Embedded System Design Techniques™

From Baremetal to RTOS

Session 2: Getting Started using Real-time Operating Systems

April 11th, 2017

Jacob Beningo

DesignNews

CEC CONTINUING EDUCATION CENTER

Presented by: Digi-Key ELECTRONICS

Course Overview

Objective:

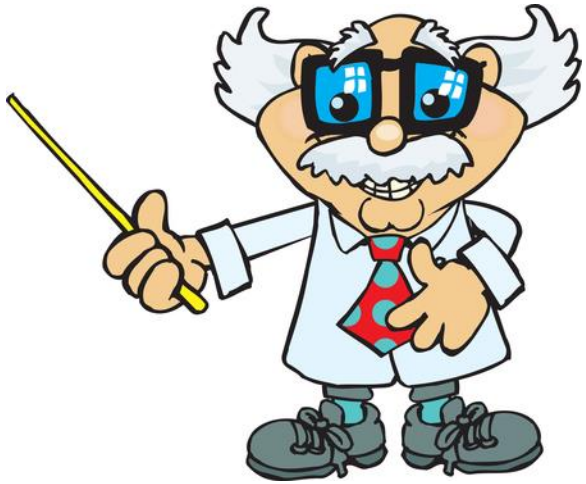
- Transitioning to using real-time operating systems

Topics:

- Reviewing Baremetal Scheduling
- **Getting Started using RTOSes**
- Real-Time Operating System Concepts
- Debugging Real-time Embedded Systems
- RTOS Best Practices

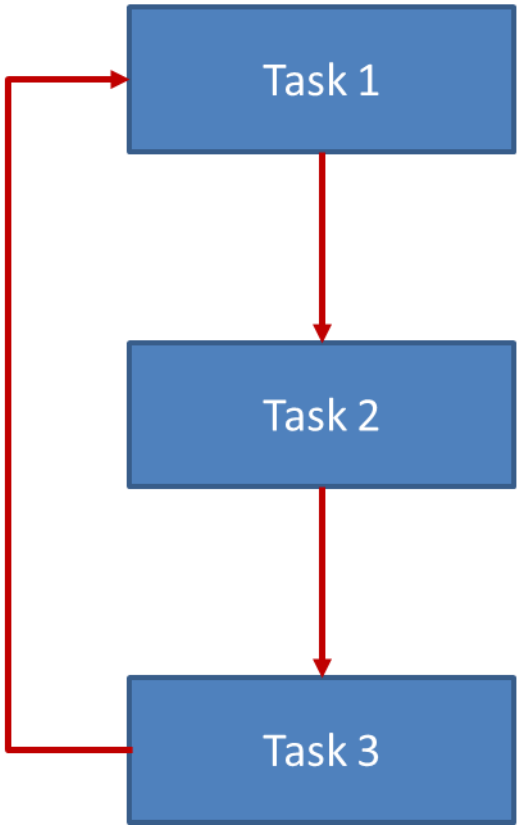
Session Overview

- Introduction
- Task Concurrency
- RTOS Characteristics
- Baremetal or RTOS?
- Task Fundamentals
- Examples
- RMA

