RISETECH INTERNSHIP
Mastering STM32CubeMX5 and Cube IDE-EMBEDDED SYSTEMS

INTERN NAME:

HUZAIFA SHAHBAZ

SUPERVISIOR:

SIR USMAN AKARM

SIR SAJID GUL KHAWAJA

DATE:

26TH JULY,2024

DEPARTMENT OF COMPUTER ENGINEERING

NUST EME COLLEGE, RAWALPINDI

PERFACE

I WOULD DEDICATE MY WORK TO MY FAMILY, TEACHERS, FRIENDS AND MY SUPERVISIOR.THANK YOU SO MUCH FOR YOUR SUPPORT.

Mastering STM32CubeMX5 and Cube IDE-EMBEDDED SYSTEMS

Contents

CHAPTER#1 ABSTRACT 37
1.1ABSTRACT: 37
CHAPTER#2 INTRODUCTION 38
2.1 INTRODUCTION: 38
CHAPTER#3 OBJECTIVES 39
3.1OBJECTIVES: 39
3.1.1 UNDERSTAND STM32 MICROCONTROLLER ARCHITECTURE: 39
3.1.2 EFFICIENTLY CONFIGURE MICROCONTOLLER USING STM32CUBEMX5: 39
3.1.3 DEVELOP EMBEDDED SYSTEMS USING STM32CUBEIDE: 40
3.1.4 IMPLEMENT AND TEST PERIPHERAL FUNCTIONS: 40
3.1.5 OPTIMIZE AND DEBUG EMBEDDED SYSTEMS: 40
3.1.6 APPLY BEST PRACTICES IN EMBEDDED SYSTEMS DESGIN: 40
CHAPTER#4 HARDWARE AND SOFTWARE TOOLS 41
4.1 HARDWARE TOOLS: 41
4.2 SOFTWARE TOOLS: 41
CHAPTER#5 FUNCTIONAL REQUIREMENTS 42
5.1 FUNCTIONAL REQUIREMENTS: 42
5.1.1 MICROCONTOLLER SELECTION AND CONFIGURATION: 42
5.1.2 PIN CONFIGURATION AND MAPPING: 42
5.1.3 CLOCK SETUP AND MANAGEMENT: 42
5.1.4 PERIPHERAL INITILZATION AND SETUP: 42
5.1.5 CODE GENERATION FOR CONFIGURATIONS: 43
5.1.6 PROJECT MANAGEMENT IN STM32CUBEIDE: 43
5.1.7 INTEGRATED DEBUGGING CAPABILITIES: 43
5.1.8 FIRMWARE PROGRAMMING AND UPDATING: 43
5.1.9 MIDDLEWARE AND LIBRARY INTEGRATION: 43
5.1.10 PERFORMANCE ANALYSIS AND TESTING TOOLS: 44
CHAPTER#6 NON-FUNCTIONAL REQUIREMENTS 44

6.1NON-FUNCTIONAL REQUIREMENTS: 44
6.1.1 PERFORMANCE: 44
6.1.2 SCALABILITY: 44
6.1.3 USABILITY: 44
6.1.4 RELIABILITY: 45
6.1.5 COMPATABILITY: 45
6.1.6 SECURITY: 45
6.1.7 DOCUMENTATION AND SUPPORT: 45
6.1.8 MAINTAINABILITY: 45
6.1.9 PORTABILITY: 45
6.1.10 INTEGRATION: 46
CHAPTER#7 INTRODUCTION OF Mastering STM32CubeMX5 and Cube IDE-EMBEDDED SYSTEMS 46
7.1 INTRODUCTIONS OF Mastering STM32CubeMX5 and Cube IDE-EMBEDDED SYSTEMS: 46
CHAPTER#8 DOWNLOAD CUBEMX5 AND CUBEIDE 48
8.1 DOWNLOAD CUBEMX5 AND CUBEIDE: 48
8.1.1 STM32CUBEMX5: 49
8.1.1.1 VISIT THE STMICROELECTRONICS WEBSITE: 49
8.1.1.2 NAVIGATE TO THE DOWNLOAD SECTION: 49
8.1.1.3 SELECT THE VERSION: 49
8.1.1.4 DOWNLOAD THE INSTALLER: 49
8.1.1.5 INSTALL STM32CUBEMX5: 49
8.1.2 STM32CUBEIDE: 49
8.1.2.1 VISIT THE STMICROELECTRONICS WEBSITE: 49
8.1.2.2 NAVIGATE TO THE DOWNLOAD SECTION: 50
8.1.2.3 SELECT THE VERSION: 50
8.1.2.4 DOWNLOAD THE INSTALLER: 50
8.1.2.5 INSTALL STM32CUBEIDE: 50
8.2 ADDITIONAL STEPS: 50
8.2.1 CREATE AN ACCOUNT: 50
8.2.2 DOCUMENTATION AND SUPPORT: 50
CHAPTER#9 INSTALLING CUBEMX5 AND CUBEIDE 51
9.1 INSTALLING STM32CUBEMX5 AND STM32CUBEIDE: 51

9.1.1 INSTALLING STM32CUBEMX5: 51
9.1.1.1 DOWNLOAD THE INSTALLER: 51
9.1.1.2 RUN THE INSTALLER: 51
9.1.1.2.1 WINDOWS: 51
9.1.1.2.2 MACOS: 51
9.1.1.2.3 LINUX: 51
9.1.1.3 FOLLOW INSTALLATION INSTRUCTIONS: 52
9.1.1.4 LAUNCH STM32CUBEMX5: 52
9.1.1.5 INITIAL SETUP: 52
9.1.2 INSTALLING STM32CUBEIDE: 52
9.1.2.1 DOWNLOAD THE INSTALLER: 52
9.1.2.2 RUN THE INSTALLER: 52
9.1.2.2.1 WINDOWS: 52
9.1.2.2.2 MACOS: 52
9.1.2.2.3 LINUX: 53
9.1.2.3 FOLLOW INSTALLATION INSTRUCTIONS: 53
9.1.2.4 LAUNCH STM32CUBEIDE: 53
9.1.2.5 INITIAL SETUP: 53
9.2 POST-INSTALLATION: 53
9.2.1 UPDATE AND CONFIGURATION: 53
9.2.2 INTEGRATION: 53
CHAPTER#10 INSTALLING CUBEMX5 PACKAGES 54
10.1 INSTALLING STM32CUBEMX5 PACKAGES: 54
10.2 INSTALL AND MANAGE STM32CUBEMX5 PACKAGES: 54
10.2.1 LAUNCH STM32CUBEMX5: 54
10.2.2 ACCESS THE PACKAGE MANAGER: 54
10.2.2.1 METHOD 1: 54
10.2.2.2 METHOD 2: 54
10.2.3 INSTALLING DEVICE AND SUPPORT PACKAGES: 55
10.2.3.1 NAVIGATING THE PACKAGE MANAGER: 55
10.2.3.2 SELECTING PACKAGES: 55
10.2.3.3 DOWNLOADING PACKAGES: 55
10.2.4 MANAGING MIDDLEWARE AND ADDITIONAL COMPONENTS: 55

10.2.4.1 MIDDLEWARE PACKAGES: 55
10.2.4.2 UPDATES AND MAINTENANCE: 56
10.2.5 POST-INSTALLATION STEPS: 56
10.2.5.1 VERIFYING INSTALLATION: 56
10.2.5.2 INTERGRATING WITH STM32CUBEIDE: 56
10.2.6 ADDITIONAL STEPS: 56
10.2.6.1 INTERNET CONNECTION: 56
10.2.6.2 STORAGE SPACE: 56
CHAPTER#11 OVERVIEW OF STM32CUBEMX5 57
11.1 OVERVIEW OF STM32CUBEMX5: 57
11.2 KEY FEATURES AND FUNCTIONALITIES OF STM32 CUBE IDE: 57
11.2.1 GRAPHICAL USER INTERFACE: 57
11.2.2 MICROCONTROLLER CONFIGURATION: 58
11.2.3 PINOUT CONFIGURATION AND MAPPING: 58
11.2.4 CLOCK CONFIGURATION: 59
11.2.5 PERIPHERAL INITIALIZATION: 60
11.2.7 MIDDLEWARE INTEGRATION: 61
11.2.8 PROJECT MANAGEMENT: 62
11.2.9 PACKAGE MANAGEMENT: 62
11.2.10 DOCUMENTATION AND SUPPORT: 63
11.3 INTEGRATION WITH STM32CUBE IDE: 65
CHAPTER#12 OVERVIEW OF THE CLOCK CONFIGURATION TAB 65
12.1 OVERVIEW OF THE CLOCK CONFIGURATION TAB IN STM32CUBEMX5: 65
12.2 KEY FEATURES AND FUNCTIONALITIES OF CLOCK CONFIGURATION TAB: 65
12.2.1 CLOCK SOURCE SELECTION: 65
12.2.2 PLL CONFIGURATION: 66
12.2.3 CLOCK TREE VISUALIZATION: 67
12.2.4 AHB, APB, AND PERIPHERAL CLOCK CONFIGURATION: 68
12.2.5 REAL-TIME CLOCK (RTC) CONFIGURATION: 69
12.2.6 CLOCK SECURITY SYSTEM (CSS): 70
12.2.7 CLOCK OUTPUT AND MONITORING: 70
12.2.8 DYNAMIC RECONFIGURATION: 71
12.2.9 POWER AND PERFORMANCE CONSIDERATION: 71

12.2.10VALIDATION AND ERROR CHECKING: 72
CHAPTER#13 DEVELOPING GERNERAL PURPOSES INPUTOUTPUT DRIVERS AND FIRMWARE 74
13.1 DEVELOPING GERNERAL PURPOSES INPUTOUTPUT DRIVERS AND FIRMWARE: . 74
13.2 UNDERSTANDING GPIO CONFIGURATION: 74
13.2.1 PIN MODES: 74
13.2.1.1 INPUT MODE: 74
13.2.1.2 OUTPUT MODE: 74
13.2.1.3 ANALOG MODE: 74
13.2.1.4 ALTERNATIVE MODE: 75
13.2.2 PULL-UP/PULL-DOWN RESISTORS: 75
13.2.3 OUTPUT TYPE AND SPEED: 75
13.2.3.1 OUTPUT TYPE: 75
13.2.3.2 OUTPUT SPEED: 75
13.3 CONFIGURING GPIO PINS IN STM32CUBEMX5: 75
13.3.1 PIN ASSIGNMENT: 75
13.3.2 CONFIGURING PARAMETERS: 76
13.3.3 CODE GENERATION: 76
13.4 WRITING GPIO DRIVERS AND FIRMWARE: 76
13.4.1 INITILIZATION: 76
13.4.2 READING AND WRITING TO GPIOs: 76
13.4.2.1 READ INPUT: 76
13.4.2.2 WRITE OUTPUTS: 76
13.4.3 MANAGING GPIO STATES: 77
13.4.4 INTERRUPTS AND CALLBACKS: 77
13.4.5 ERROR HANDLING AND DEBUGGING: 77
13.5 BEST PRACTICES AND OPTIMIZATION: 77
13.5.1 POWER OPTIMIZATION: 77
13.5.2 SIGNAL INTEGRITY: 77
13.5.3 CODE MODULARITY: 77
13.5.4 DOCUMENTATION: 78
13.6 INTEGRATION AND TESTING: 78
13.6.1 INTEGRATION: 78

13.6.2 TESTING: 78
CHAPTER#14 CODING AN INPUT OUTPUT DRIVER 78
14.1 CODING AN INPUT OUTPUT DRIVER: 78
14.2 INCLUDE NECESSARY HEADERS: 78
14.2.1CODE OF CODING AN INPUT OUTPUT DRIVER: 79
14.3 INITIALIZE GPIO PINS: 79
14.3.1CODE OF CODING AN INPUT OUTPUT DRIVER: 79
14.4 IMPLEMENT I/O FUNCTIONS: 80
14.4.1 READING AN INPUT PIN: 80
14.4.1.1 CODE OF CODING AN INPUT OUTPUT DRIVER: 80
14.4.2 WRITING AN OUTPUT PIN: 80
14.4.2.1CODE OF CODING AN INPUT OUTPUT DRIVER: 80
14.4.3 TOGGLING AN OUTPUT PIN: 81
14.4.3.1 CODE OF CODING AN INPUT OUTPUT DRIVER: 81
14.5 MAIN LOOP EXAMPLE: 81
14.6 SYSTEM CLOCK CONFIGURATION: 82
14.6.1 CODE OF SYSTEM CLOCK CONFIGURATION: 82
14.7 ADDITIONAL CONSIDERATIONS: 83
14.7.1 DEBOUNCING: 83
14.7.2 POWER MANAGEMENT: 83
14.7.3 ERROR HANDLING: 83
14.7.4 INTERRUPTS: 83
14.8 CODE OF CODING AN INPUT OUTPUT DRIVER: 83
CHAPTER#15 CODING AN INPUT INTERRUPT DRIVER 85
15.1 CODING AN INPUT INTERRUPT DRIVER: 85
15.2 SETUP IN STM32CUBEMX5: 86
15.2.1 CREATE A NEW PROJECT: 86
15.2.2 CONFIGURE THE GPIO PIN FOR INPUT AND INTERRUPT: 86
15.2.3 CONFIGURE NESTED VECTOR INTERRUPT CONTROLLER: 86
15.2.4 GENERATE THE CODE: 86
15.3 IMPLEMENTING THE INTERRUPT SERVICE ROUTINE(ISR): 86
15.3.1 GENERATE CODE INITIALIZATION: 87
15.3.1.1 GPIO INITIALIZATION IN MAIN.C: 87

15.3.2 EXTI LINE 0 INTERRUPT HANDLER IN 'STM32F4XX_IT.C: 87
15.3.2.1 CODE OF EXTI LINE 0 INTERRUPT HANDLER IN 'STM32F4XX_IT.C: 88
15.3.3 GPIO EXTI CALLBACK IN 'STM32F4XX_HAL_GPIO.C': 88
15.3.3.1 CODE OF GPIO EXTI CALLBACK IN 'STM32F4XX_HAL_GPIO.C': 88
15.4 CODE OF CODING AN INPUT INTERRUPT DRIVER: 88
CHAPTER#16 CODING AN INPUT INTERRUPT DRIVER (MULTIPLE INPUTS) 91
16.1CODING AN INPUT INTERRUPT DRIVER (MULTIPLE INPUTS): 91
16.2 HARDWARE CONFIGURATION: 91
16.2.1 MICROCONTROLLER SELECTION: 91
16.2.2 PIN ASSIGNMENT: 91
16.2.3 EXTERNAL HARDWARE: 91
16.3 STM32CUBEMX CONFIGURATION: 92
16.3.1 PROJECT SETUP: 92
16.3.2 PIN CONFIGURATION: 92
16.3.3 INTERRUPT CONFIGURATION: 92
16.4 CUBE IDE DEVELOPMENT: 92
16.4.1 CODE GENERATION: 92
16.4.2 INTERRUPT SERVICE ROUTINE(ISR): 92
16.5 DEBOUNCING: 92
16.5.1 SOFTWARE DEBOUNCING: 92
16.5.2 HARDWARE DEBOUNCING: 93
16.6 TESTING AND DEBUGGING: 93
16.6.1 SIMULATION AND TESTING: 93
16.6.2 OPTIMIZATION: 93
16.7 CODE OF MAIN.H: 93
16.8 CODE OF STM32F4XX_HAL_CONF.H: 96
16.9 CODE OF STM32F4XX_IT.H: 117
16.10 CODE OF MAIN.C: 120
16.11 CODE OF STM32F4XX_HAL_MSP.C: 129
16.12 CODE OF STM32F4XX-IT.C: 134
16.13 CODE OF SYSCALLS.C: 142
16.14 CODE OF SYSTEM.C: 149
16.15 CODE OF SYSTEM_STM32F4XX.C: 152

CHAPTER#17 EMBEDDED SYSTEMS BARE-METAL EMBEDDED-C AND ASSEMBLY PROGRAMMING WITH CUBE IDE 184
17.1EMBEDDED SYSTEMS BARE-METAL EMBEDDED-C AND ASSEMBLY PROGRAMMING WITH CUBE IDE: 184
17.2 BARE-METAL EMBEDDED-C PROGRAMMING: 184
17.2.1 SETUP AND CONFIGURATION: 184
17.2.1.1 PROJECT CREATION: 184
17.2.1.2 GPIO CONFIGURATION: 185
17.2.2 MAIN PROGRAM STRUCTURE: 185
17.2.2.1 INITIALIZATION: 185
17.2.2.2 MAIN LOOP: 185
17.2.3 INTERRUPT HANDLING: 185
17.2.3.1 ISR DEFINITION: 185
17.2.3.2 INTERRUPT PRIORITIZATION: 185
17.2.4 PERIPHERAL CONTROL: 185
17.2.4.1 DIRECT REGISTER ACCESS: 185
17.2.4.2 DRIVERS AND LIBRARIES: 186
17.2.5 CODE OPTIMIZATION: 186
17.2.5.1 SIZE AND SPEED: 186
17.2.5.2 POWER MANAGEMENT: 186
17.3 CODE OF BARE-METAL C CODE: 186
17.4 ASSEMBLY PROGRAMMING WITH STM32CUBEIDE: 187
17.4.1 INLINE ASSEMBLY IN C: 187
17.4.2 WRITING ASSEMBLY FILES: 188
17.4.3 LINKING ASSEMBLY AND C: 188
17.5 CODE OF AN ASSEMBLY FUNCTION: 188
CHAPTER#18 CODING GPIO DRIVERS FROM SCRATCH USING DATASHEET INFORMATION (PART 1) 189
18.1 CODING GPIO DRIVERS FROM SCRATCH USING DATASHEET INFORMATION(PART 1): 189
18.2 STEPS TO CODE GPIO DRIVERS FROM SCRATCH: 189
18.2.1 UNDERSTAND THE HARDWARE: 189
18.2.1.1 DATASHEET: 189
18.2.1.2 REFERENCES MANUAL: 189

18.2.2 IDENTIFY GPIO REGISTERS: 189
18.2.2.1 MODE: 190
18.2.2.2 OTYPER: 190
18.2.2.3 OSPEEDER: 190
18.2.2.4 PUPDR: 190
18.2.2.5 IDR: 190
18.2.2.6 ODR: 190
18.2.2.7 BSRR: 190
18.2.2.8 LCKR: 190
18.2.3 ENABLE GPIO CLOCK: 191
18.2.4 CONFIGURE GPIO PINS: 191
18.2.5 READ/WRITE GPIO PINS: 191
18.2.6 ATOMIC SET AND RESET OPERATIONS: 191
18.2.7 LOCKING GPIO CONFIGURATION: 191
18.3 CODE OF MAIN.H: 191
18.4 CODE OF STM32F4XX_HAL_CONFIG.H: 194
18.5 CODE OF STM32F4XX_IT.H: 215
18.6 CODE OF MAIN.C: 217
18.7 CODE OF STM32F4XX_HAL_MSP.C: 227
18.8 CODE OF STM32F4XX_IT.C: 232
18.9 CODE OF SYSCALLS.C: 240
18.10 CODE OF SYSTEM.C: 247
18.11 CODE OF SYSTEM_STM32F4XX.C: 250
CHAPTER#19 CODING GPIO DRIVERS FROM SCRATCH USING DATASHEET INFORMATION (PART 2) 282
19.1CODING GPIO DRIVERS FROM SCRATCH USING DATASHEET INFORMATION (PART 2): 282
19.2 ATOMIC OPERATIONS AND BIT MANIPLUATION: 282
19.2.1 BIT SET/RESET REGISTER (BSRR): 282
19.3 HANDLING MULTIPLE PINS SIMULTANEOUSLY: 283
19.4 INTERRUPT HANDLING FOR GPIO: 283
19.4.1 CONFIGURING INTERRUPT TRIGGER TYPE: 283
19.5 ADVANCED CONFIGURATION AND ERROR HANDLING: 283
19.5.1 DEBOUNCING: 283

19.5.2 POWER MANAGEMENT: 283
19.5.3 FAULT TOLERANCE: 284
19.6 CODE OF MAIN.H: 284
19.7 CODE OF STM32F4XX_HAL_CONF.H: 287
19.8 CODE OF STM32F4XX_IT.H: 308
19.9 CODE OF MAIN.C: 310
19.10 CODE OF STM32F4XX_HAL_MSP.C: 320
19.11 CODE OF STM32F4XX_IT.C: 325
19.12 CODE OF SYSCALLS.C: 332
19.13 CODE OF SYSMEM.C: 339
19.14 CODE OF SYSTEM_STM32F4XX.C: 342
CHAPTER#20 CODING ARM ASSEMBLY APPLICATIONS IN CUBEIDE 374
20.1CODING ARM ASSEMBLY APPLICATIONS IN CUBEIDE: 374
20.2 CREATING A NEW PROJECT: 374
20.2.1 START STM32CUBEIDE: 374
20.2.2 CONFIGURE THE PROJECT: 374
20.3 WRITING ASSEMBLY CODE: 375
20.3.1 INLINE ASSEMBLY IN C/C++ FILES: 375
20.3.2 STANDALONE ASSEMBLY FILES: 375
20.3.2.1 CREATING ASSEMBLY FILE: 375
20.3.2.2 WRITING ASSEMBLY CODE: 375
20.4 INTEGRATING ASSEMBLY WITH C/C++: 375
20.5 COMPLING AND LINKING: 375
20.6 DEBUGGING AND OPTIMIZATION: 376
20.6.1 DEBUGGING: 376
20.6.2 OPTIMIZATION: 376
20.7 ASSEMBLER DIRECTIVES: 376
20.7.1 '.GLOBAL': 376
20.7.2 '.TYPE': 376
20.7.3 '.SECTION': 376
20.7.4 '.ALIGN': 376
20.8 CODE OF ARMASSEMBLY.ASM: 377
20.9 CODE OF MAIN.H: 384

20.10 CODE OF STM32F4XX_HAL_CONF.H: 387
20.11 CODE OF STM32F4XX_IT.H: 408
20.12 CODE OF MAIN.C: 410
20.12 CODE OF STM32F4XX_HAL_MSP.C: 420
20.13 CODE OF SYSCALLS.C: 427
20.14 CODE OF SYSMEM.C: 434
20.15 CODE OF SYSTEM_STM32F4XX.C: 438
20.16 CODE OF STM32F4111RETX_FLASH.LD: 469
CHAPTER#21CODING GPIO DRIVERS FROM SCRATCH IN ASSEMBLY LANGUAGE 477
21.1CODING GPIO DRIVERS FROM SCRATCH IN ASSEMBLY LANGUAGE: 477
21.2 PREREQUISITES: 478
21.2.1 ARM ARCHITECTURE KNOWLEDGE: 478
21.2.2 DATASHEET AND REFERENCES MANUAL: 478
21.2.3 DEVELOPMENT ENVIRONMENT: 478
21.3 BASIC CONCEPTS 478
21.3.1 REGISTERS: 478
21.3.2 MEMORY MAPPED I/O: 478
21.3.3 ASSEMBLY INSTRUCTIONS: 478
21.4 STEPS TO CODE GPIO DRIVERS IN ASSEMBLY: 479
21.4.1 SETTING UP THE PROJECT: 479
21.4.2 DEFINING MEMORY ADDRESSES: 479
21.4.3 ENABLING THE GPIO CLOCK: 479
21.4.4 CONFIGURING GPIO PIN MODES: 479
21.4.5 WRITING TO AND READING FROM GPIO PINS: 479
21.4.6 HANDLING MULTIPLE PINS: 479
21.4.7 INTERRUPT HANDLING: 480
21.5 CODE OF MAIN.H: 480
21.6 CODE OF STM32F4XX_HAL_CONF.H: 483
21.7 CODE OF STM32F4XX_IT.H: 504
21.8 CODE OF MAIN.C: 506
21.9 CODE OF MAIN.S: 516
21.10 CODE OF STM32F4XX_HAL_MSP.C: 517
21.11 CODE OF STM32F4XX_IT.C: 522

21.12 CODE OF SYSCALLS.C: 529
21.13 CODE OF SYTSEM_STM32F4XX.C: 536
CHAPTER#22 CODING GPIO DRIVERS FROM SCRATCH IN ASSEMBLY LANGUAGE (PART 2) 568
22.1 CODING GPIO DRIVERS FROM SCRATCH IN ASSEMBLY LANGUAGE (PART 2): 568
22.2 INTERRUPT HANDLING FOR GPIO: 568
22.2.1 INTERRUPT SERVICE ROUTINE (ISR): 568
22.3 DEBOUNCING MECHANISM: 568
22.4 MANAGING MULTIPLE GPIO PINS: 568
22.5 OPIMIZATION AND BEST PRACTICES: 569
22.5.1 USE BIT-BANDING: 569
22.5.2 MINIMIZE INTERRUPT LATENCY: 569
22.5.3 ERROR HANDLING: 569
22.5.4 CODE COMMENTS: 569
22.6 CODE OF MAIN.H: 569
22.7 CODE OF STM3F4XX_HAL_CONF.H: 572
22.8 CODE OF STM32F4XX_IT.H: 593
22.9 CODE OF MAIN.C: 595
22.10 CODE OF MAIN.S: 605
22.11 CODE OF STM32F4XX_HAL_MSP.C: 607
22.12 CODE OF STM32F4XX_IT.C: 612
22.13 CODE OF SYSCALLS.C: 620
22.14 CODE OF SYSMEM.C: 626
22.15 CODE OF SYSTEM_STM32F4XX.C: 630
CHAPTER#23 CONVERTING KEIL MDK ARM ASSEMBLY SYNTAX TO GCC ASSEMBLY SYNTAX 662
23.1CONVERTING KEIL MDK ARM ASSEMBLY SYNTAX TO GCC ASSEMBLY SYNTAX: 662
23.2 CONFIGURING GPIO PINS IN ASSEMBLY: 662
23.2.1 DEFINE GPIO REGISTER ADDRESSES: 662
23.2.2 ENABLE GPIO PINS: 662
23.2.3 SET GPIO MODES: 662
23.2.4 SET GPIO OUTPUT TYPE: 663
23.2.5 SET GPIO SPEED: 663
23.2.6 SET GPIO PULL-UP/PULL-DOWN: 663

23.3 CODE OF MAIN.H: 663
23.4 CODE OF STM32F4XX_HAL_CONF.H: 666
23.5 CODE OF STM32F4XX_IT.H: 687
23.6 CODE OF MAIN.C: 689
23.7 CODE OF MAIN.S: 699
23.8 CODE OF STM32F4XX_HAL_MSP.C: 701
23.9 CODE OF STM32F4XX_IT.C: 706
23.10 CODE OF SYSCALLS.C: 713
23.12 CODE OF SYSMEM.C: 720
23.13 CODE OF SYSTEM_STM32F4XX.C: 723
CHAPTER#24 CODING A HARDWARE TIMER TO GENERATE PRECISE DELAYS 755
24.1CODING A HARDWARE TIMER TO GENERATE PRECISE DELAYS: 755
24.2 SELECT THE TIMER: 755
24.3 CONFIGURE THE TIMER: 755
24.4 SET THE TIMER MODE: 756
24.5 CALCULATE TIMER OVERFLOW OR COMPARE VALUE: 756
24.6 ENABLE TIMER INTERRUPT: 756
24.7 START THE TIMER: 756
24.8 MONITOR OR WAIT FOR THE EVENT TIMER EVENT: 756
24.9 CODE OF MAIN.H: 756
24.10 CODE OF STM32F4XX_HAL_CONF.H: 759
24.11 CODE OF STM32F4XX_IT.H: 780
24.12 CODE OF MAIN.C: 783
24.13 CODE OF STM32F4XX_HAL_MSP.C: 794
24.14 CODE OF STM32F4XX_IT.C: 801
24.15 CODE OF SYSCALLS.C: 808
24.16 CODE OF SYSMEM.C: 815
24.17 CODE OF SYSTEM_STM32F4XX.C: 818
CHAPTER#25 CODING A HARDWARE TIMER DRIVER FOR OUTPUT COMPARE FUNCTIONALITY 850
25.1CODING A HARDWARE TIMER DRIVER FOR OUTPUT COMPARE FUNCTIONALITY: 850
25.2 STEPS TO IMPLEMENT OUTPUT COMPARE: 850
25.2.1 CONFIGURE THE TIMER CLOCK: 850

25.2.2 SET THE OUTPUT COMPARE MODE: 850
25.2.3 SET THE OUTPUT COMPARE VALUE: 851
25.5.4 ENABLE THE TIMER AND OUTPUT COMPARE CHANNEL: 851
25.5.5 HANDLE INTERRUPTS: 851
25.6 CODE OF MAIN.H: 851
25.7 CODE OF STM32F4XX_HAL_CONF.H: 854
25.8 CODE OF STMF4XX_IT.H: 875
25.9 CODE OF MAIN.C: 877
25.10 CODE OF STM32F4XX_HAL_MSP.C: 889
25.11 CODE OF STM32F4XX_IT.C: 895
25.12 CODE OF SYSCALLS.C: 903
25.13 CODE OF SYSMEM.C: 909
25.14 CODE OF SYSTEM_STM32F4XX.C: 913
CHAPTER#26 CODING A HARDWARE TIMER DRIVER FOR COUNTING EVENTS 945
26.1CODING A HARDWARE TIMER DRIVER FOR COUNTING EVENTS: 945
26.2 UNDERSTANDING THE TIME PERIPHERAL: 945
26.2.1TIME BASE MODE: 945
26.2.2 INPUT CAPTURE MODE: 945
26.2.3 OUTPUT COMPARE MODE: 945
26.2.4 PULSE WIDTH MODULATION (PWM) MODE: 945
26.3 SETTING UP TIMER: 26.3.1 ENABLING THE TIMER CLOCK: 946
26.3.2 CONFIGURING THE TIMER: 946
26.3.3 STARTING THE TIMER: 946
26.4 COUNTING EXTERNAL EVENTS: 946
26.4.1USING INPUT CAPTURE MODE: 946
26.4.2 USING EXTERNAL CLOCK MODE: 946
26.5 READING THE COUNTER VALUE: 946
26.6 HANDLING TIMER INTERRUPTS: 947
26.7 OPTIMIZATION AND CONSIDERATIONS: 947
26.7.1 PRESCALER SETTINGS: 947
26.7.2 COUNTER SIZE: 947
26.7.3 INTERRUPT HANDLING: 947
26.8 CODE OF MAIN.H: 947

26.9 CODE OF STM32F4XX_HAL_CONF.H: 950
26.10 CODE OF STM32F4XX_IT.H: 971
26.11CODE OF MAIN.C: 973
26.12 CODE OF STM32F4XX_HAL_MSP.C: 983
26.13 CODE OF STM32F4XX_IT.C: 988
26.14 CODE OF SYSCALLS.C: 995
26.15 CODE OF SYSMEM.C: 1002
26.16 CODE OF SYSTEM_STM32F4XX.C: 1005
CHAPTER#27 CODING A HARDWARE TIMER DRIVER FOR GENERATING PWM SIGNALS 1037
27.1CODING A HARDWARE TIMER DRIVER FOR GENERATING PWM SIGNALS: 1037
27.2 UNDERSTANDING THE TIMER PERIPHERAL AND PWM MODE: 1037
27.3 SETTING UP THE TIMER FOR PWM: 1038
27.3.1ENABLING THE TIMER CLOCK: 1038
27.3.2 CONFIGURE THE TIMER FOR PWM: 1038
27.3.3 SETTING THE DUTY CYCLE: 1038
27.3.4 CONFIGURING OUTPUT MODE: 1038
27.4 STARTING THE TIMER: 1038
27.5 OUTPUT PIN CONFIGURATION: 1039
27.6ADJUSTING DUTY CYCLE AND FREQUENCY: 1039
27.7 OPTIMIZATION AND CONSIDERATION: 27.7.1 AVOIDING GLITCHES: 1039
27.7.2 USING DMA: 1039
27.7.3 OUTPUT POLARITY: 1039
27.8 CODE OF MAIN.H: 1039
27.9 CODE OF STM32F4XX_HAL_CONF.H: 1042
27.10 CODE OF STM32F4XX_IT.H: 1063
27.11 CODE OF MAIN.C: 1066
27.11 CODE OF STM32F4XX_HAL_MSP.C: 1075
27.12 CODE OF STM32F4XX_IT.C: 1080
27.13 CODE OF SYSCALLS.C: 1088
27.14 CODE OF SYSMEM.C: 1094
27.15 CODE OF SYSTEM_STM32F4XX.C: 1098
CHAPTER#28 CODING A HARDWARE TIMER DRIVER FOR GENERATING PRECISE INTERRUPTS 1130

28.1CODING A HARDWARE TIMER DRIVER FORGENERATING PRECISE INTERRUPTS: 1130
28.2 UNDERSTANDING TIMER INTERRUPTS: 1130
28.3 SETTING UP TIMER: 1131
28.3.1 ENABLING THE TIMER CLOCK: 1131
28.3.2 CONFIGURING THE TIMER: 1131
28.3.2.1 CALCULATING THE INTERVAL: 1131
28.3.3 ENABLING THE TIMER: 1131
28.4 START THE TIMER: 1131
28.5 IMPLEMENTING THE INTERRUPT SERVICE ROUTINE (ISR): 1131
28.6CONFIGURING NVIC FOR TIMER INTERRUPTS: 1132
28.7 OPTIMIZATIONS AND BEST PRACTICES: 1132
28.7.1 CRITICAL SECTIONS: 1132
28.7.2 MINIMIZE ISR DURATION: 1132
28.7.3 INTERRUPT PRIORITIES: 1132
28.8 TESTING AND VERIFICATION: 1132
28.9 CODE OF MAIN.H: 1132
28.10 CODE OF STM32F4XX_HAL_CONF.H: 1135
28.11 CODE OF STM32F4XX_IT.H: 1156
28.12 CODE OF MAIN.C: 1159
28.13 CODE OF STM32F4XX_HAL_MSP.C: 1168
28.14 CODE OF STM32F4XX_IT.C: 1173
28.15 CODE OF SYSCALLS.C: 1181
28.16 CODE OF SYSMEM.C: 1187
28.17 CODE OF SYSTEM_STM32F4XX.C: 1191
CHAPTER#29 DEVELOPING ANALOG-TO-DIGITAL CONVERSION (ADC) DRIVERS AND FIRMWARE 1223
29.1DEVELOPING ANALOG-TO-DIGITAL CONVERSION (ADC) DRIVERS AND FIRMWARE: 1223
29.2 UNDERSTANDING ADC OPERATION: 1223
29.2.1 RESOLUTION: 1223
29.2.2 REFERNCE VOLTAGE(V_REF): 1223
29.2.3 SAMPLING RATE: 1224
29.3 SETTING UP THE ADC: 1224

29.3.1 ENABLING THE ADC CLOCK: 1224
29.3.2 CONFIGURING THE ADC: 1224
29.3.3 SETTING THE CONVERSION SEQUENCE: 1224
29.4 STARTING AN ADC CONVERSION: 1224
29.5 READING ADC DATA: 1224
29.6 USING INTERRUPTS OR DMA FOR ADC: 1224
29.6.1 INTERRUPTS: 1225
29.6.2 DMA: 1225
29.7 CALLIBRATION AND REFERENCE VOLTAGE: 1225
29.8 TESTING AND VALIDATION: 1225
29.8.1 ACCURACY: 1225
29.8.2 NOISE REDUCTION: 1225
29.8.3 SAMPLING RATE: 1225
CHAPTER#30 CODING AN ADC DRIVER FOR SIGNAL CONVERSION MODES 1226
30.1CODING AN ADC DRIVER FOR SIGNAL CONVERSION MODES: 1226
30.2 UNDERSTANDING SIGNAL CONVERSION MODES: 1226
30.2.1 SINGLE CONVERSION MODES: 1226
30.2.2 CONTINUOUS CONVERSION MODES: 1226
30.2.3 SCAN MODE: 1226
30.2.4 DISCONTINUOUS MODE: 1227
30.2.5 TRIGGERED CONVERSION: 1227
30.3 BASIC ADC INITILIZATION: 1227
30.3.1 ENABLING ADC CLOCK: 1227
30.3.2 CONFIGURING ADC PARAMETERS: 1227
30.4 SETTING UP CONVERSION MODES: 1227
30.4.1 SINGLE CONVERSION MODES: 1227
30.4.2 CONTINUOUS CONVERSION MODES: 1227
30.4.3 SCAN MODE (MULTI-CHANNEL CONVERSION): 1227
30.4.4 DISCONTINUOUS MODE: 1228
30.4.5 TRIGGERED CONVERSION: 1228
30.5 HANDLING ADC DATA AND INTERRUPTS: 1228
30.5.1DATA HANDLING: 1228
30.5.2 INTERRUPTS: 1228

30.6 OPTIMIZATION AND BEST PRACTICES: 1228
30.6.1MINIMIZE ISR WORKLOAD: 1228
30.6.2 USE DMA FOR LARGE DATA: 1228
30.6.3 CALIBRATION AND REFERENCES VOLTAGE: 1229
30.7 CODE OF MAIN.H: 1229
30.8 CODE OF STM32F4XX_HAL_CONF.H: 1234
30.9 CODE OF STM32F4XX_IT.H: 1255
30.10 CODE OF MAIN.C: 1257
30.11 CODE OF STM32F4XX_HAL_CONF.H: 1274
30.12 CODE OF SYSCALLS.C: 1286
30.13 CODE OF SYSMEM.C: 1293
30.14 CODE OF SYSTEM_STM32F4XX.C: 1297
CHAPTER#31 6_ADC SIGNAL CONVERSION MODE 1328
31.1 6_ADC SIGNAL CONVERSION MODE: 1328
31.1.1 SIGNAL CONVERSION MODE: 1329
31.1.1.1 USE CASE: 1329
31.1.1.2 CONFIGURATION STEPS: 1329
31.1.2 CONTINUOUS CONVERSION MODE: 1329
31.1.2.1USE CASES: 1329
31.1.2.2 CONFIGURATION STEPS: 1329
31.1.3 SCAN MODE: 1330
31.1.3.1 USE CASES: 1330
31.1.3.2 CONFIGURATION STEPS: 1330
31.1.4 DISCONTINUOUS MODE: 1330
31.1.4.1 USE CASE: 1330
31.1.4.2 CONFIGURATION STEPS: 1330
31.1.5 TRIGGERED CONVERSION MODE: 1331
31.1.5.1 USE CASES: 1331
31.1.5.2 CONFIGURATION STEPS: 1331
31.1.6 INJECTED CONVERSION MODE: 1331
31.1.6.1 USE CASE: 1331
31.1.6.2 CONFIGURATION STEPS: 1332
31.2 CODE OF MAIN.H: 1332

31.3 CODE OF STM32F4XX_HAL_CONF.H: 1337
31.4 CODE OF STM32F4XX_IT.H: 1358
31.5 CODE OF MAIN.C: 1360
31.6 CODE OF STM32F4XX_HAL_MSP.C: 1377
31.7 CODE OF STM32F4XX_IT.C: 1389
31.8 CODE OF SYSCALLS.C: 1397
31.8 CODE OF SYSMEM.C: 1404
31.9 CODE OF SYSTEM_STM32F4XX.C: 1407
CHAPTER#32 CODING AN ADC DRIVER FOR CONTINUOUS MODE CONVERSION 1439
32.1CODING AN ADC DRIVER FOR CONTINUOUS MODE CONVERSION: 1439
32.1.1 GENERAL STEPS: 1439
32.1.1.1 INITIALIZE THE ADC PERIPHERAL: 1439
31.1.1.2 CONFIGURE GPIO: 1440
31.1.1.3 START ADC CONVERSION: 1440
31.1.1.4 READ ADC DATA: 1440
31.1.1.5 HANDLE CONVERSION DATA: 1440
31.2 CODE OF ADC DRIVER: 1440
31.3 KEY CONSIDERATIONS: 1443
31.3.1 MCU-SPECIFIC LIBRARIES: 1443
31.3.2 ADC CONFIGURATION: 1443
31.3.3 POLLING VS INTERRUPT: 1443
31.3.4 ERROR HANDLING: 1443
CHAPTER#32 CODING AN ADC DRIVER FOR INTERRUPT CONVERSION MODES 1444
32.1CODING AN ADC DRIVER FOR INTERRUPT CONVERSION MODES: 1444
32.2 KEY COMPONENTS: 1444
32.2.1 ADC INITILIZATION: 1444
32.2.2 INTERRUPT CONFIGURATION: 1444
32.2.3 ADC START: 1444
32.2.4 ISR HANDLING: 1445
33.3 CODE OF ADC DRIVER FOR INTERRUPT CONVERSION MODES: 1445
33.4 KEY POINTS: 1448
33.4.1 ADC INITILIZATION: 1448
33.4.2 START ADC CONVERSION WITH INTERRUPTS: 1448

33.4.3 INTERRUPT SERVICES ROUTINE(ISR): 1448
33.4.4 NVIC CONFIGURATION: 1448
34.4.5 ERROR HANDLING: 1449
34.5 PLATFORM-SPECIFIC CONFIGURATION: 1449
34.5.1 INTERRUPT PRIORITY: 1449
34.5.2 NVIC AND ISR NAMES: 1449
CHAPTER#35 CODING AN ADC DRIVER FOR SIGNAL-CHANNEL DMA MODE 1449
35.1CODING AN ADC DRIVER FOR SIGNAL-CHANNEL DMA MODE: 1449
35.2 KEY COMPONENTS: 1449
35.2.1 ADC INITILIZATION: 1449
35.2.2 DMA INITILIZATION: 1450
35.2.3 START ADC AND DMA: 1450
35.2.4 BUFFER MANAGEMENT: 1450
35.3 CODE OF CODING AN ADC DRIVER FOR SIGNAL-CHANNEL DMA MODE: 1450
35.4 KEY POINTS: 1454
35.4.1 ADC AND DMA INITILIZATION: 1454
35.4.2 DMA CONFIGURATION: 1454
35.4.3 BUFFER MANAGEMENT: 1454
35.4.4 INTERRUPT HANDLING: 1455
35.4.5 ERROR HANDLING: 1455
35.5 PLATFORM-SPECIFIC CONSIDERATIONS: 1455
35.5.1 PERIPHERAL CLOCKS: 1455
35.5.2 DMA STREAMS/CHANNELS: 1455
35.5.3 DATA ALIGNMENT AND SIZE: 1455
CHAPTER#36 CODING AN ADC DRIVER FOR MULTI-CHANNEL DMA SCAN MODULE 1456
36.1CODING AN ADC DRIVER FOR MULTI-CHANNEL DMA SCAN MODULE: 1456
36.2 KEY COMPONENTS: 1456
36.2.1 ADC INITILIZATION: 1456
36.2.2 DMA INITILIZATION: 1456
36.2.3 START ADC AND DMA: 1456
36.2.4 BUFFER MANAGEMENT: 1456
36.3 CODE OF CODING AN ADC DRIVER FOR MULTI-CHANNEL DMA SCAN MODULE: 1457

36.4 KEY POINTS: 1461
36.4.1 ADC CONFIGURATION: 1461
36.4.2 DMA CONFIGURATION: 1461
36.4.3 CIRCULAR MODE: 1461
36.4.4 INTERRUPT HANDLING: 1462
36.4.5 BUFFER MANAGEMENT: 1462
36.5 PLATFORM-SPECIFIC CONSIDERTIONS: 1462
36.5.1 CHANNEL AND STREAM SELECTION: 1462
36.5.2 DATA ALIGNMENT AND SIZE: 1462
36.5.3 NVIC CONFIGURATION: 1462
CHAPTER#37 CODING SPI DRIVERS FOR POLLING TRANSFER MODE 1463
37.1CODING SPI DRIVERS FOR POLLING TRANSFER MODE: 1463
37.2 SETUP AND CONFIGURATION: 1463
37.2.1 DEFINE SPI CONFIGURATION PARAMETERS: 1463
37.2.2 INITIALIZE SPI PERIPHERAL: 1463
37.3 DATA TRANSFER FUNCTIONS: 1463
37.3.1 TRANSMIT DATA(POLLING): 1463
37.3.2 RECEIVE DATA(POLLING): 1463
37.4 POLLING MECHANISM: 1464
37.4.1 WAIT FOR TRANSMISSION/RECEPTION TO COMPLETE: 1464
37.4.2 ERROR HANDLING: 1464
37.5 CODE OF CODING SPI DRIVERS FOR POLLING TRANSFER MODE: 1464
CHAPTER#38 CODING AN SPI DRIVERS FOR INTERRUPT TRANSFER MODE 1466
38.1CODING AN SPI DRIVERS FOR INTERRUPT TRANSFER MODE: 1466
38.2 SETUP AND CONFIGURATION: 1466
38.2.1 SPI CONFIGURATION PARAMETERS: 1466
38.2.2 ENABLE SPI PERIPHERAL: 1466
38.2.3 ENABLE INTERRUPTS: 1466
38.3 DATA TRANSFER FUNCTION: 1467
38.3.1TRANSMIT DATA: 1467
38.3.2 RECEIVE DATA: 1467
38.4 INTERRUPT SERVICE ROUTINES(ISRs): 1467
38.4.1TRANSMIT ISR: 1467

38.4.2 RECEIVE ISR: 1467
38.4.3 ERROR HANDLING: 1467
38.5 CODE OF CODING AN SPI DRIVERS FOR INTERRUPT TRANSFER MODE: 1467
38.6 KEY POINTS: 1471
38.6.1 ISR DESIGN: 1471
38.6.2 GLOBAL VARIABLES: 1471
38.6.3 CONCURRENCY: 1471
CHAPTER#39 CODING SPI DRIVERS FOR DMA TRANSFER MODE 1471
39.1CODING SPI DRIVERS FOR DMA TRANSFER MODE: 1471
39.2 KEY COMPONENTS: 1471
39.2.1 SPI PERIPHERAL: 1471
39.2.2 DMA CONTROLLER: 1472
39.2.3 INTERRUPTS: 1472
39.3 GENERAL STEPS: 1472
39.3.1 SETUP AND CONFIGURATION: 1472
39.3.1.1 SPI CONFIGURATION: 1472
39.3.1.2 DMA CONFIGURATION: 1472
39.3.2 DATA TRANSFER FUNCTIONS: 1472
39.3.2.1 START TRANSMISSION: 1472
39.3.2.2 START RECEPTION: 1472
39.3.3 INTERRUPT HANDLING: 1473
39.3.3.1 DMA TRANSFER COMPLETE: 1473
39.4 CODE OF CODING SPI DRIVERS FOR DMA TRANSFER MODE: 1473
39.5 KEY CONSIDERATIONS: 1477
39.5.1 DMA CONFIGURATION: 1477
39.5.2 INTERRUPT HANDLING: 1477
39.5.3 BUFFER MANAGEMENT: 1477
39.5.4CONCURRENCY AND SYNCHRONIZATION: 1477
CHAPTER#40 CODING UART DRIVERS FOR POLLING TRANSFER MODE 1478
40.1CODING UART DRIVERS FOR POLLING TRANSFER MODE: 1478
40.2SETUP THE DEVLEOPMENT ENVIRONMENT: 1478
40.3 UART CONFIGURATION UAING STM32CUBEMX: 1478
40.3.1 SELECT THE PERIPHERAL: 1478

40.3.2 PIN CONFIGURATION: 1478
40.3.3 GENERATE THE INITIALIZATION CODE: 1478
40.4 CODE OF CODING UART DRIVERS FOR POLLING TRANSFER MODE: 1479
40.5 KEY POINTS: 1481
40.5.1POLLING MODE: 1481
40.5.2 ERROR HANDLING: 1481
40.5.3 TIMEOUTS: 1481
CHAPTER#41 CODING UART DRIVERS FOR INTERRUPT TRANSFER MODE 1481
41.1CODING UART DRIVERS FOR INTERRUPT TRANSFER MODE: 1481
41.2 SETUP THE DEVELOPMENT ENVIRONMENT: 1481
41.3 UART CONFIGURATION USING STM32CUBEMX: 1482
41.3.1SELECT UART PERIPHERAL: 1482
41.3.2 PIN CONFIGURATION: 1482
41.3.3 ENABLE INTERRUPTS: 1482
41.3.4 GENERATE THE INITIALIZATION CODE: 1482
41.4 WRITING THE UART INTERRUPT DRIVER CODE: 1482
41.5 CODE OF CODING UART DRIVERS FOR INTERRUPT TRANSFER MODE: 1483
41.6 EXPLANATION: 1485
41.6.1 INITILIZATION: 1485
41.6.2 INTERRUPT CALLBACKS: 1485
41.6.2.1 HAL_UART_TxCPLTCALLBACK: 1485
41.6.2.2 HAL_UART_RXCPLTCALLBACK: 1486
41.6.2.3 HAL_UART_ERRORCALLBACK: 1486
41.6.3 MAIN LOOP: 1486
41.7 KEY CONSIDERATIONS: 1486
41.7.1 INTERRUPT PRIORITIES: 1486
41.7.2 BUFFER MANAGEMENT: 1486
41.7.3 ERROR HANDLING: 1486
CHAPTER#42 CODING UART FOR DRIVERS FOR DMA TRANSFER MODE 1487
42.1CODING UART FOR DRIVERS FOR DMA TRANSFER MODE: 1487
42.2 SETUP THE DEVELOPMENT ENVIRONMENT: 1487
42.3 UART AND DMA CONFIGURATION USING STM32CUBEMX: 1487
42.3.1 SELECT UART PERIPHERAL: 1487

42.3.2 PIN CONFIGURATION: 1487
42.3.3 ENABLE DMA FOR UART: 1488
42.3.4 ENABLE NVIC FOR UART AND DMA: 1488
42.3.5 GENERATE THE INITILZATION CODE: 1488
42.4 WRITING THE UART DMA DRIVER CODE: 1488
42.5 CODE OF CODING UART FOR DRIVERS FOR DMA TRANSFER MODE: 1488
42.6 EXPALNATION: 1491
42.6.1INITILIZATION: 1491
42.6.2 DMA FUNCTIONS: 1491
42.6.3 CALLBACKS: 1491
42.6.3.1 HAL_UART_TxCPLTCALLBACK: 1491
42.6.3.2 HAL_UART_RxCPLTCALLBACK: 1491
42.6.3.3 HAL_UART_ERRORCALLBACK: 1491
42.7 KEY CONSIDERATIONS: 1491
42.7.1 BUFFER MANAGEMENT: 1491
42.7.2 INTERRUPTS AND PRIORITIES: 1492
42.7.3 ERROR HANDLING: 1492
CHAPTER#43 CODING MULTIPLE UART MODULES 1492
43.1CODING MULTIPLE UART MODULES: 1492
43.2 SETUP THE DEVELOPMENT ENVIRONMENT: 1492
43.3 UART CONFIGURATION USING STM32CUBEMX: 1492
43.3.1 SELECT UART PERIPHERALS: 1492
43.3.2 PIN CONFIGURATION: 1493
43.3.3 ENABLE INTERRUPTS: 1493
43.3.4 GENERATE THE INITILZATION CODE: 1493
43.4 WRITING THE UART DRIVER CODE: 1493
43.5 CODE OF CODING MULTIPLE UART MODULES: 1493
43.6 EXPLANATION: 1497
43.6.1INITILIZATION: 1497
43.6.1.1MX_USART1_UART_Init(): 1497
43.6.1.2MX_USART2_UART_Init(): 1497
43.6.2 INTERRUPT FUNCTIONS: 1497
43.6.2.1 HAL_UART_TRANSMIT(): 1497

43.6.2.2 HAL_UART_RECEIVE_IT(): 1497
43.6.3 CALLBACKS: 1498
43.6.3.1 HAL_UART_TxCPLTCALLBACK: 1498
43.6.3.2 HAL_UART_RxCPLTCALLBACK: 1498
43.6.3.3 HAL_UART_ERRORCALLBACK: 1498
43.7 MAIN LOOP: 1498
43.8 KEY CONSIDERATIONS: 1498
43.8.1BUFFER MANAGEMENT: 1498
43.8.2 INTERRUPTS AND PRIORITIES: 1498
43.8.3 ERROR HANDLING: 1499
CHAPTER#44 UNDERSTANDING THE USB CAPABILITIES OF DIFFERENT DEV BOARDS 1499
44.1UNDERSTANDING THE USB CAPABILITIES OF DIFFERENT DEV BOARDS: 1499
44.2 KEY ASPECTS TO CONSIDER: 1499
44.2.1USB PERIPHERAL SUPPORT: 1499
44.2.1.1 USB DEVICES: 1499
44.2.1.2 USB HOST: 1499
44.2.1.3 USB OTG(ON-THE-GO): 1500
44.2.2 STM32 FAMILIES AND USB CAPABILITIES: 1500
44.2.2.1STM32F0: 1500
44.2.2.2STM32F1: 1500
44.2.2.3 STM32F2: 1501
44.2.2.4 STM32F3: 1501
44.2.2.5 STM32F4: 1502
44.2.2.6 STM32F7: 1503
44.2.2.7STM32H7: 1504
44.2.2.8 STM32L0/L1/L4: 1505
44.2.2.9 STM32G0/G4: 1505
44.2.3 USB SPEED: 1506
44.2.3.1 FAST-SPEED(FS): 1506
44.2.3.2 HIGH-SPEED(HS): 1507
44.2.4 DEVELOPMENT TOOLS: 1507
44.2.4.1STM32CUBEMX: 1507

44.2.4.2 STM32CUBEIDE: 1508
44.2.4.3 USB LIBRARIES: 1508
44.3 EXAMPLES BOARS AND THEIR CAPABILITES: 1509
44.3.1 STM32F103C8T6(“BLUE PILL”): 1509
44.3.2 STM32F407 DISCOVERY: 1510
44.3.3 STM32F746G-DISCO: 1511
44.3.4 STM32H7 NUCLEO: 1511
44.4 EXAMPLE APPLICATIONS: 1512
44.4.1USB DEVICE: 1512
44.4.2 USB HOST: 1513
44.4.3 USB OTG: 1514
44.5 GETTING STARTED: 1514
44.5.1HARDWARE SETUP: 1514
44.5.2 SOFTWARE TOOLS: 1514
44.5.3 CONFIGURATION: 1515
44.5.4 DEVELOPMENT: 1515
CHAPTER#45 CODING USB DRIVERS FOR HUMAN INTERFACE DEVICE(HID)-MOUSE FUNCTIONALITY 1515
45.1CODING USB DRIVERS FOR HUMAN INTERFACE DEVICE(HID)-MOUSE FUNCTIONALITY: 1515
45.2 STEP 1: UNDERSTAND USB AND HID PROTOCOLS: 1515
45.2.1 USB PROTOCOL BASICS: 1515
45.2.2 HID PROTOCOL: 1516
45.3 STEP 2: SET UP YOUR DEVELOPMENT ENVIRONMENT: 1516
45.3.1DEVELOPMENT BOARD: 1516
45.3.2 DEVELOPMENT TOOLS: 1516
45.4 STEP 3: WRITE USB HOST CODE: 1516
45.4.1INITIALIZE USB HOST: 1516
45.4.2 ENUMERATE DEVICES: 1516
45.5 STEP 4: IMPLMENT HID MOUSE DRIVER: 1516
45.5.1PRAISE HID DESCRIPTOR: 1516
45.5.2 READ HID REPORTS: 1517
45.5.3 PROCESS INPUT DATA: 1517
45.6 STEP 5: INTEGRATE WITH YOUR APPLICATION: 1517

45.6.1 EVENT HANDLING: 1517
45.6.2 APPLICATION LOGIC: 1517
45.7 CODE OF CODING USB DRIVERS FOR HUMAN INTERFACE DEVICE(HID)-MOUSE FUNCTIONALITY: 1517
45.8 EXPLANATION OF THE CODE: 1519
45.8.1 USB HOST INITILIZATION: 1519
45.8.2 USER PROCESS CALLBACKS: 1520
45.8.3 MOUSE DATA PROCESSING: 1520
45.8.4 MAIN LOOP: 1520
CHAPTER#46 CODING USB DRIVERS FOR HUMAN INTERFACE DEVICE (HID)-KEYBOARD FUNCTIONALITY 1520
46.1CODING USB DRIVERS FOR HUMAN INTERFACE DEVICE (HID)-KEYBOARD FUNCTIONALITY: 1520
46.2 STEP 1: UNDERSTAND USB AND HID PROTOCOLS: 1520
46.2.1 USB PROTOCOLS BASICS: 1520
46.2.2 HID PROTOCOL: 1521
46.3 STEP 2: SET UP YOUR DEVELOPMENT ENVIRONMENT: 1521
46.3.1 DEVELOPMENT BOARDS: 1521
46.3.2 DEVELOPMENT TOOLS: 1521
46.4 STEP 3: WRITE USB HOST CODE: 1521
46.4.1INITIALIZE USB HOST: 1521
46.4.2ENUMERATE DEVICES: 1521
46.5 STEP4: IMPLEMENT HID KEYBOARD DRIVER: 1522
46.5.1 PARSE HID DESCRIPTOR: 1522
46.5.2 READ HID REPORTS: 1522
46.5.3 PROCESS INPUT DATA: 1522
46.6 STEP5: INTEGRATE WITH YOUR APPLICATION: 1522
46.6.1EVENT HANDLING: 1522
46.6.2 APPLICATION LOGIC: 1522
46.7CODE OF CODING USB DRIVERS FOR HUMAN INTERFACE DEVICE (HID)-KEYBOARD FUNCTIONALITY: 1522
46.8 EXPLANATION OF THE CODE: 1525
46.8.1USB HOST INITILIZATION: 1525
46.8.2 USER PROCES CALLBACK: 1525

46.8.3 KEYBOARD DATA PROCESSING: 1525
46.8.4 MAIN LOOP: 1525
46.9KEY PARTS OF HID KEYBOARD DRIVER: 1525
46.9.1HID DESCRIPTOR PRASING: 1525
46.9.2 INTERRUPT ENDPOINT SETUP: 1525
46.9.3 DATA REPORT DECODING: 1526
CHAPTER#47CODING USB DRIVERS FOR VIRTUAL COM PORT FUNCTIONALITY (TX ONLY) 1526
47.1CODING USB DRIVERS FOR VIRTUAL COM PORT FUNCTIONALITY (TX ONLY): 1526
47.2STEP1: UNDERSTAND USB CDC (COMMUNICATION DEVICE CONTROL) PROTCOL: 1526
47.2.1USB PROTOCOL BASICS: 1526
47.2.2 CDC CLASS PROTOCOL: 1526
47.3STEP2: SET UP YOUR DEVELOPMENT ENVIRONMENT: 1527
47.3.1DEVELOPMENT BOARD: 1527
47.3.2 DEVELOPMENT TOOLS: 1527
47.4 STEP3: WRITE THE USB DEVICE CODE: 1527
47.4.1 INITIALIZE THE USB: 1527
47.4.2 CONFIGURE ENDPOINTS: 1527
47.5 STEP 4: IMPLEMENT VIRTUAL COM PORT FUNCTIONALITY: 1527
47.5.1CDC DESCRIPTORS: 1527
47.5.2 TRANSMIT DATA: 1527
47.6 STEP5: INTEGRATE WITH YOUR APPLICATION: 1528
47.6.1TRANSMIT FUNCTION: 1528
47.6.2 APPLICATION LOGIC: 1528
47.7 CODE OF CODING USB DRIVERS FOR VIRTUAL COM PORT FUNCTIONALITY (TX ONLY): 1528
47.8 EXPLANATION OF THE CODE: 1530
47.8.1 USB DEVICE INITIALIZATION: 1530
47.8.2 TRANSMIT DATA: 1530
47.8.3 MAIN LOOP: 1530
47.9 KEY PARTS OF USB CDC DRIVER: 1530
47.9.1 CDC DESCRIPTORS: 1530
47.9.2 ENDPOINT CONFIGURATION: 1530

47.9.3 DATA TRANSMISSION: 1530
CHAPTER#48 CODING USB DRIVERS FOR VIRTUAL COM PORT FUNCTIONALITY (RX AND TX) 1531
48.1CODING USB DRIVERS FOR VIRTUAL COM PORT FUNCTIONALITY (RX AND TX): 1531
48.2 STEP1: SET UP YOUR DEVELOPMENT ENVIRONMENT: 1531
48.2.1STM32CUBEMX: 1531
48.2.2 STM32CUBEIDE: 1531
48.2.3 DEVELOPMENT BOARD: 1531
48.3 STEP2: GENERATE THE PROJECT WITH STM32CUBEMX: 1532
48.3.1 OPEN STM32CUBEMX: 1532
48.3.2 CONFIGURE USB: 1532
48.3.3 CONFIGURE USART: 1532
48.3.4 GENERATE CODE: 1532
48.4 STEP3: IMPLMENT USB CDC (COMMUNICATION DEVICE CLASS) FUNCTIONALITY: 1532
48.4.1 INCLUDE NECESSARY HEADERS: 1532
48.4.2 TRANSMIT DATA(TX): 1533
48.4.3 RECEIVE DATA(RX): 1533
48.5 STEP4: MODIFY USBD_CDC_IF.C FOR RX AND TX: 1533
48.5.1 CODE OF TRANSMIT FUNCTION: 1533
48.5.2 CODE OF RECEIVE DATA CALLBACK: 1534
48.6STEP5: MAIN APPLICATION LOGIC: 1534
48.7STEP6: COMPILE AND FLASH THE FIRMWARE: 1535
48.7.1 BUILD THE PROJECT: 1535
48.7.2 FLASH THE MICROCONTOLLER: 1536
48.8 KEY PARTS OF USB CDC DRIVER: 1536
48.8.1 CDC DESCRIPTORS: 1536
48.8.2 ENDPOINT CONFIGURATION: 1536
48.8.3 DATA TRANSMISSION AND RECEPTION: 1536
CHAPTER#49 CODING EXPERIMENTAL WITH TASKS AND PRIORITIES 1536
49.1CODING EXPERIMENTAL WITH TASKS AND PRIORITIES: 1536
49.2 DEFINE THE TASK STRUCTURE: 1536
49.3 CHOOSE A TASK MANAGEMENT METHOD: 1537

49.3.1TO-DO LISTS: 1537
49.3.2 KANBAN BOARDS: 1537
49.3.3GANTT CHARTS: 1537
49.3.4 AGILE/SCUM: 1537
49.4CODE OF CODING EXPERIMENTAL WITH TASKS AND PRIORITIES: 1537
49.5 MANAGING TASK PRIORITIES: 1541
49.5.1URGENCY: 1541
49.5.2IMPACT: 1541
49.5.3DEPENDENCIES: 1541
49.5.4 RESOURCES: 1541
49.6 USING TASK MANAGEMENT TOOLS: 1541
49.6.1TRELLO: 1542
49.6.2 JIRA: 1542
49.6.3 ASANA: 1542
49.6.4 MONDAY.COM: 1542
CHAPTER#50 CODING ACCESSING HARDWARE DRIVERS FROM MULTIPLE THERADS 1542
50.1CODING ACCESSING HARDWARE DRIVERS FROM MULTIPLE THERADS: 1542
50.2 PREREQUISTES: 1542
50.2.1STM32CUBEMX: 1542
50.2.2STM32CUBEIDE: 1543
50.3 STEP1: SET UP STM32CUBEMX: 1543
50.4 STEP2: MODIFY 'FREERTOSCONFIG.H': 1543
50.5 STEP3: IMPLEMENT THE MAIN CODE: 1543
50.5.1 CODE OF CODING ACCESSING HARDWARE DRIVERS FROM MULTIPLE THERADS: 1543
50.5.2 EXPLANATION: 1547
50.5.2.1MUTEX CREATION: 1547
50.5.2.2TASK CREATION: 1547
50.5.2.3TASK FUNCTION: 1547
50.6 STEP4: BUILD THE FLASH THE PROJECT: 1547
CHAPTER#51 CODING ACCESSING HARDWARE DRIVERS FROM MULTIPLE THREADS USING A MUTEX 1547

51.1CODING ACCESSING HARDWARE DRIVERS FROM MULTIPLE THREADS USING A MUTEX: 1547
51.2 PREREQUISTES: 1548
51.2.1STM32CUBEMX: 1548
51.2.2 STM32CUBEIDE: 1548
51.3STEP1: SET UP STM32CUBEMX: 1548
51.4 STEP2: MODIFY 'FREERTOSCONFIG.H': 1548
51.5 STEP3: IMPLEMENT THE MAIN CODE: 1548
51.5.1CODE OF CODING ACCESSING HARDWARE DRIVERS FROM MULTIPLE THREADS USING A MUTEX: 1549
51.5.2 EXPLANATION: 1553
51.5.2.1MUTEX CREATION: 1553
51.5.2.2TASK CREATION: 1554
51.5.2.3TASK FUNCTION: 1554
51.5.2.4 PERIPHERAL INITILIZATION: 1554
51.6 STEP4: BUILD THE FLASH THE PROJECT: 1554
CHAPTER#52 CODING CREATING THEADS MANUALLY 1554
52.1CODING CREATING THEADS MANUALLY: 1554
52.2 PREREQUISTES: 1555
52.2.1STM32CUBEMX: 1555
52.2.2 STM32CUBEIDE: 1555
52.3STEP1:SET UP STM32CUBEMX: 1555
52.4 STEP2: IMPLEMENT COOPERATIVE MULTITASKING: 1555
52.4.1CODE OF CODING CREATING THEADS MANUALLY: 1555
52.4.2 EXPLANATION: 1559
52.4.2.1TASK FUNCTION: 1559
52.4.2.2TASK SWITCHING: 1559
52.4.2.3 SYSTICK CONFIGURATION: 1559
52.4.2.4 MAIN LOOP: 1560
52.5 STEP3: BUILD THE FLASH THE PROJECT: 1560
CHAPTER#53 CODING SENDING NOTIFICATIONS BETWEEN THEADS 1560
53.1CODING SENDING NOTIFICATIONS BETWEEN THEADS: 1560
53.2 PREREQUISTES: 1560
53.2.1STM32CUBEMX: 1560

53.2.2 STM32CUBEIDE: 1560
53.3STEP1: SET UP STM32CUBEMX: 1561
53.4 STEP2: IMPLEMENT THE MAIN CODE: 1561
53.4.1CODE OF CODING SENDING NOTIFICATIONS BETWEEN THEADS: 1561
53.4.2EXPLANATION: 1565
53.4.2.1TASK NOTIFICATIONS: 1565
53.4.2.2TASK CREATION: 1565
53.4.2.3 TASK 1: 1565
53.4.2.4 TASK 2: 1566
53.4.2.5 PERIPHERAL INITILIZATION: 1566
53.5 STEP3: BUILD THE FLASH THE PROJECT: 1566
CHAPTER#54 CONCLUSION 1566
54CONCLUSION: 1566
REFERENCES: 1567

Mastering STM32CubeMX5 and Cube IDE-EMBEDDED SYSTEMS

Figure 7.1.1:STM32 MICROCONTOLLER 47
Figure 7.1.2: STM32CUBEMX5 48
Figure 7.1.3:STM32CUBEIDE 48
Figure 11.2.4:GRAPHICAL USER INTERFACE OF STM32CUBEMX5 57
Figure 11.2.5:MICROCONTROLLER CONFIGURATION OF STM32CUBEMX5 58
Figure 11.2.6:PINOUT CONFIGURATION AND MAPPING OF STM32CUBEMX5 59
Figure 11.2.7:CLOCK CONFIGURATION OF STM32CUBEMX5 60
Figure 11.2.8:PERIPHERAL CONFIGURATION STM32CUBEMX5 61
Figure 11.2.9:MIDDLEWARE INTEGRATION OF STM32CUBEMX5 61
Figure 11.2.10:PROJECT MANAGEMENT OF STM32CUBEMX5 62
Figure 11.2.11:PACKAGE MANAGEMENT OF STM32CUBEMX5 63
Figure 11.10.12:DOCUMENTATION AND SUPPORT OF STM32CUBEMX5 64
Figure 12.2.13:CLOCK SOURCE SELECTION OF STM32 CUBEMX5 66
Figure 12.2.14:PLL CONFIGURATION OF STM32CUBEMX5 67
Figure 12.2.15:CLOCK TREE VISUALIZATION OF STM32 CUBE MX5 68
Figure 12.2.16:AHB,APB AND PERIPHERAL CLOCK CONFIGURATION OF STM32CUBEMX5 69
Figure 12.2.17:REAL TIME CLOCK CONFIGURATION OF STM32CUBEMX5 69
Figure 12.2.18:CLOCK SECURITY SYSTEM OF STM32CUBEMX5 70
Figure 12.2.19:CLOCK OUTPUT AND MONITORING OF STM32CUBEMX5 71
Figure 12.2.20:DYNAMIC CONFIGURATION OF STM32CUBEMX5 71
Figure 12.2.21:POWER AND PERFORMANCE CONSIDERATION 72
Figure 12.2.22:ERROR CHECKING OF STM32CUBEMX5 73
Figure 12.2.23:VALIDATION OF SM32CUBEMX5 73
Figure 44.2.24:STM32F0 1500
Figure 44.2.25:STM32F1 1501
Figure 44.2.26:STM32F2 1501
Figure 44.2.27:STM32F3 1502
Figure 44.2.28:STM32F4 1503
Fi0gure 44.2.29:STM32F7 1504
Figure 44.2.30:STM32H7 1505
Figure 44.2.31:STM32L0/L1/L4 1505
Figure 44.2.32:STM32G0/G4 1505
Figure 44.2.33:USB FAST SPEED 1506
Figure 44.2.34:HIGH-SPEED USB 1507
Figure 44.2.35:STM32CUBEMX 1508
Figure 44.2.36:STM32CUBEIDE 1508
Figure 44.2.37:USB LIBRARIES 1509
Figure 44.3.38:STM32F103C8T6 1510
Figure 44.3.39:STM32F407 DISCOVERY 1510

Figure 44.3.40:STM32F746G-DISCO 1511
Figure 44.3.41:STM32H7 NUCLEO 1512
Figure 44.4.42:USB DEVICE 1513
Figure 44.4.43:USB HOST: 1513
Figure 44.4.44:USB OTG 1514

Mastering STM32CubeMX5 and Cube IDE-EMBEDDED SYSTEMS

CHAPTER#1 ABSTRACT

1.1ABSTRACT:

The evolution of embedded systems has seen a dramatic increase in complexity and capability, driven by advancements in microcontroller technology. This abstract outline the core concepts and practical applications of STM32CubeMX5 and STM32CubeIDE, two critical tools for mastering the STM32 microcontroller series.STM32CubeMX5 is a graphical tool that facilitates the configuration of STM32 microcontrollers and the initialization of peripherals. It provides a user-friendly interface to configure pin assignments, clock settings, and middleware, thus streamlining the process of microcontroller setup. With its interactive configuration and code generation capabilities, STM32CubeMX5 significantly reduces development time and minimizes the risk of configuration errors.STM32CubeIDE complements STM32CubeMX5 by offering an integrated development environment tailored for STM32 microcontrollers. It combines features of the Eclipse IDE with STM32-specific tools and plugins, including support for debugging, project management, and code editing. STM32CubeIDE's integration with STM32CubeMX5 allows for seamless transition from configuration to coding, enabling developers to efficiently implement and test their embedded systems. Mastering STM32CubeMX5 and STM32CubeIDE involves understanding their interconnected functionalities and leveraging their combined capabilities for effective embedded system development. This involves familiarizing oneself with the STM32 microcontroller architecture, configuring hardware resources, generating initialization code, and utilizing debugging tools. Proficiency

in these tools not only accelerates development but also enhances the ability to create robust, high-performance embedded solutions.

CHAPTER#2 INTRODUCTION

2.1 INTRODUCTION:

Embedded systems are at the heart of modern electronics, driving innovations across a myriad of applications, from consumer electronics to industrial automation. The STM32 microcontroller series by STMicroelectronics represents a significant advancement in embedded technology, offering a wide range of performance, power, and peripheral options suitable for diverse applications. Mastering the tools used to develop STM32 microcontrollers is crucial for leveraging their full potential.STM32CubeMX5 is a pivotal tool in this ecosystem. It serves as a graphical user interface that simplifies the configuration of STM32 microcontrollers. By providing an intuitive environment for setting up peripherals, clock configurations, and middleware, STM32CubeMX5 reduces the complexity associated with initial hardware setup and accelerates the development process. Its code generation feature ensures that developers have a solid foundation upon which to build their application-specific logic. Complementing STM32CubeMX5 is STM32CubeIDE, an integrated development environment that integrates the capabilities of Eclipse with STM32-specific features. STM32CubeIDE provides a comprehensive suite of development tools, including project management, debugging, and code editing functionalities. The synergy between STM32CubeMX5 and STM32CubeIDE allows for a streamlined workflow, from microcontroller configuration to application development and debugging. Understanding and mastering these tools is essential for engineers and developers who aim to create efficient and high-performing embedded systems. The integration of STM32CubeMX5 and STM32CubeIDE into the development process not only enhances productivity but also

improves the reliability and quality of the final product. This introduction sets the stage for exploring the functionalities, benefits, and best practices associated with these tools, providing a foundation for effective embedded system design and development.

CHAPTER#3 OBJECTIVES

3.1OBJECTIVES:

The primary objectives of mastering STM32CubeMX5 and STM32CubeIDE for embedded systems development are as follows:

3.1.1 UNDERSTAND STM32 MICROCONTROLLER ARCHITECTURE:

Gain a comprehensive understanding of the STM32 microcontroller series, including its hardware architecture, peripheral options, and available performance levels. Learn how to choose the appropriate STM32 microcontroller for specific application requirements.

3.1.2 EFFICIENTLY CONFIGURE MICROCONTOLLER USING STM32CUBEMX5:

Learn to use STM32CubeMX5 to configure microcontroller settings, including pin assignments, clock configurations, and peripheral initialization. Explore the generation of initialization code and middleware setup to accelerate the development process. Understand how to manage and update configuration files and integrate third-party middleware components.

3.1.3 DEVELOP EMBEDDED SYSTEMS USING STM32CUBEIDE:

Familiarize with STM32CubeIDE's integrated development environment features, including project management, code editing, and debugging tools. Learn how to integrate STM32CubeMX5 configurations into STM32CubeIDE projects. Develop, compile, and debug application code for STM32 microcontrollers using STM32CubeIDE.

3.1.4 IMPLEMENT AND TEST PERIPHERAL FUNCTIONS:

Acquire skills to implement and test various peripheral functions such as GPIO, UART, SPI, I2C, and timers. Understand how to handle interrupts, DMA, and other advanced peripheral features.

3.1.5 OPTIMIZE AND DEBUG EMBEDDED SYSTEMS:

Learn techniques for optimizing code and improving the performance of embedded applications. Utilize STM32CubeIDE's debugging tools to diagnose and resolve issues, including using breakpoints, watch variables, and real-time debugging features.

3.1.6 APPLY BEST PRACTICES IN EMBEDDED SYSTEMS DESGIN:

Understand and apply best practices in embedded systems design, including efficient resource management, power optimization, and code modularity. Explore strategies for maintaining and evolving embedded systems over their lifecycle.

CHAPTER#4 HARDWARE AND SOFTWARE TOOLS

4.1 HARDWARE TOOLS:

1: STM32 DEVELOPMENT BOARDS

2: STM32 NUCLEO BOARDS

3: STM32 DISCOVERY KITS

4: STM32 EVALUATION BOARDS

5: PROGRAMMING TOOLS

6: ST-LINK/V2

7: JTAG/SWD ADAPTERS

8: BREADBOARDS AND PROTOTYPING KITS

9: SENSORS AND ACTUATORS

4.2 SOFTWARE TOOLS:

1: STM32 CUBEMX5

2: GRAPHICAL CONFIGURATION

3: MIDDLEWARE INTEGRATION

4:STM32 CUBE IDE

5:STM 32 CUBE PROGRAMMER

6: STM32 CUBE LIBRARY

CHAPTER#5 FUNCTIONAL REQUIREMENTS

5.1 FUNCTIONAL REQUIREMENTS:

Here are ten functional requirements for mastering STM32CubeMX5 and STM32CubeIDE:

5.1.1 MICROCONTOLLER SELECTION AND CONFIGURATION:

Users must be able to select and configure the STM32 microcontroller model appropriate for their application, including setting up memory options, core features, and peripheral configurations.

5.1.2 PIN CONFIGURATION AND MAPPING:

The tool should allow users to configure GPIO pins for different functions (input, output, analog, etc.) and enable the mapping and remapping of pins to different peripheral functions as needed.

5.1.3 CLOCK SETUP AND MANAGEMENT:

Users must be able to configure the system, peripheral, and real-time clocks, including the selection of clock sources, setting prescalers, and managing clock frequencies to meet application timing requirements.

5.1.4 PERIPHERAL INITILZATION AND SETUP:

The tool should support configuration and initialization of various peripherals such as UART, SPI, I2C, ADC, DAC, timers, and PWM, including setting up modes, interrupts, and data formats.

5.1.5 CODE GENERATION FOR CONFIGURATIONS:

The system should automatically generate initialization code based on the user's configuration settings in STM32CubeMX5, ensuring compatibility with STM32CubeIDE for seamless integration.

5.1.6 PROJECT MANAGEMENT IN STM32CUBEIDE:

Users must be able to create, manage, and organize projects in STM32CubeIDE, including setting up build configurations, adding source files, and defining project properties.

5.1.7 INTEGRATED DEBUGGING CAPABILITIES:

The environment should provide robust debugging features, including the ability to set breakpoints, step through code, monitor variables, and inspect memory in real-time.

5.1.8 FIRMWARE PROGRAMMING AND UPDATING:

The toolchain must support programming and updating firmware on STM32 microcontrollers using interfaces like ST-LINK and integrate firmware updating features within STM32CubeIDE.

5.1.9 MIDDLEWARE AND LIBRARY INTEGRATION:

Users should be able to integrate and configure middleware components such as USB, TCP/IP stacks, and real-time operating systems (RTOS), as well as manage and use STM32CubeLibrary for peripheral and middleware functionalities.

5.1.10 PERFORMANCE ANALYSIS AND TESTING TOOLS:

The system should include tools for performance analysis, such as profiling and timing measurements, to evaluate the efficiency and responsiveness of the application, along with facilities for testing and validating peripheral functionalities.

CHAPTER#6 NON-FUNCTIONAL REQUIREMENTS

6.1NON-FUNCTIONAL REQUIREMENTS:

Non-functional requirements describe the quality attributes and constraints that the system must adhere to. Here are ten non-functional requirements for mastering STM32CubeMX5 and STM32CubeIDE:

6.1.1 PERFORMANCE:

The tools must provide fast response times for configuration, code generation, and debugging operations, ensuring efficient workflow without noticeable delays.

6.1.2 SCALABILITY:

The tools should handle projects of varying sizes and complexities, from simple configurations to large-scale embedded systems, without performance degradation.

6.1.3 USABILITY:

The interface of STM32CubeMX5 and STM32CubeIDE must be intuitive and user-friendly, providing a smooth learning curve for new users and efficient workflow for experienced developers.

6.1.4 RELIABILITY:

The tools should consistently perform their intended functions without crashing or generating errors. They must provide accurate code generation and reliable debugging capabilities.

6.1.5 COMPATABILITY:

STM32CubeMX5 and STM32CubeIDE must be compatible with a wide range of STM32 microcontrollers and peripherals, as well as integrate seamlessly with other development tools and environments.

6.1.6 SECURITY:

The tools must ensure that user data, including configuration settings and source code, are securely managed and protected from unauthorized access or corruption.

6.1.7 DOCUMENTATION AND SUPPORT:

Comprehensive and up-to-date documentation, tutorials, and support resources must be available to assist users in understanding and utilizing the tools effectively.

6.1.8 MAINTAINABILITY:

The tools should be designed for ease of maintenance, including the ability to update and upgrade with new features and bug fixes without disrupting ongoing projects.

6.1.9 PORTABILITY:

STM32CubeMX5 and STM32CubeIDE must be compatible with multiple operating systems (e.g., Windows, macOS, Linux) to accommodate a diverse user base and development environments.

6.1.10 INTEGRATION:

The tools should integrate smoothly with version control systems (e.g., Git), build systems, and continuous integration/continuous deployment (CI/CD) pipelines to support collaborative development and automation.

CHAPTER#7 INTRODUCTION OF Mastering STM32CubeMX5 and Cube IDE-EMBEDDED SYSTEMS

7.1 INTRODUCTIONS OF Mastering STM32CubeMX5 and Cube IDE-EMBEDDED SYSTEMS:

The landscape of embedded systems development has become increasingly sophisticated, necessitating powerful tools to efficiently manage the complexities of modern microcontroller-based applications. The STM32 microcontroller series by STMicroelectronics is at the forefront of this evolution, offering a diverse range of high-performance, low-power microcontrollers tailored for various applications. To harness the full potential of STM32 microcontrollers, developers rely on specialized tools that streamline configuration, coding, and debugging processes.STM32CubeMX5 and STM32CubeIDE are two integral components of the STM32 development ecosystem designed to simplify and enhance the embedded system development workflow. STM32CubeMX5 is a graphical configuration tool that facilitates the initialization and setup of STM32 microcontrollers. It provides an intuitive interface for configuring hardware resources, including pin assignments, clock settings, and peripheral initialization. By generating initialization code based on user-defined configurations, STM32CubeMX5 significantly reduces the complexity and time required to prepare a microcontroller for application development. Complementing

STM32CubeMX5, STM32CubeIDE is an integrated development environment tailored for STM32 microcontrollers. Based on the Eclipse IDE, STM32CubeIDE offers a comprehensive suite of development tools, including code editing, project management, and debugging functionalities. Its seamless integration with STM32CubeMX5 ensures a smooth transition from microcontroller configuration to application development and testing. Mastering these tools involves understanding their functionalities, leveraging their combined capabilities, and applying best practices for efficient embedded system development. The integration of STM32CubeMX5 and STM32CubeIDE not only enhances productivity but also enables developers to build reliable, high-performance embedded systems. This introduction sets the foundation for exploring how STM32CubeMX5 and STM32CubeIDE can be effectively utilized to streamline development processes, from microcontroller configuration to application deployment, ultimately advancing the capabilities and efficiency of embedded systems engineering.

Figure 7.1.1:STM32 MICROCONTOLLER

Figure 7.1.2: STM32CUBEMX5

Figure 7.1.3:STM32CUBEIDE

CHAPTER#8 DOWNLOAD CUBEMX5 AND CUBEIDE

8.1 DOWNLOAD CUBEMX5 AND CUBEIDE:

To download STM32CubeMX5 and STM32CubeIDE, follow these steps:

8.1.1 STM32CUBEMX5:

8.1.1.1 VISIT THE STMICROELECTRONICS WEBSITE:

Go to the STMicroelectronics STM32CubeMX page.

8.1.1.2 NAVIGATE TO THE DOWNLOAD SECTION:

On the STM32CubeMX page, look for the "Download" or "Get Software" section.

8.1.1.3 SELECT THE VERSION:

Ensure you are downloading the latest version, which should be STM32CubeMX5 or later. The version number is typically listed in the download section.

8.1.1.4 DOWNLOAD THE INSTALLER:

Choose the appropriate installer for your operating system (Windows, macOS, or Linux). Click the download link to start the process.

8.1.1.5 INSTALL STM32CUBEMX5:

Once the download is complete, run the installer and follow the on-screen instructions to install STM32CubeMX5 on your system.

8.1.2 STM32CUBEIDE:

8.1.2.1 VISIT THE STMICROELECTRONICS WEBSITE:

Go to the STMicroelectronics STM32CubeIDE page.

8.1.2.2 NAVIGATE TO THE DOWNLOAD SECTION:

Find the "Download" or "Get Software" section on the STM32CubeIDE page.

8.1.2.3 SELECT THE VERSION:

Ensure you are downloading the latest version of STM32CubeIDE. The version information should be available in the download section.

8.1.2.4 DOWNLOAD THE INSTALLER:

Choose the appropriate installer for your operating system (Windows, macOS, or Linux). Click the download link to start the process.

8.1.2.5 INSTALL STM32CUBEIDE:

After the download is complete, run the installer and follow the on-screen instructions to install STM32CubeIDE on your system.

8.2 ADDITIONAL STEPS:

8.2.1 CREATE AN ACCOUNT:

We might need to create a free account with STMicroelectronics to access the download links.

8.2.2 DOCUMENTATION AND SUPPORT:

After installation, refer to the documentation provided with the tools for setup and usage guidance.

CHAPTER#9 INSTALLING CUBEMX5 AND CUBEIDE

9.1 INSTALLING STM32CUBEMX5 AND STM32CUBEIDE:

Here's a step-by-step guide for installing STM32CubeMX5 and STM32CubeIDE on your system:

9.1.1 INSTALLING STM32CUBEMX5:

9.1.1.1 DOWNLOAD THE INSTALLER:

Visit the STM32CubeMX download page and download the latest version of the STM32CubeMX5 installer appropriate for your operating system (Windows, macOS, or Linux).

9.1.1.2 RUN THE INSTALLER:

9.1.1.2.1 WINDOWS:

Double-click the .exe file you downloaded to start the installation.

9.1.1.2.2 MACOS:

Open the .dmg file, then drag the STM32CubeMX application to your Applications folder.

9.1.1.2.3 LINUX:

Extract the downloaded.tar.gz file and run the provided installation script in the terminal.

9.1.1.3 FOLLOW INSTALLATION INSTRUCTIONS:

Follow the on-screen prompts to complete the installation. Choose the installation directory and other preferences as needed.

9.1.1.4 LAUNCH STM32CUBEMX5:

Once installed, open STM32CubeMX5 from your applications menu or desktop shortcut.

9.1.1.5 INITIAL SETUP:

On the first launch, STM32CubeMX5 might prompt you to update or configure additional settings. Follow the instructions to complete the setup.

9.1.2 INSTALLING STM32CUBEIDE:

9.1.2.1 DOWNLOAD THE INSTALLER:

Visit the STM32CubeIDE download page and download the latest version of the STM32CubeIDE installer suitable for your operating system (Windows, macOS, or Linux).

9.1.2.2 RUN THE INSTALLER:

9.1.2.2.1 WINDOWS:

Double-click the .exe file you downloaded to start the installation.

9.1.2.2.2 MACOS:

Open the .dmg file, then drag the STM32CubeMX application to your Applications folder.

9.1.2.2.3 LINUX:

Extract the downloaded.tar.gz file and run the provided installation script in the terminal.

9.1.2.3 FOLLOW INSTALLATION INSTRUCTIONS:

Follow the prompts provided by the installer. Choose the installation directory and any additional options as needed.

9.1.2.4 LAUNCH STM32CUBEIDE:

After installation, open STM32CubeIDE from your applications menu or desktop shortcut.

9.1.2.5 INITIAL SETUP:

On first launch, STM32CubeIDE may prompt you to select a workspace directory (where your projects will be stored). Choose or create a directory and proceed with the setup.

9.2 POST-INSTALLATION:

9.2.1 UPDATE AND CONFIGURATION:

Check for any available updates for both STM32CubeMX5 and STM32CubeIDE. Updates might include important bug fixes and new features. Configure the tools as needed for your development environment, including setting up connectivity options for programming and debugging.

9.2.2 INTEGRATION:

If we have both STM32CubeMX5 and STM32CubeIDE installed, ensure they are properly integrated. STM32CubeIDE can import configurations

from STM32CubeMX5 directly, streamlining your development workflow.

CHAPTER#10 INSTALLING CUBEMX5 PACKAGES

10.1 INSTALLING STM32CUBEMX5 PACKAGES:

STM32CubeMX5 uses software packs to manage device support, middleware, and other resources. These packs are essential for developing specific STM32 microcontrollers as they contain necessary drivers, libraries, and tools.

10.2 INSTALL AND MANAGE STM32CUBEMX5 PACKAGES:

Here's how to install and manage STM32CubeMX5 packages:

10.2.1 LAUNCH STM32CUBEMX5:

Open the STM32CubeMX5 application on your system.

10.2.2 ACCESS THE PACKAGE MANAGER:

10.2.2.1 METHOD 1:

When we first launch STM32CubeMX5, it may prompt you to download packages if they are missing or need updating.

10.2.2.2 METHOD 2:

We can also manually open the Package Manager. Go to the **Help** menu and select **Manage embedded software packages**.

10.2.3 INSTALLING DEVICE AND SUPPORT PACKAGES:

10.2.3.1 NAVIGATING THE PACKAGE MANAGER:

The Package Manager window will display available device families and versions. We can see which packages are installed, available for download, or need updating.

10.2.3.2 SELECTING PACKAGES:

Browse through the list of STM32 microcontroller families and find the one that matches your development board or project requirements. Check the boxes next to the packages you wish to install or update. These include device-specific support files, middleware, and utilities.

10.2.3.3 DOWNLOADING PACKAGES:

Click the **Install Now** button or the equivalent option to download and install the selected packages.STM32CubeMX5 will connect to the STMicroelectronics server and download the necessary files. Ensure you have an internet connection during this process.

10.2.4 MANAGING MIDDLEWARE AND ADDITIONAL COMPONENTS:

10.2.4.1 MIDDLEWARE PACKAGES:

Besides device support, you can also install middleware packages such as USB, TCP/IP, file systems, and real-time operating systems (RTOS). These can be selected from the Package Manager and installed similarly to device support packages.

10.2.4.2 UPDATES AND MAINTENANCE:

Periodically check for updates to installed packages. STM32CubeMX5 may notify you of available updates, or you can manually check through the Package Manager.

10.2.5 POST-INSTALLATION STEPS:

10.2.5.1 VERIFYING INSTALLATION:

After installing the necessary packages, verify that they are available in our project setup by creating or opening a project and checking the availability of the installed packages.

10.2.5.2 INTERGRATING WITH STM32CUBEIDE:

If you are using STM32CubeIDE, ensure that it recognizes the installed packages, which is crucial for importing configurations and developing projects seamlessly.

10.2.6 ADDITIONAL STEPS:

10.2.6.1 INTERNET CONNECTION:

Ensure a stable internet connection during downloads to prevent interruptions or corrupted files.

10.2.6.2 STORAGE SPACE:

Be aware of the storage requirements for the packages, as some middleware and device support files can be sizable.

CHAPTER#11 OVERVIEW OF STM32CUBEMX5

11.1 OVERVIEW OF STM32CUBEMX5:

STM32CubeMX5 is a comprehensive graphical software configuration tool developed by STMicroelectronics for STM32 microcontrollers. It simplifies the configuration and initialization of STM32 microcontrollers and associated peripherals, enhancing the development process of embedded systems.

11.2 KEY FEATURES AND FUNCTIONALITIES OF STM32 CUBE IDE:

11.2.1 GRAPHICAL USER INTERFACE:

STM32CubeMX5 features an intuitive graphical interface that allows developers to configure STM32 microcontroller settings visually. This interface facilitates easy navigation and setup, making it accessible for both beginners and experienced developers.

Figure 11.2.4:GRAPHICAL USER INTERFACE OF STM32CUBEMX5

11.2.2 MICROCONTROLLER CONFIGURATION:

STM32CubeMX5 enables the selection and configuration of various STM32 microcontroller families and models. Developers can specify the microcontroller's features, including core settings, memory configurations, and peripheral options.

Figure 11.2.5:MICROCONTROLLER CONFIGURATION OF STM32CUBEMX5

11.2.3 PINOUT CONFIGURATION AND MAPPING:

A critical feature of STM32CubeMX5 is the pinout configuration tool, which allows users to assign functions to microcontroller pins. The tool provides a visual representation of the microcontroller's package, making it easier to map pins to peripherals and avoid conflicts.

Figure 11.2.6:PINOUT CONFIGURATION AND MAPPING OF STM32CUBEMX5

11.2.4 CLOCK CONFIGURATION:

STM32CubeMX5 includes a clock configuration tool that helps set up the system clock, peripheral clocks, and real-time clock (RTC). Users can configure clock sources, prescalers, and frequencies to optimize performance and power consumption.

Figure 11.2.7:CLOCK CONFIGURATION OF STM32CUBEMX5

11.2.5 PERIPHERAL INITIALIZATION:

The tool provides a straightforward method to initialize and configure various microcontroller peripherals such as GPIO, UART, SPI, I2C, ADC, DAC, timers, and more. It offers options for setting peripheral modes, enabling interrupts, and configuring data formats.

Figure 11.2.8:PERIPHERAL CONFIGURATION STM32CUBEMX5

11.2.7 MIDDLEWARE INTEGRATION:

After configuring the microcontroller and peripherals, STM32CubeMX5 generates initialization code and middleware configuration files. This code is structured to be easily integrated into projects within STM32CubeIDE or other development environments, streamlining the development process.

Figure 11.2.9:MIDDLEWARE INTEGRATION OF STM32CUBEMX5

11.2.8 PROJECT MANAGEMENT:

STM32CubeMX5 includes project management features, allowing developers to create, save, and manage configurations for different projects. This functionality helps in maintaining consistency and efficiency across multiple development efforts.

Figure 11.2.10:PROJECT MANAGEMENT OF STM32CUBEMX5

11.2.9 PACKAGE MANAGEMENT:

The tool includes a package manager for downloading and updating device support packages, middleware, and additional components. This ensures that developers have access to the latest features and support for new microcontrollers.

Figure 11.2.11:PACKAGE MANAGEMENT OF STM32CUBEMX5

11.2.10 DOCUMENTATION AND SUPPORT:

STM32CubeMX5 offers comprehensive documentation, including user manuals, datasheets, and example projects. It also provides online resources and community support, assisting developers in troubleshooting and maximizing the tool's capabilities.

life.augmented

UM1718
User manual

STM32CubeMX for STM32 configuration and initialization C code generation

Introduction

STM32CubeMX is a graphical tool for STM32 microcontrollers. It is part of the STMCube™ initiative (see *Section 2*) and is available either as a standalone application or as an Eclipse plug-in for integration in Integrated Development Environments (IDEs).

STM32CubeMX has the following key features:

- Easy microcontroller selection covering the whole STM32 portfolio
- Board selection from a list of STMicroelectronics boards
- **Easy microcontroller configuration** (pins, clock tree, peripherals, middleware) and generation of the corresponding initialization C code
- **Easy switching to another microcontroller** by importing a previously-saved configuration to a new MCU project
- **Easy exporting of current configuration to a compatible MCU**
- **Generation of configuration reports**
- **Generation of embedded C projects** for a selection of integrated development environment tool chains. STM32CubeMX projects include the generated initialization C code, MISRA 2004 compliant STM32 HAL drivers, the middleware stacks required for the user configuration, and all the relevant files for opening and building the project in the selected IDE.
- **Power consumption calculation** for a user-defined application sequence
- **Self-updates** allowing the user to keep STM32CubeMX up-to-date
- Download and update of STM32Cube embedded software required for user application development (see *Appendix E: STM32Cube embedded software packages* for details on the STM32Cube embedded software offer)

Although STM32CubeMX offers a user interface and generates C code compliant with STM32 MCU design and firmware solutions, users need to refer to the product technical documentation for details on actual implementations of microcontroller peripherals and firmware.

The following documents are available from *www.st.com*:

- STM32 microcontroller reference manuals and datasheets
- *STM32Cube HAL/LL driver user manuals for STM32F0 (UM1785), STM32F1 (UM1850), STM32F2 (UM1940), STM32F3 (UM1786), STM32F4 (UM1725), STM32F7 (UM1905), STM32L0 (UM1749), STM32L1 (UM1816), STM32L4/L4+ (UM1884) and STM32H7 (UM2217).*

March 2018 UM1718 Rev 25 1/319

www.st.com

Figure 11.10.12:DOCUMENTATION AND SUPPORT OF STM32CUBEMX5

11.3 INTEGRATION WITH STM32CUBE IDE:

STM32CubeMX5 seamlessly integrates with STM32CubeIDE, STMicroelectronics' integrated development environment. This integration allows for smooth transitions from configuration and code generation to coding, debugging, and testing, providing a unified development workflow.STM32CubeMX5 is a vital tool for developers working with STM32 microcontrollers, offering a streamlined and efficient path from concept to implementation in embedded system projects.

CHAPTER#12 OVERVIEW OF THE CLOCK CONFIGURATION TAB

12.1 OVERVIEW OF THE CLOCK CONFIGURATION TAB IN STM32CUBEMX5:

The Clock Configuration tab in STM32CubeMX5 is a vital component for setting up and managing the clock system of STM32 microcontrollers. This configuration is crucial as it determines the operational speed of the microcontroller's core and peripherals, impacting performance, power consumption, and timing accuracy.

12.2 KEY FEATURES AND FUNCTIONALITIES OF CLOCK CONFIGURATION TAB:

12.2.1 CLOCK SOURCE SELECTION:

The tab allows users to select the primary clock source for the system, which could be an internal oscillator (such as the High-Speed Internal (HSI) oscillator), an external crystal oscillator (High-Speed External (HSE)), or a phase-locked loop (PLL) configuration that multiplies a base frequency.

Figure 12.2.13:CLOCK SOURCE SELECTION OF STM32 CUBEMX5

12.2.2 PLL CONFIGURATION:

The Phase-Locked Loop (PLL) settings are critical for generating high-frequency clock signals from a lower-frequency source. Users can configure the input clock source, multiplication and division factors, and output frequencies. The PLL settings are often used to achieve the desired system clock frequency (SYSCLK).

Figure 12.2.14:PLL CONFIGURATION OF STM32CUBEMX5

12.2.3 CLOCK TREE VISUALIZATION:

The Clock Configuration tab provides a visual representation of the clock tree, which outlines the relationships between the various clock sources, prescalers, and derived clock signals used by different parts of the microcontroller (e.g., core, peripherals, buses). This visualization helps in understanding how different clock settings affect the overall system.

Figure 12.2.15:CLOCK TREE VISUALIZATION OF STM32 CUBE MX5

12.2.4 AHB, APB, AND PERIPHERAL CLOCK CONFIGURATION:

The tab allows for the configuration of the Advanced High-performance Bus (AHB) and Advanced Peripheral Bus (APB) clock dividers. These settings determine the clock frequency for different bus systems within the microcontroller, which in turn affects the speed at which peripherals connected to these buses operate. Users can configure the clocks for various peripherals such as timers, ADCs, DACs, UARTs, and more, either directly or through specific bus settings.

Figure 12.2.16:AHB,APB AND PERIPHERAL CLOCK CONFIGURATION OF STM32CUBEMX5

12.2.5 REAL-TIME CLOCK (RTC) CONFIGURATION:

If the microcontroller includes an RTC, the Clock Configuration tab provides options to set up the RTC clock source, which can be separate from the main system clock sources. This setup is essential for applications requiring precise timekeeping.

Figure 12.2.17:REAL TIME CLOCK CONFIGURATION OF STM32CUBEMX5

12.2.6 CLOCK SECURITY SYSTEM (CSS):

The Clock Configuration tab may include options for enabling the Clock Security System, which monitors the clock source for failures (e.g., if an external crystal oscillator stops functioning). CSS can automatically switch to a backup clock source if a failure is detected, ensuring continued operation.

Figure 12.2.18:CLOCK SECURITY SYSTEM OF STM32CUBEMX5

12.2.7 CLOCK OUTPUT AND MONITORING:

The tab provides options for routing specific clock signals to output pins, which can be useful for debugging or for providing clock signals to external devices. It also includes settings for internal clock monitoring features.

Device Name	Sync	Mute	Clock Source	Domain Status	Primary v1 Multicast	Primary v2 Multicast	Secondary v1 Multicast	Secondary v2 Multicast	Preferred Leader	Enable Sync To External
192.168.20.0/24										
Amp1-CH	■		Dante	Disabled	Follower	N/A	N/A	N/A	☐	N/A
Amp2-CH	■		Dante	N/A	Follower	N/A	N/A	N/A	☐	N/A
Stagebox2-CH	■		Dante	**Leader**	**Leader**	Disabled	N/A	N/A	☐	☐
192.168.30.0/24										
Desk-CH-Main	■		Dante	Disabled	Follower	Disabled	N/A	N/A	☐	☐
Stagebox1-CH	■		Dante	Disabled	**Leader**	Disabled	N/A	N/A	☐	☐

Figure 12.2.19:CLOCK OUTPUT AND MONITORING OF STM32CUBEMX5

12.2.8 DYNAMIC RECONFIGURATION:

Some STM32 microcontrollers support dynamic reconfiguration of clock settings, which can be configured in this tab. This feature allows the clock settings to be changed during runtime, useful for applications that need to adjust power consumption dynamically.

Figure 12.2.20:DYNAMIC CONFIGURATION OF STM32CUBEMX5

12.2.9 POWER AND PERFORMANCE CONSIDERATION:

The Clock Configuration tab includes tools and options to help optimize the balance between performance and power consumption. Users can experiment with different clock configurations to achieve the desired balance for their specific application needs.

Figure 12.2.21:POWER AND PERFORMANCE CONSIDERATION

12.2.10VALIDATION AND ERROR CHECKING:

The tab provides real-time validation of clock settings, highlighting any configuration errors or conflicts, such as exceeding maximum frequency limits for specific peripherals or buses. This feature helps ensure that the configured clock system will function correctly without causing hardware issues.

Figure 12.2.22:ERROR CHECKING OF STM32CUBEMX5

Figure 12.2.23:VALIDATION OF SM32CUBEMX5

CHAPTER#13 DEVELOPING GERNERAL PURPOSES INPUTOUTPUT DRIVERS AND FIRMWARE

13.1 DEVELOPING GERNERAL PURPOSES INPUTOUTPUT DRIVERS AND FIRMWARE:

General-Purpose Input/Output (GPIO) pins are fundamental components of microcontrollers, including the STM32 family. They are used for interfacing with various external devices, such as sensors, switches, LEDs, and other peripherals. Developing GPIO drivers and firmware involves configuring these pins, managing their states, and integrating them into broader application logic.

13.2 UNDERSTANDING GPIO CONFIGURATION:

13.2.1 PIN MODES:

13.2.1.1 INPUT MODE:

Configures the pin to read digital signals from external devices.

13.2.1.2 OUTPUT MODE:

Sets the pin to drive a signal out, typically used for controlling LEDs, relays, etc.

13.2.1.3 ANALOG MODE:

Used for analog signal applications like ADC inputs.

13.2.1.4 ALTERNATIVE MODE:

Configures the pin for specific peripheral functions like UART, SPI, I2C, etc.

13.2.2 PULL-UP/PULL-DOWN RESISTORS:

Internal pull-up or pull-down resistors can be enabled to ensure a known state when the pin is not actively driven.

13.2.3 OUTPUT TYPE AND SPEED:

13.2.3.1 OUTPUT TYPE:

Can be configured as push-pull (standard) or open-drain.

13.2.3.2 OUTPUT SPEED:

Determines the slew rate of the pin, important for controlling electromagnetic interference (EMI) and signal integrity.

13.3 CONFIGURING GPIO PINS IN STM32CUBEMX5:

STM32CubeMX5 simplifies GPIO configuration with a graphical interface. Here's how to set up GPIOs using STM32CubeMX5:

13.3.1 PIN ASSIGNMENT:

Select the microcontroller and its package in STM32CubeMX5. Assign GPIO functions to specific pins via the Pinout & Configuration tab.

13.3.2 CONFIGURING PARAMETERS:

Set the mode (input, output, analog, alternate function) for each pin. Configure the pull-up/pull-down resistors, output type, and speed. Enable interrupts if needed for input pins.

13.3.3 CODE GENERATION:

STM32CubeMX5 generates initialization code based on the configuration. This includes setting up the GPIO registers to match the chosen settings.

13.4 WRITING GPIO DRIVERS AND FIRMWARE:

13.4.1 INITILIZATION:

Use the generated code from STM32CubeMX5 as a starting point. This typically includes a function like MX_GPIO_Init() which configures the GPIO settings.

13.4.2 READING AND WRITING TO GPIOs:

13.4.2.1 READ INPUT:

Use HAL (Hardware Abstraction Layer) functions or direct register access to read the state of an input pin, e.g., HAL_GPIO_ReadPin(GPIOx, GPIO_Pin).

13.4.2.2 WRITE OUTPUTS:

Set or reset an output pin using functions like HAL_GPIO_WritePin(GPIOx, GPIO_Pin, GPIO_PinState).

13.4.3 MANAGING GPIO STATES:

Implement logic to handle GPIO states, such as toggling LEDs or reading button states. Use debounce logic for buttons to handle mechanical noise.

13.4.4 INTERRUPTS AND CALLBACKS:

Configure GPIO pins to generate interrupts on state changes (rising edge, falling edge, both). Implement ISR (Interrupt Service Routine) or callback functions to handle these interrupts.

13.4.5 ERROR HANDLING AND DEBUGGING:

Include error handling mechanisms to manage unexpected states or configuration issues. Use debugging tools and GPIO state monitoring for testing.

13.5 BEST PRACTICES AND OPTIMIZATION:

13.5.1 POWER OPTIMIZATION:

Optimize GPIO settings to reduce power consumption, especially in low-power applications. For instance, configure unused pins as analog to minimize leakage currents.

13.5.2 SIGNAL INTEGRITY:

Manage signal integrity by appropriately setting output speeds and ensuring proper PCB layout for high-speed signals.

13.5.3 CODE MODULARITY:

Write modular and reusable GPIO code, encapsulating configuration and operations in functions or classes, which makes it easier to manage and adapt the firmware for different projects.

13.5.4 DOCUMENTATION:

Thoroughly document GPIO configurations and driver implementations, including pin mappings and the purpose of each GPIO pin.

13.6 INTEGRATION AND TESTING:

13.6.1 INTEGRATION:

Integrate GPIO drivers with other system components, ensuring proper interaction with other peripherals and the main application logic.

13.6.2 TESTING:

Perform thorough testing of GPIO functionality, including checking input responsiveness, output control, and handling of edge cases or noise.

CHAPTER#14 CODING AN INPUT OUTPUT DRIVER

14.1 CODING AN INPUT OUTPUT DRIVER:

To code a basic input/output (I/O) driver for an STM32 microcontroller, you'll typically use the STM32Cube HAL (Hardware Abstraction Layer) library. This example demonstrates setting up GPIO pins for both input and output and includes basic operations such as reading an input state and toggling an output pin.

14.2 INCLUDE NECESSARY HEADERS:

First, include the necessary header files. The main ones you'll need are the HAL library headers and any specific headers for your board or microcontroller.

14.2.1CODE OF CODING AN INPUT OUTPUT DRIVER:

```
#include "stm32f4xx_hal.h"  // Replace with your specific STM32 series
#include "stm32f4xx_hal_gpio.h"
```

14.3 INITIALIZE GPIO PINS:

Set up the GPIO pins for input and output. Typically, this is done in an initialization function.

14.3.1CODE OF CODING AN INPUT OUTPUT DRIVER:

```
void MX_GPIO_Init(void) {
    GPIO_InitTypeDef GPIO_InitStruct = {0};
    // Enable the GPIO clocks (modify according to your MCU and used ports)
    __HAL_RCC_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();
    // Configure GPIO pin for output (e.g., an LED)
    GPIO_InitStruct.Pin = GPIO_PIN_5;  // Adjust pin number according to your setup
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);  // Adjust port according to your setup
// Configure GPIO pin for input (e.g., a button)
```

```
    GPIO_InitStruct.Pin = GPIO_PIN_6;  // Adjust pin number according to your setup
    GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
    GPIO_InitStruct.Pull = GPIO_PULLUP;  // Use pull-up or pull-down as needed
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);  // Adjust port according to your setup
}
```

14.4 IMPLEMENT I/O FUNCTIONS:

14.4.1 READING AN INPUT PIN:

To read the state of an input pin, use the HAL_GPIO_ReadPin() function.

14.4.1.1 CODE OF CODING AN INPUT OUTPUT DRIVER:

```
GPIO_PinState ReadButton(void) {
    return HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_6);  // Adjust according to your setup
}
```

14.4.2 WRITING AN OUTPUT PIN:

To control an output pin, use HAL_GPIO_WritePin().

14.4.2.1CODE OF CODING AN INPUT OUTPUT DRIVER:

```
void SetLED(GPIO_PinState state)
```

```
{
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, state);  // Adjust according to your setup
}
```

14.4.3 TOGGLING AN OUTPUT PIN:

We can also toggle an output pin using HAL_GPIO_TogglePin().

14.4.3.1 CODE OF CODING AN INPUT OUTPUT DRIVER:

```
void ToggleLED(void)
{
    HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);  // Adjust according to your setup
}
```

14.5 MAIN LOOP EXAMPLE:

```
int main(void)
{
    // HAL initialization
    HAL_Init();
    // Configure the system clock
    SystemClock_Config();
    // Initialize all configured peripherals
    MX_GPIO_Init();
    // Main loop
```

```
    while (1) {
        // Check if button is pressed (button state is active low)
        if (ReadButton() == GPIO_PIN_RESET) {
            // Toggle LED
            ToggleLED();
            // Simple debounce delay
            HAL_Delay(200);  // 200ms delay
        }
    }
}
```

14.6 SYSTEM CLOCK CONFIGURATION:

Ensure that our SystemClock_Config() function is properly set up to configure the system clocks. This function is typically auto-generated by STM32CubeMX.

14.6.1 CODE OF SYSTEM CLOCK CONFIGURATION:

```
void SystemClock_Config(void)
{
    // System Clock Configuration
    // This function should be generated by STM32CubeMX based on your clock settings
}
```

14.7 ADDITIONAL CONSIDERATIONS:

14.7.1 DEBOUNCING:

Buttons and mechanical switches can cause "bouncing," which means the input signal may fluctuate when pressed or released. Implement debouncing logic or use a library to manage this.

14.7.2 POWER MANAGEMENT:

For power-sensitive applications, consider setting unused GPIO pins to analog mode to reduce power consumption.

14.7.3 ERROR HANDLING:

Implement error handling for unexpected states or hardware failures.

14.7.4 INTERRUPTS:

For more complex applications, consider using interrupts instead of polling to handle inputs, especially for time-critical operations.

14.8 CODE OF CODING AN INPUT OUTPUT DRIVER:

```
#include "main.h"

void SystemClock_Config(void);

static void MX_GPIO_Init(void);

int main(void)

{

  HAL_Init();

  SystemClock_Config();
```

```
  MX_GPIO_Init();


  while (1)
  {
    if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13) ==
GPIO_PIN_RESET)
    {
      // Toggle the LED
      HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
      // Add a delay to debounce the button
      HAL_Delay(200);
    }
  }
}
void SystemClock_Config(void)
{
  // System Clock Configuration - Set according to your setup
}
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOA_CLK_ENABLE();
```

```
  __HAL_RCC_GPIOC_CLK_ENABLE();
  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
  /*Configure GPIO pin : PA5 (LED) */
  GPIO_InitStruct.Pin = GPIO_PIN_5;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
  /*Configure GPIO pin : PC13 (Button) */
  GPIO_InitStruct.Pin = GPIO_PIN_13;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_PULLDOWN;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
}
```

CHAPTER#15 CODING AN INPUT INTERRUPT DRIVER

15.1 CODING AN INPUT INTERRUPT DRIVER:

To create an input interrupt driver for an STM32 microcontroller using STM32CubeMX5, you need to configure a GPIO pin to generate an interrupt on an input event, such as a button press. The following example demonstrates how to set up an interrupt-driven input using STM32CubeMX5 and the HAL library.

15.2 SETUP IN STM32CUBEMX5:

15.2.1 CREATE A NEW PROJECT:

Select your specific STM32 microcontroller and initialize the project.

15.2.2 CONFIGURE THE GPIO PIN FOR INPUT AND INTERRUPT:

Select the pin you want to use as an input (e.g., PA0 for a button).Set the pin mode to "GPIO_Input". Set the pin configuration to use a pull-up or pull-down resistor as needed. Enable the external interrupt (EXTI) line for the pin (e.g., EXTI0 for PA0).

15.2.3 CONFIGURE NESTED VECTOR INTERRUPT CONTROLLER:

Select the pin you want to use as an input (e.g., PA0 for a button). Set the pin mode to "GPIO_Input". Set the pin configuration to use a pull-up or pull-down resistor as needed. Enable the external interrupt (EXTI) line for the pin (e.g., EXTI0 for PA0).

15.2.4 GENERATE THE CODE:

Once configured, generate the project code, which will include the necessary initialization and HAL function calls.

15.3 IMPLEMENTING THE INTERRUPT SERVICE ROUTINE(ISR):

After generating the code, you'll implement the ISR that handles the interrupt.

15.3.1 GENERATE CODE INITIALIZATION:

STM32CubeMX will generate code in the main.c and associated stm32f4xx_it.c (or equivalent for your MCU series) files. The relevant parts include:

15.3.1.1 GPIO INITIALIZATION IN MAIN.C:

```
void MX_GPIO_Init(void)
{
   GPIO_InitTypeDef GPIO_InitStruct = {0};
   // Enable GPIO clock
   __HAL_RCC_GPIOA_CLK_ENABLE();
   // Configure GPIO pin (e.g., PA0)
   GPIO_InitStruct.Pin = GPIO_PIN_0;
   GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;  // Interrupt
on falling edge
   GPIO_InitStruct.Pull = GPIO_PULLUP;          // Enable pull-up
resistor
   HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
   // Enable and set EXTI line 0 interrupt priority
   HAL_NVIC_SetPriority(EXTI0_IRQn, 2, 0);
   HAL_NVIC_EnableIRQ(EXTI0_IRQn);
}
```

15.3.2 EXTI LINE 0 INTERRUPT HANDLER IN ‘STM32F4XX_IT.C:

STM32CubeMX generates a weakly defined handler that you can modify.

15.3.2.1 CODE OF EXTI LINE 0 INTERRUPT HANDLER IN 'STM32F4XX_IT.C:

```
void EXTI0_IRQHandler(void)
{
   // Handle the interrupt
   HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_0);
}
```

15.3.3 GPIO EXTI CALLBACK IN 'STM32F4XX_HAL_GPIO.C':

Implement the callback to handle the interrupt event.

15.3.3.1 CODE OF GPIO EXTI CALLBACK IN 'STM32F4XX_HAL_GPIO.C':

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) {
   if (GPIO_Pin == GPIO_PIN_0) {
      // Your code to handle the interrupt (e.g., toggle an LED)
      HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);  // Example:
Toggle LED on PA5
   }
}
```

15.4 CODE OF CODING AN INPUT INTERRUPT DRIVER:

```
#include "main.h"
// Main program
```

```
int main(void)
{
    // Initialize the HAL Library
    HAL_Init();
    // Configure the system clock
    SystemClock_Config();
    // Initialize all configured peripherals
    MX_GPIO_Init();
    // Main loop
    while (1) {
        // Main loop can handle other tasks or low-power modes
    }
}
// GPIO Initialization function
void MX_GPIO_Init(void) {
    GPIO_InitTypeDef GPIO_InitStruct = {0};
    // Enable GPIO clock
    __HAL_RCC_GPIOA_CLK_ENABLE();
    // Configure GPIO pin for button (PA0)
    GPIO_InitStruct.Pin = GPIO_PIN_0;
    GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;  // Interrupt
on falling edge
    GPIO_InitStruct.Pull = GPIO_PULLUP;           // Pull-up resistor
```

```
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
    // Configure GPIO pin for LED (PA5)
    GPIO_InitStruct.Pin = GPIO_PIN_5;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
    // Enable and set EXTI line 0 Interrupt to the lowest priority
    HAL_NVIC_SetPriority(EXTI0_IRQn, 2, 0);
    HAL_NVIC_EnableIRQ(EXTI0_IRQn);
}
// EXTI line0 interrupt handler
void EXTI0_IRQHandler(void) {
    HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_0);
}
// Callback function for EXTI line 0
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) {
    if (GPIO_Pin == GPIO_PIN_0) {
        // Toggle the LED (PA5) when the button (PA0) is pressed
        HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
    }
}
// System Clock Configuration function
```

```
void SystemClock_Config(void) {
    // Your system clock configuration code
    // This will be generated by STM32CubeMX
}
```

CHAPTER#16 CODING AN INPUT INTERRUPT DRIVER (MULTIPLE INPUTS)

16.1CODING AN INPUT INTERRUPT DRIVER (MULTIPLE INPUTS):

Coding an input interrupt driver with multiple inputs in embedded systems involves setting up hardware and software to handle interrupts triggered by various input sources. Here's a general outline of the steps involved, which are often covered in resources like "Mastering STM32CubeMX and Cube IDE".

16.2 HARDWARE CONFIGURATION:

16.2.1 MICROCONTROLLER SELECTION:

Choose a microcontroller that supports multiple interrupt inputs.

16.2.2 PIN ASSIGNMENT:

Map the desired input sources (e.g., buttons, sensors) to specific GPIO pins on the microcontroller.

16.2.3 EXTERNAL HARDWARE:

Ensure any necessary external components (pull-up/pull-down resistors) are correctly configured.

16.3 STM32CUBEMX CONFIGURATION:

16.3.1 PROJECT SETUP:

Create a new project and select the target microcontroller.

16.3.2 PIN CONFIGURATION:

Assign the GPIO pins as input and configure them with interrupt capabilities.

16.3.3 INTERRUPT CONFIGURATION:

Enable the external interrupt (EXTI) lines for the configured GPIO pins. Set the trigger conditions (e.g., rising edge, falling edge, or both).

16.4 CUBE IDE DEVELOPMENT:

16.4.1 CODE GENERATION:

Use STM32CubeMX to generate the initialization code, including the HAL (Hardware Abstraction Layer) libraries.

16.4.2 INTERRUPT SERVICE ROUTINE(ISR):

Implement ISRs for each EXTI line to handle the interrupts. In these routines, you can add logic to process the input events.

16.5 DEBOUNCING:

16.5.1 SOFTWARE DEBOUNCING:

Implement debouncing in software to ensure that mechanical switches or buttons don't generate multiple interrupts from a single press.

16.5.2 HARDWARE DEBOUNCING:

Alternatively, use hardware debouncing techniques if necessary.

16.6 TESTING AND DEBUGGING:

16.6.1 SIMULATION AND TESTING:

Use debugging tools and peripherals to test the interrupt handling and ensure the system behaves as expected.

16.6.2 OPTIMIZATION:

Optimize the ISRs and overall system performance as needed.

16.7 CODE OF MAIN.H:

```
/* Provide support for both ANSI and non-ANSI environments.  */

/* To get a strict ANSI C environment, define macro
__STRICT_ANSI__.  This will

   "comment out" the non-ANSI parts of the ANSI header files (non-
ANSI header

   files aren't affected).  */

#ifndef      _ANSIDECL_H_

#define      _ANSIDECL_H_

#include <newlib.h>

#include <sys/config.h>

/*  ISO C++.  */

#ifdef __cplusplus

#if !(defined(_BEGIN_STD_C) && defined(_END_STD_C))
```

```
#ifdef _HAVE_STD_CXX
#define _BEGIN_STD_C namespace std { extern "C" {
#define _END_STD_C  } }
#else
#define _BEGIN_STD_C extern "C" {
#define _END_STD_C  }
#endif
#if __GNUC_PREREQ (3, 3)
#define _NOTHROW __attribute__ ((__nothrow__))
#else
#define _NOTHROW throw()
#endif
#endif
#else
#define _BEGIN_STD_C
#define _END_STD_C
#define _NOTHROW
#endif
#ifndef _LONG_DOUBLE
#define _LONG_DOUBLE long double
#endif
/* Support gcc's __attribute__ facility.  */
#ifdef __GNUC__
```

```
#define _ATTRIBUTE(attrs) __attribute__ (attrs)
#else
#define _ATTRIBUTE(attrs)
#endif
/*  The traditional meaning of 'extern inline' for GCC is not
  to emit the function body unless the address is explicitly
  taken.  However this behaviour is changing to match the C99
  standard, which uses 'extern inline' to indicate that the
  function body *must* be emitted.  Likewise, a function declared
  without either 'extern' or 'static' defaults to extern linkage
  (C99 6.2.2p5), and the compiler may choose whether to use the
  inline version or call the extern linkage version (6.7.4p6).
  If we are using GCC, but do not have the new behaviour, we need
  to use extern inline; if we are using a new GCC with the
  C99-compatible behaviour, or a non-GCC compiler (which we will
  have to hope is C99, since there is no other way to achieve the
  effect of omitting the function if it isn't referenced) we use
  'static inline', which c99 defines to mean more-or-less the same
  as the Gnu C 'extern inline'.  */
#if defined(__GNUC__) && !defined(__GNUC_STDC_INLINE__)
/* We're using GCC, but without the new C99-compatible behaviour.  */
#define _ELIDABLE_INLINE extern __inline__ _ATTRIBUTE
((__always_inline__))
```

```
#else
/* We're using GCC in C99 mode, or an unknown compiler which
  we just have to hope obeys the C99 semantics of inline.  */
#define _ELIDABLE_INLINE static __inline__
#endif
#if __GNUC_PREREQ (3, 1)
#define _NOINLINE          __attribute__ ((__noinline__))
#define _NOINLINE_STATIC     _NOINLINE static
#else
/* On non-GNU compilers and GCC prior to version 3.1 the compiler
can't be
   trusted not to inline if it is static. */
#define _NOINLINE
#define _NOINLINE_STATIC
#endif
#endif /* _ANSIDECL_H_ */
```

16.8 CODE OF STM32F4XX_HAL_CONF.H:

```
/* USER CODE BEGIN Header */
/**
*****************************************************************
*******************
  * @file    stm32f4xx_hal_conf_template.h
  * @author  MCD Application Team
  * @brief   HAL configuration template file.
```

```
  *         This file should be copied to the application folder and renamed
  *         to stm32f4xx_hal_conf.h.

******************************************************************************
  * @attention
  *
  * Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
******************************************************************************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_HAL_CONF_H
#define __STM32F4xx_HAL_CONF_H
#ifdef __cplusplus
 extern "C" {
```

```
#endif
/* Exported types ------------------------------------------------------------*/
/* Exported constants --------------------------------------------------------*/
/* ############################ Module Selection
################################## */
/**
  * @brief This is the list of modules to be used in the HAL driver
  */
#define HAL_MODULE_ENABLED
  /* #define HAL_CRYP_MODULE_ENABLED */
/* #define HAL_ADC_MODULE_ENABLED */
/* #define HAL_CAN_MODULE_ENABLED */
/* #define HAL_CRC_MODULE_ENABLED */
/* #define HAL_CAN_LEGACY_MODULE_ENABLED */
/* #define HAL_DAC_MODULE_ENABLED */
/* #define HAL_DCMI_MODULE_ENABLED */
/* #define HAL_DMA2D_MODULE_ENABLED */
/* #define HAL_ETH_MODULE_ENABLED */
/* #define HAL_ETH_LEGACY_MODULE_ENABLED */
/* #define HAL_NAND_MODULE_ENABLED */
/* #define HAL_NOR_MODULE_ENABLED */
/* #define HAL_PCCARD_MODULE_ENABLED */
/* #define HAL_SRAM_MODULE_ENABLED */
```

```
/* #define HAL_SDRAM_MODULE_ENABLED */
/* #define HAL_HASH_MODULE_ENABLED */
/* #define HAL_I2C_MODULE_ENABLED */
/* #define HAL_I2S_MODULE_ENABLED */
/* #define HAL_IWDG_MODULE_ENABLED */
/* #define HAL_LTDC_MODULE_ENABLED */
/* #define HAL_RNG_MODULE_ENABLED */
/* #define HAL_RTC_MODULE_ENABLED */
/* #define HAL_SAI_MODULE_ENABLED */
/* #define HAL_SD_MODULE_ENABLED */
/* #define HAL_MMC_MODULE_ENABLED */
/* #define HAL_SPI_MODULE_ENABLED */
/* #define HAL_TIM_MODULE_ENABLED */
#define HAL_UART_MODULE_ENABLED
/* #define HAL_USART_MODULE_ENABLED */
/* #define HAL_IRDA_MODULE_ENABLED */
/* #define HAL_SMARTCARD_MODULE_ENABLED */
/* #define HAL_SMBUS_MODULE_ENABLED */
/* #define HAL_WWDG_MODULE_ENABLED */
/* #define HAL_PCD_MODULE_ENABLED */
/* #define HAL_HCD_MODULE_ENABLED */
/* #define HAL_DSI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
```

```
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_CEC_MODULE_ENABLED */
/* #define HAL_FMPI2C_MODULE_ENABLED */
/* #define HAL_FMPSMBUS_MODULE_ENABLED */
/* #define HAL_SPDIFRX_MODULE_ENABLED */
/* #define HAL_DFSDM_MODULE_ENABLED */
/* #define HAL_LPTIM_MODULE_ENABLED */
#define HAL_GPIO_MODULE_ENABLED
#define HAL_EXTI_MODULE_ENABLED
#define HAL_DMA_MODULE_ENABLED
#define HAL_RCC_MODULE_ENABLED
#define HAL_FLASH_MODULE_ENABLED
#define HAL_PWR_MODULE_ENABLED
#define HAL_CORTEX_MODULE_ENABLED
/* ############################ HSE/HSI Values adaptation
###################### */
/**
  * @brief Adjust the value of External High Speed oscillator (HSE)
used in your application.
  *        This value is used by the RCC HAL module to compute the
system frequency
  *        (when HSE is used as system clock source, directly or through
the PLL).
  */
```

```
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    8000000U /*!< Value of the External
oscillator in Hz */
#endif /* HSE_VALUE */
#if !defined  (HSE_STARTUP_TIMEOUT)
  #define HSE_STARTUP_TIMEOUT    100U   /*!< Time out for
HSE start up, in ms */
#endif /* HSE_STARTUP_TIMEOUT */
/**
  * @brief Internal High Speed oscillator (HSI) value.
  *        This value is used by the RCC HAL module to compute the
system frequency
  *        (when HSI is used as system clock source, directly or through the
PLL).
  */
#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000U) /*!< Value of the
Internal oscillator in Hz*/
#endif /* HSI_VALUE */
/**
  * @brief Internal Low Speed oscillator (LSI) value.
  */
#if !defined  (LSI_VALUE)
 #define LSI_VALUE  32000U       /*!< LSI Typical Value in Hz*/
```

```
#endif /* LSI_VALUE */                    /*!< Value of the Internal Low
Speed oscillator in Hz
                                  The real value may vary depending on the
variations
                                  in voltage and temperature.*/
/**
  * @brief External Low Speed oscillator (LSE) value.
  */
#if !defined  (LSE_VALUE)
 #define LSE_VALUE  32768U    /*!< Value of the External Low Speed
oscillator in Hz */
#endif /* LSE_VALUE */
#if !defined  (LSE_STARTUP_TIMEOUT)
  #define LSE_STARTUP_TIMEOUT    5000U   /*!< Time out for
LSE start up, in ms */
#endif /* LSE_STARTUP_TIMEOUT */
/**
  * @brief External clock source for I2S peripheral
  *       This value is used by the I2S HAL module to compute the I2S
clock source
  *       frequency, this source is inserted directly through I2S_CKIN
pad.
  */
#if !defined  (EXTERNAL_CLOCK_VALUE)
```

```
  #define EXTERNAL_CLOCK_VALUE    12288000U /*!< Value of
the External audio frequency in Hz*/

#endif /* EXTERNAL_CLOCK_VALUE */

/* Tip: To avoid modifying this file each time you need to use different
HSE,

   ===  you can define the HSE value in your toolchain compiler
preprocessor. */

/* ############################ System Configuration
######################### */

/**

  * @brief This is the HAL system configuration section

  */

#define  VDD_VALUE                    3300U /*!< Value of VDD in mv */

#define  TICK_INT_PRIORITY            0U   /*!< tick interrupt
priority */

#define  USE_RTOS                     0U

#define  PREFETCH_ENABLE              1U

#define  INSTRUCTION_CACHE_ENABLE     1U

#define  DATA_CACHE_ENABLE            1U

#define  USE_HAL_ADC_REGISTER_CALLBACKS        0U /*
ADC register callback disabled       */

#define  USE_HAL_CAN_REGISTER_CALLBACKS        0U /*
CAN register callback disabled       */

#define  USE_HAL_CEC_REGISTER_CALLBACKS        0U /*
CEC register callback disabled       */
```

#define USE_HAL_CRYP_REGISTER_CALLBACKS 0U /* CRYP register callback disabled */

#define USE_HAL_DAC_REGISTER_CALLBACKS 0U /* DAC register callback disabled */

#define USE_HAL_DCMI_REGISTER_CALLBACKS 0U /* DCMI register callback disabled */

#define USE_HAL_DFSDM_REGISTER_CALLBACKS 0U /* DFSDM register callback disabled */

#define USE_HAL_DMA2D_REGISTER_CALLBACKS 0U /* DMA2D register callback disabled */

#define USE_HAL_DSI_REGISTER_CALLBACKS 0U /* DSI register callback disabled */

#define USE_HAL_ETH_REGISTER_CALLBACKS 0U /* ETH register callback disabled */

#define USE_HAL_HASH_REGISTER_CALLBACKS 0U /* HASH register callback disabled */

#define USE_HAL_HCD_REGISTER_CALLBACKS 0U /* HCD register callback disabled */

#define USE_HAL_I2C_REGISTER_CALLBACKS 0U /* I2C register callback disabled */

#define USE_HAL_FMPI2C_REGISTER_CALLBACKS 0U /* FMPI2C register callback disabled */

#define USE_HAL_FMPSMBUS_REGISTER_CALLBACKS 0U /* FMPSMBUS register callback disabled */

#define USE_HAL_I2S_REGISTER_CALLBACKS 0U /* I2S register callback disabled */

#define USE_HAL_IRDA_REGISTER_CALLBACKS 0U /* IRDA register callback disabled */

#define USE_HAL_LPTIM_REGISTER_CALLBACKS 0U /* LPTIM register callback disabled */

#define USE_HAL_LTDC_REGISTER_CALLBACKS 0U /* LTDC register callback disabled */

#define USE_HAL_MMC_REGISTER_CALLBACKS 0U /* MMC register callback disabled */

#define USE_HAL_NAND_REGISTER_CALLBACKS 0U /* NAND register callback disabled */

#define USE_HAL_NOR_REGISTER_CALLBACKS 0U /* NOR register callback disabled */

#define USE_HAL_PCCARD_REGISTER_CALLBACKS 0U /* PCCARD register callback disabled */

#define USE_HAL_PCD_REGISTER_CALLBACKS 0U /* PCD register callback disabled */

#define USE_HAL_QSPI_REGISTER_CALLBACKS 0U /* QSPI register callback disabled */

#define USE_HAL_RNG_REGISTER_CALLBACKS 0U /* RNG register callback disabled */

#define USE_HAL_RTC_REGISTER_CALLBACKS 0U /* RTC register callback disabled */

#define USE_HAL_SAI_REGISTER_CALLBACKS 0U /* SAI register callback disabled */

#define USE_HAL_SD_REGISTER_CALLBACKS 0U /* SD register callback disabled */

#define USE_HAL_SMARTCARD_REGISTER_CALLBACKS 0U /* SMARTCARD register callback disabled */

#define USE_HAL_SDRAM_REGISTER_CALLBACKS 0U /* SDRAM register callback disabled */

#define USE_HAL_SRAM_REGISTER_CALLBACKS 0U /* SRAM register callback disabled */

#define USE_HAL_SPDIFRX_REGISTER_CALLBACKS 0U /* SPDIFRX register callback disabled */

#define USE_HAL_SMBUS_REGISTER_CALLBACKS 0U /* SMBUS register callback disabled */

#define USE_HAL_SPI_REGISTER_CALLBACKS 0U /* SPI register callback disabled */

#define USE_HAL_TIM_REGISTER_CALLBACKS 0U /* TIM register callback disabled */

#define USE_HAL_UART_REGISTER_CALLBACKS 0U /* UART register callback disabled */

#define USE_HAL_USART_REGISTER_CALLBACKS 0U /* USART register callback disabled */

#define USE_HAL_WWDG_REGISTER_CALLBACKS 0U /* WWDG register callback disabled */

/* ########################### Assert Selection ################################ */

/**

* @brief Uncomment the line below to expanse the "assert_param" macro in the

* HAL drivers code

*/

```
/* #define USE_FULL_ASSERT    1U */

/* ################### Ethernet peripheral configuration
###################### */

/* Section 1 : Ethernet peripheral configuration */

/* MAC ADDRESS:
MAC_ADDR0:MAC_ADDR1:MAC_ADDR2:MAC_ADDR3:MAC_A
DDR4:MAC_ADDR5 */

#define MAC_ADDR0   2U

#define MAC_ADDR1   0U

#define MAC_ADDR2   0U

#define MAC_ADDR3   0U

#define MAC_ADDR4   0U

#define MAC_ADDR5   0U

/* Definition of the Ethernet driver buffers size and count */

#define ETH_RX_BUF_SIZE             ETH_MAX_PACKET_SIZE
/* buffer size for receive             */

#define ETH_TX_BUF_SIZE             ETH_MAX_PACKET_SIZE
/* buffer size for transmit            */

#define ETH_RXBUFNB                 4U      /* 4 Rx buffers of size
ETH_RX_BUF_SIZE  */

#define ETH_TXBUFNB                 4U      /* 4 Tx buffers of size
ETH_TX_BUF_SIZE  */

/* Section 2: PHY configuration section */

/* DP83848_PHY_ADDRESS Address*/

#define DP83848_PHY_ADDRESS
```

```
/* PHY Reset delay these values are based on a 1 ms Systick interrupt*/
#define PHY_RESET_DELAY              0x000000FFU
/* PHY Configuration delay */
#define PHY_CONFIG_DELAY             0x00000FFFU
#define PHY_READ_TO                  0x0000FFFFU
#define PHY_WRITE_TO                 0x0000FFFFU
/* Section 3: Common PHY Registers */
#define PHY_BCR                      ((uint16_t)0x0000U)    /*!<
Transceiver Basic Control Register   */
#define PHY_BSR                      ((uint16_t)0x0001U)    /*!<
Transceiver Basic Status Register    */
#define PHY_RESET                    ((uint16_t)0x8000U)  /*!< PHY
Reset */
#define PHY_LOOPBACK                 ((uint16_t)0x4000U)  /*!<
Select loop-back mode */
#define PHY_FULLDUPLEX_100M          ((uint16_t)0x2100U)
/*!< Set the full-duplex mode at 100 Mb/s */
#define PHY_HALFDUPLEX_100M          ((uint16_t)0x2000U)
/*!< Set the half-duplex mode at 100 Mb/s */
#define PHY_FULLDUPLEX_10M           ((uint16_t)0x0100U)
/*!< Set the full-duplex mode at 10 Mb/s  */
#define PHY_HALFDUPLEX_10M           ((uint16_t)0x0000U)
/*!< Set the half-duplex mode at 10 Mb/s  */
#define PHY_AUTONEGOTIATION          ((uint16_t)0x1000U)
/*!< Enable auto-negotiation function     */
```

```
#define PHY_RESTART_AUTONEGOTIATION
((uint16_t)0x0200U)  /*!< Restart auto-negotiation function    */

#define PHY_POWERDOWN                ((uint16_t)0x0800U)  /*!<
Select the power down mode          */

#define PHY_ISOLATE                  ((uint16_t)0x0400U)  /*!< Isolate
PHY from MII                 */

#define PHY_AUTONEGO_COMPLETE        ((uint16_t)0x0020U)
/*!< Auto-Negotiation process completed   */

#define PHY_LINKED_STATUS            ((uint16_t)0x0004U)  /*!<
Valid link established              */

#define PHY_JABBER_DETECTION         ((uint16_t)0x0002U)
/*!< Jabber condition detected           */

/* Section 4: Extended PHY Registers */

#define PHY_SR                       ((uint16_t))    /*!< PHY status register
Offset                    */

#define PHY_SPEED_STATUS             ((uint16_t))  /*!< PHY Speed
mask                          */

#define PHY_DUPLEX_STATUS            ((uint16_t))  /*!< PHY
Duplex mask                          */

/* ################### SPI peripheral configuration
############################ */

/* CRC FEATURE: Use to activate CRC feature inside HAL SPI Driver

* Activated: CRC code is present inside driver

* Deactivated: CRC code cleaned from driver

*/

#define USE_SPI_CRC                  0U
```

```
/* Includes ------------------------------------------------------------------*/
/**
  * @brief Include module's header file
  */
#ifdef HAL_RCC_MODULE_ENABLED
  #include "stm32f4xx_hal_rcc.h"
#endif /* HAL_RCC_MODULE_ENABLED */
#ifdef HAL_GPIO_MODULE_ENABLED
  #include "stm32f4xx_hal_gpio.h"
#endif /* HAL_GPIO_MODULE_ENABLED */
#ifdef HAL_EXTI_MODULE_ENABLED
  #include "stm32f4xx_hal_exti.h"
#endif /* HAL_EXTI_MODULE_ENABLED */
#ifdef HAL_DMA_MODULE_ENABLED
  #include "stm32f4xx_hal_dma.h"
#endif /* HAL_DMA_MODULE_ENABLED */
#ifdef HAL_CORTEX_MODULE_ENABLED
  #include "stm32f4xx_hal_cortex.h"
#endif /* HAL_CORTEX_MODULE_ENABLED */
#ifdef HAL_ADC_MODULE_ENABLED
  #include "stm32f4xx_hal_adc.h"
#endif /* HAL_ADC_MODULE_ENABLED */
#ifdef HAL_CAN_MODULE_ENABLED
```

```
  #include "stm32f4xx_hal_can.h"
#endif /* HAL_CAN_MODULE_ENABLED */
#ifdef HAL_CAN_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_can_legacy.h"
#endif /* HAL_CAN_LEGACY_MODULE_ENABLED */
#ifdef HAL_CRC_MODULE_ENABLED
  #include "stm32f4xx_hal_crc.h"
#endif /* HAL_CRC_MODULE_ENABLED */
#ifdef HAL_CRYP_MODULE_ENABLED
  #include "stm32f4xx_hal_cryp.h"
#endif /* HAL_CRYP_MODULE_ENABLED */
#ifdef HAL_DMA2D_MODULE_ENABLED
  #include "stm32f4xx_hal_dma2d.h"
#endif /* HAL_DMA2D_MODULE_ENABLED */
#ifdef HAL_DAC_MODULE_ENABLED
  #include "stm32f4xx_hal_dac.h"
#endif /* HAL_DAC_MODULE_ENABLED */
#ifdef HAL_DCMI_MODULE_ENABLED
  #include "stm32f4xx_hal_dcmi.h"
#endif /* HAL_DCMI_MODULE_ENABLED */
#ifdef HAL_ETH_MODULE_ENABLED
  #include "stm32f4xx_hal_eth.h"
#endif /* HAL_ETH_MODULE_ENABLED */
```

```
#ifdef HAL_ETH_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_eth_legacy.h"
#endif /* HAL_ETH_LEGACY_MODULE_ENABLED */
#ifdef HAL_FLASH_MODULE_ENABLED
  #include "stm32f4xx_hal_flash.h"
#endif /* HAL_FLASH_MODULE_ENABLED */
#ifdef HAL_SRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sram.h"
#endif /* HAL_SRAM_MODULE_ENABLED */
#ifdef HAL_NOR_MODULE_ENABLED
  #include "stm32f4xx_hal_nor.h"
#endif /* HAL_NOR_MODULE_ENABLED */
#ifdef HAL_NAND_MODULE_ENABLED
  #include "stm32f4xx_hal_nand.h"
#endif /* HAL_NAND_MODULE_ENABLED */
#ifdef HAL_PCCARD_MODULE_ENABLED
  #include "stm32f4xx_hal_pccard.h"
#endif /* HAL_PCCARD_MODULE_ENABLED */
#ifdef HAL_SDRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sdram.h"
#endif /* HAL_SDRAM_MODULE_ENABLED */
#ifdef HAL_HASH_MODULE_ENABLED
 #include "stm32f4xx_hal_hash.h"
```

```
#endif /* HAL_HASH_MODULE_ENABLED */
#ifdef HAL_I2C_MODULE_ENABLED
 #include "stm32f4xx_hal_i2c.h"
#endif /* HAL_I2C_MODULE_ENABLED */
#ifdef HAL_SMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_smbus.h"
#endif /* HAL_SMBUS_MODULE_ENABLED */
#ifdef HAL_I2S_MODULE_ENABLED
 #include "stm32f4xx_hal_i2s.h"
#endif /* HAL_I2S_MODULE_ENABLED */
#ifdef HAL_IWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_iwdg.h"
#endif /* HAL_IWDG_MODULE_ENABLED */
#ifdef HAL_LTDC_MODULE_ENABLED
 #include "stm32f4xx_hal_ltdc.h"
#endif /* HAL_LTDC_MODULE_ENABLED */
#ifdef HAL_PWR_MODULE_ENABLED
 #include "stm32f4xx_hal_pwr.h"
#endif /* HAL_PWR_MODULE_ENABLED */
#ifdef HAL_RNG_MODULE_ENABLED
 #include "stm32f4xx_hal_rng.h"
#endif /* HAL_RNG_MODULE_ENABLED */
#ifdef HAL_RTC_MODULE_ENABLED
```

```
 #include "stm32f4xx_hal_rtc.h"
#endif /* HAL_RTC_MODULE_ENABLED */
#ifdef HAL_SAI_MODULE_ENABLED
 #include "stm32f4xx_hal_sai.h"
#endif /* HAL_SAI_MODULE_ENABLED */
#ifdef HAL_SD_MODULE_ENABLED
 #include "stm32f4xx_hal_sd.h"
#endif /* HAL_SD_MODULE_ENABLED */
#ifdef HAL_SPI_MODULE_ENABLED
 #include "stm32f4xx_hal_spi.h"
#endif /* HAL_SPI_MODULE_ENABLED */
#ifdef HAL_TIM_MODULE_ENABLED
 #include "stm32f4xx_hal_tim.h"
#endif /* HAL_TIM_MODULE_ENABLED */
#ifdef HAL_UART_MODULE_ENABLED
 #include "stm32f4xx_hal_uart.h"
#endif /* HAL_UART_MODULE_ENABLED */
#ifdef HAL_USART_MODULE_ENABLED
 #include "stm32f4xx_hal_usart.h"
#endif /* HAL_USART_MODULE_ENABLED */
#ifdef HAL_IRDA_MODULE_ENABLED
 #include "stm32f4xx_hal_irda.h"
#endif /* HAL_IRDA_MODULE_ENABLED */
```

```
#ifdef HAL_SMARTCARD_MODULE_ENABLED
 #include "stm32f4xx_hal_smartcard.h"
#endif /* HAL_SMARTCARD_MODULE_ENABLED */
#ifdef HAL_WWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_wwdg.h"
#endif /* HAL_WWDG_MODULE_ENABLED */
#ifdef HAL_PCD_MODULE_ENABLED
 #include "stm32f4xx_hal_pcd.h"
#endif /* HAL_PCD_MODULE_ENABLED */
#ifdef HAL_HCD_MODULE_ENABLED
 #include "stm32f4xx_hal_hcd.h"
#endif /* HAL_HCD_MODULE_ENABLED */
#ifdef HAL_DSI_MODULE_ENABLED
 #include "stm32f4xx_hal_dsi.h"
#endif /* HAL_DSI_MODULE_ENABLED */
#ifdef HAL_QSPI_MODULE_ENABLED
 #include "stm32f4xx_hal_qspi.h"
#endif /* HAL_QSPI_MODULE_ENABLED */
#ifdef HAL_CEC_MODULE_ENABLED
 #include "stm32f4xx_hal_cec.h"
#endif /* HAL_CEC_MODULE_ENABLED */
#ifdef HAL_FMPI2C_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpi2c.h"
```

```
#endif /* HAL_FMPI2C_MODULE_ENABLED */
#ifdef HAL_FMPSMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpsmbus.h"
#endif /* HAL_FMPSMBUS_MODULE_ENABLED */
#ifdef HAL_SPDIFRX_MODULE_ENABLED
 #include "stm32f4xx_hal_spdifrx.h"
#endif /* HAL_SPDIFRX_MODULE_ENABLED */
#ifdef HAL_DFSDM_MODULE_ENABLED
 #include "stm32f4xx_hal_dfsdm.h"
#endif /* HAL_DFSDM_MODULE_ENABLED */
#ifdef HAL_LPTIM_MODULE_ENABLED
 #include "stm32f4xx_hal_lptim.h"
#endif /* HAL_LPTIM_MODULE_ENABLED */
#ifdef HAL_MMC_MODULE_ENABLED
 #include "stm32f4xx_hal_mmc.h"
#endif /* HAL_MMC_MODULE_ENABLED */
/* Exported macro ------------------------------------------------------------*/
#ifdef  USE_FULL_ASSERT
/**
  * @brief  The assert_param macro is used for function's parameters check.
  * @param  expr If expr is false, it calls assert_failed function
  *         which reports the name of the source file and the source
```

```
  *        line number of the call that failed.
  *        If expr is true, it returns no value.
  * @retval None
  */
  #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t
*)__FILE__, __LINE__))
/* Exported functions ---------------------------------------------------- */
  void assert_failed(uint8_t* file, uint32_t line);
#else
  #define assert_param(expr) ((void)0U)
#endif /* USE_FULL_ASSERT */
#ifdef __cplusplus
}
#endif
#endif /* __STM32F4xx_HAL_CONF_H */
```

16.9 CODE OF STM32F4XX_IT.H:

```
/* USER CODE BEGIN Header */
/**
*****************************************************************
*******************
  * @file    stm32f4xx_it.h
  * @brief   This file contains the headers of the interrupt handlers.
```

```
*****************************************************************
*******************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*****************************************************************
*******************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_IT_H
#define __STM32F4xx_IT_H
#ifdef __cplusplus
 extern "C" {
#endif
/* Private includes ----------------------------------------------------------*/
```

```
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */
/* USER CODE END EC */
/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */
/* USER CODE END EM *
/* Exported functions prototypes ---------------------------------------------*/
void NMI_Handler(void);
void HardFault_Handler(void);
void MemManage_Handler(void);
void BusFault_Handler(void);
void UsageFault_Handler(void);
void SVC_Handler(void);
void DebugMon_Handler(void);
void PendSV_Handler(void);
void SysTick_Handler(void);
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
```

```
#ifdef __cplusplus
}
#endif
#endif /* __STM32F4xx_IT_H */
```

16.10 CODE OF MAIN.C:

```
/* USER CODE BEGIN Header */
/**
**********************************************************
********************
  * @file           : main.c
  * @brief          : Main program body
**********************************************************
********************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
```

```
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
/* USER CODE END PTD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
UART_HandleTypeDef huart2;
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
```

```
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */
  /* USER CODE END 1 */
  /* MCU Configuration--------------------------------------------------------*/
  /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
  HAL_Init();
  /* USER CODE BEGIN Init */
  /* USER CODE END Init */
  /* Configure the system clock */
```

```
  SystemClock_Config();
  /* USER CODE BEGIN SysInit */
  /* USER CODE END SysInit */
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  /* USER CODE BEGIN 2 */
  /* USER CODE END 2 */
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
```

```
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();

__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_
VOLTAGE_SCALE1);
  /** Initializes the RCC Oscillators according to the specified
parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue =
RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 16;
  RCC_OscInitStruct.PLL.PLLN = 336;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV4;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
```

```
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK

|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource =
RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct,
FLASH_LATENCY_2) != HAL_OK)
  {
    Error_Handler();
  }
}
/**
  * @brief USART2 Initialization Function
  * @param None
  * @retval None
  */
```

