ECE 2049: Lecture 8 Notes

Administrivia

- Lab 1: Signoff due Monday (6/21) by 7PM EDT
  - Live lab session today
  - I will add office hrs over the weekend (probably Sun. afternoon)
  - Whenever you start the lab, you should submit the prelab
- HW4 (short!) due Tues (6/22)
  - One short problem
  - Anonymous mid-term survey
- Lab 2: Starts next week
- Exam 1: Hope to have done by Monday

Office Hours

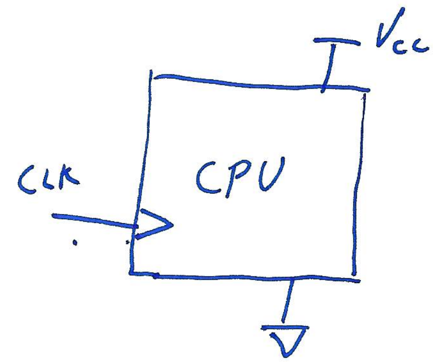
- Today: 2-4PM, 5-7PM EDT (Nick)
- Weekend: Probably Sun. afternoon (EDT) (details soon)
- Monday: 11AM-1PM (Chintan), 5-7 PM EDT (Nick)

Module 7. Intro to Clocks and Timers

Clocks

A microcontroller and its peripherals are just sequential logic circuits. Remember that sequential logic circuits need a *clock signal*. Before a CPU can operate, it must have power, a clock signal, and ground.

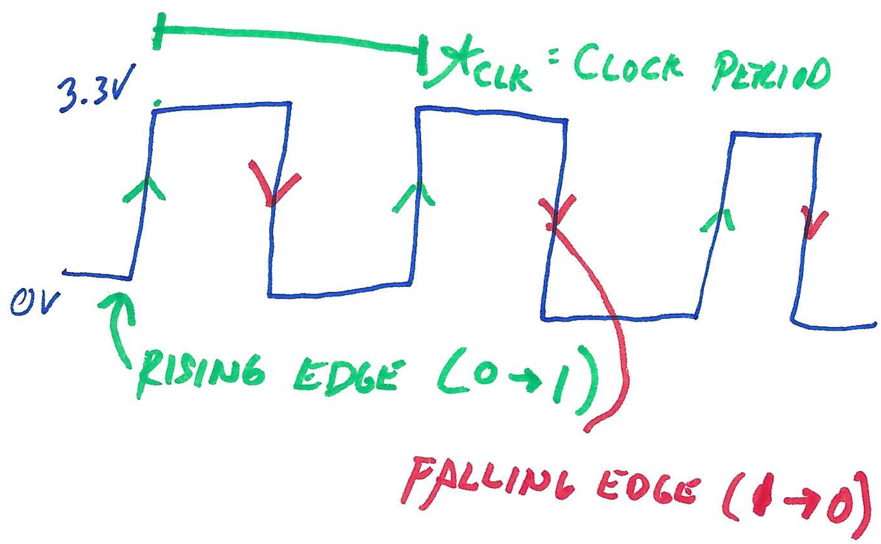
What does a clock signal look like?



- Provides time reference

- Drives code execution

All CPU instructions execute in some number of clock cycles.



1 clock period = 1 "tick"

Clock frequency

$f_{CLK} = 1/T_{CLK}$

Clocks on the MSP430: The Unified Clock System (UCS)

Microprocessors usually allow you to configure the clocks used by the system. On the MSP430, this tasks is handled by the **Unified Clock System** (UCS), which is billed as "full featured and capable" (read: complex and confusing)!

Like most microcontrollers, the MSP430 has a variety of configurable *clock sources* and clock *signals*:

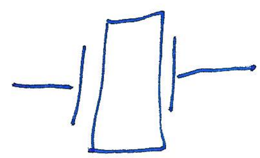
SOURCES: CIRCUITS THAT PROVIDE TIME REFERENCE

⇓

SIGNALS: DRIVE PERIPHERALS + CPU CORE.

