Operační Systém Reálného Času

Obsah

1 Úvod

RTOS (Real-Time Operating System) je operační systém určený pro aplikace, které musí reagovat na události v předem známém čase. Nejde tedy primárně o rychlost, ale o předvídatelnost - úloha musí být provedena ve správný okamžik.

RTOS obvykle nebývá tak velký a komplexní jako běžné operační systémy (Windows, Linux), ale naopak je co nejmenší, efektivní a uzpůsobený pro běh na mikrokontrolérech s omezenými zdroji (RAM, flash paměť, rychlost CPU).

1.1 Hlavní vlastnosti RTOS

- **Determinismus** . systém dokáže garantovat, že úloha bude spuštěna ve stanoveném čase (tzv. deadlines).
- **Multitasking** - schopnost spouštět více úloh (tasks) souběžně.
- **Plánovač (scheduler)** - algoritmus, který rozhoduje, která úloha poběží v daném okamžiku.
- **Prioritizace** - důležité úlohy mají vyšší prioritu a systém jim dává přednost.
- **Nízká režie (overhead)** - aby co nejméně zdrojů padlo na samotný chod RTOSu.

1.2 Kde se RTOS používá

RTOS nachází uplatnění všude tam, kde je nutné časově spolehlivé řízení hardwaru nebo procesů:

- **Vestavěné systémy (embedded systems)** - domácí spotřebiče, IoT zařízení.
- **Automotive** - řídicí jednotky motorů, airbagy, ABS (musí reagovat v řádu milisekund).
- **Robotika a drony** - řízení motorů, stabilizace letu, zpracování senzorů.
- **Telekomunikace** - směrovače, switche, mobilní zařízení.
- **Průmyslová automatizace** - řízení výrobních linek, CNC stroje.
- **Zdravotnická technika** - přístroje, které musí reagovat přesně a spolehlivě.

1.3 Proč se RTOS používá

Použití RTOSu má několik hlavních důvodů:

- **Předvídatelnost** - úlohy musí být vykonány ve správném čase, aby nedošlo k chybě (např. opožděné otevření airbagu = fatální problém).
- **Organizace úloh** - RTOS umožňuje logicky oddělit části programu (např. komunikace, ovládání motoru, zpracování senzorů).
- **Zjednodušení vývoje** - programátor nemusí řešit složité načasování v jedné hlavní smyčce, ale rozloží úlohy do samostatných tasků.
- **Rozšiřitelnost** - přidání nové funkce obvykle znamená jen přidání nové úlohy.
- **Opakovatelnost a testovatelnost** - úlohy běží v definovaných časových intervalech, což usnadňuje testování a ladění.
- **Přenositelnost** - RTOS vytváří jednotné rozhraní pro přístup k periferiím a tvorbě tasků na různých MCU/CPU, díky tomu je jednodušší daný software přenést i na jiný HW.

1.4 Typy real-time systémů

- **Hard real-time** - deadline musí být vždy dodržena (např. řízení airbagu, pacemaker).
- **Firm real-time** - občasné porušení deadline je tolerováno, ale snižuje kvalitu systému (např. streamování audia).
- **Soft real-time** - deadline je důležitá, ale překročení nevede k fatálnímu selhání (např. videohry).

2 Základní části RTOS

2.1 Úlohy (Tasks / Threads)

Úloha (někdy označovaná jako task nebo vlákno) je základní jednotkou běhu programu v RTOSu.

2.1.1 Kód

Funkce, kterou vykonává.

2.1.2 Stav

Vnitřní stav ve kterém se úloha v systému nachází. RTOS se stará o přechody mezi těmito stavy.

- **Running (běží)** - úloha právě využívá CPU.
- **Ready (připravená)** - úloha čeká na spuštění, jakmile jí to plánovač dovolí.
- **Blocked (čeká)** - úloha čeká na splnění podmínky (např. signál, dokončení I/O).
- **Suspended (pozastavená)** - úloha je zastavená až do další aktivace.

2.1.3 Priorita

Číselná hodnota, která určuje, jak je úloha důležitá.

2.1.4 Stack

Vlastní zásobník pro uložení kontextu, který se skládá z lokálních proměnných a návratových adres.

