Hardware Timers

Polling Hardware Timers

LECTURE 7

IOULIIA SKLIAROVA

Hardware Timer IPs

AXI Timer

- AXI Timer/Counter is a 32/64-bit timer module that interfaces to the AXI4-Lite interface.

Fixed Interval Timer (FIT)

- The FIT core is a peripheral that generates a strobe (interrupt) signal at fixed intervals and is not attached to any bus.

AXI Timer

Two programmable interval timers with interrupt, event generation, and event capture capabilities.

Each timer module has an associated load register that is used to hold either the initial value for the counter for event generation or a capture value, depending on the mode of the timer.

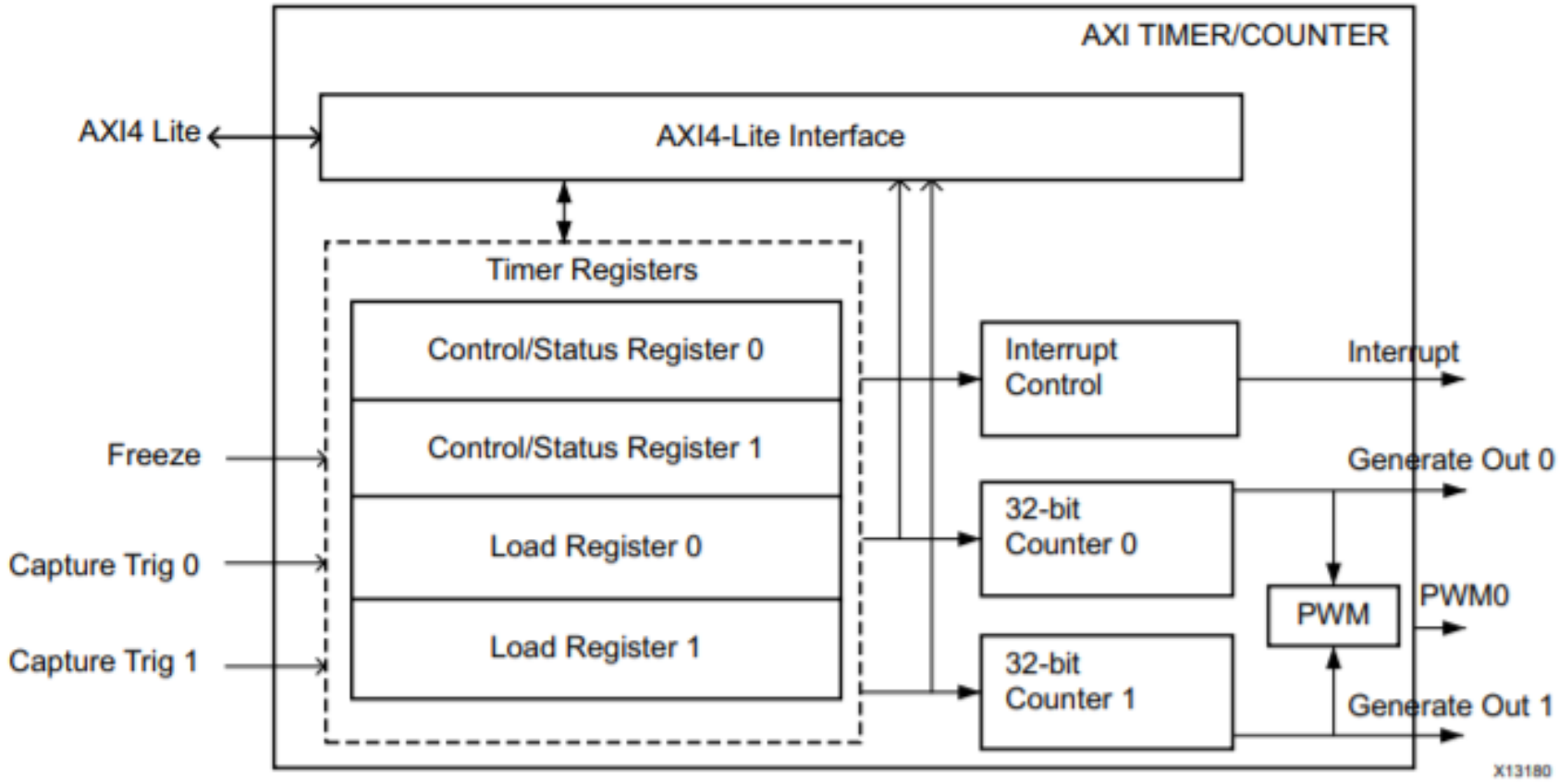
In the Generate mode, the value in the load register is loaded into the counter. The counter, when enabled, begins to count up or down, depending on the selection of the Up/Down Count Timer (UDT) bit in the Timer Control Status Register (TCSR).

