WDTCSR	Watchdog Timer Control Register	RW	0x00	0x10004000	8

### 1.2.1 Watchdog Timer Control Register (WDTCSR)

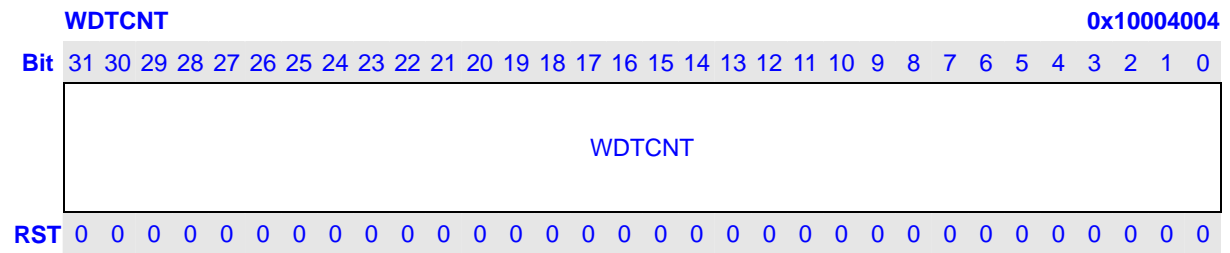
The WDTCSR is an 8-bit read/write register. It is initialized to 0x00 by power-on reset. The WDTCSR is also initialized by reset signal generated by WDT.



Bits	Name	Description	RW
7:5	Reserved	These bits always read 0, and written are ignored.	R
4	WDTE	Timer begin. To start the timer counting. Once it is written 1, can only be cleared by reset. 0: Timer stop. Ignored when write 0 1: Timer running	RW
3:0	Reserved	These bits always read 0, and written are ignored.	R

### 1.2.2 Watchdog Timer Counter (WDTCNT)

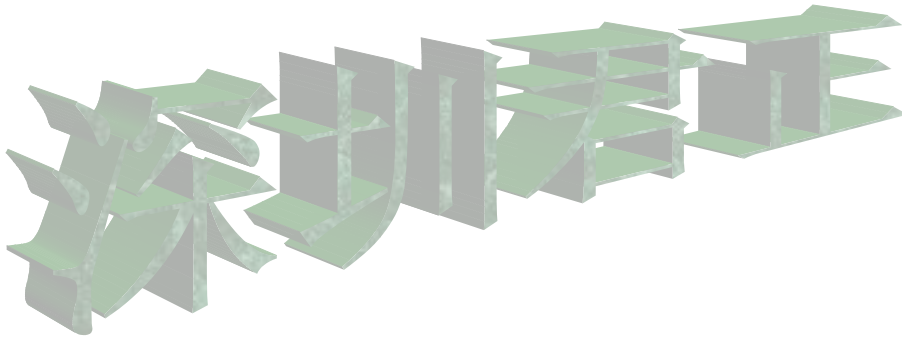
The watchdog timer counter (WDTCNT) is an 32-bit read/write counter working on the RTC clock. When counter overflow, a WDT reset is generated. The WTCNT is initialized to 0x00000000 by a power-on reset.



### 1.3 Watchdog Timer Function

The following describes steps of using WDT:

1. Set WDE bit in WTCSR to 1. The counter begins to count. Note, Once it is written 1, can only be cleared by reset.
2. Software should clear the WTCNT to 0x00000000 periodically so that the counter cannot overflow.
3. If counter overflows, a WDT reset will be generated.



# 1 External Memory Controller

## 1.1 Overview

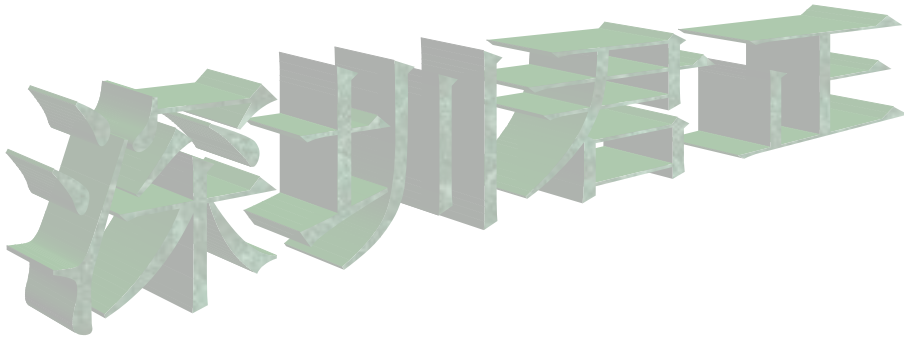
**EMC (External Memory Controller)** supports 4 different kinds of memories: Static memory, NAND flash, SDRAM and PC card.

Features:

- Static memory interface
  - Direct interface to ROM, Burst ROM, SRAM and NOR Flash
  - Six banks of static memory are supported from bank 0 to 5. Each bank can be configured separately
  - When boot from CS0, data width of static bank 0 is set by external pin. When boot from NAND flash, data width of static bank 3 is set by external pin. Data width of other banks may be 8, 16 or 32 bits
  - 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 a 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 CS3, sharing with static memory bank 3.
  - 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 address cycles are supported.
  - Hardware ECC generation
  - Support read/erase/program NAND flash memory
  - System can be configured to boot from NAND flash.
- Synchronous DRAM Interface
  - 2 banks with changeable size and base address are supported
  - Both 32- and 16-bit bus width is supported
  - Multiplexes row/column addresses according to SDRAM capacity
  - Two-bank or four-bank SDRAM is supported
  - Supports burst operation
  - Has both auto-refresh and self-refresh functions
  - 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

- PC Card Interface
- The HOST adapter interface fully compliant with the release of March 1997 of PC Card standard (16-bit PC Card).
- Two PC Card sockets, requiring only external buffering and some logic.
  - Each socket independently supports DMA function.

To support dynamic I/O bus sizing



## 1.2 Block Diagram

Following figure shows the functional block diagram of the EMC.

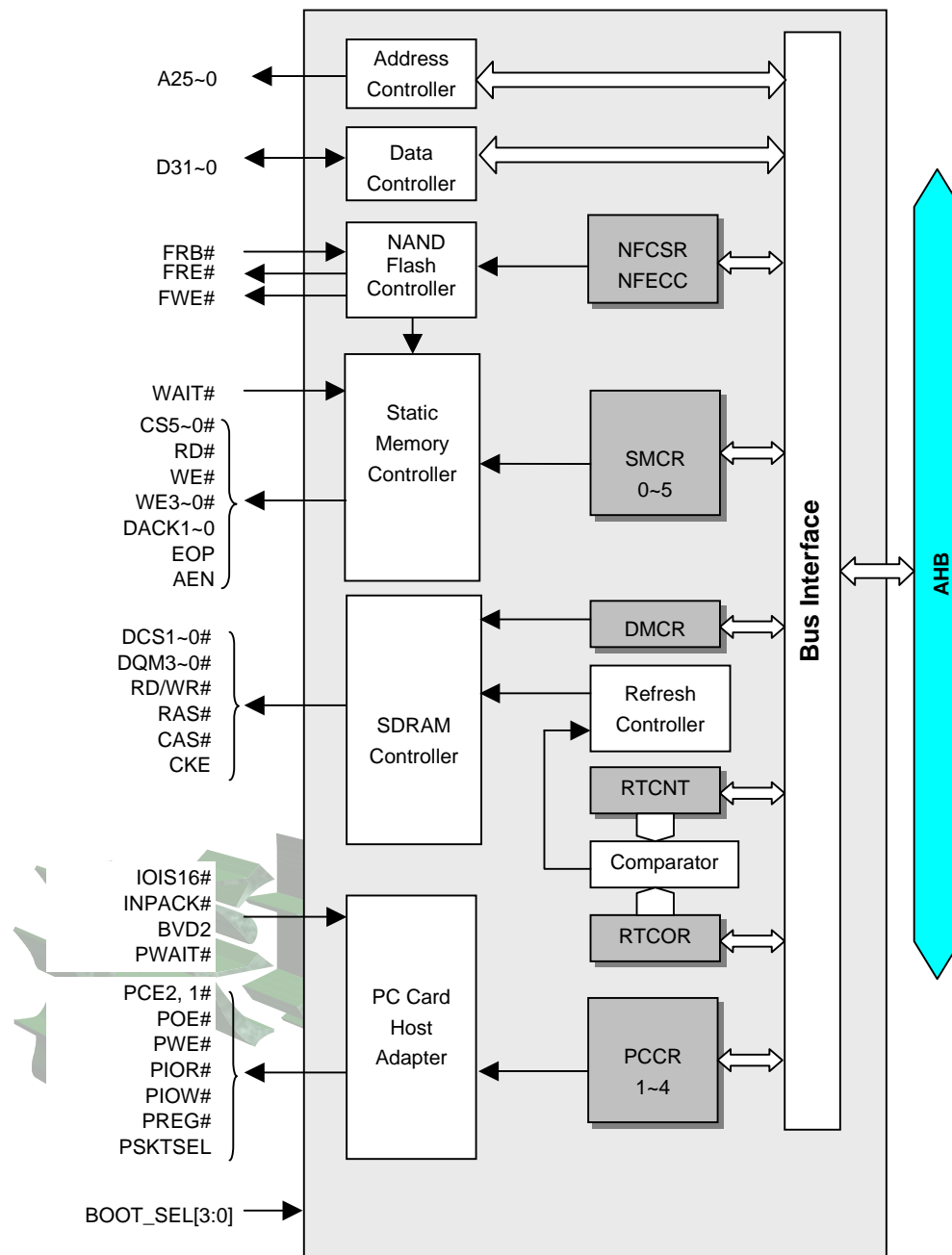


Figure 1-1 EMC Block Diagram



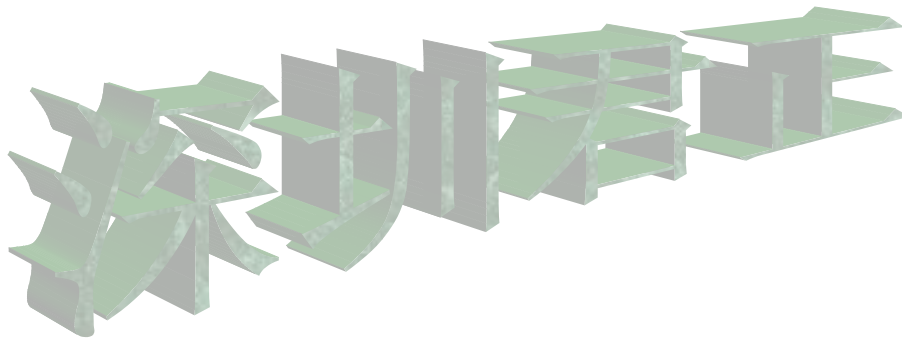
### 1.3 Pin Description

Following table list the EMC pins.

**Table 1-1 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 5 ~ 0	O	CS5~0#	Chip select signal that indicates the static bank being accessed (Specially, CS3 can act as NAND flash CS).
SDRAM chip select 1 ~ 0	O	DCS1~0#	Chip select signal that indicates the SDRAM bank being accessed
Read enable	O	RD# / POE#	Static memory read enable signal Also used as PC card memory and attribute space read strobe signal
Write enable	O	WE# / PWE#	Static memory write enable signal Also used as PC card memory and attribute space write strobe 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 / PIOW#	When non-byte-control static memory is used, D7-0 write enable signal When byte-control static memory is used, selection signal for D7-0 When SDRAM is used, selection signal for D7–D0 When PC card is used, PC card I/O space write strobe
Byte enable 1	O	WE1# / BE1# / DQM1 / PIOR#	When non-byte-control static memory is used, D15-8 write enable signal When byte-control static memory is used, selection signal for D15-8 When SDRAM is used, selection signal for D15–D8 When PC is used, PC card I/O space read strobe
Byte enable 2	O	WE2# / BE2# / DQM2 / PREG#	When static memory is used, D23-16 write enable signal When byte-control static memory is used, selection signal for D23-16 When SDRAM is used, selection signal for D23–D16 When PC card is used, indicates a attribute space access
Byte enable 3	O	WE3# / BE3# / DQM3	When static memory is used, D31-24 write enable signal When byte-control static memory is used, selection signal for D31-24 When SDRAM is used, selection signal for D31–D24.
SDRAM Clock enable	O	CKE	Enable the SDRAM clock
DMAC0 acknowledge signal	O	DACK0	DMAC data acknowledge to DREQ0
DMAC1 acknowledge signal	O	DACK1	DMAC data acknowledge to DREQ1
End of process	O	EOP	Indicates the end of DMA transfer

Address enable	O	AEN	Active high to indicate that the system is in DMA transfer mode
Wait	I	Wait# / PWAIT#	External wait state request signal for memory-like devices Also used as external wait state request signal from PC card
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
PC card enable 1	O	PCE1#	Selects a PC card and enables low byte
PC card enable 2	O	PCE2#	Selects a PC card and enables high byte
PC card socket select	O	PSKTSEL	Control external steering logic. When PSKTSEL is low, socket 0 is selected. When PSKTSEL is high, socket 1 is selected
PC card IO select 16	I	IOIS16	Indicates that current address is 16-bit I/O address. Also can be used as DMA request
PC card battery voltage detect 2	I	BVD2	Used as DMA request for I/O card
PC card input port acknowledge	I	INPACK#	Used as DMA request for I/O card
Boot mode configuration	I	BOOT_SEL[3:0]	Configure the boot memory



## 1.4 Boot Configuration

BOOT\_SEL[3:0] pins are used to configure the boot memory, memory size and other information. BOOT\_SEL[3] selects CS0 or NAND flash (CS3). When boot from CS0, BOOT\_SEL[1:0] is used to configure the memory width. When boot from NAND flash, BOOT\_SEL[0] is used to configure the memory width, BOOT\_SEL[1] is used to configure NAND flash page size, BOOT\_SEL[2] is used to configure NAND flash address cycles. Following table lists all boot configurations.

**Table 1-2 Boot Configuration**

BOOT_SEL [3]	BOOT_SEL [2]	BOOT_SEL [1]	BOOT_SEL [0]	Description
0	X	0	0	Boot from CS0 with 32-bit width
0	X	0	1	Boot from CS0 with 16-bit width
0	X	1	0	Boot from CS0 with 8-bit width
0	X	1	1	Reserved
1	0	0	0	Boot from NAND Flash on CS3 with 8-bit width, 512B page size, 2 page address cycles
1	0	0	1	Boot from NAND Flash on CS3 with 16-bit width, 512B page size, 2 page address cycles
1	0	1	0	Boot from NAND Flash on CS3 with 8-bit width, 2KB page size, 2 page address cycles
1	0	1	1	Boot from NAND Flash on CS3 with 16-bit width, 2KB page size, 2 page address cycles
1	1	0	0	Boot from NAND Flash on CS3 with 8-bit width, 512B page size, 3 page address cycles
1	1	0	1	Boot from NAND Flash on CS3 with 16-bit width, 512B page size, 3 page address cycles
1	1	1	0	Boot from NAND Flash on CS3 with 8-bit width, 2KB page size, 3 page address cycles
1	1	1	1	Boot from NAND Flash on CS3 with 16-bit width, 2KB page size, 3 page address cycles

### Notes:

When system is configured to boot from CS0, the data width of static memory bank 0 is configured by BOOT\_SEL[1:0] pins. Data width of other static memory banks can be configured to 8, 16 or 32 bits by software. When system is configured to boot from NAND flash, the data width of static memory bank 3 is configured by BOOT\_SEL[0] pins. Data width of other static memory banks can be configured to 8, 16 or 32 bits by software.

When system is configured to boot from NAND flash, the upper 4KB (0x1FC00000 to 0x1FFFFFFF) of static bank 0 is mapped to on-chip boot ROM.

To support large SDRAM space, EMC re-maps the physical address 0x00000000-0x07FFFFFF to 0x20000000-0x27FFFFFF. Software must configure the SDRAM base address by the re-mapped address.

Following figure shows the physical space map.

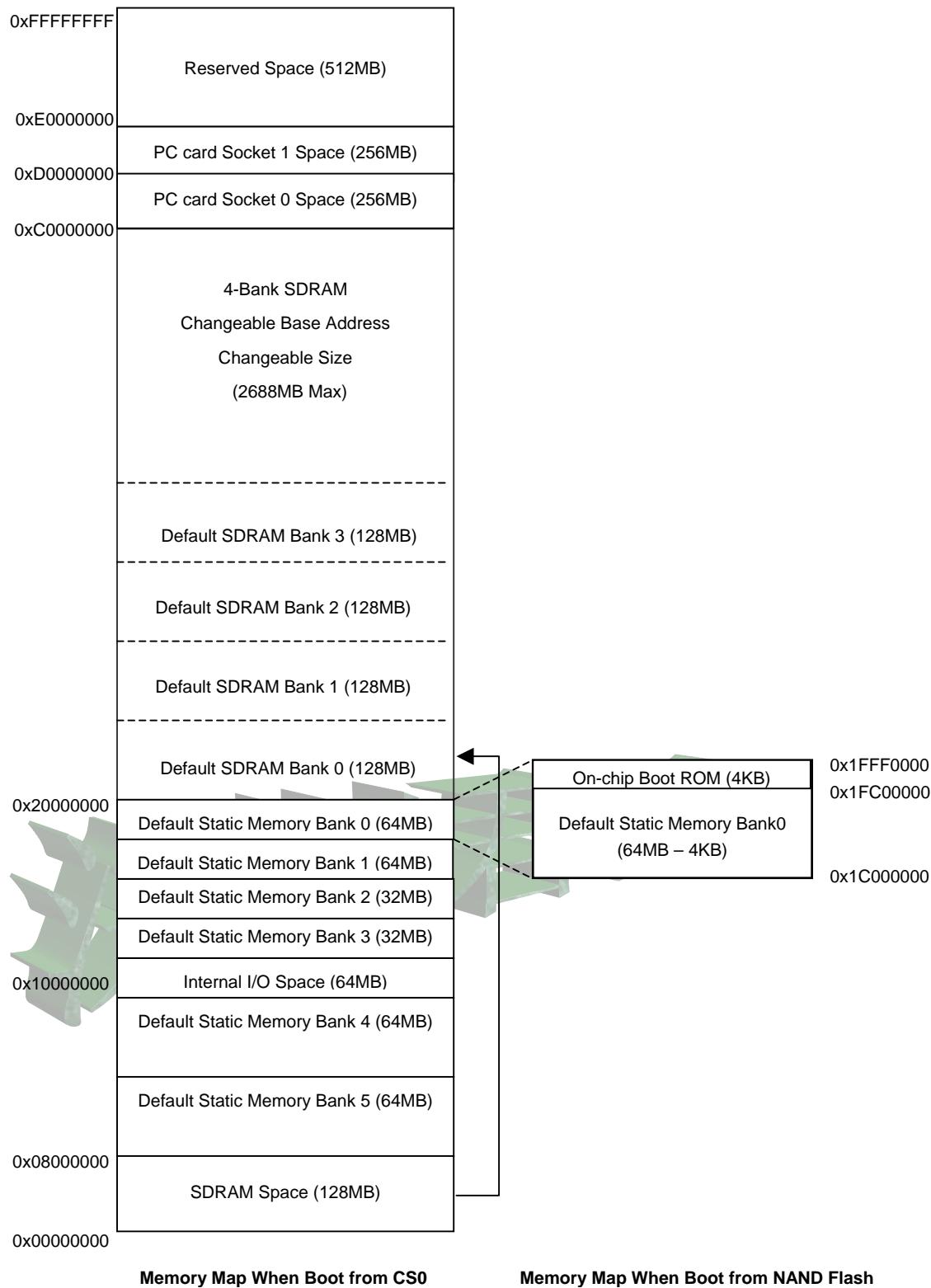
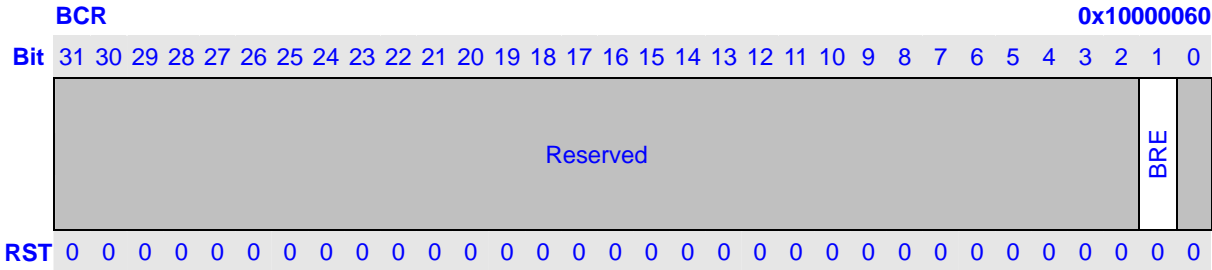


Figure 1-2 Physical Address Space Map

1.5 Bus Control

A 32-bit read/write bus control register (BCR) is defined to specify the behavior of EMC on the system bus. It is initialized to 0x00000001 by any reset.



Bits	Name	Description	RW
31:2	Reserved		R
1			
23	Reserved	Writes to these bits have no effect and always read as 0	R
0	BRE	Bus Release Enable. When clear, one 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. <div>0 The system bus can not be released during an access</div> <div>1 The system bus can be released during an access</div>	RW

## 1.6 Static Memory Interface

### 1.6.1 Static Memory Registers

Table 1-3 Static Memory Registers

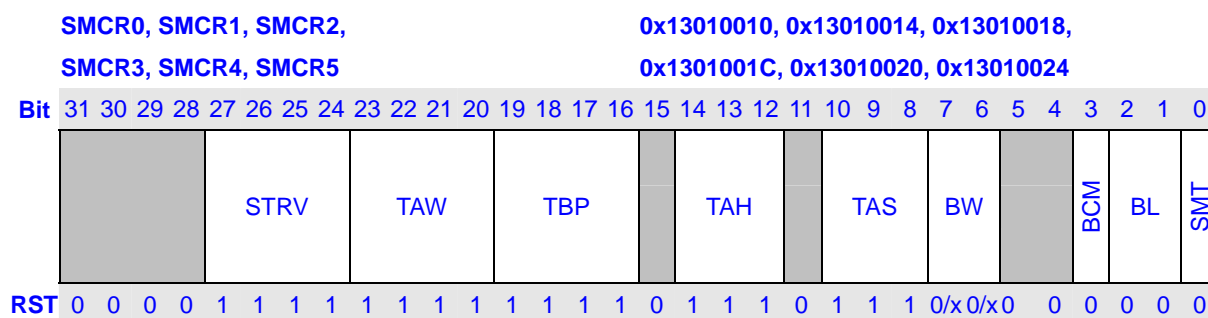
Name	Description	RW	Reset Value	Address	Access Width
SMCR0	Static memory control register 0	RW	0x0FFF7700 <sup>*1</sup>	0x13010010	32
SMCR1	Static memory control register 1	RW	0x0FFF7700	0x13010014	32
SMCR2	Static memory control register 2	RW	0x0FFF7700	0x13010018	32
SMCR3	Static memory control register 3	RW	0x0FFF7700 <sup>*2</sup>	0x1301001C	32
SMCR4	Static memory control register 4	RW	0x0FFF7700	0x13010020	32
SMCR5	Static memory control register 5	RW	0x0FFF7700	0x13010024	32
SACR0	Static memory bank 0 address configuration register	RW	0x00001CFC	0x13010030	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
SACR5	Static memory bank 5 address configuration register	RW	0x0000 08FC	0x13010044	32

**Note:**

- 1 When system is configured to boot from CS0, the BW field of SMCR0 is configured by BOOT\_SEL[1:0] pins.
- 2 When system is configured to boot from NAND flash, the BW field of SMCR3 is configured by BOOT\_SEL[0] pins.

### 1.6.1.1 Static Memory Control Register (SMCR0~5)

SMCR0-5 are 32-bit read/write registers that contain control bits for configuring static memory. On reset, SMCR0~5 are initialized to 0x0FFF7700, with exception of BW fields of SMCR0 and SMCR3. When system is configured to boot from CS0, the BW field of SMCR0 is configured by BOOT\_SEL[1:0] pins. When system is configured to boot from NAND flash, the BW field of SMCR3 is configured by BOOT\_SEL[0] pin.



Bits	Name	Description	RW
31:28	Reserved		R
27:24	STRV	<b>Static Memory Recovery Time:</b> Specify 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	<b>Access Wait Time:</b> 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. <div> <div>TAW3~0 Wait cycle</div> <div>Wait# Pin</div> <div>0000 0 cycle Ignord</div> <div>0001 1 cycle Enabled</div> <div>0010 2 cycles Enabled</div> <div>0011 3 cycles Enabled</div> <div>0100 4 cycles Enabled</div> <div>0101 5 cycles Enabled</div> <div>0110 6 cycles Enabled</div> <div>0111 7 cycles Enabled</div> <div>1000 8 cycles Enabled</div> <div>1001 9 cycles Enabled</div> <div>1010 10 cycles Enabled</div> <div>1011 12 cycles Enabled</div> <div>1100 15 cycles Enabled</div> <div>1101 20 cycles Enabled</div> </div>	RW

		1110 25 cycles Enabled 1111 31 cycles Enabled (Initial Value)	
19:16	TBP	<b>Burst Pitch Time:</b> 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. <b>TBP3~0 Wait cycle Wait# Pin</b> 0000 0 cycle Ignord 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		R
14:12	TAH	<b>Address Hold Time:</b> These bits specify the number of wait cycles to be inserted from negation of read/write strobe to address. <b>TAH2~0 Wait cycle</b> 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
11	Reserved		R
10:8	TAS	<b>Address Setup Time:</b> These bits specify the number of wait cycles (0~7 cycles) to be inserted from address to assertion of read/write strobe. Its initial value is 0x7 (7 cycles).	
7:6	BW	<b>Bus Width :</b> These bits specify the bus width. When system is configured to boot from CS0, this field of SMCR0 is read only and decoded from the value of BOOT_SEL[1:0] in a reset. When system is configured to boot from NAND flash, this field of SMCR3 is read only and decoded from	



		<p>value of BOOT_SEL[0] in a reset. For other banks, this filed is writeable and are initialized to 0 by a reset.</p> <p><b>BW1~0    Bus Width</b></p> <table><tr><td>00</td><td>8 bits (Initial Value)</td></tr><tr><td>01</td><td>16 bits</td></tr><tr><td>10</td><td>32 bits</td></tr><tr><td>11</td><td>Reserved</td></tr></table>	00	8 bits (Initial Value)	01	16 bits	10	32 bits	11	Reserved	
00	8 bits (Initial Value)										
01	16 bits										
10	32 bits										
11	Reserved										
5:4	Reserved		R								
3	BCM	<p><b>SRAM Byte Control Mode (BCM):</b> When SRAM is connected; this bit specifies the type of SRAM. This bit is only valid when SMT is set to 0.</p> <p><b>BCM        Description</b></p> <table><tr><td>0</td><td>SRAM is set to normal mode (Initial Value)</td></tr><tr><td>1</td><td>SRAM is set to byte control mode</td></tr></table>	0	SRAM is set to normal mode (Initial Value)	1	SRAM is set to byte control mode	RW				
0	SRAM is set to normal mode (Initial Value)										
1	SRAM is set to byte control mode										
2:1	BL	<p><b>Burst Length (BL1, BL0):</b> When Burst ROM is connected; these bits specify the number of burst in a access. These bits are only valid when SMT is set to 1.</p> <p><b>BL1~0    Burst Length</b></p> <table><tr><td>00</td><td>4 consecutive accesses. Can be used with 8-, 16-, or 32-bit bus width (Initial Value).</td></tr><tr><td>01</td><td>8 consecutive accesses. Can be used with 8-, 16-, or 32-bit bus width</td></tr><tr><td>10</td><td>16 consecutive accesses. Can only be used with 8- or 16-bit bus width. Do not specify for 32-bit bus width</td></tr><tr><td>11</td><td>32 consecutive accesses. Can only be used with 8-bit bus width</td></tr></table>	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	
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	<p><b>Static Memory Type (SMT):</b> This bit specifies the type of static memory.</p> <p><b>SMT        Description</b></p> <table><tr><td>0</td><td>Normal Memory (Initial Value)</td></tr><tr><td>1</td><td>Burst ROM</td></tr></table>	0	Normal Memory (Initial Value)	1	Burst ROM	RW				
0	Normal Memory (Initial Value)										
1	Burst ROM										

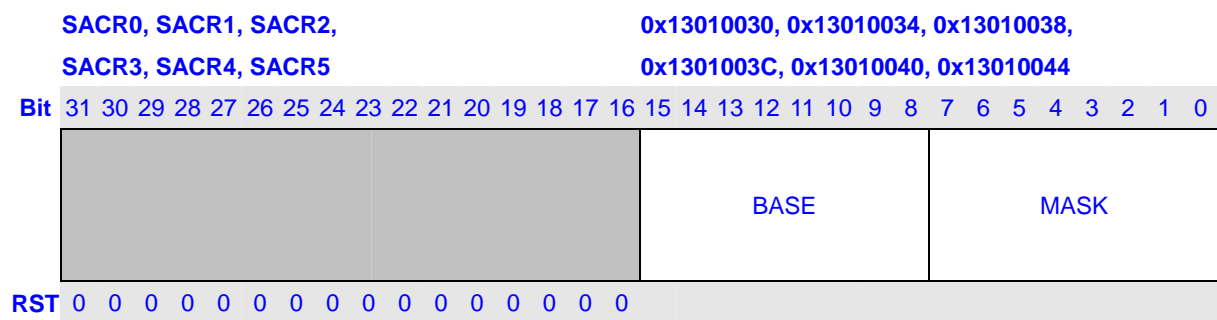
### 1.6.1.2 Static Bank Address Configuration Register (SACR0~5)

The Static Bank Address Configuration Registers (SACR0~5) define the physical address for static memory bank 0 to 5, 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.



Bits	Name	Description	RW
31:16	Reserved	Writes to these bits have no effect and read always as 0.	R
15:8	BASE	<b>Address Base:</b> Defines the base address of Static Bank $n$ ( $n = 0$ to 5). The initial values are: SACR0.BASE      0x1C SACR1.BASE      0x18 SACR2.BASE      0x16 SACR3.BASE      0x14 SACR4.BASE      0x0D SACR5.BASE      0x0C	RW
23:20	MASK	<b>Address Mask:</b> Defines the mask of Static Bank $n$ ( $n = 0$ to 5). The initial values are: SACR0.MASK      0xFC SACR1.MASK      0xFC SACR2.MASK      0xFE SACR3.MASK      0xFE SACR4.MASK      0xFE SACR5.MASK      0xFE	RW

## 1.7 NAND Flash Interface

NAND flash can be connected to static memory bank 3. Both 8-bit and 16-bit NAND flashes are supported. Hardware ECC generator is implemented (for hardware detecting and software correcting). System can be configured to boot from NAND flash.

### 1.7.1 NAND FLASH Registers

Table 1-4 NAND Flash Registers

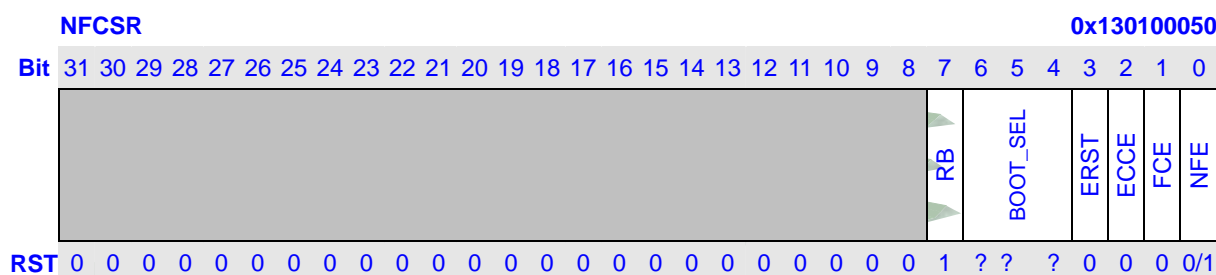
Name	Description	RW	Reset Value	Address	Access Width
NFCSR	NAND flash control/status register	RW	Pin dependent* <sup>1</sup>	0x13010050	32
NFECC	NAND flash ECC data register	R	Undefined	0x13010054	32

**Note:**

1. Depend on BOOT\_SEL[2:0] pins.

#### 1.7.1.1 NAND Flash Control/Status Register (NFCSR)

NFCSR are 32-bit read/write registers that configure NAND flash. It is initialized by any reset and the initial value depends on BOOT\_SEL pins.

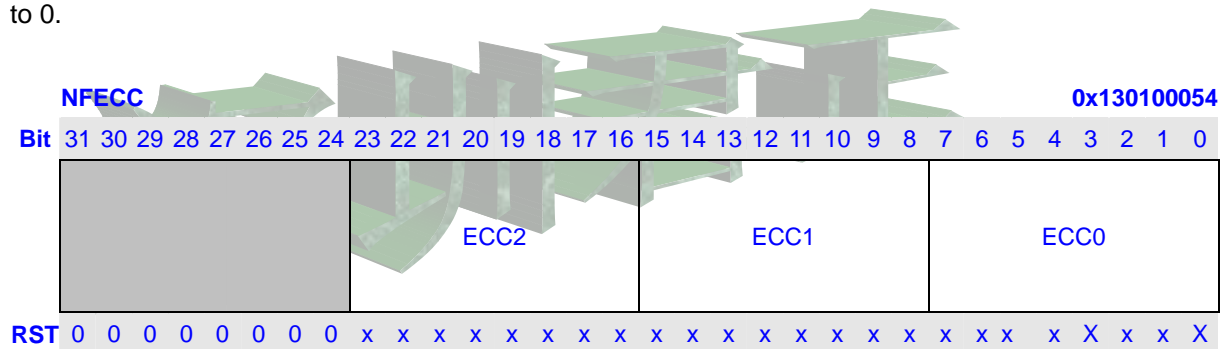


Bits	Name	Description	RW
31:16	Reserved	Writes to these bits have no effect and read always as 0.	R
7	RB	<b>NAND Flash FRB# Status</b> : NAND flash ready/busy status. RB      Description 0      NAND flash is busy 1      NAND flash is ready(Initial value)	R
6:4	BOOT_SEL	Value of BOOT_SEL[2:0] pin. The value of BOOT_SEL[2:0] is locked into this field during reset. This field is read only.	R
3	ERST	<b>NAND Flash ECC Reset</b> : Reset the ECC generator. When set, the ECC generator is initialized. This bit is cleared automatically by hardware and always read as zero. ERST    Description 0      ECC generator is not reset (Initial value) 1      ECC generator is reset	R

2	ECCE	<b>NAND Flash ECC Enable (ECCE):</b> ECC generator enable/disable. <b>ECCE    Description</b> 0        ECC generator is disabled. (Initial value) 1        ECC generator is enabled	RW
1	FCE	<b>NAND Flash FCE# Assertion Control :</b> Controls the assertion of NAND Flash FCE#. When set, FCE# is always asserted until this bit is cleared. When the NAND flash require FCE# to be asserted during read busy time, this bit should be set <b>FCE        Description</b> 0        FCE# is asserted as normal static chip enable(Initial value) 1        FCE# is always asserted	RW
0	NFE	<b>NAND Flash Enable:</b> Specifies if NAND flash is connected to static bank 3. When system is configured to boot from CS0, this bit is initialized to 0. When system is configured to boot from NAND flash, this bit is initialized to 1. <b>NFE        Description</b> 0        Static bank 3 is not used as NAND flash. 1        Static bank 3 is used as NAND flash.	RW

### 1.7.1.2 NAND Flash ECC Data Register (NFECC)

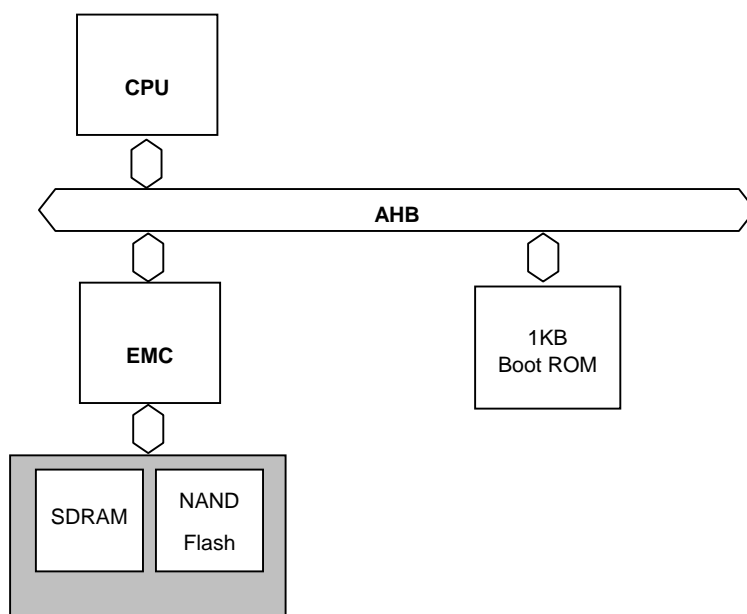
NAND Flash ECC Data Register (NFECC) is a 32-bit read only register that contains the result of ECC calculation. It is not initialized by any reset. When ERST of NFCSR is set, NFCEE is initialized to 0.



Bits	Name	Description	RW
31:24	Reserved	Writes to these bits have no effect and read always as 0.	R
23:16	ECC2	Byte 2 of ECC	R
15:8	ECC1	Byte 1 of ECC	R
7:0	ECC0	Byte 0 of ECC	R

### 1.7.2 NAND Flash Boot

To support boot from NAND flash, 1KB on-chip Boot ROM is implemented. Following figure illustrates the structure of NAND Flash Boot Loader.



**Figure 1-3 Structure of NAND Flash Boot Loader**

When system is configured to boot from NAND flash, the Boot ROM is mapped to physical address of 0x1FC00000, which is just the boot address of CPU. After reset, the program in Boot ROM is executed and the program will copy the first 4K bytes of NAND flash to cache. The boot code in the cache is executed.

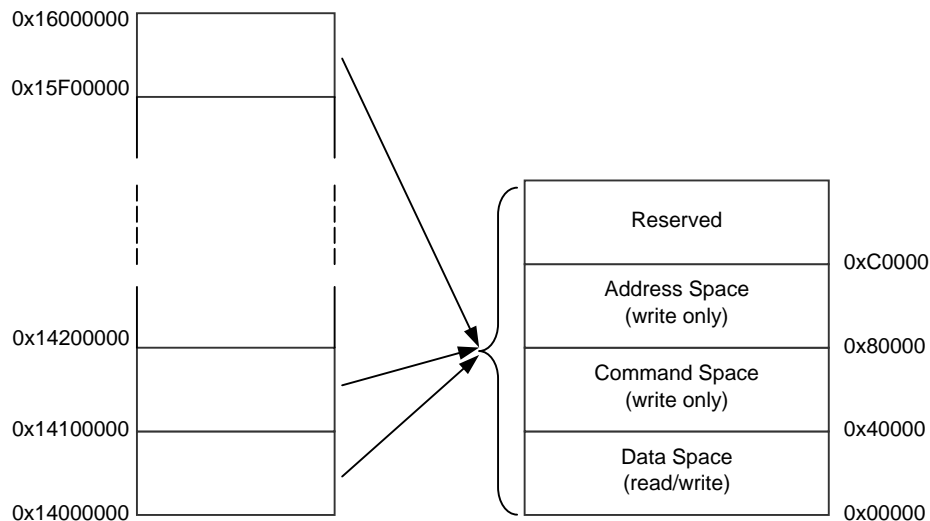
Generally, the boot code will copy more NAND flash content to SDRAM. Hardware ECC can be utilized to check the data validity. Then the main program will be executed on SDRAM.

BOOT\_SEL[3:0] is used to configure boot mode and NAND flash information. Table 1-2 illustrates the detail of all configurations. Software may know the value of BOOT\_SEL[2:0] pins through BOOT\_SEL bits of NFCSR register. When system is configured to boot from NAND flash, BW field of SMCR3 is initialized according to BOOT\_SEL[0] and NFE bit of NFCSR is set to 1 automatically.

**Note:** Because Boot ROM is mapped to 0x1FC00000 in NAND flash boot mode, which overlaps with static bank 0 (from 0x1C000000 to 0x1FFFFFFF). Software should re-configure static bank 0. Or else the upper 4K bytes of CS0 cannot be accessed.

### 1.7.3 NAND Flash Operations

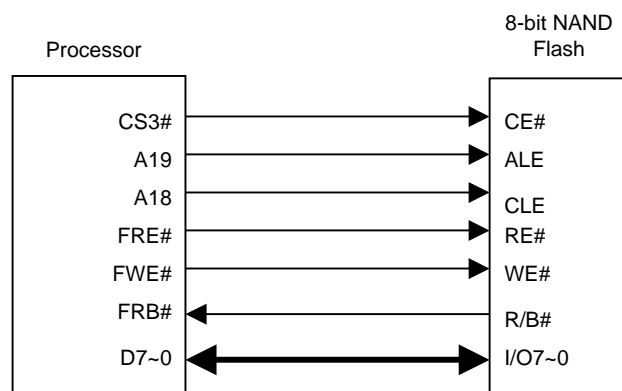
Set NFE bit of NAND Flash Control/Status Register (NFCSR) will enable access to NAND flash. The partition of static bank 3 is changed as following figure. Writes to any of address space will be translated to NAND flash address cycle. Writes to any of command space will be translated to NAND flash command 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 enlarge the speed of data read/write. The DMA access cannot exceed the page boundary (512 bytes or 2K bytes) of NAND flash.



**Figure 1-4 Static Bank 3 Partition When NAND Flash is Used**

The timing of NAND flash access is configured by SMCR3 and is same as normal static memory timing, except that CS3# is controlled by NFCE bit NFCSR. CS3# is always asserted when NFCE is 1. When NFCE is 0, CS3# is asserted as normal static memory access.

The control signals for direction connection of NAND flash are CS3#, FRE#, FEW#, FRB#, A19 and A18. Following figure shows the connection between processor and NAND Flash.



**Figure 1-5 Example of 8-bit NAND Flash Connection**

Hardware ECC generation for 8-/16-bit organization is implemented. Hamming algorithm is adopted. ECC parity code consists of 24 bits per 512 bytes (256 halfwords) and 22 bits per 256 bytes. Following table shows the ECC code assignment.

**Table 1-5 512-Byte ECC Parity Code Assignment Table For 8-bit NAND Flash**

	Data7	Data6	Data5	Data4	Data3	Data2	Data1	Data0
ECC0	P64	P64'	P32	P32'	P16	P16'	P8	P8'
ECC1	P1024	P1024'	P512	P512'	P256	P256'	P128	P128'
ECC2	P4	P4'	P2	P2'	P1	P1'	P2048	P2048'

24-bit ECC parity code = 18-bit line parity + 6-bit column parity

**Table 1-6 256-Byte ECC Parity Code Assignment Table For 8-bit NAND Flash**

	Data7	Data6	Data5	Data4	Data3	Data2	Data1	Data0
ECC0	P64	P64'	P32	P32'	P16	P16'	P8	P8'
ECC1	P1024	P1024'	P512	P512'	P256	P256'	P128	P128'
ECC2	P4	P4'	P2	P2'	P1	P1'	X	X

22-bit ECC parity code = 16-bit line parity + 6-bit column parity

**Table 1-7 256-Halfword ECC Parity Code Assignment Table For 16-bit NAND Flash**

	Data7	Data6	Data5	Data4	Data3	Data2	Data1	Data0
ECC0	P128	P128'	P64	P64'	P32	P32'	P16	P16'
ECC1	P2048	P2048'	P1024	P1024'	P512	P512'	P256	P256'
ECC2	P8	P8'	P4	P4'	P2	P2'	P1	P1'

24-bit ECC parity code = 16-bit line parity + 8-bit column parity

---

### NAND Flash Initialize Sequence

1. Configure SMCR3 according to the NAND flash AC characteristics.
2. Set NFE bit of NFCSR to 1

### NAND Flash Program Sequence

1. Set NFCE bit of NFCSR to 1 to assert CS3# continuously
2. Write 0x80 to command space to issue Page Program command
3. Write 2 or 3 bytes of address to address space
4. Enable and reset the ECC generator by setting ECCE and ERST bits of NFCSR.
5. Write 256 or 512 bytes data to data space
6. Clear ECCE bit of NFCSR to disable ECC generator
7. Read out the ECC parity code from NFECC register
8. Write 16-byte redundant data
9. Write 0x10 to command space to issue Page Program command
10. Clear NFCE bit of NFCSR to deassert CS3#
11. Check RB bit of NFCSR to wait the program complete

### NAND Flash Read Sequence

1. Set NFCE bit of NFCSR to 1 to assert CS3# continuously
2. Write 0x00 to command space to issue Read command
3. Write 2 or 3 bytes of address-to-address space
4. Enable and reset the ECC generator by setting ECCE and ERST bits of NFCSR.
5. Check RB bit of NFCSR to wait for NAND flash is not busy
6. Read 256 or 512 bytes from NAND flash (DMA can be used)
7. Clear ECCE bit of NFCSR to disable ECC generator
8. Read 16-byte redundant data from NAND flash
9. Read out the ECC parity code from NFECC register
10. Compare these two parity codes and correct the error bit or run the error routine.
11. If continuous pages are needed to be read, repeat steps from 5 to 10.
12. Clear NFCE bit of NFCSR to deassert CS3#

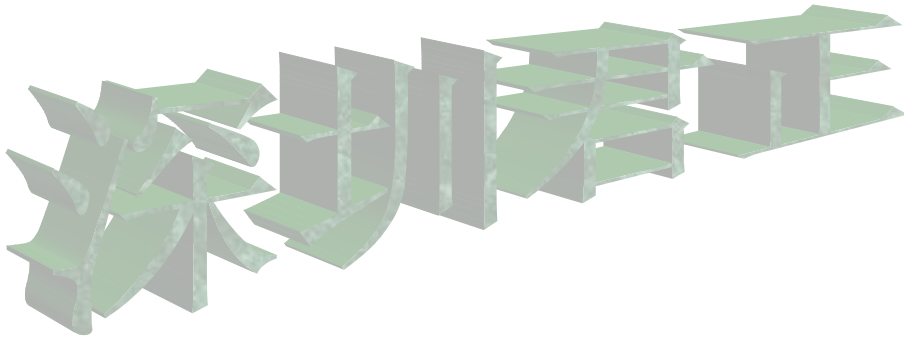
### ECC Generation Sequence for large page

In large page (2KBX8/1KHWX16 org.), 24-bit ECC code is generated for every 512 bytes or 256 halfwords data. Software gets ECC codes as following steps:

1. Set ERST bit of NFCSR to reset ECC generator
  2. Read/write 512 bytes / 256 halfwords data
  3. Read out 24-bit ECC code from NFECC register
  4. Repeat step 1 to 3 until whole page is completed
-



5. Write spare area according to above ECC codes.



## 1.8 SDRAM Interface

### 1.8.1 SDRAM Registers

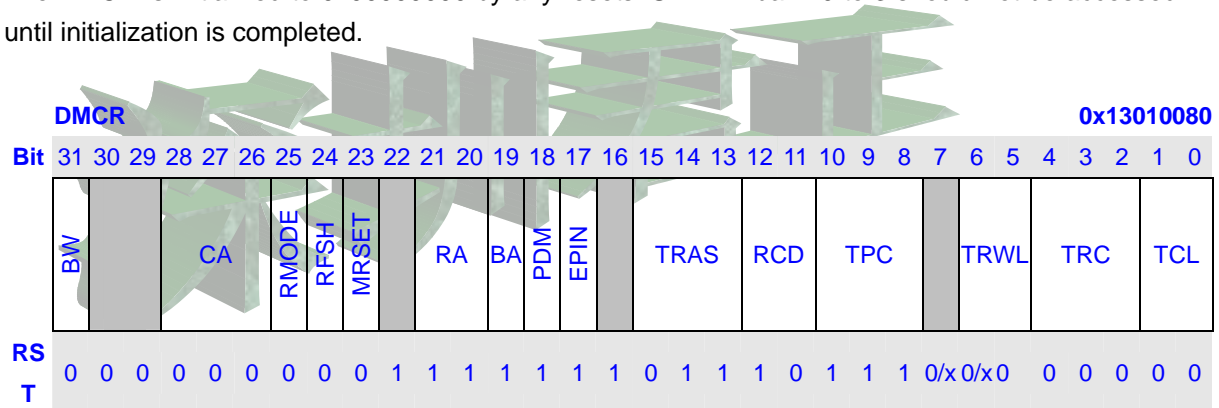
Table 1-8 SDRAM Registers

Name	Description	RW	Reset Value	Address	Access Width
SDRAM					
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
SDMR0	Mode register of SDRAM bank 0	W	0x??	0x1301A000	8
SDMR1	Mode register of SDRAM bank 1	W	0x??	0x1301B000	8

#### 1.8.1.1 SDRAM Control Register (DMCR)

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

The DMCR is initialized to 0x00000000 by any resets. SDRAM bank 0 to 3 should not be accessed until initialization is completed.



Bits	Name	Description	RW
31	BW	Specifies the data bus width of SDRAM  <b>BW Description</b> 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	<b>Column Address Width:</b> Specify the column address width of connected SDRAM chip.	RW

		<b>CA Description</b> 000 8 bits column address 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	<b>Refresh Mode.</b> <b>RMODE Description</b> 0 Auto-refresh 1 Self-refresh	RW
24	RFSH	<b>Refresh Control.</b> <b>RFSH Description</b> 0 No refresh is performed (Initial value) 1 Refresh is performed	RW
23	MRSET	<b>Mode Register Set:</b> 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. <b>MRSET Description</b> 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	<b>Row Address Width:</b> Specify the column address width of connected SDRAM. <b>RA Description</b> 00 11-bit row address (Initial value) 01 12-bit row address 10 13-bit row address 11 Reserved	RW
19	BA	<b>Bank Address Width:</b> Specify the number of bank select signals for one chip select. <b>BA Description</b> <b>0</b> 1-bit bank address is used (2 banks each chip select) (Initial value) <b>1</b> 2-bit bank address is used (4 banks each chip select)	RW
18	PDM	<b>Power Down Mode:</b> 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. <b>PDM Description</b>	RW

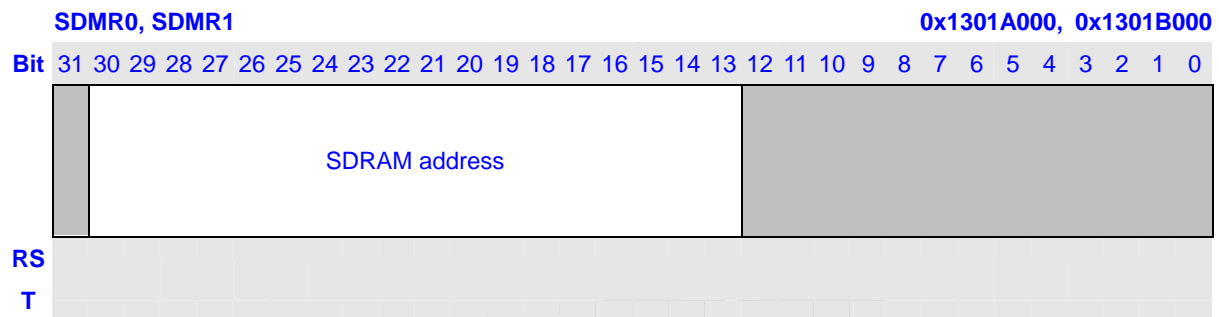
		<div>0Non-power-down mode (Initial value)</div> <div>1Power-down mode</div>	
17	EPIN	<b>CKE Pin Control:</b> Controls the level of CKE pin. Clearing this bit by software causes a power-down command (if CKOEN of CPM is 1). Caution: after power-down command, all commands except power-down-exit are prohibited. Setting this bit by software causes a power-down-exit command. Setting EPIN is a part of initializes procedure for SDRAM. <div><div>EPIN</div><div>Description</div><div>0CKE pin is deserted (Initial value)</div><div>1CKE pin is asserted</div></div>	RW
16	Reserved	Writes to these bits have no effect and always read as 0.	R
15:13	TRAS	<b>RAS Assertion Time:</b> When synchronous DRAM is connected, these bits set the minimum CKE negation time after self-refresh command is issued. <div><div>TRAS</div><div>Description</div><div>0004 (Initial value)</div><div>0015</div><div>0106</div><div>0117</div><div>1008</div><div>1019</div><div>11010</div><div>11111</div></div>	RW
12:11	RCD	<b>RAS–CAS Delay:</b> Set the SDRAM bank active-read/write command delay time. <div><div>RCD</div><div>Description</div><div>001(Initial value)</div><div>012</div><div>103</div><div>114</div></div>	RW
10:8	TPC	<b>RAS Precharge Time:</b> Specify the minimum number of cycles until the next bank active command is output after precharging. <div><div>TPC</div><div>Description</div><div>0001 cycle (Initial value)</div><div>0012 cycles</div><div>0103 cycles</div><div>0114 cycles</div><div>1005 cycles</div><div>1016 cycles</div><div>1107 cycles</div><div>1118 cycles</div></div>	RW
7	Reserved	Writes to these bits have no effect and always read as 0.	R
6:5	TRWL	<b>Write Precharge Time:</b> Set the SDRAM write precharge delay time. In	RW

		<p>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.</p> <table><thead><tr><th>TRWL</th><th>Description</th></tr></thead><tbody><tr><td>00</td><td>1 cycle (Initial value)</td></tr><tr><td>01</td><td>2 cycles</td></tr><tr><td>10</td><td>3 cycles</td></tr><tr><td>11</td><td>4 cycles</td></tr></tbody></table>	TRWL	Description	00	1 cycle (Initial value)	01	2 cycles	10	3 cycles	11	4 cycles									
TRWL	Description																				
00	1 cycle (Initial value)																				
01	2 cycles																				
10	3 cycles																				
11	4 cycles																				
4:2	TRC	<p><b>RAS Cycle Time:</b> For synchronous DRAM, 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.</p> <table><thead><tr><th>TRC</th><th>Description</th></tr></thead><tbody><tr><td>000</td><td>1 cycle (Initial value)</td></tr><tr><td>001</td><td>3 cycle</td></tr><tr><td>010</td><td>5 cycle</td></tr><tr><td>011</td><td>7 cycle</td></tr><tr><td>100</td><td>9 cycle</td></tr><tr><td>101</td><td>11 cycle</td></tr><tr><td>110</td><td>13 cycle</td></tr><tr><td>111</td><td>15 cycle</td></tr></tbody></table>	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
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																				
1:0	TCL	<p><b>CAS Latency:</b> Specify the delay from read command to data becomes available at the outputs.</p> <table><thead><tr><th>TCL</th><th>Description</th></tr></thead><tbody><tr><td>00</td><td>Inhibit (Initial value)</td></tr><tr><td>01</td><td>2 cycles</td></tr><tr><td>10</td><td>3 cycles</td></tr><tr><td>11</td><td>Inhibit</td></tr></tbody></table>	TCL	Description	00	Inhibit (Initial value)	01	2 cycles	10	3 cycles	11	Inhibit	RW								
TCL	Description																				
00	Inhibit (Initial value)																				
01	2 cycles																				
10	3 cycles																				
11	Inhibit																				

### 1.8.1.2 SDRAM Mode Register (SDMR)

The SDRAM mode register (SDMR) is written to via the synchronous DRAM address bus and is a 10-bit write-only register. It sets SDRAM mode for SDRAM bank 0 to 1. 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. Since A0 of the synchronous DRAM is connected to A2 of the processor and A1 of the Synchronous DRAM is connected to A3 of the processor, the value actually written to the synchronous DRAM is the X value shifted two bits right. For example, when 0x230 is written to the SDMR register of SDRAM bank 0, random data is written to the address offset 0xB000 (address Y) + 0x8C0 (value X), or 0xB8C0. As a result, 0x230 is written to the SDMR register. When 0x230 is written to the SDMR register of SDRAM bank 1, random data is written to the address offset 0xC000 (address Y) + 0x8C0 (value X), or 0xC8C0. As a result, 0x230 is written to the SDMR register. The range for value X is 0x000 to 0xFFC.



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.

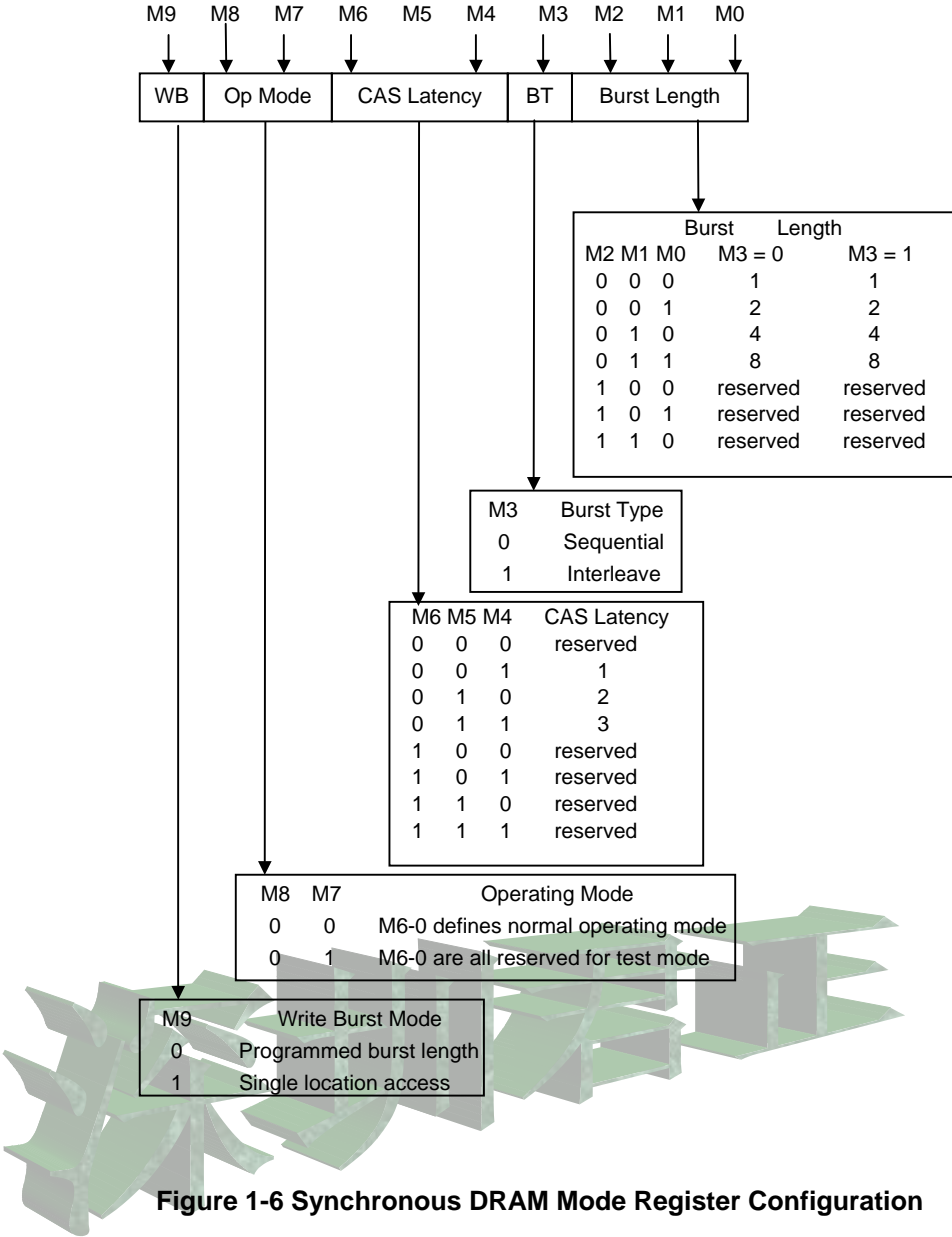
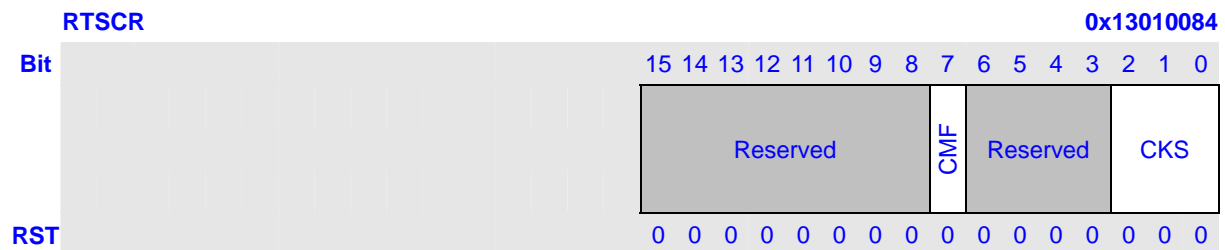


Figure 1-6 Synchronous DRAM Mode Register Configuration

### 1.8.1.3 Refresh Timer Control/Status Register (RTCSR)

The refresh timer control/status register (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.

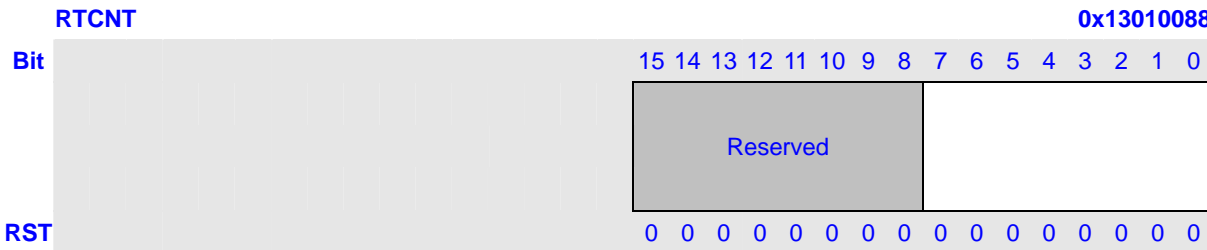


Bits	Name	Description	RW																		
15:8	Reserved	These bits always read 0. Data written to these bits are ignored	R																		
7	CMF	<b>Compare-Match Flag (CMF):</b> Status flag that indicates a match between the refresh timer counter (RTCNT) and refresh time constant register (RTCOR) values. Writes to 1 of this bit have no effect. <table><tr><th>CMF</th><th>Description</th></tr><tr><td>0</td><td>RTCNT and RTCOR values do not match (Initial value) Clear condition: When 0 is written</td></tr><tr><td>1</td><td>RTCNT and RTCOR values match Set condition: When RTCNT = RTCOR</td></tr></table>	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													
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																				
2:0	CKS	<b>Refresh Clock Select Bits:</b> 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. <table><tr><th>CKS</th><th>Description</th></tr><tr><td>000</td><td>Disable clock input (Initial value)</td></tr><tr><td>001</td><td>Bus lock CKO/4</td></tr><tr><td>010</td><td>CKO/16</td></tr><tr><td>011</td><td>CKO/64</td></tr><tr><td>100</td><td>CKO/256</td></tr><tr><td>101</td><td>CKO/1024</td></tr><tr><td>110</td><td>CKO/2048</td></tr><tr><td>111</td><td>CKO/4096</td></tr></table>	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	
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																				



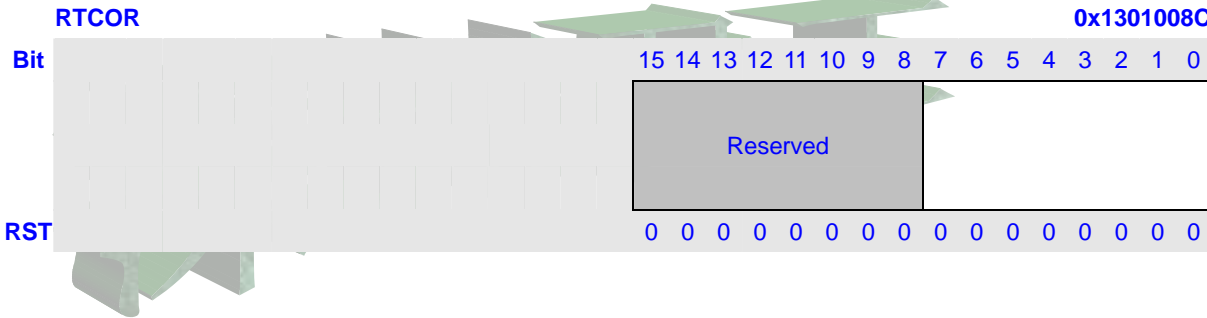
1.8.1.4 Refresh Timer Counter (RTCNT)

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.



1.8.1.5 Refresh Time Constant Register (RTCOR)

The refresh time constant register (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.

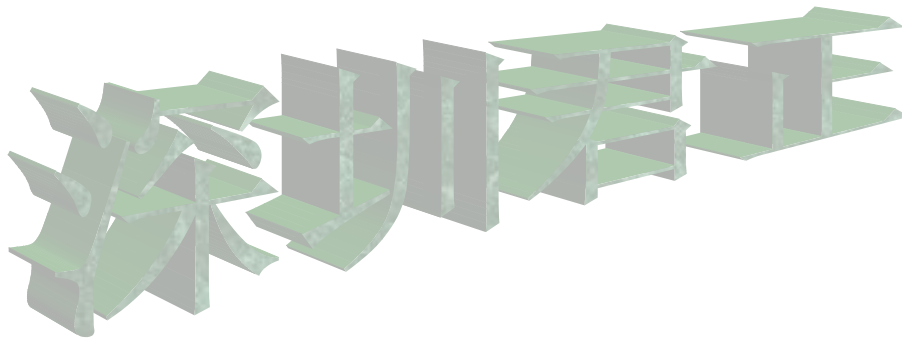


# 1 DMA Controller

DMA controller (DMAC) is dedicated to transfers data between on-chip peripherals (LCD, ETH, UART, etc.) external memories, external devices with DACK and memory-mapped external devices

## 1.1 Features

- Support up to 8 independent DMA channels
- Transfer data units: byte, 2-byte (half word), 4-byte (word), 16-byte or 32-byte
- Transfer number of data unit:  $1 \sim 2^{16} - 2$
- Independent source and destination port width: 8-bit, 16-bit, 32-bit
- External DMA request detection mode: level or edge detection
- Two channel priority modes: fixed, round robin.



## 1.2 CHIP PIN Description

Table 1-1 CHIP PINs for DMAC

Name	Type	Description
DREQ0	Input	Channel 0 External DMA Request
DREQ1	Input	Channel 1 External DMA Request
DACK0	Output	Channel 0 External DMA Acknowledge
DACK1	Output	Channel 1 External DMA Acknowledge
EOP	Output	Channel 0 -1 External DMA Transfer End
AEN	Output	Active High to Enable External DMA Transfer

## 1.3 Register Descriptions

Table 1-2 DMAC Registers

Name	Description	RW	Reset Value	Address	Access Size (bit)
DSA0	DMA Source Address 0	RW	0x0	0x13020000	32
DDA0	DMA Destination 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
DSA1	DMA Source Address 1	RW	0x0	0x13020020	32
DDA1	DMA Destination 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
DSA2	DMA Source Address 2	RW	0x0	0x13020040	32
DDA2	DMA Destination 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
DSA3	DMA Source Address 3	RW	0x0	0x13020060	32
DDA3	DMA Destination 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
DSA4	DMA Source Address 4	RW	0x0	0x13020080	32
DDA4	DMA Destination 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
DSA5	DMA Source Address 5	RW	0x0	0x130200A0	32
DDA5	DMA Destination 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
DSA6	DMA Source Address 6	R/W	0x0	0x130200C0	32
DDA6	DMA Destination Address 6	R/W	0x0	0x130200C4	32
DTC6	DMA Transfer Count 6	R/W	0x0	0x130200C8	32
DRT6	DMA Request Source 6	R/W	0x0	0x130200CC	32
DCS6	DMA Channel Control/Status 6	R/W	0x0	0x130200D0	32
DSA7	DMA Source Address 7	R/W	0x0	0x130200E0	32
DDA7	DMA Destination Address 7	R/W	0x0	0x130200E4	32
DTC7	DMA Transfer Count 7	R/W	0x0	0x130200E8	32
DRT7	DMA Request Source 7	R/W	0x0	0x130200EC	32
DCS7	DMA Channel Control/Status 7	R/W	0x0	0x130200F0	32
DIRQP	DMA Interrupt Pending	R	0x0	0x130200F8	32
DMAC	DMA Control	R/W	0x0	0x130200FC	32

Notes: no channel 6 and 7 in JZ4730 CHIP

### 1.3.1 DMA Source Address (DSAn, n = 0 ~ 7)

DSA0, DSA1, DSA2, 0x13020000, 0x13020020, 0x13020040,  
 DSA3, DSA4, DSA5, 0x13020060, 0x13020080, 0x130200a0,  
 DSA6, DSA7 0x130200c0, 0x130200e0,

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

Bits	Name	Description	RW
31:0	SA	Source address	RW

### 1.3.2 DMA Destination Address (DDAn, n = 0 ~ 7)

DDA0, DDA1, DDA2, 0x13020004, 0x13020024, 0x13020044,  
 DDA3, DDA4, DDA5, 0x13020064, 0x13020084, 0x130200a4,  
 DDA6, DDA7 0x130200c4, 0x130200e4,

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

DA

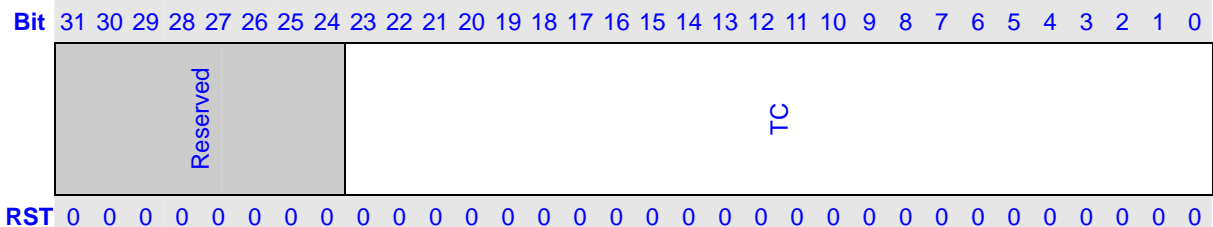
RST 0

Bits	Name	Description	RW
31:0	DA	Destination address	RW

31:0	DA	Destination address	RW
------	----	---------------------	----

### 1.3.3 DMA Transfer Count (DTCn, n = 0 ~ 7)

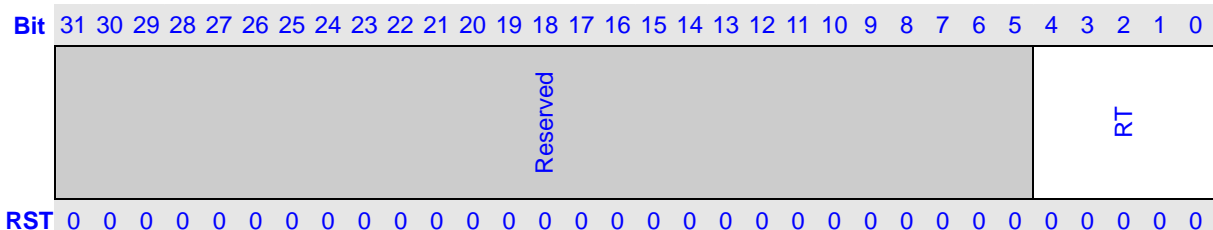
DTC0, DTC1, DTC2, 0x13020008, 0x13020028, 0x13020048,  
DTC3, DTC4, DTC5, 0x13020068, 0x13020088, 0x130200a8,  
DTC6, DTC7 0x130200c8, 0x130200e8,



Bits	Name	Description	RW
31:24	Reserved	Write has no effect, read as zero	R
23:0	TC	Transfer count	RW

### 1.3.4 DMA Request Types (DRTn, n = 0 ~ 7)

DRT0, DRT1, DRT2, 0x1302000c, 0x1302002c, 0x1302004c,  
DRT3, DRT4, DRT5, 0x1302006c, 0x1302008c, 0x130200ac,  
DRT6, DRT7 0x130200cc, 0x130200ec,



Bits	Name	Description	RW
31:5	Reserved	Write has no effect, read as zero	R
4:0	RT	Transfer request type	RW

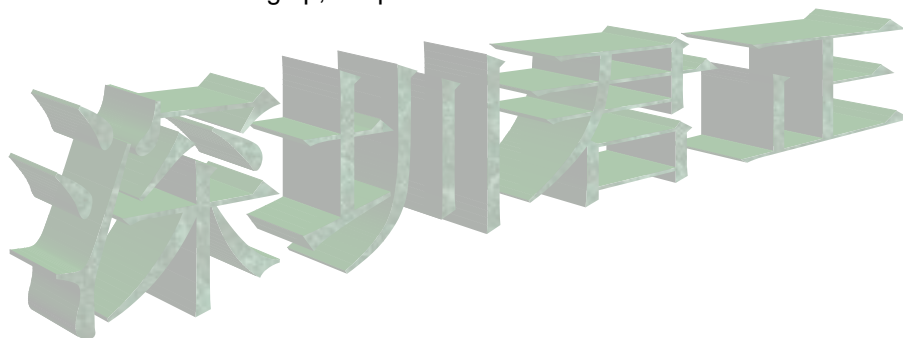
Table 1-3 Transfer Request Types

RS4	Description
0	External request with DREQn, (external address $\leftrightarrow$ external device with DACKn)
1~3	Reserved
4	PCMCIA $\rightarrow$ external address
5	PCMCIA $\leftarrow$ external address
6~7	Reserved
8	Auto-request (ignore RDIL3-0, external address $\rightarrow$ external address)
9	Reserved
10	DES $\rightarrow$ external address
11	DES $\leftarrow$ external address
12~13	Reserved
14	UART3 transmit-fifo-empty transfer request (external address $\rightarrow$ UTHR3)

15	UART3 receive-fifo-full transfer request (URBR3 → external address)
16	UART2 transmit-fifo-empty transfer request (external address → UTHR2)
17	UART2 receive-fifo-full transfer request (URBR2 → external address)
18	UART1 transmit-fifo-empty transfer request (external address → UTHR1)
19	UART1 receive-fifo-full transfer request (URBR1 → external address)
20	UART0 transmit-fifo-empty transfer request (external address → UTHR0)
21	UART0 receive-fifo-full transfer request (URBR0 → external address)
22	SSI transmit-fifo-empty transfer request
23	SSI receive-fifo-full transfer request
24	AIC transmit-fifo-empty transfer request
25	AIC receive-fifo-full transfer request
26	MSC transmit-fifo-empty transfer request
27	MSC receive-fifo-full transfer request
28	OST channel 2 (underflow interrupt, external address→external address space)
29~31	Reserved

**NOTES:**

1. External request 0 and 1 are corresponding to channel 0 and 1 respectively.
2. Only auto request can be concurrently selected in all channels with different source and destination address.
3. For on-chip device DMA request except OST and PCMCIA, the corresponding source or destination address that map to on-chip device must be set as fixed.
4. For external request and PCMCIA/CF request, source or destination address can be set as either fixed or counting up, but port width must be set the same.



