

Control systems which carry out a sequence of steps are very common.

In the old days of relays and contactors, these were implemented by means of motorized drum switches, in which a synchronous motor advanced the switch to the next position as each step completed.

Using AmbiLogic we can construct sequencers which are much more versatile. This is an example of a sequencing control system, modelled on an imaginary washing machine.

This diagram sheet shows the basic sequencer, built from 3 data selectors.

The selector on the right, "Sequence" controls which step is active.

The selector at top left, F0108, issues a digital signal to tell "Sequence" to select its own output (therefore holding the current step), or the analogue output from F0109 which is the step we want to move to next.

This scheme means that we can jump from any step to any other.

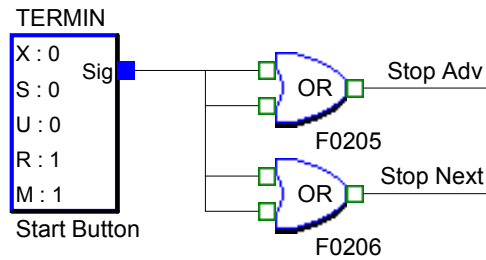
Note that the Sel inputs of F0108 and F0109 are connected to the Sequence output

This means that each step will remain latched until we do two things:

1. Make the "Adv" signal TRUE so that the sequencer becomes controlled by F0109
2. Set the "Next" signal to the step we want to execute next.

Using the Compare Equal blocks, we have generated a digital signal for each step.

The next sheets show how the steps are implemented

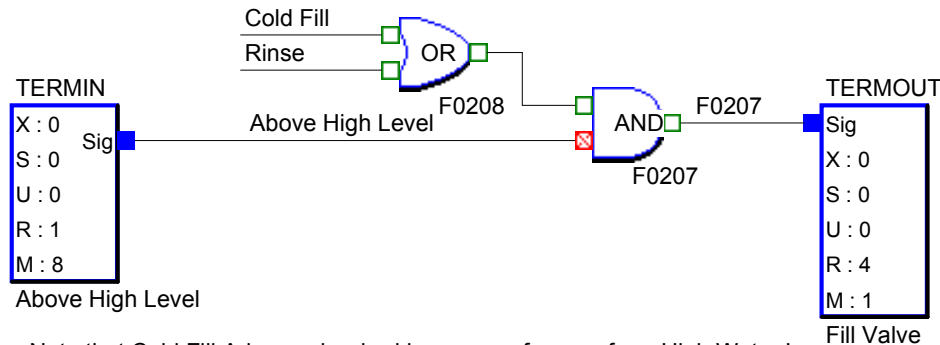


This is how we start the sequence. Pressing the Start button (ISW0 on a CPDA-01) generates a TRUE on Stop Adv and a sneaky value of 1 (see the digital to analogue connection rules) on Stop Next. This means that the sequencer advances to the Cold Fill state

We cannot connect signals with different names together, so we have buffered them with OR gates with their inputs tied together. This means that Stop Adv and Stop Next follow the Start Button.

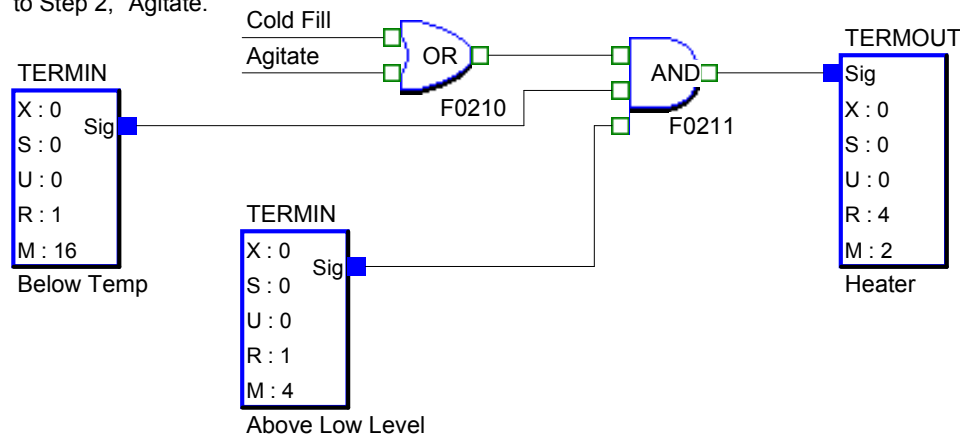
Look back at Sheet 1. Stop Next is not looked at until Stop Adv becomes active. This means that Stop Next can be replaced by a constant '1'.

Stop Adv is only looked at during sequence Stop. This means that we could have wired Start Button directly into the wire currently shown as Stop Adv. Pressing the Start Button during any active step of the sequence will have no effect.



Once we get into Cold Fill, we need to open the Fill Valve (OTR0 on a CPDA-01) The valve needs to be turned off when the water reaches its High level. Notice the inverted input pin on the AND gate. We also noted that we need to fill the drum during the Rinse cycle, so we simply OR the two state signals together.

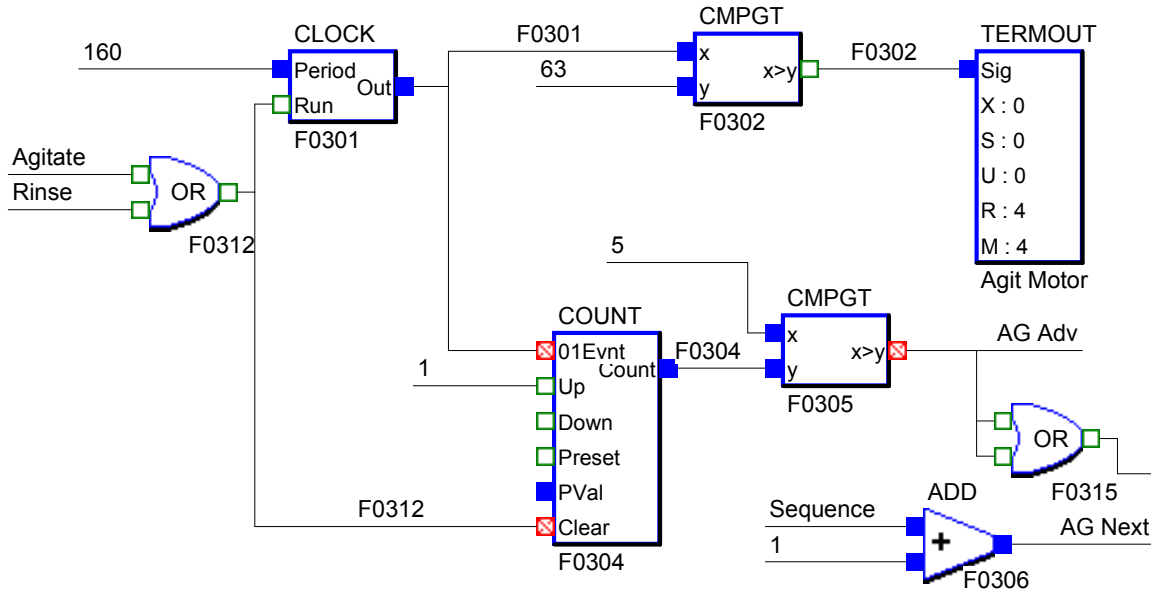
Note that Cold Fill Advance is wired by cross-reference from High Water Level. Also Cold Fill Next has a constant 2 wired into it. This means that as soon as the water reaches its high level, the sequence advances to Step 2, "Agitate."



We want the heater on during Cold Fill or Agitate sequences. But we don't want it on if the water is below the minimum level. (Assume that the level switches are ON for water above the level) The temperature switch is ON if the water temperature is below the set temperature.

During the Agitate mode, we are going to run the Agitator motor for 5 bursts of 6 seconds with a 4 second rest period between. In practice, the period would need to be much longer, but this is a bench demonstration.

Running the heater during Agitate mode has already been covered on Sheet 2.

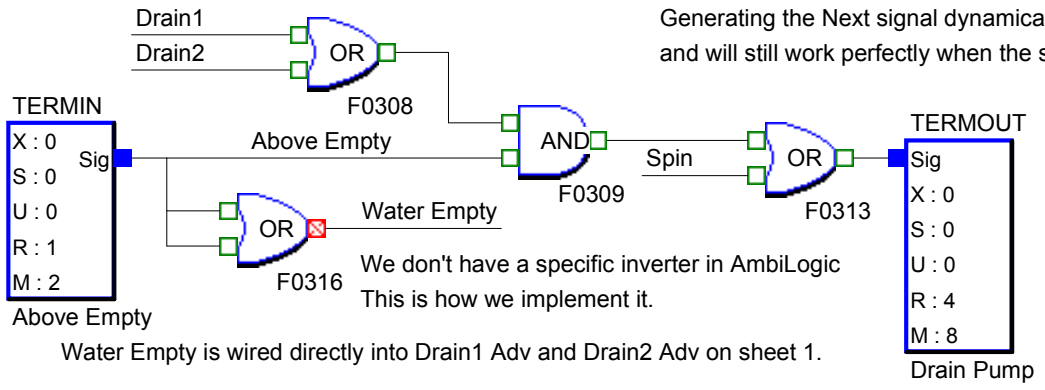


The Clock generates a period of 160 scans. The CPDA-01 scans at 16 Hz, so the period is 10 seconds  
 When Agitate goes TRUE, the Clock output goes to 159 and starts to count down.  
 The CMPGT output goes TRUE and remains so for 159 - 63 = 96 scans i.e. 6 seconds.  
 We could use a Select function to have different numbers of agitator cycles in the Agitate and Rinse steps.

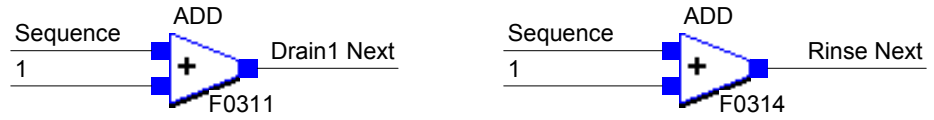
Note the use of the inversion of the output of the CMPGT function. This gives us a TRUE when Y is equal to or greater than X.  
 The counter is cleared when the sequence is not in Agitate mode and counts UP from 0 at the end of each CLOCK period when the output goes from 1 to 0.  
 Rinse Adv

This is a sweet way of generating the Next signal  
 Because the sequencer is on another sheet, we might at some stage change the sequence and therefore the step number for different operations.

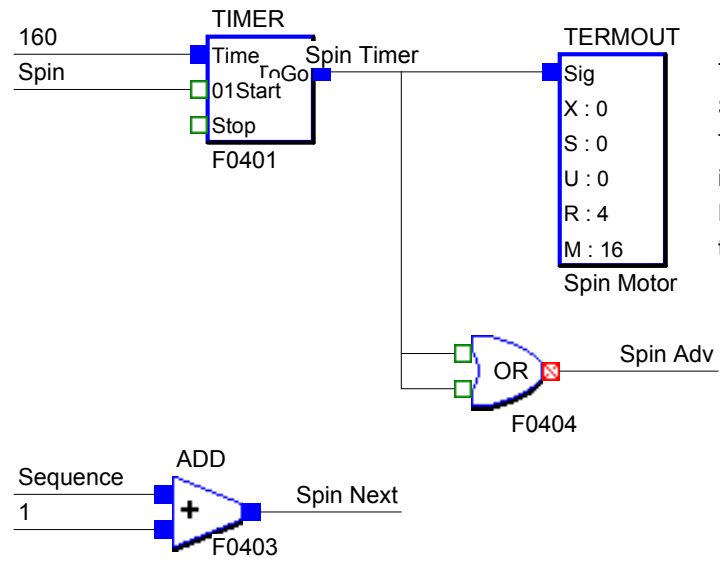
Generating the Next signal dynamically with the adder means that this part of the diagram does not need to be changed, and will still work perfectly when the sequence is changed.



Water Empty is wired directly into Drain1 Adv and Drain2 Adv on sheet 1.



This method of generating Next is used several times in the diagram. The same block could be used for all the instances, but it makes the diagram harder to read.



The spin cycle runs for  $160 / 16 = 10$  seconds.

Spin Adv is a signal which feeds an analogue pin on a data selector.

This means that the pin cannot be inverted, so we have to introduce our own inversion by means of the OR gate.

Remember that the Advance signal can be digital - and we want the sequence to advance when the timer runs down to 0.