On transition of the carry out of the counter, the counter stops or automatically reloads the generate value from the load register and, after reaching the timeout value, continues counting as selected by the Auto Reload/Hold (ARHT) bit in the TCSR.

The Timer Interrupt Status (TINT) bit is set in TCSR and, if enabled, the external GenerateOut signal is driven to 1 for one clock cycle.

If enabled, the interrupt signal for the timer is driven to 1 when reaching the timeout value. Clear the interrupt by writing a 1 to the Timer Interrupt register. Use this mode for generating repetitive interrupts or external signals with a specified interval.

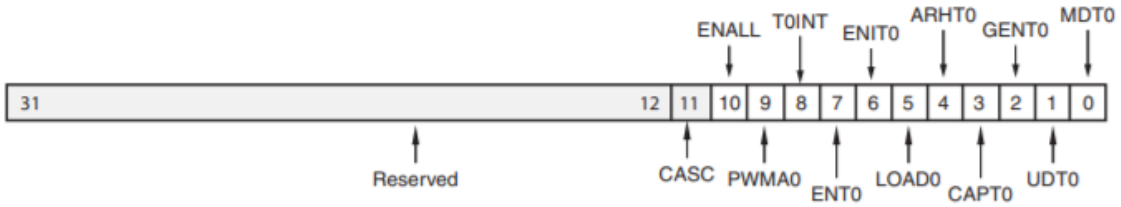
AXI Timer's Block Diagram



AXI Timer Control/Status Register

T0INT - Timer 0 Interrupt Indicates that the condition for an interrupt on this timer has occurred.

- Read:
  - 0 = No interrupt has occurred
  - 1 = Interrupt has occurred
- Write:
  - 0 = No change in state of T0INT
  - 1 = Clear T0INT (clear to 0)



ENT0 - Enable Timer 0

- 0 = Disable timer (counter halts)
- 1 = Enable timer (counter runs)

LOAD0 - Load Timer 0

- 0 = No load
- 1 = Loads timer with value in TLR0

ARHT0 - Auto Reload/Hold Timer 0 (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).

- 0 = Hold counter
- 1 = Reload generate value

UDT0 – Up/Down Count Timer 0

- 0 = Timer functions as up counter
- 1 = Timer functions as down counter

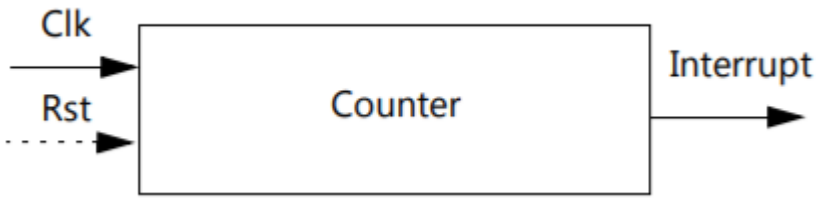
Fixed Interval Timer (FIT)

The FIT core is a peripheral that generates a strobe (interrupt) signal at fixed intervals and is not attached to any bus.

