ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

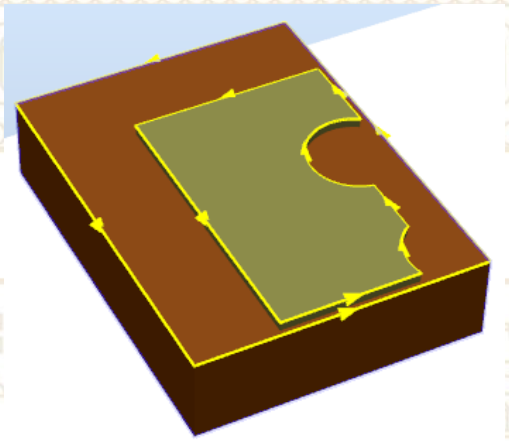
ABB RobotStudio Gyakorlat

II. Mérési Feladat

Egyszerű Robotprogram készítése

Mérés Célja:

A mérés során a hallgatónak egy egyszerű robotprogramot kell elkészítenie. A manipulátor feladata, hogy az 1. ábrán látható munkadarab sárgával kijelölt kontúrjait lekövesse. A feladat megoldása során a hallgató megismerkedik az offline robotprogramozás alapjaival, a robotprogramozás során alkalmazott mozgás típusokkal és a RAPID program nyelv alapvető szintaktikai elemivel.



1. ábra Bejárni kívánt pálya

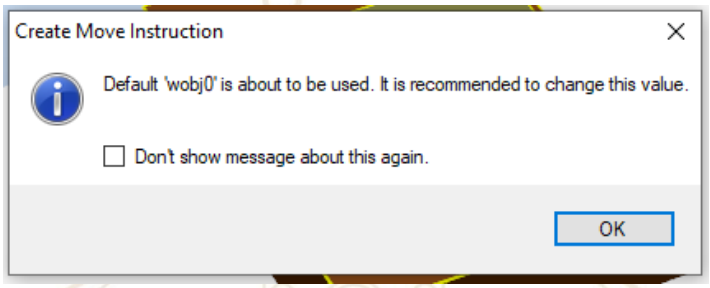
A mérés során a hallgatónak az ABB RobotStudio Gyakorlat – I. Mérési Feladatban megalkotott robotcellát kell felhasználnia. Nyissa meg a cellát a RobotStudio segítségével!

A mérésről a hallgatónak jegyzőkönyvet kell késztenie, amelyben az egyes részfeladatok végén található kérdéseket kell megválaszolni.

1. feladat – Workobject konfigurálása és pályapontok felvétele

A robot programjának megírásához először a bejárni kívánt pálya fontosabb pontjait kell meghatároznunk, majd ezt követően felvehetjük az egyes pontokat összekötő pálya geometriáját és konfigurálhatjuk a robotunk mozgását.

Azonban mielőtt belekezdenénk a pálya pontok felvételébe, először meg kell határoznunk, mely **WorkObject**hez szeretnénk hozzárendelni a pontokat. Amennyiben nem hozunk létre új **Workobject**et a RobotStudio az alapértelmezett **Workobejct** koordináta rendszerhez (wobj0) fogja hozzárendelni a **Target**eket és pályapontok felévtelekor a *2. ábrán* látható hibaüzenettel figyelmeztet bennünket.



2. ábra Alapértelmezett Workobeject használata esetén felugró hibaüzenet

Mivel a mi feladatunk, hogy lekövessük a munkadarab kontúrját a robotunk segítségével, így esetünkben érdemes a munkadarabhoz egy **Workobject**et rendelni. Ezáltal a munkadarab újra pozícionálása esetén, a pályapontok pozíciójának módosítása is automatikus lesz.

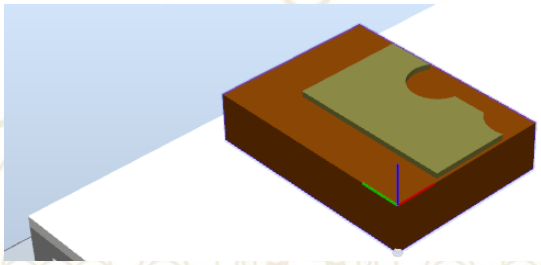
Az új **Workobject** felvételéhez kattintson a **Home** fülre felül, majd nyissa le az **Other** menüpontot a szalagmenüben és válassza ki a **Create Workobject** utasítást (3. ábra-1).

3. ábra Workobject létrehozása

A bal felső sarokban megjelenő ablakban adhatjuk meg a koordináta rendszer paramétereit (3. ábra-2.). Jelenleg csak két adatot kell megadnunk, a többi paraméter működésének megértéséhez a RobotStudio mélyebb szintű ismerete szükséges. Először adjunk egy jól felismerhető nevet a **Workobject**nek, ügyeljünk rá hogy akkor is felismerhető legyen ha a szimuláció során több koordináta rendszert is használunk. Ezt az adatot a **Misc data>Name** menüpontnál adhatjuk meg. Ezt követően adjuk meg a **Workobject** koordináta rendszer pozícióját a térben. Ehhez nyissuk le az **Object Frame>Frame By Point** menüpontot. Itt a koordináta rendszer három pontjának megadásával tudjuk meghatározni annak pozícióját és az orientációját a térben. Az első pont (**Position (mm)**) a Workobject koordináta rendszer origójának a pozícióját adja meg, a második pont (Point on X axis (mm)) az x tengely egy pontja,

a harmadik pont (**Point on XY plane (mm)**) pedig az xy sík, esetünkben az y tengely egy pontja lesz.

A **Selection** és **Snap Tool** helyes beállításai mellett vegye fel a három pont koordinátáit, úgy hogy a **Workobject** a 4. ábrának megfelelően helyezkedjen el a szimulációs térben.



4. ábra Workobject elhelyezkedése

Figyelem!

A Workobject koordináta rendszer felvétele során érdemes olyan pontokat kiválasztani, amelyek a valós munkacellában is pontosan megközelíthetők a robot segítségével, mivel így a pályapontok, illetve a robot programjának valós cellához való igazítása jelentősen felgyorsítható. A mi esetünkben a munkadarab sarokpontjai kiváló megoldást jelentenek, mivel a használt szerszám csúcsos kialakítása miatt a sarokpontok a valóságban is jó referencia pontként szolgálnak.

A létrehozott **Workobjecet**et a baloldali menüben is megtalálhatjuk. Ehhez válasszuk ki felül a **Home** fület, majd a baloldali menüben kattintsunk a **Path&Targets** fülre. A **Worobject**eket az **IRB1600_6kg_1.45m>T_ROB1>Worobjects & Targets** menüpont alatt találhatjuk.

