LPC 2148 Interrupt Programming

As per wiki : “An interrupt is a signal sent to the CPU which indicates that a system event has a occurred which needs immediate attention“. An ‘Interrupt ReQuest‘ i.e an ‘IRQ‘ can be thought of as a special request to the CPU to execute a function(small piece of code) when an interrupt occurs. This function or ‘small piece of code’ is technically called an ‘Interrupt Service Routine‘ or ‘ISR‘. So when an IRQ arrives to the CPU , it stops executing the code current code and start executing the ISR. After the ISR execution has finished the CPU gets back to where it had stopped.

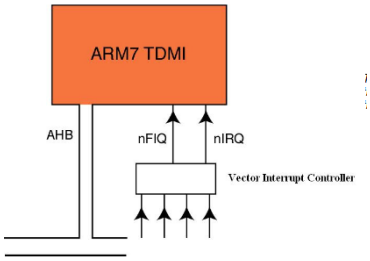
Interrupts in LPC214x are handled by Vectored Interrupt Controller (VIC) (which is specific to ARM based microcontrollers and CPUs) and are classified into 3 types based on the priority levels.

VECTORED INTERRUPT CONTROLLER (VIC) & Writing ISRs

VIC present inside the LPC 2148, manages all the interrupts generated from the ARM core, peripherals present inside the IC and also external interrupts. Each peripheral device has one interrupt line connected to the VIC (but inside the peripheral, several interrupt flags are present representing the different events for the interrupt generation).

VIC receives total 32 interrupt requests from different sources like UART, PWM unit, Timers, USB, EINT0-3 etc and categorizes them as,

a) FIQ (Fast interrupt request, has the highest priority, serviced first by FIQ handler. If more than one FIQ is occurred, FIQ handler must read the contents of FIQ status register to check the source and initiate the corresponding service routine)
b) IRQ –Medium priority, Vectored interrupt request (16 of the total 32 interrupts can be assigned to IRQ, referred as IRQ slots, 0 to 15, with slot0 having the highest priority, and slot15 has the lowest, every slot is provided with two registers, namely VICVectAddr and VICVectCntl to specify the slots interrupt vector and enable the slot, since it is vectored. Programmer has to map the peripheral i.e interrupt source to the slot, using the above registers.
c) Non-vectored IRQ – Lower priority (interrupt vector, in this case is common for all non-vectored IRQs, and is to loaded into the register VICDefVectAddr)



The VIC generates FIQ signal combining all the FIQ interrupt requests and combines the requests from all the vectored and non-vectored IRQ’s to produce IRQ signal to the ARM processor.

The VIC OR’s the request from all the vectored and non-vectored IRQs to produce the IRQ signal to the ARM processor. The IRQ service routine can start by reading a register from the

VIC and jumping there. If any of the vectored IRQs are requesting, the VIC provides address of the highest priority requesting IRQs service routine, otherwise it provides address of a default routine that is shared by all the non-vectored IRQs. The default routine can read another VIC register to see what IRQ's are active.

FIQ is serviced first, if no FIQ then IRQ related to Vectored interrupts is serviced, based on the priority assignment (IRQ0 highest, IRQ15 lowest), if no vectored interrupt request, then non-vectored IRQ is generated.

VICIntEnable : (Interrupt Enable Register): This register controls which of the 32 interrupt requests and software interrupts are enabled to contribute to FIQ or IRQ. This register used to Enable/ disable interrupts coming from different sources.
Ex: to enable timer0 interrupt - VICIntEnable = 0x0000 0010 ; bit D4 corresponds to Timer0

VICIntSelect : (Interrupt Select Register) : This register classifies each of the 32 interrupt requests as contributing to FIQ or IRQ. This register used to classify interrupt as FIQ or IRQ , by default all are IRQ. (Selection of vectored IRQ or non-vectored IRQ, that depends on the programmer using IRQ slots or not, as explained in the below section)

VICVectCntl0-15 : Vector Control Registers 0-15 each control one of the 16 vectored IRQ slots. Slot0 has the highest priority and slot15 has the lowest priority.
Vector Control registers used to map peripheral to IRQ slot and enable it, let us map timer0 to slot 4

VICCntl4 = 0x0000 0024

(D31……D5 .D4 D3 D2 D1 D0) D4 to D0 used to indicate the peripheral,
1 0 0 1 0 0 D5 to enable the channel

Note: every peripheral is given some position, in the list of 32 source of interrupts, for Timer0 it is 4, (refer data sheet/user manual of LPC 2148.)

VICVectAddr0 – 15: Vector Address Registers, used to hold interrupt vector corresponding to slot,

```
VICVectAddr4 = (unsigned long) Timer0_isr;  // Timer0_isr is the isr function
```

VICectAddr0 to VICVectAddr15: (Vector Address Registers, 16 in number): Vector Address Registers 0-15 hold the addresses of the ISRs for the 16 vectored IRQ slots.

