**DEĞİŞKENLER (VARIABLES)**

Değişkenler program çalıştığı sürece hafızada değiştirilebilir verilerin tutulmasını sağlar. Değer ataması değişken tanımlanırken veya program akışı esnasında yapılabilir.

Değişken tanımlamak için DATA ifadesi kullanılır. Tanımlanırken VALUE ifadesi ile değer atanabilir. Program içerisinde dinamik olarak değer atamak için eşittir “=” kullanılır.

Değişkenin geçerliliği tanımlandığı blok ile sınırlıdır. Tanımlandığı blok dışına çıkıp tekrar girilirse değişkenin değeri sıfırlanır.

STATICS: Prosedürlerde durağan geçerliliği olan nesneler oluşturmak için kullanılırlar. Data ifadesi ile tanımlanan değişkenlerin ömürleri tanımlandıkları bloğa bağlıdır. STATICS ifadesi ile değişkene atanan değer tanımlanan bloğa bağlı kalmadan korunabilir. Program bloğu tekrar çalıştırıldığında bloktan çıkmadan önce atanan değer bulunur.

CLASS-DATA: Sınıflarda durağan geçerliliği olan nesneler oluşturmak için kullanılırlar.

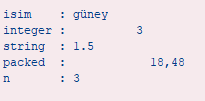
PARAMETERS: Seçim ekranındaki bir giriş alanına tayin edilir.

SELECT-OPTIONS: Seçim ekranındaki bir giriş ekranına tayin edilen seçim (internal) tablosudur.

RANGES: SELECT-OPTIONS yapısında olan bir giriş ekranına tayin edilmeyen seçim (internal) tablosudur.

Örnek: Veri tipi tanımlamayı, değişken tanımlamayı ve değişkene değer atanmasını gösteren örnek.

TYPES ty\_isim TYPE c LENGTH 20.  
DATA: gv\_isim    TYPE ty\_isim VALUE ‘güney’,  
gv\_integer TYPE i VALUE 3,  
gv\_string  TYPE string,  
gv\_packed  TYPE p DECIMALS 2,  
gv\_n       TYPE n.  
gv\_string = gv\_integer / 2.  
gv\_packed = ‘18.48’.  
gv\_n      = ‘3’.  
WRITE:/ ‘isim    :’, gv\_isim,  
/ ‘integer :’, gv\_integer,  
/ ‘string  :’, gv\_string,  
/ ‘packed  :’, gv\_packed,  
/ ‘n       :’, gv\_n.



Örnek: LIKE ifadesi ile başka bir değişkenin tipinde değişken tanımlamayı gösteren örnek.

TYPES ty\_isim    TYPE c length 20.  
DATA: gv\_isim    TYPE ty\_isim,  
gv\_soyisim LIKE gv\_isim.  
gv\_isim    = ‘güney’.  
gv\_soyisim = ‘Rencuz’.  
WRITE: gv\_isim, gv\_soyisim.



**Yapısal Değişkenler (Structured Variables)**

Birden fazla değişkeni tek bir yapı altında toplayarak kullanmak mümkündür. Değişkenler tanımlanan yapının bileşenleri olur. Değişkenler farklı veri tiplerinde olabilir. Yapı içerisindeki değişkenlere yapı isminden sonra tire (-) işareti ekleyerek erişilir.

Yapılar birkaç şekilde tanımlanabilirler.

· Daha önce tanımlanan değişkenler **LIKE** kelimesi kullanılarak tip olarak gösterilebilir. Daha önce tanımlanan tipler veya ön tanımlı tipler **TYPE** anahtar kelimesi kullanılarak tanımlanabilir. Bu tür kullanımda**DATA** anahtar kelimesinden sonra**BEGIN OF** ve **END OF** anahtar kelimeleri kullanılarak yapısal değişken oluşturulur.

· **LIKE** kullanılarak daha önce tanımlanan bir yapısal değişkenden veya **TYPE** anahtar kelimesi kullanılarak daha önce tanımlanan yapısal bir veri tipinden oluşturulabilirler.