```
static void MX_USART2_UART_Init(void)
{
  /* USER CODE BEGIN USART2_Init 0 */
  /* USER CODE END USART2_Init 0 */
  /* USER CODE BEGIN USART2_Init 1 */
  /* USER CODE END USART2_Init 1 */
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 115200;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART2_Init 2 */
  /* USER CODE END USART2_Init 2 */
}
/**
  * @brief GPIO Initialization Function
```

```
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */
  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin,
GPIO_PIN_RESET);
  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);
  /*Configure GPIO pin : LD2_Pin */
  GPIO_InitStruct.Pin = LD2_Pin;
```

```
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);
/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}
/* USER CODE BEGIN 4 */
/* USER CODE END 4 */
/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return
state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
```

```
}
#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

16.11 CODE OF STM32F4XX_HAL_MSP.C:

```
/* USER CODE BEGIN Header */
/**
```

```
**********************************************************
*******************
  * @file        stm32f4xx_hal_msp.c
  * @brief       This file provides code for the MSP Initialization
  *              and de-Initialization codes.
**********************************************************
*******************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* USER CODE BEGIN Includes */
```

```
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN TD */
/* USER CODE END TD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN Define */
/* USER CODE END Define */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN Macro */
/* USER CODE END Macro */
/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* External functions --------------------------------------------------------*/
/* USER CODE BEGIN ExternalFunctions */
/* USER CODE END ExternalFunctions */
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/**
  * Initializes the Global MSP.
```

```
  */
void HAL_MspInit(void)
{
  /* USER CODE BEGIN MspInit 0 */
  /* USER CODE END MspInit 0 */
  __HAL_RCC_SYSCFG_CLK_ENABLE();
  __HAL_RCC_PWR_CLK_ENABLE();
  HAL_NVIC_SetPriorityGrouping(NVIC_PRIORITYGROUP_0);
  /* System interrupt init*/
  /* USER CODE BEGIN MspInit 1 */
  /* USER CODE END MspInit 1 */
}
/**
* @brief UART MSP Initialization
* This function configures the hardware resources used in this example
* @param huart: UART handle pointer
* @retval None
*/
void HAL_UART_MspInit(UART_HandleTypeDef* huart)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  if(huart->Instance==USART2)
  {
```

```
  /* USER CODE BEGIN USART2_MspInit 0 */
  /* USER CODE END USART2_MspInit 0 */
    /* Peripheral clock enable */
    __HAL_RCC_USART2_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();
    /**USART2 GPIO Configuration
    PA2     ------> USART2_TX
    PA3     ------> USART2_RX
    */
    GPIO_InitStruct.Pin = USART_TX_Pin|USART_RX_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF7_USART2;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
  /* USER CODE BEGIN USART2_MspInit 1 */
  /* USER CODE END USART2_MspInit 1 */
  }
}
/**
* @brief UART MSP De-Initialization
* This function freeze the hardware resources used in this example
* @param huart: UART handle pointer
```

```
* @retval None
*/
void HAL_UART_MspDeInit(UART_HandleTypeDef* huart)
{
  if(huart->Instance==USART2)
  {
  /* USER CODE BEGIN USART2_MspDeInit 0 */
  /* USER CODE END USART2_MspDeInit 0 */
    /* Peripheral clock disable */
    __HAL_RCC_USART2_CLK_DISABLE();
    /**USART2 GPIO Configuration
    PA2     ------> USART2_TX
    PA3     ------> USART2_RX
    */
    HAL_GPIO_DeInit(GPIOA, USART_TX_Pin|USART_RX_Pin);
  /* USER CODE BEGIN USART2_MspDeInit 1 */
  /* USER CODE END USART2_MspDeInit 1 */
  }
}
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

16.12 CODE OF STM32F4XX-IT.C:

```
/* USER CODE BEGIN Header */
```

```
/**
**********************************************************
*******************
  * @file    stm32f4xx_it.c
  * @brief   Interrupt Service Routines.
**********************************************************
*******************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stm32f4xx_it.h"
```

```
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN TD */
/* USER CODE END TD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/* External variables --------------------------------------------------------*/
/* USER CODE BEGIN EV */
```

```
/* USER CODE END EV */
/******************************************************
*********************/
/*          Cortex-M4 Processor Interruption and Exception Handlers
*/
/******************************************************
*********************/
/**
  * @brief This function handles Non maskable interrupt.
  */
void NMI_Handler(void)
{
  /* USER CODE BEGIN NonMaskableInt_IRQn 0 */
  /* USER CODE END NonMaskableInt_IRQn 0 */
  /* USER CODE BEGIN NonMaskableInt_IRQn 1 */
   while (1)
  {
  }
  /* USER CODE END NonMaskableInt_IRQn 1 */
}
/**
  * @brief This function handles Hard fault interrupt.
  */
void HardFault_Handler(void)
```

```
{
  /* USER CODE BEGIN HardFault_IRQn 0 */
  /* USER CODE END HardFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_HardFault_IRQn 0 */
    /* USER CODE END W1_HardFault_IRQn 0 */
  }
}
/**
  * @brief This function handles Memory management fault.
  */
void MemManage_Handler(void)
{
  /* USER CODE BEGIN MemoryManagement_IRQn 0 */
  /* USER CODE END MemoryManagement_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_MemoryManagement_IRQn 0 */
    /* USER CODE END W1_MemoryManagement_IRQn 0 */
  }
}
/**
```

```
  * @brief This function handles Pre-fetch fault, memory access fault.
  */
void BusFault_Handler(void)
{
  /* USER CODE BEGIN BusFault_IRQn 0 */
  /* USER CODE END BusFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_BusFault_IRQn 0 */
    /* USER CODE END W1_BusFault_IRQn 0 */
  }
}
/**
  * @brief This function handles Undefined instruction or illegal state.
  */
void UsageFault_Handler(void)
{
  /* USER CODE BEGIN UsageFault_IRQn 0 */
  /* USER CODE END UsageFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_UsageFault_IRQn 0 */
    /* USER CODE END W1_UsageFault_IRQn 0 */
```

```
  }
}
/**
  * @brief This function handles System service call via SWI
instruction.
  */
void SVC_Handler(void)
{
  /* USER CODE BEGIN SVCall_IRQn 0 */
  /* USER CODE END SVCall_IRQn 0 */
  /* USER CODE BEGIN SVCall_IRQn 1 */
  /* USER CODE END SVCall_IRQn 1 */
}
/**
  * @brief This function handles Debug monitor.
  */
void DebugMon_Handler(void)
{
  /* USER CODE BEGIN DebugMonitor_IRQn 0 */
  /* USER CODE END DebugMonitor_IRQn 0 */
  /* USER CODE BEGIN DebugMonitor_IRQn 1 */
  /* USER CODE END DebugMonitor_IRQn 1 */
}
```

```
/**
  * @brief This function handles Pendable request for system service.
  */
void PendSV_Handler(void)
{
  /* USER CODE BEGIN PendSV_IRQn 0 */
  /* USER CODE END PendSV_IRQn 0 */
  /* USER CODE BEGIN PendSV_IRQn 1 */
  /* USER CODE END PendSV_IRQn 1 */
}
/**
  * @brief This function handles System tick timer.
  */
void SysTick_Handler(void)
{
  /* USER CODE BEGIN SysTick_IRQn 0 */
  /* USER CODE END SysTick_IRQn 0 */
  HAL_IncTick();
  /* USER CODE BEGIN SysTick_IRQn 1 */
  /* USER CODE END SysTick_IRQn 1 */
}
/*****************************************************
*********************/
```

```
/* STM32F4xx Peripheral Interrupt Handlers                                    */
/* Add here the Interrupt Handlers for the used peripherals.                  */
/* For the available peripheral interrupt handler names,                      */
/* please refer to the startup file (startup_stm32f4xx.s).                   */
/*****************************************************************************/
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

16.13 CODE OF SYSCALLS.C:

```
/**
*****************************************************************************
 * @file      syscalls.c
 * @author    Auto-generated by STM32CubeIDE
 * @brief     STM32CubeIDE Minimal System calls file
 *
 *            For more information about which c-functions
 *            need which of these lowlevel functions
 *            please consult the Newlib libc-manual

*****************************************************************************
 * @attention
 *
```

```
 * Copyright (c) 2020-2024 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the
LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
**********************************************************
*******************
 */
/* Includes */
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
#include <sys/times.h>
/* Variables */
extern int __io_putchar(int ch) __attribute__((weak));
extern int __io_getchar(void) __attribute__((weak));
char *__env[1] = { 0 };
```

```
char **environ = __env;
/* Functions */
void initialise_monitor_handles()
{
}
int _getpid(void)
{
  return 1;
}
int _kill(int pid, int sig)
{
  (void)pid;
  (void)sig;
  errno = EINVAL;
  return -1;
}
void _exit (int status)
{
  _kill(status, -1);
  while (1) {}    /* Make sure we hang here */
}
__attribute__((weak)) int _read(int file, char *ptr, int len)
{
```

```
  (void)file;
  int DataIdx;
  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    *ptr++ = __io_getchar();
  }
  return len;
}
__attribute__((weak)) int _write(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;
  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    __io_putchar(*ptr++);
  }
  return len;
}
int _close(int file)
{
  (void)file;
  return -1;
}
```

```
int _fstat(int file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
int _isatty(int file)
{
  (void)file;
  return 1;
}
int _lseek(int file, int ptr, int dir)
{
  (void)file;
  (void)ptr;
  (void)dir;
  return 0;
}
int _open(char *path, int flags, ...)
{
  (void)path;
  (void)flags;
  /* Pretend like we always fail */
```

```
  return -1;
}
int _wait(int *status)
{
  (void)status;
  errno = ECHILD;
  return -1;
}
int _unlink(char *name)
{
  (void)name;
  errno = ENOENT;
  return -1;
}
int _times(struct tms *buf)
{
  (void)buf;
  return -1;
}
int _stat(char *file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
```

```
  return 0;
}
int _link(char *old, char *new)
{
  (void)old;
  (void)new;
  errno = EMLINK;
  return -1;
}
int _fork(void)
{
  errno = EAGAIN;
  return -1;
}
int _execve(char *name, char **argv, char **env)
{
  (void)name;
  (void)argv;
  (void)env;
  errno = ENOMEM;
  return -1;
}
```

16.14 CODE OF SYSTEM.C:

```
/**
**********************************************************
*******************
 * @file      sysmem.c
 * @author    Generated by STM32CubeIDE
 * @brief     STM32CubeIDE System Memory calls file
 *
 *          For more information about which C functions
 *          need which of these lowlevel functions
 *          please consult the newlib libc manual
**********************************************************
********************
 * @attention
 *
 * Copyright (c) 2024 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the
LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
**********************************************************
*******************
```

```
 */
/* Includes */
#include <errno.h>
#include <stdint.h>
/**
 * Pointer to the current high watermark of the heap usage
 */
static uint8_t *__sbrk_heap_end = NULL;
/**
 * @brief _sbrk() allocates memory to the newlib heap and is used by malloc
 *        and others from the C library
 *
 * @verbatim
 *
##################################################################################
 * #  .data  #  .bss  #       newlib heap       #          MSP stack          #
 * #         #        #                          # Reserved by _Min_Stack_Size #
 *
##################################################################################
 * ^-- RAM start      ^-- _end                          _estack, RAM end --^
 * @endverbatim
 *
```

```
 * This implementation starts allocating at the '_end' linker symbol
 * The '_Min_Stack_Size' linker symbol reserves a memory for the MSP stack
 * The implementation considers '_estack' linker symbol to be RAM end
 * NOTE: If the MSP stack, at any point during execution, grows larger than the
 * reserved size, please increase the '_Min_Stack_Size'.
 *
 * @param incr Memory size
 * @return Pointer to allocated memory
 */
void *_sbrk(ptrdiff_t incr)
{
  extern uint8_t _end; /* Symbol defined in the linker script */
  extern uint8_t _estack; /* Symbol defined in the linker script */
  extern uint32_t _Min_Stack_Size; /* Symbol defined in the linker script */
  const uint32_t stack_limit = (uint32_t)&_estack - (uint32_t)&_Min_Stack_Size;
  const uint8_t *max_heap = (uint8_t *)stack_limit;
  uint8_t *prev_heap_end;
  /* Initialize heap end at first call */
  if (NULL == __sbrk_heap_end)
  {
```

```
    __sbrk_heap_end = &_end;
  }
  /* Protect heap from growing into the reserved MSP stack */
  if (__sbrk_heap_end + incr > max_heap)
  {
    errno = ENOMEM;
    return (void *)-1;
  }
  prev_heap_end = __sbrk_heap_end;
  __sbrk_heap_end += incr;
  return (void *)prev_heap_end;
}
```

16.15 CODE OF SYSTEM_STM32F4XX.C:

```
/**
*****************************************************************
*******************
  * @file    system_stm32f4xx.c
  * @author  MCD Application Team
  * @brief   CMSIS Cortex-M4 Device Peripheral Access Layer System
Source File.
  *
  *   This file provides two functions and one global variable to be called
from
  *   user application:
```

```
  *      - SystemInit(): This function is called at startup just after reset and
  *                      before branch to main program. This call is made inside
  *                      the "startup_stm32f4xx.s" file.
  *
  *      - SystemCoreClock variable: Contains the core clock (HCLK), it
can be used
  *                              by the user application to setup the SysTick
  *                              timer or configure other parameters.
  *
  *      - SystemCoreClockUpdate(): Updates the variable
SystemCoreClock and must
  *                             be called whenever the core clock is changed
  *                             during program execution.
  *
  *

**************************************************************
*******************
  * @attention
  *
  * Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
```

```
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*****************************************************************
*******************
  */
/** @addtogroup CMSIS
  * @{
  */
/** @addtogroup stm32f4xx_system
  * @{
  */
/** @addtogroup STM32F4xx_System_Private_Includes
  * @{
  */
#include "stm32f4xx.h"
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    ((uint32_t)25000000) /*!< Default value of
the External oscillator in Hz */
#endif /* HSE_VALUE */
#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000) /*!< Value of the
Internal oscillator in Hz*/
#endif /* HSI_VALUE */
```

```
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_TypesDefinitions
  * @{
  */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Defines
  * @{
  */
/************************* Miscellaneous Configuration
************************/
/*!< Uncomment the following line if you need to use external SRAM or
SDRAM as data memory  */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx) || defined(STM32F417xx)\
 || defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx) ||
defined(STM32F412Zx) || defined(STM32F412Vx)
/* #define DATA_IN_ExtSRAM */
#endif /* STM32F40xxx || STM32F41xxx || STM32F42xxx ||
STM32F43xxx || STM32F469xx || STM32F479xx ||\
```

```
        STM32F412Zx || STM32F412Vx */
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F446xx) || defined(STM32F469xx) ||
defined(STM32F479xx)
/* #define DATA_IN_ExtSDRAM */
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F446xx || STM32F469xx ||\
        STM32F479xx */
/* Note: Following vector table addresses must be defined in line with
linker
        configuration. */
/*!< Uncomment the following line if you need to relocate the vector
table
    anywhere in Flash or Sram, else the vector table is kept at the
automatic
    remap of boot address selected */
/* #define USER_VECT_TAB_ADDRESS */
#if defined(USER_VECT_TAB_ADDRESS)
/*!< Uncomment the following line if you need to relocate your vector
Table
    in Sram else user remap will be done in Flash. */
/* #define VECT_TAB_SRAM */
#if defined(VECT_TAB_SRAM)
```

```
#define VECT_TAB_BASE_ADDRESS   SRAM_BASE       /*!< Vector Table base address field.
                                                This value must be a multiple of 0x200. */
#define VECT_TAB_OFFSET         0x00000000U     /*!< Vector Table base offset field.
                                                This value must be a multiple of 0x200. */
#else
#define VECT_TAB_BASE_ADDRESS   FLASH_BASE      /*!< Vector Table base address field.
                                                This value must be a multiple of 0x200. */
#define VECT_TAB_OFFSET         0x00000000U     /*!< Vector Table base offset field.
                                                This value must be a multiple of 0x200. */
#endif /* VECT_TAB_SRAM */
#endif /* USER_VECT_TAB_ADDRESS */
/*****************************************************************************/
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Macros
  * @{
```

```
  */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Variables
  * @{
  */
  /* This variable is updated in three ways:
     1) by calling CMSIS function SystemCoreClockUpdate()
     2) by calling HAL API function HAL_RCC_GetHCLKFreq()
     3) each time HAL_RCC_ClockConfig() is called to configure the system clock frequency
        Note: If you use this function to configure the system clock; then there
              is no need to call the 2 first functions listed above, since SystemCoreClock
              variable is updated automatically.
  */
uint32_t SystemCoreClock = 16000000;
const uint8_t AHBPrescTable[16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9};
const uint8_t APBPrescTable[8]  = {0, 0, 0, 0, 1, 2, 3, 4};
/**
  * @}
```

```
  */
/** @addtogroup STM32F4xx_System_Private_FunctionPrototypes
  * @{
  */
#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  static void SystemInit_ExtMemCtl(void);
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Functions
  * @{
  */
/**
  * @brief  Setup the microcontroller system
  *         Initialize the FPU setting, vector table location and External memory
  *         configuration.
  * @param  None
  * @retval None
  */
void SystemInit(void)
{
```

```
  /* FPU settings ------------------------------------------------------------*/
  #if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
    SCB->CPACR |= ((3UL << 10*2)|(3UL << 11*2));  /* set CP10 and
CP11 Full Access */
  #endif
#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  SystemInit_ExtMemCtl();
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */
  /* Configure the Vector Table location -------------------------------------*/
#if defined(USER_VECT_TAB_ADDRESS)
  SCB->VTOR = VECT_TAB_BASE_ADDRESS |
VECT_TAB_OFFSET; /* Vector Table Relocation in Internal SRAM */
#endif /* USER_VECT_TAB_ADDRESS */
}
/**
   * @brief  Update SystemCoreClock variable according to Clock
Register Values.
  *         The SystemCoreClock variable contains the core clock (HCLK),
it can
  *         be used by the user application to setup the SysTick timer or
configure
  *         other parameters.
  *
  * @note   Each time the core clock (HCLK) changes, this function
must be called
```

```
  *        to update SystemCoreClock variable value. Otherwise, any configuration
  *        based on this variable will be incorrect.
  *
  * @note   - The system frequency computed by this function is not the real
  *           frequency in the chip. It is calculated based on the predefined
  *           constant and the selected clock source:
  *
  *           - If SYSCLK source is HSI, SystemCoreClock will contain the HSI_VALUE(*)
  *
  *           - If SYSCLK source is HSE, SystemCoreClock will contain the HSE_VALUE(**)
  *
  *           - If SYSCLK source is PLL, SystemCoreClock will contain the HSE_VALUE(**)
  *             or HSI_VALUE(*) multiplied/divided by the PLL factors.
  *
  *         (*) HSI_VALUE is a constant defined in stm32f4xx_hal_conf.h file (default value
  *             16 MHz) but the real value may vary depending on the variations
  *             in voltage and temperature.
  *
```

```
  *        (**) HSE_VALUE is a constant defined in
stm32f4xx_hal_conf.h file (its value
  *             depends on the application requirements), user has to ensure
that HSE_VALUE
  *             is same as the real frequency of the crystal used. Otherwise,
this function
  *             may have wrong result.
  *
  *        - The result of this function could be not correct when using
fractional
  *          value for HSE crystal.
  *
  * @param  None
  * @retval None
  */
void SystemCoreClockUpdate(void)
{
  uint32_t tmp = 0, pllvco = 0, pllp = 2, pllsource = 0, pllm = 2;
  /* Get SYSCLK source -------------------------------------------------------
*/
  tmp = RCC->CFGR & RCC_CFGR_SWS;
  switch (tmp)
  {
    case 0x00:  /* HSI used as system clock source */
      SystemCoreClock = HSI_VALUE;
```

```
    break;
  case 0x04:  /* HSE used as system clock source */
    SystemCoreClock = HSE_VALUE;
    break;
  case 0x08:  /* PLL used as system clock source */
    /* PLL_VCO = (HSE_VALUE or HSI_VALUE / PLL_M) * PLL_N
       SYSCLK = PLL_VCO / PLL_P
       */
    pllsource = (RCC->PLLCFGR & RCC_PLLCFGR_PLLSRC) >>
22;
    pllm = RCC->PLLCFGR & RCC_PLLCFGR_PLLM;
    if (pllsource != 0)
    {
      /* HSE used as PLL clock source */
      pllvco = (HSE_VALUE / pllm) * ((RCC->PLLCFGR &
RCC_PLLCFGR_PLLN) >> 6);
    }
    else
    {
      /* HSI used as PLL clock source */
      pllvco = (HSI_VALUE / pllm) * ((RCC->PLLCFGR &
RCC_PLLCFGR_PLLN) >> 6);
    }
```

```
    pllp = (((RCC->PLLCFGR & RCC_PLLCFGR_PLLP) >>16) + 1 )
*2;
    SystemCoreClock = pllvco/pllp;
    break;
   default:
    SystemCoreClock = HSI_VALUE;
    break;
  }
  /* Compute HCLK frequency ----------------------------------------------
-*/
  /* Get HCLK prescaler */
  tmp = AHBPrescTable[((RCC->CFGR & RCC_CFGR_HPRE) >> 4)];
  /* HCLK frequency */
  SystemCoreClock >>= tmp;
}
#if defined (DATA_IN_ExtSRAM) && defined
(DATA_IN_ExtSDRAM)
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
/**
  * @brief  Setup the external memory controller.
  *         Called in startup_stm32f4xx.s before jump to main.
  *         This function configures the external memories
(SRAM/SDRAM)
```

```
  *        This SRAM/SDRAM will be used as program data memory
(including heap and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
  register __IO uint32_t index;
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and
GPIOI interface clock */
  RCC->AHB1ENR |= 0x000001F8;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x00CCC0CC;
  GPIOD->AFR[1]  = 0xCCCCCCCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xAAAA0A8A;
  /* Configure PDx pins speed to 100 MHz */
  GPIOD->OSPEEDR = 0xFFFF0FCF;
  /* Configure PDx pins Output type to push-pull */
  GPIOD->OTYPER  = 0x00000000;
```

```
/* No pull-up, pull-down for PDx pins */
GPIOD->PUPDR   = 0x00000000;
/* Connect PEx pins to FMC Alternate function */
GPIOE->AFR[0]  = 0xC00CC0CC;
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA828A;
/* Configure PEx pins speed to 100 MHz */
GPIOE->OSPEEDR = 0xFFFFC3CF;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0xCCCCCCCC;
GPIOF->AFR[1]  = 0xCCCCCCCC;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA800AAA;
/* Configure PFx pins speed to 50 MHz */
GPIOF->OSPEEDR = 0xAA800AAA;
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
```

```
GPIOF->PUPDR   = 0x00000000;
/* Connect PGx pins to FMC Alternate function */
GPIOG->AFR[0]  = 0xCCCCCCCC;
GPIOG->AFR[1]  = 0xCCCCCCCC;
/* Configure PGx pins in Alternate function mode */
GPIOG->MODER   = 0xAAAAAAAA;
/* Configure PGx pins speed to 50 MHz */
GPIOG->OSPEEDR = 0xAAAAAAAA;
/* Configure PGx pins Output type to push-pull */
GPIOG->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PGx pins */
GPIOG->PUPDR   = 0x00000000;
/* Connect PHx pins to FMC Alternate function */
GPIOH->AFR[0]  = 0x00C0CC00;
GPIOH->AFR[1]  = 0xCCCCCCCC;
/* Configure PHx pins in Alternate function mode */
GPIOH->MODER   = 0xAAAA08A0;
/* Configure PHx pins speed to 50 MHz */
GPIOH->OSPEEDR = 0xAAAA08A0;
/* Configure PHx pins Output type to push-pull */
GPIOH->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PHx pins */
GPIOH->PUPDR   = 0x00000000;
```

```
  /* Connect PIx pins to FMC Alternate function */
  GPIOI->AFR[0]  = 0xCCCCCCCC;
  GPIOI->AFR[1]  = 0x00000CC0;
  /* Configure PIx pins in Alternate function mode */
  GPIOI->MODER   = 0x0028AAAA;
  /* Configure PIx pins speed to 50 MHz */
  GPIOI->OSPEEDR = 0x0028AAAA;
  /* Configure PIx pins Output type to push-pull */
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;
/*-- FMC Configuration ------------------------------------------------------*/
  /* Enable the FMC interface clock */
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
  FMC_Bank5_6->SDTR[0] = 0x01115351;
  /* SDRAM initialization sequence */
  /* Clock enable command */
  FMC_Bank5_6->SDCMR = 0x00000011;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  while((tmpreg != 0) && (timeout-- > 0))
```

```
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* Delay */
for (index = 0; index<1000; index++);
/* PALL command */
FMC_Bank5_6->SDCMR = 0x00000012;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* Auto refresh command */
FMC_Bank5_6->SDCMR = 0x00000073;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* MRD register program */
FMC_Bank5_6->SDCMR = 0x00046014;
```

```
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Set refresh count */
  tmpreg = FMC_Bank5_6->SDRTR;
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
  /* Disable write protection */
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
```

```
  FMC_Bank1E->BWTR[2] = 0x0ffffffff;
#endif /* STM32F469xx || STM32F479xx */
  (void)(tmp);
}
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F469xx || STM32F479xx */
#elif defined (DATA_IN_ExtSRAM) || defined
(DATA_IN_ExtSDRAM)
/**
  * @brief  Setup the external memory controller.
  *         Called in startup_stm32f4xx.s before jump to main.
  *         This function configures the external memories
(SRAM/SDRAM)
  *         This SRAM/SDRAM will be used as program data memory
(including heap and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F446xx) || defined(STM32F469xx) ||
defined(STM32F479xx)
```

```
#if defined (DATA_IN_ExtSDRAM)
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
  register __IO uint32_t index;
#if defined(STM32F446xx)
  /* Enable GPIOA, GPIOC, GPIOD, GPIOE, GPIOF, GPIOG interface
     clock */
  RCC->AHB1ENR |= 0x0000007D;
#else
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and
GPIOI interface
     clock */
  RCC->AHB1ENR |= 0x000001F8;
#endif /* STM32F446xx */
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);
#if defined(STM32F446xx)
  /* Connect PAx pins to FMC Alternate function */
  GPIOA->AFR[0]  |= 0xC0000000;
  GPIOA->AFR[1]  |= 0x00000000;
  /* Configure PDx pins in Alternate function mode */
  GPIOA->MODER   |= 0x00008000;
  /* Configure PDx pins speed to 50 MHz */
  GPIOA->OSPEEDR |= 0x00008000;
```

```
  /* Configure PDx pins Output type to push-pull */
  GPIOA->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOA->PUPDR   |= 0x00000000;
  /* Connect PCx pins to FMC Alternate function */
  GPIOC->AFR[0]  |= 0x00CC0000;
  GPIOC->AFR[1]  |= 0x00000000;
  /* Configure PDx pins in Alternate function mode */
  GPIOC->MODER   |= 0x00000A00;
  /* Configure PDx pins speed to 50 MHz */
  GPIOC->OSPEEDR |= 0x00000A00;
  /* Configure PDx pins Output type to push-pull */
  GPIOC->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOC->PUPDR   |= 0x00000000;
#endif /* STM32F446xx */
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x000000CC;
  GPIOD->AFR[1]  = 0xCC000CCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xA02A000A;
  /* Configure PDx pins speed to 50 MHz */
  GPIOD->OSPEEDR = 0xA02A000A;
```

```
/* Configure PDx pins Output type to push-pull */
GPIOD->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PDx pins */
GPIOD->PUPDR   = 0x00000000;
/* Connect PEx pins to FMC Alternate function */
GPIOE->AFR[0]  = 0xC00000CC;
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA800A;
/* Configure PEx pins speed to 50 MHz */
GPIOE->OSPEEDR = 0xAAAA800A;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0xCCCCCCCC;
GPIOF->AFR[1]  = 0xCCCCCCCC;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA800AAA;
/* Configure PFx pins speed to 50 MHz */
GPIOF->OSPEEDR = 0xAA800AAA;
/* Configure PFx pins Output type to push-pull */
```

```
  GPIOF->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PFx pins */
  GPIOF->PUPDR   = 0x00000000;
  /* Connect PGx pins to FMC Alternate function */
  GPIOG->AFR[0]  = 0xCCCCCCCC;
  GPIOG->AFR[1]  = 0xCCCCCCCC;
  /* Configure PGx pins in Alternate function mode */
  GPIOG->MODER   = 0xAAAAAAAA;
  /* Configure PGx pins speed to 50 MHz */
  GPIOG->OSPEEDR = 0xAAAAAAAA;
  /* Configure PGx pins Output type to push-pull */
  GPIOG->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
  /* Connect PHx pins to FMC Alternate function */
  GPIOH->AFR[0]  = 0x00C0CC00;
  GPIOH->AFR[1]  = 0xCCCCCCCC;
  /* Configure PHx pins in Alternate function mode */
  GPIOH->MODER   = 0xAAAA08A0;
  /* Configure PHx pins speed to 50 MHz */
```

```
  GPIOH->OSPEEDR = 0xAAAA08A0;
  /* Configure PHx pins Output type to push-pull */
  GPIOH->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PHx pins */
  GPIOH->PUPDR   = 0x00000000;
  /* Connect PIx pins to FMC Alternate function */
  GPIOI->AFR[0]  = 0xCCCCCCCC;
  GPIOI->AFR[1]  = 0x00000CC0;
  /* Configure PIx pins in Alternate function mode */
  GPIOI->MODER   = 0x0028AAAA;
  /* Configure PIx pins speed to 50 MHz */
  GPIOI->OSPEEDR = 0x0028AAAA;
  /* Configure PIx pins Output type to push-pull */
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F469xx || STM32F479xx */
/*-- FMC Configuration -----------------------------------------------------*/
  /* Enable the FMC interface clock */
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
```

```
  /* Configure and enable SDRAM bank1 */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCR[0] = 0x00001954;
#else
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
#endif /* STM32F446xx */
  FMC_Bank5_6->SDTR[0] = 0x01115351;
  /* SDRAM initialization sequence */
  /* Clock enable command */
  FMC_Bank5_6->SDCMR = 0x00000011;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Delay */
  for (index = 0; index<1000; index++);
  /* PALL command */
  FMC_Bank5_6->SDCMR = 0x00000012;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
```

```
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Auto refresh command */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x000000F3;
#else
  FMC_Bank5_6->SDCMR = 0x00000073;
#endif /* STM32F446xx */
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* MRD register program */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x00044014;
#else
  FMC_Bank5_6->SDCMR = 0x00046014;
#endif /* STM32F446xx */
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
```

```
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Set refresh count */
  tmpreg = FMC_Bank5_6->SDRTR;
#if defined(STM32F446xx)
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000050C<<1));
#else
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
#endif /* STM32F446xx */
  /* Disable write protection */
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#endif /* DATA_IN_ExtSDRAM */
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F446xx || STM32F469xx || STM32F479xx */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx) || defined(STM32F417xx)\
 || defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx) ||
defined(STM32F412Zx) || defined(STM32F412Vx)
#if defined(DATA_IN_ExtSRAM)
/*-- GPIOs Configuration -----------------------------------------------------*/
```

```
 /* Enable GPIOD, GPIOE, GPIOF and GPIOG interface clock */
RCC->AHB1ENR   |= 0x00000078;
/* Delay after an RCC peripheral clock enabling */
tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIODEN);
/* Connect PDx pins to FMC Alternate function */
GPIOD->AFR[0]  = 0x00CCC0CC;
GPIOD->AFR[1]  = 0xCCCCCCCC;
/* Configure PDx pins in Alternate function mode */
GPIOD->MODER   = 0xAAAA0A8A;
/* Configure PDx pins speed to 100 MHz */
GPIOD->OSPEEDR = 0xFFFF0FCF;
/* Configure PDx pins Output type to push-pull */
GPIOD->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PDx pins */
GPIOD->PUPDR   = 0x00000000;
/* Connect PEx pins to FMC Alternate function */
GPIOE->AFR[0]  = 0xC00CC0CC;
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA828A;
/* Configure PEx pins speed to 100 MHz */
GPIOE->OSPEEDR = 0xFFFFC3CF;
/* Configure PEx pins Output type to push-pull */
```

```
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0x00CCCCCC;
GPIOF->AFR[1]  = 0xCCCC0000;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA000AAA;
/* Configure PFx pins speed to 100 MHz */
GPIOF->OSPEEDR = 0xFF000FFF;
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
GPIOF->PUPDR   = 0x00000000;
/* Connect PGx pins to FMC Alternate function */
GPIOG->AFR[0]  = 0x00CCCCCC;
GPIOG->AFR[1]  = 0x000000C0;
/* Configure PGx pins in Alternate function mode */
GPIOG->MODER   = 0x00085AAA;
/* Configure PGx pins speed to 100 MHz */
GPIOG->OSPEEDR = 0x000CAFFF;
/* Configure PGx pins Output type to push-pull */
GPIOG->OTYPER  = 0x00000000;
```

```
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;
/*-- FMC/FSMC Configuration ----------------------------------------------
-*/
  /* Enable the FMC/FSMC interface clock */
  RCC->AHB3ENR         |= 0x00000001;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F469xx || STM32F479xx */
```

```
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx)|| defined(STM32F417xx)\
   || defined(STM32F412Zx) || defined(STM32F412Vx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FSMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FSMC_Bank1->BTCR[2]  = 0x00001011;
  FSMC_Bank1->BTCR[3]  = 0x00000201;
  FSMC_Bank1E->BWTR[2] = 0x0FFFFFFF;
#endif /* STM32F405xx || STM32F415xx || STM32F407xx ||
STM32F417xx || STM32F412Zx || STM32F412Vx */
#endif /* DATA_IN_ExtSRAM */
#endif /* STM32F405xx || STM32F415xx || STM32F407xx ||
STM32F417xx || STM32F427xx || STM32F437xx ||\
         STM32F429xx || STM32F439xx || STM32F469xx ||
STM32F479xx || STM32F412Zx || STM32F412Vx  */
  (void)(tmp);
}
#endif /* DATA_IN_ExtSRAM && DATA_IN_ExtSDRAM */
/**
  * @}
  */
/**
  * @}
```

```
  */
/**
  * @}
  */
```

CHAPTER#17 EMBEDDED SYSTEMS BARE-METAL EMBEDDED-C AND ASSEMBLY PROGRAMMING WITH CUBE IDE

17.1EMBEDDED SYSTEMS BARE-METAL EMBEDDED-C AND ASSEMBLY PROGRAMMING WITH CUBE IDE:

Bare-metal programming in embedded systems involves writing software that runs directly on the hardware without an operating system. This approach is commonly used in resource-constrained environments where efficiency and control are crucial. In the context of STM32 microcontrollers and using STM32Cube IDE, you can program in both C and assembly languages.

17.2 BARE-METAL EMBEDDED-C PROGRAMMING:

17.2.1 SETUP AND CONFIGURATION:

17.2.1.1 PROJECT CREATION:

Start a new STM32 project in STM32Cube IDE and configure the microcontroller and peripherals using STM32CubeMX.

17.2.1.2 GPIO CONFIGURATION:

Set up GPIO pins for input, output, or alternate functions as needed.

17.2.2 MAIN PROGRAM STRUCTURE:

17.2.2.1 INITIALIZATION:

Initialize the system clock, peripherals, and GPIOs.

17.2.2.2 MAIN LOOP:

Implement the main application logic in a continuous loop, which can include reading sensors, controlling actuators, and handling communication protocols.

17.2.3 INTERRUPT HANDLING:

17.2.3.1 ISR DEFINITION:

Define interrupt service routines (ISRs) for handling asynchronous events like button presses or communication interrupts.

17.2.3.2 INTERRUPT PRIORITIZATION:

Configure the priority of different interrupts as required.

17.2.4 PERIPHERAL CONTROL:

17.2.4.1 DIRECT REGISTER ACCESS:

For performance-critical sections, you might directly manipulate hardware registers instead of using HAL libraries.

17.2.4.2 DRIVERS AND LIBRARIES:

Use or write peripheral drivers that abstract hardware-specific details.

17.2.5 CODE OPTIMIZATION:

17.2.5.1 SIZE AND SPEED:

Optimize code for size and speed, considering the constraints of the microcontroller.

17.2.5.2 POWER MANAGEMENT:

Implement power-saving techniques, such as sleep modes or peripheral clock gating.

17.3 CODE OF BARE-METAL C CODE:

```
#include "stm32f4xx.h" // Include device-specific header file

void SystemClock_Config(void);

void GPIO_Init(void);

int main(void)

{

   HAL_Init(); // Initialize the Hardware Abstraction Layer

   SystemClock_Config(); // Configure the system clock

   GPIO_Init(); // Initialize GPIOs

   while (1)

   {

      // Toggle an LED

      HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
```

```
        HAL_Delay(500); // Delay 500ms
    }
}
void SystemClock_Config(void)
{
    // System clock configuration code
}
void GPIO_Init(void)
{
    __HAL_RCC_GPIOA_CLK_ENABLE(); // Enable GPIOA clock
    GPIO_InitTypeDef GPIO_InitStruct = {0};
    GPIO_InitStruct.Pin = GPIO_PIN_5;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct); // Initialize GPIO
}
```

17.4 ASSEMBLY PROGRAMMING WITH STM32CUBEIDE:

17.4.1 INLINE ASSEMBLY IN C:

We can insert assembly instructions directly into your C code using GC C's inline assembly syntax.

```
asm volatile ("nop"); // Insert a no-operation instruction
```

17.4.2 WRITING ASSEMBLY FILES:

Create.s or .asm files for standalone assembly code. These files can define startup routines, ISRs, or critical sections of code. Use ARM Cortex-M assembly instructions, which include data processing, branching, and control instructions.

17.4.3 LINKING ASSEMBLY AND C:

Ensure proper calling conventions when interfacing between C and assembly, especially regarding parameter passing and stack management. Use global labels and declare them as extern in C code if they need to be accessible across files.

17.5 CODE OF AN ASSEMBLY FUNCTION:

```
.global Reset_Handler

.type Reset_Handler, %function

Reset_Handler:

    LDR sp, =_estack    /* Load the stack pointer */

    BL main             /* Branch to main */

    B .                 /* Infinite loop in case main returns */
```

In bare-metal development, careful management of resources and understanding of the hardware is crucial. Using STM32Cube IDE, developers can leverage a powerful environment to manage both C and assembly code effectively.

CHAPTER#18 CODING GPIO DRIVERS FROM SCRATCH USING DATASHEET INFORMATION (PART 1)

18.1 CODING GPIO DRIVERS FROM SCRATCH USING DATASHEET INFORMATION(PART 1):

Coding GPIO drivers from scratch involves directly interacting with the microcontroller's hardware registers, typically using information from the microcontroller's datasheet and reference manual. Here's a general guide to help create GPIO drivers for an STM32 microcontroller, which you can adapt to other platforms with similar principles.

18.2 STEPS TO CODE GPIO DRIVERS FROM SCRATCH:

18.2.1 UNDERSTAND THE HARDWARE:

18.2.1.1 DATASHEET:

Provides an overview of the microcontroller, including pin configuration and electrical characteristics.

18.2.1.2 REFERENCES MANUAL:

Contains detailed descriptions of the microcontroller's peripherals and registers.

18.2.2 IDENTIFY GPIO REGISTERS:

The reference manual will describe registers associated with GPIO, including:

18.2.2.1 MODE:

GPIO port mode register (configures the mode: input, output, alternate function, analog).

18.2.2.2 OTYPER:

GPIO port output type register (configures output type: push-pull, open-drain).

18.2.2.3 OSPEEDER:

GPIO port output speed register (configures the output speed).

18.2.2.4 PUPDR:

GPIO port pull-up/pull-down register (configures internal pull-up/pull-down resistors).

18.2.2.5 IDR:

GPIO port input data register (reads the state of input pins).

18.2.2.6 ODR:

GPIO port output data register (sets the state of output pins).

18.2.2.7 BSRR:

GPIO port bit set/reset register (atomic bit set/reset).

18.2.2.8 LCKR:

GPIO port configuration lock register (locks the configuration of GPIO pins).

18.2.3 ENABLE GPIO CLOCK:

Before configuring GPIO, the peripheral clock must be enabled. This is usually done via the RCC (Reset and Clock Control) registers.

18.2.4 CONFIGURE GPIO PINS:

Set the mode, type, speed, and pull-up/pull-down settings for each GPIO pin.

18.2.5 READ/WRITE GPIO PINS:

Use the IDR and ODR registers to read from and write to GPIO pins.

18.2.6 ATOMIC SET AND RESET OPERATIONS:

Use the BSRR register for atomic bit set and reset operations, which are crucial in interrupt-driven or multi-threaded environments.

18.2.7 LOCKING GPIO CONFIGURATION:

Optionally, lock the GPIO configuration using the LCKR register to prevent accidental changes.

18.3 CODE OF MAIN.H:

```
/* USER CODE BEGIN Header */
/**
**********************************************************
*******************
  * @file           : main.h
  * @brief          : Header for main.c file.
  *                   This file contains the common defines of the application.
```

```
*****************************************************
*******************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*****************************************************
*******************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -----------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H
#ifdef __cplusplus
extern "C" {
#endif
/* Includes ------------------------------------------------------------------*/
#include "stm32f4xx_hal.h"
```

```
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */
/* USER CODE END EC */
/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */
/* USER CODE END EM */
/* Exported functions prototypes ---------------------------------------------*/
void Error_Handler(void);
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
/* Private defines -----------------------------------------------------------*/
#define B1_Pin GPIO_PIN_13
#define B1_GPIO_Port GPIOC
#define USART_TX_Pin GPIO_PIN_2
#define USART_TX_GPIO_Port GPIOA
#define USART_RX_Pin GPIO_PIN_3
#define USART_RX_GPIO_Port GPIOA
```

```
#define LD2_Pin GPIO_PIN_5
#define LD2_GPIO_Port GPIOA
#define TMS_Pin GPIO_PIN_13
#define TMS_GPIO_Port GPIOA
#define TCK_Pin GPIO_PIN_14
#define TCK_GPIO_Port GPIOA
#define SWO_Pin GPIO_PIN_3
#define SWO_GPIO_Port GPIOB
/* USER CODE BEGIN Private defines */
/* USER CODE END Private defines */
#ifdef __cplusplus
}
#endif
#endif /* __MAIN_H *
```

18.4 CODE OF STM32F4XX_HAL_CONFIG.H:

```
/* USER CODE BEGIN Header */
/**
*****************************************************************
*******************
  * @file    stm32f4xx_hal_conf_template.h
  * @author  MCD Application Team
  * @brief   HAL configuration template file.
  *          This file should be copied to the application folder and renamed
```

```
  *          to stm32f4xx_hal_conf.h.

**********************************************************
*******************
  * @attention
  *
  * Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *

**********************************************************
*******************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -----------------------------------*/
#ifndef __STM32F4xx_HAL_CONF_H
#define __STM32F4xx_HAL_CONF_H
#ifdef __cplusplus
 extern "C" {
#endif
```

```
/* Exported types ------------------------------------------------------------*/
/* Exported constants --------------------------------------------------------*/
/* ############################ Module Selection
################################## */
/**
  * @brief This is the list of modules to be used in the HAL driver
  */
#define HAL_MODULE_ENABLED
  /* #define HAL_CRYP_MODULE_ENABLED */
/* #define HAL_ADC_MODULE_ENABLED */
/* #define HAL_CAN_MODULE_ENABLED */
/* #define HAL_CRC_MODULE_ENABLED */
/* #define HAL_CAN_LEGACY_MODULE_ENABLED */
/* #define HAL_DAC_MODULE_ENABLED */
/* #define HAL_DCMI_MODULE_ENABLED */
/* #define HAL_DMA2D_MODULE_ENABLED */
/* #define HAL_ETH_MODULE_ENABLED */
/* #define HAL_ETH_LEGACY_MODULE_ENABLED */
/* #define HAL_NAND_MODULE_ENABLED */
/* #define HAL_NOR_MODULE_ENABLED */
/* #define HAL_PCCARD_MODULE_ENABLED */
/* #define HAL_SRAM_MODULE_ENABLED */
/* #define HAL_SDRAM_MODULE_ENABLED */
```

```
/* #define HAL_HASH_MODULE_ENABLED */
/* #define HAL_I2C_MODULE_ENABLED */
/* #define HAL_I2S_MODULE_ENABLED */
/* #define HAL_IWDG_MODULE_ENABLED */
/* #define HAL_LTDC_MODULE_ENABLED */
/* #define HAL_RNG_MODULE_ENABLED */
/* #define HAL_RTC_MODULE_ENABLED */
/* #define HAL_SAI_MODULE_ENABLED */
/* #define HAL_SD_MODULE_ENABLED */
/* #define HAL_MMC_MODULE_ENABLED */
/* #define HAL_SPI_MODULE_ENABLED */
/* #define HAL_TIM_MODULE_ENABLED */
#define HAL_UART_MODULE_ENABLED
/* #define HAL_USART_MODULE_ENABLED */
/* #define HAL_IRDA_MODULE_ENABLED */
/* #define HAL_SMARTCARD_MODULE_ENABLED */
/* #define HAL_SMBUS_MODULE_ENABLED */
/* #define HAL_WWDG_MODULE_ENABLED */
/* #define HAL_PCD_MODULE_ENABLED */
/* #define HAL_HCD_MODULE_ENABLED */
/* #define HAL_DSI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
```

```
/* #define HAL_CEC_MODULE_ENABLED */
/* #define HAL_FMPI2C_MODULE_ENABLED */
/* #define HAL_FMPSMBUS_MODULE_ENABLED */
/* #define HAL_SPDIFRX_MODULE_ENABLED */
/* #define HAL_DFSDM_MODULE_ENABLED */
/* #define HAL_LPTIM_MODULE_ENABLED */
#define HAL_GPIO_MODULE_ENABLED
#define HAL_EXTI_MODULE_ENABLED
#define HAL_DMA_MODULE_ENABLED
#define HAL_RCC_MODULE_ENABLED
#define HAL_FLASH_MODULE_ENABLED
#define HAL_PWR_MODULE_ENABLED
#define HAL_CORTEX_MODULE_ENABLED
/* ############################ HSE/HSI Values adaptation
###################### */
/**
  * @brief Adjust the value of External High Speed oscillator (HSE)
used in your application.
  *        This value is used by the RCC HAL module to compute the
system frequency
  *        (when HSE is used as system clock source, directly or through
the PLL).
  */
#if !defined  (HSE_VALUE)
```

```
  #define HSE_VALUE    8000000U /*!< Value of the External
oscillator in Hz */

#endif /* HSE_VALUE */

#if !defined  (HSE_STARTUP_TIMEOUT)

  #define HSE_STARTUP_TIMEOUT    100U   /*!< Time out for
HSE start up, in ms */

#endif /* HSE_STARTUP_TIMEOUT */

/**

  * @brief Internal High Speed oscillator (HSI) value.

  *        This value is used by the RCC HAL module to compute the
system frequency

  *        (when HSI is used as system clock source, directly or through the
PLL).

  */

#if !defined  (HSI_VALUE)

  #define HSI_VALUE    ((uint32_t)16000000U) /*!< Value of the
Internal oscillator in Hz*/

#endif /* HSI_VALUE */

/**

  * @brief Internal Low Speed oscillator (LSI) value.

  */

#if !defined  (LSI_VALUE)

 #define LSI_VALUE  32000U       /*!< LSI Typical Value in Hz*/

#endif /* LSI_VALUE */                    /*!< Value of the Internal Low
Speed oscillator in Hz
```

```
                                The real value may vary depending on the variations
                                in voltage and temperature.*/
/**
  * @brief External Low Speed oscillator (LSE) value.
  */
#if !defined  (LSE_VALUE)
 #define LSE_VALUE  32768U    /*!< Value of the External Low Speed oscillator in Hz */
#endif /* LSE_VALUE */
#if !defined  (LSE_STARTUP_TIMEOUT)
  #define LSE_STARTUP_TIMEOUT    5000U   /*!< Time out for LSE start up, in ms */
#endif /* LSE_STARTUP_TIMEOUT */
/**
  * @brief External clock source for I2S peripheral
  *        This value is used by the I2S HAL module to compute the I2S clock source
  *        frequency, this source is inserted directly through I2S_CKIN pad.
  */
#if !defined  (EXTERNAL_CLOCK_VALUE)
  #define EXTERNAL_CLOCK_VALUE    12288000U /*!< Value of the External audio frequency in Hz*/
#endif /* EXTERNAL_CLOCK_VALUE */
```

/* Tip: To avoid modifying this file each time you need to use different HSE,

=== you can define the HSE value in your toolchain compiler preprocessor. */

/* ############################## System Configuration ########################### */

/**

* @brief This is the HAL system configuration section

*/

#define VDD_VALUE 3300U /*!< Value of VDD in mv */

#define TICK_INT_PRIORITY 0U /*!< tick interrupt priority */

#define USE_RTOS 0U

#define PREFETCH_ENABLE 1U

#define INSTRUCTION_CACHE_ENABLE 1U

#define DATA_CACHE_ENABLE 1U

#define USE_HAL_ADC_REGISTER_CALLBACKS 0U /* ADC register callback disabled */

#define USE_HAL_CAN_REGISTER_CALLBACKS 0U /* CAN register callback disabled */

#define USE_HAL_CEC_REGISTER_CALLBACKS 0U /* CEC register callback disabled */

#define USE_HAL_CRYP_REGISTER_CALLBACKS 0U /* CRYP register callback disabled */

#define USE_HAL_DAC_REGISTER_CALLBACKS 0U /* DAC register callback disabled */

#define USE_HAL_DCMI_REGISTER_CALLBACKS 0U /* DCMI register callback disabled */

#define USE_HAL_DFSDM_REGISTER_CALLBACKS 0U /* DFSDM register callback disabled */

#define USE_HAL_DMA2D_REGISTER_CALLBACKS 0U /* DMA2D register callback disabled */

#define USE_HAL_DSI_REGISTER_CALLBACKS 0U /* DSI register callback disabled */

#define USE_HAL_ETH_REGISTER_CALLBACKS 0U /* ETH register callback disabled */

#define USE_HAL_HASH_REGISTER_CALLBACKS 0U /* HASH register callback disabled */

#define USE_HAL_HCD_REGISTER_CALLBACKS 0U /* HCD register callback disabled */

#define USE_HAL_I2C_REGISTER_CALLBACKS 0U /* I2C register callback disabled */

#define USE_HAL_FMPI2C_REGISTER_CALLBACKS 0U /* FMPI2C register callback disabled */

#define USE_HAL_FMPSMBUS_REGISTER_CALLBACKS 0U /* FMPSMBUS register callback disabled */

#define USE_HAL_I2S_REGISTER_CALLBACKS 0U /* I2S register callback disabled */

#define USE_HAL_IRDA_REGISTER_CALLBACKS 0U /* IRDA register callback disabled */

#define USE_HAL_LPTIM_REGISTER_CALLBACKS 0U /* LPTIM register callback disabled */

#define USE_HAL_LTDC_REGISTER_CALLBACKS 0U /* LTDC register callback disabled */

#define USE_HAL_MMC_REGISTER_CALLBACKS 0U /* MMC register callback disabled */

#define USE_HAL_NAND_REGISTER_CALLBACKS 0U /* NAND register callback disabled */

#define USE_HAL_NOR_REGISTER_CALLBACKS 0U /* NOR register callback disabled */

#define USE_HAL_PCCARD_REGISTER_CALLBACKS 0U /* PCCARD register callback disabled */

#define USE_HAL_PCD_REGISTER_CALLBACKS 0U /* PCD register callback disabled */

#define USE_HAL_QSPI_REGISTER_CALLBACKS 0U /* QSPI register callback disabled */

#define USE_HAL_RNG_REGISTER_CALLBACKS 0U /* RNG register callback disabled */

#define USE_HAL_RTC_REGISTER_CALLBACKS 0U /* RTC register callback disabled */

#define USE_HAL_SAI_REGISTER_CALLBACKS 0U /* SAI register callback disabled */

#define USE_HAL_SD_REGISTER_CALLBACKS 0U /* SD register callback disabled */

#define USE_HAL_SMARTCARD_REGISTER_CALLBACKS 0U /* SMARTCARD register callback disabled */

#define USE_HAL_SDRAM_REGISTER_CALLBACKS 0U /* SDRAM register callback disabled */

```
#define  USE_HAL_SRAM_REGISTER_CALLBACKS       0U /*
SRAM register callback disabled      */

#define  USE_HAL_SPDIFRX_REGISTER_CALLBACKS     0U /*
SPDIFRX register callback disabled   */

#define  USE_HAL_SMBUS_REGISTER_CALLBACKS       0U /*
SMBUS register callback disabled     */

#define  USE_HAL_SPI_REGISTER_CALLBACKS         0U /* SPI
register callback disabled       */

#define  USE_HAL_TIM_REGISTER_CALLBACKS         0U /*
TIM register callback disabled       */

#define  USE_HAL_UART_REGISTER_CALLBACKS        0U /*
UART register callback disabled      */

#define  USE_HAL_USART_REGISTER_CALLBACKS       0U /*
USART register callback disabled     */

#define  USE_HAL_WWDG_REGISTER_CALLBACKS        0U /*
WWDG register callback disabled      */

/* ############################ Assert Selection
################################## */

/**

  * @brief Uncomment the line below to expanse the "assert_param"
macro in the

  *        HAL drivers code

  */

/* #define USE_FULL_ASSERT    1U */

/* ################### Ethernet peripheral configuration
###################### */
```

```
/* Section 1 : Ethernet peripheral configuration */

/* MAC ADDRESS:
MAC_ADDR0:MAC_ADDR1:MAC_ADDR2:MAC_ADDR3:MAC_A
DDR4:MAC_ADDR5 */

#define MAC_ADDR0   2U

#define MAC_ADDR1   0U

#define MAC_ADDR2   0U

#define MAC_ADDR3   0U

#define MAC_ADDR4   0U

#define MAC_ADDR5   0U

/* Definition of the Ethernet driver buffers size and count */

#define ETH_RX_BUF_SIZE             ETH_MAX_PACKET_SIZE
/* buffer size for receive            */

#define ETH_TX_BUF_SIZE             ETH_MAX_PACKET_SIZE
/* buffer size for transmit           */

#define ETH_RXBUFNB                 4U      /* 4 Rx buffers of size
ETH_RX_BUF_SIZE  */

#define ETH_TXBUFNB                 4U      /* 4 Tx buffers of size
ETH_TX_BUF_SIZE  */

/* Section 2: PHY configuration section */

/* DP83848_PHY_ADDRESS Address*/

#define DP83848_PHY_ADDRESS

/* PHY Reset delay these values are based on a 1 ms Systick interrupt*/

#define PHY_RESET_DELAY             0x000000FFU

/* PHY Configuration delay */
```

```
#define PHY_CONFIG_DELAY          0x00000FFFU
#define PHY_READ_TO               0x0000FFFFU
#define PHY_WRITE_TO              0x0000FFFFU
/* Section 3: Common PHY Registers */
#define PHY_BCR                   ((uint16_t)0x0000U)    /*!<
Transceiver Basic Control Register   */
#define PHY_BSR                   ((uint16_t)0x0001U)    /*!<
Transceiver Basic Status Register    */
#define PHY_RESET                 ((uint16_t)0x8000U)  /*!< PHY
Reset */
#define PHY_LOOPBACK              ((uint16_t)0x4000U)  /*!<
Select loop-back mode */
#define PHY_FULLDUPLEX_100M       ((uint16_t)0x2100U)
/*!< Set the full-duplex mode at 100 Mb/s */
#define PHY_HALFDUPLEX_100M       ((uint16_t)0x2000U)
/*!< Set the half-duplex mode at 100 Mb/s */
#define PHY_FULLDUPLEX_10M        ((uint16_t)0x0100U)
/*!< Set the full-duplex mode at 10 Mb/s  */
#define PHY_HALFDUPLEX_10M        ((uint16_t)0x0000U)
/*!< Set the half-duplex mode at 10 Mb/s  */
#define PHY_AUTONEGOTIATION       ((uint16_t)0x1000U)
/*!< Enable auto-negotiation function     */
#define PHY_RESTART_AUTONEGOTIATION
((uint16_t)0x0200U)  /*!< Restart auto-negotiation function    */
#define PHY_POWERDOWN             ((uint16_t)0x0800U)  /*!<
Select the power down mode          */
```

```
#define PHY_ISOLATE                    ((uint16_t)0x0400U)  /*!< Isolate
PHY from MII                 */

#define PHY_AUTONEGO_COMPLETE          ((uint16_t)0x0020U)
/*!< Auto-Negotiation process completed   */

#define PHY_LINKED_STATUS              ((uint16_t)0x0004U)  /*!<
Valid link established               */

#define PHY_JABBER_DETECTION           ((uint16_t)0x0002U)
/*!< Jabber condition detected            */

/* Section 4: Extended PHY Registers */

#define PHY_SR                         ((uint16_t))    /*!< PHY status register
Offset                    */

#define PHY_SPEED_STATUS               ((uint16_t))  /*!< PHY Speed
mask                           */

#define PHY_DUPLEX_STATUS              ((uint16_t))  /*!< PHY
Duplex mask                            */

/* ################### SPI peripheral configuration
############################ */

/* CRC FEATURE: Use to activate CRC feature inside HAL SPI Driver

* Activated: CRC code is present inside driver

* Deactivated: CRC code cleaned from driver

*/

#define USE_SPI_CRC                    0U

/* Includes ------------------------------------------------------------------*/

/**

  * @brief Include module's header file
```

```
  */
#ifdef HAL_RCC_MODULE_ENABLED
  #include "stm32f4xx_hal_rcc.h"
#endif /* HAL_RCC_MODULE_ENABLED */
#ifdef HAL_GPIO_MODULE_ENABLED
  #include "stm32f4xx_hal_gpio.h"
#endif /* HAL_GPIO_MODULE_ENABLED */
#ifdef HAL_EXTI_MODULE_ENABLED
  #include "stm32f4xx_hal_exti.h"
#endif /* HAL_EXTI_MODULE_ENABLED */
#ifdef HAL_DMA_MODULE_ENABLED
  #include "stm32f4xx_hal_dma.h"
#endif /* HAL_DMA_MODULE_ENABLED */
#ifdef HAL_CORTEX_MODULE_ENABLED
  #include "stm32f4xx_hal_cortex.h"
#endif /* HAL_CORTEX_MODULE_ENABLED */
#ifdef HAL_ADC_MODULE_ENABLED
  #include "stm32f4xx_hal_adc.h"
#endif /* HAL_ADC_MODULE_ENABLED */
#ifdef HAL_CAN_MODULE_ENABLED
  #include "stm32f4xx_hal_can.h"
#endif /* HAL_CAN_MODULE_ENABLED */
#ifdef HAL_CAN_LEGACY_MODULE_ENABLED
```

```
  #include "stm32f4xx_hal_can_legacy.h"
#endif /* HAL_CAN_LEGACY_MODULE_ENABLED */
#ifdef HAL_CRC_MODULE_ENABLED
  #include "stm32f4xx_hal_crc.h"
#endif /* HAL_CRC_MODULE_ENABLED */
#ifdef HAL_CRYP_MODULE_ENABLED
  #include "stm32f4xx_hal_cryp.h"
#endif /* HAL_CRYP_MODULE_ENABLED */
#ifdef HAL_DMA2D_MODULE_ENABLED
  #include "stm32f4xx_hal_dma2d.h"
#endif /* HAL_DMA2D_MODULE_ENABLED */
#ifdef HAL_DAC_MODULE_ENABLED
  #include "stm32f4xx_hal_dac.h"
#endif /* HAL_DAC_MODULE_ENABLED */
#ifdef HAL_DCMI_MODULE_ENABLED
  #include "stm32f4xx_hal_dcmi.h"
#endif /* HAL_DCMI_MODULE_ENABLED */
#ifdef HAL_ETH_MODULE_ENABLED
  #include "stm32f4xx_hal_eth.h"
#endif /* HAL_ETH_MODULE_ENABLED */
#ifdef HAL_ETH_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_eth_legacy.h"
#endif /* HAL_ETH_LEGACY_MODULE_ENABLED */
```

```
#ifdef HAL_FLASH_MODULE_ENABLED
  #include "stm32f4xx_hal_flash.h"
#endif /* HAL_FLASH_MODULE_ENABLED */
#ifdef HAL_SRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sram.h"
#endif /* HAL_SRAM_MODULE_ENABLED */
#ifdef HAL_NOR_MODULE_ENABLED
  #include "stm32f4xx_hal_nor.h"
#endif /* HAL_NOR_MODULE_ENABLED */
#ifdef HAL_NAND_MODULE_ENABLED
  #include "stm32f4xx_hal_nand.h"
#endif /* HAL_NAND_MODULE_ENABLED */
#ifdef HAL_PCCARD_MODULE_ENABLED
  #include "stm32f4xx_hal_pccard.h"
#endif /* HAL_PCCARD_MODULE_ENABLED */
#ifdef HAL_SDRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sdram.h"
#endif /* HAL_SDRAM_MODULE_ENABLED */
#ifdef HAL_HASH_MODULE_ENABLED
 #include "stm32f4xx_hal_hash.h"
#endif /* HAL_HASH_MODULE_ENABLED */
#ifdef HAL_I2C_MODULE_ENABLED
 #include "stm32f4xx_hal_i2c.h"
```

```
#endif /* HAL_I2C_MODULE_ENABLED */
#ifdef HAL_SMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_smbus.h"
#endif /* HAL_SMBUS_MODULE_ENABLED */
#ifdef HAL_I2S_MODULE_ENABLED
 #include "stm32f4xx_hal_i2s.h"
#endif /* HAL_I2S_MODULE_ENABLED */
#ifdef HAL_IWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_iwdg.h"
#endif /* HAL_IWDG_MODULE_ENABLED */
#ifdef HAL_LTDC_MODULE_ENABLED
 #include "stm32f4xx_hal_ltdc.h"
#endif /* HAL_LTDC_MODULE_ENABLED */
#ifdef HAL_PWR_MODULE_ENABLED
 #include "stm32f4xx_hal_pwr.h"
#endif /* HAL_PWR_MODULE_ENABLED */
#ifdef HAL_RNG_MODULE_ENABLED
 #include "stm32f4xx_hal_rng.h"
#endif /* HAL_RNG_MODULE_ENABLED */
#ifdef HAL_RTC_MODULE_ENABLED
 #include "stm32f4xx_hal_rtc.h"
#endif /* HAL_RTC_MODULE_ENABLED */
#ifdef HAL_SAI_MODULE_ENABLED
```

```
 #include "stm32f4xx_hal_sai.h"
#endif /* HAL_SAI_MODULE_ENABLED */
#ifdef HAL_SD_MODULE_ENABLED
 #include "stm32f4xx_hal_sd.h"
#endif /* HAL_SD_MODULE_ENABLED */
#ifdef HAL_SPI_MODULE_ENABLED
 #include "stm32f4xx_hal_spi.h"
#endif /* HAL_SPI_MODULE_ENABLED */
#ifdef HAL_TIM_MODULE_ENABLED
 #include "stm32f4xx_hal_tim.h"
#endif /* HAL_TIM_MODULE_ENABLED */
#ifdef HAL_UART_MODULE_ENABLED
 #include "stm32f4xx_hal_uart.h"
#endif /* HAL_UART_MODULE_ENABLED */
#ifdef HAL_USART_MODULE_ENABLED
 #include "stm32f4xx_hal_usart.h"
#endif /* HAL_USART_MODULE_ENABLED */
#ifdef HAL_IRDA_MODULE_ENABLED
 #include "stm32f4xx_hal_irda.h"
#endif /* HAL_IRDA_MODULE_ENABLED */
#ifdef HAL_SMARTCARD_MODULE_ENABLED
 #include "stm32f4xx_hal_smartcard.h"
#endif /* HAL_SMARTCARD_MODULE_ENABLED */
```

```
#ifdef HAL_WWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_wwdg.h"
#endif /* HAL_WWDG_MODULE_ENABLED */
#ifdef HAL_PCD_MODULE_ENABLED
 #include "stm32f4xx_hal_pcd.h"
#endif /* HAL_PCD_MODULE_ENABLED */
#ifdef HAL_HCD_MODULE_ENABLED
 #include "stm32f4xx_hal_hcd.h"
#endif /* HAL_HCD_MODULE_ENABLED */
#ifdef HAL_DSI_MODULE_ENABLED
 #include "stm32f4xx_hal_dsi.h"
#endif /* HAL_DSI_MODULE_ENABLED */
#ifdef HAL_QSPI_MODULE_ENABLED
 #include "stm32f4xx_hal_qspi.h"
#endif /* HAL_QSPI_MODULE_ENABLED */
#ifdef HAL_CEC_MODULE_ENABLED
 #include "stm32f4xx_hal_cec.h"
#endif /* HAL_CEC_MODULE_ENABLED */
#ifdef HAL_FMPI2C_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpi2c.h"
#endif /* HAL_FMPI2C_MODULE_ENABLED */
#ifdef HAL_FMPSMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpsmbus.h"
```

```
#endif /* HAL_FMPSMBUS_MODULE_ENABLED */
#ifdef HAL_SPDIFRX_MODULE_ENABLED
 #include "stm32f4xx_hal_spdifrx.h"
#endif /* HAL_SPDIFRX_MODULE_ENABLED */
#ifdef HAL_DFSDM_MODULE_ENABLED
 #include "stm32f4xx_hal_dfsdm.h"
#endif /* HAL_DFSDM_MODULE_ENABLED */
#ifdef HAL_LPTIM_MODULE_ENABLED
 #include "stm32f4xx_hal_lptim.h"
#endif /* HAL_LPTIM_MODULE_ENABLED */
#ifdef HAL_MMC_MODULE_ENABLED
 #include "stm32f4xx_hal_mmc.h"
#endif /* HAL_MMC_MODULE_ENABLED */
/* Exported macro ------------------------------------------------------------*/
#ifdef  USE_FULL_ASSERT
/**
  * @brief  The assert_param macro is used for function's parameters
check.
  * @param  expr If expr is false, it calls assert_failed function
  *         which reports the name of the source file and the source
  *         line number of the call that failed.
  *         If expr is true, it returns no value.
  * @retval None
```

```
  */
  #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t
*)__FILE__, __LINE__))
/* Exported functions ---------------------------------------------------- */
  void assert_failed(uint8_t* file, uint32_t line);
#else
  #define assert_param(expr) ((void)0U)
#endif /* USE_FULL_ASSERT */
#ifdef __cplusplus
}
#endif
#endif /* __STM32F4xx_HAL_CONF_H */
```

18.5 CODE OF STM32F4XX_IT.H:

```
/* USER CODE BEGIN Header */
/**
*************************************************************
********************
  * @file    stm32f4xx_it.h
  * @brief   This file contains the headers of the interrupt handlers.
*************************************************************
********************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
```

```
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_IT_H
#define __STM32F4xx_IT_H
#ifdef __cplusplus
 extern "C" {
#endif
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants --------------------------------------------------------*/
```

```
/* USER CODE BEGIN EC */
/* USER CODE END EC */
/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */
/* USER CODE END EM */
/* Exported functions prototypes ---------------------------------------------*/
void NMI_Handler(void);
void HardFault_Handler(void);
void MemManage_Handler(void);
void BusFault_Handler(void);
void UsageFault_Handler(void);
void SVC_Handler(void);
void DebugMon_Handler(void);
void PendSV_Handler(void);
void SysTick_Handler(void);
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
#ifdef __cplusplus
}
#endif
#endif /* __STM32F4xx_IT_H */
```

18.6 CODE OF MAIN.C:

```
/* USER CODE BEGIN Header */
```

```
/**
**********************************************************
*******************
  * @file           : main.c
  * @brief          : Main program body
**********************************************************
*******************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
```

```
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
/* USER CODE END PTD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
UART_HandleTypeDef huart2;
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
```

```
/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */
  /* USER CODE END 1 */
  /* MCU Configuration--------------------------------------------------------*/
  /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
  HAL_Init();
  /* USER CODE BEGIN Init */
  /* USER CODE END Init */
  /* Configure the system clock */
  SystemClock_Config();
  /* USER CODE BEGIN SysInit */
  /* USER CODE END SysInit */
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  /* USER CODE BEGIN 2 */
  /* USER CODE END 2 */
```

```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */


    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_
VOLTAGE_SCALE1);
```

```
  /** Initializes the RCC Oscillators according to the specified
parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue =
RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 16;
  RCC_OscInitStruct.PLL.PLLN = 336;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV4;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK

|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
```

```
  RCC_ClkInitStruct.SYSCLKSource =
RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct,
FLASH_LATENCY_2) != HAL_OK)
  {
    Error_Handler();
  }
}
/**
  * @brief USART2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART2_UART_Init(void)
{
  /* USER CODE BEGIN USART2_Init 0 */
  /* USER CODE END USART2_Init 0 */
  /* USER CODE BEGIN USART2_Init 1 */
  /* USER CODE END USART2_Init 1 */
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 115200;
```

```
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART2_Init 2 */
  /* USER CODE END USART2_Init 2 */
}
/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */
```

```
  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin,
GPIO_PIN_RESET);
  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);
  /*Configure GPIO pin : LD2_Pin */
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);
/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}
/* USER CODE BEGIN 4 */
```

```
/* USER CODE END 4 */
/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return
state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}
#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line
number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
```

```
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

18.7 CODE OF STM32F4XX_HAL_MSP.C:

```
/* USER CODE BEGIN Header */
/**
**********************************************************************
  * @file        stm32f4xx_hal_msp.c
  * @brief       This file provides code for the MSP Initialization
  *              and de-Initialization codes.
**********************************************************************
  * @attention
```

```
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *

**********************************************************
********************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN TD */
/* USER CODE END TD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN Define */
/* USER CODE END Define */
```

```
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN Macro */
/* USER CODE END Macro */
/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* External functions --------------------------------------------------------*/
/* USER CODE BEGIN ExternalFunctions */
/* USER CODE END ExternalFunctions */
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/**
  * Initializes the Global MSP.
  */
void HAL_MspInit(void)
{
  /* USER CODE BEGIN MspInit 0 */
  /* USER CODE END MspInit 0 */
  __HAL_RCC_SYSCFG_CLK_ENABLE();
  __HAL_RCC_PWR_CLK_ENABLE();
```

```
  HAL_NVIC_SetPriorityGrouping(NVIC_PRIORITYGROUP_0);
  /* System interrupt init*/
  /* USER CODE BEGIN MspInit 1 */
  /* USER CODE END MspInit 1 */
}
/**
* @brief UART MSP Initialization
* This function configures the hardware resources used in this example
* @param huart: UART handle pointer
* @retval None
*/
void HAL_UART_MspInit(UART_HandleTypeDef* huart)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  if(huart->Instance==USART2)
  {
  /* USER CODE BEGIN USART2_MspInit 0 */
  /* USER CODE END USART2_MspInit 0 */
    /* Peripheral clock enable */
    __HAL_RCC_USART2_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();
    /**USART2 GPIO Configuration
    PA2     ------> USART2_TX
```

```
    PA3     ------> USART2_RX
    */
    GPIO_InitStruct.Pin = USART_TX_Pin|USART_RX_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF7_USART2;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
  /* USER CODE BEGIN USART2_MspInit 1 */
  /* USER CODE END USART2_MspInit 1 */
  }
}
/**
* @brief UART MSP De-Initialization
* This function freeze the hardware resources used in this example
* @param huart: UART handle pointer
* @retval None
*/
void HAL_UART_MspDeInit(UART_HandleTypeDef* huart)
{
  if(huart->Instance==USART2)
  {
  /* USER CODE BEGIN USART2_MspDeInit 0 */
```

```
  /* USER CODE END USART2_MspDeInit 0 */
    /* Peripheral clock disable */
    __HAL_RCC_USART2_CLK_DISABLE();
    /**USART2 GPIO Configuration
    PA2     ------> USART2_TX
    PA3     ------> USART2_RX
    */
    HAL_GPIO_DeInit(GPIOA, USART_TX_Pin|USART_RX_Pin);
  /* USER CODE BEGIN USART2_MspDeInit 1 */
  /* USER CODE END USART2_MspDeInit 1 */
  }
}
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

18.8 CODE OF STM32F4XX_IT.C:

```
/* USER CODE BEGIN Header */
/**

**********************************************************
*******************
  * @file    stm32f4xx_it.c
  * @brief   Interrupt Service Routines.
```

```
**********************************************************
*******************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *

**********************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stm32f4xx_it.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
```

```
/* USER CODE BEGIN TD */
/* USER CODE END TD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/* External variables --------------------------------------------------------*/
/* USER CODE BEGIN EV */
/* USER CODE END EV */
/******************************************************************************/
/*          Cortex-M4 Processor Interruption and Exception Handlers          */
```

```
/*****************************************************
*********************/
/**
  * @brief This function handles Non maskable interrupt.
  */
void NMI_Handler(void)
{
  /* USER CODE BEGIN NonMaskableInt_IRQn 0 */
  /* USER CODE END NonMaskableInt_IRQn 0 */
  /* USER CODE BEGIN NonMaskableInt_IRQn 1 */
   while (1)
  {
  }
  /* USER CODE END NonMaskableInt_IRQn 1 */
}
/**
  * @brief This function handles Hard fault interrupt.
  */
void HardFault_Handler(void)
{
  /* USER CODE BEGIN HardFault_IRQn 0 */
  /* USER CODE END HardFault_IRQn 0 */
  while (1)
```

```
  {
    /* USER CODE BEGIN W1_HardFault_IRQn 0 */
    /* USER CODE END W1_HardFault_IRQn 0 */
  }
}
/**
  * @brief This function handles Memory management fault.
  */
void MemManage_Handler(void)
{
  /* USER CODE BEGIN MemoryManagement_IRQn 0 */
  /* USER CODE END MemoryManagement_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_MemoryManagement_IRQn 0 */
    /* USER CODE END W1_MemoryManagement_IRQn 0 */
  }
}
/**
  * @brief This function handles Pre-fetch fault, memory access fault.
  */
void BusFault_Handler(void)
{
```

```
  /* USER CODE BEGIN BusFault_IRQn 0 */
  /* USER CODE END BusFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_BusFault_IRQn 0 */
    /* USER CODE END W1_BusFault_IRQn 0 */
  }
}
/**
  * @brief This function handles Undefined instruction or illegal state.
  */
void UsageFault_Handler(void)
{
  /* USER CODE BEGIN UsageFault_IRQn 0 */
  /* USER CODE END UsageFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_UsageFault_IRQn 0 */
    /* USER CODE END W1_UsageFault_IRQn 0 */
  }
}
/**
```

```
  * @brief This function handles System service call via SWI
instruction.
  */
void SVC_Handler(void)
{
  /* USER CODE BEGIN SVCall_IRQn 0 */
  /* USER CODE END SVCall_IRQn 0 */
  /* USER CODE BEGIN SVCall_IRQn 1 */
  /* USER CODE END SVCall_IRQn 1 */
}
/**
  * @brief This function handles Debug monitor.
  */
void DebugMon_Handler(void)
{
  /* USER CODE BEGIN DebugMonitor_IRQn 0 */
  /* USER CODE END DebugMonitor_IRQn 0 */
  /* USER CODE BEGIN DebugMonitor_IRQn 1 */
  /* USER CODE END DebugMonitor_IRQn 1 */
}
/**
  * @brief This function handles Pendable request for system service.
  */
```

```
void PendSV_Handler(void)
{
  /* USER CODE BEGIN PendSV_IRQn 0 */
  /* USER CODE END PendSV_IRQn 0 */
  /* USER CODE BEGIN PendSV_IRQn 1 */
  /* USER CODE END PendSV_IRQn 1 */
}
/**
  * @brief This function handles System tick timer.
  */
void SysTick_Handler(void)
{
  /* USER CODE BEGIN SysTick_IRQn 0 */
  /* USER CODE END SysTick_IRQn 0 */
  HAL_IncTick();
  /* USER CODE BEGIN SysTick_IRQn 1 */
  /* USER CODE END SysTick_IRQn 1 */
}
/******************************************************
*********************/
/* STM32F4xx Peripheral Interrupt Handlers                    */
/* Add here the Interrupt Handlers for the used peripherals.  */
/* For the available peripheral interrupt handler names,      */
```

```
/* please refer to the startup file (startup_stm32f4xx.s).                    */
/*****************************************************************************/
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

18.9 CODE OF SYSCALLS.C:

```
/**
*****************************************************************************
 * @file      syscalls.c
 * @author    Auto-generated by STM32CubeIDE
 * @brief     STM32CubeIDE Minimal System calls file
 *
 *            For more information about which c-functions
 *            need which of these lowlevel functions
 *            please consult the Newlib libc-manual
*****************************************************************************
 * @attention
 *
 * Copyright (c) 2020-2024 STMicroelectronics.
 * All rights reserved.
```

```
 *
 * This software is licensed under terms that can be found in the
LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *

**********************************************************
*******************
 */
/* Includes */
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
#include <sys/times.h>
/* Variables */
extern int __io_putchar(int ch) __attribute__((weak));
extern int __io_getchar(void) __attribute__((weak));
char *__env[1] = { 0 };
char **environ = __env;
```

```
/* Functions */
void initialise_monitor_handles()
{
}
int _getpid(void)
{
  return 1;
}
int _kill(int pid, int sig)
{
  (void)pid;
  (void)sig;
  errno = EINVAL;
  return -1;
}
void _exit (int status)
{
  _kill(status, -1);
  while (1) {}    /* Make sure we hang here */
}
__attribute__((weak)) int _read(int file, char *ptr, int len)
{
  (void)file;
```

```
  int DataIdx;
  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    *ptr++ = __io_getchar();
  }
  return len;
}
__attribute__((weak)) int _write(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;
  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    __io_putchar(*ptr++);
  }
  return len;
}
int _close(int file)
{
  (void)file;
  return -1;
}
int _fstat(int file, struct stat *st)
```

```
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
int _isatty(int file)
{
  (void)file;
  return 1;
}
int _lseek(int file, int ptr, int dir)
{
  (void)file;
  (void)ptr;
  (void)dir;
  return 0;
}
int _open(char *path, int flags, ...)
{
  (void)path;
  (void)flags;
  /* Pretend like we always fail */
  return -1;
```

```
}
int _wait(int *status)
{
  (void)status;
  errno = ECHILD;
  return -1;
}
int _unlink(char *name)
{
  (void)name;
  errno = ENOENT;
  return -1;
}
int _times(struct tms *buf)
{
  (void)buf;
  return -1;
}
int _stat(char *file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
```

```
}
int _link(char *old, char *new)
{
  (void)old;
  (void)new;
  errno = EMLINK;
  return -1;
}
int _fork(void)
{
  errno = EAGAIN;
  return -1;
}
int _execve(char *name, char **argv, char **env)
{
  (void)name;
  (void)argv;
  (void)env;
  errno = ENOMEM;
  return -1;
}
```

18.10 CODE OF SYSTEM.C:

```
/**
*****************************************************************
*******************
 * @file      sysmem.c
 * @author    Generated by STM32CubeIDE
 * @brief     STM32CubeIDE System Memory calls file
 *
 *            For more information about which C functions
 *            need which of these lowlevel functions
 *            please consult the newlib libc manual
*****************************************************************
*******************
 * @attention
 *
 * Copyright (c) 2024 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the
LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
*****************************************************************
*******************
```

```
 */
/* Includes */
#include <errno.h>
#include <stdint.h>
/**
 * Pointer to the current high watermark of the heap usage
 */
static uint8_t *__sbrk_heap_end = NULL;
/**
 * @brief _sbrk() allocates memory to the newlib heap and is used by malloc
 *        and others from the C library
 *
 * @verbatim
 *
##################################################################
 * #  .data  #  .bss  #       newlib heap       #          MSP stack          #
 * #         #        #                          # Reserved by _Min_Stack_Size #
 *
##################################################################
 * ^-- RAM start      ^-- _end                          _estack, RAM end --^
 * @endverbatim
 *
```

```
 * This implementation starts allocating at the '_end' linker symbol
 * The '_Min_Stack_Size' linker symbol reserves a memory for the MSP stack
 * The implementation considers '_estack' linker symbol to be RAM end
 * NOTE: If the MSP stack, at any point during execution, grows larger than the
 * reserved size, please increase the '_Min_Stack_Size'.
 *
 * @param incr Memory size
 * @return Pointer to allocated memory
 */
void *_sbrk(ptrdiff_t incr)
{
  extern uint8_t _end; /* Symbol defined in the linker script */
  extern uint8_t _estack; /* Symbol defined in the linker script */
  extern uint32_t _Min_Stack_Size; /* Symbol defined in the linker script */
  const uint32_t stack_limit = (uint32_t)&_estack - (uint32_t)&_Min_Stack_Size;
  const uint8_t *max_heap = (uint8_t *)stack_limit;
  uint8_t *prev_heap_end;
  /* Initialize heap end at first call */
  if (NULL == __sbrk_heap_end)
  {
```

```
    __sbrk_heap_end = &_end;
  }
  /* Protect heap from growing into the reserved MSP stack */
  if (__sbrk_heap_end + incr > max_heap)
  {
    errno = ENOMEM;
    return (void *)-1;
  }
  prev_heap_end = __sbrk_heap_end;
  __sbrk_heap_end += incr;
  return (void *)prev_heap_end;
}
```

18.11 CODE OF SYSTEM_STM32F4XX.C:

```
/**
*****************************************************************
*******************
  * @file    system_stm32f4xx.c
  * @author  MCD Application Team
  * @brief   CMSIS Cortex-M4 Device Peripheral Access Layer System
Source File.
  *
  *   This file provides two functions and one global variable to be called
from
  *   user application:
```

```
  *      - SystemInit(): This function is called at startup just after reset and
  *                      before branch to main program. This call is made inside
  *                      the "startup_stm32f4xx.s" file.
  *
  *      - SystemCoreClock variable: Contains the core clock (HCLK), it
can be used
  *                              by the user application to setup the SysTick
  *                              timer or configure other parameters.
  *
  *      - SystemCoreClockUpdate(): Updates the variable
SystemCoreClock and must
  *                             be called whenever the core clock is changed
  *                             during program execution.
  *
  *

**************************************************************
*******************
  * @attention
  *
  * Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
```

```
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*****************************************************************
*******************
  */
/** @addtogroup CMSIS
  * @{
  */
/** @addtogroup stm32f4xx_system
  * @{
  */
/** @addtogroup STM32F4xx_System_Private_Includes
  * @{
  */
#include "stm32f4xx.h"
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    ((uint32_t)25000000) /*!< Default value of
the External oscillator in Hz */
#endif /* HSE_VALUE */
#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000) /*!< Value of the
Internal oscillator in Hz*/
#endif /* HSI_VALUE */
```

```
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_TypesDefinitions
  * @{
  */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Defines
  * @{
  */
/************************* Miscellaneous Configuration
************************/
/*!< Uncomment the following line if you need to use external SRAM or
SDRAM as data memory  */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx) || defined(STM32F417xx)\
 || defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx) ||
defined(STM32F412Zx) || defined(STM32F412Vx)
/* #define DATA_IN_ExtSRAM */
#endif /* STM32F40xxx || STM32F41xxx || STM32F42xxx ||
STM32F43xxx || STM32F469xx || STM32F479xx ||\
```

```
        STM32F412Zx || STM32F412Vx */
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F446xx) || defined(STM32F469xx) ||
defined(STM32F479xx)
/* #define DATA_IN_ExtSDRAM */
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F446xx || STM32F469xx ||\
        STM32F479xx */
/* Note: Following vector table addresses must be defined in line with
linker
        configuration. */
/*!< Uncomment the following line if you need to relocate the vector
table
    anywhere in Flash or Sram, else the vector table is kept at the
automatic
    remap of boot address selected */
/* #define USER_VECT_TAB_ADDRESS */
#if defined(USER_VECT_TAB_ADDRESS)
/*!< Uncomment the following line if you need to relocate your vector
Table
    in Sram else user remap will be done in Flash. */
/* #define VECT_TAB_SRAM */
#if defined(VECT_TAB_SRAM)
```

```
#define VECT_TAB_BASE_ADDRESS   SRAM_BASE       /*!< Vector Table base address field.
                                                This value must be a multiple of 0x200. */
#define VECT_TAB_OFFSET         0x00000000U     /*!< Vector Table base offset field.
                                                This value must be a multiple of 0x200. */
#else
#define VECT_TAB_BASE_ADDRESS   FLASH_BASE      /*!< Vector Table base address field.
                                                This value must be a multiple of 0x200. */
#define VECT_TAB_OFFSET         0x00000000U     /*!< Vector Table base offset field.
                                                This value must be a multiple of 0x200. */
#endif /* VECT_TAB_SRAM */
#endif /* USER_VECT_TAB_ADDRESS */
/*****************************************************************************/
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Macros
  * @{
```

```
  */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Variables
  * @{
  */
  /* This variable is updated in three ways:
     1) by calling CMSIS function SystemCoreClockUpdate()
     2) by calling HAL API function HAL_RCC_GetHCLKFreq()
     3) each time HAL_RCC_ClockConfig() is called to configure the system clock frequency
        Note: If you use this function to configure the system clock; then there
              is no need to call the 2 first functions listed above, since SystemCoreClock
              variable is updated automatically.
  */
uint32_t SystemCoreClock = 16000000;
const uint8_t AHBPrescTable[16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9};
const uint8_t APBPrescTable[8]  = {0, 0, 0, 0, 1, 2, 3, 4};
/**
  * @}
```

```
  */
/** @addtogroup STM32F4xx_System_Private_FunctionPrototypes
  * @{
  */
#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  static void SystemInit_ExtMemCtl(void);
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Functions
  * @{
  */
/**
  * @brief  Setup the microcontroller system
  *         Initialize the FPU setting, vector table location and External memory
  *         configuration.
  * @param  None
  * @retval None
  */
void SystemInit(void)
{
```

```
  /* FPU settings ------------------------------------------------------------*/
  #if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
    SCB->CPACR |= ((3UL << 10*2)|(3UL << 11*2));  /* set CP10 and
CP11 Full Access */
  #endif
#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  SystemInit_ExtMemCtl();
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */
  /* Configure the Vector Table location -------------------------------------*/
#if defined(USER_VECT_TAB_ADDRESS)
  SCB->VTOR = VECT_TAB_BASE_ADDRESS |
VECT_TAB_OFFSET; /* Vector Table Relocation in Internal SRAM */
#endif /* USER_VECT_TAB_ADDRESS */
}
/**
   * @brief  Update SystemCoreClock variable according to Clock
Register Values.
  *         The SystemCoreClock variable contains the core clock (HCLK),
it can
  *         be used by the user application to setup the SysTick timer or
configure
  *         other parameters.
  *
  * @note   Each time the core clock (HCLK) changes, this function
must be called
```

```
  *        to update SystemCoreClock variable value. Otherwise, any configuration
  *        based on this variable will be incorrect.
  *
  * @note   - The system frequency computed by this function is not the real
  *           frequency in the chip. It is calculated based on the predefined
  *           constant and the selected clock source:
  *
  *           - If SYSCLK source is HSI, SystemCoreClock will contain the HSI_VALUE(*)
  *
  *           - If SYSCLK source is HSE, SystemCoreClock will contain the HSE_VALUE(**)
  *
  *           - If SYSCLK source is PLL, SystemCoreClock will contain the HSE_VALUE(**)
  *             or HSI_VALUE(*) multiplied/divided by the PLL factors.
  *
  *         (*) HSI_VALUE is a constant defined in stm32f4xx_hal_conf.h file (default value
  *             16 MHz) but the real value may vary depending on the variations
  *             in voltage and temperature.
  *
```

```
  *        (**) HSE_VALUE is a constant defined in
stm32f4xx_hal_conf.h file (its value
  *             depends on the application requirements), user has to ensure
that HSE_VALUE
  *             is same as the real frequency of the crystal used. Otherwise,
this function
  *             may have wrong result.
  *
  *        - The result of this function could be not correct when using
fractional
  *          value for HSE crystal.
  *
  * @param  None
  * @retval None
  */
void SystemCoreClockUpdate(void)
{
  uint32_t tmp = 0, pllvco = 0, pllp = 2, pllsource = 0, pllm = 2;
  /* Get SYSCLK source -------------------------------------------------------
*/
  tmp = RCC->CFGR & RCC_CFGR_SWS;
  switch (tmp)
  {
    case 0x00:  /* HSI used as system clock source */
      SystemCoreClock = HSI_VALUE;
```

```
    break;
  case 0x04:  /* HSE used as system clock source */
    SystemCoreClock = HSE_VALUE;
    break;
  case 0x08:  /* PLL used as system clock source */
    /* PLL_VCO = (HSE_VALUE or HSI_VALUE / PLL_M) * PLL_N
       SYSCLK = PLL_VCO / PLL_P
       */
    pllsource = (RCC->PLLCFGR & RCC_PLLCFGR_PLLSRC) >>
22;
    pllm = RCC->PLLCFGR & RCC_PLLCFGR_PLLM;
    if (pllsource != 0)
    {
      /* HSE used as PLL clock source */
      pllvco = (HSE_VALUE / pllm) * ((RCC->PLLCFGR &
RCC_PLLCFGR_PLLN) >> 6);
    }
    else
    {
      /* HSI used as PLL clock source */
      pllvco = (HSI_VALUE / pllm) * ((RCC->PLLCFGR &
RCC_PLLCFGR_PLLN) >> 6);
    }
```

```
    pllp = (((RCC->PLLCFGR & RCC_PLLCFGR_PLLP) >>16) + 1 )
*2;
    SystemCoreClock = pllvco/pllp;
    break;
   default:
    SystemCoreClock = HSI_VALUE;
    break;
  }
  /* Compute HCLK frequency ----------------------------------------------
-*/
  /* Get HCLK prescaler */
  tmp = AHBPrescTable[((RCC->CFGR & RCC_CFGR_HPRE) >> 4)];
  /* HCLK frequency */
  SystemCoreClock >>= tmp;
}
#if defined (DATA_IN_ExtSRAM) && defined
(DATA_IN_ExtSDRAM)
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
/**
  * @brief  Setup the external memory controller.
  *         Called in startup_stm32f4xx.s before jump to main.
  *         This function configures the external memories
(SRAM/SDRAM)
```

```
  *        This SRAM/SDRAM will be used as program data memory
(including heap and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
  register __IO uint32_t index;
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and
GPIOI interface clock */
  RCC->AHB1ENR |= 0x000001F8;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x00CCC0CC;
  GPIOD->AFR[1]  = 0xCCCCCCCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xAAAA0A8A;
  /* Configure PDx pins speed to 100 MHz */
  GPIOD->OSPEEDR = 0xFFFF0FCF;
  /* Configure PDx pins Output type to push-pull */
  GPIOD->OTYPER  = 0x00000000;
```

```
/* No pull-up, pull-down for PDx pins */
GPIOD->PUPDR   = 0x00000000;
/* Connect PEx pins to FMC Alternate function */
GPIOE->AFR[0]  = 0xC00CC0CC;
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA828A;
/* Configure PEx pins speed to 100 MHz */
GPIOE->OSPEEDR = 0xFFFFC3CF;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0xCCCCCCCC;
GPIOF->AFR[1]  = 0xCCCCCCCC;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA800AAA;
/* Configure PFx pins speed to 50 MHz */
GPIOF->OSPEEDR = 0xAA800AAA;
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
```

```
GPIOF->PUPDR   = 0x00000000;
/* Connect PGx pins to FMC Alternate function */
GPIOG->AFR[0]  = 0xCCCCCCCC;
GPIOG->AFR[1]  = 0xCCCCCCCC;
/* Configure PGx pins in Alternate function mode */
GPIOG->MODER   = 0xAAAAAAAA;
/* Configure PGx pins speed to 50 MHz */
GPIOG->OSPEEDR = 0xAAAAAAAA;
/* Configure PGx pins Output type to push-pull */
GPIOG->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PGx pins */
GPIOG->PUPDR   = 0x00000000;
/* Connect PHx pins to FMC Alternate function */
GPIOH->AFR[0]  = 0x00C0CC00;
GPIOH->AFR[1]  = 0xCCCCCCCC;
/* Configure PHx pins in Alternate function mode */
GPIOH->MODER   = 0xAAAA08A0;
/* Configure PHx pins speed to 50 MHz */
GPIOH->OSPEEDR = 0xAAAA08A0;
/* Configure PHx pins Output type to push-pull */
GPIOH->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PHx pins */
GPIOH->PUPDR   = 0x00000000;
```

```
  /* Connect PIx pins to FMC Alternate function */
  GPIOI->AFR[0]  = 0xCCCCCCCC;
  GPIOI->AFR[1]  = 0x00000CC0;
  /* Configure PIx pins in Alternate function mode */
  GPIOI->MODER   = 0x0028AAAA;
  /* Configure PIx pins speed to 50 MHz */
  GPIOI->OSPEEDR = 0x0028AAAA;
  /* Configure PIx pins Output type to push-pull */
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;
/*-- FMC Configuration ------------------------------------------------------*/
  /* Enable the FMC interface clock */
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
  FMC_Bank5_6->SDTR[0] = 0x01115351;
  /* SDRAM initialization sequence */
  /* Clock enable command */
  FMC_Bank5_6->SDCMR = 0x00000011;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  while((tmpreg != 0) && (timeout-- > 0))
```

```
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* Delay */
for (index = 0; index<1000; index++);
/* PALL command */
FMC_Bank5_6->SDCMR = 0x00000012;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* Auto refresh command */
FMC_Bank5_6->SDCMR = 0x00000073;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* MRD register program */
FMC_Bank5_6->SDCMR = 0x00046014;
```

```
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Set refresh count */
  tmpreg = FMC_Bank5_6->SDRTR;
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
  /* Disable write protection */
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
```

```
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F469xx || STM32F479xx */


  (void)(tmp);
}
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F469xx || STM32F479xx */
#elif defined (DATA_IN_ExtSRAM) || defined
(DATA_IN_ExtSDRAM)
/**
  * @brief  Setup the external memory controller.
  *         Called in startup_stm32f4xx.s before jump to main.
  *         This function configures the external memories
(SRAM/SDRAM)
  *         This SRAM/SDRAM will be used as program data memory
(including heap and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
```

```
 || defined(STM32F446xx) || defined(STM32F469xx) ||
defined(STM32F479xx)
#if defined (DATA_IN_ExtSDRAM)
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
  register __IO uint32_t index;
#if defined(STM32F446xx)
  /* Enable GPIOA, GPIOC, GPIOD, GPIOE, GPIOF, GPIOG interface
     clock */
  RCC->AHB1ENR |= 0x0000007D;
#else
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and
GPIOI interface
     clock */
  RCC->AHB1ENR |= 0x000001F8;
#endif /* STM32F446xx */
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);
#if defined(STM32F446xx)
  /* Connect PAx pins to FMC Alternate function */
  GPIOA->AFR[0]  |= 0xC0000000;
  GPIOA->AFR[1]  |= 0x00000000;
  /* Configure PDx pins in Alternate function mode */
  GPIOA->MODER   |= 0x00008000;
  /* Configure PDx pins speed to 50 MHz */
```

```
  GPIOA->OSPEEDR |= 0x00008000;
  /* Configure PDx pins Output type to push-pull */
  GPIOA->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOA->PUPDR   |= 0x00000000;
  /* Connect PCx pins to FMC Alternate function */
  GPIOC->AFR[0]  |= 0x00CC0000;
  GPIOC->AFR[1]  |= 0x00000000;
  /* Configure PDx pins in Alternate function mode */
  GPIOC->MODER   |= 0x00000A00;
  /* Configure PDx pins speed to 50 MHz */
  GPIOC->OSPEEDR |= 0x00000A00;
  /* Configure PDx pins Output type to push-pull */
  GPIOC->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOC->PUPDR   |= 0x00000000;
#endif /* STM32F446xx */
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x000000CC;
  GPIOD->AFR[1]  = 0xCC000CCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xA02A000A;
  /* Configure PDx pins speed to 50 MHz */
```

```
GPIOD->OSPEEDR = 0xA02A000A;
/* Configure PDx pins Output type to push-pull */
GPIOD->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PDx pins */
GPIOD->PUPDR   = 0x00000000;
/* Connect PEx pins to FMC Alternate function */
GPIOE->AFR[0]  = 0xC00000CC;
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA800A;
/* Configure PEx pins speed to 50 MHz */
GPIOE->OSPEEDR = 0xAAAA800A;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0xCCCCCCCC;
GPIOF->AFR[1]  = 0xCCCCCCCC;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA800AAA;
/* Configure PFx pins speed to 50 MHz */
GPIOF->OSPEEDR = 0xAA800AAA;
```

```
  /* Configure PFx pins Output type to push-pull */
  GPIOF->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PFx pins */
  GPIOF->PUPDR   = 0x00000000;
  /* Connect PGx pins to FMC Alternate function */
  GPIOG->AFR[0]  = 0xCCCCCCCC;
  GPIOG->AFR[1]  = 0xCCCCCCCC;
  /* Configure PGx pins in Alternate function mode */
  GPIOG->MODER   = 0xAAAAAAAA;
  /* Configure PGx pins speed to 50 MHz */
  GPIOG->OSPEEDR = 0xAAAAAAAA;
  /* Configure PGx pins Output type to push-pull */
  GPIOG->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
  /* Connect PHx pins to FMC Alternate function */
  GPIOH->AFR[0]  = 0x00C0CC00;
  GPIOH->AFR[1]  = 0xCCCCCCCC;
  /* Configure PHx pins in Alternate function mode */
  GPIOH->MODER   = 0xAAAA08A0;
```

```
  /* Configure PHx pins speed to 50 MHz */
  GPIOH->OSPEEDR = 0xAAAA08A0;
  /* Configure PHx pins Output type to push-pull */
  GPIOH->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PHx pins */
  GPIOH->PUPDR   = 0x00000000;
  /* Connect PIx pins to FMC Alternate function */
  GPIOI->AFR[0]  = 0xCCCCCCCC;
  GPIOI->AFR[1]  = 0x00000CC0;
  /* Configure PIx pins in Alternate function mode */
  GPIOI->MODER   = 0x0028AAAA;
  /* Configure PIx pins speed to 50 MHz */
  GPIOI->OSPEEDR = 0x0028AAAA;
  /* Configure PIx pins Output type to push-pull */
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F469xx || STM32F479xx */
/*-- FMC Configuration ------------------------------------------------------*/
  /* Enable the FMC interface clock */
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
```

```
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable SDRAM bank1 */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCR[0] = 0x00001954;
#else
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
#endif /* STM32F446xx */
  FMC_Bank5_6->SDTR[0] = 0x01115351;
  /* SDRAM initialization sequence */
  /* Clock enable command */
  FMC_Bank5_6->SDCMR = 0x00000011;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Delay */
  for (index = 0; index<1000; index++);
  /* PALL command */
  FMC_Bank5_6->SDCMR = 0x00000012;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
```

```
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Auto refresh command */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x000000F3;
#else
  FMC_Bank5_6->SDCMR = 0x00000073;
#endif /* STM32F446xx */
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* MRD register program */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x00044014;
#else
  FMC_Bank5_6->SDCMR = 0x00046014;
#endif /* STM32F446xx */
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
```

```
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Set refresh count */
  tmpreg = FMC_Bank5_6->SDRTR;
#if defined(STM32F446xx)
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000050C<<1));
#else
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
#endif /* STM32F446xx */
  /* Disable write protection */
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#endif /* DATA_IN_ExtSDRAM */
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F446xx || STM32F469xx || STM32F479xx */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx) || defined(STM32F417xx)\
 || defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx) ||
defined(STM32F412Zx) || defined(STM32F412Vx)
#if defined(DATA_IN_ExtSRAM)
```

```
/*-- GPIOs Configuration ----------------------------------------------------*/
  /* Enable GPIOD, GPIOE, GPIOF and GPIOG interface clock */
  RCC->AHB1ENR   |= 0x00000078;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIODEN);
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x00CCC0CC;
  GPIOD->AFR[1]  = 0xCCCCCCCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xAAAA0A8A;
  /* Configure PDx pins speed to 100 MHz */
  GPIOD->OSPEEDR = 0xFFFF0FCF;
  /* Configure PDx pins Output type to push-pull */
  GPIOD->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOD->PUPDR   = 0x00000000;
  /* Connect PEx pins to FMC Alternate function */
  GPIOE->AFR[0]  = 0xC00CC0CC;
  GPIOE->AFR[1]  = 0xCCCCCCCC;
  /* Configure PEx pins in Alternate function mode */
  GPIOE->MODER   = 0xAAAA828A;
  /* Configure PEx pins speed to 100 MHz */
  GPIOE->OSPEEDR = 0xFFFFC3CF;
```

```
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0x00CCCCCC;
GPIOF->AFR[1]  = 0xCCCC0000;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA000AAA;
/* Configure PFx pins speed to 100 MHz */
GPIOF->OSPEEDR = 0xFF000FFF;
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
GPIOF->PUPDR   = 0x00000000;
/* Connect PGx pins to FMC Alternate function */
GPIOG->AFR[0]  = 0x00CCCCCC;
GPIOG->AFR[1]  = 0x000000C0;
/* Configure PGx pins in Alternate function mode */
GPIOG->MODER   = 0x00085AAA;
/* Configure PGx pins speed to 100 MHz */
GPIOG->OSPEEDR = 0x000CAFFF;
/* Configure PGx pins Output type to push-pull */
```

```
  GPIOG->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;
/*-- FMC/FSMC Configuration ----------------------------------------------
-*/
  /* Enable the FMC/FSMC interface clock */
  RCC->AHB3ENR         |= 0x00000001;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
```

```
#endif /* STM32F469xx || STM32F479xx */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx)|| defined(STM32F417xx)\
  || defined(STM32F412Zx) || defined(STM32F412Vx)
 /* Delay after an RCC peripheral clock enabling */
 tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FSMCEN);
 /* Configure and enable Bank1_SRAM2 */
 FSMC_Bank1->BTCR[2]  = 0x00001011;
 FSMC_Bank1->BTCR[3]  = 0x00000201;
 FSMC_Bank1E->BWTR[2] = 0x0FFFFFFF;
#endif /* STM32F405xx || STM32F415xx || STM32F407xx ||
STM32F417xx || STM32F412Zx || STM32F412Vx */
#endif /* DATA_IN_ExtSRAM */
#endif /* STM32F405xx || STM32F415xx || STM32F407xx ||
STM32F417xx || STM32F427xx || STM32F437xx ||\
       STM32F429xx || STM32F439xx || STM32F469xx ||
STM32F479xx || STM32F412Zx || STM32F412Vx  */
 (void)(tmp);
}
#endif /* DATA_IN_ExtSRAM && DATA_IN_ExtSDRAM */
/**
 * @}
 */
/**
```

```
  * @}
  */
/**
  * @}
  */
```

CHAPTER#19 CODING GPIO DRIVERS FROM SCRATCH USING DATASHEET INFORMATION (PART 2)

19.1CODING GPIO DRIVERS FROM SCRATCH USING DATASHEET INFORMATION (PART 2):

Continuing from the foundational concepts discussed earlier, coding GPIO drivers from scratch involves deeper interactions with the hardware, ensuring robust and efficient control of GPIO functionalities.

19.2 ATOMIC OPERATIONS AND BIT MANIPLUATION:

Atomic operations are crucial for setting or clearing individual bits in registers without affecting other bits. This is especially important in multi-threaded environments or interrupt contexts where non-atomic operations might lead to race conditions.

19.2.1 BIT SET/RESET REGISTER (BSRR):

This register allows for atomic set and reset operations on GPIO pins.

19.3 HANDLING MULTIPLE PINS SIMULTANEOUSLY:

When dealing with multiple GPIO pins, especially in applications where timing and efficiency are critical, you may need to configure or manipulate multiple pins at once.

19.4 INTERRUPT HANDLING FOR GPIO:

Handling GPIO interrupts involves configuring the microcontroller to respond to changes on GPIO pins. This usually requires:

19.4.1 CONFIGURING INTERRUPT TRIGGER TYPE:

Rising edge, falling edge, or both. Enabling the Interrupt in the NVIC (Nested Vector Interrupt Controller). Writing an Interrupt Service Routine (ISR).

19.5 ADVANCED CONFIGURATION AND ERROR HANDLING:

When developing GPIO drivers, consider edge cases and error handling:

19.5.1 DEBOUNCING:

For mechanical switches, implement debouncing to prevent multiple triggers from a single press.

19.5.2 POWER MANAGEMENT:

Properly manage power modes to save energy, especially in battery-powered applications.

19.5.3 FAULT TOLERANCE:

Implement checks and error handling for potential faults, such as short circuits or unexpected voltage levels.

19.6 CODE OF MAIN.H:

```
/* USER CODE BEGIN Header */
/**
*******************************************************
********************
  * @file           : main.h
  * @brief          : Header for main.c file.
  *                   This file contains the common defines of the application.

*******************************************************
********************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
```

```
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -----------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H
#ifdef __cplusplus
extern "C" {
#endif
/* Includes ------------------------------------------------------------------*/
#include "stm32f4xx_hal.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */
/* USER CODE END EC */
/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */
```

```
/* USER CODE END EM */
/* Exported functions prototypes ---------------------------------------------*/
void Error_Handler(void);
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
/* Private defines -----------------------------------------------------------*/
#define B1_Pin GPIO_PIN_13
#define B1_GPIO_Port GPIOC
#define USART_TX_Pin GPIO_PIN_2
#define USART_TX_GPIO_Port GPIOA
#define USART_RX_Pin GPIO_PIN_3
#define USART_RX_GPIO_Port GPIOA
#define LD2_Pin GPIO_PIN_5
#define LD2_GPIO_Port GPIOA
#define TMS_Pin GPIO_PIN_13
#define TMS_GPIO_Port GPIOA
#define TCK_Pin GPIO_PIN_14
#define TCK_GPIO_Port GPIOA
#define SWO_Pin GPIO_PIN_3
#define SWO_GPIO_Port GPIOB
/* USER CODE BEGIN Private defines */
/* USER CODE END Private defines */
#ifdef __cplusplus
```

```
}
#endif
#endif /* __MAIN_H */
```

19.7 CODE OF STM32F4XX_HAL_CONF.H:

```
/* USER CODE BEGIN Header */
/**

**********************************************************
*******************
  * @file    stm32f4xx_hal_conf_template.h
  * @author  MCD Application Team
  * @brief   HAL configuration template file.
  *          This file should be copied to the application folder and renamed
  *          to stm32f4xx_hal_conf.h.
**********************************************************
*******************
  * @attention
  *
  * Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
```

```
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *

**********************************************************
*******************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_HAL_CONF_H
#define __STM32F4xx_HAL_CONF_H
#ifdef __cplusplus
 extern "C" {
#endif
/* Exported types ------------------------------------------------------------*/
/* Exported constants --------------------------------------------------------*/
/* ############################ Module Selection
################################## */
/**
  * @brief This is the list of modules to be used in the HAL driver
  */
#define HAL_MODULE_ENABLED
  /* #define HAL_CRYP_MODULE_ENABLED */
/* #define HAL_ADC_MODULE_ENABLED */
/* #define HAL_CAN_MODULE_ENABLED */
```

```
/* #define HAL_CRC_MODULE_ENABLED */
/* #define HAL_CAN_LEGACY_MODULE_ENABLED */
/* #define HAL_DAC_MODULE_ENABLED */
/* #define HAL_DCMI_MODULE_ENABLED */
/* #define HAL_DMA2D_MODULE_ENABLED */
/* #define HAL_ETH_MODULE_ENABLED */
/* #define HAL_ETH_LEGACY_MODULE_ENABLED */
/* #define HAL_NAND_MODULE_ENABLED */
/* #define HAL_NOR_MODULE_ENABLED */
/* #define HAL_PCCARD_MODULE_ENABLED */
/* #define HAL_SRAM_MODULE_ENABLED */
/* #define HAL_SDRAM_MODULE_ENABLED */
/* #define HAL_HASH_MODULE_ENABLED */
/* #define HAL_I2C_MODULE_ENABLED */
/* #define HAL_I2S_MODULE_ENABLED */
/* #define HAL_IWDG_MODULE_ENABLED */
/* #define HAL_LTDC_MODULE_ENABLED */
/* #define HAL_RNG_MODULE_ENABLED */
/* #define HAL_RTC_MODULE_ENABLED */
/* #define HAL_SAI_MODULE_ENABLED */
/* #define HAL_SD_MODULE_ENABLED */
/* #define HAL_MMC_MODULE_ENABLED */
/* #define HAL_SPI_MODULE_ENABLED */
```

```
/* #define HAL_TIM_MODULE_ENABLED */
#define HAL_UART_MODULE_ENABLED
/* #define HAL_USART_MODULE_ENABLED */
/* #define HAL_IRDA_MODULE_ENABLED */
/* #define HAL_SMARTCARD_MODULE_ENABLED */
/* #define HAL_SMBUS_MODULE_ENABLED */
/* #define HAL_WWDG_MODULE_ENABLED */
/* #define HAL_PCD_MODULE_ENABLED */
/* #define HAL_HCD_MODULE_ENABLED */
/* #define HAL_DSI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_CEC_MODULE_ENABLED */
/* #define HAL_FMPI2C_MODULE_ENABLED */
/* #define HAL_FMPSMBUS_MODULE_ENABLED */
/* #define HAL_SPDIFRX_MODULE_ENABLED */
/* #define HAL_DFSDM_MODULE_ENABLED */
/* #define HAL_LPTIM_MODULE_ENABLED */
#define HAL_GPIO_MODULE_ENABLED
#define HAL_EXTI_MODULE_ENABLED
#define HAL_DMA_MODULE_ENABLED
#define HAL_RCC_MODULE_ENABLED
#define HAL_FLASH_MODULE_ENABLED
```

```
#define HAL_PWR_MODULE_ENABLED
#define HAL_CORTEX_MODULE_ENABLED
/* ########################### HSE/HSI Values adaptation ###################### */
/**
  * @brief Adjust the value of External High Speed oscillator (HSE) used in your application.
  *        This value is used by the RCC HAL module to compute the system frequency
  *        (when HSE is used as system clock source, directly or through the PLL).
  */
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    8000000U /*!< Value of the External oscillator in Hz */
#endif /* HSE_VALUE */
#if !defined  (HSE_STARTUP_TIMEOUT)
  #define HSE_STARTUP_TIMEOUT    100U   /*!< Time out for HSE start up, in ms */
#endif /* HSE_STARTUP_TIMEOUT */
/**
  * @brief Internal High Speed oscillator (HSI) value.
  *        This value is used by the RCC HAL module to compute the system frequency
```

```
  *       (when HSI is used as system clock source, directly or through the
PLL).
  */
#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000U) /*!< Value of the
Internal oscillator in Hz*/
#endif /* HSI_VALUE */
/**
  * @brief Internal Low Speed oscillator (LSI) value.
  */
#if !defined  (LSI_VALUE)
 #define LSI_VALUE  32000U       /*!< LSI Typical Value in Hz*/
#endif /* LSI_VALUE */                    /*!< Value of the Internal Low
Speed oscillator in Hz
                                  The real value may vary depending on the
variations
                                  in voltage and temperature.*/
/**
  * @brief External Low Speed oscillator (LSE) value.
  */
#if !defined  (LSE_VALUE)
 #define LSE_VALUE  32768U    /*!< Value of the External Low Speed
oscillator in Hz */
#endif /* LSE_VALUE */
#if !defined  (LSE_STARTUP_TIMEOUT)
```

```
  #define LSE_STARTUP_TIMEOUT    5000U   /*!< Time out for
LSE start up, in ms */

#endif /* LSE_STARTUP_TIMEOUT */

/**

  * @brief External clock source for I2S peripheral

  *        This value is used by the I2S HAL module to compute the I2S
clock source

  *        frequency, this source is inserted directly through I2S_CKIN
pad.

  */

#if !defined  (EXTERNAL_CLOCK_VALUE)

  #define EXTERNAL_CLOCK_VALUE    12288000U /*!< Value of
the External audio frequency in Hz*/

#endif /* EXTERNAL_CLOCK_VALUE */

/* Tip: To avoid modifying this file each time you need to use different
HSE,

   ===  you can define the HSE value in your toolchain compiler
preprocessor. */

/* ############################# System Configuration
########################### */

/**

  * @brief This is the HAL system configuration section

  */

#define  VDD_VALUE                    3300U /*!< Value of VDD in mv */
```

```
#define  TICK_INT_PRIORITY          0U   /*!< tick interrupt
priority */

#define  USE_RTOS                   0U

#define  PREFETCH_ENABLE            1U

#define  INSTRUCTION_CACHE_ENABLE     1U

#define  DATA_CACHE_ENABLE          1U

#define  USE_HAL_ADC_REGISTER_CALLBACKS        0U /*
ADC register callback disabled       */

#define  USE_HAL_CAN_REGISTER_CALLBACKS        0U /*
CAN register callback disabled       */

#define  USE_HAL_CEC_REGISTER_CALLBACKS        0U /*
CEC register callback disabled       */

#define  USE_HAL_CRYP_REGISTER_CALLBACKS       0U /*
CRYP register callback disabled      */

#define  USE_HAL_DAC_REGISTER_CALLBACKS        0U /*
DAC register callback disabled       */

#define  USE_HAL_DCMI_REGISTER_CALLBACKS       0U /*
DCMI register callback disabled      */

#define  USE_HAL_DFSDM_REGISTER_CALLBACKS      0U /*
DFSDM register callback disabled     */

#define  USE_HAL_DMA2D_REGISTER_CALLBACKS      0U /*
DMA2D register callback disabled     */

#define  USE_HAL_DSI_REGISTER_CALLBACKS        0U /* DSI
register callback disabled       */

#define  USE_HAL_ETH_REGISTER_CALLBACKS        0U /*
ETH register callback disabled       */
```

#define USE_HAL_HASH_REGISTER_CALLBACKS 0U /* HASH register callback disabled */

#define USE_HAL_HCD_REGISTER_CALLBACKS 0U /* HCD register callback disabled */

#define USE_HAL_I2C_REGISTER_CALLBACKS 0U /* I2C register callback disabled */

#define USE_HAL_FMPI2C_REGISTER_CALLBACKS 0U /* FMPI2C register callback disabled */

#define USE_HAL_FMPSMBUS_REGISTER_CALLBACKS 0U /* FMPSMBUS register callback disabled */

#define USE_HAL_I2S_REGISTER_CALLBACKS 0U /* I2S register callback disabled */

#define USE_HAL_IRDA_REGISTER_CALLBACKS 0U /* IRDA register callback disabled */

#define USE_HAL_LPTIM_REGISTER_CALLBACKS 0U /* LPTIM register callback disabled */

#define USE_HAL_LTDC_REGISTER_CALLBACKS 0U /* LTDC register callback disabled */

#define USE_HAL_MMC_REGISTER_CALLBACKS 0U /* MMC register callback disabled */

#define USE_HAL_NAND_REGISTER_CALLBACKS 0U /* NAND register callback disabled */

#define USE_HAL_NOR_REGISTER_CALLBACKS 0U /* NOR register callback disabled */

#define USE_HAL_PCCARD_REGISTER_CALLBACKS 0U /* PCCARD register callback disabled */

#define USE_HAL_PCD_REGISTER_CALLBACKS 0U /* PCD register callback disabled */

#define USE_HAL_QSPI_REGISTER_CALLBACKS 0U /* QSPI register callback disabled */

#define USE_HAL_RNG_REGISTER_CALLBACKS 0U /* RNG register callback disabled */

#define USE_HAL_RTC_REGISTER_CALLBACKS 0U /* RTC register callback disabled */

#define USE_HAL_SAI_REGISTER_CALLBACKS 0U /* SAI register callback disabled */

#define USE_HAL_SD_REGISTER_CALLBACKS 0U /* SD register callback disabled */

#define USE_HAL_SMARTCARD_REGISTER_CALLBACKS 0U /* SMARTCARD register callback disabled */

#define USE_HAL_SDRAM_REGISTER_CALLBACKS 0U /* SDRAM register callback disabled */

#define USE_HAL_SRAM_REGISTER_CALLBACKS 0U /* SRAM register callback disabled */

#define USE_HAL_SPDIFRX_REGISTER_CALLBACKS 0U /* SPDIFRX register callback disabled */

#define USE_HAL_SMBUS_REGISTER_CALLBACKS 0U /* SMBUS register callback disabled */

#define USE_HAL_SPI_REGISTER_CALLBACKS 0U /* SPI register callback disabled */

#define USE_HAL_TIM_REGISTER_CALLBACKS 0U /* TIM register callback disabled */

```
#define  USE_HAL_UART_REGISTER_CALLBACKS       0U /*
UART register callback disabled      */

#define  USE_HAL_USART_REGISTER_CALLBACKS      0U /*
USART register callback disabled     */

#define  USE_HAL_WWDG_REGISTER_CALLBACKS       0U /*
WWDG register callback disabled      */

/* ############################ Assert Selection
################################## */

/**

  * @brief Uncomment the line below to expanse the "assert_param"
macro in the

  *        HAL drivers code

  */

/* #define USE_FULL_ASSERT    1U */

/* #################### Ethernet peripheral configuration
###################### */

/* Section 1 : Ethernet peripheral configuration */

/* MAC ADDRESS:
MAC_ADDR0:MAC_ADDR1:MAC_ADDR2:MAC_ADDR3:MAC_A
DDR4:MAC_ADDR5 */

#define MAC_ADDR0   2U

#define MAC_ADDR1   0U

#define MAC_ADDR2   0U

#define MAC_ADDR3   0U

#define MAC_ADDR4   0U
```

```
#define MAC_ADDR5   0U

/* Definition of the Ethernet driver buffers size and count */

#define ETH_RX_BUF_SIZE              ETH_MAX_PACKET_SIZE
/* buffer size for receive            */

#define ETH_TX_BUF_SIZE              ETH_MAX_PACKET_SIZE
/* buffer size for transmit           */

#define ETH_RXBUFNB                  4U      /* 4 Rx buffers of size
ETH_RX_BUF_SIZE  */

#define ETH_TXBUFNB                  4U      /* 4 Tx buffers of size
ETH_TX_BUF_SIZE  */

/* Section 2: PHY configuration section */

/* DP83848_PHY_ADDRESS Address*/

#define DP83848_PHY_ADDRESS

/* PHY Reset delay these values are based on a 1 ms Systick interrupt*/

#define PHY_RESET_DELAY              0x000000FFU

/* PHY Configuration delay */

#define PHY_CONFIG_DELAY             0x00000FFFU

#define PHY_READ_TO                  0x0000FFFFU

#define PHY_WRITE_TO                 0x0000FFFFU

/* Section 3: Common PHY Registers */

#define PHY_BCR                      ((uint16_t)0x0000U)    /*!<
Transceiver Basic Control Register   */

#define PHY_BSR                      ((uint16_t)0x0001U)    /*!<
Transceiver Basic Status Register    */
```

```
#define PHY_RESET                    ((uint16_t)0x8000U)  /*!< PHY
Reset */

#define PHY_LOOPBACK                 ((uint16_t)0x4000U)  /*!<
Select loop-back mode */

#define PHY_FULLDUPLEX_100M          ((uint16_t)0x2100U)
/*!< Set the full-duplex mode at 100 Mb/s */

#define PHY_HALFDUPLEX_100M          ((uint16_t)0x2000U)
/*!< Set the half-duplex mode at 100 Mb/s */

#define PHY_FULLDUPLEX_10M           ((uint16_t)0x0100U)
/*!< Set the full-duplex mode at 10 Mb/s  */

#define PHY_HALFDUPLEX_10M           ((uint16_t)0x0000U)
/*!< Set the half-duplex mode at 10 Mb/s  */

#define PHY_AUTONEGOTIATION          ((uint16_t)0x1000U)
/*!< Enable auto-negotiation function     */

#define PHY_RESTART_AUTONEGOTIATION
((uint16_t)0x0200U)  /*!< Restart auto-negotiation function    */

#define PHY_POWERDOWN                ((uint16_t)0x0800U)  /*!<
Select the power down mode           */

#define PHY_ISOLATE                  ((uint16_t)0x0400U)  /*!< Isolate
PHY from MII                 */

#define PHY_AUTONEGO_COMPLETE        ((uint16_t)0x0020U)
/*!< Auto-Negotiation process completed   */

#define PHY_LINKED_STATUS            ((uint16_t)0x0004U)  /*!<
Valid link established               */

#define PHY_JABBER_DETECTION         ((uint16_t)0x0002U)
/*!< Jabber condition detected            */

/* Section 4: Extended PHY Registers */
```

```
#define PHY_SR                    ((uint16_t))    /*!< PHY status register
Offset                  */

#define PHY_SPEED_STATUS              ((uint16_t))  /*!< PHY Speed
mask                          */

#define PHY_DUPLEX_STATUS             ((uint16_t))  /*!< PHY
Duplex mask                           */

/* ################### SPI peripheral configuration
############################ */

/* CRC FEATURE: Use to activate CRC feature inside HAL SPI Driver

* Activated: CRC code is present inside driver

* Deactivated: CRC code cleaned from driver

*/

#define USE_SPI_CRC                   0U

/* Includes ------------------------------------------------------------------*/

/**

  * @brief Include module's header file

  */

#ifdef HAL_RCC_MODULE_ENABLED

  #include "stm32f4xx_hal_rcc.h"

#endif /* HAL_RCC_MODULE_ENABLED */

#ifdef HAL_GPIO_MODULE_ENABLED

  #include "stm32f4xx_hal_gpio.h"

#endif /* HAL_GPIO_MODULE_ENABLED */

#ifdef HAL_EXTI_MODULE_ENABLED
```

```
  #include "stm32f4xx_hal_exti.h"
#endif /* HAL_EXTI_MODULE_ENABLED */
#ifdef HAL_DMA_MODULE_ENABLED
  #include "stm32f4xx_hal_dma.h"
#endif /* HAL_DMA_MODULE_ENABLED */
#ifdef HAL_CORTEX_MODULE_ENABLED
  #include "stm32f4xx_hal_cortex.h"
#endif /* HAL_CORTEX_MODULE_ENABLED */
#ifdef HAL_ADC_MODULE_ENABLED
  #include "stm32f4xx_hal_adc.h"
#endif /* HAL_ADC_MODULE_ENABLED */
#ifdef HAL_CAN_MODULE_ENABLED
  #include "stm32f4xx_hal_can.h"
#endif /* HAL_CAN_MODULE_ENABLED */
#ifdef HAL_CAN_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_can_legacy.h"
#endif /* HAL_CAN_LEGACY_MODULE_ENABLED */
#ifdef HAL_CRC_MODULE_ENABLED
  #include "stm32f4xx_hal_crc.h"
#endif /* HAL_CRC_MODULE_ENABLED */
#ifdef HAL_CRYP_MODULE_ENABLED
  #include "stm32f4xx_hal_cryp.h"
#endif /* HAL_CRYP_MODULE_ENABLED */
```

```
#ifdef HAL_DMA2D_MODULE_ENABLED
  #include "stm32f4xx_hal_dma2d.h"
#endif /* HAL_DMA2D_MODULE_ENABLED */
#ifdef HAL_DAC_MODULE_ENABLED
  #include "stm32f4xx_hal_dac.h"
#endif /* HAL_DAC_MODULE_ENABLED */
#ifdef HAL_DCMI_MODULE_ENABLED
  #include "stm32f4xx_hal_dcmi.h"
#endif /* HAL_DCMI_MODULE_ENABLED */
#ifdef HAL_ETH_MODULE_ENABLED
  #include "stm32f4xx_hal_eth.h"
#endif /* HAL_ETH_MODULE_ENABLED */
#ifdef HAL_ETH_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_eth_legacy.h"
#endif /* HAL_ETH_LEGACY_MODULE_ENABLED */
#ifdef HAL_FLASH_MODULE_ENABLED
  #include "stm32f4xx_hal_flash.h"
#endif /* HAL_FLASH_MODULE_ENABLED */
#ifdef HAL_SRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sram.h"
#endif /* HAL_SRAM_MODULE_ENABLED */
#ifdef HAL_NOR_MODULE_ENABLED
  #include "stm32f4xx_hal_nor.h"
```

```
#endif /* HAL_NOR_MODULE_ENABLED */
#ifdef HAL_NAND_MODULE_ENABLED
  #include "stm32f4xx_hal_nand.h"
#endif /* HAL_NAND_MODULE_ENABLED */
#ifdef HAL_PCCARD_MODULE_ENABLED
  #include "stm32f4xx_hal_pccard.h"
#endif /* HAL_PCCARD_MODULE_ENABLED */
#ifdef HAL_SDRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sdram.h"
#endif /* HAL_SDRAM_MODULE_ENABLED */
#ifdef HAL_HASH_MODULE_ENABLED
 #include "stm32f4xx_hal_hash.h"
#endif /* HAL_HASH_MODULE_ENABLED */
#ifdef HAL_I2C_MODULE_ENABLED
 #include "stm32f4xx_hal_i2c.h"
#endif /* HAL_I2C_MODULE_ENABLED */
#ifdef HAL_SMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_smbus.h"
#endif /* HAL_SMBUS_MODULE_ENABLED */
#ifdef HAL_I2S_MODULE_ENABLED
 #include "stm32f4xx_hal_i2s.h"
#endif /* HAL_I2S_MODULE_ENABLED */
#ifdef HAL_IWDG_MODULE_ENABLED
```

```
 #include "stm32f4xx_hal_iwdg.h"
#endif /* HAL_IWDG_MODULE_ENABLED */
#ifdef HAL_LTDC_MODULE_ENABLED
 #include "stm32f4xx_hal_ltdc.h"
#endif /* HAL_LTDC_MODULE_ENABLED */
#ifdef HAL_PWR_MODULE_ENABLED
 #include "stm32f4xx_hal_pwr.h"
#endif /* HAL_PWR_MODULE_ENABLED */
#ifdef HAL_RNG_MODULE_ENABLED
 #include "stm32f4xx_hal_rng.h"
#endif /* HAL_RNG_MODULE_ENABLED */
#ifdef HAL_RTC_MODULE_ENABLED
 #include "stm32f4xx_hal_rtc.h"
#endif /* HAL_RTC_MODULE_ENABLED */
#ifdef HAL_SAI_MODULE_ENABLED
 #include "stm32f4xx_hal_sai.h"
#endif /* HAL_SAI_MODULE_ENABLED */
#ifdef HAL_SD_MODULE_ENABLED
 #include "stm32f4xx_hal_sd.h"
#endif /* HAL_SD_MODULE_ENABLED */
#ifdef HAL_SPI_MODULE_ENABLED
 #include "stm32f4xx_hal_spi.h"
#endif /* HAL_SPI_MODULE_ENABLED */
```

```
#ifdef HAL_TIM_MODULE_ENABLED
 #include "stm32f4xx_hal_tim.h"
#endif /* HAL_TIM_MODULE_ENABLED */
#ifdef HAL_UART_MODULE_ENABLED
 #include "stm32f4xx_hal_uart.h"
#endif /* HAL_UART_MODULE_ENABLED */
#ifdef HAL_USART_MODULE_ENABLED
 #include "stm32f4xx_hal_usart.h"
#endif /* HAL_USART_MODULE_ENABLED */
#ifdef HAL_IRDA_MODULE_ENABLED
 #include "stm32f4xx_hal_irda.h"
#endif /* HAL_IRDA_MODULE_ENABLED */
#ifdef HAL_SMARTCARD_MODULE_ENABLED
 #include "stm32f4xx_hal_smartcard.h"
#endif /* HAL_SMARTCARD_MODULE_ENABLED */
#ifdef HAL_WWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_wwdg.h"
#endif /* HAL_WWDG_MODULE_ENABLED */
#ifdef HAL_PCD_MODULE_ENABLED
 #include "stm32f4xx_hal_pcd.h"
#endif /* HAL_PCD_MODULE_ENABLED */
#ifdef HAL_HCD_MODULE_ENABLED
 #include "stm32f4xx_hal_hcd.h"
```

```
#endif /* HAL_HCD_MODULE_ENABLED */
#ifdef HAL_DSI_MODULE_ENABLED
 #include "stm32f4xx_hal_dsi.h"
#endif /* HAL_DSI_MODULE_ENABLED */
#ifdef HAL_QSPI_MODULE_ENABLED
 #include "stm32f4xx_hal_qspi.h"
#endif /* HAL_QSPI_MODULE_ENABLED */
#ifdef HAL_CEC_MODULE_ENABLED
 #include "stm32f4xx_hal_cec.h"
#endif /* HAL_CEC_MODULE_ENABLED */
#ifdef HAL_FMPI2C_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpi2c.h"
#endif /* HAL_FMPI2C_MODULE_ENABLED */
#ifdef HAL_FMPSMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpsmbus.h"
#endif /* HAL_FMPSMBUS_MODULE_ENABLED */
#ifdef HAL_SPDIFRX_MODULE_ENABLED
 #include "stm32f4xx_hal_spdifrx.h"
#endif /* HAL_SPDIFRX_MODULE_ENABLED */
#ifdef HAL_DFSDM_MODULE_ENABLED
 #include "stm32f4xx_hal_dfsdm.h"
#endif /* HAL_DFSDM_MODULE_ENABLED */
#ifdef HAL_LPTIM_MODULE_ENABLED
```

```
 #include "stm32f4xx_hal_lptim.h"
#endif /* HAL_LPTIM_MODULE_ENABLED */
#ifdef HAL_MMC_MODULE_ENABLED
 #include "stm32f4xx_hal_mmc.h"
#endif /* HAL_MMC_MODULE_ENABLED */
/* Exported macro ------------------------------------------------------------*/
#ifdef  USE_FULL_ASSERT
/**
  * @brief  The assert_param macro is used for function's parameters check.
  * @param  expr If expr is false, it calls assert_failed function
  *         which reports the name of the source file and the source
  *         line number of the call that failed.
  *         If expr is true, it returns no value.
  * @retval None
  */
  #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t *)__FILE__, __LINE__))
/* Exported functions ------------------------------------------------------- */
  void assert_failed(uint8_t* file, uint32_t line);
#else
  #define assert_param(expr) ((void)0U)
#endif /* USE_FULL_ASSERT */
#ifdef __cplusplus
```

```
}
#endif
#endif /* __STM32F4xx_HAL_CONF_H */
```

19.8 CODE OF STM32F4XX_IT.H:

```
/* USER CODE BEGIN Header */
/**
**********************************************************
********************
  * @file    stm32f4xx_it.h
  * @brief   This file contains the headers of the interrupt handlers.
**********************************************************
********************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
********************
```

```
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_IT_H
#define __STM32F4xx_IT_H
#ifdef __cplusplus
 extern "C" {
#endif
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */
/* USER CODE END EC */
/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */
/* USER CODE END EM */
/* Exported functions prototypes ---------------------------------------------*/
void NMI_Handler(void);
void HardFault_Handler(void);
```

```
void MemManage_Handler(void);
void BusFault_Handler(void);
void UsageFault_Handler(void);
void SVC_Handler(void);
void DebugMon_Handler(void);
void PendSV_Handler(void);
void SysTick_Handler(void);
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
#ifdef __cplusplus
}
#endif
#endif /* __STM32F4xx_IT_H */
```

19.9 CODE OF MAIN.C:

```
/* USER CODE BEGIN Header */
/**
*****************************************************************
*******************
  * @file           : main.c
  * @brief          : Main program body

*****************************************************************
*******************
  * @attention
```

```
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *

**********************************************************
********************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
/* USER CODE END PTD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
```

```
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
UART_HandleTypeDef huart2;
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
```

```
  /* USER CODE BEGIN 1 */
  /* USER CODE END 1 */
  /* MCU Configuration--------------------------------------------------------*/
  /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
  HAL_Init();
  /* USER CODE BEGIN Init */
  /* USER CODE END Init */
  /* Configure the system clock */
  SystemClock_Config();
  /* USER CODE BEGIN SysInit */
  /* USER CODE END SysInit */
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  /* USER CODE BEGIN 2 */
  /* USER CODE END 2 */
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
```

```
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();

__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_
VOLTAGE_SCALE1);
  /** Initializes the RCC Oscillators according to the specified
parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
```

```
  RCC_OscInitStruct.HSICalibrationValue =
RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 16;
  RCC_OscInitStruct.PLL.PLLN = 336;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV4;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK

|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource =
RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct,
FLASH_LATENCY_2) != HAL_OK)
```

```
  {
    Error_Handler();
  }
}
/**
  * @brief USART2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART2_UART_Init(void)
{
  /* USER CODE BEGIN USART2_Init 0 */
  /* USER CODE END USART2_Init 0 */
  /* USER CODE BEGIN USART2_Init 1 */
  /* USER CODE END USART2_Init 1 */
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 115200;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
```

```
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART2_Init 2 */
  /* USER CODE END USART2_Init 2 */
}
/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */
  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  /*Configure GPIO pin Output Level */
```

```
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin,
GPIO_PIN_RESET);
  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);
  /*Configure GPIO pin : LD2_Pin */
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);
/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}
/* USER CODE BEGIN 4 */
/* USER CODE END 4 */
/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
```

```
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return
state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}
#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line
number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line
number,
```

```
    ex: printf("Wrong parameters value: file %s on line %d\r\n", file,
line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

19.10 CODE OF STM32F4XX_HAL_MSP.C:

```
/* USER CODE BEGIN Header */
/**
*******************************************************
*******************
  * @file         stm32f4xx_hal_msp.c
  * @brief        This file provides code for the MSP Initialization
  *               and de-Initialization codes.
*******************************************************
*******************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
```

```
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN TD */
/* USER CODE END TD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN Define */
/* USER CODE END Define */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN Macro */
/* USER CODE END Macro */
/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
```

```
/* External functions ------------------------------------------------------*/
/* USER CODE BEGIN ExternalFunctions */
/* USER CODE END ExternalFunctions */
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/**
  * Initializes the Global MSP.
  */
void HAL_MspInit(void)
{
  /* USER CODE BEGIN MspInit 0 */
  /* USER CODE END MspInit 0 */
  __HAL_RCC_SYSCFG_CLK_ENABLE();
  __HAL_RCC_PWR_CLK_ENABLE();
  HAL_NVIC_SetPriorityGrouping(NVIC_PRIORITYGROUP_0);
  /* System interrupt init*/
  /* USER CODE BEGIN MspInit 1 */
  /* USER CODE END MspInit 1 */
}
/**
* @brief UART MSP Initialization
* This function configures the hardware resources used in this example
* @param huart: UART handle pointer
```

```
* @retval None
*/
void HAL_UART_MspInit(UART_HandleTypeDef* huart)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  if(huart->Instance==USART2)
  {
  /* USER CODE BEGIN USART2_MspInit 0 */
  /* USER CODE END USART2_MspInit 0 */
    /* Peripheral clock enable */
    __HAL_RCC_USART2_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();
    /**USART2 GPIO Configuration
    PA2     ------> USART2_TX
    PA3     ------> USART2_RX
    */
    GPIO_InitStruct.Pin = USART_TX_Pin|USART_RX_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF7_USART2;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
  /* USER CODE BEGIN USART2_MspInit 1 */
```

```
  /* USER CODE END USART2_MspInit 1 */
  }
}
/**
* @brief UART MSP De-Initialization
* This function freeze the hardware resources used in this example
* @param huart: UART handle pointer
* @retval None
*/
void HAL_UART_MspDeInit(UART_HandleTypeDef* huart)
{
  if(huart->Instance==USART2)
  {
  /* USER CODE BEGIN USART2_MspDeInit 0 */
  /* USER CODE END USART2_MspDeInit 0 */
    /* Peripheral clock disable */
    __HAL_RCC_USART2_CLK_DISABLE();
    /**USART2 GPIO Configuration
    PA2     ------> USART2_TX
    PA3     ------> USART2_RX
    */
    HAL_GPIO_DeInit(GPIOA, USART_TX_Pin|USART_RX_Pin);
  /* USER CODE BEGIN USART2_MspDeInit 1 */
```

```
  /* USER CODE END USART2_MspDeInit 1 */
  }
}
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

19.11 CODE OF STM32F4XX_IT.C:

```
/* USER CODE BEGIN Header */
/**
*******************************************************************
  * @file    stm32f4xx_it.c
  * @brief   Interrupt Service Routines.
*******************************************************************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
```

```
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stm32f4xx_it.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN TD */
/* USER CODE END TD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
```

```
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/* External variables --------------------------------------------------------*/
/* USER CODE BEGIN EV */
/* USER CODE END EV */
/******************************************************************************/
/*            Cortex-M4 Processor Interruption and Exception Handlers          */
/******************************************************************************/
/**
  * @brief This function handles Non maskable interrupt.
  */
void NMI_Handler(void)
{
  /* USER CODE BEGIN NonMaskableInt_IRQn 0 */
  /* USER CODE END NonMaskableInt_IRQn 0 */
  /* USER CODE BEGIN NonMaskableInt_IRQn 1 */
   while (1)
   {
```

```
  }
  /* USER CODE END NonMaskableInt_IRQn 1 */
}
/**
  * @brief This function handles Hard fault interrupt.
  */
void HardFault_Handler(void)
{
  /* USER CODE BEGIN HardFault_IRQn 0 */
  /* USER CODE END HardFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_HardFault_IRQn 0 */
    /* USER CODE END W1_HardFault_IRQn 0 */
  }
}
/**
  * @brief This function handles Memory management fault.
  */
void MemManage_Handler(void)
{
  /* USER CODE BEGIN MemoryManagement_IRQn 0 */
  /* USER CODE END MemoryManagement_IRQn 0 */
```

```
  while (1)
  {
    /* USER CODE BEGIN W1_MemoryManagement_IRQn 0 */
    /* USER CODE END W1_MemoryManagement_IRQn 0 */
  }
}
/**
  * @brief This function handles Pre-fetch fault, memory access fault.
  */
void BusFault_Handler(void)
{
  /* USER CODE BEGIN BusFault_IRQn 0 */
  /* USER CODE END BusFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_BusFault_IRQn 0 */
    /* USER CODE END W1_BusFault_IRQn 0 */
  }
}
/**
  * @brief This function handles Undefined instruction or illegal state.
  */
void UsageFault_Handler(void)
```

```
{
  /* USER CODE BEGIN UsageFault_IRQn 0 */
  /* USER CODE END UsageFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_UsageFault_IRQn 0 */
    /* USER CODE END W1_UsageFault_IRQn 0 */
  }
}
/**
  * @brief This function handles System service call via SWI
instruction.
  */
void SVC_Handler(void)
{
  /* USER CODE BEGIN SVCall_IRQn 0 */
  /* USER CODE END SVCall_IRQn 0 */
  /* USER CODE BEGIN SVCall_IRQn 1 */
  /* USER CODE END SVCall_IRQn 1 */
}
/**
  * @brief This function handles Debug monitor.
  */
```

```
void DebugMon_Handler(void)
{
  /* USER CODE BEGIN DebugMonitor_IRQn 0 */
  /* USER CODE END DebugMonitor_IRQn 0 */
  /* USER CODE BEGIN DebugMonitor_IRQn 1 */
  /* USER CODE END DebugMonitor_IRQn 1 */
}
/**
  * @brief This function handles Pendable request for system service.
  */
void PendSV_Handler(void)
{
  /* USER CODE BEGIN PendSV_IRQn 0 */
  /* USER CODE END PendSV_IRQn 0 */
  /* USER CODE BEGIN PendSV_IRQn 1 */
  /* USER CODE END PendSV_IRQn 1 */
}
/**
  * @brief This function handles System tick timer.
  */
void SysTick_Handler(void)
{
  /* USER CODE BEGIN SysTick_IRQn 0 */
```

```
  /* USER CODE END SysTick_IRQn 0 */
  HAL_IncTick();
  /* USER CODE BEGIN SysTick_IRQn 1 */
  /* USER CODE END SysTick_IRQn 1 */
}
/*****************************************************************************/
/* STM32F4xx Peripheral Interrupt Handlers                                   */
/* Add here the Interrupt Handlers for the used peripherals.                 */
/* For the available peripheral interrupt handler names,                     */
/* please refer to the startup file (startup_stm32f4xx.s).                  */
/*****************************************************************************/
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

19.12 CODE OF SYSCALLS.C:

```
/**
*****************************************************************************
 * @file      syscalls.c
 * @author    Auto-generated by STM32CubeIDE
 * @brief     STM32CubeIDE Minimal System calls file
 *
 *            For more information about which c-functions
```

```
 *          need which of these lowlevel functions
 *          please consult the Newlib libc-manual
**********************************************************
*******************
 * @attention
 *
 * Copyright (c) 2020-2024 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the
LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
**********************************************************
*******************
 */
/* Includes */
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
```

```
#include <sys/times.h>
/* Variables */
extern int __io_putchar(int ch) __attribute__((weak));
extern int __io_getchar(void) __attribute__((weak));
char *__env[1] = { 0 };
char **environ = __env;
/* Functions */
void initialise_monitor_handles()
{
}
int _getpid(void)
{
  return 1;
}
int _kill(int pid, int sig)
{
  (void)pid;
  (void)sig;
  errno = EINVAL;
  return -1;
}
void _exit (int status)
{
```

```
  _kill(status, -1);
  while (1) {}    /* Make sure we hang here */
}
__attribute__((weak)) int _read(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;
  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    *ptr++ = __io_getchar();
  }
  return len;
}
__attribute__((weak)) int _write(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;
  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    __io_putchar(*ptr++);
  }
  return len;
}
```

```
int _close(int file)
{
  (void)file;
  return -1;
}
int _fstat(int file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
int _isatty(int file)
{
  (void)file;
  return 1;
}
int _lseek(int file, int ptr, int dir)
{
  (void)file;
  (void)ptr;
  (void)dir;
  return 0;
}
```

```
int _open(char *path, int flags, ...)
{
  (void)path;
  (void)flags;
  /* Pretend like we always fail */
  return -1;
}
int _wait(int *status)
{
  (void)status;
  errno = ECHILD;
  return -1;
}
int _unlink(char *name)
{
  (void)name;
  errno = ENOENT;
  return -1;
}
int _times(struct tms *buf)
{
  (void)buf;
  return -1;
```

```
}
int _stat(char *file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
int _link(char *old, char *new)
{
  (void)old;
  (void)new;
  errno = EMLINK;
  return -1;
}
int _fork(void)
{
  errno = EAGAIN;
  return -1;
}
int _execve(char *name, char **argv, char **env)
{
  (void)name;
  (void)argv;
```

```
  (void)env;
  errno = ENOMEM;
  return -1;
}
```

19.13 CODE OF SYSMEM.C:

```
/**
*****************************************************************
*******************
 * @file      sysmem.c
 * @author    Generated by STM32CubeIDE
 * @brief     STM32CubeIDE System Memory calls file
 *
 *            For more information about which C functions
 *            need which of these lowlevel functions
 *            please consult the newlib libc manual
*****************************************************************
*******************
 * @attention
 *
 * Copyright (c) 2024 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the
LICENSE file
```

```
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
**********************************************************
*******************
 */
/* Includes */
#include <errno.h>
#include <stdint.h>
/**
 * Pointer to the current high watermark of the heap usage
 */
static uint8_t *__sbrk_heap_end = NULL;
/**
 * @brief _sbrk() allocates memory to the newlib heap and is used by
malloc
 *        and others from the C library
 *
 * @verbatim
 *
#################################################################
##################
 * #  .data  #  .bss  #       newlib heap       #          MSP stack          #
 * #         #        #                          # Reserved by _Min_Stack_Size #
```

```
 *
##################################################################
##################
 * ^-- RAM start      ^-- _end                          _estack, RAM end --^
 * @endverbatim
 *
 * This implementation starts allocating at the '_end' linker symbol
 * The '_Min_Stack_Size' linker symbol reserves a memory for the MSP
stack
 * The implementation considers '_estack' linker symbol to be RAM end
 * NOTE: If the MSP stack, at any point during execution, grows larger
than the
 * reserved size, please increase the '_Min_Stack_Size'.
 *
 * @param incr Memory size
 * @return Pointer to allocated memory
 */
void *_sbrk(ptrdiff_t incr)
{
  extern uint8_t _end; /* Symbol defined in the linker script */
  extern uint8_t _estack; /* Symbol defined in the linker script */
  extern uint32_t _Min_Stack_Size; /* Symbol defined in the linker
script */
  const uint32_t stack_limit = (uint32_t)&_estack -
(uint32_t)&_Min_Stack_Size;
```

```
  const uint8_t *max_heap = (uint8_t *)stack_limit;
  uint8_t *prev_heap_end;
  /* Initialize heap end at first call */
  if (NULL == __sbrk_heap_end)
  {
    __sbrk_heap_end = &_end;
  }
  /* Protect heap from growing into the reserved MSP stack */
  if (__sbrk_heap_end + incr > max_heap)
  {
    errno = ENOMEM;
    return (void *)-1;
  }
  prev_heap_end = __sbrk_heap_end;
  __sbrk_heap_end += incr;
  return (void *)prev_heap_end;
}
```

19.14 CODE OF SYSTEM_STM32F4XX.C:

```
/**
*****************************************************************
*******************
  * @file    system_stm32f4xx.c
  * @author  MCD Application Team
```

```
  * @brief   CMSIS Cortex-M4 Device Peripheral Access Layer System
Source File.
  *
  *   This file provides two functions and one global variable to be called
from
  *   user application:
  *      - SystemInit(): This function is called at startup just after reset and
  *                      before branch to main program. This call is made inside
  *                      the "startup_stm32f4xx.s" file.
  *
  *      - SystemCoreClock variable: Contains the core clock (HCLK), it
can be used
  *                              by the user application to setup the SysTick
  *                              timer or configure other parameters.
  *
  *      - SystemCoreClockUpdate(): Updates the variable
SystemCoreClock and must
  *                             be called whenever the core clock is changed
  *                             during program execution.
  *
  *

*****************************************************************
*******************
  * @attention
```

```
  *
  * Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
*******************
  */
/** @addtogroup CMSIS
  * @{
  */
/** @addtogroup stm32f4xx_system
  * @{
  */
/** @addtogroup STM32F4xx_System_Private_Includes
  * @{
  */
#include "stm32f4xx.h"
#if !defined  (HSE_VALUE)
```

```
  #define HSE_VALUE    ((uint32_t)25000000) /*!< Default value of
the External oscillator in Hz */

#endif /* HSE_VALUE */

#if !defined  (HSI_VALUE)

  #define HSI_VALUE    ((uint32_t)16000000) /*!< Value of the
Internal oscillator in Hz*/

#endif /* HSI_VALUE */

/**

  * @}

  */

/** @addtogroup STM32F4xx_System_Private_TypesDefinitions

  * @{

  */

/**

  * @}

  */

/** @addtogroup STM32F4xx_System_Private_Defines

  * @{

  */

/************************* Miscellaneous Configuration
************************/

/*!< Uncomment the following line if you need to use external SRAM or
SDRAM as data memory  */

#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx) || defined(STM32F417xx)\
```

```
 || defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx) ||
defined(STM32F412Zx) || defined(STM32F412Vx)
/* #define DATA_IN_ExtSRAM */
#endif /* STM32F40xxx || STM32F41xxx || STM32F42xxx ||
STM32F43xxx || STM32F469xx || STM32F479xx ||\
        STM32F412Zx || STM32F412Vx */
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F446xx) || defined(STM32F469xx) ||
defined(STM32F479xx)
/* #define DATA_IN_ExtSDRAM */
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F446xx || STM32F469xx ||\
        STM32F479xx */
/* Note: Following vector table addresses must be defined in line with
linker
        configuration. */
/*!< Uncomment the following line if you need to relocate the vector
table
    anywhere in Flash or Sram, else the vector table is kept at the
automatic
    remap of boot address selected */
/* #define USER_VECT_TAB_ADDRESS */
#if defined(USER_VECT_TAB_ADDRESS)
```

```
/*!< Uncomment the following line if you need to relocate your vector Table
    in Sram else user remap will be done in Flash. */
/* #define VECT_TAB_SRAM */
#if defined(VECT_TAB_SRAM)
#define VECT_TAB_BASE_ADDRESS   SRAM_BASE       /*!< Vector Table base address field.
                                                This value must be a multiple of 0x200. */
#define VECT_TAB_OFFSET         0x00000000U     /*!< Vector Table base offset field.
                                                This value must be a multiple of 0x200. */
#else
#define VECT_TAB_BASE_ADDRESS   FLASH_BASE      /*!< Vector Table base address field.
                                                This value must be a multiple of 0x200. */
#define VECT_TAB_OFFSET         0x00000000U     /*!< Vector Table base offset field.
                                                This value must be a multiple of 0x200. */
#endif /* VECT_TAB_SRAM */
#endif /* USER_VECT_TAB_ADDRESS */
/*****************************************************************************/
```

```
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Macros
  * @{
  */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Variables
  * @{
  */
  /* This variable is updated in three ways:
     1) by calling CMSIS function SystemCoreClockUpdate()
     2) by calling HAL API function HAL_RCC_GetHCLKFreq()
     3) each time HAL_RCC_ClockConfig() is called to configure the system clock frequency
        Note: If you use this function to configure the system clock; then there
              is no need to call the 2 first functions listed above, since SystemCoreClock
              variable is updated automatically.
  */
uint32_t SystemCoreClock = 16000000;
```

```
const uint8_t AHBPrescTable[16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7,
8, 9};
const uint8_t APBPrescTable[8]  = {0, 0, 0, 0, 1, 2, 3, 4};
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_FunctionPrototypes
  * @{
  */
#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  static void SystemInit_ExtMemCtl(void);
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Functions
  * @{
  */
/**
  * @brief  Setup the microcontroller system
  *         Initialize the FPU setting, vector table location and External
memory
  *         configuration.
  * @param  None
```

```
  * @retval None
  */
void SystemInit(void)
{
  /* FPU settings ------------------------------------------------------------*/
  #if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
    SCB->CPACR |= ((3UL << 10*2)|(3UL << 11*2));  /* set CP10 and
CP11 Full Access */
  #endif
#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  SystemInit_ExtMemCtl();
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */
  /* Configure the Vector Table location -------------------------------------*/
#if defined(USER_VECT_TAB_ADDRESS)
  SCB->VTOR = VECT_TAB_BASE_ADDRESS |
VECT_TAB_OFFSET; /* Vector Table Relocation in Internal SRAM */
#endif /* USER_VECT_TAB_ADDRESS */
}
/**
   * @brief  Update SystemCoreClock variable according to Clock
Register Values.
  *         The SystemCoreClock variable contains the core clock (HCLK),
it can
  *         be used by the user application to setup the SysTick timer or
configure
```

```
  *        other parameters.
  *
  * @note   Each time the core clock (HCLK) changes, this function
must be called
  *        to update SystemCoreClock variable value. Otherwise, any
configuration
  *        based on this variable will be incorrect.
  *
  * @note   - The system frequency computed by this function is not the
real
  *          frequency in the chip. It is calculated based on the predefined
  *          constant and the selected clock source:
  *
  *          - If SYSCLK source is HSI, SystemCoreClock will contain the
HSI_VALUE(*)
  *
  *          - If SYSCLK source is HSE, SystemCoreClock will contain the
HSE_VALUE(**)
  *
  *          - If SYSCLK source is PLL, SystemCoreClock will contain the
HSE_VALUE(**)
  *            or HSI_VALUE(*) multiplied/divided by the PLL factors.
  *
  *        (*) HSI_VALUE is a constant defined in stm32f4xx_hal_conf.h
file (default value
```

```
  *            16 MHz) but the real value may vary depending on the variations
  *            in voltage and temperature.
  *
  *        (**) HSE_VALUE is a constant defined in stm32f4xx_hal_conf.h file (its value
  *             depends on the application requirements), user has to ensure that HSE_VALUE
  *             is same as the real frequency of the crystal used. Otherwise, this function
  *             may have wrong result.
  *
  *        - The result of this function could be not correct when using fractional
  *          value for HSE crystal.
  *
  * @param  None
  * @retval None
  */
void SystemCoreClockUpdate(void)
{
  uint32_t tmp = 0, pllvco = 0, pllp = 2, pllsource = 0, pllm = 2;
  /* Get SYSCLK source -------------------------------------------------------*/
  tmp = RCC->CFGR & RCC_CFGR_SWS;
```

```
  switch (tmp)
  {
    case 0x00:  /* HSI used as system clock source */
      SystemCoreClock = HSI_VALUE;
      break;
    case 0x04:  /* HSE used as system clock source */
      SystemCoreClock = HSE_VALUE;
      break;
    case 0x08:  /* PLL used as system clock source */
      /* PLL_VCO = (HSE_VALUE or HSI_VALUE / PLL_M) * PLL_N
         SYSCLK = PLL_VCO / PLL_P
         */
      pllsource = (RCC->PLLCFGR & RCC_PLLCFGR_PLLSRC) >>
22;
      pllm = RCC->PLLCFGR & RCC_PLLCFGR_PLLM;
      if (pllsource != 0)
      {
        /* HSE used as PLL clock source */
        pllvco = (HSE_VALUE / pllm) * ((RCC->PLLCFGR &
RCC_PLLCFGR_PLLN) >> 6);
      }
      else
      {
        /* HSI used as PLL clock source */
```

```
      pllvco = (HSI_VALUE / pllm) * ((RCC->PLLCFGR &
RCC_PLLCFGR_PLLN) >> 6);
    }
    pllp = (((RCC->PLLCFGR & RCC_PLLCFGR_PLLP) >>16) + 1 )
*2;
    SystemCoreClock = pllvco/pllp;
    break;
   default:
    SystemCoreClock = HSI_VALUE;
    break;
  }
  /* Compute HCLK frequency ----------------------------------------------
-*/
  /* Get HCLK prescaler */
  tmp = AHBPrescTable[((RCC->CFGR & RCC_CFGR_HPRE) >> 4)];
  /* HCLK frequency */
  SystemCoreClock >>= tmp;
}
#if defined (DATA_IN_ExtSRAM) && defined
(DATA_IN_ExtSDRAM)
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
/**
  * @brief  Setup the external memory controller.
```

```
  *        Called in startup_stm32f4xx.s before jump to main.
  *        This function configures the external memories
(SRAM/SDRAM)
  *        This SRAM/SDRAM will be used as program data memory
(including heap and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
  register __IO uint32_t index;
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and
GPIOI interface clock */
  RCC->AHB1ENR |= 0x000001F8;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x00CCC0CC;
  GPIOD->AFR[1]  = 0xCCCCCCCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xAAAA0A8A;
  /* Configure PDx pins speed to 100 MHz */
```

```
GPIOD->OSPEEDR = 0xFFFF0FCF;
/* Configure PDx pins Output type to push-pull */
GPIOD->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PDx pins */
GPIOD->PUPDR   = 0x00000000;
/* Connect PEx pins to FMC Alternate function */
GPIOE->AFR[0]  = 0xC00CC0CC;
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA828A;
/* Configure PEx pins speed to 100 MHz */
GPIOE->OSPEEDR = 0xFFFFC3CF;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0xCCCCCCCC;
GPIOF->AFR[1]  = 0xCCCCCCCC;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA800AAA;
/* Configure PFx pins speed to 50 MHz */
GPIOF->OSPEEDR = 0xAA800AAA;
```

```
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
GPIOF->PUPDR   = 0x00000000;
/* Connect PGx pins to FMC Alternate function */
GPIOG->AFR[0]  = 0xCCCCCCCC;
GPIOG->AFR[1]  = 0xCCCCCCCC;
/* Configure PGx pins in Alternate function mode */
GPIOG->MODER   = 0xAAAAAAAA;
/* Configure PGx pins speed to 50 MHz */
GPIOG->OSPEEDR = 0xAAAAAAAA;
/* Configure PGx pins Output type to push-pull */
GPIOG->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PGx pins */
GPIOG->PUPDR   = 0x00000000;
/* Connect PHx pins to FMC Alternate function */
GPIOH->AFR[0]  = 0x00C0CC00;
GPIOH->AFR[1]  = 0xCCCCCCCC;
/* Configure PHx pins in Alternate function mode */
GPIOH->MODER   = 0xAAAA08A0;
/* Configure PHx pins speed to 50 MHz */
GPIOH->OSPEEDR = 0xAAAA08A0;
/* Configure PHx pins Output type to push-pull */
```

```
  GPIOH->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PHx pins */
  GPIOH->PUPDR   = 0x00000000;
  /* Connect PIx pins to FMC Alternate function */
  GPIOI->AFR[0]  = 0xCCCCCCCC;
  GPIOI->AFR[1]  = 0x00000CC0;
  /* Configure PIx pins in Alternate function mode */
  GPIOI->MODER   = 0x0028AAAA;
  /* Configure PIx pins speed to 50 MHz */
  GPIOI->OSPEEDR = 0x0028AAAA;
  /* Configure PIx pins Output type to push-pull */
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;
/*-- FMC Configuration ------------------------------------------------------*/
  /* Enable the FMC interface clock */
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
  FMC_Bank5_6->SDTR[0] = 0x01115351;
  /* SDRAM initialization sequence */
  /* Clock enable command */
```

```
FMC_Bank5_6->SDCMR = 0x00000011;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* Delay */
for (index = 0; index<1000; index++);
/* PALL command */
FMC_Bank5_6->SDCMR = 0x00000012;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* Auto refresh command */
FMC_Bank5_6->SDCMR = 0x00000073;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
```

```
  }
  /* MRD register program */
  FMC_Bank5_6->SDCMR = 0x00046014;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Set refresh count */
  tmpreg = FMC_Bank5_6->SDRTR;
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
  /* Disable write protection */
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
```

```
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F469xx || STM32F479xx */
  (void)(tmp);
}
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F469xx || STM32F479xx */
#elif defined (DATA_IN_ExtSRAM) || defined
(DATA_IN_ExtSDRAM)
/**
  * @brief  Setup the external memory controller.
  *         Called in startup_stm32f4xx.s before jump to main.
  *         This function configures the external memories
(SRAM/SDRAM)
  *         This SRAM/SDRAM will be used as program data memory
(including heap and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;
```

```
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F446xx) || defined(STM32F469xx) ||
defined(STM32F479xx)
#if defined (DATA_IN_ExtSDRAM)
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
  register __IO uint32_t index;
#if defined(STM32F446xx)
  /* Enable GPIOA, GPIOC, GPIOD, GPIOE, GPIOF, GPIOG interface
     clock */
  RCC->AHB1ENR |= 0x0000007D;
#else
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and
GPIOI interface
     clock */
  RCC->AHB1ENR |= 0x000001F8;
#endif /* STM32F446xx */
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);
#if defined(STM32F446xx)
  /* Connect PAx pins to FMC Alternate function */
  GPIOA->AFR[0]  |= 0xC0000000;
  GPIOA->AFR[1]  |= 0x00000000;
  /* Configure PDx pins in Alternate function mode */
```

```
  GPIOA->MODER   |= 0x00008000;
  /* Configure PDx pins speed to 50 MHz */
  GPIOA->OSPEEDR |= 0x00008000;
  /* Configure PDx pins Output type to push-pull */
  GPIOA->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOA->PUPDR   |= 0x00000000;
  /* Connect PCx pins to FMC Alternate function */
  GPIOC->AFR[0]  |= 0x00CC0000;
  GPIOC->AFR[1]  |= 0x00000000;
  /* Configure PDx pins in Alternate function mode */
  GPIOC->MODER   |= 0x00000A00;
  /* Configure PDx pins speed to 50 MHz */
  GPIOC->OSPEEDR |= 0x00000A00;
  /* Configure PDx pins Output type to push-pull */
  GPIOC->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOC->PUPDR   |= 0x00000000;
#endif /* STM32F446xx */
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x000000CC;
  GPIOD->AFR[1]  = 0xCC000CCC;
  /* Configure PDx pins in Alternate function mode */
```

```
GPIOD->MODER   = 0xA02A000A;
/* Configure PDx pins speed to 50 MHz */
GPIOD->OSPEEDR = 0xA02A000A;
/* Configure PDx pins Output type to push-pull */
GPIOD->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PDx pins */
GPIOD->PUPDR   = 0x00000000;
/* Connect PEx pins to FMC Alternate function */
GPIOE->AFR[0]  = 0xC00000CC;
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA800A;
/* Configure PEx pins speed to 50 MHz */
GPIOE->OSPEEDR = 0xAAAA800A;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0xCCCCCCCC;
GPIOF->AFR[1]  = 0xCCCCCCCC;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA800AAA;
```

```
  /* Configure PFx pins speed to 50 MHz */
  GPIOF->OSPEEDR = 0xAA800AAA;
  /* Configure PFx pins Output type to push-pull */
  GPIOF->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PFx pins */
  GPIOF->PUPDR   = 0x00000000;
  /* Connect PGx pins to FMC Alternate function */
  GPIOG->AFR[0]  = 0xCCCCCCCC;
  GPIOG->AFR[1]  = 0xCCCCCCCC;
  /* Configure PGx pins in Alternate function mode */
  GPIOG->MODER   = 0xAAAAAAAA;
  /* Configure PGx pins speed to 50 MHz */
  GPIOG->OSPEEDR = 0xAAAAAAAA;
  /* Configure PGx pins Output type to push-pull */
  GPIOG->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
  /* Connect PHx pins to FMC Alternate function */
  GPIOH->AFR[0]  = 0x00C0CC00;
  GPIOH->AFR[1]  = 0xCCCCCCCC;
```

```
  /* Configure PHx pins in Alternate function mode */
  GPIOH->MODER   = 0xAAAA08A0;
  /* Configure PHx pins speed to 50 MHz */
  GPIOH->OSPEEDR = 0xAAAA08A0;
  /* Configure PHx pins Output type to push-pull */
  GPIOH->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PHx pins */
  GPIOH->PUPDR   = 0x00000000;
  /* Connect PIx pins to FMC Alternate function */
  GPIOI->AFR[0]  = 0xCCCCCCCC;
  GPIOI->AFR[1]  = 0x00000CC0;
  /* Configure PIx pins in Alternate function mode */
  GPIOI->MODER   = 0x0028AAAA;
  /* Configure PIx pins speed to 50 MHz */
  GPIOI->OSPEEDR = 0x0028AAAA;
  /* Configure PIx pins Output type to push-pull */
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F469xx || STM32F479xx */
/*-- FMC Configuration ------------------------------------------------------*/
  /* Enable the FMC interface clock */
```

```
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable SDRAM bank1 */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCR[0] = 0x00001954;
#else
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
#endif /* STM32F446xx */
  FMC_Bank5_6->SDTR[0] = 0x01115351;
  /* SDRAM initialization sequence */
  /* Clock enable command */
  FMC_Bank5_6->SDCMR = 0x00000011;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Delay */
  for (index = 0; index<1000; index++);
  /* PALL command */
  FMC_Bank5_6->SDCMR = 0x00000012;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
```

```
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Auto refresh command */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x000000F3;
#else
  FMC_Bank5_6->SDCMR = 0x00000073;
#endif /* STM32F446xx */
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* MRD register program */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x00044014;
#else
  FMC_Bank5_6->SDCMR = 0x00046014;
#endif /* STM32F446xx */
```

```
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Set refresh count */
  tmpreg = FMC_Bank5_6->SDRTR;
#if defined(STM32F446xx)
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000050C<<1));
#else
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
#endif /* STM32F446xx */
  /* Disable write protection */
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#endif /* DATA_IN_ExtSDRAM */
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F446xx || STM32F469xx || STM32F479xx */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx) || defined(STM32F417xx)\
 || defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
```

```
 || defined(STM32F469xx) || defined(STM32F479xx) ||
defined(STM32F412Zx) || defined(STM32F412Vx)
#if defined(DATA_IN_ExtSRAM)
/*-- GPIOs Configuration -----------------------------------------------------*/
  /* Enable GPIOD, GPIOE, GPIOF and GPIOG interface clock */
  RCC->AHB1ENR   |= 0x00000078;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIODEN);
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x00CCC0CC;
  GPIOD->AFR[1]  = 0xCCCCCCCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xAAAA0A8A;
  /* Configure PDx pins speed to 100 MHz */
  GPIOD->OSPEEDR = 0xFFFF0FCF;
  /* Configure PDx pins Output type to push-pull */
  GPIOD->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOD->PUPDR   = 0x00000000;
  /* Connect PEx pins to FMC Alternate function */
  GPIOE->AFR[0]  = 0xC00CC0CC;
  GPIOE->AFR[1]  = 0xCCCCCCCC;
  /* Configure PEx pins in Alternate function mode */
```

```
GPIOE->MODER   = 0xAAAA828A;
/* Configure PEx pins speed to 100 MHz */
GPIOE->OSPEEDR = 0xFFFFC3CF;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0x00CCCCCC;
GPIOF->AFR[1]  = 0xCCCC0000;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA000AAA;
/* Configure PFx pins speed to 100 MHz */
GPIOF->OSPEEDR = 0xFF000FFF;
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
GPIOF->PUPDR   = 0x00000000;
/* Connect PGx pins to FMC Alternate function */
GPIOG->AFR[0]  = 0x00CCCCCC;
GPIOG->AFR[1]  = 0x000000C0;
/* Configure PGx pins in Alternate function mode */
GPIOG->MODER   = 0x00085AAA;
```

```
  /* Configure PGx pins speed to 100 MHz */
  GPIOG->OSPEEDR = 0x000CAFFF;
  /* Configure PGx pins Output type to push-pull */
  GPIOG->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;
/*-- FMC/FSMC Configuration ----------------------------------------------
-*/
  /* Enable the FMC/FSMC interface clock */
  RCC->AHB3ENR         |= 0x00000001;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
```

```
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F469xx || STM32F479xx */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx)|| defined(STM32F417xx)\
   || defined(STM32F412Zx) || defined(STM32F412Vx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FSMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FSMC_Bank1->BTCR[2]  = 0x00001011;
  FSMC_Bank1->BTCR[3]  = 0x00000201;
  FSMC_Bank1E->BWTR[2] = 0x0FFFFFFF;
#endif /* STM32F405xx || STM32F415xx || STM32F407xx ||
STM32F417xx || STM32F412Zx || STM32F412Vx */
#endif /* DATA_IN_ExtSRAM */
#endif /* STM32F405xx || STM32F415xx || STM32F407xx ||
STM32F417xx || STM32F427xx || STM32F437xx ||\
        STM32F429xx || STM32F439xx || STM32F469xx ||
STM32F479xx || STM32F412Zx || STM32F412Vx  */
  (void)(tmp);
}
#endif /* DATA_IN_ExtSRAM && DATA_IN_ExtSDRAM */
/**
```

```
  * @}
  */
/**
  * @}
  */
/**
  * @}
  */
```

CHAPTER#20 CODING ARM ASSEMBLY APPLICATIONS IN CUBEIDE

20.1CODING ARM ASSEMBLY APPLICATIONS IN CUBEIDE:

Coding ARM assembly applications in STM32CubeIDE involves integrating assembly language code within a C/C++ project. This can be useful for performance-critical sections, low-level hardware interactions, or optimizing specific algorithms.

