ECE 2049 Lecture 14

Today
- Exam Review
- Digital Interfaces

Office Hours
- Thurs: 2-6 PM (Nick)
- Friday: 2-4 PM (Nick)
- Tues/Wed: Yes, maybe remote (schedule TBA)

Administrivia
- Exam 2: Out today, due Tuesday 7/5 by 11:59 PM EDT
  - Details in Canvas
  - If this deadline is a problem, let me know!
- Lab 2
  - Signoff due ~~Thursday by 6PM~~ Friday by 4PM
  - Report due Wednesday 7/6 by 11:59 PM
- Grades for everything except Lab 2: tomorrow.
- Course evaluations!
  - Look for an email (+ one from me)

IF YOU HAVE OUTSTANDING WORK OTHER THAN LAB 2 AND YOU HAVE NOT CONTACTED ME ALREADY, YOU SHOULD DO SO ASAP.

EXAM 2 TOPICS

HW 4-7, LECTURE 9 ONWARDS

- CLOCKS

- INTERRUPTS + HOW THEY WORK

TIMERS

- UTC TIME CONVERSION
- USING THE TIMER VARIABLE (GLOBAL COUNT)
- TIMER ACCURACY

ADC1

- CONCEPTS: NOT REQUIRED TO READ/WRITE ADC12 CONFIG CODE
- SENSOR DATA ↔ VOLTAGE ↔ COLS
- REASONING ABOUT SENSORS
- POWER MODES
- BASIC CONCEPTS ABOUT DIGITAL INTERFACES (TODAY)

Module 11. Intro to Digital Interfaces

Digital Interface: Connection between two or more digital devices.

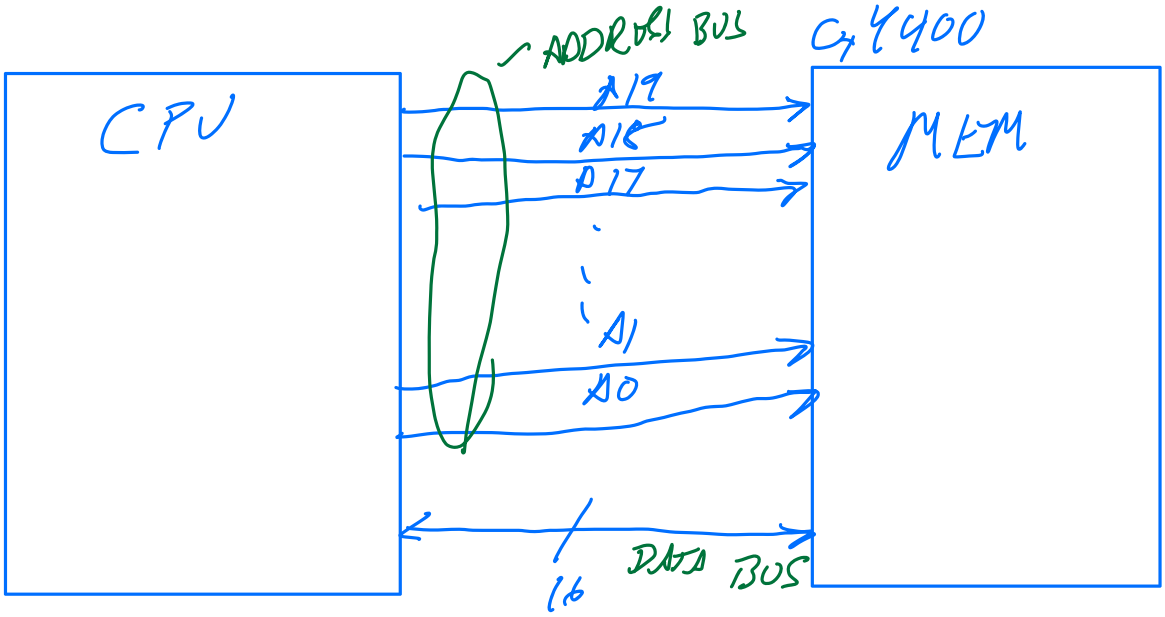
INSTEAD OF A VOLTAGE SCALE, ENCODE INFO IN DIGITAL LOGIC (JUST LIKE A CPU!)

Types of Digital Interfaces

There are two classifications for how data is transmitted in a digital interface: serial and parallel

Parallel Interfaces = Each bit has its own electrical connection (interconnect, trace, wire)

>> Advantages = Fast, easier to synchronize (1 clock edge transfers all bits together)



>> Used almost exclusively inside a CPU (or other) chip

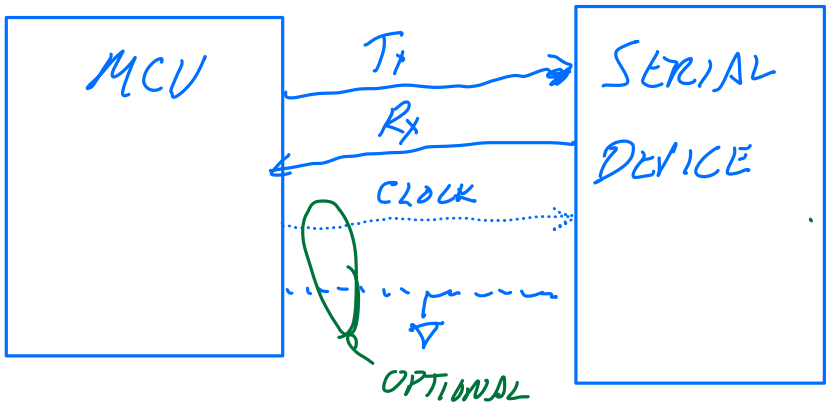
— MSP430 (MEMORY BUS: CPU → MEMORY, PERIPHERALS).

