Research Work – 3

- 31-01-2023

1> What is an IDE and compiler ? List out differences.

-->IDE stands for Integrated Development Environment. It is a software application that provides facilities for developing software.

so basically, an IDE is an application that facilitates application development, and gives you a central interface featuring all the tools you'll need like:

- A code editor that's designed to help you write and edit your code. It also helps you make it more readabile and clean.
- A compiler that transforms code written by a human into machine-readable form.
- A debugger that helps you eliminate errors from your programs so that your code executes and performs the way it should. This feature provides tools to help you examine your code.

## Compiler:

- A compiler is a translator that converts the high-level language into the machine language.
- High-level language is written by a developer and machine language can be understood by the processor.
- Compiler is used to show errors to the programmer.
- The main purpose of compiler is to change the code written in one language without changing the meaning of the program.
- When you execute a program which is written in HLL programming language then it executes into two parts.
- In the first part, the source program compiled and translated into the object program (low level language).
- In the second part, object program translated into the target program through the assembler.

2>What is a Bootloader and how does it work?

---> The boot-loader is a short program used to burn the firmware to the microcontroller without any programmer device either like FLASH or volatile like RAM and jumps to the desired program from there it takes care of execution. The process of burning the provided data to the program memory is controlled by the boot-loader. A boot-loader is a program which will be the first thing to run and can load other applications into specific places in memory and is provided to the serial interface.

A boot-loader is an application whose primary purpose is to allow the systems software that has to be updated without using any specialized hardware such as a JTAG programmer.

- The boot-loader manages the systems images.
- It receives new program information externally via some communication means and writes that information to the program memory of the processor.
- They can communicate over a variety of protocols such as USART, CAN, I2C, Ethernet, USB and the list goes on for as many protocols that exist.
- There are many different sizes and varieties to embedded boot-loaders.

3> What do you understand by OTA update

--->OTA updates provide a convenient and efficient way to deliver software updates and bug fixes, as well as new features, to a large number of devices at once.

This reduces the need for manual updates and eliminates the need for users to physically connect their devices to a computer.

this makes it easier and more convenient to keep devices up-to-date, as well as improving their functionality and security.

4>List the differences between baremetal vs RTOS programming

--->Baremetal and RTOS (Real-Time Operating System) programming are two different approaches to programming embedded systems. The key differences between the two are:

| Sr no | Feature | Baremetal | RTOS |
|---|---|---|---|
| 1 | Operating System | : Baremetal programming runs directly on the hardware without an operating system | while RTOS programming uses a real-time operating system to manage the system resources and provide scheduling and synchronization services |
| 2 | Resource Management | In BareMetal programming, the programmer has to manage all the system resources, including memory, peripherals, and other hardware components. | In RTOS programming, the operating system handles most of the resource management, freeing the programmer to focus on application-specific tasks. |
| 3 | Real-Time Performance | BareMetal programming can provide deterministic real-time performance,<br>but it requires careful and complex coding, as well as the management of all system resources | RTOS programming offers higher reliability and more predictable real-time performance,<br>as the operating system takes care of many of the low-level details |
| 4 | Development Time: | Bare Metal programming can take longer to develop, as the programmer has to manage all the system resources, | while RTOS programming can be quicker and more straightforward, as the operating system provides many of the required services |
| 5 | Portability | Bare Metal programming is often less portable, as it is tied to the specific hardware platform, | while RTOS programming is typically more portable, as the operating system abstracts many of the low-level hardware details. |

5>How to choose between baremetal and RTOS for project

-->The choice between programming and RTOS programming for a project depends on the requirements and constraints of the project, including:

| Sr no | Feature | BareMetal | RTOS |
|---|---|---|---|
| 1 | Real-Time Performance | If real-time performance is a critical requirement, Bare Metal programming may be a better option as it can provide deterministic real-time performance. | However, RTOS programming can also provide real-time performance, especially if the operating system has been optimized for real-time behavior |
| 2 | System Resources | If the system has limited resources, such as low memory or processing power,<br>Bare Metal programming may be more appropriate, as it eliminates the overhead of an operating system. | On the other hand, if the system has more resources, an RTOS may provide more efficient resource management. |
| 3 | Development Time | Bare Metal programming especially for projects that require simple resource management or real-time behavior | If development time is a major consideration, RTOS programming may be a better option, as it can be quicker and more straightforward than Bare Metal programming especially for projects that require complex resource management or real-time behavior |
| 4 | Portability | Bare Metal programming can be less portable, as it is tied to the specific hardware platform. | RTOS programming may be a better option, as the operating system abstracts many of the low-level hardware details. |

If the development team has experience with Bare Metal programming or RTOS programming

it may be best to stick with the approach they are familiar with. If the team is new to embedded systems development,

RTOS programming may be a good place to start, as it can provide a higher-level abstraction and more services

In conclusion, the choice between Bare Metal and RTOS programming depends on the specific requirements and constraints of the project.

It's important to carefully consider the trade-offs between the two approaches and choose the one that best fits the project needs.