



**Novos<sup>TM</sup>**

Environments for Embedded Systems

# Operating Systems for Embedded Applications

**SAMPLE**

Matching the OS to the Application

by

Tom Barrett  
Embedded Environments Co.  
2018

Embedded Environments Co.  
[www.novos.us](http://www.novos.us)

## **Copyrights**

Ownership of and all copyrights to this document and the Novos™ software environment as it relates to this document are held and/or reserved by Embedded Environments Co. (the “Company”). This document is licensed to you as the sole user. No part of this document may be reproduced, photocopied, stored on a retrieval system, redistributed or transmitted without the express written consent of the Company.

The Company further reserves the right to revise this publication at any time and to make changes to its content and the software it represents and without obligation to inform you of such revisions or changes.

If you are not already familiar with its terms and conditions of use, please read the complete text of the Novos Documentation License, which is available at the Company website, [www.ee-novos.com](http://www.ee-novos.com).

## **Disclaimers**

THIS DOCUMENTATION IS PROVIDED "AS-IS" AND WITHOUT ANY WARRANTY, INCLUDING BUT NOT LIMITED TO ANY EXPRESSED OR IMPLIED WARRANTY OF NON-INFRINGEMENT, SATISFACTORY QUALITY, MERCHANTABILITY OR FITNESS FOR A PARTICULAR USE. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY WHATSOEVER, INCLUDING DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES RESULTING FROM LOSS OF USE, DATA OR PROFITS OR BUSINESS INTERRUPTION, WHETHER IN AN ACTION OF CONTRACT, STRICT LIABILITY OR TORT, INCLUDING NEGLIGENCE, ARISING FROM OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS DOCUMENTATION.

## **Trademarks**

Novos and the following derivatives are trademarks of Embedded Environments Co:

- Novos,
- Novos FB, Novos Foreground/Background,
- Novos EFB, Novos Extended Foreground/Background,
- Novos FCFS, Novos First Come, First Served Scheduling,
- Novos RRS, Novos Round Robin Scheduling,
- Novos PPS, Novos Pre-emptive Priority Scheduling

Other product and company names mentioned in this document may be the trademarks or registered trademarks of their respective owners.

Version #	Date	Author	Revised Sections
1.0	23-May-2018	ATB	Initial release

*This document is licensed only to the individual named in the footer below.  
Duplication in any form strictly prohibited - violators will be prosecuted.  
All rights reserved - Copyright Embedded Environments Co.*

# Contents

Contents.....	4
Figures .....	10
Tables .....	11
Examples .....	12
<b>PART 1.....</b>	<b>13</b>
<b>Architecture benefits, Application Tasks and Real-Time Needs .....</b>	<b>13</b>
Chapter 1.....	14
Introduction.....	14
Scope.....	14
Benefits of a System Architecture.....	15
Chapter 2.....	16
Definitions and Terms .....	16
Definitions .....	16
Tasks .....	18
Properties of a Task.....	19
Types of Tasks .....	20
Periodic Tasks.....	20
Aperiodic Tasks.....	21
Sporadic Tasks .....	21
Real-Time .....	21
Hard Real-time .....	22
Soft Real-time .....	22
Firm Real-time.....	22
Architectural Realities .....	22
<b>PART 2.....</b>	<b>23</b>
<b>Overview of Primary Embedded System Architectures .....</b>	<b>23</b>
Chapter 3.....	24
System Architectures .....	24
State Machines .....	24
Super Loop Systems .....	25
Simple Schedulers.....	25
Multitasking Systems .....	26

