ECE 2049 Lecture 13

Office Hours

- Today: 5-7 PM EDT
- Wed: - 1-3PM official + I will be around
- Thurs,: 2-4PM, 5-7PM EDT

Administrivia

- Exam 2: Out tonight, due **Thurs 7/9 11:59 PM EDT**
  - Same format as last time
  - Designed for 1-2 hours
  - Problems like HW, + short answer questions
    - May need to read/write a bit of code.
- Lab 2, Stage 2 (+ Stage 3) Due **Sunday, 7/13 11:59 PM EDT**
  - Feel free to ask if you have questions (Signoff + Report)
- Lab 1: Grades by tomorrow
  -
- Lecture on Thursday: Course discussion
  - Informal: I will talk about other ECE/CS classes ~~and~~ you might like
  - Can talk about class or anything
    - Can post questions to Piazza

- THIS IS OUR FINAL WEEK
  - IF YOU HAVE ANY MISSING/OUT OF DATE GRADES (EXCEPT LAB1, LAB2), LET ME KNOW.

- IF YOU HAVE ANY OUTSTANDING LABS (NOT LAB 2) AND YOU HAVE NOT CONTACTED ME ALREADY, YOU SHOULD DO SO ASAP!!!

Exam 2 Topics

HW 4-6
Lecture & onward

- Clocks
- Interrupts + How they work
- Timers
  - Read/Write Config
  - UTC Time Conversion
  - Using the Timer ~~for~~ variable (global count)
- ADCs
  - Concepts: Not required to know how to read/write ADC 12 Config.
  - Sensor data ↔ Voltage ↔ Codes
  - Reasoning about sensors
- Power modes
- Basic concepts about digital interfaces

ECE2049 -- Exam 2 Review
Clocks, Timers, ADCs, and Operating Modes

Problem 1: Answer the questions below completely.

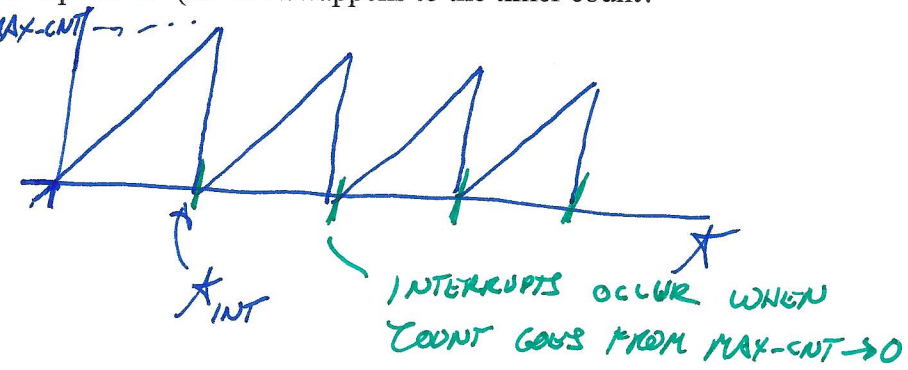
a. How does the CPU know where to go on an interrupt? How does it find the correct ISR?

INTERRUPT VECTOR TABLE MAPS ISR FUNCTION TO ADDRESS OF ISR IN CODE MEMORY.

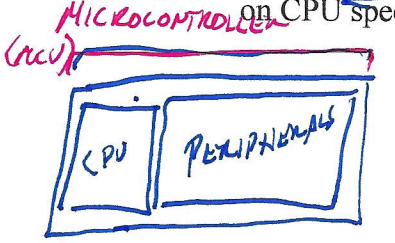
EX. TIMER, ADC, I/O PINS, ETC

b. Explain the operation of a Timer in "up mode" (ie. What happens to the timer count? When is an interrupt triggered?)

IN UP MODE, TIMER COUNTS FROM 0 TO MAX-CNT



c. True or False: The operation of peripherals like the Timers or ADC12 causes a big drain on CPU speed. Why or why not?



TIMERS + ADC ARE "PERIPHERALS" — HARDWARE. THEY OPERATE INDEPENDENTLY OF THE CPU!

d. True or False: Interrupt Service Routines (ISRs) should be kept short because the CPU will stop executing them after 255 instructions.

FALSE! ISR CAN TAKE A LONG TIME, BUT MUST BE DONE BEFORE NEXT INTERRUPT ARRIVES.

Module 11. Intro to Digital Interfaces

Digital Interface: Connection between two or more digital devices.

