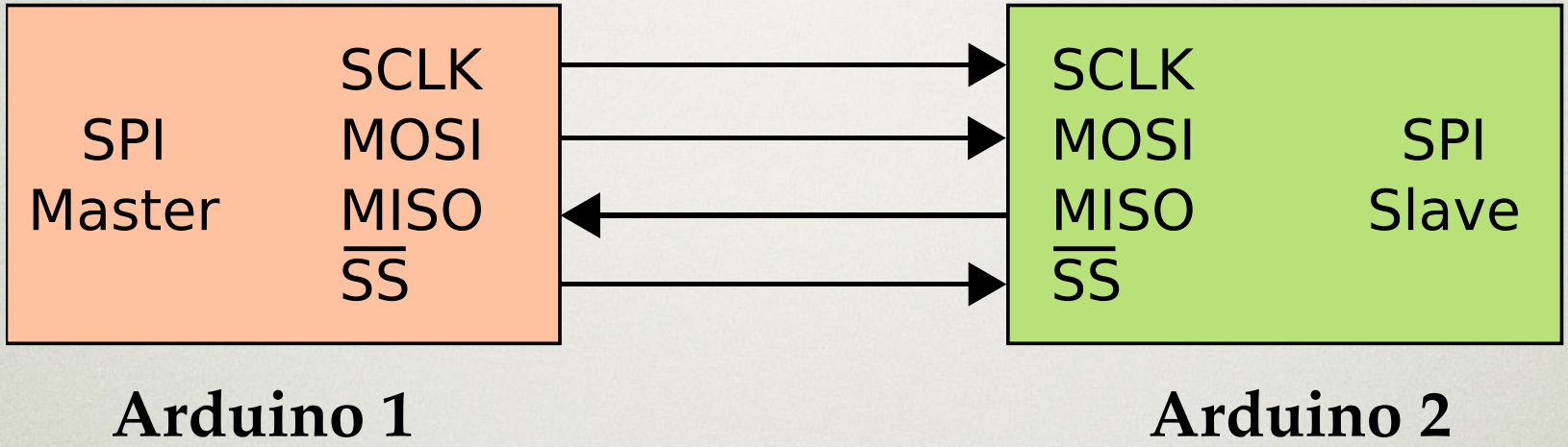
Hands-On Arduino 5B

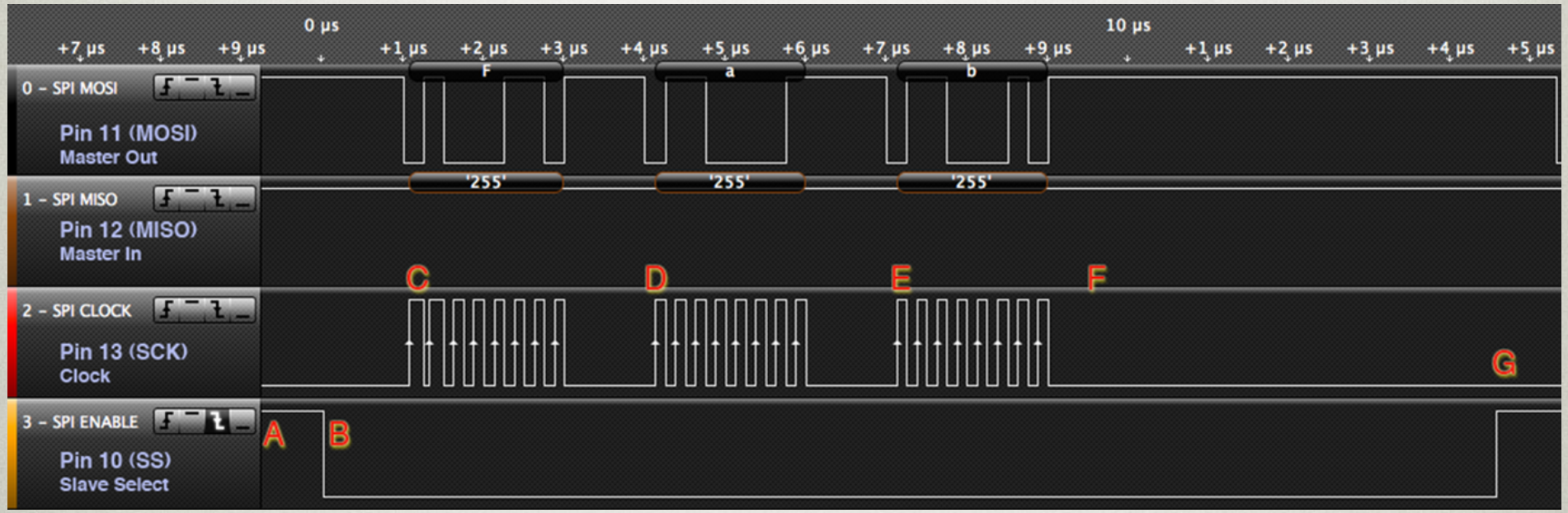
SPI

Arduino to Arduino

The SPI Connection



The SPI Signals



A: Prior to transmission

B: Enable the slave

C: Character "F" (0b01000110) transmitted.

D: The letter "a" transmitted.

E: The letter "b" transmitted.

F: No more data.

G: Slave released.

SPI Timing: Default

- Each byte takes 3μsec (all 3 take 9μsec)
- Each clock cycle takes 0.25μsec (4MHz)
- It is theoretically possible to transmit bytes in 1.125μsec, but many devices do not support this.

SPI: Slave Select

- Each slave device has its own $\overline{\texttt{SS}}$ pin.
- When the $\overline{\texttt{SS}}$ is high, each slave should have its other connections (`MOSI`, `MISO`, `SCK`) in a Hi-Z state.
- A slave's `MISO` line should be configured to be an output if and only if the $\overline{\texttt{SS}}$ line is taken low.

SPI Communication Protocols

- SPI doesn't specify a protocol
- Master/Slave must agree on what the data means.