### 1.3.5 DMA Channel Control/Status (DCSn, n = 0 ~ 7)

DCS0, DCS1, DCS2, 0x13020010, 0x13020030, 0x13020050,  
 DCS3, DCS4, DCS5, 0x13020070, 0x13020090, 0x130200b0,  
 DCS6, DCS7 0x130200d0, 0x130200f0,

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	EOPM	Reserved	SAI	DAI	Reserved	RDIL	SP	DP	Reserved	TSZ	TM	Reserved	AR	CT	HLT	TIE	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	EACKS	External DMA Acknowledge DACKn (n=0, 1) Output Level Select: select DACKn is active-high or active-low 0, active high; 1, active low	RW
30	EACKM	External DMA Acknowledge DACKn (n=0, 1) Output Mode Select: select DACKn is asserted in read or write cycle. 0, DACK is output in read cycle; 1, DACK is output in write cycle	RW
29:28	ERDM	External Request Detection Mode: 00, Low level detection; 01, Falling edge detection; 10, High level detection; 11, Rising edge detection	RW
27	EOPM	End Of Process Mode (EOPM): set output mode of external transfer end signal EOP. 0, active high; 1, active low	RW
26:24	Reserved	Write has no effect, read as zero	R
23	SAI	Source Address Increment: 0, no increment; 1, increment	RW
22	DAI	Destination 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 1-4	RW
15:14	SP	Source port width: 00, 32-bit; 01, 8-bit; 10, 16-bit; 11, reserved	RW
13:12	DP	Destination port width: 00, 32-bit; 01, 8-bit; 10, 16-bit; 11, reserved	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	TM	Transfer Mode: 0, single mode; 1, block mode	RW
6:5	Reserved	Write has no effect, read as zero	R

4	AR	Address Error (AR): 0, no address error; 1, address error	RW
3	CT	Count Terminate (CT): 0, DMA transfer does not end; 1, DMA transfer does end	RW
2	HLT	DMA halt: 0, DMA transfer is in progress; 1, DMA halt	RW
1	TIE	Transfer Interrupt Enable (TIE): 0, disable interrupt; 1, enable interrupt	RW
0	CTE	Channel transfer enable: 0, disable; 1, enable	RW

Table 1-4 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

### 1.3.6 DMA Interrupt Pending

DIRQP

0x130200F8

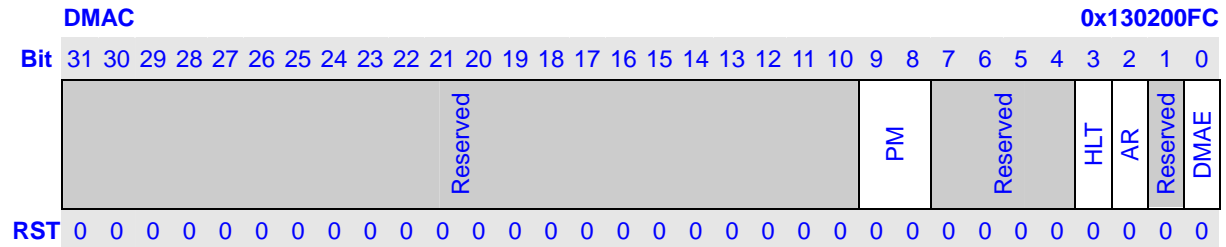
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																CIRQ0	CIRQ1	CIRQ2	CIRQ3	CIRQ4	CIRQ5	CIRQ6	CIRQ7	Reserved							
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31:16	Reserved	Write has no effect, read as zero.	R
15:8	CIRQn	CIRQn (n=0~7) denotes pending status for corresponding channel	RW



		0, no abnormal situation or normal DMA transfer is in progress 1, abnormal situation occurred or normal DMA transfer done	
7:0	Reserved	Write has no effect, read as zero	R

### 1.3.7 DMA Control



Bits	Name	Description	RW
31:10	Reserved	Write has no effect, read as zero.	R
9:8	PM	Channel priority mode: 00, CH0 > CH1 > CH2 > CH3 > CH4 > CH5 > CH6 > CH7 01, CH0 > CH2 > CH3 > CH1 > CH4 > CH6 > CH7 > CH5 10, CH2 > CH0 > CH1 > CH3 > CH6 > CH4 > CH5 > CH7 11, round robin	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. 0, disable DMA channel transfer 1, enable DMA channel transfer	RW

## 1.4 DMA manipulation

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.CT=0 and DTCn=0.
2. For each channel n, initialize DSA<sub>n</sub>, DDAn, DTC<sub>n</sub>, DRT<sub>n</sub>, DCS<sub>n</sub> properly
3. Set DMAC.DMAE=1 and expected DCS<sub>n</sub>.CTE=1 to launch DAM transfer

For a channel with auto-request (DRT<sub>n</sub>.RT=0x8), the transfer begins automatically when the DCS<sub>n</sub>.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 DTC<sub>n</sub> value is decremented by 1 for each successful transaction of a data unit. When the specified number of transfer data unit has been completed (DTC<sub>n</sub> = 0), the transfer ends normally. Meanwhile corresponding bit of DIRQP is set to 1. If DCS<sub>n</sub>.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 DCS<sub>n</sub>.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 DCS<sub>n</sub>.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 DCS<sub>n</sub>.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.

## 1.5 DMA Requests

DMA transfer requests are normally generated from either the data transfer source or destination, but also they can be issued by external devices or on-chip peripherals that are neither the source nor the destination. There are three DMA transfer request types: auto-request, external request, and on-chip peripheral request. For any channel n, its transfer request type is determined through DRT<sub>n</sub>.

### 1.5.1 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 DCS<sub>n</sub>.CTE are set to 1, the transfer begins immediately in channel n which DRT<sub>n</sub>=0x8.

### 1.5.2 External Request

In the mode, up to two transfer request signals DREQ<sub>n</sub> (n=0, 1) from external devices can be accepted by channel 0 and channel 1, respectively. When DMA transfer preparation done, transfer starts when active DREQ<sub>n</sub> asserts. DCS<sub>n</sub>.ERDM determines the detection mode for DREQ<sub>n</sub>. Note that DREQ<sub>n</sub> can be issued either from source or destination.

### 1.5.3 On-Chip Peripheral Request

In the mode, transfer request signals come from on-chip peripherals. All request types except 0x0 and 0x8 (value of DRT) belong to the mode. Both single and block transfer mode are available. Note that in single mode, the transfer byte number for one request detection according to DCSn.RDIL must be equal or less than the byte number according to receive or transmit trigger value of source or destination devices.

## 1.6 DMA Transfer Modes

Each channel can toggles between two transfer modes: single and block

### 1.6.1 Single Mode

A channel with single mode will periodically detect the request signal according to presetting detection interval length (DCSn.RDIL). Moreover, during the transfer, after a transaction of a data unit (8-bit, 16-bit, 32-bit, 16-byte, or 32-byte), an internal arbitrator in the DMA will arbitrate all active channels again to select one to represent DMA's bus request to participate the AHBA bus arbitration. Above process will repeat when the channel captures the bus again until corresponding DCSn.CT bit equals to 1 or abnormal situation (address error, halt) occurs.

### 1.6.2 Block Mode

Once a channel with block mode captures the bus, it will do data transfer continuously until all data units are transferred or abnormal situation occurs. During the process, it does not release the bus so that neither other channels nor other bus masters can take up the bus. In the mode, the channel just detects the request signal once and corresponding DCSn.RDIL is ignored.

## 1.7 Channel Priorities

There are two priority modes: fixed, round robin

### 1.7.1 Fixed Mode

The relative channel priorities are unvaried in the mode.

- CH0 > CH1 > CH2 > CH3 > CH4 > CH5 > CH6 > CH7
- CH0 > CH2 > CH3 > CH1 > CH4 > CH6 > CH7 > CH5
- CH2 > CH0 > CH1 > CH3 > CH6 > CH4 > CH5 > CH7

### 1.7.2 Round Robin Mode

In the mode, there are two priority groups: CH0~CH3 and CH4~CH7. Round robin is performed in each group, and CH0~CH3 always has higher priority than CH4~CH7.

**Table 1-5 Relationship among DMA Transfer connection, Request Mode and Transfer Mode**

Transfer Connection	Request Mode	Transfer Mode	Data Size (bits)	Channel
External memory and external memory	Auto on-chip	Block/Single	8/16/32 16-byte/32-byte	0~7
External memory and memory-mapped external device with DACK	External	Block/Single	8/16/32 16-byte/32-byte	0,1
Memory-mapped external device and memory-mapped external device without DACK	Auto on-chip	Block /Single	8/16/32 16-byte/32-byte	0~7
External memory or memory-mapped external device without DACK and on-chip peripheral module	Auto on-chip	Block/Single	8/16/32 16-byte/32-byte	0~7

## 1.8 Examples

### 1.8.1 Memory-to-memory auto request 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:

- Check if (DMAC.AR==0 && DMAC.HLT==0 && DCS3.AR==0 && DCS3.HLT==0 && DCS3.CT==0 && DTC3==0)
- If above condition is true, set value 0 to DCS3.CTE to disable the channel 3 temporarily
- Set source address 0x20001000 to DSA3 and destination address 0x20011000 to DDA3
- Suppose the data unit is word, set transfer count number 256 (1024/4) to DTC3
- Set auto-request (0x8) to DRT3
- Up to now, only the most important channel control register DCS3 is left, set it carefully:
  - Ignore bit31~27 in the case because there is no external request
  - Set value 1 to SAI and DAI<sup>\*1</sup>
  - Ignore RDIL because in the case there is no explicit request signal can be detected
  - Set word size (0) to SP and DP<sup>\*2</sup>
  - Set single mode (0) to TM<sup>\*3</sup>
  - Set value 1 to TIE to let CPU do some post process after the transfer done
- Set value 1 to DRT3.CTE and DMAC.DMAE to launch the transfer in channels 3
- When the transfer terminates normally (DTC3==0 && DCS3.CT==1), DIRQP.CIRQ3 will automatically be set value 1 and an interrupt request will be sent to CPU
- 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:

- Either source or destination is a FIFO, must not enable corresponding address increment
- When either source or destination 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

3. Block mode may block bus for a long time, do not use the mode unless the data are emergency

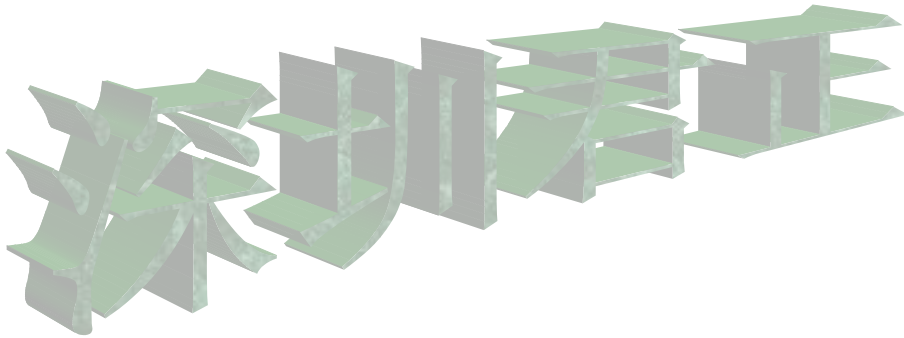




Table 1-1 GPIO group 0 pins summary

Bit N	GPn	Pull (U/D)	Shared Function Pin Selected by GPFSLx/GPFShx (Input/Ouput)		
			01	10	11
0	0	U	CIM_DD0(in) <sup>*1</sup>		
1	1	U	CIM_DD1(in) <sup>*1</sup>		
2	2	U	CIM_DD2(in) <sup>*1</sup>		
3	3	U	CIM_DD3(in) <sup>*1</sup>		
4	4	D	CIM_DD4(in) <sup>*1</sup>		
5	5	D	CIM_DD5(in) <sup>*1</sup>		
6	6	D	CIM_DD6(in) <sup>*1</sup>		
7	7	D	CIM_DD7(in) <sup>*1</sup>		
8	8	D	CIM_VSYNC(in) <sup>*1</sup>		
9	9	U	CIM_HSYNC(in) <sup>*1</sup>		
10	10	D	CIM_PCLK(in) <sup>*1</sup>		
11	11	D	CIM_MCLK(out) <sup>*1</sup>		
12	12	U	DMA_DREQ0(in)		
13	13	U	DMA_DACK0(out)		
14	14				
15	15				
16	16	U	UART3_RxD(in)		
17	17	U	UART3_CTS(in)		
18	18				
19	19				
20	20				
21	21	U	UART3_TxD(out)		
22	22				
23	23	U	UART3_RTS(out)		
24	24	U	UART1_RxD(in)		
25	25	U	UART1_TxD(out)		
26	26	U	DMA_AEN(out)		
27	27	U	DMA_EOP(out)		
28	28	U	USB_CLK(in)		
29	29	U	USB PPWR0(out)		
30	30				
31	31				

Notes:

1. This function pins are available only in jz4730. No function pin shared in jz4720

Table 1-2 GPIO group 1 pins summary

Bit N	GPn	Pull (U/D)	Shared Function Pin Selected by GPFSLx/GPFSHx(Input/Output)		
			01	10	11
0	32	U	PS2_KCLK(io)		
1	33	U	PS2_KDATA(io)		
2	34	U	MSC_DAT0(io) <sup>*1</sup> / PRT_PM_P0(out) <sup>*2</sup>		
3	35	U	MSC_DAT1(io) <sup>*1</sup> / PRT_PM_P(out) <sup>*2</sup>		
4	36	U	MSC_DAT2(io) <sup>*1</sup> / PRT_PM_I0(out) <sup>*2</sup>		
5	37	U	MSC_DAT3(io) <sup>*1</sup> / PRT_PM_I1(out) <sup>*2</sup>		
6	38	U	MSC_CMD(io) <sup>*1</sup> / PRT_CM_P0(out) <sup>*2</sup>		
7	39	U	MSC_CLK(out) <sup>*1</sup> / PRT_CM_P1(out) <sup>*2</sup>		
8	40	U	LCD_D0(out)		
9	41	U	LCD_D1(out)		
10	42	U	LCD_D2(out)		
11	43	U	LCD_D3(out)		
12	44	U	LCD_D4(out)		
13	45	U	LCD_D5(out)		
14	46	U	LCD_D6(out)		
15	47	U	LCD_D7(out)		
16	48	D	LCD_D8(out)		
17	49	D	LCD_D9(out)		
18	50	D	LCD_D10(out)		
19	51	D	LCD_D11(out)		
20	52	D	LCD_D12(out)		
21	53	D	LCD_D13(out)		
22	54	D	LCD_D14(out)		
23	55	D	LCD_D15(out)		
24	56	D	LCD_VSYNC(in)	LCD_VSYNC(out)	
25	57	U	LCD_HSYNC(in)	LCD_HSYNC(out)	
26	58	D	LCD_PCLK(in)	LCD_PCLK(out)	
27	59	D	LCD_DE(out)		
28	60	U	LCD_SPL(out)		
29	61	U	LCD_CLS(out)		
30	62	U	LCD_PS(out)		
31	63	U	LCD_REV(out)		

Notes:

1. This function pins are available only in jz4730
2. This function pins are available only in jz4720



Table 1-3 GPIO group 2 pins summary

Bit N	GPn	Pull (U/D)	Shared Function Pin Selected by GPFSLx/GPFShx (Input/Ouput)		
			01	10	11
0	64	U	SCC0_DAT(io)		
1	65	U	SCC1_DAT(io)		
2	66	U	SCC0_CLK(out)		
3	67	U	SCC1_CLK(out)		
4	68	U	SYS_CLK(out)		
5	69	U	ACRESET_(out)		
6	70	U	SDATA_OUT(out)		
7	71	U	SDATA_IN(in)		
8	72	U	SSI_CLK(out)		
9	73	U	SSI_CE_(out)		
10	74	U	SSI_DT(out)		
11	75	U	SSI_DR(in)		
12	76	U	SSI_CE2_/SSI_GPC(out)		
13	77	U	BITCLK_IN(in)	BITCLK_OUT(out)	
14	78	U	SYNC_IN(in)	SYNC_OUT(out)	
15	79	U	FRE_(out)		
16	80	U	FWE_(out)		
17	81	U	RB_(in)		
18	82	U	DCS1_(out)		
19	83	U	CS1_(out)		
20	84	U	CS2_(out)		
21	85	U	CS3_(out)		
22	86	U	CS4_(out)		
23	87	U	CS5_(out)		
24	88	U	INPACK_(in)		
25	89	U	BVD2(in)		
26	90	U	PCE1_(out)		
27	91	U	PSKTSEL_(out)		
28	92	U	IOIS16_(in)		
29	93	U	PCE2_(out)		
30	94	U	PWM0(out)		
31	95	U	PWM1(out)		

Table 1-4 GPIO group 3 pins summary

Bit N	GPn	Pull (U/D)	Shared Function Pin Selected by GPFSLx/GPFShx (Input/Output)		
			01	10	11
0	96	U	PRT_SOLENOID0(out) <sup>*1</sup>		
1	97	U	PRT_SOLENOID1(out) <sup>*1</sup>		
2	98	U	PRT_SOLENOID2(out) <sup>*1</sup>		
3	99	U	PRT_SOLENOID3(out) <sup>*1</sup>		
4	100	U	PRT_SOLENOID4(out) <sup>*1</sup>		
5	101	U	PRT_SOLENOID5(out) <sup>*1</sup>		
6	102	U	PRT_SOLENOID6(out) <sup>*1</sup>		
7	103	U	PRT_SOLENOID7(out) <sup>*1</sup>		
8	104	U	PRT_CM_I0(out) <sup>*1</sup>		
9	105	U	PRT_CM_I1(out) <sup>*1</sup>		
10	106	U	PRT_POWER(out) <sup>*1</sup>		
11	107	U	PRT_BM_(in) <sup>*1</sup>		
12	108	U	PRT_PE_(in) <sup>*1</sup>		
13	109	U	PRT_CH_(in) <sup>*1</sup>		
14	110	U	PRT_CHP_(in) <sup>*1</sup>		
15	111	U	UART2_RxD(in)		
16	112	U	MII_TX_EN(out)		
17	113	U	MII_RX_DV(in)		
18	114	U	MII_RX_ER(in)		
19	115	U	MII_COL(in)		
20	116	U	MII_CRIS(in)		
21	117	U	MII_TxD0(out)		
22	118	U	MII_TxD1(out)		
23	119	U	MII_TxD2(out)		
24	120	U	MII_TxD3(out)		
25	121	U	MII_RxD0(in)		
26	122	U	MII_RxD1(in)		
27	123	U	MII_RxD2(in)		
28	124	U	MII_RxD3(in)		
29	125	U	UART2_TxD(out)		
30	126	U	UART0_RxD(in)		
31	127	U	UART0_TxD(out)		

Notes:

1. This function pins are available only in jz4720. No function pin shared in jz4730

## 1.2 Register Description

Table 1-5 summarized all memory-mapped registers, which can be programmed to operate GPIO pin and function pin sharing configuration.

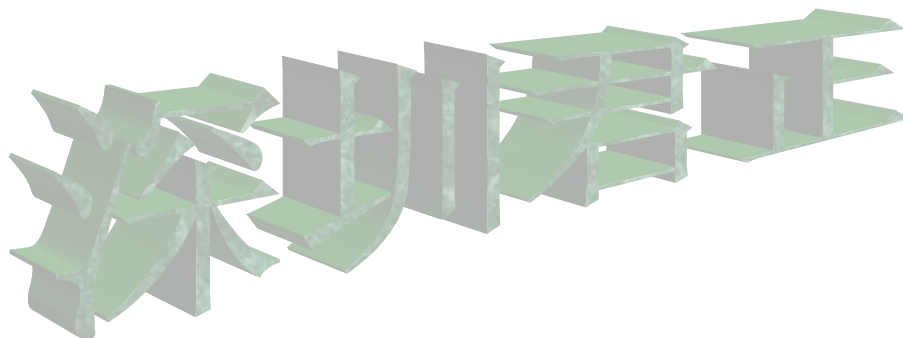
All registers are in 32-bits width. Usually, 1 bit in the register affects a corresponding GPIO pin and every GPIO pin can be operated independently. For function selection registers (GPFSLx/GPFSHx) and interrupt triggering configuring registers (GPITLx/GPITHx), the designator L signifies that the 16 lower half of each group of GPIOs are configured by that register and H designates that the 16 upper half of each group of GPIOs are configured by that register.

The bit 31, 30, 22, 20, 19, 18, 15 and 14 in group 0 GPIO registers, GPD0, GPDIR0 and etc, are not available since there are no these GPIO pins. Writing to these bits is ignored and reading value is undefined.

**Table 1-5 GPIO Registers**

Name	Description	RW	Reset Value	Address	Size
GPD0	GPIO data register 0	RW	0x00000000	0x10010000	32
GPDIR0	GPIO in/out direction register 0	RW	0x00000000	0x10010004	32
GPPE0	GPIO pull up/down enable register 0	RW	0xFFFFFFFF	0x1001000C	32
GPFSL0	GPIO function selection low register 0	RW	0x00000000	0x10010010	32
GPFSH0	GPIO function selection high register 0	RW	0x00000000	0x10010014	32
GPITL0	GPIO interrupt triggering low register 0	RW	0x00000000	0x10010018	32
GPITH0	GPIO interrupt triggering high register 0	RW	0x00000000	0x1001001C	32
GPIE0	GPIO interrupt enable register 0	RW	0x00000000	0x10010020	32
GPIM0	GPIO interrupt mask register 0	RW	0x00000000	0x10010024	32
GPIRQ0	GPIO interrupt flag register 0	RW	0x00000000	0x10010028	32
GPD1	GPIO data register 1	RW	0x00000000	0x10010030	32
GPDIR1	GPIO in/out direction register 1	RW	0x00000000	0x10010034	32
GPPE1	GPIO pull up/down enable register 1	RW	0xFFFFFFFF	0x1001003C	32
GPFSL1	GPIO function selection low register 1	RW	0x00000000	0x10010040	32
GPFSH1	GPIO function selection high register 1	RW	0x00000000	0x10010044	32
GPITL1	GPIO interrupt triggering low register 1	RW	0x00000000	0x10010048	32
GPITH1	GPIO interrupt triggering high register 1	RW	0x00000000	0x1001004C	32
GPIE1	GPIO interrupt enable register 1	RW	0x00000000	0x10010050	32
GPIM1	GPIO interrupt mask register 1	RW	0x00000000	0x10010054	32
GPIRQ1	GPIO interrupt flag register 1	RW	0x00000000	0x10010058	32
GPD2	GPIO data register 2	RW	0x00000000	0x10010060	32
GPDIR2	GPIO in/out direction register 2	RW	0x00000000	0x10010064	32

GPPE2	GPIO pull up/down enable register 2	RW	0xFFFFFFFF	0x1001006C	32
GPFSL2	GPIO function selection low register 2	RW	0x00000000	0x10010070	32
GPFSH2	GPIO function selection high register 2	RW	0x00000000	0x10010074	32
GPITL2	GPIO interrupt triggering low register 2	RW	0x00000000	0x10010078	32
GPITH2	GPIO interrupt triggering high register 2	RW	0x00000000	0x1001007C	32
GPIE2	GPIO interrupt enable register 2	RW	0x00000000	0x10010080	32
GPIM2	GPIO interrupt mask register 2	RW	0x00000000	0x10010084	32
GPIRQ2	GPIO interrupt flag register 2	RW	0x00000000	0x10010088	32
GPD3	GPIO data register 3	RW	0x00000000	0x10010090	32
GPDIR3	GPIO in/out direction register 3	RW	0x00000000	0x10010094	32
GPPE3	GPIO pull up/down enable register 3	RW	0xFFFFFFFF	0x1001009C	32
GPFSL3	GPIO function selection low register 3	RW	0x00000000	0x100100A0	32
GPFSH3	GPIO function selection high register 3	RW	0x00000000	0x100100A4	32
GPITL3	GPIO interrupt triggering low register 3	RW	0x00000000	0x100100A8	32
GPITH3	GPIO interrupt triggering high register 3	RW	0x00000000	0x100100AC	32
GPIE3	GPIO interrupt enable register 3	RW	0x00000000	0x100100B0	32
GPIM3	GPIO interrupt mask register 3	RW	0x00000000	0x100100B4	32
GPIRQ3	GPIO interrupt flag register 3	RW	0x00000000	0x100100B8	32



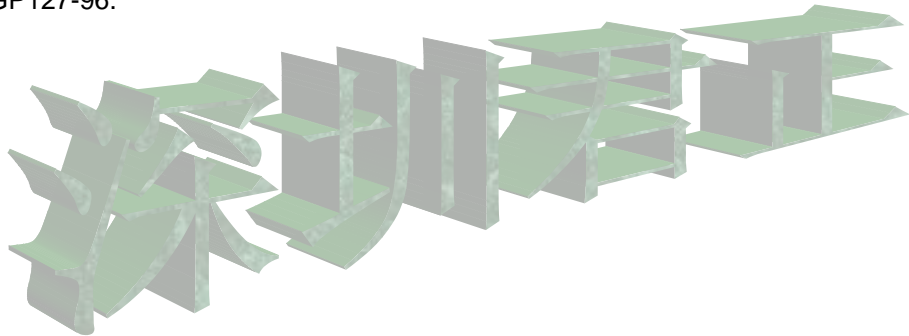
1.2.1 GPIO data register for group 0, 1, 2 and 3 (GPD0, GPD1, GPD2, GPD3)

GPD0, GPD1, GPD2 and GPD3 are four 32-bit GPIO data registers.

GPD0, GPD1, GPD2, GPD3																0x10010000, 0x10010030, 0x10010060, 0x10010090																
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	DATA35	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 <sub>n</sub>	Where n = 0 ~ 31 and DATA <sub>n</sub> = DATA00 ~ DATA31 When writing, the writing value is stored to the register. When reading, if the bit corresponding GPIO is set as GPIO input or interrupt source, or it is set as a function input or io pin, the pin's voltage level is read, else, the previous stored data is read. If the bit corresponding GPIO is set as GPIO output, the GPIO pin voltage level is decided by this bit: 0 for low voltage and 1 for high voltage.	RW

GPD0 bits 31-0 correspond to GPIO pins GP31-0; GPD1 to GP63-32; GPD2 to GP95-64 and GPD3 to GP127-96.



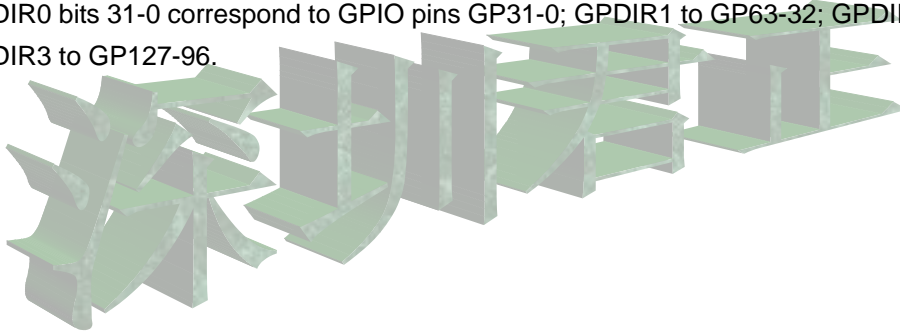
### 1.2.2 GPIO pin direction register for group 0, 1, 2 and 3 (GPDIR0, GPDIR1, GPDIR2, GPDIR3)

GPDIR0, GPDIR1, GPDIR2 and GPDIR3 are four 32-bit GPIO pin in/out direction registers.

GPDIR0, GPDIR1, GPDIR2, GPDIR3																0x1001000c, 0x1001003c, 0x1001006c, 0x1001009c																
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	DIR35	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	

Bits	Name	Description	R/W						
n	DIRn	<p>Where n = 0 ~ 31 and DIRn = DIR00 ~ DIR31</p> <p>The bit controls corresponding GPIO pin's input/output direction</p> <table><tr><th>DIRn value</th><th>Corresponding GPIO pin direction</th></tr><tr><td>0</td><td>Input</td></tr><tr><td>1</td><td>Output</td></tr></table>	DIRn value	Corresponding GPIO pin direction	0	Input	1	Output	RW
DIRn value	Corresponding GPIO pin direction								
0	Input								
1	Output								

GPDIR0 bits 31-0 correspond to GPIO pins GP31-0; GPDIR1 to GP63-32; GPDIR2 to GP95-64 and GPDIR3 to GP127-96.



### 1.2.3 GPIO pin pull enable register for group 0, 1, 2 and 3 (GPPE0, GPPE1, GPPE2, GPPE3)

GPPE0, GPPE1, GPPE2 and GPPE3 are four 32-bit GPIO pin pull up/down enable registers.

GPPE0, GPPE1, GPPE2, GPPE3																0x10010004, 0x10010034, 0x10010064, 0x10010094																
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
	PE31	PE30	PE29	PE28	PE27	PE26	PE35	PE24	PE23	PE22	PE21	PE20	PE19	PE18	PE17	PE16	PE15	PE14	PE13	PE12	PE11	PE10	PE09	PE08	PE07	PE06	PE05	PE04	PE03	PE02	PE01	PE00
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	

Bits	Name	Description	R/W						
n	PEn	<p>Where n = 0 ~ 31 and PEn = PE00 ~ PE31</p> <p>The bit controls whether to enable the corresponding GPIO pin's pull up or pull down resistor</p> <table><tr><th>PEn value</th><th>Corresponding GPIO pin's pull up/down resistor</th></tr><tr><td>0</td><td>No pull up or pull down resistor connects to the pin</td></tr><tr><td>1</td><td>An internal pull up or pull down resistor connects to the pin. Up or down is pin dependence. Please reference to Table 1-1 ~ Table 1-4 for it.</td></tr></table>	PEn value	Corresponding GPIO pin's pull up/down resistor	0	No pull up or pull down resistor connects to the pin	1	An internal pull up or pull down resistor connects to the pin. Up or down is pin dependence. Please reference to Table 1-1 ~ Table 1-4 for it.	RW
PEn value	Corresponding GPIO pin's pull up/down resistor								
0	No pull up or pull down resistor connects to the pin								
1	An internal pull up or pull down resistor connects to the pin. Up or down is pin dependence. Please reference to Table 1-1 ~ Table 1-4 for it.								

GPPE0 bits 31-0 correspond to GPIO pins GP31-0; GPPE1 to GP63-32; GPPE2 to GP95-64 and GPPE3 to GP127-96.

### 1.2.4 GPIO pin share function selection L/H register for group 0, 1, 2 and 3 (GPFSL0/GPFSLH0, GPFSL1/GPFSLH1, GPFSL2/GPFSLH2, GPFSL3/GPFSLH3)

GPFSL0/GPFSLH0, GPFSL1/GPFSLH1, GPFSL2/GPFSLH2 and GPFSL3/GPFSLH3 are eight 32-bit GPIO pin share function selection control registers.

GPFSL0, GPFSL1, GPFSL2, GPFSL3																0x10010010, 0x10010040, 0x10010070, 0x100100a0																																																
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																																
	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	0																															

GPFSLH0, GPFSLH1, GPFSLH2, GPFSLH3																0x10010014, 0x10010044, 0x10010074, 0x100100a4																																																
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			
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	0																															

Bits	Name	Description	R/W										
2n:2n+1	SELn	<p>Where n = 0 ~ 15 for both GPFSLx and GPFSLHx registers and SELn = SEL00 ~ SEL31 when it includes both GPFSLx and GPFSLHx registers</p> <p>In most cases, GPIO pin is shared with one or more peripheral functions. SELn controls the owner of the pin n.</p> <table><tr><th>SELn value</th><th>Pin n owner</th></tr><tr><td>00</td><td>GPIO group x bit n, which can be general-purpose I/O pin or interrupt source</td></tr><tr><td>01</td><td>Function 1 pin<sup>*1</sup></td></tr><tr><td>10</td><td>Function 2 pin<sup>*1</sup></td></tr><tr><td>11</td><td>Function 3 pin<sup>*1</sup></td></tr></table> <p>Note: 1. Please reference to Table 1-1 ~ Table 1-4 for the details</p>	SELn value	Pin n owner	00	GPIO group x bit n, which can be general-purpose I/O pin or interrupt source	01	Function 1 pin <sup>*1</sup>	10	Function 2 pin <sup>*1</sup>	11	Function 3 pin <sup>*1</sup>	RW
SELn value	Pin n owner												
00	GPIO group x bit n, which can be general-purpose I/O pin or interrupt source												
01	Function 1 pin <sup>*1</sup>												
10	Function 2 pin <sup>*1</sup>												
11	Function 3 pin <sup>*1</sup>												

GPFSL0 bits 31-0 correspond to GPIO pins GP15-0; GPFSLH0 to GP31~16; GPFSL1 to GP47-32; GPFSLH1 to GP63-48; GPFSL2 to GP79-64; GPFSLH2 to GP95-80; GPFSL3 to GP111-96 and GPFSLH3 to GP127-112



### 1.2.5 GPIO interrupt triggering configure L/H register for group 0, 1, 2 and 3 (GPITL0/GPITH0, GPITL1/GPITH1, GPITL2/GPITH2, GPITL3/GPITH3)

GPITL0/GPITH0, GPITL1/GPITH1, GPITL2/GPITH2 and GPITL3/GPITH3 are eight 32-bit GPIO interrupt triggering configure control registers.

GPITL0, GPITL1, GPITL2, GPITL3																0x10010018, 0x10010048, 0x10010078, 0x100100a8																																																
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																																
	ITCFG15				ITCFG14				ITCFG13				ITCFG12				ITCFG11				ITCFG10				ITCFG09				ITCFG08				ITCFG07				ITCFG06				ITCFG05				ITCFG04				ITCFG03				ITCFG02				ITCFG01				ITCFG00			
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	0																															

GPITH0, GPITH1, GPITH2, GPITH3																0x1001001c, 0x1001004c, 0x1001007c, 0x100100ac																																																
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																																
	ITCFG31				ITCFG30				ITCFG29				ITCFG28				ITCFG27				ITCFG26				ITCFG25				ITCFG24				ITCFG23				ITCFG22				ITCFG21				ITCFG20				ITCFG19				ITCFG18				ITCFG17				ITCFG16			
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	0																															

Bits	Name	Description	R/W										
2n:2n+1	ITCFGn	<p>Where n = 0 ~ 15 for both GPITLx and GPITHx registers, and ITCFGn = ITCFG00 ~ ITCFG31 when it includes both GPITLx and GPITHx registers.</p> <p>When a GPIO pin is set as an interrupt source, the corresponding ITCFGn is used to decide the interrupt trig type.</p> <table><tr><th>ITCFGn value</th><th>Interrupt Trig Type</th></tr><tr><td>00</td><td>Low-level sensitive</td></tr><tr><td>01</td><td>High-level sensitive</td></tr><tr><td>10</td><td>Falling-edge trig</td></tr><tr><td>11</td><td>Rising-edge trig</td></tr></table>	ITCFGn value	Interrupt Trig Type	00	Low-level sensitive	01	High-level sensitive	10	Falling-edge trig	11	Rising-edge trig	RW
ITCFGn value	Interrupt Trig Type												
00	Low-level sensitive												
01	High-level sensitive												
10	Falling-edge trig												
11	Rising-edge trig												

GPITL0 bits 31-0 correspond to GPIO pins GP15-0; GPITH0 to GP31~16; GPITL1 to GP47-32; GPITH1 to GP63-48; GPITL2 to GP79-64; GPITH2 GP95-80; GPITL3 to GP111-96 and GPITH3 to GP127-112

### 1.2.6 GPIO interrupt enable register for group 0, 1, 2 and 3 (GPIE0, GPIE1, GPIE2, GPIE3)

GPIE0, GPIE1, GPIE2 and GPIE3 are four 32-bit GPIO interrupt enable registers.

GPIE0, GPIE1, GPIE2, GPIE3																0x10010020, 0x10010050, 0x10010080, 0x100100b0																
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
	IE31	IE30	IE29	IE28	IE27	IE26	IE35	IE24	IE23	IE22	IE21	IE20	IE19	IE18	IE17	IE16	IE15	IE14	IE13	IE12	IE11	IE10	IE09	IE08	IE07	IE06	IE05	IE04	IE03	IE02	IE01	IE00
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	IEn	<p>Where n = 0 ~ 31 and IEn = IE00 ~ IE31</p> <p>If a GPIO pin is set as GPIO input, the corresponding IEn = 1 enables the interrupt function for this pin while retains the GPIO input function in the same time.</p> <table><tr><th>IEn value</th><th>Description</th></tr><tr><td>0</td><td>Disable the pin as an interrupt source</td></tr><tr><td>1</td><td>Enable the pin as an interrupt source</td></tr></table>	IEn value	Description	0	Disable the pin as an interrupt source	1	Enable the pin as an interrupt source	RW
IEn value	Description								
0	Disable the pin as an interrupt source								
1	Enable the pin as an interrupt source								

GPIE0 bits 31-0 correspond to GPIO pins GP31-0; GPIE1 to GP63-32; GPIE2 to GP95-64 and GPIE3 to GP127-96.

1.2.7 GPIO interrupt mask register for group 0, 1, 2 and 3 (GPIM0, GPIM1, GPIM2, GPIM3)

GPIM0, GPIM1, GPIM2 and GPIM3 are four 32-bit GPIO interrupt mask registers.

GPIM0, GPIM1, GPIM2, GPIM3																0x10010024, 0x10010054, 0x10010084, 0x100100b4																
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
	IM31	IM30	IM29	IM28	IM27	IM26	IM35	IM24	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16	IM15	IM14	IM13	IM12	IM11	IM10	IM09	IM08	IM07	IM06	IM05	IM04	IM03	IM02	IM01	IM00
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	IMn	<p>Where n = 0 ~ 31 and IMn = IM00 ~ IM31</p> <p>This bit is used to mask the issuing of the interrupt to INTC module from the corresponding GPIO pin set as an interrupt source.</p> <table><tr><th>IMn value</th><th>Description</th></tr><tr><td>0</td><td>Interrupt can be issued</td></tr><tr><td>1</td><td>Interrupt cannot be issued</td></tr></table>	IMn value	Description	0	Interrupt can be issued	1	Interrupt cannot be issued	RW
IMn value	Description								
0	Interrupt can be issued								
1	Interrupt cannot be issued								

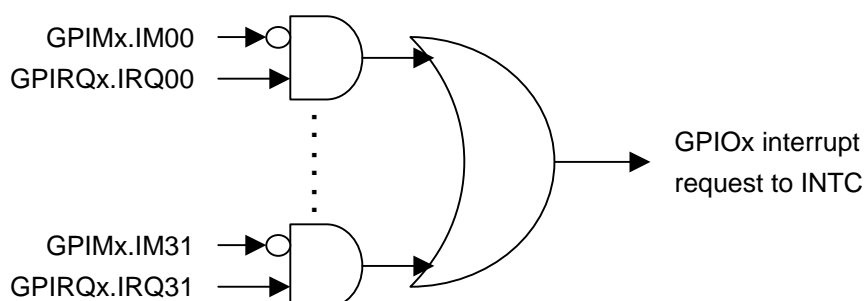
GPIM0 bits 31-0 correspond to GPIO pins GP31-0; GPIM1 to GP63-32; GPIM2 to GP95-64 and GPIM3 to GP127-96.

### 1.2.8 GPIO interrupt request register for group 0, 1, 2 and 3 (GPIRQ0, GPIRQ1, GPIRQ2, GPIRQ3)

GPIRQ0, GPIRQ1, GPIRQ2 and GPIRQ3 are four 32-bit GPIO interrupt request status registers.

GPIRQ0, GPIRQ1, GPIRQ2, GPIRQ3																0x10010028, 0x10010058, 0x10010088, 0x100100b8																
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
	IRQ31	IRQ30	IRQ29	IRQ28	IRQ27	IRQ26	IRQ35	IRQ24	IRQ23	IRQ22	IRQ21	IRQ20	IRQ19	IRQ18	IRQ17	IRQ16	IRQ15	IRQ14	IRQ13	IRQ12	IRQ11	IRQ10	IRQ09	IRQ08	IRQ07	IRQ06	IRQ05	IRQ04	IRQ03	IRQ02	IRQ01	IRQ00
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	

Bits	Name	Description	R/W						
n	IRQn	<p>Where n = 0 ~ 31 and IRQn = IRQ00 ~ IRQ31</p> <p>This bit is used to reflect the interrupt status of the corresponding GPIO pin set as an interrupt source. Reading 1 means an interrupt is pending. If the corresponding GPIOMx.IMn is 0 (interrupt is not masked), the interrupt will be issued to INTC (chip interrupt controller). All 32 interrupt sources' requests are aggregated to send 1 interrupt to INTC. There totally 4 interrupt request sources in INTC from GPIO. Figure 1-1 illustrate the logic.</p> <p>This bit is read only if the corresponding GPIO pin is set as level sensitive interrupt trig type. If it is set as edge trig type, writing 1 to the bit will clear it to 0 and writing 0 is ignored.</p> <table><tr><th>IRQn value</th><th>Description</th></tr><tr><td>0</td><td>Interrupt is not found</td></tr><tr><td>1</td><td>Interrupt is pending</td></tr></table>	IRQn value	Description	0	Interrupt is not found	1	Interrupt is pending	RW
IRQn value	Description								
0	Interrupt is not found								
1	Interrupt is pending								



**Figure 1-1 GPIO group x interrupt request logic diagram**

GPIRQ0 bits 31-0 correspond to GPIO pins GP31-0; GPIRQ1 to GP63-32; GPIRQ2 to GP95-64 and GPIRQ3 to GP127-96.

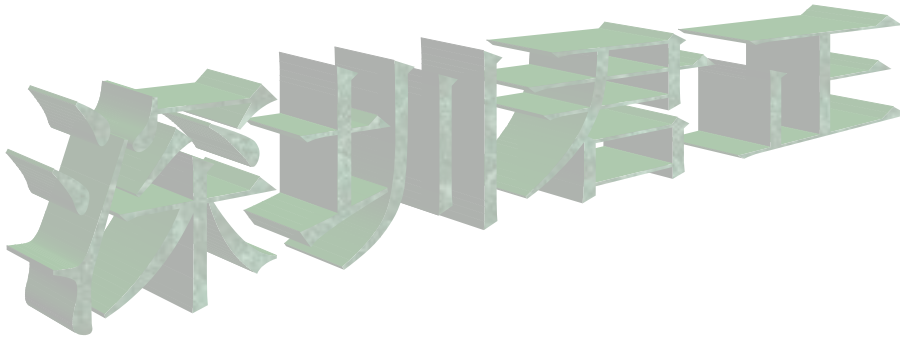
# 1 Pulse Width Modulator

## 1.1 Overview

The Pulse Width Modulator (PWM) is used to control the back light inverter or adjust bright or contrast of LCD panel. PWM consists of a simple free-running counter with two compared registers, each compare register performs a particular task when it matches the count value. The period comparator causes the output pin to be set and the free-running counter to reset when it matches the period value. The width comparator causes the output pin to reset when the counter value matches.

PWM has the following features:

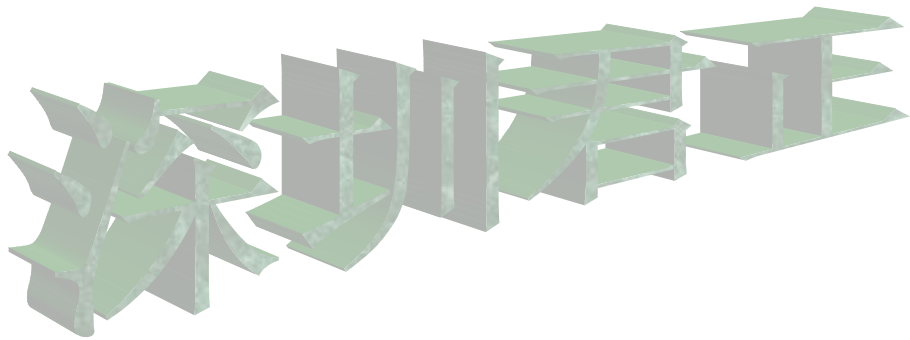
- Enhanced period control through a 6-bit clock divider and a 10-bit period counter
- 10-bit pulse control



1.2 Pin Description

Table 1-1 PWM Pins Description

Name	I/O	Description
PWM [1:0]	Output	PWM channel output signals.

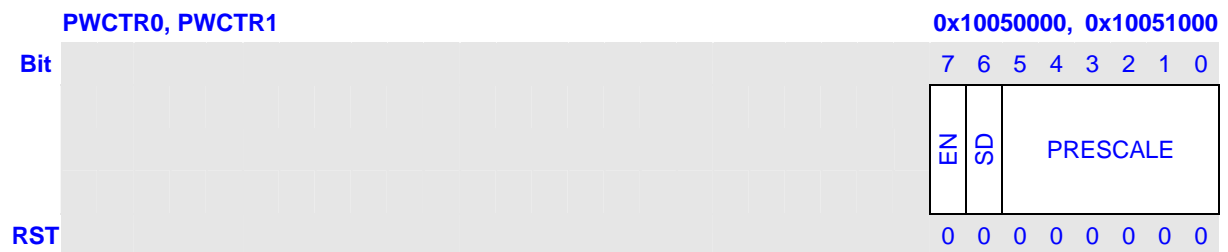


### 1.3 Register Description

Table 1-2 PWM Registers Description

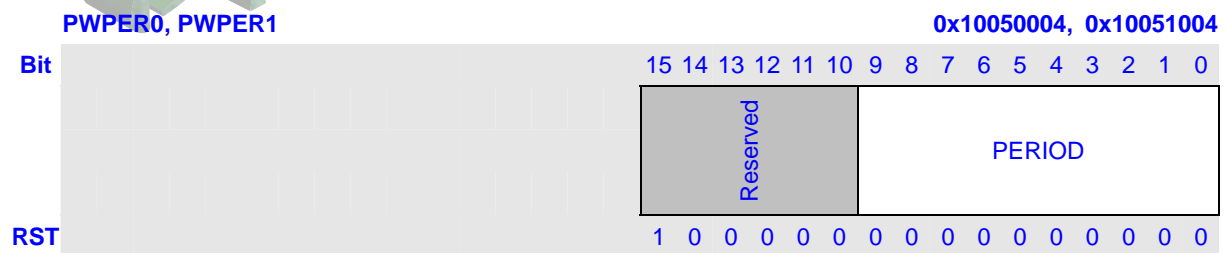
Name	RW	Reset Value	Address	Access Size
PWMCTR0	RW	0x00	0x10050000	8
PWMPER0	RW	0x0004	0x10050004	16
PWMDUT0	RW	0x0000	0x10050008	16
PWMCTR1	RW	0x00	0x10051000	8
PWMPER1	RW	0x0004	0x10051004	16
PWMDUT1	RW	0x0000	0x10051008	16

#### 1.3.1 PWM Control Register (PWCTR0,1)



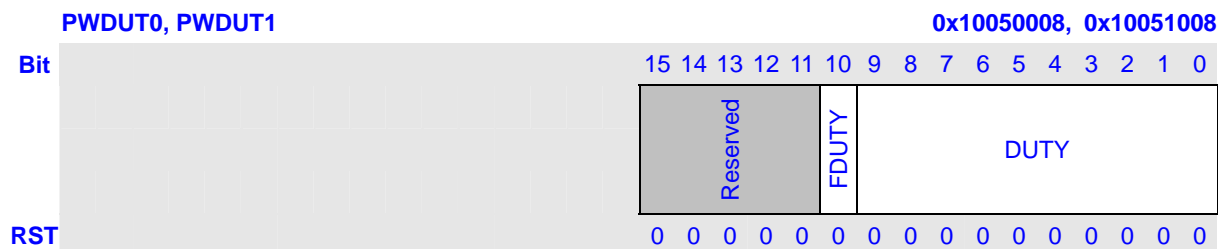
Bits	Name	Description	RW
7	EN	PWM enable. 0 – The PWM is disabled. 1 – The PWM is enabled.	RW
6	SD	PWM Shut Down mode. 0 – Graceful shutdown. 1 – Abrupt shutdown.	RW
5:0	PRESCALE	PWM clock Prescale. 0 – PWCLK_PWM = 3.6864MHz. N – PWCLK_PWM = 3.6864MHz / (N + 1).	RW

#### 1.3.2 PWM Period Register (PWPER0,1)



Bits	Name	Description	RW
15:10	Reserved	Reserved bits. Writes to these bits have no effect and always read as 0.	R
9:0	PERIOD	PWM period control.	RW

### 1.3.3 PWM DUTY Register (PWDUT0,1)



Bits	Name	Description	RW
15:11	Reserved	Reserved bits. Writes to these bits have no effect and always read as 0.	R
10	FDUTY	PWM Full duty cycles control. 0 – PWM_OUT duty cycle determined by DUTY field. 1 – PWM_OUT is set high and does not toggle (Full Duty).	RW
9:0	DUTY	PWM duty cycles control.	RW

## 1.4 PWM Operation

The PWM unit is clocked at the 3.6864 MHz oscillator output. Each channel of Pulse Width Modulator Unit (PWM) are controlled by three registers:

- Pulse Width Control Register (PWCTR)
- Period Control Register (PWPER)
- Duty Cycle Control Register (PWDUT)

By setting the values in these registers the PWM unit produces a pulse width modulated output signal. Each register contains one or more fields which determine an attribute of the PWM\_OUT waveform. PWCTR [PRESCALE] specifies the divisor for the PWM module clock. Note that the actual PWM module clock divisor used is 1 greater than the value programmed into PWCTR [PRESCALE]. This divided PWM module clock drives a 10 bits up counter. This up counter feeds 2 separate comparators. The first comparator contains the value of PWDUT [DUTY]. When the values match, the PWM\_OUT signal is set high. The other comparator contains PWPER [PERIOD] and clears the PWM\_OUT signal low when PWPER [PERIOD] + 1 and the 10-bit up counter are equal. Both PWPER [PERIOD] and PWDUT [DUTY] are 10 bit fields.

**Notes:** Take care to ensure that the value of the PWPER register remain larger than PWDUT register. In the case where PWPER is less than PWDUT the output maintains a high state.

### 1.4.1 Reset Sequence and Power Management Requirements

A system resets results in no pulse width modulated signal. During system reset the PWCTR and PWDUT registers are reset to 0x0 and the PWPER register is set to 0x004. This sets the



PWM\_OUT pin low with a zero duty cycle. The six bits down-counter is reset to 0x0 and thus the 3.6864 MHz input clock directly drives the 10 bit up-counter. The PWM\_OUT pin remains reset low until the PWDUT register is programmed with a non-zero value. PWM may be disabled through a pair of clock enable bits in the CPM module. If the clock is disabled, PWM stops immediately.

### 1.4.2 PWM Output Example

Figure 1-1 is an example of the PWM output for reference.

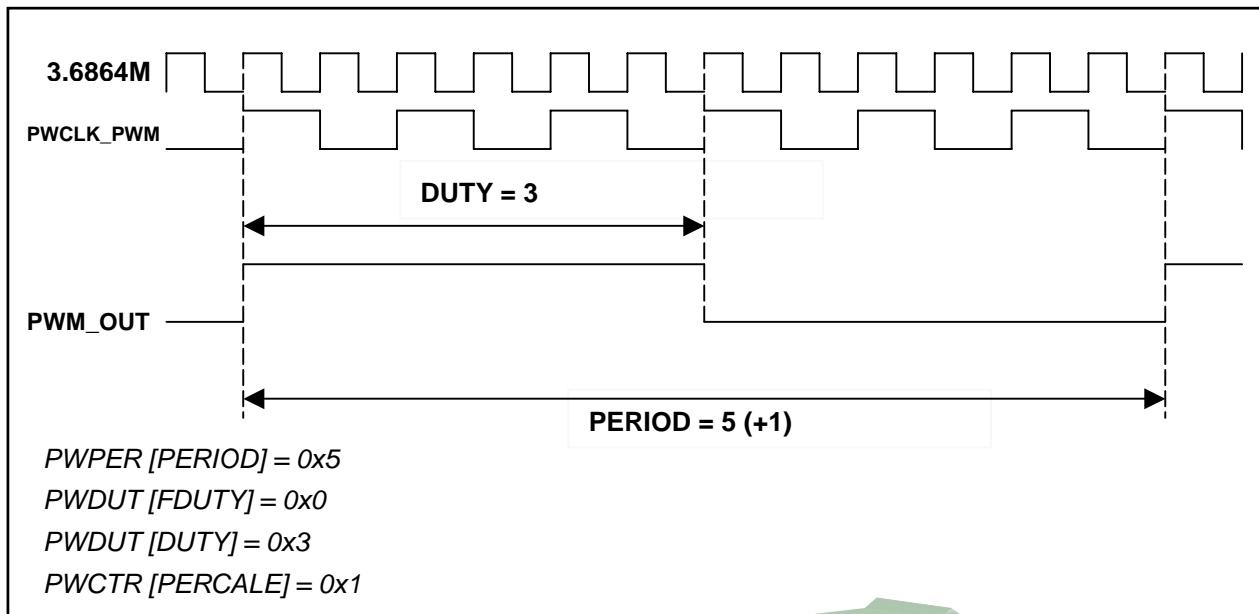


Figure 1-1 An Example of PWM Output

# 1 USB Host Controller

## 1.1 Overview

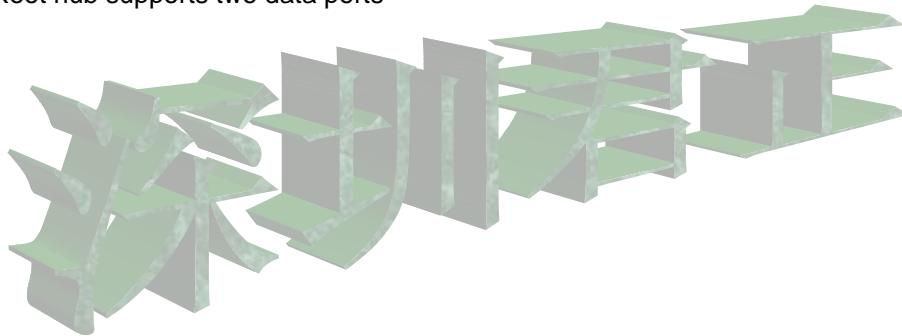
This chapter describes the Universal Serial Bus host controller (UHC) implemented in the JZ47XX Processor .

The Universal Serial Bus (USB) supports serial data exchanges between a host computer and a variety of simultaneously accessible peripherals. The attached peripherals share USB bandwidth through a host-scheduled, token-based protocol. Peripherals can be attached, configured, used, and detached, while the host and other peripherals continue operation.

Familiarity with the *Universal Serial Bus Specification*, Revision 1.1 and the OHCI specification are necessary to fully understand the material contained in this section

Features :

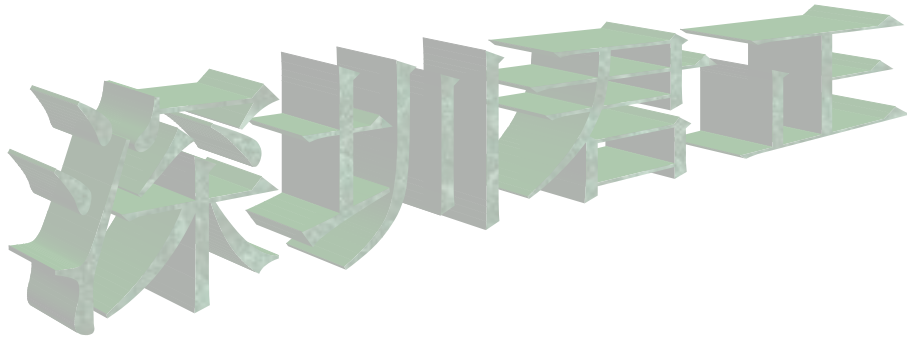
- USB Rev. 1.1 compatible
- Supports both low-speed and full-speed USB devices
- Open Host Controller Interface (OHCI) Rev 1.0 compatible
- Root hub supports two data ports



1.2 Pin Description

Table 1-1 UHC Pins Description

Name	Type	Description
DPLS0	Inout	Data Positive to Port 0
DPLS1	Inout	Data Positive to Port 1
DMNS0	Inout	Data Minus to Port 0
DMNS1	Inout	Data Minus to Port 1



### 1.3 Register Description

The Host Controller (HC) contains a set of on-chip operational registers which are mapped into a noncacheable portion of the system addressable space. These registers are used by the Host Controller Driver (HCD). According to the function of these registers, they are divided into four partitions, specifically for Control and Status, Memory Pointer, Frame Counter and Root Hub. All of the registers should be read and written as words.

Register Name	Description	RW	Reset Value	Address	Access Size
HcRevision	Control and Status group	R	0x00000010	0x13030000	32
HcControl		RW	0x00000000	0x13030004	32
HcCommandStatus		RW	0x00000000	0x13030008	32
HcInterruptStatus		RW	0x00000000	0x1303000C	32
HcInterruptEnable		RW	0x00000000	0x13030010	32
HcInterruptDisable		RW	0x00000000	0x13030014	32
HcHCCA	Memory pointer group	RW	0x00000000	0x13030018	32
HcPeriodCurrentED		R	0x00000000	0x1303001C	32
HcControlHeadED		RW	0x00000000	0x13030020	32
HcControlCurrentED		RW	0x00000000	0x13030024	32
HcBulkHeadED		RW	0x00000000	0x13030028	32
HcBulkCurrentED		RW	0x00000000	0x1303002C	32
HcDoneHead	Frame counter group	R	0x00000000	0x13030030	32
HcFmInterval		RW	0x00002EDF	0x13030034	32
HcFmRemaining		R	0x00000000	0x13030038	32
HcFmNumber		R	0x00000000	0x1303003C	32
HcPeriodicStart		RW	0x00000000	0x13030040	32
HcLSThreshold		RW	0x00000628	0x13030044	32
HcRhDescriptorA	Root hub group	R/W	0x02000902	0x13030048	32
HcRhDescriptorB		RW	0x00060000	0x1303004C	32
HcRhStatus		RW	0x00000000	0x13030050	32
HcRhPortStatus 1		RW	0x00000100	0x13030054	32
HcRhPortStatus 2		RW	0x00000100	0x13030058	32

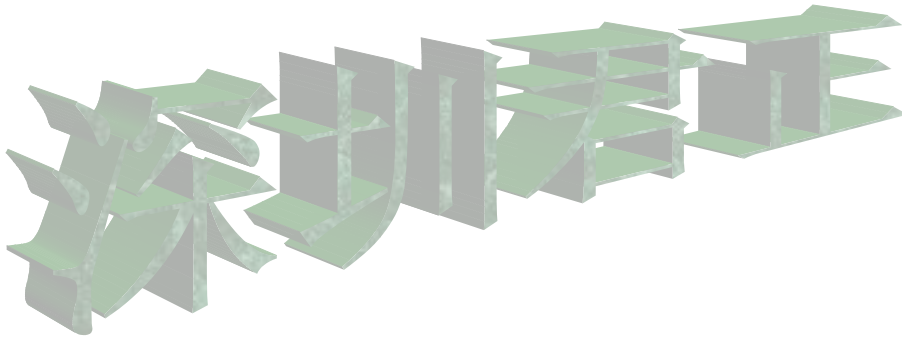
**Notes:** Open HCI – Open Host Controller Specification for USB for details of the each register.

## 1.4 Introduction

The Host Controller is the device which is located between the USB bus and the Host Controller Driver in the OpenHCI architecture. The Host Controller is charged with processing all of the Data Type lists built by the Host Controller Driver. Additionally, the USB Root Hub is attached to the Host Controller.

The main functions as following:

- **USB States** : the Host Controller Operation with respect to the possible USB Bus states.
- **Frame Management** : all aspects of managing the 1-ms USB Frame.
- **List Processing** : the main function of the Host Controller. the detailed processing of the HCD-built Data Type lists.
- **Interrupt Processing** : the interrupt events tracked by the Host Controller and how the Host Controller provides interrupts for those events.
- **Root Hub** : the Root Hub support.



# 1 USB Device Controller

## 1.1 Overview

USB device controller (UDC) is revision 1.1-compliant full-speed device.

### FEATURE

- Fully compliant with USB Specification Version 1.1
- Full speed (12Mbps) device
- Integrated USB Transceiver
- Supports control, interrupt and bulk transfer
- Supports suspend and remote wake-up function
- Supports one configuration, 3 interfaces, 8 physical endpoints (Endpoint 0, 1, 2, 3, 4, 5, 6, 7) and 9 logic endpoints (IN Endpoint 0, 1, 2, 3, 4 OUT Endpoint 0, 5, 6, 7)

### Note:

USB device port is multiplexed with USB host port 0 and is selected by SBA\_CNTL.USBCFG register.

Table 1-1 Endpoint List

EP Number	EP Transfer Type	Direction	Endpoint MaxPktSize
0	Control	Bi-direction	8/16/32 bytes
1	Interrupt	IN	8/16/32/64 bytes
2	Bulk	IN	8/16/32/64 bytes
3	Bulk	IN	8/16/32/64 bytes
4	Isochronous	IN	8/16/32/64 bytes
5	Bulk	OUT	8/16/32/64 bytes
6	Bulk	OUT	8/16/32/64 bytes
7	Isochronous	OUT	8/16/32/64 bytes

## 1.2 Register Description

All UDC register 32bit access address is UDC physical address.

**Table 1-2 UDC Registers Description**

Name	Description	RW	Reset Value	Address	Acc. Size
UICR0	Control Register – IN Endpoint 0	RW	0x00000000	0x13040000	32
UISR0	Status Register – IN Endpoint 0	RW	0x00000000	0x13040004	32
UIBSR0	Buffer Size Register – IN Endpoint 0	RW	0x00000000	0x13040008	32
UIMPR0	Max Packet Size Register – IN Endpoint 0	RW	0x00000000	0x1304000C	32
UICR1	Control Register – IN Endpoint 1	RW	0x00000000	0x13040020	32
UISR1	Status Register – IN Endpoint 1	RW	0x00000000	0x13040024	32
UIBSR1	Buffer Size Register – IN Endpoint 1	RW	0x00000000	0x13040028	32
UIMPR1	Max Packet Size Register – IN Endpoint 1	RW	0x00000000	0x1304002C	32
UICR2	Control Register – IN Endpoint 2	RW	0x00000000	0x13040040	32
UISR2	Status Register – IN Endpoint 2	RW	0x00000000	0x13040044	32
UIBSR2	Buffer Size Register – IN Endpoint 2	RW	0x00000000	0x13040048	32
UIMPR2	Max Packet Size Register – IN Endpoint 2	RW	0x00000000	0x1304004C	32
UICR3	Control Register – IN Endpoint 3	RW	0x00000000	0x13040060	32
UISR3	Status Register – IN Endpoint 3	RW	0x00000000	0x13040064	32
UIBSR3	Buffer Size Register – IN Endpoint 3	RW	0x00000000	0x13040068	32
UIMPR3	Max Packet Size Register – IN Endpoint 3	RW	0x00000000	0x1304006C	32
UICR4	Control Register – IN Endpoint 4	RW	0x00000000	0x13040080	32
UISR4	Status Register – IN Endpoint 4	RW	0x00000000	0x13040084	32
UIBSR4	Buffer Size Register – IN Endpoint 4	RW	0x00000000	0x13040088	32
UIMPR4	Max Packet Size Register – IN Endpoint 4	RW	0x00000000	0x1304008C	32
UOCR0	Control Register – OUT Endpoint 0	RW	0x00000000	0x13040200	32
UOSR0	Status Register – OUT Endpoint 0	RW	0x00000000	0x13040204	32
UOFNR0	Packet Frame number Register – OUT Endpoint 0	RW	0x00000000	0x13040208	32
UOMPR0	Max Packet Size Register – OUT Endpoint 0	RW	0x00000000	0x1304020C	32
UOBPR0	Setup Buffer Pointer – OUT Endpoint 0	RW	0x00000000	0x13040210	32
UOCR5	Control Register – OUT Endpoint 5	RW	0x00000000	0x130402A0	32
UOSR5	Status Register – OUT Endpoint 5	RW	0x00000000	0x130402A4	32

UOFNR5	Packet Frame number Register – OUT Endpoint 5	RW	0x00000000	0x130402A8	32
UOMPR5	Max Packet Size Register – OUT Endpoint 5	RW	0x00000000	0x130402AC	32
UOCR6	Control Register – OUT Endpoint 6	RW	0x00000000	0x130402C0	32
UOSR6	Status Register – OUT Endpoint 6	RW	0x00000000	0x130402C4	32
UOFNR6	Packet Frame number Register – OUT Endpoint 6	RW	0x00000000	0x130402C8	32
UOMPR6	Max Packet Size Register – OUT Endpoint 6	RW	0x00000000	0x130402CC	32
UOCR7	Control Register – OUT Endpoint 7	RW	0x00000000	0x130402E0	32
UOSR7	Status Register – OUT Endpoint 7	RW	0x00000000	0x130402E4	32
UOFNR7	Packet Frame number Register – OUT Endpoint 7	RW	0x00000000	0x130402E8	32
UOMPR7	Max Packet Size Register – OUT Endpoint 7	RW	0x00000000	0x130402EC	32
UDCFGR	UDC Device Configuration Register	RW	0x00000000	0x13040400	32
UDCR	UDC Device Control Register	RW	0x00000000	0x13040404	32
UDSR	UDC Device Status Register	RW	0x00000000	0x13040408	32
UDIPR	UDC Device Interrupt Pending Register	RW	0x00000000	0x1304040C	32
UDIMR	UDC Device Interrupt Mask Register	RW	0x00000000	0x13040410	32
UIPR	Interrupt Pending Register	RW	0x00000000	0x13040414	32
UIMR	Interrupt Mask Register	RW	0x00000000	0x13040418	32
USCAR	Setup Command Address register	RW	0x00000000	0x13040500	32
UINFR0	Information register – Endpoint 0	RW	0x00000000	0x13040504	32
UINFR1	Information register – Endpoint 1	RW	0x00000000	0x13040508	32
UINFR2	Information register – Endpoint 2	RW	0x00000000	0x1304050C	32
UINFR3	Information register – Endpoint 3	RW	0x00000000	0x13040510	32
UINFR4	Information register – Endpoint 4	RW	0x00000000	0x13040514	32
UINFR5	Information register – Endpoint 5	RW	0x00000000	0x13040518	32
UINFR6	Information register – Endpoint 6	RW	0x00000000	0x1304051C	32
UINFR7	Information register – Endpoint 7	RW	0x00000000	0x13040520	32
RFIFO	Out Buffer	R	0x????????	0x13040800	32
TFIFO0	IN Buffer 0 for Endpoint 0	W	0x????????	0x13040840	32
TFIFO1	IN Buffer 1 for Endpoint 1	W	0x????????	0x13040840 + [UIBSR0]*4	32
TFIFO2	IN Buffer 2 for Endpoint 2	W	0x????????	0x13040840 +[UIBSR0]*4 +[UIBSR1]*4	32
TFIFO3	IN Buffer 3 for Endpoint 3	W	0x????????	0x13040840 + [UIBSR0]*4	32



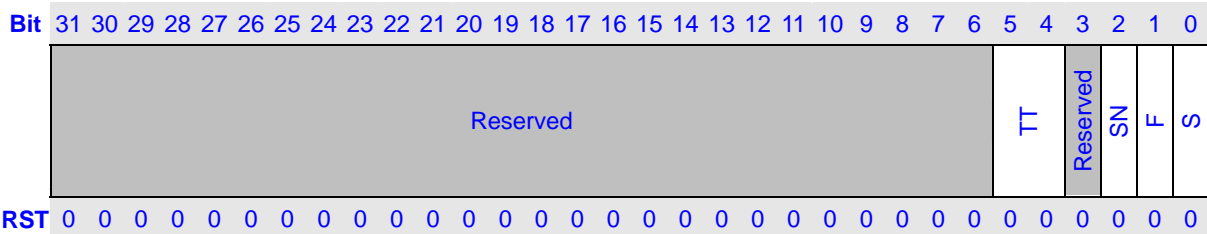
				+ [UIBSR1]*4 + [UIBSR2]*4	
TFIFO4	IN Buffer 4 for Endpoint 4	W	0x????????	0x13040840 + [UIBSR0]*4 + [UIBSR1]*4 + [UIBSR2]*4 + [UIBSR3]*4	32
TCFM	TFIFO Data Confirm	W	0x????????	0x1304041C	32
TZERO	Transmit Zero Packet	W	0x????????	0x13040420	32

## 1.2.1 UDC Endpoint Registers

### 1.2.1.1 Control Register (UICRn, n = 0, 1, 2, 3, 4, UOCRn, n = 0, 5, 6, 7)

UDC control register is used to define each endpoint transfer type and control. Except Endpoint 0 that is a bi-directional endpoint and has two control registers for each direction, each endpoint has one control register.

UICR0, UICR1, UICR2, 0x13040000, 0x13040020, 0x13040040,  
UICR3, UICR4, UOCR0, 0x13040060, 0x13040080, 0x13040200,  
UOCR5, UOCR6, UOCR7 0x130402A0, 0x130402C0, 0x130402E0

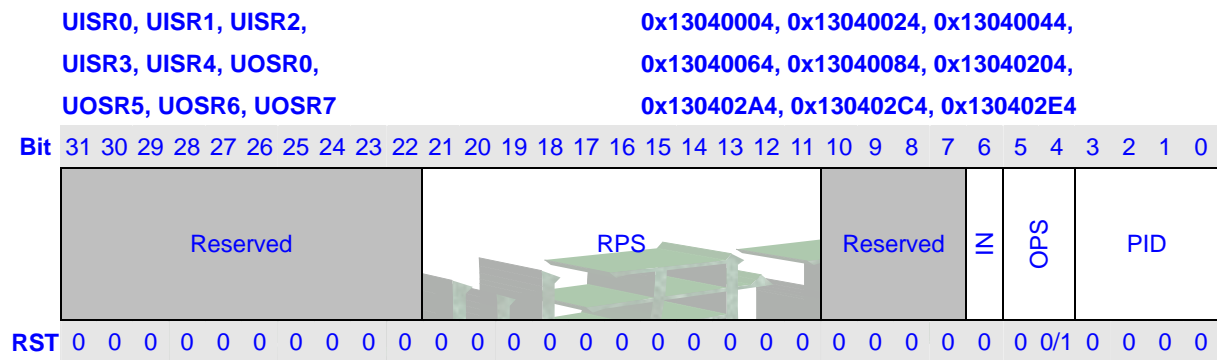


Bits	Name	Description	RW
31:6	Reserved	Always read 0, should only be written with 0	R
5:4	TT	<b>Endpoint Transfer Type</b> Indicate transfer type of endpoint. <b>Note:</b> Software should set the transfer type for each endpoint according to Table 1-1. 0b00 = Control 0b01 = ISO 0b10 = Bulk 0b11 = Interrupt	RW
3	Reserved	Always read 0, should only be written with 0	R
2	SN	<b>Snoop Mode</b> Set when UDC will not check the Correctness of the OUT packet before transferring it to system memory. This bit is ignored by IN endpoint. 0 = Non-Snoop Mode	RW

		1 = Snoop Mode	
1	F	<b>Flush</b> Set when UDC will flush the endpoint IN data FIFO. This bit is ignored by OUT endpoint. 0 = Not flush endpoint IN data FIFO 1 = Flush endpoint IN data FIFO	RW
0	S	<b>Stall</b> Set so that any request from the USB host will be stalled. 0 = Not stall USB host request 1 = Stall USB host request	RW

### 1.2.1.2 Status Register (UISRn, n = 0, 1, 2, 3, 4, UOSRn, n = 0, 5, 6, 7)

UDC Status Register is used to determine each endpoint status. Except Endpoint 0 that is a bi-directional endpoint and has two status registers for each direction, each endpoint has one status register.



Bits	Name	Description	RW
31:22	Reserved	Always read 0, should only be written with 0	R
21:11	RPS	<b>Receive Packet Size</b> The 12-bit field indicates number of bytes received for the current packet by the endpoint. They are ignored by IN endpoint. Software writes 0 to clear the status register.	RW
10:7	Reserved	Always read 0, should only be written with 0	R
6	IN	<b>IN Packet Status</b> Set to indicate that an IN token is received by this endpoint. It is ignored by OUT endpoint. Software writes 0 to clear the status register. 0 = No IN token received 1 = IN token received	RW
5:4	OPS	<b>OUT Packet Status</b> Set to indicate that an OUT packet status received by this endpoint. They are ignored by IN endpoint. The initial value for OUT endpoint is 2'b01 and for IN endpoint is 2'b00. Software writes 0 to clear the status	RW

		register. 0b00 = None OUT Packet received 0b01 = OUT Data received 0b10 = Setup Data (8 bytes) received 0b11 = Reserved	
3:0	PID	<b>PID</b> The read-only PID of the received Packet can be used for ISO OUT transactions where the CPU decides whether it missed any ISO Packets in the current frame. They are ignored by IN endpoint.	R

### 1.2.1.3 Buffer Size Register for IN / Packet Frame Number for OUT (UIBSRn, n = 0, 1, 2, 3, 4, UOFNRn, n = 0, 5, 6, 7)

UDC Buffer Size register is used to hold IN endpoint buffer size. UDC Receive Packet Frame Number Register is used to indicate the frame number of the packet received for OUT endpoint.

For IN endpoint:

UIBSR0, UIBSR1, UIBSR2, UIBSR3, UIBSR4																0x13040008, 0x13040028, 0x13040048, 0x13040068, 0x13040088																
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																BS																
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	Always read 0, should only be written with 0	R
15:0	BS	<b>Buffer Size</b> It is used to define the buffer size for each endpoint. Buffer size must be equal to Max Packet Size divided by 4 that means the 32-bit entry number in FIFO. They are ignored by OUT endpoint.	RW

For OUT endpoint:

UOFNR0, UOFNR5, UOFNR6, UOFNR7																0x13040208, 0x130402A8, 0x130402C8,0x130402E8																
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																FN															
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	

Bits	Name	Description	RW
31:14	Reserved	Always read 0, should only be written with 0	R
13:0	FN	<b>Frame Number</b> The read-only field is used to indicate the frame number of the packet received and may be mainly useful for ISO transaction. They are ignored by IN endpoint.	R

#### 1.2.1.4 MaxPktSize Register (UIMPRn, n = 0, 1, 2, 3, 4, UOMPRn, n = 0, 5, 6, 7)

UDC MaxPktSize Register is used to define the maximum packet size for each endpoint.

UIMPR0, UIMPR1, UIMPR2, 0x1304000C, 0x1304002C, 0x1304004C,  
 UIMPR3, UIMPR4, UOMPR0, 0x1304006C, 0x1304008C, 0x1304020C,  
 UOMPR5, UOMPR6, UOMPR7 0x130402AC, 0x130402CC, 0x130402EC

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																MPS															
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	Always read 0, should only be written with 0	R
15:0	MPS	<b>Maximum Packet Size</b> It is used to define maximum packet size in bytes for each endpoint. When this field is changed, the corresponding buffer size may need to be changed too. Be careful to set the value according to Table 1-1 for each endpoint.	RW

#### 1.2.1.5 Interrupt Pending Register (UIPR)

UDC interrupt pending register is used to indicate the interrupt status for each endpoint.

