CEC Continuing Education Center

DesignNews

Expert C Techniques to Master Baremetal Programming

DAY 2 : Building Cooperative Schedulers and Command Parsers

Sponsored by

DigiKey

BENINGO EMBEDDED GROUP

informa markets

Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Group Chat’ by maximizing the chat widget in your dock.

THE SPEAKER

Jacob Beningo

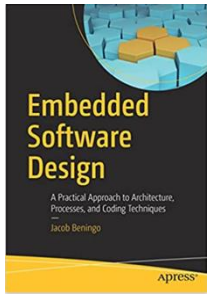
Jacob@beningo.com

jacobbeningo

Beningo Embedded Group – CEO / Founder

Focus: Software Architecture, Processes, and Dev Skills

At Beningo Embedded Group, we believe everyone deserves the skills to confidently advance their careers, meet deadlines, and deliver quality embedded systems. We provide modern strategies, insights, and hands-on training to equip developers and teams with the tools they need to succeed.



EMBEDDED SOFTWARE ACADEMY BY BENINGO

Visit **www.beningo.com** to learn more

Baremetal Scheduling

01

- No scheduler. No fancy operating system. Just a developer, a timer and the hardware.

Baremetal Scheduling

Round Robin Scheduling

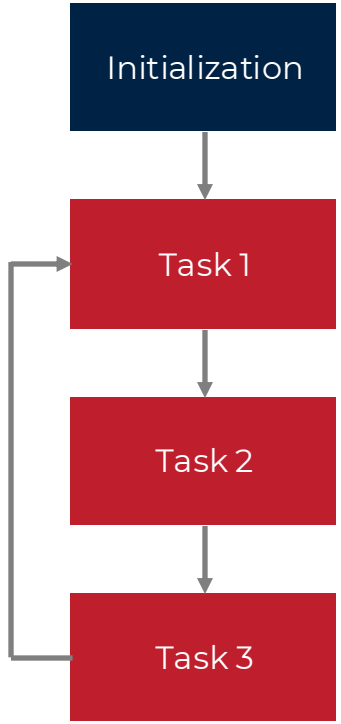
```
int main (void)
{
  System_Init();

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

    // Run the first task
    Task2();

    // Run the first task
    Task3();
  }

  return 1;
}
```



Background

Baremetal Scheduling

Round Robin Scheduling with Interrupts

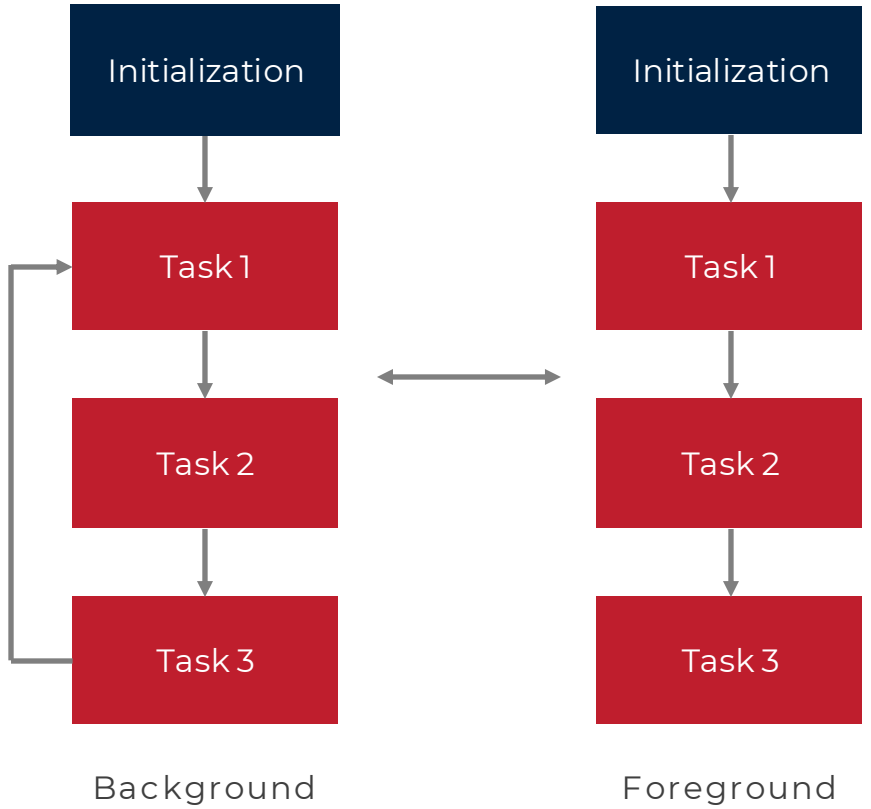
```
int main (void)
{
  System_Init();

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

    // Run the first task
    Task2();

    // Run the first task
    Task3();
  }

  return 1;
}
```



