Embedded System Design Techniques™

From Baremetal to RTOS

Session 1: Reviewing Baremetal Scheduling Techniques

April 10th, 2017

Jacob Beningo

Presented by:

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

The Lecturer – Jacob Beningo

Jacob Beningo
Principal Consultant

Social Media / Contact

E	: jacob@beningo.com
T	: 248-719-6850
	: Jacob_Beningo
f	: Beningo Engineering
in	: 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

Presented by:

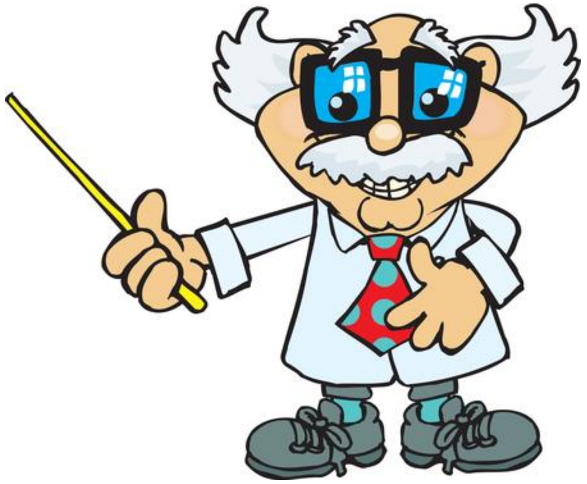
CEC CONTINUING EDUCATION CENTER

Jacobs CEC Courses

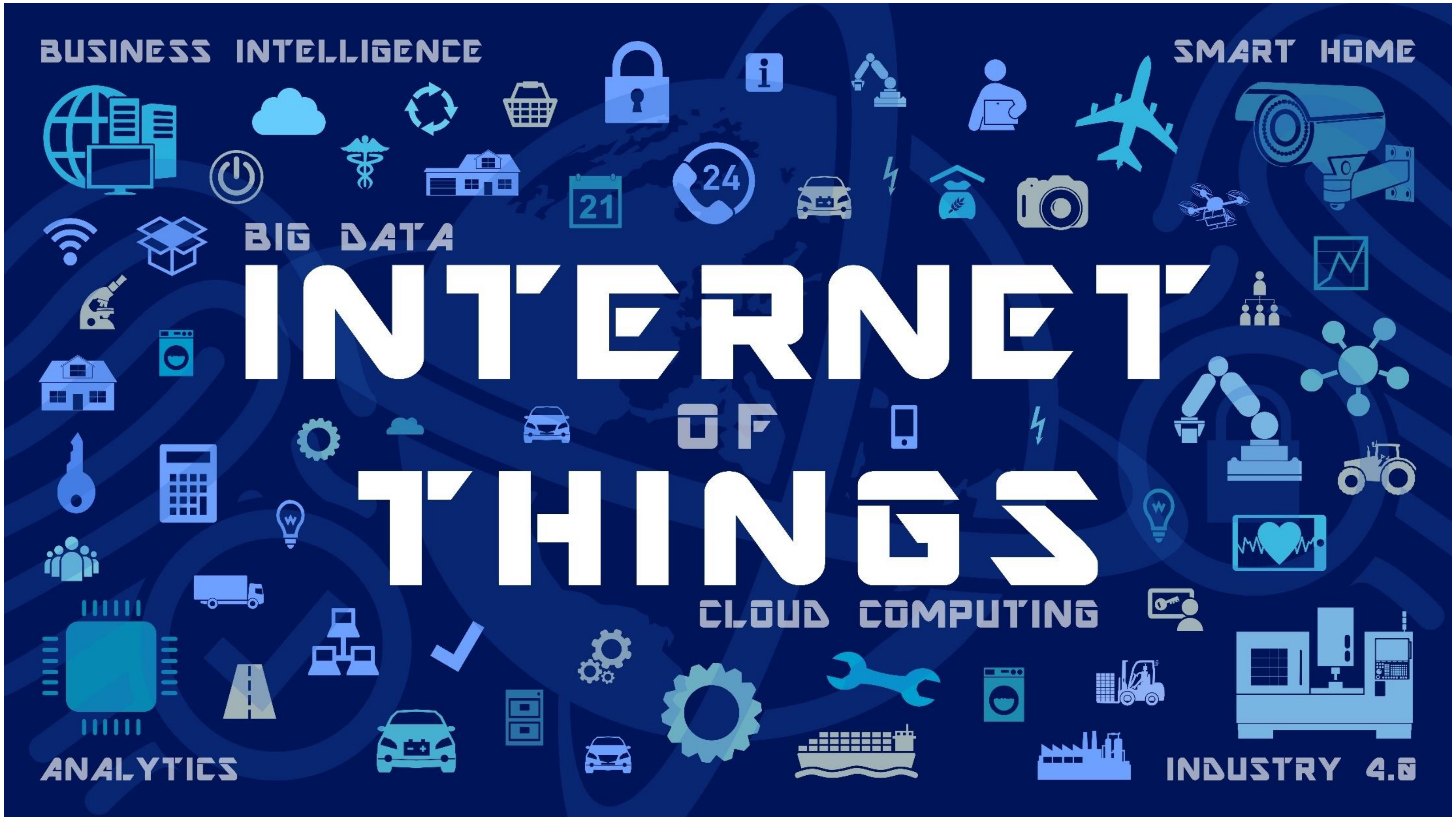
CEC 2013 – 2015	CEC 2016 2017	Side Topics 2017
Fundamentals of Embedded Software (2013)	Bootloader Design for MCUs (2016)	Real-Time Software using Micro Python
Mastering the Software Design Cycle (2014)	Rapid Prototyping w/ Micro Python (2016)	Embedded Bytes Newsletter
Python for Embedded Systems(2014)	Debugging (2016)	http://bit.ly/1BAHYXm
Software Architecture Design (2014)	Professional Firmware (2016)	
Baremetal C (2015)	API's and HAL's February 2017	
Mastering the ARM Cortex-M Processor (2015)	Baremetal to RTOS April 2017	
Writing Portable and Robust Firmware in C (2015)	Designing IoT Sensor Nodes July 2017	
Design Patterns and the Internet (2015)	From C to C++ October 2017	

