Embedded System Design Techniques™

Rapid Prototyping Embedded Systems using MicroPython

Session 2: Libraries and Peripheral Control

May 3rd, 2016

Jacob Beningo, CSDP

DesignNews

CEC CONTINUING EDUCATION CENTER

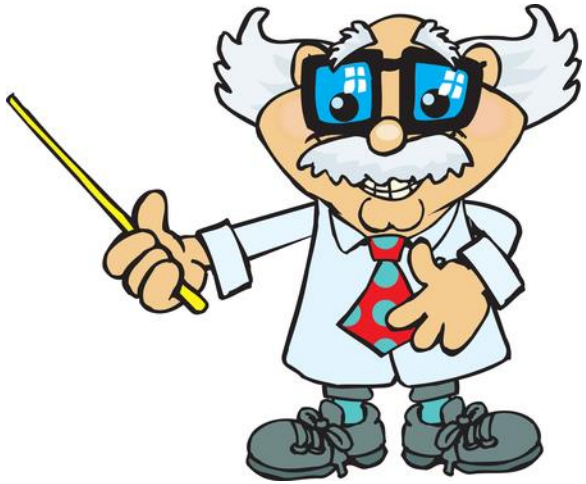
Presented by: Digi-Key ELECTRONICS

Course Overview

- Introduction to MicroPython
- **Libraries and Peripheral Control**
- Rapid Prototyping
- Building and Customizing Micro Python
- Python Scripting for Testing and Debug

Session Overview

- MicroPython Library Overview
- GPIO
- Timers
- SPI
- UART



MicroPython Library Overview

http://docs.micropython.org/en/latest/pyboard/library/index.html

Classes	Description
Accel	Accelerometer Control
ADC	analog to digital conversion
LED	LED object
SPI	master spi serial protocol
Switch	switch object
Timer	control internal timers
UART	serial bus communication

MicroPython Library Overview

http://docs.micropython.org/en/latest/pyboard/library/index.html

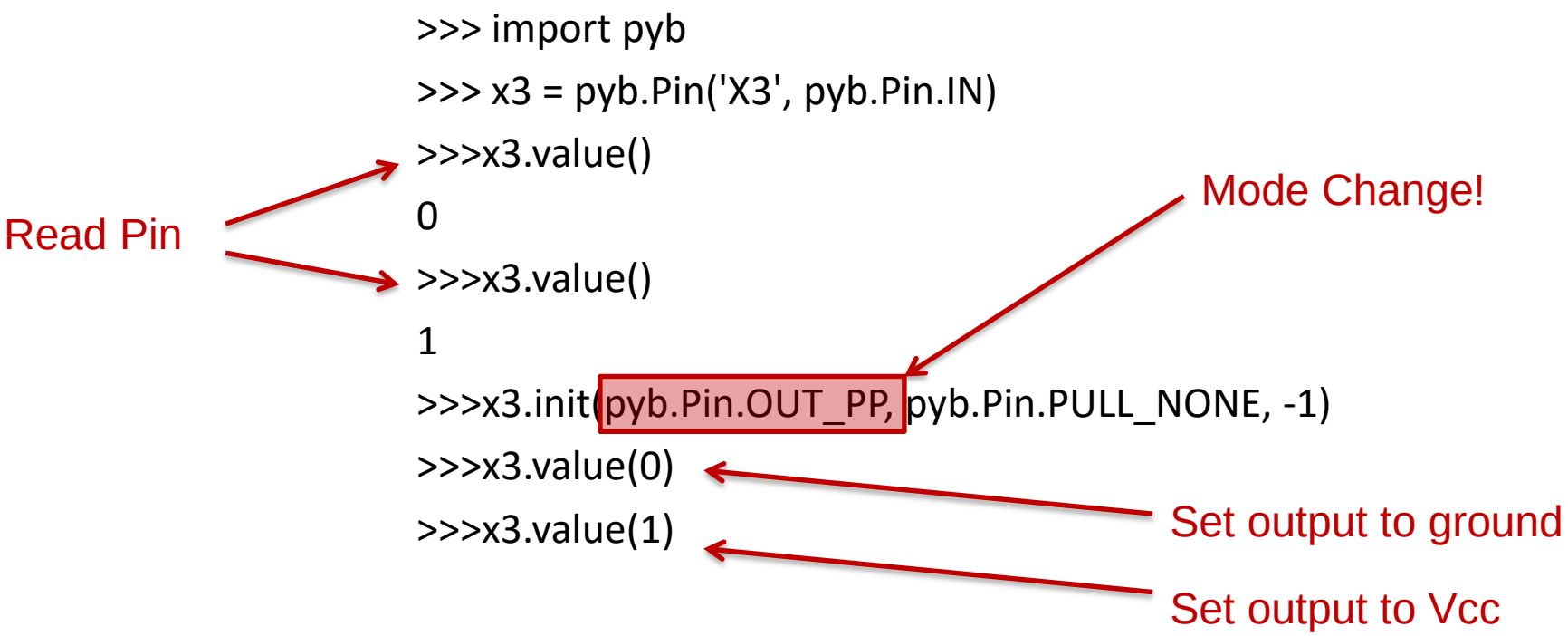
Classes	Description
gc	Control the garbage collector
os	“operating system“ services
uhashlib	Hashing algorithm
ujson	JSON encoding and decoding
usocket	Socket module
network	Network configuration
zlib	Decompression of gzip archiver files (no compression)

