*E x* *p* *e* *r* *i* *m* *e* *n* *t* *4*

Implementation of PIP and PCP protocol

Objectives:

In this experiment you will learn how to implement the Priority

Inheritence and Priority Ceiling Protocol

**TABLE OF CONTENTS**

* 1. Objective
  2. References
  3. Theoretical Background
  4. Pre Lab Assignment
  5. In-Lab Experimental Procedure
  6. Outcomes

Appendix A : In-Lab Report

* 1. Objective

This experiment aims to implement the priority Inheritance and priority Ceiling protocol.

* 1. References

1. Cheddae Release 3.x user’s guide

* 1. Theoretical Background

In addition to a processor, each job may require some other resource in order to execute. Serially reusable resources are allocated to jobs on a *non-preemptive basis and used in a mutually exclusive manner.* It means when a unit of resource is granted to a job, this unit is no longer available to other jobs until the job frees the unit.

A lock-based concurrency control mechanism assumed to be used to enforce mutual exclusive access to resources. When a job wants to use *ηi units of a* resource *Ri, it executes a lock L(Ri, ηi)* to request them. When the job no longer needs the resources, it releases them be executing an *unlock U(Ri, ηi).* When a lock request fails, the requesting job is blocked and loses the processor. It stays blocked until the scheduler grants the resources the job is waiting for . If a resource has only 1 unit the simpler notations *L(Ri)* and *U(Ri)* are used for lock and unlock

Jobs attempt to lock a resource before starting to use it, and unlock the resource afterwards; the time the resource is locked is the critical section. Critical sections are denoted by [*R, η; e],* where *R* gives the *name* and *η* the *number of* units of a resource and *e* the *(maximum) execution time* of the critical section. If there is only one unit of a resource the simpler notation [*R; e] is used*

* 1. Pre Lab Assignment

Q1: What is critical Instance ?

Q2: What is priority inversion in context to real time scheduling?

Q3: What are the types of priority inversion that a task might undergo on account of lower priority task under PCP?

Q4. How are deadlocks and unbounded priority inversion prevented using PCP?