UIPR 0x13040414

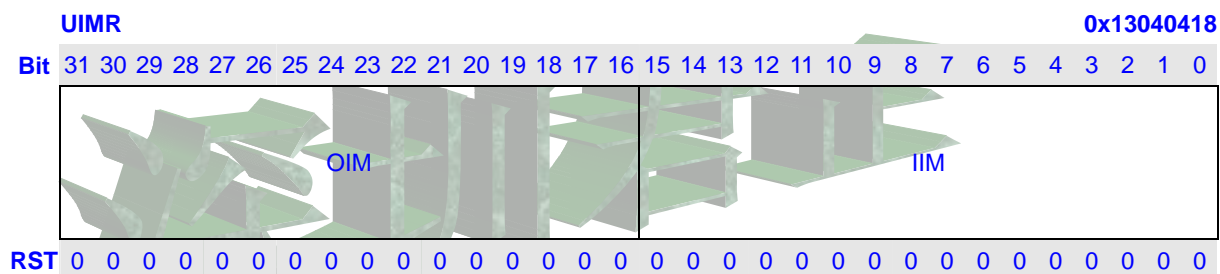
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	OIP																IIP															
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
------	------	-------------	----

31:16	OIP	<b>Interrupt Pending for OUT Endpoint</b> Each bit of the field stands for the interrupt pending status of the corresponding OUT endpoint. For example, bit 5/6/7 is for endpoint 5/6/7 respectively, and bit 0 is for endpoint 0. Other bits reserved and always read as 0. Software writes 1'b1 to corresponding bit to clear that interrupt pending bit to 0. 0 = No interrupt pending 1 = Interrupt pending	RW
15:0	IIP	<b>Interrupt Pending for IN Endpoint</b> Each bit of the field stands for the interrupt pending status of the corresponding IN endpoint. For example, bit 1/2/3/4 is for endpoint 1/2/3/4 respectively, and bit 0 is for endpoint 0. Other bits reserved and always read as 0. Software writes 1'b1 to corresponding bit to clear that interrupt pending bit to 0. 0 = No interrupt pending 1 = Interrupt pending	RW

### 1.2.1.6 Interrupt Mask Register (UIMR)

UDC Interrupt Mask Register is used to mask corresponding endpoint pending interrupts in UIPR.

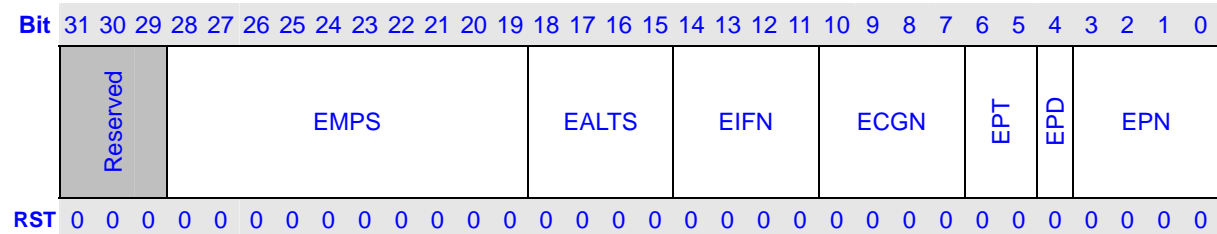


Bits	Name	Description	RW
31:16	OIM	<b>Interrupt Mask for OUT Endpoint</b> Each bit of the field is used to mask the pending interrupt in the corresponding bit in UIPR.OIP. 0 = Non-mask 1 = Mask the pending interrupt	RW
15:0	IIM	<b>Interrupt Mask for IN Endpoint</b> Each bit of the field is used to mask the pending interrupt in the corresponding bit in UIPR.IIP. 0 = Non-mask 1 = Mask the pending interrupt	RW

### 1.2.1.7 Information Register (UINFRn, n = 0, 1, 2, 3, 4, 5, 6, 7)

UDC information register contains the information for each endpoint.

UINFR0, UINFR1, UINFR2, 0x13040504, 0x13040508, 0x1304050C,  
 UINFR3, UINFR4, UINFR5, 0x13040510, 0x13040514, 0x13040518,  
 UINFR6, UINFR7 0x1304051C, 0x13040520



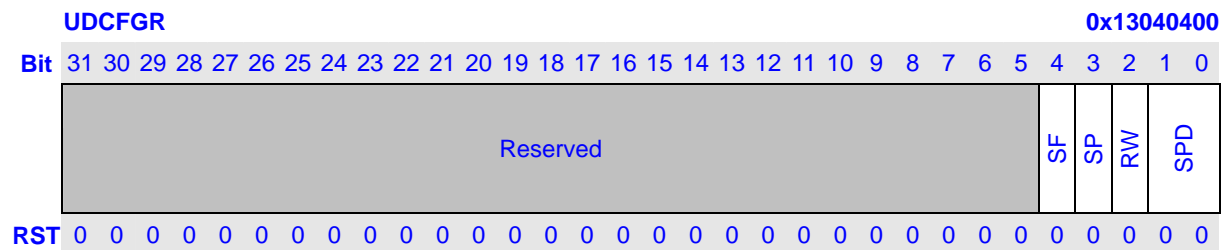
Bits	Name	Description	RW
31:29	Reserved	Always read 0, should only be written with 0	R
28:19	EMPS	<b>Endpoint Maximum Packet Size</b> Software should set this field equal to that in the corresponding MaxPktSize register for each endpoint. For example, UINFR0.EMPS = UIMPR0.MPS; UINFR5.EMPS = UOMPR5.MPS.	RW
18:15	EALTS	<b>Endpoint Alternate Setting</b> This field stands for alternate setting that this endpoint belongs to. Software should write 4'h0 to the field because UDC only supports one alternate setting.	RW
14:11	EIFN	<b>Endpoint Interface Number</b> This field stands for interface number that this endpoint belongs to. Software should write 4'h0 or 4'h1 or 4'h2 to the field because UDC only supports three interfaces.	RW
10:7	ECGN	<b>Endpoint Configuration Number</b> This field stands for configuration number that this endpoint belongs to. Software should write 4'h1 to the field because UDC only supports one configuration.	RW
6:5	EPT	<b>Endpoint Type</b> This field stands for endpoint transfer type that this endpoint belongs to. Software should write the value to each endpoint according to Table 1-1. And it is equal to that in the corresponding UDC control register. For example, UINFR0.EPT = UICR0.TT. 0b00 = Control 0b01 = Isochronous 0b10 = Bulk 0b11 = Interrupt	RW
4	EPD	<b>Endpoint Direction</b> This bit stands for endpoint direction that this endpoint belongs to. Software should write the value to each endpoint according to Table	RW

		1-1. 0 = OUT Endpoint 1 = IN Endpoint	
3:0	EPN	<b>Endpoint Number</b> This field stands for endpoint number that this endpoint belongs to. Software should write 0, 1, 2, 3, 4, 5, 6, 7 to this field of endpoint 0, 1, 2, 3, 4, 5, 6, 7 information register respectively.	RW

## 1.2.2 UDC Device Registers

### 1.2.2.1 UDC Device Configuration Register (UDCFGR)

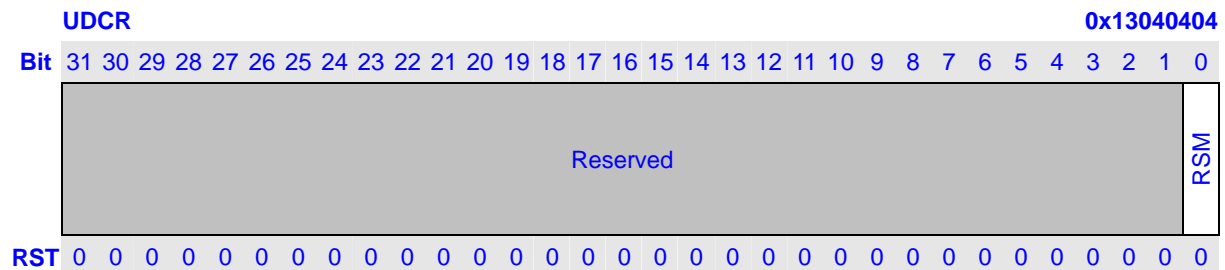
UDC Device Configuration Register is used to configure the device during initialization.



Bits	Name	Description	RW
31:5	Reserved	Always read 0, write is ignored	R
4	SF	<b>Sync Frame</b> Set when Device supports the Sync Frame. 0 = Not Support Sync Frame 1 = Support Sync Frame	RW
3	SP	<b>Self-Powered</b> Set when the Device is Self-Powered. 0 = Bus-Powered 1 = Self-Powered	RW
2	RW	<b>Remote Wakeup</b> Set when the Device supports Remote Wakeup. 0 = Not support Remote Wakeup 1 = Support Remote Wakeup	RW
1:0	SPD	<b>Speed</b> This field indicates the speed the Device supports. Here for UDC, 0b11 must be set because UDC only supports Full speed. 0b00 = HS 0b01 = Reserved 0b10 = LS 0b11 = FS	RW

### 1.2.2.2 UDC Device Control Register (UDCR)

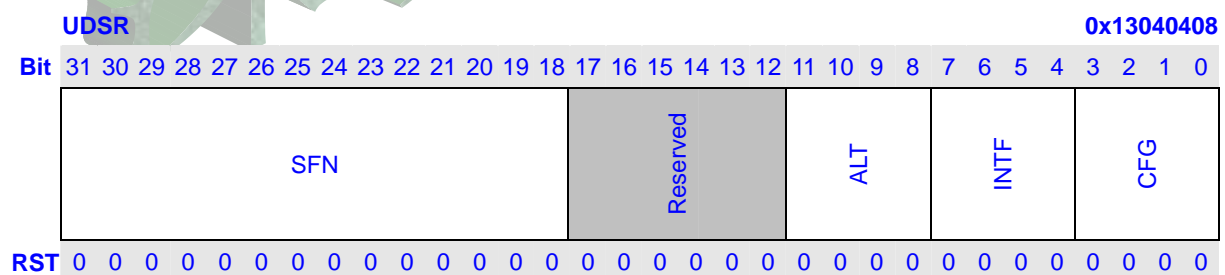
UDC Device Control Register is used to send RESUME signal during run time.



Bits	Name	Description	RW
31:1	Reserved	Always read 0, should only be written with 0	R
0	RSM	<b>Resume</b> Set when the system will do a resume signaling on the USB. User must issue set_feature request to ACTIVE UDC DEVICE_REMOTE_WAKEUP before setting RES bit, otherwise Remote Wakeup will not generate. 0 = No Resume on USB 1 = Generate Resume on USB	RW

### 1.2.2.3 UDC Device Status Register (UDSR)

The read-only UDC Device Status Register reflects some status information that is necessary for servicing some of the interrupts.



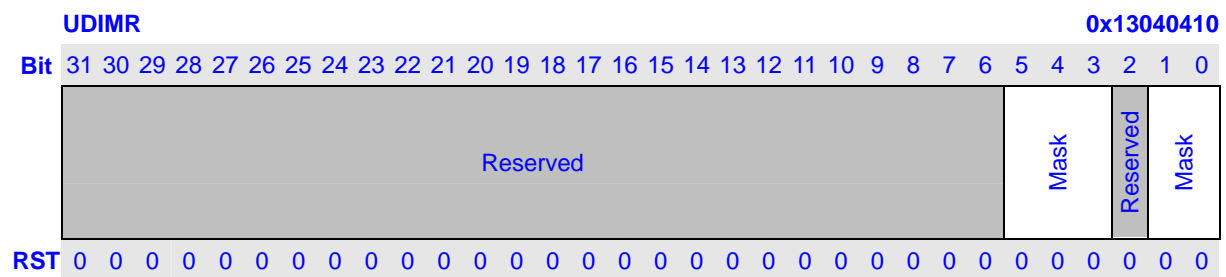
Bits	Name	Description	RW
31:18	SFN	<b>SOF Frame Number</b> This field indicates the frame number of the received SOF.	R
17:12	Reserved	Always read 0, should only be written with 0	R
11:8	ALT	<b>Alternate Setting</b> This field indicates Alternate Setting to which the current interface.	R



7:4	INTF	<b>Interface Number</b> This field indicates the interface number set by <i>SetInterface</i> command.	R
3:0	CFG	<b>Configuration Number</b> This field indicates the configuration set by <i>SetConfiguration</i> command.	R

#### 1.2.2.4 UDC Device Interrupt Mask Register (UDIMR)

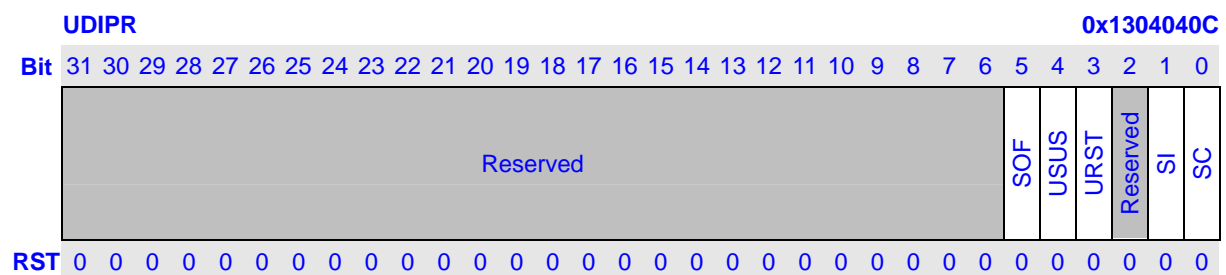
UDC Device Interrupt Mask Register is used to mask the corresponding system-level interrupts in UDIPR.



Bits	Name	Description	RW
31:6	Reserved	Always read 0, should only be written with 0	R
5:3	Mask	<b>Interrupt Mask</b> Software writes 1 in each bit of this field will mask the corresponding interrupt in UDC Device Interrupt Pending Register. 0 = Non-mask 1 = Mask pending interrupt	RW
2	Reserved	Always read 0, should only be written with 0	R
1:0	Mask	Same as bit 5:3	RW

#### 1.2.2.5 UDC Device Interrupt Pending Register (UDIPR)

UDC Device Interrupt Pending Register reflects the status of some system-level events.



Bits	Name	Description	RW
31:6	Reserved	Always read 0, should only be written with 0	R
5	SOF	<b>SOF Token</b> Set when a SOF token is detected on USB. Software writes 1 to clear it to 0. 0 = No SOF Token Detected 1 = SOF Token Detected	RW
4	USUS	<b>USB Suspend</b> Set when a Suspend is detected on USB. Software writes 1 to clear it to 0. 0 = No Suspend Detected 1 = Suspend Detected	RW
3	URST	<b>USB Reset</b> Set when a Reset is detected on USB. Software writes 1 to clear it to 0. 0 = No Reset Detected 1 = Reset Detected	RW
2	Reserved	Always read 0, should only be written with 0	R
1	SI	<b>Set Interface</b> Set when a <i>SetInterface</i> command is detected on USB and Endpoint 0 corresponding interrupt flag will not be set when detecting <i>SetInterface</i> command. Software writes 1'b1 to clear it to 0. 0 = No Set Interface Command Detected 1 = Set Interface Command Detected	RW
0	SC	<b>Set Configuration</b> Set when a <i>SetConfiguration</i> is detected on USB and Endpoint 0 corresponding interrupt flag will not be set when detecting <i>SetConfiguration</i> command. Software writes 1 to clear it to 0. 0 = No Set Configuration Command Detected 1 = Set Configuration Command Detected	RW

#### 1.2.2.6 Setup Command Address Register (USCAR)

UDC Setup Command Address Register contains the address pointer that is to indicate that current transfer data on the USB is setup command. When using UDC, this value should be set to 0xFFFF.

USCAR																0x13040500																
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																SCA															
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	

Bits	Name	Description	RW
31:16	Reserved	Always read 0, should only be written with 0	R
15:0	SCA	<b>Setup Command Address</b> This field is to indicate that current transfer data on the USB is setup command. When using UDC, this value should be set to 0xFFFF.	RW

## 1.3 Operation

### 1.3.1 UDC Device Request Support

The following table shows whether UDC hardware automatically identifies or software identifies the USB device request from USB host. Software handles the device request by UDC interrupt. Refer to USB1.1 protocol for details about the USB device request.

**Table 1-3 UDC Device Request Support**

Device Request	Hardware/Software Support	Interrupt
CLEAR_FEATURE	Hardware	No
SET_FEATURE	Hardware	No
GET_CONFIGURATION	Hardware	No
SET_CONFIGURATION	Hardware	Device Interrupt
GET_DESCRIPTOR	Software	EP0 Interrupt
SET_DESCRIPTOR	Software	EP0 Interrupt
GET_STATUS	Hardware	No
SET_ADDRESS	Hardware	No
GET_INTERFACE	Hardware	No
SET_INTERFACE	Hardware	Device Interrupt
SYNC_FRAME	Software	EP0 and Device Interrupt

### 1.3.2 UDC Structure

According to USB 1.1 protocol, all the USB transactions are initiated by USB Host. So USB Host initiates USB transfer and UDC responds to all the device request from the USB Host.

UDC is a slave of the internal system bus (AHB), so data between USB bus and system memory are transferred by CPU reading/writing the data from/to the memory mapped FIFOs of UDC. All the data transfers are interrupt-driven.

#### 1.3.2.1 Receive FIFO

UDC receives the OUT data from USB bus and stores it in one 16-depth word Receive FIFO RAM and all OUT endpoints data share this common FIFO that means no matter setup data or normal OUT data will go through this FIFO. The entrance address of the Receive FIFO is 0x13040800.

### 1.3.2.2 Transmit FIFO

UDC transmits the IN data to USB bus by one 64-depth word Transmit FIFO RAM. The transmit FIFO is not shared by all IN endpoints. And it is an endpoint-specific FIFO, that is, each IN endpoint has its own FIFO controller and the address generated by the FIFO controller loops around a small address range within the RAM. The size of this address range is equal to the Buffer Size programmed in buffer size register of each endpoint. The base address (entrance address) for each FIFO is dependent on the buffer size and base address of the preceding FIFO. For IN Endpoint 0, the base address is 0x13040840 (start of the Transmit RAM). Then, for IN Endpoint 1, its base address is the sum of 0x13040840 and  $UIBSR0 \times 4$ . It's similar for Endpoint 2, 3 and 4. The Buffer Size that is programmed in the UDC Buffer Size registers must be multiple of 4. So all IN Endpoint transmit FIFO has separate FIFO controller but the whole space is successive and it is seamless between one and another.

## 1.3.3 Flow

### 1.3.3.1 Initialization

Before any transaction, UDC's memory mapped registers must be initialized. A flow example is as the following:

1. Device Initialization
  - Set UDCFG, for example, write 0x00000017 to UDCFG
  - Set UDCR, for example, write 0x00000000 to UDCR
  - Set UDIMR, for example, write 0x0000001F to UDIMR
  - Set UIMR, for example, write 0x00000000 to UIMR
  - Set USCAR, for example, write 0x0000FFFF to USCAR
2. Endpoint Configuration
  - Set EP0 OUT: UOCR0, UOMPR0, for example, write 0x00000000 to UOCR0, write 0x00000020 to UOMPR0;
  - Set EP0 IN: UICR0, UIMPR0, UIBSR0, for example, write 0x00000002 to UICR0 to flush IN buffer, then write 0x00000000 to UICR0 and write 0x00000020 to UIMPR0 and write 0x00000008 to UIBSR0;
  - Set EP1 IN: UICR1, UIMPR1, UIBSR1, for example, write 0x00000002 to UICR1 to flush IN buffer, then write 0x00000030 to UICR1 and write 0x00000040 to UIMPR1 and write 0x00000010 to UIBSR1;
  - Set EP2 IN: UICR2, UIMPR2, UIBSR2, for example, write 0x00000002 to UICR2 to flush IN buffer, then write 0x00000020 to UICR2 and write 0x00000040 to UIMPR2 and write

0x00000010 to UIBSR2;

Set EP5 OUT: UOCR5, UOMPR5, for example, write 0x00000020 to UOCR5, write 0x00000040 to UOMPR5;

Set EP3 IN: UICR3, UIMPR3, UIBSR3, for example, write 0x00000002 to UICR3 to flush IN buffer, then write 0x00000030 to UICR3 and write 0x00000040 to UIMPR3 and write 0x00000010 to UIBSR3;

Set EP6 OUT: UOCR6, UOMPR6, for example, write 0x00000020 to UOCR6, write 0x00000040 to UOMPR6;

Set EP4 IN: UICR4, UIMPR4, UIBSR4, for example, write 0x00000002 to UICR4 to flush IN buffer, then write 0x00000010 to UICR4 and write 0x00000040 to UIMPR4 and write 0x00000010 to UIBSR4;

Set EP7 OUT: UOCR7, UOMPR7, for example, write 0x00000010 to UOCR7, write 0x00000040 to UOMPR7;

Set UINFR<sub>n</sub>, n=0,1,2,3,4,5,6,7, for example, write 0x01000080 to UINFR0, write 0x020000F1 to UINFR1, write 0x020000D2 to UINFR2, write 0x020008D3 to UINFR3, write 0x020010B4 to UINFR4, write 0x020000C5 to UINFR5, write 0x020008C6 to UINFR6 and write 0x020010A7 to UINFR7.

3. Prepare interrupt handler routine

### 1.3.3.2 IN Request Transaction

If UDC receives an IN token for an endpoint, UDC will automatically check whether the Transmit Data FIFO has data. If there is data available, UDC will read the FIFO and transmit the data to USB bus. If the FIFO does not contain any data, UDC will generate an interrupt to CPU and this token is retried on the USB bus.

Once CPU gets the interrupt, it probes the Interrupt Pending Register (UIPR and UDIPR) to know which endpoint has requested the interrupt. After determining the endpoint, CPU probes the UDC Status register to know the cause of the interrupt. If it is an IN token for the endpoint, CPU will write the packet data directly to the Endpoint IN Data FIFO. After the packet data is written to the FIFO, CPU performs a single write to transmit data confirm address (0x1304041C) that indicates to UDC the packet data transfer from system memory to the endpoint FIFO is done. Once the USB Host retries the IN token, UDC transfers the data from the transmit FIFO to USB bus. Transmit FIFO can be flushed by setting the F bit of UDC Control register. The following is the example sequence of these events for an IN endpoint.

1. Initial all necessary UDC control registers as 1.3.3.1.
2. Wait for that USB Host initiates a transaction.

The data transfer between USB bus and system memory is interrupt-driven. But for isochronous and interrupt transaction, USB Host will probe the isochronous or interrupt endpoint periodically, so UDC software may prepare the data in the isochronous or interrupt endpoint buffer before the interrupt occurs, then write 0x1304041C once to confirm the data

---

and wait for USB Host to access the data.

3. IN transaction received.
4. UDC check for the endpoint number, set the corresponding interrupt pending bit in UDC interrupt pending register and check whether the endpoint transmit FIFO is empty or not. If the FIFO is not empty and the data is already confirmed, UDC transfers the data directly to USB bus; if the FIFO is empty and the interrupt is not masked, UDC generates an interrupt to CPU, otherwise, CPU needs probing the UDC interrupt pending register.
5. When CPU knows the IN token for the endpoint 0, 1, 2, 3 or 4 by reading UDC status registers, it writes the packet data directly to the base address of endpoint 0, 1, 2, 3 or 4 transmit FIFO as described in 1.3.2.2.
6. After writing data to transmit FIFO, CPU should write 0x1304041C once to confirm the packet data in FIFO, then UDC starts to transfer the data in transmit FIFO to USB bus once USB Host retried the IN token on bus.
7. If CPU wants to transmit a zero packet through endpoint 0, 1, 2, 3 or 4, first write transmit zero packet address (0x13040420) once, after that, write the base address of endpoint 0, 1, 2, 3 or 4 transmit FIFO once (refer to Table 1-2), then write 0x1304041C once to confirm the zero packet data, then UDC will transfer a zero packet data on USB bus.

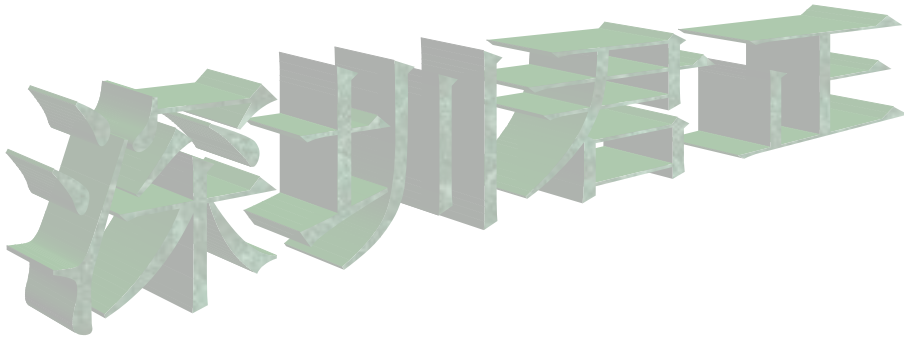
### 1.3.3.3 OUT Request Transaction

Similar to IN Request Transaction, once UDC receives OUT or SETUP data from the USB Host, UDC will transfer the data to the receive FIFO if it has the space available for the packet, otherwise the packet is retried. Once the packet is transferred to the receive FIFO, UDC generates an interrupt to CPU. CPU reads the interrupt pending registers and UDC status registers of the addressed endpoint and knows the number of bytes received in the packet. Then CPU reads the received number of bytes from the receive FIFO. The following is the example sequence of these events for an OUT endpoint.

1. Initial all necessary UDC control registers as 1.3.3.1.
2. Wait USB Host initiates a transaction.
3. OUT transaction received.
4. UDC transfers the data to the receive FIFO if it has the space available for the packet, otherwise it will response NAK on USB bus. USB Host will retry the packet transfer.
5. Once the packet data is transferred to the UDC receive FIFO, if the interrupt is not masked, UDC will generate an interrupt to CPU, otherwise, CPU needs probing the UDC interrupt pending register.
6. CPU knows the OUT token for endpoint 0, 5, 6 or 7 and also the number of bytes received in the packet by reading UDC status registers and then CPU reads the received number of bytes from the UDC receive FIFO.
7. When a zero packet is received, CPU must read 0x1304041C once to confirm zero length packet reception.

#### 1.3.3.4 Zero Length Packet

1. To confirm zero length packet reception after receiving a zero length packet, read 0x1304041C.
2. To transmit a zero length packet, first write 0x13040420 once, write the entrance address of IN endpoint 0(or 1 or 2 or 3 or 4) FIFO once and write 0x1304041C once, then UDC will transmit a zero length packet to USB bus.



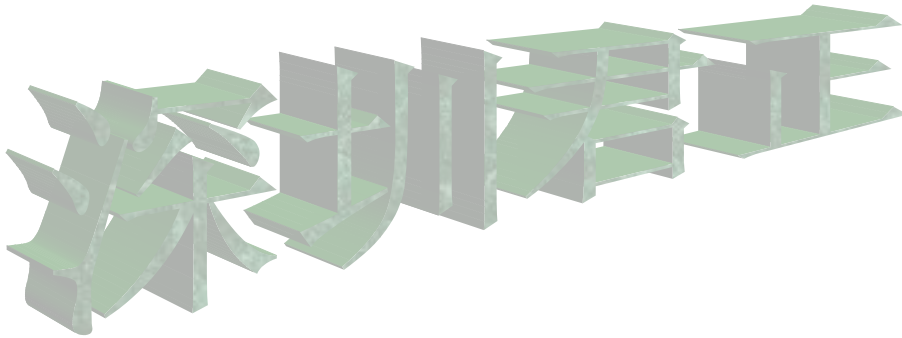
# 1 LCD Controller

## 1.1 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. Spatio-temporal dithering (frame rate modulation) is supported for STN type LCD panels.

Features:

- Single and Dual panel displays in STN mode.
- Single panel displays in TFT mode.
- Up to 4096 colors in STN mode.
- Up to 65536 colors in TFT mode.
- Display size up to 1024x1024.
- Internal palette RAM 256x16 bits.
- Support 1, 2, 4, 8 pins STN panel.
- Support generic TFT data format.
- Support ITU601/656 data format.

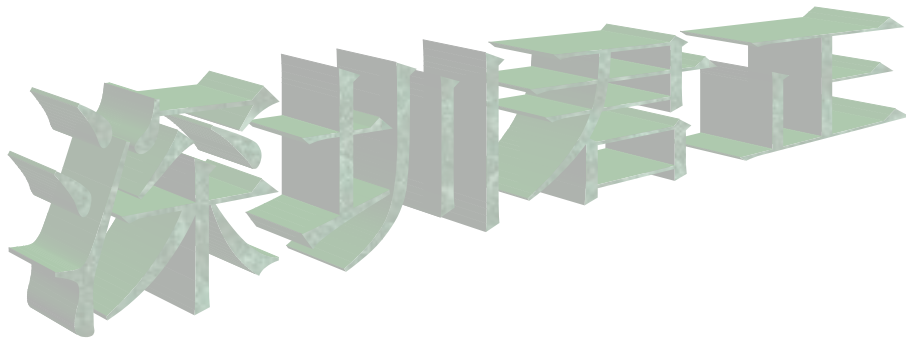




## 1.2 Pin Description

**Table 1-1 LCD Controller Pins Description**

Name	I/O	Description
Lcd_d[15:0]	Output	Display device data pins
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_spl	Output	Special pin for SAMSUNG, SHARP and CASIO TFT LCD
Lcd_cls	Output	Special pin for SAMSUNG, SHARP and CASIO TFT LCD
Lcd_ps	Output	Special pin for SAMSUNG, SHARP and CASIO TFT LCD
Lcd_rev	Output	Special pin for SAMSUNG, SHARP and CASIO TFT LCD



### 1.3 Register Description

Table 1-2 LCD Controller Registers Description

Name	RW	Reset Value	Address	Access Size
LCDCFG	RW	0x00000000	0x13050000	32
LCDVSYNC	RW	0x00000000	0x13050004	32
LCDHSYNC	RW	0x00000000	0x13050008	32
LCDVAT	RW	0x00000000	0x1305000C	32
LCDDAH	RW	0x00000000	0x13050010	32
LCDDAV	RW	0x00000000	0x13050014	32
LCDPS <sup>*1</sup>	RW	0x00000000	0x13050018	32
LCDCLS <sup>*1</sup>	RW	0x00000000	0x1305001C	32
LCDSPL <sup>*1</sup>	RW	0x00000000	0x13050020	32
LCDREV <sup>*1</sup>	RW	0x00000000	0x13050024	32
LCDCTRL	RW	0x00000000	0x13050030	32
LCDSTATE	RW	0x00000000	0x13050034	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
LCDDA1 <sup>*2</sup>	RW	0x00000000	0x13050050	32
LCDSA1 <sup>*2</sup>	R	0x00000000	0x13050054	32
LCDFID1 <sup>*2</sup>	R	0x00000000	0x13050058	32
LCDCMD1 <sup>*2</sup>	R	0x00000000	0x1305005C	32

**Note:** \*1: These registers are only used for SPECIAL TFT panels.

\*2: These registers are only used for Dual Panel STN panels.

#### 1.3.1 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
	Reserved								PSM	CLSM	SPLM	REVM	HSYNM	PCLKM	INVDAT	SYNDIR	PSP	CLSP	SPLP	REVP	HSP	PCP	DEP	VSP	Reserved	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
31:24	Reserved		R
23	PSM	PS signal mode bit: 1 – disabled, 0 – enabled.	RW
22	CLSM	CLS signal mode bit: 1 – disabled, 0 – enabled.	RW

21	SPLM	SPL signal mode bit: 1 – disabled, 0 – enabled.	RW																				
20	REVM	REV signal mode bit: 1 – disabled, 0 – enabled.	RW																				
19	HSYNM	H-Sync signal polarity choice function: 1 – disabled, 0 – enabled.	RW																				
18	PCLKM	Dot clock signal polarity choice function: 1 – disabled, 0 – enabled.	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																				
7:6	Reserved		R																				
5:4	PDW	STN pins utilization <table><tr><td></td><td>Signal Panel</td></tr><tr><td>00</td><td>Lcd_d[0]</td></tr><tr><td>01</td><td>Lcd_d[0:1]</td></tr><tr><td>10</td><td>Lcd_d[0:3]</td></tr><tr><td>11</td><td>Lcd_d[0:7]</td></tr><tr><td></td><td>Dual-Monochrome Panel</td></tr><tr><td>00</td><td>Reserved</td></tr><tr><td>01</td><td>Reserved</td></tr><tr><td>10</td><td>Upper panel: lcd_d[3:0], lower panel: lcd_d[11:8]</td></tr><tr><td>11</td><td>Upper panel: lcd_d[7:0], lower panel: lcd_d[15:8]</td></tr></table>		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
	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]																						

3:0	MODE	Display Device Mode Select	RW																												
		<table><tr><td></td><td>LCD Panel</td></tr><tr><td>0000</td><td>Generic TFT Panel</td></tr><tr><td>0001</td><td>SHARP HR-TFT Panel</td></tr><tr><td>0010</td><td>CASIO TFT Panel</td></tr><tr><td>0011</td><td>SAMSUNG α-TFT Panel</td></tr><tr><td>0100</td><td>Non-Interlaced CCIR656</td></tr><tr><td>0101</td><td>Reserved</td></tr><tr><td>0110</td><td>Interlaced CCIR656</td></tr><tr><td>0111</td><td>Reserved</td></tr><tr><td>1000</td><td>Single-Color STN Panel</td></tr><tr><td>1001</td><td>Single-Monochrome STN Panel</td></tr><tr><td>1010</td><td>Dual-Color STN Panel</td></tr><tr><td>1011</td><td>Dual-Monochrome STN Panel</td></tr><tr><td>1011~1111</td><td>Reserved</td></tr></table>		LCD Panel	0000	Generic TFT Panel	0001	SHARP HR-TFT Panel	0010	CASIO TFT Panel	0011	SAMSUNG α-TFT Panel	0100	Non-Interlaced CCIR656	0101	Reserved	0110	Interlaced CCIR656	0111	Reserved	1000	Single-Color STN Panel	1001	Single-Monochrome STN Panel	1010	Dual-Color STN Panel	1011	Dual-Monochrome STN Panel	1011~1111	Reserved	
	LCD Panel																														
0000	Generic TFT Panel																														
0001	SHARP HR-TFT Panel																														
0010	CASIO TFT Panel																														
0011	SAMSUNG α-TFT Panel																														
0100	Non-Interlaced CCIR656																														
0101	Reserved																														
0110	Interlaced CCIR656																														
0111	Reserved																														
1000	Single-Color STN Panel																														
1001	Single-Monochrome STN Panel																														
1010	Dual-Color STN Panel																														
1011	Dual-Monochrome STN Panel																														
1011~1111	Reserved																														

### 1.3.2 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	
31:27	Reserved																														R	
26:16	VPS		V-Sync Pulse start position, fixed to 0 (in line clock)																												R	
15:11	Reserved																														R	
10:0	VPE		V-Sync Pulse end position (in line clock)																												RW	

### 1.3.3 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
31:27	Reserved		R
26:16	HPS	H-Sync pulse start position (in dot clock)	RW
15:11	Reserved		R
10:0	HPE	H-Sync pulse end position (in dot clock)	RW

### 1.3.4 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
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		R
26:16	HT	Horizontal Total size (in dot clock, sum of display area and blank space)	WR
15:11	Reserved		R
10:0	VT	Vertical Total size (in line clock, sum of display area and blank space)	WR

### 1.3.5 Display Area Horizontal Start/End Point (LCDDAH)

LCDDAH

0x13050010

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
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		R
26:16	HDS	Horizontal display area start (in dot clock)	WR
15:11	Reserved		R
10:0	HDE	Horizontal display area end (in dot clock)	WR

### 1.3.6 Display Area Vertical Start/End Point (LCDDAV)

LCDDAV

0x13050014

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
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		R
26:16	VDS	Vertical display area start position (in line clock)	WR
15:11	Reserved		R
10:0	VDE	Vertical display area end position (in line clock)	WR

### 1.3.7 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
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		R
26:16	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 every N lines per frame. PSS defines the Toggle position.	WR
15:11	Reserved		R
10:0	PSE	PS signal end position (in dot clock). In STN mode, PSE defines N, which described in PSS.	WR

### 1.3.8 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
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		R
26:16	CLSS	CLS signal start position (in dot clock)	WR
15:11	Reserved		R
10:0	CLSE	CLS signal end position (in dot clock)	WR

### 1.3.9 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
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		R
26:16	SPLS	SPL signal start position (in dot clock)	WR
15:11	Reserved		R
10:0	SPLE	SPL signal end position (in dot clock)	WR

### 1.3.10 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
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		R
26:16	REVS	REV signal start position (in dot clock)	WR
15:11	Reserved		R
10:0	REVE	REV signal end position (in dot clock)	WR

## 1.3.11 Control Register (LCDCTRL)

LCDCTRL

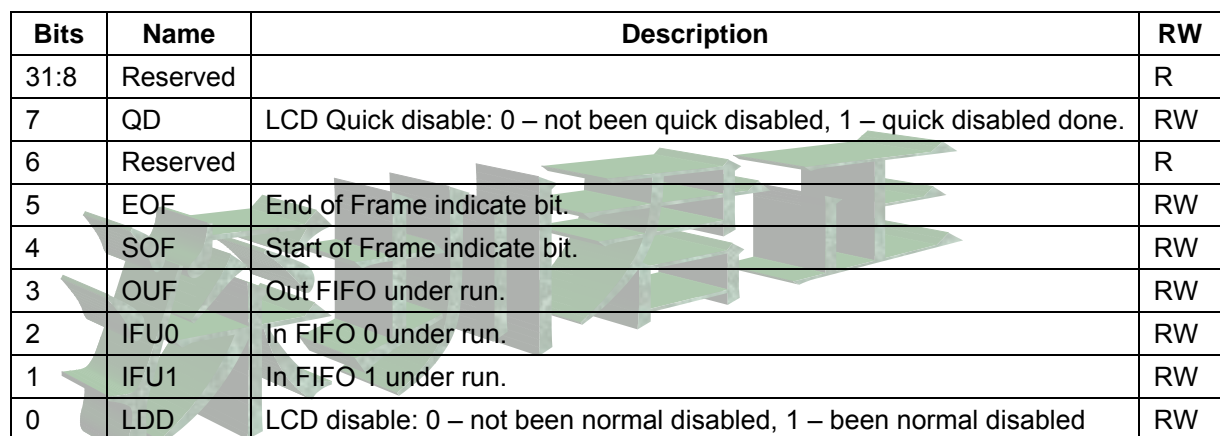
0x13050030

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												
	Reserved		BST		RGB		OFUP		FRC		PDD						Reserved		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	0	0	0	0	0	0	0	0	0			

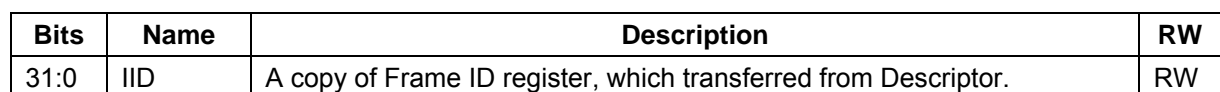
Bits	Name	Description	RW	
31:30	Reserved		R	
29:28	BST	Burst Length Selection	RW	
				Burst Length
		00		4 word
		01		8 word
		10		16 word
		11		Reserved
27	RGB	RGB mode: 0 – RGB565, 1 – RGB555.	RW	
26	OFUP	Output FIFO under run protection: 0 – disable, 1 – enable.	RW	
25:24	FRC	STN FRC Algorithm Selection	RW	
				Grayscale
		00		16 grayscale
		01		4 grayscale
		10		2 grayscale
		11		Reserved
23:16	PDD	Load Palette Delay Counter	RW	
15:14	Reserved		R	
13	EOFM	Mask end of frame interrupt: 0 – enabled, 1 – masked	RW	
12	SOFM	Mask start of frame interrupt: 0 – enabled, 1 – masked	RW	
11	OFUM	Mask out FIFO under run interrupt: 0 – enabled, 1 – masked	RW	
10	IFUM0	Mask in FIFO 0 under run interrupt: 0 – enabled, 1 – masked	RW	
9	IFUM1	Mask in FIFO 1 under run interrupt: 0 – enabled, 1 – masked	RW	
8	LDDM	Mask LCD disable done interrupt: 0 – enabled, 1 – masked	RW	
7	QDM	Mask LCD quick disable done interrupt: 0 – enabled, 1 – masked	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	



## 0x13050034



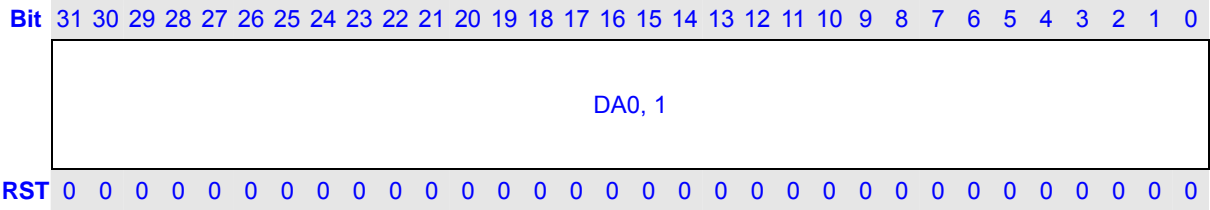
## 0x13050038



### 1.3.14 Descriptor Address Register0, 1 (LCDDA0, 1)

LCDDA0, LCDDA1

0x13050040, 0x13050050

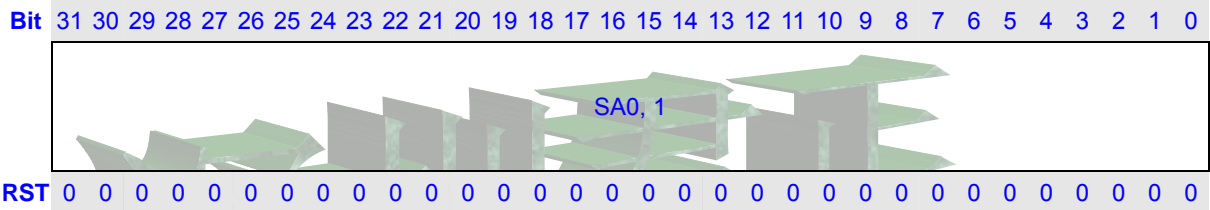


Bits	Name	Description	RW
31:0	DA0, 1	Next descriptor physical address. And descriptor structure as following:  WORD [0]: next descriptor physical address. WORD [1]: the buffer physical address. WORD [2]: the buffer ID value. (Only for debug) WORD [3]: the buffer property. The value is same as LCDCMD.	RW

### 1.3.15 Source Address Register0, 1 (LCDSA0, 1)

LCDSA0, LCDSA1

0x13050044, 0x13050054

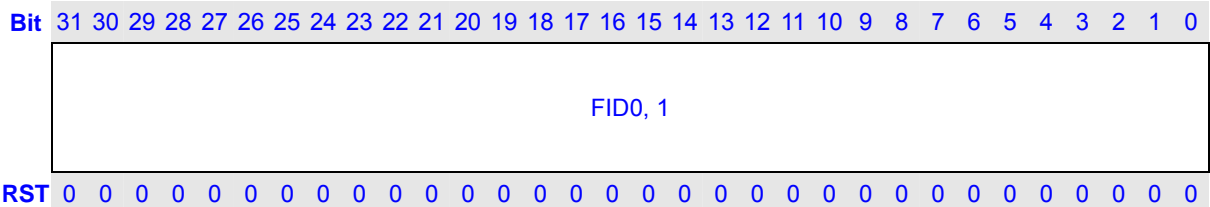


Bits	Name	Description	RW
31:0	SA0, 1	Buffer start address. (Only for driver debug)	R

### 1.3.16 Frame ID Register0 (LCDFID0,1)

LCDFID0, LCDFID1

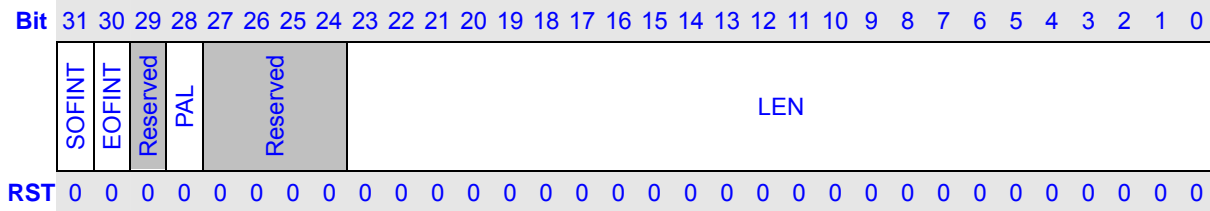
0x13050048, 0x13050058



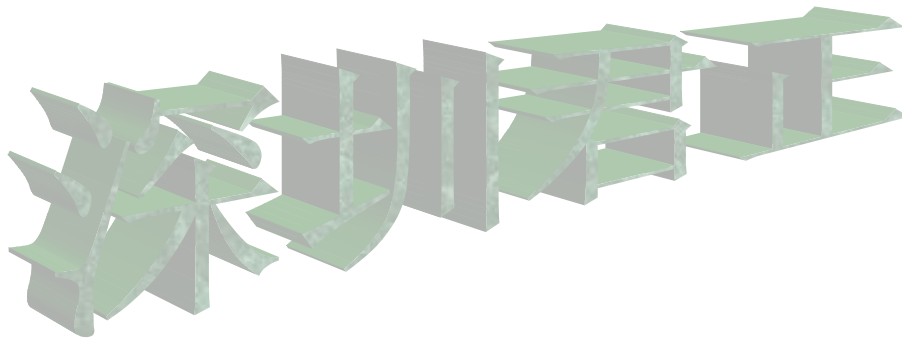
Bits	Name	Description	RW
31:0	FID0, 1	Frame ID. (Only for debug)	R

**LCDCMD0, LCDCMD1** **0x1305004C, 0x1305005C**

**0x1305004C, 0x1305005C**



Bits	Name	Description	RW
31	SOFINT	Enable start of frame interrupt	R
30	EOFINT	Enable end of frame interrupt	R
29	Reserved		R
28	PAL	The descriptor contains a palette buffer.	R
27:24	Reserved		R
23:0	LEN	The buffer length value (in WORD)	R



## 1.4 LCD Controller Pin Mapping

There are several mapping schemes for different LCD panels.

### 1.4.1 TFT and CCIR656 Pin Mapping

Pin	Generic TFT	SHARP HR-TFT	CASIO TFT	SAMSUNG $\alpha$ -TFT	CCIR656 8-bit	CCIR656 16-bit
Lcd_pclk	CLK	DCLK	CLK	HCLK	CLK	CLK
Lcd_vsync	VSYNC	SPS	GSRT	STV	VSYNC	VSYNC
Lcd_hsync	HSYNC	LP	GPCK	STH	HSYNC	HSYNC
Lcd_de	DE	-	-	-	-	-
Lcd_ps	-	PS	POL	CKV	-	-
Lcd_cls	-	CLS	GRES	LD	-	-
Lcd_rev	-	REV	FRP	INV	-	-
Lcd_spl	-	SPL	STH	VCOM	-	-
Lcd_dat15	R5	R5	R5	R5	-	D15
Lcd_dat14	R4	R4	R4	R4	-	D14
Lcd_dat13	R3	R3	R3	R3	-	D13
Lcd_dat12	R2	R2	R2	R2	-	D12
Lcd_dat11	R1	R1	R1	R1	-	D11
Lcd_dat10	G5	G5	G5	G5	-	D10
Lcd_dat9	G4	G4	G4	G4	-	D9
Lcd_dat8	G3	G3	G3	G3	-	D8
Lcd_dat7	G2	G2	G2	G2	D7	D7
Lcd_dat6	G1	G1	G1	G1	D6	D6
Lcd_dat5	G0	G0	G0	G0	D5	D5
Lcd_dat4	B5	B5	B5	B5	D4	D4
Lcd_dat3	B4	B4	B4	B4	D3	D3
Lcd_dat2	B3	B3	B3	B3	D2	D2
Lcd_dat1	B2	B2	B2	B2	D1	D1
Lcd_dat0	B1	B1	B1	B1	D0	D0

### 1.4.2 Single STN Pin Mapping

Pin	Color STN	Mono STN			
		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_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

### 1.4.3 Dual Panel STN Pin Mapping

Pin	Color STN	Mono STN			
		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_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

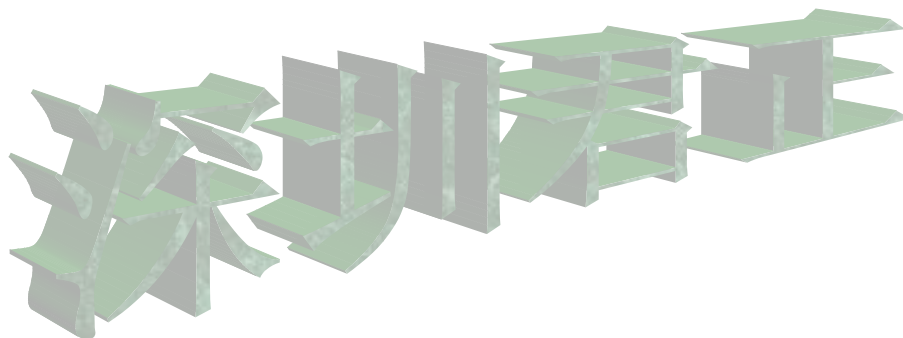
# 1 Camera Interface Module

## 1.1 Overview

The camera interface module (CIM) connects to a CMOS or CCD type image sensor. The CIM sources the digital image stream through a common parallel digital protocol. The CIM can be configured to connect directly to external image sensors and CCIR656 standard video decoders.

The CIM has the following features:

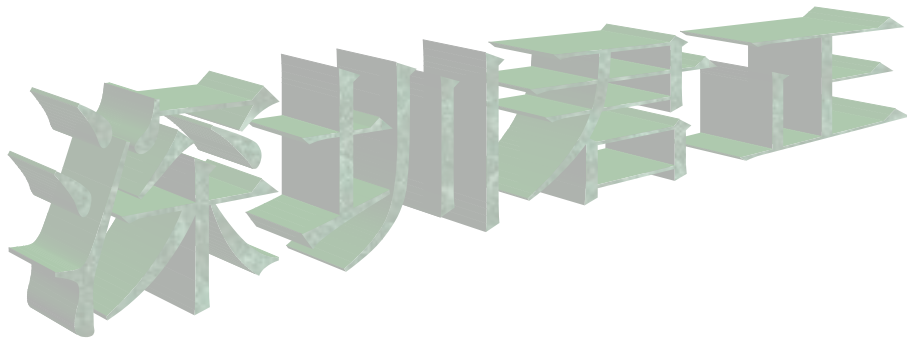
- Input image size up to 2048x2048 pixels
- Integrated DMA support
- Supports generic image data format
- Supports CCIR656 data format
- Configurable CIM\_VSYNC and CIM\_HSYNC signals: active high/low
- Configurable CIM\_PCLK: active edge rising/falling
- 32x32 image data receive FIFO (RXFIFO)



1.2 Pin Description

Table 1-1 Camera Interface Pins Description

Name	I/O	Description
CIM_MCLK	Output	Master clock to Image Sensor
CIM_PCLK	Input	Pixel clock from Image Sensor
CIM_VSYNC	Input	VSYNC from Image Sensor
CIM_HSYNC	Input	HSYNC from Image Sensor
CIM_DATA[7:0]	Input	Data bus from Image Sensor





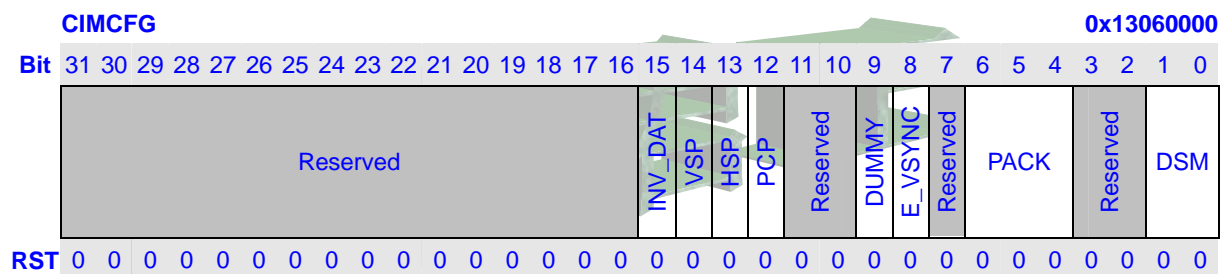
### 1.3 Register Description

The CIM has nine registers to configure camera interface and DMA operation for the input data. The table below list these registers.

**Table 1-2 CIM Registers**

Name	RW	Reset Value	Address	Access Size
CIMCFG	RW	0x00000000	0x13060000	32
CIMCR	RW	0x00000000	0x13060004	32
CIMST	RW	0x00000000	0x13060008	32
CIMIID	R	0x00000000	0x1306000C	32
CIMRXFIFO	R	0x????????	0x13060010	32
CIMDA	RW	0x00000000	0x13060020	32
CIMFA	R	0x00000000	0x13060024	32
CIMFID	R	0x00000000	0x13060028	32
CIMCMD	R	0x00000000	0x1306002C	32

#### 1.3.1 CIM Configuration Register Register (CIMCFG)



Bits	Name	Description	RW
31:16	Reserved		R
15	INV_DAT	Inverse every bit of input data. 0 – not inverse; 1 – inverse	RW
14	VSP	VSYNC polarity selection. When VSYNC signal is input from pin CIM_VSYNC, this bit specifies the VSYNC signal active level and leading edge. When VSYNC is retrieved from SAV&EAV, this bit is ignored. 0 – VSYNC signal active high, VSYNC signal leading edge is rising edge; 1 – VSYNC signal active low, VSYNC signal leading edge is falling edge	RW
13	HSP	Specifies the HSYNC signal active level and leading edge.	RW

		0 – HSYNC signal active high, HSYNC signal leading edge is rising edge; 1 – HSYNC signal active low, HSYNC signal leading edge is falling edge																			
12	PCP	Specifies the PCLK working edge. 0 – Data is sampled by PCLK rising edge; 1 – Data is sampled by PCLK falling edge	RW																		
11:10	Reserved		R																		
9	DUMMY	DUMMY zero function. When DUMMY is 1, CIM hardware adds one byte zero to every 3 input data bytes to form 32-bit data. 0 – DUMMY zero function disabled; 1 – DUMMY zero function enabled	RW																		
8	E_VSYN C	External / internal VSYNC selection. When DSM is CCIR656 Progressive Mode, VSYNC can be external (provided by sensor) or internal (retrieved from SAV&EAV). This bit only valid for CCIR656 Progressive Mode; In other DSM modes, this bit is ignored. 0 – Internal VSYNC mode, pin CIM_VSYNC is ignored; 1 – External VSYNC mode, VSYNC is provided by image sensor via pin CIM_VSYNC	RW																		
7	Reserved		R																		
6:4	PACK	<div><div>Data packing mode, pack 8-bit input data into 32-bit data for FIFO.<table><thead><tr><th>PACK</th><th>Description</th></tr></thead><tbody><tr><td>3'b000</td><td>0x 11 22 33 44</td></tr><tr><td>3'b001</td><td>0x 22 33 44 11</td></tr><tr><td>3'b010</td><td>0x 33 44 11 22</td></tr><tr><td>3'b011</td><td>0x 44 11 22 33</td></tr><tr><td>3'b100</td><td>0x 44 33 22 11</td></tr><tr><td>3'b101</td><td>0x 33 22 11 44</td></tr><tr><td>3'b110</td><td>0x 22 11 44 33</td></tr><tr><td>3'b111</td><td>0x 11 44 33 22</td></tr></tbody></table></div><div>In this table, 0x11, 0x22, 0x33 and 0x44 means the received data from the sensor, 0x11 is received first and 0x44 is received last.</div></div>	PACK	Description	3'b000	0x 11 22 33 44	3'b001	0x 22 33 44 11	3'b010	0x 33 44 11 22	3'b011	0x 44 11 22 33	3'b100	0x 44 33 22 11	3'b101	0x 33 22 11 44	3'b110	0x 22 11 44 33	3'b111	0x 11 44 33 22	RW
PACK	Description																				
3'b000	0x 11 22 33 44																				
3'b001	0x 22 33 44 11																				
3'b010	0x 33 44 11 22																				
3'b011	0x 44 11 22 33																				
3'b100	0x 44 33 22 11																				
3'b101	0x 33 22 11 44																				
3'b110	0x 22 11 44 33																				
3'b111	0x 11 44 33 22																				
3:2	Reserved		R																		
1:0	DSM	<div><div>Data sample mode. Please refer to the table below.<table><thead><tr><th>DSM</th><th>Description</th></tr></thead><tbody><tr><td>2'b00</td><td>CCIR656 Progressive Mode</td></tr><tr><td>2'b01</td><td>CCIR656 Interlace Mode</td></tr><tr><td>2'b10</td><td>Gated Clock Mode</td></tr><tr><td>2'b11</td><td>Non-Gated Clock Mode</td></tr></tbody></table></div></div>	DSM	Description	2'b00	CCIR656 Progressive Mode	2'b01	CCIR656 Interlace Mode	2'b10	Gated Clock Mode	2'b11	Non-Gated Clock Mode	RW								
DSM	Description																				
2'b00	CCIR656 Progressive Mode																				
2'b01	CCIR656 Interlace Mode																				
2'b10	Gated Clock Mode																				
2'b11	Non-Gated Clock Mode																				

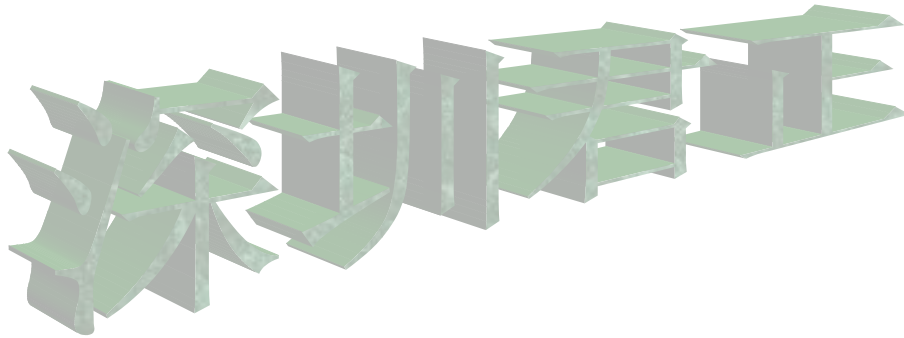
### 1.3.2 CIM Control Register (CIMCR)

CIMCR																0x13060004																												
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												
	MCLKDIV								Reserved				FRC				Reserved		VDDM		DMA_SOFM		DMA_EOFM		DMA_STOPM		RF_TRIGM		RF_OFM		Reserved		RF_TRIG				Reserved		DMA_EN		RF_RST		ENA	
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	0	0	0	0	0							

Bits	Name	Description	RW												
31:24	MCLKDIV	The parameter for master clock MCLK generation from device clock. MCLK = (Device clock) / (MCLKDIV + 1)	RW												
23:20	Reserved		R												
19:16	FRC	CIM frame rate control. Specifies the sampling frame data rate. If FRC = N, CIM sampling one frame of every N+1 frames from the sensor. In this way, CIM reduces the frame rate of sensor. Another way to reduce frame rate is to decrease the MCLK frequency output to the image sensor. <table border="1"><thead><tr><th>FRC</th><th>Description</th></tr></thead><tbody><tr><td>4'b0000</td><td>Sample every frame from the sensor</td></tr><tr><td>4'b0001</td><td>Sample 1 frame of every 2 frames from the sensor</td></tr><tr><td>.....</td><td>.....</td></tr><tr><td>4'b1110</td><td>Sample 1 frame of every 15 frames from the sensor</td></tr><tr><td>4'b1111</td><td>Sample 1 frame of every 16 frames from the sensor</td></tr></tbody></table>	FRC	Description	4'b0000	Sample every frame from the sensor	4'b0001	Sample 1 frame of every 2 frames from the sensor	.....	.....	4'b1110	Sample 1 frame of every 15 frames from the sensor	4'b1111	Sample 1 frame of every 16 frames from the sensor	RW
FRC	Description														
4'b0000	Sample every frame from the sensor														
4'b0001	Sample 1 frame of every 2 frames from the sensor														
.....	.....														
4'b1110	Sample 1 frame of every 15 frames from the sensor														
4'b1111	Sample 1 frame of every 16 frames from the sensor														
15:14	Reserved		R												
13	VDDM	The enable control bit for VDD interrupt. 0 – disable; 1 – enable	RW												
12	DMA_SOFM	The enable control bit for DMA_SOF interrupt. 0 – disable; 1 – enable	RW												
11	DMA_EOFM	The enable control bit for DMA_EOF interrupt. 0 – disable; 1 – enable													
10	DMA_STOPM	The enable control bit for DMA_STOP interrupt. 0 – disable; 1 – enable	RW												
9	RF_TRIGM	The enable control bit for RXF_TRIG interrupt. 0 – disable; 1 – enable	RW												
8	RF_OFM	The enable control bit for RXF_OF interrupt. 0 – disable; 1 – enable	RW												

## Camera Interface Module

7	Reserved		R	
6:4	RF_TRIG	Specifies the trigger value of RXFIFO.		RW
		<b>RXF_TRIG</b>	<b>Description</b>	
		0	Trigger Value is 4	
		1	Trigger Value is 8	
		2	Trigger Value is 12	
		3	Trigger Value is 16	
		4	Trigger Value is 20	
		5	Trigger Value is 24	
		6	Trigger Value is 28	
		7	Trigger Value is 32	
3	Reserved		R	
2	DMA_EN	Enable / disable the DMA function. 0 – disable DMA; 1 – enable DMA	RW	
1	RF_RST	RXFIFO software reset. Setting 1 to RXF_RST can reset RXFIFO immediately. Setting 0 to RXF_RST can stop resetting RXFIFO. After reset, RXFIFO is empty.	RW	
0	ENA	Enable / disable the CIM module. Setting 1 to ENA can enable CIM. When CIM is working, clear ENA to 0 can stop CIM immediately. 0 – CIM is not enabled, or disable CIM immediately; 1 – CIM is enabled, or enabling CIM	RW	



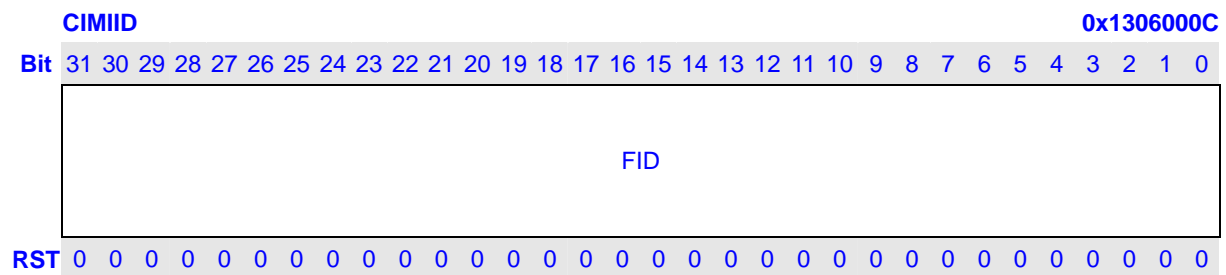
### 1.3.3 CIM Status Register (CIMST)

CIMST																								0x13060008								
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																								DMA_SOF	DMA_EOF	DMA_STOP	RF_OF	RF_TRIG	RF_EMPTY	VDD	
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	0	0	0	0	0

Bits	Name	Description	RW
31:7	Reserved		R
6	DMA_SOF	When set to 1, Indicate the DMA start transferring the first data from RXFIFO to frame buffer. Can generate interrupt if CIMCR.DMA_SOFM bit is set. Writing 0 to this bit will clear it, writing 1 will be ignored.	RW
5	DMA_EOF	When set to 1, indicate the DMA complete transferring one frame data from RXFIFO to frame buffer. Can generate interrupt if CIMCR.DMA_EOFM bit is set. Writing 0 to this bit will clear it, writing 1 will be ignored.	RW
4	DMA_STOP	When set to 1, indicate the DMA complete transferring data and stop the operation. Can generate interrupt if CIMCR.DMA_STOPM bit is set. Writing 0 to this bit will clear it, writing 1 will be ignored.	RW
3	RF_OF	RXFIFO over flow. When RXFIFO over flow happens, RXF_OF is set 1. Can generate interrupt if CIMCR.RF_OFM bit is set. Writing 0 to this bit will clear it, writing 1 will be ignored.	RW
2	RF_TRIG	RXFIFO trigger. Indicates whether RXFIFO meet the trigger value or not. When the valid data number in RXFIFO reaches the trig value, RXF_TRIG is set 1; when the valid data number in RXFIFO do not reach the trig value, RXF_TRIG is set 0. Can generate interrupt if CIMCR.RF_TRIGM bit is set. 0 – RXFIFO does not meets the trigger value; 1 – RXFIFO meets the trigger value	R
1	RF_EMPTY	RXFIFO empty. Indicates whether RXFIFO is empty or not. After reset, RXFIFO is empty, and RXF_EMPTY is 1. 0 – RXFIFO is not empty; 1 – RXFIFO is empty	R
0	VDD	CIM disable done. Indicate this module is disabled after clear the CIMCR.ENA bit to disable the CIM module. Can generate interrupt if CIMCR.DMA_VDDM bit is set.	RW

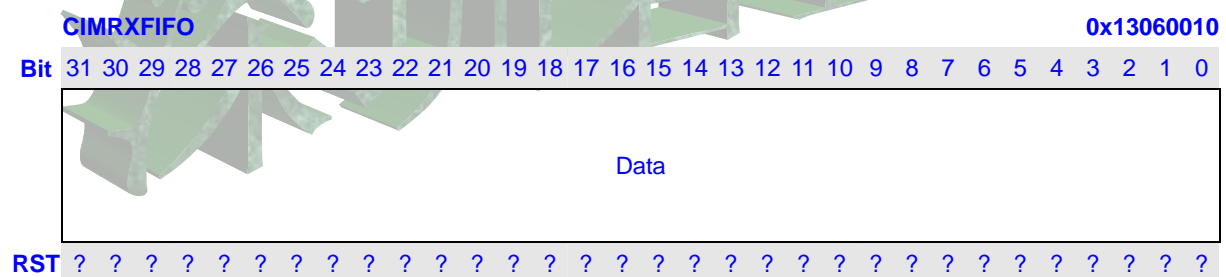
		0 – CIM has not been disabled; 1 – CIM has been disabled Writing 0 to this bit will clear it, writing 1 will be ignored.	
--	--	--	--

### 1.3.4 CIM Interrupt ID Register (CIMIID)



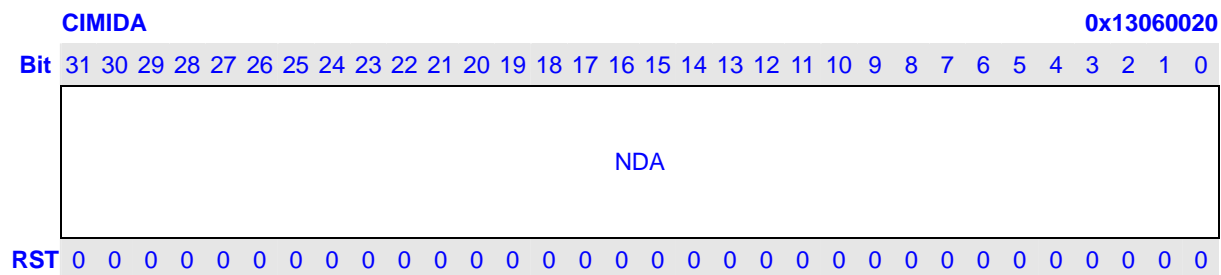
Bits	Name	Description	RW
31:0	FID	Frame ID. Contains a copy of the Frame ID register (CIMFID) from the descriptor currently being processed when a DMA_SOF or DMA_EOF interrupt is generated. CIMIID is written to only when CIMCMD.SOFINT or CIMCMD.EOFINT is high. As such, the register is considered to be sticky and will be overwritten only when the associated interrupt is cleared by writing the CIM state register.	R

### 1.3.5 CIM RXFIFO Register (CIMRXFIFO)



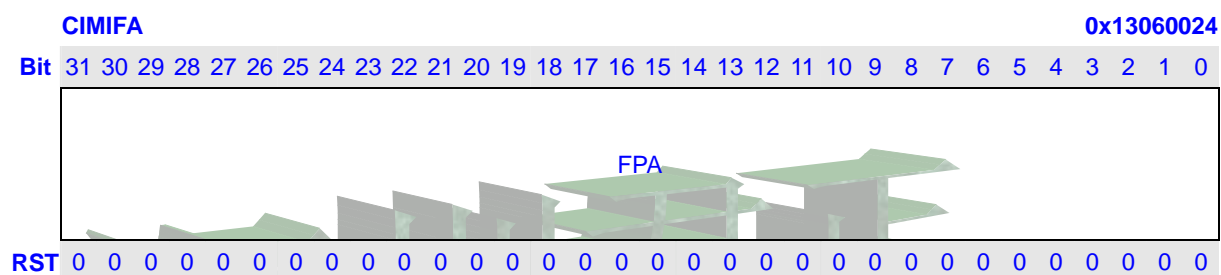
Bits	Name	Description	RW
31:0	Data	This register provides a port for software to read image data directly. When the software start CIM with DMA_EN=1, this register should not be read. Otherwise, the DMA data may be damaged.	R

### 1.3.6 CIM Descriptor Address (CIMDA)



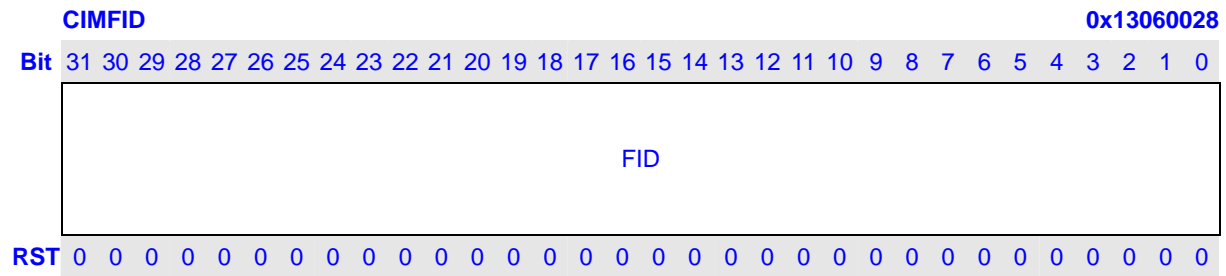
Bits	Name	Description	RW
31:0	NDA	Hold the physical address of the next descriptor in external memory. The DMAC fetches the descriptor at this location after finishing the current descriptor. The target address Bits [3:0] must be zero to be aligned to 16-byte boundary.	RW

### 1.3.7 CIM Frame buffer Address Register (CIMFA)



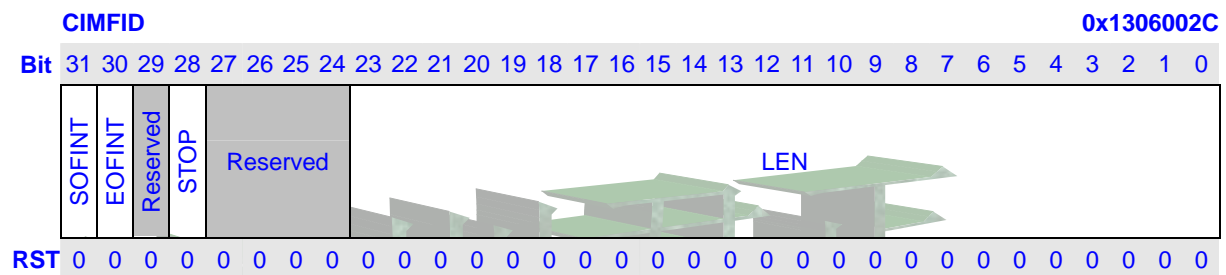
Bits	Name	Description	RW
31:0	FPA	Hold the physical address of frame buffer in external memory. When starts CIM, DMA transfers data from RXFIFO to frame buffer. This address is incremented by hardware as the DMAC writes data to memory. The target address Bits [3:0] must be zero to be aligned to 16-byte boundary.	R

### 1.3.8 CIM Frame ID Register (CIMFID)



Bits	Name	Description	RW
31:0	FID	Hold the 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 CIM Interrupt ID Register when an interrupt occurs.	R

### 1.3.9 CIM DMA Command Register (CIMCMD)

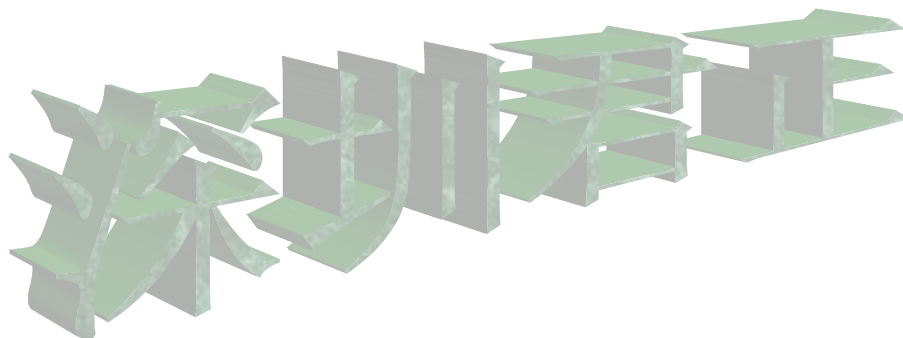


Bits	Name	Description	RW
31	SOFINT	DMA start transferring frame data interrupt. When set to 1, the DMA sets the start of frame bit (CIMSTATE.DMA_SOF) when start transferring image data.	R
30	EOFINT	DMA end transferring frame data interrupt. When set to 1, the DMA sets the end of frame bit (CIMSTATE.DMA_EOF) when complete transferring image data.	R
29	Reserved		R
28	STOP	DMA stop. When DMA complete transferring data, STOP bit decides whether DMA should loading next descriptor or not. 0 – DMA start loading next descriptor; 1 – DMA stopped, and CIMSTATE.DMA_STOP bit is set 1 by hardware	R
27:24	Reserved		R
23:0	LEN	Length of transfer in words. Indicate the number of words to be	R



---

		transferred by DMA. LEN = 0 is not valid. DMA transfers data according to LEN. Each time one or more word(s) been transferred, LEN is decreased automatically.	
--	--	--	--



## 1.4 CIM Data Sampling Modes

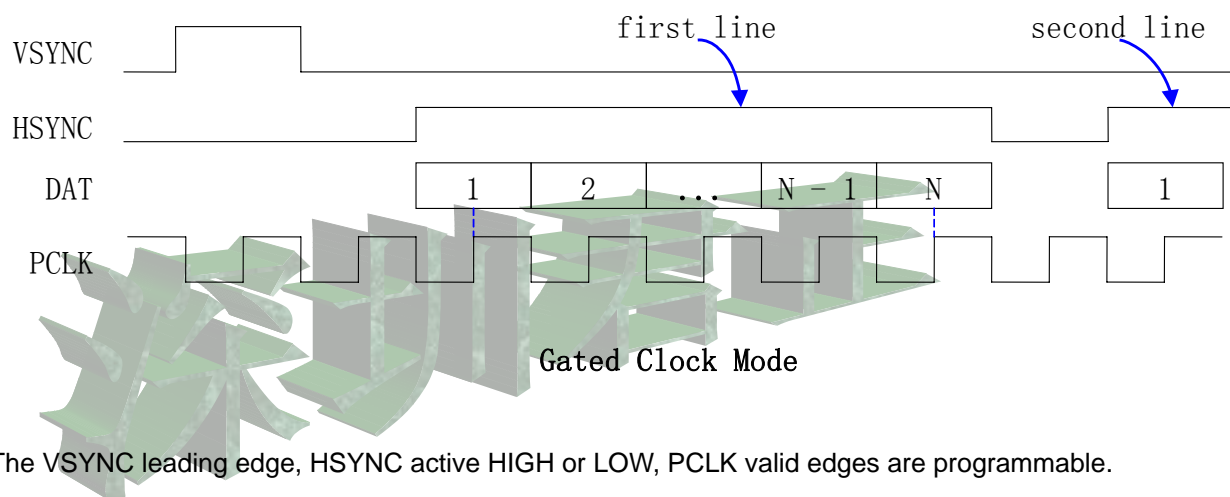
This module support 4 data sampling mode:

1. Gated Clock Mode
2. Non-Gated Clock Mode
3. CCIR656 Interlace Mode
4. CCIR656 Progressive Mode

### 1.4.1 Gated Clock Mode

CIM\_VSYNC, CIM\_HSYNC, and CIM\_PCLK signals are used in this mode.

