Name: Aradhya Shrivastava
Roll No: 2021 IMG-015

Date / /
Page No.

ESD Lab Assignment-2

1. What is the purpose of the CMSIS in the ARM-based embedded systems?

Ans 1. The CMSIS, or cortex Microcontroller software Interface standard, serves as a standardised interface for software components in ARM-based embedded systems. It provides a consistent framework for device abstraction, peripheral access, and real-time operating system (RTOS) support. This standardisation facilitates software portability and reusability across different ARM Cortex-M microcontrollers, simplifying development and promoting a more efficient use of resources in embedded systems.

2. Explain the significance of the startup file (startup_stm32fxx.s) in a Keil project.

i) Vector Table Initialization: The startup file initializes the vector table, which is a table of function pointers at the beginning of the program memory.

ii) Processor Initialization: The startup file sets up the initial values for the core registers, such as stack pointer & the program counter

iii) System Initialization: It may also include sections to initialize the system, configure the clock, and setup the memory regions.

iv) Linker Script Integration: The startup file is closely linked with the linker script. The linker script defines the memory layout of the microcontroller.

v) Exception Handling: Exception handlers for various events like faults, interrupts, and system calls are often implemented in the startup file.

3. What is the purpose of the SystemInit function in the STM32 cube initialization code?

i) Clock configuration: 'SystemInit' typically includes code to setup the system clock. Configuring the clock is crucial because it determines the timing for the entire microcontroller and impacts the performance of the system.

ii) Flash Latency Configuration: Adjusting the flash latency is also common in 'SystemInit'. The flash latency is the no. of wait states inserted to

ensure proper flash memory access timing.

iii) Peripheral Initialization: Some low-level peripheral initialization might be performed in 'systemInit'. This could include setting up the NVIC (Nested Vectored Interrupt Controller), enabling certain peripherals.

iv) Default Configuration: 'systemInit' often provides a baseline configuration for the system, ensuring that the microcontroller starts with reasonable default settings.

4. How can you configure and use GPIO pins in Keil for STM 32 microcontrollers?

i) Include Necessary Files: These files typically include device-specific header files. For example, for STM32F4xx microcontrollers, you might include "stm32f4xx.h."

ii) Initialize the GPIO Pins: Use the appropriate 'GPIO_InitTypeDef' structure to set parameters such as pin mode (input, output), speed etc.

iii) Read and write to GPIO Pins: Use the 'GPIO_ReadInputDataBit' and 'GPIO_WriteBit' functions to read and write to GPIO pins, respectively.

iv) Toggle GPIO Pins: To toggle the state of a GPIO pin, you can use 'GPIO_ToggleBits' function.

v) Handle Interrupts (Optional): If you need to handle interrupts on GPIO pins, configure the NVIC and setup the corresponding interrupt service routine.

5. What is the purpose of the HAL Delay function in STM32CubeIDE?

1. Simple Delay Mechanism: 'HAL_Delay' provides a simple and convenient way to introduce delays in the program. It is useful in scenarios where specific amount of time delay is needed.

2. Resolution Based on SysTick: The implementation of 'HAL_Delay' typically relies on the SysTick timer, which is a system timer present in most STM32 microcontrollers.

Date / /
Page No.
GENIUS MIND

3. Blocking Execution: The function operates as a blocking delay, meaning that the program execution is paused during the delay period.

6. How do you use the NVIC in ARM Cortex-M microcontrollers?

i) Enable global Interrupts: Use the '__enable_irq()' function to enable global interrupts.

ii) Configure Interrupt Priority: You need to configure the priority of interrupts based on your applicant's requirements.

iii) Enable / Disable Interrupts: Use the 'NVIC_Enable IRQ' and 'NVIC_Disable IRQ' function.

iv) Set Pending Interrupt: Use the 'NVIC_Set Pending IRQ' function.

v) Clear Pending Interrupt: Use the 'NVIC_Clear Pending IRQ' function.

vi) Check Active Interrupt: The 'NVIC_GetActive' function.

vii) System Handler Control and Status: Use function like 'NVIC_SystemReset'.

viii) Interrupt service Routines: Write your ISRs for specific interrupts, using '__irq' key word (compiler-dependent).

7. Explain the role of the linker script (STM32F446xxFLASH.ld) in the Keil Project.

i) Memory Layout Definition: The linker script defines the memory layout of the STM32 microcontroller, specifying the start and end addresses of different memory regions.

ii) Memory Section Placement: It determines how different sections of your program are placed in memory.

iii) Symbol Definitions: The linker script contains definitions for various symbols that are used by the linker to resolve adresses.

iv) Stack and Heap Configuration: It defines the stack and heap regions, which are crucial for managing function call execution and dynamic memory allocation.

v) Initialization Code: The linker script may include sections for specifying where the startup code and initialization routines should be placed in memory.

Date / /
Page No.
GENIUS MIND

8. What is the purpose of the vector table in ARM Cortex-M microcontrollers?

i) Exception Handling: Vector Table contains entries for various exceptions and interrupts that can occur during program execution.

ii) Reset Vector: The first entry in the vector table is the reset vector.

iii) (ISRs): For each interrupt source or exception type, there is a specific entry in the vector table.

iv) Fault Handling: The vector table includes entries for fault exceptions, such as hard faults, memory faults etc.

v) User-defined vectors: In some micro-controllers particularly those with an external interrupt controller, vector table may include entries for user-defined interrupt sources.

9. How do you use the UART (USART) peripheral for serial communication in Keil?

i) Include Necessary files (eg. "stm32f9xx.h") and the USART library.

ii) Configure GPIO pins that are used for UART communication.

iii) Configure UART settings such as baud rate, data bits, stop bits and parity.

iv) Enable UART and interrupts, if needed like receiving data

v) Implement UART interrupt Handler to process received data or handle other events.

vi) Send and Receive Data use functions like 'USART_sendData' and 'USART_ReceiveData'.

10. What are the steps involved in configuring and using an external interrupt on an STM32 microcontroller in Keil?

i) Include Necessary files in your project such as device-specific header file.

ii) Configure GPIO Pins to be used for the external interrupt.

iii) Configure External Interrupt by configuring its settings, such as trigger Edge and priority.

iv) Implement external interrupt Handler to handle the external interrupt.

v) Enable Global Interrupts and the external interrupt.