Lab4: Interrupt Handling

Goal

- Write subroutines to initialize devices to cause interrupts
- Enable/Disable interrupts in the processor
- Write interrupt service routines

Interrupts

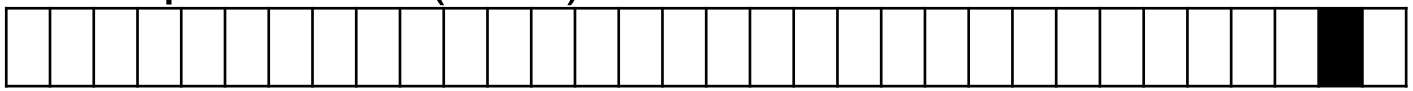
- External events that need immediate attention from the processor
- Processor receives interrupts through its interrupt request lines
- Processor can enable/disable interrupts
- When an interrupt occurs and interrupts are enabled:
  - Allow the current instruction to complete its execution
  - Store the address of the next instruction to be executed (return address) in a dedicated register
  - PC is loaded with the memory address of a particular routine, i.e. interrupt service routine (interrupt handler)

Interrupt Service Routine

- Update the stack pointer
- Store all registers on the stack
- Service the interrupt
- Restore the registers
- Return from interrupt
  - A dedicated register keeps the return address

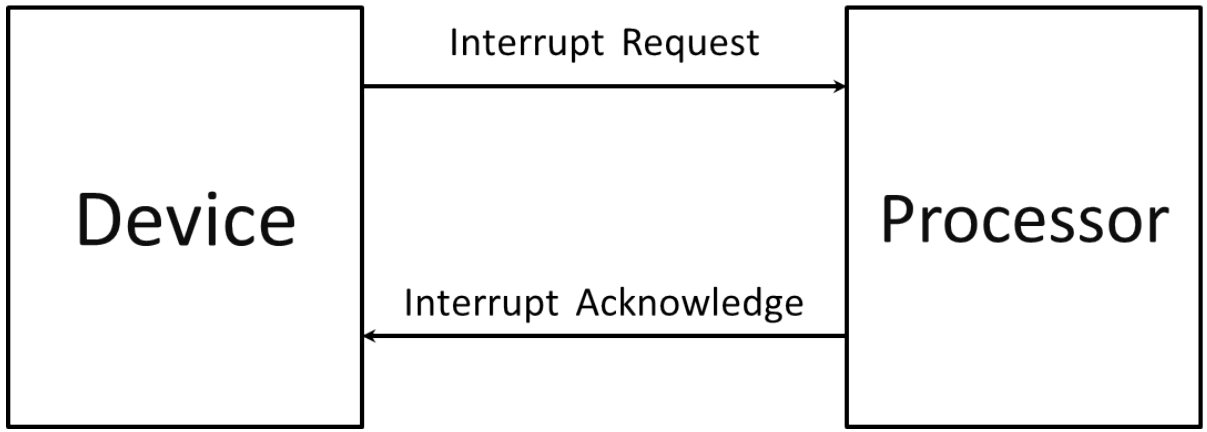
MicroBlaze interrupts

- Single interrupt request line
- Machine Status Register enables/disables interrupts
  - Interrupt Enable (Bit 1)



  - If IE is set to ‘1’ interrupts are enabled
  - If IE is set to ‘0’ interrupts are disabled
- MSR can be updated by using `MTS (move to special purpose register)` instruction
- Register R14 keeps the return address
- Interrupt service routine resides at a fixed memory address, unless an interrupt controller is used

