ZEBEX

TO BE THE BEST IN AUTO ID

WINTASKGEN PROGRAMLAMA KILAVUZU

BİLKUR

Barkod Çözümlerinde Aklın Yolu "BİLKUR"

www.bilkur.com.tr

İ Ç İ N D E K İ L E R

WinTaskGen Programının Kurulumu

1) Zebex PDL20–16 paketinde bulunan CD'yi CD-ROM sürücüsüne takın.

2) Masaüstünde bulunan ve yandaki şekilde de görülen Bilgisayarım simgesini çift tıklayarak açın.

3) Bilgisayarım penceresinin içindeki WintaskGen **(D:)** simgesini çift tıklayarak açın.

Not: Yandaki resimde görülen WintaskGen **(D:)** satırındaki **D** harfi bilgisayara göre değişkendir.

4) WintaskGen (D:) penceresi içerisinde bulunan ve yandaki şekilde de görülen WinTaskGenV302 klasörünü çift tıklayarak açın.

Not: Yandaki resimde görülen WintaskGenV302 satırındaki **V302** programın sürümünü belirttiğinden tarihe göre değişkendir.

5) WintaskGenV302 penceresi içerisinde bulunan ve yandaki şekilde de görülen Setup simgesini çift tıklayarak çalıştırın.

6) Setup simgesi çift tıklandığında açılan **Welcome** Diyalog kutusunda **Next** düğmesini tıklayın.

7) 6.adımda tıklanan Next düğmesi ile açılan **User Information** diyalog kutusunda **Name** kutucuğuna isminizi, **Company** kutucuğuna firma adını yazarak **Next** düğmesini tıklayın.

8) 7.adımda tıklanan Next düğmesi ile açılan **Choose Destination Location** diyalog kutusunda **Destination Directory** bölümünde Wintaskgen programının kurulması istenilen klasörü (dizin) seçerek **Next** düğmesini tıklayın.

9) 8.adımda tıklanan Next düğmesi ile açılan **Select Program Folder** diyalog kutusunda **Program Folder** bölümüne Wintaskgen programının kurulum sonrası çalıştırma klasörünün adı yazıldıktan sonra **Next** düğmesini tıklayın.

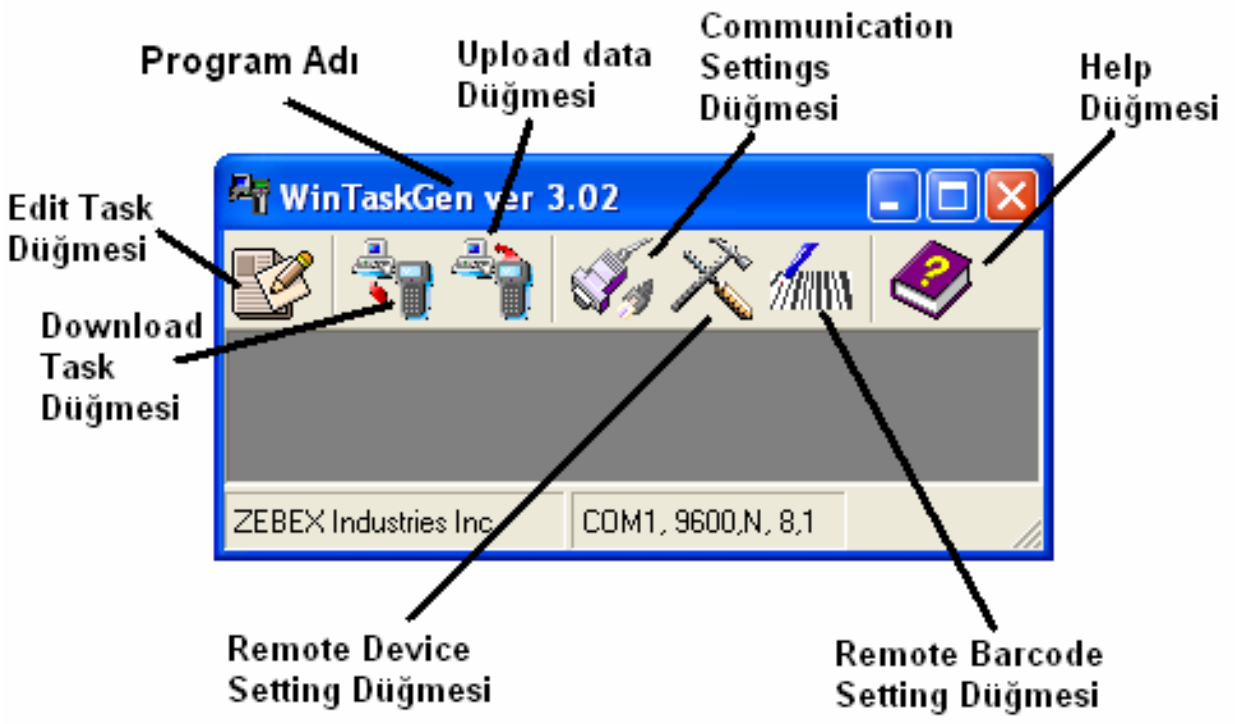
Not: Program Folder bölümünde çalıştırma klasörü otomatik olarak **WintaskGen** olarak seçili durumdadır.

10) 9.adımda tıklanan Next düğmesi ile açılan **Start Copying Files** diyalog kutusunda 7 nci adımdan itibaren girilen bilgilerin doğruluğu kontrol edildikten sonra **Next** düğmesini tıklayarak programın kurulumunu başlatın.

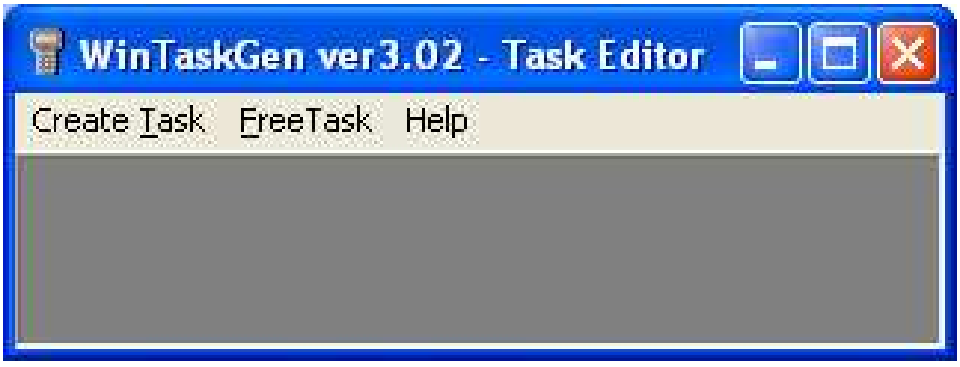
11) WintaskGen Ver 3.02 Programın kurulumunun tamamlanmasından çıkan **Setup Complete** diyalog kutusunda **finish** düğmesini tıklayarak kurulumu bitirin.

WinTaskGen Programının Çalıştırılması

WintaskGen Programını çalıştırabilmek için Başlat/programlar/Wintaskgen menüsünden Wintaskgen simgesini çalıştırın. Wintaskgen programının ilk görüntüsü aşağıdaki gibidir.



WinTaskGen programında program yazım ekranına girebilmek için yukarıdaki ekranda da görülen Edit Task düğmesi tıklanmalıdır. Edit Task düğmesi tıklandığında açılan Task editör penceresinin görünümü aşağıda ki gibidir.

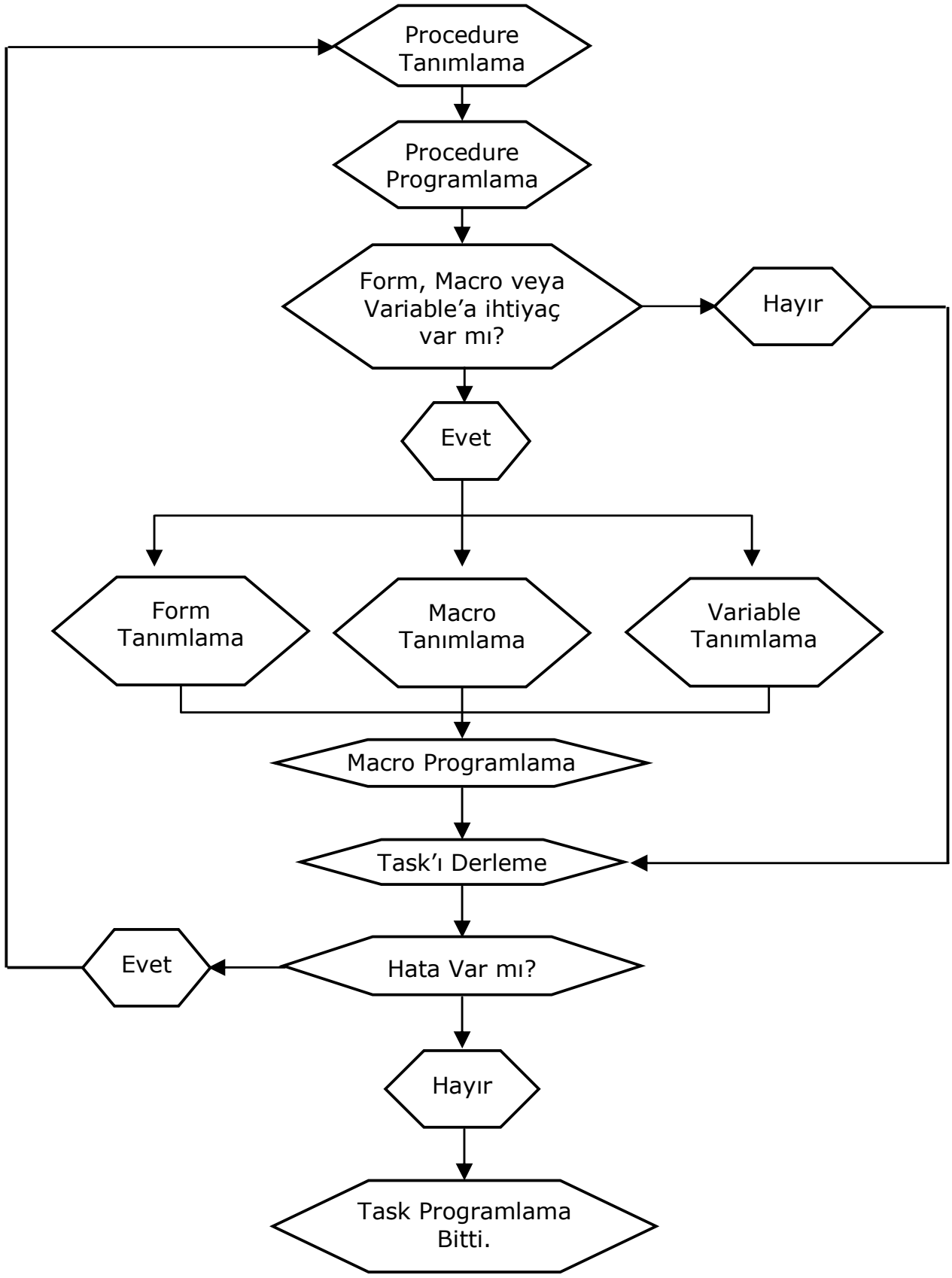


Task:
Wintaskgen programı ile yazılan programlara **Task (**Görev**)** denir.
Not:

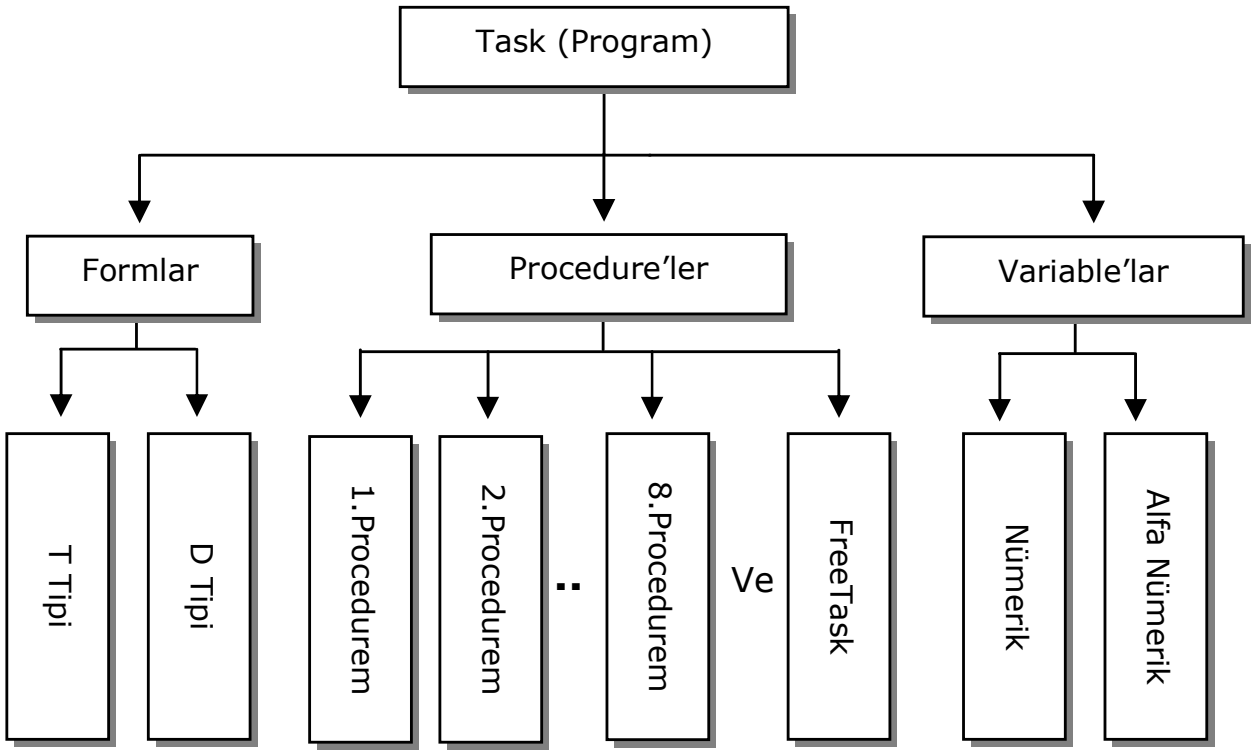
√ Zebex PDL20–16 taşınabilir Data terminaline aynı anda 1 (bir) task yüklenebilir.

√ Zebex PDL20–16 taşınabilir Data terminaline yüklü Freetask adlı silinemeyen bir task bulunmaktadır.

Task Oluşturma Akış Diyagramı



Task (Program) Yapısı



Procedure'lerin Yapısı ve Oluşturulması

Procedure' ler task'ın işleyişin sağlamak için kısıtlı komut kullanılarak hazırlanan genelde Macro'ların çalıştırılmasını sağlayan program bloklarıdır.

Not:

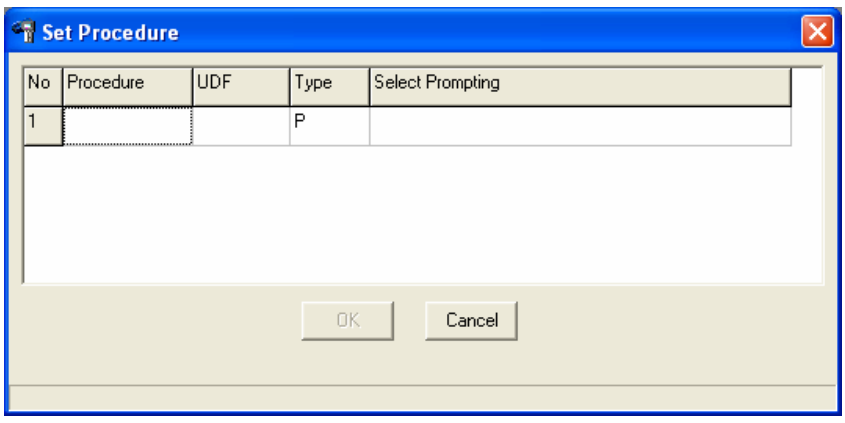
- Bir Task en fazla 8 adet procedure içerebilir.
- Bir Procedure teorik olarak sonsuz macro içerebilmesine rağmen pratikte procedure 99 adet satır içerebildiği için bir procedure en fazla 99 adet macro içerebilir.

Procedure'ler iki yöntem ile oluşturulabilir.

1) Task'ı ilk kez oluştururken oluşturma,
2) Task'ı programlama anında oluşturma,

1) Task'ı ilk kez oluştururken procedure oluşturma:

Task'ı ilk defa oluşturmak için **Task Editor** penceresinde **Create Task** menüsünden **New** seçeneği çalıştırılmalıdır. **New** seçeneği çalıştırıldığında aşağıdaki şekilde de görülen **set procedure** diyalog kutusu ekrana çıkacaktır. Bu bölümde task'ın içereceği procedure'lerin tanımlanması gerekmektedir.



Set Procedure diyalog kutusunda oluşturulacak Task'ın procedure'leri ve özellikleri belirlenir. Bu bölümdeki seçenekler ve özellikleri şunlardır:

Procedure:

Geçerli Task'a (program) bağlı program parçasının adıdır. PDL20–16'ya yüklü **Freetask** adlı silinemeyen bir task bulunmaktadır.

- Bir procedure adı en fazla 8 karakter olabilir.
- Bir procedure adı Türkçe karakter içermemelidir.

Not: Bir task en fazla 8 adet procedure'den oluşabilir. Oluşturulan task ile birlikte Freetask adlı task'da bulunacaksa procedure sayısı en fazla 7 adet olabilir.

UDF:

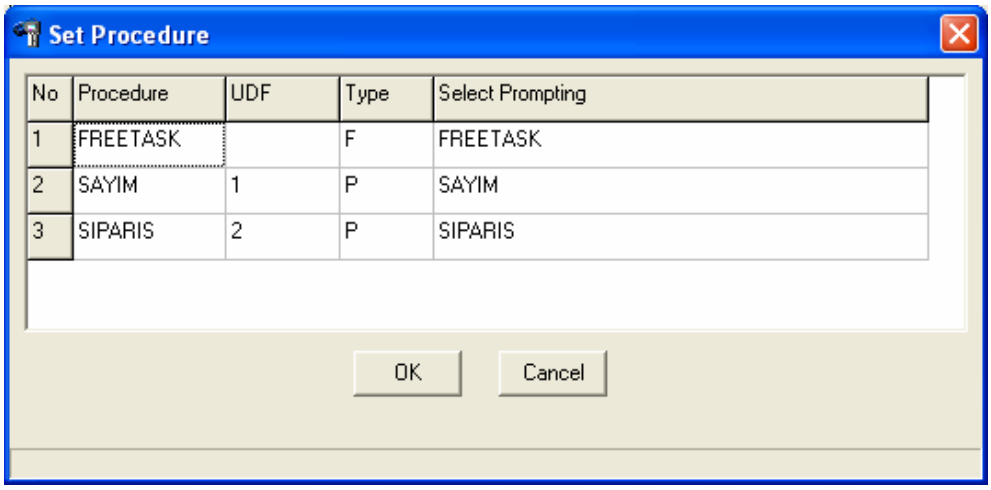
Tanımlı procedure'ün standart olarak kullanacağı formun (Database) sıra numarasıdır.

Type:

Tanımlı procedure'ün tipidir. **P** ve **F** olarak iki seçeneğe sahiptir. Standart olarak **P** kullanılmaktadır. Eğer **Freetask** adlı task'da (silinemeyen task) hazırlanan task ile birlikte yüklenecekse tipi **F** olmalıdır.

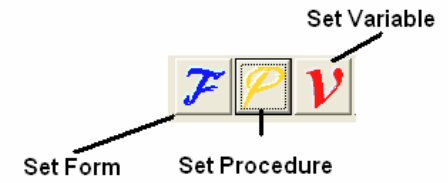
Select Prompting

Belirtilen Procedure'ün task çalıştırıldığında PDL20–16 ekranda görünmesi gereken mesajın belirtilmesini sağlar. PDL20–16 ekranında bir satırda 16 karakter yazılabildiğinden dolayı genelde bu bölümdeki mesajın 16 karakteri geçmemesi gerekmektedir.
Set Procedure diyalog kutusunda gerekli tanımlamaların yapılması ile oluşan procedure tanımlama ekranı aşağıdadır.



