Language Framework for Optimal Schedulers (LFOS)

Guideline for Users

Güner Orhan

December 19, 2016

1 Required Modules

In order to use LFOS frameowrk API, a programmer should import the required module:

```
1 from LFOS.Scheduler.Scheduler import Scheduler
2 from LFOS.Resource.Resource import *
3 from LFOS.Task.Task import *
4 from LFOS.Scheduling.Characteristic.Time import Time
5 from LFOS.macros import *
```

Listing 1: Importing required modules

2 Scheduler

Based on the selected instance, the scheduler instance, namely "sched", can be generated by following the following procedures one-by-one:

2.1 Resource Initialization for "cpu1"

For *abstraction*, ACTIVE is selected. Therefore, you can create the type object using the following code segment:

```
1 cpu1_t = Type(ResourceTypeList.ACTIVE, 'cpu1_t')
```

Listing 2: Active resource type object instantiation

According to the specification, a programmer can create the resource giving type object and a name of the resource as arguments to the class method of the ResourceFactory class shown in Listing 3. For active resources, an object belonging to TerminalResource class is instantiated using factory method pattern to handle the optional feature under *Abstraction* sub-feature.

```
cpu1 = ResourceFactory.create_instance(cpu1_t, 'cpu1')
```

Listing 3: Active resource instantiation using ResourceFactory class

2.1.1 Setting mode

There are three possible types for this ttribute. These are:

- ModeTypeList.SHARED
- ModeTypeList.CB_EXCLUSIVE
- ModeTypeList.SB_EXCLUSIVE
- ModeTypeList.CB_AND_SB_EXCLUSIVE

The functionality of these modes are discussed in the article.

As shown in Table **??**, the mode is initially set to ModeTypeList.CB_EXCLUSIVE. A programmer can change the mode of a resource by using the following code segment:

```
cpu1.set_mode(mode)
# mode --> ResourceTypeList::Enum
```

Listing 4: Setting the mode of a resource after creating a resource.

In order to check the mode of the resource, you can use the functions depicted in Listing 5.

```
cpu1.is_mode(mode) # mode --> ResourceTypeList::Enum
# returns True if the argument matches with the mode of the resource.

cpu1.is_exclusive()
# returns True if the mode of the resource is any one of the exclusive mode
    .
```

Listing 5: The functions for resource mode check.

According to your specification, you have selected at least CB_EXCLUSIVE mode for your resource "cpu1". Since the resource is set to this mode initially, you do not need to set it again.

In addition to the CB_EXCLUSIVE mode, you have selected the SB_EXCLUSIVE mode for your resource "cpu1". Therefore, you should manually set it after resource creation using the following code segment:

```
cpu1.set_mode(ModeTypeList.CB_AND_SB_EXCLUSIVE)
```

Listing 6: The resource is set to CB_AND_SB_EXCLUSIVE mode.

Due to semantic-based exclusive property of the resource, you can define exclusive resources by using the following formula:

```
cpu1.add_exclusive_resource(resource)
# returns True if the SB\_EXCLUSIVE mode is selected and the resource
    argument is not in the list of exclusive resources. Otherwise, it
    returns False.
```

Listing 7: A function for adding exclusive resources.

2.2 Resource Initialization for "cpu2"

For *abstraction*, ACTIVE is selected. Therefore, you can create the type object using the following code segment:

```
cpu2_t = Type(ResourceTypeList.ACTIVE, 'cpu2_t')
```

Listing 8: Active resource type object instantiation

According to the specification, a programmer can create the resource giving type object and a name of the resource as arguments to the class method of the ResourceFactory class shown in Listing 9. For active resources, an object belonging to TerminalResource class is instantiated using factory method pattern to handle the optional feature under *Abstraction* sub-feature.

```
cpu2 = ResourceFactory.create_instance(cpu2_t, 'cpu2')
```

Listing 9: Active resource instantiation using ResourceFactory class

2.2.1 Setting mode

There are three possible types for this ttribute. These are:

- ModeTypeList.SHARED
- ModeTypeList.CB_EXCLUSIVE
- ModeTypeList.SB_EXCLUSIVE
- ModeTypeList.CB_AND_SB_EXCLUSIVE

The functionality of these modes are discussed in the article.

As shown in Table **??**, the mode is initially set to ModeTypeList.CB_EXCLUSIVE. A programmer can change the mode of a resource by using the following code segment:

```
cpu2.set_mode(mode)
# mode --> ResourceTypeList::Enum
```

Listing 10: Setting the mode of a resource after creating a resource.

In order to check the mode of the resource, you can use the functions depicted in Listing 11.

```
1 cpu2.is_mode(mode) # mode --> ResourceTypeList::Enum
2 # returns True if the argument matches with the mode of the resource.
3
4 cpu2.is_exclusive()
5 # returns True if the mode of the resource is any one of the exclusive mode
    .
```

Listing 11: The functions for resource mode check.

According to your specification, you have selected at least CB_EXCLUSIVE mode for your resource "cpu2". Since the resource is set to this mode initially, you do not need to set it again.