20.2 CREATING A NEW PROJECT:

20.2.1 START STM32CUBEIDE:

Open STM32CubeIDE and create a new STM32 project. Select your target STM32 microcontroller.

20.2.2 CONFIGURE THE PROJECT:

Use STM32CubeMX to configure the peripherals and generate initialization code.

20.3 WRITING ASSEMBLY CODE:

There are two main ways to include assembly code in your project:

20.3.1 INLINE ASSEMBLY IN C/C++ FILES:

Inline assembly is useful for including small sections of assembly code within C/C++ functions. GCC's inline assembly syntax is commonly used.

20.3.2 STANDALONE ASSEMBLY FILES:

Standalone assembly files allow you to write more extensive assembly code separately from C/C++ code. These files typically have a .s or .asm extension.

20.3.2.1 CREATING ASSEMBLY FILE:

In STM32CubeIDE, create a new file in your project with a '.s' or .asm extension.

20.3.2.2 WRITING ASSEMBLY CODE:

Write your ARM assembly code using ARM's instruction set. Ensure to follow proper syntax and conventions.

20.4 INTEGRATING ASSEMBLY WITH C/C++:

To use functions written in assembly from C/C++, you need to declare them as extern in your C/C++ files.

20.5 COMPLING AND LINKING:

When you build your project, STM32CubeIDE compiles the C/C++ and assembly files, then links them together into the final executable. The IDE uses GCC (GNU Compiler Collection), which supports ARM architecture.

20.6 DEBUGGING AND OPTIMIZATION:

20.6.1 DEBUGGING:

Use STM32CubeIDE's debugging tools to set breakpoints, inspect registers, and step through both C and assembly code.

20.6.2 OPTIMIZATION:

Inline assembly allows for precise optimization. Be cautious with optimizations, as they can lead to complex debugging scenarios.

20.7 ASSEMBLER DIRECTIVES:

Assembler directives are instructions that help control the assembly process. Some common ARM directives include:

20.7.1 '.GLOBAL':

Makes a symbol visible to the linker, allowing it to be referenced in other files.

20.7.2 '.TYPE':

Specifies the type of a symbol (e.g., function, object).

20.7.3 '.SECTION':

Defines sections for different types of data (e.g., .text for code, .data for initialized data).

20.7.4 '.ALIGN':

Aligns data to a specific byte boundary, which can be important for performance.

20.8 CODE OF ARMASSEMBLY.ASM:

```
/*
*****************************************************
*******************
**
** @file        : LinkerScript.ld (debug in RAM dedicated)
**
** @author      : Auto-generated by STM32CubeIDE
**
**  Abstract    : Linker script for NUCLEO-F411RE Board embedding
STM32F411RETx Device from stm32f4 series
**                  512KBytes FLASH
**                  128KBytes RAM
**
**              Set heap size, stack size and stack location according
**              to application requirements.
**
**              Set memory bank area and size if external memory is used
**
**  Target      : STMicroelectronics STM32
**
**  Distribution: The file is distributed as is, without any warranty
**              of any kind.
**
```

```
******************************************************
*******************
** @attention
**
** Copyright (c) 2024 STMicroelectronics.
** All rights reserved.
**
** This software is licensed under terms that can be found in the
LICENSE file
** in the root directory of this software component.
** If no LICENSE file comes with this software, it is provided AS-IS.
**
******************************************************
*******************
*/
/* Entry Point */
ENTRY(Reset_Handler)
/* Highest address of the user mode stack */
_estack = ORIGIN(RAM) + LENGTH(RAM); /* end of "RAM" Ram
type memory */
_Min_Heap_Size = 0x200; /* required amount of heap */
_Min_Stack_Size = 0x400; /* required amount of stack */
/* Memories definition */
MEMORY
```

```
{
  RAM    (xrw)    : ORIGIN = 0x20000000,   LENGTH = 128K
  FLASH    (rx)    : ORIGIN = 0x8000000,   LENGTH = 512K
}
/* Sections */
SECTIONS
{
  /* The startup code into "RAM" Ram type memory */
  .isr_vector :
  {
    . = ALIGN(4);
    KEEP(*(.isr_vector)) /* Startup code */
    . = ALIGN(4);
  } >RAM
  /* The program code and other data into "RAM" Ram type memory */
  .text :
  {
    . = ALIGN(4);
    *(.text)           /* .text sections (code) */
    *(.text*)          /* .text* sections (code) */
    *(.glue_7)         /* glue arm to thumb code */
    *(.glue_7t)        /* glue thumb to arm code */
    *(.eh_frame)
```

```
    *(.RamFunc)        /* .RamFunc sections */
    *(.RamFunc*)       /* .RamFunc* sections */
    KEEP (*(.init))
    KEEP (*(.fini))
    . = ALIGN(4);
    _etext = .;        /* define a global symbols at end of code */
  } >RAM
  /* Constant data into "RAM" Ram type memory */
  .rodata :
  {
    . = ALIGN(4);
    *(.rodata)         /* .rodata sections (constants, strings, etc.) */
    *(.rodata*)        /* .rodata* sections (constants, strings, etc.) */
    . = ALIGN(4);
  } >RAM
  .ARM.extab (READONLY) : /* The "READONLY" keyword is only
supported in GCC11 and later, remove it if using GCC10 or earlier. */
  {
    . = ALIGN(4);
    *(.ARM.extab* .gnu.linkonce.armextab.*)
    . = ALIGN(4);
  } >RAM
  .ARM (READONLY) : /* The "READONLY" keyword is only
supported in GCC11 and later, remove it if using GCC10 or earlier. */
```

```
  {
    . = ALIGN(4);
    __exidx_start = .;
    *(.ARM.exidx*)
    __exidx_end = .;
    . = ALIGN(4);
  } >RAM
  .preinit_array (READONLY) : /* The "READONLY" keyword is only supported in GCC11 and later, remove it if using GCC10 or earlier. */
  {
    . = ALIGN(4);
    PROVIDE_HIDDEN (__preinit_array_start = .);
    KEEP (*(.preinit_array*))
    PROVIDE_HIDDEN (__preinit_array_end = .);
    . = ALIGN(4);
  } >RAM
  .init_array (READONLY) : /* The "READONLY" keyword is only supported in GCC11 and later, remove it if using GCC10 or earlier. */
  {
    . = ALIGN(4);
    PROVIDE_HIDDEN (__init_array_start = .);
    KEEP (*(SORT(.init_array.*)))
    KEEP (*(.init_array*))
    PROVIDE_HIDDEN (__init_array_end = .);
```

```
    . = ALIGN(4);
  } >RAM
  .fini_array (READONLY) : /* The "READONLY" keyword is only
supported in GCC11 and later, remove it if using GCC10 or earlier. */
  {
    . = ALIGN(4);
    PROVIDE_HIDDEN (__fini_array_start = .);
    KEEP (*(SORT(.fini_array.*)))
    KEEP (*(.fini_array*))
    PROVIDE_HIDDEN (__fini_array_end = .);
    . = ALIGN(4);
  } >RAM
  /* Used by the startup to initialize data */
  _sidata = LOADADDR(.data);
  /* Initialized data sections into "RAM" Ram type memory */
  .data :
  {
    . = ALIGN(4);
    _sdata = .;        /* create a global symbol at data start */
    *(.data)           /* .data sections */
    *(.data*)          /* .data* sections */

    . = ALIGN(4);
```

```
  _edata = .;        /* define a global symbol at data end */
} >RAM
/* Uninitialized data section into "RAM" Ram type memory */
. = ALIGN(4);
.bss :
{
  /* This is used by the startup in order to initialize the .bss section */
  _sbss = .;         /* define a global symbol at bss start */
  __bss_start__ = _sbss;
  *(.bss)
  *(.bss*)
  *(COMMON)
  . = ALIGN(4);
  _ebss = .;         /* define a global symbol at bss end */
  __bss_end__ = _ebss;
} >RAM
/* User_heap_stack section, used to check that there is enough "RAM"
Ram  type memory left */
._user_heap_stack :
{
  . = ALIGN(8);
  PROVIDE ( end = . );
  PROVIDE ( _end = . );
```

```
    . = . + _Min_Heap_Size;
    . = . + _Min_Stack_Size;
    . = ALIGN(8);
  } >RAM
  /* Remove information from the compiler libraries */
  /DISCARD/ :
  {
    libc.a ( * )
    libm.a ( * )
    libgcc.a ( * )
  }
  .ARM.attributes 0 : { *(.ARM.attributes) }
}
```

20.9 CODE OF MAIN.H:

```
/* USER CODE BEGIN Header */
/**
*******************************************************************
  * @file           : main.h
  * @brief          : Header for main.c file.
  *                   This file contains the common defines of the application.
*******************************************************************
  * @attention
```

```
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H
#ifdef __cplusplus
extern "C" {
#endif
/* Includes ------------------------------------------------------------------*/
#include "stm32f4xx_hal.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
```

```
/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */
/* USER CODE END EC */
/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */
/* USER CODE END EM */
/* Exported functions prototypes ---------------------------------------------*/
void Error_Handler(void);
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
/* Private defines -----------------------------------------------------------*/
#define B1_Pin GPIO_PIN_13
#define B1_GPIO_Port GPIOC
#define USART_TX_Pin GPIO_PIN_2
#define USART_TX_GPIO_Port GPIOA
#define USART_RX_Pin GPIO_PIN_3
#define USART_RX_GPIO_Port GPIOA
#define LD2_Pin GPIO_PIN_5
#define LD2_GPIO_Port GPIOA
#define TMS_Pin GPIO_PIN_13
```

```
#define TMS_GPIO_Port GPIOA
#define TCK_Pin GPIO_PIN_14
#define TCK_GPIO_Port GPIOA
#define SWO_Pin GPIO_PIN_3
#define SWO_GPIO_Port GPIOB
/* USER CODE BEGIN Private defines */
/* USER CODE END Private defines */
#ifdef __cplusplus
}
#endif
#endif /* __MAIN_H */
```

20.10 CODE OF STM32F4XX_HAL_CONF.H:

```
/* USER CODE BEGIN Header */
/**
*****************************************************************
*******************
  * @file    stm32f4xx_hal_conf_template.h
  * @author  MCD Application Team
  * @brief   HAL configuration template file.
  *          This file should be copied to the application folder and renamed
  *          to stm32f4xx_hal_conf.h.
*****************************************************************
*******************
```

```
  * @attention
  *
  * Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
********************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_HAL_CONF_H
#define __STM32F4xx_HAL_CONF_H
#ifdef __cplusplus
 extern "C" {
#endif
/* Exported types ------------------------------------------------------------*/
/* Exported constants --------------------------------------------------------*/
/* ############################ Module Selection
############################### */
```

```
/**
  * @brief This is the list of modules to be used in the HAL driver
  */
#define HAL_MODULE_ENABLED
  /* #define HAL_CRYP_MODULE_ENABLED */
/* #define HAL_ADC_MODULE_ENABLED */
/* #define HAL_CAN_MODULE_ENABLED */
/* #define HAL_CRC_MODULE_ENABLED */
/* #define HAL_CAN_LEGACY_MODULE_ENABLED */
/* #define HAL_DAC_MODULE_ENABLED */
/* #define HAL_DCMI_MODULE_ENABLED */
/* #define HAL_DMA2D_MODULE_ENABLED */
/* #define HAL_ETH_MODULE_ENABLED */
/* #define HAL_ETH_LEGACY_MODULE_ENABLED */
/* #define HAL_NAND_MODULE_ENABLED */
/* #define HAL_NOR_MODULE_ENABLED */
/* #define HAL_PCCARD_MODULE_ENABLED */
/* #define HAL_SRAM_MODULE_ENABLED */
/* #define HAL_SDRAM_MODULE_ENABLED */
/* #define HAL_HASH_MODULE_ENABLED */
/* #define HAL_I2C_MODULE_ENABLED */
/* #define HAL_I2S_MODULE_ENABLED */
/* #define HAL_IWDG_MODULE_ENABLED */
```

```
/* #define HAL_LTDC_MODULE_ENABLED */
/* #define HAL_RNG_MODULE_ENABLED */
/* #define HAL_RTC_MODULE_ENABLED */
/* #define HAL_SAI_MODULE_ENABLED */
/* #define HAL_SD_MODULE_ENABLED */
/* #define HAL_MMC_MODULE_ENABLED */
/* #define HAL_SPI_MODULE_ENABLED */
/* #define HAL_TIM_MODULE_ENABLED */
#define HAL_UART_MODULE_ENABLED
/* #define HAL_USART_MODULE_ENABLED */
/* #define HAL_IRDA_MODULE_ENABLED */
/* #define HAL_SMARTCARD_MODULE_ENABLED */
/* #define HAL_SMBUS_MODULE_ENABLED */
/* #define HAL_WWDG_MODULE_ENABLED */
/* #define HAL_PCD_MODULE_ENABLED */
/* #define HAL_HCD_MODULE_ENABLED */
/* #define HAL_DSI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_CEC_MODULE_ENABLED */
/* #define HAL_FMPI2C_MODULE_ENABLED */
/* #define HAL_FMPSMBUS_MODULE_ENABLED */
/* #define HAL_SPDIFRX_MODULE_ENABLED */
```

```
/* #define HAL_DFSDM_MODULE_ENABLED */
/* #define HAL_LPTIM_MODULE_ENABLED */
#define HAL_GPIO_MODULE_ENABLED
#define HAL_EXTI_MODULE_ENABLED
#define HAL_DMA_MODULE_ENABLED
#define HAL_RCC_MODULE_ENABLED
#define HAL_FLASH_MODULE_ENABLED
#define HAL_PWR_MODULE_ENABLED
#define HAL_CORTEX_MODULE_ENABLED
/* ############################ HSE/HSI Values adaptation
###################### */
/**
  * @brief Adjust the value of External High Speed oscillator (HSE)
used in your application.
  *        This value is used by the RCC HAL module to compute the
system frequency
  *        (when HSE is used as system clock source, directly or through
the PLL).
  */
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    8000000U /*!< Value of the External
oscillator in Hz */
#endif /* HSE_VALUE */
#if !defined  (HSE_STARTUP_TIMEOUT)
```

```
  #define HSE_STARTUP_TIMEOUT    100U   /*!< Time out for
HSE start up, in ms */
#endif /* HSE_STARTUP_TIMEOUT */
/**
  * @brief Internal High Speed oscillator (HSI) value.
  *        This value is used by the RCC HAL module to compute the
system frequency
  *        (when HSI is used as system clock source, directly or through the
PLL).
  */
#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000U) /*!< Value of the
Internal oscillator in Hz*/
#endif /* HSI_VALUE */
/**
  * @brief Internal Low Speed oscillator (LSI) value.
  */
#if !defined  (LSI_VALUE)
 #define LSI_VALUE  32000U       /*!< LSI Typical Value in Hz*/
#endif /* LSI_VALUE */                    /*!< Value of the Internal Low
Speed oscillator in Hz
                                 The real value may vary depending on the
variations
                                 in voltage and temperature.*/
/**
```

```
  * @brief External Low Speed oscillator (LSE) value.
  */
#if !defined  (LSE_VALUE)
 #define LSE_VALUE  32768U    /*!< Value of the External Low Speed
oscillator in Hz */
#endif /* LSE_VALUE */
#if !defined  (LSE_STARTUP_TIMEOUT)
  #define LSE_STARTUP_TIMEOUT    5000U   /*!< Time out for
LSE start up, in ms */
#endif /* LSE_STARTUP_TIMEOUT */

/**
  * @brief External clock source for I2S peripheral
  *        This value is used by the I2S HAL module to compute the I2S
clock source
  *        frequency, this source is inserted directly through I2S_CKIN
pad.
  */
#if !defined  (EXTERNAL_CLOCK_VALUE)
  #define EXTERNAL_CLOCK_VALUE    12288000U /*!< Value of
the External audio frequency in Hz*/
#endif /* EXTERNAL_CLOCK_VALUE */
/* Tip: To avoid modifying this file each time you need to use different
HSE,
```

```
  === you can define the HSE value in your toolchain compiler
preprocessor. */

/* ############################## System Configuration
########################### */

/**

  * @brief This is the HAL system configuration section

  */

#define  VDD_VALUE                    3300U /*!< Value of VDD in mv */

#define  TICK_INT_PRIORITY            0U   /*!< tick interrupt
priority */

#define  USE_RTOS                     0U

#define  PREFETCH_ENABLE              1U

#define  INSTRUCTION_CACHE_ENABLE     1U

#define  DATA_CACHE_ENABLE            1U

#define  USE_HAL_ADC_REGISTER_CALLBACKS        0U /*
ADC register callback disabled       */

#define  USE_HAL_CAN_REGISTER_CALLBACKS        0U /*
CAN register callback disabled       */

#define  USE_HAL_CEC_REGISTER_CALLBACKS        0U /*
CEC register callback disabled       */

#define  USE_HAL_CRYP_REGISTER_CALLBACKS       0U /*
CRYP register callback disabled      */

#define  USE_HAL_DAC_REGISTER_CALLBACKS        0U /*
DAC register callback disabled       */

#define  USE_HAL_DCMI_REGISTER_CALLBACKS       0U /*
DCMI register callback disabled      */
```

#define USE_HAL_DFSDM_REGISTER_CALLBACKS 0U /* DFSDM register callback disabled */

#define USE_HAL_DMA2D_REGISTER_CALLBACKS 0U /* DMA2D register callback disabled */

#define USE_HAL_DSI_REGISTER_CALLBACKS 0U /* DSI register callback disabled */

#define USE_HAL_ETH_REGISTER_CALLBACKS 0U /* ETH register callback disabled */

#define USE_HAL_HASH_REGISTER_CALLBACKS 0U /* HASH register callback disabled */

#define USE_HAL_HCD_REGISTER_CALLBACKS 0U /* HCD register callback disabled */

#define USE_HAL_I2C_REGISTER_CALLBACKS 0U /* I2C register callback disabled */

#define USE_HAL_FMPI2C_REGISTER_CALLBACKS 0U /* FMPI2C register callback disabled */

#define USE_HAL_FMPSMBUS_REGISTER_CALLBACKS 0U /* FMPSMBUS register callback disabled */

#define USE_HAL_I2S_REGISTER_CALLBACKS 0U /* I2S register callback disabled */

#define USE_HAL_IRDA_REGISTER_CALLBACKS 0U /* IRDA register callback disabled */

#define USE_HAL_LPTIM_REGISTER_CALLBACKS 0U /* LPTIM register callback disabled */

#define USE_HAL_LTDC_REGISTER_CALLBACKS 0U /* LTDC register callback disabled */

#define USE_HAL_MMC_REGISTER_CALLBACKS 0U /* MMC register callback disabled */

#define USE_HAL_NAND_REGISTER_CALLBACKS 0U /* NAND register callback disabled */

#define USE_HAL_NOR_REGISTER_CALLBACKS 0U /* NOR register callback disabled */

#define USE_HAL_PCCARD_REGISTER_CALLBACKS 0U /* PCCARD register callback disabled */

#define USE_HAL_PCD_REGISTER_CALLBACKS 0U /* PCD register callback disabled */

#define USE_HAL_QSPI_REGISTER_CALLBACKS 0U /* QSPI register callback disabled */

#define USE_HAL_RNG_REGISTER_CALLBACKS 0U /* RNG register callback disabled */

#define USE_HAL_RTC_REGISTER_CALLBACKS 0U /* RTC register callback disabled */

#define USE_HAL_SAI_REGISTER_CALLBACKS 0U /* SAI register callback disabled */

#define USE_HAL_SD_REGISTER_CALLBACKS 0U /* SD register callback disabled */

#define USE_HAL_SMARTCARD_REGISTER_CALLBACKS 0U /* SMARTCARD register callback disabled */

#define USE_HAL_SDRAM_REGISTER_CALLBACKS 0U /* SDRAM register callback disabled */

#define USE_HAL_SRAM_REGISTER_CALLBACKS 0U /* SRAM register callback disabled */

```
#define  USE_HAL_SPDIFRX_REGISTER_CALLBACKS     0U /*
SPDIFRX register callback disabled   */

#define  USE_HAL_SMBUS_REGISTER_CALLBACKS      0U /*
SMBUS register callback disabled     */

#define  USE_HAL_SPI_REGISTER_CALLBACKS        0U /* SPI
register callback disabled       */

#define  USE_HAL_TIM_REGISTER_CALLBACKS        0U /*
TIM register callback disabled       */

#define  USE_HAL_UART_REGISTER_CALLBACKS       0U /*
UART register callback disabled      */

#define  USE_HAL_USART_REGISTER_CALLBACKS      0U /*
USART register callback disabled     */

#define  USE_HAL_WWDG_REGISTER_CALLBACKS       0U /*
WWDG register callback disabled      */

/* ############################ Assert Selection
################################## */

/**

  * @brief Uncomment the line below to expanse the "assert_param"
macro in the

  *        HAL drivers code

  */

/* #define USE_FULL_ASSERT    1U */

/* #################### Ethernet peripheral configuration
###################### */

/* Section 1 : Ethernet peripheral configuration */
```

/* MAC ADDRESS:
MAC_ADDR0:MAC_ADDR1:MAC_ADDR2:MAC_ADDR3:MAC_ADDR4:MAC_ADDR5 */

#define MAC_ADDR0 2U

#define MAC_ADDR1 0U

#define MAC_ADDR2 0U

#define MAC_ADDR3 0U

#define MAC_ADDR4 0U

#define MAC_ADDR5 0U

/* Definition of the Ethernet driver buffers size and count */

#define ETH_RX_BUF_SIZE ETH_MAX_PACKET_SIZE /* buffer size for receive */

#define ETH_TX_BUF_SIZE ETH_MAX_PACKET_SIZE /* buffer size for transmit */

#define ETH_RXBUFNB 4U /* 4 Rx buffers of size ETH_RX_BUF_SIZE */

#define ETH_TXBUFNB 4U /* 4 Tx buffers of size ETH_TX_BUF_SIZE */

/* Section 2: PHY configuration section */

/* DP83848_PHY_ADDRESS Address*/

#define DP83848_PHY_ADDRESS

/* PHY Reset delay these values are based on a 1 ms Systick interrupt*/

#define PHY_RESET_DELAY 0x000000FFU

/* PHY Configuration delay */

#define PHY_CONFIG_DELAY 0x00000FFFU

```
#define PHY_READ_TO                    0x0000FFFFU
#define PHY_WRITE_TO                   0x0000FFFFU
/* Section 3: Common PHY Registers */
#define PHY_BCR                        ((uint16_t)0x0000U)    /*!<
Transceiver Basic Control Register   */
#define PHY_BSR                        ((uint16_t)0x0001U)    /*!<
Transceiver Basic Status Register    */
#define PHY_RESET                      ((uint16_t)0x8000U)  /*!< PHY
Reset */
#define PHY_LOOPBACK                   ((uint16_t)0x4000U)  /*!<
Select loop-back mode */
#define PHY_FULLDUPLEX_100M            ((uint16_t)0x2100U)
/*!< Set the full-duplex mode at 100 Mb/s */
#define PHY_HALFDUPLEX_100M            ((uint16_t)0x2000U)
/*!< Set the half-duplex mode at 100 Mb/s */
#define PHY_FULLDUPLEX_10M             ((uint16_t)0x0100U)
/*!< Set the full-duplex mode at 10 Mb/s  */
#define PHY_HALFDUPLEX_10M             ((uint16_t)0x0000U)
/*!< Set the half-duplex mode at 10 Mb/s  */
#define PHY_AUTONEGOTIATION            ((uint16_t)0x1000U)
/*!< Enable auto-negotiation function     */
#define PHY_RESTART_AUTONEGOTIATION
((uint16_t)0x0200U)  /*!< Restart auto-negotiation function    */
#define PHY_POWERDOWN                  ((uint16_t)0x0800U)  /*!<
Select the power down mode           */
#define PHY_ISOLATE                    ((uint16_t)0x0400U)  /*!< Isolate
PHY from MII                 */
```

```
#define PHY_AUTONEGO_COMPLETE          ((uint16_t)0x0020U)
/*!< Auto-Negotiation process completed   */

#define PHY_LINKED_STATUS              ((uint16_t)0x0004U)  /*!<
Valid link established              */

#define PHY_JABBER_DETECTION           ((uint16_t)0x0002U)
/*!< Jabber condition detected           */

/* Section 4: Extended PHY Registers */

#define PHY_SR                         ((uint16_t))    /*!< PHY status register
Offset                    */

#define PHY_SPEED_STATUS               ((uint16_t))  /*!< PHY Speed
mask                              */

#define PHY_DUPLEX_STATUS              ((uint16_t))  /*!< PHY
Duplex mask                           */

/* ################### SPI peripheral configuration
########################### */

/* CRC FEATURE: Use to activate CRC feature inside HAL SPI Driver

* Activated: CRC code is present inside driver

* Deactivated: CRC code cleaned from driver

*/

#define USE_SPI_CRC                    0U

/* Includes ------------------------------------------------------------------*/

/**

  * @brief Include module's header file

  */

#ifdef HAL_RCC_MODULE_ENABLED
```

```
  #include "stm32f4xx_hal_rcc.h"
#endif /* HAL_RCC_MODULE_ENABLED */
#ifdef HAL_GPIO_MODULE_ENABLED
  #include "stm32f4xx_hal_gpio.h"
#endif /* HAL_GPIO_MODULE_ENABLED */
#ifdef HAL_EXTI_MODULE_ENABLED
  #include "stm32f4xx_hal_exti.h"
#endif /* HAL_EXTI_MODULE_ENABLED */
#ifdef HAL_DMA_MODULE_ENABLED
  #include "stm32f4xx_hal_dma.h"
#endif /* HAL_DMA_MODULE_ENABLED */
#ifdef HAL_CORTEX_MODULE_ENABLED
  #include "stm32f4xx_hal_cortex.h"
#endif /* HAL_CORTEX_MODULE_ENABLED */
#ifdef HAL_ADC_MODULE_ENABLED
  #include "stm32f4xx_hal_adc.h"
#endif /* HAL_ADC_MODULE_ENABLED */
#ifdef HAL_CAN_MODULE_ENABLED
  #include "stm32f4xx_hal_can.h"
#endif /* HAL_CAN_MODULE_ENABLED */
#ifdef HAL_CAN_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_can_legacy.h"
#endif /* HAL_CAN_LEGACY_MODULE_ENABLED */
```

```
#ifdef HAL_CRC_MODULE_ENABLED
  #include "stm32f4xx_hal_crc.h"
#endif /* HAL_CRC_MODULE_ENABLED */
#ifdef HAL_CRYP_MODULE_ENABLED
  #include "stm32f4xx_hal_cryp.h"
#endif /* HAL_CRYP_MODULE_ENABLED */
#ifdef HAL_DMA2D_MODULE_ENABLED
  #include "stm32f4xx_hal_dma2d.h"
#endif /* HAL_DMA2D_MODULE_ENABLED */
#ifdef HAL_DAC_MODULE_ENABLED
  #include "stm32f4xx_hal_dac.h"
#endif /* HAL_DAC_MODULE_ENABLED */
#ifdef HAL_DCMI_MODULE_ENABLED
  #include "stm32f4xx_hal_dcmi.h"
#endif /* HAL_DCMI_MODULE_ENABLED */
#ifdef HAL_ETH_MODULE_ENABLED
  #include "stm32f4xx_hal_eth.h"
#endif /* HAL_ETH_MODULE_ENABLED */
#ifdef HAL_ETH_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_eth_legacy.h"
#endif /* HAL_ETH_LEGACY_MODULE_ENABLED */
#ifdef HAL_FLASH_MODULE_ENABLED
  #include "stm32f4xx_hal_flash.h"
```

```
#endif /* HAL_FLASH_MODULE_ENABLED */
#ifdef HAL_SRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sram.h"
#endif /* HAL_SRAM_MODULE_ENABLED */
#ifdef HAL_NOR_MODULE_ENABLED
  #include "stm32f4xx_hal_nor.h"
#endif /* HAL_NOR_MODULE_ENABLED */
#ifdef HAL_NAND_MODULE_ENABLED
  #include "stm32f4xx_hal_nand.h"
#endif /* HAL_NAND_MODULE_ENABLED */
#ifdef HAL_PCCARD_MODULE_ENABLED
  #include "stm32f4xx_hal_pccard.h"
#endif /* HAL_PCCARD_MODULE_ENABLED */
#ifdef HAL_SDRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sdram.h"
#endif /* HAL_SDRAM_MODULE_ENABLED */
#ifdef HAL_HASH_MODULE_ENABLED
 #include "stm32f4xx_hal_hash.h"
#endif /* HAL_HASH_MODULE_ENABLED */
#ifdef HAL_I2C_MODULE_ENABLED
 #include "stm32f4xx_hal_i2c.h"
#endif /* HAL_I2C_MODULE_ENABLED */
#ifdef HAL_SMBUS_MODULE_ENABLED
```

```
 #include "stm32f4xx_hal_smbus.h"
#endif /* HAL_SMBUS_MODULE_ENABLED */
#ifdef HAL_I2S_MODULE_ENABLED
 #include "stm32f4xx_hal_i2s.h"
#endif /* HAL_I2S_MODULE_ENABLED */
#ifdef HAL_IWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_iwdg.h"
#endif /* HAL_IWDG_MODULE_ENABLED */
#ifdef HAL_LTDC_MODULE_ENABLED
 #include "stm32f4xx_hal_ltdc.h"
#endif /* HAL_LTDC_MODULE_ENABLED */
#ifdef HAL_PWR_MODULE_ENABLED
 #include "stm32f4xx_hal_pwr.h"
#endif /* HAL_PWR_MODULE_ENABLED */
#ifdef HAL_RNG_MODULE_ENABLE
 #include "stm32f4xx_hal_rng.h"
#endif /* HAL_RNG_MODULE_ENABLED */
#ifdef HAL_RTC_MODULE_ENABLED
 #include "stm32f4xx_hal_rtc.h"
#endif /* HAL_RTC_MODULE_ENABLED */
#ifdef HAL_SAI_MODULE_ENABLED
 #include "stm32f4xx_hal_sai.h"
#endif /* HAL_SAI_MODULE_ENABLED */
```

```
#ifdef HAL_SD_MODULE_ENABLED
 #include "stm32f4xx_hal_sd.h"
#endif /* HAL_SD_MODULE_ENABLED */
#ifdef HAL_SPI_MODULE_ENABLED
 #include "stm32f4xx_hal_spi.h"
#endif /* HAL_SPI_MODULE_ENABLED */
#ifdef HAL_TIM_MODULE_ENABLED
 #include "stm32f4xx_hal_tim.h"
#endif /* HAL_TIM_MODULE_ENABLED */
#ifdef HAL_UART_MODULE_ENABLED
 #include "stm32f4xx_hal_uart.h"
#endif /* HAL_UART_MODULE_ENABLED */
#ifdef HAL_USART_MODULE_ENABLED
 #include "stm32f4xx_hal_usart.h"
#endif /* HAL_USART_MODULE_ENABLED */
#ifdef HAL_IRDA_MODULE_ENABLED
 #include "stm32f4xx_hal_irda.h"
#endif /* HAL_IRDA_MODULE_ENABLED */
#ifdef HAL_SMARTCARD_MODULE_ENABLED
 #include "stm32f4xx_hal_smartcard.h"
#endif /* HAL_SMARTCARD_MODULE_ENABLED */
#ifdef HAL_WWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_wwdg.h"
```

```
#endif /* HAL_WWDG_MODULE_ENABLED */
#ifdef HAL_PCD_MODULE_ENABLED
 #include "stm32f4xx_hal_pcd.h"
#endif /* HAL_PCD_MODULE_ENABLED */
#ifdef HAL_HCD_MODULE_ENABLED
 #include "stm32f4xx_hal_hcd.h"
#endif /* HAL_HCD_MODULE_ENABLED */
#ifdef HAL_DSI_MODULE_ENABLED
 #include "stm32f4xx_hal_dsi.h"
#endif /* HAL_DSI_MODULE_ENABLED */
#ifdef HAL_QSPI_MODULE_ENABLED
 #include "stm32f4xx_hal_qspi.h"
#endif /* HAL_QSPI_MODULE_ENABLED */
#ifdef HAL_CEC_MODULE_ENABLED
 #include "stm32f4xx_hal_cec.h"
#endif /* HAL_CEC_MODULE_ENABLED */
#ifdef HAL_FMPI2C_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpi2c.h"
#endif /* HAL_FMPI2C_MODULE_ENABLED */
#ifdef HAL_FMPSMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpsmbus.h"
#endif /* HAL_FMPSMBUS_MODULE_ENABLED */
#ifdef HAL_SPDIFRX_MODULE_ENABLED
```

```
 #include "stm32f4xx_hal_spdifrx.h"
#endif /* HAL_SPDIFRX_MODULE_ENABLED */
#ifdef HAL_DFSDM_MODULE_ENABLED
 #include "stm32f4xx_hal_dfsdm.h"
#endif /* HAL_DFSDM_MODULE_ENABLED */
#ifdef HAL_LPTIM_MODULE_ENABLED
 #include "stm32f4xx_hal_lptim.h"
#endif /* HAL_LPTIM_MODULE_ENABLED */
#ifdef HAL_MMC_MODULE_ENABLED
 #include "stm32f4xx_hal_mmc.h"
#endif /* HAL_MMC_MODULE_ENABLED */
/* Exported macro ------------------------------------------------------------*/
#ifdef  USE_FULL_ASSERT
/**
  * @brief  The assert_param macro is used for function's parameters check.
  * @param  expr If expr is false, it calls assert_failed function
  *         which reports the name of the source file and the source
  *         line number of the call that failed.
  *         If expr is true, it returns no value.
  * @retval None
  */
  #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t *)__FILE__, __LINE__))
```

```
/* Exported functions ------------------------------------------------------- */
  void assert_failed(uint8_t* file, uint32_t line);
#else
  #define assert_param(expr) ((void)0U)
#endif /* USE_FULL_ASSERT */
#ifdef __cplusplus
}
#endif
#endif /* __STM32F4xx_HAL_CONF_H */
```

20.11 CODE OF STM32F4XX_IT.H:

```
/* USER CODE BEGIN Header */
/**
******************************************************************************
  * @file    stm32f4xx_it.h
  * @brief   This file contains the headers of the interrupt handlers.
******************************************************************************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
```

```
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
********************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_IT_H
#define __STM32F4xx_IT_H
#ifdef __cplusplus
 extern "C" {
#endif
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */
/* USER CODE END EC */
```

```
/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */
/* USER CODE END EM */
/* Exported functions prototypes ---------------------------------------------*/
void NMI_Handler(void);
void HardFault_Handler(void);
void MemManage_Handler(void);
void BusFault_Handler(void);
void UsageFault_Handler(void);
void SVC_Handler(void);
void DebugMon_Handler(void);
void PendSV_Handler(void);
void SysTick_Handler(void);
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
#ifdef __cplusplus
}
#endif
#endif /* __STM32F4xx_IT_H */
```

20.12 CODE OF MAIN.C:

```
/* USER CODE BEGIN Header */
/**
```

```
*****************************************************
*******************
  * @file           : main.c
  * @brief          : Main program body
*****************************************************
*******************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*****************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
```

```
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
/* USER CODE END PTD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
UART_HandleTypeDef huart2;
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
```

```
/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */
  /* USER CODE END 1 */
  /* MCU Configuration--------------------------------------------------------*/
  /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
  HAL_Init();
  /* USER CODE BEGIN Init */
  /* USER CODE END Init */
  /* Configure the system clock */
  SystemClock_Config();
  /* USER CODE BEGIN SysInit */
  /* USER CODE END SysInit */
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  /* USER CODE BEGIN 2 */
  /* USER CODE END 2 */
```

```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_
VOLTAGE_SCALE1);
  /** Initializes the RCC Oscillators according to the specified
parameters
```

```
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue =
RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 16;
  RCC_OscInitStruct.PLL.PLLN = 336;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV4;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK

|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource =
RCC_SYSCLKSOURCE_PLLCLK;
```

```
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct,
FLASH_LATENCY_2) != HAL_OK)
  {
    Error_Handler();
  }
}
/**
  * @brief USART2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART2_UART_Init(void)
{
  /* USER CODE BEGIN USART2_Init 0 */
  /* USER CODE END USART2_Init 0 */
  /* USER CODE BEGIN USART2_Init 1 */
  /* USER CODE END USART2_Init 1 */
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 115200;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
```

```
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART2_Init 2 */
  /* USER CODE END USART2_Init 2 */
}
/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */
  /* GPIO Ports Clock Enable */
```

```
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin,
GPIO_PIN_RESET);
  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);
  /*Configure GPIO pin : LD2_Pin */
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);
/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}
/* USER CODE BEGIN 4 */
/* USER CODE END 4 */
```

```
/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return
state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}
#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line
number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
```

```
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

20.12 CODE OF STM32F4XX_HAL_MSP.C:

```
/* USER CODE BEGIN Header */
/**
*******************************************************************
  * @file    stm32f4xx_it.c
  * @brief   Interrupt Service Routines.
*******************************************************************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
```

```
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stm32f4xx_it.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN TD */
/* USER CODE END TD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
```

```
/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/* External variables --------------------------------------------------------*/
/* USER CODE BEGIN EV */
/* USER CODE END EV */
/******************************************************************************/
/*            Cortex-M4 Processor Interruption and Exception Handlers         */
/******************************************************************************/
/**
  * @brief This function handles Non maskable interrupt.
  */
void NMI_Handler(void)
{
  /* USER CODE BEGIN NonMaskableInt_IRQn 0 */
```

```
  /* USER CODE END NonMaskableInt_IRQn 0 */
  /* USER CODE BEGIN NonMaskableInt_IRQn 1 */
   while (1)
  {
  }
  /* USER CODE END NonMaskableInt_IRQn 1 */
}
/**
  * @brief This function handles Hard fault interrupt.
  */
void HardFault_Handler(void)
{
  /* USER CODE BEGIN HardFault_IRQn 0 */
  /* USER CODE END HardFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_HardFault_IRQn 0 */
    /* USER CODE END W1_HardFault_IRQn 0 */
  }
}
/**
  * @brief This function handles Memory management fault.
  */
```

```
void MemManage_Handler(void)
{
  /* USER CODE BEGIN MemoryManagement_IRQn 0 */
  /* USER CODE END MemoryManagement_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_MemoryManagement_IRQn 0 */
    /* USER CODE END W1_MemoryManagement_IRQn 0 */
  }
}
/**
  * @brief This function handles Pre-fetch fault, memory access fault.
  */
void BusFault_Handler(void)
{
  /* USER CODE BEGIN BusFault_IRQn 0 */
  /* USER CODE END BusFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_BusFault_IRQn 0 */
    /* USER CODE END W1_BusFault_IRQn 0 */
  }
}
```

```
/**
  * @brief This function handles Undefined instruction or illegal state.
  */
void UsageFault_Handler(void)
{
  /* USER CODE BEGIN UsageFault_IRQn 0 */


  /* USER CODE END UsageFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_UsageFault_IRQn 0 */
    /* USER CODE END W1_UsageFault_IRQn 0 */
  }
}
/**
  * @brief This function handles System service call via SWI
instruction.
  */
void SVC_Handler(void)
{
  /* USER CODE BEGIN SVCall_IRQn 0 */
  /* USER CODE END SVCall_IRQn 0 */
  /* USER CODE BEGIN SVCall_IRQn 1 */
```

```
  /* USER CODE END SVCall_IRQn 1 */
}
/**
  * @brief This function handles Debug monitor.
  */
void DebugMon_Handler(void)
{
  /* USER CODE BEGIN DebugMonitor_IRQn 0 */
  /* USER CODE END DebugMonitor_IRQn 0 */
  /* USER CODE BEGIN DebugMonitor_IRQn 1 */
  /* USER CODE END DebugMonitor_IRQn 1 */
}
/**
  * @brief This function handles Pendable request for system service.
  */
void PendSV_Handler(void)
{
  /* USER CODE BEGIN PendSV_IRQn 0 */
  /* USER CODE END PendSV_IRQn 0 */
  /* USER CODE BEGIN PendSV_IRQn 1 */
  /* USER CODE END PendSV_IRQn 1 */
}
/**
```

```
  * @brief This function handles System tick timer.
  */
void SysTick_Handler(void)
{
  /* USER CODE BEGIN SysTick_IRQn 0 */
  /* USER CODE END SysTick_IRQn 0 */
  HAL_IncTick();
  /* USER CODE BEGIN SysTick_IRQn 1 */
  /* USER CODE END SysTick_IRQn 1 */
}
/******************************************************************************/
/* STM32F4xx Peripheral Interrupt Handlers                                    */
/* Add here the Interrupt Handlers for the used peripherals.                  */
/* For the available peripheral interrupt handler names,                      */
/* please refer to the startup file (startup_stm32f4xx.s).                   */
/******************************************************************************/
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

20.13 CODE OF SYSCALLS.C:

```
/**
*******************************************************************************
```

```
 * @file      syscalls.c
 * @author    Auto-generated by STM32CubeIDE
 * @brief     STM32CubeIDE Minimal System calls file
 *
 *            For more information about which c-functions
 *            need which of these lowlevel functions
 *            please consult the Newlib libc-manual
**********************************************************
********************
 * @attention
 *
 * Copyright (c) 2020-2024 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the
LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *

**********************************************************
********************
 */
/* Includes */
#include <sys/stat.h>
```

```
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
#include <sys/times.h>
/* Variables */
extern int __io_putchar(int ch) __attribute__((weak));
extern int __io_getchar(void) __attribute__((weak));
char *__env[1] = { 0 };
char **environ = __env;
/* Functions */
void initialise_monitor_handles()
{
}
int _getpid(void)
{
  return 1;
}
int _kill(int pid, int sig)
{
  (void)pid;
```

```
  (void)sig;
  errno = EINVAL;
  return -1;
}
void _exit (int status)
{
  _kill(status, -1);
  while (1) {}    /* Make sure we hang here */
}
__attribute__((weak)) int _read(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;
  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    *ptr++ = __io_getchar();
  }
  return len;
}
__attribute__((weak)) int _write(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;
```

```
  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    __io_putchar(*ptr++);
  }
  return len;
}
int _close(int file)
{
  (void)file;
  return -1;
}
int _fstat(int file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
int _isatty(int file)
{
  (void)file;
  return 1;
}
int _lseek(int file, int ptr, int dir)
```

```
{
  (void)file;
  (void)ptr;
  (void)dir;
  return 0;
}
int _open(char *path, int flags, ...)
{
  (void)path;
  (void)flags;
  /* Pretend like we always fail */
  return -1;
}
int _wait(int *status)
{
  (void)status;
  errno = ECHILD;
  return -1;
}
int _unlink(char *name)
{
  (void)name;
  errno = ENOENT;
```

```
  return -1;
}
int _times(struct tms *buf)
{
  (void)buf;
  return -1;
}
int _stat(char *file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
int _link(char *old, char *new)
{
  (void)old;
  (void)new;
  errno = EMLINK;
  return -1;
}
int _fork(void)
{
  errno = EAGAIN;
```

```
  return -1;
}
int _execve(char *name, char **argv, char **env)
{
  (void)name;
  (void)argv;
  (void)env;
  errno = ENOMEM;
  return -1;
}
```

20.14 CODE OF SYSMEM.C:

```
/**
*****************************************************************
*******************
 * @file      sysmem.c
 * @author    Generated by STM32CubeIDE
 * @brief     STM32CubeIDE System Memory calls file
 *
 *            For more information about which C functions
 *            need which of these lowlevel functions
 *            please consult the newlib libc manual
*****************************************************************
*******************
 * @attention
```

```
 *
 * Copyright (c) 2024 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the
LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *

**********************************************************
********************
 */

/* Includes */
#include <errno.h>
#include <stdint.h>

/**
 * Pointer to the current high watermark of the heap usage
 */
static uint8_t *__sbrk_heap_end = NULL;

/**
```

```
 * @brief _sbrk() allocates memory to the newlib heap and is used by malloc
 *        and others from the C library
 *
 * @verbatim
 *
 ##########################################################################
 * #  .data  #  .bss  #       newlib heap       #          MSP stack          #
 * #         #        #                         # Reserved by _Min_Stack_Size #
 *
 ##########################################################################
 * ^-- RAM start      ^-- _end                              _estack, RAM end --^
 * @endverbatim
 *
 * This implementation starts allocating at the '_end' linker symbol
 * The '_Min_Stack_Size' linker symbol reserves a memory for the MSP stack
 * The implementation considers '_estack' linker symbol to be RAM end
 * NOTE: If the MSP stack, at any point during execution, grows larger than the
 * reserved size, please increase the '_Min_Stack_Size'.
 *
 * @param incr Memory size
```

```
 * @return Pointer to allocated memory
 */
void *_sbrk(ptrdiff_t incr)
{
  extern uint8_t _end; /* Symbol defined in the linker script */
  extern uint8_t _estack; /* Symbol defined in the linker script */
  extern uint32_t _Min_Stack_Size; /* Symbol defined in the linker
script */
  const uint32_t stack_limit = (uint32_t)&_estack -
(uint32_t)&_Min_Stack_Size;
  const uint8_t *max_heap = (uint8_t *)stack_limit;
  uint8_t *prev_heap_end;
  /* Initialize heap end at first call */
  if (NULL == __sbrk_heap_end)
  {
    __sbrk_heap_end = &_end;
  }
  /* Protect heap from growing into the reserved MSP stack */
  if (__sbrk_heap_end + incr > max_heap)
  {
    errno = ENOMEM;
    return (void *)-1;
  }
  prev_heap_end = __sbrk_heap_end;
```

```
  __sbrk_heap_end += incr;
  return (void *)prev_heap_end;
}
```

20.15 CODE OF SYSTEM_STM32F4XX.C:

```
/**
**********************************************************
*******************
  * @file    system_stm32f4xx.c
  * @author  MCD Application Team
  * @brief   CMSIS Cortex-M4 Device Peripheral Access Layer System
Source File.
  *
  *   This file provides two functions and one global variable to be called
from
  *   user application:
  *      - SystemInit(): This function is called at startup just after reset and
  *                      before branch to main program. This call is made inside
  *                      the "startup_stm32f4xx.s" file.
  *
  *      - SystemCoreClock variable: Contains the core clock (HCLK), it
can be used
  *                              by the user application to setup the SysTick
  *                              timer or configure other parameters.
  *
```

```
  *      - SystemCoreClockUpdate(): Updates the variable
SystemCoreClock and must
  *                          be called whenever the core clock is changed
  *                          during program execution.
  *
  *
*****************************************************************
*******************
  * @attention
  *
  * Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *

*****************************************************************
*******************
  */
/** @addtogroup CMSIS
  * @{
  */
```

```
/** @addtogroup stm32f4xx_system
  * @{
  */
/** @addtogroup STM32F4xx_System_Private_Includes
  * @{
  */
#include "stm32f4xx.h"
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    ((uint32_t)25000000) /*!< Default value of
the External oscillator in Hz */
#endif /* HSE_VALUE */
#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000) /*!< Value of the
Internal oscillator in Hz*/
#endif /* HSI_VALUE */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_TypesDefinitions
  * @{
  */
/**
  * @}
  */
```

```
/** @addtogroup STM32F4xx_System_Private_Defines
  * @{
  */
/************************* Miscellaneous Configuration
************************/
/*!< Uncomment the following line if you need to use external SRAM or
SDRAM as data memory  */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx) || defined(STM32F417xx)\
 || defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx) ||
defined(STM32F412Zx) || defined(STM32F412Vx)
/* #define DATA_IN_ExtSRAM */
#endif /* STM32F40xxx || STM32F41xxx || STM32F42xxx ||
STM32F43xxx || STM32F469xx || STM32F479xx ||\
        STM32F412Zx || STM32F412Vx */
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F446xx) || defined(STM32F469xx) ||
defined(STM32F479xx)
/* #define DATA_IN_ExtSDRAM */
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F446xx || STM32F469xx ||\
        STM32F479xx */
```

/* Note: Following vector table addresses must be defined in line with linker

configuration. */

/*!< Uncomment the following line if you need to relocate the vector table

anywhere in Flash or Sram, else the vector table is kept at the automatic

remap of boot address selected */

/* #define USER_VECT_TAB_ADDRESS */

#if defined(USER_VECT_TAB_ADDRESS)

/*!< Uncomment the following line if you need to relocate your vector Table

in Sram else user remap will be done in Flash. */

/* #define VECT_TAB_SRAM */

#if defined(VECT_TAB_SRAM)

#define VECT_TAB_BASE_ADDRESS SRAM_BASE /*!< Vector Table base address field.

This value must be a multiple of 0x200. */

#define VECT_TAB_OFFSET 0x00000000U /*!< Vector Table base offset field.

This value must be a multiple of 0x200. */

#else

#define VECT_TAB_BASE_ADDRESS FLASH_BASE /*!< Vector Table base address field.

```
                                        This value must be a multiple of 0x200.
*/
#define VECT_TAB_OFFSET         0x00000000U     /*!< Vector Table
base offset field.
                                        This value must be a multiple of 0x200.
*/
#endif /* VECT_TAB_SRAM */
#endif /* USER_VECT_TAB_ADDRESS */
/*****************************************************************
*********************/
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Macros
  * @{
  */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Variables
  * @{
  */
  /* This variable is updated in three ways:
     1) by calling CMSIS function SystemCoreClockUpdate()
```

```
     2) by calling HAL API function HAL_RCC_GetHCLKFreq()
     3) each time HAL_RCC_ClockConfig() is called to configure the system clock frequency
        Note: If you use this function to configure the system clock; then there
              is no need to call the 2 first functions listed above, since SystemCoreClock
              variable is updated automatically.
  */
uint32_t SystemCoreClock = 16000000;
const uint8_t AHBPrescTable[16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9};
const uint8_t APBPrescTable[8]  = {0, 0, 0, 0, 1, 2, 3, 4};
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_FunctionPrototypes
  * @{
  */
#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  static void SystemInit_ExtMemCtl(void);
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */
/**
  * @}
```

```
  */
/** @addtogroup STM32F4xx_System_Private_Functions
  * @{
  */
/**
  * @brief  Setup the microcontroller system
  *         Initialize the FPU setting, vector table location and External memory
  *         configuration.
  * @param  None
  * @retval None
  */
void SystemInit(void)
{
  /* FPU settings ------------------------------------------------------------*/
  #if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
    SCB->CPACR |= ((3UL << 10*2)|(3UL << 11*2));  /* set CP10 and CP11 Full Access */
  #endif
#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  SystemInit_ExtMemCtl();
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */
  /* Configure the Vector Table location -------------------------------------*/
#if defined(USER_VECT_TAB_ADDRESS)
```

```
  SCB->VTOR = VECT_TAB_BASE_ADDRESS |
VECT_TAB_OFFSET; /* Vector Table Relocation in Internal SRAM */
#endif /* USER_VECT_TAB_ADDRESS */
}
/**
   * @brief  Update SystemCoreClock variable according to Clock
Register Values.
  *         The SystemCoreClock variable contains the core clock (HCLK),
it can
  *         be used by the user application to setup the SysTick timer or
configure
  *         other parameters.
  *
  * @note   Each time the core clock (HCLK) changes, this function
must be called
  *         to update SystemCoreClock variable value. Otherwise, any
configuration
  *         based on this variable will be incorrect.
  *
  * @note   - The system frequency computed by this function is not the
real
  *           frequency in the chip. It is calculated based on the predefined
  *           constant and the selected clock source:
  *
```

```
  *          - If SYSCLK source is HSI, SystemCoreClock will contain the
HSI_VALUE(*)
  *
  *          - If SYSCLK source is HSE, SystemCoreClock will contain the
HSE_VALUE(**)
  *
  *          - If SYSCLK source is PLL, SystemCoreClock will contain the
HSE_VALUE(**)
  *            or HSI_VALUE(*) multiplied/divided by the PLL factors.
  *
  *        (*) HSI_VALUE is a constant defined in stm32f4xx_hal_conf.h
file (default value
  *            16 MHz) but the real value may vary depending on the
variations
  *            in voltage and temperature.
  *
  *        (**) HSE_VALUE is a constant defined in
stm32f4xx_hal_conf.h file (its value
  *             depends on the application requirements), user has to ensure
that HSE_VALUE
  *             is same as the real frequency of the crystal used. Otherwise,
this function
  *             may have wrong result.
  *
  *        - The result of this function could be not correct when using
fractional
```

```
  *          value for HSE crystal.
  *
  * @param  None
  * @retval None
  */
void SystemCoreClockUpdate(void)
{
  uint32_t tmp = 0, pllvco = 0, pllp = 2, pllsource = 0, pllm = 2;
  /* Get SYSCLK source -------------------------------------------------------
*/
  tmp = RCC->CFGR & RCC_CFGR_SWS;
  switch (tmp)
  {
    case 0x00:  /* HSI used as system clock source */
      SystemCoreClock = HSI_VALUE;
      break;
    case 0x04:  /* HSE used as system clock source */
      SystemCoreClock = HSE_VALUE;
      break;
    case 0x08:  /* PLL used as system clock source */
      /* PLL_VCO = (HSE_VALUE or HSI_VALUE / PLL_M) * PLL_N
         SYSCLK = PLL_VCO / PLL_P
         */
```

```
    pllsource = (RCC->PLLCFGR & RCC_PLLCFGR_PLLSRC) >>
22;
    pllm = RCC->PLLCFGR & RCC_PLLCFGR_PLLM;
    if (pllsource != 0)
    {
      /* HSE used as PLL clock source */
      pllvco = (HSE_VALUE / pllm) * ((RCC->PLLCFGR &
RCC_PLLCFGR_PLLN) >> 6);
    }
    else
    {
      /* HSI used as PLL clock source */
      pllvco = (HSI_VALUE / pllm) * ((RCC->PLLCFGR &
RCC_PLLCFGR_PLLN) >> 6);
    }

    pllp = (((RCC->PLLCFGR & RCC_PLLCFGR_PLLP) >>16) + 1 )
*2;
    SystemCoreClock = pllvco/pllp;
    break;
   default:
    SystemCoreClock = HSI_VALUE;
    break;
  }
```

```
  /* Compute HCLK frequency ----------------------------------------------
-*/
  /* Get HCLK prescaler */
  tmp = AHBPrescTable[((RCC->CFGR & RCC_CFGR_HPRE) >> 4)];
  /* HCLK frequency */
  SystemCoreClock >>= tmp;
}
#if defined (DATA_IN_ExtSRAM) && defined
(DATA_IN_ExtSDRAM)
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
/**
  * @brief  Setup the external memory controller.
  *         Called in startup_stm32f4xx.s before jump to main.
  *         This function configures the external memories
(SRAM/SDRAM)
  *         This SRAM/SDRAM will be used as program data memory
(including heap and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;
```

```
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
  register __IO uint32_t index;
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and
GPIOI interface clock */
  RCC->AHB1ENR |= 0x000001F8;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x00CCC0CC;
  GPIOD->AFR[1]  = 0xCCCCCCCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xAAAA0A8A;
  /* Configure PDx pins speed to 100 MHz */
  GPIOD->OSPEEDR = 0xFFFF0FCF;
  /* Configure PDx pins Output type to push-pull */
  GPIOD->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOD->PUPDR   = 0x00000000;
  /* Connect PEx pins to FMC Alternate function */
  GPIOE->AFR[0]  = 0xC00CC0CC;
  GPIOE->AFR[1]  = 0xCCCCCCCC;
  /* Configure PEx pins in Alternate function mode */
  GPIOE->MODER   = 0xAAAA828A;
```

```
/* Configure PEx pins speed to 100 MHz */
GPIOE->OSPEEDR = 0xFFFFC3CF;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0xCCCCCCCC;
GPIOF->AFR[1]  = 0xCCCCCCCC;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA800AAA;
/* Configure PFx pins speed to 50 MHz */
GPIOF->OSPEEDR = 0xAA800AAA;
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
GPIOF->PUPDR   = 0x00000000;
/* Connect PGx pins to FMC Alternate function */
GPIOG->AFR[0]  = 0xCCCCCCCC;
GPIOG->AFR[1]  = 0xCCCCCCCC;
/* Configure PGx pins in Alternate function mode */
GPIOG->MODER   = 0xAAAAAAAA;
/* Configure PGx pins speed to 50 MHz */
```

```
GPIOG->OSPEEDR = 0xAAAAAAAA;
/* Configure PGx pins Output type to push-pull */
GPIOG->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PGx pins */
GPIOG->PUPDR   = 0x00000000;
/* Connect PHx pins to FMC Alternate function */
GPIOH->AFR[0]  = 0x00C0CC00;
GPIOH->AFR[1]  = 0xCCCCCCCC;
/* Configure PHx pins in Alternate function mode */
GPIOH->MODER   = 0xAAAA08A0;
/* Configure PHx pins speed to 50 MHz */
GPIOH->OSPEEDR = 0xAAAA08A0;
/* Configure PHx pins Output type to push-pull */
GPIOH->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PHx pins */
GPIOH->PUPDR   = 0x00000000;
/* Connect PIx pins to FMC Alternate function */
GPIOI->AFR[0]  = 0xCCCCCCCC;
GPIOI->AFR[1]  = 0x00000CC0;
/* Configure PIx pins in Alternate function mode */
GPIOI->MODER   = 0x0028AAAA;
/* Configure PIx pins speed to 50 MHz */
GPIOI->OSPEEDR = 0x0028AAAA;
```

```
  /* Configure PIx pins Output type to push-pull */
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;
/*-- FMC Configuration ------------------------------------------------------*/
  /* Enable the FMC interface clock */
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
  FMC_Bank5_6->SDTR[0] = 0x01115351;
  /* SDRAM initialization sequence */
  /* Clock enable command */
  FMC_Bank5_6->SDCMR = 0x00000011;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Delay */
  for (index = 0; index<1000; index++);
  /* PALL command */
  FMC_Bank5_6->SDCMR = 0x00000012;
```

```
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* Auto refresh command */
FMC_Bank5_6->SDCMR = 0x00000073;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* MRD register program */
FMC_Bank5_6->SDCMR = 0x00046014;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* Set refresh count */
```

```
  tmpreg = FMC_Bank5_6->SDRTR;
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
  /* Disable write protection */
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F469xx || STM32F479xx */
  (void)(tmp);
}
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F469xx || STM32F479xx */
```

```
#elif defined (DATA_IN_ExtSRAM) || defined
(DATA_IN_ExtSDRAM)
/**
  * @brief  Setup the external memory controller.
  *         Called in startup_stm32f4xx.s before jump to main.
  *         This function configures the external memories
(SRAM/SDRAM)
  *         This SRAM/SDRAM will be used as program data memory
(including heap and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F446xx) || defined(STM32F469xx) ||
defined(STM32F479xx)
#if defined (DATA_IN_ExtSDRAM)
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
  register __IO uint32_t index;
#if defined(STM32F446xx)
  /* Enable GPIOA, GPIOC, GPIOD, GPIOE, GPIOF, GPIOG interface
     clock */
```

```
  RCC->AHB1ENR |= 0x0000007D;
#else
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and
GPIOI interface
     clock */
  RCC->AHB1ENR |= 0x000001F8;
#endif /* STM32F446xx */
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);
#if defined(STM32F446xx)
  /* Connect PAx pins to FMC Alternate function */
  GPIOA->AFR[0]  |= 0xC0000000;
  GPIOA->AFR[1]  |= 0x00000000;
  /* Configure PDx pins in Alternate function mode */
  GPIOA->MODER   |= 0x00008000;
  /* Configure PDx pins speed to 50 MHz */
  GPIOA->OSPEEDR |= 0x00008000;
  /* Configure PDx pins Output type to push-pull */
  GPIOA->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOA->PUPDR   |= 0x00000000;
  /* Connect PCx pins to FMC Alternate function */
  GPIOC->AFR[0]  |= 0x00CC0000;
```

```
  GPIOC->AFR[1]  |= 0x00000000;
  /* Configure PDx pins in Alternate function mode */
  GPIOC->MODER   |= 0x00000A00;
  /* Configure PDx pins speed to 50 MHz */
  GPIOC->OSPEEDR |= 0x00000A00;
  /* Configure PDx pins Output type to push-pull */
  GPIOC->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOC->PUPDR   |= 0x00000000;
#endif /* STM32F446xx */
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x000000CC;
  GPIOD->AFR[1]  = 0xCC000CCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xA02A000A;
  /* Configure PDx pins speed to 50 MHz */
  GPIOD->OSPEEDR = 0xA02A000A;
  /* Configure PDx pins Output type to push-pull */
  GPIOD->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOD->PUPDR   = 0x00000000;
  /* Connect PEx pins to FMC Alternate function */
  GPIOE->AFR[0]  = 0xC00000CC;
```

```
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA800A;
/* Configure PEx pins speed to 50 MHz */
GPIOE->OSPEEDR = 0xAAAA800A;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0xCCCCCCCC;
GPIOF->AFR[1]  = 0xCCCCCCCC;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA800AAA;
/* Configure PFx pins speed to 50 MHz */
GPIOF->OSPEEDR = 0xAA800AAA;
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
GPIOF->PUPDR   = 0x00000000;
/* Connect PGx pins to FMC Alternate function */
GPIOG->AFR[0]  = 0xCCCCCCCC;
GPIOG->AFR[1]  = 0xCCCCCCCC;
```

```
  /* Configure PGx pins in Alternate function mode */
  GPIOG->MODER   = 0xAAAAAAAA;
  /* Configure PGx pins speed to 50 MHz */
  GPIOG->OSPEEDR = 0xAAAAAAAA;
  /* Configure PGx pins Output type to push-pull */
  GPIOG->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
  /* Connect PHx pins to FMC Alternate function */
  GPIOH->AFR[0]  = 0x00C0CC00;
  GPIOH->AFR[1]  = 0xCCCCCCCC;
  /* Configure PHx pins in Alternate function mode */
  GPIOH->MODER   = 0xAAAA08A0;
  /* Configure PHx pins speed to 50 MHz */
  GPIOH->OSPEEDR = 0xAAAA08A0;
  /* Configure PHx pins Output type to push-pull */
  GPIOH->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PHx pins */
  GPIOH->PUPDR   = 0x00000000;
  /* Connect PIx pins to FMC Alternate function */
```

```
  GPIOI->AFR[0]  = 0xCCCCCCCC;
  GPIOI->AFR[1]  = 0x00000CC0;
  /* Configure PIx pins in Alternate function mode */
  GPIOI->MODER   = 0x0028AAAA;
  /* Configure PIx pins speed to 50 MHz */
  GPIOI->OSPEEDR = 0x0028AAAA;
  /* Configure PIx pins Output type to push-pull */
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F469xx || STM32F479xx */
/*-- FMC Configuration ------------------------------------------------------*/
  /* Enable the FMC interface clock */
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable SDRAM bank1 */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCR[0] = 0x00001954;
#else
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
#endif /* STM32F446xx */
```

```
  FMC_Bank5_6->SDTR[0] = 0x01115351;
  /* SDRAM initialization sequence */
  /* Clock enable command */
  FMC_Bank5_6->SDCMR = 0x00000011;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Delay */
  for (index = 0; index<1000; index++);
  /* PALL command */
  FMC_Bank5_6->SDCMR = 0x00000012;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Auto refresh command */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x000000F3;
#else
```

```
  FMC_Bank5_6->SDCMR = 0x00000073;
#endif /* STM32F446xx */
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* MRD register program */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x00044014;
#else
  FMC_Bank5_6->SDCMR = 0x00046014;
#endif /* STM32F446xx */
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Set refresh count */
  tmpreg = FMC_Bank5_6->SDRTR;
#if defined(STM32F446xx)
```

```
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000050C<<1));
#else
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
#endif /* STM32F446xx */
  /* Disable write protection */
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#endif /* DATA_IN_ExtSDRAM */
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F446xx || STM32F469xx || STM32F479xx */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx) || defined(STM32F417xx)\
 || defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx) ||
defined(STM32F412Zx) || defined(STM32F412Vx)
#if defined(DATA_IN_ExtSRAM)
/*-- GPIOs Configuration -----------------------------------------------------*/
   /* Enable GPIOD, GPIOE, GPIOF and GPIOG interface clock */
  RCC->AHB1ENR   |= 0x00000078;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIODEN);
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x00CCC0CC;
```

```
GPIOD->AFR[1]  = 0xCCCCCCCC;
/* Configure PDx pins in Alternate function mode */
GPIOD->MODER   = 0xAAAA0A8A;
/* Configure PDx pins speed to 100 MHz */
GPIOD->OSPEEDR = 0xFFFF0FCF;
/* Configure PDx pins Output type to push-pull */
GPIOD->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PDx pins */
GPIOD->PUPDR   = 0x00000000;
/* Connect PEx pins to FMC Alternate function */
GPIOE->AFR[0]  = 0xC00CC0CC;
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA828A;
/* Configure PEx pins speed to 100 MHz */
GPIOE->OSPEEDR = 0xFFFFC3CF;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0x00CCCCCC;
GPIOF->AFR[1]  = 0xCCCC0000;
```

```
  /* Configure PFx pins in Alternate function mode */
  GPIOF->MODER   = 0xAA000AAA;
  /* Configure PFx pins speed to 100 MHz */
  GPIOF->OSPEEDR = 0xFF000FFF;
  /* Configure PFx pins Output type to push-pull */
  GPIOF->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PFx pins */
  GPIOF->PUPDR   = 0x00000000;
  /* Connect PGx pins to FMC Alternate function */
  GPIOG->AFR[0]  = 0x00CCCCCC;
  GPIOG->AFR[1]  = 0x000000C0;
  /* Configure PGx pins in Alternate function mode */
  GPIOG->MODER   = 0x00085AAA;
  /* Configure PGx pins speed to 100 MHz */
  GPIOG->OSPEEDR = 0x000CAFFF;
  /* Configure PGx pins Output type to push-pull */
  GPIOG->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;
/*-- FMC/FSMC Configuration ----------------------------------------------
-*/
  /* Enable the FMC/FSMC interface clock */
  RCC->AHB3ENR         |= 0x00000001;
```

```
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F469xx || STM32F479xx */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx)|| defined(STM32F417xx)\
   || defined(STM32F412Zx) || defined(STM32F412Vx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FSMCEN);
  /* Configure and enable Bank1_SRAM2 */
```

```
  FSMC_Bank1->BTCR[2]  = 0x00001011;
  FSMC_Bank1->BTCR[3]  = 0x00000201;
  FSMC_Bank1E->BWTR[2] = 0x0FFFFFFF;
#endif /* STM32F405xx || STM32F415xx || STM32F407xx ||
STM32F417xx || STM32F412Zx || STM32F412Vx */
#endif /* DATA_IN_ExtSRAM */
#endif /* STM32F405xx || STM32F415xx || STM32F407xx ||
STM32F417xx || STM32F427xx || STM32F437xx ||\
        STM32F429xx || STM32F439xx || STM32F469xx ||
STM32F479xx || STM32F412Zx || STM32F412Vx  */
  (void)(tmp);
}
#endif /* DATA_IN_ExtSRAM && DATA_IN_ExtSDRAM */
/**
  * @}
  */
/**
  * @}
  */
/**
  * @}
  */
```

20.16 CODE OF STM32F4111RETX_FLASH.LD:

```
/*
```

```
*****************************************************
*******************
**
** @file        : LinkerScript.ld
**
** @author      : Auto-generated by STM32CubeIDE
**
**  Abstract    : Linker script for NUCLEO-F411RE Board embedding
STM32F411RETx Device from stm32f4 series
**                  512KBytes FLASH
**                  128KBytes RAM
**
**              Set heap size, stack size and stack location according
**              to application requirements.
**
**              Set memory bank area and size if external memory is used
**
**  Target      : STMicroelectronics STM32
**
**  Distribution: The file is distributed as is, without any warranty
**              of any kind.
**
*****************************************************
*******************
```

```
** @attention
**
** Copyright (c) 2024 STMicroelectronics.
** All rights reserved.
**
** This software is licensed under terms that can be found in the
LICENSE file
** in the root directory of this software component.
** If no LICENSE file comes with this software, it is provided AS-IS.
**
*****************************************************************
*******************
*/
/* Entry Point */
ENTRY(Reset_Handler)
/* Highest address of the user mode stack */
_estack = ORIGIN(RAM) + LENGTH(RAM); /* end of "RAM" Ram
type memory */
_Min_Heap_Size = 0x200; /* required amount of heap */
_Min_Stack_Size = 0x400; /* required amount of stack */
/* Memories definition */
MEMORY
{
  RAM    (xrw)    : ORIGIN = 0x20000000,   LENGTH = 128K
```

```
  FLASH    (rx)    : ORIGIN = 0x8000000,   LENGTH = 512K
}
/* Sections */
SECTIONS
{
  /* The startup code into "FLASH" Rom type memory */
  .isr_vector :
  {
    . = ALIGN(4);
    KEEP(*(.isr_vector)) /* Startup code */
    . = ALIGN(4);
  } >FLASH

  /* The program code and other data into "FLASH" Rom type memory
*/
  .text :
  {
    . = ALIGN(4);
    *(.text)          /* .text sections (code) */
    *(.text*)         /* .text* sections (code) */
    *(.glue_7)        /* glue arm to thumb code */
    *(.glue_7t)       /* glue thumb to arm code */
    *(.eh_frame)
```

```
    KEEP (*(.init))
    KEEP (*(.fini))

    . = ALIGN(4);
    _etext = .;        /* define a global symbols at end of code */
  } >FLASH
  /* Constant data into "FLASH" Rom type memory */
  .rodata :
  {
    . = ALIGN(4);
    *(.rodata)         /* .rodata sections (constants, strings, etc.) */
    *(.rodata*)        /* .rodata* sections (constants, strings, etc.) */
    . = ALIGN(4);
  } >FLASH
  .ARM.extab (READONLY) : /* The "READONLY" keyword is only
supported in GCC11 and later, remove it if using GCC10 or earlier. */
  {
    . = ALIGN(4);
    *(.ARM.extab* .gnu.linkonce.armextab.*)
    . = ALIGN(4);
  } >FLASH
  .ARM (READONLY) : /* The "READONLY" keyword is only
supported in GCC11 and later, remove it if using GCC10 or earlier. */
```

```
  {
    . = ALIGN(4);
    __exidx_start = .;
    *(.ARM.exidx*)
    __exidx_end = .;
    . = ALIGN(4);
  } >FLASH
  .preinit_array (READONLY) : /* The "READONLY" keyword is only
supported in GCC11 and later, remove it if using GCC10 or earlier. */
  {
    . = ALIGN(4);
    PROVIDE_HIDDEN (__preinit_array_start = .);
    KEEP (*(.preinit_array*))
    PROVIDE_HIDDEN (__preinit_array_end = .);
    . = ALIGN(4);
  } >FLASH
  .init_array (READONLY) : /* The "READONLY" keyword is only
supported in GCC11 and later, remove it if using GCC10 or earlier. */
  {
    . = ALIGN(4);
    PROVIDE_HIDDEN (__init_array_start = .);
    KEEP (*(SORT(.init_array.*)))
    KEEP (*(.init_array*))
    PROVIDE_HIDDEN (__init_array_end = .);
```

```
    . = ALIGN(4);
  } >FLASH
  .fini_array (READONLY) : /* The "READONLY" keyword is only
supported in GCC11 and later, remove it if using GCC10 or earlier. */
  {
    . = ALIGN(4);
    PROVIDE_HIDDEN (__fini_array_start = .);
    KEEP (*(SORT(.fini_array.*)))
    KEEP (*(.fini_array*))
    PROVIDE_HIDDEN (__fini_array_end = .);
    . = ALIGN(4);
  } >FLASH
  /* Used by the startup to initialize data */
  _sidata = LOADADDR(.data);
  /* Initialized data sections into "RAM" Ram type memory */
  .data :
  {
    . = ALIGN(4);
    _sdata = .;        /* create a global symbol at data start */
    *(.data)           /* .data sections */
    *(.data*)          /* .data* sections */
    *(.RamFunc)        /* .RamFunc sections */
    *(.RamFunc*)       /* .RamFunc* sections */
```

```
  . = ALIGN(4);
  _edata = .;       /* define a global symbol at data end */
} >RAM AT> FLASH
/* Uninitialized data section into "RAM" Ram type memory */
. = ALIGN(4);
.bss :
{
  /* This is used by the startup in order to initialize the .bss section */
  _sbss = .;        /* define a global symbol at bss start */
  __bss_start__ = _sbss;
  *(.bss)
  *(.bss*)
  *(COMMON)
  . = ALIGN(4);
  _ebss = .;        /* define a global symbol at bss end */
  __bss_end__ = _ebss;
} >RAM
/* User_heap_stack section, used to check that there is enough "RAM"
Ram  type memory left */
._user_heap_stack :
{
  . = ALIGN(8);
  PROVIDE ( end = . );
```

```
    PROVIDE ( _end = . );
    . = . + _Min_Heap_Size;
    . = . + _Min_Stack_Size;
    . = ALIGN(8);
  } >RAM
  /* Remove information from the compiler libraries */
  /DISCARD/ :
  {
    libc.a ( * )
    libm.a ( * )
    libgcc.a ( * )
  }
  .ARM.attributes 0 : { *(.ARM.attributes) }
}
```

CHAPTER#21CODING GPIO DRIVERS FROM SCRATCH IN ASSEMBLY LANGUAGE

21.1CODING GPIO DRIVERS FROM SCRATCH IN ASSEMBLY LANGUAGE:

Coding GPIO drivers from scratch in assembly language involves direct manipulation of hardware registers using ARM assembly instructions. This approach provides fine-grained control over the hardware, essential for low-level system programming. Here's a guide to help you create GPIO drivers in assembly for an ARM-based microcontroller, such as those from the STM32 family.

21.2 PREREQUISITES:

21.2.1 ARM ARCHITECTURE KNOWLEDGE:

Familiarity with the ARM instruction set, particularly ARM Cortex-M series.

21.2.2 DATASHEET AND REFERENCES MANUAL:

Understanding of the specific microcontroller's GPIO registers and memory mapping.

21.2.3 DEVELOPMENT ENVIRONMENT:

STM32CubeIDE or another suitable IDE with ARM toolchain support.

21.3 BASIC CONCEPTS

21.3.1 REGISTERS:

GPIO configuration and data registers.

21.3.2 MEMORY MAPPED I/O:

Direct access to hardware registers via memory addresses.

21.3.3 ASSEMBLY INSTRUCTIONS:

ARM instructions like LDR, STR, B, MOV, etc.

21.4 STEPS TO CODE GPIO DRIVERS IN ASSEMBLY:

21.4.1 SETTING UP THE PROJECT:

Create a new project in your development environment. Use the linker script and startup file to define the memory layout and entry point.

21.4.2 DEFINING MEMORY ADDRESSES:

Define the base addresses for GPIO ports and specific registers. These addresses are found in the microcontroller's reference manual.

21.4.3 ENABLING THE GPIO CLOCK:

Enable the clock for the GPIO port using the RCC (Reset and Clock Control) registers.

21.4.4 CONFIGURING GPIO PIN MODES:

Set the mode for each GPIO pin using the MODER (mode register). Each pin requires two bits for mode configuration.

21.4.5 WRITING TO AND READING FROM GPIO PINS:

Use the ODR (Output Data Register) and IDR (Input Data Register) to write to and read from GPIO pins.

21.4.6 HANDLING MULTIPLE PINS:

To handle multiple GPIO pins simultaneously, you can configure or modify multiple bits at once, using bitwise operations to ensure only the intended bits are affected.

21.4.7 INTERRUPT HANDLING:

For handling GPIO interrupts, set up the NVIC (Nested Vectored Interrupt Controller) and configure GPIO pins for external interrupt lines. The actual implementation of interrupt service routines (ISRs) depends on the specific microcontroller and project requirements.

21.5 CODE OF MAIN.H:

```
/* USER CODE BEGIN Header */
/**
**********************************************************
********************
  * @file           : main.h
  * @brief          : Header for main.c file.
  *                   This file contains the common defines of the application.
**********************************************************
********************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
```

```
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H
#ifdef __cplusplus
extern "C" {
#endif
/* Includes ------------------------------------------------------------------*/
#include "stm32f4xx_hal.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */
/* USER CODE END EC */
/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */
```

```
/* USER CODE END EM */
/* Exported functions prototypes ---------------------------------------------*/
void Error_Handler(void);
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
/* Private defines -----------------------------------------------------------*/
#define B1_Pin GPIO_PIN_13
#define B1_GPIO_Port GPIOC
#define USART_TX_Pin GPIO_PIN_2
#define USART_TX_GPIO_Port GPIOA
#define USART_RX_Pin GPIO_PIN_3
#define USART_RX_GPIO_Port GPIOA
#define LD2_Pin GPIO_PIN_5
#define LD2_GPIO_Port GPIOA
#define TMS_Pin GPIO_PIN_13
#define TMS_GPIO_Port GPIOA
#define TCK_Pin GPIO_PIN_14
#define TCK_GPIO_Port GPIOA
#define SWO_Pin GPIO_PIN_3
#define SWO_GPIO_Port GPIOB
/* USER CODE BEGIN Private defines */
/* USER CODE END Private defines */
#ifdef __cplusplus
```

```
}
#endif
#endif /* __MAIN_H */
```

21.6 CODE OF STM32F4XX_HAL_CONF.H:

```
/* USER CODE BEGIN Header */
/**
*******************************************************************
  * @file    stm32f4xx_hal_conf_template.h
  * @author  MCD Application Team
  * @brief   HAL configuration template file.
  *          This file should be copied to the application folder and renamed
  *          to stm32f4xx_hal_conf.h.

*******************************************************************
  * @attention
  *
  * Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
```

```
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *

**********************************************************
*******************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_HAL_CONF_H
#define __STM32F4xx_HAL_CONF_H
#ifdef __cplusplus
 extern "C" {
#endif
/* Exported types ------------------------------------------------------------*/
/* Exported constants --------------------------------------------------------*/
/* ############################ Module Selection
################################## */
/**
  * @brief This is the list of modules to be used in the HAL driver
  */
#define HAL_MODULE_ENABLED
  /* #define HAL_CRYP_MODULE_ENABLED */
/* #define HAL_ADC_MODULE_ENABLED */
/* #define HAL_CAN_MODULE_ENABLED */
```

```
/* #define HAL_CRC_MODULE_ENABLED */
/* #define HAL_CAN_LEGACY_MODULE_ENABLED */
/* #define HAL_DAC_MODULE_ENABLED */
/* #define HAL_DCMI_MODULE_ENABLED */
/* #define HAL_DMA2D_MODULE_ENABLED */
/* #define HAL_ETH_MODULE_ENABLED */
/* #define HAL_ETH_LEGACY_MODULE_ENABLED */
/* #define HAL_NAND_MODULE_ENABLED */
/* #define HAL_NOR_MODULE_ENABLED */
/* #define HAL_PCCARD_MODULE_ENABLED */
/* #define HAL_SRAM_MODULE_ENABLED */
/* #define HAL_SDRAM_MODULE_ENABLED */
/* #define HAL_HASH_MODULE_ENABLED */
/* #define HAL_I2C_MODULE_ENABLED */
/* #define HAL_I2S_MODULE_ENABLED */
/* #define HAL_IWDG_MODULE_ENABLED */
/* #define HAL_LTDC_MODULE_ENABLED */
/* #define HAL_RNG_MODULE_ENABLED */
/* #define HAL_RTC_MODULE_ENABLED */
/* #define HAL_SAI_MODULE_ENABLED */
/* #define HAL_SD_MODULE_ENABLED */
/* #define HAL_MMC_MODULE_ENABLED */
/* #define HAL_SPI_MODULE_ENABLED */
```

```
/* #define HAL_TIM_MODULE_ENABLED */
#define HAL_UART_MODULE_ENABLED
/* #define HAL_USART_MODULE_ENABLED */
/* #define HAL_IRDA_MODULE_ENABLED */
/* #define HAL_SMARTCARD_MODULE_ENABLED */
/* #define HAL_SMBUS_MODULE_ENABLED */
/* #define HAL_WWDG_MODULE_ENABLED */
/* #define HAL_PCD_MODULE_ENABLED */
/* #define HAL_HCD_MODULE_ENABLED */
/* #define HAL_DSI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_CEC_MODULE_ENABLED */
/* #define HAL_FMPI2C_MODULE_ENABLED */
/* #define HAL_FMPSMBUS_MODULE_ENABLED */
/* #define HAL_SPDIFRX_MODULE_ENABLED */
/* #define HAL_DFSDM_MODULE_ENABLED */
/* #define HAL_LPTIM_MODULE_ENABLED */
#define HAL_GPIO_MODULE_ENABLED
#define HAL_EXTI_MODULE_ENABLED
#define HAL_DMA_MODULE_ENABLED
#define HAL_RCC_MODULE_ENABLED
#define HAL_FLASH_MODULE_ENABLED
```

```
#define HAL_PWR_MODULE_ENABLED
#define HAL_CORTEX_MODULE_ENABLED
/* ############################ HSE/HSI Values adaptation
###################### */
/**
  * @brief Adjust the value of External High Speed oscillator (HSE)
used in your application.
  *        This value is used by the RCC HAL module to compute the
system frequency
  *        (when HSE is used as system clock source, directly or through
the PLL).
  */
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    8000000U /*!< Value of the External
oscillator in Hz */
#endif /* HSE_VALUE */
#if !defined  (HSE_STARTUP_TIMEOUT)
  #define HSE_STARTUP_TIMEOUT    100U   /*!< Time out for
HSE start up, in ms */
#endif /* HSE_STARTUP_TIMEOUT */

/**
  * @brief Internal High Speed oscillator (HSI) value.
  *        This value is used by the RCC HAL module to compute the
system frequency
```

```
  *       (when HSI is used as system clock source, directly or through the
PLL).
  */
#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000U) /*!< Value of the
Internal oscillator in Hz*/
#endif /* HSI_VALUE */
/**
  * @brief Internal Low Speed oscillator (LSI) value.
  */
#if !defined  (LSI_VALUE)
 #define LSI_VALUE  32000U       /*!< LSI Typical Value in Hz*/
#endif /* LSI_VALUE */                    /*!< Value of the Internal Low
Speed oscillator in Hz
                                  The real value may vary depending on the
variations
                                  in voltage and temperature.*/
/**
  * @brief External Low Speed oscillator (LSE) value.
  */
#if !defined  (LSE_VALUE)
 #define LSE_VALUE  32768U    /*!< Value of the External Low Speed
oscillator in Hz */
#endif /* LSE_VALUE */
#if !defined  (LSE_STARTUP_TIMEOUT)
```

```
  #define LSE_STARTUP_TIMEOUT    5000U   /*!< Time out for
LSE start up, in ms */

#endif /* LSE_STARTUP_TIMEOUT */

/**

  * @brief External clock source for I2S peripheral

  *        This value is used by the I2S HAL module to compute the I2S
clock source

  *        frequency, this source is inserted directly through I2S_CKIN
pad.

  */

#if !defined  (EXTERNAL_CLOCK_VALUE)

  #define EXTERNAL_CLOCK_VALUE    12288000U /*!< Value of
the External audio frequency in Hz*/

#endif /* EXTERNAL_CLOCK_VALUE */

/* Tip: To avoid modifying this file each time you need to use different
HSE,

   ===  you can define the HSE value in your toolchain compiler
preprocessor. */

/* ############################# System Configuration
########################### */

/**

  * @brief This is the HAL system configuration section

  */

#define  VDD_VALUE                    3300U /*!< Value of VDD in mv */
```

```
#define  TICK_INT_PRIORITY          0U   /*!< tick interrupt
priority */

#define  USE_RTOS                   0U

#define  PREFETCH_ENABLE            1U

#define  INSTRUCTION_CACHE_ENABLE     1U

#define  DATA_CACHE_ENABLE          1U

#define  USE_HAL_ADC_REGISTER_CALLBACKS        0U /*
ADC register callback disabled       */

#define  USE_HAL_CAN_REGISTER_CALLBACKS        0U /*
CAN register callback disabled       */

#define  USE_HAL_CEC_REGISTER_CALLBACKS        0U /*
CEC register callback disabled       */

#define  USE_HAL_CRYP_REGISTER_CALLBACKS       0U /*
CRYP register callback disabled      */

#define  USE_HAL_DAC_REGISTER_CALLBACKS        0U /*
DAC register callback disabled       */

#define  USE_HAL_DCMI_REGISTER_CALLBACKS       0U /*
DCMI register callback disabled      */

#define  USE_HAL_DFSDM_REGISTER_CALLBACKS      0U /*
DFSDM register callback disabled     */

#define  USE_HAL_DMA2D_REGISTER_CALLBACKS      0U /*
DMA2D register callback disabled     */

#define  USE_HAL_DSI_REGISTER_CALLBACKS        0U /* DSI
register callback disabled       */

#define  USE_HAL_ETH_REGISTER_CALLBACKS        0U /*
ETH register callback disabled       */
```

#define USE_HAL_HASH_REGISTER_CALLBACKS 0U /* HASH register callback disabled */

#define USE_HAL_HCD_REGISTER_CALLBACKS 0U /* HCD register callback disabled */

#define USE_HAL_I2C_REGISTER_CALLBACKS 0U /* I2C register callback disabled */

#define USE_HAL_FMPI2C_REGISTER_CALLBACKS 0U /* FMPI2C register callback disabled */

#define USE_HAL_FMPSMBUS_REGISTER_CALLBACKS 0U /* FMPSMBUS register callback disabled */

#define USE_HAL_I2S_REGISTER_CALLBACKS 0U /* I2S register callback disabled */

#define USE_HAL_IRDA_REGISTER_CALLBACKS 0U /* IRDA register callback disabled */

#define USE_HAL_LPTIM_REGISTER_CALLBACKS 0U /* LPTIM register callback disabled */

#define USE_HAL_LTDC_REGISTER_CALLBACKS 0U /* LTDC register callback disabled */

#define USE_HAL_MMC_REGISTER_CALLBACKS 0U /* MMC register callback disabled */

#define USE_HAL_NAND_REGISTER_CALLBACKS 0U /* NAND register callback disabled */

#define USE_HAL_NOR_REGISTER_CALLBACKS 0U /* NOR register callback disabled */

#define USE_HAL_PCCARD_REGISTER_CALLBACKS 0U /* PCCARD register callback disabled */

#define USE_HAL_PCD_REGISTER_CALLBACKS 0U /* PCD register callback disabled */

#define USE_HAL_QSPI_REGISTER_CALLBACKS 0U /* QSPI register callback disabled */

#define USE_HAL_RNG_REGISTER_CALLBACKS 0U /* RNG register callback disabled */

#define USE_HAL_RTC_REGISTER_CALLBACKS 0U /* RTC register callback disabled */

#define USE_HAL_SAI_REGISTER_CALLBACKS 0U /* SAI register callback disabled */

#define USE_HAL_SD_REGISTER_CALLBACKS 0U /* SD register callback disabled */

#define USE_HAL_SMARTCARD_REGISTER_CALLBACKS 0U /* SMARTCARD register callback disabled */

#define USE_HAL_SDRAM_REGISTER_CALLBACKS 0U /* SDRAM register callback disabled */

#define USE_HAL_SRAM_REGISTER_CALLBACKS 0U /* SRAM register callback disabled */

#define USE_HAL_SPDIFRX_REGISTER_CALLBACKS 0U /* SPDIFRX register callback disabled */

#define USE_HAL_SMBUS_REGISTER_CALLBACKS 0U /* SMBUS register callback disabled */

#define USE_HAL_SPI_REGISTER_CALLBACKS 0U /* SPI register callback disabled */

#define USE_HAL_TIM_REGISTER_CALLBACKS 0U /* TIM register callback disabled */

#define USE_HAL_UART_REGISTER_CALLBACKS 0U /* UART register callback disabled */

#define USE_HAL_USART_REGISTER_CALLBACKS 0U /* USART register callback disabled */

#define USE_HAL_WWDG_REGISTER_CALLBACKS 0U /* WWDG register callback disabled */

/* ############################ Assert Selection ################################## */

/**

* @brief Uncomment the line below to expanse the "assert_param" macro in the

* HAL drivers code

*/

/* #define USE_FULL_ASSERT 1U */

/* #################### Ethernet peripheral configuration ######################## */

/* Section 1 : Ethernet peripheral configuration */

/* MAC ADDRESS: MAC_ADDR0:MAC_ADDR1:MAC_ADDR2:MAC_ADDR3:MAC_ADDR4:MAC_ADDR5 */

#define MAC_ADDR0 2U

#define MAC_ADDR1 0U

#define MAC_ADDR2 0U

#define MAC_ADDR3 0U

#define MAC_ADDR4 0U

```
#define MAC_ADDR5   0U

/* Definition of the Ethernet driver buffers size and count */

#define ETH_RX_BUF_SIZE              ETH_MAX_PACKET_SIZE
/* buffer size for receive            */

#define ETH_TX_BUF_SIZE              ETH_MAX_PACKET_SIZE
/* buffer size for transmit           */

#define ETH_RXBUFNB                  4U      /* 4 Rx buffers of size
ETH_RX_BUF_SIZE  */

#define ETH_TXBUFNB                  4U      /* 4 Tx buffers of size
ETH_TX_BUF_SIZE  */

/* Section 2: PHY configuration section */

/* DP83848_PHY_ADDRESS Address*/

#define DP83848_PHY_ADDRESS

/* PHY Reset delay these values are based on a 1 ms Systick interrupt*/

#define PHY_RESET_DELAY              0x000000FFU

/* PHY Configuration delay */

#define PHY_CONFIG_DELAY             0x00000FFFU

#define PHY_READ_TO                  0x0000FFFFU

#define PHY_WRITE_TO                 0x0000FFFFU

/* Section 3: Common PHY Registers */

#define PHY_BCR                      ((uint16_t)0x0000U)    /*!<
Transceiver Basic Control Register   */

#define PHY_BSR                      ((uint16_t)0x0001U)    /*!<
Transceiver Basic Status Register    */
```

```
#define PHY_RESET                    ((uint16_t)0x8000U)  /*!< PHY
Reset */

#define PHY_LOOPBACK                 ((uint16_t)0x4000U)  /*!<
Select loop-back mode */

#define PHY_FULLDUPLEX_100M          ((uint16_t)0x2100U)
/*!< Set the full-duplex mode at 100 Mb/s */

#define PHY_HALFDUPLEX_100M          ((uint16_t)0x2000U)
/*!< Set the half-duplex mode at 100 Mb/s */

#define PHY_FULLDUPLEX_10M           ((uint16_t)0x0100U)
/*!< Set the full-duplex mode at 10 Mb/s  */

#define PHY_HALFDUPLEX_10M           ((uint16_t)0x0000U)
/*!< Set the half-duplex mode at 10 Mb/s  */

#define PHY_AUTONEGOTIATION          ((uint16_t)0x1000U)
/*!< Enable auto-negotiation function     */

#define PHY_RESTART_AUTONEGOTIATION
((uint16_t)0x0200U)  /*!< Restart auto-negotiation function    */

#define PHY_POWERDOWN                ((uint16_t)0x0800U)  /*!<
Select the power down mode           */

#define PHY_ISOLATE                  ((uint16_t)0x0400U)  /*!< Isolate
PHY from MII                 */

#define PHY_AUTONEGO_COMPLETE        ((uint16_t)0x0020U)
/*!< Auto-Negotiation process completed   */

#define PHY_LINKED_STATUS            ((uint16_t)0x0004U)  /*!<
Valid link established               */

#define PHY_JABBER_DETECTION         ((uint16_t)0x0002U)
/*!< Jabber condition detected            */

/* Section 4: Extended PHY Registers */
```

```
#define PHY_SR                    ((uint16_t))    /*!< PHY status register
Offset                  */

#define PHY_SPEED_STATUS              ((uint16_t))  /*!< PHY Speed
mask                          */

#define PHY_DUPLEX_STATUS             ((uint16_t))  /*!< PHY
Duplex mask                           */

/* ################### SPI peripheral configuration
############################ */

/* CRC FEATURE: Use to activate CRC feature inside HAL SPI Driver

* Activated: CRC code is present inside driver

* Deactivated: CRC code cleaned from driver

*/

#define USE_SPI_CRC                   0U

/* Includes ------------------------------------------------------------------*/

/**

  * @brief Include module's header file

  */

#ifdef HAL_RCC_MODULE_ENABLED

  #include "stm32f4xx_hal_rcc.h"

#endif /* HAL_RCC_MODULE_ENABLED */

#ifdef HAL_GPIO_MODULE_ENABLED

  #include "stm32f4xx_hal_gpio.h"

#endif /* HAL_GPIO_MODULE_ENABLED */

#ifdef HAL_EXTI_MODULE_ENABLED
```

```
  #include "stm32f4xx_hal_exti.h"
#endif /* HAL_EXTI_MODULE_ENABLED */
#ifdef HAL_DMA_MODULE_ENABLED
  #include "stm32f4xx_hal_dma.h"
#endif /* HAL_DMA_MODULE_ENABLED */
#ifdef HAL_CORTEX_MODULE_ENABLED
  #include "stm32f4xx_hal_cortex.h"
#endif /* HAL_CORTEX_MODULE_ENABLED */
#ifdef HAL_ADC_MODULE_ENABLED
  #include "stm32f4xx_hal_adc.h"
#endif /* HAL_ADC_MODULE_ENABLED */
#ifdef HAL_CAN_MODULE_ENABLED
  #include "stm32f4xx_hal_can.h"
#endif /* HAL_CAN_MODULE_ENABLED */
#ifdef HAL_CAN_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_can_legacy.h"
#endif /* HAL_CAN_LEGACY_MODULE_ENABLED */
#ifdef HAL_CRC_MODULE_ENABLED
  #include "stm32f4xx_hal_crc.h"
#endif /* HAL_CRC_MODULE_ENABLED */
#ifdef HAL_CRYP_MODULE_ENABLED
  #include "stm32f4xx_hal_cryp.h"
#endif /* HAL_CRYP_MODULE_ENABLED */
```

```
#ifdef HAL_DMA2D_MODULE_ENABLED
  #include "stm32f4xx_hal_dma2d.h"
#endif /* HAL_DMA2D_MODULE_ENABLED */
#ifdef HAL_DAC_MODULE_ENABLED
  #include "stm32f4xx_hal_dac.h"
#endif /* HAL_DAC_MODULE_ENABLED */
#ifdef HAL_DCMI_MODULE_ENABLED
  #include "stm32f4xx_hal_dcmi.h"
#endif /* HAL_DCMI_MODULE_ENABLED */
#ifdef HAL_ETH_MODULE_ENABLED
  #include "stm32f4xx_hal_eth.h"
#endif /* HAL_ETH_MODULE_ENABLED */
#ifdef HAL_ETH_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_eth_legacy.h"
#endif /* HAL_ETH_LEGACY_MODULE_ENABLED */
#ifdef HAL_FLASH_MODULE_ENABLED
  #include "stm32f4xx_hal_flash.h"
#endif /* HAL_FLASH_MODULE_ENABLED */
#ifdef HAL_SRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sram.h"
#endif /* HAL_SRAM_MODULE_ENABLED */
#ifdef HAL_NOR_MODULE_ENABLED
  #include "stm32f4xx_hal_nor.h"
```

```
#endif /* HAL_NOR_MODULE_ENABLED */
#ifdef HAL_NAND_MODULE_ENABLED
  #include "stm32f4xx_hal_nand.h"
#endif /* HAL_NAND_MODULE_ENABLED */
#ifdef HAL_PCCARD_MODULE_ENABLED
  #include "stm32f4xx_hal_pccard.h"
#endif /* HAL_PCCARD_MODULE_ENABLED */
#ifdef HAL_SDRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sdram.h"
#endif /* HAL_SDRAM_MODULE_ENABLED */
#ifdef HAL_HASH_MODULE_ENABLED
 #include "stm32f4xx_hal_hash.h"
#endif /* HAL_HASH_MODULE_ENABLED */
#ifdef HAL_I2C_MODULE_ENABLED
 #include "stm32f4xx_hal_i2c.h"
#endif /* HAL_I2C_MODULE_ENABLED */
#ifdef HAL_SMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_smbus.h"
#endif /* HAL_SMBUS_MODULE_ENABLED */
#ifdef HAL_I2S_MODULE_ENABLED
 #include "stm32f4xx_hal_i2s.h"
#endif /* HAL_I2S_MODULE_ENABLED */
#ifdef HAL_IWDG_MODULE_ENABLED
```

```
 #include "stm32f4xx_hal_iwdg.h"
#endif /* HAL_IWDG_MODULE_ENABLED */
#ifdef HAL_LTDC_MODULE_ENABLED
 #include "stm32f4xx_hal_ltdc.h"
#endif /* HAL_LTDC_MODULE_ENABLED */
#ifdef HAL_PWR_MODULE_ENABLED
 #include "stm32f4xx_hal_pwr.h"
#endif /* HAL_PWR_MODULE_ENABLED */
#ifdef HAL_RNG_MODULE_ENABLED
 #include "stm32f4xx_hal_rng.h"
#endif /* HAL_RNG_MODULE_ENABLED */
#ifdef HAL_RTC_MODULE_ENABLED
 #include "stm32f4xx_hal_rtc.h"
#endif /* HAL_RTC_MODULE_ENABLED */
#ifdef HAL_SAI_MODULE_ENABLED
 #include "stm32f4xx_hal_sai.h"
#endif /* HAL_SAI_MODULE_ENABLED */
#ifdef HAL_SD_MODULE_ENABLED
 #include "stm32f4xx_hal_sd.h"
#endif /* HAL_SD_MODULE_ENABLED */
#ifdef HAL_SPI_MODULE_ENABLED
 #include "stm32f4xx_hal_spi.h"
#endif /* HAL_SPI_MODULE_ENABLED */
```

```
#ifdef HAL_TIM_MODULE_ENABLED
 #include "stm32f4xx_hal_tim.h"
#endif /* HAL_TIM_MODULE_ENABLED */
#ifdef HAL_UART_MODULE_ENABLED
 #include "stm32f4xx_hal_uart.h"
#endif /* HAL_UART_MODULE_ENABLED */
#ifdef HAL_USART_MODULE_ENABLED
 #include "stm32f4xx_hal_usart.h"
#endif /* HAL_USART_MODULE_ENABLED */
#ifdef HAL_IRDA_MODULE_ENABLED
 #include "stm32f4xx_hal_irda.h"
#endif /* HAL_IRDA_MODULE_ENABLED */
#ifdef HAL_SMARTCARD_MODULE_ENABLED
 #include "stm32f4xx_hal_smartcard.h"
#endif /* HAL_SMARTCARD_MODULE_ENABLED */
#ifdef HAL_WWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_wwdg.h"
#endif /* HAL_WWDG_MODULE_ENABLED */
#ifdef HAL_PCD_MODULE_ENABLED
 #include "stm32f4xx_hal_pcd.h"
#endif /* HAL_PCD_MODULE_ENABLED */
#ifdef HAL_HCD_MODULE_ENABLED
 #include "stm32f4xx_hal_hcd.h"
```

```
#endif /* HAL_HCD_MODULE_ENABLED */
#ifdef HAL_DSI_MODULE_ENABLED
 #include "stm32f4xx_hal_dsi.h"
#endif /* HAL_DSI_MODULE_ENABLED */
#ifdef HAL_QSPI_MODULE_ENABLED
 #include "stm32f4xx_hal_qspi.h"
#endif /* HAL_QSPI_MODULE_ENABLED */
#ifdef HAL_CEC_MODULE_ENABLED
 #include "stm32f4xx_hal_cec.h"
#endif /* HAL_CEC_MODULE_ENABLED */
#ifdef HAL_FMPI2C_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpi2c.h"
#endif /* HAL_FMPI2C_MODULE_ENABLED */
#ifdef HAL_FMPSMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpsmbus.h"
#endif /* HAL_FMPSMBUS_MODULE_ENABLED */
#ifdef HAL_SPDIFRX_MODULE_ENABLED
 #include "stm32f4xx_hal_spdifrx.h"
#endif /* HAL_SPDIFRX_MODULE_ENABLED */
#ifdef HAL_DFSDM_MODULE_ENABLED
 #include "stm32f4xx_hal_dfsdm.h"
#endif /* HAL_DFSDM_MODULE_ENABLED */
#ifdef HAL_LPTIM_MODULE_ENABLED
```

```
 #include "stm32f4xx_hal_lptim.h"
#endif /* HAL_LPTIM_MODULE_ENABLED */
#ifdef HAL_MMC_MODULE_ENABLED
 #include "stm32f4xx_hal_mmc.h"
#endif /* HAL_MMC_MODULE_ENABLED */
/* Exported macro ------------------------------------------------------------*/
#ifdef  USE_FULL_ASSERT
/**
  * @brief  The assert_param macro is used for function's parameters check.
  * @param  expr If expr is false, it calls assert_failed function
  *         which reports the name of the source file and the source
  *         line number of the call that failed.
  *         If expr is true, it returns no value.
  * @retval None
  */
  #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t *)__FILE__, __LINE__))
/* Exported functions ------------------------------------------------------- */
  void assert_failed(uint8_t* file, uint32_t line);
#else
  #define assert_param(expr) ((void)0U)
#endif /* USE_FULL_ASSERT */
#ifdef __cplusplus
```

```
}
#endif
#endif /* __STM32F4xx_HAL_CONF_H */
```

21.7 CODE OF STM32F4XX_IT.H:

```
/* USER CODE BEGIN Header */
/**
**********************************************************
********************
  * @file    stm32f4xx_it.h
  * @brief   This file contains the headers of the interrupt handlers.
**********************************************************
********************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *

**********************************************************
********************
```

```
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_IT_H
#define __STM32F4xx_IT_H
#ifdef __cplusplus
 extern "C" {
#endif
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */
/* USER CODE END EC */
/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */
/* USER CODE END EM */
/* Exported functions prototypes ---------------------------------------------*/
void NMI_Handler(void);
void HardFault_Handler(void);
```

```
void MemManage_Handler(void);
void BusFault_Handler(void);
void UsageFault_Handler(void);
void SVC_Handler(void);
void DebugMon_Handler(void);
void PendSV_Handler(void);
void SysTick_Handler(void);
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
#ifdef __cplusplus
}
#endif
#endif /* __STM32F4xx_IT_H */
```

21.8 CODE OF MAIN.C:

```
/* USER CODE BEGIN Header */
/**
*****************************************************************
*******************
  * @file           : main.c
  * @brief          : Main program body
*****************************************************************
*******************
  * @attention
  *
```

```
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
/* USER CODE END PTD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
```

```
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
UART_HandleTypeDef huart2;
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */
  /* USER CODE END 1 */
```

```
  /* MCU Configuration----------------------------------------------------------*/
  /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
  HAL_Init();
  /* USER CODE BEGIN Init */
  /* USER CODE END Init */
  /* Configure the system clock */
  SystemClock_Config();
  /* USER CODE BEGIN SysInit */
  /* USER CODE END SysInit */
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  /* USER CODE BEGIN 2 */
  /* USER CODE END 2 */
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
```

```
}
/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();

__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_
VOLTAGE_SCALE1);
  /** Initializes the RCC Oscillators according to the specified
parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue =
RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
```

```
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 16;
  RCC_OscInitStruct.PLL.PLLN = 336;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV4;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK

|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource =
RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct,
FLASH_LATENCY_2) != HAL_OK)
  {
    Error_Handler();
```

```
  }
}
/**
  * @brief USART2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART2_UART_Init(void)
{
  /* USER CODE BEGIN USART2_Init 0 */
  /* USER CODE END USART2_Init 0 */
  /* USER CODE BEGIN USART2_Init 1 */
  /* USER CODE END USART2_Init 1 */
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 115200;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
```

```
    Error_Handler();
  }
  /* USER CODE BEGIN USART2_Init 2 */
  /* USER CODE END USART2_Init 2 */
}
/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */
  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin,
GPIO_PIN_RESET);
```

```
  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);
  /*Configure GPIO pin : LD2_Pin */
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);
/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}
/* USER CODE BEGIN 4 */
/* USER CODE END 4 */
/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
```

```
  /* User can add his own implementation to report the HAL error return
state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}
#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line
number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line
number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file,
line) */
```

```
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

21.9 CODE OF MAIN.S:

```
    .section .text
    .global _start
_start:
    /* Set up the stack pointer */
    LDR     R0, =_stack_top
    MOV     SP, R0
    /* Enable GPIO clock */
    LDR     R0, =RCC_AHB1ENR
    LDR     R1, =0x00000001        /* Enable clock for GPIOA (bit 0) */
    STR     R1, [R0]
    /* Set GPIO pin mode (e.g., output) */
    LDR     R0, =GPIOA_MODER
    LDR     R1, [R0]
    BIC     R1, R1, #(0x3 << (2 * 5))  /* Clear mode for pin PA5 */
    ORR     R1, R1, #(0x1 << (2 * 5))  /* Set mode to output for PA5 */
    STR     R1, [R0]
    /* Turn on LED connected to PA5 */
    LDR     R0, =GPIOA_ODR
    LDR     R1, [R0]
```

```
    ORR     R1, R1, #(0x1 << 5)    /* Set PA5 high */
    STR     R1, [R0]
loop:
    B loop
```

21.10 CODE OF STM32F4XX_HAL_MSP.C:

```
/* USER CODE BEGIN Header */
/**
*******************************************************
*******************
  * @file         stm32f4xx_hal_msp.c
  * @brief        This file provides code for the MSP Initialization
  *               and de-Initialization codes.
*******************************************************
*******************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
```

```
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN TD */
/* USER CODE END TD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN Define */
/* USER CODE END Define */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN Macro */
/* USER CODE END Macro */
/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
```

```
/* External functions ------------------------------------------------------*/
/* USER CODE BEGIN ExternalFunctions */
/* USER CODE END ExternalFunctions */
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/**
  * Initializes the Global MSP.
  */
void HAL_MspInit(void)
{
  /* USER CODE BEGIN MspInit 0 */
  /* USER CODE END MspInit 0 */
  __HAL_RCC_SYSCFG_CLK_ENABLE();
  __HAL_RCC_PWR_CLK_ENABLE();
  HAL_NVIC_SetPriorityGrouping(NVIC_PRIORITYGROUP_0);
  /* System interrupt init*/
  /* USER CODE BEGIN MspInit 1 */
  /* USER CODE END MspInit 1 */
}
/**
* @brief UART MSP Initialization
* This function configures the hardware resources used in this example
* @param huart: UART handle pointer
```

```
* @retval None
*/
void HAL_UART_MspInit(UART_HandleTypeDef* huart)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  if(huart->Instance==USART2)
  {
  /* USER CODE BEGIN USART2_MspInit 0 */
  /* USER CODE END USART2_MspInit 0 */
    /* Peripheral clock enable */
    __HAL_RCC_USART2_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();
    /**USART2 GPIO Configuration
    PA2     ------> USART2_TX
    PA3     ------> USART2_RX
    */
    GPIO_InitStruct.Pin = USART_TX_Pin|USART_RX_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF7_USART2;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
  /* USER CODE BEGIN USART2_MspInit 1 */
```

```
  /* USER CODE END USART2_MspInit 1 */
  }
}
/**
* @brief UART MSP De-Initialization
* This function freeze the hardware resources used in this example
* @param huart: UART handle pointer
* @retval None
*/
void HAL_UART_MspDeInit(UART_HandleTypeDef* huart)
{
  if(huart->Instance==USART2)
  {
  /* USER CODE BEGIN USART2_MspDeInit 0 */
  /* USER CODE END USART2_MspDeInit 0 */
    /* Peripheral clock disable */
    __HAL_RCC_USART2_CLK_DISABLE();
    /**USART2 GPIO Configuration
    PA2     ------> USART2_TX
    PA3     ------> USART2_RX
    */
    HAL_GPIO_DeInit(GPIOA, USART_TX_Pin|USART_RX_Pin);
  /* USER CODE BEGIN USART2_MspDeInit 1 */
```

```
  /* USER CODE END USART2_MspDeInit 1 */
  }
}
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

21.11 CODE OF STM32F4XX_IT.C:

```
/* USER CODE BEGIN Header */
/**
*********************************************************
********************
  * @file    stm32f4xx_it.c
  * @brief   Interrupt Service Routines.
*********************************************************
********************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
```

```
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stm32f4xx_it.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN TD */
/* USER CODE END TD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
```

```
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/* External variables ---------------------------------------------------------*/
/* USER CODE BEGIN EV */
/* USER CODE END EV */
/******************************************************************************/
/*            Cortex-M4 Processor Interruption and Exception Handlers          */
/******************************************************************************/
/**
  * @brief This function handles Non maskable interrupt.
  */
void NMI_Handler(void)
{
  /* USER CODE BEGIN NonMaskableInt_IRQn 0 */
  /* USER CODE END NonMaskableInt_IRQn 0 */
  /* USER CODE BEGIN NonMaskableInt_IRQn 1 */
   while (1)
   {
```

```
  }
  /* USER CODE END NonMaskableInt_IRQn 1 */
}
/**
  * @brief This function handles Hard fault interrupt.
  */
void HardFault_Handler(void)
{
  /* USER CODE BEGIN HardFault_IRQn 0 */
  /* USER CODE END HardFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_HardFault_IRQn 0 */
    /* USER CODE END W1_HardFault_IRQn 0 */
  }
}
/**
  * @brief This function handles Memory management fault.
  */
void MemManage_Handler(void)
{
  /* USER CODE BEGIN MemoryManagement_IRQn 0 */
  /* USER CODE END MemoryManagement_IRQn 0 */
```

```
  while (1)
  {
    /* USER CODE BEGIN W1_MemoryManagement_IRQn 0 */
    /* USER CODE END W1_MemoryManagement_IRQn 0 */
  }
}
/**
  * @brief This function handles Pre-fetch fault, memory access fault.
  */
void BusFault_Handler(void)
{
  /* USER CODE BEGIN BusFault_IRQn 0 */
  /* USER CODE END BusFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_BusFault_IRQn 0 */
    /* USER CODE END W1_BusFault_IRQn 0 */
  }
}
/**
  * @brief This function handles Undefined instruction or illegal state.
  */
void UsageFault_Handler(void)
```

```
{
  /* USER CODE BEGIN UsageFault_IRQn 0 */
  /* USER CODE END UsageFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_UsageFault_IRQn 0 */
    /* USER CODE END W1_UsageFault_IRQn 0 */
  }
}
/**
  * @brief This function handles System service call via SWI
instruction.
  */
void SVC_Handler(void)
{
  /* USER CODE BEGIN SVCall_IRQn 0 */
  /* USER CODE END SVCall_IRQn 0 */
  /* USER CODE BEGIN SVCall_IRQn 1 */
  /* USER CODE END SVCall_IRQn 1 */
}

/**
  * @brief This function handles Debug monitor.
```

```
  */
void DebugMon_Handler(void)
{
  /* USER CODE BEGIN DebugMonitor_IRQn 0 */
  /* USER CODE END DebugMonitor_IRQn 0 */
  /* USER CODE BEGIN DebugMonitor_IRQn 1 */
  /* USER CODE END DebugMonitor_IRQn 1 */
}
/**
  * @brief This function handles Pendable request for system service.
  */
void PendSV_Handler(void)
{
  /* USER CODE BEGIN PendSV_IRQn 0 */
  /* USER CODE END PendSV_IRQn 0 */
  /* USER CODE BEGIN PendSV_IRQn 1 */
  /* USER CODE END PendSV_IRQn 1 */
}
/**
  * @brief This function handles System tick timer.
  */
void SysTick_Handler(void)
{
```

```
  /* USER CODE BEGIN SysTick_IRQn 0 */
  /* USER CODE END SysTick_IRQn 0 */
  HAL_IncTick();
  /* USER CODE BEGIN SysTick_IRQn 1 */
  /* USER CODE END SysTick_IRQn 1 */
}
/*****************************************************************************/
/* STM32F4xx Peripheral Interrupt Handlers                                   */
/* Add here the Interrupt Handlers for the used peripherals.                 */
/* For the available peripheral interrupt handler names,                     */
/* please refer to the startup file (startup_stm32f4xx.s).                  */
/*****************************************************************************/
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

21.12 CODE OF SYSCALLS.C:

```
/**
*****************************************************************************
 * @file      syscalls.c
 * @author    Auto-generated by STM32CubeIDE
 * @brief     STM32CubeIDE Minimal System calls file
 *
```

```
 *          For more information about which c-functions
 *          need which of these lowlevel functions
 *          please consult the Newlib libc-manual
**********************************************************
********************
 * @attention
 *
 * Copyright (c) 2020-2024 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the
LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
**********************************************************
********************
 */
/* Includes */
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <signal.h>
```

```
#include <time.h>
#include <sys/time.h>
#include <sys/times.h>
/* Variables */
extern int __io_putchar(int ch) __attribute__((weak));
extern int __io_getchar(void) __attribute__((weak));
char *__env[1] = { 0 };
char **environ = __env;
/* Functions */
void initialise_monitor_handles()
{
}
int _getpid(void)
{
  return 1;
}
int _kill(int pid, int sig)
{
  (void)pid;
  (void)sig;
  errno = EINVAL;
  return -1;
}
```

```
void _exit (int status)
{
  _kill(status, -1);
  while (1) {}    /* Make sure we hang here */
}
__attribute__((weak)) int _read(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;
  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    *ptr++ = __io_getchar();
  }
  return len;
}
__attribute__((weak)) int _write(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;

  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    __io_putchar(*ptr++);
```

```
  }
  return len;
}
int _close(int file)
{
  (void)file;
  return -1;
}
int _fstat(int file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
int _isatty(int file)
{
  (void)file;
  return 1;
}
int _lseek(int file, int ptr, int dir)
{
  (void)file;
  (void)ptr;
```

```
  (void)dir;
  return 0;
}
int _open(char *path, int flags, ...)
{
  (void)path;
  (void)flags;
  /* Pretend like we always fail */
  return -1;
}
int _wait(int *status)
{
  (void)status;
  errno = ECHILD;
  return -1;
}
int _unlink(char *name)
{
  (void)name;
  errno = ENOENT;
  return -1;
}
int _times(struct tms *buf)
```

```
{
  (void)buf;
  return -1;
}
int _stat(char *file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
int _link(char *old, char *new)
{
  (void)old;
  (void)new;
  errno = EMLINK;
  return -1;
}
int _fork(void)
{
  errno = EAGAIN;
  return -1;
}
int _execve(char *name, char **argv, char **env)
```

```
{
  (void)name;
  (void)argv;
  (void)env;
  errno = ENOMEM;
  return -1;
}
```

21.13 CODE OF SYTSEM_STM32F4XX.C:

```
/**
*****************************************************************
*******************
  * @file    system_stm32f4xx.c
  * @author  MCD Application Team
  * @brief   CMSIS Cortex-M4 Device Peripheral Access Layer System
Source File.
  *
  *   This file provides two functions and one global variable to be called
from
  *   user application:
  *      - SystemInit(): This function is called at startup just after reset and
  *                      before branch to main program. This call is made inside
  *                      the "startup_stm32f4xx.s" file.
  *
```

```
  *      - SystemCoreClock variable: Contains the core clock (HCLK), it
can be used
  *                              by the user application to setup the SysTick
  *                              timer or configure other parameters.
  *
  *      - SystemCoreClockUpdate(): Updates the variable
SystemCoreClock and must
  *                             be called whenever the core clock is changed
  *                             during program execution.
  *
  *
*****************************************************************
*******************
  * @attention
  *
  * Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*****************************************************************
*******************
```

```
  */
/** @addtogroup CMSIS
  * @{
  */
/** @addtogroup stm32f4xx_system
  * @{
  */
/** @addtogroup STM32F4xx_System_Private_Includes
  * @{
  */
#include "stm32f4xx.h"
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    ((uint32_t)25000000) /*!< Default value of
the External oscillator in Hz */
#endif /* HSE_VALUE */
#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000) /*!< Value of the
Internal oscillator in Hz*/
#endif /* HSI_VALUE */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_TypesDefinitions
  * @{
```

```
  */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Defines
  * @{
  */
/************************* Miscellaneous Configuration ************************/
/*!< Uncomment the following line if you need to use external SRAM or SDRAM as data memory  */
#if defined(STM32F405xx) || defined(STM32F415xx) || defined(STM32F407xx) || defined(STM32F417xx)\
 || defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx) || defined(STM32F412Zx) || defined(STM32F412Vx)
/* #define DATA_IN_ExtSRAM */
#endif /* STM32F40xxx || STM32F41xxx || STM32F42xxx || STM32F43xxx || STM32F469xx || STM32F479xx ||\
        STM32F412Zx || STM32F412Vx */
#if defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F446xx) || defined(STM32F469xx) || defined(STM32F479xx)
```

/* #define DATA_IN_ExtSDRAM */

#endif /* STM32F427xx || STM32F437xx || STM32F429xx || STM32F439xx || STM32F446xx || STM32F469xx ||\

STM32F479xx */

/* Note: Following vector table addresses must be defined in line with linker

configuration. */

/*!< Uncomment the following line if you need to relocate the vector table

anywhere in Flash or Sram, else the vector table is kept at the automatic

remap of boot address selected */

/* #define USER_VECT_TAB_ADDRESS */

#if defined(USER_VECT_TAB_ADDRESS)

/*!< Uncomment the following line if you need to relocate your vector Table

in Sram else user remap will be done in Flash. */

/* #define VECT_TAB_SRAM */

#if defined(VECT_TAB_SRAM)

#define VECT_TAB_BASE_ADDRESS SRAM_BASE /*!< Vector Table base address field.

This value must be a multiple of 0x200. */

#define VECT_TAB_OFFSET 0x00000000U /*!< Vector Table base offset field.

