T.C.

ERCİYES ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

MEKATRONİK MÜHENDİSLİĞİ BÖLÜMÜ

MEKATRONİK LABORATUVARI – II

COSİMİR LAB PROGRAMI VE UYGULAMALARI

DENEY SORUMLUSU

Arş. Gör. Mehmet Safa BİNGÖL

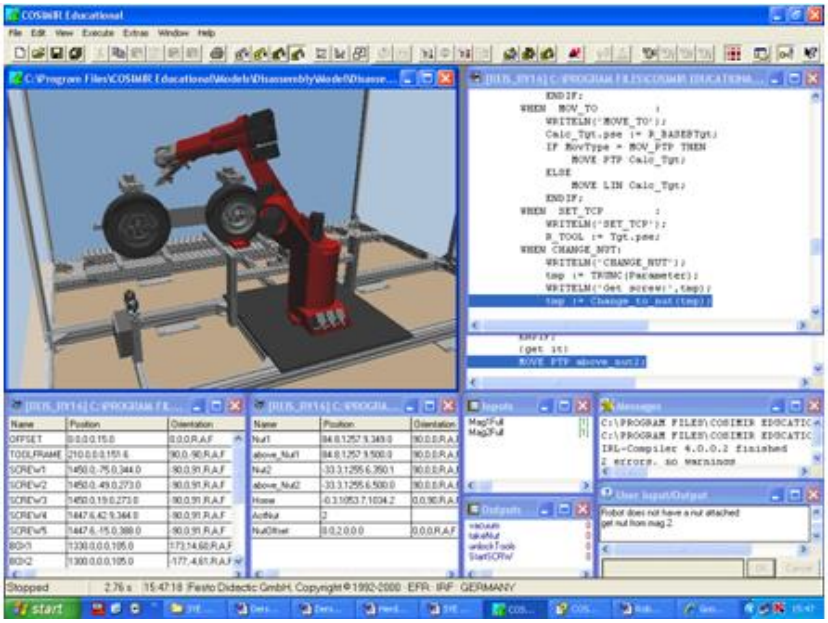
ŞUBAT 2023

KAYSERİ

COSİMİR LAB PROGRAMI VE UYGULAMALARI

1. GİRİŞ

Robot programlama eğitim sistemi olan Cosimir Lab programı ile imalatta kullanılan çeşitli tip robotların programlanması, bu robotların bulundukları çalışma düzeneklerinin üç boyutlu olarak modellenmesi ve hareketlerinin yine üç boyutlu olarak benzetimi mümkün olmaktadır. Böylece gerçek sistem üzerinde meydana gelebilecek herhangi bir nesne ile çarpışma sorununun daha önceden fark edilip düzeltilebilmesi mümkün kılınırken programlama sürecinin de kısalması sağlanmış olur.



Şekil 1. COSIMIR programı örnek çalışma ara yüzü.

Robot sistemlerini monte etmek ve ayırmak, taşımak ve kullanıma sokmak oldukça zor, fakat eğitimi oldukça ilginç konulardır. Bunun yanında endüstride kullanılan parçalar ve bunların montajlarını daha simülasyon aşamasındayken görebilmek son kullanıcının bilgilendirilebilmesi açısından oldukça önemli bir konudur.

Programın sağlamış olduğu bir diğer önemli faydada daha simülasyon aşamasında iken robot programlarının oluşturulup denenebilmesidir. Robot programının simülasyon aşamasında iken denenmesi ile sistem çalışması esnasında oluşabilecek ciddi hataları engellenebilmektedir.

Ayrıca Melfa-Basic gibi programlar sayesinde gerçek endüstriyel robotlar ile program arasında eş zamanlı çalışma gibi opsiyonlar bulunmaktadır. Bu işlem aynı PLC programında olduğu gibi online çalışma olarak gerçekleştirilebilmektedir. Bu programın kullanılmasındaki temel hedefler şu şekilde listelenebilir.

- Montaj işlemi süresince endüstriyel robotun birleştirilmesi
- Karmaşık montaj ortamlarında robotların eğitimi
- Karmaşık sistemlerin idare edilmesi
- Karmaşık sistemlerin bakımı, hizmeti ve hata tespiti
- Endüstriyel robotların planlaması

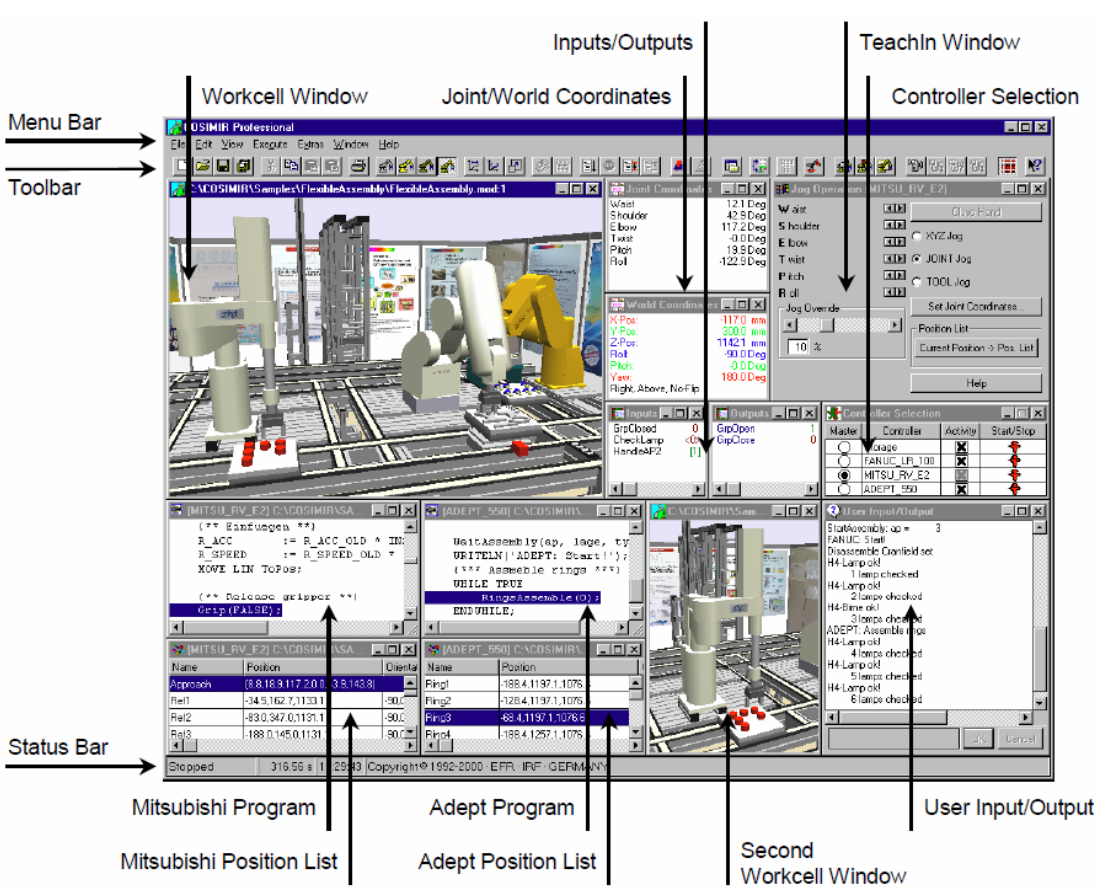
2. DENEYİN AMACI

Bu deneyde mekatronik mühendisliği öğrencilerinin ilerideki meslek hayatlarında faydalanabilecekleri COSIMIR LAB programı tanıtılacaktır. Bu program kullanılarak endüstriyel son kullanıcının isteklerini basit bir şekilde projelendirebilecekleri ve sistemlerin çalışmasının daha fabrika kurulmadan üç boyutlu olarak animasyon gerçekleştirebilecekleri gibi olanaklar sunulabilmektedir. Bu deney kapsamında öncelikle program tanıtımı gerçekleştirilecek, daha sonra ise seçilecek uygun bir proje üzerinde simülasyon uygulaması yapılacaktır.

3. ÖN BİLGİ

3.1. COSIMIR LAB Programı

Bu programda hedeflenen endüstriyel ortamda bulunabilecek pnömatik piston, yürüyen bant, endüstriyel robot kol, sensörler gibi çeşitli elemanların program kütüphanesinden kolayca eklenerek istenilen iş istasyonlarına benzer bir yapı kurmaktır. Bu amaçla program ana ekranında iş hücresi ekranı, eksen koordinatları, programlama ekranı gibi birçok alt bölüm bulunmaktadır. Şekil 2 de COSIMIR LAB programında örnek bir çalışma ekranı görülmektedir.



