# Designing Embedded Systems using Micro Python

## Class 4: Developing Real-time Application Projects

June 13, 2019

Jacob Beningo

DesignNews

Presented by:

CEC CONTINUING EDUCATION CENTER

Digi-Key ELECTRONICS

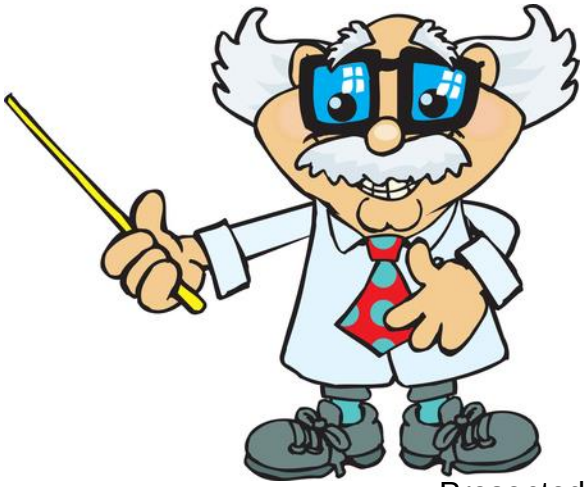
# Course Overview

**Topics:**

- Designing Products with MicroPython
- Getting Started with the Pyboard D-Series
- Customizing the MicroPython Kernel for Production
- **Developing Real-time Application Projects**
- Testing MicroPython Projects

# Session Overview

- Real-time Scheduling
- Scheduling Techniques Overview
- Scheduling Examples



Presented by:

# The Need for Real-time Scheduling

## Real-time System Characteristics

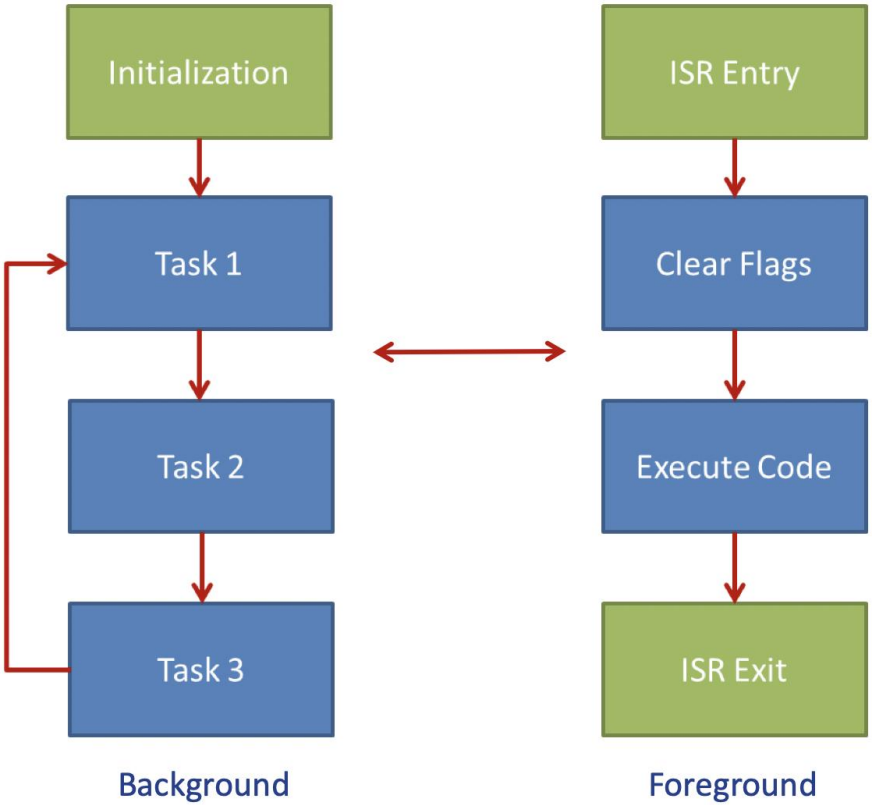
- They are event driven; do not poll inputs
- Deterministic; given the same initial conditions, they produce the same outputs in the same time frame.
- Often resource constrained in some manner such as:
  - Clock speed
  - Memory
  - Energy consumption
- Use a dedicated microcontroller-based processor
- May a RTOS to manage system tasks