VICVectAddr: (Vector Address Register): When an IRQ interrupt occurs, the IRQ service routine can read this register and jump to the value read.

VICDefVectAddr: (Default Vector Address Register): This register holds the address of the ISR for non-vectored IRQs.

Example Program

Using timer0 and interrupts (ISR) generate the 50Hz waveform on P1.16 (Assume PCLK is 15MHz)

Count calculation:
50Hz means, T = 1/50Hz, = 20msec, hence Ton=Toff = 10mesec
Count to be loaded in MR0 = 10msec/(1/15Mhz) = 1,50,000

```
#include <LPC2148x.h>
unsigned int x=0;
__irq   void  Timer0_ISR(void)   // an ISR program
{
        x = x ^ 1;
        if (x)
          I0SET1  =  1 << 16;   //P1.16  =  1
        else
          I0CLR1 =   1 <<16;   // P1.16  = 0
        T0IR  =  0x01; // clear match0 interrupt, and get ready for the next
        interrupt
        VICVectAddr = 0x00000000 ; //End of interrupt
}
int  main(void)
{
        I0DIR1 =  0x0001 0000;  //set P1.16 as output
        T0TCR = 0x00; // stop the timer, to initialize different registers

        T0MCR= 0x0003;  // Enable Interrupt  and  reset timer after match
        T0TC   = 0x00;  // make TC = 0
        T0MR0 = 150000; // generates 10ms

//load interrupt related registers , assigning Timer0 to IRQ  slot 4

        VICVectAdd4   = (unsigned long)Timer0_ISR;  // set the timer ISR vector address
        VICVectCntl4  =  0x0000024;  // set the channel
        VICIntEnable  =  0x0000010;  // enable the timer0 interrupt

        T0TCR = 0x01; // start the timer
         while(1)
             {
               //do other works
             };  // now timer interrupt is serviced automatically using the ISR
}
```

Note: refer Video for more explanation on interrupts

Extra Information (not in syllabus):
(taken from LPC 2148 user manual)

Interrupt Sources:

7.5 Interrupt sources

Table 63 lists the interrupt sources for each peripheral function. Each peripheral device has one interrupt line connected to the Vectored Interrupt Controller, but may have several internal interrupt flags. Individual interrupt flags may also represent more than one interrupt source.

Table 63. Connection of interrupt sources to the Vectored Interrupt Controller (VIC)

Block	Flag(s)	VIC Channel # and Hex Mask	
WDT	Watchdog Interrupt (WDINT)	0	0x0000 0001
-	Reserved for Software Interrupts only	1	0x0000 0002
ARM Core	Embedded ICE, DbgCommRx	2	0x0000 0004
ARM Core	Embedded ICE, DbgCommTX	3	0x0000 0008
TIMER0	Match 0 - 3 (MR0, MR1, MR2, MR3) Capture 0 - 3 (CR0, CR1, CR2, CR3)	4	0x0000 0010
TIMER1	Match 0 - 3 (MR0, MR1, MR2, MR3) Capture 0 - 3 (CR0, CR1, CR2, CR3)	5	0x0000 0020
UART0	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI)	6	0x0000 0040
UART1	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI) Modem Status Interrupt (MSI)[1]	7	0x0000 0080
PWM0	Match 0 - 6 (MR0, MR1, MR2, MR3, MR4, MR5, MR6)	8	0x0000 0100
I²C0	SI (state change)	9	0x0000 0200
SPI0	SPI Interrupt Flag (SPIF) Mode Fault (MODF)	10	0x0000 0400
SPI1 (SSP)	TX FIFO at least half empty (TXRIS) Rx FIFO at least half full (RXRIS) Receive Timeout condition (RTRIS) Receive overrun (RORRIS)	11	0x0000 0800
PLL	PLL Lock (PLOCK)	12	0x0000 1000
RTC	Counter Increment (RTCCIF) Alarm (RTCALF)	13	0x0000 2000

Table 63. Connection of interrupt sources to the Vectored Interrupt Controller (VIC)

Block	Flag(s)	VIC Channel # and Hex Mask	
System Control	External Interrupt 0 (EINT0)	14	0x0000 4000
	External Interrupt 1 (EINT1)	15	0x0000 8000

Interrupt Enable Register:

Interrupt Enable register (VICIntEnable - address 0xFFFF F010) bit allocation

Reset value: 0x0000 0000

Bit	31	30	29	28	27	26	25	24
Symbol	-	-	-	-	-	-	-	-
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Symbol	-	USB	AD1	BOD	I2C1	AD0	EINT3	EINT2
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Symbol	EINT1	EINT0	RTC	PLL	SPI1/SSP	SPI0	I2C0	PWM0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Symbol	UART1	UART0	TIMER1	TIMER0	ARMCore1	ARMCore0	-	WDT
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7.4.9 Vector Control registers 0-15 (VICVectCntl0-15 - 0xFFFF F200-23C)

These are a read/write accessible registers. Each of these registers controls one of the 16 vectored IRQ slots. Slot 0 has the highest priority and slot 15 the lowest. Note that disabling a vectored IRQ slot in one of the VICVectCntl registers does not disable the interrupt itself, the interrupt is simply changed to the non-vectored form.

