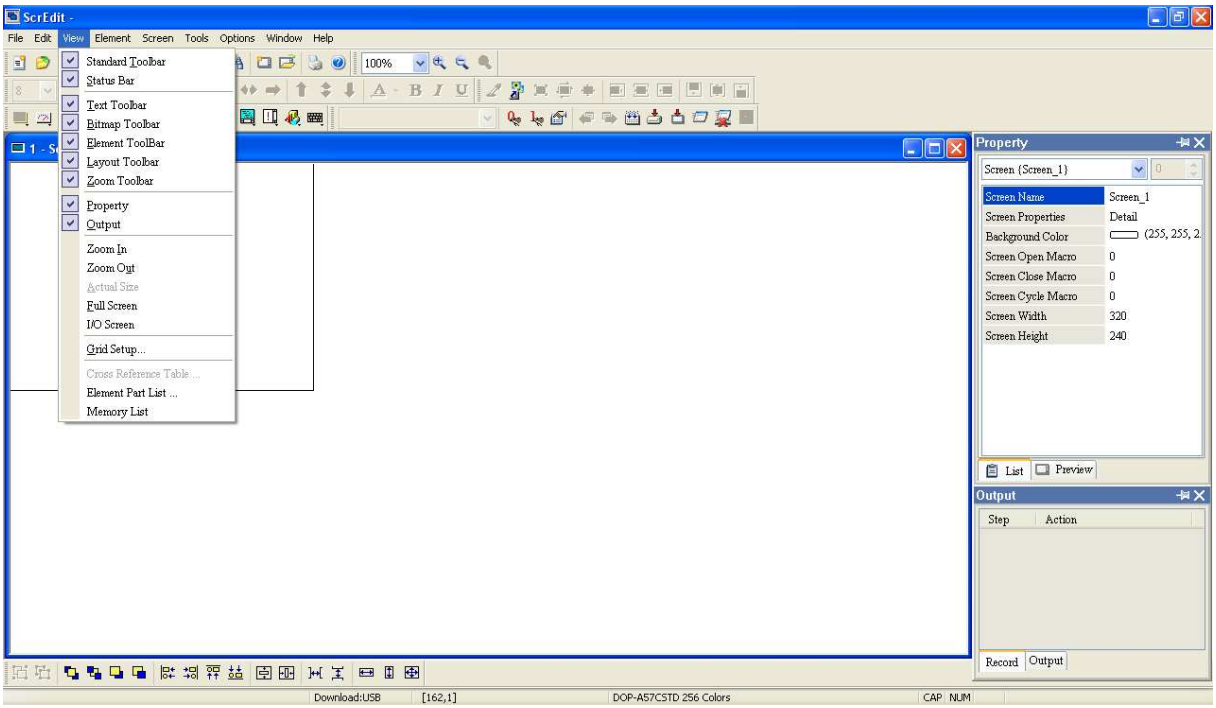
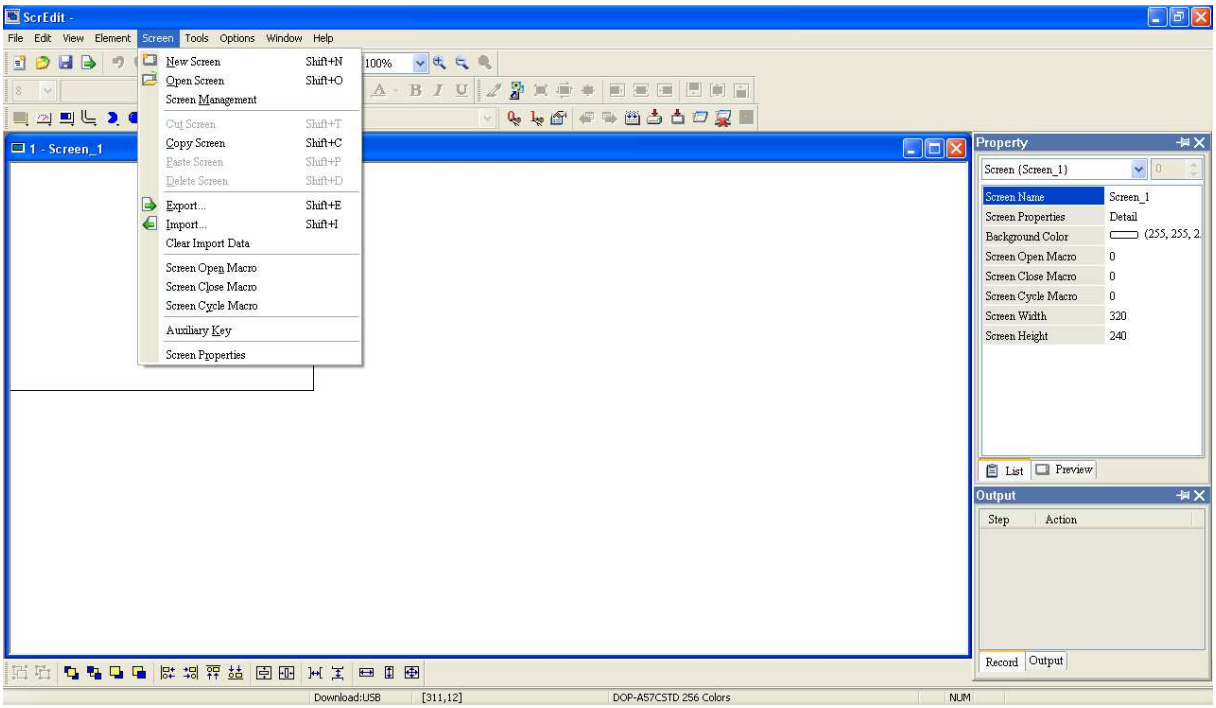
MAKROLAR