There are two types of clock sources:

- External sources:
  - OSCILLATOR CRYSTALS (XTAL)
    CONNECTED TO SPECIAL PINS



- Internal sources:
  - ON-CHIP CIRCUIT THAT MAKES AN OSCILLATOR
    (HOW? MICRO II)

Why is all of this configurability important?

- PRECISE CONTROL OF CLOCK SPEEDS FOR APPLICATION

⇒ MAXIMIZE POWER EFFICIENCY

The MSP430F5529 has 5 possible clock sources:

XT1CLK LOW-FREQUENCY OSCILLATOR (LFXTAL)
32768Hz CRYSTAL (32.768 kHz)

XT2CLK HIGH FREQUENCY OSCILLATOR (HFXTAL)
4 MHz CRYSTAL

DCOCLK
↳ DIGITALLY CONTROLLED OSCILLATOR

REFOCLK
VLOCLK
↳ INTERNAL OSCILLATORS

These provide 3 clock signals to the CPU and peripherals:

ACLK - Auxiliary Clock:
- USED FOR PERIPHERALS (XT1CLK)
- USUALLY 32768 Hz

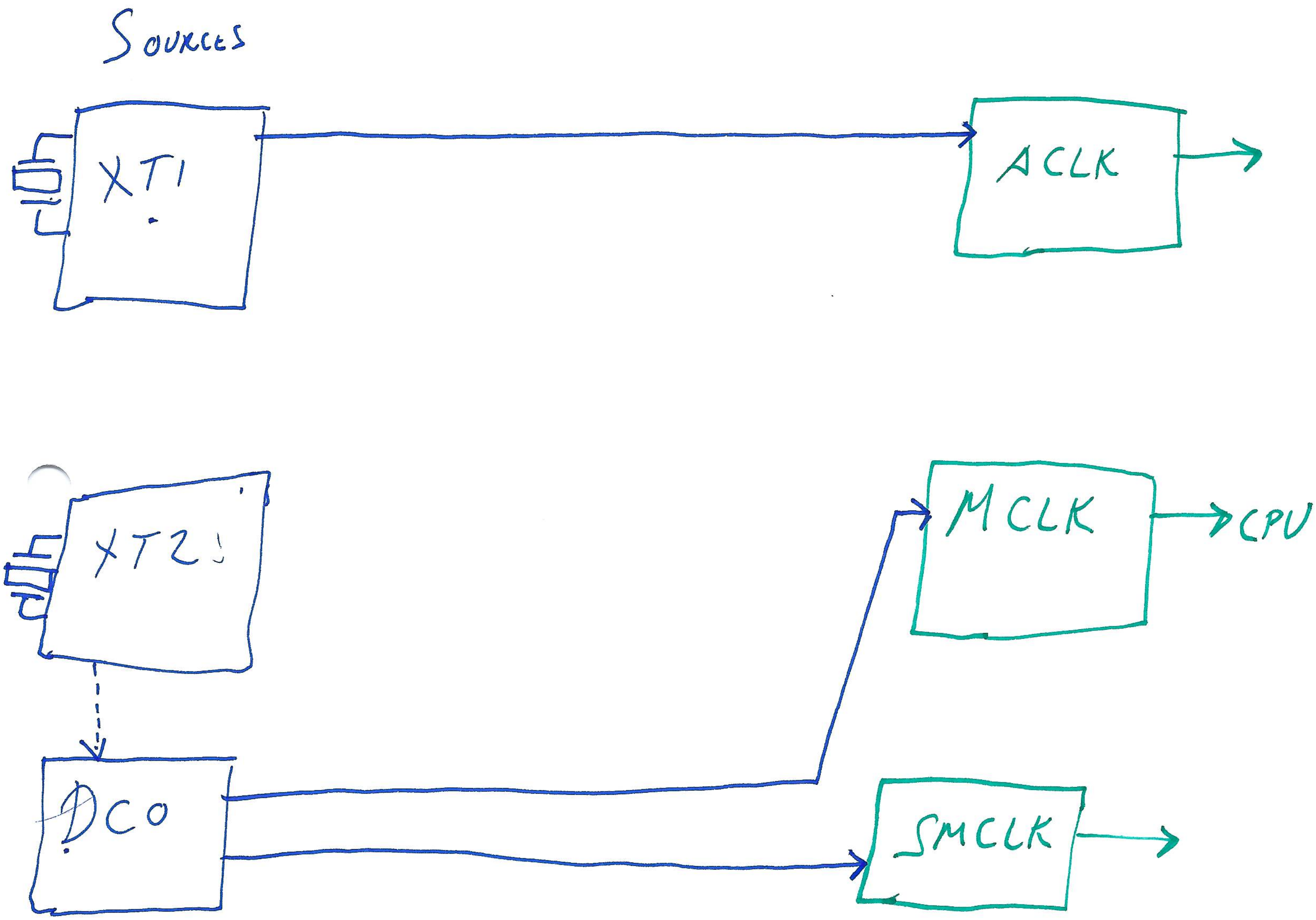
MCLK - Main or Master Clock:
- USED BY CPU CORE (HOW FAST CODE RUNS)
- MSP430: 20kHz – 20MHz DEFAULT: 1.048576 MHz