Örnek: LIKE ve TYPE anahtar kelimeleri kullanılarak oluşturulan yapısal değişken.

DATA : gv\_isim(20) TYPE c,  
gv\_id       TYPE n,  
gv\_kilo     TYPE p DECIMALS 2,  
gv\_boy      TYPE i.  
DATA : BEGIN OF ls\_aday,  
isim      LIKE gv\_isim,  
id        LIKE gv\_id,  
kilo      LIKE gv\_kilo,  
boy       LIKE gv\_boy,  
adres(30) TYPE c,  
tarih     TYPE sy-datum,  
END OF ls\_aday.  
ls\_aday-isim  = ‘güney’.  
ls\_aday-id    = ‘1’.  
ls\_aday-kilo  = ‘89.20’.  
ls\_aday-boy   = 193.  
ls\_aday-adres = ‘Abap sokak’.  
ls\_aday-tarih = sy-datum.  
WRITE: / ls\_aday-isim, ls\_aday-id, ls\_aday-kilo, ls\_aday-boy, ls\_aday-adres, ls\_aday-tarih.



Örnek: LIKE ve TYPE anahtar kelimesi kullanılarak yapısal veri tipi ve yapısal veri değişkeni kullanılarak oluşturulan yapısal değişken.

TYPES: ty\_isim(20) TYPE c,  
ty\_id       TYPE n,  
ty\_kilo     TYPE p DECIMALS 2,  
ty\_boy      TYPE i.  
TYPES: BEGIN OF ty\_aday,  
isim      TYPE ty\_isim,  
id        TYPE ty\_id,  
kilo      TYPE ty\_kilo,  
boy       TYPE ty\_boy,  
adres(30) TYPE c,  
tarih     TYPE sy-datum,  
END OF ty\_aday.  
DATA: ls\_aday TYPE ty\_aday.  
DATA: ls\_yeni\_aday LIKE ls\_aday.  
ls\_aday-isim  = ‘güney’.  
ls\_aday-id    = ‘1’.  
ls\_aday-kilo  = ‘89.20’.  
ls\_aday-boy   = 193.  
ls\_aday-adres = ‘Abap sokak’.  
ls\_aday-tarih = sy-datum.  
WRITE: / ls\_aday-isim, ls\_aday-id, ls\_aday-kilo, ls\_aday-boy, ls\_aday-adres, ls\_aday-tarih.  
ls\_yeni\_aday-isim  = ‘Mehmet’.  
ls\_yeni\_aday-id    = ‘2’.  
ls\_yeni\_aday-kilo  = ’74’.  
ls\_yeni\_aday-boy   = 176.  
ls\_yeni\_aday-adres = ‘Sap sokak’.  
ls\_yeni\_aday-tarih = sy-datum.  
WRITE: / ls\_yeni\_aday-isim, ls\_yeni\_aday-id, ls\_yeni\_aday-kilo, ls\_yeni\_aday-boy, ls\_yeni\_aday-adres, ls\_yeni\_aday-tarih.



**CLEAR**

Veri nesnelerinin değerini başlangıç değerine getirir. Kullanımı aşağıdaki gibidir.

CLEAR <veri\_nesnesi>.

Farklı veri nesneleri için aşağıdaki şekilde çalışır.

· **Basit ABAP tiplerine sahip veri nesneleri:** Başlangıç değerine döndürür.

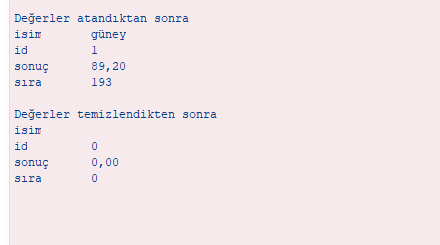
· **Referans tiplerine sahip veri nesneleri:** Referans değişkenini başlangıç değerine döndürür. Bir referansa atanmamış haline.

· **Structure tipleri:** Structure içerisindeki bütün alanların değerlerini başlangıç değerlerine getirir.

· **İnternal tablolar:** İnternal tablonun bütün kayıtlarını siler.

