**ET ROBOCON 2014**

**MODELING STANDARDS**

*# TODO: Fill in with latest standards*

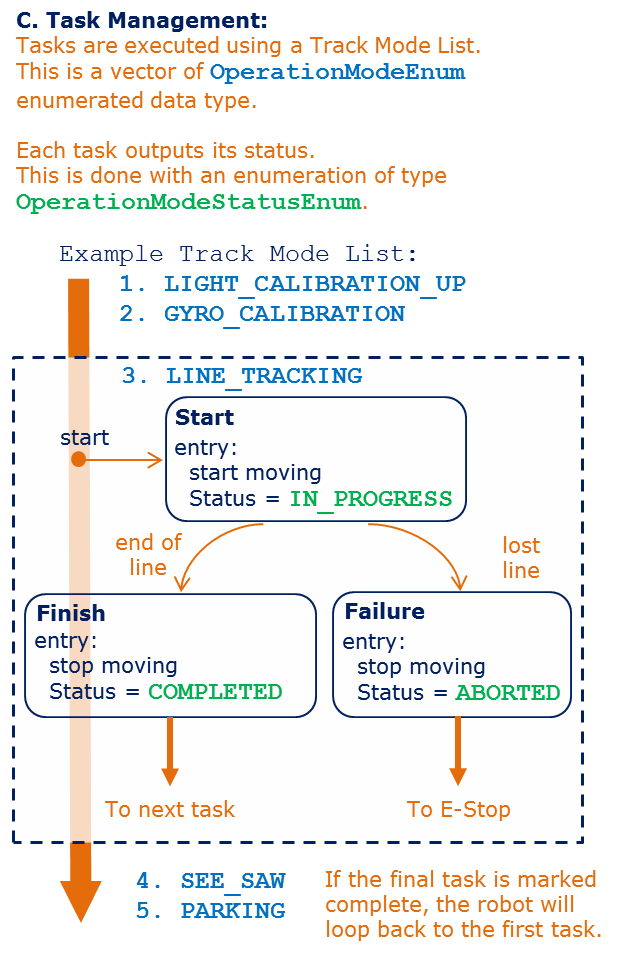
**I. SPECIFICATION FOR CREATING TASK COMPONENTS**

1. INTERFACE CONTROL  
     
   Your component must be defined as an atomic subchart. The chart data can be mapped to any subset of the inputs and outputs of the Main Controller state chart.
   1. INPUTS

|  |  |  |
| --- | --- | --- |
| **Input Name** | **Data Type** | **Description** |
| states | States\_Bus | Estimated states used for control. |
| sensors | Sensors\_Bus | Raw sensor values from hardware or simulation. |

* 1. OUTPUTS

|  |  |  |
| --- | --- | --- |
| **Output Name** | **Data Type** | **Description** |
| PWM | PWM\_bus | PWM values (0-100) for the left, right, and tail motors. |
| init\_flags | Init\_Bus | Typically zero; set values to 1 for a single time step to re-initialize state estimation integrators (encoder/gyro) to zero. |
| BT\_status\_out | single | Used for Bluetooth Communications. Will not be used for this training module. |

1. EXECUTING TASKS  
   1. TASK EXECUTION LIST  
        
      The data dictionary contains a variable named  
      **TRACK\_MODE\_LIST**. This is an array of  
      **OperationModeEnum** enumerations which  
      correspond to different atomic subcharts to be  
      executed in order.

* 1. STATUS REPORTING  
       
     The status of each task should be assigned an  
      **OperationModeStatusEnum** enumeration.   
     As long as a task is **IN\_PROGRESS**, the state chart  
     will remain in that task. When the task is completed,  
     it must broadcast its status as  
     **SUCCESSFULLY\_COMPLETED**.

* 1. CURRENT MODE STATUS  
       
     The status in part *b.* above is associated with the  
     variable **current\_mode\_status** in the Main Controller chart.   
     You must create a Data Store Memory scoped variable in your  
     component to communicate the **OperationModeStatusEnum** value.

**II. TRAINING TASK SYSTEM REQUIREMENTS**

Each task name must be defined as an **OperationModeEnum**. For example, Task 2 in Option A should be entered when the current task execution list is **OperationModeEnum.SIM\_MOTION**.

NOTE: The balancing and turning controllers for the simulation task can be found in the **SimControllerFcns** library in the **models\Simulink\_Fcn\_Library** folder. The model **plantNXT\_testharness** in the **models\plant** folder shows how you can use these functions in Simulink. Your task may involve using them in Stateflow.

OPTION A.  SIMULATION

*TASK 1:  BALANCING*

* When the program starts, the robot must balance in place for **T\_balance** seconds.
* After **T\_balance** seconds, the robot must automatically transition to Task 2.

*TASK 2:  SIM\_MOTION*

* On the first time step in this task, the wheel encoder angles must be reinitialized to zero using the **init\_flags** chart output.
* The robot must use a combination of balancing and turning controllers – the outputs of these controllers can simply be added together.
* The turning controller gains must be provided in the parameters **Kp** and **Ki**.
* The reference linear and angular speeds should be given in the parameters **V\_ref** and **omega\_ref,** respectively.

OPTION B.  HARDWARE

*TASK 1: PREPARATION*

* When the program starts, the robot must move its tail down **theta\_tail** degrees and stand still on it.
* After **T\_prep** seconds, the robot must beep with the **Speak** Simulink Function to indicate it is ready for the next task.
* The next task is unlocked only when the user presses the touch sensor.

*TASK 2: HW\_MOTION*

* The robot must move forward with the same speed on each wheel.
* After the wheels have turned **N\_rev** revolutions, the robot must turn in place with equal and opposite wheel speeds.
* The duty cycle (PWM Magnitude) in both cases should be dictated by a parameter **duty**.