DEVICE DRIVERS AND INTERRUPTS SERVICE MECHANISM

Lesson-8: Masking of Interrupt Sources and Interrupt Status or Pending Mechanism

1. Masking of Interrupt Sources

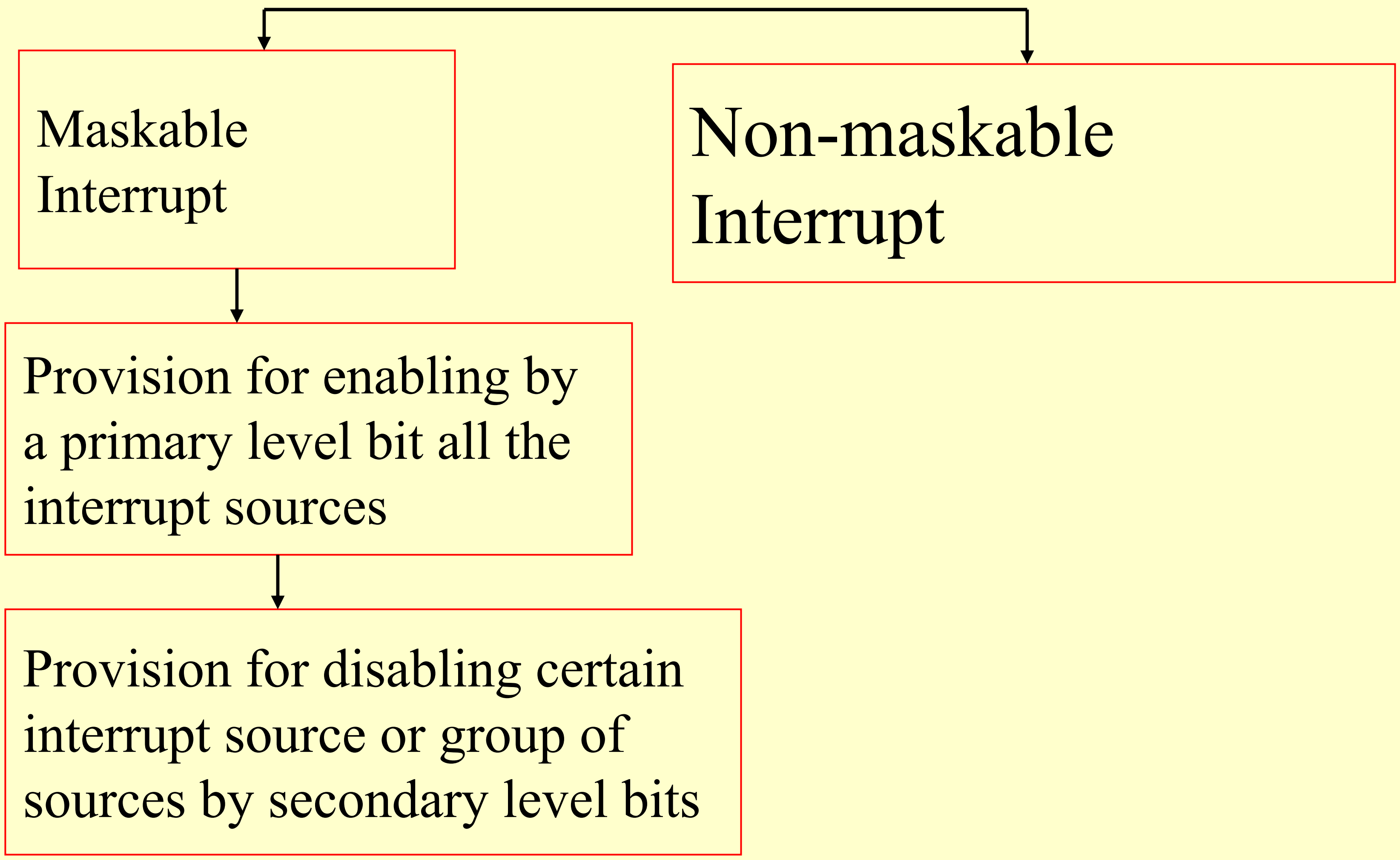
Maskable Interrupt Sources

- Maskable sources of interrupt provides for masking and unmasking the interrupt service (diversion to the ISR).
- Execution of a device interrupt source or source group can be masked.
- On a pin, an external interrupt request can be masked.
- Execution of a software interrupt (trap or exception or signal) can be masked.
- Most interrupt sources are maskable.

Non-maskable Interrupt Sources (NMIs)

- A few specific interrupt cannot be masked.
- A few specific interrupt can be declared non-maskable within few clock cycle of processor reset, else that is maskable.

