Last Time

- Timers and interrupts
- Pulse width modulation
- DC motor control and H-bridges

Today

- Device-to-device communication: buses
- Project 4

Administrivia

- Project 3 due today
  - Demonstration
  - Group report
  - Personal report
- Reading:
  - Today: ESA 7
  - Thursday: ESA 9-9.2.1

RC Heli Example

RC Heli Example II

Device-to-Device Communication

Device communication occurs at multiple levels:

- Within the processor:
  - CPU to memory
  - CPU to serial I/O device
- Processor to processor:
  - Each sensor could have its own processor
  - Some process must then coordinate sensor data from various sources to make control decisions

Device-to-Device Communication

It is common for this communication to take place on a **bus**:

- Often consists of multiple (parallel) wires
- Allows various devices to drive the bus (but at different times)
- Allows for a certain degree of flexibility: devices can be added or removed at boot time or (in some case) real time

Buses

System buses

- Data bus
- Address bus: transmission of element address information (e.g., a memory address)

- High-speed and local to CPU

Buses

Examples: backplane

Connect CPU with memory and I/O control devices (video, audio, disks, etc.)

- PCI: (what you have in your PCs)
- ISA: the old PC standard
- VME: a competing standard (industrial control)

- Local to computer

Buses

Examples: I/O

Connect computer with external devices

- Universal Serial Bus (USB)
- Firewire
- I^2C
- SPI
- CAN

- External to computer

Bus Control

Critical bus issues

- Who is in control?
  - Solved with some form of **bus arbitration** mechanism
- How is timing determined?
  - Typically make use of a clock

Bus Arbitration

- Master device: can initiate a transfer of data over the bus
- Slave device: must wait for transfer to be initiated

- When there are multiple masters: must have some way to determine which one is in control **now**

Bus Arbitration

- Central control: a single device decides
- Serial control:
  - Control is given to one master
  - If it does not need it, control is passed to the next master
- Self-selection: the masters decide for themselves
- Preemption: interrupting one master to allow another to take control

Bus Use

Once arbitration is complete, we have:

- One device that will write to the bus
- One device that will read from it
- (in some cases, one device will both read and write during a data exchange)

- Clock signal tells the reader when data from the writer is valid

Bus Use

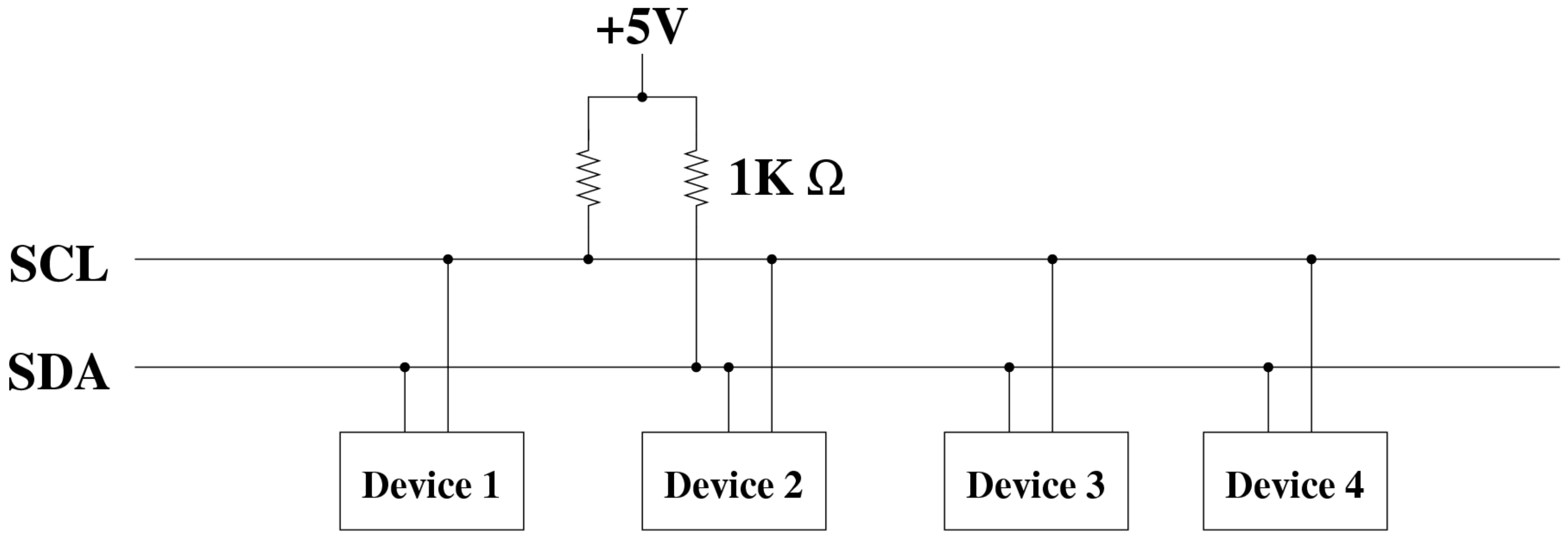
Protocol: rules for how data will be exchanged

