

CHEDDAR 3.0

AADL Inspector

Plugin Manual

Ellidiss Technologies
<http://www.ellidiss.fr>
aadl@ellidiss.fr

1 Cheddar

Cheddar is a free real time scheduling tool. Cheddar is designed for checking task temporal constraints of a real time application/system. Systems to analyse can be described with [AADL](#) or with the Cheddar architecture design language. It can help you for quick prototyping of real time schedulers. It can also be used for educational purpose. Cheddar is developed and maintained by a team composed of the [LISyC laboratory/Université de Bretagne Occidentale](#) and [Ellidiss Technologies](#).

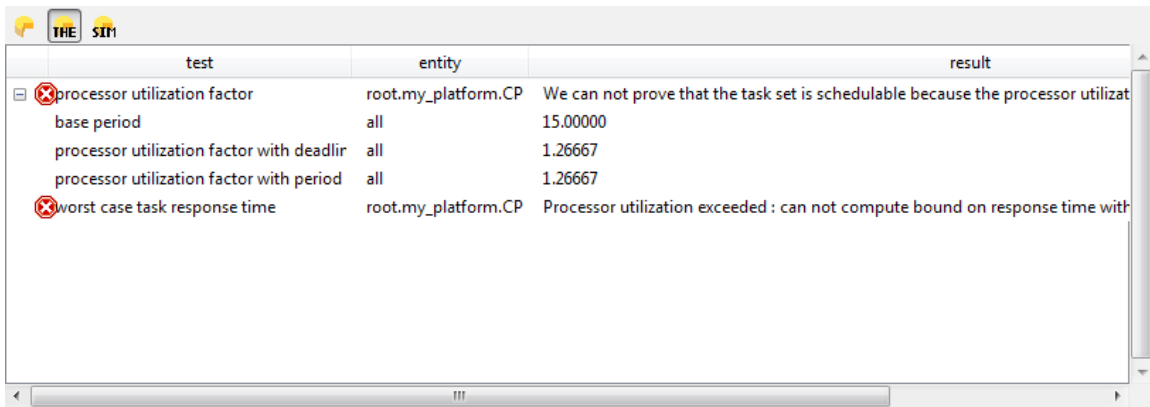
Cheddar provides two kind of features: a simulation engine (static simulation) and feasibility tests. Feasibility tests allow a user to study a real time application/system without computing a scheduling. In the contrary, the simulation engine can be used firstly to computing a scheduling and secondly, to automatically looking for task constraint properties in the computed scheduling. Most of the time, feasibility tests are less complex tools but they are available only for few schedulers and tasks models. To solve this problem, the Cheddar simulation engine provides tools to design specific schedulers and task models: the system is then analysed according to a given scheduling.

2 Schedulability Plugin

The schedulability plugin use Cheddar facilities in order to analyse temporal constraints on AADL files. Real time system simulation computes by cheddar is displayed in a time line widget, whereas theoretical and simulation results are displayed in tables. The table has the following columns :

- The test column gives the cheddar service name.
- The entity column give the name of the entity concerns by the service (thread, processor, ...).
- The result column contains the result of the service :
 - Numerical results for processor utilization factor, response time, ...
 - Textual explanation for schedulability test.

2.1 Theoretical Results



test	entity	result
<input checked="" type="checkbox"/> processor utilization factor	root.my_platform.CP	We can not prove that the task set is schedulable because the processor utilizat
base period	all	15.00000
processor utilization factor with deadlir	all	1.26667
processor utilization factor with period	all	1.26667
<input checked="" type="checkbox"/> worst case task response time	root.my_platform.CP	Processor utilization exceeded : can not compute bound on response time with

2.1.1 Theoretical Results based on Processor Utilization Factor

The theoretical results based on processor utilization factor are :

- Base period or Hyperperiod : cheddar computes, if possible, the period of time where the scheduling repeats itself.
- Processor utilization with period : it is the fraction of processor time spent in the

