

Table of Contents

| | |
|--|-----------|
| CHAPTER 1. SEE A PROGRAM RUNNING | 1 |
| 1.1 TRANSLATING C TO MACHINE CODE..... | 1 |
| 1.2 LOADING A MACHINE PROGRAM INTO MEMORY..... | 4 |
| 1.2.1 <i>Harvard versus Von Neumann Architecture</i> | 4 |
| 1.2.2 <i>Creating Runtime Memory Image</i> | 6 |
| 1.3 REGISTERS..... | 10 |
| 1.3.1 <i>Reusing Registers to Improve Performance</i> | 10 |
| 1.3.2 <i>Processor Registers</i> | 12 |
| 1.4 EXECUTING A MACHINE PROGRAM..... | 15 |
| 1.4.1 <i>Loading a Program</i> | 16 |
| 1.4.2 <i>Starting the Execution</i> | 17 |
| 1.4.3 <i>Program Completion</i> | 20 |
| 1.5 SELF-REVIEW EXERCISES | 21 |
| 1.6 EXERCISES..... | 24 |
| CHAPTER 2. DATA REPRESENTATION | 25 |
| 2.1 BIT, BYTE, AND WORD..... | 25 |
| 2.2 BINARY, OCTAL, DECIMAL, AND HEXADECIMAL..... | 27 |
| 2.3 UNSIGNED INTEGERS..... | 28 |
| 2.4 SIGNED INTEGERS..... | 29 |
| 2.4.1 <i>Sign-and-Magnitude</i> | 31 |
| 2.4.2 <i>One's Complement</i> | 32 |
| 2.4.3 <i>Two's Complement</i> | 33 |
| 2.4.4 <i>Carry Flag for Unsigned Addition and Subtraction</i> | 35 |
| 2.4.5 <i>Overflow Flag for Signed Addition and Subtraction</i> | 37 |
| 2.5 CHARACTER STRING..... | 46 |
| 2.6 SELF-REVIEW EXERCISES | 48 |
| 2.7 EXERCISES..... | 49 |
| CHAPTER 3. ARM INSTRUCTION SET ARCHITECTURE | 51 |
| 3.1 INSTRUCTION SET ARCHITECTURE (ISA)..... | 51 |
| 3.2 ARM INSTRUCTION SET ARCHITECTURE | 52 |
| 3.3 ARM CORTEX-M ORGANIZATION | 55 |
| 3.4 GOING FROM C TO ASSEMBLY | 57 |
| 3.5 ASSEMBLY INSTRUCTION FORMAT..... | 60 |
| 3.6 ANATOMY OF AN ASSEMBLY PROGRAM..... | 62 |
| 3.7 ASSEMBLER DIRECTIVES | 65 |
| 3.8 SELF-REVIEW EXERCISES | 69 |
| 3.9 EXERCISES..... | 71 |
| CHAPTER 4. ARITHMETIC AND LOGIC INSTRUCTIONS | 73 |
| 4.1 PROGRAM STATUS REGISTER..... | 73 |
| 4.2 UPDATING PROGRAM STATUS FLAGS..... | 75 |

| | | |
|---|---|------------|
| 4.3 | SHIFT AND ROTATE | 76 |
| 4.4 | ARITHMETIC INSTRUCTIONS..... | 78 |
| 4.4.1 | <i>Addition and Subtraction</i> | 79 |
| 4.4.2 | <i>Short Multiplication and Division</i> | 80 |
| 4.4.3 | <i>Long Multiplication</i> | 80 |
| 4.4.4 | <i>Saturation</i> | 81 |
| 4.5 | IMPLEMENTING A BARREL SHIFTER..... | 82 |
| 4.6 | BITWISE LOGIC OPERATIONS..... | 83 |
| 4.7 | REVERSING THE ORDER OF BITS AND BYTES..... | 87 |
| 4.8 | SIGN AND ZERO EXTENSION | 89 |
| 4.9 | TEST AND COMPARISON | 90 |
| 4.10 | DATA MOVEMENT BETWEEN REGISTERS | 91 |
| 4.11 | BIT FIELD EXTRACT..... | 92 |