- How is the slave specified? (addressed)
- What do each of the bytes mean?
- When will they be transmitted?
- How does the writer know when the reader has received the data?

Inter IC (I^2C) Bus

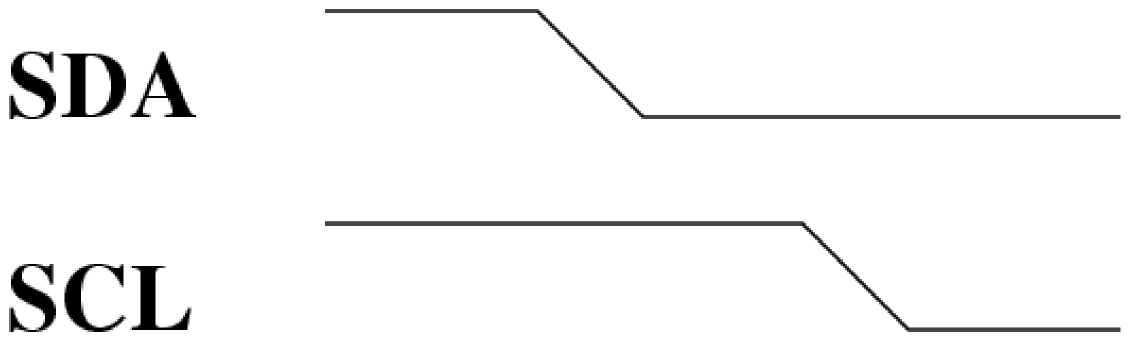
- Allows for multiple masters
  - Arbitration: self selection
- Two wires only!
  - Data transfer: SDA
  - Clock: SCL (up to 400 kbs)
  - (and a common ground)
- Slave addresses are transmitted on SDA before other data is sent

I^{2}C Bus



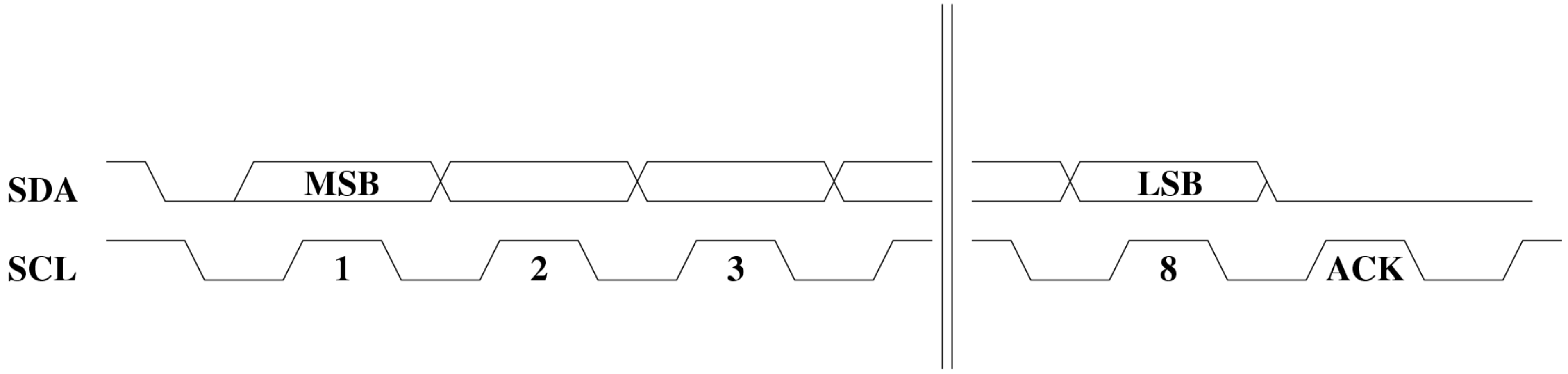
- The writer of a line may pull the line low (otherwise, it leaves the line floating)
- Master always runs the clock