MicroPython - GPIO

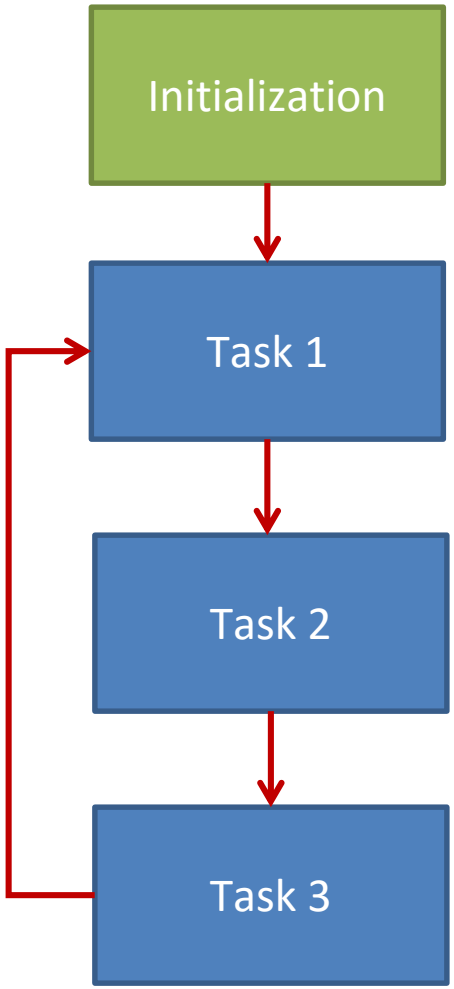
http://docs.micropython.org/en/latest/library/pyb.Pin.html

Methods	Description
pin.init(mode, pull, function)	Initializes the gpio pin
pin.high()	Sets the output pin level high
pin.low()	Sets the output pin level low
pin.value()	Gets the state of an input pin
pin.name()	Gets the pin name
pin.mode()	Returns current pin mode setting
pin.af()	Returns pin alternate function value

MicroPython - GPIO



Round Robin Scheduling



Main.py

```
import pyb

System_Init()

while True:

    // Run the first task
    Task1();

    // Run the first task
    Task2();

    // Run the first task
    Task3();

    pyb.delay(10)
```

