1 Interrupt Controller

1.1 Overview

This chapter describes the interrupt controller included in the JZ4XX 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	0xFFFFFFFF	0x10001004	32
ICMSR	Interrupt controller Mask Set Register	W	0x????????	0x10001008	32
ICMCR	Interrupt controller Mask Clear Register	W	0x????????	0x1000100C	32
ICPR	Interrupt controller Pending Register	R	0x00000000	0x10001010	32

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	IPU	GPIO0	GPIO1	GPIO2	GPIO3	UDC	TCU0	TCU1	TCU2	DMA	Reserved	AIC	CIM	SSI	RTC	MSC	Reserved	SADC	Reserved	Reserved	UART0	Reserved	Reserved	Reserved	Reserved	Reserved	UHC	EMC	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	IPU	GPIO0	GPIO1	GPIO2	GPIO3	UDC	TCU0	TCU1	TCU2	DMA	Reserved	AIC	CIM	SSI	RTC	MSC	Reserved	SADC	Reserved	Reserved	UART0	Reserved	Reserved	Reserved	Reserved	Reserved	UHC	EMC	I2C	Reserved
RST	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits 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	IPU	GPIO0	GPIO1	GPIO2	GPIO3	UDC	TCU0	TCU1	TCU2	DMA	Reserved	AIC	CIM	SSI	RTC	MSC	Reserved	SADC	Reserved	Reserved	UART0	Reserved	Reserved	Reserved	Reserved	Reserved	UHC	EMC	I2C	Reserved
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

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	IPU	GPIO0	GPIO1	GPIO2	GPIO3	UDC	TCU0	TCU1	TCU2	DMA	Reserved	AIC	CIM	SSI	RTC	MSC	Reserved	SADC	Reserved	Reserved	UART0	Reserved	Reserved	Reserved	Reserved	Reserved	UHC	EMC	I2C	Reserved
RST	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

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	IPU	GPIO0	GPIO1	GPIO2	GPIO3	UDC	TCU0	TCU1	TCU2	DMA	Reserved	AIC	CIM	SSI	RTC	MSC	Reserved	SADC	Reserved	Reserved	UART0	Reserved	Reserved	Reserved	Reserved	Reserved	UHC	EMC	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.