Audience POLL Question

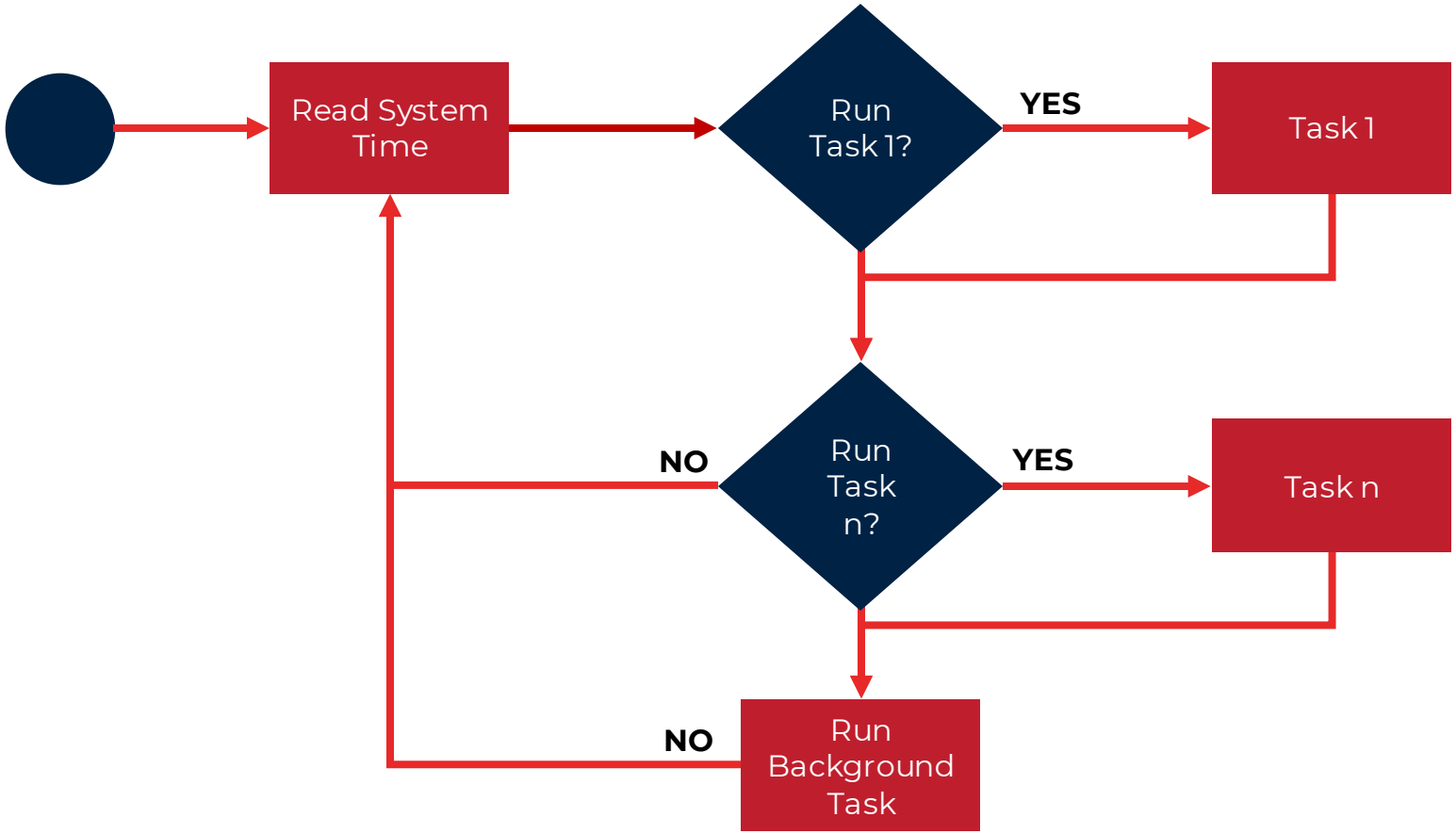
What type of scheduler do you use?

