

---

Peter Gliwa

# Embedded Software Timing

Methodology, Analysis and Practical  
Tips with a Focus on Automotive

Peter Gliwa  
Gliwa GmbH  
Weilheim, Germany

ISBN 978-3-030-64143-6      ISBN 978-3-030-64144-3 (eBook)  
<https://doi.org/10.1007/978-3-030-64144-3>

Translated and Extended from the German Edition P. Gliwa “Embedded Software Timing” © 2020  
Springer Fachmedien Wiesbaden GmbH

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2021

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG.  
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

---

## Preface

Embedded software makes up only a comparatively small part of the larger topic of computer science. Within this, the topic of “timing” focuses only on one specific aspect. So, is the topic of “Embedded Software Timing” one that is only relevant to a few experts?

At this very moment, billions of embedded systems are in use worldwide. Embedded software is running on every one of those devices with each system having its own set of timing requirements. If those timing requirements are not met due to a software error, the range of possible outcomes varies enormously. Depending on the product and situation, this may range from not being noticed, to being an annoyance for the user, to costing lives.

A good understanding for the timing challenges of embedded systems enables the development of better, more reliable embedded software. In addition, it is not only safety and reliability that can be improved. There are also considerable cost savings to be had across the entire development life cycle. These are not purely theoretical, as the practical examples in Chapter 6 highlight. The potential for cost savings extends across the various phases of development:

- Early consideration for the issue of timing in the design of an embedded system and its software helps decisively in increasing development efficiency and prevents timing problems from arising in the first place.

See, among others, Sections 3.3, 6.2, and 8.1 and Chapter 9.

- Timing analysis can save time and money if the correct timing analysis technique for the given application is used. Chapter 5 provides an overview of the different techniques. Each has its own phases that describe its functional principle and workflow, highlighting use cases and limitations. In addition, an interview with one or two experts in the respective domain completes these descriptions. This livens up the topic and provides some objectivity. If the milk has already spilled—that is, if a project is already facing acute problems—troubleshooting often resembles the search for a needle in a haystack, especially in the case of timing problems. Here, too, the use of the optimal timing analysis technique delivers decisive advantages.
- Automated tests help to save costs: this is a truism. Unfortunately, existing testing all too often lacks explicit timing-related tests and focuses only on functional aspects. Section 9.6 provides recommendations in the form of concrete measures

to counteract this by ensuring embedded software timing can be verified in a well-structured manner.

- If a project reaches the point where the processor is permanently or sporadically overloaded, measures must be taken to relieve the load. This often occurs when deadlines are approaching, and, therefore, a task force is formed to relieve the situation. Chapter 8 offers knowledge that can serve as a valuable basis for such task forces. Section 8.4, which rounds off the chapter, can be used as a starting point.

The focus throughout the book is to provide a close proximity to practice. Theory is always illustrated with examples, and there are plenty of concrete tips for design, implementation, debugging, and verification.

The chapters are structured in such a way that they can be read most easily in the given order. Nevertheless, while writing this book, an attempt has been made to give each chapter a certain degree of independence, so that the reader is not lost when it is used to look something up or when reading selectively.

I would be grateful for any suggestions, criticism, and hints regarding mistakes and also welcome contact for direct professional discussion.

With that, I wish you lots of enjoyment and technical insight while reading.

Weilheim, Germany

Peter Gliwa

May 2020

[peter.gliwa@gliwa.com](mailto:peter.gliwa@gliwa.com)

All brand names and trademarks in this book are the property of their rightful owners and are used for description only.

---

## Acknowledgments

Time is the theme and focus of this book. To organize it in such a way that no major timing problems arose when writing over 300 pages, and creating over 100 illustrations, was sometimes a challenge.

I could not have undertaken and mastered this without the active support of many dear people.

First of all, there is Nick (Dr. Nicholas Merriam), from whom I learned a lot, such as an understanding of caches, pipelines, and spinlocks. In general, a lot of my knowledge about multi-core and runtime optimization has Nick as its source and this can now be found in the book. Thanks a lot for that, Nick!

I would also like to thank all the interview partners, not only for the time they took for the interviews but also for our collaborative work in standardization committees—they would be much less fun without you.

A big thanks goes to Stuart Cording ([www.cordingconsulting.com](http://www.cordingconsulting.com)) who not only brought the English in this book to an acceptable level but also found several flaws in its contents. Many thanks!

I would like to thank Birgit Tamkus, Peter Stief, Christian Herget, Mark Russell, and Christian Wenzel-Benner very much for their reviews and suggestions. Many thanks to you all also for supporting me in many aspects of my daily work during my writing-intensive periods.

I would like to thank my publisher *Springer*—especially, Mr. Ralf Gerstner—for their pleasant, uncomplicated, and constructive cooperation.