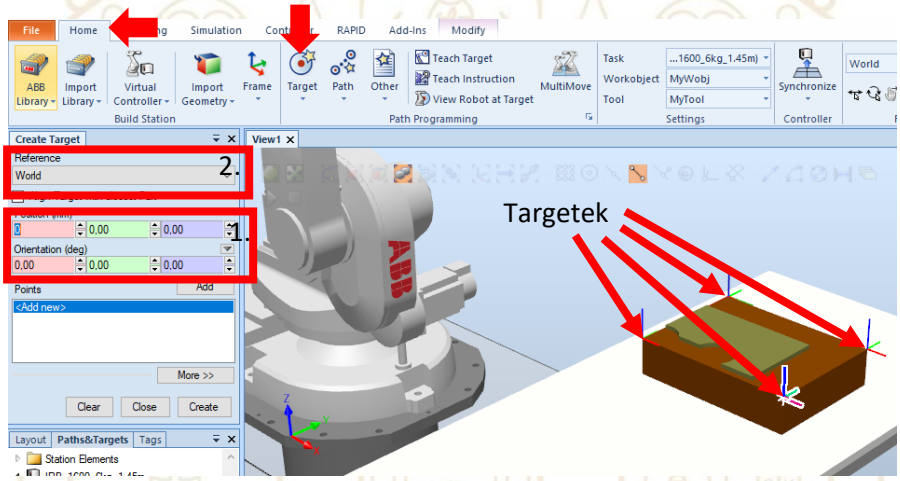
Továbbá láthatjuk, hogy a **Home** fül alatt, a **Settings** résznél a RobotStudio automatikusan az általunk utoljára létrehozott **Workobject**et állítja be, mint aktív koordináta rendszer.

Következő lépésként kényszerezzük hozzá az általunk létrehozott **Workobject** koordináta rendszert a munkadarabunkhoz, így a munkadarab mozgatása esetén automatikusan módosul a koordináta rendszer pozíciója is.

Ehhez kattintsunk jobb egérgombbal a **Workobject** nevére a baloldali menüben, majd a legördülő listából válasszuk az **Attach to** menüpontot. A megjelenő menüből válasszuk ki a munkadarab nevét, amikor a RobotStudio rákérdez, hogy szeretnénk-e frissíteni a koordináta rendszer pozícióját, kattintson a Nem gombra, mivel a **Workobeject** pozícióját már beállítottuk és a továbbiakban szeretnénk megőrizni a koordinátarendszer és a munkadarab alapkoordináta rendszerének relatív pozícióját. Így hozzá kényszereztük a koordináta rendszerünket a munkadarabhoz, tehát a **Workobject** innentől kezdve együtt mozog a munkadarabbal. Mozgassa a munkadarabot a robot munkaterébe a **Move** utasítás segítségével!

A **Workobject** felvételét követően elkezdhetjük a pályapontok meghatározását. Osszuk két részre a feladatot! Először készítsük el a munkadarab külső kontúrjának bejárásához szükséges programot. A pályapontok felvételéhez válasszuk a **Home** fület felül, majd a szalagmenüben kattintsunk a **Target** utasításra! A baloldalon felül megjelenő ablakban adhatjuk meg a **Target**ek pozícióját, illetve orientációját (5. ábra-1.). Továbbá ez esetben is változtathatjuk a referencia koordináta rendszert (5. ábra- 2.) a pontok egyszerűbb felvétele érdekében.

A **Selection** és **Snap Tool** megfelelő beállításai mellett rendeljen egy-egy **Target**et a munkadarab sarokpontjaihoz (5.ábra)! A **Target**eket a **Create** gombbal tudja létrehozni.



5. ábra Pálya pontok/Targetek létrehozása

A **Target**ek a baloldali menüben is megtalálhatók. Kattintson a **Home** fülre felül, majd válassza ki a baloldali menüben a **Path&Target**s fület. A **Target**eket az **IRB1600_6kg_1.45m>T_ROB1>Workbjects & Targets>Workobject_Neve** menüpont alatt találhatjuk.

Figyelem!

A pálya pontok felvétele előtt mindig ellenőrizze, hogy az aktív Workobject helyesen van-e beállítva.

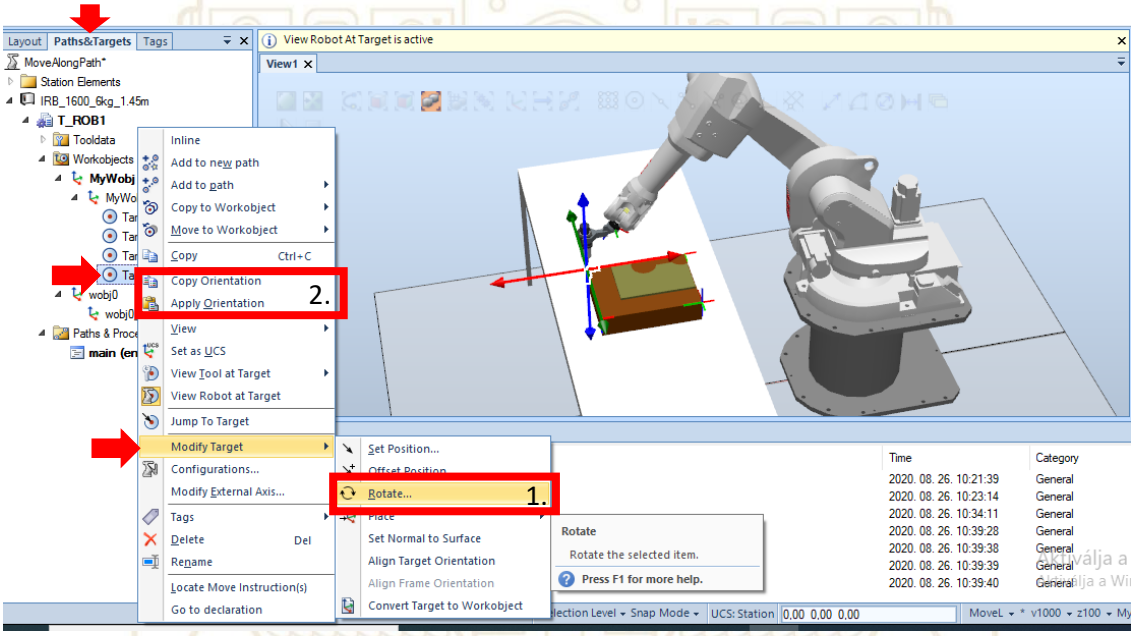
Kérdések:

1. Miért nem elegendő a pályapontokat egyetlen helyvektorral jellemezni? Miért alkalmazunk koordináta rendszereket (Target) a pálya leírásához?
2. A jelenlegi konfiguráció szerint milyen hibákat észlel a felvett pálya pontok esetében? (A kérdés megválaszolásához a baloldali menüben kattintson jobb egérgombbal a Targetek nevére és használja a View Tool at Target és View Robot at Target utasításokat, illetve vizsgálja az Output Windowt)

2. feladat – Mozgáspálya létrehozása és robotmozgások konfigurálása

A robotunk a mozgása során a megírt program szerint az általunk felvett **Targetek**nek megfelelően pozícionálja az **End-Effector**t, úgy hogy az aktív szerszám koordináta rendszer és a mozgásutasítás által meghatározott **Target** pozíciója és orientációja megegyezzen.

