MSc in ROBOTICS

ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΕΛΛΑΔΟΣ

ΠΑΡΟΥΣΙΑΣΗ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ: ΚΑΤΑΣΚΕΥΗ ΚΑΙ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΡΟΜΠΟΤΙΚΟΥ ΒΡΑΧΙΟΝΑ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΣΑΓΡΗΣ ΔΗΜΗΤΡΙΟΣ
ΟΝΟΜΑ ΦΟΙΤΗΤΗ: ΤΡΑΠΑΛΗΣ ΞΕΝΟΦΩΝ

ΕΙΣΑΓΩΓΗ

Η ρομποτική τις τελευταίες δεκαετίες φέρει μεγάλη ανάπτυξη σε πολλούς τομείς επαγγελμάτων. Ένας απ αυτούς είναι και η βιομηχανία. Στο κομμάτι της βιομηχανίας έχουμε μπει στην 4η βιομηχανική επανάσταση, όπου στόχος της είναι όχι μόνο η αυτοματοποίηση των γραμμών παραγωγής με ρομπότ αλλά η αυτοματοποίηση ολόκληρων μονάδων παραγωγής.

Για τον λόγο αυτό λοιπόν η παρούσα εργασία ασχολείται με την κατασκευή και τον προγραμματισμό ενός ρομποτικού βραχίονα.

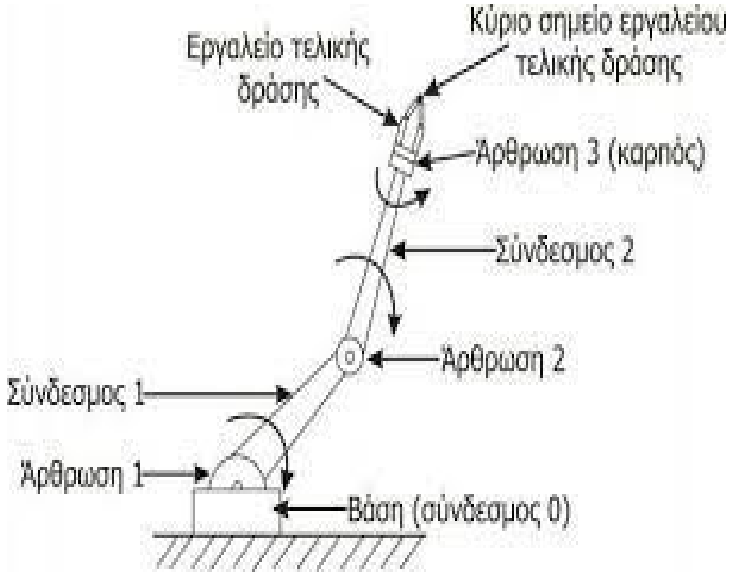
ΑΝΤΙΚΕΙΜΕΝΟ ΕΡΓΑΣΙΑΣ

Το αντικείμενο της εργασίας αφορά τη μελέτη του κινηματικού προβλήματος ενός ρομποτικού βραχίονα έξι (6) αξόνων και τον προγραμματισμό αυτού με CODESYS PLC σύμφωνα με το διεθνές πρότυπο IEC 61131-3 με χρήση της βιβλιοθήκης SoftMotion. Επιπέον,χρησιμοποιείται το πρόγραμμα CoppeliaSim το οποίο είναι πρόγραμμα προσομοίωσης V-rep (virtual robot exrerimentation platform). Η επικοινωνία CODESYS PLC και CoppeliaSim γίνεται με Modbus TCP μέσω Python.

ΔΟΜΗ ΡΟΜΠΟΤΙΚΟΥ ΒΡΑΧΙΟΝΑ

Ο ρομποτικός μας βραχίονας αποτελείται από:

- Την βάση του
- Τους συνδέσμους
- Τις αρθρώσεις
- Το εργαλείο τελικής δράσης
- Το κύριο σημείο του εργαλείου τελικής δράσης



ΚΙΝΗΜΑΤΙΚΗ

Η κινηματική είναι ο κλάδος της φυσικής που ασχολείται με την γεωμετρική κίνηση των σωμάτων και περιγράφει:

- Την ταχύτητα
- Τη μετατόπιση
- Τον χρόνο και
- Επιτάχυνση

ΚΙΝΗΜΑΤΙΚΗ

Κινηματική αλυσίδα ονομάζεται η διαδοχική σύνδεση των συνδέσμων και των αρθρώσεων του ρομπότ. Ένας ρομποτικός βραχίονας που αποτελείται από n αρθρώσεις θα έχει n + 1 συνδέσμους. Για την επίλυση του κινηματικού προβλήματος αριθμούμε τις αρθρώσεις του βραχίονα από το 0 έως το n και τους συνδέσμους από το 1 έως το n + 1. Ως μηδενική άρθρωση θεωρείται η βάση του βραχίονα.

Για την επίλυση του κινηματικού προβλήματος επιλύσουμε τον ομογενή μετασχηματισμό για κάθε άρθρωση του ρομπότ:

$$H_n^{n-1} = \begin{bmatrix} R_n^{n-1} & d_n^{n-1} \\ 0 \times 3 & 1 \end{bmatrix}$$

ΚΙΝΗΜΑΤΙΚΗ

Στην περίπτωση που η περιστροφή ενός άξονα είναι συνδυαστική τότε οι πίνακες περιστροφής συνδυάζονται αναλόγως.

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \qquad R_y = \begin{bmatrix} \cos\theta & 0 & \sin\vartheta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\vartheta \end{bmatrix} \qquad R_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

ΚΙΝΗΜΑΤΙΚΗ

Στην συνέχεια θα πρέπει να υπολογιστεί το διάνυσμα μετατόπισης της άρθρωσης n ως προς την άρθρωση n – 1. Το διάνυσμα μετατόπισης αποτελείται από μόνο μία στήλη όπου η κάθε γραμμή αντιπροσωπεύει την μετατόπιση x, y, z αντίστοιχα της άρθρωσης n ώς προς την άρθρωση n – 1.

$$d_n^{n-1} = \begin{bmatrix} x_n^{n-1} \\ y_n^{n-1} \\ z_n^{n-1} \end{bmatrix}$$

Σύμφωνα με τα παραπάνω ο τελικός ομογενής μ $H_n^0 = \begin{bmatrix} R_n^0 & d_n^0 \\ 3\times 0 & 1 \end{bmatrix}$ **ρομποτικό βραχίονα είναι ο εξής:**

Denavit – Hartenberg

Κανόνες εύρεσης πλαισίων αναφοράς:

- Ο z άξονας θα πρέπει να τοποθετηθεί κατά μήκος του άξονα περιστροφής.
- Ο x άξονας θα πρέπει να είναι συγγραμικός με την κοινή νοητή γραμμή που είναι κάθετη και με τις δύο αρθρώσεις.
- Ο y άξονας βρίσκεται με τον κανόνα του δεξιού χεριού.

Denavit – Hartenberg

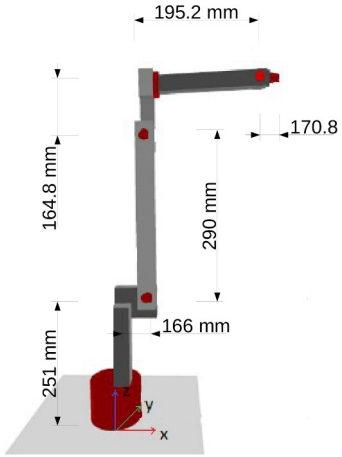
Η μέθοδος Denavit – Hartenberg είναι η πιο διαδεδομένη μέθοδος επίλυσης του κινηματικού προβλήματος. Η μέθοδος αυτή είναι ένα σύνολο κανόνων η οποία ορίζει το πως πρέπει να οριστούν τα πλαίσια αναφοράς που σχετίζονται με κάθε σύνδεσμο. Ξεκινώντας από την βάση του ρομπότ κάθε πλαίσιο αναφοράς μπορεί να υπολογιστεί από 4 παραμέτρους γνωστές ως Denavit – Hartenberg παράμετροι.

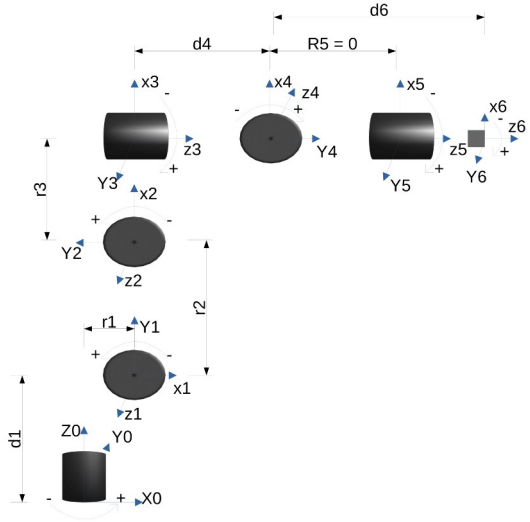
- d: είναι η απόσταση από την βάση του προηγούμενου z άξονα έως την νοητή κάθετη γραμμή των δύο αρθρώσεων που μελετούνται.
- θ: είναι η γωνία του προηγούμενου z άξονα σε σχέση με τον z άξονα που μελετάμε μετατοπίζοντας τον προηγούμενο x άξονα κατά τον καινούργιο.
- r: είναι η απόσταση κατά των x άξονα που μελετάμε μεταξύ του προηγούμενου z άξονα και του z αξονα του πλαισίου αναφοράς που μελετάμε.
- α: είναι η γωνία που δημιουργείται μετατοπίζοντας των προηγούμενο z άξονα ως προς των z άξονα του πλαισίου που μελετάμε.

Είναι μία μέθοδος που μας επιτρέπει να χρησιμοποιήσουμε τις λιγότερες δυνατές παραμέτρους για την επίλυση του κινηματικού προβλήματος και χρησιμοποιείται ευρέος.