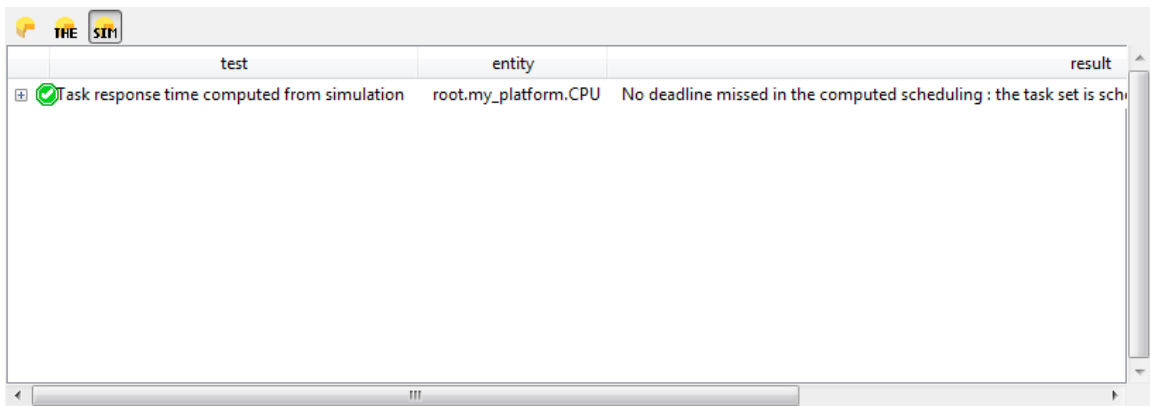
execution of the task set.

- Processor utilization with deadline : idem.

2.1.2 Theoretical Results based on Worst case response time

Cheddar computes for each tasks the worst case response time. The response time is the time elapsed between the dispatch (time when task is ready to execute) to the time when it finishes its job.

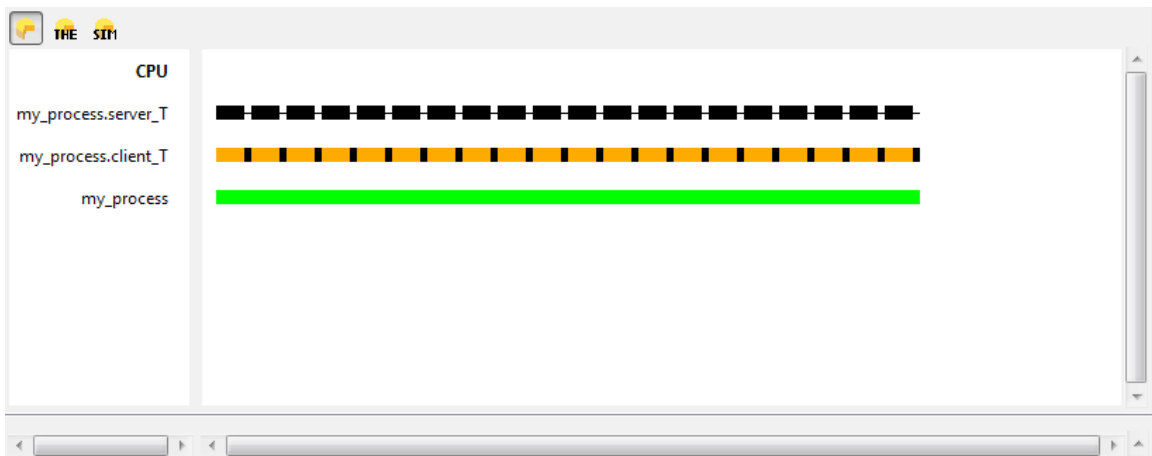
2.2 Simulation Results



Cheddar can simulate a scheduling and deduce a number of results :

- Number of thread preemptions : a thread of higher priority is executed, interrupting a thread of lower priority.
- Number of thread context switches : represent the number of thread execution switching.
- Response time for each threads.