— INSIDE THE CPU ITSELF (CPU HARDWARE TO REGISTERS.)

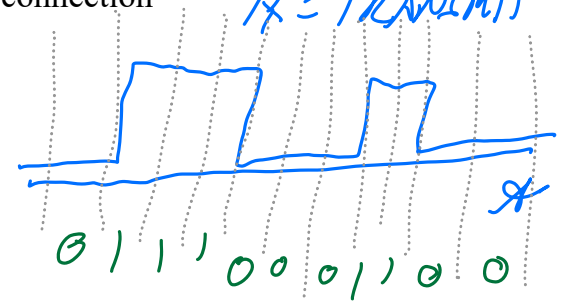
>> Disadvantage = Each bit must have its own electrical connection

- INEFFICIENT IN TERMS OF "REAL ESTATE"
- HARD TO ADD LOTS OF PINS TO A CHIP (OR INSIDE A CHIP)

Serial Interfaces – Bits sent one after another along a single connection



RX = RECEIVE
TX = TRANSMIT



>> Used almost exclusively to make connections off-chip (and off-computer, through the Internet, out to the Mars Rover, etc.)

>> Advantages = Simpler/fewer connections between CPU and peripheral (2-4 lines)

Examples include:

Device Select/Enable (CS)

Synchronizing CLK (SCLK)

Data Line(s) (SDI and SDO)

Common ground (GND): Often already established if devices are on the same board or IC

ETHERNET: 1 GBPS
USB 2.0: 480 MBPS
USB 3.0: 5 GBPS

MODERN INTERFACES USE BOTH STRATEGIES: MULTIPLE SERIAL LINES!

>> "Connection" = PCB trace, wire, RF, acoustic, optical, etc.
>> Disadvantages = ~~"Slower"~~, more complicated synchronization, potential timing issues

PER CLOCK CYCLE
MODERN SERIAL INTERFACES ARE VERY FAST: MEGABITS/GIGABITS PER SECOND!

How are digital interfaces implemented?

Most microcontrollers contain hardware peripherals that implement the interface in hardware. On the MSP430, we have hardware peripherals to implement a few types of serial interfaces.

On the MSP430: Universal Serial Communications Interface (UCSI)

>> Basically acts as a parallel-to-serial and serial-to-parallel converter

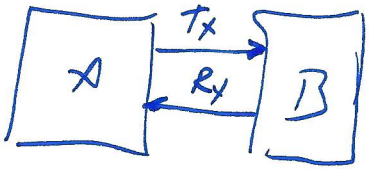
>> Most modern microprocessors/microcontrollers will have built-in Serial interface
-- MSP430F5229 has a total of four, in two types

>> Role of Serial interfaces has grown with growing sophistication and speed of serial links (SPI and I^2C to USB and others)

We will discuss three types of interfaces: UART, SPI, and I2C. There are many more types of digital interfaces!

What kind of information is exchanged?

Just like in a CPU, information in a digital interface is represented in bits. The interface defines the manner in which bits are transmitted:



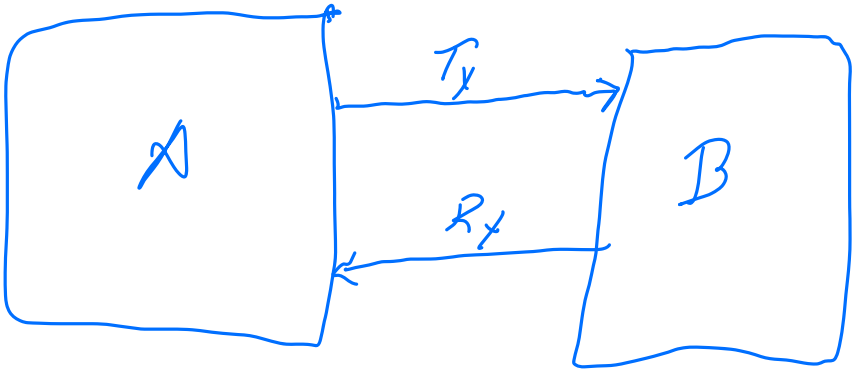
- WE CAN SEND WHATEVER WE WANT

- IN PRACTICE THERE ARE DEFINED FORMATS FOR HOW DATA IS SENT.
  ↳ (DATASHEETS!)

