Timers

The Need

Reliable, repeatable, stable time base

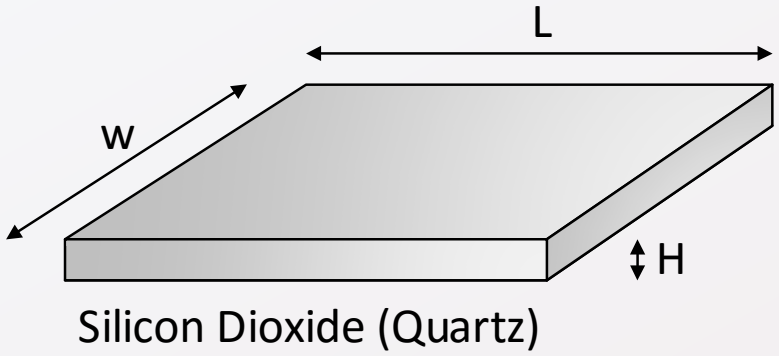
- Memory Access
- Interval/Event timers
- ADC
- DAC

Time Base: Crystal Oscillator

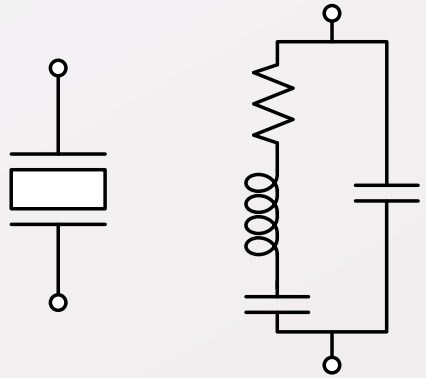
Silicon Dioxide forms a “piezoelectric” crystal that can deform in eclectic field, or produce an electric field when deformed.

When a voltage is applied, the crystal mechanically resonates at a stable frequency determined by its geometry.

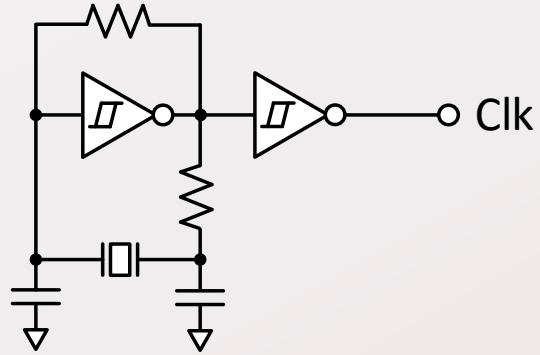
Resonance produces a stable oscillating voltage that can be amplified into a clock signal.



Quartz crystal size determines resonant frequency



Crystal symbol and equivalent circuit



Typical oscillator (clock) circuit

Time Bases

Crystal oscillators are cheap (~20 cents), effective, stable, and used a lot: watches, radios, computers, etc.

Important specifications: frequency, frequency tolerance, frequency stability (jitter – ppm), aging, voltage variation, load variation, power output, start-up time, package size

Recently, MEMS oscillators are becoming popular. They use a small vibrating/resonating structure and PLLs to create a stable time base rivaling crystals.

MEMs can offer lower cost, wider environmental performance, higher integration, and in-system programmability.

Timers

Many processes need precise times, or need regular interval service

Measuring elapsed time: How long, and how precise? How many bits, and what frequency?

Measuring intervals: How long, and how precise? How many bits, and what frequency?

At 10MHz, new count value every 100ns.