A frame start with VSYNC leading edge, then HSYNC goes active and holds the entire line. Data is sampled at the valid edge of PCLK when HSYNC is active; That means, HSYNC functions like “data enable” signal. Please refer to the figure below.



The VSYNC leading edge, HSYNC active HIGH or LOW, PCLK valid edges are programmable.

### 1.4.2 Non-Gated Clock Mode

CIM\_VSYNC and CIM\_PCLK signals are used in this mode. CIM\_HSYNC signal is ignored.

A frame starts with VSYNC leading edge, and samples data at every PCLK valid edge. Please refer to the figure below.

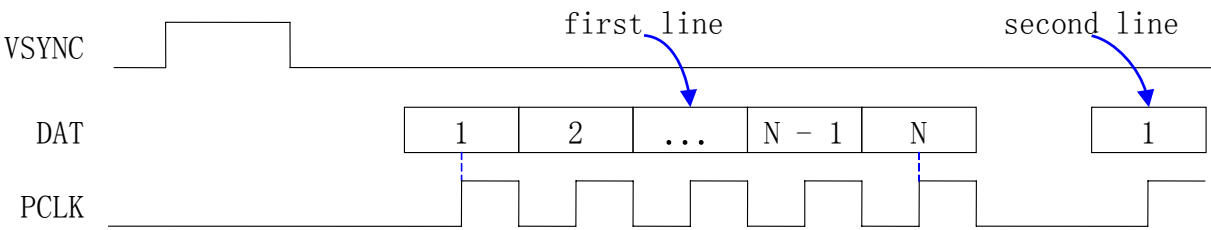


Figure 1-1 Non-Gated Clock Mode

1.4.3 CCIR656 Interlace Mode

CIM\_PCLK and CIM\_DAT signals are used in this mode, CIM\_VSYNC, CIM\_HSYNC signals are ignored.

CIM utilizes the SAV & EAV code within CCIR656 data stream to get active video data.

The following diagrams and tables are quoted from CCIR656 standard. Only the PAL format is shown. CIM supports both NTSC and PAL formats. For more information about CCIR656, please refer to CCIR656 standard.

1.4.3.1 PAL Vertical Timing

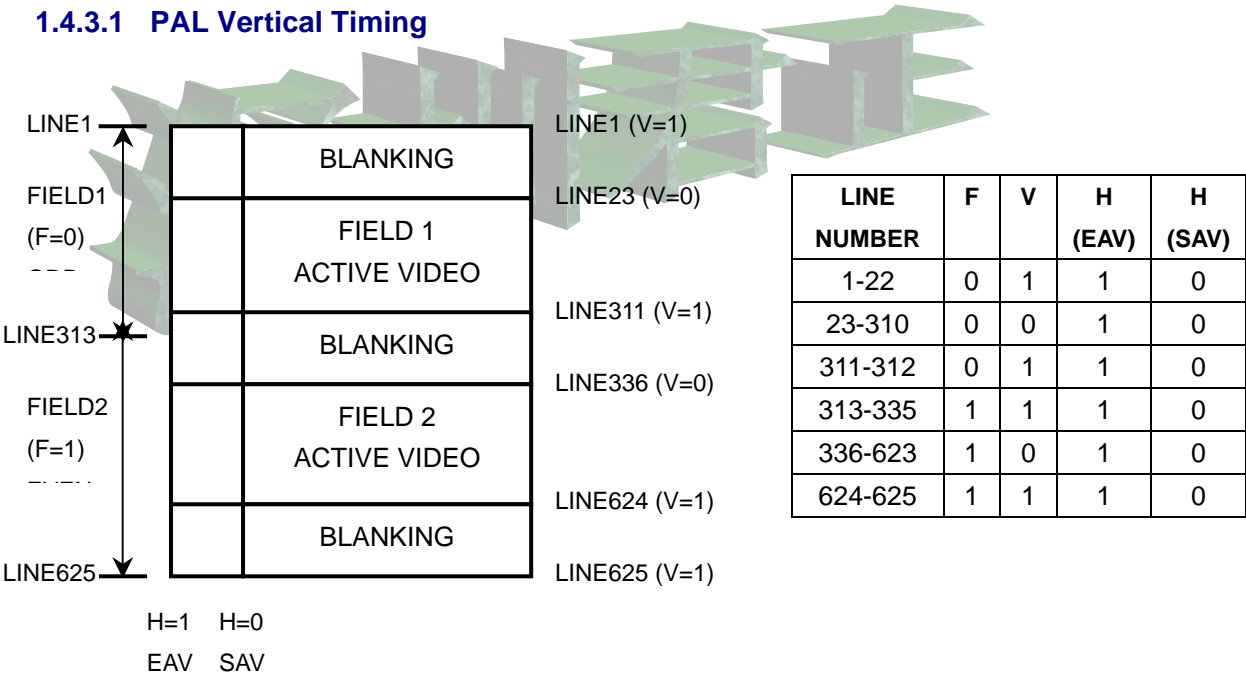


Figure 1-2 Typical BT.656 Vertical Blanking Intervals for 625/50 Video Systems

1.4.3.2 PAL Horizontal Timing

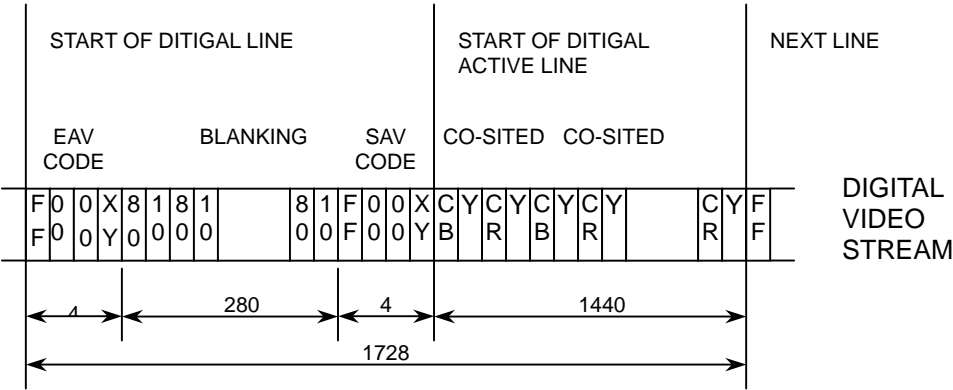


Figure 1-3 BT.656 8-BIT Parallel Interface Data Format for 625/50 Video Systems

1.4.3.3 Coding for SAV and EAV

Data Pin Number	1 <sup>st</sup> Byte 0xFF	2 <sup>nd</sup> Byte 0x00	3 <sup>rd</sup> Byte 0x00	4 <sup>th</sup> Byte 0xXY
7 (MSB)	1	0	0	1
6	1	0	0	F
5	1	0	0	V
4	1	0	0	H
3	1	0	0	P3
2	1	0	0	P2
1	1	0	0	P1
0 (LSB)	1	0	0	P0

### 1.4.3.4 Coding for Protection Bits

F	V	H	P3	P2	P1	P0
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	1	1
1	0	1	1	0	1	0
1	1	0	1	1	0	0
1	1	1	0	0	0	1

### 1.4.4 CCIR656 Progressive Mode

CIM\_PCLK and CIM\_DAT signals are used in this mode. CIM\_HSYNC signal is ignored.

CIM\_VSYNC is optional in this mode. When the start of frame information is retrieved from SAV and EAV, it is known as internal VSYNC mode. When CIM\_VSYNC is provided by sensor directly, it is known as external VSYNC mode. CIM supports both internal and external VSYNC modes.

CCIR656 Progressive Mode is a kind of Non-Interlace Mode. The image data are encoded within only one field. The F-bit of SAV and EAV are ignored. Most sensors support CCIR656 Progressive Mode.

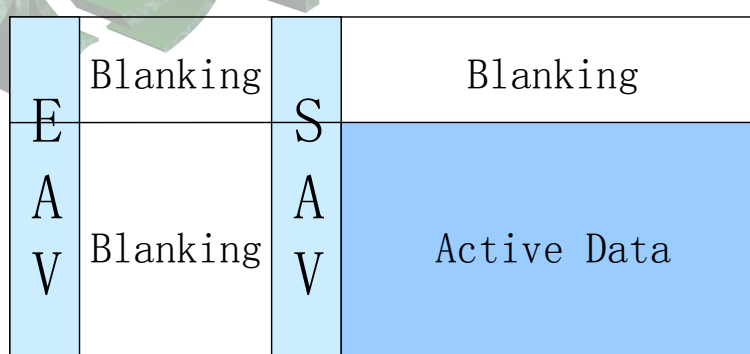


Figure 1-4 CCIR656 Progressive Mode

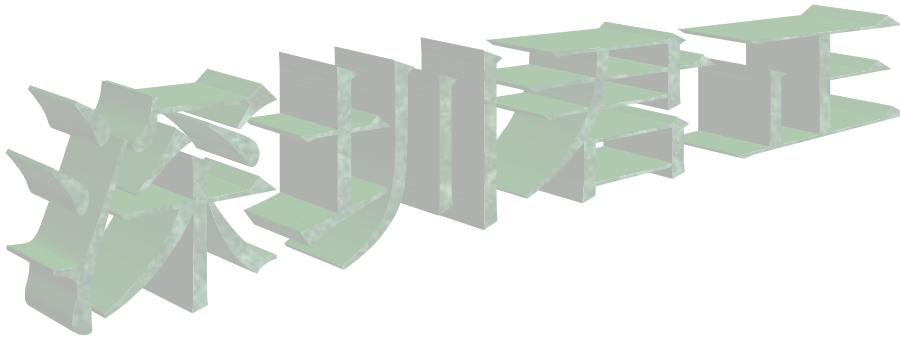
## 1.5 DMA Descriptors

A DMA descriptor is a 4-word block corresponding to the four DMA registers – CIMDA, CIMFA, CIMFID, and CIMCMD, aligned on 4-word (16-byte) boundary, in external memory:

- word [0] contains the physical address for next CIMDA
- word [1] contains the physical address for CIMFA
- word [2] contains the value for CIMFID
- word [3] contains the value for CIMCMD

Software must write the physical address of the first descriptor to CIMDA before enabling the CIM. Once the CIM is enabled, the first descriptor is read, and all 4 registers are written by the DMAC. The next DMA descriptor pointed to by CIMDA is loaded into the registers after all data for the current descriptor has been transferred.

**Note:** If only one frame buffer is used in external memory, the CIMDA field (word [0] of the DMA descriptor) must point back to itself. That is to say, the value of CIMDA is the physical address of itself.



## 1.6 Interrupt Generation

CIM has next interrupt sources:

- **RXFIFO FULL Interrupt (RF\_TRIG)**  
When the valid data number of RXFIFO reaches trigger value, CIMST.RF\_TRIG bit is set. At the same time, if RF\_TRIGM is 1, RF\_TRIG interrupt is generated.
- **RXFIFO Over Flow Interrupt (RF\_OF)**  
When the valid data number of RXFIFO reaches 32 and one more data are written to RXFIFO, CIMST.RF\_OF bit is set. At the same time, if RF\_OFM is 1, RF\_OF interrupt is generated.
- **DMA Start Of Frame Data Transferring Interrupt (DMA\_SOF)**  
When the CIMCMD.SOFINT bit is 1 and DMA start transferring the first data from RXFIFO to frame buffer, CIMST.DMA\_SOF bit is set. At the same time, if DMA\_SOFM is 1, DMA\_SOF interrupt is generated.
- **DMA End Of Frame Data Transferring Interrupt (DMA\_EOF)**  
When the CIMCMD.EOFINT bit is 1 and DMA complete transferring the last data from RXFIFO to frame buffer, CIMST.DMA\_EOF bit is set. At the same time, if DMA\_EOFM is 1, DMA\_EOF interrupt is generated.
- **DMA Stop Transferring Interrupt (DMA\_STOP)**  
When the CIMCMD.STOP bit is 1 and DMA complete transferring the last data from RXFIFO to frame buffer, CIMST.DMA\_STOP bit is set. At the same time, if DMA\_STOPM is 1, DMA\_STOP interrupt is generated.
- **CIM Disable Done Interrupt (VDD)**  
When disable the module by clearing the CIMCR.ENA, the module should be disabled after transferring current valid data. Then set the CIMST.VDD bit, at the same time, if VDDM is set, VDD interrupt is generated.

## 1.7 Software Operation

### 1.7.1 Enable CIM with DMA

1. Configure register CIMCFG;
2. Prepare frame buffer and descriptors;
3. Configure register CIMDA;
4. Write 0 to register CIMSTATE; // clear state register
5. Configure register CIMCTRL with DMA\_EN=1, RXF\_RST=1, ENA=0; // resetting RXFIFO
6. Configure register CIMCTRL with DMA\_EN=1, RXF\_RST=0, ENA=0; // stop resetting RXFIFO
7. Configure register CIMCTRL with DMA\_EN=1, RXF\_RST=0, ENA=1; // enable CIM

### 1.7.2 Enable CIM without DMA

1. Configure register CIMCFG;
2. Write 0 to register CIMSTATE; // clear state register
3. Configure register CIMCTRL with DMA\_EN=0, RXF\_RST=1, ENA=0; // resetting RXFIFO
4. Configure register CIMCTRL with DMA\_EN=0, RXF\_RST=0, ENA=0; // stop resetting RXFIFO
5. Configure register CIMCTRL with DMA\_EN=0, RXF\_RST=0, ENA=1; // enable CIM

### 1.7.3 Disable CIM

Method 1:

1. Configure register CIMCTRL with RXF\_RST=0, ENA=0; // quick disable
2. Write 0 to register CIMSTATE; // clear state register

Method 2:

When DMA is enabled, the following sequence is recommended:

1. Configure descriptor with STOP = 1;
2. Wait DMA\_STOP interrupt, when it occurs, write 0 to CIMCTRL.ENA.
3. Write 0 to register CIMSTATE; // clear state register



# 1 Ethernet MAC Controller

## 1.1 Overview

The JZ4730 processor contains one Ethernet media access controller (MAC), each capable of supporting 10/100Mbps Ethernet. The MAC provides the interface between the host application and the PHY layer through the media independent interface (MII). The PHY layer device is external to the processor.

The MAC supports the protocol requirements to meet the Ethernet/IEEE 802.3 specification. The MAC operates in both half and full duplex modes. In half-duplex mode, the controller supports the IEEE802.3 Carrier Sense Multiple Access with Collision Detection (CSMA/CD) protocol. In full-duplex mode, it supports the IEEE802.3 MAC Control Layer, including the pause operation for flow control.

A dedicated DMA engine is implemented to support the MAC so that the general purpose DMA is not required.

The MAC features are:

- IEEE 802.3, 802.3u specification compliance
- 10/100 Mbps data transfer rates
- IEEE 802.3 compliant MII interface to talk to an external PHY
- Full and half duplex
- CSMA/CD in half duplex
- Flow control support for full duplex
- Collision detection and auto retransmit on collisions in half duplex
- Automatic 32-bit CRC generation and checking
- Optional to insert PAD/CRC32 on transmit packets
- Optional automatic Pad stripping on the receive packets
- External and internal loopback support on the MII
- Filtering modes supported on the Ethernet side:
  - One 48 bit Perfect address
  - 64 hash-filtered multicast addresses
  - Pass all multicast addresses
  - Promiscuous Mode
  - Pass all incoming packets with a status report
  - Toss bad packets
- Big/Little Endian data format support
- Dedicated DMA engine using burst mode
- Dual buffers and linked list Descriptor Chaining support
- Descriptor architecture allows large blocks of data transfer with minimum CPU intervention
- Remote wake-up frame and magic packet frame support
- VLAN support

## 1.2 Ethernet Signals

The following table shows the signals associated with the Ethernet MAC MII interfaces.

**Table 1-1 Ethernet Signals**

Signal	I/O	Description
RX_CLK	Input	Continuous clock that provides the timing reference for the data transfer from the PHY to the MAC. RX_CLK is sourced by the PHY. RX_CLK must have a frequency equal to 25% of the data rate of the received signal data stream (typically 25 MHz at 100-Mbps and 2.5 MHz at 10-Mbps).
RX_DV	Input	RX_DV is driven by the external Ethernet PHY and indicates that a receive frame is in process and that the data on RXD[3:0] is valid.
RX_ER	Input	RX_ER is driven by the Ethernet PHY. RX_ER shall be asserted for one or more RX_CLK periods to indicate to the MAC that an error was detected somewhere in the frame presently being transferred from the PHY to the MAC.
RXD [3:0]	Input	RXD[3:0] is a 4-bit wide data bus driven by the PHY to the MAC synchronous with RX_CLK. For each RX_CLK period in which RX_DV is asserted, RXD[3:0] transfers four bits of recovered data from the PHY to the MAC. While RX_DV is negated, RXD[3:0] has no effect on the MAC.
TX_CLK	Input	Continuous clock input for synchronization of transmit data. 25 MHz when operating at 100-Mbps and 2.5 MHz when operating at 10-Mbps.
TX_EN	Output	Indicates that the data on TXD[3:0] is valid.
TXD [3:0]	Output	4-bit wide data bus synchronous to TX_CLK. For each TX_CLK period in which TX_EN is asserted, TXD[3:0] presents valid data to the PHY. While TX_EN is negated the data presented on TXD[3:0] is not valid.
CRS	Input	The PHY asserts CRS when either transmit or receive medium is non idle. The PHY negates CRS when both the transmit and receive medium are idle. CRS is an asynchronous input.
COL	Input	The PHY asserts COL upon detection of a collision on the medium and continues to assert COL while the collision condition persists. COL is an asynchronous input. The COL signal is ignored by the MAC when operating in full duplex mode.
MDC	Output	MDC is sourced by the MAC to the PHY as the timing reference for transfer of information on the MDIO signal. MDC is an aperiodic signal that has no maximum high or low times. The MDC frequency is fixed at system bus clock divided by 160.
MDIO	I/O	MDIO is the bidirectional data signal between the MAC and the PHY that is clocked by MDC.

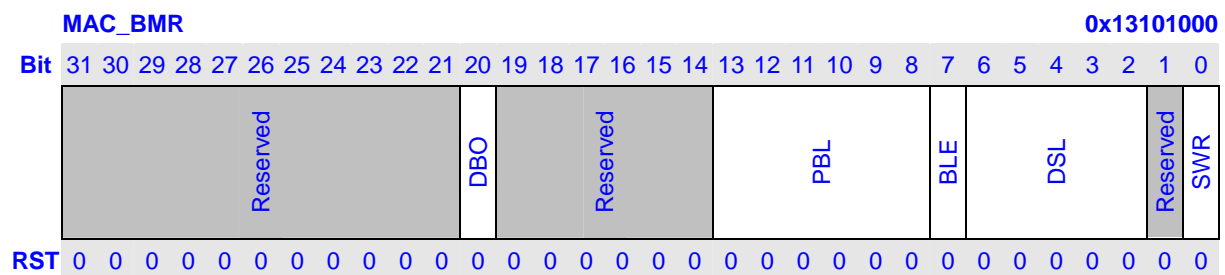
### 1.3 Register Description

The following table shows the Ethernet MAC registers.

**Table 1-2 MAC Registers Description**

Name	RW	Reset Value	Address	Access Size
MAC_BMR	RW	0x00000000	0x13101000	32
MAC_TPDR	W	0x????????	0x13101004	32
MAC_RPDR	W	0x????????	0x13101008	32
MAC_RBAR	RW	0x00000000	0x1310100C	32
MAC_TBAR	RW	0x00000000	0x13101010	32
MAC_SR	RW	0x00000000	0x13101014	32
MAC_OMR	RW	0x00000000	0x13101018	32
MAC_IER	RW	0x00000000	0x1310101C	32
MAC_MFC	R	0x00000000	0x13101020	32
MAC_HTAR	R	0x00000000	0x13101050	32
MAC_HRAR	R	0x00000000	0x13101054	32
MAC_CR	RW	0x00040000	0x13100000	32
MAC_AHR	RW	0x0000FFFF	0x13100004	32
MAC_ALR	RW	0xFFFFFFFF	0x13100008	32
MAC_HTH	RW	0x00000000	0x1310000C	32
MAC_HTL	RW	0x00000000	0x13100010	32
MAC_MIA	RW	0x00000000	0x13100014	32
MAC_MID	RW	0x00000000	0x13100018	32
MAC_FCR	RW	0x00000000	0x1310001C	32
MAC_VTR1	RW	0x0000FFFF	0x13100020	32
MAC_VTR2	RW	0x0000FFFF	0x13100024	32
MAC_WKFR	W	0x00000000	0x13100028	32
MAC_PMBR	RW	0x00000000	0x1310002C	32

#### 1.3.1 Bus Mode Register (MAC\_BMR)



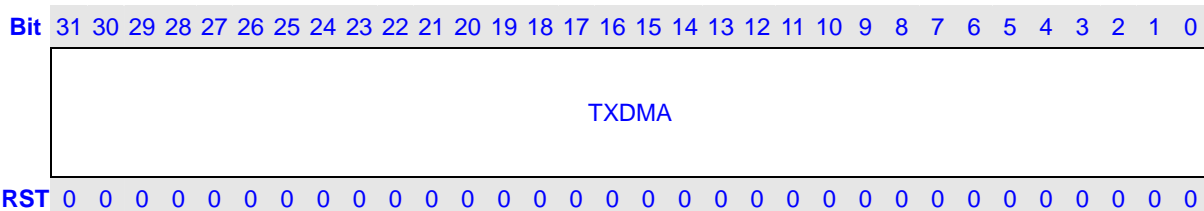
Bits	Name	Description	RW
31:21	Reserved		R

20	DBO	DBO specifies the Descriptor Byte Ordering: 1 – big endian, 0 – little endian	RW
19:14	Reserved		R
13:8	PBL	PBL (Programmable Burst Length) specifies the maximum number of WORDs to be transferred in one DMA transaction. PBL can only be programmed with permissible values of 1 and 4.	RW
7	BLE	BLE (Big/Little Endian) specifies byte ordering of the host data buffers: 1 – big endian, 0 – little endian	RW
6:2	DSL	DSL (Descriptor Skip Length) specifies the number of WORDs to be skipped between two unchained descriptors.	RW
1	Reserved		R
0	SWR	Software Reset. This bit is automatically cleared by hardware.	RW

### 1.3.2 Transmit Poll Demand Register (MAC\_TPDR)

MAC\_TPDR

0x13101004

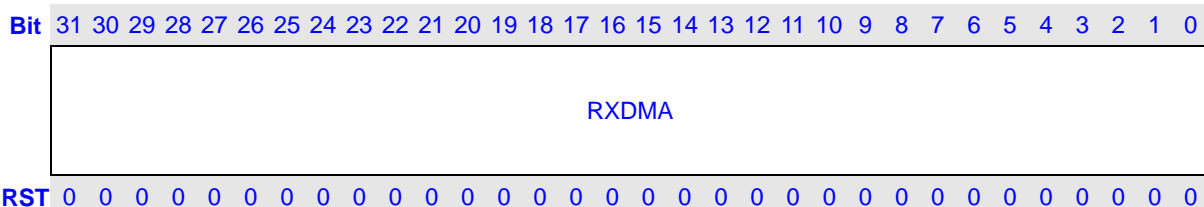


Bits	Name	Description	RW
31:0	TXDMA	Writing any value to this register will enable the Transmit DMA to check for any new descriptors. If the descriptor is available, the MAC controller starts the transmit process. If no descriptor is available, the transmit process returns to the suspended state and MAC_SR[2] is set.	W

### 1.3.3 Receive Poll Demand Register (MAC\_RPDR)

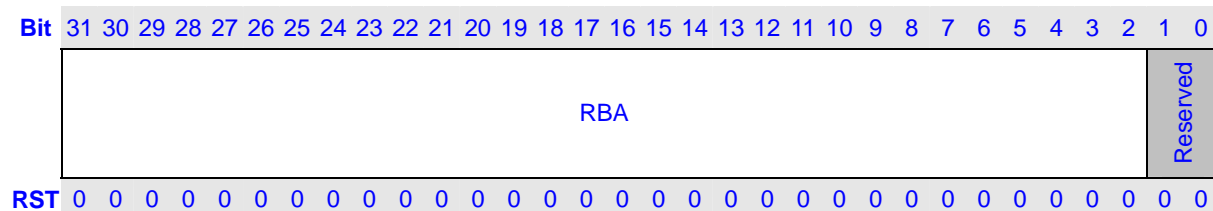
MAC\_RPDR

0x13101008



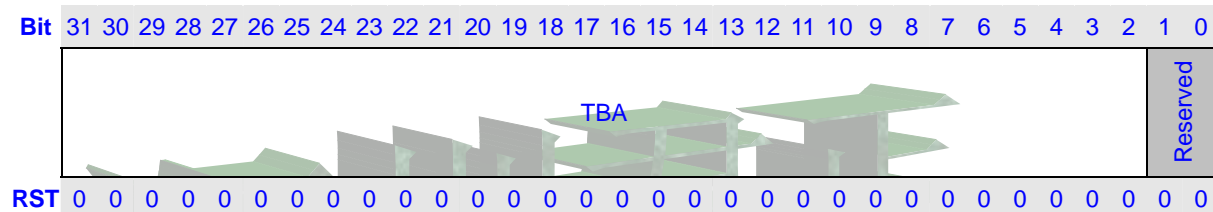
Bits	Name	Description	RW
31:0	RXDMA	Writing any value to this register will enable the Receive DMA to check for any new descriptors. If the descriptor is available, the MAC controller	W

## MAC RBAR 0x1310100C



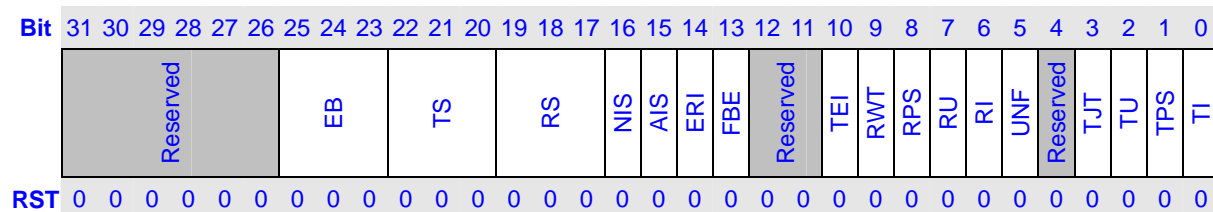
Bits	Name	Description	RW
31:2	RBA	RBA specifies the physical address of the first receive descriptor. RBA must be WORD aligned.	RW

## MAC RBAR 0x13101010



Bits	Name	Description	RW
31:2	TBA	TBA specifies the physical address of the first transmit descriptor. TBA must be WORD aligned.	RW

## MAC\_SR 0x13101014



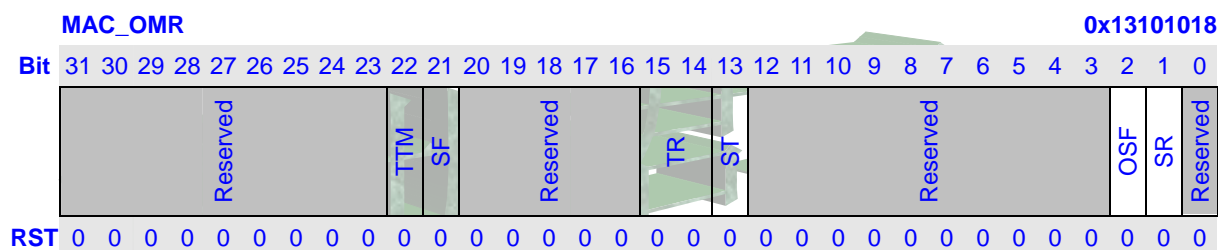
Bits	Name	Description	RW
31:26	Reserved		R

25:23	EB	<p>EB (Error Bits) indicates the type of error that is caused by bus error. EB is valid only when FBE bit is set. EB does not generate interrupt.</p> <table><tr><th>25:23</th><th>Description</th></tr><tr><td>000</td><td>No error</td></tr><tr><td>001</td><td>Tx Abort: Host received ABORT during TX process</td></tr><tr><td>010</td><td>Rx Abort: Host received ABORT during RX process</td></tr><tr><td>1xx</td><td>Reserved</td></tr></table>	25:23	Description	000	No error	001	Tx Abort: Host received ABORT during TX process	010	Rx Abort: Host received ABORT during RX process	1xx	Reserved	R								
25:23	Description																				
000	No error																				
001	Tx Abort: Host received ABORT during TX process																				
010	Rx Abort: Host received ABORT during RX process																				
1xx	Reserved																				
22:20	TS	<p>TS (Transmit State) indicates the state of the transmit process. TS does not generate interrupt.</p> <table><tr><th>22:20</th><th>Description</th></tr><tr><td>000</td><td>Stopped: RESET or STOP command issued</td></tr><tr><td>001</td><td>Running: Fetching the transmit descriptor</td></tr><tr><td>010</td><td>Running: Waiting for end of transmission</td></tr><tr><td>011</td><td>Running: Reading the data from memory and queuing the data into the transmit buffer</td></tr><tr><td>100</td><td>Reserved</td></tr></table>	22:20	Description	000	Stopped: RESET or STOP command issued	001	Running: Fetching the transmit descriptor	010	Running: Waiting for end of transmission	011	Running: Reading the data from memory and queuing the data into the transmit buffer	100	Reserved	R						
22:20	Description																				
000	Stopped: RESET or STOP command issued																				
001	Running: Fetching the transmit descriptor																				
010	Running: Waiting for end of transmission																				
011	Running: Reading the data from memory and queuing the data into the transmit buffer																				
100	Reserved																				
19:17	RS	<p>RS (Receive State) indicates the state of the receive process. RS does not generate interrupt.</p> <table><tr><th>19:17</th><th>Description</th></tr><tr><td>000</td><td>Stopped: RESET or STOP command issued</td></tr><tr><td>001</td><td>Running: Fetching the receive descriptor</td></tr><tr><td>010</td><td>Running: Checking for end of receive packet before prefetching next descriptor</td></tr><tr><td>011</td><td>Running: Waiting for Receive Packet</td></tr><tr><td>100</td><td>Suspended: Unavailable receive buffer</td></tr><tr><td>101</td><td>Running: Closing receive descriptor</td></tr><tr><td>110</td><td>Running: Flushing the current frame from the receive buffer because of unavailable receive buffer</td></tr><tr><td>111</td><td>Running: Queuing the receive frame from the receive buffer into the host memory</td></tr></table>	19:17	Description	000	Stopped: RESET or STOP command issued	001	Running: Fetching the receive descriptor	010	Running: Checking for end of receive packet before prefetching next descriptor	011	Running: Waiting for Receive Packet	100	Suspended: Unavailable receive buffer	101	Running: Closing receive descriptor	110	Running: Flushing the current frame from the receive buffer because of unavailable receive buffer	111	Running: Queuing the receive frame from the receive buffer into the host memory	R
19:17	Description																				
000	Stopped: RESET or STOP command issued																				
001	Running: Fetching the receive descriptor																				
010	Running: Checking for end of receive packet before prefetching next descriptor																				
011	Running: Waiting for Receive Packet																				
100	Suspended: Unavailable receive buffer																				
101	Running: Closing receive descriptor																				
110	Running: Flushing the current frame from the receive buffer because of unavailable receive buffer																				
111	Running: Queuing the receive frame from the receive buffer into the host memory																				
16	NIS	<p>NIS (Normal Interrupt Summary) is the logical OR of TI, TU, RI, and ERI. NIS must be cleared each time a corresponding bit that causes NIS to be set is cleared. Once NIS is cleared, it will be allowed to set again if there is</p>	RW																		

		another pending interrupt. 1 – Normal interrupt occurs, 0 – No normal interrupt	
15	AIS	AIS (Abnormal Interrupt Summary) is the logical OR of TPS, TJT, UNF, RU, RPS, RWT, ETI and FBE. AIS must be cleared each time a corresponding bit that causes AIS to be set is cleared. Once the bit is cleared, it will be allowed to set again if there is another pending interrupt. 1 – Abnormal interrupt occurs, 0 – No abnormal interrupt	RW
14	ERI	ERI (Early Receive Interrupt) indicates that the DMA controller had filled the first data buffer of the packet. Receive interrupt RI clears this bit automatically. 1 – Early receive interrupt occurs, 0 – No early receive interrupt.	RW
13	FBE	FBE (Fatal Bus Error) indicates that a fatal bus error occurred. EB field (bit 25~23) reflect the error cause. When this bit is set, the DMA controller disables all its bus accesses. 1 – Fatal bus error occurs, 0 – No fatal bus error	RW
12:11	Reserved		R
10	ETI	ETI (Early Transmit Interrupt) indicates that the packet to be transmitted was fully transferred into the destination port. Transmit interrupt TI clears this bit automatically. 1 – Early transmit error occurs, 0 – No early transmit error	RW
9	RWT	RWT (Receive Watchdog Timeout) reflects the line status and indicates that the receive watchdog timer has expired while another node is still active on the network. 1 – Receive watchdog timeout occurs, 0 – No receive watchdog timeout	RW
8	RPS	RPS (Receive Process Stopped) indicates that the receive process has entered stopped state. 1 – Receive process enters stopped state 0 – Receive process doesn't enter stopped state	RW
7	RU	RU (Receive Buffer Unavailable) indicates that the next descriptor in the receive list is owned by the host and cannot be acquired by the DMA controller. The reception process is suspended. To resume processing of receive descriptors, the host should change the ownership of the descriptor and issue a receive poll demand command. 1 – Receive buffer is unavailable, 0 – Receive buffer is available	RW
6	RI	RI (Receive Interrupt) indicates the completion of the frame reception. Specific frame status information has been posted in the descriptor. The reception process remains in the running state. 1 – Receive interrupt occurs, 0 – No receive interrupt	RW
5	UNF	UNF (Transmit Underflow) indicates that the transmit buffer had an underflow condition during the packet transmission. The transmit process enters the suspended state and underflow error TDES0[1] is set. 1 – Transmit underflow occurs, 0 – No transmit underflow	RW
4	Reserved		R

3	TJT	TJT (Transmit Jabber Timeout) indicates that the transmit jabber timer had expired. The transmission process is aborted and set to the stopped state. This also causes the transmit jabber timeout TDES0[14] flag to assert. 1 – Transmit jabber timeout occurs, 0 – No transmit jabber timeout	RW
2	TU	TU (Transmit Buffer Unavailable) indicates that the next descriptor in the transmit list is owned by the host and cannot be acquired by the DMA Controller. The transmission process is suspended. To resume processing transmit descriptors, the host should change the ownership of the bit of the descriptor and then issue a transmit poll demand command, unless transmit automatic polling is enabled. 1 – Transmit buffer is unavailable, 0 – Transmit buffer is available	RW
1	TPS	TPS (Transmit Process Stopped) indicates that the transmit process has entered stopped state. 1 – Transmit process enters stopped state 0 – Transmit process doesn't enter stopped state	RW
0	TI	TI (Transmit Interrupt) indicates that a frame transmission was completed and TDES1[31] is set in the first descriptor.	RW

### 1.3.7 Operation Mode Register (MAC\_OMR)



Bits	Name	Description	RW										
31:23	Reserved		R										
22	TTM	<p>TTM (Transmit Threshold Mode Bit[22]) and TR (Threshold Control Bits Bit[15:14]) control the transmit threshold values that the application may use. These bits are not used by the DMA controller and are application specific. These bits are used when SF bit is set. The intent is to allow the application to transfer data to the final destination only after the threshold value is met.</p> <table><tr><th>22,15,14</th><th>Transmit FIFO Threshold</th></tr><tr><td>000</td><td>32 words (100 Mbps)</td></tr><tr><td>001</td><td>64 words (100 Mbps)</td></tr><tr><td>010</td><td>128 words (100 Mbps)</td></tr><tr><td>011</td><td>256 words (100 Mbps)</td></tr></table>	22,15,14	Transmit FIFO Threshold	000	32 words (100 Mbps)	001	64 words (100 Mbps)	010	128 words (100 Mbps)	011	256 words (100 Mbps)	RW
22,15,14	Transmit FIFO Threshold												
000	32 words (100 Mbps)												
001	64 words (100 Mbps)												
010	128 words (100 Mbps)												
011	256 words (100 Mbps)												



		<table><tr><td>100</td><td>18 words (10 Mbps)</td></tr><tr><td>101</td><td>24 words (10 Mbps)</td></tr><tr><td>110</td><td>32 words (10 Mbps)</td></tr><tr><td>111</td><td>40 words (10 Mbps)</td></tr></table>	100	18 words (10 Mbps)	101	24 words (10 Mbps)	110	32 words (10 Mbps)	111	40 words (10 Mbps)	
100	18 words (10 Mbps)										
101	24 words (10 Mbps)										
110	32 words (10 Mbps)										
111	40 words (10 Mbps)										
21	SF	SF (Store and Forward) instructs the application to store a frame of transmit data in application buffer before forwarding to final destination. 1 – Start transmission to MAC until one full frame is available in the transmit FIFO 0 – Start transmission to MAC once the Transmit FIFO level crosses the threshold	RW								
20:16	Reserved		R								
15:14	TR	Horizontal display area end (in dot clock)	RW								
13	ST	Start/Stop Transmission Command: 1 – Start the transmission process 0 – Stop the transmission process	RW								
12:3	Reserved		R								
2	OSF	Operate on Second Frame: 1 – The DMA will prefetch the second frame and put it into transmit FIFO before the first frame is completed. 0 – The DMA will prefetch the next frame from the memory only after the MAC has completely processed the frame and the DMA has released the descriptors.	RW								
1	SR	Start/Stop Receive Command: 1 – Start the receive process 0 – Stop the receive process	RW								
0	Reserved		R								

### 1.3.8 Interrupt Enable Register (MAC\_IER)

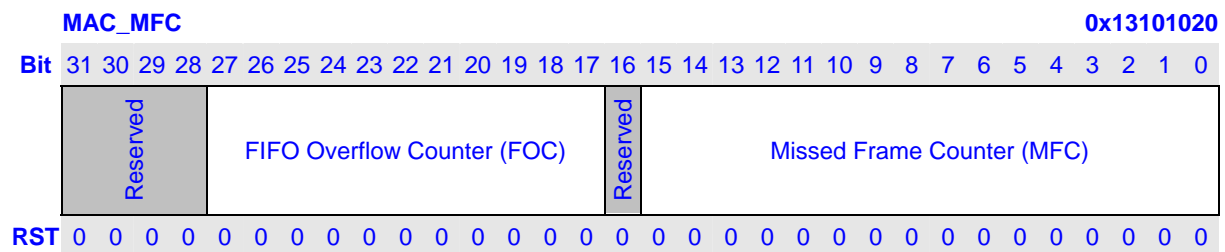
MAC_IER																0x1310101C																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																NI	AI	ERE	FBE	Reserved	ET	RWE	RS	RU	RI	UN	Reserved	TJ	TU	TS	TI
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Bits	Name	Description	RW
31:17	Reserved		R
16	NI	NI (Normal Interrupt Summary Enable) enables the interrupt of MAC_SR.TI, MAC_SR.TU, MAC_SR.RI and MAC_SR.ERI.	RW

		1 – Normal interrupt is enabled, 0 – Normal interrupt is disabled	
15	AI	AI (Abnormal Interrupt Summary Enable) enables the interrupt of MAC_SR.TPS, MAC_SR.TJT, MAC_SR.UNF, MAC_SR.RU, MAC_SR.RPS, MAC_SR.RWT, MAC_SR.ETI and MAC_SR.FBE. 1 – Abnormal interrupt is enabled, 0 – Abnormal interrupt is disabled	RW
14	ERE	Early Receive Interrupt Enable: 1 – When set together with MAC_IER.NI, early receive interrupt is enabled 0 – Early receive interrupt is disabled	RW
13	FBE	Fatal Bus Error Interrupt Enable: 1 – When set together with MAC_IER.AI, fatal bus error interrupt is enabled 0 – Fatal bus error interrupt is disabled	RW
12:11	Reserved		R
10	ET	Early Transmit Interrupt Enable: 1 – When set together with MAC_IER.AI, early transmit interrupt is enabled 0 – Early transmit interrupt is disabled	RW
9	RWE	Receive Watchdog Timeout Interrupt Enable: 1 – When set together with MAC_IER.AI, receive watchdog timeout interrupt is enabled 0 – Receive watchdog timeout interrupt is disabled	RW
8	RS	Receive Stopped Interrupt Enable: 1 – When set together with MAC_IER.AI, receive stopped interrupt is enabled 0 – Receive stopped interrupt is disabled	RW
7	RU	Receive Buffer Unavailable interrupt Enable: 1 – When set together with MAC_IER.AI, receive buffer unavailable interrupt is enabled 0 – Receive buffer unavailable interrupt is disabled	RW
6	RI	Receive Interrupt Enable: 1 – When set together with MAC_IER.NI, receive interrupt is enabled 0 – Receive interrupt is disabled	RW
5	UN	Transmit Underflow Interrupt Enable 1 – When set together with MAC_IER.AI, transmit underflow interrupt is enabled 0 – Transmit underflow interrupt is disabled	RW
4	Reserved		R
3	TJ	Transmit Jabber Timeout Interrupt Enable: 1 – When set together with MAC_IER.AI, transmit jabber timeout interrupt is enabled 0 – Transmit jabber timeout interrupt is disabled	RW
2	TU	Transmit Buffer Unavailable Interrupt Enable:	RW

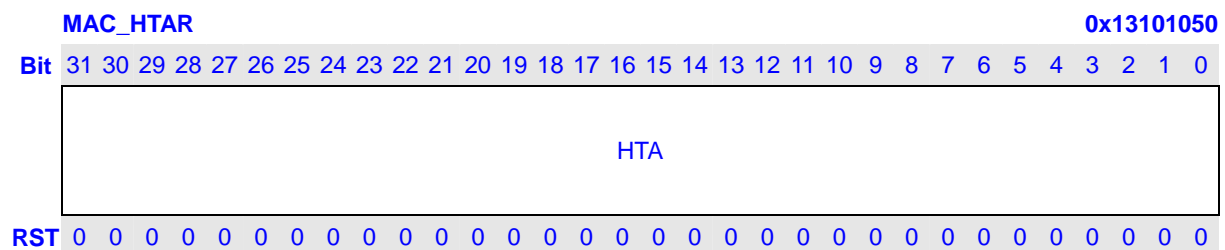
		Receive Interrupt Enable: 1 – When set together with MAC_IER.NI, transmit buffer unavailable interrupt is enabled 0 – Transmit buffer unavailable interrupt is disabled	
1	TS	Transmit Stopped Interrupt Enable: 1 – When set together with MAC_IER.AI, transmit stopped interrupt is enabled 0 – Transmit stopped interrupt is disabled	RW
0	TI	Transmit Interrupt Enable: 1 – When set together with MAC_IER.NI, transmit interrupt is enabled 0 – Transmit interrupt is disabled	RW

### 1.3.9 Missed Frame and Buffer Overflow Counter Register (MAC\_MFC)



Bits	Name	Description	RW
31:28	Reserved		R
27:17	FOC	FOC indicates the number of frames missed by the controller due to host receive FIFO overflow. This counter is incremented each time the controller discards an incoming frame. The counter is cleared when this register is read.	RW
16	Reserved		R
15:0	MFC	MFC is incremented every time the receive DMA gets a frame and the host's descriptor is not available. The counter is cleared when this register is read.	RW

### 1.3.10 Current Host Transmit Buffer Address Register (MAC\_HTAR)



Bits	Name	Description	RW
31:0	HTA	HTA points to the current transmit buffer address being read by the DMA controller.	R

### 1.3.11 Current Host Receive Buffer Address Register (MAC\_HRAR)

MAC\_HRAR

0x13101054

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
	<div>HRA</div>																															
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	HRA	HRA points to the current receive buffer address being written by the DMA controller.	R

### 1.3.12 MAC Control Register (MAC\_CR)

MAC\_CR

0x13100000

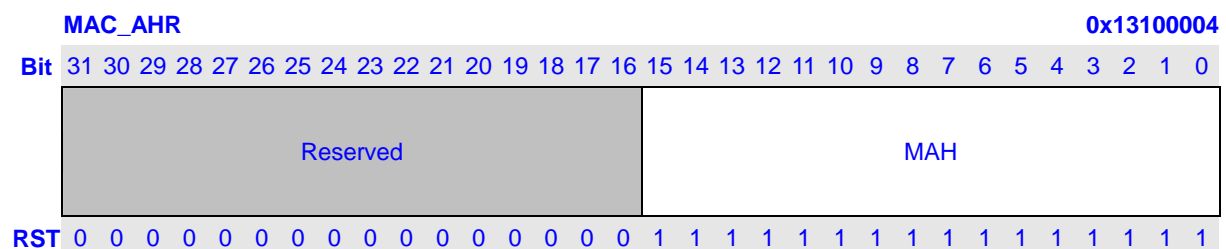
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RA	Reserved	HBD	PS	Reserved	DRO	OM	FDM	PM	PR	IF	PB	HO	Reserved	HP	LCC	DBF	DTRY	Reserved	ASTP	BOLMT	DC	Reserved	TE	RE	Reserved						
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
31	RA	1 – Receive all incoming packets, regardless of the destination address 0 – Do not receive all incoming packets	RW
30:29	Reserved		R
28	HBD	Heart Beat Disable bit. This bit should be set in the MII Mode. 1 – Heartbeat signal quality (SQE) generator function is disabled 0 – Heartbeat signal quality (SQE) generator function is enabled	RW
27	PS	Port Select bit: 1 – The SRL (ENDEC) port is selected for transmit/receive operations on the Ethernet side 0 – The MII port is selected.	RW
26:24	Reserved		R
23	DRO	Disable Receive Own bit. This bit should be reset when the Full Duplex Mode bit is set or the Operating Mode is set to other than 'Normal Mode'. 1 – The MAC disables the reception of frames when the TXEN is asserted	RW

		0 – The MAC receives all the packets that are given by the PHY											
22:21	OM	<div>OM (Loopback Operating Mode) selects the loopback operation modes for the MAC. This is only for FULL Duplex Mode.</div> <table><tr><td>22:21</td><td>Description</td></tr><tr><td>00</td><td>Normal: No loopback</td></tr><tr><td>01</td><td>Internal loopback through MII</td></tr><tr><td>10</td><td>External loopback through PHY</td></tr><tr><td>11</td><td>Reserved</td></tr></table>	22:21	Description	00	Normal: No loopback	01	Internal loopback through MII	10	External loopback through PHY	11	Reserved	RW
22:21	Description												
00	Normal: No loopback												
01	Internal loopback through MII												
10	External loopback through PHY												
11	Reserved												
20	FDM	Full Duplex Mode select: 1 – Full-duplex mode 0 – Half-duplex mode	RW										
19	PM	Pass All Multicast bit. When set, indicates that all the incoming frames with a multicast destination address (first bit in the destination address field is '1') are received.	RW										
18	PR	Promiscuous Mode bit. When set, indicates that any incoming valid frame is received regardless of its destination address.	RW										
17	IF	Inverse Filtering bit. When set, Address Check block operates in the inverse filtering mode. This is valid only during perfect filtering mode.	RW										
16	PB	Pass Bad Frames bit. When set, all incoming frames that passed the address filtering are received, including runt frames, collided frames, or truncated frames caused by Buffer underflow.	RW										
15	HO	Hash Only Filtering Mode bit. When set, the Address Check block operates in the imperfect address filtering mode both for physical and multicast addresses.	RW										
14	Reserved		R										
13	HP	Hash/Perfect Filtering Mode bit. When reset, the Address Check block does a perfect address filter of incoming frames according to the address specified in the MAC Address register. When set, the Address Check block does imperfect address filtering of multicast incoming frames according to the hash table specified in the multicast Hash Table Register. If the Hash Only (HO) is set, then physical addresses are imperfect filtered too. If Hash Only bit (HO) is reset, then physical addresses are perfect address filtered according to the MAC Address Register.	RW										
12	LCC	Late Collision Control bit. 1 – Enable the retransmission of the collided frame 0 – Disable the frame transmission on a late collision	RW										
11	DBF	Disable Broadcast frames bit. 1 – Disable the reception of broadcast frames 0 – Enable the reception of broadcast frames	RW										

10	DTRY	Disable Retry bit. 1 – The MAC will attempt only one transmission 0 – The MAC will attempt 16 transmissions before signaling a retry error	RW										
9	Reserved		R										
8	ASTP	Automatic Pad Stripping bit. 1 – the MAC will strip the pad field on all the incoming frames, if the length field is less than 46 bytes 0 – The MAC will pass all the incoming frames to the host unmodified	RW										
7:6	BOLTM	BOLTM (BackOff Limit) bits allow the user to set its BackOff limit in a relaxed or aggressive mode. <table border="1"><thead><tr><th>7:6</th><th>BackOff Limit</th></tr></thead><tbody><tr><td>00</td><td>10</td></tr><tr><td>01</td><td>8</td></tr><tr><td>10</td><td>4</td></tr><tr><td>11</td><td>1</td></tr></tbody></table>	7:6	BackOff Limit	00	10	01	8	10	4	11	1	RW
7:6	BackOff Limit												
00	10												
01	8												
10	4												
11	1												
5	DC	Deferral Check bit. 1 – The deferral check is enabled in the MAC 0 – The deferral check is disabled in the MAC	RW										
4	Reserved		R										
3	TE	Transmitter Enable bit. 1 – the MAC's transmitter is enabled 0 – the MAC's transmitter is disabled	RW										
2	RE	Receiver Enable bit. 1 – the MAC's Receiver is enabled 0 – the MAC's Receiver is disabled	RW										
1:0	Reserved		R										

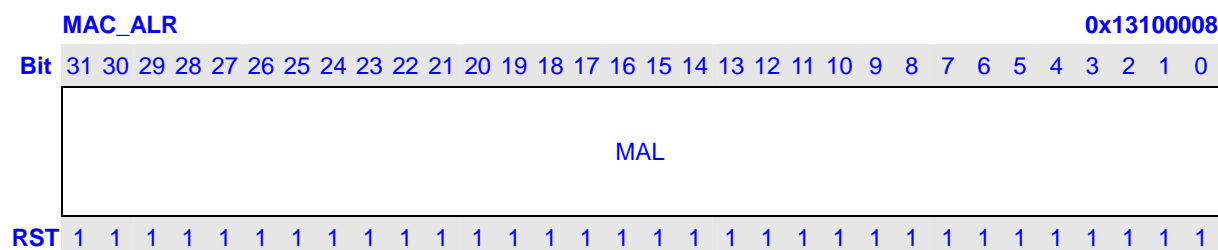
### 1.3.13 MAC Address Hi Register (MAC\_AHR)



Bits	Name	Description	RW
31:16	Reserved		R

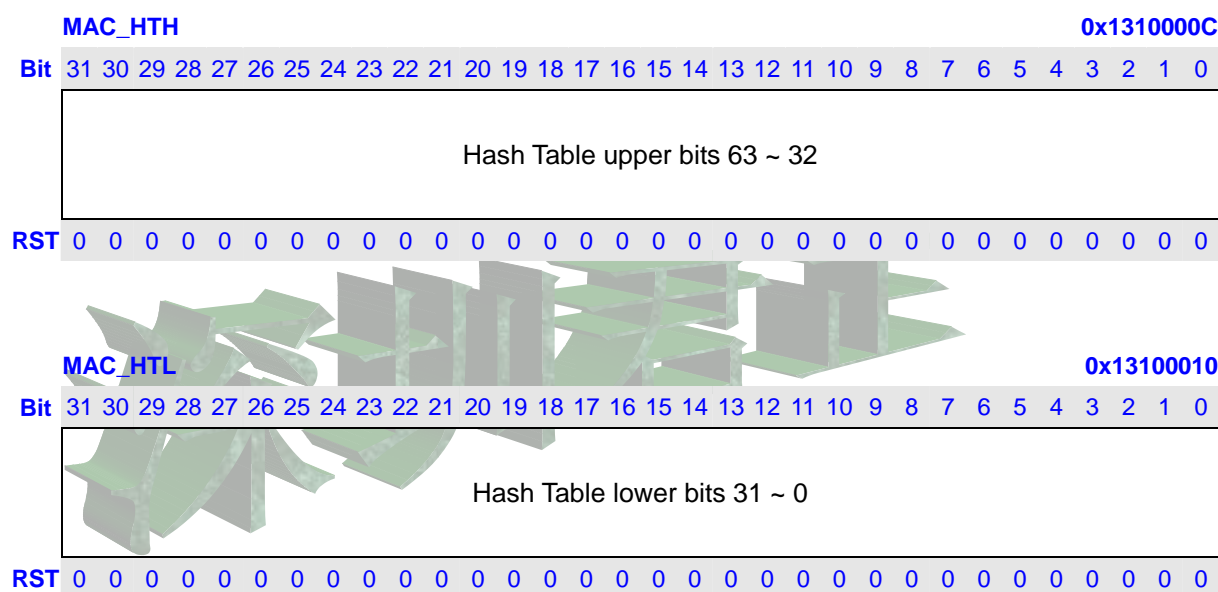
15:0	MAH	MAH contains the upper bits 47 ~ 32 of the MAC Physical Address.	RW
------	-----	--	----

### 1.3.14 MAC Address Lo Register (MAC\_ALR)



Bits	Name	Description	RW
31:0	MAH	MAL contains the lower bits 31 ~ 0 of the MAC Physical Address.	RW

### 1.3.15 Multicast Hash Table Hi/Lo Register (MAC\_HTH/MAC\_HTL)

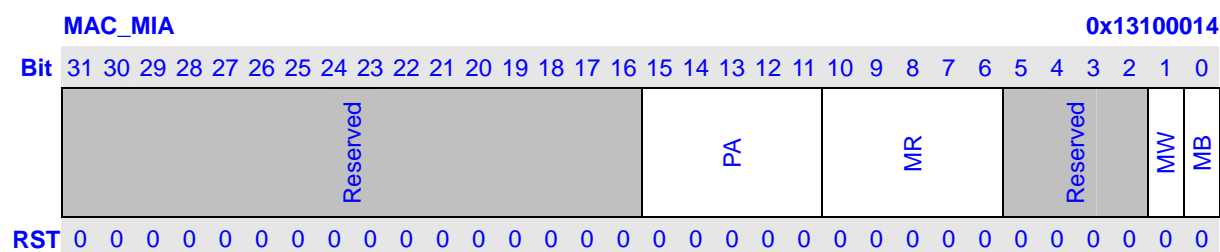


The 64-bit multicast table is used for group address filtering. For hash filtering, the contents of the destination address in the incoming frame is passed through the CRC logic and the upper 6 bits of the CRC register are used to index the contents of the Hash table. The most significant bit determines the register to be used (Hi/Lo), while the other five bits determine the bit within the register. A value of '00000' selects the bit 0 of the selected register and a value of '11111' selects the bit 31 of the selected register.

If the corresponding bit is '1', then the multicast frame is accepted else it is rejected. If the Pass All Multicast is set, then all multi-cast frames are accepted regardless of the multi-cast hash values.

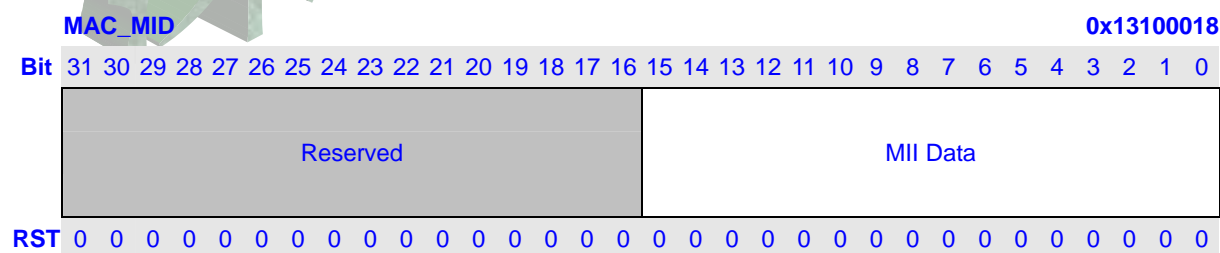
The Multi Cast Hash Table Hi Register contains the higher 32 bits of the hash table and the Multi Cast Hash Table Low Register contains the lower 32 bits of the hash table.

### 1.3.16 MII Address Register (MAC\_MIA)



Bits	Name	Description	RW
31:16	Reserved		R
15:11	PA	PHY Address bits tell which of the 32 possible PHY devices are being accessed.	RW
10:6	MR	MII Register selects the desired MII register in the selected PHY device.	RW
5:2	Reserved		R
1	MW	MII Write bit. Setting this bit tells the PHY that this will be a write operation using the MII data register. If this bit is not set, this will be a read operation, placing the data in the MII data register.	RW
0	MB	MII Busy bit. 1 – A MII read/write access is in progress. 0 – All MII read/write access are finished.	RW

### 1.3.17 MII Data Register (MAC\_MID)



During MII register write process, the MII Data Register contains the data to be written to the PHY register. During MII register read process, the MII Data Register contains the data reading from the PHY register.

### 1.3.18 Flow Control Register (MAC\_FCR)



MAC_FCR																0x1310001C																
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
	PauseTime																Reserved										PCF	FCE	Busy			
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	

The Flow Control Register is used to control the generation and reception of the Control (PAUSE Command) frames by the MAC's Flow control block. A write to register with busy bit set to '1' triggers the Flow Control block to generate a Control frame.

Bits	Name	Description	RW
31:16	PauseTime	PauseTime is the value to be used in the PAUSE TIME field in the control frame.	RW
15:3	Reserved		R
2	PCF	Pass Control Frames bit. 1 – The MAC will decode the control frame (PAUSE) and pass the frame to the Host. 0 – The MAC will decode the control frames but will not pass the frames to the Host.	RW
1	FCE	Flow Control Enable bit. 1 – The MAC is enabled for flow control operation and it will decode all the incoming frames for control frames. 0 – The flow control operation in the MAC is disabled and the MAC does not decode the frames for control frames.	RW
0	Busy	Flow Control Busy bit. 1 – A control frame transfer is in progress. 0 – The control frame transmission has completed.	RW

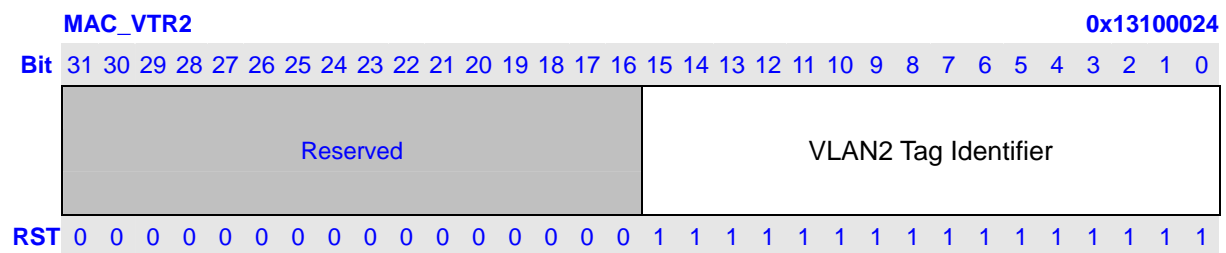
### 1.3.19 VLAN1 Tag Register (MAC\_VTR1)

MAC_VTR1																0x13100020																
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																VLAN1 Tag Identifier															
RST	0	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	

Bits	Name	Description	RW
31:16	Reserved		R
15:0	VTI1	VLAN1 Tag Identifier contains the Length/Type field to identify the VLAN1 frames. The MAC compares the 13th and 14th bytes of the incoming	RW

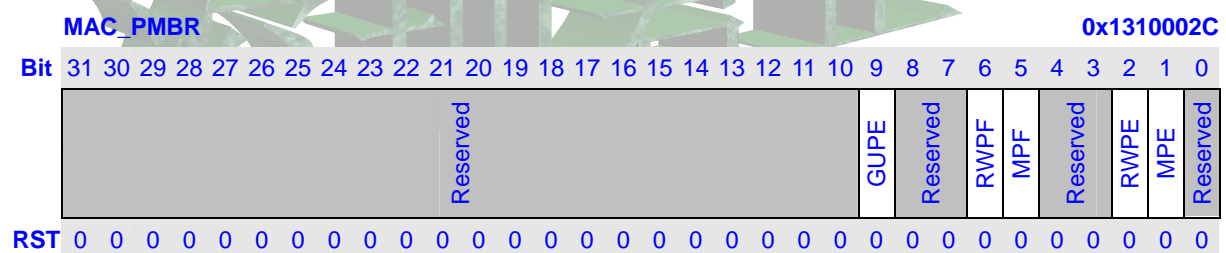
		frame (Length/Type) field and if a match occurs, it sets the VLAN1 tag to the received frame. The legal length of the frame is increased from 1518 bytes to 1522 bytes.	
--	--	---	--

### 1.3.20 VLAN2 Tag Register (MAC\_VTR2)



Bits	Name	Description	RW
31:16	Reserved		R
15:0	VTI2	VLAN2 Tag Identifier contains the Length/Type field to identify the VLAN2 frames. The MAC compares the 13th and 14th bytes of the incoming frame (Length/Type) field and if a match occurs, it sets the VLAN2 tag to the received frame. The legal length of the frame is increased from 1518 bytes to 1522 bytes.	RW

### 1.3.21 PMB Control and Status Register (MAC\_PMBR)



MAC\_PMBR is used to control the Power Management Block (PMB) and to reflect the status.

Bits	Name	Description	RW
31:10	Reserved		R
9	GUPE	Global Unicast Packet Enable bit. 1 – Enable unicast packet in remote wake-up frame detections. 0 – Disable unicast packet in remote wake-up frame detections.	RW
8:7	Reserved		R
6	RWPF	Remote Wake-Up Packet Flag bit. Writing 1 to clear this bit. 1 – A remote wake-up packet is received. 0 – No remote wake-up packet is received.	RW

5	MPF	Magic Packet Flag bit. Writing 1 to clear this bit. 1 – A magic packet is received. 0 – No magic packet is received.	RW
4:3	Reserved		R
2	RWPE	Remote Wake-Up Packet Enable bit. 1 – The PMB is activated and remote wake-up packet is checked. 0 – Remote wake-up packet is not checked.	RW
1	MPE	Magic Packet Enable bit. 1 – The PMB is activated and magic packet is checked. 0 – Magic packet is not checked.	RW
0	Reserved		R

### 1.3.22 Wake-Up Packet Filter Registers (MAC\_WKFR)

Wake-Up Packet Filter Registers are eight 32-bit write-only registers that are used to control the remote wake-up frame detection. The eight registers share the same register address. Software must do eight writes to these registers in the following sequence: Filter 0 Byte Mask Register, Filter 1 Byte Mask Register, Filter 2 Byte Mask Register, Filter 3 Byte Mask Register, Filter Command Register, Filter Offset Register, Filter 0-1 CRC Register, and Filter 2-3 CRC Register.

**Filter 0 Byte Mask Register**

**0x13100028**

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



RST 0

**Filter 1 Byte Mask Register**

**0x13100028**

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



RST 0

**Filter 2 Byte Mask Register**

**0x13100028**

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



RST 0

**Filter 3 Byte Mask Register**

**0x13100028**

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



## Filter 3 Byte Mask

RST 0

Bits	Name	Description	RW
31	ZERO	This bit must be set to zero.	W
30:0	Filter n Byte Mask	Byte Mask bits. If a bit number j of the byte mask is set, byte number patternoffset+j of the incoming frame is processed by the CRC block. Otherwise, byte patternoffset+j is ignored.	W

## Filter Command Register

0x13100028

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	CMD3	Reserved	CMD2	Reserved	CMD1	Reserved	CMD0
----------	------	----------	------	----------	------	----------	------

RST 0

Bits	Name	Description	RW
31:28	Reserved		W
27:24	CMD3	CMD3 is used to control the operation of filter 3. Bit 24: 1 – Filter is enabled, 0 – Filter is disabled. Bit 25-26: Reserved Bit 27: Address type, defines the destination address type of the pattern. 1 – The pattern applies only to multicast frames. 0 – The pattern applies only to unicast frames.	W
23:20	Reserved		W
19:16	CMD2	CMD2 is used to control the operation of filter 2. Bits definitions are the same as CMD3.	W
15:12	Reserved		W
11:8	CMD1	CMD1 is used to control the operation of filter 1. Bits definitions are the same as CMD3.	W
7:4	Reserved		W
3:0	CMD0	CMD0 is used to control the operation of filter 0. Bits definitions are the same as CMD3.	W

## Filter Offset Register

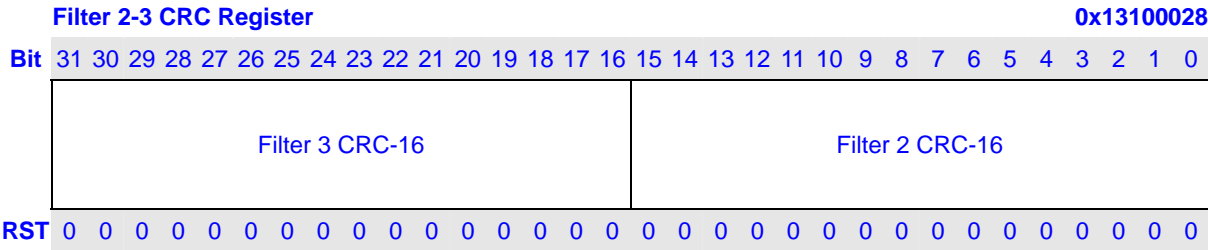
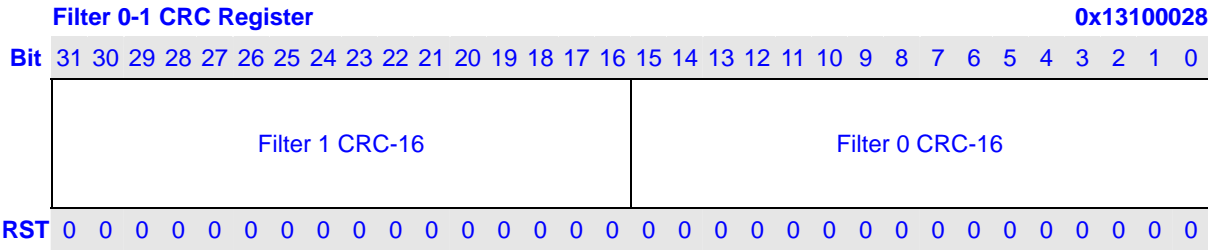
0x13100028

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

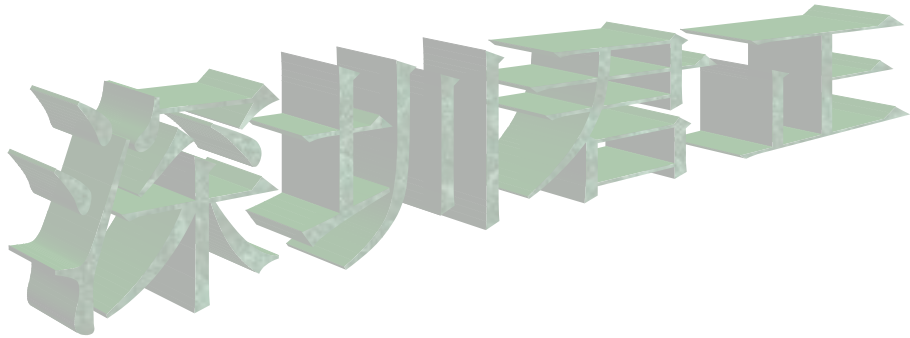
Filter 3 Offset	Filter 2 Offset	Filter 1 Offset	Filter 0 Offset
-----------------	-----------------	-----------------	-----------------

RST 0

Filter i (i=0~3) Offset Register defines the offset in the frame's destination address field from which the frames are examined by filter i. The minimum value of this field needs to be 12, since there should be no CRC check for the Destination Address and the Source Address Fields.



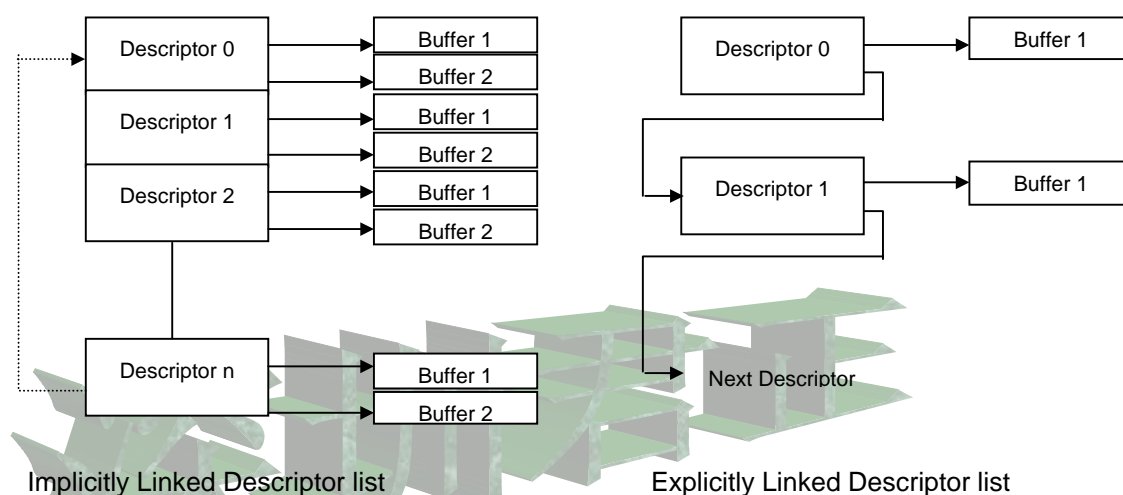
Filter i (i=0~3) CRC Register contains the CRC-16 result of the frame that should pass filter i. These values are compared against the CRC calculated on the incoming frame, and a match indicated the reception of a wake-up frame.



## 1.4 DMA Descriptors and Buffers

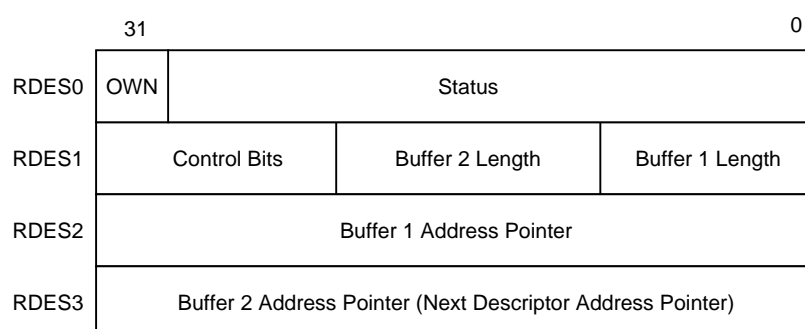
The Ethernet MAC controller uses a dedicated DMA controller to transfer data between the Ethernet MAC module and the host memory. The data in the host memory are described in the descriptor list format. The descriptor lists reside in the host physical memory address space. Each descriptor can point to maximum of two buffers. The Receive and Transmit Descriptors addresses must be WORD (32-bit) aligned. Receive data buffers always have to be WORD aligned, while transmit data buffers do not have any restriction on data alignment.

A descriptor list is forward linked, either implicitly or explicitly. Explicit chaining of descriptors is accomplished by setting the second address chained in both receive and transmit descriptors (RDES\_1[24] and TDES\_1[24]). The last descriptor may point back to the first descriptor to create a ring structure. Descriptors of a Zero Buffer length are not supported at the initial and Final Descriptor of a Chain. Following figures illustrate the descriptor organization.



**Figure 1-1 Descriptor Ring and Chain Structure**

### 1.4.1 Receive Descriptors Format



**Figure 1-2 Receive Descriptor Format**

### 1.4.1.1 Receive Descriptor 0 (RDES0)

#### RDES0

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
	OWN	FF															ES	DE	Reserved	LE	FR	MF	FS	LS	FTL	CS	FT	WT	ME	DB	CE	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	R/W
31	OWN	OWN bit. The DMA controller clears this bit either when it completes the frame reception or when the buffers that are associated with this descriptor are full. 1 – The descriptor is owned by the DMA controller 0 – The descriptor is owned by the host	RW
30	FF	Filtering Failure bit. 1 – The destination address field in the current frame failed the address filtering. 0 – The destination address field in the current frame passed the address filtering.	RW
29:16	FL	FL (Frame Length) indicates the length in bytes of the received frame that was transferred to host memory, Including the CRC. This field is valid only when the last descriptor (RDES_0[8]) is set and the descriptor error (RDES_0[14]) is reset.	RW
15	ES	ES (Error Summary) indicates the logical OR of the following bits: [1] CRC Error [6] Collision Seen [7] Frame too long [11] Runt Frame [14] Descriptor Error This bit is the generic Error Status bit set by the DMA controller, and is valid only when the last descriptor (RDES_0[8]) is set.	RW
14	DE	Descriptor Error bit. When set, indicates a frame truncation caused by a frame that does not fit within the current descriptor buffers, and that the DMA controller does not own the next descriptor. The frame is truncated. This field is valid only when the last descriptor (RDES_0[8]) is set.	RW
13	Reserved		R
12	LE	Length Error bit. When set, indicates that for the current frame the Length value is inconsistent with the total number of bytes received in the current frame.	RW

		This is valid when the Frame Type is set to '0' (802.3 Frame). This field is valid only when the last descriptor (RDES_0[8]) is set.	
11	FR	<p>Runt Frame bit.</p> <p>When set, indicates that this frame was damaged by a collision or premature termination before the collision window (64 bytes default) passed. Runt frames are passed on to the host only if the pass bad frames bit CSR6[3] is set. This field is valid only when the last descriptor (RDES_0[8]) is set.</p>	RW
10	MF	<p>Multicast Frame bit.</p> <p>When set, it indicates that the Destination Address in the current frame is a multicast address, i.e., the first bit of the DA is 1. This field is valid only when the last descriptor (RDES_0[8]) is set.</p>	RW
9	FS	<p>First Descriptor bit.</p> <p>When set, indicates that this descriptor contains the first buffer of the frame. If the buffer size of the first buffer is 0, the second buffer contains the beginning of the frame. If the buffer size of the second is also 0, the second descriptor contains the beginning of the frame.</p>	RW
8	LS	<p>Last Descriptor bit.</p> <p>When set, indicates that the buffers pointed to by this descriptor are the last buffers of the frame.</p>	RW
7	FTL	<p>Frame too Long bit.</p> <p>When set, indicates that the frame length exceeds the maximum Ethernet specified size of 1518 bytes. Frame too long is only a frame length indication and does not cause any frame truncation. This field is valid only when the last descriptor (RDES_0[8]) is set.</p>	RW
6	CS	<p>Collision Seen bit.</p> <p>When set, indicates that the frame was damaged by a collision that occurred after the 64 bytes following the start of frame delimiter (SFD). This is a late collision. This field is valid only when the last descriptor (RDES_0[8]) is set.</p>	RW
5	FT	<p>Frame Type bit.</p> <p>When set, indicates that the frame is an Ethernet-type frame (frame length field is greater than 1500 bytes). When clear, indicates the frame is an IEEE 802.3 frame. This bit is not valid for runt frames of less than 14 bytes. This field is valid only when the last descriptor (RDES_0[8]) is set.</p>	RW
4	WT	<p>Watchdog Timeout bit.</p> <p>The watchdog timer monitors the time of each packet reception. If a single packet reception exceeds ( 2048 bytes) the watchdog circuitry disables the receive path. This field is valid only when the last descriptor (RDES_0[8]) is set.</p>	RW
3	ME	<p>MII Error bit.</p> <p>When set, indicates that the MII RX_ER was asserted during the reception of this frame. This field is valid only when the last descriptor</p>	RW



		(RDES_0[8]) is set.	
2	DB	Dribbling Bit. When set, indicates that the frame contained a non-integer multiple of 8 bits. This bit is not valid if either collision seen or runt frame bits are set. If set, and the CRC error is reset, then the packet is valid. This field is valid only when the last descriptor (RDES_0[8]) is set.	RW
1	CE	CRC Error bit. When set, indicates that a cyclic redundancy check (CRC) error occurred on the received frame. This bit is also set when the MII RX_ER pin is asserted during the reception of a receive packet even though the CRC may be correct. This field is valid only when the last descriptor (RDES_0[8]) is set.	RW
0	Reserved		R

#### 1.4.1.2 Receive Descriptor 1 (RDES1)

##### RDES1

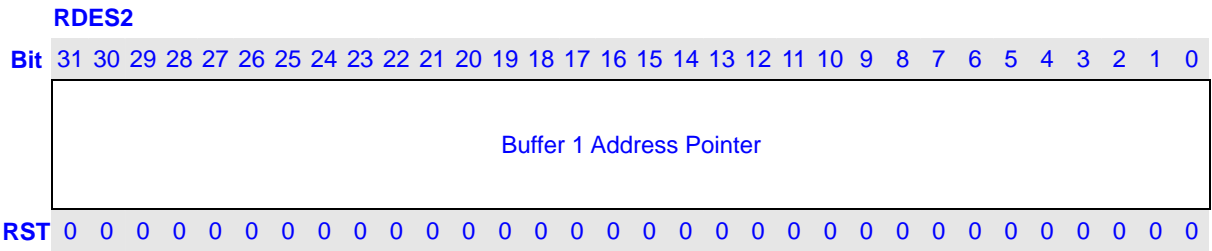
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved						RER	RCH	Reserved	RBS2												RBS1										
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
31:26	Reserved		R
25	RER	Receive End of Ring bit. When set, indicates that the descriptor list reached its final descriptor. The DMA controller returns to the base address of the list, creating a descriptor ring.	RW
24	RCH	Second Address Chained bit. When set, indicates that the second address in the descriptor is the next descriptor address, rather than the second buffer address. When this field is set RBS_2 (RDES_1[20:10]) should be all zeros. RDES_1[25] takes precedence over RDES_1[24].	RW
23:22	Reserved		R
21:11	RBS2	Receive Buffer 2 Size. Indicates the size, in bytes, of the second data buffer. This field is not valid if RDES_1[24] is set.	RW
10:0	RBS1	Receive Buffer 1 Size. Indicates the size, in bytes, of the first data buffer. If this field is 0, the DMA controller ignores this buffer and uses buffer 2. (This field cannot be zero if the descriptor chaining is used – Second Address Chained (RDES_1[24])	RW

		is set).	
--	--	----------	--

1.4.1.3 Receive Descriptor 2 (RDES2)

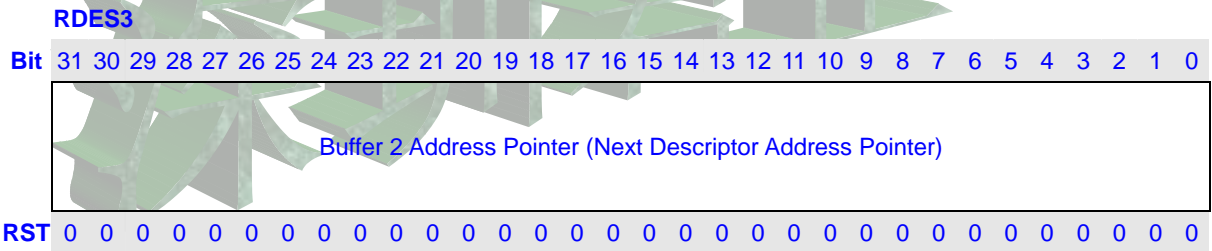
RDES2 indicates the physical address of the data buffer 1. There are no limitations on the buffer address alignment.



