Lecture 12: Board Level Serial Com. i²C / SPI

Vikram Iyer

Adapted from material by Blake Hannaford

Announcements and reminders

Lab 2 is posted, due 5/8
- Deadline extended to Monday
- Tip: for scheduling, start with simple tasks

Midterm on 5/12
- Next Friday, here (MOR 230) in class
- Everything through today
- C programming
- Number representation and logical operators
- Pointers
- Schedulers
- Interrupts

Last time and plan for today

- Last time
  - Reading analog data and working with the joystick
  - Review context switching
  - Concept of stack
  - Introduction to interrupts

- Today
  - Code examples
  - Communication protocols (I2C, SPI)

Motivation

PCB Complexity

High pin-counts

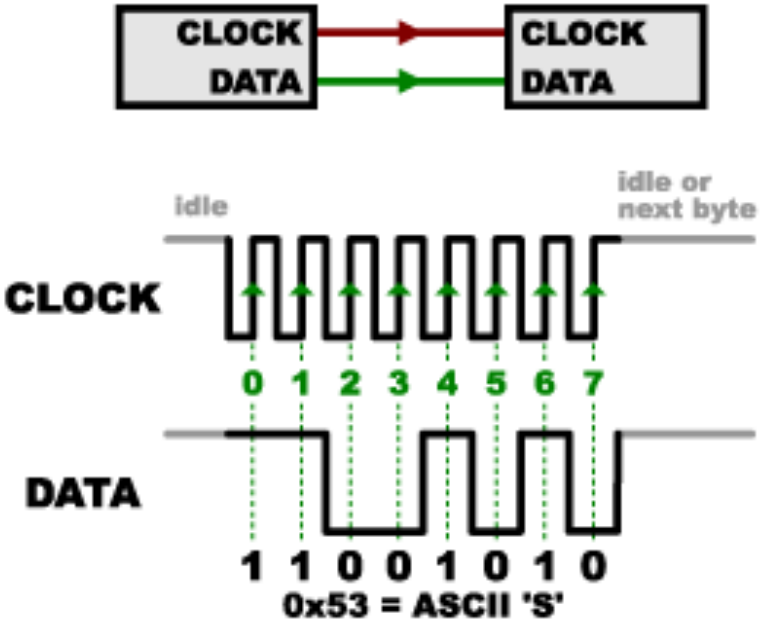
Ex: Google Pixel 5 Motherboard

(source: PBK.com
via arstechnica.com)

How can we send data between chips?

How can we send data between chips?

How can we send data between chips?

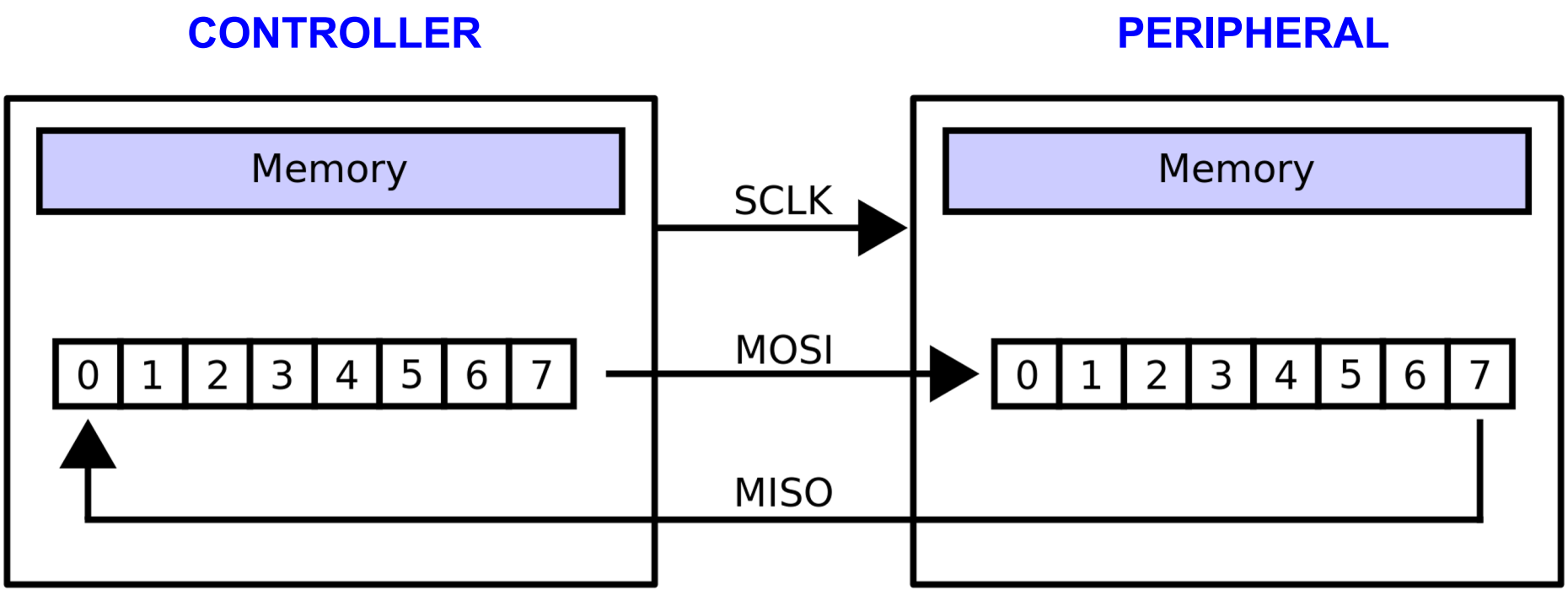


SPI: Serial Peripheral Interface

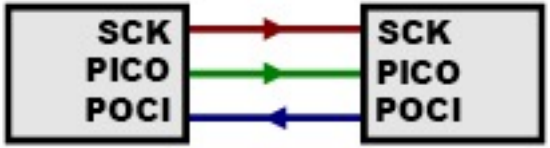
- Purpose: to communicate between ICs on a board with fewer pins and traces.
- Developed by Motorola in 1980's
- Reference: Wikipedia
- 4-wire, Serial, "data-loop"

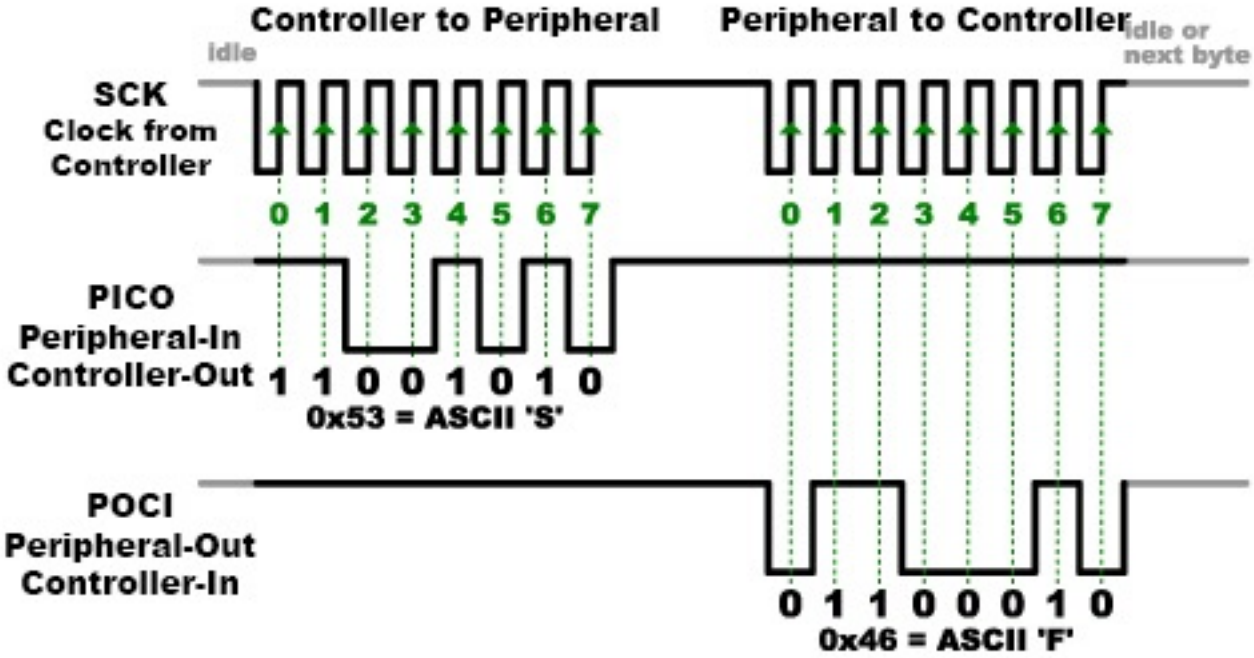
Obsolete Name	Replacement Name
Master	Controller
Slave	Peripheral
MISO	POCI
MOSI	PICO
SS	CS