INSTEAD OF VOLTAGE SCALE

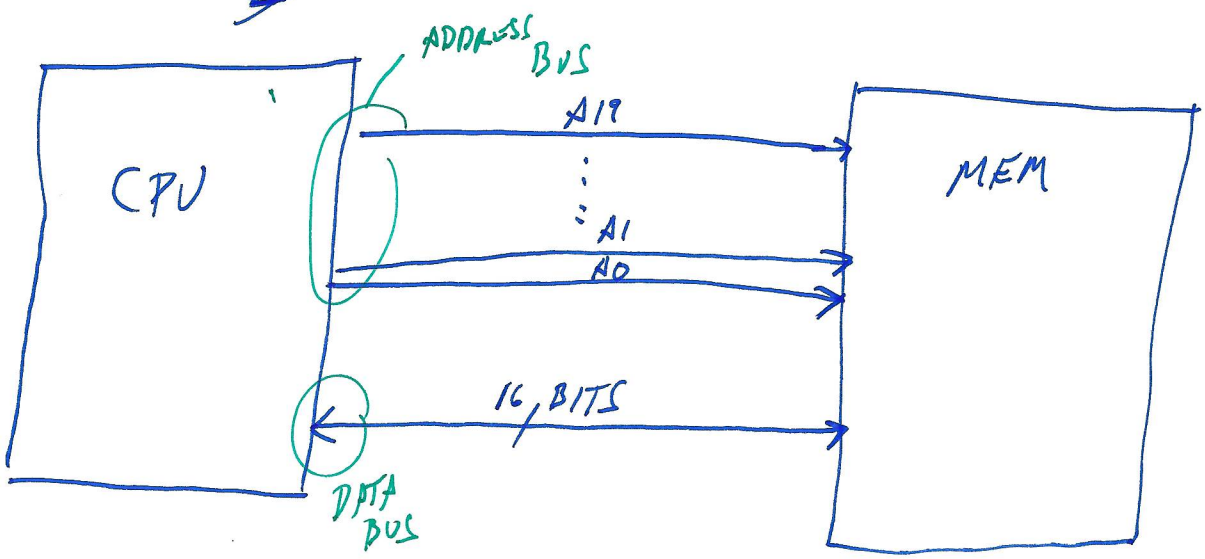
ENCODE INFO IN DIGITAL LOGIC (JUST LIKE A CPU)

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: I/O (CPU TO MEMORY + PERIPHERALS

- INSIDE CPU ITSELF

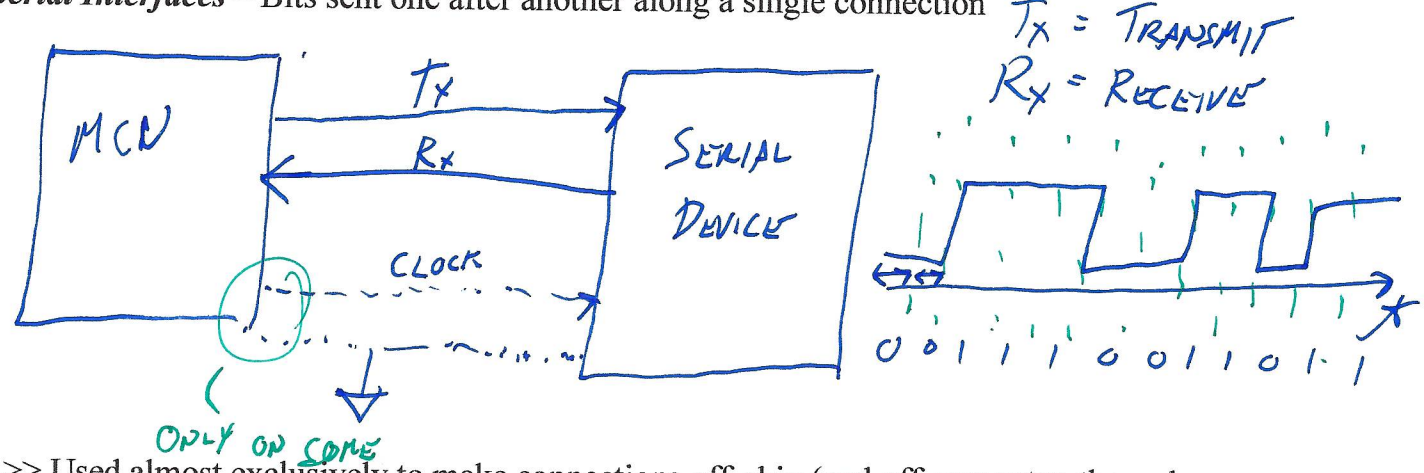
  (REGISTERS TO ALU)

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

– INEFFICIENT IN "REAL ESTATE"

– HARD TO ADD SO MANY PINS TO A CHIP, OR EVEN INSIDE A CHIP.

