Spike

PH_XferIn

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

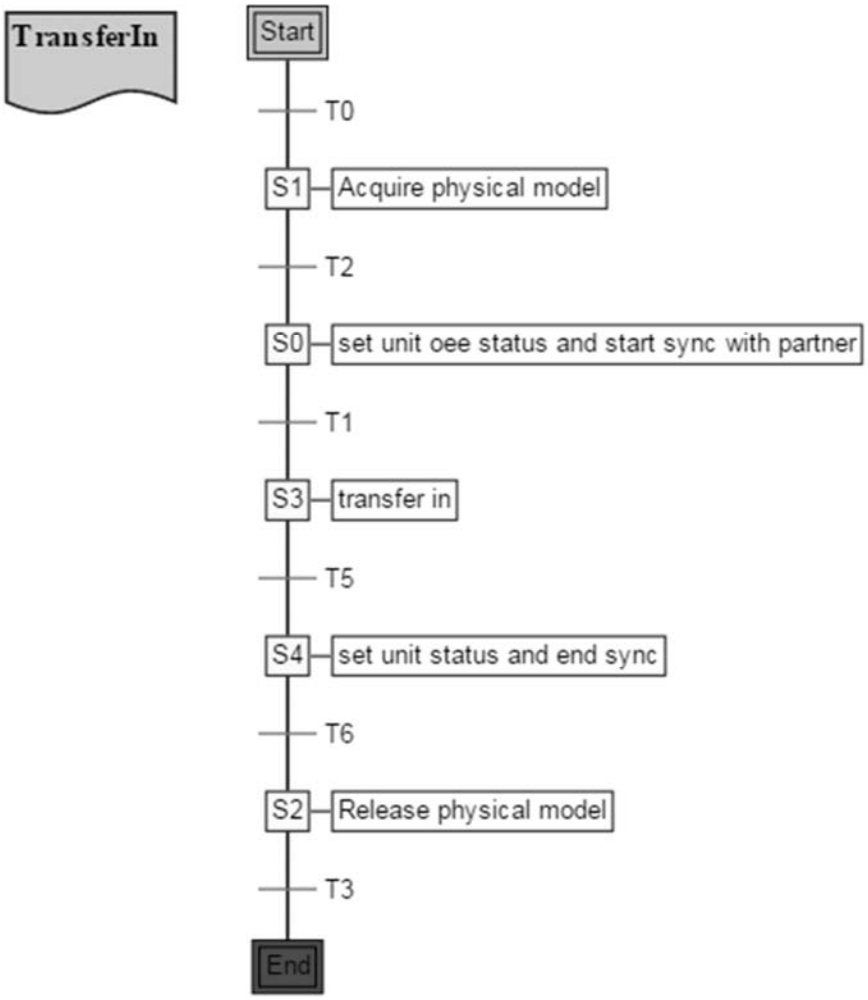
Table of Contents

Reference Documentation

Document	Ref.	Title

1. PH_XferIn

1.1.1. Running



1.1.1.1. S0 Logic

```
##########################################
# Step: Running/S0
##########################################
```

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


        # Set OEE status
        unit.SetOEEState(OEEStatus.Production)


        # Set sync partner
        partnerUnit = unit.SupportModule.PullParameter('PartnerUnit')
        unit.SupportModule.SetSyncPartner(partnerUnit.Value)


        # Wait for system to sync both phases
        unit.SupportModule.WaitForSync()


        # Set sync message
        unit.SupportModule.SetSyncMessage(SyncMessage.ReadyRestart)
```

1.1.1.2. S1 Logic

```
    ##########################################
    # Step: Running/S1
    ##########################################


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


        # Step code...
        # 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:
        #       PartnerUnit:    is the name of the partner unit for the synchronised transfer
        #       Watchdog:       is the duratino of the watchdog timer.
        #    In addition, the unit may encompass a number of equipment modules that are
identified through the unit parameters or formula parameters;
        # 2. A Formula containing the following parameters must be within campaign scope
        #       PhaseWatchdog:  For failure monitor logic, if tank is not filled within a
particular time, then raise error and drive phase to hold. (step parameter type, double value
type, units secs)
        # 3. The watchdog timer value (in the formula), must be written to a unit parameter in
order for the failure monitor to read it.


        # Read formula parameter
        phaseWatchdogTimer = getStepParameter('PhaseWatchdog')


        # Update unit parameter
        unit.SupportModule.UpdateParameter('Watchdog', phaseWatchdogTimer.Value)
```

1.1.1.3. S2 Logic

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


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

```
    # Release physical model
    info('Releasing equipment modules and control modules')
```

1.1.1.4. S3 Logic

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

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

    # manipulate the physical model in order to transfer into the tank
    info('Transferring in to the unit')

    # sleep is for demonstration purposes only!
    sleep(40000)
```

1.1.1.5. S4 Logic

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

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

    # Wait for partner phase to stop transfer.
    unit.SupportModule.WaitForMessage(SyncMessage.TransferComplete)

    # manipulate physical model to end the transfer in



    # Set OEE status
    unit.SetOEEState(OEEStatus.Idle)
```

1.1.1.6. T0 Logic

```
##########################################
# Transition: Running/T0
##########################################

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

    # Transaction code...
```

1.1.1.7. T1 Logic

```
##########################################
# Transition: Running/T1
##########################################


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


    # Transaction code...
```

1.1.1.8. T2 Logic

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


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


    # Transition code...
```

1.1.1.9. T3 Logic

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


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


    # Transition code...
```

1.1.1.10. T5 Logic

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


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


    # Transaction code...
```

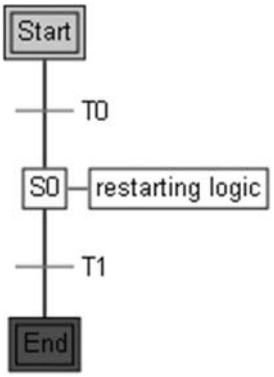
1.1.1.11. T6 Logic

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

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

    # Transaction code...
```

1.1.2. Restarting



1.1.2.1. S0 Logic

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

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

    # Reset timer
    monitorTimer = unit.SupportModule.GetTimer('FailureMonitorTimer')
    monitorTimer.Reset()
```

1.1.2.2. T0 Logic

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

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

    # Transition code...
```

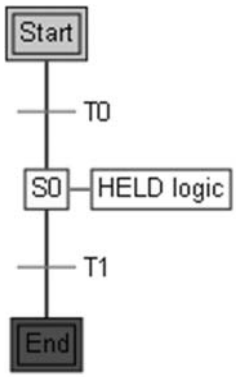
1.1.2.3. T1 Logic

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

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

    # Transition code...
```

1.1.3. Held



1.1.3.1. S0 Logic

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

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

    # Reset sync partner
    unit.SupportModule.ResetSync()

    # Set OEE state
    unit.SetOEEState(OEEStatus.QA_Held)
```

1.1.3.2. T0 Logic

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


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


    # Transition code...
```

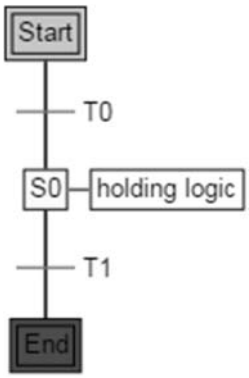
1.1.3.3. T1 Logic

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


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


    # Transition code...
```

1.1.4. Holding



1.1.4.1. S0 Logic

```
##########################################
# Step: Holding/S0
##########################################


def main():
```

```
    """Main Function"""

    #holding logic
```

1.1.4.2. T0 Logic

```
##########################################
# Transition: Holding/T0
##########################################

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

    # Transition code...
```

1.1.4.3. T1 Logic

```
##########################################
# Transition: Holding/T1
##########################################

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

    # Transition code...
```

1.1.4.4. Failure Monitor Logic

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

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

    # Failure monitor would normally monitor CM & EM alarms & watchdog timeouts. It can
drive the phase into the HELD state based on a failure

    # Enter alarm monitoring logic here
    # for eg.. if opc(module).IsInAlarm:
    #          phase.Hold()

    # Get watchdog timer from formula
    watchdogTimer = unit.SupportModule.PullParameter('Watchdog')

    # Create timer on initial scan
    if firstScan:
        monitorTimer = unit.SupportModule.NewTimer('FailureMonitorTimer', 1200)
        monitorTimer.Start()
        info(phase.Name + str(' failure monitor watchdog timer started'))
```

```
        # Get timer from unit support module
        monitorTimer = unit.SupportModule.GetTimer('FailureMonitorTimer')


        # Log watchdog timer value every 5 seconds, helpful for troubleshooting
        if firstScan == False and monitorTimer.Value % 5 == 0:
            info(phase.Name + ' watchdog timer value = ' + str(monitorTimer.Value))


        # If phase is running & failure monitor running longer than watchdog timer, drive the
running phase to the HELD state
        if phase.CurrentState == State.Running and monitorTimer.Value > watchdogTimer.Value:
            info(phase.Name + ' failure')
            phase.Hold()
```

2. Formula Name

Formula Name