SPI Basic data loop

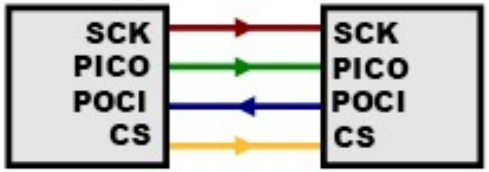


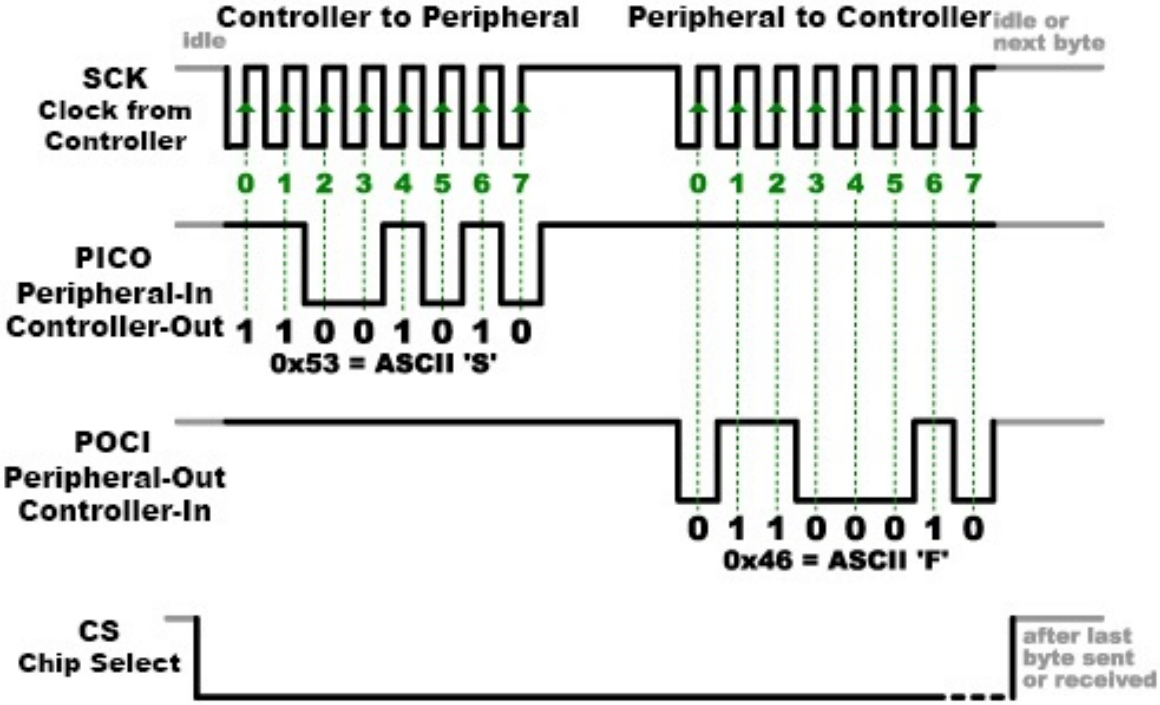
3-Wire SPI


SCK
PICO
POCI
SCK
PICO
POCI

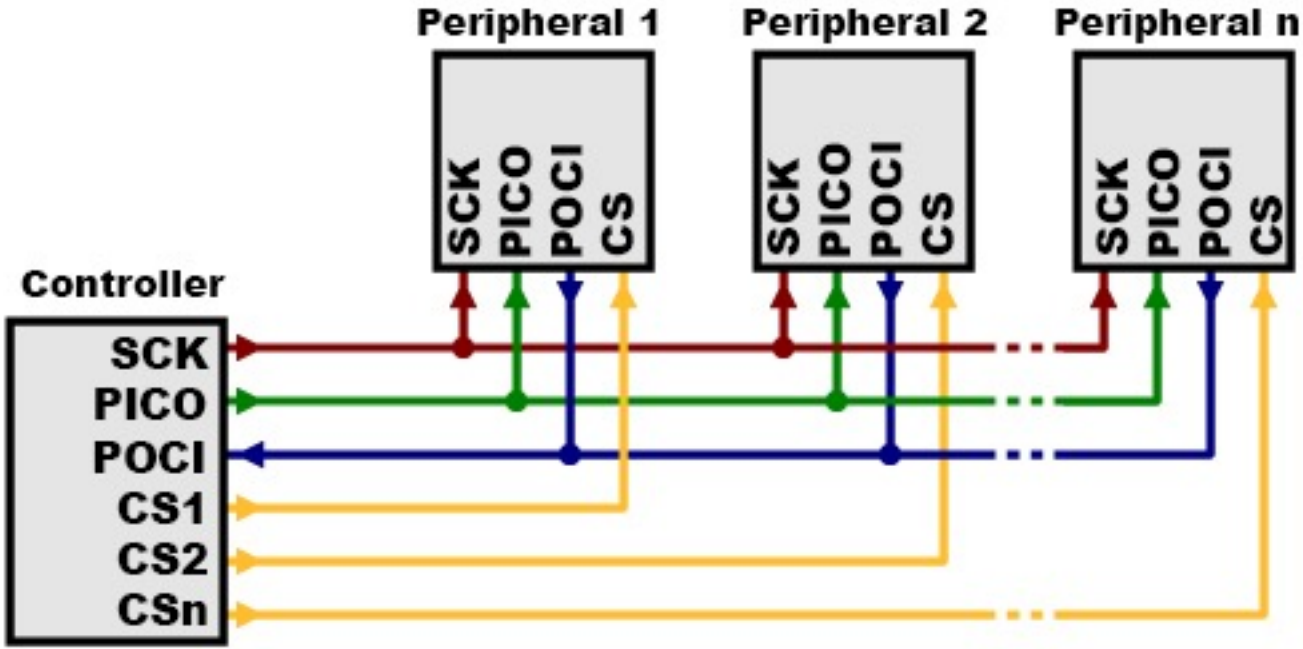

Controller to Peripheral
Peripheral to Controller
idle
idle or next byte
SCK
Clock from Controller
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
PICO
Peripheral-In
Controller-Out
1 1 0 0 1 0 1 0
0x53 = ASCII 'S'
POCI
Peripheral-Out
Controller-In
0 1 1 0 0 0 1 0
0x46 = ASCII 'F'

