CSCE 4114
Timers

David Andrews
dandrews@uark.edu

Timers and Counters

- Timers and counters are very similar
  - Timer := a register that increments/decrements with a periodic trigger. Trigger can be:
    - System Clock
    - External periodic Signal
  - Counter := a register that increments with a trigger that can be asynchronous or non-periodic. Can also be viewed as timer if trigger is synchronous.

Timer Usage

- Most Embedded Systems have multiple timers used for different things.
- Operating Systems Scheduling
  - Periodic Scheduling: When timer interrupt goes off, the scheduler is invoked to check for new decision.
    -What is this periodic timer called in Linux ?
  - "One-Shot": Operating system sets a new point in time for evaluating the scheduling decision.
  - Periodic scheduling is the norm in non-real time (or embedded systems) but not in real-time systems. Why ?

Timer Usage

- Operating Systems Monitoring: Suppose a character is sent to the keyboard or across network and the system waits on a response.
  - What happens if the keyboard or network hangs ?
  - "Watchdog Timer": A drop dead time that if no response is received, the timer generates an interrupt to kick OS into error processing.
    - If all is well, when response is received, OS "cancels" the watchdog timer by resetting

Timer Usage

- Operating System:
  - Performance meters. OS uses timers to measure performance, system evaluation, tuning.
  - DRAM refresh. Must periodically read all DRAM memory locations and re-write to maintain data.
  - And of course…..keeping time of day……

Timer Usage

- Applications use timers for co-ordinating activities such as timed input/outputs.
  - After detecting a car coming to an intersection, turn the pedestrian signal red and wait 20 seconds, then turn pedestrian signal green.
  - Controlling a diamond cutter: Once cut is started, continue cutting for 15 msec's then stop !

Hw/Sw Timers

- Systems contain multiple, but not great numbers of hardware timers.
- Multiple "software" timers can be provided that use a single hardware timer.
- -Multiple "one-shot" requests can be maintained as a linked list. Software sorts requests into time ordering and programs the hardware timer accordingly