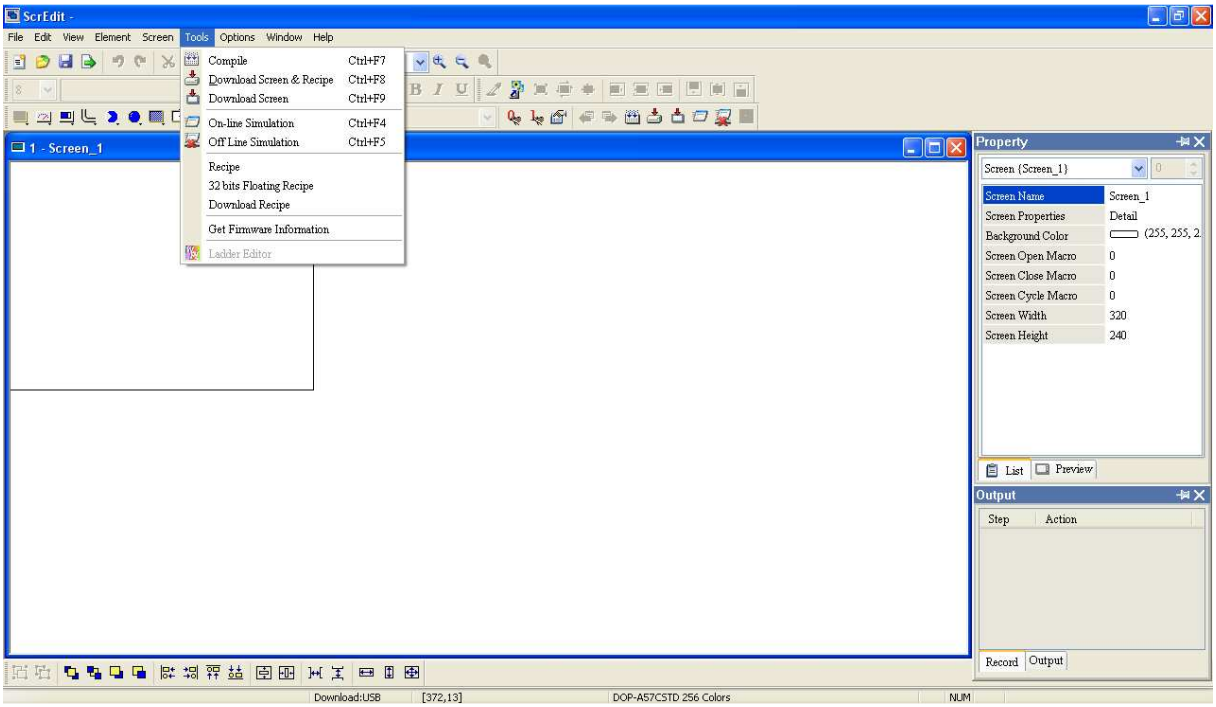
DOP HMI'da karışık hesaplamalar makrolar kullanılarak gerçekleştirilebilir. Ayrıca haberleşme Makroları ile COM port üzerinden farklı sistemlere bağlantı kurulabilir.



