

The Art of
**PROGRAMMING
EMBEDDED
SYSTEMS**

JACK G. GANSSE

The Art of Programming Embedded Systems

This page intentionally left blank

THE ART OF PROGRAMMING EMBEDDED SYSTEMS

Jack G. Ganssle

Softaid, Inc.

Columbia, Maryland



ACADEMIC PRESS, INC.

Harcourt Brace Jovanovich, Publishers

San Diego New York Boston

London Sydney Tokyo Toronto

This book is printed on acid-free paper. (∞)

Copyright © 1992 by ACADEMIC PRESS, INC.

All Rights Reserved.

No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy, recording, or any information storage and retrieval system, without permission in writing from the publisher.

Academic Press, Inc.

San Diego, California 92101

United Kingdom Edition published by

Academic Press Limited

24-28 Oval Road, London NW1 7DX

Library of Congress Cataloging-in-Publication Data

Ganssle, Jack G.

The art of programming embedded systems / Jack G. Ganssle.

p. cm.

Includes bibliographical references and index.

ISBN 0-12-274880-8

1. Embedded computer systems--Programming. I. Title.

QA76.6.G334 1991

005.1--dc20

91-24153

CIP

PRINTED IN THE UNITED STATES OF AMERICA

91 92 93 94 9 8 7 6 5 4 3 2 1

**This book is dedicated to
Cathy, Graham, and Kristina**

This page intentionally left blank

Contents

Preface **xi**

Acknowledgments **xiii**

1	Introduction	1	
	On the Algorithm Collection		4
	Corporate Knowledge	7	
	Basic Assumptions	8	
2	Initial Considerations	9	
	Business Issues	9	
	Picking the Processor	11	
	Estimating Memory Requirements		17
	Selecting I/O Devices	20	
	Languages: The HLL Dilemma		22
	Transitioning to ROM	29	
3	Elegant Structures	36	
	Designs with One CPU	39	
	State Machines	40	
	Distributed Processing	45	
	Watchdogs	51	
	The Software Engineering Methodology		53

4	Design for Debugging	63
	Introduction	63
	Debugging Tools	66
	Adding Debugging Code	86
	Common Debugging Problems	98
	Common Sense	100
5	Design for Test	101
	Internal Diagnostics	102
	External Diagnostics	118
6	Memory Management	121
	How Memory Mappers Work	123
	Memory Management Units	126
	Practical Memory Management	134
	Tips and Techniques	139
	Language Support	140
7	Approximations	151
	Errors	152
	Square Roots	153
	Higher Order Roots	159
	Logarithms	160
	Exponentials	163
	Cosine, Sine, and Tangent	166
	Inverse Trig Functions	170
8	Interrupt Management	174
	Interrupts	174
	Interrupt Service	178
	Nonmaskable Interrupts	180
	Queue Handling	183
	Problem Areas	186
9	Real-Time Operating Systems	194
	Tasking and Scheduling	195
	Using an RTOS	201
	Commercial Operating Systems	202
	A Poor Man's RTOS	203

10	Signal Sampling and Smoothing	223
	Averaging	225
	Convolutions	228
	Differentiation	233
	Linear Calibrations	234
	Nonlinear Calibrations	237
	Standards	244
	Conclusion	246
11	A Final Perspective	247
	Schedule Panics	247
	Make Yourself More Valuable	250
	The Future of Embedded Systems	252
	Appendix A: Magazines	255
	Appendix B: File Format	257
	Intel Hex Format	257
	Motorola S-Records	259
	Appendix C: Serial Communications	261
	ASCII	261
	RS-232 Data Transmission	262
	Bit Banging	264
	Autobauding	267
	Bibliography	271
	Index	275

This page intentionally left blank

Preface

DeMarco and Lister, in their wonderful book *Peopleware*, complain that too many software people just don't read technical publications. What a frightening thought! No other industry is so characterized by constant and profound change. Somehow, we embedded programmers must access every scrap of information in the fight to stay up to date.

In the embedded world it seems most of us learn via on-the-job training. While many colleges offer embedded programming courses, few go beyond descriptions of the sort of simple projects even a high-school hacker might build.

With embedded systems ranging in size from a few hundred lines for a trivial controller to multi-million-line tracking systems, it's impossible to cover all of the issues encountered in designing embedded systems in one or even a dozen volumes. In this book I've tried to address the subject from three different approaches: design, solutions to practical problems, and planning.

While we're all familiar with the tenets of top-down design, far too little has been written on the subject of designing code that is debuggable and useful in the production environment that our embedded code is targeted toward. Chapters 4 and 5 deal with these subjects.

Much of the book is taken up with algorithms and techniques for solving a number of problems common to many embedded systems. Chapters 6 through 10 address these system problems, with example solutions to each.

Finally, perhaps an underlying theme throughout the book is the concept of the software engineer as a manager. "Coders" worry about writing software, period. I believe that true software engineers should

be concerned with every aspect of the system, from its high-level design to the low-level code, even to business issues that affect the end-product's marketability. What will adding a floating-point package do to the cost of goods? How does time-to-market impact the product's viability? When we select a processor, what business issues are important, perhaps even as important as the technical trade-offs? Although Chapters 1, 2, and 11 specifically deal with these questions, this also pervades most of the rest of the book.

Acknowledgments

Academic Press came up with the idea and inspiration for the book. Randy Gilleland read the manuscript quite a few times, in all its various forms, and contributed code, ideas, and enthusiasm. Scott Rosenthal made numerous suggestions about the book's technical content.

The stories and experiences that make up so much of this book all stem from work I've done with hundreds of others over the years. I can't acknowledge the individuals, but thanks to y'all.

Thanks to my three-year-old son Graham for not destroying the temptingly huge stacks of files associated with this project.

Finally, special thanks to my wife Cathy for taking care of the kids while I worked on this book and for encouraging and supporting the project through eighteen months of effort.

This page intentionally left blank