Παράμετροι Denavit – Hertenberg του ρομπότ

Ο ρομποτικός βραχίονας του οποίου θα μελετήσουμε την κινηματική είναι ένα ρομπότ έξι βαθμών ελευθερίας.





Παράμετροι Denavit – Hertenberg του ρομπότ

Για την επίλυση του κινηματικού προβλήματος ξεκινάμε πάντα από την βάση του ρομπότ. Μελετάμε κάθε n άξονα σε σχέση με τον n – 1 άξονα και επιλύουμε έτσι ώστε να εντοπίσουμε της παραμέτρους του Denavit – Hartenberg για κάθε άξονα ωσότου έχουμε τον πλήρη πίνακα των παραμέτρων για όλους τους άξονες του ρομπότ. Παράλληλα δημιουργούμε τον πίνακα ομογενούς μετασχηματισμού για κάθε άξονα που μελετάμε.

$$
{}^{n-1}T_n = \left[\begin{array}{ccc|c} \cos(\theta_n) & -\sin(\theta_n)\cdot\cos(\alpha_n) & \sin(\theta_n)\cdot\sin(\alpha_n) & r_n\cdot\cos(\theta_n) \\ \sin(\theta_n) & \cos(\theta_n)\cdot\cos(\alpha_n) & -\cos(\theta_n)\cdot\sin(\alpha_n) & r_n\cdot\sin(\theta_n) \\ 0 & \sin(\alpha_n) & \cos(\alpha_n) & d_n \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{c|c} R & T \\ \hline 0\ \ 0\ \ 0 & 1 \end{array}\right]
$$

Link (n)	$d_n(m)$	$\theta_n(deg)$	$r_n(m)$	$\alpha_n(deg)$
1	$d_1=0.251\,m$	$\theta_1=0^\circ$	$r_1=0.166\,m$	$\alpha_1=90^\circ$
2	$d_2=0\,m$	$\theta_2=90^\circ$	$r_2=0.290\,m$	$\alpha_2=0^\circ$
3	$d_3=0\,m$	$\theta_3=90^\circ$	$r_3=0.1648\,m$	$\alpha_3=90^\circ$
4	$d_4=0.1952\,m$	$\theta_4=0^\circ$	$r_4=0\,m$	$\alpha_4=90^\circ$
5	$d_5=0\,m$	$\theta_5=0^\circ$	$r_5=0\,m$	$\alpha_5=-90^\circ$
6	$d_6=0.1708\,m$	$\theta_6=0^\circ$	$r_6=0\,m$	$\alpha_6=0^\circ$

Παράμετροι Denavit – Hertenberg του ρομπότ

Για να βρούμε τον πίνακα του ομογενούς μετασχηματισμού από την βάση του ρομπότ έως το τελικό σημείο του ρομπότ αρκεί να πολλαπλασιάσουμε τους πίνακες ομογενούς μετασχηματισμού των επιμέρους συνδέσμων:

$$^{0}T_{6} = {}^{0}T_{1} * {}^{1}T_{2} * {}^{2}T_{3} * {}^{3}T_{4} * {}^{4}T_{5} * {}^{5}T_{6}$$

Λύνοντας τον παραπάνω πολλαπλασιασμό των ομογενών πινάκων καταλήγουμε στο παρακάτω αποτέλεσμα

$$^{0}T_{6} = \begin{bmatrix} 0 & 0 & 1 & 0.532 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0.7058 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & R & & T \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Προγραμματιστικό μέρος

- **CODESYS (περιβάλλον ανάπτυξης προγραμματισμού εφαρμογών ελεγκτή).**
- **SoftMotion (βιβλιοθήκη η οποία ενσωματώνεται στο CODESYS και χρησιμοποιείται για τον έλεγχο κίνησης οποιουδήποτε κινηματικού συστήματος).**
- **HMI (Human Machine Interface).**
- **CoppeliaSim (V – Rep) (πρόγραμμα προσομοίωσης).**
- **LUA (κώδικας στο CoppeliaSim).**
- **Modbus TCP (Πρωτόκολλο επικοινωνίας).**
- **Python**

SoftMotion

- Αποτελείται από πολλά μπλοκ λειτουργιών τα οποία καλύπτουν όλες τις απαιτήσεις για την κίνηση ενός κινητήρα, ενδεικτικά, μερικά από τα πιο βασικά για την λειτουργία ενός Servo κινητήρα, είναι τα παρακάτω:

- MC_Power
- MC_Reset
- MC_Stop
- MC_Jog
- MC_Home
- MC_MoveAbsolute
- MC_MoveRelative
- SMC_SetMovementType
- SMC_SetSoftwareLimits

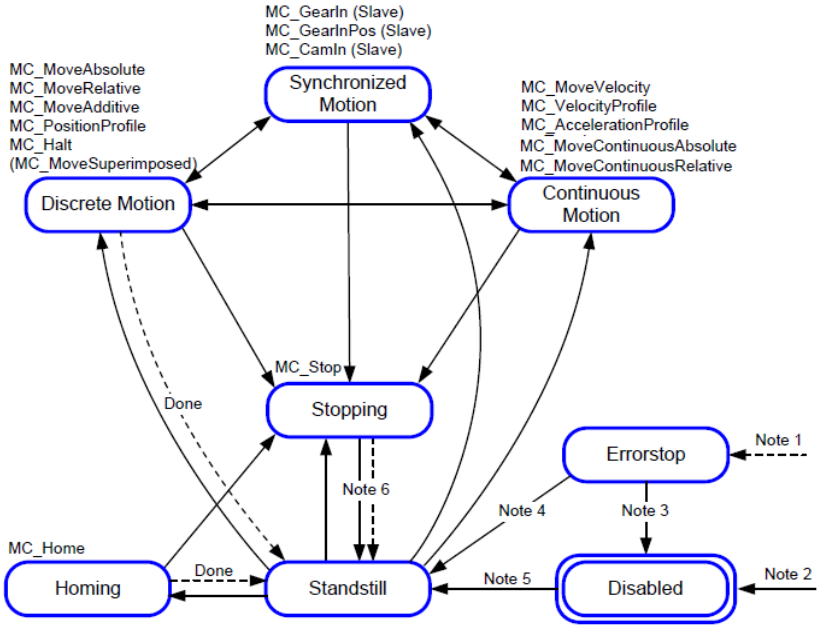
SoftMotion

Επιπλέον υπάρχουν και μπλοκ τα οποία αφορούν ολόκληρα γκρουπ αφού έχουμε ορίσει την κινηματική του γκρουπ.

- MC_GroupPower
- MC_GroupEnable
- MC_GroupDisable
- MC_GroupReset
- MC_GroupStop
- MC_Jog
- MC_MoveDirectAbsolute
- MC_MoveLinearAbsolute
- MC_SetCoordinateTransform
- SMC_Interpolator
- SMC_NCInterpreter
- SMC_ReadNCFile2

SoftMotion

Σύμφωνα με το SoftMotion οι πιθανές καταστάσεις ενός άξονα είναι οι παρακάτω και υπάρχουν συγκεκριμένες συνθήκες για να μεταπηδήσει από την μία κατάσταση στην άλλη. Πάντα ξεκινάει από την κατάσταση απενεργοποιημένος (Disabled/ power_Off) και καταλήγει στην κατάσταση ακινητοποιημένος (Standstill) αφού πρώτα έχει ενεργοποιηθεί (MC_Power).



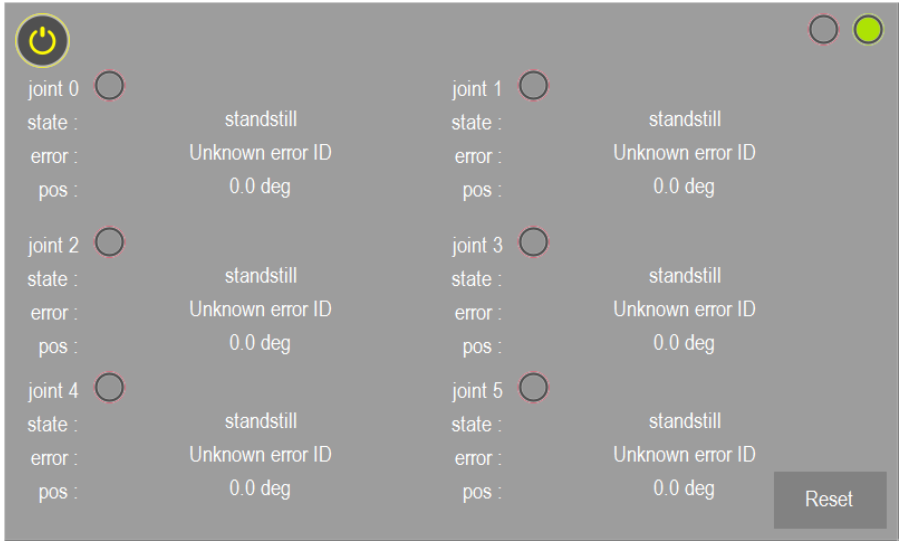
Note 1: From any state. An error in the axis has occured.
Note 2: From any state. MC_Power.Enable = FALSE . There is no error in the axis.
Note 3: MC_Reset and MC_Power.Status = FALSE
Note 4: MC_Reset and MC_Power.Status = TRUE and MC_Power.Enable = TRUE
Note 5: MC_Power.Enable = TRUE and MC_Power.Status = TRUE
Note 6: MC_Stop.Done = TRUE and MC_Stop.Execute = FALSE

HMI

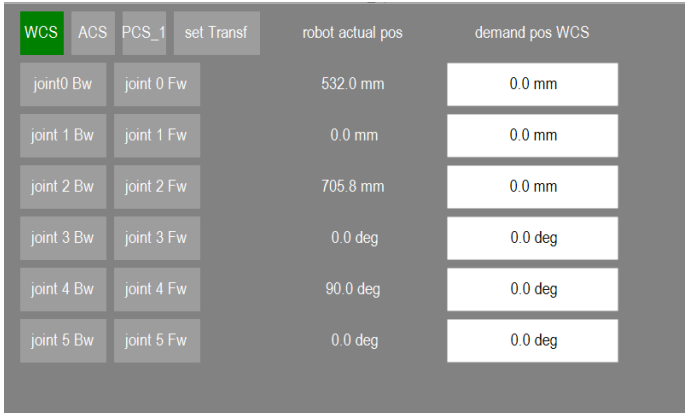
Το Human Machine Interface ή αλλιώς HMI είναι η οθόνη που έχει ο χειριστής έτσι ώστε να μπορεί να χειρίζεται το εκάστοτε σύστημα. Είναι ο τρόπος για να μπορεί ο χειριστής να επικοινωνεί με την μηχανή. Στο CODESYS υπάρχει η επιλογή να δημιουργήσεις το HMI είτε κατευθείαν στο PLC αν σε αυτό υπάρχει και οθόνη είτε στο δίκτυο σε σελίδα .htm με διεύθυνση http://localhost:8080/webvisu.htm , όπου localhost θέτουμε την IP του PLC.