a) Round Robin
b) Round Robin with Interrupts
c) Cooperative Scheduler
d) RTOS
e) Other

02 Designing a Cooperative Scheduler

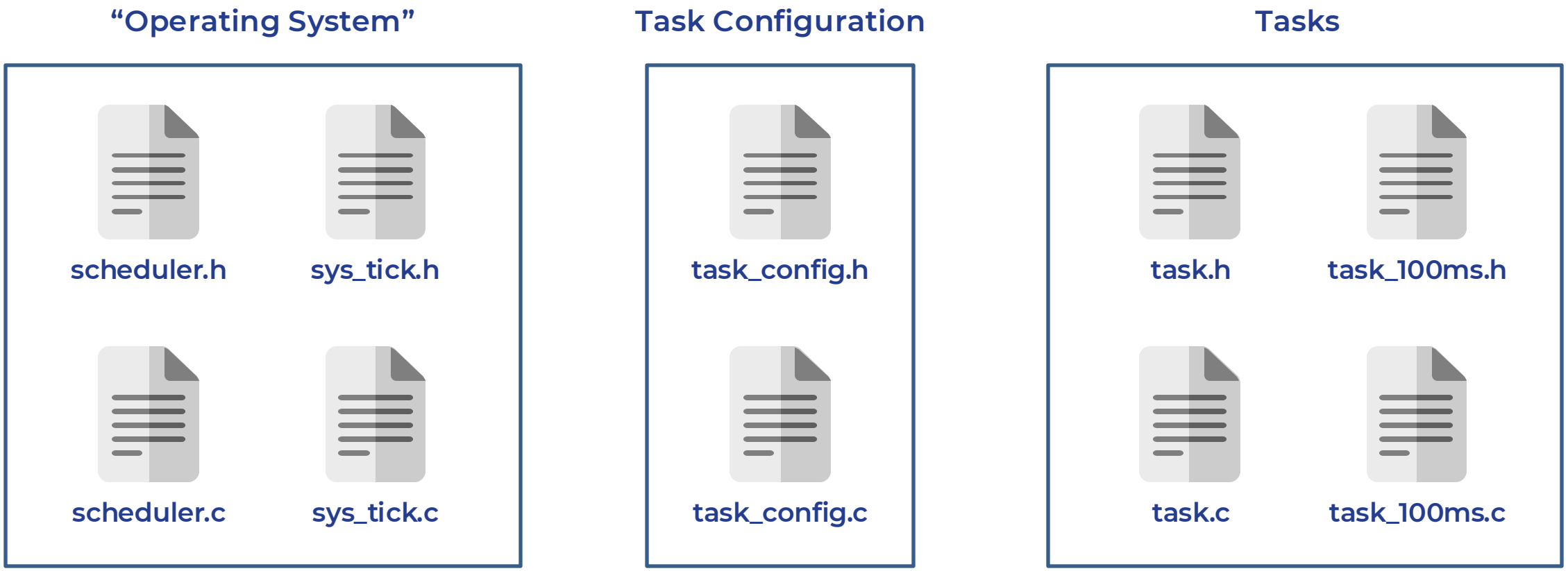
Designing a Cooperative Scheduler

Design



Designing a Cooperative Scheduler

Organization



Designing a Cooperative Scheduler

Example Main

main.c

```
int main(void)
{
    // Initialize Peripherals and Application
    System_Initialize();

    // Initialize the scheduler
    OS_Init();

    for(;;)
    {
        OS_Run(HAL_GetTick);
    }

    return 0;
}
```

Setup the scheduler

Run forever!