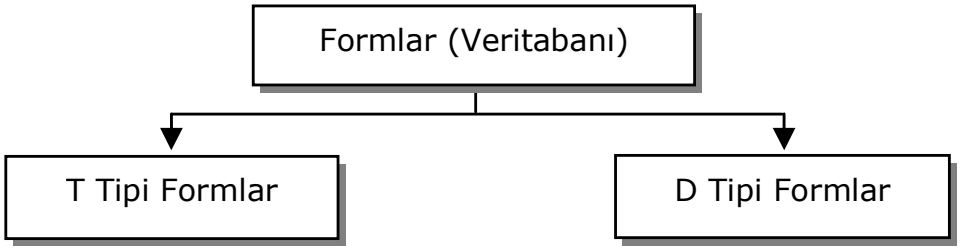
2) Task'ı programlama anında procedure oluşturma:

Task Editor ekranında bulunan ve aşağıdaki resimde de görülen Set Procedure düğmesi tıklanarak açılan ve yukarıdaki resimde de görülen Set procedure diyalog kutusu ile oluşturulabilir.

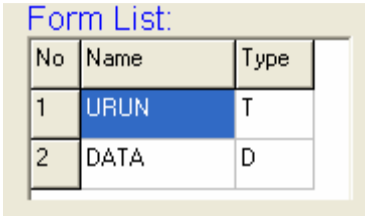


Form'ların yapısı ve oluşturulması

Formlar PDL20–16 içerisinde bilgilerin kayıt edildiği (saklandığı) bölümlerdir. 1 veya 1 den fazla alandan (field) oluşabilir. Bilgisayar programlarındaki Database'in karşılığıdır. 2 tip Form bulunmaktadır.

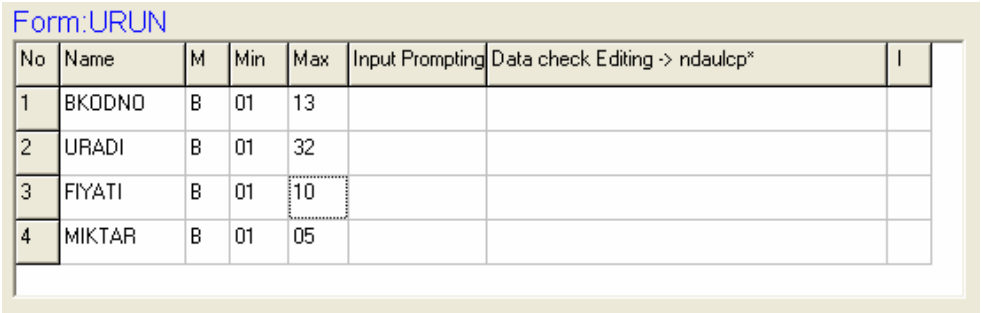


Bir Form oluşturabilmek için **Task Editor** ekranındaki **Set Form** düğmesi tıklanarak açılan ve yandaki şekilde de görülen **Form Editor** diyalog kutusu kullanılmalıdır.



Forma ait alanların tanımlanması

Seçili Form'a ait alanları tanımlayabilmek için Form Editor ekranında Form List bölümünde alanları tanımlanacak form seçili iken aşağıdaki ekranda da görülen alt bölümdeki ekranda alanlar tanımlanabilir.



Not: Bir Task (Program) en fazla 8 adet Form içerebilir.
Name : Forma ait alan adı
M : Bu alan adı program içerisinden kullanıldığında veri giriş şeklinin nasıl yapılacağının belirlenmesi sağlar. Seçenekleri B, K, S dir.
B: Tuş ve lazer tarama tuşu ile aynı anda giriş yapılmasını sağlar.
K: Sadece tuşlarla giriş yapılmasını sağlar.
S: Sadece lazer tarama tuşu ile giriş yapılmasını sağlar.
Min : Formun bu alanına kaydedilebilecek verinin en az karakter sayısını belirlenmesini sağlar. (En az 01 olabilir)
Max : Formun bu alanına kaydedilebilecek verinin en fazla karakter sayısını belirlenmesini sağlar. (En fazla 64 olabilir)
Input Prompt: Programlamada bu alanın kullanılması halinde ekranda çıkacak mesajın belirlenmesini sağlar.

Data check Editing: Programlamada bu alanın kullanılması halinde ekranda gösterilecek örnek verinin belirlenmesini sağlar.
I: Programlamada bu alanın kullanılması halinde ekranda gösterilecek örnek verinin Forma kayıt edilip edilmemesini sağlar. Seçenekler **Y**, **N** 'dir.
Y:Örnek verinin forma kayıt edilmesini sağlar.
N:Örnek verinin forma kayıt edilmemesini sağlar.

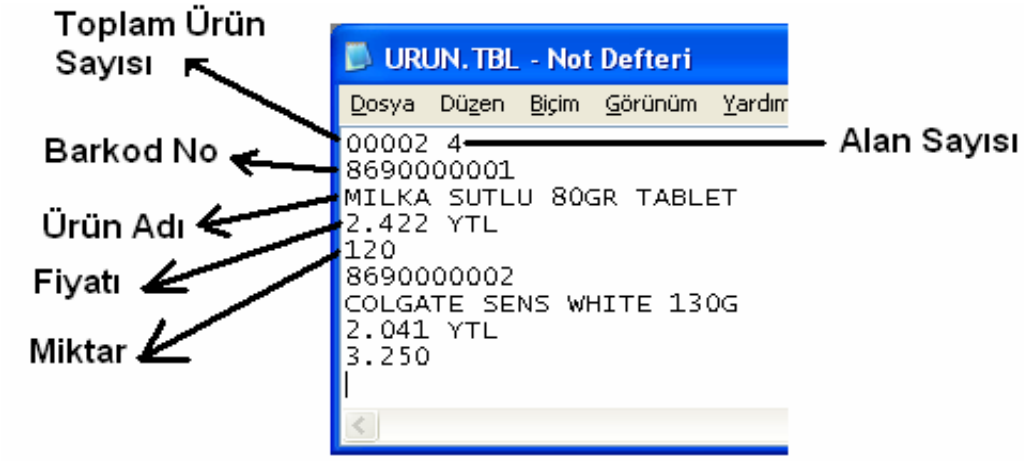
T (Table) Tipi Form:

Bu tip formlardaki veriler bilgisayardan task'ın (program) PDL20-16'ya yüklenmesi ile birlikte PDL20-16'ya kaydedilen ve sadece okunabilir özelliğe sahip verilerdir. Program içerisinden bu verileri değiştirme ve silme özelliği yoktur.

Not:

√ Bilgisayardan PDL20-16'ya verilerin aktarılabilmesi için Task'daki (program) form tipi **T** olmalıdır.
√ Programın kayıtlı olduğu klasörde form adı ile aynı isme sahip, uzantısı **TBL** olan bir dosya bulunmalıdır.
√ **TBL** dosyası içeriği her alan bir satır oluşturacak şekilde olmalıdır.

Not: Yukarıdaki (Form List) şekle göre 2 adet Form tanımlanmıştır. **URUN** adlı form **T** tipinde belirtildiğinden dolayı Task'ın bulunduğu konumda **URUN.TBL** adlı bir dosya içerisinde bilgisayara aktarılacak bilgiler her alan bir satır oluşturacak şekilde oluşturulmalıdır. **URUN.TBL** dosyasının örneği aşağıdadır.



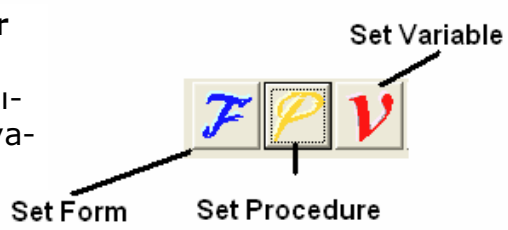
Yukarıdaki şekilde görülen **URUN.TBL** dosyasının ilk satırında bulunan **00002** toplam kaç üründen oluştuğu, **4** rakamı her ürünün kaç alandan oluştuğu anlamına gelir. 2–5 nci satırlardaki bilgiler ilk ürünün bilgileridir.
Not: TBL uzantılı dosyadaki bilgiler, form tanımlamasındaki sıra ile olmalıdır.

D (Data) Tipi Form:

Bu tip formlar programın çalışma durumunda yapılan veri girişlerinin saklanabilmesi, değiştirilebilmesi ve istenildiğinde okunabilmesini sağlayan özelliğe sahip formlardır.

Variable'ların yapısı ve oluşturulması

Variable'lar PDL20-16'ya yüklenilen Task'ın (programın) çalışma anında duruma bağlı olarak farklı değerler alarak (yüklenilerek) programın işleyişini sağlayan hafıza bölümleridir. **2 (iki)** tip **Variable** bulunmaktadır.



Bir Variable oluşturabilmek için **Task Editor** ekranındaki yandaki şekilde de görülen **Set Variable** düğmesi tıklanarak açılan ve aşağıdaki şekilde de görülen **Variable Editor** diyalog kutusu kullanılmalıdır.

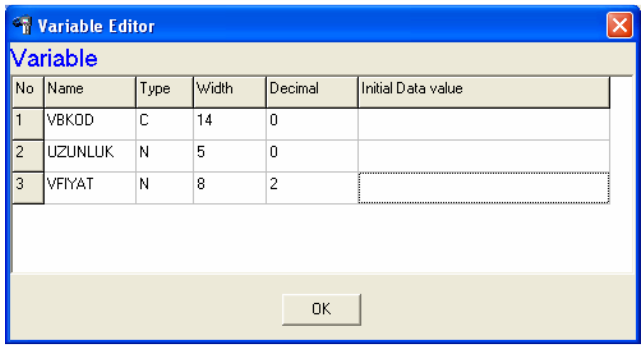
Name : Task içerisinde kullanılabilecek değişkenini adıdır.

√ Türkçe karakter içermemelidir.
√ En Fazla 8 karakter olmalıdır.

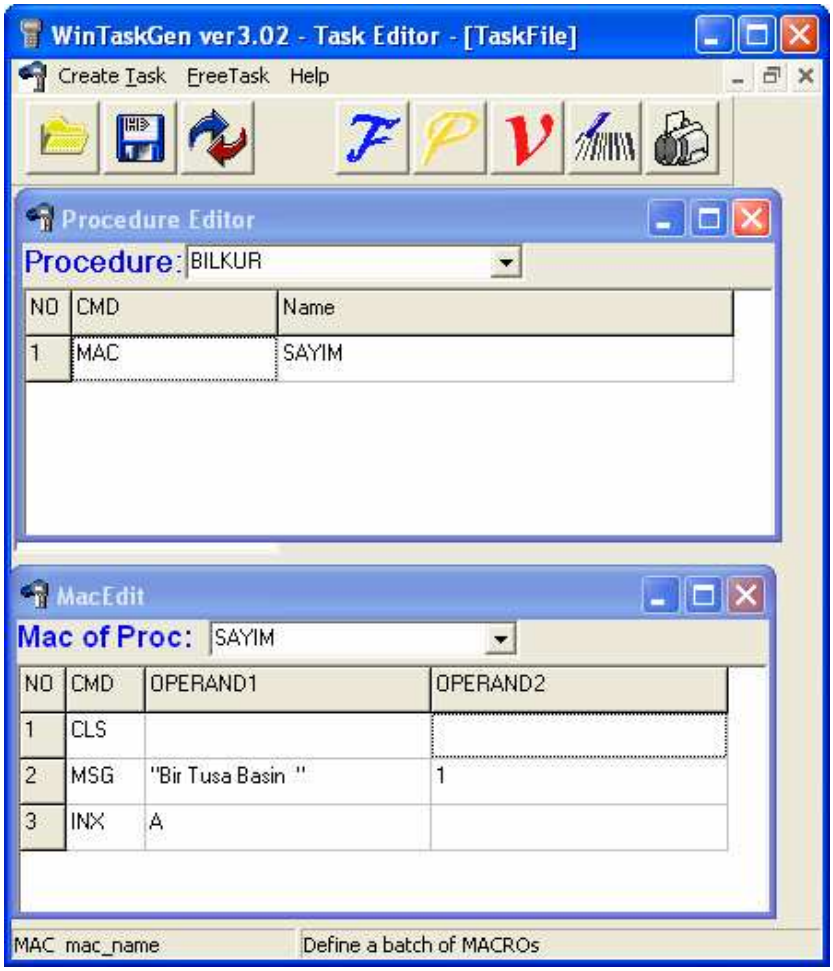
Type : Task içerisinde kullanılabilecek değişkenin tipidir.

C : Harf veya sayı içerebilen, aritmetik işlemlere tabi tutulamayan değişkenlerdir.

N : Sadece sayısal veri içerebilen, aritmetik işlemlere tabi tutulabilen değişkenlerdir.



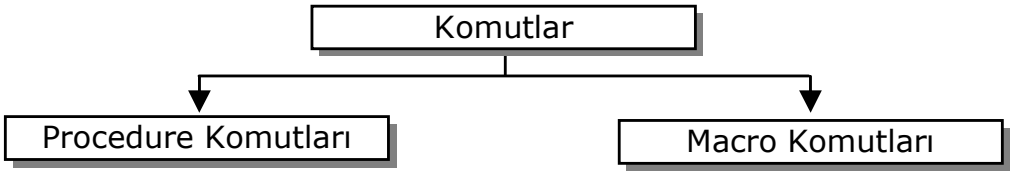
Width : Değişkenin en fazla içerebileceği karakter sayısıdır.
Decimal : Sayısal değişkenin ondalık hane sayısıdır.
İntial Data Value: Değişkenin alacağı ilk değerdir.
Not: RECORDP (Aktif kayıt numarasını gösteren) ve * (yıldız) adlı iki değişken tanımlanmadan kullanılabilmektedir.
Not: Procedure, Form ve Variable tanımlama işlemleri tamamlandıktan sonra Task Editor ekranı aşağıdaki gibidir.



Program Yazımı

Bir Task'ın (Programın) işleyişini sağlamak için kendine has kuralları olan **Komutlar** ve **etiketler** kullanılabilir.

Komut : Programın işleyişini sağlamak için kullanılan, farklı görevleri bulunan ve **3'**er harften oluşan özel kelimelerdir.



Procedure komutlarının Kullanım Şekli:

CMD	Name
Komut	**Seçenek–1**

Örnek:

CMD	Name
MAC	**Bilkur**

Macro komutlarının Kullanım Şekli:

CMD	OPERAND1	OPERAND2
Komut	**Seçenek–1**	**Seçenek–2**

Örnek:

CMD	OPERAND1	OPERAND2
MSG	**"Bilkur"**	**1**

Not: Bazı komutlar sadece seçenek-1'i, bazı komutlar her iki seçeneği kullanırken bazı komutlar seçenek kullanmaz.

Etiket : Programın işleyişini sağlamak için duruma bağlı olarak programın akışını yönlendirildiği konumlara verilen isimlerdir.

Kullanım Şekli:

CMD	OPERAND1	OPERAND2
:ETIKET-ADI		

Örnek:

CMD	OPERAND1	OPERAND2
…	…	…
JMP	EQ	**:HATA**
…	…	…
:HATA		

Procedure Programlama

Bir procedure; programın işleyişinden çok macro program bloklarının çalıştırılmasını sağlamak için tasarlanır. Bazı durumlarda procedure'ün programlanması ile PDL20-16'dan istenen ihtiyacı karşılayabilir. Procedure'lerde kullanılabilecek komut sayısı **6 (Altı)** adettir.

Procedure Komutları

MAC

Kullanılabilen Bölüm : Procedure Editor
İşlevi : Procedure bağlı bir macro'nun çalıştırılmasını sağlar.
Kullanım Şekli

CMD	Name
MAC	**Macro Adı**

Örnek

CMD	Name
MAC	**SAYIM**

Açıklama

SAYIM Adlı macro'nun çalıştırılmasını (işletilmesini) sağlar.

Not:

1) Bir procedure teorik olarak sonsuz sayıda Macro içerebilir.
2) Procedure Editor ekranında en fazla 99 satır yazma imkânı vardır.
3) Bir procedure pratikte 99 adet Macro içerebilir.

LST

Kullanılabilen Bölüm : Procedure Editor
İşlevi : Döngüsel bir işlemin şartsız olarak başlatılmasını sağlar.
Not : **LOP** komutu ile birlikte kullanılır.
Kullanım Şekli

CMD	Name
LST	

Örnek

CMD	Name
LST	
MAC	SAYIM
LOP	

Açıklama

SAYIM Adlı macro'nun birden fazla çalıştırılabilmesi için döngüyü başlatır.

LOP

Kullanılabilen Bölüm : Procedure Editor
İşlevi : Döngüsel bir işlemin şartsız olarak bitirilmesini sağlar.
Not : **LST** komutu ile birlikte kullanılır.

Kullanım Şekli

CMD	Name
LOP	

Örnek

CMD	Name
LST	
MAC	SAYIM
LOP	

Açıklama

SAYIM Adlı macro'nun birden fazla çalıştırılabilmesi için döngüyü başlatır.

IPF

Kullanılabilen Bölüm : Procedure Editor
İşlevi : Klavye veya Lazer tarama tuşu ile okunan verinin formun belirtilen alanına aktarılmasını sağlar.

Not
- Belirtilen alanın formun bir alanı olması gerekmektedir.
- Veri girişi için çıkması gereken mesaj form tanımlama bölümünde **Input promting** bölümünde belirtilmelidir.
- Forma kayıt için NRC komutunun kullanılması gerekmektedir.

Kullanım Şekli

CMD	Name
IPF	**Alan_Adı**

Örnek

CMD	Name
IPF	**BKOD**
NRC	

Açıklama

Klavyeden veya lazer tarama tuşu ile girilen verinin BKOD adlı Form'a ait alana aktarılmasını sağlar.
Not: IPF komutu ile alınan verinin forma kaydedilmesi için **NRC** komutunun kullanılması gerekir.

SBF

Kullanılabilen Bölüm : Procedure Editor
İşlevi : Hafızadaki veriyi form alanına kaydedilmesini sağlar.

Not
- Belirtilen alanın formun bir alanı olması gerekmektedir.
- Forma kayıt için NRC komutunun kullanılması gerekmektedir.

Kullanım Şekli

CMD	Name
SBF	**Alan_Adı**

Örnek

CMD	Name
...	**...**
SBF	**BKOD**
NRC	

Açıklama

Genel değişken olan ***** (yıldız) değişkenini içeriğini belirtilen alana aktarılmasını sağlar.
Not: SBF komutu ile alınan verinin forma kaydedilmesi için **NRC** komutunun kullanılması gerekir.

NRC

Kullanılabilen Bölüm : Procedure ve Macro Editor
İşlevi : Hafızadaki verinin forma kaydedilmesini sağlar.

Not

- Hafızada verinin form alanlarına aktarılması gerekmektedir.
- Forma kayıt için **NRC** komutunun kullanılması gerekmektedir.

Kullanım Şekli

CMD	Name
NRC	

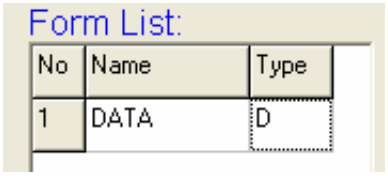
Örnek

CMD	Name
...	...
...	...
NRC	

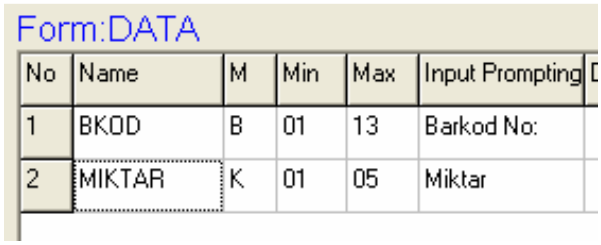
Açıklama

Hafıza değişkenlerinde bulunan veriyi ilgili formun sonuna kayıt edilmesini sağlar.

