Spencer West

Ryan Golden

Tyler Rose

CSCD 350

Project 5

Test Sensor B.1: Standalone Sensor

1.The rationale behind the test; i.e., what is it testing and why we care.

Answer: The purpose of this test is to have a minimal sensor object with no groups or mapper or watchdog. It is simple attached to the network and monitors that network. This is a position sensor that should be attached to an actuator to monitor the position of that actuator but the test is to examine if it is possible to attach a watchdog to the network.

2. A general English description of the initial conditions of the test.

Answer: the parser is giving the text command to create a sensor called mySensor1. The follow creation commands are given as well such as a controller called myController1 that registers the sensor to it. The controller is a must for the sensor. The last three commands given are create a network with components of mySensor1 followed by a set command to set the value of the identification of the sensor to 35 and a command to get that value.

3. The commands for (2), which must appear in a standalone form that could be directly copied into a text file to reproduce the test without manual intervention. Do not cross-reference other tests.

```
@CONFIGURE LOG \"a.txt\" DOT SEQUENCE \"b.txt\" NETWORK \"c.txt\" XML \"d.txt\"
CREATE SENSOR POSITION mySensor1
BUILD NETWORK WITH COMPONENTS mySensor1
CREATE CONTROLLER FORWARDING myController1 WITH COMPONENTS mySensor1
SET SENSOR mySensor1 VALUE 35
GET SENSOR mySensor1 VALUE
@exit
```

4. A brief English narrative of the expected results of executing the test. (Proper testing discipline expects that you do this before running the test.)

Answer: The parser will parse the sensor creation creating a sensor object of position called mySensor1. There is a network that needs to be created that the sensor will be attached to. After both the network and the sensor are created and attached the next command is to set a value to the sensor id and then retrieve it

5. At least one representation of the actual results. The form is your choice.

```
Welcome to your ParserHelper
Welcome to your Startup class
PARSE> @CONFIGURE LOG "a.txt" DOT SEQUENCE "b.txt" NETWORK "c.txt" XML "d.txt"
SCHEDULE | @CONFIGURE LOG "a.txt" DOT SEQUENCE "b.txt" NETWORK "c.txt" XML
"d.txt"
PARSE> CREATE SENSOR POSITION mySensor1
SCHEDULE | CREATE SENSOR POSITION mySensor1
PARSE> BUILD NETWORK WITH COMPONENTS mySensor1
```

```
SCHEDULE | BUILD NETWORK WITH COMPONENTS mySensor1
PARSE> CREATE CONTROLLER FORWARDING myController1 WITH COMPONENTS mySensor1
SCHEDULE | CREATE CONTROLLER FORWARDING myController1 WITH COMPONENTS mySensor1
PARSE> SET SENSOR mySensor1 VALUE 35
SCHEDULE | SET SENSOR mySensor1 VALUE 35
PARSE> GET SENSOR mySensor1 VALUE
SCHEDULE | GET SENSOR mySensor1 VALUE
PARSE> @exit
SCHEDULE | @exit
TIME     | 0.02
EXECUTE  | @CONFIGURE LOG "a.txt" DOT SEQUENCE "b.txt" NETWORK "c.txt" XML
"d.txt"
EXECUTE  | CREATE SENSOR POSITION mySensor1
EXECUTE  | BUILD NETWORK WITH COMPONENTS mySensor1
EXECUTE  | CREATE CONTROLLER FORWARDING myController1 WITH COMPONENTS mySensor1
EXECUTE  | SET SENSOR mySensor1 VALUE 35
EXECUTE  | GET SENSOR mySensor1 VALUE
The value of Identifier{name=mySensor1} is 35.0
EXECUTE  | @exit
EXITING  |
```

6. A brief discussion on how the actual results differ from the expected results.

Answer: The result of the set and get value differ as the input was an integer for set however the output is a double. The sensor does attach to the network but doesn't monitor anything but the network position. The attachment of the sensor to the network is otherwise successful.

7. A suggestion for how to extend this test to cover related aspects not required here. Your document must be formatted professionally. It must be consistent in all respects across.

Answer: The sensor could have other components such as watchdogs or mapper to apply a more strenuous test experience.

Test C.1: Passthrough Mapper

1. The rationale behind the test; i.e., what is it testing and why we care.

Answer: This is testing the creation of a passthrough mapper and assigning it to a position sensor, and then getting it's value. We care about this because it shows that the passthrough mapper creation is working and that it gets assigned to a sensor.