I still remain today grateful to Hans Sarnowski of BMW for encouraging me, back in 2002, to found a company that, from the very beginning, specialized in embedded software timing. We have also achieved many successes together on the front line of timing problems—and each and every one of them was a lot of fun.

Finally, I would like to express my greatest thanks to my wife, Priscilla, without whom I could not have written the book. Especially in the spring of 2020—and despite the Corona crisis—she freed up time for me, took care of our four children, and made it possible for me to spend many weekends and holidays in peace and quiet at the office. Many thanks for that!

---

# Contents

<b>1</b>	<b>General Basics</b>	1
1.1	Real-Time	1
1.2	Phase Driven Process Model: The V-Model	2
1.3	Build Process: From the Model to the Executable	3
1.4	Summary	11
<b>2</b>	<b>Microprocessor Technology Basics</b>	13
2.1	Microprocessor Design	13
2.2	Code Execution	16
2.3	Memory Addressing and Addressing Modes	18
2.4	Wait States and Burst Accesses	24
2.5	Cache	25
2.6	Pipeline	30
2.7	Interrupts	31
2.8	Traps/Exceptions	32
2.9	Data Consistency	32
2.10	Comparison of Desktop Processors Versus Embedded Processors	34
2.11	Summary	36
<b>3</b>	<b>Operating Systems</b>	37
3.1	No OS: Endless-Loop Plus Interrupts	38
3.2	OSEK/VDX	40
3.3	Cooperative and Preemptive Multitasking	46
3.4	POSIX	54
3.5	Summary	60
<b>4</b>	<b>Timing Theory</b>	61
4.1	Timing Parameters	62
4.2	Statistical Aspects	68
4.3	CPU Load	72
4.4	Bus Load	79
4.5	Logical Execution Time (LET)	79
4.6	Summary	81

<b>5</b>	<b>Timing Analysis Techniques</b>	83
5.1	Overview, Layered View	83
5.2	Definitions of Terms	87
5.3	Static Code Analysis	88
5.4	Code Simulation	97
5.5	Timing Measurement	105
5.6	Hardware-Based Tracing	117
5.7	Instrumentation-Based Tracing	131
5.8	Scheduling Simulation	148
5.9	Static Scheduling Analysis	156
5.10	Optimization Using Evolutionary Algorithms	166
5.11	Timing Analysis Techniques in the V-Model	168
<b>6</b>	<b>Practical Examples of Timing Problems</b>	171
6.1	Where Do All the Interrupts Come From?	171
6.2	OSEK ECC: Rarely the Best Choice	173
6.3	Rare Crashes 17 min After Reset	175
6.4	Missing or Duplicated Sensor Data	178
6.5	In a Race with the Handbrake On	183
6.6	Measurement Delivers WCET Results Greater Than Those from Static Code Analysis	184
6.7	Network Management Messages Appear too Soon	185
6.8	Seamless Timing Process in a Mass-Production Project	187
6.9	Timing Analysis Saves OEM €12 m	187
6.10	Summary	188
<b>7</b>	<b>Multi-Core, Many-Core, and Multi-ECU Timing</b>	189
7.1	Multi-Core Basics	189
7.2	Different Types of Parallel Execution	193
7.3	Data Consistency, Spinlocks	202
7.4	Cloning of Memory Addresses	207
7.5	Summary	210
<b>8</b>	<b>Timing Optimization</b>	213
8.1	Timing Optimization at the Scheduling Level	213
8.2	Timing Optimization of Memory Usage	218
8.3	Timing Optimization at Code Level	223
8.4	Summary and Guidelines for Timing Optimization	243
<b>9</b>	<b>Methodology During the Development Process</b>	247
9.1	Requirements Related to Timing	247
9.2	Collaboration During the Development	256
9.3	Timing Concept, Scheduling Layout, and OS Configuration	257
9.4	Timing Debugging	258
9.5	Timing Optimization	258
9.6	Timing Verification	259
9.7	Early Consideration for Future Functionality	261

---

9.8	Timing Supervision in the Final Product.....	262
9.9	Positive Example: CoReMa by Vitesco Technologies.....	263
9.10	Summary.....	265
<b>10</b>	<b>AUTOSAR</b> .....	267
10.1	AUTOSAR Classical Platform (CP) .....	268
10.2	AUTOSAR Adaptive Platform (AP) .....	271
10.3	TIMEX (AUTOSAR Timing Extensions).....	280
10.4	ARTI (AUTOSAR/ASAM Run-Time Interface) .....	282
10.5	Technical Report “Timing Analysis”.....	286
10.6	Summary.....	286
<b>11</b>	<b>Safety and ISO 26262</b> .....	289
11.1	Basics .....	289
11.2	Safety Standards, Timing, and Timing Verification .....	293
11.3	Tools for Timing Verification .....	294
11.4	Legal Aspects.....	295
11.5	Summary.....	295
<b>12</b>	<b>Outlook</b> .....	297
	<b>References</b> .....	299
	<b>Index</b> .....	301