```
                                        This value must be a multiple of 0x200.
*/
#else
#define VECT_TAB_BASE_ADDRESS   FLASH_BASE      /*!< Vector Table base address field.
                                        This value must be a multiple of 0x200.
*/
#define VECT_TAB_OFFSET         0x00000000U     /*!< Vector Table base offset field.
                                        This value must be a multiple of 0x200.
*/
#endif /* VECT_TAB_SRAM */
#endif /* USER_VECT_TAB_ADDRESS */
/*****************************************************************************/
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Macros
  * @{
  */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Variables
```

```
  * @{
  */
  /* This variable is updated in three ways:
     1) by calling CMSIS function SystemCoreClockUpdate()
     2) by calling HAL API function HAL_RCC_GetHCLKFreq()
     3) each time HAL_RCC_ClockConfig() is called to configure the system clock frequency
        Note: If you use this function to configure the system clock; then there
              is no need to call the 2 first functions listed above, since SystemCoreClock
              variable is updated automatically.
  */
uint32_t SystemCoreClock = 16000000;
const uint8_t AHBPrescTable[16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9};
const uint8_t APBPrescTable[8]  = {0, 0, 0, 0, 1, 2, 3, 4};
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_FunctionPrototypes
  * @{
  */
#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
```

```
  static void SystemInit_ExtMemCtl(void);
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */
/**
  * @}
  */

/** @addtogroup STM32F4xx_System_Private_Functions
  * @{
  */
/**
  * @brief  Setup the microcontroller system
  *         Initialize the FPU setting, vector table location and External memory
  *         configuration.
  * @param  None
  * @retval None
  */
void SystemInit(void)
{
  /* FPU settings ------------------------------------------------------------*/
  #if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
    SCB->CPACR |= ((3UL << 10*2)|(3UL << 11*2));  /* set CP10 and CP11 Full Access */
  #endif
```

```
#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  SystemInit_ExtMemCtl();
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */
  /* Configure the Vector Table location -----------------------------------*/
#if defined(USER_VECT_TAB_ADDRESS)
  SCB->VTOR = VECT_TAB_BASE_ADDRESS |
VECT_TAB_OFFSET; /* Vector Table Relocation in Internal SRAM */
#endif /* USER_VECT_TAB_ADDRESS */
}

/**
   * @brief  Update SystemCoreClock variable according to Clock
Register Values.
  *         The SystemCoreClock variable contains the core clock (HCLK),
it can
  *         be used by the user application to setup the SysTick timer or
configure
  *         other parameters.
  *
  * @note   Each time the core clock (HCLK) changes, this function
must be called
  *         to update SystemCoreClock variable value. Otherwise, any
configuration
  *         based on this variable will be incorrect.
  *
```

```
  * @note   - The system frequency computed by this function is not the
real
  *          frequency in the chip. It is calculated based on the predefined
  *          constant and the selected clock source:
  *
  *          - If SYSCLK source is HSI, SystemCoreClock will contain the
HSI_VALUE(*)
  *
  *          - If SYSCLK source is HSE, SystemCoreClock will contain the
HSE_VALUE(**)
  *
  *          - If SYSCLK source is PLL, SystemCoreClock will contain the
HSE_VALUE(**)
  *            or HSI_VALUE(*) multiplied/divided by the PLL factors.
  *
  *        (*) HSI_VALUE is a constant defined in stm32f4xx_hal_conf.h
file (default value
  *            16 MHz) but the real value may vary depending on the
variations
  *            in voltage and temperature.
  *
  *        (**) HSE_VALUE is a constant defined in
stm32f4xx_hal_conf.h file (its value
  *             depends on the application requirements), user has to ensure
that HSE_VALUE
```

```
  *             is same as the real frequency of the crystal used. Otherwise, this function
  *             may have wrong result.
  *
  *         - The result of this function could be not correct when using fractional
  *           value for HSE crystal.
  *
  * @param  None
  * @retval None
  */
void SystemCoreClockUpdate(void)
{
  uint32_t tmp = 0, pllvco = 0, pllp = 2, pllsource = 0, pllm = 2;
  /* Get SYSCLK source -------------------------------------------------------*/
  tmp = RCC->CFGR & RCC_CFGR_SWS;
  switch (tmp)
  {
    case 0x00:  /* HSI used as system clock source */
      SystemCoreClock = HSI_VALUE;
      break;
    case 0x04:  /* HSE used as system clock source */
      SystemCoreClock = HSE_VALUE;
```

```
    break;
   case 0x08:  /* PLL used as system clock source */
    /* PLL_VCO = (HSE_VALUE or HSI_VALUE / PLL_M) * PLL_N
       SYSCLK = PLL_VCO / PLL_P
       */
    pllsource = (RCC->PLLCFGR & RCC_PLLCFGR_PLLSRC) >>
22;
    pllm = RCC->PLLCFGR & RCC_PLLCFGR_PLLM;
    if (pllsource != 0)
    {
      /* HSE used as PLL clock source */
      pllvco = (HSE_VALUE / pllm) * ((RCC->PLLCFGR &
RCC_PLLCFGR_PLLN) >> 6);
    }
    else
    {
      /* HSI used as PLL clock source */
      pllvco = (HSI_VALUE / pllm) * ((RCC->PLLCFGR &
RCC_PLLCFGR_PLLN) >> 6);
    }
    pllp = (((RCC->PLLCFGR & RCC_PLLCFGR_PLLP) >>16) + 1 )
*2;
    SystemCoreClock = pllvco/pllp;
    break;
```

```
    default:
      SystemCoreClock = HSI_VALUE;
      break;
  }
  /* Compute HCLK frequency ----------------------------------------------
-*/
  /* Get HCLK prescaler */
  tmp = AHBPrescTable[((RCC->CFGR & RCC_CFGR_HPRE) >> 4)];
  /* HCLK frequency */
  SystemCoreClock >>= tmp;
}
#if defined (DATA_IN_ExtSRAM) && defined
(DATA_IN_ExtSDRAM)
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
/**
  * @brief  Setup the external memory controller.
  *         Called in startup_stm32f4xx.s before jump to main.
  *         This function configures the external memories
(SRAM/SDRAM)
  *         This SRAM/SDRAM will be used as program data memory
(including heap and stack).
  * @param  None
  * @retval None
```

```
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
  register __IO uint32_t index;
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and
GPIOI interface clock */
  RCC->AHB1ENR |= 0x000001F8;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x00CCC0CC;
  GPIOD->AFR[1]  = 0xCCCCCCCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xAAAA0A8A;
  /* Configure PDx pins speed to 100 MHz */
  GPIOD->OSPEEDR = 0xFFFF0FCF;
  /* Configure PDx pins Output type to push-pull */
  GPIOD->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOD->PUPDR   = 0x00000000;
  /* Connect PEx pins to FMC Alternate function */
```

```
GPIOE->AFR[0]  = 0xC00CC0CC;
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA828A;
/* Configure PEx pins speed to 100 MHz */
GPIOE->OSPEEDR = 0xFFFFC3CF;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0xCCCCCCCC;
GPIOF->AFR[1]  = 0xCCCCCCCC;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA800AAA;
/* Configure PFx pins speed to 50 MHz */
GPIOF->OSPEEDR = 0xAA800AAA;
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
GPIOF->PUPDR   = 0x00000000;
/* Connect PGx pins to FMC Alternate function */
GPIOG->AFR[0]  = 0xCCCCCCCC;
```

```
GPIOG->AFR[1]  = 0xCCCCCCCC;
/* Configure PGx pins in Alternate function mode */
GPIOG->MODER   = 0xAAAAAAAA;
/* Configure PGx pins speed to 50 MHz */
GPIOG->OSPEEDR = 0xAAAAAAAA;
/* Configure PGx pins Output type to push-pull */
GPIOG->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PGx pins */
GPIOG->PUPDR   = 0x00000000;
/* Connect PHx pins to FMC Alternate function */
GPIOH->AFR[0]  = 0x00C0CC00;
GPIOH->AFR[1]  = 0xCCCCCCCC;
/* Configure PHx pins in Alternate function mode */
GPIOH->MODER   = 0xAAAA08A0;
/* Configure PHx pins speed to 50 MHz */
GPIOH->OSPEEDR = 0xAAAA08A0;
/* Configure PHx pins Output type to push-pull */
GPIOH->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PHx pins */
GPIOH->PUPDR   = 0x00000000;
/* Connect PIx pins to FMC Alternate function */
GPIOI->AFR[0]  = 0xCCCCCCCC;
GPIOI->AFR[1]  = 0x00000CC0;
```

```
  /* Configure PIx pins in Alternate function mode */
  GPIOI->MODER   = 0x0028AAAA;
  /* Configure PIx pins speed to 50 MHz */
  GPIOI->OSPEEDR = 0x0028AAAA;
  /* Configure PIx pins Output type to push-pull */
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;
/*-- FMC Configuration ------------------------------------------------------*/
  /* Enable the FMC interface clock */
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
  FMC_Bank5_6->SDTR[0] = 0x01115351;
  /* SDRAM initialization sequence */
  /* Clock enable command */
  FMC_Bank5_6->SDCMR = 0x00000011;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
```

```
/* Delay */
for (index = 0; index<1000; index++);
/* PALL command */
FMC_Bank5_6->SDCMR = 0x00000012;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* Auto refresh command */
FMC_Bank5_6->SDCMR = 0x00000073;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* MRD register program */
FMC_Bank5_6->SDCMR = 0x00046014;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
```

```
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Set refresh count */
  tmpreg = FMC_Bank5_6->SDRTR;
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
  /* Disable write protection */
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F469xx || STM32F479xx */
  (void)(tmp);
```

```
}
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F469xx || STM32F479xx */
#elif defined (DATA_IN_ExtSRAM) || defined
(DATA_IN_ExtSDRAM)
/**
  * @brief  Setup the external memory controller.
  *         Called in startup_stm32f4xx.s before jump to main.
  *         This function configures the external memories
(SRAM/SDRAM)
  *         This SRAM/SDRAM will be used as program data memory
(including heap and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F446xx) || defined(STM32F469xx) ||
defined(STM32F479xx)
#if defined (DATA_IN_ExtSDRAM)
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
  register __IO uint32_t index;
```

```
#if defined(STM32F446xx)
  /* Enable GPIOA, GPIOC, GPIOD, GPIOE, GPIOF, GPIOG interface
     clock */
  RCC->AHB1ENR |= 0x0000007D;
#else
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and
GPIOI interface
     clock */
  RCC->AHB1ENR |= 0x000001F8;
#endif /* STM32F446xx */
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);
#if defined(STM32F446xx)
  /* Connect PAx pins to FMC Alternate function */
  GPIOA->AFR[0]  |= 0xC0000000;
  GPIOA->AFR[1]  |= 0x00000000;
  /* Configure PDx pins in Alternate function mode */
  GPIOA->MODER   |= 0x00008000;
  /* Configure PDx pins speed to 50 MHz */
  GPIOA->OSPEEDR |= 0x00008000;
  /* Configure PDx pins Output type to push-pull */
  GPIOA->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
```

```
  GPIOA->PUPDR   |= 0x00000000;
  /* Connect PCx pins to FMC Alternate function */
  GPIOC->AFR[0]  |= 0x00CC0000;
  GPIOC->AFR[1]  |= 0x00000000;
  /* Configure PDx pins in Alternate function mode */
  GPIOC->MODER   |= 0x00000A00;
  /* Configure PDx pins speed to 50 MHz */
  GPIOC->OSPEEDR |= 0x00000A00;
  /* Configure PDx pins Output type to push-pull */
  GPIOC->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOC->PUPDR   |= 0x00000000;
#endif /* STM32F446xx */
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x000000CC;
  GPIOD->AFR[1]  = 0xCC000CCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xA02A000A;
  /* Configure PDx pins speed to 50 MHz */
  GPIOD->OSPEEDR = 0xA02A000A;
  /* Configure PDx pins Output type to push-pull */
  GPIOD->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PDx pins */
```

```
GPIOD->PUPDR   = 0x00000000;
/* Connect PEx pins to FMC Alternate function */
GPIOE->AFR[0]  = 0xC00000CC;
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA800A;
/* Configure PEx pins speed to 50 MHz */
GPIOE->OSPEEDR = 0xAAAA800A;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0xCCCCCCCC;
GPIOF->AFR[1]  = 0xCCCCCCCC;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA800AAA;
/* Configure PFx pins speed to 50 MHz */
GPIOF->OSPEEDR = 0xAA800AAA;
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
GPIOF->PUPDR   = 0x00000000;
```

```
  /* Connect PGx pins to FMC Alternate function */
  GPIOG->AFR[0]  = 0xCCCCCCCC;
  GPIOG->AFR[1]  = 0xCCCCCCCC;
  /* Configure PGx pins in Alternate function mode */
  GPIOG->MODER   = 0xAAAAAAAA;
  /* Configure PGx pins speed to 50 MHz */
  GPIOG->OSPEEDR = 0xAAAAAAAA;
  /* Configure PGx pins Output type to push-pull */
  GPIOG->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
  /* Connect PHx pins to FMC Alternate function */
  GPIOH->AFR[0]  = 0x00C0CC00;
  GPIOH->AFR[1]  = 0xCCCCCCCC;
  /* Configure PHx pins in Alternate function mode */
  GPIOH->MODER   = 0xAAAA08A0;
  /* Configure PHx pins speed to 50 MHz */
  GPIOH->OSPEEDR = 0xAAAA08A0;
  /* Configure PHx pins Output type to push-pull */
  GPIOH->OTYPER  = 0x00000000;
```

```
  /* No pull-up, pull-down for PHx pins */
  GPIOH->PUPDR   = 0x00000000;
  /* Connect PIx pins to FMC Alternate function */
  GPIOI->AFR[0]  = 0xCCCCCCCC;
  GPIOI->AFR[1]  = 0x00000CC0;
  /* Configure PIx pins in Alternate function mode */
  GPIOI->MODER   = 0x0028AAAA;
  /* Configure PIx pins speed to 50 MHz */
  GPIOI->OSPEEDR = 0x0028AAAA;
  /* Configure PIx pins Output type to push-pull */
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F469xx || STM32F479xx */
/*-- FMC Configuration -------------------------------------------------------*/
  /* Enable the FMC interface clock */
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable SDRAM bank1 */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCR[0] = 0x00001954;
```

```
#else
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
#endif /* STM32F446xx */
  FMC_Bank5_6->SDTR[0] = 0x01115351;
  /* SDRAM initialization sequence */
  /* Clock enable command */
  FMC_Bank5_6->SDCMR = 0x00000011;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Delay */
  for (index = 0; index<1000; index++);
  /* PALL command */
  FMC_Bank5_6->SDCMR = 0x00000012;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Auto refresh command */
```

```
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x000000F3;
#else
  FMC_Bank5_6->SDCMR = 0x00000073;
#endif /* STM32F446xx */
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* MRD register program */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x00044014;
#else
  FMC_Bank5_6->SDCMR = 0x00046014;
#endif /* STM32F446xx */
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
```

```
  /* Set refresh count */
  tmpreg = FMC_Bank5_6->SDRTR;
#if defined(STM32F446xx)
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000050C<<1));
#else
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
#endif /* STM32F446xx */
  /* Disable write protection */
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#endif /* DATA_IN_ExtSDRAM */
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F446xx || STM32F469xx || STM32F479xx */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx) || defined(STM32F417xx)\
 || defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx) ||
defined(STM32F412Zx) || defined(STM32F412Vx)
#if defined(DATA_IN_ExtSRAM)
/*-- GPIOs Configuration -----------------------------------------------------*/
   /* Enable GPIOD, GPIOE, GPIOF and GPIOG interface clock */
  RCC->AHB1ENR   |= 0x00000078;
  /* Delay after an RCC peripheral clock enabling */
```

```
tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIODEN);
/* Connect PDx pins to FMC Alternate function */
GPIOD->AFR[0]  = 0x00CCC0CC;
GPIOD->AFR[1]  = 0xCCCCCCCC;
/* Configure PDx pins in Alternate function mode */
GPIOD->MODER   = 0xAAAA0A8A;
/* Configure PDx pins speed to 100 MHz */
GPIOD->OSPEEDR = 0xFFFF0FCF;
/* Configure PDx pins Output type to push-pull */
GPIOD->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PDx pins */
GPIOD->PUPDR   = 0x00000000;
/* Connect PEx pins to FMC Alternate function */
GPIOE->AFR[0]  = 0xC00CC0CC;
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA828A;
/* Configure PEx pins speed to 100 MHz */
GPIOE->OSPEEDR = 0xFFFFC3CF;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
```

```
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0x00CCCCCC;
GPIOF->AFR[1]  = 0xCCCC0000;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA000AAA;
/* Configure PFx pins speed to 100 MHz */
GPIOF->OSPEEDR = 0xFF000FFF;
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
GPIOF->PUPDR   = 0x00000000;
/* Connect PGx pins to FMC Alternate function */
GPIOG->AFR[0]  = 0x00CCCCCC;
GPIOG->AFR[1]  = 0x000000C0;
/* Configure PGx pins in Alternate function mode */
GPIOG->MODER   = 0x00085AAA;
/* Configure PGx pins speed to 100 MHz */
GPIOG->OSPEEDR = 0x000CAFFF;
/* Configure PGx pins Output type to push-pull */
GPIOG->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PGx pins */
GPIOG->PUPDR   = 0x00000000;
```

```
/*-- FMC/FSMC Configuration ----------------------------------------------
-*/
  /* Enable the FMC/FSMC interface clock */
  RCC->AHB3ENR         |= 0x00000001;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F469xx || STM32F479xx */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx)|| defined(STM32F417xx)\
```

```
  || defined(STM32F412Zx) || defined(STM32F412Vx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FSMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FSMC_Bank1->BTCR[2]  = 0x00001011;
  FSMC_Bank1->BTCR[3]  = 0x00000201;
  FSMC_Bank1E->BWTR[2] = 0x0FFFFFFF;
#endif /* STM32F405xx || STM32F415xx || STM32F407xx ||
STM32F417xx || STM32F412Zx || STM32F412Vx */
#endif /* DATA_IN_ExtSRAM */
#endif /* STM32F405xx || STM32F415xx || STM32F407xx ||
STM32F417xx || STM32F427xx || STM32F437xx ||\
        STM32F429xx || STM32F439xx || STM32F469xx ||
STM32F479xx || STM32F412Zx || STM32F412Vx  */
  (void)(tmp);
}
#endif /* DATA_IN_ExtSRAM && DATA_IN_ExtSDRAM */
/**
  * @}
  */
/**
  * @}
  */
/**
```

```
* @}
*/
```

CHAPTER#22 CODING GPIO DRIVERS FROM SCRATCH IN ASSEMBLY LANGUAGE (PART 2)

22.1 CODING GPIO DRIVERS FROM SCRATCH IN ASSEMBLY LANGUAGE (PART 2):

Continuing from the basics of coding GPIO drivers in assembly language, let's dive into more advanced topics and techniques.

22.2 INTERRUPT HANDLING FOR GPIO:

Handling GPIO interrupts involves configuring the microcontroller to generate an interrupt on a specific GPIO event (e.g., rising edge, falling edge).

22.2.1 INTERRUPT SERVICE ROUTINE (ISR):

Write the ISR to handle the interrupt. This routine should clear the interrupt flag and perform the desired action.

22.3 DEBOUNCING MECHANISM:

Mechanical switches can generate multiple transitions (bounce) when pressed or released. Debouncing ensures that only one transition is registered.

22.4 MANAGING MULTIPLE GPIO PINS:

If you need to manage multiple GPIO ports, follow similar steps as configuring a single port but replicate the process for each port.

22.5 OPIMIZATION AND BEST PRACTICES:

22.5.1 USE BIT-BANDING:

Some ARM Cortex-M microcontrollers support bit-banding, which allows atomic bit manipulation. Refer to the microcontroller's reference manual for bit-banding capabilities and address calculation.

22.5.2 MINIMIZE INTERRUPT LATENCY:

Keep ISRs short and efficient. Perform lengthy operations outside the ISR context.

22.5.3 ERROR HANDLING:

Implement error checks where necessary, such as verifying successful peripheral initialization.

22.5.4 CODE COMMENTS:

Use comments generously to document the purpose and functionality of each code section, especially in assembly where the intent may not be immediately obvious.

22.6 CODE OF MAIN.H:

```
/* USER CODE BEGIN Header */
/**
*******************************************************
*******************
  * @file          : main.h
  * @brief         : Header for main.c file.
```

```
  *                This file contains the common defines of the application.
**********************************************************************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H
#ifdef __cplusplus
extern "C" {
#endif
/* Includes ------------------------------------------------------------------*/
```

```
#include "stm32f4xx_hal.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */
/* USER CODE END EC */
/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */
/* USER CODE END EM */
/* Exported functions prototypes ---------------------------------------------*/
void Error_Handler(void);
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
/* Private defines -----------------------------------------------------------*/
#define B1_Pin GPIO_PIN_13
#define B1_GPIO_Port GPIOC
#define USART_TX_Pin GPIO_PIN_2
#define USART_TX_GPIO_Port GPIOA
#define USART_RX_Pin GPIO_PIN_3
```

```
#define USART_RX_GPIO_Port GPIOA
#define LD2_Pin GPIO_PIN_5
#define LD2_GPIO_Port GPIOA
#define TMS_Pin GPIO_PIN_13
#define TMS_GPIO_Port GPIOA
#define TCK_Pin GPIO_PIN_14
#define TCK_GPIO_Port GPIOA
#define SWO_Pin GPIO_PIN_3
#define SWO_GPIO_Port GPIOB
/* USER CODE BEGIN Private defines */
/* USER CODE END Private defines */
#ifdef __cplusplus
}
#endif
#endif /* __MAIN_H */
```

22.7 CODE OF STM3F4XX_HAL_CONF.H:

```
/* USER CODE BEGIN Header */
/**
*****************************************************************
*******************
  * @file    stm32f4xx_hal_conf_template.h
  * @author  MCD Application Team
  * @brief   HAL configuration template file.
```

```
  *         This file should be copied to the application folder and renamed
  *         to stm32f4xx_hal_conf.h.
*******************************************************************
  * @attention
  *
  * Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*******************************************************************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -----------------------------------*/
#ifndef __STM32F4xx_HAL_CONF_H
#define __STM32F4xx_HAL_CONF_H
#ifdef __cplusplus
 extern "C" {
#endif
```

```
/* Exported types ------------------------------------------------------------*/
/* Exported constants --------------------------------------------------------*/
/* ############################ Module Selection
################################## */
/**
  * @brief This is the list of modules to be used in the HAL driver
  */
#define HAL_MODULE_ENABLED
  /* #define HAL_CRYP_MODULE_ENABLED */
/* #define HAL_ADC_MODULE_ENABLED */
/* #define HAL_CAN_MODULE_ENABLED */
/* #define HAL_CRC_MODULE_ENABLED */
/* #define HAL_CAN_LEGACY_MODULE_ENABLED */
/* #define HAL_DAC_MODULE_ENABLED */
/* #define HAL_DCMI_MODULE_ENABLED */
/* #define HAL_DMA2D_MODULE_ENABLED */
/* #define HAL_ETH_MODULE_ENABLED */
/* #define HAL_ETH_LEGACY_MODULE_ENABLED */
/* #define HAL_NAND_MODULE_ENABLED */
/* #define HAL_NOR_MODULE_ENABLED */
/* #define HAL_PCCARD_MODULE_ENABLED */
/* #define HAL_SRAM_MODULE_ENABLED */
/* #define HAL_SDRAM_MODULE_ENABLED */
```

```
/* #define HAL_HASH_MODULE_ENABLED */
/* #define HAL_I2C_MODULE_ENABLED */
/* #define HAL_I2S_MODULE_ENABLED */
/* #define HAL_IWDG_MODULE_ENABLED */
/* #define HAL_LTDC_MODULE_ENABLED */
/* #define HAL_RNG_MODULE_ENABLED */
/* #define HAL_RTC_MODULE_ENABLED */
/* #define HAL_SAI_MODULE_ENABLED */
/* #define HAL_SD_MODULE_ENABLED */
/* #define HAL_MMC_MODULE_ENABLED */
/* #define HAL_SPI_MODULE_ENABLED */
/* #define HAL_TIM_MODULE_ENABLED */
#define HAL_UART_MODULE_ENABLED
/* #define HAL_USART_MODULE_ENABLED */
/* #define HAL_IRDA_MODULE_ENABLED */
/* #define HAL_SMARTCARD_MODULE_ENABLED */
/* #define HAL_SMBUS_MODULE_ENABLED */
/* #define HAL_WWDG_MODULE_ENABLED */
/* #define HAL_PCD_MODULE_ENABLED */
/* #define HAL_HCD_MODULE_ENABLED */
/* #define HAL_DSI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
```

```
/* #define HAL_CEC_MODULE_ENABLED */
/* #define HAL_FMPI2C_MODULE_ENABLED */
/* #define HAL_FMPSMBUS_MODULE_ENABLED */
/* #define HAL_SPDIFRX_MODULE_ENABLED */
/* #define HAL_DFSDM_MODULE_ENABLED */
/* #define HAL_LPTIM_MODULE_ENABLED */
#define HAL_GPIO_MODULE_ENABLED
#define HAL_EXTI_MODULE_ENABLED
#define HAL_DMA_MODULE_ENABLED
#define HAL_RCC_MODULE_ENABLED
#define HAL_FLASH_MODULE_ENABLED
#define HAL_PWR_MODULE_ENABLED
#define HAL_CORTEX_MODULE_ENABLED
/* ############################ HSE/HSI Values adaptation
###################### */
/**
  * @brief Adjust the value of External High Speed oscillator (HSE)
used in your application.
  *        This value is used by the RCC HAL module to compute the
system frequency
  *        (when HSE is used as system clock source, directly or through
the PLL).
  */
#if !defined  (HSE_VALUE)
```

```
  #define HSE_VALUE    8000000U /*!< Value of the External
oscillator in Hz */

#endif /* HSE_VALUE */

#if !defined  (HSE_STARTUP_TIMEOUT)

  #define HSE_STARTUP_TIMEOUT    100U   /*!< Time out for
HSE start up, in ms */

#endif /* HSE_STARTUP_TIMEOUT */

/**

  * @brief Internal High Speed oscillator (HSI) value.

  *        This value is used by the RCC HAL module to compute the
system frequency

  *        (when HSI is used as system clock source, directly or through the
PLL).

  */

#if !defined  (HSI_VALUE)

  #define HSI_VALUE    ((uint32_t)16000000U) /*!< Value of the
Internal oscillator in Hz*/

#endif /* HSI_VALUE */

/**

  * @brief Internal Low Speed oscillator (LSI) value.

  */

#if !defined  (LSI_VALUE)

 #define LSI_VALUE  32000U       /*!< LSI Typical Value in Hz*/

#endif /* LSI_VALUE */                    /*!< Value of the Internal Low
Speed oscillator in Hz
```

The real value may vary depending on the variations

in voltage and temperature.*/

/**

* @brief External Low Speed oscillator (LSE) value.

*/

#if !defined (LSE_VALUE)

#define LSE_VALUE 32768U /*!< Value of the External Low Speed oscillator in Hz */

#endif /* LSE_VALUE */

#if !defined (LSE_STARTUP_TIMEOUT)

#define LSE_STARTUP_TIMEOUT 5000U /*!< Time out for LSE start up, in ms */

#endif /* LSE_STARTUP_TIMEOUT */

/**

* @brief External clock source for I2S peripheral

* This value is used by the I2S HAL module to compute the I2S clock source

* frequency, this source is inserted directly through I2S_CKIN pad.

*/

#if !defined (EXTERNAL_CLOCK_VALUE)

#define EXTERNAL_CLOCK_VALUE 12288000U /*!< Value of the External audio frequency in Hz*/

#endif /* EXTERNAL_CLOCK_VALUE */

/* Tip: To avoid modifying this file each time you need to use different HSE,

=== you can define the HSE value in your toolchain compiler preprocessor. */

/* ############################## System Configuration ########################### */

/**

* @brief This is the HAL system configuration section

*/

#define VDD_VALUE 3300U /*!< Value of VDD in mv */

#define TICK_INT_PRIORITY 0U /*!< tick interrupt priority */

#define USE_RTOS 0U

#define PREFETCH_ENABLE 1U

#define INSTRUCTION_CACHE_ENABLE 1U

#define DATA_CACHE_ENABLE 1U

#define USE_HAL_ADC_REGISTER_CALLBACKS 0U /* ADC register callback disabled */

#define USE_HAL_CAN_REGISTER_CALLBACKS 0U /* CAN register callback disabled */

#define USE_HAL_CEC_REGISTER_CALLBACKS 0U /* CEC register callback disabled */

#define USE_HAL_CRYP_REGISTER_CALLBACKS 0U /* CRYP register callback disabled */

#define USE_HAL_DAC_REGISTER_CALLBACKS 0U /* DAC register callback disabled */

#define USE_HAL_DCMI_REGISTER_CALLBACKS 0U /* DCMI register callback disabled */

#define USE_HAL_DFSDM_REGISTER_CALLBACKS 0U /* DFSDM register callback disabled */

#define USE_HAL_DMA2D_REGISTER_CALLBACKS 0U /* DMA2D register callback disabled */

#define USE_HAL_DSI_REGISTER_CALLBACKS 0U /* DSI register callback disabled */

#define USE_HAL_ETH_REGISTER_CALLBACKS 0U /* ETH register callback disabled */

#define USE_HAL_HASH_REGISTER_CALLBACKS 0U /* HASH register callback disabled */

#define USE_HAL_HCD_REGISTER_CALLBACKS 0U /* HCD register callback disabled */

#define USE_HAL_I2C_REGISTER_CALLBACKS 0U /* I2C register callback disabled */

#define USE_HAL_FMPI2C_REGISTER_CALLBACKS 0U /* FMPI2C register callback disabled */

#define USE_HAL_FMPSMBUS_REGISTER_CALLBACKS 0U /* FMPSMBUS register callback disabled */

#define USE_HAL_I2S_REGISTER_CALLBACKS 0U /* I2S register callback disabled */

#define USE_HAL_IRDA_REGISTER_CALLBACKS 0U /* IRDA register callback disabled */

#define USE_HAL_LPTIM_REGISTER_CALLBACKS 0U /* LPTIM register callback disabled */

#define USE_HAL_LTDC_REGISTER_CALLBACKS 0U /* LTDC register callback disabled */

#define USE_HAL_MMC_REGISTER_CALLBACKS 0U /* MMC register callback disabled */

#define USE_HAL_NAND_REGISTER_CALLBACKS 0U /* NAND register callback disabled */

#define USE_HAL_NOR_REGISTER_CALLBACKS 0U /* NOR register callback disabled */

#define USE_HAL_PCCARD_REGISTER_CALLBACKS 0U /* PCCARD register callback disabled */

#define USE_HAL_PCD_REGISTER_CALLBACKS 0U /* PCD register callback disabled */

#define USE_HAL_QSPI_REGISTER_CALLBACKS 0U /* QSPI register callback disabled */

#define USE_HAL_RNG_REGISTER_CALLBACKS 0U /* RNG register callback disabled */

#define USE_HAL_RTC_REGISTER_CALLBACKS 0U /* RTC register callback disabled */

#define USE_HAL_SAI_REGISTER_CALLBACKS 0U /* SAI register callback disabled */

#define USE_HAL_SD_REGISTER_CALLBACKS 0U /* SD register callback disabled */

#define USE_HAL_SMARTCARD_REGISTER_CALLBACKS 0U /* SMARTCARD register callback disabled */

#define USE_HAL_SDRAM_REGISTER_CALLBACKS 0U /* SDRAM register callback disabled */

#define USE_HAL_SRAM_REGISTER_CALLBACKS 0U /* SRAM register callback disabled */

#define USE_HAL_SPDIFRX_REGISTER_CALLBACKS 0U /* SPDIFRX register callback disabled */

#define USE_HAL_SMBUS_REGISTER_CALLBACKS 0U /* SMBUS register callback disabled */

#define USE_HAL_SPI_REGISTER_CALLBACKS 0U /* SPI register callback disabled */

#define USE_HAL_TIM_REGISTER_CALLBACKS 0U /* TIM register callback disabled */

#define USE_HAL_UART_REGISTER_CALLBACKS 0U /* UART register callback disabled */

#define USE_HAL_USART_REGISTER_CALLBACKS 0U /* USART register callback disabled */

#define USE_HAL_WWDG_REGISTER_CALLBACKS 0U /* WWDG register callback disabled */

/* ############################ Assert Selection ################################ */

/**

* @brief Uncomment the line below to expanse the "assert_param" macro in the

* HAL drivers code

*/

/* #define USE_FULL_ASSERT 1U */

/* ################### Ethernet peripheral configuration ###################### */

/* Section 1 : Ethernet peripheral configuration */

/* MAC ADDRESS: MAC_ADDR0:MAC_ADDR1:MAC_ADDR2:MAC_ADDR3:MAC_ADDR4:MAC_ADDR5 */

#define MAC_ADDR0 2U

#define MAC_ADDR1 0U

#define MAC_ADDR2 0U

#define MAC_ADDR3 0U

#define MAC_ADDR4 0U

#define MAC_ADDR5 0U

/* Definition of the Ethernet driver buffers size and count */

#define ETH_RX_BUF_SIZE ETH_MAX_PACKET_SIZE /* buffer size for receive */

#define ETH_TX_BUF_SIZE ETH_MAX_PACKET_SIZE /* buffer size for transmit */

#define ETH_RXBUFNB 4U /* 4 Rx buffers of size ETH_RX_BUF_SIZE */

#define ETH_TXBUFNB 4U /* 4 Tx buffers of size ETH_TX_BUF_SIZE */

/* Section 2: PHY configuration section */

/* DP83848_PHY_ADDRESS Address*/

#define DP83848_PHY_ADDRESS

/* PHY Reset delay these values are based on a 1 ms Systick interrupt*/

#define PHY_RESET_DELAY 0x000000FFU

/* PHY Configuration delay */

```
#define PHY_CONFIG_DELAY            0x00000FFFU
#define PHY_READ_TO                 0x0000FFFFU
#define PHY_WRITE_TO                0x0000FFFFU
/* Section 3: Common PHY Registers */
#define PHY_BCR                     ((uint16_t)0x0000U)    /*!<
Transceiver Basic Control Register   */
#define PHY_BSR                     ((uint16_t)0x0001U)    /*!<
Transceiver Basic Status Register    */
#define PHY_RESET                   ((uint16_t)0x8000U)  /*!< PHY
Reset */
#define PHY_LOOPBACK                ((uint16_t)0x4000U)  /*!<
Select loop-back mode */
#define PHY_FULLDUPLEX_100M         ((uint16_t)0x2100U)
/*!< Set the full-duplex mode at 100 Mb/s */
#define PHY_HALFDUPLEX_100M         ((uint16_t)0x2000U)
/*!< Set the half-duplex mode at 100 Mb/s */
#define PHY_FULLDUPLEX_10M          ((uint16_t)0x0100U)
/*!< Set the full-duplex mode at 10 Mb/s  */
#define PHY_HALFDUPLEX_10M          ((uint16_t)0x0000U)
/*!< Set the half-duplex mode at 10 Mb/s  */
#define PHY_AUTONEGOTIATION         ((uint16_t)0x1000U)
/*!< Enable auto-negotiation function     */
#define PHY_RESTART_AUTONEGOTIATION
((uint16_t)0x0200U)  /*!< Restart auto-negotiation function    */
#define PHY_POWERDOWN               ((uint16_t)0x0800U)  /*!<
Select the power down mode          */
```

```
#define PHY_ISOLATE                    ((uint16_t)0x0400U)  /*!< Isolate
PHY from MII                 */

#define PHY_AUTONEGO_COMPLETE          ((uint16_t)0x0020U)
/*!< Auto-Negotiation process completed   */

#define PHY_LINKED_STATUS              ((uint16_t)0x0004U)  /*!<
Valid link established               */

#define PHY_JABBER_DETECTION           ((uint16_t)0x0002U)
/*!< Jabber condition detected            */

/* Section 4: Extended PHY Registers */

#define PHY_SR                         ((uint16_t))    /*!< PHY status register
Offset                    */

#define PHY_SPEED_STATUS               ((uint16_t))  /*!< PHY Speed
mask                           */

#define PHY_DUPLEX_STATUS              ((uint16_t))  /*!< PHY
Duplex mask                            */

/* ################### SPI peripheral configuration
############################ */

/* CRC FEATURE: Use to activate CRC feature inside HAL SPI Driver

* Activated: CRC code is present inside driver

* Deactivated: CRC code cleaned from driver

*/

#define USE_SPI_CRC                    0U

/* Includes ------------------------------------------------------------------*/

/**

  * @brief Include module's header file
```

```
  */
#ifdef HAL_RCC_MODULE_ENABLED
  #include "stm32f4xx_hal_rcc.h"
#endif /* HAL_RCC_MODULE_ENABLED */
#ifdef HAL_GPIO_MODULE_ENABLED
  #include "stm32f4xx_hal_gpio.h"
#endif /* HAL_GPIO_MODULE_ENABLED */
#ifdef HAL_EXTI_MODULE_ENABLED
  #include "stm32f4xx_hal_exti.h"
#endif /* HAL_EXTI_MODULE_ENABLED */
#ifdef HAL_DMA_MODULE_ENABLED
  #include "stm32f4xx_hal_dma.h"
#endif /* HAL_DMA_MODULE_ENABLED */
#ifdef HAL_CORTEX_MODULE_ENABLED
  #include "stm32f4xx_hal_cortex.h"
#endif /* HAL_CORTEX_MODULE_ENABLED */
#ifdef HAL_ADC_MODULE_ENABLED
  #include "stm32f4xx_hal_adc.h"
#endif /* HAL_ADC_MODULE_ENABLED */
#ifdef HAL_CAN_MODULE_ENABLED
  #include "stm32f4xx_hal_can.h"
#endif /* HAL_CAN_MODULE_ENABLED */
#ifdef HAL_CAN_LEGACY_MODULE_ENABLED
```

```
  #include "stm32f4xx_hal_can_legacy.h"
#endif /* HAL_CAN_LEGACY_MODULE_ENABLED */
#ifdef HAL_CRC_MODULE_ENABLED
  #include "stm32f4xx_hal_crc.h"
#endif /* HAL_CRC_MODULE_ENABLED */
#ifdef HAL_CRYP_MODULE_ENABLED
  #include "stm32f4xx_hal_cryp.h"
#endif /* HAL_CRYP_MODULE_ENABLED */
#ifdef HAL_DMA2D_MODULE_ENABLED
  #include "stm32f4xx_hal_dma2d.h"
#endif /* HAL_DMA2D_MODULE_ENABLED */
#ifdef HAL_DAC_MODULE_ENABLED
  #include "stm32f4xx_hal_dac.h"
#endif /* HAL_DAC_MODULE_ENABLED */
#ifdef HAL_DCMI_MODULE_ENABLED
  #include "stm32f4xx_hal_dcmi.h"
#endif /* HAL_DCMI_MODULE_ENABLED */
#ifdef HAL_ETH_MODULE_ENABLED
  #include "stm32f4xx_hal_eth.h"
#endif /* HAL_ETH_MODULE_ENABLED */
#ifdef HAL_ETH_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_eth_legacy.h"
#endif /* HAL_ETH_LEGACY_MODULE_ENABLED */
```

```
#ifdef HAL_FLASH_MODULE_ENABLED
  #include "stm32f4xx_hal_flash.h"
#endif /* HAL_FLASH_MODULE_ENABLED */
#ifdef HAL_SRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sram.h"
#endif /* HAL_SRAM_MODULE_ENABLED */
#ifdef HAL_NOR_MODULE_ENABLED
  #include "stm32f4xx_hal_nor.h"
#endif /* HAL_NOR_MODULE_ENABLED */
#ifdef HAL_NAND_MODULE_ENABLED
  #include "stm32f4xx_hal_nand.h"
#endif /* HAL_NAND_MODULE_ENABLED */
#ifdef HAL_PCCARD_MODULE_ENABLED
  #include "stm32f4xx_hal_pccard.h"
#endif /* HAL_PCCARD_MODULE_ENABLED */
#ifdef HAL_SDRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sdram.h"
#endif /* HAL_SDRAM_MODULE_ENABLED */
#ifdef HAL_HASH_MODULE_ENABLED
 #include "stm32f4xx_hal_hash.h"
#endif /* HAL_HASH_MODULE_ENABLED */
#ifdef HAL_I2C_MODULE_ENABLED
 #include "stm32f4xx_hal_i2c.h"
```

```
#endif /* HAL_I2C_MODULE_ENABLED */
#ifdef HAL_SMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_smbus.h"
#endif /* HAL_SMBUS_MODULE_ENABLED */
#ifdef HAL_I2S_MODULE_ENABLED
 #include "stm32f4xx_hal_i2s.h"
#endif /* HAL_I2S_MODULE_ENABLED */
#ifdef HAL_IWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_iwdg.h"
#endif /* HAL_IWDG_MODULE_ENABLED */
#ifdef HAL_LTDC_MODULE_ENABLED
 #include "stm32f4xx_hal_ltdc.h"
#endif /* HAL_LTDC_MODULE_ENABLED */
#ifdef HAL_PWR_MODULE_ENABLED
 #include "stm32f4xx_hal_pwr.h"
#endif /* HAL_PWR_MODULE_ENABLED */
#ifdef HAL_RNG_MODULE_ENABLED
 #include "stm32f4xx_hal_rng.h"
#endif /* HAL_RNG_MODULE_ENABLED */
#ifdef HAL_RTC_MODULE_ENABLED
 #include "stm32f4xx_hal_rtc.h"
#endif /* HAL_RTC_MODULE_ENABLED */
#ifdef HAL_SAI_MODULE_ENABLED
```

```
 #include "stm32f4xx_hal_sai.h"
#endif /* HAL_SAI_MODULE_ENABLED */
#ifdef HAL_SD_MODULE_ENABLED
 #include "stm32f4xx_hal_sd.h"
#endif /* HAL_SD_MODULE_ENABLED */
#ifdef HAL_SPI_MODULE_ENABLED
 #include "stm32f4xx_hal_spi.h"
#endif /* HAL_SPI_MODULE_ENABLED */
#ifdef HAL_TIM_MODULE_ENABLED
 #include "stm32f4xx_hal_tim.h"
#endif /* HAL_TIM_MODULE_ENABLED */
#ifdef HAL_UART_MODULE_ENABLED
 #include "stm32f4xx_hal_uart.h"
#endif /* HAL_UART_MODULE_ENABLED */
#ifdef HAL_USART_MODULE_ENABLED
 #include "stm32f4xx_hal_usart.h"
#endif /* HAL_USART_MODULE_ENABLED */
#ifdef HAL_IRDA_MODULE_ENABLED
 #include "stm32f4xx_hal_irda.h"
#endif /* HAL_IRDA_MODULE_ENABLED */
#ifdef HAL_SMARTCARD_MODULE_ENABLED
 #include "stm32f4xx_hal_smartcard.h"
#endif /* HAL_SMARTCARD_MODULE_ENABLED */
```

```
#ifdef HAL_WWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_wwdg.h"
#endif /* HAL_WWDG_MODULE_ENABLED */
#ifdef HAL_PCD_MODULE_ENABLED
 #include "stm32f4xx_hal_pcd.h"
#endif /* HAL_PCD_MODULE_ENABLED */
#ifdef HAL_HCD_MODULE_ENABLED
 #include "stm32f4xx_hal_hcd.h"
#endif /* HAL_HCD_MODULE_ENABLED */
#ifdef HAL_DSI_MODULE_ENABLED
 #include "stm32f4xx_hal_dsi.h"
#endif /* HAL_DSI_MODULE_ENABLED */
#ifdef HAL_QSPI_MODULE_ENABLED
 #include "stm32f4xx_hal_qspi.h"
#endif /* HAL_QSPI_MODULE_ENABLED */
#ifdef HAL_CEC_MODULE_ENABLED
 #include "stm32f4xx_hal_cec.h"
#endif /* HAL_CEC_MODULE_ENABLED */
#ifdef HAL_FMPI2C_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpi2c.h"
#endif /* HAL_FMPI2C_MODULE_ENABLED */
#ifdef HAL_FMPSMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpsmbus.h"
```

```
#endif /* HAL_FMPSMBUS_MODULE_ENABLED */
#ifdef HAL_SPDIFRX_MODULE_ENABLED
 #include "stm32f4xx_hal_spdifrx.h"
#endif /* HAL_SPDIFRX_MODULE_ENABLED */
#ifdef HAL_DFSDM_MODULE_ENABLED
 #include "stm32f4xx_hal_dfsdm.h"
#endif /* HAL_DFSDM_MODULE_ENABLED */
#ifdef HAL_LPTIM_MODULE_ENABLED
 #include "stm32f4xx_hal_lptim.h"
#endif /* HAL_LPTIM_MODULE_ENABLED */
#ifdef HAL_MMC_MODULE_ENABLED
 #include "stm32f4xx_hal_mmc.h"
#endif /* HAL_MMC_MODULE_ENABLED */
/* Exported macro ------------------------------------------------------------*/
#ifdef  USE_FULL_ASSERT
/**
  * @brief  The assert_param macro is used for function's parameters
check.
  * @param  expr If expr is false, it calls assert_failed function
  *         which reports the name of the source file and the source
  *         line number of the call that failed.
  *         If expr is true, it returns no value.
  * @retval None
```

```
  */
  #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t
*)__FILE__, __LINE__))
/* Exported functions ---------------------------------------------------- */
  void assert_failed(uint8_t* file, uint32_t line);
#else
  #define assert_param(expr) ((void)0U)
#endif /* USE_FULL_ASSERT */
#ifdef __cplusplus
}
#endif
#endif /* __STM32F4xx_HAL_CONF_H */
```

22.8 CODE OF STM32F4XX_IT.H:

```
/* USER CODE BEGIN Header */
/**
*******************************************************************
  * @file    stm32f4xx_it.h
  * @brief   This file contains the headers of the interrupt handlers.
*******************************************************************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
```

```
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_IT_H
#define __STM32F4xx_IT_H
#ifdef __cplusplus
 extern "C" {
#endif
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants --------------------------------------------------------*/
```

```
/* USER CODE BEGIN EC */
/* USER CODE END EC */
/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */
/* USER CODE END EM */
/* Exported functions prototypes ---------------------------------------------*/
void NMI_Handler(void);
void HardFault_Handler(void);
void MemManage_Handler(void);
void BusFault_Handler(void);
void UsageFault_Handler(void);
void SVC_Handler(void);
void DebugMon_Handler(void);
void PendSV_Handler(void);
void SysTick_Handler(void);
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
#ifdef __cplusplus
}
#endif
#endif /* __STM32F4xx_IT_H */
```

22.9 CODE OF MAIN.C:

```
/* USER CODE BEGIN Header */
```

```
/**
*****************************************************************
*******************
  * @file           : main.c
  * @brief          : Main program body
*****************************************************************
*******************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*****************************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
```

```
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
/* USER CODE END PTD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
UART_HandleTypeDef huart2;
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
```

```
/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */
  /* USER CODE END 1 */
  /* MCU Configuration--------------------------------------------------------*/
  /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
  HAL_Init();
  /* USER CODE BEGIN Init */
  /* USER CODE END Init */
  /* Configure the system clock */
  SystemClock_Config();
  /* USER CODE BEGIN SysInit */
  /* USER CODE END SysInit */
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  /* USER CODE BEGIN 2 */
  /* USER CODE END 2 */
```

```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();

__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_
VOLTAGE_SCALE1);
```

```
  /** Initializes the RCC Oscillators according to the specified
parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue =
RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 16;
  RCC_OscInitStruct.PLL.PLLN = 336;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV4;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK

|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
```

```
  RCC_ClkInitStruct.SYSCLKSource =
RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct,
FLASH_LATENCY_2) != HAL_OK)
  {
    Error_Handler();
  }
}
/**
  * @brief USART2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART2_UART_Init(void)
{
  /* USER CODE BEGIN USART2_Init 0 */
  /* USER CODE END USART2_Init 0 */
  /* USER CODE BEGIN USART2_Init 1 */
  /* USER CODE END USART2_Init 1 */
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 115200;
```

```
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART2_Init 2 */
  /* USER CODE END USART2_Init 2 */
}
/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */
```

```
  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin,
GPIO_PIN_RESET);
  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);
  /*Configure GPIO pin : LD2_Pin */
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);
/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}
/* USER CODE BEGIN 4 */
```

```
/* USER CODE END 4 */
/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return
state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}
#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line
number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
```

```
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

22.10 CODE OF MAIN.S:

```
; Assuming the use of STM32 microcontroller and GPIO port A
; Base address of GPIOA
.equ GPIOA_BASE, 0x48000000
; Register offsets
.equ GPIOA_MODER_OFFSET, 0x00  ; Mode register offset
.equ GPIOA_OTYPER_OFFSET, 0x04 ; Output type register offset
.equ GPIOA_OSPEEDR_OFFSET, 0x08 ; Output speed register offset
.equ GPIOA_PUPDR_OFFSET, 0x0C  ; Pull-up/pull-down register offset
; Clock enable register (specific to microcontroller family)
.equ RCC_AHBENR, 0x40021014     ; RCC AHB peripheral clock enable register
```

```
.equ RCC_IOPAEN, 0x00020000     ; Bit for enabling GPIOA clock
; Addresses
.equ GPIOA_MODER, GPIOA_BASE + GPIOA_MODER_OFFSET
.equ GPIOA_OTYPER, GPIOA_BASE + GPIOA_OTYPER_OFFSET
.equ GPIOA_OSPEEDR, GPIOA_BASE +
GPIOA_OSPEEDR_OFFSET
.equ GPIOA_PUPDR, GPIOA_BASE + GPIOA_PUPDR_OFFSET
.global main
.syntax unified
main:
   ; Enable clock for GPIOA
   LDR R0, =RCC_AHBENR
   LDR R1, [R0]              ; Load current RCC_AHBENR value
   ORR R1, R1, #RCC_IOPAEN     ; Set bit for GPIOA clock enable
   STR R1, [R0]              ; Write back to RCC_AHBENR
   ; Configure GPIOA pin 0 as output
   LDR R0, =GPIOA_MODER
   LDR R1, [R0]              ; Load current GPIOA_MODER value
   BIC R1, R1, #(0x3 << (0 * 2)); Clear mode bits for pin 0
   ORR R1, R1, #(0x1 << (0 * 2)); Set mode bits to output (01)
   STR R1, [R0]              ; Write back to GPIOA_MODER
   ; Configure GPIOA pin 0 as push-pull (default, so no need to set
GPIOA_OTYPER)
   ; Set GPIO speed to medium for pin 0 (01)
```

```
LDR R0, =GPIOA_OSPEEDR
LDR R1, [R0]              ; Load current GPIOA_OSPEEDR value
BIC R1, R1, #(0x3 << (0 * 2)); Clear speed bits for pin 0
ORR R1, R1, #(0x1 << (0 * 2)); Set speed bits to medium speed (01)
STR R1, [R0]              ; Write back to GPIOA_OSPEEDR
; Disable pull-up/pull-down for pin 0 (00)
LDR R0, =GPIOA_PUPDR
LDR R1, [R0]              ; Load current GPIOA_PUPDR value
BIC R1, R1, #(0x3 << (0 * 2)); Clear pull-up/pull-down bits for pin 0
STR R1, [R0]              ; Write back to GPIOA_PUPDR
; End of configuration
B . ; Infinite loop to keep the program running
```

22.11 CODE OF STM32F4XX_HAL_MSP.C:

```
/* USER CODE BEGIN Header */
/**
*******************************************************************
  * @file        stm32f4xx_hal_msp.c
  * @brief       This file provides code for the MSP Initialization
  *              and de-Initialization codes.
*******************************************************************
  * @attention
  *
```

```
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN TD */
/* USER CODE END TD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN Define */
/* USER CODE END Define */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN Macro */
```

```
/* USER CODE END Macro */
/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* External functions --------------------------------------------------------*/
/* USER CODE BEGIN ExternalFunctions */
/* USER CODE END ExternalFunctions */
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/**
  * Initializes the Global MSP.
  */
void HAL_MspInit(void)
{
  /* USER CODE BEGIN MspInit 0 */
  /* USER CODE END MspInit 0 */
  __HAL_RCC_SYSCFG_CLK_ENABLE();
  __HAL_RCC_PWR_CLK_ENABLE();
  HAL_NVIC_SetPriorityGrouping(NVIC_PRIORITYGROUP_0);
  /* System interrupt init*/
```

```
  /* USER CODE BEGIN MspInit 1 */
  /* USER CODE END MspInit 1 */
}
/**
* @brief UART MSP Initialization
* This function configures the hardware resources used in this example
* @param huart: UART handle pointer
* @retval None
*/
void HAL_UART_MspInit(UART_HandleTypeDef* huart)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  if(huart->Instance==USART2)
  {
  /* USER CODE BEGIN USART2_MspInit 0 */
  /* USER CODE END USART2_MspInit 0 */
    /* Peripheral clock enable */
    __HAL_RCC_USART2_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();
    /**USART2 GPIO Configuration
    PA2     ------> USART2_TX
    PA3     ------> USART2_RX
    */
```

```
    GPIO_InitStruct.Pin = USART_TX_Pin|USART_RX_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF7_USART2;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
  /* USER CODE BEGIN USART2_MspInit 1 */
  /* USER CODE END USART2_MspInit 1 */
  }
}
/**
* @brief UART MSP De-Initialization
* This function freeze the hardware resources used in this example
* @param huart: UART handle pointer
* @retval None
*/
void HAL_UART_MspDeInit(UART_HandleTypeDef* huart)
{
  if(huart->Instance==USART2)
  {
  /* USER CODE BEGIN USART2_MspDeInit 0 */
  /* USER CODE END USART2_MspDeInit 0 */
    /* Peripheral clock disable */
```

```
    __HAL_RCC_USART2_CLK_DISABLE();
    /**USART2 GPIO Configuration
    PA2     ------> USART2_TX
    PA3     ------> USART2_RX
    */
    HAL_GPIO_DeInit(GPIOA, USART_TX_Pin|USART_RX_Pin);
  /* USER CODE BEGIN USART2_MspDeInit 1 */
  /* USER CODE END USART2_MspDeInit 1 */
  }
}
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

22.12 CODE OF STM32F4XX_IT.C:

```
/* USER CODE BEGIN Header */
/**
*******************************************************
*******************
  * @file    stm32f4xx_it.c
  * @brief   Interrupt Service Routines.
*******************************************************
*******************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
```

```
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stm32f4xx_it.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN TD */
/* USER CODE END TD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
```

```
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/* External variables --------------------------------------------------------*/
/* USER CODE BEGIN EV */
/* USER CODE END EV */
/******************************************************************************/
/*            Cortex-M4 Processor Interruption and Exception Handlers          */
/******************************************************************************/
/**
  * @brief This function handles Non maskable interrupt.
  */
void NMI_Handler(void)
```

```
{
  /* USER CODE BEGIN NonMaskableInt_IRQn 0 */
  /* USER CODE END NonMaskableInt_IRQn 0 */
  /* USER CODE BEGIN NonMaskableInt_IRQn 1 */
   while (1)
  {
  }
  /* USER CODE END NonMaskableInt_IRQn 1 */
}
/**
  * @brief This function handles Hard fault interrupt.
  */
void HardFault_Handler(void)
{
  /* USER CODE BEGIN HardFault_IRQn 0 */
  /* USER CODE END HardFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_HardFault_IRQn 0 */
    /* USER CODE END W1_HardFault_IRQn 0 */
  }
}
/**
```

```
  * @brief This function handles Memory management fault.
  */
void MemManage_Handler(void)
{
  /* USER CODE BEGIN MemoryManagement_IRQn 0 */
  /* USER CODE END MemoryManagement_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_MemoryManagement_IRQn 0 */
    /* USER CODE END W1_MemoryManagement_IRQn 0 */
  }
}
/**
  * @brief This function handles Pre-fetch fault, memory access fault.
  */
void BusFault_Handler(void)
{
  /* USER CODE BEGIN BusFault_IRQn 0 */
  /* USER CODE END BusFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_BusFault_IRQn 0 */
    /* USER CODE END W1_BusFault_IRQn 0 */
```

```
  }
}
/**
  * @brief This function handles Undefined instruction or illegal state.
  */
void UsageFault_Handler(void)
{
  /* USER CODE BEGIN UsageFault_IRQn 0 */
  /* USER CODE END UsageFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_UsageFault_IRQn 0 */
    /* USER CODE END W1_UsageFault_IRQn 0 */
  }
}
/**
  * @brief This function handles System service call via SWI
instruction.
  */
void SVC_Handler(void)
{
  /* USER CODE BEGIN SVCall_IRQn 0 */
  /* USER CODE END SVCall_IRQn 0 */
```

```
  /* USER CODE BEGIN SVCall_IRQn 1 */
  /* USER CODE END SVCall_IRQn 1 */
}
/**
  * @brief This function handles Debug monitor.
  */
void DebugMon_Handler(void)
{
  /* USER CODE BEGIN DebugMonitor_IRQn 0 */
  /* USER CODE END DebugMonitor_IRQn 0 */
  /* USER CODE BEGIN DebugMonitor_IRQn 1 */
  /* USER CODE END DebugMonitor_IRQn 1 */
}
/**
  * @brief This function handles Pendable request for system service.
  */
void PendSV_Handler(void)
{
  /* USER CODE BEGIN PendSV_IRQn 0 */
  /* USER CODE END PendSV_IRQn 0 */
  /* USER CODE BEGIN PendSV_IRQn 1 */
  /* USER CODE END PendSV_IRQn 1 */
}
```

```
/**
  * @brief This function handles System tick timer.
  */
void SysTick_Handler(void)
{
  /* USER CODE BEGIN SysTick_IRQn 0 */
  /* USER CODE END SysTick_IRQn 0 */
  HAL_IncTick();
  /* USER CODE BEGIN SysTick_IRQn 1 */
  /* USER CODE END SysTick_IRQn 1 */
}
/******************************************************************************/
/* STM32F4xx Peripheral Interrupt Handlers                                    */
/* Add here the Interrupt Handlers for the used peripherals.                  */
/* For the available peripheral interrupt handler names,                      */
/* please refer to the startup file (startup_stm32f4xx.s).                   */
/******************************************************************************/
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

22.13 CODE OF SYSCALLS.C:

```
/**
**********************************************************
*******************
 * @file      syscalls.c
 * @author    Auto-generated by STM32CubeIDE
 * @brief     STM32CubeIDE Minimal System calls file
 *
 *            For more information about which c-functions
 *            need which of these lowlevel functions
 *            please consult the Newlib libc-manual
**********************************************************
*******************
 * @attention
 *
 * Copyright (c) 2020-2024 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the
LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
```

```
*****************************************************
*******************
 */
/* Includes */
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
#include <sys/times.h>
/* Variables */
extern int __io_putchar(int ch) __attribute__((weak));
extern int __io_getchar(void) __attribute__((weak));
char *__env[1] = { 0 };
char **environ = __env;
/* Functions */
void initialise_monitor_handles()
{
}
int _getpid(void)
{
```

```
  return 1;
}
int _kill(int pid, int sig)
{
  (void)pid;
  (void)sig;
  errno = EINVAL;
  return -1;
}
void _exit (int status)
{
  _kill(status, -1);
  while (1) {}    /* Make sure we hang here */
}
__attribute__((weak)) int _read(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;
  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    *ptr++ = __io_getchar();
  }
  return len;
```

```
}
__attribute__((weak)) int _write(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;
  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    __io_putchar(*ptr++);
  }
  return len;
}
int _close(int file)
{
  (void)file;
  return -1;
}
int _fstat(int file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
int _isatty(int file)
```

```
{
  (void)file;
  return 1;
}
int _lseek(int file, int ptr, int dir)
{
  (void)file;
  (void)ptr;
  (void)dir;
  return 0;
}
int _open(char *path, int flags, ...)
{
  (void)path;
  (void)flags;
  /* Pretend like we always fail */
  return -1;
}
int _wait(int *status)
{
  (void)status;
  errno = ECHILD;
  return -1;
```

```
}
int _unlink(char *name)
{
  (void)name;
  errno = ENOENT;
  return -1;
}
int _times(struct tms *buf)
{
  (void)buf;
  return -1;
}
int _stat(char *file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
int _link(char *old, char *new)
{
  (void)old;
  (void)new;
  errno = EMLINK;
```

```
  return -1;
}
int _fork(void)
{
  errno = EAGAIN;
  return -1;
}
int _execve(char *name, char **argv, char **env)
{
  (void)name;
  (void)argv;
  (void)env;
  errno = ENOMEM;
  return -1;
}
```

22.14 CODE OF SYSMEM.C:

```
/**
*****************************************************************
*******************
 * @file      sysmem.c
 * @author    Generated by STM32CubeIDE
 * @brief     STM32CubeIDE System Memory calls file
 *
```

```
 *          For more information about which C functions
 *          need which of these lowlevel functions
 *          please consult the newlib libc manual
**********************************************************
*******************
 * @attention
 *
 * Copyright (c) 2024 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the
LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
**********************************************************
*******************
 */
/* Includes */
#include <errno.h>
#include <stdint.h>
/**
 * Pointer to the current high watermark of the heap usage
 */
static uint8_t *__sbrk_heap_end = NULL;
```

```
/**
 * @brief _sbrk() allocates memory to the newlib heap and is used by malloc
 *        and others from the C library
 *
 * @verbatim
 *
 ##########################################################################
 * #  .data  #  .bss  #       newlib heap       #          MSP stack          #
 * #         #        #                         # Reserved by _Min_Stack_Size #
 *
 ##########################################################################
 * ^-- RAM start      ^-- _end                          _estack, RAM end --^
 * @endverbatim
 *
 * This implementation starts allocating at the '_end' linker symbol
 * The '_Min_Stack_Size' linker symbol reserves a memory for the MSP stack
 * The implementation considers '_estack' linker symbol to be RAM end
 * NOTE: If the MSP stack, at any point during execution, grows larger than the
 * reserved size, please increase the '_Min_Stack_Size'.
 *
```

```
 * @param incr Memory size
 * @return Pointer to allocated memory
 */
void *_sbrk(ptrdiff_t incr)
{
  extern uint8_t _end; /* Symbol defined in the linker script */
  extern uint8_t _estack; /* Symbol defined in the linker script */
  extern uint32_t _Min_Stack_Size; /* Symbol defined in the linker
script */
  const uint32_t stack_limit = (uint32_t)&_estack -
(uint32_t)&_Min_Stack_Size;
  const uint8_t *max_heap = (uint8_t *)stack_limit;
  uint8_t *prev_heap_end;
  /* Initialize heap end at first call */
  if (NULL == __sbrk_heap_end)
  {
    __sbrk_heap_end = &_end;
  }
  /* Protect heap from growing into the reserved MSP stack */
  if (__sbrk_heap_end + incr > max_heap)
  {
    errno = ENOMEM;
    return (void *)-1;
  }
```

```
  prev_heap_end = __sbrk_heap_end;
  __sbrk_heap_end += incr;
  return (void *)prev_heap_end;
}
```

22.15 CODE OF SYSTEM_STM32F4XX.C:

```
/**
*****************************************************************
*******************
  * @file    system_stm32f4xx.c
  * @author  MCD Application Team
  * @brief   CMSIS Cortex-M4 Device Peripheral Access Layer System
Source File.
  *
  *   This file provides two functions and one global variable to be called
from
  *   user application:
  *      - SystemInit(): This function is called at startup just after reset and
  *                      before branch to main program. This call is made inside
  *                      the "startup_stm32f4xx.s" file.
  *
  *      - SystemCoreClock variable: Contains the core clock (HCLK), it
can be used
  *                                  by the user application to setup the SysTick
  *                                  timer or configure other parameters.
```

```
  *
  *      - SystemCoreClockUpdate(): Updates the variable
SystemCoreClock and must
  *                                be called whenever the core clock is changed
  *                                during program execution.
  *
  *
**********************************************************
********************
  * @attention
  *
  * Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
********************
  */
/** @addtogroup CMSIS
  * @{
```

```
  */
/** @addtogroup stm32f4xx_system
  * @{
  */
/** @addtogroup STM32F4xx_System_Private_Includes
  * @{
  */
#include "stm32f4xx.h"
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    ((uint32_t)25000000) /*!< Default value of
the External oscillator in Hz */
#endif /* HSE_VALUE */
#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000) /*!< Value of the
Internal oscillator in Hz*/
#endif /* HSI_VALUE */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_TypesDefinitions
  * @{
  */
/**
  * @}
```

```
  */
/** @addtogroup STM32F4xx_System_Private_Defines
  * @{
  */
/************************* Miscellaneous Configuration ************************/
/*!< Uncomment the following line if you need to use external SRAM or SDRAM as data memory  */
#if defined(STM32F405xx) || defined(STM32F415xx) || defined(STM32F407xx) || defined(STM32F417xx)\
 || defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx) || defined(STM32F412Zx) || defined(STM32F412Vx)
/* #define DATA_IN_ExtSRAM */
#endif /* STM32F40xxx || STM32F41xxx || STM32F42xxx || STM32F43xxx || STM32F469xx || STM32F479xx ||\
        STM32F412Zx || STM32F412Vx */
#if defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F446xx) || defined(STM32F469xx) || defined(STM32F479xx)
/* #define DATA_IN_ExtSDRAM */
#endif /* STM32F427xx || STM32F437xx || STM32F429xx || STM32F439xx || STM32F446xx || STM32F469xx ||\
        STM32F479xx */
```

/* Note: Following vector table addresses must be defined in line with linker

configuration. */

/*!< Uncomment the following line if you need to relocate the vector table

anywhere in Flash or Sram, else the vector table is kept at the automatic

remap of boot address selected */

/* #define USER_VECT_TAB_ADDRESS */

#if defined(USER_VECT_TAB_ADDRESS)

/*!< Uncomment the following line if you need to relocate your vector Table

in Sram else user remap will be done in Flash. */

/* #define VECT_TAB_SRAM */

#if defined(VECT_TAB_SRAM)

#define VECT_TAB_BASE_ADDRESS SRAM_BASE /*!< Vector Table base address field.

This value must be a multiple of 0x200. */

#define VECT_TAB_OFFSET 0x00000000U /*!< Vector Table base offset field.

This value must be a multiple of 0x200. */

#else

#define VECT_TAB_BASE_ADDRESS FLASH_BASE /*!< Vector Table base address field.

```
                                        This value must be a multiple of 0x200.
*/
#define VECT_TAB_OFFSET        0x00000000U     /*!< Vector Table
base offset field.
                                        This value must be a multiple of 0x200.
*/
#endif /* VECT_TAB_SRAM */
#endif /* USER_VECT_TAB_ADDRESS */
/*****************************************************************
*********************/
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Macros
  * @{
  */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Variables
  * @{
  */
  /* This variable is updated in three ways:
     1) by calling CMSIS function SystemCoreClockUpdate()
```

2) by calling HAL API function HAL_RCC_GetHCLKFreq()

3) each time HAL_RCC_ClockConfig() is called to configure the system clock frequency

Note: If you use this function to configure the system clock; then there

is no need to call the 2 first functions listed above, since SystemCoreClock

variable is updated automatically.

```
  */
uint32_t SystemCoreClock = 16000000;
const uint8_t AHBPrescTable[16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9};
const uint8_t APBPrescTable[8]  = {0, 0, 0, 0, 1, 2, 3, 4};
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_FunctionPrototypes
  * @{
  */
#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  static void SystemInit_ExtMemCtl(void);
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */
/**
  * @}
```

```
  */
/** @addtogroup STM32F4xx_System_Private_Functions
  * @{
  */
/**
  * @brief  Setup the microcontroller system
  *         Initialize the FPU setting, vector table location and External memory
  *         configuration.
  * @param  None
  * @retval None
  */
void SystemInit(void)
{
  /* FPU settings ------------------------------------------------------------*/
  #if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
    SCB->CPACR |= ((3UL << 10*2)|(3UL << 11*2));  /* set CP10 and CP11 Full Access */
  #endif
#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  SystemInit_ExtMemCtl();
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */
  /* Configure the Vector Table location -------------------------------------*/
#if defined(USER_VECT_TAB_ADDRESS)
```

```
  SCB->VTOR = VECT_TAB_BASE_ADDRESS |
VECT_TAB_OFFSET; /* Vector Table Relocation in Internal SRAM */
#endif /* USER_VECT_TAB_ADDRESS */
}
/**
   * @brief  Update SystemCoreClock variable according to Clock
Register Values.
  *         The SystemCoreClock variable contains the core clock (HCLK),
it can
  *         be used by the user application to setup the SysTick timer or
configure
  *         other parameters.
  *
  * @note   Each time the core clock (HCLK) changes, this function
must be called
  *         to update SystemCoreClock variable value. Otherwise, any
configuration
  *         based on this variable will be incorrect.
  *
  * @note   - The system frequency computed by this function is not the
real
  *           frequency in the chip. It is calculated based on the predefined
  *           constant and the selected clock source:
  *
```

```
  *          - If SYSCLK source is HSI, SystemCoreClock will contain the
HSI_VALUE(*)
  *
  *          - If SYSCLK source is HSE, SystemCoreClock will contain the
HSE_VALUE(**)
  *
  *          - If SYSCLK source is PLL, SystemCoreClock will contain the
HSE_VALUE(**)
  *            or HSI_VALUE(*) multiplied/divided by the PLL factors.
  *
  *        (*) HSI_VALUE is a constant defined in stm32f4xx_hal_conf.h
file (default value
  *            16 MHz) but the real value may vary depending on the
variations
  *            in voltage and temperature.
  *
  *        (**) HSE_VALUE is a constant defined in
stm32f4xx_hal_conf.h file (its value
  *             depends on the application requirements), user has to ensure
that HSE_VALUE
  *             is same as the real frequency of the crystal used. Otherwise,
this function
  *             may have wrong result.
  *
  *        - The result of this function could be not correct when using
fractional
```

```
  *          value for HSE crystal.
  *
  * @param  None
  * @retval None
  */
void SystemCoreClockUpdate(void)
{
  uint32_t tmp = 0, pllvco = 0, pllp = 2, pllsource = 0, pllm = 2;
  /* Get SYSCLK source -------------------------------------------------------
*/
  tmp = RCC->CFGR & RCC_CFGR_SWS;
  switch (tmp)
  {
    case 0x00:  /* HSI used as system clock source */
      SystemCoreClock = HSI_VALUE;
      break;
    case 0x04:  /* HSE used as system clock source */
      SystemCoreClock = HSE_VALUE;
      break;
    case 0x08:  /* PLL used as system clock source */
      /* PLL_VCO = (HSE_VALUE or HSI_VALUE / PLL_M) * PLL_N
         SYSCLK = PLL_VCO / PLL_P
         */
```

```
    pllsource = (RCC->PLLCFGR & RCC_PLLCFGR_PLLSRC) >>
22;
    pllm = RCC->PLLCFGR & RCC_PLLCFGR_PLLM;
    if (pllsource != 0)
    {
      /* HSE used as PLL clock source */
      pllvco = (HSE_VALUE / pllm) * ((RCC->PLLCFGR &
RCC_PLLCFGR_PLLN) >> 6);
    }
    else
    {
      /* HSI used as PLL clock source */
      pllvco = (HSI_VALUE / pllm) * ((RCC->PLLCFGR &
RCC_PLLCFGR_PLLN) >> 6);
    }
    pllp = (((RCC->PLLCFGR & RCC_PLLCFGR_PLLP) >>16) + 1 )
*2;
    SystemCoreClock = pllvco/pllp;
    break;
   default:
    SystemCoreClock = HSI_VALUE;
    break;
  }
  /* Compute HCLK frequency ----------------------------------------------
-*/
```

```
  /* Get HCLK prescaler */
  tmp = AHBPrescTable[((RCC->CFGR & RCC_CFGR_HPRE) >> 4)];
  /* HCLK frequency */
  SystemCoreClock >>= tmp;
}
#if defined (DATA_IN_ExtSRAM) && defined
(DATA_IN_ExtSDRAM)
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
/**
  * @brief  Setup the external memory controller.
  *         Called in startup_stm32f4xx.s before jump to main.
  *         This function configures the external memories
(SRAM/SDRAM)
  *         This SRAM/SDRAM will be used as program data memory
(including heap and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
```

```
  register __IO uint32_t index;
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and
GPIOI interface clock */
  RCC->AHB1ENR |= 0x000001F8;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x00CCC0CC;
  GPIOD->AFR[1]  = 0xCCCCCCCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xAAAA0A8A;
  /* Configure PDx pins speed to 100 MHz */
  GPIOD->OSPEEDR = 0xFFFF0FCF;
  /* Configure PDx pins Output type to push-pull */
  GPIOD->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOD->PUPDR   = 0x00000000;
  /* Connect PEx pins to FMC Alternate function */
  GPIOE->AFR[0]  = 0xC00CC0CC;
  GPIOE->AFR[1]  = 0xCCCCCCCC;
  /* Configure PEx pins in Alternate function mode */
  GPIOE->MODER   = 0xAAAA828A;
  /* Configure PEx pins speed to 100 MHz */
```

```
GPIOE->OSPEEDR = 0xFFFFC3CF;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0xCCCCCCCC;
GPIOF->AFR[1]  = 0xCCCCCCCC;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA800AAA;
/* Configure PFx pins speed to 50 MHz */
GPIOF->OSPEEDR = 0xAA800AAA;
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
GPIOF->PUPDR   = 0x00000000;
/* Connect PGx pins to FMC Alternate function */
GPIOG->AFR[0]  = 0xCCCCCCCC;
GPIOG->AFR[1]  = 0xCCCCCCCC;
/* Configure PGx pins in Alternate function mode */
GPIOG->MODER   = 0xAAAAAAAA;
/* Configure PGx pins speed to 50 MHz */
GPIOG->OSPEEDR = 0xAAAAAAAA;
```

```
/* Configure PGx pins Output type to push-pull */
GPIOG->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PGx pins */
GPIOG->PUPDR   = 0x00000000;
/* Connect PHx pins to FMC Alternate function */
GPIOH->AFR[0]  = 0x00C0CC00;
GPIOH->AFR[1]  = 0xCCCCCCCC;
/* Configure PHx pins in Alternate function mode */
GPIOH->MODER   = 0xAAAA08A0;
/* Configure PHx pins speed to 50 MHz */
GPIOH->OSPEEDR = 0xAAAA08A0;
/* Configure PHx pins Output type to push-pull */
GPIOH->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PHx pins */
GPIOH->PUPDR   = 0x00000000;
/* Connect PIx pins to FMC Alternate function */
GPIOI->AFR[0]  = 0xCCCCCCCC;
GPIOI->AFR[1]  = 0x00000CC0;
/* Configure PIx pins in Alternate function mode */
GPIOI->MODER   = 0x0028AAAA;
/* Configure PIx pins speed to 50 MHz */
GPIOI->OSPEEDR = 0x0028AAAA;
/* Configure PIx pins Output type to push-pull */
```

```
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;
/*-- FMC Configuration ------------------------------------------------------*/
  /* Enable the FMC interface clock */
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
  FMC_Bank5_6->SDTR[0] = 0x01115351;
  /* SDRAM initialization sequence */
  /* Clock enable command */
  FMC_Bank5_6->SDCMR = 0x00000011;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Delay */
  for (index = 0; index<1000; index++);
  /* PALL command */
  FMC_Bank5_6->SDCMR = 0x00000012;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
```

```
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* Auto refresh command */
FMC_Bank5_6->SDCMR = 0x00000073;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* MRD register program */
FMC_Bank5_6->SDCMR = 0x00046014;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* Set refresh count */
tmpreg = FMC_Bank5_6->SDRTR;
```

```
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
  /* Disable write protection */
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F469xx || STM32F479xx */
  (void)(tmp);
}
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F469xx || STM32F479xx */
#elif defined (DATA_IN_ExtSRAM) || defined
(DATA_IN_ExtSDRAM)
```

```
/**
  * @brief  Setup the external memory controller.
  *         Called in startup_stm32f4xx.s before jump to main.
  *         This function configures the external memories
(SRAM/SDRAM)
  *         This SRAM/SDRAM will be used as program data memory
(including heap and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F446xx) || defined(STM32F469xx) ||
defined(STM32F479xx)
#if defined (DATA_IN_ExtSDRAM)
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
  register __IO uint32_t index;
#if defined(STM32F446xx)
  /* Enable GPIOA, GPIOC, GPIOD, GPIOE, GPIOF, GPIOG interface
     clock */
  RCC->AHB1ENR |= 0x0000007D;
```

```
#else
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and
GPIOI interface
     clock */
  RCC->AHB1ENR |= 0x000001F8;
#endif /* STM32F446xx */
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);
#if defined(STM32F446xx)
  /* Connect PAx pins to FMC Alternate function */
  GPIOA->AFR[0]  |= 0xC0000000;
  GPIOA->AFR[1]  |= 0x00000000;
  /* Configure PDx pins in Alternate function mode */
  GPIOA->MODER   |= 0x00008000;
  /* Configure PDx pins speed to 50 MHz */
  GPIOA->OSPEEDR |= 0x00008000;
  /* Configure PDx pins Output type to push-pull */
  GPIOA->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOA->PUPDR   |= 0x00000000;
  /* Connect PCx pins to FMC Alternate function */
  GPIOC->AFR[0]  |= 0x00CC0000;
  GPIOC->AFR[1]  |= 0x00000000;
```

```
  /* Configure PDx pins in Alternate function mode */
  GPIOC->MODER   |= 0x00000A00;
  /* Configure PDx pins speed to 50 MHz */
  GPIOC->OSPEEDR |= 0x00000A00;
  /* Configure PDx pins Output type to push-pull */
  GPIOC->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOC->PUPDR   |= 0x00000000;
#endif /* STM32F446xx */
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x000000CC;
  GPIOD->AFR[1]  = 0xCC000CCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xA02A000A;
  /* Configure PDx pins speed to 50 MHz */
  GPIOD->OSPEEDR = 0xA02A000A;
  /* Configure PDx pins Output type to push-pull */
  GPIOD->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOD->PUPDR   = 0x00000000;
  /* Connect PEx pins to FMC Alternate function */
  GPIOE->AFR[0]  = 0xC00000CC;
  GPIOE->AFR[1]  = 0xCCCCCCCC;
```

```
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA800A;
/* Configure PEx pins speed to 50 MHz */
GPIOE->OSPEEDR = 0xAAAA800A;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0xCCCCCCCC;
GPIOF->AFR[1]  = 0xCCCCCCCC;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA800AAA;
/* Configure PFx pins speed to 50 MHz */
GPIOF->OSPEEDR = 0xAA800AAA;
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
GPIOF->PUPDR   = 0x00000000;
/* Connect PGx pins to FMC Alternate function */
GPIOG->AFR[0]  = 0xCCCCCCCC;
GPIOG->AFR[1]  = 0xCCCCCCCC;
/* Configure PGx pins in Alternate function mode */
```

```
  GPIOG->MODER   = 0xAAAAAAAA;
  /* Configure PGx pins speed to 50 MHz */
  GPIOG->OSPEEDR = 0xAAAAAAAA;
  /* Configure PGx pins Output type to push-pull */
  GPIOG->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
  /* Connect PHx pins to FMC Alternate function */
  GPIOH->AFR[0]  = 0x00C0CC00;
  GPIOH->AFR[1]  = 0xCCCCCCCC;
  /* Configure PHx pins in Alternate function mode */
  GPIOH->MODER   = 0xAAAA08A0;
  /* Configure PHx pins speed to 50 MHz */
  GPIOH->OSPEEDR = 0xAAAA08A0;
  /* Configure PHx pins Output type to push-pull */
  GPIOH->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PHx pins */
  GPIOH->PUPDR   = 0x00000000;
  /* Connect PIx pins to FMC Alternate function */
  GPIOI->AFR[0]  = 0xCCCCCCCC;
```

```
  GPIOI->AFR[1]  = 0x00000CC0;
  /* Configure PIx pins in Alternate function mode */
  GPIOI->MODER   = 0x0028AAAA;
  /* Configure PIx pins speed to 50 MHz */
  GPIOI->OSPEEDR = 0x0028AAAA;
  /* Configure PIx pins Output type to push-pull */
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F469xx || STM32F479xx */
/*-- FMC Configuration ------------------------------------------------------*/
  /* Enable the FMC interface clock */
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable SDRAM bank1 */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCR[0] = 0x00001954;
#else
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
#endif /* STM32F446xx */
  FMC_Bank5_6->SDTR[0] = 0x01115351;
```

```
  /* SDRAM initialization sequence */
  /* Clock enable command */
  FMC_Bank5_6->SDCMR = 0x00000011;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Delay */
  for (index = 0; index<1000; index++);
  /* PALL command */
  FMC_Bank5_6->SDCMR = 0x00000012;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Auto refresh command */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x000000F3;
#else
  FMC_Bank5_6->SDCMR = 0x00000073;
```

```
#endif /* STM32F446xx */
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* MRD register program */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x00044014;
#else
  FMC_Bank5_6->SDCMR = 0x00046014;
#endif /* STM32F446xx */
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Set refresh count */
  tmpreg = FMC_Bank5_6->SDRTR;
#if defined(STM32F446xx)
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000050C<<1));
```

```
#else
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
#endif /* STM32F446xx */
  /* Disable write protection */
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#endif /* DATA_IN_ExtSDRAM */
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F446xx || STM32F469xx || STM32F479xx */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx) || defined(STM32F417xx)\
 || defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx) ||
defined(STM32F412Zx) || defined(STM32F412Vx)
#if defined(DATA_IN_ExtSRAM)
/*-- GPIOs Configuration -----------------------------------------------------*/
   /* Enable GPIOD, GPIOE, GPIOF and GPIOG interface clock */
  RCC->AHB1ENR   |= 0x00000078;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIODEN);
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x00CCC0CC;
  GPIOD->AFR[1]  = 0xCCCCCCCC;
```

```
/* Configure PDx pins in Alternate function mode */
GPIOD->MODER   = 0xAAAA0A8A;
/* Configure PDx pins speed to 100 MHz */
GPIOD->OSPEEDR = 0xFFFF0FCF;
/* Configure PDx pins Output type to push-pull */
GPIOD->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PDx pins */
GPIOD->PUPDR   = 0x00000000;
/* Connect PEx pins to FMC Alternate function */
GPIOE->AFR[0]  = 0xC00CC0CC;
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA828A;
/* Configure PEx pins speed to 100 MHz */
GPIOE->OSPEEDR = 0xFFFFC3CF;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0x00CCCCCC;
GPIOF->AFR[1]  = 0xCCCC0000;
/* Configure PFx pins in Alternate function mode */
```

```
  GPIOF->MODER   = 0xAA000AAA;
  /* Configure PFx pins speed to 100 MHz */
  GPIOF->OSPEEDR = 0xFF000FFF;
  /* Configure PFx pins Output type to push-pull */
  GPIOF->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PFx pins */
  GPIOF->PUPDR   = 0x00000000;
  /* Connect PGx pins to FMC Alternate function */
  GPIOG->AFR[0]  = 0x00CCCCCC;
  GPIOG->AFR[1]  = 0x000000C0;
  /* Configure PGx pins in Alternate function mode */
  GPIOG->MODER   = 0x00085AAA;
  /* Configure PGx pins speed to 100 MHz */
  GPIOG->OSPEEDR = 0x000CAFFF;
  /* Configure PGx pins Output type to push-pull */
  GPIOG->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;
/*-- FMC/FSMC Configuration ----------------------------------------------
-*/
  /* Enable the FMC/FSMC interface clock */
  RCC->AHB3ENR         |= 0x00000001;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)
```

```
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F469xx || STM32F479xx */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx)|| defined(STM32F417xx)\
   || defined(STM32F412Zx) || defined(STM32F412Vx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FSMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FSMC_Bank1->BTCR[2]  = 0x00001011;
  FSMC_Bank1->BTCR[3]  = 0x00000201;
```

```
  FSMC_Bank1E->BWTR[2] = 0x0FFFFFFF;
#endif /* STM32F405xx || STM32F415xx || STM32F407xx ||
STM32F417xx || STM32F412Zx || STM32F412Vx */
#endif /* DATA_IN_ExtSRAM */
#endif /* STM32F405xx || STM32F415xx || STM32F407xx ||
STM32F417xx || STM32F427xx || STM32F437xx ||\
        STM32F429xx || STM32F439xx || STM32F469xx ||
STM32F479xx || STM32F412Zx || STM32F412Vx  */
  (void)(tmp);
}
#endif /* DATA_IN_ExtSRAM && DATA_IN_ExtSDRAM */
/**
  * @}
  */
/**
  * @}
  */
/**
  * @}
  */
```

CHAPTER#23 CONVERTING KEIL MDK ARM ASSEMBLY SYNTAX TO GCC ASSEMBLY SYNTAX

23.1CONVERTING KEIL MDK ARM ASSEMBLY SYNTAX TO GCC ASSEMBLY SYNTAX:

Continuing from a previous discussion, here's a follow-up on coding GPIO drivers in assembly language, focusing on setting up and controlling GPIO pins. This assumes that you are working with a microcontroller that uses ARM Cortex-M architecture, which is common in many microcontrollers.

23.2 CONFIGURING GPIO PINS IN ASSEMBLY:

23.2.1 DEFINE GPIO REGISTER ADDRESSES:

Typically, GPIO registers are located at specific memory addresses. For ARM Cortex-M microcontrollers, these addresses are often specified in the reference manual.

23.2.2 ENABLE GPIO PINS:

Before configuring GPIO pins, ensure that the peripheral clock for the GPIO port is enabled.

23.2.3 SET GPIO MODES:

GPIO pins can function in various modes, such as input, output, analog, or alternate function. The mode is set by configuring the GPIOx_MODER register.

23.2.4 SET GPIO OUTPUT TYPE:

For output pins, configure whether they are push-pull or open-drain using the GPIOx_OTYPER register.

23.2.5 SET GPIO SPEED:

The output speed of GPIO pins can be adjusted through the GPIOx_OSPEEDR register.

23.2.6 SET GPIO PULL-UP/PULL-DOWN:

To configure internal pull-up or pull-down resistors, use the GPIOx_PUPDR register.

23.3 CODE OF MAIN.H:

```
/* USER CODE BEGIN Header */
/**
**********************************************************
*******************
  * @file           : main.h
  * @brief          : Header for main.c file.
  *                   This file contains the common defines of the application.
**********************************************************
*******************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
```

```
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H
#ifdef __cplusplus
extern "C" {
#endif
/* Includes ------------------------------------------------------------------*/
#include "stm32f4xx_hal.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants --------------------------------------------------------*/
```

```
/* USER CODE BEGIN EC */
/* USER CODE END EC */
/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */
/* USER CODE END EM */
/* Exported functions prototypes ---------------------------------------------*/
void Error_Handler(void);
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
/* Private defines -----------------------------------------------------------*/
#define B1_Pin GPIO_PIN_13
#define B1_GPIO_Port GPIOC
#define USART_TX_Pin GPIO_PIN_2
#define USART_TX_GPIO_Port GPIOA
#define USART_RX_Pin GPIO_PIN_3
#define USART_RX_GPIO_Port GPIOA
#define LD2_Pin GPIO_PIN_5
#define LD2_GPIO_Port GPIOA
#define TMS_Pin GPIO_PIN_13
#define TMS_GPIO_Port GPIOA
#define TCK_Pin GPIO_PIN_14
#define TCK_GPIO_Port GPIOA
#define SWO_Pin GPIO_PIN_3
```

```
#define SWO_GPIO_Port GPIOB
/* USER CODE BEGIN Private defines */
/* USER CODE END Private defines */
#ifdef __cplusplus
}
#endif
#endif /* __MAIN_H */
```

23.4 CODE OF STM32F4XX_HAL_CONF.H:

```
/* USER CODE BEGIN Header */
/**
*******************************************************************
  * @file    stm32f4xx_hal_conf_template.h
  * @author  MCD Application Team
  * @brief   HAL configuration template file.
  *          This file should be copied to the application folder and renamed
  *          to stm32f4xx_hal_conf.h.
*******************************************************************
  * @attention
  *
  * Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.
```

```
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_HAL_CONF_H
#define __STM32F4xx_HAL_CONF_H
#ifdef __cplusplus
 extern "C" {
#endif
/* Exported types ------------------------------------------------------------*/
/* Exported constants --------------------------------------------------------*/
/* ########################### Module Selection
############################### */
/**
  * @brief This is the list of modules to be used in the HAL driver
  */
#define HAL_MODULE_ENABLED
```

```
  /* #define HAL_CRYP_MODULE_ENABLED */
/* #define HAL_ADC_MODULE_ENABLED */
/* #define HAL_CAN_MODULE_ENABLED */
/* #define HAL_CRC_MODULE_ENABLED */
/* #define HAL_CAN_LEGACY_MODULE_ENABLED */
/* #define HAL_DAC_MODULE_ENABLED */
/* #define HAL_DCMI_MODULE_ENABLED */
/* #define HAL_DMA2D_MODULE_ENABLED */
/* #define HAL_ETH_MODULE_ENABLED */
/* #define HAL_ETH_LEGACY_MODULE_ENABLED */
/* #define HAL_NAND_MODULE_ENABLED */
/* #define HAL_NOR_MODULE_ENABLED */
/* #define HAL_PCCARD_MODULE_ENABLED */
/* #define HAL_SRAM_MODULE_ENABLED */
/* #define HAL_SDRAM_MODULE_ENABLED */
/* #define HAL_HASH_MODULE_ENABLED */
/* #define HAL_I2C_MODULE_ENABLED */
/* #define HAL_I2S_MODULE_ENABLED */
/* #define HAL_IWDG_MODULE_ENABLED */
/* #define HAL_LTDC_MODULE_ENABLED */
/* #define HAL_RNG_MODULE_ENABLED */
/* #define HAL_RTC_MODULE_ENABLED */
/* #define HAL_SAI_MODULE_ENABLED */
```

```
/* #define HAL_SD_MODULE_ENABLED */
/* #define HAL_MMC_MODULE_ENABLED */
/* #define HAL_SPI_MODULE_ENABLED */
/* #define HAL_TIM_MODULE_ENABLED */
#define HAL_UART_MODULE_ENABLED
/* #define HAL_USART_MODULE_ENABLED */
/* #define HAL_IRDA_MODULE_ENABLED */
/* #define HAL_SMARTCARD_MODULE_ENABLED */
/* #define HAL_SMBUS_MODULE_ENABLED */
/* #define HAL_WWDG_MODULE_ENABLED */
/* #define HAL_PCD_MODULE_ENABLED */
/* #define HAL_HCD_MODULE_ENABLED */
/* #define HAL_DSI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_CEC_MODULE_ENABLED */
/* #define HAL_FMPI2C_MODULE_ENABLED */
/* #define HAL_FMPSMBUS_MODULE_ENABLED */
/* #define HAL_SPDIFRX_MODULE_ENABLED */
/* #define HAL_DFSDM_MODULE_ENABLED */
/* #define HAL_LPTIM_MODULE_ENABLED */
#define HAL_GPIO_MODULE_ENABLED
#define HAL_EXTI_MODULE_ENABLED
```

```
#define HAL_DMA_MODULE_ENABLED
#define HAL_RCC_MODULE_ENABLED
#define HAL_FLASH_MODULE_ENABLED
#define HAL_PWR_MODULE_ENABLED
#define HAL_CORTEX_MODULE_ENABLED
/* ############################ HSE/HSI Values adaptation
###################### */
/**
  * @brief Adjust the value of External High Speed oscillator (HSE)
used in your application.
  *        This value is used by the RCC HAL module to compute the
system frequency
  *        (when HSE is used as system clock source, directly or through
the PLL).
  */
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    8000000U /*!< Value of the External
oscillator in Hz */
#endif /* HSE_VALUE */
#if !defined  (HSE_STARTUP_TIMEOUT)
  #define HSE_STARTUP_TIMEOUT    100U   /*!< Time out for
HSE start up, in ms */
#endif /* HSE_STARTUP_TIMEOUT */
/**
  * @brief Internal High Speed oscillator (HSI) value.
```

```
  *       This value is used by the RCC HAL module to compute the
system frequency
  *       (when HSI is used as system clock source, directly or through the
PLL).
  */
#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000U) /*!< Value of the
Internal oscillator in Hz*/
#endif /* HSI_VALUE */
/**
  * @brief Internal Low Speed oscillator (LSI) value.
  */
#if !defined  (LSI_VALUE)
 #define LSI_VALUE  32000U       /*!< LSI Typical Value in Hz*/
#endif /* LSI_VALUE */                    /*!< Value of the Internal Low
Speed oscillator in Hz
                                  The real value may vary depending on the
variations
                                  in voltage and temperature.*/
/**
  * @brief External Low Speed oscillator (LSE) value.
  */
#if !defined  (LSE_VALUE)
 #define LSE_VALUE  32768U    /*!< Value of the External Low Speed
oscillator in Hz */
```

```
#endif /* LSE_VALUE */
#if !defined  (LSE_STARTUP_TIMEOUT)
  #define LSE_STARTUP_TIMEOUT    5000U   /*!< Time out for
LSE start up, in ms */
#endif /* LSE_STARTUP_TIMEOUT */
/**
  * @brief External clock source for I2S peripheral
  *        This value is used by the I2S HAL module to compute the I2S
clock source
  *        frequency, this source is inserted directly through I2S_CKIN
pad.
  */
#if !defined  (EXTERNAL_CLOCK_VALUE)
  #define EXTERNAL_CLOCK_VALUE    12288000U /*!< Value of
the External audio frequency in Hz*/
#endif /* EXTERNAL_CLOCK_VALUE */
/* Tip: To avoid modifying this file each time you need to use different
HSE,
   ===  you can define the HSE value in your toolchain compiler
preprocessor. */
/* ############################## System Configuration
########################### */
/**
  * @brief This is the HAL system configuration section
  */
```

```
#define  VDD_VALUE                    3300U /*!< Value of VDD in mv */
#define  TICK_INT_PRIORITY            0U   /*!< tick interrupt
priority */
#define  USE_RTOS                     0U
#define  PREFETCH_ENABLE              1U
#define  INSTRUCTION_CACHE_ENABLE     1U
#define  DATA_CACHE_ENABLE            1U
#define  USE_HAL_ADC_REGISTER_CALLBACKS        0U /*
ADC register callback disabled       */
#define  USE_HAL_CAN_REGISTER_CALLBACKS        0U /*
CAN register callback disabled       */
#define  USE_HAL_CEC_REGISTER_CALLBACKS        0U /*
CEC register callback disabled       */
#define  USE_HAL_CRYP_REGISTER_CALLBACKS       0U /*
CRYP register callback disabled      */
#define  USE_HAL_DAC_REGISTER_CALLBACKS        0U /*
DAC register callback disabled       */
#define  USE_HAL_DCMI_REGISTER_CALLBACKS       0U /*
DCMI register callback disabled      */
#define  USE_HAL_DFSDM_REGISTER_CALLBACKS      0U /*
DFSDM register callback disabled     */
#define  USE_HAL_DMA2D_REGISTER_CALLBACKS      0U /*
DMA2D register callback disabled     */
#define  USE_HAL_DSI_REGISTER_CALLBACKS        0U /* DSI
register callback disabled       */
```

#define USE_HAL_ETH_REGISTER_CALLBACKS 0U /* ETH register callback disabled */

#define USE_HAL_HASH_REGISTER_CALLBACKS 0U /* HASH register callback disabled */

#define USE_HAL_HCD_REGISTER_CALLBACKS 0U /* HCD register callback disabled */

#define USE_HAL_I2C_REGISTER_CALLBACKS 0U /* I2C register callback disabled */

#define USE_HAL_FMPI2C_REGISTER_CALLBACKS 0U /* FMPI2C register callback disabled */

#define USE_HAL_FMPSMBUS_REGISTER_CALLBACKS 0U /* FMPSMBUS register callback disabled */

#define USE_HAL_I2S_REGISTER_CALLBACKS 0U /* I2S register callback disabled */

#define USE_HAL_IRDA_REGISTER_CALLBACKS 0U /* IRDA register callback disabled */

#define USE_HAL_LPTIM_REGISTER_CALLBACKS 0U /* LPTIM register callback disabled */

#define USE_HAL_LTDC_REGISTER_CALLBACKS 0U /* LTDC register callback disabled */

#define USE_HAL_MMC_REGISTER_CALLBACKS 0U /* MMC register callback disabled */

#define USE_HAL_NAND_REGISTER_CALLBACKS 0U /* NAND register callback disabled */

#define USE_HAL_NOR_REGISTER_CALLBACKS 0U /* NOR register callback disabled */

#define USE_HAL_PCCARD_REGISTER_CALLBACKS 0U /* PCCARD register callback disabled */

#define USE_HAL_PCD_REGISTER_CALLBACKS 0U /* PCD register callback disabled */

#define USE_HAL_QSPI_REGISTER_CALLBACKS 0U /* QSPI register callback disabled */

#define USE_HAL_RNG_REGISTER_CALLBACKS 0U /* RNG register callback disabled */

#define USE_HAL_RTC_REGISTER_CALLBACKS 0U /* RTC register callback disabled */

#define USE_HAL_SAI_REGISTER_CALLBACKS 0U /* SAI register callback disabled */

#define USE_HAL_SD_REGISTER_CALLBACKS 0U /* SD register callback disabled */

#define USE_HAL_SMARTCARD_REGISTER_CALLBACKS 0U /* SMARTCARD register callback disabled */

#define USE_HAL_SDRAM_REGISTER_CALLBACKS 0U /* SDRAM register callback disabled */

#define USE_HAL_SRAM_REGISTER_CALLBACKS 0U /* SRAM register callback disabled */

#define USE_HAL_SPDIFRX_REGISTER_CALLBACKS 0U /* SPDIFRX register callback disabled */

#define USE_HAL_SMBUS_REGISTER_CALLBACKS 0U /* SMBUS register callback disabled */

#define USE_HAL_SPI_REGISTER_CALLBACKS 0U /* SPI register callback disabled */

```
#define  USE_HAL_TIM_REGISTER_CALLBACKS        0U /*
TIM register callback disabled       */

#define  USE_HAL_UART_REGISTER_CALLBACKS       0U /*
UART register callback disabled      */

#define  USE_HAL_USART_REGISTER_CALLBACKS      0U /*
USART register callback disabled     */

#define  USE_HAL_WWDG_REGISTER_CALLBACKS       0U /*
WWDG register callback disabled      */

/* ############################ Assert Selection
################################ */

/**

  * @brief Uncomment the line below to expanse the "assert_param"
macro in the

  *        HAL drivers code

  */

/* #define USE_FULL_ASSERT    1U */

/* ################### Ethernet peripheral configuration
###################### */

/* Section 1 : Ethernet peripheral configuration */

/* MAC ADDRESS:
MAC_ADDR0:MAC_ADDR1:MAC_ADDR2:MAC_ADDR3:MAC_A
DDR4:MAC_ADDR5 */

#define MAC_ADDR0   2U

#define MAC_ADDR1   0U

#define MAC_ADDR2   0U

#define MAC_ADDR3   0U
```

```
#define MAC_ADDR4   0U
#define MAC_ADDR5   0U
/* Definition of the Ethernet driver buffers size and count */
#define ETH_RX_BUF_SIZE             ETH_MAX_PACKET_SIZE
/* buffer size for receive             */
#define ETH_TX_BUF_SIZE             ETH_MAX_PACKET_SIZE
/* buffer size for transmit            */
#define ETH_RXBUFNB                 4U      /* 4 Rx buffers of size
ETH_RX_BUF_SIZE  */
#define ETH_TXBUFNB                 4U      /* 4 Tx buffers of size
ETH_TX_BUF_SIZE  */
/* Section 2: PHY configuration section */
/* DP83848_PHY_ADDRESS Address*/
#define DP83848_PHY_ADDRESS
/* PHY Reset delay these values are based on a 1 ms Systick interrupt*/
#define PHY_RESET_DELAY             0x000000FFU
/* PHY Configuration delay */
#define PHY_CONFIG_DELAY            0x00000FFFU
#define PHY_READ_TO                 0x0000FFFFU
#define PHY_WRITE_TO                0x0000FFFFU
/* Section 3: Common PHY Registers */
#define PHY_BCR                     ((uint16_t)0x0000U)    /*!<
Transceiver Basic Control Register   */
#define PHY_BSR                     ((uint16_t)0x0001U)    /*!<
Transceiver Basic Status Register    */
```

```
#define PHY_RESET                    ((uint16_t)0x8000U)  /*!< PHY
Reset */

#define PHY_LOOPBACK                 ((uint16_t)0x4000U)  /*!<
Select loop-back mode */

#define PHY_FULLDUPLEX_100M          ((uint16_t)0x2100U)
/*!< Set the full-duplex mode at 100 Mb/s */

#define PHY_HALFDUPLEX_100M          ((uint16_t)0x2000U)
/*!< Set the half-duplex mode at 100 Mb/s */

#define PHY_FULLDUPLEX_10M           ((uint16_t)0x0100U)
/*!< Set the full-duplex mode at 10 Mb/s  */

#define PHY_HALFDUPLEX_10M           ((uint16_t)0x0000U)
/*!< Set the half-duplex mode at 10 Mb/s  */

#define PHY_AUTONEGOTIATION          ((uint16_t)0x1000U)
/*!< Enable auto-negotiation function     */

#define PHY_RESTART_AUTONEGOTIATION
((uint16_t)0x0200U)  /*!< Restart auto-negotiation function    */

#define PHY_POWERDOWN                ((uint16_t)0x0800U)  /*!<
Select the power down mode           */

#define PHY_ISOLATE                  ((uint16_t)0x0400U)  /*!< Isolate
PHY from MII                 */

#define PHY_AUTONEGO_COMPLETE        ((uint16_t)0x0020U)
/*!< Auto-Negotiation process completed   */

#define PHY_LINKED_STATUS            ((uint16_t)0x0004U)  /*!<
Valid link established               */

#define PHY_JABBER_DETECTION         ((uint16_t)0x0002U)
/*!< Jabber condition detected            */

/* Section 4: Extended PHY Registers */
```

```
#define PHY_SR                    ((uint16_t))    /*!< PHY status register
Offset                  */

#define PHY_SPEED_STATUS              ((uint16_t))  /*!< PHY Speed
mask                          */

#define PHY_DUPLEX_STATUS             ((uint16_t))  /*!< PHY
Duplex mask                           */

/* ################### SPI peripheral configuration
############################ */

/* CRC FEATURE: Use to activate CRC feature inside HAL SPI Driver

* Activated: CRC code is present inside driver

* Deactivated: CRC code cleaned from driver

*/

#define USE_SPI_CRC                   0U

/* Includes ------------------------------------------------------------------*/

/**

  * @brief Include module's header file

  */

#ifdef HAL_RCC_MODULE_ENABLED

  #include "stm32f4xx_hal_rcc.h"

#endif /* HAL_RCC_MODULE_ENABLED */

#ifdef HAL_GPIO_MODULE_ENABLED

  #include "stm32f4xx_hal_gpio.h"

#endif /* HAL_GPIO_MODULE_ENABLED */

#ifdef HAL_EXTI_MODULE_ENABLED
```

```
  #include "stm32f4xx_hal_exti.h"
#endif /* HAL_EXTI_MODULE_ENABLED */
#ifdef HAL_DMA_MODULE_ENABLED
  #include "stm32f4xx_hal_dma.h"
#endif /* HAL_DMA_MODULE_ENABLED */
#ifdef HAL_CORTEX_MODULE_ENABLED
  #include "stm32f4xx_hal_cortex.h"
#endif /* HAL_CORTEX_MODULE_ENABLED */
#ifdef HAL_ADC_MODULE_ENABLED
  #include "stm32f4xx_hal_adc.h"
#endif /* HAL_ADC_MODULE_ENABLED */
#ifdef HAL_CAN_MODULE_ENABLED
  #include "stm32f4xx_hal_can.h"
#endif /* HAL_CAN_MODULE_ENABLED */
#ifdef HAL_CAN_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_can_legacy.h"
#endif /* HAL_CAN_LEGACY_MODULE_ENABLED */
#ifdef HAL_CRC_MODULE_ENABLED
  #include "stm32f4xx_hal_crc.h"
#endif /* HAL_CRC_MODULE_ENABLED */
#ifdef HAL_CRYP_MODULE_ENABLED
  #include "stm32f4xx_hal_cryp.h"
#endif /* HAL_CRYP_MODULE_ENABLED */
```

```
#ifdef HAL_DMA2D_MODULE_ENABLED
  #include "stm32f4xx_hal_dma2d.h"
#endif /* HAL_DMA2D_MODULE_ENABLED */
#ifdef HAL_DAC_MODULE_ENABLED
  #include "stm32f4xx_hal_dac.h"
#endif /* HAL_DAC_MODULE_ENABLED */
#ifdef HAL_DCMI_MODULE_ENABLED
  #include "stm32f4xx_hal_dcmi.h"
#endif /* HAL_DCMI_MODULE_ENABLED */
#ifdef HAL_ETH_MODULE_ENABLED
  #include "stm32f4xx_hal_eth.h"
#endif /* HAL_ETH_MODULE_ENABLED */
#ifdef HAL_ETH_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_eth_legacy.h"
#endif /* HAL_ETH_LEGACY_MODULE_ENABLED */
#ifdef HAL_FLASH_MODULE_ENABLED
  #include "stm32f4xx_hal_flash.h"
#endif /* HAL_FLASH_MODULE_ENABLED */
#ifdef HAL_SRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sram.h"
#endif /* HAL_SRAM_MODULE_ENABLED */
#ifdef HAL_NOR_MODULE_ENABLED
  #include "stm32f4xx_hal_nor.h"
```

```
#endif /* HAL_NOR_MODULE_ENABLED */
#ifdef HAL_NAND_MODULE_ENABLED
  #include "stm32f4xx_hal_nand.h"
#endif /* HAL_NAND_MODULE_ENABLED */
#ifdef HAL_PCCARD_MODULE_ENABLED
  #include "stm32f4xx_hal_pccard.h"
#endif /* HAL_PCCARD_MODULE_ENABLED */
#ifdef HAL_SDRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sdram.h"
#endif /* HAL_SDRAM_MODULE_ENABLED */
#ifdef HAL_HASH_MODULE_ENABLED
 #include "stm32f4xx_hal_hash.h"
#endif /* HAL_HASH_MODULE_ENABLED */
#ifdef HAL_I2C_MODULE_ENABLED
 #include "stm32f4xx_hal_i2c.h"
#endif /* HAL_I2C_MODULE_ENABLED */
#ifdef HAL_SMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_smbus.h"
#endif /* HAL_SMBUS_MODULE_ENABLED */
#ifdef HAL_I2S_MODULE_ENABLED
 #include "stm32f4xx_hal_i2s.h"
#endif /* HAL_I2S_MODULE_ENABLED */
#ifdef HAL_IWDG_MODULE_ENABLED
```

```
 #include "stm32f4xx_hal_iwdg.h"
#endif /* HAL_IWDG_MODULE_ENABLED */
#ifdef HAL_LTDC_MODULE_ENABLED
 #include "stm32f4xx_hal_ltdc.h"
#endif /* HAL_LTDC_MODULE_ENABLED */
#ifdef HAL_PWR_MODULE_ENABLED
 #include "stm32f4xx_hal_pwr.h"
#endif /* HAL_PWR_MODULE_ENABLED */
#ifdef HAL_RNG_MODULE_ENABLED
 #include "stm32f4xx_hal_rng.h"
#endif /* HAL_RNG_MODULE_ENABLED */
#ifdef HAL_RTC_MODULE_ENABLED
 #include "stm32f4xx_hal_rtc.h"
#endif /* HAL_RTC_MODULE_ENABLED */
#ifdef HAL_SAI_MODULE_ENABLED
 #include "stm32f4xx_hal_sai.h"
#endif /* HAL_SAI_MODULE_ENABLED */
#ifdef HAL_SD_MODULE_ENABLED
 #include "stm32f4xx_hal_sd.h"
#endif /* HAL_SD_MODULE_ENABLED */
#ifdef HAL_SPI_MODULE_ENABLED
 #include "stm32f4xx_hal_spi.h"
#endif /* HAL_SPI_MODULE_ENABLED */
```

```
#ifdef HAL_TIM_MODULE_ENABLED
 #include "stm32f4xx_hal_tim.h"
#endif /* HAL_TIM_MODULE_ENABLED */
#ifdef HAL_UART_MODULE_ENABLED
 #include "stm32f4xx_hal_uart.h"
#endif /* HAL_UART_MODULE_ENABLED */
#ifdef HAL_USART_MODULE_ENABLED
 #include "stm32f4xx_hal_usart.h"
#endif /* HAL_USART_MODULE_ENABLED */
#ifdef HAL_IRDA_MODULE_ENABLED
 #include "stm32f4xx_hal_irda.h"
#endif /* HAL_IRDA_MODULE_ENABLED */
#ifdef HAL_SMARTCARD_MODULE_ENABLED
 #include "stm32f4xx_hal_smartcard.h"
#endif /* HAL_SMARTCARD_MODULE_ENABLED */
#ifdef HAL_WWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_wwdg.h"
#endif /* HAL_WWDG_MODULE_ENABLED */
#ifdef HAL_PCD_MODULE_ENABLED
 #include "stm32f4xx_hal_pcd.h"
#endif /* HAL_PCD_MODULE_ENABLED */
#ifdef HAL_HCD_MODULE_ENABLED
 #include "stm32f4xx_hal_hcd.h"
```

```
#endif /* HAL_HCD_MODULE_ENABLED */
#ifdef HAL_DSI_MODULE_ENABLED
 #include "stm32f4xx_hal_dsi.h"
#endif /* HAL_DSI_MODULE_ENABLED */
#ifdef HAL_QSPI_MODULE_ENABLED
 #include "stm32f4xx_hal_qspi.h"
#endif /* HAL_QSPI_MODULE_ENABLED */
#ifdef HAL_CEC_MODULE_ENABLED
 #include "stm32f4xx_hal_cec.h"
#endif /* HAL_CEC_MODULE_ENABLED */
#ifdef HAL_FMPI2C_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpi2c.h"
#endif /* HAL_FMPI2C_MODULE_ENABLED */
#ifdef HAL_FMPSMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpsmbus.h"
#endif /* HAL_FMPSMBUS_MODULE_ENABLED */
#ifdef HAL_SPDIFRX_MODULE_ENABLED
 #include "stm32f4xx_hal_spdifrx.h"
#endif /* HAL_SPDIFRX_MODULE_ENABLED */
#ifdef HAL_DFSDM_MODULE_ENABLED
 #include "stm32f4xx_hal_dfsdm.h"
#endif /* HAL_DFSDM_MODULE_ENABLED */
#ifdef HAL_LPTIM_MODULE_ENABLED
```

```
 #include "stm32f4xx_hal_lptim.h"
#endif /* HAL_LPTIM_MODULE_ENABLED */
#ifdef HAL_MMC_MODULE_ENABLED
 #include "stm32f4xx_hal_mmc.h"
#endif /* HAL_MMC_MODULE_ENABLED */
/* Exported macro ------------------------------------------------------------*/
#ifdef  USE_FULL_ASSERT
/**
  * @brief  The assert_param macro is used for function's parameters check.
  * @param  expr If expr is false, it calls assert_failed function
  *         which reports the name of the source file and the source
  *         line number of the call that failed.
  *         If expr is true, it returns no value.
  * @retval None
  */
  #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t *)__FILE__, __LINE__))
/* Exported functions ------------------------------------------------------- */
  void assert_failed(uint8_t* file, uint32_t line);
#else
  #define assert_param(expr) ((void)0U)
#endif /* USE_FULL_ASSERT */
#ifdef __cplusplus
```

```
}
#endif
#endif /* __STM32F4xx_HAL_CONF_H */
```

23.5 CODE OF STM32F4XX_IT.H:

```
/* USER CODE BEGIN Header */
/**
*******************************************************************
  * @file    stm32f4xx_it.h
  * @brief   This file contains the headers of the interrupt handlers.
*******************************************************************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*******************************************************************
  */
```

```
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_IT_H
#define __STM32F4xx_IT_H
#ifdef __cplusplus
 extern "C" {
#endif
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */
/* USER CODE END EC */
/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */
/* USER CODE END EM */
/* Exported functions prototypes ---------------------------------------------*/
void NMI_Handler(void);
void HardFault_Handler(void);
void MemManage_Handler(void);
```

```
void BusFault_Handler(void);
void UsageFault_Handler(void);
void SVC_Handler(void);
void DebugMon_Handler(void);
void PendSV_Handler(void);
void SysTick_Handler(void);
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
#ifdef __cplusplus
}
#endif
#endif /* __STM32F4xx_IT_H */
```

23.6 CODE OF MAIN.C:

```
/* USER CODE BEGIN Header */
/**
*******************************************************************
  * @file           : main.c
  * @brief          : Main program body
*******************************************************************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
```

```
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
/* USER CODE END PTD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
```

```
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
UART_HandleTypeDef huart2;
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/**
  * @brief  The application entry point.
  * @retval int
 */
int main(void)
{
  /* USER CODE BEGIN 1 */
  /* USER CODE END 1 */
  /* MCU Configuration--------------------------------------------------------*/
```

```
  /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
  HAL_Init();
  /* USER CODE BEGIN Init */
  /* USER CODE END Init */
  /* Configure the system clock */
  SystemClock_Config();
  /* USER CODE BEGIN SysInit */
  /* USER CODE END SysInit */
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  /* USER CODE BEGIN 2 */
  /* USER CODE END 2 */
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

```
/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_
VOLTAGE_SCALE1);
  /** Initializes the RCC Oscillators according to the specified
parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue =
RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 16;
```

```
  RCC_OscInitStruct.PLL.PLLN = 336;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV4;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource =
RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct,
FLASH_LATENCY_2) != HAL_OK)
  {
    Error_Handler();
  }
}
/**
```

```
  * @brief USART2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART2_UART_Init(void)
{
  /* USER CODE BEGIN USART2_Init 0 */
  /* USER CODE END USART2_Init 0 */
  /* USER CODE BEGIN USART2_Init 1 */
  /* USER CODE END USART2_Init 1 */
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 115200;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART2_Init 2 */
```

```
  /* USER CODE END USART2_Init 2 */
}
/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */
  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin,
GPIO_PIN_RESET);
  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
```

```
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);
  /*Configure GPIO pin : LD2_Pin */
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);
/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}
/* USER CODE BEGIN 4 */
/* USER CODE END 4 */
/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return
state */
  __disable_irq();
```

```
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}
#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

23.7 CODE OF MAIN.S:

```
; Assuming the use of STM32 microcontroller and GPIO port A

; Base address of GPIOA

.equ GPIOA_BASE, 0x48000000

; Register offsets

.equ GPIOA_MODER_OFFSET, 0x00  ; Mode register offset

.equ GPIOA_OTYPER_OFFSET, 0x04 ; Output type register offset

.equ GPIOA_OSPEEDR_OFFSET, 0x08 ; Output speed register offset

.equ GPIOA_PUPDR_OFFSET, 0x0C  ; Pull-up/pull-down register
offset

; Clock enable register (specific to microcontroller family)

.equ RCC_AHBENR, 0x40021014     ; RCC AHB peripheral clock
enable register

.equ RCC_IOPAEN, 0x00020000     ; Bit for enabling GPIOA clock

; Addresses

.equ GPIOA_MODER, GPIOA_BASE + GPIOA_MODER_OFFSET

.equ GPIOA_OTYPER, GPIOA_BASE + GPIOA_OTYPER_OFFSET

.equ GPIOA_OSPEEDR, GPIOA_BASE +
GPIOA_OSPEEDR_OFFSET

.equ GPIOA_PUPDR, GPIOA_BASE + GPIOA_PUPDR_OFFSET

.global main

.syntax unified

main:

    ; Enable clock for GPIOA
```

```
LDR R0, =RCC_AHBENR
LDR R1, [R0]            ; Load current RCC_AHBENR value
ORR R1, R1, #RCC_IOPAEN     ; Set bit for GPIOA clock enable
STR R1, [R0]            ; Write back to RCC_AHBENR
; Configure GPIOA pin 0 as output
LDR R0, =GPIOA_MODER
LDR R1, [R0]            ; Load current GPIOA_MODER value
BIC R1, R1, #(0x3 << (0 * 2)); Clear mode bits for pin 0
ORR R1, R1, #(0x1 << (0 * 2)); Set mode bits to output (01)
STR R1, [R0]            ; Write back to GPIOA_MODER
; Configure GPIOA pin 0 as push-pull (default, so no need to set
GPIOA_OTYPER)
; Set GPIO speed to medium for pin 0 (01)
LDR R0, =GPIOA_OSPEEDR
LDR R1, [R0]            ; Load current GPIOA_OSPEEDR value
BIC R1, R1, #(0x3 << (0 * 2)); Clear speed bits for pin 0
ORR R1, R1, #(0x1 << (0 * 2)); Set speed bits to medium speed (01)
STR R1, [R0]            ; Write back to GPIOA_OSPEEDR
; Disable pull-up/pull-down for pin 0 (00)
LDR R0, =GPIOA_PUPDR
LDR R1, [R0]            ; Load current GPIOA_PUPDR value
BIC R1, R1, #(0x3 << (0 * 2)); Clear pull-up/pull-down bits for pin 0
STR R1, [R0]            ; Write back to GPIOA_PUPDR
```

```
    ; End of configuration
    B . ; Infinite loop to keep the program running
```

23.8 CODE OF STM32F4XX_HAL_MSP.C:

```
/* USER CODE BEGIN Header */
/**
*****************************************************************************
  * @file        stm32f4xx_hal_msp.c
  * @brief       This file provides code for the MSP Initialization
  *              and de-Initialization codes.
*****************************************************************************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*****************************************************************************
  */
```

```
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN TD */
/* USER CODE END TD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN Define */
/* USER CODE END Define */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN Macro */
/* USER CODE END Macro */
/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* External functions --------------------------------------------------------*/
/* USER CODE BEGIN ExternalFunctions */
/* USER CODE END ExternalFunctions */
```

```
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/**
  * Initializes the Global MSP.
  */
void HAL_MspInit(void)
{
  /* USER CODE BEGIN MspInit 0 */
  /* USER CODE END MspInit 0 */
  __HAL_RCC_SYSCFG_CLK_ENABLE();
  __HAL_RCC_PWR_CLK_ENABLE();
  HAL_NVIC_SetPriorityGrouping(NVIC_PRIORITYGROUP_0);
  /* System interrupt init*/
  /* USER CODE BEGIN MspInit 1 */
  /* USER CODE END MspInit 1 */
}
/**
* @brief UART MSP Initialization
* This function configures the hardware resources used in this example
* @param huart: UART handle pointer
* @retval None
*/
void HAL_UART_MspInit(UART_HandleTypeDef* huart)
```

```
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  if(huart->Instance==USART2)
  {
  /* USER CODE BEGIN USART2_MspInit 0 */
  /* USER CODE END USART2_MspInit 0 */
    /* Peripheral clock enable */
    __HAL_RCC_USART2_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();
    /**USART2 GPIO Configuration
    PA2     ------> USART2_TX
    PA3     ------> USART2_RX
    */
    GPIO_InitStruct.Pin = USART_TX_Pin|USART_RX_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF7_USART2;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
  /* USER CODE BEGIN USART2_MspInit 1 */
  /* USER CODE END USART2_MspInit 1 */
  }
}
```

```
/**
* @brief UART MSP De-Initialization
* This function freeze the hardware resources used in this example
* @param huart: UART handle pointer
* @retval None
*/
void HAL_UART_MspDeInit(UART_HandleTypeDef* huart)
{
  if(huart->Instance==USART2)
  {
  /* USER CODE BEGIN USART2_MspDeInit 0 */
  /* USER CODE END USART2_MspDeInit 0 */
    /* Peripheral clock disable */
    __HAL_RCC_USART2_CLK_DISABLE();
    /**USART2 GPIO Configuration
    PA2     ------> USART2_TX
    PA3     ------> USART2_RX
    */
    HAL_GPIO_DeInit(GPIOA, USART_TX_Pin|USART_RX_Pin);
  /* USER CODE BEGIN USART2_MspDeInit 1 */
  /* USER CODE END USART2_MspDeInit 1 */
  }
}
```

```
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

23.9 CODE OF STM32F4XX_IT.C:

```
/* USER CODE BEGIN Header */
/**
*******************************************************
*******************
  * @file    stm32f4xx_it.c
  * @brief   Interrupt Service Routines.
*******************************************************
*******************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*******************************************************
*******************
  */
/* USER CODE END Header */
```

```
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stm32f4xx_it.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN TD */
/* USER CODE END TD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
/* USER CODE END PV *
/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
```

```
/* USER CODE END 0 */
/* External variables ---------------------------------------------------------*/
/* USER CODE BEGIN EV */
/* USER CODE END EV */
/******************************************************************************/
/*            Cortex-M4 Processor Interruption and Exception Handlers          */
/******************************************************************************/
/**
  * @brief This function handles Non maskable interrupt.
  */
void NMI_Handler(void)
{
  /* USER CODE BEGIN NonMaskableInt_IRQn 0 */
  /* USER CODE END NonMaskableInt_IRQn 0 */
  /* USER CODE BEGIN NonMaskableInt_IRQn 1 */
   while (1)
  {
  }
  /* USER CODE END NonMaskableInt_IRQn 1 */
}
/**
```

```
  * @brief This function handles Hard fault interrupt.
  */
void HardFault_Handler(void)
{
  /* USER CODE BEGIN HardFault_IRQn 0 */
  /* USER CODE END HardFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_HardFault_IRQn 0 */
    /* USER CODE END W1_HardFault_IRQn 0 */
  }
}
/**
  * @brief This function handles Memory management fault.
  */
void MemManage_Handler(void)
{
  /* USER CODE BEGIN MemoryManagement_IRQn 0 */
  /* USER CODE END MemoryManagement_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_MemoryManagement_IRQn 0 */
    /* USER CODE END W1_MemoryManagement_IRQn 0 */
```

```
  }
}
/**
  * @brief This function handles Pre-fetch fault, memory access fault.
  */
void BusFault_Handler(void)
{
  /* USER CODE BEGIN BusFault_IRQn 0 */
  /* USER CODE END BusFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_BusFault_IRQn 0 */
    /* USER CODE END W1_BusFault_IRQn 0 */
  }
}
/**
  * @brief This function handles Undefined instruction or illegal state.
  */
void UsageFault_Handler(void)
{
  /* USER CODE BEGIN UsageFault_IRQn 0 */
  /* USER CODE END UsageFault_IRQn 0 */
  while (1)
```

```
  {
    /* USER CODE BEGIN W1_UsageFault_IRQn 0 */
    /* USER CODE END W1_UsageFault_IRQn 0 */
  }
}
/**
  * @brief This function handles System service call via SWI
instruction.
  */
void SVC_Handler(void)
{
  /* USER CODE BEGIN SVCall_IRQn 0 */
  /* USER CODE END SVCall_IRQn 0 */
  /* USER CODE BEGIN SVCall_IRQn 1 */
  /* USER CODE END SVCall_IRQn 1 */
}
/**
  * @brief This function handles Debug monitor.
  */
void DebugMon_Handler(void)
{
  /* USER CODE BEGIN DebugMonitor_IRQn 0 */
  /* USER CODE END DebugMonitor_IRQn 0 */
```

```
  /* USER CODE BEGIN DebugMonitor_IRQn 1 */
  /* USER CODE END DebugMonitor_IRQn 1 */
}
/**
  * @brief This function handles Pendable request for system service.
  */
void PendSV_Handler(void)
{
  /* USER CODE BEGIN PendSV_IRQn 0 */
  /* USER CODE END PendSV_IRQn 0 */
  /* USER CODE BEGIN PendSV_IRQn 1 */
  /* USER CODE END PendSV_IRQn 1 */
}
/**
  * @brief This function handles System tick timer.
  */
void SysTick_Handler(void)
{
  /* USER CODE BEGIN SysTick_IRQn 0 */
  /* USER CODE END SysTick_IRQn 0 */
  HAL_IncTick();
  /* USER CODE BEGIN SysTick_IRQn 1 */
  /* USER CODE END SysTick_IRQn 1 */
```

```
}
/*****************************************************
********************/
/* STM32F4xx Peripheral Interrupt Handlers                    */
/* Add here the Interrupt Handlers for the used peripherals.  */
/* For the available peripheral interrupt handler names,      */
/* please refer to the startup file (startup_stm32f4xx.s).    */
/*****************************************************
********************/
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

23.10 CODE OF SYSCALLS.C:

```
/**
*****************************************************
*******************
 * @file      syscalls.c
 * @author    Auto-generated by STM32CubeIDE
 * @brief     STM32CubeIDE Minimal System calls file
 *
 *            For more information about which c-functions
 *            need which of these lowlevel functions
 *            please consult the Newlib libc-manual
```

```
*****************************************************
*******************
 * @attention
 *
 * Copyright (c) 2020-2024 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the
LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
*****************************************************
********************
 */
/* Includes */
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
#include <sys/times.h>
```

```
/* Variables */
extern int __io_putchar(int ch) __attribute__((weak));
extern int __io_getchar(void) __attribute__((weak));
char *__env[1] = { 0 };
char **environ = __env;
/* Functions */
void initialise_monitor_handles()
{
}
int _getpid(void)
{
  return 1;
}
int _kill(int pid, int sig)
{
  (void)pid;
  (void)sig;
  errno = EINVAL;
  return -1;
}
void _exit (int status)
{
  _kill(status, -1);
```

```
  while (1) {}    /* Make sure we hang here */
}
__attribute__((weak)) int _read(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;
  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    *ptr++ = __io_getchar();
  }
  return len;
}
__attribute__((weak)) int _write(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;
  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    __io_putchar(*ptr++);
  }
  return len;
}
int _close(int file)
```

```
{
  (void)file;
  return -1;
}
int _fstat(int file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
int _isatty(int file)
{
  (void)file;
  return 1;
}
int _lseek(int file, int ptr, int dir)
{
  (void)file;
  (void)ptr;
  (void)dir;
  return 0;
}
int _open(char *path, int flags, ...)
```

```
{
  (void)path;
  (void)flags;
  /* Pretend like we always fail */
  return -1;
}
int _wait(int *status)
{
  (void)status;
  errno = ECHILD;
  return -1;
}
int _unlink(char *name)
{
  (void)name;
  errno = ENOENT;
  return -1;
}
int _times(struct tms *buf)
{
  (void)buf;
  return -1;
}
```

```
int _stat(char *file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
int _link(char *old, char *new)
{
  (void)old;
  (void)new;
  errno = EMLINK;
  return -1;
}
int _fork(void)
{
  errno = EAGAIN;
  return -1;
}
int _execve(char *name, char **argv, char **env)
{
  (void)name;
  (void)argv;
  (void)env;
```

```
  errno = ENOMEM;
  return -1;
}
```

23.12 CODE OF SYSMEM.C:

```
/**
**********************************************************
********************
 * @file      sysmem.c
 * @author    Generated by STM32CubeIDE
 * @brief     STM32CubeIDE System Memory calls file
 *
 *            For more information about which C functions
 *            need which of these lowlevel functions
 *            please consult the newlib libc manual
**********************************************************
********************
 * @attention
 *
 * Copyright (c) 2024 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the
LICENSE file
 * in the root directory of this software component.
```

```
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
**********************************************************
*******************
 */
/* Includes */
#include <errno.h>
#include <stdint.h>
/**
 * Pointer to the current high watermark of the heap usage
 */
static uint8_t *__sbrk_heap_end = NULL;
/**
 * @brief _sbrk() allocates memory to the newlib heap and is used by
malloc
 *        and others from the C library
 *
 * @verbatim
 *
#################################################################
##################
 * #  .data  #  .bss  #       newlib heap       #          MSP stack          #
 * #         #        #                         # Reserved by _Min_Stack_Size #
```

```
 *
##################################################################
##################
 * ^-- RAM start      ^-- _end                          _estack, RAM end --^
 * @endverbatim
 *
 * This implementation starts allocating at the '_end' linker symbol
 * The '_Min_Stack_Size' linker symbol reserves a memory for the MSP
stack
 * The implementation considers '_estack' linker symbol to be RAM end
 * NOTE: If the MSP stack, at any point during execution, grows larger
than the
 * reserved size, please increase the '_Min_Stack_Size'.
 *
 * @param incr Memory size
 * @return Pointer to allocated memory
 */
void *_sbrk(ptrdiff_t incr)
{
  extern uint8_t _end; /* Symbol defined in the linker script */
  extern uint8_t _estack; /* Symbol defined in the linker script */
  extern uint32_t _Min_Stack_Size; /* Symbol defined in the linker
script */
  const uint32_t stack_limit = (uint32_t)&_estack -
(uint32_t)&_Min_Stack_Size;
```

```
  const uint8_t *max_heap = (uint8_t *)stack_limit;
  uint8_t *prev_heap_end;
  /* Initialize heap end at first call */
  if (NULL == __sbrk_heap_end)
  {
    __sbrk_heap_end = &_end;
  }
  /* Protect heap from growing into the reserved MSP stack */
  if (__sbrk_heap_end + incr > max_heap)
  {
    errno = ENOMEM;
    return (void *)-1;
  }
  prev_heap_end = __sbrk_heap_end;
  __sbrk_heap_end += incr;
  return (void *)prev_heap_end;
}
```

23.13 CODE OF SYSTEM_STM32F4XX.C:

```
/**
*****************************************************************
*******************
  * @file    system_stm32f4xx.c
  * @author  MCD Application Team
```

```
  * @brief   CMSIS Cortex-M4 Device Peripheral Access Layer System
Source File.
  *
  *   This file provides two functions and one global variable to be called
from
  *   user application:
  *      - SystemInit(): This function is called at startup just after reset and
  *                      before branch to main program. This call is made inside
  *                      the "startup_stm32f4xx.s" file.
  *
  *      - SystemCoreClock variable: Contains the core clock (HCLK), it
can be used
  *                              by the user application to setup the SysTick
  *                              timer or configure other parameters.
  *
  *      - SystemCoreClockUpdate(): Updates the variable
SystemCoreClock and must
  *                             be called whenever the core clock is changed
  *                             during program execution.
  *
  *
**************************************************************
*******************
  * @attention
  *
```

```
  * Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*****************************************************************
*******************
  */
/** @addtogroup CMSIS
  * @{
  */
/** @addtogroup stm32f4xx_system
  * @{
  */
/** @addtogroup STM32F4xx_System_Private_Includes
  * @{
  */
#include "stm32f4xx.h"
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    ((uint32_t)25000000) /*!< Default value of
the External oscillator in Hz */
```

```
#endif /* HSE_VALUE */
#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000) /*!< Value of the
Internal oscillator in Hz*/
#endif /* HSI_VALUE */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_TypesDefinitions
  * @{
  */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Defines
  * @{
  */
/************************* Miscellaneous Configuration
************************/
/*!< Uncomment the following line if you need to use external SRAM or
SDRAM as data memory  */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx) || defined(STM32F417xx)\
 || defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
```

|| defined(STM32F469xx) || defined(STM32F479xx) || defined(STM32F412Zx) || defined(STM32F412Vx)

/* #define DATA_IN_ExtSRAM */

#endif /* STM32F40xxx || STM32F41xxx || STM32F42xxx || STM32F43xxx || STM32F469xx || STM32F479xx ||\

STM32F412Zx || STM32F412Vx */

#if defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) || defined(STM32F439xx)\

|| defined(STM32F446xx) || defined(STM32F469xx) || defined(STM32F479xx)

/* #define DATA_IN_ExtSDRAM */

#endif /* STM32F427xx || STM32F437xx || STM32F429xx || STM32F439xx || STM32F446xx || STM32F469xx ||\

STM32F479xx */

/* Note: Following vector table addresses must be defined in line with linker

configuration. */

/*!< Uncomment the following line if you need to relocate the vector table

anywhere in Flash or Sram, else the vector table is kept at the automatic

remap of boot address selected */

/* #define USER_VECT_TAB_ADDRESS */

#if defined(USER_VECT_TAB_ADDRESS)

/*!< Uncomment the following line if you need to relocate your vector Table

in Sram else user remap will be done in Flash. */

/* #define VECT_TAB_SRAM */

#if defined(VECT_TAB_SRAM)

#define VECT_TAB_BASE_ADDRESS SRAM_BASE /*!< Vector Table base address field.

This value must be a multiple of 0x200. */

#define VECT_TAB_OFFSET 0x00000000U /*!< Vector Table base offset field.

This value must be a multiple of 0x200. */

#else

#define VECT_TAB_BASE_ADDRESS FLASH_BASE /*!< Vector Table base address field.

This value must be a multiple of 0x200. */

#define VECT_TAB_OFFSET 0x00000000U /*!< Vector Table base offset field.

This value must be a multiple of 0x200. */

#endif /* VECT_TAB_SRAM */

#endif /* USER_VECT_TAB_ADDRESS */

/**
********************/

/**

* @}

```
  */
/** @addtogroup STM32F4xx_System_Private_Macros
  * @{
  */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Variables
  * @{
  */
  /* This variable is updated in three ways:
     1) by calling CMSIS function SystemCoreClockUpdate()
     2) by calling HAL API function HAL_RCC_GetHCLKFreq()
     3) each time HAL_RCC_ClockConfig() is called to configure the system clock frequency
        Note: If you use this function to configure the system clock; then there
              is no need to call the 2 first functions listed above, since SystemCoreClock
              variable is updated automatically.
  */
uint32_t SystemCoreClock = 16000000;
const uint8_t AHBPrescTable[16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9};
```

```
const uint8_t APBPrescTable[8]  = {0, 0, 0, 0, 1, 2, 3, 4};
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_FunctionPrototypes
  * @{
  */
#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  static void SystemInit_ExtMemCtl(void);
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Functions
  * @{
  */
/**
  * @brief  Setup the microcontroller system
  *         Initialize the FPU setting, vector table location and External memory
  *         configuration.
  * @param  None
  * @retval None
```

```
  */
void SystemInit(void)
{
  /* FPU settings ------------------------------------------------------------*/
  #if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
    SCB->CPACR |= ((3UL << 10*2)|(3UL << 11*2));  /* set CP10 and
CP11 Full Access */
  #endif
#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  SystemInit_ExtMemCtl();
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */
  /* Configure the Vector Table location -------------------------------------*/
#if defined(USER_VECT_TAB_ADDRESS)
  SCB->VTOR = VECT_TAB_BASE_ADDRESS |
VECT_TAB_OFFSET; /* Vector Table Relocation in Internal SRAM */
#endif /* USER_VECT_TAB_ADDRESS */
}
/**
   * @brief  Update SystemCoreClock variable according to Clock
Register Values.
  *         The SystemCoreClock variable contains the core clock (HCLK),
it can
  *         be used by the user application to setup the SysTick timer or
configure
  *         other parameters.
```

```
  *
  * @note   Each time the core clock (HCLK) changes, this function must be called
  *         to update SystemCoreClock variable value. Otherwise, any configuration
  *         based on this variable will be incorrect.
  *
  * @note   - The system frequency computed by this function is not the real
  *           frequency in the chip. It is calculated based on the predefined
  *           constant and the selected clock source:
  *
  *           - If SYSCLK source is HSI, SystemCoreClock will contain the HSI_VALUE(*)
  *
  *           - If SYSCLK source is HSE, SystemCoreClock will contain the HSE_VALUE(**)
  *
  *           - If SYSCLK source is PLL, SystemCoreClock will contain the HSE_VALUE(**)
  *             or HSI_VALUE(*) multiplied/divided by the PLL factors.
  *
  *         (*) HSI_VALUE is a constant defined in stm32f4xx_hal_conf.h file (default value
```

```
  *            16 MHz) but the real value may vary depending on the variations
  *            in voltage and temperature.
  *
  *        (**) HSE_VALUE is a constant defined in stm32f4xx_hal_conf.h file (its value
  *             depends on the application requirements), user has to ensure that HSE_VALUE
  *             is same as the real frequency of the crystal used. Otherwise, this function
  *             may have wrong result.
  *
  *        - The result of this function could be not correct when using fractional
  *          value for HSE crystal.
  *
  * @param  None
  * @retval None
  */
void SystemCoreClockUpdate(void)
{
  uint32_t tmp = 0, pllvco = 0, pllp = 2, pllsource = 0, pllm = 2;
  /* Get SYSCLK source -------------------------------------------------------*/
  tmp = RCC->CFGR & RCC_CFGR_SWS;
```

```
  switch (tmp)
  {
    case 0x00:  /* HSI used as system clock source */
      SystemCoreClock = HSI_VALUE;
      break;
    case 0x04:  /* HSE used as system clock source */
      SystemCoreClock = HSE_VALUE;
      break;
    case 0x08:  /* PLL used as system clock source */

      /* PLL_VCO = (HSE_VALUE or HSI_VALUE / PLL_M) * PLL_N
         SYSCLK = PLL_VCO / PLL_P
         */
      pllsource = (RCC->PLLCFGR & RCC_PLLCFGR_PLLSRC) >>
22;
      pllm = RCC->PLLCFGR & RCC_PLLCFGR_PLLM;
      if (pllsource != 0)
      {
        /* HSE used as PLL clock source */
        pllvco = (HSE_VALUE / pllm) * ((RCC->PLLCFGR &
RCC_PLLCFGR_PLLN) >> 6);
      }
      else
      {
```

```
      /* HSI used as PLL clock source */
      pllvco = (HSI_VALUE / pllm) * ((RCC->PLLCFGR &
RCC_PLLCFGR_PLLN) >> 6);
    }
    pllp = (((RCC->PLLCFGR & RCC_PLLCFGR_PLLP) >>16) + 1 )
*2;
    SystemCoreClock = pllvco/pllp;
    break;
   default:
    SystemCoreClock = HSI_VALUE;
    break;
  }
  /* Compute HCLK frequency ----------------------------------------------
-*/
  /* Get HCLK prescaler */
  tmp = AHBPrescTable[((RCC->CFGR & RCC_CFGR_HPRE) >> 4)];
  /* HCLK frequency */
  SystemCoreClock >>= tmp;
}
#if defined (DATA_IN_ExtSRAM) && defined
(DATA_IN_ExtSDRAM)
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
/**
```

```
  * @brief  Setup the external memory controller.
  *         Called in startup_stm32f4xx.s before jump to main.
  *         This function configures the external memories
(SRAM/SDRAM)
  *         This SRAM/SDRAM will be used as program data memory
(including heap and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
  register __IO uint32_t index;
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and
GPIOI interface clock */
  RCC->AHB1ENR |= 0x000001F8;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x00CCC0CC;
  GPIOD->AFR[1]  = 0xCCCCCCCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xAAAA0A8A;
```

```
/* Configure PDx pins speed to 100 MHz */
GPIOD->OSPEEDR = 0xFFFF0FCF;
/* Configure PDx pins Output type to push-pull */
GPIOD->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PDx pins */
GPIOD->PUPDR   = 0x00000000;
/* Connect PEx pins to FMC Alternate function */
GPIOE->AFR[0]  = 0xC00CC0CC;
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA828A;
/* Configure PEx pins speed to 100 MHz */
GPIOE->OSPEEDR = 0xFFFFC3CF;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0xCCCCCCCC;
GPIOF->AFR[1]  = 0xCCCCCCCC;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA800AAA;
/* Configure PFx pins speed to 50 MHz */
```

```
GPIOF->OSPEEDR = 0xAA800AAA;
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
GPIOF->PUPDR   = 0x00000000;
/* Connect PGx pins to FMC Alternate function */
GPIOG->AFR[0]  = 0xCCCCCCCC;
GPIOG->AFR[1]  = 0xCCCCCCCC;
/* Configure PGx pins in Alternate function mode */
GPIOG->MODER   = 0xAAAAAAAA;
/* Configure PGx pins speed to 50 MHz */
GPIOG->OSPEEDR = 0xAAAAAAAA;
/* Configure PGx pins Output type to push-pull */
GPIOG->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PGx pins */
GPIOG->PUPDR   = 0x00000000;
/* Connect PHx pins to FMC Alternate function */
GPIOH->AFR[0]  = 0x00C0CC00;
GPIOH->AFR[1]  = 0xCCCCCCCC;
/* Configure PHx pins in Alternate function mode */
GPIOH->MODER   = 0xAAAA08A0;
/* Configure PHx pins speed to 50 MHz */
GPIOH->OSPEEDR = 0xAAAA08A0;
```

```
  /* Configure PHx pins Output type to push-pull */
  GPIOH->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PHx pins */
  GPIOH->PUPDR   = 0x00000000;
  /* Connect PIx pins to FMC Alternate function */
  GPIOI->AFR[0]  = 0xCCCCCCCC;
  GPIOI->AFR[1]  = 0x00000CC0;
  /* Configure PIx pins in Alternate function mode */
  GPIOI->MODER   = 0x0028AAAA;
  /* Configure PIx pins speed to 50 MHz */
  GPIOI->OSPEEDR = 0x0028AAAA;
  /* Configure PIx pins Output type to push-pull */
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;
/*-- FMC Configuration ------------------------------------------------------*/
  /* Enable the FMC interface clock */
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
  FMC_Bank5_6->SDTR[0] = 0x01115351;
  /* SDRAM initialization sequence */
```

```
/* Clock enable command */
FMC_Bank5_6->SDCMR = 0x00000011;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* Delay */
for (index = 0; index<1000; index++);
/* PALL command */
FMC_Bank5_6->SDCMR = 0x00000012;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* Auto refresh command */
FMC_Bank5_6->SDCMR = 0x00000073;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
```

```
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* MRD register program */
  FMC_Bank5_6->SDCMR = 0x00046014;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Set refresh count */
  tmpreg = FMC_Bank5_6->SDRTR;
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
  /* Disable write protection */
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx */
```

```
#if defined(STM32F469xx) || defined(STM32F479xx)
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F469xx || STM32F479xx */
  (void)(tmp);
}
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F469xx || STM32F479xx */
#elif defined (DATA_IN_ExtSRAM) || defined
(DATA_IN_ExtSDRAM)
/**
  * @brief  Setup the external memory controller.
  *         Called in startup_stm32f4xx.s before jump to main.
  *         This function configures the external memories
(SRAM/SDRAM)
  *         This SRAM/SDRAM will be used as program data memory
(including heap and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
```

```
  __IO uint32_t tmp = 0x00;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F446xx) || defined(STM32F469xx) ||
defined(STM32F479xx)
#if defined (DATA_IN_ExtSDRAM)
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
  register __IO uint32_t index;
#if defined(STM32F446xx)
  /* Enable GPIOA, GPIOC, GPIOD, GPIOE, GPIOF, GPIOG interface
     clock */
  RCC->AHB1ENR |= 0x0000007D;
#else
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and
GPIOI interface
     clock */
  RCC->AHB1ENR |= 0x000001F8;
#endif /* STM32F446xx */
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);
#if defined(STM32F446xx)
  /* Connect PAx pins to FMC Alternate function */
  GPIOA->AFR[0]  |= 0xC0000000;
  GPIOA->AFR[1]  |= 0x00000000;
```

```
  /* Configure PDx pins in Alternate function mode */
  GPIOA->MODER   |= 0x00008000;
  /* Configure PDx pins speed to 50 MHz */
  GPIOA->OSPEEDR |= 0x00008000;
  /* Configure PDx pins Output type to push-pull */
  GPIOA->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOA->PUPDR   |= 0x00000000;
  /* Connect PCx pins to FMC Alternate function */
  GPIOC->AFR[0]  |= 0x00CC0000;
  GPIOC->AFR[1]  |= 0x00000000;
  /* Configure PDx pins in Alternate function mode */
  GPIOC->MODER   |= 0x00000A00;
  /* Configure PDx pins speed to 50 MHz */
  GPIOC->OSPEEDR |= 0x00000A00;
  /* Configure PDx pins Output type to push-pull */
  GPIOC->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOC->PUPDR   |= 0x00000000;
#endif /* STM32F446xx */
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x000000CC;
  GPIOD->AFR[1]  = 0xCC000CCC;
```

```
/* Configure PDx pins in Alternate function mode */
GPIOD->MODER   = 0xA02A000A;
/* Configure PDx pins speed to 50 MHz */
GPIOD->OSPEEDR = 0xA02A000A;
/* Configure PDx pins Output type to push-pull */
GPIOD->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PDx pins */
GPIOD->PUPDR   = 0x00000000;
/* Connect PEx pins to FMC Alternate function */
GPIOE->AFR[0]  = 0xC00000CC;
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA800A;
/* Configure PEx pins speed to 50 MHz */
GPIOE->OSPEEDR = 0xAAAA800A;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0xCCCCCCCC;
GPIOF->AFR[1]  = 0xCCCCCCCC;
/* Configure PFx pins in Alternate function mode */
```

```
  GPIOF->MODER   = 0xAA800AAA;
  /* Configure PFx pins speed to 50 MHz */
  GPIOF->OSPEEDR = 0xAA800AAA;
  /* Configure PFx pins Output type to push-pull */
  GPIOF->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PFx pins */
  GPIOF->PUPDR   = 0x00000000;
  /* Connect PGx pins to FMC Alternate function */
  GPIOG->AFR[0]  = 0xCCCCCCCC;
  GPIOG->AFR[1]  = 0xCCCCCCCC;
  /* Configure PGx pins in Alternate function mode */
  GPIOG->MODER   = 0xAAAAAAAA;
  /* Configure PGx pins speed to 50 MHz */
  GPIOG->OSPEEDR = 0xAAAAAAAA;
  /* Configure PGx pins Output type to push-pull */
  GPIOG->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
  /* Connect PHx pins to FMC Alternate function */
  GPIOH->AFR[0]  = 0x00C0CC00;
```

```
  GPIOH->AFR[1]  = 0xCCCCCCCC;
  /* Configure PHx pins in Alternate function mode */
  GPIOH->MODER   = 0xAAAA08A0;
  /* Configure PHx pins speed to 50 MHz */
  GPIOH->OSPEEDR = 0xAAAA08A0;
  /* Configure PHx pins Output type to push-pull */
  GPIOH->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PHx pins */
  GPIOH->PUPDR   = 0x00000000;
  /* Connect PIx pins to FMC Alternate function */
  GPIOI->AFR[0]  = 0xCCCCCCCC;
  GPIOI->AFR[1]  = 0x00000CC0;
  /* Configure PIx pins in Alternate function mode */
  GPIOI->MODER   = 0x0028AAAA;
  /* Configure PIx pins speed to 50 MHz */
  GPIOI->OSPEEDR = 0x0028AAAA;
  /* Configure PIx pins Output type to push-pull */
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F469xx || STM32F479xx */
/*-- FMC Configuration ------------------------------------------------------*/
```

```
  /* Enable the FMC interface clock */
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable SDRAM bank1 */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCR[0] = 0x00001954;
#else
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
#endif /* STM32F446xx */
  FMC_Bank5_6->SDTR[0] = 0x01115351;
  /* SDRAM initialization sequence */
  /* Clock enable command */
  FMC_Bank5_6->SDCMR = 0x00000011;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Delay */
  for (index = 0; index<1000; index++);
  /* PALL command */
  FMC_Bank5_6->SDCMR = 0x00000012;
```

```
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Auto refresh command */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x000000F3;
#else
  FMC_Bank5_6->SDCMR = 0x00000073;
#endif /* STM32F446xx */
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* MRD register program */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x00044014;
#else
  FMC_Bank5_6->SDCMR = 0x00046014;
```

```
#endif /* STM32F446xx */
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Set refresh count */
  tmpreg = FMC_Bank5_6->SDRTR;
#if defined(STM32F446xx)
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000050C<<1));
#else
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
#endif /* STM32F446xx */
  /* Disable write protection */
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#endif /* DATA_IN_ExtSDRAM */
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F446xx || STM32F469xx || STM32F479xx */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx) || defined(STM32F417xx)\
 || defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
```

```
 || defined(STM32F469xx) || defined(STM32F479xx) ||
defined(STM32F412Zx) || defined(STM32F412Vx)
#if defined(DATA_IN_ExtSRAM)
/*-- GPIOs Configuration -----------------------------------------------------*/
  /* Enable GPIOD, GPIOE, GPIOF and GPIOG interface clock */
  RCC->AHB1ENR   |= 0x00000078;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIODEN);
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x00CCC0CC;
  GPIOD->AFR[1]  = 0xCCCCCCCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xAAAA0A8A;
  /* Configure PDx pins speed to 100 MHz */
  GPIOD->OSPEEDR = 0xFFFF0FCF;
  /* Configure PDx pins Output type to push-pull */
  GPIOD->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOD->PUPDR   = 0x00000000;
  /* Connect PEx pins to FMC Alternate function */
  GPIOE->AFR[0]  = 0xC00CC0CC;
  GPIOE->AFR[1]  = 0xCCCCCCCC;
  /* Configure PEx pins in Alternate function mode */
```

```
GPIOE->MODER   = 0xAAAA828A;
/* Configure PEx pins speed to 100 MHz */
GPIOE->OSPEEDR = 0xFFFFC3CF;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0x00CCCCCC;
GPIOF->AFR[1]  = 0xCCCC0000;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA000AAA;
/* Configure PFx pins speed to 100 MHz */
GPIOF->OSPEEDR = 0xFF000FFF;
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
GPIOF->PUPDR   = 0x00000000;
/* Connect PGx pins to FMC Alternate function */
GPIOG->AFR[0]  = 0x00CCCCCC;
GPIOG->AFR[1]  = 0x000000C0;
/* Configure PGx pins in Alternate function mode */
GPIOG->MODER   = 0x00085AAA;
```

```
  /* Configure PGx pins speed to 100 MHz */
  GPIOG->OSPEEDR = 0x000CAFFF;
  /* Configure PGx pins Output type to push-pull */
  GPIOG->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;
/*-- FMC/FSMC Configuration ----------------------------------------------
-*/
  /* Enable the FMC/FSMC interface clock */
  RCC->AHB3ENR         |= 0x00000001;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
```

```
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F469xx || STM32F479xx */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx)|| defined(STM32F417xx)\
   || defined(STM32F412Zx) || defined(STM32F412Vx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FSMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FSMC_Bank1->BTCR[2]  = 0x00001011;
  FSMC_Bank1->BTCR[3]  = 0x00000201;
  FSMC_Bank1E->BWTR[2] = 0x0FFFFFFF;
#endif /* STM32F405xx || STM32F415xx || STM32F407xx ||
STM32F417xx || STM32F412Zx || STM32F412Vx */
#endif /* DATA_IN_ExtSRAM */
#endif /* STM32F405xx || STM32F415xx || STM32F407xx ||
STM32F417xx || STM32F427xx || STM32F437xx ||\
        STM32F429xx || STM32F439xx || STM32F469xx ||
STM32F479xx || STM32F412Zx || STM32F412Vx  */
  (void)(tmp);
}
#endif /* DATA_IN_ExtSRAM && DATA_IN_ExtSDRAM */
/**
```

```
  * @}
  */
/**
  * @}
  */
/**
  * @}
  */
```

CHAPTER#24 CODING A HARDWARE TIMER TO GENERATE PRECISE DELAYS

24.1CODING A HARDWARE TIMER TO GENERATE PRECISE DELAYS:

To generate precise delays using a hardware timer, you can use a microcontroller's timer peripherals. These timers are more accurate and consistent compared to software-based delays.

24.2 SELECT THE TIMER:

Microcontrollers usually have multiple timers, each with different capabilities and prescalers. Choose a timer that best suits your requirements in terms of resolution and range.

24.3 CONFIGURE THE TIMER:

The prescaler divides the system clock to a slower frequency to feed the timer. The prescaler value is chosen based on the desired timer resolution and the maximum delay needed.

24.4 SET THE TIMER MODE:

Timers can operate in different modes, such as up-counting, down-counting, or up/down-counting. Choose the appropriate mode for your application.

24.5 CALCULATE TIMER OVERFLOW OR COMPARE VALUE:

Calculate the value to load into the timer's register, which determines when the timer will overflow or trigger an interrupt. This calculation is based on the desired delay, the timer's clock frequency, and the prescaler value.

24.6 ENABLE TIMER INTERRUPT:

If delay requires precise actions (e.g., toggling an output pin), enable the timer's interrupt and write an interrupt service routine (ISR) to handle the event.

24.7 START THE TIMER:

Enable the timer to start counting.

24.8 MONITOR OR WAIT FOR THE EVENT TIMER EVENT:

Depending on application, you may poll a flag, wait in a loop, or handle the timer interrupt when the timer reaches the set value.