2.3 Real time system Simulation

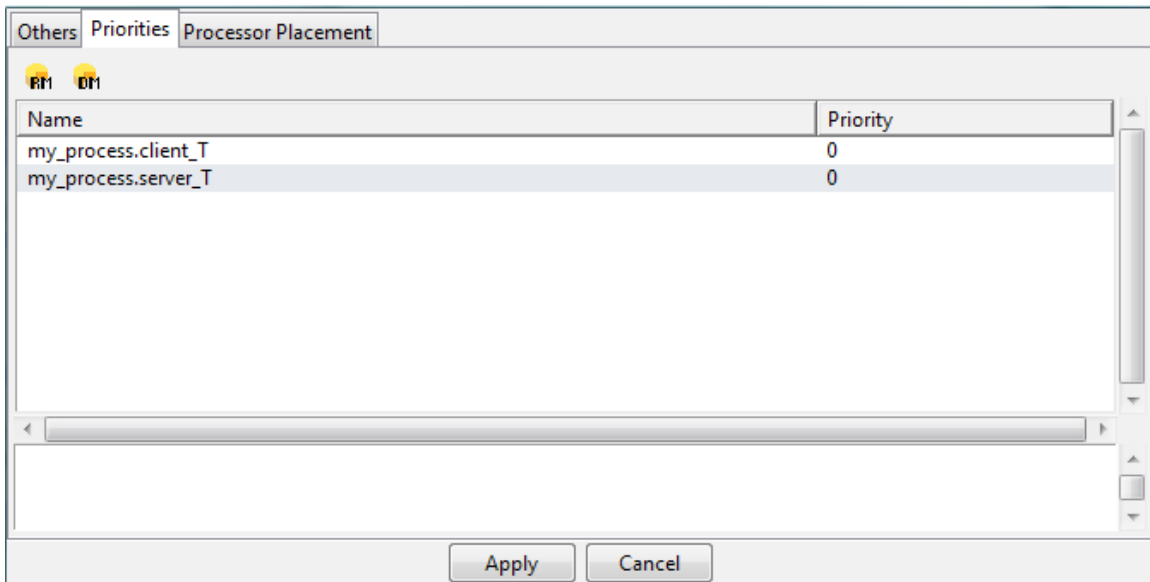


3 Real time Properties edition

For the two following features, a new proxy package is created. This package is used in order to apply the wanted modification(s) (see the “select root” and/or “Edit thread properties” paragraph of the user manual).

3.1 *Priorities Computation*

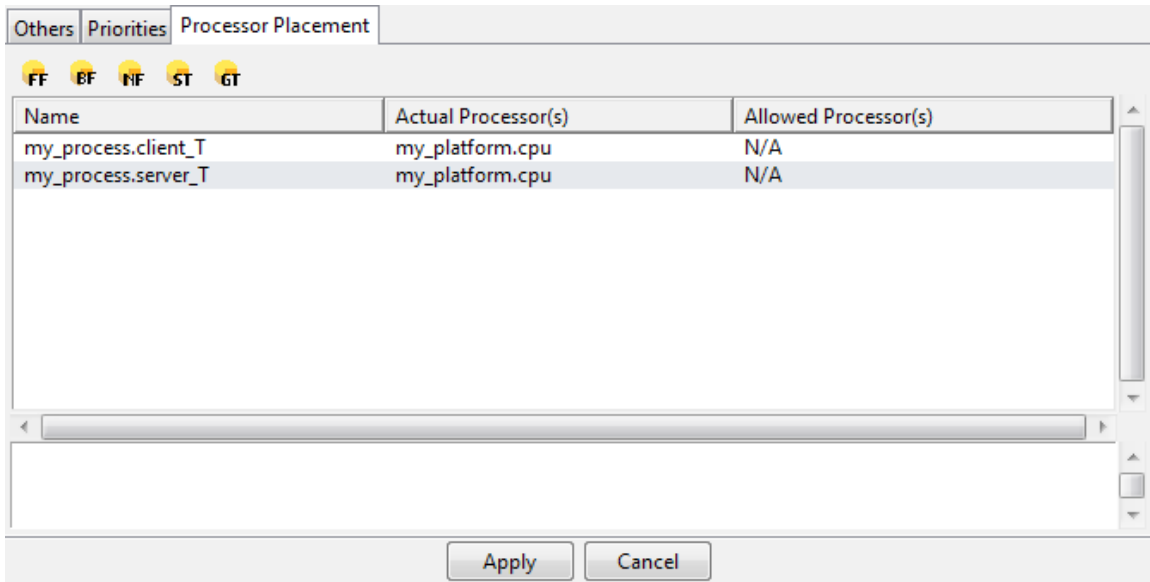
Cheddar can compute thread priority according to two scheduling protocols : Rate Monotonic and Deadline Monotonic.