4-Wire SPI


SCK
PICO
POCI
CS
SCK
PICO
POCI
CS

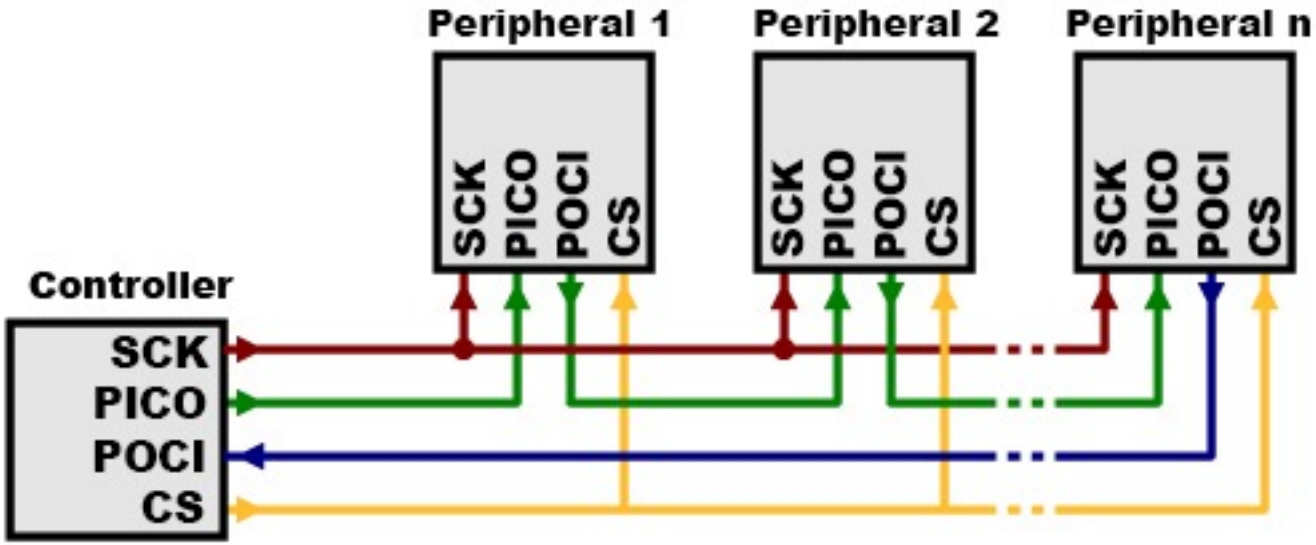

Controller to Peripheral
Peripheral to Controller
idle
idle or next byte
SCK
Clock from Controller
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
PICO
Peripheral-In
Controller-Out
1 1 0 0 1 0 1 0
0x53 = ASCII 'S'
POCI
Peripheral-Out
Controller-In
0 1 1 0 0 0 1 0
0x46 = ASCII 'F'
CS
Chip Select
after last byte sent or received

Multiple SPI Devices, Option 1: Chip Select



- + Individual chip access
- - more pins

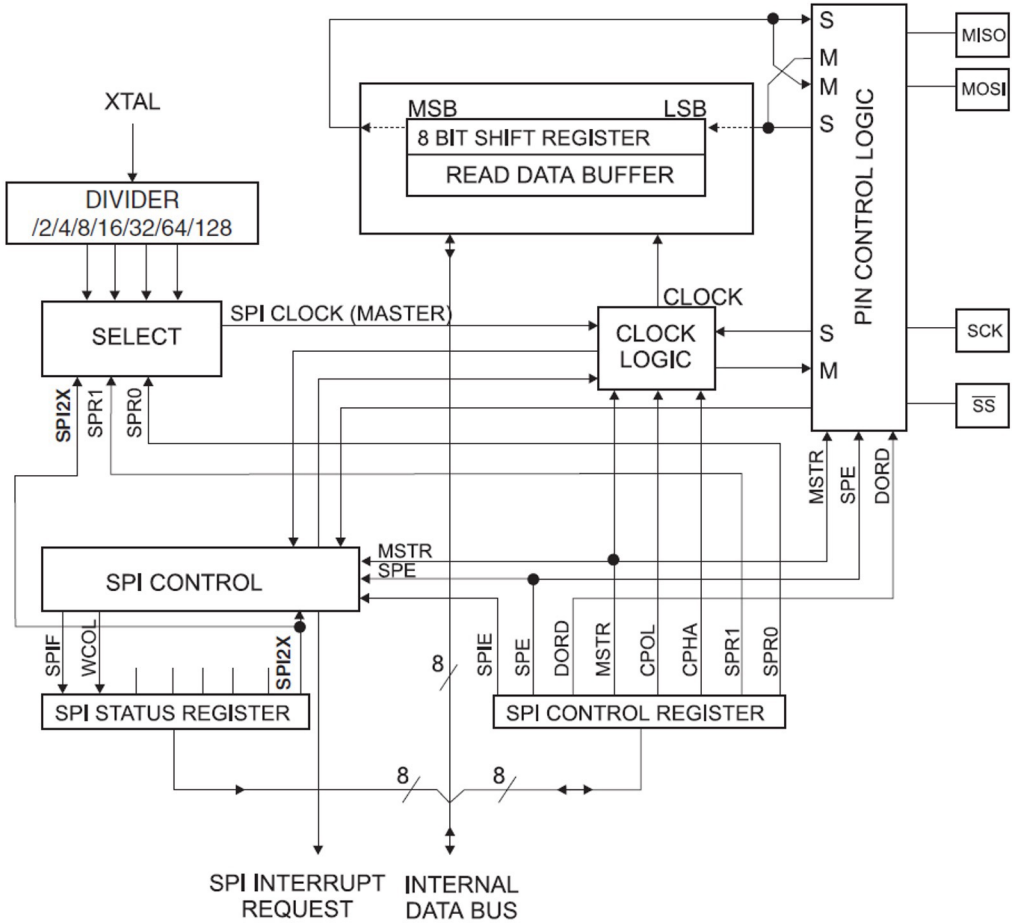
Multiple SPI Devices, Option 2: Big loop