Multitasking Basics.....	26
Monolithic Kernels .....	28
Micro-Kernels .....	29
Components of a Multitasking OS.....	30
Scheduler .....	30
OS Services.....	30
Object Classes .....	30
Memory Management/Protection .....	31
Idle Task.....	31
Multitasking Benefits .....	32
Predictability .....	32
System Functionality .....	32
Micro-kernel Multitasking Models.....	33
<b>PART 3.....</b>	<b>34</b>
<b>Super Loop in more detail .....</b>	<b>34</b>
Chapter 4.....	35
Super Loop Systems .....	35
Power-Controlled Super Loops.....	36
Prioritized Super Loop .....	37
Super Loop Summary.....	38
<b>PART 4.....</b>	<b>39</b>
<b>Simple Schedulers in more detail.....</b>	<b>39</b>
Chapter 5.....	40
Simple Schedulers .....	40
Introduction .....	40
Task Model.....	40
One Stack .....	40
Advantages.....	41
Disadvantages .....	42
Variations.....	42
Chapter 6.....	43
Event-driven, Non-Preemptive Scheduler (ENPS).....	43
Ready Queue.....	43
Operation .....	44
ENPS Summary.....	45

Chapter 7 .....	46
Static, Clock-driven Cyclic Scheduler (SCCS) .....	46
Scheduling Table .....	46
Operation .....	48
SCCS Summary .....	49
Chapter 8 .....	50
Fixed Priority Pre-emptive Scheduler (FPPS) .....	50
The Priority Queue .....	50
Task Priorities .....	50
FPPS with Sporadic or Aperiodic Tasks .....	51
FPPS with Periodic Tasks .....	52
FPPS Summary .....	54
<b>PART 5 .....</b>	<b>55</b>
<b>Multitasking Architectures in more detail .....</b>	<b>55</b>
Chapter 9 .....	56
Introduction to Multitasking .....	56
Task Model .....	56
Task Stack .....	56
Task Priority .....	56
Idle Task .....	57
Multiple Stacks .....	57
Variations .....	58
Non-Preemptive Multitasking .....	58
Advantages .....	59
Disadvantages .....	59
Convoy Effect .....	59
Pre-emptive Multitasking .....	59
Advantages .....	60
Disadvantages .....	60
Chapter 10 .....	62
First Come, First Served Scheduling (FCFS) .....	62
FCFS Operation .....	62
Ready Queue Operations .....	63
FCFS Caveats .....	63
FCFS Benefits .....	65
Chapter 11 .....	66
Non-Preemptive Priority Scheduling (NPPS) .....	66

NPPS Operation .....	66
Ready Queue Operations .....	66
Ready Queue with One Task and the Idle Task .....	67
Tasks with Same Priority .....	67
Scheduling Points .....	67
NPPS Caveats .....	67
NPPS Benefits .....	68
Chapter 12 .....	70
Round Robin Scheduling (RRS) .....	70
Tick Quantum .....	70
RRS Operation .....	70
Tick Quantum Expiration .....	71
Unexpired Tick Quantum .....	71
Voluntarily Yielding .....	71
Pausing Execution or Termination .....	71
RRS Caveats .....	72
RRS Benefits .....	72
Chapter 13 .....	73
Pre-emptive Priority Scheduling (PPS) .....	73
The Scheduler .....	73
Priority .....	74
Priority Ordering .....	74
Priority Assignment .....	74
Idle Task Priority .....	74
Task Switching .....	75
Ready Queue .....	75
Simple Organization .....	76
Linked List Organization .....	76
Priority Queue Organization .....	77
One Task per Slot .....	78
Multiple Tasks per Slot .....	78
Priority Queue Caveats .....	79
PPS Operation .....	79
PPS Benefits .....	81
Combinations .....	81
<b>PART 6 .....</b>	<b>82</b>
<b>Overview of Data Objects and System Services .....</b>	<b>82</b>