16 bits: 2^{16} is 65536 or 6.5ms

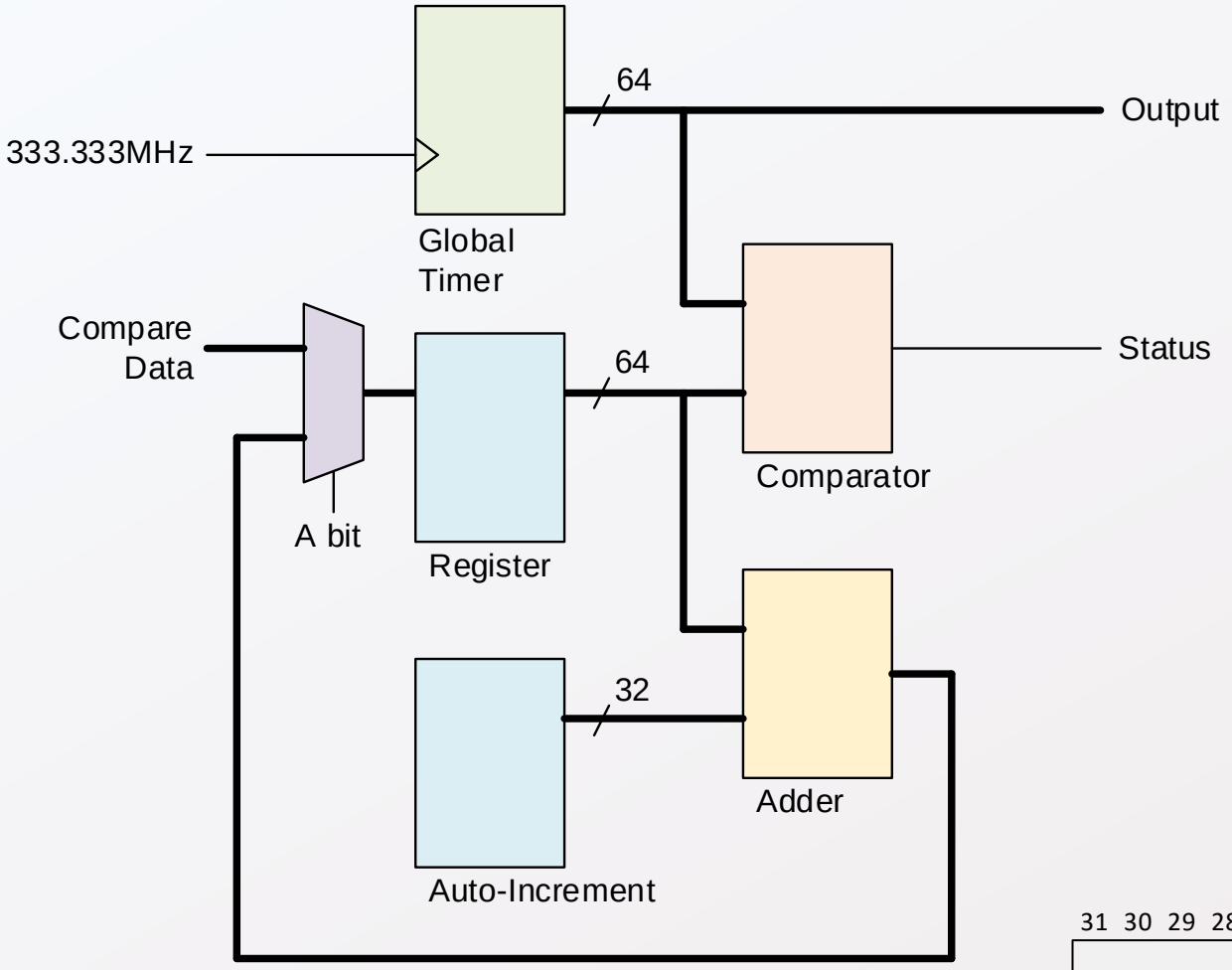
32 bits: 2^{32} is 4,294,967,296 or 430 seconds (7 minutes)

64 bits: 2^{64} is 18,446,744,073,709,551,616 or 58,500 years

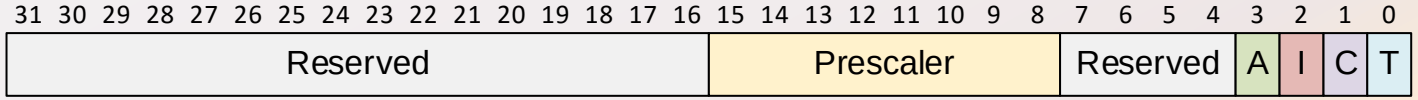
At 1GHz, 585 years with 1ns precision

Elapsed Timer: Global Timer

64-bit timer with fixed clock input (1/2 processor clock; 333.333MHz on Blackboard)



Global Timer Registers (0xF8F0 0000)		
Address	Function	Bits
0x200	Counter lower 32 bits	32
0x204	Counter upper 32 bits	32
0x208	Control	12
0x20C	Interrupt Status	1
0x210	Compare value lower 32 bits	32
0x214	Compare value upper 32 bits	32
0x218	Auto increment value	32



Control Register

Note: +1!