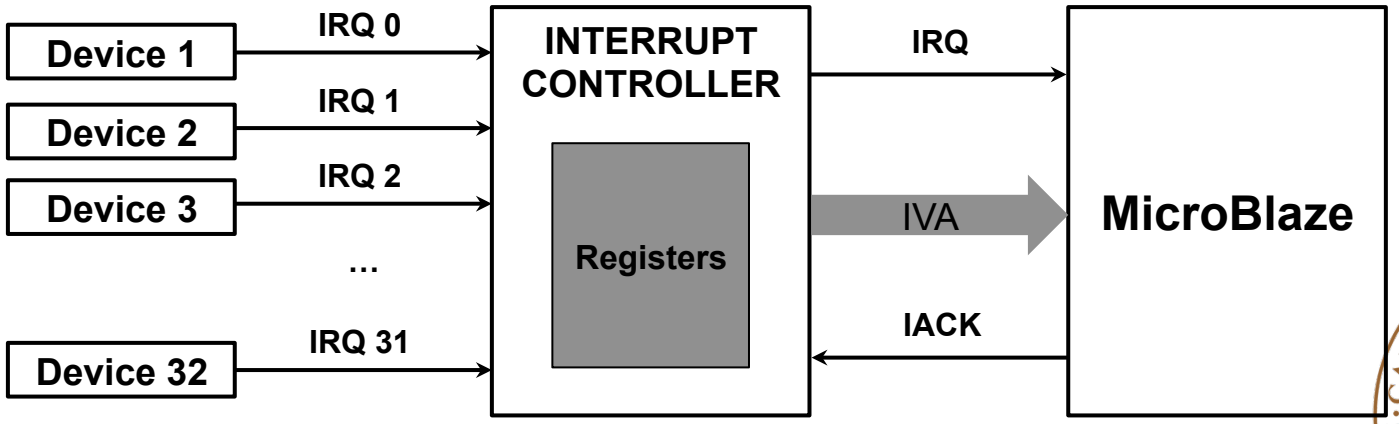
MicroBlaze Interrupts



- How about multiple devices?

Interrupt Controller

- Multiple input interrupt request lines
- Single output interrupt request line
- Interrupt vector address output
  - Memory address of the interrupt service routine
- Set of registers

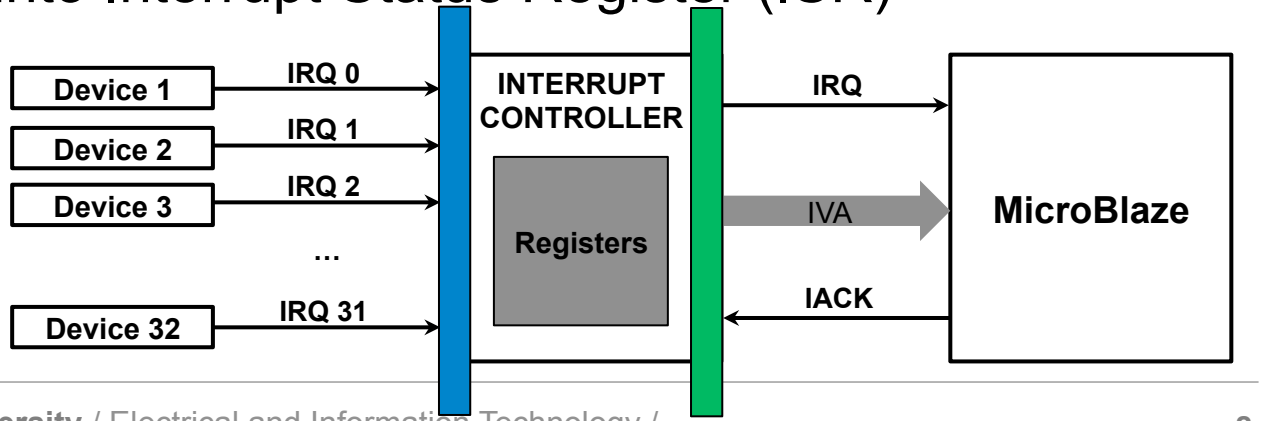


Interrupt Controller- Registers

- Master Enable Register (MER)
- Interrupt Status Register (ISR)
- Interrupt Enable Register (IER)
- Interrupt Pending Register (IPR)
- Interrupt Acknowledge Register (IAR)
- Interrupt Vector Address Registers (IVAR0-IVAR31)