Types of Interfaces

UART

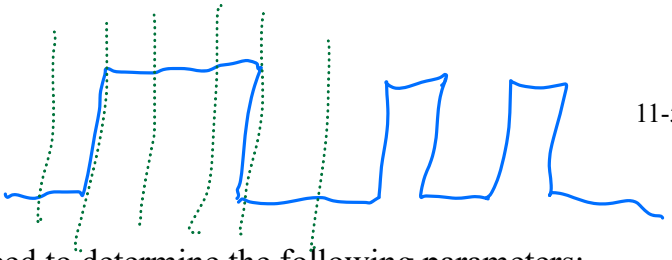
>> UART: Universal Asynchronous Receiver/Transmitter
>> UART mode configures basic 2-wire *asynchronous* serial communications
>> Connect to UCSI with 2 external pins (MSP430: `UCAxRXD` and `UCAxTXD`)



>> Not synchronous (no shared clock) = Asynchronous

>> To use serial communications both devices must know data format and baud rate

- These are set using UARTs control registers
- Implies make data format & baud rate decisions at design time



UART: Fundamental Parameters

In order to use UART for an application, you need to determine the following parameters:

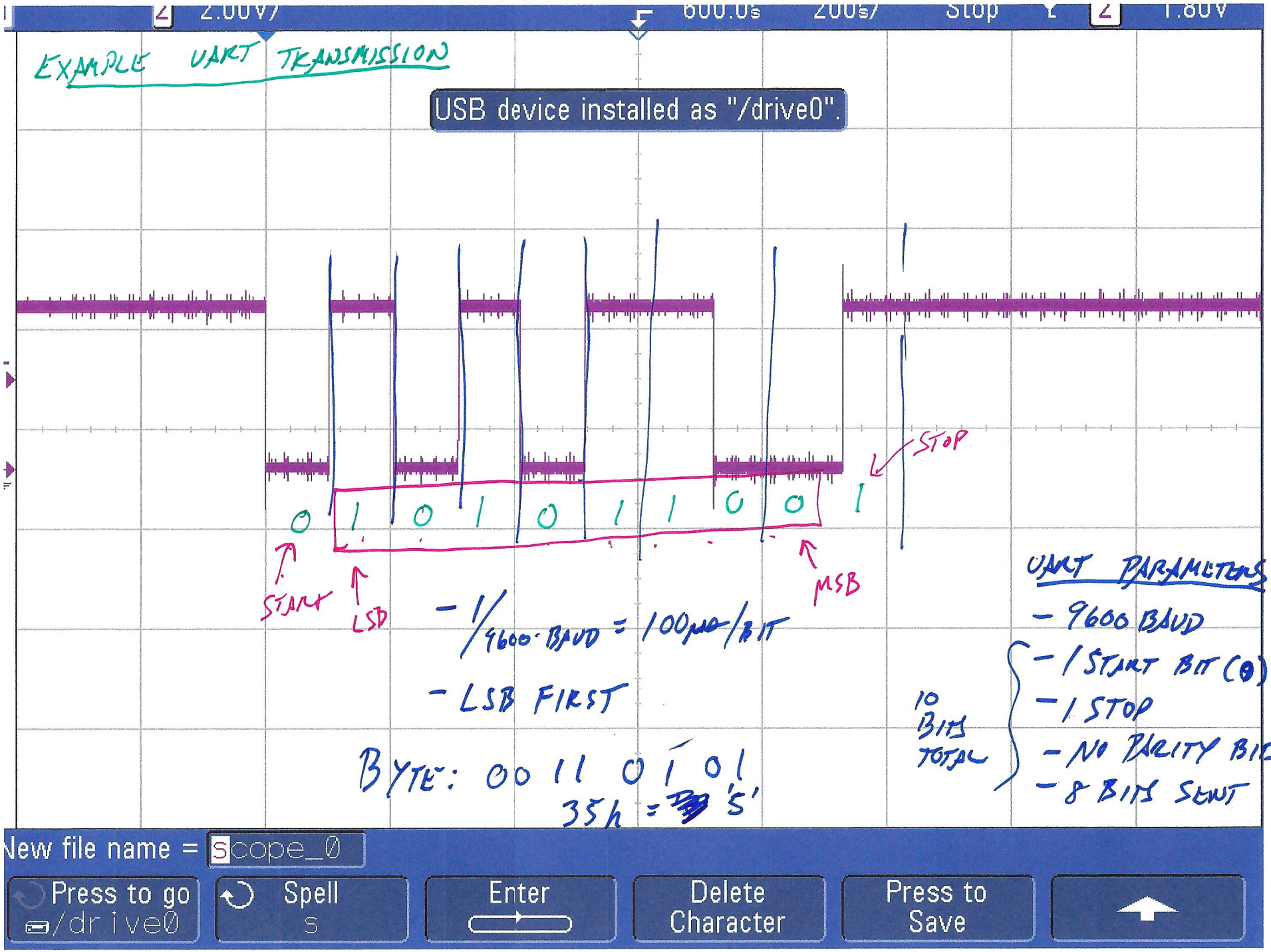
- Start Bit = 1 bit ("low")
- Data Bits = 7 or 8 bits
- Parity = Even, Odd, or None
  >> Even Parity = 1 when number of 1's including parity is even
  >> Odd Parity = 1 when number of 1's including parity is odd
- Stop bit(s) = 1 or 2 bits ("high")
- Baud rate (bits/sec): Common rates are 200, 2400, 9600, 19200, 115200

CAN BE USED FOR (WEAK) ERROR CHECKING

RS-232 – "Old standby" for serial format →Actually is a specific standard with associated voltage ranges and baud rated now often misused to mean any asynchronous serial communication

-->Data sent Least Significant Bit (LSB) first**

** Most UARTs send asynchronous serial data LSB first (because RS-232 is LSB first) but MSP430F5229 UCSI_A is configurable and can be set to send MSB first, so that it can be compatible with various types of serial interfaces.