2. A general English description of the initial conditions of the test.

Answer: The initial conditions are that there is no mapper, no sensor, no actuators, and no reporters. The mapper must be made before the sensor, the actuators need to be made before the reporter, and the reporter needs to be made before the sensor.

3. The commands for (2), which must appear in a standalone form that could be directly copied into a text file to reproduce the test without manual intervention. Do not cross-reference other tests.

```
@CONFIGURE LOG \"a.txt\" DOT SEQUENCE \"b.txt\" NETWORK \"c.txt\" XML \"d.txt\"

CREATE MAPPER myMapper EQUATION PASSTHROUGH

CREATE SENSOR POSITION mySensor1 MAPPER myMapper

CREATE CONTROLLER FORWARDING myController1 WITH COMPONENTS mySensor1

GET SENSOR mySensor1 VALUE

SET SENSOR mySensor1 VALUE 10

GET SENSOR mySensor1 VALUE

@exit
```

4. A brief English narrative of the expected results of executing the test. (Proper testing discipline expects that you do this before running the test.)

Answer: The expected results are for the GET SENSOR command to retrieve the value of the sensor and output it to the screen once before we set the value to see what the default is and once after to see how the mapper changed it, if it did. The expected result from passthrough is that the mapper doesn't change the value at all so the value should be 0 at first, then 10 after it gets set.

5. At least one representation of the actual results. The form is your choice.

```
Welcome to your ParserHelper

Welcome to your Startup class

PARSE> @CONFIGURE LOG "a.txt" DOT SEQUENCE "b.txt" NETWORK "c.txt" XML "d.txt"

SCHEDULE | @CONFIGURE LOG "a.txt" DOT SEQUENCE "b.txt" NETWORK "c.txt" XML
"d.txt"

PARSE> CREATE MAPPER myMapper EQUATION PASSTHROUGH

SCHEDULE | CREATE MAPPER myMapper EQUATION PASSTHROUGH

PARSE> CREATE SENSOR POSITION mySensor1 MAPPER myMapper

SCHEDULE | CREATE SENSOR POSITION mySensor1 MAPPER myMapper

PARSE> CREATE CONTROLLER FORWARDING myController1 WITH COMPONENTS mySensor1

SCHEDULE | CREATE CONTROLLER FORWARDING myController1 WITH COMPONENTS mySensor1

PARSE> GET SENSOR mySensor1 VALUE

SCHEDULE | GET SENSOR mySensor1 VALUE

PARSE> SET SENSOR mySensor1 VALUE 10

SCHEDULE | SET SENSOR mySensor1 VALUE 10

PARSE> GET SENSOR mySensor1 VALUE
```

```
SCHEDULE | GET SENSOR mySensor1 VALUE

PARSE> @exit

SCHEDULE | @exit

TIME     | 0.02

EXECUTE  | @CONFIGURE LOG "a.txt" DOT SEQUENCE "b.txt" NETWORK "c.txt" XML
"d.txt"

EXECUTE  | CREATE MAPPER myMapper EQUATION PASSTHROUGH

EXECUTE  | CREATE SENSOR POSITION mySensor1 MAPPER myMapper

EXECUTE  | CREATE CONTROLLER FORWARDING myController1 WITH COMPONENTS mySensor1

EXECUTE  | GET SENSOR mySensor1 VALUE

The value of Identifier{name=mySensor1} is 0.0

EXECUTE  | SET SENSOR mySensor1 VALUE 10

EXECUTE  | GET SENSOR mySensor1 VALUE

The value of Identifier{name=mySensor1} is 10.0

EXECUTE  | @exit

EXITING  |
```

6. A brief discussion on how the actual results differ from the expected results.

Answer: The value was not affected by the mapper at all, which is expected as the passthrough mapper wasn't supposed to change the value.

7. A suggestion for how to extend this test to cover related aspects not required here

Answer: Reporter or Watchdog could be added in to further test how mapper alters the value.

Task C.2: Scaled Mapper

1. The rationale behind the test; i.e., what is it testing and why we care.

Answer: This is testing the creation of a scaled mapper and assigning it to a position sensor, and then getting it's value. We care about this because it shows that the scaled mapper creation is working and that it gets assigned to a sensor.

2. A general English description of the initial conditions of the test.

Answer: The initial conditions are that there is no mapper, no sensor, no actuators, and no reporters. The mapper must be made before the sensor, the actuators need to be made before the reporter, and the reporter needs to be made before the sensor.