# Scheduling in MicroPython

Four techniques for scheduling

- Round Robin scheduling
- Periodic scheduling using timers
- Cooperative scheduling
- MicroPython threads

# Round Robin Scheduling



# Round Robin Scheduling

```
# main.py
import pyb   # For uPython MCU features

# Setup the MCU and application code to starting conditions
# The blue LED will start on, the yellow LED will be off
def System_Init():
        print("Initializing system ...")
        pyb.LED(4).on()
        pyb.LED(3).off()
        print("Starting application ...")

# Toggle the blue LED
def Task1():
        pyb.LED(4).toggle()

# Toggle the yellow LED
def Task2():
        pyb.LED(3).toggle()
```

# Round Robin Scheduling

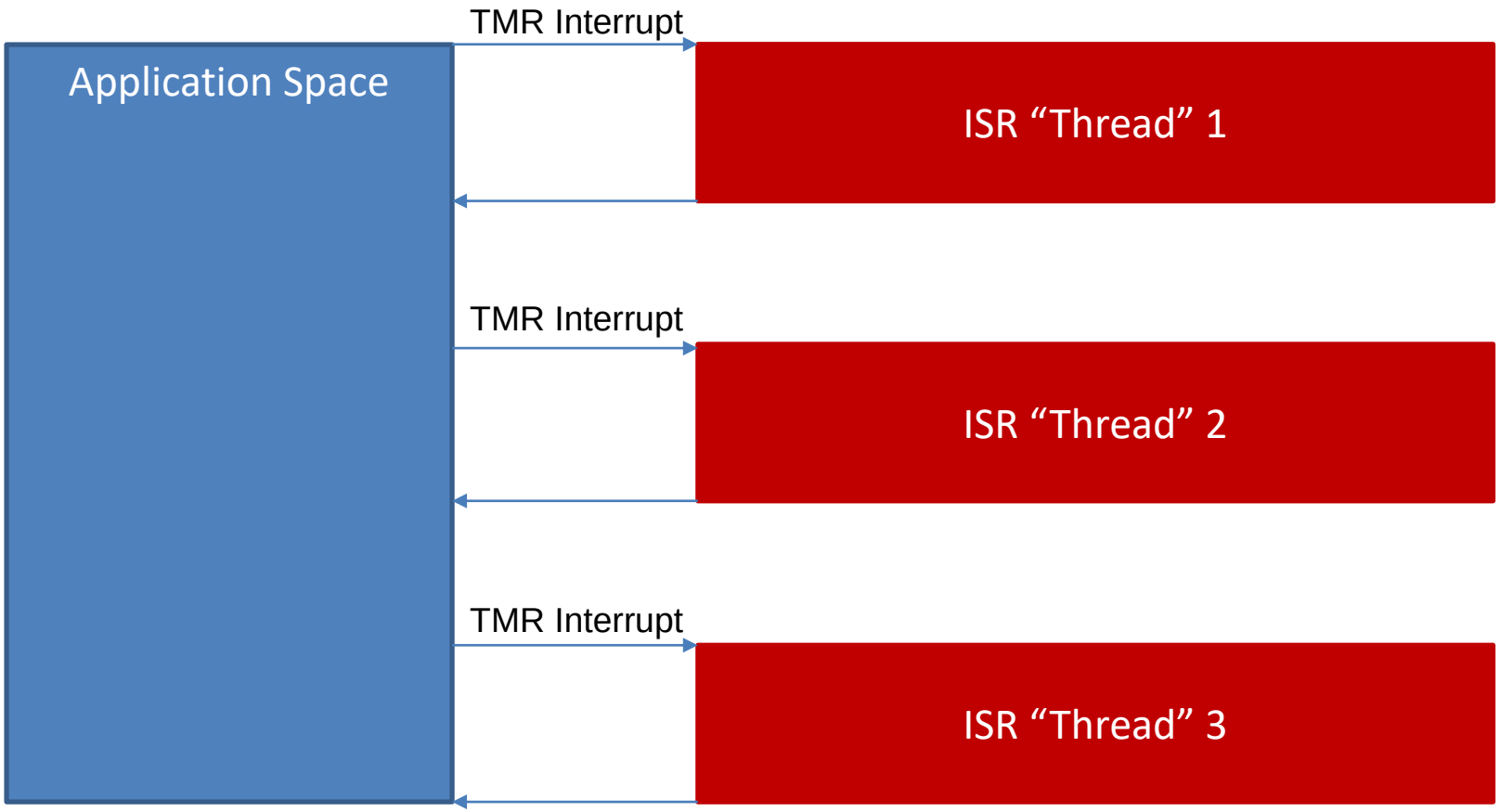
```
#################################################################
#
# Start script execution ...
#
#################################################################
# Initialize the system
System_Init()

# Main application loop
while True:
    # Run the first task
    Task1()

    #Run the second task
    Task2()

    #Delay 100 ms
    pyb.delay(150)
```