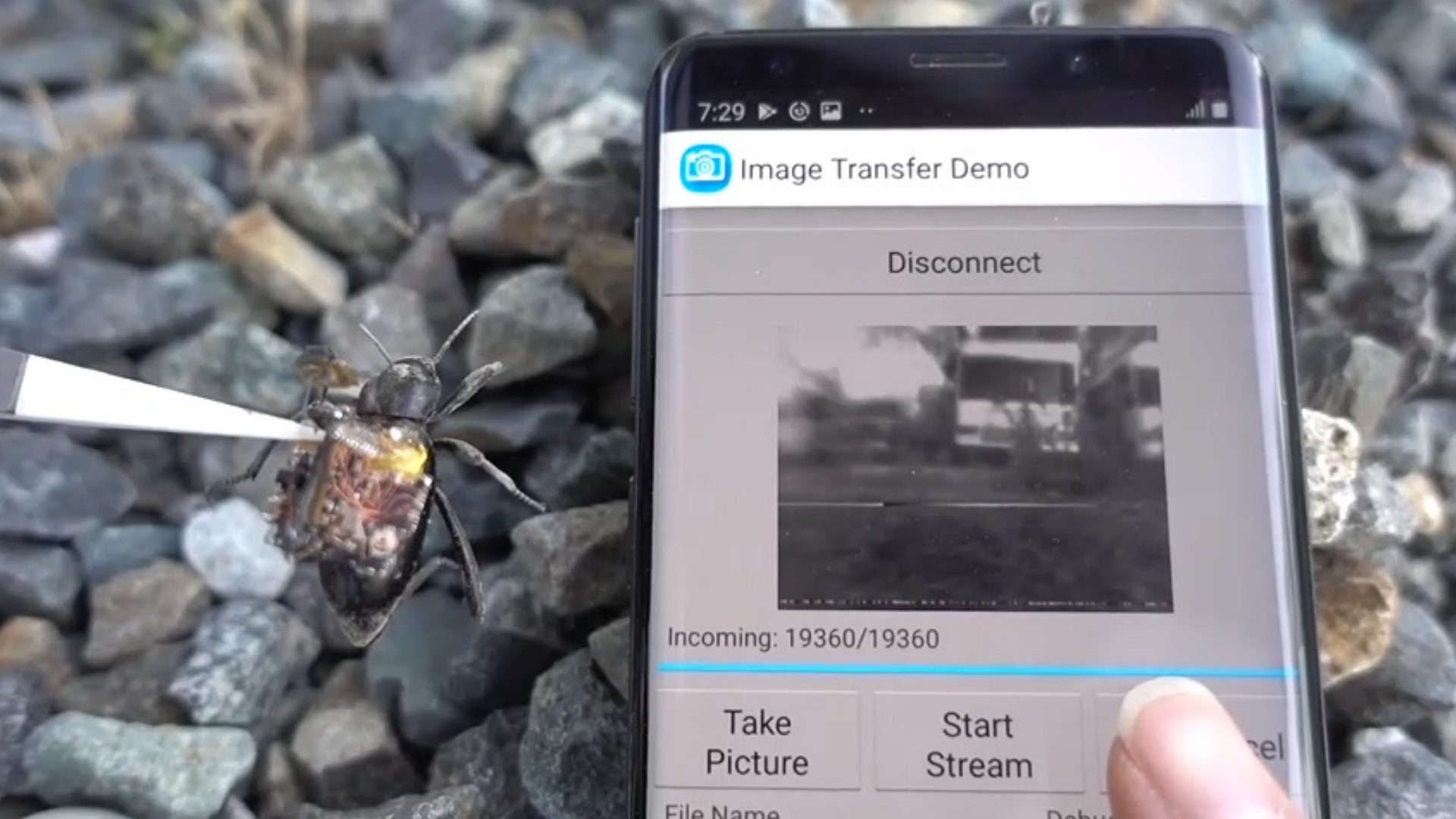


- + fewer pins
- - slower, all at one time

SPI on our AtMega 2560

Figure 21-1. SPI Block Diagram[1]




7:29
Image Transfer Demo
Disconnect
Incoming: 19360/19360
Take Picture
Start Stream

Example of SPI: Biodegradable mouse

I^2C (inter-integrated circuit bus)

- Purpose: to communicate between ICs on a board with fewer pins and traces.
- Developed by Philips in 1980's (now NXP)
- Reference: NXP Documentation
  - The name I^2C is proprietary, some chips call this "Two Wire Interface" (TWI)
- 2-wire, Serial, bi-directional
- SCL: clock wire
- SDA: data wire

I²C (inter-integrated circuit bus)

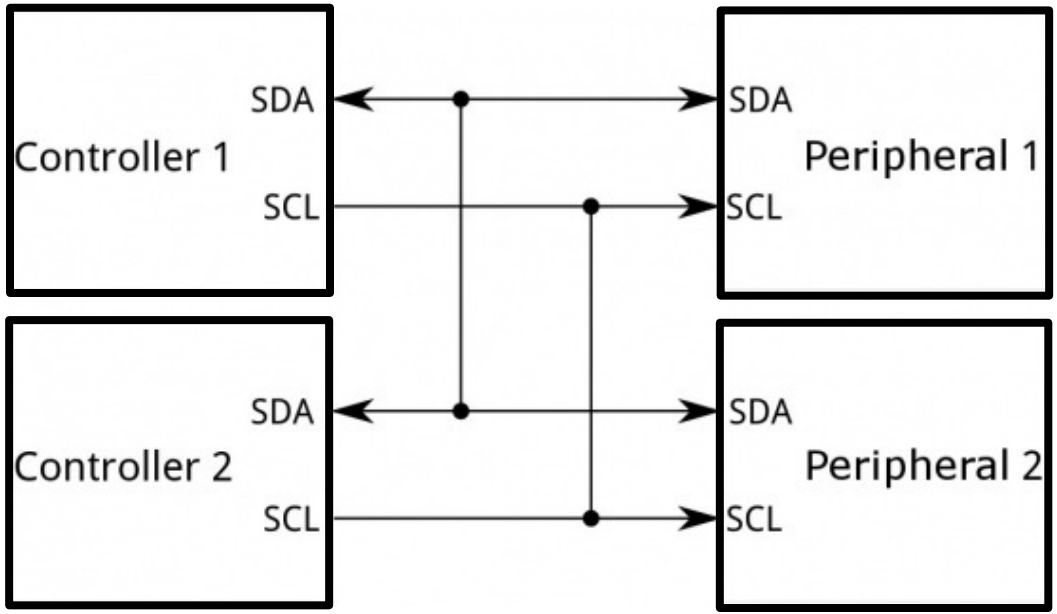
Controller: One device on the I2C bus which controls transmission and supplies the clock signal.

Peripheral: One or more other devices on I2C bus which send and/or receive data.

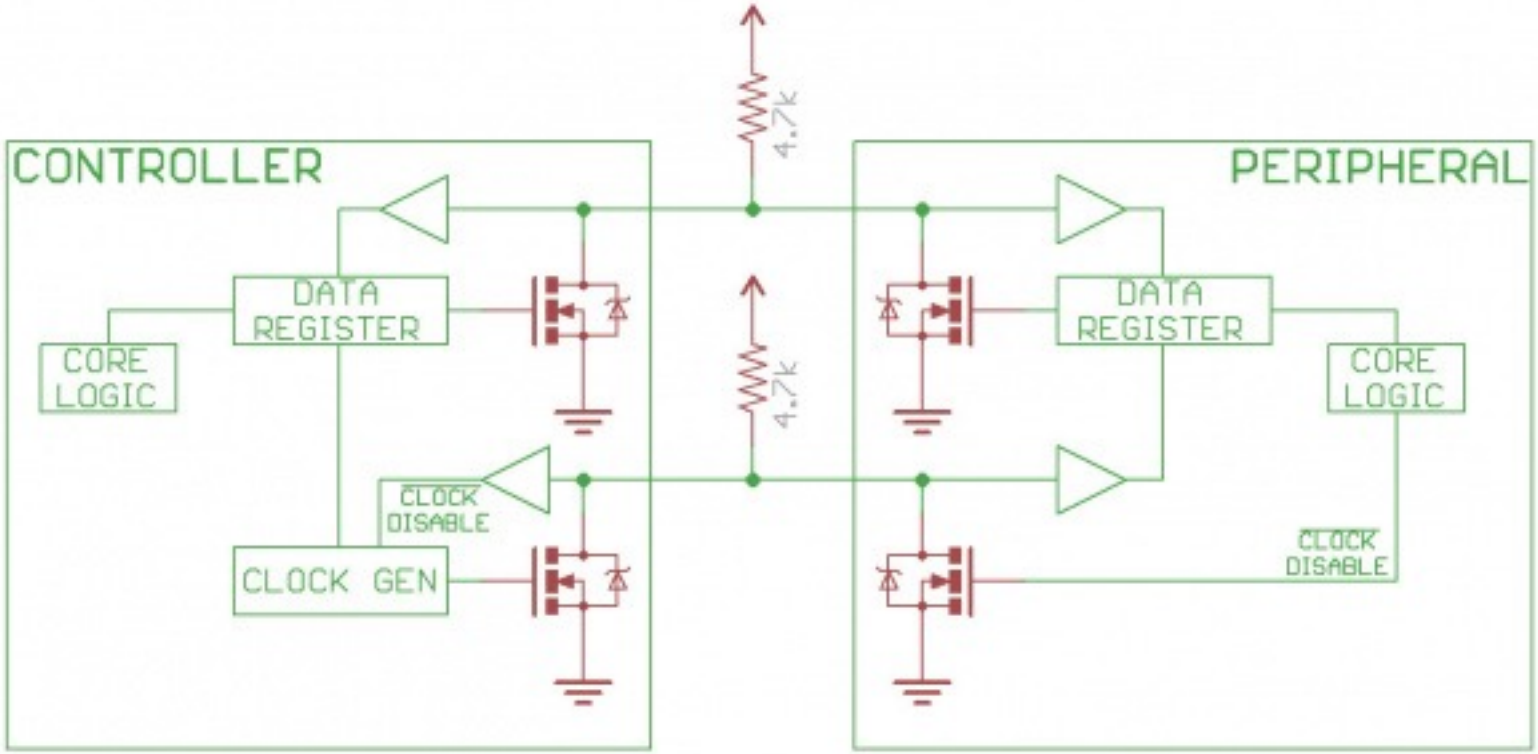
Obsolete Name	Replacement Name
Master	Controller
Slave	Peripheral