3. The commands for (2), which must appear in a standalone form that could be directly copied into a text file to reproduce the test without manual intervention. Do not cross-reference other tests.

```
@CONFIGURE LOG \"a.txt\" DOT SEQUENCE \"b.txt\" NETWORK \"c.txt\" XML \"d.txt\"

CREATE MAPPER myMapper2 EQUATION SCALE 10

CREATE SENSOR POSITION mySensor2 MAPPER myMapper2

CREATE CONTROLLER FORWARDING myController2 WITH COMPONENTS mySensor2

GET SENSOR mySensor2 VALUE

SET SENSOR mySensor2 VALUE 1

GET SENSOR mySensor2 VALUE

@exit
```

4. A brief English narrative of the expected results of executing the test. (Proper testing discipline expects that you do this before running the test.)

Answer: The expected results are for the GET SENSOR command to retrieve the value of the sensor and output it to the screen once before we set the value to see what the default is and once after to see how the mapper changed it, if it did. The expected result is that the default value is 0, and then after we set the value to 1, it should be 10 as we are scaling it by 10. So the value should be multiplied by 10.

5. At least one representation of the actual results. The form is your choice.

```
Welcome to your ParserHelper

Welcome to your Startup class

PARSE> @CONFIGURE LOG "a.txt" DOT SEQUENCE "b.txt" NETWORK "c.txt" XML "d.txt"

SCHEDULE | @CONFIGURE LOG "a.txt" DOT SEQUENCE "b.txt" NETWORK "c.txt" XML
"d.txt"

PARSE> CREATE MAPPER myMapper2 EQUATION SCALE 10

SCHEDULE | CREATE MAPPER myMapper2 EQUATION SCALE 10

PARSE> CREATE SENSOR POSITION mySensor2 MAPPER myMapper2

SCHEDULE | CREATE SENSOR POSITION mySensor2 MAPPER myMapper2

PARSE> CREATE CONTROLLER FORWARDING myController2 WITH COMPONENTS mySensor2

SCHEDULE | CREATE CONTROLLER FORWARDING myController2 WITH COMPONENTS mySensor2

PARSE> GET SENSOR mySensor2 VALUE

SCHEDULE | GET SENSOR mySensor2 VALUE
```

```
PARSE> SET SENSOR mySensor2 VALUE 1

SCHEDULE | SET SENSOR mySensor2 VALUE 1

PARSE> GET SENSOR mySensor2 VALUE

SCHEDULE | GET SENSOR mySensor2 VALUE

PARSE> @exit

SCHEDULE | @exit

TIME     | 0.02

EXECUTE  | @CONFIGURE LOG "a.txt" DOT SEQUENCE "b.txt" NETWORK "c.txt" XML
"d.txt"

EXECUTE  | CREATE MAPPER myMapper2 EQUATION SCALE 10

EXECUTE  | CREATE SENSOR POSITION mySensor2 MAPPER myMapper2

EXECUTE  | CREATE CONTROLLER FORWARDING myController2 WITH COMPONENTS mySensor2

EXECUTE  | GET SENSOR mySensor2 VALUE

The value of Identifier{name=mySensor2} is 0.0

EXECUTE  | SET SENSOR mySensor2 VALUE 1

EXECUTE  | GET SENSOR mySensor2 VALUE

The value of Identifier{name=mySensor2} is 10.0

EXECUTE  | @exit

EXITING  |
```

6. A brief discussion on how the actual results differ from the expected results.

Answer: The value of Sensor was unchanged at 0 as 0*10 = 0, and when it was changed to 1 then it became 10, which was expected.

7. A suggestion for how to extend this test to cover related aspects not required here

Answer: Reporter or Watchdog could be added in to further test how mapper alters the value.

Task C.3: Normalized Mapper

1. The rationale behind the test; i.e., what is it testing and why we care.

Answer: This is testing the creation of a normalized mapper and assigning it to a position sensor, and then getting it's value. We care about this because it shows that the normalized mapper creation is working and that it gets assigned to a sensor.

2. A general English description of the initial conditions of the test.

Answer: The initial conditions are that there is no mapper, no sensor, no actuators, and no reporters. The mapper must be made before the sensor, the actuators need to be made before the reporter, and the reporter needs to be made before the sensor.