Şekil 2. COSIMIR programı ana çalışma ekranı.

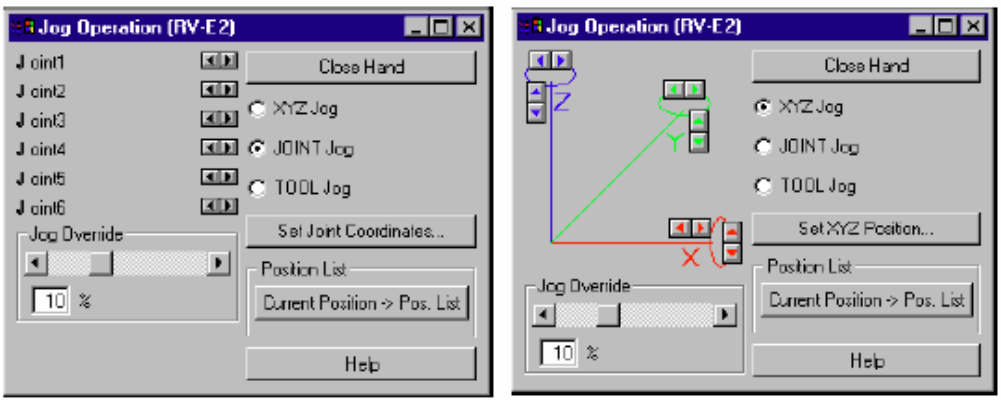
Program ana ekranında bulunan alt ekranların tanıtımı yapılacaktır. Şekil 3'te görülen ekranda sistem çalışmasının animasyonu üç boyutlu olarak görülebilmektedir. Bu görüntüye ait farklı gösterimler için View Menu altındaki seçenekler kullanılabilir.

Şekil 3. COSIMIR programı iş hücresi(workcell) ekranı.

Sistemde eğer robot kol kullanılacaksa bu robot kola ait parametrelerin gösterimi için Şekil 4'te kullanılan eksen koordinatları ve global koordinatlar ekranı kullanılır. Eksen koordinat ekranında robota ait her bir mafsalın aldığı açılar derece cinsinden, global koordinatlar ekranında ise robot end-efektörünün pozisyonu ve yönlenmesi görüntülenebilmektedir.

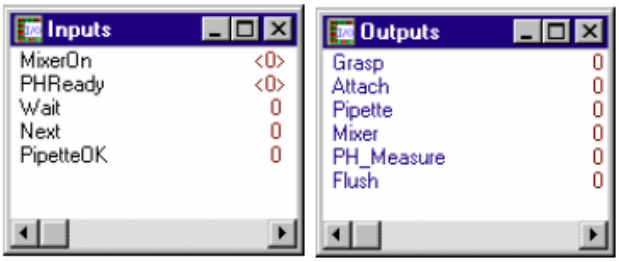
Şekil 4. COSIMIR programı eksen koordinatları ve global koordinatlar ekranı.

Sistemde bulunan robotun pozisyonu değiştirilmek istenirse aynı ekranlara benzer bir şekilde kullanım sağlayan Şekil 5'teki kontrol ekranları kullanılabilir.



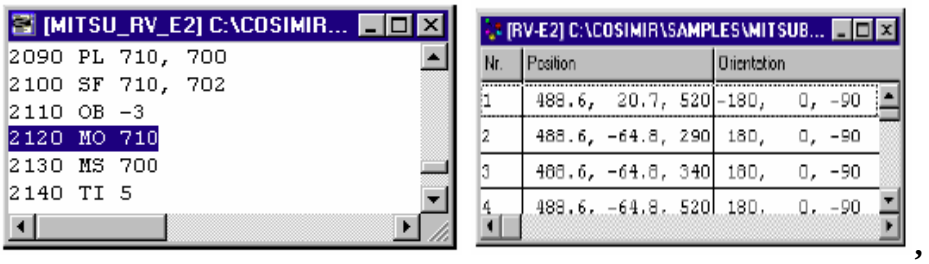
Şekil 5. COSIMIR programı eksen koordinatları ve global koordinatları kontrol ekranı.

Sistemde bulunabilecek; sensör, konveyör, gibi elemanların giriş ve çıkış değişkenlerinin görülebilmesi için şekil 6'da bulunan giriş çıkış ekranlarından faydalanılır.



Şekil 6. COSIMIR programı giriş çıkış parametreleri

Yapılacak olan simülasyonda yapılacak olan kontrolde sistem belirli bir koda ihtiyaç duyabilir. Buna örnek olarak uygulamasını gerçekleştireceğimiz bir robot kolun MELFA-BASIC'te yazılan bir kod verilebilir Şekil 7'de bu kod ekranı ve bu kodun kullanmış olduğu pozisyonların listesi görülmektedir.



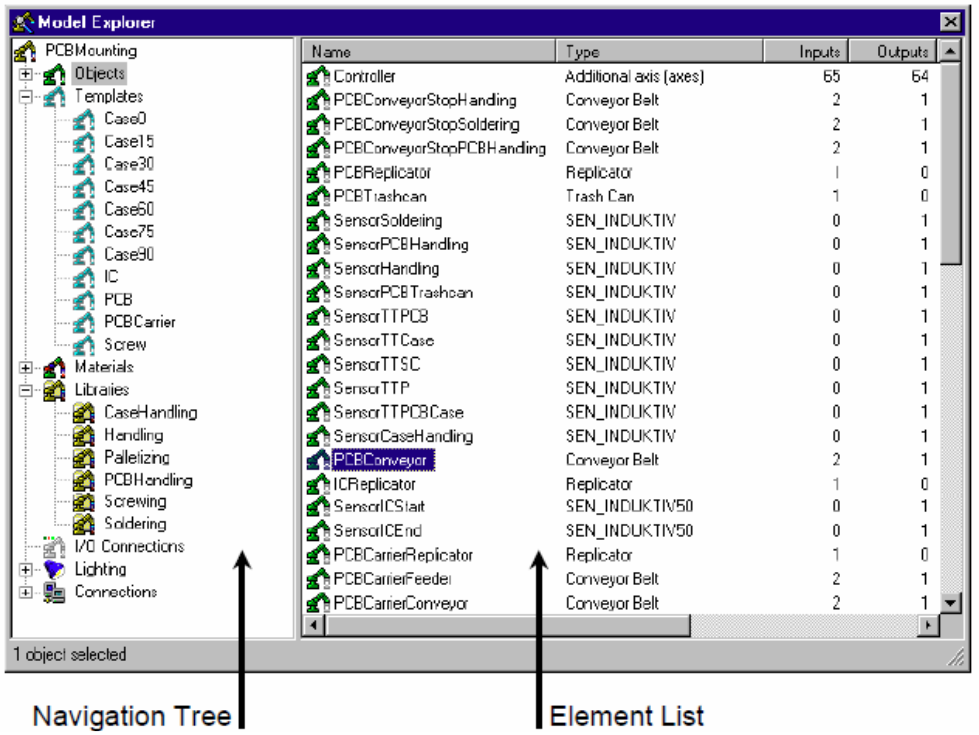
,

Şekil 7. COSIMIR programı kod ekranı ve pozisyonlar listesi

Sistem simülasyonu veya gerçek zamanlı uygulama esnasında kullanıcı girişi veya sistem durumunun gösterilebilmesi için bir çıkış ekranına ihtiyaç duyabilir. Şekil 8'de bu amaç için kullanılan kullanıcı giriş çıkış ekranı verilmiştir.

Şekil 8. COSIMIR programı kullanıcı giriş çıkış ekranı.