| 4.12 | SELF-REVIEW EXERCISES..... | 92 |
| 4.13 | EXERCISES | 95 |
| CHAPTER 5. LOAD, STORE, AND STACK | | 97 |
| 5.1 | LOADING CONSTANT INTO REGISTERS..... | 97 |
| 5.1.1 | <i>Data Movement Instruction MOV and MVN</i> | 97 |
| 5.1.2 | <i>Pseudo Instruction LDR and ADR</i> | 98 |
| 5.1.3 | <i>Comparing LDR, ADR, and MOV</i> | 99 |
| 5.2 | BIG AND LITTLE ENDIAN | 99 |
| 5.3 | ACCESSING DATA IN MEMORY..... | 100 |
| 5.4 | MEMORY ADDRESSING..... | 101 |
| 5.4.1 | <i>Pre-index, Post-index, and Pre-index with Update</i> | 101 |
| 5.4.2 | <i>Load and Store Instructions</i> | 103 |
| 5.4.3 | <i>PC-relative Addressing</i> | 103 |
| 5.4.4 | <i>Example of Accessing an Array</i> | 104 |
| 5.5 | LOADING AND STORING MULTIPLE REGISTERS..... | 105 |
| 5.6 | ACCESSING STACK MEMORY..... | 107 |
| 5.6.1 | <i>Stack Types</i> | 107 |
| 5.6.2 | <i>PUSH and POP Operations</i> | 108 |
| 5.6.3 | <i>Implementing Stacks with STM and LDM</i> | 109 |
| 5.7 | SELF-REVIEW EXERCISES..... | 111 |
| 5.8 | EXERCISES | 113 |
| CHAPTER 6. BRANCH AND CONDITIONAL EXECUTION..... | | 117 |
| 6.1 | CONDITION TESTING..... | 117 |
| 6.2 | BRANCH INSTRUCTIONS | 119 |
| 6.3 | CONDITIONAL EXECUTION..... | 122 |
| 6.4 | IF STATEMENT | 123 |
| 6.5 | IF-ELSE STATEMENT..... | 126 |
| 6.6 | FOR LOOP | 127 |
| 6.7 | WHILE LOOP..... | 128 |
| 6.8 | DO WHILE LOOP..... | 129 |
| 6.9 | CONTINUE STATEMENT..... | 130 |

| | | |
|---|---|------------|
| 6.10 | BREAK STATEMENT..... | 131 |
| 6.11 | SWITCH STATEMENT | 132 |
| 6.12 | SELF-REVIEW EXERCISES | 134 |
| 6.13 | EXERCISES..... | 137 |
| CHAPTER 7. STRUCTURED PROGRAMMING..... | | 141 |
| 7.1 | BASIC CONTROL STRUCTURES..... | 141 |
| 7.2 | STEPWISE REFINEMENT..... | 143 |
| 7.3 | REGISTER ALLOCATION..... | 146 |
| 7.4 | EXAMPLE: FACTORIAL NUMBERS..... | 149 |
| 7.5 | EXAMPLE: COUNTING ONES IN A WORD..... | 150 |
| 7.6 | EXAMPLE: FINDING THE MAXIMUM VALUE IN AN ARRAY | 152 |
| 7.7 | EXAMPLE: COUNTING DIGITS..... | 154 |
| 7.8 | EXAMPLE: PARITY BIT | 155 |
| 7.9 | EXAMPLE: PERFECT NUMBERS | 157 |
| 7.10 | EXAMPLE: ARMSTRONG NUMBERS | 159 |
| 7.11 | EXAMPLE: PALINDROME STRING | 160 |
| 7.12 | EXAMPLE: CONVERTING STRING TO INTEGER (atoi)..... | 162 |
| 7.13 | EXAMPLE: BINARY SEARCH..... | 163 |
| 7.14 | EXAMPLE: BUBBLE SORT | 165 |
| 7.15 | SELF-REVIEW EXERCISES | 167 |
| 7.16 | EXERCISES..... | 168 |
| CHAPTER 8. SUBROUTINES..... | | 171 |
| 8.1 | CALLING A SUBROUTINE..... | 172 |