3. The commands for (2), which must appear in a standalone form that could be directly copied into a text file to reproduce the test without manual intervention. Do not cross-reference other tests.

```
@CONFIGURE LOG \"a.txt\" DOT SEQUENCE \"b.txt\" NETWORK \"c.txt\" XML \"d.txt\"

CREATE MAPPER myMapper3 EQUATION NORMALIZE 10 20

CREATE SENSOR POSITION mySensor3 MAPPER myMapper3

CREATE CONTROLLER FORWARDING myController3 WITH COMPONENTS mySensor3

GET SENSOR mySensor3 VALUE

SET SENSOR mySensor3 VALUE 15

GET SENSOR mySensor3 VALUE

@exit
```

4. A brief English narrative of the expected results of executing the test. (Proper testing discipline expects that you do this before running the test.)

Answer: The expected results are for the GET SENSOR command to retrieve the value of the sensor and output it to the screen once before we set the value to see what the default is and once after to see how the mapper changed it, if it did. The expected value at the default is 0 as 0 lies outside the bounds of 10 and 20 so it is 0. After it's set to 15 it should return a value of 50 as 50 is halfway between 10 and 20.

5. At least one representation of the actual results. The form is your choice.

```
Welcome to your ParserHelper

Welcome to your Startup class

PARSE> @CONFIGURE LOG "a.txt" DOT SEQUENCE "b.txt" NETWORK "c.txt" XML "d.txt"

SCHEDULE | @CONFIGURE LOG "a.txt" DOT SEQUENCE "b.txt" NETWORK "c.txt" XML
"d.txt"

PARSE> CREATE MAPPER myMapper3 EQUATION NORMALIZE 10 20

SCHEDULE | CREATE MAPPER myMapper3 EQUATION NORMALIZE 10 20

PARSE> CREATE SENSOR POSITION mySensor3 MAPPER myMapper3

SCHEDULE | CREATE SENSOR POSITION mySensor3 MAPPER myMapper3
```

```
PARSE> CREATE CONTROLLER FORWARDING myController3 WITH COMPONENTS mySensor3

SCHEDULE | CREATE CONTROLLER FORWARDING myController3 WITH COMPONENTS mySensor3

PARSE> GET SENSOR mySensor3 VALUE

SCHEDULE | GET SENSOR mySensor3 VALUE

PARSE> SET SENSOR mySensor3 VALUE 15

SCHEDULE | SET SENSOR mySensor3 VALUE 15

PARSE> GET SENSOR mySensor3 VALUE

SCHEDULE | GET SENSOR mySensor3 VALUE

PARSE> @exit

SCHEDULE | @exit

TIME     | 0.02

EXECUTE  | @CONFIGURE LOG "a.txt" DOT SEQUENCE "b.txt" NETWORK "c.txt" XML
"d.txt"

EXECUTE  | CREATE MAPPER myMapper3 EQUATION NORMALIZE 10 20

EXECUTE  | CREATE SENSOR POSITION mySensor3 MAPPER myMapper3

EXECUTE  | CREATE CONTROLLER FORWARDING myController3 WITH COMPONENTS mySensor3

EXECUTE  | GET SENSOR mySensor3 VALUE

The value of Identifier{name=mySensor3} is 0.0

EXECUTE  | SET SENSOR mySensor3 VALUE 15

EXECUTE  | GET SENSOR mySensor3 VALUE

The value of Identifier{name=mySensor3} is 50.0

EXECUTE  | @exit

EXITING  |
```

6. A brief discussion on how the actual results differ from the expected results.

Answer: The actual results is that at the default it remained 0 then after we set it to 15 it turned into 50.0 which was the expected value.

7. A suggestion for how to extend this test to cover related aspects not required here.

Answer: Reporter or Watchdog could be added in to further test how mapper alters the value.

Test E.1 Instantaneous Band Watchdog

1. The rationale behind the test; i.e., what is it testing and why we care.

Answer: The purpose of this the band watchdog is to create a watchdog that will measure the immediate value of the sensor whether it is a position or speed sensor. This is to check if the watchdog does read the value of the sensor and reports back if what the sensor is within range of acceptable values for the watchdog.

2. A general English description of the initial conditions of the test.

Answer: The parser takes the command to create an instantaneous band watchdog object with a high and low value call myWatchdog1. An invalid instantaneous band watchdog command will also be present called myWatchdog2 that has a low and high value beyond that of the sensor. The sensor will be created shortly afterwards with both watchdogs attached. The value for sensor will then then be set with a value of 2 which will fall within the valid case but not the invalid case. A clock will also be set to .03 seconds.