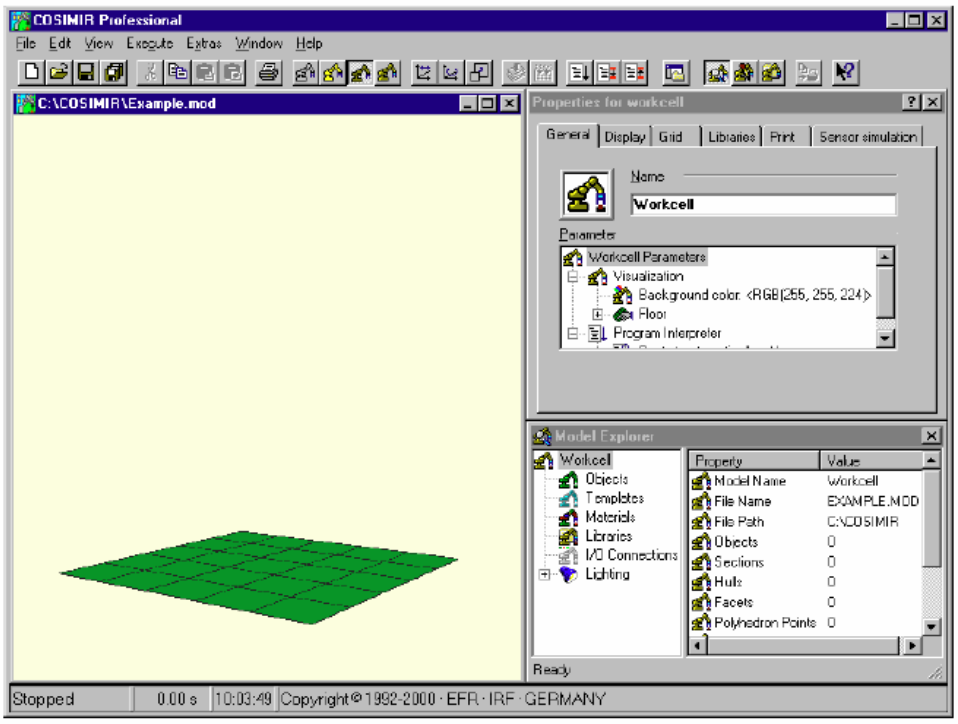
COSIMIR LAB programında bahsedilmesi gereken en önemli konulardan biride eleman kütüphaneleridir. Değişik lisanslara göre sahip olunan kütüphane sayısı artırılabilir. Simülasyon ekranına eklenmek istenen elemanın bu kütüphanede olması gerekmektedir. Şekil 9'da Model Explorer altında kütüphanede bulunan elemanlar görülebilmektedir.



Şekil 9. COSIMIR programı sistem kütüphanesi elemanları.

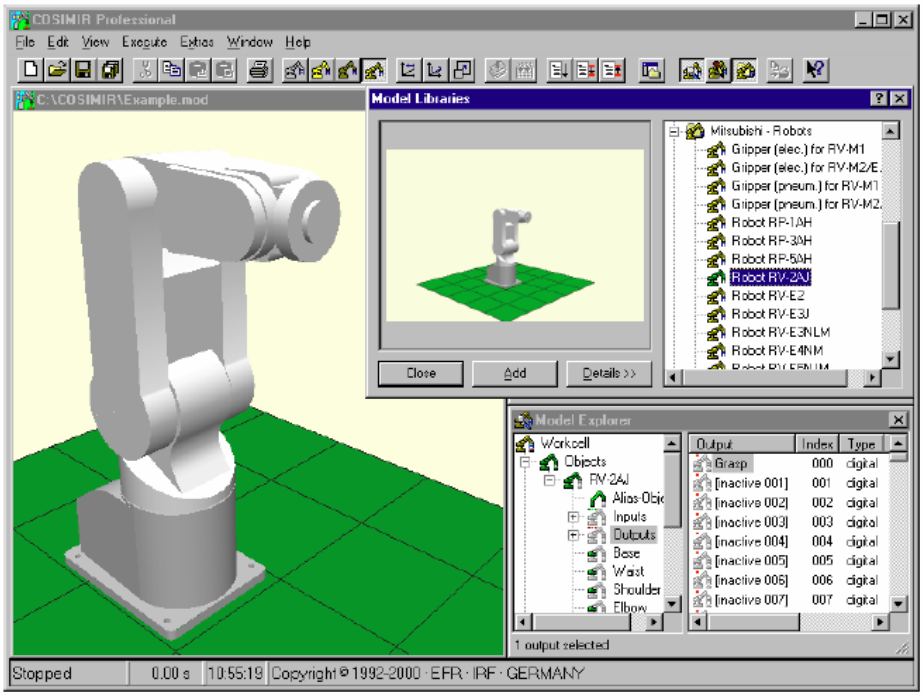
3.2. COSIMIR LAB Programı sistem modellenmesi

Sistem modellenmesine örnek olması açısından kütüphanede bulunan bir robot kol, gripper ve üst üste duran iki küpten oluşan bir senaryo kurgulanacaktır. Bu amaçla öncelikle Dosya menüsünden yeni workcell seçilerek Şekil 10'da görülen sekmelerden workcell kurulur.



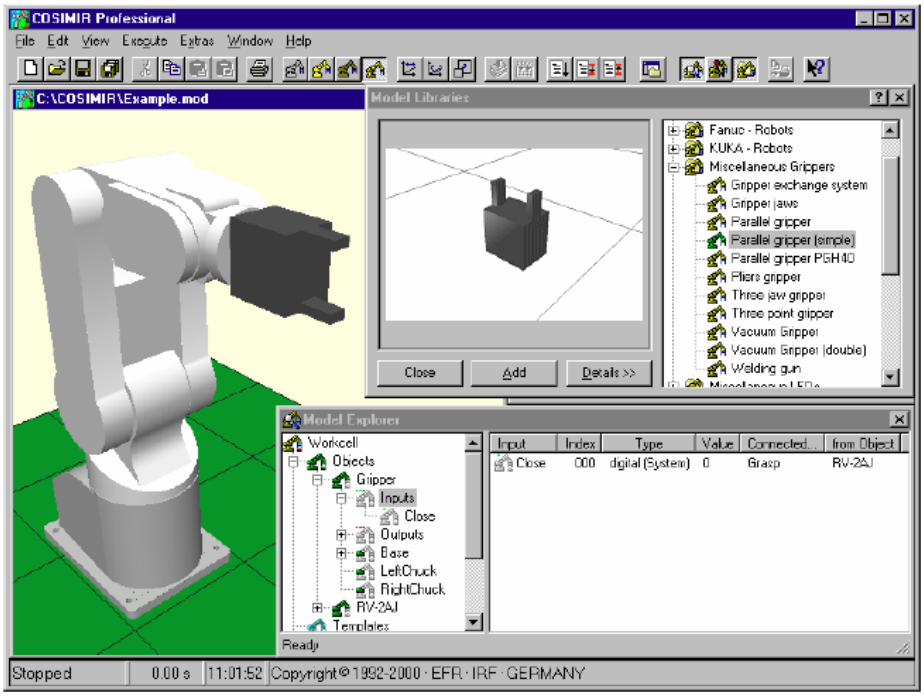
Şekil 10. COSIMIR programı workcell oluşturma.

Daha sonra model kütüphanesinden eklenmek istenen robot kol seçilerek Şekil 11'deki gibi ekrana eklenir. Model Explorer menüsünden robot kola ait giriş çıkış parametrelerinin ayarlanması sağlanır.



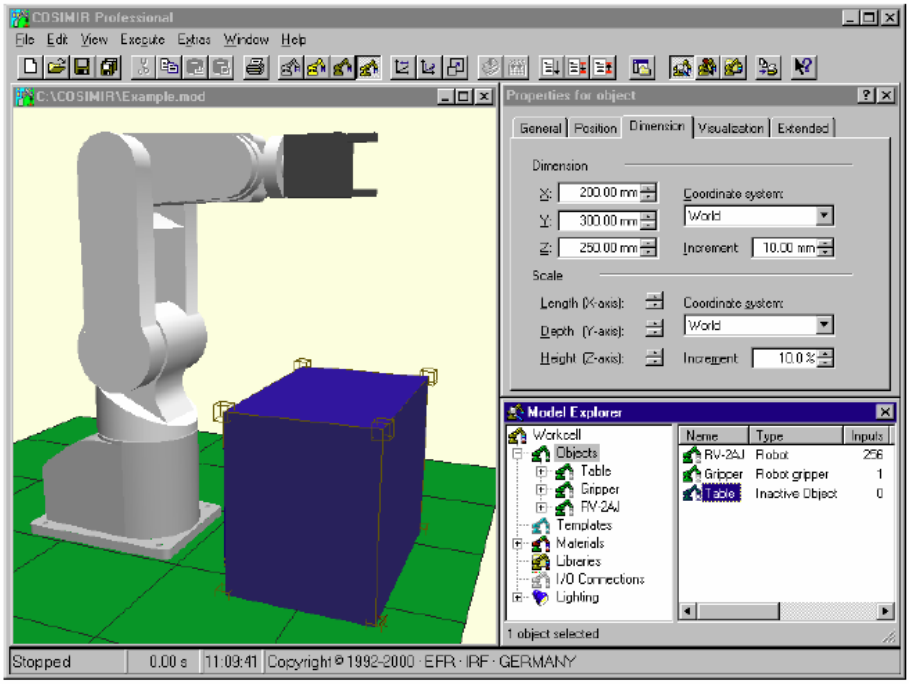
Şekil 11. COSIMIR programı kütüphaneden robot kol eklenmesi.