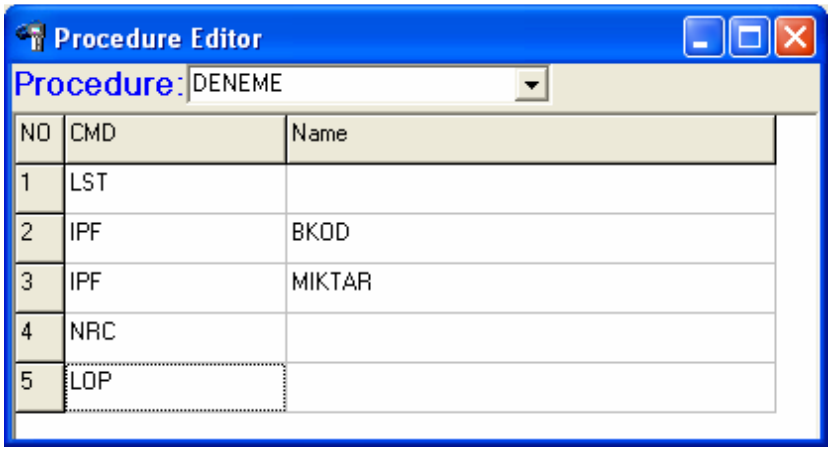
ÖRNEK PROGRAM



Şekil–1



Şekil–2



Şekil–3

Açıklama

Yukarıdaki 3 şekil ile procedure programlama yöntemi kullanılarak, Barkod ve miktar bilgisinin DATA formuna kaydedilmesini sağlanmıştır.

- Şekil-1'de DATA adlı D tipinde bir Form tanımlanmıştır.
- Şekil-2'de ise DATA formunun alanları belirlenmiştir.
- DATA formunun BKOD alanına klavye ve lazer tarama tuşu ile en az 1, en fazla 13 karakter girilebileceği ve BKOD alanı procedure editor'de kullanılması durumunda ekranda görünmesi gereken mesaj "Barkod No" olarak Belirlenmiştir.

- MIKTAR adlı alan ise sadece klavyeden giriş yapılacak şekilde en az 1, en fazla 5 karakter girilebileceği ve MIKTAR alanı procedure editor'de kullanılması durumunda ekranda görünmesi gereken mesaj "Miktar" olarak belirlenmiştir.

Macro Programlama

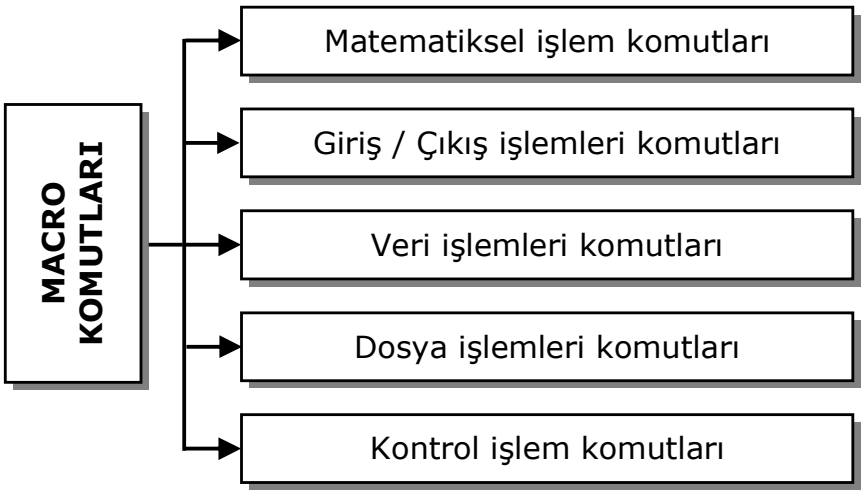
Macro'lar Task'ın (programın) işleyişini sağlamak için macro komutları kullanılarak hazırlanan program bloklarıdır.

Not:

- Kullanılabilir komut sayısı 35 adettir.
- Bu komutlardan **NRC** adlı komut procedure ve macro programlamada ortaktır.
- Toplam kullanılabilir satır sayısı 99 adettir.

Macro Komutları

Macro komutları işlevlerine göre 5 gruba ayrılmaktadır.



Matematiksel İşlem Komutları

Bu gruptaki komutlar sayısal veriler üzerinde toplama, çıkarma, bölme ve çarpma gibi aritmetik işlemlerin yapılmasını sağlayan komutlardır. Toplam 4 adet komuttan oluşmaktadır. Bu komutlar ve kullanım şekli şunlardır:

1) **ADD** : Sayısal değerlerin toplanmasını, sayısal olmayan değerlerin eklenmesini sağlar.
2) **SUB** : Sayısal değerlerin birbirlerinden çıkartılmasını sağlar.
3) **DIV** : Sayısal değerlerin birbirlerine bölümünü sağlar.
4) **MUL** : Sayısal değerlerin birbirleri ile çarpılmasını sağlar.

ADD

Kullanılabilen Bölüm : Macro Editor
İşlevi : Sayısal değişkenlerin birbirleri toplanmasını, sayısal olmayan verinin birbirlerine eklenmesini sağlar.

Not

- Sonuç operand2 bölümündeki değişken-2'ye aktarılır.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
ADD	**DEĞİŞKEN–1**	**DEĞİŞKEN–2**

Örnek

CMD	OPERAND1	OPERAND2
ADD	**SAYI1**	**SAYI2**

Açıklama

— Yukarıdaki örnekte SAYI1 ve SAYI2 sayısal değerler içermesi durumunda, SAYI1 değişkeninin içeriği ile SAYI2 değişkeninin içeriği toplanarak SAYI2 değişkenine aktarılır.

— Yukarıdaki örnekte SAYI1 ve SAYI2 sayısal olmayan değerler içermesi durumunda, SAYI1 değişkeninin içeriği ile SAYI2 değişkeninin içeriği birlerine eklenerek SAYI2 değişkenine aktarılır.

SUB

Kullanılabilen Bölüm : Macro Editor
İşlevi : Sayısal verilerin birbirlerinden çıkartılmasını sağlar.

Not

- Sonuç operand2 bölümündeki değişken-2'ye aktarılır.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
SUB	**DEĞİŞKEN–1**	**DEĞİŞKEN–2**

Örnek

CMD	OPERAND1	OPERAND2
SUB	**SAYI1**	**SAYI2**

Açıklama

— Yukarıdaki örnekte SAYI2 sayısal değerinden SAYI1 değeri çıkartılarak (eksiltilerek) sonucu SAYI2 değişkenine aktarılır.

DIV

Kullanılabilen Bölüm : Macro Editor
İşlevi : Sayısal verilerin birbirlerine bölünmesini sağlar.

Not

- Sonuç operand2 bölümündeki değişken-2'ye aktarılır.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
DIV	**DEĞİŞKEN–1**	**DEĞİŞKEN–2**

Örnek

CMD	OPERAND1	OPERAND2
DIV	**SAYI1**	**SAYI2**

Açıklama

— Yukarıdaki örnekte SAYI2 sayısal değeri SAYI1 değerine bölünerek sonucu SAYI2 değişkenine aktarılır.

MUL

Kullanılabilen Bölüm : Macro Editor
İşlevi : Sayısal verilerin birbirleriyle çarpılmasını sağlar.

Not
- Sonuç operand2 bölümündeki değişken-2'ye aktarılır.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
MUL	**DEĞİŞKEN–1**	**DEĞİŞKEN–2**

Örnek

CMD	OPERAND1	OPERAND2
MUL	**SAYI1**	**SAYI2**

Açıklama

— Yukarıdaki örnekte SAYI2 sayısal değeri SAYI1 değeri ile çarpılır ve sonucu SAYI2 değişkenine aktarılır.

Giriş-Çıkış İşlem Komutları

Bu gruptaki komutlar, dışarıdan PDL20'ye veri alınmasını, PDL20'den dışarıya veri gönderilmesi, ekranda mesaj gösterme, mesaj silme, ses ile uyarı verme ve gibi komutları içermektedir. Toplam 8 adet komuttan oluşmaktadır. Bu komutlar ve kullanım şekli şunlardır:

1) **BEP** : PDL20'den sesli uyarı vermesini sağlar.
2) **CLL** : Ekranda belirtilen satır görüntüsünün silinmesini sağlar.
3) **CLS** : Ekrandaki tüm görüntünün silinmesini sağlar.
4) **INP** : klavye veya Lazer tarama tuşu ile veri girişi yapılmasını sağlar.
5) **INX** : klavyeden tek bir veri girişi yapılmasını sağlar.
6) **MSG** : Ekranda herhangi bir mesajın gösterilmesini sağlar.
7) **RCV** : Seri porttan veri alınmasını sağlar.
8) **SND** : Seri porta veri gönderilmesini sağlar.

BEP

Kullanılabilen Bölüm : Macro Editor
İşlevi : Belirtilen süre ve frekansda hoparlörden ses çıkartılmasını sağlar.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
BEP	**DEĞİŞKEN–1**	**DEĞİŞKEN–2**

DEĞİŞKEN–1: Sesin çıkartılacağı süredir.1–32.767 arası bir değerdir. (1 saniye=1000)
DEĞİŞKEN–2: Sesin çıkartılacağı frekanstır.100–3000 Hertz arası bir değerdir.

Örnek

CMD	OPERAND1	OPERAND2
BEP	**1000**	**600**

Açıklama

— Yukarıdaki örnekte PDL20 hoparlöründen 1 saniye süre ile (1 saniye=1000 olduğundan) 600 Hertz frekansında bir sesin çıkması sağlanır.

CLL

Kullanılabilen Bölüm : Macro Editor
İşlevi : Ekranın belirtilen satırındaki görüntünün silinmesini (temizlenmesi) sağlar.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
CLL	**DEĞİŞKEN–1**	

DEĞİŞKEN–1: Silinecek (temizlenecek) satırın numarasıdır.

- Satır numarası 1-6 arası bir değerdir.

Örnek

CMD	OPERAND1	OPERAND2
CLL	**6**	

Açıklama

— Yukarıdaki örnekte PDL20 ekranında 6 ncı satırdaki görüntünün silinmesi sağlanır.

CLS

Kullanılabilen Bölüm : Macro Editor
İşlevi : Ekranın tüm satırlarındaki görüntünün silinmesini (temizlenmesi) sağlar.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
CLS		

Örnek

CMD	OPERAND1	OPERAND2
CLS		

Açıklama

— Yukarıdaki örnekte PDL20 ekranın tüm görüntünün silinmesi sağlanır.

INP

Kullanılabilen Bölüm : Macro Editor
İşlevi : PDL20'nin klavye tuşlarından veya lazer tarama tuşundan herhangi bir verinin alınmasını ve belirtilen değişkene aktarılmasını sağlar.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
INP	**DEĞİŞKEN–1**	

Not:

— DEĞİŞKEN–1'in tipi ve boyutu önceden tanımlanmış olmalıdır.
— DEĞİŞKEN–1 bir Form alanı ise giriş şekli formdaki tanımlamaya bağlı olarak kullanılabilir. (Formda tip olarak **K** belirtilmişse bu değişkene sadece klavyeden giriş yapılabilir, **S** olarak belirtilmişse Sadece lazer tarama tuşundan giriş yapılabilir veya **B** olarak belirtilmişse hem klavyeden hem lazer tarama tuşundan giriş yapılabilir.

Örnek

CMD	OPERAND1	OPERAND2
INP	**BKODNO**	

Açıklama

— Yukarıdaki örnekte klavyeden veya lazer tarama tuşu ile girilen veri BKODNO adlı değişkene aktarımı sağlanır.

INX

Kullanılabilen Bölüm : Macro Editor
İşlevi : PDL20'nin klavye tuşlarından girilen bir karakterlik verinin belirtilen değişkene aktarılması sağlar.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
INX	**DEĞİŞKEN–1**	

Not:

— DEĞİŞKEN–1'in önceden tanımlanmış olması gereklidir.
— ***** (yıldız) değişkeni tanımlanmadan kullanılabilen bir değişkendir.
— Genelde tuş bekletme durumlarında kullanılır.
— INX komutu kullanıldığında ekranda kürsor (imleç) görünmez.

Örnek

CMD	OPERAND1	OPERAND2
INX	**TUS**	

Açıklama

— Yukarıdaki örnekte klavyeden girilen bir karakterlik veri **TUS** adlı değişkene aktarımı sağlanır.

MSG

Kullanılabilen Bölüm : Macro Editor
İşlevi : PDL20 ekranının belirtilen bir konumuna belirtilen bir mesajın yazılmasını (gösterilmesini) sağlar.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
MSG	**DEĞİŞKEN–1**	**DEĞİŞKEN–2**

Not:

— DEĞİŞKEN–1 yazdırılacak mesajdır.
— Mesaj **doğrudan** yazdırılabilir veya bir değişkene aktarılarak yazdırılabilir.
— Mesaj doğrudan yazdırılacak ise DEĞİŞKEN–1 en fazla **16** karakter ve **" "** (çift tırnak) içerisinde olmalıdır.
— Mesaj **değişken** yardımı ile yazdırılacak ise DEĞİŞKEN–1 en fazla **32** karakter olabilir.
— DEĞİŞKEN–2 mesajın ekranda yazdırılacağı koordinattır.
— PDL20 ekranı 6 satır ve her satırda 16 karakterlik bir mesaj yazdırabilme özelliğe sahiptir.
— 1–16 arası koordinatlar 1 nci satır, 17–32 arası koordinatlar 2 nci satır, 33–48 arası koordinatlar 3 ncü satır, 49–64 arası koordinatlar 4 ncü satır, 65–80 arası koordinatlar 5 nci satır ve 81–96 arası koordinatlar 6 ncı satır aralığındadır.

Örnek

CMD	OPERAND1	OPERAND2
CLS		
MSG	**"BILKUR"**	**5**

Açıklama

— Yukarıdaki örnekte PDL20 ekranı **CLS** komutu ile tamamen temizlendikten sonra **MSG** komutu ile 5 nci koordinata (1 nci satırda soldan 5 nci koordinat) BILKUR mesajının yazılması sağlanır.

Örnek

CMD	OPERAND1	OPERAND2
CLS		
MSG	**URAD**	**81**

Açıklama

— Yukarıdaki örnekte PDL20 ekranı **CLS** komutu ile tamamen temizlendikten sonra **MSG** komutu ile 81 nci koordinata (6 ncı satırda soldan 1 nci koordinat) URAD değişkenin içeriğinin yazılması sağlanır.

RCV

Kullanılabilen Bölüm : Macro Editor
İşlevi : PDL20'nin bağlı olduğu haberleşme portundan (seri porttan) belirtilen süre içerisinde okuduğu veriyi belirtilen değişkene aktarılmasının sağlar.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
RCV	**DEĞİŞKEN–1**	**DEĞİŞKEN–2**

— DEĞİŞKEN–1 haberleşme portundan verinin alınacağı süredir. Süre 1–32.767 arası bir değerdir. (1 saniye=1000)
— DEĞİŞKEN–2 haberleşme portundan alınan verinin aktarılacağı değişkendir.

Örnek

CMD	OPERAND1	OPERAND2
RCV	**1000**	**URKOD**

Açıklama

— Yukarıdaki örnekte PDL20'nin bağlı olduğu haberleşme portundan 1 saniye içerisine gönderilen herhangi bir veri URKOD değişkeninin tanımlanma durumuna bağlı olarak URKOD değişkenine aktarılır.
— Örneğin URKOD 10 hane olarak tanımlanmış kabul edilirse, haberleşme portundan 10 karakterden fazla bir veri gelmesi durumunda sadece ilk 10 karakter URKOD değişkenine aktarılır.

SND

Kullanılabilen Bölüm : Macro Editor
İşlevi : PDL20'nin bağlı olduğu haberleşme portuna (seri porta) belirtilen değişken içeriğinin gönderilmesini sağlar.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
SND	**DEĞİŞKEN–1**	**DEĞİŞKEN–2**

— DEĞİŞKEN–1 haberleşme portuna gönderilecek veri değişkenidir.
— DEĞİŞKEN–2 haberleşme portuna gönderilen veriden sonraki mesaj sonlandırma tipidir.

Mesaj Sonlandırma tipleri

0 : Bir sonraki mesajın aynı satırda ve satır başı yapılarak gönderilmesini sağlamak için verinin sonuna **CR** ekler.
1 : Bir sonraki mesajın bir alt satıra ve aynı konumda yazdırılabilmesi için gönderilen verinin sonuna **LF** ekler.
2 : Bir sonraki mesajın bir alt satırda ve satır başı yapılarak yazdırılabilmesini sağlamak için verinin sonuna **CR /LF** ekler.
3 : Bir sonraki mesajın aynı satırda ve bir önceki mesajın devamına yazdırılmasını sağlar.

Örnek

CMD	OPERAND1	OPERAND2
SND	**SIRANO**	**3**
SND	**“ “**	**3**
SND	**URKOD**	**2**

Açıklama

— Yukarıdaki örnekte PDL20’nin bağlı olduğu haberleşme portuna **SIRANO** değişkeni içeriği, aynı satırda ve **SIRANO** verisinin devamına **“ ” (boşluk)** içerisindeki miktar kadar boşluk ve aynı satırda **“ ” (boşluk)** dan sonra **URKOD** değişkeninin yazdırılmasını ve ardından bir alt satıra ve satır başına geçecek şekilde gönderilmesini sağlar.
— Bu yöntemi kullanarak seri port bağlantılı herhangi bir yazıcıya yazdırma sağlanabilir.

Veri İşlemleri Komutları

Bu gruptaki komutlar, verilerin karşılaştırılmasını, farklı veri tiplerine dönüştürülmesini, farkı değişkenlere aktarma, belirli bir kısmının alınması, karakter uzunluğunun hesaplanması gibi işlemlerin yapılabilmesini sağlayan komutları içerir. Toplam 10 adet komuttan oluşmaktadır. Bu komutlar ve kullanım şekli şunlardır:

1) **CDV :** MSI tipi barkodlarda kontrol karakterinin (check dijit) yapılmasını sağlar.
2) **CKD :** Değişkenin Tarih tipinde bir değişken olup-olmadığının kontrol edilmesini sağlar.
3) **CKT :** Değişkenin Saat tipinde bir değişken olup-olmadığının kontrol edilmesini sağlar.
4) **CMP :** İki değişkenin benzer olup-olmadığının kontrol edilmesini sağlar.
5) **LEN :** Değişken içeriğinin karakter sayısını hesaplanmasını sağlar.
6) **LFT :** Değişkenin soldan belirtilen sayı kadar karakterinin belirtilen bir değişkene aktarılmasını sağlar.
7) **RHT :** Değişkenin sağdan belirtilen sayı kadar karakterinin belirtilen bir değişkene aktarılmasını sağlar.
8) **MOV :** Değişken içeriğinin başka bir değişkene aktarılmasını sağlar.
9) **STR :** Sayısal bir değişkenin alfa sayısal tipe çevrilmesini sağlar.
10) **VAL :** alfa sayısal bir değişkenin sayısal tipe çevrilmesini sağlar.

CDV

Kullanılabilen Bölüm : Macro Editor
İşlevi : MSI tipi barkodlarda MOD 10 veya MOD 11'e göre kontrol karakterinin (Check dijit) kontrol edilmesini sağlar. Eğer kontrol karakteri var ise **EQ** kontrol değişkeni, kontrol karakteri yok ise **NE** kontrol değişkeni doğru değerini içerir.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
CDV	**DEĞİŞKEN–1**	

— DEĞİŞKEN–1 Herhangi bir değer içeren değişkendir.

Örnek

CMD	OPERAND1	OPERAND2
INP	**ABC**	
CDV	**ABC**	
JMP	**EQ**	**15**
JMP	**NE**	**20**

Açıklama

— Yukarıdaki örnekte ABC barkod değişkeninin MSI tipinde olması durumunda kontrol karakterinin MOD 10/11 olup olmadığı kontrol edilmekte ve ABC değişkeninin MOD 10/11 olması durumunda **EQ** kontrol değişkeni doğru değeri, ABC değişkeninin MOD 10/11 olmaması durumunda **NE** kontrol değişkeni doğru değeri alacaktır. Eğer **EQ** kontrol değişkeni doğru değer içeriyor ise programın 15 nci satıra yönlendirilmesini, Eğer **NE** kontrol değişkeni doğru değer içeriyor ise programın 20 nci satıra yönlendirilmesini sağlamaktadır.