* 1. In-Lab Experimental Procedure

1.4.1 Starting the simulator

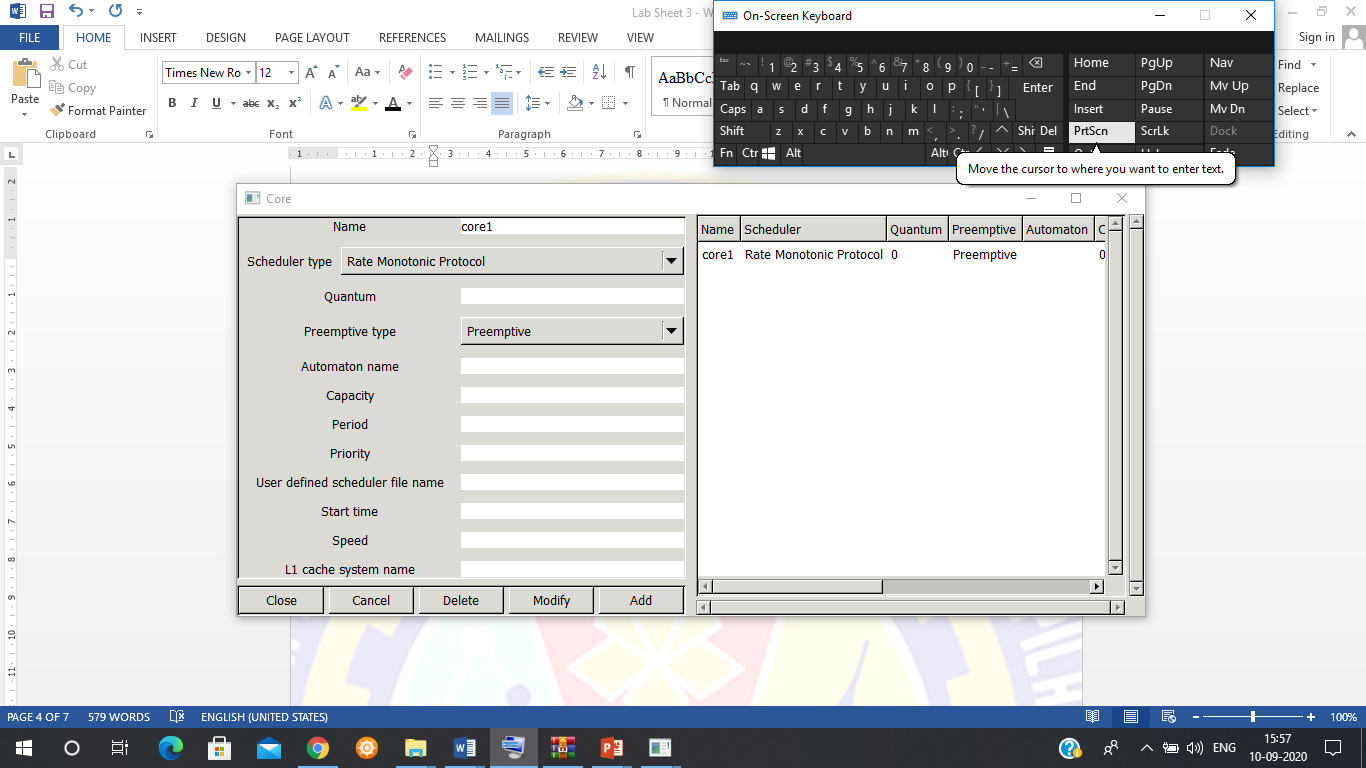
Double click on the cheddar.exe from the ‘Cheddar-3.0-win32-bin’ folder. This will

launch the application.