Interval Timer: Private/Watchdog Timers

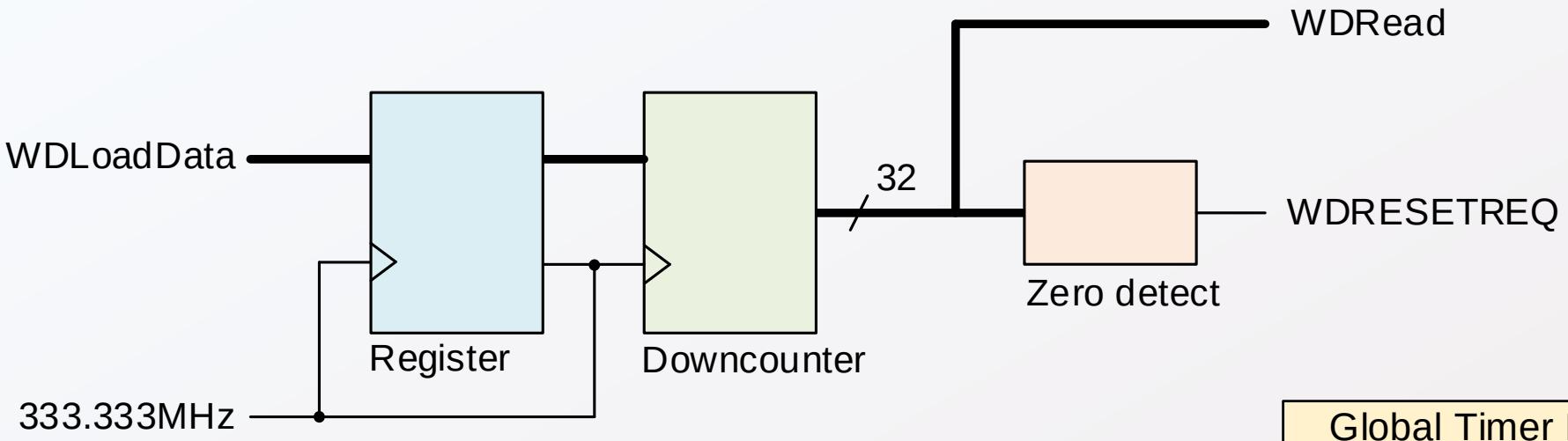
Timers to measure certain/critical events

Watchdog timer: Sets a “timeout” that if reached, can reset computer

- Countdown timer that must be reset periodically, or interrupt happens
- “Scheduling” software must reset timer before timeout period
- If ISR ever executed, then scheduler didn’t run, and processor assumed hung
- Can be nested; inner stages might check less critical functions; last state resets

“Private” timers perform similar functions, but for applications (not system)

Interval Timer: Private/Watchdog Timers



ZYNQ adds an additional 24-bit WDT that can timeout in the 330us to 687s time range. Why?

Global Timer Registers (0xF8F0 0000)		
Address	Function	Bits
0x00	Private timer load	32
0x04	Private timer counter (read)	32
0x08	Private timer control	11
0x0C	Private timer interrupt status	1
0x20	Watchdog load	32
0x24	Watchdog counter (read)	32
0x28	Watchdog control	12
0x2C	Watchdog interrupt status	1
0x30	Watchdog reset status	1
0x34	Watchdog disable	32

Interval Timer: More applications

Generate a time base for periodic events (e.g., reading a sensor repetitively)

Generate periodic signals (e.g., a PWM signal, or a specific low-frequency clock)