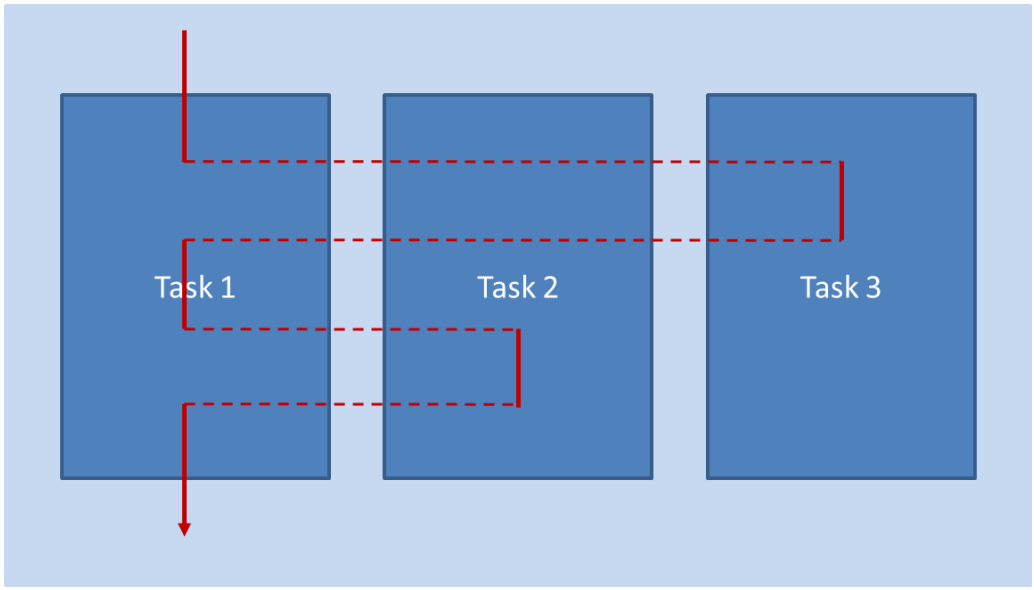


Task Concurrency

- Sequential Tasks

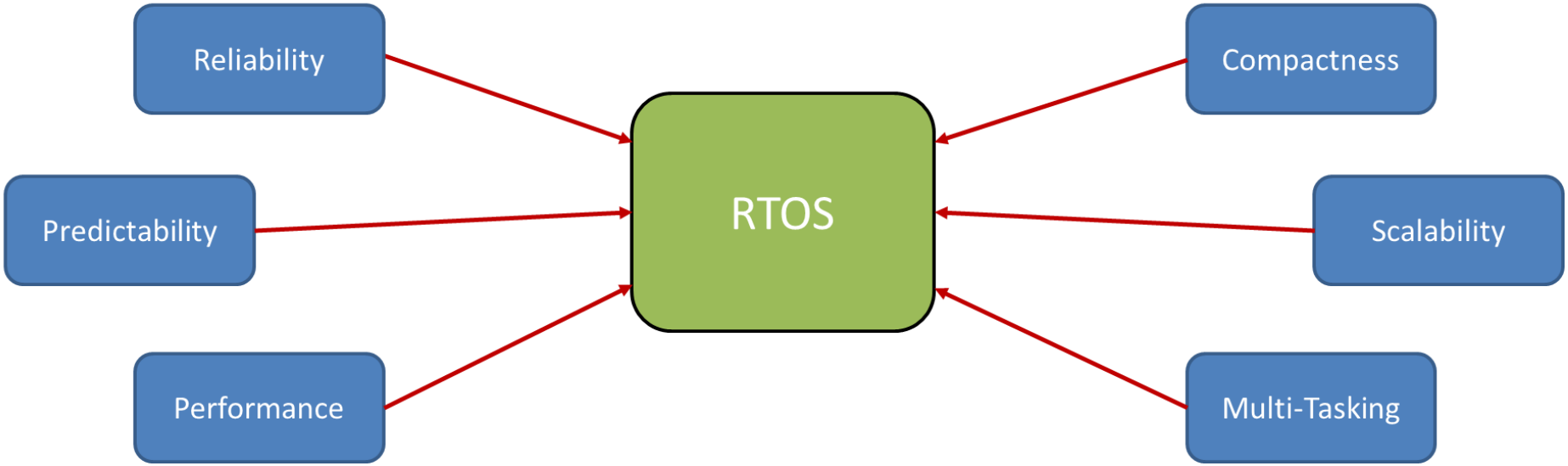


- Concurrent Tasks



RTOS Characteristics

A **Real-Time Operating System (RTOS)** is an operating system designed to manage hardware resources of an embedded system with very precise timing and a high degree of reliability.