Chapter 14 .....	83
Data Objects .....	83
Object Creation .....	83
Object Handle .....	83
Object Types .....	84
Static Objects .....	84
Dynamic Objects .....	84
Object Properties .....	84
Object Classes .....	85
Chapter 15 .....	86
System Services .....	86
System Service Types .....	86
System Service Invocation .....	86
Caveats .....	87
<b>PART 7 .....</b>	<b>88</b>
<b>Handling Interrupts, Priority Inversions and Task Schedulability .....</b>	<b>88</b>
Chapter 16 .....	89
Interrupt Handling .....	89
Synchronous Interrupt Requests .....	89
Asynchronous Interrupt Requests .....	89
Interrupt Request Handling .....	89
ISR Entry and Exit .....	90
Switching Stacks in an ISR .....	90
Nested Interrupts .....	90
General Interrupt Handling Models .....	90
Unified Processing .....	91
Split Level Processing .....	91
Chapter 17 .....	93
Priority Inversions .....	93
Description of a Priority Inversion .....	93
Bounded Priority Inversion .....	93
Unbounded Priority Inversion .....	94
Exclusive Access .....	95
Binary Semaphore .....	95
Mutex .....	96
Usage Protocol .....	96



Nested Acquisitions .....	96
Deadlocks .....	97
Priority Inversion Handling .....	97
Priority Ceiling Protocol .....	98
Priority Inheritance Protocol .....	98
Chapter 18 .....	99
Schedulability of Tasks .....	99
Basics .....	99
Rate and Deadline Monotonic Analysis .....	99
Example .....	100
Calculations .....	100
Conclusions and Assumptions .....	101
Interference, etc. ....	102
<b>PART 8 .....</b>	<b>103</b>
<b>Appendices .....</b>	<b>103</b>
Appendix A .....	104
References .....	104

This document is licensed only to the individual named in the footer below.  
 Duplication in any form strictly prohibited - violators will be prosecuted.  
 All rights reserved - Copyright Embedded Environments Co.

# Figures

Figure 1: Characteristics of a Task .....	20
Figure 2: Typical CPU Design.....	27
Figure 3: Task as a Virtual CPU .....	27
Figure 4: OS with Monolithic Kernel.....	29
Figure 5: Multitasking OS with Micro-Kernel.....	30
Figure 6: Timeline for Task with Interrupts.....	41
Figure 7: Stack Content Changes per Figure 6 .....	41
Figure 8: Scheduling Periods within the Major Cycle.....	47
Figure 9: Example of Major Cycle Schedule Sequence.....	47
Figure 10: Event-Driven Scheduling of Sporadic Tasks .....	52
Figure 11: Scheduling of Periodic Tasks .....	53
Figure 12: Operation of Ready Queue in FCFS Scheduling (not to scale) .....	63
Figure 13: Ready Queue Results without Task blocking (not to scale) .....	65
Figure 14: Ready Queue as a Doubly Linked List .....	77
Figure 15: Priority Queue with One Task per Priority .....	78
Figure 16: Priority Queue with Multiple Tasks per Priority .....	79
Figure 17: Simple Task Preemption.....	80
Figure 18: Deferred Access to CPU.....	81
Figure 19: Bounded Priority Inversion.....	94
Figure 20: Unbounded Priority Inversion .....	95
Figure 21: Deadlock with Two Tasks and Two Mutexes.....	97
Figure 22: RM Scheduling of Tasks in Table 7 .....	102

This document is licensed only to the individual named in the header above.  
 Duplication in any form strictly prohibited - violation will be prosecuted.  
 All rights reserved - Copyright Embedded Environments Co.

