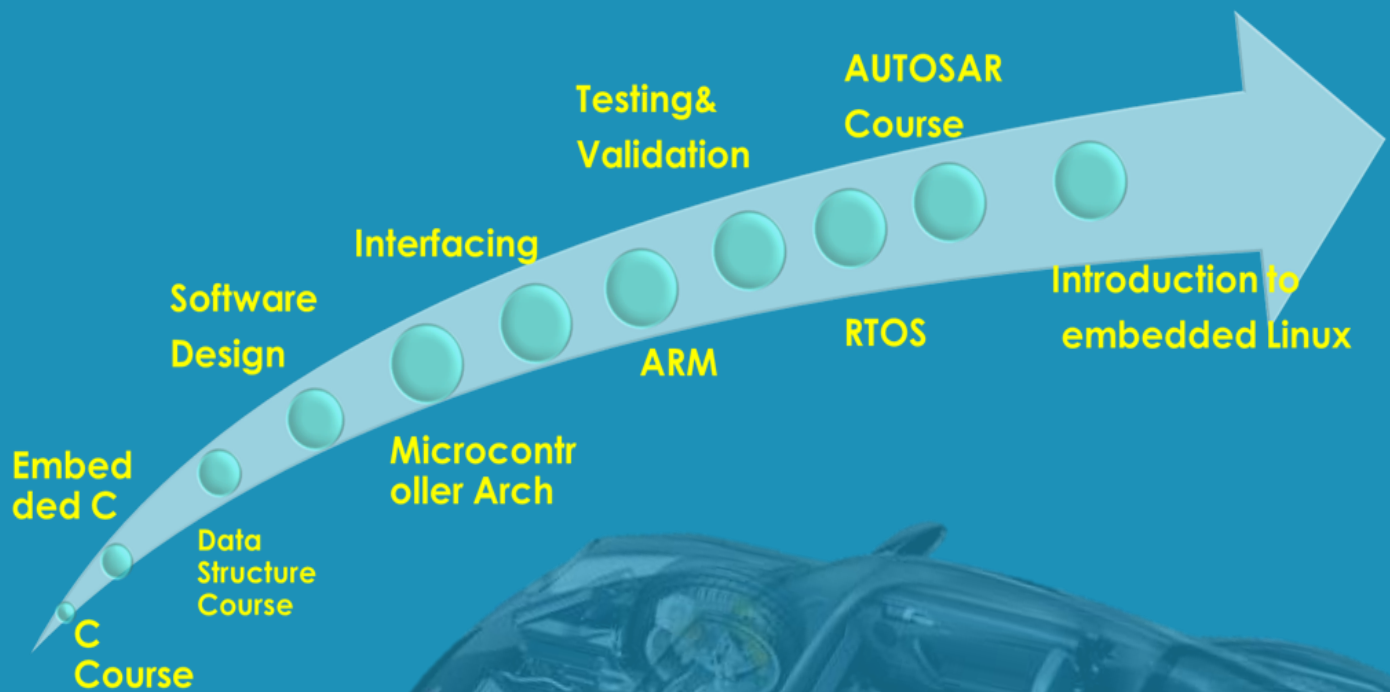


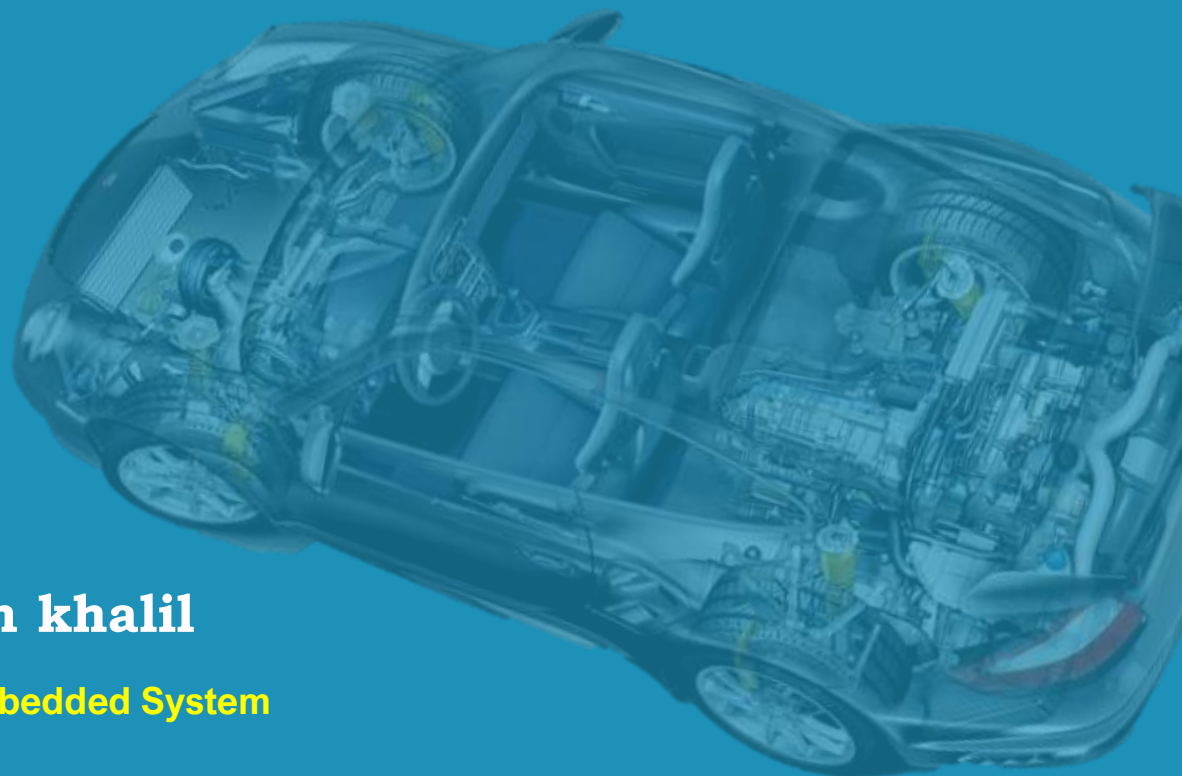
Mastering Embedded System From Scratch



First Edition

Keroles karam khalil

Be Professional in Embedded System



DISCLAIMER

© www.learn-in-depth.com
All Rights Reserved

In the DISCLAIMER section of your book, you can write something like this:

Disclaimer: The information contained in this book is provided for educational and informational purposes only. The author has made every effort to ensure the accuracy and completeness of the information provided. However, the author, publisher, and any other parties involved in the creation, publication, or distribution of this book shall not be held responsible for any errors, inaccuracies, or omissions, or for any results obtained from the use of this information.

Readers are encouraged to verify any information they intend to rely upon and consult with appropriate professionals to ensure the information is accurate and suitable for their specific needs. The author is not liable for any loss, damage, or other consequences arising from the use of the information contained in this book.

