

UCD30xx Central Interrupt Module (CIM) Programmer's Manual

Literature Number: xxxxxx

Date

Table of Contents

1	Description.....	3
2	Interrupt Vector Table	3
3	Interrupt Handling at the CPU	5
4	Interrupt Generation at the Peripheral	5
5	CIM Interrupt Management	5
5.1	CIM Input Channel Management.....	6
5.2	CIM Prioritization	8
5.3	CIM Operation	8
6	Register Map	8
7	Register Description.....	8
7.1	IRQ Index Offset Vector Register (IRQIVEC)	9
7.2	FIQ Index Offset Vector Register (FIQIVEC)	9
7.3	Reserved Register	10
7.4	FIQ/IRQ Program Control Register (FIRQPR)	10
7.5	Pending Interrupt Read Location Register (INTREQ)	10
7.6	Interrupt Mask Register (REQMASK)	10

1 Description

The Central Interrupt Module accepts 32 interrupt requests for meeting firmware timing requirements. The ARM itself only supports two levels of interrupts, FIQ and IRQ. With FIQ being the higher interrupt to IRQ. The CIM provides hardware expansion of interrupts by use of FIQ/IRQ vector registers for providing the offset index in a vector table. This numerical index value indicates the highest precedence channel with a pending interrupt and is used to locate the interrupt vector address from the interrupt vector table. Interrupt channel 31 has the highest precedence and interrupt channel 0 has the lowest precedence. The CIM is level sensitive to the interrupt requests and each peripheral will need to keep the request high until the ARM responds to it. To remove the interrupt request, the firmware should clear the request as the first action in the interrupt service routine. The request channels are maskable to selectively disable individual channels

2 Interrupt Vector Table

Name	Memory Module Name	Module Component or Register	Description	Priority
Unused				(lowest) 0
BRN_OUT_INT	Misc. Analog Control	Brownout	Brownout interrupt	1
EXT_INT	GIO	External Interrupts	Interrupt on one or all external input pins.	2
WDRST_INT	Timer	Watchdog Control	Interrupt from watchdog exceeded (reset)	3
WDWAKE_INT	Timer	Watchdog Control	Wakeup interrupt when watchdog equals half of set watch time	4
SCI_ERR_INT	UART or SCI	UART or SCI Control	UART or SCI error Interrupt. Frame, parity or Overrun	5
SPI_INT	SPI	SPI Control	SPI related interrupt for overrun and/or end of SPI transmission	6
SCI_RX_INT	UART or SCI	UART or SCI Control	UART RX buffer has a byte	7
SCI_TX_INT	UART or SCI	UART or SCI Control	UART TX buffer empty	8
PMBUS_INT	PMBUS		PMBus related interrupt	9
COMP_INT	Misc. Analog Control	Analog Comparator Control	Analog comparator interrupt	10
DIG_COMP_INT	ADC	12-bit ADC Control	Digital comparator interrupt	11
OVF16_4_INT	Timer	16-bit Timer PWM 4	16-bit Timer PWM4 counter overflow interrupt	12
PWM4CMP_INT	Timer	16-bit Timer PWM 4	16-bit Timer PWM4 counter Compare interrupt	13

OVF16_3_INT	Timer	16-bit Timer PWM 3	16-bit Timer PWM3 counter overflow interrupt	14
PWM3CMP_INT	Timer	16-bit Timer PWM 3	16-bit Timer PWM3 counter Compare interrupt	15
OVF16_2_INT	Timer	16-bit Timer PWM 2	16-bit Timer PWM2 counter overflow interrupt	16
PWM2CMP_INT	Timer	16-bit Timer PWM 2	16-bit Timer PWM2 counter Compare interrupt	17
OVF16_1_INT	Timer	16-bit Timer PWM 1	16-bit Timer PWM1 counter overflow interrupt	18
PWM1CMP_INT	Timer	16-bit Timer PWM 1	16-bit Timer PWM1 counter Compare interrupt	19
OVF24_INT	Timer	24-bit Timer Control	24-bit Timer counter overflow interrupt	20
CAP1_INT	Timer	24-bit Timer Control	24-bit Timer Capture 1 interrupt	21
CMP1_INT	Timer		24-bit Timer Compare 1 interrupt	22
CMP0_INT	Timer		24-bit Timer Compare 0 interrupt	23
CAP0_INT	Timer		24-bit Timer Capture 0 interrupt	24
ADC_CONV_INT	ADC	12-bit ADC Control	ADC end of conversion interrupt	25
HS Loop4	DPWM	Loop 4	same for Loop 4	26
HS Loop3	DPWM	Loop 3	same for Loop 3	27
HS Loop1	DPWM	Loop 1	1) Every (1-16) switching cycles 2) CLF flag shutdown	28
HS Loop2	DPWM	Loop 2	same for Loop 2	29
FAULT_INT	GIO	External Faults	Fault pin interrupt	30
SYS_SSI_INT	SYS	System Software	System software interrupt	(highest) 31