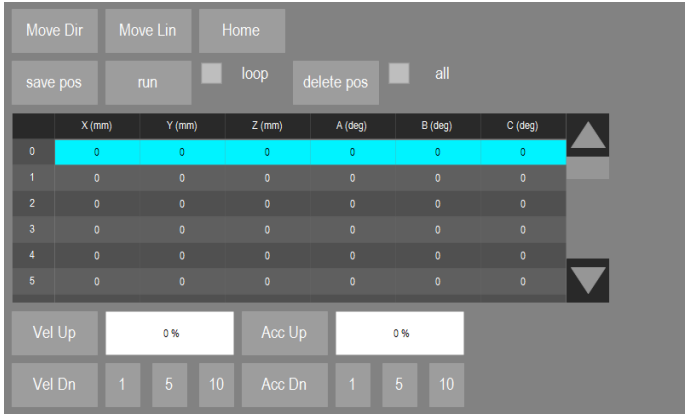
HMI

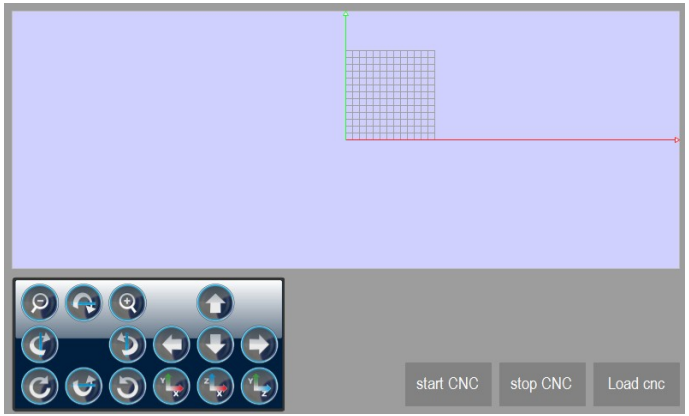
Συνολικά έχουν δημιουργηθεί πέντε καρτέλες για τον συνολικό έλεγχο του ρομπότ.

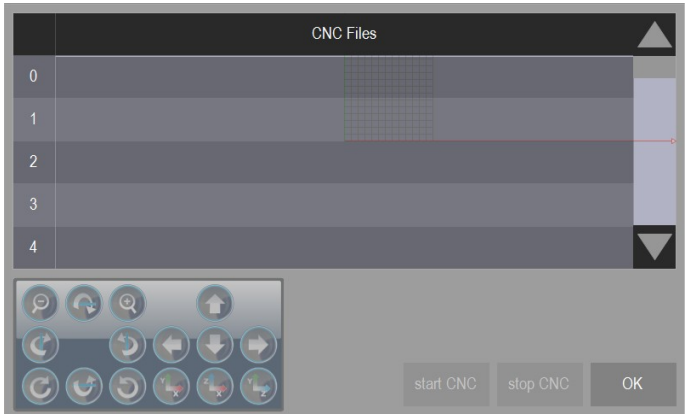


HMI


WCS
ACS
PCS_1
set Transf
robot actual pos
demand pos WCS
joint0 Bw
joint 0 Fw
532.0 mm
0.0 mm
joint 1 Bw
joint 1 Fw
0.0 mm
0.0 mm
joint 2 Bw
joint 2 Fw
705.8 mm
0.0 mm
joint 3 Bw
joint 3 Fw
0.0 deg
0.0 deg
joint 4 Bw
joint 4 Fw
90.0 deg
0.0 deg
joint 5 Bw
joint 5 Fw
0.0 deg
0.0 deg


Move Dir
Move Lin
Home
save pos
run
loop
delete pos
all
X (mm)
Y (mm)
Z (mm)
A (deg)
B (deg)
C (deg)
Vel Up
0 %
Acc Up
0 %
Vel Dn
1
5
10
Acc Dn
1
5
10


start CNC
stop CNC
Load cnc

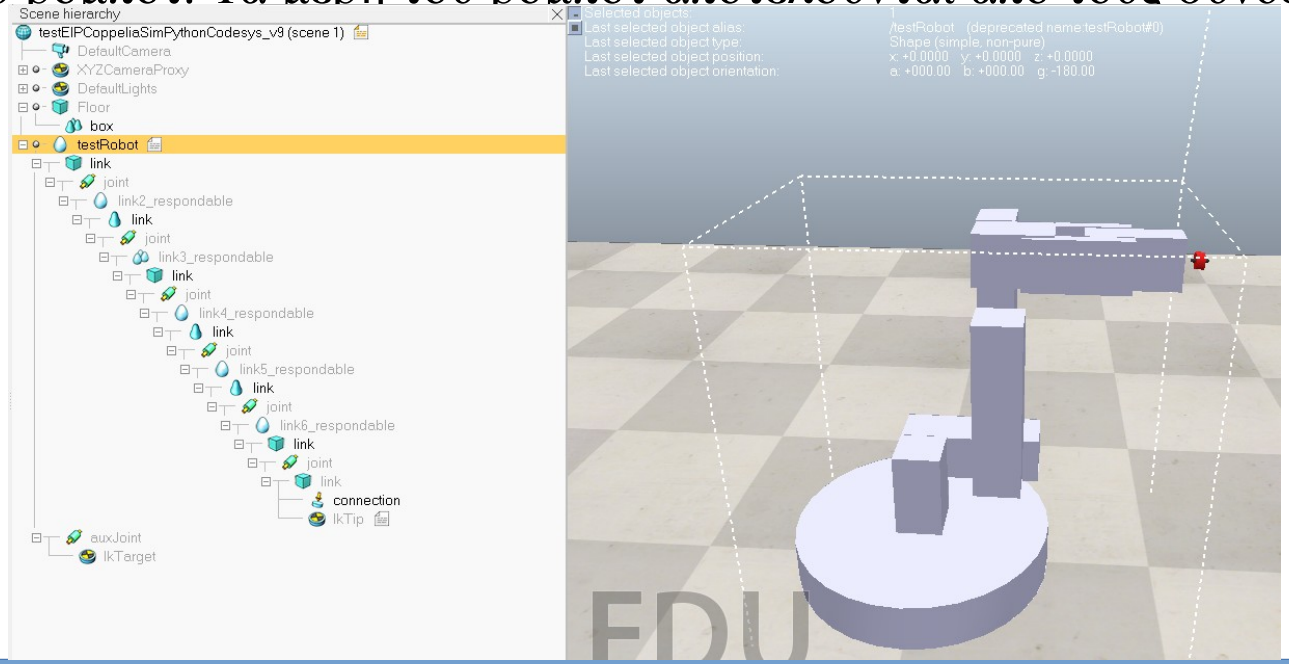

CNC Files
start CNC
stop CNC
OK

CoppeliaSim

Για την οπτική απεικόνιση του ρομπότ χρησιμοποιήθηκε το CoppeliaSim. Το CoppeliaSim γνωστό ως V – Rep είναι ένα πρόγραμμα το οποίο χρησιμοποιείται στην βιομηχανία, στην εκπαίδευση αλλά και στην έρευνα. Αρχικά αναπτύχθηκε από την Toshiba R&D και πλέον αναπτύσεται από την Robotics AG. Είναι ένα πρόγραμμα προσομοίωσης το οποίο υποστηρίζει Python και Lua κώδικα ακόμα και C/C++. Επιπλέον μπορεί ασύγχρονα να συνεργάζεται με εξωτερικά προγράμματα όπως το ROS, Remote API, ZeroMQ με γλώσσες όπως python, Java, C/C++ και Matlab. Το CoppeliaSim μπορεί να επιλύσει κινηματικά προβλήματα καθώς επίσης υποστηρίζει διάφορες βιβλιοθήκες φυσικής όπως MuJoCo, Bullet, ODE, Vortex, Newton Game Dynamics.

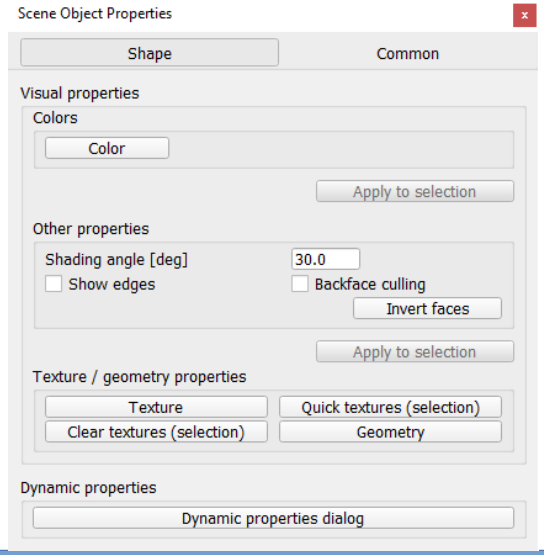
CoppeliaSim

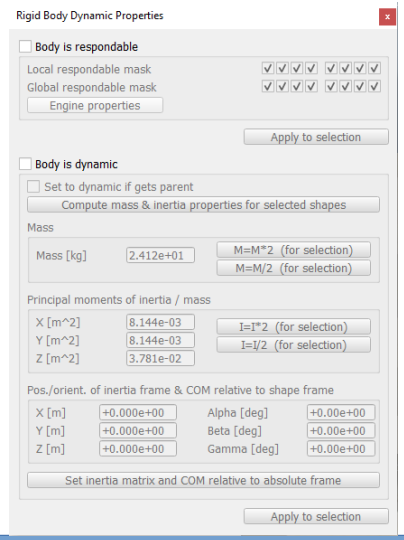
Για τον σχεδιασμό του ρομπότ ξεκινάμε δημιουργώντας το δέντρο ιεραρχίας των επιμέρους μερών του ρομπότ. Τα μέρη του ρομπότ αποτελούνται από τους συνδέσμους και τις αρθρώσεις.