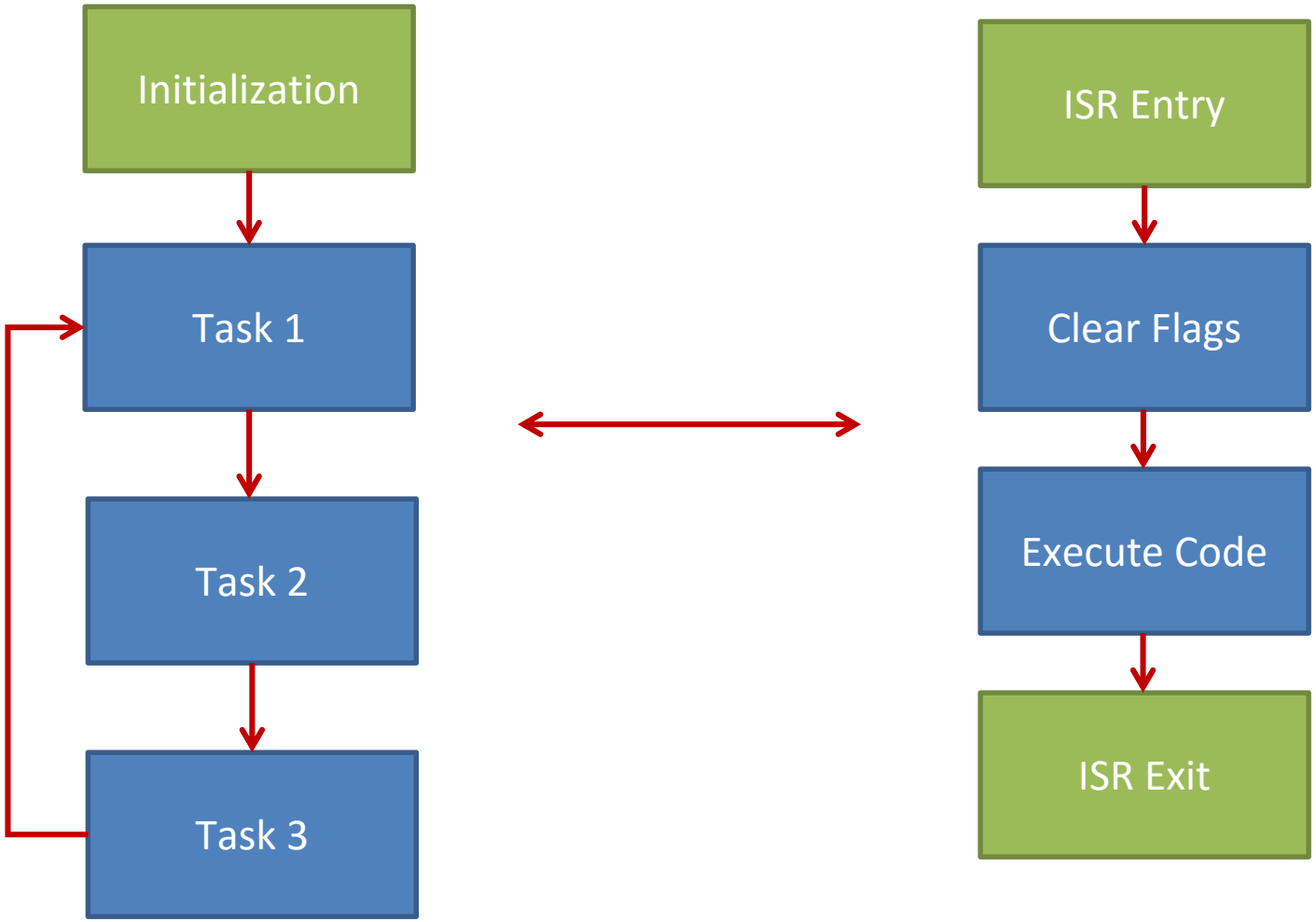
MicroPython - Timers

- Using timers to schedule a task

```
def Led_RedToggle(timer):
    pyb.LED(1).toggle()

    return

def Led_GreenToggle(timer):
    pyb.LED(1).toggle()

    return

def X1_Toggle(timer):
    global X1_State
    #toggle LED code

    return
```

