Developing Reusable Firmware

A Practical Approach to APIs, HALs and Drivers

Jacob Beningo

BENINGO EMBEDDED GROUP

Contents

Acknowledgements xiii

Foreword xv

Preface xvii

Introduction 1

Chapter 1 Concepts for Developing Portable Firmware 3

- Why Code Reuse Matters 3
- Portable Firmware 5
- Modularity 11
- Module Coupling and Cohesion 12
- Following A Standard 14
- Portability Issues in C – Data Types 15
- Portability Issues in C – Structures and Unions 17
- Portability Issues in C – Bit Fields 18
- Portability Issues in C – Preprocessor Directives 19
- Hardware Abstraction Layers (HAL) 21
- Application Programming Interfaces (APIs) 28
- Project Organization 28
- Getting Started Writing Portable Firmware 29
- Going Further 32

Chapter 2 API and HAL Fundamentals 37

- The Wonderful World of HALs 37
- APIs versus HALs 38
- The API and HAL Landscape 40
- The Good, Bad and Ugly 42
- Potential Issues and the Boogeyman 42
- Characteristics Every HAL should Exhibit 46
- To Build or Not to Build 56
- A First Look at a HAL 58
- The API Scope 59
- API Characteristics to look for 60
- Designing your own APIs 65
- A First Look at an API 65
- Going Further 67

Chapter 3 Device Driver Fundamentals in C....73
Understanding the Memory Map....73
Identifying Driver Interfaces....77
Design by Contract....80
Assertion Fundamentals....83
Device Driver Models....85
Polled versus Interrupt Driven Drivers....87
Deciphering the volatile, const and static keywords....93
Driver Module Definition....93
Naming Convention Recommendations....95
Object Oriented Programming in C....96
Encapsulation and Data Hiding....105
Abstract Data Types....105
Callback Functions....105
Error Handling....110
Expected Results and Recommendations....113
Leverage Design Patterns....114
Going Further....114

Chapter 4 Writing Reusable Drivers....119
It's All about the Interface....119
How much Abstraction is too much?....120
Wrapping the Interface....120
Memory Mapping Methodologies....128
Mapping Memory Directly....129
Mapping Memory with Pointers....130
Mapping Memory with Structures....133
Using Pointer Arrays in Driver Design....136
Creating a Timer Driver Overview....138
Step #1 – Define the Timers Configuration Table....138
Step #2 – Define the Timers Peripheral Channels....140
Step #3 – Populate the Timers Configuration Table....141
Step #4 – Create the Timer Pointer Arrays....143
Step #5 – Create the Initialization Function....145
Step #6 – Fill in the Timer Driver Interface....150
Step #7 – Maintain and Port the Design Pattern....150
Selecting the Right Driver Implementation....151
Going Further....152

Chapter 5 Documenting Firmware with Doxygen....157
The Importance of Good Documentation....157
Easing the Documentation Load....159
An Introduction to Doxygen....162

Confidential – ©2016 Jacob Beningo, All Rights Reserved, DRAFT A08

Installing Doxygen **164**
Documentation Project Setup **166**
Doxygen Comment Fundamentals **173**
Documenting enum and struct **174**
Documenting Functions **175**
Documenting Modules **181**
Creating a Reusable Template **183**
Generating a Main Page **184**
10 Tips for Commenting C Code **187**
Tip #1- Explain the Why not the How 188
Tip #2 – Comment Before Coding 189
Tip #3 – Use Doxygen Tags 189
Tip #4 – Adopt a Code Style Guide 190
Tip #5 – Use a File Header 190
Tip #6 – Create a Commenting Template 191
Tip #7 – Consistent Comment Location 191
Tip #8 – Don't Comment Every Line 192
Tip #9 – Start Mathematical type Identifiers with the Type 193
Tip #10 – Update Comments with Code Updates 193
A Few Final Thoughts on Documentation **194**
Going Further **194**