I²C (inter-integrated circuit bus)

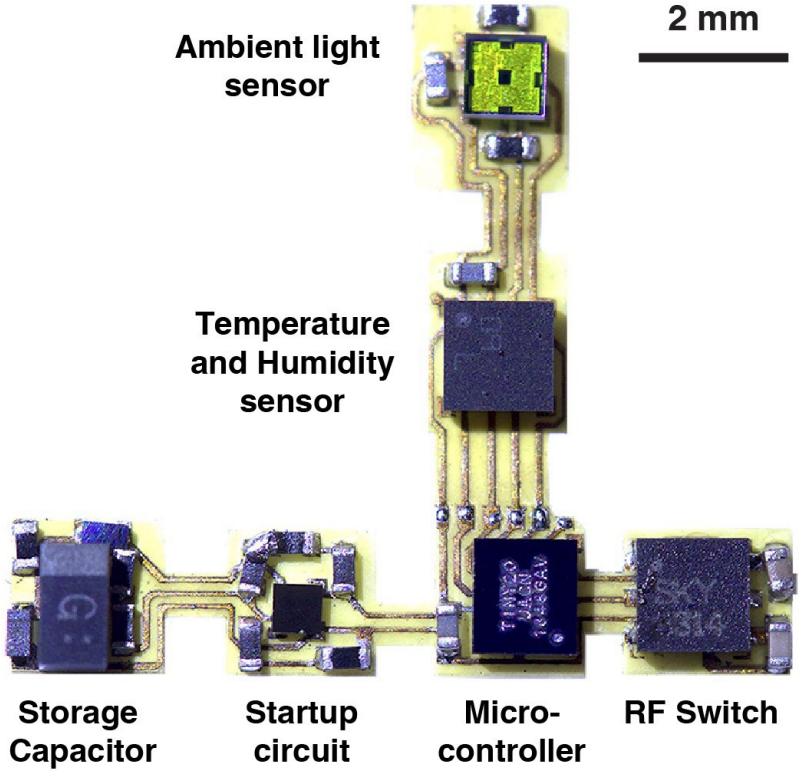
- SCL: clock line provided by Controller
- SDA: bi-directional serial data

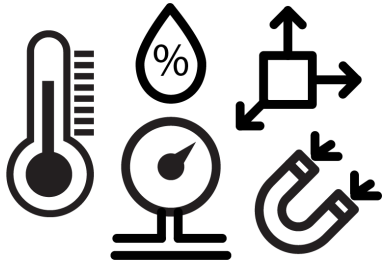


I²C (inter-integrated circuit bus)



I²C (inter-integrated circuit bus)



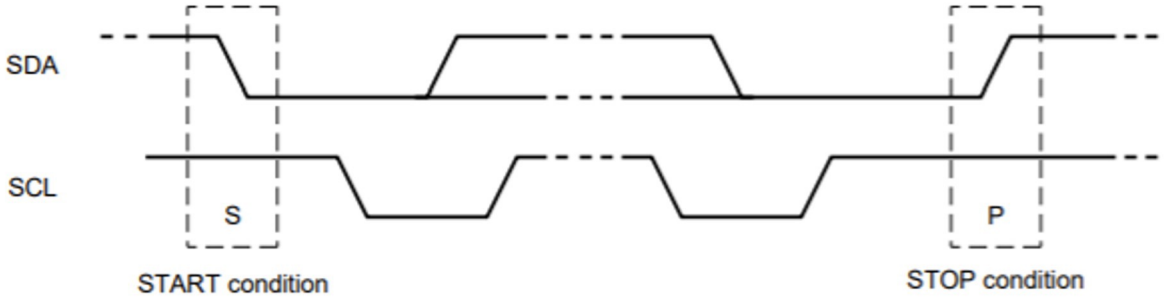


Advantage:

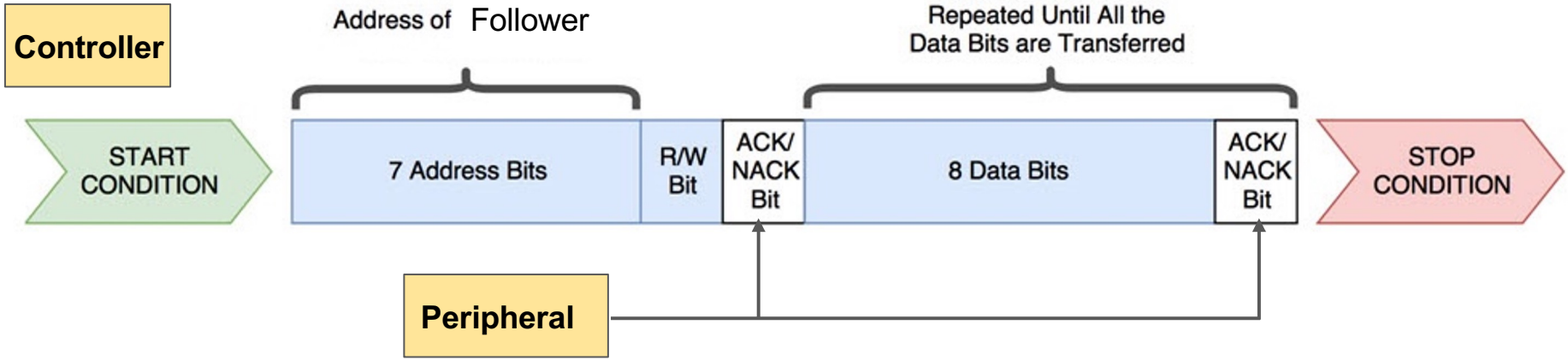
- Only requires 4 wires and can connect many devices
- Available on many small chips