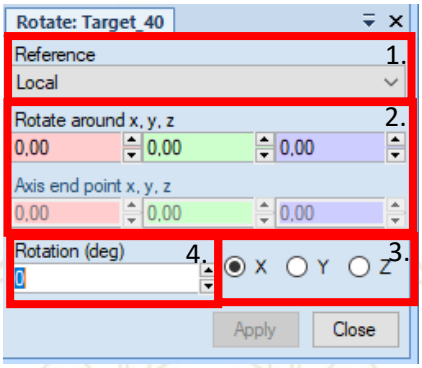
Módosítsa úgy a felvett **Target**ek orientációját, hogy a robot megfelelően álljon pozícióba a munkadarabnál. Ehhez kattintson a jobb egérgombbal a baloldali menüben a **Target** nevére, majd nyissa le a **Modify Target...** menüpontot és válassza ki a **Rotate** utasítást (6. ábra-1). Amennyiben egy **Target** orientációját helyesen beállította, kattintson jobb egérgombbal a pálya pont nevére a baloldali menüben, majd válassza ki a **Copy Orient** utasítást (6. ábra-2). Ezt követően a többi pontot kijelölve az **Apply Orient** utasítással (6. ábra-2) átmásolhatja a helyesen beállított orientációt a többi pálya pontra is, így nincs szükség az orientáció egyesével történő beállítására.



6. ábra Pálya pontok/Targetek orientációinak beállítása

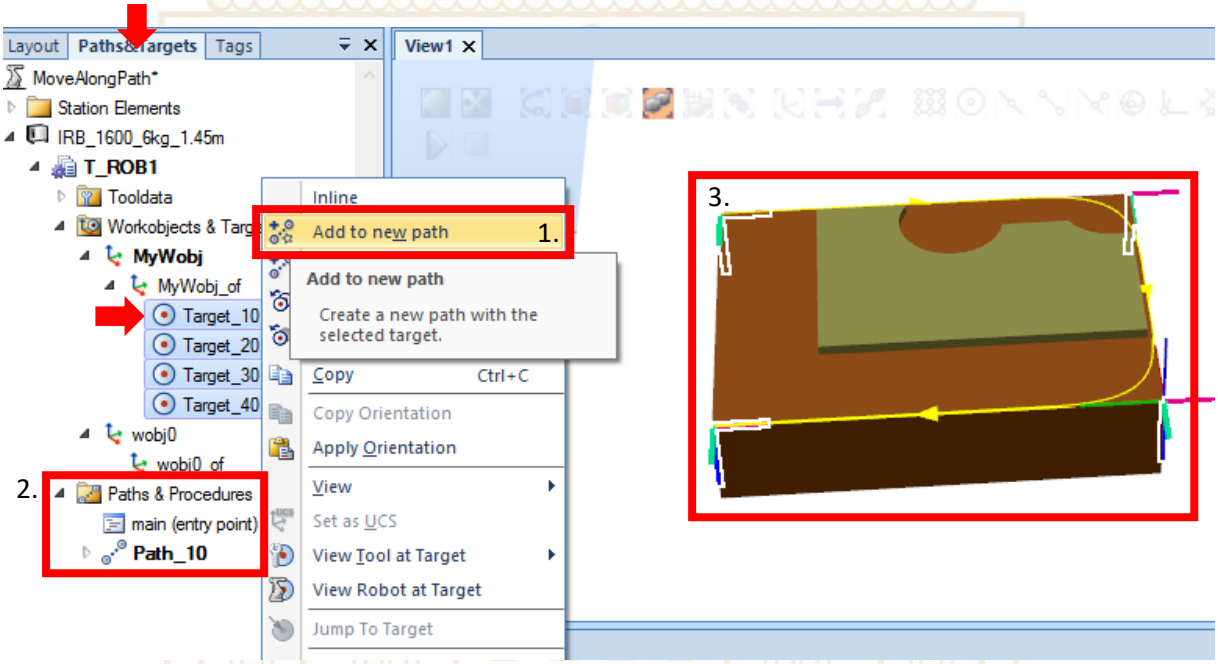
A Rotate utasítás alkalmazása során négy paramétert adhatunk meg (7. ábra):

1. Referencia koordináta rendszer
2. Forgatás tengelye, egyénileg megadva (csak abban az esetben kell megadni, ha nem nevezetes tengely körül forgatunk)
3. Forgatás x, y vagy z tengely körül
4. Forgatás szöge



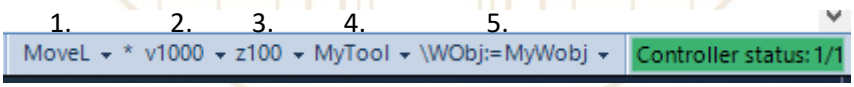
7. ábra Rotate utasítás paraméterezése

A pálya pontok helyes orientációjának beállítását követően megkezdhetjük a robot pályájának felvételét. Ehhez jelöljük ki az összes **Target**et a baloldali menüben, majd kattintsunk rá jobb egérgombbal és a legördülő menüből válasszuk az **Add to New Path** utasítást (8. ábra-1). A létrehozott pálya a baloldali menüben a **IRB1600_6kg_1.45m>T_ROB1>Paths & Procedures** menüpont alatt látható (8. ábra-2), a szimulációs térben pedig sárgával jelölve láthatjuk (8. ábra-3.).



8. ábra Pálya generálása

A pálya geometriájára tekintve látatjuk, hogy egyeltalán nem felel meg az elvárásainknak. Ez a mozgások rossz paraméterezéséből adódik. A jelenleg használt mozgás paraméterek a képernyő jobb alsó sarkában láthatók, amennyiben felül a **Home** fülre kattintunk (9. ábra).



9. ábra Mozgás paraméterek

A mozgás utasítások az alábbi módon paraméterezhetők (9. ábra):

1. Mozgástípus kiválasztása
   A menüpontot lenyitva négy utasítást ajánl fel a RobotStudio (**MoveJ, MoveL, MoveAbsJ, MoveExtJ**). Jelenleg nekünk a **MoveL (Move Linear)** utasítás van kiválasztva. Ezen utasítás esetében a robot az aktuális pozíciójából a kiválasztott pálya pontig egy egyenes vonal mentén elmozdulva jut el. Tehát az elmozdulás egy előre jól definiált pálya mentén történik (**CP/ Continous Path movement**), interpolált mozgásról van szó. A robot mozgásának tervezése a világ koordináták szintjén történik.
   Az interpolált/ **CP** mozgások egy másik gyakori fajtája a körpálya mentén történő elmozdulás. A jelenlegi konfigurációban ezt a mozgást nem ajánlja fel közvetlen a RobotStudio, mivel a körpálya meghatározásához három pontra van szükségünk (a robot aktuális pozíciója, a körív egy tetszőleges pontja, illetve a körív végpontja). Körpályán a **MoveC (Move Circular)** utasítással vezérelhetjük a robotot.

Figyelem!

Mivel CP mozgások esetében a trajektória tervezés a világkoordináták szintjén történik, így szinguláris robotpozíciókat kerülni kell!

