Updated: Tuesday, August 3rd 2021

Version 4.1

Application Programming Interface: Python v4.1

MachineMotion

Overview

This is the developers reference manual for version 4.1 of MachineMotion's Python API. This API is compatible with Python version 3.6 and later.

If this is your first time using the MachineMotion controller, please follow the Python programming manual here.

Version 4.1 of the API is compatible with MachineMotion controller software version 1.12 and later. If your controller software is 1.2.11 or earlier, please refer to Python API v2.2.

To get your controller software version, refer to the user manual here.

Download

The Python API comes pre-installed on your MachineMotion controller. To update the API verison or download it to an external computer, follow this link to version 4.1 of the Python API on Vention's Github.

MachineMotion v2 One-Drive

For use with MachineMotion 2 One-Drive (CE-CL-010-0001) please refer to the provided example which can be found here

API Reference

bindeStopEvent (callback_function)

Compatibility: MachineMotion v1 & MachineMotion v2

Description

Configures a user defined function to execute immediately after an E-stop event.

Parameters

- **callback_function** (Required) function - The function to be executed after an e-stop is triggered or released.

Return Value

None

MachineMotion V1 Example Code

```
#!/usr/bin/python
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases how to control eStop on MachineMotion v1. ###

### MachineMotion CONFIGURATION ###

# Create MachineMotion instance
mm = MachineMotion()
time.sleep(0.1) # Wait to initialize internal eStop topics.

# Define a callback to process estop status
def templateCallback(estop_status):
    print("eStop status is : " + str(estop_status))
    if estop_status == True :
        # executed when you enter estop
        print("MachineMotion is in estop state.")
    elif estop_status == False :
        # executed when you exit estop
        print("MachineMotion is not in estop state.")
mm.bindeStopEvent(templateCallback)

### ESTOP TRIGGER ###

result = mm.triggerEstop()
if result == True   : print("--> Software stop triggered")
else                : print("--> Failed to trigger software stop")

time.sleep(3.0)

### ESTOP RELEASE ###

result = mm.releaseEstop()
if result == True   : print("--> Software stop released")
else                : print("--> Failed to release software stop")

time.sleep(3.0)

### SYSTEM RESET ###

result = mm.resetSystem()
if result == True   : print("--> System has been reset")
else                : print("--> Failed to reset system")

print("--> Example completed")
```

MachineMotion V2 Example Code

```
#!/usr/bin/python
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases how to control eStop on MachineMotion v2. ###

### MachineMotion CONFIGURATION ###

# Create MachineMotion instance
mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)
time.sleep(0.1) # Wait to initialize internal eStop topics.

# Define a callback to process estop status
def templateCallback(estop_status):
    print("eStop status is : " + str(estop_status))
    if estop_status == True :
        # executed when you enter estop
        print("MachineMotion is in estop state.")
    elif estop_status == False :
        # executed when you exit estop
        print("MachineMotion is not in estop state.")
mm.bindeStopEvent(templateCallback)

### ESTOP TRIGGER ###

result = mm.triggerEstop()
if result == True   : print("--> Software stop triggered")
else                : print("--> Failed to trigger software stop")

time.sleep(3.0)

### ESTOP RELEASE ###

result = mm.releaseEstop()
if result == True   : print("--> Software stop released")
else                : print("--> Failed to release software stop")

time.sleep(3.0)

### SYSTEM RESET ###

result = mm.resetSystem()
if result == True   : print("--> System has been reset")
else                : print("--> Failed to reset system")

print("--> Example completed")
```

configAxis (axis,uStep,mechGain)

Compatibility: MachineMotion V1 only

Description

Configures motion parameters for a single axis on the MachineMotion v1.

Parameters

- **axis** (Required) Number - The axis to configure.
- **uStep** (Required) Number - The microstep setting of the axis.
- **mechGain** (Required) Number - The distance moved by the actuator for every full rotation of the stepper motor, in mm/revolution.

Return Value

None

NOTE: The uStep setting is hardcoded into the machinemotion controller through a DIP switch and is by default set to 8. The value here must match the value on the DIP Switch.

MachineMotion V1 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example configures an actuator for a MachineMotion v1. ###

mm = MachineMotion()

# Configure the axis number 1, 8 uSteps and 150 mm / turn for a timing belt
axis = AXIS_NUMBER.DRIVE1
uStep = MICRO_STEPS.ustep_8
mechGain = MECH_GAIN.timing_belt_150mm_turn
mm.configAxis(axis, uStep, mechGain)
print("Axis " + str(axis) + " configured with " + str(uStep) + " microstepping and " + str(mechGain) + "mm/turn mechanical gain")
```

configAxisDirection (axis,direction)

Compatibility: MachineMotion V1 only

Description

Configures a single axis to operate in either clockwise (normal) or counterclockwise (reverse) mode. Refer to the Automation System Diagram for the correct axis setting.

Parameters

- **axis** (Required) Number - The specified axis.
- **direction** (Required) String - A string from the DIRECTION class. Either 'DIRECTION.NORMAL' or 'DIRECTION.REVERSE'. Normal direction means the axis will home towards end stop sensor A and reverse will make the axis home towards end stop B.

Return Value

None

MachineMotion V1 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example configures actuator direction for MachineMotion v1. ###

mm = MachineMotion()

#When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Configure the axis number 1, 8 uSteps and 150 mm / turn for a timing belt
axis = AXIS_NUMBER.DRIVE1
uStep = MICRO_STEPS.ustep_8
mechGain = MECH_GAIN.timing_belt_150mm_turn
mm.configAxis(axis, uStep, mechGain)

homesTowards = {
    DIRECTION.NORMAL: "sensor " + str(axis) + "A",
    DIRECTION.REVERSE: "sensor " + str(axis) + "B"
}

# Change axis direction, and see how it affects subsequent moves
direction = DIRECTION.REVERSE
mm.configAxisDirection(axis, direction)
print("Axis " + str(axis) + " is set to " + direction + " mode. It will now home towards " + homesTowards[direction] + "." )
mm.emitHome(axis)

direction = DIRECTION.NORMAL
mm.configAxisDirection(axis, direction)
print("Axis " + str(axis) + " is set to " + direction + " mode. It will now home towards " + homesTowards[direction] + "." )
mm.emitHome(axis)
```

configHomingSpeed (axes,speeds,units)

Compatibility: MachineMotion v1 & MachineMotion v2

Description

Sets homing speed for all selected axes.

Parameters

- **axes** (Required) List of Numbers - A list of the axes to configure. ex - [1,2,3]
- **speeds** (Required) List of Numbers - A list of homing speeds to set for each axis. ex - [50, 50, 100]
- **units** (Optional. Default = UNITS_SPEED.mm_per_sec) String - Units for speed. Can be switched to UNITS_SPEED.mm_per_min

Return Value

None

NOTE: Once set, the homing speed will apply to all programs, including MachineLogic applications.

MachineMotion V1 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example configures homing speed for MachineMotion v1. ###

mm = MachineMotion()

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

### Configuring ###

axes = [1,2]
homingSpeeds = [50,100] # The homing speeds to set for each axis, in mm/sec

mm.configAxis(1, MICRO_STEPS.ustep_8, MECH_GAIN.timing_belt_150mm_turn)
mm.configAxis(2, MICRO_STEPS.ustep_8, MECH_GAIN.timing_belt_150mm_turn)
mm.configHomingSpeed(axes, homingSpeeds)    # Sets homing speeds for all selected axes.

### Testing the configuration ###

axis = 1    # The axis to move

print("Moving axis " + str(axis) + " by 100mm.")
mm.emitRelativeMove(axis, DIRECTION.POSITIVE, 100)
mm.waitForMotionCompletion()

#Homes the axis at the newly configured homing speed.
print("Homing axis " + str(axis))
mm.emitHome(axis)
```

MachineMotion V2 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example configures homing speed for MachineMotion v2. ###

mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

### Configuring ###

axes = [1,2]
homingSpeeds = [50,100] # The homing speeds to set for each axis, in mm/sec

mm.configServo(1, MECH_GAIN.timing_belt_150mm_turn, DIRECTION.NORMAL, 5.0)
mm.configServo(2, MECH_GAIN.timing_belt_150mm_turn, DIRECTION.NORMAL, 5.0)
mm.configHomingSpeed(axes, homingSpeeds)    # Sets homing speeds for all selected axes.

### Testing the configuration ###

axis = 1    # The axis to move

print("Moving axis " + str(axis) + " by 100mm.")
mm.emitRelativeMove(axis, DIRECTION.POSITIVE, 100)
mm.waitForMotionCompletion()

#Homes the axis at the newly configured homing speed.
print("Homing axis " + str(axis))
mm.emitHome(axis)
```

configServo (drive,mechGain,direction,motorCurrent,tuningProfile)

Compatibility: MachineMotion V1 only

Description

Configures motion parameters as a servo motor, for a single drive on the MachineMotion v2.

Parameters

- **drive** (Required) Number - The drive to configure.
- **mechGain** (Required) Number - The distance moved by the actuator for every full rotation of the stepper motor, in mm/revolution.
- **direction** (Required) String from DIRECTION class - The direction of the axis
- **motorCurrent** (Required) Number - The current to power the motor with, in Amps.
- **tuningProfile** (Required) String - The tuning profile of the smartDrive.

Return Value

None

NOTE: Warning, changing the configuration can de-energize motors and thus cause unintended behaviour on vertical axes.