Note : When the user push one of the button, the priorities are updated in the table according to cheddar results. The newly computed priorities will be assigned to the thread only if the apply button is pushed.

3.2 *Processor Placement*

Cheddar can compute thread placement on multi-processor systems.



Note : When the user push one of the button, the thread's actual processors are updated in the table according to cheddar results. The newly computed placement will be assigned to the thread only if the apply button is pushed.

The following placement algorithm are available :

3.2.1 Next-Fit

Tasks are sorted in decreasing order of their period before the assignment is started. Tasks are assigned to a so-called current processor until the schedulability condition is violated, in which case the current processor is marked full and a new processor is selected.

3.2.2 First-Fit

First-Fit is similar to *Next-Fit* algorithm but it first tries to accommodate a task in a processor marked as full before assigning it to the current processor.

3.2.3 Best-Fit

Similar to *First-Fit*, *Best-Fit* attempts to assign tasks to processor that have been marked as full. However, in *Best-Fit*, the full processors are inspected in a specific order.

3.2.4 Small-Task

Small-Task considers task set with a small computation demand (load factor) compared to the processing power of the processors. This algorithm is similar to the previous ones but proposes a new schedulability condition.

3.2.5 *General-Task*

General-Task is applicable to unrestricted task sets. The task set is partition into two groups, such that the load factor of tasks in the first is inferior to $1/3$ and the second one is superior to $1/3$. Tasks in the first group are assigned to processors with the *Small-Task* algorithm and the second one with the *First-Fit* algorithm.

4 Technical Aspects

4.1 Input AADL Supported Properties

The following properties are supported in AADLInspector Cheddar services :

4.1.1 Processor supported properties

Supported Processor Properties	
Scheduling_Protocol	This property specifies the processor scheduling protocol. See “Supported Scheduling protocols” table.
Preemptive_Scheduler	This property specifies if a thread can be preempted during its execution time by a scheduler in order to run an other thread. Can be True or False.
Scheduler_Quantum	This property specifies the quantum of a given processor. The quantum is a maximum bound on the delay a thread can held the processor without being preempted. A quantum is typically used in time sharing scheduling and in POSIX 1003.1b scheduling (with the SCHED_RR policy for example).

Specific Cheddar properties (the property set name is Cheddar_Properties) :

Supported Processor Cheddar Properties	
Source_Text	Name of the parametric scheduler file (i.e. “rm.sc”). Used when the scheduling protocol is Pipeline_User_Defined_Protocol.

Supported Scheduling Protocols	
Rate_Monotonic_Protocol (RM, RMA or RMS)	Threads have to be periodic, and their deadline must be equal to their period. Threads are scheduled according to their Period property (the shorter the period is, the higher is the thread's priority).
Deadline_Monotonic_Protocol (DM)	Threads have to be periodic and are scheduled according to their Deadline property (the shorter the deadline is, the higher is the thread's priority).
Earliest_Deadline_First_Protocol	Threads can be periodic or not and are

(EDF)	scheduled according to their deadline which is computed from the Deadline, the Period and the First_Dispatch_Time properties (the highest priority thread is the one closest to its deadline).
Least_Laxity_First_Protocol (LLF)	Threads can be periodic or not and are scheduled according to their laxity which is computed from the Deadline, the Period and the First_Dispatch_Time properties (the highest priority thread is the one with the lowest laxity).
POSIX_1003_Highest_Priority_First_Protocol (HPF) and queueing policies (SCHED_FIFO, SCHED_RR and SCHED_OTHERS)	Threads can be periodic or not. Threads are scheduled according to the priority and the policy of the threads. This scheduling protocol shares the processor according to the Priority, the POSIX_Scheduling_Policy and the First_Dispatch_Time properties.
Pipeline_User_Defined_Protocol	This scheduler allow users to define their own scheduler into Cheddar. A “*.sc” file, which contained the scheduler code must be provided through the Source_Text property.