CKD

Kullanılabilen Bölüm : Macro Editor
İşlevi : Belirtilen değişkenin tarih tipinde bir değer olup-olmadığının kontrol edilmesini sağlar. Eğer tarih tipinde bir değer ise **EQ** kontrol değişkeni, tarih tipinde bir değer değil ise **NE** kontrol değişkeni doğru değerini içerir.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
CKD	**DEĞİŞKEN–1**	

— DEĞİŞKEN–1 Herhangi bir değer içeren değişkendir.

Örnek

CMD	OPERAND1	OPERAND2
CKD	**DEF**	
JMP	**EQ**	**5**
JMP	**NE**	**10**

Açıklama

— Yukarıdaki örnekte DEF değişkeninin içeriğinin tarih tipinde olup olmadığı kontrol edilmekte ve DEF değişkeninin içeriğinin tarih tipinde olması durumunda **EQ** kontrol değişkeni doğru değeri, DEF değişkeninin içeriğinin tarih tipinde olmaması durumunda **NE** kontrol değişkeni doğru değeri alacaktır. Eğer **EQ** kontrol değişkeni doğru değer içeriyor ise programın 5 nci satıra yönlendirilmesini, Eğer **NE** kontrol değişkeni doğru değer içeriyor ise programın 10 nci satıra yönlendirilmesini sağlamaktadır.

CKT

Kullanılabilen Bölüm : Macro Editor
İşlevi : Belirtilen değişkenin saat tipinde bir değer olup-olmadığının kontrol edilmesini sağlar. Eğer saat tipinde bir değer ise **EQ** kontrol değişkeni, saat tipinde bir değer değil ise **NE** kontrol değişkeni doğru değerini içerir.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
CKT	**DEĞİŞKEN–1**	

— DEĞİŞKEN–1 Herhangi bir değer içeren değişkendir.

Örnek

CMD	OPERAND1	OPERAND2
CKT	**ZXC**	
JMP	**EQ**	**3**
JMP	**NE**	**6**

Açıklama

— Yukarıdaki örnekte ZXC değişkeninin içeriğinin saat tipinde olup olmadığı kontrol edilmekte ve ZXC değişkeninin içeriğinin saat tipinde olması durumunda **EQ** kontrol değişkeni doğru değeri, ZXC değişkeninin içeriğinin saat tipinde olmaması durumunda **NE** kontrol değişkeni doğru değeri alacaktır. Eğer **EQ** kontrol değişkeni doğru değer içeriyor ise programın 3 nci satıra yönlendirilmesini, Eğer **NE** kontrol değişkeni doğru değer içeriyor ise programın 6 ncı satıra yönlendirilmesini sağlamaktadır.

CMP

Kullanılabilen Bölüm : Macro Editor
İşlevi : İki değişkenin birbirleri ile karşılaştırılmasını sağlar.
İki değişkenin karşılaştırma durumuna bağlı olarak **EQ, NE, GT, LT** kontrol değişkenlerinden sadece biri doğru değeri almaktadır.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
CMP	**DEĞİŞKEN–1**	**DEĞİŞKEN–1**

— Eğer Değişken–1 Değişken-2'ye eşit ise **EQ** kontrol değişkeni içeriği doğru değerini içerir.
— Eğer Değişken–1 Değişken-2'ye eşit değilse **NE** kontrol değişkeni içeriği doğru değerini içerir.
— Eğer Değişken–1 Değişken–2' den büyük ise **GT** kontrol değişkeni içeriği doğru değerini içerir.
— Eğer Değişken–1 Değişken–2' den küçük ise **LT** kontrol değişkeni içeriği doğru değerini içerir.

Örnek

CMD	OPERAND1	OPERAND2
CMP	**AKODU**	**BKODU**
JMP	**EQ**	**10**
JMP	**NE**	**20**
JMP	**GT**	**30**
JMP	**LT**	**40**

Açıklama

— Yukarıdaki örnekte AKODU değişkeni ile BKODU değişkeni içerikleri karşılaştırılmaktadır. Bu karşılaştırmanın sonucuna bağlı olarak;
— Eğer AKODU değişkeninin içeriği BKODU değişkeninin içeriği ile aynı ise program 10 ncu satıra yönlendirilmiştir.
— Eğer AKODU değişkeninin içeriği BKODU değişkeninin içeriğinden farklı ise program 20 ncu satıra yönlendirilmiştir.
— Eğer AKODU değişkeninin içeriği BKODU değişkeninin içeriğinden büyükse program 30 ncu satıra yönlendirilmiştir. (**Not:** Değişkenler sayısal içerikli olmalıdır)
— Eğer AKODU değişkeninin içeriği BKODU değişkeninin içeriğinden küçükse program 40 ncu satıra yönlendirilmiştir. (**Not:** Değişkenler sayısal içerikli olmalıdır)

LEN

Kullanılabilen Bölüm : Macro Editor
İşlevi : Belirtilen bir değişkenin karakter uzunluğunun hesaplanmasını sağlar.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
LEN	**DEĞİŞKEN–1**	**DEĞİŞKEN–2**

— DEĞİŞKEN–2 sayısal tipte bir değişken olmalıdır.

Örnek

CMD	OPERAND1	OPERAND2
LEN	**URKOD**	**UZ**
CMP	**UZ**	**0**
JMP	**EQ**	**1**

Açıklama

— Yukarıdaki örnekte;
1. satırda URKOD değişkeninin içeriğinin karakter sayısı hesaplanarak UZ değişkenine aktarılmaktadır.
2. satırda UZ değişkeninin içeriği 0 (sıfır) ile karşılaştırılmaktadır.
3. UZ değişkenin 0 (sıfır) olması durumunda programın çalışması 1 nci satıra yönlendirilmektedir.

LFT

Kullanılabilen Bölüm : Macro Editor
İşlevi : Bir değişkenin soldan itibaren belirtilen sayıda karakterini aynı değişkene aktarılmasını sağlar.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
LFT	**DEĞİŞKEN–1**	**DEĞİŞKEN–2**

— DEĞİŞKEN–1 sayısal tipte bir değişken veya değer olmalıdır.

Örnek

CMD	OPERAND1	OPERAND2
LFT	**5**	**URKOD**

CMD	OPERAND1	OPERAND2
LFT	**UZ**	**URKOD**

Açıklama

— Yukarıdaki 1 nci örnekte;
URKOD değişkeni içeriğinin baştan 5 karakteri alınarak yine URKOD değişkenine aktarılmaktadır.
— Yukarıdaki 2 nci örnekte;
URKOD değişkeni içeriğinin baştan UZ sayısal değişkeni değeri kadar karakteri alınarak yine URKOD değişkenine aktarılmaktadır.

RHT

Kullanılabilen Bölüm : Macro Editor
İşlevi : Bir değişkenin sağdan itibaren belirtilen sayıda karakterini aynı değişkene aktarılmasını sağlar.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
RHT	**DEĞİŞKEN–1**	**DEĞİŞKEN–2**

— DEĞİŞKEN–1 sayısal tipte bir değişken veya değer olmalıdır.

Örnek

CMD	OPERAND1	OPERAND2
RHT	**5**	**URKOD**

CMD	OPERAND1	OPERAND2
RHT	**UZ**	**URKOD**

Açıklama

— Yukarıdaki 1 nci örnekte;
URKOD değişkeni içeriğinin sondan 5 karakteri alınarak yine URKOD değişkenine aktarılmaktadır.
— Yukarıdaki 2 nci örnekte;
URKOD değişkeni içeriğinin sondan UZ sayısal değişkeni değeri kadar karakteri alınarak yine URKOD değişkenine aktarılmaktadır.

MOV

Kullanılabilen Bölüm : Macro Editor
İşlevi : Bir değişkenin içeriğini veya bir değeri belirtilen başka bir değişkene aktarılmasını sağlar.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
MOV	**DEĞİŞKEN–1**	**DEĞİŞKEN–2**

— DEĞİŞKEN–1 ve DEĞİŞKEN–2 aynı tipte değişkenler olmalıdır.

Örnek

CMD	OPERAND1	OPERAND2
MOV	**1**	**SATIR**

CMD	OPERAND1	OPERAND2
MOV	**AKOD**	**BKOD**

Açıklama

— Yukarıdaki 1 nci örnekte SATIR değişkenine 1 (bir) değeri aktarılmaktadır.
— Yukarıdaki 2 nci örnekte AKOD değişkeni içeriği BKOD değişkenine aktarılmaktadır.

STR

Kullanılabilen Bölüm : Macro Editor
İşlevi : Sayısal bir değişkenin içeriğini veya sayısal bir değeri alfa-sayısal tipe dönüştürülerek belirtilen bir değişkene aktarılmasını sağlar.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
STR	**DEĞİŞKEN–1**	**DEĞİŞKEN–2**

— DEĞİŞKEN–1 sayısal bir değer veya sayısal tipte bir değişken olmalıdır.

Örnek

CMD	OPERAND1	OPERAND2
STR	**12345**	**YAZI**

CMD	OPERAND1	OPERAND2
STR	**RAKAM**	**YAZI**

Açıklama

— Yukarıdaki 1 nci örnekte 12345 sayısal değeri alfa-sayısal tipe dönüştürülerek YAZI değişkenine aktarılmaktadır.
— Yukarıdaki 2 nci örnekte sayısal tipteki RAKAM değişkeni içeriği alfa-sayısal tipe dönüştürülerek YAZI değişkenine aktarılmaktadır.

VAL

Kullanılabilen Bölüm : Macro Editor
İşlevi : Alfa-sayısal bir değişkenin içeriğini veya alfa-sayısal bir değeri sayısal tipe dönüştürülerek belirtilen bir değişkene aktarılmasını sağlar.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
VAL	**DEĞİŞKEN–1**	**DEĞİŞKEN–2**

— DEĞİŞKEN–1 sadece rakamlardan oluşan bir değer veya değişken olmalıdır.

Örnek

CMD	OPERAND1	OPERAND2
VAL	**100**	**RAKAM**

CMD	OPERAND1	OPERAND2
VAL	**YAZI**	**RAKAM**

Açıklama

— Yukarıdaki 1 nci örnekte 100 alfa sayısal değeri sayısal tipe dönüştürülerek RAKAM değişkenine aktarılmaktadır.
— Yukarıdaki 2 nci örnekte alfa-sayısal tipteki YAZI değişkeni içeriği sayısal tipe dönüştürülerek RAKAM değişkenine aktarılmaktadır.

Dosya İşlemleri Komutları

Bu gruptaki komutlar, PDL20'de oluşturulmuş formlar (Database) üzerinde seçme, silme, kaydetme, verilere ulaşma, arama, okuma gibi işlemlerin yapılmasını sağlayan komutları içermektedir. Toplam 8 adet komuttan oluşmaktadır. Bu komutlar ve kullanım şekli şunlardır:

1) SEL : Belirtilen bir Form'un (Database) seçilmesini sağlar.
2) SEK : Formdaki belirtilen kaydı aktif hale getirilmesini sağlar.
3) GTO : Formdaki belirtilen kaydı aktif hale getirilmesini sağlar.
4) SRH : Seçili Formda herhangi bir verinin aranmasını sağlar.
5) GET : Belirtilen alan içeriğin belirtilen değişkene aktarılmasını sağlar.
6) PUT : Belirtilen değişken içeriğini formun belirtilen alanına aktarılmasını sağlar.
7) NRC : Form değişkenlerine aktarılan verilerin forma kaydedilmesini sağlar.
8) DLF : Seçili formun (Database) içeriğini siler.

SEL

Kullanılabilen Bölüm : Macro Editor
İşlevi : PDL20'de kayıtlı herhangi bir formun (Database-Veritabanı) üzerinde işlem yapabilmek için seçilmesini sağlar.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
SEL	**DEĞİŞKEN–1**	

— DEĞİŞKEN–1 bir form adı olmalıdır.

Örnek

CMD	OPERAND1	OPERAND2
SEL	**STOK**	

Açıklama

— Yukarıdaki örnekte STOK adlı formu (Database-Veritabanı) arama, veri alma, veri yazma gibi işlemler için seçilmesini sağlar.
— Seçilen form Procedure tanımlamada belirtilen sıradaki form ise bu komutun kullanılması gerekmez.

SEK

Kullanılabilen Bölüm : Macro Editor
İşlevi : Bir formda işlem yapabilmek için istenilen bir kaydı aktif hale getirmek için kullanılır.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
SEK	**DEĞİŞKEN–1**	

— DEĞİŞKEN–1 bir sayısal bir değer olmalıdır.
— DEĞİŞKEN–1 1 (bir) ile toplam kayıt sayısı arasında bir değer olmalıdır.

Örnek

CMD	OPERAND1	OPERAND2
SEL	**STOK**	
SEK	**12**	

Açıklama

— Yukarıdaki örnekte STOK adlı form (Database-Veritabanı) aktif form yapıldıktan sonra baştan 12 nci (on iki) kaydın aktif hale getirilmesini sağlar.

GTO

Kullanılabilen Bölüm : Macro Editor
İşlevi : Bir formda işlem yapabilmek için istenilen bir kaydı aktif hale getirmek için kullanılır.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
GTO	**DEĞİŞKEN–1**	

Not:
— DEĞİŞKEN–1 bir sayısal bir değer olmalıdır.
— DEĞİŞKEN–1 1 (bir) ile toplam kayıt sayısı arasında bir değer olmalıdır.
— Son kaydı aktif hale getirebilmek için DEĞİŞKEN–1 **RECORDL** olmalıdır.

Örnek

CMD	OPERAND1	OPERAND2
SEL	**STOK**	
GTO	**5**	

Açıklama

— Yukarıdaki örnekte STOK adlı form (Database-Veritabanı) aktif form yapıldıktan sonra baştan 5 nci (Beş) kaydın aktif hale getirilmesini sağlar.

Örnek

CMD	OPERAND1	OPERAND2
SEL	**STOK**	
GTO	**RECORDL**	

Açıklama

— Yukarıdaki örnekte STOK adlı form (Database-Veritabanı) aktif form yapıldıktan en son kaydın aktif hale getirilmesini sağlar.

SRH

Kullanılabilen Bölüm : Macro Editor
İşlevi : Bir formda aktif kayıttan itibaren belirtilen alanda belirtilen verinin aranmasını sağlar.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
SRH	**DEĞİŞKEN–1**	**DEĞİŞKEN–2**

Değişken–1: Arama işleminin yapılacağı form alanın adı veya sıra numarasıdır.

Değişken–2: Aranacak veriyi içeren değişkendir.

Not:

— Aramaya başlamadan önce aramanın yapılacağı form aktif (seçili) olmalıdır.
— Aramaya başlamadan önce ilk kaydın aktif hale getirilmesi gerekmektedir.
— Aranan veri bulunduğunda aktif kayıt verinin bulunduğu kayıt olacak ve **EQ** kontrol değişkeni doğru değeri içerecektir.
— Aranan veri bulunamadığında **NE** kontrol değişkeni doğru değeri içerecektir.

Örnek

CMD	OPERAND1	OPERAND2
SEL	**STOK**	
GTO	**1**	
SRH	**1**	**VKOD**
JMP	**EQ**	**38**
JMP	**NE**	**50**

Açıklama

— Yukarıdaki örnekte STOK adlı form (Database-Veritabanı) seçildikten, aktif kayıt 1 nci kayıt yapıldıktan sonra STOK formundaki 1 nci alanda VKOD değişkeni içeriği aranması sağlanmıştır.
— Eğer STOK formunda VKOD değişkeni içeriği bulunursa program 38 nci satıra, bulunamazsa 50 nci satıra yönlendirilmesi sağlanmıştır.

Örnek

CMD	OPERAND1	OPERAND2
SEL	**STOK**	
GTO	**1**	
SRH	**BKOD**	**VKOD**
JMP	**EQ**	**23**
JMP	**NE**	**44**

Açıklama

— Yukarıdaki örnekte STOK adlı form (Database-Veritabanı) seçildikten, aktif kayıt 1 nci kayıt yapıldıktan sonra STOK formundaki BKOD adlı alanda VKOD değişkeni içeriği aranması sağlanmıştır.
— Eğer STOK formunda VKOD değişkeni içeriği bulunursa program 23 ncü satıra, bulunamazsa 44 ncü satıra yönlendirilmesi sağlanmıştır.

GET

Kullanılabilen Bölüm : Macro Editor
İşlevi : Aktif kayıtta belirtilen alandaki veriyi belirtilen değişkene aktarılmasını sağlar.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
GET	**DEĞİŞKEN–1**	**DEĞİŞKEN–2**

Not:
— DEĞİŞKEN–1 formdaki alanın sıra numarası veya alanın adı olmalıdır.

— DEĞİŞKEN–2 verinin aktarılacağı değişkendir.

Örnek

CMD	OPERAND1	OPERAND2
SEL	**STOK**	
GTO	**25**	
GET	**1**	**VKOD**

Açıklama

— Yukarıdaki örnekte STOK adlı form (Database-Veritabanı) aktif form, baştan 25 nci (Yirmi beş) kayıt aktif kayıt hale getirildikten sonra 1 nci sıradaki alandaki veri VKOD değişkenine aktarılması sağlanmaktadır.

Örnek

CMD	OPERAND1	OPERAND2
SEL	**STOK**	
GTO	**15**	
GET	**BARKOD**	**VKOD**

Açıklama

— Yukarıdaki örnekte STOK adlı form (Database-Veritabanı) aktif form, baştan 15 nci (On beş) kayıt aktif kayıt hale getirildikten sonra BARKOD adlı form alanındaki veri VKOD değişkenine aktarılması sağlanmaktadır.

PUT

Kullanılabilen Bölüm : Macro Editor
İşlevi : Aktif kayıtta belirtilen alana belirtilen verinin aktarılmasını sağlar.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
PUT	**DEĞİŞKEN–1**	**DEĞİŞKEN–2**

Not:

— DEĞİŞKEN–1 formdaki alanın sıra numarası veya alanın adı olmalıdır.

— DEĞİŞKEN–2 aktarılacak veriyi içeren değişkendir.

Örnek

CMD	OPERAND1	OPERAND2
SEL	**STOK**	
GTO	**15**	
PUT	**2**	**VURKOD**

Açıklama

— Yukarıdaki örnekte STOK adlı form (Database-Veritabanı) aktif form, baştan 15 nci (On beş) kayıt aktif kayıt hale getirildikten sonra VURKOD değişkeni içeriği STOK formunun 2 nci sıradaki alanına aktarılması sağlanmaktadır.

Örnek

CMD	OPERAND1	OPERAND2
SEL	**STOK**	
GTO	**5**	
PUT	**3**	**"BILKUR"**

Açıklama

— Yukarıdaki örnekte STOK adlı form (Database-Veritabanı) aktif form, baştan 5 nci (Beş) kayıt aktif kayıt hale getirildikten sonra STOK formunun 3 ncü sıradaki alanına BILKUR metninin aktarılması sağlanmaktadır.

NRC

Kullanılabilen Bölüm : Procedure ve Macro Editor
İşlevi : Form değişkenlerindeki veriyi aktif Hafızadaki verinin forma kaydedilmesini sağlar.

Not:

- Hafızada verinin form alanlarına aktarılması gerekmektedir.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
NRC		

Örnek

CMD	OPERAND1	OPERAND2
SEL	**STOK**	
MOV	**VKOD**	**BARKOD**
MOV	**VURAD**	**URAD**
NRC		

Açıklama

VKOD, VURAD birer hafıza değişkeni, BARKOD, URAD birer form değişkeni olarak belirtilmiştir.

Yukarıdaki örnekte STOK formu seçildikten, VKOD değişkeni içeriği BARKOD değişkenine, VURAD değişkeni içeriği URAD değişkenine aktarıldıktan sonra form değişkenleri içerikleri STOK formunun sonuna yeni bir kayıt olarak kaydedilmesini sağlamaktadır.

DLF

Kullanılabilen Bölüm : Macro Editor
İşlevi : Seçili form içeriğinin silinmesini sağlar.
Kullanım Şekli

CMD	OPERAND1	OPERAND2
DLF		

Örnek

CMD	OPERAND1	OPERAND2
SEL	**STOK**	
DLF		

Açıklama

Yukarıdaki örnekte STOK formu seçildikten sonra STOK dosyası içeriğinin silinmesi sağlanır.