2.2 Plánovač (Scheduler)

Plánovač je jádro RTOSu. Rozhoduje, která úloha poběží v daný okamžik. Existuje několik hlavních algoritmů:

- **Round Robin** - jednoduché střídání úloh v kruhu (každá úloha dostane krátký časový úsek - time slice).
- **Prioritní plánování** - úlohy s vyšší prioritou běží přednostně.
- **Preemptivní plánování** - pokud se objeví úloha s vyšší prioritou, RTOS okamžitě přeruší běžící úlohu a spustí důležitější.
- **Kooperativní plánování** - úloha musí sama předat řízení, RTOS ji nesmí přerušit (jednodušší implementace, ale horší spolehlivost).

  Většina moderních RTOS používá preemptivní prioritní plánování.

2.3 Kontext a jeho přepínání (Context Switching)

Protože více úloh sdílí jedno CPU, musí RTOS při přepnutí úlohy:

- uložit stav registrů (obsah CPU registrů, čítač instrukcí, zásobník)
- obnovit stav jiné úlohy

Tento proces se nazývá **context switch**. Je to klíčová operace RTOSu a musí být co nejrychlejší, aby systém nebyl zahlcen jen přepínáním.

2.4 Synchronizace úloh

Když více úloh spolupracuje nebo sdílí zdroje (např. UART, I2C, paměť), je nutný určitý způsob synchronizace přístupu ke sdíleným zdrojům:

- **Semafory** - binární nebo počítané signály, které určují přístup k prostředku.
- **Mutexy (Mutual Exclusion)** - zajišťují, že prostředek používá vždy jen jedna úloha.
- **Fronty (Queues)** - úlohy si posílají zprávy / data prostřednictvím fronty.
- **Události (Events)** - RTOS umožňuje úlohu probudit, jakmile nastane určitá podmínka.

2.5 Časovač

RTOS má obvykle systémový časovač (tick timer), který:

- udává rytmus pro plánovač (např. každých 1 ms)
- umožňuje nastavit zpoždění (delay) nebo periodické úlohy
- umožňuje měřit čas

2.6 Přerušení (Interrupts)

RTOS spolupracuje s přerušením od hardwaru:

- **ISR (Interrupt Service Routine)** - funkce, která se vykoná při přerušení.
- ISR by měla být co nejkratší a pokud je potřeba složitější akce, měla by ISR jen probudit úlohu, která ji vykoná.

3 Architektura

RTOS architektura se typicky skládá ze tří hlavních částí:

- **Jádro (Kernel)**: Jádro je srdcem každého RTOS. Je zodpovědné za správu úloh, plánování, mezitaskovou komunikaci a správu paměti. Jádro je optimalizováno pro minimalizaci latence a jitteru, což je kritické pro systémy s pevnými termíny.
- **Plánovač (Scheduler)**: Plánovač je klíčovou součástí jádra. Jeho úkolem je rozhodovat, která úloha má v daný okamžik běžet. Používá speciální plánovací algoritmy, jako je preemptivní plánování s prioritami, kde je vždy spuštěna úloha s nejvyšší prioritou. V některých pokročilých RTOS se používají i algoritmy jako Earliest Deadline First (EDF), které upřednostňují úlohy s nejbližším termínem dokončení.
- **Abstrakční vrstva pro hardware (Hardware Abstraction Layer - HAL)**: Tato vrstva odděluje jádro RTOS od konkrétního hardwaru. Díky HAL může být stejné jádro použito na různých procesorech a platformách. HAL řeší nízkoúrovňové detaily, jako je správa přerušení, časovače a komunikace s registry.

3.1 Typy architektur

RTOS lze rozdělit do dvou hlavních kategorií na základě jejich struktury:

- **Monolitická architektura**: V tomto modelu jsou všechny komponenty RTOS, včetně plánovače, správce paměti a ovladačů zařízení, integrovány do jednoho velkého modulu.
- **Mikrojádrová architektura (Microkernel)**: Zde je jádro minimalistické a poskytuje jen základní funkce, jako je plánování a správa mezitaskové komunikace. Ostatní služby (např. ovladače zařízení, souborový systém) běží jako samostatné procesy v uživatelském prostoru. Tento model je robustnější a modulárnější, protože chyba v jedné službě neohrozí celý systém, ale obvykle má vyšší režii.

4 Přepínání kontextu