

---

# **SoloProject**

**Report on Configuration  
DefaultConfig**

# PACKAGES

## PIC\_serial

### GLOBALS:

#### Relations:

##### **itsPICserial**

Composition of PICserial, Multiplicity of 1, Uni-directional

### CLASSES:

---

#### **PICserial**

#### Subclasses:

myPIC

#### Operations:

##### **getAllKeyData**

*Gets status/value of the Keypad buttons.  
Status is returned in the upper 2 bytes.  
Values are returned in the bottom 2 bytes.  
The parameter COSreset will*

Primitive-operation , Public, Return type is int

#### Args:

int COSreset = 0xFFFF

#### Body

```
union keydata {
    long temp;
    char t4[4];
} KeyData;
KeyData.t4[3]=getKeypadStatus(0,(char)((COSreset>>8)& 0xFF));
KeyData.t4[2]=getKeypadStatus(1,(char)((COSreset)& 0xFF));
KeyData.t4[1]=getKeypadValues(0);
KeyData.t4[0]=getKeypadValues(1);
```

```
return KeyData.temp;
```

##### **getAnalogValues**

*Returns the value of each of the 3 analog values on the PIC board.*

Primitive-operation , Public, Return type is 'unsigned char'

#### Args:

'int' i1

Constant

#### Body

```
i1= min(max(i1,0),2);
return Analog[i1];
```

##### **getKeypadStatus**

*Gets value of the Keypad character status.*

Primitive-operation , Protected, Return type is 'unsigned char'

#### Args:

int i1

char COSreset

*Which bits get reset when a status read is performed.*

#### Body

```
il= min(max(il,0),1);
char temp = KeyStatus[i1];
KeyStatus[i1] &= (COSreset^0xFF);
return temp;
```

### **getKeypadValues**

*Gets value of the Keypad character.*

Primitive-operation , Protected, Return type is 'unsigned char'

#### Args:

'int' i1

#### Body

```
il= min(max(i1,0),1);
char temp = Keypad[i1];
Keypad[i1] &= 0x7f;
return temp;
```

### **PICserial**

Constructor , Public

#### Args:

int ComPort = 1

*This is the number of the communications port of this object.*

#### Initializers

leds(0)

#### Body

```
if(ComPort==1)fd = open("/tyCo/0",O_RDWR,0644);
if(ComPort==2)fd = open("/tyCo/1",O_RDWR,0644);
io_status = ioctl(fd,FIOBAUDRATE,2400);
//cout << "ioctl status for " << fd << " = " << io_status << endl;
for(int i=0;i<32;i++) Display[i]=0;
Keypad[0]=0;
Keypad[1]=0;
KeyStatus[0]=0;
KeyStatus[1]=0;
```

### **ReadData**

*5 characters are read from the PIC board.*

*The first 2 are the Keypad and the last 3 are the analog pots.*

*A check is made to determine if a COS has occurred in any of the Keypad values. TheKeypadStatus bits are set if there has been a change of state.*

Primitive-operation , Protected, Return type is OMBoolean

#### Body

```
io_status = ioctl (fd, FIONREAD, (int)readBuffer);
if(readBuffer[0]== 5) {
    read(fd, readBuffer, 5); //read in 5 characters.

    KeyStatus[0] |= (readBuffer[0] ^ Keypad[0]);
    Keypad[0]=readBuffer[0];

    KeyStatus[1] |= (readBuffer[1] ^ Keypad[1]);
    Keypad[1]=readBuffer[1];

    Analog[0]= readBuffer[2];
    Analog[1]= readBuffer[3];
    Analog[2]= readBuffer[4];

    //GEN(evGoodReadPIC_Serial);
    return(true);
}
else
{
    cout << "receive error in PIC_Serial" << endl;
```

```

        io_status = ioctl (fd, FIOFLUSH, 0);
        //GEN(evRetryPIC_SerialRead);
        return(false);
    }

```

### setDisplayChar

*Allows the user to set each element of the 32 character Display on the PIC board.*

Primitive-operation , Public, Return type is 'void'

#### Args:

'int' i1  
'char' p\_Display

#### Body

```

i1= min(max(i1,0),31);
Display[i1] = p_Display;

```

### setLeds

*Allows the user to set the 8 led's on the PIC board.*

Primitive-operation , Public, Return type is 'void'

#### Args:

'char' p\_leds

#### Body

```

leds = p_leds;

```

### writeLeds

*Write the leds value to the PIC board.*

Primitive-operation , Protected, Return type is void

#### Body

```

ControlChar[0] = 'L';
io_status = write(fd, ControlChar, 1); //write an 'R' to the PIC board to read the
buffers.
io_status = write(fd, (char*)&leds, 1); //write in 1 character.
//cout << "Led's = " << ControlChar[0] << (char) leds << leds << endl;

```

### ~PICserial

*Shutdown this object;*

Virtual, Destructor , Public

#### Body

```

close(fd);

```

### Attributes:

#### Analog

*This is the location of the three analog inputs.*

Type of 'char %s[3]', Public

#### ControlChar

*This character is set to the control for the PIC device.*

'L' is used to set the value of the LEDs

- L#

'B' is used to set the 32 char board display

- Bcccccccccccccccccccccccccccccccccccc

'R' is used to initiate a transmission of analog and button values.