Aynı robot kol eklendiği gibi sistem kütüphanesinden robot kol ucunda kullanılacak olan gripper(tutucu) eklenmesi gerçekleştirilir.

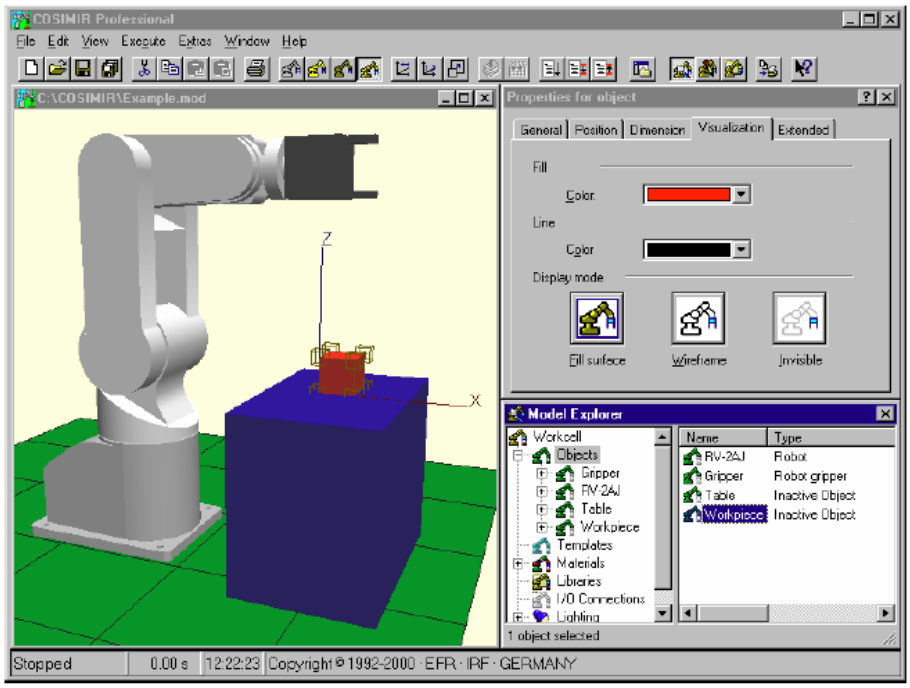


Şekil 12. COSIMIR programı kütüphaneden gripper eklenmesi.

Sırası ile aynı robot kol eklendiği gibi model kütüphanesinde bulunan Miscellaneous Primitives menüsü altında bulunan box nesneleri eklenir ve Şekil 13 ve Şekil 14'deki gibi boyut pozisyon gibi konfigürasyonları yapılır.



Şekil 13. COSIMIR programı kütüphaneden box eklenmesi.



Şekil 14. COSIMIR programı kütüphaneden eklenen box elemanının ayarlanması.

Görüldüğü gibi sistem modellenmesinde zengin kütüphanesi dolayısı ile simülasyon üzerine eleman eklenmesi ve ayarlanması oldukça kolay bir şekilde gerçekleştirilebilmektedir. Bu aşamadan sonra simülasyonun çalışması için sistemin programlanması gerekmektedir.

3.3. Melfa-Basic Robot Programlama dili

Burada robot programlama konusuna açıklık getirilecektir. Açıklamalarda, günümüzde de Mitsubishi robotlarının programlanmasında kullanılan *MELFA BASIC* dili baz alınacaktır. Öğrenilecek bu programlama konsepti diğer endüstriyel programlama dillerinde uygulanabilir.

Bir robot programı deklarasyonlar ve komutlardan oluşur. Bu komutalın bazıları robot kontrolörlerine özgüdür (her kontrolör için değişiklik gösterir).

Robot programının temel içeriği şu şekildedir.

- Hareket komutları
- İletişim ve haberleşme komutları
- Kontrolörün ayar değerlerini değiştiren komutlar.

Modern robot programlama dilleri modüler yapıdadır. Basit komutlardan oluşan komut setleri kolay anlaşılır ve uygulanır. Uzman olmayanlar dahi bu komutları kullanabilir.

BASIC nedir?

Melfa Basic, üniversal BASIC dili tabanlı bir robot programlama dilidir. BASIC, öğrencilere basit şekliyle programlama dili eğitimi vermek amacıyla ABD de Dartmounth Kolejinde geliştirilmiştir. Kelimenin açılımı "**B**eginners **A**ll-purpose **S**ymbolic **I**nstuction **C**ode" (temel seviye genel amaçlı sembolik komut kodu) şeklindedir.

O günden beri BASIC çok kullanılan bir dil olmuştur. Ancak bu dilin bir standardı yoktur. Bugün farklı amaçlar için geliştirilmiş bir çok BASIC sürümü(variant) mevcuttur. MELFA-BASIC de bir BASIC sürümüdür. Hareket kontrolü ve iş hücresindeki diğer aygıtlarla haberleşebilmesi için ilave komutlara sahiptir.

Programlama Stratejisi

Bir robot programı oluştururken takip edilecek genel bir strateji vardır. Şekil 15'teki şema bunu açıklamaktadır.

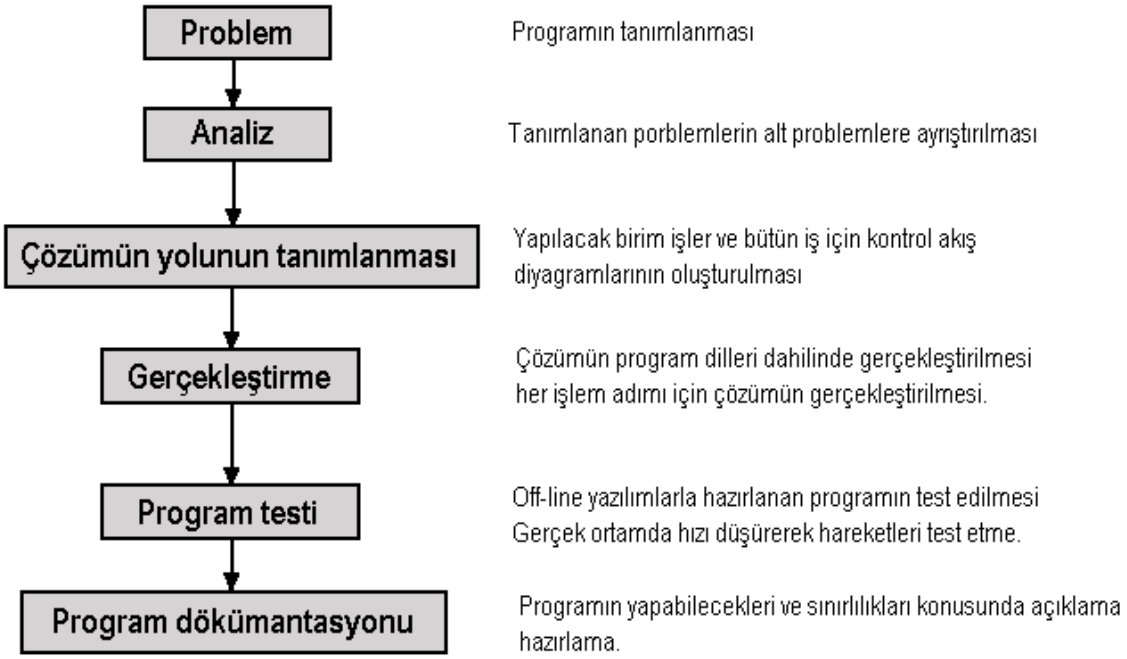
Programlama Tekniği

Eğer karşılaşılan programlama işi karmaşık ise program alt programlara bölünmelidir. Bu işlem problemin daha küçük ve basit yapılarda incelenmesini sağlar.

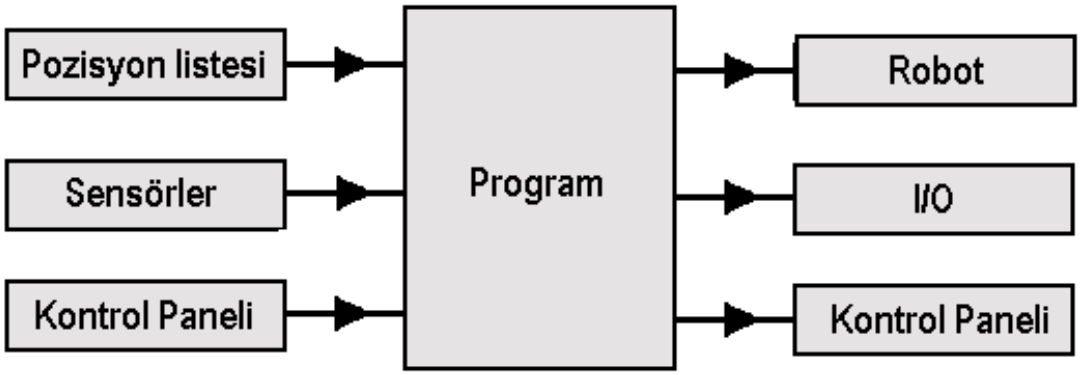
Alt programlar oluşturmak için programlama dilindeki fonksiyonlardan faydalanılır. Ana iş birçok alt iş grubuyla rahatlıkla düzenlenir. Bu yapısal programlama olarak adlandırılır.