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



>> 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

USB2.0 480 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

(1 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

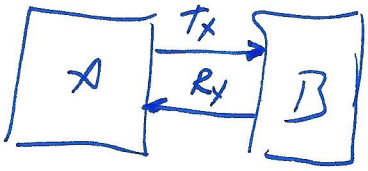
>> Most modern microprocessors/microcontrollers will have built-in Serial interface
-- MSP430F5229 has a total of four, in two types

>> 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:



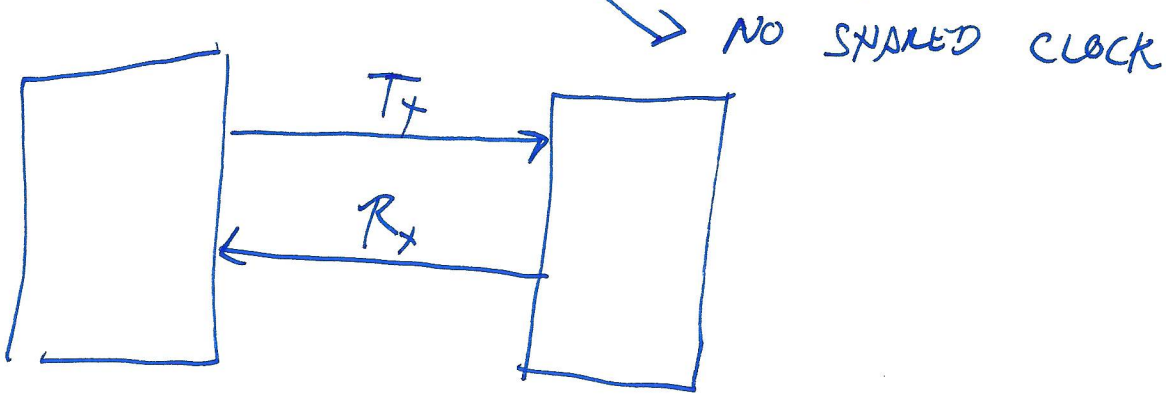
- WE CAN SEND WHATEVER WE WANT

- 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

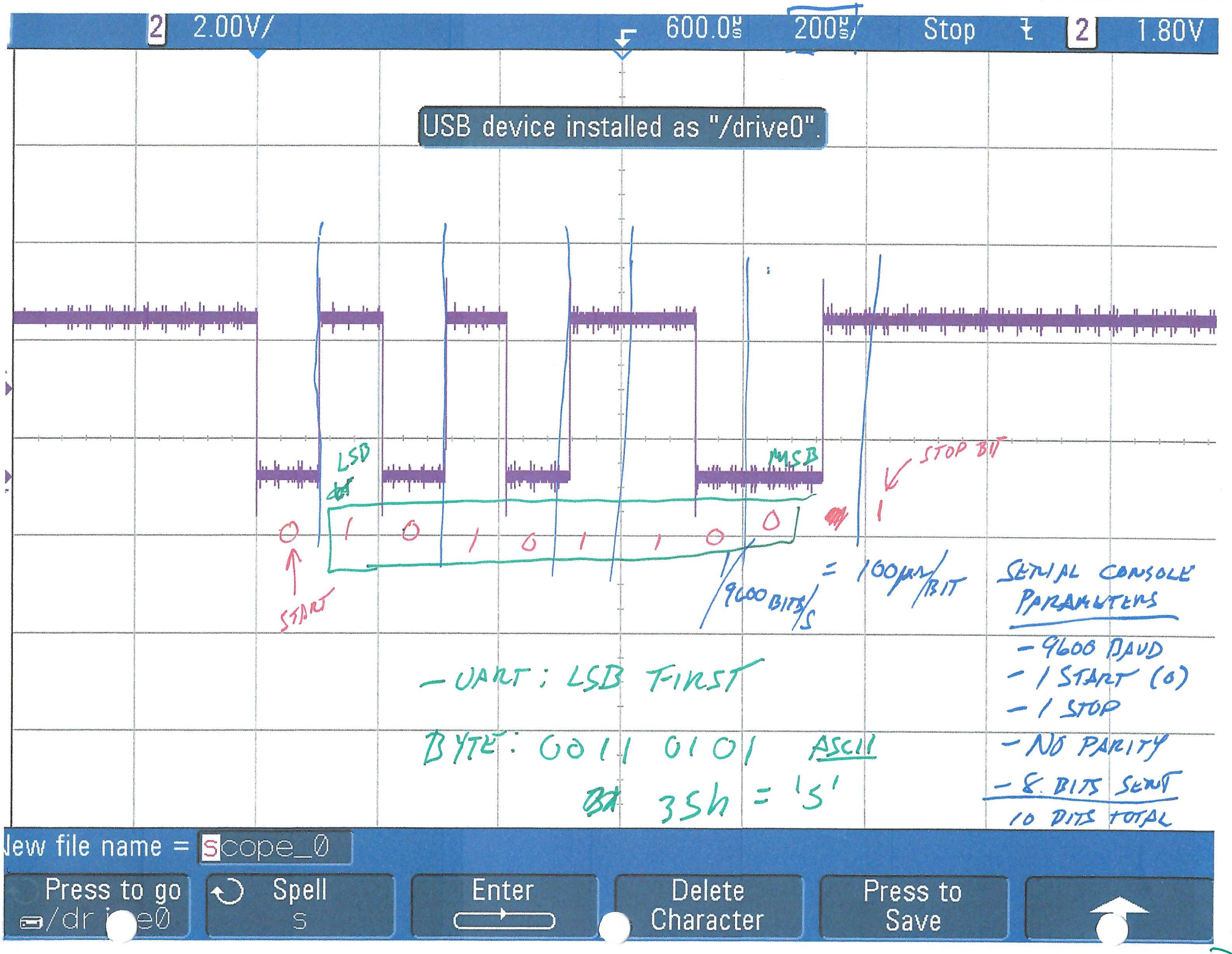
In order to use UART for an application, you need to determine the following parameters:

- Start Bit = 1 bit ("low")
- Data Bits = 7 or 8 bits
- Parity = Even, Odd, or None ← CAN BE USED FOR ERROR CHECKING.
  >> 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

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**

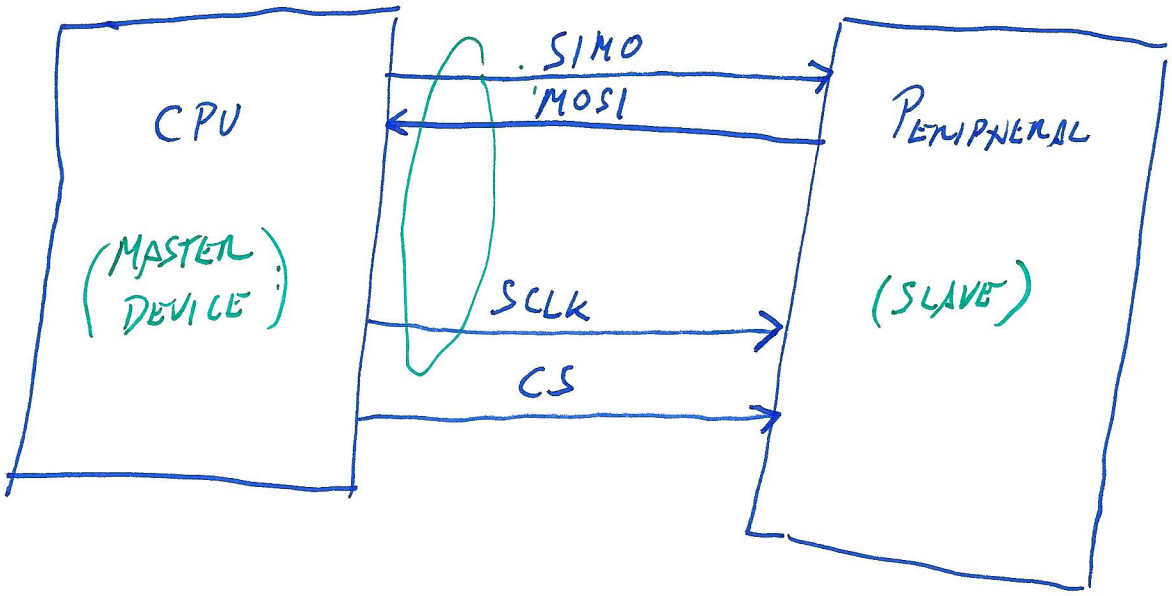
** Most UARTs send asynchronous serial data LSB first (because RS-232 is LSB first) but MSP430F5229 UCSI_A is configurable and can be set to send MSB first, so that it can be compatible with various types of serial interfaces.