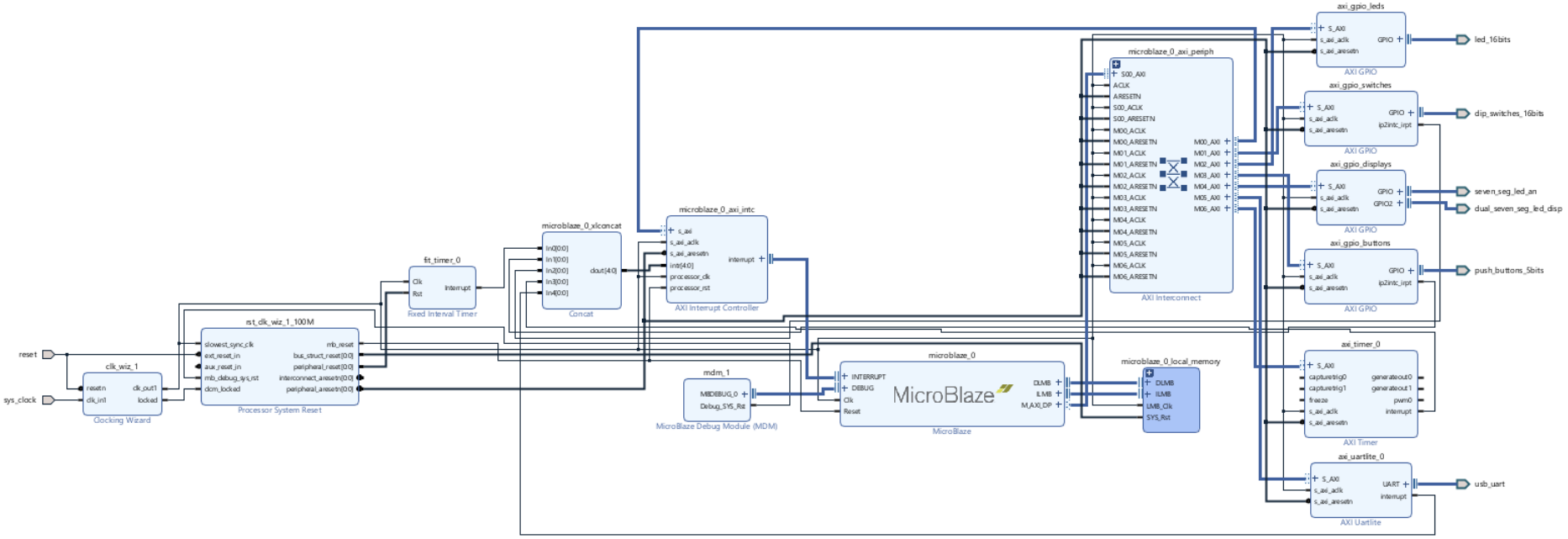
The FIT core generates an interrupt every C_NO_CLOCKS.

The interrupt signal is held high for one clock cycle.

The core begins operation immediately after device configuration if the reset is not connected.



Block Design (BD)



Polling

Polling means repeatedly reading a timer register and testing the input data value.