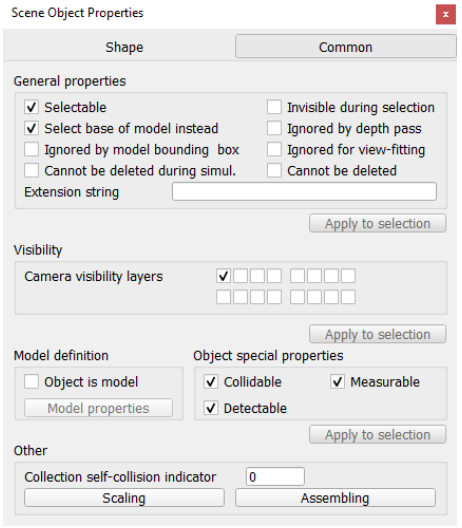


CoppeliaSim

Κάθε αντικείμενο που προσθέτουμε στο σχέδιο μας επιτρέπει να του ορίσουμε τις ιδιότητές του. Στους συνδέσμους μας δίνεται η δυνατότητα να ορίσουμε την γεωμετρία του αντικειμένου, τα texture του αντικειμένου καθώς επίσης και τις δυναμικές ιδιότητές του.

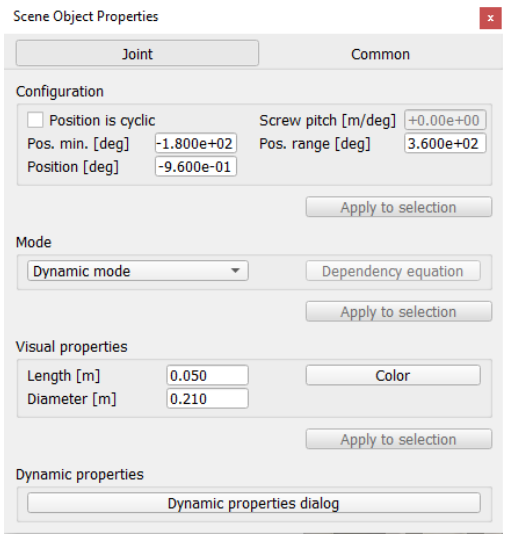


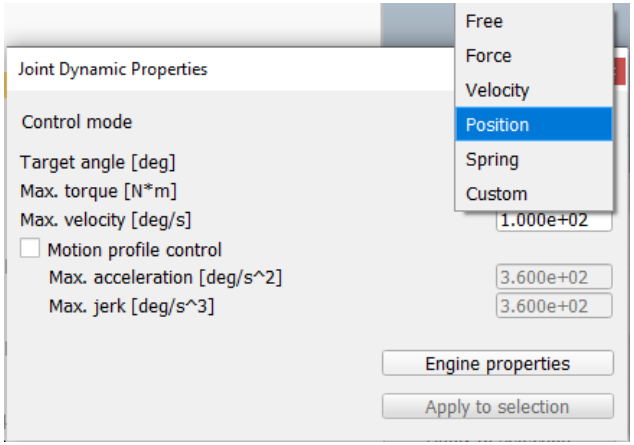




CoppeliaSim

Επιπλέον στης αρθρώσεις μπορούμε να ορίσουμε τον τύπο της άρθρωσης που έχουμε, γραμμική ή κυκλική, και στις δυναμικές ιδιότητες μπορούμε να ορίσουμε τον τύπο της κίνησης που θα κάνει η άρθρωση.





CoppeliaSim - LUA

Για την κίνηση του ρομπότ στο CoppeliaSim αλλά και για την επικοινωνία με το Codesys πρέπει να δημιουργήσουμε κώδικα σε LUA στο CoppeliaSim. Η επικοινωνία στο CoppeliaSim γίνεται με το Remote API.

```
-- Functions called by the legacy remote API client:
-------------------------------------------------------
function legacyRemoteApi_movementDataFunction(intData,floatData,stringData,buffer)
    if not messagePack then
        messagePack=require('messagePack')
        messagePack.set_string('string')
    end
    local movData=messagePack.unpack(buffer)
    allMovementData[movData.id]=movData
    return {},{},{},''
end

function legacyRemoteApi_executeMovement(intData,floatData,stringData,buffer)
    movementToExecute[#movementToExecute+1]=buffer
    return {},{},{},''
end
```

CoppeliaSim - LUA

```lua
function sysCall_init()
    stringSignalName='/testRobot_executedMovId'
    movementToExecute={}
    allMovementData={}
    currentVel={0,0,0,0,0,0,0}
    currentAccel={0,0,0,0,0,0,0}
    maxJerk={100,100,100,100,100,100}
    sim.setStringSignal(stringSignalName,'ready')
    jointHandles={-1,-1,-1,-1,-1,-1}
    currentPosVelAccel={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
    for i=1,6,1 do
        jointHandles[i]=sim.getObject('./joint',{index=i-1})
    end
    corout = coroutine.create(coroutineStart)
end
```

```lua
function coroutineStart()
    while true do
        if #movementToExecute>0 then
            local id=table.remove(movementToExecute,1)
            local movementData=allMovementData[id]
            allMovementData[id]=nil
            if movementData.type=='mov' then
                local currentConfig={}
                for i=1,#jointHandles,1 do
                    currentConfig[i]=sim.getJointPosition(jointHandles[i])
                end
                newPos,currentVel,currentAccel=sim.moveToConfig(-1,currentConfig,currentVel
                print("maxVel", movementData.maxVel)
            end
            if movementData.type=='pts' then
                executePtpMovement(jointHandles,movementData)
            end
            print(movementData.type)
            sim.setStringSignal(stringSignalName,id)
        else
            sim.switchThread()
        end
    end
end
```

```lua
function executePtpMovement(handles,data)
    -- Apply joint configs in an interpolated manner:
    local lb=sim.setThreadAutomaticSwitch(false)
    local path={}
    for i=1,#data.times,1 do
        path[(i-1)*6+1]=data.j1[i]
        path[(i-1)*6+2]=data.j2[i]
        path[(i-1)*6+3]=data.j3[i]
        path[(i-1)*6+4]=data.j4[i]
        path[(i-1)*6+5]=data.j5[i]
        path[(i-1)*6+6]=data.j6[i]
    end
    local startTime=sim.getSimulationTime()
    local t=0
    while t<data.times[#data.times] do
        local conf=sim.getPathInterpolatedConfig(path,data.times,t)
        applyJointTargetPositions(handles,conf)
        sim.switchThread()
        t=sim.getSimulationTime()-startTime
    end
    local conf=sim.getPathInterpolatedConfig(path,data.times,data.times[#data.times])
    applyJointTargetPositions(handles,conf)
    sim.setThreadAutomaticSwitch(lb)
end
```

Modbus TCP / Python

Η Python χρησιμοποιείται για την επικοινωνία του CODESYS PLC με το CoppeliaSim. Στον κώδικα της Python εκτελείται το Modbus TCP για την επικοινωνία με το PLC και την λήψη των θέσεων των αξόνων του ρομποτικού βραχίονα κατά την εκτέλεση της κίνησης σε πραγματικό χρόνο. Ταυτόχρονα εκτελείται το Remote API για την επικοινωνία με το CoppeliaSim και να έχουμε την τρισδιάστατη απεικόνιση της κίνησης. Οι βιβλιοθήκες που χρησιμοποιούνται είναι:

- **Sim (βιβλιοθήκη του CoppeliaSim για την αποστολή των δεδομένων).**
- **Msgpack (επιτρέπει την αποστολή πακέτων δεδομένων).**
- **PyModbusTCP (βιβλιοθήκη για το Modbus TCP).**

Modbus TCP / Python

```python
ModClient = ModbusClient(host='192.168.1.49', debug=False, auto_open=True)

class Client:
    def __enter__(self):
        self.executedMovId1 = 'notReady'
        sim.simxFinish(-1)  # just in case, close all opened connections
        self.id = sim.simxStart('127.0.0.1', 19997, True, True, 5000, 5)  # Connect to CoppeliaSim
        return self

    def __exit__(self, *err):
        sim.simxFinish(-1)
        print('Program ended')


with Client() as client:
    print("running")

    if client.id != -1:
        print('Connected to remote API server')

        targetArm = '/testRobot'

        client.stringSignalName1 = targetArm + '_executedMovId'
```

```python
while True:
    # read 10 registers at address 0, store result in regs list
    ModbusTCPDataRead = ModClient.read_input_registers(0, 14)

    if ModbusTCPDataRead:
        # positions over Modbus TCP
        joint1 = (ModbusTCPDataRead[0] - c) / m
        joint2 = (ModbusTCPDataRead[1] - c) / m
        joint3 = (ModbusTCPDataRead[2] - c) / m
        joint4 = (ModbusTCPDataRead[3] - c) / m
        joint5 = (ModbusTCPDataRead[4] - c) / m
        joint6 = (ModbusTCPDataRead[5] - c) / m
        maxAcc = ModbusTCPDataRead[6]
        vel = ModbusTCPDataRead[7]
        maxVel = ModbusTCPDataRead[8]
        acc = ModbusTCPDataRead[9]
    else:
        print('unable to read registers')
```