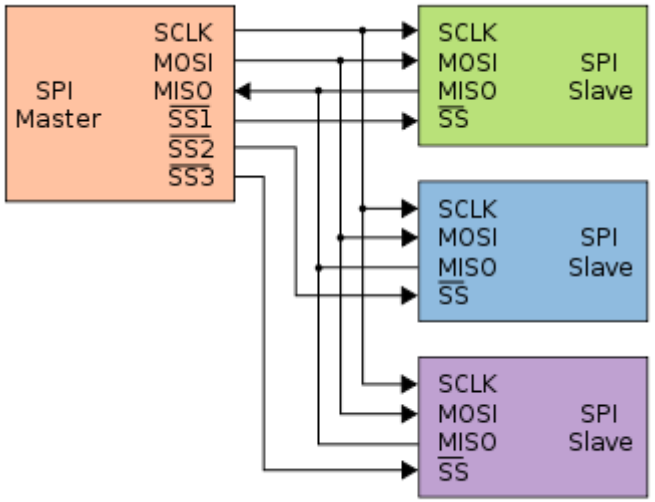
```
# Create a timer objects
TimerRedLed = pyb.Timer(1)
TimerGreenLed = pyb.Timer(2)
TimerX1 = pyb.Timer(4)

# Initialize timers TimerRedLed.init(freq=3)
TimerGreenLed.init(freq=5)
TimerX1.init(freq=10)

# Assign a function
TimerRedLed.callback(Led_RedToggle)
TimerGreenLed.callback(Led_GreenToggle)
TimerX1.callback(X1_Toggle)
```