Kontrol İşlemleri Komutları

Bu gruptaki komutlar işlemlerin geçici olarak bekletilmesini, sonlandırılmasını, programın istenilen bir bölüme yönlendirilmesini ve programdan çıkışı sağlayan komutları içerir. 4 adet komuttan oluşmaktadır. Bu komutlar ve kullanım şekli şunlardır:

1) DLY : Belirtilen bir Form'un (Database) seçilmesini sağlar.
2) JMP : Formdaki belirtilen kaydı aktif hale getirilmesini sağlar.
3) END : Formdaki belirtilen kaydı aktif hale getirilmesini sağlar.
4) EXT : Seçili Formda herhangi bir verinin aranmasını sağlar.

DLY

Kullanılabilen Bölüm : Macro Editor
İşlevi : Belirtilen süre kadar programın çalışmasını durdurur.
Kullanım Şekli

CMD	OPERAND1	OPERAND2
DLY	**DEĞİŞKEN–1**	

Not:
— DEĞİŞKEN–1 milisaniye cinsinden 0–32.767 arası bir değerdir.
— 1 saniye 1.000 milisaniyedir.

Örnek

CMD	OPERAND1	OPERAND2
CLS		
MSG	**"TOPLAM:"**	**1**
MSG	**VTOP**	**11**
DLY	**2000**	
CLS		

Açıklama

Yukarıdaki örnekte PDL20 ekranı silindikten sonra ekranda 1 nci koordinatta (1. Satır, 1 sütun) **TOPLAM:** mesajı, 11 nci koordinatta **VTOP** değişkeni içeriği görüntülenmekte ve ardından 2 saniye süre ile işlem durdurulmaktadır. 2 saniye durdurma (bekletme) işleminden sonra programın işleyişi bir alt satırdan devam edilmesi sağlanmaktadır.

JMP

Kullanılabilen Bölüm : Macro Editor
İşlevi : Programın işleyişini kontrol değişkeninin durumuna bağlı olarak istenilen bir program bölümüne (satırına) yönlendirilmesini sağlar.

Kullanım Şekli

CMD	OPERAND1	OPERAND2
JMP	**DEĞİŞKEN–1**	**DEĞİŞKEN–2**

Not:
— JMP komutundan önce bir karşılaştırma yapılması gerekmektedir.
— DEĞİŞKEN–1 kontrol değişkenidir.
— DEĞİŞKEN–2 macro içerisindeki bir satır numarası veya bir etiket olabilir.

Kontrol Değişkenleri
EQ : Karşılaştırılan değişkenlerin eşit bilgi içermesi
NE : Karşılaştırılan değişkenlerin eşit bilgi içermemesi
LT : Değişken-1'in Değişken-2'den küçük olması durumu
GT : Değişken-1'in Değişken-2'den büyük olması durumu
UC : Kontrol değişkeninin durumuna bakılmaksızın yönlendirilmesini sağlar.

Örnek

CMD	OPERAND1	OPERAND2
CMP	**AKOD**	**BKOD**
JMP	**EQ**	**:ABC**
JMP	**LT**	**10**
JMP	**GT**	**:DEF**
JMP	**NE**	**20**
:ABC		
..	**..**	**..**
:DEF	**..**	**..**

Açıklama

Yukarıdaki örnekte ilk satırda AKOD değişkeni ile BKOD değişkeni içerikleri karşılaştırılmaktadır. Karşılaştırmanın sonucuna göre;
— İki değişkenin eşit bilgi içermesi durumunda programın ABC adlı etiket satırına yönlendirilmesi sağlanmıştır.
— AKOD değişkeni içeriği BKOD değişkeni içeriğinden küçük olması durumunda programın 10 nci satıra yönlendirilmesi sağlanmıştır.
— AKOD değişkeni içeriği BKOD değişkeni içeriğinden büyük olması durumunda programın DEF adlı etiket satıra 30 ncu satıra yönlendirilmesi sağlanmıştır.

— AKOD değişkeni içeriği BKOD değişkeni içeriğinden farklı olması durumunda programın 40 ncı satıra yönlendirilmesi sağlanması sağlanmaktadır.

END

Kullanılabilen Bölüm : Macro Editor
İşlevi : Çalışan makronun çalışmasını sona erdirir.
Kullanım Şekli

CMD	OPERAND1	OPERAND2
END		

Örnek

CMD	OPERAND1	OPERAND2
MSG	**"1.DEVAM"**	**1**
MSG	**"2.CIKIS"**	**17**
INX	**TUS**	
CMP	**TUS**	**"2"**
JMP	**EQ**	**:SON**
JMP	**NE**	**1**
..	**..**	**..**
:SON		
END		

Açıklama

Örnekte programın işleyişi sırasında PDL20 ekranında 1 nci koordinatta DEVAM mesajı görüntülenmekte, 17 nci koordinatta CIKIS mesajı görüntülenmekte ve klavyeden sadece bir tuşa basılması beklenmektedir. Basılan tuşun 2 olması durumunda programın SON adlı etiket satırına yönlendirilmesi sağlanmaktadır. Son etiket satırından sonra END komutu ile makronun sonlandırılması sağlanmaktadır. Eğer basılan tuş 2 dışında bir rakam ise programın 1 nci satıra yönlendirilmesini sağlamaktadır.

EXT

Kullanılabilen Bölüm : Macro Editor
İşlevi : Çalışan Task'ı sona erdirir.
Kullanım Şekli

CMD	OPERAND1	OPERAND2
EXT		

Not
— **İşlem** sonucu RUN TASK menüsüne dönüş yapılır.
Örnek

CMD	OPERAND1	OPERAND2
INX	**TUS**	
CMP	**TUS**	**"1"**
JMP	**EQ**	**:CIKIS**
JMP	**NE**	**1**
..	**..**	**..**
:SON		
EXT		

Açıklama

Yukarıdaki örnekte programın işleyişi sırasında klavyeden sadece bir tuşa basılması beklenmektedir. Basılan tuşun 1 olması durumunda programın CIKIS adlı etiket satırına yönlendirilmesi sağlanmaktadır. CIKIS etiket satırından sonra EXT komutu ile task'ın sonlandırılması sağlanmaktadır. Eğer basılan tuş 1 dışında bir rakam ise programın 1 nci satıra yönlendirilmesini sağlamaktadır.

Örnek Program

Aşağıdaki program bir işletmede ürünlerin satışının yapılabilmesini ve gerektiğinde sayım yapılabilmesini sağlayan bir özelliğe sahiptir.

Ürünlerin bilgilerini içeren bir dosya program ilk kez yüklenirken program ile birlikte PDL20'ye yüklenerek kullanılmaktadır.

SATIŞ bölümünde ilk olarak satış yapılacak müşteri için bir müşteri kodu girilmesi istenmektedir. Müşteri numarası girildikten sonra ürün satışına geçilmektedir. Ürünün barkodu okutulduktan sonra okutulan barkod önceden yüklenmiş STOK formunda aranmakta, eğer okutulan barkoda ait bir ürün bulanamaz ise "ÜRÜN BULUNAMADI" diye bir mesaj gösterilmekte ve program tekrar barkod sorma bölümüne dönmektedir. Eğer okutulan barkod kayıtlı bir ürüne aitse ürünün BARKODU, ADI, FİYATI ve STOK miktarı PDL20 ekranında gösterilmekte ve satılacak miktar sorulmaktadır. Miktar girildikten sonra ekranda o ürüne ait tutar ve o müşteriye ait toplam tutar gösterilmektedir. Ardından müşteri no, barkod, ürün adı, miktar ve fiyat SATIS adlı formun en sonuna kayıt edilmekte ve program ürün satışına devam için barkod okutma bölümüne dönmektedir.

SAYIM bölümünde okutulan barkod öncelikle STOK formunda aranmakta, aranan veri STOK dosyasında bulunamazsa "ÜRÜN BULUNAMADI" mesajı gösterilmekte ve ardından barkod okutma bölümüne dönmektedir. Eğer okutulan barkod STOK dosyasında bulunursa ürün adı ekranda gösterildikten sonra Miktar sorulmakta ve Barkod no ve girilen miktar SAYIM dosyasının sonuna kayıt edilmektedir.

Program 3 adet formdan oluşmaktadır. Bu formlar ve özellikleri şunlardır:

STOK: Program ilk kez PDL20'ye yüklenirken birlikte yüklenen ve ürünlerin barkod numarasını, adını, fiyatını ve stok miktarını içeren formdur.

SATIS: Yapılan satışların kayıt edildiği ve satış(sipariş) numarası, ürünlerin barkod numarası, ürün adı, miktar, birim fiyatı bilgilerini içeren formdur.

SAYIM: Yapılan sayım işleminin kayıt edildiği ve ürünlerin barkod numarasını, ürün adını ve miktarını içeren formdur.

Programlama Adımları

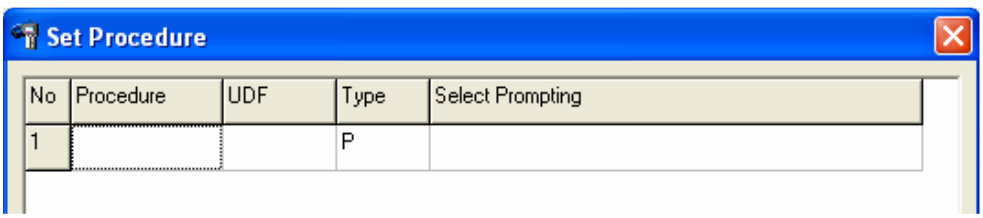
1. WinTaskGen programının çalıştırılması

WinTaskGen programını çalıştırabilmek için Başlat / Programlar / WinTaskGen menüsünden WinTaskGen simgesini çalıştırın. (Ayrıntılı bilgi için sayfa:**3**'deki WinTaskGen programının çalıştırılması bölümüne bakınız.)

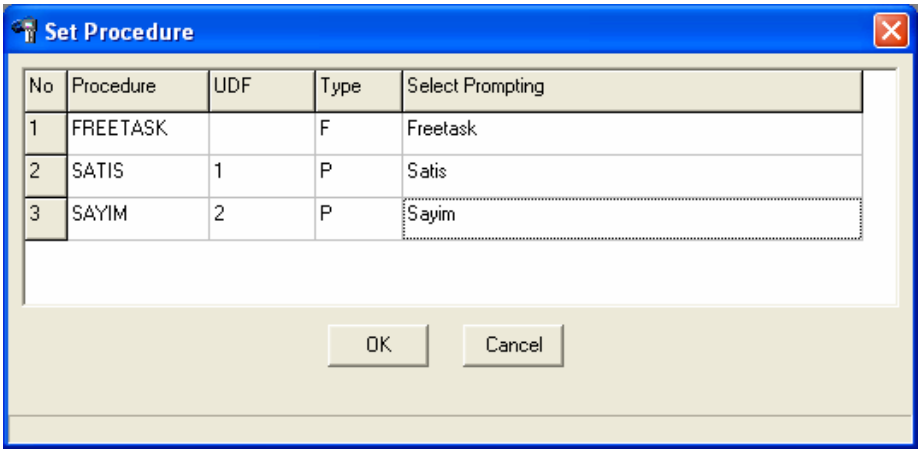
Not: WinTaskGen Programın sürümü tarihe göre farklı olabilir. Programın en son sürümü için **www.bilkur.com.tr** adresinden **download** bölümüne bakınız.

2. Task'ın Procedure'lerini oluşturma

WinTaskGen programı ile yazılan programlara task denilir. Bir task'ı oluşturmanın ilk adımı ise o Task'a bağlı procedure'lerin oluşturulmasıdır. İlk defa procedure oluşturabilmek için WinTaskGen penceresinde bulunan Edit Task düğmesi tıklanarak açılan Task editör penceresinde Create Task menüsünden New komutu çalıştırılmadır. Aşağıdaki şekilde de görülen açılan diyalog kutusunda oluşturulacak Task'ın içereceği procedure'ler belirlenmelidir.



Not: Procedure tanımlama kuralları ile ilgili ayrıntılı bilgi sayfa 5'de bulunmaktadır.



Yukarıdaki ekranda 3 adet procedure tanımlanmıştır.

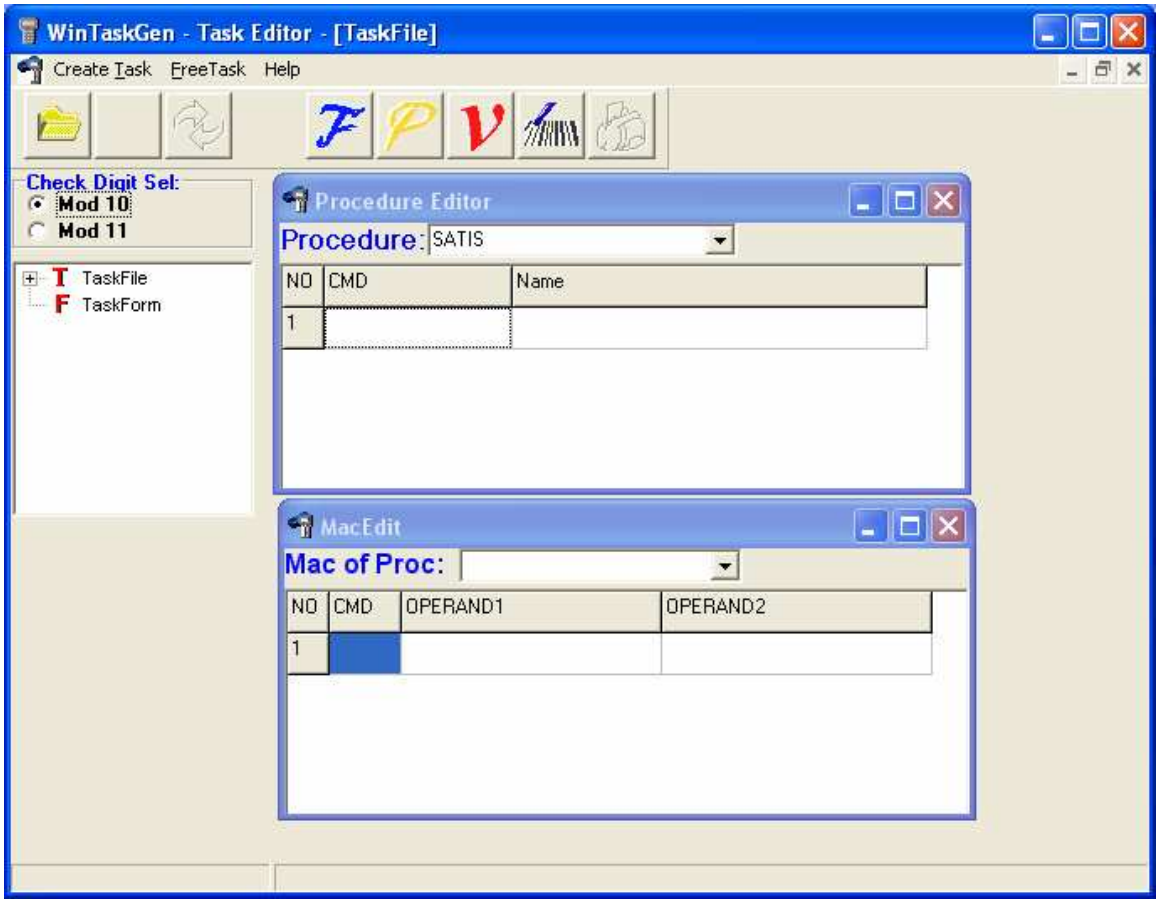
FREETASK procedure'ünün PDL20 ekranında 1 nci sırada Freetask mesajı ile görüntüleneceği belirtilmiştir. FREETASK procedure'ünün programlanmasına gerek bulunmamaktadır. FREETASK hakkındaki ayrıntılı bilgi sayfa 6'de bulanmaktadır.

SATIS procedure'ünün 1 numaralı formu kullanacağı ve PDL20 ekranında 2 nci sırada Satis mesajı ile görüntüleneceği belirtilmiştir. SATIS procedure'ünün programlanması ile ilgili ayrıntı sayfa 39'de bulanmaktadır.

SAYIM procedure'ünün 2 numaralı formu kullanacağı ve PDL20 ekranında 3 nci sırada Satis mesajı ile görüntüleneceği belirtilmiştir. SATIS procedure'ünün programlanması ile ilgili ayrıntı sayfa 44'de bulanmaktadır.

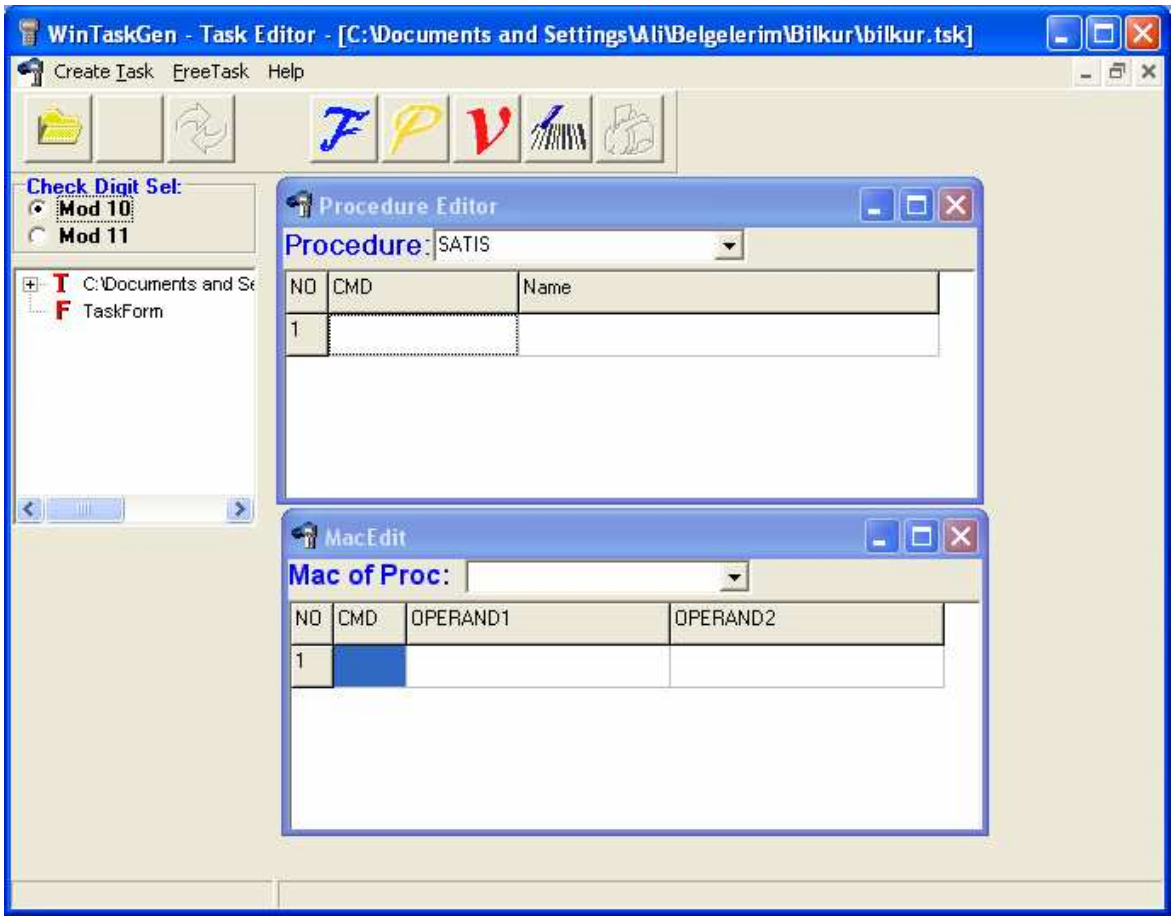
Procedure tanımlamaları tamamlandıktan sonra Task Editör ekranı aşağıdaki gibidir.