This book is not intended to replace or substitute for professional advice, and the author disclaims any liability, loss, or risk incurred as a consequence, directly or indirectly, of the use and application of any of the contents of this book.

This book contains information derived from various open-source websites, public resources, and references in the embedded systems field. The author has made every effort to acknowledge and cite these sources appropriately in the reference section at the end of the book. However, the author does not claim any ownership or responsibility for the original content from these sources. The purpose of including this information is to provide a comprehensive and consolidated resource for readers who are interested in learning and mastering embedded systems. Readers are encouraged to explore these references and sources for further information and to gain a deeper understanding of the subject matter.

"Mastering Embedded Systems From Scratch" is your ultimate guide to becoming a professional embedded systems engineer. Curated from 24 authoritative references, this comprehensive book will fuel your passion and inspire success in the fast-paced world of embedded systems. Dive in and unleash your potential!



CONTENTS

TABLE OF CONTENTS

CONTENTS	3
PREFACE	26
OVERVIEW	26
ACKNOWLEDGMENTS	27
ABOUT THE AUTHOR	28
CHAPTER 1. INTRODUCTION TO EMBEDDED SYSTEM	31
ABSTRACT	31
EMBEDDED SYSTEM WHAT'S THAT ?	31
EMBEDDED SYSTEM CLASSIFICATION	32
HYPERVERSORS	33
ADDRESS BINDING (TO BUILD EMBEDDED SW EXECUTABLE)	34
EXAMPLES FOR SoCs	35
EMBEDDED SYSTEM LEARNING CHEAT SHEET	38
VERSION CONTROL SYSTEMS (VCS)	39
WHAT IS GIT?	40
GIT CONCEPTS	40
CHECKOUT A REPOSITORY	43
WORKFLOW	43
PUSHING CHANGES	43
GITK	44
BRANCHING	44
UPDATE & MERGE	45
LOG	46
GITHUB	46
GIT COMMANDS CHEAT SHEET	48
CHAPTER 2. C PROGRAMMING	51

ABSTRACT

51

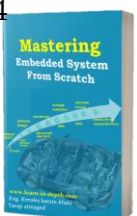
PAGE 3 OF 1884

Mastering Embedded System from Scratch

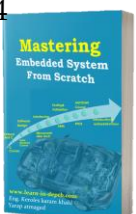
Copyright © 2023 – Kerolos Khalil, eng.kerolos.karam@gmail.com



CH2.C - PART 1 : C-BASICS	52
WHY C, AND NOT ANOTHER LANGUAGE?	52
HISTORY	52
DEVELOPMENT ENVIRONMENTS	53
C - ENVIRONMENT SETUP	53
INSTALL MINGW GCC STEPS	54
VARIABLE NAME	55
COMMENTS	56
INTEGER VALUES	57
FLOATING-POINT TYPES	58
2'S COMPLEMENT	58
INTEGER AND FLOAT CONVERSIONS	60
TYPE CONVERSION IN C	62
HIERARCHY OF OPERATIONS	64
C PROGRAMMING INPUT OUTPUT (I/O): PRINTF() AND SCANF()	65
C FLOATS INPUT/OUTPUT	67
MATHEMATICAL AND LOGICAL EXPRESSIONS	72
IDENTIFIERS	74
C KEYWORDS	75
C FUNDAMENTALS SUMMARY	76
CH2.C - PART 2 : C-CONDITION & LOOP	77
CONTROLLING PROGRAM FLOW	77
CONDITIONS	77
INLINE CONDITION / CONDITIONAL OPERATORS	80
SWITCH STATEMENT	82
FOR STATEMENT	83
WHILE STATEMENT	87
DO...WHILE STATEMENT	88
BREAK STATEMENT	90
CONTINUE STATEMENT	91
CH2.C - PART 3 : ARRAY/STRING	94
C ARRAY	94
STRINGS IN C	100
ARRAY OF STRINGS	104
CH2.C - PART 4: FUNCTION/STORAGE CLASSES	112
DIFFERENCE BETWEEN VARIABLE DEFINITION AND DECLARATION	112
FUNCTIONS	112
THE COMPILER GIVES AN ERROR AT THE LINE PRINTWELCOME(); IN THE MAIN FUNCTION, THE ERROR STATE THAT "THE FUNCTION PRINTWELCOME IS UNDEFINED". WHICH MEANS THAT THE COMPILER CANNOT LOCATE THE FUNCTION BEFORE THE MAIN, EVEN IF IT IS LOCATED AFTER THE MAIN?	114



DIFFERENCE BETWEEN PASSING SINGLE VALUES AND ARRAYS	117
WHAT IS MEMORY LAYOUT OF C-PROGRAM ?	121
STORAGE CLASSES IN C	122
STORAGE CLASSES SUMMARY	135
CALLING MECHANISM	136
INLINE ASSEMBLY	141
CH2.C - PART 5: STRUCTURES/UNIONS/ENUM/MACROS	144
STRUCTURE IN C	144
ENUM IN C	165
UNION IN C	167
CH2.C - PART 6: MACROS/CONSTANT/#PRAGMA	173
C – PREPROCESSOR DIRECTIVES	173
MACRO IN C	176
__VA_ARGS__	179
#UNDEF DIRECTIVE IN C	183
PREDEFINED MACROS	186
CONDITIONAL COMPILATION IN C	188
PRAGMA DIRECTIVE IN C ○ PRAGMA IS IMPLEMENTATION SPECIFIC DIRECTIVE I.E EACH PRAGMA DIRECTIVE HAS DIFFERENT IMPLEMENTATION RULE AND USE .	192
TYPE QUALIFIER IN C	195
CONSTANT IN C	196
CH2.C - PART 7: POINTERS	202
WHAT ARE POINTERS?	202
WHY DO WE NEED POINTER?	203
POINTER SIZE	204
POINTER CASTING	204
POINTER ARITHMETIC	207
POINTER TO ARRAY	210
POINTER TO STRUCTURE	211
POINTERS AND FUNCTIONS	213
POINTER WITH UNKNOWN TYPE (VOID*)	218
MULTIPLE INDIRECTION: POINTER TO POINTER	221
NULL AND UNASSIGNED POINTERS	223
POINTER TO FUNCTION	227
POINTER TRICKS	230
OTHERS POINTERS TRICKS IN C	245
CHAPTER 2. C PROGRAMMING (ASSIGNMENTS)	248
PART 1 : C-BASICS ASSIGNMENT 1	248
PART 2 : C-CONDTION & LOOP ASSIGNMENT	250
PART 3 : ARRAY/STRING ASSIGNMENTS	254