1.4.1.4 Receive Descriptor 3 (RDES3)

RDES3 indicates the physical address of the data buffer 2 when the descriptor chain is used. There are no limitations on the buffer address alignment.

If the Second Address Chained (RDES1[24]) bit is set, then this address contains the pointer to the physical memory where the next descriptor is located. The address of next descriptor must be WORD (32-bit) aligned.



## 1.4.2 Transmit Descriptor Format

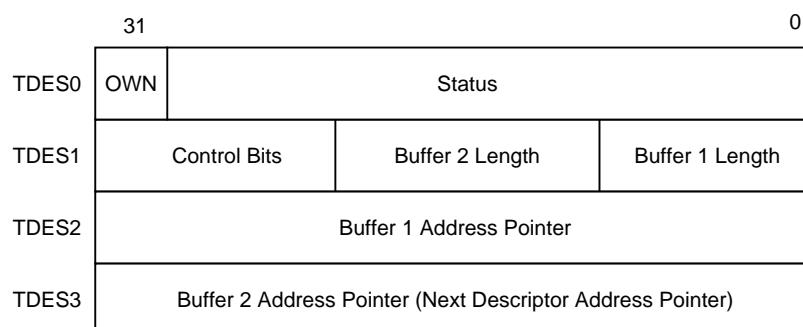
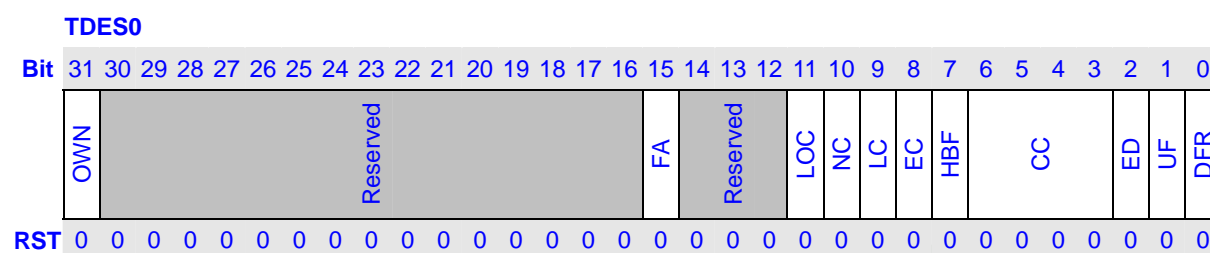


Figure 1-3 Transmit Descriptor Format

### 1.4.2.1 Transmit Descriptor 0 (TDES0)



Bits	Name	Description	R/W
31	OWN	Owner bit. The DMA controller clears this bit either when it completes the frame transmission or when the buffers allocated in the descriptor are empty.  1 – The descriptor is owned by the DMA controller 0 – The descriptor is owned by the host	RW
30:16	Reserved		R
15	FA	Frame Aborted bit.  When set, it indicates that the Host has aborted the transmission of the current frame due to one of the following conditions: <ul style="list-style-type: none"> <li>– No Carrier</li> <li>– Loss of Carrier</li> <li>– Excessive Deferral</li> <li>– Late Collision</li> <li>– Retry Count exceeds the attempt Limit</li> <li>– Data under run</li> </ul> If this bit is reset, it indicates that the current frame was successfully transmitted onto the MII Interface.	RW
14:12	Reserved		R

11	LOC	<p>Loss of Carrier bit.</p> <p>When set, indicates that the loss of carrier occurred during the frame's transmission (i.e., the MII CRS input was inactive for one or more bit times when the frame is being transmitted). This is valid only for the frames transmitter without collision and when the MAC is operating in half duplex mode.</p>	RW
10	NC	<p>No Carrier bit.</p> <p>When set, indicates that the carrier signal from the transceiver was not present during transmission.</p>	RW
9	LC	<p>Late Collision bit.</p> <p>When set, indicates that the frame transmission was aborted due to collision occurring after the collision window of 64 bytes. Not valid if under run error is set.</p>	RW
8	EC	<p>Excessive Collisions bit.</p> <p>When set, indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current frame. If the DTRY (bit 10 of MAC Control Register) is asserted, this bit is set after the first collision and the transmission of the frame will be aborted.</p>	RW
7	HBF	<p>Heart Beat Failure bit.</p> <p>This is effective only in 10-Mb/s operation mode and the Serial port is selected. When set, this bit indicates a heartbeat collision check failure (the transceiver failed to return a collision pulse as a check after the transmission). This bit is not valid if underflow error bit is set.</p>	RW
6:3	CC	<p>Collision Count bits.</p> <p>The 4-bit counter indicates the number of collisions that occurred before the frame was transmitted. Not valid when the Excessive Collisions bit is set.</p>	RW
2	ED	<p>Excessive Deferral bit.</p> <p>When set, indicates that the transmission has ended because of excessive deferral of over 24,288 bit times during the transmission, if the defer bit is set high in the control register.</p>	RW
1	UF	<p>Underflow Error bit.</p> <p>When set, indicates that the transmitter aborted the frame because data arrived late from memory. Underflow error indicates that the DMA controller encountered an empty transmit buffer while transmitting the frame. The transmission process enters the suspended state and sets both transmit underflow (MAC_SR[5]) and transmit interrupt (MAC_SR[0]).</p>	RW
0	DFR	<p>Deferred bit.</p> <p>When set, indicates that the transmitter had to defer while ready to transmit a frame because the carrier was asserted.</p>	RW

### 1.4.2.2 Transmit Descriptor 1 (TDES1)

#### TDES1

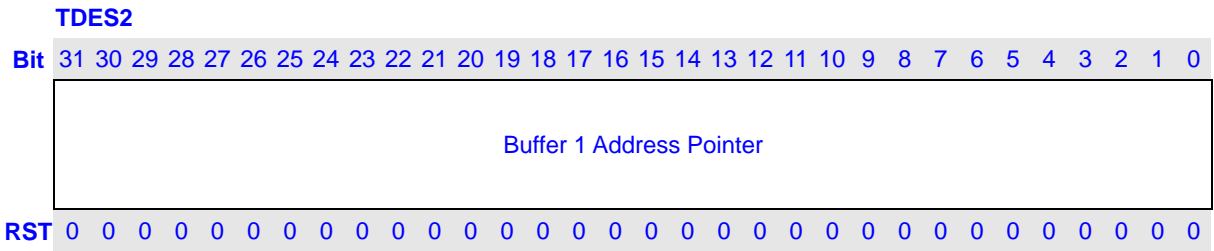
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	IOC	LS	FS	Reserved		ACD	TER	TCH	DPD	Reserved																						
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	R/W
31	IOC	Interrupt on Completion bit. When set, the sets transmit interrupt (MAC_SR[0]) after the present frame has been transmitted.	RW
30	LS	Last Segment bit. When set, indicates that the buffer contains the last segment of the frame.	RW
29	FS	First Segment bit. When set, indicates that the buffer contains the first segment of a frame.	RW
28:27	Reserved		R
26	ACD	Add CRC Disable bit. When set, the does not append the cyclic redundancy check (CRC) to the end of the transmitted frame. This is valid only when the first segment (TDES_1[29]) is set.	RW
25	TER	Transmit End of Ring bit. When set, indicates that the descriptor list reached its final descriptor. The returns to the base address of the list, creating a descriptor ring.	RW
24	TCH	Second Address Chained bit. When set-indicates that the second address in the descriptor is the next descriptor address, rather than the second buffer address. When this bit is set, the TBS2 (TDES1[2:11]) should be all zeros. TDES1[25] takes precedence over TDES1[24].	RW
23	DPD	Disable Padding bit. When set, the does not automatically add a padding field, to a packet shorter than 64 bytes. When reset, the DMA controller automatically adds a padding field and also a CRC field to a packet shorter than 64 bytes and the CRC field is added despite the state of the add CRC disable (TDES1[26]) flag. This is valid only when the first segment (TDES1[2]) is set.	
22	Reserved		R
21:11	TBS2	Transmit Buffer 2 Size. Indicates the size, in bytes, of the second data buffer. This field is not valid if TDES1[24] is set.	RW
10:0	TBS1	Transmit Buffer 1 Size. Indicates the size, in bytes, of the first data buffer. If this field is 0, the DMA	RW

		controller ignores this buffer and uses buffer 2.	
--	--	---	--

1.4.2.3 Transmit Descriptor 2 (TDES2)

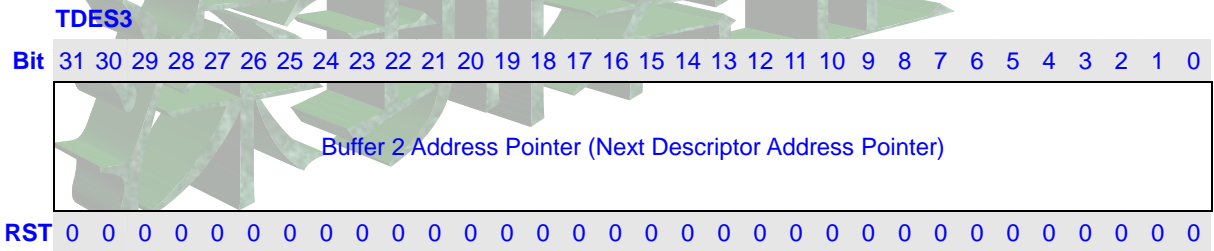
TDES2 indicates the physical address of the data buffer 1. There are no limitations on the buffer address alignment.



1.4.2.4 Transmit Descriptor 3 (TDES3)

TDES3 indicates the physical address of the data buffer 2 when the descriptor chain is used. There are no limitations on the buffer address alignment.

If the Second Address Chained (TDES1[24]) bit is set, then this address contains the pointer to the physical memory where the next descriptor is located. The address of next descriptor must be WORD (32-bit) aligned.



## 1.5 Ethernet MAC Operations

The Ethernet MAC is supported with a dedicated DMA controller. Once configured, the Ethernet MAC receives and transmits packets with minimal software intervention. In the following sections, Transmit and Receive Operations are discussed.

### 1.5.1 Transmit Operation

#### 1.5.1.1 Initialization

The software is responsible for the initialization of the Ethernet MAC controller. After setting the mode of operation of the Ethernet MAC controller, the software creates the “Transmit” descriptors for the transmit frame, and sets the “OWN” bit in the transmit descriptor 0. Then it initiates the transmit process by setting the MAC\_OMR Start/Stop (ST) bit.

#### 1.5.1.2 Transmit Process

Once the ST bit is set, the DMA controller will obtain the descriptor located in the host memory at the address written in the MAC\_TBAR register. If the OWN bit in the descriptor is not set, the DMA will assume that the Host is still busy with this descriptor, and will go into suspend state. If the OWN bit is set, then the DMA will obtain the pointer to the frame data from the descriptor, and download the data from the Host memory to the TX FIFO. On Completion, the MAC and DMA update the MAC status register and the TDES0 status bits, and the host is interrupted to check for the status of the transfer.

#### 1.5.1.3 Transmit Status

Once the transmission is completed, the Ethernet MAC will update the DMA status by writing the MAC\_SR register and the status bits in the TDES0 register. Then it interrupts the Host by writing to MAC\_IER register, depending on whether the corresponding interrupt enable bit is set in MAC\_IER.

### 1.5.2 Receive Operation

#### 1.5.2.1 Initialization

The software is responsible for the initialization of the Ethernet MAC controller. After setting the mode of operation of the Ethernet MAC controller, the software creates the “Receive” descriptors for the receive frame, and sets the “OWN” bit in the receive descriptor 0. Then it initiates the receive process by setting the MAC\_OMR Start/Stop (ST) bit.

### 1.5.2.2 Receive Process

Once the ST bit is set, the DMA is in the running mode, and will always acquire an extra descriptor for it received frame. If the “OWN” bit is not set in the Descriptor's Status field, the DMA Receive engine will go into suspend mode until the software writes to the MAC\_RPDR register. If the “OWN” bit was found to be set, then the DMA will transfer the data from the RX FIFO to the “Receive” Descriptors' data buffer. On Completion, the MAC and DMA update the MAC status register and the RDES0 status bits, and the host is interrupted to check for the status of the transfer.

### 1.5.2.3 Receive Status

Once the receive is completed, the Ethernet MAC will update the DMA status by writing the MAC\_SR register and the status bits in the RDES0 register. Then it interrupts the Host by writing to MAC\_IER register, depending on whether the corresponding interrupt enable bit is set in MAC\_IER.

## 1.5.3 MAC Interrupts

There are a number of events that may cause interrupts. MAC\_SR register contains all the bits that might cause interrupts. MAC\_IER contains an enable bit for each of the events that can cause interrupt. Interrupt signal is asserted if any of the interrupt conditions are satisfied.

There are two groups of interrupts, normal and abnormal interrupts as outlined in MAC\_SR. Interrupts are cleared by writing a '1' to the bit position. When all the enabled interrupts within a group are cleared, the corresponding summary bit is cleared. When both of the summary bits are cleared the interrupt signal is de-asserted.

An interrupt is generated only once for simultaneous, multiple events. The software must scan MAC\_SR for the interrupt cause. The interrupt is not generated again, unless a new interrupting event occurs after the software has cleared the appropriate MAC\_SR bit.



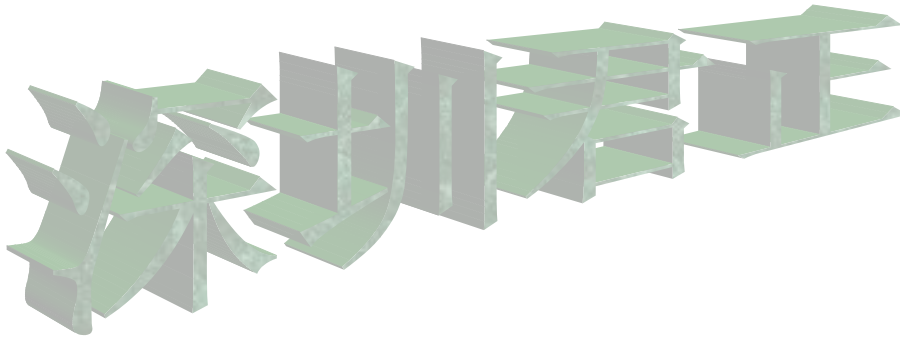
## 1.6 MII Management

### 1.6.1 MII Write Operation

To write to any register in the PHY, the software sets the PHY address, the MII register address, the MII Write bit in the MII Address Register and the writing data in the MII Data Register, then set the MII Busy bit to 1 in the MII Address Register. The MAC at this point starts a write operation on the MII Management Interface using the Management Frame Format specified in the MII Specification. The software should not change these registers while the transaction is going on. After the write operation is completed, the MAC will clear the MII Busy bit to 0.

### 1.6.2 MII Read Operation

To read any register in the PHY, the software sets the PHY address, the MII register address, and clear the MII Write bit in the MII Address Register, then set the MII Busy bit to 1 in the MII Address Register. The MAC at this point starts a read operation on the MII Management Interface using the Management Frame Format specified in the MII Specification. The software should not change these registers while the transaction is going on. After the read operation is completed, the MAC will clear the MII Busy bit and presents the read data in the MII Data Register.



## 1.7 Power Management Block

The power management block is activated, by setting the RWPE or MPE bit of the PMB Control and Status register. When the bits are set, all the receive traffic to the MAC is blocked. The PMB module responds internally to the MAC RX module for every frame and checks for either the Wake-Up Frame or the Magic Packet. Once either one of the frame is detected, the PMB module causes an interrupt to the system. If system is in standby mode, standby mode will be waked-up.

A network wake-up frame is typically a frame that is sent by existing network protocols such as ARP requests or IP frames addressed to the machine. Before putting the device in to Standby Mode, the host should program the byte masks in the wake-up filter registers. Each byte mask defines which bytes of the incoming frame should be compared with the corresponding sample frame in order to determine whether or not to accept an incoming frame as wake-up event.

The Magic Packet frame is based on the mechanism that use Advanced Micro Device's Magic Packet Technology to power up the sleeping device on the network. In this mechanism when the MAC receives a specific packet of information, called a Magic Packet, addressed to the node on the network. Magic packets that pass the address filtering (physical or broadcast) will be checked to meet the data format with the MAC address appearing 16 times.

### 1.7.1 Wake-Up Frame Detection

Wake-Up Frame Detection is enabled by setting the RWPE bit of the PMB Control and Status register. Before enable Wake-Up Detection, software should configure the PMB with a list of sample frames and the corresponding byte masks.

An incoming frame is recognized as a wake-up frame if it passes the MAC address filtering and PMB CRC calculation result match in the Wake-Up Frame register. In order to know which bytes of the frames should be checked by the CRC module, the PMB uses a programmable byte mask and a programmable pattern offset for each of four supported filters. The pattern's offset defines the location of the first byte that should be checked in the frame. Since the destination address is checked by the address filtering block, the pattern offset is always greater than 12. The byte mask is a 31-bit field that specifies for each of the 31 contiguous bytes with the frame, beginning in the pattern offset, whether or not it should be checked. If bit  $j$  in the byte mask is set, byte offset  $+ j$  in the frame is checked.

### 1.7.2 Magic Packet Detection

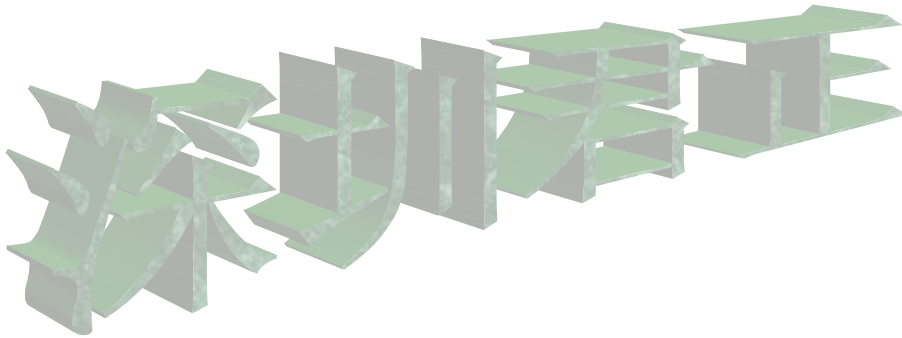
Magic Packet Detection is enabled by setting the MPE bit of the PMB Control and Status register. Only packets with the MAC address or a broadcast address will be checked to meet the Magic

Packet requirement. Each frame received is checked for a synchronization stream, 48'h FF\_FF\_FF\_FF\_FF\_FF pattern after the Destination and Source Address field. Then 16 repetitions of the MAC address without any breaks or interruptions are looked for in the frame. In case of a break in the 16 repetitions, the 48'h FF\_FF\_FF\_FF\_FF\_FF pattern is scanned for again. The 16 repetitions can be anywhere in the frame but must be preceded by the synchronization stream. The device will also accept a multicast frame, as long as the 16 duplications of the MAC address is detected. In case the MAC address is 48'h 00\_11\_22\_33\_44\_55, then the valid Magic Packet should be:

```

Destination Address Source Address ... .. FF FF FF FF FF FF
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
... CRC

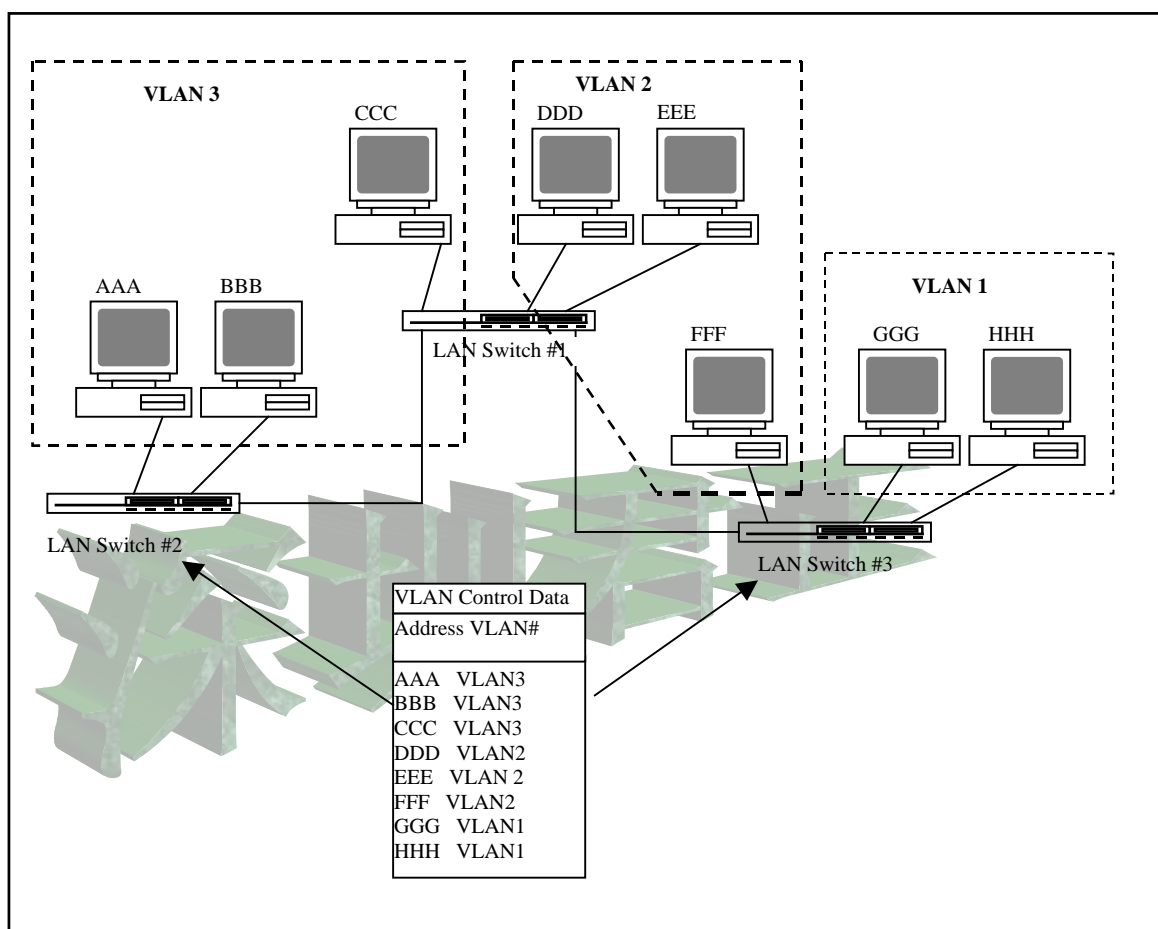
```



## 1.8 Virtual Local Area Network (VLAN)

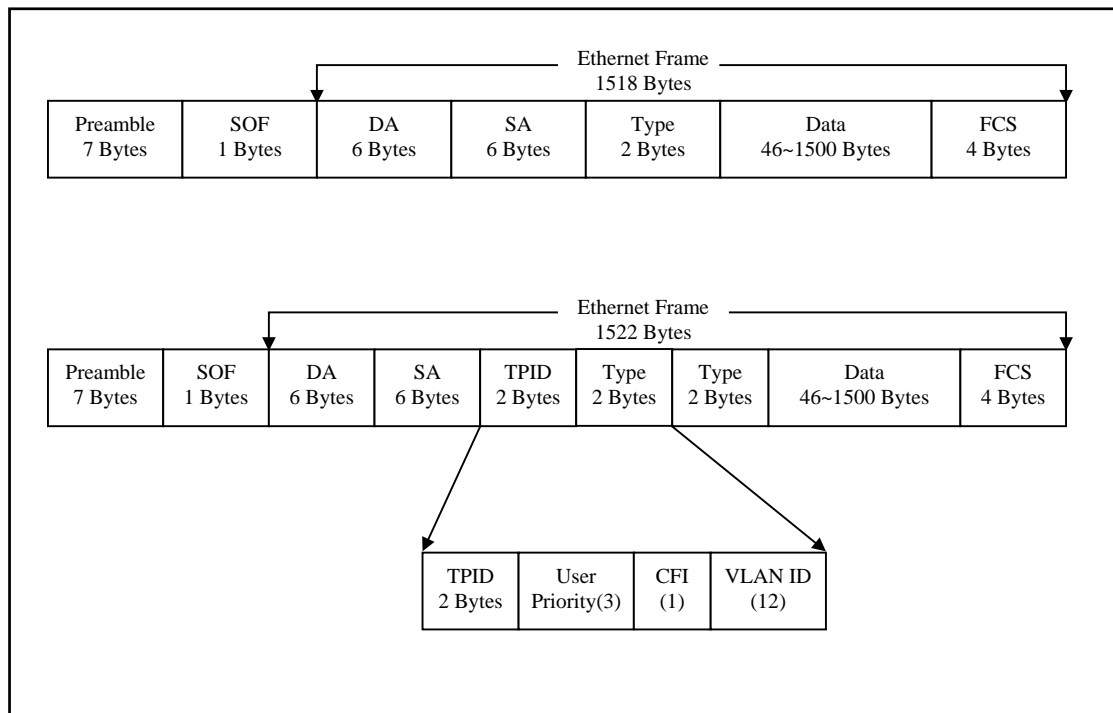
VLAN is a means to form a broadcast domain without the restriction on the physical or geographical location on the members of that domain. VLAN can be implemented in any number of different factors such as: Physical Port, MAC Address, Layer-3 unicast address, Multicast Address, Date/Time in combination with MAC Address, etc.

An example of a VLAN is described in the following figure. It demonstrates the freedom from physical constraint on the network, and the ability to divide a single switched network into a smaller broadcast domain.



**Figure 1-4 VLAN Topology**

When the members of a VLAN are not located on the same physical medium, the VLAN uses a Tag to help it determined how to forward the frame from one member to another. The Tag structure has been proprietary until IEEE has released a supplement to the 802.3 frames where the VLAN Frame Structure including the tag has been defined. This new frame structure for VLAN is depicted in the following figure.



**Figure 1-5 VLAN Frame**

The MAC block recognizes transmit and received frames that are tagged with either one-level or two-level VLAN IDs. The MAC compares the thirteenth and fourteenth bytes of transmit and receive frames to the contents of both the one-level VLAN tag register and two-level VLAN tag register. If a match is found, the MAC block identifies the frame as either a one-level or two-level VLAN frame, depending on where the match was found. Upon recognizing that a frame has a VLAN tag, counter thresholds are adjusted to account for the extra bytes that the VLAN tag adds to the frame. That is the maximum length of the good packet is changed from 1518 bytes to 1522 bytes.

# 1 AC97/I2S Controller

## 1.1 Overview

This chapter describes the AIC (AC'97 and I<sup>2</sup>S Controller) included in the Jz47xx 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 Jz47xx processor system memory and 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 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.

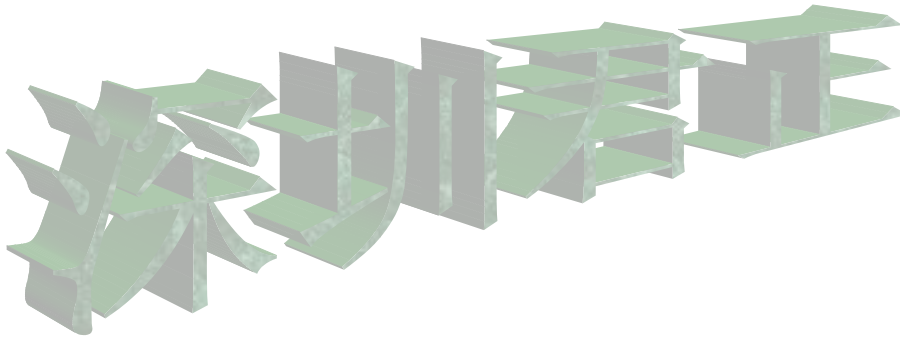
For I2S systems that support the L3 control bus protocol, additional pins are required to control the external CODEC. CODECs that use an L3 control bus require 3 signals: L3\_CLK, L3\_DATA, and L3\_MODE for writing bytes into the L3 bus register. The AIC supports the L3 bus protocol via software control of the general-purpose I/O (GPIO) pins. The AIC does not provide hardware control for the L3 bus protocol.

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. This doc documented both AIC V1 and AIC V2. Notes are used to describe the difference. AIC in Jz47xx is V1.

## 1.2 Features

AIC support following AC97/I2S features:

- 8, 16, 18, 20 and 24 bit audio sample data sizes supported. For V1, 24 bit sample size only supported in I2S/MSB-justified format
- 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 (V2)
- Support endian switch on 16-bits audio samples play back (V2)
- In AC-link format, V1 support only 48k fixed sample rate where V2 can support variable sample rate
- 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
- Two FIFOs for transmit and receive respectively with 16 (for V1) or 32 (for V2) samples capacity in every direction.



### 1.3 Signal Descriptions

There are all 6 pins used to connect between AIC and an external audio CODEC device. They are listed and described in Table 1-1.

**Table 1-1 AIC Pins Description**

Name	I/O	Description
RESET#	O	Only for AC-link format. Active-low CODEC reset.
SYS_CLK	O	I2S/MSB-Justified formats only. Supply 2.048 MHz ~ 24.576 MHz 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-Justified.
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

#### 1.3.1 RESET# 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.

#### 1.3.2 SYS\_CLK Pin

SYS\_CLK outputs the system clock to CODEC. This pin is useful only in I2S/MSB-justified format master mode. It generates a frequency between approximately 2.048 MHz and 12.288 (V1) or 24.576 (V2) MHz by dividing down the PLL clock with a programmable divisor. For V1, this frequency is always 256 times of the audio sampling frequency, and for V2, it can be 256, 384, 512 and etc. times of the audio sampling frequency. If AIC is disabled, it retains the high. If AIC is not in I2S/MSB-justified format master mode, SYS\_CLK pin can be used for a GPIO pin.

#### 1.3.3 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 1-5 (V1) and Table 1-6 (V2) list 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.



### 1.3.4 SYNC Pin

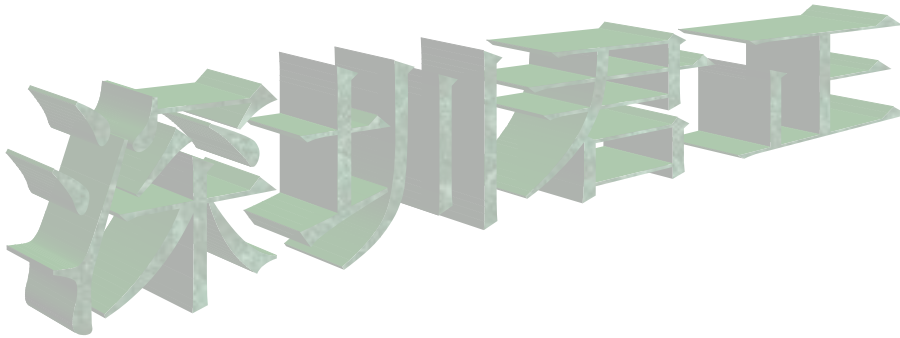
In AC-link format, SYNC provides frame synchronization, fixed to 48kHz, by specifying beginning of an audio sample frame and outputs to CODEC. In I2S/MSB-Justified formats, SYNC is used to indicate left- or right-channel sample data and toggled in sample rate frequency. It outputs to CODEC in master 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.

### 1.3.5 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.

### 1.3.6 SDATA\_IN Pin

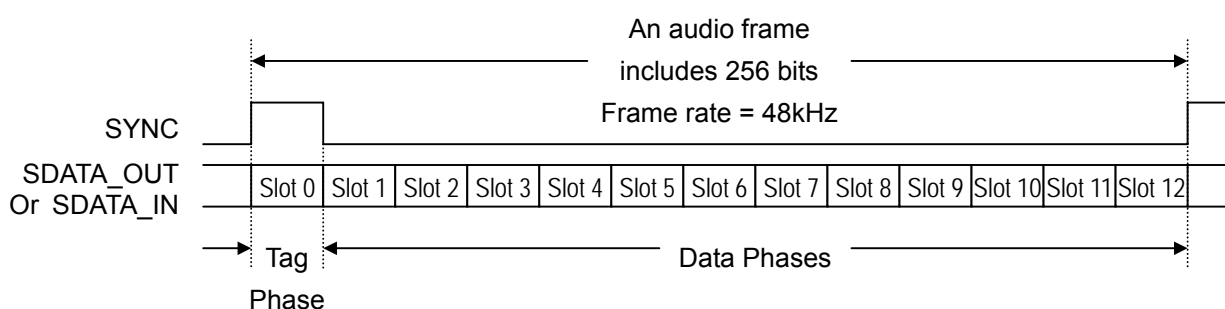
SDATA\_IN is AIC inputs data pin, which inputs serial audio data or data of AC97 CODEC register status from an external audio CODEC device. If AIC is disabled, its state is undefined.



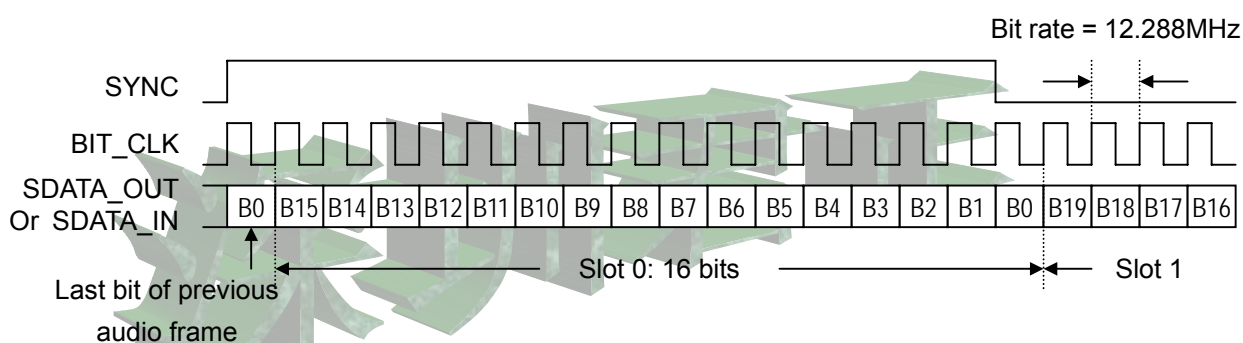
## 1.4 Serial Interface Protocol

### 1.4.1 AC-link serial data format

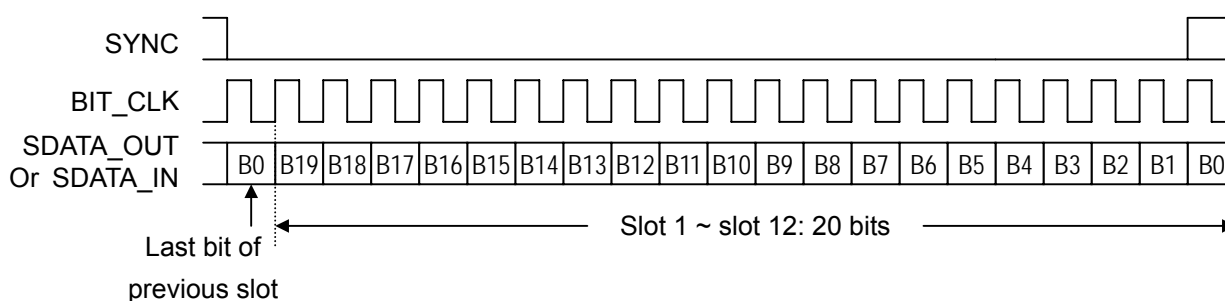
Following figures are AC-link serial data format. Audio data is MSB adjusted, regardless of 8, 16, 18, 20, 24 bits sample size. 24 bits sample size is support only in V2. When a 24-bits sample is transmitted, the LSB 4-bits are truncated. When try to record 24-bits sample, 4-bits of 0 are appended in 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 1-1 AC-link audio frame format**



**Figure 1-2 AC-link tag phase, slot 0 format**



**Figure 1-3 AC-link data phases, slot 1 ~ slot 12 format**

### 1.4.2 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.

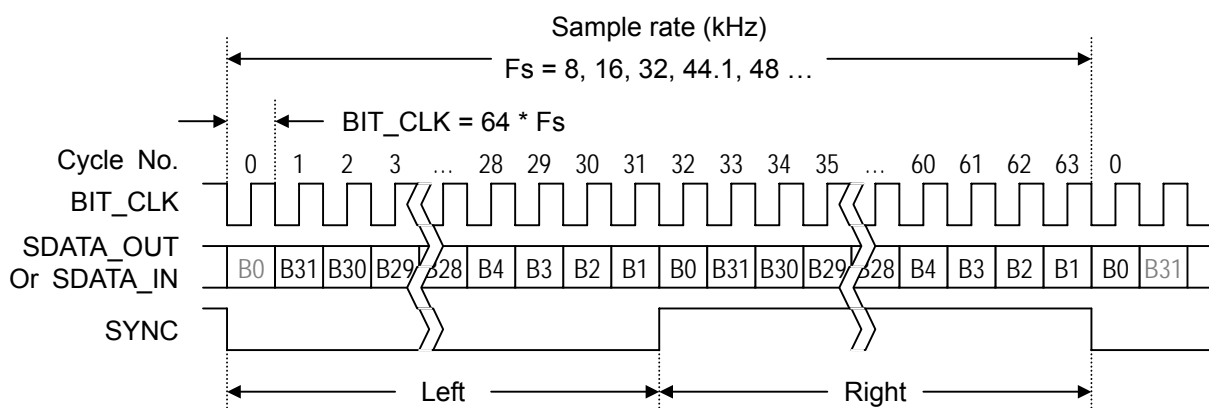


Figure 1-4 I2S data format

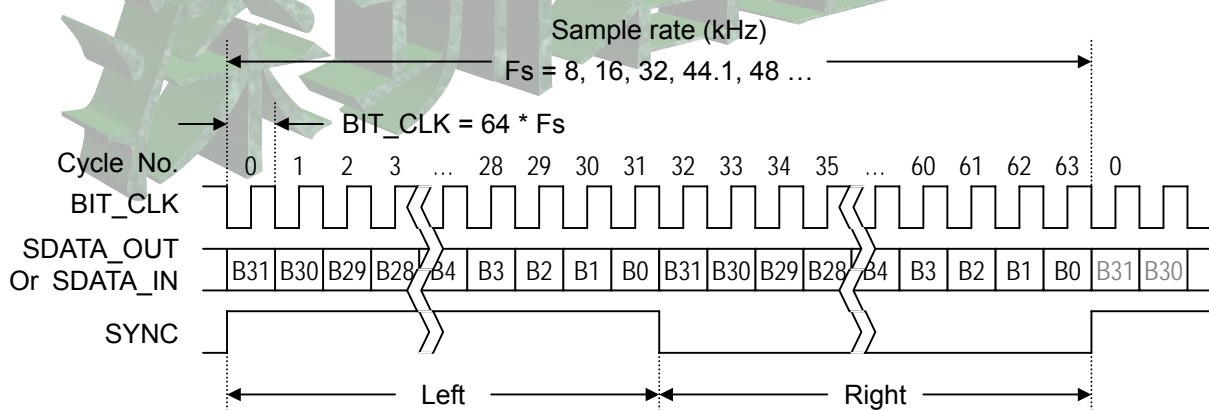


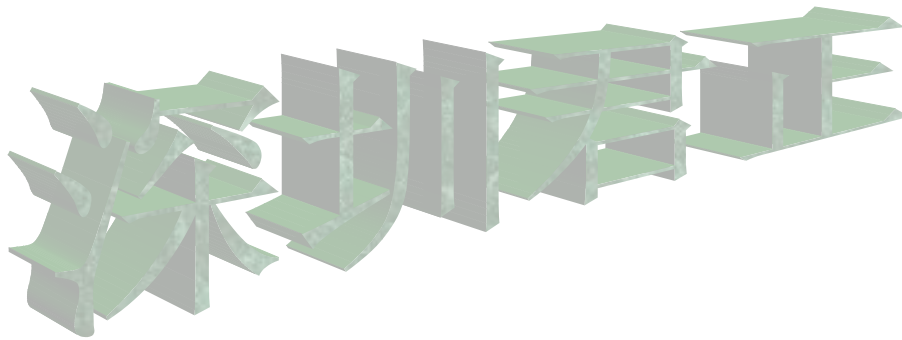
Figure 1-5 MSB-justified data format

Figure 1-4 and Figure 1-5 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.



### 1.4.3 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 1-2 described the how sample data bits are transferred.

**Table 1-2 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

## 1.5 Operation

The AIC can be accessed either by the processor using programmed I/O instructions or by the DMA controller. Jz47xx 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 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 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 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 1.6.1 for further details regarding AICFR.TFTH.
- The receive FIFO request is based on the receive trigger-threshold (AICFR.RFTH) setting. See 1.6.1 for further details regarding AICFR.RFTH.

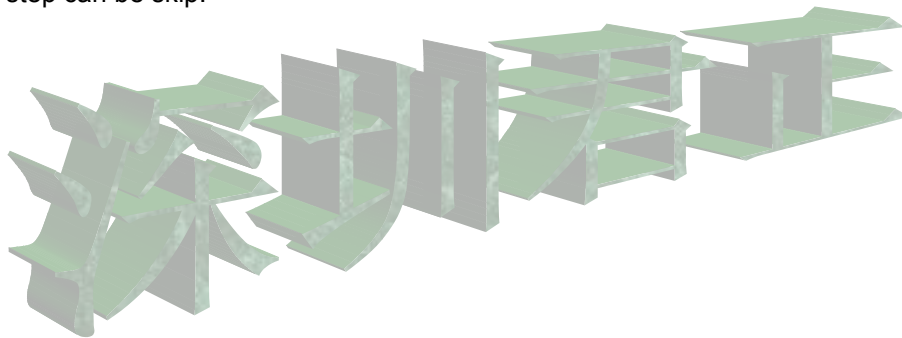
Before operation to AIC, you may need to set proper PIN function selection from GPIO using if the pin is shared with GPIO.

Please also reference to “AC '97 Component Specification Revision 2.3, 2002” when deal with AIC AC-link operations.

### 1.5.1 Initialization

At power-on or other hardware reset (WDT and etc), AIC is disabled. Software must initiate AIC and the external CODEC after power-on or reset. If errors found in data transferring, or in other places, software must initial AIC and optional, external CODEC. Here is the initial flow.

1. Select AC-link or I2S/MSB-Justified (AICFR.AUSEL). If the resettlement without involving link format and architecture changing, this step can be skip.
2. 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 1-5 (V1) or Table 1-6 (V2). A clock divider between PLL clock out and AIC also must be set.
3. Enable AIC by set AICFR.ENB
4. If it needs to reset AIC registers and flush FIFOs, set AICFR.RST. If it need only flush FIFOs, set AICCR.FLUSH. BIT\_CLK must exist hereafter.
5. In AC-link format, issue a warm or cold CODEC reset.
6. 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.
7. In I2S/MSB-Justified format, configure I2S/MSB-justified CODEC via I2C or L3 bus. If the resettlement without involving I2S/MSB-justified CODEC or ADC/DAC function changing, this step can be skip.



## 1.5.2 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).

## 1.5.3 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.

### 1.5.3.1 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.

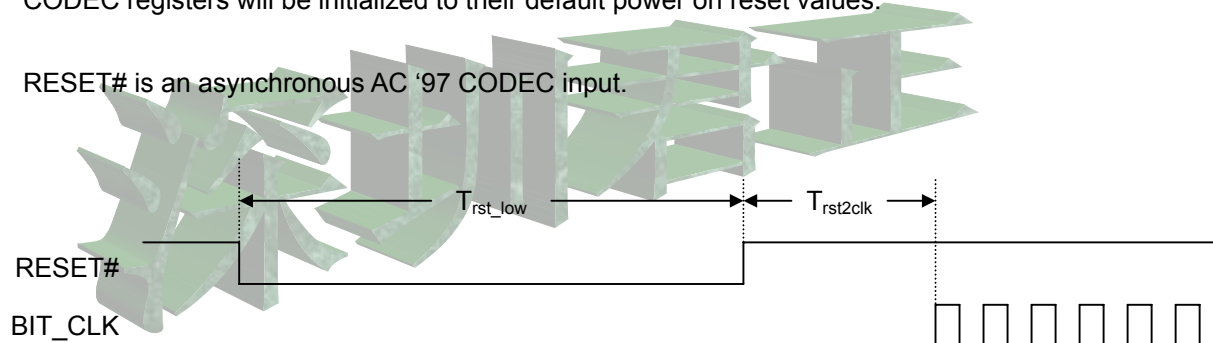


Figure 1-6 Cold AC '97 CODEC Reset Timing

Table 1-3 Cold AC '97 CODEC Reset Timing parameters

Parameter	Symbol	Min	Type	Max	Units
RESET# active low pulse width	$T_{rst\_low}$	1.0	-	-	$\mu s$
RESET# inactive to BIT_CLK startup delay	$T_{rst2clk}$	162.8	-	-	ns



### 1.5.3.2 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  $\mu$ s 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.

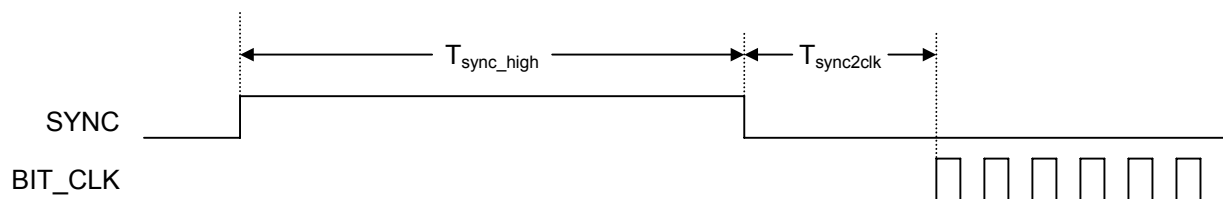


Figure 1-7 Warm AC '97 CODEC Reset Timing

Table 1-4 Warm AC '97 CODEC Reset Timing Parameters

Parameter	Symbol	Min	Type	Max	Units
SYNC active high pulse width	$T_{sync\_high}$	1.0	-	-	$\mu$ s
SYNC inactive to BIT_CLK startup delay	$T_{sync2clk}$	162.8	-	-	ns

### 1.5.4 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 or other control bus (for instance, I2C bus). The L3 bus operation is emulated by software controlling 3 GPIO pins. 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 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 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 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

### 1.5.5 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 the 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 set. 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 is send to CODEC even if AICCR.ERPL is 1.

Here is the audio replay flow:

1. Configure sample size by AICCR.OSS (V2) or ACCR2.OASS/I2SCR.WL (V1)
2. Configure sample rate by clock dividers (for I2S/MSB-Justified) or by CODEC registers (for AC-link in V2 only)
3. For AC-link, configure replay channels by ACCR1.XS
4. Some other configurations: mono to stereo (V2 only), endian switch (V2 only), signed/unsigned data transfer (V2 only), transmit FIFO configuration, and etc.
5. Set 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).
6. 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 set, and if AICCR.ETUR is 1, AIC issues an interrupt. Please reference to 1.5.7 for detail description on FIFO.
7. 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
8. Clear AICCR.ERPL.

### 1.5.6 Audio Record

Incoming audio sample data (from CODEC to AIC) is received from SDATA\_IN 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 set. 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 sample size by AICCR.ISS (V2) or ACCR2.IASS/I2SCR.WL (V1)
2. Configure sample rate by clock dividers (for I2S/MSB-Justified) or by CODEC registers (for AC-link in V2 only)
3. Some other configurations: signed/unsigned data transfer (V2 only), receive FIFO configuration, and etc.
4. Set 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
5. 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 set, and if AICCR.EROR is 1, AIC issues an interrupt. Please reference to 1.5.7 for detail description on FIFO. For AC-link, ACCR1.RS tells which channels are recorded.
6. Clear AICCR.EREC.
7. Take sample data form 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.

### 1.5.7 FIFOs and DMA 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 16 (V1) or 32 (V2) 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.

DMA operation is enabled by set AICCR.TDMS and AICCR.RDMS for transmit and receive respectively. One 24, 20, 18 bits audio sample 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. For 8-bits sample, 8-bits width DMA must be used.

When the processor accesses the FIFOs in small sample size, software must do the data pack/unpack, for instance, unpack 4 8-bits samples from one memory word and send them to transmit FIFO with 4 store instructions. Polling approach, which is in very low efficiency, can be used by the processor to detect when there are samples/rooms in the transmit/receive FIFOs. Interrupt approach also can be used through set AICCR.ETFS and AICCR.ERFS. DMA approach is in most efficient.

The AICFR.TFTH and AICFR.RFTH are used to set the FIFO level threshold, in which it determines the DMA request is issued and/or the AIC interrupt is issued if AICCR.ETFS and AICCR.ERFS are set. 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 during transmit under-run conditions. If AICCR.ETUR is set, this can trigger an interrupt. During transmit under-run conditions, zero 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 during receive over-run conditions. If AICCR.EROR is set, 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.

### 1.5.8 Serial Audio Clocks and Sampling Frequencies

For AC-link, the bit clock is input from chip external and is fixed to 12.288MHz. The sample frequency is fixed to 48kHz for V1. To replay other sample rate audio data, software has to do the rate transfer. Variable Sample Rate feature is supported in V2. If the CODEC supports this feature, sample rate other than 48kHz audio data can be replay directly. Double rate, 96kHz or even 88.2kHz (V2 only) audio is supported with proper CODEC.

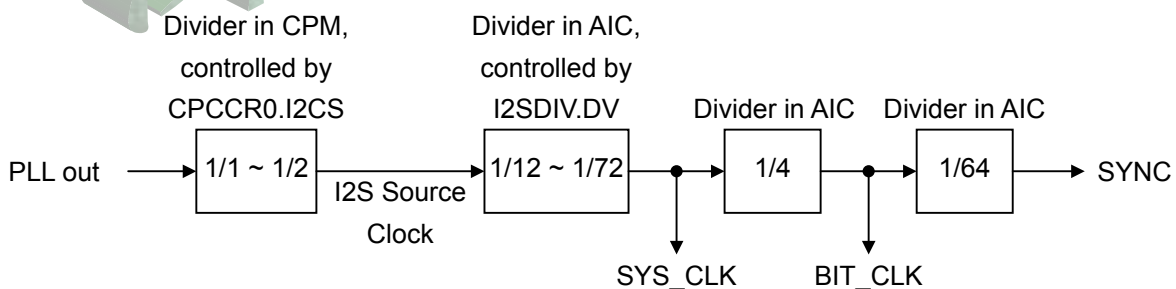
Following are for BIT\_CLK/SYS\_CLK configuration in I2S/MSB-Justified format.

The BIT\_CLK is the rate at which audio data bits enter or leave the AIC. SYS\_CLK is required by the CODEC to run delta sigma ADC operations.

BIT\_CLK can be supplied either by the CODEC or an internally PLL. If it is supplied internally, BIT\_CLK and SYS\_CLK are configured as output pins, and both are supplied to the CODEC. If BIT\_CLK is supplied by the CODEC, then it is configured as an input pin. In this case, the SYS\_CLK pin can be used as a GPIO pin.

BIT\_CLK varies by sampling frequencies (see Table 1-5 and Table 1-6). If BIT\_CLK is chosen as an output, the dividers divides the PLL clock to generate SYS\_CLK. SYS\_CLK is further divided by four or some other value (V2 only) to generate BIT\_CLK. The sampling frequency is the frequency of the SYNC signal, which is generated by dividing the BITCLK by 64,  $f_{\text{BIT\_CLK}} = 64 f_s$ .

In AIC V1, SYS\_CLK is fixed to  $256 f_s$  or  $4 f_{\text{BIT\_CLK}}$ . In Jz47xx, which includes an AIC V1, there is a clock divider for AIC/I2S source clock controlled by register CPCCR0.I2CS. This divider can divide PLL out by 1 or 2. The relation of SYS\_CLK, BIT\_CLK and SYNC is illustrated in Figure 1-8. It is suggested to set the I2S source clock to 147.456MHz, a common multiple of all SYS\_CLK frequencies. So the PLL out can be 147.456MHz or 294.912MHz, which are generated from 3.6864MHz input clock multiplied by 40 or 80.



**Figure 1-8 AIC V1 SYS\_CLK, BIT\_CLK and SYNC generation diagram**

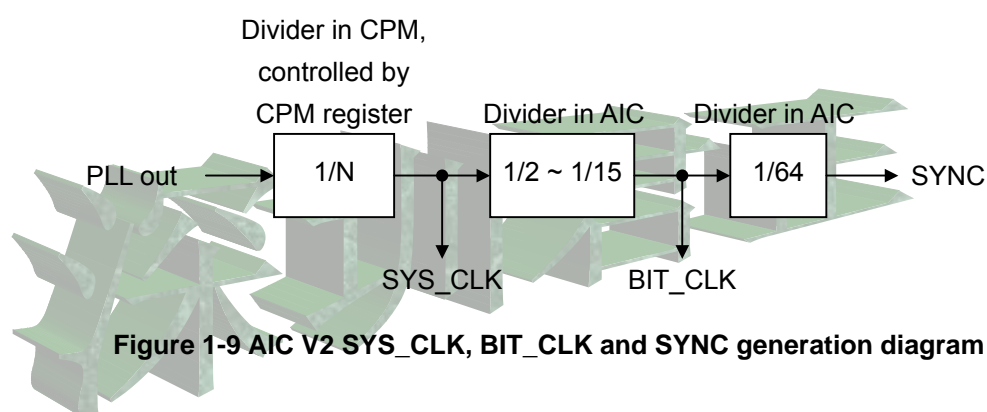
If SYS\_CLK and BIT\_CLK are chosen as output, the I2S clock divider, controlled by I2SDIV.DV, divides the 147.456MHz internal clock to generate the SYS\_CLK and BIT\_CLK as shown in Table 1-5. The sampling frequency is the frequency of the SYNC signal, which is generated by dividing the

BIT\_CLK by 64. See 1.6.11 for further details about the register.

**Table 1-5 AIC V1 Supported Sampling Frequencies**

I2SDIV.DV	SYS_CLK (MHz) = 147.456 MHz / I2SDIV	BIT_CLK (MHz) = SYS_CLK / 4	Sample Rate = BIT_CLK / 64
12	12.288	3.072	48.000 kHz (closest std = 48 kHz)
13	11.343	2.836	44.308 kHz (closest std = 44.1 kHz)
18	8.192	2.048	32.000 kHz (closest std = 32 kHz)
26	5.671	1.418	22.154 kHz (closest std = 22.05 kHz)
36	4.096	1.024	16.000 kHz (closest std = 16.00 kHz)
52	2.836	0.70892	11.077 kHz (closest std = 11.025 kHz)
72	2.048	0.512	8.000 kHz (closest std = 8.00 kHz)

The clock generation approach in AIC V2 is different from V1. In AIC V2, SYS\_CLK is generated in CPM module dividing the PLL output clock controlled by corresponding registers. Please reference to CPM spec for the details. The frequency of SYS\_CLK can be  $256 f_s$ ,  $384 f_s$ ,  $512 f_s$  and so on, depends on CODEC requirement. I2SDIV.DV is used to set the divider value to generate  $64 f_s$  BIT\_CLK from SYS\_CLK shown in Table 1-6.



**Table 1-6 AIC V2 BIT\_CLK divider setting**

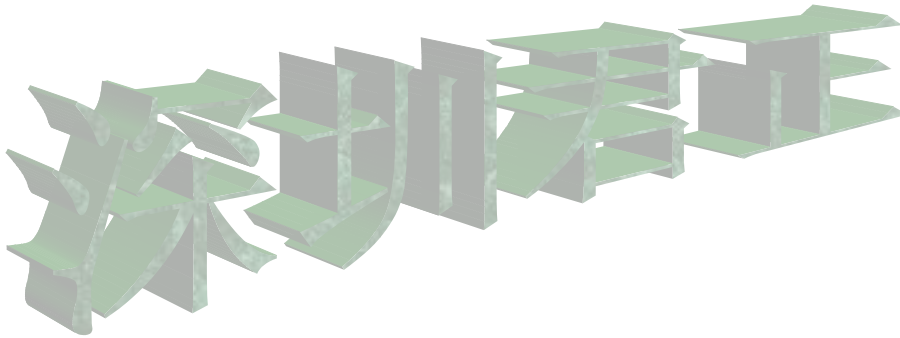
I2SDIV.DV	$f_{\text{SYS\_CLK}}$	$f_{\text{BIT\_CLK}}$	$f_{\text{SYS\_CLK}} / f_{\text{BIT\_CLK}}$
0x1	$128 f_s$	$64 f_s$	2
0x2	$196 f_s$	$64 f_s$	3
0x3	$256 f_s$	$64 f_s$	4
0x5	$384 f_s$	$64 f_s$	6
0x7	$512 f_s$	$64 f_s$	8
0xB	$768 f_s$	$64 f_s$	12

The software can stop the BIT\_CLK no matter whether it is generated internally or inputted from the external source. The system disables clock through the operation of I2SCR.STPBK. 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 set I2SCR.STPBK
4. Resume the BIT\_CLK by clear I2SCR.STPBK
5. Flush FIFO by set AICCR.FLUSH. This step and the following 2 steps are not required if you are sure all the FIFOs are empty.
6. Polling AICSR.RFL till it is 0
7. Polling AICSR.TFL till it is 0



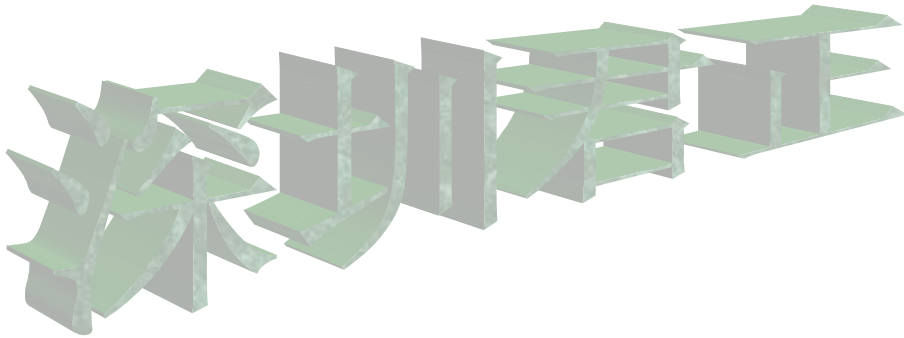


### 1.5.9 Interrupts

The following status bits, if enabled, interrupt the processor:

- Receive FIFO Service DMA Request (AICSR.RFS)
- Transmit FIFO Service DMA Request (AICSR.TFS)
- 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 register descriptions section.



## 1.6 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 1-7 AIC Registers Description**

Name	Description	RW	Reset value	Address	Size
AICFR	AIC Configuration Register	RW	0x00007700 <sup>1</sup> 0x00007800 <sup>2</sup>	0x10020000	32
AICCR	AIC Common Control Register	RW	0x00000000	0x10020004	32
ACCR1	AIC AC-link Control Register 1	RW	0x0???0000 <sup>1,3</sup> 0x00000000 <sup>2</sup>	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	0x???00???? <sup>1,4</sup> 0x00000008 <sup>2</sup>	0x10020014	32
ACSR	AIC AC-link Status Register	RW	0x000?0000 <sup>1,3</sup> 0x00000000 <sup>2</sup>	0x10020018	32
I2SSR	AIC I2S/MSB-justified Status Register	RW	0x0000000? <sup>1,3</sup> 0x00000000 <sup>2</sup>	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	0x000????? <sup>1,3</sup> 0x00000000 <sup>2</sup>	0x10020028	32
ACSDR	AIC AC97 CODEC Status Data Register	R	0x000????? <sup>1,3</sup> 0x00000000 <sup>2</sup>	0x1002002C	32
I2SDIV	AIC I2S/MSB-justified Clock Divider Register	RW	0x0000001A <sup>1</sup> 0x00000003 <sup>2</sup>	0x10020030	32
AICDR	AIC FIFO Data Port Register	RW	0x????????	0x10020034	32

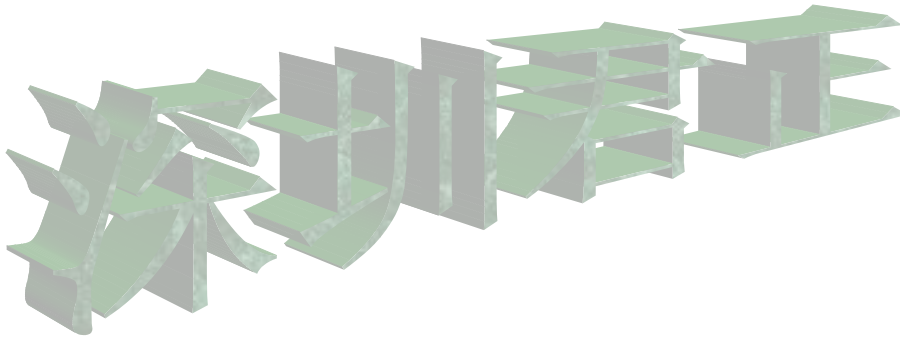
Note:

1. AIC V1
2. AIC V2
3. If BIT\_CLK is available in initialization, this value is 0x00000000; otherwise, undefined.
4. If BIT\_CLK is available in initialization, this value is 0x00000008; otherwise, undefined.

- AICFR is used to control FIFO threshold, AC-link or I2S/MSB-justified selection, AIC reset,

master/slave selection, and AIC enable.

- AICCR is used to control DMA mode, FIFO flush, interrupt enable, internal loop-back, play back and recording enable. In V2, it also controls sample size and signed/unsigned data transfer.
- ACCR1 is used to reflect/control valid incoming/outgoing slots of AC97.
- ACCR2 is used to control interrupt enable, output/input sample size, and alternative control of RESET#, SYNC and SDATA\_OUT pins in AC-link.
- I2SCR is used to control BIT\_CLK stop, audio sample size, I2S or MSB-justified selection in I2S/MSB-justified.
- AICSR is used to reflect FIFOs status
- ACSR is used to reflect the status of the connected external CODEC in AC-link.
- I2SSR is used to reflect AIC status in I2S/MSB-justified.
- ACCAR and ACCDR are used to hold address and data for AC-link CODEC register read/write.
- ACSAR and ACSDR are used to receive AC-link CODEC registers address and data
- I2SDIV is used to set clock divider for SYS\_CLK generating in I2S/MSB-justified format.
- AICDR is act as data input/output port to/from transmit/receive FIFO when write/read.



### 1.6.1 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	Reserved	Reserved	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	Note 1			0	0	0	0	0	0	0	0	0

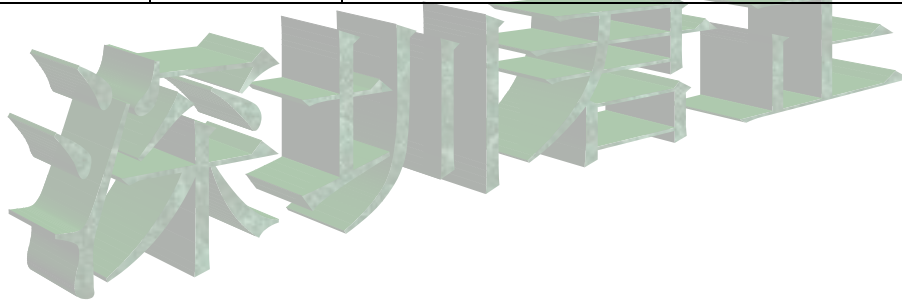
Note 1: TFTH initial to 0x7 in V1 and 0x8 in V2

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 for V1 and (RFTH + 1) * 2 for V2. 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 valie value 0 ~ 15.  This value represents a threshold value of TFTH + 1 for V1 and TFTH * 2 for V2. 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						
7:5	Reserved	Writes to these bits have no effect and always read as 0	R						
4	AUSEL	Audio Unit Select. Select between AC-link or I2S/MSB-justified. <table><tr><th>AUSEL</th><th>Selected</th></tr><tr><td>0</td><td>Select AC-link format</td></tr><tr><td>1</td><td>Select I2S/MSB-justified format</td></tr></table>	AUSEL	Selected	0	Select AC-link format	1	Select I2S/MSB-justified format	RW
AUSEL	Selected								
0	Select AC-link format								
1	Select I2S/MSB-justified format								
3	RST	Write to this bit reset AIC registers and FIFOs except I2SDIV register.	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. <table><tr><th>BCKD</th><th>BIT_CLK Direction</th></tr><tr><td>0</td><td>BIT_CLK is input from an external source.</td></tr></table>	BCKD	BIT_CLK Direction	0	BIT_CLK is input from an external source.	RW		
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.							
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. <table><tr><th>SYNCD</th><th>SYNC Direction</th></tr><tr><td>0</td><td>SYNC is input from an external source.</td></tr><tr><td>1</td><td>SYNC is generated internally and driven out to the CODEC.</td></tr></table>		SYNCD	SYNC Direction	0	SYNC is input from an external source.	1	SYNC is generated internally and driven out to the CODEC.	RW
SYNCD	SYNC Direction									
0	SYNC is input from an external source.									
1	SYNC is generated internally and driven out to the CODEC.									
0	ENB	Enable AIC function. This bit is used to enable or disable the AIC function. <table><tr><th>ENB</th><th>Description</th></tr><tr><td>0</td><td>Disable AIC Controller</td></tr><tr><td>1</td><td>Enable AIC Controller</td></tr></table>		ENB	Description	0	Disable AIC Controller	1	Enable AIC Controller	RW
ENB	Description									
0	Disable AIC Controller									
1	Enable AIC Controller									

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



## 1.6.2 AIC Common Control Register (AICCR)

AICCR contains bits to control DMA mode, FIFO flush, interrupt enable, internal loop-back, play back and recording enable. In V2, 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 <sup>1</sup>			ISS <sup>1</sup>			RDMS	TDMS	Reserved	Reserved	M2S <sup>1</sup>	ENDSW <sup>1</sup>	ASVTSU <sup>1</sup>	FLUSH	Reserved	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

Note 1: These fields are valid in V2 and are reserved in V1

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 (V2 only). 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. <table><tr><th>OSS</th><th>Sample Size</th></tr><tr><td>0x0</td><td>8 bit</td></tr><tr><td>0x1</td><td>16 bit</td></tr><tr><td>0x2</td><td>18 bit</td></tr><tr><td>0x3</td><td>20 bit</td></tr><tr><td>0x4</td><td>24 bit</td></tr><tr><td>0x5~0x7</td><td>Reserved</td></tr></table>	OSS	Sample Size	0x0	8 bit	0x1	16 bit	0x2	18 bit	0x3	20 bit	0x4	24 bit	0x5~0x7	Reserved	RW
OSS	Sample Size																
0x0	8 bit																
0x1	16 bit																
0x2	18 bit																
0x3	20 bit																
0x4	24 bit																
0x5~0x7	Reserved																
18:16	ISS	Input Sample Size (V2 only). 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. <table><tr><th>ISS</th><th>Sample Size</th></tr><tr><td>0x0</td><td>8 bit</td></tr><tr><td>0x1</td><td>16 bit</td></tr><tr><td>0x2</td><td>18 bit</td></tr><tr><td>0x3</td><td>20 bit</td></tr><tr><td>0x4</td><td>24 bit</td></tr><tr><td>0x5~0x7</td><td>Reserved</td></tr></table>	ISS	Sample Size	0x0	8 bit	0x1	16 bit	0x2	18 bit	0x3	20 bit	0x4	24 bit	0x5~0x7	Reserved	RW
ISS	Sample Size																
0x0	8 bit																
0x1	16 bit																
0x2	18 bit																
0x3	20 bit																
0x4	24 bit																
0x5~0x7	Reserved																
15	RDMS	Receive DMA enable. This bit is used to enable or disable the DMA during receiving audio data. <table><tr><th>RDMS</th><th>Receive DMA</th></tr><tr><td>0</td><td>Disabled</td></tr><tr><td>1</td><td>Enabled</td></tr></table>	RDMS	Receive DMA	0	Disabled	1	Enabled	RW								
RDMS	Receive DMA																
0	Disabled																
1	Enabled																
14	TDMS	Transmit DMA enable. This bit is used to enable or disable the DMA	RW														

		during transmit audio data.							
		<table><tr><th>TDMS</th><th>Transmit DMA</th></tr><tr><td>0</td><td>Disabled</td></tr><tr><td>1</td><td>Enabled</td></tr></table>	TDMS	Transmit DMA	0	Disabled	1	Enabled	
TDMS	Transmit DMA								
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 (V2 only). 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.	RW						
		<table><tr><th>M2S</th><th>Description</th></tr><tr><td>0</td><td>No mono to stereo expansion</td></tr><tr><td>1</td><td>Do mono to stereo expansion</td></tr></table>	M2S	Description	0	No mono to stereo expansion	1	Do mono to stereo expansion	
M2S	Description								
0	No mono to stereo expansion								
1	Do mono to stereo expansion								
10	ENDSW	Endian Switch (V2 only). This bit control endian change on outgoing 16-bits size audio sample by swapping high and low bytes in the sample data.	RW						
		<table><tr><th>ENDSW</th><th>Description</th></tr><tr><td>0</td><td>No change on outgoing sample data</td></tr><tr><td>1</td><td>Swap high and low byte for outgoing 16-bits size sample data</td></tr></table>	ENDSW	Description	0	No change on outgoing sample data	1	Swap high and low byte for outgoing 16-bits size sample data	
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 (V2 only). 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						
		<table><tr><th>ASVTSU</th><th>Description</th></tr><tr><td>0</td><td>No audio sample value signed ↔ unsigned transfer</td></tr><tr><td>1</td><td>Do audio sample value signed ↔ unsigned transfer</td></tr></table>	ASVTSU	Description	0	No audio sample value signed ↔ unsigned transfer	1	Do audio sample value signed ↔ unsigned transfer	
ASVTSU	Description								
0	No audio sample value signed ↔ unsigned transfer								
1	Do audio sample value signed ↔ unsigned transfer								
8	FLUSH	FIFO Flush. Write to this bit flush transmit/receive FIFOs to empty.	W						
7	Reserved	Writes to these bits have no effect and always read as 0	R						
6	EROR	Enable ROR Interrupt. This bit is used to control the ROR interrupt enable or disable.	RW						
		<table><tr><th>EROR</th><th>ROR Interrupt</th></tr><tr><td>0</td><td>Disabled</td></tr><tr><td>1</td><td>Enabled</td></tr></table>	EROR	ROR Interrupt	0	Disabled	1	Enabled	
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						
		<table><tr><th>ETUR</th><th>TUR Interrupt</th></tr><tr><td>0</td><td>Disabled</td></tr><tr><td>1</td><td>Enabled</td></tr></table>	ETUR	TUR Interrupt	0	Disabled	1	Enabled	
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						
		<table><tr><th>ERFS</th><th>RFS Interrupt</th></tr><tr><td>0</td><td>Disabled</td></tr></table>	ERFS	RFS Interrupt	0	Disabled			
ERFS	RFS Interrupt								
0	Disabled								

		0	Disabled		
		1	Enabled		
3	ETFS	Enable TFS Interrupt. This bit is used to control the TFS interrupt enable or disable.			RW
		<b>ETFS</b>	<b>TFS Interrupt</b>		
		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
		<b>ENLBF</b>	<b>Description</b>		
		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
		<b>ERPL</b>	<b>Description</b>		
		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
		<b>EREC</b>	<b>Description</b>		
		0	AIC Recording Function is Disabled		
		1	AIC Recording Function is Enabled		



### 1.6.3 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.

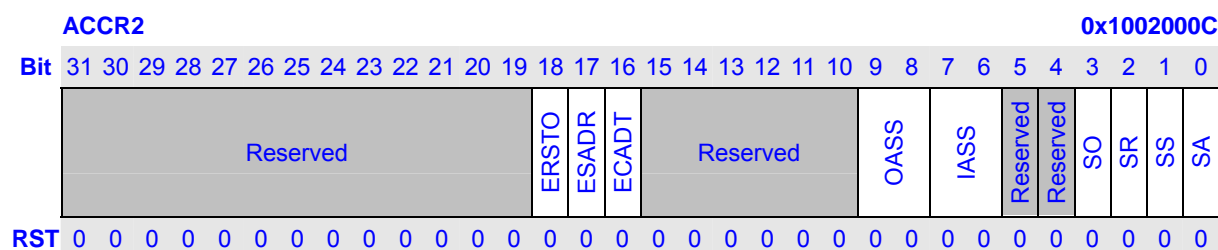
ACCR1																0x10020008																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved						RS						Reserved						XS													
RST	0	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note: 1. In V1, if BIT\_CLK is available in initialization, the reset value is 0; otherwise, undefined. In V2, the reset value is 0.