Chapter 6 The Hardware Abstraction Layer Design Process 197
Why use a HAL? **197**
Characteristics of a Good HAL **199**
The HAL Design Process **200**
Step #1 – Review the Microcontroller Peripheral Datasheet **202**
Step #2 – Identify Peripheral Features **203**
Step #3 – Design and Create the Interface **204**
Step #4 – Create Stubs and Documentation Templates **206**
Step #5 – Implement for Target Processor(s) **208**
Step #6 – Test, Test, Test **210**
Step #7 – Repeat for the Next Peripheral **212**
10 Tips for Designing a HAL **214**
Tip #1- Identify Core Features 214
Tip #2 – Avoid an All Encompassing HAL 215
Tip #3 – Add Register Access Hooks 216
Tip #4 – Use Doxygen to Outline the HAL 217
Tip #5 – Get a Second set of Eyes 218
Tip #6 – Create a Commenting Template 219
Tip #7 – Keep the View at 30,000 Feet 220
Tip #8 – Use Appropriate Naming Conventions 221
Tip #9 – Include a Parameter for Initialization 222

Confidential – ©2016 Jacob Beningo, All Rights Reserved, DRAFT A08

Tip #10 – Deploy on Multiple Development Kits 223
Going Further 224

Chapter 7 HAL Design for GPIO 227
An Overview of GPIO Peripherals 227
Step #1 – Review the GPIO Peripheral Datasheet 228
Step #2 – GPIO Peripheral Features 230
Step #3 – Design and Create the GPIO HAL Interface 232
Step #4 – Create GPIO Stubs and Documentation Templates 235
Step #5 – Implement GPIO HAL for Target Processor 237
Step #6 – Test, Test, Test 239
Step #7 – Repeat for the Next Peripheral 241
Going Further 242

Chapter 8 HAL Design for SPI 245
An Overview of SPI Peripherals 245
Step #1 – Review the SPI Peripheral Datasheet 246
Step #2 – SPI Peripheral Features 249
Step #3 – Design and Create the SPI HAL Interface 251
Step #4 – Create SPI Stubs and Documentation Templates 254
Step #5 – Implement SPI HAL for Target Processor 256
Step #6 – Test, Test, Test 257
Step #7 – Repeat for the Next Peripheral 258
Going Further 260

Chapter 9 HAL Design for EEPROM and Memory Devices 263
An Overview of Memory Devices 263
Step #1 – Review the EEPROM Peripheral Datasheet 266
Step #2 – EEPROM Peripheral Features 269
Step #3 – Design and Create the EEPROM HAL Interface 272
Step #4 – Create EEPROM Stubs and Documentation Templates 275
Step #5 – Implement EEPROM HAL for Target Processor 277
Step #6 – Test, Test, Test 278
Step #7 – Repeat for the Next Peripheral 279
Going Further 280

Chapter 10 API Design for Embedded Applications 283
Applications Made Easier 283
Software Components 285
Application Frameworks 289
Libraries 293
The Real-time Operating System (RTOS) 296
Open Source Software 298
Integration Nightmares 300

Confidential – ©2016 Jacob Beningo, All Rights Reserved, DRAFT A08

Common Software Frameworks – System Configuration 303
Common Software Frameworks – Console Applications..............306
Common Software Frameworks – Bootloaders 309
Common Software Frameworks – Thread Tracing..........................312
Common Software Frameworks – FAT File System.........................315
Going Further 320

Chapter 11 Testing Portable Embedded Software 323
Cross Your Fingers and Pray...323
Standard Interface .... Standard Tests 326
Test Driven Development 328
Taking Advantage of Cyclomatic Complexity for Testing 334
Automating Tests 338
Python for Testing 342
Developing GPIO Test Cases...348
Developing SPI Test Cases 356
Test Cases for EEPROM 360
Using Trace to Verify Application Software 364
Going Further 366

Chapter 12 A Practical Approach to Code Reuse.....................369
Being Practical in an Unpractical Environment 369
Phases and Baby Steps 372
Getting the Most from Metrics..374
Metrics Worth Tracking 376
Recognizing Design Patterns 378
Creating Templates and Checklists 380
Version Control is your Best Friend..383
Results – Decreasing Time-to-Market...385
Results – Decreasing Development Costs 387
Results –Increased Quality 389
What is the Cost to do Nothing? 391
Going Further 394

Appendix A..398

Appendix B..415

Appendix C..430

References 445

Glossary 450

Confidential – ©2016 Jacob Beningo, All Rights Reserved, DRAFT A08