Spike

Myfillphase

Functional Design Specification for Manufacturing Control System

APPROVALS

Functional Area	Required Approver	Date
Author / Originator	Print Name:	
	Sign Name:	
Engineering	Print Name:	
	Sign Name:	
Operations	Print Name:	
	Sign Name:	
Automation	Print Name:	
	Sign Name:	
QAV	Print Name:	
	Sign Name:	

Revision History

Revision	Date	Author	Description

This phase template is for filling a vessel. It has built in failure monitor logic and watchdog timers. It requires the following in order to successfully run:

unit with specific parameters as outlined below

creation of CM and EM physical model as outlined below

creation of formula with specific parameters.

Finally, the phase must be encased in a procedure to be executed via the campaign manager.

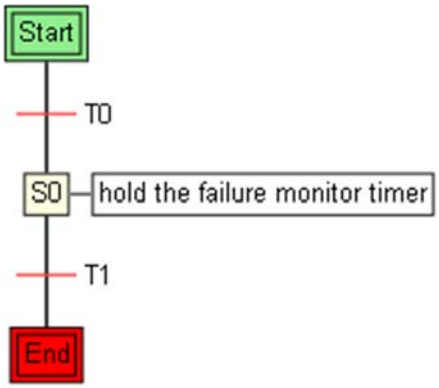
Table of Contents

Reference Documentation

Document	Ref.	Title

1. Myfillphase

1.1.1. Held



1.1.1.1. S0 Logic

```
##########################################
# Step: Held/S0
##########################################


def main():
    """Main Function"""


    # failure monitor timer hold.


    # get a handle to the timer
    mytimer = unit.SupportModule.GetTimer('FailureMonitorTimer')


    # if in HELD state, stop timer
    mytimer.Stop()


    # print status to consle window


    info('phase HELD')
```

1.1.1.2. T0 Logic

```
##########################################
```

```
# Transition: Held/T0
##########################################

def main():
    """Main Function"""

    # Transition code...
```

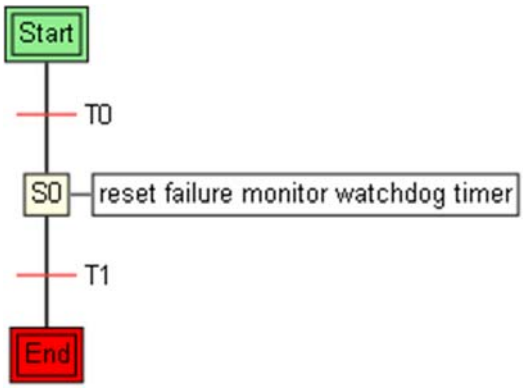
1.1.1.3. T1 Logic

```
##########################################
# Transition: Held/T1
##########################################

def main():
    """Main Function"""

    # Transition code...
```

1.1.2. Restarting



1.1.2.1. S0 Logic

```
##########################################
# Step: Restarting/S0
##########################################

def main():
    """Main Function"""
```

```
    # get a handle to the failure monitor timer

    mytimer = unit.SupportModule.GetTimer('FailureMonitorTimer')

    # reset the failure monitor watchdog timer upon restarting the phase
    mytimer.Reset()

    # start the timer again
    mytimer.Start()
```

1.1.2.2. T0 Logic

```
##########################################
# Transition: Restarting/T0
##########################################

def main():
    """Main Function"""

    # Transition code...
```