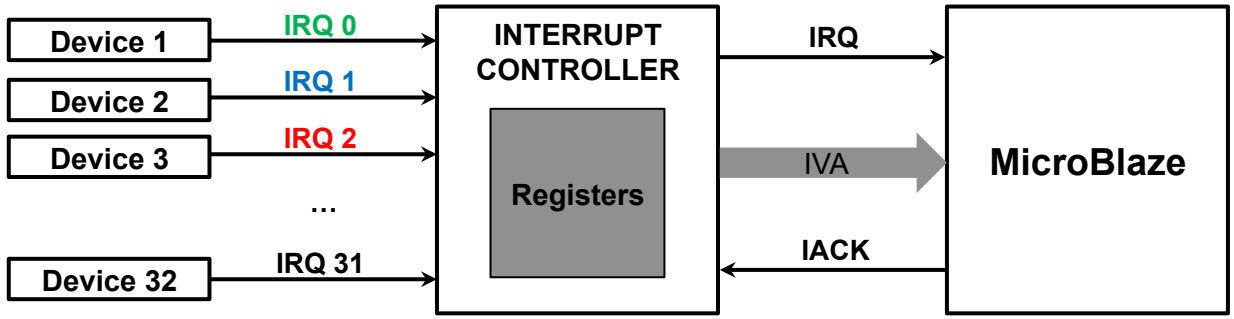
Master Enable Register

- Global Interrupt Enable (GIE) bit
  - '1' Enables the Interrupt Request output
  - '0' Disables the Interrupt Request output
- Hardware Interrupt Enable (HIE) bit
  - '1' captures the state of the Interrupt Request input line into Interrupt Status Register (ISR)



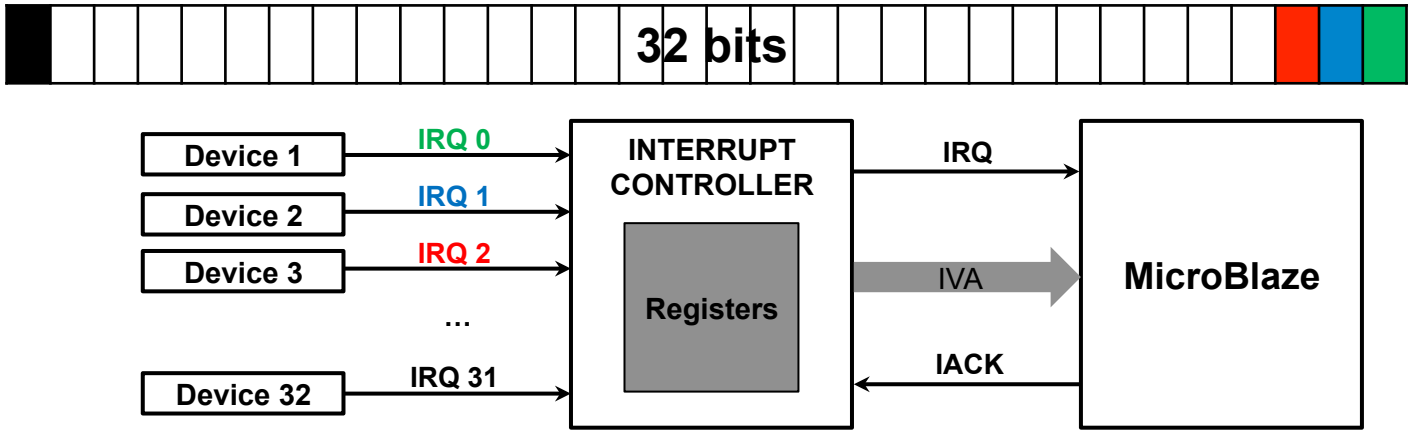
Interrupt Status Register

- When HIE bit in the MER is set to ‘1’, the state of the Interrupt Request input lines are latched in this register