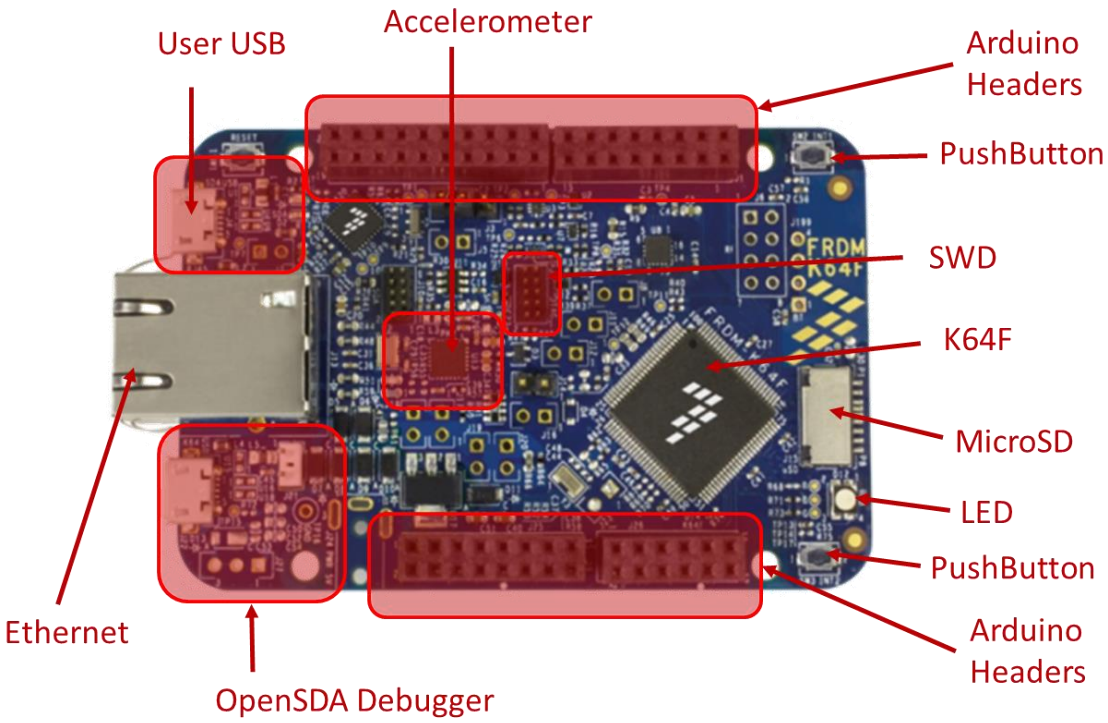
Baremetal or RTOS?

- 7 Reasons to choose an RTOS
  - Concurrency
  - Pre-emption
  - Available RAM
  - Available flash
  - Synchronization tools
  - 3rd party software
  - Ease of use

"Of course we can make fast decisions ... once we have considered the 4872 factors."

