Cyclic Executive Scheduling  
(Time Triggered Approach)

Embedded Systems Project

**Participant Name:** Siraj Tayyab Khan & Zain-ul-Abidin

**Participant Roll:** MSEE-22002 & MSEE-22007

**University:** Information Technology University, Lahore

**EE-580:** Embedded Systems Design

**Instructor:** Dr. Rehan Ahmed

**Teaching Assistant:** Emad Ali & Rabia Ahmed

**May 23th, 2023**

Content

**[Introduction2](#Introduction)**

[Scheduling Algorithms Overview2](#Introduction)

[Table of comparison2](#Table)

**[Cyclic Executive Scheduling3](#CES)**

[Why do we need CES?3](#CES)

[Notations, Terminologies, and Scheduling Scheme (without violations)3](#CES)

**Implementation4**

Procedure5

Algorithm6

**Results4**

Type chapter title (level 2)5

Type chapter title (level 3)6

**~ (Cyclic Executive Scheduling (Time Triggered Approach)) ~**

**Introduction:**

In the world of Embedded Systems, we define a mechanism or a set of rules by which our embedded device handles various kinds of tasks, and we call it a scheduling scheme. We opt for a particular scheduling scheme based on our scenario of tasks and requirements that we want to meet. Generally, we can classify embedded scheduling scheme based on static nature, periodicity, pre-emption, and dependence like table of some scheduling algorithms, we discussed thoroughly in this course. But despite mentioning all of the given below algorithms, we are particularly interested in **Time Triggered Cyclic Executive Scheduling (TTCES)** for ouriproject.

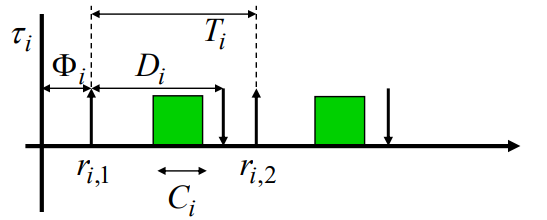
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Static/Dynamic** | **Pre-emptive** | **Periodic/ Aperiodic** | **Dependency** |
| Simple Time **(STT)** | Static | Not | Periodic | No |
| Time Triggered Cyclic Executive **(TTCE)** | Static | Not | Periodic | No |
| Earliest Deadline **Due (EDD)** | - | Not | Aperiodic | No |
| Earliest Deadline First  **(EDF)** | Dynamic | Yes | Aperiodic | No |
| Earliest Deadline First Star **(EDF\*)** | Dynamic | Yes | Aperiodic | Yes |
| Rate Monotonic **(RM)** | Static | Yes | Periodic | No |
| Deadline Monotonic **(DM)** | Dynamic | Yes | Periodic | No |
| Rate Monotonic Polling Server **(RMPS)** | Dynamic | Yes | Both | No |
| Total Bandwidth Server **(TBS)** | Static | Yes | Both | No |
| YDS Online Dynamic Voltage Frequency Scaling **(DVFS)** | Dynamic | - | - | - |
| YDS Offline Dynamic Voltage Frequency Scaling **(DVFS)** | Static | - | - | - |

**Figure 1.1 Table of comparison**

**Cyclic Executive Scheduling (CES):**

Cyclic Executive Scheduling is needed when we want to schedule a taskset (multiple tasks), such that ***each task belonging to the taskset can have different time period cycle of execution to the other member belonging to the same taskset.*** Secondly, taskset needs to respect deadlines for all its members.

Before discussing more about CES, let us define a task. A task is a member of taskset with its task number , and instance where Arrival time is denoted by , Computation time Deadline , Phase , and Time period .



**Figure 1.2 Example of a Task Schedule**

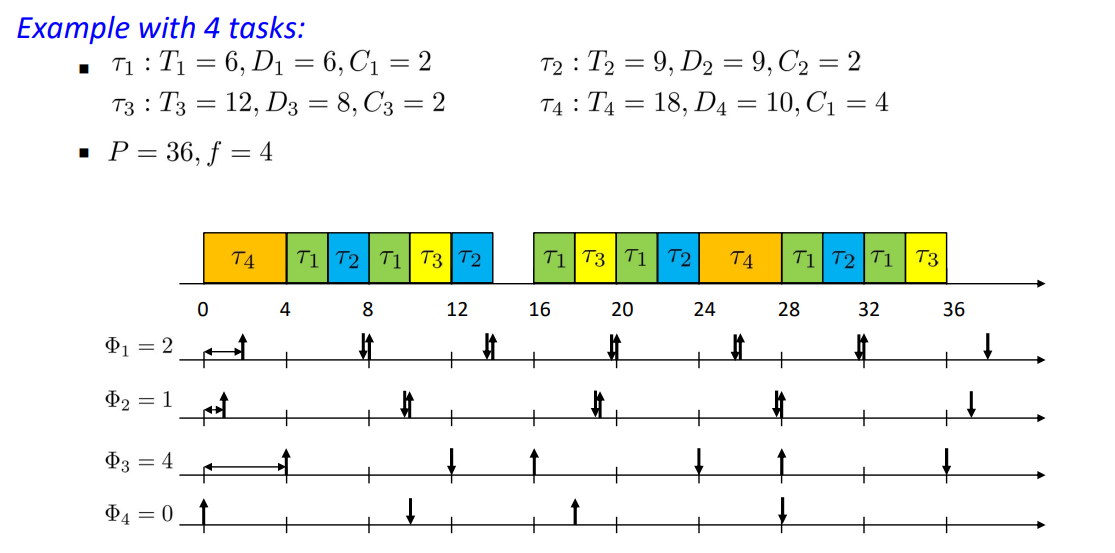
In CES, we first calculate hyper period of the taskset by taking Least Common Multiple (LCM) of all periods of the taskset. Special thing about hyper period is that if we are able to schedule all tasks, and their corresponding instances within the hyper period, then we can just repeat that schedule till forever without needing to calculate any further.

Then we divide whole hype period into frames of frame size , where each frame has at least one or more task scheduled in it. There are other constraints for the frame size as well like a frame size cannot be lesser than the compute time of any member of the taskset, and on the flip side it cannot exceed the time period of any member of the taskset.

Lastly for the frame size that it should always be a positive integer multiple of the hyper period.

There should be at least one full frame between the deadline and arrival of the task.

Last part is that once you have somehow generated your schedule it should not violate things like an instance cannot start before its arrival, and deadlines should be respected irrespective of the phase.



**Figure 1.3 Example of a Taskset Schedule**

**Implementation:**