Type of 'char %s[1]', Public

#### Display

*32 char display buffer on the PIC board*

Type of 'char %s[32]', Public

#### fd

*This is the file descriptor of the serial port returned by the open statement.*

Type of int, Public

#### goodread

*This is used to indicate the status of the read operation to the PIC board.*

Type of OMBoolean, Public

**io\_status**

*contains the status of all i/o operations.*

Type of int, Public

**Keypad**

*There are 2 bytes of keypad*

Type of 'char %s[2]', Public

**KeyStatus**

*There are 2 bytes of keypad status*

Type of 'char %s[2]', Public

**leds**

*the led display outputs*

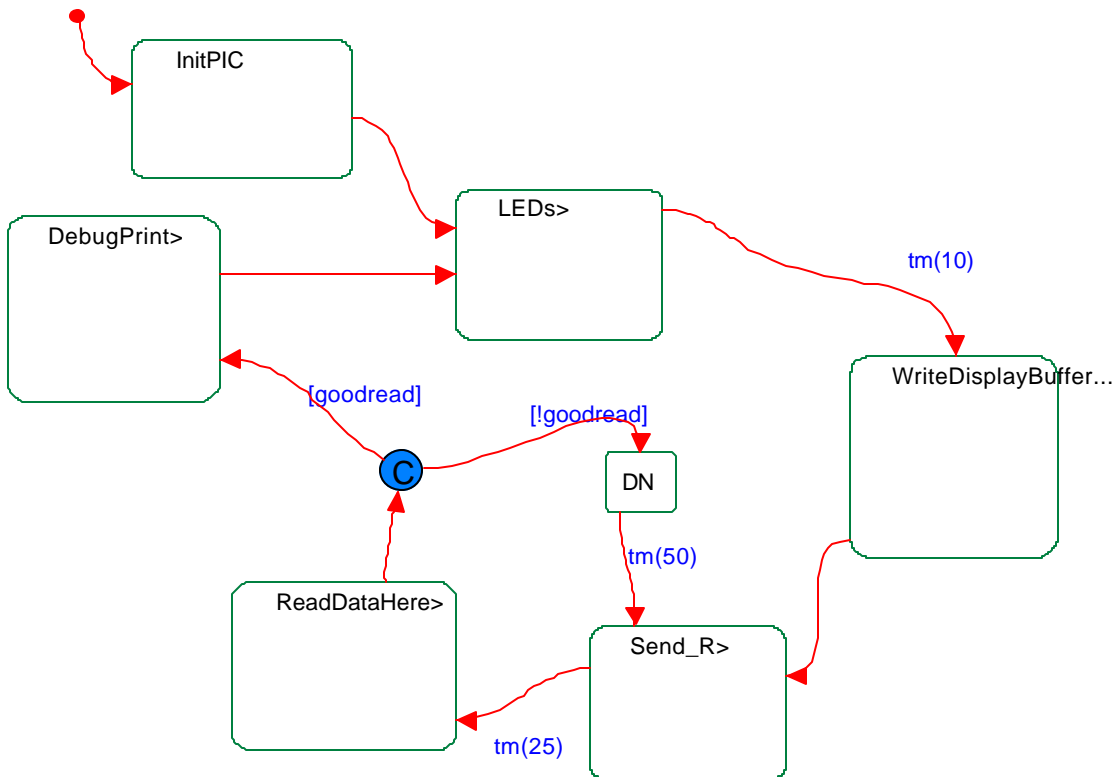
Type of char, Public

**readBuffer**

*buffer to place the data read in by the ReadData operation.*

Type of 'char %s[5]', Public

Statechart



**ROOT**

Or-state

Substates:

DebugPrint

DN

InitPIC

LEDs

ReadDataHere

Send\_R

WriteDisplayBuffer

Default Transition

Target:  
InitPIC

### DebugPrint

*Debug state to determine any possible difficulties with the PIC board.*

Or-state

#### EntryAction

```
//cout << " keys " << setbase(16) << setfill('0') << setw(8) << getAllKeyData() ;  
//cout << " a0 " << (int)getAnalogValues(0);  
//cout << " a1 " << (int)getAnalogValues(1);  
//cout << " a2 " << (int)getAnalogValues(2);  
//cout << endl;  
  
//cout << setbase(16) << setfill('0') << setw(8) << getAllKeyData() << " " <<  
(int)getAnalogValues(0) << " " << (int)getAnalogValues(1) << " " << (int)getAnalogValues(2)  
<< endl;  
  
//for (int i=0;i<5;i++)  
//{ cout << " " << setbase(16) << (int) readBuffer[i]; }  
//cout << endl;
```

Out Transition

Target:  
LEDs

### DN

*Do nothing state used to create a small delay in the I/O stream.*

Or-state

Out Transition

tm(50)

Target:  
Send\_R

### InitPIC

*A do-nothing state for future possible use.*

Or-state

Out Transition

Target:  
LEDs

### LEDs

*Write the leds attribute to the PIC board.*

Or-state

#### EntryAction

```
writeLeds();
```

#### ExitAction

```
//leds++;
```

Out Transition

tm(10)