Örnek: Basit veri tipleri için CLEAR ifadesi kullanımı.

DATA: gv\_isim  TYPE c LENGTH 20 VALUE ‘güney’,  
gv\_id    TYPE i VALUE ‘1’,  
gv\_sonuc TYPE p DECIMALS 2 VALUE ‘89.20’,  
gv\_sira  TYPE i VALUE 193.  
WRITE ‘Değerler atandıktan sonra’.  
WRITE:/ ‘isim’,  12 gv\_isim  LEFT-JUSTIFIED,  
/ ‘id’,    12 gv\_id    LEFT-JUSTIFIED,  
/ ‘sonuç’, 12 gv\_sonuc LEFT-JUSTIFIED,  
/ ‘sıra’,  12 gv\_sira  LEFT-JUSTIFIED.  
CLEAR: gv\_isim, gv\_id, gv\_sonuc, gv\_sira.  
SKIP.  
WRITE / ‘Değerler temizlendikten sonra’.  
WRITE:/ ‘isim’,  12 gv\_isim  LEFT-JUSTIFIED,  
/ ‘id’,    12 gv\_id    LEFT-JUSTIFIED,  
/ ‘sonuç’, 12 gv\_sonuc LEFT-JUSTIFIED,  
/ ‘sıra’,  12 gv\_sira  LEFT-JUSTIFIED.



**DEĞER ATAMALARI**

**MOVE**

Kaynak veri nesnesindeki değeri hedef veri nesnesine taşır. Kaynak veri nesnesi ve hedef veri nesnesi tipleri birbirine uygun veya hedef veri nesnesi içeriğinin kaynak veri nesnesi içeriğine çevrilebilir olması gerekir. Kullanımı aşağıdaki şekildedir.

MOVE {kaynak\_veri\_nesnesi TO hedef\_veri\_nesnesi}

| {kaynak\_veri\_nesnesi TO hedef\_veri\_nesnesi}.

**Ekler:**

EXACT: Birbirinden uyumsuz iki veri nesnesi arasında atama yapılacaksa atama yapılacak değerleri kontrol eder. Aşağıdaki kontroller yapılır.

· Atanacak değerin, kaynak veri nesnesinin veri tipi için geçerli olup olmadığı

· Kaynak veri nesnesine atama yapılması değer kaybına neden olup olmadığı

· Değerin, hedef veri nesnesine ataması sırasında yapılan çevrim sonrası geçerli olup olmadığı

**MOVE-CORRESPONDING**

Kaynak structure’ daki alanları hedef structure’ daki alanlara taşır. Sadece kaynak ve hedef yapılarında aynı isme sahip alanlar transfer edilir.

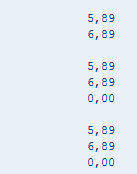
MOVE-CORRESPONDING [EXACT] structure1 TO structure2.

**Ekler:**

EXACT: MOVE komutundaki kullanım ile aynıdır.

Örnek: MOVE ve MOVE-CORRESPONDING anahtar kelimeleri kullanarak değer ataması yapmayı gösteren örnek..

DATA : gv1 TYPE p DECIMALS 2,  
gv2 TYPE p DECIMALS 2.  
TYPES: BEGIN OF gy\_move,  
v1 TYPE p DECIMALS 2,  
v2 TYPE p DECIMALS 2,  
v3 type p decimals 2,  
END OF gy\_move.  
DATA : gs\_move1 TYPE gy\_move,  
gs\_move2 TYPE gy\_move.  
gv1 = ‘5.89’.  
MOVE gv1 TO gv2.  
gv2 = gv2 + 1.  
gs\_move1-v1 = gv1.  
gs\_move1-v2 = gv2.  
MOVE-CORRESPONDING gs\_move1 TO gs\_move2.  
WRITE :/ gv1,  
/ gv2.  
SKIP.  
WRITE :/ gs\_move1-v1,  
/ gs\_move1-v2,  
/ gs\_move1-v3.  
SKIP.  
WRITE :/ gs\_move2-v1,  
/ gs\_move2-v2,  
/ gs\_move2-v3.