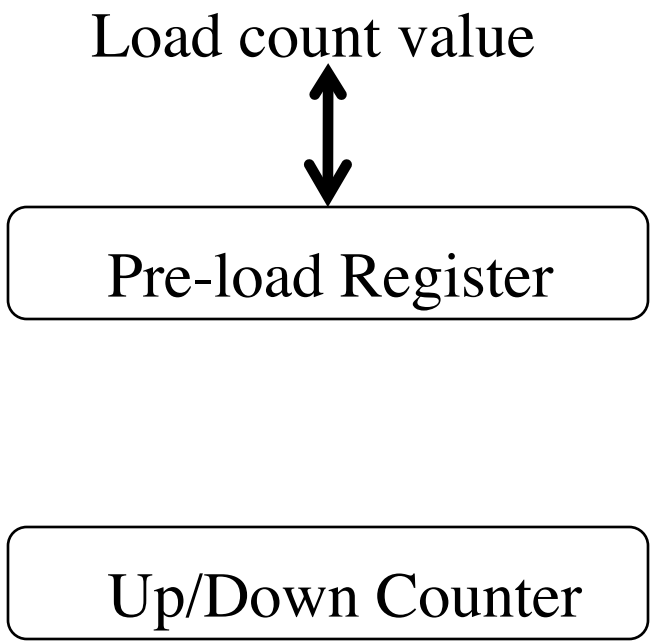
Hardware Timers

Pre-load Register

Up/Down Counter

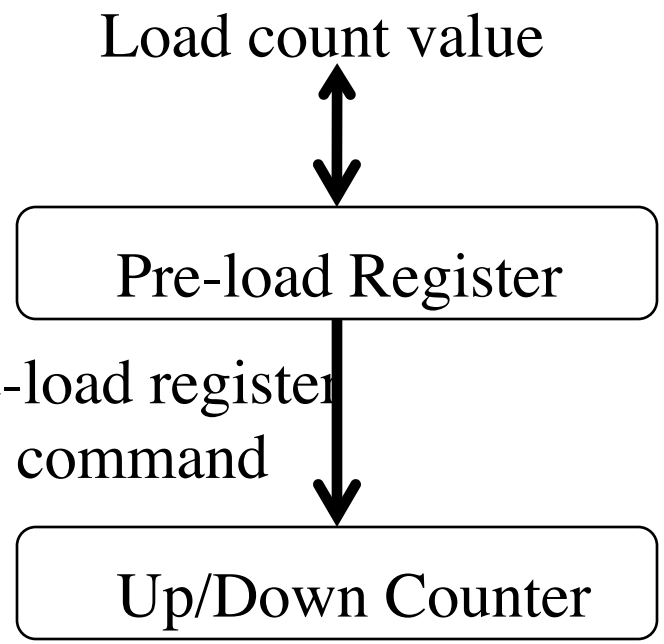
Hardware Timers

Load initial value into a pre-load register

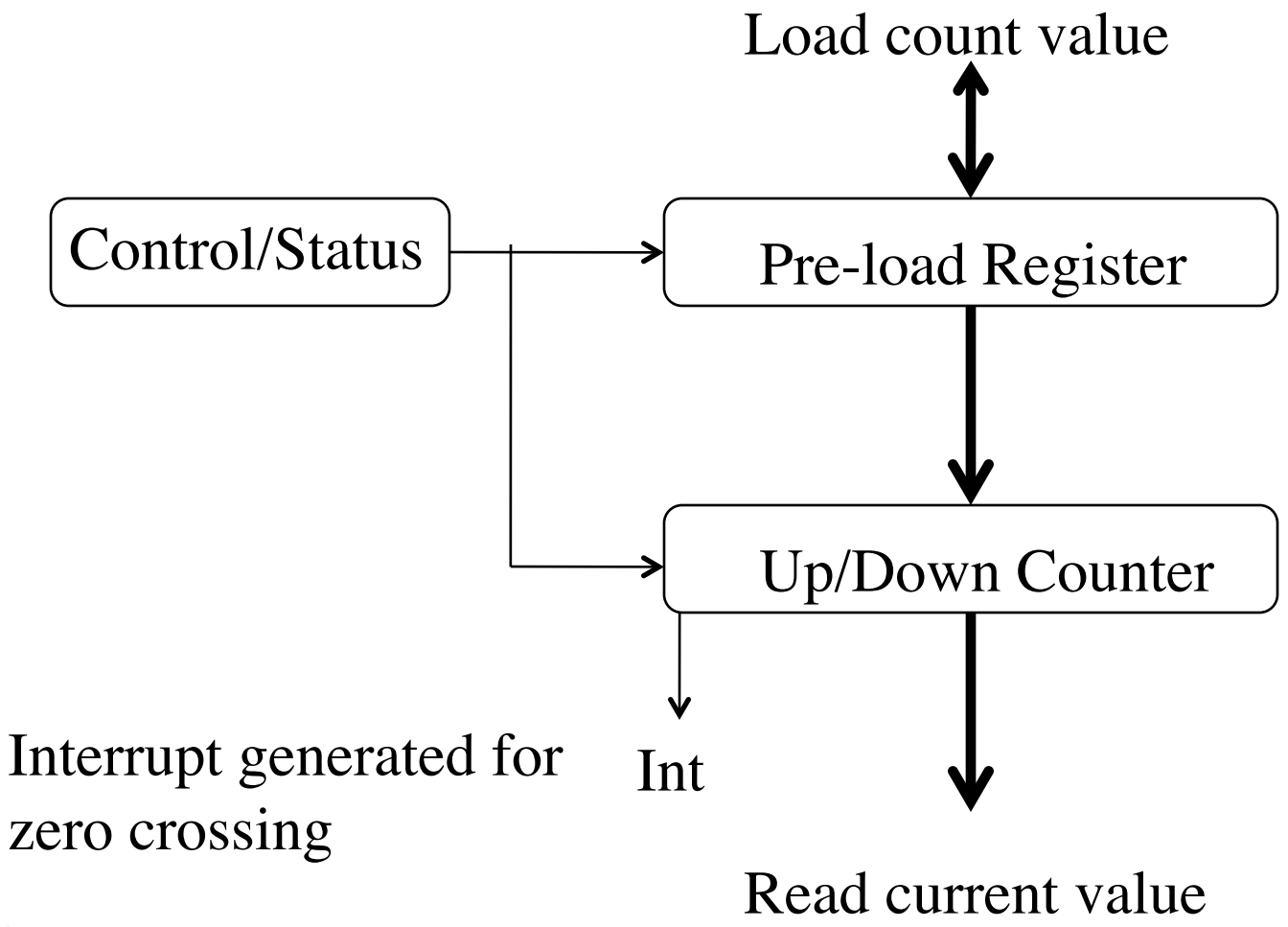


Hardware Timers

Load initial value into
a pre-load register
-counter loaded from pre-load register
:on separate “go” command
:automatically



Hardware Timers



Hardware Timers

Control/Status

Control: bits to allow configuration

Hardware Timers

Control/Status

Control: bits to allow configuration

-periodic: on zero crossing counter automatically loaded from pre-load immediately

Hardware Timers

Control/Status

Control: bits to allow configuration
-periodic: on zero crossing counter automatically loaded from pre-load immediately
-one-shot: on zero crossing, counter not automatically loaded
-start/stop: self explanatory

Hardware Timers

Control/Status

Control: bits to allow configuration
-periodic: on zero crossing counter automatically loaded from pre-load immediately
-one-shot: on zero crossing, counter not automatically loaded
-start/stop: self explanatory
-interrupt Enable/Disable: self explanatory