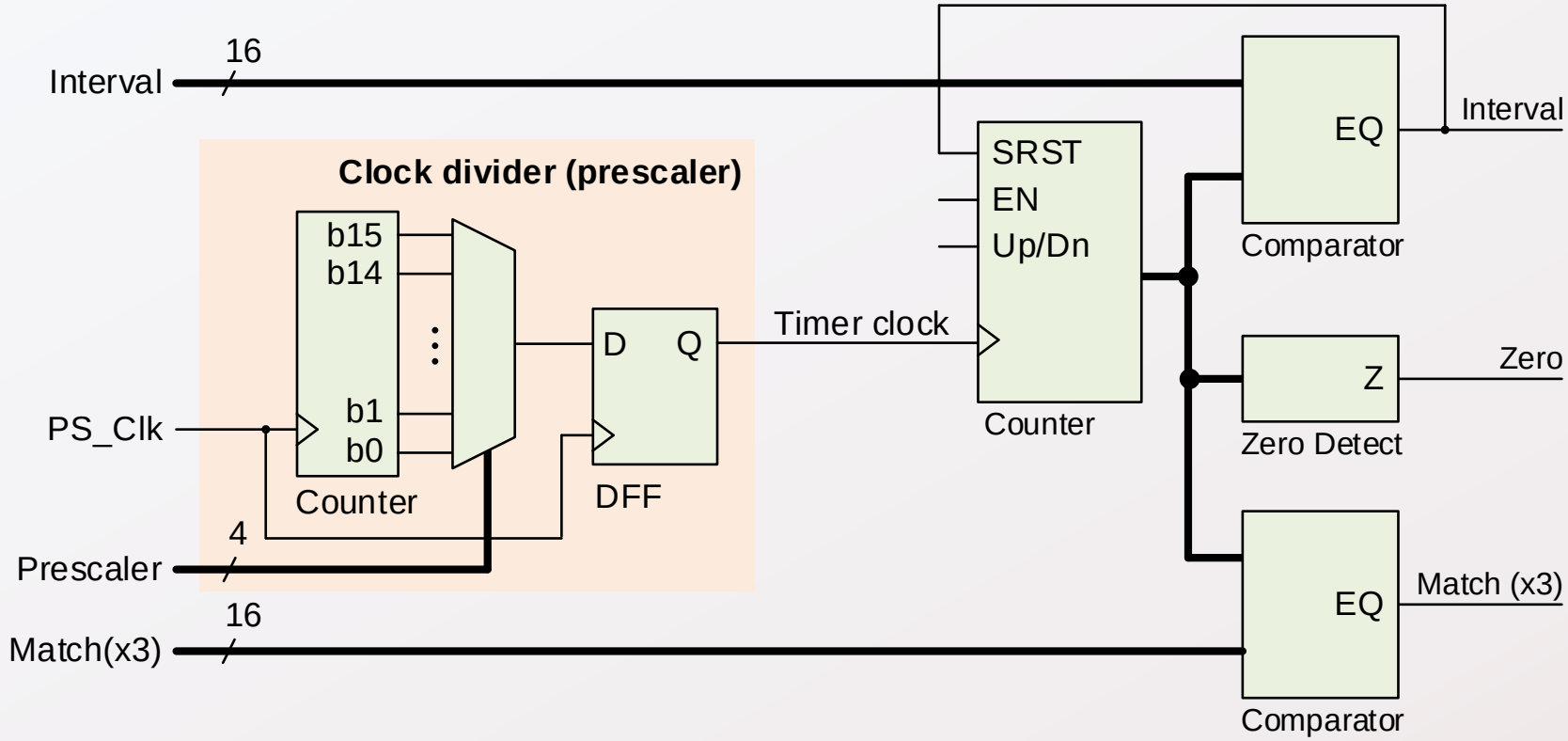
Create “one-shot” signal sequences (e.g., controlling a specific device)