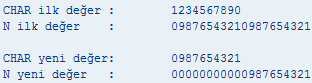


**Değer Atamalarında Çevrimleri**

Farklı veri tipine sahip veri nesneleri birbirine atanırken değer üzerinde çevrimler yapılır.

Örnek: Farklı uzunluktaki veri tipine sahip değişkenlere değer ataması yapılması sırasında oluşan çevrimleri gösteren örnek.

DATA: gv\_char TYPE c LENGTH 10 VALUE ‘1234567890’,  
gv\_n    TYPE n LENGTH 20 VALUE ‘09876543210987654321’.  
WRITE: / ‘CHAR ilk değer :’, 25 gv\_char,  
/ ‘N ilk değer    :’, 25 gv\_n.  
SKIP.  
MOVE gv\_n TO gv\_char.  
WRITE: / ‘CHAR yeni değer:’, 25 gv\_char.  
MOVE gv\_char TO gv\_n.  
WRITE: / ‘N yeni değer   :’, 25 gv\_n.



**ALT DEĞER**

Veri nesnesinin belirli bir pozisyondaki (offset) kısmına ulaşılabilir. Alt bölgeye ulaşmak için pozisyon ve uzunluğu belirtmek yeterlidir.

veri\_nesnesi[+pozisyon][(uzunluk)]

DATA: gv\_metin  TYPE c LENGTH 30,  
gv\_metin1 TYPE c LENGTH 30,  
gv\_metin2 TYPE c LENGTH 30.  
gv\_metin = ‘0123456789 – ABCDEFGHIJ’.  
gv\_metin1 = gv\_metin+0(10).  
gv\_metin2 = gv\_metin+13(10).  
WRITE gv\_metin.  
WRITE / gv\_metin1.  
WRITE / gv\_metin2.

http://iuyanik.com/abaptr/wp-content/uploads/2013/11/img_52754058c5a71.png

**ALAN SEMBOLLERİ (FIELD SYMBOLS)**

Alan sembolleri fiziksel olarak hafızada yer kaplamayan, kendisine atanan nesneyi içeriğine erişimi saylayan veri nesnelerdir. ASSIGN anahtar kelimesi kullanılarak bir veri nesnesine atanırlar. Kullanmak için önce atama yapılması gerekir yoksa hata alınır. Tanımlama esnasında veri tipi tanımlanabilir, veri tipi tanımlanmaz ise atandığı alanın tipini alır.

FIELD-SYMBOLS <fs> [tip\_tanımı].

**Ekler:**

Tip\_tanimi: Tip tanımı için genel veri tipleri veya tanımlanmış veri tipi kullanılabilir.

Genel veri tipi için kullanım aşağıdaki şekildedir.

TYPE genel\_veri\_tipi

Tanımlanmış veri tipi için kullanım aşağıdaki şekildedir. TYPE ile tip, LIKE ile veri nesnesinin tipi kullanılabilir.

{ TYPE {[LINE OF] tanimli\_tip}

| {REF TO tip} }

| { LIKE {[LINE OF] veri\_nesnesi}

| {REF TO veri\_nesnesi} }

Örnek: Genel veri tipi tanımı kullanan bir field symbols örneği.

DATA          gv1   TYPE c LENGTH 10.  
FIELD-SYMBOLS <fs1> TYPE ANY.  
gv1 = ‘degişken 1′.  
ASSIGN gv1 TO <fs1>.  
WRITE <fs1>.

http://iuyanik.com/abaptr/wp-content/uploads/2013/11/img_52754066bee5e.png

Örnek: Bir internal tabloya atanan ve atamadan sonra içerisinden kayıt okuması yapılan field symbols örneği.