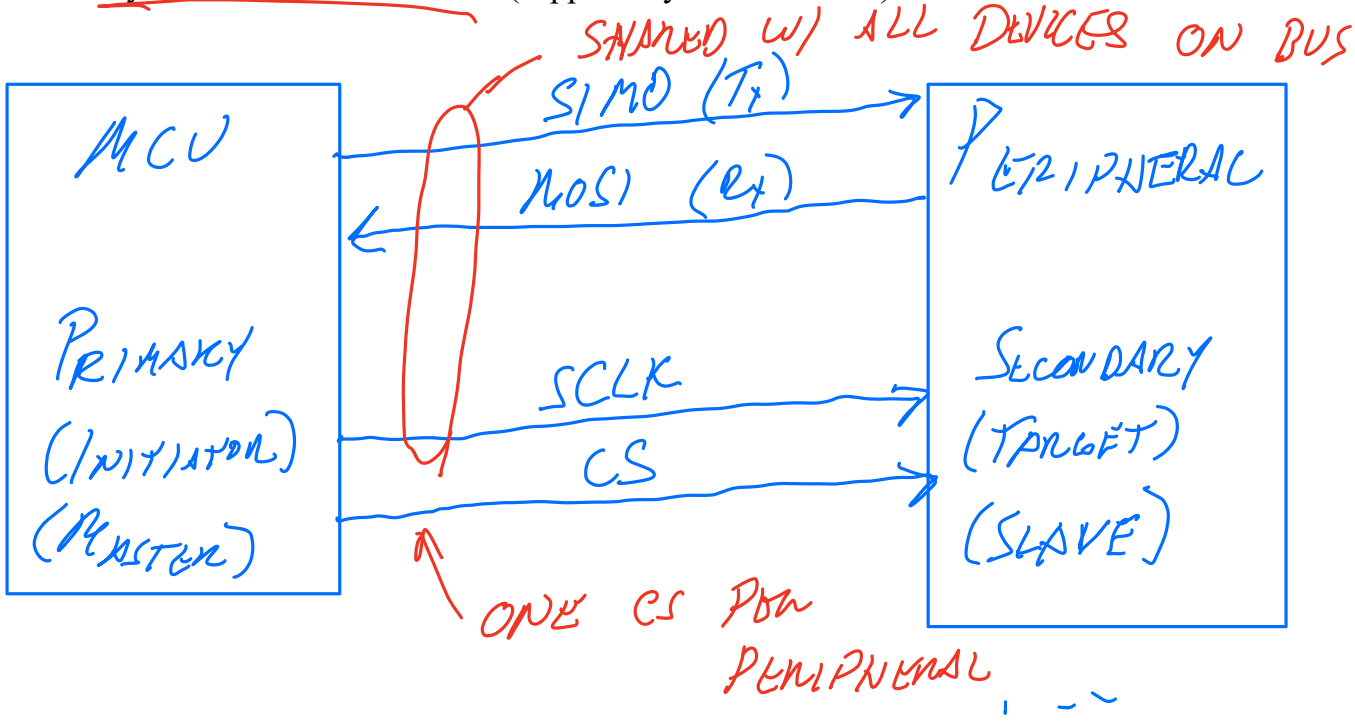

EXAMPLE UART TRANSMISSION
USB device installed as "/drive0".
0 1 0 1 0 1 1 0 0 1
START
LSB
MSB
STOP
- 1/9600 BAUD = 100μs/BIT
- LSB FIRST
BYTE: 00 11 01 01
35h = '5'
UART PARAMETERS
- 9600 BAUD
10 BITS TOTAL
- 1 START BIT (0)
- 1 STOP
- NO PARITY BIT
- 8 BITS SENT
New file name = Scope_0
Press to go /drive0
Spell s
Enter
Delete Character
Press to Save

6x
Nov 7

Serial Peripheral Interface Bus (SPI)

>> Used primarily for synchronous serial communications between a CPU and peripherals "within the box"

-- Synchronous = shared clock (supplied by master device)



>> Usually a 4 wire connection (sometimes 3-wire)

SECONDARY PRIMARY
SIMO = [redacted] In/[redacted] Out data line

SECONDARY PRIMARY
SOMI = [redacted] Out/[redacted] In data

SCLK = Serial Clock (Called `UCAxCLK` in MSP430 Documentation)

CS = Chip Select

↳ SELECTS WHICH DEVICE IS ACTIVE.

>> SPI is somewhat loose standard --> *Different* from I^2C

↳ DON'T NEED TO SPECIFY BAUD RATE/CLOCK FREQ

How will you know what to use?