Precisely measure input signal relative timing

Precisely count external events

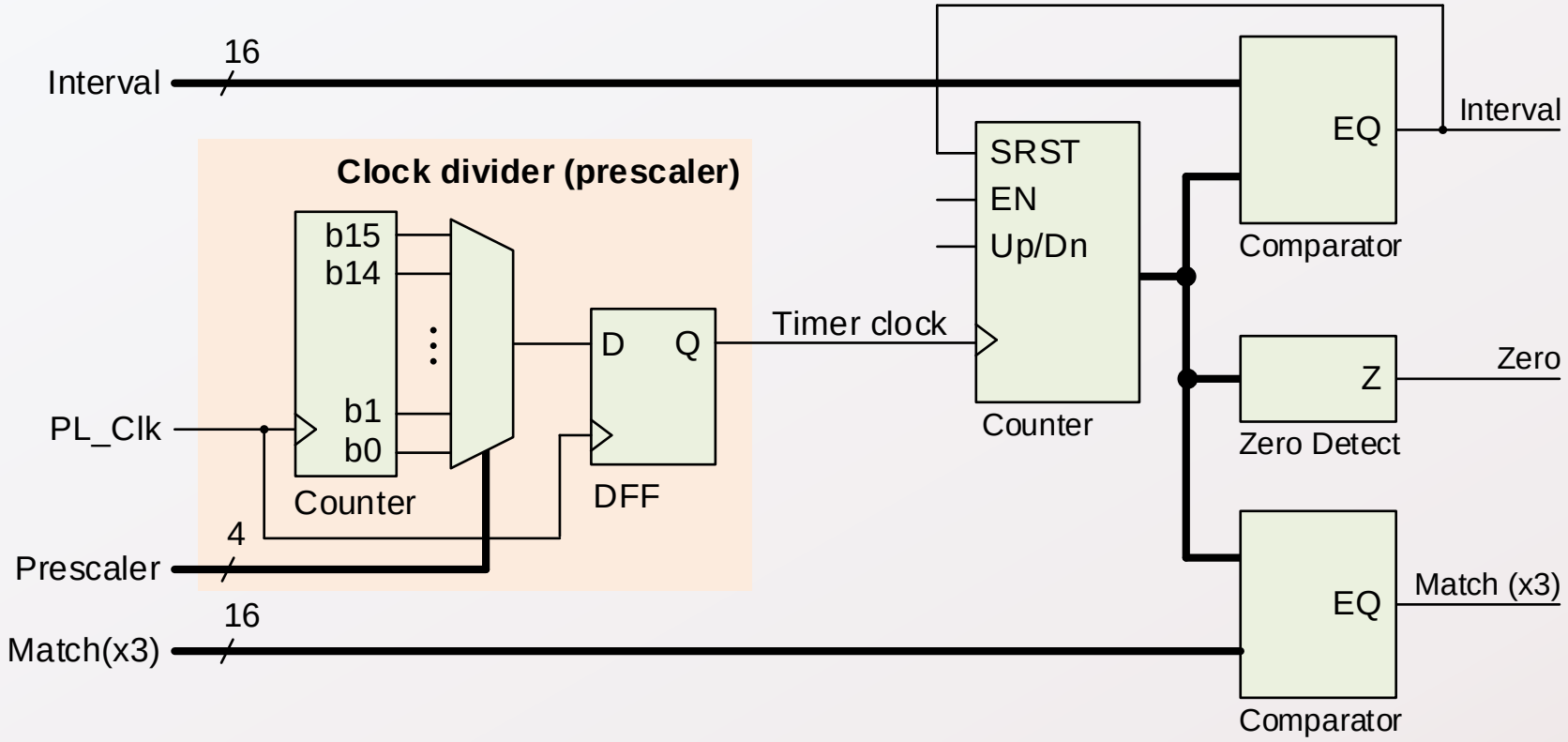
ZYNQ's Interval Timers: Triple-Timer Counter

ZYNQ contains two TTC modules, and each TTC contains three counters like the one shown.



ZYNQ's Interval Timers: Triple-Timer Counter

Two of the timers can receive a clock from the PL; these can be configured as event timers.



ZYNQ's Interval Timers: Triple-Timer Counter

TTC0 Counter 1 Configuration and Status Registers (Base Address 0xF800 1000)			
Name	**Function**	**Address**	**Bits**
CLK_CNTRL (Configure Clock)	Input clock setup (edge, source, prescale)	0x 0000 0000	7
CNT_CNTRL (Configure Counter)	Define configuration (output polarity, enables, up/dwn. Etc)	0x 0000 000C	7
CNT_VAL (Current count value)	Current counter value	0x 0000 0018	16
INTERVAL (Interval value)	Sets value counted to before reset	0x 0000 0024	16
Match1 (Counter1, Match1)	Match1 value for counter 1, can set output waveform to 1 or 0	0x 0000 0030	16
INT_MASK (Interrupt Mask)	Mask bits for interrupts (match1,2,3, interval, overflow, timer)	0x 0000 0054	6
INT EN (Interrupt Enables)	Enables for interrupts	0x 0000 0060	6
EVENT TIMER (Event timer setup)	Enables for event timers and counter	0x 0000 006C	3
EVENT REG (Event counter value)	Event counter value	0x 0000 0078	16

ZYNQ's Interval Timers: Creating a PWM signal

Each timer can generate an output waveform that can be sent to the PL.