TYPES: BEGIN OF gy,  
v1 TYPE c LENGTH 10,  
v2 TYPE c LENGTH 10,  
END OF gy.  
DATA : gt TYPE STANDARD TABLE OF gy,  
gs TYPE gy,  
gv\_key(4) TYPE c VALUE ‘v1′.  
FIELD-SYMBOLS: <fs> TYPE ANY TABLE.  
gs-v1 = ‘güney’.  
gs-v2 = ‘rencuz’.  
APPEND gs TO gt.  
gs-v1 = ‘mehmet’.  
gs-v2 = ‘rencuz’.  
APPEND gs TO gt.  
ASSIGN gt TO <fs>.  
CLEAR gs.  
READ TABLE <fs> WITH KEY (gv\_key) = ‘mehmet’ INTO gs.  
WRITE :/ ‘isim   :’, 10 gs-v1,  
/ ‘soyisim:’, 10 gs-v2.



**DÖNGÜLER**

Döngüler için DO, WHILE, LOOP ENDLOOP ifadeleri kullanılır.

**DO – ENDDO**

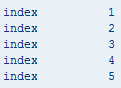
Koşulsuz döngü için kullanılır. Kod bloğunu belirtilen sayı kadar çalışır. Çalıştırma sayısı belirtilmez ise döngü sonsuza kadar çalışır, belirtilen sayıdan önce çıkmak için EXIT ifadesi kullanılabilir.

Kullanımı aşağıdaki şekilde şekildedir.

DO [n TIMES].  
[kod\_blogu]  
ENDDO.

Örnek: Kod bloğunu 5 defa çalıştıran örnek.

DO 5 TIMES.  
WRITE :/ ‘index’, sy-index.  
ENDDO.



Örnek: Döngü sayısı 3 den büyük olduğu zaman döngüden çıkan örnek.

DO.  
IF sy-index > 3.  
EXIT.  
ENDIF.  
WRITE :/ ‘index’, sy-index.  
ENDDO.

http://iuyanik.com/abaptr/wp-content/uploads/2013/11/img_5275408293c91.png

Koşullu döngü için kullanılır. Kod bloğunu mantıksal ifade doğru olduğu sürece çalıştırır. Kod bloğundan çıkmak için EXIT ifadesi de kullanılabilir.

WHILE mantıksal\_ifade.

  [kod\_blogu]

ENDWHILE.

Örnek: Kod bloğunu sy-index değişkeni 3 ten küçük olduğu sürece çalıştıran örnek.

WHILE sy-index < 3.  
WRITE :/ ‘index’, sy-index.  
ENDWHILE.

http://iuyanik.com/abaptr/wp-content/uploads/2013/11/img_5275409391710.png

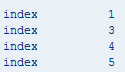
**DÖNGÜ İÇERİSİNDE KULLANILAN ANAHTAR KELİMELER**

**CONTINUE**

Koşulsuz olarak geçerli döngü sonlandırılır, CONTINUE ifadesinden sonraki kod bloğu çalıştırılmaz ve bir sonraki döngü başlar.

Örnek: 5 defa çalışan döngü örneği sy-index değeri 2’ ye ulaştığı zaman, CONTINUE ifadesinden sonraki kodlar çalıştırılmadan bir sonraki döngüye geçilir.

DO 5 TIMES.  
IF sy-index = 2.  
CONTINUE.  
ENDIF.  
WRITE :/ ‘index’, sy-index.  
ENDDO.



**CHECK**

Koşula bağlı olarak geçerli döngü sonlandırılır, bir sonraki döngüye başlar. Eğer koşul yanlış ise CHECK ifadesinden sonraki kod bloğu işlenmeden kontrol bir sonraki döngüye geçer.

Örnek: sy-index değişkeni 5 değerinden küçük olduğu sürece çalışan örnek.

DO 5 TIMES.  
CHECK sy-index < 3.  
WRITE :/ ‘index’, sy-index.  
ENDDO.

http://iuyanik.com/abaptr/wp-content/uploads/2013/11/img_527540a49480f.png

**EXIT**

Koşulsuz olarak geçerli döngüyü sonlandırılır. Program akışı ENDLOOP ifadesinden sonraki satırdan devam eder.