PART 4: FUNCTION/STORAGE CLASSES ASSIGNMENTS	256
PART 5: STRUCTURES/UNIONS/ENUM/MACROS ASSIGNMENTS	256
PART 7: POINTERS ASSIGNMENTS	259
CHAPTER 2. C PROGRAMMING (QUIZZES)	261
CHAPTER 3. EMBEDDED C	286
<hr/>	
ABSTRACT	286
CH3 EMBEDDED C - PART1	287
TYPEDEF COMMAND	287
HEADER PROTECTION	291
C OPTIMIZATION	292
VOLATILE TYPE QUALIFIER	294
CH3. EMBEDDED C - PART2	301
BARE METAL EMBEDDED SW	301
COMPILATION PROCESS	305
CH3. EMBEDDED C - PART3	315
BOOTING SEQUENCE	315
RUNNING MODE	318
BOOTLOADER VS STARTUP	320
EMBEDDED C LAB1	326
WHAT IS QEMU ?	328
GDB DEBUGGER COMMANDS	354
DEBUGGING REMOTE EMBEDDED SW	356
CH3. EMBEDDED C - PART4	364
MAKEFILE TUTORIAL	364
CMAKE - BASIC USAGE	369
LAB2: LET US LEARN TOGETHER EVERY THING WITH AN EXAMPLE WRITE BAREMETAL SW ON FUNCTIONAL ATTRIBUTE: WEAK AND ALIAS IN EMBEDDED C	370
CH3. EMBEDDED C - PART5	388
LAB3: LET US LEARN TOGETHER EVERY THING WITH AN EXAMPLE WRITE BAREMETAL SW ON TM4C123 ARM CORTEXM4	388
DEBUGGING MECHANISM	398
DEBUGGING MECHANISM THROUGH DEBUG CIRCUIT	400
DEBUG TM4C123 BY OPENOCD GDBSERVER	402
DEBUG TM4C123 BY OPENOCD GDBSERVER WITH IDE (ECLIPSE)	405
DEBUG TM4C123 BY KEIL-UVISION	406
FAMOUS JTAG/SWD (EMULATORS/DEBUGGERS_ADAPTORS)	407
LAUTERBACH \ TRACE32	411
RENESAS E1/E2 DEBUGGER	416



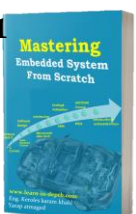
CH3. EMBEDDED C - PART6	420
DYNAMIC ALLOCATION	420
WRITE A DYNAMIC ALLOCATION CODE IN EMBEDDED C (MODIFY LAB 2)	422
WHAT ARE THE UNDEFINED SYMBOLS _SBRK ?	424
C STANDARD LIBRARY	424
IMPLEMENT _SBRK TO SUPPORT MALLOC IN EMBEDDED C	426
LET US SUPPORT PRINTF IN EMBEDDED C	429

CHAPTER 4. DATA STRUCTURE/SW DESIGN **431**

ABSTRACT	431
CH4. PART1 - DATA STRUCTURE FOR EMBEDDED SYSTEM	432
INTRODUCTION TO DATA STRUCTURES	432
LIFO BUFFER	434
CIRCULAR BUFFER & FIFOS	443
LINEAR VS NON-LINEAR DATA STRUCTURE	452
DATA STRUCTURE - LINKED LIST	452
DATA STRUCTURE - DOUBLY LINKED LIST	458
CH4. PART 2 - EMBEDDED SYSTEMS ARCHITECTING	475
STATE MACHINES IN C	475
DESIGNING EMBEDDED SYSTEMS	492
CH4. PART3 - SYSTEM EXPLORATION	495
UML FOR EMBEDDED SYSTEM	495
EMBEDDED SYSTEM ARCHITECTING/DESIGN SEQUENCE	499
UML: USE CASE DIAGRAM	504
UML: ACTIVITY DIAGRAM	505
UML: SEQUENCE DIAGRAM	507
SYSTEM DESIGN	508
STATE MACHINE - STATES AND TRANSITIONS	508
BLOCKS COMMUNICATION	510
PRESSURE CONTROLLER PROJECT	511
WINDOWS HOST MACHINE NATIVE TOOL CHAIN UTILIZATION USING ECLIPSE IDE	537
TESTING & SIMULATION	538
HARDWARE PROTEUS SIMULATION	542
MISRA-C RULES VIOLATION CHECKER	544

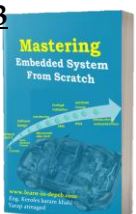
CHAPTER 5. MICROCONTROLLER FUNDAMENTALS **550**

ABSTRACT	550
CH5. PART1- MCU FUNDAMENTALS	551



MCU FUNDAMENTALS	551
DIFFERENCE BETWEEN IC, MPU, MCU, SoC AND ECU	551
ELECTRONIC CONTROL UNIT (ECU)	558
CACHE MEMORY	560
CACHE COHERENCE	562
FLOATING-POINT UNIT (FPU)	563
MEMORY PROTECTION UNIT (MPU)	564
MCU MEMORY TYPES	567
VON NEUMANN Vs HARVARD AND MODIFIED HARVARD	575
PIPELINE TECHNIQUE	578
CISC Vs RISC	580
CH5. PART2- MCU MEMORY SPACE	582
PORT-MAPPED I/O	582
MEMORY-MAPPED I/O	582
NAVIGATE “MEMORY MAP” FOR STM32 IN TRM	592
NAVIGATE “MEMORY MAP” FOR TM4C123 IN TRM	594
MCU BUS INTERFACES	600
UNDERSTANDING AMBA BUS ARCHITECTURE AND PROTOCOLS	601
BIT, BYTE, HALFWORD, WORD, DOUBLE-WORD AND NIBBLES	613
BIG AND LITTLE ENDIAN	614
CH5. PART3- MCU CLOCKS	616
MCU CLOCKING	616
MCU CLOCK SOURCES	621
UNDERSTANDING CLOCK TREE	623
RCC REGISTERS IN STM32F103C8T6	630
CH5. PART4- MCU INTERRUPTS	647
WHAT IS AN INTERRUPT?	647
INTERRUPT SERVICE ROUTINE	647
FUNCTIONAL ATTRIBUTE: WEAK AND ALIAS FOR ISR	651
SERVICING INTERRUPTS	651
INTERRUPT VECTOR TABLE	653
POLLING VS. INTERRUPT	654
INTERRUPT PROCESSING	655
WHAT IS INTERRUPT LATENCY?	656
SEQUENTIAL INTERRUPT PROCESSING VS NESTED INTERRUPT PROCESSING	656
TYPES OF INTERRUPTS	657
INTERRUPT CONTROLLER IC	659
WHAT IS INTERRUPT OVERLOAD?	671