24.9 CODE OF MAIN.H:

```
/* USER CODE BEGIN Header */

/**
```

```
**********************************************************
*******************
  * @file           : main.h
  * @brief          : Header for main.c file.
  *                   This file contains the common defines of the application.
**********************************************************
*******************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H
#ifdef __cplusplus
```

```
extern "C" {
#endif
/* Includes ------------------------------------------------------------------*/
#include "stm32f4xx_hal.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */
/* USER CODE END EC */
/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */
/* USER CODE END EM */
/* Exported functions prototypes ---------------------------------------------*/
void Error_Handler(void);
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
/* Private defines -----------------------------------------------------------*/
#define B1_Pin GPIO_PIN_13
#define B1_GPIO_Port GPIOC
```

```
#define USART_TX_Pin GPIO_PIN_2
#define USART_TX_GPIO_Port GPIOA
#define USART_RX_Pin GPIO_PIN_3
#define USART_RX_GPIO_Port GPIOA
#define LD2_Pin GPIO_PIN_5
#define LD2_GPIO_Port GPIOA
#define TMS_Pin GPIO_PIN_13
#define TMS_GPIO_Port GPIOA
#define TCK_Pin GPIO_PIN_14
#define TCK_GPIO_Port GPIOA
#define SWO_Pin GPIO_PIN_3
#define SWO_GPIO_Port GPIOB
/* USER CODE BEGIN Private defines */
/* USER CODE END Private defines */
#ifdef __cplusplus
}
#endif
#endif /* __MAIN_H */
```

24.10 CODE OF STM32F4XX_HAL_CONF.H:

```
/* USER CODE BEGIN Header */
/**
*****************************************************************
*******************
```

```
  * @file    stm32f4xx_hal_conf_template.h
  * @author  MCD Application Team
  * @brief   HAL configuration template file.
  *          This file should be copied to the application folder and renamed
  *          to stm32f4xx_hal_conf.h.
****************************************************************************
  * @attention
  *
  * Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
****************************************************************************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_HAL_CONF_H
#define __STM32F4xx_HAL_CONF_H
#ifdef __cplusplus
```

```
 extern "C" {
#endif
/* Exported types ------------------------------------------------------------*/
/* Exported constants --------------------------------------------------------*/
/* ############################ Module Selection
################################## */
/**
  * @brief This is the list of modules to be used in the HAL driver
  */
#define HAL_MODULE_ENABLED
  /* #define HAL_CRYP_MODULE_ENABLED */
/* #define HAL_ADC_MODULE_ENABLED */
/* #define HAL_CAN_MODULE_ENABLED */
/* #define HAL_CRC_MODULE_ENABLED */
/* #define HAL_CAN_LEGACY_MODULE_ENABLED */
/* #define HAL_DAC_MODULE_ENABLED */
/* #define HAL_DCMI_MODULE_ENABLED */
/* #define HAL_DMA2D_MODULE_ENABLED */
/* #define HAL_ETH_MODULE_ENABLED */
/* #define HAL_ETH_LEGACY_MODULE_ENABLED */
/* #define HAL_NAND_MODULE_ENABLED */
/* #define HAL_NOR_MODULE_ENABLED */
/* #define HAL_PCCARD_MODULE_ENABLED */
```

```
/* #define HAL_SRAM_MODULE_ENABLED */
/* #define HAL_SDRAM_MODULE_ENABLED */
/* #define HAL_HASH_MODULE_ENABLED */
/* #define HAL_I2C_MODULE_ENABLED */
/* #define HAL_I2S_MODULE_ENABLED */
/* #define HAL_IWDG_MODULE_ENABLED */
/* #define HAL_LTDC_MODULE_ENABLED */
/* #define HAL_RNG_MODULE_ENABLED */
/* #define HAL_RTC_MODULE_ENABLED */
/* #define HAL_SAI_MODULE_ENABLED */
/* #define HAL_SD_MODULE_ENABLED */
/* #define HAL_MMC_MODULE_ENABLED */
/* #define HAL_SPI_MODULE_ENABLED */
#define HAL_TIM_MODULE_ENABLED
#define HAL_UART_MODULE_ENABLED
/* #define HAL_USART_MODULE_ENABLED */
/* #define HAL_IRDA_MODULE_ENABLED */
/* #define HAL_SMARTCARD_MODULE_ENABLED */
/* #define HAL_SMBUS_MODULE_ENABLED */
/* #define HAL_WWDG_MODULE_ENABLED */
/* #define HAL_PCD_MODULE_ENABLED */
/* #define HAL_HCD_MODULE_ENABLED */
/* #define HAL_DSI_MODULE_ENABLED */
```

```
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_CEC_MODULE_ENABLED */
/* #define HAL_FMPI2C_MODULE_ENABLED */
/* #define HAL_FMPSMBUS_MODULE_ENABLED */
/* #define HAL_SPDIFRX_MODULE_ENABLED */
/* #define HAL_DFSDM_MODULE_ENABLED */
/* #define HAL_LPTIM_MODULE_ENABLED */
#define HAL_GPIO_MODULE_ENABLED
#define HAL_EXTI_MODULE_ENABLED
#define HAL_DMA_MODULE_ENABLED
#define HAL_RCC_MODULE_ENABLED
#define HAL_FLASH_MODULE_ENABLED
#define HAL_PWR_MODULE_ENABLED
#define HAL_CORTEX_MODULE_ENABLED
/* ############################ HSE/HSI Values adaptation
###################### */
/**
  * @brief Adjust the value of External High Speed oscillator (HSE)
used in your application.
  *        This value is used by the RCC HAL module to compute the
system frequency
  *        (when HSE is used as system clock source, directly or through
the PLL).
```

```
  */
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    8000000U /*!< Value of the External
oscillator in Hz */
#endif /* HSE_VALUE */
#if !defined  (HSE_STARTUP_TIMEOUT)
  #define HSE_STARTUP_TIMEOUT    100U   /*!< Time out for
HSE start up, in ms */
#endif /* HSE_STARTUP_TIMEOUT */
/**
  * @brief Internal High Speed oscillator (HSI) value.
  *        This value is used by the RCC HAL module to compute the
system frequency
  *        (when HSI is used as system clock source, directly or through the
PLL).
  */
#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000U) /*!< Value of the
Internal oscillator in Hz*/
#endif /* HSI_VALUE */
/**
  * @brief Internal Low Speed oscillator (LSI) value.
  */
#if !defined  (LSI_VALUE)
 #define LSI_VALUE  32000U       /*!< LSI Typical Value in Hz*/
```

```
#endif /* LSI_VALUE */                    /*!< Value of the Internal Low Speed oscillator in Hz
                                The real value may vary depending on the variations
                                in voltage and temperature.*/
/**
  * @brief External Low Speed oscillator (LSE) value.
  */
#if !defined  (LSE_VALUE)
 #define LSE_VALUE  32768U    /*!< Value of the External Low Speed oscillator in Hz */
#endif /* LSE_VALUE */
#if !defined  (LSE_STARTUP_TIMEOUT)
  #define LSE_STARTUP_TIMEOUT    5000U   /*!< Time out for LSE start up, in ms */
#endif /* LSE_STARTUP_TIMEOUT */
/**
  * @brief External clock source for I2S peripheral
  *       This value is used by the I2S HAL module to compute the I2S clock source
  *       frequency, this source is inserted directly through I2S_CKIN pad.
  */
#if !defined  (EXTERNAL_CLOCK_VALUE)
```

```
  #define EXTERNAL_CLOCK_VALUE    12288000U /*!< Value of
the External audio frequency in Hz*/

#endif /* EXTERNAL_CLOCK_VALUE */

/* Tip: To avoid modifying this file each time you need to use different
HSE,

   ===  you can define the HSE value in your toolchain compiler
preprocessor. */

/* ############################ System Configuration
######################### */

/**

  * @brief This is the HAL system configuration section

  */

#define  VDD_VALUE                    3300U /*!< Value of VDD in mv */

#define  TICK_INT_PRIORITY            0U   /*!< tick interrupt
priority */

#define  USE_RTOS                     0U

#define  PREFETCH_ENABLE              1U

#define  INSTRUCTION_CACHE_ENABLE     1U

#define  DATA_CACHE_ENABLE            1U

#define  USE_HAL_ADC_REGISTER_CALLBACKS        0U /*
ADC register callback disabled       */

#define  USE_HAL_CAN_REGISTER_CALLBACKS        0U /*
CAN register callback disabled       */

#define  USE_HAL_CEC_REGISTER_CALLBACKS        0U /*
CEC register callback disabled       */
```

#define USE_HAL_CRYP_REGISTER_CALLBACKS 0U /* CRYP register callback disabled */

#define USE_HAL_DAC_REGISTER_CALLBACKS 0U /* DAC register callback disabled */

#define USE_HAL_DCMI_REGISTER_CALLBACKS 0U /* DCMI register callback disabled */

#define USE_HAL_DFSDM_REGISTER_CALLBACKS 0U /* DFSDM register callback disabled */

#define USE_HAL_DMA2D_REGISTER_CALLBACKS 0U /* DMA2D register callback disabled */

#define USE_HAL_DSI_REGISTER_CALLBACKS 0U /* DSI register callback disabled */

#define USE_HAL_ETH_REGISTER_CALLBACKS 0U /* ETH register callback disabled */

#define USE_HAL_HASH_REGISTER_CALLBACKS 0U /* HASH register callback disabled */

#define USE_HAL_HCD_REGISTER_CALLBACKS 0U /* HCD register callback disabled */

#define USE_HAL_I2C_REGISTER_CALLBACKS 0U /* I2C register callback disabled */

#define USE_HAL_FMPI2C_REGISTER_CALLBACKS 0U /* FMPI2C register callback disabled */

#define USE_HAL_FMPSMBUS_REGISTER_CALLBACKS 0U /* FMPSMBUS register callback disabled */

#define USE_HAL_I2S_REGISTER_CALLBACKS 0U /* I2S register callback disabled */

#define USE_HAL_IRDA_REGISTER_CALLBACKS 0U /* IRDA register callback disabled */

#define USE_HAL_LPTIM_REGISTER_CALLBACKS 0U /* LPTIM register callback disabled */

#define USE_HAL_LTDC_REGISTER_CALLBACKS 0U /* LTDC register callback disabled */

#define USE_HAL_MMC_REGISTER_CALLBACKS 0U /* MMC register callback disabled */

#define USE_HAL_NAND_REGISTER_CALLBACKS 0U /* NAND register callback disabled */

#define USE_HAL_NOR_REGISTER_CALLBACKS 0U /* NOR register callback disabled */

#define USE_HAL_PCCARD_REGISTER_CALLBACKS 0U /* PCCARD register callback disabled */

#define USE_HAL_PCD_REGISTER_CALLBACKS 0U /* PCD register callback disabled */

#define USE_HAL_QSPI_REGISTER_CALLBACKS 0U /* QSPI register callback disabled */

#define USE_HAL_RNG_REGISTER_CALLBACKS 0U /* RNG register callback disabled */

#define USE_HAL_RTC_REGISTER_CALLBACKS 0U /* RTC register callback disabled */

#define USE_HAL_SAI_REGISTER_CALLBACKS 0U /* SAI register callback disabled */

#define USE_HAL_SD_REGISTER_CALLBACKS 0U /* SD register callback disabled */

#define USE_HAL_SMARTCARD_REGISTER_CALLBACKS 0U /* SMARTCARD register callback disabled */

#define USE_HAL_SDRAM_REGISTER_CALLBACKS 0U /* SDRAM register callback disabled */

#define USE_HAL_SRAM_REGISTER_CALLBACKS 0U /* SRAM register callback disabled */

#define USE_HAL_SPDIFRX_REGISTER_CALLBACKS 0U /* SPDIFRX register callback disabled */

#define USE_HAL_SMBUS_REGISTER_CALLBACKS 0U /* SMBUS register callback disabled */

#define USE_HAL_SPI_REGISTER_CALLBACKS 0U /* SPI register callback disabled */

#define USE_HAL_TIM_REGISTER_CALLBACKS 0U /* TIM register callback disabled */

#define USE_HAL_UART_REGISTER_CALLBACKS 0U /* UART register callback disabled */

#define USE_HAL_USART_REGISTER_CALLBACKS 0U /* USART register callback disabled */

#define USE_HAL_WWDG_REGISTER_CALLBACKS 0U /* WWDG register callback disabled */

/* ############################ Assert Selection ################################ */

/**

* @brief Uncomment the line below to expanse the "assert_param" macro in the

* HAL drivers code

*/

/* #define USE_FULL_ASSERT 1U */

/* ################### Ethernet peripheral configuration ###################### */

/* Section 1 : Ethernet peripheral configuration */

/* MAC ADDRESS: MAC_ADDR0:MAC_ADDR1:MAC_ADDR2:MAC_ADDR3:MAC_ADDR4:MAC_ADDR5 */

#define MAC_ADDR0 2U

#define MAC_ADDR1 0U

#define MAC_ADDR2 0U

#efine MAC_ADDR3 0U

#define MAC_ADDR4 0U

#define MAC_ADDR5 0U

/* Definition of the Ethernet driver buffers size and count */

#define ETH_RX_BUF_SIZE ETH_MAX_PACKET_SIZE /* buffer size for receive */

#define ETH_TX_BUF_SIZE ETH_MAX_PACKET_SIZE /* buffer size for transmit */

#define ETH_RXBUFNB 4U /* 4 Rx buffers of size ETH_RX_BUF_SIZE */

#define ETH_TXBUFNB 4U /* 4 Tx buffers of size ETH_TX_BUF_SIZE */

/* Section 2: PHY configuration section */

/* DP83848_PHY_ADDRESS Address*/

#define DP83848_PHY_ADDRESS

/* PHY Reset delay these values are based on a 1 ms Systick interrupt*/

#define PHY_RESET_DELAY 0x000000FFU

/* PHY Configuration delay */

#define PHY_CONFIG_DELAY 0x00000FFFU

#define PHY_READ_TO 0x0000FFFFU

#define PHY_WRITE_TO 0x0000FFFFU

/* Section 3: Common PHY Registers */

#define PHY_BCR ((uint16_t)0x0000U) /*!< Transceiver Basic Control Register */

#define PHY_BSR ((uint16_t)0x0001U) /*!< Transceiver Basic Status Register */

#define PHY_RESET ((uint16_t)0x8000U) /*!< PHY Reset */

#define PHY_LOOPBACK ((uint16_t)0x4000U) /*!< Select loop-back mode */

#define PHY_FULLDUPLEX_100M ((uint16_t)0x2100U) /*!< Set the full-duplex mode at 100 Mb/s */

#define PHY_HALFDUPLEX_100M ((uint16_t)0x2000U) /*!< Set the half-duplex mode at 100 Mb/s */

#define PHY_FULLDUPLEX_10M ((uint16_t)0x0100U) /*!< Set the full-duplex mode at 10 Mb/s */

#define PHY_HALFDUPLEX_10M ((uint16_t)0x0000U) /*!< Set the half-duplex mode at 10 Mb/s */

#define PHY_AUTONEGOTIATION ((uint16_t)0x1000U) /*!< Enable auto-negotiation function */

#define PHY_RESTART_AUTONEGOTIATION ((uint16_t)0x0200U) /*!< Restart auto-negotiation function */

#define PHY_POWERDOWN ((uint16_t)0x0800U) /*!< Select the power down mode */

#define PHY_ISOLATE ((uint16_t)0x0400U) /*!< Isolate PHY from MII */

#define PHY_AUTONEGO_COMPLETE ((uint16_t)0x0020U) /*!< Auto-Negotiation process completed */

#define PHY_LINKED_STATUS ((uint16_t)0x0004U) /*!< Valid link established */

#define PHY_JABBER_DETECTION ((uint16_t)0x0002U) /*!< Jabber condition detected */

/* Section 4: Extended PHY Registers */

#define PHY_SR ((uint16_t)) /*!< PHY status register Offset */

#define PHY_SPEED_STATUS ((uint16_t)) /*!< PHY Speed mask */

#define PHY_DUPLEX_STATUS ((uint16_t)) /*!< PHY Duplex mask */

/* ################### SPI peripheral configuration ############################ */

/* CRC FEATURE: Use to activate CRC feature inside HAL SPI Driver

* Activated: CRC code is present inside driver

* Deactivated: CRC code cleaned from driver

*/

#define USE_SPI_CRC 0U

```
/* Includes ------------------------------------------------------------------*/
/**
  * @brief Include module's header file
  */
#ifdef HAL_RCC_MODULE_ENABLED
  #include "stm32f4xx_hal_rcc.h"
#endif /* HAL_RCC_MODULE_ENABLED */
#ifdef HAL_GPIO_MODULE_ENABLED
  #include "stm32f4xx_hal_gpio.h"
#endif /* HAL_GPIO_MODULE_ENABLED */
#ifdef HAL_EXTI_MODULE_ENABLED
  #include "stm32f4xx_hal_exti.h"
#endif /* HAL_EXTI_MODULE_ENABLED */
#ifdef HAL_DMA_MODULE_ENABLED
  #include "stm32f4xx_hal_dma.h"
#endif /* HAL_DMA_MODULE_ENABLED */
#ifdef HAL_CORTEX_MODULE_ENABLED
  #include "stm32f4xx_hal_cortex.h"
#endif /* HAL_CORTEX_MODULE_ENABLED */
#ifdef HAL_ADC_MODULE_ENABLED
  #include "stm32f4xx_hal_adc.h"
#endif /* HAL_ADC_MODULE_ENABLED */
#ifdef HAL_CAN_MODULE_ENABLED
```

```
  #include "stm32f4xx_hal_can.h"
#endif /* HAL_CAN_MODULE_ENABLED */
#ifdef HAL_CAN_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_can_legacy.h"
#endif /* HAL_CAN_LEGACY_MODULE_ENABLED */
#ifdef HAL_CRC_MODULE_ENABLED
  #include "stm32f4xx_hal_crc.h"
#endif /* HAL_CRC_MODULE_ENABLED */
#ifdef HAL_CRYP_MODULE_ENABLED
  #include "stm32f4xx_hal_cryp.h"
#endif /* HAL_CRYP_MODULE_ENABLED */
#ifdef HAL_DMA2D_MODULE_ENABLED
  #include "stm32f4xx_hal_dma2d.h"
#endif /* HAL_DMA2D_MODULE_ENABLED */
#ifdef HAL_DAC_MODULE_ENABLED
  #include "stm32f4xx_hal_dac.h"
#endif /* HAL_DAC_MODULE_ENABLED */
#ifdef HAL_DCMI_MODULE_ENABLED
  #include "stm32f4xx_hal_dcmi.h"
#endif /* HAL_DCMI_MODULE_ENABLED */
#ifdef HAL_ETH_MODULE_ENABLED
  #include "stm32f4xx_hal_eth.h"
#endif /* HAL_ETH_MODULE_ENABLED */
```

```
#ifdef HAL_ETH_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_eth_legacy.h"
#endif /* HAL_ETH_LEGACY_MODULE_ENABLED */
#ifdef HAL_FLASH_MODULE_ENABLED
  #include "stm32f4xx_hal_flash.h"
#endif /* HAL_FLASH_MODULE_ENABLED */
#ifdef HAL_SRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sram.h"
#endif /* HAL_SRAM_MODULE_ENABLED */
#ifdef HAL_NOR_MODULE_ENABLED
  #include "stm32f4xx_hal_nor.h"
#endif /* HAL_NOR_MODULE_ENABLED */
#ifdef HAL_NAND_MODULE_ENABLED
  #include "stm32f4xx_hal_nand.h"
#endif /* HAL_NAND_MODULE_ENABLED */
#ifdef HAL_PCCARD_MODULE_ENABLED
  #include "stm32f4xx_hal_pccard.h"
#endif /* HAL_PCCARD_MODULE_ENABLED */
#ifdef HAL_SDRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sdram.h"
#endif /* HAL_SDRAM_MODULE_ENABLED */
#ifdef HAL_HASH_MODULE_ENABLED
 #include "stm32f4xx_hal_hash.h"
```

```
#endif /* HAL_HASH_MODULE_ENABLED */
#ifdef HAL_I2C_MODULE_ENABLED
 #include "stm32f4xx_hal_i2c.h"
#endif /* HAL_I2C_MODULE_ENABLED */
#ifdef HAL_SMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_smbus.h"
#endif /* HAL_SMBUS_MODULE_ENABLED */
#ifdef HAL_I2S_MODULE_ENABLED
 #include "stm32f4xx_hal_i2s.h"
#endif /* HAL_I2S_MODULE_ENABLED */
#ifdef HAL_IWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_iwdg.h"
#endif /* HAL_IWDG_MODULE_ENABLED */
#ifdef HAL_LTDC_MODULE_ENABLED
 #include "stm32f4xx_hal_ltdc.h"
#endif /* HAL_LTDC_MODULE_ENABLED */
#ifdef HAL_PWR_MODULE_ENABLED
 #include "stm32f4xx_hal_pwr.h"
#endif /* HAL_PWR_MODULE_ENABLED */
#ifdef HAL_RNG_MODULE_ENABLED
 #include "stm32f4xx_hal_rng.h"
#endif /* HAL_RNG_MODULE_ENABLED */
#ifdef HAL_RTC_MODULE_ENABLED
```

```
 #include "stm32f4xx_hal_rtc.h"
#endif /* HAL_RTC_MODULE_ENABLED */
#ifdef HAL_SAI_MODULE_ENABLED
 #include "stm32f4xx_hal_sai.h"
#endif /* HAL_SAI_MODULE_ENABLED */
#ifdef HAL_SD_MODULE_ENABLED
 #include "stm32f4xx_hal_sd.h"
#endif /* HAL_SD_MODULE_ENABLED */
#ifdef HAL_SPI_MODULE_ENABLED
 #include "stm32f4xx_hal_spi.h"
#endif /* HAL_SPI_MODULE_ENABLED */
#ifdef HAL_TIM_MODULE_ENABLED
 #include "stm32f4xx_hal_tim.h"
#endif /* HAL_TIM_MODULE_ENABLED */
#ifdef HAL_UART_MODULE_ENABLED
 #include "stm32f4xx_hal_uart.h"
#endif /* HAL_UART_MODULE_ENABLED */
#ifdef HAL_USART_MODULE_ENABLED
 #include "stm32f4xx_hal_usart.h"
#endif /* HAL_USART_MODULE_ENABLED */
#ifdef HAL_IRDA_MODULE_ENABLED
 #include "stm32f4xx_hal_irda.h"
#endif /* HAL_IRDA_MODULE_ENABLED */
```

```
#ifdef HAL_SMARTCARD_MODULE_ENABLED
 #include "stm32f4xx_hal_smartcard.h"
#endif /* HAL_SMARTCARD_MODULE_ENABLED */
#ifdef HAL_WWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_wwdg.h"
#endif /* HAL_WWDG_MODULE_ENABLED */
#ifdef HAL_PCD_MODULE_ENABLED
 #include "stm32f4xx_hal_pcd.h"
#endif /* HAL_PCD_MODULE_ENABLED */
#ifdef HAL_HCD_MODULE_ENABLED
 #include "stm32f4xx_hal_hcd.h"
#endif /* HAL_HCD_MODULE_ENABLED */
#ifdef HAL_DSI_MODULE_ENABLED
 #include "stm32f4xx_hal_dsi.h"
#endif /* HAL_DSI_MODULE_ENABLED */
#ifdef HAL_QSPI_MODULE_ENABLED
 #include "stm32f4xx_hal_qspi.h"
#endif /* HAL_QSPI_MODULE_ENABLED */
#ifdef HAL_CEC_MODULE_ENABLED
 #include "stm32f4xx_hal_cec.h"
#endif /* HAL_CEC_MODULE_ENABLED */
#ifdef HAL_FMPI2C_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpi2c.h"
```

```
#endif /* HAL_FMPI2C_MODULE_ENABLED */
#ifdef HAL_FMPSMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpsmbus.h"
#endif /* HAL_FMPSMBUS_MODULE_ENABLED */
#ifdef HAL_SPDIFRX_MODULE_ENABLED
 #include "stm32f4xx_hal_spdifrx.h"
#endif /* HAL_SPDIFRX_MODULE_ENABLED */
#ifdef HAL_DFSDM_MODULE_ENABLED
 #include "stm32f4xx_hal_dfsdm.h"
#endif /* HAL_DFSDM_MODULE_ENABLED */
#ifdef HAL_LPTIM_MODULE_ENABLED
 #include "stm32f4xx_hal_lptim.h"
#endif /* HAL_LPTIM_MODULE_ENABLED */
#ifdef HAL_MMC_MODULE_ENABLED
 #include "stm32f4xx_hal_mmc.h"
#endif /* HAL_MMC_MODULE_ENABLED */
/* Exported macro ------------------------------------------------------------*/
#ifdef  USE_FULL_ASSERT
/**
  * @brief  The assert_param macro is used for function's parameters check.
  * @param  expr If expr is false, it calls assert_failed function
  *         which reports the name of the source file and the source
```

```
  *        line number of the call that failed.
  *        If expr is true, it returns no value.
  * @retval None
  */
  #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t
*)__FILE__, __LINE__))
/* Exported functions ---------------------------------------------------- */
  void assert_failed(uint8_t* file, uint32_t line);
#else
  #define assert_param(expr) ((void)0U)
#endif /* USE_FULL_ASSERT */
#ifdef __cplusplus
}
#endif
#endif /* __STM32F4xx_HAL_CONF_H */
```

24.11 CODE OF STM32F4XX_IT.H:

```
/* USER CODE BEGIN Header */
/**
*******************************************************************
  * @file    stm32f4xx_it.h
  * @brief   This file contains the headers of the interrupt handlers.
*******************************************************************
```

```
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
********************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_IT_H
#define __STM32F4xx_IT_H
#ifdef __cplusplus
 extern "C" {
#endif
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Exported types ------------------------------------------------------------*/
```

```
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants ---------------------------------------------------------*/
/* USER CODE BEGIN EC */
/* USER CODE END EC */
/* Exported macro -------------------------------------------------------------*/
/* USER CODE BEGIN EM */
/* USER CODE END EM */
/* Exported functions prototypes ----------------------------------------------*/
void NMI_Handler(void);
void HardFault_Handler(void);
void MemManage_Handler(void);
void BusFault_Handler(void);
void UsageFault_Handler(void);
void SVC_Handler(void);
void DebugMon_Handler(void);
void PendSV_Handler(void);
void SysTick_Handler(void);
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
#ifdef __cplusplus
}
#endif
```

```
#endif /* __STM32F4xx_IT_H */
```

24.12 CODE OF MAIN.C:

```
/* USER CODE BEGIN Header */
/**
**********************************************************
********************
  * @file           : main.c
  * @brief          : Main program body
**********************************************************
********************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
********************
  */
/* USER CODE END Header */
```

```
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
/* USER CODE END PTD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
TIM_HandleTypeDef htim2;
UART_HandleTypeDef huart2;
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM2_Init(void);
```

```
static void MX_USART2_UART_Init(void);
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */
  /* USER CODE END 1 */
  /* MCU Configuration--------------------------------------------------------*/
  /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
  HAL_Init();
  /* USER CODE BEGIN Init */
  /* USER CODE END Init */
  /* Configure the system clock */
  SystemClock_Config();
  /* USER CODE BEGIN SysInit */
```

```
  /* USER CODE END SysInit */
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_TIM2_Init();
  MX_USART2_UART_Init();
  /* USER CODE BEGIN 2 */
  /* USER CODE END 2 */
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
```

```
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_
VOLTAGE_SCALE1);
  /** Initializes the RCC Oscillators according to the specified
parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue =
RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 8;
  RCC_OscInitStruct.PLL.PLLN = 96;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
```

```
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource =
RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct,
FLASH_LATENCY_3) != HAL_OK)
  {
    Error_Handler();
  }
}
/**
  * @brief TIM2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM2_Init(void)
{
  /* USER CODE BEGIN TIM2_Init 0 */
```

```
  /* USER CODE END TIM2_Init 0 */
  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};
  /* USER CODE BEGIN TIM2_Init 1 */
  /* USER CODE END TIM2_Init 1 */
  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 48000;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 4294967295;
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim2.Init.AutoReloadPreload =
TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource =
TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig)
!= HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
```

```
  sMasterConfig.MasterSlaveMode =
TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim2,
&sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM2_Init 2 */
  /* USER CODE END TIM2_Init 2 */
}
/**
  * @brief USART2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART2_UART_Init(void)
{
  /* USER CODE BEGIN USART2_Init 0 */
  /* USER CODE END USART2_Init 0 */
  /* USER CODE BEGIN USART2_Init 1 */
  /* USER CODE END USART2_Init 1 */
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 115200;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
```

```
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART2_Init 2 */
  /* USER CODE END USART2_Init 2 */
}
/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */
  /* GPIO Ports Clock Enable */
```

```
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin,
GPIO_PIN_RESET);
  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);
  /*Configure GPIO pin : LD2_Pin */
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);
/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}
/* USER CODE BEGIN 4 */
/* USER CODE END 4 */
```

```
/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return
state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}
#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line
number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
```

```
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

24.13 CODE OF STM32F4XX_HAL_MSP.C:

```
/* USER CODE BEGIN Header */
/**
*******************************************************************
  * @file        stm32f4xx_hal_msp.c
  * @brief       This file provides code for the MSP Initialization
  *              and de-Initialization codes.
*******************************************************************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
```

```
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN TD */
/* USER CODE END TD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN Define */
/* USER CODE END Define */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN Macro */
/* USER CODE END Macro */
/* Private variables ---------------------------------------------------------*/
```

```
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* External functions --------------------------------------------------------*/
/* USER CODE BEGIN ExternalFunctions */
/* USER CODE END ExternalFunctions */
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/**
  * Initializes the Global MSP.
  */
void HAL_MspInit(void)
{
  /* USER CODE BEGIN MspInit 0 */
  /* USER CODE END MspInit 0 */
  __HAL_RCC_SYSCFG_CLK_ENABLE();
  __HAL_RCC_PWR_CLK_ENABLE();
  HAL_NVIC_SetPriorityGrouping(NVIC_PRIORITYGROUP_0);
  /* System interrupt init*/
  /* USER CODE BEGIN MspInit 1 */
  /* USER CODE END MspInit 1 */
```

```
}
/**
* @brief TIM_Base MSP Initialization
* This function configures the hardware resources used in this example
* @param htim_base: TIM_Base handle pointer
* @retval None
*/
void HAL_TIM_Base_MspInit(TIM_HandleTypeDef* htim_base)
{
  if(htim_base->Instance==TIM2)
  {
  /* USER CODE BEGIN TIM2_MspInit 0 */
  /* USER CODE END TIM2_MspInit 0 */
    /* Peripheral clock enable */
    __HAL_RCC_TIM2_CLK_ENABLE();
  /* USER CODE BEGIN TIM2_MspInit 1 */
  /* USER CODE END TIM2_MspInit 1 */
  }
}
/**
* @brief TIM_Base MSP De-Initialization
* This function freeze the hardware resources used in this example
* @param htim_base: TIM_Base handle pointer
```

```
* @retval None
*/
void HAL_TIM_Base_MspDeInit(TIM_HandleTypeDef* htim_base)
{
  if(htim_base->Instance==TIM2)
  {
  /* USER CODE BEGIN TIM2_MspDeInit 0 */
  /* USER CODE END TIM2_MspDeInit 0 */
    /* Peripheral clock disable */
    __HAL_RCC_TIM2_CLK_DISABLE();
  /* USER CODE BEGIN TIM2_MspDeInit 1 */
  /* USER CODE END TIM2_MspDeInit 1 */
  }
}
/**
* @brief UART MSP Initialization
* This function configures the hardware resources used in this example
* @param huart: UART handle pointer
* @retval None
*/
void HAL_UART_MspInit(UART_HandleTypeDef* huart)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
```

```
  if(huart->Instance==USART2)
  {
  /* USER CODE BEGIN USART2_MspInit 0 */
  /* USER CODE END USART2_MspInit 0 */
    /* Peripheral clock enable */
    __HAL_RCC_USART2_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();
    /**USART2 GPIO Configuration
    PA2     ------> USART2_TX
    PA3     ------> USART2_RX
    */
    GPIO_InitStruct.Pin = USART_TX_Pin|USART_RX_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF7_USART2;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
  /* USER CODE BEGIN USART2_MspInit 1 */
  /* USER CODE END USART2_MspInit 1 */
  }
}
/**
* @brief UART MSP De-Initialization
```

```
* This function freeze the hardware resources used in this example
* @param huart: UART handle pointer
* @retval None
*/
void HAL_UART_MspDeInit(UART_HandleTypeDef* huart)
{
  if(huart->Instance==USART2)
  {
  /* USER CODE BEGIN USART2_MspDeInit 0 */
  /* USER CODE END USART2_MspDeInit 0 */
    /* Peripheral clock disable */
    __HAL_RCC_USART2_CLK_DISABLE();
    /**USART2 GPIO Configuration
    PA2     ------> USART2_TX
    PA3     ------> USART2_RX
    */
    HAL_GPIO_DeInit(GPIOA, USART_TX_Pin|USART_RX_Pin);
  /* USER CODE BEGIN USART2_MspDeInit 1 */
  /* USER CODE END USART2_MspDeInit 1 */
  }
}
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

24.14 CODE OF STM32F4XX_IT.C:

```
/* USER CODE BEGIN Header */
/**
*****************************************************************
*******************
  * @file    stm32f4xx_it.c
  * @brief   Interrupt Service Routines.
*****************************************************************
*******************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*****************************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
```

```
#include "stm32f4xx_it.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN TD */
/* USER CODE END TD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/* External variables --------------------------------------------------------*/
```

```
/* USER CODE BEGIN EV */
/* USER CODE END EV */
/******************************************************
*********************/
/*          Cortex-M4 Processor Interruption and Exception Handlers
*/
/******************************************************
*********************/
/**
  * @brief This function handles Non maskable interrupt.
  */
void NMI_Handler(void)
{
  /* USER CODE BEGIN NonMaskableInt_IRQn 0 */

  /* USER CODE END NonMaskableInt_IRQn 0 */
  /* USER CODE BEGIN NonMaskableInt_IRQn 1 */
   while (1)
  {
  }
  /* USER CODE END NonMaskableInt_IRQn 1 */
}
/**
  * @brief This function handles Hard fault interrupt.
```

```
  */
void HardFault_Handler(void)
{
  /* USER CODE BEGIN HardFault_IRQn 0 */
  /* USER CODE END HardFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_HardFault_IRQn 0 */
    /* USER CODE END W1_HardFault_IRQn 0 */
  }
}
/**
  * @brief This function handles Memory management fault.
  */
void MemManage_Handler(void)
{
  /* USER CODE BEGIN MemoryManagement_IRQn 0 */
  /* USER CODE END MemoryManagement_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_MemoryManagement_IRQn 0 */
    /* USER CODE END W1_MemoryManagement_IRQn 0 */
  }
```

```
}
/**
  * @brief This function handles Pre-fetch fault, memory access fault.
  */
void BusFault_Handler(void)
{
  /* USER CODE BEGIN BusFault_IRQn 0 */
  /* USER CODE END BusFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_BusFault_IRQn 0 */
    /* USER CODE END W1_BusFault_IRQn 0 */
  }
}
/**
  * @brief This function handles Undefined instruction or illegal state.
  */
void UsageFault_Handler(void)
{
  /* USER CODE BEGIN UsageFault_IRQn 0 */
  /* USER CODE END UsageFault_IRQn 0 */
  while (1)
  {
```

```
    /* USER CODE BEGIN W1_UsageFault_IRQn 0 */
    /* USER CODE END W1_UsageFault_IRQn 0 */
  }
}
/**
  * @brief This function handles System service call via SWI
instruction.
  */
void SVC_Handler(void)
{
  /* USER CODE BEGIN SVCall_IRQn 0 */
  /* USER CODE END SVCall_IRQn 0 */
  /* USER CODE BEGIN SVCall_IRQn 1 */
  /* USER CODE END SVCall_IRQn 1 */
}
/**
  * @brief This function handles Debug monitor.
  */
void DebugMon_Handler(void)
{
  /* USER CODE BEGIN DebugMonitor_IRQn 0 */
  /* USER CODE END DebugMonitor_IRQn 0 */
  /* USER CODE BEGIN DebugMonitor_IRQn 1 */
```

```
  /* USER CODE END DebugMonitor_IRQn 1 */
}
/**
  * @brief This function handles Pendable request for system service.
  */
void PendSV_Handler(void)
{
  /* USER CODE BEGIN PendSV_IRQn 0 */
  /* USER CODE END PendSV_IRQn 0 */
  /* USER CODE BEGIN PendSV_IRQn 1 */
  /* USER CODE END PendSV_IRQn 1 */
}
/**
  * @brief This function handles System tick timer.
  */
void SysTick_Handler(void)
{
  /* USER CODE BEGIN SysTick_IRQn 0 */
  /* USER CODE END SysTick_IRQn 0 */
  HAL_IncTick();
  /* USER CODE BEGIN SysTick_IRQn 1 */
  /* USER CODE END SysTick_IRQn 1 */
}
```

```
/*****************************************************
********************/
/* STM32F4xx Peripheral Interrupt Handlers                    */
/* Add here the Interrupt Handlers for the used peripherals.  */
/* For the available peripheral interrupt handler names,      */
/* please refer to the startup file (startup_stm32f4xx.s).    */
/*****************************************************
********************/
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

24.15 CODE OF SYSCALLS.C:

```
/**
*****************************************************
******************
 * @file      syscalls.c
 * @author    Auto-generated by STM32CubeIDE
 * @brief     STM32CubeIDE Minimal System calls file
 *
 *            For more information about which c-functions
 *            need which of these lowlevel functions
 *            please consult the Newlib libc-manual
*****************************************************
******************
 * @attention
```

```
 *
 * Copyright (c) 2020-2024 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the
LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
*******************************************************
*******************
 */
/* Includes */
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
#include <sys/times.h>
/* Variables */
extern int __io_putchar(int ch) __attribute__((weak));
extern int __io_getchar(void) __attribute__((weak));
```

```
char *__env[1] = { 0 };
char **environ = __env;
/* Functions */
void initialise_monitor_handles()
{
}
int _getpid(void)
{
  return 1;
}
int _kill(int pid, int sig)
{
  (void)pid;
  (void)sig;
  errno = EINVAL;
  return -1;
}
void _exit (int status)
{
  _kill(status, -1);
  while (1) {}    /* Make sure we hang here */
}
__attribute__((weak)) int _read(int file, char *ptr, int len)
```

```
{
  (void)file;
  int DataIdx;
  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    *ptr++ = __io_getchar();
  }
  return len;
}
__attribute__((weak)) int _write(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;
  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    __io_putchar(*ptr++);
  }
  return len;
}
int _close(int file)
{
  (void)file;
  return -1;
```

```
}
int _fstat(int file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
int _isatty(int file)
{
  (void)file;
  return 1;
}
int _lseek(int file, int ptr, int dir)
{
  (void)file;
  (void)ptr;
  (void)dir;
  return 0;
}
int _open(char *path, int flags, ...)
{
  (void)path;
  (void)flags;
```

```
  /* Pretend like we always fail */
  return -1;
}
int _wait(int *status)
{
  (void)status;
  errno = ECHILD;
  return -1;
}
int _unlink(char *name)
{
  (void)name;
  errno = ENOENT;
  return -1;
}
int _times(struct tms *buf)
{
  (void)buf;
  return -1;
}
int _stat(char *file, struct stat *st)
{
  (void)file;
```

```
  st->st_mode = S_IFCHR;
  return 0;
}
int _link(char *old, char *new)
{
  (void)old;
  (void)new;
  errno = EMLINK;
  return -1;
}
int _fork(void)
{
  errno = EAGAIN;
  return -1;
}
int _execve(char *name, char **argv, char **env)
{
  (void)name;
  (void)argv;
  (void)env;
  errno = ENOMEM;
  return -1;
}
```

24.16 CODE OF SYSMEM.C:

```
/**
**********************************************************
*******************
 * @file      sysmem.c
 * @author    Generated by STM32CubeIDE
 * @brief     STM32CubeIDE System Memory calls file
 *
 *            For more information about which C functions
 *            need which of these lowlevel functions
 *            please consult the newlib libc manual
**********************************************************
*******************
 * @attention
 *
 * Copyright (c) 2024 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the
LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
```

```
**********************************************************
*******************
 */
/* Includes */
#include <errno.h>
#include <stdint.h>
/**
 * Pointer to the current high watermark of the heap usage
 */
static uint8_t *__sbrk_heap_end = NULL;
/**
 * @brief _sbrk() allocates memory to the newlib heap and is used by
malloc
 *        and others from the C library
 *
 * @verbatim
 *
#################################################################
##################
 * #  .data  #  .bss  #       newlib heap       #          MSP stack          #
 * #         #        #                          # Reserved by _Min_Stack_Size #
 *
#################################################################
##################
```

```
 * ^-- RAM start      ^-- _end                          _estack, RAM end --^
 * @endverbatim
 *
 * This implementation starts allocating at the '_end' linker symbol
 * The '_Min_Stack_Size' linker symbol reserves a memory for the MSP stack
 * The implementation considers '_estack' linker symbol to be RAM end
 * NOTE: If the MSP stack, at any point during execution, grows larger than the
 * reserved size, please increase the '_Min_Stack_Size'.
 *
 * @param incr Memory size
 * @return Pointer to allocated memory
 */
void *_sbrk(ptrdiff_t incr)
{
  extern uint8_t _end; /* Symbol defined in the linker script */
  extern uint8_t _estack; /* Symbol defined in the linker script */
  extern uint32_t _Min_Stack_Size; /* Symbol defined in the linker script */
  const uint32_t stack_limit = (uint32_t)&_estack - (uint32_t)&_Min_Stack_Size;
  const uint8_t *max_heap = (uint8_t *)stack_limit;
  uint8_t *prev_heap_end;
```

```
  /* Initialize heap end at first call */
  if (NULL == __sbrk_heap_end)
  {
    __sbrk_heap_end = &_end;
  }
  /* Protect heap from growing into the reserved MSP stack */
  if (__sbrk_heap_end + incr > max_heap)
  {
    errno = ENOMEM;
    return (void *)-1;
  }
  prev_heap_end = __sbrk_heap_end;
  __sbrk_heap_end += incr;
  return (void *)prev_heap_end;
}
```

24.17 CODE OF SYSTEM_STM32F4XX.C:

```
/**
*****************************************************************
*******************
  * @file    system_stm32f4xx.c
  * @author  MCD Application Team
  * @brief   CMSIS Cortex-M4 Device Peripheral Access Layer System
Source File.
```

```
  *
  *   This file provides two functions and one global variable to be called from
  *   user application:
  *      - SystemInit(): This function is called at startup just after reset and
  *                      before branch to main program. This call is made inside
  *                      the "startup_stm32f4xx.s" file.
  *
  *      - SystemCoreClock variable: Contains the core clock (HCLK), it can be used
  *                          by the user application to setup the SysTick
  *                          timer or configure other parameters.
  *
  *      - SystemCoreClockUpdate(): Updates the variable SystemCoreClock and must
  *                         be called whenever the core clock is changed
  *                         during program execution.
  *
  *
*******************************************************************************
  * @attention
  *
  * Copyright (c) 2017 STMicroelectronics.
```

```
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
*******************
  */
/** @addtogroup CMSIS
  * @{
  */
/** @addtogroup stm32f4xx_system
  * @{
  */
/** @addtogroup STM32F4xx_System_Private_Includes
  * @{
  */
#include "stm32f4xx.h"
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    ((uint32_t)25000000) /*!< Default value of
the External oscillator in Hz */
#endif /* HSE_VALUE */
```

```
#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000) /*!< Value of the
Internal oscillator in Hz*/
#endif /* HSI_VALUE */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_TypesDefinitions
  * @{
  */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Defines
  * @{
  */
/************************* Miscellaneous Configuration
************************/
/*!< Uncomment the following line if you need to use external SRAM or
SDRAM as data memory  */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx) || defined(STM32F417xx)\
 || defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
```

|| defined(STM32F469xx) || defined(STM32F479xx) || defined(STM32F412Zx) || defined(STM32F412Vx)

/* #define DATA_IN_ExtSRAM */

#endif /* STM32F40xxx || STM32F41xxx || STM32F42xxx || STM32F43xxx || STM32F469xx || STM32F479xx ||\

STM32F412Zx || STM32F412Vx */

#if defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) || defined(STM32F439xx)\

|| defined(STM32F446xx) || defined(STM32F469xx) || defined(STM32F479xx)

/* #define DATA_IN_ExtSDRAM */

#endif /* STM32F427xx || STM32F437xx || STM32F429xx || STM32F439xx || STM32F446xx || STM32F469xx ||\

STM32F479xx */

/* Note: Following vector table addresses must be defined in line with linker

configuration. */

/*!< Uncomment the following line if you need to relocate the vector table

anywhere in Flash or Sram, else the vector table is kept at the automatic

remap of boot address selected */

/* #define USER_VECT_TAB_ADDRESS */

#if defined(USER_VECT_TAB_ADDRESS)

/*!< Uncomment the following line if you need to relocate your vector Table

in Sram else user remap will be done in Flash. */

/* #define VECT_TAB_SRAM */

#if defined(VECT_TAB_SRAM)

#define VECT_TAB_BASE_ADDRESS SRAM_BASE /*!< Vector Table base address field.

This value must be a multiple of 0x200. */

#define VECT_TAB_OFFSET 0x00000000U /*!< Vector Table base offset field.

This value must be a multiple of 0x200. */

#else

#define VECT_TAB_BASE_ADDRESS FLASH_BASE /*!< Vector Table base address field.

This value must be a multiple of 0x200. */

#define VECT_TAB_OFFSET 0x00000000U /*!< Vector Table base offset field.

This value must be a multiple of 0x200. */

#endif /* VECT_TAB_SRAM */

#endif /* USER_VECT_TAB_ADDRESS */

/**********/

/**

* @}

```
  */
/** @addtogroup STM32F4xx_System_Private_Macros
  * @{
  */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Variables
  * @{
  */
  /* This variable is updated in three ways:
     1) by calling CMSIS function SystemCoreClockUpdate()
     2) by calling HAL API function HAL_RCC_GetHCLKFreq()
     3) each time HAL_RCC_ClockConfig() is called to configure the system clock frequency
        Note: If you use this function to configure the system clock; then there
              is no need to call the 2 first functions listed above, since SystemCoreClock
              variable is updated automatically.
  */
uint32_t SystemCoreClock = 16000000;
const uint8_t AHBPrescTable[16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9};
```

```
const uint8_t APBPrescTable[8]  = {0, 0, 0, 0, 1, 2, 3, 4};
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_FunctionPrototypes
  * @{
  */
#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  static void SystemInit_ExtMemCtl(void);
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Functions
  * @{
  */
/**
  * @brief  Setup the microcontroller system
  *         Initialize the FPU setting, vector table location and External memory
  *         configuration.
  * @param  None
  * @retval None
```

```
  */
void SystemInit(void)
{
  /* FPU settings ------------------------------------------------------------*/
  #if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
    SCB->CPACR |= ((3UL << 10*2)|(3UL << 11*2));  /* set CP10 and
CP11 Full Access */
  #endif
#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  SystemInit_ExtMemCtl();
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */
  /* Configure the Vector Table location -------------------------------------*/
#if defined(USER_VECT_TAB_ADDRESS)
  SCB->VTOR = VECT_TAB_BASE_ADDRESS |
VECT_TAB_OFFSET; /* Vector Table Relocation in Internal SRAM */
#endif /* USER_VECT_TAB_ADDRESS */
}

/**
   * @brief  Update SystemCoreClock variable according to Clock
Register Values.
  *         The SystemCoreClock variable contains the core clock (HCLK),
it can
  *         be used by the user application to setup the SysTick timer or
configure
```

```
  *         other parameters.
  *
  * @note   Each time the core clock (HCLK) changes, this function must be called
  *         to update SystemCoreClock variable value. Otherwise, any configuration
  *         based on this variable will be incorrect.
  *
  * @note   - The system frequency computed by this function is not the real
  *           frequency in the chip. It is calculated based on the predefined
  *           constant and the selected clock source:
  *
  *           - If SYSCLK source is HSI, SystemCoreClock will contain the HSI_VALUE(*)
  *
  *           - If SYSCLK source is HSE, SystemCoreClock will contain the HSE_VALUE(**)
  *
  *           - If SYSCLK source is PLL, SystemCoreClock will contain the HSE_VALUE(**)
  *             or HSI_VALUE(*) multiplied/divided by the PLL factors.
  *
  *         (*) HSI_VALUE is a constant defined in stm32f4xx_hal_conf.h file (default value
```

```
  *            16 MHz) but the real value may vary depending on the variations
  *            in voltage and temperature.
  *
  *        (**) HSE_VALUE is a constant defined in stm32f4xx_hal_conf.h file (its value
  *             depends on the application requirements), user has to ensure that HSE_VALUE
  *             is same as the real frequency of the crystal used. Otherwise, this function
  *             may have wrong result.
  *
  *        - The result of this function could be not correct when using fractional
  *          value for HSE crystal.
  *
  * @param  None
  * @retval None
  */
void SystemCoreClockUpdate(void)
{
  uint32_t tmp = 0, pllvco = 0, pllp = 2, pllsource = 0, pllm = 2;
  /* Get SYSCLK source -------------------------------------------------------*/
  tmp = RCC->CFGR & RCC_CFGR_SWS;
```

```
  switch (tmp)
  {
    case 0x00:  /* HSI used as system clock source */
      SystemCoreClock = HSI_VALUE;
      break;
    case 0x04:  /* HSE used as system clock source */
      SystemCoreClock = HSE_VALUE;
      break;
    case 0x08:  /* PLL used as system clock source */
      /* PLL_VCO = (HSE_VALUE or HSI_VALUE / PLL_M) * PLL_N
         SYSCLK = PLL_VCO / PLL_P
         */
      pllsource = (RCC->PLLCFGR & RCC_PLLCFGR_PLLSRC) >>
22;
      pllm = RCC->PLLCFGR & RCC_PLLCFGR_PLLM;
      if (pllsource != 0)
      {
        /* HSE used as PLL clock source */
        pllvco = (HSE_VALUE / pllm) * ((RCC->PLLCFGR &
RCC_PLLCFGR_PLLN) >> 6);
      }
      else
      {
        /* HSI used as PLL clock source */
```

```
      pllvco = (HSI_VALUE / pllm) * ((RCC->PLLCFGR &
RCC_PLLCFGR_PLLN) >> 6);
    }
    pllp = (((RCC->PLLCFGR & RCC_PLLCFGR_PLLP) >>16) + 1 )
*2;
    SystemCoreClock = pllvco/pllp;
    break;
   default:
    SystemCoreClock = HSI_VALUE;
    break;
  }
  /* Compute HCLK frequency ----------------------------------------------
-*/
  /* Get HCLK prescaler */
  tmp = AHBPrescTable[((RCC->CFGR & RCC_CFGR_HPRE) >> 4)];
  /* HCLK frequency */
  SystemCoreClock >>= tmp;
}
#if defined (DATA_IN_ExtSRAM) && defined
(DATA_IN_ExtSDRAM)
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
/**
  * @brief  Setup the external memory controller.
```

```
  *        Called in startup_stm32f4xx.s before jump to main.
  *        This function configures the external memories
(SRAM/SDRAM)
  *        This SRAM/SDRAM will be used as program data memory
(including heap and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
  register __IO uint32_t index;
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and
GPIOI interface clock */
  RCC->AHB1ENR |= 0x000001F8;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x00CCC0CC;
  GPIOD->AFR[1]  = 0xCCCCCCCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xAAAA0A8A;
  /* Configure PDx pins speed to 100 MHz */
```

```
GPIOD->OSPEEDR = 0xFFFF0FCF;
/* Configure PDx pins Output type to push-pull */
GPIOD->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PDx pins */
GPIOD->PUPDR   = 0x00000000;
/* Connect PEx pins to FMC Alternate function */
GPIOE->AFR[0]  = 0xC00CC0CC;
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA828A;
/* Configure PEx pins speed to 100 MHz */
GPIOE->OSPEEDR = 0xFFFFC3CF;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0xCCCCCCCC;
GPIOF->AFR[1]  = 0xCCCCCCCC;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA800AAA;
/* Configure PFx pins speed to 50 MHz */
GPIOF->OSPEEDR = 0xAA800AAA;
```

```
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
GPIOF->PUPDR   = 0x00000000;
/* Connect PGx pins to FMC Alternate function */
GPIOG->AFR[0]  = 0xCCCCCCCC;
GPIOG->AFR[1]  = 0xCCCCCCCC;
/* Configure PGx pins in Alternate function mode */
GPIOG->MODER   = 0xAAAAAAAA;
/* Configure PGx pins speed to 50 MHz */
GPIOG->OSPEEDR = 0xAAAAAAAA;
/* Configure PGx pins Output type to push-pull */
GPIOG->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PGx pins */
GPIOG->PUPDR   = 0x00000000;
/* Connect PHx pins to FMC Alternate function */
GPIOH->AFR[0]  = 0x00C0CC00;
GPIOH->AFR[1]  = 0xCCCCCCCC;
/* Configure PHx pins in Alternate function mode */
GPIOH->MODER   = 0xAAAA08A0;
/* Configure PHx pins speed to 50 MHz */
GPIOH->OSPEEDR = 0xAAAA08A0;
/* Configure PHx pins Output type to push-pull */
```

```
  GPIOH->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PHx pins */
  GPIOH->PUPDR   = 0x00000000;
  /* Connect PIx pins to FMC Alternate function */
  GPIOI->AFR[0]  = 0xCCCCCCCC;
  GPIOI->AFR[1]  = 0x00000CC0;
  /* Configure PIx pins in Alternate function mode */
  GPIOI->MODER   = 0x0028AAAA;
  /* Configure PIx pins speed to 50 MHz */
  GPIOI->OSPEEDR = 0x0028AAAA;
  /* Configure PIx pins Output type to push-pull */
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;
/*-- FMC Configuration ------------------------------------------------------*/
  /* Enable the FMC interface clock */
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
  FMC_Bank5_6->SDTR[0] = 0x01115351;
  /* SDRAM initialization sequence */
  /* Clock enable command */
```

```
FMC_Bank5_6->SDCMR = 0x00000011;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* Delay */
for (index = 0; index<1000; index++);
/* PALL command */
FMC_Bank5_6->SDCMR = 0x00000012;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* Auto refresh command */
FMC_Bank5_6->SDCMR = 0x00000073;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
```

```
  }
  /* MRD register program */
  FMC_Bank5_6->SDCMR = 0x00046014;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Set refresh count */
  tmpreg = FMC_Bank5_6->SDRTR;
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
  /* Disable write protection */
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
```

```
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F469xx || STM32F479xx */
  (void)(tmp);
}
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F469xx || STM32F479xx */
#elif defined (DATA_IN_ExtSRAM) || defined
(DATA_IN_ExtSDRAM)
/**
  * @brief  Setup the external memory controller.
  *         Called in startup_stm32f4xx.s before jump to main.
  *         This function configures the external memories
(SRAM/SDRAM)
  *         This SRAM/SDRAM will be used as program data memory
(including heap and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;
```

```
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F446xx) || defined(STM32F469xx) ||
defined(STM32F479xx)
#if defined (DATA_IN_ExtSDRAM)
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
  register __IO uint32_t index;
#if defined(STM32F446xx)
  /* Enable GPIOA, GPIOC, GPIOD, GPIOE, GPIOF, GPIOG interface
     clock */
  RCC->AHB1ENR |= 0x0000007D;
#else
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and
GPIOI interface
     clock */
  RCC->AHB1ENR |= 0x000001F8;
#endif /* STM32F446xx */
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);
#if defined(STM32F446xx)
  /* Connect PAx pins to FMC Alternate function */
  GPIOA->AFR[0]  |= 0xC0000000;
  GPIOA->AFR[1]  |= 0x00000000;
  /* Configure PDx pins in Alternate function mode */
```

```
  GPIOA->MODER   |= 0x00008000;
  /* Configure PDx pins speed to 50 MHz */
  GPIOA->OSPEEDR |= 0x00008000;
  /* Configure PDx pins Output type to push-pull */
  GPIOA->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOA->PUPDR   |= 0x00000000;
  /* Connect PCx pins to FMC Alternate function */
  GPIOC->AFR[0]  |= 0x00CC0000;
  GPIOC->AFR[1]  |= 0x00000000;
  /* Configure PDx pins in Alternate function mode */
  GPIOC->MODER   |= 0x00000A00;
  /* Configure PDx pins speed to 50 MHz */
  GPIOC->OSPEEDR |= 0x00000A00;
  /* Configure PDx pins Output type to push-pull */
  GPIOC->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOC->PUPDR   |= 0x00000000;
#endif /* STM32F446xx */
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x000000CC;
  GPIOD->AFR[1]  = 0xCC000CCC;
  /* Configure PDx pins in Alternate function mode */
```

```
GPIOD->MODER   = 0xA02A000A;
/* Configure PDx pins speed to 50 MHz */
GPIOD->OSPEEDR = 0xA02A000A;
/* Configure PDx pins Output type to push-pull */
GPIOD->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PDx pins */
GPIOD->PUPDR   = 0x00000000;
/* Connect PEx pins to FMC Alternate function */
GPIOE->AFR[0]  = 0xC00000CC;
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA800A;
/* Configure PEx pins speed to 50 MHz */
GPIOE->OSPEEDR = 0xAAAA800A;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0xCCCCCCCC;
GPIOF->AFR[1]  = 0xCCCCCCCC;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA800AAA;
```

```
  /* Configure PFx pins speed to 50 MHz */
  GPIOF->OSPEEDR = 0xAA800AAA;
  /* Configure PFx pins Output type to push-pull */
  GPIOF->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PFx pins */
  GPIOF->PUPDR   = 0x00000000;
  /* Connect PGx pins to FMC Alternate function */
  GPIOG->AFR[0]  = 0xCCCCCCCC;
  GPIOG->AFR[1]  = 0xCCCCCCCC;
  /* Configure PGx pins in Alternate function mode */
  GPIOG->MODER   = 0xAAAAAAAA;
  /* Configure PGx pins speed to 50 MHz */
  GPIOG->OSPEEDR = 0xAAAAAAAA;
  /* Configure PGx pins Output type to push-pull */
  GPIOG->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
  /* Connect PHx pins to FMC Alternate function */
  GPIOH->AFR[0]  = 0x00C0CC00;
  GPIOH->AFR[1]  = 0xCCCCCCCC;
```

```
  /* Configure PHx pins in Alternate function mode */
  GPIOH->MODER   = 0xAAAA08A0;
  /* Configure PHx pins speed to 50 MHz */
  GPIOH->OSPEEDR = 0xAAAA08A0;
  /* Configure PHx pins Output type to push-pull */
  GPIOH->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PHx pins */
  GPIOH->PUPDR   = 0x00000000;
  /* Connect PIx pins to FMC Alternate function */
  GPIOI->AFR[0]  = 0xCCCCCCCC;
  GPIOI->AFR[1]  = 0x00000CC0;
  /* Configure PIx pins in Alternate function mode */
  GPIOI->MODER   = 0x0028AAAA;
  /* Configure PIx pins speed to 50 MHz */
  GPIOI->OSPEEDR = 0x0028AAAA;
  /* Configure PIx pins Output type to push-pull */
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F469xx || STM32F479xx */
/*-- FMC Configuration ------------------------------------------------------*/
  /* Enable the FMC interface clock */
```

```
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable SDRAM bank1 */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCR[0] = 0x00001954;
#else
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
#endif /* STM32F446xx */
  FMC_Bank5_6->SDTR[0] = 0x01115351;
  /* SDRAM initialization sequence */
  /* Clock enable command */
  FMC_Bank5_6->SDCMR = 0x00000011;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Delay */
  for (index = 0; index<1000; index++);
  /* PALL command */
  FMC_Bank5_6->SDCMR = 0x00000012;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
```

```
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Auto refresh command */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x000000F3;
#else
  FMC_Bank5_6->SDCMR = 0x00000073;
#endif /* STM32F446xx */
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* MRD register program */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x00044014;
#else
  FMC_Bank5_6->SDCMR = 0x00046014;
#endif /* STM32F446xx */
```

```
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Set refresh count */
  tmpreg = FMC_Bank5_6->SDRTR;
#if defined(STM32F446xx)
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000050C<<1));
#else
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
#endif /* STM32F446xx */
  /* Disable write protection */
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#endif /* DATA_IN_ExtSDRAM */
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F446xx || STM32F469xx || STM32F479xx */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx) || defined(STM32F417xx)\
 || defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
```

```
 || defined(STM32F469xx) || defined(STM32F479xx) ||
defined(STM32F412Zx) || defined(STM32F412Vx)
#if defined(DATA_IN_ExtSRAM)
/*-- GPIOs Configuration -----------------------------------------------------*/
  /* Enable GPIOD, GPIOE, GPIOF and GPIOG interface clock */
  RCC->AHB1ENR   |= 0x00000078;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIODEN);
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x00CCC0CC;
  GPIOD->AFR[1]  = 0xCCCCCCCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xAAAA0A8A;
  /* Configure PDx pins speed to 100 MHz */
  GPIOD->OSPEEDR = 0xFFFF0FCF;
  /* Configure PDx pins Output type to push-pull */
  GPIOD->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOD->PUPDR   = 0x00000000;
  /* Connect PEx pins to FMC Alternate function */
  GPIOE->AFR[0]  = 0xC00CC0CC;
  GPIOE->AFR[1]  = 0xCCCCCCCC;
  /* Configure PEx pins in Alternate function mode */
```

```
GPIOE->MODER   = 0xAAAA828A;
/* Configure PEx pins speed to 100 MHz */
GPIOE->OSPEEDR = 0xFFFFC3CF;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0x00CCCCCC;
GPIOF->AFR[1]  = 0xCCCC0000;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA000AAA;
/* Configure PFx pins speed to 100 MHz */
GPIOF->OSPEEDR = 0xFF000FFF;
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
GPIOF->PUPDR   = 0x00000000;
/* Connect PGx pins to FMC Alternate function */
GPIOG->AFR[0]  = 0x00CCCCCC;
GPIOG->AFR[1]  = 0x000000C0;
/* Configure PGx pins in Alternate function mode */
GPIOG->MODER   = 0x00085AAA;
```

```
  /* Configure PGx pins speed to 100 MHz */
  GPIOG->OSPEEDR = 0x000CAFFF;
  /* Configure PGx pins Output type to push-pull */
  GPIOG->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;
/*-- FMC/FSMC Configuration ----------------------------------------------
-*/
  /* Enable the FMC/FSMC interface clock */
  RCC->AHB3ENR         |= 0x00000001;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
```

```
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F469xx || STM32F479xx */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx)|| defined(STM32F417xx)\
   || defined(STM32F412Zx) || defined(STM32F412Vx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FSMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FSMC_Bank1->BTCR[2]  = 0x00001011;
  FSMC_Bank1->BTCR[3]  = 0x00000201;
  FSMC_Bank1E->BWTR[2] = 0x0FFFFFFF;
#endif /* STM32F405xx || STM32F415xx || STM32F407xx ||
STM32F417xx || STM32F412Zx || STM32F412Vx */
#endif /* DATA_IN_ExtSRAM */
#endif /* STM32F405xx || STM32F415xx || STM32F407xx ||
STM32F417xx || STM32F427xx || STM32F437xx ||\
         STM32F429xx || STM32F439xx || STM32F469xx ||
STM32F479xx || STM32F412Zx || STM32F412Vx  */
  (void)(tmp);
}
#endif /* DATA_IN_ExtSRAM && DATA_IN_ExtSDRAM */
/**
```

```
  * @}
  */
/**
  * @}
  */
/**
  * @}
  */
```

CHAPTER#25 CODING A HARDWARE TIMER DRIVER FOR OUTPUT COMPARE FUNCTIONALITY

25.1CODING A HARDWARE TIMER DRIVER FOR OUTPUT COMPARE FUNCTIONALITY:

Output Compare functionality is used to generate specific waveforms (like PWM), toggle pins, or trigger interrupts at precise intervals. Below is an example of how to write a hardware timer driver with Output Compare functionality, using an STM32 microcontroller as a reference.

25.2 STEPS TO IMPLEMENT OUTPUT COMPARE:

25.2.1 CONFIGURE THE TIMER CLOCK:

Set the clock source and prescaler for the timer.

25.2.2 SET THE OUTPUT COMPARE MODE:

Choose the mode (toggle, active level, inactive level, or PWM).

25.2.3 SET THE OUTPUT COMPARE VALUE:

This is the value at which the timer will compare with the counter and take action.

25.5.4 ENABLE THE TIMER AND OUTPUT COMPARE CHANNEL:

Start the timer and enable the output compare channel.

25.5.5 HANDLE INTERRUPTS:

If using interrupts, set up the interrupt service routine.

25.6 CODE OF MAIN.H:

```
/* USER CODE BEGIN Header */
/**
**********************************************************
*******************
  * @file           : main.h
  * @brief          : Header for main.c file.
  *                   This file contains the common defines of the application.
**********************************************************
*******************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
```

```
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
********************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H
#ifdef __cplusplus
extern "C" {
#endif
/* Includes ------------------------------------------------------------------*/
#include "stm32f4xx_hal.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants --------------------------------------------------------*/
```

```
/* USER CODE BEGIN EC */
/* USER CODE END EC */
/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */
/* USER CODE END EM */
/* Exported functions prototypes ---------------------------------------------*/
void Error_Handler(void);
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
/* Private defines -----------------------------------------------------------*/
#define B1_Pin GPIO_PIN_13
#define B1_GPIO_Port GPIOC
#define USART_TX_Pin GPIO_PIN_2
#define USART_TX_GPIO_Port GPIOA
#define USART_RX_Pin GPIO_PIN_3
#define USART_RX_GPIO_Port GPIOA
#define LD2_Pin GPIO_PIN_5
#define LD2_GPIO_Port GPIOA
#define TMS_Pin GPIO_PIN_13
#define TMS_GPIO_Port GPIOA
#define TCK_Pin GPIO_PIN_14
#define TCK_GPIO_Port GPIOA
#define SWO_Pin GPIO_PIN_3
```

```
#define SWO_GPIO_Port GPIOB
/* USER CODE BEGIN Private defines */
/* USER CODE END Private defines */
#ifdef __cplusplus
}
#endif
#endif /* __MAIN_H */
```

25.7 CODE OF STM32F4XX_HAL_CONF.H:

```
/* USER CODE BEGIN Header */
/**
*******************************************************************
  * @file    stm32f4xx_hal_conf_template.h
  * @author  MCD Application Team
  * @brief   HAL configuration template file.
  *          This file should be copied to the application folder and renamed
  *          to stm32f4xx_hal_conf.h.
*******************************************************************
  * @attention
  *
  * Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.
```

```
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_HAL_CONF_H
#define __STM32F4xx_HAL_CONF_H
#ifdef __cplusplus
 extern "C" {
#endif
/* Exported types ------------------------------------------------------------*/
/* Exported constants --------------------------------------------------------*/
/* ############################ Module Selection
################################## */
/**
  * @brief This is the list of modules to be used in the HAL driver
  */
#define HAL_MODULE_ENABLED
```

```
  /* #define HAL_CRYP_MODULE_ENABLED */
/* #define HAL_ADC_MODULE_ENABLED */
/* #define HAL_CAN_MODULE_ENABLED */
/* #define HAL_CRC_MODULE_ENABLED */
/* #define HAL_CAN_LEGACY_MODULE_ENABLED */
/* #define HAL_DAC_MODULE_ENABLED */
/* #define HAL_DCMI_MODULE_ENABLED */
/* #define HAL_DMA2D_MODULE_ENABLED */
/* #define HAL_ETH_MODULE_ENABLED */
/* #define HAL_ETH_LEGACY_MODULE_ENABLED */
/* #define HAL_NAND_MODULE_ENABLED */
/* #define HAL_NOR_MODULE_ENABLED */
/* #define HAL_PCCARD_MODULE_ENABLED */
/* #define HAL_SRAM_MODULE_ENABLED */
/* #define HAL_SDRAM_MODULE_ENABLED */
/* #define HAL_HASH_MODULE_ENABLED */
/* #define HAL_I2C_MODULE_ENABLED */
/* #define HAL_I2S_MODULE_ENABLED */
/* #define HAL_IWDG_MODULE_ENABLED */
/* #define HAL_LTDC_MODULE_ENABLED */
/* #define HAL_RNG_MODULE_ENABLED */
/* #define HAL_RTC_MODULE_ENABLED */
/* #define HAL_SAI_MODULE_ENABLED */
```

```
/* #define HAL_SD_MODULE_ENABLED */
/* #define HAL_MMC_MODULE_ENABLED */
/* #define HAL_SPI_MODULE_ENABLED */
#define HAL_TIM_MODULE_ENABLED
#define HAL_UART_MODULE_ENABLED
/* #define HAL_USART_MODULE_ENABLED */
/* #define HAL_IRDA_MODULE_ENABLED */
/* #define HAL_SMARTCARD_MODULE_ENABLED */
/* #define HAL_SMBUS_MODULE_ENABLED */
/* #define HAL_WWDG_MODULE_ENABLED */
/* #define HAL_PCD_MODULE_ENABLED */
/* #define HAL_HCD_MODULE_ENABLED */
/* #define HAL_DSI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_CEC_MODULE_ENABLED */
/* #define HAL_FMPI2C_MODULE_ENABLED */
/* #define HAL_FMPSMBUS_MODULE_ENABLED */
/* #define HAL_SPDIFRX_MODULE_ENABLED */
/* #define HAL_DFSDM_MODULE_ENABLED */
/* #define HAL_LPTIM_MODULE_ENABLED */
#define HAL_GPIO_MODULE_ENABLED
#define HAL_EXTI_MODULE_ENABLED
```

```
#define HAL_DMA_MODULE_ENABLED
#define HAL_RCC_MODULE_ENABLED
#define HAL_FLASH_MODULE_ENABLED
#define HAL_PWR_MODULE_ENABLED
#define HAL_CORTEX_MODULE_ENABLED
/* ############################ HSE/HSI Values adaptation
###################### */
/**
  * @brief Adjust the value of External High Speed oscillator (HSE)
used in your application.
  *        This value is used by the RCC HAL module to compute the
system frequency
  *        (when HSE is used as system clock source, directly or through
the PLL).
  */
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    8000000U /*!< Value of the External
oscillator in Hz */
#endif /* HSE_VALUE */
#if !defined  (HSE_STARTUP_TIMEOUT)
  #define HSE_STARTUP_TIMEOUT    100U   /*!< Time out for
HSE start up, in ms */
#endif /* HSE_STARTUP_TIMEOUT */
/**
  * @brief Internal High Speed oscillator (HSI) value.
```

```
  *       This value is used by the RCC HAL module to compute the
system frequency
  *       (when HSI is used as system clock source, directly or through the
PLL).
  */
#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000U) /*!< Value of the
Internal oscillator in Hz*/
#endif /* HSI_VALUE */
/**
  * @brief Internal Low Speed oscillator (LSI) value.
  */
#if !defined  (LSI_VALUE)
 #define LSI_VALUE  32000U       /*!< LSI Typical Value in Hz*/
#endif /* LSI_VALUE */                    /*!< Value of the Internal Low
Speed oscillator in Hz
                                  The real value may vary depending on the
variations
                                  in voltage and temperature.*/
/**
  * @brief External Low Speed oscillator (LSE) value.
  */
#if !defined  (LSE_VALUE)
 #define LSE_VALUE  32768U    /*!< Value of the External Low Speed
oscillator in Hz */
```

```
#endif /* LSE_VALUE */
#if !defined  (LSE_STARTUP_TIMEOUT)
  #define LSE_STARTUP_TIMEOUT    5000U   /*!< Time out for
LSE start up, in ms */
#endif /* LSE_STARTUP_TIMEOUT */
/**
  * @brief External clock source for I2S peripheral
  *        This value is used by the I2S HAL module to compute the I2S
clock source
  *        frequency, this source is inserted directly through I2S_CKIN
pad.
  */
#if !defined  (EXTERNAL_CLOCK_VALUE)
  #define EXTERNAL_CLOCK_VALUE    12288000U /*!< Value of
the External audio frequency in Hz*/
#endif /* EXTERNAL_CLOCK_VALUE */
/* Tip: To avoid modifying this file each time you need to use different
HSE,
   ===  you can define the HSE value in your toolchain compiler
preprocessor. */
/* ############################## System Configuration
########################### */
/**
  * @brief This is the HAL system configuration section
  */
```

```
#define  VDD_VALUE                    3300U /*!< Value of VDD in mv */
#define  TICK_INT_PRIORITY            0U   /*!< tick interrupt
priority */
#define  USE_RTOS                     0U
#define  PREFETCH_ENABLE              1U
#define  INSTRUCTION_CACHE_ENABLE     1U
#define  DATA_CACHE_ENABLE            1U
#define  USE_HAL_ADC_REGISTER_CALLBACKS        0U /*
ADC register callback disabled       */
#define  USE_HAL_CAN_REGISTER_CALLBACKS        0U /*
CAN register callback disabled       */
#define  USE_HAL_CEC_REGISTER_CALLBACKS        0U /*
CEC register callback disabled       */
#define  USE_HAL_CRYP_REGISTER_CALLBACKS       0U /*
CRYP register callback disabled      */
#define  USE_HAL_DAC_REGISTER_CALLBACKS        0U /*
DAC register callback disabled       */
#define  USE_HAL_DCMI_REGISTER_CALLBACKS       0U /*
DCMI register callback disabled      */
#define  USE_HAL_DFSDM_REGISTER_CALLBACKS      0U /*
DFSDM register callback disabled     */
#define  USE_HAL_DMA2D_REGISTER_CALLBACKS      0U /*
DMA2D register callback disabled     */
#define  USE_HAL_DSI_REGISTER_CALLBACKS        0U /* DSI
register callback disabled       */
```

#define USE_HAL_ETH_REGISTER_CALLBACKS 0U /* ETH register callback disabled */

#define USE_HAL_HASH_REGISTER_CALLBACKS 0U /* HASH register callback disabled */

#define USE_HAL_HCD_REGISTER_CALLBACKS 0U /* HCD register callback disabled */

#define USE_HAL_I2C_REGISTER_CALLBACKS 0U /* I2C register callback disabled */

#define USE_HAL_FMPI2C_REGISTER_CALLBACKS 0U /* FMPI2C register callback disabled */

#define USE_HAL_FMPSMBUS_REGISTER_CALLBACKS 0U /* FMPSMBUS register callback disabled */

#define USE_HAL_I2S_REGISTER_CALLBACKS 0U /* I2S register callback disabled */

#define USE_HAL_IRDA_REGISTER_CALLBACKS 0U /* IRDA register callback disabled */

#define USE_HAL_LPTIM_REGISTER_CALLBACKS 0U /* LPTIM register callback disabled */

#define USE_HAL_LTDC_REGISTER_CALLBACKS 0U /* LTDC register callback disabled */

#define USE_HAL_MMC_REGISTER_CALLBACKS 0U /* MMC register callback disabled */

#define USE_HAL_NAND_REGISTER_CALLBACKS 0U /* NAND register callback disabled */

#define USE_HAL_NOR_REGISTER_CALLBACKS 0U /* NOR register callback disabled */

#define USE_HAL_PCCARD_REGISTER_CALLBACKS 0U /* PCCARD register callback disabled */

#define USE_HAL_PCD_REGISTER_CALLBACKS 0U /* PCD register callback disabled */

#define USE_HAL_QSPI_REGISTER_CALLBACKS 0U /* QSPI register callback disabled */

#define USE_HAL_RNG_REGISTER_CALLBACKS 0U /* RNG register callback disabled */

#define USE_HAL_RTC_REGISTER_CALLBACKS 0U /* RTC register callback disabled */

#define USE_HAL_SAI_REGISTER_CALLBACKS 0U /* SAI register callback disabled */

#define USE_HAL_SD_REGISTER_CALLBACKS 0U /* SD register callback disabled */

#define USE_HAL_SMARTCARD_REGISTER_CALLBACKS 0U /* SMARTCARD register callback disabled */

#define USE_HAL_SDRAM_REGISTER_CALLBACKS 0U /* SDRAM register callback disabled */

#define USE_HAL_SRAM_REGISTER_CALLBACKS 0U /* SRAM register callback disabled */

#define USE_HAL_SPDIFRX_REGISTER_CALLBACKS 0U /* SPDIFRX register callback disabled */

#define USE_HAL_SMBUS_REGISTER_CALLBACKS 0U /* SMBUS register callback disabled */

#define USE_HAL_SPI_REGISTER_CALLBACKS 0U /* SPI register callback disabled */

```
#define  USE_HAL_TIM_REGISTER_CALLBACKS        0U /*
TIM register callback disabled       */

#define  USE_HAL_UART_REGISTER_CALLBACKS       0U /*
UART register callback disabled      */

#define  USE_HAL_USART_REGISTER_CALLBACKS      0U /*
USART register callback disabled     */

#define  USE_HAL_WWDG_REGISTER_CALLBACKS       0U /*
WWDG register callback disabled      */

/* ############################ Assert Selection
################################## */

/**

  * @brief Uncomment the line below to expanse the "assert_param"
macro in the

  *        HAL drivers code

  */

/* #define USE_FULL_ASSERT    1U */

/* ################### Ethernet peripheral configuration
###################### */

/* Section 1 : Ethernet peripheral configuration */

/* MAC ADDRESS:
MAC_ADDR0:MAC_ADDR1:MAC_ADDR2:MAC_ADDR3:MAC_A
DDR4:MAC_ADDR5 */

#define MAC_ADDR0   2U

#define MAC_ADDR1   0U

#define MAC_ADDR2   0U

#define MAC_ADDR3   0U
```

```
#define MAC_ADDR4   0U
#define MAC_ADDR5   0U
/* Definition of the Ethernet driver buffers size and count */
#define ETH_RX_BUF_SIZE             ETH_MAX_PACKET_SIZE
/* buffer size for receive             */
#define ETH_TX_BUF_SIZE             ETH_MAX_PACKET_SIZE
/* buffer size for transmit            */
#define ETH_RXBUFNB                 4U      /* 4 Rx buffers of size
ETH_RX_BUF_SIZE  */
#define ETH_TXBUFNB                 4U      /* 4 Tx buffers of size
ETH_TX_BUF_SIZE  */
/* Section 2: PHY configuration section */
/* DP83848_PHY_ADDRESS Address*/
#define DP83848_PHY_ADDRESS
/* PHY Reset delay these values are based on a 1 ms Systick interrupt*/
#define PHY_RESET_DELAY             0x000000FFU
/* PHY Configuration delay */
#define PHY_CONFIG_DELAY            0x00000FFFU
#define PHY_READ_TO                 0x0000FFFFU
#define PHY_WRITE_TO                0x0000FFFFU
/* Section 3: Common PHY Registers */
#define PHY_BCR                     ((uint16_t)0x0000U)    /*!<
Transceiver Basic Control Register   */
#define PHY_BSR                     ((uint16_t)0x0001U)    /*!<
Transceiver Basic Status Register    */
```

```
#define PHY_RESET                    ((uint16_t)0x8000U)  /*!< PHY
Reset */

#define PHY_LOOPBACK                 ((uint16_t)0x4000U)  /*!<
Select loop-back mode */

#define PHY_FULLDUPLEX_100M          ((uint16_t)0x2100U)
/*!< Set the full-duplex mode at 100 Mb/s */

#define PHY_HALFDUPLEX_100M          ((uint16_t)0x2000U)
/*!< Set the half-duplex mode at 100 Mb/s */

#define PHY_FULLDUPLEX_10M           ((uint16_t)0x0100U)
/*!< Set the full-duplex mode at 10 Mb/s  */

#define PHY_HALFDUPLEX_10M           ((uint16_t)0x0000U)
/*!< Set the half-duplex mode at 10 Mb/s  */

#define PHY_AUTONEGOTIATION          ((uint16_t)0x1000U)
/*!< Enable auto-negotiation function     */

#define PHY_RESTART_AUTONEGOTIATION
((uint16_t)0x0200U)  /*!< Restart auto-negotiation function    */

#define PHY_POWERDOWN                ((uint16_t)0x0800U)  /*!<
Select the power down mode           */

#define PHY_ISOLATE                  ((uint16_t)0x0400U)  /*!< Isolate
PHY from MII                 */

#define PHY_AUTONEGO_COMPLETE        ((uint16_t)0x0020U)
/*!< Auto-Negotiation process completed   */

#define PHY_LINKED_STATUS            ((uint16_t)0x0004U)  /*!<
Valid link established               */

#define PHY_JABBER_DETECTION         ((uint16_t)0x0002U)
/*!< Jabber condition detected            */

/* Section 4: Extended PHY Registers */
```

```
#define PHY_SR                    ((uint16_t))    /*!< PHY status register
Offset                  */

#define PHY_SPEED_STATUS              ((uint16_t))  /*!< PHY Speed
mask                          */

#define PHY_DUPLEX_STATUS             ((uint16_t))  /*!< PHY
Duplex mask                           */

/* ################### SPI peripheral configuration
############################ */

/* CRC FEATURE: Use to activate CRC feature inside HAL SPI Driver

* Activated: CRC code is present inside driver

* Deactivated: CRC code cleaned from driver

*/

#define USE_SPI_CRC                   0U

/* Includes ------------------------------------------------------------------*/

/**

  * @brief Include module's header file

  */

#ifdef HAL_RCC_MODULE_ENABLED

  #include "stm32f4xx_hal_rcc.h"

#endif /* HAL_RCC_MODULE_ENABLED */

#ifdef HAL_GPIO_MODULE_ENABLED

  #include "stm32f4xx_hal_gpio.h"

#endif /* HAL_GPIO_MODULE_ENABLED */

#ifdef HAL_EXTI_MODULE_ENABLED
```

```
  #include "stm32f4xx_hal_exti.h"
#endif /* HAL_EXTI_MODULE_ENABLED */
#ifdef HAL_DMA_MODULE_ENABLED
  #include "stm32f4xx_hal_dma.h"
#endif /* HAL_DMA_MODULE_ENABLED */
#ifdef HAL_CORTEX_MODULE_ENABLED
  #include "stm32f4xx_hal_cortex.h"
#endif /* HAL_CORTEX_MODULE_ENABLED */
#ifdef HAL_ADC_MODULE_ENABLED
  #include "stm32f4xx_hal_adc.h"
#endif /* HAL_ADC_MODULE_ENABLED */
#ifdef HAL_CAN_MODULE_ENABLED
  #include "stm32f4xx_hal_can.h"
#endif /* HAL_CAN_MODULE_ENABLED */
#ifdef HAL_CAN_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_can_legacy.h"
#endif /* HAL_CAN_LEGACY_MODULE_ENABLED */
#ifdef HAL_CRC_MODULE_ENABLED
  #include "stm32f4xx_hal_crc.h"
#endif /* HAL_CRC_MODULE_ENABLED */
#ifdef HAL_CRYP_MODULE_ENABLED
  #include "stm32f4xx_hal_cryp.h"
#endif /* HAL_CRYP_MODULE_ENABLED */
```

```
#ifdef HAL_DMA2D_MODULE_ENABLED
  #include "stm32f4xx_hal_dma2d.h"
#endif /* HAL_DMA2D_MODULE_ENABLED */
#ifdef HAL_DAC_MODULE_ENABLED
  #include "stm32f4xx_hal_dac.h"
#endif /* HAL_DAC_MODULE_ENABLED */
#ifdef HAL_DCMI_MODULE_ENABLED
  #include "stm32f4xx_hal_dcmi.h"
#endif /* HAL_DCMI_MODULE_ENABLED */
#ifdef HAL_ETH_MODULE_ENABLED
  #include "stm32f4xx_hal_eth.h"
#endif /* HAL_ETH_MODULE_ENABLED */
#ifdef HAL_ETH_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_eth_legacy.h"
#endif /* HAL_ETH_LEGACY_MODULE_ENABLED */
#ifdef HAL_FLASH_MODULE_ENABLED
  #include "stm32f4xx_hal_flash.h"
#endif /* HAL_FLASH_MODULE_ENABLED */
#ifdef HAL_SRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sram.h"
#endif /* HAL_SRAM_MODULE_ENABLED */
#ifdef HAL_NOR_MODULE_ENABLED
  #include "stm32f4xx_hal_nor.h"
```

```
#endif /* HAL_NOR_MODULE_ENABLED */
#ifdef HAL_NAND_MODULE_ENABLED
  #include "stm32f4xx_hal_nand.h"
#endif /* HAL_NAND_MODULE_ENABLED */
#ifdef HAL_PCCARD_MODULE_ENABLED
  #include "stm32f4xx_hal_pccard.h"
#endif /* HAL_PCCARD_MODULE_ENABLED */
#ifdef HAL_SDRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sdram.h"
#endif /* HAL_SDRAM_MODULE_ENABLED */
#ifdef HAL_HASH_MODULE_ENABLED
 #include "stm32f4xx_hal_hash.h"
#endif /* HAL_HASH_MODULE_ENABLED */
#ifdef HAL_I2C_MODULE_ENABLED
 #include "stm32f4xx_hal_i2c.h"
#endif /* HAL_I2C_MODULE_ENABLED */
#ifdef HAL_SMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_smbus.h"
#endif /* HAL_SMBUS_MODULE_ENABLED */
#ifdef HAL_I2S_MODULE_ENABLED
 #include "stm32f4xx_hal_i2s.h"
#endif /* HAL_I2S_MODULE_ENABLED */
#ifdef HAL_IWDG_MODULE_ENABLED
```

```
 #include "stm32f4xx_hal_iwdg.h"
#endif /* HAL_IWDG_MODULE_ENABLED */
#ifdef HAL_LTDC_MODULE_ENABLED
 #include "stm32f4xx_hal_ltdc.h"
#endif /* HAL_LTDC_MODULE_ENABLED */
#ifdef HAL_PWR_MODULE_ENABLED
 #include "stm32f4xx_hal_pwr.h"
#endif /* HAL_PWR_MODULE_ENABLED */
#ifdef HAL_RNG_MODULE_ENABLED
 #include "stm32f4xx_hal_rng.h"
#endif /* HAL_RNG_MODULE_ENABLED */
#ifdef HAL_RTC_MODULE_ENABLED
 #include "stm32f4xx_hal_rtc.h"
#endif /* HAL_RTC_MODULE_ENABLED */
#ifdef HAL_SAI_MODULE_ENABLED
 #include "stm32f4xx_hal_sai.h"
#endif /* HAL_SAI_MODULE_ENABLED */
#ifdef HAL_SD_MODULE_ENABLED
 #include "stm32f4xx_hal_sd.h"
#endif /* HAL_SD_MODULE_ENABLED */
#ifdef HAL_SPI_MODULE_ENABLED
 #include "stm32f4xx_hal_spi.h"
#endif /* HAL_SPI_MODULE_ENABLED */
```

```
#ifdef HAL_TIM_MODULE_ENABLED
 #include "stm32f4xx_hal_tim.h"
#endif /* HAL_TIM_MODULE_ENABLED */
#ifdef HAL_UART_MODULE_ENABLED
 #include "stm32f4xx_hal_uart.h"
#endif /* HAL_UART_MODULE_ENABLED */
#ifdef HAL_USART_MODULE_ENABLED
 #include "stm32f4xx_hal_usart.h"
#endif /* HAL_USART_MODULE_ENABLED */
#ifdef HAL_IRDA_MODULE_ENABLED
 #include "stm32f4xx_hal_irda.h"
#endif /* HAL_IRDA_MODULE_ENABLED */
#ifdef HAL_SMARTCARD_MODULE_ENABLED
 #include "stm32f4xx_hal_smartcard.h"
#endif /* HAL_SMARTCARD_MODULE_ENABLED */
#ifdef HAL_WWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_wwdg.h"
#endif /* HAL_WWDG_MODULE_ENABLED */
#ifdef HAL_PCD_MODULE_ENABLED
 #include "stm32f4xx_hal_pcd.h"
#endif /* HAL_PCD_MODULE_ENABLED */
#ifdef HAL_HCD_MODULE_ENABLED
 #include "stm32f4xx_hal_hcd.h"
```

```
#endif /* HAL_HCD_MODULE_ENABLED */
#ifdef HAL_DSI_MODULE_ENABLED
 #include "stm32f4xx_hal_dsi.h"
#endif /* HAL_DSI_MODULE_ENABLED */
#ifdef HAL_QSPI_MODULE_ENABLED
 #include "stm32f4xx_hal_qspi.h"
#endif /* HAL_QSPI_MODULE_ENABLED */
#ifdef HAL_CEC_MODULE_ENABLED
 #include "stm32f4xx_hal_cec.h"
#endif /* HAL_CEC_MODULE_ENABLED */
#ifdef HAL_FMPI2C_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpi2c.h"
#endif /* HAL_FMPI2C_MODULE_ENABLED */
#ifdef HAL_FMPSMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpsmbus.h"
#endif /* HAL_FMPSMBUS_MODULE_ENABLED */
#ifdef HAL_SPDIFRX_MODULE_ENABLED
 #include "stm32f4xx_hal_spdifrx.h"
#endif /* HAL_SPDIFRX_MODULE_ENABLED */
#ifdef HAL_DFSDM_MODULE_ENABLED
 #include "stm32f4xx_hal_dfsdm.h"
#endif /* HAL_DFSDM_MODULE_ENABLED */
#ifdef HAL_LPTIM_MODULE_ENABLED
```

```
 #include "stm32f4xx_hal_lptim.h"
#endif /* HAL_LPTIM_MODULE_ENABLED */
#ifdef HAL_MMC_MODULE_ENABLED
 #include "stm32f4xx_hal_mmc.h"
#endif /* HAL_MMC_MODULE_ENABLED */
/* Exported macro ------------------------------------------------------------*/
#ifdef  USE_FULL_ASSERT
/**
  * @brief  The assert_param macro is used for function's parameters check.
  * @param  expr If expr is false, it calls assert_failed function
  *         which reports the name of the source file and the source
  *         line number of the call that failed.
  *         If expr is true, it returns no value.
  * @retval None
  */
  #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t *)__FILE__, __LINE__))
/* Exported functions ------------------------------------------------------- */
  void assert_failed(uint8_t* file, uint32_t line);
#else
  #define assert_param(expr) ((void)0U)
#endif /* USE_FULL_ASSERT */
#ifdef __cplusplus
```

```
}
#endif
#endif /* __STM32F4xx_HAL_CONF_H */
```

25.8 CODE OF STMF4XX_IT.H:

```
/* USER CODE BEGIN Header */
/**
**********************************************************
********************
  * @file    stm32f4xx_it.h
  * @brief   This file contains the headers of the interrupt handlers.
**********************************************************
********************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
```

```
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_IT_H
#define __STM32F4xx_IT_H
#ifdef __cplusplus
 extern "C" {
#endif
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */
/* USER CODE END EC */
/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */
/* USER CODE END EM */
/* Exported functions prototypes ---------------------------------------------*/
```

```
void NMI_Handler(void);
void HardFault_Handler(void);
void MemManage_Handler(void);
void BusFault_Handler(void);
void UsageFault_Handler(void);
void SVC_Handler(void);
void DebugMon_Handler(void);
void PendSV_Handler(void);
void SysTick_Handler(void);
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
#ifdef __cplusplus
}
#endif
#endif /* __STM32F4xx_IT_H */
```

25.9 CODE OF MAIN.C:

```
/* USER CODE BEGIN Header */
/**
*****************************************************************
*******************
  * @file           : main.c
  * @brief          : Main program body
*****************************************************************
*******************
```

```
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
********************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
/* USER CODE END PTD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
```

```
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
TIM_HandleTypeDef htim2;
UART_HandleTypeDef huart2;
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM2_Init(void);
static void MX_USART2_UART_Init(void);
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/**
  * @brief  The application entry point.
  * @retval int
  */
```

```
int main(void)
{
  /* USER CODE BEGIN 1 */
  /* USER CODE END 1 */
  /* MCU Configuration--------------------------------------------------------*/
  /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
  HAL_Init();
  /* USER CODE BEGIN Init */
  /* USER CODE END Init */
  /* Configure the system clock */
  SystemClock_Config();
  /* USER CODE BEGIN SysInit */
  /* USER CODE END SysInit */
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_TIM2_Init();
  MX_USART2_UART_Init();
  /* USER CODE BEGIN 2 */
  /* USER CODE END 2 */
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
```

```
  {
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_
VOLTAGE_SCALE1);
  /** Initializes the RCC Oscillators according to the specified
parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
```

```
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue =
RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 8;
  RCC_OscInitStruct.PLL.PLLN = 96;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource =
RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct,
FLASH_LATENCY_3) != HAL_OK)
```

```
  {
    Error_Handler();
  }
}
/**
  * @brief TIM2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM2_Init(void)
{
  /* USER CODE BEGIN TIM2_Init 0 */
  /* USER CODE END TIM2_Init 0 */
  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};
  /* USER CODE BEGIN TIM2_Init 1 */
  /* USER CODE END TIM2_Init 1 */
  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 48000;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 4294967295;
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
```

```
  htim2.Init.AutoReloadPreload =
TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource =
TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig)
!= HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode =
TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim2,
&sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM2_Init 2 */
  /* USER CODE END TIM2_Init 2 */
}
/**
```

```
  * @brief USART2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART2_UART_Init(void)
{
  /* USER CODE BEGIN USART2_Init 0 */
  /* USER CODE END USART2_Init 0 */
  /* USER CODE BEGIN USART2_Init 1 */
  /* USER CODE END USART2_Init 1 */
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 115200;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART2_Init 2 */
```

```
  /* USER CODE END USART2_Init 2 */
}
/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */
  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin,
GPIO_PIN_RESET);
  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
```

```
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);
  /*Configure GPIO pin : LD2_Pin */
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);
/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}
/* USER CODE BEGIN 4 */
/* USER CODE END 4 */
/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return
state */
  __disable_irq();
```

```
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}
#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

25.10 CODE OF STM32F4XX_HAL_MSP.C:

```
/* USER CODE BEGIN Header */
/**
*****************************************************************************
  * @file       stm32f4xx_hal_msp.c
  * @brief      This file provides code for the MSP Initialization
  *             and de-Initialization codes.
*****************************************************************************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*****************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
```

```
#include "main.h"
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN TD */
/* USER CODE END TD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN Define */
/* USER CODE END Define */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN Macro */
/* USER CODE END Macro */
/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* External functions --------------------------------------------------------*/
/* USER CODE BEGIN ExternalFunctions */
/* USER CODE END ExternalFunctions */
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
```

```
/**
  * Initializes the Global MSP.
  */
void HAL_MspInit(void)
{
  /* USER CODE BEGIN MspInit 0 */
  /* USER CODE END MspInit 0 */
  __HAL_RCC_SYSCFG_CLK_ENABLE();
  __HAL_RCC_PWR_CLK_ENABLE();
  HAL_NVIC_SetPriorityGrouping(NVIC_PRIORITYGROUP_0);
  /* System interrupt init*/
  /* USER CODE BEGIN MspInit 1 */
  /* USER CODE END MspInit 1 */
}
/**
* @brief TIM_Base MSP Initialization
* This function configures the hardware resources used in this example
* @param htim_base: TIM_Base handle pointer
* @retval None
*/
void HAL_TIM_Base_MspInit(TIM_HandleTypeDef* htim_base)
{
  if(htim_base->Instance==TIM2)
```

```
  {
  /* USER CODE BEGIN TIM2_MspInit 0 */
  /* USER CODE END TIM2_MspInit 0 */
    /* Peripheral clock enable */
    __HAL_RCC_TIM2_CLK_ENABLE();
  /* USER CODE BEGIN TIM2_MspInit 1 */
  /* USER CODE END TIM2_MspInit 1 */
  }
}
/**
* @brief TIM_Base MSP De-Initialization
* This function freeze the hardware resources used in this example
* @param htim_base: TIM_Base handle pointer
* @retval None
*/
void HAL_TIM_Base_MspDeInit(TIM_HandleTypeDef* htim_base)
{
  if(htim_base->Instance==TIM2)
  {
  /* USER CODE BEGIN TIM2_MspDeInit 0 */
  /* USER CODE END TIM2_MspDeInit 0 */
    /* Peripheral clock disable */
    __HAL_RCC_TIM2_CLK_DISABLE();
```

```
  /* USER CODE BEGIN TIM2_MspDeInit 1 */
  /* USER CODE END TIM2_MspDeInit 1 */
  }
}
/**
* @brief UART MSP Initialization
* This function configures the hardware resources used in this example
* @param huart: UART handle pointer
* @retval None
*/
void HAL_UART_MspInit(UART_HandleTypeDef* huart)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  if(huart->Instance==USART2)
  {
  /* USER CODE BEGIN USART2_MspInit 0 */
  /* USER CODE END USART2_MspInit 0 */
    /* Peripheral clock enable */
    __HAL_RCC_USART2_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();
    /**USART2 GPIO Configuration
    PA2     ------> USART2_TX
    PA3     ------> USART2_RX
```

```
    */
    GPIO_InitStruct.Pin = USART_TX_Pin|USART_RX_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF7_USART2;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
  /* USER CODE BEGIN USART2_MspInit 1 */
  /* USER CODE END USART2_MspInit 1 */
  }
}
/**
* @brief UART MSP De-Initialization
* This function freeze the hardware resources used in this example
* @param huart: UART handle pointer
* @retval None
*/
void HAL_UART_MspDeInit(UART_HandleTypeDef* huart)
{
  if(huart->Instance==USART2)
  {
  /* USER CODE BEGIN USART2_MspDeInit 0 */
  /* USER CODE END USART2_MspDeInit 0 */
```

```
    /* Peripheral clock disable */
    __HAL_RCC_USART2_CLK_DISABLE();
    /**USART2 GPIO Configuration
    PA2     ------> USART2_TX
    PA3     ------> USART2_RX
    */
    HAL_GPIO_DeInit(GPIOA, USART_TX_Pin|USART_RX_Pin);
  /* USER CODE BEGIN USART2_MspDeInit 1 */
  /* USER CODE END USART2_MspDeInit 1 */
  }
}
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

25.11 CODE OF STM32F4XX_IT.C:

```
/* USER CODE BEGIN Header */
/**
*******************************************************
********************
  * @file    stm32f4xx_it.c
  * @brief   Interrupt Service Routines.
*******************************************************
********************
  * @attention
  *
```

```
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*****************************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stm32f4xx_it.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN TD */
/* USER CODE END TD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
```

```
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/* External variables --------------------------------------------------------*/
/* USER CODE BEGIN EV */
/* USER CODE END EV */
/******************************************************************************/
/*           Cortex-M4 Processor Interruption and Exception Handlers          */
/******************************************************************************/
/**
  * @brief This function handles Non maskable interrupt.
  */
```

```
void NMI_Handler(void)
{
  /* USER CODE BEGIN NonMaskableInt_IRQn 0 */
  /* USER CODE END NonMaskableInt_IRQn 0 */
  /* USER CODE BEGIN NonMaskableInt_IRQn 1 */
   while (1)
  {
  }
  /* USER CODE END NonMaskableInt_IRQn 1 */
}
/**
  * @brief This function handles Hard fault interrupt.
  */
void HardFault_Handler(void)
{
  /* USER CODE BEGIN HardFault_IRQn 0 */
  /* USER CODE END HardFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_HardFault_IRQn 0 */
    /* USER CODE END W1_HardFault_IRQn 0 */
  }
}
```

```
/**
  * @brief This function handles Memory management fault.
  */
void MemManage_Handler(void)
{
  /* USER CODE BEGIN MemoryManagement_IRQn 0 */
  /* USER CODE END MemoryManagement_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_MemoryManagement_IRQn 0 */
    /* USER CODE END W1_MemoryManagement_IRQn 0 */
  }
}
/**
  * @brief This function handles Pre-fetch fault, memory access fault.
  */
void BusFault_Handler(void)
{
  /* USER CODE BEGIN BusFault_IRQn 0 */
  /* USER CODE END BusFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_BusFault_IRQn 0 */
```

```
    /* USER CODE END W1_BusFault_IRQn 0 */
  }
}
/**
  * @brief This function handles Undefined instruction or illegal state.
  */
void UsageFault_Handler(void)
{
  /* USER CODE BEGIN UsageFault_IRQn 0 */
  /* USER CODE END UsageFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_UsageFault_IRQn 0 */
    /* USER CODE END W1_UsageFault_IRQn 0 */
  }
}
/**
  * @brief This function handles System service call via SWI
instruction.
  */
void SVC_Handler(void)
{
  /* USER CODE BEGIN SVCall_IRQn 0 */
```

```
  /* USER CODE END SVCall_IRQn 0 */
  /* USER CODE BEGIN SVCall_IRQn 1 */
  /* USER CODE END SVCall_IRQn 1 */
}
/**
  * @brief This function handles Debug monitor.
  */
void DebugMon_Handler(void)
{
  /* USER CODE BEGIN DebugMonitor_IRQn 0 */
  /* USER CODE END DebugMonitor_IRQn 0 */
  /* USER CODE BEGIN DebugMonitor_IRQn 1 */
  /* USER CODE END DebugMonitor_IRQn 1 */
}
/**
  * @brief This function handles Pendable request for system service.
  */
void PendSV_Handler(void)
{
  /* USER CODE BEGIN PendSV_IRQn 0 */
  /* USER CODE END PendSV_IRQn 0 */
  /* USER CODE BEGIN PendSV_IRQn 1 */
  /* USER CODE END PendSV_IRQn 1 */
```

```
}
/**
  * @brief This function handles System tick timer.
  */
void SysTick_Handler(void)
{
  /* USER CODE BEGIN SysTick_IRQn 0 */
  /* USER CODE END SysTick_IRQn 0 */
  HAL_IncTick();
  /* USER CODE BEGIN SysTick_IRQn 1 */
  /* USER CODE END SysTick_IRQn 1 */
}
/******************************************************************************/
/* STM32F4xx Peripheral Interrupt Handlers                                    */
/* Add here the Interrupt Handlers for the used peripherals.                  */
/* For the available peripheral interrupt handler names,                      */
/* please refer to the startup file (startup_stm32f4xx.s).                   */
/******************************************************************************/
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

25.12 CODE OF SYSCALLS.C:

```
/**
**********************************************************
*******************
 * @file      syscalls.c
 * @author    Auto-generated by STM32CubeIDE
 * @brief     STM32CubeIDE Minimal System calls file
 *
 *            For more information about which c-functions
 *            need which of these lowlevel functions
 *            please consult the Newlib libc-manual
**********************************************************
*******************
 * @attention
 *
 * Copyright (c) 2020-2024 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the
LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
**********************************************************
*******************
```

```
 */
/* Includes */
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
#include <sys/times.h>
/* Variables */
extern int __io_putchar(int ch) __attribute__((weak));
extern int __io_getchar(void) __attribute__((weak));
char *__env[1] = { 0 };
char **environ = __env;
/* Functions */
void initialise_monitor_handles()
{
}
int _getpid(void)
{
  return 1;
}
```

```
int _kill(int pid, int sig)
{
  (void)pid;
  (void)sig;
  errno = EINVAL;
  return -1;
}
void _exit (int status)
{
  _kill(status, -1);
  while (1) {}    /* Make sure we hang here */
}
__attribute__((weak)) int _read(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;
  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    *ptr++ = __io_getchar();
  }
  return len;
}
__attribute__((weak)) int _write(int file, char *ptr, int len)
```

```
{
  (void)file;
  int DataIdx;
  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    __io_putchar(*ptr++);
  }
  return len;
}
int _close(int file)
{
  (void)file;
  return -1;
}
int _fstat(int file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
int _isatty(int file)
{
  (void)file;
```

```
  return 1;
}
int _lseek(int file, int ptr, int dir)
{
  (void)file;
  (void)ptr;
  (void)dir;
  return 0;
}
int _open(char *path, int flags, ...)
{
  (void)path;
  (void)flags;
  /* Pretend like we always fail */
  return -1;
}
int _wait(int *status)
{
  (void)status;
  errno = ECHILD;
  return -1;
}
int _unlink(char *name)
```

```
{
  (void)name;
  errno = ENOENT;
  return -1;
}
int _times(struct tms *buf)
{
  (void)buf;
  return -1;
}
int _stat(char *file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
int _link(char *old, char *new)
{
  (void)old;
  (void)new;
  errno = EMLINK;
  return -1;
}
```

```
int _fork(void)
{
  errno = EAGAIN;
  return -1;
}
int _execve(char *name, char **argv, char **env)
{
  (void)name;
  (void)argv;
  (void)env;
  errno = ENOMEM;
  return -1;
}
```

25.13 CODE OF SYSMEM.C:

```
/**
**********************************************************
*******************
 * @file      sysmem.c
 * @author    Generated by STM32CubeIDE
 * @brief     STM32CubeIDE System Memorycalls file
 *
 *            For more information about which C functions
 *            need which of these lowlevel functions
```

```
 *          please consult the newlib libc manual
**********************************************************
*******************
 * @attention
 *
 * Copyright (c) 2024 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the
LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
**********************************************************
*******************
 */
/* Includes */
#include <errno.h>
#include <stdint.h>
/**
 * Pointer to the current high watermark of the heap usage
 */
static uint8_t *__sbrk_heap_end = NULL;
/**
```

```
 * @brief _sbrk() allocates memory to the newlib heap and is used by malloc
 *        and others from the C library
 *
 * @verbatim
 *
 ##########################################################################
 * #  .data  #  .bss  #       newlib heap       #          MSP stack          #
 * #         #        #                         # Reserved by _Min_Stack_Size #
 *
 ##########################################################################
 * ^-- RAM start      ^-- _end                          _estack, RAM end --^
 * @endverbatim
 *
 * This implementation starts allocating at the '_end' linker symbol
 * The '_Min_Stack_Size' linker symbol reserves a memory for the MSP stack
 * The implementation considers '_estack' linker symbol to be RAM end
 * NOTE: If the MSP stack, at any point during execution, grows larger than the
 * reserved size, please increase the '_Min_Stack_Size'.
 *
 * @param incr Memory size
```

```
 * @return Pointer to allocated memory
 */
void *_sbrk(ptrdiff_t incr)
{
  extern uint8_t _end; /* Symbol defined in the linker script */
  extern uint8_t _estack; /* Symbol defined in the linker script */
  extern uint32_t _Min_Stack_Size; /* Symbol defined in the linker
script */
  const uint32_t stack_limit = (uint32_t)&_estack -
(uint32_t)&_Min_Stack_Size;
  const uint8_t *max_heap = (uint8_t *)stack_limit;
  uint8_t *prev_heap_end;
  /* Initialize heap end at first call */
  if (NULL == __sbrk_heap_end)
  {
    __sbrk_heap_end = &_end;
  }
  /* Protect heap from growing into the reserved MSP stack */
  if (__sbrk_heap_end + incr > max_heap)
  {
    errno = ENOMEM;
    return (void *)-1;
  }
  prev_heap_end = __sbrk_heap_end;
```

```
  __sbrk_heap_end += incr;
  return (void *)prev_heap_end;
}
```

25.14 CODE OF SYSTEM_STM32F4XX.C:

```
/**
**********************************************************
*******************
  * @file    system_stm32f4xx.c
  * @author  MCD Application Team
  * @brief   CMSIS Cortex-M4 Device Peripheral Access Layer System
Source File.
  *
  *   This file provides two functions and one global variable to be called
from
  *   user application:
  *      - SystemInit(): This function is called at startup just after reset and
  *                      before branch to main program. This call is made inside
  *                      the "startup_stm32f4xx.s" file.
  *
  *      - SystemCoreClock variable: Contains the core clock (HCLK), it
can be used
  *                              by the user application to setup the SysTick
  *                              timer or configure other parameters.
  *
```

```
  *      - SystemCoreClockUpdate(): Updates the variable
SystemCoreClock and must
  *                          be called whenever the core clock is changed
  *                          during program execution.
  *
  *
**************************************************************
********************
  * @attention
  *
  * Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**************************************************************
********************
  */
/** @addtogroup CMSIS
  * @{
  */
```

```
/** @addtogroup stm32f4xx_system
  * @{
  */
/** @addtogroup STM32F4xx_System_Private_Includes
  * @{
  */
#include "stm32f4xx.h"
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    ((uint32_t)25000000) /*!< Default value of
the External oscillator in Hz */
#endif /* HSE_VALUE */
#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000) /*!< Value of the
Internal oscillator in Hz*/
#endif /* HSI_VALUE */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_TypesDefinitions
  * @{
  */
/**
  * @}
  */
```

```
/** @addtogroup STM32F4xx_System_Private_Defines
  * @{
  */
/************************* Miscellaneous Configuration
************************/
/*!< Uncomment the following line if you need to use external SRAM or
SDRAM as data memory  */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx) || defined(STM32F417xx)\
 || defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx) ||
defined(STM32F412Zx) || defined(STM32F412Vx)
/* #define DATA_IN_ExtSRAM */
#endif /* STM32F40xxx || STM32F41xxx || STM32F42xxx ||
STM32F43xxx || STM32F469xx || STM32F479xx ||\
        STM32F412Zx || STM32F412Vx */
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F446xx) || defined(STM32F469xx) ||
defined(STM32F479xx)
/* #define DATA_IN_ExtSDRAM */
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F446xx || STM32F469xx ||\
        STM32F479xx */
```

/* Note: Following vector table addresses must be defined in line with linker

configuration. */

/*!< Uncomment the following line if you need to relocate the vector table

anywhere in Flash or Sram, else the vector table is kept at the automatic

remap of boot address selected */

/* #define USER_VECT_TAB_ADDRESS */

#if defined(USER_VECT_TAB_ADDRESS)

/*!< Uncomment the following line if you need to relocate your vector Table

in Sram else user remap will be done in Flash. */

/* #define VECT_TAB_SRAM */

#if defined(VECT_TAB_SRAM)

#define VECT_TAB_BASE_ADDRESS SRAM_BASE /*!< Vector Table base address field.

This value must be a multiple of 0x200. */

#define VECT_TAB_OFFSET 0x00000000U /*!< Vector Table base offset field.

This value must be a multiple of 0x200. */

#else

#define VECT_TAB_BASE_ADDRESS FLASH_BASE /*!< Vector Table base address field.

This value must be a multiple of 0x200.
*/

#define VECT_TAB_OFFSET 0x00000000U /*!< Vector Table base offset field.

This value must be a multiple of 0x200.
*/

#endif /* VECT_TAB_SRAM */

#endif /* USER_VECT_TAB_ADDRESS */

/**********/

/**

* @}

*/

/** @addtogroup STM32F4xx_System_Private_Macros

* @{

*/

/**

* @}

*/

/** @addtogroup STM32F4xx_System_Private_Variables

* @{

*/

/* This variable is updated in three ways:

1) by calling CMSIS function SystemCoreClockUpdate()

```
     2) by calling HAL API function HAL_RCC_GetHCLKFreq()
     3) each time HAL_RCC_ClockConfig() is called to configure the system clock frequency
        Note: If you use this function to configure the system clock; then there
              is no need to call the 2 first functions listed above, since SystemCoreClock
              variable is updated automatically.
  */
uint32_t SystemCoreClock = 16000000;
const uint8_t AHBPrescTable[16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9};
const uint8_t APBPrescTable[8]  = {0, 0, 0, 0, 1, 2, 3, 4};
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_FunctionPrototypes
  * @{
  */
#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  static void SystemInit_ExtMemCtl(void);
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */
/**
  * @}
```

```
  */
/** @addtogroup STM32F4xx_System_Private_Functions
  * @{
  */
/**
  * @brief  Setup the microcontroller system
  *         Initialize the FPU setting, vector table location and External memory
  *         configuration.
  * @param  None
  * @retval None
  */
void SystemInit(void)
{
  /* FPU settings ------------------------------------------------------------*/
  #if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
    SCB->CPACR |= ((3UL << 10*2)|(3UL << 11*2));  /* set CP10 and CP11 Full Access */
  #endif
#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  SystemInit_ExtMemCtl();
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */
  /* Configure the Vector Table location -------------------------------------*/
#if defined(USER_VECT_TAB_ADDRESS)
```

```
  SCB->VTOR = VECT_TAB_BASE_ADDRESS |
VECT_TAB_OFFSET; /* Vector Table Relocation in Internal SRAM */
#endif /* USER_VECT_TAB_ADDRESS */
}
/**
   * @brief  Update SystemCoreClock variable according to Clock
Register Values.
  *         The SystemCoreClock variable contains the core clock (HCLK),
it can
  *         be used by the user application to setup the SysTick timer or
configure
  *         other parameters.
  *
  * @note   Each time the core clock (HCLK) changes, this function
must be called
  *         to update SystemCoreClock variable value. Otherwise, any
configuration
  *         based on this variable will be incorrect.
  *
  * @note   - The system frequency computed by this function is not the
real
  *           frequency in the chip. It is calculated based on the predefined
  *           constant and the selected clock source:
  *
```

```
  *          - If SYSCLK source is HSI, SystemCoreClock will contain the
HSI_VALUE(*)
  *
  *          - If SYSCLK source is HSE, SystemCoreClock will contain the
HSE_VALUE(**)
  *
  *          - If SYSCLK source is PLL, SystemCoreClock will contain the
HSE_VALUE(**)
  *            or HSI_VALUE(*) multiplied/divided by the PLL factors.
  *
  *        (*) HSI_VALUE is a constant defined in stm32f4xx_hal_conf.h
file (default value
  *            16 MHz) but the real value may vary depending on the
variations
  *            in voltage and temperature.
  *
  *        (**) HSE_VALUE is a constant defined in
stm32f4xx_hal_conf.h file (its value
  *             depends on the application requirements), user has to ensure
that HSE_VALUE
  *             is same as the real frequency of the crystal used. Otherwise,
this function
  *             may have wrong result.
  *
  *        - The result of this function could be not correct when using
fractional
```

```
  *          value for HSE crystal.
  *
  * @param  None
  * @retval None
  */
void SystemCoreClockUpdate(void)
{
  uint32_t tmp = 0, pllvco = 0, pllp = 2, pllsource = 0, pllm = 2;
  /* Get SYSCLK source -------------------------------------------------------
*/
  tmp = RCC->CFGR & RCC_CFGR_SWS;
  switch (tmp)
  {
    case 0x00:  /* HSI used as system clock source */
      SystemCoreClock = HSI_VALUE;
      break;
    case 0x04:  /* HSE used as system clock source */
      SystemCoreClock = HSE_VALUE;
      break;
    case 0x08:  /* PLL used as system clock source */
      /* PLL_VCO = (HSE_VALUE or HSI_VALUE / PLL_M) * PLL_N
         SYSCLK = PLL_VCO / PLL_P
         */
```

```
    pllsource = (RCC->PLLCFGR & RCC_PLLCFGR_PLLSRC) >>
22;
    pllm = RCC->PLLCFGR & RCC_PLLCFGR_PLLM;
    if (pllsource != 0)
    {
      /* HSE used as PLL clock source */
      pllvco = (HSE_VALUE / pllm) * ((RCC->PLLCFGR &
RCC_PLLCFGR_PLLN) >> 6);
    }
    else
    {
      /* HSI used as PLL clock source */
      pllvco = (HSI_VALUE / pllm) * ((RCC->PLLCFGR &
RCC_PLLCFGR_PLLN) >> 6);
    }
    pllp = (((RCC->PLLCFGR & RCC_PLLCFGR_PLLP) >>16) + 1 )
*2;
    SystemCoreClock = pllvco/pllp;
    break;
   default:
    SystemCoreClock = HSI_VALUE;
    break;
  }
  /* Compute HCLK frequency ----------------------------------------------
-*/
```

```
  /* Get HCLK prescaler */
  tmp = AHBPrescTable[((RCC->CFGR & RCC_CFGR_HPRE) >> 4)];
  /* HCLK frequency */
  SystemCoreClock >>= tmp;
}
#if defined (DATA_IN_ExtSRAM) && defined
(DATA_IN_ExtSDRAM)
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
/**
  * @brief  Setup the external memory controller.
  *         Called in startup_stm32f4xx.s before jump to main.
  *         This function configures the external memories
(SRAM/SDRAM)
  *         This SRAM/SDRAM will be used as program data memory
(including heap and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
```

```
  register __IO uint32_t index;
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and
GPIOI interface clock */
  RCC->AHB1ENR |= 0x000001F8;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x00CCC0CC;
  GPIOD->AFR[1]  = 0xCCCCCCCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xAAAA0A8A;
  /* Configure PDx pins speed to 100 MHz */
  GPIOD->OSPEEDR = 0xFFFF0FCF;
  /* Configure PDx pins Output type to push-pull */
  GPIOD->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOD->PUPDR   = 0x00000000;
  /* Connect PEx pins to FMC Alternate function */
  GPIOE->AFR[0]  = 0xC00CC0CC;
  GPIOE->AFR[1]  = 0xCCCCCCCC;
  /* Configure PEx pins in Alternate function mode */
  GPIOE->MODER   = 0xAAAA828A;
  /* Configure PEx pins speed to 100 MHz */
```

```
GPIOE->OSPEEDR = 0xFFFFC3CF;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0xCCCCCCCC;
GPIOF->AFR[1]  = 0xCCCCCCCC;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA800AAA;
/* Configure PFx pins speed to 50 MHz */
GPIOF->OSPEEDR = 0xAA800AAA;
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
GPIOF->PUPDR   = 0x00000000;
/* Connect PGx pins to FMC Alternate function */
GPIOG->AFR[0]  = 0xCCCCCCCC;
GPIOG->AFR[1]  = 0xCCCCCCCC;
/* Configure PGx pins in Alternate function mode */
GPIOG->MODER   = 0xAAAAAAAA;
/* Configure PGx pins speed to 50 MHz */
GPIOG->OSPEEDR = 0xAAAAAAAA;
```

```
/* Configure PGx pins Output type to push-pull */
GPIOG->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PGx pins */
GPIOG->PUPDR   = 0x00000000;
/* Connect PHx pins to FMC Alternate function */
GPIOH->AFR[0]  = 0x00C0CC00;
GPIOH->AFR[1]  = 0xCCCCCCCC;
/* Configure PHx pins in Alternate function mode */
GPIOH->MODER   = 0xAAAA08A0;
/* Configure PHx pins speed to 50 MHz */
GPIOH->OSPEEDR = 0xAAAA08A0;
/* Configure PHx pins Output type to push-pull */
GPIOH->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PHx pins */
GPIOH->PUPDR   = 0x00000000;
/* Connect PIx pins to FMC Alternate function */
GPIOI->AFR[0]  = 0xCCCCCCCC;
GPIOI->AFR[1]  = 0x00000CC0;
/* Configure PIx pins in Alternate function mode */
GPIOI->MODER   = 0x0028AAAA;
/* Configure PIx pins speed to 50 MHz */
GPIOI->OSPEEDR = 0x0028AAAA;
/* Configure PIx pins Output type to push-pull */
```

```
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;
/*-- FMC Configuration ------------------------------------------------------*/
  /* Enable the FMC interface clock */
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
  FMC_Bank5_6->SDTR[0] = 0x01115351;
  /* SDRAM initialization sequence */
  /* Clock enable command */
  FMC_Bank5_6->SDCMR = 0x00000011;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Delay */
  for (index = 0; index<1000; index++);
  /* PALL command */
  FMC_Bank5_6->SDCMR = 0x00000012;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
```

```
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* Auto refresh command */
FMC_Bank5_6->SDCMR = 0x00000073;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* MRD register program */
FMC_Bank5_6->SDCMR = 0x00046014;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* Set refresh count */
tmpreg = FMC_Bank5_6->SDRTR;
```

```
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
  /* Disable write protection */
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F469xx || STM32F479xx */
  (void)(tmp);
}
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F469xx || STM32F479xx */
#elif defined (DATA_IN_ExtSRAM) || defined
(DATA_IN_ExtSDRAM)
```

```
/**
  * @brief  Setup the external memory controller.
  *         Called in startup_stm32f4xx.s before jump to main.
  *         This function configures the external memories
(SRAM/SDRAM)
  *         This SRAM/SDRAM will be used as program data memory
(including heap and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F446xx) || defined(STM32F469xx) ||
defined(STM32F479xx)
#if defined (DATA_IN_ExtSDRAM)
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
  register __IO uint32_t index;
#if defined(STM32F446xx)
  /* Enable GPIOA, GPIOC, GPIOD, GPIOE, GPIOF, GPIOG interface
     clock */
  RCC->AHB1ENR |= 0x0000007D;
```

```
#else
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and
GPIOI interface
     clock */
  RCC->AHB1ENR |= 0x000001F8;
#endif /* STM32F446xx */
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);
#if defined(STM32F446xx)
  /* Connect PAx pins to FMC Alternate function */
  GPIOA->AFR[0]  |= 0xC0000000;
  GPIOA->AFR[1]  |= 0x00000000;
  /* Configure PDx pins in Alternate function mode */
  GPIOA->MODER   |= 0x00008000;
  /* Configure PDx pins speed to 50 MHz */
  GPIOA->OSPEEDR |= 0x00008000;
  /* Configure PDx pins Output type to push-pull */
  GPIOA->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOA->PUPDR   |= 0x00000000;
  /* Connect PCx pins to FMC Alternate function */
  GPIOC->AFR[0]  |= 0x00CC0000;
  GPIOC->AFR[1]  |= 0x00000000;
```

```
  /* Configure PDx pins in Alternate function mode */
  GPIOC->MODER   |= 0x00000A00;
  /* Configure PDx pins speed to 50 MHz */
  GPIOC->OSPEEDR |= 0x00000A00;
  /* Configure PDx pins Output type to push-pull */
  GPIOC->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOC->PUPDR   |= 0x00000000;
#endif /* STM32F446xx */
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x000000CC;
  GPIOD->AFR[1]  = 0xCC000CCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xA02A000A;
  /* Configure PDx pins speed to 50 MHz */
  GPIOD->OSPEEDR = 0xA02A000A;
  /* Configure PDx pins Output type to push-pull */
  GPIOD->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOD->PUPDR   = 0x00000000;
  /* Connect PEx pins to FMC Alternate function */
  GPIOE->AFR[0]  = 0xC00000CC;
  GPIOE->AFR[1]  = 0xCCCCCCCC;
```

```
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA800A;
/* Configure PEx pins speed to 50 MHz */
GPIOE->OSPEEDR = 0xAAAA800A;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0xCCCCCCCC;
GPIOF->AFR[1]  = 0xCCCCCCCC;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA800AAA;
/* Configure PFx pins speed to 50 MHz */
GPIOF->OSPEEDR = 0xAA800AAA;
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
GPIOF->PUPDR   = 0x00000000;
/* Connect PGx pins to FMC Alternate function */
GPIOG->AFR[0]  = 0xCCCCCCCC;
GPIOG->AFR[1]  = 0xCCCCCCCC;
/* Configure PGx pins in Alternate function mode */
```

```
  GPIOG->MODER   = 0xAAAAAAAA;
  /* Configure PGx pins speed to 50 MHz */
  GPIOG->OSPEEDR = 0xAAAAAAAA;
  /* Configure PGx pins Output type to push-pull */
  GPIOG->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
  /* Connect PHx pins to FMC Alternate function */
  GPIOH->AFR[0]  = 0x00C0CC00;
  GPIOH->AFR[1]  = 0xCCCCCCCC;
  /* Configure PHx pins in Alternate function mode */
  GPIOH->MODER   = 0xAAAA08A0;
  /* Configure PHx pins speed to 50 MHz */
  GPIOH->OSPEEDR = 0xAAAA08A0;
  /* Configure PHx pins Output type to push-pull */
  GPIOH->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PHx pins */
  GPIOH->PUPDR   = 0x00000000;
  /* Connect PIx pins to FMC Alternate function */
  GPIOI->AFR[0]  = 0xCCCCCCCC;
```

```
  GPIOI->AFR[1]  = 0x00000CC0;
  /* Configure PIx pins in Alternate function mode */
  GPIOI->MODER   = 0x0028AAAA;
  /* Configure PIx pins speed to 50 MHz */
  GPIOI->OSPEEDR = 0x0028AAAA;
  /* Configure PIx pins Output type to push-pull */
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F469xx || STM32F479xx */
/*-- FMC Configuration ------------------------------------------------------*/
  /* Enable the FMC interface clock */
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable SDRAM bank1 */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCR[0] = 0x00001954;
#else
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
#endif /* STM32F446xx */
  FMC_Bank5_6->SDTR[0] = 0x01115351;
```

```
  /* SDRAM initialization sequence */
  /* Clock enable command */
  FMC_Bank5_6->SDCMR = 0x00000011;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Delay */
  for (index = 0; index<1000; index++);
  /* PALL command */
  FMC_Bank5_6->SDCMR = 0x00000012;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Auto refresh command */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x000000F3;
#else
  FMC_Bank5_6->SDCMR = 0x00000073;
```

```
#endif /* STM32F446xx */
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* MRD register program */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x00044014;
#else
  FMC_Bank5_6->SDCMR = 0x00046014;
#endif /* STM32F446xx */
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Set refresh count */
  tmpreg = FMC_Bank5_6->SDRTR;
#if defined(STM32F446xx)
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000050C<<1));
```

```
#else
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
#endif /* STM32F446xx */
  /* Disable write protection */
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#endif /* DATA_IN_ExtSDRAM */
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F446xx || STM32F469xx || STM32F479xx */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx) || defined(STM32F417xx)\
 || defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx) ||
defined(STM32F412Zx) || defined(STM32F412Vx)
#if defined(DATA_IN_ExtSRAM)
/*-- GPIOs Configuration -----------------------------------------------------*/
   /* Enable GPIOD, GPIOE, GPIOF and GPIOG interface clock */
  RCC->AHB1ENR   |= 0x00000078;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIODEN);
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x00CCC0CC;
  GPIOD->AFR[1]  = 0xCCCCCCCC;
```

```
/* Configure PDx pins in Alternate function mode */
GPIOD->MODER   = 0xAAAA0A8A;
/* Configure PDx pins speed to 100 MHz */
GPIOD->OSPEEDR = 0xFFFF0FCF;
/* Configure PDx pins Output type to push-pull */
GPIOD->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PDx pins */
GPIOD->PUPDR   = 0x00000000;
/* Connect PEx pins to FMC Alternate function */
GPIOE->AFR[0]  = 0xC00CC0CC;
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA828A;
/* Configure PEx pins speed to 100 MHz */
GPIOE->OSPEEDR = 0xFFFFC3CF;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0x00CCCCCC;
GPIOF->AFR[1]  = 0xCCCC0000;
/* Configure PFx pins in Alternate function mode */
```

```
  GPIOF->MODER   = 0xAA000AAA;
  /* Configure PFx pins speed to 100 MHz */
  GPIOF->OSPEEDR = 0xFF000FFF;
  /* Configure PFx pins Output type to push-pull */
  GPIOF->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PFx pins */
  GPIOF->PUPDR   = 0x00000000;
  /* Connect PGx pins to FMC Alternate function */
  GPIOG->AFR[0]  = 0x00CCCCCC;
  GPIOG->AFR[1]  = 0x000000C0;
  /* Configure PGx pins in Alternate function mode */
  GPIOG->MODER   = 0x00085AAA;
  /* Configure PGx pins speed to 100 MHz */
  GPIOG->OSPEEDR = 0x000CAFFF;
  /* Configure PGx pins Output type to push-pull */
  GPIOG->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;
/*-- FMC/FSMC Configuration ----------------------------------------------
-*/
  /* Enable the FMC/FSMC interface clock */
  RCC->AHB3ENR         |= 0x00000001;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)
```

```
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F469xx || STM32F479xx */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx)|| defined(STM32F417xx)\
   || defined(STM32F412Zx) || defined(STM32F412Vx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FSMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FSMC_Bank1->BTCR[2]  = 0x00001011;
  FSMC_Bank1->BTCR[3]  = 0x00000201;
```

```
  FSMC_Bank1E->BWTR[2] = 0x0FFFFFFF;
#endif /* STM32F405xx || STM32F415xx || STM32F407xx ||
STM32F417xx || STM32F412Zx || STM32F412Vx */
#endif /* DATA_IN_ExtSRAM */
#endif /* STM32F405xx || STM32F415xx || STM32F407xx ||
STM32F417xx || STM32F427xx || STM32F437xx ||\
        STM32F429xx || STM32F439xx || STM32F469xx ||
STM32F479xx || STM32F412Zx || STM32F412Vx  */
  (void)(tmp);
}
#endif /* DATA_IN_ExtSRAM && DATA_IN_ExtSDRAM */
/**
  * @}
  */
/**
  * @}
  */
/**
  * @}
  */
```

CHAPTER#26 CODING A HARDWARE TIMER DRIVER FOR COUNTING EVENTS

26.1CODING A HARDWARE TIMER DRIVER FOR COUNTING EVENTS:

Coding a hardware timer driver for counting events involves setting up a timer peripheral on a microcontroller to keep track of occurrences of specific events. This can be useful for measuring time intervals, counting pulses, or generating periodic interrupts. Below is a general guide on how to implement a hardware timer driver in assembly language for an ARM Cortex-M based microcontroller, such as those in the STM32 family.

26.2 UNDERSTANDING THE TIME PERIPHERAL:

Microcontrollers typically have multiple timer peripherals, which can operate in various modes, such as:

26.2.1TIME BASE MODE:

Generates periodic interrupts.

26.2.2 INPUT CAPTURE MODE:

Measures the time between external events.

26.2.3 OUTPUT COMPARE MODE:

Generates events at specific times.

26.2.4 PULSE WIDTH MODULATION (PWM) MODE:

Generates PWM signals.

26.3 SETTING UP TIMER:

26.3.1 ENABLING THE TIMER CLOCK:

Enable the clock for the timer peripheral in the RCC (Reset and Clock Control) register.

26.3.2 CONFIGURING THE TIMER:

Configure the timer’s prescaler and auto-reload values to set the counting rate and period.

26.3.3 STARTING THE TIMER:

Enable the timer by setting the appropriate bits in the control register.

26.4 COUNTING EXTERNAL EVENTS:

To count external events, such as pulses, configure the timer's input capture mode or use the external clock source.

26.4.1USING INPUT CAPTURE MODE:

Set up the timer to capture on a specific channel when an event occurs on the associated GPIO pin.

26.4.2 USING EXTERNAL CLOCK MODE:

Alternatively, we can set the timer to use an external clock signal as its counting source.

26.5 READING THE COUNTER VALUE:

To read the counter value, access the CNT (Counter) register. This can be done in your main loop or periodically using interrupts.

26.6 HANDLING TIMER INTERRUPTS:

If the application requires reacting to timer events, such as overflow or specific capture events, enable and handle timer interrupts.

26.7 OPTIMIZATION AND CONSIDERATIONS:

26.7.1 PRESCALER SETTINGS:

Adjust the prescaler value to match the event rate with the timer's counting capability.

26.7.2 COUNTER SIZE:

Use appropriate counter size (16-bit or 32-bit) based on your application's requirements.

26.7.3 INTERRUPT HANDLING:

Ensure interrupt handlers are efficient to avoid missing events.