## 1.4.2 Building a new project:

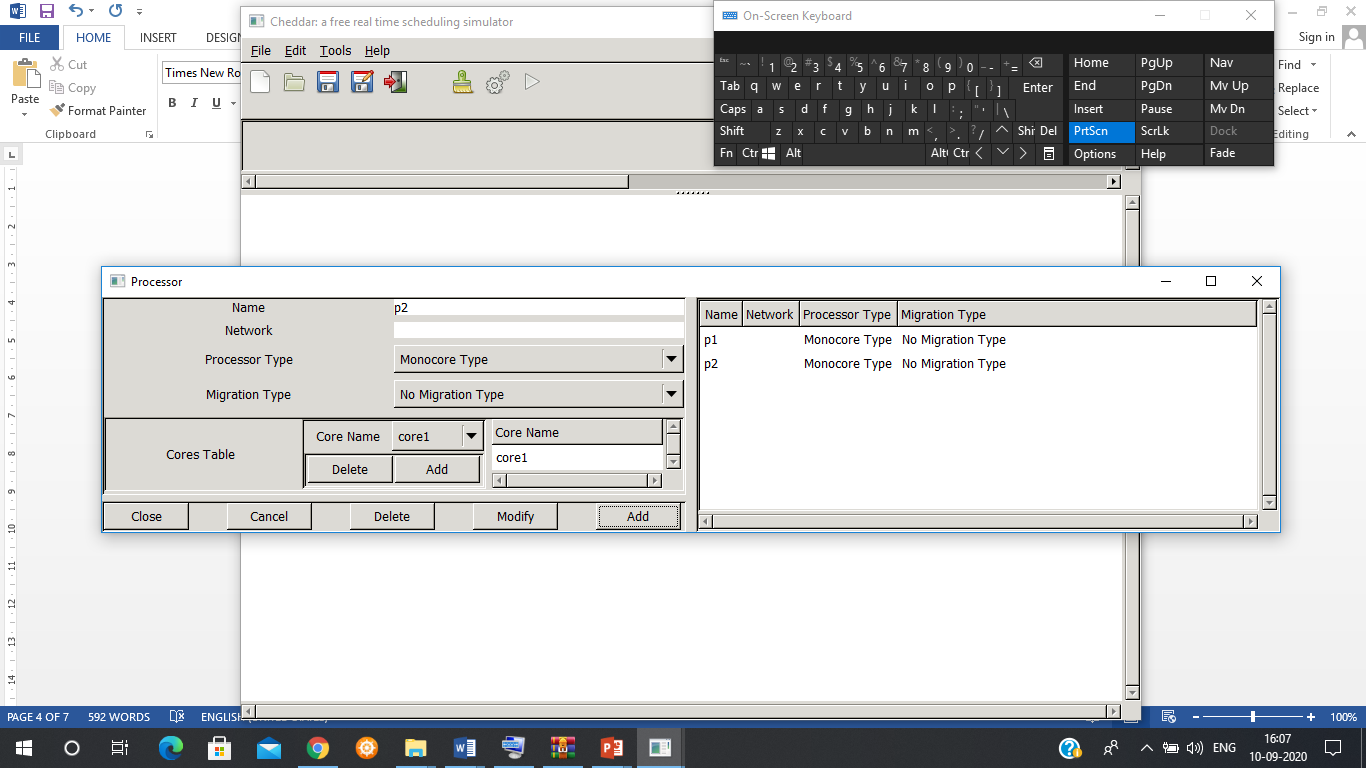
1. From the file icon in the menu bar, choose ‘new’ XML project. All newly created files have to be saved in the current path with an extension of .sc.
2. **Adding a core:**