   A **MoveJ (Move Joint)** utasítás csuklómozgást jelöl. Ezesetben a robot mozgásának tervezése a csuklókoordináták szintjén történik, ami azt eredményezi, hogy pálya tervezés pillanatában nem tudjuk pontosan meghatározni a szerszám koordináta rendszer pályáját, csak a mozgás kezdő- és végpontját ismerjük pontosan. Az ilyen típusú mozgásokat **PTP (Point-to-Point**) mozgásoknak nevezzük. A csuklómozgás során minden robot csukló egyidejűleg kezdi meg, illetve fejezi be a mozgást. A **MoveJ** utasítás finom mozgást eredményez, így kedvező a robot hajtásrendszere számára. A **PTP** típusú mozgások alkalmazásának előnye, hogy a trajektória tervezés a csuklókoordináták szintjén történik ezáltal nincs szükség a Jacobi mátrixunk inverzének számítására, így a robotunk könnyedén áthalad az esetleges szinguláris pozíciókon, amelyek **CP** mozgások esetén megakasztanák a programunk futását.
   A **MoveAbsJ** szintén egy **PTP** mozgás, a csuklósebességek számítása hasonlón történik, mint a **MoveJ** utasítás esetében, az egyetlen különbség, hogy ez az utasítás kizárólag csuklókoordinátákkal megadott pályapontokon értelmezhető.
   MoveExtJ a pedig külső tengelyeink mozgásának vezérlésére alkalmas.
2. Sebesség kiválasztása
   Itt a RobotStudio által előre definiált sebesség beállításokból választhatjuk ki a program végrehajtás során megvalósított szerszám, azaz a szerszám koordináta rendszer mozgásának sebességét.

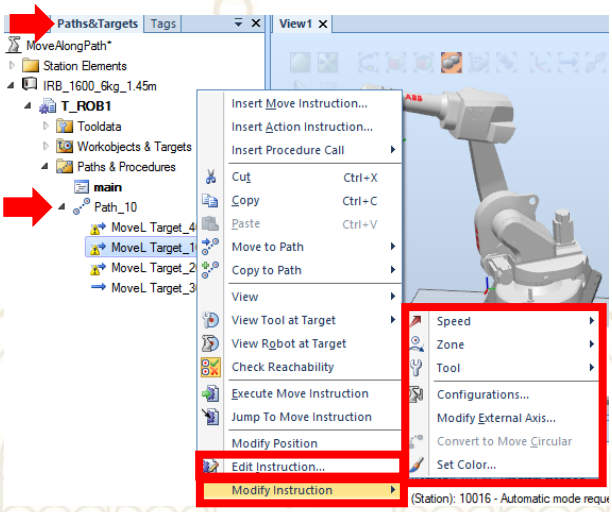
3. Közelítés **(zona data)**
   Itt azt adhatjuk meg, hogyan történjen a mozgás utasításunk végrehajtása. Például mennyire közelítsük meg az adott pályapontot vagy mikor kezdjük meg a következő program utasítás végrehajtását.
   A listát lenyitva rengeteg opció tárul elénk, azonban a robot mozgása és a program végrehajtása szempontjából két lényegesen különböző paraméter közül választhatunk.
   A **fine** paramétert kiválasztva, az aktuális pályapontot úgynevezett stop pozícióként konfiguráljuk. Ezesteben a robotnak pontosan pozícióba kell állnia mielőtt a program végrehajtás folytatódna.
   A másik lehetőség, hogy valamely **z**xxx adatot választjuk, ezesetben az aktuális pályapontot úgynevezett „fly-by" pontként konfiguráljuk. A robot mozgása során nem éri el az előre beprogramozott pozíciót. Ehelyett a mozgás iránya már a póz elérése előtt megváltozik. A robot mozgása és a program végrehajtás folyamatos marad, a mozgás megkezdését követően, a programmutató automatikusan a következő sorra ugrik.

Figyelem!

Fly-by pontok alkalmazásával a ciklusidő jól optimalizálható. Továbbá egyenletesebb mozgást eredményez mivel a robot nem áll meg minden pályapontnál, így nincs szükség folyamatos intenzív gyorsítási-lassítási ciklusokra. Ez nagyban kíméli a manipulátor hajtás rendszerét, így növelhető a berendezés élettartama is.

4. Szerszám koordinátarendszer kiválasztása
   Itt azt adhatjuk meg, hogy a mozgás végrehajtása során mely szerszám koordinátarendszer legyen aktív.
5. **Workobject** kiválasztása
   Itt azt adhatjuk meg, hogy a kiválasztott pályapontot mely **workobject**ben értelmezzük.

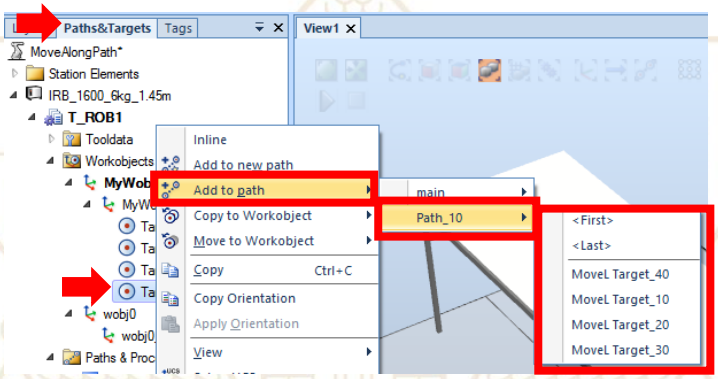
A korábban már létrehozott mozgásutasítások esetében a paraméterek módosításához a baloldali menüben nyissuk le a **IRB1600_6kg_1.45m>T_ROB1>Paths & Procedures>Path_10** menüpontot, majd kattintsunk jobb egérgombbal valamely mozgásutasítás nevére. Ezt követően a paraméterek az **Edit Instruction...** vagy **Modify Instruction** opciót kiválasztva módosíthatók.



10. ábra Mozgás paraméterek megváltoztatása

Módosítsa úgy a pályát, illetve a létrehozott mozgás utasításokat, hogy a robot kellő pontossággal, de folyamatos mozgással haladjon végig a téglalap kontúrján.

Amennyiben új pontot szeretne a pályához hozzáadni, a baloldali menüben kattintson valamely **Target** nevére jobb egérgombbal, válassza az **Add to path** utasítást.



11. ábra Target hozzáadása valamely pályához

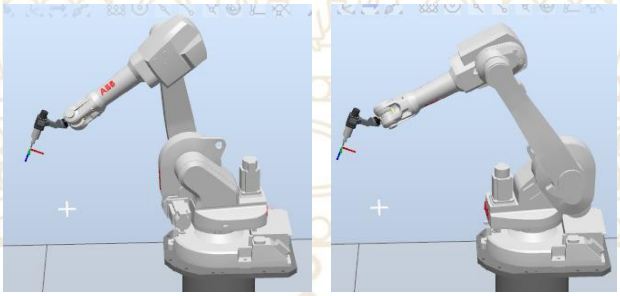
Kérdések:

3. Mutassa be milyen feltételek szerint állította be a Targetek orientációját!
4. Röviden írja le mit értünk szingularitás alatt robotikában! Mutasson be 3 különböző szinguláris pozíciót az IRB1600 manipulátor esetében!
5. Mi a különbség a v100, vrot100 és vlin100 sebesség paraméterek között? Használja segítségül a Robotics Technical Reference Manual - RAPID Instructions, Functions and Data types 3. fejezetét.
6. Mutassa be hogyan építette fel a robot mozgás pályáját (Targetek sorrendje) és milyen mozgás paramétereket adott meg a konfiguráció során.