Session Overview

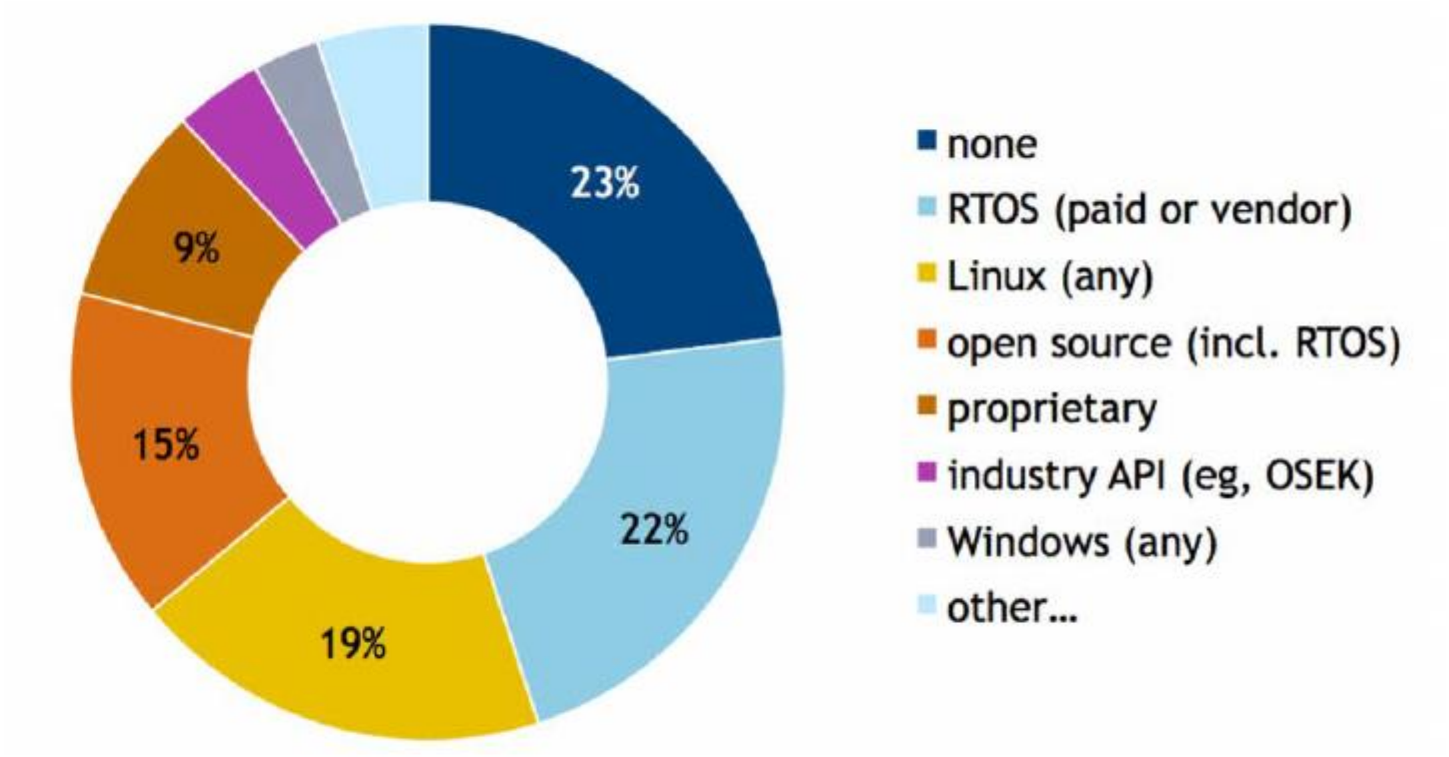
- Introduction
- Review - Round Robin
- Review – Round Robin w/ Interrupts
- Cooperative Scheduling
- Creating your own scheduler