* From the menu bar, Edit -> hardware -> Core.



**Adding the Processor:**

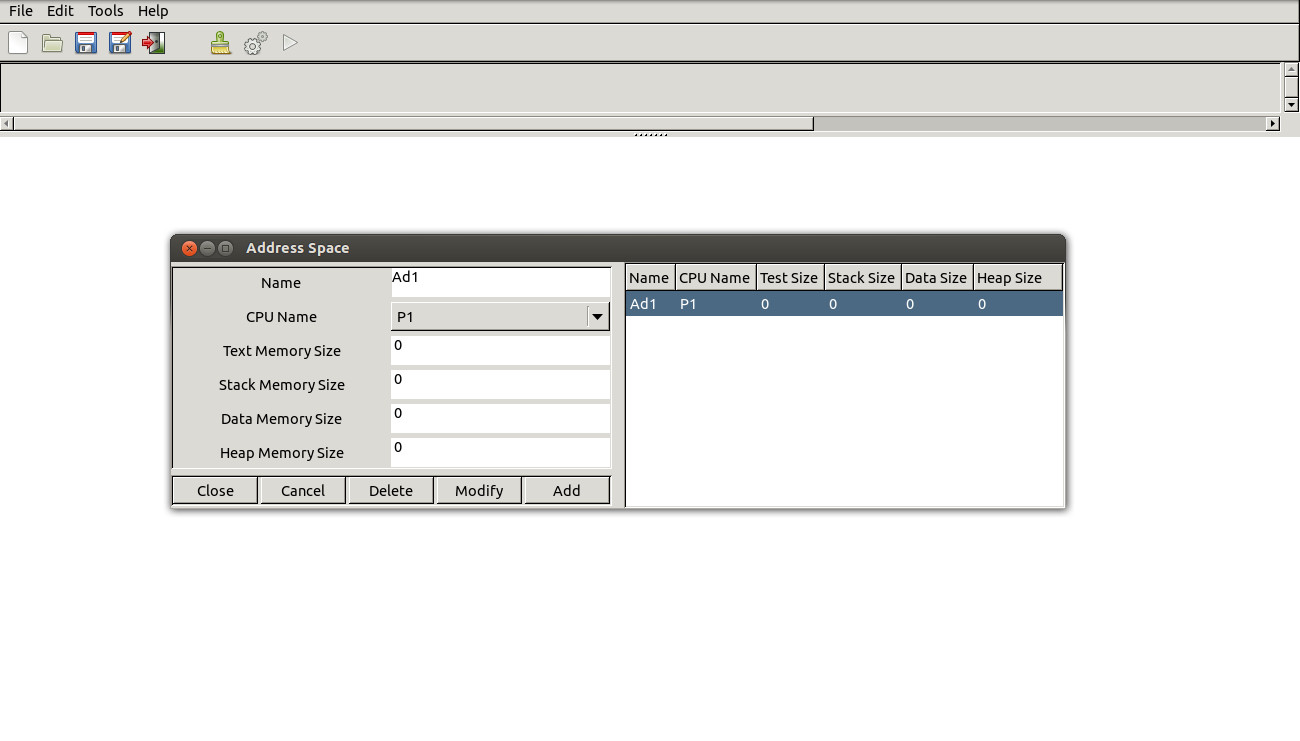
* From the menu bar, Edit -> Hardware -> Processor.