| 8.2 | PRESERVING CALLER'S RUNTIME ENVIRONMENT VIA STACK..... | 173 |
| 8.3 | PASSING ARGUMENTS TO SUBROUTINE VIA REGISTERS..... | 176 |
| 8.4 | PASS-BY-VALUE AND PASS-BY-REFERENCE | 177 |
| 8.4.1 | <i>Example of Passing by Value: Sum of an Array</i> | 179 |
| 8.4.2 | <i>Example of Passing by Reference: Swap two characters</i> | 181 |
| 8.5 | CALLING SUBROUTINES IN A DIFFERENT FILE | 182 |
| 8.6 | EXAMPLE: GREATEST COMMON DIVISOR | 183 |
| 8.7 | EXAMPLE: CONCATENATING TWO STRINGS..... | 185 |
| 8.8 | EXAMPLE: COMPARING TWO STRINGS..... | 186 |
| 8.9 | EXAMPLE: INSERTING AN INTEGER INTO A SORTED ARRAY | 187 |
| 8.10 | EXAMPLE: CONVERTING INTEGER TO STRING (itoa) | 188 |
| 8.11 | EXAMPLE: MATRIX TRANSPOSE..... | 190 |
| 8.12 | EXAMPLE: REMOVING A CHARACTER FROM A STRING..... | 192 |
| 8.12.1 | <i>Example: Reversing a String</i> | 193 |
| 8.13 | EXAMPLE: FINDING UNIQUE NUMBERS IN AN ARRAY | 194 |
| 8.14 | PASSING ARGUMENTS THROUGH STACK..... | 198 |
| 8.15 | RECURSIVE FUNCTIONS..... | 200 |
| 8.15.1 | <i>Example: Factorial Numbers</i> | 202 |
| 8.15.2 | <i>Example: String Permutation</i> | 203 |
| 8.15.3 | <i>Tail Recursion</i> | 205 |
| 8.16 | SELF-REVIEW EXERCISES | 207 |

| | | |
|---|---|------------|
| 8.17 | EXERCISES | 210 |
| CHAPTER 9. 64-BIT DATA PROCESSING | | 213 |
| 9.1 | 64-BIT ADDITION | 213 |
| 9.2 | 64-BIT SUBTRACTION | 214 |
| 9.3 | 64-BIT COUNTING LEADING ZEROS..... | 215 |
| 9.4 | 64-BIT SIGN EXTENSION | 215 |
| 9.5 | 64-BIT LOGICAL SHIFT LEFT..... | 216 |
| 9.6 | 64-BIT LOGICAL SHIFT RIGHT..... | 217 |
| 9.7 | 64-BIT MULTIPLICATION | 218 |
| 9.8 | 64-BIT UNSIGNED DIVISION..... | 219 |
| 9.9 | 64-BIT SIGNED DIVISION | 221 |
| 9.10 | SELF-REVIEW EXERCISES..... | 223 |
| 9.11 | EXERCISES | 225 |
| CHAPTER 10. MIXING C AND ASSEMBLY CODE | | 227 |
| 10.1 | DATA TYPES AND ACCESS..... | 227 |
| 10.1.1 | <i>Signed or Unsigned Integers.....</i> | 228 |
| 10.1.2 | <i>Data Alignment</i> | 229 |
| 10.1.3 | <i>Data Structure Padding.....</i> | 230 |
| 10.2 | SPECIAL VARIABLES..... | 233 |
| 10.2.1 | <i>Static Variables</i> | 233 |
| 10.2.2 | <i>Volatile Variables.....</i> | 237 |
| 10.3 | INLINE ASSEMBLY | 239 |
| 10.3.1 | <i>Assembly Functions in a C Program.....</i> | 239 |
| 10.3.2 | <i>Inline Assembly Instructions in a C Program</i> | 240 |
| 10.4 | CALLING ASSEMBLY SUBROUTINES FROM A C PROGRAM | 241 |
| 10.4.1 | <i>Example of Calling an Assembly Subroutine.....</i> | 241 |