Target:  
WriteDisplayBuffer

### ReadDataHere

*Read in the 5 characters that the PIC board returns in response to a request for data.*

Or-state

#### EntryAction

```
goodread = ReadData();
```

Out Transition

Condition Connector

Branches:

[goodread]

Target:

DebugPrint

[!goodread]

Target:

DN

**Send\_R**

*Initiate the reading of the data from the PIC board by sending an 'R' to the board.*

Or-state

EntryAction

```
ControlChar[0] = 'R';  
write(fd, ControlChar, 1); //write an 'R' to the PIC board to read the buffers.
```

Out Transition

tm(25)

Target:

ReadDataHere

**WriteDisplayBuffer**

*initiate the execution of the output to the Display on the PIC board.  
This state uses a sub\_state\_chart to accomplish this operation.*

Nested Statechart

Or-state

EntryAction

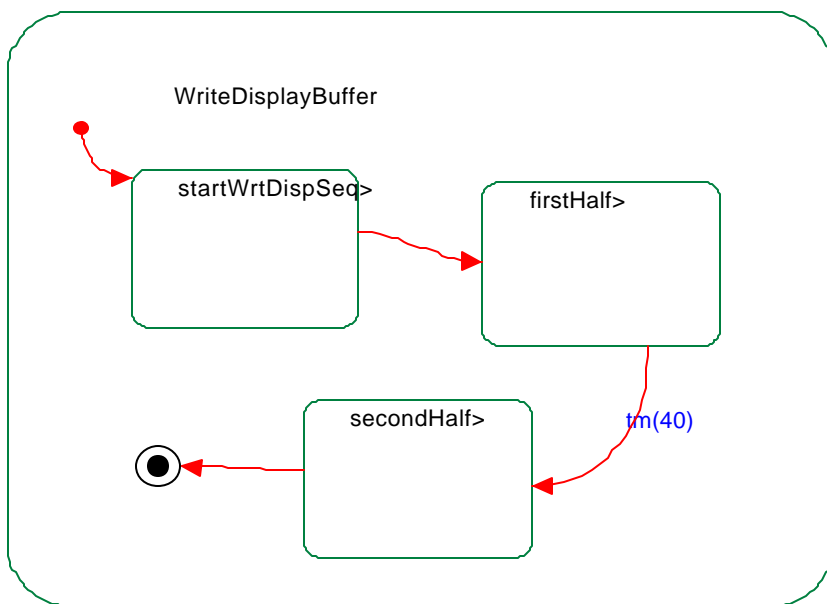
```
//for(int i=0;i<32;i++) Display[i]=leds;
```

Out Transition

Target:

Send\_R

Statechart of WriteDisplayBuffer



**ROOT**

Or-state

Substates:

WriteDisplayBuffer

**WriteDisplayBuffer**

Or-state

Substates:

firstHalf

secondHalf

startWrtDispSeq

state\_9

Default Transition

Target:

startWrtDispSeq

**firstHalf**

*The actual transmission of characters is broken down into 2 halves.*

*Send out the first 16 characters.*

Or-state

EntryAction

```
write(fd, Display, 16); //write out 16 characters.
```

Out Transition

```
tm(40)
```

Target:

secondHalf

**secondHalf**

*After a short delay, send out the second 16 characters.*

Or-state

EntryAction

```
write(fd, &Display[16], 16); //write out 16 characters.
```

Out Transition

Target:

state\_9

**startWrtDispSeq**

*Write the control character 'B' to the PIC board to initiate the sequence of characters to be displayed on the 32 element character display.*

Or-state

EntryAction

```
ControlChar[0] = 'B';
```

```
write(fd, ControlChar, 1); //write an 'B' to the PIC board to read the buffers.
```

Out Transition

Target:

firstHalf

**state\_9**

Local Termination State

# Users

GLOBALS:

CLASSES:

## myPIC

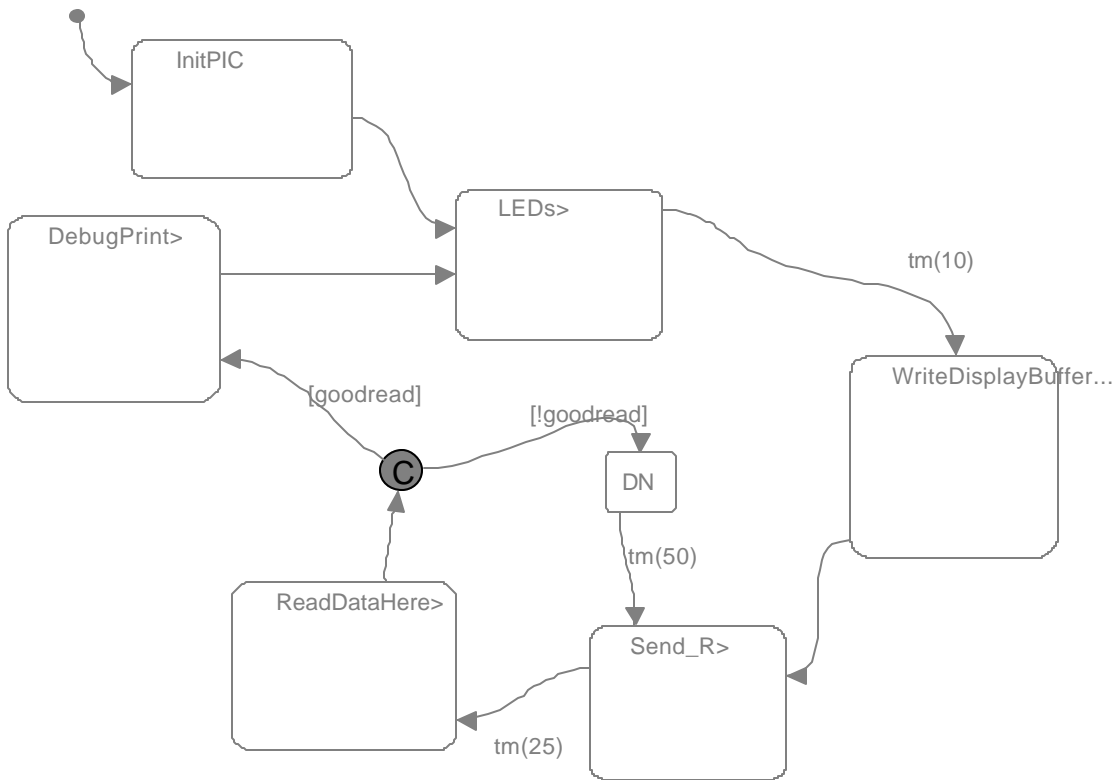
*This is the PIC board where the user inputs will be to set attributes of the washmachine. Also the PIC board will have outputs to show the user what step is currently running.*

Superclasses:

PICserial

Public

Statechart



## ROOT

Or-state

Inherited

Substates:

DebugPrint

DN

InitPIC

LEDs

ReadDataHere

Send\_R

WriteDisplayBuffer

Default Transition

Inherited

Target:

InitPIC

**DebugPrint**

Or-state

Inherited

EntryAction

```
//cout << " keys " << setbase(16) << setfill('0') << setw(8) << getAllKeyData() ;
//cout << " a0 " << (int)getAnalogValues(0);
//cout << " a1 " << (int)getAnalogValues(1);
//cout << " a2 " << (int)getAnalogValues(2);
//cout << endl;

//cout << setbase(16) << setfill('0') << setw(8) << getAllKeyData() << " " <<
(int)getAnalogValues(0) << " " << (int)getAnalogValues(1) << " " << (int)getAnalogValues(2)
<< endl;

//for (int i=0;i<5;i++)
//{ cout << " " << setbase(16) << (int) readBuffer[i]; }
//cout << endl;
```

Out Transition

Inherited

Target:

LEDs

**DN**

Or-state

Inherited

Out Transition

Inherited

tm(50)

Target:

Send\_R

**InitPIC**

Or-state

Inherited

Out Transition

Inherited

Target:

LEDs

**LEDs**

Or-state

Inherited

EntryAction

writeLeds();

ExitAction

//leds++;

Out Transition

Inherited

tm(10)

Target:

WriteDisplayBuffer

**ReadDataHere**

Or-state

Inherited

EntryAction

```
goodread = ReadData();
```

Out Transition

Inherited

Condition Connector

Branches:

```
[goodread]
```

Target:

DebugPrint

```
[!goodread]
```

Target:

DN

**Send\_R**

Or-state

Inherited

EntryAction

```
ControlChar[0] = 'R';
```

```
write(fd, ControlChar, 1); //write an 'R' to the PIC board to read the buffers.
```

Out Transition

Inherited

```
tm(25)
```

Target:

ReadDataHere

**WriteDisplayBuffer**

Nested Statechart

Or-state

Inherited

EntryAction

```
//for(int i=0;i<32;i++) Display[i]=leds;
```

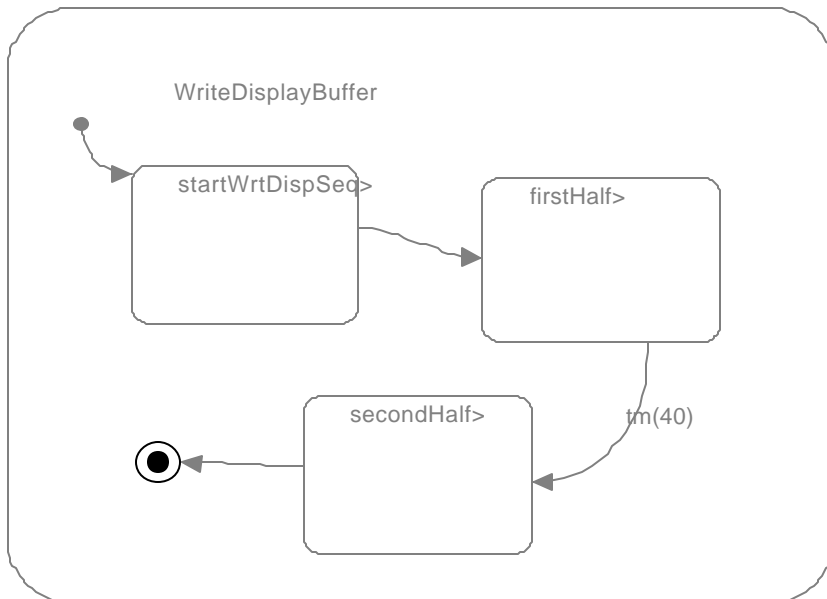
Out Transition

Inherited

Target:

Send\_R

## Statechart of WriteDisplayBuffer



### ROOT

Or-state

Inherited

Substates:

WriteDisplayBuffer

### WriteDisplayBuffer

Or-state

Inherited

Substates:

firstHalf

secondHalf

startWrtDispSeq

state\_0

Default Transition

Inherited

Target:

startWrtDispSeq

### firstHalf

Or-state

Inherited

EntryAction

write(fd, Display, 16); //write out 16 characters.

Out Transition

Inherited

tm(40)

Target:

secondHalf

### secondHalf

Or-state

Inherited

EntryAction

write(fd, &Display[16], 16); //write out 16 characters.

```

    Out Transition
    Inherited
    Target:
        state_0
startWrtDispSeq
    Or-state
    Inherited
    EntryAction
        ControlChar[0] = 'B';
        write(fd, ControlChar, 1); //write an 'B' to the PIC board to read the buffers.

    Out Transition
    Inherited
    Target:
        firstHalf
state_0
    Local Termination State

```

## ACTORS:

---

### Programer

*This person is the "washer." She puts the clothes and sets the wash machine into play.*

Relations:

**itsSet Operational Mode**

Association with Set Operational Mode, Multiplicity of 1, Bi-directional

**itsSet Washing Parameters**

Association with Set Washing Parameters, Multiplicity of 1, Bi-directional

---

### WashDaMachine

*This is the actual machine and the different things that must change as the clothes are being washed*

Relations:

**itsSet Agitator Speed**

Association with Set Agitator Speed, Multiplicity of 1, Bi-directional

**itsSet Cycle**

Association with Set Cycle, Multiplicity of 1, Bi-directional

**itsSet Water Attributes**

Association with Set Water Attributes, Multiplicity of 1, Bi-directional

**itsSet Basket Speed**

Association with Set Basket Speed, Multiplicity of 1, Bi-directional

**itsDrain Water**

Association with Drain Water, Multiplicity of 1, Bi-directional

---

## WashMachine

*I plan to make a "washing machine."*

*· Analog Input 0 would choose the laundry load (Small to Large). The size of the load would determine how much water is to be placed in the washer. Though this knob is continuous the load size would be determined as follows:*

- o If the value was from 0 – 63, the load would be small*
  - § Water fills for 10 sec. (this is a shorten time for simulation purposes)*
- o If the value was from 64 – 127, the load would be medium*
  - § Water fills for 20 seconds*
- o If the value was from 128 – 191, the load would be the large*
  - § Water fills for 30 seconds*
- o If the value was from 192 – 255, the load would be super*

§ Water fills for 40 seconds

\*\*\* NOTE if the water knob changes from a smaller to larger or from larger to smaller the latter time for filling will be used. Unless the later time has already surpassed water will then cease to fill, it will not be emptied from the tub. \*\*\*

- Analog Input 1 would determine the temperature of the water to place in the tub.
  - o If the value was from 0 – 63, Warm Cold Wash
  - o If the value was from 64 – 127, Hot Cold Wash
  - o If the value was from 128 – 191, Warm Warm Wash
  - o If the value was from 192 – 255, Cold Cold Wash

\*\*\* The input of the water being hot or cold would be shown on the LCD display. During the soak sequence a two bars would show up with the higher the bar the more of that particular heat of water will go into the tub. If during the fill sequence the temperature is changed only the water going into the tub would change and the water that was already in the tub would stay in the tub. Also the amount of water going into the tub would be constant at all times.

- Analog Input 2 would determine the wash cycle time. Turning the knob after the cycle has started will stop the washer and reset the cycle to the new selection. Press start to begin the new cycle selection. The wash cycles is as follows:

COTTON

- o If the value was from 0 – 19, Auto Soak 30 Min
- o If the value was from 29 – 39, Auto Soak 15 Min
- o If the value was from 40 – 59, Wash 12 sec
- o If the value was from 60 – 79, Wash 9 sec
- o If the value was from 80 – 99, Wash 6 sec

EASY CARE

- o If the value was from 100 – 119, Wash 12 sec
- o If the value was from 120 – 139, Wash 9 sec.
- o If the value was from 140 – 159, Wash 6 sec

Delicates

- o If the value was from 160 – 179, Wash 12 sec
- o If the value was from 180 – 199, Wash 9 sec
- o If the value was from 200 – 219, Wash 6 sec
- o If the value was from 220 – 239, Quick Rinse
- o If the value was from 240 – 255, Spin Only

- The LCD outputs for the cycle stage would be following

- o Fill
- o Soak
- o Wash Time (12 sec left)
- o Wash Time (9 sec left)
- o Wash Time (6 sec left)
- o Rinse
- o 2 nd Rinse (optional)
- o Final Spin

- The following binary inputs would be used to determine the wash/spin speed.