* Add the name of the processor.
* Add the core from the ‘cores table’.
* The Processor name along with the other parameters will be reflected on the right hand side.

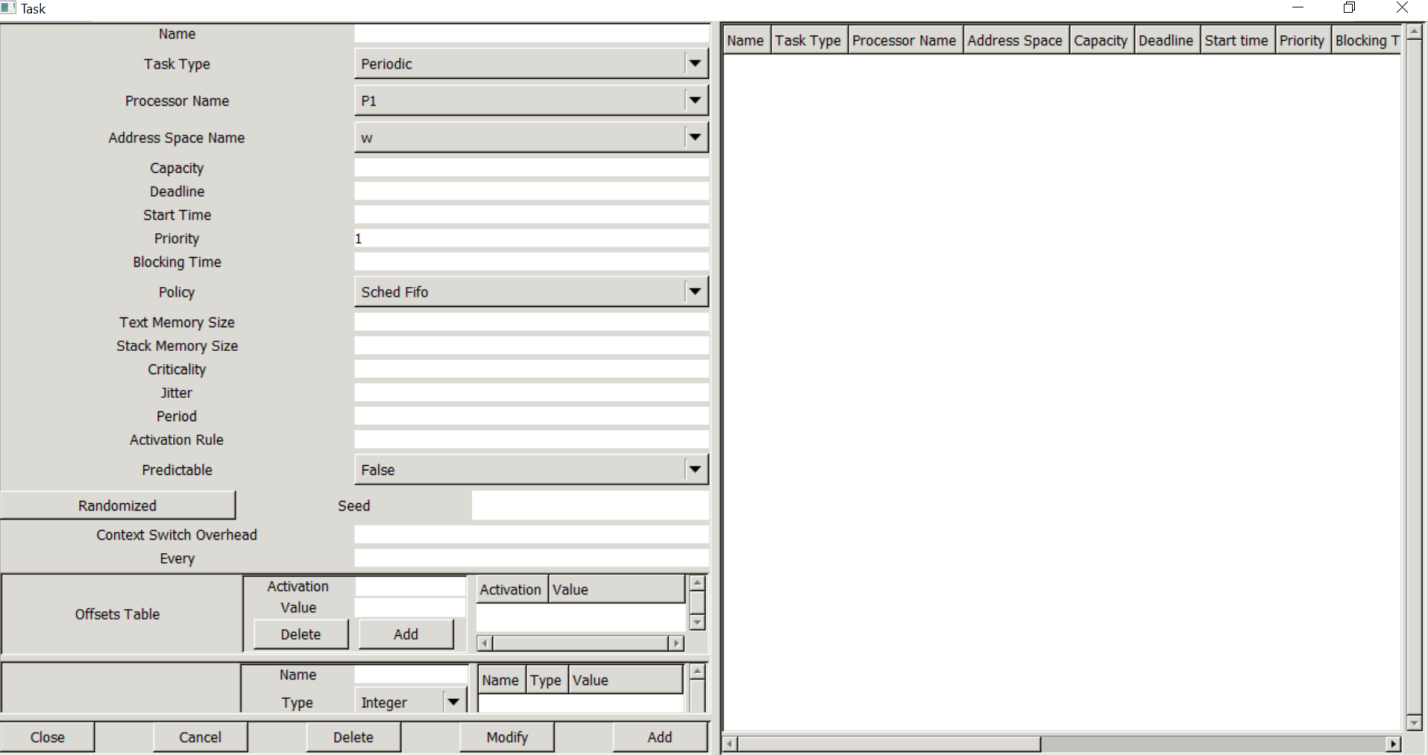
**Adding the address space:**

From the menu bar, Edit -> software -> Address space.



**Adding a task:**

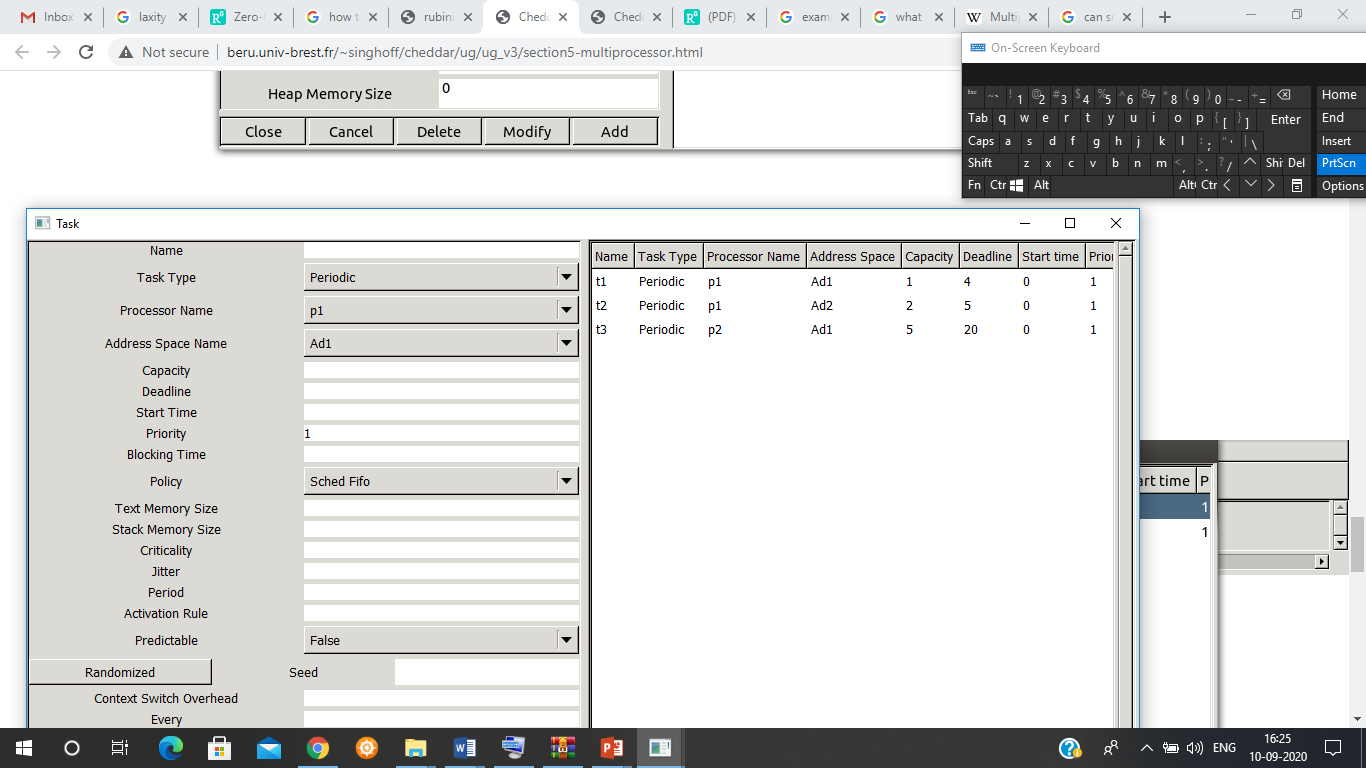
* From the menu bar, Edit -> software -> task.