--> SPI, I^2C, asynchronous serial (RS-232), other?
Sensors or other peripheral devices will specify the interfaces with which they are compatible

>> ***To use SPI the programmer must...***

1) Enable USCI_A or USCI_B for SPI mode (and set pins for function mode)
2) Select data format (data size, clock edge)
3) Configure a clock frequency

When communicating with a device, the programmer must also ...

4) Select desired SPI peripheral using its chip select (CS)

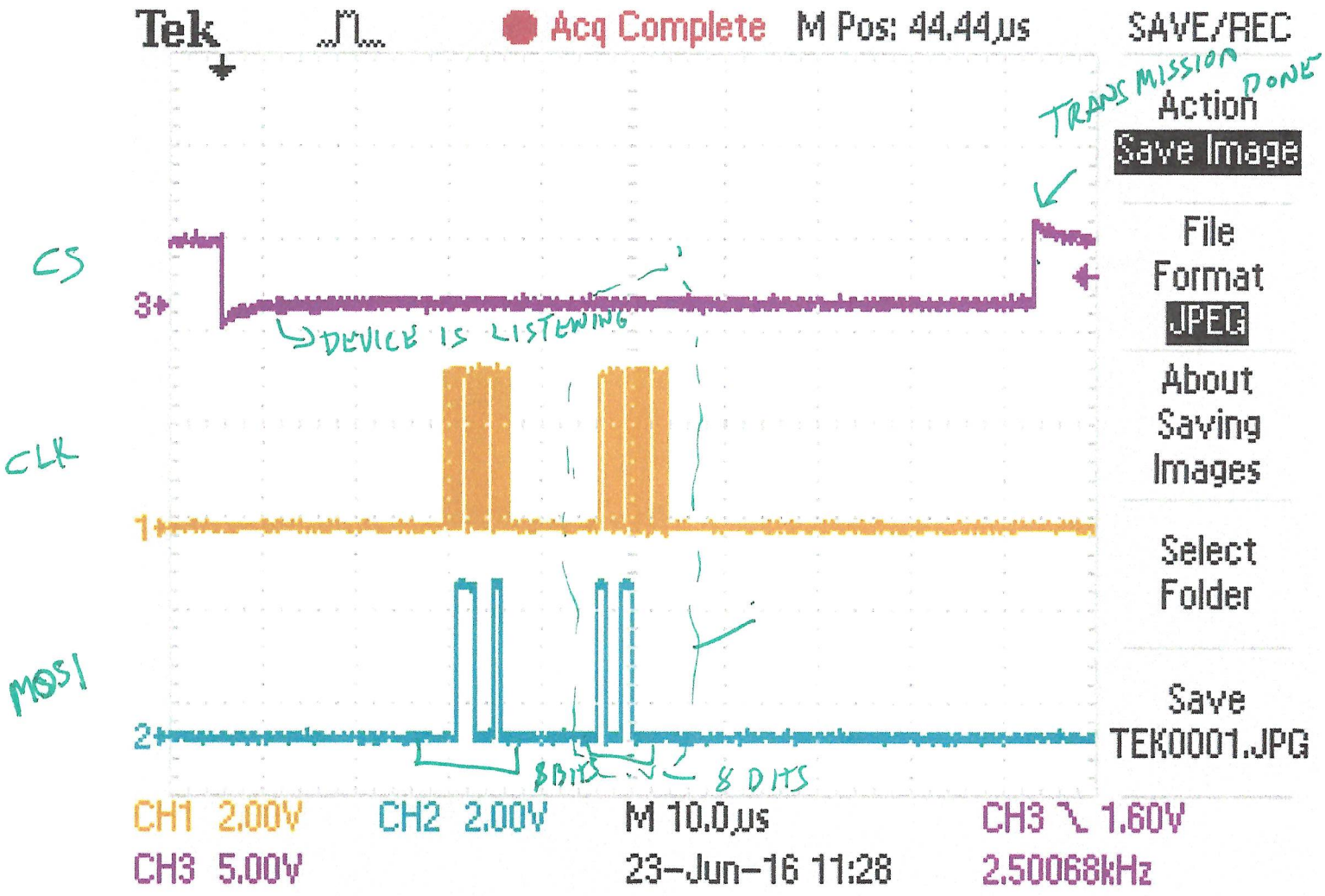
First, what is the function of a CS?