CHAPTER 6. MCU ESSENTIAL PERIPHERALS **673**



ABSTRACT	673
CH6. PART1 - GPIO	674
WHAT ARE PERIPHERALS?	674
GPIO CONTROLLERS	676
GPIO INTERNAL CIRCUIT	680
GPIO OUTPUT SPEED: SLEW RATE	684
GPIO INPUT MODES: PULL UP AND PULL DOWN	685
HOW TO READ GPIO MODULE FROM TRM	687
CH6. PART2 - GPIO IOs/ALT	695
THINKING QUESTION	695
ALTERNATE FUNCTIONS (AF) FUNCTIONAL DESCRIPTION	698
DEEP DIVE INSIDE SoC TO SEE PA.0 CONFIGURED TO BE EXTERNAL INTERRUPT (BLUEPILL)	699
CH6. PART3 - GPIO/EXTI DRIVER	703
GPIO DRIVER FOR (STM32F103C8T6)	703
MCU DEVICE HEADER	703
GPIO DRIVER	706
EXTI DRIVER FOR (STM32F103C8T6)	711
CH6. PART4 - TIMERS	715
WHAT IS A TIMER ?	715
TIMER IN DEPTH	716
CASE STUDIES: CASE STUDY 1	718
TIMER INFRA STRUCTURE	719
ATMEGA 32 TIMER 0 MODES	720
TIMER INPUT CAPTURE MODE	720
STM32F103XX TIMERS	721
PWM (PULSE WIDTH MODULATION)	723
HOW DOES A SERVO MOTOR WORK?	724
DC MOTOR	725
REAL-TIME CLOCK (RTC)	726
WATCHDOG (WDOG)	727
THINKING QUESTIONS	729
CH6. PART5 - ADC	741
ADC INTRODUCTION	741
SIGNALS CONCEPT	742
MAJOR CHARACTERISTICS OF THE ADC	751
SIGNAL TO NOISE RATIO SNR	752
IN YOUR OPINION HOW THE ACCURACY OF AN ADC CAN BE IMPROVED ?	753
ADC TYPES	754
ADC MODE IN (STM32F103C8T6)	756
ADC CALIBRATION	759



ABSTRACT	760
CH7. PART1- MCU IO ELECTRICAL CHARACTERISTICS	761
MCU IO ELECTRICAL CHARACTERISTICS	761
EXAMPLE: CONNECT TWO STM32F103C8X TOGETHER	772
CASE STUDY 2: TM4C123	773
CASE STUDY 3: ATMEGA32	775
CASE STUDY 4: ATMEGA32 CONNECTED WITH 3.3V SENSOR	777
INTERFACING MAIN CONCEPTS	779
CH7. PART2- UART	784
UART "UNIVERSAL ASYNCHRONOUS RECEIVER-TRANSMITTER"	784
WIRE STANDARD CONJUGATION WITH USART PROTOCOL	790
USART CONTROLLER GENERAL CIRCUIT	793
USART BLOCK DIAGRAM ON STM32F103X	794
NRZ (NON-RETURN-TO-ZERO)	797
UART OVER SAMPLING MECHANISM	798
THREE DIFFERENT METHODS OF SENDING OR RECEIVING DATA VIA UART PORTS:	802
WRITE UART DRIVER (STM32F103C8T6)	805
UART DRIVER FOR (STM32F103C8T6)	809
CASE STUDY 1 WITH DEBUGGING : USING POLLING MECHANISM	817
CASE STUDY 2 WITH DEBUGGING : USING INTERRUPT MECHANISM	818
CH7. PART3- SPI PROTOCOL	820
SERIAL PERIPHERAL INTERFACE (SPI)	820
HOW SPI WORKS ?	823
SPI BUS CONFIGURATION	825
SPI WITH FLASH MEMORIES (CUSTOM PROTOCOL)	828
SPI BUS 3-WIRE	829
QUAD SPI	830
SPI CONTROLLER (READ FROM TRM AND WRITE A DRIVER)	832
SPI DRIVER FOR (STM32F103C8T6)	832
LAB1: SPI DEBUG/ANALYZE SPI MASTER	843
LAB2: TERMINAL1 <----> USART1 : MCU1 : (SPI1 MASTER) ---> (SPI2 SLAVE) :MCU2: USART2 --->	
TERMINAL2	844
CH7. PART4- I2C PROTOCOL	846
WHAT IS I ² C	846
I2C CHARACTERISTICS	846
I2C SPEED CATEGORIES	847
I2C PINS	848
I2C BUS DEFINITIONS	849



START AND STOP CONDITIONS	850
HOW I2C WORK ?	850
I2C DATA SAMPLING	855
WHAT HAPPEN IF TWO MASTERS INITIATE START CONDITION AT THE SAME TIME OR HAVE MULTIPLE SLAVE DEVICES THAT RESPOND TO A SINGLE ADDRESS?	860
I2C CLOCK STRETCHING	861
MULTIBYTE BURST WRITE/READ	862
I2C CLOCK SYNCHRONIZATION	863
HOW TO SELECT BETWEEN I2C AND SPI	864
I2C ADDRESSING	865
SENDING DATA TO I2C SLAVE VIA POLLING	866
RECEIVING DATA FROM I2C SLAVE VIA POLLING	868
TRANSFERRING DATA VIA DMA ON I2C MASTER	869
I2C PROGRAMMABLE TIMINGS	870
RISE TIME AND FALL TIME	871
DATA HOLD TIME	871
DATA SETUP TIME	872
MASTER CLOCK'S MINIMUM HIGH AND LOW TIME	872
SYSTEM MANAGEMENT BUS (SMBUS)	873
I2C CONTROLLER DRIVER IN STM32F103XX	874
LET US WRITE (HAL:E2PROMDRIVER AND MCAL:I2CDRIVER) TOGETHER	883

CHAPTER 8. SW TESTING **886**

ABSTRACT	886
CH8. PART1-VERIFICATION AND VALIDATION	887
WHAT IS EMBEDDED SOFTWARE TESTING?	887
EMBEDDED SYSTEM SOFTWARE QUALITY IS MANDATORY	887
FIRST DEFINITION: ERRORS, FAULTS AND FAILURE	888
SECOND DEFINITION: TESTING	889
THIRD DEFINITION: VERIFICATION VS VALIDATION	890
SOFTWARE QUALITY CHARACTERISTICS	890
STATIC & DYNAMIC TESTING	891
QUALITY MANAGEMENT	892
QA VS QC	893
FALSE POSITIVE VS FALSE NEGATIVE	894
ROOT CAUSE ANALYSIS	895
TESTING VS DEBUGGING	895
SYSTEM ARCHITECTING/DESIGN SEQUENCE	896
TEST LEVELS	899



REGRESSION TESTING	901
WHITE, GRAY AND BLACK BOX TESTING	901
FUNCTIONAL VS NON-FUNCTIONAL TESTING	902
WATERFALL MODEL IN SOFTWARE DEVELOPMENT LIFE CYCLE	904
SPIRAL MODEL IN SOFTWARE DEVELOPMENT LIFE CYCLE	905
V MODEL IN SDLC	906
CH8. PART2-TEST_CASE_DESIGN_TECHNIQUES	908
TEST CASE DESIGN	908
HOW TO WRITE TEST CASES?	908
TEST_CASE_DESIGN_TECHNIQUES	909
OPEN-SOURCE CODE COVERAGE TOOLS	920
CH8. PART3-AGILE SCRUM METHODOLOGY	922
AGILE SCRUM METHODOLOGY	922
SCRUM TEAM	922
ARTIFACTS IN AGILE SCRUM METHODOLOGY	923
MEETINGS IN AGILE SCRUM METHODOLOGY:	924
ATLASSIAN JIRA	925
CHAPTER 9. ARM FUNDAMENTALS	928