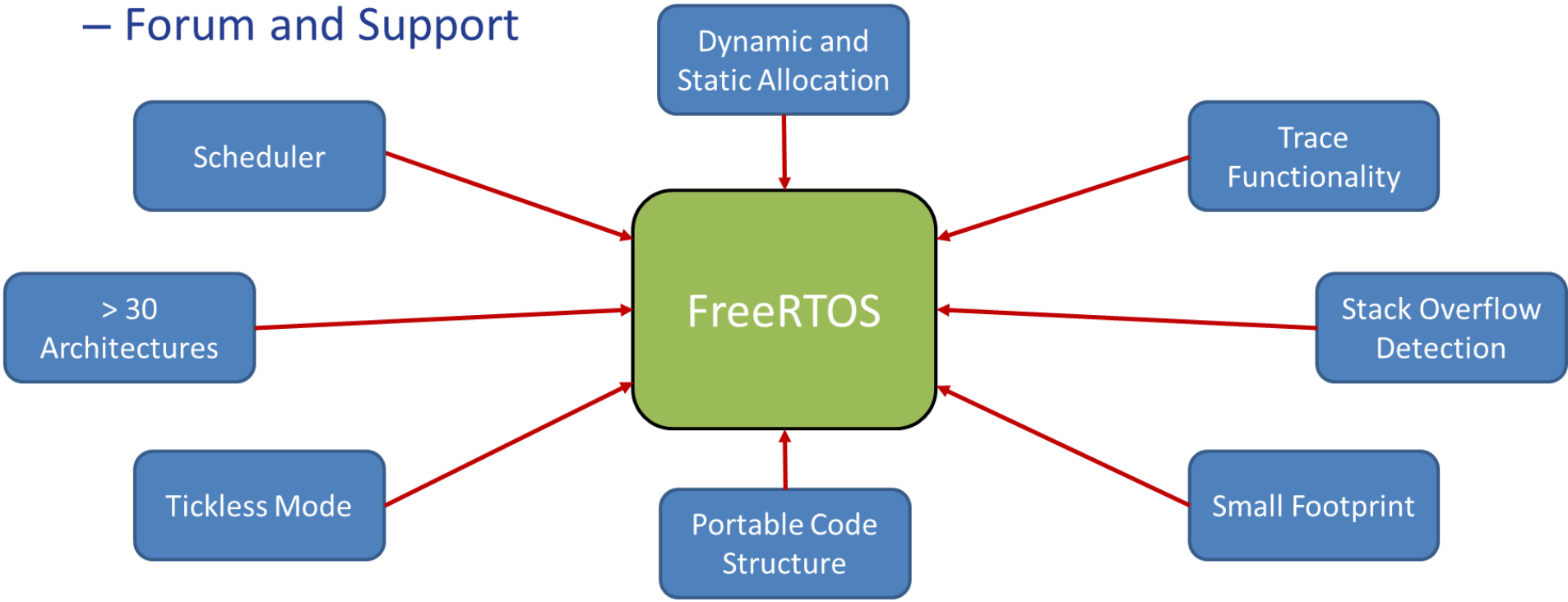
Hardware Overview

- NXP K64F Freedom Board
  - ARM Cortex-M4
  - 120 MHz
  - 1 MB Flash
  - 256 KB RAM
  - Tri-Color LED
  - Accelerometer
  - Built-in Debugger
  - Ethernet
  - $35



