1. V cycle development process

a. a software development process model that represents a variation of the traditional Waterfall method.
b. It is called the V-Model because its graphical representation looks like the letter "V."
c. It emphasizes the relationship between the different stages of development and their corresponding testing phases (on the right side of the "V").
d. This model highlights verification and validation activities during the development process.

(Development Phases - Verification)

a. **Requirements Analysis**
The first stage involves gathering and documenting user and system requirements. This stage focuses on understanding the user's needs.
[**Corresponding testing**: Acceptance Testing is planned at this stage to validate user needs at the end of the process.]

b. **System Design**
In this phase, the overall system architecture and high-level design are created. It defines the system components, their interactions, and data flow.
Corresponding testing: System Testing will later verify the entire system's functioning against the system design.

c. **High-Level Design (HLD)**
The system is broken down into modules and components. The HLD phase describes the system’s architecture, interfaces, and key components at a higher level.
Corresponding testing: Integration Testing is planned to ensure that the individual modules interact as intended.

d. **Low-Level Design (LLD)**
Each module or component is designed in detail. This includes defining the logic for individual modules, data structures, and algorithms.
- **Corresponding testing**: Unit Testing will be done to verify that each individual component works as specified.

e. **Implementation/Coding**
The actual code is written in this phase based on the LLD. It is the phase where developers implement the design into a functional program.

Right Side (Testing Phases - Validation)

The right side of the V-Model represents various levels of testing, which are linked to their respective development stages.

Unit Testing
- Testing individual components or modules. This phase ensures that each unit functions correctly according to the LLD.

Integration Testing
After units are tested individually, they are integrated, and their interactions are tested to ensure they work well together. This corresponds to the HLD phase.

System Testing
- The entire system is tested as a whole to verify that all system components are working correctly and fulfilling the specified requirements.

Acceptance Testing
- This final phase involves testing the system in the real environment by the end user or client. It ensures the system meets user requirements and is ready for deployment. This corresponds to the initial requirements analysis phase.

Key Characteristics of the V-Model:
- **Emphasis on Testing**: One of the V-Model's main strengths is that it integrates testing early in the development lifecycle, reducing the chances of defects later in the process.
- **Sequential Flow**: Like the Waterfall model, each stage of the V-Model must be completed before moving to the next phase.
- **Verification and Validation**: The V-Model enforces both verification (does the system meet the design specifications?) and validation (does the system meet the user's needs?).

Advantages:

- ****Clear and Structured Approach****: Each phase has specific deliverables, making the process easy to understand and manage.
- ****Early Defect Detection****: Since testing is planned alongside development phases, defects can be identified early, reducing potential issues later in the project.
- ****Rigorous Documentation****: The model encourages thorough documentation, which helps in project transparency and future maintenance.

Disadvantages:

- ****Lack of Flexibility****: Like the Waterfall model, changes to requirements or design can be expensive and difficult to implement once the project has progressed through certain stages.
- ****Not Ideal for Complex or Agile Projects****: The V-Model works best when requirements are clear from the start and do not change often, which makes it less suitable for projects with evolving requirements or fast-paced development.

When to Use the V-Model:

- Projects where requirements are well-defined and stable.
- Systems that require a high degree of reliability and rigorous validation, such as healthcare or aerospace systems.
- Projects where quality assurance and testing are critical from the beginning.

5. Indicate the microcontroller parameters (see TIVA-Data-Sheet documentation available on

Moodle):

a. Size of the memories (explain the different kind of memories)

■ SRAM – static random-access memory

Size: 32 KB

Purpose: This is the high-speed, volatile memory where data is stored during program execution (runtime).

It holds variables, the stack, and other dynamic data structures.

Characteristics:

Volatile: Data is lost when power is turned off.

Fast access times compared to flash memory.

Usage: Used for **temporary data, including variables, buffers, and execution stacks**.

Address - internal SRAM is mapped at address **0x2000.0000**

The SRAM is divided into two **32-bit-wide banks**:

- **Even Bank**: Contains all even-addressed words.
- **Odd Bank**: Contains all odd-addressed words.

Dual-Bank Design

The two banks (even and odd) are designed to improve performance by allowing simultaneous operations across both banks:

- A **write operation** to one bank (e.g., the even bank) followed by a **read operation** from the other bank (odd bank) can happen without any delays, i.e., back-to-back in successive clock cycles.
- However, if a **write** is followed by a **read** on the **same bank** (e.g., both operations happen on the even bank), the microcontroller incurs a **stall** of **one clock cycle** due to bank contention.

[If you write data to the **same bank** (either the even or odd memory bank) and then immediately try to read data from that same bank, the microcontroller will experience a **one clock cycle delay**. This delay happens because the system needs time to manage both the write and the read from the same bank, and it can't do both operations at the same time. This is called **bank contention**.]