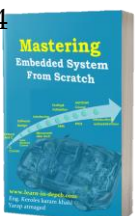
ABSTRACT	928
CH9.PART1-CORTEXM3/4 MODES/OPERATIONS/REGISTERS	929
DATA SIZE AND INSTRUCTION SET	929
MODES OF OPERATION AND EXECUTION ARM CORTEX M3/4	929
CORTEX-M REGISTERS	932
PROCESSOR REGISTERS	934
SWITCHING FROM USER TO PRIVILEGED ACCESS MODE	939
THUMB / ARM / THUMB2 ISA	940
ARM LAB1: ARM MODES	942
CH9.PART2- INLINEASSEMBLY	944
THE CORTEX-M4 INSTRUCTION SET	944
LAB: WRITE ASSEMBLY TO ACHIEVE THE CLOSER FUNCTIONALITY.	947
MOST COMMON INSTRUCTIONS	947
LOAD / STORE INSTRUCTIONS	948
INLINE FUNCTIONS	950
CH9.PART3 - CORTEXM RESETSEQUENCE	960
THE EXECUTION PROGRAM STATUS REGISTER (EPSR) PC & T ADDRESSING	965
3-STAGE INSTRUCTION PIPELINE	968
CH9. PART4 - STACK MEMORY	970
CORTEXM3/4 MEMORY MAP	970



CORTEXM3/4 STACK MEMORY	971
SHADOWED STACK POINTER	976
ARM PROCEDURE CALL STANDARD	980
CONTEXT SAVING ON STACK (EXCEPTION/ INTERRUPT)	989
MEMORY PROTECTION UNIT (MPU)	990
CH9. PART5 - EXCEPTIONS AND INTERRUPTS	993
EXCEPTION TYPES	994
CMSIS (CORTEX MICROCONTROLLER SOFTWARE INTERFACE STANDARD)	995
OVERVIEW OF INTERRUPT MANAGEMENT	999
HOW TO MAKE THE INTERRUPT REACH TO THE CPU	1002
CORTEX-M3/4 PERIPHERALS	1003
ACCEPTANCE OF EXCEPTION REQUEST	1006
CPU'S EXCEPTION PROCESSING	1007
CPU'S EXCEPTION PROCESSING IN DETAILS	1007
EXITING AN EXCEPTION HANDLER	1014
EXCEPTION ENTRANCE SEQUENCE	1016
CAN EXAMINE VECTOR TABLE WITH DEBUGGER ?	1017
INTERRUPT RESPONSE LATENCY	1018
MAXIMUM INTERRUPT RATE	1019
EXCEPTION HANDLING: TAIL CHAINING	1020
LATE ARRIVAL EXCEPTION BEHAVIOR	1022
POP PREEMPTION	1023
CH9. PART6 - SVC/PENDSV EXCEPTIONS	1025
ARM CORTEX-M SVC EXCEPTION	1025
EXECUTING SVC	1028
SVC INSTRUCTION	1028
SVC HANDLER	1029
EXTRACTION OF THE SVC SERVICE NUMBER IN ASSEMBLY	1032
"SVC_HANDLER" IS STANDARDIZED IN CMSIS	1033
PENDSV EXCEPTION AND CONTEXT SWITCHING IN ARM CORTEX-M	1036

CHAPTER 10. RTOS **1043**

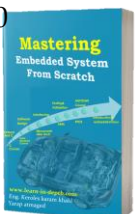
ABSTRACT	1043
CH10. PART1 - RTOS CONCEPTS	1045
HOW TO EXECUTE TWO FUNCTION AT THE SAME TIME ON THE BAREMETAL APPLICATION?	1045
WHAT IS AN OPERATING SYSTEM	1049
CONCURRENCY VERSUS PARALLELISM	1052
TASK SWITCHING LATENCY	1052
REAL TIME OS (RTOS)	1054



SOFT/HARD REAL-TIME SYSTEMS	1055
DEFINITION OF REAL TIME EMBEDDED SYSTEMS	1056
KEY CHARACTERISTICS OF AN RTOS	1058
RTOS SCHEDULING CLASSIFICATION	1059
REAL-TIME SCHEDULING	1061
PREEMPTIVE SCHEDULING	1063
SCHEDULING ALGORITHMS IN TIME-SHARED SYSTEM	1064
SCHEDULING - PRIORITIES	1066
ADVANTAGES:	1067
DISADVANTAGES:	1067
ADVANTAGES:	1067
DISADVANTAGES:	1067
LINUX FOR REAL TIME APPLICATIONS	1069
FIVE REAL-TIME SCHEDULING CLASSES	1070
RATE MONTONIC SCHEDULING	1072
FOREGROUND/BACKGROUND SYSTEMS	1072
HANDLING AN INTERRUPT IN RTOS	1073
PROCESS OR TASK IN RTOS	1074
SCHEDULER CONCEPTS	1079
SHARED DATA BETWEEN TASKS IN FREERTOS	1080
TASK SYNCHRONIZATION	1086
REENTRANT VS. NON-REENTRANT FUNCTIONS	1087
SYNCHRONOUS VS ASYNCHRONOUS FUNCTIONS	1091
COMMON DESIGN PROBLEMS	1092
DIFFERENCE BETWEEN STARVATION AND DEADLOCK	1100
CH10. PART2 - CREATE YOUR OWN RTOS SCHEDULER	1102
THE OBJECTIVE	1102
PREREQUISITES	1103
UML DIAGRAMS FOR THE MYRTOS PROJECT:	1124
TASKS QUEUES/BUFFERS	1126
DEBUGGING SEQUENCE	1129
FUTURE WORK	1144

CHAPTER 11. AUTOMOTIVE PROTOCOLS **1145**

ABSTRACT	1145
HERE IS A COMPARISON TABLE BETWEEN LIN, CAN, CAN FD, TTCAN, AND FLEXRAY:	1146
CH11. PART1 - CAN PROTOCOL	1147
IN-VEHICLE NETWORKS (IVN)	1147
WHY CAN BUS ?	1150