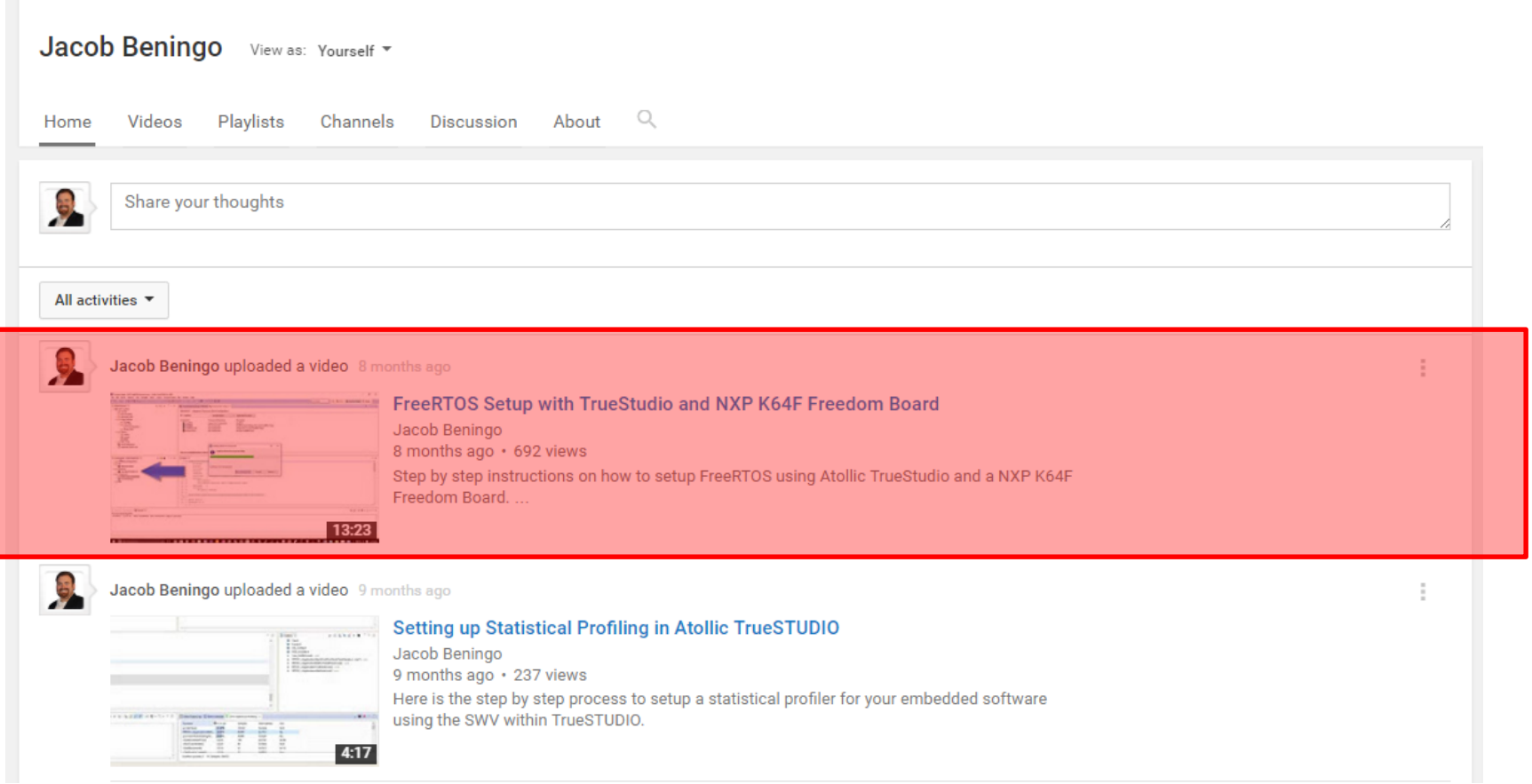
FreeRTOS

- FreeRTOS.org
  - Ports and source available
  - Forum and Support



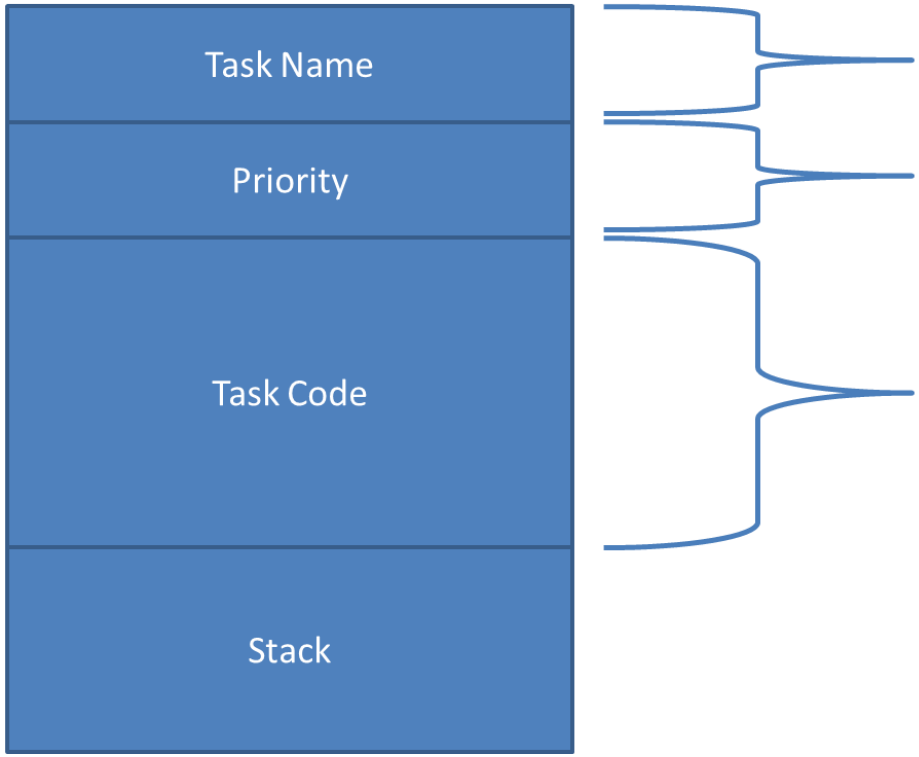
YouTube Demonstrations

https://www.youtube.com/channel/UC9k8GahBTE0IVJxOsL4WhOA



Task Fundamentals

The Task Control Block (TCB)