Timebase

Cooperative scheduler

task_10ms.c

```
void Task_10ms()
{
    Led_Toggle(LED_GREEN);
}
```

task_100ms.c

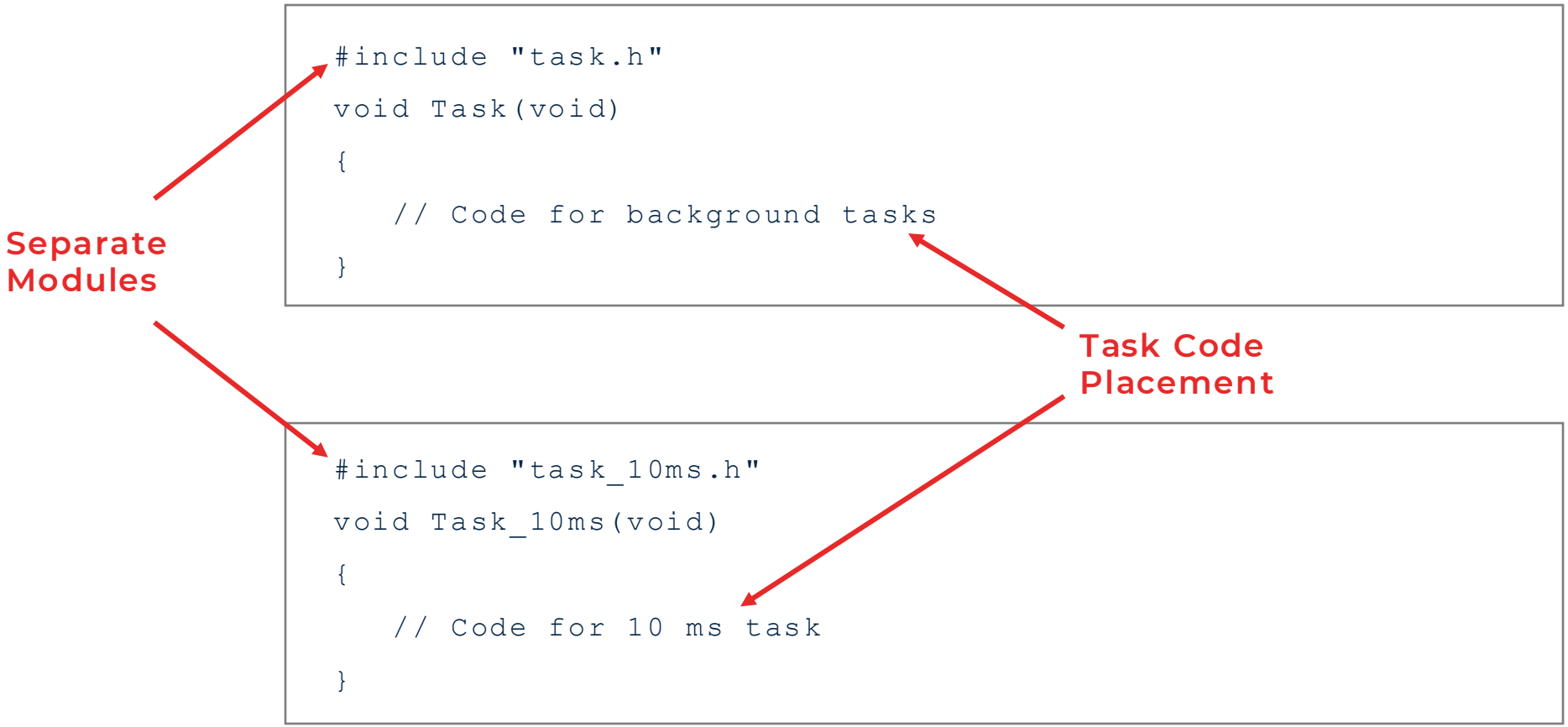
```
void Task_100ms()
{
    Led_Toggle(LED_RED);
}
```

task_500ms.c

```
void Task_500ms()
{
    Led_Toggle(LED_BLUE);
}
```

Designing a Cooperative Scheduler

Creating Tasks



Audience POLL Question

How reusable do you think this scheduler is?

a) Not at all
b) Somewhat
c) Very
d) Other

03

Cooperative Scheduler Implementation

Cooperative Scheduler Implementation

Task Management

What is the minimum amount of information required to schedule a periodic task?