DO 5 TIMES.  
IF sy-index = 3.  
EXIT.  
ENDIF.  
WRITE :/ ‘index’, sy-index.  
ENDDO.

http://iuyanik.com/abaptr/wp-content/uploads/2013/11/img_527540ae0f60a.png

**ABAP Sistem Alanları**

ABAP sistem alanları programın çalışması sırasında otomatik olarak doldurulur. Sistem ve yapılan işlemlerle ilgili durum bilgilerini barındırır. sy-repid dışındaki alanlar SYST structure’ ında tanımlıdır. Program çalıştığında “sy” isimli structure otomatik olarak programa dâhil edilir.

|  |  |  |
| --- | --- | --- |
| **Alan** | **Tipi** | **Tanım** |
| sy-abcde | c(26) | Latin alfabesini içerir. Pozisyon (offset) ve uzunluk ile istenilen harfe erişilebilir. |
| sy-batch | c(1) | Program arka planda çalışıyor ise “X” değerini alır. |
| sy-binpt | c(1) | Batch input oturumu sırasında ve CALL TRANSACTION USING kullanılarak çalıştırılan programlarda “X” değerini alır. |
| sy-calld | c(1) | Program ilk çağrılma esnasında boş karakter içerir. |
| sy-callr | c(8) | Liste çıktısı alırken, çıktı bağlangıç değerini içerir. |
| sy-colno | i | Liste oluşturma sırasındaki geçerli pozisyon. Değer artışı 1’ den başlar. |
| sy-cpage | i | Görüntülenen ekrandaki sayfa numarası. 1’ den başlar. |
| sy-cprog | c(40) | Harici olarak çağrılan prosedürlerde, çağıran programın ismi yoksa programın ismi. |
| sy-cucol | i | Ekrandaki yatay imleç pozisyonu. Değer artışı 2’ den başlar. |
| sy-curow | i | Ekrandaki dikey imleç pozisyonu. Değer artışı 1’ den başlar. |
| sy-datar | c(1) | PAI (Proses after input) Ekrandaki giriş alanlarından birisi değişti ise veya veri transfer edildi ise “X” değerini alır. |
| sy-datlo | d | Kullanıcının yerel tarihi. |
| sy-datum | d | ABAP sisteminin yerel tarihi. |
| sy-dayst | c(1) | Yaz mevsiminde “X” değerini alır. |
| sy-dbcnt | i | SQL ifadelerinde tabloda işlenen kayıt sayısı atanır. |
| sy-dbnam | c(20) | Çalıştırılabilir programlarda bağlanmış mantıksal veritabanı (logical database). |
| sy-dbsys | c(10) | Merkezi veritabanı sistemi. |
| sy-dyngr | c(4) | Ekran grubu. |
| sy-dynnr | c(4) | Ekran numarası. |
| sy-fdayw | b | Fabrika takvimi hafta içi günleri. |
| sy-fdpos | i | Byte tipi ve karakter tipi veri nesnelerinin aratma işlemi sonrası bulunduğu pozisyon. |
| sy-host | c(32) | Uygulama sunucusunun bilgisayar ismi. |
| sy-index | i | Döngü indeksi. |
| sy-langu | c(1) | Tek karakterle ifade edilen dil anahtarı. |
| sy-ldbpg | c(40) | Çalıştırılabilir programlarda, mantıksal veritabanının (logical database) veritabanı programı. |
| sy-lilli | i | Olayı tetiklenen liste satırı. Değer artışı 1’ den başlar ve sayfa başlığını dahil eder. |
| sy-linct | i | Listenin sayfa uzunluğu. |
| sy-linno | i | Listenin satırı. |
| sy-linsz | i | Listenin satır genişliği. |
| sy-lisel | c(255) | Liste olayı tetiklendiğinde imlecin konumlandığı noktanın içeriği. |
| sy-listi | i | Olayı tetiklenen listenin düzeyi. |
| sy-loopc | i | Table control içerisinde gösterilen satır sayısı. |
| sy-lsind | i | Oluşturulan liste düzeyi. |