Yukarıda yapılan tanımlamalardan en az 1'i isteğe bağlı olarak task'ın ilk oluşturulmasında yapılabilir, diğerleri programlama esnasında ihtiyaca göre ilave edilebilir. Procedure tanımlamalarının ardından task editör ekranı aşağıdaki gibidir.



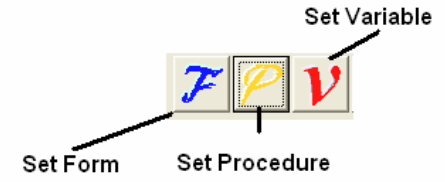
3. Task'ı Kaydetme

Aktif Task'ı isteğe bağlı olarak kayıt etmek için Task Editör penceresinde Create Task menüsünden **"Save"** menü komutunu çalıştırın. Yukarıdaki örnekte oluşturulan Task'ın kayıt edilmesinden sonraki görüntüsü aşağıdaki gibidir.

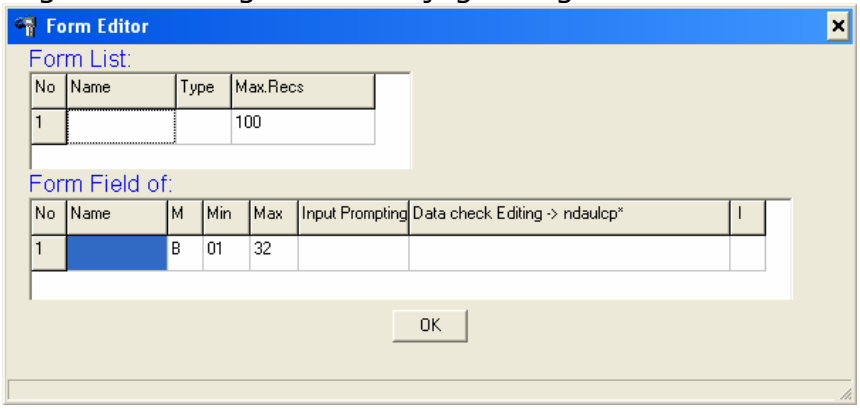


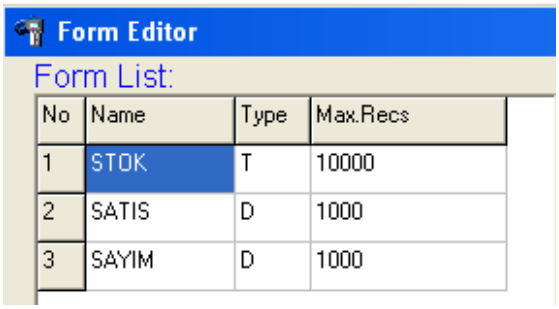
4. Task'ın Form'larını Oluşturma

Programın işleyişi sırasında ihtiyaç duyulan verilerin saklanması, yapılan girişlerin kalıcı olarak saklanmasını sağlayan bölümler olan Form veya Formların oluşturulabilmesi için Task Editör ekranında bulunan ve aşağıdaki şekilde de görülen Set form düğmesi tıklanarak açılan **"Set Form"** diyalog kutusu kullanılmalıdır.



Set Form diyalog kutusunun görüntüsü aşağıdaki gibidir.




Form Editor

Form List:

No	Name	Type	Max.Recs
1	STOK	T	10000
2	SATIS	D	1000
3	SAYIM	D	1000

Yukarıdaki şekilde 3 adet form tanımlanmıştır.
STOK formu en fazla 10.000 adet kayıt içerebileceği ve programın (task) PDL20'ye yüklenirken STOK verilerinin de PDL20'ye yükleneceği belirlenmiştir.
(**Not:** Stok dosyasının oluşturulması hakkındaki detaylı bilgi sayfa 46'da bulunmaktadır)
SATIŞ formu en fazla 1.000 adet kayıt içerebileceği ve programın işleyişi sırasında yapılan girişlerin kayıt edileceği belirlenmiştir. **SAYIM** formu en fazla 1.000 adet kayıt içerebileceği ve programın işleyişi sırasında yapılan girişlerin kayıt edileceği belirlenmiştir.
STOK formunun alanlarının belirlenmiş hali aşağıdaki gibidir.

Form:STOK

No	Name	M	Min	Max	Input Prompting	Data check Editing -> ndaulcp*	I
1	URKOD	B	01	13			
2	URAD	B	01	32			
3	FIYAT	B	01	10			
4	MIKTAR	B	01	05			

SATIS formunun alanlarının belirlenmiş hali aşağıdaki gibidir.

Form:SATIS

No	Name	M	Min	Max	Input Prompting	Data check Editing -> ndaulcp*	I
1	SIPNO	B	01	05			
2	URKOD	B	01	13			
3	URAD	B	01	32			
4	URMIKT	B	01	05			
5	URFIYAT	B	01	10			

SAYIM formunun alanlarının belirlenmiş hali aşağıdaki gibidir.

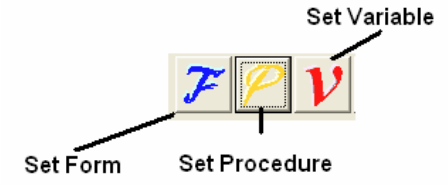
Form:SAYIM

No	Name	M	Min	Max	Input Prompting	Data check Editing -> ndaulcp*	I
1	URKOD	B	01	13			
2	URAD	B	01	32			
3	URMIKT	B	01	05			

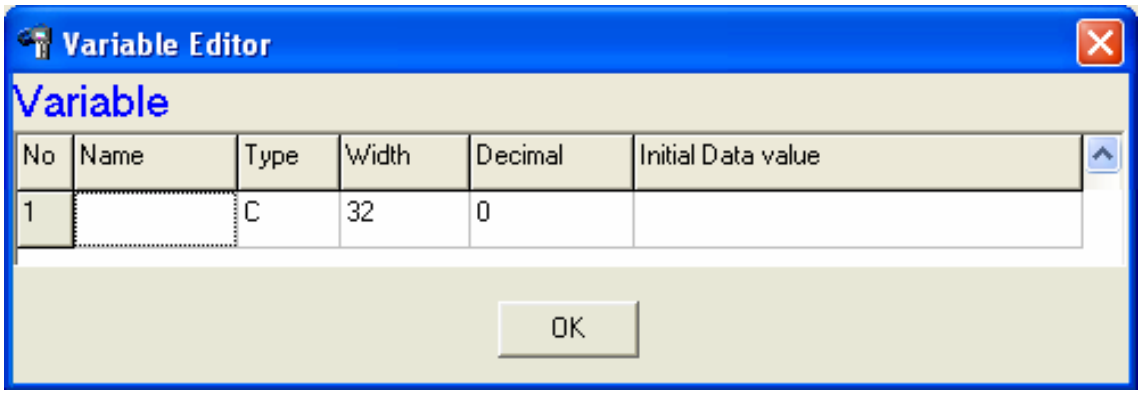
Not: Form tanımlama kuralları ile ilgili ayrıntılı bilgi sayfa 7'de bulunmaktadır.

5. Değişken Tanımlama

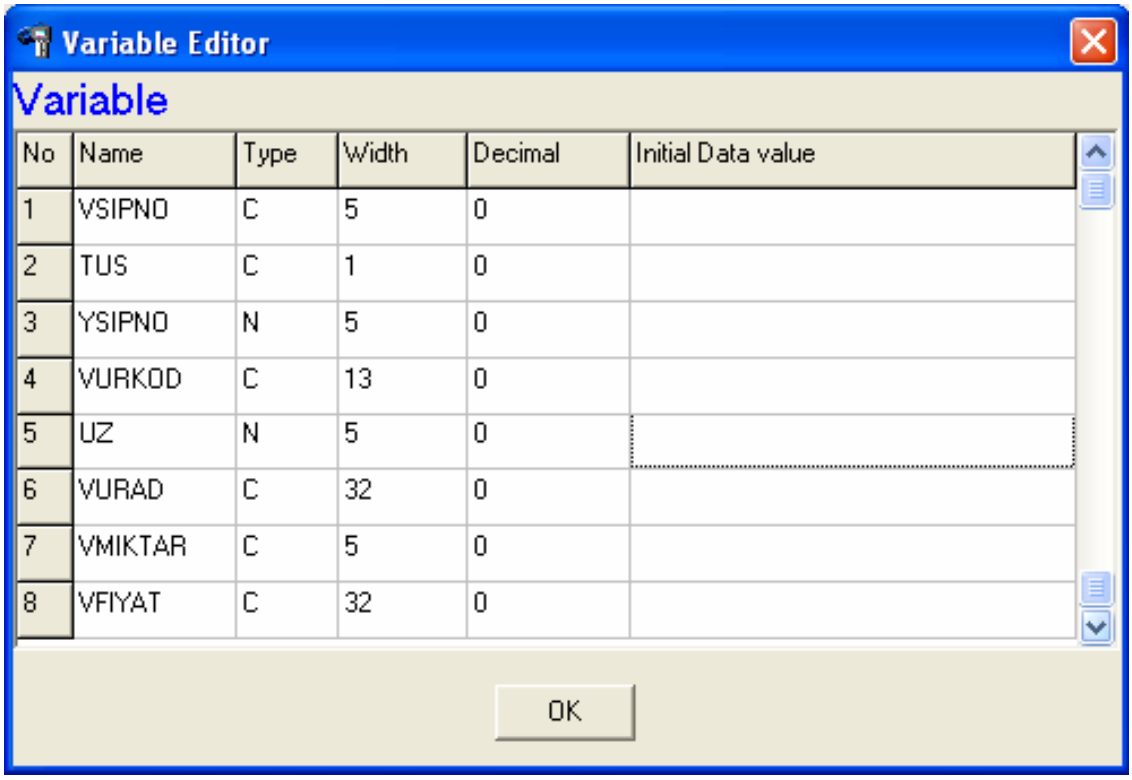
Programın işleyişi sırasında gerekli değişkenleri tanımlayabilmek için Task Editör ekranında bulunan ve aşağıdaki şekilde de görülen Set Variable düğmesi tıklanarak açılan **"Set Variable"** diyalog kutusu kullanılmalıdır.



Set Form diyalog kutusunun görüntüsü aşağıdaki gibidir.



Örnek programın işleyişinde ihtiyaç duyulan değişkenlerin tanımlanmış hali aşağıdaki şekilde görülmektedir.

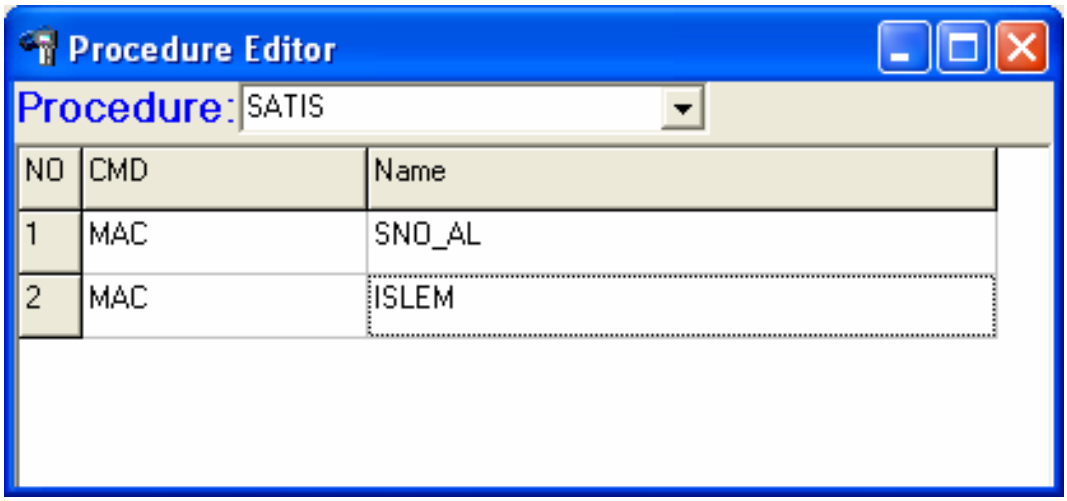


Not: Değişken tanımlama hakkındaki ayrıntılı bilgi sayfa 8'de bulunmaktadır.

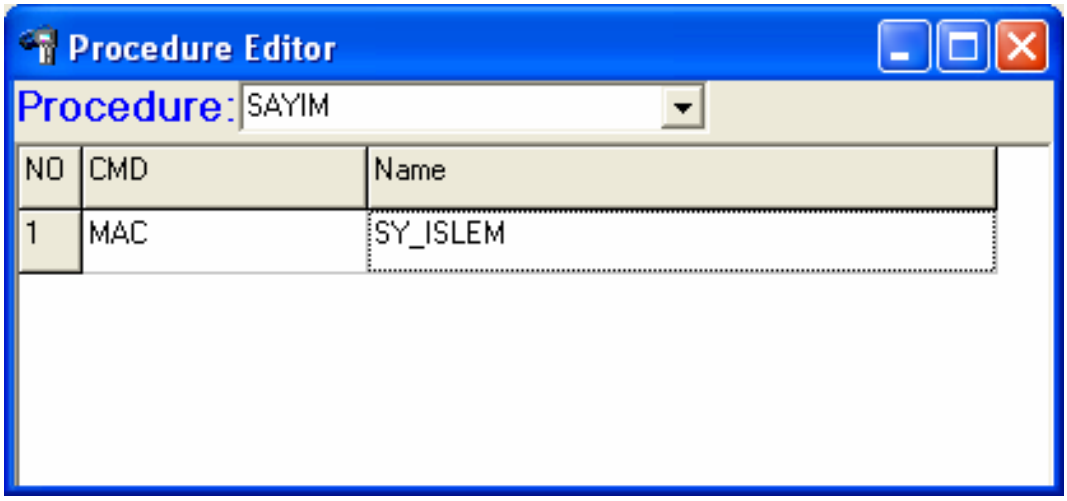
6. Makro Tanımlama ve programlama

Programın (Task) işleyişini sağlamak için procedure'lere bağlı macro'lara ihtiyaç duyulmaktadır. Örnek programda FREETASK, SATIS ve SAYIM adlı 3 procedure bulunmaktadır. Aşağıdaki şekillerde SATIS ve SAYIM procedure'leri için tanımlanmış macro'lar bulunmaktadır.

Not: FREETASK procedure'ü aslında başlı başına bir task'dır. PDL20'ye fabrikadan yüklenen ve silinemeyen özelliğe sahiptir. Yeni bir task yüklendiğinde FREETASK silinmez sadece kullanıma kapatılmış olmaktadır. Örnek programda sayfa 34'de procedure tanımlama bölümünde FREETASK tanımlamasında tip olarak F yazılarak FREETASK programı örnek programa dahil edilmiştir. Böylece FREETASK örnek program içerisinden kullanıma açılmıştır.

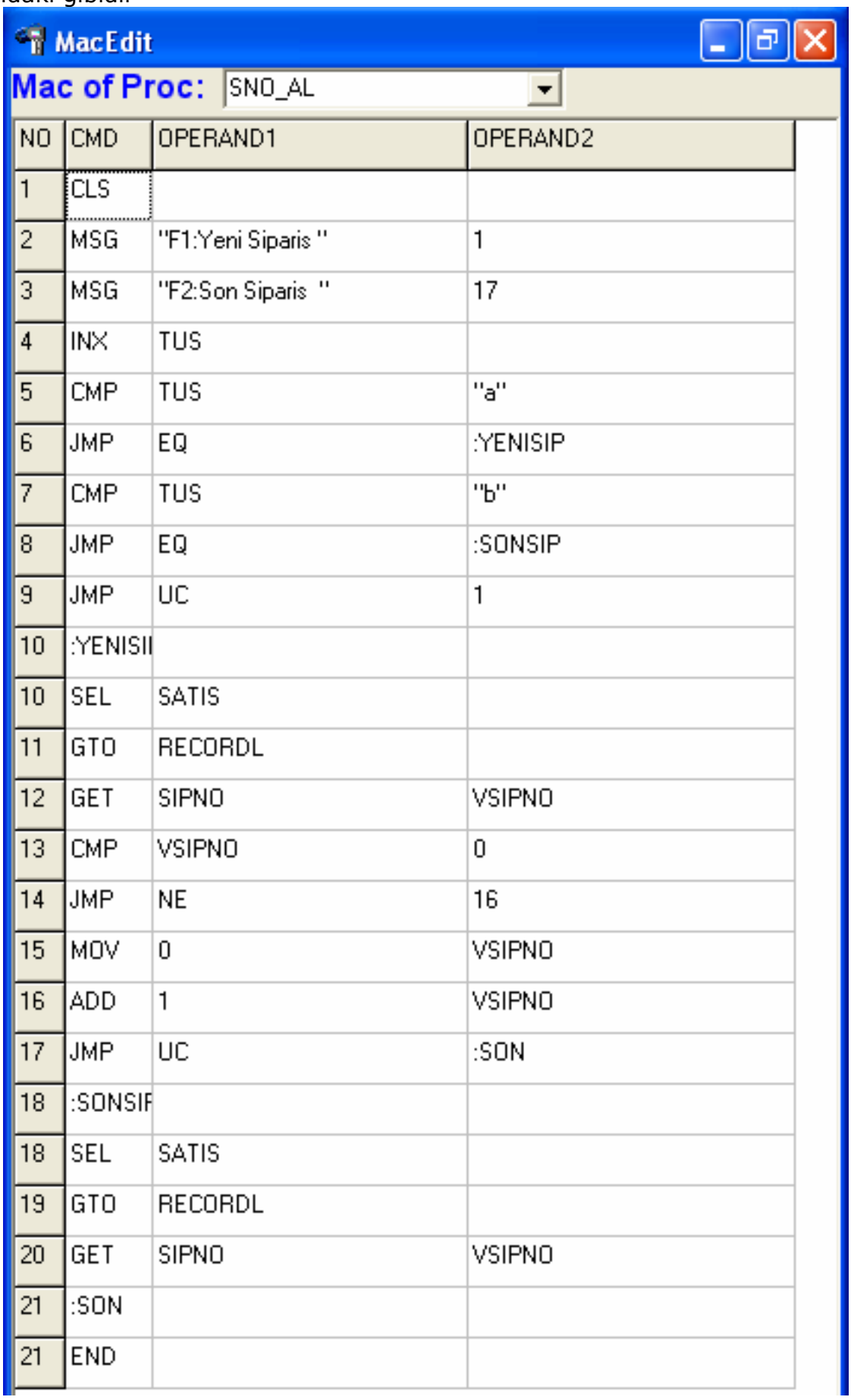


Yukarıdaki şekilde SATIS procedure'ünün SNO_AL ve ISLEM olmak üzere 2 adet macro'ya sahip olduğu görülmektedir. Bu Macro'ların programlama detayları 39-41 nci sayfalardadır.



Yukarıdaki şekilde SAYIM procedure'ünün SY_ISLEM adlı 1 adet macro'ya sahip olduğu görülmektedir. Bu Macro'nun programlama detayı 42-43 ncü sayfalardadır.

SATIS procedure'ünün **SNO_AL** Macro'sunun programlanmış halinin görüntüsü aşağıdaki gibidir



NO	CMD	OPERAND1	OPERAND2
1	CLS		
2	MSG	"F1:Yeni Siparis "	1
3	MSG	"F2:Son Siparis "	17
4	INX	TUS	
5	CMP	TUS	"a"
6	JMP	EQ	:YENISIP
7	CMP	TUS	"b"
8	JMP	EQ	:SONSIP
9	JMP	UC	1
10	:YENISII		
10	SEL	SATIS	
11	GTO	RECORDL	
12	GET	SIPNO	VSIPNO
13	CMP	VSIPNO	0
14	JMP	NE	16
15	MOV	0	VSIPNO
16	ADD	1	VSIPNO
17	JMP	UC	:SON
18	:SONSIF		
18	SEL	SATIS	
19	GTO	RECORDL	
20	GET	SIPNO	VSIPNO
21	:SON		
21	END		

SNO_AL Macro satırlarının Açıklaması

1. Ekran Temizle.

2. Ekranın 1 nci koordinatında (1 nci satır,1 nci sütun) F1:Yeni Siparis yaz.

3. Ekranın 17 nci koordinatında (2 nci satır,1 nci sütun) F2:Son Siparis yaz.

4. Tuş takımından girilen 1 karakterlik veriyi TUS değişkenine aktar.

5. 4 ncü satırda basılan tuşu F1 tuşunun kodu (F1=a) ile karşılaştır.