Program Yapısı

Bir program yapılacak işin gerektirdiği bilgileri içerir. Bu bilgiler veri ve komut olarak ikiye ayrılır. Genellikle yapılan işle ilgili veriler programda bulunmaz. Bunlar çeşitli veri toplama birimlerinden derlenir. Veri toplama birimleri; veri hafızası (pozisyon listesi vs.), ölçme kontrol sistemleri (dokunma sensörü vs.), kullanıcı girişi (kumanda paneli) gibi çevre aygıtlarından oluşur.



Şekil 15. Programlama Stratejisi.



Şekil 16. COSIMIR programı kütüphaneden eklenen box elemanının ayarlanması.

Program komutları, komutları ve parametreleri içerir. MELFA-BASIC de program iki parçadan oluşur. Birincisi kullanılan verilerin işlenmesiyle ilgili bildirgelerdir (declarations). Diğeri komut ifadeleridir (statement). Bu ifadeler robot hareketlerini kontrol eder, I/O kanallarıyla çevresel aygıtlarla haberleşir, kontrol paneli aracılığıyla kullanıcıyı programın durumu hakkında bilgilendirir.

Programlamaya Hızlı Bakış

Robot programı yazmak otomasyon süreçleri ve bu süreçlerin alt birimlerine ayrılabilirliği konusunda derin bilgi gerektirir. Ayrıca karşılaşılan problem üzerinde çalışan bir program yazmak için programlama diline hakim olmak gerekir. Bu bölümde MELFA-BASIC IV dili baz alınarak robot programlama hakkında kısa ve pratik bilgiler verilecektir.

Açıklama

Her ne kadar programlama problemlerinin kesin bir çözümü olmasa da, genelde her türlü programın yazımında Şekil 16'da yer alan strateji uygulanır. Aynı iş için aynı sonucu veren birbirinden farklı birkaç çözüm olabilir.

Var olan programların sık bir şekilde revizyonu, geliştirilmesi ve değişen iş akışlarına uyarlanması tecrübe kazandırır. Böylece makul, kullanışlı ve hızlı programlar oluşturulabilir. Robot programlamaya yeni başlayanlar özellikle geniş çaplı karmaşık programları yazarken program yapısına dikkat etmelidir. Yeni başlayanlar program yapısını öğrenmekle bir imalat sürecinin, satır yapılı programlama metoduyla nasıl programlanacağını öğrenir.

MELFA-BASIC Program Yapısı

MELFA-BASIC, komut satırlarıyla programlama yapılan (satır yapılı) bir dildir. Her program satırı satır numarası ile başlar.
Program iki parçadan oluşur.

1. **Demeçler** bölümünde programcı, robot durum değişkenleri ve dış değişkenler haricindeki bütün değişkenleri tanımlar. Pozisyon değişkenleri pozisyon listesinden (*position list*) tanımlanır.

*Önemli not: program satırlarında **REM** kullanıldığında ardından yazılanlar kontrolör için komut değeri taşımaz. Programcı buraya kendi anlayabileceği program, işlem ismi vs. yazabilir.*
Örnek:

```
10 REM ***demeç***          ' programcının açıklaması'
20 DEF POS P1               ' P1 pozisyon değişkeninin tanımlanması'
```

2. **Komut ifadeleri** bölümü bütün programla komutlarını içerir.

Örnek:

Pozisyon listesindeki PGoto değeri P1 pozisyon değişkenine atanır. Ardından noktadan noktaya hareketle robot P1 pozisyonuna hareket eder.

```
30   REM ***komut ifadeleri***   'programcının açıklaması
40   P1 = PGoto                  'PGoto değerini P1' ata
50   MOV P1                      'P1 pozisyonuna git
60   END                         'program sonu
```

Ardışık Programlama

Ardışık programlamanın felsefesi, sade, düz akışlı programlama üzerinedir. Ardışık programlamada satırlar arası atlamalardan ve dallanmalardan kaçınılır. Böylece program kodları kolay okunur ve anlaşılır olur. Programın içinde dallanma ne kadar fazla ise okunması o derece zor olur.

Aşağıdaki örnek incelendiğinde bu daha iyi anlaşılacaktır.

P1 pozisyon değişkeni tanımlanır. Eğer girişin 0.biti 0 ise PGoto değeri P1 değişkenine atanacaktır. Giriş biti 1 ise P_SAFE değeri P1 değişkenine atanacaktır.

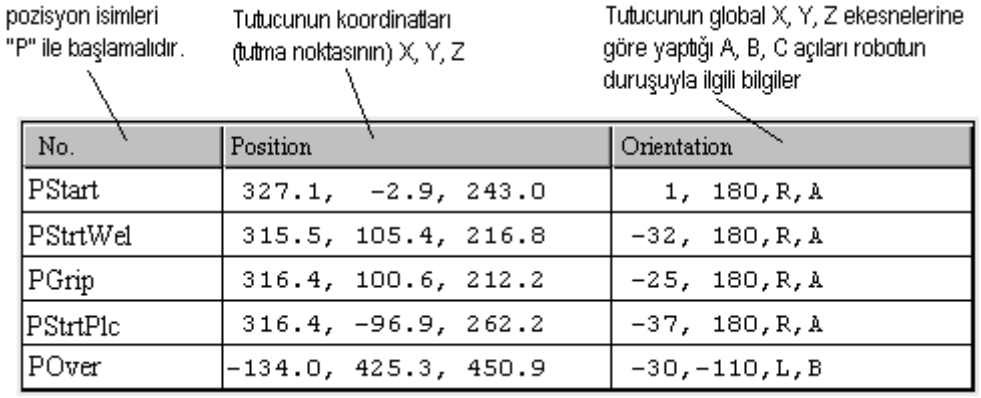
```
10      REM***dallanmasız program***
20      REM*** değişken tanımlaması***
30  DEF POS P1                                  'P1 pozisyonun adı
40  REM***şart ifadesi***
50  IF M_IN(0) = 0 THEN P1 = PGoto ELSE P1 = P_SAFE
    eğer sayısal girişin 0 numaralı kanalının değeri 0 a eşitse
    "goto" pozisyonuna git. Değilse "safe" pozisyonuna git.
 60 MOV P1                              'şartla seçilen pozisyona git.
 70 END                             'Program sonu

 10 REM***dallanmalı program***
 20 REM***değişken tanımlama***
 30 DEF POS P1                                  'P1 pozisyonun adı
 40 REM***şart ifadesi***
 50 IF M_IN(0) = 0 THEN GOTO 100 ELSE GOTO 200
    eğer sayısal girişin 0 numaralı kanalının değeri 0 a eşitse
    100 numaralı satıra git. Değilse 200 numaralı satıra git.
 60 MOV P1                              'şartla seçilen pozisyona git.
 70 END                                 'Program sonu
100 P1 = PGoto                          'PGoto pozisyonunu seç
110 GOTO 60                             '60 numaralı satıra git.
200 P1 = P_SAFE                         'P_SAFE pozisyonunu seç
210 GOTO 60                             '60 numaralı satıra git
```

Robot Hareketleri

Her robot programının temel gayesi robotu hareket ettirerek son etkileyici ile montaj, taşıma, kaynak gibi işleri yapmaktır. Bu nedenle MELFA BASIC deki hareket komutları çok önemlidir.

En çok bilinen hareket tipleri lineer hareket (lineer enterpolasyon) ve noktadan noktaya hareket (eksen enterpolasyonu) dur. Pozisyonlar robotun hareket hattını oluşturan noktaları tanımlar. Pozisyon listesinde tanımlıdırlar. Öğretme (teach-in) metodu, tutucunun pozisyon bilgilerini edinmede en çok bilinen ve uygulama alanı bulan metottur. Veriler pozisyon listesinde saklanır. Her programın kendi pozisyon listesi vardır. Program ve pozisyon listeleri isimleriyle atanırlar. Pozisyon listesinin adı program ismine uygun olmalıdır.