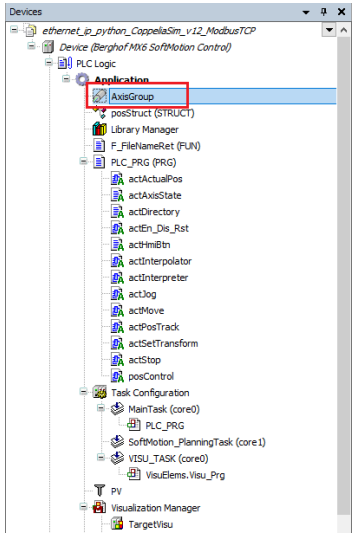
Modbus TCP / Python

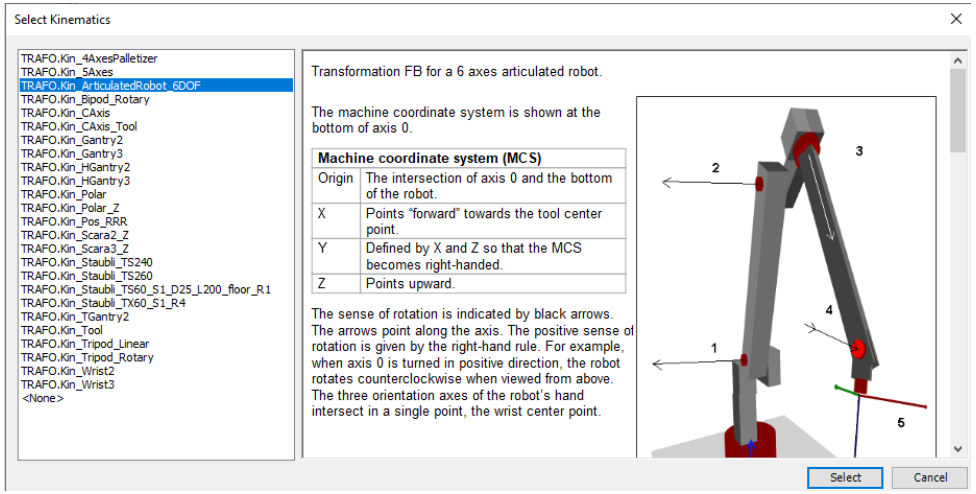
```
if joint1 != prJoint1 or joint2 != prJoint2 or joint3 != prJoint3 or joint4 != prJoint4 or joint5 != prJoint5 or joint6 != prJoint6:

    # Send first movement sequence:
    targetConfig = [joint1, joint2, joint3, joint4, joint5,
                    joint6]
    movSeq = "movSeq" + str(posCount)
    movementData = {"id": movSeq, "type": "mov", "targetConfig": targetConfig,
                    "maxVel": targetVel, "maxAccel": currentAccel}
    packedMovementData = msgpack.packb(movementData)
    sim.simxCallScriptFunction(client.id, targetArm, sim.sim_scripttype_childscript,
                               'legacyRemoteApi_movementDataFunction', [], [], [],
                               packedMovementData,
                               sim.simx_opmode_oneshot)

    # Execute first movement sequence:
    sim.simxCallScriptFunction(client.id, targetArm, sim.sim_scripttype_childscript,
                               'legacyRemoteApi_executeMovement', [], [], [], movSeq,
                               sim.simx_opmode_oneshot)

    # Wait until above movement sequence finished executing:
    waitForMovementExecuted1(movSeq)
```

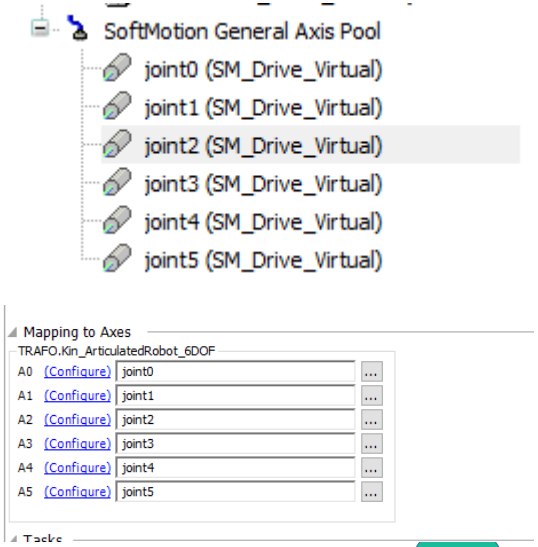
```
def waitForMovementExecuted1(id):
    while client.executedMovId1 != id:
        retCode, s = sim.simxGetStringSignal(client.id, client.stringSignalName1, sim.simx_opmode_buffer)
        if retCode == sim.simx_return_ok:
            if type(s) == bytearray:
                s = s.decode('ascii')  # python2/python3 differences
            client.executedMovId1 = s
            return True
```

CODESYS

- προσθέτουμε ένα γκρουπ αξόνων.
- Επιλέγουμε το κινηματικό που θέλουμε να επιλύσουμε.
- Ορίζουμε τους άξονες στο γκρουπ των αξόνων.



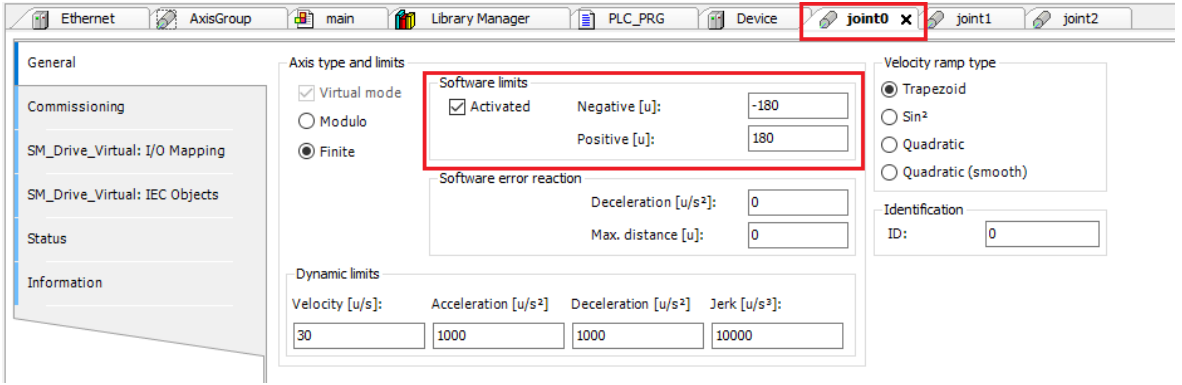




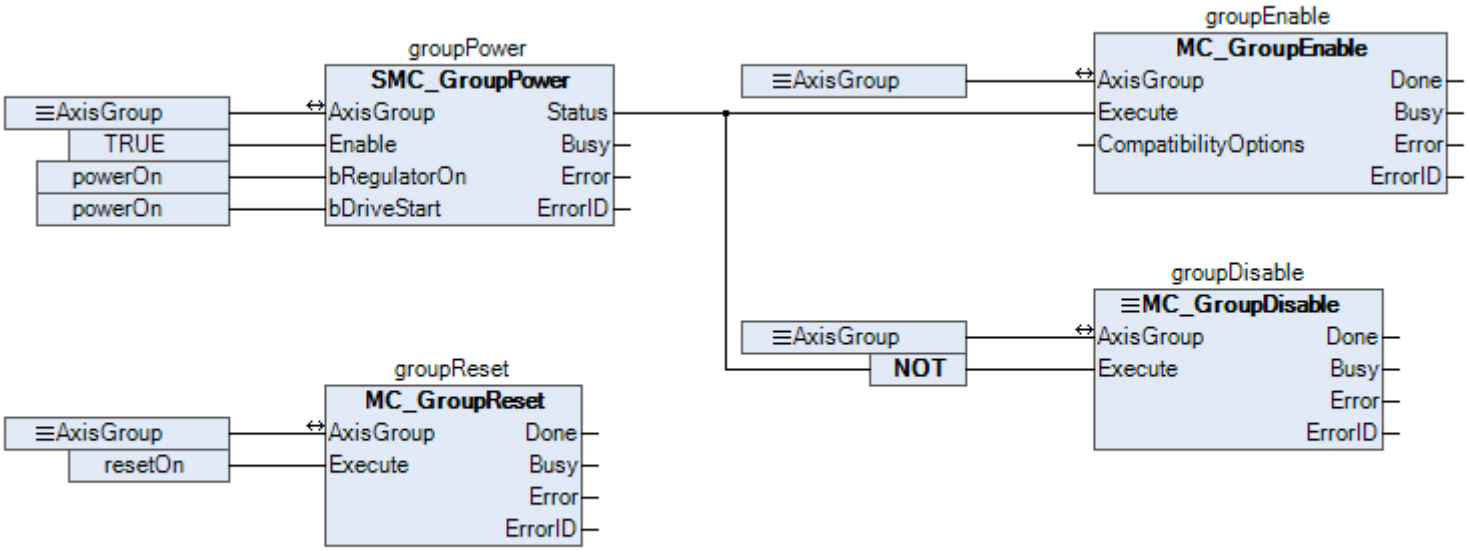
CODESYS

Από το κινηματικό δίνονται τα όρια του κάθε άξονα τα οποία τα ορίζουμε στους άξονες που έχουμε ορίσει.