Introduction



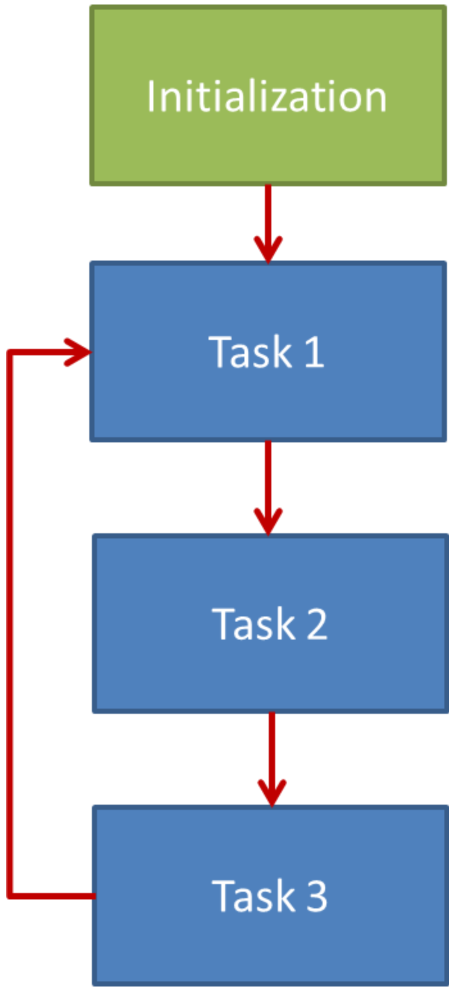
Baremetal or RTOS?