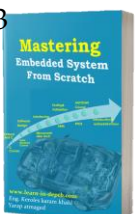
ISO 11898 (CAN SPECIFICATION)	1151
CANBUS TIMELINE	1151
CANBUS PHYSICAL LAYER	1152
CAN “CONTROLLER AREA NETWORK” SUMMARY	1154
CAN ERROR DETECTION AND FAULT CONFINEMENT	1171
BIT MONITORING	1177
BIT STUFFING	1177
CAN BUS WIRING	1178
INSTALL CANOE 16 DEMO VERSION	1183
CASE STUDY : (MONITOR CAN FRAMES)	1186
CH 11. PART2 - CAN-FD	1188
WHY THE NEED FOR CAN-FD?	1188
WHAT DOES CAN-FD BRING?	1188
CAN-FD FRAME FORMATS	1189
CAN-FD FRAME –ARBITRATION FIELD	1190
CANFD EXAMPLES	1190
CH 11. PART3 - TTCAN	1192
WHY WE NEED TIME-TRIGGERED CAN	1192
INTRODUCTION:	1192
FEATURES OF TTCAN:	1192
ARCHITECTURE OF TTCAN:	1192
BACKUP MASTERS ON TTCAN	1193
CONCLUSION:	1194
CH 11. PART4 - LIN	1195
LIN OVERVIEW	1195
INTRODUCTION:	1195
LIN PROTOCOL STACK:	1195
LIN BUS TOPOLOGY:	1196
EXAMPLE:	1196
CONCLUSION:	1197
CH 11. PART5 - FLEXRAY	1198
OVERVIEW:	1198
FLEXRAY FRAME:	1199
FLEXRAY NETWORK TOPOLOGY:	1199
FLEXRAY PROTOCOL FEATURES:	1199
FRAME FORMAT FIELDS FOR FLEXRAY	1200
THE COMMUNICATION CYCLE	1201
FLEXRAY APPLICATIONS:	1201
CONCLUSION:	1202
CH11. PART6 - AUTOMOTIVE ETHERNET	1203



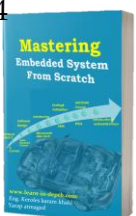
IEEE STANDARD 802.3	1203
NETWORK ARCHITECTURE LAYERS OSI : OSI OPEN SYSTEMS INTERCONNECT	1203
TCP/IP FIVE LAYER SOFTWARE MODEL TERMINOLOGY REFERENCE	1205
IFG (INTER FRAME GAP)	1206
ETHERNET ARCHITECTURE	1207
STM32F4 ETHERNET MAC	1209
ETHERNET II FRAME	1214
VIRTUAL LOCAL AREA NETWORKS (VLANs)	1216
EMBEDDED LINUX ETHERNET STACK	1216
AUTOMOTIVE ETHERNET (MOTIVATION)	1217
DATA RATE FOR MAIN AUTOMOTIVE NETWORKS	1219
WHY AUTOMOTIVE ETHERNET?	1219
AUTOMOTIVE ETHERNET	1219
AUTOMOTIVE PHYSICAL LAYER 10/1000MBPS	1221
AUTOMOTIVE ETHERNET PROTOCOLS	1221
WHAT IS TSN – IT BEGAN WITH AVB	1222
802.3BR – INTERSPERSING EXPRESS TRAFFIC (FRAME PREEMPTION)	1223
MAC MERGE SUBLAYER	1223
MFRAME FORMAT ELEMENTS	1225
AUTOMOTIVE ETHERNET STACK	1226
AUTOMOTIVE ETHERNET SWITCH	1227
ETHERNET MEDIA ACCESS CONTROLLER (MAC) ON SoC	1229

CHAPTER 12. INTRODUCTION TO AUTOSAR **1232**

ABSTRACT	1232
CH12. PART1 - AUTOSAR SW LAYERS ARCHITECTURE	1235
AUTOSAR BACKGROUND	1235
WHAT IS THE MAIN OBJECTIVE OF AUTOSAR	1235
WHAT IS THE AUTOSAR STANDARD AND WHY IS IT CREATED?	1236
AUTOSAR STANDARD	1237
NEW TERMS IN AUTOSAR	1239
START WORKING WITH AUTOSAR	1239
AUTOSAR SOFTWARE LAYERS	1244
WHAT THE AUTOSAR BASIC SOFTWARE MODULE (BSW) RESPONSIBLE FOR ?	1247
COMMUNICATION BETWEEN LAYERS	1248
EXAMPLE: COMMUNICATION BETWEEN SWCS	1248
EXAMPLE CODE: SWC IMPLEMENTATION	1248
CAN COMMUNICATION STACK	1249
AUTOSAR MEMORY STACK	1253



AUTOSAR DIAGNOSTIC STACK	1255
CONCLUSION	1256
CH12. PART2 - CONFIGURATION PARAMETERS	1257
AUTOSAR CONFIGURATION PARAMETERS CONCEPT	1258
CONFIGURATION CLASSES IN AUTOSAR	1259
CH12. PART 3 - ASW LAYER CONCEPTS	1264
USE CASE 'PEDAL MANAGEMENT' VIEW FOR ONE ECU	1264
USE CASE 'PEDAL MANAGEMENT' VIEW FOR TWO ECUS	1264
USE CASE 'FRONT-LIGHT MANAGEMENT' IN AUTOSAR	1264
AUTOSAR SW "SYSTEM DESIGN PROCESS"	1265
DESIGN SEQUENCE USING AUTOSAR METHODOLOGY	1266
INTRA- AND INTER-ECU COMMUNICATION	1270
IOC (INTER OS COMMUNICATION)	1272
AUTOSAR SWC TEMPLATE	1274
AUTOSAR SWC (IMPLEMENTATION SEQUENCE)	1277
CH12. PART 4 -ASW_SWCs TYPES	1278
APPLICATION SOFTWARE COMPONENT	1278
SERVICE SOFTWARE COMPONENT	1279
NVBLOCKSWCOMPONENT	1280
COMPLEX DRIVER SOFTWARE COMPONENT	1282
SENSORACTUATOR SOFTWARE COMPONENT	1283
ECU ABSTRACTION SWC	1283
PARAMETER SOFTWARE COMPONENT	1286
SERVICEPROXY SOFTWARE COMPONENT	1286
COMPOSITION SOFTWARE COMPONENT	1287
SOFTWARE COMPONENTS ARCHITECTURE CASE STUDY	1289
CH12. PART 5 -ASW PORTS AND INTERFACES	1291
INTRODUCTION	1291
PORTS	1291
INTERFACES TYPES	1291
SENDER RECEIVER INTERFACE	1293
RTE WRITE/READ APIs WHICH IS USED BY SENDER RECEIVER INTERFACE	1301
RTE_WRITE	1302
RTE_READ	1302
EXAMPLE	1302
SENDER-RECIEVER IMPLICIT Vs EXPLICIT	1304
AUTOSAR RTE APIs FOR THE VARIOUS SEND AND RECEIVE COMMUNICATION MODES	1305
CLIENT-SERVER	1306
CLIENT SERVER INTERFACE ARXML LAB	1310
LET US APPLY THE PORTS AND INTERFACES ON THE EABS ARCHITECTURE	1314