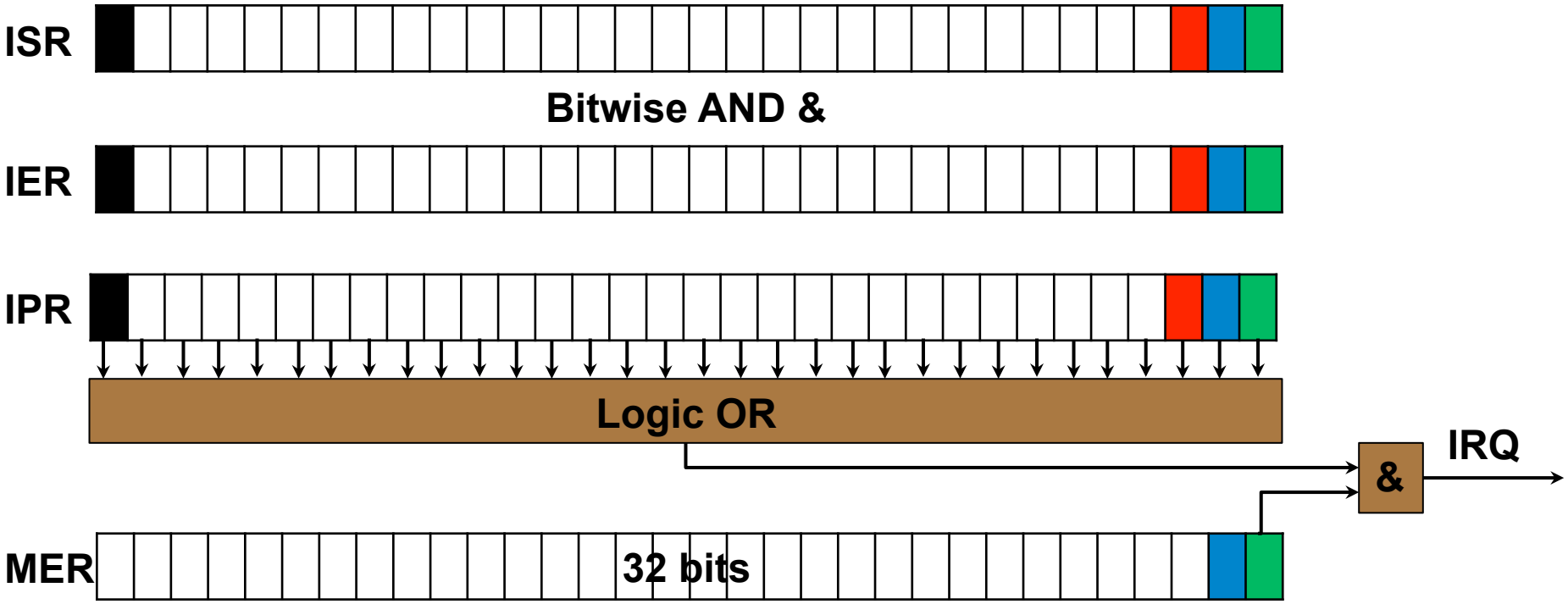
Interrupt Enable Register

- Serves as filter to enable/disable interrupt requests from different devices
  - ‘1’ enables interrupt requests from a particular device
  - ‘0’ disables (masks) interrupt requests