Axis	Configurable	Default	Min/Max
a0	YES	[-180°, 180°]	Unlimited
a1	YES	[-180°, 180°]	Unlimited
a2	YES	[-90°, 180°]	Unlimited
a3	YES	[-180°, 180°]	Unlimited
a4	YES	[-180°, 180°]	[-180°, 180°]
a5	YES	[0°, 360°]	Unlimited



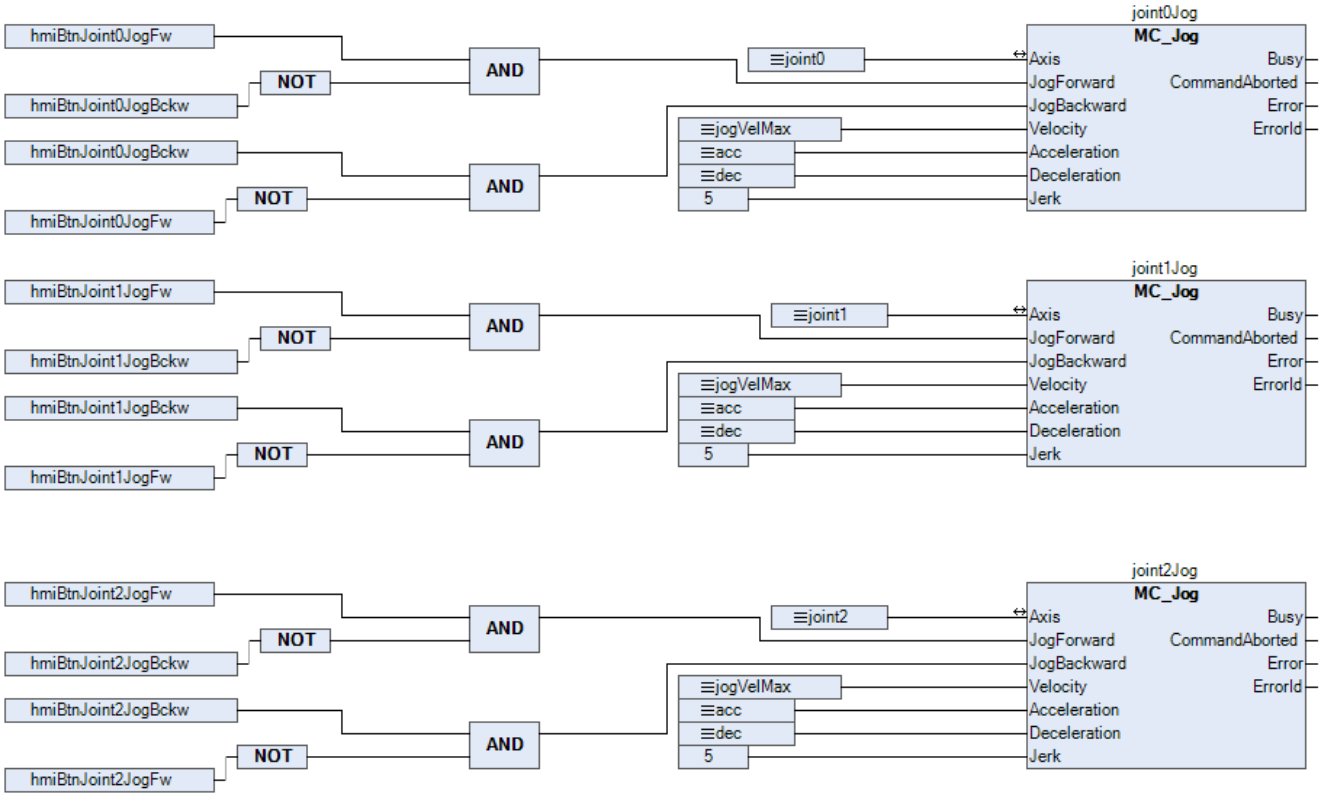
CODESYS


groupPower
SMC_GroupPower
≡AxisGroup
AxisGroup
Status
TRUE
Enable
Busy
powerOn
bRegulatorOn
Error
powerOn
bDriveStart
ErrorID
groupEnable
MC_GroupEnable
≡AxisGroup
AxisGroup
Done
Execute
Busy
CompatibilityOptions
Error
ErrorID
groupDisable
≡MC_GroupDisable
≡AxisGroup
AxisGroup
Done
NOT
Execute
Busy
Error
ErrorID
groupReset
MC_GroupReset
≡AxisGroup
AxisGroup
Done
resetOn
Execute
Busy
Error
ErrorID

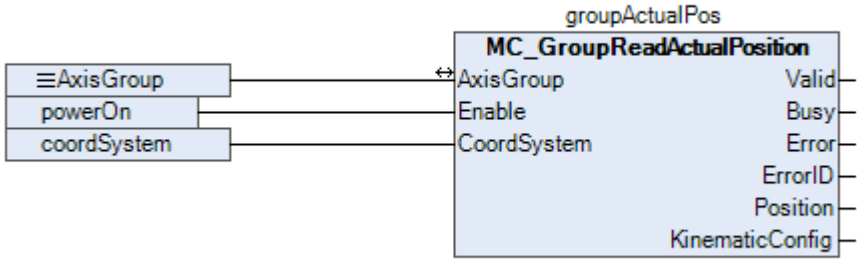
CODESYS

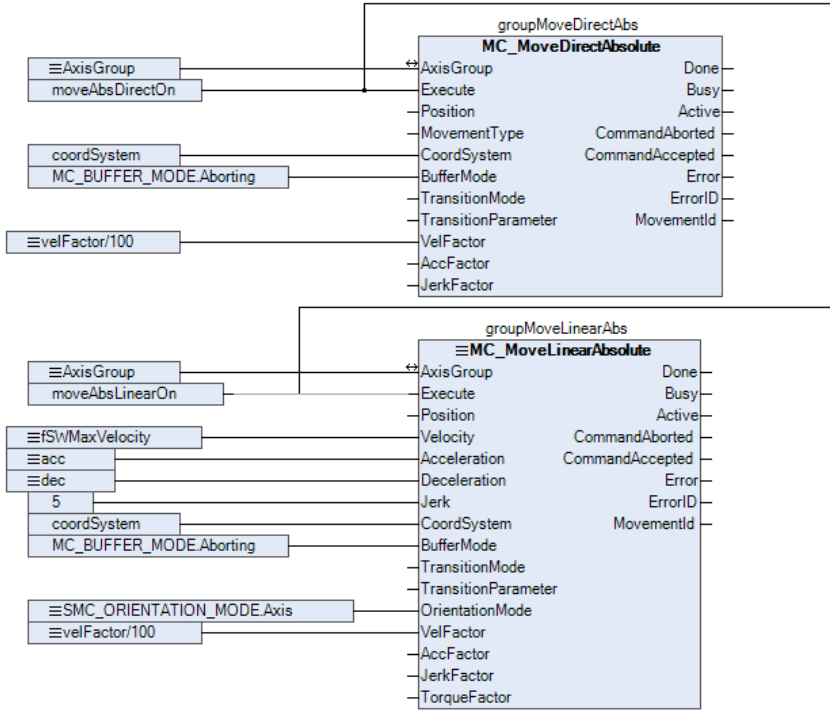
```
IF joint0.nAxisState = 0 THEN
    joint0State     :=  'power_off';
ELSIF joint0.nAxisState = 1 THEN
    joint0State     :=  'errorstop';
    joint0Error     :=  TRUE;
ELSIF joint0.nAxisState = 2 THEN
    joint0State     :=  'stopping';
ELSIF joint0.nAxisState = 3 THEN
    joint0State     :=  'standstill';
ELSIF joint0.nAxisState = 4 THEN
    joint0State     :=  'discrete_motion';
ELSIF joint0.nAxisState = 5 THEN
    joint0State     :=  'continuous_motion';
ELSIF joint0.nAxisState = 6 THEN
    joint0State     :=  'synchronized_motion';
ELSIF joint0.nAxisState = 7 THEN
    joint0State     :=  'homing';
END_IF
```

CODESYS

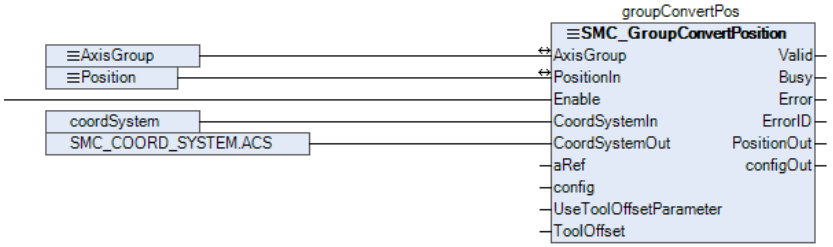

hmiBtnJoint0JogFw
NOT
hmiBtnJoint0JogBckw
hmiBtnJoint0JogBckw
NOT
hmiBtnJoint0JogFw
AND
AND
≡joint0
≡jogVelMax
≡acc
≡dec
5
joint0Jog
MC_Jog
Axis
JogForward
JogBackward
Velocity
Acceleration
Deceleration
Jerk
Busy
CommandAborted
Error
ErrorId
hmiBtnJoint1JogFw
NOT
hmiBtnJoint1JogBckw
hmiBtnJoint1JogBckw
NOT
hmiBtnJoint1JogFw
AND
AND
≡joint1
≡jogVelMax
≡acc
≡dec
5
joint1Jog
MC_Jog
Axis
JogForward
JogBackward
Velocity
Acceleration
Deceleration
Jerk
Busy
CommandAborted
Error
ErrorId
hmiBtnJoint2JogFw
NOT
hmiBtnJoint2JogBckw
hmiBtnJoint2JogBckw
NOT
hmiBtnJoint2JogFw
AND
AND
≡joint2
≡jogVelMax
≡acc
≡dec
5
joint2Jog
MC_Jog
Axis
JogForward
JogBackward
Velocity
Acceleration
Deceleration
Jerk
Busy
CommandAborted
Error
ErrorId