| 10.4.2 | <i>Example of Accessing C Variables in Assembly.....</i> | 242 |
| 10.5 | CALLING C FUNCTIONS FROM ASSEMBLY PROGRAMS..... | 243 |
| 10.5.1 | <i>Example of Calling a C Function.....</i> | 243 |
| 10.5.2 | <i>Example of Accessing Assembly Data in a C Program.....</i> | 244 |
| 10.6 | SELF-REVIEW EXERCISES..... | 245 |
| 10.7 | EXERCISES | 246 |
| CHAPTER 11. INTERRUPTS | | 249 |
| 11.1 | INTRODUCTION TO INTERRUPTS | 249 |
| 11.2 | INTERRUPT NUMBERS | 250 |
| 11.3 | INTERRUPT SERVICE ROUTINES | 252 |
| 11.4 | INTERRUPT VECTOR TABLE..... | 253 |
| 11.5 | INTERRUPT STACKING AND UNSTACKING..... | 255 |
| 11.6 | NESTED VECTORED INTERRUPT CONTROLLER (NVIC) | 257 |
| 11.6.1 | <i>Enable and Disable Peripheral Interrupts</i> | 258 |
| 11.6.2 | <i>Interrupt Priority.....</i> | 260 |
| 11.6.3 | <i>Global Interrupt Enable and Disable.....</i> | 264 |
| 11.7 | SYSTEM TIMER..... | 265 |

| | | |
|---|--|------------|
| 11.7.1 | <i>Timer Diagram and Registers</i> | 266 |
| 11.7.2 | <i>Timer Resolution and Period</i> | 268 |
| 11.7.3 | <i>Implementing a Delay Function</i> | 269 |
| 11.8 | EXTERNAL INTERRUPT | 272 |
| 11.9 | SOFTWARE INTERRUPT | 275 |
| 11.10 | SELF-REVIEW EXERCISES | 276 |
| 11.11 | EXERCISES..... | 279 |
| CHAPTER 12. FIXED-POINT ARITHMETIC | | 281 |
| 12.1 | FIXED-POINT REPRESENTATION | 282 |
| 12.1.1 | <i>Unsigned Fixed-point Representation</i> | 283 |
| 12.1.2 | <i>Signed Fixed-point Representation</i> | 284 |
| 12.1.3 | <i>Converting to Fixed-point Format</i> | 285 |
| 12.2 | FIXED-POINT RANGE AND RESOLUTION TRADEOFF..... | 286 |
| 12.3 | FIXED-POINT ADDITION AND SUBTRACTION..... | 287 |
| 12.4 | FIXED-POINT MULTIPLICATION..... | 289 |
| 12.5 | FIXED-POINT DIVISION | 290 |
| 12.6 | SELF-REVIEW EXERCISES | 291 |
| 12.7 | EXERCISES..... | 292 |
| CHAPTER 13. FLOATING-POINT ARITHMETIC | | 293 |
| 13.1 | FLOATING-POINT REPRESENTATION..... | 293 |
| 13.1.1 | <i>Special Values</i> | 297 |
| 13.1.2 | <i>Subnormal Numbers</i> | 297 |
| 13.1.3 | <i>Overflow and Underflow</i> | 298 |
| 13.1.4 | <i>Tradeoff between Numeric Range and Resolution</i> | 299 |
| 13.1.5 | <i>Rounding Rules</i> | 301 |
| 13.2 | SOFTWARE-BASED FLOATING-POINT OPERATIONS..... | 303 |
| 13.2.1 | <i>Floating-point Addition</i> | 304 |
| 13.2.2 | <i>Floating-point Multiplication</i> | 308 |
| 13.3 | HARDWARE-BASED FLOATING-POINT OPERATIONS..... | 310 |
| 13.3.1 | <i>FPU Registers</i> | 311 |
| 13.3.2 | <i>Load and Store Floating-point Numbers</i> | 319 |