3. The commands for (2), which must appear in a standalone form that could be directly copied into a text file to reproduce the test without manual intervention. Do not cross-reference other tests.

```
@CONFIGURE LOG \"a.txt\" DOT SEQUENCE \"b.txt\" NETWORK \"c.txt\" XML \"d.txt\"
CREATE WATCHDOG BAND myWatchdog1 MODE INSTANTANEOUS THRESHOLD LOW 1 HIGH 3
CREATE WATCHDOG BAND myWatchdog2 MODE INSTANTANEOUS THRESHOLD LOW 10 HIGH 30
CREATE SENSOR POSITION mySensor1 WATCHDOGS myWatchdog1 myWatchdog2
BUILD NETWORK WITH COMPONENTS mySensor1
SET SENSOR mySensor1 VALUE 2
@CLOCK wait FOR .03
@exit
```

4. A brief English narrative of the expected results of executing the test. (Proper testing discipline expects that you do this before running the test.)

Answer: The result is that the valid command for instantaneous band watchdog will be created with its values. There is also an invalid version of a watchdog with a point of both low and high values that are above what the sensor value is. The invalid should create an error based off what the sensor value is registered at whereas the valid version should not bring up any issue.

5. At least one representation of the actual results. The form is your choice.

```
Welcome to your ParserHelper
Welcome to your Startup class
PARSE> @CONFIGURE LOG "a.txt" DOT SEQUENCE "b.txt" NETWORK "c.txt" XML "d.txt"
SCHEDULE | @CONFIGURE LOG "a.txt" DOT SEQUENCE "b.txt" NETWORK "c.txt" XML
"d.txt"
PARSE> CREATE WATCHDOG BAND myWatchdog1 MODE INSTANTANEOUS THRESHOLD LOW 1 HIGH
3
SCHEDULE | CREATE WATCHDOG BAND myWatchdog1 MODE INSTANTANEOUS THRESHOLD LOW 1
HIGH 3
PARSE> CREATE WATCHDOG BAND myWatchdog2 MODE INSTANTANEOUS THRESHOLD LOW 10
HIGH 30
SCHEDULE | CREATE WATCHDOG BAND myWatchdog2 MODE INSTANTANEOUS THRESHOLD LOW 10
HIGH 30
PARSE> CREATE SENSOR POSITION mySensor1 WATCHDOGS myWatchdog1 myWatchdog2
```

```
SCHEDULE | CREATE SENSOR POSITION mySensor1 WATCHDOGS myWatchdog1 myWatchdog2
PARSE> BUILD NETWORK WITH COMPONENTS mySensor1
SCHEDULE | BUILD NETWORK WITH COMPONENTS mySensor1
PARSE> SET SENSOR mySensor1 VALUE 2
SCHEDULE | SET SENSOR mySensor1 VALUE 2
PARSE> @CLOCK wait FOR .03
SCHEDULE | @CLOCK wait FOR .03
PARSE> @exit
SCHEDULE | @exit
TIME     | 0.02
EXECUTE  | @CONFIGURE LOG "a.txt" DOT SEQUENCE "b.txt" NETWORK "c.txt" XML
"d.txt"
EXECUTE  | CREATE WATCHDOG BAND myWatchdog1 MODE INSTANTANEOUS THRESHOLD LOW 1
HIGH 3
WatchdogBand{thresholdLow=1.0 thresholdHigh=3.0
mode=WatchdogModeInstantaneous{} complianceFailureCount=0
complianceFailureThreshold=0}
EXECUTE  | CREATE WATCHDOG BAND myWatchdog2 MODE INSTANTANEOUS THRESHOLD LOW 10
HIGH 30
WatchdogBand{thresholdLow=10.0 thresholdHigh=30.0
mode=WatchdogModeInstantaneous{} complianceFailureCount=0
complianceFailureThreshold=0}
EXECUTE  | CREATE SENSOR POSITION mySensor1 WATCHDOGS myWatchdog1 myWatchdog2
EXECUTE  | BUILD NETWORK WITH COMPONENTS mySensor1
EXECUTE  | SET SENSOR mySensor1 VALUE 2
WATCHDOG | complianceFailureCount=reset
EXECUTE  | @CLOCK wait FOR .03
TIME     | 0.03
TIME     | 0.04
TIME     | 0.05
EXECUTE  | @exit
EXITING  |
```

6. A brief discussion on how the actual results differ from the expected results.

Answer: I expected an error to be throw at the invalid case of the watchdog where it should not compile. The parser did take the command of the invalid version but once the value of the sensor was set the execution of the invalid version brought up a handle case for a value lower value of the sensor on behalf of the invalid version.