Interrupt Source Types


Maskable Interrupt

Non-maskable Interrupt

Provision for enabling by a primary level bit all the interrupt sources

Provision for disabling certain interrupt source or group of sources by secondary level bits

Classification of all interrupts as Non Maskable and Maskable Interrupts

- Nonmaskable— Examples are RAM parity error in a PC and error interrupts like division by zero. These must be serviced.
- Maskable: Maskable interrupts are those for which the service may be temporarily disabled to let higher priority ISRs be executed uninterruptedly
- Nonmaskable only when defined so within few clock cycles after reset: Certain processors like 68HC11 have this provision

Enabling (Unmasking) and Disabling (Masking) in case of Maskable Interrupt Sources

- Interrupt control bits in devices
- One bit EA (enable all)─ also called the primary level bit
- EA may be for enabling or disabling the complete interrupt system except for NMIs.

Use of DI and EI for a Critical section of codes

- Instruction DI (disable interrupts) is executed at the beginning of the critical section when a routine or ISR is executing in the codes in a critical section which must complete, Another instruction EI (enable interrupts) is executed at the end of the critical section.
- DI instruction resets EA bit and EI instruction sets the EA bit.

Examples of Critical Section codes

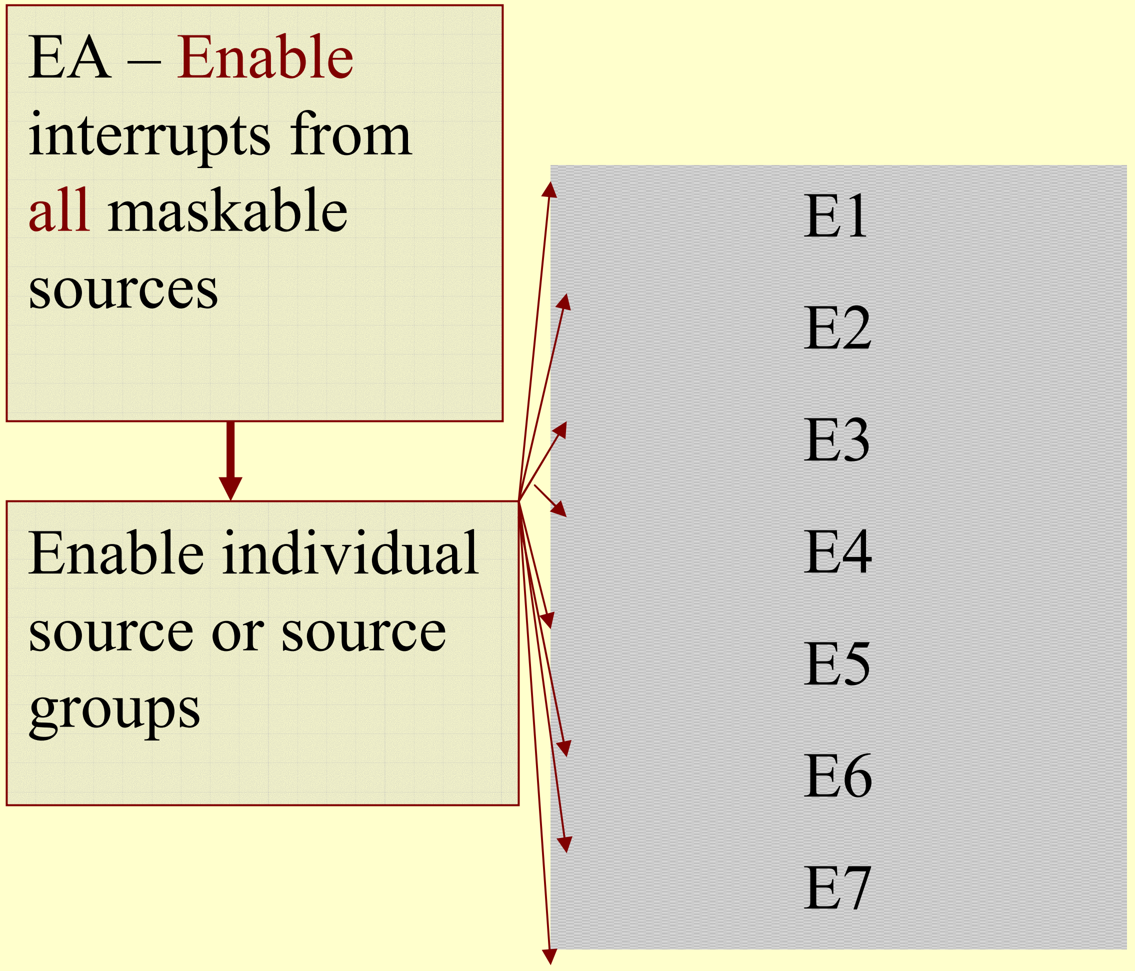
- Synchronous communication of bits from a port
- Writing some byte(s), for example, time, which are shared with other routines. In case the bytes for time are not written at that instance the system timing reference will be incorrect

Example of critical section codes

- Assume that an ISR is transferring data to the printer buffer, which is common to multiple ISRs and functions. No other ISR of function should transfer the data at that instant to the print-buffer, else the bytes at the buffer shall be from multiple sources.
- A data shared by several ISRs and routines needs to be generated or used by protecting its modification by another ISR or routine.