| 13.3.3 | <i>Copy Floating-point Numbers</i> | 320 |
| 13.3.4 | <i>Copy and Set the Status and Control Register</i> | 320 |
| 13.3.5 | <i>Single-precision Arithmetic Operations</i> | 321 |
| 13.3.6 | <i>Single-precision comparisons</i> | 322 |
| 13.3.7 | <i>Precision Conversion</i> | 323 |
| 13.3.8 | <i>FPU Exception and Exception handling</i> | 325 |
| 13.3.9 | <i>Example Assembly Programs</i> | 328 |
| 13.4 | SELF-REVIEW EXERCISES | 330 |
| 13.5 | EXERCISES..... | 334 |
| CHAPTER 14. INSTRUCTION ENCODING AND DECODING..... | | 337 |
| 14.1 | TRADEOFF BETWEEN CODE DENSITY AND PERFORMANCE..... | 337 |
| 14.2 | DIVIDING BIT STREAMS INTO 16- OR 32-BIT INSTRUCTIONS..... | 338 |

| | | |
|--|---|------------|
| 14.3 | ENCODING 16-BIT THUMB INSTRUCTIONS..... | 339 |
| 14.4 | ENCODING 32-BIT INSTRUCTIONS..... | 340 |
| 14.5 | ENCODING IMMEDIATE NUMBERS..... | 342 |
| 14.6 | CALCULATING TARGET MEMORY ADDRESS..... | 343 |
| 14.7 | INSTRUCTION DECODING EXAMPLE 1..... | 344 |
| 14.8 | INSTRUCTION DECODING EXAMPLE 2..... | 349 |
| 14.9 | SELF-REVIEW EXERCISES..... | 353 |
| 14.10 | EXERCISES..... | 354 |
| CHAPTER 15. GENERAL PURPOSE I/O (GPIO)..... | | 355 |
| 15.1 | INTRODUCTION TO GENERAL PURPOSE I/O (GPIO)..... | 355 |
| 15.2 | GPIO INPUT MODES: PULL UP AND PULL DOWN..... | 356 |
| 15.3 | GPIO INPUT: SCHMITT TRIGGER..... | 357 |
| 15.4 | GPIO OUTPUT MODES: PUSH-PULL AND OPEN-DRAIN..... | 359 |
| 15.4.1 | <i>Push-Pull Output</i> | 359 |
| 15.4.2 | <i>Open-Drain Output</i> | 360 |
| 15.5 | GPIO OUTPUT SPEED: SLEW RATE..... | 362 |
| 15.6 | MEMORY-MAPPED I/O..... | 363 |
| 15.7 | LIGHTING UP AN LED..... | 366 |
| 15.8 | PUSH BUTTON..... | 371 |
| 15.9 | KEYPAD SCAN..... | 376 |
| 15.10 | STEPPER MOTOR CONTROL..... | 382 |
| 15.10.1 | <i>Bipolar and Unipolar Stepper Motor</i> | 382 |
| 15.10.2 | <i>Step Angle</i> | 383 |
| 15.10.3 | <i>Wave Stepping</i> | 384 |
| 15.10.4 | <i>Full Stepping</i> | 385 |
| 15.10.5 | <i>Half Stepping</i> | 386 |
| 15.10.6 | <i>Micro-stepping</i> | 388 |
| 15.10.7 | <i>Driving Stepper Motor</i> | 390 |
| 15.11 | LCD..... | 391 |
| 15.11.1 | <i>External Connection Diagram</i> | 391 |
| 15.11.2 | <i>Internal Font Encoding</i> | 394 |
| 15.11.3 | <i>Sending Commands and Data to LCD</i> | 395 |
| 15.11.4 | <i>Programming Fonts</i> | 398 |
| 15.12 | SELF-REVIEW EXERCISES..... | 399 |
| 15.13 | EXERCISES..... | 402 |
| CHAPTER 16. GENERAL-PURPOSE TIMERS..... | | 405 |