SMCLK - Sub-main Clock:
- USED FOR PERIPHERALS
- USUALLY 1.048576 MHz

The three clock signals are *software selectable*, meaning that the user can configure the clock sources and speeds for the CPU and peripherals **at runtime**.

Configuring the UCS: The Gist

In general, configuring the UCS boils down to connecting the various clock sources (XT1, XT2, DCO, etc.) to the 3 clock signals (ACLK, MCLK, SMCLK):



In addition, you also need to configure some parameters for the sources (like the DCO), and the signals (like clock dividers).

Configuration notes

Configuring XT1 and XT2

The low frequency and high frequency crystals XT1 and XT2 are connected via pins on the MSP430. On the MSP430F5529, these pins are multiplexed with P5.4-5 (for XT1) and P5.2-3 (for XT2).

If you want to use XT1 or XT2, you need to configure these pins for **function mode** (as opposed to digital I/O mode) by setting their corresponding bits in P5SEL to 1:

```
P5SEL |= (BIT5|BIT4|BIT3|BIT2);
```

FUNCTION MODE!

In our lab, this is already done for us in the template in the `configDisplay` function.

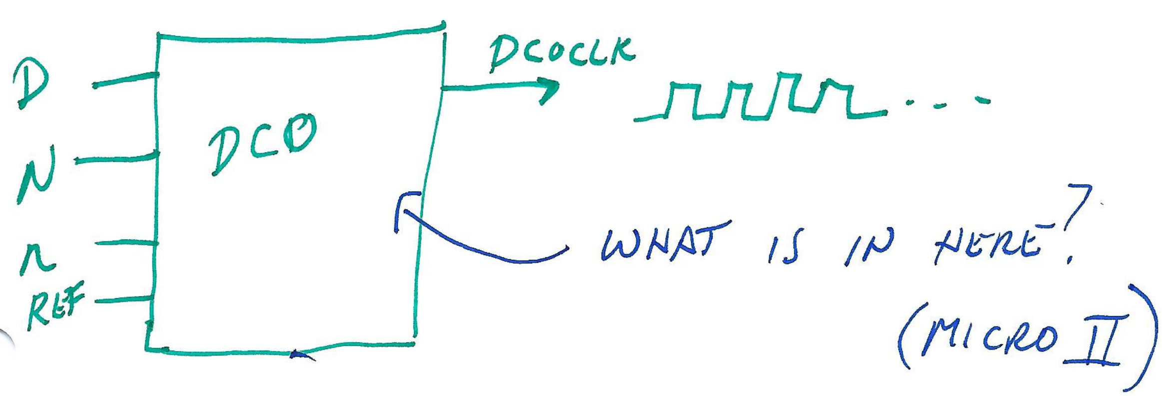
The DCO (Digitally-controlled oscillator)

The DCO is a *digitally-controlled oscillator*, which means that you can configure its frequency in software. The UCS module provides a frequency-locked loop (FLL) to stabilize the DCO. The frequency for the DCO is defined by the following formula:

$$f_{DCO} = D*(N+1)(f_{FLLREFCLK} / n)$$

SOME CLOCK SOURCE XT1, XT2, ...