MachineMotion V2 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example configures actuators for a MachineMotion v2. ###

mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Configure a timing belt actuator in servo mode, on drive 1
drive = 1
mechGain = MECH_GAIN.timing_belt_150mm_turn
direction = DIRECTION.NORMAL
motorCurrent = 5.0 # Amperes
mm.configServo(drive, mechGain, direction, motorCurrent)

# Configure an electric cylinder actuator in stepper mode, on drive 2
drive = 2
mechGain = MECH_GAIN.electric_cylinder_mm_turn
direction = DIRECTION.NORMAL
motorCurrent = 1.6 # Amperes
mm.configStepper(drive, mechGain, direction, motorCurrent) # Microsteps will default to 8.
# If you need to use a different microstepping :
#mm.configStepper(drive, mechGain, direction, motorCurrent, microSteps = MICRO_STEPS.ustep_4)
```

configStepper (drive,mechGain,direction,motorCurrent,microSteps)

Compatibility: MachineMotion V1 only

Description

Configures motion parameters as a stepper motor, for a single drive on the MachineMotion v2.

Parameters

- **drive** (Required) Number - The drive to configure.
- **mechGain** (Required) Number - The distance moved by the actuator for every full rotation of the stepper motor, in mm/revolution.
- **direction** (Required) String from DIRECTION class - The direction of the axis
- **motorCurrent** (Required) Number - The current to power the motor with, in Amps.
- **microSteps** (Required) Number from MICRO_STEPS class - The microstep setting of the drive.

Return Value

None

NOTE: Warning, changing the configuration can de-energize motors and thus cause unintended behaviour on vertical axes.

MachineMotion V2 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example configures actuators for a MachineMotion v2. ###

mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Configure a timing belt actuator in servo mode, on drive 1
drive = 1
mechGain = MECH_GAIN.timing_belt_150mm_turn
direction = DIRECTION.NORMAL
motorCurrent = 5.0 # Amperes
mm.configServo(drive, mechGain, direction, motorCurrent)

# Configure an electric cylinder actuator in stepper mode, on drive 2
drive = 2
mechGain = MECH_GAIN.electric_cylinder_mm_turn
direction = DIRECTION.NORMAL
motorCurrent = 1.6 # Amperes
mm.configStepper(drive, mechGain, direction, motorCurrent) # Microsteps will default to 8.
# If you need to use a different microstepping :
#mm.configStepper(drive, mechGain, direction, motorCurrent, microSteps = MICRO_STEPS.ustep_4)
```

detectIOModules ( )

Compatibility: MachineMotion v1 & MachineMotion v2

Description

Returns a dictionary containing all detected IO Modules.

Return Value

- Dictionary with keys of format “Digital IO Network Id [id]” and values [id] where [id] is the network IDs of all connected digital IO modules.

NOTE: For more information, please see the digital IO datasheet here

---

MachineMotion V1 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example reads digital inputs for MachineMotion v1. ###

mm = MachineMotion()

# Detect all connected digital IO Modules
detectedIOModules = mm.detectIOModules()

# Read and print all input values
if detectedIOModules is not None :
  for IO_Name, IO_NetworkID in detectedIOModules.items():
    readPins = [0, 1, 2, 3]
    for readPin in readPins:
        pinValue = mm.digitalRead(IO_NetworkID, readPin)
        print("Pin " + str(readPin) + " on " + IO_Name + " has value " + str(pinValue))
```

MachineMotion V2 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example reads digital inputs for MachineMotion v2. ###

mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)

# Detect all connected digital IO Modules
detectedIOModules = mm.detectIOModules()

# Read and print all input values
if detectedIOModules is not None :
  for IO_Name, IO_NetworkID in detectedIOModules.items():
    readPins = [0, 1, 2, 3]
    for readPin in readPins:
        pinValue = mm.digitalRead(IO_NetworkID, readPin)
        print("Pin " + str(readPin) + " on " + IO_Name + " has value " + str(pinValue))
```

digitalRead (deviceNetworkId,pin)

Compatibility: MachineMotion v1 & MachineMotion v2

Description

Reads the state of a digital IO modules input pins.

Parameters

- **deviceNetworkId** (Required) Integer - The IO Modules device network ID. It can be found printed on the product sticker on the back of the digital IO module.

- **pin** (Required) Integer - The index of the input pin.

Return Value

- Returns 1 if the input pin is logic HIGH (24V) and returns 0 if the input pin is logic LOW (0V).

NOTE: On older digital IO modules, the pin labels on the digital IO module (pin 1, pin 2, pin 3, pin 4) correspond in software to (0, 1, 2, 3). Therefore, digitalRead(deviceNetworkId, 2) will read the value on input pin 3.

MachineMotion V1 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example reads digital inputs for MachineMotion v1. ###

mm = MachineMotion()

# Detect all connected digital IO Modules
detectedIOModules = mm.detectIOModules()

# Read and print all input values
if detectedIOModules is not None :
   for IO_Name, IO_NetworkID in detectedIOModules.items():
      readPins = [0, 1, 2, 3]
      for readPin in readPins:
          pinValue = mm.digitalRead(IO_NetworkID, readPin)
          print("Pin " + str(readPin) + " on " + IO_Name + " has value " + str(pinValue))
```

MachineMotion V2 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example reads digital inputs for MachineMotion v2. ###

mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)

# Detect all connected digital IO Modules
detectedIOModules = mm.detectIOModules()

# Read and print all input values
if detectedIOModules is not None :
   for IO_Name, IO_NetworkID in detectedIOModules.items():
      readPins = [0, 1, 2, 3]
      for readPin in readPins:
          pinValue = mm.digitalRead(IO_NetworkID, readPin)
          print("Pin " + str(readPin) + " on " + IO_Name + " has value " + str(pinValue))
```

digitalWrite (deviceNetworkId,pin,value)

Compatibility: MachineMotion v1 & MachineMotion v2

Description

Sets voltage on specified pin of digital IO output pin to either logic HIGH (24V) or LOW (0V).

Parameters

- **deviceNetworkId** (Required) Integer - The IO Modules device network ID. It can be found printed on the product sticker on the back of the digital IO module.
- **pin** (Required) Integer - The output pin number to write to.
- **value** (Required) Integer - Writing '1' or HIGH will set digial output to 24V, writing 0 will set digital output to 0V.

Return Value

None

NOTE: Output pins maximum sourcing current is 75 mA and the maximum sinking current is 100 mA. On older digital IO modules, the pin labels on the digital IO module (pin 1, pin 2, pin 3, pin 4) correspond in software to (0, 1, 2, 3). Therefore, digitalWrite(deviceNetworkId, 2, 1) will set output pin 3 to 24V.

MachineMotion V1 Example Code

```
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example writes digital outputs for MachineMotion v1. ###

mm = MachineMotion()

# Detect all connected digital IO Modules
detectedIOModules = mm.detectIOModules()

# Toggles the output pins
if detectedIOModules is not None :
    for IO_Name, IO_NetworkID in detectedIOModules.items():
        writePins = [0, 1, 2, 3]
        for writePin in writePins :
            print("Pin " + str(writePin) + " on " + IO_Name + " is going to flash twice")

            mm.digitalWrite(IO_NetworkID, writePin, 1)
            time.sleep(1)
            mm.digitalWrite(IO_NetworkID, writePin, 0)
            time.sleep(1)
            mm.digitalWrite(IO_NetworkID, writePin, 1)
            time.sleep(1)
            mm.digitalWrite(IO_NetworkID, writePin, 0)
            time.sleep(1)
```

MachineMotion V2 Example Code

```
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example writes digital outputs for MachineMotion v2. ###

mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)

# Detect all connected digital IO Modules
detectedIOModules = mm.detectIOModules()

# Toggles the output pins
if detectedIOModules is not None :
    for IO_Name, IO_NetworkID in detectedIOModules.items():
        writePins = [0, 1, 2, 3]
        for writePin in writePins :
            print("Pin " + str(writePin) + " on " + IO_Name + " is going to flash twice")

            mm.digitalWrite(IO_NetworkID, writePin, 1)
            time.sleep(1)
            mm.digitalWrite(IO_NetworkID, writePin, 0)
            time.sleep(1)
            mm.digitalWrite(IO_NetworkID, writePin, 1)
            time.sleep(1)
            mm.digitalWrite(IO_NetworkID, writePin, 0)
            time.sleep(1)
```

emitAbsoluteMove (axis,position)

Compatibility: MachineMotion v1 & MachineMotion v2

Description

Moves the specified axis to a desired end location.

Parameters

- **axis** (Required) Number - The axis which will perform the absolute move command.
- **position** (Required) Number - The desired end position of the axis movement.

Return Value

None

MachineMotion V1 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases absolute moves with MachineMotion v1. ###

mm = MachineMotion()

#When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Axis configuration
axis = 1            #The axis that you'd like to move
mm.configAxis(axis, MICRO_STEPS.ustep_8, MECH_GAIN.timing_belt_150mm_turn)

# Movement configuration
position = 100      #The absolute position you'd like to move to

# Home Axis before absolute move
print("Axis " + str(axis) + " is going home.")
mm.emitHome(axis)
print("Axis " + str(axis) + " homed.")

# Move
mm.emitAbsoluteMove(axis, position)
print("Axis " + str(axis) + " is moving towards position " + str(position) + "mm.")
mm.waitForMotionCompletion()
print("Axis " + str(axis) + " is at position " + str(position) + "mm.")
```

MachineMotion V2 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases absolute moves with MachineMotion v2. ###

mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Axis configuration
axis = 1            #The axis that you'd like to move
mm.configServo(axis, MECH_GAIN.timing_belt_150mm_turn, DIRECTION.NORMAL, 5.0)

# Movement configuration
position = 100      #The absolute position you'd like to move to

# Home Axis before absolute move
print("Axis " + str(axis) + " is going home.")
mm.emitHome(axis)
print("Axis " + str(axis) + " homed.")

# Move
mm.emitAbsoluteMove(axis, position)
print("Axis " + str(axis) + " is moving towards position " + str(position) + "mm")
mm.waitForMotionCompletion()
print("Axis " + str(axis) + " is at position " + str(position) + "mm")
```

emitAcceleration (mm_per_sec_sqr,units)

Compatibility: MachineMotion v1 & MachineMotion v2

Description

Sets the global acceleration for all movement commands on all axes.