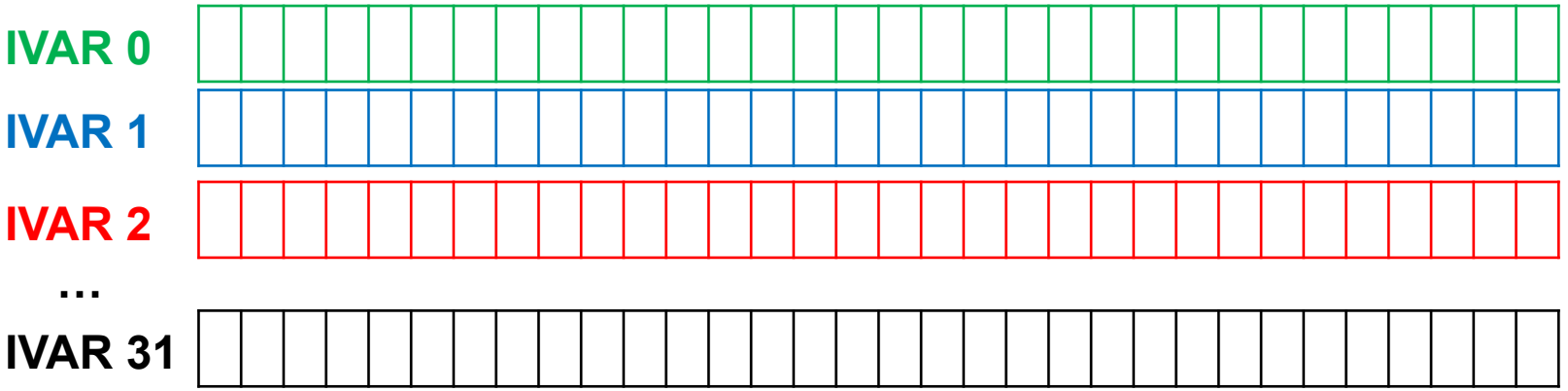
Interrupt Pending Register

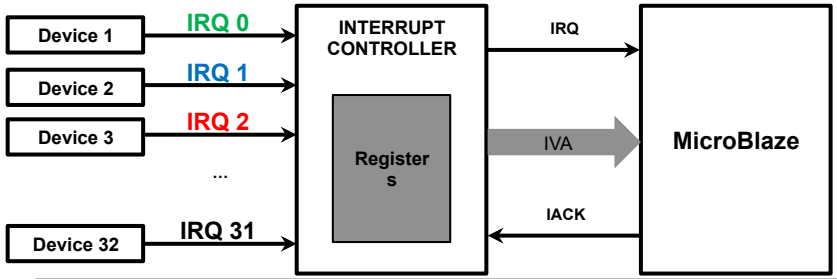
- Indicates presence or absence of active interrupts
- If at least one active interrupt exists and the GIE bit in the MER is set to ‘1’, the output interrupt request line is activated
- Logical AND of the bits in the ISR and the IER



Interrupt Vector Address Registers

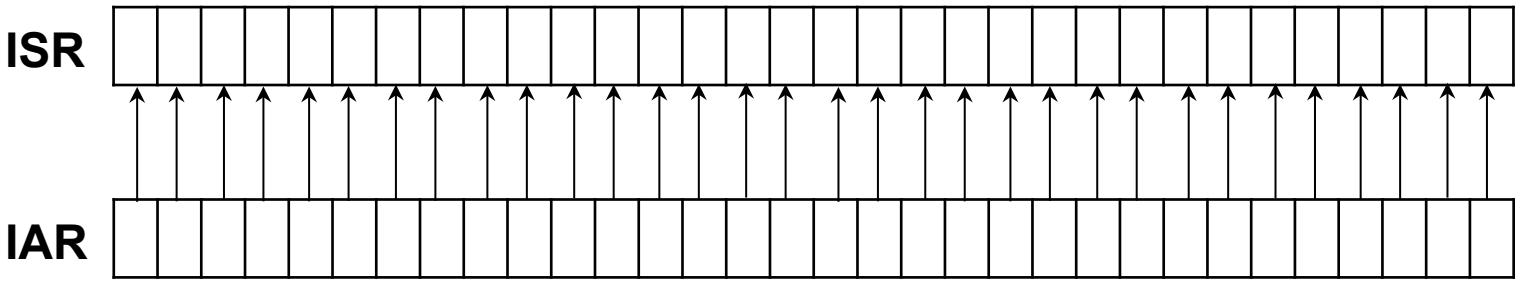
- Thirty two 32-bit registers
- Each register keeps the address of the Interrupt Service Routine for each of the devices connected to the controller