3. feladat – Robotprogram generálása

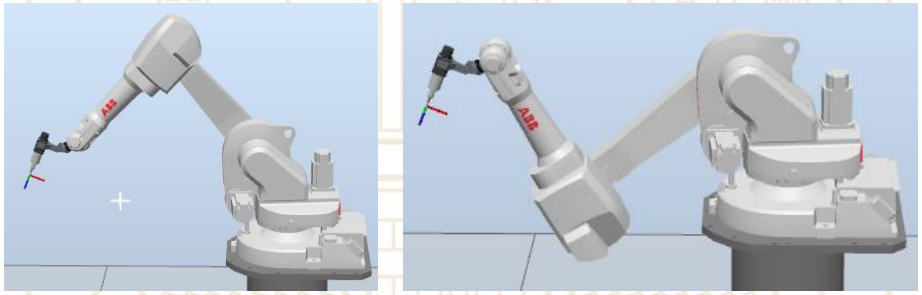
Amennyiben létrehoztuk a megfelelő pályát a következő lépés, hogy minden pályaponthoz kiválasszuk a megfelelő robotkonfigurációt. Erre azért van szükség mivel, korábbi tanulmányainkból már tudjuk, hogy az inverz geometriai feladatnak több megoldása is lehet. Ez gyakorlati szempontból azt eredményezi, hogy egy adott **Target** által meghatározott cartesian pózba többféle módon, azaz más-más csukló koordinátákkal is képes a robot beállni és nekünk ebből kell kiválasztani a számunkra legmegfelelőbbet. A leggyakrabban az alábbi nevezetes konfigurációkat különböztetjük meg:

- Jobbkezes és balkezes konfiguráció



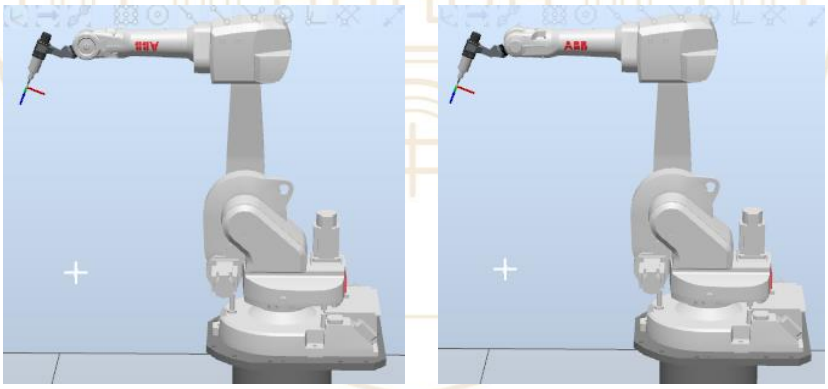
12. ábra Jobbkezes (bal oldal) és balkezes (jobb oldal) robotkonfiguráció

- Könyök fent és könyök lent konfiguráció



13. ábra Könyök fent (bal oldal) és könyök lent (jobb oldal) robotkonfiguráció

- Flip és non-flip konfiguráció



14. ábra Flip (bal oldal) és non-flip (jobb oldal) konfigurációk

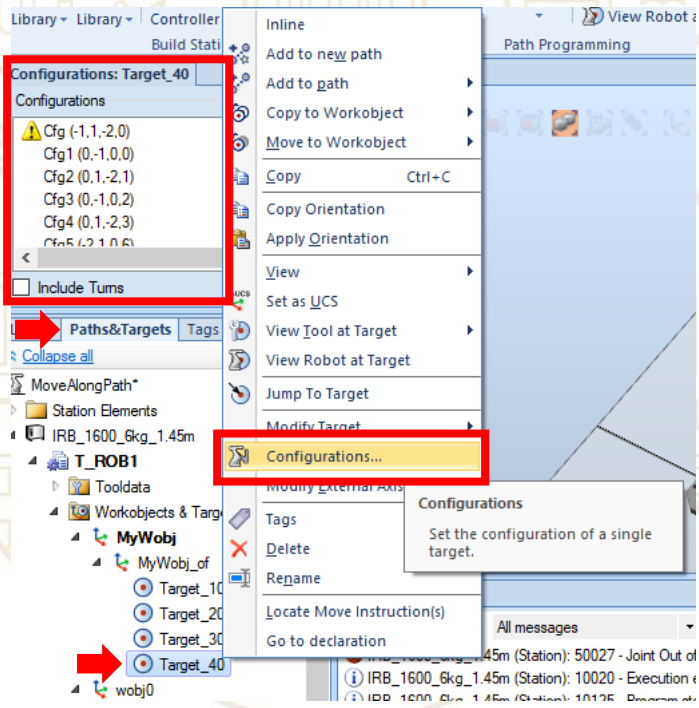
A jól látható a fenti ábrákon, hogy habár a cartesian póz az egyes konfigurációk esetében páronként megegyezik a csuklókoordináták mégis különböznek. Így a programozás során nekünk minden pozícióhoz ki kell választanunk a megfelelő konfigurációt.

Figyelem!

Konfiguráció váltás során a robot szinguláris pozíciókon halad át, tehát konfiguráció váltás csak csuklómozgással lehetséges és ekkor is csak nagy körültekintéssel hajtható végre, mert a robot nem várt mozgását eredményezheti! Interpolált / CP mozgások esetében (pl.: egyenes vagy körpálya) minden Target azonos konfigurációval kell, hogy rendelkezzen.

A **Robotstudio** esetében a legoptimálisabb robotkonfiguráció kiválasztását adott esetben a szoftver elvégzi számunkra. Ehhez kattintsunk jobb egérgombbal a baloldali menüben a **IRB1600_6kg_1.45m>T_ROB1>Paths & Procedures>Path_10** menüpontra. Ezután a legördülő menüben válasszuk az **Auto Configuration** opciót. Az automata konfiguráció kiválasztás során folyamatosan figyeljük az **output windowt**, amennyibe a **RobotStudio** nem talál megfelelő robotkonfigurációt a pályapontokhoz, módosítsuk a munkadarab (és így a pályapontok) pozícióját.

A pályapontokhoz rendelt robotkonfigurációt és a további konfigurációs lehetőségeket a baloldali menüben valamely **Target** nevére kattintva a jobb egérgombbal, majd a legördülő menüben az **Configurations...** opciót kiválasztva tekinthetjük meg (15. ábra).