| 16.1 | TIMER ORGANIZATION AND COUNTING MODES..... | 405 |
| 16.2 | OUTPUT COMPARE..... | 408 |
| 16.2.1 | <i>Setting Output Mode</i> | 409 |
| 16.2.2 | <i>Example of Using a Timer to Toggle LED</i> | 411 |
| 16.2.3 | <i>Timer Update Events</i> | 414 |
| 16.3 | PWM OUTPUT..... | 416 |
| 16.3.1 | <i>PWM Alignment</i> | 420 |
| 16.3.2 | <i>PWM Programming Flowchart</i> | 421 |

| | | |
|---|--|------------|
| 16.4 | INPUT CAPTURE..... | 425 |
| 16.4.1 | <i>Configuring Input Capture</i> | 429 |
| 16.4.2 | <i>Input Capture in Slave Mode with Reset</i> | 434 |
| 16.4.3 | <i>Interfacing to Ultrasonic Distance Sensor</i> | 436 |
| 16.5 | REAL-TIME CLOCK..... | 442 |
| 16.5.1 | <i>UNIX Epoch Time</i> | 442 |
| 16.5.2 | <i>RTC Frequency Settings</i> | 444 |
| 16.5.3 | <i>Oscillator Frequency Accuracy</i> | 445 |
| 16.5.4 | <i>Binary Coded Decimal (BCD) Encoding</i> | 446 |
| 16.5.5 | <i>RTC Initialization</i> | 447 |
| 16.5.6 | <i>RTC Alarm</i> | 451 |
| 16.5.7 | <i>Using RTC to Wake Processors up from Sleep Mode</i> | 453 |
| 16.6 | SELF-REVIEW EXERCISES..... | 456 |
| 16.7 | EXERCISES..... | 459 |
| CHAPTER 17. DIRECT MEMORY ACCESS (DMA) | | 461 |
| 17.1 | ADVANCED MICROCONTROLLER BUS ARCHITECTURE (AMBA)..... | 462 |
| 17.2 | INTERFACING A PERIPHERAL WITHOUT AND WITH DMA..... | 464 |
| 17.3 | DMA CHANNELS..... | 465 |
| 17.4 | PROGRAMMING DMA..... | 468 |
| 17.5 | DMA CIRCULAR MODE..... | 470 |
| 17.6 | DMA INTERRUPTS..... | 471 |
| 17.7 | SELF-REVIEW EXERCISES..... | 472 |
| 17.8 | EXERCISES..... | 474 |
| CHAPTER 18. ANALOG INPUT AND OUTPUT..... | | 475 |
| 18.1 | ANALOG-TO-DIGITAL CONVERTER (ADC)..... | 475 |
| 18.1.1 | <i>ADC Architecture</i> | 475 |
| 18.1.2 | <i>Digital Quantization</i> | 476 |
| 18.1.3 | <i>Sample and Hold</i> | 477 |
| 18.1.4 | <i>ADC Quantization Error</i> | 478 |
| 18.1.5 | <i>ADC Diagram</i> | 480 |
| 18.1.6 | <i>ADC Conversion Modes</i> | 481 |
| 18.1.7 | <i>ADC Data Alignment</i> | 483 |
| 18.1.8 | <i>ADC Input Channels</i> | 484 |
| 18.1.9 | <i>ADC Triggers</i> | 485 |
| 18.1.10 | <i>Measuring the Input Voltage</i> | 488 |
| 18.1.11 | <i>ADC Configuration Flowchart</i> | 489 |
| 18.1.12 | <i>ADC with DMA</i> | 493 |
| 18.1.13 | <i>DMA with Ping-Pong Buffering</i> | 495 |
| 18.1.14 | <i>ADC Calibration</i> | 497 |
| 18.2 | DIGITAL-TO-ANALOG CONVERTER (DAC)..... | 498 |
| 18.2.1 | <i>DAC Architecture</i> | 498 |