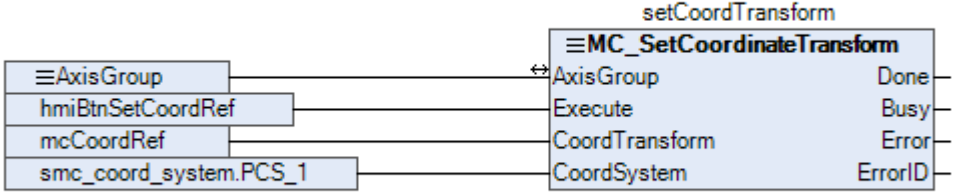
CODESYS

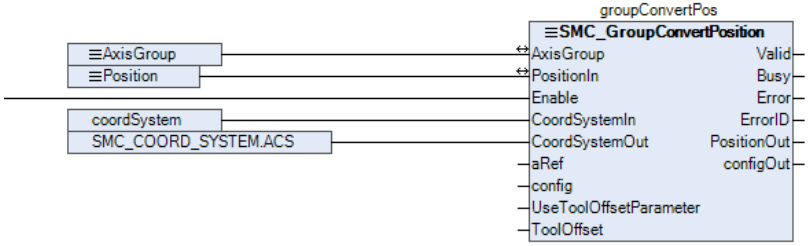

groupActualPos
MC_GroupReadActualPosition
≡AxisGroup
powerOn
coordSystem
AxisGroup
Enable
CoordSystem
Valid
Busy
Error
ErrorID
Position
KinematicConfig


groupMoveDirectAbs
MC_MoveDirectAbsolute
≡AxisGroup
moveAbsDirectOn
coordSystem
MC_BUFFER_MODE.Aborting
≡velFactor/100
AxisGroup
Execute
Position
MovementType
CoordSystem
BufferMode
TransitionMode
TransitionParameter
VelFactor
AccFactor
JerkFactor
Done
Busy
Active
CommandAborted
CommandAccepted
Error
ErrorID
MovementId
groupMoveLinearAbs
≡MC_MoveLinearAbsolute
≡AxisGroup
moveAbsLinearOn
≡fSWMaxVelocity
≡acc
≡dec
5
coordSystem
MC_BUFFER_MODE.Aborting
≡SMC_ORIENTATION_MODE.Axis
≡velFactor/100
AxisGroup
Execute
Position
Velocity
Acceleration
Deceleration
Jerk
CoordSystem
BufferMode
TransitionMode
TransitionParameter
OrientationMode
VelFactor
AccFactor
JerkFactor
TorqueFactor
Done
Busy
Active
CommandAborted
CommandAccepted
Error
ErrorID
MovementId

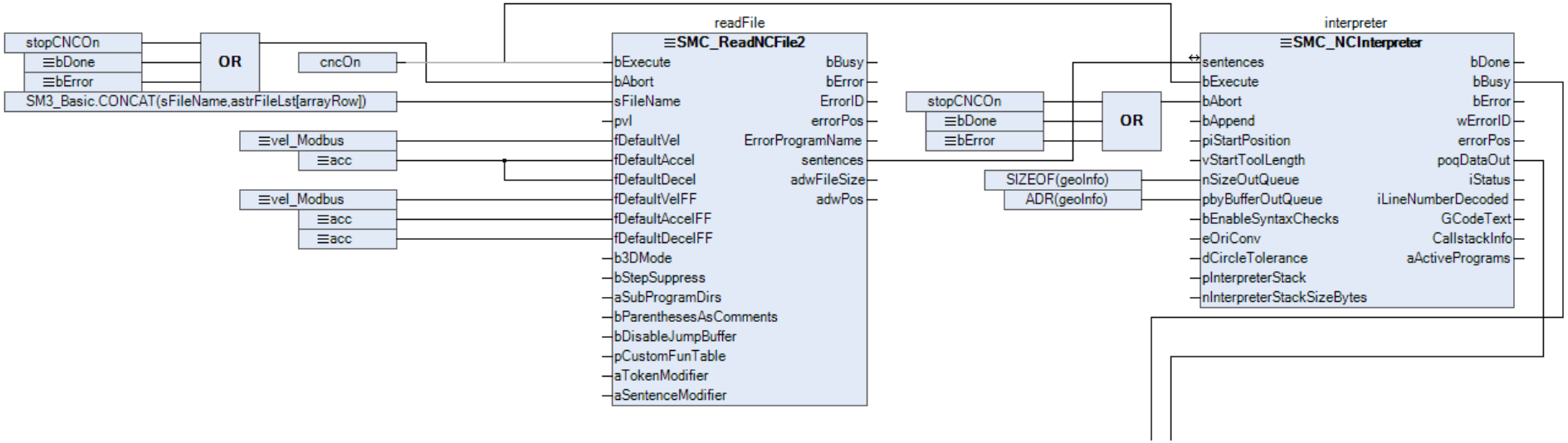
CODESYS


groupConvertPos
SMC_GroupConvertPosition
AxisGroup
Position
coordSystem
SMC_COORD_SYSTEM.ACS
AxisGroup
PositionIn
Enable
CoordSystemIn
CoordSystemOut
aRef
config
UseToolOffsetParameter
ToolOffset
Valid
Busy
Error
ErrorID
PositionOut
configOut

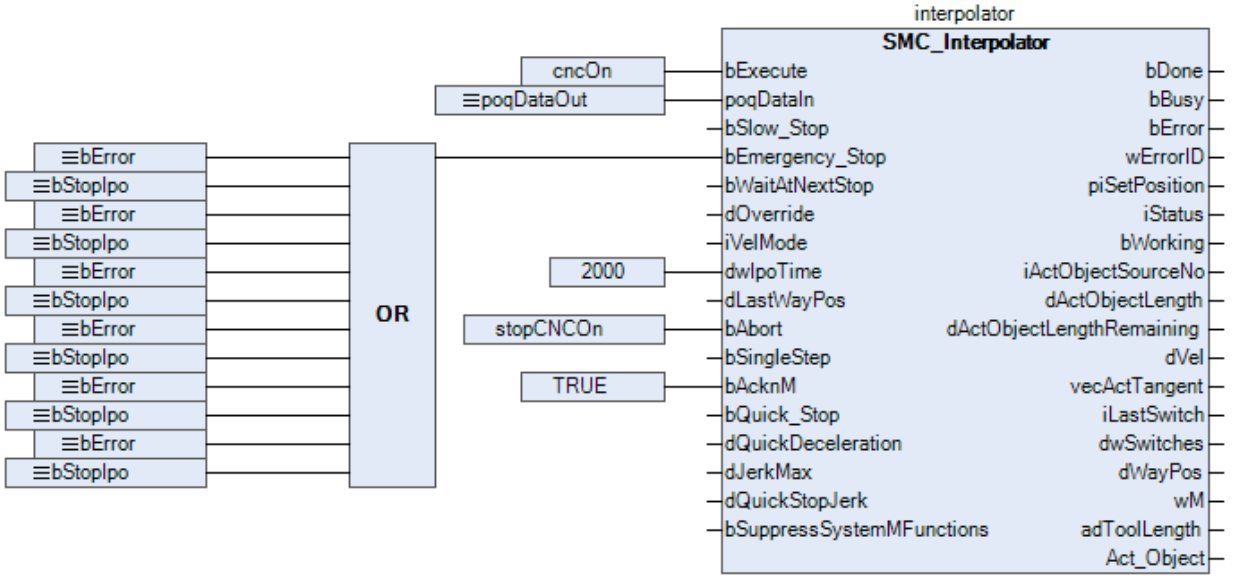

setCoordTransform
MC_SetCoordinateTransform
AxisGroup
hmiBtnSetCoordRef
mcCoordRef
smc_coord_system.PCS_1
AxisGroup
Execute
CoordTransform
CoordSystem
Done
Busy
Error
ErrorID


groupConvertPos
SMC_GroupConvertPosition
AxisGroup
Position
coordSystem
SMC_COORD_SYSTEM.ACS
AxisGroup
PositionIn
Enable
CoordSystemIn
CoordSystemOut
aRef
config
UseToolOffsetParameter
ToolOffset
Valid
Busy
Error
ErrorID
PositionOut
configOut