■ Flash memory

Size: 256 KB

Purpose: This is the **non-volatile** memory where the program code is stored. Flash memory retains its contents even after the power is turned off.Can also store constant data.

- Supports in-system reprogramming via software (**write and erase operations during runtime**).

Access Speed: Fast read access, but writing and erasing are slower.

Usage: Ideal for program code, non-volatile storage of configurations, and data logs.

■ **EEPROM** Electronically erasable programmable read only memory

Size: **2** KB

Purpose: Non-volatile memory used to store small amounts of data that must be retained after power-down, such as **configurations, logs, and settings**.

Unlike Flash memory, EEPROM allows for **byte-level access and has more write cycles** before wear-out. [**EEPROM** allows you to **read** or **write** data **one byte at a time** (a byte is 8 bits). This means you can modify individual bytes of data without affecting the rest of the memory.]

Usage: Store user configurations, calibration data, or other settings that need to be preserved between power cycles but are frequently updated.

■ Internal ROM loaded with TivaWare™ for C Series software

Size: 2 KB (Boot ROM)

Purpose: Stores the microcontroller's factory default bootloader.

The bootloader in ROM enables firmware updates via various interfaces (e.g., UART, I2C, SPI).

Characteristics: Non-volatile and read-only; used only for bootstrapping the system.

Usage: Responsible for the initial boot process and loading of user programs into the system.

Components in ROM:

1. **TivaWare™ Boot Loader and Vector Table**:
    - The **boot loader** is a small program that loads and runs the main application, especially when **Flash memory** is empty (e.g., during the initial setup or after a reset).
    - It also allows for **firmware upgrades** by jumping back to the boot loader when needed.
    - The **vector table** contains **addresses of interrupt and exception handlers**, which are essential for handling system events. [The **vector table** is a list stored in memory that holds the **addresses** of functions (also called **interrupt handlers** or **exception handlers**) used to respond to specific events]
    - Interrupt and Exception Handlers:

Interrupts: When a hardware event occurs (e.g., a button is pressed or data arrives via a serial port), the microcontroller temporarily stops its current task to handle the event. The interrupt handler is the function that runs in response to the event.

B **Exceptions**: These are special types of system errors or conditions (e.g., illegal memory access). The exception handler deals with these situations to prevent the system from crashing.

2. **TivaWare Peripheral Driver Library (DriverLib)**:
    - The **Driver Library (DriverLib)** contains functions (APIs) for interacting with the microcontroller's peripherals (like GPIOs, UART, I2C, timers, etc.).
    - These pre-programmed APIs in ROM reduce the need to store peripheral drivers in **Flash memory**, freeing up space for user applications. Instead of writing and storing your own driver code, you can call the **ROM-based** functions directly.
3. **Advanced Encryption Standard (AES) Cryptography Tables**:
    - **AES** is a widely-used encryption standard, especially in **security applications**.
    - The ROM includes **precomputed tables** that support AES encryption, allowing the microcontroller to perform encryption tasks more efficiently without using extra **Flash** space.
4. **Cyclic Redundancy Check (CRC) Error Detection**:
    - CRC is a widely used technique to detect errors in data. It ensures that data being transmitted or stored has not been corrupted or altered.
    - It works by calculating a checksum (a small numerical value) for a block of data before transmission or storage. When the data is received or retrieved, the CRC checksum is recalculated and compared to the original value. If the checksums match, the data is likely intact. If they don't match, it indicates that the data has been corrupted.
    - The microcontroller has the CRC functionality built into its **ROM**, meaning that the CRC algorithm is pre-installed and does not need to be stored or implemented by the user in **Flash memory**.
    - By having the CRC function in ROM, applications can use it without needing to store or write their own CRC code, which saves space in **Flash memory** (which has limited capacity).

b. Clock frequency (explain)

- Clock frequency refers to the **rate** at which a **processor or microcontroller's clock generates pulses to synchronize operations**
- Measured in **hertz (Hz)**.
- In the context of a **CPU**, the clock frequency controls **how many** cycles (or **instructions**) can be executed per second
- **Higher frequencies** generally allow for **faster processing of instructions**.
- **Main Clock Frequency (System Clock)**