12 Character Max (TaskGreenLED)

0 – 31 (configMAX_PRIORITIES – 1)

```
void Led_GreenBlink( void *pvParameters )
{
   for(;;)
   {
     /* Place task code here */
   }
}
```

Task Fundamentals

- Creating a task in FreeRTOS

```
xTaskCreate(
  Led_BlueBlink,                    /* Task Pointer */
  (const char* const)"led_blue",    /* Task Name */
  configMINIMAL_STACK_SIZE,         /* Stack Depth */
  0,                                /* Parameters to Pass to task */
  1,                                /* Task Priority */
  0);                               /* Pass handle to created task */
```

Example

```
128 /*lint -save  -e970 Disable MISRA rule (6.3) checking. */
129 int main(void)
130 /*lint -restore Enable MISRA rule (6.3) checking. */
131 {
132   /* Write your local variable definition here */
133
134   /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
135   PE_low_level_init();
136   /*** End of Processor Expert internal initialization.                    ***/
137
138   /* Write your code here */
139   /* For example: for(;;) { } */
140   xTaskCreate(Led_GreenBlink,               /* Task Pointer */
141               (const char* const)"led_green",/* Task Name */
142               configMINIMAL_STACK_SIZE,     /* Stack Depth */
143               0,                            /* Parameters to pass to task*/
144               3,                            /* Task Priority */
145               0);                           /* Pass handle to created task */
146
147   xTaskCreate(Led_RedBlink,                 /* Task Pointer */
148               (const char* const)"led_red", /* Task Name */
149               configMINIMAL_STACK_SIZE,     /* Stack Depth */
150               0,                            /* Parameters to pass to task*/
151               2,                            /* Task Priority */
152               0);                           /* Pass handle to created task */
153
154   xTaskCreate(Led_BlueBlink,                /* Task Pointer */
155               (const char* const)"led_blue",/* Task Name */
156               configMINIMAL_STACK_SIZE,     /* Stack Depth */
157               0,                            /* Parameters to pass to task*/
158               1,                            /* Task Priority */
159               0);                           /* Pass handle to created task */
```