CH12. PART 6 -ASW COMPOSITION AND CONNECTORS	1316
COMPOSITIONS	1316
COMPOSITION AND CONNECTORS	1318
COMPOSITION-TYPE	1319
ASSEMBLY CONNECTORS	1320
CONNECTOR PROTOTYPES	1322
FLATTENING OF A HIERARCHIC SOFTWARE COMPOSITION	1324
PRACTICAL LAB (COMPOSITION)	1324
CH12. PART 7 -ASW RUNNABLES	1328
RUNNABLE ENTITIES	1328
APPLICATION PARTITIONING ON THE TARGETED MULTI-CORE PLATFORM	1331
RUNNABLE ENTITIES IMPLEMENTATION	1332
CASE STUDY 1	1334
CH12. PART 8 -ASW RTE EVENTS	1339
RTE EVENTS	1339
PRACTICAL LAB	1341
EXCLUSIVE AREAS	1345
CH12. PART 9 -RTE LAYER	1347
RUNTIME ENVIRONMENT (RTE)	1347
RTE GENERATOR	1349
RTE CONFIGURATION (EVENT TO TASK MAPPING)	1354
CH12. PART10 - AUTOSAR TYPES AND DATA CONVERSION	1357
APPLICATION DATA TYPES	1357
SWBASETYPES	1360
IMPLEMENTATION DATA TYPES	1362
LET US UNDERSTAND PLATFORMTYPES.XML	1363
CH12. PART11 - ASWL PROJECT	1365
DOORLOCK INDICATOR	1365
DOORLOCK INDICATOR AUTOSAR ASWL PROJECT	1365
NON-AUTOSAR IMPLEMENTATION	1366
AUTOSAR IMPLEMENTATION	1367
CH12. PART12 - AUTOSAR OS (OSEK-VDX STANDARED)	1376
AUTOSAR OS	1376
OSEK/VDX OVERVIEW	1377
OSEK SPECIFICATIONS	1378
OSEK OS + OIL APPLICATION	1380
TASKS IN OSEK	1382
OSEK SCHEDULING POLICY	1383
PROCESSING LEVELS	1384
TASKS' SERVICES	1385



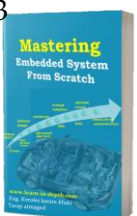
TASKS' SYNCHRONIZATION	1394
OS EVENTS	1394
EVENTS SERVICES	1395
COUNTERS	1405
DEFINING A COUNTER IN OIL	1406
USING A COUNTER IN C	1406
ALARMS	1407
DEFINING AN ALARM IN OIL	1408
USING AN ALARM IN C	1408
LAB5 (PERIODIC TASK BY ALARM)	1411
SHARED RESOURCES	1413
TASKS GROUP	1415
LAB6: RESOURCES (EXPECT WHAT THE OUTPUT AND DRAW THE TASK TIMELINE)	1416
CONFORMANCE CLASSES (CCs)	1422
HOOK ROUTINES	1423
LEVELS OF ERROR	1427
"RES_SCHEDULER" RESOURCE	1428

CHAPTER 13. INTRODUCTION TO EMBEDDED LINUX **1432**

ABSTRACT	1432
CH13. PART1 - LINUX HISTORY/MEMORY CONCEPTS	1433
HISTORICAL BACKGROUND	1433
TERMINOLOGIES	1436
LINUX VERSION HISTORY	1436
LINUX KERNEL COMPONENTS	1437
DEFINE, LINKING AND ADDRESS BINDING	1438
MEMORY MANAGEMENT CONCEPTS	1440
MEMORY MANAGEMENT UNIT (MMU)	1444
FRAGMENTATION	1447
CH13. PART2 - VIRTUAL MEMORY UPON THE PAGING TECHNIQUES	1450
DIFFERENCE BETWEEN CONTIGUOUS AND NONCONTIGUOUS MEMORY ALLOCATION	1450
PAGING	1452
SO WHAT THE ADVANTAGES OF PAGING ... ?	1457
SHARED PAGES	1457
T.L.B (TRANSLATION LOOK-ASIDE BUFFER)	1459
MULTI-LEVEL PAGE TABLE	1464
VIRTUAL MEMORY UPON THE PAGING TECHNIQUES.	1471
THRASHING	1485
SUMMARY	1487



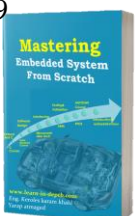
CH13. PART3 - KERNEL MODULES/PROCESS CONECPPTS	1488
BASIC I/O CONCEPTS	1488
COMPARISON - MEMORY-MAPPED VS PORT-MAPPED	1496
DMA DIRECT MEMORY ACCESS	1496
POLLING VS. INTERRUPT	1497
INTERRUPT	1497
KERNEL MODULES/PROCESS MANAGEMENT IN LINUX	1498
PROCESS MANAGEMENT IN LINUX PROCESS DESCRIPTOR	1499
A CIRCULAR DOUBLY LINKED LIST CALLED THE TASK LIST FOR THREE ELEMENTS	1502
PROCESSES VS. TASKS	1503
LINUX THREADS	1505
THE LINUX SCHEDULER	1508
LINUX MEMORY MANAGEMENT	1511
BUDDY SYSTEM	1513
ELF IMAGE	1514
MM_STRUCT FOR VIRTUAL MEMORY	1515
LINUX FILESYSTEM STRUCTURE	1518
OS: ABSTRACTION PROVIDER	1519
SYSTEM CALLS	1520
POSIX HISTORY	1521
PROCESSES VS THREADS	1522
PROCESS HIERARCHY	1524
CH13. PART3 - KERNEL MODULES/PROCESS CONECPPTS	1531
BASIC I/O CONCEPTS	1531
COMPARISON - MEMORY-MAPPED VS PORT-MAPPED	1539
DMA DIRECT MEMORY ACCESS	1540
POLLING VS. INTERRUPT	1541
INTERRUPT	1541
KERNEL MODULES/PROCESS MANAGEMENT IN LINUX	1542
PROCESS MANAGEMENT IN LINUX PROCESS DESCRIPTOR	1543
A CIRCULAR DOUBLY LINKED LIST CALLED THE TASK LIST FOR THREE ELEMENTS	1546
PROCESSES VS. TASKS	1547
LINUX THREADS	1549
THE LINUX SCHEDULER	1552
LINUX MEMORY MANAGEMENT	1555
BUDDY SYSTEM	1557
ELF IMAGE	1558
MM_STRUCT FOR VIRTUAL MEMORY	1559
LINUX FILESYSTEM STRUCTURE	1562
OS: ABSTRACTION PROVIDER	1563