4.1.2 Thread supported properties

Supported Thread Properties	
Period	Period of time between 2 thread dispatches. Required for Periodic and Sporadic threads.
Deadline	Deadline of the thread.
Priority	Required when the scheduling protocol is POSIX_1003_Highest_Priority_First_Protocol
Compute_Execution_Time	This is the worst case execution time of the thread. Required when no behaviour annex is specified.
POSIX_Scheduling_Policy	Queueing policies for POSIX_1003_Highest_Priority_First_Protocol. See “Supported Queueing Policies” table.
Dispatch_Protocol	See “Supported Dispatch Protocols” table.
First_Dispatch_Time	Time of the first dispatch (default 0).

Specific Cheddar properties (the property set name is Cheddar_Properties) :

<i>Supported Thread Cheddar Properties</i>	
Bound_On_Data_Blocking_Time	Maximum duration a thread has to wait for the access to a protected data. Used to compute the worst case response time of a thread (theoretical).
Dispatch_Seed_Value	This property specifies the seed value which is used by the Cheddar simulator engine. To compute a scheduling with Background, Poisson or Sporadic threads, the simulator engine has to generate random dispatching time. If this property is omitted, the seed random generator is initialized to zero : it means that several simulations of an AADL specification will lead to the same computed scheduling.
Dispatch_Seed_is_Predictable	This property specifies how the seed has to be initialized. If its value is true, the seed is initialized with the same value for all scheduling simulation. In this case, the successive simulations of an AADL specification lead to the same computed scheduling. If you plan to run several simulations of an AADL specification and if you plan to generate different scheduling for each simulation, then, this property has to be initialized with false. If this property is omitted, the seed is randomly initialized.

<i>Supported Dispatch Protocols</i>	
Periodic	The thread periodically runs a job. A periodic thread has a start time (First_Dispatch_Time). Its period (see the Period property) stores the fixed delay between two successive thread wakeup times.
Aperiodic (or Background)	The thread is dispatched at a time specified by the property First_Dispatch_Time, run a job and leaves the system.
Sporadic	The thread is dispatched with a minimal inter-waking up period delay. The minimum delay is stored in the Period property.

Poisson	The thread periodically runs a job. The Period property stores the average delay between two successive threads wakeup times. The effective delay between two wakeup times is computed with an exponential random generator.
---------	--

Supported Queueing Policies	
SCHED_FIFO	Implements a FIFO scheduling protocol : a thread stays on the processor until it has terminated or until a highest priority thread is released.
SCHED_RR	Is similar to SCHED_FIFO except that the quantum is used. At the end of the quantum, the running thread leaves the processor.
SCHED_OTHERS	implements a time sharing scheduling. A thread with a SCHED_OTHERS policy must have a priority equal to zero.

Note : a « Behavior_Specification » can be added in order to described the resource(s) access(es).

4.1.3 Data supported properties

Supported Data Properties	
Concurrency_Control_Protocol	Specifies the data access protocol. See “Supported Shared Resource Protocols” table.

Supported Shared Resource Protocols	
Priority Inheritance (PIP)	The Cheddar scheduling simulator increases the priority of a lower priority thread which blocks a higher priority thread. The Concurrency_Control_Protocol is set to Priority_Inheritance.
Priority Ceiling (PCP)	This protocol applies the priority inheritance of Priority Inheritance (PIP) and introduces a new thread blocking condition in order to avoid deadlock and to decrease thread blocking time. The Concurrency_Control_Protocol is set to Priority_Ceiling.
Immediate Priority Ceiling	This last protocol is similar to Priority Ceiling.

	The Concurrency_Control_Protocol is set to Immediate_Priority_Ceiling_Protocol.
None	No particular protocol is used when a data is accessed by threads. The Concurrency_Control_Protocol is set to None_Specified.