- Enter – Normal/Fast
- 4 – Normal/Slow
- 5 – Gentle/Fast
- 6 – Gentle/Slow

WASH SPEEDS:

Normal will use a high speed for the first few minutes, then the agitator slows down to a moderate speed.

Gentle the agitator will move at a slow speed.

## SPIN SPEEDS:

*Fast speed will be used for normal items.*

*Slow speed will be used for delicates.*

*If one changes the speed during the cycle the last button pressed will be used to determine speeds. The speed of the agitator and basket will be shown on the LCD Display as a bar chart. With a higher bar representing a faster turn.*

*· The CHS button would be as a start/stop/pause button used. If the machine was off the machine would be turned on, if during a cycle the button was pressed the machine would pause.*

*· If the '.' button is pressed would mean that the washer was open. This would stop the cycle just as a pause button..*

*· There would be one last option for a 2nd rinse The '0' button. If this option is not selected there would be no second rinse.*

## GLOBALS:

### Relations:

#### **itsBasket**

Composition of Basket, Multiplicity of 1, Uni-directional

#### **itsDrain**

Composition of Drain, Multiplicity of 1, Uni-directional

#### **itsHotPipe**

Composition of HotPipe, Multiplicity of 1, Uni-directional

#### **itsColdPipe**

Composition of ColdPipe, Multiplicity of 1, Uni-directional

#### **itsAgitator**

Composition of Agitator, Multiplicity of 1, Uni-directional

## CLASSES:

---

### **Agitator**

*This is the agitator in the wash machine. It's speed will vary based on*

#### Attributes:

##### **speed\_AgitatorAtt**

*This value sets the speed the agitator will run.*

Type of int, Public

##### **timeOn\_AgitatorAtt**

*This is the amount of time that the agitator would be on. To mix clothes and water.*

Type of int, Public

---

### **Basket**

#### Attributes:

##### **attribute\_1**

Type of int, Public

##### **timeOn\_BasketAtt**

*The amount of time that the basket will be on. For spinning to help dry.*

Type of int, Public

---

## ColdPipe

Superclasses:  
waterPipes  
Public

---

## Drain

Attributes:  
**attribute\_0**  
Type of int, Public

---

## HotPipe

Superclasses:  
waterPipes  
Public

---

## waterPipes

*This is the superclass of the pipes. There are two attributes the amount of water going into the washer at a time and the time that expires before it will stop.*

Subclasses:

ColdPipe  
HotPipe

Attributes:

**pressureIn\_PipeAtt**

*This is the amount of pressure of water that would go in. Based the temperature of the water that is going into the wash machine.*

Type of int, Public

**timeOn\_PipeAtt**

*This is the amount of time that elapses before the water turns off.*

Type of int, Public

## USE CASES:

---

### Drain Water

Relations:  
**itsWashDaMachine**

---

### Set Agitator Speed

*If one changes the speed during the cycle the last button pressed will be used to determine speeds. The speed of the agitator and basket will be shown on the LCD Display as a bar chart. With a higher bar representing a faster turn.*

Relations:

**itsWashDaMachine**

---

### Set Basket Speed

*If one changes the speed during the cycle the last button pressed will be used to determine speeds. The speed of the agitator and basket will be shown on the LCD Display as a bar chart. With a higher bar representing a faster turn.*

Relations:

**itsWashDaMachine**

---

### Set Cycle

- The LCD outputs for the cycle stage would be following
  - o Fill
  - o Soak
  - o Wash Time (12 sec left)

- o Wash Time (9 sec left)
- o Wash Time (6 sec left)
- o Rinse
- o 2nd Rinse (optional)
- o Final Spin

Relations:

**itsWashDaMachine**

## Set Operational Mode

- The CHS button would be as a start/stop/pause button used. If the machine was off the machine would be turned on, if during a cycle the button was pressed the machine would pause.
- If the '.' button is pressed would mean that the washer was open. This would stop the cycle just as a pause button..
- There would be one last option for a 2nd rinse The '0' button. If this option is not selected there would be no second rinse.

Relations:

**itsProgramer**

## Set Washing Parameters

- Analog Input 0 would choose the laundry load (Small to Large). The size of the load would determine how much water is to be placed in the washer. Though this knob is continuous the load size would be determined as follows:
  - o If the value was from 0 – 63, the load would be small
    - § Water fills for 10 sec. (this is a shorten time for simulation purposes)
  - o If the value was from 64 – 127, the load would be medium
    - § Water fills for 20 seconds
  - o If the value was from 128 – 191, the load would be the large
    - § Water fills for 30 seconds
  - o If the value was from 192 – 255, the load would be super
    - § Water fills for 40 seconds
- Analog Input 1 would determine the temperature of the water to place in the tub.
  - o If the value was from 0 – 63, Warm Cold Wash
  - o If the value was from 64 – 127, Hot Cold Wash
  - o If the value was from 128 – 191, Warm Warm Wash
  - o If the value was from 192 – 255, Cold Cold Wash
- Analog Input 2 would determine the wash cycle time. Turning the knob after the cycle has started will stop the washer and reset the cycle to the new selection. Press start to begin the new cycle selection. The wash cycles is as follows:
  - COTTON
    - o If the value was from 0 – 19, Auto Soak 30 Min
    - o If the value was from 29 – 39, Auto Soak 15 Min
    - o If the value was from 40 – 59, Wash 12 sec
    - o If the value was from 60 – 79, Wash 9 sec
    - o If the value was from 80 – 99, Wash 6 sec
  - EASY CARE
    - o If the value was from 100 – 119, Wash 12 sec
    - o If the value was from 120 – 139, Wash 9 sec.
    - o If the value was from 140 – 159, Wash 6 sec
    - Delicates
      - o If the value was from 160 – 179, Wash 12 sec
      - o If the value was from 180 – 199, Wash 9 sec
      - o If the value was from 200 – 219, Wash 6 sec

