TechDhaba Embedded Systems Division

Module 1 – Zephyr RTOS Architecture & Modern Embedded Evolution

“Turning modularity into momentum — the new face of real-time systems.”

Prepared by: **TechDhaba Embedded Systems Team**

Date: October 12, 2025

1. Introduction to Zephyr RTOS

Zephyr RTOS is a modular, vendor-neutral, open-source operating system designed for connected and resource-constrained devices. It supports multiple architectures such as ARM, RISC-V, ARC, and x86. Zephyr provides a unified environment for building scalable, secure, and real-time applications — bridging the gap between microcontroller-level control and modern cloud-integrated workflows.

Feature	Zephyr RTOS	FreeRTOS	Linux Kernel
Architecture	Microkernel-inspired, modular	Monolithic task scheduler	Monolithic
User Space	Supported via MPU isolation	Not available	Full user/kernel separation
Device Tree	Yes (Linux-style)	No	Yes
SMP Support	Yes	No	Yes
Security	MPU + TEE + MCUBoot	Minimal	SELinux/AppArmor

2. Kernel Architecture Overview

Zephyr's kernel is microkernel-inspired, emphasizing modularity and determinism. It provides thread management, scheduling, interrupts, memory domains, synchronization, and timing services. The modular design allows inclusion of only required components, minimizing memory footprint.

Key Components include:

- Kernel Scheduler and Thread Management
- Inter-thread Communication and Work Queues
- Interrupt Service Routine (ISR) Handling
- Memory Management and Stack Protection
- Synchronization Primitives (Mutexes, Semaphores, Queues)
- Timing and Clock Services

Kernel Structure:

```
+--------------------+
| Application Space |
+--------------------+
| System Services    |
+--------------------+
| Kernel Core        |
+--------------------+
| HAL / Drivers      |
+--------------------+
| Hardware Layer     |
```

3. User Space vs Kernel Space

Zephyr introduced a user-space mechanism similar to Linux kernel design, where user threads execute with restricted privileges under the MPU (Memory Protection Unit). Kernel threads handle system-critical operations, ensuring robust isolation between application and kernel layers.

```
K_THREAD_DEFINE(app_task, 1024, user_fn, NULL, NULL, NULL, 3, K_USER, 0);
```

4. Memory Management & Protection

Zephyr provides multiple memory allocation mechanisms, including heap, pool, and slab allocators. Each thread has its own stack, and MPU regions protect memory domains from unauthorized access. This approach ensures determinism while maintaining safety criticality.

```
K_HEAP_DEFINE(app_heap, 2048);
void *mem = k_heap_alloc(&app_heap, 128, K_NO_WAIT);
```

5. Device Driver Framework & Device Tree

Zephyr adopts the Linux-inspired Device Tree model for describing hardware. Each device is instantiated dynamically at boot time through DEVICE_DT_DEFINE() macros, ensuring portability across MCUs and boards.

```
&i2c1 {
    status = "okay";
    ssd1306@3c {
        compatible = "solomon,ssd1306fb";
        reg = <0x3c>;
    };
};
```

6. Why Zephyr Is in High Demand

Zephyr is quickly becoming the standard for modern embedded development due to its vendor-neutral design, open governance, and Linux-style development workflow. Its ability to bridge MCU firmware with cloud infrastructure and IoT frameworks has made it indispensable for companies building scalable embedded ecosystems.

Industry Adoption:

• Intel – AI sensor hubs & edge processing units
• Nordic – nRF Connect SDK foundation
• Google – Embedded TensorFlow Lite support
• NXP – Automotive control systems
• Bosch – Industrial IoT edge devices

Market Insight: Zephyr-related job demand has grown 500% in two years, especially across IoT, Edge AI, and Automotive sectors.

7. TechDhaba Insights

At TechDhaba, we integrate Zephyr RTOS into our embedded training and consulting ecosystem. Our engineers use Zephyr to bridge AI inference workloads on MCUs, real-time multimedia streaming, and connected sensor networks. Zephyr's modular and secure design aligns perfectly with our belief — turning complexity into clarity.

"Zephyr is the Linux of Microcontrollers — modern, secure, and scalable."