Parameters

- **mm_per_sec_sqr** (Required) Number - The global acceleration in mm/s^2.
- **units** (Optional. Default = UNITS_ACCEL.mm_per_sec_sqr) String - Units for speed. Can be switched to UNITS_ACCEL.mm_per_min_sqr

Return Value

None

MachineMotion V1 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example configures system acceleration for MachineMotion v1. ###

mm = MachineMotion()

acceleration = 500      # The max acceleration [mm/s^2] that all subsequent moves will move at
mm.emitAcceleration(acceleration)
print("Global acceleration set to " + str(acceleration) + "mm/s^2.")
```

MachineMotion V2 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example configures system acceleration for MachineMotion v2. ###

mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)

acceleration = 500      # The max acceleration [mm/s^2] that all subsequent moves will move at
mm.emitAcceleration(acceleration)
print("Global acceleration set to " + str(acceleration) + "mm/s^2.")
```

emitCombinedAxesAbsoluteMove (axes,positions)

Compatibility: MachineMotion v1 & MachineMotion v2

Description

Moves multiple specified axes to their desired end locations synchronously. Combined moves are possible only on axis 1, 2 and 3.

Parameters

- **axes** (Required) List - The axes which will perform the move commands. Ex - [1 ,3]
- **positions** (Required) List - The desired end position of all axess movement. Ex - [50, 10]

Return Value

None

NOTE: The current speed and acceleration settings are applied to the combined motion of the axes.

MachineMotion V1 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases combined absolute moves with MachineMotion v1. ###

mm = MachineMotion()

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Configure actuators
axesToMove = [1,2,3]
for axis in axesToMove:
    mm.configAxis(axis, MICRO_STEPS.ustep_8, MECH_GAIN.timing_belt_150mm_turn)

### HOMING ###

# Home actuators before performing absolute moves
print("All Axes Moving Home Sequentially")
mm.emitHomeAll()
print("All Axes homed.")

### SIMULTANEOUS ABSOLUTE MOVES OF THREE AXES ###

#   Moves axis 1 to absolute position 50mm
#   Moves axis 2 to absolute position 100mm
#   Moves axis 3 to absolute position 50mm
positions = [50, 100, 50]
mm.emitCombinedAxesAbsoluteMove(axesToMove, positions)
mm.waitForMotionCompletion()
for index, axis in enumerate(axesToMove):
    print("Axis " + str(axis) + " moved to position " + str(positions[index]) + "mm")
```

MachineMotion V2 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases combined absolute moves with MachineMotion v2. ###

mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Configure actuators
axesToMove = [1,2,3] #Important Note : Combined moves are possible only on axes 1, 2 and 3.
for axis in axesToMove:
    mm.configServo(axis, MECH_GAIN.timing_belt_150mm_turn, DIRECTION.NORMAL, 5.0)

# Home actuators before performing absolute moves
print("Axes Moving Home Sequentially")
for axis in axesToMove:
    mm.emitHome(axis)
print("Axes homed")

# Simultaneously moves three axis:
#   Moves axis 1 to absolute position 50mm
#   Moves axis 2 to absolute position 100mm
#   Moves axis 3 to absolute position 50mm
positions = [50, 100, 50]
mm.emitCombinedAxesAbsoluteMove(axesToMove, positions)
mm.waitForMotionCompletion()
for index, axis in enumerate(axesToMove):
    print("Axis " + str(axis) + " moved to position " + str(positions[index]) + "mm")
```

emitCombinedAxesRelativeMove (axes,directions,distances)

Compatibility: MachineMotion v1 & MachineMotion v2

Description

Moves the multiple specified axes the specified distances in the specified directions. Combined moves are possible only on axis 1, 2 and 3.

Parameters

- **axes** (Required) List of Integers - The axes to move. Ex-[1,3]
- **directions** (Required) List of Strings pertaining to the DIRECTION class - The direction of travel of each specified axis. Ex - [DIRECTION.POSITIVE, DIRECTION.NEGATIVE]
- **distances** (Required) List of Numbers - The travel distances in mm. Ex - [10, 40]

Return Value

None

NOTE: The current speed and acceleration settings are applied to the combined motion of the axes.

MachineMotion V1 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases combined relative moves with MachineMotion v1. ###

mm = MachineMotion()

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Configure actuators
axesToMove = [1,2,3]
for axis in axesToMove:
    mm.configAxis(axis, MICRO_STEPS.ustep_8, MECH_GAIN.timing_belt_150mm_turn)

# Simultaneously moves three axis:
#   Move axis 1 in the positive direction by 50 mm
#   Move axis 2 in the negative direction by 100 mm
#   Move axis 3 in the positive direction by 50 mm
distances = [50, 100, 50]
directions = [DIRECTION.POSITIVE, DIRECTION.POSITIVE, DIRECTION.POSITIVE]
mm.emitCombinedAxesRelativeMove(axesToMove, directions, distances)
mm.waitForMotionCompletion()
for index, axis in enumerate(axesToMove):
    print("Axis " + str(axis) + " moved " + str(distances[index]) + "mm in the " + directions[index] + " direction.")
```

MachineMotion V2 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases combined relative moves with MachineMotion v1. ###

mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Configure actuators
axesToMove = [1,2,3] #Important Note : Combined moves are possible only on axes 1, 2 and 3.
for axis in axesToMove:
    mm.configServo(axis, MECH_GAIN.timing_belt_150mm_turn, DIRECTION.NORMAL, 5.0)

# Simultaneously moves three axis:
#   Move axis 1 in the positive direction by 50 mm
#   Move axis 2 in the negative direction by 100 mm
#   Move axis 3 in the positive direction by 50 mm
distances = [50, 100, 50]
directions = [DIRECTION.POSITIVE, DIRECTION.POSITIVE, DIRECTION.POSITIVE]
mm.emitCombinedAxesRelativeMove(axesToMove, directions, distances)
mm.waitForMotionCompletion()
for index, axis in enumerate(axesToMove):
    print("Axis " + str(axis) + " moved " + str(distances[index]) + "mm in the " + directions[index] + " direction.")
```

emitHome (axis)

Compatibility: MachineMotion v1 & MachineMotion v2

Description

Initiates the homing sequence for the specified axis.

Parameters

- **axis** (Required) Number - The axis to be homed.

Return Value

None

NOTE: If configAxisDirection is set to “normal” on axis 1, axis 1 will home itself towards sensor 1A. If configAxisDirection is set to “reverse” on axis 1, axis 1 will home itself towards sensor 1B.

MachineMotion V1 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases how to home actuators with MachineMotion v1. ###

mm = MachineMotion()

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Configure the actuator
axis = AXIS_NUMBER.DRIVE1
mm.configAxis(axis, MICRO_STEPS.ustep_8, MECH_GAIN.timing_belt_150mm_turn)

# Home the actuator
print ("Axis "+ str(axis) +" is going home")
mm.emitHome(axis)
print("Axis "+ str(axis) +" is at home")
```

MachineMotion V2 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases how to home actuators with MachineMotion v2. ###

mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Configure the actuator
axis = AXIS_NUMBER.DRIVE1
mm.configServo(axis, MECH_GAIN.timing_belt_150mm_turn, DIRECTION.NORMAL, 5.0)

# Home the actuator
print ("Axis "+ str(axis) +" is going home")
mm.emitHome(axis)
print("Axis "+ str(axis) +" is at home")
```

emitHomeAll ( )

Compatibility: MachineMotion v1 & MachineMotion v2

Description

Initiates the homing sequence of all axes. All axes will home sequentially.

Return Value

None

MachineMotion V1 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases how to home actuators with MachineMotion v1. ###

### MachineMotion configuration ###

mm = MachineMotion()

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

### Home all axes sequentially
print ("All Axes Moving Home sequentially")
mm.emitHomeAll()
print("All Axes Homed")
```

MachineMotion V2 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases how to home actuators with MachineMotion v2. ###

### MachineMotion configuration ###

mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

### Home all axes sequentially
print ("All Axes Moving Home sequentially")
mm.emitHomeAll()
print("All Axes Homed")
```

emitRelativeMove (axis,direction,distance)

Compatibility: MachineMotion v1 & MachineMotion v2

Description

Moves the specified axis the specified distance in the specified direction.

Parameters

- **axis** (Required) Integer - The axis to move.
- **direction** (Required) String pertaining to the DIRECTION class. - The direction of travel. Ex - DIRECTION.POSITIVE or DIRECTION.NEGATIVE
- **distance** (Required) Number - The travel distance in mm.

Return Value

None

MachineMotion V1 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases relative moves with MachineMotion v1. ###

mm = MachineMotion()

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Configure the actuator
axis = 1
mm.configAxis(axis, MICRO_STEPS.ustep_8, MECH_GAIN.rack_pinion_mm_turn)

# Begin Relative Move
distance = 100 # in mm
direction = DIRECTION.POSITIVE
mm.emitRelativeMove(axis, direction, distance)
mm.waitForMotionCompletion()
print("Axis " + str(axis) + " moved " + str(distance) + "mm in the " + direction + " direction.")
```

MachineMotion V2 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases relative moves with MachineMotion v2. ###

mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Configure the actuator
axis = 1
mm.configServo(axis, MECH_GAIN.timing_belt_150mm_turn, DIRECTION.NORMAL, 5.0)

# Begin Relative Move
distance = 100 # in mm
direction = DIRECTION.POSITIVE
mm.emitRelativeMove(axis, direction, distance)
mm.waitForMotionCompletion()
print("Axis " + str(axis) + " moved " + str(distance) + "mm in the " + direction + " direction.")
```

emitSpeed (speed,units)

Compatibility: MachineMotion v1 & MachineMotion v2

Description

Sets the global speed for all movement commands on all axes.