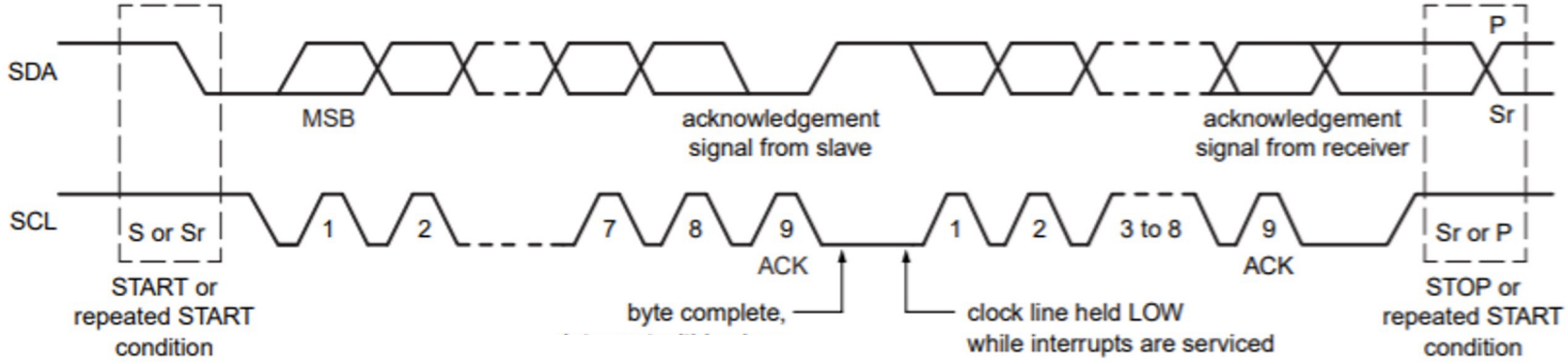
I2C "Start Condition" & "Stop Condition"

	Idle	Start Condition	Stop Condition
SDA:	HIGH	HIGH→ LOW	LOW→ HIGH
SCL:	HIGH	HIGH	HIGH

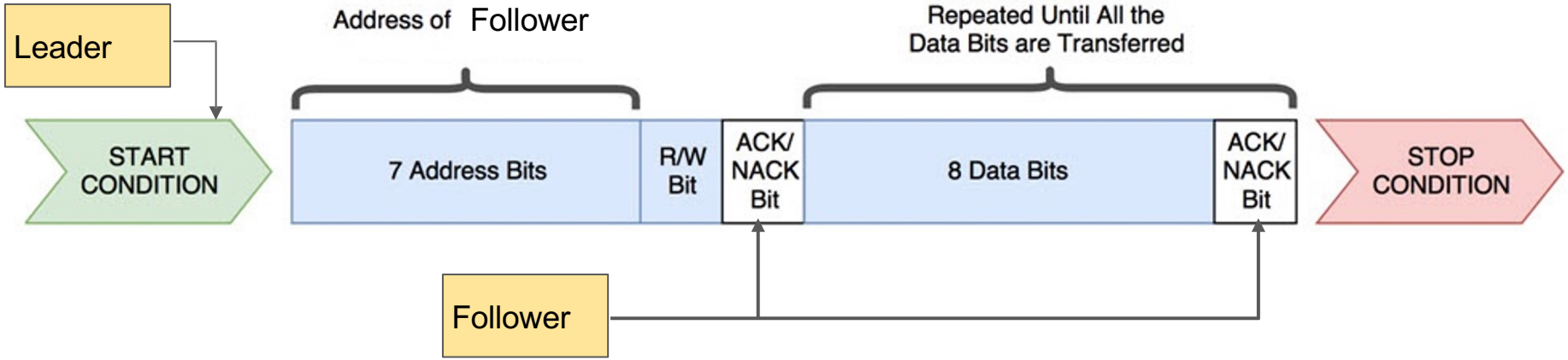


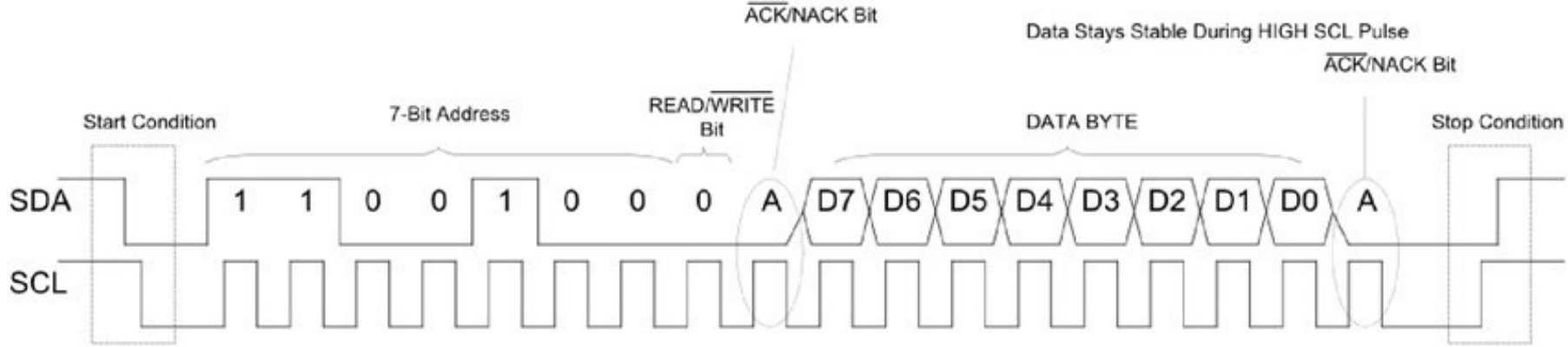
I2C Protocol





I2C Protocol





I2C Addresses

I2C address is hard coded into the chip but there are only 128 (2^7) of them!

[Adafruit's List of I2C addrs]

What if you want to use 2 chips with same I2C addr???

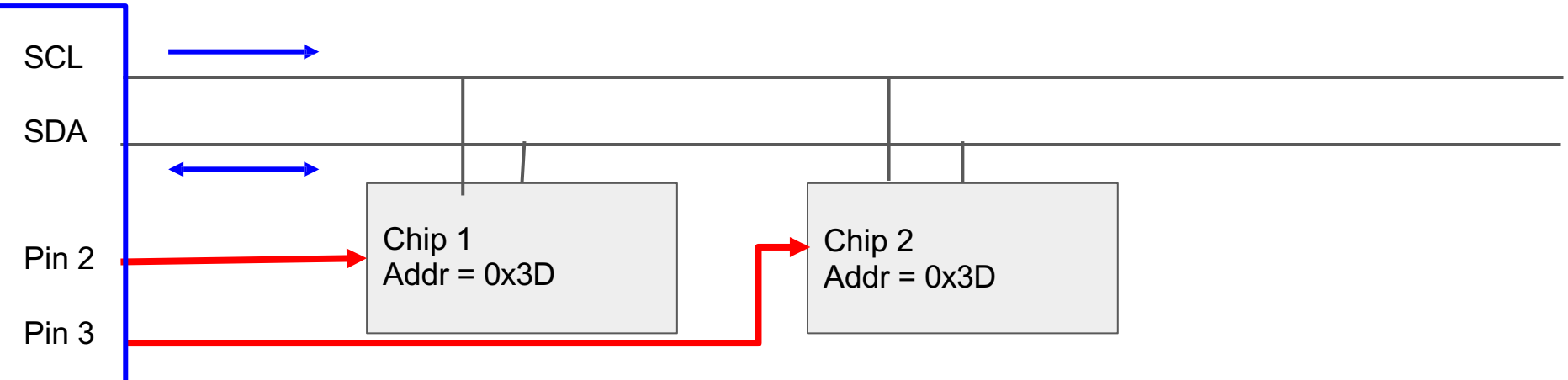
I2C Addresses

I2C address is hard coded into the chip but there are only 128 (2^7) of them!