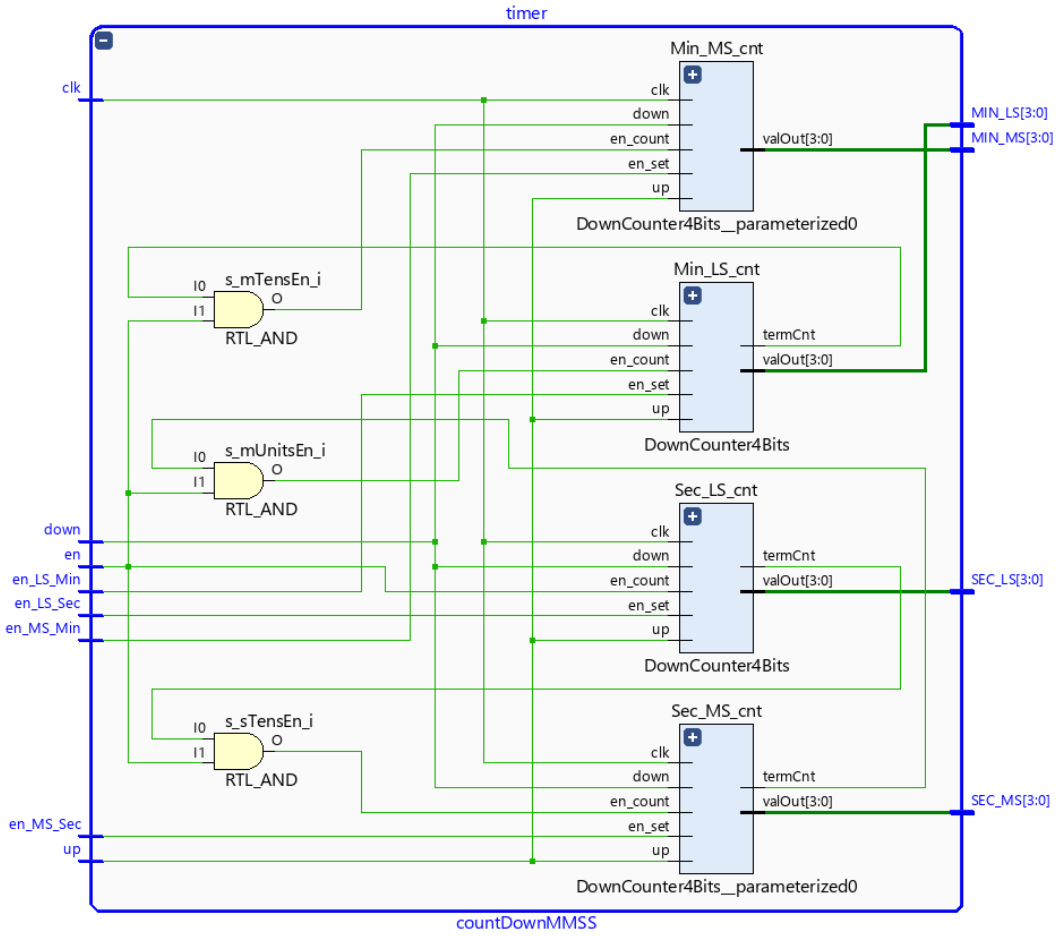
- simple to code
- not very efficient because the polling happens even if the timer reading has not achieved the required value

Low-level driver functions (or macros) that can be used to access the timer:

- `XTmrCtr_SetControlStatusReg`
- `XTmrCtr_GetControlStatusReg`
- `XTmrCtr_SetLoadReg`
- `XTmrCtr_GetTimerCounterReg`