| sy-macol | i | Liste yazdırırken, sol kenar sütun sayısı. |
| sy-mandt | c(3) | Kullanıcının oturum açtığı üst birim. |
| sy-marow | i | Liste yazdırırken, üst sayfa boşluğunun satır sayısı. |
| sy-modno | i | Harici oturum sayısı. |
| sy-msgid | c(20) | MESSAGE ifadesinden sonra gelen mesaj sınıfı. |
| sy-msgno | n(3) | MESSAGE ifadesinden sonra gelen mesaj numarası. |
| sy-msgty | c(1) | MESSAGE ifadesinde kullanılan mesaj tipi. |
| sy-msgv1 … sy-msgv4 | c(50) | MESSAGE ifadesinden sonra gelen mesaj içeriği. |
| sy-opsys | c(10) | Çalışılan uygulama sunucusunun işletim sistemi. |
| sy-pagno | i | Liste oluşturmadaki geçerli sayfa. |
| sy-pfkey | c(20) | Ekranda kullanılan GUI status. |
| sy-prdsn | c(6) | Liste çıktısı alırken, çıktı kuyruğu (spool) dosyasının ismi. |
| sy-repid | c(40) | ABAP programının ismi. |
| sy-saprl | c(4) | ABAP sisteminin yayınlanma durumu. |
| sy-scols | i | Ekrandaki sütun sayısı. |
| sy-slset | c(14) | Seçim ekranını doldurmak için kullanılan variant. |
| sy-spono | n(10) | Liste yazdırırken, spool numarası ismini içerir. |
| sy-srows | i | Geçerli ekrandaki satır sayısı. |
| sy-staco | i | Olay tarafından tetiklenmiş sistemde görüntülenen ilk sütun sayısı. |
| sy-staro | i | Listenin üzerinde görüntülenen satır sayısı.. |
| sy-stepl | i | Table control’ daki geçerli satır numarası. |
| sy-subrc | i | Çalıştırılan ABAP ifadesinin dönüş değeri. Genel olarak “0” işlem başarılı demektir. |
| sy-sysid | c(8) | ABAP sisteminin ismi. |
| sy-tabix | i | Tablo indeksi. Standart ve sıralı internal tablonun en son erişilen satır numarası. Hash tablolarda 0’ dır. |
| sy-tcode | c(20) | Çalışan işlem kodu. |
| sy-tfill | i | DESCRIBE TABLE, LOOP AT ve REAT TABLE ifadelerinde, internal tablonun adreslenen satır sayısı ile dolar. |
| sy-timlo | t | Kullanıcının yerel zamanı. |
| sy-title | c(70) | Başlık çubuğunda görünen metin. |
| sy-tleng | i | DESCRIBE TABLE, LOOP AT ve REAT TABLE ifadelerinde, internal tablonun adreslenen kayıt uzunluğu ile dolar. |
| sy-tvar0 … sy-tvar9 | c(20) | TOP-OF-PAGE olayında programın metinlerine göre liste ve sütun başlıkları yer imi “&0″ den “&9″ a kadar değişir. |
| sy-tzone | i | Eşgüdümlü Evrensel Zaman’ a (UTC) göre saniye olarak zaman farkı. |
| sy-ucomm | c(70) | PAI olayında tetiklenen fonksiyon kodu. |
| sy-uline | c(255) | Liste çıktıları için 255 uzunluğunda yatay çizgi satırı. |
| sy-uname | c(12) | Sisteme bağlanan kullanıcı kodu. |
| sy-uzeit | t | ABAP sisteminin yerel zamanı. |
| sy-vline | c(1) | Liste gösterimleri için dikey çizgi (|). |
| sy-wtitl | c(1) | REPORT, PROGRAM ve FUNCTION-POOL ifadelerinde eğer NO STANDARD PAGE HEADING kullanıldıysa “N” değeri atanır. |
| sy-zonlo | c(6) | Kullanıcının zaman dilimi. |

Örnek: Bazı ABAP sistem alanlarını ekrana yazdıran örnek.

WRITE:/ ‘ABAP Sistem Alanları’,  
/ ‘Saat  :’, sy-uzeit,  
/ ‘Kulla.:’, sy-uname.

http://iuyanik.com/abaptr/wp-content/uploads/2013/11/img_527540bcc0865.png