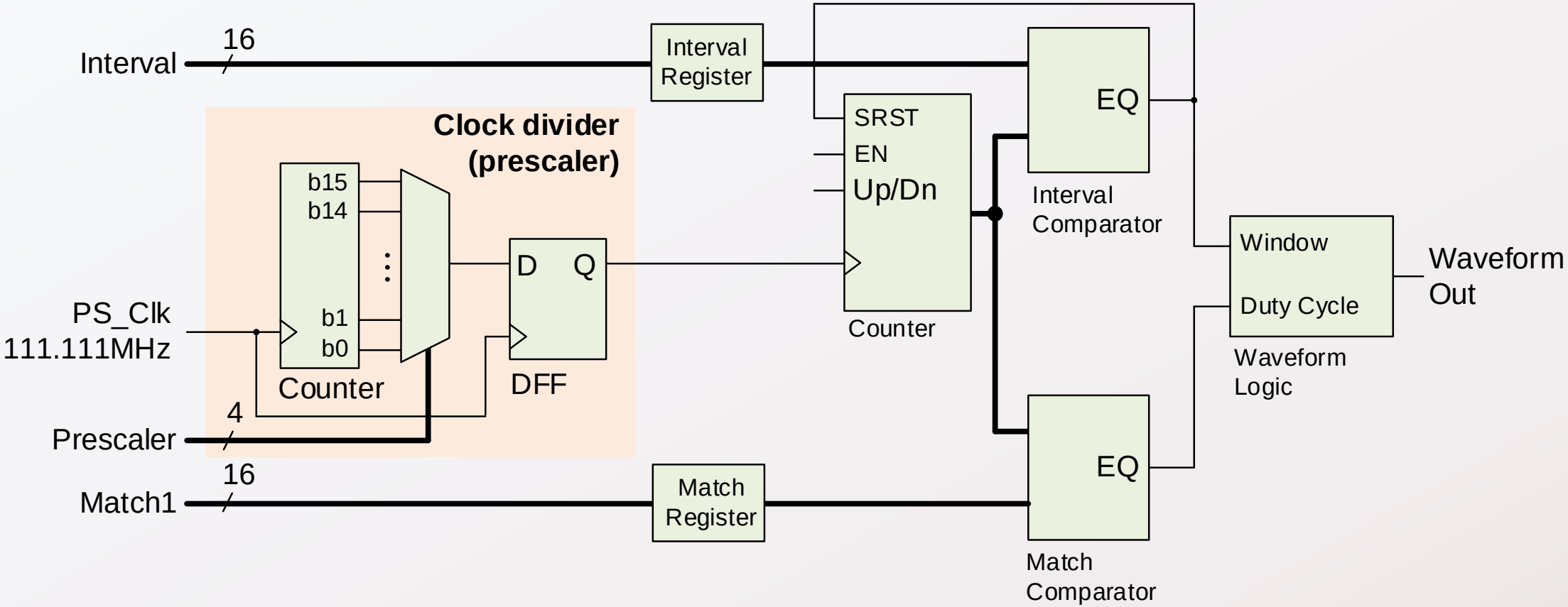
Interval and Match1 registers can be used to define a PWM signal.

Interval sets PWM window frequency (and resets counter)