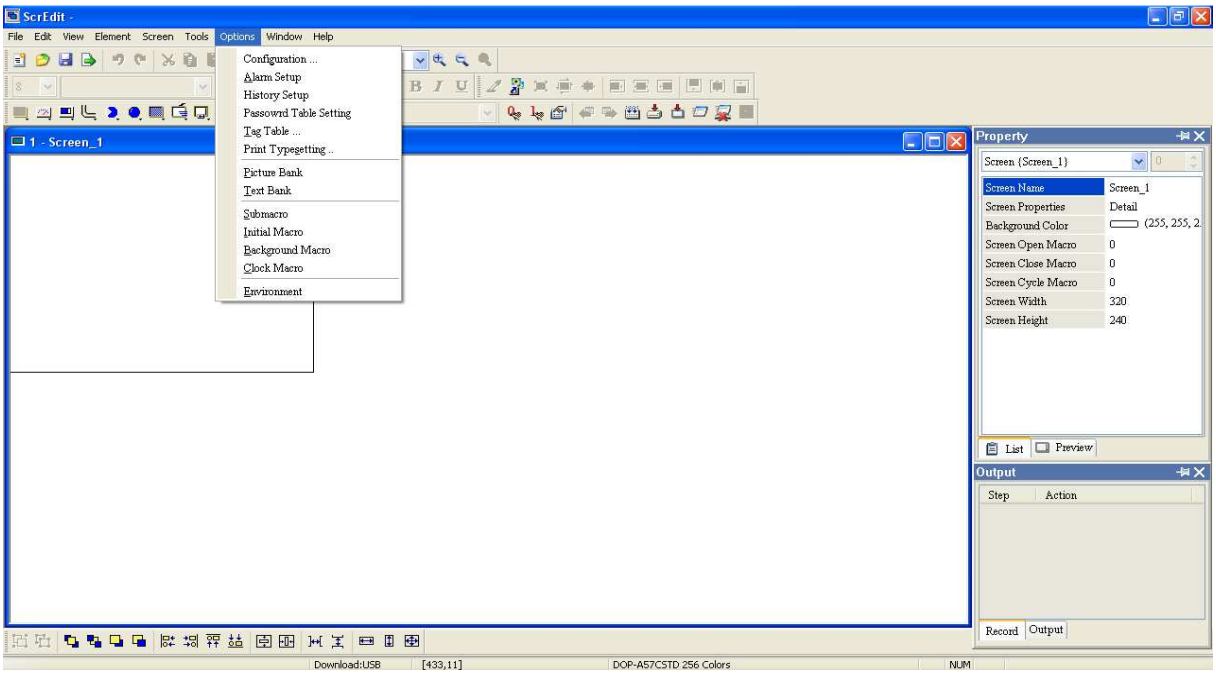
*** View → Full Screen** yapılarak o sayfa ile ilgili makrolar görünebilir.
*** View → I/O Screen** den butonlarda ayarlanan makrolar, ekran ile ilgili makrolar ve nesnelerin yazma ve okuma adresleri görüntülenir.
*** View → Cross Reference Table**'dan DOP programında seçilen herhangi bir nesnenin yazma, okuma ve tetikleme adresi programın başka nerelerinde nesne olarak ve makro olarak kullanıldığı listelenir. Bu sayede aynı datayı yanlışlıkla 2 kere kullanılması engellenir.