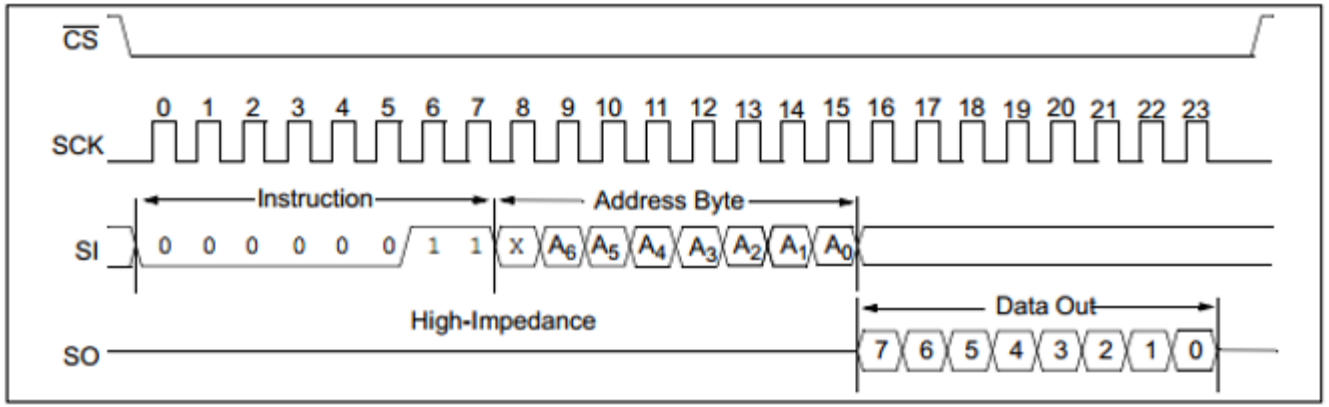
MicroPython Timers



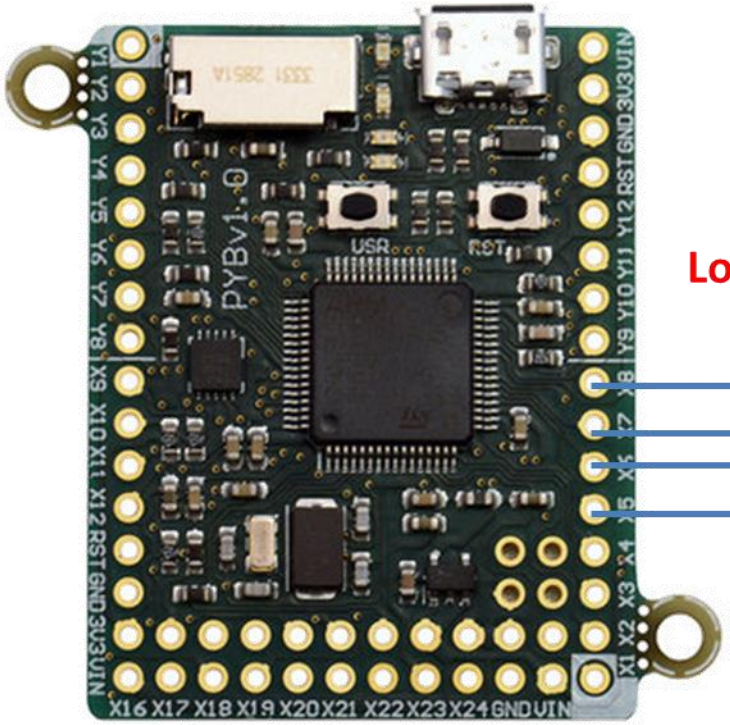
MicroPython - SPI

- Serial Peripheral Interface
  - Full Duplex
  - 3 Spi Master Outputs
  - 1 Spi Slave Input
  - 12 Mbps capable





MicroPython - SPI



Loopback MOSI to MISO

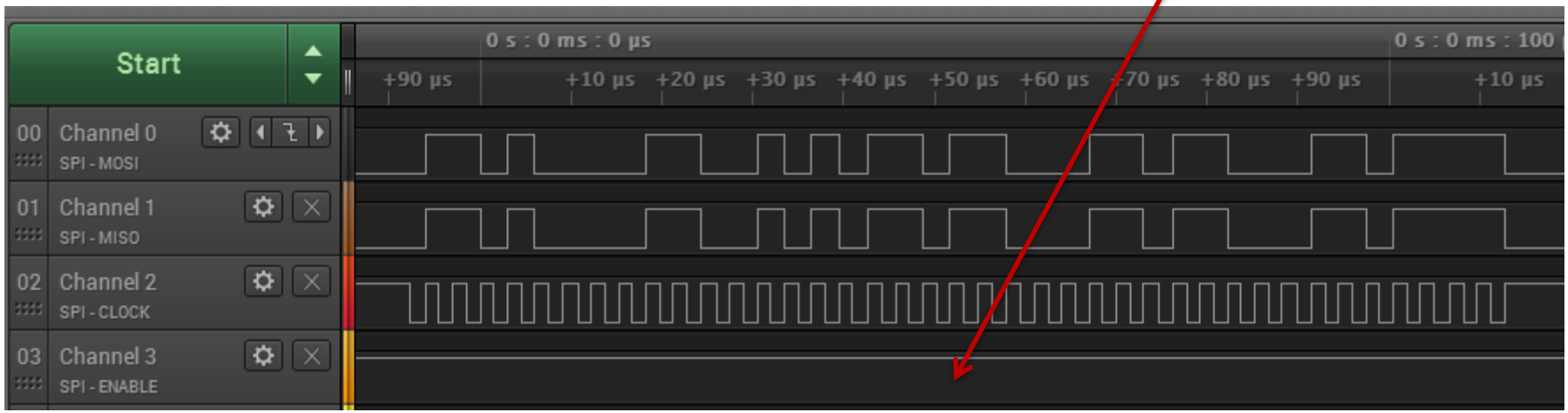
X8 - MOSI
X7 - MISO
X6 - CLK
X5 - SS

MicroPython - SPI

REPL Example:

```
# Create a SPI object using SPI1
>>> Spi1 = SPI(1, SPI.MASTER, baudrate=500000, polarity=1, phase=0)

# Setup bytearray buffer
>>> SpiBuffer = bytearray(2)

# Setup bytearray buffer
>>> Spi1.send_recv(b'14', buf)
```

SS not linked to command!



MicroPython - SPI

```
# tasks.py

# Configure the SPI device
Spi1 = SP (1, SPI.MASTER, baudrate = 500000, polarity=1, phase=0)

# Configure X5 as the Slave Select Output
SS1 = pyb.Pin.board.X5
SS1.init(pyb.Pin.OUT_PP, pyb.Pin.PULL_NONE, -1)

def SpiTransfer():

    # Set the SS low and transmit then raise SS
    SS1.value(0)
    Spi1.send(b'Hello Device!')
    SS1.value(1)
```