DEFINED IN REGISTERS



Module 8. Timers: Theory and Practice

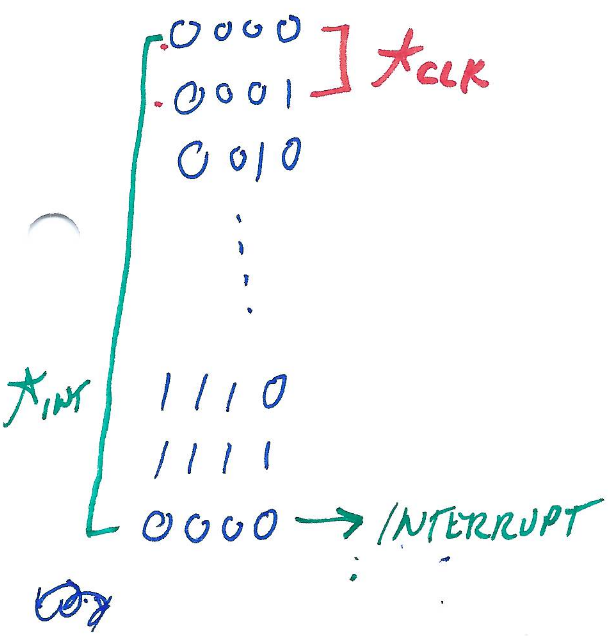
Topics

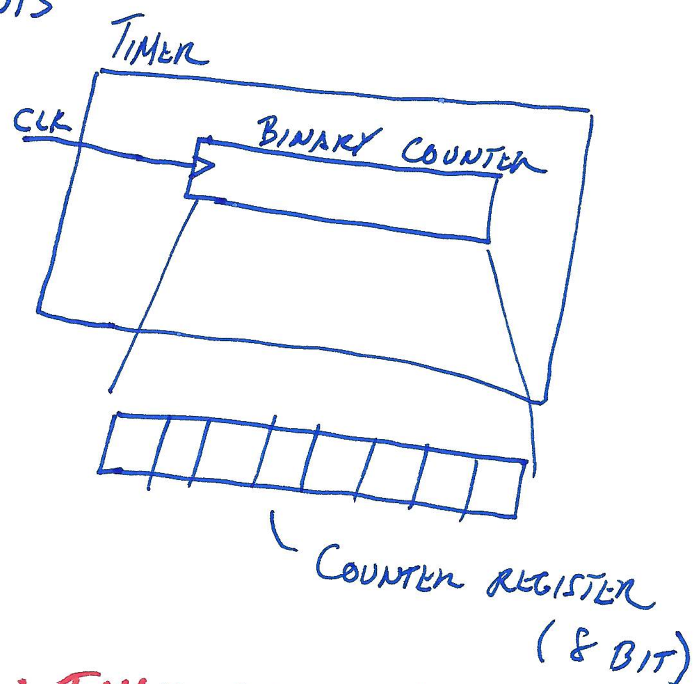
- Intro to Interrupts
- Intro to Timers

WHY TIMERS?
- CAN'T RELY ON EXECUTION OF CODE FOR PRECISE TIMING.

But first… what is a timer?

TIMER: CIRCUIT THAT COUNTS CLOCK TICKS





t_{CLK}: TIME BETWEEN CLOCK TICKS

t_{INT}: TIME BETWEEN INTERRUPTS

Most microcontrollers have timers in some form. Timers can be used to generate **interrupts** at particular intervals, generate PWM signals, measure frequency of input signals, and more! In this course, we will focus on the generation of timer interrupts, which tell the CPU that a certain amount of time has passed.