Parameters

- **speed** (Required) Number - The global max speed in mm/sec, or mm/min according to the units parameter.
- **units** (Optional. Default = UNITS_SPEED.mm_per_sec) String - Units for speed. Can be switched to UNITS_SPEED.mm_per_min

Return Value

None

MachineMotion V1 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example configures system speed for MachineMotion v1. ###

mm = MachineMotion()

speed = 500      # The max speed [mm/s] that all subsequent moves will move at
mm.emitSpeed(speed)
print("Global speed set to " + str(speed) + "mm/s.")
```

MachineMotion V2 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example configures system speed for MachineMotion v2. ###

mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)

speed = 500      # The max speed [mm/s] that all subsequent moves will move at
mm.emitSpeed(speed)
print("Global speed set to " + str(speed) + "mm/s.")
```

emitStop ( )

Compatibility: MachineMotion v1 & MachineMotion v2

Description

Immediately stops all motion of all axes.

Return Value

None

NOTE: This function is a hard stop. It is not a controlled stop and consequently does not decelerate smoothly to a stop. Additionally, this function is not intended to serve as an emergency stop since this stop mechanism does not have safety ratings.

MachineMotion V1 Example Code

```
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases how to stop motion on MachineMotion v1. ###

mm = MachineMotion()

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Configure the actuator
axis = 1
mm.configAxis(axis, MICRO_STEPS.ustep_8, MECH_GAIN.rack_pinion_mm_turn)

# Configure Move Parameters
speed = 200
mm.emitSpeed(speed)

# Begin Relative Move
distance = 1000
direction = DIRECTION.POSITIVE
mm.emitRelativeMove(axis, direction, distance)
print("Axis " + str(axis) + " is moving " + str(distance) + "mm in the " + direction + " direction.")
# This move should take 5 seconds to complete (distance/speed). Instead, we wait 2 seconds and then stop the machine.
time.sleep(2)
mm.emitStop()
print("Axis " + str(axis) + " stopped.")
```

MachineMotion V2 Example Code

```
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases how to stop motion on MachineMotion v2. ###

mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Configure the actuator
axis = 1
mm.configServo(axis, MECH_GAIN.timing_belt_150mm_turn, DIRECTION.NORMAL, 5.0)

# Configure Move Parameters
speed = 200
mm.emitSpeed(speed)

# Begin Relative Move
distance = 1000
direction = DIRECTION.POSITIVE
mm.emitRelativeMove(axis, direction, distance)
print("Axis " + str(axis) + " is moving " + str(distance) + "mm in the " + direction + " direction")
# This move should take 5 seconds to complete (distance/speed). Instead, we wait 2 seconds and then stop the machine.
time.sleep(2)
mm.emitStop()
print("Axis " + str(axis) + " stopped.")
```

emitgCode (gCode)

Compatibility: MachineMotion V1 only

Description

Executes raw gCode on the controller.

Parameters

- **gCode** (Required) string - The g-code that will be passed directly to the controller.

Return Value

None

NOTE: All movement commands sent to the controller are by default in mm.

MachineMotion V1 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases how to emit custom gcode with MachineMotion v1. ###

# Define a callback to process controller gCode responses (if desired)
def templateCallback(data):
   print ( "Controller gCode responses " + data )

mm = MachineMotion(DEFAULT_IP, gCodeCallback = templateCallback)

#When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Use the G2 to perform combined arc move on the first 2 axes
gCodeCommand = "G2 I10 J10"
reply = mm.emitgCode(gCodeCommand)
mm.waitForMotionCompletion()
print("G-code command '" + gCodeCommand + "' completed by MachineMotion.")
print("Reply is : " + reply)
```

getActualPositions (axis (optional))

Compatibility: MachineMotion v1 & MachineMotion v2

Description

Returns the current position of the axes.

Parameters

- **axis (optional)** (Required) Number - The axis to get the current position of.

Return Value

- The position of the axis if that parameter was specified, or a dictionary containing the current position of every axis.

MachineMotion V1 Example Code

```
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases how to read actuator positions with MachineMotion v1. ###

###### CONFIGURING MACHINEMOTION ######

mm = MachineMotion()

# When starting a program, one must remove the software stop before moving
```

```
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Configure actuator
axis = 1
print("--> Configuring actuator")
mm.configAxis(axis, MICRO_STEPS.ustep_8, MECH_GAIN.rack_pinion_mm_turn)

# Home Axis Before Moving
print("--> Axis " + str(axis) + " moving home")
mm.emitHome(axis)
print("--> Axis " + str(axis) + " homed")


###### READ POSITIONS ######

# Read the position of one axis
print("--> Read the position of one axis")
desiredPosition_axis = mm.getDesiredPositions(axis)
actualPosition_axis  = mm.getActualPositions(axis)
print("Desired position of axis " + str(axis) + " is : " + str(desiredPosition_axis) + " mm.")
print("Actual position of axis " + str(axis) + " is : " + str(actualPosition_axis) + " mm.")

# Read the position of several axes
print("--> Read the position of several axes")
desiredPositions = mm.getDesiredPositions()
actualPositions  = mm.getActualPositions()
print("Desired position of axis 1 is : " + str(desiredPositions[1]) + " mm.")
print("Desired position of axis 2 is : " + str(desiredPositions[2]) + " mm.")
print("Desired position of axis 3 is : " + str(desiredPositions[3]) + " mm.")
print("Actual position of axis 1 is : " + str(actualPositions[1]) + " mm.")
print("Actual position of axis 2 is : " + str(actualPositions[2]) + " mm.")
print("Actual position of axis 3 is : " + str(actualPositions[3]) + " mm.")


###### MOVE AND READ POSITIONS ######

# Define Motion Parameters
distance = 500
move_direction = DIRECTION.POSITIVE

# Move 500mm and check position again
mm.emitRelativeMove(axis, move_direction, distance)
desiredPosition_axis = mm.getDesiredPositions(axis)
print("--> Move ongoing")
print("Desired position of axis " + str(axis) + " is : " + str(desiredPosition_axis) + " mm.")
while not mm.isMotionCompleted():
    actualPosition_axis  = mm.getActualPositions(axis)
    print("Actual position of axis " + str(axis) + " is : " + str(actualPosition_axis) + " mm.")

mm.waitForMotionCompletion()
print("--> Move completed")
desiredPosition_axis = mm.getDesiredPositions(axis)
actualPosition_axis  = mm.getActualPositions(axis)
print("Desired position of axis " + str(axis) + " is : " + str(desiredPosition_axis) + " mm.")
print("Actual position of axis " + str(axis) + " is : " + str(actualPosition_axis) + " mm.")

print("--> End of example")
```

MachineMotion V2 Example Code

```
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases how to read actuator positions with MachineMotion v2. ###

###### CONFIGURING MACHINEMOTION ######

mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Configure actuator
axis = 1
print("--> Configuring actuator")
mm.configServo(axis, MECH_GAIN.rack_pinion_mm_turn, DIRECTION.POSITIVE, 5.0)

# Home Axis Before Moving
print("--> Axis " + str(axis) + " moving home")
mm.emitHome(axis)
print("--> Axis " + str(axis) + " homed")


###### READ POSITIONS ######

# Read the position of one axis
print("--> Read the position of one axis")
actualPosition_axis  = mm.getActualPositions(axis)
print("Actual position of axis " + str(axis) + " is : " + str(actualPosition_axis) + " mm.")

# Read the position of several axes
print("--> Read the position of several axes")
actualPositions  = mm.getActualPositions()
print("Actual position of axis 1 is : " + str(actualPositions[1]) + " mm.")
print("Actual position of axis 2 is : " + str(actualPositions[2]) + " mm.")
print("Actual position of axis 3 is : " + str(actualPositions[3]) + " mm.")
print("Actual position of axis 4 is : " + str(actualPositions[4]) + " mm.")

###### MOVE AND READ POSITIONS ######

# Define Motion Parameters
distance = 100
move_direction = DIRECTION.POSITIVE

# Move 100mm and check position again
mm.emitRelativeMove(axis, move_direction, distance)
print("--> Move ongoing")
while not mm.isMotionCompleted():
    actualPosition_axis  = mm.getActualPositions(axis)
    print("Actual position of axis " + str(axis) + " is : " + str(actualPosition_axis) + " mm.")

mm.waitForMotionCompletion()
print("--> Move completed")
actualPosition_axis  = mm.getActualPositions(axis)
print("Actual position of axis " + str(axis) + " is : " + str(actualPosition_axis) + " mm.")

print("--> End of example")
```

getBrakeState (aux_port_number,safety_adapter_presence)

Compatibility: MachineMotion v1 & MachineMotion v2

Description

Read the current state of the brake connected to a given AUX port of the MachineMotion.

Parameters

- **aux_port_number** (Required) Integer - The number of the AUX port the brake is connected to.
- **safety_adapter_presence** (Required) Boolean - Is a yellow safety adapter plugged in between the brake cable and the AUX port.

Return Value

- The current state of the brake, as determined according to the current voltage of the AUX port (0V or 24V). The returned String can be "locked", "unlocked", or "unknown" (for MachineMotions prior to the V1F hardware version), as defined by the BRAKE_STATES class.

NOTE: This function is compatible only with V1F and more recent MachineMotions.

MachineMotion V1 Example Code

```
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example control brakes on MachineMotion v1. ###

print(" ----- WARNING ------ Does your hardware version support brakes?")

mm = MachineMotion()

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Define the brake parameters
brake_port_number = 1           # The AUX port number the brake is plugged in
safety_adapter_presence = True  # Is a yellow safety adapter plugged in between the brake cable and the AUX port

# Read brake state
brake_state = mm.getBrakeState (brake_port_number, safety_adapter_presence)
print ("The brake connected to AUX" + str(brake_port_number) + " is : " + brake_state)

# Lock the brake
mm.lockBrake (brake_port_number, safety_adapter_presence)
time.sleep(0.2);                # Wait before reading brake state, for it to get correctly updated

# Read brake state
brake_state = mm.getBrakeState (brake_port_number, safety_adapter_presence)
print ("The brake connected to AUX" + str(brake_port_number) + " is : " + brake_state)

# DO NOT MOVE WHILE BRAKE IS ENGAGED
print ("Waiting two seconds. Do not move the actuator while the brake is locked !")
time.sleep(2)                   # Unlock the brake after two seconds

# Release the brakes
mm.unlockBrake (brake_port_number, safety_adapter_presence)
time.sleep(0.2);                # Wait before reading brake state, for it to get correctly updated

# Read brake state
brake_state = mm.getBrakeState (brake_port_number, safety_adapter_presence)
print ("The brake connected to AUX" + str(brake_port_number) + " is : " + brake_state)
```

MachineMotion V2 Example Code

```
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example control brakes on MachineMotion v2. ###

mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Define the brake parameters
axis = 1                        # Drive number
safety_adapter_presence = True  # Is a yellow safety adapter plugged in between the brake cable and the brake port
                                # Important Note : This flag must always be set to "True" on MachineMotions v2.