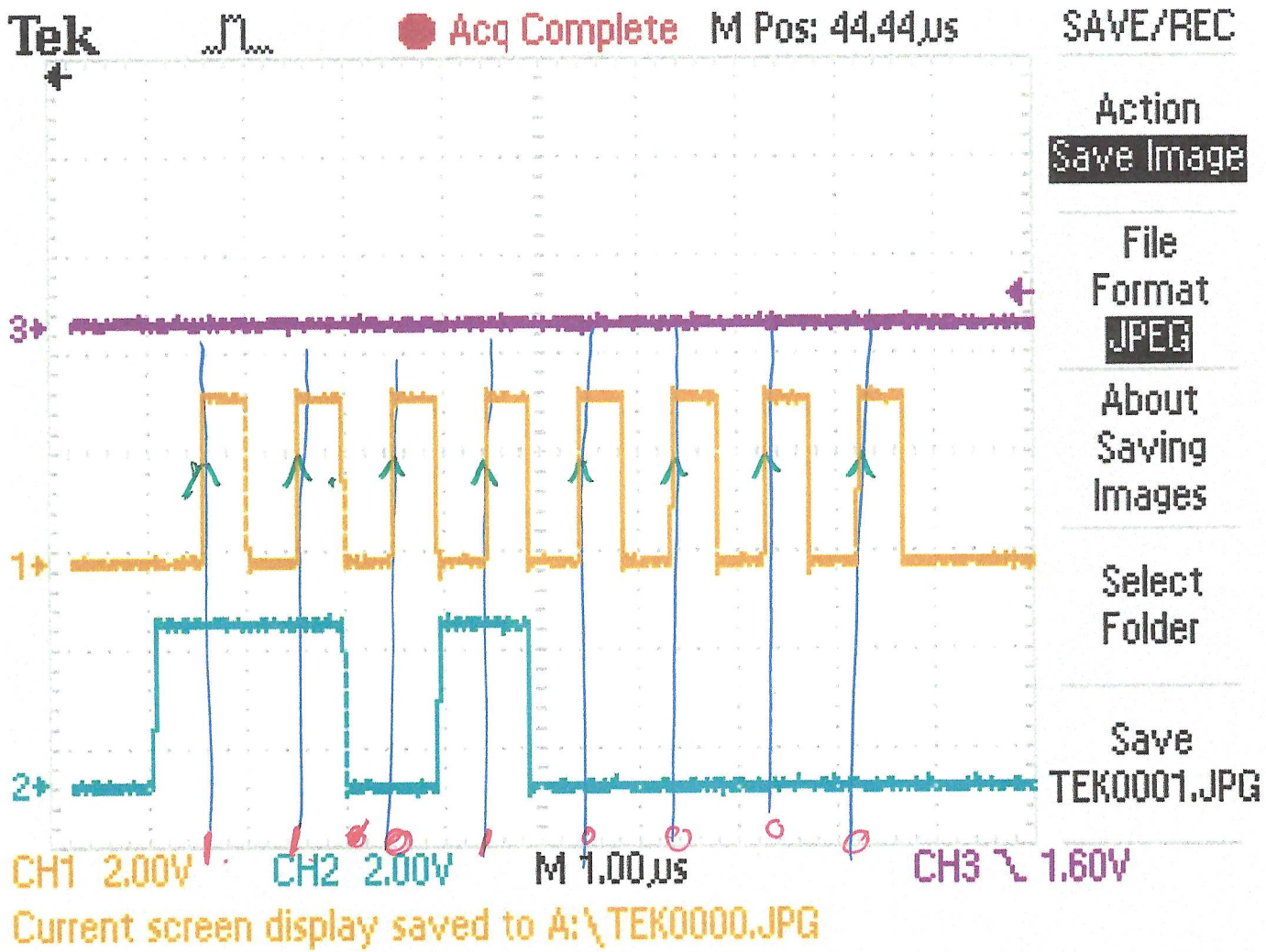
>> Likely to be multiple peripherals using SPI bus

>> Only 1 Slave device and Master (MSP430) can use SPI bus at a time

>> CS is typically an ACTIVE LOW signal implemented using a Digital IO pins
CS = 0 = Device is Enabled (will read and write to SPI data lines)
CS = 1= Device is Disabled (outputs are high impedance)

EXAMPLE SPI TRANSMISSION





NEED
- MSB FIRST (USUALLY)
- READ ON RISING OR FALLING EDGE

DATA: 1101 0000

D CH

Example Data format: MCP4921 Digital to Analog Converter (DAC)