| 18.2.2 | <i>DAC on STM32L Processors</i> | 500 |
| 18.2.3 | <i>Conversion Trigger</i> | 501 |
| 18.2.4 | <i>Buffered Output</i> | 501 |

| | | |
|---|--|------------|
| 18.2.5 | <i>Generating a Sine Wave via Table Lookup</i> | 502 |
| 18.2.6 | <i>DAC with Software Trigger</i> | 506 |
| 18.2.7 | <i>Using Timer as a Trigger to DAC</i> | 507 |
| 18.2.8 | <i>Musical Synthesizing</i> | 510 |
| 18.3 | SELF-REVIEW EXERCISES | 515 |
| 18.4 | EXERCISES | 518 |
| CHAPTER 19. SERIAL COMMUNICATION PROTOCOLS | | 521 |
| 19.1 | UNIVERSAL ASYNCHRONOUS RECEIVER AND TRANSMITTER | 521 |
| 19.1.1 | <i>Communication Frame</i> | 522 |
| 19.1.2 | <i>Bit Rate and Baud Rate</i> | 523 |
| 19.1.3 | <i>UART Standards</i> | 525 |
| 19.1.4 | <i>UART Communication via Polling</i> | 526 |
| 19.1.5 | <i>UART Communication via Interrupt</i> | 530 |
| 19.1.6 | <i>UART Communication via DMA</i> | 533 |
| 19.1.7 | <i>Serial Communication to Bluetooth Module</i> | 536 |
| 19.2 | INTER-INTEGRATED CIRCUIT (I ² C) | 539 |
| 19.2.1 | <i>I²C Pins</i> | 539 |
| 19.2.2 | <i>I²C Protocol</i> | 540 |
| 19.2.3 | <i>I²C Data Frame</i> | 542 |
| 19.2.4 | <i>Interfacing Serial Digital Thermal Sensors via I²C</i> | 543 |
| 19.2.5 | <i>I²C Programmable Timings</i> | 545 |
| 19.2.6 | <i>Sending Data to I²C Slave via Polling</i> | 551 |
| 19.2.7 | <i>Receiving Data from I²C Slave via Polling</i> | 552 |
| 19.2.8 | <i>Interfacing a Temperature Sensor via Polling</i> | 553 |
| 19.2.9 | <i>Transferring Data via DMA on I²C Master</i> | 558 |
| 19.3 | SERIAL PERIPHERAL INTERFACE BUS (SPI) | 560 |
| 19.3.1 | <i>Data Exchange</i> | 560 |
| 19.3.2 | <i>Clock Configuration</i> | 562 |
| 19.3.3 | <i>Using SPI to Interface a Gyroscope</i> | 563 |
| 19.4 | UNIVERSAL SERIAL BUS (USB) | 569 |
| 19.4.1 | <i>USB Bus Layer</i> | 570 |
| 19.4.2 | <i>USB Device Layer</i> | 572 |
| 19.4.3 | <i>USB Function Layer</i> | 574 |
| 19.4.4 | <i>USB Class Layer</i> | 582 |
| 19.4.5 | <i>Human Interface Device (HID)</i> | 582 |
| 19.5 | SELF-REVIEW EXERCISES | 587 |
| 19.6 | EXERCISES | 589 |
| CHAPTER 20. MULTITASKING | | 591 |
| 20.1 | PROCESSOR MODE AND PRIVILEGE LEVEL | 592 |
| 20.1.1 | <i>Control Register</i> | 593 |
| 20.1.2 | <i>Exception Return Value (EXC_RETURN)</i> | 594 |
| 20.1.3 | <i>Selection of MSP and PSP in Thread Mode</i> | 596 |
| 20.2 | SUPERVISOR CALL (SVC) | 597 |
| 20.3 | CPU SCHEDULING | 600 |

| | | |
|--|--|------------|
| 20.4 | EXAMPLE OF ROUND ROBIN SCHEDULING | 602 |