2 2.00V/
600.0µs
200µs/
Stop
2 1.80V
USB device installed as "/drive0".
LSD
MSB
STOP BIT
0 1 0 1 0 1 1 0 0 1
START
9600 BITS/S = 100µs/BIT
SERIAL CONSOLE PARAMETERS
- 9600 BAUD
- 1 START (0)
- 1 STOP
- NO PARITY
- 8 BITS SENT
10 BITS TOTAL
- UART: LSB FIRST
BYTE: 0011 0101 ASCII
35h = '5'
New file name = scope_0
Press to go /drive0
Spell s
Enter
Delete Character
Press to Save

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)

SIMO = Slave In/Master Out data line

SOMI = Slave Out/ Master In data

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

CS = Chip Select

↳ SELECTS WHICH DEVICE IS ACTIVE

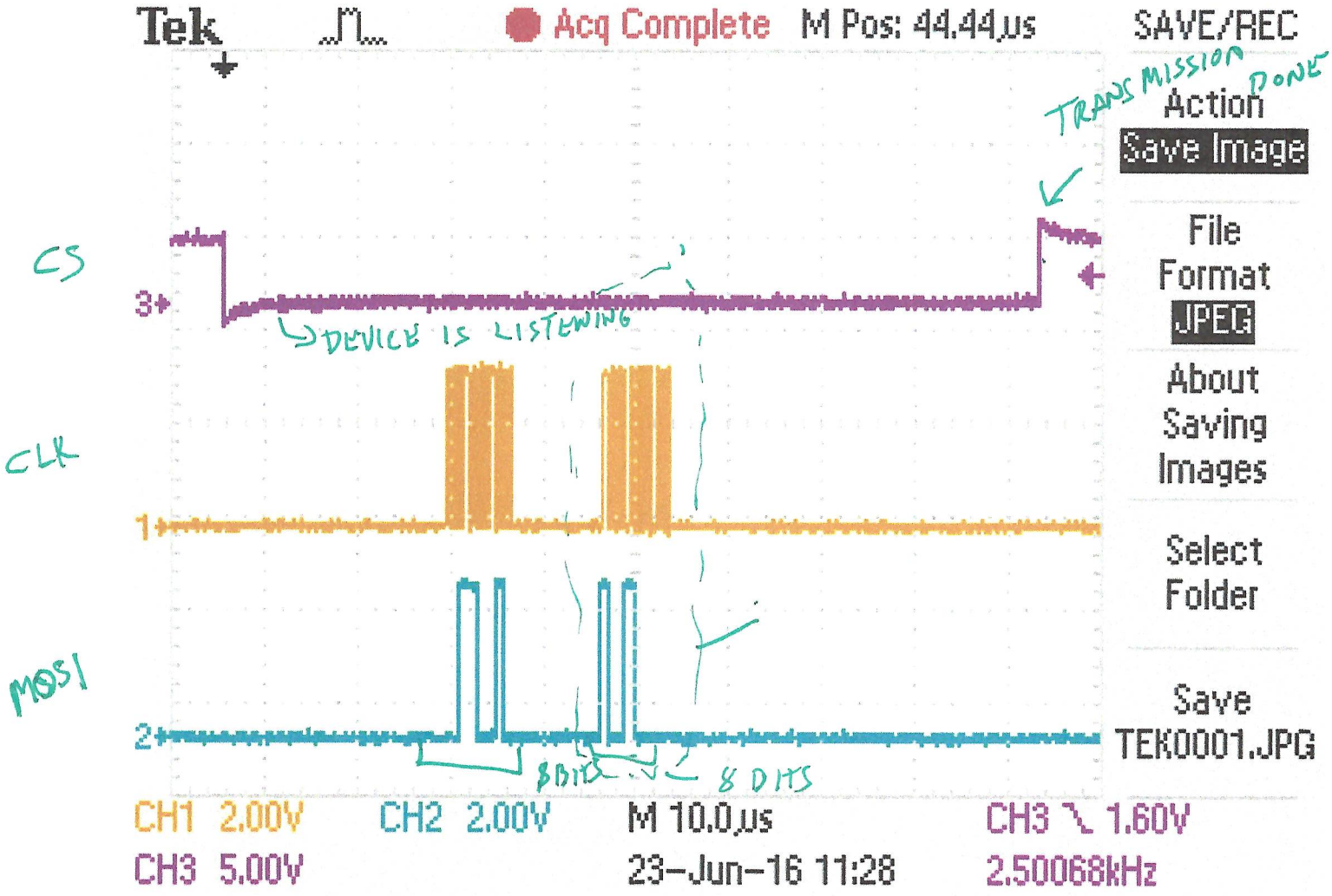
>> SPI is somewhat loose standard --> *Different* from I^2C