# Read brake state
brake_state = mm.getBrakeState(axis, safety_adapter_presence)
print ("The brake connected to port " + str(axis) + " is : " + brake_state)

# Lock the brake
mm.lockBrake(axis, safety_adapter_presence)
time.sleep(0.2);                # Wait before reading brake state, for it to get correctly updated

# Read brake state
brake_state = mm.getBrakeState(axis, safety_adapter_presence)
print ("The brake connected to port " + str(axis) + " is : " + brake_state)

# DO NOT MOVE WHILE BRAKE IS ENGAGED
print ("Waiting two seconds. Do not move the actuator while the brake is locked !")
time.sleep(2)                   # Unlock the brake after two seconds

# Release the brakes
mm.unlockBrake(axis, safety_adapter_presence)
time.sleep(0.2);                # Wait before reading brake state, for it to get correctly updated

# Read brake state
brake_state = mm.getBrakeState(axis, safety_adapter_presence)
print ("The brake connected to port " + str(axis) + " is : " + brake_state)
```

getDesiredPositions (axis (optional))

Compatibility: MachineMotion V1 only

Description

Returns the desired position of the axes.

Parameters

- **axis (optional)** (Required) Number - The axis to get the desired position of.

Return Value

- The position of the axis if that parameter was specified, or a dictionary containing the desired position of every axis.

NOTE: This function returns the 'open loop' position of each axis.

MachineMotion V1 Example Code

```
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases how to read actuator positions with MachineMotion v1. ###

###### CONFIGURING MACHINEMOTION ######

mm = MachineMotion()

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Configure actuator
axis = 1
print("--> Configuring actuator")
mm.configAxis(axis, MICRO_STEPS.ustep_8, MECH_GAIN.rack_pinion_mm_turn)

# Home Axis Before Moving
print("--> Axis " + str(axis) + " moving home")
mm.emitHome(axis)
print("--> Axis " + str(axis) + " homed")


###### READ POSITIONS ######

# Read the position of one axis
print("--> Read the position of one axis")
desiredPosition_axis = mm.getDesiredPositions(axis)
actualPosition_axis  = mm.getActualPositions(axis)
print("Desired position of axis " + str(axis) + " is : " + str(desiredPosition_axis) + " mm.")
print("Actual position of axis " + str(axis) + " is : " + str(actualPosition_axis) + " mm.")

# Read the position of several axes
print("--> Read the position of several axes")
desiredPositions = mm.getDesiredPositions()
actualPositions  = mm.getActualPositions()
print("Desired position of axis 1 is : " + str(desiredPositions[1]) + " mm.")
print("Desired position of axis 2 is : " + str(desiredPositions[2]) + " mm.")
print("Desired position of axis 3 is : " + str(desiredPositions[3]) + " mm.")
print("Actual position of axis 1 is : " + str(actualPositions[1]) + " mm.")
print("Actual position of axis 2 is : " + str(actualPositions[2]) + " mm.")
print("Actual position of axis 3 is : " + str(actualPositions[3]) + " mm.")


###### MOVE AND READ POSITIONS ######

# Define Motion Parameters
distance = 500
move_direction = DIRECTION.POSITIVE

# Move 500mm and check position again
mm.emitRelativeMove(axis, move_direction, distance)
desiredPosition_axis = mm.getDesiredPositions(axis)
print("--> Move ongoing")
```

```
    print("Desired position of axis " + str(axis) + " is : " + str(desiredPosition_axis) + " mm.")
    while not mm.isMotionCompleted():
        actualPosition_axis  = mm.getActualPositions(axis)
        print("Actual position of axis " + str(axis) + " is : " + str(actualPosition_axis) + " mm.")

    mm.waitForMotionCompletion()
    print("--> Move completed")
    desiredPosition_axis = mm.getDesiredPositions(axis)
    actualPosition_axis  = mm.getActualPositions(axis)
    print("Desired position of axis " + str(axis) + " is : " + str(desiredPosition_axis) + " mm.")
    print("Actual position of axis " + str(axis) + " is : " + str(actualPosition_axis) + " mm.")

    print("--> End of example")
```

getEndStopState ( )

Compatibility: MachineMotion v1 & MachineMotion v2

Description

Returns the current state of all home and end sensors.

Return Value

- The states of all end stop sensors {x_min, x_max, y_min, y_max, z_min, z_max} “TRIGGERED” or “open”

MachineMotion V1 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases how to read enstop sensor states with MachineMotion v1. ###

mm = MachineMotion()

# When starting a program, one must remove the software stop
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Get End Stop State
endstopStates = mm.getEndStopState()
# If the direction of the axis is normal,
#   then the home sensor is the "_min" sensor, and the end sensor is the "_max" sensor.
# If the direction of the axis reversed,
#   then the home sensor is the "_max" sensor, and the end sensor is the "_min" sensor.
axis1_home_sensor_status      = endstopStates['x_min']
axis1_endstop_sensor_status   = endstopStates['x_max']
axis2_home_sensor_status      = endstopStates['y_min']
axis2_endstop_sensor_status   = endstopStates['y_max']
axis3_home_sensor_status      = endstopStates['z_min']
axis3_endstop_sensor_status   = endstopStates['z_max']

print("Axis 1 : " + "Home sensor is : " + str(axis1_home_sensor_status) )
print("Axis 1 : " + "End sensor is : " + str(axis1_endstop_sensor_status) )
print("Axis 2 : " + "Home sensor is : " + str(axis2_home_sensor_status) )
print("Axis 2 : " + "End sensor is : " + str(axis2_endstop_sensor_status) )
print("Axis 3 : " + "Home sensor is : " + str(axis3_home_sensor_status) )
print("Axis 3 : " + "End sensor is : " + str(axis3_endstop_sensor_status) )
```

MachineMotion V2 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases how to read enstop sensor states with MachineMotion v2. ###

mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)

# When starting a program, one must remove the software stop
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Get End Stop State
endstopStates = mm.getEndStopState()
# If the direction of the axis is normal,
#   then the home sensor is the "_min" sensor, and the end sensor is the "_max" sensor.
# If the direction of the axis reversed,
#   then the home sensor is the "_max" sensor, and the end sensor is the "_min" sensor.
axis1_home_sensor_status      = endstopStates['x_min']
axis1_endstop_sensor_status   = endstopStates['x_max']
axis2_home_sensor_status      = endstopStates['y_min']
axis2_endstop_sensor_status   = endstopStates['y_max']
axis3_home_sensor_status      = endstopStates['z_min']
axis3_endstop_sensor_status   = endstopStates['z_max']
axis4_home_sensor_status      = endstopStates['w_min']
axis4_endstop_sensor_status   = endstopStates['w_max']

print("Axis 1 : " + "Home sensor is : " + str(axis1_home_sensor_status) )
print("Axis 1 : " + "End sensor is : " + str(axis1_endstop_sensor_status) )
print("Axis 2 : " + "Home sensor is : " + str(axis2_home_sensor_status) )
print("Axis 2 : " + "End sensor is : " + str(axis2_endstop_sensor_status) )
print("Axis 3 : " + "Home sensor is : " + str(axis3_home_sensor_status) )
print("Axis 3 : " + "End sensor is : " + str(axis3_endstop_sensor_status) )
print("Axis 4 : " + "Home sensor is : " + str(axis4_home_sensor_status) )
print("Axis 4 : " + "End sensor is : " + str(axis4_endstop_sensor_status) )
```

isMotionCompleted ( )

Compatibility: MachineMotion v1 & MachineMotion v2

Description

Indicates if the last move command has completed.

Return Value

- Returns false if the machine is currently executing a movement command.

NOTE: isMotionCompleted does not account for on-going continuous moves.

MachineMotion V1 Example Code

```
import sys, time
```

```
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases how to read actuator positions with MachineMotion v1. ###

###### CONFIGURING MACHINEMOTION ######

mm = MachineMotion()

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Configure actuator
axis = 1
print("--> Configuring actuator")
mm.configAxis(axis, MICRO_STEPS.ustep_8, MECH_GAIN.rack_pinion_mm_turn)

# Home Axis Before Moving
print("--> Axis " + str(axis) + " moving home")
mm.emitHome(axis)
print("--> Axis " + str(axis) + " homed")


###### READ POSITIONS ######

# Read the position of one axis
print("--> Read the position of one axis")
desiredPosition_axis = mm.getDesiredPositions(axis)
actualPosition_axis  = mm.getActualPositions(axis)
print("Desired position of axis " + str(axis) + " is : " + str(desiredPosition_axis) + " mm.")
print("Actual position of axis " + str(axis) + " is : " + str(actualPosition_axis) + " mm.")

# Read the position of several axes
print("--> Read the position of several axes")
desiredPositions = mm.getDesiredPositions()
actualPositions  = mm.getActualPositions()
print("Desired position of axis 1 is : " + str(desiredPositions[1]) + " mm.")
print("Desired position of axis 2 is : " + str(desiredPositions[2]) + " mm.")
print("Desired position of axis 3 is : " + str(desiredPositions[3]) + " mm.")
print("Actual position of axis 1 is : " + str(actualPositions[1]) + " mm.")
print("Actual position of axis 2 is : " + str(actualPositions[2]) + " mm.")
print("Actual position of axis 3 is : " + str(actualPositions[3]) + " mm.")