15. ábra Robotkonfiguráció kiválasztása

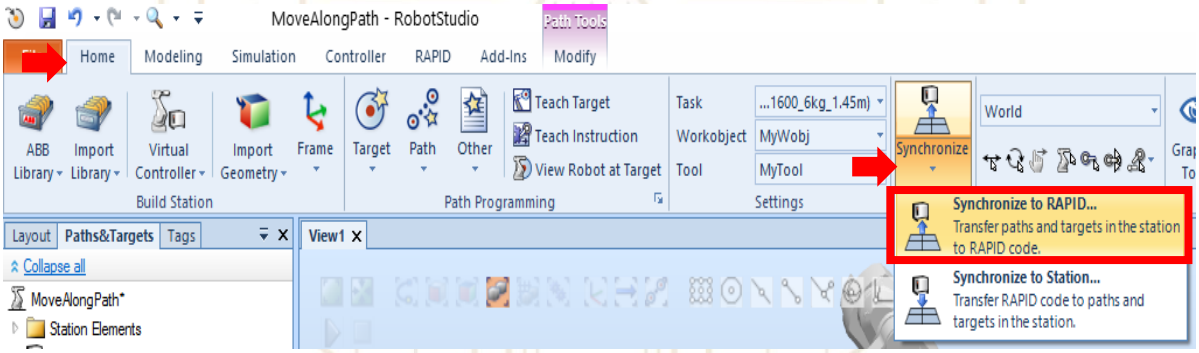
Az ABB robotok esetében a robotkonfigurációt négy számmal jelöljük:

- Az első számjegy azt határozza meg, hogy az 1. csukló koordináta mely térnegyedben helyezkedik el. 0 esetén a koordináta az 1. térnegyedben található, tehát a keresett csuklókoordináta 0° és 90° között van. A negatív számok óramutató járásával ellentétes irányú forgatást jelölnek.
- A második számjegy a 4. tengely elhelyezkedését határozza meg.
- A harmadik számjegy pedig a 6. csukló elhelyezkedését írja le.
- Az utolsó, azaz a negyedik számjegy pedig a kéztőpont (5. csukló) elhelyezkedését adja meg a többi csuklóhoz képest.

Amennyiben a robotot betanító panel segítségével programozzuk (online programozás), nincs szükség a konfiguráció külön megadására, hiszen ilyenkor a robotot minden esetben a betanítani kívánt pozícióba mozgatjuk, így a robotkonfiguráció automatikusan kiválasztásra kerül.

Ha elvégeztük az automatikus konfigurációt a baloldali menüben a pálya nevére kattintva a jobb egérgombbal és a **Move Along Path** utasítást kiválasztva ellenőrizhetjük a robotunk mozgását. Ha mindent rendben találunk megkezdhetjük a Rapid kód generálását az elkészített szimuláció alapján. A Rapid egy magasszintű programozási nyelv, amely segítségével az ABB robotvezérlőkre készíthetünk programot.

A RobotStudio lehetővé teszi számunkra, hogy egy általunk elkészített szimulációból generáljunk automatikusan robot programot. Így jelentősen felgyorsítható akár a programfejlesztés menete és rengeteg „írás gyakorlattól" menti meg a felhasználót, mivel nincs szükség az egyes mozgás utasítások vagy változó deklarációk kódsorainak begépelésére. Az automatikus kódgenerálásnak köszönhetően egy szintaktikailag is megfelelő kódot kapunk, amely szinte egyből futtatható. Ehhez válasszuk felül a **Home** fület, majd nyissuk le a **Synchronize...** menüpontot és kattintsunk a **Synchronize to Rapid...** utasításra.



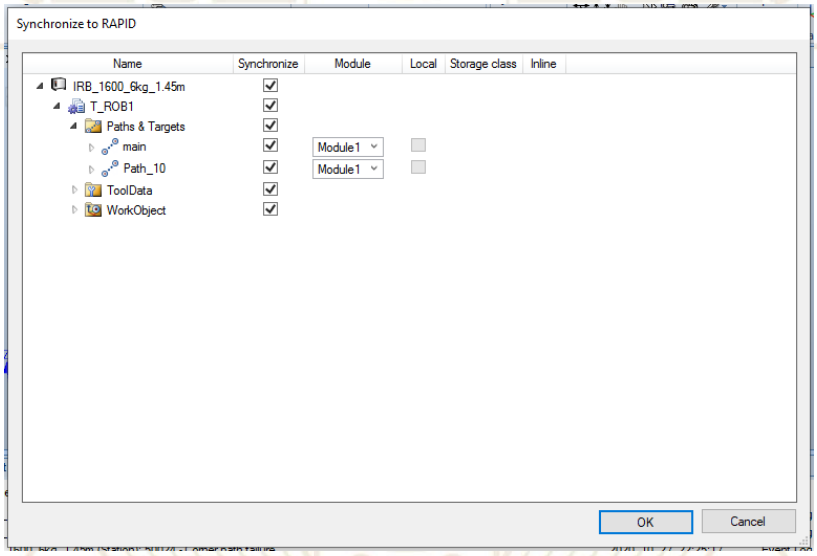
16. ábra Automatikus kódgenerálás

A kódgenerálás során ügyeljünk rá, hogy a megfelelő utasítást válasszuk ki a menüből. A **Synchronize to Station...** utasítás esetén a Rapid kód alapján módosítja a szimulációt a RobotStudio.

Figyelem!

A nem megfelelő utasítás kiválasztása bizonyos adatok törlését okozhatja. Mindig ügyeljen rá, hogy melyik irányba kíván szinkronizálni (Szimuláció ->Rapid vagy Rapid->Szimuláció)

A megfelelő utasítás kiválasztását követően a felugró ablakban pipáljuk ki a szinkronizálni kívánt elemeket. A RobotStudio lehetőséget biztosít részleges szinkronizációra, például amennyiben új pályapontokat veszünk fel és csak azokat szeretnénk a Rapid programba másolni, de a pálya módosítás nem szükséges. Jelen esetben pipáljuk ki az összes elemet, hiszen ez az első szinkronizációnk és minden adatot, paramétert át kell másolnunk a Rapidba.



17. ábra Szinkronizált adatok

A szinkronizáció sikerességét az **output window**ban ellenőrizhetjük. A generált kódot pedig a felül a Rapid fület kiválasztva, majd a baloldali menüben **Controller** fülre kattintva az **IRB1600_6kg_1.45m>RAPID>T_ROB1>Modul1** menüpontot kiválasztva tekinthetjük meg. Azonban a kód áttekintése előtt vizsgáljuk meg, milyen állományokat generált számunkra a RobotStudio és hogyan történik a kód rendszerezése.

A robot program mindig taszkokból áll (pl: T_ROB1 taszk), legegyszerűbb esetben mindössze egyből, viszont amennyiben párhuzamosan több feladatot szeretnénk végrehajtani további taszkok létrehozására van szükség. Például a későbbi gyakorlatok során áttekintjük, hogyan tudunk egy vezérlőről akár két teljesértékű robotkart vezérelni multitaszkos rendszer kialakításával.