Program içerisinde kullanılan pozisyon isimleri pozisyon listesinde de bulunmalıdır. Pozisyon listesinde pozisyon isimleri program içerisinde pozisyon değişkenlerine değerlerin atanmasında kullanılmaktadır.



No.	Position	Orientation
PStart	327.1, -2.9, 243.0	1, 180,R,A
PStrtWel	315.5, 105.4, 216.8	-32, 180,R,A
PGrip	316.4, 100.6, 212.2	-25, 180,R,A
PStrtPlc	316.4, -96.9, 262.2	-37, 180,R,A
POver	-134.0, 425.3, 450.9	-30,-110,L,B

Şekil 17. Pozisyon listesi.

Lineer Hareket

Takım merkezinin önceden tanımlanmış bir doğrusal yörünge üzerinde hareket ettirilmesi sık sık başvurulan bir yöntemdir. Çoğu montaj operasyonu robotun doğrusal yörüngede hareketini gerektirir. Doğrusal hareketler iki nokta ile tanımlanır. Programlamada hedef pozisyonun belirtilmesi doğrusal hareket için yeterlidir.

```
10    MVS  P1    (P1 hedef pozisyon. Bu komutla bulunulan pozisyondan hedef
                  pozisyona doğrusal bir yörüngede hareket edilir.)
```

Karmaşık yörüngelerde hedef noktaya ulaşmak için arada üçüncü bir nokta tanımlanmalıdır. Robot bu ara noktalarda durmadan geçer ve hedef noktaya ulaşır.

Örnek:

P1 den P4 e kadar pozisyonlar pozisyon listesinde tanımlanır. Robot doğrusal bir yörünge üzerinde P1 den P2 ve P3 noktalarından geçerek P4 noktasına hareket eder. CNT 1 değişkeni robot kontrolörünün ara noktalarda yumuşak geçiş yapmasını sağlar. Robot doğrusal yörünge boyunca SPD komutu ile tanımlanan hızla hareket eder (650 mm/s).

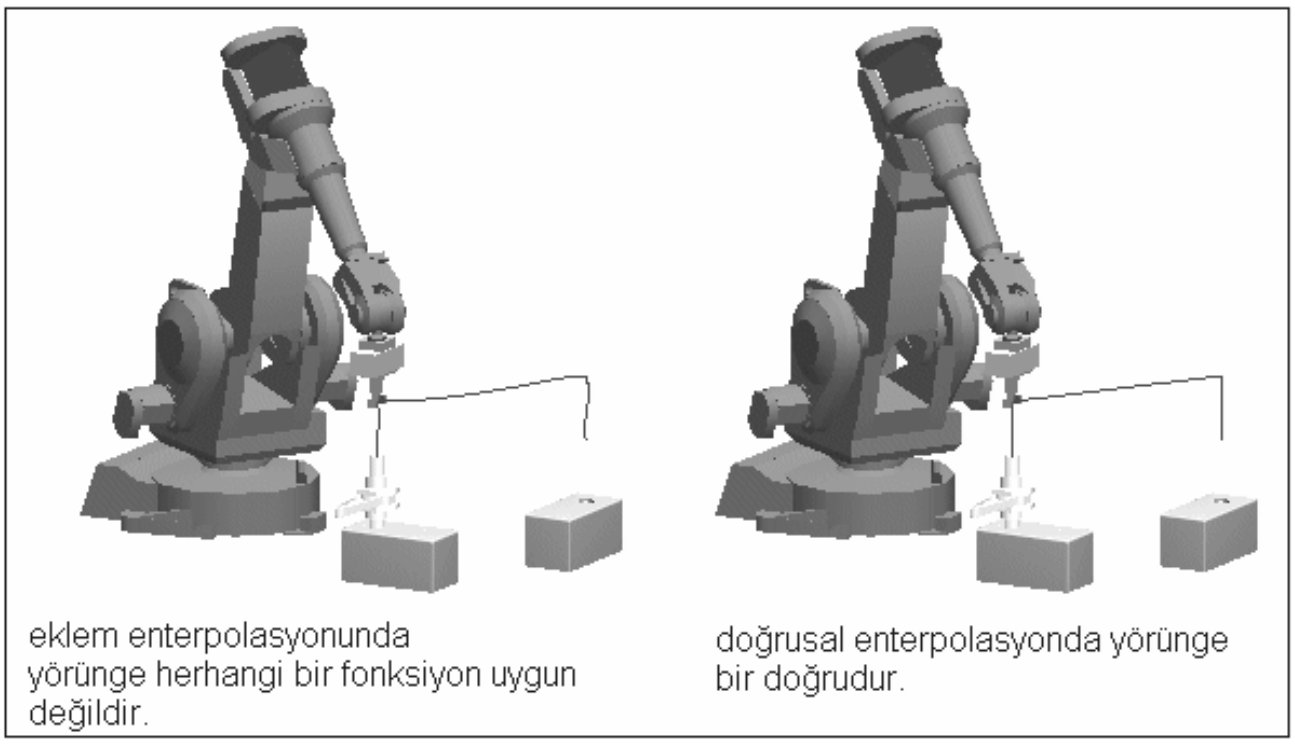
```
1     REM ***örnek program***
20    REM ***değişken tanımlama***
30    REM ***komutlar***
40    CNT 1                    (yumuşak geçişler etkin)
50    SPD 650                  (takım merkezinin hızı 650 mm/s)
60    MVS P1                   (P1 pozisyonuna doğrusal hareket)
70    MVS P2                   (P2 pozisyonuna doğrusal hareket)
80    MVS P3                   (P3 pozisyonuna doğrusal hareket)
90    MVS P4                   (P4 pozisyonuna doğrusal hareket)
100   END                            (program sonu)
```

Noktadan Noktaya Hareket

Parçaların bir P1 noktasından başka bir P2 noktasına taşınması gibi robot hareketlerinde noktadan noktaya hareket kullanılır. Noktadan noktaya harekette izlenen yörüngenin bir önemi yoktur. Hareketin başladığı ve bittiği noktalar önemlidir. Programda hedef pozisyonun tanımlanması gereklidir. Her eksen motoru hedef pozisyona ulaşmak için kendi üzerine düşen hareketi bir diğerinden bağımsız olarak gerçekleştirilir. Bu nedenle noktadan noktaya harekette takım merkezinin izlediği yörüngenin net bir fonksiyonu yoktur. Noktadan noktaya hareket kullanılırken izlenecek yörüngenin net olmayışı çarpışma sorunlarına yol açabilir. Robotlarla çalıştıkça kazanılan tecrübe bu tür harekette yörüngenin önceden kestirilmesine yardımcı olur.

Örnek:

```
1     REM ***örnek program***
10    REM ***değişken tanımlama***
20    REM          P1 den P3 e kadar pozisyonlar pozisyon listesinde tanımlı
30    REM ***komutlar***
40    JOVRD 60                    (eklem hızları %60 düzeyinde)
50    MOV P1                      (takım merkezinin hızı 650 mm/s)
60    MOV P2                      (P1 pozisyonuna doğrusal hareket)
70    MOV P3                      (P2 pozisyonuna doğrusal hareket)
80    END                               (program sonu)
```



Şekil 18. Lineer ve noktadan noktaya hareket durumunda yörünge.

Tutucu Kontrolü

Robot, iş parçası üzerinde gerçekleştireceği işlemleri takımlar ve tutucular ile yerine getirir. MELFA BASIC dili tutucuları HOPEN ve HCLOSE komutları ile kontrol eder. Aşağıdaki örnekte bir iş parçasının nasıl tutup kaldırılacağına ilişkin bir program görülmektedir.

```
10    REM ***örnek program***
20    REM is parçası kaldırma
30    HOPEN 1                    (1 numaralı tutucuyu aç)
40    MOV P1                     (P1 noktasına gel)
50    HCLOSE 1                   ( 1 numaralı tutucuyu kapat)
60    MVS P1,-50                 (iş parçasını 50 mm kaldır)
70    END                              (Program sonu)
```

Takımlar (kaynak torcu, lazer kesici, polisaj parlatıcısı vs.) ve tutucular robot kontrolörüne sayısal giriş-çıkış ara yüzü (I/O interface) ile bağlanır. Bu nedenle robot iş hücresi COSIMIR ile simüle edilirken tutucu, robotun sayısal giriş-çıkış(digital I/O) ara yüzüne *model explorer* üzerinden ayarlanarak bağlanmalıdır. Robot çalıştırılmadan önce takım yada tutucu giriş-çıkış ara yüzü üzerinde etkin edilmelidir. Bu işlem iki basamakta gerçekleştirilir.