###### MOVE AND READ POSITIONS ######

# Define Motion Parameters
distance = 500
move_direction = DIRECTION.POSITIVE

# Move 500mm and check position again
mm.emitRelativeMove(axis, move_direction, distance)
desiredPosition_axis = mm.getDesiredPositions(axis)
print("--> Move ongoing")
print("Desired position of axis " + str(axis) + " is : " + str(desiredPosition_axis) + " mm.")
while not mm.isMotionCompleted():
    actualPosition_axis  = mm.getActualPositions(axis)
    print("Actual position of axis " + str(axis) + " is : " + str(actualPosition_axis) + " mm.")

mm.waitForMotionCompletion()
print("--> Move completed")
desiredPosition_axis = mm.getDesiredPositions(axis)
actualPosition_axis  = mm.getActualPositions(axis)
```

```
        print("Desired position of axis " + str(axis) + " is : " + str(desiredPosition_axis) + " mm.")
        print("Actual position of axis " + str(axis) + " is : " + str(actualPosition_axis) + " mm.")

        print("--> End of example")
```

MachineMotion V2 Example Code

```python
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases how to read actuator positions with MachineMotion v2. ###

###### CONFIGURING MACHINEMOTION ######

mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Configure actuator
axis = 1
print("--> Configuring actuator")
mm.configServo(axis, MECH_GAIN.rack_pinion_mm_turn, DIRECTION.POSITIVE, 5.0)

# Home Axis Before Moving
print("--> Axis " + str(axis) + " moving home")
mm.emitHome(axis)
print("--> Axis " + str(axis) + " homed")


###### READ POSITIONS ######

# Read the position of one axis
print("--> Read the position of one axis")
actualPosition_axis  = mm.getActualPositions(axis)
print("Actual position of axis " + str(axis) + " is : " + str(actualPosition_axis) + " mm.")

# Read the position of several axes
print("--> Read the position of several axes")
actualPositions  = mm.getActualPositions()
print("Actual position of axis 1 is : " + str(actualPositions[1]) + " mm.")
print("Actual position of axis 2 is : " + str(actualPositions[2]) + " mm.")
print("Actual position of axis 3 is : " + str(actualPositions[3]) + " mm.")
print("Actual position of axis 4 is : " + str(actualPositions[4]) + " mm.")

###### MOVE AND READ POSITIONS ######

# Define Motion Parameters
distance = 100
move_direction = DIRECTION.POSITIVE

# Move 100mm and check position again
mm.emitRelativeMove(axis, move_direction, distance)
print("--> Move ongoing")
while not mm.isMotionCompleted():
    actualPosition_axis  = mm.getActualPositions(axis)
    print("Actual position of axis " + str(axis) + " is : " + str(actualPosition_axis) + " mm.")

mm.waitForMotionCompletion()
print("--> Move completed")
actualPosition_axis  = mm.getActualPositions(axis)
print("Actual position of axis " + str(axis) + " is : " + str(actualPosition_axis) + " mm.")

print("--> End of example")
```

lockBrake (aux_port_number,safety_adapter_presence)

Compatibility: MachineMotion v1 & MachineMotion v2

Description

Lock the brake, by shutting off the power of the designated AUX port of the MachineMotion (0V).

Parameters

- **aux_port_number** (Required) Integer - The number of the AUX port the brake is connected to.
- **safety_adapter_presence** (Required) Boolean - Is a yellow safety adapter plugged in between the brake cable and the AUX port.

Return Value

None

NOTE: This function is compatible only with V1F and more recent MachineMotions.

MachineMotion V1 Example Code

```
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example control brakes on MachineMotion v1. ###

print(" ----- WARNING ------ Does your hardware version support brakes?")

mm = MachineMotion()

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Define the brake parameters
brake_port_number = 1           # The AUX port number the brake is plugged in
safety_adapter_presence = True  # Is a yellow safety adapter plugged in between the brake cable and the AUX port

# Read brake state
brake_state = mm.getBrakeState (brake_port_number, safety_adapter_presence)
print ("The brake connected to AUX" + str(brake_port_number) + " is : " + brake_state)

# Lock the brake
mm.lockBrake (brake_port_number, safety_adapter_presence)
time.sleep(0.2);                 # Wait before reading brake state, for it to get correctly updated

# Read brake state
brake_state = mm.getBrakeState (brake_port_number, safety_adapter_presence)
print ("The brake connected to AUX" + str(brake_port_number) + " is : " + brake_state)

# DO NOT MOVE WHILE BRAKE IS ENGAGED
print ("Waiting two seconds. Do not move the actuator while the brake is locked !")
time.sleep(2)                    # Unlock the brake after two seconds

# Release the brakes
mm.unlockBrake (brake_port_number, safety_adapter_presence)
time.sleep(0.2);                 # Wait before reading brake state, for it to get correctly updated

# Read brake state
brake_state = mm.getBrakeState (brake_port_number, safety_adapter_presence)
print ("The brake connected to AUX" + str(brake_port_number) + " is : " + brake_state)
```

MachineMotion V2 Example Code

```
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example control brakes on MachineMotion v2. ###

mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Define the brake parameters
axis = 1                        # Drive number
safety_adapter_presence = True  # Is a yellow safety adapter plugged in between the brake cable and the brake port
                                # Important Note : This flag must always be set to "True" on MachineMotions v2.

# Read brake state
brake_state = mm.getBrakeState(axis, safety_adapter_presence)
print ("The brake connected to port " + str(axis) + " is : " + brake_state)

# Lock the brake
mm.lockBrake(axis, safety_adapter_presence)
time.sleep(0.2);                # Wait before reading brake state, for it to get correctly updated

# Read brake state
brake_state = mm.getBrakeState(axis, safety_adapter_presence)
print ("The brake connected to port " + str(axis) + " is : " + brake_state)

# DO NOT MOVE WHILE BRAKE IS ENGAGED
print ("Waiting two seconds. Do not move the actuator while the brake is locked !")
time.sleep(2)                   # Unlock the brake after two seconds

# Release the brakes
mm.unlockBrake(axis, safety_adapter_presence)
time.sleep(0.2);                # Wait before reading brake state, for it to get correctly updated

# Read brake state
brake_state = mm.getBrakeState(axis, safety_adapter_presence)
print ("The brake connected to port " + str(axis) + " is : " + brake_state)
```

readEncoder (encoder,readingType)

Compatibility: MachineMotion V1 only

Description

Returns the last received encoder position in counts.

Parameters

- **encoder** (Required) Integer - The identifier of the encoder to read
- **readingType** (Required) String - Either 'real time' or 'stable'. In 'real time' mode, readEncoder will return the most recently received encoder information. In 'stable' mode, readEncoder will update its return value only after the encoder output has stabilized around a specific value, such as when the axis has stopped motion.

Return Value

- The current position of the encoder, in counts. The encoder has 3600 counts per revolution.

NOTE: The encoder position returned by this function may be delayed by up to 250 ms due to internal propogation delays.

MachineMotion V1 Example Code

```
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases how to read encoder positions with MachineMotion v1. ###

mm = MachineMotion()

# Adjust this line to match whichever AUX port the encoder is plugged into
encoderPort = AUX_PORTS.aux_1

checkInput = True
print("Reading and printing encoder output for 10 seconds:")
for i in range(1,30):
  realTimeOutput = mm.readEncoder(encoderPort, ENCODER_TYPE.real_time)
  stableOutput = mm.readEncoder(encoderPort, ENCODER_TYPE.stable)
  print("Encoder on port AUX " + str(encoderPort+1) + "\t Realtime Position =" + str(realTimeOutput) + " counts\t StablePosition = " + str(sta
  time.sleep(.25)

  if realTimeOutput != 0:
    checkInput = False

if(checkInput):
  print("The encoder is not receiving any data. Please check the following: \n\t Is the encoder plugged into AUX " + str(encoderPort+1) + "?
```

releaseEstop ( )

Compatibility: MachineMotion v1 & MachineMotion v2

Description

Releases the software E-stop and provides power back to the drives.

Return Value

- The success of the operation.

MachineMotion V1 Example Code

```
#!/usr/bin/python
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases how to control eStop on MachineMotion v1. ###

### MachineMotion CONFIGURATION ###

# Create MachineMotion instance
mm = MachineMotion()
time.sleep(0.1) # Wait to initialize internal eStop topics.

# Define a callback to process estop status
def templateCallback(estop_status):
    print("eStop status is : " + str(estop_status))
    if estop_status == True :
        # executed when you enter estop
        print("MachineMotion is in estop state.")
    elif estop_status == False :
        # executed when you exit estop
        print("MachineMotion is not in estop state.")
mm.bindeStopEvent(templateCallback)

### ESTOP TRIGGER ###

result = mm.triggerEstop()
if result == True   : print("--> Software stop triggered")
else                : print("--> Failed to trigger software stop")

time.sleep(3.0)

### ESTOP RELEASE ###

result = mm.releaseEstop()
if result == True   : print("--> Software stop released")
else                : print("--> Failed to release software stop")

time.sleep(3.0)

### SYSTEM RESET ###

result = mm.resetSystem()
if result == True   : print("--> System has been reset")
else                : print("--> Failed to reset system")

print("--> Example completed")
```

MachineMotion V2 Example Code

```
#!/usr/bin/python
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases how to control eStop on MachineMotion v2. ###

### MachineMotion CONFIGURATION ###

# Create MachineMotion instance
mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)
time.sleep(0.1) # Wait to initialize internal eStop topics.