26.8 CODE OF MAIN.H:

```
/* USER CODE BEGIN Header */
/**
*******************************************************
********************
  * @file           : main.h
  * @brief          : Header for main.c file.
  *                   This file contains the common defines of the application.
*******************************************************
********************
  * @attention
```

```
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*******************************************************************
*******************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H
#ifdef __cplusplus
extern "C" {
#endif
/* Includes ------------------------------------------------------------------*/
#include "stm32f4xx_hal.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
```

```
/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */
/* USER CODE END EC */
/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */
/* USER CODE END EM */
/* Exported functions prototypes ---------------------------------------------*/
void Error_Handler(void);
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
/* Private defines -----------------------------------------------------------*/
#define B1_Pin GPIO_PIN_13
#define B1_GPIO_Port GPIOC
#define USART_TX_Pin GPIO_PIN_2
#define USART_TX_GPIO_Port GPIOA
#define USART_RX_Pin GPIO_PIN_3
#define USART_RX_GPIO_Port GPIOA
#define LD2_Pin GPIO_PIN_5
#define LD2_GPIO_Port GPIOA
#define TMS_Pin GPIO_PIN_13
```

```
#define TMS_GPIO_Port GPIOA
#define TCK_Pin GPIO_PIN_14
#define TCK_GPIO_Port GPIOA
#define SWO_Pin GPIO_PIN_3
#define SWO_GPIO_Port GPIOB
/* USER CODE BEGIN Private defines */
/* USER CODE END Private defines */
#ifdef __cplusplus
}
#endif
#endif /* __MAIN_H */
```

26.9 CODE OF STM32F4XX_HAL_CONF.H:

```
/* USER CODE BEGIN Header */
/**
*****************************************************************
*******************
  * @file    stm32f4xx_hal_conf_template.h
  * @author  MCD Application Team
  * @brief   HAL configuration template file.
  *          This file should be copied to the application folder and renamed
  *          to stm32f4xx_hal_conf.h.

*****************************************************************
*******************
```

```
  * @attention
  *
  * Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*****************************************************************
*******************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_HAL_CONF_H
#define __STM32F4xx_HAL_CONF_H
#ifdef __cplusplus
 extern "C" {
#endif
/* Exported types ------------------------------------------------------------*/
/* Exported constants --------------------------------------------------------*/
/* ############################ Module Selection
################################## */
```

```
/**
  * @brief This is the list of modules to be used in the HAL driver
  */
#define HAL_MODULE_ENABLED
  /* #define HAL_CRYP_MODULE_ENABLED */
/* #define HAL_ADC_MODULE_ENABLED */
/* #define HAL_CAN_MODULE_ENABLED */
/* #define HAL_CRC_MODULE_ENABLED */
/* #define HAL_CAN_LEGACY_MODULE_ENABLED */
/* #define HAL_DAC_MODULE_ENABLED */
/* #define HAL_DCMI_MODULE_ENABLED */
/* #define HAL_DMA2D_MODULE_ENABLED */
/* #define HAL_ETH_MODULE_ENABLED */
/* #define HAL_ETH_LEGACY_MODULE_ENABLED */
/* #define HAL_NAND_MODULE_ENABLED */
/* #define HAL_NOR_MODULE_ENABLED */
/* #define HAL_PCCARD_MODULE_ENABLED */
/* #define HAL_SRAM_MODULE_ENABLED */
/* #define HAL_SDRAM_MODULE_ENABLED */
/* #define HAL_HASH_MODULE_ENABLED */
/* #define HAL_I2C_MODULE_ENABLED */
/* #define HAL_I2S_MODULE_ENABLED */
/* #define HAL_IWDG_MODULE_ENABLED */
```

```
/* #define HAL_LTDC_MODULE_ENABLED */
/* #define HAL_RNG_MODULE_ENABLED */
/* #define HAL_RTC_MODULE_ENABLED */
/* #define HAL_SAI_MODULE_ENABLED */
/* #define HAL_SD_MODULE_ENABLED */
/* #define HAL_MMC_MODULE_ENABLED */
/* #define HAL_SPI_MODULE_ENABLED */
/* #define HAL_TIM_MODULE_ENABLED */
#define HAL_UART_MODULE_ENABLED
/* #define HAL_USART_MODULE_ENABLED */
/* #define HAL_IRDA_MODULE_ENABLED */
/* #define HAL_SMARTCARD_MODULE_ENABLED */
/* #define HAL_SMBUS_MODULE_ENABLED */
/* #define HAL_WWDG_MODULE_ENABLED */
/* #define HAL_PCD_MODULE_ENABLED */
/* #define HAL_HCD_MODULE_ENABLED */
/* #define HAL_DSI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_CEC_MODULE_ENABLED */
/* #define HAL_FMPI2C_MODULE_ENABLED */
/* #define HAL_FMPSMBUS_MODULE_ENABLED */
/* #define HAL_SPDIFRX_MODULE_ENABLED */
```

```
/* #define HAL_DFSDM_MODULE_ENABLED */
/* #define HAL_LPTIM_MODULE_ENABLED */
#define HAL_GPIO_MODULE_ENABLED
#define HAL_EXTI_MODULE_ENABLED
#define HAL_DMA_MODULE_ENABLED
#define HAL_RCC_MODULE_ENABLED
#define HAL_FLASH_MODULE_ENABLED
#define HAL_PWR_MODULE_ENABLED
#define HAL_CORTEX_MODULE_ENABLED
/* ############################ HSE/HSI Values adaptation
###################### */
/**
  * @brief Adjust the value of External High Speed oscillator (HSE)
used in your application.
  *        This value is used by the RCC HAL module to compute the
system frequency
  *        (when HSE is used as system clock source, directly or through
the PLL).
  */
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    8000000U /*!< Value of the External
oscillator in Hz */
#endif /* HSE_VALUE */
#if !defined  (HSE_STARTUP_TIMEOUT)
```

```
  #define HSE_STARTUP_TIMEOUT    100U   /*!< Time out for
HSE start up, in ms */
#endif /* HSE_STARTUP_TIMEOUT */
/**
  * @brief Internal High Speed oscillator (HSI) value.
  *        This value is used by the RCC HAL module to compute the
system frequency
  *        (when HSI is used as system clock source, directly or through the
PLL).
  */
#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000U) /*!< Value of the
Internal oscillator in Hz*/
#endif /* HSI_VALUE */
/**
  * @brief Internal Low Speed oscillator (LSI) value.
  */
#if !defined  (LSI_VALUE)
 #define LSI_VALUE  32000U       /*!< LSI Typical Value in Hz*/
#endif /* LSI_VALUE */                    /*!< Value of the Internal Low
Speed oscillator in Hz
                                  The real value may vary depending on the
variations
                                  in voltage and temperature.*/
/**
```

```
  * @brief External Low Speed oscillator (LSE) value.
  */
#if !defined  (LSE_VALUE)
 #define LSE_VALUE  32768U    /*!< Value of the External Low Speed
oscillator in Hz */
#endif /* LSE_VALUE */
#if !defined  (LSE_STARTUP_TIMEOUT)
  #define LSE_STARTUP_TIMEOUT    5000U   /*!< Time out for
LSE start up, in ms */
#endif /* LSE_STARTUP_TIMEOUT */
/**
  * @brief External clock source for I2S peripheral
  *       This value is used by the I2S HAL module to compute the I2S
clock source
  *       frequency, this source is inserted directly through I2S_CKIN
pad.
  */
#if !defined  (EXTERNAL_CLOCK_VALUE)
  #define EXTERNAL_CLOCK_VALUE    12288000U /*!< Value of
the External audio frequency in Hz*/
#endif /* EXTERNAL_CLOCK_VALUE */
/* Tip: To avoid modifying this file each time you need to use different
HSE,
   ===  you can define the HSE value in your toolchain compiler
preprocessor. */
```

```
/* ############################## System Configuration
########################### */

/**

  * @brief This is the HAL system configuration section

  */

#define  VDD_VALUE                    3300U /*!< Value of VDD in mv */

#define  TICK_INT_PRIORITY            0U   /*!< tick interrupt
priority */

#define  USE_RTOS                     0U

#define  PREFETCH_ENABLE              1U

#define  INSTRUCTION_CACHE_ENABLE     1U

#define  DATA_CACHE_ENABLE            1U

#define  USE_HAL_ADC_REGISTER_CALLBACKS        0U /*
ADC register callback disabled       */

#define  USE_HAL_CAN_REGISTER_CALLBACKS        0U /*
CAN register callback disabled       */

#define  USE_HAL_CEC_REGISTER_CALLBACKS        0U /*
CEC register callback disabled       */

#define  USE_HAL_CRYP_REGISTER_CALLBACKS       0U /*
CRYP register callback disabled      */

#define  USE_HAL_DAC_REGISTER_CALLBACKS        0U /*
DAC register callback disabled       */

#define  USE_HAL_DCMI_REGISTER_CALLBACKS       0U /*
DCMI register callback disabled      */

#define  USE_HAL_DFSDM_REGISTER_CALLBACKS      0U /*
DFSDM register callback disabled     */
```

#define USE_HAL_DMA2D_REGISTER_CALLBACKS 0U /* DMA2D register callback disabled */

#define USE_HAL_DSI_REGISTER_CALLBACKS 0U /* DSI register callback disabled */

#define USE_HAL_ETH_REGISTER_CALLBACKS 0U /* ETH register callback disabled */

#define USE_HAL_HASH_REGISTER_CALLBACKS 0U /* HASH register callback disabled */

#define USE_HAL_HCD_REGISTER_CALLBACKS 0U /* HCD register callback disabled */

#define USE_HAL_I2C_REGISTER_CALLBACKS 0U /* I2C register callback disabled */

#define USE_HAL_FMPI2C_REGISTER_CALLBACKS 0U /* FMPI2C register callback disabled */

#define USE_HAL_FMPSMBUS_REGISTER_CALLBACKS 0U /* FMPSMBUS register callback disabled */

#define USE_HAL_I2S_REGISTER_CALLBACKS 0U /* I2S register callback disabled */

#define USE_HAL_IRDA_REGISTER_CALLBACKS 0U /* IRDA register callback disabled */

#define USE_HAL_LPTIM_REGISTER_CALLBACKS 0U /* LPTIM register callback disabled */

#define USE_HAL_LTDC_REGISTER_CALLBACKS 0U /* LTDC register callback disabled */

#define USE_HAL_MMC_REGISTER_CALLBACKS 0U /* MMC register callback disabled */

#define USE_HAL_NAND_REGISTER_CALLBACKS 0U /* NAND register callback disabled */

#define USE_HAL_NOR_REGISTER_CALLBACKS 0U /* NOR register callback disabled */

#define USE_HAL_PCCARD_REGISTER_CALLBACKS 0U /* PCCARD register callback disabled */

#define USE_HAL_PCD_REGISTER_CALLBACKS 0U /* PCD register callback disabled */

#define USE_HAL_QSPI_REGISTER_CALLBACKS 0U /* QSPI register callback disabled */

#define USE_HAL_RNG_REGISTER_CALLBACKS 0U /* RNG register callback disabled */

#define USE_HAL_RTC_REGISTER_CALLBACKS 0U /* RTC register callback disabled */

#define USE_HAL_SAI_REGISTER_CALLBACKS 0U /* SAI register callback disabled */

#define USE_HAL_SD_REGISTER_CALLBACKS 0U /* SD register callback disabled */

#define USE_HAL_SMARTCARD_REGISTER_CALLBACKS 0U /* SMARTCARD register callback disabled */

#define USE_HAL_SDRAM_REGISTER_CALLBACKS 0U /* SDRAM register callback disabled */

#define USE_HAL_SRAM_REGISTER_CALLBACKS 0U /* SRAM register callback disabled */

#define USE_HAL_SPDIFRX_REGISTER_CALLBACKS 0U /* SPDIFRX register callback disabled */

#define USE_HAL_SMBUS_REGISTER_CALLBACKS 0U /* SMBUS register callback disabled */

#define USE_HAL_SPI_REGISTER_CALLBACKS 0U /* SPI register callback disabled */

#define USE_HAL_TIM_REGISTER_CALLBACKS 0U /* TIM register callback disabled */

#define USE_HAL_UART_REGISTER_CALLBACKS 0U /* UART register callback disabled */

#define USE_HAL_USART_REGISTER_CALLBACKS 0U /* USART register callback disabled */

#define USE_HAL_WWDG_REGISTER_CALLBACKS 0U /* WWDG register callback disabled */

/* ############################ Assert Selection ################################ */

/**

* @brief Uncomment the line below to expanse the "assert_param" macro in the

* HAL drivers code

*/

/* #define USE_FULL_ASSERT 1U */

/* ################### Ethernet peripheral configuration ###################### */

/* Section 1 : Ethernet peripheral configuration */

/* MAC ADDRESS: MAC_ADDR0:MAC_ADDR1:MAC_ADDR2:MAC_ADDR3:MAC_ADDR4:MAC_ADDR5 */

```
#define MAC_ADDR0   2U
#define MAC_ADDR1   0U
#define MAC_ADDR2   0U
#define MAC_ADDR3   0U
#define MAC_ADDR4   0U
#define MAC_ADDR5   0U
/* Definition of the Ethernet driver buffers size and count */
#define ETH_RX_BUF_SIZE          ETH_MAX_PACKET_SIZE
/* buffer size for receive          */
#define ETH_TX_BUF_SIZE          ETH_MAX_PACKET_SIZE
/* buffer size for transmit         */
#define ETH_RXBUFNB              4U      /* 4 Rx buffers of size
ETH_RX_BUF_SIZE  */
#define ETH_TXBUFNB              4U      /* 4 Tx buffers of size
ETH_TX_BUF_SIZE  */
/* Section 2: PHY configuration section */
/* DP83848_PHY_ADDRESS Address*/
#define DP83848_PHY_ADDRESS
/* PHY Reset delay these values are based on a 1 ms Systick interrupt*/
#define PHY_RESET_DELAY          0x000000FFU
/* PHY Configuration delay */
#define PHY_CONFIG_DELAY         0x00000FFFU
#define PHY_READ_TO              0x0000FFFFU
#define PHY_WRITE_TO             0x0000FFFFU
```

/* Section 3: Common PHY Registers */

#define PHY_BCR ((uint16_t)0x0000U) /*!< Transceiver Basic Control Register */

#define PHY_BSR ((uint16_t)0x0001U) /*!< Transceiver Basic Status Register */

#define PHY_RESET ((uint16_t)0x8000U) /*!< PHY Reset */

#define PHY_LOOPBACK ((uint16_t)0x4000U) /*!< Select loop-back mode */

#define PHY_FULLDUPLEX_100M ((uint16_t)0x2100U) /*!< Set the full-duplex mode at 100 Mb/s */

#define PHY_HALFDUPLEX_100M ((uint16_t)0x2000U) /*!< Set the half-duplex mode at 100 Mb/s */

#define PHY_FULLDUPLEX_10M ((uint16_t)0x0100U) /*!< Set the full-duplex mode at 10 Mb/s */

#define PHY_HALFDUPLEX_10M ((uint16_t)0x0000U) /*!< Set the half-duplex mode at 10 Mb/s */

#define PHY_AUTONEGOTIATION ((uint16_t)0x1000U) /*!< Enable auto-negotiation function */

#define PHY_RESTART_AUTONEGOTIATION ((uint16_t)0x0200U) /*!< Restart auto-negotiation function */

#define PHY_POWERDOWN ((uint16_t)0x0800U) /*!< Select the power down mode */

#define PHY_ISOLATE ((uint16_t)0x0400U) /*!< Isolate PHY from MII */

#define PHY_AUTONEGO_COMPLETE ((uint16_t)0x0020U) /*!< Auto-Negotiation process completed */

```
#define PHY_LINKED_STATUS            ((uint16_t)0x0004U)  /*!< Valid link established          */
#define PHY_JABBER_DETECTION         ((uint16_t)0x0002U)  /*!< Jabber condition detected       */
/* Section 4: Extended PHY Registers */
#define PHY_SR                       ((uint16_t))    /*!< PHY status register Offset          */
#define PHY_SPEED_STATUS             ((uint16_t))  /*!< PHY Speed mask                      */
#define PHY_DUPLEX_STATUS            ((uint16_t))  /*!< PHY Duplex mask                     */
/* ################### SPI peripheral configuration ############################ */
/* CRC FEATURE: Use to activate CRC feature inside HAL SPI Driver
* Activated: CRC code is present inside driver
* Deactivated: CRC code cleaned from driver
*/
#define USE_SPI_CRC                  0U
/* Includes ------------------------------------------------------------------*/
/**
  * @brief Include module's header file
  */
#ifdef HAL_RCC_MODULE_ENABLED
  #include "stm32f4xx_hal_rcc.h"
#endif /* HAL_RCC_MODULE_ENABLED */
```

```
#ifdef HAL_GPIO_MODULE_ENABLED
  #include "stm32f4xx_hal_gpio.h"
#endif /* HAL_GPIO_MODULE_ENABLED */
#ifdef HAL_EXTI_MODULE_ENABLED
  #include "stm32f4xx_hal_exti.h"
#endif /* HAL_EXTI_MODULE_ENABLED */
#ifdef HAL_DMA_MODULE_ENABLED
  #include "stm32f4xx_hal_dma.h"
#endif /* HAL_DMA_MODULE_ENABLED */
#ifdef HAL_CORTEX_MODULE_ENABLED
  #include "stm32f4xx_hal_cortex.h"
#endif /* HAL_CORTEX_MODULE_ENABLED */
#ifdef HAL_ADC_MODULE_ENABLED
  #include "stm32f4xx_hal_adc.h"
#endif /* HAL_ADC_MODULE_ENABLED */
#ifdef HAL_CAN_MODULE_ENABLED
  #include "stm32f4xx_hal_can.h"
#endif /* HAL_CAN_MODULE_ENABLED */
#ifdef HAL_CAN_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_can_legacy.h"
#endif /* HAL_CAN_LEGACY_MODULE_ENABLED */
#ifdef HAL_CRC_MODULE_ENABLED
  #include "stm32f4xx_hal_crc.h"
```

```
#endif /* HAL_CRC_MODULE_ENABLED */
#ifdef HAL_CRYP_MODULE_ENABLED
  #include "stm32f4xx_hal_cryp.h"
#endif /* HAL_CRYP_MODULE_ENABLED */
#ifdef HAL_DMA2D_MODULE_ENABLED
  #include "stm32f4xx_hal_dma2d.h"
#endif /* HAL_DMA2D_MODULE_ENABLED */
#ifdef HAL_DAC_MODULE_ENABLED
  #include "stm32f4xx_hal_dac.h"
#endif /* HAL_DAC_MODULE_ENABLED */
#ifdef HAL_DCMI_MODULE_ENABLED
  #include "stm32f4xx_hal_dcmi.h"
#endif /* HAL_DCMI_MODULE_ENABLED */
#ifdef HAL_ETH_MODULE_ENABLED
  #include "stm32f4xx_hal_eth.h"
#endif /* HAL_ETH_MODULE_ENABLED */
#ifdef HAL_ETH_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_eth_legacy.h"
#endif /* HAL_ETH_LEGACY_MODULE_ENABLED */
#ifdef HAL_FLASH_MODULE_ENABLED
  #include "stm32f4xx_hal_flash.h"
#endif /* HAL_FLASH_MODULE_ENABLED */
#ifdef HAL_SRAM_MODULE_ENABLED
```

```
  #include "stm32f4xx_hal_sram.h"
#endif /* HAL_SRAM_MODULE_ENABLED */
#ifdef HAL_NOR_MODULE_ENABLED
  #include "stm32f4xx_hal_nor.h"
#endif /* HAL_NOR_MODULE_ENABLED */
#ifdef HAL_NAND_MODULE_ENABLED
  #include "stm32f4xx_hal_nand.h"
#endif /* HAL_NAND_MODULE_ENABLED */
#ifdef HAL_PCCARD_MODULE_ENABLED
  #include "stm32f4xx_hal_pccard.h"
#endif /* HAL_PCCARD_MODULE_ENABLED */
#ifdef HAL_SDRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sdram.h"
#endif /* HAL_SDRAM_MODULE_ENABLED */
#ifdef HAL_HASH_MODULE_ENABLED
 #include "stm32f4xx_hal_hash.h"
#endif /* HAL_HASH_MODULE_ENABLED */
#ifdef HAL_I2C_MODULE_ENABLED
 #include "stm32f4xx_hal_i2c.h"
#endif /* HAL_I2C_MODULE_ENABLED */
#ifdef HAL_SMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_smbus.h"
#endif /* HAL_SMBUS_MODULE_ENABLED */
```

```
#ifdef HAL_I2S_MODULE_ENABLED
 #include "stm32f4xx_hal_i2s.h"
#endif /* HAL_I2S_MODULE_ENABLED */
#ifdef HAL_IWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_iwdg.h"
#endif /* HAL_IWDG_MODULE_ENABLED */
#ifdef HAL_LTDC_MODULE_ENABLED
 #include "stm32f4xx_hal_ltdc.h"
#endif /* HAL_LTDC_MODULE_ENABLED */
#ifdef HAL_PWR_MODULE_ENABLED
 #include "stm32f4xx_hal_pwr.h"
#endif /* HAL_PWR_MODULE_ENABLED */
#ifdef HAL_RNG_MODULE_ENABLED
 #include "stm32f4xx_hal_rng.h"
#endif /* HAL_RNG_MODULE_ENABLED */
#ifdef HAL_RTC_MODULE_ENABLED
 #include "stm32f4xx_hal_rtc.h"
#endif /* HAL_RTC_MODULE_ENABLED */
#ifdef HAL_SAI_MODULE_ENABLED
 #include "stm32f4xx_hal_sai.h"
#endif /* HAL_SAI_MODULE_ENABLED */
#ifdef HAL_SD_MODULE_ENABLED
 #include "stm32f4xx_hal_sd.h"
```

```
#endif /* HAL_SD_MODULE_ENABLED */
#ifdef HAL_SPI_MODULE_ENABLED
 #include "stm32f4xx_hal_spi.h"
#endif /* HAL_SPI_MODULE_ENABLED */
#ifdef HAL_TIM_MODULE_ENABLED
 #include "stm32f4xx_hal_tim.h"
#endif /* HAL_TIM_MODULE_ENABLED */
#ifdef HAL_UART_MODULE_ENABLED
 #include "stm32f4xx_hal_uart.h"
#endif /* HAL_UART_MODULE_ENABLED */
#ifdef HAL_USART_MODULE_ENABLED
 #include "stm32f4xx_hal_usart.h"
#endif /* HAL_USART_MODULE_ENABLED */
#ifdef HAL_IRDA_MODULE_ENABLED
 #include "stm32f4xx_hal_irda.h"
#endif /* HAL_IRDA_MODULE_ENABLED */
#ifdef HAL_SMARTCARD_MODULE_ENABLED
 #include "stm32f4xx_hal_smartcard.h"
#endif /* HAL_SMARTCARD_MODULE_ENABLED */
#ifdef HAL_WWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_wwdg.h"
#endif /* HAL_WWDG_MODULE_ENABLED */
#ifdef HAL_PCD_MODULE_ENABLED
```

```
 #include "stm32f4xx_hal_pcd.h"
#endif /* HAL_PCD_MODULE_ENABLED */
#ifdef HAL_HCD_MODULE_ENABLED
 #include "stm32f4xx_hal_hcd.h"
#endif /* HAL_HCD_MODULE_ENABLED */
#ifdef HAL_DSI_MODULE_ENABLED
 #include "stm32f4xx_hal_dsi.h"
#endif /* HAL_DSI_MODULE_ENABLED */
#ifdef HAL_QSPI_MODULE_ENABLED
 #include "stm32f4xx_hal_qspi.h"
#endif /* HAL_QSPI_MODULE_ENABLED */
#ifdef HAL_CEC_MODULE_ENABLED
 #include "stm32f4xx_hal_cec.h"
#endif /* HAL_CEC_MODULE_ENABLED */
#ifdef HAL_FMPI2C_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpi2c.h"
#endif /* HAL_FMPI2C_MODULE_ENABLED */
#ifdef HAL_FMPSMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpsmbus.h"
#endif /* HAL_FMPSMBUS_MODULE_ENABLED */
#ifdef HAL_SPDIFRX_MODULE_ENABLED
 #include "stm32f4xx_hal_spdifrx.h"
#endif /* HAL_SPDIFRX_MODULE_ENABLED */
```

```
#ifdef HAL_DFSDM_MODULE_ENABLED
 #include "stm32f4xx_hal_dfsdm.h"
#endif /* HAL_DFSDM_MODULE_ENABLED */
#ifdef HAL_LPTIM_MODULE_ENABLED
 #include "stm32f4xx_hal_lptim.h"
#endif /* HAL_LPTIM_MODULE_ENABLED */
#ifdef HAL_MMC_MODULE_ENABLED
 #include "stm32f4xx_hal_mmc.h"
#endif /* HAL_MMC_MODULE_ENABLED */
/* Exported macro ------------------------------------------------------------*/
#ifdef  USE_FULL_ASSERT
/**
  * @brief  The assert_param macro is used for function's parameters check.
  * @param  expr If expr is false, it calls assert_failed function
  *         which reports the name of the source file and the source
  *         line number of the call that failed.
  *         If expr is true, it returns no value.
  * @retval None
  */
  #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t *)__FILE__, __LINE__))
/* Exported functions ------------------------------------------------------- */
  void assert_failed(uint8_t* file, uint32_t line);
```

```
#else
  #define assert_param(expr) ((void)0U)
#endif /* USE_FULL_ASSERT */
#ifdef __cplusplus
}
#endif
#endif /* __STM32F4xx_HAL_CONF_H */
```

26.10 CODE OF STM32F4XX_IT.H:

```
/* USER CODE BEGIN Header */
/**
*******************************************************************
  * @file    stm32f4xx_it.h
  * @brief   This file contains the headers of the interrupt handlers.
*******************************************************************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
```

```
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_IT_H
#define __STM32F4xx_IT_H
#ifdef __cplusplus
 extern "C" {
#endif
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */
/* USER CODE END EC */
/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */
/* USER CODE END EM */
```

```
/* Exported functions prototypes ---------------------------------------------*/
void NMI_Handler(void);
void HardFault_Handler(void);
void MemManage_Handler(void);
void BusFault_Handler(void);
void UsageFault_Handler(void);
void SVC_Handler(void);
void DebugMon_Handler(void);
void PendSV_Handler(void);
void SysTick_Handler(void);
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
#ifdef __cplusplus
}
#endif
#endif /* __STM32F4xx_IT_H */
```

26.11CODE OF MAIN.C:

```
/* USER CODE BEGIN Header */
/**
*****************************************************************************
  * @file           : main.c
```

```
  * @brief          : Main program body
**********************************************************
*******************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
```

```
/* USER CODE END PTD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
UART_HandleTypeDef huart2;
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/**
  * @brief  The application entry point.
  * @retval int
```

```
  */
int main(void)
{
  /* USER CODE BEGIN 1 */
  /* USER CODE END 1 */
  /* MCU Configuration--------------------------------------------------------*/
  /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
  HAL_Init();
  /* USER CODE BEGIN Init */
  /* USER CODE END Init */
  /* Configure the system clock */
  SystemClock_Config();
  /* USER CODE BEGIN SysInit */
  /* USER CODE END SysInit */
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  /* USER CODE BEGIN 2 */
  /* USER CODE END 2 */
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
```

```
  {
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_
VOLTAGE_SCALE1);
  /** Initializes the RCC Oscillators according to the specified
parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
```

```
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue =
RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 16;
  RCC_OscInitStruct.PLL.PLLN = 336;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV4;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource =
RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct,
FLASH_LATENCY_2) != HAL_OK)
```

```
  {
    Error_Handler();
  }
}
/**
  * @brief USART2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART2_UART_Init(void)
{
  /* USER CODE BEGIN USART2_Init 0 */
  /* USER CODE END USART2_Init 0 */
  /* USER CODE BEGIN USART2_Init 1 */
  /* USER CODE END USART2_Init 1 */
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 115200;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
```

```
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART2_Init 2 */
  /* USER CODE END USART2_Init 2 */
}
/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */
  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  /*Configure GPIO pin Output Level */
```

```
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin,
GPIO_PIN_RESET);
  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);
  /*Configure GPIO pin : LD2_Pin */
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);
/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}
/* USER CODE BEGIN 4 */
/* USER CODE END 4 */
/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
```

```
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return
state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}
#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line
number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line
number,
```

```
    ex: printf("Wrong parameters value: file %s on line %d\r\n", file,
line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

26.12 CODE OF STM32F4XX_HAL_MSP.C:

```
/* USER CODE BEGIN Header */
/**
*******************************************************************
  * @file        stm32f4xx_hal_msp.c
  * @brief       This file provides code for the MSP Initialization
  *              and de-Initialization codes.
*******************************************************************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
```

```
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN TD */
/* USER CODE END TD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN Define */
/* USER CODE END Define */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN Macro */
/* USER CODE END Macro */
/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
```

```
/* External functions ------------------------------------------------------*/
/* USER CODE BEGIN ExternalFunctions */
/* USER CODE END ExternalFunctions */
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/**
  * Initializes the Global MSP.
  */
void HAL_MspInit(void)
{
  /* USER CODE BEGIN MspInit 0 */
  /* USER CODE END MspInit 0 */
  __HAL_RCC_SYSCFG_CLK_ENABLE();
  __HAL_RCC_PWR_CLK_ENABLE();
  HAL_NVIC_SetPriorityGrouping(NVIC_PRIORITYGROUP_0);
  /* System interrupt init*/
  /* USER CODE BEGIN MspInit 1 */
  /* USER CODE END MspInit 1 */
}
/**
* @brief UART MSP Initialization
* This function configures the hardware resources used in this example
* @param huart: UART handle pointer
```

```
* @retval None
*/
void HAL_UART_MspInit(UART_HandleTypeDef* huart)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  if(huart->Instance==USART2)
  {
  /* USER CODE BEGIN USART2_MspInit 0 */
  /* USER CODE END USART2_MspInit 0 */
    /* Peripheral clock enable */
    __HAL_RCC_USART2_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();
    /**USART2 GPIO Configuration
    PA2     ------> USART2_TX
    PA3     ------> USART2_RX
    */
    GPIO_InitStruct.Pin = USART_TX_Pin|USART_RX_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF7_USART2;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
  /* USER CODE BEGIN USART2_MspInit 1 */
```

```
  /* USER CODE END USART2_MspInit 1 */
  }
}
/**
* @brief UART MSP De-Initialization
* This function freeze the hardware resources used in this example
* @param huart: UART handle pointer
* @retval None
*/
void HAL_UART_MspDeInit(UART_HandleTypeDef* huart)
{
  if(huart->Instance==USART2)
  {
  /* USER CODE BEGIN USART2_MspDeInit 0 */
  /* USER CODE END USART2_MspDeInit 0 */
    /* Peripheral clock disable */
    __HAL_RCC_USART2_CLK_DISABLE();
    /**USART2 GPIO Configuration
    PA2     ------> USART2_TX
    PA3     ------> USART2_RX
    */
    HAL_GPIO_DeInit(GPIOA, USART_TX_Pin|USART_RX_Pin);
  /* USER CODE BEGIN USART2_MspDeInit 1 */
```

```
  /* USER CODE END USART2_MspDeInit 1 */
  }
}
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

26.13 CODE OF STM32F4XX_IT.C:

```
/* USER CODE BEGIN Header */
/**
*******************************************************************
  * @file    stm32f4xx_it.c
  * @brief   Interrupt Service Routines.
*******************************************************************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
```

```
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stm32f4xx_it.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN TD */
/* USER CODE END TD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
```

```
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/* External variables ---------------------------------------------------------*/
/* USER CODE BEGIN EV */
/* USER CODE END EV */
/******************************************************************************/
/*            Cortex-M4 Processor Interruption and Exception Handlers          */
/******************************************************************************/
/**
  * @brief This function handles Non maskable interrupt.
  */
void NMI_Handler(void)
{
  /* USER CODE BEGIN NonMaskableInt_IRQn 0 */
  /* USER CODE END NonMaskableInt_IRQn 0 */
  /* USER CODE BEGIN NonMaskableInt_IRQn 1 */
   while (1)
   {
```

```
  }
  /* USER CODE END NonMaskableInt_IRQn 1 */
}
/**
  * @brief This function handles Hard fault interrupt.
  */
void HardFault_Handler(void)
{
  /* USER CODE BEGIN HardFault_IRQn 0 */
  /* USER CODE END HardFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_HardFault_IRQn 0 */
    /* USER CODE END W1_HardFault_IRQn 0 */
  }
}
/**
  * @brief This function handles Memory management fault.
  */
void MemManage_Handler(void)
{
  /* USER CODE BEGIN MemoryManagement_IRQn 0 */
  /* USER CODE END MemoryManagement_IRQn 0 */
```

```
  while (1)
  {
    /* USER CODE BEGIN W1_MemoryManagement_IRQn 0 */
    /* USER CODE END W1_MemoryManagement_IRQn 0 */
  }
}
/**
  * @brief This function handles Pre-fetch fault, memory access fault.
  */
void BusFault_Handler(void)
{
  /* USER CODE BEGIN BusFault_IRQn 0 */
  /* USER CODE END BusFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_BusFault_IRQn 0 */
    /* USER CODE END W1_BusFault_IRQn 0 */
  }
}
/**
  * @brief This function handles Undefined instruction or illegal state.
  */
void UsageFault_Handler(void)
```

```
{
  /* USER CODE BEGIN UsageFault_IRQn 0 */
  /* USER CODE END UsageFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_UsageFault_IRQn 0 */
    /* USER CODE END W1_UsageFault_IRQn 0 */
  }
}
/**
  * @brief This function handles System service call via SWI
instruction.
  */
void SVC_Handler(void)
{
  /* USER CODE BEGIN SVCall_IRQn 0 */
  /* USER CODE END SVCall_IRQn 0 */
  /* USER CODE BEGIN SVCall_IRQn 1 */
  /* USER CODE END SVCall_IRQn 1 */
}
/**
  * @brief This function handles Debug monitor.
  */
```

```
void DebugMon_Handler(void)
{
  /* USER CODE BEGIN DebugMonitor_IRQn 0 */
  /* USER CODE END DebugMonitor_IRQn 0 */
  /* USER CODE BEGIN DebugMonitor_IRQn 1 */
  /* USER CODE END DebugMonitor_IRQn 1 */
}
/**
  * @brief This function handles Pendable request for system service.
  */
void PendSV_Handler(void)
{
  /* USER CODE BEGIN PendSV_IRQn 0 */
  /* USER CODE END PendSV_IRQn 0 */
  /* USER CODE BEGIN PendSV_IRQn 1 */
  /* USER CODE END PendSV_IRQn 1 */
}
/**
  * @brief This function handles System tick timer.
  */
void SysTick_Handler(void)
{
  /* USER CODE BEGIN SysTick_IRQn 0 */
```

```
  /* USER CODE END SysTick_IRQn 0 */
  HAL_IncTick();
  /* USER CODE BEGIN SysTick_IRQn 1 */
  /* USER CODE END SysTick_IRQn 1 */
}
/*****************************************************************************/
/* STM32F4xx Peripheral Interrupt Handlers                                   */
/* Add here the Interrupt Handlers for the used peripherals.                 */
/* For the available peripheral interrupt handler names,                     */
/* please refer to the startup file (startup_stm32f4xx.s).                  */
/*****************************************************************************/
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

26.14 CODE OF SYSCALLS.C:

```
/**
*****************************************************************************
 * @file      syscalls.c
 * @author    Auto-generated by STM32CubeIDE
 * @brief     STM32CubeIDE Minimal System calls file
 *
 *            For more information about which c-functions
```

```
 *          need which of these lowlevel functions
 *          please consult the Newlib libc-manual
**********************************************************
*******************
 * @attention
 *
 * Copyright (c) 2020-2024 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the
LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
**********************************************************
*******************
 */
/* Includes */
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
```

```
#include <sys/times.h>
/* Variables */
extern int __io_putchar(int ch) __attribute__((weak));
extern int __io_getchar(void) __attribute__((weak));
char *__env[1] = { 0 };
char **environ = __env;
/* Functions */
void initialise_monitor_handles()
{
}
int _getpid(void)
{
  return 1;
}
int _kill(int pid, int sig)
{
  (void)pid;
  (void)sig;
  errno = EINVAL;
  return -1;
}
void _exit (int status)
{
```

```
  _kill(status, -1);
  while (1) {}    /* Make sure we hang here */
}
__attribute__((weak)) int _read(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;
  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    *ptr++ = __io_getchar();
  }
  return len;
}
__attribute__((weak)) int _write(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;
  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    __io_putchar(*ptr++);
  }
  return len;
}
```

```
int _close(int file)
{
  (void)file;
  return -1;
}
int _fstat(int file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
int _isatty(int file)
{
  (void)file;
  return 1;
}
int _lseek(int file, int ptr, int dir)
{
  (void)file;
  (void)ptr;
  (void)dir;
  return 0;
}
```

```
int _open(char *path, int flags, ...)
{
  (void)path;
  (void)flags;
  /* Pretend like we always fail */
  return -1;
}
int _wait(int *status)
{
  (void)status;
  errno = ECHILD;
  return -1;
}
int _unlink(char *name)
{
  (void)name;
  errno = ENOENT;
  return -1;
}
int _times(struct tms *buf)
{
  (void)buf;
  return -1;
```

```
}
int _stat(char *file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
int _link(char *old, char *new)
{
  (void)old;
  (void)new;
  errno = EMLINK;
  return -1;
}
int _fork(void)
{
  errno = EAGAIN;
  return -1;
}
int _execve(char *name, char **argv, char **env)
{
  (void)name;
  (void)argv;
```

```
  (void)env;
  errno = ENOMEM;
  return -1;
}
```

26.15 CODE OF SYSMEM.C:

```
/**
*****************************************************************
*******************
 * @file      sysmem.c
 * @author    Generated by STM32CubeIDE
 * @brief     STM32CubeIDE System Memory calls file
 *
 *            For more information about which C functions
 *            need which of these lowlevel functions
 *            please consult the newlib libc manual
*****************************************************************
*******************
 * @attention
 *
 * Copyright (c) 2024 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the
LICENSE file
```

```
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
*****************************************************************
*******************
 */
/* Includes */
#include <errno.h>
#include <stdint.h>
/**
 * Pointer to the current high watermark of the heap usage
 */
static uint8_t *__sbrk_heap_end = NULL;
/**
 * @brief _sbrk() allocates memory to the newlib heap and is used by
malloc
 *        and others from the C library
 *
 * @verbatim
 *
#################################################################
##################
 * #  .data  #  .bss  #       newlib heap       #          MSP stack          #
 * #         #        #                          # Reserved by _Min_Stack_Size #
```

```
 *
##################################################################
##################
 * ^-- RAM start      ^-- _end                          _estack, RAM end --^
 * @endverbatim
 *
 * This implementation starts allocating at the '_end' linker symbol
 * The '_Min_Stack_Size' linker symbol reserves a memory for the MSP
stack
 * The implementation considers '_estack' linker symbol to be RAM end
 * NOTE: If the MSP stack, at any point during execution, grows larger
than the
 * reserved size, please increase the '_Min_Stack_Size'.
 *
 * @param incr Memory size
 * @return Pointer to allocated memory
 */
void *_sbrk(ptrdiff_t incr)
{
  extern uint8_t _end; /* Symbol defined in the linker script */
  extern uint8_t _estack; /* Symbol defined in the linker script */
  extern uint32_t _Min_Stack_Size; /* Symbol defined in the linker
script */
  const uint32_t stack_limit = (uint32_t)&_estack -
(uint32_t)&_Min_Stack_Size;
```

```
  const uint8_t *max_heap = (uint8_t *)stack_limit;
  uint8_t *prev_heap_end;
  /* Initialize heap end at first call */
  if (NULL == __sbrk_heap_end)
  {
    __sbrk_heap_end = &_end;
  }
  /* Protect heap from growing into the reserved MSP stack */
  if (__sbrk_heap_end + incr > max_heap)
  {
    errno = ENOMEM;
    return (void *)-1;
  }
  prev_heap_end = __sbrk_heap_end;
  __sbrk_heap_end += incr;
  return (void *)prev_heap_end;
}
```

26.16 CODE OF SYSTEM_STM32F4XX.C:

```
/**
*****************************************************************
*******************
  * @file    system_stm32f4xx.c
  * @author  MCD Application Team
```

```
  * @brief   CMSIS Cortex-M4 Device Peripheral Access Layer System
Source File.
  *
  *   This file provides two functions and one global variable to be called
from
  *   user application:
  *      - SystemInit(): This function is called at startup just after reset and
  *                      before branch to main program. This call is made inside
  *                      the "startup_stm32f4xx.s" file.
  *
  *       - SystemCoreClock variable: Contains the core clock (HCLK), it
can be used
  *                              by the user application to setup the SysTick
  *                              timer or configure other parameters.
  *
  *                      - SystemCoreClockUpdate(): Updates the variable
SystemCoreClock and must
  *                              be called whenever the core clock is changed
  *                              during program execution.
  *
  *
*****************************************************************
******************
  * @attention
  *
```

```
  * Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*****************************************************************************
  */
/** @addtogroup CMSIS
  * @{
  */
/** @addtogroup stm32f4xx_system
  * @{
  */
/** @addtogroup STM32F4xx_System_Private_Includes
  * @{
  */
#include "stm32f4xx.h"
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    ((uint32_t)25000000) /*!< Default value of the External oscillator in Hz */
```

```
#endif /* HSE_VALUE */
#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000) /*!< Value of the Internal oscillator in Hz*/
#endif /* HSI_VALUE */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_TypesDefinitions
  * @{
  */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Defines
  * @{
  */
/************************* Miscellaneous Configuration ************************/
/*!< Uncomment the following line if you need to use external SRAM or SDRAM as data memory  */
#if defined(STM32F405xx) || defined(STM32F415xx) || defined(STM32F407xx) || defined(STM32F417xx)\
 || defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) || defined(STM32F439xx)\
```

|| defined(STM32F469xx) || defined(STM32F479xx) || defined(STM32F412Zx) || defined(STM32F412Vx)

/* #define DATA_IN_ExtSRAM */

#endif /* STM32F40xxx || STM32F41xxx || STM32F42xxx || STM32F43xxx || STM32F469xx || STM32F479xx ||\

STM32F412Zx || STM32F412Vx */

#if defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) || defined(STM32F439xx)\

|| defined(STM32F446xx) || defined(STM32F469xx) || defined(STM32F479xx)

/* #define DATA_IN_ExtSDRAM */

#endif /* STM32F427xx || STM32F437xx || STM32F429xx || STM32F439xx || STM32F446xx || STM32F469xx ||\

STM32F479xx */

/* Note: Following vector table addresses must be defined in line with linker

configuration. */

/*!< Uncomment the following line if you need to relocate the vector table

anywhere in Flash or Sram, else the vector table is kept at the automatic

remap of boot address selected */

/* #define USER_VECT_TAB_ADDRESS */

#if defined(USER_VECT_TAB_ADDRESS)

/*!< Uncomment the following line if you need to relocate your vector Table

in Sram else user remap will be done in Flash. */

```
/* #define VECT_TAB_SRAM */
#if defined(VECT_TAB_SRAM)
#define VECT_TAB_BASE_ADDRESS  SRAM_BASE      /*!< Vector Table base address field.
                                              This value must be a multiple of 0x200.
*/
#define VECT_TAB_OFFSET        0x00000000U    /*!< Vector Table base offset field.
                                              This value must be a multiple of 0x200.
*/
#else
#define VECT_TAB_BASE_ADDRESS  FLASH_BASE     /*!< Vector Table base address field.
                                              This value must be a multiple of 0x200.
*/
#define VECT_TAB_OFFSET        0x00000000U    /*!< Vector Table base offset field.
                                              This value must be a multiple of 0x200.
*/
#endif /* VECT_TAB_SRAM */
#endif /* USER_VECT_TAB_ADDRESS */
/*****************************************************************************/
/**
  * @}
  */
```

```
/** @addtogroup STM32F4xx_System_Private_Macros
  * @{
  */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Variables
  * @{
  */
  /* This variable is updated in three ways:
     1) by calling CMSIS function SystemCoreClockUpdate()
     2) by calling HAL API function HAL_RCC_GetHCLKFreq()
     3) each time HAL_RCC_ClockConfig() is called to configure the system clock frequency
        Note: If you use this function to configure the system clock; then there
              is no need to call the 2 first functions listed above, since SystemCoreClock
              variable is updated automatically.
  */
uint32_t SystemCoreClock = 16000000;
const uint8_t AHBPrescTable[16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9};
const uint8_t APBPrescTable[8]  = {0, 0, 0, 0, 1, 2, 3, 4};
```

```
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_FunctionPrototypes
  * @{
  */
#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  static void SystemInit_ExtMemCtl(void);
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Functions
  * @{
  */
/**
  * @brief  Setup the microcontroller system
  *         Initialize the FPU setting, vector table location and External memory
  *         configuration.
  * @param  None
  * @retval None
  */
```

```
void SystemInit(void)
{
  /* FPU settings ------------------------------------------------------------*/
  #if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
    SCB->CPACR |= ((3UL << 10*2)|(3UL << 11*2));  /* set CP10 and CP11 Full Access */
  #endif
#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  SystemInit_ExtMemCtl();
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */
  /* Configure the Vector Table location -------------------------------------*/
#if defined(USER_VECT_TAB_ADDRESS)
  SCB->VTOR = VECT_TAB_BASE_ADDRESS | VECT_TAB_OFFSET; /* Vector Table Relocation in Internal SRAM */
#endif /* USER_VECT_TAB_ADDRESS */
}
/**
  * @brief  Update SystemCoreClock variable according to Clock Register Values.
  *         The SystemCoreClock variable contains the core clock (HCLK), it can
  *         be used by the user application to setup the SysTick timer or configure
  *         other parameters.
  *
```

```
  * @note   Each time the core clock (HCLK) changes, this function must be called
  *         to update SystemCoreClock variable value. Otherwise, any configuration
  *         based on this variable will be incorrect.
  *
  * @note   - The system frequency computed by this function is not the real
  *           frequency in the chip. It is calculated based on the predefined
  *           constant and the selected clock source:
  *
  *           - If SYSCLK source is HSI, SystemCoreClock will contain the HSI_VALUE(*)
  *
  *           - If SYSCLK source is HSE, SystemCoreClock will contain the HSE_VALUE(**)
  *
  *           - If SYSCLK source is PLL, SystemCoreClock will contain the HSE_VALUE(**)
  *             or HSI_VALUE(*) multiplied/divided by the PLL factors.
  *
  *         (*) HSI_VALUE is a constant defined in stm32f4xx_hal_conf.h file (default value
  *           16 MHz) but the real value may vary depending on the variations
  *             in voltage and temperature.
```

```
  *
  *        (**) HSE_VALUE is a constant defined in stm32f4xx_hal_conf.h file (its value
  *             depends on the application requirements), user has to ensure that HSE_VALUE
  *             is same as the real frequency of the crystal used. Otherwise, this function
  *             may have wrong result.
  *
  *           - The result of this function could be not correct when using fractional
  *           value for HSE crystal.
  *
  * @param  None
  * @retval None
  */
void SystemCoreClockUpdate(void)
{
  uint32_t tmp = 0, pllvco = 0, pllp = 2, pllsource = 0, pllm = 2;
  /* Get SYSCLK source -------------------------------------------------------*/
  tmp = RCC->CFGR & RCC_CFGR_SWS;
  switch (tmp)
  {
    case 0x00:  /* HSI used as system clock source */
```

```
    SystemCoreClock = HSI_VALUE;
    break;
  case 0x04:  /* HSE used as system clock source */
    SystemCoreClock = HSE_VALUE;
    break;
  case 0x08:  /* PLL used as system clock source */
    /* PLL_VCO = (HSE_VALUE or HSI_VALUE / PLL_M) * PLL_N
       SYSCLK = PLL_VCO / PLL_P
       */
    pllsource = (RCC->PLLCFGR & RCC_PLLCFGR_PLLSRC) >> 22;
    pllm = RCC->PLLCFGR & RCC_PLLCFGR_PLLM;
    if (pllsource != 0)
    {
      /* HSE used as PLL clock source */
      pllvco = (HSE_VALUE / pllm) * ((RCC->PLLCFGR & RCC_PLLCFGR_PLLN) >> 6);
    }
    else
    {
      /* HSI used as PLL clock source */
      pllvco = (HSI_VALUE / pllm) * ((RCC->PLLCFGR & RCC_PLLCFGR_PLLN) >> 6);
    }
```

```
    pllp = (((RCC->PLLCFGR & RCC_PLLCFGR_PLLP) >>16) + 1 )
*2;
    SystemCoreClock = pllvco/pllp;
    break;
   default:
    SystemCoreClock = HSI_VALUE;
    break;
  }
  /* Compute HCLK frequency ---------------------------------------------
*/
  /* Get HCLK prescaler */
  tmp = AHBPrescTable[((RCC->CFGR & RCC_CFGR_HPRE) >> 4)];
  /* HCLK frequency */
  SystemCoreClock >>= tmp;
}
#if defined (DATA_IN_ExtSRAM) && defined
(DATA_IN_ExtSDRAM)
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
/**
  * @brief  Setup the external memory controller.
  *         Called in startup_stm32f4xx.s before jump to main.
  *        This function configures the external memories (SRAM/SDRAM)
```

```
  *          This SRAM/SDRAM will be used as program data memory (including heap and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
  register __IO uint32_t index;
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and GPIOI interface clock */
  RCC->AHB1ENR |= 0x000001F8;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x00CCC0CC;
  GPIOD->AFR[1]  = 0xCCCCCCCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xAAAA0A8A;
  /* Configure PDx pins speed to 100 MHz */
  GPIOD->OSPEEDR = 0xFFFF0FCF;
  /* Configure PDx pins Output type to push-pull */
  GPIOD->OTYPER  = 0x00000000;
```

```
/* No pull-up, pull-down for PDx pins */
GPIOD->PUPDR   = 0x00000000;
/* Connect PEx pins to FMC Alternate function */
GPIOE->AFR[0]  = 0xC00CC0CC;
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA828A;
/* Configure PEx pins speed to 100 MHz */
GPIOE->OSPEEDR = 0xFFFFC3CF;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0xCCCCCCCC;
GPIOF->AFR[1]  = 0xCCCCCCCC;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA800AAA;
/* Configure PFx pins speed to 50 MHz */
GPIOF->OSPEEDR = 0xAA800AAA;
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
```

```
GPIOF->PUPDR   = 0x00000000;
/* Connect PGx pins to FMC Alternate function */
GPIOG->AFR[0]  = 0xCCCCCCCC;
GPIOG->AFR[1]  = 0xCCCCCCCC;
/* Configure PGx pins in Alternate function mode */
GPIOG->MODER   = 0xAAAAAAAA;
/* Configure PGx pins speed to 50 MHz */
GPIOG->OSPEEDR = 0xAAAAAAAA;
/* Configure PGx pins Output type to push-pull */
GPIOG->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PGx pins */
GPIOG->PUPDR   = 0x00000000;
/* Connect PHx pins to FMC Alternate function */
GPIOH->AFR[0]  = 0x00C0CC00;
GPIOH->AFR[1]  = 0xCCCCCCCC;
/* Configure PHx pins in Alternate function mode */
GPIOH->MODER   = 0xAAAA08A0;
/* Configure PHx pins speed to 50 MHz */
GPIOH->OSPEEDR = 0xAAAA08A0;
/* Configure PHx pins Output type to push-pull */
GPIOH->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PHx pins */
GPIOH->PUPDR   = 0x00000000;
```

```
  /* Connect PIx pins to FMC Alternate function */
  GPIOI->AFR[0]  = 0xCCCCCCCC;
  GPIOI->AFR[1]  = 0x00000CC0;
  /* Configure PIx pins in Alternate function mode */
  GPIOI->MODER   = 0x0028AAAA;
  /* Configure PIx pins speed to 50 MHz */
  GPIOI->OSPEEDR = 0x0028AAAA;
  /* Configure PIx pins Output type to push-pull */
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;
/*-- FMC Configuration ------------------------------------------------------*/
  /* Enable the FMC interface clock */
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
  FMC_Bank5_6->SDTR[0] = 0x01115351;
  /* SDRAM initialization sequence */
  /* Clock enable command */
  FMC_Bank5_6->SDCMR = 0x00000011;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  while((tmpreg != 0) && (timeout-- > 0))
```

```
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* Delay */
for (index = 0; index<1000; index++);
/* PALL command */
FMC_Bank5_6->SDCMR = 0x00000012;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* Auto refresh command */
FMC_Bank5_6->SDCMR = 0x00000073;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* MRD register program */
FMC_Bank5_6->SDCMR = 0x00046014;
```

```
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Set refresh count */
  tmpreg = FMC_Bank5_6->SDRTR;
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
  /* Disable write protection */
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#if defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) || defined(STM32F439xx)
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx || STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
```

```
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F469xx || STM32F479xx */
  (void)(tmp);
}
#endif /* STM32F427xx || STM32F437xx || STM32F429xx || STM32F439xx || STM32F469xx || STM32F479xx */
#elif defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
/**
  * @brief  Setup the external memory controller.
  *         Called in startup_stm32f4xx.s before jump to main.
  *        This function configures the external memories (SRAM/SDRAM)
  *            This SRAM/SDRAM will be used as program data memory (including heap and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;
#if defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F446xx) || defined(STM32F469xx) || defined(STM32F479xx)
#if defined (DATA_IN_ExtSDRAM)
```

```
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
  register __IO uint32_t index;
#if defined(STM32F446xx)
  /* Enable GPIOA, GPIOC, GPIOD, GPIOE, GPIOF, GPIOG interface
     clock */
  RCC->AHB1ENR |= 0x0000007D;
#else
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and GPIOI
interface
     clock */
  RCC->AHB1ENR |= 0x000001F8;
#endif /* STM32F446xx */
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);
#if defined(STM32F446xx)
  /* Connect PAx pins to FMC Alternate function */
  GPIOA->AFR[0]  |= 0xC0000000;
  GPIOA->AFR[1]  |= 0x00000000;
  /* Configure PDx pins in Alternate function mode */
  GPIOA->MODER   |= 0x00008000;
  /* Configure PDx pins speed to 50 MHz */
  GPIOA->OSPEEDR |= 0x00008000;
  /* Configure PDx pins Output type to push-pull */
```

```
  GPIOA->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOA->PUPDR   |= 0x00000000;
  /* Connect PCx pins to FMC Alternate function */
  GPIOC->AFR[0]  |= 0x00CC0000;
  GPIOC->AFR[1]  |= 0x00000000;
  /* Configure PDx pins in Alternate function mode */
  GPIOC->MODER   |= 0x00000A00;
  /* Configure PDx pins speed to 50 MHz */
  GPIOC->OSPEEDR |= 0x00000A00;
  /* Configure PDx pins Output type to push-pull */
  GPIOC->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOC->PUPDR   |= 0x00000000;
#endif /* STM32F446xx */
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x000000CC;
  GPIOD->AFR[1]  = 0xCC000CCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xA02A000A;
  /* Configure PDx pins speed to 50 MHz */
  GPIOD->OSPEEDR = 0xA02A000A;
  /* Configure PDx pins Output type to push-pull */
```

```
GPIOD->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PDx pins */
GPIOD->PUPDR   = 0x00000000;
/* Connect PEx pins to FMC Alternate function */
GPIOE->AFR[0]  = 0xC00000CC;
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA800A;
/* Configure PEx pins speed to 50 MHz */
GPIOE->OSPEEDR = 0xAAAA800A;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0xCCCCCCCC;
GPIOF->AFR[1]  = 0xCCCCCCCC;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA800AAA;
/* Configure PFx pins speed to 50 MHz */
GPIOF->OSPEEDR = 0xAA800AAA;
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
```

```
  /* No pull-up, pull-down for PFx pins */
  GPIOF->PUPDR   = 0x00000000;
  /* Connect PGx pins to FMC Alternate function */
  GPIOG->AFR[0]  = 0xCCCCCCCC;
  GPIOG->AFR[1]  = 0xCCCCCCCC;
  /* Configure PGx pins in Alternate function mode */
  GPIOG->MODER   = 0xAAAAAAAA;
  /* Configure PGx pins speed to 50 MHz */
  GPIOG->OSPEEDR = 0xAAAAAAAA;
  /* Configure PGx pins Output type to push-pull */
  GPIOG->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;
#if defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
  /* Connect PHx pins to FMC Alternate function */
  GPIOH->AFR[0]  = 0x00C0CC00;
  GPIOH->AFR[1]  = 0xCCCCCCCC;
  /* Configure PHx pins in Alternate function mode */
  GPIOH->MODER   = 0xAAAA08A0;
  /* Configure PHx pins speed to 50 MHz */
  GPIOH->OSPEEDR = 0xAAAA08A0;
```

```
  /* Configure PHx pins Output type to push-pull */
  GPIOH->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PHx pins */
  GPIOH->PUPDR   = 0x00000000;
  /* Connect PIx pins to FMC Alternate function */
  GPIOI->AFR[0]  = 0xCCCCCCCC;
  GPIOI->AFR[1]  = 0x00000CC0;
  /* Configure PIx pins in Alternate function mode */
  GPIOI->MODER   = 0x0028AAAA;
  /* Configure PIx pins speed to 50 MHz */
  GPIOI->OSPEEDR = 0x0028AAAA;
  /* Configure PIx pins Output type to push-pull */
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx || STM32F439xx || STM32F469xx || STM32F479xx */
/*-- FMC Configuration -------------------------------------------------------*/
  /* Enable the FMC interface clock */
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable SDRAM bank1 */
```

```
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCR[0] = 0x00001954;
#else
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
#endif /* STM32F446xx */
  FMC_Bank5_6->SDTR[0] = 0x01115351;
  /* SDRAM initialization sequence */
  /* Clock enable command */
  FMC_Bank5_6->SDCMR = 0x00000011;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Delay */
  for (index = 0; index<1000; index++);
  /* PALL command */
  FMC_Bank5_6->SDCMR = 0x00000012;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
```

```
  }
  /* Auto refresh command */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x000000F3;
#else
  FMC_Bank5_6->SDCMR = 0x00000073;
#endif /* STM32F446xx */
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* MRD register program */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x00044014;
#else
  FMC_Bank5_6->SDCMR = 0x00046014;
#endif /* STM32F446xx */
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
```

```
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Set refresh count */
  tmpreg = FMC_Bank5_6->SDRTR;
#if defined(STM32F446xx)
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000050C<<1));
#else
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
#endif /* STM32F446xx */
  /* Disable write protection */
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#endif /* DATA_IN_ExtSDRAM */
#endif /* STM32F427xx || STM32F437xx || STM32F429xx || STM32F439xx || STM32F446xx || STM32F469xx || STM32F479xx */
#if defined(STM32F405xx) || defined(STM32F415xx) || defined(STM32F407xx) || defined(STM32F417xx)\
 || defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx) || defined(STM32F412Zx) || defined(STM32F412Vx)
#if defined(DATA_IN_ExtSRAM)
/*-- GPIOs Configuration -----------------------------------------------------*/
   /* Enable GPIOD, GPIOE, GPIOF and GPIOG interface clock */
```

```
RCC->AHB1ENR   |= 0x00000078;
/* Delay after an RCC peripheral clock enabling */
tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIODEN);
/* Connect PDx pins to FMC Alternate function */
GPIOD->AFR[0]  = 0x00CCC0CC;
GPIOD->AFR[1]  = 0xCCCCCCCC;
/* Configure PDx pins in Alternate function mode */
GPIOD->MODER   = 0xAAAA0A8A;
/* Configure PDx pins speed to 100 MHz */
GPIOD->OSPEEDR = 0xFFFF0FCF;
/* Configure PDx pins Output type to push-pull */
GPIOD->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PDx pins */
GPIOD->PUPDR   = 0x00000000;
/* Connect PEx pins to FMC Alternate function */
GPIOE->AFR[0]  = 0xC00CC0CC;
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA828A;
/* Configure PEx pins speed to 100 MHz */
GPIOE->OSPEEDR = 0xFFFFC3CF;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
```

```
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0x00CCCCCC;
GPIOF->AFR[1]  = 0xCCCC0000;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA000AAA;
/* Configure PFx pins speed to 100 MHz */
GPIOF->OSPEEDR = 0xFF000FFF;
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
GPIOF->PUPDR   = 0x00000000;
/* Connect PGx pins to FMC Alternate function */
GPIOG->AFR[0]  = 0x00CCCCCC;
GPIOG->AFR[1]  = 0x000000C0;
/* Configure PGx pins in Alternate function mode */
GPIOG->MODER   = 0x00085AAA;
/* Configure PGx pins speed to 100 MHz */
GPIOG->OSPEEDR = 0x000CAFFF;
/* Configure PGx pins Output type to push-pull */
GPIOG->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PGx pins */
```

```
  GPIOG->PUPDR   = 0x00000000;
/*-- FMC/FSMC Configuration ------------------------------------------------
-*/
  /* Enable the FMC/FSMC interface clock */
  RCC->AHB3ENR         |= 0x00000001;
#if defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) || defined(STM32F439xx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx || STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F469xx || STM32F479xx */
```

```
#if defined(STM32F405xx) || defined(STM32F415xx) || defined(STM32F407xx)|| defined(STM32F417xx)\
  || defined(STM32F412Zx) || defined(STM32F412Vx)
 /* Delay after an RCC peripheral clock enabling */
 tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FSMCEN);
 /* Configure and enable Bank1_SRAM2 */
 FSMC_Bank1->BTCR[2]  = 0x00001011;
 FSMC_Bank1->BTCR[3]  = 0x00000201;
 FSMC_Bank1E->BWTR[2] = 0x0FFFFFFF;
#endif /* STM32F405xx || STM32F415xx || STM32F407xx || STM32F417xx || STM32F412Zx || STM32F412Vx */
#endif /* DATA_IN_ExtSRAM */
#endif /* STM32F405xx || STM32F415xx || STM32F407xx || STM32F417xx || STM32F427xx || STM32F437xx ||\
      STM32F429xx || STM32F439xx || STM32F469xx || STM32F479xx || STM32F412Zx || STM32F412Vx  */
 (void)(tmp);
}
#endif /* DATA_IN_ExtSRAM && DATA_IN_ExtSDRAM */
/**
 * @}
 */
/**
 * @}
```

```
  */
/**
  * @}
  */
```

CHAPTER#27 CODING A HARDWARE TIMER DRIVER FOR GENERATING PWM SIGNALS

27.1CODING A HARDWARE TIMER DRIVER FOR GENERATING PWM SIGNALS:

Generating PWM (Pulse Width Modulation) signals using a hardware timer involves configuring the timer in PWM mode, setting the duty cycle, and controlling the output frequency. PWM is commonly used for applications such as motor control, LED dimming, and signal generation. Below is a step-by-step guide to coding a hardware timer driver for generating PWM signals in assembly language for an ARM Cortex-M microcontroller, such as those in the STM32 family.

27.2 UNDERSTANDING THE TIMER PERIPHERAL AND PWM MODE:

Timers in microcontrollers can operate in various modes, including PWM mode, where they generate a signal with a specific frequency and duty cycle. The duty cycle represents the percentage of one period in which the signal is high (on time) versus low (off time).

27.3 SETTING UP THE TIMER FOR PWM:

27.3.1ENABLING THE TIMER CLOCK:

Enable the clock for the timer peripheral in the RCC (Reset and Clock Control) register.

27.3.2 CONFIGURE THE TIMER FOR PWM:

Configure the prescaler and auto-reload values to set the PWM frequency. The PWM frequency is determined by the timer's clock frequency, the prescaler, and the auto-reload value (ARR).

PWM Frequency= Timer Clock⁄(Prescaler+1) ×(ARR+1)Timer Clock

27.3.3 SETTING THE DUTY CYCLE:

Set the duty cycle by configuring the capture/compare register (CCR). The duty cycle percentage is given by:

Duty Cycle (%) =CCR/ARR+1×100

27.3.4 CONFIGURING OUTPUT MODE:

Configure the timer to output PWM on the desired channel. Use the capture/compare mode register to select PWM mode and the capture/compare enable register to enable the output on the corresponding channel.

27.4 STARTING THE TIMER:

Finally, start the timer by enabling the counter.

```
// Enable the counter

LDR R0, =TIM2_CR1

LDR R1, [R0]
```

```
ORR R1, R1, #1          // Set the CEN bit (Counter enable)
STR R1, [R0]
```

27.5 OUTPUT PIN CONFIGURATION:

Ensure that the GPIO pin corresponding to the timer channel is configured as an alternate function to output the PWM signal. For example, if the PWM output is on PA0 (assuming it maps to TIM2_CH1).

27.6ADJUSTING DUTY CYCLE AND FREQUENCY:

To change the duty cycle, update the CCR register. To adjust the frequency, modify the PSC (prescaler) or ARR (auto-reload) values accordingly. Always ensure that changes to these registers are applied safely, especially in real-time applications.

27.7 OPTIMIZATION AND CONSIDERATION:

27.7.1 AVOIDING GLITCHES:

Make changes to CCR, PSC, and ARR registers during a timer update event to avoid glitches.

27.7.2 USING DMA:

If changing the duty cycle frequently, consider using DMA to update CCR values.

27.7.3 OUTPUT POLARITY:

Configure the output polarity if an inverted signal is required.

27.8 CODE OF MAIN.H:

```
/* USER CODE BEGIN Header */
```

```
/**
*****************************************************************
*******************
  * @file           : main.h
  * @brief          : Header for main.c file.
  *                   This file contains the common defines of the application.
*****************************************************************
*******************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*****************************************************************
*******************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -----------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H
```

```
#ifdef __cplusplus
extern "C" {
#endif
/* Includes ------------------------------------------------------------------*/
#include "stm32f4xx_hal.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */
/* USER CODE END EC */
/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */
/* USER CODE END EM */
/* Exported functions prototypes ---------------------------------------------*/
void Error_Handler(void);
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
/* Private defines -----------------------------------------------------------*/
#define B1_Pin GPIO_PIN_13
```

```
#define B1_GPIO_Port GPIOC
#define USART_TX_Pin GPIO_PIN_2
#define USART_TX_GPIO_Port GPIOA
#define USART_RX_Pin GPIO_PIN_3
#define USART_RX_GPIO_Port GPIOA
#define LD2_Pin GPIO_PIN_5
#define LD2_GPIO_Port GPIOA
#define TMS_Pin GPIO_PIN_13
#define TMS_GPIO_Port GPIOA
#define TCK_Pin GPIO_PIN_14
#define TCK_GPIO_Port GPIOA
#define SWO_Pin GPIO_PIN_3
#define SWO_GPIO_Port GPIOB
/* USER CODE BEGIN Private defines */
/* USER CODE END Private defines */
#ifdef __cplusplus
}
#endif
#endif /* __MAIN_H */
```

27.9 CODE OF STM32F4XX_HAL_CONF.H:

```
/* USER CODE BEGIN Header */
```

```
/**
*****************************************************************
******************
  * @file    stm32f4xx_hal_conf_template.h
  * @author  MCD Application Team
  * @brief   HAL configuration template file.
  *          This file should be copied to the application folder and renamed
  *          to stm32f4xx_hal_conf.h.
*****************************************************************
******************
  * @attention
  *
  * Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*****************************************************************
******************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
```

```
#ifndef __STM32F4xx_HAL_CONF_H
#define __STM32F4xx_HAL_CONF_H
#ifdef __cplusplus
 extern "C" {
#endif
/* Exported types ------------------------------------------------------------*/
/* Exported constants --------------------------------------------------------*/
/* ############################ Module Selection
################################## */
/**
  * @brief This is the list of modules to be used in the HAL driver
  */
#define HAL_MODULE_ENABLED
  /* #define HAL_CRYP_MODULE_ENABLED */
/* #define HAL_ADC_MODULE_ENABLED */
/* #define HAL_CAN_MODULE_ENABLED */
/* #define HAL_CRC_MODULE_ENABLED */
/* #define HAL_CAN_LEGACY_MODULE_ENABLED */
/* #define HAL_DAC_MODULE_ENABLED */
/* #define HAL_DCMI_MODULE_ENABLED */
/* #define HAL_DMA2D_MODULE_ENABLED */
/* #define HAL_ETH_MODULE_ENABLED */
/* #define HAL_ETH_LEGACY_MODULE_ENABLED */
```

```
/* #define HAL_NAND_MODULE_ENABLED */
/* #define HAL_NOR_MODULE_ENABLED */
/* #define HAL_PCCARD_MODULE_ENABLED */
/* #define HAL_SRAM_MODULE_ENABLED */
/* #define HAL_SDRAM_MODULE_ENABLED */
/* #define HAL_HASH_MODULE_ENABLED */
/* #define HAL_I2C_MODULE_ENABLED */
/* #define HAL_I2S_MODULE_ENABLED */
/* #define HAL_IWDG_MODULE_ENABLED */
/* #define HAL_LTDC_MODULE_ENABLED */
/* #define HAL_RNG_MODULE_ENABLED */
/* #define HAL_RTC_MODULE_ENABLED */
/* #define HAL_SAI_MODULE_ENABLED */
/* #define HAL_SD_MODULE_ENABLED */
/* #define HAL_MMC_MODULE_ENABLED */
/* #define HAL_SPI_MODULE_ENABLED */
/* #define HAL_TIM_MODULE_ENABLED */
#define HAL_UART_MODULE_ENABLED
/* #define HAL_USART_MODULE_ENABLED */
/* #define HAL_IRDA_MODULE_ENABLED */
/* #define HAL_SMARTCARD_MODULE_ENABLED */
/* #define HAL_SMBUS_MODULE_ENABLED */
/* #define HAL_WWDG_MODULE_ENABLED */
```

```
/* #define HAL_PCD_MODULE_ENABLED */
/* #define HAL_HCD_MODULE_ENABLED */
/* #define HAL_DSI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_CEC_MODULE_ENABLED */
/* #define HAL_FMPI2C_MODULE_ENABLED */
/* #define HAL_FMPSMBUS_MODULE_ENABLED */
/* #define HAL_SPDIFRX_MODULE_ENABLED */
/* #define HAL_DFSDM_MODULE_ENABLED */
/* #define HAL_LPTIM_MODULE_ENABLED */
#define HAL_GPIO_MODULE_ENABLED
#define HAL_EXTI_MODULE_ENABLED
#define HAL_DMA_MODULE_ENABLED
#define HAL_RCC_MODULE_ENABLED
#define HAL_FLASH_MODULE_ENABLED
#define HAL_PWR_MODULE_ENABLED
#define HAL_CORTEX_MODULE_ENABLED
/* ############################ HSE/HSI Values adaptation
###################### */
/**
  * @brief Adjust the value of External High Speed oscillator (HSE)
used in your application.
```

```
  *       This value is used by the RCC HAL module to compute the
system frequency
  *       (when HSE is used as system clock source, directly or through
the PLL).
  */
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    8000000U /*!< Value of the External
oscillator in Hz */
#endif /* HSE_VALUE */
#if !defined  (HSE_STARTUP_TIMEOUT)
  #define HSE_STARTUP_TIMEOUT    100U   /*!< Time out for
HSE start up, in ms */
#endif /* HSE_STARTUP_TIMEOUT */
/**
  * @brief Internal High Speed oscillator (HSI) value.
  *       This value is used by the RCC HAL module to compute the
system frequency
  *       (when HSI is used as system clock source, directly or through the
PLL).
  */
#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000U) /*!< Value of the
Internal oscillator in Hz*/
#endif /* HSI_VALUE */
/**
```

```
  * @brief Internal Low Speed oscillator (LSI) value.
  */
#if !defined  (LSI_VALUE)
 #define LSI_VALUE  32000U       /*!< LSI Typical Value in Hz*/
#endif /* LSI_VALUE */                    /*!< Value of the Internal Low
Speed oscillator in Hz
                                 The real value may vary depending on the
variations
                                 in voltage and temperature.*/
/**
  * @brief External Low Speed oscillator (LSE) value.
  */
#if !defined  (LSE_VALUE)
 #define LSE_VALUE  32768U    /*!< Value of the External Low Speed
oscillator in Hz */
#endif /* LSE_VALUE */
#if !defined  (LSE_STARTUP_TIMEOUT)
  #define LSE_STARTUP_TIMEOUT    5000U   /*!< Time out for
LSE start up, in ms */
#endif /* LSE_STARTUP_TIMEOUT */
/**
  * @brief External clock source for I2S peripheral
  *        This value is used by the I2S HAL module to compute the I2S
clock source
```

```
  *       frequency, this source is inserted directly through I2S_CKIN
pad.
  */
#if !defined  (EXTERNAL_CLOCK_VALUE)
  #define EXTERNAL_CLOCK_VALUE    12288000U /*!< Value of
the External audio frequency in Hz*/
#endif /* EXTERNAL_CLOCK_VALUE */
/* Tip: To avoid modifying this file each time you need to use different
HSE,
   ===  you can define the HSE value in your toolchain compiler
preprocessor. */
/* ############################## System Configuration
########################### */
/**
  * @brief This is the HAL system configuration section
  */
#define  VDD_VALUE                    3300U /*!< Value of VDD in mv */
#define  TICK_INT_PRIORITY            0U   /*!< tick interrupt
priority */
#define  USE_RTOS                     0U
#define  PREFETCH_ENABLE              1U
#define  INSTRUCTION_CACHE_ENABLE     1U
#define  DATA_CACHE_ENABLE            1U
#define  USE_HAL_ADC_REGISTER_CALLBACKS        0U /*
ADC register callback disabled       */
```

#define USE_HAL_CAN_REGISTER_CALLBACKS 0U /* CAN register callback disabled */

#define USE_HAL_CEC_REGISTER_CALLBACKS 0U /* CEC register callback disabled */

#define USE_HAL_CRYP_REGISTER_CALLBACKS 0U /* CRYP register callback disabled */

#define USE_HAL_DAC_REGISTER_CALLBACKS 0U /* DAC register callback disabled */

#define USE_HAL_DCMI_REGISTER_CALLBACKS 0U /* DCMI register callback disabled */

#define USE_HAL_DFSDM_REGISTER_CALLBACKS 0U /* DFSDM register callback disabled */

#define USE_HAL_DMA2D_REGISTER_CALLBACKS 0U /* DMA2D register callback disabled */

#define USE_HAL_DSI_REGISTER_CALLBACKS 0U /* DSI register callback disabled */

#define USE_HAL_ETH_REGISTER_CALLBACKS 0U /* ETH register callback disabled */

#define USE_HAL_HASH_REGISTER_CALLBACKS 0U /* HASH register callback disabled */

#define USE_HAL_HCD_REGISTER_CALLBACKS 0U /* HCD register callback disabled */

#define USE_HAL_I2C_REGISTER_CALLBACKS 0U /* I2C register callback disabled */

#define USE_HAL_FMPI2C_REGISTER_CALLBACKS 0U /* FMPI2C register callback disabled */

#define USE_HAL_FMPSMBUS_REGISTER_CALLBACKS 0U /* FMPSMBUS register callback disabled */

#define USE_HAL_I2S_REGISTER_CALLBACKS 0U /* I2S register callback disabled */

#define USE_HAL_IRDA_REGISTER_CALLBACKS 0U /* IRDA register callback disabled */

#define USE_HAL_LPTIM_REGISTER_CALLBACKS 0U /* LPTIM register callback disabled */

#define USE_HAL_LTDC_REGISTER_CALLBACKS 0U /* LTDC register callback disabled */

#define USE_HAL_MMC_REGISTER_CALLBACKS 0U /* MMC register callback disabled */

#define USE_HAL_NAND_REGISTER_CALLBACKS 0U /* NAND register callback disabled */

#define USE_HAL_NOR_REGISTER_CALLBACKS 0U /* NOR register callback disabled */

#define USE_HAL_PCCARD_REGISTER_CALLBACKS 0U /* PCCARD register callback disabled */

#define USE_HAL_PCD_REGISTER_CALLBACKS 0U /* PCD register callback disabled */

#define USE_HAL_QSPI_REGISTER_CALLBACKS 0U /* QSPI register callback disabled */

#define USE_HAL_RNG_REGISTER_CALLBACKS 0U /* RNG register callback disabled */

#define USE_HAL_RTC_REGISTER_CALLBACKS 0U /* RTC register callback disabled */

#define USE_HAL_SAI_REGISTER_CALLBACKS 0U /* SAI register callback disabled */

#define USE_HAL_SD_REGISTER_CALLBACKS 0U /* SD register callback disabled */

#define USE_HAL_SMARTCARD_REGISTER_CALLBACKS 0U /* SMARTCARD register callback disabled */

#define USE_HAL_SDRAM_REGISTER_CALLBACKS 0U /* SDRAM register callback disabled */

#define USE_HAL_SRAM_REGISTER_CALLBACKS 0U /* SRAM register callback disabled */

#define USE_HAL_SPDIFRX_REGISTER_CALLBACKS 0U /* SPDIFRX register callback disabled */

#define USE_HAL_SMBUS_REGISTER_CALLBACKS 0U /* SMBUS register callback disabled */

#define USE_HAL_SPI_REGISTER_CALLBACKS 0U /* SPI register callback disabled */

#define USE_HAL_TIM_REGISTER_CALLBACKS 0U /* TIM register callback disabled */

#define USE_HAL_UART_REGISTER_CALLBACKS 0U /* UART register callback disabled */

#define USE_HAL_USART_REGISTER_CALLBACKS 0U /* USART register callback disabled */

#define USE_HAL_WWDG_REGISTER_CALLBACKS 0U /* WWDG register callback disabled */

/* ############################ Assert Selection ################################ */

/**

```
  * @brief Uncomment the line below to expanse the "assert_param"
macro in the
  *        HAL drivers code
  */
/* #define USE_FULL_ASSERT    1U */
/* ################### Ethernet peripheral configuration
###################### */
/* Section 1 : Ethernet peripheral configuration */
/* MAC ADDRESS:
MAC_ADDR0:MAC_ADDR1:MAC_ADDR2:MAC_ADDR3:MAC_A
DDR4:MAC_ADDR5 */
#define MAC_ADDR0   2U
#define MAC_ADDR1   0U
#define MAC_ADDR2   0U
#define MAC_ADDR3   0U
#define MAC_ADDR4   0U
#define MAC_ADDR5   0U
/* Definition of the Ethernet driver buffers size and count */
#define ETH_RX_BUF_SIZE             ETH_MAX_PACKET_SIZE
/* buffer size for receive              */
#define ETH_TX_BUF_SIZE             ETH_MAX_PACKET_SIZE
/* buffer size for transmit             */
#define ETH_RXBUFNB                 4U       /* 4 Rx buffers of size
ETH_RX_BUF_SIZE  */
```

#define ETH_TXBUFNB 4U /* 4 Tx buffers of size ETH_TX_BUF_SIZE */

/* Section 2: PHY configuration section */

/* DP83848_PHY_ADDRESS Address*/

#define DP83848_PHY_ADDRESS

/* PHY Reset delay these values are based on a 1 ms Systick interrupt*/

#define PHY_RESET_DELAY 0x000000FFU

/* PHY Configuration delay */

#define PHY_CONFIG_DELAY 0x00000FFFU

#define PHY_READ_TO 0x0000FFFFU

#define PHY_WRITE_TO 0x0000FFFFU

/* Section 3: Common PHY Registers */

#define PHY_BCR ((uint16_t)0x0000U) /*!< Transceiver Basic Control Register */

#define PHY_BSR ((uint16_t)0x0001U) /*!< Transceiver Basic Status Register */

#define PHY_RESET ((uint16_t)0x8000U) /*!< PHY Reset */

#define PHY_LOOPBACK ((uint16_t)0x4000U) /*!< Select loop-back mode */

#define PHY_FULLDUPLEX_100M ((uint16_t)0x2100U) /*!< Set the full-duplex mode at 100 Mb/s */

#define PHY_HALFDUPLEX_100M ((uint16_t)0x2000U) /*!< Set the half-duplex mode at 100 Mb/s */

#define PHY_FULLDUPLEX_10M ((uint16_t)0x0100U) /*!< Set the full-duplex mode at 10 Mb/s */

#define PHY_HALFDUPLEX_10M ((uint16_t)0x0000U) /*!< Set the half-duplex mode at 10 Mb/s */

#define PHY_AUTONEGOTIATION ((uint16_t)0x1000U) /*!< Enable auto-negotiation function */

#define PHY_RESTART_AUTONEGOTIATION ((uint16_t)0x0200U) /*!< Restart auto-negotiation function */

#define PHY_POWERDOWN ((uint16_t)0x0800U) /*!< Select the power down mode */

#define PHY_ISOLATE ((uint16_t)0x0400U) /*!< Isolate PHY from MII */

#define PHY_AUTONEGO_COMPLETE ((uint16_t)0x0020U) /*!< Auto-Negotiation process completed */

#define PHY_LINKED_STATUS ((uint16_t)0x0004U) /*!< Valid link established */

#define PHY_JABBER_DETECTION ((uint16_t)0x0002U) /*!< Jabber condition detected */

/* Section 4: Extended PHY Registers */

#define PHY_SR ((uint16_t)) /*!< PHY status register Offset */

#define PHY_SPEED_STATUS ((uint16_t)) /*!< PHY Speed mask */

#define PHY_DUPLEX_STATUS ((uint16_t)) /*!< PHY Duplex mask */

/* ################### SPI peripheral configuration ############################ */

```
/* CRC FEATURE: Use to activate CRC feature inside HAL SPI Driver
* Activated: CRC code is present inside driver
* Deactivated: CRC code cleaned from driver
*/
#define USE_SPI_CRC                    0U
/* Includes ------------------------------------------------------------------*/
/**
  * @brief Include module's header file
  */
#ifdef HAL_RCC_MODULE_ENABLED
  #include "stm32f4xx_hal_rcc.h"
#endif /* HAL_RCC_MODULE_ENABLED */
#ifdef HAL_GPIO_MODULE_ENABLED
  #include "stm32f4xx_hal_gpio.h"
#endif /* HAL_GPIO_MODULE_ENABLED */
#ifdef HAL_EXTI_MODULE_ENABLED
  #include "stm32f4xx_hal_exti.h"
#endif /* HAL_EXTI_MODULE_ENABLED */
#ifdef HAL_DMA_MODULE_ENABLED
  #include "stm32f4xx_hal_dma.h"
#endif /* HAL_DMA_MODULE_ENABLED */
#ifdef HAL_CORTEX_MODULE_ENABLED
  #include "stm32f4xx_hal_cortex.h"
```

```
#endif /* HAL_CORTEX_MODULE_ENABLED */
#ifdef HAL_ADC_MODULE_ENABLED
  #include "stm32f4xx_hal_adc.h"
#endif /* HAL_ADC_MODULE_ENABLED */
#ifdef HAL_CAN_MODULE_ENABLED
  #include "stm32f4xx_hal_can.h"
#endif /* HAL_CAN_MODULE_ENABLED */
#ifdef HAL_CAN_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_can_legacy.h"
#endif /* HAL_CAN_LEGACY_MODULE_ENABLED */
#ifdef HAL_CRC_MODULE_ENABLED
  #include "stm32f4xx_hal_crc.h"
#endif /* HAL_CRC_MODULE_ENABLED */
#ifdef HAL_CRYP_MODULE_ENABLED
  #include "stm32f4xx_hal_cryp.h"
#endif /* HAL_CRYP_MODULE_ENABLED */
#ifdef HAL_DMA2D_MODULE_ENABLED
  #include "stm32f4xx_hal_dma2d.h"
#endif /* HAL_DMA2D_MODULE_ENABLED */
#ifdef HAL_DAC_MODULE_ENABLED
  #include "stm32f4xx_hal_dac.h"
#endif /* HAL_DAC_MODULE_ENABLED */
#ifdef HAL_DCMI_MODULE_ENABLED
```

```
  #include "stm32f4xx_hal_dcmi.h"
#endif /* HAL_DCMI_MODULE_ENABLED */
#ifdef HAL_ETH_MODULE_ENABLED
  #include "stm32f4xx_hal_eth.h"
#endif /* HAL_ETH_MODULE_ENABLED */
#ifdef HAL_ETH_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_eth_legacy.h"
#endif /* HAL_ETH_LEGACY_MODULE_ENABLED */
#ifdef HAL_FLASH_MODULE_ENABLED
  #include "stm32f4xx_hal_flash.h"
#endif /* HAL_FLASH_MODULE_ENABLED */
#ifdef HAL_SRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sram.h"
#endif /* HAL_SRAM_MODULE_ENABLED */
#ifdef HAL_NOR_MODULE_ENABLED
  #include "stm32f4xx_hal_nor.h"
#endif /* HAL_NOR_MODULE_ENABLED */
#ifdef HAL_NAND_MODULE_ENABLED
  #include "stm32f4xx_hal_nand.h"
#endif /* HAL_NAND_MODULE_ENABLED */
#ifdef HAL_PCCARD_MODULE_ENABLED
  #include "stm32f4xx_hal_pccard.h"
#endif /* HAL_PCCARD_MODULE_ENABLED */
```

```
#ifdef HAL_SDRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sdram.h"
#endif /* HAL_SDRAM_MODULE_ENABLED */
#ifdef HAL_HASH_MODULE_ENABLED
 #include "stm32f4xx_hal_hash.h"
#endif /* HAL_HASH_MODULE_ENABLED */
#ifdef HAL_I2C_MODULE_ENABLED
 #include "stm32f4xx_hal_i2c.h"
#endif /* HAL_I2C_MODULE_ENABLED */
#ifdef HAL_SMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_smbus.h"
#endif /* HAL_SMBUS_MODULE_ENABLED */
#ifdef HAL_I2S_MODULE_ENABLED
 #include "stm32f4xx_hal_i2s.h"
#endif /* HAL_I2S_MODULE_ENABLED */
#ifdef HAL_IWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_iwdg.h"
#endif /* HAL_IWDG_MODULE_ENABLED */
#ifdef HAL_LTDC_MODULE_ENABLED
 #include "stm32f4xx_hal_ltdc.h"
#endif /* HAL_LTDC_MODULE_ENABLED */
#ifdef HAL_PWR_MODULE_ENABLED
 #include "stm32f4xx_hal_pwr.h"
```

```
#endif /* HAL_PWR_MODULE_ENABLED */
#ifdef HAL_RNG_MODULE_ENABLED
 #include "stm32f4xx_hal_rng.h"
#endif /* HAL_RNG_MODULE_ENABLED */
#ifdef HAL_RTC_MODULE_ENABLED
 #include "stm32f4xx_hal_rtc.h"
#endif /* HAL_RTC_MODULE_ENABLED */
#ifdef HAL_SAI_MODULE_ENABLED
 #include "stm32f4xx_hal_sai.h"
#endif /* HAL_SAI_MODULE_ENABLED */
#ifdef HAL_SD_MODULE_ENABLED
 #include "stm32f4xx_hal_sd.h"
#endif /* HAL_SD_MODULE_ENABLED */
#ifdef HAL_SPI_MODULE_ENABLED
 #include "stm32f4xx_hal_spi.h"
#endif /* HAL_SPI_MODULE_ENABLED */
#ifdef HAL_TIM_MODULE_ENABLED
 #include "stm32f4xx_hal_tim.h"
#endif /* HAL_TIM_MODULE_ENABLED */
#ifdef HAL_UART_MODULE_ENABLED
 #include "stm32f4xx_hal_uart.h"
#endif /* HAL_UART_MODULE_ENABLED */
#ifdef HAL_USART_MODULE_ENABLED
```

```
 #include "stm32f4xx_hal_usart.h"
#endif /* HAL_USART_MODULE_ENABLED */
#ifdef HAL_IRDA_MODULE_ENABLED
 #include "stm32f4xx_hal_irda.h"
#endif /* HAL_IRDA_MODULE_ENABLED */
#ifdef HAL_SMARTCARD_MODULE_ENABLED
 #include "stm32f4xx_hal_smartcard.h"
#endif /* HAL_SMARTCARD_MODULE_ENABLED */
#ifdef HAL_WWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_wwdg.h"
#endif /* HAL_WWDG_MODULE_ENABLED */
#ifdef HAL_PCD_MODULE_ENABLED
 #include "stm32f4xx_hal_pcd.h"
#endif /* HAL_PCD_MODULE_ENABLED */
#ifdef HAL_HCD_MODULE_ENABLED
 #include "stm32f4xx_hal_hcd.h"
#endif /* HAL_HCD_MODULE_ENABLED */
#ifdef HAL_DSI_MODULE_ENABLED
 #include "stm32f4xx_hal_dsi.h"
#endif /* HAL_DSI_MODULE_ENABLED */
#ifdef HAL_QSPI_MODULE_ENABLED
 #include "stm32f4xx_hal_qspi.h"
#endif /* HAL_QSPI_MODULE_ENABLED */
```

```
#ifdef HAL_CEC_MODULE_ENABLED
 #include "stm32f4xx_hal_cec.h"
#endif /* HAL_CEC_MODULE_ENABLED */
#ifdef HAL_FMPI2C_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpi2c.h"
#endif /* HAL_FMPI2C_MODULE_ENABLED */
#ifdef HAL_FMPSMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpsmbus.h"
#endif /* HAL_FMPSMBUS_MODULE_ENABLED */
#ifdef HAL_SPDIFRX_MODULE_ENABLED
 #include "stm32f4xx_hal_spdifrx.h"
#endif /* HAL_SPDIFRX_MODULE_ENABLED */
#ifdef HAL_DFSDM_MODULE_ENABLED
 #include "stm32f4xx_hal_dfsdm.h"
#endif /* HAL_DFSDM_MODULE_ENABLED */
#ifdef HAL_LPTIM_MODULE_ENABLED
 #include "stm32f4xx_hal_lptim.h"
#endif /* HAL_LPTIM_MODULE_ENABLED */
#ifdef HAL_MMC_MODULE_ENABLED
 #include "stm32f4xx_hal_mmc.h"
#endif /* HAL_MMC_MODULE_ENABLED */
/* Exported macro ------------------------------------------------------------*/
#ifdef  USE_FULL_ASSERT
```

```
/**
  * @brief  The assert_param macro is used for function's parameters check.
  * @param  expr If expr is false, it calls assert_failed function
  *         which reports the name of the source file and the source
  *         line number of the call that failed.
  *         If expr is true, it returns no value.
  * @retval None
  */
  #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t *)__FILE__, __LINE__))
/* Exported functions ---------------------------------------------------------- */
  void assert_failed(uint8_t* file, uint32_t line);
#else
  #define assert_param(expr) ((void)0U)
#endif /* USE_FULL_ASSERT */
#ifdef __cplusplus
}
#endif
#endif /* __STM32F4xx_HAL_CONF_H */
```

27.10 CODE OF STM32F4XX_IT.H:

```
/* USER CODE BEGIN Header */
/**
```

```
*****************************************************
*******************
  * @file    stm32f4xx_it.h
  * @brief   This file contains the headers of the interrupt handlers.
*****************************************************
*******************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*****************************************************
*******************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -----------------------------------*/
#ifndef __STM32F4xx_IT_H
#define __STM32F4xx_IT_H
#ifdef __cplusplus
 extern "C" {
```

```
#endif
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */
/* USER CODE END EC */
/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */
/* USER CODE END EM */
/* Exported functions prototypes ---------------------------------------------*/
void NMI_Handler(void);
void HardFault_Handler(void);
void MemManage_Handler(void);
void BusFault_Handler(void);
void UsageFault_Handler(void);
void SVC_Handler(void);
void DebugMon_Handler(void);
void PendSV_Handler(void);
void SysTick_Handler(void);
```

```
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
#ifdef __cplusplus
}
#endif
#endif /* __STM32F4xx_IT_H */
```

27.11 CODE OF MAIN.C:

```
/* USER CODE BEGIN Header */
/**
*******************************************************************
  * @file           : main.c
  * @brief          : Main program body
*******************************************************************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
```

```
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
/* USER CODE END PTD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
UART_HandleTypeDef huart2;
/* USER CODE BEGIN PV */
/* USER CODE END PV */
```

```
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */
  /* USER CODE END 1 */
  /* MCU Configuration--------------------------------------------------------*/
  /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
  HAL_Init();
  /* USER CODE BEGIN Init */
  /* USER CODE END Init */
```

```
  /* Configure the system clock */
  SystemClock_Config();
  /* USER CODE BEGIN SysInit */
  /* USER CODE END SysInit */
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  /* USER CODE BEGIN 2 */
  /* USER CODE END 2 */
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
```

```
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_
VOLTAGE_SCALE1);
  /** Initializes the RCC Oscillators according to the specified
parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue =
RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 16;
  RCC_OscInitStruct.PLL.PLLN = 336;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV4;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
```

```
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource =
RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct,
FLASH_LATENCY_2) != HAL_OK)
  {
    Error_Handler();
  }
}
/**
  * @brief USART2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART2_UART_Init(void)
```

```
{
  /* USER CODE BEGIN USART2_Init 0 */
  /* USER CODE END USART2_Init 0 */
  /* USER CODE BEGIN USART2_Init 1 */
  /* USER CODE END USART2_Init 1 */
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 115200;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART2_Init 2 */
  /* USER CODE END USART2_Init 2 */
}
/**
  * @brief GPIO Initialization Function
  * @param None
```

```
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */
  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin,
GPIO_PIN_RESET);
  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);
  /*Configure GPIO pin : LD2_Pin */
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
```

```
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);
/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}
/* USER CODE BEGIN 4 */
/* USER CODE END 4 */
/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return
state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}
```

```
#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

27.11 CODE OF STM32F4XX_HAL_MSP.C:

```
/* USER CODE BEGIN Header */
/**
*******************************************************************
*******************
```

```
  * @file         stm32f4xx_hal_msp.c
  * @brief        This file provides code for the MSP Initialization
  *               and de-Initialization codes.
**********************************************************
*******************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
```

```
/* USER CODE BEGIN TD */
/* USER CODE END TD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN Define */
/* USER CODE END Define */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN Macro */
/* USER CODE END Macro */
/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* External functions --------------------------------------------------------*/
/* USER CODE BEGIN ExternalFunctions */
/* USER CODE END ExternalFunctions */
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/**
  * Initializes the Global MSP.
  */
void HAL_MspInit(void)
```

```
{
  /* USER CODE BEGIN MspInit 0 */
  /* USER CODE END MspInit 0 */
  __HAL_RCC_SYSCFG_CLK_ENABLE();
  __HAL_RCC_PWR_CLK_ENABLE();
  HAL_NVIC_SetPriorityGrouping(NVIC_PRIORITYGROUP_0);
  /* System interrupt init*/
  /* USER CODE BEGIN MspInit 1 */
  /* USER CODE END MspInit 1 */
}
/**
* @brief UART MSP Initialization
* This function configures the hardware resources used in this example
* @param huart: UART handle pointer
* @retval None
*/
void HAL_UART_MspInit(UART_HandleTypeDef* huart)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  if(huart->Instance==USART2)
  {
  /* USER CODE BEGIN USART2_MspInit 0 */
  /* USER CODE END USART2_MspInit 0 */
```

```
    /* Peripheral clock enable */
    __HAL_RCC_USART2_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();
    /**USART2 GPIO Configuration
    PA2     ------> USART2_TX
    PA3     ------> USART2_RX
    */
    GPIO_InitStruct.Pin = USART_TX_Pin|USART_RX_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF7_USART2;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
  /* USER CODE BEGIN USART2_MspInit 1 */
  /* USER CODE END USART2_MspInit 1 */
  }
}
/**
* @brief UART MSP De-Initialization
* This function freeze the hardware resources used in this example
* @param huart: UART handle pointer
* @retval None
*/
```

```
void HAL_UART_MspDeInit(UART_HandleTypeDef* huart)
{
  if(huart->Instance==USART2)
  {
  /* USER CODE BEGIN USART2_MspDeInit 0 */
  /* USER CODE END USART2_MspDeInit 0 */
    /* Peripheral clock disable */
    __HAL_RCC_USART2_CLK_DISABLE();
    /**USART2 GPIO Configuration
    PA2     ------> USART2_TX
    PA3     ------> USART2_RX
    */
    HAL_GPIO_DeInit(GPIOA, USART_TX_Pin|USART_RX_Pin);
  /* USER CODE BEGIN USART2_MspDeInit 1 */
  /* USER CODE END USART2_MspDeInit 1 */
  }
}
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

27.12 CODE OF STM32F4XX_IT.C:

```
/* USER CODE BEGIN Header */
/**
```

```
**********************************************************
*******************
  * @file    stm32f4xx_it.c
  * @brief   Interrupt Service Routines.
**********************************************************
*******************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stm32f4xx_it.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
```

```
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN TD */
/* USER CODE END TD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/* External variables --------------------------------------------------------*/
/* USER CODE BEGIN EV */
/* USER CODE END EV */
```

```
/*****************************************************************
*********************/
/*          Cortex-M4 Processor Interruption and Exception Handlers
*/
/*****************************************************************
*********************/
/**
  * @brief This function handles Non maskable interrupt.
  */
void NMI_Handler(void)
{
  /* USER CODE BEGIN NonMaskableInt_IRQn 0 */
  /* USER CODE END NonMaskableInt_IRQn 0 */
  /* USER CODE BEGIN NonMaskableInt_IRQn 1 */
   while (1)
  {
  }
  /* USER CODE END NonMaskableInt_IRQn 1 */
}
/**
  * @brief This function handles Hard fault interrupt.
  */
void HardFault_Handler(void)
{
```

```
  /* USER CODE BEGIN HardFault_IRQn 0 */
  /* USER CODE END HardFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_HardFault_IRQn 0 */
    /* USER CODE END W1_HardFault_IRQn 0 */
  }
}
/**
  * @brief This function handles Memory management fault.
  */
void MemManage_Handler(void)
{
  /* USER CODE BEGIN MemoryManagement_IRQn 0 */
  /* USER CODE END MemoryManagement_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_MemoryManagement_IRQn 0 */
    /* USER CODE END W1_MemoryManagement_IRQn 0 */
  }
}
/**
  * @brief This function handles Pre-fetch fault, memory access fault.
```

```
  */
void BusFault_Handler(void)
{
  /* USER CODE BEGIN BusFault_IRQn 0 */
  /* USER CODE END BusFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_BusFault_IRQn 0 */
    /* USER CODE END W1_BusFault_IRQn 0 */
  }
}
/**
  * @brief This function handles Undefined instruction or illegal state.
  */
void UsageFault_Handler(void)
{
  /* USER CODE BEGIN UsageFault_IRQn 0 */
  /* USER CODE END UsageFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_UsageFault_IRQn 0 */
    /* USER CODE END W1_UsageFault_IRQn 0 */
  }
```

```
}
/**
  * @brief This function handles System service call via SWI
instruction.
  */
void SVC_Handler(void)
{
  /* USER CODE BEGIN SVCall_IRQn 0 */
  /* USER CODE END SVCall_IRQn 0 */
  /* USER CODE BEGIN SVCall_IRQn 1 */
  /* USER CODE END SVCall_IRQn 1 */
}
/**
  * @brief This function handles Debug monitor.
  */
void DebugMon_Handler(void)
{
  /* USER CODE BEGIN DebugMonitor_IRQn 0 */
  /* USER CODE END DebugMonitor_IRQn 0 */
  /* USER CODE BEGIN DebugMonitor_IRQn 1 */
  /* USER CODE END DebugMonitor_IRQn 1 */
}
/**
```

```
  * @brief This function handles Pendable request for system service.
  */
void PendSV_Handler(void)
{
  /* USER CODE BEGIN PendSV_IRQn 0 */
  /* USER CODE END PendSV_IRQn 0 */
  /* USER CODE BEGIN PendSV_IRQn 1 */
  /* USER CODE END PendSV_IRQn 1 */
}
/**
  * @brief This function handles System tick timer.
  */
void SysTick_Handler(void)
{
  /* USER CODE BEGIN SysTick_IRQn 0 */
  /* USER CODE END SysTick_IRQn 0 */
  HAL_IncTick();
  /* USER CODE BEGIN SysTick_IRQn 1 */
  /* USER CODE END SysTick_IRQn 1 */
}
/******************************************************************************/
/* STM32F4xx Peripheral Interrupt Handlers                                    */
```

```
/* Add here the Interrupt Handlers for the used peripherals.                  */
/* For the available peripheral interrupt handler names,                      */
/* please refer to the startup file (startup_stm32f4xx.s).                    */
/*****************************************************************************/
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

27.13 CODE OF SYSCALLS.C:

```
/**
*****************************************************************************
 * @file      syscalls.c
 * @author    Auto-generated by STM32CubeIDE
 * @brief     STM32CubeIDE Minimal System calls file
 *
 *            For more information about which c-functions
 *            need which of these lowlevel functions
 *            please consult the Newlib libc-manual
*****************************************************************************
 * @attention
 *
 * Copyright (c) 2020-2024 STMicroelectronics.
 * All rights reserved.
```

```
 *
 * This software is licensed under terms that can be found in the
LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
*****************************************************************
*******************
 */
/* Includes */
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
#include <sys/times.h>
/* Variables */
extern int __io_putchar(int ch) __attribute__((weak));
extern int __io_getchar(void) __attribute__((weak));
char *__env[1] = { 0 };
char **environ = __env;
/* Functions */
```

```
void initialise_monitor_handles()
{
}
int _getpid(void)
{
  return 1;
}
int _kill(int pid, int sig)
{
  (void)pid;
  (void)sig;
  errno = EINVAL;
  return -1;
}
void _exit (int status)
{
  _kill(status, -1);
  while (1) {}    /* Make sure we hang here */
}
__attribute__((weak)) int _read(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;
```

```
  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    *ptr++ = __io_getchar();
  }
  return len;
}
__attribute__((weak)) int _write(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;
  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    __io_putchar(*ptr++);
  }
  return len;
}
int _close(int file)
{
  (void)file;
  return -1;
}
int _fstat(int file, struct stat *st)
{
```

```
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
int _isatty(int file)
{
  (void)file;
  return 1;
}
int _lseek(int file, int ptr, int dir)
{
  (void)file;
  (void)ptr;
  (void)dir;
  return 0;
}
int _open(char *path, int flags, ...)
{
  (void)path;
  (void)flags;
  /* Pretend like we always fail */
  return -1;
}
```

```
int _wait(int *status)
{
  (void)status;
  errno = ECHILD;
  return -1;
}
int _unlink(char *name)
{
  (void)name;
  errno = ENOENT;
  return -1;
}
int _times(struct tms *buf)
{
  (void)buf;
  return -1;
}
int _stat(char *file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
```

```
int _link(char *old, char *new)
{
  (void)old;
  (void)new;
  errno = EMLINK;
  return -1;
}
int _fork(void)
{
  errno = EAGAIN;
  return -1;
}
int _execve(char *name, char **argv, char **env)
{
  (void)name;
  (void)argv;
  (void)env;
  errno = ENOMEM;
  return -1;
}
```

27.14 CODE OF SYSMEM.C:

```
/**
```

```
*****************************************************
*******************
 * @file      sysmem.c
 * @author    Generated by STM32CubeIDE
 * @brief     STM32CubeIDE System Memory calls file
 *
 *          For more information about which C functions
 *          need which of these lowlevel functions
 *          please consult the newlib libc manual
*****************************************************
*******************
 * @attention
 *
 * Copyright (c) 2024 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the
LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
*****************************************************
*******************
 */
/* Includes */
```

```
#include <errno.h>
#include <stdint.h>
/**
 * Pointer to the current high watermark of the heap usage
 */
static uint8_t *__sbrk_heap_end = NULL;
/**
 * @brief _sbrk() allocates memory to the newlib heap and is used by malloc
 *        and others from the C library
 *
 * @verbatim
 *
 ############################################################################
 * #  .data  #  .bss  #       newlib heap       #          MSP stack          #
 * #         #        #                         # Reserved by _Min_Stack_Size #
 *
 ############################################################################
 * ^-- RAM start      ^-- _end                              _estack, RAM end --^
 * @endverbatim
 *
 * This implementation starts allocating at the '_end' linker symbol
```

```
 * The '_Min_Stack_Size' linker symbol reserves a memory for the MSP stack
 * The implementation considers '_estack' linker symbol to be RAM end
 * NOTE: If the MSP stack, at any point during execution, grows larger than the
 * reserved size, please increase the '_Min_Stack_Size'.
 *
 * @param incr Memory size
 * @return Pointer to allocated memory
 */
void *_sbrk(ptrdiff_t incr)
{
  extern uint8_t _end; /* Symbol defined in the linker script */
  extern uint8_t _estack; /* Symbol defined in the linker script */
  extern uint32_t _Min_Stack_Size; /* Symbol defined in the linker script */
  const uint32_t stack_limit = (uint32_t)&_estack - (uint32_t)&_Min_Stack_Size;
  const uint8_t *max_heap = (uint8_t *)stack_limit;
  uint8_t *prev_heap_end;
  /* Initialize heap end at first call */
  if (NULL == __sbrk_heap_end)
  {
    __sbrk_heap_end = &_end;
```

```
  }
  /* Protect heap from growing into the reserved MSP stack */
  if (__sbrk_heap_end + incr > max_heap)
  {
    errno = ENOMEM;
    return (void *)-1;
  }
  prev_heap_end = __sbrk_heap_end;
  __sbrk_heap_end += incr;
  return (void *)prev_heap_end;
}
```

27.15 CODE OF SYSTEM_STM32F4XX.C:

```
/**
*****************************************************************
*******************
  * @file    system_stm32f4xx.c
  * @author  MCD Application Team
  * @brief   CMSIS Cortex-M4 Device Peripheral Access Layer System
Source File.
  *
  *   This file provides two functions and one global variable to be called
from
  *   user application:
  *      - SystemInit(): This function is called at startup just after reset and
```

```
  *                 before branch to main program. This call is made inside
  *                 the "startup_stm32f4xx.s" file.
  *
  *     - SystemCoreClock variable: Contains the core clock (HCLK), it
can be used
  *                          by the user application to setup the SysTick
  *                          timer or configure other parameters.
  *
  *     - SystemCoreClockUpdate(): Updates the variable
SystemCoreClock and must
  *                         be called whenever the core clock is changed
  *                         during program execution.
  *
  *
*****************************************************************
*******************
  * @attention
  *
  * Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
```

```
  *
**********************************************************
*******************
  */
/** @addtogroup CMSIS
  * @{
  */
/** @addtogroup stm32f4xx_system
  * @{
  */
/** @addtogroup STM32F4xx_System_Private_Includes
  * @{
  */
#include "stm32f4xx.h"
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    ((uint32_t)25000000) /*!< Default value of
the External oscillator in Hz */
#endif /* HSE_VALUE */
#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000) /*!< Value of the
Internal oscillator in Hz*/
#endif /* HSI_VALUE */
/**
  * @}
```

```
  */
/** @addtogroup STM32F4xx_System_Private_TypesDefinitions
  * @{
  */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Defines
  * @{
  */
/************************* Miscellaneous Configuration ************************/
/*!< Uncomment the following line if you need to use external SRAM or SDRAM as data memory  */
#if defined(STM32F405xx) || defined(STM32F415xx) || defined(STM32F407xx) || defined(STM32F417xx)\
 || defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx) || defined(STM32F412Zx) || defined(STM32F412Vx)
/* #define DATA_IN_ExtSRAM */
#endif /* STM32F40xxx || STM32F41xxx || STM32F42xxx || STM32F43xxx || STM32F469xx || STM32F479xx ||\
        STM32F412Zx || STM32F412Vx */
```

#if defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) || defined(STM32F439xx)\

|| defined(STM32F446xx) || defined(STM32F469xx) || defined(STM32F479xx)

/* #define DATA_IN_ExtSDRAM */

#endif /* STM32F427xx || STM32F437xx || STM32F429xx || STM32F439xx || STM32F446xx || STM32F469xx ||\

STM32F479xx */

/* Note: Following vector table addresses must be defined in line with linker

configuration. */

/*!< Uncomment the following line if you need to relocate the vector table

anywhere in Flash or Sram, else the vector table is kept at the automatic

remap of boot address selected */

/* #define USER_VECT_TAB_ADDRESS */

#if defined(USER_VECT_TAB_ADDRESS)

/*!< Uncomment the following line if you need to relocate your vector Table

in Sram else user remap will be done in Flash. */

/* #define VECT_TAB_SRAM */

#if defined(VECT_TAB_SRAM)

#define VECT_TAB_BASE_ADDRESS SRAM_BASE /*!< Vector Table base address field.

```
                                        This value must be a multiple of 0x200.
*/
#define VECT_TAB_OFFSET        0x00000000U     /*!< Vector Table
base offset field.
                                        This value must be a multiple of 0x200.
*/
#else
#define VECT_TAB_BASE_ADDRESS   FLASH_BASE      /*!<
Vector Table base address field.
                                        This value must be a multiple of 0x200.
*/
#define VECT_TAB_OFFSET        0x00000000U     /*!< Vector Table
base offset field.
                                        This value must be a multiple of 0x200.
*/
#endif /* VECT_TAB_SRAM */
#endif /* USER_VECT_TAB_ADDRESS */
/*****************************************************************
********************/
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Macros
  * @{
  */
```

```
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Variables
  * @{
  */
  /* This variable is updated in three ways:
      1) by calling CMSIS function SystemCoreClockUpdate()
      2) by calling HAL API function HAL_RCC_GetHCLKFreq()
      3) each time HAL_RCC_ClockConfig() is called to configure the
system clock frequency
         Note: If you use this function to configure the system clock; then
there
               is no need to call the 2 first functions listed above, since
SystemCoreClock
               variable is updated automatically.
  */
uint32_t SystemCoreClock = 16000000;
const uint8_t AHBPrescTable[16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7,
8, 9};
const uint8_t APBPrescTable[8]  = {0, 0, 0, 0, 1, 2, 3, 4};
/**
  * @}
  */
```

```
/** @addtogroup STM32F4xx_System_Private_FunctionPrototypes
  * @{
  */
#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  static void SystemInit_ExtMemCtl(void);
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Functions
  * @{
  */
/**
  * @brief  Setup the microcontroller system
  *         Initialize the FPU setting, vector table location and External memory
  *         configuration.
  * @param  None
  * @retval None
  */
void SystemInit(void)
{
  /* FPU settings ------------------------------------------------------------*/
```

```
  #if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
    SCB->CPACR |= ((3UL << 10*2)|(3UL << 11*2));  /* set CP10 and
CP11 Full Access */
  #endif
#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  SystemInit_ExtMemCtl();
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */
  /* Configure the Vector Table location -----------------------------------*/
#if defined(USER_VECT_TAB_ADDRESS)
  SCB->VTOR = VECT_TAB_BASE_ADDRESS |
VECT_TAB_OFFSET; /* Vector Table Relocation in Internal SRAM */
#endif /* USER_VECT_TAB_ADDRESS */
}
/**
   * @brief  Update SystemCoreClock variable according to Clock
Register Values.
  *         The SystemCoreClock variable contains the core clock (HCLK),
it can
  *         be used by the user application to setup the SysTick timer or
configure
  *         other parameters.
  *
  * @note   Each time the core clock (HCLK) changes, this function
must be called
```

```
  *         to update SystemCoreClock variable value. Otherwise, any configuration
  *         based on this variable will be incorrect.
  *
  * @note   - The system frequency computed by this function is not the real
  *           frequency in the chip. It is calculated based on the predefined
  *           constant and the selected clock source:
  *
  *           - If SYSCLK source is HSI, SystemCoreClock will contain the HSI_VALUE(*)
  *
  *           - If SYSCLK source is HSE, SystemCoreClock will contain the HSE_VALUE(**)
  *
  *           - If SYSCLK source is PLL, SystemCoreClock will contain the HSE_VALUE(**)
  *             or HSI_VALUE(*) multiplied/divided by the PLL factors.
  *
  *         (*) HSI_VALUE is a constant defined in stm32f4xx_hal_conf.h file (default value
  *             16 MHz) but the real value may vary depending on the variations
  *             in voltage and temperature.
  *
```

```
  *        (**) HSE_VALUE is a constant defined in
stm32f4xx_hal_conf.h file (its value
  *             depends on the application requirements), user has to ensure
that HSE_VALUE
  *             is same as the real frequency of the crystal used. Otherwise,
this function
  *             may have wrong result.
  *
  *        - The result of this function could be not correct when using
fractional
  *          value for HSE crystal.
  *
  * @param  None
  * @retval None
  */
void SystemCoreClockUpdate(void)
{
  uint32_t tmp = 0, pllvco = 0, pllp = 2, pllsource = 0, pllm = 2;
  /* Get SYSCLK source -------------------------------------------------------
*/
  tmp = RCC->CFGR & RCC_CFGR_SWS;
  switch (tmp)
  {
    case 0x00:  /* HSI used as system clock source */
      SystemCoreClock = HSI_VALUE;
```

```
    break;
  case 0x04:  /* HSE used as system clock source */
    SystemCoreClock = HSE_VALUE;
    break;
  case 0x08:  /* PLL used as system clock source */
    /* PLL_VCO = (HSE_VALUE or HSI_VALUE / PLL_M) * PLL_N
       SYSCLK = PLL_VCO / PLL_P
       */
    pllsource = (RCC->PLLCFGR & RCC_PLLCFGR_PLLSRC) >>
22;
    pllm = RCC->PLLCFGR & RCC_PLLCFGR_PLLM;
    if (pllsource != 0)
    {
      /* HSE used as PLL clock source */
      pllvco = (HSE_VALUE / pllm) * ((RCC->PLLCFGR &
RCC_PLLCFGR_PLLN) >> 6);
    }
    else
    {
      /* HSI used as PLL clock source */
      pllvco = (HSI_VALUE / pllm) * ((RCC->PLLCFGR &
RCC_PLLCFGR_PLLN) >> 6);
    }
```

```
    pllp = (((RCC->PLLCFGR & RCC_PLLCFGR_PLLP) >>16) + 1 )
*2;
    SystemCoreClock = pllvco/pllp;
    break;
   default:
    SystemCoreClock = HSI_VALUE;
    break;
  }
  /* Compute HCLK frequency ----------------------------------------------
-*/
  /* Get HCLK prescaler */
  tmp = AHBPrescTable[((RCC->CFGR & RCC_CFGR_HPRE) >> 4)];
  /* HCLK frequency */
  SystemCoreClock >>= tmp;
}
#if defined (DATA_IN_ExtSRAM) && defined
(DATA_IN_ExtSDRAM)
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
/**
  * @brief  Setup the external memory controller.
  *         Called in startup_stm32f4xx.s before jump to main.
  *         This function configures the external memories
(SRAM/SDRAM)
```

```
  *        This SRAM/SDRAM will be used as program data memory
(including heap and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
  register __IO uint32_t index;
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and
GPIOI interface clock */
  RCC->AHB1ENR |= 0x000001F8;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x00CCC0CC;
  GPIOD->AFR[1]  = 0xCCCCCCCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xAAAA0A8A;
  /* Configure PDx pins speed to 100 MHz */
  GPIOD->OSPEEDR = 0xFFFF0FCF;
  /* Configure PDx pins Output type to push-pull */
  GPIOD->OTYPER  = 0x00000000;
```

```
/* No pull-up, pull-down for PDx pins */
GPIOD->PUPDR   = 0x00000000;
/* Connect PEx pins to FMC Alternate function */
GPIOE->AFR[0]  = 0xC00CC0CC;
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA828A;
/* Configure PEx pins speed to 100 MHz */
GPIOE->OSPEEDR = 0xFFFFC3CF;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0xCCCCCCCC;
GPIOF->AFR[1]  = 0xCCCCCCCC;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA800AAA;
/* Configure PFx pins speed to 50 MHz */
GPIOF->OSPEEDR = 0xAA800AAA;
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
```

```
GPIOF->PUPDR   = 0x00000000;
/* Connect PGx pins to FMC Alternate function */
GPIOG->AFR[0]  = 0xCCCCCCCC;
GPIOG->AFR[1]  = 0xCCCCCCCC;
/* Configure PGx pins in Alternate function mode */
GPIOG->MODER   = 0xAAAAAAAA;
/* Configure PGx pins speed to 50 MHz */
GPIOG->OSPEEDR = 0xAAAAAAAA;
/* Configure PGx pins Output type to push-pull */
GPIOG->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PGx pins */
GPIOG->PUPDR   = 0x00000000;
/* Connect PHx pins to FMC Alternate function */
GPIOH->AFR[0]  = 0x00C0CC00;
GPIOH->AFR[1]  = 0xCCCCCCCC;
/* Configure PHx pins in Alternate function mode */
GPIOH->MODER   = 0xAAAA08A0;
/* Configure PHx pins speed to 50 MHz */
GPIOH->OSPEEDR = 0xAAAA08A0;
/* Configure PHx pins Output type to push-pull */
GPIOH->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PHx pins */
GPIOH->PUPDR   = 0x00000000;
```

```
  /* Connect PIx pins to FMC Alternate function */
  GPIOI->AFR[0]  = 0xCCCCCCCC;
  GPIOI->AFR[1]  = 0x00000CC0;
  /* Configure PIx pins in Alternate function mode */
  GPIOI->MODER   = 0x0028AAAA;
  /* Configure PIx pins speed to 50 MHz */
  GPIOI->OSPEEDR = 0x0028AAAA;
  /* Configure PIx pins Output type to push-pull */
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;
/*-- FMC Configuration ------------------------------------------------------*/
  /* Enable the FMC interface clock */
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
  FMC_Bank5_6->SDTR[0] = 0x01115351;
  /* SDRAM initialization sequence */
  /* Clock enable command */
  FMC_Bank5_6->SDCMR = 0x00000011;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  while((tmpreg != 0) && (timeout-- > 0))
```

```
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* Delay */
for (index = 0; index<1000; index++);
/* PALL command */
FMC_Bank5_6->SDCMR = 0x00000012;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* Auto refresh command */
FMC_Bank5_6->SDCMR = 0x00000073;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* MRD register program */
FMC_Bank5_6->SDCMR = 0x00046014;
```

```
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Set refresh count */
  tmpreg = FMC_Bank5_6->SDRTR;
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
  /* Disable write protection */
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
```

```
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F469xx || STM32F479xx */
  (void)(tmp);
}
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F469xx || STM32F479xx */
#elif defined (DATA_IN_ExtSRAM) || defined
(DATA_IN_ExtSDRAM)
/**
  * @brief  Setup the external memory controller.
  *         Called in startup_stm32f4xx.s before jump to main.
  *         This function configures the external memories
(SRAM/SDRAM)
  *         This SRAM/SDRAM will be used as program data memory
(including heap and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F446xx) || defined(STM32F469xx) ||
defined(STM32F479xx)
```

```
#if defined (DATA_IN_ExtSDRAM)
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
  register __IO uint32_t index;
#if defined(STM32F446xx)
  /* Enable GPIOA, GPIOC, GPIOD, GPIOE, GPIOF, GPIOG interface
     clock */
  RCC->AHB1ENR |= 0x0000007D;
#else
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and
GPIOI interface
     clock */
  RCC->AHB1ENR |= 0x000001F8;
#endif /* STM32F446xx */
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);
#if defined(STM32F446xx)
  /* Connect PAx pins to FMC Alternate function */
  GPIOA->AFR[0]  |= 0xC0000000;
  GPIOA->AFR[1]  |= 0x00000000;
  /* Configure PDx pins in Alternate function mode */
  GPIOA->MODER   |= 0x00008000;
  /* Configure PDx pins speed to 50 MHz */
  GPIOA->OSPEEDR |= 0x00008000;
```

```
  /* Configure PDx pins Output type to push-pull */
  GPIOA->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOA->PUPDR   |= 0x00000000;
  /* Connect PCx pins to FMC Alternate function */
  GPIOC->AFR[0]  |= 0x00CC0000;
  GPIOC->AFR[1]  |= 0x00000000;
  /* Configure PDx pins in Alternate function mode */
  GPIOC->MODER   |= 0x00000A00;
  /* Configure PDx pins speed to 50 MHz */
  GPIOC->OSPEEDR |= 0x00000A00;
  /* Configure PDx pins Output type to push-pull */
  GPIOC->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOC->PUPDR   |= 0x00000000;
#endif /* STM32F446xx */
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x000000CC;
  GPIOD->AFR[1]  = 0xCC000CCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xA02A000A;
  /* Configure PDx pins speed to 50 MHz */
  GPIOD->OSPEEDR = 0xA02A000A;
```

```
/* Configure PDx pins Output type to push-pull */
GPIOD->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PDx pins */
GPIOD->PUPDR   = 0x00000000;
/* Connect PEx pins to FMC Alternate function */
GPIOE->AFR[0]  = 0xC00000CC;
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA800A;
/* Configure PEx pins speed to 50 MHz */
GPIOE->OSPEEDR = 0xAAAA800A;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0xCCCCCCCC;
GPIOF->AFR[1]  = 0xCCCCCCCC;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA800AAA;
/* Configure PFx pins speed to 50 MHz */
GPIOF->OSPEEDR = 0xAA800AAA;
/* Configure PFx pins Output type to push-pull */
```

```
  GPIOF->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PFx pins */
  GPIOF->PUPDR   = 0x00000000;
  /* Connect PGx pins to FMC Alternate function */
  GPIOG->AFR[0]  = 0xCCCCCCCC;
  GPIOG->AFR[1]  = 0xCCCCCCCC;
  /* Configure PGx pins in Alternate function mode */
  GPIOG->MODER   = 0xAAAAAAAA;
  /* Configure PGx pins speed to 50 MHz */
  GPIOG->OSPEEDR = 0xAAAAAAAA;
  /* Configure PGx pins Output type to push-pull */
  GPIOG->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
  /* Connect PHx pins to FMC Alternate function */
  GPIOH->AFR[0]  = 0x00C0CC00;
  GPIOH->AFR[1]  = 0xCCCCCCCC;
  /* Configure PHx pins in Alternate function mode */
  GPIOH->MODER   = 0xAAAA08A0;
  /* Configure PHx pins speed to 50 MHz */
```

```
  GPIOH->OSPEEDR = 0xAAAA08A0;
  /* Configure PHx pins Output type to push-pull */
  GPIOH->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PHx pins */
  GPIOH->PUPDR   = 0x00000000;
  /* Connect PIx pins to FMC Alternate function */
  GPIOI->AFR[0]  = 0xCCCCCCCC;
  GPIOI->AFR[1]  = 0x00000CC0;
  /* Configure PIx pins in Alternate function mode */
  GPIOI->MODER   = 0x0028AAAA;
  /* Configure PIx pins speed to 50 MHz */
  GPIOI->OSPEEDR = 0x0028AAAA;
  /* Configure PIx pins Output type to push-pull */
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F469xx || STM32F479xx */
/*-- FMC Configuration -----------------------------------------------------*/
  /* Enable the FMC interface clock */
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
```

```
  /* Configure and enable SDRAM bank1 */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCR[0] = 0x00001954;
#else
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
#endif /* STM32F446xx */
  FMC_Bank5_6->SDTR[0] = 0x01115351;
  /* SDRAM initialization sequence */
  /* Clock enable command */
  FMC_Bank5_6->SDCMR = 0x00000011;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Delay */
  for (index = 0; index<1000; index++);
  /* PALL command */
  FMC_Bank5_6->SDCMR = 0x00000012;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
```

```
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Auto refresh command */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x000000F3;
#else
  FMC_Bank5_6->SDCMR = 0x00000073;
#endif /* STM32F446xx */
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* MRD register program */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x00044014;
#else
  FMC_Bank5_6->SDCMR = 0x00046014;
#endif /* STM32F446xx */
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
```

```
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Set refresh count */
  tmpreg = FMC_Bank5_6->SDRTR;
#if defined(STM32F446xx)
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000050C<<1));
#else
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
#endif /* STM32F446xx */
  /* Disable write protection */
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#endif /* DATA_IN_ExtSDRAM */
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F446xx || STM32F469xx || STM32F479xx */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx) || defined(STM32F417xx)\
 || defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx) ||
defined(STM32F412Zx) || defined(STM32F412Vx)
#if defined(DATA_IN_ExtSRAM)
/*-- GPIOs Configuration -----------------------------------------------------*/
```

```
 /* Enable GPIOD, GPIOE, GPIOF and GPIOG interface clock */
RCC->AHB1ENR   |= 0x00000078;
/* Delay after an RCC peripheral clock enabling */
tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIODEN);
/* Connect PDx pins to FMC Alternate function */
GPIOD->AFR[0]  = 0x00CCC0CC;
GPIOD->AFR[1]  = 0xCCCCCCCC;
/* Configure PDx pins in Alternate function mode */
GPIOD->MODER   = 0xAAAA0A8A;
/* Configure PDx pins speed to 100 MHz */
GPIOD->OSPEEDR = 0xFFFF0FCF;
/* Configure PDx pins Output type to push-pull */
GPIOD->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PDx pins */
GPIOD->PUPDR   = 0x00000000;
/* Connect PEx pins to FMC Alternate function */
GPIOE->AFR[0]  = 0xC00CC0CC;
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA828A;
/* Configure PEx pins speed to 100 MHz */
GPIOE->OSPEEDR = 0xFFFFC3CF;
/* Configure PEx pins Output type to push-pull */
```

```
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0x00CCCCCC;
GPIOF->AFR[1]  = 0xCCCC0000;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA000AAA;
/* Configure PFx pins speed to 100 MHz */
GPIOF->OSPEEDR = 0xFF000FFF;
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
GPIOF->PUPDR   = 0x00000000;
/* Connect PGx pins to FMC Alternate function */
GPIOG->AFR[0]  = 0x00CCCCCC;
GPIOG->AFR[1]  = 0x000000C0;
/* Configure PGx pins in Alternate function mode */
GPIOG->MODER   = 0x00085AAA;
/* Configure PGx pins speed to 100 MHz */
GPIOG->OSPEEDR = 0x000CAFFF;
/* Configure PGx pins Output type to push-pull */
GPIOG->OTYPER  = 0x00000000;
```

```
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;
/*-- FMC/FSMC Configuration ----------------------------------------------
-*/
  /* Enable the FMC/FSMC interface clock */
  RCC->AHB3ENR         |= 0x00000001;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F469xx || STM32F479xx */
```

```
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx)|| defined(STM32F417xx)\
   || defined(STM32F412Zx) || defined(STM32F412Vx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FSMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FSMC_Bank1->BTCR[2]  = 0x00001011;
  FSMC_Bank1->BTCR[3]  = 0x00000201;
  FSMC_Bank1E->BWTR[2] = 0x0FFFFFFF;
#endif /* STM32F405xx || STM32F415xx || STM32F407xx ||
STM32F417xx || STM32F412Zx || STM32F412Vx */
#endif /* DATA_IN_ExtSRAM */
#endif /* STM32F405xx || STM32F415xx || STM32F407xx ||
STM32F417xx || STM32F427xx || STM32F437xx ||\
         STM32F429xx || STM32F439xx || STM32F469xx ||
STM32F479xx || STM32F412Zx || STM32F412Vx  */
  (void)(tmp);
}
#endif /* DATA_IN_ExtSRAM && DATA_IN_ExtSDRAM */
/**
  * @}
  */
/**
  * @}
```

```
  */
/**
  * @}
  */
```

CHAPTER#28 CODING A HARDWARE TIMER DRIVER FOR GENERATING PRECISE INTERRUPTS

28.1CODING A HARDWARE TIMER DRIVER FORGENERATING PRECISE INTERRUPTS:

Generating precise interrupts using a hardware timer involves configuring the timer to trigger interrupts at specific intervals. This capability is crucial for time-sensitive applications such as real-time systems, task scheduling, and periodic sampling. Below is a guide to coding a hardware timer driver for generating precise interrupts in assembly language for an ARM Cortex-M based microcontroller, such as those in the STM32 family.

28.2 UNDERSTANDING TIMER INTERRUPTS:

Timers can be configured to generate interrupts based on several events, such as:

1: Reaching a certain count value (compare match).

2: Overflowing the counter (update event).

3: Input capture events

For generating periodic interrupts, we typically use the update event, which occurs when the counter overflows and resets.

28.3 SETTING UP TIMER:

28.3.1 ENABLING THE TIMER CLOCK:

Enable the clock for the timer peripheral in the RCC (Reset and Clock Control) register.

28.3.2 CONFIGURING THE TIMER:

Set the timer's prescaler and auto-reload value to determine the interrupt interval.

28.3.2.1 CALCULATING THE INTERVAL:

The timer generates an update event when the counter reaches the value in the ARR (auto-reload register) and then resets. The interval between interrupts (in seconds) is calculated as:

INTERVAL=(ARR+1) *(PSC+1)/TIMER CLOCK

28.3.3 ENABLING THE TIMER:

Enable the timer interrupt by setting the appropriate bits in the DIER (DMA/Interrupt Enable Register).

28.4 START THE TIMER:

Start the timer by setting the CEN (Counter Enable) bit in the CR1 register.

28.5 IMPLEMENTING THE INTERRUPT SERVICE ROUTINE (ISR):

Write an ISR to handle the timer interrupt. The ISR should clear the interrupt flag and perform the desired action.

28.6CONFIGURING NVIC FOR TIMER INTERRUPTS:

Enable the timer interrupt in the NVIC (Nested Vectored Interrupt Controller).

28.7 OPTIMIZATIONS AND BEST PRACTICES:

28.7.1 CRITICAL SECTIONS:

Disable interrupts briefly while updating shared resources to avoid race conditions.

28.7.2 MINIMIZE ISR DURATION:

Keep the ISR short and efficient to ensure timely handling of interrupts and avoid missing subsequent events.

28.7.3 INTERRUPT PRIORITIES:

Set appropriate priorities for the timer interrupt if multiple interrupts are used, especially in real-time applications.

28.8 TESTING AND VERIFICATION:

After implementing the timer driver, test the system thoroughly to ensure that interrupts are generated at the expected intervals and that the ISR behaves as intended.

28.9 CODE OF MAIN.H:

```
/* USER CODE BEGIN Header */
/**
**********************************************************
*******************
```

```
  * @file           : main.h
  * @brief          : Header for main.c file.
  *                   This file contains the common defines of the application.
**********************************************************************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H
#ifdef __cplusplus
extern "C" {
```

```
#endif
/* Includes ------------------------------------------------------------------*/
#include "stm32f4xx_hal.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */
/* USER CODE END EC */
/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */
/* USER CODE END EM */
/* Exported functions prototypes ---------------------------------------------*/
void Error_Handler(void);
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
/* Private defines -----------------------------------------------------------*/
#define B1_Pin GPIO_PIN_13
#define B1_GPIO_Port GPIOC
#define USART_TX_Pin GPIO_PIN_2
```

```
#define USART_TX_GPIO_Port GPIOA
#define USART_RX_Pin GPIO_PIN_3
#define USART_RX_GPIO_Port GPIOA
#define LD2_Pin GPIO_PIN_5
#define LD2_GPIO_Port GPIOA
#define TMS_Pin GPIO_PIN_13
#define TMS_GPIO_Port GPIOA
#define TCK_Pin GPIO_PIN_14
#define TCK_GPIO_Port GPIOA
#define SWO_Pin GPIO_PIN_3
#define SWO_GPIO_Port GPIOB
/* USER CODE BEGIN Private defines */
/* USER CODE END Private defines */
#ifdef __cplusplus
}
#endif
#endif /* __MAIN_H */
```

28.10 CODE OF STM32F4XX_HAL_CONF.H:

```
/* USER CODE BEGIN Header */
/**
*****************************************************************
*******************
  * @file    stm32f4xx_hal_conf_template.h
```

```
  * @author  MCD Application Team
  * @brief   HAL configuration template file.
  *          This file should be copied to the application folder and renamed
  *          to stm32f4xx_hal_conf.h.
******************************************************************************
  * @attention
  *
  * Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
******************************************************************************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_HAL_CONF_H
#define __STM32F4xx_HAL_CONF_H
#ifdef __cplusplus
 extern "C" {
```

```
#endif
/* Exported types ------------------------------------------------------------*/
/* Exported constants --------------------------------------------------------*/
/* ############################ Module Selection
################################## */
/**
  * @brief This is the list of modules to be used in the HAL driver
  */
#define HAL_MODULE_ENABLED
  /* #define HAL_CRYP_MODULE_ENABLED */
/* #define HAL_ADC_MODULE_ENABLED */
/* #define HAL_CAN_MODULE_ENABLED */
/* #define HAL_CRC_MODULE_ENABLED */
/* #define HAL_CAN_LEGACY_MODULE_ENABLED */
/* #define HAL_DAC_MODULE_ENABLED */
/* #define HAL_DCMI_MODULE_ENABLED */
/* #define HAL_DMA2D_MODULE_ENABLED */
/* #define HAL_ETH_MODULE_ENABLED */
/* #define HAL_ETH_LEGACY_MODULE_ENABLED */
/* #define HAL_NAND_MODULE_ENABLED */
/* #define HAL_NOR_MODULE_ENABLED */
/* #define HAL_PCCARD_MODULE_ENABLED */
/* #define HAL_SRAM_MODULE_ENABLED */
```

```
/* #define HAL_SDRAM_MODULE_ENABLED */
/* #define HAL_HASH_MODULE_ENABLED */
/* #define HAL_I2C_MODULE_ENABLED */
/* #define HAL_I2S_MODULE_ENABLED */
/* #define HAL_IWDG_MODULE_ENABLED */
/* #define HAL_LTDC_MODULE_ENABLED */
/* #define HAL_RNG_MODULE_ENABLED */
/* #define HAL_RTC_MODULE_ENABLED */
/* #define HAL_SAI_MODULE_ENABLED */
/* #define HAL_SD_MODULE_ENABLED */
/* #define HAL_MMC_MODULE_ENABLED */
/* #define HAL_SPI_MODULE_ENABLED */
/* #define HAL_TIM_MODULE_ENABLED */
#define HAL_UART_MODULE_ENABLED
/* #define HAL_USART_MODULE_ENABLED */
/* #define HAL_IRDA_MODULE_ENABLED */
/* #define HAL_SMARTCARD_MODULE_ENABLED */
/* #define HAL_SMBUS_MODULE_ENABLED */
/* #define HAL_WWDG_MODULE_ENABLED */
/* #define HAL_PCD_MODULE_ENABLED */
/* #define HAL_HCD_MODULE_ENABLED */
/* #define HAL_DSI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
```

```
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_CEC_MODULE_ENABLED */
/* #define HAL_FMPI2C_MODULE_ENABLED */
/* #define HAL_FMPSMBUS_MODULE_ENABLED */
/* #define HAL_SPDIFRX_MODULE_ENABLED */
/* #define HAL_DFSDM_MODULE_ENABLED */
/* #define HAL_LPTIM_MODULE_ENABLED */
#define HAL_GPIO_MODULE_ENABLED
#define HAL_EXTI_MODULE_ENABLED
#define HAL_DMA_MODULE_ENABLED
#define HAL_RCC_MODULE_ENABLED
#define HAL_FLASH_MODULE_ENABLED
#define HAL_PWR_MODULE_ENABLED
#define HAL_CORTEX_MODULE_ENABLED
/* ############################ HSE/HSI Values adaptation
###################### */
/**
  * @brief Adjust the value of External High Speed oscillator (HSE)
used in your application.
  *        This value is used by the RCC HAL module to compute the
system frequency
  *        (when HSE is used as system clock source, directly or through
the PLL).
  */
```

```
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    8000000U /*!< Value of the External
oscillator in Hz */
#endif /* HSE_VALUE */
#if !defined  (HSE_STARTUP_TIMEOUT)
  #define HSE_STARTUP_TIMEOUT    100U   /*!< Time out for
HSE start up, in ms */
#endif /* HSE_STARTUP_TIMEOUT */
/**
  * @brief Internal High Speed oscillator (HSI) value.
  *        This value is used by the RCC HAL module to compute the
system frequency
  *        (when HSI is used as system clock source, directly or through the
PLL).
  */
#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000U) /*!< Value of the
Internal oscillator in Hz*/
#endif /* HSI_VALUE */
/**
  * @brief Internal Low Speed oscillator (LSI) value.
  */
#if !defined  (LSI_VALUE)
 #define LSI_VALUE  32000U       /*!< LSI Typical Value in Hz*/
```

```
#endif /* LSI_VALUE */                    /*!< Value of the Internal Low Speed oscillator in Hz
                                The real value may vary depending on the variations
                                in voltage and temperature.*/
/**
  * @brief External Low Speed oscillator (LSE) value.
  */
#if !defined  (LSE_VALUE)
 #define LSE_VALUE  32768U    /*!< Value of the External Low Speed oscillator in Hz */
#endif /* LSE_VALUE */
#if !defined  (LSE_STARTUP_TIMEOUT)
  #define LSE_STARTUP_TIMEOUT    5000U   /*!< Time out for LSE start up, in ms */
#endif /* LSE_STARTUP_TIMEOUT */
/**
  * @brief External clock source for I2S peripheral
  *       This value is used by the I2S HAL module to compute the I2S clock source
  *       frequency, this source is inserted directly through I2S_CKIN pad.
  */
#if !defined  (EXTERNAL_CLOCK_VALUE)
```

```
  #define EXTERNAL_CLOCK_VALUE    12288000U /*!< Value of
the External audio frequency in Hz*/

#endif /* EXTERNAL_CLOCK_VALUE */

/* Tip: To avoid modifying this file each time you need to use different
HSE,

   ===  you can define the HSE value in your toolchain compiler
preprocessor. */

/* ############################ System Configuration
######################### */

/**

  * @brief This is the HAL system configuration section

  */

#define  VDD_VALUE                    3300U /*!< Value of VDD in mv */

#define  TICK_INT_PRIORITY            0U   /*!< tick interrupt
priority */

#define  USE_RTOS                     0U

#define  PREFETCH_ENABLE              1U

#define  INSTRUCTION_CACHE_ENABLE     1U

#define  DATA_CACHE_ENABLE            1U

#define  USE_HAL_ADC_REGISTER_CALLBACKS        0U /*
ADC register callback disabled       */

#define  USE_HAL_CAN_REGISTER_CALLBACKS        0U /*
CAN register callback disabled       */

#define  USE_HAL_CEC_REGISTER_CALLBACKS        0U /*
CEC register callback disabled       */
```

#define USE_HAL_CRYP_REGISTER_CALLBACKS 0U /* CRYP register callback disabled */

#define USE_HAL_DAC_REGISTER_CALLBACKS 0U /* DAC register callback disabled */

#define USE_HAL_DCMI_REGISTER_CALLBACKS 0U /* DCMI register callback disabled */

#define USE_HAL_DFSDM_REGISTER_CALLBACKS 0U /* DFSDM register callback disabled */

#define USE_HAL_DMA2D_REGISTER_CALLBACKS 0U /* DMA2D register callback disabled */

#define USE_HAL_DSI_REGISTER_CALLBACKS 0U /* DSI register callback disabled */

#define USE_HAL_ETH_REGISTER_CALLBACKS 0U /* ETH register callback disabled */

#define USE_HAL_HASH_REGISTER_CALLBACKS 0U /* HASH register callback disabled */

#define USE_HAL_HCD_REGISTER_CALLBACKS 0U /* HCD register callback disabled */

#define USE_HAL_I2C_REGISTER_CALLBACKS 0U /* I2C register callback disabled */

#define USE_HAL_FMPI2C_REGISTER_CALLBACKS 0U /* FMPI2C register callback disabled */

#define USE_HAL_FMPSMBUS_REGISTER_CALLBACKS 0U /* FMPSMBUS register callback disabled */

#define USE_HAL_I2S_REGISTER_CALLBACKS 0U /* I2S register callback disabled */

#define USE_HAL_IRDA_REGISTER_CALLBACKS 0U /* IRDA register callback disabled */

#define USE_HAL_LPTIM_REGISTER_CALLBACKS 0U /* LPTIM register callback disabled */

#define USE_HAL_LTDC_REGISTER_CALLBACKS 0U /* LTDC register callback disabled */

#define USE_HAL_MMC_REGISTER_CALLBACKS 0U /* MMC register callback disabled */

#define USE_HAL_NAND_REGISTER_CALLBACKS 0U /* NAND register callback disabled */

#define USE_HAL_NOR_REGISTER_CALLBACKS 0U /* NOR register callback disabled */

#define USE_HAL_PCCARD_REGISTER_CALLBACKS 0U /* PCCARD register callback disabled */

#define USE_HAL_PCD_REGISTER_CALLBACKS 0U /* PCD register callback disabled */

#define USE_HAL_QSPI_REGISTER_CALLBACKS 0U /* QSPI register callback disabled */

#define USE_HAL_RNG_REGISTER_CALLBACKS 0U /* RNG register callback disabled */

#define USE_HAL_RTC_REGISTER_CALLBACKS 0U /* RTC register callback disabled */

#define USE_HAL_SAI_REGISTER_CALLBACKS 0U /* SAI register callback disabled */

#define USE_HAL_SD_REGISTER_CALLBACKS 0U /* SD register callback disabled */

#define USE_HAL_SMARTCARD_REGISTER_CALLBACKS 0U /* SMARTCARD register callback disabled */

#define USE_HAL_SDRAM_REGISTER_CALLBACKS 0U /* SDRAM register callback disabled */

#define USE_HAL_SRAM_REGISTER_CALLBACKS 0U /* SRAM register callback disabled */

#define USE_HAL_SPDIFRX_REGISTER_CALLBACKS 0U /* SPDIFRX register callback disabled */

#define USE_HAL_SMBUS_REGISTER_CALLBACKS 0U /* SMBUS register callback disabled */

#define USE_HAL_SPI_REGISTER_CALLBACKS 0U /* SPI register callback disabled */

#define USE_HAL_TIM_REGISTER_CALLBACKS 0U /* TIM register callback disabled */

#define USE_HAL_UART_REGISTER_CALLBACKS 0U /* UART register callback disabled */

#define USE_HAL_USART_REGISTER_CALLBACKS 0U /* USART register callback disabled */

#define USE_HAL_WWDG_REGISTER_CALLBACKS 0U /* WWDG register callback disabled */

/* ########################### Assert Selection ################################ */

/**

* @brief Uncomment the line below to expanse the "assert_param" macro in the

* HAL drivers code

*/

/* #define USE_FULL_ASSERT 1U */

/* ################### Ethernet peripheral configuration ###################### */

/* Section 1 : Ethernet peripheral configuration */

/* MAC ADDRESS: MAC_ADDR0:MAC_ADDR1:MAC_ADDR2:MAC_ADDR3:MAC_ADDR4:MAC_ADDR5 */

#define MAC_ADDR0 2U

#define MAC_ADDR1 0U

#define MAC_ADDR2 0U

#define MAC_ADDR3 0U

#define MAC_ADDR4 0U

#define MAC_ADDR5 0U

/* Definition of the Ethernet driver buffers size and count */

#define ETH_RX_BUF_SIZE ETH_MAX_PACKET_SIZE /* buffer size for receive */

#define ETH_TX_BUF_SIZE ETH_MAX_PACKET_SIZE /* buffer size for transmit */

#define ETH_RXBUFNB 4U /* 4 Rx buffers of size ETH_RX_BUF_SIZE */

#define ETH_TXBUFNB 4U /* 4 Tx buffers of size ETH_TX_BUF_SIZE */

/* Section 2: PHY configuration section */

/* DP83848_PHY_ADDRESS Address*/

#define DP83848_PHY_ADDRESS

/* PHY Reset delay these values are based on a 1 ms Systick interrupt*/

#define PHY_RESET_DELAY 0x000000FFU

/* PHY Configuration delay */

#define PHY_CONFIG_DELAY 0x00000FFFU

#define PHY_READ_TO 0x0000FFFFU

#define PHY_WRITE_TO 0x0000FFFFU

/* Section 3: Common PHY Registers */

#define PHY_BCR ((uint16_t)0x0000U) /*!< Transceiver Basic Control Register */

#define PHY_BSR ((uint16_t)0x0001U) /*!< Transceiver Basic Status Register */

#define PHY_RESET ((uint16_t)0x8000U) /*!< PHY Reset */

#define PHY_LOOPBACK ((uint16_t)0x4000U) /*!< Select loop-back mode */

#define PHY_FULLDUPLEX_100M ((uint16_t)0x2100U) /*!< Set the full-duplex mode at 100 Mb/s */

#define PHY_HALFDUPLEX_100M ((uint16_t)0x2000U) /*!< Set the half-duplex mode at 100 Mb/s */

#define PHY_FULLDUPLEX_10M ((uint16_t)0x0100U) /*!< Set the full-duplex mode at 10 Mb/s */

#define PHY_HALFDUPLEX_10M ((uint16_t)0x0000U) /*!< Set the half-duplex mode at 10 Mb/s */

#define PHY_AUTONEGOTIATION ((uint16_t)0x1000U) /*!< Enable auto-negotiation function */

#define PHY_RESTART_AUTONEGOTIATION ((uint16_t)0x0200U) /*!< Restart auto-negotiation function */

#define PHY_POWERDOWN ((uint16_t)0x0800U) /*!< Select the power down mode */

#define PHY_ISOLATE ((uint16_t)0x0400U) /*!< Isolate PHY from MII */

#define PHY_AUTONEGO_COMPLETE ((uint16_t)0x0020U) /*!< Auto-Negotiation process completed */

#define PHY_LINKED_STATUS ((uint16_t)0x0004U) /*!< Valid link established */

#define PHY_JABBER_DETECTION ((uint16_t)0x0002U) /*!< Jabber condition detected */

/* Section 4: Extended PHY Registers */

#define PHY_SR ((uint16_t)) /*!< PHY status register Offset */

#define PHY_SPEED_STATUS ((uint16_t)) /*!< PHY Speed mask */

#define PHY_DUPLEX_STATUS ((uint16_t)) /*!< PHY Duplex mask */

/* ################### SPI peripheral configuration ############################ */

/* CRC FEATURE: Use to activate CRC feature inside HAL SPI Driver

* Activated: CRC code is present inside driver

* Deactivated: CRC code cleaned from driver

*/

#define USE_SPI_CRC 0U

```
/* Includes ------------------------------------------------------------------*/
/**
  * @brief Include module's header file
  */
#ifdef HAL_RCC_MODULE_ENABLED
  #include "stm32f4xx_hal_rcc.h"
#endif /* HAL_RCC_MODULE_ENABLED */
#ifdef HAL_GPIO_MODULE_ENABLED
  #include "stm32f4xx_hal_gpio.h"
#endif /* HAL_GPIO_MODULE_ENABLED */
#ifdef HAL_EXTI_MODULE_ENABLED
  #include "stm32f4xx_hal_exti.h"
#endif /* HAL_EXTI_MODULE_ENABLED */
#ifdef HAL_DMA_MODULE_ENABLED
  #include "stm32f4xx_hal_dma.h"
#endif /* HAL_DMA_MODULE_ENABLED */
#ifdef HAL_CORTEX_MODULE_ENABLED
  #include "stm32f4xx_hal_cortex.h"
#endif /* HAL_CORTEX_MODULE_ENABLED */
#ifdef HAL_ADC_MODULE_ENABLED
  #include "stm32f4xx_hal_adc.h"
#endif /* HAL_ADC_MODULE_ENABLED */
#ifdef HAL_CAN_MODULE_ENABLED
```

```
  #include "stm32f4xx_hal_can.h"
#endif /* HAL_CAN_MODULE_ENABLED */
#ifdef HAL_CAN_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_can_legacy.h"
#endif /* HAL_CAN_LEGACY_MODULE_ENABLED */
#ifdef HAL_CRC_MODULE_ENABLED
  #include "stm32f4xx_hal_crc.h"
#endif /* HAL_CRC_MODULE_ENABLED */
#ifdef HAL_CRYP_MODULE_ENABLED
  #include "stm32f4xx_hal_cryp.h"
#endif /* HAL_CRYP_MODULE_ENABLED */
#ifdef HAL_DMA2D_MODULE_ENABLED
  #include "stm32f4xx_hal_dma2d.h"
#endif /* HAL_DMA2D_MODULE_ENABLED */
#ifdef HAL_DAC_MODULE_ENABLED
  #include "stm32f4xx_hal_dac.h"
#endif /* HAL_DAC_MODULE_ENABLED */
#ifdef HAL_DCMI_MODULE_ENABLED
  #include "stm32f4xx_hal_dcmi.h"
#endif /* HAL_DCMI_MODULE_ENABLED */
#ifdef HAL_ETH_MODULE_ENABLED
  #include "stm32f4xx_hal_eth.h"
#endif /* HAL_ETH_MODULE_ENABLED */
```

```
#ifdef HAL_ETH_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_eth_legacy.h"
#endif /* HAL_ETH_LEGACY_MODULE_ENABLED */
#ifdef HAL_FLASH_MODULE_ENABLED
  #include "stm32f4xx_hal_flash.h"
#endif /* HAL_FLASH_MODULE_ENABLED */
#ifdef HAL_SRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sram.h"
#endif /* HAL_SRAM_MODULE_ENABLED */
#ifdef HAL_NOR_MODULE_ENABLED
  #include "stm32f4xx_hal_nor.h"
#endif /* HAL_NOR_MODULE_ENABLED */
#ifdef HAL_NAND_MODULE_ENABLED
  #include "stm32f4xx_hal_nand.h"
#endif /* HAL_NAND_MODULE_ENABLED */
#ifdef HAL_PCCARD_MODULE_ENABLED
  #include "stm32f4xx_hal_pccard.h"
#endif /* HAL_PCCARD_MODULE_ENABLED */
#ifdef HAL_SDRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sdram.h"
#endif /* HAL_SDRAM_MODULE_ENABLED */
#ifdef HAL_HASH_MODULE_ENABLED
 #include "stm32f4xx_hal_hash.h"
```

```
#endif /* HAL_HASH_MODULE_ENABLED */
#ifdef HAL_I2C_MODULE_ENABLED
 #include "stm32f4xx_hal_i2c.h"
#endif /* HAL_I2C_MODULE_ENABLED */
#ifdef HAL_SMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_smbus.h"
#endif /* HAL_SMBUS_MODULE_ENABLED */
#ifdef HAL_I2S_MODULE_ENABLED
 #include "stm32f4xx_hal_i2s.h"
#endif /* HAL_I2S_MODULE_ENABLED */
#ifdef HAL_IWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_iwdg.h"
#endif /* HAL_IWDG_MODULE_ENABLED */
#ifdef HAL_LTDC_MODULE_ENABLED
 #include "stm32f4xx_hal_ltdc.h"
#endif /* HAL_LTDC_MODULE_ENABLED */
#ifdef HAL_PWR_MODULE_ENABLED
 #include "stm32f4xx_hal_pwr.h"
#endif /* HAL_PWR_MODULE_ENABLED */
#ifdef HAL_RNG_MODULE_ENABLED
 #include "stm32f4xx_hal_rng.h"
#endif /* HAL_RNG_MODULE_ENABLED */
#ifdef HAL_RTC_MODULE_ENABLED
```

```
 #include "stm32f4xx_hal_rtc.h"
#endif /* HAL_RTC_MODULE_ENABLED */
#ifdef HAL_SAI_MODULE_ENABLED
 #include "stm32f4xx_hal_sai.h"
#endif /* HAL_SAI_MODULE_ENABLED */
#ifdef HAL_SD_MODULE_ENABLED
 #include "stm32f4xx_hal_sd.h"
#endif /* HAL_SD_MODULE_ENABLED */
#ifdef HAL_SPI_MODULE_ENABLED
 #include "stm32f4xx_hal_spi.h"
#endif /* HAL_SPI_MODULE_ENABLED */
#ifdef HAL_TIM_MODULE_ENABLED
 #include "stm32f4xx_hal_tim.h"
#endif /* HAL_TIM_MODULE_ENABLED */
#ifdef HAL_UART_MODULE_ENABLED
 #include "stm32f4xx_hal_uart.h"
#endif /* HAL_UART_MODULE_ENABLED */
#ifdef HAL_USART_MODULE_ENABLED
 #include "stm32f4xx_hal_usart.h"
#endif /* HAL_USART_MODULE_ENABLED */
#ifdef HAL_IRDA_MODULE_ENABLED
 #include "stm32f4xx_hal_irda.h"
#endif /* HAL_IRDA_MODULE_ENABLED */
```

```
#ifdef HAL_SMARTCARD_MODULE_ENABLED
 #include "stm32f4xx_hal_smartcard.h"
#endif /* HAL_SMARTCARD_MODULE_ENABLED */
#ifdef HAL_WWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_wwdg.h"
#endif /* HAL_WWDG_MODULE_ENABLED */
#ifdef HAL_PCD_MODULE_ENABLED
 #include "stm32f4xx_hal_pcd.h"
#endif /* HAL_PCD_MODULE_ENABLED */
#ifdef HAL_HCD_MODULE_ENABLED
 #include "stm32f4xx_hal_hcd.h"
#endif /* HAL_HCD_MODULE_ENABLED */
#ifdef HAL_DSI_MODULE_ENABLED
 #include "stm32f4xx_hal_dsi.h"
#endif /* HAL_DSI_MODULE_ENABLED */
#ifdef HAL_QSPI_MODULE_ENABLED
 #include "stm32f4xx_hal_qspi.h"
#endif /* HAL_QSPI_MODULE_ENABLED */
#ifdef HAL_CEC_MODULE_ENABLED
 #include "stm32f4xx_hal_cec.h"
#endif /* HAL_CEC_MODULE_ENABLED */
#ifdef HAL_FMPI2C_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpi2c.h"
```

```
#endif /* HAL_FMPI2C_MODULE_ENABLED */
#ifdef HAL_FMPSMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpsmbus.h"
#endif /* HAL_FMPSMBUS_MODULE_ENABLED */
#ifdef HAL_SPDIFRX_MODULE_ENABLED
 #include "stm32f4xx_hal_spdifrx.h"
#endif /* HAL_SPDIFRX_MODULE_ENABLED */
#ifdef HAL_DFSDM_MODULE_ENABLED
 #include "stm32f4xx_hal_dfsdm.h"
#endif /* HAL_DFSDM_MODULE_ENABLED */
#ifdef HAL_LPTIM_MODULE_ENABLED
 #include "stm32f4xx_hal_lptim.h"
#endif /* HAL_LPTIM_MODULE_ENABLED */
#ifdef HAL_MMC_MODULE_ENABLED
 #include "stm32f4xx_hal_mmc.h"
#endif /* HAL_MMC_MODULE_ENABLED */
/* Exported macro ------------------------------------------------------------*/
#ifdef  USE_FULL_ASSERT
/**
  * @brief  The assert_param macro is used for function's parameters check.
  * @param  expr If expr is false, it calls assert_failed function
  *         which reports the name of the source file and the source
```

```
  *        line number of the call that failed.
  *        If expr is true, it returns no value.
  * @retval None
  */
  #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t
*)__FILE__, __LINE__))
/* Exported functions ---------------------------------------------------- */
  void assert_failed(uint8_t* file, uint32_t line);
#else
  #define assert_param(expr) ((void)0U)
#endif /* USE_FULL_ASSERT */
#ifdef __cplusplus
}
#endif
#endif /* __STM32F4xx_HAL_CONF_H */
```

28.11 CODE OF STM32F4XX_IT.H:

```
/* USER CODE BEGIN Header */
/**
*********************************************************
********************
  * @file    stm32f4xx_it.h
  * @brief   This file contains the headers of the interrupt handlers.
*********************************************************
********************
```

```
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*****************************************************************
*******************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_IT_H
#define __STM32F4xx_IT_H
#ifdef __cplusplus
 extern "C" {
#endif
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Exported types ------------------------------------------------------------*/
```

```
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants ------------------------------------------------------*/
/* USER CODE BEGIN EC */
/* USER CODE END EC */
/* Exported macro ----------------------------------------------------------*/
/* USER CODE BEGIN EM */
/* USER CODE END EM */
/* Exported functions prototypes -------------------------------------------*/
void NMI_Handler(void);
void HardFault_Handler(void);
void MemManage_Handler(void);
void BusFault_Handler(void);
void UsageFault_Handler(void);
void SVC_Handler(void);
void DebugMon_Handler(void);
void PendSV_Handler(void);
void SysTick_Handler(void);
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
#ifdef __cplusplus
}
#endif
```

```
#endif /* __STM32F4xx_IT_H */
```

28.12 CODE OF MAIN.C:

```
/* USER CODE BEGIN Header */
/**
**********************************************************
********************
  * @file           : main.c
  * @brief          : Main program body
**********************************************************
********************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
********************
  */
/* USER CODE END Header */
```

```
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
/* USER CODE END PTD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
UART_HandleTypeDef huart2;
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
/* USER CODE BEGIN PFP */
```

```
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */
  /* USER CODE END 1 */
  /* MCU Configuration--------------------------------------------------------*/
  /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
  HAL_Init();
  /* USER CODE BEGIN Init */
  /* USER CODE END Init */
  /* Configure the system clock */
  SystemClock_Config();
  /* USER CODE BEGIN SysInit */
  /* USER CODE END SysInit */
  /* Initialize all configured peripherals */
```

```
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  /* USER CODE BEGIN 2 */
  /* USER CODE END 2 */
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  /** Configure the main internal regulator output voltage
  */
```

```
  __HAL_RCC_PWR_CLK_ENABLE();
__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_
VOLTAGE_SCALE1);
  /** Initializes the RCC Oscillators according to the specified
parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue =
RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 16;
  RCC_OscInitStruct.PLL.PLLN = 336;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV4;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
```

```
  RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource =
RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct,
FLASH_LATENCY_2) != HAL_OK)
  {
    Error_Handler();
  }
}
/**
  * @brief USART2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART2_UART_Init(void)
{
  /* USER CODE BEGIN USART2_Init 0 */
  /* USER CODE END USART2_Init 0 */
  /* USER CODE BEGIN USART2_Init 1 */
```

```
  /* USER CODE END USART2_Init 1 */
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 115200;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART2_Init 2 */
  /* USER CODE END USART2_Init 2 */
}
/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
```

```
  GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */
  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin,
GPIO_PIN_RESET);
  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);
  /*Configure GPIO pin : LD2_Pin */
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);
/* USER CODE BEGIN MX_GPIO_Init_2 */
```

```
/* USER CODE END MX_GPIO_Init_2 */
}
/* USER CODE BEGIN 4 */
/* USER CODE END 4 */
/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return
state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}
#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line
number
  *         where the assert_param error has occurred.
```

```
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

28.13 CODE OF STM32F4XX_HAL_MSP.C:

```
/* USER CODE BEGIN Header */
/**
*******************************************************************
  * @file         stm32f4xx_hal_msp.c
  * @brief        This file provides code for the MSP Initialization
  *               and de-Initialization codes.
*******************************************************************
```

```
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN TD */
/* USER CODE END TD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN Define */
/* USER CODE END Define */
```

```
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN Macro */
/* USER CODE END Macro */
/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* External functions --------------------------------------------------------*/
/* USER CODE BEGIN ExternalFunctions */
/* USER CODE END ExternalFunctions */
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/**
  * Initializes the Global MSP.
  */
void HAL_MspInit(void)
{
  /* USER CODE BEGIN MspInit 0 */
  /* USER CODE END MspInit 0 */
  __HAL_RCC_SYSCFG_CLK_ENABLE();
  __HAL_RCC_PWR_CLK_ENABLE();
```

```
  HAL_NVIC_SetPriorityGrouping(NVIC_PRIORITYGROUP_0);
  /* System interrupt init*/
  /* USER CODE BEGIN MspInit 1 */
  /* USER CODE END MspInit 1 */
}
/**
* @brief UART MSP Initialization
* This function configures the hardware resources used in this example
* @param huart: UART handle pointer
* @retval None
*/
void HAL_UART_MspInit(UART_HandleTypeDef* huart)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  if(huart->Instance==USART2)
  {
  /* USER CODE BEGIN USART2_MspInit 0 */
  /* USER CODE END USART2_MspInit 0 */
    /* Peripheral clock enable */
    __HAL_RCC_USART2_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();
    /**USART2 GPIO Configuration
    PA2     ------> USART2_TX
```

```
    PA3     ------> USART2_RX
    */
    GPIO_InitStruct.Pin = USART_TX_Pin|USART_RX_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF7_USART2;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
  /* USER CODE BEGIN USART2_MspInit 1 */
  /* USER CODE END USART2_MspInit 1 */
  }
}
/**
* @brief UART MSP De-Initialization
* This function freeze the hardware resources used in this example
* @param huart: UART handle pointer
* @retval None
*/
void HAL_UART_MspDeInit(UART_HandleTypeDef* huart)
{
  if(huart->Instance==USART2)
  {
  /* USER CODE BEGIN USART2_MspDeInit 0 */
```

```
  /* USER CODE END USART2_MspDeInit 0 */
    /* Peripheral clock disable */
    __HAL_RCC_USART2_CLK_DISABLE();
    /**USART2 GPIO Configuration
    PA2     ------> USART2_TX
    PA3     ------> USART2_RX
    */
    HAL_GPIO_DeInit(GPIOA, USART_TX_Pin|USART_RX_Pin);
  /* USER CODE BEGIN USART2_MspDeInit 1 */
  /* USER CODE END USART2_MspDeInit 1 */
  }
}
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

28.14 CODE OF STM32F4XX_IT.C:

```
/* USER CODE BEGIN Header */
/**
*************************************************************
********************
  * @file    stm32f4xx_it.c
  * @brief   Interrupt Service Routines.
*************************************************************
********************
  * @attention
```

```
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*****************************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stm32f4xx_it.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN TD */
/* USER CODE END TD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
```

```
/* USER CODE END PD */
/* Private macro -------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------*/
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------*/
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/* External variables --------------------------------------------------*/
/* USER CODE BEGIN EV */
/* USER CODE END EV */
/******************************************************************************/
/*           Cortex-M4 Processor Interruption and Exception Handlers          */
/******************************************************************************/
/**
  * @brief This function handles Non maskable interrupt.
```

```
  */
void NMI_Handler(void)
{
  /* USER CODE BEGIN NonMaskableInt_IRQn 0 */
  /* USER CODE END NonMaskableInt_IRQn 0 */
  /* USER CODE BEGIN NonMaskableInt_IRQn 1 */
   while (1)
  {
  }
  /* USER CODE END NonMaskableInt_IRQn 1 */
}
/**
  * @brief This function handles Hard fault interrupt.
  */
void HardFault_Handler(void)
{
  /* USER CODE BEGIN HardFault_IRQn 0 */
  /* USER CODE END HardFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_HardFault_IRQn 0 */
    /* USER CODE END W1_HardFault_IRQn 0 */
  }
```

```
}
/**
  * @brief This function handles Memory management fault.
  */
void MemManage_Handler(void)
{
  /* USER CODE BEGIN MemoryManagement_IRQn 0 */
  /* USER CODE END MemoryManagement_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_MemoryManagement_IRQn 0 */
    /* USER CODE END W1_MemoryManagement_IRQn 0 */
  }
}
/**
  * @brief This function handles Pre-fetch fault, memory access fault.
  */
void BusFault_Handler(void)
{
  /* USER CODE BEGIN BusFault_IRQn 0 */
  /* USER CODE END BusFault_IRQn 0 */
  while (1)
  {
```

```
    /* USER CODE BEGIN W1_BusFault_IRQn 0 */
    /* USER CODE END W1_BusFault_IRQn 0 */
  }
}
/**
  * @brief This function handles Undefined instruction or illegal state.
  */
void UsageFault_Handler(void)
{
  /* USER CODE BEGIN UsageFault_IRQn 0 */
  /* USER CODE END UsageFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_UsageFault_IRQn 0 */
    /* USER CODE END W1_UsageFault_IRQn 0 */
  }
}
/**
  * @brief This function handles System service call via SWI
instruction.
  */
void SVC_Handler(void)
{
```

```
  /* USER CODE BEGIN SVCall_IRQn 0 */
  /* USER CODE END SVCall_IRQn 0 */
  /* USER CODE BEGIN SVCall_IRQn 1 */
  /* USER CODE END SVCall_IRQn 1 */
}
/**
  * @brief This function handles Debug monitor.
  */
void DebugMon_Handler(void)
{
  /* USER CODE BEGIN DebugMonitor_IRQn 0 */
  /* USER CODE END DebugMonitor_IRQn 0 */
  /* USER CODE BEGIN DebugMonitor_IRQn 1 */
  /* USER CODE END DebugMonitor_IRQn 1 */
}
/**
  * @brief This function handles Pendable request for system service.
  */
void PendSV_Handler(void)
{
  /* USER CODE BEGIN PendSV_IRQn 0 */
  /* USER CODE END PendSV_IRQn 0 */
  /* USER CODE BEGIN PendSV_IRQn 1 */
```

```
  /* USER CODE END PendSV_IRQn 1 */
}
/**
  * @brief This function handles System tick timer.
  */
void SysTick_Handler(void)
{
  /* USER CODE BEGIN SysTick_IRQn 0 */
  /* USER CODE END SysTick_IRQn 0 */
  HAL_IncTick();
  /* USER CODE BEGIN SysTick_IRQn 1 */
  /* USER CODE END SysTick_IRQn 1 */
}
/******************************************************************************/
/* STM32F4xx Peripheral Interrupt Handlers                                    */
/* Add here the Interrupt Handlers for the used peripherals.                  */
/* For the available peripheral interrupt handler names,                      */
/* please refer to the startup file (startup_stm32f4xx.s).                   */
/******************************************************************************/
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

28.15 CODE OF SYSCALLS.C:

```
/**
**********************************************************
*******************
 * @file      syscalls.c
 * @author    Auto-generated by STM32CubeIDE
 * @brief     STM32CubeIDE Minimal System calls file
 *
 *            For more information about which c-functions
 *            need which of these lowlevel functions
 *            please consult the Newlib libc-manual
**********************************************************
*******************
 * @attention
 *
 * Copyright (c) 2020-2024 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the
LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
**********************************************************
*******************
```

```
 */
/* Includes */
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
#include <sys/times.h>
/* Variables */
extern int __io_putchar(int ch) __attribute__((weak));
extern int __io_getchar(void) __attribute__((weak));
char *__env[1] = { 0 };
char **environ = __env;
/* Functions */
void initialise_monitor_handles()
{
}
int _getpid(void)
{
  return 1;
}
```

```
int _kill(int pid, int sig)
{
  (void)pid;
  (void)sig;
  errno = EINVAL;
  return -1;
}
void _exit (int status)
{
  _kill(status, -1);
  while (1) {}    /* Make sure we hang here */
}
__attribute__((weak)) int _read(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;
  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    *ptr++ = __io_getchar();
  }
  return len;
}
__attribute__((weak)) int _write(int file, char *ptr, int len)
```

```
{
  (void)file;
  int DataIdx;
  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    __io_putchar(*ptr++);
  }
  return len;
}
int _close(int file)
{
  (void)file;
  return -1;
}
int _fstat(int file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
int _isatty(int file)
{
  (void)file;
```

```
  return 1;
}
int _lseek(int file, int ptr, int dir)
{
  (void)file;
  (void)ptr;
  (void)dir;
  return 0;
}
int _open(char *path, int flags, ...)
{
  (void)path;
  (void)flags;
  /* Pretend like we always fail */
  return -1;
}
int _wait(int *status)
{
  (void)status;
  errno = ECHILD;
  return -1;
}
int _unlink(char *name)
```

```
{
  (void)name;
  errno = ENOENT;
  return -1;
}
int _times(struct tms *buf)
{
  (void)buf;
  return -1;
}
int _stat(char *file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
int _link(char *old, char *new)
{
  (void)old;
  (void)new;
  errno = EMLINK;
  return -1;
}
```

```
int _fork(void)
{
  errno = EAGAIN;
  return -1;
}
int _execve(char *name, char **argv, char **env)
{
  (void)name;
  (void)argv;
  (void)env;
  errno = ENOMEM;
  return -1;
}
```

28.16 CODE OF SYSMEM.C:

```
/**
*****************************************************************
*******************
 * @file      sysmem.c
 * @author    Generated by STM32CubeIDE
 * @brief     STM32CubeIDE System Memory calls file
 *
 *            For more information about which C functions
 *            need which of these lowlevel functions
```

```
 *          please consult the newlib libc manual
**********************************************************
*******************
 * @attention
 *
 * Copyright (c) 2024 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the
LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
**********************************************************
*******************
 */
/* Includes */
#include <errno.h>
#include <stdint.h>
/**
 * Pointer to the current high watermark of the heap usage
 */
static uint8_t *__sbrk_heap_end = NULL;
/**
```

```
 * @brief _sbrk() allocates memory to the newlib heap and is used by malloc
 *        and others from the C library
 *
 * @verbatim
 *
 ##########################################################################
 * #  .data  #  .bss  #       newlib heap       #          MSP stack          #
 * #         #        #                         # Reserved by _Min_Stack_Size #
 *
 ##########################################################################
 * ^-- RAM start      ^-- _end                          _estack, RAM end --^
 * @endverbatim
 *
 * This implementation starts allocating at the '_end' linker symbol
 * The '_Min_Stack_Size' linker symbol reserves a memory for the MSP stack
 * The implementation considers '_estack' linker symbol to be RAM end
 * NOTE: If the MSP stack, at any point during execution, grows larger than the
 * reserved size, please increase the '_Min_Stack_Size'.
 *
 * @param incr Memory size
```

```
 * @return Pointer to allocated memory
 */
void *_sbrk(ptrdiff_t incr)
{
  extern uint8_t _end; /* Symbol defined in the linker script */
  extern uint8_t _estack; /* Symbol defined in the linker script */
  extern uint32_t _Min_Stack_Size; /* Symbol defined in the linker
script */
  const uint32_t stack_limit = (uint32_t)&_estack -
(uint32_t)&_Min_Stack_Size;
  const uint8_t *max_heap = (uint8_t *)stack_limit;
  uint8_t *prev_heap_end;
  /* Initialize heap end at first call */
  if (NULL == __sbrk_heap_end)
  {
    __sbrk_heap_end = &_end;
  }
  /* Protect heap from growing into the reserved MSP stack */
  if (__sbrk_heap_end + incr > max_heap)
  {
    errno = ENOMEM;
    return (void *)-1;
  }
  prev_heap_end = __sbrk_heap_end;
```

```
  __sbrk_heap_end += incr;
  return (void *)prev_heap_end;
}
```

28.17 CODE OF SYSTEM_STM32F4XX.C:

```
/**
*****************************************************************
*******************
  * @file    system_stm32f4xx.c
  * @author  MCD Application Team
  * @brief   CMSIS Cortex-M4 Device Peripheral Access Layer System
Source File.
  *
  *   This file provides two functions and one global variable to be called
from
  *   user application:
  *      - SystemInit(): This function is called at startup just after reset and
  *                      before branch to main program. This call is made inside
  *                      the "startup_stm32f4xx.s" file.
  *
  *      - SystemCoreClock variable: Contains the core clock (HCLK), it
can be used
  *                                  by the user application to setup the SysTick
  *                                  timer or configure other parameters.
  *
```

```
  *      - SystemCoreClockUpdate(): Updates the variable
SystemCoreClock and must
  *                              be called whenever the core clock is changed
  *                              during program execution.
  *
  *
**********************************************************
********************
  * @attention
  *
  * Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
********************
  */
/** @addtogroup CMSIS
  * @{
  */
/** @addtogroup stm32f4xx_system
```

```
  * @{
  */
/** @addtogroup STM32F4xx_System_Private_Includes
  * @{
  */
#include "stm32f4xx.h"
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    ((uint32_t)25000000) /*!< Default value of
the External oscillator in Hz */
#endif /* HSE_VALUE */
#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000) /*!< Value of the
Internal oscillator in Hz*/
#endif /* HSI_VALUE */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_TypesDefinitions
  * @{
  */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Defines
```

```
  * @{
  */
/************************* Miscellaneous Configuration ************************/
/*!< Uncomment the following line if you need to use external SRAM or SDRAM as data memory  */
#if defined(STM32F405xx) || defined(STM32F415xx) || defined(STM32F407xx) || defined(STM32F417xx)\
 || defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx) || defined(STM32F412Zx) || defined(STM32F412Vx)
/* #define DATA_IN_ExtSRAM */
#endif /* STM32F40xxx || STM32F41xxx || STM32F42xxx || STM32F43xxx || STM32F469xx || STM32F479xx ||\
        STM32F412Zx || STM32F412Vx */
#if defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F446xx) || defined(STM32F469xx) || defined(STM32F479xx)
/* #define DATA_IN_ExtSDRAM */
#endif /* STM32F427xx || STM32F437xx || STM32F429xx || STM32F439xx || STM32F446xx || STM32F469xx ||\
        STM32F479xx */
/* Note: Following vector table addresses must be defined in line with linker
```

configuration. */

/*!< Uncomment the following line if you need to relocate the vector table

anywhere in Flash or Sram, else the vector table is kept at the automatic

remap of boot address selected */

/* #define USER_VECT_TAB_ADDRESS */

#if defined(USER_VECT_TAB_ADDRESS)

/*!< Uncomment the following line if you need to relocate your vector Table

in Sram else user remap will be done in Flash. */

/* #define VECT_TAB_SRAM */

#if defined(VECT_TAB_SRAM)

#define VECT_TAB_BASE_ADDRESS SRAM_BASE /*!< Vector Table base address field.

This value must be a multiple of 0x200. */

#define VECT_TAB_OFFSET 0x00000000U /*!< Vector Table base offset field.

This value must be a multiple of 0x200. */

#else

#define VECT_TAB_BASE_ADDRESS FLASH_BASE /*!< Vector Table base address field.

This value must be a multiple of 0x200. */

```
#define VECT_TAB_OFFSET        0x00000000U     /*!< Vector Table
base offset field.
                                This value must be a multiple of 0x200.
*/
#endif /* VECT_TAB_SRAM */
#endif /* USER_VECT_TAB_ADDRESS */
/*****************************************************************
*********************/
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Macros
  * @{
  */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Variables
  * @{
  */
  /* This variable is updated in three ways:
     1) by calling CMSIS function SystemCoreClockUpdate()
     2) by calling HAL API function HAL_RCC_GetHCLKFreq()
```

```
     3) each time HAL_RCC_ClockConfig() is called to configure the
system clock frequency

        Note: If you use this function to configure the system clock; then
there

              is no need to call the 2 first functions listed above, since
SystemCoreClock

              variable is updated automatically.