| 20.5 | SELF-REVIEW EXERCISES | 608 |
| 20.6 | EXERCISES | 610 |
| CHAPTER 21. DIGITAL SIGNAL PROCESSING (DSP) | | 611 |
| 21.1 | FIXED-POINT AND FLOATING-POINT DSP..... | 611 |
| 21.2 | FIXED-POINT DATA TYPES IN DSP..... | 612 |
| 21.3 | SATURATION..... | 613 |
| 21.4 | ARITHMETIC INSTRUCTIONS..... | 615 |
| 21.4.1 | <i>Parallel 8-bit Add and Subtract</i> | 616 |
| 21.4.2 | <i>Parallel 16-bit Add and Subtract</i> | 620 |
| 21.4.3 | <i>32-bit Add and Subtract</i> | 622 |
| 21.4.4 | <i>Sum of Absolute Difference</i> | 623 |
| 21.4.5 | <i>Extension and Add</i> | 624 |
| 21.4.6 | <i>Add and Subtract Half-words with Exchange</i> | 625 |
| 21.4.7 | <i>16-bit and 32-bit Multiplication</i> | 627 |
| 21.4.8 | <i>16-bit Multiply and Accumulate with 64-bit Result</i> | 630 |
| 21.4.9 | <i>16-bit Multiply and Accumulate with 32-bit Result</i> | 633 |
| 21.4.10 | <i>16×32 Multiply and Accumulate with 32-bit Result</i> | 635 |
| 21.4.11 | <i>32×32 Multiply and Accumulate with 32-bit Result</i> | 635 |
| 21.4.12 | <i>Unsigned Long Multiply with Accumulate Accumulate</i> | 635 |
| 21.5 | PACKING HALF-WORDS INTO A WORD | 637 |
| 21.6 | SIGNED AND UNSIGNED EXTENSION..... | 638 |
| 21.7 | GE FLAGS..... | 639 |
| 21.8 | BYTE SELECTION INSTRUCTION..... | 640 |
| 21.9 | BASIC DSP FUNCTIONS..... | 641 |
| 21.9.1 | <i>Vector Negate</i> | 641 |
| 21.9.2 | <i>Vector Absolution Value</i> | 642 |
| 21.9.3 | <i>Vector Offset with Saturation</i> | 644 |
| 21.9.4 | <i>Vector Shift Left with Saturation</i> | 645 |
| 21.9.5 | <i>Vector Mean</i> | 647 |
| 21.9.6 | <i>Vector Pairwise Multiplication</i> | 648 |
| 21.9.7 | <i>Vector Dot Product</i> | 650 |
| 21.9.8 | <i>Vector Min and Max</i> | 652 |
| 21.10 | SELF-REVIEW EXERCISES | 653 |
| 21.11 | EXERCISES..... | 654 |
| APPENDIX A: ASCII CHARACTERS | | 656 |
| APPENDIX B: GNU COMPILER..... | | 657 |
| APPENDIX C: CORTEX-M3/M4 INSTRUCTIONS..... | | 668 |
| APPENDIX D: FLOATING-POINT INSTRUCTIONS..... | | 670 |
| APPENDIX E: DSP INSTRUCTIONS ON CORTEX-M4 AND CORTEX-M7 | | 672 |
| APPENDIX F: CORTEX-M0/M0+/M1 INSTRUCTIONS | | 675 |
| APPENDIX G: CORTEX-M3/M4 16-BIT THUMB-2 INSTRUCTION ENCODING..... | | 677 |

| | |
|--|------------|
| APPENDIX H: CORTEX-M3/M4 32-BIT THUMB-2 INSTRUCTION ENCODING..... | 679 |
| APPENDIX I: GPIO ALTERNATE FUNCTIONS FOR STM32L4..... | 686 |
| BIBLIOGRAPHY | 693 |
| INDEX | 699 |
| SOLUTIONS TO SELF-REVIEW EXERCISES | 713 |