# Define a callback to process estop status
def templateCallback(estop_status):
    print("eStop status is : " + str(estop_status))
    if estop_status == True :
        # executed when you enter estop
        print("MachineMotion is in estop state.")
    elif estop_status == False :
        # executed when you exit estop
        print("MachineMotion is not in estop state.")
mm.bindeStopEvent(templateCallback)

### ESTOP TRIGGER ###

result = mm.triggerEstop()
if result == True   : print("--> Software stop triggered")
else                : print("--> Failed to trigger software stop")

time.sleep(3.0)

### ESTOP RELEASE ###

result = mm.releaseEstop()
if result == True   : print("--> Software stop released")
else                : print("--> Failed to release software stop")

time.sleep(3.0)

### SYSTEM RESET ###

result = mm.resetSystem()
if result == True   : print("--> System has been reset")
else                : print("--> Failed to reset system")

print("--> Example completed")
```

resetSystem ( )

Compatibility: MachineMotion v1 & MachineMotion v2

Description

Resets the system after an eStop event

Return Value

- The success of the operation.

MachineMotion V1 Example Code

```
#!/usr/bin/python
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases how to control eStop on MachineMotion v1. ###

### MachineMotion CONFIGURATION ###

# Create MachineMotion instance
mm = MachineMotion()
time.sleep(0.1) # Wait to initialize internal eStop topics.

# Define a callback to process estop status
def templateCallback(estop_status):
    print("eStop status is : " + str(estop_status))
    if estop_status == True :
        # executed when you enter estop
        print("MachineMotion is in estop state.")
    elif estop_status == False :
        # executed when you exit estop
        print("MachineMotion is not in estop state.")
mm.bindeStopEvent(templateCallback)

### ESTOP TRIGGER ###

result = mm.triggerEstop()
if result == True   : print("--> Software stop triggered")
else                : print("--> Failed to trigger software stop")

time.sleep(3.0)

### ESTOP RELEASE ###

result = mm.releaseEstop()
if result == True   : print("--> Software stop released")
else                : print("--> Failed to release software stop")

time.sleep(3.0)

### SYSTEM RESET ###

result = mm.resetSystem()
if result == True   : print("--> System has been reset")
else                : print("--> Failed to reset system")

print("--> Example completed")
```

MachineMotion V2 Example Code

```
#!/usr/bin/python
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases how to control eStop on MachineMotion v2. ###

### MachineMotion CONFIGURATION ###

# Create MachineMotion instance
mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)
time.sleep(0.1) # Wait to initialize internal eStop topics.

# Define a callback to process estop status
def templateCallback(estop_status):
    print("eStop status is : " + str(estop_status))
    if estop_status == True :
        # executed when you enter estop
        print("MachineMotion is in estop state.")
    elif estop_status == False :
        # executed when you exit estop
        print("MachineMotion is not in estop state.")
mm.bindeStopEvent(templateCallback)

### ESTOP TRIGGER ###

result = mm.triggerEstop()
if result == True   : print("--> Software stop triggered")
else                : print("--> Failed to trigger software stop")

time.sleep(3.0)

### ESTOP RELEASE ###

result = mm.releaseEstop()
if result == True   : print("--> Software stop released")
else                : print("--> Failed to release software stop")

time.sleep(3.0)

### SYSTEM RESET ###

result = mm.resetSystem()
if result == True   : print("--> System has been reset")
else                : print("--> Failed to reset system")

print("--> Example completed")
```

setContinuousMove (axis,speed,accel)

Compatibility: MachineMotion v1 & MachineMotion v2

Description

Starts an axis using speed mode.

Parameters

- **axis** (Required) Number - Axis to move
- **speed** (Required) Number - Speed to move the axis at in mm / sec
- **accel** (Required) Number - Acceleration used to reach the desired speed, in mm / sec^2

Return Value

None

MachineMotion V1 Example Code

```
#!/usr/bin/python
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases continuous moves with MachineMotion v1. ###

### MachineMotion CONFIGURATION ###

# Create MachineMotion instance
mm = MachineMotion()

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Configure actuator
conveyor_axis = 1
mm.configAxis(conveyor_axis, MICRO_STEPS.ustep_8, MECH_GAIN.roller_conveyor_mm_turn)

### CONTINUOUS MOVES ###

conveyor_stop_acc = 500 # Deceleration of 500mm/s^2

print("Start Conveyor Move...")
print("Continuous move: speed 100mm/s & acceleration 50mm/s^2")
mm.setContinuousMove(conveyor_axis, 100, 50)
time.sleep(5)

# Change speed while moving
print("Continuous move: speed 500mm/s & acceleration 250mm/s^2")
mm.setContinuousMove(conveyor_axis, 500, 250)
time.sleep(5)

# Stop the continuous move
mm.stopContinuousMove(conveyor_axis, conveyor_stop_acc)
time.sleep(2)

# Reverse direction of conveyor by changing the sign of the speed
print("Reverse continuous move: speed -1000mm/s & acceleration 500mm/s^2")
mm.setContinuousMove(conveyor_axis, -1000, 500)
time.sleep(5)

# Stop the continuous move
print("Stop Conveyor Move...")
mm.stopContinuousMove(conveyor_axis, conveyor_stop_acc)
time.sleep(3)
print("--> Example completed")
```

MachineMotion V2 Example Code

```
#!/usr/bin/python
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases continuous moves with MachineMotion v2. ###

### MachineMotion CONFIGURATION ###

# Create MachineMotion instance
mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Configure actuator
conveyor_axis = 1
mm.configServo(conveyor_axis, MECH_GAIN.roller_conveyor_mm_turn, DIRECTION.NORMAL, 5.0)

### CONTINUOUS MOVES ###

# Start the continuous move
print("Start Conveyor Move...")
print("Continuous move: speed 100mm/s & acceleration 50mm/s^2")
mm.setContinuousMove(conveyor_axis, 100, 50)
time.sleep(5)

# Change speed while moving
print("Continuous move: speed 500mm/s & acceleration 250mm/s^2")
mm.setContinuousMove(conveyor_axis, 500, 250)
time.sleep(5)

# Reverse direction of conveyor by changing the sign of the speed
print("Reverse continuous move: speed -1000mm/s & acceleration 500mm/s^2")
mm.setContinuousMove(conveyor_axis, -1000, 500)
time.sleep(5)

# Stop the continuous move
print("Stop Conveyor Move...")
mm.stopContinuousMove(conveyor_axis, 500)
time.sleep(3)
print("--> Example completed")
```

setPosition (axis,position)

Compatibility: MachineMotion v1 & MachineMotion v2

Description

Override the current position of the specified axis to a new value.

Parameters

- **axis** (Required) Number - Overrides the position on this axis.

- **position** (Required) Number - The new position value in mm.

Return Value

None

MachineMotion V1 Example Code

```
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases how to set actuator position with MachineMotion v1. ###

mm = MachineMotion()

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Configure the actuator
axis = 1
mm.configAxis(axis, MICRO_STEPS.ustep_8, MECH_GAIN.rack_pinion_mm_turn)

# Home the actuator
print("Axis " + str(axis) + " will home")
mm.emitHome(axis)
print("Axis " + str(axis) + " homed")

### Perform asolute moves with different reference points ###

print("Absolute Moves are referenced from home")
position = 100
mm.emitAbsoluteMove(axis, position)
mm.waitForMotionCompletion()
print("Axis " + str(axis) + " is " + str(position) + "mm away from home.")

# Change the reference point to the current position
mm.setPosition(axis, 0)
print("Absolute moves on axis " + str(axis) + " are now referenced from " +  str(position) + "mm from home. ")
time.sleep(2)

# Move again
position2 = 30
print("Now moving to absolute position " + str(position2) + " mm, referenced from location 'setPosition' was called")
mm.emitAbsoluteMove(axis, position2)
mm.waitForMotionCompletion()
print("Axis " + str(axis) + " is now " + str(position2) + "mm from reference position and " + str(position + position2) + "mm from home")
```

MachineMotion V2 Example Code

```
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases how to set actuator position with MachineMotion v2. ###

mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Configure the actuator
axis = 1
mm.configServo(axis, MECH_GAIN.timing_belt_150mm_turn, DIRECTION.NORMAL, 5.0)

# Home the actuator
print("Axis " + str(axis) + " will home")
mm.emitHome(axis)
print("Axis " + str(axis) + " homed")

### Perform asolute moves with different reference points ###

print("Absolute Moves are referenced from home")
position = 100
mm.emitAbsoluteMove(axis, position)
mm.waitForMotionCompletion()
print("Axis " + str(axis) + " is " + str(position) + "mm away from home.")