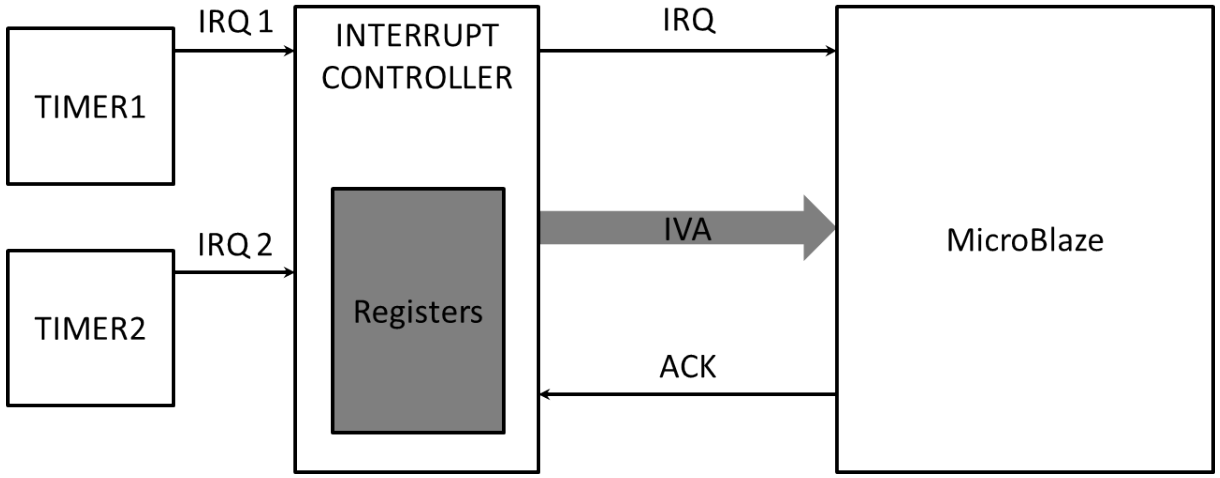


Interrupt Acknowledge Register

- Is used to acknowledge the coming interrupt requests
- Writing a ‘1’ to a particular bit position of this register, clears (sets to zero) the corresponding bit position in the ISR