* Add the name of the task.
* Task type
* Processor name: Drop down list contains all the processors defined. Choose the appropriate processor for the task.
* Capacity: Worst case Execution time of the task.
* Deadline: Absolute deadline. Refers to the completion time for the task.
* Start time: The time offset for the task, that is, the time at which the task will be ready for execution.

Priority: To define the priority of the task. When 2 or more tasks have the same priority, the scheduler will execute the task depending upon the scheduling policy.

* Period: The periodicity of the task, that is, the time after which the task will reoccur. (note: Deadline must be lesser than the periodicity). Do not specify periodicity for aperiodic and sporadic task.



We consider three periodic tasks, synchronous and with deadlines on request : tasks T1, T2 and T3. They are defined as follow:

Task T1 : Period=Deadline=6, Capacity=2, Start time = 0

Task T2 : Period=Deadline=8, Capacity=2, Start time = 0

Task T3 : Period=Deadline=12, Capacity=5, Start time = 0

We assume a preemptive fixed priority scheduling policy and we apply Rate Monotonic to assign priorities.

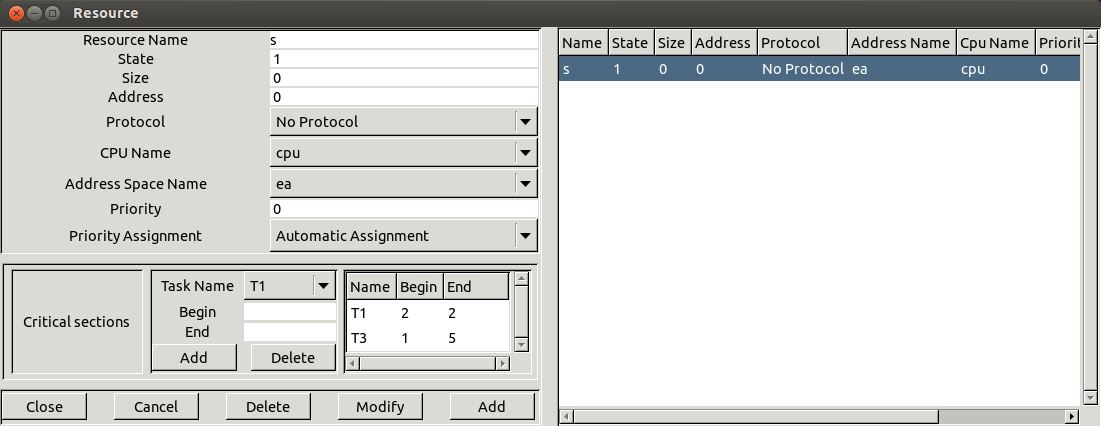
We also assume that tasks T1 and T3 share a resource named S. T1 and T3 access to S in mutual exclusion:

T3 needs S during all its capacity.