3 Interrupt Handling at the CPU

The ARM7 CPU provides two vectors for interrupt requests—fast interrupt requests (FIQ) and normal interrupt requests (IRQ). The CPU may enable these interrupt request channels individually within the CPSR; CPSR bits 6 and 7 must be cleared to enable the FIQ and IRQ interrupt requests at the CPU. When both interrupt requests are enabled, the FIQ interrupt request has higher priority than the IRQ and is handled first.

When the CPU recognizes an interrupt request, the CPSR changes mode to either the FIQ or IRQ mode. When an IRQ interrupt is recognized, the CPU disables other IRQ interrupts by setting CPSR bit 7. When an FIQ interrupt is recognized, the CPU disables both IRQ and FIQ interrupts by setting CPSR bits 6 and 7. After the interrupt is recognized by the CPU, the program counter jumps to the appropriate interrupt vector—0x0018 for IRQ and 0x001C for FIQ.

4 Interrupt Generation at the Peripheral

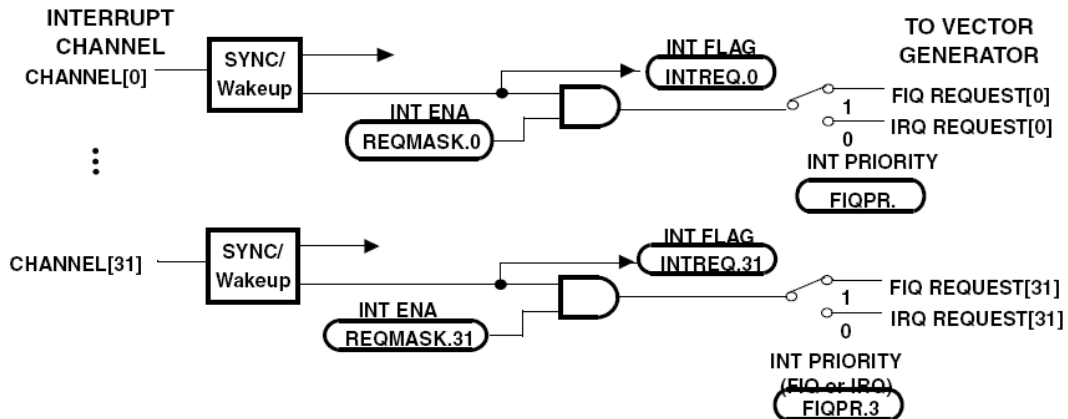
Interrupts begin when an event occurs within a peripheral module. Some examples of interrupt-capable events are expiration of a counter within a timer module, receipt of a character in a communications module, and completion of a conversion in an analog-to-digital converter (ADC) module.

Interrupts are not always generated when an event occurs; the peripheral must make an interrupt request to the central interrupt manager (CIM) based upon the event occurrence. Typically, the peripheral contains:

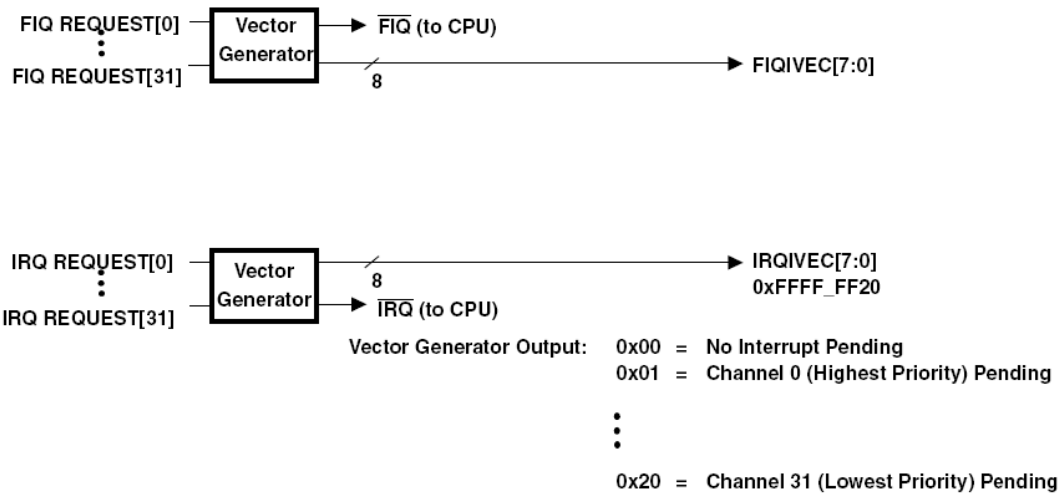
- An interrupt flag bit for each event to signify the event occurrence
- An interrupt-enable bit to control whether the event occurrence causes an interrupt request to the CIM

5 CIM Interrupt Management

A block diagram of the CIM is shown below:



(a) CIM Interrupt Channel Control



(b) CIM Vector Generator

The CIM can support 32 interrupt request lines (channel [0] to channel [31]) from the peripherals. These peripheral interrupt requests are hardwired to each of the CIM 32 channels. All requests pass through a synchronizer to prevent setup time violations; the CIM samples the interrupt requests from the synchronizer every system clock (SYSCLK) cycle. The CIM combines the 32 channels into two outputs – an FIQ request to the CPU and an IRQ request to the CPU. The CIM performs the following functions:

- Manages the input channels
- Prioritizes the interrupt requests to the CPU

5.1 CIM Input Channel Management

On the input side, the CIM enables channels on a channel-by-channel basis (in the REQMASK register); unused channels may be masked to prevent spurious interrupts. Each interrupt channel can be designated to send either an FIQ or IRQ request to the CPU (in the FIQPR register).

Interrupt Mask Register REQMASK and FIQ/IRQ Program Control Register FIRQPR are writeable in privilege mode only. A write in user mode to these Registers causes a peripheral illegal access exception. One way of setting the CPU in privilege mode is through software interrupt. A software interrupt is a synchronous exception generated by the execution of a particular instruction. A C application can invoke a software interrupt by associating a software interrupt number with a function name through use of the SWI_ALIAS pragma and then calling the software interrupt as if it were a function. A C code example of using software interrupt is shown below:

```
#pragma SWI_ALIAS (write_firqpr, 8)
void write_firqpr(unsigned long value);

#pragma SWI_ALIAS (write_reqmask, 9)
void write_reqmask(unsigned long value);

#pragma INTERRUPT (software_interrupt, SWI)
void software_interrupt (Uint32 arg1, Uint32 arg2, Uint32 arg3, Uint8 swi_number)
{
    //make sure interrupts are disabled
    asm(" MRS   r3, cpsr ");           // get psr
    asm(" ORR   r3, r3, #0xc0 ");      // set interrupt disables
    asm(" MSR   cpsr, r3");            // restore psr

    asm(" LDRB  R3,[R14,#-1]"); //get swi number into R3 as fourth operand

    switch (swi_number) //handle flash write/erase and ROM backdoor first
    {
        ...
        case 8: //write to fiq/irq program_control_register
            CimRegs.FIRQPR.all = arg1;
            return;
        case 9: //write to fiq/irq program_control_register
            CimRegs.REQMASK.all = arg1;
            return;
        ...
        default:
            break;
    }
}
```

The INTERRUPT pragma enables you to handle interrupts directly with C code. This pragma specifies that the function to which it is applied is an interrupt. The type of interrupt is specified by the pragma. The software interrupt will change the CPU to privilege mode. The SWI_ALIAS pragma tells the function write_firqpr() and

write_reqmask() are software interrupts. Calls to these functions are compiled as software interrupts. A C code example of calling these functions is shown below.

```
write_firqpr(0xc000000); //make them all irqs except dpwm4 and dpwm3.
write_reqmask(0xc080000); //enable only pwm1cmp, dpwm3 and dpwm4
```

5.2 CIM Prioritization

The CIM prioritizes the received interrupts based upon a hardware and software prioritization scheme. The software prioritization scheme is user configurable. The CIM can send two interrupt requests to the CPU simultaneously—one IRQ and one FIQ. If both interrupt types are enabled at the CPU, then the FIQ has greater priority and is handled first. The hardware prioritization scheme sends the highest numbered active channel (in each FIQ and IRQ interrupt request) to the CPU. Within the FIQ and IRQ classes of interrupts, the highest channel has the highest priority interrupt. The CIM sends the highest priority interrupt of both the IRQ and FIQ classes of interrupt requests to the CPU.