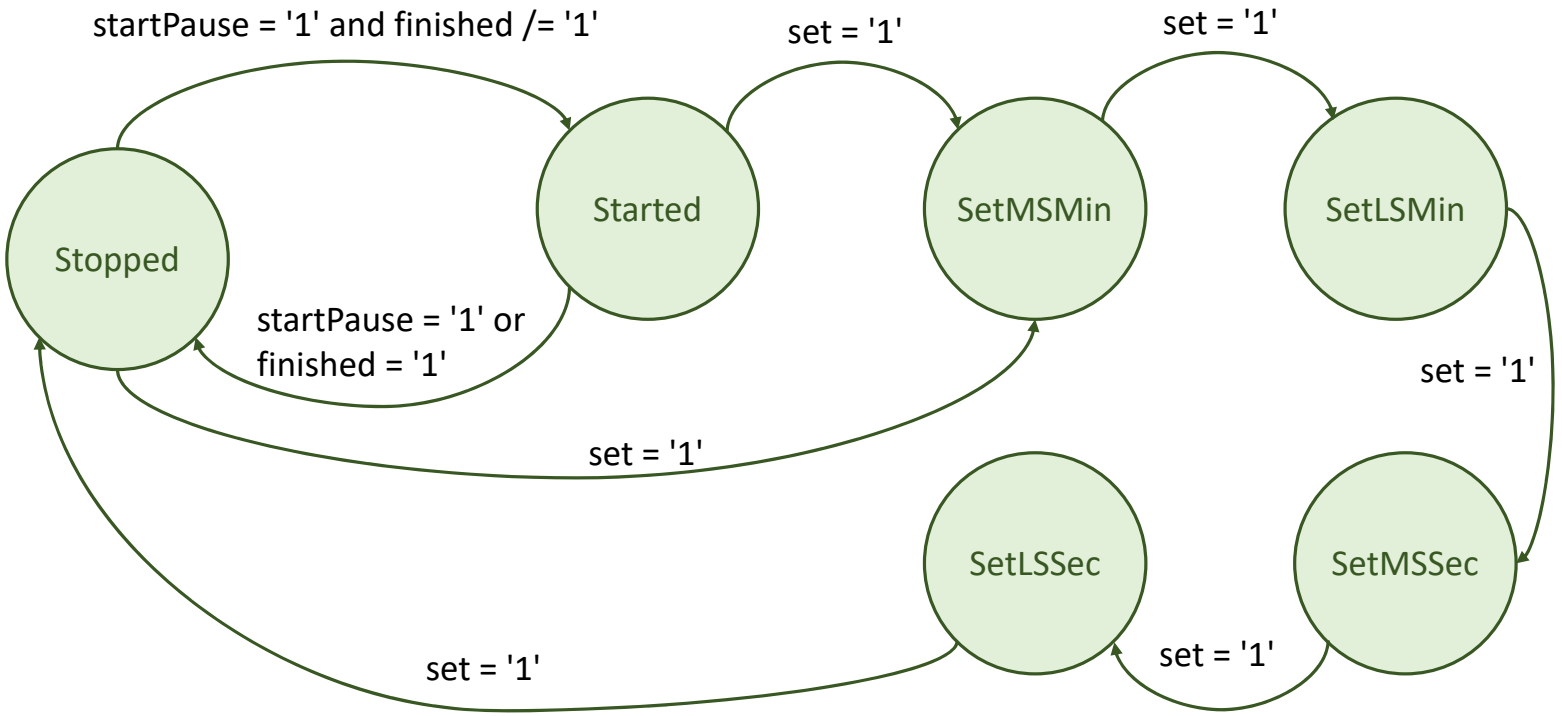
Countdown Timer Datapath

Min_MS | Min_LS | Sec_MS | Sec_LS



Countdown Timer Controlpath

set = btnR
startPause = btnC



Countdown Timer in Software

What periodic operations have to be executed and at what frequencies?

1 Hz

- to decrement the countdown timer

800 Hz

- to refresh the 7-segment displays

2 Hz

- to set the time
- to blink the point separating minutes and seconds (with 1Hz frequency)

4 Hz

- to make the digit being set to blink (with 2Hz frequency)

8 Hz

- to read the buttons status

C Code for the Countdown Timer

```c
// State machine data type
typedef enum {Stopped, Started, SetLSSec, SetMSSec, SetLSMin, SetMSMin} TFSMState;

// Buttons GPIO masks
#define BUTTON_UP_MASK      0x01
#define BUTTON_DOWN_MASK    0x04
#define BUTTON_RIGHT_MASK   0x08
#define BUTTON_CENTER_MASK  0x10

// Data structure to store buttons status
typedef struct SButtonStatus
{
    bool upPressed;
    bool downPressed;
    bool setPressed;
    bool startPressed;

    bool setPrevious;
    bool startPrevious;
} TButtonStatus;

// Data structure to store countdown timer value
typedef struct STimerValue
{
    unsigned int minMSValue;
    unsigned int minLSValue;
    unsigned int secMSValue;
    unsigned int secLSValue;
} TTimerValue;
```

Test 1

Structure

- True/False questions – no penalty for wrong answers
- Multiple-choice questions – 33% discount for wrong answers; no discount for not answering
- Cloze questions (complete a given code, calculate some values) – no penalty

Organization

- During the class on May 11
- 1h
- Use of calculators is allowed
- Prepare a pen and paper
- No consultation is allowed

Material

- Everything considered in the previous classes
- However, there will be no questions about software development

Final Remarks

At the end of this lecture you should be able to:

- write C programs that poll a hardware timer

To do:

- Lab. 6 part 1