# Tables

Table 1: Four Fixed Priority Tasks for Cyclic Scheduling .....	46
Table 2: Scheduling Table for Figure 9.....	48
Table 3: Periodic Task Set.....	52
Table 4: Periodic Task Set with Priorities .....	52
Table 5: Object Classes and Their Uses .....	85
Table 6: Task Set for RMA Scheduling.....	100
Table 7: Task Set for RMA Scheduling with Priorities .....	100

*This document is licensed only to the individual named in the footer below.  
Duplication in any form strictly prohibited - violators will be prosecuted.  
All rights reserved - Copyright Embedded Environments Co.*

# Examples

Example 1: Basic Super Loop Code Structure .....	35
Example 2: Power-Controlled Super Loop Code Structure Version I .....	36
Example 3: Power-Controlled Super Loop Code Structure Version II .....	37
Example 4: Prioritized Super Loop Code Structure .....	38
Example 5: Static, Fixed Priority Non-Preemptive Scheduler .....	44
Example 6: Yielding CPU Control .....	64
Example 7: Two ways to give up CPU Control .....	68

*This document is licensed only to the individual named in the footer below.  
Duplication in any form strictly prohibited - violators will be prosecuted.  
All rights reserved - Copyright Embedded Environments Co.*

# PART 1

## Architecture benefits, Application Tasks and Real-Time Needs

This document is licensed only to the individual named in the footer below.  
Duplication in any form strictly prohibited. Violators will be prosecuted.  
All rights reserved - Copyright Embedded Environments Co.

# Chapter 1

## Introduction

There has always been a variety of opinions amongst embedded system developers about the best system architecture for their applications. The subject has endured for over forty years and continues to this day. The debate tipped in favor of using a defined software architecture primarily because of the growth in the complexity of embedded applications.

Developing a complex application that incorporates a communications protocol, file-based storage, graphical output along with data acquisition and control is virtually impossible to do without some organizational structure on which to build the application code. But even in small systems, a good argument can be made for an organized structure for application development.

Those for whom a system architecture is not a big consideration tend to be developers of small or very simple applications. There are still a lot of these applications around due to the steady decline in the prices of microcontrollers and the scale of peripheral devices integrated into them. With the Internet of Things (IoT) starting to burgeon, there is an increasing likelihood of growth in that segment of the embedded systems world.

The subject of system architectures in general is large and literature about them is extensive, far beyond the space available here. Many system architecture implementations exist for a broad spectrum of applications but this document will confine the discussion to an overview of those system architectures primarily used in embedded systems.

Any products directly mentioned in this document are the trademarks of their developers.

## Scope

In the world of embedded systems, there are many system architectures, or, environments, that provide a foundation on which the application developer designs and implements the software that becomes the application. As befits the applications themselves, some system architectures are better suited to a given type of application than others. For example, the cruise control on an automobile has tight timing needs whereas a smart thermostat does not. They are both considered embedded applications but their architectural needs differ greatly.

The intent of this document is to provide a top-level, generic overview of system architectures commonly used in embedded systems. Some of them have real-time capabilities and some do not. Nevertheless, they are useful in a broad range of embedded applications where their capabilities match the needs of the application.

The information provided in this document is general and does not pertain explicitly

to any of the Novos family of operating system environments for embedded systems. If the reader wants to delve more deeply into the subject, the source code for the Novos operating system environments is freely available at [www.ee-novos.com](http://www.ee-novos.com) and can serve as a sound basis for further exploration.

## Benefits of a System Architecture

Whether large and complex or small and simple, a well-designed and documented system architecture provides a number of tangible and/or financial benefits to the developer. It

- simplifies application development and improves developer productivity by utilizing high level abstractions (i.e., objects) to ease the handling of complex functions,
- provides a solid development infrastructure that provides consistency and repeatability,
- abstracts away the complexities of the processor, making it easier to migrate the application to another processor to meet pricing or competitive requirements,
- optimizes use of system resources and improves product reliability, maintainability and quality of the application, and,
- manages resources and serves as the point of integration for middleware.

A system architecture provides a way to bring all those elements together into a platform that allows the application developer to begin development at a much higher point, which enables a shorter time-to-market with higher reliability and lower risk. Most importantly, valuable engineering time is directed towards domain-specific endeavors that advance the system development rather than being spent trying to create an architecture for the application. In the final analysis, it is the domain-specific work that yields the financial return for the application developer, not the work on developing an ad hoc system architecture.