Fundamental timer counting modes

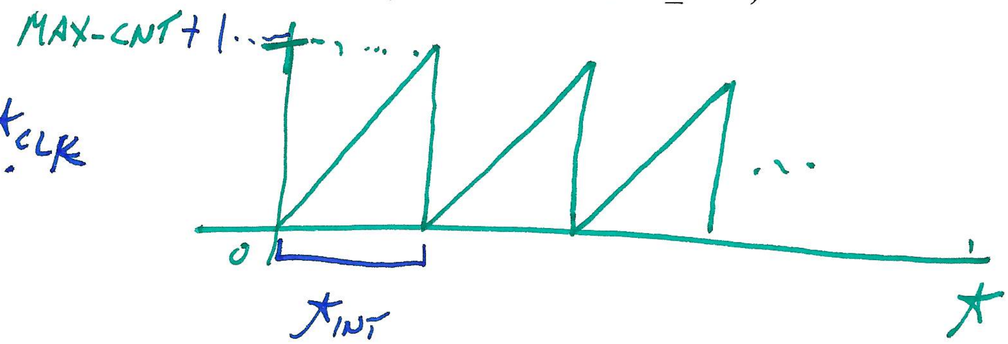
Most timers have a number of *counting modes*:

Unidirectional mode (called "Up mode" on the MSP430)

Count from 0 to a *programmer set* maximum count value (which we call MAX_CNT).

$t_{INT} = (MAX\_CNT + 1)\, t_{CLK}$

$t_{CLK} = 1/f_{CLK}$



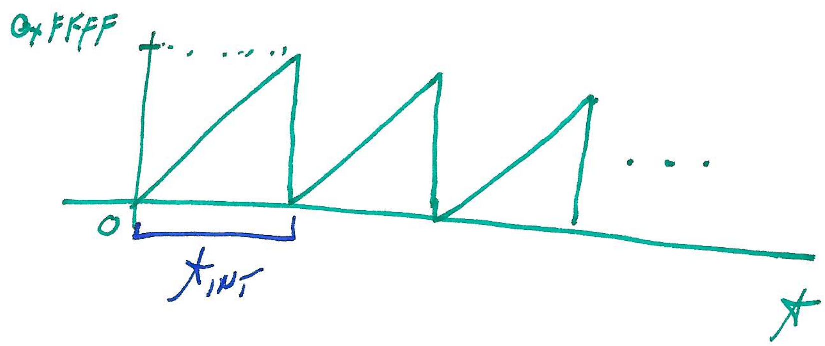
Continuous mode

Count from 0 to full count of timer (8, 12, 16 bits, etc.) For a 16 bit timer, this means:

FOR A 16 BIT TIMER

$0 - (2^{16} - 1) = 0 - 65535$

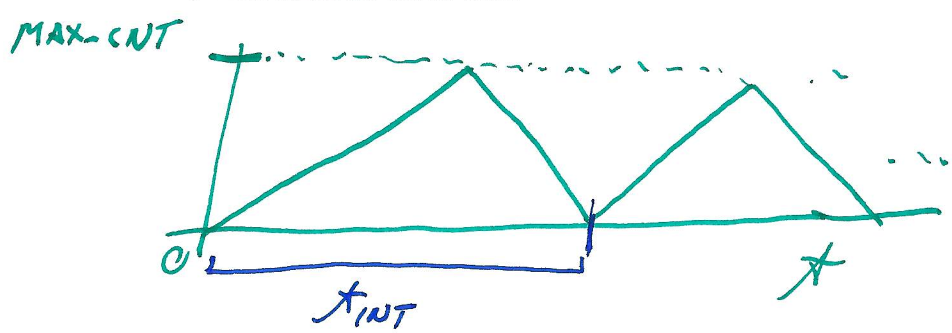
$t_{INT} = 2^{16}\, t_{CLK}$



Up/Down Mode

Counts from 0 to *programmer set* maximum count, then back down to zero

$t_{INT} = (2 * MAX\_CNT)\, t_{CLK}$



In each mode, most timers (like those on the MSP430) will trigger an interrupt when the count transitions back to 0.

Most timer peripherals have two "operating modes", which control how they use the counter:

- **Capture mode**: Records the counter value when a certain input changes
  - ↳ FREQUENCY COUNTING.
- **Compare mode**: Performs an operation when the counter value reaches a certain value
  - ↳ PERIODIC INTERRUPTS
  - PWM GENERATION.