Xilinx Timer/Counter

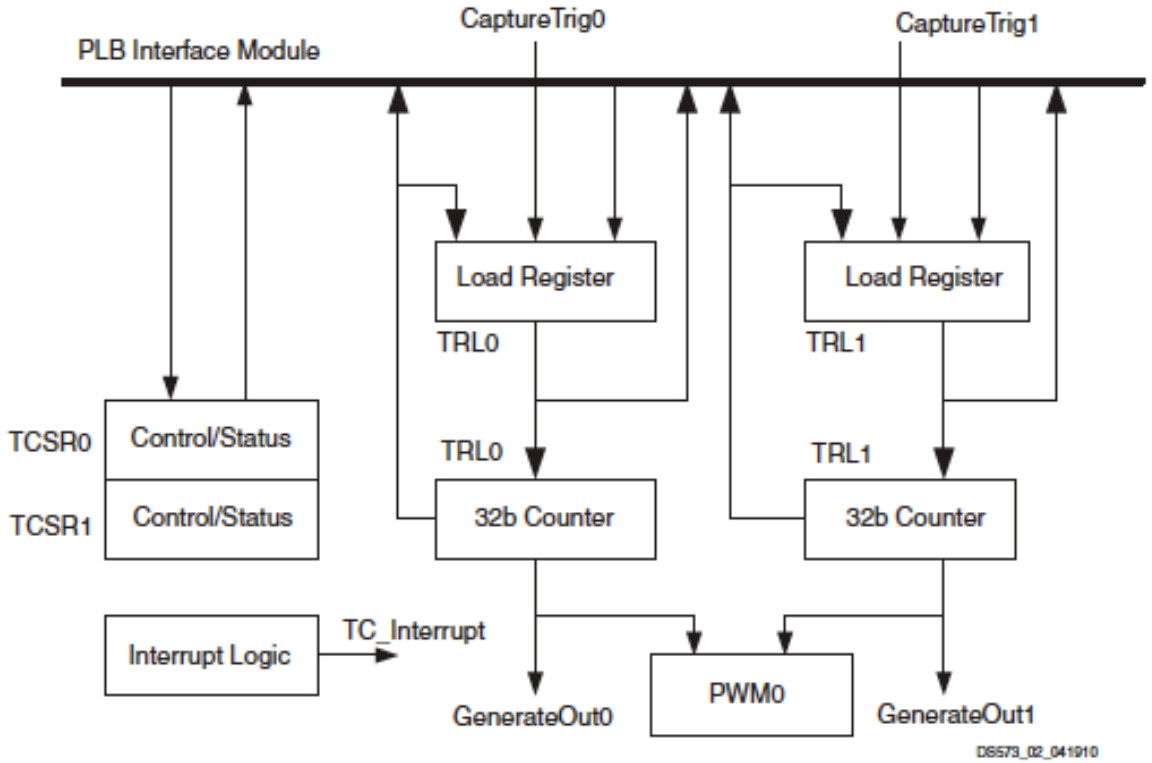


Figure 2: **XPS Timer/Counter Detailed Block Diagram**

Xilinx Timer/Counter

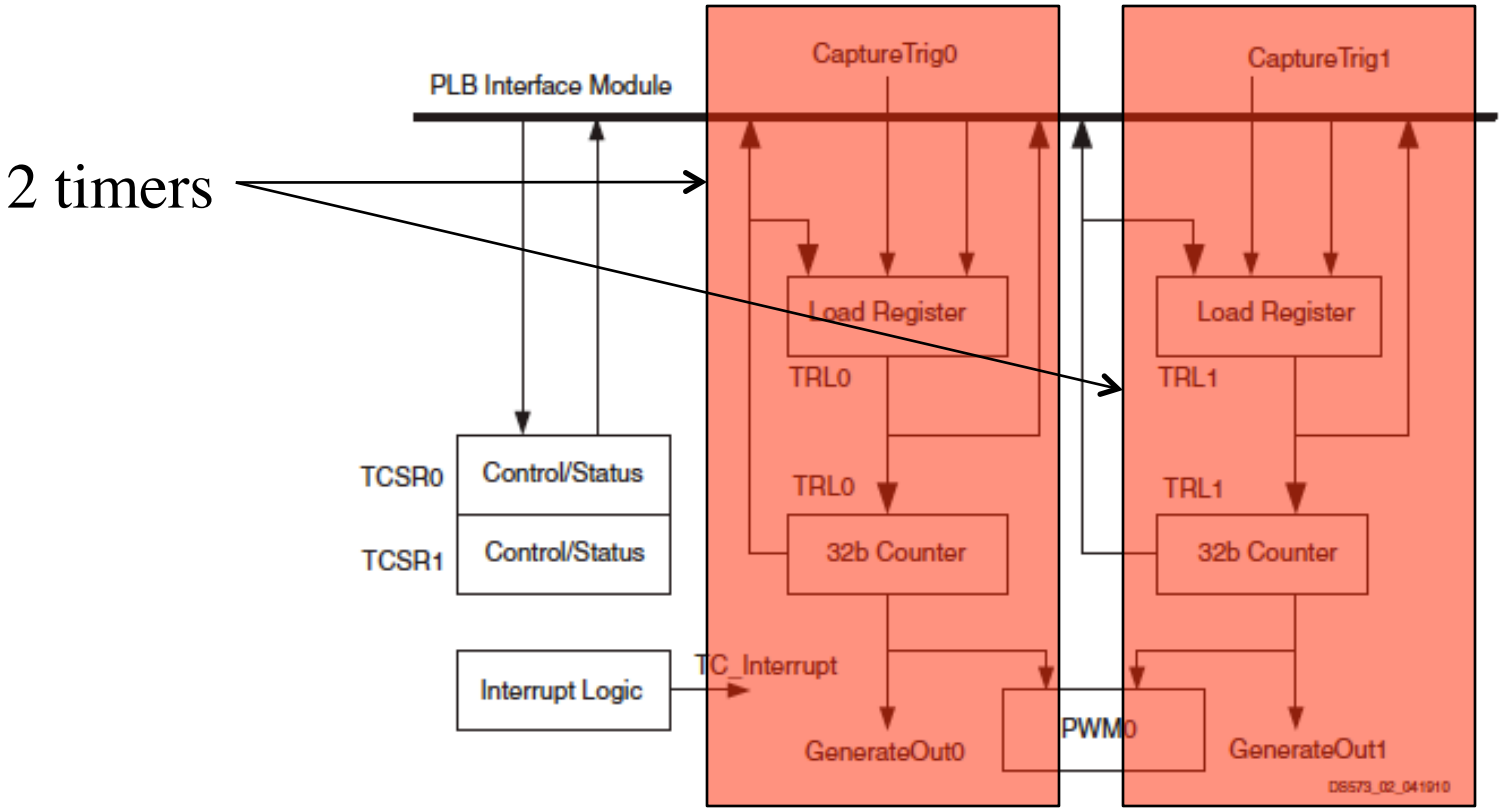


Figure 2: **XPS Timer/Counter Detailed Block Diagram**

Xilinx Timer/Counter

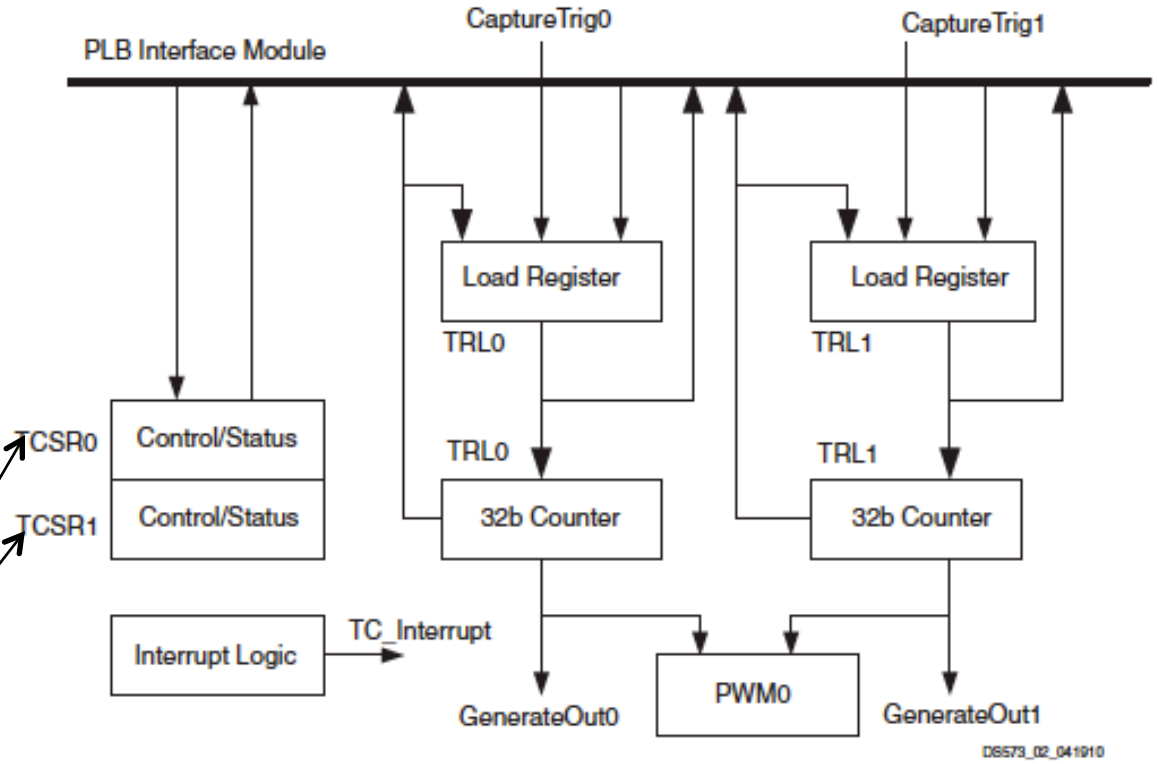


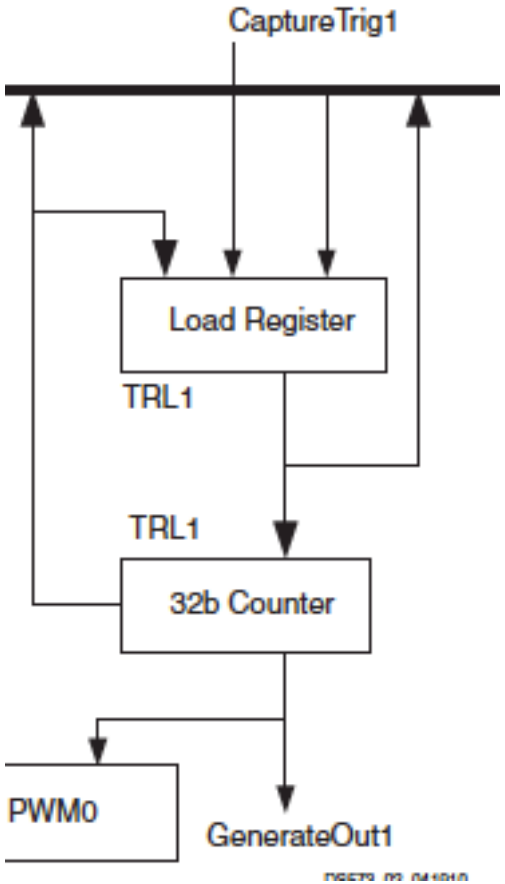
Figure 2: **XPS Timer/Counter Detailed Block Diagram**