  */

uint32_t SystemCoreClock = 16000000;

const uint8_t AHBPrescTable[16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7,
8, 9};

const uint8_t APBPrescTable[8]  = {0, 0, 0, 0, 1, 2, 3, 4};

/**

  * @}

  */

/** @addtogroup STM32F4xx_System_Private_FunctionPrototypes

  * @{

  */

#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)

  static void SystemInit_ExtMemCtl(void);

#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */

/**

  * @}

  */
```

```
/** @addtogroup STM32F4xx_System_Private_Functions
  * @{
  */
/**
  * @brief  Setup the microcontroller system
  *         Initialize the FPU setting, vector table location and External memory
  *         configuration.
  * @param  None
  * @retval None
  */
void SystemInit(void)
{
  /* FPU settings ------------------------------------------------------------*/
  #if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
    SCB->CPACR |= ((3UL << 10*2)|(3UL << 11*2));  /* set CP10 and CP11 Full Access */
  #endif
#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  SystemInit_ExtMemCtl();
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */
  /* Configure the Vector Table location -------------------------------------*/
#if defined(USER_VECT_TAB_ADDRESS)
```

```
  SCB->VTOR = VECT_TAB_BASE_ADDRESS |
VECT_TAB_OFFSET; /* Vector Table Relocation in Internal SRAM */
#endif /* USER_VECT_TAB_ADDRESS */
}
/**
   * @brief  Update SystemCoreClock variable according to Clock
Register Values.
  *         The SystemCoreClock variable contains the core clock (HCLK),
it can
  *         be used by the user application to setup the SysTick timer or
configure
  *         other parameters.
  *
  * @note   Each time the core clock (HCLK) changes, this function
must be called
  *         to update SystemCoreClock variable value. Otherwise, any
configuration
  *         based on this variable will be incorrect.
  *
  * @note   - The system frequency computed by this function is not the
real
  *           frequency in the chip. It is calculated based on the predefined
  *           constant and the selected clock source:
  *
```

```
  *          - If SYSCLK source is HSI, SystemCoreClock will contain the
HSI_VALUE(*)
  *
  *          - If SYSCLK source is HSE, SystemCoreClock will contain the
HSE_VALUE(**)
  *
  *          - If SYSCLK source is PLL, SystemCoreClock will contain the
HSE_VALUE(**)
  *            or HSI_VALUE(*) multiplied/divided by the PLL factors.
  *
  *        (*) HSI_VALUE is a constant defined in stm32f4xx_hal_conf.h
file (default value
  *            16 MHz) but the real value may vary depending on the
variations
  *            in voltage and temperature.
  *
  *        (**) HSE_VALUE is a constant defined in
stm32f4xx_hal_conf.h file (its value
  *             depends on the application requirements), user has to ensure
that HSE_VALUE
  *             is same as the real frequency of the crystal used. Otherwise,
this function
  *             may have wrong result.
  *
  *        - The result of this function could be not correct when using
fractional
```

```
  *          value for HSE crystal.
  *
  * @param  None
  * @retval None
  */
void SystemCoreClockUpdate(void)
{
  uint32_t tmp = 0, pllvco = 0, pllp = 2, pllsource = 0, pllm = 2;
  /* Get SYSCLK source -------------------------------------------------------
*/
  tmp = RCC->CFGR & RCC_CFGR_SWS;
  switch (tmp)
  {
    case 0x00:  /* HSI used as system clock source */
      SystemCoreClock = HSI_VALUE;
      break;
    case 0x04:  /* HSE used as system clock source */
      SystemCoreClock = HSE_VALUE;
      break;
    case 0x08:  /* PLL used as system clock source */
      /* PLL_VCO = (HSE_VALUE or HSI_VALUE / PLL_M) * PLL_N
         SYSCLK = PLL_VCO / PLL_P
         */
```

```
    pllsource = (RCC->PLLCFGR & RCC_PLLCFGR_PLLSRC) >>
22;
    pllm = RCC->PLLCFGR & RCC_PLLCFGR_PLLM;


    if (pllsource != 0)
    {
      /* HSE used as PLL clock source */
      pllvco = (HSE_VALUE / pllm) * ((RCC->PLLCFGR &
RCC_PLLCFGR_PLLN) >> 6);
    }
    else
    {
      /* HSI used as PLL clock source */
      pllvco = (HSI_VALUE / pllm) * ((RCC->PLLCFGR &
RCC_PLLCFGR_PLLN) >> 6);
    }
    pllp = (((RCC->PLLCFGR & RCC_PLLCFGR_PLLP) >>16) + 1 )
*2;
    SystemCoreClock = pllvco/pllp;
    break;
   default:
    SystemCoreClock = HSI_VALUE;
    break;
  }
```

```
  /* Compute HCLK frequency ----------------------------------------------
-*/
  /* Get HCLK prescaler */
  tmp = AHBPrescTable[((RCC->CFGR & RCC_CFGR_HPRE) >> 4)];
  /* HCLK frequency */
  SystemCoreClock >>= tmp;
}
#if defined (DATA_IN_ExtSRAM) && defined
(DATA_IN_ExtSDRAM)
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
/**
  * @brief  Setup the external memory controller.
  *         Called in startup_stm32f4xx.s before jump to main.
  *         This function configures the external memories
(SRAM/SDRAM)
  *         This SRAM/SDRAM will be used as program data memory
(including heap and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;
```

```
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
  register __IO uint32_t index;
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and
GPIOI interface clock */
  RCC->AHB1ENR |= 0x000001F8;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x00CCC0CC;
  GPIOD->AFR[1]  = 0xCCCCCCCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xAAAA0A8A;
  /* Configure PDx pins speed to 100 MHz */
  GPIOD->OSPEEDR = 0xFFFF0FCF;
  /* Configure PDx pins Output type to push-pull */
  GPIOD->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOD->PUPDR   = 0x00000000;
  /* Connect PEx pins to FMC Alternate function */
  GPIOE->AFR[0]  = 0xC00CC0CC;
  GPIOE->AFR[1]  = 0xCCCCCCCC;
  /* Configure PEx pins in Alternate function mode */
  GPIOE->MODER   = 0xAAAA828A;
```

```
/* Configure PEx pins speed to 100 MHz */
GPIOE->OSPEEDR = 0xFFFFC3CF;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0xCCCCCCCC;
GPIOF->AFR[1]  = 0xCCCCCCCC;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA800AAA;
/* Configure PFx pins speed to 50 MHz */
GPIOF->OSPEEDR = 0xAA800AAA;
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
GPIOF->PUPDR   = 0x00000000;
/* Connect PGx pins to FMC Alternate function */
GPIOG->AFR[0]  = 0xCCCCCCCC;
GPIOG->AFR[1]  = 0xCCCCCCCC;
/* Configure PGx pins in Alternate function mode */
GPIOG->MODER   = 0xAAAAAAAA;
/* Configure PGx pins speed to 50 MHz */
```

```
GPIOG->OSPEEDR = 0xAAAAAAAA;
/* Configure PGx pins Output type to push-pull */
GPIOG->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PGx pins */
GPIOG->PUPDR   = 0x00000000;
/* Connect PHx pins to FMC Alternate function */
GPIOH->AFR[0]  = 0x00C0CC00;
GPIOH->AFR[1]  = 0xCCCCCCCC;
/* Configure PHx pins in Alternate function mode */
GPIOH->MODER   = 0xAAAA08A0;
/* Configure PHx pins speed to 50 MHz */
GPIOH->OSPEEDR = 0xAAAA08A0;
/* Configure PHx pins Output type to push-pull */
GPIOH->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PHx pins */
GPIOH->PUPDR   = 0x00000000;
/* Connect PIx pins to FMC Alternate function */
GPIOI->AFR[0]  = 0xCCCCCCCC;
GPIOI->AFR[1]  = 0x00000CC0;
/* Configure PIx pins in Alternate function mode */
GPIOI->MODER   = 0x0028AAAA;
/* Configure PIx pins speed to 50 MHz */
GPIOI->OSPEEDR = 0x0028AAAA;
```

```
  /* Configure PIx pins Output type to push-pull */
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;
/*-- FMC Configuration ------------------------------------------------------*/
  /* Enable the FMC interface clock */
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
  FMC_Bank5_6->SDTR[0] = 0x01115351;
  /* SDRAM initialization sequence */
  /* Clock enable command */
  FMC_Bank5_6->SDCMR = 0x00000011;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Delay */
  for (index = 0; index<1000; index++);
  /* PALL command */
  FMC_Bank5_6->SDCMR = 0x00000012;
```

```
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* Auto refresh command */
FMC_Bank5_6->SDCMR = 0x00000073;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* MRD register program */
FMC_Bank5_6->SDCMR = 0x00046014;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* Set refresh count */
```

```
  tmpreg = FMC_Bank5_6->SDRTR;
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
  /* Disable write protection */
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F469xx || STM32F479xx */
  (void)(tmp);
}
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F469xx || STM32F479xx */
```

```
#elif defined (DATA_IN_ExtSRAM) || defined
(DATA_IN_ExtSDRAM)
/**
  * @brief  Setup the external memory controller.
  *         Called in startup_stm32f4xx.s before jump to main.
  *         This function configures the external memories
(SRAM/SDRAM)
  *         This SRAM/SDRAM will be used as program data memory
(including heap and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F446xx) || defined(STM32F469xx) ||
defined(STM32F479xx)
#if defined (DATA_IN_ExtSDRAM)
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
  register __IO uint32_t index;
#if defined(STM32F446xx)
  /* Enable GPIOA, GPIOC, GPIOD, GPIOE, GPIOF, GPIOG interface
     clock */
```

```
  RCC->AHB1ENR |= 0x0000007D;
#else
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and
GPIOI interface
     clock */
  RCC->AHB1ENR |= 0x000001F8;
#endif /* STM32F446xx */
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);
#if defined(STM32F446xx)
  /* Connect PAx pins to FMC Alternate function */
  GPIOA->AFR[0]  |= 0xC0000000;
  GPIOA->AFR[1]  |= 0x00000000;
  /* Configure PDx pins in Alternate function mode */
  GPIOA->MODER   |= 0x00008000;
  /* Configure PDx pins speed to 50 MHz */
  GPIOA->OSPEEDR |= 0x00008000;
  /* Configure PDx pins Output type to push-pull */
  GPIOA->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOA->PUPDR   |= 0x00000000;
  /* Connect PCx pins to FMC Alternate function */
  GPIOC->AFR[0]  |= 0x00CC0000;
```

```
  GPIOC->AFR[1]  |= 0x00000000;
  /* Configure PDx pins in Alternate function mode */
  GPIOC->MODER   |= 0x00000A00;
  /* Configure PDx pins speed to 50 MHz */
  GPIOC->OSPEEDR |= 0x00000A00;
  /* Configure PDx pins Output type to push-pull */
  GPIOC->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOC->PUPDR   |= 0x00000000;
#endif /* STM32F446xx */
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x000000CC;
  GPIOD->AFR[1]  = 0xCC000CCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xA02A000A;
  /* Configure PDx pins speed to 50 MHz */
  GPIOD->OSPEEDR = 0xA02A000A;
  /* Configure PDx pins Output type to push-pull */
  GPIOD->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOD->PUPDR   = 0x00000000;
  /* Connect PEx pins to FMC Alternate function */
  GPIOE->AFR[0]  = 0xC00000CC;
```

```
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA800A;
/* Configure PEx pins speed to 50 MHz */
GPIOE->OSPEEDR = 0xAAAA800A;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0xCCCCCCCC;
GPIOF->AFR[1]  = 0xCCCCCCCC;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA800AAA;
/* Configure PFx pins speed to 50 MHz */
GPIOF->OSPEEDR = 0xAA800AAA;
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
GPIOF->PUPDR   = 0x00000000;
/* Connect PGx pins to FMC Alternate function */
GPIOG->AFR[0]  = 0xCCCCCCCC;
GPIOG->AFR[1]  = 0xCCCCCCCC;
```

```
  /* Configure PGx pins in Alternate function mode */
  GPIOG->MODER   = 0xAAAAAAAA;
  /* Configure PGx pins speed to 50 MHz */
  GPIOG->OSPEEDR = 0xAAAAAAAA;
  /* Configure PGx pins Output type to push-pull */
  GPIOG->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
  /* Connect PHx pins to FMC Alternate function */
  GPIOH->AFR[0]  = 0x00C0CC00;
  GPIOH->AFR[1]  = 0xCCCCCCCC;
  /* Configure PHx pins in Alternate function mode */
  GPIOH->MODER   = 0xAAAA08A0;
  /* Configure PHx pins speed to 50 MHz */
  GPIOH->OSPEEDR = 0xAAAA08A0;
  /* Configure PHx pins Output type to push-pull */
  GPIOH->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PHx pins */
  GPIOH->PUPDR   = 0x00000000;
  /* Connect PIx pins to FMC Alternate function */
```

```
  GPIOI->AFR[0]  = 0xCCCCCCCC;
  GPIOI->AFR[1]  = 0x00000CC0;
  /* Configure PIx pins in Alternate function mode */
  GPIOI->MODER   = 0x0028AAAA;
  /* Configure PIx pins speed to 50 MHz */
  GPIOI->OSPEEDR = 0x0028AAAA;
  /* Configure PIx pins Output type to push-pull */
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F469xx || STM32F479xx */
/*-- FMC Configuration ------------------------------------------------------*/
  /* Enable the FMC interface clock */
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable SDRAM bank1 */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCR[0] = 0x00001954;
#else
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
#endif /* STM32F446xx */
```

```
  FMC_Bank5_6->SDTR[0] = 0x01115351;
  /* SDRAM initialization sequence */
  /* Clock enable command */
  FMC_Bank5_6->SDCMR = 0x00000011;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Delay */
  for (index = 0; index<1000; index++);
  /* PALL command */
  FMC_Bank5_6->SDCMR = 0x00000012;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Auto refresh command */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x000000F3;
#else
```

```
  FMC_Bank5_6->SDCMR = 0x00000073;
#endif /* STM32F446xx */
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* MRD register program */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x00044014;
#else
  FMC_Bank5_6->SDCMR = 0x00046014;
#endif /* STM32F446xx */
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Set refresh count */
  tmpreg = FMC_Bank5_6->SDRTR;
#if defined(STM32F446xx)
```

```
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000050C<<1));
#else
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
#endif /* STM32F446xx */
  /* Disable write protection */
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#endif /* DATA_IN_ExtSDRAM */
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F446xx || STM32F469xx || STM32F479xx */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx) || defined(STM32F417xx)\
 || defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx) ||
defined(STM32F412Zx) || defined(STM32F412Vx)
#if defined(DATA_IN_ExtSRAM)
/*-- GPIOs Configuration -----------------------------------------------------*/
   /* Enable GPIOD, GPIOE, GPIOF and GPIOG interface clock */
  RCC->AHB1ENR   |= 0x00000078;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIODEN);
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x00CCC0CC;
```

```
GPIOD->AFR[1]  = 0xCCCCCCCC;
/* Configure PDx pins in Alternate function mode */
GPIOD->MODER   = 0xAAAA0A8A;
/* Configure PDx pins speed to 100 MHz */
GPIOD->OSPEEDR = 0xFFFF0FCF;
/* Configure PDx pins Output type to push-pull */
GPIOD->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PDx pins */
GPIOD->PUPDR   = 0x00000000;
/* Connect PEx pins to FMC Alternate function */
GPIOE->AFR[0]  = 0xC00CC0CC;
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA828A;
/* Configure PEx pins speed to 100 MHz */
GPIOE->OSPEEDR = 0xFFFFC3CF;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0x00CCCCCC;
GPIOF->AFR[1]  = 0xCCCC0000;
```

```
  /* Configure PFx pins in Alternate function mode */
  GPIOF->MODER   = 0xAA000AAA;
  /* Configure PFx pins speed to 100 MHz */
  GPIOF->OSPEEDR = 0xFF000FFF;
  /* Configure PFx pins Output type to push-pull */
  GPIOF->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PFx pins */
  GPIOF->PUPDR   = 0x00000000;
  /* Connect PGx pins to FMC Alternate function */
  GPIOG->AFR[0]  = 0x00CCCCCC;
  GPIOG->AFR[1]  = 0x000000C0;
  /* Configure PGx pins in Alternate function mode */
  GPIOG->MODER   = 0x00085AAA;
  /* Configure PGx pins speed to 100 MHz */
  GPIOG->OSPEEDR = 0x000CAFFF;
  /* Configure PGx pins Output type to push-pull */
  GPIOG->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;
/*-- FMC/FSMC Configuration ----------------------------------------------
-*/
  /* Enable the FMC/FSMC interface clock */
  RCC->AHB3ENR         |= 0x00000001;
```

```
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F469xx || STM32F479xx */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx)|| defined(STM32F417xx)\
   || defined(STM32F412Zx) || defined(STM32F412Vx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FSMCEN);
  /* Configure and enable Bank1_SRAM2 */
```

```
  FSMC_Bank1->BTCR[2]  = 0x00001011;
  FSMC_Bank1->BTCR[3]  = 0x00000201;
  FSMC_Bank1E->BWTR[2] = 0x0FFFFFFF;
#endif /* STM32F405xx || STM32F415xx || STM32F407xx ||
STM32F417xx || STM32F412Zx || STM32F412Vx */
#endif /* DATA_IN_ExtSRAM */
#endif /* STM32F405xx || STM32F415xx || STM32F407xx ||
STM32F417xx || STM32F427xx || STM32F437xx ||\
        STM32F429xx || STM32F439xx || STM32F469xx ||
STM32F479xx || STM32F412Zx || STM32F412Vx  */
  (void)(tmp);
}
#endif /* DATA_IN_ExtSRAM && DATA_IN_ExtSDRAM */
/**
  * @}
  */
/**
  * @}
  */
/**
  * @}
  */
```

CHAPTER#29 DEVELOPING ANALOG-TO-DIGITAL CONVERSION (ADC) DRIVERS AND FIRMWARE

29.1DEVELOPING ANALOG-TO-DIGITAL CONVERSION (ADC) DRIVERS AND FIRMWARE:

Developing drivers and firmware for Analog-to-Digital Converters (ADC) involves configuring the ADC peripheral, managing data acquisition, and processing the digitized signals. ADCs are crucial in embedded systems for interfacing with analog sensors, such as temperature sensors, light sensors, and pressure sensors, converting their analog signals into digital data that can be processed by a microcontroller. Here's a general guide to developing ADC drivers and firmware for an ARM Cortex-M based microcontroller, like those in the STM32 family.

29.2 UNDERSTANDING ADC OPERATION:

ADCs convert analog voltage levels into digital values. Key parameters include:

29.2.1 RESOLUTION:

The number of bits used to represent the analog value (e.g., 8-bit, 12-bit, 16-bit). Higher resolution provides more precise readings.

29.2.2 REFERNCE VOLTAGE(V_REF):

Determines the range of input voltages the ADC can measure. The digital output represents the ratio of the input voltage to V_ref.

29.2.3 SAMPLING RATE:

How quickly the ADC can sample the input signal. Higher rates allow capturing faster changes in the signal.

29.3 SETTING UP THE ADC:

29.3.1 ENABLING THE ADC CLOCK:

Enable the clock for the ADC peripheral in the RCC (Reset and Clock Control) register.

29.3.2 CONFIGURING THE ADC:

Configure the ADC resolution, data alignment, and sample time.

29.3.3 SETTING THE CONVERSION SEQUENCE:

Set the conversion sequence to specify which channels are converted and in what order.

29.4 STARTING AN ADC CONVERSION:

To start a conversion, set the appropriate bits in the ADC control register.

29.5 READING ADC DATA:

After the conversion, read the converted data from the ADC data register.

29.6 USING INTERRUPTS OR DMA FOR ADC:

For continuous or high-speed data acquisition, consider using interrupts or DMA:

29.6.1 INTERRUPTS:

Configure the ADC to generate an interrupt upon completion of a conversion.

29.6.2 DMA:

Use DMA to transfer ADC data to memory automatically, freeing up the CPU for other tasks.

29.7 CALLIBRATION AND REFERENCE VOLTAGE:

Ensure the ADC is calibrated and understand how the reference voltage affects measurements. Calibration compensates for any offset and gain errors in the ADC.

29.8 TESTING AND VALIDATION:

29.8.1 ACCURACY:

Verify the accuracy of ADC readings against known voltage references.

29.8.2 NOISE REDUCTION:

Implement averaging, filtering, or oversampling to reduce noise and improve measurement stability.

29.8.3 SAMPLING RATE:

Ensure the sampling rate meets the requirements of your application, especially if the signal frequency is high.

CHAPTER#30 CODING AN ADC DRIVER FOR SIGNAL CONVERSION MODES

30.1CODING AN ADC DRIVER FOR SIGNAL CONVERSION MODES:

Developing an ADC driver to support various signal conversion modes involves configuring the ADC peripheral to handle different types of conversions, such as single-channel, multi-channel, continuous, and triggered conversions. These modes allow the ADC to be versatile in applications such as sensor data acquisition, signal monitoring, and more. Here's a guide to coding an ADC driver for different signal conversion modes in an ARM Cortex-M microcontroller, like the STM32 series.

30.2 UNDERSTANDING SIGNAL CONVERSION MODES:

30.2.1 SINGLE CONVERSION MODES:

The ADC converts a single channel once, then stops.

30.2.2 CONTINUOUS CONVERSION MODES:

The ADC continuously converts the selected channel(s) without stopping after a single conversion.

30.2.3 SCAN MODE:

The ADC converts multiple channels in sequence, typically used with multi-channel sensors.

30.2.4 DISCONTINUOUS MODE:

Similar to scan mode but with a pause between the conversion of each channel.

30.2.5 TRIGGERED CONVERSION:

Conversion starts in response to an external event or signal.

30.3 BASIC ADC INITILIZATION:

30.3.1 ENABLING ADC CLOCK:

First, ensure the ADC peripheral clock is enabled.

30.3.2 CONFIGURING ADC PARAMETERS:

Set the ADC resolution, data alignment, and sample time.

30.4 SETTING UP CONVERSION MODES:

30.4.1 SINGLE CONVERSION MODES:

This mode is useful for applications where data is required at specific times.

30.4.2 CONTINUOUS CONVERSION MODES:

In continuous mode, the ADC repeatedly converts the selected channel.

30.4.3 SCAN MODE (MULTI-CHANNEL CONVERSION):

In scan mode, the ADC scans multiple channels in sequence.

30.4.4 DISCONTINUOUS MODE:

In this mode, the ADC pauses between conversions in a sequence.

30.4.5 TRIGGERED CONVERSION:

Triggered mode starts conversion based on an external event.

30.5 HANDLING ADC DATA AND INTERRUPTS:

30.5.1DATA HANDLING:

Read the ADC conversion result from the data register after each conversion.

30.5.2 INTERRUPTS:

Use interrupts to handle end-of-conversion events, which is efficient for real-time applications.

30.6 OPTIMIZATION AND BEST PRACTICES:

30.6.1MINIMIZE ISR WORKLOAD:

Keep the interrupt service routine short and offload processing to the main loop or a lower-priority task.

30.6.2 USE DMA FOR LARGE DATA:

For applications requiring high sampling rates or large amounts of data, use Direct Memory Access (DMA) to transfer data without CPU intervention.

30.6.3 CALIBRATION AND REFERENCES VOLTAGE:

Ensure ADC calibration is performed and consider the reference voltage's accuracy, as it affects the ADC's measurement range and accuracy.

30.7 CODE OF MAIN.H:

```
/* USER CODE BEGIN Header */
/**
**********************************************************
********************
  * @file           : main.h
  * @brief          : Header for main.c file.
  *                   This file contains the common defines of the application.
**********************************************************
********************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
```

```
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H
#ifdef __cplusplus
extern "C" {
#endif
/* Includes ------------------------------------------------------------------*/
#include "stm32f4xx_hal.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */
/* USER CODE END EC */
/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */
```

```
/* USER CODE END EM */
/* Exported functions prototypes ---------------------------------------------*/
void Error_Handler(void);
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
/* Private defines -----------------------------------------------------------*/
#define DATA_Ready_Pin GPIO_PIN_2
#define DATA_Ready_GPIO_Port GPIOE
#define CS_I2C_SPI_Pin GPIO_PIN_3
#define CS_I2C_SPI_GPIO_Port GPIOE
#define INT1_Pin GPIO_PIN_4
#define INT1_GPIO_Port GPIOE
#define INT2_Pin GPIO_PIN_5
#define INT2_GPIO_Port GPIOE
#define PC14_OSC32_IN_Pin GPIO_PIN_14
#define PC14_OSC32_IN_GPIO_Port GPIOC
#define PC15_OSC32_OUT_Pin GPIO_PIN_15
#define PC15_OSC32_OUT_GPIO_Port GPIOC
#define PH0_OSC_IN_Pin GPIO_PIN_0
#define PH0_OSC_IN_GPIO_Port GPIOH
#define PH1_OSC_OUT_Pin GPIO_PIN_1
#define PH1_OSC_OUT_GPIO_Port GPIOH
#define OTG_FS_PowerSwitchOn_Pin GPIO_PIN_0
```

```
#define OTG_FS_PowerSwitchOn_GPIO_Port GPIOC
#define PDM_OUT_Pin GPIO_PIN_3
#define PDM_OUT_GPIO_Port GPIOC
#define I2S3_WS_Pin GPIO_PIN_4
#define I2S3_WS_GPIO_Port GPIOA
#define SPI1_SCK_Pin GPIO_PIN_5
#define SPI1_SCK_GPIO_Port GPIOA
#define SPI1_MISO_Pin GPIO_PIN_6
#define SPI1_MISO_GPIO_Port GPIOA
#define SPI1_MOSI_Pin GPIO_PIN_7
#define SPI1_MOSI_GPIO_Port GPIOA
#define CLK_IN_Pin GPIO_PIN_10
#define CLK_IN_GPIO_Port GPIOB
#define LD4_Pin GPIO_PIN_12
#define LD4_GPIO_Port GPIOD
#define LD3_Pin GPIO_PIN_13
#define LD3_GPIO_Port GPIOD
#define LD5_Pin GPIO_PIN_14
#define LD5_GPIO_Port GPIOD
#define LD6_Pin GPIO_PIN_15
#define LD6_GPIO_Port GPIOD
#define I2S3_MCK_Pin GPIO_PIN_7
#define I2S3_MCK_GPIO_Port GPIOC
```

```
#define VBUS_FS_Pin GPIO_PIN_9
#define VBUS_FS_GPIO_Port GPIOA
#define OTG_FS_ID_Pin GPIO_PIN_10
#define OTG_FS_ID_GPIO_Port GPIOA
#define OTG_FS_DM_Pin GPIO_PIN_11
#define OTG_FS_DM_GPIO_Port GPIOA
#define OTG_FS_DP_Pin GPIO_PIN_12
#define OTG_FS_DP_GPIO_Port GPIOA
#define SWDIO_Pin GPIO_PIN_13
#define SWDIO_GPIO_Port GPIOA
#define SWCLK_Pin GPIO_PIN_14
#define SWCLK_GPIO_Port GPIOA
#define I2S3_SCK_Pin GPIO_PIN_10
#define I2S3_SCK_GPIO_Port GPIOC
#define I2S3_SD_Pin GPIO_PIN_12
#define I2S3_SD_GPIO_Port GPIOC
#define Audio_RST_Pin GPIO_PIN_4
#define Audio_RST_GPIO_Port GPIOD
#define OTG_FS_OverCurrent_Pin GPIO_PIN_5
#define OTG_FS_OverCurrent_GPIO_Port GPIOD
#define SWO_Pin GPIO_PIN_3
#define SWO_GPIO_Port GPIOB
#define Audio_SCL_Pin GPIO_PIN_6
```

```
#define Audio_SCL_GPIO_Port GPIOB
#define Audio_SDA_Pin GPIO_PIN_9
#define Audio_SDA_GPIO_Port GPIOB
#define MEMS_INT2_Pin GPIO_PIN_1
#define MEMS_INT2_GPIO_Port GPIOE
/* USER CODE BEGIN Private defines */
/* USER CODE END Private defines */
#ifdef __cplusplus
}
#endif
#endif /* __MAIN_H */
```

30.8 CODE OF STM32F4XX_HAL_CONF.H:

```
/* USER CODE BEGIN Header */
/**
*******************************************************
*******************
  * @file    stm32f4xx_hal_conf_template.h
  * @author  MCD Application Team
  * @brief   HAL configuration template file.
  *          This file should be copied to the application folder and renamed
  *          to stm32f4xx_hal_conf.h.
*******************************************************
*******************
```

```
  * @attention
  *
  * Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*******************************************************************
*******************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_HAL_CONF_H
#define __STM32F4xx_HAL_CONF_H
#ifdef __cplusplus
 extern "C" {
#endif
/* Exported types ------------------------------------------------------------*/
/* Exported constants --------------------------------------------------------*/
/* ############################ Module Selection
################################## */
```

```
/**
  * @brief This is the list of modules to be used in the HAL driver
  */
#define HAL_MODULE_ENABLED
  /* #define HAL_CRYP_MODULE_ENABLED */
/* #define HAL_ADC_MODULE_ENABLED */
/* #define HAL_CAN_MODULE_ENABLED */
/* #define HAL_CRC_MODULE_ENABLED */
/* #define HAL_CAN_LEGACY_MODULE_ENABLED */
/* #define HAL_DAC_MODULE_ENABLED */
/* #define HAL_DCMI_MODULE_ENABLED */
/* #define HAL_DMA2D_MODULE_ENABLED */
/* #define HAL_ETH_MODULE_ENABLED */
/* #define HAL_ETH_LEGACY_MODULE_ENABLED */
/* #define HAL_NAND_MODULE_ENABLED */
/* #define HAL_NOR_MODULE_ENABLED */
/* #define HAL_PCCARD_MODULE_ENABLED */
/* #define HAL_SRAM_MODULE_ENABLED */
/* #define HAL_SDRAM_MODULE_ENABLED */
/* #define HAL_HASH_MODULE_ENABLED */
#define HAL_I2C_MODULE_ENABLED
#define HAL_I2S_MODULE_ENABLED
/* #define HAL_IWDG_MODULE_ENABLED */
```

```
/* #define HAL_LTDC_MODULE_ENABLED */
/* #define HAL_RNG_MODULE_ENABLED */
/* #define HAL_RTC_MODULE_ENABLED */
/* #define HAL_SAI_MODULE_ENABLED */
/* #define HAL_SD_MODULE_ENABLED */
/* #define HAL_MMC_MODULE_ENABLED */
#define HAL_SPI_MODULE_ENABLED
/* #define HAL_TIM_MODULE_ENABLED */
/* #define HAL_UART_MODULE_ENABLED */
/* #define HAL_USART_MODULE_ENABLED */
/* #define HAL_IRDA_MODULE_ENABLED */
/* #define HAL_SMARTCARD_MODULE_ENABLED */
/* #define HAL_SMBUS_MODULE_ENABLED */
/* #define HAL_WWDG_MODULE_ENABLED */
/* #define HAL_PCD_MODULE_ENABLED */
#define HAL_HCD_MODULE_ENABLED
/* #define HAL_DSI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_CEC_MODULE_ENABLED */
/* #define HAL_FMPI2C_MODULE_ENABLED */
/* #define HAL_FMPSMBUS_MODULE_ENABLED */
/* #define HAL_SPDIFRX_MODULE_ENABLED */
```

```
/* #define HAL_DFSDM_MODULE_ENABLED */
/* #define HAL_LPTIM_MODULE_ENABLED */
#define HAL_GPIO_MODULE_ENABLED
#define HAL_EXTI_MODULE_ENABLED
#define HAL_DMA_MODULE_ENABLED
#define HAL_RCC_MODULE_ENABLED
#define HAL_FLASH_MODULE_ENABLED
#define HAL_PWR_MODULE_ENABLED
#define HAL_CORTEX_MODULE_ENABLED
/* ############################ HSE/HSI Values adaptation
###################### */
/**
  * @brief Adjust the value of External High Speed oscillator (HSE)
used in your application.
  *        This value is used by the RCC HAL module to compute the
system frequency
  *        (when HSE is used as system clock source, directly or through
the PLL).
  */
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    8000000U /*!< Value of the External
oscillator in Hz */
#endif /* HSE_VALUE */
#if !defined  (HSE_STARTUP_TIMEOUT)
```

```
  #define HSE_STARTUP_TIMEOUT    100U   /*!< Time out for
HSE start up, in ms */
#endif /* HSE_STARTUP_TIMEOUT */
/**
  * @brief Internal High Speed oscillator (HSI) value.
  *        This value is used by the RCC HAL module to compute the
system frequency
  *        (when HSI is used as system clock source, directly or through the
PLL).
  */
#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000U) /*!< Value of the
Internal oscillator in Hz*/
#endif /* HSI_VALUE */
/**
  * @brief Internal Low Speed oscillator (LSI) value.
  */
#if !defined  (LSI_VALUE)
 #define LSI_VALUE  32000U       /*!< LSI Typical Value in Hz*/
#endif /* LSI_VALUE */                    /*!< Value of the Internal Low
Speed oscillator in Hz
                                 The real value may vary depending on the
variations
                                 in voltage and temperature.*/
/**
```

```
  * @brief External Low Speed oscillator (LSE) value.
  */
#if !defined  (LSE_VALUE)
 #define LSE_VALUE  32768U    /*!< Value of the External Low Speed
oscillator in Hz */
#endif /* LSE_VALUE */
#if !defined  (LSE_STARTUP_TIMEOUT)
  #define LSE_STARTUP_TIMEOUT    5000U   /*!< Time out for
LSE start up, in ms */
#endif /* LSE_STARTUP_TIMEOUT */
/**
  * @brief External clock source for I2S peripheral
  *       This value is used by the I2S HAL module to compute the I2S
clock source
  *       frequency, this source is inserted directly through I2S_CKIN
pad.
  */
#if !defined  (EXTERNAL_CLOCK_VALUE)
  #define EXTERNAL_CLOCK_VALUE    12288000U /*!< Value of
the External audio frequency in Hz*/
#endif /* EXTERNAL_CLOCK_VALUE */
/* Tip: To avoid modifying this file each time you need to use different
HSE,
   ===  you can define the HSE value in your toolchain compiler
preprocessor. */
```

```
/* ############################## System Configuration
########################### */

/**

  * @brief This is the HAL system configuration section

  */

#define  VDD_VALUE                    3300U /*!< Value of VDD in mv */

#define  TICK_INT_PRIORITY            0U   /*!< tick interrupt
priority */

#define  USE_RTOS                     0U

#define  PREFETCH_ENABLE              1U

#define  INSTRUCTION_CACHE_ENABLE     1U

#define  DATA_CACHE_ENABLE            1U

#define  USE_HAL_ADC_REGISTER_CALLBACKS        0U /*
ADC register callback disabled       */

#define  USE_HAL_CAN_REGISTER_CALLBACKS        0U /*
CAN register callback disabled       */

#define  USE_HAL_CEC_REGISTER_CALLBACKS        0U /*
CEC register callback disabled       */

#define  USE_HAL_CRYP_REGISTER_CALLBACKS       0U /*
CRYP register callback disabled      */

#define  USE_HAL_DAC_REGISTER_CALLBACKS        0U /*
DAC register callback disabled       */

#define  USE_HAL_DCMI_REGISTER_CALLBACKS       0U /*
DCMI register callback disabled      */

#define  USE_HAL_DFSDM_REGISTER_CALLBACKS      0U /*
DFSDM register callback disabled     */
```

#define USE_HAL_DMA2D_REGISTER_CALLBACKS 0U /* DMA2D register callback disabled */

#define USE_HAL_DSI_REGISTER_CALLBACKS 0U /* DSI register callback disabled */

#define USE_HAL_ETH_REGISTER_CALLBACKS 0U /* ETH register callback disabled */

#define USE_HAL_HASH_REGISTER_CALLBACKS 0U /* HASH register callback disabled */

#define USE_HAL_HCD_REGISTER_CALLBACKS 0U /* HCD register callback disabled */

#define USE_HAL_I2C_REGISTER_CALLBACKS 0U /* I2C register callback disabled */

#define USE_HAL_FMPI2C_REGISTER_CALLBACKS 0U /* FMPI2C register callback disabled */

#define USE_HAL_FMPSMBUS_REGISTER_CALLBACKS 0U /* FMPSMBUS register callback disabled */

#define USE_HAL_I2S_REGISTER_CALLBACKS 0U /* I2S register callback disabled */

#define USE_HAL_IRDA_REGISTER_CALLBACKS 0U /* IRDA register callback disabled */

#define USE_HAL_LPTIM_REGISTER_CALLBACKS 0U /* LPTIM register callback disabled */

#define USE_HAL_LTDC_REGISTER_CALLBACKS 0U /* LTDC register callback disabled */

#define USE_HAL_MMC_REGISTER_CALLBACKS 0U /* MMC register callback disabled */

#define USE_HAL_NAND_REGISTER_CALLBACKS 0U /* NAND register callback disabled */

#define USE_HAL_NOR_REGISTER_CALLBACKS 0U /* NOR register callback disabled */

#define USE_HAL_PCCARD_REGISTER_CALLBACKS 0U /* PCCARD register callback disabled */

#define USE_HAL_PCD_REGISTER_CALLBACKS 0U /* PCD register callback disabled */

#define USE_HAL_QSPI_REGISTER_CALLBACKS 0U /* QSPI register callback disabled */

#define USE_HAL_RNG_REGISTER_CALLBACKS 0U /* RNG register callback disabled */

#define USE_HAL_RTC_REGISTER_CALLBACKS 0U /* RTC register callback disabled */

#define USE_HAL_SAI_REGISTER_CALLBACKS 0U /* SAI register callback disabled */

#define USE_HAL_SD_REGISTER_CALLBACKS 0U /* SD register callback disabled */

#define USE_HAL_SMARTCARD_REGISTER_CALLBACKS 0U /* SMARTCARD register callback disabled */

#define USE_HAL_SDRAM_REGISTER_CALLBACKS 0U /* SDRAM register callback disabled */

#define USE_HAL_SRAM_REGISTER_CALLBACKS 0U /* SRAM register callback disabled */

#define USE_HAL_SPDIFRX_REGISTER_CALLBACKS 0U /* SPDIFRX register callback disabled */

#define USE_HAL_SMBUS_REGISTER_CALLBACKS 0U /* SMBUS register callback disabled */

#define USE_HAL_SPI_REGISTER_CALLBACKS 0U /* SPI register callback disabled */

#define USE_HAL_TIM_REGISTER_CALLBACKS 0U /* TIM register callback disabled */

#define USE_HAL_UART_REGISTER_CALLBACKS 0U /* UART register callback disabled */

#define USE_HAL_USART_REGISTER_CALLBACKS 0U /* USART register callback disabled */

#define USE_HAL_WWDG_REGISTER_CALLBACKS 0U /* WWDG register callback disabled */

/* ############################ Assert Selection ################################ */

/**

* @brief Uncomment the line below to expanse the "assert_param" macro in the

* HAL drivers code

*/

/* #define USE_FULL_ASSERT 1U */

/* ################### Ethernet peripheral configuration ###################### */

/* Section 1 : Ethernet peripheral configuration */

/* MAC ADDRESS: MAC_ADDR0:MAC_ADDR1:MAC_ADDR2:MAC_ADDR3:MAC_ADDR4:MAC_ADDR5 */

```
#define MAC_ADDR0   2U
#define MAC_ADDR1   0U
#define MAC_ADDR2   0U
#define MAC_ADDR3   0U
#define MAC_ADDR4   0U
#define MAC_ADDR5   0U
/* Definition of the Ethernet driver buffers size and count */
#define ETH_RX_BUF_SIZE          ETH_MAX_PACKET_SIZE
/* buffer size for receive          */
#define ETH_TX_BUF_SIZE          ETH_MAX_PACKET_SIZE
/* buffer size for transmit         */
#define ETH_RXBUFNB              4U      /* 4 Rx buffers of size
ETH_RX_BUF_SIZE  */
#define ETH_TXBUFNB              4U      /* 4 Tx buffers of size
ETH_TX_BUF_SIZE  */
/* Section 2: PHY configuration section */
/* DP83848_PHY_ADDRESS Address*/
#define DP83848_PHY_ADDRESS
/* PHY Reset delay these values are based on a 1 ms Systick interrupt*/
#define PHY_RESET_DELAY          0x000000FFU
/* PHY Configuration delay */
#define PHY_CONFIG_DELAY         0x00000FFFU
#define PHY_READ_TO              0x0000FFFFU
#define PHY_WRITE_TO             0x0000FFFFU
```

/* Section 3: Common PHY Registers */

#define PHY_BCR ((uint16_t)0x0000U) /*!< Transceiver Basic Control Register */

#define PHY_BSR ((uint16_t)0x0001U) /*!< Transceiver Basic Status Register */

#define PHY_RESET ((uint16_t)0x8000U) /*!< PHY Reset */

#define PHY_LOOPBACK ((uint16_t)0x4000U) /*!< Select loop-back mode */

#define PHY_FULLDUPLEX_100M ((uint16_t)0x2100U) /*!< Set the full-duplex mode at 100 Mb/s */

#define PHY_HALFDUPLEX_100M ((uint16_t)0x2000U) /*!< Set the half-duplex mode at 100 Mb/s */

#define PHY_FULLDUPLEX_10M ((uint16_t)0x0100U) /*!< Set the full-duplex mode at 10 Mb/s */

#define PHY_HALFDUPLEX_10M ((uint16_t)0x0000U) /*!< Set the half-duplex mode at 10 Mb/s */

#define PHY_AUTONEGOTIATION ((uint16_t)0x1000U) /*!< Enable auto-negotiation function */

#define PHY_RESTART_AUTONEGOTIATION ((uint16_t)0x0200U) /*!< Restart auto-negotiation function */

#define PHY_POWERDOWN ((uint16_t)0x0800U) /*!< Select the power down mode */

#define PHY_ISOLATE ((uint16_t)0x0400U) /*!< Isolate PHY from MII */

#define PHY_AUTONEGO_COMPLETE ((uint16_t)0x0020U) /*!< Auto-Negotiation process completed */

```
#define PHY_LINKED_STATUS            ((uint16_t)0x0004U)  /*!< Valid link established          */
#define PHY_JABBER_DETECTION         ((uint16_t)0x0002U)  /*!< Jabber condition detected       */
/* Section 4: Extended PHY Registers */
#define PHY_SR                       ((uint16_t))    /*!< PHY status register Offset                */
#define PHY_SPEED_STATUS             ((uint16_t))  /*!< PHY Speed mask                        */
#define PHY_DUPLEX_STATUS            ((uint16_t))  /*!< PHY Duplex mask                       */
/* ################### SPI peripheral configuration ############################ */
/* CRC FEATURE: Use to activate CRC feature inside HAL SPI Driver
* Activated: CRC code is present inside driver
* Deactivated: CRC code cleaned from driver
*/
#define USE_SPI_CRC                  0U
/* Includes ------------------------------------------------------------------*/
/**
  * @brief Include module's header file
  */
#ifdef HAL_RCC_MODULE_ENABLED
  #include "stm32f4xx_hal_rcc.h"
#endif /* HAL_RCC_MODULE_ENABLED */
```

```
#ifdef HAL_GPIO_MODULE_ENABLED
  #include "stm32f4xx_hal_gpio.h"
#endif /* HAL_GPIO_MODULE_ENABLED */
#ifdef HAL_EXTI_MODULE_ENABLED
  #include "stm32f4xx_hal_exti.h"
#endif /* HAL_EXTI_MODULE_ENABLED */
#ifdef HAL_DMA_MODULE_ENABLED
  #include "stm32f4xx_hal_dma.h"
#endif /* HAL_DMA_MODULE_ENABLED */
#ifdef HAL_CORTEX_MODULE_ENABLED
  #include "stm32f4xx_hal_cortex.h"
#endif /* HAL_CORTEX_MODULE_ENABLED */
#ifdef HAL_ADC_MODULE_ENABLED
  #include "stm32f4xx_hal_adc.h"
#endif /* HAL_ADC_MODULE_ENABLED */
#ifdef HAL_CAN_MODULE_ENABLED
  #include "stm32f4xx_hal_can.h"
#endif /* HAL_CAN_MODULE_ENABLED */
#ifdef HAL_CAN_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_can_legacy.h"
#endif /* HAL_CAN_LEGACY_MODULE_ENABLED */
#ifdef HAL_CRC_MODULE_ENABLED
  #include "stm32f4xx_hal_crc.h"
```

```
#endif /* HAL_CRC_MODULE_ENABLED */
#ifdef HAL_CRYP_MODULE_ENABLED
  #include "stm32f4xx_hal_cryp.h"
#endif /* HAL_CRYP_MODULE_ENABLED */
#ifdef HAL_DMA2D_MODULE_ENABLED
  #include "stm32f4xx_hal_dma2d.h"
#endif /* HAL_DMA2D_MODULE_ENABLED */
#ifdef HAL_DAC_MODULE_ENABLED
  #include "stm32f4xx_hal_dac.h"
#endif /* HAL_DAC_MODULE_ENABLED */
#ifdef HAL_DCMI_MODULE_ENABLED
  #include "stm32f4xx_hal_dcmi.h"
#endif /* HAL_DCMI_MODULE_ENABLED */
#ifdef HAL_ETH_MODULE_ENABLED
  #include "stm32f4xx_hal_eth.h"
#endif /* HAL_ETH_MODULE_ENABLED */
#ifdef HAL_ETH_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_eth_legacy.h"
#endif /* HAL_ETH_LEGACY_MODULE_ENABLED */
#ifdef HAL_FLASH_MODULE_ENABLED
  #include "stm32f4xx_hal_flash.h"
#endif /* HAL_FLASH_MODULE_ENABLED */
#ifdef HAL_SRAM_MODULE_ENABLED
```

```
  #include "stm32f4xx_hal_sram.h"
#endif /* HAL_SRAM_MODULE_ENABLED */
#ifdef HAL_NOR_MODULE_ENABLED
  #include "stm32f4xx_hal_nor.h"
#endif /* HAL_NOR_MODULE_ENABLED */
#ifdef HAL_NAND_MODULE_ENABLED
  #include "stm32f4xx_hal_nand.h"
#endif /* HAL_NAND_MODULE_ENABLED */
#ifdef HAL_PCCARD_MODULE_ENABLED
  #include "stm32f4xx_hal_pccard.h"
#endif /* HAL_PCCARD_MODULE_ENABLED */
#ifdef HAL_SDRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sdram.h"
#endif /* HAL_SDRAM_MODULE_ENABLED */
#ifdef HAL_HASH_MODULE_ENABLED
 #include "stm32f4xx_hal_hash.h"
#endif /* HAL_HASH_MODULE_ENABLED */
#ifdef HAL_I2C_MODULE_ENABLED
 #include "stm32f4xx_hal_i2c.h"
#endif /* HAL_I2C_MODULE_ENABLED */
#ifdef HAL_SMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_smbus.h"
#endif /* HAL_SMBUS_MODULE_ENABLED */
```

```
#ifdef HAL_I2S_MODULE_ENABLED
 #include "stm32f4xx_hal_i2s.h"
#endif /* HAL_I2S_MODULE_ENABLED */
#ifdef HAL_IWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_iwdg.h"
#endif /* HAL_IWDG_MODULE_ENABLED */
#ifdef HAL_LTDC_MODULE_ENABLED
 #include "stm32f4xx_hal_ltdc.h"
#endif /* HAL_LTDC_MODULE_ENABLED */
#ifdef HAL_PWR_MODULE_ENABLED
 #include "stm32f4xx_hal_pwr.h"
#endif /* HAL_PWR_MODULE_ENABLED */
#ifdef HAL_RNG_MODULE_ENABLED
 #include "stm32f4xx_hal_rng.h"
#endif /* HAL_RNG_MODULE_ENABLED */
#ifdef HAL_RTC_MODULE_ENABLED
 #include "stm32f4xx_hal_rtc.h"
#endif /* HAL_RTC_MODULE_ENABLED */
#ifdef HAL_SAI_MODULE_ENABLED
 #include "stm32f4xx_hal_sai.h"
#endif /* HAL_SAI_MODULE_ENABLED */
#ifdef HAL_SD_MODULE_ENABLED
 #include "stm32f4xx_hal_sd.h"
```

```
#endif /* HAL_SD_MODULE_ENABLED */
#ifdef HAL_SPI_MODULE_ENABLED
 #include "stm32f4xx_hal_spi.h"
#endif /* HAL_SPI_MODULE_ENABLED */
#ifdef HAL_TIM_MODULE_ENABLED
 #include "stm32f4xx_hal_tim.h"
#endif /* HAL_TIM_MODULE_ENABLED */
#ifdef HAL_UART_MODULE_ENABLED
 #include "stm32f4xx_hal_uart.h"
#endif /* HAL_UART_MODULE_ENABLED */
#ifdef HAL_USART_MODULE_ENABLED
 #include "stm32f4xx_hal_usart.h"
#endif /* HAL_USART_MODULE_ENABLED */
#ifdef HAL_IRDA_MODULE_ENABLED
 #include "stm32f4xx_hal_irda.h"
#endif /* HAL_IRDA_MODULE_ENABLED */
#ifdef HAL_SMARTCARD_MODULE_ENABLED
 #include "stm32f4xx_hal_smartcard.h"
#endif /* HAL_SMARTCARD_MODULE_ENABLED */
#ifdef HAL_WWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_wwdg.h"
#endif /* HAL_WWDG_MODULE_ENABLED */
#ifdef HAL_PCD_MODULE_ENABLED
```

```
 #include "stm32f4xx_hal_pcd.h"
#endif /* HAL_PCD_MODULE_ENABLED */
#ifdef HAL_HCD_MODULE_ENABLED
 #include "stm32f4xx_hal_hcd.h"
#endif /* HAL_HCD_MODULE_ENABLED */
#ifdef HAL_DSI_MODULE_ENABLED
 #include "stm32f4xx_hal_dsi.h"
#endif /* HAL_DSI_MODULE_ENABLED */
#ifdef HAL_QSPI_MODULE_ENABLED
 #include "stm32f4xx_hal_qspi.h"
#endif /* HAL_QSPI_MODULE_ENABLED */
#ifdef HAL_CEC_MODULE_ENABLED
 #include "stm32f4xx_hal_cec.h"
#endif /* HAL_CEC_MODULE_ENABLED */
#ifdef HAL_FMPI2C_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpi2c.h"
#endif /* HAL_FMPI2C_MODULE_ENABLED */
#ifdef HAL_FMPSMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpsmbus.h"
#endif /* HAL_FMPSMBUS_MODULE_ENABLED */
#ifdef HAL_SPDIFRX_MODULE_ENABLED
 #include "stm32f4xx_hal_spdifrx.h"
#endif /* HAL_SPDIFRX_MODULE_ENABLED */
```

```
#ifdef HAL_DFSDM_MODULE_ENABLED
 #include "stm32f4xx_hal_dfsdm.h"
#endif /* HAL_DFSDM_MODULE_ENABLED */
#ifdef HAL_LPTIM_MODULE_ENABLED
 #include "stm32f4xx_hal_lptim.h"
#endif /* HAL_LPTIM_MODULE_ENABLED */
#ifdef HAL_MMC_MODULE_ENABLED
 #include "stm32f4xx_hal_mmc.h"
#endif /* HAL_MMC_MODULE_ENABLED */
/* Exported macro ------------------------------------------------------------*/
#ifdef  USE_FULL_ASSERT
/**
  * @brief  The assert_param macro is used for function's parameters check.
  * @param  expr If expr is false, it calls assert_failed function
  *         which reports the name of the source file and the source
  *         line number of the call that failed.
  *         If expr is true, it returns no value.
  * @retval None
  */
  #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t *)__FILE__, __LINE__))
/* Exported functions ------------------------------------------------------- */
  void assert_failed(uint8_t* file, uint32_t line);
```

```
#else
  #define assert_param(expr) ((void)0U)
#endif /* USE_FULL_ASSERT */
#ifdef __cplusplus
}
#endif
#endif /* __STM32F4xx_HAL_CONF_H */
```

30.9 CODE OF STM32F4XX_IT.H:

```
/* USER CODE BEGIN Header */
/**
*****************************************************************************
  * @file    stm32f4xx_it.h
  * @brief   This file contains the headers of the interrupt handlers.
*****************************************************************************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file
  * in the root directory of this software component.
```

```
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_IT_H
#define __STM32F4xx_IT_H
#ifdef __cplusplus
 extern "C" {
#endif
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */
/* USER CODE END EC */
/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */
/* USER CODE END EM */
```

```
/* Exported functions prototypes ---------------------------------------------*/
void NMI_Handler(void);
void HardFault_Handler(void);
void MemManage_Handler(void);
void BusFault_Handler(void);
void UsageFault_Handler(void);
void SVC_Handler(void);
void DebugMon_Handler(void);
void PendSV_Handler(void);
void SysTick_Handler(void);
void OTG_FS_IRQHandler(void);
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
#ifdef __cplusplus
}
#endif
#endif /* __STM32F4xx_IT_H */
```

30.10 CODE OF MAIN.C:

```
/* USER CODE BEGIN Header */
/**
*******************************************************************
  * @file           : main.c
```

```
  * @brief          : Main program body
**********************************************************
*******************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "usb_host.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
```

```
/* USER CODE END PTD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
I2C_HandleTypeDef hi2c1;
I2S_HandleTypeDef hi2s2;
I2S_HandleTypeDef hi2s3;
SPI_HandleTypeDef hspi1;
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
void PeriphCommonClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_I2C1_Init(void);
static void MX_I2S2_Init(void);
static void MX_I2S3_Init(void);
static void MX_SPI1_Init(void);
void MX_USB_HOST_Process(void);
```

```
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */
  /* USER CODE END 1 */
  /* MCU Configuration--------------------------------------------------------*/
  /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
  HAL_Init();
  /* USER CODE BEGIN Init */
  /* USER CODE END Init */
  /* Configure the system clock */
  SystemClock_Config();
  /* Configure the peripherals common clocks */
  PeriphCommonClock_Config();
```

```
  /* USER CODE BEGIN SysInit */
  /* USER CODE END SysInit */
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_I2C1_Init();
  MX_I2S2_Init();
  MX_I2S3_Init();
  MX_SPI1_Init();
  MX_USB_HOST_Init();
  /* USER CODE BEGIN 2 */
  /* USER CODE END 2 */
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
    MX_USB_HOST_Process();
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
/**
  * @brief System Clock Configuration
```

```
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_
VOLTAGE_SCALE1);
  /** Initializes the RCC Oscillators according to the specified
parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType =
RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_BYPASS;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 4;
  RCC_OscInitStruct.PLL.PLLN = 192;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV4;
  RCC_OscInitStruct.PLL.PLLQ = 8;
```

```
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
 |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource =
RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct,
FLASH_LATENCY_3) != HAL_OK)
  {
    Error_Handler();
  }
}
/**
  * @brief Peripherals Common Clock Configuration
  * @retval None
  */
```

```
void PeriphCommonClock_Config(void)
{
  RCC_PeriphCLKInitTypeDef PeriphClkInitStruct = {0};
  /** Initializes the peripherals clock
  */
  PeriphClkInitStruct.PeriphClockSelection = RCC_PERIPHCLK_I2S;
  PeriphClkInitStruct.PLLI2S.PLLI2SN = 200;
  PeriphClkInitStruct.PLLI2S.PLLI2SM = 5;
  PeriphClkInitStruct.PLLI2S.PLLI2SR = 2;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInitStruct) !=
HAL_OK)
  {
    Error_Handler();
  }
}
/**
  * @brief I2C1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_I2C1_Init(void)
{
  /* USER CODE BEGIN I2C1_Init 0 */
```

```
  /* USER CODE END I2C1_Init 0 */
  /* USER CODE BEGIN I2C1_Init 1 */
  /* USER CODE END I2C1_Init 1 */
  hi2c1.Instance = I2C1;
  hi2c1.Init.ClockSpeed = 100000;
  hi2c1.Init.DutyCycle = I2C_DUTYCYCLE_2;
  hi2c1.Init.OwnAddress1 = 0;
  hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
  hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
  hi2c1.Init.OwnAddress2 = 0;
  hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
  hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
  if (HAL_I2C_Init(&hi2c1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN I2C1_Init 2 */
  /* USER CODE END I2C1_Init 2 */
}
/**
  * @brief I2S2 Initialization Function
  * @param None
  * @retval None
```

```
  */
static void MX_I2S2_Init(void)
{
  /* USER CODE BEGIN I2S2_Init 0 */
  /* USER CODE END I2S2_Init 0 */
  /* USER CODE BEGIN I2S2_Init 1 */
  /* USER CODE END I2S2_Init 1 */
  hi2s2.Instance = SPI2;
  hi2s2.Init.Mode = I2S_MODE_MASTER_TX;
  hi2s2.Init.Standard = I2S_STANDARD_PHILIPS;
  hi2s2.Init.DataFormat = I2S_DATAFORMAT_16B;
  hi2s2.Init.MCLKOutput = I2S_MCLKOUTPUT_DISABLE;
  hi2s2.Init.AudioFreq = I2S_AUDIOFREQ_96K;
  hi2s2.Init.CPOL = I2S_CPOL_LOW;
  hi2s2.Init.ClockSource = I2S_CLOCK_PLL;
  hi2s2.Init.FullDuplexMode = I2S_FULLDUPLEXMODE_ENABLE;
  if (HAL_I2S_Init(&hi2s2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN I2S2_Init 2 */
  /* USER CODE END I2S2_Init 2 */
}
```

```
/**
  * @brief I2S3 Initialization Function
  * @param None
  * @retval None
  */
static void MX_I2S3_Init(void)
{
  /* USER CODE BEGIN I2S3_Init 0 */
  /* USER CODE END I2S3_Init 0 */
  /* USER CODE BEGIN I2S3_Init 1 */
  /* USER CODE END I2S3_Init 1 */
  hi2s3.Instance = SPI3;
  hi2s3.Init.Mode = I2S_MODE_MASTER_TX;
  hi2s3.Init.Standard = I2S_STANDARD_PHILIPS;
  hi2s3.Init.DataFormat = I2S_DATAFORMAT_16B;
  hi2s3.Init.MCLKOutput = I2S_MCLKOUTPUT_ENABLE;
  hi2s3.Init.AudioFreq = I2S_AUDIOFREQ_96K;
  hi2s3.Init.CPOL = I2S_CPOL_LOW;
  hi2s3.Init.ClockSource = I2S_CLOCK_PLL;
  hi2s3.Init.FullDuplexMode = I2S_FULLDUPLEXMODE_DISABLE;
  if (HAL_I2S_Init(&hi2s3) != HAL_OK)
  {
    Error_Handler();
```

```
  }
  /* USER CODE BEGIN I2S3_Init 2 */
  /* USER CODE END I2S3_Init 2 */
}
/**
  * @brief SPI1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_SPI1_Init(void)
{
  /* USER CODE BEGIN SPI1_Init 0 */
  /* USER CODE END SPI1_Init 0 */
  /* USER CODE BEGIN SPI1_Init 1 */
  /* USER CODE END SPI1_Init 1 */
  /* SPI1 parameter configuration*/
  hspi1.Instance = SPI1;
  hspi1.Init.Mode = SPI_MODE_MASTER;
  hspi1.Init.Direction = SPI_DIRECTION_2LINES;
  hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
  hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
  hspi1.Init.NSS = SPI_NSS_SOFT;
```

```
  hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_2;
  hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
  hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
  hspi1.Init.CRCPolynomial = 10;
  if (HAL_SPI_Init(&hspi1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN SPI1_Init 2 */
  /* USER CODE END SPI1_Init 2 */
}
/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */
  /* GPIO Ports Clock Enable */
```

```
  __HAL_RCC_GPIOE_CLK_ENABLE();
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();
  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(CS_I2C_SPI_GPIO_Port, CS_I2C_SPI_Pin, GPIO_PIN_RESET);
  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(OTG_FS_PowerSwitchOn_GPIO_Port, OTG_FS_PowerSwitchOn_Pin, GPIO_PIN_SET);
  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOD, LD4_Pin|LD3_Pin|LD5_Pin|LD6_Pin
                    |Audio_RST_Pin, GPIO_PIN_RESET);
  /*Configure GPIO pin : DATA_Ready_Pin */
  GPIO_InitStruct.Pin = DATA_Ready_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(DATA_Ready_GPIO_Port, &GPIO_InitStruct);
  /*Configure GPIO pin : CS_I2C_SPI_Pin */
  GPIO_InitStruct.Pin = CS_I2C_SPI_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
```

```
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(CS_I2C_SPI_GPIO_Port, &GPIO_InitStruct);
  /*Configure GPIO pins : INT1_Pin INT2_Pin MEMS_INT2_Pin */
  GPIO_InitStruct.Pin = INT1_Pin|INT2_Pin|MEMS_INT2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);
  /*Configure GPIO pin : OTG_FS_PowerSwitchOn_Pin */
  GPIO_InitStruct.Pin = OTG_FS_PowerSwitchOn_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(OTG_FS_PowerSwitchOn_GPIO_Port,
&GPIO_InitStruct);
  /*Configure GPIO pin : PA0 */
  GPIO_InitStruct.Pin = GPIO_PIN_0;
  GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
  /*Configure GPIO pins : LD4_Pin LD3_Pin LD5_Pin LD6_Pin
                    Audio_RST_Pin */
  GPIO_InitStruct.Pin = LD4_Pin|LD3_Pin|LD5_Pin|LD6_Pin
```

```
                    |Audio_RST_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
  /*Configure GPIO pin : OTG_FS_OverCurrent_Pin */
  GPIO_InitStruct.Pin = OTG_FS_OverCurrent_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(OTG_FS_OverCurrent_GPIO_Port,
&GPIO_InitStruct);
/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}
/* USER CODE BEGIN 4 */
/* USER CODE END 4 */
/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
```

```
  /* User can add his own implementation to report the HAL error return
state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}
#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line
number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line
number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file,
line) */
```

```
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

30.11 CODE OF STM32F4XX_HAL_CONF.H:

```
/* USER CODE BEGIN Header */
/**
*****************************************************************
*******************
  * @file        stm32f4xx_hal_msp.c
  * @brief       This file provides code for the MSP Initialization
  *              and de-Initialization codes.
*****************************************************************
*******************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*****************************************************************
*******************
```

```
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN TD */
/* USER CODE END TD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN Define */
/* USER CODE END Define */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN Macro */
/* USER CODE END Macro */
/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* External functions --------------------------------------------------------*/
/* USER CODE BEGIN ExternalFunctions */
```

```
/* USER CODE END ExternalFunctions */
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/**
  * Initializes the Global MSP.
  */
void HAL_MspInit(void)
{
  /* USER CODE BEGIN MspInit 0 */
  /* USER CODE END MspInit 0 */
  __HAL_RCC_SYSCFG_CLK_ENABLE();
  __HAL_RCC_PWR_CLK_ENABLE();
  HAL_NVIC_SetPriorityGrouping(NVIC_PRIORITYGROUP_0);
  /* System interrupt init*/
  /* USER CODE BEGIN MspInit 1 */
  /* USER CODE END MspInit 1 */
}
/**
* @brief I2C MSP Initialization
* This function configures the hardware resources used in this example
* @param hi2c: I2C handle pointer
* @retval None
*/
```

```
void HAL_I2C_MspInit(I2C_HandleTypeDef* hi2c)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  if(hi2c->Instance==I2C1)
  {
  /* USER CODE BEGIN I2C1_MspInit 0 */
  /* USER CODE END I2C1_MspInit 0 */
    __HAL_RCC_GPIOB_CLK_ENABLE();
    /**I2C1 GPIO Configuration
    PB6     ------> I2C1_SCL
    PB9     ------> I2C1_SDA
    */
    GPIO_InitStruct.Pin = Audio_SCL_Pin|Audio_SDA_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_OD;
    GPIO_InitStruct.Pull = GPIO_PULLUP;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    GPIO_InitStruct.Alternate = GPIO_AF4_I2C1;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
    /* Peripheral clock enable */
    __HAL_RCC_I2C1_CLK_ENABLE();
  /* USER CODE BEGIN I2C1_MspInit 1 */
  /* USER CODE END I2C1_MspInit 1 */
  }
```

```
}
/**
* @brief I2C MSP De-Initialization
* This function freeze the hardware resources used in this example
* @param hi2c: I2C handle pointer
* @retval None
*/
void HAL_I2C_MspDeInit(I2C_HandleTypeDef* hi2c)
{
  if(hi2c->Instance==I2C1)
  {
  /* USER CODE BEGIN I2C1_MspDeInit 0 */
  /* USER CODE END I2C1_MspDeInit 0 */
    /* Peripheral clock disable */
    __HAL_RCC_I2C1_CLK_DISABLE();
    /**I2C1 GPIO Configuration
    PB6     ------> I2C1_SCL
    PB9     ------> I2C1_SDA
    */
    HAL_GPIO_DeInit(Audio_SCL_GPIO_Port, Audio_SCL_Pin);
    HAL_GPIO_DeInit(Audio_SDA_GPIO_Port, Audio_SDA_Pin);
  /* USER CODE BEGIN I2C1_MspDeInit 1 */
  /* USER CODE END I2C1_MspDeInit 1 */
```

```
  }
}
/**
* @brief I2S MSP Initialization
* This function configures the hardware resources used in this example
* @param hi2s: I2S handle pointer
* @retval None
*/
void HAL_I2S_MspInit(I2S_HandleTypeDef* hi2s)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  if(hi2s->Instance==SPI2)
  {
  /* USER CODE BEGIN SPI2_MspInit 0 */
  /* USER CODE END SPI2_MspInit 0 */
    /* Peripheral clock enable */
    __HAL_RCC_SPI2_CLK_ENABLE();
    __HAL_RCC_GPIOC_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();
    /**I2S2 GPIO Configuration
    PC2     ------> I2S2_ext_SD
    PC3     ------> I2S2_SD
    PB10     ------> I2S2_CK
```

```
  PB12     ------> I2S2_WS
  */
  GPIO_InitStruct.Pin = GPIO_PIN_2;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.Alternate = GPIO_AF6_I2S2ext;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
  GPIO_InitStruct.Pin = PDM_OUT_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.Alternate = GPIO_AF5_SPI2;
  HAL_GPIO_Init(PDM_OUT_GPIO_Port, &GPIO_InitStruct);
  GPIO_InitStruct.Pin = CLK_IN_Pin|GPIO_PIN_12;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.Alternate = GPIO_AF5_SPI2;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
/* USER CODE BEGIN SPI2_MspInit 1 */
/* USER CODE END SPI2_MspInit 1 */
}
```

```
else if(hi2s->Instance==SPI3)
{
/* USER CODE BEGIN SPI3_MspInit 0 */
/* USER CODE END SPI3_MspInit 0 */
  /* Peripheral clock enable */
  __HAL_RCC_SPI3_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOC_CLK_ENABLE();
  /**I2S3 GPIO Configuration
  PA4     ------> I2S3_WS
  PC7     ------> I2S3_MCK
  PC10     ------> I2S3_CK
  PC12     ------> I2S3_SD
  */
  GPIO_InitStruct.Pin = I2S3_WS_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.Alternate = GPIO_AF6_SPI3;
  HAL_GPIO_Init(I2S3_WS_GPIO_Port, &GPIO_InitStruct);
  GPIO_InitStruct.Pin = I2S3_MCK_Pin|I2S3_SCK_Pin|I2S3_SD_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
```

```
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    GPIO_InitStruct.Alternate = GPIO_AF6_SPI3;
    HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
  /* USER CODE BEGIN SPI3_MspInit 1 */
  /* USER CODE END SPI3_MspInit 1 */
  }
}
/**
* @brief I2S MSP De-Initialization
* This function freeze the hardware resources used in this example
* @param hi2s: I2S handle pointer
* @retval None
*/
void HAL_I2S_MspDeInit(I2S_HandleTypeDef* hi2s)
{
  if(hi2s->Instance==SPI2)
  {
  /* USER CODE BEGIN SPI2_MspDeInit 0 */
  /* USER CODE END SPI2_MspDeInit 0 */
    /* Peripheral clock disable */
    __HAL_RCC_SPI2_CLK_DISABLE();
    /**I2S2 GPIO Configuration
    PC2     ------> I2S2_ext_SD
```

```
  PC3     ------> I2S2_SD
  PB10    ------> I2S2_CK
  PB12    ------> I2S2_WS
  */
  HAL_GPIO_DeInit(GPIOC, GPIO_PIN_2|PDM_OUT_Pin);
  HAL_GPIO_DeInit(GPIOB, CLK_IN_Pin|GPIO_PIN_12);
/* USER CODE BEGIN SPI2_MspDeInit 1 */
/* USER CODE END SPI2_MspDeInit 1 */
}
else if(hi2s->Instance==SPI3)
{
/* USER CODE BEGIN SPI3_MspDeInit 0 */
/* USER CODE END SPI3_MspDeInit 0 */
  /* Peripheral clock disable */
  __HAL_RCC_SPI3_CLK_DISABLE();
  /**I2S3 GPIO Configuration
  PA4     ------> I2S3_WS
  PC7     ------> I2S3_MCK
  PC10    ------> I2S3_CK
  PC12    ------> I2S3_SD
  */
  HAL_GPIO_DeInit(I2S3_WS_GPIO_Port, I2S3_WS_Pin);
```

```
    HAL_GPIO_DeInit(GPIOC,
I2S3_MCK_Pin|I2S3_SCK_Pin|I2S3_SD_Pin);
  /* USER CODE BEGIN SPI3_MspDeInit 1 */
  /* USER CODE END SPI3_MspDeInit 1 */
  }
}
/**
* @brief SPI MSP Initialization
* This function configures the hardware resources used in this example
* @param hspi: SPI handle pointer
* @retval None
*/
void HAL_SPI_MspInit(SPI_HandleTypeDef* hspi)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  if(hspi->Instance==SPI1)
  {
  /* USER CODE BEGIN SPI1_MspInit 0 */
  /* USER CODE END SPI1_MspInit 0 */
    /* Peripheral clock enable */
    __HAL_RCC_SPI1_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();
    /**SPI1 GPIO Configuration
```

```
    PA5     ------> SPI1_SCK
    PA6     ------> SPI1_MISO
    PA7     ------> SPI1_MOSI
    */
    GPIO_InitStruct.Pin =
SPI1_SCK_Pin|SPI1_MISO_Pin|SPI1_MOSI_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF5_SPI1;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
  /* USER CODE BEGIN SPI1_MspInit 1 */
  /* USER CODE END SPI1_MspInit 1 */
  }
}
/**
* @brief SPI MSP De-Initialization
* This function freeze the hardware resources used in this example
* @param hspi: SPI handle pointer
* @retval None
*/
void HAL_SPI_MspDeInit(SPI_HandleTypeDef* hspi)
{
```

```
  if(hspi->Instance==SPI1)
  {
  /* USER CODE BEGIN SPI1_MspDeInit 0 */
  /* USER CODE END SPI1_MspDeInit 0 */
    /* Peripheral clock disable */
    __HAL_RCC_SPI1_CLK_DISABLE();
    /**SPI1 GPIO Configuration
    PA5     ------> SPI1_SCK
    PA6     ------> SPI1_MISO
    PA7     ------> SPI1_MOSI
    */
    HAL_GPIO_DeInit(GPIOA,
SPI1_SCK_Pin|SPI1_MISO_Pin|SPI1_MOSI_Pin);
  /* USER CODE BEGIN SPI1_MspDeInit 1 */
  /* USER CODE END SPI1_MspDeInit 1 */
  }
}
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

30.12 CODE OF SYSCALLS.C:

```
/**
*****************************************************************
*******************
 * @file      syscalls.c
```

```
 * @author    Auto-generated by STM32CubeIDE
 * @brief     STM32CubeIDE Minimal System calls file
 *
 *          For more information about which c-functions
 *          need which of these lowlevel functions
 *          please consult the Newlib libc-manual
**********************************************************
********************
 * @attention
 *
 * Copyright (c) 2020-2024 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the
LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
**********************************************************
********************
 */
/* Includes */
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
```

```
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
#include <sys/times.h>
/* Variables */
extern int __io_putchar(int ch) __attribute__((weak));
extern int __io_getchar(void) __attribute__((weak));
char *__env[1] = { 0 };
char **environ = __env;
/* Functions */
void initialise_monitor_handles()
{
}
int _getpid(void)
{
  return 1;
}
int _kill(int pid, int sig)
{
  (void)pid;
  (void)sig;
  errno = EINVAL;
```

```
  return -1;
}
void _exit (int status)
{
  _kill(status, -1);
  while (1) {}    /* Make sure we hang here */
}
__attribute__((weak)) int _read(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;
  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    *ptr++ = __io_getchar();
  }
  return len;
}
__attribute__((weak)) int _write(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;
  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
```

```
    __io_putchar(*ptr++);
  }
  return len;
}
int _close(int file)
{
  (void)file;
  return -1;
}
int _fstat(int file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
int _isatty(int file)
{
  (void)file;
  return 1;
}
int _lseek(int file, int ptr, int dir)
{
  (void)file;
```

```
  (void)ptr;
  (void)dir;
  return 0;
}
int _open(char *path, int flags, ...)
{
  (void)path;
  (void)flags;
  /* Pretend like we always fail */
  return -1;
}
int _wait(int *status)
{
  (void)status;
  errno = ECHILD;
  return -1;
}
int _unlink(char *name)
{
  (void)name;
  errno = ENOENT;
  return -1;
}
```

```
int _times(struct tms *buf)
{
  (void)buf;
  return -1;
}
int _stat(char *file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
int _link(char *old, char *new)
{
  (void)old;
  (void)new;
  errno = EMLINK;
  return -1;
}
int _fork(void)
{
  errno = EAGAIN;
  return -1;
}
```

```
int _execve(char *name, char **argv, char **env)
{
  (void)name;
  (void)argv;
  (void)env;
  errno = ENOMEM;
  return -1;
}
```

30.13 CODE OF SYSMEM.C:

```
/**
*****************************************************************
 * @file      sysmem.c
 * @author    Generated by STM32CubeIDE
 * @brief     STM32CubeIDE System Memory calls file
 *
 *            For more information about which C functions
 *            need which of these lowlevel functions
 *            please consult the newlib libc manual
*****************************************************************
 * @attention
 *
```

```
 * Copyright (c) 2024 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the
LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
**********************************************************
********************
 */
/* Includes */
#include <errno.h>
#include <stdint.h>
/**
 * Pointer to the current high watermark of the heap usage
 */
static uint8_t *__sbrk_heap_end = NULL;
/**
 * @brief _sbrk() allocates memory to the newlib heap and is used by
malloc
 *        and others from the C library
 *
 * @verbatim
```

```
 *
##################################################################
 * #  .data  #  .bss  #       newlib heap       #          MSP stack          #
 * #        #        #                          # Reserved by _Min_Stack_Size #
 *
##################################################################
 * ^-- RAM start      ^-- _end                              _estack, RAM end --^
 * @endverbatim
 *
 * This implementation starts allocating at the '_end' linker symbol
 * The '_Min_Stack_Size' linker symbol reserves a memory for the MSP stack
 * The implementation considers '_estack' linker symbol to be RAM end
 * NOTE: If the MSP stack, at any point during execution, grows larger than the
 * reserved size, please increase the '_Min_Stack_Size'.
 *
 * @param incr Memory size
 * @return Pointer to allocated memory
 */
void *_sbrk(ptrdiff_t incr)
{
  extern uint8_t _end; /* Symbol defined in the linker script */
```

```
  extern uint8_t _estack; /* Symbol defined in the linker script */
  extern uint32_t _Min_Stack_Size; /* Symbol defined in the linker
script */
  const uint32_t stack_limit = (uint32_t)&_estack -
(uint32_t)&_Min_Stack_Size;
  const uint8_t *max_heap = (uint8_t *)stack_limit;
  uint8_t *prev_heap_end;
  /* Initialize heap end at first call */
  if (NULL == __sbrk_heap_end)
  {
    __sbrk_heap_end = &_end;
  }
  /* Protect heap from growing into the reserved MSP stack */
  if (__sbrk_heap_end + incr > max_heap)
  {
    errno = ENOMEM;
    return (void *)-1;
  }
  prev_heap_end = __sbrk_heap_end;
  __sbrk_heap_end += incr;
  return (void *)prev_heap_end;
}
```

30.14 CODE OF SYSTEM_STM32F4XX.C:

```
/**
*****************************************************************
*******************

  * @file    system_stm32f4xx.c
  * @author  MCD Application Team
  * @brief   CMSIS Cortex-M4 Device Peripheral Access Layer System
Source File.
  *
  *   This file provides two functions and one global variable to be called
from
  *   user application:
  *      - SystemInit(): This function is called at startup just after reset and
  *                      before branch to main program. This call is made inside
  *                      the "startup_stm32f4xx.s" file.
  *
  *      - SystemCoreClock variable: Contains the core clock (HCLK), it
can be used
  *                              by the user application to setup the SysTick
  *                              timer or configure other parameters.
  *
  *      - SystemCoreClockUpdate(): Updates the variable
SystemCoreClock and must
  *                             be called whenever the core clock is changed
  *                             during program execution.
```

```
  *
  *
*****************************************************************
*******************
  * @attention
  *
  * Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*****************************************************************
*******************
  */
/** @addtogroup CMSIS
  * @{
  */
/** @addtogroup stm32f4xx_system
  * @{
  */
/** @addtogroup STM32F4xx_System_Private_Includes
  * @{
```

```
  */
#include "stm32f4xx.h"
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    ((uint32_t)25000000) /*!< Default value of
the External oscillator in Hz */
#endif /* HSE_VALUE */
#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000) /*!< Value of the
Internal oscillator in Hz*/
#endif /* HSI_VALUE */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_TypesDefinitions
  * @{
  */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Defines
  * @{
  */
/************************* Miscellaneous Configuration
************************/
```

/*!< Uncomment the following line if you need to use external SRAM or SDRAM as data memory */

#if defined(STM32F405xx) || defined(STM32F415xx) || defined(STM32F407xx) || defined(STM32F417xx)\

 || defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) || defined(STM32F439xx)\

 || defined(STM32F469xx) || defined(STM32F479xx) || defined(STM32F412Zx) || defined(STM32F412Vx)

/* #define DATA_IN_ExtSRAM */

#endif /* STM32F40xxx || STM32F41xxx || STM32F42xxx || STM32F43xxx || STM32F469xx || STM32F479xx ||\

STM32F412Zx || STM32F412Vx */

#if defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) || defined(STM32F439xx)\

 || defined(STM32F446xx) || defined(STM32F469xx) || defined(STM32F479xx)

/* #define DATA_IN_ExtSDRAM */

#endif /* STM32F427xx || STM32F437xx || STM32F429xx || STM32F439xx || STM32F446xx || STM32F469xx ||\

STM32F479xx */

/* Note: Following vector table addresses must be defined in line with linker

configuration. */

/*!< Uncomment the following line if you need to relocate the vector table

anywhere in Flash or Sram, else the vector table is kept at the automatic

remap of boot address selected */

/* #define USER_VECT_TAB_ADDRESS */

#if defined(USER_VECT_TAB_ADDRESS)

/*!< Uncomment the following line if you need to relocate your vector Table

in Sram else user remap will be done in Flash. */

/* #define VECT_TAB_SRAM */

#if defined(VECT_TAB_SRAM)

#define VECT_TAB_BASE_ADDRESS SRAM_BASE /*!< Vector Table base address field.

This value must be a multiple of 0x200. */

#define VECT_TAB_OFFSET 0x00000000U /*!< Vector Table base offset field.

This value must be a multiple of 0x200. */

#else

#define VECT_TAB_BASE_ADDRESS FLASH_BASE /*!< Vector Table base address field.

This value must be a multiple of 0x200. */

#define VECT_TAB_OFFSET 0x00000000U /*!< Vector Table base offset field.

```
                                        This value must be a multiple of 0x200.
*/
#endif /* VECT_TAB_SRAM */
#endif /* USER_VECT_TAB_ADDRESS */
/*****************************************************************************/
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Macros
  * @{
  */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Variables
  * @{
  */
  /* This variable is updated in three ways:
     1) by calling CMSIS function SystemCoreClockUpdate()
     2) by calling HAL API function HAL_RCC_GetHCLKFreq()
     3) each time HAL_RCC_ClockConfig() is called to configure the system clock frequency
```

```
        Note: If you use this function to configure the system clock; then there
              is no need to call the 2 first functions listed above, since SystemCoreClock
              variable is updated automatically.
  */
uint32_t SystemCoreClock = 16000000;
const uint8_t AHBPrescTable[16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9};
const uint8_t APBPrescTable[8]  = {0, 0, 0, 0, 1, 2, 3, 4};
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_FunctionPrototypes
  * @{
  */
#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  static void SystemInit_ExtMemCtl(void);
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Functions
  * @{
```

```
  */
/**
  * @brief  Setup the microcontroller system
  *         Initialize the FPU setting, vector table location and External memory
  *         configuration.
  * @param  None
  * @retval None
  */
void SystemInit(void)
{
  /* FPU settings ------------------------------------------------------------*/
  #if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
    SCB->CPACR |= ((3UL << 10*2)|(3UL << 11*2));  /* set CP10 and CP11 Full Access */
  #endif
#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  SystemInit_ExtMemCtl();
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */
  /* Configure the Vector Table location -------------------------------------*/
#if defined(USER_VECT_TAB_ADDRESS)
  SCB->VTOR = VECT_TAB_BASE_ADDRESS | VECT_TAB_OFFSET; /* Vector Table Relocation in Internal SRAM */
#endif /* USER_VECT_TAB_ADDRESS */
```

```
}
/**
  * @brief  Update SystemCoreClock variable according to Clock
Register Values.
  *         The SystemCoreClock variable contains the core clock (HCLK),
it can
  *         be used by the user application to setup the SysTick timer or
configure
  *         other parameters.
  *
  * @note   Each time the core clock (HCLK) changes, this function
must be called
  *         to update SystemCoreClock variable value. Otherwise, any
configuration
  *         based on this variable will be incorrect.
  *
  * @note   - The system frequency computed by this function is not the
real
  *           frequency in the chip. It is calculated based on the predefined
  *           constant and the selected clock source:
  *
  *           - If SYSCLK source is HSI, SystemCoreClock will contain the
HSI_VALUE(*)
  *
```

```
  *          - If SYSCLK source is HSE, SystemCoreClock will contain the
HSE_VALUE(**)
  *
  *          - If SYSCLK source is PLL, SystemCoreClock will contain the
HSE_VALUE(**)
  *            or HSI_VALUE(*) multiplied/divided by the PLL factors.
  *
  *        (*) HSI_VALUE is a constant defined in stm32f4xx_hal_conf.h
file (default value
  *            16 MHz) but the real value may vary depending on the
variations
  *            in voltage and temperature.
  *
  *        (**) HSE_VALUE is a constant defined in
stm32f4xx_hal_conf.h file (its value
  *             depends on the application requirements), user has to ensure
that HSE_VALUE
  *             is same as the real frequency of the crystal used. Otherwise,
this function
  *             may have wrong result.
  *
  *        - The result of this function could be not correct when using
fractional
  *          value for HSE crystal.
  *
```

```
  * @param  None
  * @retval None
  */
void SystemCoreClockUpdate(void)
{
  uint32_t tmp = 0, pllvco = 0, pllp = 2, pllsource = 0, pllm = 2;
  /* Get SYSCLK source -------------------------------------------------------
*/
  tmp = RCC->CFGR & RCC_CFGR_SWS;
  switch (tmp)
  {
    case 0x00:  /* HSI used as system clock source */
      SystemCoreClock = HSI_VALUE;
      break;
    case 0x04:  /* HSE used as system clock source */
      SystemCoreClock = HSE_VALUE;
      break;
    case 0x08:  /* PLL used as system clock source */
      /* PLL_VCO = (HSE_VALUE or HSI_VALUE / PLL_M) * PLL_N
         SYSCLK = PLL_VCO / PLL_P
         */
      pllsource = (RCC->PLLCFGR & RCC_PLLCFGR_PLLSRC) >>
22;
      pllm = RCC->PLLCFGR & RCC_PLLCFGR_PLLM;
```

```
    if (pllsource != 0)
    {
      /* HSE used as PLL clock source */
      pllvco = (HSE_VALUE / pllm) * ((RCC->PLLCFGR &
RCC_PLLCFGR_PLLN) >> 6);
    }
    else
    {
      /* HSI used as PLL clock source */
      pllvco = (HSI_VALUE / pllm) * ((RCC->PLLCFGR &
RCC_PLLCFGR_PLLN) >> 6);
    }
    pllp = (((RCC->PLLCFGR & RCC_PLLCFGR_PLLP) >>16) + 1 )
*2;
    SystemCoreClock = pllvco/pllp;
    break;
   default:
    SystemCoreClock = HSI_VALUE;
    break;
  }
  /* Compute HCLK frequency ----------------------------------------------
-*/
  /* Get HCLK prescaler */
  tmp = AHBPrescTable[((RCC->CFGR & RCC_CFGR_HPRE) >> 4)];
```

```
  /* HCLK frequency */
  SystemCoreClock >>= tmp;
}
#if defined (DATA_IN_ExtSRAM) && defined
(DATA_IN_ExtSDRAM)
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
/**
  * @brief  Setup the external memory controller.
  *         Called in startup_stm32f4xx.s before jump to main.
  *         This function configures the external memories
(SRAM/SDRAM)
  *         This SRAM/SDRAM will be used as program data memory
(including heap and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
  register __IO uint32_t index;
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and
GPIOI interface clock */
```

```
RCC->AHB1ENR |= 0x000001F8;
/* Delay after an RCC peripheral clock enabling */
tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);
/* Connect PDx pins to FMC Alternate function */
GPIOD->AFR[0]  = 0x00CCC0CC;
GPIOD->AFR[1]  = 0xCCCCCCCC;
/* Configure PDx pins in Alternate function mode */
GPIOD->MODER   = 0xAAAA0A8A;
/* Configure PDx pins speed to 100 MHz */
GPIOD->OSPEEDR = 0xFFFF0FCF;
/* Configure PDx pins Output type to push-pull */
GPIOD->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PDx pins */
GPIOD->PUPDR   = 0x00000000;
/* Connect PEx pins to FMC Alternate function */
GPIOE->AFR[0]  = 0xC00CC0CC;
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA828A;
/* Configure PEx pins speed to 100 MHz */
GPIOE->OSPEEDR = 0xFFFFC3CF;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
```

```
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0xCCCCCCCC;
GPIOF->AFR[1]  = 0xCCCCCCCC;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA800AAA;
/* Configure PFx pins speed to 50 MHz */
GPIOF->OSPEEDR = 0xAA800AAA;
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
GPIOF->PUPDR   = 0x00000000;
/* Connect PGx pins to FMC Alternate function */
GPIOG->AFR[0]  = 0xCCCCCCCC;
GPIOG->AFR[1]  = 0xCCCCCCCC;
/* Configure PGx pins in Alternate function mode */
GPIOG->MODER   = 0xAAAAAAAA;
/* Configure PGx pins speed to 50 MHz */
GPIOG->OSPEEDR = 0xAAAAAAAA;
/* Configure PGx pins Output type to push-pull */
GPIOG->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PGx pins */
```

```
GPIOG->PUPDR   = 0x00000000;
/* Connect PHx pins to FMC Alternate function */
GPIOH->AFR[0]  = 0x00C0CC00;
GPIOH->AFR[1]  = 0xCCCCCCCC;
/* Configure PHx pins in Alternate function mode */
GPIOH->MODER   = 0xAAAA08A0;
/* Configure PHx pins speed to 50 MHz */
GPIOH->OSPEEDR = 0xAAAA08A0;
/* Configure PHx pins Output type to push-pull */
GPIOH->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PHx pins */
GPIOH->PUPDR   = 0x00000000;
/* Connect PIx pins to FMC Alternate function */
GPIOI->AFR[0]  = 0xCCCCCCCC;
GPIOI->AFR[1]  = 0x00000CC0;
/* Configure PIx pins in Alternate function mode */
GPIOI->MODER   = 0x0028AAAA;
/* Configure PIx pins speed to 50 MHz */
GPIOI->OSPEEDR = 0x0028AAAA;
/* Configure PIx pins Output type to push-pull */
GPIOI->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PIx pins */
GPIOI->PUPDR   = 0x00000000;
```

```
/*-- FMC Configuration ------------------------------------------------------*/
  /* Enable the FMC interface clock */
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
  FMC_Bank5_6->SDTR[0] = 0x01115351;
  /* SDRAM initialization sequence */
  /* Clock enable command */
  FMC_Bank5_6->SDCMR = 0x00000011;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Delay */
  for (index = 0; index<1000; index++);
  /* PALL command */
  FMC_Bank5_6->SDCMR = 0x00000012;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
```

```
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* Auto refresh command */
FMC_Bank5_6->SDCMR = 0x00000073;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* MRD register program */
FMC_Bank5_6->SDCMR = 0x00046014;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* Set refresh count */
tmpreg = FMC_Bank5_6->SDRTR;
FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
/* Disable write protection */
tmpreg = FMC_Bank5_6->SDCR[0];
```

```
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F469xx || STM32F479xx */
  (void)(tmp);
}
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F469xx || STM32F479xx */
#elif defined (DATA_IN_ExtSRAM) || defined
(DATA_IN_ExtSDRAM)
/**
  * @brief  Setup the external memory controller.
  *         Called in startup_stm32f4xx.s before jump to main.
```

```
  *        This function configures the external memories
(SRAM/SDRAM)
  *        This SRAM/SDRAM will be used as program data memory
(including heap and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F446xx) || defined(STM32F469xx) ||
defined(STM32F479xx)
#if defined (DATA_IN_ExtSDRAM)
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
  register __IO uint32_t index;
#if defined(STM32F446xx)
  /* Enable GPIOA, GPIOC, GPIOD, GPIOE, GPIOF, GPIOG interface
     clock */
  RCC->AHB1ENR |= 0x0000007D;
#else
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and
GPIOI interface
     clock */
```

```
  RCC->AHB1ENR |= 0x000001F8;
#endif /* STM32F446xx */
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);
#if defined(STM32F446xx)
  /* Connect PAx pins to FMC Alternate function */
  GPIOA->AFR[0]  |= 0xC0000000;
  GPIOA->AFR[1]  |= 0x00000000;
  /* Configure PDx pins in Alternate function mode */
  GPIOA->MODER   |= 0x00008000;
  /* Configure PDx pins speed to 50 MHz */
  GPIOA->OSPEEDR |= 0x00008000;
  /* Configure PDx pins Output type to push-pull */
  GPIOA->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOA->PUPDR   |= 0x00000000;
  /* Connect PCx pins to FMC Alternate function */
  GPIOC->AFR[0]  |= 0x00CC0000;
  GPIOC->AFR[1]  |= 0x00000000;
  /* Configure PDx pins in Alternate function mode */
  GPIOC->MODER   |= 0x00000A00;
  /* Configure PDx pins speed to 50 MHz */
  GPIOC->OSPEEDR |= 0x00000A00;
```

```
  /* Configure PDx pins Output type to push-pull */
  GPIOC->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOC->PUPDR   |= 0x00000000;
#endif /* STM32F446xx */
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x000000CC;
  GPIOD->AFR[1]  = 0xCC000CCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xA02A000A;
  /* Configure PDx pins speed to 50 MHz */
  GPIOD->OSPEEDR = 0xA02A000A;
  /* Configure PDx pins Output type to push-pull */
  GPIOD->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOD->PUPDR   = 0x00000000;
  /* Connect PEx pins to FMC Alternate function */
  GPIOE->AFR[0]  = 0xC00000CC;
  GPIOE->AFR[1]  = 0xCCCCCCCC;
  /* Configure PEx pins in Alternate function mode */
  GPIOE->MODER   = 0xAAAA800A;
  /* Configure PEx pins speed to 50 MHz */
  GPIOE->OSPEEDR = 0xAAAA800A;
```

```
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0xCCCCCCCC;
GPIOF->AFR[1]  = 0xCCCCCCCC;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA800AAA;
/* Configure PFx pins speed to 50 MHz */
GPIOF->OSPEEDR = 0xAA800AAA;
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
GPIOF->PUPDR   = 0x00000000;
/* Connect PGx pins to FMC Alternate function */
GPIOG->AFR[0]  = 0xCCCCCCCC;
GPIOG->AFR[1]  = 0xCCCCCCCC;
/* Configure PGx pins in Alternate function mode */
GPIOG->MODER   = 0xAAAAAAAA;
/* Configure PGx pins speed to 50 MHz */
GPIOG->OSPEEDR = 0xAAAAAAAA;
/* Configure PGx pins Output type to push-pull */
```

```
  GPIOG->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
  /* Connect PHx pins to FMC Alternate function */
  GPIOH->AFR[0]  = 0x00C0CC00;
  GPIOH->AFR[1]  = 0xCCCCCCCC;
  /* Configure PHx pins in Alternate function mode */
  GPIOH->MODER   = 0xAAAA08A0;
  /* Configure PHx pins speed to 50 MHz */
  GPIOH->OSPEEDR = 0xAAAA08A0;
  /* Configure PHx pins Output type to push-pull */
  GPIOH->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PHx pins */
  GPIOH->PUPDR   = 0x00000000;
  /* Connect PIx pins to FMC Alternate function */
  GPIOI->AFR[0]  = 0xCCCCCCCC;
  GPIOI->AFR[1]  = 0x00000CC0;
  /* Configure PIx pins in Alternate function mode */
  GPIOI->MODER   = 0x0028AAAA;
  /* Configure PIx pins speed to 50 MHz */
```

```
  GPIOI->OSPEEDR = 0x0028AAAA;
  /* Configure PIx pins Output type to push-pull */
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F469xx || STM32F479xx */
/*-- FMC Configuration -------------------------------------------------------*/
  /* Enable the FMC interface clock */
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable SDRAM bank1 */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCR[0] = 0x00001954;
#else
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
#endif /* STM32F446xx */
  FMC_Bank5_6->SDTR[0] = 0x01115351;
  /* SDRAM initialization sequence */
  /* Clock enable command */
  FMC_Bank5_6->SDCMR = 0x00000011;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
```

```
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Delay */
  for (index = 0; index<1000; index++);
  /* PALL command */
  FMC_Bank5_6->SDCMR = 0x00000012;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Auto refresh command */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x000000F3;
#else
  FMC_Bank5_6->SDCMR = 0x00000073;
#endif /* STM32F446xx */
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
```

```
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* MRD register program */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x00044014;
#else
  FMC_Bank5_6->SDCMR = 0x00046014;
#endif /* STM32F446xx */
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Set refresh count */
  tmpreg = FMC_Bank5_6->SDRTR;
#if defined(STM32F446xx)
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000050C<<1));
#else
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
#endif /* STM32F446xx */
  /* Disable write protection */
```

```
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#endif /* DATA_IN_ExtSDRAM */
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F446xx || STM32F469xx || STM32F479xx */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx) || defined(STM32F417xx)\
 || defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx) ||
defined(STM32F412Zx) || defined(STM32F412Vx)
#if defined(DATA_IN_ExtSRAM)
/*-- GPIOs Configuration -----------------------------------------------------*/
   /* Enable GPIOD, GPIOE, GPIOF and GPIOG interface clock */
  RCC->AHB1ENR   |= 0x00000078;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIODEN);
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x00CCC0CC;
  GPIOD->AFR[1]  = 0xCCCCCCCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xAAAA0A8A;
  /* Configure PDx pins speed to 100 MHz */
  GPIOD->OSPEEDR = 0xFFFF0FCF;
```

```
/* Configure PDx pins Output type to push-pull */
GPIOD->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PDx pins */
GPIOD->PUPDR   = 0x00000000;
/* Connect PEx pins to FMC Alternate function */
GPIOE->AFR[0]  = 0xC00CC0CC;
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA828A;
/* Configure PEx pins speed to 100 MHz */
GPIOE->OSPEEDR = 0xFFFFC3CF;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0x00CCCCCC;
GPIOF->AFR[1]  = 0xCCCC0000;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA000AAA;
/* Configure PFx pins speed to 100 MHz */
GPIOF->OSPEEDR = 0xFF000FFF;
/* Configure PFx pins Output type to push-pull */
```

```
  GPIOF->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PFx pins */
  GPIOF->PUPDR   = 0x00000000;
  /* Connect PGx pins to FMC Alternate function */
  GPIOG->AFR[0]  = 0x00CCCCCC;
  GPIOG->AFR[1]  = 0x000000C0;
  /* Configure PGx pins in Alternate function mode */
  GPIOG->MODER   = 0x00085AAA;
  /* Configure PGx pins speed to 100 MHz */
  GPIOG->OSPEEDR = 0x000CAFFF;
  /* Configure PGx pins Output type to push-pull */
  GPIOG->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;
/*-- FMC/FSMC Configuration ----------------------------------------------
-*/
  /* Enable the FMC/FSMC interface clock */
  RCC->AHB3ENR         |= 0x00000001;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
```

```
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F469xx || STM32F479xx */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx)|| defined(STM32F417xx)\
   || defined(STM32F412Zx) || defined(STM32F412Vx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FSMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FSMC_Bank1->BTCR[2]  = 0x00001011;
  FSMC_Bank1->BTCR[3]  = 0x00000201;
  FSMC_Bank1E->BWTR[2] = 0x0FFFFFFF;
#endif /* STM32F405xx || STM32F415xx || STM32F407xx ||
STM32F417xx || STM32F412Zx || STM32F412Vx */
#endif /* DATA_IN_ExtSRAM */
```

```
#endif /* STM32F405xx || STM32F415xx || STM32F407xx ||
STM32F417xx || STM32F427xx || STM32F437xx ||\
        STM32F429xx || STM32F439xx || STM32F469xx ||
STM32F479xx || STM32F412Zx || STM32F412Vx  */
  (void)(tmp);
}
#endif /* DATA_IN_ExtSRAM && DATA_IN_ExtSDRAM */
/**
  * @}
  */
/**
  * @}
  */
/**
  * @}
  */
```

CHAPTER#31 6_ADC SIGNAL CONVERSION MODE

31.1 6_ADC SIGNAL CONVERSION MODE:

In the context of ADC (Analog-to-Digital Converter) drivers, the term "signal conversion mode" generally refers to the different ways in which the ADC can convert analog signals to digital data. Here are six common ADC signal conversion modes:

31.1.1 SIGNAL CONVERSION MODE:

In Single Conversion Mode, the ADC converts the analog signal from a single channel once and then stops. This mode is useful when you need a one-time reading from a sensor or an input pin.

31.1.1.1 USE CASE:

Ideal for applications requiring sporadic measurements or one-time data capture, such as configuration settings or user inputs.

31.1.1.2 CONFIGURATION STEPS:

1: SELECT THE CHANNEL TO CONVERT.

2: SET THE ADC TO SINGLE CONVERSION MODE.

3: START THE CONVERSION AND WAIT FOR IT TO COMPLETE.

31.1.2 CONTINUOUS CONVERSION MODE:

Continuous Conversion Mode allows the ADC to continuously convert the analog signal on a single or multiple channels. This mode is beneficial for real-time monitoring applications where data needs to be continuously sampled.

31.1.2.1USE CASES:

Suitable for real-time data logging, audio signal processing, or sensor data acquisition.

31.1.2.2 CONFIGURATION STEPS:

1: SET THE ADC TO CONTINUOUSCONVERSION MODE.

2: START THE CONVERSION MODE

3: CONTINUOUSLY READ THE ADC DATA REGISTER AS NEW DATA BECOMES AVAILABLE.

31.1.3 SCAN MODE:

Scan Mode is used to convert multiple analog channels sequentially. This mode is beneficial when you need to sample multiple sensors or signals.

31.1.3.1 USE CASES:

Useful in systems with multiple sensors, such as environmental monitoring stations or multi-sensor fusion systems.

31.1.3.2 CONFIGURATION STEPS:

1: Enable scan mode in the ADC settings.

2: Define the sequence of channels to be converted.

3: Set the ADC to either single scan or continuous scan mode.

4: Start the conversion and read data as each channel is converted.

31.1.4 DISCONTINUOUS MODE:

In Discontinuous Mode, the ADC converts groups of channels with a pause in between. This mode is useful for applications that need to minimize CPU load or power consumption by avoiding continuous sampling.

31.1.4.1 USE CASE:

Efficient in battery-powered systems or when multiple measurements need to be separated by a specific time interval.

31.1.4.2 CONFIGURATION STEPS:

1: Enable discontinuous mode in the ADC settings.

2: Specify the number of conversions per discontinuous cycle.

3: Define the channel sequence.

4: Start the conversion process.

31.1.5 TRIGGERED CONVERSION MODE:

Triggered Conversion Mode allows the ADC to start a conversion based on an external event or trigger, such as a timer, external signal, or software command.

31.1.5.1 USE CASES:

Critical for synchronizing ADC measurements with specific events, such as capturing data at specific intervals or in response to a sensor trigger.

31.1.5.2 CONFIGURATION STEPS:

1: Configure the trigger source (e.g., external pin, timer).

2: Set the ADC to start conversion on the selected trigger.

3: Wait for the trigger event to occur and then process the data.

31.1.6 INJECTED CONVERSION MODE:

Injected Conversion Mode is similar to regular conversion but typically used for high-priority measurements that interrupt ongoing regular conversions. It allows for immediate sampling of a signal without waiting for the completion of ongoing regular conversions.

31.1.6.1 USE CASE:

Useful in situations where critical measurements must be taken promptly, such as emergency monitoring systems or when specific sensor data must be prioritized.

31.1.6.2 CONFIGURATION STEPS:

1: Set the ADC to injected mode.

2: Define the injected channel sequence and conversion settings.

3: Start the injected conversion either manually or via a trigger.

31.2 CODE OF MAIN.H:

```
/* USER CODE BEGIN Header */
/**
*****************************************************************
*******************
  * @file           : main.h
  * @brief          : Header for main.c file.
  *                   This file contains the common defines of the application.
*****************************************************************
*******************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
```

```
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H
#ifdef __cplusplus
extern "C" {
#endif
/* Includes ------------------------------------------------------------------*/
#include "stm32f4xx_hal.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */
/* USER CODE END EC */
/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */
```

```
/* USER CODE END EM */
/* Exported functions prototypes -----------------------------------------*/
void Error_Handler(void);
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
/* Private defines -------------------------------------------------------*/
#define DATA_Ready_Pin GPIO_PIN_2
#define DATA_Ready_GPIO_Port GPIOE
#define CS_I2C_SPI_Pin GPIO_PIN_3
#define CS_I2C_SPI_GPIO_Port GPIOE
#define INT1_Pin GPIO_PIN_4
#define INT1_GPIO_Port GPIOE
#define INT2_Pin GPIO_PIN_5
#define INT2_GPIO_Port GPIOE
#define PC14_OSC32_IN_Pin GPIO_PIN_14
#define PC14_OSC32_IN_GPIO_Port GPIOC
#define PC15_OSC32_OUT_Pin GPIO_PIN_15
#define PC15_OSC32_OUT_GPIO_Port GPIOC
#define PH0_OSC_IN_Pin GPIO_PIN_0
#define PH0_OSC_IN_GPIO_Port GPIOH
#define PH1_OSC_OUT_Pin GPIO_PIN_1
#define PH1_OSC_OUT_GPIO_Port GPIOH
#define OTG_FS_PowerSwitchOn_Pin GPIO_PIN_0
```

```
#define OTG_FS_PowerSwitchOn_GPIO_Port GPIOC
#define PDM_OUT_Pin GPIO_PIN_3
#define PDM_OUT_GPIO_Port GPIOC
#define I2S3_WS_Pin GPIO_PIN_4
#define I2S3_WS_GPIO_Port GPIOA
#define SPI1_SCK_Pin GPIO_PIN_5
#define SPI1_SCK_GPIO_Port GPIOA
#define SPI1_MISO_Pin GPIO_PIN_6
#define SPI1_MISO_GPIO_Port GPIOA
#define SPI1_MOSI_Pin GPIO_PIN_7
#define SPI1_MOSI_GPIO_Port GPIOA
#define CLK_IN_Pin GPIO_PIN_10
#define CLK_IN_GPIO_Port GPIOB
#define LD4_Pin GPIO_PIN_12
#define LD4_GPIO_Port GPIOD
#define LD3_Pin GPIO_PIN_13
#define LD3_GPIO_Port GPIOD
#define LD5_Pin GPIO_PIN_14
#define LD5_GPIO_Port GPIOD
#define LD6_Pin GPIO_PIN_15
#define LD6_GPIO_Port GPIOD
#define I2S3_MCK_Pin GPIO_PIN_7
#define I2S3_MCK_GPIO_Port GPIOC
```

```
#define VBUS_FS_Pin GPIO_PIN_9
#define VBUS_FS_GPIO_Port GPIOA
#define OTG_FS_ID_Pin GPIO_PIN_10
#define OTG_FS_ID_GPIO_Port GPIOA
#define OTG_FS_DM_Pin GPIO_PIN_11
#define OTG_FS_DM_GPIO_Port GPIOA
#define OTG_FS_DP_Pin GPIO_PIN_12
#define OTG_FS_DP_GPIO_Port GPIOA
#define SWDIO_Pin GPIO_PIN_13
#define SWDIO_GPIO_Port GPIOA
#define SWCLK_Pin GPIO_PIN_14
#define SWCLK_GPIO_Port GPIOA
#define I2S3_SCK_Pin GPIO_PIN_10
#define I2S3_SCK_GPIO_Port GPIOC
#define I2S3_SD_Pin GPIO_PIN_12
#define I2S3_SD_GPIO_Port GPIOC
#define Audio_RST_Pin GPIO_PIN_4
#define Audio_RST_GPIO_Port GPIOD
#define OTG_FS_OverCurrent_Pin GPIO_PIN_5
#define OTG_FS_OverCurrent_GPIO_Port GPIOD
#define SWO_Pin GPIO_PIN_3
#define SWO_GPIO_Port GPIOB
#define Audio_SCL_Pin GPIO_PIN_6
```

```
#define Audio_SCL_GPIO_Port GPIOB
#define Audio_SDA_Pin GPIO_PIN_9
#define Audio_SDA_GPIO_Port GPIOB
#define MEMS_INT2_Pin GPIO_PIN_1
#define MEMS_INT2_GPIO_Port GPIOE
/* USER CODE BEGIN Private defines */
/* USER CODE END Private defines */
#ifdef __cplusplus
}
#endif
#endif /* __MAIN_H */
```

31.3 CODE OF STM32F4XX_HAL_CONF.H:

```
/* USER CODE BEGIN Header */
/**
*****************************************************************
*******************
  * @file    stm32f4xx_hal_conf_template.h
  * @author  MCD Application Team
  * @brief   HAL configuration template file.
  *          This file should be copied to the application folder and renamed
  *          to stm32f4xx_hal_conf.h.
*****************************************************************
*******************
  * @attention
```

```
  *
  * Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*****************************************************************
*******************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_HAL_CONF_H
#define __STM32F4xx_HAL_CONF_H
#ifdef __cplusplus
 extern "C" {
#endif
/* Exported types ------------------------------------------------------------*/
/* Exported constants --------------------------------------------------------*/
/* ############################ Module Selection
################################## */
/**
```

```
* @brief This is the list of modules to be used in the HAL driver
  */
#define HAL_MODULE_ENABLED
  /* #define HAL_CRYP_MODULE_ENABLED */
/* #define HAL_ADC_MODULE_ENABLED */
/* #define HAL_CAN_MODULE_ENABLED */
/* #define HAL_CRC_MODULE_ENABLED */
/* #define HAL_CAN_LEGACY_MODULE_ENABLED */
/* #define HAL_DAC_MODULE_ENABLED */
/* #define HAL_DCMI_MODULE_ENABLED */
/* #define HAL_DMA2D_MODULE_ENABLED */
/* #define HAL_ETH_MODULE_ENABLED */
/* #define HAL_ETH_LEGACY_MODULE_ENABLED */
/* #define HAL_NAND_MODULE_ENABLED */
/* #define HAL_NOR_MODULE_ENABLED */
/* #define HAL_PCCARD_MODULE_ENABLED */
/* #define HAL_SRAM_MODULE_ENABLED */
/* #define HAL_SDRAM_MODULE_ENABLED */
/* #define HAL_HASH_MODULE_ENABLED */
#define HAL_I2C_MODULE_ENABLED
#define HAL_I2S_MODULE_ENABLED
/* #define HAL_IWDG_MODULE_ENABLED */
/* #define HAL_LTDC_MODULE_ENABLED */
```

```
/* #define HAL_RNG_MODULE_ENABLED */
/* #define HAL_RTC_MODULE_ENABLED */
/* #define HAL_SAI_MODULE_ENABLED */
/* #define HAL_SD_MODULE_ENABLED */
/* #define HAL_MMC_MODULE_ENABLED */
#define HAL_SPI_MODULE_ENABLED
/* #define HAL_TIM_MODULE_ENABLED */
/* #define HAL_UART_MODULE_ENABLED */
/* #define HAL_USART_MODULE_ENABLED */
/* #define HAL_IRDA_MODULE_ENABLED */
/* #define HAL_SMARTCARD_MODULE_ENABLED */
/* #define HAL_SMBUS_MODULE_ENABLED */
/* #define HAL_WWDG_MODULE_ENABLED */
/* #define HAL_PCD_MODULE_ENABLED */
#define HAL_HCD_MODULE_ENABLED
/* #define HAL_DSI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_CEC_MODULE_ENABLED */
/* #define HAL_FMPI2C_MODULE_ENABLED */
/* #define HAL_FMPSMBUS_MODULE_ENABLED */
/* #define HAL_SPDIFRX_MODULE_ENABLED */
/* #define HAL_DFSDM_MODULE_ENABLED */
```

```
/* #define HAL_LPTIM_MODULE_ENABLED */
#define HAL_GPIO_MODULE_ENABLED
#define HAL_EXTI_MODULE_ENABLED
#define HAL_DMA_MODULE_ENABLED
#define HAL_RCC_MODULE_ENABLED
#define HAL_FLASH_MODULE_ENABLED
#define HAL_PWR_MODULE_ENABLED
#define HAL_CORTEX_MODULE_ENABLED
/* ############################ HSE/HSI Values adaptation
###################### */
/**
  * @brief Adjust the value of External High Speed oscillator (HSE)
used in your application.
  *        This value is used by the RCC HAL module to compute the
system frequency
  *        (when HSE is used as system clock source, directly or through
the PLL).
  */
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    8000000U /*!< Value of the External
oscillator in Hz */
#endif /* HSE_VALUE */
#if !defined  (HSE_STARTUP_TIMEOUT)
  #define HSE_STARTUP_TIMEOUT    100U   /*!< Time out for
HSE start up, in ms */
```

```
#endif /* HSE_STARTUP_TIMEOUT */
/**
  * @brief Internal High Speed oscillator (HSI) value.
  *        This value is used by the RCC HAL module to compute the
system frequency
  *        (when HSI is used as system clock source, directly or through the
PLL).
  */
#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000U) /*!< Value of the
Internal oscillator in Hz*/
#endif /* HSI_VALUE */
/**
  * @brief Internal Low Speed oscillator (LSI) value.
  */
#if !defined  (LSI_VALUE)
 #define LSI_VALUE  32000U       /*!< LSI Typical Value in Hz*/
#endif /* LSI_VALUE */                    /*!< Value of the Internal Low
Speed oscillator in Hz
                                  The real value may vary depending on the
variations
                                  in voltage and temperature.*/
/**
  * @brief External Low Speed oscillator (LSE) value.
  */
```

```
#if !defined  (LSE_VALUE)
 #define LSE_VALUE  32768U    /*!< Value of the External Low Speed
oscillator in Hz */
#endif /* LSE_VALUE */
#if !defined  (LSE_STARTUP_TIMEOUT)
  #define LSE_STARTUP_TIMEOUT    5000U   /*!< Time out for
LSE start up, in ms */
#endif /* LSE_STARTUP_TIMEOUT */
/**
  * @brief External clock source for I2S peripheral
  *        This value is used by the I2S HAL module to compute the I2S
clock source
  *        frequency, this source is inserted directly through I2S_CKIN
pad.
  */
#if !defined  (EXTERNAL_CLOCK_VALUE)
  #define EXTERNAL_CLOCK_VALUE    12288000U /*!< Value of
the External audio frequency in Hz*/
#endif /* EXTERNAL_CLOCK_VALUE */
/* Tip: To avoid modifying this file each time you need to use different
HSE,
   ===  you can define the HSE value in your toolchain compiler
preprocessor. */
/* ############################ System Configuration
########################### */
```

```
/**
  * @brief This is the HAL system configuration section
  */
#define  VDD_VALUE                    3300U /*!< Value of VDD in mv */
#define  TICK_INT_PRIORITY            0U   /*!< tick interrupt
priority */
#define  USE_RTOS                     0U
#define  PREFETCH_ENABLE              1U
#define  INSTRUCTION_CACHE_ENABLE     1U
#define  DATA_CACHE_ENABLE            1U
#define  USE_HAL_ADC_REGISTER_CALLBACKS        0U /*
ADC register callback disabled       */
#define  USE_HAL_CAN_REGISTER_CALLBACKS        0U /*
CAN register callback disabled       */
#define  USE_HAL_CEC_REGISTER_CALLBACKS        0U /*
CEC register callback disabled       */
#define  USE_HAL_CRYP_REGISTER_CALLBACKS       0U /*
CRYP register callback disabled      */
#define  USE_HAL_DAC_REGISTER_CALLBACKS        0U /*
DAC register callback disabled       */
#define  USE_HAL_DCMI_REGISTER_CALLBACKS       0U /*
DCMI register callback disabled      */
#define  USE_HAL_DFSDM_REGISTER_CALLBACKS      0U /*
DFSDM register callback disabled     */
#define  USE_HAL_DMA2D_REGISTER_CALLBACKS      0U /*
DMA2D register callback disabled     */
```

#define USE_HAL_DSI_REGISTER_CALLBACKS 0U /* DSI register callback disabled */

#define USE_HAL_ETH_REGISTER_CALLBACKS 0U /* ETH register callback disabled */

#define USE_HAL_HASH_REGISTER_CALLBACKS 0U /* HASH register callback disabled */

#define USE_HAL_HCD_REGISTER_CALLBACKS 0U /* HCD register callback disabled */

#define USE_HAL_I2C_REGISTER_CALLBACKS 0U /* I2C register callback disabled */

#define USE_HAL_FMPI2C_REGISTER_CALLBACKS 0U /* FMPI2C register callback disabled */

#define USE_HAL_FMPSMBUS_REGISTER_CALLBACKS 0U /* FMPSMBUS register callback disabled */

#define USE_HAL_I2S_REGISTER_CALLBACKS 0U /* I2S register callback disabled */

#define USE_HAL_IRDA_REGISTER_CALLBACKS 0U /* IRDA register callback disabled */

#define USE_HAL_LPTIM_REGISTER_CALLBACKS 0U /* LPTIM register callback disabled */

#define USE_HAL_LTDC_REGISTER_CALLBACKS 0U /* LTDC register callback disabled */

#define USE_HAL_MMC_REGISTER_CALLBACKS 0U /* MMC register callback disabled */

#define USE_HAL_NAND_REGISTER_CALLBACKS 0U /* NAND register callback disabled */

#define USE_HAL_NOR_REGISTER_CALLBACKS 0U /* NOR register callback disabled */

#define USE_HAL_PCCARD_REGISTER_CALLBACKS 0U /* PCCARD register callback disabled */

#define USE_HAL_PCD_REGISTER_CALLBACKS 0U /* PCD register callback disabled */

#define USE_HAL_QSPI_REGISTER_CALLBACKS 0U /* QSPI register callback disabled */

#define USE_HAL_RNG_REGISTER_CALLBACKS 0U /* RNG register callback disabled */

#define USE_HAL_RTC_REGISTER_CALLBACKS 0U /* RTC register callback disabled */

#define USE_HAL_SAI_REGISTER_CALLBACKS 0U /* SAI register callback disabled */

#define USE_HAL_SD_REGISTER_CALLBACKS 0U /* SD register callback disabled */

#define USE_HAL_SMARTCARD_REGISTER_CALLBACKS 0U /* SMARTCARD register callback disabled */

#define USE_HAL_SDRAM_REGISTER_CALLBACKS 0U /* SDRAM register callback disabled */

#define USE_HAL_SRAM_REGISTER_CALLBACKS 0U /* SRAM register callback disabled */

#define USE_HAL_SPDIFRX_REGISTER_CALLBACKS 0U /* SPDIFRX register callback disabled */

#define USE_HAL_SMBUS_REGISTER_CALLBACKS 0U /* SMBUS register callback disabled */

#define USE_HAL_SPI_REGISTER_CALLBACKS 0U /* SPI register callback disabled */

#define USE_HAL_TIM_REGISTER_CALLBACKS 0U /* TIM register callback disabled */

#define USE_HAL_UART_REGISTER_CALLBACKS 0U /* UART register callback disabled */

#define USE_HAL_USART_REGISTER_CALLBACKS 0U /* USART register callback disabled */

#define USE_HAL_WWDG_REGISTER_CALLBACKS 0U /* WWDG register callback disabled */

/* ########################### Assert Selection ################################ */

/**

* @brief Uncomment the line below to expanse the "assert_param" macro in the

* HAL drivers code

*/

/* #define USE_FULL_ASSERT 1U */

/* ################### Ethernet peripheral configuration ###################### */

/* Section 1 : Ethernet peripheral configuration */

/* MAC ADDRESS: MAC_ADDR0:MAC_ADDR1:MAC_ADDR2:MAC_ADDR3:MAC_ADDR4:MAC_ADDR5 */

#define MAC_ADDR0 2U

#define MAC_ADDR1 0U

```
#define MAC_ADDR2   0U
#define MAC_ADDR3   0U
#define MAC_ADDR4   0U
#define MAC_ADDR5   0U
/* Definition of the Ethernet driver buffers size and count */
#define ETH_RX_BUF_SIZE             ETH_MAX_PACKET_SIZE
/* buffer size for receive            */
#define ETH_TX_BUF_SIZE             ETH_MAX_PACKET_SIZE
/* buffer size for transmit           */
#define ETH_RXBUFNB                 4U      /* 4 Rx buffers of size
ETH_RX_BUF_SIZE  */
#define ETH_TXBUFNB                 4U      /* 4 Tx buffers of size
ETH_TX_BUF_SIZE  */
/* Section 2: PHY configuration section */
/* DP83848_PHY_ADDRESS Address*/
#define DP83848_PHY_ADDRESS
/* PHY Reset delay these values are based on a 1 ms Systick interrupt*/
#define PHY_RESET_DELAY             0x000000FFU
/* PHY Configuration delay */
#define PHY_CONFIG_DELAY            0x00000FFFU
#define PHY_READ_TO                 0x0000FFFFU
#define PHY_WRITE_TO                0x0000FFFFU
/* Section 3: Common PHY Registers */
#define PHY_BCR                     ((uint16_t)0x0000U)    /*!<
Transceiver Basic Control Register   */
```

#define PHY_BSR ((uint16_t)0x0001U) /*!< Transceiver Basic Status Register */

#define PHY_RESET ((uint16_t)0x8000U) /*!< PHY Reset */

#define PHY_LOOPBACK ((uint16_t)0x4000U) /*!< Select loop-back mode */

#define PHY_FULLDUPLEX_100M ((uint16_t)0x2100U) /*!< Set the full-duplex mode at 100 Mb/s */

#define PHY_HALFDUPLEX_100M ((uint16_t)0x2000U) /*!< Set the half-duplex mode at 100 Mb/s */

#define PHY_FULLDUPLEX_10M ((uint16_t)0x0100U) /*!< Set the full-duplex mode at 10 Mb/s */

#define PHY_HALFDUPLEX_10M ((uint16_t)0x0000U) /*!< Set the half-duplex mode at 10 Mb/s */

#define PHY_AUTONEGOTIATION ((uint16_t)0x1000U) /*!< Enable auto-negotiation function */

#define PHY_RESTART_AUTONEGOTIATION ((uint16_t)0x0200U) /*!< Restart auto-negotiation function */

#define PHY_POWERDOWN ((uint16_t)0x0800U) /*!< Select the power down mode */

#define PHY_ISOLATE ((uint16_t)0x0400U) /*!< Isolate PHY from MII */

#define PHY_AUTONEGO_COMPLETE ((uint16_t)0x0020U) /*!< Auto-Negotiation process completed */

#define PHY_LINKED_STATUS ((uint16_t)0x0004U) /*!< Valid link established */

```
#define PHY_JABBER_DETECTION          ((uint16_t)0x0002U)
/*!< Jabber condition detected          */

/* Section 4: Extended PHY Registers */

#define PHY_SR                        ((uint16_t))    /*!< PHY status register
Offset                    */

#define PHY_SPEED_STATUS              ((uint16_t))  /*!< PHY Speed
mask                          */

#define PHY_DUPLEX_STATUS             ((uint16_t))  /*!< PHY
Duplex mask                           */

/* ################### SPI peripheral configuration
########################### */

/* CRC FEATURE: Use to activate CRC feature inside HAL SPI Driver

* Activated: CRC code is present inside driver

* Deactivated: CRC code cleaned from driver

*/

#define USE_SPI_CRC                   0U

/* Includes ------------------------------------------------------------------*/

/**

  * @brief Include module's header file

  */

#ifdef HAL_RCC_MODULE_ENABLED

  #include "stm32f4xx_hal_rcc.h"

#endif /* HAL_RCC_MODULE_ENABLED */

#ifdef HAL_GPIO_MODULE_ENABLED

  #include "stm32f4xx_hal_gpio.h"
```

```
#endif /* HAL_GPIO_MODULE_ENABLED */
#ifdef HAL_EXTI_MODULE_ENABLED
  #include "stm32f4xx_hal_exti.h"
#endif /* HAL_EXTI_MODULE_ENABLED */
#ifdef HAL_DMA_MODULE_ENABLED
  #include "stm32f4xx_hal_dma.h"
#endif /* HAL_DMA_MODULE_ENABLED */
#ifdef HAL_CORTEX_MODULE_ENABLED
  #include "stm32f4xx_hal_cortex.h"
#endif /* HAL_CORTEX_MODULE_ENABLED */
#ifdef HAL_ADC_MODULE_ENABLED
  #include "stm32f4xx_hal_adc.h"
#endif /* HAL_ADC_MODULE_ENABLED */
#ifdef HAL_CAN_MODULE_ENABLED
  #include "stm32f4xx_hal_can.h"
#endif /* HAL_CAN_MODULE_ENABLED */
#ifdef HAL_CAN_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_can_legacy.h"
#endif /* HAL_CAN_LEGACY_MODULE_ENABLED */
#ifdef HAL_CRC_MODULE_ENABLED
  #include "stm32f4xx_hal_crc.h"
#endif /* HAL_CRC_MODULE_ENABLED */
#ifdef HAL_CRYP_MODULE_ENABLED
```

```
  #include "stm32f4xx_hal_cryp.h"
#endif /* HAL_CRYP_MODULE_ENABLED */
#ifdef HAL_DMA2D_MODULE_ENABLED
  #include "stm32f4xx_hal_dma2d.h"
#endif /* HAL_DMA2D_MODULE_ENABLED */
#ifdef HAL_DAC_MODULE_ENABLED
  #include "stm32f4xx_hal_dac.h"
#endif /* HAL_DAC_MODULE_ENABLED */
#ifdef HAL_DCMI_MODULE_ENABLED
  #include "stm32f4xx_hal_dcmi.h"
#endif /* HAL_DCMI_MODULE_ENABLED */
#ifdef HAL_ETH_MODULE_ENABLED
  #include "stm32f4xx_hal_eth.h"
#endif /* HAL_ETH_MODULE_ENABLED */
#ifdef HAL_ETH_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_eth_legacy.h"
#endif /* HAL_ETH_LEGACY_MODULE_ENABLED */
#ifdef HAL_FLASH_MODULE_ENABLED
  #include "stm32f4xx_hal_flash.h"
#endif /* HAL_FLASH_MODULE_ENABLED */
#ifdef HAL_SRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sram.h"
#endif /* HAL_SRAM_MODULE_ENABLED */
```

```
#ifdef HAL_NOR_MODULE_ENABLED
  #include "stm32f4xx_hal_nor.h"
#endif /* HAL_NOR_MODULE_ENABLED */
#ifdef HAL_NAND_MODULE_ENABLED
  #include "stm32f4xx_hal_nand.h"
#endif /* HAL_NAND_MODULE_ENABLED */
#ifdef HAL_PCCARD_MODULE_ENABLED
  #include "stm32f4xx_hal_pccard.h"
#endif /* HAL_PCCARD_MODULE_ENABLED */
#ifdef HAL_SDRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sdram.h"
#endif /* HAL_SDRAM_MODULE_ENABLED */
#ifdef HAL_HASH_MODULE_ENABLED
 #include "stm32f4xx_hal_hash.h"
#endif /* HAL_HASH_MODULE_ENABLED */
#ifdef HAL_I2C_MODULE_ENABLED
 #include "stm32f4xx_hal_i2c.h"
#endif /* HAL_I2C_MODULE_ENABLED */
#ifdef HAL_SMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_smbus.h"
#endif /* HAL_SMBUS_MODULE_ENABLED */
#ifdef HAL_I2S_MODULE_ENABLED
 #include "stm32f4xx_hal_i2s.h"
```

```
#endif /* HAL_I2S_MODULE_ENABLED */
#ifdef HAL_IWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_iwdg.h"
#endif /* HAL_IWDG_MODULE_ENABLED */
#ifdef HAL_LTDC_MODULE_ENABLED
 #include "stm32f4xx_hal_ltdc.h"
#endif /* HAL_LTDC_MODULE_ENABLED */
#ifdef HAL_PWR_MODULE_ENABLED
 #include "stm32f4xx_hal_pwr.h"
#endif /* HAL_PWR_MODULE_ENABLED */
#ifdef HAL_RNG_MODULE_ENABLED
 #include "stm32f4xx_hal_rng.h"
#endif /* HAL_RNG_MODULE_ENABLED */
#ifdef HAL_RTC_MODULE_ENABLED
 #include "stm32f4xx_hal_rtc.h"
#endif /* HAL_RTC_MODULE_ENABLED */
#ifdef HAL_SAI_MODULE_ENABLED
 #include "stm32f4xx_hal_sai.h"
#endif /* HAL_SAI_MODULE_ENABLED */
#ifdef HAL_SD_MODULE_ENABLED
 #include "stm32f4xx_hal_sd.h"
#endif /* HAL_SD_MODULE_ENABLED */
#ifdef HAL_SPI_MODULE_ENABLED
```

```
 #include "stm32f4xx_hal_spi.h"
#endif /* HAL_SPI_MODULE_ENABLED */
#ifdef HAL_TIM_MODULE_ENABLED
 #include "stm32f4xx_hal_tim.h"
#endif /* HAL_TIM_MODULE_ENABLED */
#ifdef HAL_UART_MODULE_ENABLED
 #include "stm32f4xx_hal_uart.h"
#endif /* HAL_UART_MODULE_ENABLED */
#ifdef HAL_USART_MODULE_ENABLED
 #include "stm32f4xx_hal_usart.h"
#endif /* HAL_USART_MODULE_ENABLED */
#ifdef HAL_IRDA_MODULE_ENABLED
 #include "stm32f4xx_hal_irda.h"
#endif /* HAL_IRDA_MODULE_ENABLED */
#ifdef HAL_SMARTCARD_MODULE_ENABLED
 #include "stm32f4xx_hal_smartcard.h"
#endif /* HAL_SMARTCARD_MODULE_ENABLED */
#ifdef HAL_WWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_wwdg.h"
#endif /* HAL_WWDG_MODULE_ENABLED */
#ifdef HAL_PCD_MODULE_ENABLED
 #include "stm32f4xx_hal_pcd.h"
#endif /* HAL_PCD_MODULE_ENABLED */
```

```
#ifdef HAL_HCD_MODULE_ENABLED
 #include "stm32f4xx_hal_hcd.h"
#endif /* HAL_HCD_MODULE_ENABLED */
#ifdef HAL_DSI_MODULE_ENABLED
 #include "stm32f4xx_hal_dsi.h"
#endif /* HAL_DSI_MODULE_ENABLED */
#ifdef HAL_QSPI_MODULE_ENABLED
 #include "stm32f4xx_hal_qspi.h"
#endif /* HAL_QSPI_MODULE_ENABLED */
#ifdef HAL_CEC_MODULE_ENABLED
 #include "stm32f4xx_hal_cec.h"
#endif /* HAL_CEC_MODULE_ENABLED */
#ifdef HAL_FMPI2C_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpi2c.h"
#endif /* HAL_FMPI2C_MODULE_ENABLED */
#ifdef HAL_FMPSMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpsmbus.h"
#endif /* HAL_FMPSMBUS_MODULE_ENABLED */
#ifdef HAL_SPDIFRX_MODULE_ENABLED
 #include "stm32f4xx_hal_spdifrx.h"
#endif /* HAL_SPDIFRX_MODULE_ENABLED */
#ifdef HAL_DFSDM_MODULE_ENABLED
 #include "stm32f4xx_hal_dfsdm.h"
```

```
#endif /* HAL_DFSDM_MODULE_ENABLED */
#ifdef HAL_LPTIM_MODULE_ENABLED
 #include "stm32f4xx_hal_lptim.h"
#endif /* HAL_LPTIM_MODULE_ENABLED */
#ifdef HAL_MMC_MODULE_ENABLED
 #include "stm32f4xx_hal_mmc.h"
#endif /* HAL_MMC_MODULE_ENABLED */
/* Exported macro ------------------------------------------------------------*/
#ifdef  USE_FULL_ASSERT
/**
  * @brief  The assert_param macro is used for function's parameters
check.
  * @param  expr If expr is false, it calls assert_failed function
  *         which reports the name of the source file and the source
  *         line number of the call that failed.
  *         If expr is true, it returns no value.
  * @retval None
  */
  #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t
*)__FILE__, __LINE__))
/* Exported functions ------------------------------------------------------- */
  void assert_failed(uint8_t* file, uint32_t line);
#else
  #define assert_param(expr) ((void)0U)
```

```
#endif /* USE_FULL_ASSERT */
#ifdef __cplusplus
}
#endif
#endif /* __STM32F4xx_HAL_CONF_H */
```

31.4 CODE OF STM32F4XX_IT.H:

```
/* USER CODE BEGIN Header */
/**
*******************************************************************
  * @file    stm32f4xx_it.h
  * @brief   This file contains the headers of the interrupt handlers.
*******************************************************************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
```

```
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_IT_H
#define __STM32F4xx_IT_H
#ifdef __cplusplus
 extern "C" {
#endif
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */
/* USER CODE END EC */
/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */
/* USER CODE END EM */
/* Exported functions prototypes ---------------------------------------------*/
```

```
void NMI_Handler(void);
void HardFault_Handler(void);
void MemManage_Handler(void);
void BusFault_Handler(void);
void UsageFault_Handler(void);
void SVC_Handler(void);
void DebugMon_Handler(void);
void PendSV_Handler(void);
void SysTick_Handler(void);
void OTG_FS_IRQHandler(void);
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
#ifdef __cplusplus
}
#endif
#endif /* __STM32F4xx_IT_H */
```

31.5 CODE OF MAIN.C:

```
/* USER CODE BEGIN Header */
/**
*******************************************************
*******************
  * @file           : main.c
```

```
  * @brief          : Main program body
**********************************************************
*******************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
********************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "usb_host.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
```

```
/* USER CODE END PTD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
I2C_HandleTypeDef hi2c1;
I2S_HandleTypeDef hi2s2;
I2S_HandleTypeDef hi2s3;
SPI_HandleTypeDef hspi1;
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
void PeriphCommonClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_I2C1_Init(void);
static void MX_I2S2_Init(void);
static void MX_I2S3_Init(void);
static void MX_SPI1_Init(void);
void MX_USB_HOST_Process(void);
```

```
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */
  /* USER CODE END 1 */
  /* MCU Configuration--------------------------------------------------------*/
  /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
  HAL_Init();
  /* USER CODE BEGIN Init */
  /* USER CODE END Init */
  /* Configure the system clock */
  SystemClock_Config();
  /* Configure the peripherals common clocks */
  PeriphCommonClock_Config();
```

```
  /* USER CODE BEGIN SysInit */
  /* USER CODE END SysInit */
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_I2C1_Init();
  MX_I2S2_Init();
  MX_I2S3_Init();
  MX_SPI1_Init();
  MX_USB_HOST_Init();
  /* USER CODE BEGIN 2 */
  /* USER CODE END 2 */
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
    MX_USB_HOST_Process();
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
/**
  * @brief System Clock Configuration
```

```
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_
VOLTAGE_SCALE1);
  /** Initializes the RCC Oscillators according to the specified
parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType =
RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_BYPASS;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 4;
  RCC_OscInitStruct.PLL.PLLN = 192;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV4;
  RCC_OscInitStruct.PLL.PLLQ = 8;
```

```
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
 |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource =
RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct,
FLASH_LATENCY_3) != HAL_OK)
  {
    Error_Handler();
  }
}
/**
  * @brief Peripherals Common Clock Configuration
  * @retval None
  */
```

```
void PeriphCommonClock_Config(void)
{
  RCC_PeriphCLKInitTypeDef PeriphClkInitStruct = {0};
  /** Initializes the peripherals clock
  */
  PeriphClkInitStruct.PeriphClockSelection = RCC_PERIPHCLK_I2S;
  PeriphClkInitStruct.PLLI2S.PLLI2SN = 200;
  PeriphClkInitStruct.PLLI2S.PLLI2SM = 5;
  PeriphClkInitStruct.PLLI2S.PLLI2SR = 2;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInitStruct) !=
HAL_OK)
  {
    Error_Handler();
  }
}
/**
  * @brief I2C1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_I2C1_Init(void)
{
  /* USER CODE BEGIN I2C1_Init 0 */
```

```
  /* USER CODE END I2C1_Init 0 */
  /* USER CODE BEGIN I2C1_Init 1 */
  /* USER CODE END I2C1_Init 1 */
  hi2c1.Instance = I2C1;
  hi2c1.Init.ClockSpeed = 100000;
  hi2c1.Init.DutyCycle = I2C_DUTYCYCLE_2;
  hi2c1.Init.OwnAddress1 = 0;
  hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
  hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
  hi2c1.Init.OwnAddress2 = 0;
  hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
  hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
  if (HAL_I2C_Init(&hi2c1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN I2C1_Init 2 */
  /* USER CODE END I2C1_Init 2 */
}
/**
  * @brief I2S2 Initialization Function
  * @param None
  * @retval None
```

```
  */
static void MX_I2S2_Init(void)
{
  /* USER CODE BEGIN I2S2_Init 0 */
  /* USER CODE END I2S2_Init 0 */
  /* USER CODE BEGIN I2S2_Init 1 */
  /* USER CODE END I2S2_Init 1 */
  hi2s2.Instance = SPI2;
  hi2s2.Init.Mode = I2S_MODE_MASTER_TX;
  hi2s2.Init.Standard = I2S_STANDARD_PHILIPS;
  hi2s2.Init.DataFormat = I2S_DATAFORMAT_16B;
  hi2s2.Init.MCLKOutput = I2S_MCLKOUTPUT_DISABLE;
  hi2s2.Init.AudioFreq = I2S_AUDIOFREQ_96K;
  hi2s2.Init.CPOL = I2S_CPOL_LOW;
  hi2s2.Init.ClockSource = I2S_CLOCK_PLL;
  hi2s2.Init.FullDuplexMode = I2S_FULLDUPLEXMODE_ENABLE;
  if (HAL_I2S_Init(&hi2s2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN I2S2_Init 2 */
  /* USER CODE END I2S2_Init 2 */
}
```

```
/**
  * @brief I2S3 Initialization Function
  * @param None
  * @retval None
  */
static void MX_I2S3_Init(void)
{
  /* USER CODE BEGIN I2S3_Init 0 */
  /* USER CODE END I2S3_Init 0 */
  /* USER CODE BEGIN I2S3_Init 1 */
  /* USER CODE END I2S3_Init 1 */
  hi2s3.Instance = SPI3;
  hi2s3.Init.Mode = I2S_MODE_MASTER_TX;
  hi2s3.Init.Standard = I2S_STANDARD_PHILIPS;
  hi2s3.Init.DataFormat = I2S_DATAFORMAT_16B;
  hi2s3.Init.MCLKOutput = I2S_MCLKOUTPUT_ENABLE;
  hi2s3.Init.AudioFreq = I2S_AUDIOFREQ_96K;
  hi2s3.Init.CPOL = I2S_CPOL_LOW;
  hi2s3.Init.ClockSource = I2S_CLOCK_PLL;
  hi2s3.Init.FullDuplexMode = I2S_FULLDUPLEXMODE_DISABLE;
  if (HAL_I2S_Init(&hi2s3) != HAL_OK)
  {
    Error_Handler();
```

```
  }
  /* USER CODE BEGIN I2S3_Init 2 */
  /* USER CODE END I2S3_Init 2 */
}
/**
  * @brief SPI1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_SPI1_Init(void)
{
  /* USER CODE BEGIN SPI1_Init 0 */
  /* USER CODE END SPI1_Init 0 */
  /* USER CODE BEGIN SPI1_Init 1 */
  /* USER CODE END SPI1_Init 1 */
  /* SPI1 parameter configuration*/
  hspi1.Instance = SPI1;
  hspi1.Init.Mode = SPI_MODE_MASTER;
  hspi1.Init.Direction = SPI_DIRECTION_2LINES;
  hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
  hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
  hspi1.Init.NSS = SPI_NSS_SOFT;
```

```
  hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_2;
  hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
  hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
  hspi1.Init.CRCPolynomial = 10;
  if (HAL_SPI_Init(&hspi1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN SPI1_Init 2 */
  /* USER CODE END SPI1_Init 2 */
}
/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */
  /* GPIO Ports Clock Enable */
```

__HAL_RCC_GPIOE_CLK_ENABLE();

__HAL_RCC_GPIOC_CLK_ENABLE();

__HAL_RCC_GPIOH_CLK_ENABLE();

__HAL_RCC_GPIOA_CLK_ENABLE();

__HAL_RCC_GPIOB_CLK_ENABLE();

__HAL_RCC_GPIOD_CLK_ENABLE();

/*Configure GPIO pin Output Level */

HAL_GPIO_WritePin(CS_I2C_SPI_GPIO_Port, CS_I2C_SPI_Pin, *GPIO_PIN_RESET*);

/*Configure GPIO pin Output Level */

HAL_GPIO_WritePin(OTG_FS_PowerSwitchOn_GPIO_Port, OTG_FS_PowerSwitchOn_Pin, *GPIO_PIN_SET*);

/*Configure GPIO pin Output Level */

HAL_GPIO_WritePin(GPIOD, LD4_Pin|LD3_Pin|LD5_Pin|LD6_Pin

|Audio_RST_Pin, *GPIO_PIN_RESET*);

/*Configure GPIO pin : DATA_Ready_Pin */

GPIO_InitStruct.Pin = DATA_Ready_Pin;

GPIO_InitStruct.Mode = GPIO_MODE_INPUT;

GPIO_InitStruct.Pull = GPIO_NOPULL;

HAL_GPIO_Init(DATA_Ready_GPIO_Port, &GPIO_InitStruct);

/*Configure GPIO pin : CS_I2C_SPI_Pin */

GPIO_InitStruct.Pin = CS_I2C_SPI_Pin;

GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;

```
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(CS_I2C_SPI_GPIO_Port, &GPIO_InitStruct);
  /*Configure GPIO pins : INT1_Pin INT2_Pin MEMS_INT2_Pin */
  GPIO_InitStruct.Pin = INT1_Pin|INT2_Pin|MEMS_INT2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);
  /*Configure GPIO pin : OTG_FS_PowerSwitchOn_Pin */
  GPIO_InitStruct.Pin = OTG_FS_PowerSwitchOn_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(OTG_FS_PowerSwitchOn_GPIO_Port,
&GPIO_InitStruct);
  /*Configure GPIO pin : PA0 */
  GPIO_InitStruct.Pin = GPIO_PIN_0;
  GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
  /*Configure GPIO pins : LD4_Pin LD3_Pin LD5_Pin LD6_Pin
                    Audio_RST_Pin */
  GPIO_InitStruct.Pin = LD4_Pin|LD3_Pin|LD5_Pin|LD6_Pin
```

```
                    |Audio_RST_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
  /*Configure GPIO pin : OTG_FS_OverCurrent_Pin */
  GPIO_InitStruct.Pin = OTG_FS_OverCurrent_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(OTG_FS_OverCurrent_GPIO_Port,
&GPIO_InitStruct);
/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}
/* USER CODE BEGIN 4 */
/* USER CODE END 4 */
/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
```

```
  /* User can add his own implementation to report the HAL error return
state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}
#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line
number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line
number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file,
line) */
```

```
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

31.6 CODE OF STM32F4XX_HAL_MSP.C:

```
/* USER CODE BEGIN Header */
/**
*****************************************************************
*******************
  * @file         stm32f4xx_hal_msp.c
  * @brief        This file provides code for the MSP Initialization
  *               and de-Initialization codes.
*****************************************************************
*******************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*****************************************************************
*******************
```

```
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN TD */
/* USER CODE END TD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN Define */
/* USER CODE END Define */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN Macro */
/* USER CODE END Macro */
/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* External functions --------------------------------------------------------*/
/* USER CODE BEGIN ExternalFunctions */
```

```
/* USER CODE END ExternalFunctions */
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/**
  * Initializes the Global MSP.
  */
void HAL_MspInit(void)
{
  /* USER CODE BEGIN MspInit 0 */
  /* USER CODE END MspInit 0 */
  __HAL_RCC_SYSCFG_CLK_ENABLE();
  __HAL_RCC_PWR_CLK_ENABLE();
  HAL_NVIC_SetPriorityGrouping(NVIC_PRIORITYGROUP_0);
  /* System interrupt init*/
  /* USER CODE BEGIN MspInit 1 */
  /* USER CODE END MspInit 1 */
}
/**
* @brief I2C MSP Initialization
* This function configures the hardware resources used in this example
* @param hi2c: I2C handle pointer
* @retval None
*/
```

```
void HAL_I2C_MspInit(I2C_HandleTypeDef* hi2c)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  if(hi2c->Instance==I2C1)
  {
  /* USER CODE BEGIN I2C1_MspInit 0 */
  /* USER CODE END I2C1_MspInit 0 */
    __HAL_RCC_GPIOB_CLK_ENABLE();
    /**I2C1 GPIO Configuration
    PB6     ------> I2C1_SCL
    PB9     ------> I2C1_SDA
    */
    GPIO_InitStruct.Pin = Audio_SCL_Pin|Audio_SDA_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_OD;
    GPIO_InitStruct.Pull = GPIO_PULLUP;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    GPIO_InitStruct.Alternate = GPIO_AF4_I2C1;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
    /* Peripheral clock enable */
    __HAL_RCC_I2C1_CLK_ENABLE();
  /* USER CODE BEGIN I2C1_MspInit 1 */
  /* USER CODE END I2C1_MspInit 1 */
  }
```

```
}
/**
* @brief I2C MSP De-Initialization
* This function freeze the hardware resources used in this example
* @param hi2c: I2C handle pointer
* @retval None
*/
void HAL_I2C_MspDeInit(I2C_HandleTypeDef* hi2c)
{
  if(hi2c->Instance==I2C1)
  {
  /* USER CODE BEGIN I2C1_MspDeInit 0 */
  /* USER CODE END I2C1_MspDeInit 0 */
    /* Peripheral clock disable */
    __HAL_RCC_I2C1_CLK_DISABLE();
    /**I2C1 GPIO Configuration
    PB6     ------> I2C1_SCL
    PB9     ------> I2C1_SDA
    */
    HAL_GPIO_DeInit(Audio_SCL_GPIO_Port, Audio_SCL_Pin);
    HAL_GPIO_DeInit(Audio_SDA_GPIO_Port, Audio_SDA_Pin);
  /* USER CODE BEGIN I2C1_MspDeInit 1 */
  /* USER CODE END I2C1_MspDeInit 1 */
```

```
  }
}
/**
* @brief I2S MSP Initialization
* This function configures the hardware resources used in this example
* @param hi2s: I2S handle pointer
* @retval None
*/
void HAL_I2S_MspInit(I2S_HandleTypeDef* hi2s)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  if(hi2s->Instance==SPI2)
  {
  /* USER CODE BEGIN SPI2_MspInit 0 */
  /* USER CODE END SPI2_MspInit 0 */
    /* Peripheral clock enable */
    __HAL_RCC_SPI2_CLK_ENABLE();
    __HAL_RCC_GPIOC_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();
    /**I2S2 GPIO Configuration
    PC2     ------> I2S2_ext_SD
    PC3     ------> I2S2_SD
    PB10     ------> I2S2_CK
```

```
  PB12     ------> I2S2_WS
  */
  GPIO_InitStruct.Pin = GPIO_PIN_2;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.Alternate = GPIO_AF6_I2S2ext;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
  GPIO_InitStruct.Pin = PDM_OUT_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.Alternate = GPIO_AF5_SPI2;
  HAL_GPIO_Init(PDM_OUT_GPIO_Port, &GPIO_InitStruct);
  GPIO_InitStruct.Pin = CLK_IN_Pin|GPIO_PIN_12;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.Alternate = GPIO_AF5_SPI2;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
/* USER CODE BEGIN SPI2_MspInit 1 */
/* USER CODE END SPI2_MspInit 1 */
}
```

```
else if(hi2s->Instance==SPI3)
{
/* USER CODE BEGIN SPI3_MspInit 0 */
/* USER CODE END SPI3_MspInit 0 */
  /* Peripheral clock enable */
  __HAL_RCC_SPI3_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOC_CLK_ENABLE();
  /**I2S3 GPIO Configuration
  PA4     ------> I2S3_WS
  PC7     ------> I2S3_MCK
  PC10     ------> I2S3_CK
  PC12     ------> I2S3_SD
  */
  GPIO_InitStruct.Pin = I2S3_WS_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.Alternate = GPIO_AF6_SPI3;
  HAL_GPIO_Init(I2S3_WS_GPIO_Port, &GPIO_InitStruct);
  GPIO_InitStruct.Pin = I2S3_MCK_Pin|I2S3_SCK_Pin|I2S3_SD_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
```

```
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    GPIO_InitStruct.Alternate = GPIO_AF6_SPI3;
    HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
  /* USER CODE BEGIN SPI3_MspInit 1 */
  /* USER CODE END SPI3_MspInit 1 */
  }
}
/**
* @brief I2S MSP De-Initialization
* This function freeze the hardware resources used in this example
* @param hi2s: I2S handle pointer
* @retval None
*/
void HAL_I2S_MspDeInit(I2S_HandleTypeDef* hi2s)
{
  if(hi2s->Instance==SPI2)
  {
  /* USER CODE BEGIN SPI2_MspDeInit 0 */
  /* USER CODE END SPI2_MspDeInit 0 */
    /* Peripheral clock disable */
    __HAL_RCC_SPI2_CLK_DISABLE();
    /**I2S2 GPIO Configuration
    PC2     ------> I2S2_ext_SD
```

```
  PC3     ------> I2S2_SD
  PB10    ------> I2S2_CK
  PB12    ------> I2S2_WS
  */
  HAL_GPIO_DeInit(GPIOC, GPIO_PIN_2|PDM_OUT_Pin);
  HAL_GPIO_DeInit(GPIOB, CLK_IN_Pin|GPIO_PIN_12);
/* USER CODE BEGIN SPI2_MspDeInit 1 */
/* USER CODE END SPI2_MspDeInit 1 */
}
else if(hi2s->Instance==SPI3)
{
/* USER CODE BEGIN SPI3_MspDeInit 0 */
/* USER CODE END SPI3_MspDeInit 0 */
  /* Peripheral clock disable */
  __HAL_RCC_SPI3_CLK_DISABLE();
  /**I2S3 GPIO Configuration
  PA4     ------> I2S3_WS
  PC7     ------> I2S3_MCK
  PC10    ------> I2S3_CK
  PC12    ------> I2S3_SD
  */
  HAL_GPIO_DeInit(I2S3_WS_GPIO_Port, I2S3_WS_Pin);
```

```
    HAL_GPIO_DeInit(GPIOC,
I2S3_MCK_Pin|I2S3_SCK_Pin|I2S3_SD_Pin);
  /* USER CODE BEGIN SPI3_MspDeInit 1 */
  /* USER CODE END SPI3_MspDeInit 1 */
  }
}
/**
* @brief SPI MSP Initialization
* This function configures the hardware resources used in this example
* @param hspi: SPI handle pointer
* @retval None
*/
void HAL_SPI_MspInit(SPI_HandleTypeDef* hspi)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  if(hspi->Instance==SPI1)
  {
  /* USER CODE BEGIN SPI1_MspInit 0 */
  /* USER CODE END SPI1_MspInit 0 */
    /* Peripheral clock enable */
    __HAL_RCC_SPI1_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();
    /**SPI1 GPIO Configuration
```

```
    PA5     ------> SPI1_SCK
    PA6     ------> SPI1_MISO
    PA7     ------> SPI1_MOSI
    */
    GPIO_InitStruct.Pin =
SPI1_SCK_Pin|SPI1_MISO_Pin|SPI1_MOSI_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF5_SPI1;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
  /* USER CODE BEGIN SPI1_MspInit 1 */
  /* USER CODE END SPI1_MspInit 1 */
  }
}
/**
* @brief SPI MSP De-Initialization
* This function freeze the hardware resources used in this example
* @param hspi: SPI handle pointer
* @retval None
*/
void HAL_SPI_MspDeInit(SPI_HandleTypeDef* hspi)
{
```

```
  if(hspi->Instance==SPI1)
  {
  /* USER CODE BEGIN SPI1_MspDeInit 0 */
  /* USER CODE END SPI1_MspDeInit 0 */
    /* Peripheral clock disable */
    __HAL_RCC_SPI1_CLK_DISABLE();
    /**SPI1 GPIO Configuration
    PA5     ------> SPI1_SCK
    PA6     ------> SPI1_MISO
    PA7     ------> SPI1_MOSI
    */
    HAL_GPIO_DeInit(GPIOA,
SPI1_SCK_Pin|SPI1_MISO_Pin|SPI1_MOSI_Pin);
  /* USER CODE BEGIN SPI1_MspDeInit 1 */
  /* USER CODE END SPI1_MspDeInit 1 */
  }
}
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

31.7 CODE OF STM32F4XX_IT.C:

```
/* USER CODE BEGIN Header */
/**
*****************************************************************
*******************
```

```
  * @file    stm32f4xx_it.c
  * @brief   Interrupt Service Routines.
**********************************************************
*******************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
**********************************************************
*******************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stm32f4xx_it.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
```

```
/* USER CODE BEGIN TD */
/* USER CODE END TD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/* External variables --------------------------------------------------------*/
extern HCD_HandleTypeDef hhcd_USB_OTG_FS;
/* USER CODE BEGIN EV */
/* USER CODE END EV */
/**********************************************************************
*********************/
```

```
/*          Cortex-M4 Processor Interruption and Exception Handlers
*/
/*****************************************************************
*********************/
/**
  * @brief This function handles Non maskable interrupt.
  */
void NMI_Handler(void)
{
  /* USER CODE BEGIN NonMaskableInt_IRQn 0 */
  /* USER CODE END NonMaskableInt_IRQn 0 */
  /* USER CODE BEGIN NonMaskableInt_IRQn 1 */
   while (1)
  {
  }
  /* USER CODE END NonMaskableInt_IRQn 1 */
}
/**
  * @brief This function handles Hard fault interrupt.
  */
void HardFault_Handler(void)
{
  /* USER CODE BEGIN HardFault_IRQn 0 */
  /* USER CODE END HardFault_IRQn 0 */
```

```
  while (1)
  {
    /* USER CODE BEGIN W1_HardFault_IRQn 0 */
    /* USER CODE END W1_HardFault_IRQn 0 */
  }
}
/**
  * @brief This function handles Memory management fault.
  */
void MemManage_Handler(void)
{
  /* USER CODE BEGIN MemoryManagement_IRQn 0 */
  /* USER CODE END MemoryManagement_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_MemoryManagement_IRQn 0 */
    /* USER CODE END W1_MemoryManagement_IRQn 0 */
  }
}
/**
  * @brief This function handles Pre-fetch fault, memory access fault.
  */
void BusFault_Handler(void)
```

```
{
  /* USER CODE BEGIN BusFault_IRQn 0 */
  /* USER CODE END BusFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_BusFault_IRQn 0 */
    /* USER CODE END W1_BusFault_IRQn 0 */
  }
}
/**
  * @brief This function handles Undefined instruction or illegal state.
  */
void UsageFault_Handler(void)
{
  /* USER CODE BEGIN UsageFault_IRQn 0 */
  /* USER CODE END UsageFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_UsageFault_IRQn 0 */
    /* USER CODE END W1_UsageFault_IRQn 0 */
  }
}
/**
```

```
  * @brief This function handles System service call via SWI
instruction.
  */
void SVC_Handler(void)
{
  /* USER CODE BEGIN SVCall_IRQn 0 */
  /* USER CODE END SVCall_IRQn 0 */
  /* USER CODE BEGIN SVCall_IRQn 1 */
  /* USER CODE END SVCall_IRQn 1 */
}
/**
  * @brief This function handles Debug monitor.
  */
void DebugMon_Handler(void)
{
  /* USER CODE BEGIN DebugMonitor_IRQn 0 */
  /* USER CODE END DebugMonitor_IRQn 0 */
  /* USER CODE BEGIN DebugMonitor_IRQn 1 */
  /* USER CODE END DebugMonitor_IRQn 1 */
}
/**
  * @brief This function handles Pendable request for system service.
  */
```

```
void PendSV_Handler(void)
{
  /* USER CODE BEGIN PendSV_IRQn 0 */
  /* USER CODE END PendSV_IRQn 0 */
  /* USER CODE BEGIN PendSV_IRQn 1 */
  /* USER CODE END PendSV_IRQn 1 */
}
/**
  * @brief This function handles System tick timer.
  */
void SysTick_Handler(void)
{
  /* USER CODE BEGIN SysTick_IRQn 0 */
  /* USER CODE END SysTick_IRQn 0 */
  HAL_IncTick();
  /* USER CODE BEGIN SysTick_IRQn 1 */
  /* USER CODE END SysTick_IRQn 1 */
}
/*****************************************************
*********************/
/* STM32F4xx Peripheral Interrupt Handlers                    */
/* Add here the Interrupt Handlers for the used peripherals.  */
/* For the available peripheral interrupt handler names,      */
```

```
/* please refer to the startup file (startup_stm32f4xx.s).                    */
/*****************************************************************************/
/**
  * @brief This function handles USB On The Go FS global interrupt.
  */
void OTG_FS_IRQHandler(void)
{
  /* USER CODE BEGIN OTG_FS_IRQn 0 */
  /* USER CODE END OTG_FS_IRQn 0 */
  HAL_HCD_IRQHandler(&hhcd_USB_OTG_FS);
  /* USER CODE BEGIN OTG_FS_IRQn 1 */
  /* USER CODE END OTG_FS_IRQn 1 */
}
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

31.8 CODE OF SYSCALLS.C:

```
/**
*****************************************************************************
 * @file      syscalls.c
 * @author    Auto-generated by STM32CubeIDE
 * @brief     STM32CubeIDE Minimal System calls file
 *
```

```
 *          For more information about which c-functions
 *          need which of these lowlevel functions
 *          please consult the Newlib libc-manual
**********************************************************
********************
 * @attention
 *
 * Copyright (c) 2020-2024 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the
LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
**********************************************************
********************
 */
/* Includes */
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <time.h>
```

```
#include <sys/time.h>
#include <sys/times.h>
/* Variables */
extern int __io_putchar(int ch) __attribute__((weak));
extern int __io_getchar(void) __attribute__((weak));
char *__env[1] = { 0 };
char **environ = __env;
/* Functions */
void initialise_monitor_handles()
{
}
int _getpid(void)
{
  return 1;
}
int _kill(int pid, int sig)
{
  (void)pid;
  (void)sig;
  errno = EINVAL;
  return -1;
}
void _exit (int status)
```

```
{
  _kill(status, -1);
  while (1) {}    /* Make sure we hang here */
}
__attribute__((weak)) int _read(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;
  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    *ptr++ = __io_getchar();
  }
  return len;
}
__attribute__((weak)) int _write(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;
  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    __io_putchar(*ptr++);
  }
  return len;
```

```
}
int _close(int file)
{
  (void)file;
  return -1;
}
int _fstat(int file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
int _isatty(int file)
{
  (void)file;
  return 1;
}
int _lseek(int file, int ptr, int dir)
{
  (void)file;
  (void)ptr;
  (void)dir;
  return 0;
```

```
}
int _open(char *path, int flags, ...)
{
  (void)path;
  (void)flags;
  /* Pretend like we always fail */
  return -1;
}
int _wait(int *status)
{
  (void)status;
  errno = ECHILD;
  return -1;
}
int _unlink(char *name)
{
  (void)name;
  errno = ENOENT;
  return -1;
}
int _times(struct tms *buf)
{
  (void)buf;
```

```
  return -1;
}
int _stat(char *file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}
int _link(char *old, char *new)
{
  (void)old;
  (void)new;
  errno = EMLINK;
  return -1;
}
int _fork(void)
{
  errno = EAGAIN;
  return -1;
}
int _execve(char *name, char **argv, char **env)
{
  (void)name;
```

```
  (void)argv;
  (void)env;
  errno = ENOMEM;
  return -1;
}
```

31.8 CODE OF SYSMEM.C:

```
/**
*****************************************************************
*******************
 * @file      sysmem.c
 * @author    Generated by STM32CubeIDE
 * @brief     STM32CubeIDE System Memory calls file
 *
 *            For more information about which C functions
 *            need which of these lowlevel functions
 *            please consult the newlib libc manual
*****************************************************************
*******************
 * @attention
 *
 * Copyright (c) 2024 STMicroelectronics.
 * All rights reserved.
 *
```

```
 * This software is licensed under terms that can be found in the
LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
**********************************************************
********************
 */
/* Includes */
#include <errno.h>
#include <stdint.h>
/**
 * Pointer to the current high watermark of the heap usage
 */
static uint8_t *__sbrk_heap_end = NULL;
/**
 * @brief _sbrk() allocates memory to the newlib heap and is used by
malloc
 *        and others from the C library
 *
 * @verbatim
 *
#################################################################
##################
 * #  .data  #  .bss  #       newlib heap       #          MSP stack          #
```

```
 * #        #        #                    # Reserved by _Min_Stack_Size #
 *
##################################################################
###################
 * ^-- RAM start      ^-- _end                          _estack, RAM end --^
 * @endverbatim
 *
 * This implementation starts allocating at the '_end' linker symbol
 * The '_Min_Stack_Size' linker symbol reserves a memory for the MSP
stack
 * The implementation considers '_estack' linker symbol to be RAM end
 * NOTE: If the MSP stack, at any point during execution, grows larger
than the
 * reserved size, please increase the '_Min_Stack_Size'.
 *
 * @param incr Memory size
 * @return Pointer to allocated memory
 */
void *_sbrk(ptrdiff_t incr)
{
  extern uint8_t _end; /* Symbol defined in the linker script */
  extern uint8_t _estack; /* Symbol defined in the linker script */
  extern uint32_t _Min_Stack_Size; /* Symbol defined in the linker
script */
```

```
  const uint32_t stack_limit = (uint32_t)&_estack -
(uint32_t)&_Min_Stack_Size;
  const uint8_t *max_heap = (uint8_t *)stack_limit;
  uint8_t *prev_heap_end;
  /* Initialize heap end at first call */
  if (NULL == __sbrk_heap_end)
  {
    __sbrk_heap_end = &_end;
  }
  /* Protect heap from growing into the reserved MSP stack */
  if (__sbrk_heap_end + incr > max_heap)
  {
    errno = ENOMEM;
    return (void *)-1;
  }
  prev_heap_end = __sbrk_heap_end;
  __sbrk_heap_end += incr;
  return (void *)prev_heap_end;
}
```

31.9 CODE OF SYSTEM_STM32F4XX.C:

```
/**
*****************************************************************
*******************
  * @file    system_stm32f4xx.c
```

```
  * @author  MCD Application Team
  * @brief   CMSIS Cortex-M4 Device Peripheral Access Layer System
Source File.
  *
  *   This file provides two functions and one global variable to be called
from
  *   user application:
  *      - SystemInit(): This function is called at startup just after reset and
  *                      before branch to main program. This call is made inside
  *                      the "startup_stm32f4xx.s" file.
  *
  *      - SystemCoreClock variable: Contains the core clock (HCLK), it
can be used
  *                        by the user application to setup the SysTick
  *                        timer or configure other parameters.
  *
  *      - SystemCoreClockUpdate(): Updates the variable
SystemCoreClock and must
  *                       be called whenever the core clock is changed
  *                       during program execution.
  *
  *
*****************************************************************
*******************
  * @attention
```

```
  *
  * Copyright (c) 2017 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
*****************************************************************
*******************
  */
/** @addtogroup CMSIS
  * @{
  */
/** @addtogroup stm32f4xx_system
  * @{
  */
/** @addtogroup STM32F4xx_System_Private_Includes
  * @{
  */
#include "stm32f4xx.h"
#if !defined  (HSE_VALUE)
```

```
  #define HSE_VALUE    ((uint32_t)25000000) /*!< Default value of
the External oscillator in Hz */

#endif /* HSE_VALUE */

#if !defined  (HSI_VALUE)

  #define HSI_VALUE    ((uint32_t)16000000) /*!< Value of the
Internal oscillator in Hz*/

#endif /* HSI_VALUE */

/**

  * @}

  */

/** @addtogroup STM32F4xx_System_Private_TypesDefinitions

  * @{

  */

/**

  * @}

  */

/** @addtogroup STM32F4xx_System_Private_Defines

  * @{

  */

/************************* Miscellaneous Configuration
************************/

/*!< Uncomment the following line if you need to use external SRAM or
SDRAM as data memory  */

#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx) || defined(STM32F417xx)\
```

|| defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) || defined(STM32F439xx)\

|| defined(STM32F469xx) || defined(STM32F479xx) || defined(STM32F412Zx) || defined(STM32F412Vx)

/* #define DATA_IN_ExtSRAM */

#endif /* STM32F40xxx || STM32F41xxx || STM32F42xxx || STM32F43xxx || STM32F469xx || STM32F479xx ||\

STM32F412Zx || STM32F412Vx */

#if defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) || defined(STM32F439xx)\

|| defined(STM32F446xx) || defined(STM32F469xx) || defined(STM32F479xx)

/* #define DATA_IN_ExtSDRAM */

#endif /* STM32F427xx || STM32F437xx || STM32F429xx || STM32F439xx || STM32F446xx || STM32F469xx ||\

STM32F479xx */

/* Note: Following vector table addresses must be defined in line with linker

configuration. */

/*!< Uncomment the following line if you need to relocate the vector table

anywhere in Flash or Sram, else the vector table is kept at the automatic

remap of boot address selected */

/* #define USER_VECT_TAB_ADDRESS */

#if defined(USER_VECT_TAB_ADDRESS)

/*!< Uncomment the following line if you need to relocate your vector Table

in Sram else user remap will be done in Flash. */

/* #define VECT_TAB_SRAM */

#if defined(VECT_TAB_SRAM)

#define VECT_TAB_BASE_ADDRESS SRAM_BASE /*!< Vector Table base address field.