7. A suggestion for how to extend this test to cover related aspects not required here. Your document must be formatted professionally. It must be consistent in all respects across

Answer: Having the watchdog attached to an actuator and a command to move the position of the actuator below the low value or above the high value. This would test if the value passed to the watchdog will trigger if the high or low are infringed.

Test E.2 Instantaneous Acceleration Watchdog

1. The rationale behind the test; i.e., what is it testing and why we care.

Answer: This test of an instantaneous acceleration watchdog is to implement a watchdog that is attached to the sensor. This will monitor the value passed store to the sensor and read that value as acceleration. That acceleration value will be check by the low and high values that accompany the construction of the watchdog. There is also an invalid version of that watchdog that will have values beyond that of the valid version an thus should throw an error.

2. A general English description of the initial conditions of the test.

Answer: The parse is given the command to create a valid version of an instantaneous acceleration watchdog and an invalid version of an instantaneous acceleration watchdog. The valid version has both a high and low value attached to it. While the invalid version has values beyond that of the valid version. A sensor is then added with both watchdogs and a network command is given that the sensor is attached to. The sensor is then set between the values of myWatchdog3 and below that of myWatchdog4. A clock is set to count to .03 seconds

3. The commands for (2), which must appear in a standalone form that could be directly copied into a text file to reproduce the test without manual intervention. Do not cross-reference other tests.

```
@CONFIGURE LOG \"a.txt\" DOT SEQUENCE \"b.txt\" NETWORK \"c.txt\" XML \"d.txt\"
CREATE WATCHDOG ACCELERATION myWatchdog3 MODE INSTANTANEOUS THRESHOLD LOW 1
HIGH 3
CREATE WATCHDOG ACCELERATION myWatchdog4 MODE INSTANTANEOUS THRESHOLD LOW 10
HIGH 30
CREATE SENSOR POSITION mySensor2 WATCHDOGS myWatchdog3 myWatchdog4
BUILD NETWORK WITH COMPONENTS mySensor2
SET SENSOR mySensor2 VALUE 2
@CLOCK wait FOR .03
@exit
```

4. A brief English narrative of the expected results of executing the test. (Proper testing discipline expects that you do this before running the test.)

Answer: The watchdog should be created and its values both low and high with an optional grace value that is not a part of this test will be registered while the invalid should throw an error. There is also a sensor that both the valid and invalid watchdog are attached to. There is a command to set the value of the sensor to within range of the valid watchdog. It should work until executing the invalid myWatchdog4 command.

5. At least one representation of the actual results. The form is your choice.

```
Welcome to your ParserHelper
Welcome to your Startup class
PARSE> @CONFIGURE LOG "a.txt" DOT SEQUENCE "b.txt" NETWORK "c.txt" XML "d.txt"
SCHEDULE | @CONFIGURE LOG "a.txt" DOT SEQUENCE "b.txt" NETWORK "c.txt" XML
"d.txt"
PARSE> CREATE WATCHDOG ACCELERATION myWatchdog3 MODE INSTANTANEOUS THRESHOLD
LOW 1 HIGH 3
SCHEDULE | CREATE WATCHDOG ACCELERATION myWatchdog3 MODE INSTANTANEOUS
THRESHOLD LOW 1 HIGH 3
```

```
PARSE> CREATE WATCHDOG ACCELERATION myWatchdog4 MODE INSTANTANEOUS THRESHOLD
LOW 10 HIGH 30
SCHEDULE | CREATE WATCHDOG ACCELERATION myWatchdog4 MODE INSTANTANEOUS
THRESHOLD LOW 10 HIGH 30
PARSE> CREATE SENSOR POSITION mySensor2 WATCHDOGS myWatchdog3 myWatchdog4
SCHEDULE | CREATE SENSOR POSITION mySensor2 WATCHDOGS myWatchdog3 myWatchdog4
PARSE> BUILD NETWORK WITH COMPONENTS mySensor2
SCHEDULE | BUILD NETWORK WITH COMPONENTS mySensor2
PARSE> SET SENSOR mySensor2 VALUE 2
SCHEDULE | SET SENSOR mySensor2 VALUE 2
PARSE> @CLOCK wait FOR .03
SCHEDULE | @CLOCK wait FOR .03
PARSE> @exit
SCHEDULE | @exit
TIME     | 0.02
EXECUTE  | @CONFIGURE LOG "a.txt" DOT SEQUENCE "b.txt" NETWORK "c.txt" XML
"d.txt"
EXECUTE  | CREATE WATCHDOG ACCELERATION myWatchdog3 MODE INSTANTANEOUS
THRESHOLD LOW 1 HIGH 3
WatchdogAcceleration{thresholdLow=1.0 thresholdHigh=3.0
mode=WatchdogModeInstantaneous{} complianceFailureCount=0
complianceFailureThreshold=0}
EXECUTE  | CREATE WATCHDOG ACCELERATION myWatchdog4 MODE INSTANTANEOUS
THRESHOLD LOW 10 HIGH 30
WatchdogAcceleration{thresholdLow=10.0 thresholdHigh=30.0
mode=WatchdogModeInstantaneous{} complianceFailureCount=0
complianceFailureThreshold=0}
EXECUTE  | CREATE SENSOR POSITION mySensor2 WATCHDOGS myWatchdog3 myWatchdog4
EXECUTE  | BUILD NETWORK WITH COMPONENTS mySensor2
EXECUTE  | SET SENSOR mySensor2 VALUE 2
WATCHDOG | complianceFailureCount=reset
EXECUTE  | @CLOCK wait FOR .03
TIME     | 0.03
TIME     | 0.04
TIME     | 0.05
EXECUTE  | @exit
EXITING  |
```