5.3 CIM Operation

When the CPU recognizes an interrupt request and responds, the program counter jumps to the appropriate interrupt vector. The interrupt vector is typically a branch statement to an interrupt table. The interrupt table reads the pending interrupt from a vector offset register (FIQIVEC.7:0 for FIQ interrupts and IRQIVEC.7:0 for IRQ interrupts).

6 Register Map

Address	Register Name	Description	Bits	Read	Write	Reset
0xFFFF_FF20	IRQIVEC	IRQ Index Offset Vector Register	8	Yes	No	8'h0
0xFFFF_FF24	FIQIVEC	FIQ Index Offset Vector Register	8	Yes	No	8'h0
0xFFFF_FF28		RESERVED				
0xFFFF_FF2C	FIRQPR	FIQ/IRQ Program Control Register	32	Yes	Yes	32'h0
0xFFFF_FF30	INTREQ	Pending Interrupt Read Location	32	Yes	Yes	32'h0
0xFFFF_FF34	REQMASK	Interrupt Mask Register	32	Yes	Yes	32'h0

7 Register Description

CIM Registers have the following attributes:

- 32-bit wide
- Addresses placed on word boundaries
- Byte, half-word and word writes permitted
- All Registers have read/write access in any mode

- Interrupt Mask and FIQ/IRQ Program Control Registers are writeable in privilege mode only. A write in user mode to these Registers causes a peripheral illegal access exception.

7.1 IRQ Index Offset Vector Register (IRQIVEC)

Address FFFFFFF20

Bit Number	7:0
Bit Name	IRQIVEC
Access	R
Default	-

Bits 7-0: IRQIVEC – Index of the IRQ Pending Interrupt (Cleared upon read)

0 = No interrupt pending

1 = Pending interrupt on Channel 0

2 = Pending interrupt on Channel 1

N = Pending interrupt on Channel N-1, where $N \leq 31$

7.2 FIQ Index Offset Vector Register (FIQIVEC)

Address FFFFFFF24

Bit Number	7:0
Bit Name	FIQIVEC
Access	R
Default	-

Bits 7-0: FIQIVEC – Index of the FIQ pending interrupt (Cleared upon read)

0 = No interrupt pending

1 = Pending interrupt on Channel 0

2 = Pending interrupt on Channel 1

N = Pending interrupt on Channel N-1, where $N \leq 31$.

7.3 Reserved Register

Address FFFFFFF28

Reserved for future use

7.4 FIQ/IRQ Program Control Register (FIRQPR)

Address FFFFFFF2C

A 32-bit FIQ/IRQ program control Register (FIRQPR) determines whether a given interrupt request will be FIQ or IRQ type.

Bit Number	31:0
Bit Name	FIRQPR
Access	R/W
Default	0000_0000_0000_0000_0000_0000_0000_0000

Bits 31-0: FIRQPR – These bits determine whether an interrupt request from a peripheral is of type FIQ or IRQ. Each bit corresponds to one request channel.

This Register is writeable in privilege mode only.

0 = Interrupt request is of IRQ type (Default)

1 = Interrupt request is of FIQ type

7.5 Pending Interrupt Read Location Register (INTREQ)

Address FFFFFFF30

Bit Number	31:0
Bit Name	INTREQ
Access	R
Default	-

Bits 31-0: INTREQ – Pending Interrupt Requests

0 = No interrupt has occurred

1 = Interrupt is pending

7.6 Interrupt Mask Register (REQMASK)

Address FFFFFFF34

Bit Number	31:0
Bit Name	REQMASK
Access	R/W
Default	0000_0000_0000_0000_0000_0000_0000_0000

Bits 31-0: REQMASK – Interrupt Request Mask Select

0 = Interrupt request channel is disabled (Default)

1 = Interrupt request channel is enabled