SYSTEM CALLS	1564
POSIX HISTORY	1565
PROCESSES VS THREADS	1566
PROCESS HIERARCHY	1568
CH13. PART4 - PTHREAD/LINUX COMMAND	1576
LINUX ALARM	1576
READ/WRITE DIRECTORIES	1580
OPEN() /READ()/WRITE() FILES	1584
LINUX HEADERS	1588
JOINABLE AND DETACHED THREADS	1590
DETACHED THREADS	1592
PTHREAD MUTEXES	1597
SOCKET PROGRAMMING	1603
LINUX COMMAND LINE INTERFACE CLI	1607
WHAT IS A FILE ?	1608
BASICS LINUX COMMANDS	1612
WILDCARDS IN LINUX COMMAND LINE INTERFACE	1620
LINUX HELP COMMANDS	1621
COMPOSITE COMMANDS	1625
I/O REDIRECTION WITH FILES	1628
COMMON DEVICES FOR REDIRECTION	1629
SHELL SCRIPT	1631
SEARCHING TEXT (GREP COMMAND)	1632
COMPARING TEXT FILES (DIFF COMMAND)	1632
HOW TO READ DIFF OUTPUT	1633
WHAT IS PATCH?	1634
CH13. PART5 - CROSS-COMPILING TOOLCHAINS	1637
TOOL-CHAIN	1637
TOOLCHAIN COMPONENTS	1639
CREATING A STATIC LIBRARY (AR COMMAND) EXAMPLE	1646
SHARED OBJECT	1648
KERNEL HEADERS	1650
GNU CROSS-PLATFORM DEVELOPMENT TOOLCHAIN	1652
USING THE TOOLCHAIN IN MAKEFILE	1653
CH13. PART6 - LINUX BOOTLOADER	1655
LINUX BOOTLOADER: AN OVERVIEW	1655
ROLE OF A BOOTLOADER	1655
3 PHASES BOOT SEQUENCE INSIDE BOOTLOADERS	1656
BOOTING SEQUENCE EXAMPLES	1660
GENERIC BOOTLOADERS FOR EMBEDDED CPUS	1661



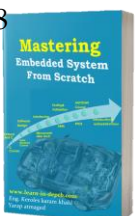
THE U-BOOT BOOTLOADER	1662
LAB1: BUILD/RUN U-BOOT FOR VEXPRESS_CA9 SoC	1663
LAB2: RUN THE TEST APPLICATION ON TOP OF U-BOOT	1673
LAB3: RUN THE TEST APPLICATION ON TOP OF U-BOOT THROUGH TFTP SERVER	1675
U-BOOT ENVIRONMENT VARIABLES	1678
LAB4:CREATE TWO ENV. (KS_BOOT_SD AND KS_BOOT_TFTP)AND SAVE THEM ON THE SDCARD	1687
LAB5:MAKE THE TFTP AUTOMATICALLY RUN, IF IT IS FAILED THEN BOOT FROM SDCARD AUTOMATICALLY.	1688
CREATING A PRE-BUILT USER ENVIRONMENT	1689
LAB6:REVERSE THE ORDER IN LAB5 BY BLOB FILE	1689
LAB7:BUILD/RUN UBOOT FOR RASPBERRY PI2	1690
LAB8: BUILD/RUN UBOOT FOR BEAGLEBONE BLACK	1695
LAB9: DEBUG THE U-BOOT CODE	1698
PORT U-BOOT TO A NEW BOARD (BASICS)	1701
CH13. PART7 - LINUX KERNEL BASICS	1707
LINUX KERNEL KEY FEATURES	1707
LINUX KERNEL MAIN ROLES	1709
LINUX KERNEL SCI (SYSTEM CALL INTERFACE)	1710
PSEUDO FILESYSTEMS	1711
LINUX Vs RTOS (ARCHITECTURE)	1712
INSIDE THE LINUX KERNEL	1712
HARDWARE ABSTRACTION LAYER (HAL)	1714
SCHEDULER	1714
FILE SYSTEM	1715
IO SUBSYSTEM	1716
NETWORKING SUBSYSTEMS	1717
LINUX USER SPACE	1718
KERNEL SOURCES	1719
BUILDING THE LINUX KERNEL	1720
KERNEL OPTION DEPENDENCIES	1725
LAB1:BUILDNG/RUNNING LINUX KERNEL ON VEXPRESS A9	1725
LAB2: BUILD/RUN LINUX ON RPI2	1728
DEVICE TREES	1731
LET US TRY TO WRITE A DEVICE TREE FROM SCRATCH TO SAMPLE MACHINE	1736
LINUX START-UP SEQUENCE	1749
MOUNTING ROOT FILE SYSTEM	1764
USER SPACE INITIALIZATION	1765
CH13. PART8 - ROOT FILE SYSTEM	1768
ROOT FILE SYSTEM	1768
MOUNTING AND UNMOUNTING A FILE SYSTEM	1769



ROOT FILESYSTEM MOUNTED BY THE KERNEL	1770
LOCATION OF THE ROOT FILESYSTEM	1771
FILE SYSTEM CONTENTS	1772
HOW TO CREATE A MINIMAL ROOT FILESYSTEM ?	1779
BUSYBOX INIT	1781
FILE SYSTEM TYPES	1785
BUILD ROOT	1786
LAB1: CREATE YOUR OWN ROOTFS BY BUILDROOT TO BE RUN ON VEXPRESS CORTEXA9	1788
KERNEL CONFIGURATION EXAMPLES (APPENDIX)	1796

CHAPTER 14. ADVANCED TOPICS **1805**

ABSTRACT	1805
CH14. PART1- ADAPTIVE AUTOSAR	1806
LIST OF DEFINITIONS	1806
OVERVIEW	1806
INTRODUCTION	1806
CLASSIC AUTOSAR VS ADAPTIVE AUTOSAR	1806
ADAPTIVE PLATFORM ARCHITECTURE	1807
EXECUTION MANAGEMENT (EM)	1808
EXECUTABLES IN ADAPTIVE AUTOSAR	1810
OVERALL STARTUP SEQUENCE	1812
MACHINE MANIFEST AND EXECUTION MANIFEST IN ADAPTIVE AUTOSAR	1814
PROCESS LIFECYCLE MANAGEMENT	1819
EXECUTION MANAGEMENT STARTUP SEQUENCE	1821
STATE MANAGEMENT	1822
MACHINE STATE	1824
STATE DEPENDENT PROCESS CONTROL	1828
INTERACTION BETWEEN STATES	1830
UPDATE AND CONFIGURATION MANAGER (UCM)	1832
SOFTWARE PACKAGE STRUCTURE	1835
CRYPTO STACK	1838
TRANSFERRING SOFTWARE PACKAGES	1840
PACKAGEMANAGERSTATUS	1841
SOFTWARE CLUSTER STATE DIAGRAM	1841
UCM MASTER	1843
CH14. PART2- INTRODUCTION TO ROS	1846
WHAT IS ROS?	1846
ROS MAIN FEATURES	1847
ROS MULTI-COMMUNICATION	1848



COMPONENTS OF ROS	1849
ROS ECOSYSTEM	1850
ROS PROJECTS EXAMPLES	1851
ROS CORE CONCEPTS	1852
ROS DISTRIBUTION RELEASES	1856
ROS SUPPORTED PLATFORMS	1858
ROS INSTALLATION	1858
RUNNING TURTLESIM_NODE IN THE TURTLESIM PACKAGE	1860
ROSRUN RQT_GRAPH RQT_GRAPH	1862
RUNNING THE MASTER	1862
ROS TCP	1865
ROS SEARCHING PACKAGES	1866
ROS FOR EMBEDDED SYSTEM	1868
GAZEBO AND RVIZ	1869
CREATING TWO NODES: A TALKER NODE AND A LISTENER NODE	1872
ROS UTILITIES	1874
<u>REFERENCES</u>	1878
<u>CONCLUSION</u>	1883