- Data may be sent+received simultaneously
- In general, a Master might not even know if the Slave is connected.

The Arduino as SPI Slave

- The Arduino wants to be the Master.
- As a slave, the transmission is interrupt-driven
  - when xfer of a byte is complete, the **`ISR (SPI_STC_vect)`** is called. The byte is in **`SPDR`**.
- The incoming data should be collected in a buffer
- A flag is set when a "significant byte" (e.g., newline) is received.

Hands-On Arduino 5B

SPI

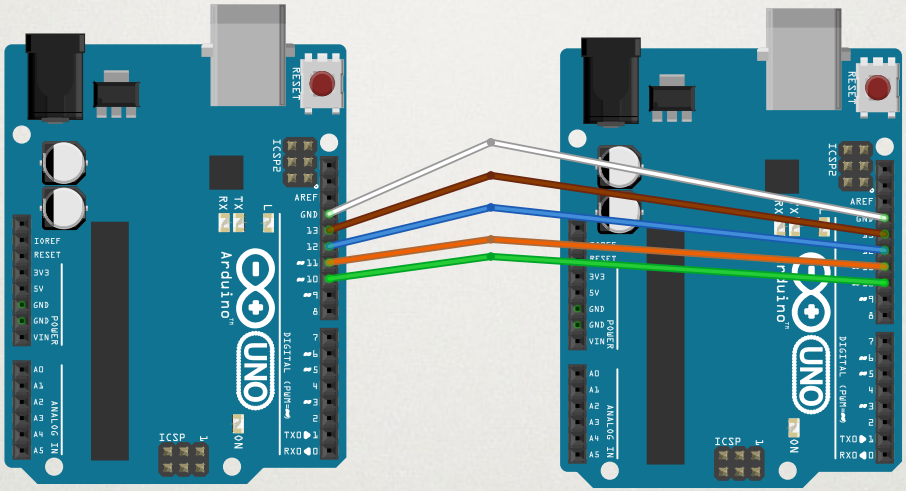
Example 1

Hello, world!*

*Credits to: Nick Gammon

http://www.gammon.com.au/forum/?id=10892

Example: Master to Slave



- Alternate “Hello, world!” and “ByeBye, world!”
- Use SPI

Master Code

```
#include <SPI.h>
int swi=0;
void setup (void)
{
 digitalWrite(SS, HIGH);  // set SS high
  // initialize the SPI for Master Mode
  SPI.begin ();
 // Slow down the Master a bit
  SPI.setClockDivider(SPI_CLOCK_DIV8);
}
void loop (void)
{
 char c;
 // enable Slave Select
  digitalWrite(SS, LOW);
```