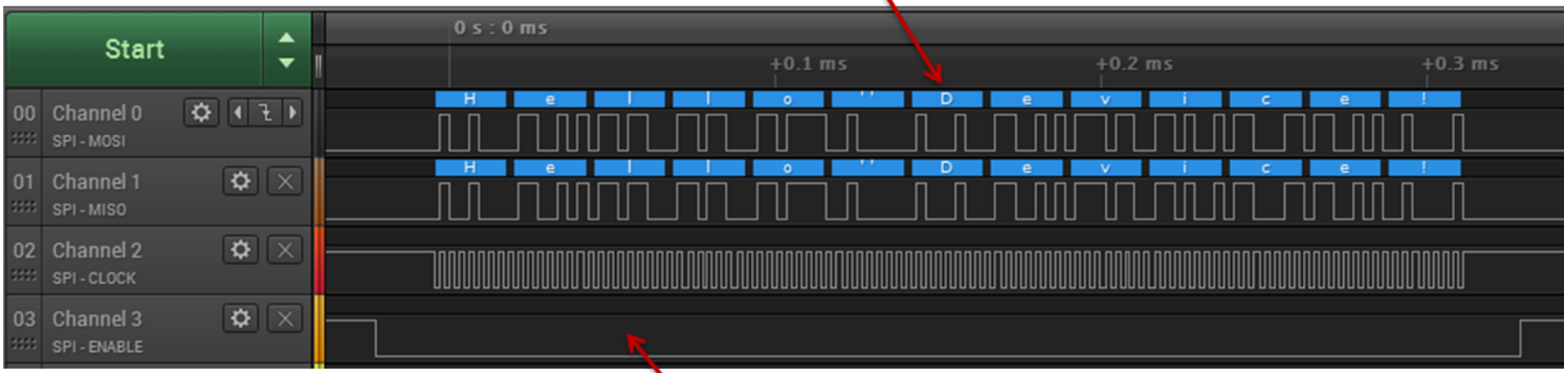
Manual SS control!

MicroPython - SPI

```
# Send and Receive from the same buffer

def SpiTransfer():
    SpiBuffer = array.array('b', range(13))

    # Set the SS low and transmit then raise SS
    SS1.value(0)
    Spi1.send_recv(b'Hello Device!', SpiBuffer)
    SS1.value(1)
```