4.1.4 Cheddar Dependencies

A dependency models an interaction between two software entities which has an impact upon the scheduling of the system. Entities handled in a dependency component can either be threads, resources and/or buffers.

Type of Dependencies	
Precedence_Dependency	<p>This dependency models a precedence relationship between two threads.</p> <p>With this dependency, the sink thread must wait for completion of the source thread before being released by the scheduler.</p>
Queuing_Buffer_Dependency	<p>This dependency models a producer/consumer relationship between threads and a buffer.</p> <p>When the source is a thread and the sink is a buffer, the dependency models the producer side.</p> <p>When the sink is a thread and the source is a buffer, the dependency models the consumer side.</p> <p>This dependency assume a fixed buffer size. A message can be read one time only and we assume that no message can be lost.</p>
Communication_Dependency	<p>This dependency models the asynchronous exchange of a message between threads.</p> <p>When the source is a thread and the sink is a buffer, the dependency models the emitter side.</p> <p>When the sink is a thread and the source is a buffer, the dependency models the receiver side.</p>

Time_Triggered_Communication_Dependency	<p>This dependency models a synchronous communication between two threads. Three protocols exist :</p> <ul style="list-style-type: none"> • Sampled_Timing : emitters and receivers send/receive synchronous messages at their own rate. From a scheduling point of view, those threads are independent. • Immediate_Timing : when an emitter completes its activation, the receiver thread is immediately released. From a scheduling point of view, this dependency is equivalent to a precedence_dependency. • Delayed_Timing : when an emitter completes its activation, the protocol enforces that the receiver will be released on one of the next activation of the emitter task. This delay is expressed by the offset of the sink thread. From a scheduling point of view, those threads are independent as we only change thread release times.
Resource_Dependency	<p>This dependency models a shared memory access between several thread and a resource.</p>
Black_Board_Buffer_Dependency	<p>This dependency models a producer/consumer relationship between threads and a buffer.</p> <p>When the source is a thread and the sink is a buffer, the dependency models the producer side.</p> <p>When the sink is a thread and the source is a buffer, the dependency models the consumer side.</p> <p>This dependency assume a buffer size of 1. A message can be read several times or never, e.g. consumers read the last written message produced</p>

by producers.

From a scheduling point of view, tasks accessing a black board are independent.

4.1.5 Cheddar Framework Configuration

The cheddar framework configuration contains restrictions on the maximum number of items:

Cheddar	AADL	Maximum	Comment
Processor	Processor	20	
Core Unit		20	
Task	Thread	200	
Message		5	
Buffer	Data	30	
Tasks Per Buffer		10	Maximum number of tasks that can use a shared buffer. Increasing this constant implies increasing cheddar memory footprint.
Ressource	Data	30	
Tasks Per Resource		10	Maximum number of tasks that can use a shared resource. Increasing this constant implies increasing cheddar memory footprint.
Address Space	Partition	25	
Dependency		150	Maximum number of dependencies of a XML Cheddar project file. Increasing this constant implies increasing cheddar memory footprint.
Offset		5	
Scheduling Period		200000	Maximum scheduling period Cheddar can compute.

4.2 XML Output

The schedulability plugin call Cheddar services. The result is an XML file which is displayed in AADLInspector graphical interface.

The output xml file (« cheddarResult.xml » in the AADLInspector temporary directory) contains services results.

All results are contained between the parent **<results>** ... **</results>** tags.

Fore each feasibility test, computation results are contained in **<feasibilityTest>** ... **</feasibilityTest>** tags. This tag has the following attributes :

name	The name of the feasibility test : <ul style="list-style-type: none">• processor utilization factor,• worst case task response time,• Task response time computed from simulation,• Set priorities according to Rate Monotonic,• Set priorities according to Deadline Monotonic.
reference	The name of the entity concerned by the feasibility test.