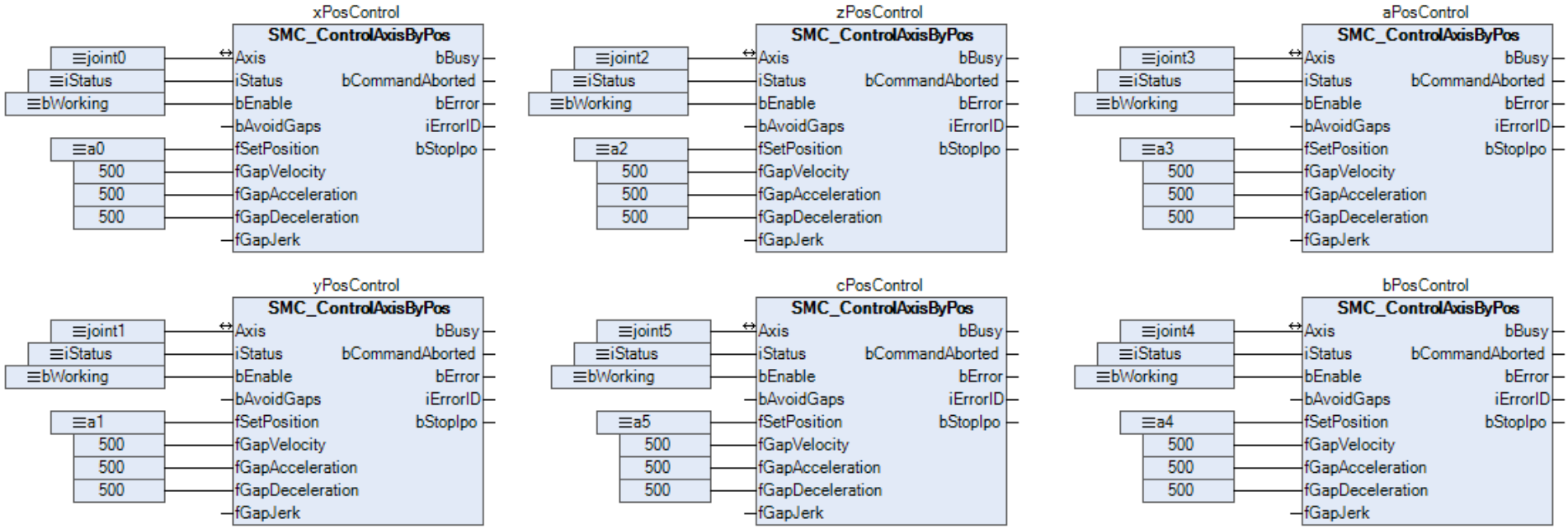
CODESYS


readFile
SMC_ReadNCFile2
bExecute
bAbort
sFileName
pvl
fDefaultVel
fDefaultAccel
fDefaultDecel
fDefaultVelFF
fDefaultAccelFF
fDefaultDecelFF
b3DMode
bStepSuppress
aSubProgramDirs
bParenthesesAsComments
bDisableJumpBuffer
pCustomFunTable
aTokenModifier
aSentenceModifier
bBusy
bError
ErrorID
errorPos
ErrorProgramName
sentences
adwFileSize
adwPos
stopCNCOn
bDone
bError
OR
cncOn
SM3_Basic.CONCAT(sFileName,astrFileLst[arrayRow])
vel_Modbus
acc
vel_Modbus
acc
acc
interpreter
SMC_NCInterpreter
sentences
bExecute
bAbort
bAppend
piStartPosition
vStartToolLength
nSizeOutQueue
pbyBufferOutQueue
bEnableSyntaxChecks
eOriConv
dCircleTolerance
pInterpreterStack
nInterpreterStackSizeBytes
bDone
bBusy
bError
wErrorID
errorPos
poqDataOut
iStatus
iLineNumberDecoded
GCodeText
CallstackInfo
aActivePrograms
stopCNCOn
bDone
bError
OR
SIZEOF(geoInfo)
ADR(geoInfo)

CODESYS

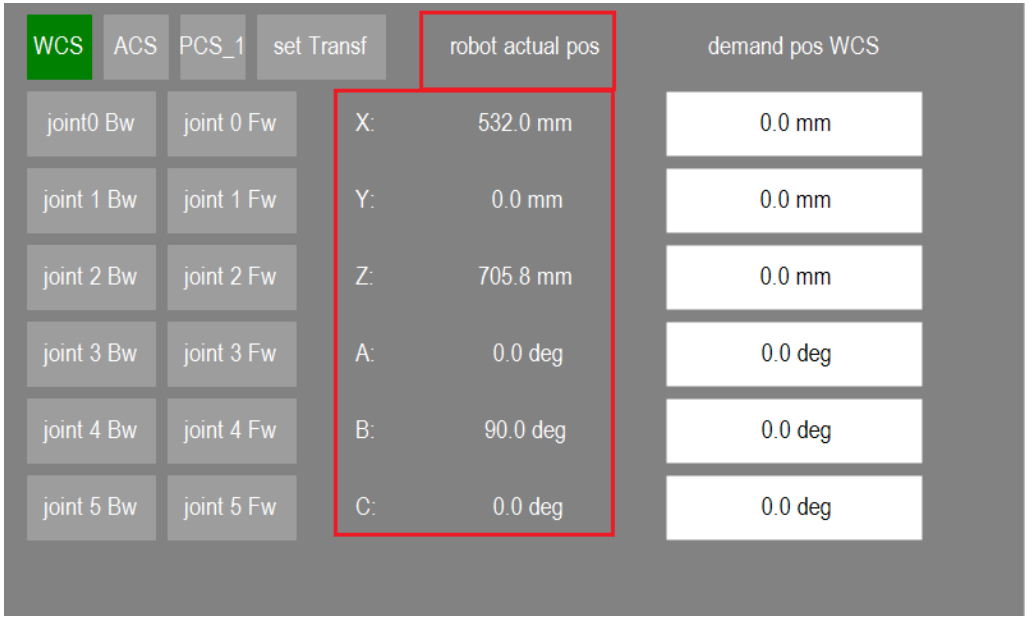

interpolator
SMC_Interpolator
cncOn
≡poqDataOut
≡bError
≡bStopIpo
≡bError
≡bStopIpo
≡bError
≡bStopIpo
≡bError
≡bStopIpo
≡bError
≡bStopIpo
≡bError
≡bStopIpo
OR
2000
stopCNCOn
TRUE
bExecute
poqDataIn
bSlow_Stop
bEmergency_Stop
bWaitAtNextStop
dOverride
iVelMode
dwIpoTime
dLastWayPos
bAbort
bSingleStep
bAcknM
bQuick_Stop
dQuickDeceleration
dJerkMax
dQuickStopJerk
bSuppressSystemMFunctions
bDone
bBusy
bError
wErrorID
piSetPosition
iStatus
bWorking
iActObjectSourceNo
dActObjectLength
dActObjectLengthRemaining
dVel
vecActTangent
iLastSwitch
dwSwitches
dWayPos
wM
adToolLength
Act_Object

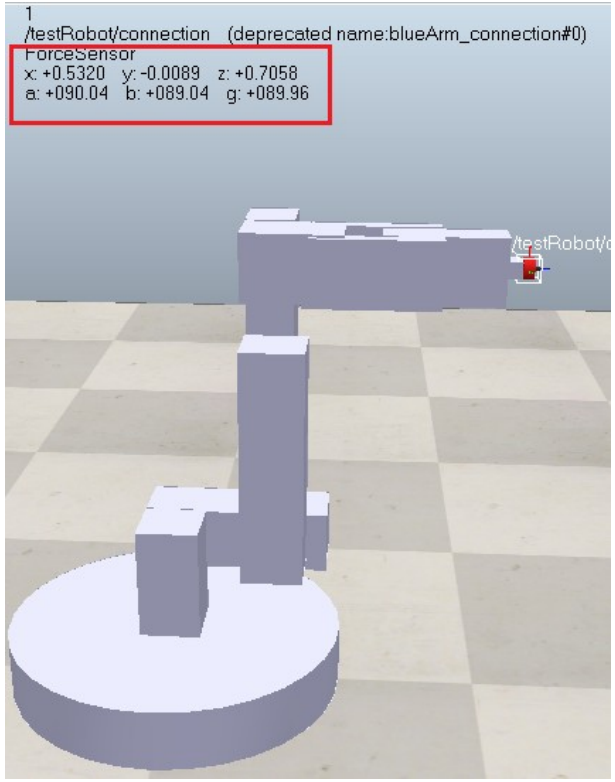
CODESYS


xPosControl
SMC_ControlAxisByPos
joint0
iStatus
bWorking
a0
500
500
500
Axis
iStatus
bEnable
bAvoidGaps
fSetPosition
fGapVelocity
fGapAcceleration
fGapDeceleration
fGapJerk
bBusy
bCommandAborted
bError
iErrorID
bStopIpo
zPosControl
SMC_ControlAxisByPos
joint2
iStatus
bWorking
a2
500
500
500
Axis
iStatus
bEnable
bAvoidGaps
fSetPosition
fGapVelocity
fGapAcceleration
fGapDeceleration
fGapJerk
bBusy
bCommandAborted
bError
iErrorID
bStopIpo
aPosControl
SMC_ControlAxisByPos
joint3
iStatus
bWorking
a3
500
500
500
Axis
iStatus
bEnable
bAvoidGaps
fSetPosition
fGapVelocity
fGapAcceleration
fGapDeceleration
fGapJerk
bBusy
bCommandAborted
bError
iErrorID
bStopIpo
yPosControl
SMC_ControlAxisByPos
joint1
iStatus
bWorking
a1
500
500
500
Axis
iStatus
bEnable
bAvoidGaps
fSetPosition
fGapVelocity
fGapAcceleration
fGapDeceleration
fGapJerk
bBusy
bCommandAborted
bError
iErrorID
bStopIpo
cPosControl
SMC_ControlAxisByPos
joint5
iStatus
bWorking
a5
500
500
500
Axis
iStatus
bEnable
bAvoidGaps
fSetPosition
fGapVelocity
fGapAcceleration
fGapDeceleration
fGapJerk
bBusy
bCommandAborted
bError
iErrorID
bStopIpo
bPosControl
SMC_ControlAxisByPos
joint4
iStatus
bWorking
a4
500
500
500
Axis
iStatus
bEnable
bAvoidGaps
fSetPosition
fGapVelocity
fGapAcceleration
fGapDeceleration
fGapJerk
bBusy
bCommandAborted
bError
iErrorID
bStopIpo

Συμπεράσματα

Θέση parking ρομποτικού βραχίονα σε CODESYS και σε CoppeliaSim

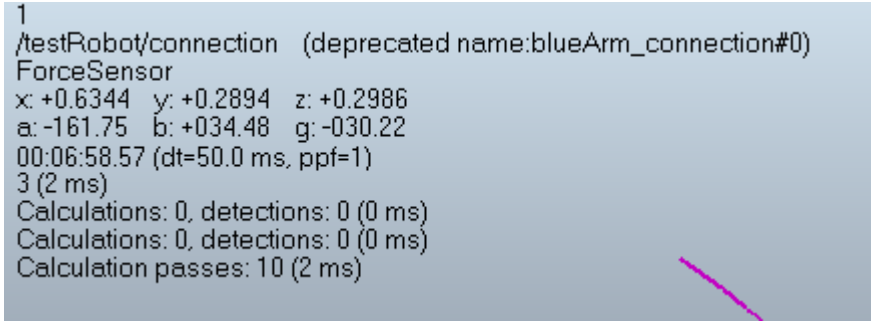




Συμπεράσματα

Τυχαία θέση ρομποτικού βραχίονα σε CODESYS (mm) και CoppeliaSim (m)

ransf	robot actual pos
X:	634.4 mm
Y:	289.4 mm
Z:	298.6 mm
A:	24.5 deg
B:	141.5 deg
C:	0.0 deg



Συμπεράσματα

Απεικόνιση G – κώδικα σε COSESYS και σε CoppeliaSim

CNCExample.cnc - Notepad

File Edit Format View Help

```
N10 G00 X50 Y50
N20 G01 X100 Y50
N30 G02 X150 Y100 R50
N40 G01 X200 Y100
N50 G01 X200 Y0
N60 M30
```