Example

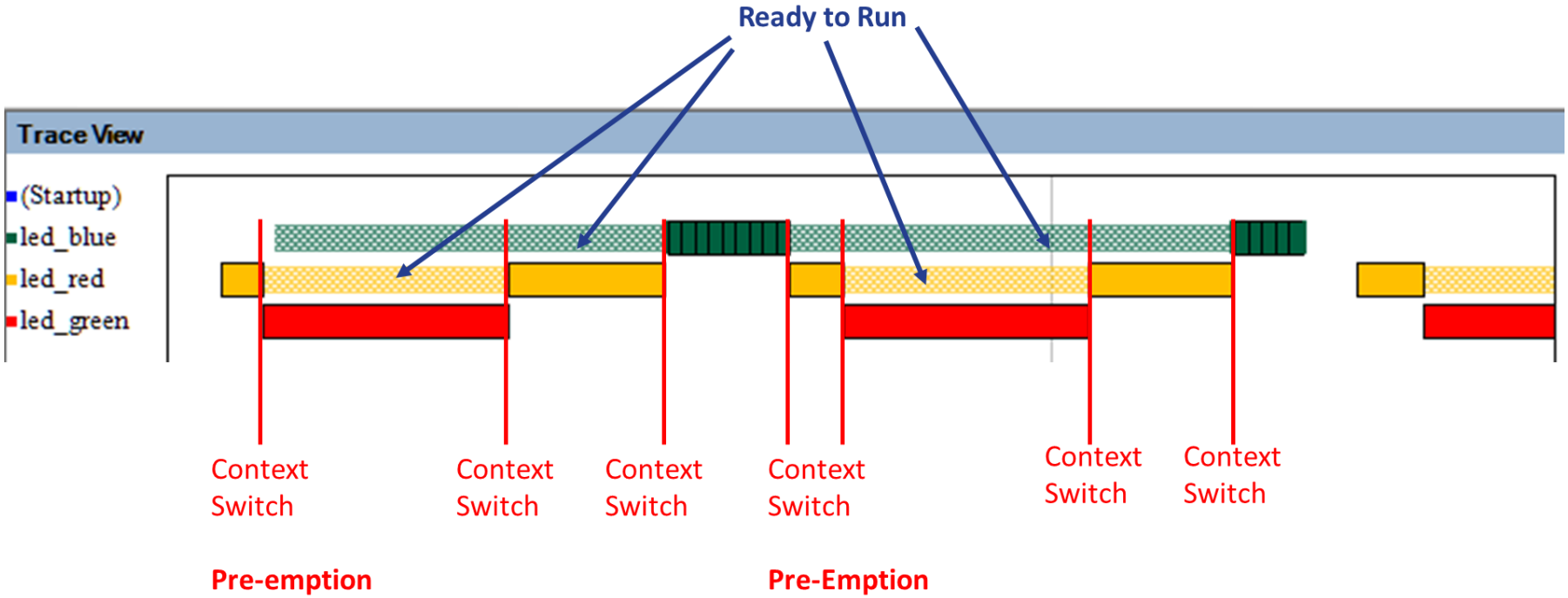
Semi-independent Programs

```
64  void Led_BlueBlink(void *pvParameters)
65  {
66      const TickType_t xDelay = 500 / portTICK_PERIOD_MS;
67      uint32_t BlueDelay = 0;
68      const uint32_t TargetCount = 160000;
69
70      for(;;)
71      {
72          LED_Blue_On();
73          Delay_Nonsense(&BlueDelay, &TargetCount);
74          vTaskDelay(xDelay);
75          LED_Blue_Off();
76          Delay_Nonsense(&BlueDelay, &TargetCount);
77          vTaskDelay(xDelay);
78      }
79  }

81  void Led_RedBlink( void *pvParameters )
82  {
83      const TickType_t xDelay = 100 / portTICK_PERIOD_MS;
84      uint32_t RedDelay = 0;
85      const uint32_t TargetCount = 160000;
86
87      for( ;; )
88      {
89          LED_Red_On();
90          Delay_Nonsense(&RedDelay, &TargetCount);
91          vTaskDelay( xDelay );
92          LED_Red_Off();
93          Delay_Nonsense(&RedDelay, &TargetCount);
94          vTaskDelay( xDelay );
95      }
96  }
```