Control/status registers

For each individual timer

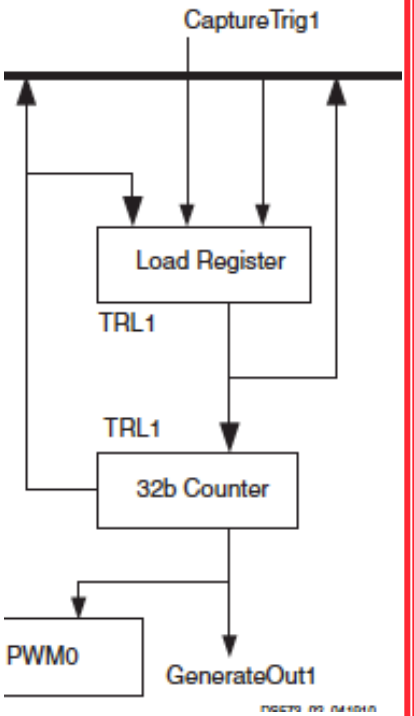
Each Timer has 3 Modes

- Generate Mode
- Capture Mode
- Pulse Width Modulation



Generate Mode

- Load register -> counter (load_bit=1)
  - Note* Must be cleared before counter is enabled
- Set counter up or down (UDT bit)
- Start Counter
- On zero crossing, stop or automatically reload (ARHT bit)
- Generate 1 cycle interrupt (TINT bit)



Control/Status Register

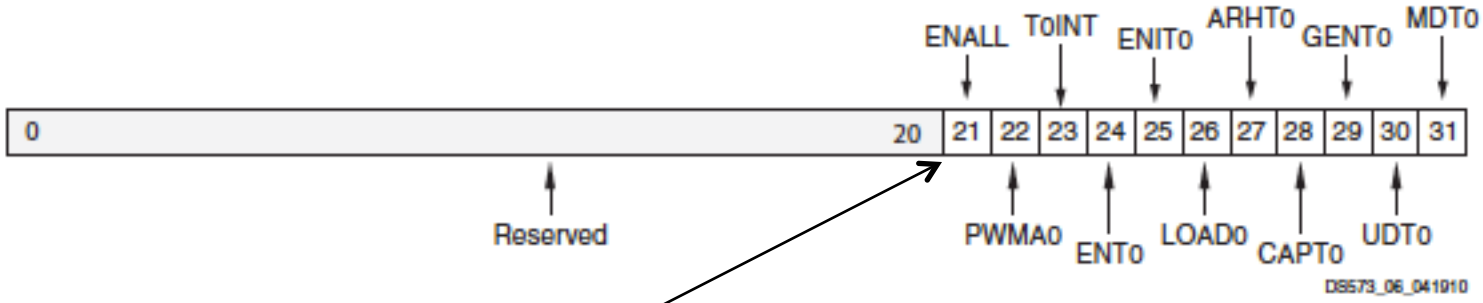


Figure 6: **Timer Control/Status Register 0 (TCSR0)**

21	ENALL	**Enable All Timers** 0 = No effect on timers 1 = Enable all timers (counters run) This bit is mirrored in all control/status registers and is used to enable all counters simultaneously. Writing a '1' to this bit sets ENALL, ENT0, and ENT1. Writing a '0' to this register clears ENALL but has no effect on ENT0 and ENT1.	0

This bit is mirrored in both Control/Status Regs

Control/Status Register

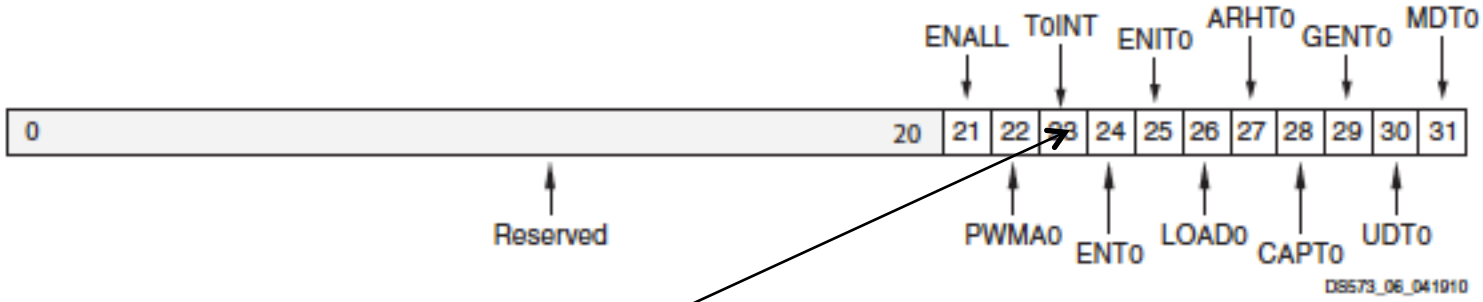


Figure 6: **Timer Control/Status Register 0 (TCSR0)**

23	T0INT	**Timer0 Interrupt** Indicates that the condition for an interrupt on this timer has occurred. If the timer mode is capture and the timer is enabled, this bit indicates a capture has occurred. If the mode is generate, this bit indicates the counter has rolled over. Must be cleared by writing a '1'. *Read:* 0 = No interrupt has occurred 1 = Interrupt has occurred *Write:* 0 = No change in state of T0INT 1 = Clear T0INT (clear to '0')	0