A taszkok modulokból épülnek fel, ezekben történik a különböző adatok és rutinok deklarációja. Kétféle modult különböztethetünk meg. A **system module** a rendszer paramétereit írja le, előre definiált adatokat tartalmaz, amelyek megkönnyítik számunkra a programozást, mivel nincs szükség ezen adatok külön deklarációjára egyből használhatók számunkra (pl. wobj0 és tool0 koordináta rendszerek). A **system module-**ok nem szerves részei az általunk fejlesztett programnak, a vezérlő indítása során automatikusan kerülnek betöltésre. Ez azt jelenti, hogy a **system module** módosítása nem csak a jelenleg betöltött programunk működését befolyásolhatja, hanem az ezután behívott programokét is. A másik típus a program modul. Egy taszkon belül tetszőleges számú program modult hozhatunk létre, azonban ezek közül csak egy rendelkezik a **main** eljárással. A programunk futtatása során tulajdonképpen ez a **main** eljárás kerül meghívásra, ide ugrik be először a program mutató.

A programunkat a baloldali menüben a **Module1** menüpontra kattintva tudjuk megnyitni. A programunk első sorában a **Module1** deklarációja látható, ezt követően pedig a változóink deklarációja történik. Mivel a programunkban az adatok deklarálása globálisan történt, így bármely modul számára elérhetők, lokális deklarációhoz a LOCAL jelzőt kell a sor elejére írnunk, így már csak az adott modulon belül érhetjük el a változót. Egy eljáráson belül létrehozott változó, az eljárás saját lokális változója lesz.

```
1  MODULE Module1
2
3      LOCAL VAR intnum myvar;
4
5  ENDMODULE
```

18. ábra Lokális változó deklarálás

A Rapid háromféle adattípus különböztet meg:

- **VAR**: A program futása során értéke megváltoztatható
- **CONST**: Konstans, értéke nem változtatható a deklarációt követően
- **PERS**: A perzisztens változó értéke futásidőben is megváltoztatható, azonban újabb értékadás során a változó inicializálási értéke kerül felülírásra. Tehát a program a következő futás során is a változó legutoljára elmentett értékét fogja látni, a kezdeti értéktől függetlenül.

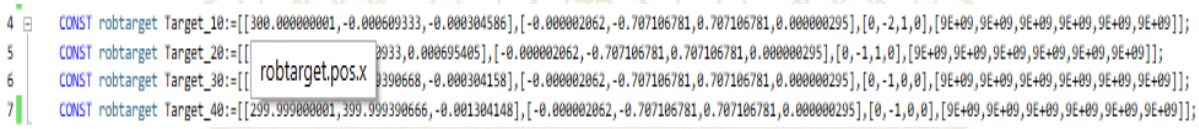
Jelen program esetében kizárólag konstans adataink vannak, ezek a szimulációs térben létrehozott **Target**ek. Nézzük meg, hogyan tudunk **Target**et deklarálni a **Rapid**ban!

Először az adat típusát kell megadnunk (**CONST**), ezt követően jöhet a változó típusa (**robtarget**), majd pedig a változó neve (**Target_10**). A **Rapid**ban az értékadás a := jelöléssel történik! **Robtarget** típusú változó esetén az adatokat szögletes zárójelbe kell írni és azon

belül is az adat egyes komponens csoportjait is külön szögletes zárójellel kell ellátni. A **robtarget** típusú változó a következő komponensekből épül fel:

- TCP pozíció – x, y, z koordináták
- TCP orientáció – q1, q2, q3, q4 kvaterniók
- Robotkonfiguráció - cf1, cf2, cf3, cf4 integerek
- Külső tengelyek csukló koordinátái – ext1, ext2, ext3, ext4, ext5, ext6

A programszerkesztőben a kurzorunkat bármely utasítás felé mozgatva és azt rövid ideig ott tartva a RobotStudio kiírja számunkra mely komponensről van szó.



19. ábra Robtarget típusú változó deklarációja

Figyelem!

A Rapid esetében minden programsort ;-vel kell lezárni!

A generált programunkban ezt követően az egyes rutinok deklarációja következik. A Rapid esetében három féle rutin deklarálható:

- **PROC:** Eljárás, nem rendelkezik visszatérési értekkel
- **FUNC**: Függvény, visszatérési értékkel rendelkezik
- **TRAP**: **Trap** rutinok segítségével kezelhetjük a megszakításainkat. Minden megszakításhoz egy-egy **trap** rutin társítható és a megszakítás esetén automatikusan ez a szubrutin kerül meghívásra. A megszakításokkal egy külön gyakorlat keretében foglalkozunk.

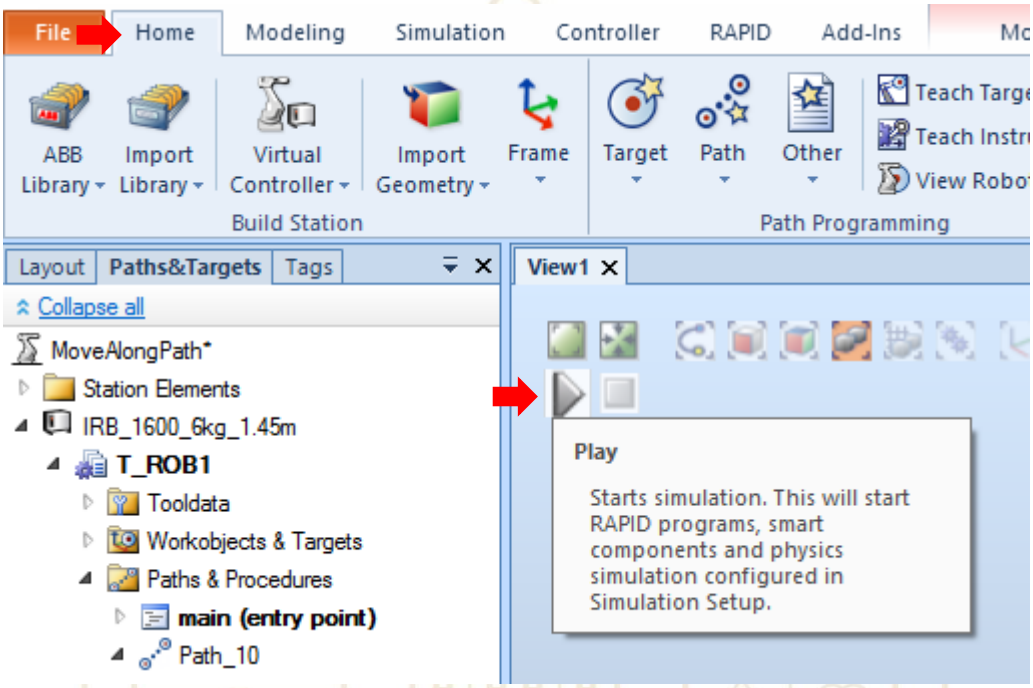
Jelen program esetében 2 eljárás került deklarálásra (**main** és **path_10**). A **Path_10** eljárásunk tartalmazza a szimulációs térben létrehozott mozgásutasításainak.

```
32    PROC Path_10()
33        MoveL Target_40,v300,z0,MyTool\WObj:=MyWobj;
34        MoveL Target_10,v300,z0,MyTool\WObj:=MyWobj;
35        MoveL Target_20,v300,z0,MyTool\WObj:=MyWobj;
36        MoveL Target_30,v300,z0,MyTool\WObj:=MyWobj;
37        MoveL Target_40,v300,z0,MyTool\WObj:=MyWobj;
38
39    ENDPROC
```

20. ábra Path_10 eljárás deklarációja

Futtassuk le a megírt kódot és figyeljük meg, hogyan alakul a szimuláció. Ehhez válasszuk felül a **Home** fület és a szimulációs ablakban kattintsunk a **Play** gombra.