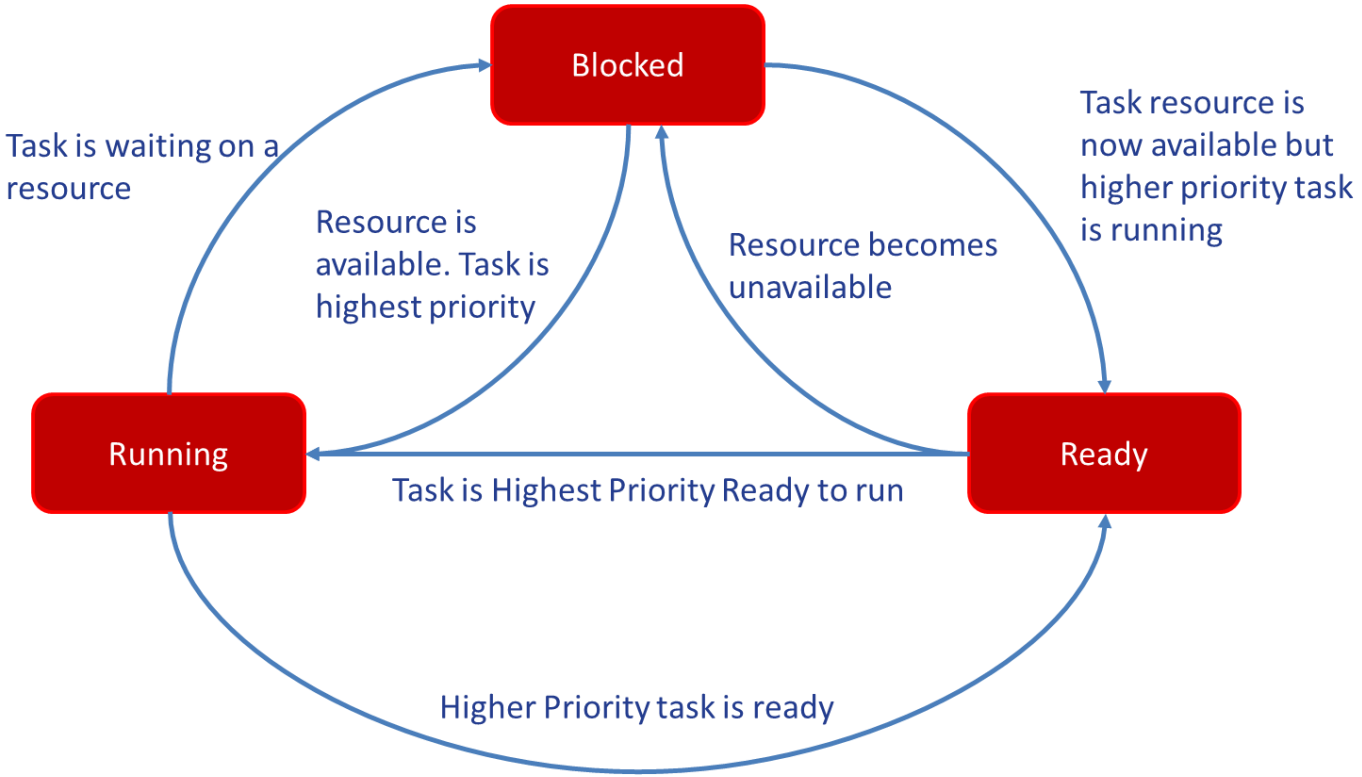
Task Fundamentals

- Context Switching



Task Fundamentals

- Task States



Rate Monotonic Analysis

Method for setting initial task priorities

Assumptions:

- Tasks are periodic
- Tasks are independent
- Preemptive scheduling is used

Rate Monotonic Analysis

- CPU Utilization (U)

$$\sum_{k=1}^{n} \frac{E_k}{T_k} \leq n\left(2^{\frac{1}{n}} - 1\right)$$

U(1) = 1.0
U(2) = 0.828
U(3) = 0.779
U(4) = 0.756
U(5) = 0.743
U(6) = 0.734
U(∞) = 0.693

Rate Monotonic Analysis

	Execution	Period	Utilization
Task 1	15	100	0.15
Task 2	30	150	0.20
Task 3	60	300	0.20

Total Utilization = 0.15 + 0.20 + 0.20 = **0.52**

$$0.52 \leq U(3)$$

$$0.52 \leq 0.779$$

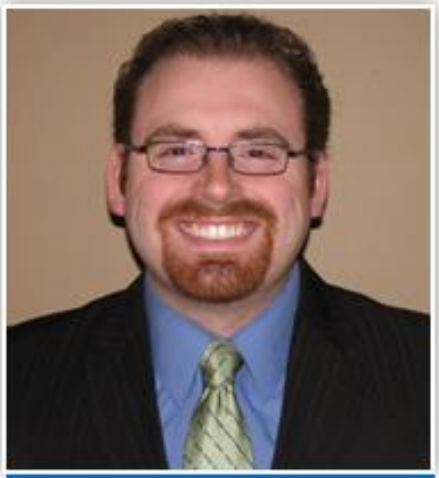
Additional Resources

- Download Course Material for
  - Updated C Doxygen Templates (Dec 2016)
  - Example source code
  - Templates
  - YouTube Videos
- Embedded Bytes Newsletter
  - http://bit.ly/1BAHYXm

From www.beningo.com under

- Blog > CEC – Baremetal to RTOS

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 Workshops

- Bootloader Design
- RTOS Workshop
- Design Acceleration

BENINGO EMBEDDED GROUP

www.beningo.com