Control/Status Register

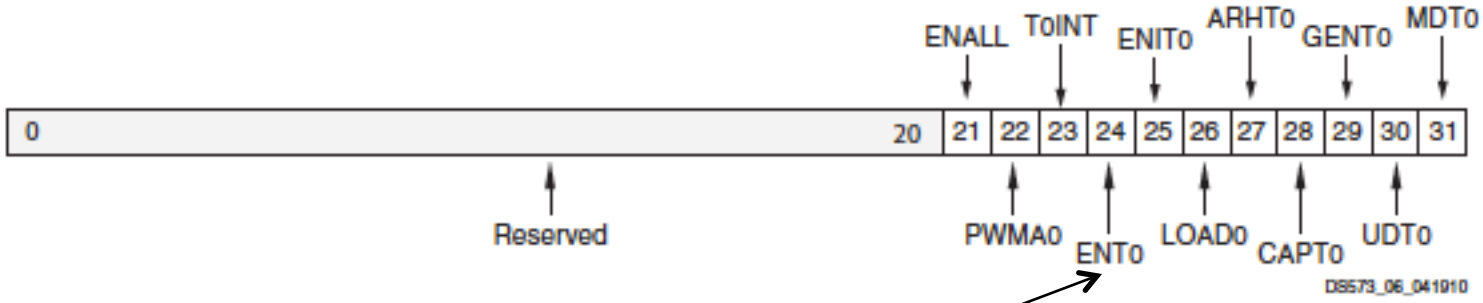


Figure 6: **Timer Control/Status Register 0 (TCSR0)**

24	ENT0	**Enable Timer0** 0 = Disable timer (counter halts) 1 = Enable timer (counter runs)	0

Control/Status Register

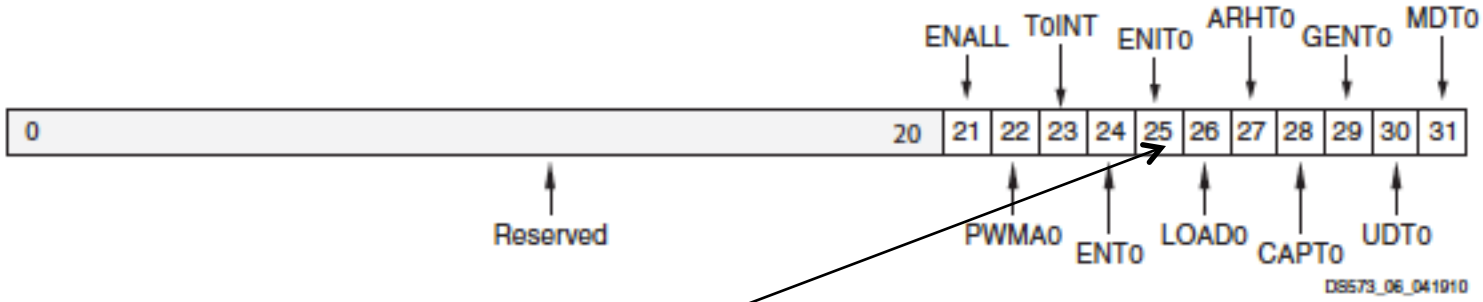


Figure 6: **Timer Control/Status Register 0 (TCSR0)**

25	ENIT0	**Enable Interrupt for Timer0** Enables the assertion of the interrupt signal for this timer. Has no effect on the interrupt flag in TCSR0. 0 = Disable interrupt signal 1 = Enable interrupt signal	0

Control/Status Register

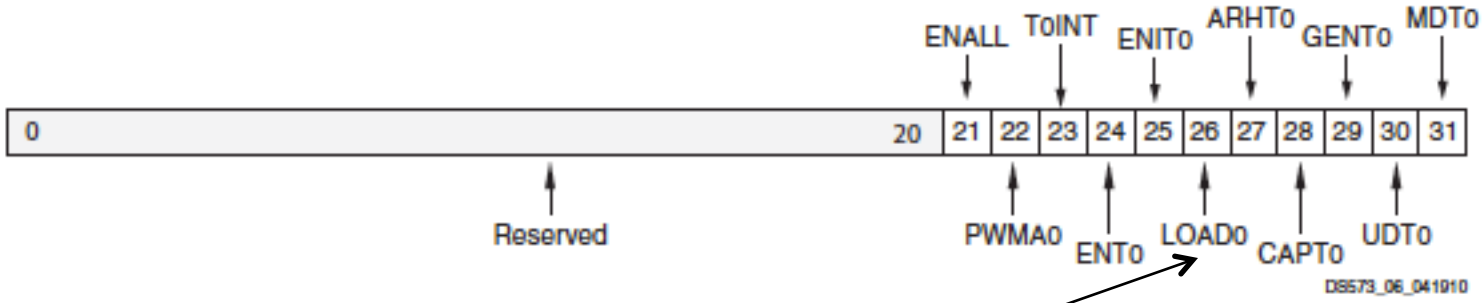


Figure 6: **Timer Control/Status Register 0 (TCSR0)**

26	LOAD0	**Load Timer0** 0 = No load 1 = Loads timer with value in TLR0	0

Control/Status Register

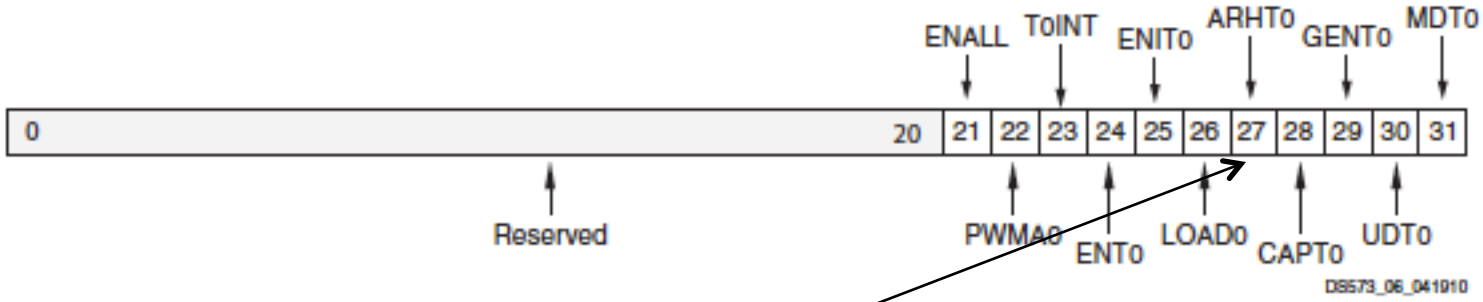


Figure 6: **Timer Control/Status Register 0 (TCSR0)**

27	ARHT0	**Auto Reload/Hold Timer0** When the timer is in Generate Mode, this bit determines whether the counter reloads the generate value and continues running or holds at the termination value. In Capture Mode, this bit determines whether a new capture trigger overwrites the previous captured value or if the previous value is held. 0 = Hold counter or capture value 1 = Reload generate value or overwrite capture value	0