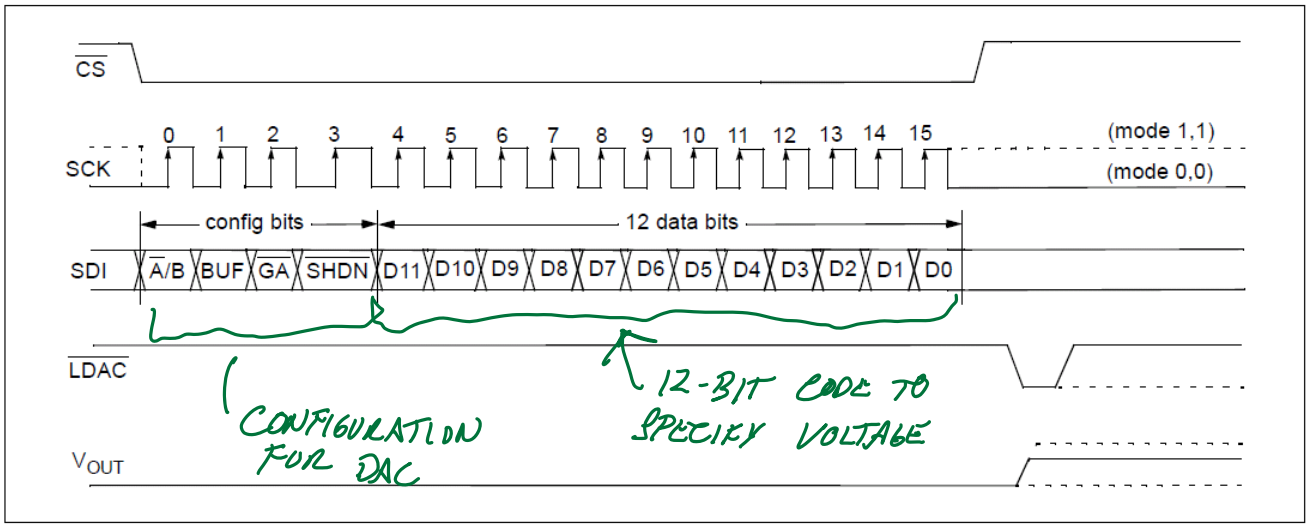
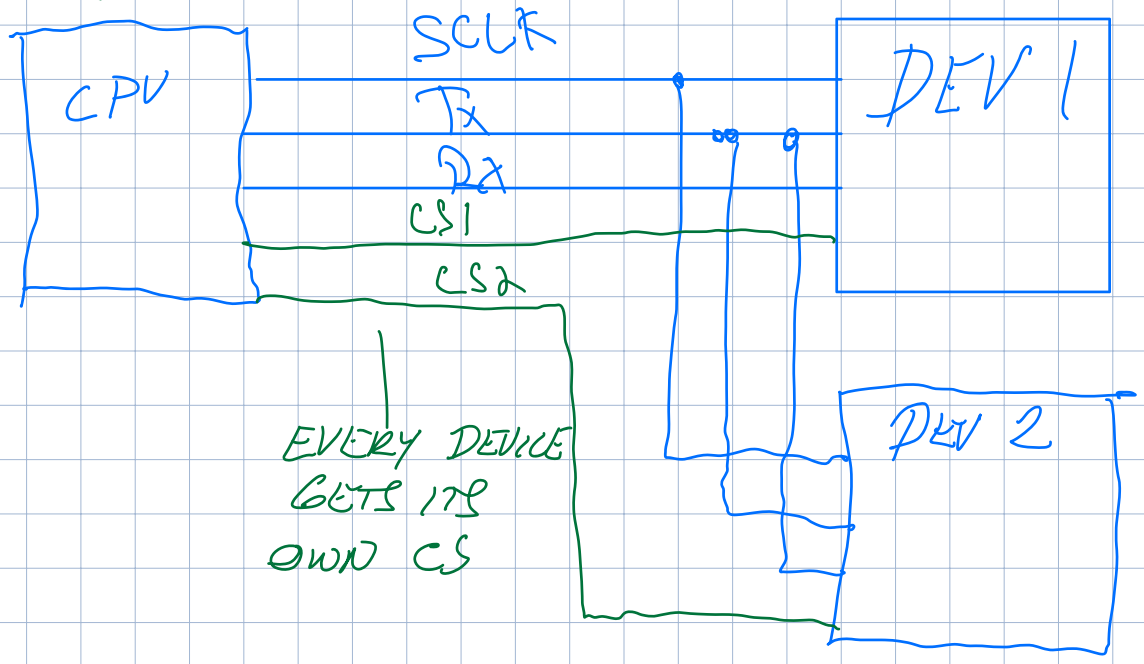


FIGURE 5-1: *Write Command.*

How do we connect multiple devices to the SPI bus?

- The "bus" lines (SCLK, MOSI, MISO) can be shared across all connected devices
- Each device needs its own chip select (CS) line, which tells which device when to wake up and watch the bus for input (or to write some output)

CONNECTING MULTIPLE DEVICES TO A SPI BUS



Example SPI Peripherals

Our lab boards have 2 peripherals that are SPI devices, the LCD screen and the digital-to-analog converter (DAC). However, we could readily connect others as the USCI pins and some digital IO pins are available through the headers