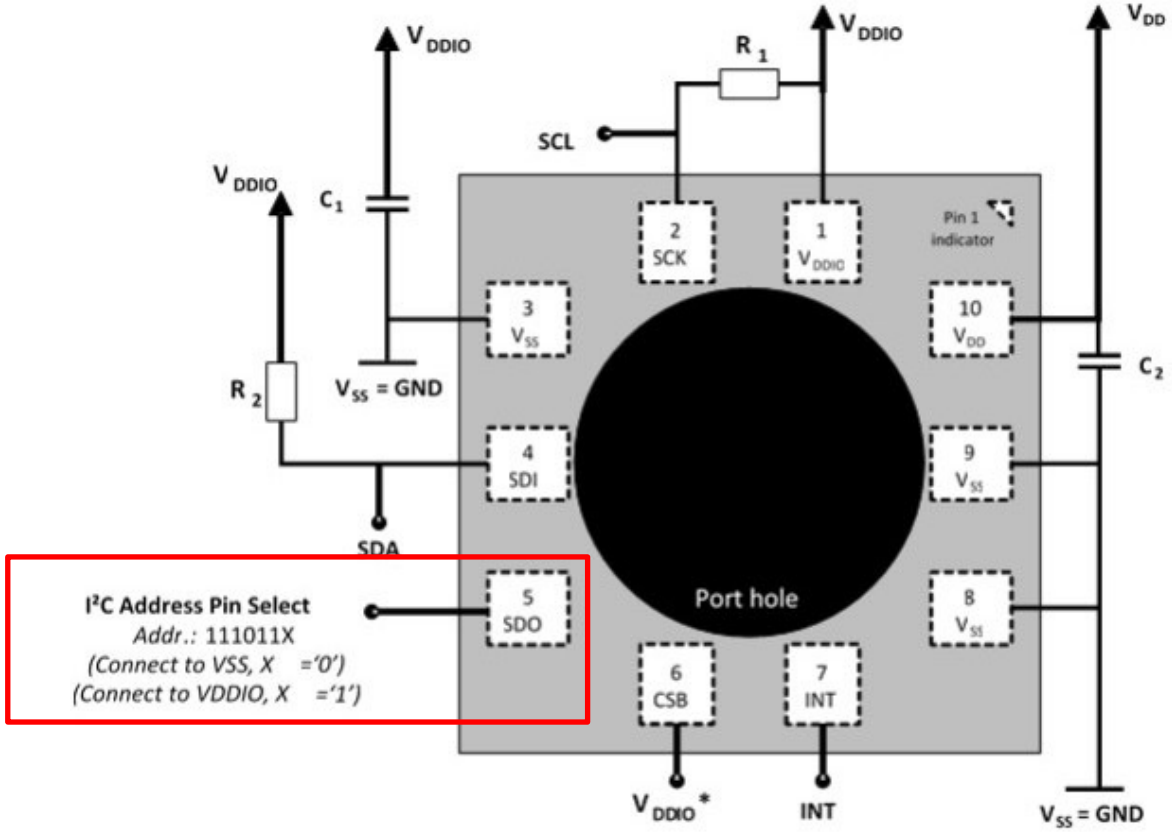
[Adafruit's List of I2C addrs]

What if you want to use 2 chips with same I2C addr???

Use chip select / enable bits (if available) (driven by random I/O pins)



I2C Protocol



Step 1: Find the registers to read/write from

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
67	103	I2C_MST_DELAY_CTRL	R/W	DELAY_ES_SHADOW	-	-	I2C_SLV4_DLY_EN	I2C_SLV3_DLY_EN	I2C_SLV2_DLY_EN	I2C_SLV1_DLY_EN	I2C_SLV0_DLY_EN
68	104	SIGNAL_PATH_RESET	R/W	-	-	-	-	-	GYRO_RESET	ACCEL_RESET	TEMP_RESET
6A	106	USER_CTRL	R/W	-	FIFO_EN	I2C_MST_EN	I2C_IF_DIS	-	FIFO_RESET	I2C_MST_RESET	SIG_COND_RESET
6B	107	PWR_MGMT_1	R/W	DEVICE_RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		
6C	108	PWR_MGMT_2	R/W	LP_WAKE_CTRL[1:0]		STBY_XA	STBY_YA	STBY_ZA	STBY_XG	STBY_YG	STBY_ZG
72	114	FIFO_COUNTH	R/W	FIFO_COUNT[15:8]							
73	115	FIFO_COUNTL	R/W	FIFO_COUNT[7:0]							
74	116	FIFO_R_W	R/W	FIFO_DATA[7:0]							
75	117	WHO_AM_I	R	-	WHO_AM_I[6:1]						-

4.28 Register 107 – Power Management 1
PWR_MGMT_1

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
6B	107	DEVICE_RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		

Parameters:

DEVICE_RESET When set to 1, this bit resets all internal registers to their default values.

The bit automatically clears to 0 once the reset is done.

The default values for each register can be found in Section 3.

SLEEP When set to 1, this bit puts the MPU-60X0 into sleep mode.

CYCLE When this bit is set to 1 and *SLEEP* is disabled, the MPU-60X0 will cycle between sleep mode and waking up to take a single sample of data from active sensors at a rate determined by *LP_WAKE_CTRL* (register 108).

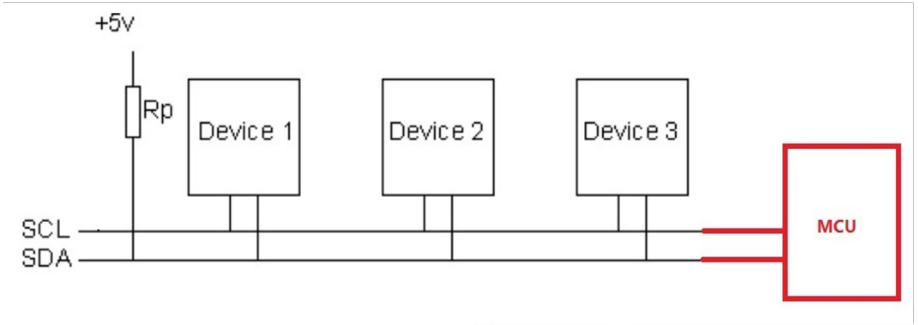
TEMP_DIS When set to 1, this bit disables the temperature sensor.