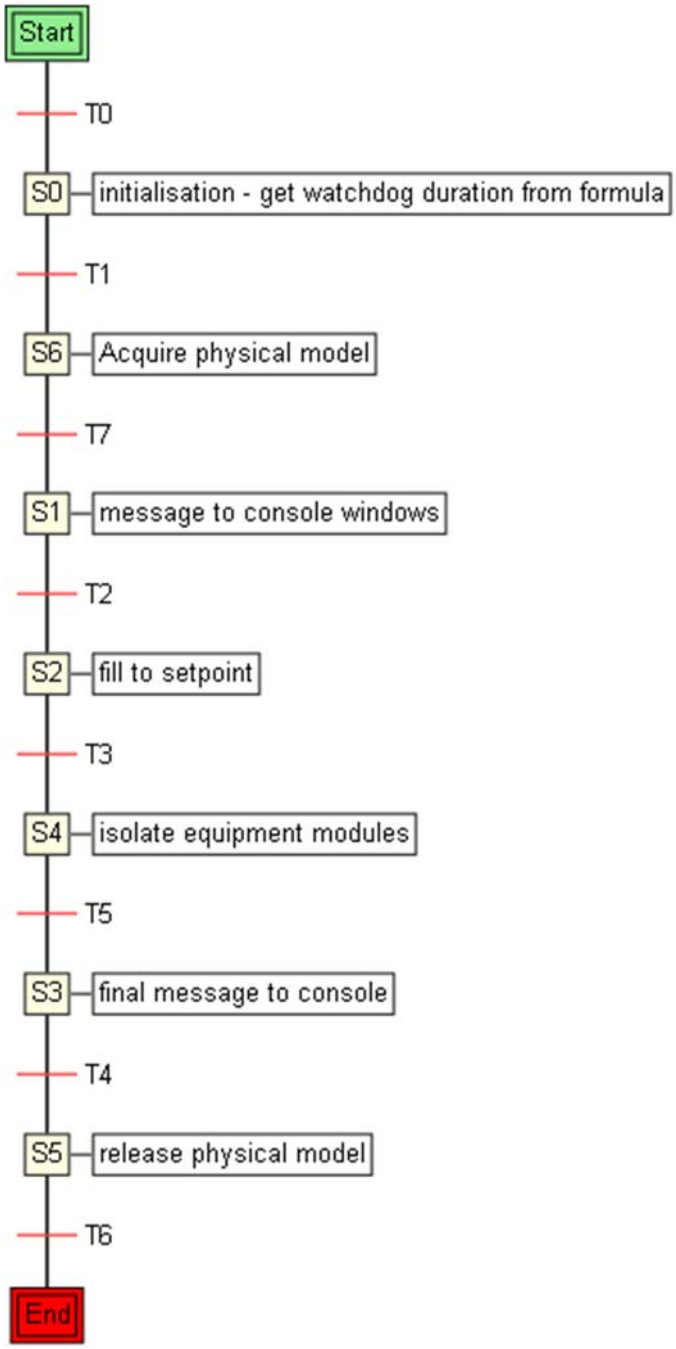
1.1.2.3. T1 Logic

```
##########################################
# Transition: Restarting/T1
##########################################

def main():
    """Main Function"""

    # Transition code...
```

1.1.3. Running



1.1.3.1. S0 Logic

```
##############################################
```

```
    # Step: 7/S0
    ##########################################


    def main():
        """Main Function"""


        # There are a number of requirements that must be met in order to run this phase:
        #
        # 1. It must run on a unit with the following unit parameters:
        # em_vent ;         this is the name of a tank vent equipment module (string)
        # em_tank_fill  ;   this is the name of a tank fill equipment module (string)
        # em_water_drop ;   this is the name of a water supply equipment module that draws
from a water utility distribution header.(string)
        # leveltransmitter; this is the analogue input instrument that reads the level in
the tank. (string)
        # watchdogtimer ;   this is the maximum duration that a fill should take for this
vessel. Phase goes to HELD state, if fill setpoint not achieved in time. (double - secs)
        #
        # 2.In addition, the unit must encompass a number of equipment modules .These
modules are identified through the unit parameters or formula parameters;
        # Here, we have listed out the equipment module names directly in the unit module.
        #
        # 3.The following equipment module types and equipment module methods are required
(if you dont have them already, you will have to create them):
        #
        # em_vent (with 'isolate', 'vent', 'pressurise' methods)
        # em_tank_fill (with 'isolate', 'fill', 'cip' methods)
        # em_water_drop (with 'isolate', 'fill', 'flush' methods)
        #
        # 4. A Formula containing the following parameters must be within campaign scope ;
for example the formula must be associated with this phase or one of its parents.
        #
        # TankFillSetpoint - Fill the tank to this setpoint.(process parameter type, double
value type, units perC)
        # FillWatchdogtimer - For the failure monitor logic; if the tank is not filled
within a particular time, then raise error and drive phase to hold. (step parameter type,
double value type, units secs)
        #
        # 5. The watchdog timer value (in the formula), must be written to a unit parameter
in order for the failure monitor to read it.



        #get formula watchdogtimer and store to unitwatchdog timer


        myformulawatchdogtimer = getStepParameter('FillWatchdogtimer')


        unit.SupportModule.UpdateParameter('watchdogtimer', myformulawatchdogtimer.Value)
```

1.1.3.2. S1 Logic

```
##########################################
# Step: 7/S1
##########################################

def main():
    """Main Function"""

    #get fill setpoint from formula

    fillsetpoint = getProcessParameter('TankFillSetpoint')

    # write to console window

    info('about to start filling tank to setpoint: ' + str(fillsetpoint.Value))
```

1.1.3.3. S2 Logic

```
##########################################
# Step: Running/S2
##########################################

def main():
    """Main Function"""

    # get physical model from unit parameters...

    EMTankInletinstance = unit.SupportModule.PullParameter('em_tank_fill')
    EMProcessAirinstance = unit.SupportModule.PullParameter('em_vent')
    EMWAterDropinstance = unit.SupportModule.PullParameter('em_water_drop')
    unitleveltransitter = unit.SupportModule.PullParameter('leveltransmitter')



    # Confirm user updated Unit parameters with the names of assigned equipment modules
    if EMTankInletinstance.Value == 'placeholder':
        raiseException('Unit parameters not updated from default template values.
Please assign physical model to this unit and update unit parameters with relevant
equipment module names.')

    # setup physical model for a fill

    opc(EMTankInletinstance.Value).GetMethod('fill').Run()

    # sleep command is for demonstration purposes only ..
    sleep(1000)

    opc(EMProcessAirinstance.Value).GetMethod('vent').Run()
```

```
        # sleep command is for demonstration purposes only ..
       sleep(1000)


       opc(EMWAterDropinstance.Value).GetMethod('flush').Run()
        # sleep command is for demonstration purposes only ..
       sleep(1000)


       opc(EMWAterDropinstance.Value).GetMethod('fill').Run()
        # sleep command is for demonstration purposes only ..
       sleep(1000)



       #wait for level in tank to reach setpoint


       # tank level setpoint from formula
       tanklevelsetpoint = getProcessParameter('TankFillSetpoint')


       #use a helper function to wait until tank has reached setpoint


       waitFor(unitleveltransitter.Value, 'Value', Condition.GreaterThanOrEqual,
tanklevelsetpoint.Value)
```

1.1.3.4. S3 Logic

```
##########################################
# Step: Running/S3
##########################################


def main():
    """Main Function"""


    # write to console window


    info('Tank level has reached setpoint')
```

1.1.3.5. S4 Logic

```
##########################################
# Step: Running/S4
##########################################


def main():
    """Main Function"""


    # get physical model from unit parameters...
```

```
    EMTankInletinstance = unit.SupportModule.PullParameter('em_tank_fill')
    EMProcessAirinstance = unit.SupportModule.PullParameter('em_vent')
    EMWAterDropinstance = unit.SupportModule.PullParameter('em_water_drop')
    unitleveltransitter = unit.SupportModule.PullParameter('leveltransmitter')


    # setup physical model for tank isolation


    opc(EMWAterDropinstance.Value).GetMethod('isolate').Run()
    # sleep command is for demonstration purposes only ..
    sleep(1000)


    opc(EMTankInletinstance.Value).GetMethod('isolate').Run()
    # sleep command is for demonstration purposes only ..
    sleep(1000)


    opc(EMProcessAirinstance.Value).GetMethod('isolate').Run()
    # sleep command is for demonstration purposes only ..
    sleep(1000)
```

1.1.3.6. S5 Logic

```
##########################################
# Step: Running/S5
##########################################


def main():
    """Main Function"""


    # get handle to physical model...
    EMTankInletinstance = unit.SupportModule.PullParameter('em_tank_fill')
    EMProcessAirinstance = unit.SupportModule.PullParameter('em_vent')
    EMWaterDropinstance = unit.SupportModule.PullParameter('em_water_drop')
    unitleveltransitter = unit.SupportModule.PullParameter('leveltransmitter')



    opc(EMTankInletinstance.Value).OwnerId = ''
    opc(EMProcessAirinstance.Value).OwnerId = ''
    opc(EMWaterDropinstance.Value).OwnerId = ''
    opc(unitleveltransitter.Value).OwnerId = ''
```

1.1.3.7. S6 Logic

```
##########################################
# Step: Running/S6
##########################################


def main():
    """Main Function"""


    # get handle to physical model...
    EMTankInletinstance = unit.SupportModule.PullParameter('em_tank_fill')
    EMProcessAirinstance = unit.SupportModule.PullParameter('em_vent')
```

```
    EMWaterDropinstance = unit.SupportModule.PullParameter('em_water_drop')
    unitleveltransitter = unit.SupportModule.PullParameter('leveltransmitter')



    opc(EMTankInletinstance.Value).OwnerId = str(this)
    opc(EMProcessAirinstance.Value).OwnerId = str(this)
    opc(EMWaterDropinstance.Value).OwnerId = str(this)
    opc(unitleveltransitter.Value).OwnerId = str(this)
```

1.1.3.8. T0 Logic

```
##########################################
# Transition: 7/T0
##########################################


def main():
    """Main Function"""

    # Transition code...
```

1.1.3.9. T1 Logic

```
##########################################
# Transition: 7/T1
##########################################


def main():
    """Main Function"""

    # Transition code...
```

1.1.3.10. T2 Logic

```
##########################################
# Transition: Running/T2
##########################################


def main():
    """Main Function"""

    # Transition code...
```

1.1.3.11. T3 Logic

```
##########################################
# Transition: Running/T3
##########################################


def main():
```

```
    """Main Function"""

    # Transition code...
```

1.1.3.12. T4 Logic

```
##########################################
# Transition: Running/T4
##########################################

def main():
    """Main Function"""

    # Transition code...
```

1.1.3.13. T5 Logic

```
##########################################
# Transition: Running/T5
##########################################

def main():
    """Main Function"""

    # Transition code...
```

1.1.3.14. T6 Logic

```
##########################################
# Transition: Running/T6
##########################################

def main():
    """Main Function"""

    # Transition code...
```

1.1.3.15. T7 Logic

```
##########################################
# Transition: Running/T7
##########################################

def main():
    """Main Function"""

    # Transition code...
```

1.1.3.16. Failure Monitor Logic

```
   ##########################################
   # Failure Monitor (FM)
   ##########################################


   def main():
       """Main Function"""


       #Failure monitor would normally monitor CM & EM alarms and watchdog timeouts.It can
drive the phase into the HELD state based on a failure.
       #Enter alarm monitoring logic here
       #
       #if OPC(module).IsInAlarm:
       #   phase.Hold()
       #
       #Failure Monitor timer watchdog
       #Fillwatchdogtimer - for the failure monitor logic ; if the tank is not filled
within a particular time, then raise error and drive phase to hold
       #Get watchdogtimervalue from formula


       watchdogtimerduration = unit.SupportModule.PullParameter('watchdogtimer')


       #Create a free running timer of 1200 seconds duration. The held logic will hold the
timer. The restart logic will reset if phase goes to HELD


       if firstScan:
       # Create the timer once only.
           mytimer = unit.SupportModule.NewTimer('FailureMonitorTimer', 1200)
           mytimer.Start()
           info('failure monitor watchdog timer started')


       mytimer = unit.SupportModule.GetTimer('FailureMonitorTimer')


       # Print out watchdog timer value  - Helpful for troubleshooitng. If this detail is
not required and the console window is unnecessarily cluttered, then just delete.
       if firstScan == False:
           info('timer value:'+str(mytimer.Value))


       # drive the running phase to the HELD state, if the step has been running longer
than the watchdog timeout
       if (mytimer.Value > watchdogtimerduration.Value) and (phase.CurrentState ==
State.Running):
           info('phase failure')
           phase.Hold()
```

2. Formula Name

Formula Name
myfillmformula