This value must be a multiple of 0x200. */

#define VECT_TAB_OFFSET 0x00000000U /*!< Vector Table base offset field.

This value must be a multiple of 0x200. */

#else

#define VECT_TAB_BASE_ADDRESS FLASH_BASE /*!< Vector Table base address field.

This value must be a multiple of 0x200. */

#define VECT_TAB_OFFSET 0x00000000U /*!< Vector Table base offset field.

This value must be a multiple of 0x200. */

#endif /* VECT_TAB_SRAM */

#endif /* USER_VECT_TAB_ADDRESS */

/**********/

```
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Macros
  * @{
  */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Variables
  * @{
  */
  /* This variable is updated in three ways:
     1) by calling CMSIS function SystemCoreClockUpdate()
     2) by calling HAL API function HAL_RCC_GetHCLKFreq()
     3) each time HAL_RCC_ClockConfig() is called to configure the system clock frequency
        Note: If you use this function to configure the system clock; then there
              is no need to call the 2 first functions listed above, since SystemCoreClock
              variable is updated automatically.
  */
uint32_t SystemCoreClock = 16000000;
```

```
const uint8_t AHBPrescTable[16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7,
8, 9};
const uint8_t APBPrescTable[8]  = {0, 0, 0, 0, 1, 2, 3, 4};
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_FunctionPrototypes
  * @{
  */
#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  static void SystemInit_ExtMemCtl(void);
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */
/**
  * @}
  */
/** @addtogroup STM32F4xx_System_Private_Functions
  * @{
  */
/**
  * @brief  Setup the microcontroller system
  *         Initialize the FPU setting, vector table location and External
memory
  *         configuration.
  * @param  None
```

```
  * @retval None
  */
void SystemInit(void)
{
  /* FPU settings ------------------------------------------------------------*/
  #if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
    SCB->CPACR |= ((3UL << 10*2)|(3UL << 11*2));  /* set CP10 and
CP11 Full Access */
  #endif
#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  SystemInit_ExtMemCtl();
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */
  /* Configure the Vector Table location -------------------------------------*/
#if defined(USER_VECT_TAB_ADDRESS)
  SCB->VTOR = VECT_TAB_BASE_ADDRESS |
VECT_TAB_OFFSET; /* Vector Table Relocation in Internal SRAM */
#endif /* USER_VECT_TAB_ADDRESS */
}
/**
   * @brief  Update SystemCoreClock variable according to Clock
Register Values.
  *         The SystemCoreClock variable contains the core clock (HCLK),
it can
  *         be used by the user application to setup the SysTick timer or
configure
```

```
  *         other parameters.
  *
  * @note   Each time the core clock (HCLK) changes, this function must be called
  *         to update SystemCoreClock variable value. Otherwise, any configuration
  *         based on this variable will be incorrect.
  *
  * @note   - The system frequency computed by this function is not the real
  *           frequency in the chip. It is calculated based on the predefined
  *           constant and the selected clock source:
  *
  *           - If SYSCLK source is HSI, SystemCoreClock will contain the HSI_VALUE(*)
  *
  *           - If SYSCLK source is HSE, SystemCoreClock will contain the HSE_VALUE(**)
  *
  *           - If SYSCLK source is PLL, SystemCoreClock will contain the HSE_VALUE(**)
  *             or HSI_VALUE(*) multiplied/divided by the PLL factors.
  *
  *         (*) HSI_VALUE is a constant defined in stm32f4xx_hal_conf.h file (default value
```

```
  *            16 MHz) but the real value may vary depending on the variations
  *            in voltage and temperature.
  *
  *        (**) HSE_VALUE is a constant defined in stm32f4xx_hal_conf.h file (its value
  *             depends on the application requirements), user has to ensure that HSE_VALUE
  *             is same as the real frequency of the crystal used. Otherwise, this function
  *             may have wrong result.
  *
  *        - The result of this function could be not correct when using fractional
  *          value for HSE crystal.
  *
  * @param  None
  * @retval None
  */
void SystemCoreClockUpdate(void)
{
  uint32_t tmp = 0, pllvco = 0, pllp = 2, pllsource = 0, pllm = 2;
  /* Get SYSCLK source -------------------------------------------------------
*/
  tmp = RCC->CFGR & RCC_CFGR_SWS;
```

```
  switch (tmp)
  {
    case 0x00:  /* HSI used as system clock source */
      SystemCoreClock = HSI_VALUE;
      break;
    case 0x04:  /* HSE used as system clock source */
      SystemCoreClock = HSE_VALUE;
      break;
    case 0x08:  /* PLL used as system clock source */
      /* PLL_VCO = (HSE_VALUE or HSI_VALUE / PLL_M) * PLL_N
         SYSCLK = PLL_VCO / PLL_P
         */
      pllsource = (RCC->PLLCFGR & RCC_PLLCFGR_PLLSRC) >>
22;
      pllm = RCC->PLLCFGR & RCC_PLLCFGR_PLLM;
      if (pllsource != 0)
      {
        /* HSE used as PLL clock source */
        pllvco = (HSE_VALUE / pllm) * ((RCC->PLLCFGR &
RCC_PLLCFGR_PLLN) >> 6);
      }
      else
      {
        /* HSI used as PLL clock source */
```

```
      pllvco = (HSI_VALUE / pllm) * ((RCC->PLLCFGR &
RCC_PLLCFGR_PLLN) >> 6);
    }
    pllp = (((RCC->PLLCFGR & RCC_PLLCFGR_PLLP) >>16) + 1 )
*2;
    SystemCoreClock = pllvco/pllp;
    break;
   default:
    SystemCoreClock = HSI_VALUE;
    break;
  }
  /* Compute HCLK frequency ----------------------------------------------
-*/
  /* Get HCLK prescaler */
  tmp = AHBPrescTable[((RCC->CFGR & RCC_CFGR_HPRE) >> 4)];
  /* HCLK frequency */
  SystemCoreClock >>= tmp;
}
#if defined (DATA_IN_ExtSRAM) && defined
(DATA_IN_ExtSDRAM)
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
/**
  * @brief  Setup the external memory controller.
```

```
  *        Called in startup_stm32f4xx.s before jump to main.
  *        This function configures the external memories
(SRAM/SDRAM)
  *        This SRAM/SDRAM will be used as program data memory
(including heap and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
  register __IO uint32_t index;
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and
GPIOI interface clock */
  RCC->AHB1ENR |= 0x000001F8;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x00CCC0CC;
  GPIOD->AFR[1]  = 0xCCCCCCCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xAAAA0A8A;
  /* Configure PDx pins speed to 100 MHz */
```

```
GPIOD->OSPEEDR = 0xFFFF0FCF;
/* Configure PDx pins Output type to push-pull */
GPIOD->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PDx pins */
GPIOD->PUPDR   = 0x00000000;
/* Connect PEx pins to FMC Alternate function */
GPIOE->AFR[0]  = 0xC00CC0CC;
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA828A;
/* Configure PEx pins speed to 100 MHz */
GPIOE->OSPEEDR = 0xFFFFC3CF;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0xCCCCCCCC;
GPIOF->AFR[1]  = 0xCCCCCCCC;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA800AAA;
/* Configure PFx pins speed to 50 MHz */
GPIOF->OSPEEDR = 0xAA800AAA;
```

```
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
GPIOF->PUPDR   = 0x00000000;
/* Connect PGx pins to FMC Alternate function */
GPIOG->AFR[0]  = 0xCCCCCCCC;
GPIOG->AFR[1]  = 0xCCCCCCCC;
/* Configure PGx pins in Alternate function mode */
GPIOG->MODER   = 0xAAAAAAAA;
/* Configure PGx pins speed to 50 MHz */
GPIOG->OSPEEDR = 0xAAAAAAAA;
/* Configure PGx pins Output type to push-pull */
GPIOG->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PGx pins */
GPIOG->PUPDR   = 0x00000000;
/* Connect PHx pins to FMC Alternate function */
GPIOH->AFR[0]  = 0x00C0CC00;
GPIOH->AFR[1]  = 0xCCCCCCCC;
/* Configure PHx pins in Alternate function mode */
GPIOH->MODER   = 0xAAAA08A0;
/* Configure PHx pins speed to 50 MHz */
GPIOH->OSPEEDR = 0xAAAA08A0;
/* Configure PHx pins Output type to push-pull */
```

```
  GPIOH->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PHx pins */
  GPIOH->PUPDR   = 0x00000000;
  /* Connect PIx pins to FMC Alternate function */
  GPIOI->AFR[0]  = 0xCCCCCCCC;
  GPIOI->AFR[1]  = 0x00000CC0;
  /* Configure PIx pins in Alternate function mode */
  GPIOI->MODER   = 0x0028AAAA;
  /* Configure PIx pins speed to 50 MHz */
  GPIOI->OSPEEDR = 0x0028AAAA;
  /* Configure PIx pins Output type to push-pull */
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;
/*-- FMC Configuration ------------------------------------------------------*/
  /* Enable the FMC interface clock */
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
  FMC_Bank5_6->SDTR[0] = 0x01115351;
  /* SDRAM initialization sequence */
  /* Clock enable command */
```

```
FMC_Bank5_6->SDCMR = 0x00000011;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* Delay */
for (index = 0; index<1000; index++);
/* PALL command */
FMC_Bank5_6->SDCMR = 0x00000012;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
}
/* Auto refresh command */
FMC_Bank5_6->SDCMR = 0x00000073;
tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
timeout = 0xFFFF;
while((tmpreg != 0) && (timeout-- > 0))
{
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
```

```
  }
  /* MRD register program */
  FMC_Bank5_6->SDCMR = 0x00046014;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Set refresh count */
  tmpreg = FMC_Bank5_6->SDRTR;
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
  /* Disable write protection */
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
```

```
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F469xx || STM32F479xx */
  (void)(tmp);
}
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F469xx || STM32F479xx */
#elif defined (DATA_IN_ExtSRAM) || defined
(DATA_IN_ExtSDRAM)
/**
  * @brief  Setup the external memory controller.
  *         Called in startup_stm32f4xx.s before jump to main.
  *         This function configures the external memories
(SRAM/SDRAM)
  *         This SRAM/SDRAM will be used as program data memory
(including heap and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;
```

```
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F446xx) || defined(STM32F469xx) ||
defined(STM32F479xx)
#if defined (DATA_IN_ExtSDRAM)
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
  register __IO uint32_t index;
#if defined(STM32F446xx)
  /* Enable GPIOA, GPIOC, GPIOD, GPIOE, GPIOF, GPIOG interface
     clock */
  RCC->AHB1ENR |= 0x0000007D;
#else
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and
GPIOI interface
     clock */
  RCC->AHB1ENR |= 0x000001F8;
#endif /* STM32F446xx */
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);
#if defined(STM32F446xx)
  /* Connect PAx pins to FMC Alternate function */
  GPIOA->AFR[0]  |= 0xC0000000;
  GPIOA->AFR[1]  |= 0x00000000;
  /* Configure PDx pins in Alternate function mode */
```

```
  GPIOA->MODER   |= 0x00008000;
  /* Configure PDx pins speed to 50 MHz */
  GPIOA->OSPEEDR |= 0x00008000;
  /* Configure PDx pins Output type to push-pull */
  GPIOA->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOA->PUPDR   |= 0x00000000;
  /* Connect PCx pins to FMC Alternate function */
  GPIOC->AFR[0]  |= 0x00CC0000;
  GPIOC->AFR[1]  |= 0x00000000;
  /* Configure PDx pins in Alternate function mode */
  GPIOC->MODER   |= 0x00000A00;
  /* Configure PDx pins speed to 50 MHz */
  GPIOC->OSPEEDR |= 0x00000A00;
  /* Configure PDx pins Output type to push-pull */
  GPIOC->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOC->PUPDR   |= 0x00000000;
#endif /* STM32F446xx */
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x000000CC;
  GPIOD->AFR[1]  = 0xCC000CCC;
  /* Configure PDx pins in Alternate function mode */
```

```
GPIOD->MODER   = 0xA02A000A;
/* Configure PDx pins speed to 50 MHz */
GPIOD->OSPEEDR = 0xA02A000A;
/* Configure PDx pins Output type to push-pull */
GPIOD->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PDx pins */
GPIOD->PUPDR   = 0x00000000;
/* Connect PEx pins to FMC Alternate function */
GPIOE->AFR[0]  = 0xC00000CC;
GPIOE->AFR[1]  = 0xCCCCCCCC;
/* Configure PEx pins in Alternate function mode */
GPIOE->MODER   = 0xAAAA800A;
/* Configure PEx pins speed to 50 MHz */
GPIOE->OSPEEDR = 0xAAAA800A;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0xCCCCCCCC;
GPIOF->AFR[1]  = 0xCCCCCCCC;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA800AAA;
```

```
  /* Configure PFx pins speed to 50 MHz */
  GPIOF->OSPEEDR = 0xAA800AAA;
  /* Configure PFx pins Output type to push-pull */
  GPIOF->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PFx pins */
  GPIOF->PUPDR   = 0x00000000;
  /* Connect PGx pins to FMC Alternate function */
  GPIOG->AFR[0]  = 0xCCCCCCCC;
  GPIOG->AFR[1]  = 0xCCCCCCCC;
  /* Configure PGx pins in Alternate function mode */
  GPIOG->MODER   = 0xAAAAAAAA;
  /* Configure PGx pins speed to 50 MHz */
  GPIOG->OSPEEDR = 0xAAAAAAAA;
  /* Configure PGx pins Output type to push-pull */
  GPIOG->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
  /* Connect PHx pins to FMC Alternate function */
  GPIOH->AFR[0]  = 0x00C0CC00;
  GPIOH->AFR[1]  = 0xCCCCCCCC;
```

```
  /* Configure PHx pins in Alternate function mode */
  GPIOH->MODER   = 0xAAAA08A0;
  /* Configure PHx pins speed to 50 MHz */
  GPIOH->OSPEEDR = 0xAAAA08A0;
  /* Configure PHx pins Output type to push-pull */
  GPIOH->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PHx pins */
  GPIOH->PUPDR   = 0x00000000;
  /* Connect PIx pins to FMC Alternate function */
  GPIOI->AFR[0]  = 0xCCCCCCCC;
  GPIOI->AFR[1]  = 0x00000CC0;
  /* Configure PIx pins in Alternate function mode */
  GPIOI->MODER   = 0x0028AAAA;
  /* Configure PIx pins speed to 50 MHz */
  GPIOI->OSPEEDR = 0x0028AAAA;
  /* Configure PIx pins Output type to push-pull */
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F469xx || STM32F479xx */
/*-- FMC Configuration ------------------------------------------------------*/
  /* Enable the FMC interface clock */
```

```
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable SDRAM bank1 */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCR[0] = 0x00001954;
#else
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
#endif /* STM32F446xx */
  FMC_Bank5_6->SDTR[0] = 0x01115351;
  /* SDRAM initialization sequence */
  /* Clock enable command */
  FMC_Bank5_6->SDCMR = 0x00000011;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Delay */
  for (index = 0; index<1000; index++);
  /* PALL command */
  FMC_Bank5_6->SDCMR = 0x00000012;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
```

```
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Auto refresh command */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x000000F3;
#else
  FMC_Bank5_6->SDCMR = 0x00000073;
#endif /* STM32F446xx */
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* MRD register program */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x00044014;
#else
  FMC_Bank5_6->SDCMR = 0x00046014;
#endif /* STM32F446xx */
```

```
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
  /* Set refresh count */
  tmpreg = FMC_Bank5_6->SDRTR;
#if defined(STM32F446xx)
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000050C<<1));
#else
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
#endif /* STM32F446xx */
  /* Disable write protection */
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#endif /* DATA_IN_ExtSDRAM */
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx || STM32F446xx || STM32F469xx || STM32F479xx */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx) || defined(STM32F417xx)\
 || defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)\
```

```
 || defined(STM32F469xx) || defined(STM32F479xx) ||
defined(STM32F412Zx) || defined(STM32F412Vx)
#if defined(DATA_IN_ExtSRAM)
/*-- GPIOs Configuration -----------------------------------------------------*/
  /* Enable GPIOD, GPIOE, GPIOF and GPIOG interface clock */
  RCC->AHB1ENR   |= 0x00000078;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIODEN);
  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x00CCC0CC;
  GPIOD->AFR[1]  = 0xCCCCCCCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xAAAA0A8A;
  /* Configure PDx pins speed to 100 MHz */
  GPIOD->OSPEEDR = 0xFFFF0FCF;
  /* Configure PDx pins Output type to push-pull */
  GPIOD->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOD->PUPDR   = 0x00000000;
  /* Connect PEx pins to FMC Alternate function */
  GPIOE->AFR[0]  = 0xC00CC0CC;
  GPIOE->AFR[1]  = 0xCCCCCCCC;
  /* Configure PEx pins in Alternate function mode */
```

```
GPIOE->MODER   = 0xAAAA828A;
/* Configure PEx pins speed to 100 MHz */
GPIOE->OSPEEDR = 0xFFFFC3CF;
/* Configure PEx pins Output type to push-pull */
GPIOE->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PEx pins */
GPIOE->PUPDR   = 0x00000000;
/* Connect PFx pins to FMC Alternate function */
GPIOF->AFR[0]  = 0x00CCCCCC;
GPIOF->AFR[1]  = 0xCCCC0000;
/* Configure PFx pins in Alternate function mode */
GPIOF->MODER   = 0xAA000AAA;
/* Configure PFx pins speed to 100 MHz */
GPIOF->OSPEEDR = 0xFF000FFF;
/* Configure PFx pins Output type to push-pull */
GPIOF->OTYPER  = 0x00000000;
/* No pull-up, pull-down for PFx pins */
GPIOF->PUPDR   = 0x00000000;
/* Connect PGx pins to FMC Alternate function */
GPIOG->AFR[0]  = 0x00CCCCCC;
GPIOG->AFR[1]  = 0x000000C0;
/* Configure PGx pins in Alternate function mode */
GPIOG->MODER   = 0x00085AAA;
```

```
  /* Configure PGx pins speed to 100 MHz */
  GPIOG->OSPEEDR = 0x000CAFFF;
  /* Configure PGx pins Output type to push-pull */
  GPIOG->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;
/*-- FMC/FSMC Configuration ----------------------------------------------
-*/
  /* Enable the FMC/FSMC interface clock */
  RCC->AHB3ENR         |= 0x00000001;
#if defined(STM32F427xx) || defined(STM32F437xx) ||
defined(STM32F429xx) || defined(STM32F439xx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx ||
STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
```

```
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F469xx || STM32F479xx */
#if defined(STM32F405xx) || defined(STM32F415xx) ||
defined(STM32F407xx)|| defined(STM32F417xx)\
   || defined(STM32F412Zx) || defined(STM32F412Vx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FSMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FSMC_Bank1->BTCR[2]  = 0x00001011;
  FSMC_Bank1->BTCR[3]  = 0x00000201;
  FSMC_Bank1E->BWTR[2] = 0x0FFFFFFF;
#endif /* STM32F405xx || STM32F415xx || STM32F407xx ||
STM32F417xx || STM32F412Zx || STM32F412Vx */
#endif /* DATA_IN_ExtSRAM */
#endif /* STM32F405xx || STM32F415xx || STM32F407xx ||
STM32F417xx || STM32F427xx || STM32F437xx ||\
         STM32F429xx || STM32F439xx || STM32F469xx ||
STM32F479xx || STM32F412Zx || STM32F412Vx  */
  (void)(tmp);
}
#endif /* DATA_IN_ExtSRAM && DATA_IN_ExtSDRAM */
/**
```

```
  * @}
  */
/**
  * @}
  */
/**
  * @}
  */
```

CHAPTER#32 CODING AN ADC DRIVER FOR CONTINUOUS MODE CONVERSION

32.1CODING AN ADC DRIVER FOR CONTINUOUS MODE CONVERSION:

To code an ADC (Analog-to-Digital Converter) driver for continuous mode conversion, you need to set up the ADC to continuously sample the analog input signal and convert it into digital data. Here's a general approach to coding such a driver, with specific implementation details depending on the microcontroller and development environment you're using. I'll provide a general C/C++ code outline that can be adapted for different platforms:

32.1.1 GENERAL STEPS:

32.1.1.1 INITIALIZE THE ADC PERIPHERAL:

1: Configure the ADC clock

2: Set the ADC resolution (e.g., 8-bit, 12-bit).

3: Select the input channel.

4: Set the reference voltage.

5: Enable continuous conversion mode

31.1.1.2 CONFIGURE GPIO:

Set up the GPIO pin(s) for the ADC input.

31.1.1.3 START ADC CONVERSION:

Start the ADC in continuous conversion mode.

31.1.1.4 READ ADC DATA:

Implement a method to read the converted data, either through polling or interrupt-based approach.

31.1.1.5 HANDLE CONVERSION DATA:

Process the ADC data as required by your application.

31.2 CODE OF ADC DRIVER:

```
#include "stm32f4xx_hal.h"  // Include the appropriate HAL header file for your MCU

ADC_HandleTypeDef hadc1;    // Define an ADC handle

// Function to initialize ADC

void ADC_Init(void) {

    __HAL_RCC_ADC1_CLK_ENABLE();  // Enable the ADC clock

    hadc1.Instance = ADC1;  // Define the ADC instance

    hadc1.Init.Resolution = ADC_RESOLUTION_12B;  // Set the resolution (12-bit in this example)
```

```
    hadc1.Init.ContinuousConvMode = ENABLE;  // Enable continuous conversion mode
    hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;  // Align data to the right
    hadc1.Init.ScanConvMode = DISABLE;  // Disable scan mode for single channel
    hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;  // Start conversion by software
    if (HAL_ADC_Init(&hadc1) != HAL_OK) {
        // Initialization Error
        Error_Handler();
    }
    // Configure ADC regular channel
    ADC_ChannelConfTypeDef sConfig = {0};
    sConfig.Channel = ADC_CHANNEL_0;  // Select the ADC channel (channel 0 in this example)
    sConfig.Rank = 1;  // Set the rank (1 for first conversion)
    sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;  // Set the sampling time
    if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK) {
        // Channel Configuration Error
        Error_Handler();
    }
}
// Function to start ADC conversion in continuous mode
```

```
void Start_ADC_Conversion(void) {
    if (HAL_ADC_Start(&hadc1) != HAL_OK) {
        // Start Error
        Error_Handler();
    }
}
// Function to read ADC converted value
uint32_t Read_ADC_Value(void) {
    if (HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY)
== HAL_OK) {
        return HAL_ADC_GetValue(&hadc1);  // Return the ADC
converted value
    }
    return 0;
}
// Main function
int main(void) {
    HAL_Init();  // Initialize the HAL Library
    SystemClock_Config();  // Configure the system clock
    ADC_Init();  // Initialize the ADC
    Start_ADC_Conversion();  // Start the ADC conversion
    while (1) {
        uint32_t adcValue = Read_ADC_Value();  // Read the ADC value
        // Process the ADC value as needed
```

```
    }
}
// Error handler function
void Error_Handler(void) {
    while (1) {
        // Stay here in case of error
    }
}
```

31.3 KEY CONSIDERATIONS:

31.3.1 MCU-SPECIFIC LIBRARIES:

The example uses STM32 HAL libraries. Ensure you include the correct libraries for your microcontroller.

31.3.2 ADC CONFIGURATION:

Set parameters like resolution, sampling time, and conversion mode according to your specific requirements.

31.3.3 POLLING VS INTERRUPT:

The example uses polling (HAL_ADC_PollForConversion). For lower latency or CPU load, consider using interrupt-driven conversion handling.

31.3.4 ERROR HANDLING:

Implement proper error handling for robustness.

CHAPTER#32 CODING AN ADC DRIVER FOR INTERRUPT CONVERSION MODES

32.1CODING AN ADC DRIVER FOR INTERRUPT CONVERSION MODES:

To code an ADC (Analog-to-Digital Converter) driver for interrupt-based conversion modes, you'll configure the ADC to generate an interrupt when a conversion is complete. This method is efficient as it allows the CPU to perform other tasks and only handles the ADC data when an interrupt occurs.

32.2 KEY COMPONENTS:

32.2.1 ADC INITILIZATION:

1: Set up the ADC configuration.

2: Configure the channel(s) to be sampled

3: Set the sampling time.

32.2.2 INTERRUPT CONFIGURATION:

1: Enable the ADC interrupt in the NVIC (Nested Vector Interrupt Controller).

2: Define an interrupt service routine (ISR) to handle the ADC conversion complete event.

32.2.3 ADC START:

Start the ADC conversion and enable interrupts.

32.2.4 ISR HANDLING:

Read the ADC data in the ISR and process or store it as needed.

33.3 CODE OF ADC DRIVER FOR INTERRUPT CONVERSION MODES:

```
#include "stm32f4xx_hal.h"  // Include the appropriate HAL header file for your MCU

ADC_HandleTypeDef hadc1;    // Define an ADC handle

uint32_t adcValue;          // Variable to store the ADC value

// Function to initialize ADC

void ADC_Init(void) {

    __HAL_RCC_ADC1_CLK_ENABLE();  // Enable the ADC clock

    hadc1.Instance = ADC1;  // Define the ADC instance

    hadc1.Init.Resolution = ADC_RESOLUTION_12B;  // Set the resolution (12-bit in this example)

    hadc1.Init.ContinuousConvMode = DISABLE;  // Disable continuous conversion mode

    hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;  // Align data to the right

    hadc1.Init.ScanConvMode = DISABLE;  // Disable scan mode for single channel

    hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;  // Start conversion by software

    if (HAL_ADC_Init(&hadc1) != HAL_OK) {

        // Initialization Error

        Error_Handler();
```

```
    }
    // Configure ADC regular channel
    ADC_ChannelConfTypeDef sConfig = {0};
    sConfig.Channel = ADC_CHANNEL_0;  // Select the ADC channel
(channel 0 in this example)
    sConfig.Rank = 1;  // Set the rank (1 for first conversion)
    sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;  // Set
the sampling time
    if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK) {
        // Channel Configuration Error
        Error_Handler();
    }
}
// Function to start ADC conversion with interrupt
void Start_ADC_Conversion_IT(void) {
    if (HAL_ADC_Start_IT(&hadc1) != HAL_OK) {
        // Start Error
        Error_Handler();
    }
}
// Interrupt Service Routine for ADC conversion complete
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc) {
    if (hadc->Instance == ADC1) {
```

```
        adcValue = HAL_ADC_GetValue(hadc);  // Read the ADC converted value
        // Process the ADC value as needed
    }
}
// Main function
int main(void) {
    HAL_Init();  // Initialize the HAL Library
    SystemClock_Config();  // Configure the system clock
    ADC_Init();  // Initialize the ADC
    HAL_NVIC_SetPriority(ADC_IRQn, 0, 0);  // Set ADC interrupt priority
    HAL_NVIC_EnableIRQ(ADC_IRQn);  // Enable ADC interrupt in NVIC
    Start_ADC_Conversion_IT();  // Start the ADC conversion with interrupt
    while (1) {
        // Main loop can perform other tasks
    }
}
// ADC interrupt handler
void ADC_IRQHandler(void) {
    HAL_ADC_IRQHandler(&hadc1);  // Handle ADC interrupt
}
```

```
// Error handler function
void Error_Handler(void) {
    while (1) {
        // Stay here in case of error
    }
}
```

33.4 KEY POINTS:

33.4.1 ADC INITILIZATION:

This includes configuring the ADC instance, resolution, alignment, and other settings. The ADC is configured to start conversion by software (ADC_SOFTWARE_START).

33.4.2 START ADC CONVERSION WITH INTERRUPTS:

The HAL_ADC_Start_IT function is used to start the ADC conversion and enable interrupts for conversion completion.

33.4.3 INTERRUPT SERVICES ROUTINE(ISR):

The HAL_ADC_ConvCpltCallback function is a callback that gets executed when an ADC conversion is complete. This function reads the converted value and can be used to trigger further processing.

33.4.4 NVIC CONFIGURATION:

NVIC (Nested Vector Interrupt Controller) is configured to handle ADC interrupts.

34.4.5 ERROR HANDLING:

Implement robust error handling to manage initialization and runtime issues.

34.5 PLATFORM-SPECIFIC CONFIGURATION:

34.5.1 INTERRUPT PRIORITY:

Set according to your system's requirements.

34.5.2 NVIC AND ISR NAMES:

These may differ based on the MCU and its associated libraries.

CHAPTER#35 CODING AN ADC DRIVER FOR SIGNAL-CHANNEL DMA MODE

35.1CODING AN ADC DRIVER FOR SIGNAL-CHANNEL DMA MODE:

Coding an ADC driver for single-channel DMA (Direct Memory Access) mode involves setting up the ADC and DMA to automatically transfer ADC conversion results to a memory buffer. This approach minimizes CPU involvement, allowing efficient and fast data handling, which is particularly useful for high-speed or continuous data acquisition systems.

35.2 KEY COMPONENTS:

35.2.1 ADC INITILIZATION:

Set up the ADC configuration (resolution, channel, etc.). Configure the ADC to trigger DMA transfers.

35.2.2 DMA INITILIZATION:

Set up DMA to transfer data from the ADC to a memory buffer. Configure the number of data items and the data width.

35.2.3 START ADC AND DMA:

Start the ADC conversion and enable DMA.

35.2.4 BUFFER MANAGEMENT:

Manage the data in the buffer as new ADC results are stored.

35.3 CODE OF CODING AN ADC DRIVER FOR SIGNAL-CHANNEL DMA MODE:

```
#include "stm32f4xx_hal.h"  // Include the appropriate HAL header file for your MCU

#define ADC_BUFFER_SIZE 100  // Define the size of the buffer

ADC_HandleTypeDef hadc1;    // ADC handle

DMA_HandleTypeDef hdma_adc1; // DMA handle

uint32_t adcBuffer[ADC_BUFFER_SIZE];  // Buffer to store ADC results

// Function to initialize ADC

void ADC_Init(void) {

    __HAL_RCC_ADC1_CLK_ENABLE();  // Enable the ADC clock

    __HAL_RCC_DMA2_CLK_ENABLE();  // Enable the DMA clock (DMA2 used in STM32F4)

    // Initialize ADC instance

    hadc1.Instance = ADC1;
```

```
    hadc1.Init.Resolution = ADC_RESOLUTION_12B;  // Set resolution
to 12 bits
    hadc1.Init.ScanConvMode = DISABLE;  // Single channel mode
    hadc1.Init.ContinuousConvMode = ENABLE;  // Continuous
conversion mode
    hadc1.Init.DiscontinuousConvMode = DISABLE;
    hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;  // Start
conversion by software
    hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;  // Right data
alignment
    hadc1.Init.NbrOfConversion = 1;  // One conversion at a time
    if (HAL_ADC_Init(&hadc1) != HAL_OK) {
        // Initialization Error
        Error_Handler();
    }
    // Configure ADC regular channel
    ADC_ChannelConfTypeDef sConfig = {0};
    sConfig.Channel = ADC_CHANNEL_0;  // Select the ADC channel
(channel 0 in this example)
    sConfig.Rank = 1;  // Rank 1 in regular group
    sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;  //
Sampling time
    if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK) {
        // Channel Configuration Error
        Error_Handler();
```

```
    }
}
// Function to initialize DMA
void DMA_Init(void) {
    // Initialize DMA for ADC
    hdma_adc1.Instance = DMA2_Stream0;
    hdma_adc1.Init.Channel = DMA_CHANNEL_0;
    hdma_adc1.Init.Direction = DMA_PERIPH_TO_MEMORY;
    hdma_adc1.Init.PeriphInc = DMA_PINC_DISABLE;
    hdma_adc1.Init.MemInc = DMA_MINC_ENABLE;
    hdma_adc1.Init.PeriphDataAlignment =
DMA_PDATAALIGN_HALFWORD;
    hdma_adc1.Init.MemDataAlignment =
DMA_MDATAALIGN_HALFWORD;
    hdma_adc1.Init.Mode = DMA_CIRCULAR;  // Circular mode
    hdma_adc1.Init.Priority = DMA_PRIORITY_HIGH;
    hdma_adc1.Init.FIFOMode = DMA_FIFOMODE_DISABLE;
    if (HAL_DMA_Init(&hdma_adc1) != HAL_OK) {
        // Initialization Error
        Error_Handler();
    }
    // Link DMA to ADC
    __HAL_LINKDMA(&hadc1, DMA_Handle, hdma_adc1);
    // Configure NVIC for DMA
```

```
    HAL_NVIC_SetPriority(DMA2_Stream0_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(DMA2_Stream0_IRQn);
}
// Function to start ADC and DMA
void Start_ADC_DMA(void) {
    if (HAL_ADC_Start_DMA(&hadc1, adcBuffer,
ADC_BUFFER_SIZE) != HAL_OK) {
        // Start Error
        Error_Handler();
    }
}
// Main function
int main(void) {
    HAL_Init();  // Initialize the HAL Library
    SystemClock_Config();  // Configure the system clock
    ADC_Init();  // Initialize the ADC
    DMA_Init();  // Initialize DMA
    Start_ADC_DMA();  // Start ADC with DMA
    while (1) {
        // Main loop can perform other tasks
        // Data from ADC is being stored in adcBuffer
    }
}
```

```
// DMA interrupt handler
void DMA2_Stream0_IRQHandler(void) {
    HAL_DMA_IRQHandler(&hdma_adc1);  // Handle DMA interrupt
}
// Error handler function
void Error_Handler(void) {
    while (1) {
        // Stay here in case of error
    }
}
```

35.4 KEY POINTS:

35.4.1 ADC AND DMA INITILIZATION:

The ADC is configured with continuous conversion mode and linked to the DMA. The DMA is set up to transfer data from the ADC to the memory buffer (adcBuffer), using a circular mode for continuous data acquisition.

35.4.2 DMA CONFIGURATION:

The DMA is configured for peripheral-to-memory transfer, with the ADC data alignment set to half-word (16-bit).

35.4.3 BUFFER MANAGEMENT:

The buffer size (ADC_BUFFER_SIZE) and handling depend on your application's needs. The DMA operates in circular mode, which means it

will wrap around to the beginning of the buffer when it reaches the end, allowing continuous data acquisition without manual intervention.

35.4.4 INTERRUPT HANDLING:

The DMA interrupt is handled by the DMA2_Stream0_IRQHandler function, which calls HAL_DMA_IRQHandler to handle the transfer complete or error events.

35.4.5 ERROR HANDLING:

The Error_Handler function is used to manage errors that occur during initialization or operation.

35.5 PLATFORM-SPECIFIC CONSIDERATIONS:

35.5.1 PERIPHERAL CLOCKS:

Ensure the correct peripherals and clocks are enabled.

35.5.2 DMA STREAMS/CHANNELS:

The DMA stream and channel configurations will vary based on the specific microcontroller.

35.5.3 DATA ALIGNMENT AND SIZE:

The DMA stream and channel configurations will vary based on the specific microcontroller.

CHAPTER#36 CODING AN ADC DRIVER FOR MULTI-CHANNEL DMA SCAN MODULE

36.1CODING AN ADC DRIVER FOR MULTI-CHANNEL DMA SCAN MODULE:

To code an ADC driver for a multi-channel DMA scan mode, the ADC is configured to scan multiple channels, and the DMA transfers the converted data from these channels into a buffer in memory. This method allows for efficient and automated data acquisition from multiple analog inputs without requiring CPU intervention for each conversion.

36.2 KEY COMPONENTS:

36.2.1 ADC INITILIZATION:

Configure the ADC to scan multiple channels. Set the resolution, alignment, and sampling time for each channel.

36.2.2 DMA INITILIZATION:

Set up DMA to transfer the data from the ADC to a buffer. Configure DMA to handle the data size and number of channels.

36.2.3 START ADC AND DMA:

Start the ADC in DMA mode to begin continuous data transfer.

36.2.4 BUFFER MANAGEMENT:

The buffer is used to store ADC data from all channels. The data can be processed or read after acquisition.

36.3 CODE OF CODING AN ADC DRIVER FOR MULTI-CHANNEL DMA SCAN MODULE:

```
#include "stm32f4xx_hal.h"  // Include the appropriate HAL header file for your MCU

#define ADC_CHANNEL_COUNT 3  // Number of ADC channels to scan

#define ADC_BUFFER_SIZE (ADC_CHANNEL_COUNT * 100)  // Buffer size (channels * number of samples)

ADC_HandleTypeDef hadc1;    // ADC handle

DMA_HandleTypeDef hdma_adc1; // DMA handle

uint32_t adcBuffer[ADC_BUFFER_SIZE];  // Buffer to store ADC results

// Function to initialize ADC

void ADC_Init(void) {

    __HAL_RCC_ADC1_CLK_ENABLE();  // Enable the ADC clock

    __HAL_RCC_DMA2_CLK_ENABLE();  // Enable the DMA clock (DMA2 used in STM32F4)

    // Initialize ADC instance

    hadc1.Instance = ADC1;

    hadc1.Init.Resolution = ADC_RESOLUTION_12B;  // Set resolution to 12 bits

    hadc1.Init.ScanConvMode = ENABLE;  // Enable scan mode for multiple channels

    hadc1.Init.ContinuousConvMode = ENABLE;  // Continuous conversion mode
```

```
    hadc1.Init.DiscontinuousConvMode = DISABLE;
    hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;  // Start
conversion by software
    hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;  // Right data
alignment
    hadc1.Init.NbrOfConversion = ADC_CHANNEL_COUNT;  //
Number of channels to convert
    if (HAL_ADC_Init(&hadc1) != HAL_OK) {
        // Initialization Error
        Error_Handler();
    }
    // Configure each ADC channel
    ADC_ChannelConfTypeDef sConfig = {0};
    for (uint32_t i = 0; i < ADC_CHANNEL_COUNT; i++) {
        sConfig.Channel = ADC_CHANNEL_0 + i;  // Select the ADC
channel
        sConfig.Rank = i + 1;  // Rank in the regular group
        sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;  //
Sampling time
        if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK) {
            // Channel Configuration Error
            Error_Handler();
        }
    }
}
```

```
// Function to initialize DMA
void DMA_Init(void) {
    // Initialize DMA for ADC
    hdma_adc1.Instance = DMA2_Stream0;
    hdma_adc1.Init.Channel = DMA_CHANNEL_0;
    hdma_adc1.Init.Direction = DMA_PERIPH_TO_MEMORY;
    hdma_adc1.Init.PeriphInc = DMA_PINC_DISABLE;
    hdma_adc1.Init.MemInc = DMA_MINC_ENABLE;
    hdma_adc1.Init.PeriphDataAlignment =
DMA_PDATAALIGN_HALFWORD;
    hdma_adc1.Init.MemDataAlignment =
DMA_MDATAALIGN_HALFWORD;
    hdma_adc1.Init.Mode = DMA_CIRCULAR;  // Circular mode
    hdma_adc1.Init.Priority = DMA_PRIORITY_HIGH;
    hdma_adc1.Init.FIFOMode = DMA_FIFOMODE_DISABLE;
    if (HAL_DMA_Init(&hdma_adc1) != HAL_OK) {
        // Initialization Error
        Error_Handler();
    }
    // Link DMA to ADC
    __HAL_LINKDMA(&hadc1, DMA_Handle, hdma_adc1);
    // Configure NVIC for DMA
    HAL_NVIC_SetPriority(DMA2_Stream0_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(DMA2_Stream0_IRQn);
```

```
}
// Function to start ADC and DMA
void Start_ADC_DMA(void) {
   if (HAL_ADC_Start_DMA(&hadc1, adcBuffer,
ADC_BUFFER_SIZE) != HAL_OK) {
      // Start Error
      Error_Handler();
   }
}
// Main function
int main(void) {
   HAL_Init();  // Initialize the HAL Library
   SystemClock_Config();  // Configure the system clock
   ADC_Init();  // Initialize the ADC
   DMA_Init();  // Initialize DMA
   Start_ADC_DMA();  // Start ADC with DMA
   while (1) {
      // Main loop can perform other tasks
      // Data from ADC is being stored in adcBuffer
   }
}
// DMA interrupt handler
void DMA2_Stream0_IRQHandler(void) {
```

```
    HAL_DMA_IRQHandler(&hdma_adc1);  // Handle DMA interrupt
}
// Error handler function
void Error_Handler(void) {
    while (1) {
        // Stay here in case of error
    }
}
```

36.4 KEY POINTS:

36.4.1 ADC CONFIGURATION:

The ADC is set up in scan mode to convert multiple channels in sequence. The number of conversions (NbrOfConversion) is set to the number of channels to be scanned. Each channel's configuration (such as sampling time) is set individually.

36.4.2 DMA CONFIGURATION:

DMA is configured to transfer data from the ADC to a buffer (adcBuffer) in memory. The buffer size is calculated based on the number of channels and the number of samples to be stored.

36.4.3 CIRCULAR MODE:

The DMA is set to circular mode, allowing continuous data acquisition by wrapping around to the beginning of the buffer when it reaches the end.

36.4.4 INTERRUPT HANDLING:

The DMA interrupt handler manages the completion of data transfers and any errors.

36.4.5 BUFFER MANAGEMENT:

The data in adcBuffer needs to be managed and processed according to your application's requirements.

36.5 PLATFORM-SPECIFIC CONSIDERTIONS:

36.5.1 CHANNEL AND STREAM SELECTION:

Adjust the ADC channel and DMA stream/channel according to your microcontroller.

36.5.2 DATA ALIGNMENT AND SIZE:

Ensure that data alignment and size are set according to the ADC resolution.

36.5.3 NVIC CONFIGURATION:

The NVIC configuration will depend on the specific microcontroller and its interrupt priority requirements.

CHAPTER#37 CODING SPI DRIVERS FOR POLLING TRANSFER MODE

37.1CODING SPI DRIVERS FOR POLLING TRANSFER MODE:

Coding SPI (Serial Peripheral Interface) drivers for polling transfer mode involves directly managing data transfer between the master and slave devices without using interrupts or DMA (Direct Memory Access).

37.2 SETUP AND CONFIGURATION:

37.2.1 DEFINE SPI CONFIGURATION PARAMETERS:

Set up parameters such as SPI mode (clock polarity and phase), clock speed, bit order, and data frame size.

37.2.2 INITIALIZE SPI PERIPHERAL:

Configure the hardware registers for the SPI peripheral with the defined parameters.

37.3 DATA TRANSFER FUNCTIONS:

37.3.1 TRANSMIT DATA(POLLING):

Write data to the SPI data register and poll the status register until the data is transmitted.

37.3.2 RECEIVE DATA(POLLING):

Poll the status register to check if data has been received, then read the data from the SPI data register.

37.4 POLLING MECHANISM:

37.4.1 WAIT FOR TRANSMISSION/RECEPTION TO COMPLETE:

Continuously check the status register bits to determine if the transmission or reception is complete.

37.4.2 ERROR HANDLING:

Check for any errors during the transmission, such as overrun or framing errors.

37.5 CODE OF CODING SPI DRIVERS FOR POLLING TRANSFER MODE:

```
#include <stdint.h>

// Define SPI registers (placeholders)

#define SPI_BASE_ADDR    0x40013000

#define SPI_CR1          (*(volatile uint32_t *)(SPI_BASE_ADDR +
0x00))

#define SPI_SR           (*(volatile uint32_t *)(SPI_BASE_ADDR +
0x08))

#define SPI_DR           (*(volatile uint32_t *)(SPI_BASE_ADDR +
0x0C))

// SPI status register flags

#define SPI_SR_TXE       (1 << 1)  // Transmit buffer empty

#define SPI_SR_RXNE      (1 << 0)  // Receive buffer not empty

#define SPI_SR_BSY       (1 << 7)  // Busy flag
```

```
// SPI configuration
void spi_init() {
    // Configure SPI (e.g., set baud rate, clock polarity, phase, etc.)
    SPI_CR1 = /* configuration settings */;
    SPI_CR1 |= (1 << 6);  // Enable SPI
}
// Transmit data (polling mode)
void spi_transmit(uint8_t data) {
    while (!(SPI_SR & SPI_SR_TXE));  // Wait for transmit buffer to be
empty
    SPI_DR = data;                   // Write data to the data register
    while (SPI_SR & SPI_SR_BSY);     // Wait until SPI is not busy
}
// Receive data (polling mode)
uint8_t spi_receive() {
    while (!(SPI_SR & SPI_SR_RXNE)); // Wait for data to be received
    return (uint8_t)SPI_DR;          // Read received data from the data
register
}
int main() {
    spi_init();  // Initialize SPI
    // Example usage
    spi_transmit(0xAB);  // Transmit a byte
    uint8_t received_data = spi_receive();  // Receive a byte
```

```
    // Further processing...
    return 0;
}
```

CHAPTER#38 CODING AN SPI DRIVERS FOR INTERRUPT TRANSFER MODE

38.1CODING AN SPI DRIVERS FOR INTERRUPT TRANSFER MODE:

Coding an SPI driver for interrupt transfer mode involves using hardware interrupts to handle data transfer events, allowing the CPU to perform other tasks while waiting for the transfer to complete. This approach can improve efficiency and responsiveness in embedded systems.

38.2 SETUP AND CONFIGURATION:

38.2.1 SPI CONFIGURATION PARAMETERS:

Similar to the polling mode, set parameters like SPI mode, clock speed, bit order, and data frame size.

38.2.2 ENABLE SPI PERIPHERAL:

Configure the SPI hardware registers and enable the SPI peripheral.

38.2.3 ENABLE INTERRUPTS:

Enable the relevant SPI interrupts in the microcontroller's interrupt controller.

38.3 DATA TRANSFER FUNCTION:

38.3.1TRANSMIT DATA:

Start data transmission and enable the transmit buffer empty interrupt.

38.3.2 RECEIVE DATA:

Enable the receive buffer not empty interrupt to handle incoming data.

38.4 INTERRUPT SERVICE ROUTINES(ISRs):

38.4.1TRANSMIT ISR:

Handle the transmission of data, typically by loading the next byte of data into the transmit register.

38.4.2 RECEIVE ISR:

Handle incoming data by reading from the receive register.

38.4.3 ERROR HANDLING:

Optionally handle errors such as overrun or framing errors within the ISRs.

38.5 CODE OF CODING AN SPI DRIVERS FOR INTERRUPT TRANSFER MODE:

```
#include <stdint.h>

#include <stdbool.h>

// Define SPI registers (placeholders)

#define SPI_BASE_ADDR    0x40013000
```

```
#define SPI_CR1          (*(volatile uint32_t *)(SPI_BASE_ADDR +
0x00))
#define SPI_SR           (*(volatile uint32_t *)(SPI_BASE_ADDR +
0x08))
#define SPI_DR           (*(volatile uint32_t *)(SPI_BASE_ADDR +
0x0C))
#define SPI_CR2          (*(volatile uint32_t *)(SPI_BASE_ADDR +
0x04))
// SPI status register flags
#define SPI_SR_TXE       (1 << 1)  // Transmit buffer empty
#define SPI_SR_RXNE      (1 << 0)  // Receive buffer not empty
#define SPI_SR_BSY       (1 << 7)  // Busy flag
// Interrupt flags (placeholders)
#define SPI_CR2_TXEIE    (1 << 7)  // TX buffer empty interrupt
enable
#define SPI_CR2_RXNEIE   (1 << 6)  // RX buffer not empty interrupt
enable
// Global buffers and indices for data transfer
volatile uint8_t txBuffer[256];
volatile uint8_t rxBuffer[256];
volatile uint8_t txIndex = 0;
volatile uint8_t rxIndex = 0;
volatile bool transferComplete = false;
void spi_init() {
    // Configure SPI (e.g., set baud rate, clock polarity, phase, etc.)
```

```
    SPI_CR1 = /* configuration settings */;
    SPI_CR1 |= (1 << 6);  // Enable SPI
    // Enable interrupts for TXE and RXNE
    SPI_CR2 |= SPI_CR2_TXEIE | SPI_CR2_RXNEIE;
    // Enable global interrupts (platform-specific)
    __enable_irq();  // Pseudo-code, use platform-specific function
}
// Transmit data using interrupt mode
void spi_transmit(uint8_t* data, uint8_t length) {
    // Copy data to the transmission buffer
    for (uint8_t i = 0; i < length; i++) {
        txBuffer[i] = data[i];
    }
    txIndex = 0;
    // Enable TXE interrupt to start transmission
    SPI_CR2 |= SPI_CR2_TXEIE;
}
// ISR for handling SPI interrupts (pseudo-code, platform-specific
function)
void SPI_IRQHandler() {
    // Check if TXE flag is set
    if (SPI_SR & SPI_SR_TXE) {
        if (txIndex < sizeof(txBuffer)) {
```

```
            SPI_DR = txBuffer[txIndex++];  // Transmit next byte
        } else {
            SPI_CR2 &= ~SPI_CR2_TXEIE;  // Disable TXE interrupt
            transferComplete = true;     // Indicate transfer completion
        }
    }
    // Check if RXNE flag is set
    if (SPI_SR & SPI_SR_RXNE) {
        rxBuffer[rxIndex++] = SPI_DR;  // Read received byte
    }
}
int main() {
    spi_init();  // Initialize SPI
    // Example usage
    uint8_t dataToSend[] = {0xAB, 0xCD, 0xEF};
    spi_transmit(dataToSend, sizeof(dataToSend));
    while (!transferComplete);  // Wait for the transmission to complete
    // Further processing...
    return 0;
}
```

38.6 KEY POINTS:

38.6.1 ISR DESIGN:

Ensure the ISR is efficient and quick, as interrupt routines should handle only essential tasks to avoid delaying other processes.

38.6.2 GLOBAL VARIABLES:

Use volatile qualifiers for shared variables accessed in both the main code and ISR.

38.6.3 CONCURRENCY:

Consider concurrency issues and race conditions, particularly with shared resources or flags.

CHAPTER#39 CODING SPI DRIVERS FOR DMA TRANSFER MODE

39.1CODING SPI DRIVERS FOR DMA TRANSFER MODE:

Coding SPI drivers for DMA (Direct Memory Access) transfer mode involves setting up the SPI peripheral and a DMA controller to transfer data directly between memory and the SPI data register. This method is efficient for high-speed data transfers and reduces CPU load.

39.2 KEY COMPONENTS:

39.2.1 SPI PERIPHERAL:

Configured to handle the data transfer with the appropriate settings (mode, speed, etc.).

39.2.2 DMA CONTROLLER:

Configured to move data between memory and the SPI peripheral without CPU intervention.

39.2.3 INTERRUPTS:

Used to signal the completion of DMA transfers.

39.3 GENERAL STEPS:

39.3.1 SETUP AND CONFIGURATION:

39.3.1.1 SPI CONFIGURATION:

Set parameters such as SPI mode, clock speed, bit order, and data frame size.

39.3.1.2 DMA CONFIGURATION:

Set up the source and destination addresses, data size, and transfer direction.

39.3.2 DATA TRANSFER FUNCTIONS:

39.3.2.1 START TRANSMISSION:

Configure the DMA channel for transmission, link it with the SPI peripheral, and start the transfer.

39.3.2.2 START RECEPTION:

Similar to transmission, but the DMA is configured to move data from the SPI data register to memory.

39.3.3 INTERRUPT HANDLING:

39.3.3.1 DMA TRANSFER COMPLETE:

Interrupts are used to handle the completion of the data transfer, allowing the system to proceed with post-transfer tasks.

39.4 CODE OF CODING SPI DRIVERS FOR DMA TRANSFER MODE:

```
#include <stdint.h>

#include <stdbool.h>

// Define SPI registers (placeholders)

#define SPI_BASE_ADDR    0x40013000

#define SPI_CR1          (*(volatile uint32_t *)(SPI_BASE_ADDR +
0x00))

#define SPI_CR2          (*(volatile uint32_t *)(SPI_BASE_ADDR +
0x04))

#define SPI_SR           (*(volatile uint32_t *)(SPI_BASE_ADDR +
0x08))

#define SPI_DR           (*(volatile uint32_t *)(SPI_BASE_ADDR +
0x0C))

// Define DMA registers (placeholders)

#define DMA_BASE_ADDR    0x40020000

#define DMA_CCR1         (*(volatile uint32_t *)(DMA_BASE_ADDR
+ 0x08))

#define DMA_CNDTR1       (*(volatile uint32_t
*)(DMA_BASE_ADDR + 0x0C))
```

```
#define DMA_CPAR1        (*(volatile uint32_t *)(DMA_BASE_ADDR
+ 0x10))
#define DMA_CMAR1        (*(volatile uint32_t
*)(DMA_BASE_ADDR + 0x14))
// DMA configuration flags (placeholders)
#define DMA_CCR1_EN      (1 << 0)
#define DMA_CCR1_TCIE    (1 << 1)  // Transfer complete interrupt
enable
// Global buffer pointers and completion flags
volatile uint8_t* txBufferPtr;
volatile uint8_t* rxBufferPtr;
volatile uint8_t txLength;
volatile uint8_t rxLength;
volatile bool dmaTransferComplete = false;
void spi_init() {
    // Configure SPI (e.g., set baud rate, clock polarity, phase, etc.)
    SPI_CR1 = /* configuration settings */;
    SPI_CR1 |= (1 << 6);  // Enable SPI
}
void dma_init() {
    // Configure DMA for SPI TX and RX
    // DMA_CCR1: Set transfer type, increment, size, and enable
interrupts
    DMA_CCR1 = /* configuration settings */;
```

```
    // Other DMA configurations as needed
}
void spi_dma_transmit(uint8_t* data, uint8_t length) {
    txBufferPtr = data;
    txLength = length;
    // Configure DMA for SPI transmit
    DMA_CPAR1 = (uint32_t)&SPI_DR;        // Peripheral address (SPI
data register)
    DMA_CMAR1 = (uint32_t)txBufferPtr;    // Memory address (data
buffer)
    DMA_CNDTR1 = txLength;                // Number of data items to
transfer
    DMA_CCR1 |= DMA_CCR1_EN;              // Enable DMA channel
    // Enable SPI DMA request
    SPI_CR2 |= (1 << 1);                  // Set appropriate bits to enable DMA
mode
}
void spi_dma_receive(uint8_t* buffer, uint8_t length) {
    rxBufferPtr = buffer;
    rxLength = length;
    // Configure DMA for SPI receive
    DMA_CPAR1 = (uint32_t)&SPI_DR;        // Peripheral address (SPI
data register)
    DMA_CMAR1 = (uint32_t)rxBufferPtr;    // Memory address (receive
buffer)
```

```
    DMA_CNDTR1 = rxLength;              // Number of data items to
transfer
    DMA_CCR1 |= DMA_CCR1_EN;            // Enable DMA channel
    // Enable SPI DMA request
    SPI_CR2 |= (1 << 1);                // Set appropriate bits to enable DMA
mode
}
// DMA transfer complete ISR (pseudo-code)
void DMA_IRQHandler() {
    // Check transfer complete flag
    if (/* DMA transfer complete flag */) {
        dmaTransferComplete = true;
        // Disable DMA channel and clear flags
        DMA_CCR1 &= ~DMA_CCR1_EN;
        // Handle additional cleanup or processing if needed
    }
}
int main() {
    spi_init();   // Initialize SPI
    dma_init();   // Initialize DMA
    // Example usage
    uint8_t dataToSend[] = {0xAB, 0xCD, 0xEF};
    uint8_t dataToReceive[3];
    spi_dma_transmit(dataToSend, sizeof(dataToSend));
```

```
    spi_dma_receive(dataToReceive, sizeof(dataToReceive));
    while (!dmaTransferComplete);  // Wait for the DMA transfer to complete
    // Further processing...
    return 0;
}
```

39.5 KEY CONSIDERATIONS:

39.5.1 DMA CONFIGURATION:

Correctly configure source and destination addresses, data size, and transfer direction.

39.5.2 INTERRUPT HANDLING:

Ensure DMA interrupts are properly configured and handled.

39.5.3 BUFFER MANAGEMENT:

Ensure that the buffers used for DMA transfers are appropriately sized and aligned according to the microcontroller's requirements.

39.5.4CONCURRENCY AND SYNCHRONIZATION:

Manage access to shared resources and handle potential race conditions, especially if DMA is being used for both transmit and receive operations simultaneously.

CHAPTER#40 CODING UART DRIVERS FOR POLLING TRANSFER MODE

40.1CODING UART DRIVERS FOR POLLING TRANSFER MODE:

Coding UART drivers for polling transfer mode in STM32 microcontrollers involves a few key steps, typically using the HAL (Hardware Abstraction Layer) provided by STMicroelectronics.

40.2SETUP THE DEVLEOPMENT ENVIRONMENT:

Ensure you have the STM32CubeIDE or another compatible IDE installed, along with the STM32CubeMX tool for configuring peripherals.

40.3 UART CONFIGURATION UAING STM32CUBEMX:

40.3.1 SELECT THE PERIPHERAL:

Open STM32CubeMX and create a new project. Select the appropriate STM32 microcontroller. Configure the UART peripheral by enabling it and setting the desired parameters (baud rate, word length, stop bits, parity, etc.).

40.3.2 PIN CONFIGURATION:

Assign the UART pins (TX and RX) to the microcontroller's physical pins.

40.3.3 GENERATE THE INITIALIZATION CODE:

After setting up the UART, generate the initialization code. STM32CubeMX will create a project with the HAL drivers pre-configured for UART.

40.4 CODE OF CODING UART DRIVERS FOR POLLING TRANSFER MODE:

```
#include "stm32f1xx_hal.h"  // Adjust according to your MCU series
UART_HandleTypeDef huart1;  // Example UART handle
// UART initialization function (usually generated by STM32CubeMX)
void MX_USART1_UART_Init(void) {
    huart1.Instance = USART1;
    huart1.Init.BaudRate = 115200;
    huart1.Init.WordLength = UART_WORDLENGTH_8B;
    huart1.Init.StopBits = UART_STOPBITS_1;
    huart1.Init.Parity = UART_PARITY_NONE;
    huart1.Init.Mode = UART_MODE_TX_RX;
    huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
    huart1.Init.OverSampling = UART_OVERSAMPLING_16;
    if (HAL_UART_Init(&huart1) != HAL_OK) {
        // Initialization error handling
    }
}
// Transmit data in polling mode
HAL_StatusTypeDef UART_Transmit(UART_HandleTypeDef *huart,
uint8_t *pData, uint16_t Size, uint32_t Timeout) {
    return HAL_UART_Transmit(huart, pData, Size, Timeout);
}
```

```
// Receive data in polling mode
HAL_StatusTypeDef UART_Receive(UART_HandleTypeDef *huart,
uint8_t *pData, uint16_t Size, uint32_t Timeout) {
    return HAL_UART_Receive(huart, pData, Size, Timeout);
}
int main(void) {
    HAL_Init();  // Initialize the HAL Library
    MX_USART1_UART_Init();  // Initialize the configured UART
    uint8_t dataToSend[] = "Hello, UART!";
    uint8_t dataReceived[100];
    // Transmit data
    if (UART_Transmit(&huart1, dataToSend, sizeof(dataToSend),
HAL_MAX_DELAY) != HAL_OK) {
        // Transmission error handling
    }
    // Receive data
    if (UART_Receive(&huart1, dataReceived, 100,
HAL_MAX_DELAY) != HAL_OK) {
        // Reception error handling
    }
    while (1) {
        // Main loop
    }
}
```

40.5 KEY POINTS:

40.5.1POLLING MODE:

The HAL_UART_Transmit and HAL_UART_Receive functions block the CPU until the operation is complete. This is simpler but not suitable for high-throughput or time-sensitive applications.

40.5.2 ERROR HANDLING:

Always check the return values of HAL functions and implement error handling as needed.

40.5.3 TIMEOUTS:

Use appropriate timeout values based on your application's requirements.

CHAPTER#41 CODING UART DRIVERS FOR INTERRUPT TRANSFER MODE

41.1CODING UART DRIVERS FOR INTERRUPT TRANSFER MODE:

Coding UART drivers for interrupt transfer mode in STM32 microcontrollers allows the CPU to perform other tasks while waiting for UART events, which is more efficient than polling.

41.2 SETUP THE DEVELOPMENT ENVIRONMENT:

Make sure you have STM32CubeIDE or another compatible IDE installed, along with STM32CubeMX for configuring peripherals.

41.3 UART CONFIGURATION USING STM32CUBEMX:

41.3.1SELECT UART PERIPHERAL:

Open STM32CubeMX and create a new project. Select the appropriate STM32 microcontroller. Enable the UART peripheral and configure the desired parameters (baud rate, word length, stop bits, parity, etc.).

41.3.2 PIN CONFIGURATION:

Assign the UART pins (TX and RX) to the appropriate physical pins on the microcontroller.

41.3.3 ENABLE INTERRUPTS:

In the NVIC (Nested Vectored Interrupt Controller) settings, enable the global interrupt for the UART peripheral you're using (e.g., USART1 global interrupt).

41.3.4 GENERATE THE INITIALIZATION CODE:

STM32CubeMX will generate the necessary initialization code, including UART setup and interrupt configuration.

41.4 WRITING THE UART INTERRUPT DRIVER CODE:

The initialization function generated by STM32CubeMX sets up the UART peripheral. You will need to implement functions to handle data transmission and reception using interrupts.

41.5 CODE OF CODING UART DRIVERS FOR INTERRUPT TRANSFER MODE:

```
#include "stm32f1xx_hal.h"  // Adjust according to your MCU series
UART_HandleTypeDef huart1;  // Example UART handle
uint8_t txBuffer[] = "Hello, UART Interrupt!";
uint8_t rxBuffer[100];
uint8_t rxComplete = 0;
void MX_USART1_UART_Init(void) {
    huart1.Instance = USART1;
    huart1.Init.BaudRate = 115200;
    huart1.Init.WordLength = UART_WORDLENGTH_8B;
    huart1.Init.StopBits = UART_STOPBITS_1;
    huart1.Init.Parity = UART_PARITY_NONE;
    huart1.Init.Mode = UART_MODE_TX_RX;
    huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
    huart1.Init.OverSampling = UART_OVERSAMPLING_16;
    if (HAL_UART_Init(&huart1) != HAL_OK) {
        // Initialization error handling
    }
}
void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart) {
    if (huart->Instance == USART1) {
        // Transmission complete callback
```

```
        // Optional: Set a flag or call another function
    }
}
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart) {
    if (huart->Instance == USART1) {
        rxComplete = 1;  // Reception complete, set a flag
    }
}
void HAL_UART_ErrorCallback(UART_HandleTypeDef *huart) {
    if (huart->Instance == USART1) {
        // Handle error
    }
}
int main(void) {
    HAL_Init();  // Initialize the HAL Library
    MX_USART1_UART_Init();  // Initialize the configured UART
    // Start receiving data in interrupt mode
    if (HAL_UART_Receive_IT(&huart1, rxBuffer, sizeof(rxBuffer)) !=
HAL_OK) {
        // Reception error handling
    }
    // Start transmitting data in interrupt mode
    if (HAL_UART_Transmit_IT(&huart1, txBuffer, sizeof(txBuffer)) !=
HAL_OK) {
```

```
        // Transmission error handling
    }
    while (1) {
        if (rxComplete) {
            // Process received data
            rxComplete = 0;
            // Restart receiving for new data
            if (HAL_UART_Receive_IT(&huart1, rxBuffer,
sizeof(rxBuffer)) != HAL_OK) {
                // Reception error handling
            }
        }
        // Other main loop code
    }
}
```

41.6 EXPLANATION:

41.6.1 INITILIZATION:

MX_USART1_UART_Init() configures the UART peripheral.

41.6.2 INTERRUPT CALLBACKS:

41.6.2.1 HAL_UART_TxCPLTCALLBACK:

Called when data transmission is complete.

41.6.2.2 HAL_UART_RXCPLTCALLBACK:

Called when data reception is complete. A flag (rxComplete) is set to indicate that data has been received.

41.6.2.3 HAL_UART_ERRORCALLBACK:

Handles errors that occur during UART operations.

41.6.3 MAIN LOOP:

The main loop checks the rxComplete flag to determine if data has been received and processed.

41.7 KEY CONSIDERATIONS:

41.7.1 INTERRUPT PRIORITIES:

Configure interrupt priorities appropriately to ensure timely handling of critical events.

41.7.2 BUFFER MANAGEMENT:

Implement proper buffer management, especially for reception, to handle data overflow and avoid data loss.

41.7.3 ERROR HANDLING:

Robust error handling is crucial for reliable communication, especially in environments prone to noise or other interference.

CHAPTER#42 CODING UART FOR DRIVERS FOR DMA TRANSFER MODE

42.1CODING UART FOR DRIVERS FOR DMA TRANSFER MODE:

Coding UART drivers for DMA (Direct Memory Access) transfer mode in STM32 microcontrollers allows efficient data transmission and reception without CPU intervention. This is particularly useful for handling large data buffers or continuous data streams.

42.2 SETUP THE DEVELOPMENT ENVIRONMENT:

Ensure you have STM32CubeIDE or another compatible IDE, along with STM32CubeMX for configuring peripherals.

42.3 UART AND DMA CONFIGURATION USING STM32CUBEMX:

42.3.1 SELECT UART PERIPHERAL:

Open STM32CubeMX and create a new project. Select your specific STM32 microcontroller. Enable the UART peripheral and set the desired parameters (baud rate, word length, stop bits, parity, etc.)

42.3.2 PIN CONFIGURATION:

Assign the UART pins (TX and RX) to the appropriate physical pins on the microcontroller.

42.3.3 ENABLE DMA FOR UART:

Navigate to the DMA settings in STM32CubeMX. Add a DMA request for both UART transmission (TX) and reception (RX). Configure the DMA settings such as data width, memory increment mode, and priority.

42.3.4 ENABLE NVIC FOR UART AND DMA:

Enable interrupts for UART and DMA (if required) in the NVIC (Nested Vectored Interrupt Controller) settings.

42.3.5 GENERATE THE INITILZATION CODE:

Generate the project files. STM32CubeMX will create the necessary initialization code, including UART and DMA configuration.

42.4 WRITING THE UART DMA DRIVER CODE:

The code generated by STM32CubeMX initializes the UART and DMA peripherals. You will implement functions to start DMA-based data transmission and reception.

42.5 CODE OF CODING UART FOR DRIVERS FOR DMA TRANSFER MODE:

```
#include "stm32f1xx_hal.h"  // Adjust according to your MCU series

UART_HandleTypeDef huart1;  // UART handle

DMA_HandleTypeDef hdma_usart1_rx;  // DMA handle for RX

DMA_HandleTypeDef hdma_usart1_tx;  // DMA handle for TX

uint8_t txBuffer[] = "Hello, UART DMA!";

uint8_t rxBuffer[100];

uint8_t rxComplete = 0;
```

```
void MX_USART1_UART_Init(void) {
    huart1.Instance = USART1;
    huart1.Init.BaudRate = 115200;
    huart1.Init.WordLength = UART_WORDLENGTH_8B;
    huart1.Init.StopBits = UART_STOPBITS_1;
    huart1.Init.Parity = UART_PARITY_NONE;
    huart1.Init.Mode = UART_MODE_TX_RX;
    huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
    huart1.Init.OverSampling = UART_OVERSAMPLING_16;
    if (HAL_UART_Init(&huart1) != HAL_OK) {
        // Initialization error handling
    }
}
void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart) {
    if (huart->Instance == USART1) {
        // Transmission complete callback
    }
}
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart) {
    if (huart->Instance == USART1) {
        rxComplete = 1;  // Reception complete
    }
}
```

```
void HAL_UART_ErrorCallback(UART_HandleTypeDef *huart) {
    if (huart->Instance == USART1) {
        // Error handling
    }
}
int main(void) {
    HAL_Init();  // Initialize the HAL Library
    MX_USART1_UART_Init();  // Initialize UART
    HAL_UART_Receive_DMA(&huart1, rxBuffer, sizeof(rxBuffer));  //
Start receiving data using DMA
    while (1) {
        if (rxComplete) {
            // Process received data
            rxComplete = 0;
            // Restart receiving for new data
            HAL_UART_Receive_DMA(&huart1, rxBuffer,
sizeof(rxBuffer));
        }
        // Other main loop code
    }
}
```

42.6 EXPALNATION:

42.6.1INITILIZATION:

MX_USART1_UART_Init(): Configures the UART peripheral with DMA settings.

42.6.2 DMA FUNCTIONS:

HAL_UART_Transmit_DMA(): Starts the transmission of data using DMA. HAL_UART_Receive_DMA(): Starts the reception of data using DMA.

42.6.3 CALLBACKS:

42.6.3.1 HAL_UART_TxCPLTCALLBACK:

Called when DMA transmission is complete.

42.6.3.2 HAL_UART_RxCPLTCALLBACK:

Called when DMA reception is complete. A flag (rxComplete) is set to indicate that data has been received.

42.6.3.3 HAL_UART_ERRORCALLBACK:

Handles UART errors, such as overrun or framing errors.

42.7 KEY CONSIDERATIONS:

42.7.1 BUFFER MANAGEMENT:

Carefully manage the data buffers, especially for reception, to handle cases where data might overflow the buffer size.

42.7.2 INTERRUPTS AND PRIORITIES:

Ensure that UART and DMA interrupts have appropriate priorities for the application's timing requirements.

42.7.3 ERROR HANDLING:

Implement robust error handling, especially for high-throughput or noisy environments, to maintain reliable communication.

CHAPTER#43 CODING MULTIPLE UART MODULES

43.1CODING MULTIPLE UART MODULES:

Coding multiple UART modules using STM32 involves configuring and handling multiple UART peripherals simultaneously. This is useful in applications requiring multiple serial communication interfaces.

43.2 SETUP THE DEVELOPMENT ENVIRONMENT:

Ensure you have STM32CubeIDE or another compatible IDE installed, along with STM32CubeMX for configuring peripherals.

43.3 UART CONFIGURATION USING STM32CUBEMX:

43.3.1 SELECT UART PERIPHERALS:

Open STM32CubeMX and create a new project. Select your specific STM32 microcontroller. Enable the UART peripherals you intend to use (e.g., USART1, USART2).

43.3.2 PIN CONFIGURATION:

Assign the UART pins (TX and RX) to the appropriate physical pins on the microcontroller for each UART peripheral.

43.3.3 ENABLE INTERRUPTS:

In the NVIC (Nested Vectored Interrupt Controller) settings, enable the global interrupt for each UART peripheral you are using (e.g., USART1 global interrupt, USART2 global interrupt).

43.3.4 GENERATE THE INITILZATION CODE:

Generate the project files. STM32CubeMX will create the necessary initialization code, including UART configuration.

43.4 WRITING THE UART DRIVER CODE:

The code generated by STM32CubeMX initializes the UART peripherals. You will implement functions to handle data transmission and reception for each UART module.

43.5 CODE OF CODING MULTIPLE UART MODULES:

```
#include "stm32f1xx_hal.h"  // Adjust according to your MCU series

UART_HandleTypeDef huart1;  // UART handle for USART1

UART_HandleTypeDef huart2;  // UART handle for USART2

uint8_t txBuffer1[] = "Hello, UART1!";

uint8_t txBuffer2[] = "Hello, UART2!";

uint8_t rxBuffer1[100];

uint8_t rxBuffer2[100];
```

```
uint8_t rxComplete1 = 0;
uint8_t rxComplete2 = 0;
void MX_USART1_UART_Init(void) {
    huart1.Instance = USART1;
    huart1.Init.BaudRate = 115200;
    huart1.Init.WordLength = UART_WORDLENGTH_8B;
    huart1.Init.StopBits = UART_STOPBITS_1;
    huart1.Init.Parity = UART_PARITY_NONE;
    huart1.Init.Mode = UART_MODE_TX_RX;
    huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
    huart1.Init.OverSampling = UART_OVERSAMPLING_16;
    if (HAL_UART_Init(&huart1) != HAL_OK) {
        // Initialization error handling
    }
}
void MX_USART2_UART_Init(void) {
    huart2.Instance = USART2;
    huart2.Init.BaudRate = 115200;
    huart2.Init.WordLength = UART_WORDLENGTH_8B;
    huart2.Init.StopBits = UART_STOPBITS_1;
    huart2.Init.Parity = UART_PARITY_NONE;
    huart2.Init.Mode = UART_MODE_TX_RX;
    huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
```

```
    huart2.Init.OverSampling = UART_OVERSAMPLING_16;
    if (HAL_UART_Init(&huart2) != HAL_OK) {
        // Initialization error handling
    }
}
void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart) {
    if (huart->Instance == USART1) {
        // Transmission complete callback for USART1
    } else if (huart->Instance == USART2) {
        // Transmission complete callback for USART2
    }
}
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart) {
    if (huart->Instance == USART1) {
        rxComplete1 = 1;  // Reception complete for USART1
    } else if (huart->Instance == USART2) {
        rxComplete2 = 1;  // Reception complete for USART2
    }
}
void HAL_UART_ErrorCallback(UART_HandleTypeDef *huart) {
    if (huart->Instance == USART1) {
        // Error handling for USART1
    } else if (huart->Instance == USART2) {
```

```
        // Error handling for USART2
    }
}
int main(void) {
    HAL_Init();  // Initialize the HAL Library
    MX_USART1_UART_Init();  // Initialize USART1
    MX_USART2_UART_Init();  // Initialize USART2
    // Start receiving data using interrupts for both UARTs
    HAL_UART_Receive_IT(&huart1, rxBuffer1, sizeof(rxBuffer1));
    HAL_UART_Receive_IT(&huart2, rxBuffer2, sizeof(rxBuffer2));
    // Start transmitting data using interrupts for both UARTs
    HAL_UART_Transmit_IT(&huart1, txBuffer1, sizeof(txBuffer1));
    HAL_UART_Transmit_IT(&huart2, txBuffer2, sizeof(txBuffer2));
    while (1) {
        if (rxComplete1) {
            // Process received data from USART1
            rxComplete1 = 0;
            // Restart receiving for new data on USART1
            HAL_UART_Receive_IT(&huart1, rxBuffer1,
sizeof(rxBuffer1));
        }
        if (rxComplete2) {
            // Process received data from USART2
```

```
        rxComplete2 = 0;
        // Restart receiving for new data on USART2
        HAL_UART_Receive_IT(&huart2, rxBuffer2,
sizeof(rxBuffer2));
    }
    // Other main loop code
  }
}
```

43.6 EXPLANATION:

43.6.1INITILIZATION:

43.6.1.1MX_USART1_UART_Init():

Configures the USART1 peripheral.

43.6.1.2MX_USART2_UART_Init():

Configures the USART2 peripheral.

43.6.2 INTERRUPT FUNCTIONS:

43.6.2.1 HAL_UART_TRANSMIT():

Starts the transmission of data using interrupts.

43.6.2.2 HAL_UART_RECEIVE_IT():

Starts the reception of data using interrupts.

43.6.3 CALLBACKS:

43.6.3.1 HAL_UART_TxCPLTCALLBACK:

Called when data transmission is complete for both USART1 and USART2.

43.6.3.2 HAL_UART_RxCPLTCALLBACK:

Called when data reception is complete for both USART1 and USART2. Flags (rxComplete1, rxComplete2) are set to indicate that data has been received.

43.6.3.3 HAL_UART_ERRORCALLBACK:

Handles errors that occur during UART operations for both USART1 and USART2.

43.7 MAIN LOOP:

The main loop checks the flags to determine if data has been received and processes it accordingly. It then restarts the reception for new data.

43.8 KEY CONSIDERATIONS:

43.8.1BUFFER MANAGEMENT:

Ensure proper buffer management to handle cases where data might overflow the buffer size.

43.8.2 INTERRUPTS AND PRIORITIES:

Configure interrupt priorities appropriately to ensure timely handling of UART events.

43.8.3 ERROR HANDLING:

Implement robust error handling to maintain reliable communication, especially in environments with potential noise or interference.

CHAPTER#44 UNDERSTANDING THE USB CAPABILITIES OF DIFFERENT DEV BOARDS

44.1UNDERSTANDING THE USB CAPABILITIES OF DIFFERENT DEV BOARDS:

Understanding the USB capabilities of different STM32 development boards involves examining their specific features, peripheral support, and use cases. STM32 microcontrollers from STMicroelectronics are highly versatile and widely used in various embedded applications, including those requiring USB connectivity.

44.2 KEY ASPECTS TO CONSIDER:

44.2.1USB PERIPHERAL SUPPORT:

44.2.1.1 USB DEVICES:

Many STM32 boards support USB device functionality, allowing the microcontroller to act as a peripheral connected to a host, such as a computer.

44.2.1.2 USB HOST:

Some STM32 boards can function as a USB host, enabling them to connect to and control other USB devices like keyboards, mice, or storage devices.

44.2.1.3 USB OTG(ON-THE-GO):

This feature allows the microcontroller to switch between host and device roles dynamically.

44.2.2 STM32 FAMILIES AND USB CAPABILITIES:

44.2.2.1STM32F0:

Basic USB device support. Suitable for simple USB applications.

Figure 44.2.24:STM32F0

44.2.2.2STM32F1:

Includes USB device support and some models with USB OTG.

Figure 44.2.25:STM32F1

44.2.2.3 STM32F2:

USB OTG FS (Full-Speed) support, higher performance than F1.

Figure 44.2.26:STM32F2

44.2.2.4 STM32F3:

Similar to F1 but with enhanced analog capabilities.

Figure 44.2.27:STM32F3

44.2.2.5 STM32F4:

USB OTG FS and HS (High-Speed) support, suitable for more demanding applications.

Figure 44.2.28:STM32F4

44.2.2.6 STM32F7:

High-performance with USB OTG FS and HS, ideal for complex applications.

Fi0gure 44.2.29:STM32F7

44.2.2.7STM32H7:

Highest performance, supporting USB OTG FS and HS, suitable for the most demanding USB applications.

Figure 44.2.30:STM32H7

44.2.2.8 STM32L0/L1/L4:

Ultra-low-power series with USB device support, designed for battery-powered applications.

Figure 44.2.31:STM32L0/L1/L4

44.2.2.9 STM32G0/G4:

General-purpose with USB device sup

Figure 44.2.32:STM32G0/G4

44.2.3 USB SPEED:

44.2.3.1 FAST-SPEED(FS):

12 Mbps, supported by many STM32 models.

Figure 44.2.33:USB FAST SPEED

44.2.3.2 HIGH-SPEED(HS):

480 Mbps, supported by higher-end models like STM32F4, F7, and H7.

Figure 44.2.34:HIGH-SPEED USB

44.2.4 DEVELOPMENT TOOLS:

44.2.4.1STM32CUBEMX:

A graphical tool for configuring STM32 peripherals, including USB. It generates initialization code for USB peripheral setup.

Figure 44.2.35:STM32CUBEMX

44.2.4.2 STM32CUBEIDE:

An integrated development environment that supports USB application development with STM32 microcontrollers.

Figure 44.2.36:STM32CUBEIDE

44.2.4.3 USB LIBRARIES:

ST provides USB libraries within the STM32Cube firmware packages, facilitating the development of USB applications.

Figure 44.2.37:USB LIBRARIES

44.3 EXAMPLES BOARS AND THEIR CAPABILITES:

44.3.1 STM32F103C8T6(“BLUE PILL”):

1: USB DEVICE IS SUPPORTED

2: USB HOST/OTG IS NOT SUPPORTED

3: USE SPEED IS FULL-SPEED

Figure 44.3.38:STM32F103C8T6

44.3.2 STM32F407 DISCOVERY:

1: USB DEVICE IS SUPPORTED

2: USB HOST/OTG IS SUPPORTED

3: USE SPEED IS FULL-SPEED AND HIGH-SPEED (WITH EXTERNAL PHY).

Figure 44.3.39:STM32F407 DISCOVERY

44.3.3 STM32F746G-DISCO:

1: USB DEVICE IS SUPPORTED

2: USB HOST/OTG IS SUPPORTED

3: USE SPEED IS FULL-SPEED AND HIGH-SPEED (WITH INTERNAL OR EXTERNAL PHY)

Figure 44.3.40:STM32F746G-DISCO

44.3.4 STM32H7 NUCLEO:

1: USB DEVICE IS SUPPORTED

2: USB HOST/OTG IS SUPPORTED

3: USE SPEED IS FULL-SPEED AND HIGH-SPEED (WITH INTERNAL OR EXTERNAL PHY)

Figure 44.3.41:STM32H7 NUCLEO

44.4 EXAMPLE APPLICATIONS:

44.4.1USB DEVICE:

Creating HID (Human Interface Devices) such as keyboards and mice, CDC (Communication Device Class) for virtual COM ports, or MSC (Mass Storage Class) for USB drives.

Figure 44.4.42:USB DEVICE

44.4.2 USB HOST:

Connecting and controlling USB peripherals like flash drives, cameras, or printers.

Figure 44.4.43:USB HOST:

44.4.3 USB OTG:

Mobile devices that can switch between connecting to a computer (device mode) and connecting to peripherals (host mode).

Figure 44.4.44:USB OTG

44.5 GETTING STARTED:

44.5.1HARDWARE SETUP:

Select an STM32 development board that matches your USB application requirements.

44.5.2 SOFTWARE TOOLS:

Install STM32CubeMX and STM32CubeIDE.

44.5.3 CONFIGURATION:

Use STM32CubeMX to configure the USB peripheral (Device/Host/OTG) and generate initialization code. Import the generated code into STM32CubeIDE.

44.5.4 DEVELOPMENT:

Utilize USB libraries and example projects provided by ST to implement your specific USB functionality. Debug and test your application on the chosen development board.

CHAPTER#45 CODING USB DRIVERS FOR HUMAN INTERFACE DEVICE(HID)-MOUSE FUNCTIONALITY

45.1CODING USB DRIVERS FOR HUMAN INTERFACE DEVICE(HID)-MOUSE FUNCTIONALITY:

Creating USB drivers for a Human Interface Device (HID) like a mouse involves several steps. This process includes understanding USB protocols, HID specifications, and writing code to interface with the hardware.

45.2 STEP 1: UNDERSTAND USB AND HID PROTOCOLS:

45.2.1 USB PROTOCOL BASICS:

Learn about USB endpoints, descriptors, transfer types (control, interrupt, bulk, isochronous), and the USB communication process.

45.2.2 HID PROTOCOL:

Understand the HID class specification, including the HID descriptor, report descriptor, and how HID devices communicate with the host.

45.3 STEP 2: SET UP YOUR DEVELOPMENT ENVIRONMENT:

45.3.1DEVELOPMENT BOARD:

Use a development board with USB host capabilities (e.g., STM32, ESP32, Raspberry Pi).

45.3.2 DEVELOPMENT TOOLS:

Install necessary tools such as compilers, IDEs (e.g., Keil, Arduino IDE, Eclipse), and SDKs for your microcontroller or processor.

45.4 STEP 3: WRITE USB HOST CODE:

45.4.1INITIALIZE USB HOST:

Set up the USB host stack and initialize the USB hardware.

45.4.2 ENUMERATE DEVICES:

Write code to enumerate connected USB devices, read descriptors, and recognize HID devices.

45.5 STEP 4: IMPLMENT HID MOUSE DRIVER:

45.5.1PRAISE HID DESCRIPTOR:

Extract the HID report descriptor and understand the format of the mouse reports.

45.5.2 READ HID REPORTS:

Set up an interrupt endpoint to read reports from the mouse.

45.5.3 PROCESS INPUT DATA:

Decode the input reports to get mouse movements and button states.

45.6 STEP 5: INTEGRATE WITH YOUR APPLICATION:

45.6.1 EVENT HANDLING:

Implement event handlers to process mouse movements and button clicks.

45.6.2 APPLICATION LOGIC:

Integrate mouse input with your application logic.

45.7 CODE OF CODING USB DRIVERS FOR HUMAN INTERFACE DEVICE(HID)-MOUSE FUNCTIONALITY:

```
#include "usb_host.h"
#include "usbh_hid.h"
#include "usbh_hid_mouse.h"
USBH_HandleTypeDef hUsbHostFS;
HID_MOUSE_Info_TypeDef *mouse_info;
void USBH_UserProcess(USBH_HandleTypeDef *phost, uint8_t id) {
    switch(id) {
        case HOST_USER_SELECT_CONFIGURATION:
```

```
            break;
        case HOST_USER_DISCONNECTION:
            USBH_HID_Stop(phost);
            break;
        case HOST_USER_CLASS_ACTIVE:
            if (USBH_HID_GetDeviceType(phost) == HID_MOUSE) {
                USBH_HID_MouseInit(phost);
            }
            break;
        default:
            break;
    }
}
void process_mouse_data(HID_MOUSE_Info_TypeDef *mouse_info) {
    if (mouse_info != NULL) {
        printf("X: %d, Y: %d, Button1: %d, Button2: %d, Button3: %d\n",
            mouse_info->x,
            mouse_info->y,
            mouse_info->buttons[0],
            mouse_info->buttons[1],
            mouse_info->buttons[2]);
    }
}
```

```
int main(void) {
    HAL_Init();
    SystemClock_Config();
    MX_USB_HOST_Init();
    while (1) {
        USBH_Process(&hUsbHostFS);
        mouse_info = USBH_HID_GetMouseInfo(&hUsbHostFS);
        if (mouse_info != NULL) {
            process_mouse_data(mouse_info);
        }
    }
}
void MX_USB_HOST_Init(void) {
    USBH_Init(&hUsbHostFS, USBH_UserProcess, HOST_FS);
    USBH_RegisterClass(&hUsbHostFS, USBH_HID_CLASS);
    USBH_Start(&hUsbHostFS);
}
```

45.8 EXPLANATION OF THE CODE:

45.8.1 USB HOST INITILIZATION:

MX_USB_HOST_Init initializes the USB host stack and registers the HID class.

45.8.2 USER PROCESS CALLBACKS:

USBH_UserProcess handles USB host state changes, such as device configuration and disconnection.

45.8.3 MOUSE DATA PROCESSING:

process_mouse_data extracts and prints mouse movement and button states.

45.8.4 MAIN LOOP:

Continuously processes USB host tasks and retrieves mouse data.

CHAPTER#46 CODING USB DRIVERS FOR HUMAN INTERFACE DEVICE (HID)-KEYBOARD FUNCTIONALITY

46.1CODING USB DRIVERS FOR HUMAN INTERFACE DEVICE (HID)-KEYBOARD FUNCTIONALITY:

Creating USB drivers for a Human Interface Device (HID) with keyboard functionality involves several steps, similar to developing a mouse driver.

46.2 STEP 1: UNDERSTAND USB AND HID PROTOCOLS:

46.2.1 USB PROTOCOLS BASICS:

Learn about USB endpoints, descriptors, transfer types (control, interrupt, bulk, isochronous), and the USB communication process.

46.2.2 HID PROTOCOL:

Understand the HID class specification, including the HID descriptor, report descriptor, and how HID devices communicate with the host.

46.3 STEP 2: SET UP YOUR DEVELOPMENT ENVIRONMENT:

46.3.1 DEVELOPMENT BOARDS:

Use a development board with USB host capabilities (e.g., STM32, ESP32, Raspberry Pi).

46.3.2 DEVELOPMENT TOOLS:

Install necessary tools such as compilers, IDEs (e.g., Keil, Arduino IDE, Eclipse), and SDKs for your microcontroller or processor.

46.4 STEP 3: WRITE USB HOST CODE:

46.4.1INITIALIZE USB HOST:

Set up the USB host stack and initialize the USB hardware.

46.4.2ENUMERATE DEVICES:

Write code to enumerate connected USB devices, read descriptors, and recognize HID devices.

46.5 STEP4: IMPLEMENT HID KEYBOARD DRIVER:

46.5.1 PARSE HID DESCRIPTOR:

Extract the HID report descriptor and understand the format of the keyboard reports.

46.5.2 READ HID REPORTS:

Set up an interrupt endpoint to read reports from the keyboard.

46.5.3 PROCESS INPUT DATA:

Decode the input reports to get key presses and releases.

46.6 STEP5: INTEGRATE WITH YOUR APPLICATION:

46.6.1EVENT HANDLING:

Implement event handlers to process key presses and releases.

46.6.2 APPLICATION LOGIC:

Integrate keyboard input with your application logic.

46.7CODE OF CODING USB DRIVERS FOR HUMAN INTERFACE DEVICE (HID)-KEYBOARD FUNCTIONALITY:

```
#include "usb_host.h"

#include "usbh_hid.h"

#include "usbh_hid_keyboard.h"
```

```
USBH_HandleTypeDef hUsbHostFS;
HID_KEYBD_Info_TypeDef *keyboard_info;
void USBH_UserProcess(USBH_HandleTypeDef *phost, uint8_t id) {
   switch(id) {
      case HOST_USER_SELECT_CONFIGURATION:
         break;
      case HOST_USER_DISCONNECTION:
         USBH_HID_Stop(phost);
         break;
      case HOST_USER_CLASS_ACTIVE:
         if (USBH_HID_GetDeviceType(phost) == HID_KEYBOARD)
{
            USBH_HID_KeybdInit(phost);
         }
         break;
      default:
         break;
   }
}
void process_keyboard_data(HID_KEYBD_Info_TypeDef
*keyboard_info) {
   if (keyboard_info != NULL) {
      for (int i = 0; i < sizeof(keyboard_info->keys); i++) {
         if (keyboard_info->keys[i] != 0) {
```

```
            printf("Key: %d\n", keyboard_info->keys[i]);
        }
    }
  }
}
int main(void) {
  HAL_Init();
  SystemClock_Config();
  MX_USB_HOST_Init();
  while (1) {
    USBH_Process(&hUsbHostFS);
    keyboard_info = USBH_HID_GetKeybdInfo(&hUsbHostFS);
    if (keyboard_info != NULL) {
      process_keyboard_data(keyboard_info);
    }
  }
}
void MX_USB_HOST_Init(void) {
  USBH_Init(&hUsbHostFS, USBH_UserProcess, HOST_FS);
  USBH_RegisterClass(&hUsbHostFS, USBH_HID_CLASS);
  USBH_Start(&hUsbHostFS);
}
```

46.8 EXPLANATION OF THE CODE:

46.8.1USB HOST INITILIZATION:

MX_USB_HOST_Init initializes the USB host stack and registers the HID class.

46.8.2 USER PROCES CALLBACK:

USBH_UserProcess handles USB host state changes, such as device configuration and disconnection

46.8.3 KEYBOARD DATA PROCESSING:

process_keyboard_data extracts and prints key presses.

46.8.4 MAIN LOOP:

Continuously processes USB host tasks and retrieves keyboard data.

46.9KEY PARTS OF HID KEYBOARD DRIVER:

46.9.1HID DESCRIPTOR PRASING:

Extracting and understanding the HID descriptor is crucial for interpreting the data reports.

46.9.2 INTERRUPT ENDPOINT SETUP:

Keyboard devices typically use interrupt endpoints for sending data to the host.

46.9.3 DATA REPORT DECODING:

Each data report from the keyboard contains information about key presses and releases that need to be decoded according to the HID report descriptor.

CHAPTER#47CODING USB DRIVERS FOR VIRTUAL COM PORT FUNCTIONALITY (TX ONLY)

47.1CODING USB DRIVERS FOR VIRTUAL COM PORT FUNCTIONALITY (TX ONLY):

Creating USB drivers for a virtual COM port (VCP) functionality, specifically for transmitting (TX) data, involves several key steps.

47.2STEP1: UNDERSTAND USB CDC (COMMUNICATION DEVICE CONTROL) PROTCOL:

47.2.1USB PROTOCOL BASICS:

Learn about USB endpoints, descriptors, transfer types (control, interrupt, bulk, isochronous), and the USB communication process.

47.2.2 CDC CLASS PROTOCOL:

Understand the CDC class specification, including the CDC descriptors, and how communication devices like virtual COM ports communicate with the host.

47.3STEP2: SET UP YOUR DEVELOPMENT ENVIRONMENT:

47.3.1DEVELOPMENT BOARD:

Use a development board with USB device capabilities (e.g., STM32, ESP32, Arduino).

47.3.2 DEVELOPMENT TOOLS:

Install necessary tools such as compilers, IDEs (e.g., Keil, Arduino IDE, Eclipse), and SDKs for your microcontroller or processor.

47.4 STEP3: WRITE THE USB DEVICE CODE:

47.4.1 INITIALIZE THE USB:

Set up the USB device stack and initialize the USB hardware.

47.4.2 CONFIGURE ENDPOINTS:

Configure the endpoints for control and bulk transfer for data transmission.

47.5 STEP 4: IMPLEMENT VIRTUAL COM PORT FUNCTIONALITY:

47.5.1CDC DESCRIPTORS:

Define the CDC descriptors required for a virtual COM port.

47.5.2 TRANSMIT DATA:

Implement the functions to send data over the bulk endpoint.

47.6 STEP5: INTEGRATE WITH YOUR APPLICATION:

47.6.1TRANSMIT FUNCTION:

Implement the function to transmit data from your application.

47.6.2 APPLICATION LOGIC:

Integrate the transmit function with your application logic.

47.7 CODE OF CODING USB DRIVERS FOR VIRTUAL COM PORT FUNCTIONALITY (TX ONLY):

```
#include "usb_device.h"

#include "usbd_cdc_if.h"

USBD_HandleTypeDef hUsbDeviceFS;

// Initialize USB Device

void MX_USB_DEVICE_Init(void) {

   USBD_Init(&hUsbDeviceFS, &FS_Desc, DEVICE_FS);

   USBD_RegisterClass(&hUsbDeviceFS, &USBD_CDC);

   USBD_CDC_RegisterInterface(&hUsbDeviceFS,
&USBD_Interface_fops_FS);

   USBD_Start(&hUsbDeviceFS);

}

// Transmit Data Function

uint8_t CDC_Transmit_FS(uint8_t* Buf, uint16_t Len) {
```

```
    uint8_t result = USBD_OK;
    USBD_CDC_HandleTypeDef *hcdc =
(USBD_CDC_HandleTypeDef*)hUsbDeviceFS.pClassData;
    if (hcdc->TxState != 0) {
        return USBD_BUSY;
    }
    USBD_CDC_SetTxBuffer(&hUsbDeviceFS, Buf, Len);
    result = USBD_CDC_TransmitPacket(&hUsbDeviceFS);
    return result;
}
// Main Function
int main(void) {
    HAL_Init();
    SystemClock_Config();
    MX_USB_DEVICE_Init();
    uint8_t message[] = "Hello, USB!";
    while (1) {
        CDC_Transmit_FS(message, sizeof(message) - 1);
        HAL_Delay(1000); // Transmit every second
    }
}
```

47.8 EXPLANATION OF THE CODE:

47.8.1 USB DEVICE INITIALIZATION:

MX_USB_DEVICE_Init initializes the USB device stack and registers the CDC class.

47.8.2 TRANSMIT DATA:

CDC_Transmit_FS sets up the transmission buffer and sends data over the USB bulk endpoint.

47.8.3 MAIN LOOP:

Continuously transmits a message every second.

47.9 KEY PARTS OF USB CDC DRIVER:

47.9.1 CDC DESCRIPTORS:

Properly defining CDC descriptors, including the device descriptor, configuration descriptor, and CDC-specific descriptors.

47.9.2 ENDPOINT CONFIGURATION:

Setting up endpoints for control and bulk transfers.

47.9.3 DATA TRANSMISSION:

Implementing non-blocking data transmission functions to send data over the USB.

CHAPTER#48 CODING USB DRIVERS FOR VIRTUAL COM PORT FUNCTIONALITY (RX AND TX)

48.1CODING USB DRIVERS FOR VIRTUAL COM PORT FUNCTIONALITY (RX AND TX):

Creating USB drivers for a virtual COM port (VCP) with both RX (receiving) and TX (transmitting) functionality using an STM32 microcontroller involves several steps. Here, we'll use the STM32CubeMX tool to generate the initial project setup and the STM32 HAL (Hardware Abstraction Layer) library to implement the functionality.

48.2 STEP1: SET UP YOUR DEVELOPMENT ENVIRONMENT:

48.2.1STM32CUBEMX:

Download and install STM32CubeMX, a graphical tool to configure STM32 microcontrollers and generate initialization C code.

48.2.2 STM32CUBEIDE:

Download and install STM32CubeIDE, an integrated development environment for STM32 microcontrollers.

48.2.3 DEVELOPMENT BOARD:

Use an STM32 development board (e.g., STM32F4 Discovery, Nucleo board).

48.3 STEP2: GENERATE THE PROJECT WITH STM32CUBEMX:

48.3.1 OPEN STM32CUBEMX:

Create a new project and select your STM32 microcontroller.

48.3.2 CONFIGURE USB:

Enable USB as a device and select "Communication Device Class (Virtual COM Port)".

48.3.3 CONFIGURE USART:

If you want to use USART for debugging or other purposes, enable and configure it.

48.3.4 GENERATE CODE:

Click "Generate Code" to generate the project files and open it in STM32CubeIDE.

48.4 STEP3: IMPLMENT USB CDC (COMMUNICATION DEVICE CLASS) FUNCTIONALITY:

The generated code provides a basic USB CDC implementation. We will add functions to handle RX and TX operations.

48.4.1 INCLUDE NECESSARY HEADERS:

Ensure you include the necessary headers in usb_device.c and usbd_cdc_if.c.

48.4.2 TRANSMIT DATA(TX):

Implement the function to send data over USB.

48.4.3 RECEIVE DATA(RX):

Implement the function to handle incoming data.

48.5 STEP4: MODIFY USBD_CDC_IF.C FOR RX AND TX:

Edit the usbd_cdc_if.c file to implement the required functionality.

48.5.1 CODE OF TRANSMIT FUNCTION:

```
#include "usbd_cdc_if.h"

// Transmit Data Function

uint8_t CDC_Transmit_FS(uint8_t* Buf, uint16_t Len) {

   uint8_t result = USBD_OK;

   USBD_CDC_HandleTypeDef *hcdc =
(USBD_CDC_HandleTypeDef*)hUsbDeviceFS.pClassData;

   if (hcdc->TxState != 0) {

      return USBD_BUSY;

   }

   USBD_CDC_SetTxBuffer(&hUsbDeviceFS, Buf, Len);

   result = USBD_CDC_TransmitPacket(&hUsbDeviceFS);

   return result;

}
```

48.5.2 CODE OF RECEIVE DATA CALLBACK:

```
#include "usbd_cdc_if.h"
// Buffer to store received data
uint8_t UserRxBufferFS[APP_RX_DATA_SIZE];
// Receive Data Callback
static int8_t CDC_Receive_FS(uint8_t* Buf, uint32_t *Len) {
    // Process received data
    // For example, you can echo received data back to the host
    CDC_Transmit_FS(Buf, *Len);
    USBD_CDC_ReceivePacket(&hUsbDeviceFS);
    return (USBD_OK);
}
// Initialize the CDC interface
USBD_CDC_ItfTypeDef USBD_Interface_fops_FS = {
    CDC_Init_FS,
    CDC_DeInit_FS,
    CDC_Control_FS,
    CDC_Receive_FS
};
```

48.6STEP5: MAIN APPLICATION LOGIC:

```
#include "main.h"
#include "usb_device.h"
// Transmit and Receive Buffers
```

```
uint8_t TxBuffer[] = "Hello, USB!";
uint8_t RxBuffer[64];
int main(void) {
    // Initialize the HAL Library
    HAL_Init();
    // Configure the system clock
    SystemClock_Config();
    // Initialize USB Device
    MX_USB_DEVICE_Init();
    while (1) {
        // Transmit data every second
        CDC_Transmit_FS(TxBuffer, sizeof(TxBuffer) - 1);
        HAL_Delay(1000);
    }
}
void SystemClock_Config(void) {
    // System Clock Configuration (generated by STM32CubeMX)
}
```

48.7STEP6: COMPILE AND FLASH THE FIRMWARE:

48.7.1 BUILD THE PROJECT:

Compile the project in STM32CubeIDE.

48.7.2 FLASH THE MICROCONTOLLER:

Upload the compiled firmware to the STM32 microcontroller using a programmer/debugger (e.g., ST-LINK).

48.8 KEY PARTS OF USB CDC DRIVER:

48.8.1 CDC DESCRIPTORS:

Properly define CDC descriptors, including the device descriptor, configuration descriptor, and CDC-specific descriptors.

48.8.2 ENDPOINT CONFIGURATION:

Set up endpoints for control, bulk transfers (for RX and TX), and interrupt transfers.

48.8.3 DATA TRANSMISSION AND RECEPTION:

Implement non-blocking data transmission and reception functions.

CHAPTER#49 CODING EXPERIMENTAL WITH TASKS AND PRIORITIES

49.1CODING EXPERIMENTAL WITH TASKS AND PRIORITIES:

To experiment with tasks and priorities in coding, you can use various methods and tools to manage and execute tasks based on their priority levels.

49.2 DEFINE THE TASK STRUCTURE:

Define the structure of your tasks. Each task can have attributes such as:

1: ID

2: Description

3: Priority (e.g., High, Medium, Low)

4: Status (e.g., Pending, In Progress, Completed)

5: Dependencies (other tasks that need to be completed first)

49.3 CHOOSE A TASK MANAGEMENT METHOD:

There are several methods to manage tasks and priorities:

49.3.1TO-DO LISTS:

Simple lists to track tasks.

49.3.2 KANBAN BOARDS:

Visual boards to track task progress (e.g., Trello, Jira).

49.3.3GANTT CHARTS:

Timelines to plan and track project schedules.

49.3.4 AGILE/SCUM:

Iterative approach using sprints and backlogs.

49.4CODE OF CODING EXPERIMENTAL WITH TASKS AND PRIORITIES:

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX_TASKS 10

#define MAX_DEPENDENCIES 5
```

```
typedef struct Task {
    int task_id;
    char description[50];
    char priority[10];
    char status[10];
    struct Task* dependencies[MAX_DEPENDENCIES];
    int dependency_count;
} Task;
typedef struct TaskManager {
    Task* tasks[MAX_TASKS];
    int task_count;
} TaskManager;
void init_task(Task* task, int task_id, const char* description, const
char* priority, const char* status) {
    task->task_id = task_id;
    strcpy(task->description, description);
    strcpy(task->priority, priority);
    strcpy(task->status, status);
    task->dependency_count = 0;
}
void add_dependency(Task* task, Task* dependency) {
    if (task->dependency_count < MAX_DEPENDENCIES) {
        task->dependencies[task->dependency_count++] = dependency;
```

```
    }
}
void add_task(TaskManager* manager, Task* task) {
    if (manager->task_count < MAX_TASKS) {
        manager->tasks[manager->task_count++] = task;
    }
}
Task* get_task(TaskManager* manager, int task_id) {
    for (int i = 0; i < manager->task_count; i++) {
        if (manager->tasks[i]->task_id == task_id) {
            return manager->tasks[i];
        }
    }
    return NULL;
}
void set_task_status(TaskManager* manager, int task_id, const char*
status) {
    Task* task = get_task(manager, task_id);
    if (task) {
        strcpy(task->status, status);
    }
}
int compare_priority(const void* a, const void* b) {
```

```
    Task* taskA = *(Task**)a;
    Task* taskB = *(Task**)b;
    return strcmp(taskA->priority, taskB->priority);
}
void list_tasks_by_priority(TaskManager* manager) {
    qsort(manager->tasks, manager->task_count, sizeof(Task*),
compare_priority);
    for (int i = 0; i < manager->task_count; i++) {
        printf("Task ID: %d, Description: %s, Priority: %s, Status: %s\n",
            manager->tasks[i]->task_id, manager->tasks[i]->description,
            manager->tasks[i]->priority, manager->tasks[i]->status);
    }
}
int main() {
    TaskManager manager = { .task_count = 0 };
    Task task1, task2, task3, task4;
    init_task(&task1, 1, "Setup development environment", "High",
"Pending");
    init_task(&task2, 2, "Write code for feature X", "Medium",
"Pending");
    init_task(&task3, 3, "Test feature X", "Low", "Pending");
    init_task(&task4, 4, "Deploy to production", "High", "Pending");
    add_dependency(&task2, &task1);
    add_dependency(&task3, &task2);
```

```
    add_dependency(&task4, &task3);
    add_task(&manager, &task1);
    add_task(&manager, &task2);
    add_task(&manager, &task3);
    add_task(&manager, &task4);
    list_tasks_by_priority(&manager);
    return 0;
}
```

49.5 MANAGING TASK PRIORITIES:

In practice, we can use various criteria to set and manage priorities:

49.5.1URGENCY:

How soon does the task need to be completed?

49.5.2IMPACT:

What is the impact of completing this task?

49.5.3DEPENDENCIES:

Are there tasks that need to be completed first?

49.5.4 RESOURCES:

What resources are required, and are they available?

49.6 USING TASK MANAGEMENT TOOLS:

Consider using existing task management tools if you prefer not to code your own:

49.6.1TRELLO:

For Kanban-style boards.

49.6.2 JIRA:

For Agile project management.

49.6.3 ASANA:

For team collaboration and task tracking.

49.6.4 MONDAY.COM:

For customizable workflows.

CHAPTER#50 CODING ACCESSING HARDWARE DRIVERS FROM MULTIPLE THERADS

50.1CODING ACCESSING HARDWARE DRIVERS FROM MULTIPLE THERADS:

Accessing hardware drivers from multiple threads in an STM32 microcontroller requires careful management of resources and synchronization to prevent race conditions and ensure data integrity. This example uses FreeRTOS, a real-time operating system widely used in embedded systems, to create multiple threads (tasks) that access hardware drivers.

50.2 PREREQUISTES:

50.2.1STM32CUBEMX:

For generating the initial project setup, including FreeRTOS and peripheral initialization.

50.2.2STM32CUBEIDE:

For coding and debugging the project.

50.3 STEP1: SET UP STM32CUBEMX:

1: Open STM32CubeMX and create a new project.

2: Select your STM32 microcontroller model.

3: Enable the peripherals you plan to use (e.g., UART, GPIO).

4: Enable FreeRTOS under Middleware > FREERTOS.

5: Configure the system clock.

6: Generate the project and open it in STM32CubeIDE.

50.4 STEP2: MODIFY 'FREERTOSCONFIG.H':

Ensure FreeRTOSConfig.h includes necessary configurations for multi-threading, such as enabling preemption, setting the heap size, and other relevant settings.

50.5 STEP3: IMPLEMENT THE MAIN CODE:

In main file, you will create tasks and ensure proper access to hardware drivers.

50.5.1 CODE OF CODING ACCESSING HARDWARE DRIVERS FROM MULTIPLE THERADS:

```
#include "main.h"

#include "cmsis_os.h"

/* UART handle declared in main.h and initialized in
MX_USART2_UART_Init() */
```

```
extern UART_HandleTypeDef huart2;
/* Task Handles */
osThreadId_t task1Handle;
osThreadId_t task2Handle;
/* Task function prototypes */
void StartTask1(void *argument);
void StartTask2(void *argument);
/* UART Mutex */
osMutexId_t uartMutexHandle;
int main(void) {
  /* STM32Cube HAL initialization */
  HAL_Init();
  SystemClock_Config();
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  /* Initialize FreeRTOS */
  osKernelInitialize();
  /* Create the mutex */
  const osMutexAttr_t uartMutexAttr = {
    "uartMutex"
  };
  uartMutexHandle = osMutexNew(&uartMutexAttr);
```

```
    /* Create the thread(s) */
    const osThreadAttr_t task1Attr = {
        .name = "task1",
        .priority = (osPriority_t) osPriorityNormal,
        .stack_size = 128
    };
    task1Handle = osThreadNew(StartTask1, NULL, &task1Attr);
    const osThreadAttr_t task2Attr = {
        .name = "task2",
        .priority = (osPriority_t) osPriorityNormal,
        .stack_size = 128
    };
    task2Handle = osThreadNew(StartTask2, NULL, &task2Attr);
    /* Start scheduler */
    osKernelStart();
    /* Infinite loop */
    while (1) {
    }
}
void StartTask1(void *argument) {
    for (;;) {
        /* Lock the mutex */
        osMutexAcquire(uartMutexHandle, osWaitForever);
```

```
        /* Access the UART */
        char *msg = "Task 1 is running\r\n";
        HAL_UART_Transmit(&huart2, (uint8_t *) msg, strlen(msg),
HAL_MAX_DELAY);
        /* Release the mutex */
        osMutexRelease(uartMutexHandle);
        /* Delay to simulate work */
        osDelay(1000);
    }
}
void StartTask2(void *argument) {
    for (;;) {
        /* Lock the mutex */
        osMutexAcquire(uartMutexHandle, osWaitForever);
        /* Access the UART */
        char *msg = "Task 2 is running\r\n";
        HAL_UART_Transmit(&huart2, (uint8_t *) msg, strlen(msg),
HAL_MAX_DELAY);
        /* Release the mutex */
        osMutexRelease(uartMutexHandle);
        /* Delay to simulate work */
        osDelay(1000);
    }
}
```

50.5.2 EXPLANATION:

50.5.2.1MUTEX CREATION:

A mutex is created to manage access to the UART resource.

50.5.2.2TASK CREATION:

Two tasks are created (StartTask1 and StartTask2), each attempting to access the UART.

50.5.2.3TASK FUNCTION:

Each task locks the mutex before accessing the UART to ensure exclusive access and releases the mutex after the operation is complete.

50.6 STEP4: BUILD THE FLASH THE PROJECT:

1: Build the project in STM32CubeIDE.

2: Flash the firmware onto your STM32 microcontroller.

3: Observe the UART output, which should show interleaved messages from both tasks, demonstrating synchronized access.

CHAPTER#51 CODING ACCESSING HARDWARE DRIVERS FROM MULTIPLE THREADS USING A MUTEX

51.1CODING ACCESSING HARDWARE DRIVERS FROM MULTIPLE THREADS USING A MUTEX:

Accessing hardware drivers from multiple threads using a mutex is essential to prevent race conditions and ensure that only one thread can

access a shared resource at a time. Below is an example demonstrating how to achieve this using an STM32 microcontroller with FreeRTOS.

51.2 PREREQUISTES:

51.2.1STM32CUBEMX:

For generating the initial project setup, including FreeRTOS and peripheral initialization.

51.2.2 STM32CUBEIDE:

For coding and debugging the project.

51.3STEP1: SET UP STM32CUBEMX:

1: Open STM32CubeMX and create a new project.

2: Select your STM32 microcontroller model.

3: Enable the peripherals you plan to use (e.g., UART, GPIO).

4: Enable FreeRTOS under Middleware > FREERTOS.

5: Configure the system clock.

6: Generate the project and open it in STM32CubeIDE.

51.4 STEP2: MODIFY 'FREERTOSCONFIG.H':

Ensure FreeRTOSConfig.h includes necessary configurations for multi-threading, such as enabling preemption, setting the heap size, and other relevant settings.

51.5 STEP3: IMPLEMENT THE MAIN CODE:

In main file, you will create tasks and ensure proper access to hardware drivers.

51.5.1CODE OF CODING ACCESSING HARDWARE DRIVERS FROM MULTIPLE THREADS USING A MUTEX:

```
#include "main.h"
#include "cmsis_os.h"
/* UART handle declared in main.h and initialized in
MX_USART2_UART_Init() */
UART_HandleTypeDef huart2;
/* Task Handles */
osThreadId_t task1Handle;
osThreadId_t task2Handle;
/* Task function prototypes */
void StartTask1(void *argument);
void StartTask2(void *argument);
/* UART Mutex */
osMutexId_t uartMutexHandle;
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
int main(void) {
    /* STM32Cube HAL initialization */
    HAL_Init();
    SystemClock_Config();
    /* Initialize all configured peripherals */
```

```
MX_GPIO_Init();
MX_USART2_UART_Init();
/* Initialize FreeRTOS */
osKernelInitialize();
/* Create the mutex */
const osMutexAttr_t uartMutexAttr = {
  "uartMutex"
};
uartMutexHandle = osMutexNew(&uartMutexAttr);
/* Create the thread(s) */
const osThreadAttr_t task1Attr = {
  .name = "task1",
  .priority = (osPriority_t) osPriorityNormal,
  .stack_size = 128
};
task1Handle = osThreadNew(StartTask1, NULL, &task1Attr);
const osThreadAttr_t task2Attr = {
  .name = "task2",
  .priority = (osPriority_t) osPriorityNormal,
  .stack_size = 128
};
task2Handle = osThreadNew(StartTask2, NULL, &task2Attr);
/* Start scheduler */
```

```
    osKernelStart();
    /* Infinite loop */
    while (1) {
    }
}
void StartTask1(void *argument) {
    for (;;) {
        /* Lock the mutex */
        osMutexAcquire(uartMutexHandle, osWaitForever);
        /* Access the UART */
        char *msg = "Task 1 is running\r\n";
        HAL_UART_Transmit(&huart2, (uint8_t *) msg, strlen(msg),
HAL_MAX_DELAY);
        /* Release the mutex */
        osMutexRelease(uartMutexHandle);
        /* Delay to simulate work */
        osDelay(1000);
    }
}
void StartTask2(void *argument) {
    for (;;) {
        /* Lock the mutex */
        osMutexAcquire(uartMutexHandle, osWaitForever);
```

```
        /* Access the UART */
        char *msg = "Task 2 is running\r\n";
        HAL_UART_Transmit(&huart2, (uint8_t *) msg, strlen(msg),
HAL_MAX_DELAY);
        /* Release the mutex */
        osMutexRelease(uartMutexHandle);
        /* Delay to simulate work */
        osDelay(1000);
    }
}
void SystemClock_Config(void) {
    /* Configure the system clock (This is usually auto-generated by
STM32CubeMX) */
    /* Implementation depends on the specific STM32 microcontroller
and clock configuration */
}
static void MX_GPIO_Init(void) {
    /* Initialize GPIO (This is usually auto-generated by
STM32CubeMX) */
    /* Implementation depends on the specific GPIO configuration */
}
static void MX_USART2_UART_Init(void) {
    /* Initialize USART2 (This is usually auto-generated by
STM32CubeMX) */
    huart2.Instance = USART2;
```

```
    huart2.Init.BaudRate = 115200;
    huart2.Init.WordLength = UART_WORDLENGTH_8B;
    huart2.Init.StopBits = UART_STOPBITS_1;
    huart2.Init.Parity = UART_PARITY_NONE;
    huart2.Init.Mode = UART_MODE_TX_RX;
    huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
    huart2.Init.OverSampling = UART_OVERSAMPLING_16;
    if (HAL_UART_Init(&huart2) != HAL_OK) {
        Error_Handler();
    }
}
void Error_Handler(void) {
    /* User can add his own implementation to report the HAL error
return state */
    while (1) {
    }
}
```

51.5.2 EXPLANATION:

51.5.2.1MUTEX CREATION:

A mutex is created to manage access to the UART resource.

51.5.2.2TASK CREATION:

Two tasks are created (StartTask1 and StartTask2), each attempting to access the UART.

51.5.2.3TASK FUNCTION:

Each task locks the mutex before accessing the UART to ensure exclusive access and releases the mutex after the operation is complete.

51.5.2.4 PERIPHERAL INITILIZATION:

UART and GPIO initialization functions are auto-generated by STM32CubeMX.

51.6 STEP4: BUILD THE FLASH THE PROJECT:

1: Build the project in STM32CubeIDE.

2: Flash the firmware onto your STM32 microcontroller.

3: Observe the UART output, which should show interleaved messages from both tasks, demonstrating synchronized access.

CHAPTER#52 CODING CREATING THEADS MANUALLY

52.1CODING CREATING THEADS MANUALLY:

Creating threads manually without using an RTOS on an STM32 microcontroller can be achieved by using timer interrupts or the SysTick timer to create cooperative multitasking. This approach is more complex and less flexible compared to using an RTOS, but it is possible for simple multitasking requirements.

52.2 PREREQUISTES:

52.2.1STM32CUBEMX:

For generating the initial project setup, including FreeRTOS and peripheral initialization.

52.2.2 STM32CUBEIDE:

For coding and debugging the project.

52.3STEP1:SET UP STM32CUBEMX:

1: Open STM32CubeMX and create a new project.

2: Select your STM32 microcontroller model.

3: Enable the peripherals you plan to use (e.g., UART, GPIO).

4: Enable FreeRTOS under Middleware > FREERTOS.

5: Configure the system clock.

6: Generate the project and open it in STM32CubeIDE.

52.4 STEP2: IMPLEMENT COOPERATIVE MULTITASKING:

In your main file, we will create tasks and switch between them manually using the SysTick timer.

52.4.1CODE OF CODING CREATING THEADS MANUALLY:

```
#include "main.h"

/* UART handle declared in main.h and initialized in
MX_USART2_UART_Init() */
```

```
UART_HandleTypeDef huart2;
/* Task function prototypes */
void Task1(void);
void Task2(void);
/* Task function pointers */
void (*currentTask)(void);
/* Task states */
typedef enum {
    TASK1,
    TASK2
} TaskState;
TaskState currentTaskState = TASK1;
/* Function prototypes */
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
void SysTick_Handler(void);
int main(void) {
    /* STM32Cube HAL initialization */
    HAL_Init();
    SystemClock_Config();
    /* Initialize all configured peripherals */
    MX_GPIO_Init();
```

```
    MX_USART2_UART_Init();
    /* Configure SysTick interrupt */
    HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq() / 1000); // 1
ms tick
    /* Set the initial task */
    currentTask = Task1;
    /* Infinite loop */
    while (1) {
        currentTask();
    }
}
void Task1(void) {
    char *msg = "Task 1 is running\r\n";
    HAL_UART_Transmit(&huart2, (uint8_t *)msg, strlen(msg),
HAL_MAX_DELAY);
    HAL_Delay(1000); // Simulate work
    currentTask = Task2; // Switch to Task2
}
void Task2(void) {
    char *msg = "Task 2 is running\r\n";
    HAL_UART_Transmit(&huart2, (uint8_t *)msg, strlen(msg),
HAL_MAX_DELAY);
    HAL_Delay(1000); // Simulate work
    currentTask = Task1; // Switch to Task1
```

```
}
void SysTick_Handler(void) {
    HAL_IncTick();
}
void SystemClock_Config(void) {
    /* Configure the system clock (This is usually auto-generated by
STM32CubeMX) */
    /* Implementation depends on the specific STM32 microcontroller
and clock configuration */
}
static void MX_GPIO_Init(void) {
    /* Initialize GPIO (This is usually auto-generated by
STM32CubeMX) */
    /* Implementation depends on the specific GPIO configuration */
}
static void MX_USART2_UART_Init(void) {
    /* Initialize USART2 (This is usually auto-generated by
STM32CubeMX) */
    huart2.Instance = USART2;
    huart2.Init.BaudRate = 115200;
    huart2.Init.WordLength = UART_WORDLENGTH_8B;
    huart2.Init.StopBits = UART_STOPBITS_1;
    huart2.Init.Parity = UART_PARITY_NONE;
    huart2.Init.Mode = UART_MODE_TX_RX;
```

```
    huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
    huart2.Init.OverSampling = UART_OVERSAMPLING_16;
    if (HAL_UART_Init(&huart2) != HAL_OK) {
        Error_Handler();
    }
}
void Error_Handler(void) {
    /* User can add his own implementation to report the HAL error
return state */
    while (1) {
    }
}
```

52.4.2 EXPLANATION:

52.4.2.1TASK FUNCTION:

Two task functions, Task1 and Task2, simulate work by sending messages over UART and delaying for 1 second.

52.4.2.2TASK SWITCHING:

The tasks switch between each other by changing the currentTask function pointer.

52.4.2.3 SYSTICK CONFIGURATION:

The SysTick timer is configured to generate an interrupt every 1 ms, which increments the HAL tick counter.

52.4.2.4 MAIN LOOP:

The main loop calls the current task function, which performs its work and then switches to the other task.

52.5 STEP3: BUILD THE FLASH THE PROJECT:

1: Build the project in STM32CubeIDE.

2: Flash the firmware onto your STM32 microcontroller.

3: Observe the UART output, which should show interleaved messages from both tasks, demonstrating synchronized access.

CHAPTER#53 CODING SENDING NOTIFICATIONS BETWEEN THEADS

53.1CODING SENDING NOTIFICATIONS BETWEEN THEADS:

Sending notifications between threads in an STM32 microcontroller can be achieved using FreeRTOS. FreeRTOS provides mechanisms such as task notifications, queues, semaphores, and event groups to facilitate communication between tasks (threads).

53.2 PREREQUISTES:

53.2.1STM32CUBEMX:

For generating the initial project setup, including FreeRTOS and peripheral initialization.

53.2.2 STM32CUBEIDE:

For coding and debugging the project.

53.3STEP1: SET UP STM32CUBEMX:

1: Open STM32CubeMX and create a new project.

2: Select your STM32 microcontroller model.

3: Enable the peripherals you plan to use (e.g., UART, GPIO).

4: Enable FreeRTOS under Middleware > FREERTOS.

5: Configure the system clock.

6: Generate the project and open it in STM32CubeIDE.

53.4 STEP2: IMPLEMENT THE MAIN CODE:

In main file, you will create tasks and use task notifications to send notifications between them.

53.4.1CODE OF CODING SENDING NOTIFICATIONS BETWEEN THEADS:

```
#include "main.h"

#include "cmsis_os.h"

/* UART handle declared in main.h and initialized in
MX_USART2_UART_Init() */

UART_HandleTypeDef huart2;

/* Task Handles */

osThreadId_t task1Handle;

osThreadId_t task2Handle;

/* Task function prototypes */

void StartTask1(void *argument);

void StartTask2(void *argument);
```

```
/* Function prototypes */
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
int main(void) {
    /* STM32Cube HAL initialization */
    HAL_Init();
    SystemClock_Config();
    /* Initialize all configured peripherals */
    MX_GPIO_Init();
    MX_USART2_UART_Init();
    /* Initialize FreeRTOS */
    osKernelInitialize();
    /* Create the thread(s) */
    const osThreadAttr_t task1Attr = {
        .name = "task1",
        .priority = (osPriority_t) osPriorityNormal,
        .stack_size = 128
    };
    task1Handle = osThreadNew(StartTask1, NULL, &task1Attr);
    const osThreadAttr_t task2Attr = {
        .name = "task2",
        .priority = (osPriority_t) osPriorityNormal,
```

```
        .stack_size = 128
    };
    task2Handle = osThreadNew(StartTask2, NULL, &task2Attr);
    /* Start scheduler */
    osKernelStart();
    /* Infinite loop */
    while (1) {
    }
}
void StartTask1(void *argument) {
    for (;;) {
        /* Simulate some work */
        HAL_Delay(1000); // Simulate work
        /* Send notification to Task 2 */
        xTaskNotifyGive(task2Handle);
    }
}
void StartTask2(void *argument) {
    for (;;) {
        /* Wait for notification from Task 1 */
        ulTaskNotifyTake(pdTRUE, portMAX_DELAY);
        /* Access the UART */
        char *msg = "Task 2 received notification\r\n";
```

```
        HAL_UART_Transmit(&huart2, (uint8_t *)msg, strlen(msg),
HAL_MAX_DELAY);
    }
}
void SystemClock_Config(void) {
    /* Configure the system clock (This is usually auto-generated by
STM32CubeMX) */
    /* Implementation depends on the specific STM32 microcontroller
and clock configuration */
}
static void MX_GPIO_Init(void) {
    /* Initialize GPIO (This is usually auto-generated by
STM32CubeMX) */
    /* Implementation depends on the specific GPIO configuration */
}
static void MX_USART2_UART_Init(void) {
    /* Initialize USART2 (This is usually auto-generated by
STM32CubeMX) */
    huart2.Instance = USART2;
    huart2.Init.BaudRate = 115200;
    huart2.Init.WordLength = UART_WORDLENGTH_8B;
    huart2.Init.StopBits = UART_STOPBITS_1;
    huart2.Init.Parity = UART_PARITY_NONE;
    huart2.Init.Mode = UART_MODE_TX_RX;
    huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
```

```
    huart2.Init.OverSampling = UART_OVERSAMPLING_16;
    if (HAL_UART_Init(&huart2) != HAL_OK) {
        Error_Handler();
    }
}
void Error_Handler(void) {
    /* User can add his own implementation to report the HAL error
return state */
    while (1) {
    }
}
```

53.4.2EXPLANATION:

53.4.2.1TASK NOTIFICATIONS:

xTaskNotifyGive() is used to send a notification from Task 1 to Task 2. ulTaskNotifyTake() is used in Task 2 to wait for a notification from Task 1.

53.4.2.2TASK CREATION:

Two tasks are created (StartTask1 and StartTask2), each performing its work and using task notifications to communicate.

53.4.2.3 TASK 1:

Simulates some work by delaying for 1 second and then sends a notification to Task 2.

53.4.2.4 TASK 2:

Waits for a notification from Task 1. Upon receiving the notification, it sends a message over UART.

53.4.2.5 PERIPHERAL INITILIZATION:

UART and GPIO initialization functions are auto-generated by STM32CubeMX.

53.5 STEP3: BUILD THE FLASH THE PROJECT:

1: Build the project in STM32CubeIDE.

2: Flash the firmware onto your STM32 microcontroller.

3: Observe the UART output, which should show interleaved messages from both tasks, demonstrating synchronized access.

CHAPTER#54 CONCLUSION

54CONCLUSION:

In conclusion, mastering STM32CubeMX 5 and CubeIDE enhances embedded systems development by providing a comprehensive toolset for configuration and coding. These tools simplify peripheral setup and code generation, significantly reducing development time. With an intuitive interface, STM32CubeMX allows for easy pin configuration and clock tree setup, while CubeIDE integrates debugging and project management features. Leveraging these tools enables efficient firmware development and deployment on STM32 microcontrollers. Mastery of STM32CubeMX 5 and CubeIDE ensures streamlined workflows and improved productivity in embedded systems projects.

REFERENCES:

1: STM32 CUBE MX

2: STM32 CUBE IDE

3: STM32 REFERENCE MANUAL

4: STM32 HAL MANUAL