Interrupt System in the HW platform



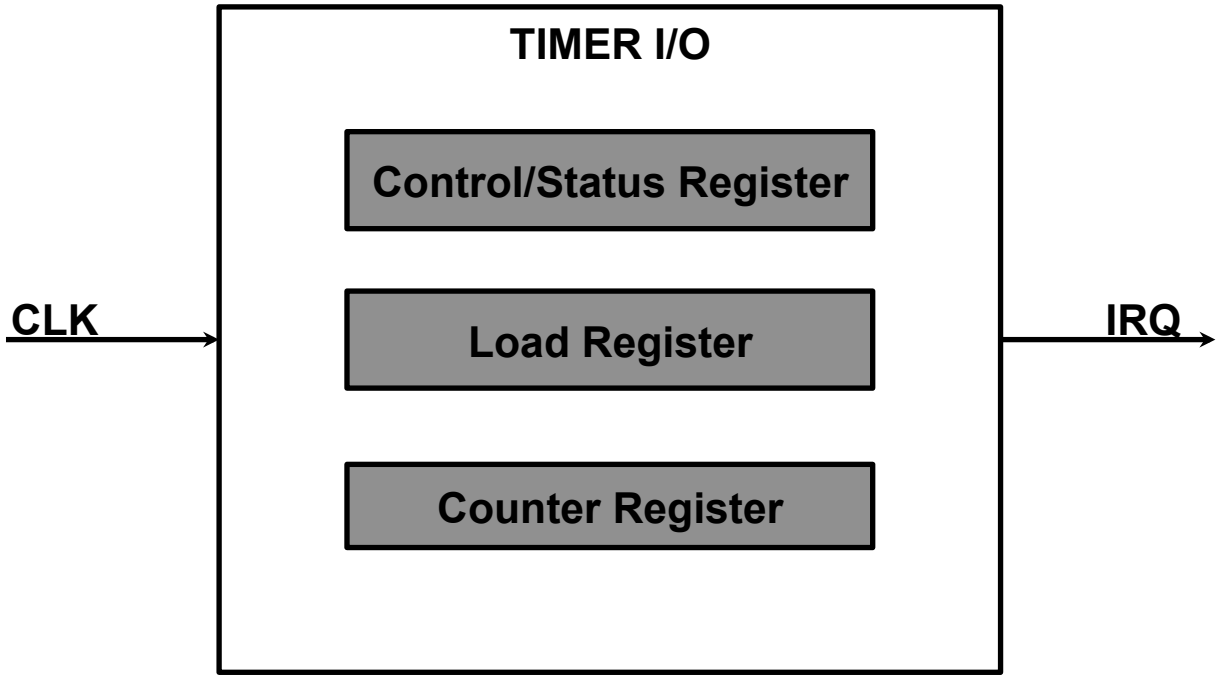
Programming Sequence

1. Write to the IER register such that interrupts from particular devices are enabled
2. Write the Interrupt Vector Address for each of the devices whose interrupts are enabled in IVAR0-IVAR31
3. Write to the MER register

Note: The interrupt service routine needs to acknowledge the interrupts, by writing to the IAR

Timer I/O

- Can generate periodic interrupts
- Can generate an interrupt once a specific time interval (programmed by the user) has elapsed from the moment when the timer is enabled



Timer- Registers

- Counter Register
- Load Register
- Control/Status Register

Counter Register

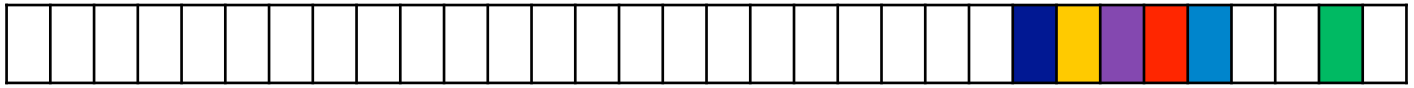
- 32-bit wide register
- When the timer is enabled, each clock cycle, the value of this register is incremented/decremented (the counting direction is controlled by a particular bit the Control/Status Register)
- When the counter overflows, it can generate an interrupt (controlled by a particular bit in the Control/Status Register)
- Upon an overflow, either the counter stops counting or it is reloaded with the value in the Load Register and keeps on counting (controlled by a particular bit in the Control/Status Register)

Load Register

- 32-bit wide register
- Keeps a specific value that is supposed to be used as initial value for the Counter Register when the timer starts running
- Depending on the value of this register, and the counting direction, it is possible to
  - generate periodic interrupts with a known rate, as long as the counter is reloaded upon overflow
  - generate an interrupt after a specific time interval has passed from the moment when the timer was enabled

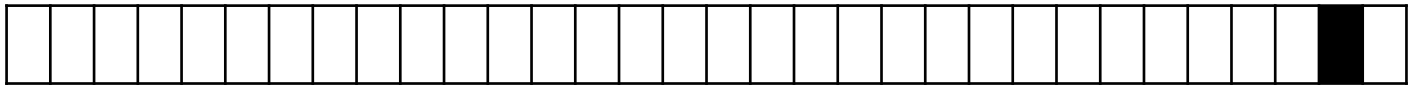
Control/Status Register

- 32-bit wide register



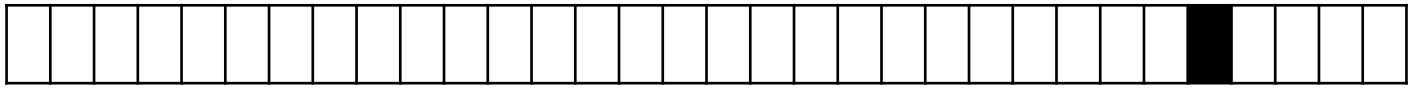
- Up/Down Count (UDC) bit
- Auto Reload/Hold (ARH) bit
- Load (L) bit
- Enable Interrupt (EI) bit
- Enable (E) bit
- Interrupt Flag (IF) bit