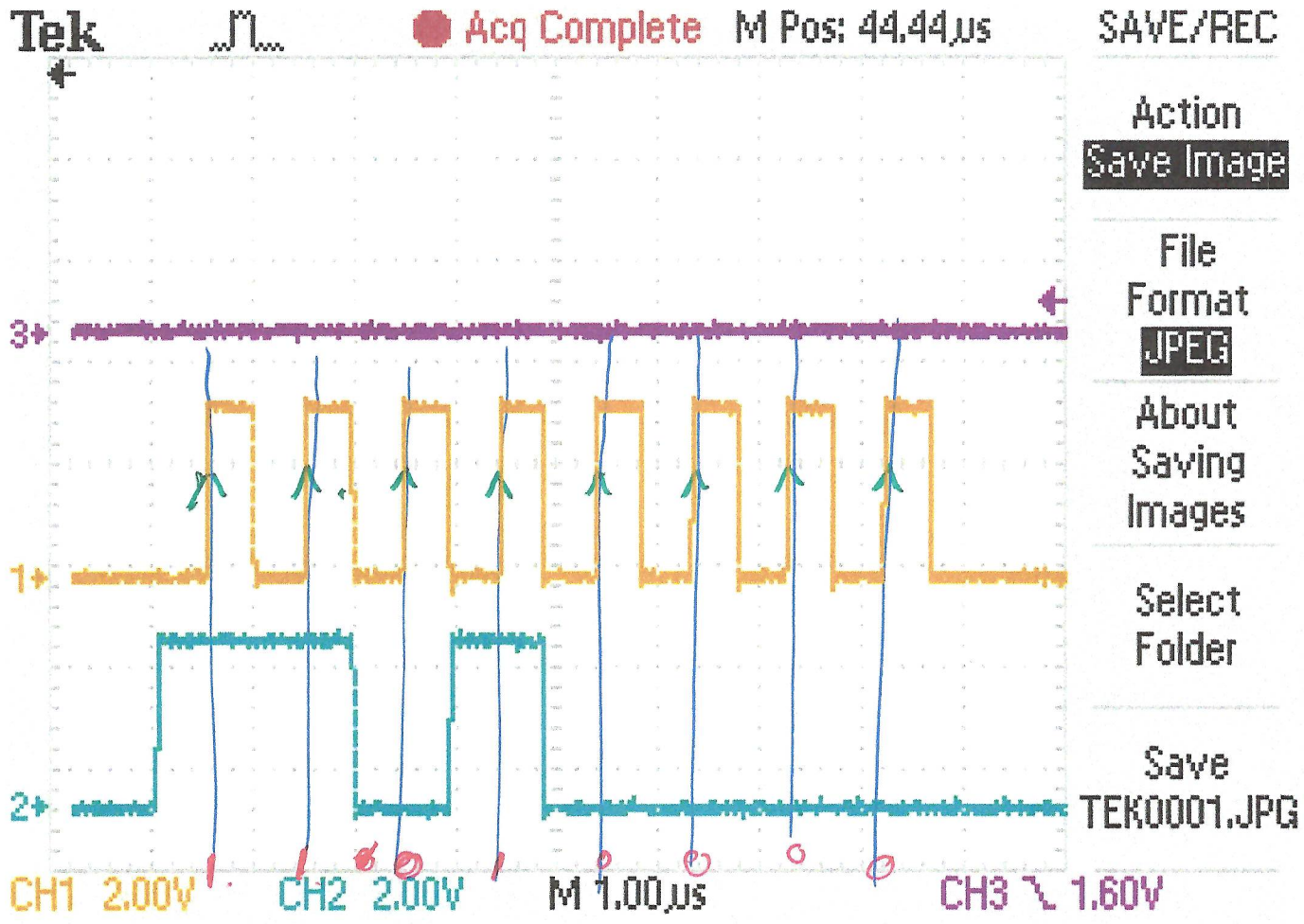
↳ DON'T NEED SPECIFIC BAUD RATE OR CLOCK FREQ.