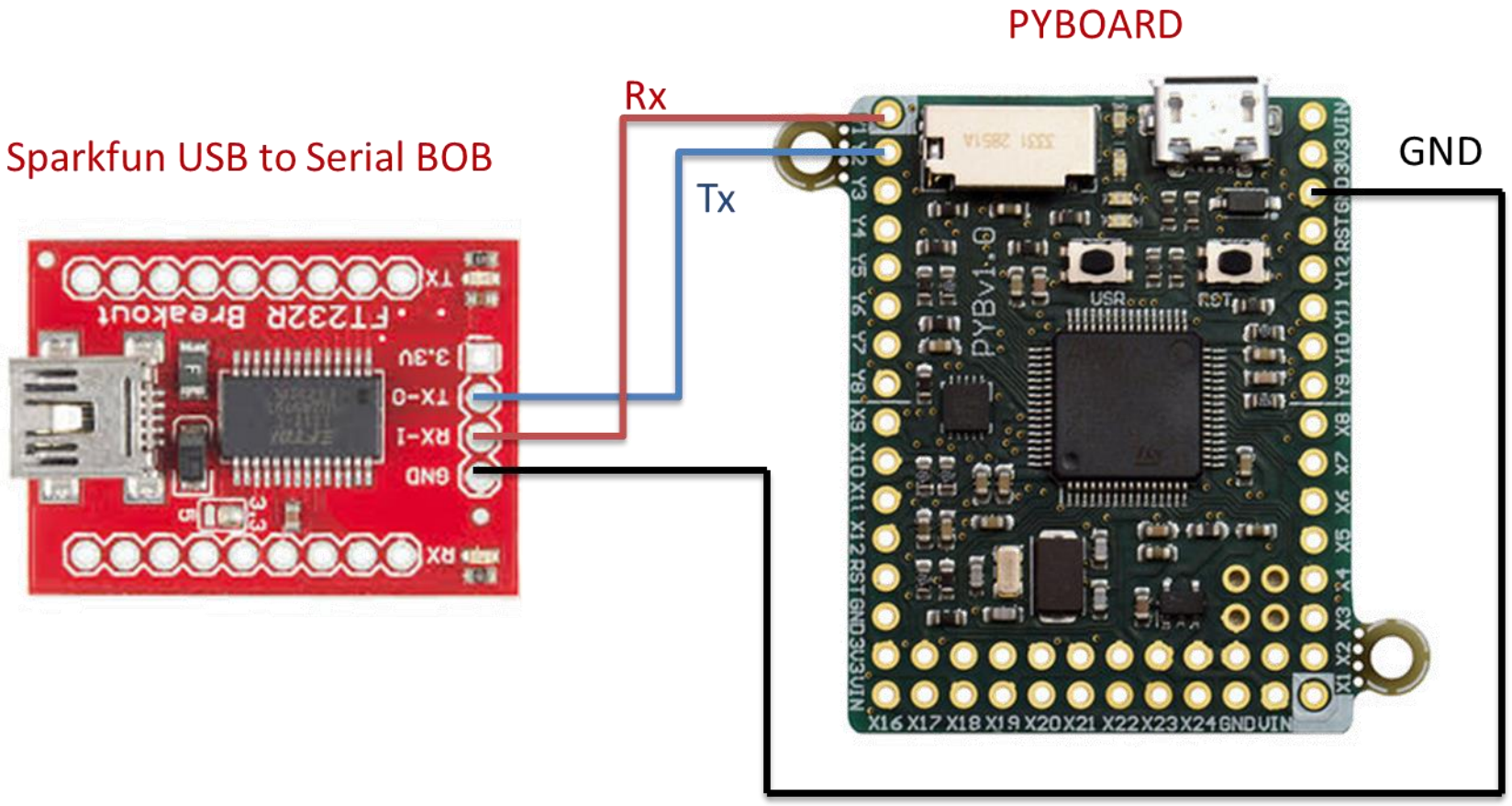
Same Size

Data loopback



Manual Control of SS

MicroPython - UART



MicroPython - UART

REPL Example:

UART Number

Baud Rate

```
# Create a uart object
>>> Uart6 = pyb.UART(6,115200)

# Initialize the uart
>>> Uart6.init(115200, bits=8, parity=None, stop=1)

# Transmit a character ASCII 2
>>> Uart6.writechar(0x32)

# Check for Rx characters
>>> Uart6.any()
True

# Read a character
>>> Uart6.readchar()
55
```

True or False

Decimal character #

MicroPython - UART

```
# tasks.py
# Configure Uart6 for communication
Uart6 = pyb.UART(6,115200)
Uart6.init(115200, bits=8, parity=None, stop=1)

# define a receive task
def UartRx():

    # Have any characters been received?
    if Uart6.any():
        # Yes read the character
        temp = Uart6.readchar()
        print (chr(temp))

        if temp == ord('q'):
            return 1
        else:
            Uart6.writechar(temp)
```

Convert to char from int

Convert to int from char

Quit application

Additional Resources

- Download Course Material for
  - Updated C Doxygen Templates (Sept 2015)
  - Example source code
  - Templates
- Microcontroller API Standard
- EDN Embedded Basics Articles
- Embedded Bytes Newsletter
  - http://bit.ly/1BAHYXm

From www.beningo.com under

- Blog > CEC Rapid Prototyping with MicroPython

The Lecturer – Jacob Beningo

Jacob Beningo
Principal Consultant

Social Media / Contact

- E : jacob@beningo.com
- T : 248-719-6850
- : Jacob_Beningo
- : Beningo Engineering
- : JacobBeningo
- EDN : Embedded Basics

CONSULTING

- Secure Bootloaders
- Code Reviews
- Architecture Design
- Real-time Software
- Expert Firmware Analysis

EMBEDDED TRAINING

DOULOS CERTIFIED TRAINING PARTNER

BENINGO EMBEDDED GROUP

www.beningo.com

Presented by: CEC CONTINUING EDUCATION CENTER Digi-Key ELECTRONICS