Bits	Name	Description	RW						
31:26	Reserved	Writes to these bits have no effect and always read as 0	R						
25:16	RS	<p>Receive Valid Slots. These read only bits are taken from the valid bits in the incoming AC'97 tag (slot 0 of SDATA_IN) and indicate which incoming slots have valid PCM data. Slot 3 is mapped to bit 16 or RS[0], slot 4 to bit 17 or RS[1] and so on. If the corresponding bit is set it indicates that valid PCM data is in the respective slot.</p> <table><tr><th>RS[n] Value</th><th>Description</th></tr><tr><td>0</td><td>Slot n+3 is invalid.</td></tr><tr><td>1</td><td>Slot n+3 has valid PCM data.</td></tr></table>	RS[n] Value	Description	0	Slot n+3 is invalid.	1	Slot n+3 has valid PCM data.	R
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	<p>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.</p> <table><tr><th>XS[n] Value</th><th>Description</th></tr><tr><td>0</td><td>Slot n+3 is invalid.</td></tr><tr><td>1</td><td>Slot n+3 has valid PCM data.</td></tr></table>	XS[n] Value	Description	0	Slot n+3 is invalid.	1	Slot n+3 has valid PCM data.	RW
XS[n] Value	Description								
0	Slot n+3 is invalid.								
1	Slot n+3 has valid PCM data.								

### 1.6.4 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.

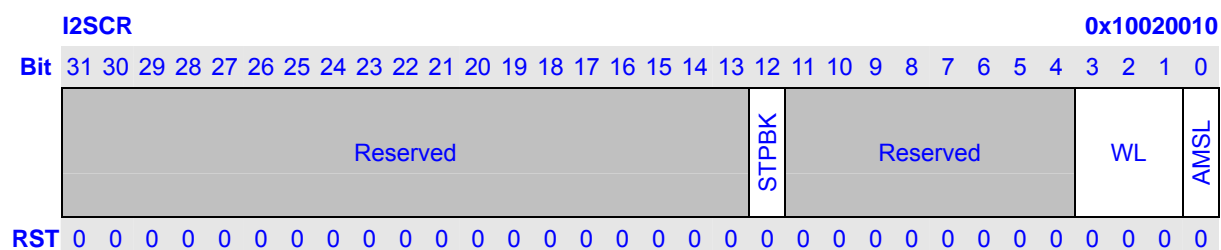


Bits	Name	Description	RW										
31:19	Reserved	Writes to these bits have no effect and always read as 0	R										
18	ERSTO	Enable RSTO Interrupt. This bit is used to control the RSTO interrupt enable or disable. <table><tr><th>ERSTO</th><th>RSTO Interrupt</th></tr><tr><td>0</td><td>Disabled</td></tr><tr><td>1</td><td>Enabled</td></tr></table>	ERSTO	RSTO Interrupt	0	Disabled	1	Enabled	RW				
ERSTO	RSTO Interrupt												
0	Disabled												
1	Enabled												
17	ESADR	Enable SADR Interrupt. This bit is used to control the SADR interrupt enable or disable. <table><tr><th>ESADR</th><th>SADR Interrupt</th></tr><tr><td>0</td><td>Disabled</td></tr><tr><td>1</td><td>Enabled</td></tr></table>	ESADR	SADR Interrupt	0	Disabled	1	Enabled	RW				
ESADR	SADR Interrupt												
0	Disabled												
1	Enabled												
16	ECADT	Enable CADT Interrupt. This bit is used to control the CADT interrupt enable or disable. <table><tr><th>ECADT</th><th>CADT Interrupt</th></tr><tr><td>0</td><td>Disabled</td></tr><tr><td>1</td><td>Enabled</td></tr></table>	ECADT	CADT Interrupt	0	Disabled	1	Enabled	RW				
ECADT	CADT Interrupt												
0	Disabled												
1	Enabled												
15:10	Reserved	Writes to these bits have no effect and always read as 0	R										
9:8	OASS	Output Sample Size for AC-link. These bits reflect output sample data size from memory or register. The data sizes supported are: 8, 16, 18 and 20 bits. The sample data is LSB-justified in memory/register. In case of V2, it's better to use AICCR.OSS instead this field to set output sample size. <table><tr><th>OASS</th><th>Sample Size</th></tr><tr><td>0x0</td><td>20 bit</td></tr><tr><td>0x1</td><td>18 bit</td></tr><tr><td>0x2</td><td>16 bit</td></tr><tr><td>0x3</td><td>8 bit</td></tr></table>	OASS	Sample Size	0x0	20 bit	0x1	18 bit	0x2	16 bit	0x3	8 bit	RW
OASS	Sample Size												
0x0	20 bit												
0x1	18 bit												
0x2	16 bit												
0x3	8 bit												
7:6	IASS	Input Sample Size for AC-link. These bits reflect input sample data size to memory or register. The data sizes supported are: 8, 16, 18 and 20 bits.	RW										

		<div>The sample data is LSB-justified in memory/register. In case of V2, it's better to use AICCR.ISS instead this field to set input sample size.</div> <table><tr><th>IASS</th><th>Sample Size</th></tr><tr><td>0x0</td><td>20 bit</td></tr><tr><td>0x1</td><td>18 bit</td></tr><tr><td>0x2</td><td>16 bit</td></tr><tr><td>0x3</td><td>8 bit</td></tr></table>	IASS	Sample Size	0x0	20 bit	0x1	18 bit	0x2	16 bit	0x3	8 bit								
IASS	Sample Size																			
0x0	20 bit																			
0x1	18 bit																			
0x2	16 bit																			
0x3	8 bit																			
5:4	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	<div>RESET# pin level. When AC-link is selected, this bit is used to drive the RESET# pin.</div> <table><tr><th>SR</th><th>RESET# Pin Voltage Level</th></tr><tr><td>0</td><td>High</td></tr><tr><td>1</td><td>Low</td></tr></table>	SR	RESET# Pin Voltage Level	0	High	1	Low	RW											
SR	RESET# Pin Voltage Level																			
0	High																			
1	Low																			
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	<div>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 controlled by the SS and SDATA_OUT is controlled by the SO. The true table of SYNC is described in following.</div> <table><tr><th>SA</th><th>SS</th><th>Description</th></tr><tr><td rowspan="4">0</td><td rowspan="2">0</td><td>When read, indicated SYNC is 0</td></tr><tr><td>When write, not effect</td></tr><tr><td rowspan="2">1</td><td>When read, indicated SYNC is 1</td></tr><tr><td>When write, not effect</td></tr><tr><td rowspan="4">1</td><td rowspan="2">0</td><td>When read, indicated SYNC is 0</td></tr><tr><td>When write, SYNC is driven to 0</td></tr><tr><td rowspan="2">1</td><td>When read, indicated SYNC is 1</td></tr><tr><td>When write, SYNC is driven to 1</td></tr></table>	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	RW
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																		

### 1.6.5 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.



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 is ignored. <table><tr><th>STPBK</th><th>Description</th></tr><tr><td>0</td><td>BIT_CLK is not stopped</td></tr><tr><td>1</td><td>BIT_CLK is stopped</td></tr></table>	STPBK	Description	0	BIT_CLK is not stopped	1	BIT_CLK is stopped	RW								
STPBK	Description																
0	BIT_CLK is not stopped																
1	BIT_CLK is stopped																
11:4	Reserved	Writes to these bits have no effect and always read as 0	R														
3:1	WL	Input/Output Sample Size for I2S/MSB-Justified. These bits reflect input/output sample data size to/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. In case of V2, it's better to use AICCR.ISS and AICCR.OSS instead this field to set input/output sample size. <table><tr><th>WL</th><th>Sample Size</th></tr><tr><td>0x0</td><td>24 bit</td></tr><tr><td>0x1</td><td>20 bit</td></tr><tr><td>0x2</td><td>18 bit</td></tr><tr><td>0x3</td><td>16 bit</td></tr><tr><td>0x4</td><td>8 bit</td></tr><tr><td>0x5~0x7</td><td>Reserved</td></tr></table>	WL	Sample Size	0x0	24 bit	0x1	20 bit	0x2	18 bit	0x3	16 bit	0x4	8 bit	0x5~0x7	Reserved	RW
WL	Sample Size																
0x0	24 bit																
0x1	20 bit																
0x2	18 bit																
0x3	16 bit																
0x4	8 bit																
0x5~0x7	Reserved																
0	AMSL	Specify Alternate Mode (I2S or MSB-Justified) Operation. <table><tr><th>AMSL</th><th>Description</th></tr><tr><td>0</td><td>Select I2S Operation Mode</td></tr><tr><td>1</td><td>Select MSB-Justified Operation Mode</td></tr></table>	AMSL	Description	0	Select I2S Operation Mode	1	Select MSB-Justified Operation Mode	RW								
AMSL	Description																
0	Select I2S Operation Mode																
1	Select MSB-Justified Operation Mode																

### 1.6.6 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 (AIC V1)																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

- Notes: 1. If BIT\_CLK is available in initialization, the reset value is 0; otherwise, undefined.  
 2. If BIT\_CLK is available in initialization, the reset value is 1; otherwise, undefined.

Bits	Name	Description (AIC V1)	RW						
31:29	Reserved	Writes to these bits have no effect and always read as 0	R						
28:24	RFL	Receive FIFO Level. The bits indicate the amount of valid PCM data in Receive FIFO. <table><tr><th>RFL Value</th><th>Description</th></tr><tr><td>0x00 ~ 0x10</td><td>RFL valid PCM data in receive FIFO</td></tr><tr><td>0x11 ~ 0x1F</td><td>Reserved</td></tr></table>	RFL Value	Description	0x00 ~ 0x10	RFL valid PCM data in receive FIFO	0x11 ~ 0x1F	Reserved	R
RFL Value	Description								
0x00 ~ 0x10	RFL valid PCM data in receive FIFO								
0x11 ~ 0x1F	Reserved								
23:13	Reserved	Writes to these bits have no effect and always read as 0	R						
12:8	TFL	Transmit FIFO Level. The bits indicate the amount of valid PCM data in Transmit FIFO. <table><tr><th>TFL Value</th><th>Description</th></tr><tr><td>0x00 ~ 0x10</td><td>TFL valid PCM data in transmit FIFO</td></tr><tr><td>0x11 ~ 0x1F</td><td>Reserved</td></tr></table>	TFL Value	Description	0x00 ~ 0x10	TFL valid PCM data in transmit FIFO	0x11 ~ 0x1F	Reserved	R
TFL Value	Description								
0x00 ~ 0x10	TFL valid PCM data in transmit FIFO								
0x11 ~ 0x1F	Reserved								

AICSR (AIC V2)														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 (AIC V2)	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. <table><tr><th>RFL Value</th><th>Description</th></tr><tr><td>0x00 ~ 0x20</td><td>RFL valid PCM data in receive FIFO</td></tr></table>	RFL Value	Description	0x00 ~ 0x20	RFL valid PCM data in receive FIFO	R
RFL Value	Description						
0x00 ~ 0x20	RFL valid PCM data in receive FIFO						

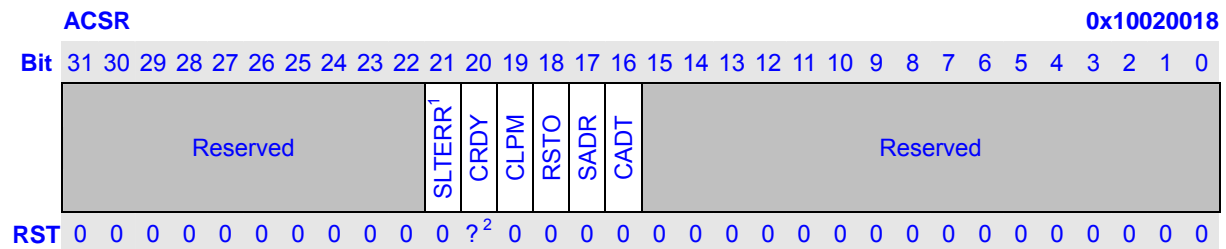
		0x21 ~ 0x3F	Reserved	
23:14	Reserved	Writes to these bits have no effect and always read as 0		R
13:8	TFL	Transmit FIFO Level. The bits indicate the amount of valid PCM data in Transmit FIFO.		R
		<b>TFL Value</b>	<b>Description</b>	
		0x00 ~ 0x20	TFL valid PCM data in transmit FIFO	
		0x21 ~ 0x3F	Reserved	

Bits	Name	Description	RW								
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. <table><tr><th>ROR</th><th>Description</th></tr><tr><td rowspan="2">0</td><td>When read, indicates over-run has not been found</td></tr><tr><td>When write, clear itself</td></tr><tr><td rowspan="2">1</td><td>When read, indicates data has even been written to full receive FIFO</td></tr><tr><td>When write, not effects</td></tr></table>	ROR	Description	0	When read, indicates over-run has not been found	When write, clear itself	1	When read, indicates data has even been written to full receive FIFO	When write, not effects	RW
ROR	Description										
0	When read, indicates over-run has not been found										
	When write, clear itself										
1	When read, indicates data has even been written to full receive FIFO										
	When write, not effects										
5	TUR	Transmit FIFO Under Run. This bit indicates that transmit FIFO has or has not experienced an under-run. <table><tr><th>TUR</th><th>Description</th></tr><tr><td rowspan="2">0</td><td>When read, indicates under-run has not been found</td></tr><tr><td>When write, clear itself</td></tr><tr><td rowspan="2">1</td><td>When read, indicates data has even been read from empty transmit FIFO</td></tr><tr><td>When write, not effects</td></tr></table>	TUR	Description	0	When read, indicates under-run has not been found	When write, clear itself	1	When read, indicates data has even been read from empty transmit FIFO	When write, not effects	RW
TUR	Description										
0	When read, indicates under-run has not been found										
	When write, clear itself										
1	When read, indicates data has even been read from empty transmit FIFO										
	When write, not effects										
4	RFS	Receive FIFO Service Request. This bit indicates that receive FIFO level is or not below RFL 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>	RFS	Description	0	Receive FIFO level below RFL threshold	1	Receive FIFO level at or above RFL threshold	R		
RFS	Description										
0	Receive FIFO level below RFL threshold										
1	Receive FIFO level at or above RFL threshold										
3	TFS	Transmit FIFO Service Request. This bit indicates that transmit FIFO level exceeds TFL 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>	TFS	Description	0	Transmit FIFO level exceeds TFL threshold	1	Transmit FIFO level at or below TFL threshold	R		
TFS	Description										
0	Transmit FIFO level exceeds TFL threshold										
1	Transmit FIFO level at or below TFL threshold										
2:0	Reserved	Writes to these bits have no effect and always read as 0	R								

### 1.6.7 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.

<For word>



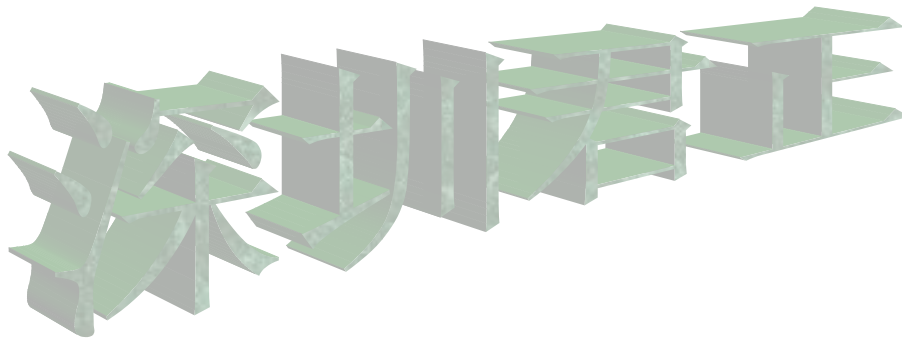
Notes: 1. This field is valid in V2 and is reserved in V1.

2. In V1, if BIT\_CLK is available in initialization, the reset value is 0; otherwise, undefined.

In V2, the reset value is 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 (V2 only). 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. <table><tr><th>CRDY</th><th>Description</th></tr><tr><td>0</td><td>CODEC is not ready</td></tr><tr><td>1</td><td>CODEC is ready</td></tr></table>	CRDY	Description	0	CODEC is not ready	1	CODEC is ready	R
CRDY	Description								
0	CODEC is not ready								
1	CODEC is ready								
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. <table><tr><th>CLPM</th><th>Description</th></tr><tr><td>0</td><td>BIT_CLK is active</td></tr><tr><td>1</td><td>CODEC is switched to low power mode</td></tr></table>	CLPM	Description	0	BIT_CLK is active	1	CODEC is switched to low power mode	R
CLPM	Description								
0	BIT_CLK is active								
1	CODEC is switched to low power mode								
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. <table><tr><th>RSTO</th><th>Description</th></tr><tr><td>0</td><td>When read, indicates time out has not occurred</td></tr></table>	RSTO	Description	0	When read, indicates time out has not occurred	RW		
RSTO	Description								
0	When read, indicates time out has not occurred								

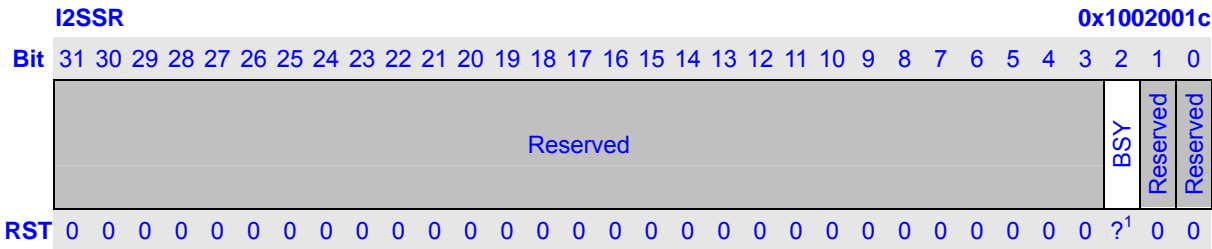
		<table><tr><td>1</td><td>When read, indicates read status time out found</td></tr></table> <p>Write 0 clear this bit and write 1 is ignored. When RSTO is 1, it may trigger an interrupt depends on the interrupt enable setting.</p>	1	When read, indicates read status time out found					
1	When read, indicates read status time out found								
17	SADR	<p>External CODEC Registers Status Address and Data Received. This bit indicates that address and data of an external AC '97 CODEC register has or has not been received.</p> <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> <p>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.</p>	SADR	Description	0	When read, indicates no register address/data received.	1	When read, indicates address/data received.	RW
SADR	Description								
0	When read, indicates no register address/data received.								
1	When read, indicates address/data received.								
16	CADT	<p>Command Address and Data Transmitted. This bit indicates that a CODEC register reading/writing command transmission has completed or not.</p> <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> <p>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.</p>	CADT	Description	0	When read, indicates the command has not done.	1	When read, indicates the command has done.	RW
CADT	Description								
0	When read, indicates the command has not done.								
1	When read, indicates the command has done.								
15:0	Reserved	Writes to these bits have no effect and always read as 0	R						





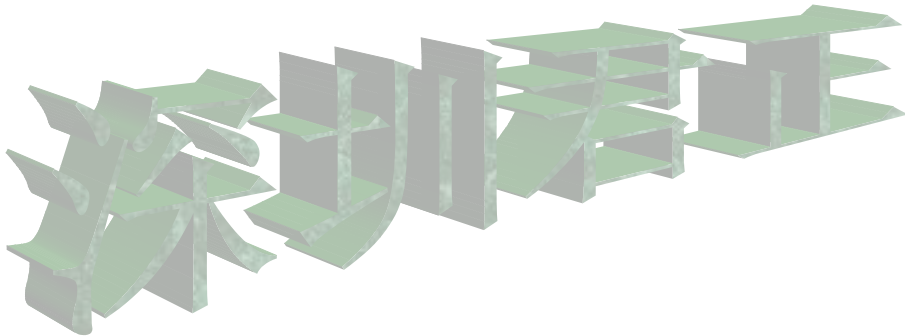
1.6.8 AIC I2S/MSB-justified Status Register (I2SSR)

I2SSR is used to reflect AIC status in I2S/MSB-justified. It is a read-only register.



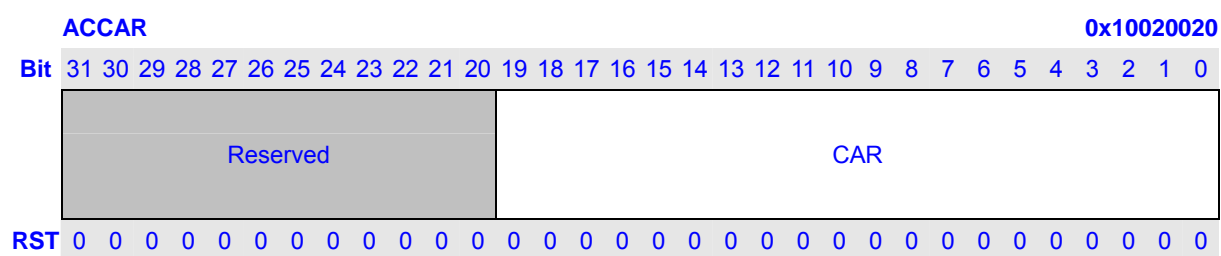
Note 1. In V1, if BIT\_CLK is available in initialization, the reset value is 0; otherwise, undefined. In V2, the reset value is 0.

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.	R						
		<table><tr><th>BSY</th><th>Description</th></tr><tr><td>0</td><td>AIC controller is idle or disabled</td></tr><tr><td>1</td><td>AIC controller currently is transmitting or receiving a frame</td></tr></table>		BSY	Description	0	AIC controller is idle or disabled	1	AIC controller currently is transmitting or receiving a frame
		BSY		Description					
		0		AIC controller is idle or disabled					
1	AIC controller currently is transmitting or receiving a frame								
1:0	Reserved	Writes to these bits have no effect and always read as 0	R						

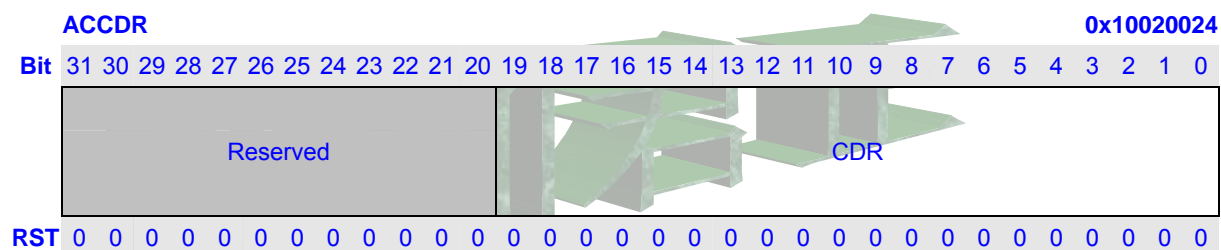


### 1.6.9 AIC AC97 CODEC Command Address Register (ACCAR) & Data Register (ACCDR)

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 1.5.4 for software flow. These registers are valid only in AC-link. It is ignored in I2S/MSB-justified format.



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.	RW



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.	RW

### 1.6.10 AIC AC97 CODEC Status Address Register (ACSAR) & Data Register (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 1.5.4 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
RST	0	0	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Note 1. In V1, if BIT\_CLK is available in initialization, the reset value is 0; otherwise, undefined. In V2, the reset value is 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

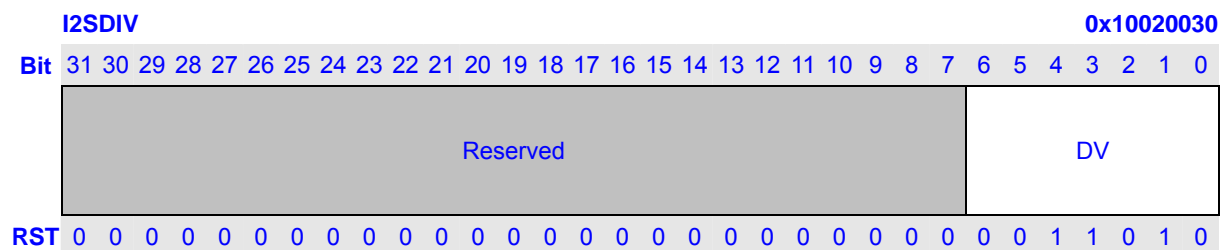
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RST	0	0	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Note 1. In V1, if BIT\_CLK is available in initialization, the reset value is 0; otherwise, undefined. In V2, the reset value is 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

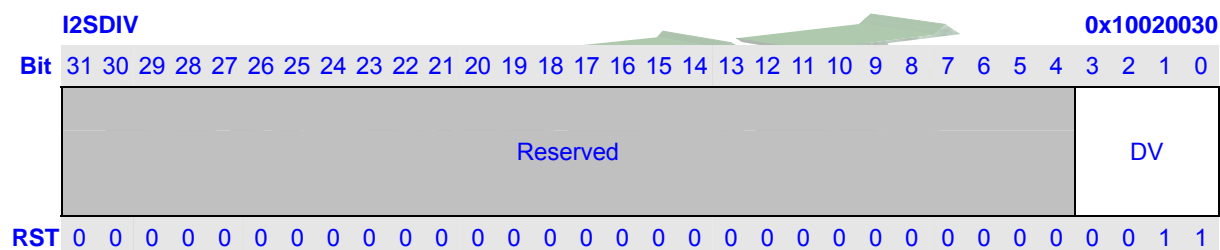
### 1.6.11 AIC I2S/MSB-justified Clock Divider Register (I2SDIV)

In V1, I2SDIV is used to set clock divider for SYS\_CLK generating in I2S/MSB-justified format.



Bits	Name	Description	RW
31:7	Reserved	Writes to these bits have no effect and always read as 0	R
6:0	DV	Audio SYS_CLK clock divider value. It controls the frequency of SYS_CLK and BIT_CLK generated inside chip. SYS_CLK is generated by dividing I2S source clock from CPM by this value. Please reference to 1.5.8 "Serial Audio Clocks and Sampling Frequencies" for further description.	RW

In V2, I2SDIV is used to set clock divider to generated BIT\_CLK from SYS\_CLK in I2S/MSB-justified format.

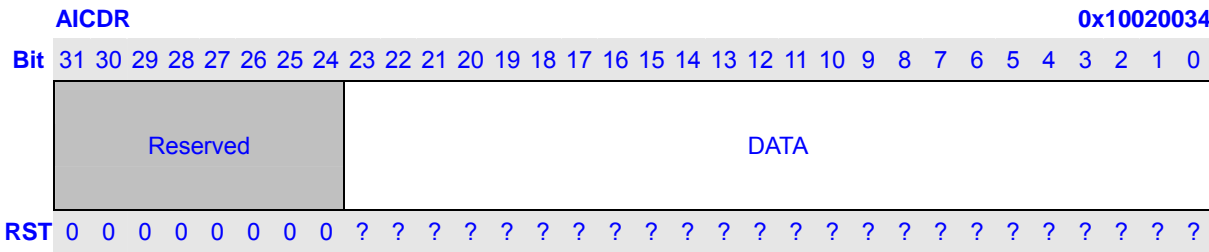


Bits	Name	Description	RW
31:4	Reserved	Writes to these bits have no effect and always read as 0	R
3:0	DV	Audio BIT_CLK clock divider value minus 1. I2SDIV.DV is used to control the generating of BIT_CLK from dividing SYS_CLK. The dividing value should be even and I2SDIV.DV should be set to the dividing 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 1.5.8 "Serial Audio Clocks and Sampling Frequencies" for further description.	RW

1.6.12 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, left channel sample is prior to 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.



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

# 1 MultiMediaCard/Secure Digital Controller

## 1.1 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 MMC communication is based on an advanced 7 pin serial bus designed to operate in a low voltage range, at medium speed (20 Mbps).

The Secure Digital (SD) card is an evolution of MMC, with an additional 2 pins and the same form factors. 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.

Features of the MultimediaCard/Secure Digital Controller include the following:

- Fully compatible with the *MMC System Specification version 3.3*
- Fully compatible with the *SD Memory Card Specification 1.01* and *SD I/O Specification 1.0* with 1 command channel and 4 data channels
- 20-80 Mbps maximum data rate
- Built-in programmable frequency divider for MMC/SD bus
- Maskable hardware interrupt for SDIO interrupt, internal status and FIFO status
- 16-entry x 32-bit built-in data FIFO
- Password protection of cards
- Multi-SD function support including multiple I/O and combined I/O and memory
- Up to 7 I/O functions plus one memory supported on single SDIO card
- IRQ supported enable card to interrupt MMC/SD controller
- Single or multi block access to the card including erase operation
- Stream access to the card
- Supports SDIO read wait, interrupt detection during 1-bit or 4-bit access
- The maximum block length is 2048 bytes

1.2 Block Diagram

MMC/SD Controller Block Diagram

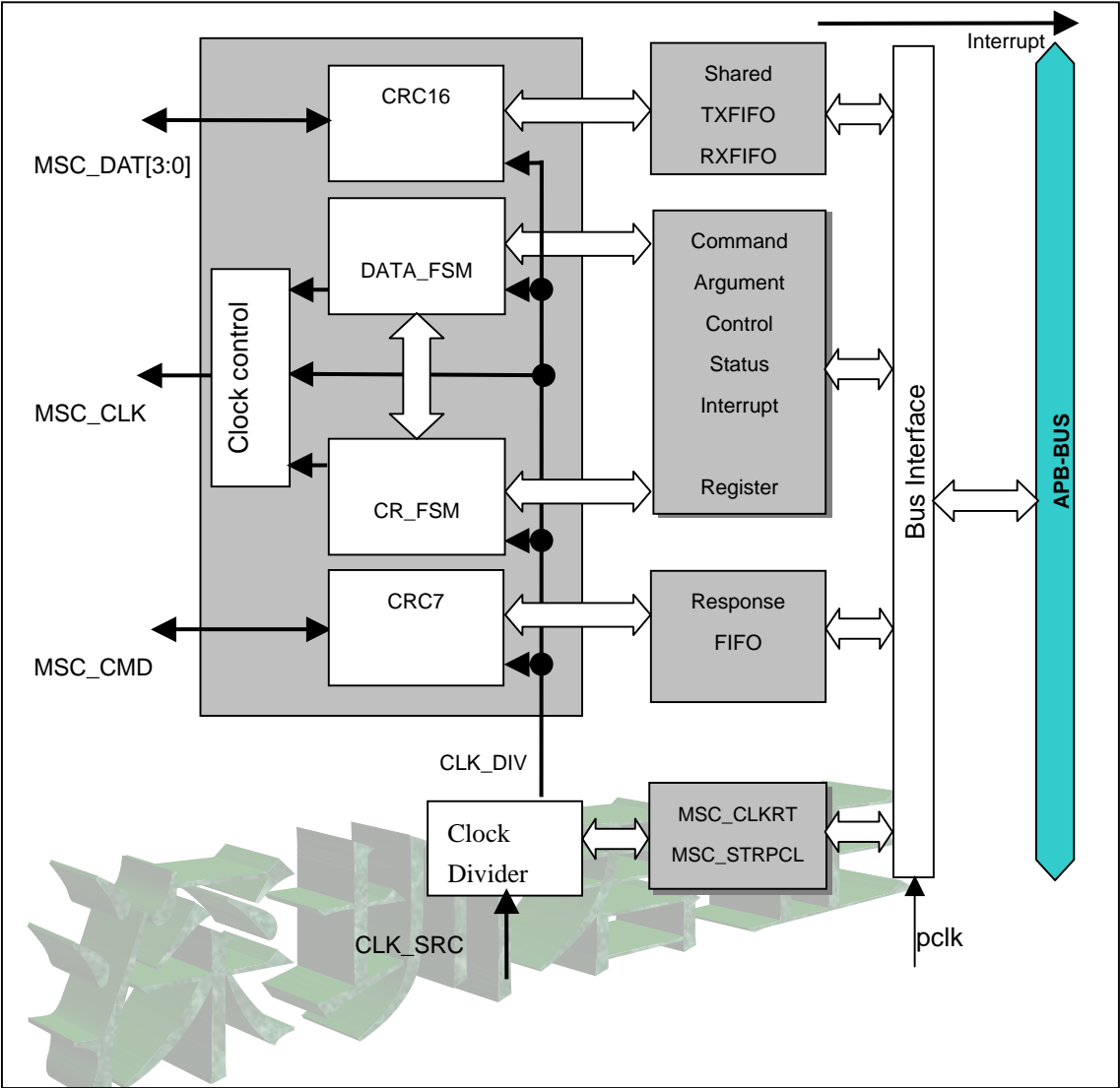
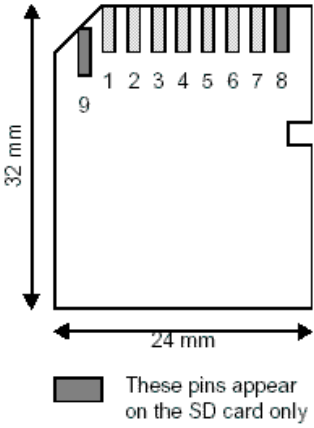


Figure 1-1 MMC/SD Controller Block Diagram

### 1.3 MMC/SD Controller Signal I/O Description

The MMC and SD cards are 7- or 9- pin cards that operate as external memory storage. The pin assignment and form factor are shown in Table 1-1.

**Table 1-1 MMC/SD Controller Signal Description**

Form factor and pinout	Pin number	MMC card	SD card	
			1-bit mode	4-bit mode
 <p>32 mm</p> <p>24 mm</p> <p>These pins appear on the SD card only</p>	1	Reserved	Not Used	Data Line [Bit 3]
	2	Command/Response (CMD)		
	3	Supply Voltage Ground (Vss1)		
	4	Supply Voltage (Vdd)		
	5	Clock (CLK)		
	6	Supply Voltage Ground (Vss2)		
	7	Data Line [Bit 0]		
	8		Interrupt (IRQ)	Data Line [Bit 1] or Interrupt (IRQ)
	9		ReadWait(RW)	Data Line [Bit 2] or ReadWait (RW)

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 1-2 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

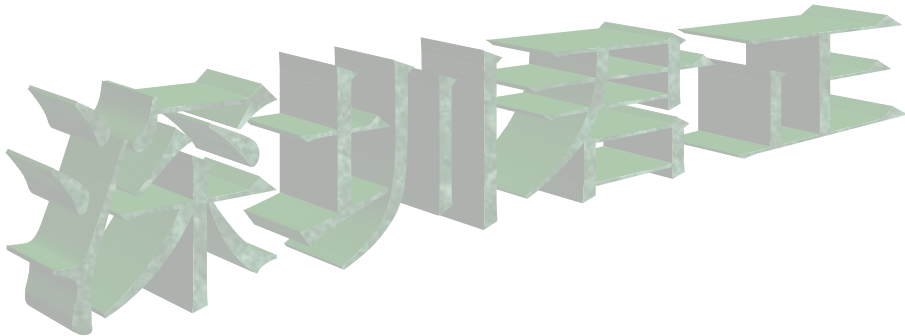
- Response:** a response is a token which is sent from an addressed card, or (synchronously) from all connected cards, to MSC as an answer to a previously received command. A response is transferred serially on the CMD line. Response tokens have varies coding schemes depending on their content.



- **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 1-3 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





		1 – Exit from multiple block read/write.	
6	EXIT_TRANSFER	<b>Only used for SDIO suspend/resume and MMC stream read.</b> 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. 1 – Start ReadWait.	W
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. 0 – 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

#### 1.4.2 MSC Status Register (MSC\_STAT)

##### MSC\_STAT

0x10021004

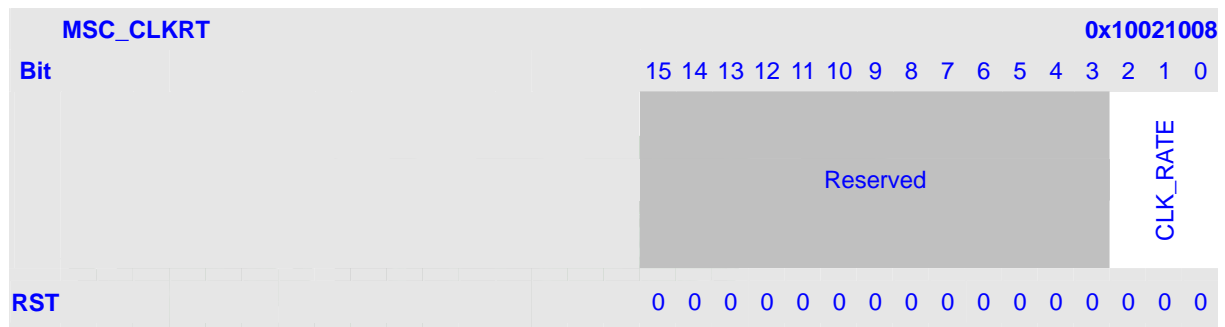
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved																IS_RESETTING	SDIO_INT_ACTIVE	PRG_DONE	DATA_TRAN_DONE	END_CMD_RES	DATA_FIFO_AFULL	IS_READWAIT	CLK_EN	DATA_FIFO_FULL	DATA_FIFO_EMPTY	CRC_RES_ERR	CRC_READ_ERROR	CRC_WRITE_ERROR	TIME_OUT_RES	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
31:16	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 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.	R
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 $\geq 15$ ). 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 enter ReadWait. 1 – Card has enter 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.	R

		1– CRC error occurred on the response.	
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. 01 – Card observed erroneous transmission of data. 10 – No CRC status is sent back. 11 – Reserved	R
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

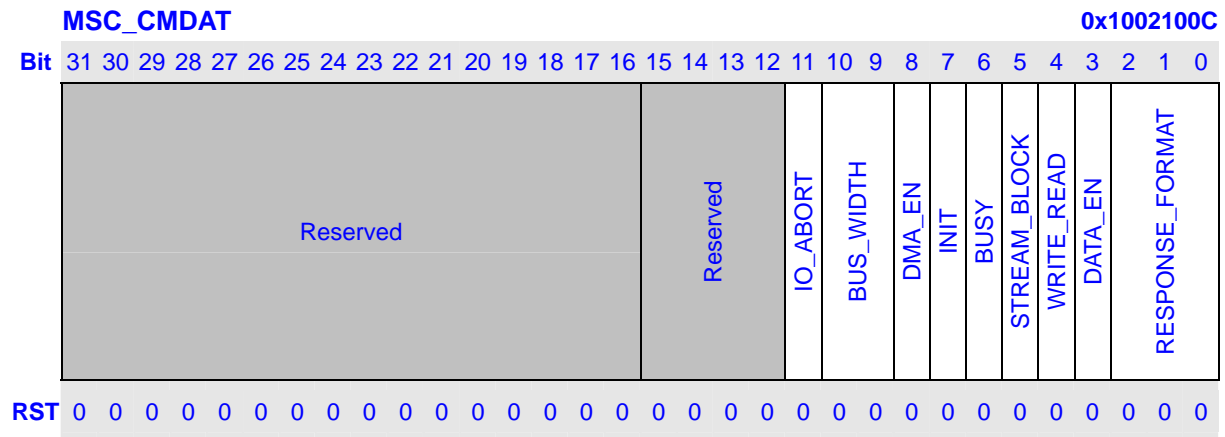
### 1.4.3 MSC Clock Rate Register (MSC\_CLKRT)

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

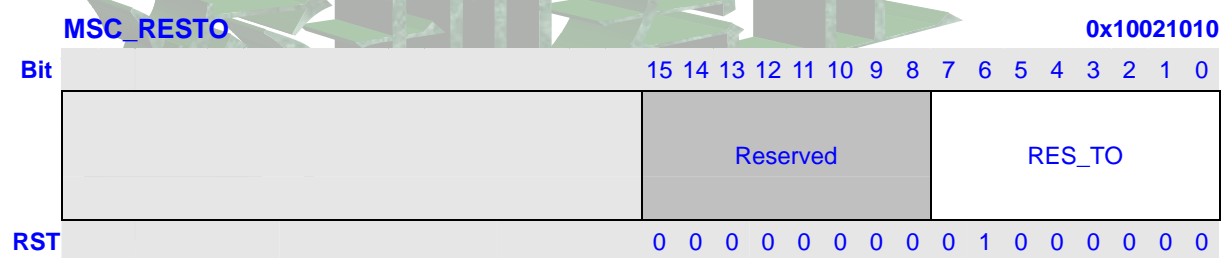
### 1.4.4 MMC/SD Command and Data Control Register (MSC\_CMDAT)



Bits	Name	Description	RW
31:12	Reserved		R
11	IO_ABORT	Specifies the current command is used to abort data transfer. <b>Only used for SDIO.</b> 0 – Nothing. 1 – The current command is used to abort transfer.	WR
10:9	BUS_WIDTH	Specifies the width of the data bus. 00 – 1-bit. 01 – Reserved. 10 – 4-bit. 11 – Reserved.	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. 0 – 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	WR

		stream mode.	
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 is a read operation. 1 – Specifies that the data transfer of the current command is a write operation.	WR
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 – Reserved.	WR

#### 1.4.5 MMC/SD Response Time Out Register (MSC\_RESTO)



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

**1.4.6 MMC/SD Read Time Out Register (MSC\_RDTO)**

MSC_RDTO																0x10021014															
Bit																15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											READ_TO																				
RST																1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

Bits	Name	Description	RW
15:0	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 CLK_SRC / 256.	WR

**1.4.7 MMC/SD Block Length Register (MSC\_BLKLEN)**

MSC_BLKLEN																0x10021018														
Bit															15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															Reserved				BLK_LEN											
RST															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	Name	Description	RW
15:12	Reserved		R
11:0	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

**1.4.8 MSC/SD Number of Block Register (MSC\_NOB)**

MSC_NOB																0x1002101C														
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

**1.4.9 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\_SNOB****0x10021020**

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

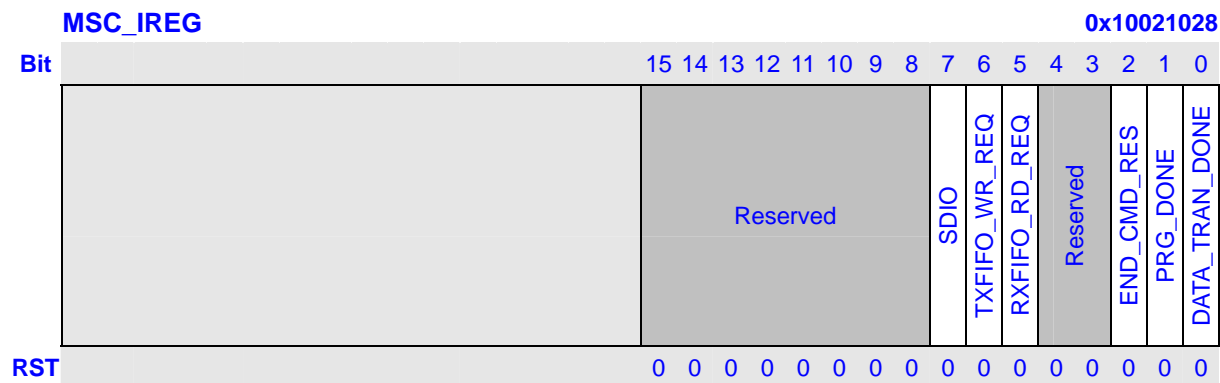
**1.4.10 MMC/SD Interrupt Mask Register (MSC\_IMASK)****MSC\_IMASK****0x10021024**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Reserved								SDIO		TXFIFO_WR_REQ	RXFIFO_RD_REQ	Reserved		END_CMD_RES	PRG_DONE	DATA_TRAN_DONE
RST	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	

Bits	Name	Description	RW
15:8	Reserved		R
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

### 1.4.11 MMC/SD Interrupt Register (MSC\_IREG)

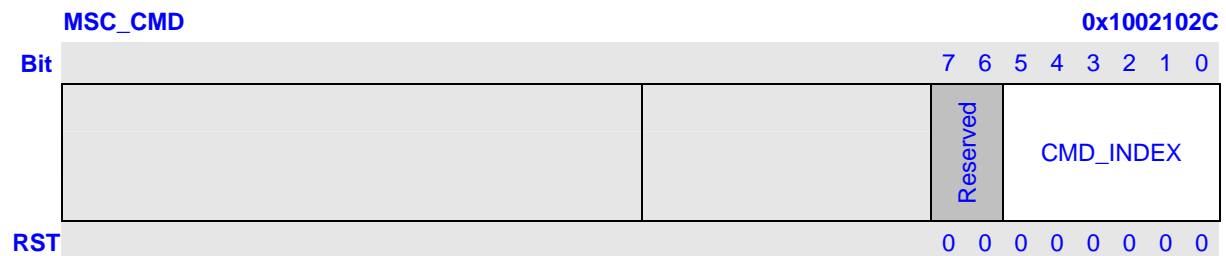
The MSC\_IREG register shows the currently requested interrupt. The FIFO request interrupts, TXFIFO\_WR\_REQ, and RXFIFO\_RD\_REQ are masked off with the DMA\_EN bit in the MSC\_CMDAT register. The software is responsible for monitoring these bit in program I/O mode.



Bits	Name	Description	RW
15:8	Reserved		R
7	SDIO	Indicates whether the interrupt from SDIO is detected. 0 – The interrupt from SDIO is not detected. 1 – The interrupt from SDIO is detected.	R
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.	WR

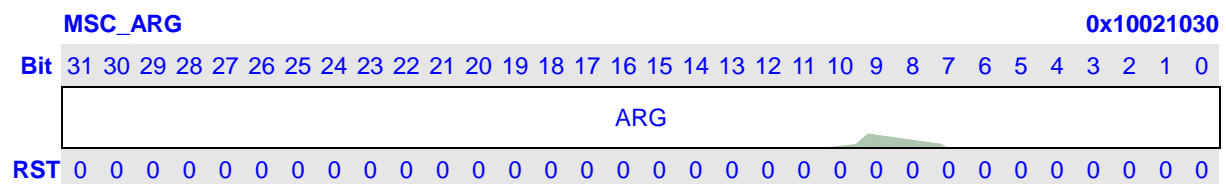
		0 – Data transfer is not complete. 1 – Data transfer has completed or an error has occurred. Write 1 to clear.	
--	--	--	--

#### 1.4.12 MMC/SD Command Index Register (MSC\_CMD)



Bits	Name	Description	RW
7:6	Reserved		R
5:0	CMD_INDEX	Specifies the command index to be executed.	WR

#### 1.4.13 MMC/SD Command Argument Register (MSC\_ARG)

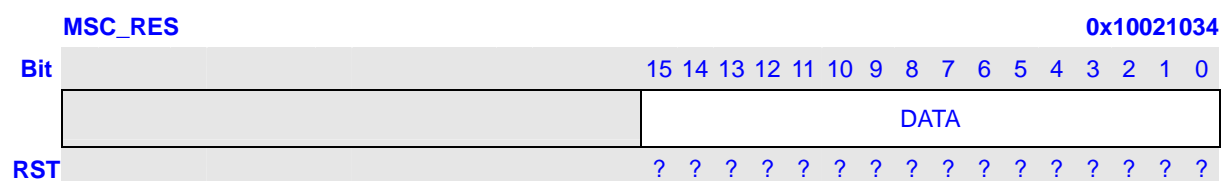


Bits	Name	Description	RW
31:0	ARG	Specifies the argument for the current command.	WR

#### 1.4.14 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 half-word read from the response FIFO is the most significant half-word of the received response.



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

#### 1.4.15 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_RXFIFO																0x10021038																
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

#### 1.4.16 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_TXFIFO																0x1002103C																
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

## 1.5 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.

### 1.5.1 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.

### 1.5.2 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).

### 1.5.3 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  $V_{dd}$  range desired by the host. This is accomplished by the host sending the required  $V_{dd}$  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.

### 1.5.4 Card Registry

Card registry on MCC and SD card are different.

For SD card, Identification process start at clock rate  $F_{od}$ , 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 than 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  $F_{od}$ . The open drain driver stages on the CMD line allow parallel card operation during card identification. After the bus is activated 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, that is CM2 and CMD3, until the host receive time-out condition to recognize completion of the identification process.

## **1.5.5 Card Access**

### **1.5.5.1 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.

### **1.5.5.2 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.

### **1.5.5.3 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.

### **1.5.5.4 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).

## **1.5.6 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.

### **1.5.6.1 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.



### 1.5.6.2 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.

### 1.5.6.3 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 1-5 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:

- Setting the Password:
  - Select a card (CMD7), if not previously selected already
  - 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.
  - 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.
  - 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 that 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.

- Reset the password:
  - Select a card (CMD7), if not previously selected already
  - 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.
  - 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.
- Locking a card:
  - Select a card (CMD7), if not previously selected already
  - 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.
  - 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 that 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.

- Unlocking the card
  - Select a card (CMD7), if not previously selected already.
  - 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.
  - 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 that 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.

- 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.

  - Select a card (CMD7), if not previously selected already.
  - 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 LOCK\_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.

### 1.5.7 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 1-6 Card Status Description**

Bits	Identifier	Type	Description	Clear Condition
31	OUT_OF_RANGE	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. 1 – 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	-	-	-

### 1.5.8 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 1-7 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			



### 1.5.9 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.

#### 1.5.9.1 SDIO Interrupts

In order to allow the SDIO card to interrupt the host, an 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.

#### 1.5.9.2 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 or 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:

1. The host determines which function currently used the DAT[] line(s).
2. The host requests the lower priority or slower transaction to suspend.



3. The host checks for the transaction suspension to complete.
4. The host begins the higher priority transaction.
5. The host waits for the completion of the higher priority transaction.
6. The host restores the suspended transaction.

### 1.5.9.3 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.

### 1.5.10 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).

### 1.5.11 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:

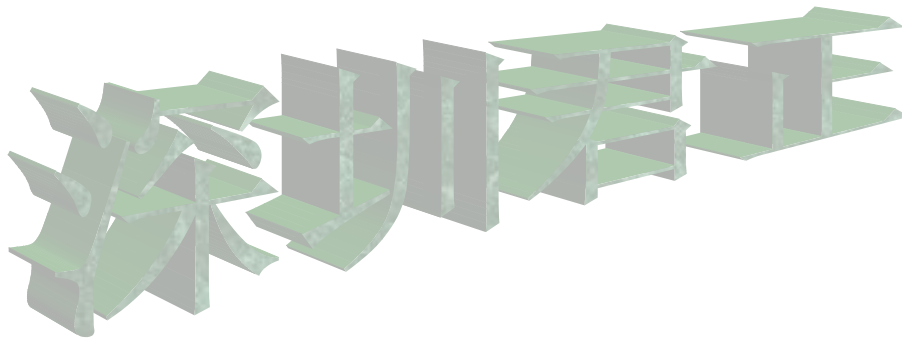
- Send APP\_CMD. The response will have the APP\_CMD bit (new status bit) set signaling to the host that ACMD is now expected.
- Send the required ACMD. The response will have the APP\_CMD bit set, indicating that the accepted command was interpreted as ACMD. If a non-ACMD is sent then it will be respected 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).



## 1.6 MMC/SD Controller Operation

### 1.6.1 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.

#### 1.6.1.1 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.

#### 1.6.1.2 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.

### 1.6.2 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  $\geq 8$  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.

Note:

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.

### 1.6.3 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.

### 1.6.4 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:

1. Stop the clock.
2. Set MSC\_STRPCL register to 0x08 to reset MSC.
3. Wait while MSC\_STAT [IS\_RESETTING] is 1.
4. Set MSC\_CMD with CMD0.
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.
6. Start the clock.
7. Start the operation (write MSC\_STRPCL with 0x04)
8. Wait for the END\_CMD\_RES interrupt.
9. Set MSC\_CMD with CMD52.
10. Set MSC\_ARG with 0x88000C08
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.
12. Start the operation.
13. Wait for the END\_CMD\_RES interrupt.

### 1.6.5 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:

1. Check whether SDIO card is inserted.
2. Check whether SDMEM card is inserted.
3. Check whether MMC cards are inserted.

#### 1.6.5.1 Check SDIO

The commands are sent as follows:

1. (Optional) Send CMD52 (IO\_RW\_DIRECT) with argument 0x88000C08 to reset SDIO card.
2. Send CMD5 (IO\_SEND\_OP\_CMD) to validate voltage.
3. If the response is correct and the number of IO functions > 0, then continue, else go to check SDMEM.
4. If C-bit in the response is ready (the initialization has finished), go to 6.
5. Send CMD5 (IO\_SEND\_OP\_CMD) to validate voltage, then go to 4.
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.
7. If it is a combo card, go to check SDMEM to initialize the memory part.
8. Send CMD3 (SET\_RELATIVE\_ADDR) to let the card publish a RCA. The RCA is returned from the response.
9. If do not accept the new RCA, go to 8, else record the new RCA.
10. Go to check MMC, because we can assure that there is no SDMEM card.

### 1.6.5.2 Check SDMEM

If there is no SDIO card or there is a combo card, continue to check SDMEM.

The commands are sent as follows:

1. (Optional) Send CMD0 (GO\_IDLE\_STATE) to reset MMC and SDMEM card. This command has no response.
2. Send CMD55. Here the default RCA 0x0000 is used for CMD55.
3. If the response is correct (CMD55 has response), then continue, else go to check MMC.
4. Send ACMD41 (SD\_SEND\_OP\_CMD) to validate voltage (the general OCR value is 0x00FF8000).
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.)
6. Send CMD55 and ACMD41 to validate voltage, and then go to 5.
7. Send CMD2 (ALL\_SEND\_CID) to get the card CID.
8. Send CMD3 (SET\_RELATIVE\_ADDR) to let card publish a RCA. The RCA is returned from the response.
9. If do not accept the new RCA, go to 8, else record the new RCA.
10. Go to check MMC.

### 1.6.5.3 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:

1. Send CMD1 (SEND\_OP\_CMD) to validate voltage (the general OCR value is 0x00FF88000).
2. If the response is correct, then continue, else goto 9.
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.)

4. Send CMD1 (SEND\_OP\_CMD) to validate voltage, and then go to 3.
5. Send CMD2 (ALL\_SEND\_CID) to get the card CID.
6. If the response timeout occurs, goto 9.
7. Send CMD3 (SET\_RELATIVE\_ADDR) to assign the card a RCA.
8. If there are other MMC cards, then go to 5.
9. Finish.

### 1.6.6 Single Data Block Write

In a single block write command, the following registers must be set before the operation is started:

1. Set MSC\_NOB register to 0x0001.
2. Set MSC\_BLKLEN to the number of bytes per block.
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.
4. Start the operation.
5. 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 [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.
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.
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.

### 1.6.7 Single Block Read

In a single block read command, the following registers must be set before the operation is started:



1. Set MSC\_NOB register to 0x0001.
2. Set MSC\_BLKLEN register to the number of bytes per block.
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.
4. Start the operation.
5. 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 [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.
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.

### 1.6.8 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 1-8 How to stop multiple block write**

Operation	Stop condition	Software processing
Open-ended or SDIO infinite	After write MSC_NOB blocks	<ol style="list-style-type: none"> <li>1. Wait for DATA_TRAN_DONE interrupt</li> <li>2. Send CMD12 or CMD52 (IO abort)</li> <li>3. Wait for END_CMD_RES and PRG_DONE interrupt</li> </ol>
Open-ended or SDIO	Stop writing in advance (not	<ol style="list-style-type: none"> <li>1. Set MSC_STRPCL [EXIT_MULTIPLE]</li> </ol>



infinite	write MSC_NOB blocks)	<ol style="list-style-type: none"> <li>2. Wait for DATA_TRAN_DONE interrupt</li> <li>3. Send CMD12 or CMD52 (IO abort)</li> <li>4. Wait for END_CMD_RES and PRG_DONE interrupt.</li> </ol>
Predefined block or SDIO finite	After writing MSC_NOB blocks	<ol style="list-style-type: none"> <li>1. Wait for DATA_TRAN_DONE interrupt</li> </ol>
Predefined block or SDIO finite	Stop writing in advance (not write MSC_NOB blocks)	<ol style="list-style-type: none"> <li>1. Set MSC_STRPCL [EXIT_MULTIPLE]</li> <li>2. Wait for DATA_TRAN_DONE interrupt</li> <li>3. Send CMD12 or CMD52 (IO abort)</li> <li>4. Wait for END_CMD_RES and PRG_DONE interrupt</li> </ol>

### 1.6.9 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 1-9 How to stop multiple block read**

Operation	Stop condition	Software processing
Open-ended or SDIO infinite	After reading MSC_NOB blocks	<ol style="list-style-type: none"> <li>1. Wait for DATA_TRAN_DONE interrupt</li> <li>2. Send CMD12 or CMD52 (IO abort)</li> <li>3. Wait for END_CMD_RES interrupt</li> </ol>
Open-ended or SDIO infinite	Stop reading in advance (not write MSC_NOB blocks)	<ol style="list-style-type: none"> <li>1. Set MSC_STRPCL [EXIT_MULTIPLE]</li> <li>2. Wait for DATA_TRAN_DONE interrupt</li> <li>3. Send CMD12 or CMD52 (IO abort)</li> <li>4. Wait for END_CMD_RES interrupt</li> </ol>
Predefined block or SDIO finite	After reading MSC_NOB blocks	<ol style="list-style-type: none"> <li>1. Wait for DATA_TRAN_DONE interrupt</li> </ol>
Predefined block or SDIO finite	Stop reading in advance (not write MSC_NOB blocks)	<ol style="list-style-type: none"> <li>1. Set MSC_STRPCL [EXIT_MULTIPLE]</li> <li>2. Wait for DATA_TRAN_DONE interrupt</li> <li>3. Send CMD12 or CMD52 (IO abort)</li> <li>4. Wait for END_CMD_RES interrupt</li> </ol>

### 1.6.10 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.

### 1.6.11 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.

#### 1.6.12 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.

#### 1.6.13 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.

#### 1.6.14 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.

#### 1.6.15 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 1-10 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				

Note: 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.

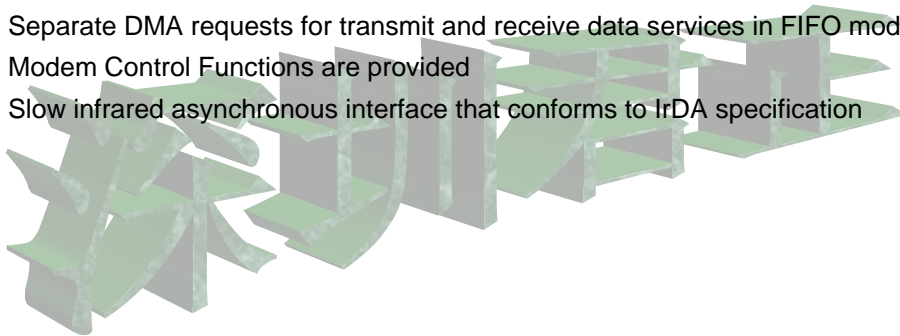
# 1 UART Interface

This chapter describes the universal asynchronous receiver/transmitter (UART) serial ports included in the JZ47XX Processor. The JZ47XX processor has four 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.

## 1.1 Overview

- 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
- 16x8bit transmit FIFO and 16x11bit receive FIFO
- Independently controlled transmit, receive (data ready or timeout), line status interrupts
- Baud rate generation allows up to 230.4Kbps
- 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
- Modem Control Functions are provided
- Slow infrared asynchronous interface that conforms to IrDA specification



## 1.2 Pin Description

**Table 1-1 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

**Note:** Jz4730 has four UART, UART3 support RxD, TxD, RTS\_, CTS\_, UART2, UART1 and UART0 support only RxD, TxD.

## 1.3 Register Description

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 1-2 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
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

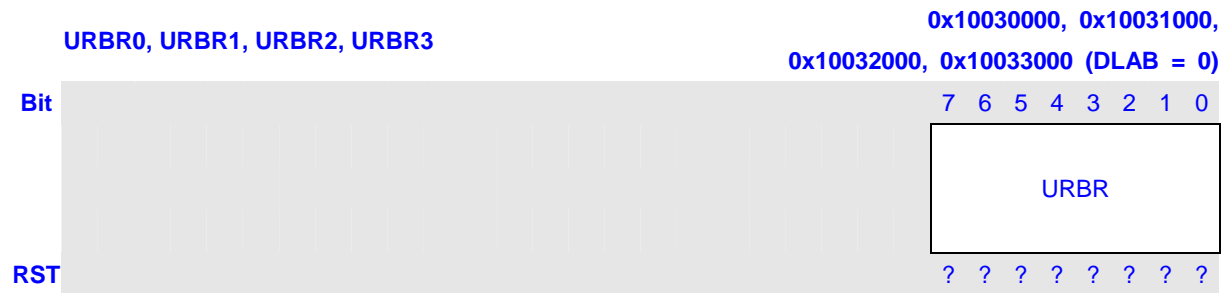


UIIR1	UART Interrupt Identification Register 1	R	0x01	0x10031008	8
UFCR1	UART FIFO Control Register 1	W	0x00	0x10031008	8
ULCR1	UART Line Control Register 1	RW	0x00	0x1003100C	8
UMCR1	UART Modem Control Register 1	RW	0x00	0x10031010	8
ULSR1	UART Line Status Register 1	R	0x00	0x10031014	8
UMSR1	UART Modem 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
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
URBR3	UART Receive Buffer Register 3	R	0x??	0x10033000	8
UTHR3	UART Transmit Hold Register 3	W	0x??	0x10033000	8
UDLLR3	UART Divisor Latch Low Register 3	RW	0x00	0x10033000	8
UDLHR3	UART Divisor Latch High Register 3	RW	0x00	0x10033004	8
UIER3	UART Interrupt Enable Register 3	RW	0x00	0x10033004	8
UIIR3	UART Interrupt Identification Register 3	R	0x01	0x10033008	8
UFCR3	UART FIFO Control Register 3	W	0x00	0x10033008	8
ULCR3	UART Line Control Register 3	RW	0x00	0x1003300C	8
UMCR3	UART Modem Control Register 3	RW	0x00	0x10033010	8
ULSR3	UART Line Status Register 3	R	0x00	0x10033014	8
UMSR3	UART Modem Status Register 3	R	0x00	0x10033018	8
USPR3	UART ScratchPad Register 3	RW	0x00	0x1003301C	8
ISR3	Infrared Selection Register 3	RW	0x00	0x10033020	8

### 1.3.1 UART Receive Buffer Register (URBR)

The read-only URBR is corresponded to one level 11bit buffer in non-FIFO mode and a 16x11bit 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.





Bits	Name	Description	RW
7:0	URBR	8-bit UART receive read data	R

### 1.3.2 UART Transmit Hold Register (UTHR)

The write-only UTHR is corresponded to one level 8 bit buffer in non-FIFO mode and a 16x8bit FIFO in FIFO mode that holds the data byte(s) to be transmitted next.



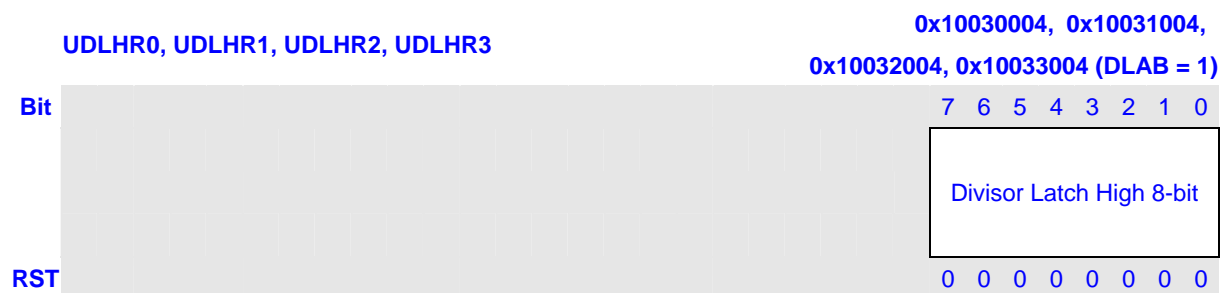
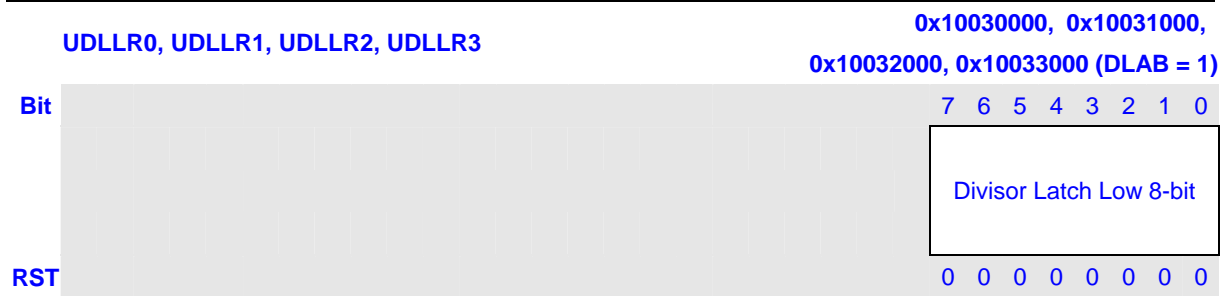
Bits	Name	Description	RW
7:0	UTHR	8-bit UART transmit write hold data	W

### 1.3.3 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 3.6864-MHz fixed-input clock and divide it by 1 to ( $2^{16} - 1$ ). 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. The maximum baud rate is 230.4kbps.

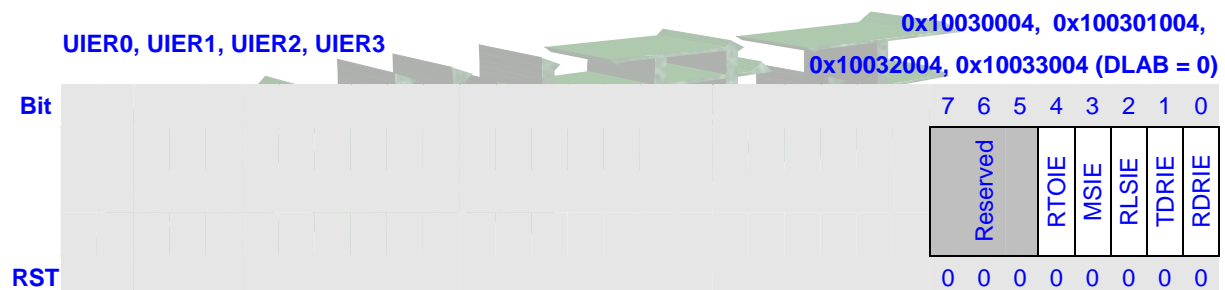
The relationship of baud rate and the value of Divisor is shown by the formula:

$$\text{Baud Rate} = 3.6864 \text{ Mhz} / (16 * \text{Divisor})$$



### 1.3.4 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.

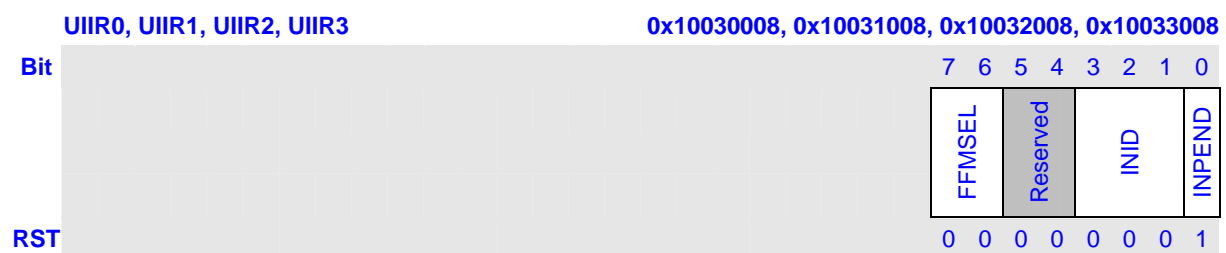


Bits	Name	Description	RW
7:5	Reserved	Always read 0, write is ignored	R
4	RTOIE	<b>Receive Timeout Interrupt Enable</b> 0 = Disable the receive timeout interrupt 1 = Enable the receive timeout interrupt  Timeout means the URDR (FIFO mode) is not empty but no character has received for a period of time T: T (bits) = 4 X Word length + 12	RW
3	MSIE	<b>Modem Status Interrupt Enable</b> 0 = Disable the modem status interrupt 1 = Enable the modem status interrupt	RW
2	RLSIE	<b>Receive Line Status Interrupt Enable</b>	RW

		0 = Disable receive line status interrupt 1 = Enable receive line status interrupt	
1	TDRIE	<b>Transmit Data Request Interrupt Enable</b> 0 = Disable the transmit data request interrupt 1 = Enable the transmit data request interrupt	RW
0	RDRIE	<b>Receive Data Ready Interrupt Enable</b> 0 = Disable the receive data ready interrupt 1 = Enable the receive data ready interrupt	RW

### 1.3.5 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.



Bits	Name	Description	RW																		
7:6	FFMSEL	<b>FIFO Mode Select</b> 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	<b>Interrupt Identifier</b> These bits identify the current highest priority pending interrupt. <table><tr><th>INID</th><th>Description</th></tr><tr><td>0b000</td><td>Modem Status</td></tr><tr><td>0b001</td><td>Transmit Data Request</td></tr><tr><td>0b010</td><td>Receive Data Ready</td></tr><tr><td>0b011</td><td>Receive Line Status</td></tr><tr><td>0b100</td><td>Reserved</td></tr><tr><td>0b101</td><td>Reserved</td></tr><tr><td>0b110</td><td>Receive Time Out</td></tr><tr><td>0b111</td><td>Reserved</td></tr></table>	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	R
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																				

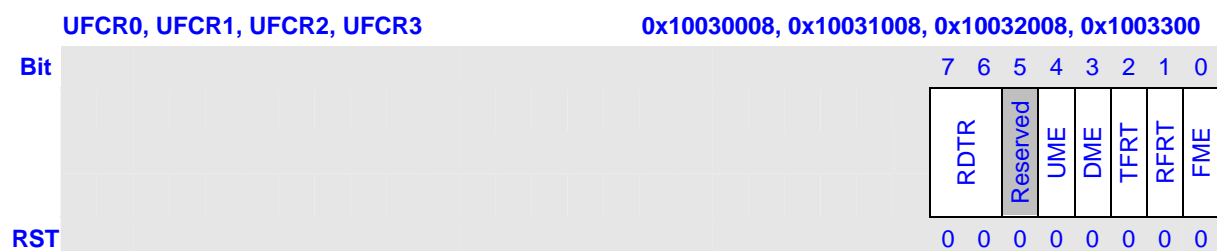
		See Table 1-3 for details	
0	INPEND	<b>Interrupt Pending</b> 0 = interrupt is pending 1 = No interrupt pending	R

Table 1-3 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

### 1.3.6 UART FIFO Control Register (UFCR)

The write-only register UFCR contains the control bits for receive and transmit FIFO.



Bits	Name	Description	RW
------	------	-------------	----

7:6	RDTR	<b>Receive Buffer Data Number Trigger</b> These bits are used to select the trigger level for the receive data ready interrupt in FIFO mode. 0b00 = 1 0b01 = 4 0b10 = 8 0b11 = 15	W
5	Reserved	Always read 0, write is ignored	R
4	UME	<b>UART Module Enable</b> 0 = Disable UART 1 = Enable UART	W
3	DME	<b>DMA Mode Enable</b> 0 = Disable DMA mode 1 = Enable DMA mode	W
2	TFRT	<b>Transmit Holding Register Reset</b> 0 = Not reset 1 = Reset transmit FIFO	W
1	RFRT	<b>Receive Buffer Reset</b> 0 = Not reset 1 = Reset receive FIFO	W
0	FME	<b>FIFO Mode Enable</b> Set this bit before the trigger levels. 0 = non-FIFO mode 1 = FIFO mode	W

### 1.3.7 UART Line Control Register (ULCR)

The ULCR defines the format for UART data transmission.