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

EXAMPLE SPI TRANSMISSION





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

DATA: 1101 0000

D0h

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

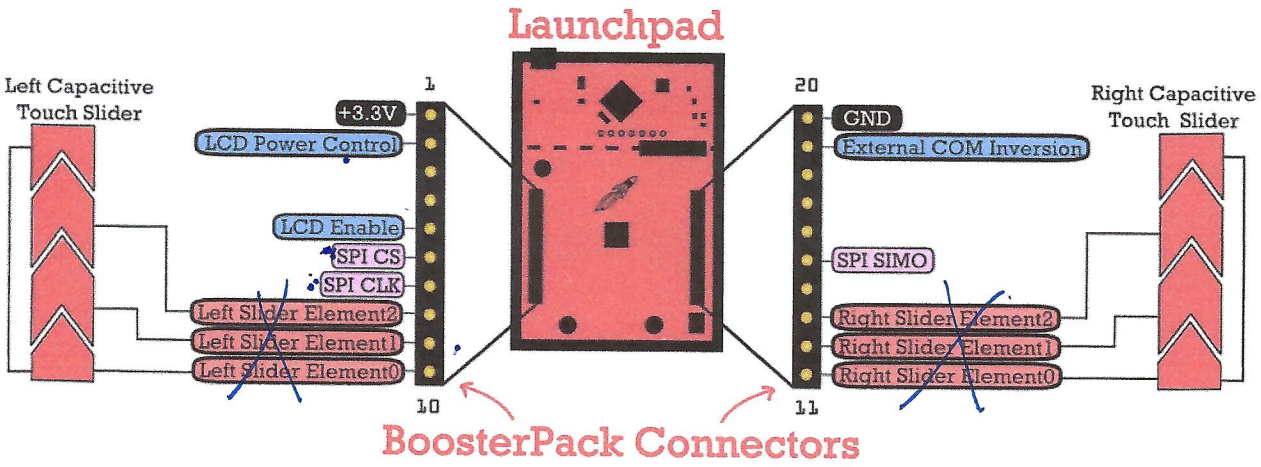


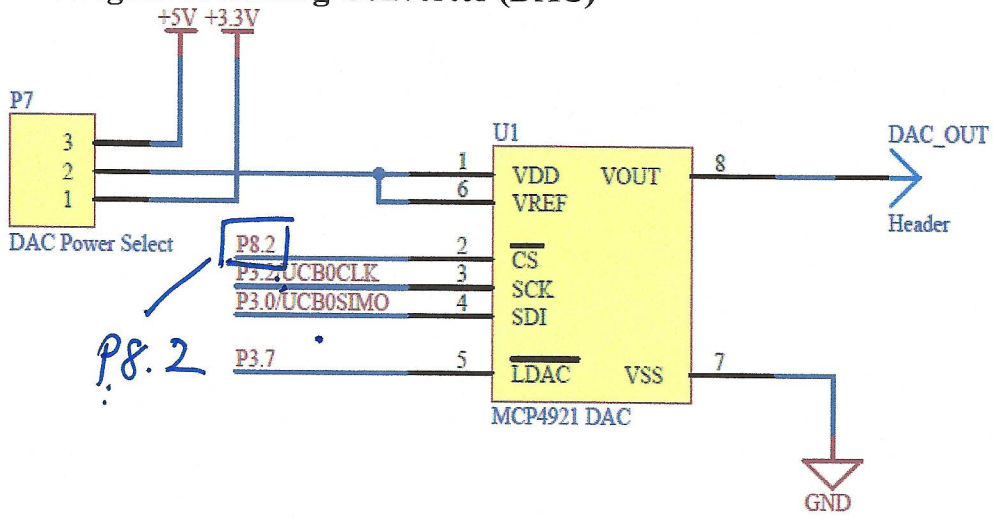
Figure 3. BoosterPack Default Pinout

Interface Pins

- **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!