6. A brief discussion on how the actual results differ from the expected results.

Answer: The result did not throw an error when creating the invalid version of the watchdog. Instead, what was outputted was an error message that read compliance error due to the value of the sensor being lower than the invalid version of the watchdog which means that a sensor could be created where its value is within the invalid form.

7. A suggestion for how to extend this test to cover related aspects not required here. Your document must be formatted professionally. It must be consistent in all respects across.

Answer: The test should have the watchdog attached to two sensors to measure the compliance of both. Having an actuator that the sensor with the watchdog attached to would give more extensive testing to examine how the watchdog responds and possibly what a consumer of this product would try and do.

Test A.1: Basic Actuator Creation

1. The rationale behind the test; i.e., what is it testing and why we care.

Answer: This test shows that the actuator is successfully created on the network.

2. A general English description of the initial conditions of the test.

Answer: The initial conditions required for this test will be an empty network.

3. The commands for (2), which must appear in a standalone form that could be directly copied into a text file to reproduce the test without manual intervention. Do not cross-reference other tests.

```
@CONFIGURE LOG \"a.txt\" DOT SEQUENCE \"b.txt\" NETWORK \"c.txt\" XML \"d.txt\"
CREATE SENSOR POSITION mySensor1
CREATE ACTUATOR LINEAR myActuator1 GROUPS g1 g2 g3 g4 SENSOR mySensor1
ACCELERATION LEADIN 0.1 LEADOUT -0.2 RELAX 0.3 VELOCITY LIMIT 5 VALUE MIN 1 MAX
20 INITIAL 2 JERK LIMIT 3
BUILD NETWORK WITH COMPONENTS myActuator1
@exit
```

4. A brief English narrative of the expected results of executing the test. (Proper testing discipline expects that you do this before running the test.)

Answer: This test will show a proper creation of an actuator component. The actuator will be created and placed into groups g1-4 and have mySensor1 attached.

5. At least one representation of the actual results. The form is your choice.

```
▼<network>
  ▼<controller id="myControllerMaster" type="MyControllerMaster">
    ▼<components>
      ▼<component id="cli" controllerID="myControllerMaster">
        ▼<groups>
            <group id="interface"/>
            <group id="all"/>
          </groups>
        </component>
      ▼<actuator id="myActuator1" controllerID="myControllerMaster">
        ▼<groups>
            <group id="g1"/>
            <group id="g2"/>
            <group id="g3"/>
            <group id="g4"/>
            <group id="actuator"/>
            <group id="all"/>
          </groups>
        ▼<sensor id="mySensor1" controllerID="myControllerMaster">
          ▼<groups>
              <group id="sensor"/>
              <group id="all"/>
            </groups>
            <watchdogs> </watchdogs>
            <reporters> </reporters>
          </sensor>
        </actuator>
      </components>
    </controller>
  </network>
```

6. A brief discussion on how the actual results differ from the expected results.

Answer: These results fit the expected results. The actuator can be seen on the network with the assigned attached components and in the correct groups.

7. A suggestion for how to extend this test to cover related aspects not required here

Answer: This test could be expanded to include more components watching the actuator and checking the data they report.

Task A.2: Basic Actuator Manipulation

1. The rationale behind the test; i.e., what is it testing and why we care.

Answer: This test shows the actuator's ability to move from one position to another.

2. A general English description of the initial conditions of the test.

Answer: The initial conditions for this test will be an empty network. The test will create an actuator and show it moves.