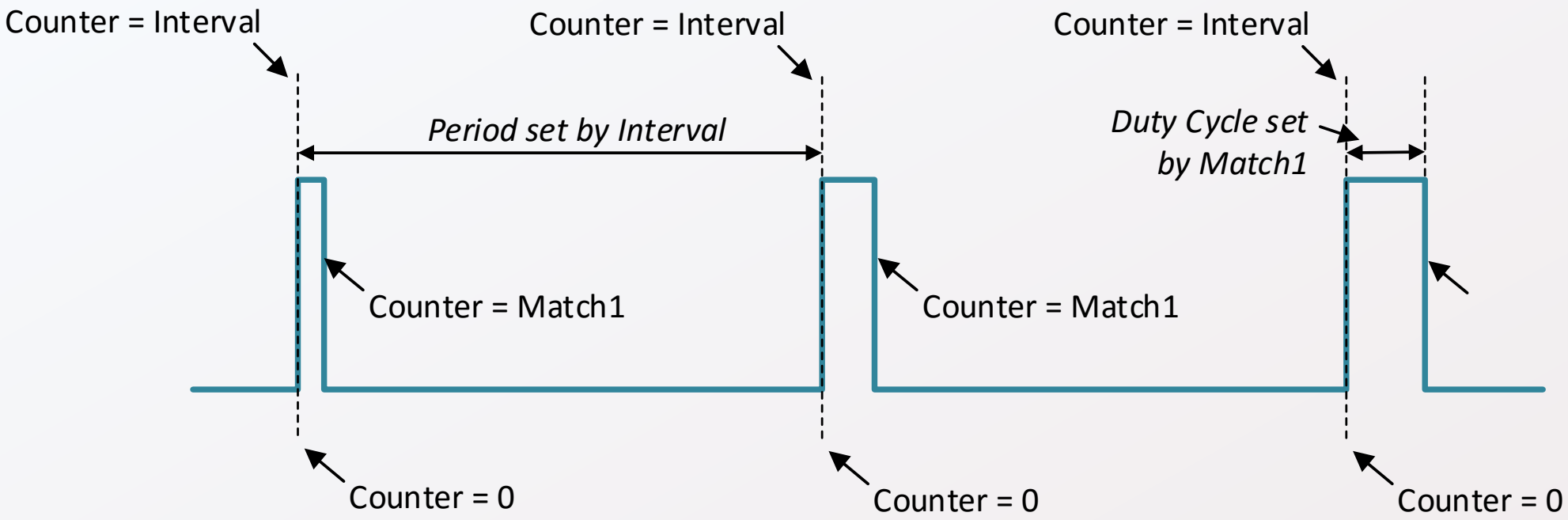
Match1 sets output waveform high on match (CLK_POL = 0) or low (CLK_POL = 1). Waveform resets at interval.

ZYNQ's Interval Timers: Creating a PWM signal

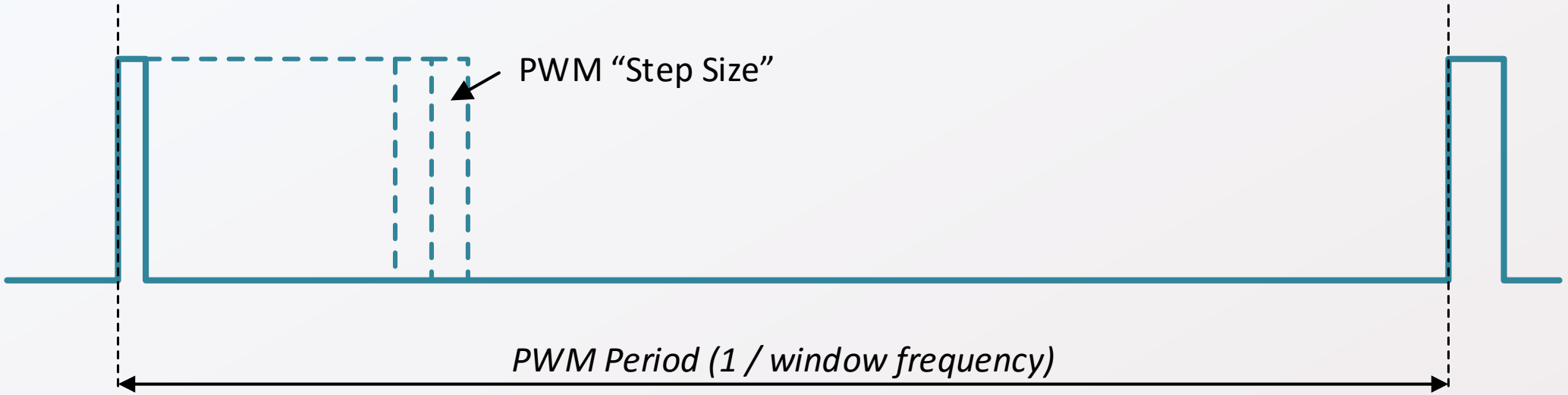
Each timer can generate an output waveform that can be sent to the PL.



Creating a PWM signal



Creating a PWM signal



Creating a PWM signal

Based on signal requirements, determine resolution (step size and number of steps required) and PWM frequency (or period). Examples…

Setup prescaler and interval counter to create PWM period (examples)

Setup CLK_POL to define output waveform polarity

Program Match1 (continuously) to define PWM duty cycle (pulse width)