In this assignment, I’ll go through some of the most frequent types of real-time operating systems. Considering the massive number of real-time operating systems, I can analyze a comparatively small number of them. Further in this assignment, I’ll describe some of the most frequent processing units in such embedded systems.

1. **RTOS types and differences**:

From a perspective, one can divide the operating systems in this area into two main categories: 1. *Open Source* RTOS, 2. *Proprietary* RTOS.

|  |  |
| --- | --- |
| Open Source RTOS | Description |
| Linux | Primary Linux kernel source. |
| uClinux | A version of Linux without memory management unit; Supports ARM, Coldfire, etc. |
| FreeRTOS | Very light and suitable for ARM, AVR32, MSP430, etc. |
| RTAI | A real-time application for Linux, with support for x86, PowerPC and ARM. |
| coscox | An embedded real-time multi-task OS, especially for ARM Cortex M4, M3a and M0 chipset. |

|  |  |
| --- | --- |
| Proprietary RTOS | Description |
| QNX | A real-time RTOS with support for ARM, MIPS, PowerPC and X86. |
| VxWorks | Another real-time RTOS with support for ARM, PowerPC, Coldfire and MIPS. |
| INTEGRITY | A hard real-time RTOS with support for ARM, PowerPC, Coldfire, X86. |
| ThreadX | The Express Logic’s advanced RTOS, designed for deeply embedded applications. Supports ARM, PowerPC and X86. |
| coscox | An embedded real-time multi-task OS, especially for ARM Cortex M4, M3a and M0 chipset. |

* **Hard Real-time Operating System**: A hard real-time system (also known as an immediate real-time system) is hardware or software that must operate within the confines of a stringent deadline. The application may be considered to have failed if it does not complete its function within the allotted time span.
* **Deeply Embedded Applications**: Embedded systems are fixed-function. They may offer very high or low performance, with a limited energy footprint. Deeply embedded systems are single-purpose devices that detect something in the environment, perform a basic level of processing, and then do something with the results.

A more detailed comparison of aforementioned operating systems is given below:

* **QNX** has the following features:

1. CPU Scheduling
2. Interprocess Communication
3. Interrupt Redirection
4. Interrupt Timers
5. Everything else runs as a user process
6. Memory Management Unit

* **uClinux** has the following features:

1. Lacks memory management unit
2. CPU Scheduling
3. Interprocess Communication
4. Interrupt Redirection and Handling with Timers

* **VxWorks** has the following features:

1. Provides robust safety and security certification, for industries such as aerospace and defense, medical devices and robotics.
2. Can be used in asymmetric multiprocessing and symmetric multiprocessing
3. Useful for IoT applications
4. Multitasking kernel with preemptive and round-robin scheduling and fast interrupt response.
5. Native 64-bit operating system.
6. Memory protection mechanisms.
7. Bluetooth, USB and CAN protocols.
8. Local and distributed message queues.
9. Error handling framework.
10. High Reliability File System (HRFS), Network File System (NFS) and FAT-based file system
11. Graphical User Interface; OpenVG stack, Open GL, Tilcon UI.