# Periodic Scheduling using Timers



# Periodic Scheduling using Timers

## Best Practices

- Keep ISR's short and fast
- Perform measurements to understand interrupt timing and latency
- Use interrupt priority settings to emulate preemption
- Make sure task variables are declared as volatile
- Avoid calling multiple functions from an ISR
- Disable interrupts as little as possible

# Periodic Scheduling using Timers

```
import micropython # For emergency exception buffer
import pyb        # For uPython MCU features

# Buffer for interrupt error messages
micropython.alloc_emergency_exception_buf(100)

# Function that contains the task code for toggling the blue LED
def Led_BlueToggle(timer):
    pyb.LED(4).toggle()

    return

# Function that contains the task code for toggling the yellow LED
def Led_YellowToggle(timer):
    pyb.LED(3).toggle()

    return
```

# Periodic Scheduling using Timers

```
# Setup the MCU and application code to starting conditions
# The blue LED will start on, the yellow LED will be off
def System_Init():
  print("Initializing system ...")
  pyb.LED(4).on()
  pyb.LED(3).off()
  print("LED's initialized ...")
```

```
# Create task timer for Blue LED
TimerBlueLed = pyb.Timer(1)
TimerBlueLed.init(freq=5)
TimerBlueLed.callback(Led_BlueToggle)
print("Blue Task initialized ...")

# Create task timer for Yellow LED
TimerYellowLed = pyb.Timer(2)
TimerYellowLed.init(freq=5)
TimerYellowLed.callback(Led_YellowToggle)
print("Yellow Task initialized ...")

print("Starting application ...")
```