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

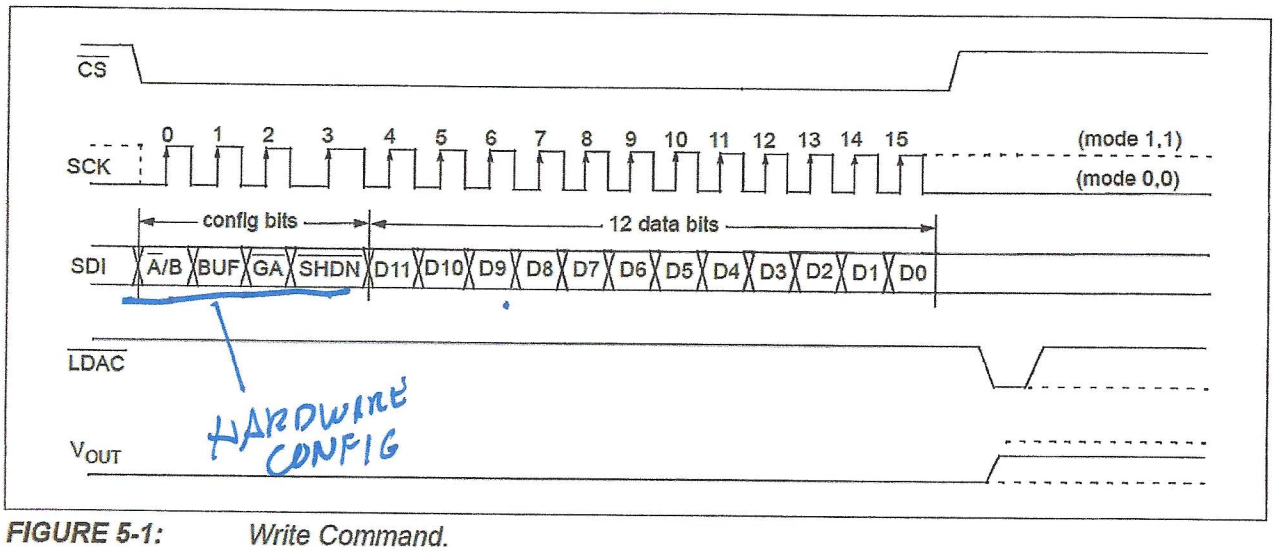
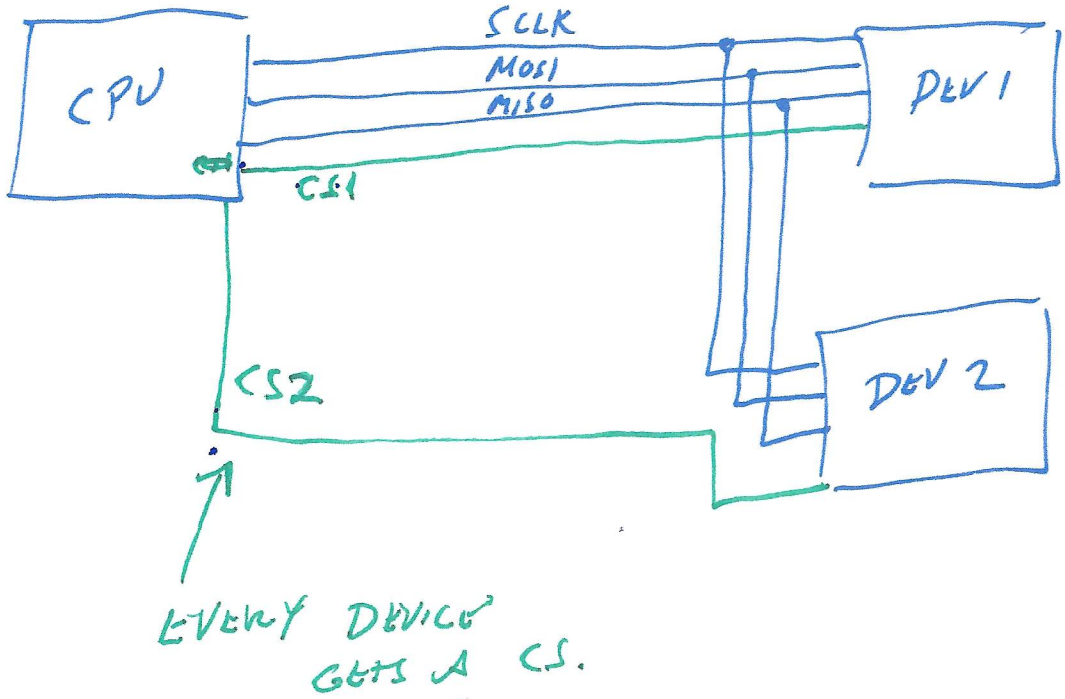


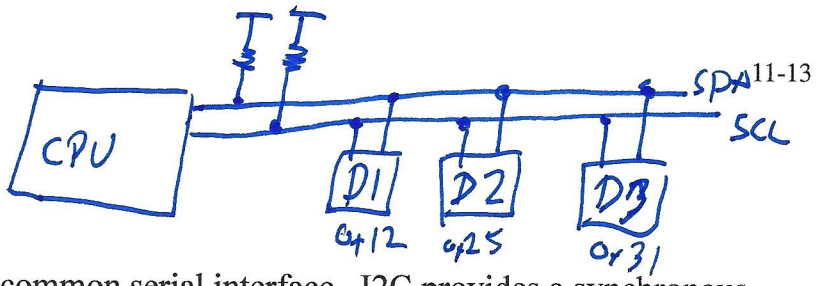
FIGURE 5-1: *Write Command.*

HERE, DAC EXPECTS 16 BITS:
- BITS 15-12: HARDWARE CONFIG.
- BITS 11-0: DAC CODE.

CONNECTING MULTIPLE DEVICES TO SPI BUS:



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.