- How often does the task need to run? **(Interval)**
- When did the task last run? **(LastTick)**
- What function should be called when it is time to run the task? **(Function)**

task_config.h

Task_t

```
typedef struct
{
    uint32_t Interval;
    uint32_t LastTick;
    void(*Function)(void);
}Task_t;
```

Function Pointer

Cooperative Scheduler Implementation

Task Intervals

task_config.h

```
// Defines the rate at which the system tick is occurring in
microseconds
#define SYS_TICK_INTERVAL                                    1000UL
```

Microseconds

Background Interval

Periodic Time Intervals

```
// Define the number of system ticks required for each interval
#define INTERVAL_0MS            0
#define INTERVAL_10MS           (10000UL / SYS_TICK_INTERVAL)
#define INTERVAL_50MS           (50000UL / SYS_TICK_INTERVAL)
#define INTERVAL_100MS          (100000UL / SYS_TICK_INTERVAL)
#define INTERVAL_500MS          (500000UL / SYS_TICK_INTERVAL)
#define INTERVAL_1000MS         (1000000UL / SYS_TICK_INTERVAL)
```

Cooperative Scheduler Implementation

OS_Init

task_config.c

```
Task_t Tasks[] =
{
    { INTERVAL_10MS ,   0, Task_10ms       },
    { INTERVAL_0MS  ,   0, Task_Background }
};
```

Tasks to run

scheduler.c

```
Task_t *Task_ptr = NULL;          // Points to a table of tasks


void OS_Init(void)
{
    // Get a pointer to the task config
    Task_ptr = Task_ConfigGet();
}
```

Scheduler now knows where Tasks[] is!

Cooperative Scheduler Implementation

Task Configuration

Need to get the pointer to the Tasks table to the scheduler

task_config.c

```
Task_t *Task_ConfigGet(void)
{
    return Tasks;
}
```

How does the scheduler know how many tasks are in the table?

task_config.c

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

Cooperative Scheduler Implementation

OS_Run

File scope variables used by the scheduler

```
static uint8_t  TaskIndex = 0;              // Task index
static uint32_t ElapsedInterval = 0;        // Time since last ran
```

```
void OS_Run(const uint32_t Tick)
{
    // Number of tasks
    const uint8_t NumberOfTasks = Task_GetNumberOf();

    /* Main Loop */
    ...
}
```

Hardware Independent!

Cooperative Scheduler Implementation

OS_Run

```
/* Main Loop */

for(TaskIndex = 0; TaskIndex < NumberOfTasks; TaskIndex++)
{
   assert(TaskIndex < NumTasks);

   if((Tick - TaskPointer[TaskIndex].LastTick) >= TaskPointer[TaskIndex].Interval)
   {
      assert(TaskPointer[TaskIndex].Func != NULL;

      (*TaskPointer[TaskIndex].Function)();          // Execute Task
      Task_Pointer[TaskIndex].LastTick = Tick;      // Save tick the task was ran at.
   }
}// end for
```

Audience POLL Question

What is the main mechanism used to execute a task?

a) A function
b) A function pointer
c) A function pointer table
d) None of the above

Next Steps

04

Going Further

https://beningo.mykajabi.com/

Download Function Pointer Example Code:

- Additional Video on Baremetal Scheduling
- Scheduling Lab Instructions
- Scheduler Source Code

Additional Resources

Please consider the resources below:

- Jacob's Blogs
- Jacob's CEC courses
- Embedded Software Academy

- Embedded Bytes Newsletter
  - http://bit.ly/1BAHYXm

Consulting | Coaching | Training

www.beningo.com

Next Steps

- ✓ Mastering Function Pointers for Baremetal Systems
- ✓ Building Cooperative Schedulers and Command Parsers
- Baremetal Performance Analysis and Architecture Design
- Assertions and Validation Techniques for Baremetal
- Managing Interrupts in Baremetal Systems

CEC Continuing Education Center

DesignNews

Thank You

Sponsored by

DigiKey

BENINGO EMBEDDED GROUP

informa markets