ULCR0, ULCR1, ULCR2, ULCR3

0x1003000C, 0x1003100C

0x1003200C, 0x1003300C

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	<b>Divisor Latch Access Bit</b> 0 = Enable to access URBR, UTHR or UIER 1 = Enable to access UDLLR or UDLHR	W
6	SBK	<b>Set Break</b>	W

		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	
5	STPAR	<b>Sticky Parity</b> Setting this bit forces parity location to be opposite of PARM bit when PARE is 1 (it is ignored when PARE is 0). 0 = Disable Sticky parity 1 = Enable Sticky parity (opposite of PARM bit)	W
4	PARM	<b>Parity Odd/Even Mode Select</b> If PARE = 0, PARM is ignored 0 = Odd parity 1 = Even parity	W
3	PARE	<b>Parity Enable</b> Enables a parity bit to be generated on transmission or checked on reception. 0 = No parity 1 = Parity	W
2	SBLS	<b>Stop Bit Length Select</b> 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	<b>Word Length Select</b> 0b00 = 5-bit character 0b01 = 6-bit character 0b10 = 7-bit character 0b11 = 8-bit character	W

### 1.3.8 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.

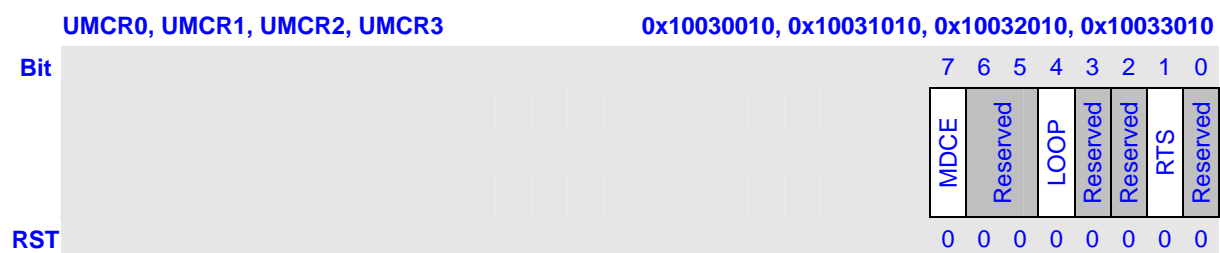
ULSR0, ULSR1, ULSR2, ULSR3				0x10030014, 0x10031014, 0x10032014, 0x10033014							
Bit	7	6	5	4	3	2	1	0			
	FIFOE	TEMP	TDRQ	BI	FMER	PARER	OVER	DRY			
RST	0	1	1	0	0	0	0	0			

Bits	Name	Description	RW
7	FIFOE	<p><b>FIFO Error Status (FIFO mode only)</b></p> <p>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.</p> <p>During DMA transfer, the error interrupt generates when FIFOE is 1, and 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.</p> <p>0 = No error data in receive buffer or non-FIFO mode 1 = One or more error character in receive buffer</p>	R
6	TEMP	<p><b>Transmit Holding Register Empty</b></p> <p>Set when both UTHR and shift register are empty. It is cleared when either the UTHR or the shift register contains a data character.</p> <p>0 = There is data in the transmit shifter and UTHR 1 = All the data in the transmit shifter and UTHR has been shifted out</p>	R
5	TDRQ	<p><b>Transmit Data Request</b></p> <p>Set when UTHR has half or more empty location (FIFO mode) or empty (non-FIFO mode).</p> <p>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.</p> <p>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</p>	R
4	BI	<p><b>Break Interrupt</b></p> <p>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.</p> <p>0 = No break signal has been received 1 = Break signal received</p>	R
3	FMER	<p><b>Framing Error</b></p> <p>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</p>	R

		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	
2	PARER	<b>Parity Error</b> 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 of 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	R
1	OVER	<b>Overrun Error</b> 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	<b>Data Ready</b> 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

### 1.3.9 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.



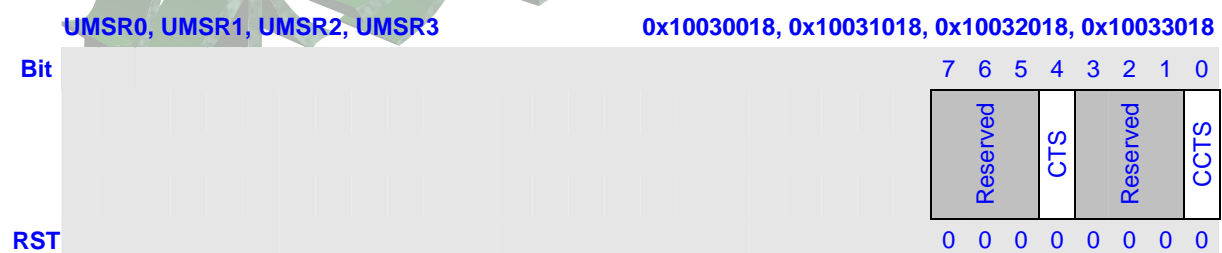
Bits	Name	Description	RW
------	------	-------------	----



7	MDCE	<b>Modem Control Enable</b> 0 = Modem function is disabled 1 = Modem function is enabled	W
6:5	Reserved	Always read 0, write is ignored	R
4	LOOP	<b>Loop Back</b> 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	<b>Request To Send</b> 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

### 1.3.10 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.

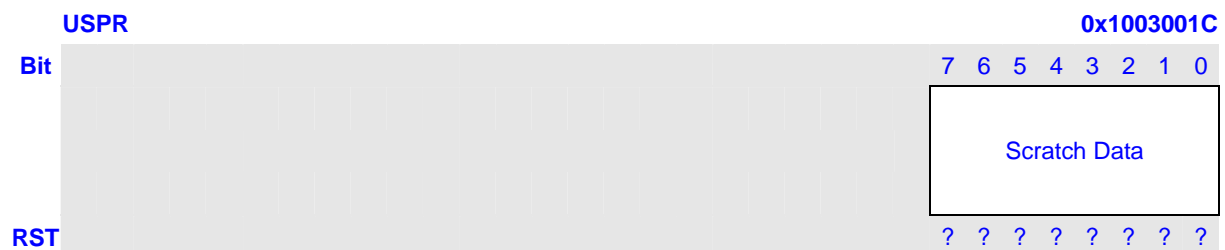


Bits	Name	Description	RW
7	Reserved	Always read 0, write is ignored	R
6	Reserved	Always read 0, write is ignored	R
5	Reserved	Always read 0, write is ignored	R
4	CTS	<b>Status of Clear To Send</b> When MDCE bit is 1, this bit is the complement of CTS_ input. If Loop bit of UMCR is 1, this bit is equivalent to RTS bit of UMCR. 0 = CTS_ pin is 1	R

		1 = CTS_ pin is 0	
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	<b>Change status of CTS_</b> 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

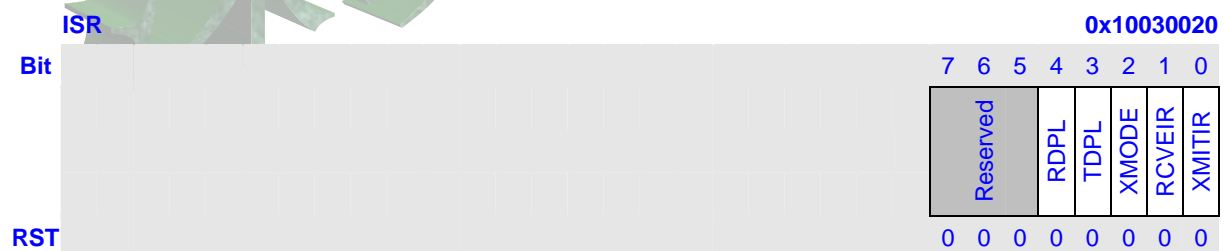
### 1.3.11 UART Scratchpad Register

This Scratchpad register is used as a scratch register for the programmer and has no effect on the UART.



### 1.3.12 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.



Bits	Name	Description	RW
7:5	Reserved	Always read 0, write is ignored	R
4	RDPL	<b>Receive Data Polarity</b> 0 = Slow-infrared (SIR) interface decoder takes positive pulses as zeros. 1 = SIR decoder takes negative pulses as zeros.	W
3	TDPL	<b>Transmit Data Polarity</b> 0 = SIR encoder generates a positive pulse for a data bit of zero.	W

		1 = SIR encoder generates a negative pulse for a data bit of zero.	
2	XMODE	<b>Transmit Pulse Width Mode</b> 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 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.	W
1	RCVEIR	<b>Receiver SIR Enable</b> 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	<b>Transmitter SIR Enable</b> 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

## 1.4 Operation

The following sections describe the UART operations that include flow of configuration, data transmission, data reception, and Infra-red mode.

### 1.4.1 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

### 1.4.2 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

### 1.4.3 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

#### 1.4.4 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

#### 1.4.5 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.

#### 1.4.6 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.

#### 1.4.7 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.

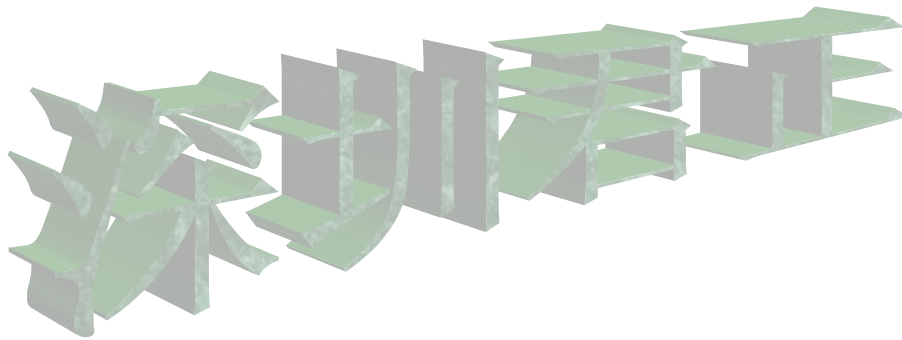
In SIR transmit mode, the transmit pulse comes out at a rate of 3/16 (when the transmit data bit is zero); in SIR receive mode, the receiver must detect the 3/16 pulsed period to recognize a zero

---

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.



# 1 Smart Card Controller

## 1.1 Overview

Smart Card Controller (SCC) interface is a primary device and communications interface for kinds of IC cards. The SCC interface supports communication with smart cards as specified in standard ISO7816-3. There are two SCC interfaces integrated in this SOC. And they perform the same functions. The SCC interface supports T=0 and T=1 protocol defined in ISO7816-3.

Software controls the session between the SCC interface and the card by SCC registers. Choosing protocol type and parameters, receiving and sending a byte to/from the card, activating/deactivating the card, and similar operations are accomplished with read/write operations to SCC registers. Transforming byte convention (inverse to direct and vice-versa, according to the session convention) is performed within SCC. Hence, software does not have to perform format inversion before character receipt. The SCC interface provides functionality to support the above standards, but it is the responsibility of software to ensure the standards are met.

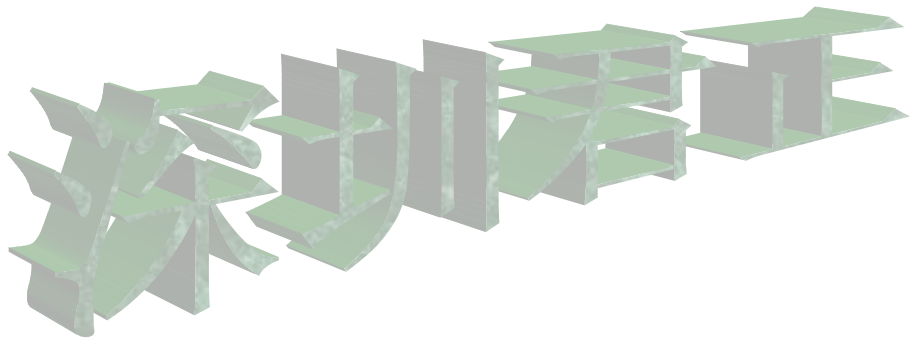
Features:

- Supports normal card and UIM card.
- 8-bit, 16-level receive-/transmit- FIFO.
- Supports asynchronous character (T=0) communication modes.
- Supports asynchronous block (T=1) communication modes.
- Supports setting of clock-rate conversion factor F (372, 512, 558, etc.), and bit-rate adjustment factor D (1, 2, 4, 8, 16, 32, 12, 20, etc.).
- Supports extra guard time waiting.
- Auto-error detection in T=0 receive mode.
- Auto-character repeat in T=0 transmit mode.
- Transforms inverted format to regular format and vice versa.
- Support stop clock function in some power consuming sensitive applications.

1.2 Pin Description

Table 1-1 Smart Card Controller Pins Description

Name	I/O	Description
SCC_CLK	Output	Serial clock connects SCC and the card.
SCC_DAT	Input/Output	Data communication pin.



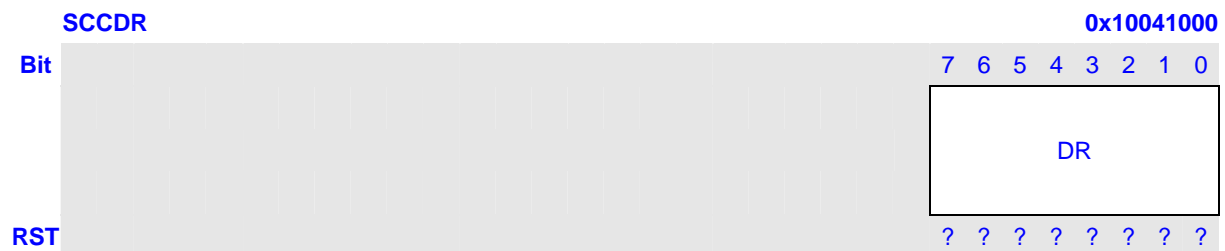


### 1.3 Register Description

Table 1-2 Smart Card Controller Registers Description

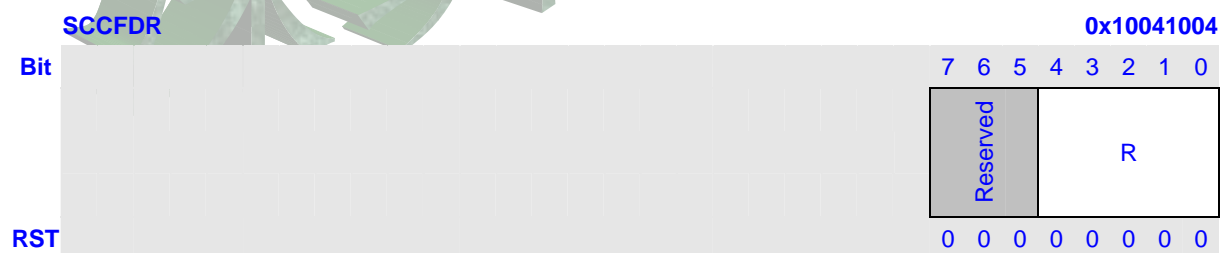
Name	RW	Reset Value	Address	Access Size
SCCDR	RW	0x??	0x10041000	8
SCCFDR	R	0x00	0x10041004	8
SCCCR	RW	0x00000000	0x10041008	32
SCCSR	RW	0x8000	0x1004100C	16
SCCTFR	RW	0x0173	0x10041010	16
SCCEGTR	RW	0x00	0x10041014	8
SCCECR	RW	0x00000000	0x10041018	32
SCCRTOR	RW	0x00	0x1004101C	8

#### 1.3.1 Transmit/Receive FIFO Data Register (SCCDR)



Bits	Name	Description	RW
7:0	DR	Data port of HW FIFO.	RW

#### 1.3.2 FIFO Data Count Register (SCCFDR)



Bits	Name	Description	RW
7:5	Reserved		R
4:0	R	Characters resisted in FIFO	R

## 1.3.3 Control Register (SCCCR)

SCCCR

0x10041008

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
	SCCE	TRS	T2R	Reserved		FDIV	FLUSH	Reserved		TRIG	TP	CONV	TXIE	RXIE	TENDIE	RTOIE	ECIE	EPIE	RETIE	EOIE	Reserved	TSEND	PX	CLKSTP								
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	SCCE	Enables or disables SCC. When disable it, the SCC_CLK will be stopped.	RW										
30	TRS	Transmit or Receive Select. 0 – Reception mode. 1 – Transmission mode.	RW										
29	T2R	Auto-T2R support. T2R means Transmit turn to Reception. 0 – controlled by SW. 1 – controlled by HW.	RW										
28:26	Reserved		R										
25:24	FDIV	Frequency Divider Select. <table><tr><td></td><td><b>SCC_CLK frequency</b></td></tr><tr><td>00</td><td>Same as device clk</td></tr><tr><td>01</td><td>Half of device clk</td></tr><tr><td>10:11</td><td>Reserved</td></tr></table>		<b>SCC_CLK frequency</b>	00	Same as device clk	01	Half of device clk	10:11	Reserved	RW		
	<b>SCC_CLK frequency</b>												
00	Same as device clk												
01	Half of device clk												
10:11	Reserved												
23	FLUSH	Flush FIFO. 0 – Does not empty the Rx/Tx FIFO. 1 – Empty the Rx/Tx FIFO.	RW										
22:18	Reserved		R										
17:16	TRIG	Receive/Transmit FIFO trigger. <table><tr><td></td><td><b>Trigger Value</b></td></tr><tr><td>00</td><td>1</td></tr><tr><td>01</td><td>4</td></tr><tr><td>10</td><td>8</td></tr><tr><td>11</td><td>14</td></tr></table>		<b>Trigger Value</b>	00	1	01	4	10	8	11	14	RW
	<b>Trigger Value</b>												
00	1												
01	4												
10	8												
11	14												
15	TP	Communicate protocol. 0 – T=0, 1 – T=1	RW										
14	CONV	Card data transfer convention. 0 – LSB first. 1 – MSB first and inverted.	RW										
13	TXIE	Tx FIFO counter meets the trigger value interrupt enable bit.	RW										
12	RXIE	Rx FIFO counter meets the trigger value interrupt enable bit.	RW										
11	TENDIE	Transmission finished interrupt enable bit. (Both FIFO and transmitter are empty)	RW										
10	RTOIE	Reception timeout interrupt enable bit.	RW										
9	ECIE	ETU counter overflow interrupt enable bit.	RW										
8	EPIE	Parity error interrupt enable bit.	RW										
7	RETIE	Re-transmitting 3 times interrupt enable bit.	RW										
6	EOIE	Receive overrun error interrupt enable bit.	RW										
5:4	Reserved		R										
3	TSEND	Receive TS character stage finished bit. 0 – TS character has not been	RW										

		received. 1 – TS character has been received.											
2:1	PX	Parameter X. SCC stop clock mode selection. <table><tr><td></td><td><b>SCC clock stop</b></td></tr><tr><td>00</td><td>Does not support SCC clock stop.</td></tr><tr><td>01</td><td>SCC_CLK stops at state low.</td></tr><tr><td>10</td><td>SCC_CLK stops at state high.</td></tr><tr><td>11</td><td>Reserved</td></tr></table>		<b>SCC clock stop</b>	00	Does not support SCC clock stop.	01	SCC_CLK stops at state low.	10	SCC_CLK stops at state high.	11	Reserved	RW
	<b>SCC clock stop</b>												
00	Does not support SCC clock stop.												
01	SCC_CLK stops at state low.												
10	SCC_CLK stops at state high.												
11	Reserved												
0	CLKSTP	SCC clock stop. 0 – SCC has left or is leaving clock stop mode. 1 – SCC has entered or is entering clock stop mode.	RW										

### 1.3.4 Status Register (SCCSR)

<b>SCCSR</b>		<b>0x1004100C</b>	
Bit		15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
		TRANS Reserved ORER RTO PER TFTG RFTG TEND Reserved RETR_3 Reserved ECNTO	
RST		1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	

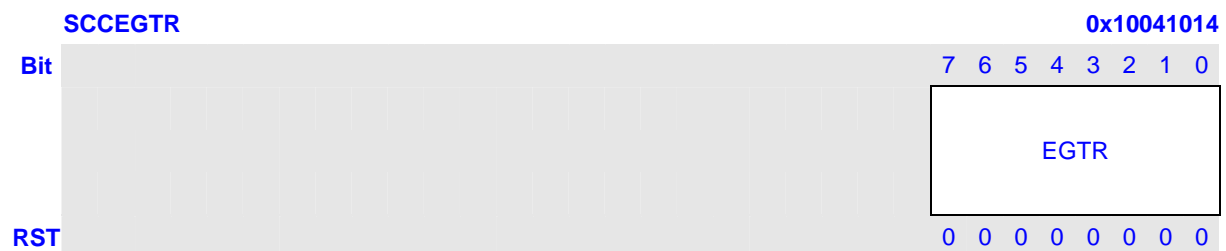
Bits	Name	Description	RW
15	TRANS	Transfer status. Specifies whether the transfer is stopped or not.	R
14:13	Reserved		R
12	ORER	Receive overrun error.	RW
11	RTO	Reception timeout.	R
10	PER	Parity error.	RW
9	TFTG	Hit Tx FIFO trigger.	R
8	RFTG	Hit Rx FIFO trigger.	R
7	TEND	Transmission end. Both Tx FIFO and transmitter are empty.	RW
6:5	Reserved		R
4	RETR_3	Re-transmit exceed 3 times.	RW
3:1	Reserved		R
0	ECNTO	ETU counter overflow.	RW

### 1.3.5 Transmission Factor Register (SCCTFR)

<b>SCCTFR</b>		<b>0x10041010</b>	
Bit		15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
		Reserved FD	
RST		0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 1	

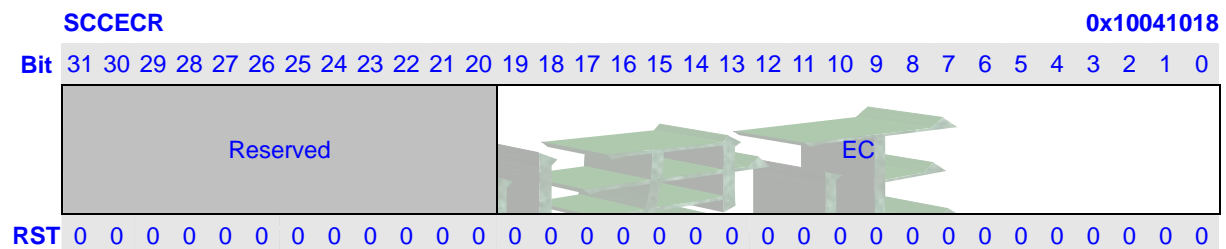
Bits	Name	Description	RW
15:11	Reserved		R
10:0	FD	Value of F/D. The initial value is 0x173(371).	RW

### 1.3.6 Extra Guard Timer Register (SCCEGTR)



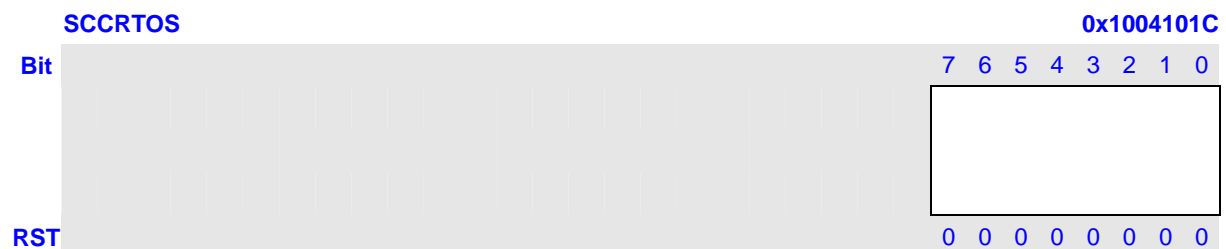
Bits	Name	Description	RW
7:0	EGTR	The register is corresponding with N value of ISO7816-3. Which indicates the extra-guard time of a transmission.	RW

### 1.3.7 ETU Counter Value Register (SCCECR)



Bits	Name	Description	RW
31:20	Reserved		R
19:0	EC	ETU counter. Write operation will clear the internal counter automatically.	RW

### 1.3.8 Reception Timeout Register (SCCRTOR)



Bits	Name	Description	RW
7:0	RTO	Retry times when parity error detected.	RW

# 1 Smart Card Controller

## 1.1 Overview

Smart Card Controller (SCC) interface is a primary device and communications interface for kinds of IC cards. The SCC interface supports communication with smart cards as specified in standard ISO7816-3. There are two SCC interfaces integrated in this SOC. And they perform the same functions. The SCC interface supports T=0 and T=1 protocol defined in ISO7816-3.

Software controls the session between the SCC interface and the card by SCC registers. Choosing protocol type and parameters, receiving and sending a byte to/from the card, activating/deactivating the card, and similar operations are accomplished with read/write operations to SCC registers. Transforming byte convention (inverse to direct and vice-versa, according to the session convention) is performed within SCC. Hence, software does not have to perform format inversion before character receipt. The SCC interface provides functionality to support the above standards, but it is the responsibility of software to ensure the standards are met.

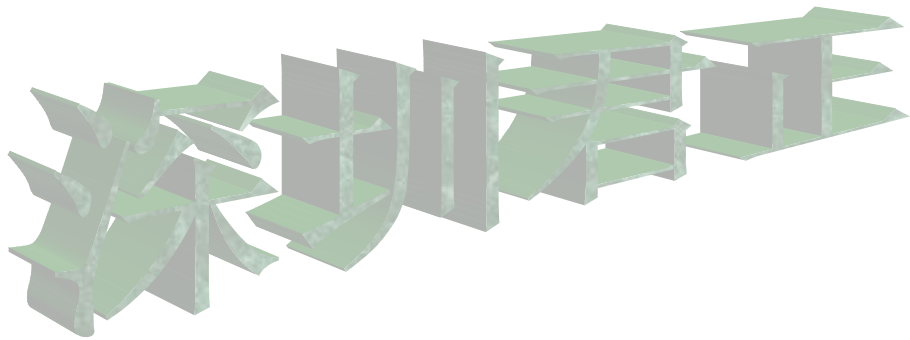
Features:

- Supports normal card and UIM card.
- 8-bit, 16-level receive-/transmit- FIFO.
- Supports asynchronous character (T=0) communication modes.
- Supports asynchronous block (T=1) communication modes.
- Supports setting of clock-rate conversion factor F (372, 512, 558, etc.), and bit-rate adjustment factor D (1, 2, 4, 8, 16, 32, 12, 20, etc.).
- Supports extra guard time waiting.
- Auto-error detection in T=0 receive mode.
- Auto-character repeat in T=0 transmit mode.
- Transforms inverted format to regular format and vice versa.
- Support stop clock function in some power consuming sensitive applications.

1.2 Pin Description

Table 1-1 Smart Card Controller Pins Description

Name	I/O	Description
SCC_CLK	Output	Serial clock connects SCC and the card.
SCC_DAT	Input/Output	Data communication pin.

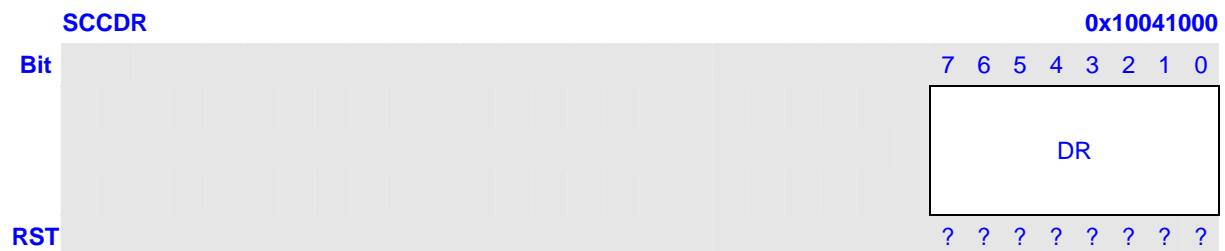


### 1.3 Register Description

Table 1-2 Smart Card Controller Registers Description

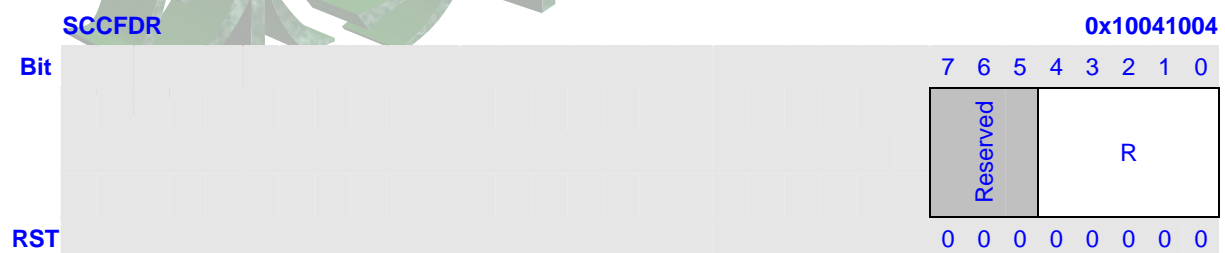
Name	RW	Reset Value	Address	Access Size
SCCDR	RW	0x??	0x10041000	8
SCCFDR	R	0x00	0x10041004	8
SCCCR	RW	0x00000000	0x10041008	32
SCCSR	RW	0x8000	0x1004100C	16
SCCTFR	RW	0x0173	0x10041010	16
SCCEGTR	RW	0x00	0x10041014	8
SCCECR	RW	0x00000000	0x10041018	32
SCCRTOR	RW	0x00	0x1004101C	8

#### 1.3.1 Transmit/Receive FIFO Data Register (SCCDR)



Bits	Name	Description	RW
7:0	DR	Data port of HW FIFO.	RW

#### 1.3.2 FIFO Data Count Register (SCCFDR)



Bits	Name	Description	RW
7:5	Reserved		R
4:0	R	Characters resisted in FIFO	R

## 1.3.3 Control Register (SCCCR)

SCCCR

0x10041008

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													
	SCCE	TRS	T2R	Reserved			FDIV		FLUSH		Reserved				TRIG			TP		CONV		TXIE		RXIE		TENDIE		RTOIE		ECIE		EPIE		RETIE		EOIE		Reserved		TSEND		PX		CLKSTP	
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	0	0	0	0	0	0	0	0	0	0			

Bits	Name	Description	RW										
31	SCCE	Enables or disables SCC. When disable it, the SCC_CLK will be stopped.	RW										
30	TRS	Transmit or Receive Select. 0 – Reception mode. 1 – Transmission mode.	RW										
29	T2R	Auto-T2R support. T2R means Transmit turn to Reception. 0 – controlled by SW. 1 – controlled by HW.	RW										
28:26	Reserved		R										
25:24	FDIV	Frequency Divider Select. <table><tr><td></td><td><b>SCC_CLK frequency</b></td></tr><tr><td>00</td><td>Same as device clk</td></tr><tr><td>01</td><td>Half of device clk</td></tr><tr><td>10:11</td><td>Reserved</td></tr></table>		<b>SCC_CLK frequency</b>	00	Same as device clk	01	Half of device clk	10:11	Reserved	RW		
	<b>SCC_CLK frequency</b>												
00	Same as device clk												
01	Half of device clk												
10:11	Reserved												
23	FLUSH	Flush FIFO. 0 – Does not empty the Rx/Tx FIFO. 1 – Empty the Rx/Tx FIFO.	RW										
22:18	Reserved		R										
17:16	TRIG	Receive/Transmit FIFO trigger. <table><tr><td></td><td><b>Trigger Value</b></td></tr><tr><td>00</td><td>1</td></tr><tr><td>01</td><td>4</td></tr><tr><td>10</td><td>8</td></tr><tr><td>11</td><td>14</td></tr></table>		<b>Trigger Value</b>	00	1	01	4	10	8	11	14	RW
	<b>Trigger Value</b>												
00	1												
01	4												
10	8												
11	14												
15	TP	Communicate protocol. 0 – T=0, 1 – T=1	RW										
14	CONV	Card data transfer convention. 0 – LSB first. 1 – MSB first and inverted.	RW										
13	TXIE	Tx FIFO counter meets the trigger value interrupt enable bit.	RW										
12	RXIE	Rx FIFO counter meets the trigger value interrupt enable bit.	RW										
11	TENDIE	Transmission finished interrupt enable bit. (Both FIFO and transmitter are empty)	RW										
10	RTOIE	Reception timeout interrupt enable bit.	RW										
9	ECIE	ETU counter overflow interrupt enable bit.	RW										
8	EPIE	Parity error interrupt enable bit.	RW										
7	RETIE	Re-transmitting 3 times interrupt enable bit.	RW										
6	EOIE	Receive overrun error interrupt enable bit.	RW										
5:4	Reserved		R										
3	TSEND	Receive TS character stage finished bit. 0 – TS character has not been	RW										



		received. 1 – TS character has been received.											
2:1	PX	Parameter X. SCC stop clock mode selection. <table><tr><td></td><td><b>SCC clock stop</b></td></tr><tr><td>00</td><td>Does not support SCC clock stop.</td></tr><tr><td>01</td><td>SCC_CLK stops at state low.</td></tr><tr><td>10</td><td>SCC_CLK stops at state high.</td></tr><tr><td>11</td><td>Reserved</td></tr></table>		<b>SCC clock stop</b>	00	Does not support SCC clock stop.	01	SCC_CLK stops at state low.	10	SCC_CLK stops at state high.	11	Reserved	RW
	<b>SCC clock stop</b>												
00	Does not support SCC clock stop.												
01	SCC_CLK stops at state low.												
10	SCC_CLK stops at state high.												
11	Reserved												
0	CLKSTP	SCC clock stop. 0 – SCC has left or is leaving clock stop mode. 1 – SCC has entered or is entering clock stop mode.	RW										

### 1.3.4 Status Register (SCCSR)

SCCSR		0x1004100C															
Bit		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		TRANS	Reserved	ORER	RTO	PER	TFTG	RFTG	TEND	Reserved	RETR_3	Reserved	Reserved	Reserved	Reserved	ECNTO	
RST		1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

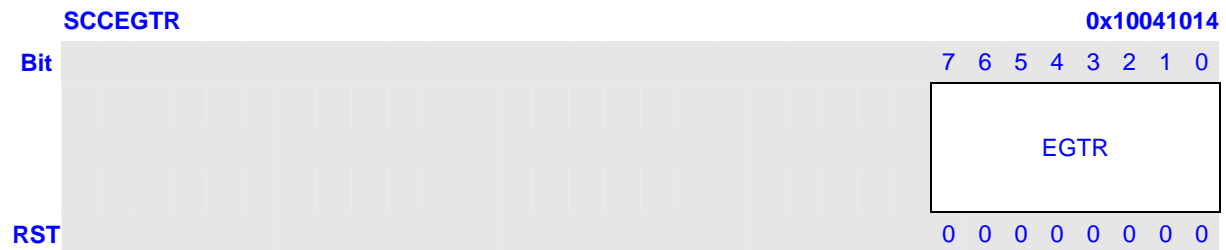
Bits	Name	Description	RW
15	TRANS	Transfer status. Specifies whether the transfer is stopped or not.	R
14:13	Reserved		R
12	ORER	Receive overrun error.	RW
11	RTO	Reception timeout.	R
10	PER	Parity error.	RW
9	TFTG	Hit Tx FIFO trigger.	R
8	RFTG	Hit Rx FIFO trigger.	R
7	TEND	Transmission end. Both Tx FIFO and transmitter are empty.	RW
6:5	Reserved		R
4	RETR_3	Re-transmit exceed 3 times.	RW
3:1	Reserved		R
0	ECNTO	ETU counter overflow.	RW

### 1.3.5 Transmission Factor Register (SCCTFR)

SCCTFR		0x10041010															
Bit		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		Reserved					FD										
RST		0	0	0	0	0	0	0	1	0	1	1	1	0	0	1	1

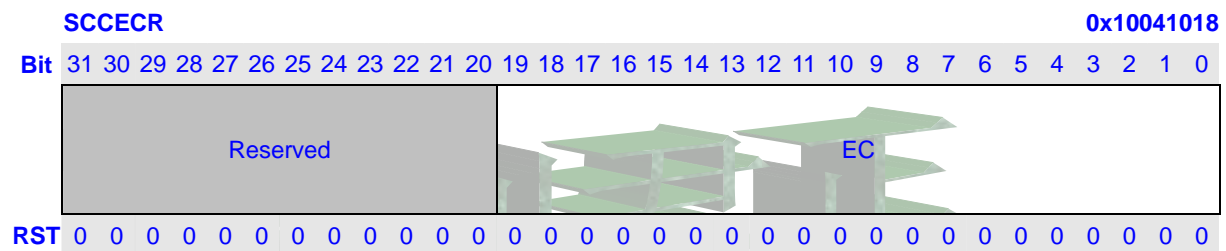
Bits	Name	Description	RW
15:11	Reserved		R
10:0	FD	Value of F/D. The initial value is 0x173(371).	RW

### 1.3.6 Extra Guard Timer Register (SCCEGTR)



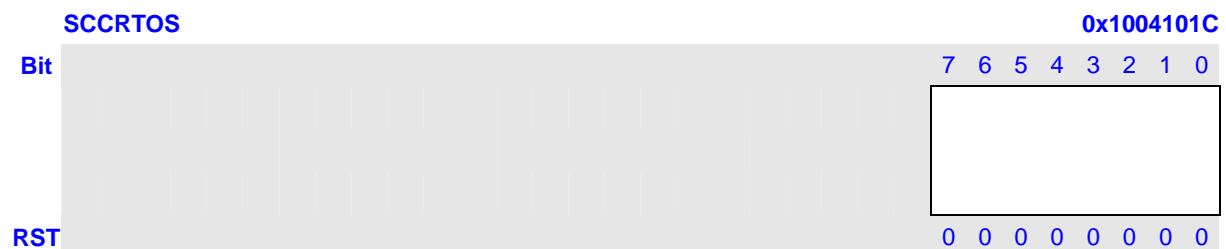
Bits	Name	Description	RW
7:0	EGTR	The register is corresponding with N value of ISO7816-3. Which indicates the extra-guard time of a transmission.	RW

### 1.3.7 ETU Counter Value Register (SCCECR)



Bits	Name	Description	RW
31:20	Reserved		R
19:0	EC	ETU counter. Write operation will clear the internal counter automatically.	RW

### 1.3.8 Reception Timeout Register (SCCRTOR)



Bits	Name	Description	RW
7:0	RTO	Retry times when parity error detected.	RW

# 1 I2C Bus Interface

## 1.1 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.

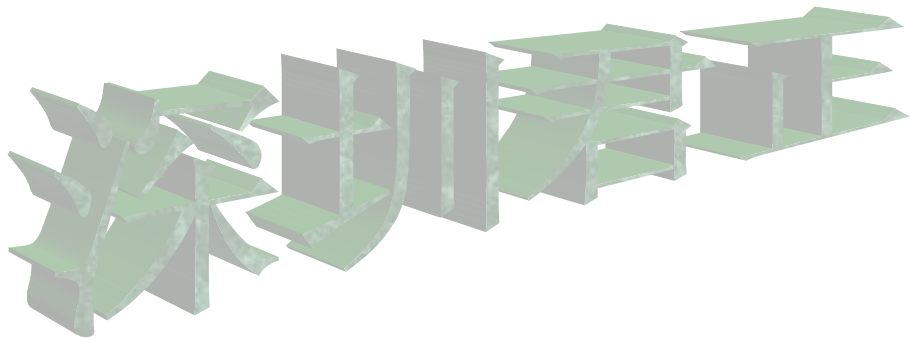
The I2C has the following features:

- 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.

1.2 Pin Description

Table 1-1 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.

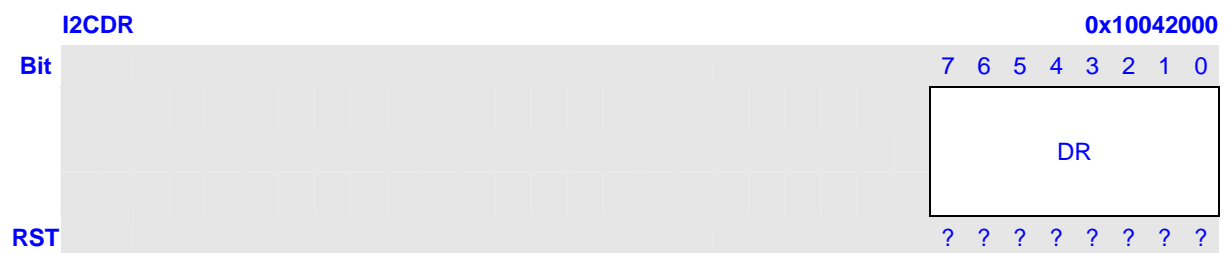


### 1.3 Register Description

Table 1-2 I2C Registers Description

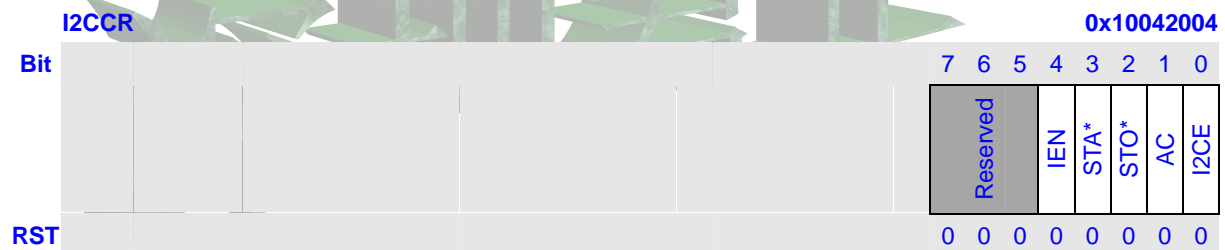
Name	RW	Reset Value	Address	Access Size
I2CDR	RW	0x??	0x10042000	8
I2CCR	RW	0x00	0x10042004	8
I2CSR	RW	0x04	0x10042008	8
I2CGR	RW	0x0000	0x1004200C	16

#### 1.3.1 Data Register (I2CDR)



Bits	Name	Description	RW
7:0	DR	Data port of HW FIFO.	RW

#### 1.3.2 Control Register (I2CCR)



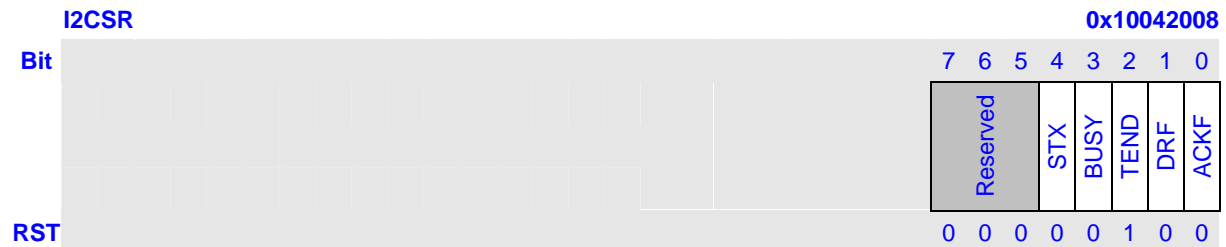
**\*Note:** STA and STO can only be written with 1.

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 <sup>2</sup> C bus. 1 – START condition will be sent to I <sup>2</sup> C bus.	RW
2	STO	I2C STOP bit. 0 – STOP condition won't be sent to I <sup>2</sup> C bus. 1 – STOP condition will be sent to I <sup>2</sup> C bus.	RW
1	AC	I2C Acknowledge Control Bit. 0 – will be sent to I <sup>2</sup> C bus as LOW level acknowledge signal. 1 – will be sent to I <sup>2</sup> C bus as HIGH level	RW

## I2C Bus Interface

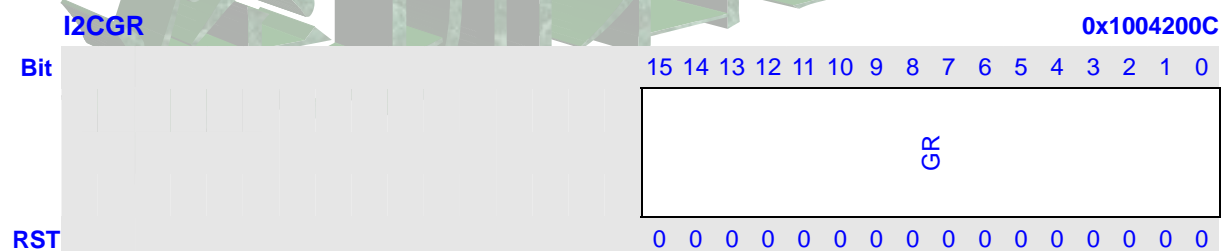
		acknowledge signal.	
0	I2CE	Enable of I2C. 0 – I2C module is disabled. 1 – I2C module is enabled.	RW

### 1.3.3 Status Register (I2CSR)



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 <sup>2</sup> C-bus is “0”. 1 – The acknowledge signal from I <sup>2</sup> C-bus is “1”.	R

### 1.3.4 Clock Generator Register (I2CGR)



Bits	Name	Description	RW
15:01	GR	Sets the frequency of serial clock. The serial clocks frequency is calculated as follows: $[\text{Value of I2CGR}] = [\text{Frequency of Device\_clock}] / (16 * [\text{SCL clock rate}]) - 1$	RW

**Note:** To make the I2C operate normally, frequency of PCLK (APB-bus clock) should not lower than transfer 2 \* [byte rate].

## 1.4 I<sup>2</sup>C-Bus Protocol

### 1.4.1 Bit Transfer

Due to the variety of different technology devices (CMOS, NMOS, bipolar) which can be connected to the I<sup>2</sup>C-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.

### 1.4.2 Data Validity

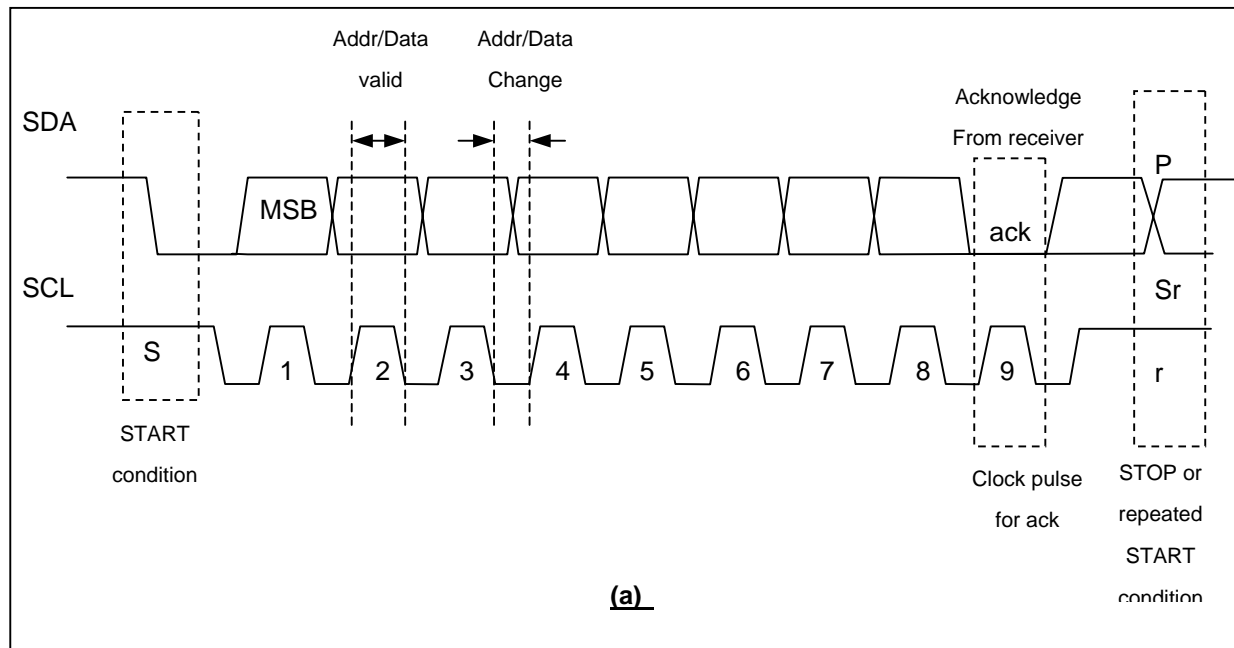
The data on the SDA line must be stable during the HIGH period of the clock. The HIGH or LOW states of the data line can only change when the clock signal on the SCL line is LOW.

### 1.4.3 START and STOP Conditions

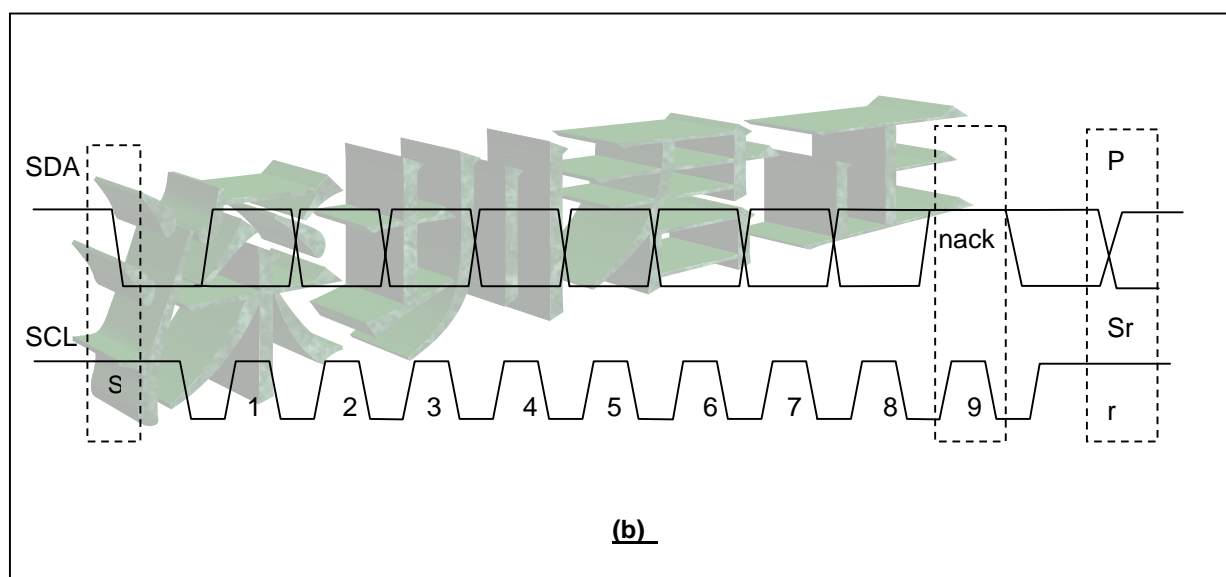
A HIGH to LOW transition on the SDA line while SCL is HIGH indicates a START condition. A LOW to HIGH transition on the SDA line while SCL is HIGH defines a STOP condition.

### 1.4.4 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 1-1 I2C-bus Protocol**



**Figure 1-2 I<sup>2</sup>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.



### 1.4.5 Data Transfer Format

#### 1.4.5.1 First Byte

The first byte is a term indicates the address byte after START condition.

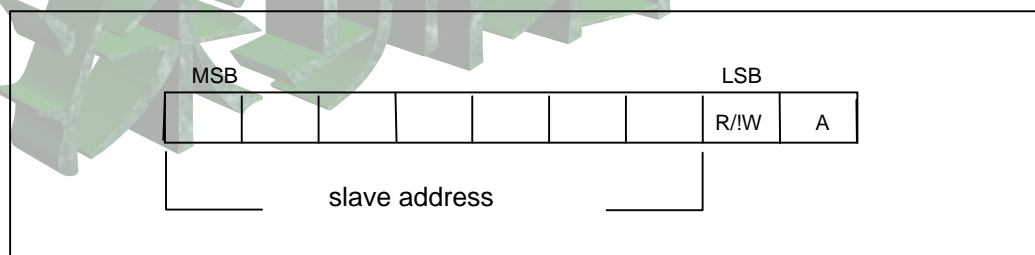
##### 1) Normal 7-bit Address:

After the START condition, the addressing procedure for the I<sup>2</sup>C-bus is such that the first byte usually determines which slave will be 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<sup>2</sup>C-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.



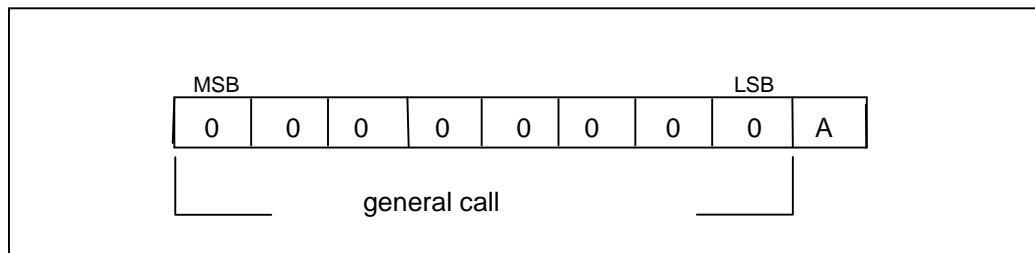
**Figure 1-3 Normal 7 Bit Address after START Condition**

##### 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 1-4 General Call Address after START Condition**

### 3) START Byte Address:

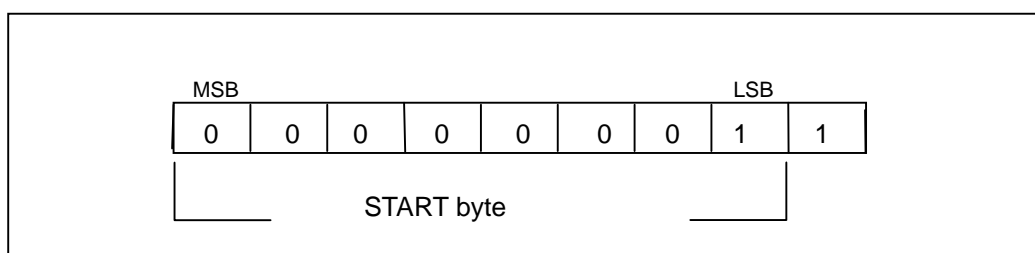
START Byte:

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 START condition (S)
- A START byte (00000001)
- An acknowledge clock pulse (ACK)\*
- 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 1-5 START Byte after START Condition**

### 1.4.5.2 Transfer Format

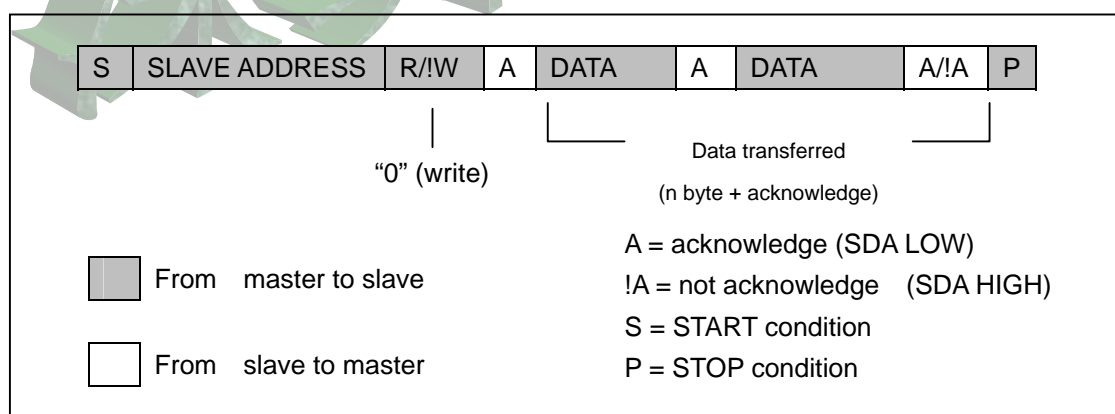
A data transfer is always terminated by a STOP condition (P) generated by the master. However, if a master still wishes to communicate on the bus, it can generate a repeated START condition (Sr) and address another slave without first generating a STOP condition. Various combinations of read/write formats are then possible within such a transfer.

**Possible data transfer formats are:**

- Master-transmitter transmits to slave-receiver. The transfer direction is not changed.
- 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.
- 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 1-6 A Master-Transmitter Addresses a Slave Receiver with a 7-Bit Address**

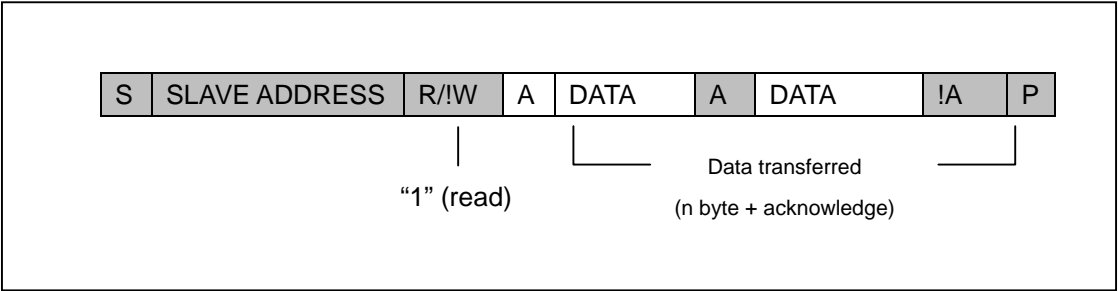
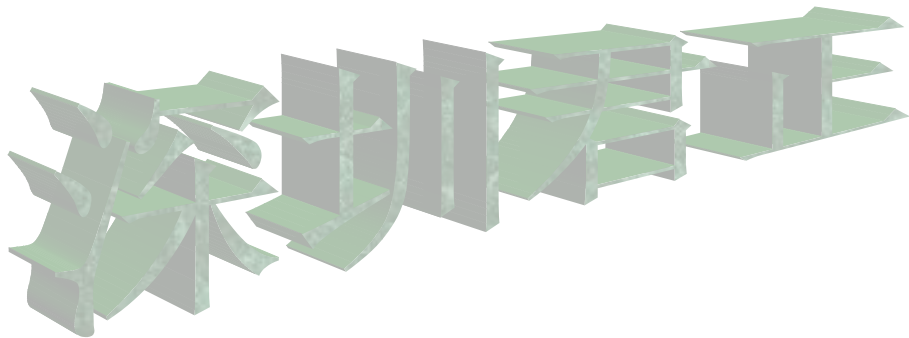


Figure 1-7 A Master Reads the Slave Immediately after the First Byte (Master-Receiver)



## 1.5 I2C Operation

### 1.5.1 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.

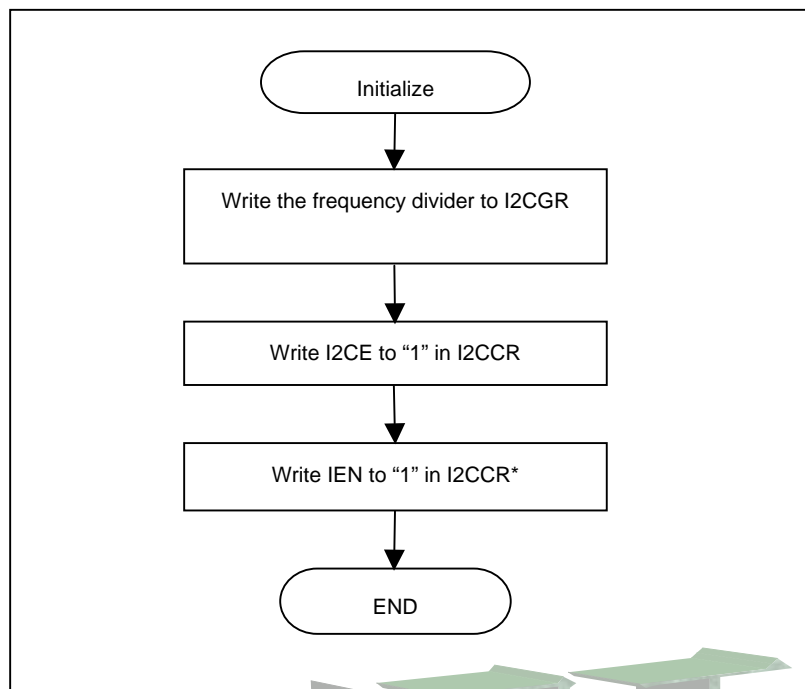


Figure 1-8 I2C Initialization

**Note:** This step is selectable.

### 1.5.2 Write Operation

Following figure illustrates the flow of a write operation.

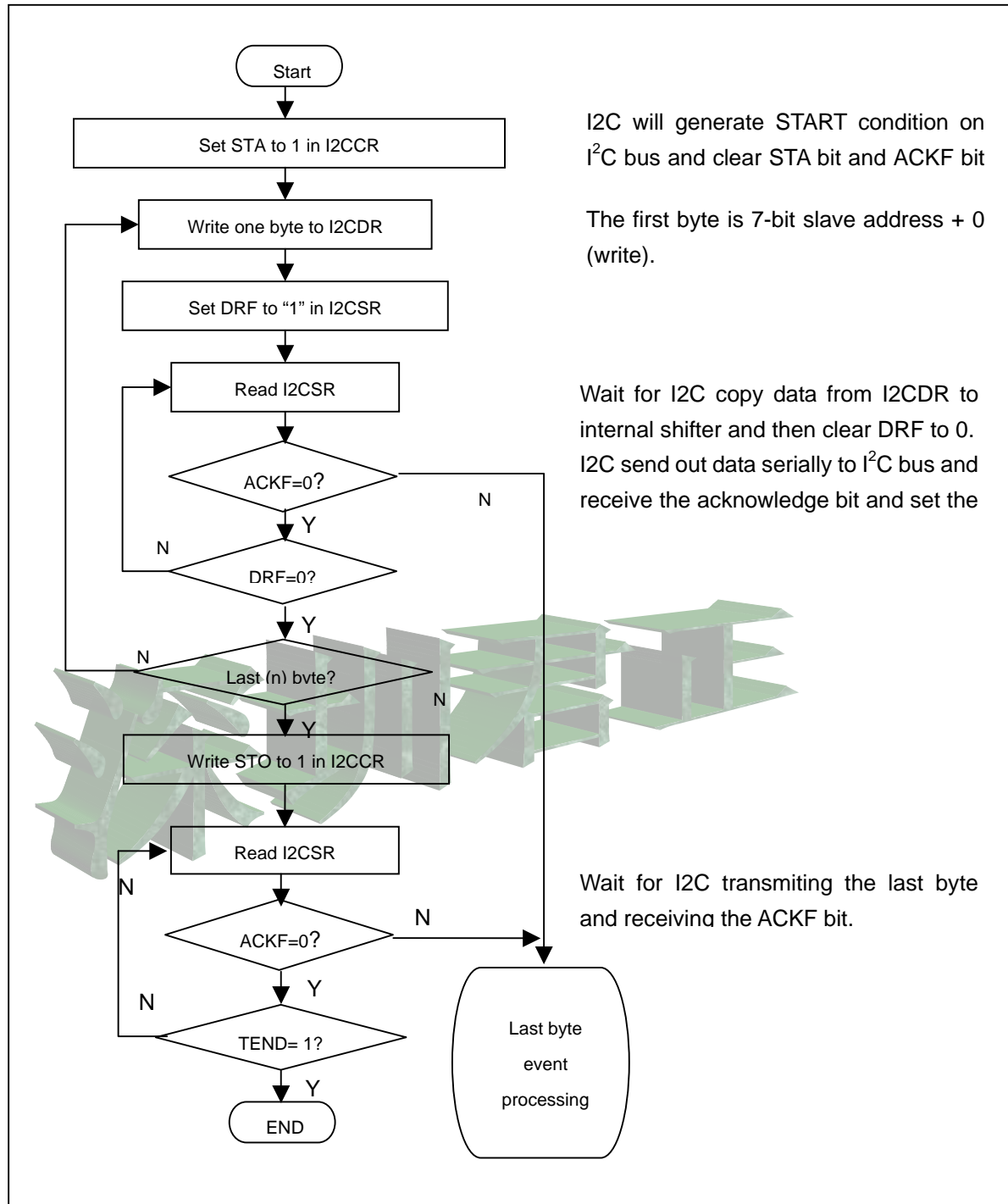


Figure 1-9 I2C Write Operation Flowchart

### 1.5.3 Read Operation

Following figure illustrates the flow of read operation.

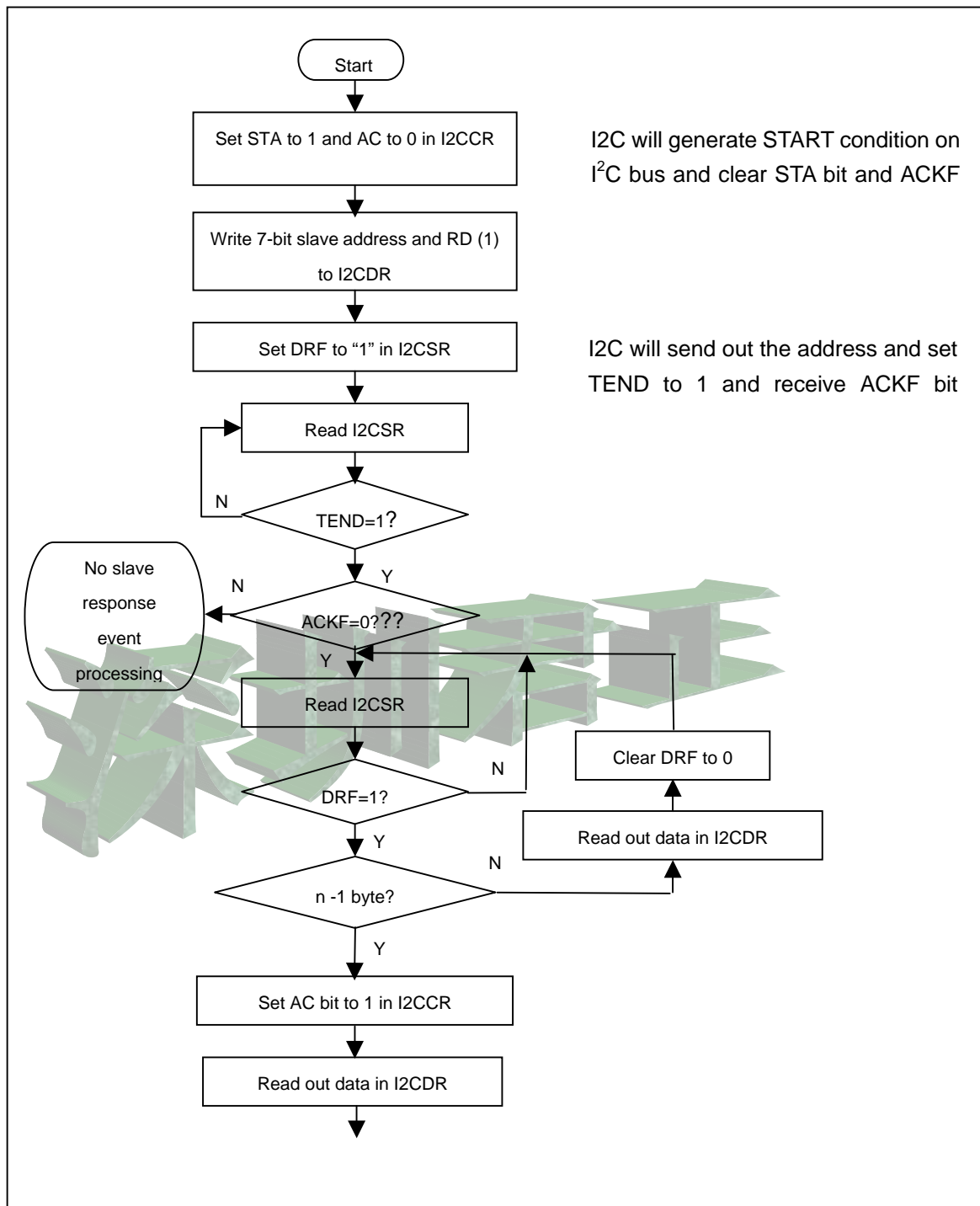


Figure 1-10 I2C Read Operation Flowchart

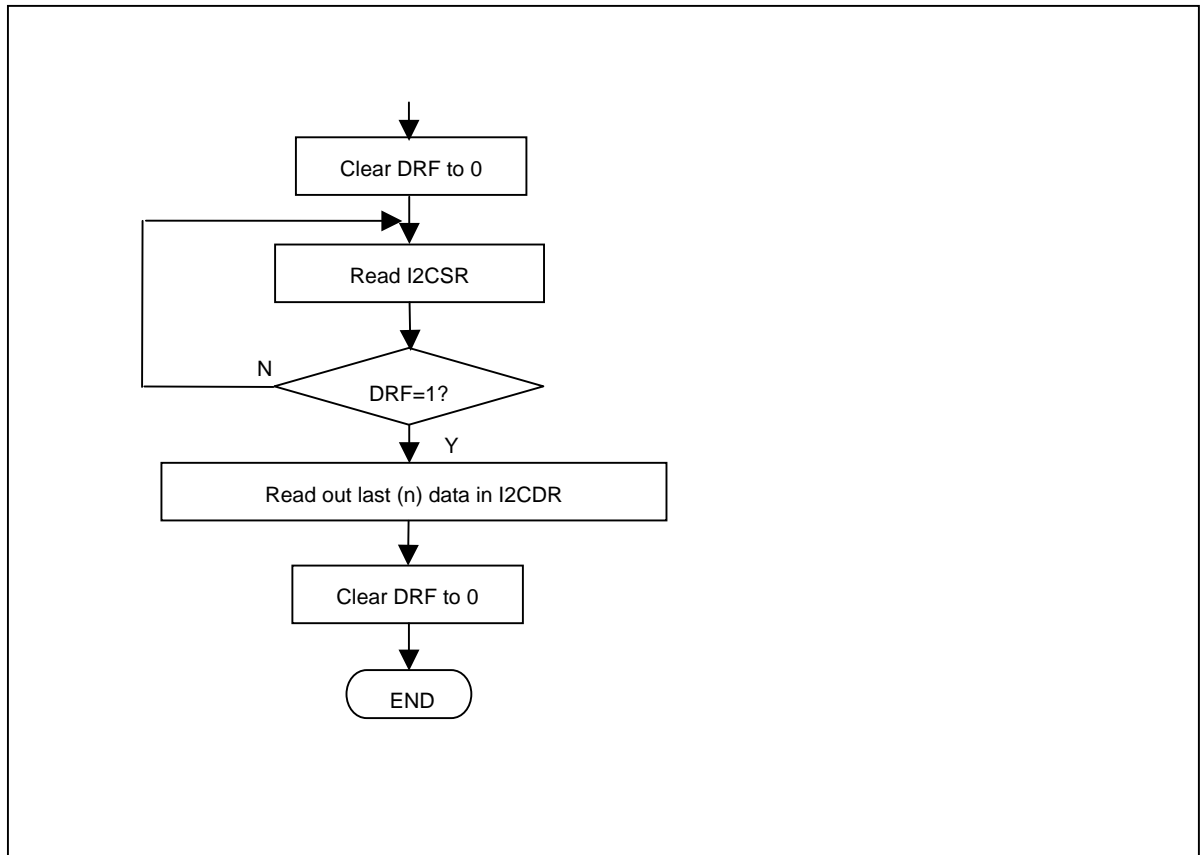
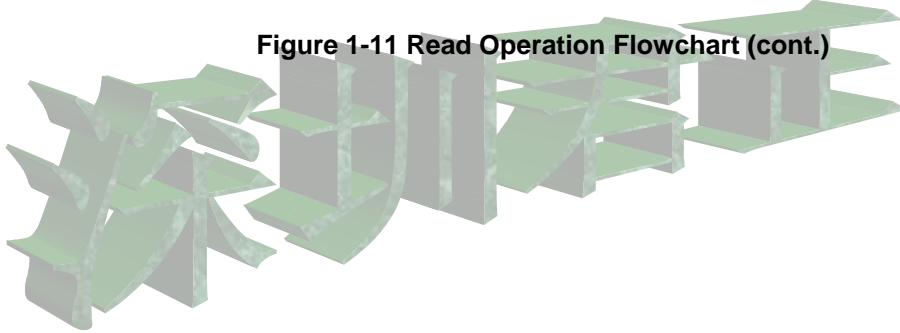


Figure 1-11 Read Operation Flowchart (cont.)





# 1 Synchronous Serial Interface

## 1.1 Overview

The SSI is a full-duplex synchronous serial interface and can connect to a variety of external analog-to-digital (A/D) converters, audio and telecom codecs, and other devices that use serial protocols for transferring data. The SSI supports National's Microwire, Texas Instruments Synchronous Serial Protocol (SSP), and Motorola's Serial Peripheral Interface (SPI) protocol.

The SSI operates in master mode (the attached peripheral functions as a slave) and supports serial bit rates from 7.2 KHz to 24 MHz. Serial data formats may range from 2 to 17 bits in length. The SSI provides 16 entries deep x 17 bits wide transmit and receive data FIFOs.

The FIFOs may be loaded or emptied by the Central Processor Unit (CPU) using programmed I/O, or DMA transfers while receiving or transmitting.

Features:

- 3 protocols support: National's Microwire, TI's SSP, and Motorola's SPI
- Full-duplex or transmit-only or receive-only operation
- Programmable transfer order: MSB first or LSB first
- 16 entries deep x 17 bits wide transmit and receive data FIFOs
- Configurable normal transfer mode or Interval transfer mode
- Programmable clock phase and polarity for Motorola's SSI format
- Two slave select signal (SSI\_CE\_ / SSI\_CE2\_) supporting up to 2 slave devices
- Back-to-back character transmission/reception mode
- Loop back mode for testing

## 1.2 Pin Description

**Table 1-1 Micro Printer Controller Pins Description**

<b>Name</b>	<b>I/O</b>	<b>Description</b>
SSI_CLK	Output	Serial bit-rate clock
SSI_CE_	Output	First slave select enable
SSI_CE2_ / SSI_GPC	Output	Second slave select enable / General purpose control signal to external chip
SSI_DT	Output	Transmit data (serial data out)
SSI_DR	Input	Receive data (serial data in)

SSI\_CLK is the bit-rate clock driven from the SSI to the peripheral. SSI\_CLK is toggled only when data is actively being transmitted and received.

SSI\_CE\_ or SSI\_CE2\_ are the framing signal, indicating the beginning and the end of a serialized data word.

SSI\_DT and SSI\_DR are the Transmit and Receive serial data lines.

SSI\_GPC is general-purpose control signal, synchronized with SSI\_CLK, can be used for LCD control.

When the multiplexed pin is configured as SSI\_GPC pin, SSI can't be configured for 17-bit (or multiples of it) data transfer. And the SSI can only perform transfer with the only slave select SPI\_CE\_.

SSI\_GPC and SSI\_CE2\_ is a multiplexed pin.

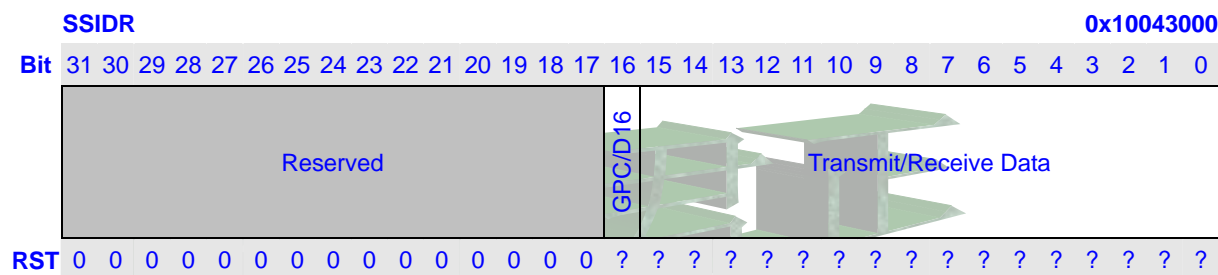
## 1.3 Register Description

The SSI has seven registers: one data, two control, one status, one bit-rate control, and two interval control registers. The table list these registers.

**Table 1-2 SSI Serial Port Registers**

Name	RW	Reset Value	Address	Access Size
SSIDR	RW	0x??	0x10043000	32
SSICR0	RW	0x0000	0x10043004	16
SSICR1	RW	0x00007060	0x10043008	32
SSISR	RW	0x00000098	0x1004300C	32
SSIITR	RW	0x0000	0x10043010	16
SSIICR	RW	0x00	0x10043014	8
SSIGR	RW	0x0000	0x10043018	16

### 1.3.1 SSI Data Register (SSIDR)

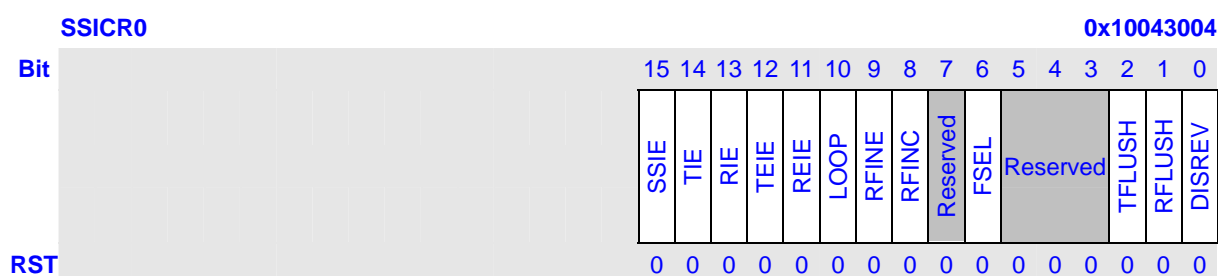


Bits	Name	Description	RW
31:17	Reserved		R
16	GPC/D16	This bit can be used as normal data bus bit 16 or GPC bit alternatively. When the multiplexed output pin is selected as SSI_CE2_, it is normal data bus bit and it's readable / writable; when multiplexed pin is selected as SSI_GPC, it is GPC bit for SSI_GPC pin output and it's write-only	RW
15:0	Transmit/Receive Data	Data word to be written to/read from Transmit/Receive FIFO. When the transfer frame length is less than 17-bit, received data is automatically right justified in the receive-FIFO and the upper unused bits are filled with '0'. For transmission, the upper unused bits of the data written into SSIDR is ignored by the transmit logic. (Note: "upper unused bits" does not include the SSIDR.GPC bit.	RW

## Synchronous Serial Interface

		<p>National microwire format includes format 1 and format2, when national microwire format 2 is selected, Bit 16 of SSIDR is defined as read/write operation judge bit, if it is 0, bit 15~0 represent one read command; if it is 1, bit 15~0 represent one write command and following is the written data. So the maximum length of one command (is defined in MCOM) is 16, the maximum length of one written or read data (is defined in FLEN) can be 17.</p> <p>Transmit-FIFO only contain one read operation command once, or one write operation command and its data once, after transmit-FIFO is empty, next command can be filled in transmit-FIFO.</p>	
--	--	--	--

### 1.3.2 SSI Control Register0 (SSICR0)



Bits	Name	Description	RW
15	SSIE	This bit is used to enable/disable SSI module: 0 – disable; 1 – enable Clearing SSIE will not reset SSI FIFO, SSICR0, SSICR1, SSIGR, SSIITR and SSIICR automatically. Software should ensure the FIFOs/registers are properly configured and be flush/reset manually when necessary before enabling SSI.	RW
14	TIE	This bit enables/disables the transmit-FIFO half-empty interrupt TXI: 0 – disable; 1 – enable	RW
13	RIE	This bit enables/disables the receive-FIFO half-full interrupt RXI: 0 – disable; 1 – enable	RW
12	TEIE	This bit enables/disables the transmit-error interrupt TEI: 0 – disable; 1 – enable	RW
11	REIE	This bit enables/disables the receive-error interrupt REI: 0 – disable; 1 – enable	RW
10	LOOP	Used for test purpose. In loop mode, the output of SSI transmit shift register is connected to input of SSI receive shift register internally. The data received should be the same as the data transmitted. And do not output any valid signals on the pins. 0 – normal SSI mode; 1 – LOOP mode	RW

9	RFINE	This bit enables/disables receive finish control function: 0 – disable; 1 – enable. For SSICR1.FMAT = B'10 (National Microwire format 1 is selected), SSICR0.RFINE must be 0	The receive finish condition list below:			RW
			RFINE	RFINC	Receive Finish Condition	
			0	x	Same as transmit completion condition (transmit-fifo is empty and SSICR1.UNFIN = 0)	
			1	0	Receive continue	
8	RFINC*	Receive finish control bit: 0 – receive continue; 1 – receive finished	1	1	Receive finish	RW
7	Reserved					R
6	FSEL	This bit sets the frame signal to be used for slave select. The unselected frame signal always output invalid level. When multiplexed pin is used as SSI_GPC, only 0 can be set. 0 – SSI_CE_ is selected; 1 – SSI_CE2_ is selected.				RW
5:3	Reserved					R
2	TFLUSH	Flush the transmit FIFO when set to 1. Always return 0 when read.				RW
1	RFLUSH	Flush the receive FIFO when set to 1. Always return 0 when read.				RW
0	DISREV	This bit enables/disables receive function: 0 – enable; 1 – disable				RW

**Note:** \*: 1) When transmitting finished or for receive-only operation, transmit function can be disabled and this bit is used to control receiving completion, and the SSI will consume less power.