- o If the value was from 220 – 239, Quick Rinse*
- o If the value was from 240 – 255, Spin Only*

- The following binary inputs would be used to determine the wash/spin speed.*
  - Enter – Normal/Fast*
  - 4 – Normal/Slow*
  - 5 – Gentle/Fast*
  - 6 – Gentle/Slow*

**WASH SPEEDS:**

- Normal will use a high speed for the first few minutes, then the agitator slows down to a moderate speed.*
- Gentle the agitator will move at a slow speed.*

**SPIN SPEEDS:**

- Fast speed will be used for normal items.*
- Slow speed will be used for delicates.*

Relations:

**itsProgrammer**

---

## **Set Water Attributes**

*\*\*\* NOTE if the water knob changes from a smaller to larger or from larger to smaller the latter time for filling will be used. Unless the later time has already surpassed water will then cease to fill, it will not be emptied from the tub. \*\*\**

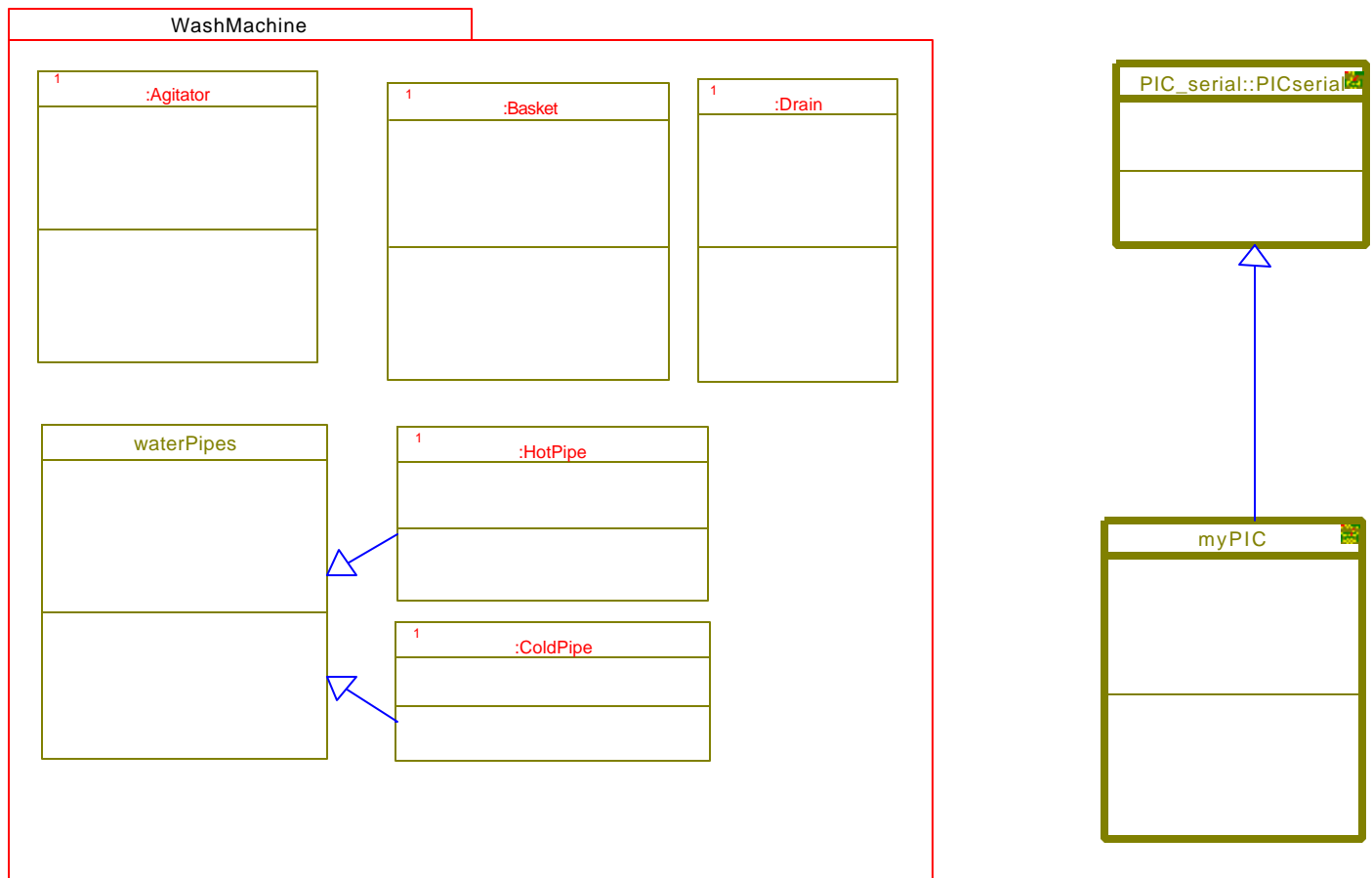
*\*\*\* The input of the water being hot or cold would be shown on the LCD display. During the soak sequence a two bars would show up with the higher the bar the more of that particular heat of water will go into the tub. If during the fill sequence the temperature is changed only the water going into the tub would change and the water that was already in the tub would stay in the tub. Also the amount of water going into the tub would be constant at all times.*