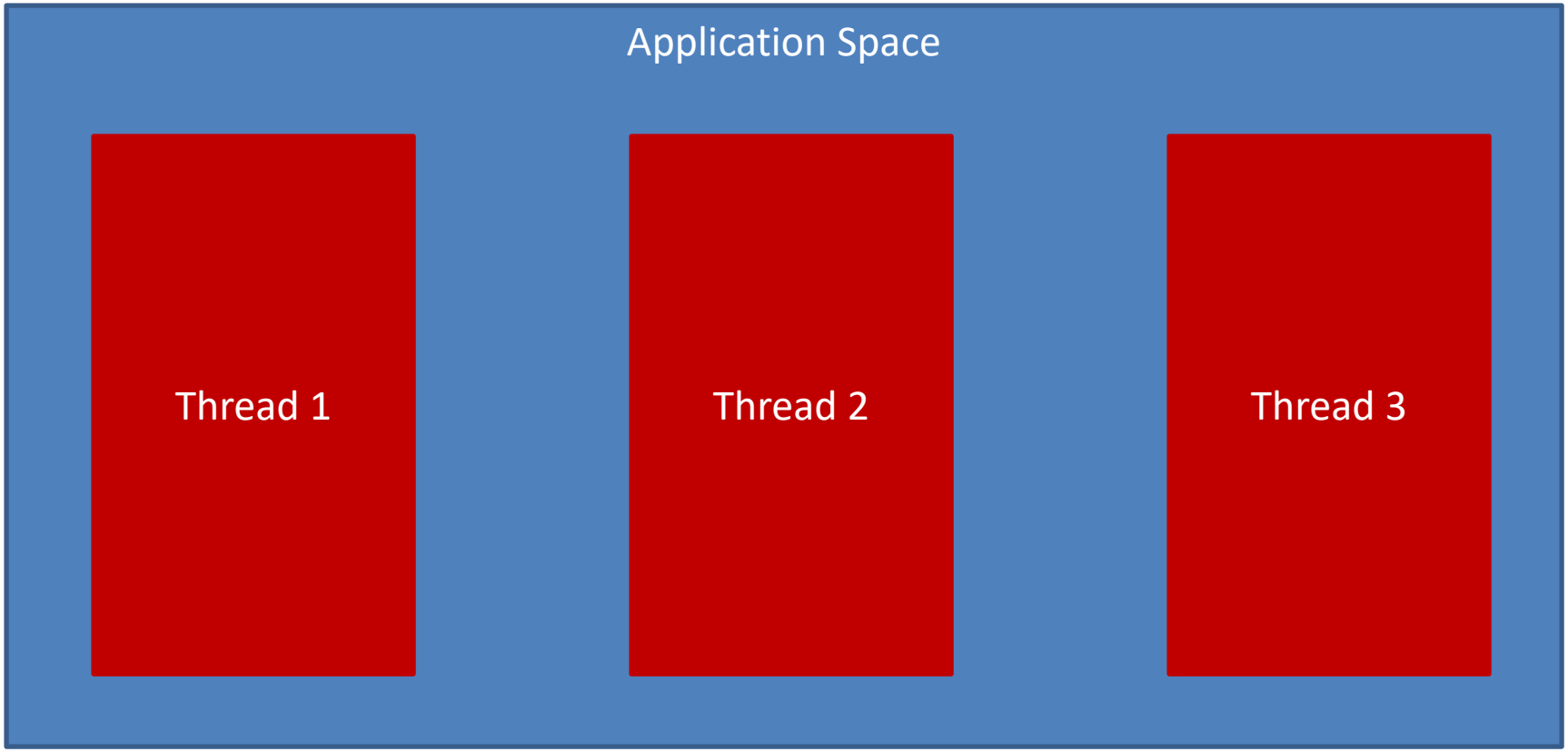
# Periodic Scheduling using Timers

```
#####################################################################
#
# Start script execution ...
#
#####################################################################
# Initialize the system
System_Init()

# Tracks seconds since program started
SecondsLive = 0

while True:
  pyb.delay(5000)
  SecondsLive = SecondsLive + 5
  print("Executing for ", SecondsLive, " seconds")
```

# Scheduling with Threads



# Scheduling with Threads

## Best Practices for using Threads

- Use locking to protect shared data between threads
- Use threads for IO related tasks
- Don’t use threads to try to speed-up processing.
- A thread will run for at most 15 ms before giving up the GIL. (Time slicing)
- Make threads safe by using atomic operations (use dis module i.e. import dis, dis.dis(function))
- Become familiar with the Python threading model at https://realpython.com/intro-to-python-threading/

# Scheduling with Threads

```
import micropython # For emergency exception buffer
import pyb        # For uPython MCU features
import _thread    # For thread support

# Buffer for interrupt error messages
micropython.alloc_emergency_exception_buf(100)

# Function that contains the task code for toggling the blue LED
def Led_BlueToggle():
   while True:
     pyb.LED(4).toggle()
     pyb.delay(250)
```

# Scheduling with Threads

```
# Function that contains the task code for toggling the yellow LED
def Led_YellowToggle():
    while True:
      pyb.LED(3).toggle()
      pyb.delay(250)

# Setup the MCU and application code to starting conditions
# The blue LED will start on, the yellow LED will be off
def System_Init():
  print("Initializing system ...")
  pyb.LED(4).on()
  pyb.LED(3).off()
  print("LED's initialized ...")
  print("Starting application ...")
```

# Scheduling with Threads

```
##################################################################
#
# Start script execution ...
#
##################################################################
# Initialize the system
System_Init()

_thread.start_new_thread(Led_BlueToggle, ())
_thread.start_new_thread(Led_YellowToggle, ())

# Tracks seconds since program started
SecondsLive = 0

while True:
  pyb.delay(5000)
  SecondsLive = SecondsLive + 5
  print("Executing for ", SecondsLive, " seconds")
```

# Additional Resources

- Download Course Material for
  - http://bit.ly/MicroPythonProjects
  - Blog
  - YouTube Videos
- Embedded Bytes Newsletter
  - http://bit.ly/1BAHYXm


From www.beningo.com under

- Blog > CEC – Designing Embedded Systems using MicroPython