2) When the finish condition is set, the receiving will complete after present character is completely shifted in, then the SSI will stop the SSI\_CLK and negate the SSI\_CE\_ / SSI\_CE2\_ if necessary. To make sure present transfer is completed, user must read and get SSISR.END = 1 (or SSISR.BUSY = 0).

### 1.3.3 SSI Control Register1 (SSICR1)

SSICR1																			0x10043008													
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
	FRMHL		TFVCK		TCKFI		LFST	ITFRM	UNFIN	MULTS	FMAT						MCOM			TTRG		RTRG	FLEN							PHA		POL
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	0	0	0	0	0

Bits	Name	Description	RW															
31:30	FRMHL	Frame valid level select, FRMHL [1: 0] correspond to SSI_CE2_ and SSI_CE_ respectively.	RW															
		<table><tr><th>FRMHL[1:0]</th><th>Description</th><th></th></tr><tr><td>00</td><td>SSI_CE_ is low level valid and SSI_CE2_ is low level valid</td><td>Initial value</td></tr><tr><td>01</td><td>SSI_CE_ is high level valid and SSI_CE2_ is low level valid</td><td></td></tr><tr><td>10</td><td>SSI_CE_ is low level valid and SSI_CE2_ is high level valid</td><td></td></tr><tr><td>11</td><td>SSI_CE_ is high level valid and SSI_CE2_ is high level valid</td><td></td></tr></table>		FRMHL[1:0]	Description		00	SSI_CE_ is low level valid and SSI_CE2_ is low level valid	Initial value	01	SSI_CE_ is high level valid and SSI_CE2_ is low level valid		10	SSI_CE_ is low level valid and SSI_CE2_ is high level valid		11	SSI_CE_ is high level valid and SSI_CE2_ is high level valid	
		FRMHL[1:0]		Description														
		00		SSI_CE_ is low level valid and SSI_CE2_ is low level valid	Initial value													
		01		SSI_CE_ is high level valid and SSI_CE2_ is low level valid														
		10		SSI_CE_ is low level valid and SSI_CE2_ is high level valid														
11	SSI_CE_ is high level valid and SSI_CE2_ is high level valid																	
29:28	TFVCK	Time from frame valid to clock start, that provide programmable time delay from frame (SSI_CE_ /SSI_CE2_) assert edge to SSI_CLK leading edge. When TFVCK = B'00, the time is fixed half SSI_CLK or one SSI_CLK cycle according to SSICR1.POL and SSICR1.PHA configuration. For SSICR1.FMAT = B'01, SSICR1.TFVCK is ignored.	RW															
		<table><tr><th>TFVCK[1:0]</th><th>Description</th><th></th></tr><tr><td>00</td><td>Ignore (default half or one SSI_CLK cycle delay time)</td><td>Initial value</td></tr><tr><td>01</td><td>1 more SSI_CLK cycle delay time is added</td><td></td></tr><tr><td>10</td><td>2 more SSI_CLK cycle delay time is added</td><td></td></tr><tr><td>11</td><td>3 more SSI_CLK cycle delay time is added</td><td></td></tr></table>		TFVCK[1:0]	Description		00	Ignore (default half or one SSI_CLK cycle delay time)	Initial value	01	1 more SSI_CLK cycle delay time is added		10	2 more SSI_CLK cycle delay time is added		11	3 more SSI_CLK cycle delay time is added	
		TFVCK[1:0]		Description														
		00		Ignore (default half or one SSI_CLK cycle delay time)	Initial value													
		01		1 more SSI_CLK cycle delay time is added														
		10		2 more SSI_CLK cycle delay time is added														
11	3 more SSI_CLK cycle delay time is added																	
27:26	TCKFI	Time from clock stop to frame invalid, provide programmable time delay from SSI_CLK last edge to frame (SSI_CE_ /SSI_CE2_) negage edge. When TCKFI = B'00, the time is fixed one SSI_CLK or half SSI_CLK cycle according to SSICR1.POL and SSICR1.PHA configuration. For SSICR1.FMAT = B'01, SSICR1.TFVCK is ignored.	RW															
		<table><tr><th>TCKFI[1:0]</th><th>Description</th><th></th></tr><tr><td>00</td><td>Ignore (default half or one SSI_CLK cycle delay time)</td><td>Initial value</td></tr><tr><td>01</td><td>1 more SSI_CLK cycle delay time is added</td><td></td></tr><tr><td>10</td><td>2 more SSI_CLK cycle delay time is added</td><td></td></tr></table>		TCKFI[1:0]	Description		00	Ignore (default half or one SSI_CLK cycle delay time)	Initial value	01	1 more SSI_CLK cycle delay time is added		10	2 more SSI_CLK cycle delay time is added				
		TCKFI[1:0]		Description														
		00		Ignore (default half or one SSI_CLK cycle delay time)	Initial value													
		01		1 more SSI_CLK cycle delay time is added														
		10		2 more SSI_CLK cycle delay time is added														

		11	3 more SSI_CLK cycle delay time is added																	
25	LFST	Set to LSB first or MSB first when transfer: 0 – MSB first; 1 – LSB first			RW															
24	ITFRM	Frame during interval, selects if the Frame (SSI_CE_ /SSI_CE2_) signal is negated or not during interval time at Interval Mode (SSICR1.FMAT = B'00 and SSIITR.IVLTM ≠ H'0000). It's ignored at Normal Mode. 0 – SSI_CE_ /SSI_CE2_ deassert during interval time at Interval Mode 1 – SSI_CE_ /SSI_CE2_ keeps asserted during interval time at Interval Mode			RW															
23	UNFIN	This bit controls whether the SSI finishes transmission or wait for data filling (underrun happen) after all data in transmit-FIFO are sent out during transfer. This bit must be cleared to 0 when SSICR1.FMAT = B'01 (TI's SSP format). 0 – Transmit-FIFO empty means end of transmission; 1 – Transmission didn't finish when transmit-FIFO is empty, SSI underrun error would occur and SSI waits for data filling; SSI_CLK and SSI_CE_ /SSI_CE2_ keeps asserted, SSI_CLK stop at the current level.  <b>Note:</b> For transmit-FIFO empty before any transfer after SSI enabled, if SSICR1.UNFIN = 1 or SSICR0.RFINE = 0, SSI will wait till transmit-FIFO isn't empty then start to transfer and no underrun error will occur; if SSICR1.UNFIN = 0 and SSICR0.RFINE = 1, after transmit-FIFO become empty, SSI will start a receive-only transfer.			RW															
22	MULTS	This bit set the multiplexed pin function. 0 – Multiplexed pin is used as SSI_CE2_; 1 – Multiplexed pin is used as SSI_GPC.			RW															
21:20	FMAT	These bits set the operating transfer format. <table><tr><th>FMAT[1:0]</th><th>Description</th><th></th></tr><tr><td>00</td><td>Motorola's SPI format</td><td>Initial value</td></tr><tr><td>01</td><td>TI's SSP format</td><td></td></tr><tr><td>10</td><td>National Microwire 1 format</td><td></td></tr><tr><td>11</td><td>National Micowire 2 format</td><td></td></tr></table>			FMAT[1:0]	Description		00	Motorola's SPI format	Initial value	01	TI's SSP format		10	National Microwire 1 format		11	National Micowire 2 format		RW
FMAT[1:0]	Description																			
00	Motorola's SPI format	Initial value																		
01	TI's SSP format																			
10	National Microwire 1 format																			
11	National Micowire 2 format																			
19:16	Reserved				R															
15:12	MCOM	When SSICR1.FMAT = B'10 or B'11 (National Microwire format 1 or 2 is selected), this bit decides the length of command from 1-bit to 16-bit. The length of written or read data is defined in FLEN. For SSICR1.FMAT ≠ B'10 or B'11, this bit is ignored. <table><tr><th>MCOM[1:0]</th><th>Description</th><th></th></tr><tr><td>0000</td><td>1-bit command selected</td><td></td></tr><tr><td>0001</td><td>2-bit command selected</td><td></td></tr><tr><td>0010</td><td>3-bit command selected</td><td></td></tr></table>			MCOM[1:0]	Description		0000	1-bit command selected		0001	2-bit command selected		0010	3-bit command selected		RW			
MCOM[1:0]	Description																			
0000	1-bit command selected																			
0001	2-bit command selected																			
0010	3-bit command selected																			

		0010	3-bit command selected		
		0011	4-bit command selected		
		0100	5-bit command selected		
		0101	6-bit command selected		
		0110	7-bit command selected		
		0111	8-bit command selected	Initial value	
		1000	9-bit command selected		
		1001	10-bit command selected		
		1010	11-bit command selected		
		1011	12-bit command selected		
		1100	13-bit command selected		
		1101	14-bit command selected		
		1110	15-bit command selected		
		1111	16-bit command selected		
11:10	TTRG	These bits set the transmit-FIFO half-empty threshold value, when equal or less characters left in transmit-FIFO, the SSISR.TFHE will be set to '1'.			RW
		<b>TTRG[1:0]</b>	<b>Description</b>		
		00	less than or equal to 1		
		01	less than or equal to 4		
		10	less than or equal to 8	Initial value	
		11	less than or equal to 14		
9:8	RTRG	Set the receive-FIFO half-full threshold value, when equal or more characters received in receive-FIFO, the SSISR.RFHF will be set to '1'.			RW
		<b>RTRG[1:0]</b>	<b>Description</b>		
		00	less than or equal to 1		
		01	less than or equal to 4	Initial value	
		10	less than or equal to 8		
		11	less than or equal to 14		
7:4	FLEN	These bits set the bit length of every character to be transmitted/received. The maximum data length can be configured is 17 bits. For data length longer than 17 bits (multiples of the SSICR1.FLEN configured length), the software should ensure properly processing. When SSI_GPC pin is used (SSICR1.MULTS = 1), the FLEN shouldn't be configured as B'1111 (17-bit data). When TI SSP mode is selected (FMAT = 2'b01), 2-bit data length (FLEN = 4'b0000) isn't supported.			RW
		<b>MCOM[1:0]</b>	<b>Description</b>		
		0000	2-bit data		
		0001	3-bit data		
		0010	4-bit data		
		0011	5-bit data		



		0011	5-bit data		
		0100	6-bit data		
		0101	7-bit data		
		0110	8-bit data	Initial value	
		0111	9-bit data		
		1000	10-bit data		
		1001	11-bit data		
		1010	12-bit data		
		1011	13-bit data		
		1100	14-bit data		
		1101	15-bit data		
		1110	16-bit data		
		1111	17-bit data		
3:2	Reserved				R
1	PHA	This bit sets the phase of the SSI_CLK from the beginning of a data frame for Motorola's SPI format (SSICR1.FMAT = B'00). 0 – The leading edge of SSI_CLK is used to sample data from SSI_DR after the SSI_CE_ /SSI_CE2_ goes valid, it is initial value; 1 – The leading edge of SSI_CLK is used to drive data onto SSI_DT after the SSI_CE_ /SSI_CE2_ goes valid.			RW
0	POL	This bit sets SSI_CLK's idle state polarity for Motorola's SPI format (SSICR1.FMAT = B'00). 0 – SSI_CLK keeps low level when idle, when SSI_CE_ /SSI_CE2_ goes valid the leading clock edge is a rising edge, it is initial value; 1 – SSI_CLK keeps high level when idle, when SSI_CE_ /SSI_CE2_ goes valid the leading clock edge is a falling edge.			RW

### 1.3.4 SSI Status Register1 (SSISR)

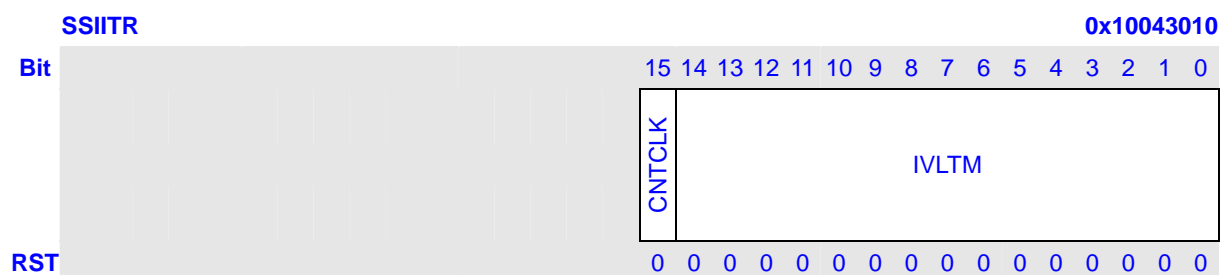
SSISR														0x1004300C																		
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															TFIFO-NUM				RFIFO-NUM				END	BUSY	TFF	RFE	TFHE	RFHF	UNDR	OVER		
RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	

Bits	Name	Description	RW
31:18	Reserved		R

## Synchronous Serial Interface

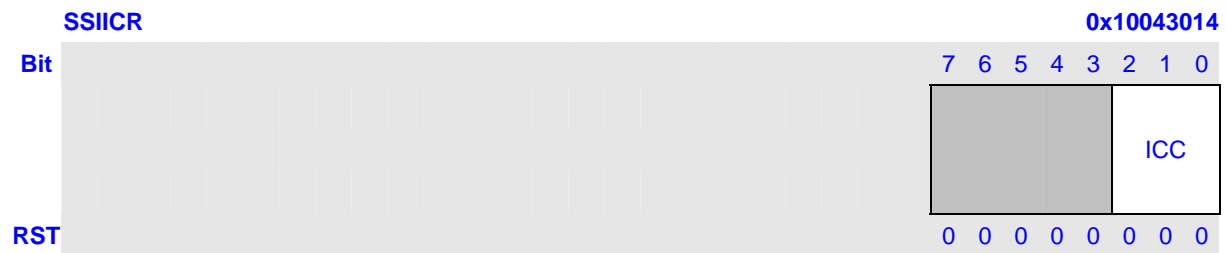
17:13	TFIFO-NUM	These bits indicate the Characters Number in Transmit-FIFO.	R
12:8	RFIFO-NUM	These bits indicate the Characters Number in Transmit-FIFO.	R
7	END	This bit indicates transfer end status. It is the inverse of SSISR.BUSY when transfer is in process, but it'll keep cleared at interval time before transfer is completed. It'll be set when transfer finished.	R
6	BUSY	This bit indicates SSI's working status. 0 – SSI is idle or at interval time; 1 – Transmission and/or reception is in process.	R
5	TFF	This bit denotes transmit-FIFO is full or not. 0 – Transmit-FIFO is not full; 1 – Transmit-FIFO is full.	R
4	RFE	This bit denotes receive-FIFO is empty or not. 0 – Receive-FIFO is not empty; 1 – Receive-FIFO is empty.	R
3	TFHE	This bit denotes whether the characters number in transmit-FIFO being less or equal to SSICR1.TTRG. 0 – The data in transmit-FIFO is more than the condition set by SSICR1.TTRG; 1 – The data in transmit-FIFO meets the condition set by SSICR1.TTRG, If SSICR0.TIE = 1, it will generate SSI TXI interrupt.	R
2	RFHF	This bit denotes whether the characters number in receive-FIFO being more or equal to the number set by SSICR1.RTRG. 0 – The data in receive-FIFO is less than the condition set by SSICR1.RTRG 1 – The data in receive-FIFO meets the condition set by SSICR1.RTRG, If SSICR0.RIE = 1, it will generate SSI RXI interrupt.	R
1	UNDR	Transmit-FIFO underrun status. When underrun happens, SSI set this bit and keeps the current status of SSI_CLK and SSI_CE_/SSI_CE2_, waiting for transmit-FIFO filling. 0 – Underrun has not occurred; 1 – Underrun has occurred, when SSICR0.TEIE is set, it will generate SSI TEI interrupt. Write '0' to clear this bit, writing '1' has no effect.	RW
0	OVER	Receive-FIFO overrun status, new received data will lose. 0 – Overrun has not occurred; 1 – Overrun has occurred, When SSICR0.REIE is set, it will generate SSI REI interrupt. Write '0' to clear this bit, writing '1' has no effect.	RW

## 1.3.5 SSI Interval Time Control Register (SSIITR)



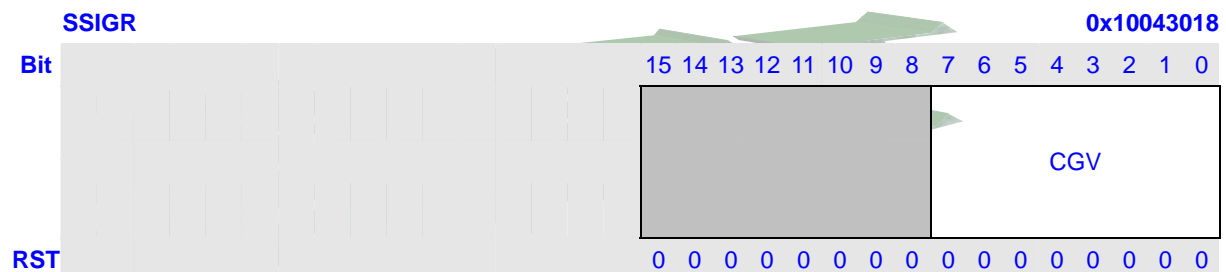
Bits	Name	Description	RW
15	CNTCLK	Counting clock source select. 0 – Use SSI bit clock (SSI_CLK) as the interval counter clock source; 1 – Use 32K clock as the interval counter clock source.	RW
14:0	IVLTM	Interval time set, set the cycle number of counting clock source for desired interval time. When SSIITR.IVLTM = 0x0000, normal mode is selected, and SSIITR.CNTCLK and SSIICR are ignored. When SSIITR.IVLTM ≠ 0x0000, interval mode is selected. The interval time is calculated as follows: $\text{Interval time} \approx [\text{CNTCLK clock period}] * [\text{Value of IVLTM}]$ The actual interval time is as follow: When SSIITR.CNTCLK = 0: $\text{Interval time} = [\text{CNTCLK clock period}] * [\text{Value of IVLTM}] + 3 * \text{device\_clock period}$ When SSIITR.CNTCLK = 1: $\text{Interval time} \geq [\text{CNTCLK clock period}] * [\text{Value of IVLTM} + 1] + 1 * \text{device\_clock period};$ $\text{Interval time} \leq [\text{CNTCLK clock period}] * [\text{Value of IVLTM} + 2] + 2 * \text{device\_clock period}$	RW

## 1.3.6 SSI Interval Character-per-frame Control Register (SSIICR)



Bits	Name	Description	RW
7:3	Reserved		R
2:0	ICC	Sets the fixed number of characters to be transmitted / received each time during SSI_CLK changing (and SSI_CE_ / SSI_CE2_ asserting) in interval mode for SSICR1.FMAT = B'00 (Motorola's SPI format is selected). SSIICR is ignored for SSICR1.FMAT ≠ B'00. The desired transfer number of characters-per-frame is (SSIICR set value + 1).	RW

## 1.3.7 SSI Clock Generator Register (SSIGR)



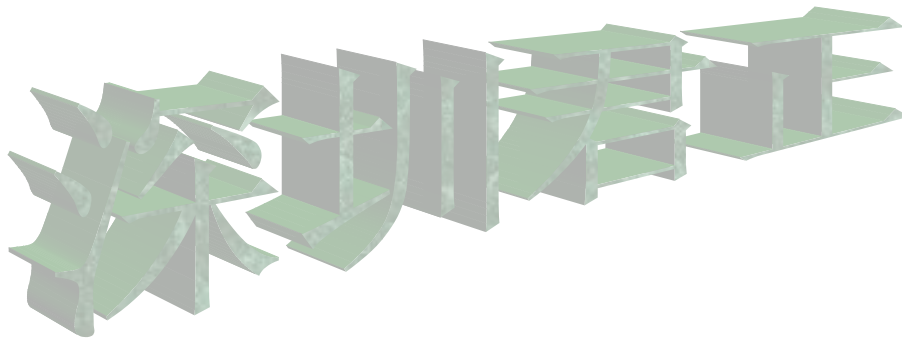
Bits	Name	Description	RW
15:8	Reserved		R
7:0	CGV	Sets the frequency of serial bit clock (SSI_CLK). The serial bit clock (SSI_CLK) is generated by dividing device-clock as follows: $F_{SSI\_CLK} = [Frequency\ of\ device\ clock] / (2 * (CGV + 1))$ Device clock is generated in CPM module. The value in SSIGR can be set from 0 to 255, and initialized to 0x0000 on power-on reset.	RW

## 1.4 Functional Description

Serial data is transferred between the processor and external peripheral through FIFO buffers in the SSI. Data transfers to system memory are handled by either the CPU (using programmed I/O) or by DMA. Operation is full duplex - separate buffers and serial data paths permit simultaneous transfers to and from the external peripheral.

Programmed I/O transmits and receives data directly between the CPU and the transmit/receive FIFO's. The DMA controller transfers data during transmit and receive operations between memory and the FIFO's.

Transmit data is written by the CPU or DMA to the SSI's transmit FIFO. The SSI then takes the data from the FIFO, serializes it, and transmits it via the SSI\_DT signal to the peripheral. Data from the peripheral is received via the SSI\_DR signal, converted to parallel words and is stored in the Receive FIFO. Read operations automatically target the receive FIFO, while write operations write data to the transmit FIFO. Both the transmit and receive FIFO buffers are 16 entries deep by 17 bits wide. As the received data fills the receive FIFO, a programmable threshold triggers an interrupt to the Interrupt Controller. If enabled, an interrupt service routine responds by identifying the source of the interrupt and then performs one or several read operations from the inbound (receive) FIFO buffer.



### 1.5 Data Formats

Four signals are used to transfer data between the processor and external peripheral. The SSI supports three formats: Motorola SPI, Texas Instruments SSP, and National Microwire. Although they have the same basic structure the three formats have significant differences, as described below.

SSI\_CE\_/SSI\_CE2\_ varies for each protocol as follows:

- For SPI and Microwire formats, SSI\_CE\_/SSI\_CE2\_ functions as a chip select to enable the external device (target of the transfer), and is held active-low during the data transfer.
- For SSP format, this signal is pulsed high for one serial bit-clock period at the start of each frame.

SSI\_CLK varies for each protocol as follows:

- For Microwire, both transmit and receive data sources switch data on the falling edge of SSI\_CLK, and sample incoming data on the rising edge.
- For SSP, transmit and receive data sources switch data on the rising edge of SSI\_CLK, and sample incoming data on the falling edge.
- For SPI, the user has the choice of which edge of SSI\_CLK to use for switching outgoing data, and for sampling incoming data. In addition, the user can move the phase of SSI\_CLK, shifting its active state one-half period earlier or later at the start and end of a frame.

While SSP and SPI are full-duplex protocols, Microwire uses a half-duplex master-slave messaging protocol. At the start of a frame, a 1 or 2-byte control message is transmitted from the controller to the peripheral. The peripheral does not send any data. The peripheral interprets the message and, if it is a READ request, responds with requested data, one clock after the last bit of the requesting message.

The serial clock (SSI\_CLK) only toggles during an active frame. At other times it is held in an inactive or idle state, as defined by its specified protocol.

#### 1.5.1 Motorola's SPI Format Details

##### 1.5.1.1 General Single Transfer Formats

The figures below show the timing of general single transfer format. SSI\_GPC is also illustrated when the multiplexed pin is selected as SSI\_GPC.

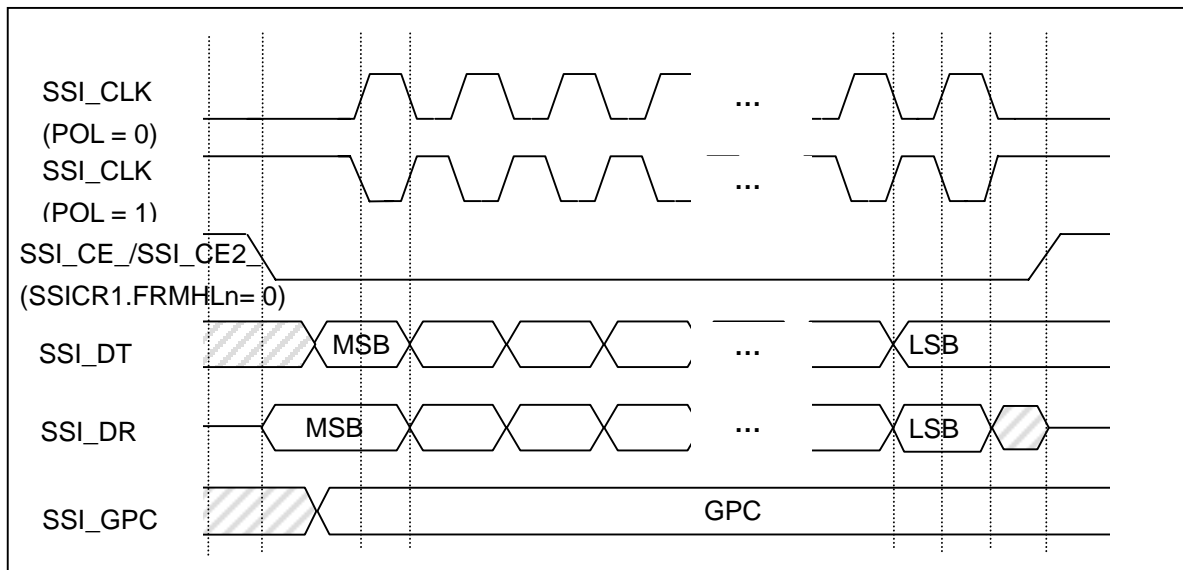


Figure 1-1 SPI Single Character Transfer Format (PHA = 0)

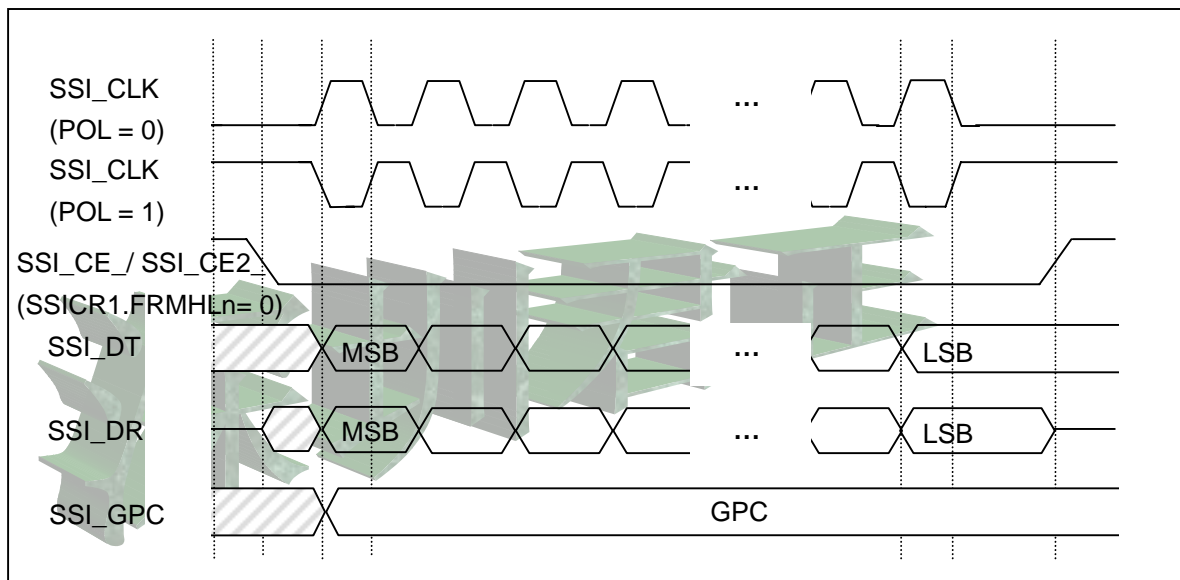


Figure 1-2 SPI Single Character Transfer Format (PHA = 1)

For SSICR1.PHA = 0, when SSICR1.TFVCK = B'00, hardware ensures the first clock edge appears one SSI\_CLK period after SSI\_CE\_ / SSI\_CE2\_ goes valid; when SSICR1.TCKFI = B'00, hardware ensures the SSI\_CE\_ / SSI\_CE2\_ negated half SSI\_CLK period after last clock change edge; when SSICR1.TFVCK ≠ B'00 or SSICR1.TCKFI ≠ B'00, 1/2/3 more clock cycles are inserted.

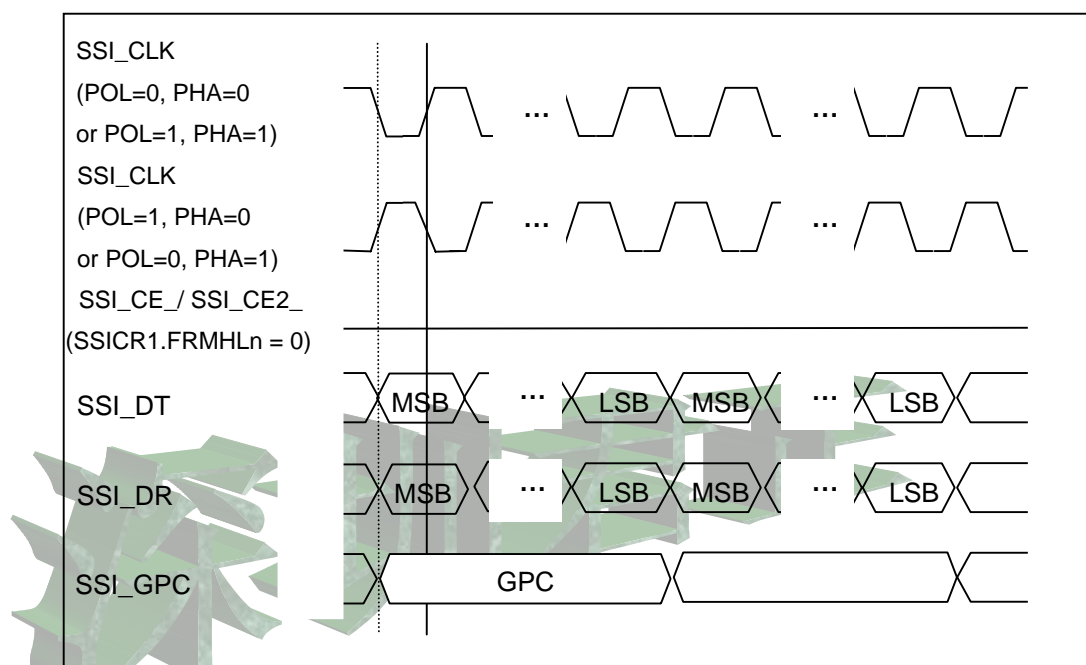
For SSICR1.PHA = 1, when SSICR1.TFVCK = B'00, hardware ensures the first clock edge appears

half SSI\_CLK period after SSI\_CE\_ / SSI\_CE2\_ goes valid; when SSICR1.TCKFI = B'00, hardware ensures the SSI\_CE\_ / SSI\_CE2\_ negated one SSI\_CLK period after last clock change edge; when SSICR1.TFVCK  $\neq$  B'00 or SSICR1.TCKFI  $\neq$  B'00, 1/2/3 more clock cycles are inserted.

Data is sampled from SSI\_DR at every rising edge (when PHA = 0, POL = 0 or PHA = 1, POL = 1) or at every falling edge (when PHA = 0, POL = 1 or PHA = 1, POL = 0). According to SPI protocol, input data on SSI\_DR should be stable at every sample clock edge.

Drive data onto SSI\_DT at every rising edge (when PHA = 0, POL = 1 or PHA = 1, POL = 0) or at every falling edge (when PHA = 0, POL = 0 or PHA = 1, POL = 1).

### 1.5.1.2 Back-to-Back Transfer Formats



**Figure 1-3 SPI Back-to-Back Transfer Format**

For Motorola's SPI format transfers those continuous characters are exchanged during SSI\_CE\_ / SSI\_CE2\_ being valid, the timing is illustrated in the figure (SSICR1.LFST = 0).

Back-to-back transfer is performed as transmit-only/full-duplex operation when transmit-FIFO is not empty before the completion of the last character's transfer or performed as receive-only operation.



### 1.5.1.3 Frame Interval Mode Transfer Format

When in interval mode (SSIITR.IVLTM  $\neq$  '0'), SSI always wait for an interval time (SSIITR.IVLTM), transfer fixed number of characters (SSIICR), then repeats the operation.

When SSICR0.RFINE = 1, if transmit-FIFO is still empty after the interval time, receive-only transfer will occur.

During interval-wait time, SSI stops SSI\_CLK, and when SSICR1.ITFRM = 0 it negates the SSI\_CE\_ / SSI\_CE2\_, when SSICR1.ITFRM = 1 it keeps asserting the SSI\_CE\_ / SSI\_CE2\_.

For transfers finished with transmit-FIFO empty, if the SSI transmit-FIFO is empty before fixed number of characters being loaded to transfer (SSICR1.UNFIN must be 1), then the SSI will set SSISR.UNDR = 1; if enabled, it'll send out a SSI underrun interrupt. At the same time, SSI will hold the SSI\_CE\_ / SSI\_CE2\_ and SSI\_CLK signals at current status and wait for the transmit-FIFO filling. The SSI will continue transfer after transmit-FIFO being filled. The SSI always stops after completion of fixed number of characters' transfer (SSICR1.UNFIN must be 0) with transmit-FIFO empty.

For transfers finished by SSICR0.RFINC being valid set, the SSI will stop after finished current character transfer and needn't wait for a whole completion of fixed number of characters' transfer.

Two Interval transfer mode are illustrated in the following figures. In these timing diagram, SSICR1.PHA = 0, SSICR1.POL = 0 and SSIICR = 0.

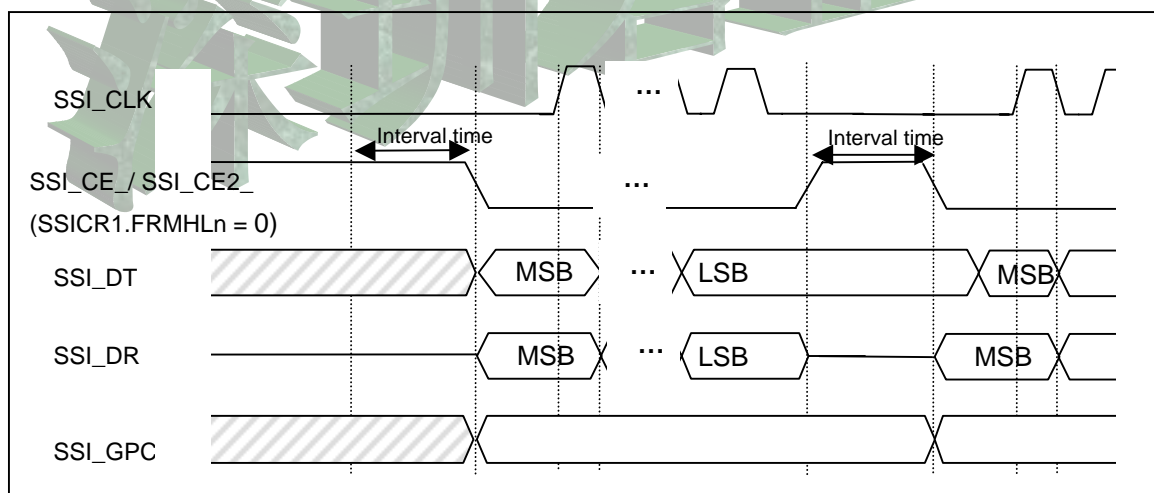


Figure 1-4 SPI Frame Interval Mode Transfer Format (ITFRM = 0, LFST = 0)

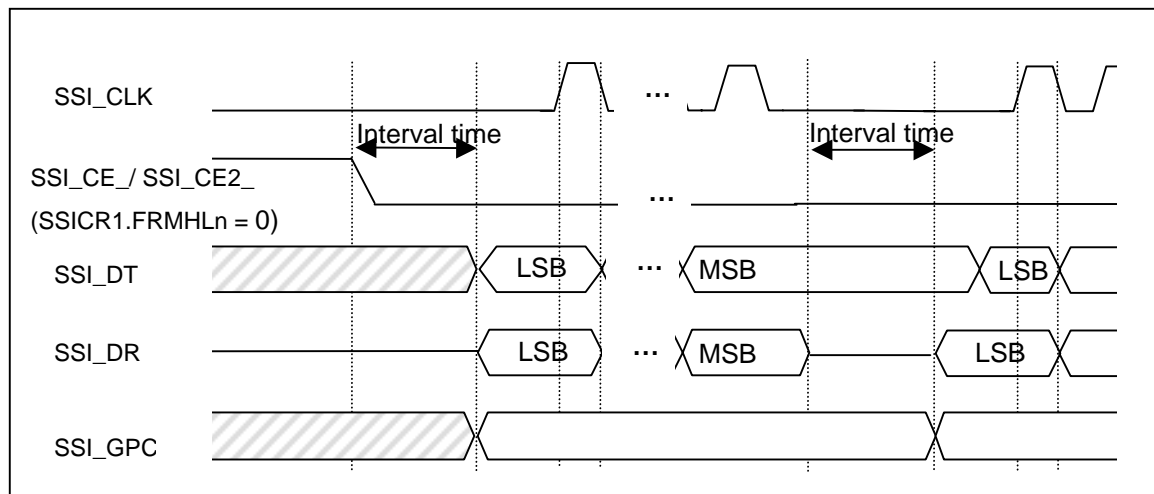


Figure 1-5 SPI Frame Interval Mode Transfer Format (ITFRM = 1, LFST = 1)

### 1.5.2 TI's SSP Format Details

In this format, each transfer begins with SSI\_CE\_ pulsed high for one SSI\_CLK period. Then both master and slave drive data at SSI\_CLK's rising edge and sample data at the falling edge. Data are transferred with MSB first or LSB first. At the end of the transfer, SSI\_DT retains the value of the last bit sent through the next idle period.

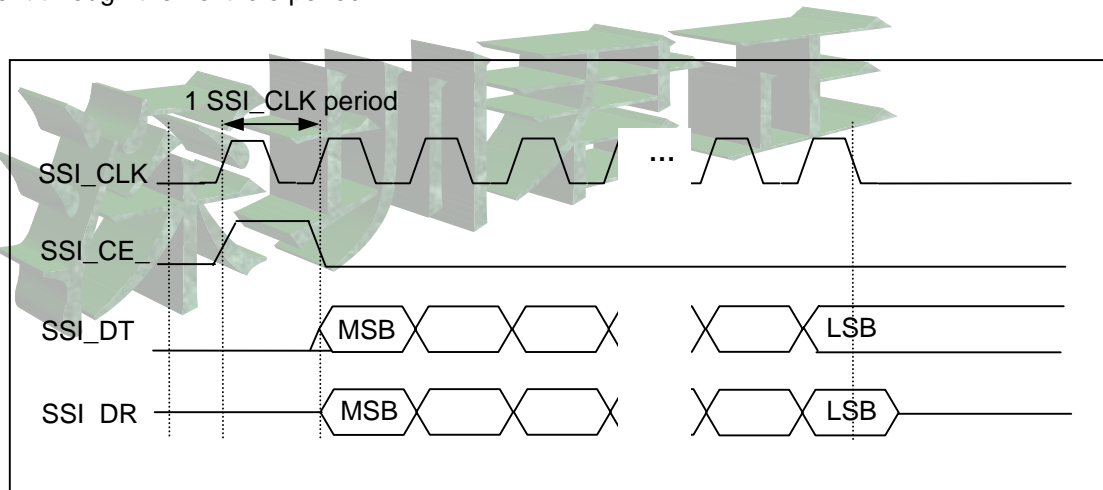


Figure 1-6 TI's SSP Single Transfer Format

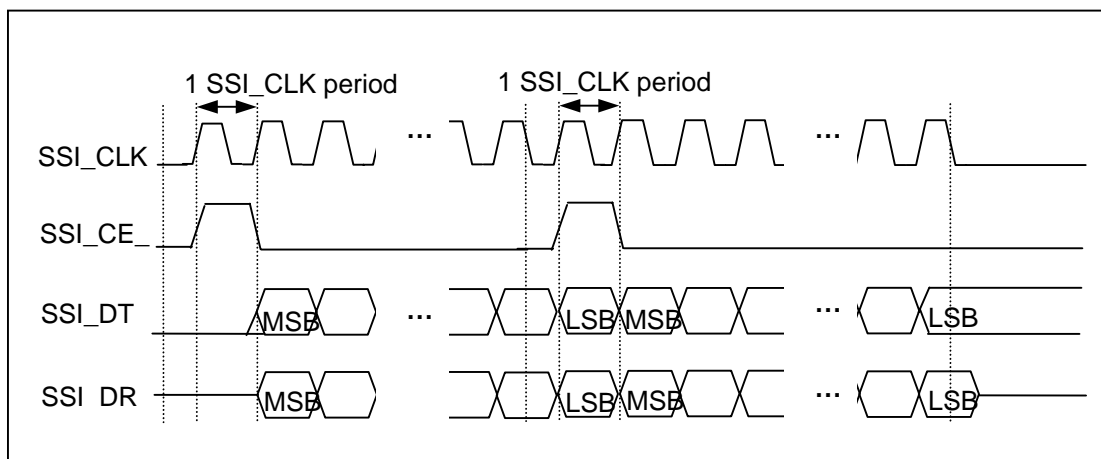


Figure 1-7 TI's SSP Back-to-back Transfer Format

### 1.5.3 National Microwire Format Details

It supports format 1 and format 2. If format 1 is selected, both master and slave drive data at SSI\_CLK falling edge and sample data at the rising edge. If format 2 is selected, master drive and sample data at SSI\_CLK falling edge, slave drive and sample data at SSI\_CLK rising edge. SSI\_CLK goes high midway through the command's most significant bit (or LSB) and continues to toggle at the bit rate. One bit clock (format 1) or half one bit clock (format 2) period after the last command bit, the external slave must return the serial data requested, with most significant bit first (or LSB first) on SSI\_DR. SSI\_CE\_ / SSI\_CE2 deasserts high half clock (SSI\_CLK) period (and 1/2/3 additional clock periods) later. Format 1 support back-to-back transfer, the start and end of back-to-back transfers are similar to those of a single transfer. However, SSI\_CE\_ / SSI\_CE2 remains asserted throughout the transfer. The end of a character data on SSI\_DR is immediately followed by the start of the next command byte on SSI\_DT.

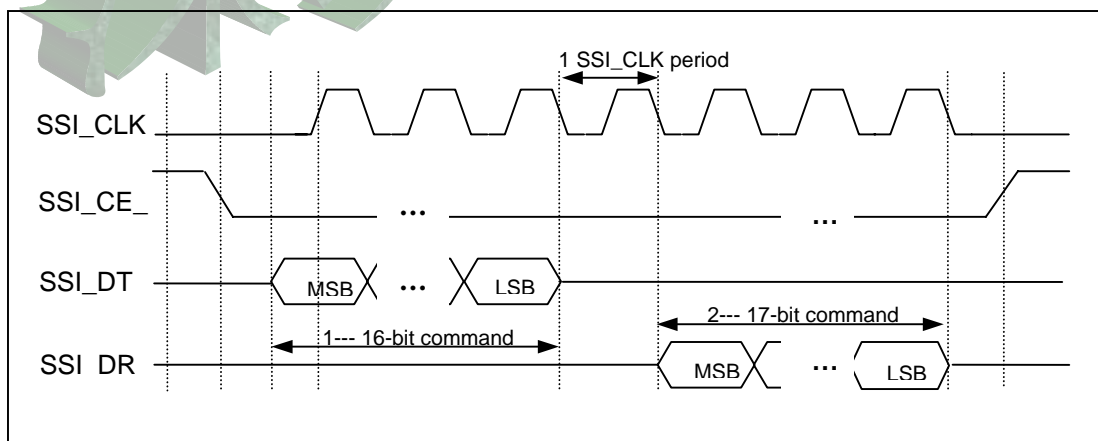


Figure 1-8 National Microwire Format 1 Single Transfer

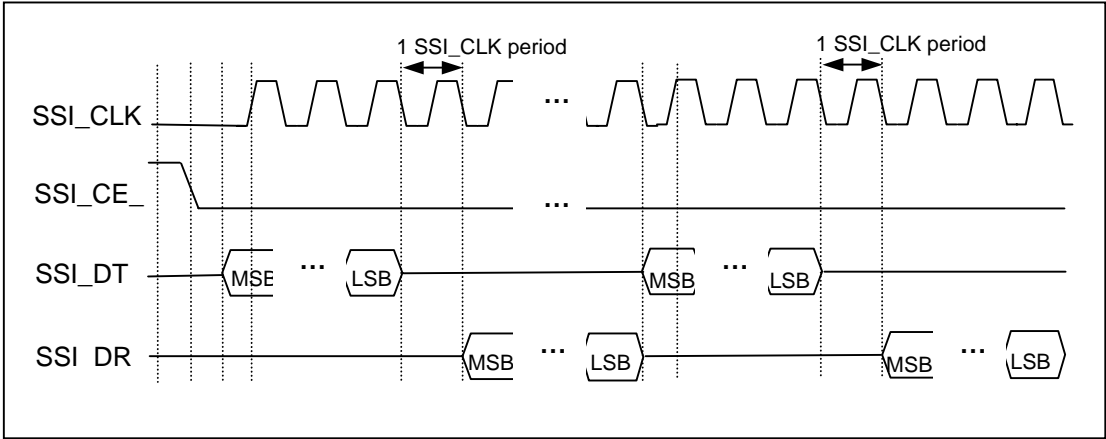


Figure 1-9 National Microwire Format 1 Back-to-back Transfer

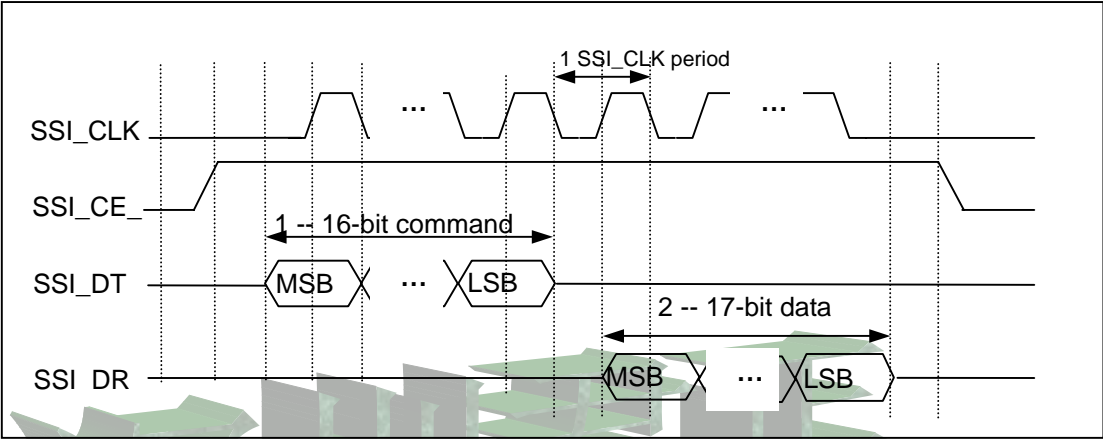


Figure 1-10 National Microwire Format 2 Read Timing

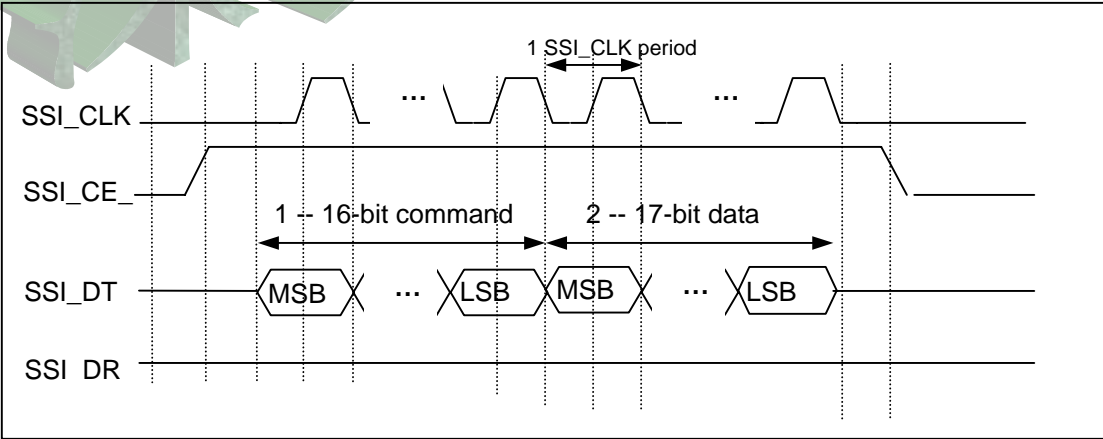


Figure 1-11 National Microwire Format 2 Write Timing

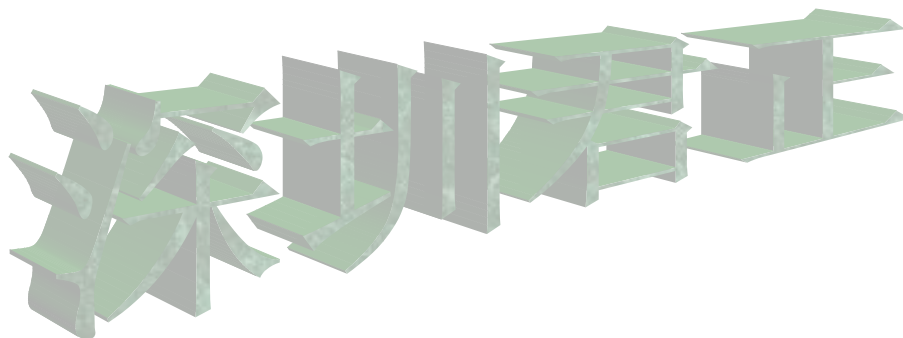
## 1.6 Interrupt Operation

In SSI, there are TXI, RXI, TEI and REI total 4 interrupts, all these interrupts are combined together to make one SSI interrupt, which can be masked by writing '1' into corresponding mask bit in INTC interrupt mask register (IMR).

**Table 1-3 SSI Interrupts**

Operation	Condition	Flag Bit	Mask Bit	Interrupt	DMAC Activation
Transmit	T-FIFO is half-empty or less	SSISR.TFHE	SSICR0.TIE	<b>TXI</b>	Possible
	Transmit underrun error	SSISR.UNDR	SSICR0.TEIE	<b>TEI</b>	Impossible
Receive	R-FIFO is half-full or more	SSISR.RFHF	SSICR0.RIE	<b>RXI</b>	Possible
	Receive overrun error	SSISR.OVER	SSICR0.REIE	<b>REI</b>	Impossible

Either SSISR.TFHE or SSISR.RFHF can activate DMA transferring when corresponding individual interrupt mask bit in SSICR0 is cleared (masked) and DMA is enabled and configured.



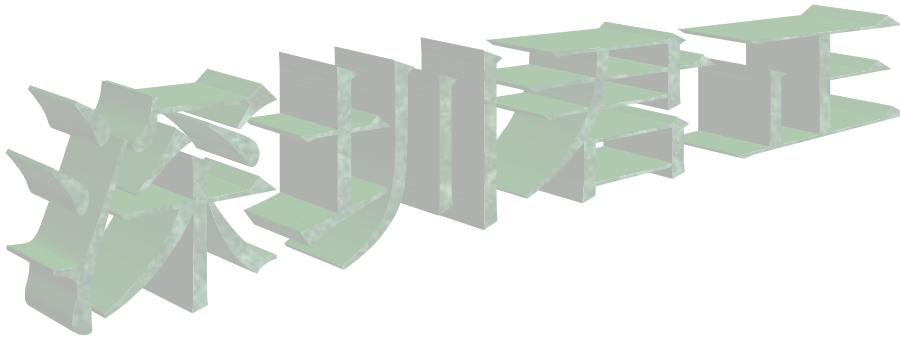
# 1 PS/2 Keyboard Controller

## 1.1 Overview

The PS/2 keyboard controller (KBC) is designed to provide the functions to a keyboard or to a PS/2 mouse. KBC receives serial data from the keyboard or mouse, checks the parity of the data, and presents the data to the system as a byte of data in its output buffer. Then, the controller will assert an interrupt to the system when data are placed in its output buffer. The keyboard and PS/2 mouse are required to acknowledge all data transmissions. No transmission should be sent to the keyboard or PS/2 mouse until acknowledge is received for the previous byte sent.

Features:

- Be compatible with 8042
- Only support PS/2 keyboard
- Bi-directional synchronous serial transfer operation
- A frame format: 1 start bit, 8 data bits, 1 odd parity bit and 1 stop bit
- Device provides about 20KHz clock to KBC
- KBC has priority over the data line and can inhibit communication from the keyboard/mouse at any time by holding Clock low
- PS/2 mouse doesn't support in JZ4730



## 1.2 Pin Description

Name	TYPE	Description
KCLK	IO	Keyboard Clock pin
KDAT	IO	Keyboard Data pin

## 1.3 Register Description

All KBC register access address is KBC physical address

Name	Description	RW	Reset Value	Address	Access Size
KTDR	KBC Transmit Data Register	W	0x????????	0x10062060	8
KRDR	KBC Receive Data Register	R	0x????????	0x10062060	8
KCCR	KBC Command Register	W	0x????????	0x10062064	8
KCSR	KBC Status Register	R	0x00	0x10062064	8

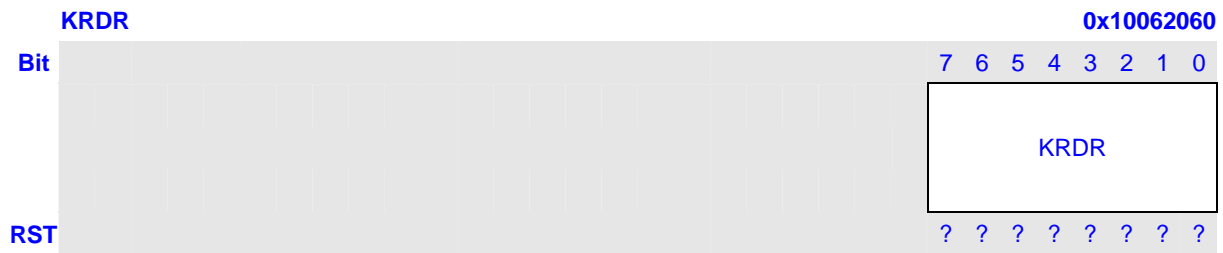
### 1.3.1 KBC Transmit Data Register (KTDR)

KTDR is a write-only input buffer. The data is written to KTDR by CPU, and then KBC read out this data from this KTDR and this data will be sent out in KDAT or MDAT pin serially. After the data in KTDR is sent out completely, KTDR becomes empty. If KTDR isn't empty, software shouldn't write KTDR again, otherwise, the new data covers the old data. According to IBF bit in KCSR, software knows whether KTDR is empty or not.



### 1.3.2 KBC Receive Data Register (KRDR)

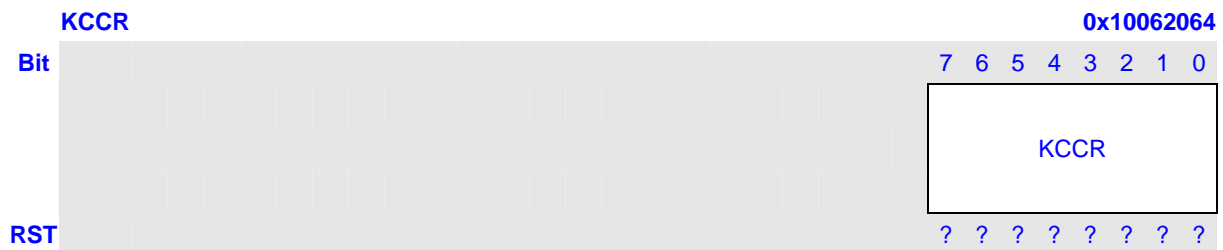
KRDR is a read-only output buffer. A byte from KDAT or MDAT pin is written to KRDR and KBC request CPU to read out this data from KRDR. Before the data in KRDR is read out by CPU, KCLK or MCLK pin is pulled low by KBC to prevent keyboard or mouse from transmitting data to KBC. After CPU reads out the data in KRDR, KBC release KCLK or MCLK pin, then keyboard or mouse can transmit data to KBC.



### 1.3.3 KBC Command Register (KCCR)

KCCR is a write-only register. Writing KCCR doesn't write to any specific register, but sends a command for the KBC to interpret. If the command accepts a parameter, this parameter is sent to KTDR. Likewise, any results returned by the command may be read from KRDR.

The usable command to KBC is described in back text.



### 1.3.4 KBC Status Register (KCSR)



Bits	Name	Description	RW
7	PER	Parity Error, if this bit is set to 1, it indicates that the data received from the keyboard or mouse had even parity. After a reading KRDR operation, PER bit is cleared to 0 automatically 0: Odd parity received 1: Even parity received	R
6	TOT	Time-Out, when this bit is set to 1, it indicates that a transmission was started by the keyboard or mouse but did not finish within the receive time-out delay (2ms), or that a transmission was started by KBC but the byte transmitted was not clocked out within the specified time limit (15ms).	R



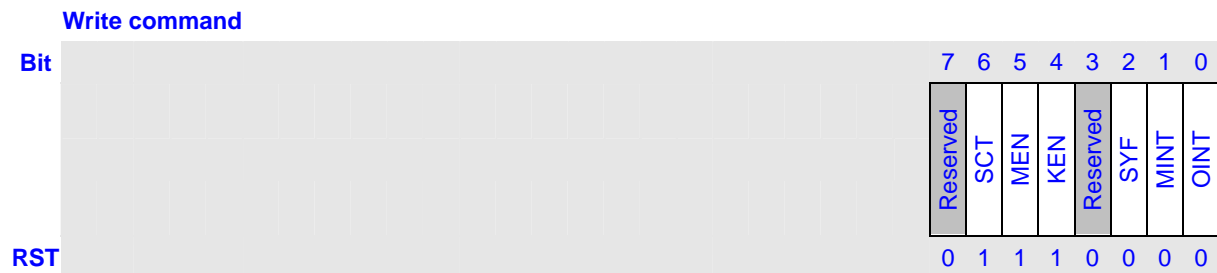
		<p>When a receive time-out occurs, KBC places a hex FF in the output buffer. When a transmit time-out occurs, KBC places a hex FE in the output buffer. After a reading KRDR operation, TOT bit is cleared to 0 automatically</p> <p>0: No Time-out 1: Time-out occurs</p>	
5	MOF	<p>Mouse Output Buffer Full. This bit works in conjunction with OF bit. When this bit and OTF bit are set to 1, mouse data is in the output buffer and MOF interrupt is asserted if INT2 bit in command byte is 1. When this bit is 0 and OTF is set to 1, keyboard or KBC command response data is in the output buffer. After a reading KRDR operation, MOF bit is cleared to 0 automatically</p> <p>0: The data in output buffer isn't mouse data 1: The data in output buffer is mouse data</p>	R
4	INF	<p>Inhibit Flag, this bit indicates whether or not keyboard communication is inhibited. If this bit is set to 0, KCLK pin is pulled low, so keyboard is inhibited</p> <p>0: Keyboard is inhibited 1: Keyboard isn't inhibited</p>	R
3	AD2	<p>Address Line A2. This bit indicates which register was last written to. If this bit is 0, the last write operation is writing KTDR, otherwise is KCCR</p> <p>0: KTDR was last written 1: KCCR was last written</p>	R
2	SYF	<p>System Flag. This bit is set to 0 or 1 by writing to the system flag bit (SYS bit in command byte). This bit is cleared to 0 after a power-on reset</p> <p>0: SYS bit in command byte is 0 1: SYS bit in command byte is 1</p>	R
1	ITF	<p>Input Buffer Full. This bit indicates whether input buffer is full. Writing KTDR operation for transmitting data to keyboard or mouse set this bit to 1. After the data in KTDR is sent out completely, this bit is cleared to 0:</p> <p>Input buffer is empty 1: Input buffer is full</p>	R
0	OTF	<p>Output Buffer Full. This bit indicates whether output buffer is full. When a byte keyboard or mouse data is put to KRDR, or KBC's command response data is put to KRDR, this bit is set to 1 and OBF interrupt is asserted if INT bit in command byte is 1. After a reading KRDR operation, OTF bit is cleared to 0 automatically</p> <p>0: Output buffer is empty 1: Output buffer is full</p>	R

## 1.4 KBC Commands

Commands are sent to the keyboard controller by writing to port 0x64 (KCCR). Command parameters are written to port 0x60 (KTDR) after command is sent. Results are returned on port 0x60 (KRDR). Always test the IBF flag before writing commands or parameters to KBC.

### 1.4.1 Write command byte (0x60)

KBC has a command byte register that is used to configure KBC. Command byte's parameter via KTDR is written in command byte register. Command byte defined as follows:



Bits	Name	Description	RW
7	Reserved	Writes to these bits have no effect and always read as 0	R
6	SCT	Scan Codes Translate. When this bit is set to 1, KBC translates the incoming keyboard scan codes to scan set 1. When this bit is set to 0, KBC passes the incoming scan codes without translation. Following power-on or a keyboard reset, the keyboard transmits using scan code set 2 0: Disable translation 1: Enable translation	RW
5	MEN	Disable Mouse, setting this bit to 1 disables the mouse interface by driving the MCLK pin low. Mouse data can't be received while the mouse interface is disabled. Clearing this bit to 0 enables the mouse interface by releasing the MCLK pin 0: Enable mouse interface 1: Disable mouse interface	RW
4	KEN	Disable Keyboard, setting this bit to 1 disables the keyboard interface by driving the KCLK pin low. Keyboard data can't be received while the keyboard interface is disabled. Clearing this bit to 0 enables the keyboard interface by releasing the KCLK pin 0: Enable keyboard interface 1: Disable keyboard interface	RW
3	Reserved	Writes to these bits have no effect and always read as 0	R
2	SYF	System Flag. Software sets this bit to tell KBC perform power-on or "warm boot" test/initialization. This bit can be used to manually set/clear SYS flag in Status register	RW

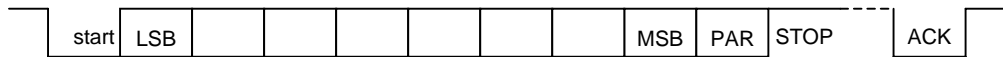
		0: Tell KBC to perform power-on test/initialization 1: Tell KBC to perform "warm boot" tests/initialization	
1	MINT	Mouse Output Buffer Full Interrupt. When set, MOF interrupt is generated when mouse data is available  0: Disable MOF interrupt 1: Enable MOF interrupt	RW
0	OINT	Output Buffer Full Interrupt. When set, OTF interrupt is generated when data is available in the output buffer  0: Disable OTF interrupt 1: Enable OTF interrupt	RW

### 1.4.2 COMMANDS

COMMAND	FUNCTION
20h	Read Command Byte of Keyboard Controller
60h	Write Command Byte of Keyboard Controller
A4h	Test Password Returns 0Fah if Password is loaded Returns 0F1h if Password is not loaded Current , KBC doesn't support password function, 0F1h always returned
A7h	Disable mouse interface
A8h	Enable mouse interface
A9h	Mouse interface test 00: No error detected 01: MCLK is stuck low 02: MCLK is stuck high 03: MDAT is stuck low 04: MDAT is stuck high
AAh	Self-test Returns 055h if self test succeeds
ABh	interface test 00: No error detected 01: KCLK is stuck low 02: KCLK is stuck high 03: KDAT is stuck low 04: KDAT is stuck high
ADh	Disable keyboard interface
A Eh	Enable keyboard interface
C0h	read its input port and send the data in its output buffer
D2h	Write keyboard output buffer

D3h	Write mouse output buffer
D4h	Write mouse input buffer
E0h	Reports the status of the test inputs

## 1.5 Frame format



1 start bit: This is always 0.

8 data bits: least significant bit first.

1 parity bit: (odd parity).

1 stop bit: This is always 1.

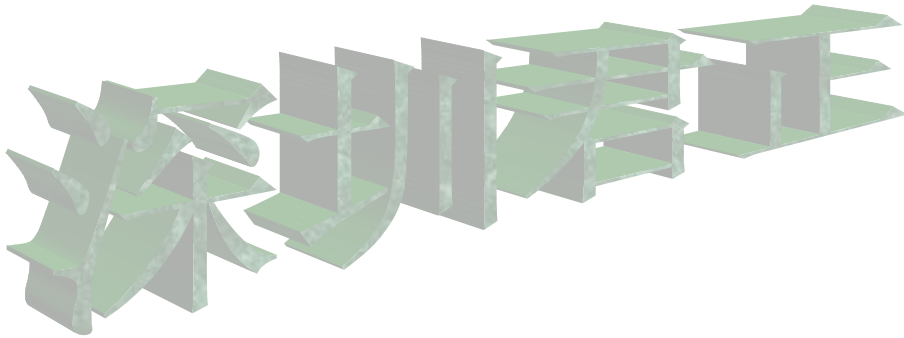
1 acknowledge bit: This is always 0 (Host-to-device communication only)

## 1.6 Transmit flow

1. transmit start
2. check if ITF == 0 then 3 else return 1
3. Disable keyboard or mouse
4. read KCSR
5. check if OTF == 0 then 6 else read KRDR
6. check if mouse transmission, then 7 else 8
7. write D4 to KCCR, then 9
8. clear SCT to 0
9. write KTDR
10. enable keyboard or mouse
11. transmit end

## 1.7 Receive flow

1. Receive start
2. Read KCSR
3. check if OTF == 1, then 4 else 1
4. check if PER == 1, then 5 else 6
5. check if MOF == 1, then go to Mouse parity error handle else go to Keyboard parity error handle
6. check if TOT == 1, then 7 else 8
7. check if MOF == 1, then go to Mouse time-out error handle or go to Keyboard time-out handle
8. check if MOF == 1, then 9 else 10
9. read KRDR for getting mouse data, then 11
10. read KRDR for getting keyboard data
11. receive end



# 1 System Bus Arbiter

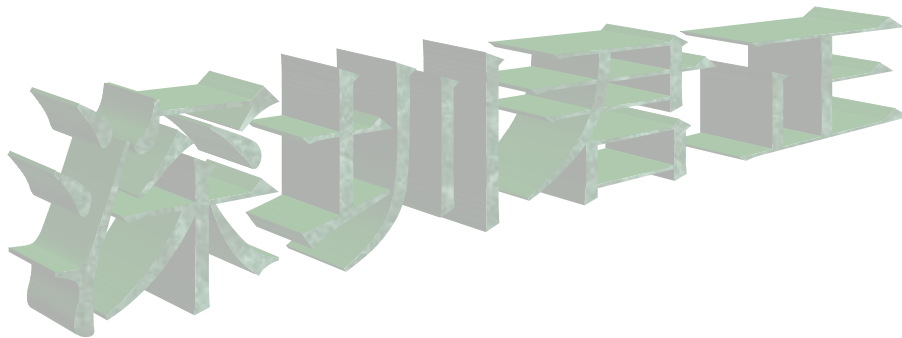
## 1.1 Overview

This chapter describes the internal system bus arbitration mechanism included in the JZ47XX processor

The JZ47XX processor system bus supports six clients — `cpu_core`, the DMA controller, the LCD controller, the USB host controller, the eth controller and the camera interface controller. The bus is multiplexed (instead of a three-state approach), and clients can request the bus without any limitations. Arbitration for bus access is performed by the arbiter, which is programmable through the `SBA_CNTL` register.

The SBA has the following features:

- Programmable client weights
- Bus locking



## 1.2 Register Description

SBA register 32-bit access address is physical address.

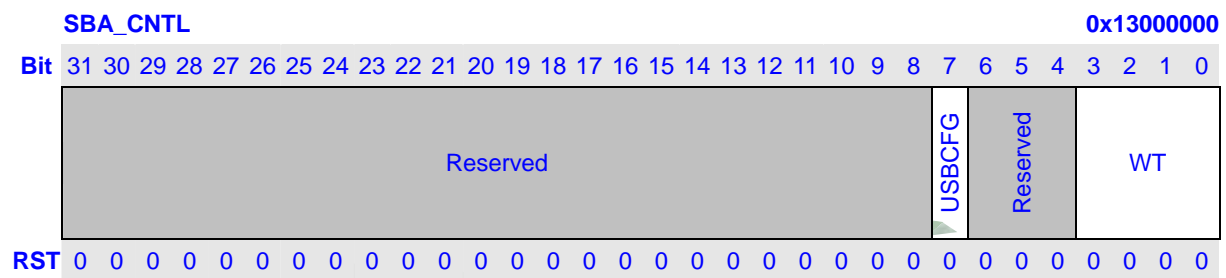
The SBA only contains a register are shown.

Name	Description	RW	Reset Value	Address	Access Size
SBA_CNTL	SBA Control Register	RW	0x00000000	0x13000000	32

**Table 1-1 System Bus Arbiter Register Description**

### 1.2.1 SBA Control Register (SBA\_CNTL)

SBA\_CNTL is used to set the weights of six clients.

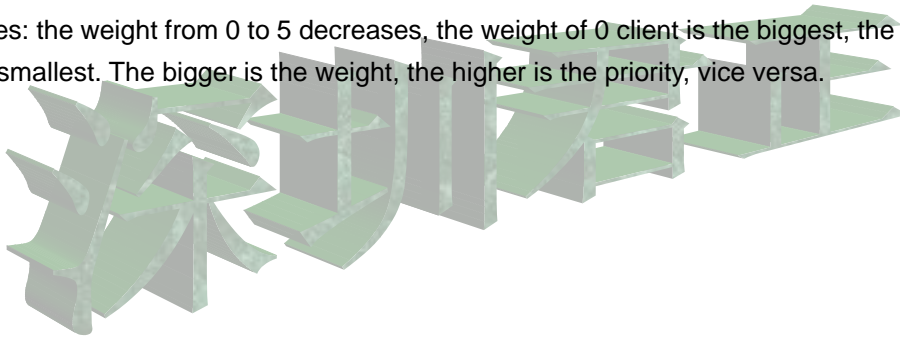


Bits	Name	Description	RW
31:8	Reserved	Only Read	R
7	USBCFG	<b>USB Configure</b> 0 = set USB port0 device function 1 = set USB port0 host function	RW
6:4	Reserved	Only Read	R
3:0	WT	<b>Weight</b> These bits are used to set the weights of six clients. See Table 1-2.	RW

Table 1-2 Weights of six clients

WT	Six Clients					
	0	1	2	3	4	5
0000	CIM	LCDC	DMAC	ETHC	UHC	CPU_CORE
0001	CIM	LCDC	ETHC	DMAC	UHC	CPU_CORE
0010	CIM	LCDC	UHC	DMAC	ETHC	CPU_CORE
0011	LCDC	CIM	DMAC	ETHC	UHC	CPU_CORE
0100	LCDC	CIM	ETHC	DMAC	UHC	CPU_CORE
0101	LCDC	CIM	UHC	DMAC	ETHC	CPU_CORE
0110	DMAC	CIM	ETHC	LCDC	UHC	CPU_CORE
0111	DMAC	CIM	UHC	LCDC	ETHC	CPU_CORE
1000	DMAC	LCDC	CIM	ETHC	UHC	CPU_CORE
1001	ETHC	CIM	LCDC	DMAC	UHC	CPU_CORE
1010	ETHC	LCDC	CIM	DMAC	UHC	CPU_CORE
1011	ETHC	UHC	CIM	LCDC	DMAC	CPU_CORE
1100	UHC	CIM	LCDC	DMAC	ETHC	CPU_CORE
1101	UHC	LCDC	CIM	DMAC	ETHC	CPU_CORE
1110	CPU_CORE	CIM	LCDC	DMAC	ETHC	UHC
1111	CPU_CORE	LCDC	CIM	DMAC	ETHC	UHC

Notes: the weight from 0 to 5 decreases, the weight of 0 client is the biggest, the weight of 5 client is the smallest. The bigger is the weight, the higher is the priority, vice versa.





# 1 Power up, Reset and Boot

## 1.1 Overview

For both hardware and runtime resets, the CPU boots from KSEG1 address 0xBFC00000, which is translated to physical address 0x1FC00000; therefore, the system designer must place the start of the boot code at 0x1FC00000.

There are some functional modes that must be set during the system reset phase by setting the state of the function mode pins, and cannot be changed after the reset. These functional mode pins are multiple usages. When the system runs normally based on the mode that is set when reset, these pins are used for different functions of the system.

The pins can be found in the pin list table of the processor. The detailed information about their function is described in relevant chapter of the manual.

## 1.2 Boot Mode – BOOT\_SEL3 ~ 0

Pins BOOT\_SEL3 ~ 0 are used to define the parameters for boot, include boot memory, data bus width and corresponding parameters at power-on reset. Please refer to the table list below.

BOOT_SEL3	BOOT_SEL2	BOOT_SEL1	BOOT_SEL0	Bank 0 Memory Size
0 :  Boot from bank 0 of static memory		0	0	Bank 0 memory size is 32 bits
		0	1	Bank 0 memory size is 16 bits
		1	0	Bank 0 memory size is 8 bits
	0			TP pins are normal work mode
	1			TP pins are used for test ports
1 :  Boot from NAND flash			0	NAND flash width is 8 bit
			1	NAND flash width is 16 bit
		0		NAND flash page size is 512B
		1		NAND flash page size is 2048B
	0			NAND flash page address cycle number is 2
	1			NAND flash page address cycle number is 3

### 1.3 Work mode – TP\_MD

The TP\_MD pin is used for observing chip internal signals from corresponding pins.

TP_MD	Description
0	Normal mode
1	CPM test

### 1.4 TAP Select – TAP\_MD

The Jz47xx processor has an internal core TAP and a boundary TAP, which can be selected by the external pin – TAP\_MD.

TAP_MD	Description
0	Select and enable boundary TAP
1	Select and enable core TAP