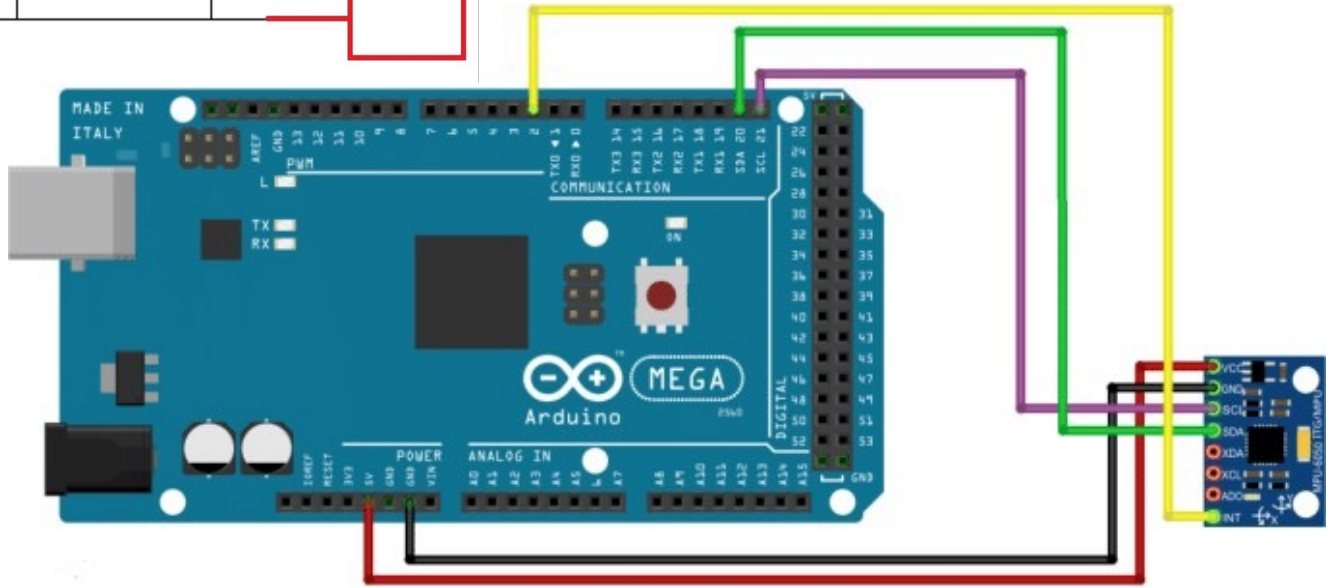
CLKSEL 3-bit unsigned value. Specifies the clock source of the device.

Step 1: Find the registers to read/write from

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
67	103	I2C_MST_DELAY_CTRL	R/W	DELAY_ES_SHADOW	-	-	I2C_SLV4_DLY_EN	I2C_SLV3_DLY_EN	I2C_SLV2_DLY_EN	I2C_SLV1_DLY_EN	I2C_SLV0_DLY_EN
68	104	SIGNAL_PATH_RESET	R/W	-	-	-	-	-	GYRO_RESET	ACCEL_RESET	TEMP_RESET
6A	106	USER_CTRL	R/W	-	FIFO_EN	I2C_MST_EN	I2C_IF_DIS	-	FIFO_RESET	I2C_MST_RESET	SIG_COND_RESET
6B	107	PWR_MGMT_1	R/W	DEVICE_RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		
6C	108	PWR_MGMT_2	R/W	LP_WAKE_CTRL[1:0]		STBY_XA	STBY_YA	STBY_ZA	STBY_XG	STBY_YG	STBY_ZG
72	114	FIFO_COUNTH	R/W	FIFO_COUNT[15:8]							
73	115	FIFO_COUNTL	R/W	FIFO_COUNT[7:0]							
74	116	FIFO_R_W	R/W	FIFO_DATA[7:0]							
75	117	WHO_AM_I	R	-	WHO_AM_I[6:1]						-

Step 2: Hardware connections





Step 3: Writing the code

```
#include "Wire.h" // I2C library.

const int MPU_ADDR = 0x68; // I2C
address
int16_t accel_x, accel_y, accel_z;
int16_t gyro_x, gyro_y, gyro_z;   // variables for
gyro raw data
int16_t temperature;                    //
variables for temperature data

uint16_t temp_H
uint16_t temp_L:;

void setup() {
  Serial.begin(9600);
  Wire.begin();
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission(true);
}
```

init I2C

I2C address

write 0 to reg 6B

send a byte of data

```
void loop() {
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(0x41);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_ADDR, 2, true);

  // Read temperature data
  temp_high = Wire.read();
  temp_low = Wire.read();
  // Convert temperatures to single value
  temperature = temp_high << 8 | temp_low;
  Serial.print("Temp H: "); Serial.println(temp_H, BIN);
  Serial.print("Temp L: "); Serial.println(temp_L, BIN);
  Serial.print("Temp all: ");
  Serial.println(temperature, BIN);

  Serial.print("Temp = ");
  Serial.println(temperature/340.00+36.53);

  // delay
  delay(1000);
}
```

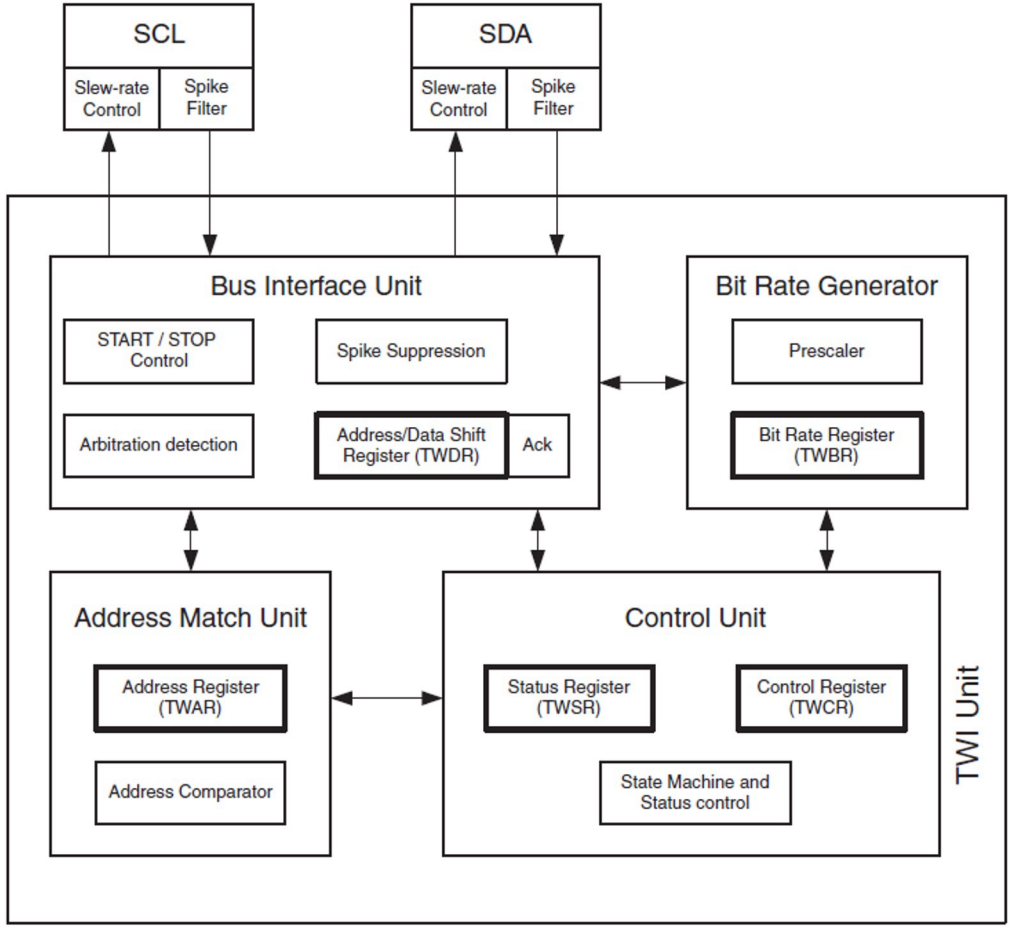
where to read data from

2 bytes

I²C on our AtMega 2560

"TWI" = 2-wire interface

Figure 24-9. Overview of the TWI Module



AtMega2560 Software Examples

SPI: https://github.com/KareemAhmed96/SPI-Configurable-Driver

I²C/TWI: Arduino AVR library: "Wire.h"

AtMega2560 Software Examples

SPI & I²C/TWI: “Bit Banging” code

- Assign your own pins to COPI, PICO, SCLK, CS
- Write code to generate the needed transitions.
- + portable, easy to understand(?), works on small chips
- - slow, blocks other code
- Example github project