Source: Barr Group 2017 Industry Survey

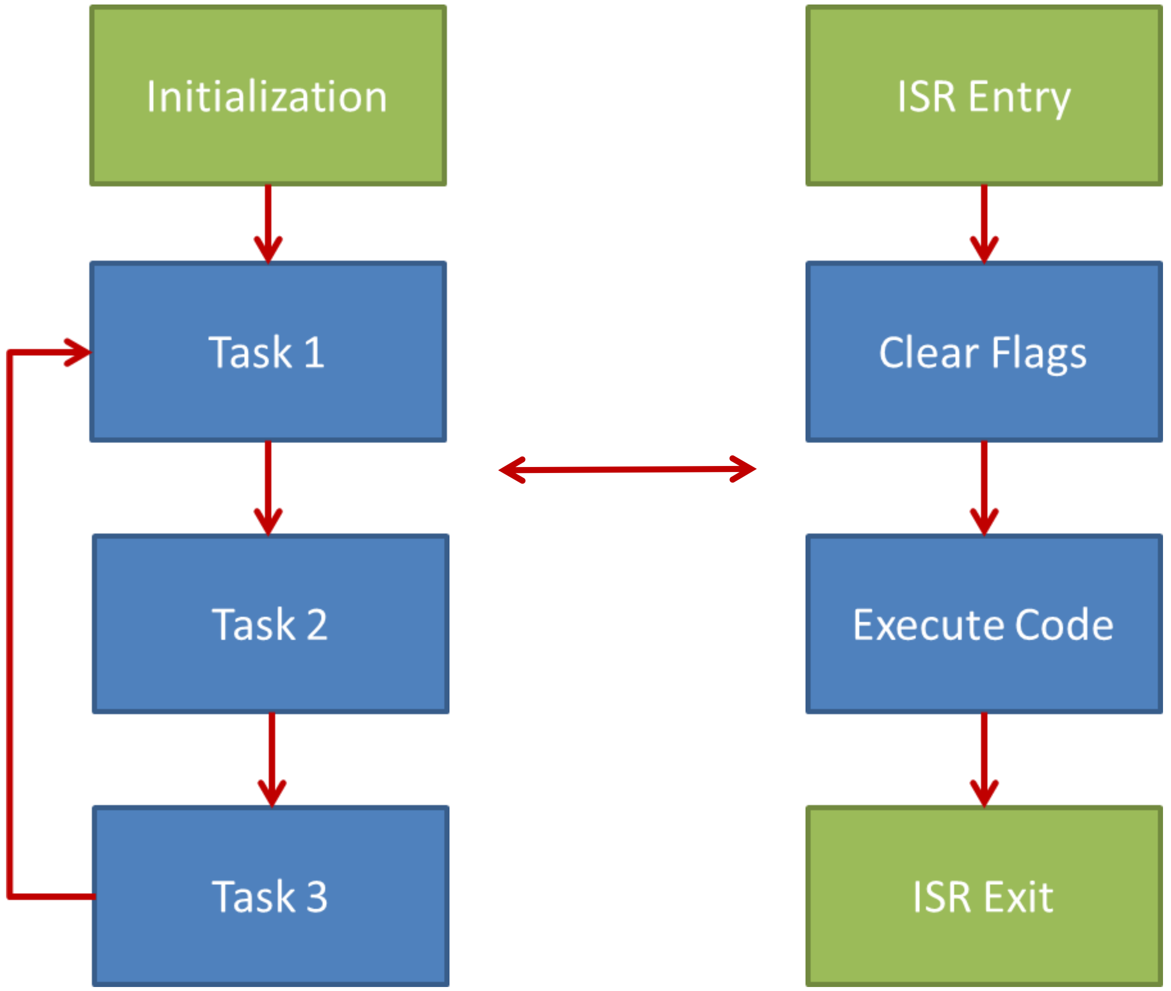
Reviewing Baremetal Scheduling

```
int main (void) {
   System_Init();

   while(1) {
      // Run the first task
      Task1();

      // Run the first task
      Task2();

      // Run the first task
      Task3();
   }

   return 1;
}
```