1. Bir sayısal giriş kanalı etkin olan rakım yada tutucu için ayarlanır.
2. Bir sayısal çıkış kanalı takım yada tutucudan gelecek geri besleme için ayarlanır. Tutucu geri bildirim verdiğinde robot sonraki hareketlerine geçer.

MELFA-BASIC de sayısal çıkış kanalları M_OUT ifadesinin (M_OUT(1)=1) şeklinde kullanılmasıyla kontrol edilir. Yukarıdaki yazımda 1 numaralı sayısal çıkış sinyal vermektedir. Bu sinyal gerekli bağlantılar yapıldıktan sonra bir dış donanımın devreye girmesi için kullanılabilir. Bir sensörden alınan bilginin programda şart ifadesi olarak kullanılması aşağıda görülmektedir. Sensör 1numaralı girişe bağlıdır. Noktalı yere yazılacak komut şart sağlanırsa gerçekleştirecektir.

```
20    IF M_IN(1) = 0 THEN …
```

Bir işlemin yapılması için bir şart sinyali yerine zaman geciktirmesi de yapılabilir. İşlem belirtilen zaman sonunda gerçekleştirilecektir.

Aşağıdaki örnekte robot tutucuyu kapadıktan sonra P1 pozisyonunda 1 saniye bekleyecektir.

```
10    REM ***ornek program***
20    REM ***pozisyon tanımlama***
25    REM          P1,P2 Pozisyonları pozisyon listesinde tanımlıdır.
27    REM *** iş tanımlaması***
30    REM parça kaldırma
40    MOV P1                          (P1 noktasına gel)
50    HOPEN 1                         ( 1 numaralı tutucuyu aç)
55    DLY1                            ( bir saniye bekle)
60    MVS P2                    (doğrusal enterpolasyonla P2 ye git)
65    HCLOSE 1                        ( tutucuyu kapat )
70    DLY 1                           ( bir saniye bekle)
80    END                                   ( Program sonu )
```

GİRİŞ/ÇIKIŞ kanallarından haberleşme (I/O communication)

MELFA-BASIC programı sayısal giriş-çıkış kanallarıyla haberleşmek için robot durum değişkenlerini kullanır (detaylı bilgi için ilgili bölüme bakınız). Bir dış donanımı (tutucu, taşıyıcı bant vs.) sürmek için M_OUT olarak tarif edilen yazmaç kullanılır. Bu yazmaç (M_OUT(1) =1) şeklinde yazıldığında 1 numaralı çıkış kanalına bağlı donanım kontrol sürülebilir. M_OUT haricindeki diğer durum değişkenleri sadece okunabilir özelliktedir. Çevre donanımlardan ve aygıtlardan gelen sayısal bilgileri alır ve girilen değerle karşılaştırırlar. Karşılaştırma sonucunda bir işlem yapılır veya şart sağlanana kadar beklenir. IF M_IN(4) = 1 THEN…(4 numaralı giriş kanalı set olursa belirtilen işlemi yap.) WAIT M_IN(4)=1 (4 numaralı giriş kanalı set olana kadar bekle).

Karmaşık sistemlerde çeşitli aygıtların birbirleri ile haberleşmesi gerekir. Bu durumlarında iletişim protokollerine başvurulur. Bu gibi durumlarda çoğunlukla merkezi işlemci bir PLC yada bilgisayar olmaktadır. Robot kontrolörü bu durumlarda bir PLC üzerinden sisteme bağlanır. Şekil 19'daki örnek bir PLC ile robot kontrolörünün bağdaşmasını gösteriyor. Burada PLC üst düzey kontrol sistemini temsil etmektedir. Robot ve diğer iş hücresi aygıtlarının birbiri ile eş zamanlı ve çalışmasını temin eder.. PLC robot kontrolörü içindeki faklı programları farklı zamanlarda çalıştırarak hangi işlemin ne zaman yapılacağına karar verir. PLC robot kontrolörüne I/O ara yüzü ile bağlıdır. İletişim sinyalleri hakkındaki bildiri şöyledir;

S_ Data1	S_ Data2	Program
0	0	program0
0	1	program1
1	0	program2
1	1	program3

Şekil 19. PLC ile robot kontrol örneği.

Yapılacak 4 farklı işlemin her biri için bir program yazılmıştır. Bu programları sayısal olarak ifade etmek için S_Data1 ve S_Data2 giriş kanalları atanmıştır. İkili sayı düzeninde 4 sayısını ifade etmek için iki basamak gerekmektedir. Yukarıda da görüldüğü gibi 00 program0'ı, 11 program3'ü ifade eder. Bunların dışında robotla PLC'nin anlaşmasını sağlayan iki sinyal daha vardır. Bunlar S_ready ve S_valid sinyalleridir. Robotun çalışmadığı anda bir program almaya hazır olduğunu göstermek için S_ready sinyali kullanılacaktır. PLC S_valid sinyali ile gönderdiği program numarasını robot kontrolörüne kabul ettirir.
Normal çalışma sırasında anlaşma sinyallerinin çalışması şu şekilde olmalıdır.

1. PLC Program numarası yalnızca robottan S_ready sinyali verirse gönderecektir.
2. PLC, programı S_data 1 ve S_data 2 kanallarından program numarasını gönderir ve ardından onaylanması için S_valid kanalından sinyal gönderir.
3. Eğer program numarası doğruysa robot S_ready sinyalini göndermeyi keser. Program çalışır.

4. Program sonlandırıldığında robot S_ready kanalından sinyal verir ve yeni programın gönderilmesini bekler.
5. Program çalışmaya başladıktan sonra PLC S_valid kanalının sinyalini 0 yapar. Robottan S_ready sinyalinin gelmesini bekler.

Bağlantıların doğru yapıldığından emin olunmalıdır. Şekil 20'deki gibi bir durum hazırlanabilir.

sinyal	Robot kontrolörü	Adres	Anlamı
S_Data1	giriş	M_IN(10)	program numarası
S_Data2	giriş	M_IN(11)	program numarası
S_ready	çıkış	M_OUT(10)	anlaşma sinyali
S_valid	giriş	M_IN(12)	anlaşma sinyali

Şekil 20. PLC ile robot kontrol örneği sonucu.

Program numarası iki kanal üzerinden aşağıdaki gibi hesaplanır:

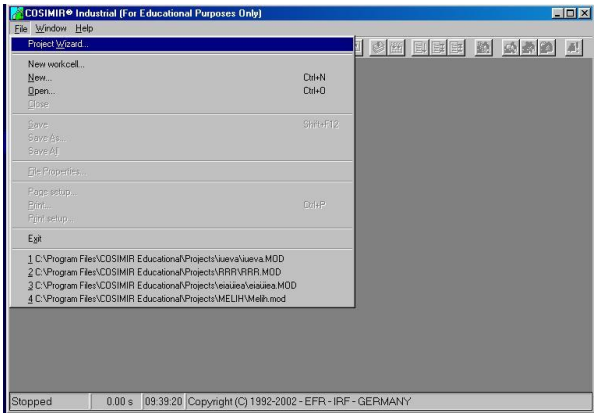
PrgNo = M_IN(11) + 2 * M_IN(10)

```
10 REM *** örnek program ***
20 REM *** tanımlamalar ***
30 DEF INTE PrgNo                    (integer değişkenli bir program numarası)
40 REM *** açıklamalar  ***
50 M_OUT(10) = 1                     (robot hazır)
60 WAIT M_IN(12) = 1                 (uygun verisi gelene kadar bekler)
70 PrgNo = M_IN(11) + 2 * M_IN(10)   (program numarasını oku)
80 M_OUT(10) = 0                     (robot çalışmaya başla)
90 SELECT PrgNo                      (program numarasını kontrol et)
100 CASE 0                           (program 0 seçilirse)
110 CALLP "program0"                 (program 0 çalışsın)
120 BREAK
130 CASE 1                           (program 1 seçilirse)
140 CALLP "program1"                 (program 1 çalışsın)
150 BREAK
160 CASE 2                           (program 2 seçilirse)
170 CALLP "program2"                 (program 2 çalışsın)
180 BREAK
190 CASE 3                           (program 3 seçilirse)
200 CALLP "program3"                 (program 3 çalışsın)
210 BREAK
220 END SELECT
230 GOTO 50                          (başlama noktasına git)
240 END                              (program sonu)
```