6. Basılan tuş F1 ise programın işleyişini YENISIP etiketine (10. satır) yönlendir.

7. 4 ncü satırda basılan tuşu F2 tuşunun kodu (F2=a) ile karşılaştır.

8. Basılan tuş F2 ise programın işleyişini SONSIP etiketine (19. satır) yönlendir.

9. Basılan tuş F1 veya F2 tuşlarından biri değilse programın işleyişini 1. satıra yönlendir.

10. YENISIP etiketi

10. SATIS formunu aktif hale getir.

11. SATIS formunun son kaydını aktif kayıt yap.

12. Aktif kayıttaki SIPNO alanının içeriğini VSIPNO değişkenine ata.

13. VSIPNO değişkeninin içeriğinin 0 (sıfır) karşılaştır.

14. 14.satırda yapılan karşılaştırmada VSIPNO değeri 0 (sıfır) değil ise programın işleyişini 16. satıra yönlendir.

15. VSIPNO değişkenin içeriğine karakter olarak 0 (sıfır) değerini ata.

16. VSIPNO değişkeninin içeriğindeki sayısal değeri 1 arttır.

17. Programın işleyişini SON etiketine (21. satır) yönlendir.

18. SONSIP etiketi

18. SATIS formunu aktif hale getir.

19. SATIS formunun son kaydını aktif kayıt yap.

20. Aktif kayıttaki SIPNO alanının içeriğini VSIPNO değişkenine ata.

21. SON etiketi.

22. Aktif Macro'nun işleyişini bitir.

SATIS procedure'ünün **ISLEM** Macro'sunun programlanmış halinin görüntüsü aşağıdaki gibidir.

MacEdit

Mac of Proc: ISLEM

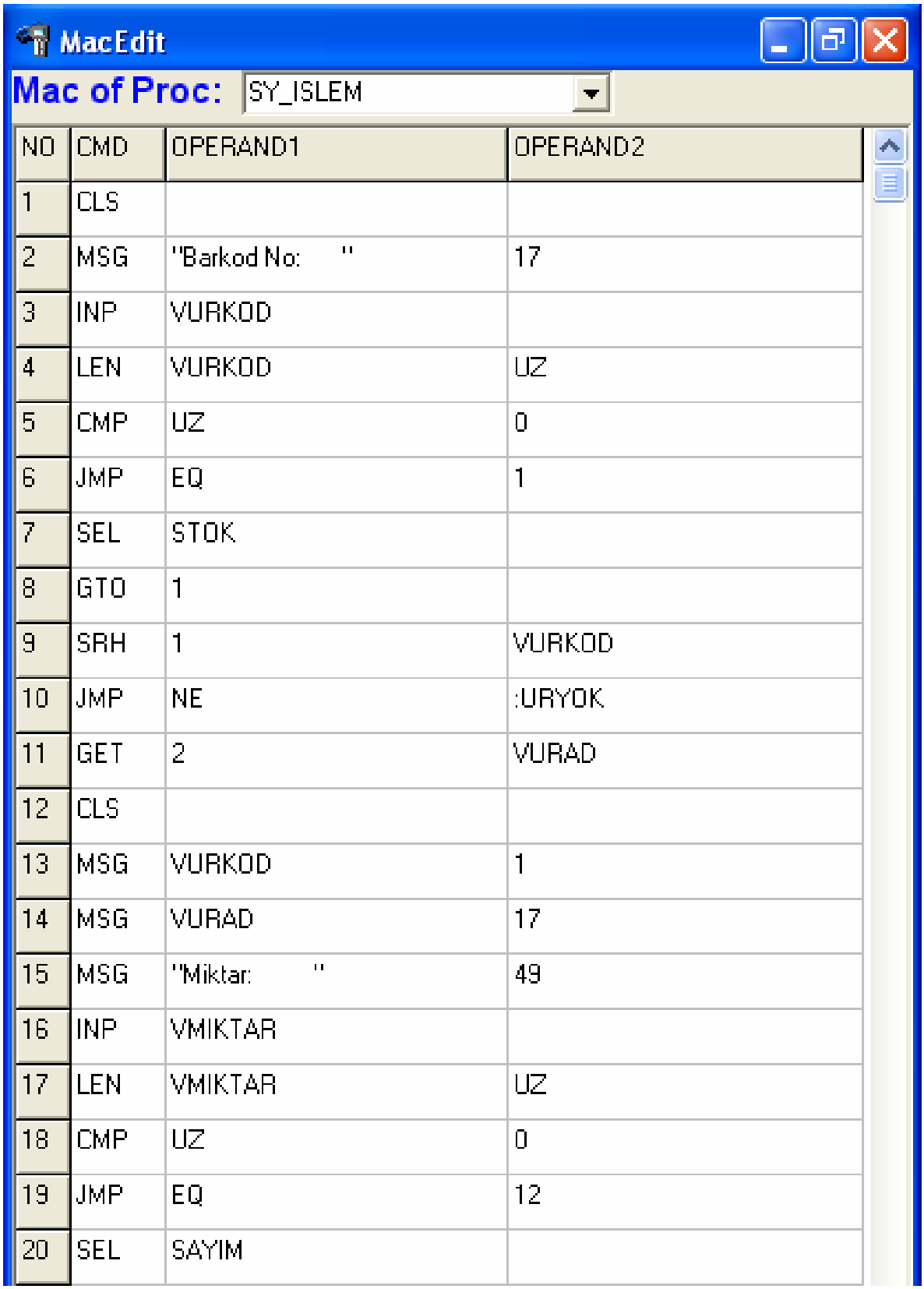
NO	CMD	OPERAND1	OPERAND2
1	CLS		
2	MSG	"Siparis No:"	1
3	MSG	YSIPNO	12
4	MSG	"Barkod No: "	17
5	INP	VURKOD	
6	LEN	VURKOD	UZ
7	CMP	UZ	0
8	JMP	EQ	4
9	SEL	STOK	
10	GTO	1	
11	SRH	1	VURKOD
12	JMP	NE	:URYOK
13	GET	2	VURAD
14	GET	3	VMIKTAR
15	GET	4	VFIYAT
16	CLS		
17	MSG	VURKOD	1
18	MSG	VURAD	17
19	MSG	VMIKTAR	49
20	MSG	VFIYAT	65
21	MSG	" <<Devam>> "	81

22	INX	TUS	
23	CLS		
24	MSG	"Miktar: "	1
25	INP	VMIKTAR	
26	LEN	VMIKTAR	UZ
27	CMP	UZ	0
28	JMP	EQ	23
29	SEL	SATIS	
30	MOV	YSIPNO	SIPNO
31	MOV	VURKOD	URKOD
32	MOV	VURAD	URAD
33	MOV	VMIKTAR	URMIKT
34	MOV	VFIYAT	URFIYAT
35	NRC		
36	JMP	UC	1
37	:URYOK		
37	CLS		
38	BEP	500	600
39	MSG	"Urun Bulunamadi."	33
40	INX	TUS	
41	JMP	UC	1

ISLEM Macro satırlarının Açıklaması

1. Ekran Temizle.
2. Ekranın 1 nci koordinatında (1 nci satır,1 nci sütun) Siparis No: yaz.
3. Ekranın 12 nci koordinatında (1 nci satır,12 nci sütun) YSIPNO değişkeninin içeriğini yaz.
4. Ekranın 17 nci koordinatında (2 nci satır,1 nci sütun) Barkod No: yaz.
5. Tuş takımından veya lazer tarama tuşundan girilen veriyi VURKOD değişkenine aktar.
6. 5 nci satırda okunan VURKOD değişkeninin karakter uzunluğunu UZ değişkenine ata.
7. UZ değişkeninin içeriğini 0 (Sıfır) ile karşılaştır.
8. 7.satırda yapılan karşılaştırmada sonuç doğru ise programın işleyişini 4. satıra yönlendir.
9. STOK formunu aktif hale getir.
10. STOK formunun 1 nci kaydını aktif kayıt yap.
11. STOK formunun 1 nci alanında VURKOD değişkenin içeriğini ara.
12. 11.satırda yapılan aramada aranılan değer bulunamamışsa programın işleyişini URYOK etiketine (37.satır) yönlendir.
13. STOK formunun aktif kaydının 2 alanını VURAD değişkenine ata.
14. STOK formunun aktif kaydının 3 alanını VMIKTAR değişkenine ata.
15. STOK formunun aktif kaydının 4 alanını VFIYAT değişkenine ata.
16. Ekranı temizle.
17. Ekranın 1 nci koordinatına VURKOD değişkeninin içeriğini yaz.
18. Ekranın 17 nci koordinatına VURAD değişkeninin içeriğini yaz.
19. Ekranın 49 ncu koordinatına VMIKTAR değişkeninin içeriğini yaz.
20. Ekranın 65 nci koordinatına VFIYAT değişkeninin içeriğini yaz.
21. Ekranın 81 nci koordinatında <<Devam>> mesajını yaz.
22. Tuş takımından girilen 1 karakterlik veriyi TUS değişkenine aktar.
23. Ekranı temizle.
24. Ekranın 1 nci koordinatında Miktar: mesajını yaz.
25. Tuş takımından veya lazer tarama tuşundan girilen veriyi VMIKTAR değişkenine aktar.
26. 25 nci satırda girilen VMIKTAR değişkeninin karakter uzunluğunu UZ değişkenine ata.
27. UZ değişkeninin içeriğini 0 (Sıfır) ile karşılaştır.
28. 27.satırda yapılan karşılaştırmada sonuç doğru ise programın işleyişini 23 ncü satıra yönlendir.
29. SATIS formunu aktif hale getir.
30. YSIPNO değişkeni içeriğini SIPNO değişkenine aktar.
31. VURKOD değişkeni içeriğini URKOD değişkenine aktar.
32. VURAD değişkeni içeriğini URAD değişkenine aktar.
33. VMIKTAR değişkeni içeriğini URMIKT değişkenine aktar.
34. VFIYAT değişkeni içeriğini URFIYAT değişkenine aktar.
35. SATIS formu değişkenleri içeriğini SATIS formunun sonuna kaydet.
36. Programın işleyişini 1 nci satıra yönlendir.
37. Ekranı temizle.
38. 500 milisaniye süre ile 600 hz frekansında ses çıkar.
39. Ekranın **33** ncü koordinatında Urun bulunamadı mesajını yaz.
40. Tuş takımından girilen 1 karakterlik veriyi TUS değişkenine aktar.
41. Programın işleyişini 1 nci satıra yönlendir.

SAYIM procedure'ünün **SY_ISLEM** Macro'sunun programlanmış halinin görüntüsü aşağıdaki gibidir.



NO	CMD	OPERAND1	OPERAND2
1	CLS		
2	MSG	"Barkod No: "	17
3	INP	VURKOD	
4	LEN	VURKOD	UZ
5	CMP	UZ	0
6	JMP	EQ	1
7	SEL	STOK	
8	GTO	1	
9	SRH	1	VURKOD
10	JMP	NE	:URYOK
11	GET	2	VURAD
12	CLS		
13	MSG	VURKOD	1
14	MSG	VURAD	17
15	MSG	"Miktar: "	49
16	INP	VMIKTAR	
17	LEN	VMIKTAR	UZ
18	CMP	UZ	0
19	JMP	EQ	12
20	SEL	SAYIM	

21	MOV	VURKOD	URKOD
22	MOV	VURAD	URAD
23	MOV	VMIKTAR	URMIKT
24	NRC		
25	JMP	UC	1
26	:URYOK		
26	CLS		
27	BEP	500	600
28	MSG	"Urun Bulunamadi."	33
29	INX	TUS	
30	JMP	UC	1

SY_ISLEM Macro satırlarının Açıklaması

1. Ekran Temizle.

2. Ekranın 1 nci koordinatında (1 nci satır,1 nci sütun) Siparis No: yaz.

3. Ekranın 12 nci koordinatında (1 nci satır,12 nci sütun) YSIPNO değişkeninin içeriğini yaz.

4. Ekranın 17 nci koordinatında (2 nci satır,1 nci sütun) Barkod No: yaz.

5. Tuş takımından veya lazer tarama tuşundan girilen veriyi VURKOD değişkenine aktar.

6. 5 nci satırda okunan VURKOD değişkeninin karakter uzunluğunu UZ değişkenine ata.

7. UZ değişkeninin içeriğini 0 (Sıfır) ile karşılaştır.

8. 7.satırda yapılan karşılaştırmada sonuç doğru ise programın işleyişini 4. satıra yönlendir.

9. STOK formunu aktif hale getir.

10. STOK formunun 1 nci kaydını aktif kayıt yap.

11. STOK formunun 1 nci alanında VURKOD değişkenin içeriğini ara.

12. 11.satırda yapılan aramada aranılan değer bulunamamışsa programın işleyişini URYOK etiketine (37.satır) yönlendir.

13. STOK formunun aktif kaydının 2 alanını VURAD değişkenine ata.

14. STOK formunun aktif kaydının 3 alanını VMIKTAR değişkenine ata.

15. STOK formunun aktif kaydının 4 alanını VFIYAT değişkenine ata.

16. Ekranı temizle.

17. Ekranın 1 nci koordinatına VURKOD değişkeninin içeriğini yaz.

18. Ekranın 17 nci koordinatına VURAD değişkeninin içeriğini yaz.

19. Ekranın 49 ncu koordinatına VMIKTAR değişkeninin içeriğini yaz.

20. Ekranın 65 nci koordinatına VFIYAT değişkeninin içeriğini yaz.

21. Ekranın 81 nci koordinatında <<Devam>> mesajını yaz.

22. Tuş takımından girilen 1 karakterlik veriyi TUS değişkenine aktar.

23. Ekranı temizle.

24. Ekranın 1 nci koordinatında Miktar: mesajını yaz.

25. Tuş takımından veya lazer tarama tuşundan girilen veriyi VMIKTAR değişkenine aktar.

26. 25 nci satırda girilen VMIKTAR değişkeninin karakter uzunluğunu UZ değişkenine ata.

27. UZ değişkeninin içeriğini 0 (Sıfır) ile karşılaştır.

28. 27.satırda yapılan karşılaştırmada sonuç doğru ise programın işleyişini 23 ncü satıra yönlendir.

29. SATIS formunu aktif hale getir.

30. YSIPNO değişkeni içeriğini SIPNO değişkenine aktar.

31. VURKOD değişkeni içeriğini URKOD değişkenine aktar.

32. VURAD değişkeni içeriğini URAD değişkenine aktar.

33. VMIKTAR değişkeni içeriğini URMIKT değişkenine aktar.

34. VFIYAT değişkeni içeriğini URFIYAT değişkenine aktar.

35. SATIS formu değişkenleri içeriğini SATIS formunun sonuna kaydet.

36. Programın işleyişini 1 nci satıra yönlendir.

37. Ekranı temizle.

38. 500 milisaniye süre ile 600 hz frekansında ses çıkar.

39. Ekranın **33** ncü koordinatında Urun bulunamadı mesajını yaz.

40. Tuş takımından girilen 1 karakterlik veriyi TUS değişkenine aktar.

41. Programın işleyişini 1 nci satıra yönlendir.

7. Programı (Task) Derleme

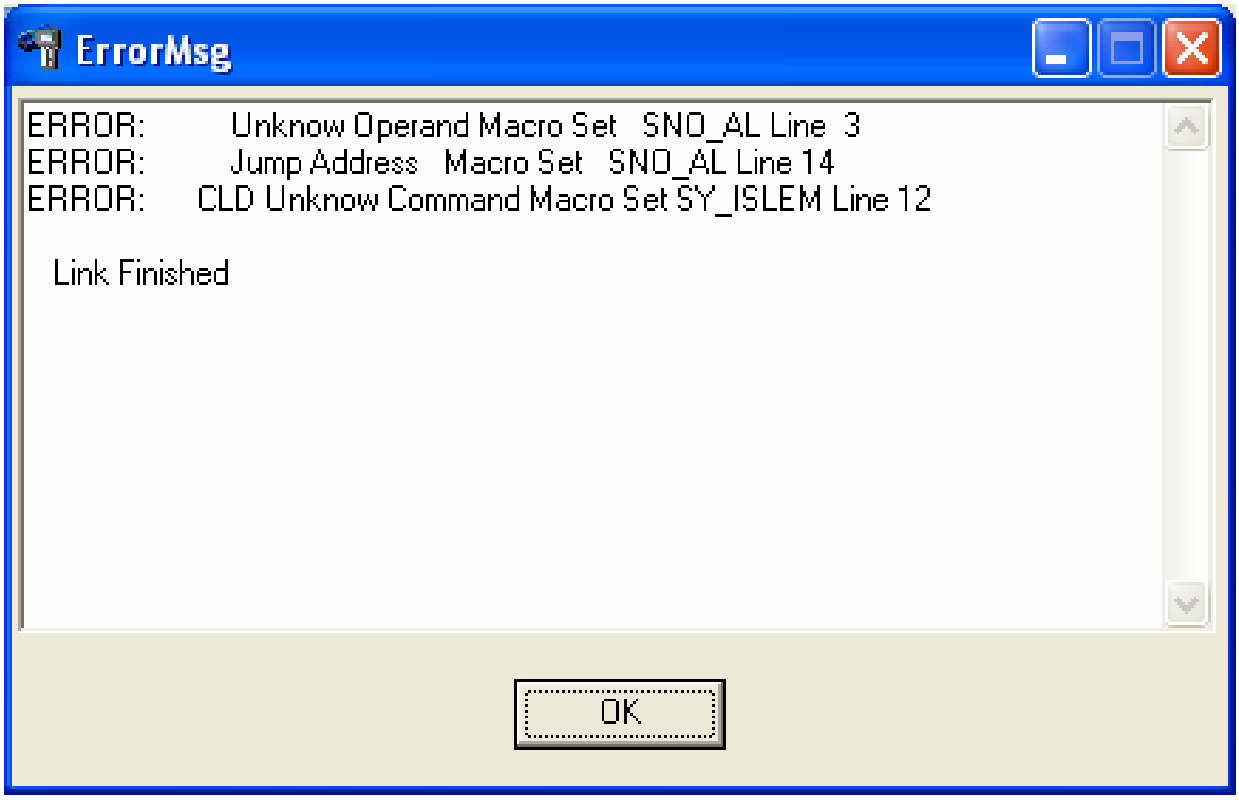
Yukarıdaki adımlarda hazırlanan PDL20 programının derlenerek (Link Task) yazım hatalarının bulunması ve gerekli düzeltmelerin yapılması sağlanmalıdır. Program derlemek için Task Editör ekranında bulunan ve aşağıdaki şekilde görülen Link Task düğmesi kullanılmalıdır.

Not: Program T tipinde bir forma sahip ise formun dosya hali derleme işleminden önce hazır olmalıdır. T tipindeki bir formun dosya halinin hazırlanması hakkındaki ayrıntılı bilgiyi sayfa 46'da bulabilirsiniz.

Programın (Task) Derleme işlemi sırasında herhangi bir hata ile karşılaşılmaması durumunda aşağıdaki şekilde de görüldüğü gibi ErrorMsg diyalog kutusunda **"Link Finished"** mesajı alınmaktadır.

Programın (Task) Derleme işlemi sırasında herhangi bir hata ile karşılaşılması durumunda aşağıdaki şekilde de görüldüğü gibi Link File diyalog kutusunda **"Link Faile, Please Try Again"** uyarısı alınmaktadır.

Link Faile diyalog kutusu onaylandığında aşağıdaki şekilde de görüldüğü gibi yazım hatalarını içeren Macro'nun adı, hatanın oluştuğu satır numarası ve yazım hatasının ne olduğunu belirten uyarıyı içeren bir diyalog kutusu görüntülenecektir.