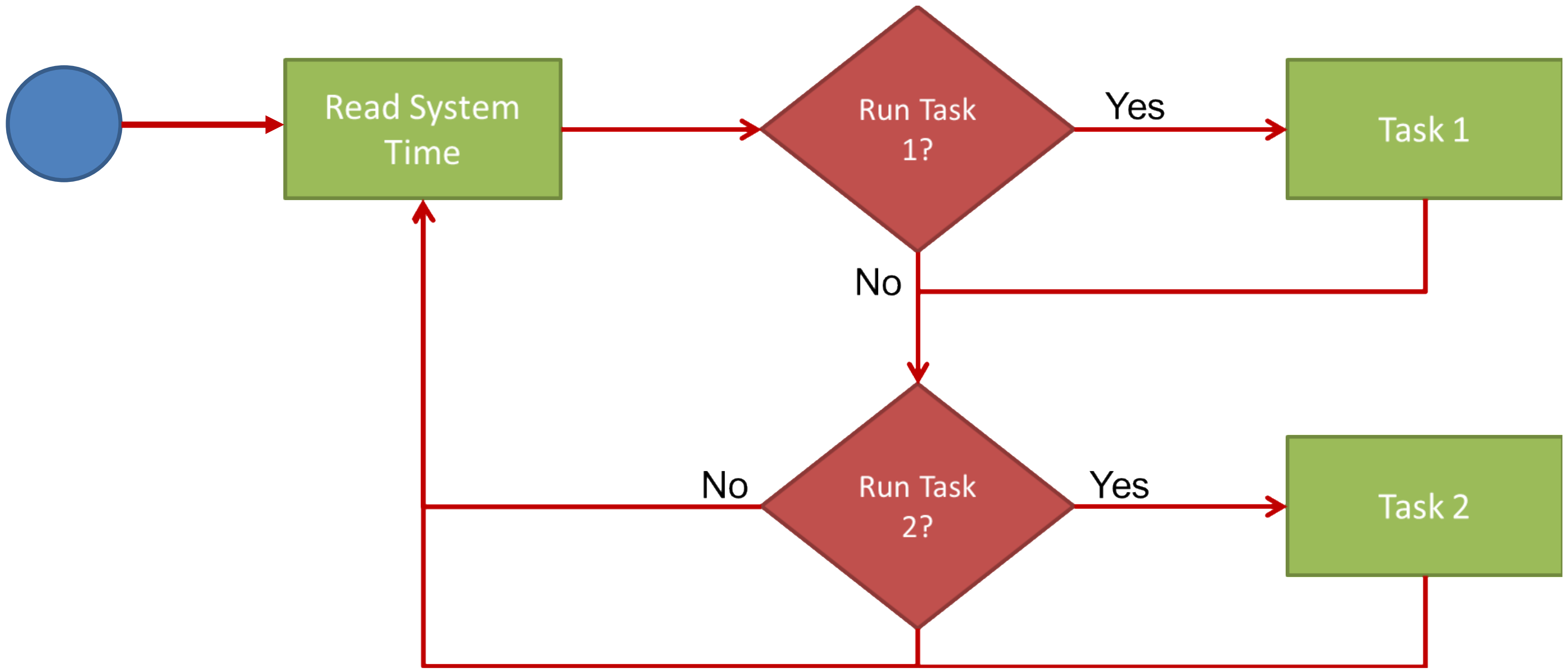


Reviewing Baremetal Scheduling

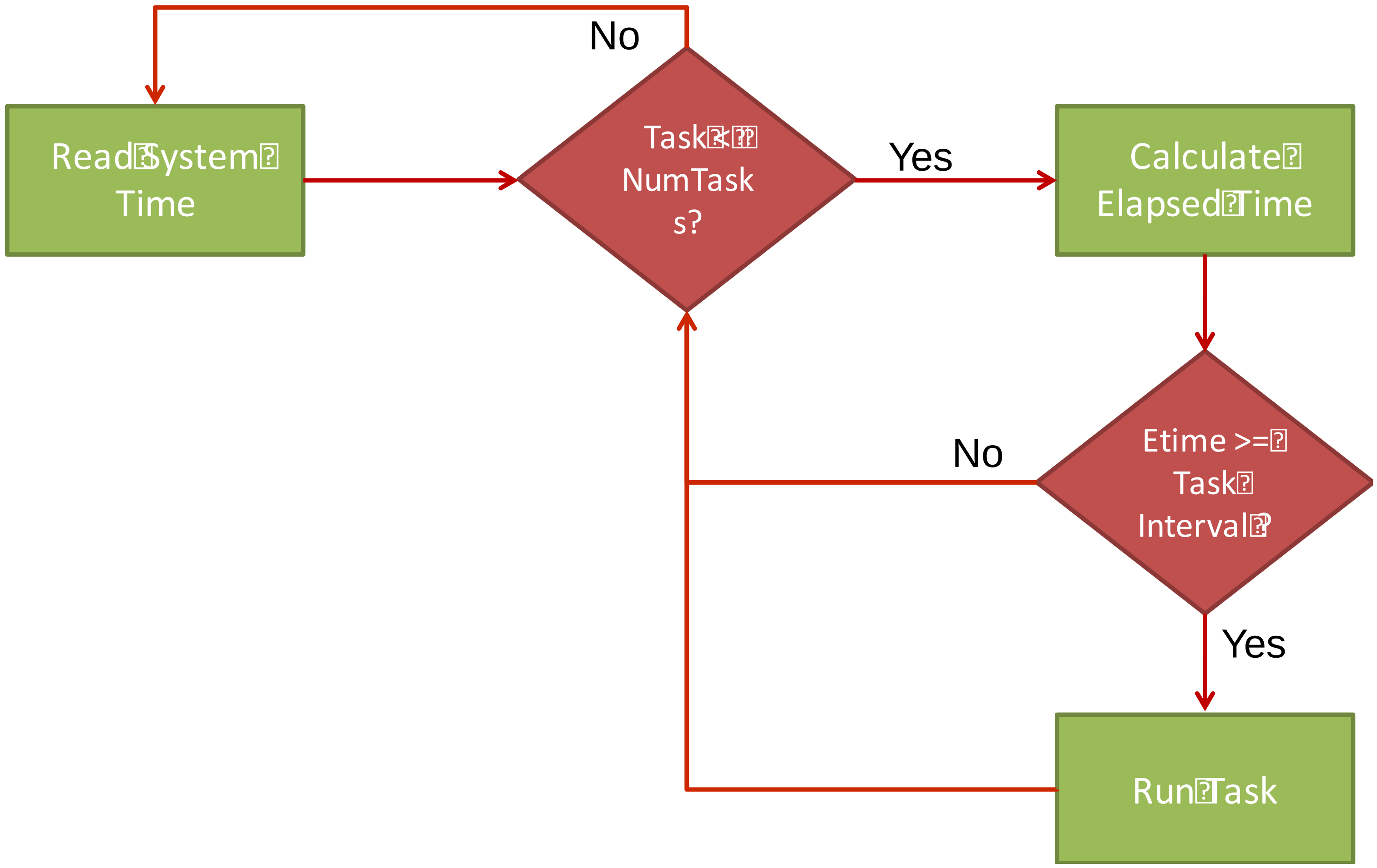
```
int main (void) {
   System_Init();

   while(1) {
      // Run the first task
      Task1();

      // Run the first task
      Task2();

      // Run the first task
      Task3();
   }

   return 1;
}
```



Cooperative Scheduling



Cooperative Scheduling



Cooperative Scheduler

```
int main(void)
{
    static uint32_t tick = 0;                                   // System tick
    static TaskType *Task_ptr;                                  // Task pointer
    static uint8_t TaskIndex = 0;                               // Task index
    const uint8_t NumTasks = Task_GetNumTasks();  // Number of tasks

    /**************************************************************************
    * System Initialization
    **************************************************************************/
    Task_ptr = Task_GetConfig();    // Get a pointer to the task configuration

    Sys_Init();   // Initializes the system and all peripherals
}
```

Cooperative Scheduler

```
/**
 * Struct TaskType
 * TaskType structure is used to define the parameters required in order to
 * configure a task.
 */
typedef struct
{
   uint16_t Interval;                          /**< Defines how often a task will run  */
   uint32_t LastTick;                          /**< Stores the last tick task was ran  */
   void (*Func)(void);                         /**< Function pointer to the task  */
}TaskType;
```

Cooperative Scheduler

```
/**
 * Task configuration table.  Holds the task interval, last time executed, and
 * the function to be executed.  A continuous task is defined as a task with
 * an interval of 0.  Last time executed is set to 0.
 */
static TaskType Tasks[] =
{
        { INTERVAL_00MS,    0, Task           },
        { INTERVAL_10MS,    0, Task_10ms   },
        { INTERVAL_100MS,  0, Task_100ms },
};

/**
 * Defines the number of tasks that are currently scheduled to run.
 */
uint8_t Task_Number = sizeof(Tasks) / sizeof(*Tasks);
```