I^{2}C Bus: Start Condition



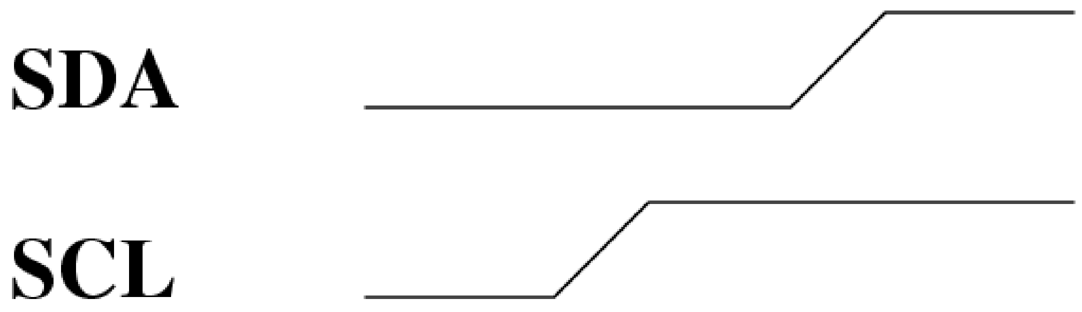
- Master starts the transmission by bringing SDA low and then SCL low
- Constantly checks the state of SCL to make sure that another has not brought it down
  - Aborts transmission if another master has started a transmission

I^{2}C Bus: Transmission of a Byte



- Master generates SCL
- Either master or slave produces the data (depending on the situation)
- ACK: acknowledgement bit

I^{2}C Bus: End Condition



- Master allows SCL to go high, and then SDA (order is important)
- After this operation, any other master can take control of the bus

I^{2}C: Transmission of a Packet

Data packet:

- A collection of bytes
- Each byte has a specific meaning (defined by some higher-level protocol)

- With I2C, the first byte of a packet contains address information (7 most significant bits)
- The 8th bit (least signficiant) determines whether this is a read or a write operation

I^2C: Master Writes to a Slave

- Start condition
- Byte 1: Address with (LSB=0: indicates write)
  - Slave acknowledges byte (ACK bit = 0)
- Byte 2: First data byte
  - Slave acknowledges byte (ACK bit = 0)

  :
- Byte N+1: Nth byte; ACK=0
- Stop condition

I^{2}C: Master Reads from a Slave

- Start condition
- Byte 1: Address with (LSB=1: indicates read)
  - Slave acknowledges byte (ACK bit = 0)
- Byte 2: First data byte from slave
  - Master acknowledges byte (ACK bit = 0)

  :
- Byte N+1: Nth byte; ACK bit =1 (last byte)
- Stop condition

I^2C Devices

- Can have a large number of I^2C devices on the same bus
  - Restrictions as to bus size

- EEPROM
- Digital potentiometers
- Thermal sensors
- Compasses
- A2D/D2A devices

I^2C on the Atmels

- Hardware support on the chip
- Can be configured as either a master or a slave

- No OUlib support yet…

Project 4

Want: FSMs with wind direction sensor

Reality: only 2 functional encoders left

Solution: groups may attempt one of two projects (which one is up to you

- Project 4a: FSMs and wind direction
- Project 4b: FSMs only

Project 4a: Robot Task

- Initial condition: robot can see beacon to the front
- Move forward until a beacon is on the left OR right
- If right: move downwind until beacon is observed on left
- Move upwind until beacon is observed on right

Project 4b: Robot Task

- Initial condition: robot can see beacon to the front
- Move forward until a beacon is on the left OR right
- If right: continue moving forward until a new beacon is observed on the right
- Orient and move toward this beacon

Group Goals for Today

- Which project will you choose?
- What does the FSM look like?

Next Time

- Operating Systems