Multiple bits E0, .. En

- For *n* source group of the interrupts in case of multiple devices present in the system and externally connected to the system
- Called mask bits and also called the secondary level bits
- For enabling or disabling the specific sources or source groups in the interrupting system.
- By the appropriate instructions in the user software, write to the primary enable bit and secondary level enable bits (or the opposite of it, mask bits) bits, either all or a part of the total maskable interrupt sources, respectively disabled


EA – Enable interrupts from all maskable sources
Enable individual source or source groups
E1
E2
E3
E4
E5
E6
E7

Secondary and primary level Control bits for Interrupts

Example

- Two timers and each timer has an interrupt control bit.
- Timer interrupt control bits ET0 and ET1
- SI device— interrupt control bit ES, common to serial transmission and serial reception. T
- EA bit to interrupt control for disabling all interrupts.
- When EA = 0, then no interrupt is recognized and timers as well as SI interrupts service is disabled.
- When EA = 1, ET0 = 0, ET1 = 1 and ES = 1, then interrupts from timer 1 and SI enabled and timer 0 interrupt is disabled

2. Interrupt Status or Pending Register

Multiple interrupt sources

- Multiple interrupt sources, an occurrence of each interrupt source (or source-group) is identifiable from a bit or bits in the status register and/or in the IPR

Properties of the interrupt identification flags

- A separate flag each for every identification of an occurrence from each of the interrupt sources
- The flag sets on the an occurrence of an interrupt,

The flag is present either

- in the internal hardware circuit of the processor or
- in the IPR or
- in status register.

Identification of a previously occurred interrupt from a source

- A local level flag (bit) in a status register which can hold one or more status-flags for the one or several of the interrupt sources or groups of the sources
- A processor pending-flag (Boolean variable) in an interrupt-pending register (IPR), and which sets by the source (setting by hardware) and auto-resets immediately by the internal hardware as soon as at a later instant, the corresponding source service starts on diversion to the corresponding ISR

Example

- Two timers and each timer has a status bit TF0 and TF1
- SI device there are two status bits TxEMPTY and RxReady for serial transmission completed and receiver data ready
- The ISR_T1 corresponding to timer 1 device— reads the status bit TF1 = 1 in status register to find that timer 1 has overflowed, as soon as the bit is read the TF1 resets to 0

Example

- The ISR_T0 corresponding to timer 0 device reads the status bit TF1 = 0 in status register to find that timer 0 has overflowed, as soon as the bit is read the TF0 resets to 0.
- The ISR corresponding to SI device common for transmitter and receiver.
- The ISR reads the status bits TxEMPTY and RxReady in status register to find whether new byte is to be sent to transmit buffer or whether byte is to be read from receiver buffer.
- As soon as byte is read the RxReady resets and as soon as byte is written into SI for transmission, TxEMPTY resets

Flag set on occurrence of an interrupt

- Used for a *read* by an instruction and for a *write* by the interrupting source hardware only.
- Flag *resets* (becomes inactive) as soon as it is *read*.
- An auto reset characteristics provided for in certain hardware designs in order to let this flag indicate the next occurrence from the same interrupt source.

Interrupt Service on flag setting

- If flag is set, it does not necessarily mean that it will be recognized or serviced later.
- Whenever a mask bit corresponding to its source exists, and then even if the flag sets, the processor may ignore it unless the mask (or enable) bit modifies later.
- Masking prevents an unwanted interrupt from being serviced by a diversion to another ISR

Example of touch screen

- Touch screen device processor generates an interrupt when a screen position is touched.
- A status bit is also set.
- It activates an external pin interrupt request IRQ.
- From the status bit, which is set, the interrupting source is recognized among the sources group (multiple sources of interrupt from same device or devices).
- The ISR_VECTORIRQ and ISRIRQ common for all the interrupts at IRQ pin.

IRQ

- IRQ results in processor (controller processing element) vectoring to an ISR_VECTORIRQ.
- Using ISR_VECTORIRQ when the ISRIRQ starts, ISRIRQ instruction reads the status register and discovers the bit as set.
- Calls for service function (get_touch_position), which reads register R_{pos} for touched screen position information.
- This action also resets the status bit when the touch screen processor provides for auto resetting of the bit soon as the R_{pos} byte for touched position is read.

Summary

We learnt

- Interrupt occurrence is identified by setting a flag (=1) the system from bit at status register or pending register
- Flag is a Boolean─ set (=1) or reset (=0).
- Flag generally auto reset when ISR starts for the interrupt
- Touch screen example

End of Lesson 8 of Chapter 4