    - primary clock signal that drives the central processing unit (CPU) and other core components of a digital system, like a microcontroller or processor
    - This is the **master clock** that sets the timing for the entire system.
    - It provides a timing reference that all other components (such as the CPU, memory, and peripherals) follow.
    - This clock speed can be derived from either an external crystal oscillator or an internal oscillator
- **internal clock sources** to provide flexibility in balancing performance, power consumption, and accuracy.
    - **Precision Internal Oscillator (PIOSC)**
        - fixed frequency of 16 MHz
        - **low power consumption** is important or when **precise timing** is not critical.
        - **PIOSC** can be used when the system doesn't need maximum performance, such as in **sleep modes** or during periods of light activity.
    - **Main Oscillator** (External Crystal Input)
        - Crystal frequencies in the range of **4 MHz to 25 MHz**
        - **Main Oscillator** can use an **external crystal** or resonator to generate clock signals.
        - The main oscillator is typically used when **precision timing** is important, such as in **communication protocols** (e.g., UART, SPI, I2C) or when **generating highly accurate time delays**.
        - It is also used when the system requires the highest possible performance, because its frequency can be **multiplied** using a **Phase-Locked Loop (PLL)** to generate the **system clock frequency**, which can go up to **80 MHz**.

c. Explain the utility of a floating-point unit
- The Floating-Point Unit (FPU) in the Cortex-M4F - designed to handle **complex math operations**, especially with **floating-point numbers** (like decimal numbers) efficiently.
- **Single-Precision Operations**: It can handle 32-bit floating-point numbers, which are commonly used in programming as "float" in C.
-
- **Fused Multiply-Accumulate (MAC)**: It can multiply two numbers and then add a third in one step, which is faster and more accurate.
- **Math Support**: The FPU can do basic math like addition, subtraction, multiplication, division, and even square roots.
- **Special Cases**: It supports tricky cases like very small numbers (denormals) and different ways of rounding numbers, which are part of the IEEE 754 standard (a set of rules for how computers should handle floating-point math).
- **Registers**: It has 32 special 32-bit registers to store floating-point numbers. These can also be used as 16 double-sized registers when needed.
- **Pipeline**: The FPU has a three-stage system for processing instructions efficiently without slowing down the processor.

d. Describe some embedded peripherals

1. **General-Purpose Input/Output (GPIO**)
GPIO Ports: **43 programmable I/O pin**s, divided into ports (Port A through Port F).
Can be configured for input or output.
Supports **digital I/O operations** and can **trigger interrupts** on certain conditions (rising/falling edges, levels).

2. **Timers**
16/32-bit General-Purpose Timers: Six timers that can operate in different modes (one 32-bit timer or two 16-bit timers).
Useful for tasks like **time delays**, p**ulse width modulation (PWM), and capturing input signals**.

3. **Analog-to-Digital Converter (ADC)**
12-bit ADC: **Two** ADC modules with up to 12 input channels.
Converts analog signals (e.g., from sensors) **into digital values** for the microcontroller to process.
Supports both single-ended and differential measurements.

4. **Pulse Width Modulation (PWM)**
PWM Module: **Two PWM modules**, each with four PWM generators.
Generates PWM signals, which are **useful in controlling** motors, LEDs, or other **devices that require variable power.**

5. **Universal Asynchronous Receiver/Transmitter (UART)**
8 UART Modules: For serial communication, used to interface with devices like sensors, modems, or other controllers.
Supports features like hardware flow control, parity, and different baud rates.

6. **Inter-Integrated Circuit (I2C)**
4 I2C Modules: Provides **communication with I2C-compatible devices** like EEPROMs, sensors, or other microcontrollers.
Supports both **master and slave modes.**

7. **Synchronous Serial Interface (SSI)**
4 SSI Modules: Used for SPI communication (Serial Peripheral Interface).
Enables high-speed communication with SPI-based peripherals like flash memory, sensors, or displays.

8. **Controller Area Network (CAN)**
CAN Module: Allows communication in automotive and industrial environments where multiple microcontrollers need to talk to each other reliably.
Supports standard and extended message frames, with filters for message management.

9. **Direct Memory Access (DMA)**

UDMA Controller: For high-speed data transfers between peripherals and memory without involving the CPU, improving performance for tasks like ADC sampling or UART data handling.

10. **Watchdog Timer**
Watchdog Timer: **A safety feature that resets the system if the software gets stuck, ensuring the system can recover from errors**.

11. **Real-Time Clock (RTC**)
RTC Feature in the Hibernation Module: Can keep **track of time even when the system is powered down**, useful for timekeeping in low-power applications.

12. **Universal Serial Bus (USB)**
USB 2.0 Controller: Supports both Host and Device modes.
Can be used for USB communication in applications **requiring file transfer or peripheral control.**

13. **Analog Comparators**
3 Analog Comparators: Compare two analog input voltages and generate digital outputs, useful in applications like threshold detection.

14. **QEI (Quadrature Encoder Interface**)
QEI Module: For interfacing with rotary encoders to **measure rotational position and speed, often used in motor control applications.**