Master Code (cont'd)

```
// send alternate strings
  if(swi==0)
  {
    for (const char *p= "Hello, world!\n" ; c = *p; p++)
    SPI.transfer (c);
  }
  else
  {
    for (const char * p = "ByeBye, world!\n" ; c = *p; p++)
    SPI.transfer (c);
  }
 // disable Slave Select
  digitalWrite(SS, HIGH);
  swi = 1-swi;

  delay (1000);  // 1 second delay
}
```

Slave Code

```
#include <SPI.h>
char buf [100];
volatile byte pos;
volatile boolean process_it;
void setup (void)
{
  Serial.begin (115200);   // debugging
 // Set MISO pin as output
  pinMode(MISO, OUTPUT);
  // turn on SPI in slave mode
  SPCR |= _BV(SPE);
  // get ready for an interrupt
  pos = 0;   // buffer empty
  process_it = false;
 // now turn on interrupts
  SPI.attachInterrupt();
}
```

Slave Code (cont'd)

```
// SPI interrupt service routine
ISR (SPI_STC_vect)
{
byte c = SPDR;  // grab byte from SPI Data Register

  // add to buffer if room
  if (pos < sizeof buf)
    {
    buf [pos++] = c;

    // example: newline means time to process buffer
    if (c == '\n')
      process_it = true;
    }  // end of available buffer
}  // end of interrupt routine SPI_STC_vect
```

Slave Code (cont'd)

```
void loop (void)
{
  //only do something if a complete transmission occurs.
  if (process_it)
    {
    buf [pos] = 0;
    Serial.println (buf);
    pos = 0;
    process_it = false;
    }

}
```

Hands-On Arduino 5B

SPI

Example 2

Simultaneous Bidirectional Communication

Bidirectional Communication

- The SPI is designed to be full-duplex
- When the slave puts something onto the **SPDR**, it is ready to transmit.
- The transmission to the Master will not occur until the $\overline{\text{SS}}$ is low and the **SCLK** starts pulsing.
- The SPDR is "double buffered." It has a receive buffer as well as a transmit buffer.

Bidirectional Communication

- Some synchronization may be required for the communication.
  - e.g., the two Arduinos probably will not start their programs at the same time.
- If synchronization is required, the Master may send some code byte that indicates time to start.
  - the slave would wait to receive that byte before putting any data into **SPDR**.

Master Code

```
#include <SPI.h>
void setup (void)
{
  Serial.begin (115200);
  digitalWrite(SS, HIGH);// initialize SS
  // initialize the SPI for Master Mode
  SPI.begin();
  // Slow the clock rate by 8
  SPI.setClockDivider(SPI_CLOCK_DIV8);
}
```

```
void loop(void)
{
  byte Mvalsent,Mvalreceived;

  digitalWrite(SS, LOW);
  for(int jj = 0; jj<255; jj++)
  {
    Mvalsent = jj;
    Mvalreceived=SPI.transfer(Mvalsent);
    Serial.print("sent: ");
    Serial.print(Mvalsent);
    Serial.print("\t received: ");
    Serial.println(Mvalreceived);
    delay(200);
  }
  digitalWrite(SS, HIGH);
  delay(1000);
}
```

Slave Code

```
#include <SPI.h>
volatile boolean process_it;
volatile byte Svalreceived,Svalsent;
int jj=0;
void setup (void)
{
  // Set MISO pin as output
  pinMode(MISO, OUTPUT);
  // turn on SPI in slave mode
  SPCR |= _BV(SPE);
  // get ready for an interrupt
 process_it = false;
 // now turn on interrupts
  SPI.attachInterrupt();
}
```

```
ISR (SPI_STC_vect)
{
  Svalreceived = SPDR;
  process_it = true;
}
void loop(void)
{
  if(process_it)
  {
    Svalsent=jj;
    SPDR = Svalsent;
    jj++;
    if(jj>254) jj=0;
    process_it = false;
  }
}
```

Hands-On Arduino 5B

SPI

More to come...