Miért nem fut a szimuláció? Keresse meg a hibát!



21. ábra Szimuláció indítása

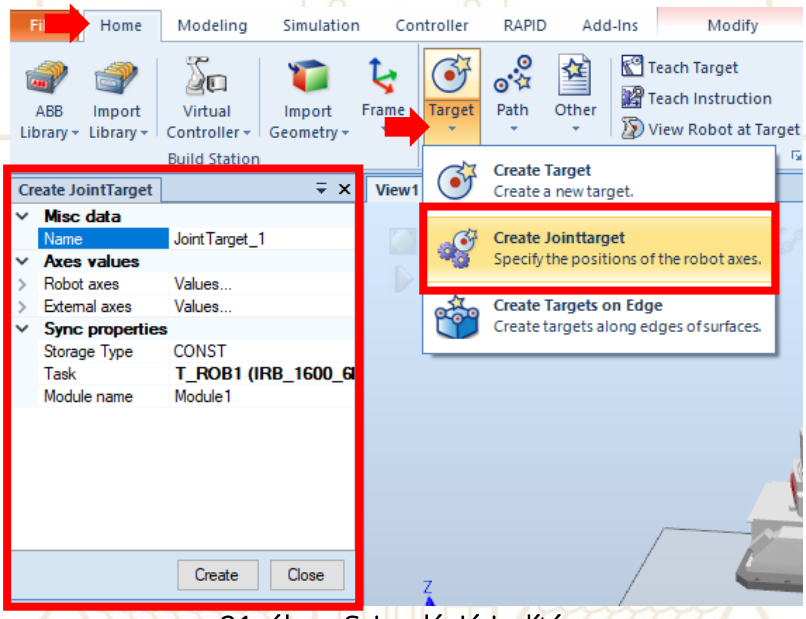
Kérdések:

7. Próbálja meg leírni mi a különbség a jobb/balkezes, a könyök fent/lent és a flip/non-flip konfigurációk között!
8. Mozgassa a robotot egy tetszőleges pozícióba balkezes, könyök fent, flip konfigurációban!
9. Miért volt hasznos a megfelelő wokrobject alkalmazása a példafeladat megoldása során?
10. Sorolja fel milyen modulokat hozott létre a RobotStudio a program készítése során. Adja meg az egyes modulok típusát is!
11. Próbálja meg értelmezni a CalibData modult! Írja le milyen adatokat, rutinokat lát ebben a modulban.
12. Egy tetszőleges mozgásutasításon mutassa be, hogy milyen paraméterei vannak az utasításnak.
13. Miért nem futott le először a program?

4. feladat – Robotprogram módosítása

Módosítsuk úgy a programunkat, hogy a munkaciklus elején és végén a robot mozogjon **home** pozícióba. Ehhez természetesen először egy új **Target**et kell felvennünk. Azonban a **home** pozíciót csukló koordinátákban kellene értelmezni a világ koordináták helyett. A csukló koordinátás megadás alkalmazásának előnye, a könnyebb értelmezés mellett, hogy legtöbb esetben a **home** pozíciónak, a q=0 szokták megadni, ez azonban szinguláris pozíció, tehát csak PTP mozgással fogjuk tudni megközelíteni könnyedén és a csukló koordinátákkal felvett pozíciók kizárólag PTP mozgással közelíthetők meg (ez egy plusz figyelem felkeltés a fejlesztő számára).

Csuklókoordinátás pozíció megadásához válasszuk felül a **Home** fület majd nyissuk le a **Target** menüpontot és kattintsunk a **Create Joint Target** menüpontra. A **Joint Target** paramétereit a bal felső sarokban megjelenő ablakban adhatjuk meg!



21. ábra Szimuláció indítása

Jelen esetben csak a Target nevét és a csukló koordinátákat kell megadnunk. Javaslom a q=[0, 0, 0, 0, 0, 30, 0] csuklókoordináta megadását.

Azonban a programunk módosítása előtt még egy pályapont felvétele szükséges. Ugyanis a mozgás során először csak megközelítjük a munkadarabot és majd csak ezt követően állunk az első pozícióba, jelen esetben z irányból egyenes mozgással érdemes ezt megtenni. Így egyrészt elkerülhetjük a munkadarabunkkal való ütközést másrészt, tudjuk a robot munkaterében nincs egyéb idegen tárgy, így a megközelítési pozícióba PTP (MoveJ) mozgással is beállhatunk, amely nagy elmozdulások esetén sokkal kedvezőbb mert nem kell az esetleges szingularitási problémákkal foglalkoznunk.

A munkadarab kontúrjának lekövetése után a szerszámot el kell emelni a munkadarabtól (erre használható ugyan az a megközelítési pozíció), amit szintén egy egyenes vonalú elmozdulással kell megtenni.

A megközelítési pozícióhoz egy nagyobb zóna értéket állíthatunk be, ezzel folytonossá téve a mozgást és javítva a ciklusidőt.

Az előzőekben leírt megfontolásokat figyelembevéve módosítsa a robot programját! Az új program megírása során törekedjen a mozgások optimalizálására.

Kérdések:

14. Másolja be a jegyzőkönyvbe a módosított programot!

5. feladat – Automatikus pályagenerálás

A RobotStudio segítségével automatikusan tudunk mozgáspályákat generálni a beimportált geometriák felhasználásával. Ehhez válasszuk a **Home** fület felül, majd a felső szalagmenüben nyissuk le a **Path** menüpontot és válasszuk az **Autopath** opciót. Ez az utasítás lehetővé teszi számunkra, hogy bármely geometria valamely éléhez mozgatva a kurzorunkat, automatikus pályát illesszünk a görbére. A **shift** billentyűt nyomva tartva a RobotStudio automatikus zárt görbét illeszt az élekre és arra generál pályát.

Módosítsa a programot a következők szerint:

1. A robot álljon Home pozícióba
2. Kövesse le a téglatest kontúrját
3. Álljon újra home pozícióba
4. Kövesse le a belső, íves geometria kontúrját
5. Végül álljon vissza home pozícióba

A program megírása során ügyeljen a megközelítési pozíciók helyes felvételére és a zóna adatok beállítására!

Az **autopath** utasítás paraméterezéséhez használja segítségül az **ABB RobotStudio – Operating Manual, Autopath** fejezetét.

Kérdések:

15. Másolja be a jegyzőkönyvbe az elkészített programot és magyarázza el működését!