# Change the reference point to the current position
mm.setPosition(axis, 0)
print("Absolute moves on axis " + str(axis) + " are now referenced from " +  str(position) + "mm from home. ")
time.sleep(2)

# Move again
position2 = 30
print("Now moving to absolute position " + str(position2) + " mm, referenced from location 'setPosition' was called")
mm.emitAbsoluteMove(axis, position2)
mm.waitForMotionCompletion()
print("Axis " + str(axis) + " is now " + str(position2) + "mm from reference position and " + str(position + position2) + "mm from home")
```

stopContinuousMove (axis,accel)

Compatibility: MachineMotion v1 & MachineMotion v2

Description

Stops an axis using speed mode.

Parameters

- **axis** (Required) Number - Axis to move
- **accel** (Required) Number - Acceleration used to reach a null speed, in mm / sec^2

Return Value

None

MachineMotion V1 Example Code

```
#!/usr/bin/python
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases continuous moves with MachineMotion v1. ###

### MachineMotion CONFIGURATION ###

# Create MachineMotion instance
mm = MachineMotion()

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Configure actuator
conveyor_axis = 1
mm.configAxis(conveyor_axis, MICRO_STEPS.ustep_8, MECH_GAIN.roller_conveyor_mm_turn)

### CONTINUOUS MOVES ###

conveyor_stop_acc = 500 # Deceleration of 500mm/s^2

print("Start Conveyor Move...")
print("Continuous move: speed 100mm/s & acceleration 50mm/s^2")
mm.setContinuousMove(conveyor_axis, 100, 50)
time.sleep(5)

# Change speed while moving
print("Continuous move: speed 500mm/s & acceleration 250mm/s^2")
mm.setContinuousMove(conveyor_axis, 500, 250)
time.sleep(5)

# Stop the continuous move
mm.stopContinuousMove(conveyor_axis, conveyor_stop_acc)
time.sleep(2)

# Reverse direction of conveyor by changing the sign of the speed
print("Reverse continuous move: speed -1000mm/s & acceleration 500mm/s^2")
mm.setContinuousMove(conveyor_axis, -1000, 500)
time.sleep(5)

# Stop the continuous move
print("Stop Conveyor Move...")
mm.stopContinuousMove(conveyor_axis, conveyor_stop_acc)
time.sleep(3)
print("--> Example completed")
```

MachineMotion V2 Example Code

```
#!/usr/bin/python
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases continuous moves with MachineMotion v2. ###

### MachineMotion CONFIGURATION ###

# Create MachineMotion instance
mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Configure actuator
conveyor_axis = 1
mm.configServo(conveyor_axis, MECH_GAIN.roller_conveyor_mm_turn, DIRECTION.NORMAL, 5.0)

### CONTINUOUS MOVES ###

# Start the continuous move
print("Start Conveyor Move...")
print("Continuous move: speed 100mm/s & acceleration 50mm/s^2")
mm.setContinuousMove(conveyor_axis, 100, 50)
time.sleep(5)

# Change speed while moving
print("Continuous move: speed 500mm/s & acceleration 250mm/s^2")
mm.setContinuousMove(conveyor_axis, 500, 250)
time.sleep(5)

# Reverse direction of conveyor by changing the sign of the speed
print("Reverse continuous move: speed -1000mm/s & acceleration 500mm/s^2")
mm.setContinuousMove(conveyor_axis, -1000, 500)
time.sleep(5)

# Stop the continuous move
print("Stop Conveyor Move...")
mm.stopContinuousMove(conveyor_axis, 500)
time.sleep(3)
print("--> Example completed")
```

triggerEstop ( )

Compatibility: MachineMotion v1 & MachineMotion v2

Description

Triggers the MachineMotion software emergency stop, cutting power to all drives and enabling brakes (if any). The software E stop must be released (using releaseEstop()) in order to re-enable the machine.

Return Value

- The success of the operation.

MachineMotion V1 Example Code

```
#!/usr/bin/python
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases how to control eStop on MachineMotion v1. ###

### MachineMotion CONFIGURATION ###

# Create MachineMotion instance
mm = MachineMotion()
time.sleep(0.1) # Wait to initialize internal eStop topics.

# Define a callback to process estop status
def templateCallback(estop_status):
    print("eStop status is : " + str(estop_status))
    if estop_status == True :
        # executed when you enter estop
        print("MachineMotion is in estop state.")
    elif estop_status == False :
        # executed when you exit estop
        print("MachineMotion is not in estop state.")
mm.bindeStopEvent(templateCallback)

### ESTOP TRIGGER ###

result = mm.triggerEstop()
if result == True   : print("--> Software stop triggered")
else                : print("--> Failed to trigger software stop")

time.sleep(3.0)

### ESTOP RELEASE ###

result = mm.releaseEstop()
if result == True   : print("--> Software stop released")
else                : print("--> Failed to release software stop")

time.sleep(3.0)

### SYSTEM RESET ###

result = mm.resetSystem()
if result == True   : print("--> System has been reset")
else                : print("--> Failed to reset system")

print("--> Example completed")
```

MachineMotion V2 Example Code

```
#!/usr/bin/python
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases how to control eStop on MachineMotion v2. ###

### MachineMotion CONFIGURATION ###

# Create MachineMotion instance
mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)
time.sleep(0.1) # Wait to initialize internal eStop topics.

# Define a callback to process estop status
def templateCallback(estop_status):
    print("eStop status is : " + str(estop_status))
    if estop_status == True :
        # executed when you enter estop
        print("MachineMotion is in estop state.")
    elif estop_status == False :
        # executed when you exit estop
        print("MachineMotion is not in estop state.")
mm.bindeStopEvent(templateCallback)

### ESTOP TRIGGER ###

result = mm.triggerEstop()
if result == True   : print("--> Software stop triggered")
else                : print("--> Failed to trigger software stop")

time.sleep(3.0)

### ESTOP RELEASE ###

result = mm.releaseEstop()
if result == True   : print("--> Software stop released")
else                : print("--> Failed to release software stop")

time.sleep(3.0)

### SYSTEM RESET ###

result = mm.resetSystem()
if result == True   : print("--> System has been reset")
else                : print("--> Failed to reset system")

print("--> Example completed")
```

unlockBrake (aux_port_number,safety_adapter_presence)

Compatibility: MachineMotion v1 & MachineMotion v2

Description

Unlock the brake, by powering on the designated AUX port of the MachineMotion (24V).

Parameters

- **aux_port_number** (Required) Integer - The number of the AUX port the brake is connected to.
- **safety_adapter_presence** (Required) Boolean - Is a yellow safety adapter plugged in between the brake cable and the AUX port.

Return Value

None

NOTE: This function is compatible only with V1F and more recent MachineMotions.

MachineMotion V1 Example Code

```
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example control brakes on MachineMotion v1. ###

print(" ----- WARNING ------ Does your hardware version support brakes?")

mm = MachineMotion()

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Define the brake parameters
brake_port_number = 1           # The AUX port number the brake is plugged in
safety_adapter_presence = True  # Is a yellow safety adapter plugged in between the brake cable and the AUX port

# Read brake state
brake_state = mm.getBrakeState (brake_port_number, safety_adapter_presence)
print ("The brake connected to AUX" + str(brake_port_number) + " is : " + brake_state)

# Lock the brake
mm.lockBrake (brake_port_number, safety_adapter_presence)
time.sleep(0.2);                # Wait before reading brake state, for it to get correctly updated

# Read brake state
brake_state = mm.getBrakeState (brake_port_number, safety_adapter_presence)
print ("The brake connected to AUX" + str(brake_port_number) + " is : " + brake_state)

# DO NOT MOVE WHILE BRAKE IS ENGAGED
print ("Waiting two seconds. Do not move the actuator while the brake is locked !")
time.sleep(2)                   # Unlock the brake after two seconds

# Release the brakes
mm.unlockBrake (brake_port_number, safety_adapter_presence)
time.sleep(0.2);                # Wait before reading brake state, for it to get correctly updated

# Read brake state
brake_state = mm.getBrakeState (brake_port_number, safety_adapter_presence)
print ("The brake connected to AUX" + str(brake_port_number) + " is : " + brake_state)
```

MachineMotion V2 Example Code

```
import sys, time
sys.path.append("../..")
from MachineMotion import *

### This Python example control brakes on MachineMotion v2. ###

mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Define the brake parameters
axis = 1                        # Drive number
safety_adapter_presence = True  # Is a yellow safety adapter plugged in between the brake cable and the brake port
                                # Important Note : This flag must always be set to "True" on MachineMotions v2.

# Read brake state
brake_state = mm.getBrakeState(axis, safety_adapter_presence)
print ("The brake connected to port " + str(axis) + " is : " + brake_state)

# Lock the brake
mm.lockBrake(axis, safety_adapter_presence)
time.sleep(0.2);                # Wait before reading brake state, for it to get correctly updated

# Read brake state
brake_state = mm.getBrakeState(axis, safety_adapter_presence)
print ("The brake connected to port " + str(axis) + " is : " + brake_state)

# DO NOT MOVE WHILE BRAKE IS ENGAGED
print ("Waiting two seconds. Do not move the actuator while the brake is locked !")
time.sleep(2)                   # Unlock the brake after two seconds

# Release the brakes
mm.unlockBrake(axis, safety_adapter_presence)
time.sleep(0.2);                # Wait before reading brake state, for it to get correctly updated

# Read brake state
brake_state = mm.getBrakeState(axis, safety_adapter_presence)
print ("The brake connected to port " + str(axis) + " is : " + brake_state)
```

waitForMotionCompletion ( )

Compatibility: MachineMotion v1 & MachineMotion v2

Description

Pauses python program execution until machine has finished its current movement.

Return Value

None

NOTE: waitForMotionCompletion does not account for on-going continuous moves.

MachineMotion V1 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases how to wait for actuator motion completion on MachineMotion v1. ###

mm = MachineMotion()

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Configure the actuator
axis = 1
mm.configAxis(axis, MICRO_STEPS.ustep_8, MECH_GAIN.rack_pinion_mm_turn)

# Move the axis by 100mm
distance = 100
direction = DIRECTION.POSITIVE

print("Moving %d mm!"  % distance)
mm.emitRelativeMove(axis, direction, distance)
print("This message gets printed immediately")
mm.waitForMotionCompletion()
print("This message gets printed once machine has finished moving")
```

MachineMotion V2 Example Code

```
import sys
sys.path.append("../..")
from MachineMotion import *

### This Python example showcases how to wait for actuator motion completion on MachineMotion v2. ###

mm = MachineMotion(machineMotionHwVersion=MACHINEMOTION_HW_VERSIONS.MMv2)

# When starting a program, one must remove the software stop before moving
print("--> Removing software stop")
mm.releaseEstop()
print("--> Resetting system")
mm.resetSystem()

# Configure the actuator
axis = 1
mm.configServo(axis, MECH_GAIN.timing_belt_150mm_turn, DIRECTION.NORMAL, 5.0)

# Move the axis by 100mm
distance = 100
direction = DIRECTION.POSITIVE

print("Moving %d mm!"  % distance)
mm.emitRelativeMove(axis, direction, distance)
print("This message gets printed immediately")
mm.waitForMotionCompletion()
print("This message gets printed once machine has finished moving")
```