Inside a feasibility test tags, one can find several sub tags :

- **<computation>** with the following attributes :

name	Name of the computation for processor utilization factor test : <ul style="list-style-type: none">• base period,• processor utilization factor with deadline,• processor utilization factor with period. Name of the computation for worst case task response time test : <ul style="list-style-type: none">• response time. Name of the computation for Task response time computed from simulation test : <ul style="list-style-type: none">• Number of preemptions,• Number of context switches,• Task response time computed from simulation. Name of the computation for Set priorities according to Rate Monotonic and Set priorities according to Deadline Monotonic
------	---

	tests : <ul style="list-style-type: none"> Updated priorities.
reference	Name of the concerned entity.
value	Computation results.
worst best average	Specific values for Task response time computed from simulation computation. Replace the value attribute.
unused	Specific for base period computation. Unused unit of time of the scheduling during the base period.
biblio	Bibliographic reference(s).

- **<schedulability>** which describes the test schedulability result. It has the following attributes :

scheduler	The scheduler concerned (See “Supported Scheduling protocols” table).
preemptive	Can be True or False. The scheduler is preemptive or not.
result	Can be True or False. The scheduling is valid or not.
explanation	A textual explanation of the results.

An example can be found below.

```

<results>
  <feasibilityTest name="processor utilization factor" reference="hpf">
    <computation name="base period" reference="all" value=210.00000
biblio="[18], page 5" unused=170.00000/>
    <computation name="processor utilization factor with deadline"
reference="all" value=0.19048 biblio="[1], page 6"/>
    <computation name="processor utilization factor with period"
reference="all" value=0.19048 biblio="[1], page 6"/>
    <schedulability
scheduler="POSIX_1003_HIGHEST_PRIORITY_FIRST_PROTOCOL"
preemptive="true" result="false" explanation="Invalid scheduler : can
not apply the feasibility test on processor utilization factor."
biblio=""/>
  </feasibilityTest>
  <feasibilityTest name="worst case task response time"
reference="hpf">

```

```

    <computation name="response time" reference="ea_hpf.tq1"
value=20.000000 biblio=""/>
    <computation name="response time" reference="ea_hpf.tq2"
value=20.000000 biblio=""/>
    <schedulability
scheduler="POSIX_1003_HIGHEST_PRIORITY_FIRST_PROTOCOL"
preemptive="true" result="true" explanation="All task deadlines will be
met : the task set is schedulable." biblio=""/>
    </feasibilityTest>
    <feasibilityTest name="Task response time computed from simulation"
reference="hpf">
        <computation name="Number of preemptions" reference="hpf" value=18
biblio=""/>
        <computation name="Number of context switches" reference="hpf"
value=19 biblio=""/>
        <computation name="Task response time computed from simulation"
reference="ea_hpf.tq1" worst=38 best=38 average=38.000000 biblio=""/>
        <computation name="Task response time computed from simulation"
reference="ea_hpf.tq2" worst=40 best=40 average=40.000000 biblio=""/>
        <schedulability
scheduler="POSIX_1003_HIGHEST_PRIORITY_FIRST_PROTOCOL"
preemptive="true" result="true" explanation="No deadline missed in the
computed scheduling : the task set seems to be schedulable."
biblio=""/>
        </feasibilityTest>
        <feasibilityTest name="Set priorities according to Rate Monotonic"
reference="hpf">
            <computation name="Updated priorities" reference="ea_hpf.tq1"
value=1 biblio=""/>
            <computation name="Updated priorities" reference="ea_hpf.tq2"
value=2 biblio=""/>
        </feasibilityTest>
        <feasibilityTest name="Set priorities according to Deadline
Monotonic" reference="hpf">
            <computation name="Updated priorities" reference="ea_hpf.tq1"
value=1 biblio=""/>
            <computation name="Updated priorities" reference="ea_hpf.tq2"
value=2 biblio=""/>
        </feasibilityTest>
    </results>

```



www.ellidiss.com

Sales office:

TNI Europe Limited
Triad House
Mountbatten Court
Worall Street
Congleton
Cheshire
CW12 1AG
UK
info@ellidiss.com
+44 1260 291 44

Technical support :

Ellidiss Technologies
24 quai de la douane
29200 Brest
Brittany
France

aadl@ellidiss.fr
+33 298 451 870