Yukarıdaki hata mesajlarında;

— SNO_AL Macro'sunda 3 ncü satırında komutun işleyişine göre uygun olmayan eksik bir değişken bulunduğu,

— SNO_AL Macro'sunda 14 ncü satırında JMP komutunun yönlendirilmesi gereken program bölümünün belirtilmediği,

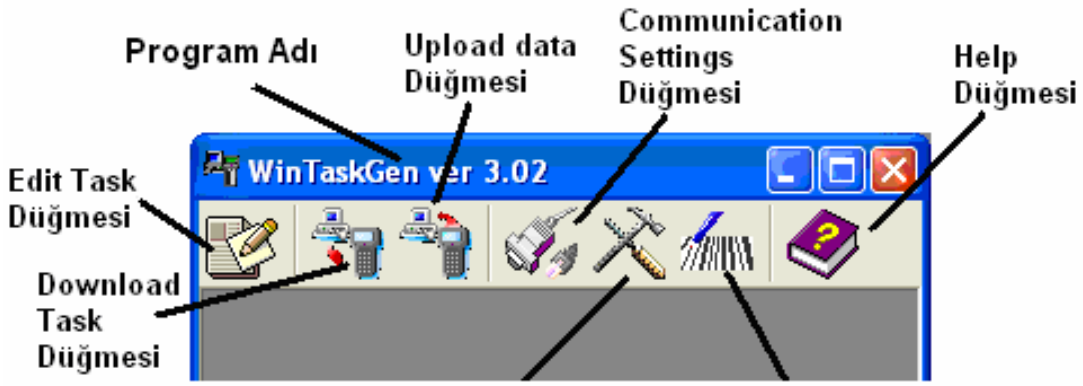
— SY_ISLEM Macro'sunda 12 satırda CLD olarak belirtilen komutun yanlış yazıldığını,

Belirtilen hataların bulunduğu anlaşılmaktadır.

Not: Yukarıda belirtilen Macro'ların uygun satırlarındaki hatalar giderildikten sonra program (Task) tekrar derlenmelidir. Derleme işlemi tüm hatalar giderilene kadar tekrar edilmelidir.

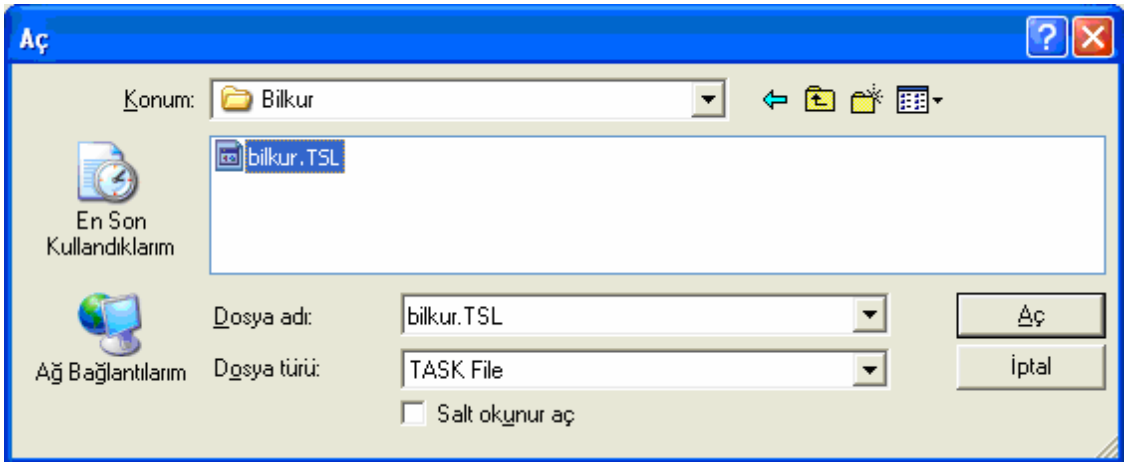
8. Programın (Task) PDL20'ye yüklenmesi

Hatasız olarak hazırlanmış programın (Task) PDL20'ye yüklenmesi için WinTaskGen ekranında bulunan ve aşağıdaki şekilde de görüldüğü gibi Download Task düğmesi kullanılmalıdır.



Not: Bu işlem sırasında PDL20 aşağıdaki şekilde de görüldüğü gibi ana ekranda olmalıdır.

Download Task düğmesi tıklandığında aşağıdaki şekilde de görüldüğü gibi açılan **"Aç"** diyalog kutusunda programın (Task) kayıtlı bulunduğu konuma ulaşıp oluşturulan programın derlenmiş hali (TSL uzantılı dosya) seçilmelidir.



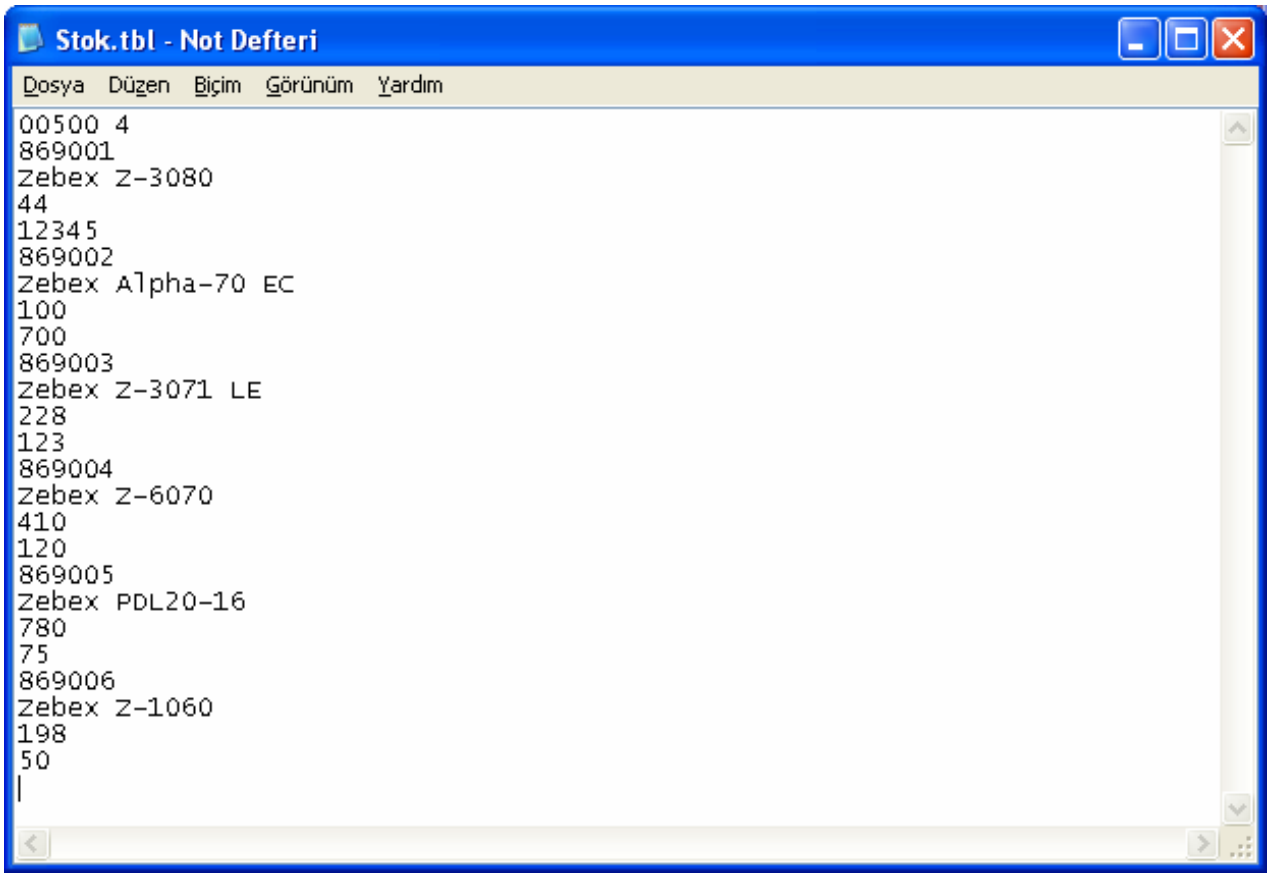
Yukarıdaki diyalog kutusunda doğru dosya seçildikten sonra **"Aç"** düğmesi tıklanarak aşağıdaki şekilde de görüldüğü gibi yükleme işlemi başlatılmış olacaktır.

Yukarıdaki şekilde görülen bölümler tek-tek yüklendikten sonra yüklemenin bittiğini gösteren ekran görüntüsü aşağıdaki şekilde ki gibidir.

STOK Formunun oluşturulması

Örnek program (Task) ile birlikte PDL20'ye yüklenmesi gereken STOK formunun (**Not:** STOK formunun tipi **T** olmalıdır.) dosya halinin yapısı şu şekilde olmalıdır.

— Dosya programın kayıtlı olduğu konumda olmalı,
— Dosya adı STOK.TBL olmalı,
— Dosyanın içeriğinde her alan bir satır oluşturmalıdır.
Aşağıda STOK.TBL dosyasının örnek hali bulunmaktadır.



Yukarıdaki şekilde STOK.TBL dosyasının içeriği görülmektedir. STOK.TBL dosyasının ilk satırında bulunan **00500 4** metni; STOK.TBL dosyasında toplam 500 (**00500**) farklı ürünün bilgilerinin bulunduğunu ve her ürünün 4 farklı bilgisini içerdiği anlamına gelmektedir.

STOK.TBL dosyası içeriği aşağıdaki şekilde de görülen STOK formunun içeriği ve sırasına uygun bir şekilde hazırlanmıştır.

Form:STOK

No	Name	M	Min	Max	Input Prompting	Data check Editing -> ndaulcp*	I
1	URKOD	B	01	13			
2	URAD	B	01	32			
3	FIYAT	B	01	10			
4	MIKTAR	B	01	05			

Program Klasörü ve içeriği

bilkur.TSK : WinTaskGen programı ile oluşturulan ve programın (Task) derlenmemiş halini içeren dosyadır.
bilkur.TSL : Programın derlenmiş ve PDL20'ye yüklenilecek olan halini içeren dosyadır. STOK.TBL dosyası bilgilerini de içeren dosyadır.
bilkur.LBL : Programın işleyişinde kullanılan ve herhangi bir konumu belirten etiket satırlarının konumlarını saklayan dosyadır.
TASKLINK.ERR : Programın derlenmesi sırasında oluşan hata mesajlarını içeren dosyadır. Derleme sırasında hata ile karşılaşılmaz ise Link Finished mesajını içerir.
STOK.TBL : Programın işleyişinde ihtiyaç duyulan ve programın yüklenmesi ile birlikte PDL20'ye yüklenen bilgileri içeren dosyadır.
Comm.SET : PDL20 ile haberleşmenin ayarlarını belirleyen dosyadır.
Upload.SET : PDL20'den bilgisayara alınacak verilerin alınma şeklinin belirlendiği dosyadır.

Comm.SET dosyası içeriği

COM1 : PDL20'nin bilgisayara bağlandığı haberleşme portu (Seçenekler: COM1, COM2, COM3 veya COM4)
9600 : Cihazın bağlantı hızı (Seçenekler: 300, 1200, 2400, 4800, 9600 veya 19200)
NONE : Eşlik (Seçenekleri None, Odd, Even, Mark veya Space)
8 : Veri bit sayısı (Seçenekler: 8 veya 7)
2 : Dur Bit sayısı (Seçenekler: 1 veya 2)

Not: Aynı ayarların PDL20'de de olması gereklidir.

Upload.SET dosyası içeriği

Space : Alanların nasıl ayrılacağını belirler. (Seçenekler: noktalı virgül (;) Virgül (**,**) , boşluk ( ), veya ayrım olmayabilir.
CR/LF : Kayıtların nasıl ayrılacağını belirler.(Seçenekler: **None, CR/LF, CR** ve **LF** 'dir.)
YES : Bilgi alınan dosya hakkındaki rapor satırının olup olmayacağını belirler. (Seçenekler: **Yes** ve **No'**dur.)
Enable : Aktarılan bilgilerin ekranda görünüp-görünmemesini belirler.(Seçenekleri **Enable** ve **Disable**'dir.)
No : Cihazdan bilgi alındıktan sonra cihazdaki bilginin silinmesini veya silinmemesini belirler. (Seçenekler: **Yes** ve **No'**dur.)

PDL20'den Bilgisayara Veri Alma

1.MS-DOS modunda veri alımı:

Bu yöntemde **UPL.EXE** adlı bir dosya aracılığı ile aktarım yapılabilmektedir. Bu yöntem ile PDL20'ye yüklü belirlenen veritabanındaki (form) bilgiler belirtilen porttan alınarak belirtilen dosyaya otomatik olarak kayıt edilir. Bu dosya uzantısı serbest olmakla beraber formatı text tabanlı bir dosya olacaktır. Bu yöntemde aktarım yapılırken bilgilerin alınma şekli **UPLOAD.SET** adlı bir dosyada belirtilmiş olması gerekir.
Kullanım şekli
UPL DOSYA-ADI COMPORT FORMNO
DOSYA-ADI : Cihazdan alınan bilgilerin kaydedildiği dosyaya verilecek isim.
COM PORT : Cihazın bilgi alırken kullanacağı haberleşme portu (COM1=**-C1**,COM2=**-C2**,COM3=**-C3**,COM4=**-C4** olabilir.)
FORMNO : Cihazdan alınacak bilginin bulunduğu veritabanı (form) numarasıdır.(1~8 arası bir numaradır)
Not:1 numaralı veritabanı FREETASK veritabanıdır.
Örnek: UPL deneme.txt –C1 1

PDL20'ya yüklü 1 numaralı veritabanındaki (form) bilgiler COM1'den okunarak deneme.txt adlı bir dosyaya kaydedilir.

2. WINDOWS Modunda veri alımı:

Windows modunda aktarım **COMM.EXE** adlı bir dosya aracılığı ile yapılabilmektedir. Bu yöntem ile PDL 20-16'ya yüklü belirlenen veritabanındaki (form) bilgiler belirtilen porttan alınarak belirtilen dosyaya otomatik olarak kayıt edilir. Bu dosya uzantısı serbest olmakla beraber formatı text tabanlı bir dosya olacaktır.

Kullanım şekli

Comm -u DOSYA_ADI COMPORT -o/-a FORMNO
DOSYA-ADI : PDL20'den alınan bilgilerin kaydedildiği dosyaya verilecek isim.
Com port : PDL20'den bilgi alınırken kullanılan haberleşme portu (COM1=**-C1**,COM2=**-C2**,COM3=**-C3**,COM4=**-C4** olabilir.)
-a : Dosyayı kaydederken eski kayıtları silmeden yeni kayıtları mevcut dosyanın sonuna ekler.
-o : Dosyayı kaydederken eski kayıtları siler, yeni kayıtları yükler.
FORMNO : PDL20'den alınacak bilginin bulunduğu veritabanı (form) numarasıdır.(1~8 arası bir numaradır) Yazılmadığında tüm formlardaki bilgiler alınır.
Not: Aynı anda birden fazla formdan bilgi alınabilir.
Örnek: Comm -u deneme. txt -o 1,2
PDL20'ye yüklü 1,2 no'lu form bilgiler COM1'den okunarak deneme.txt adlı bir dosyaya kaydedilir.
Örnek: Comm -u deneme. txt -o
PDL20'ye yüklü tüm form bilgiler COM1'den okunarak deneme.txt adlı bir dosyaya kaydedilir.

Bilgisayardan PDL20'ye program yükleme

1. MS-DOS Modunda aktarım

Bu işlem DOWNL.EXE adlı bir dosya aracılığı ile yapılabilmektedir. PDL20 için Wintaskgen programı ile hazırlanmış programının (Task) derlenmiş hali TSL uzantılı dosya PDL20'ye gönderilir.

Kullanım şekli : Downl DOSYA_ADI

DOSYA_ADI: PDL20'ye yüklenecek derlenmiş TSL uzantılı olan dosya
Örnek : Downl bilkur.tsl
(Bilkur.Tsl dosyasını PDL20'ye yükler.)

2. WINDOWS Modunda aktarım

Bu işlem COMM.EXE adlı bir dosya aracılığı ile yapılabilmektedir. PDL20 için Wintaskgen programı ile hazırlanmış programı (Task) PDL20'ya gönderilir.

Kullanım şekli : Comm –d DOSYA_ADI COMPORT

DOSYA_ADI : PDL20'ye yüklenecek derlenmiş hali TSL uzantılı olan dosya
COM PORT : PDL20'ye aktarım yapılırken kullanılan haberleşme portu (COM1=**-C1**,COM2=**-C2**,COM3=**-C3**,COM4=**-C4** olabilir.)
Not: Yüklemenin yapıldığı dizinde (klasör) COMM.SET dosyası mevcut ise –C1 parametresi kullanılmayabilir.
Örnek-1: Comm –d bilkur.tsl –c1
(Bilkur.Tsl dosyasını Com1'den PDL20'ye Yükler.)
Örnek-2: Comm –d bilkur.tsl
(Bilkur.Tsl dosyasını Comm.Set dosyasında belirtilen port aracılığı ile PDL20'ye Yükler.)

Not: Bilgisayardan PDL20'ye veya PDL20'den bilgisayara aktarım için gerekli UPL.EXE ve DOWNL.EXE DosTaskGen programının, COMM.EXE ise WinTaskGen programlarının dosyalarıdır. DosTaskGen ve WinTaskGen programlarını PDL20 ile birlikte verilen CD içerisinde veya **www.bilkur.com.tr** adresinde bulabilirsiniz.

PROGRAMLAMADA KULLANILAN TUŞ KODLARI

SIRA	TUŞ	KAREKTER
1	LEFT	L
2	RIGTH	R
3	M1	M
4	M2	N
5	Sol (Left) Enter	E
6	Sağ (Rigth) Enter	F
7	Tarama (Scan)Tuşu	S
8	F1	a
9	F2	b
10	F3	c
11	F4	d
12	F5	e
13	F6	f
14	1	1
15	2	2
16	3	3
17	4	4
18	5	5
19	6	6
20	7	7
21	8	8
22	9	9
23	0	0

Not: Yukarıda verilen tuş kodları sadece **INX** komutu ile birlikte kullanılabilir.

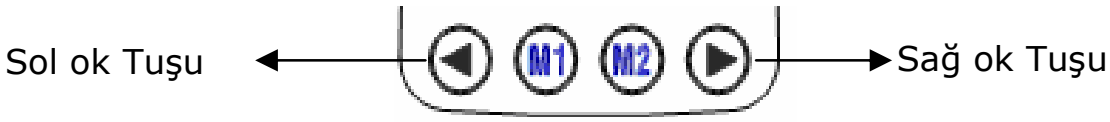
TANIMLANMADAN KULLANILABİLEN DEĞİŞKENLER

Değişken	Kullanım Şekli
*	Genel karakter değişkeni
DATE	Tarih değişkeni (kısaltılmış düzende **03**05**05** gibi)
DATES	Tarih değişkeni (Normal düzende **03**05**2005** gibi)
TIME	Zaman değişkeni (kısaltılmış düzende **12**:34 gibi)
TIMES	Zaman değişkeni (Normal düzende **12**:34:**08** gibi)
RECORDP	Aktif kayıt numarasını içerir.
RECORDL	Son kayıt numarasını içerir.

AYARLARININ SIFIRLANMASI (RESETLENMESİ)

Herhangi bir durumda PDL20'nin ayarlarının fabrika ayarlarına (standart ayarlar) çevrilmesi işlemine sıfırlama (resetleme) denilmektedir.

1) PDL20'yi PW tuşundan kapatın.



2) Yukarıdaki şekilde de görülen sol ve sağ ok tuşlarını aynı anda basılı tutarak PW tuşundan açın. Açılış ekranı aşağıdaki gibidir.

3) Yukarıdaki şekilde de görülen M2 tuşu ile M2 Sel Y/N? **N** sorusundaki N yi **Y** haline çevirin ve herhangi bir Enter tuşuna basın. Enter tuşuna basılınca ekran görüntüsü aşağıdaki gibidir.

4) PDL20'nin belleğinde herhangi bir problem yok ise yukarıdaki şekilde görülen 3 satırdaki rakam 1 den 2048'e kadar değişecektir. 2048 olduğunda işlem tamamlanarak ana ekrana geçiş olacaktır.

Not: Bu dokümandaki bilgi ve yazım hatalarından Bilkur Bilgisayar sorumlu değildir.