Table 58. Vector Control registers 0-15 (VICVectCntl0-15 - 0xFFFF F200-23C) bit description

Bit	Symbol	Description	Reset value
4:0	int_request/ sw_int_assig	The number of the interrupt request or software interrupt assigned to this vectored IRQ slot. As a matter of good programming practice, software should not assign the same interrupt number to more than one enabled vectored IRQ slot. But if this does occur, the lower numbered slot will be used when the interrupt request or software interrupt is enabled, classified as IRQ, and asserted.	0
5	IRQslot_en	When 1, this vectored IRQ slot is enabled, and can produce a unique ISR address when its assigned interrupt request or software interrupt is enabled, classified as IRQ, and asserted.	0
31:6	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

vectored interrupt controller Previous Next Results Close

31:6 - Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. NA

UM10139 All information provided in this document is subject to legal disclaimers. © NXP B.V. 2012. All rights reserved.
User manual Rev. 4 — 23 April 2012 71 of 354

For example, the following two lines assign slot 0 to SPI0 IRQ interrupt request(s) and slot 1 to TIMER0 IRQ interrupt request(s):

```
VICVectCntl0 = 0x20 | 10;
VICVectCntl1 = 0x20 | 4;
```

See Table 63 "Connection of interrupt sources to the Vectored Interrupt Controller (VIC)" on page 73 for details on interrupt source channels.

7.4.10 Vector Address registers 0-15 (VICVectAddr0-15 - 0xFFFF F100-13C)

These are a read/write accessible registers. These registers hold the addresses of the Interrupt Service routines (ISRs) for the 16 vectored IRQ slots.

Table 59. Vector Address registers (VICVectAddr0-15 - addresses 0xFFFF F100-13C) bit description

vectored interrupt controller Previous Next Results Close

Bit	Symbol	Description	Reset
31:0	IRQ_vector	When an IRQ service routine reads the Vector Address register (VICVectAddr), and no IRQ slot responds as described above, this address is returned.	0x000

7.4.12 Vector Address register (VICVectAddr - 0xFFFF F030)

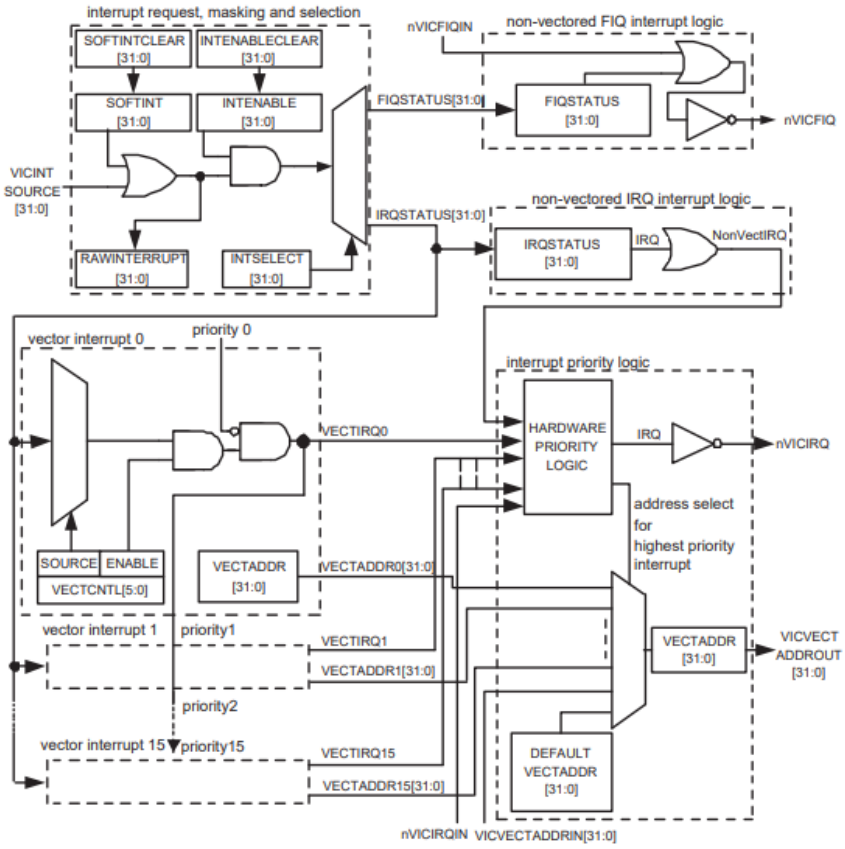
This is a read/write accessible register. When an IRQ interrupt occurs, the IRQ ser routine can read this register and jump to the value read.