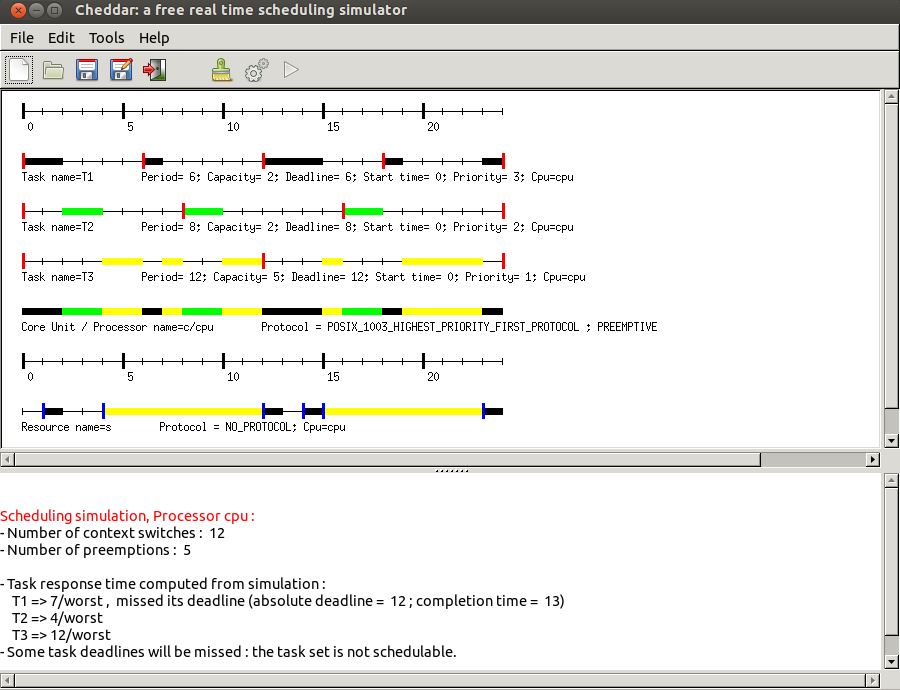
T1 needs S during the 2nd unit of time of its capacity only.

Defining the resource:

To define S, we need the Edit/Software/Resource menu which displays the following window



* Name : the unique name of the shared resource.
* CPU Name and Address space name : give where the resource is located.
* State : is the status of the resource. A Cheddar ADL resource may be seen as a counting semaphore. A Cheddar ADL resource has an integer value. When a task allocates a resource, its counter is decremented. When a task releases a resource, its counter is incremented. When this counter is equal of less than 0, a task which requests the resource is blocked as the resource is already allocated. For our exercises, State must be set to 1.
* Protocol : is the protocol to apply when a resource is handled by a task. The value No Protocol means that the simulator will apply no priority inheritance and that the blocked tasks on the semaphore are stored in a FIFO queue.
* Priority assignment : specify if the ceiling priority of the resource component will be computed automatically by Cheddar.
* Critical sections : in this part, we specify when each task needs the shared resources. Begin is the start time of each critical section and End is the critical section completion time. The buttons Add and Delete allow the designer to create or remove a critical section for a given resource.
* Compute the scheduling simulation on the hyper-period
  1. Outcomes



Change S protocol by Priority Inheritance Protocol (or PIP), and compute again the scheduling simulation. With PIP, a task which blocks another with a higher priority, runs its critical section with the blocked task priority.

Appendix A

***Experiment 4***

**In-Lab Report**

Consider two periodic tasks with the following parameters :

Task T1 : Period=Deadline=31, Capacity=8, Start time = 0

Task T2 : Period=Deadline=30, Capacity=8, Start time = 2

Prioritie are assigned according to Rate Monotonic.

T1 and T2 require the access to the shared resources R1 and R2 as follows:

T1 needs R1 from the 2nd unit of time of its capacity upto the 8th (included).

T2 needs R1 from the 6th unit of time of its capacity upto the 8th (included).

T1 needs R2 from the 4th unit of time of its capacity upto the 8th (included).

T2 needs R2 from the 2nd unit of time of its capacity upto the 8th (included).

Assume that both R1 and R2 apply PIP (Priority Inheritance Protocol).

Edit a Cheddar model for such software architecture and compute the scheduling simulation during the 30th first units of time (button ).

Compute again the scheduling simulation during the 30th units of time with the PCP resources.

* Show in the time lines when T1 and T2 lock and unlock R1 and R2.
* Say when the priorities of the tasks change due to PCP.
* Compare those results with the PIP Protocol.