Cooperative Scheduler

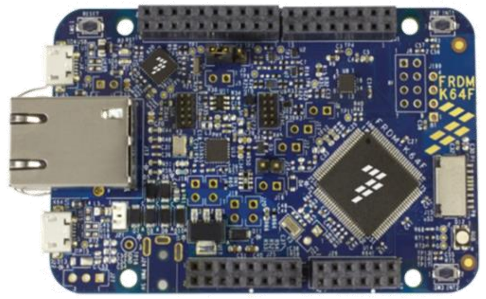
```
TaskType *Task_GetConfig(void)
{
		return Tasks;
}

uint8_t Task_GetNumTasks(void)
{
		return sizeof(Tasks) / sizeof(*Tasks);
}

void Task (void)
{
  // Continuous Task Code
}

void Task_10ms (void)
{
  // Code executed every 10ms
}
```

Cooperative Scheduler

```
// The main while loop.  This while loop will run the program forever
while(1)
{
  tick = Tmr_GetSystemTick();// Get current system tick

  // Loop through all tasks.  First, run all continuous tasks.  Then,
  // if the number of ticks since the last time the task was run is
  // greater than or equal to the task interval, execute the task.
  for(TaskIndex = 0; TaskIndex < NumTasks; TaskIndex++)
  {
    else if((tick - Task_ptr[TaskIndex].LastTick) >= Task_ptr[TaskIndex].Interval)
    {
      (*Task_ptr[TaskIndex].Func)();        // Execute Task

      Task_ptr[TaskIndex].LastTick = tick;  // Save last tick the task was ran.
    }
   }// end for
}// end while(1)
```

Hands-on Example Materials

NXP K64F Freedom Board

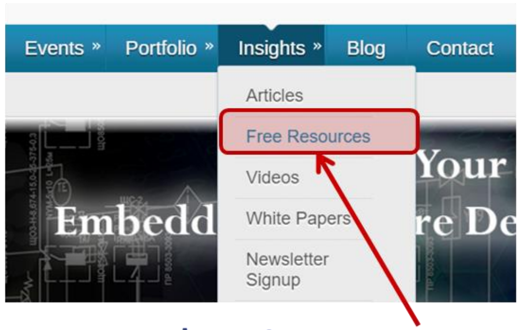


FRDM-K64F

Atollic TrueSTUDIO™

atollic.com

Source Examples



www.beningo.com

MCUonEclipse

mcuoneclipse.com

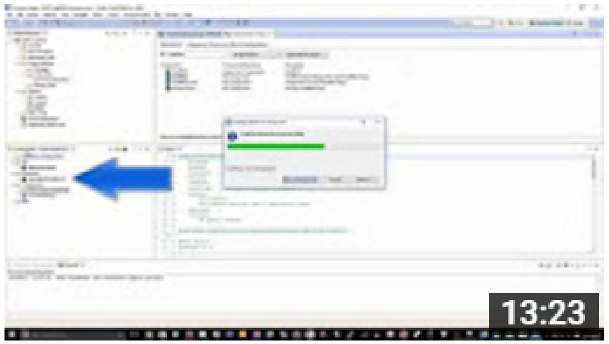
Percepio Tracealyzer

percepio.com

Demonstration Setup

- Download and Install
  - Atollic TrueStudio 7.1
  - Processor Expert Driver Suite 10.4
  - Percepio Tracealyzer Demo
  - Percepio Tracealyzer Plug-in (http://percepio.com/exporter)
  - MCUonEclipse PE Components

Google Search: YouTube Jacob Beningo



FreeRTOS Setup with TrueStudio and NXP K64F Freedom Board

Jacob Beningo

7 months ago • 518 views

Step by step instructions on how to setup FreeRTOS using Atollic TrueStudio and a NXP K64F Freedom Board. …

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
f	: Beningo Engineering
in	: 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