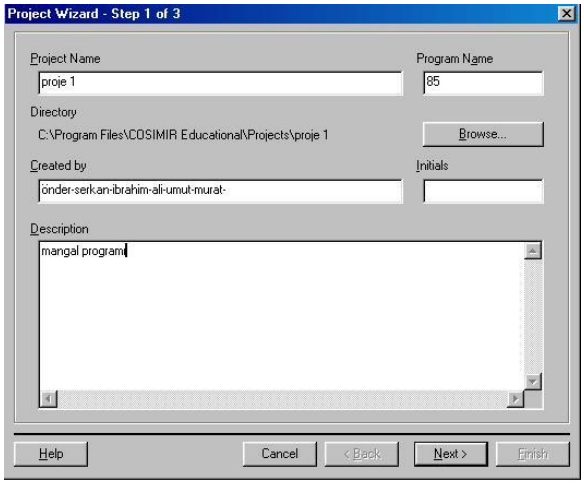
4. DENEYİN YAPILIŞI

4.1. El Sallayan robot uygulaması

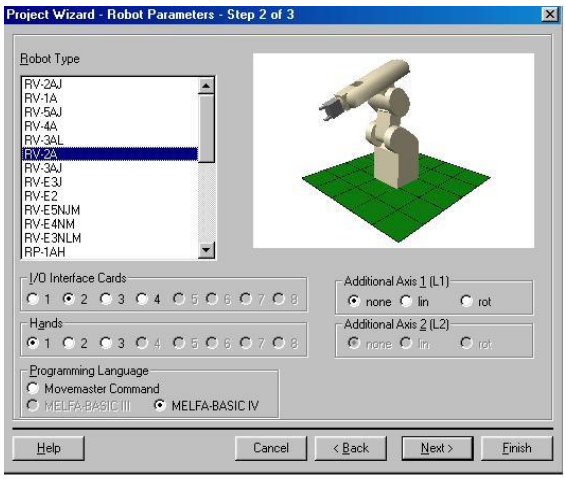
1. COSIMIR Industrial programı açılır. *File* menüsünden *Project Wizard* tıklanır.

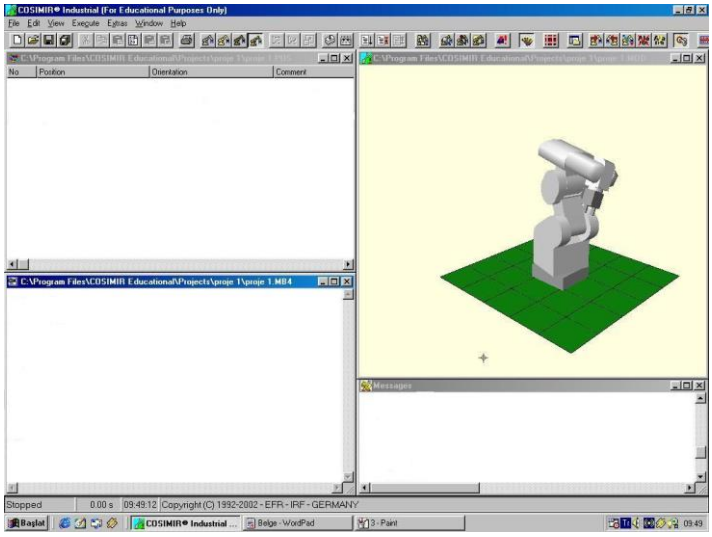


2. Açılan pencerede proje adı, program adı, kim tarafından yapıldığı ve özellikleri isteğe bağlı olarak doldurulur. *Next* tıklanarak sonraki aşamaya geçilir**.**

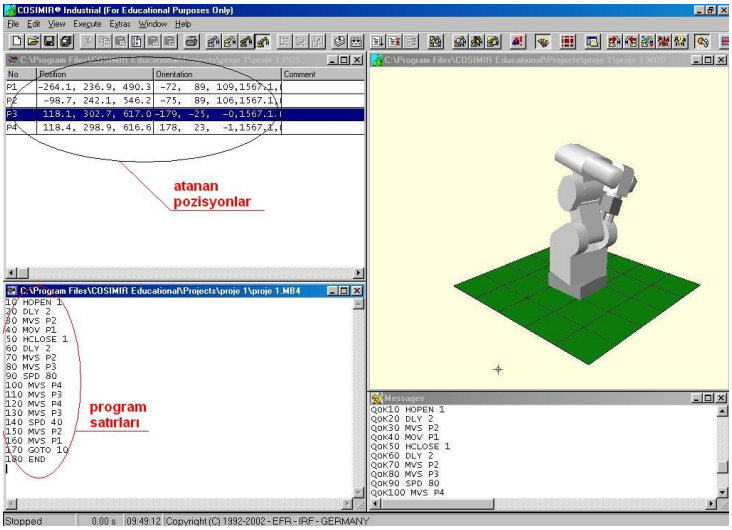


3. Bu pencereden kullanılacak robot çeşidi, bu robotun hangi dilde yazılacağı "Melfa Basic" belirlenir. *Finish* sekmesine basıldığında robot, programlanmaya hazır hale gelmiş olur.



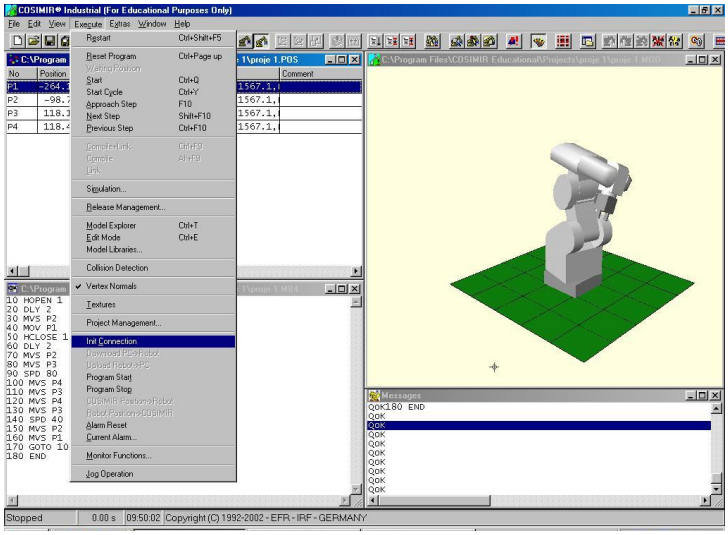


4. Daha sonra robota istenen pozisyonlar atanır ve program yazılabilir. Biz robota 4 farklı konum atayarak aşağıdaki programı yazdık.

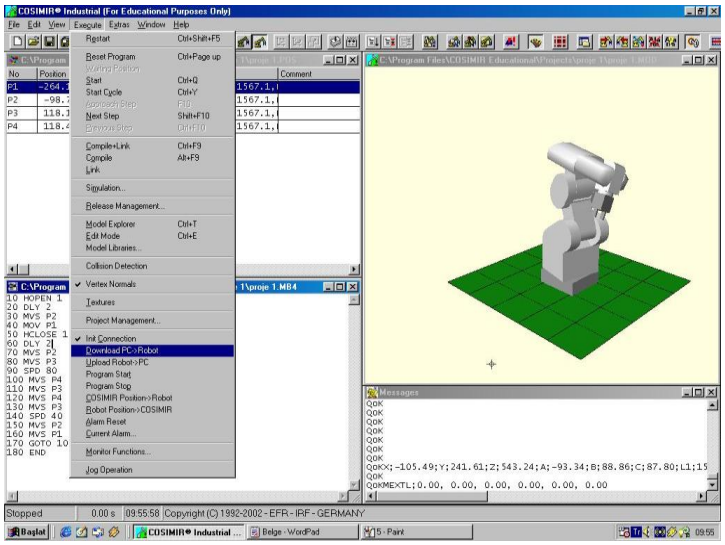



```
10 HOPEN 1
20 DLY 2
30 MVS P2
40 MOV P1
50 HCLOSE 1
60 DLY 2
70 MVS P2
80 MVS P3
90 SPD 80
100 MVS P4
110 MVS P3
120 MVS P4
130 MVS P3
140 SPD 40
150 MVS P2
160 MVS P1
170 GOTO 10
180 END
```

5. Bundan sonra *Execute* menüsünden *Init connection* tıklanarak robotla bağlantı sağlanır



6. Yazılan programı robota aktarma işlemine geldik. Bunun için *Execute* menüsünden *Download PC -> Robot* tıklandığında aşağıdaki pencere açılır.



Burada program adı doğru yazılmalıdır. Önemli olan robot kontrolör de var olan bir programın üzerine yeniden kayıt yapmamaktır. Çünkü bu durumda önceki program silinir. Biz, projeye başlarken program adını 85 olarak belirttiğimiz için bu kısma 85 yazacağız. *OK* sekmesi tıklanır.