3. The commands for (2), which must appear in a standalone form that could be directly copied into a text file to reproduce the test without manual intervention. Do not cross-reference other tests.

```
@CONFIGURE LOG \"a.txt\" DOT SEQUENCE \"b.txt\" NETWORK \"c.txt\" XML \"d.txt\"

CREATE ACTUATOR LINEAR myActuator1 GROUPS g1 g2 g3 g4 ACCELERATION LEADIN 0.1
LEADOUT -0.2 RELAX 0.3 VELOCITY LIMIT 5 VALUE MIN 1 MAX 20 INITIAL 2 JERK LIMIT
3

BUILD NETWORK WITH COMPONENTS myActuator1

@CLOCK WAIT UNTIL 0.5

@exit
```

4. A brief English narrative of the expected results of executing the test. (Proper testing discipline expects that you do this before running the test.)

Answer: The results of this test show the actuator's position at each time step. The expected results are the actuator to begin moving and to reach the final expected position of 15 and then stop.

5. At least one representation of the actual results. The form is your choice.

time	position	velocity	comment
0.04	2	0	StateAscendingLeadin
0.05	2.1	0.1	StateAscendingLeadin
0.06	2.3	0.2	StateAscendingLeadin
0.07	2.6	0.3	StateAscendingLeadin
0.08	3	0.4	StateAscendingLeadin
0.09	3.5	0.5	StateAscendingLeadin
0.1	4.1	0.6	StateAscendingLeadin
0.11	4.8	0.7	StateAscendingLeadin
0.12	5.6	0.8	StateAscendingLeadin
0.13	6.5	0.9	StateAscendingLeadin
0.14	7.5	1	StateAscendingLeadin
0.15	8.6	1.1	StateAscendingLeadin
0.16	9.8	1.2	StateAscendingLeadin
0.17	10.8	-1.2	StateAscendingLeadin
0.18	12	-1	StateAscendingLeadout
0.19	13	-0.8	StateAscendingLeadout
0.2	13.8	-0.6	StateAscendingLeadout
0.21	14.4	-0.4	StateAscendingLeadout
0.22	14.8	-0.2	StateAscendingLeadout
0.23	15	0	StateAscendingLeadout

6. A brief discussion on how the actual results differ from the expected results.

Answer: The actuator's motion meets expectation of increasing its position and velocity and then slowing to a stop at the target value of 15.

7. A suggestion for how to extend this test to cover related aspects not required here

Answer: Extending this test may include moving the actuator in both directions and changing directions while moving towards a different target destination.

Task F.1: Ping Message to Actuator

1. The rationale behind the test; i.e., what is it testing and why we care.

Answer: This test shows the message command can ping the master controller, and the forwarding master controller will send the ping to an attached actuator.

2. A general English description of the initial conditions of the test.

Answer: The initial conditions for this test are an empty network.

3. The commands for (2), which must appear in a standalone form that could be directly copied into a text file to reproduce the test without manual intervention. Do not cross-reference other tests.

```
@CONFIGURE LOG \"a.txt\" DOT SEQUENCE \"b.txt\" NETWORK \"c.txt\" XML \"d.txt\"

CREATE ACTUATOR LINEAR myActuator1 GROUPS g1 g2 g3 g4 ACCELERATION LEADIN 0.1
LEADOUT -0.2 RELAX 0.3 VELOCITY LIMIT 5 VALUE MIN 1 MAX 20 INITIAL 2 JERK LIMIT
3

BUILD NETWORK WITH COMPONENTS c1 myActuator1

SEND MESSAGE PING

@exit
```

4. A brief English narrative of the expected results of executing the test. (Proper testing discipline expects that you do this before running the test.)

Answer: The test results should show a ping message received by the controller and then forwarded to the attached components which should reply to the ping.

5. At least one representation of the actual results. The form is your choice.

index	time	tick	submitted_tick	message_id	message_type	priority	mode	sender_id	recipient_ids	recipient_groups	recipient_serviced_ids	payload
1	0.03	3	3	1	MessagePing	HIGH	CONTINUE	REQUEST	cli			all
2	0.03	3	3	2	MessagePingReply	NORMAL	CONTINUE	REQUEST	myActuator1		cli	

6. A brief discussion on how the actual results differ from the expected results.

Answer: These results meet the expected results of the test.

7. A suggestion for how to extend this test to cover related aspects not required here.

Answer: The test could be extended by including more items on the controller, and even including subcomponents on the controller child items as well.