Sharp 96x96 LCD Display

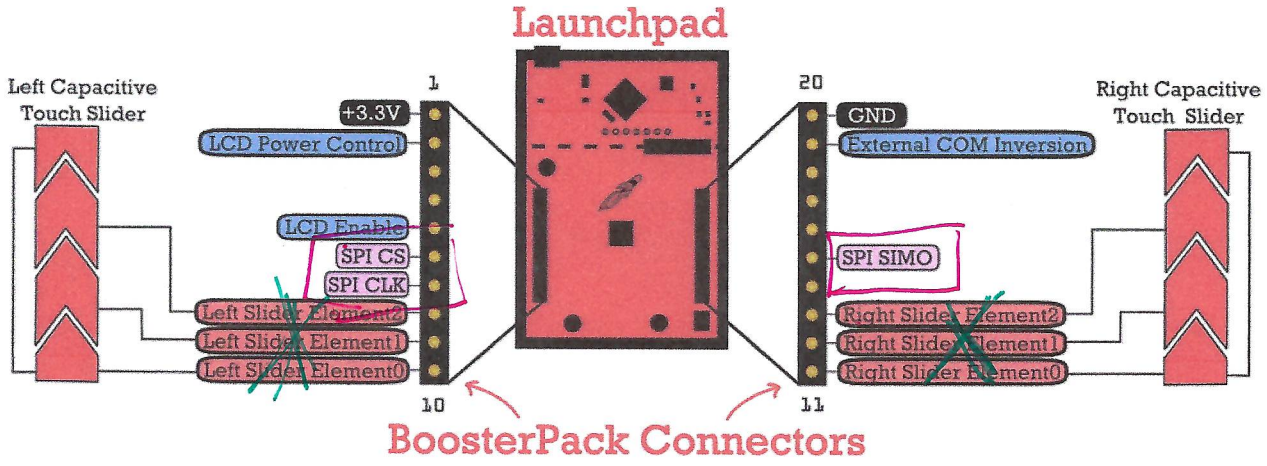


Figure 3. BoosterPack Default Pinout

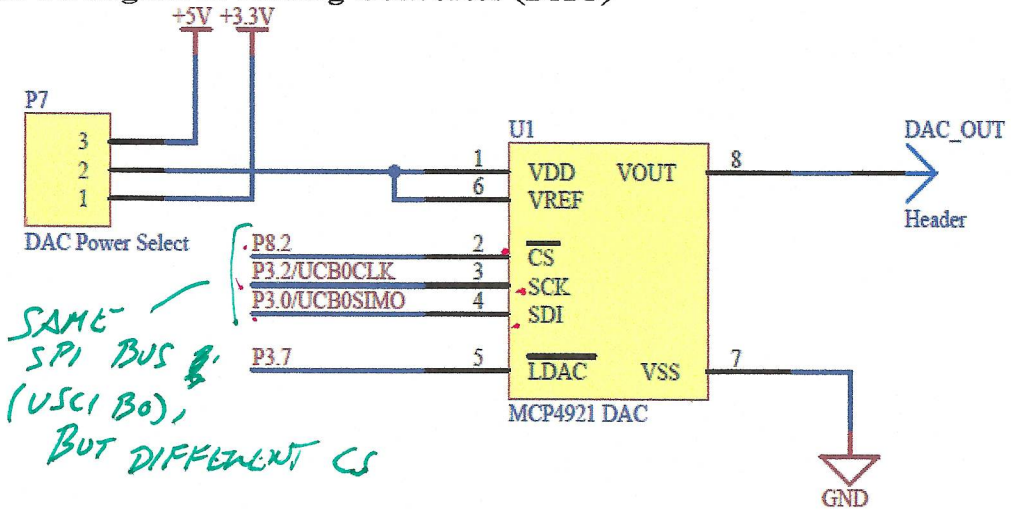
Interface Pins

CS IS JUST A DIGITAL I/O PIN

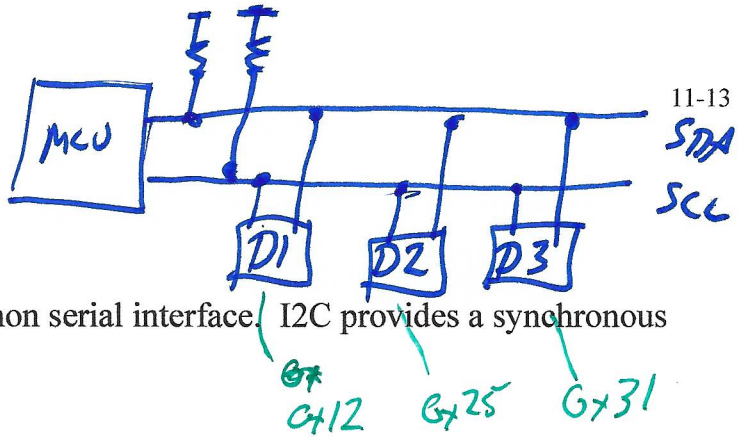
- **SPI CS: P6.6 (Digital I/O)**
- **SPI CLK: P3.2 (Function mode, UCB0CLK)**
- **SPI SIMO: P3.0 (Function mode, UCB0SIMO)**
- Two additional digital I/O pins to provide power and enable LCD (P6.5 and P1.6)

Note: LCD does not send data to the MSP430, so the SOMI line is not used!

MCP4921 12-bit Digital-to-Analog Converter (DAC)



Digital to Analog Converter: Converts 12-bit digital code into an analog voltage (in some range V_{Ref+} to V_{Ref-}). Sound familiar?
Can use this to generate analog signals!



Inter-Integrated Circuit (I^2C)

I^2C (also, I2C) is another increasingly common serial interface. I2C provides a synchronous interface using only two wires!

An I2C interface is comprised of two wires, called the "I2C bus":

- SCL: Serial clock
- SDA: Serial data

I2C bus fundamentals:

- All peripherals are connected to the same two wires
- Both SCL and SDA require *pull-up* resistors such that both lines default to logic high.
- Both lines are *bidirectional*
- Speeds are standardized: typically 100kbps (standard mode) or 400kbps (fast mode)

Unlike SPI, I2C is more rigidly standardized and has strict timing requirements on how the two lines can be used. There are a few standard operations:

- START
- STOP
- ACK: Acknowledge transmission
- NACK: Negative acknowledgement

Every device datasheet will tell you how and when it expects to receive these commands.

How are devices selected?

With only two wires, we have nothing like a chip select (CS) to wake up individual devices! In I2C, every device has an *address*, which is often programmed at the factory.

Before any transmission, the master sends the address of the device it wants to use—the device should only respond to the request if it has a matching address.