Relations:

**itsWashDaMachine**

# OBJECT MODEL DIAGRAMS

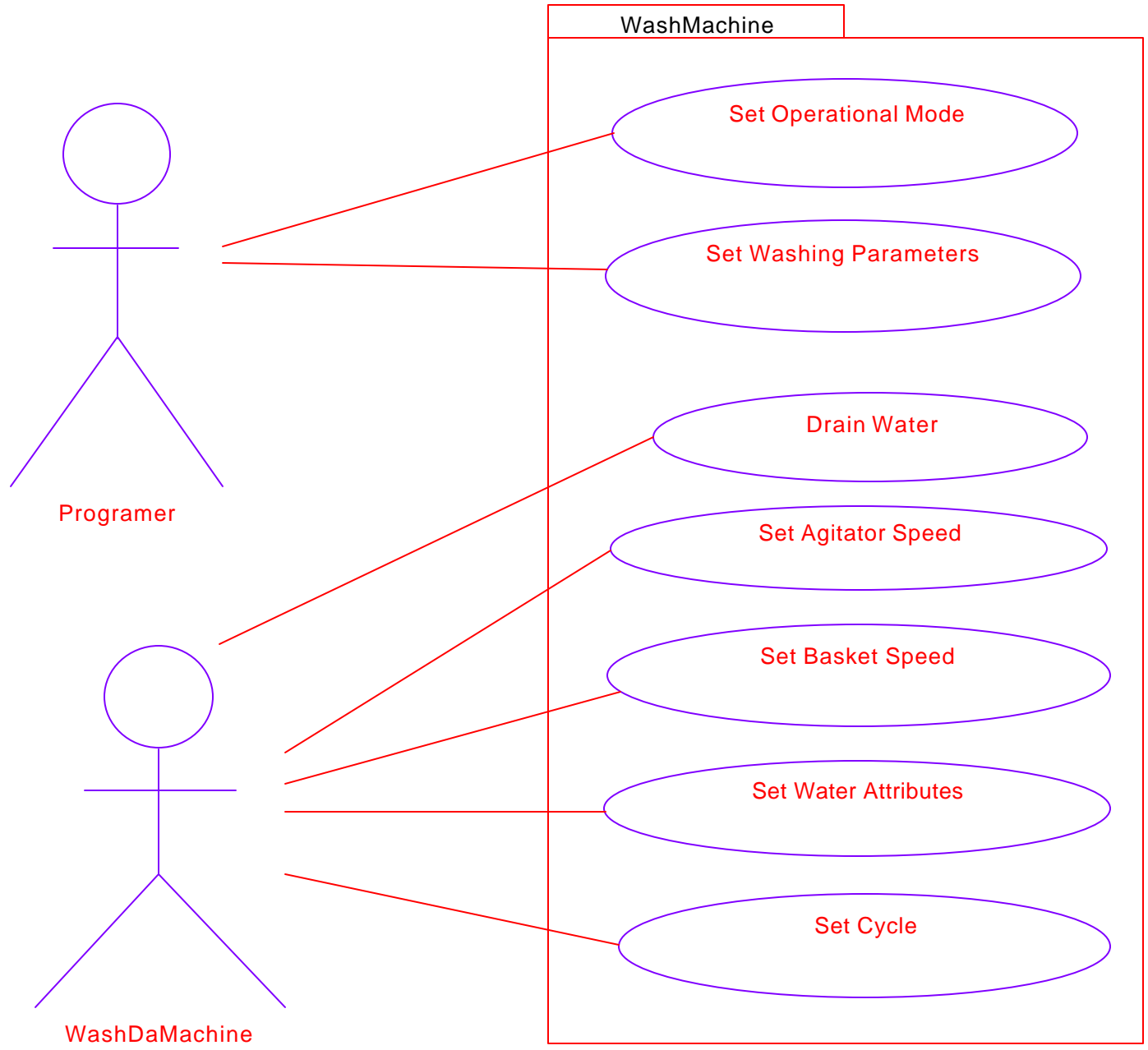
## Model1



There are no associations among the various classes. What are they?

# USE CASE DIAGRAMS

## WashMachine



# COMPONENTS

## DefaultComponent

### COMPONENT SETTINGS:

Build type: Executable

### CONFIGURATIONS:

---

#### **DefaultConfig**

Scope type: Explicit

Instrumentation type: None

Time-model type: Real-time

Statechart generation type: Flat

### FILES AND FOLDERS: