# Vertical Warehouses Software description

## Message handling (CommandReceiveTaskFunction)

### FiniteStateMachine

1. Homing
   1. HomingStateMachine.Start
2. Stop
   1. CurrentStateMachine.Stop
3. ShutterPositioning
   1. ShutterPositioningStateMachine.Start
4. ShutterControl
   1. ShutterControlStateMachine.Start
5. Positioning
   1. PositioningStateMachine.Start
6. SensorChanged
   1. Publish FieldCommandMessage (InverterStatusUpdateFieldMessageData(SensorStatus = true), InverterDriver, InverterStatusUpdate)
   2. Publish FieldCommandMessage (SensorsChangedFieldMessageData , IoDriver, SensorsChanged)
7. CheckCondition
   1. Publish NotificationMessage( Any, CheckCondition, OperationEnd)
8. DrawerOperation
9. ResetSecurity
   1. Publish FieldCommandMessage (ResetSecurityFieldMessageData, IoDriver, ResetSecurity)
   2. Publish NotificationMessage( Any, CheckCondition, OperationEnd / OperationStop / OperationError)
10. InverterStop

### InverterDriver

1. CalibrateAxis
   1. If (InverterStarted) CalibrateAxisStateMachine.Start
   2. Else PowerOnStateMachine.Start
2. InverterPowerOff
   1. If (InverterStarted) PowerOffStateMachine.Start
   2. Else Publish Notification (InverterPowerOnFieldMessageData, powerOffData.InverterToPowerOff,

FieldMessageType.InverterPowerOff,

MessageStatus.OperationEnd)

1. InverterPowerOn
   1. If (InverterStarted) Publish Notification (InverterPowerOnFieldMessageData, powerOnData.InverterToPowerOn,

FieldMessageType.InverterPowerOn,

MessageStatus.OperationEnd)

* 1. Else PowerOnStateMachine.Start

1. Positioning
   1. If (IsInverterStarted)
      1. If(MovementType == Absolute)
         1. If(currentPosition == positioningFieldData.TargetPosition) Publish Notification (

FieldMessageType.Positioning,

MessageStatus.OperationEnd)

* + - 1. Else PositioningStateMachine.Start
    1. Else if (MovementType == Relative)
       1. If (TargetPosition == 0 ) Publish Notification (

FieldMessageType.Positioning,

MessageStatus.OperationEnd)

* + - 1. Else PositioningStateMachine.Start
  1. Else PowerOnStateMachine.Start (receivedMessage)

1. ShutterPositioning
   1. If (IsInverterPoweredOn) PowerOffStateMachine.Start (receivedMessage)
   2. Else ShutterPositioningStateMachine.Start
2. InverterStatusUpdate
   1. If( SensorStatus == true)
      1. If( SensorUpdateInterval == 0 )

Sends InverterMessage(MainInverter, DigitalInputsOutputs)

* + - 1. Else Change sensorStatusUpdateTimer(SensorUpdateInterval)
    1. Else Change sensorStatusUpdateTimer(-1)
  1. If( AxisPosition== true)
     1. If( AxisUpdateInterval== 0 )

Sends InverterMessage(MainInverter, ActualPositionShaft)

* + - 1. Else Change axisPositionUpdateTimer(AxisUpdateInterval)
    1. Else Change axisPositionUpdateTimer (infinite)

1. InverterSwitchOff
   1. SwitchOffStateMachine.Start
2. InverterSwitchOn
   1. If( IsReadyToSwitchOn && IsVoltageEnabled && IsQuickStopTrue)
      1. If( AxisToSwitchOn == message.AxisToSwitchOn)
         1. If( IsSwitchedOn) Publish Notification (InverterSwitchedOnFieldMessageData, AxisToSwitchOn,

FieldMessageType.InverterSwitchOn,

MessageStatus.OperationEnd)

* + - 1. Else SwitchOnStateMachine.Start
    1. Else SwitchOffStateMachine.Start
  1. Else PowerOnStateMachine.Start (receivedMessage)

1. InverterStop
   1. StopStateMachine.Start

### IoDriver

## State Machines

The calling object( example: HostedInverterDriver) stores the state machine in the property

CurrentStateMachine

and invokes the Start method.

Each State Machine has a property that stores the

CurrentState

and each state has a Start method.

### FiniteStateMachines: ShutterControlStateMachine

1. ShutterControlStartState
   1. Sends InverterMessage(ShutterPositionFieldMessageData(Opened, Up), ShutterPositioning)
   2. Publish Notification (ShutterControlMessageData,

MessageType.ShutterControl,

MessageStatus.OperationStart)

* 1. ProcessFieldNotificationMessage(OperationEnd)
     1. ChangeState(ShutterControlOpenState)

1. ShutterControlOpenState
   1. Sends InverterMessage(ShutterPositionFieldMessageData(Closed, Down), ShutterPositioning)
   2. Publish Notification (ShutterControlMessageData,

MessageType.ShutterControl,

MessageStatus.OperationExecuting)

* 1. ProcessFieldNotificationMessage(OperationEnd)
     1. ChangeState(ShutterControlCloseState)

1. ShutterControlCloseState
   1. Sends InverterMessage(ShutterPositionFieldMessageData(Opened, Up), ShutterPositioning)
   2. ProcessFieldNotificationMessage(OperationEnd)
      1. ChangeState(ShutterControlOpenState)

### InverterDriver: CalibrateAxisStateMachine

1. CalibrateAxisStartState ( by HostedInverterDriver)
   1. Sends InverterMessage(SystemIndex, SetOperatingModeParam, Homing)
   2. Publish Notification (CalibrateAxisFieldMessageData,

FieldMessageType.CalibrateAxis,

MessageStatus.OperationStart)

* 1. ProcessFieldNotificationMessage – not defined
  2. The process FiniteStateMachines receives the notification and ignores it.
  3. The process HostedInverterDriver handles the message from the inverter:
  4. ReceiveInverterData.EvaluateWriteMessage.CurrentStateMachine.ValidateCommandMessage
     1. ChangeState(CalibrateAxisEnableOperationState)

1. CalibrateAxisEnableOperationState
   1. Sends InverterMessage(SystemIndex, ControlWordParam, EnableOperation = true)
   2. The process HostedInverterDriver handles the message from the inverter:
   3. ReceiveInverterData.EvaluateReadMessage.CurrentStateMachine.ValidateCommandResponse
      1. If(message.IsError)

ChangeState(CalibrateAxisErrorState)

* + 1. If(CommonStatusWord.IsOperationEnabled)

ChangeState(CalibrateAxisStartHomingState)

1. CalibrateAxisStartHomingState
   1. Sends InverterMessage(SystemIndex, ControlWordParam, HomingOperation = true)
   2. The process HostedInverterDriver handles the message from the inverter:
   3. ReceiveInverterData.EvaluateReadMessage.CurrentStateMachine.ValidateCommandResponse
      1. If(message.IsError)

ChangeState(CalibrateAxisErrorState)

* + 1. If(axisToCalibrate == Axis.Horizontal && HomingStatusWord.HomingAttained == true) ChangeState(CalibrateAxisDisableOperationState)

1. CalibrateAxisDisableOperationState
   1. Sends InverterMessage(SystemIndex, ControlWordParam, HomingOperation = false, EnableOperation = false)
   2. The process HostedInverterDriver handles the message from the inverter:
   3. ReceiveInverterData.EvaluateReadMessage.CurrentStateMachine.ValidateCommandResponse
      1. If(message.IsError)

ChangeState(CalibrateAxisErrorState)

* 1. If(CommonStatusWord.IsOperationEnabled == false) ChangeState(CalibrateAxisEndState)

1. CalibrateAxisEndState
   1. Publish Notification (CalibrateAxisFieldMessageData,

FieldMessageType.CalibrateAxis,

MessageStatus.OperationEnd)

* 1. ProcessFieldNotificationMessage – not defined
  2. The process FiniteStateMachines receives the notification:
     1. HomingStateMachine: trace

1. The process HostedInverterDriver handles the Notification:
   1. CurrentStateMachine = null

### InverterDriver: PowerOnStateMachine

1. PowerOnStartState ( by HostedInverterDriver)
   1. Sends InverterMessage(SystemIndex, ControlWordParam, EnableVoltage = true, QuickStop = true)
   2. Publish Notification (InverterPowerOnFieldMessageData,

FieldMessageType.InverterPowerOn,

MessageStatus.OperationStart)

* 1. ReceiveInverterData.EvaluateReadMessage.CurrentStateMachine.ValidateCommandResponse
     1. If(message.IsError)

ChangeState(PowerOnErrorState)

* + 1. If(CommonStatusWord.IsVoltageEnabled && IsQuickStopTrue && IsReadyToSwitchOn)

ChangeState(PowerOnSwitchOnState)

1. PowerOnSwitchOnState