Up/Down Count (UDC) bit



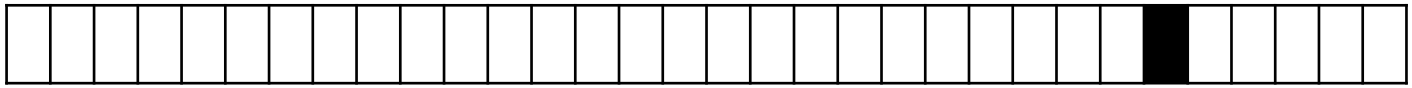
- When set to ‘1’, the counter counts down
- When set to ‘0’, the counter counts up

Auto Reload/Hold (ARH) bit



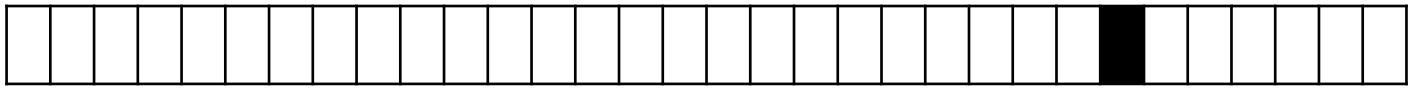
- When set to ‘1’, upon an overflow, the counter is reloaded with the value stored in the Load Register
- When set to ‘0’, upon an overflow, the counter stops counting

Load (L) bit



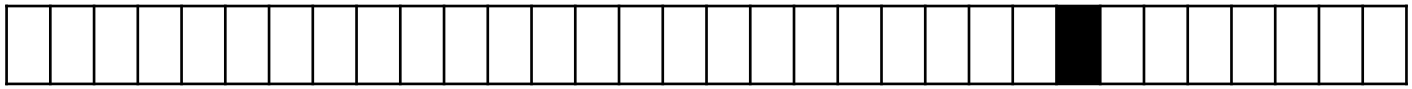
- When set to ‘1’, the counter stops counting, and the value of the Load Register is copied into the Counter Register
- When set to ‘0’, no effect

Enable Interrupt (EI) bit



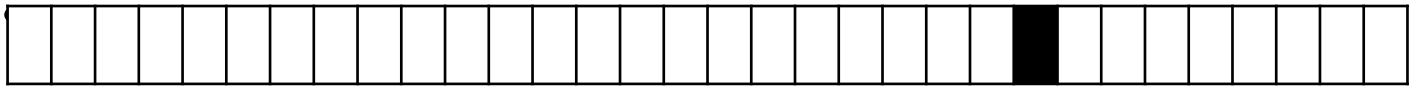
- When set to ‘1’, an overflow in the Counter Register sets the IF bit to ‘1’ (signaling an interrupt event)
- When set to ‘0’, an overflow in the Counter Register does not affect the IF bit

Enable (E) bit



- When set to ‘1’, the timer is enabled and the counter starts counting
- When set to ‘0’, the timer is disabled and counter stops counting
- It is not allowed to set this bit to ‘1’ along with setting the L bit to ‘1’

Interrupt Flag (IF) bit



- Read operation
  - ‘1’ == an interrupt has occurred
  - ‘0’ == no interrupt has occurred
- Write operation
  - ‘1’ == clears the IF bit (acknowledge)
  - ‘0’ == no effect

Programming Sequence

1. Write to the Load Register
2. Write to the Control/Status register to load the value of the Load Register into the Counter Register and set corresponding operation (up/down count, auto reload/hold, enable/disable interrupts)
3. Write to the Control/Status register to enable the timer (set the E bit and clear the L bit)

Note: The interrupt service routine needs to clear the IF bit in the Control/Status Register

LUNDS
UNIVERSITET