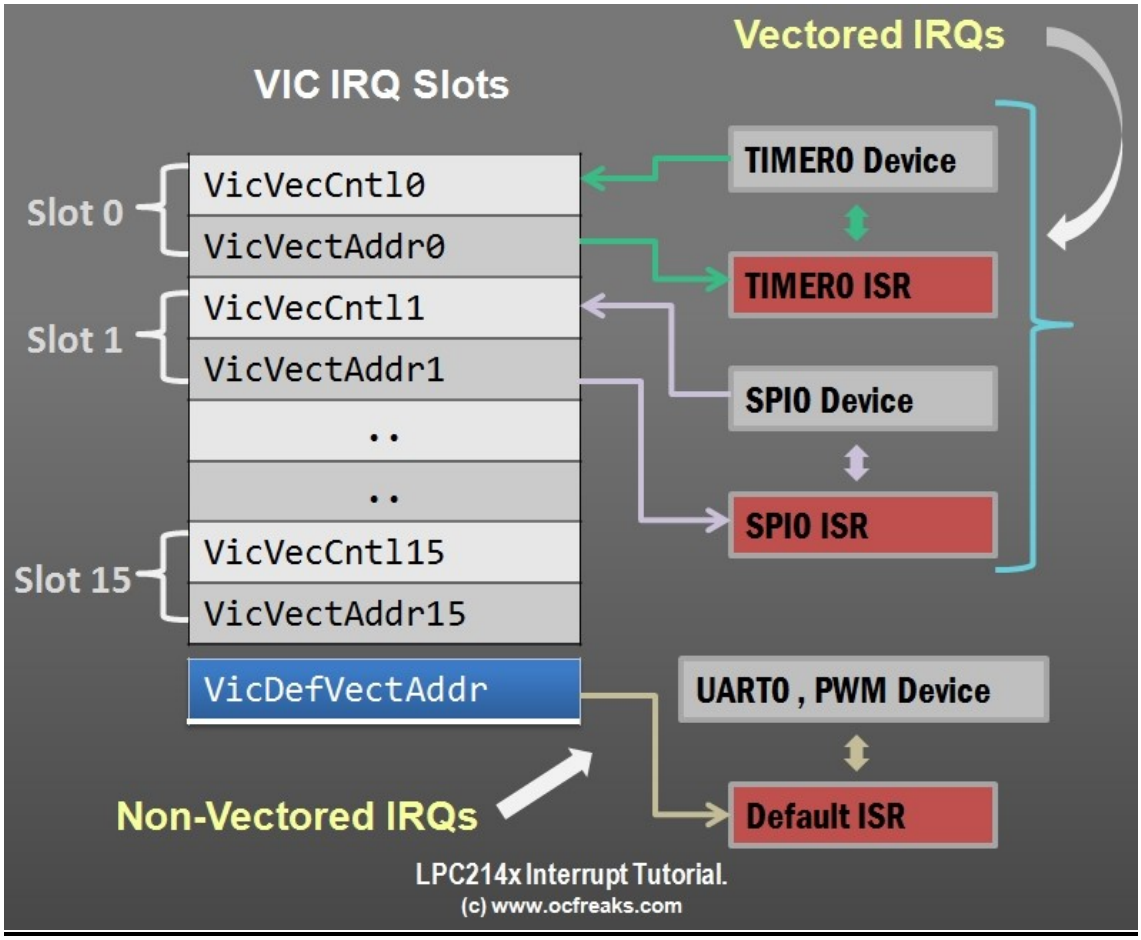
Table 61. Vector Address register (VICVectAddr - address 0xFFFF F030) bit description

Bit	Symbol	Description	Reset
31:0	IRQ_vector	If any of the interrupt requests or software interrupts that are assigned to a vectored IRQ slot is (are) enabled, classified as IRQ, and asserted, reading from this register returns the address in the Vector Address Register for the highest-priority such slot (lowest-numbered) such slot. Otherwise it returns the address in the Default Vector Address Register. Writing to this register does not set the value for future reads from it. Rather, this register should be written near the end of an ISR, to update the priority hardware.	0x000

vectored interrupt controller | Previous | Next | Results | Close

Block Diagram Of VIC





http://www.ocfreaks.com/lpc2148-interrupt-tutorial/