Control/Status Register

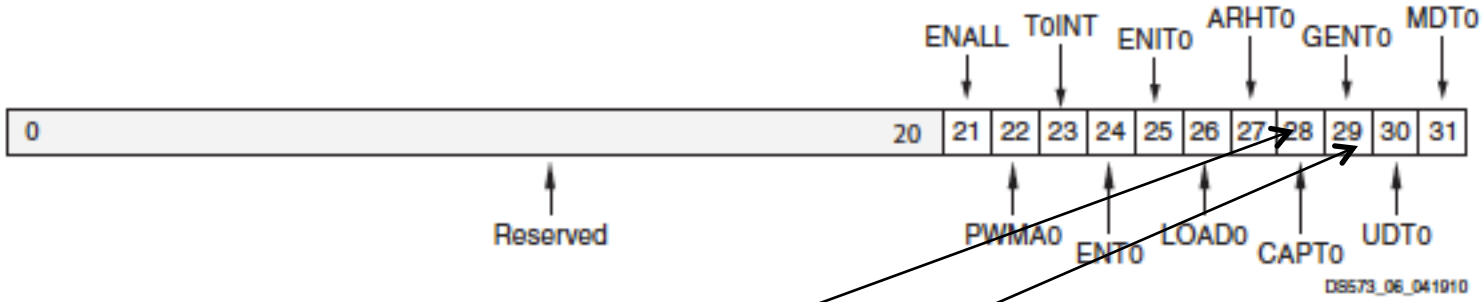


Figure 6: **Timer Control/Status Register 0 (TCSR0)**

28	CAPT0	Enable External Capture Trigger Timer0 0 = Disables external capture trigger 1 = Enables external capture trigger	0
29	GENT0	Enable External Generate Signal Timer0 0 = Disables external generate signal 1 = Enables external generate signal	0

Control/Status Register

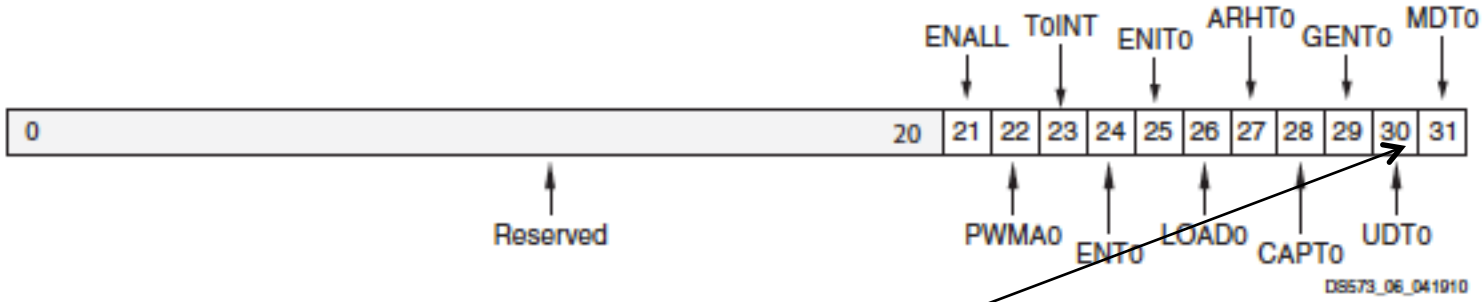


Figure 6: **Timer Control/Status Register 0 (TCSR0)**

30	UDT0	Up/Down Count Timer0 0 = Timer functions as up counter 1 = Timer functions as down counter	0

Control/Status Register

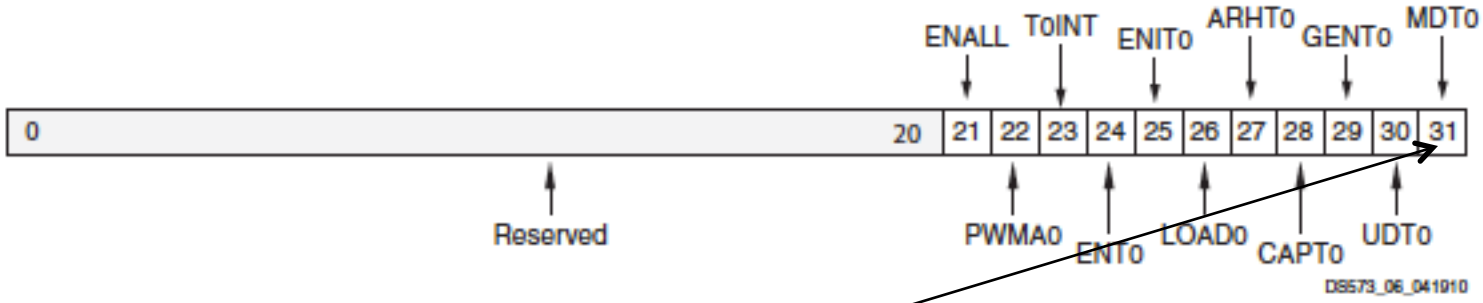


Figure 6: **Timer Control/Status Register 0 (TCSR0)**

31	MDT0	Timer0 Mode See the Timer Modes section. 0 = Timer mode is generate 1 = Timer mode is capture	0

Control/Status Register

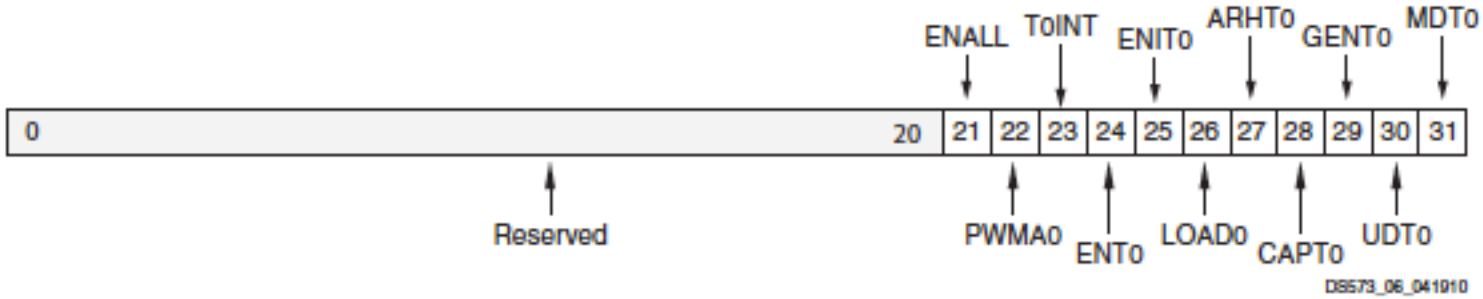


Figure 6: **Timer Control/Status Register 0 (TCSR0)**

What code word do you write to set up timer0 as without actually starting timer ?

1. Periodic
2. Generate interrupt
3. Count down

Control/Status Register

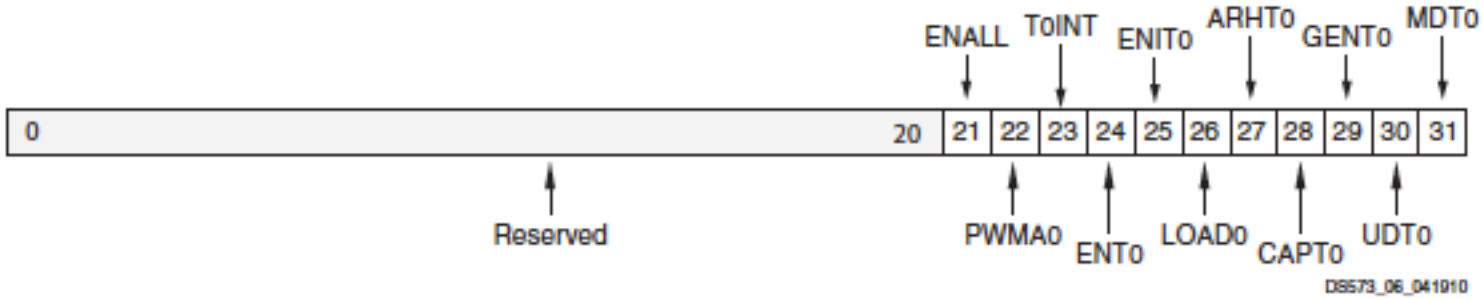


Figure 6: **Timer Control/Status Register 0 (TCSR0)**

What code word do you write to set up and load timer0 as actually starting timer for the following?

1. Periodic
2. Generate interrupt
3. Count down

0 0 [1] [0] 1 [1] [1] 0 0 [0] [0]

Control/Status Register

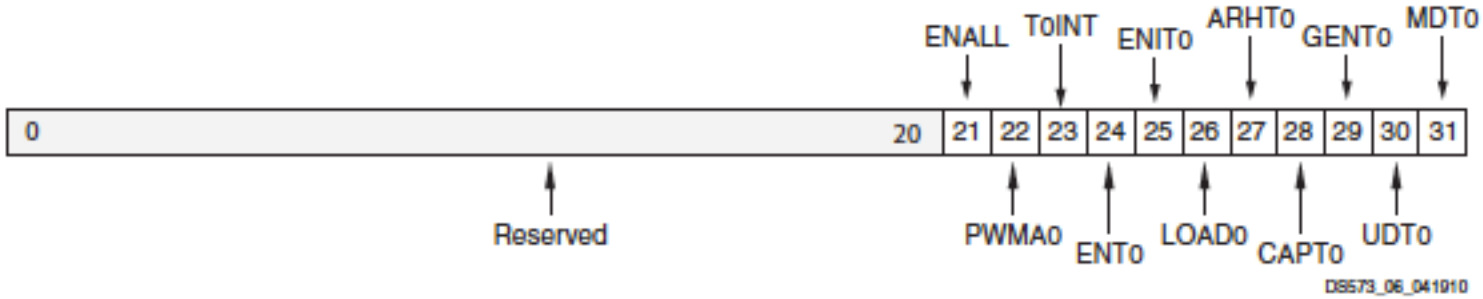


Figure 6: **Timer Control/Status Register 0 (TCSR0)**

What code word do you write to set up and load timer0 as actually starting timer for the following?

1. Periodic
2. Generate interrupt
3. Count down

Clears any left over interrupts

0 0 [1] [0] 1 [1] [1] 0 0 [0] [0]

Control/Status Register

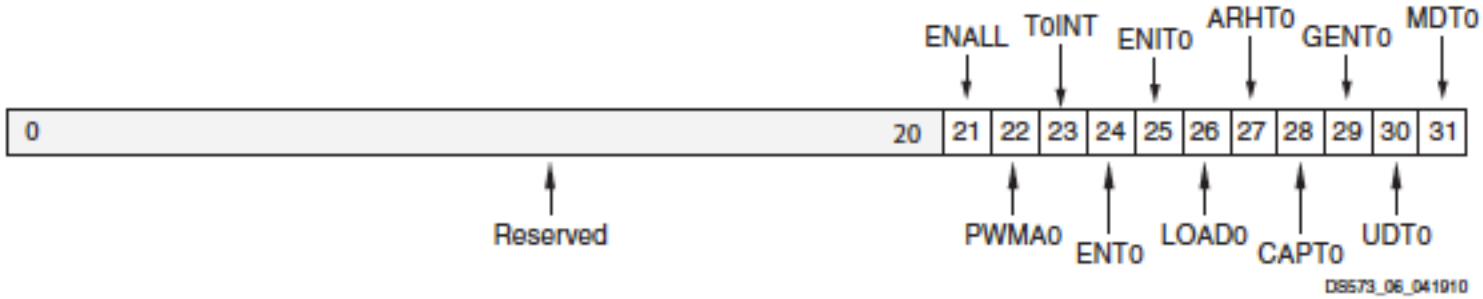


Figure 6: **Timer Control/Status Register 0 (TCSR0)**

What code word do you write to set up and load timer0 as actually starting timer for the following?

1. Periodic
2. Generate interrupt
3. Count down

Don't start yet !

0 0 [1] [0] 1 [1] [1] 0 0 [0] [0]

Control/Status Register

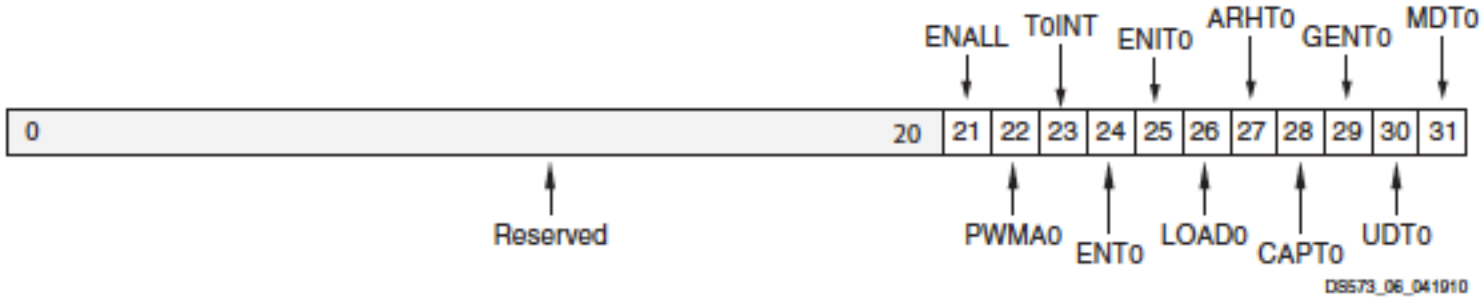


Figure 6: **Timer Control/Status Register 0 (TCSR0)**

What code word do you write to set up and load timer0 as actually starting timer for the following?

1. Periodic
2. Generate interrupt
3. Count down

Go ahead and load pre-load value

0 0 [1] [0] 1 [1] [1] 0 0 [0] [0]

Control/Status Register

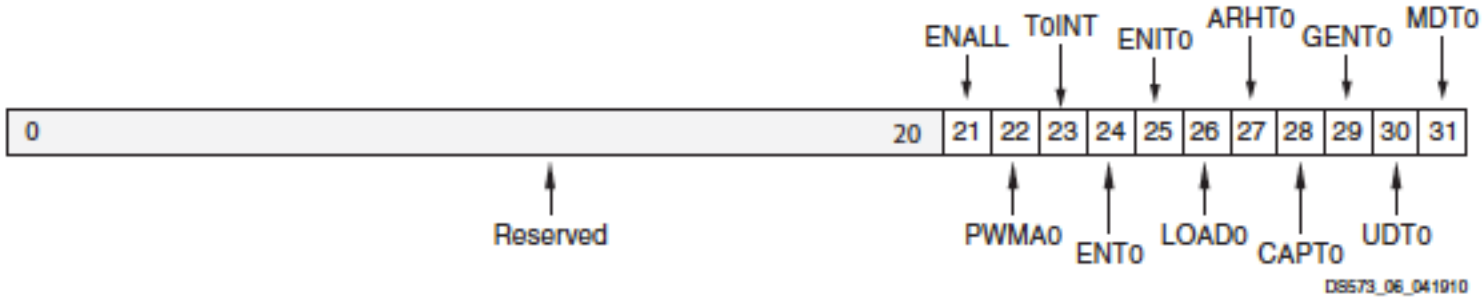


Figure 6: **Timer Control/Status Register 0 (TCSR0)**

What code word do you write to set up and load timer0 as actually starting timer for the following?

1. Periodic
2. Generate interrupt
3. Count down

periodic

0 0 [1] [0] 1 [1] [1] 0 0 [0] [0]

Control/Status Register

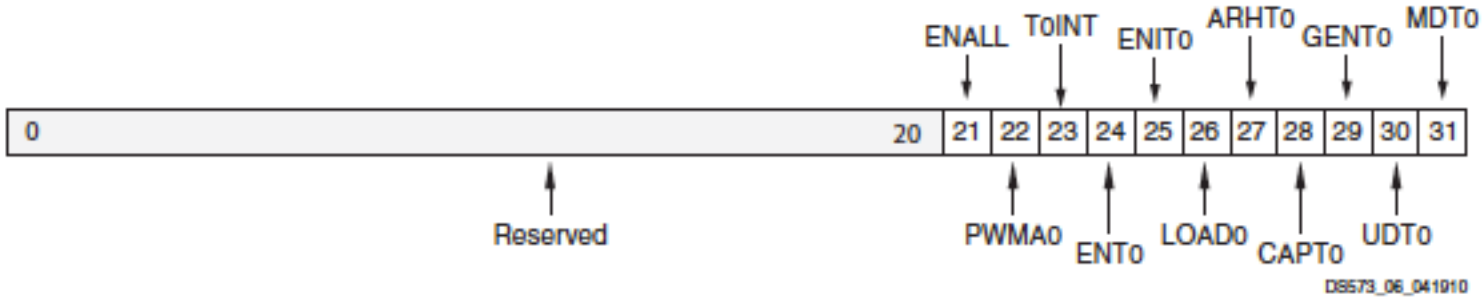


Figure 6: **Timer Control/Status Register 0 (TCSR0)**

What code word do you write to set up and load timer0 as actually starting timer for the following?

1. Periodic
2. Generate interrupt
3. Count down

Count down

0 0 [1] [0] 1 [1] [1] 0 0 [0] [0]

Control/Status Register

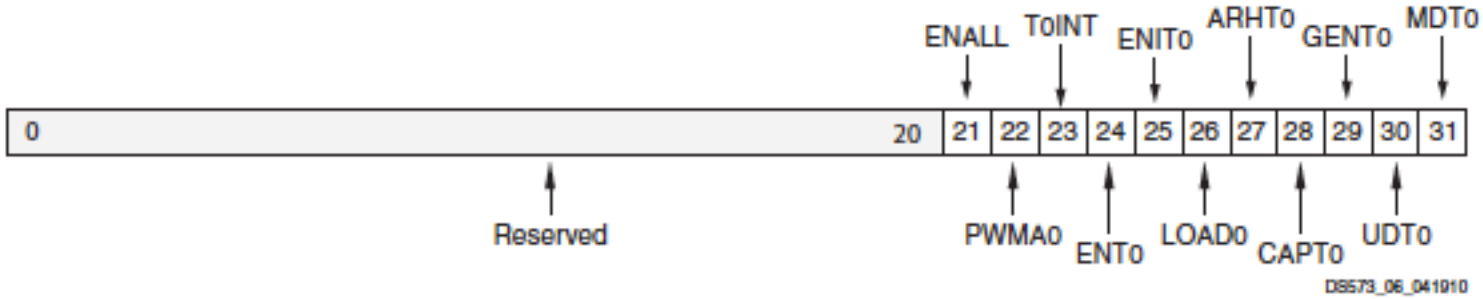


Figure 6: **Timer Control/Status Register 0 (TCSR0)**

What code word do you write to set up and load timer0 as actually starting timer for the following?

1. Periodic
2. Generate interrupt
3. Count down

Generate mode

0 0 [1] [0] 1 [1] [1] 0 0 [0] [0]

Control/Status Register

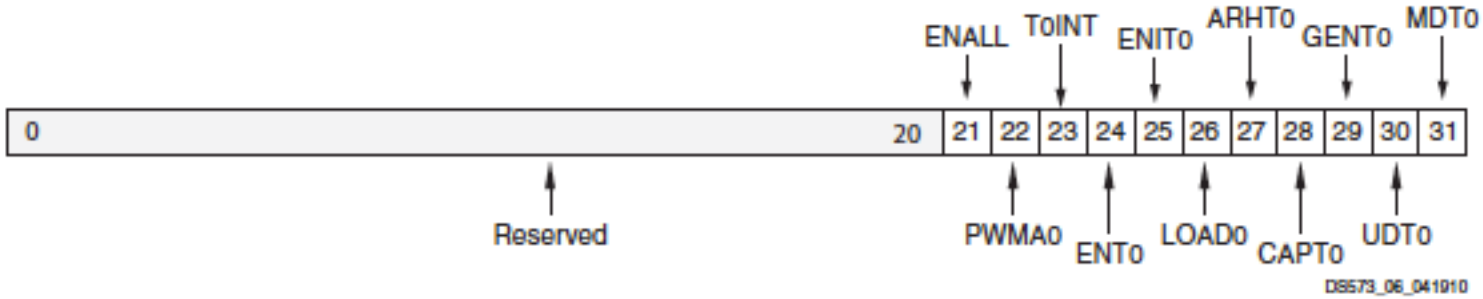


Figure 6: **Timer Control/Status Register 0 (TCSR0)**

How do you then start ?

command |= 0x80; / set the ENT0 =1 */