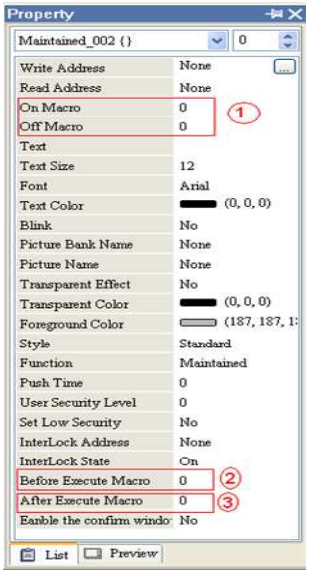
***Screen → Screen Open Macro :** Sayfa ilk açıldığı anda 1 kere çalışır.
***Screen → Screen Close Macro :** Sayfa kapandığı zaman 1 kere çalışır.
***Screen → Screen Cycle Macro :** Sayfa açık olduğu sürece
***Screen→Screen Properties** de ayarlanan sürede bir sürekli çalışan makro. (100 ms – 5 sn). Bu makrolar sağ taraftaki property penceresinden de ayarlanabilir.



***Tools → Offline Simulation** ile PLC ye bağlanılmadan ekran dataları kullanılarak çalışması test edilebilir.



*** Options →Sub macro** ile alt makrolar ayarlanır ve diğer makrolardan çağırılırlar.
*** Options →Initial macro** ile HMI'ya ilk enerji geldiği zaman sadece 1 kere çalışacak olan makro ayarlanır.
*** Options →Background** macro Program çalışırken arkaplanda sürekli çalışacak olan makro ayarlamır.
*** Options →Clock Macro** ile ayarlanan clock macro delay süresinde bir çalışan makro ayarlanır.



1- Edit On/Off Macro → Bir buton basılıp write adreste yazılı olan bit datası ON olduğu zaman ve OFF olduğu zaman çalışan makrolardır. Buton nesneleri oluşturulduktan sonra makrolar ayarlanır. Butonun write adrress de yazan bit datası harici olarak veya makro ile ON yapılrsa Buton ON/OFF makrolar çalışmaz. Bu makrolar sadece buton basıldığı zaman çalışırlar.

2- Before Execute Macro → Bu butona basıldığı zaman ilk önce tanımlanan makro gerçekleşir ardından buton işlevini yapar. Eğer write addressteki data butondan değilde başka bir yerden (harici veya makrodan) aktif edilirse makro çalışmaz.

3- After Execute Macro → Bu butona basıldığı zaman ilk önce buton işlevini yapar ardından tanımlanan makro gerçekleşir. Eğer write addressteki data butondan değilde başka bir yerden (harici veya makrodan) aktif edilirse makro çalışmaz.

Her bir makroda yazılabilecek maksimum satır sayısı 512 ve 1 satırda kullanılabilecek maksimum word 128 dir. Maksimum 512 sub-makro oluşturulabilir. Kullanıcılar makroları alt makrolar ile yorumlayabilir. Bu sayede makro programın daha kısa ve etkili olması sağlanır. Alt makro çağırmak için kullanılan komut CALL n'dir. N değeri alt makro numarasıdır. (N= 1-512 arası)

Makro Çeşitleri

DELTA DOP Serisi HMI'lar 4 gruba ayrılmış 11 çeşir makro sunar. Bunlar;

1-Element ON/OFF Makro → Buton elementi (Pushbuton, kalıcı…vb) tüm bit datası içeren elementlerde kullanılır.
2-Before/After execute Makro → Sayısal/Karakter girişi gibi tüm elementlerde kullanılır ayrıca SYS butonunuda içeren tün buton elementlerindede kullanılır.
3-Screen Open / Close / Cycle Makro → Çalışma sırasında sayfayı ele alır. Her bir sayfanın makrosu diğerlerinden bağımsızdır.
4-Initial / Background / Clock / Submacro → Çalışma esnasında programı esas alır. Her bir programın makroları diğerlerinden bağımsızdır.

Makroların Çalışma Şekli ve Sıralaması

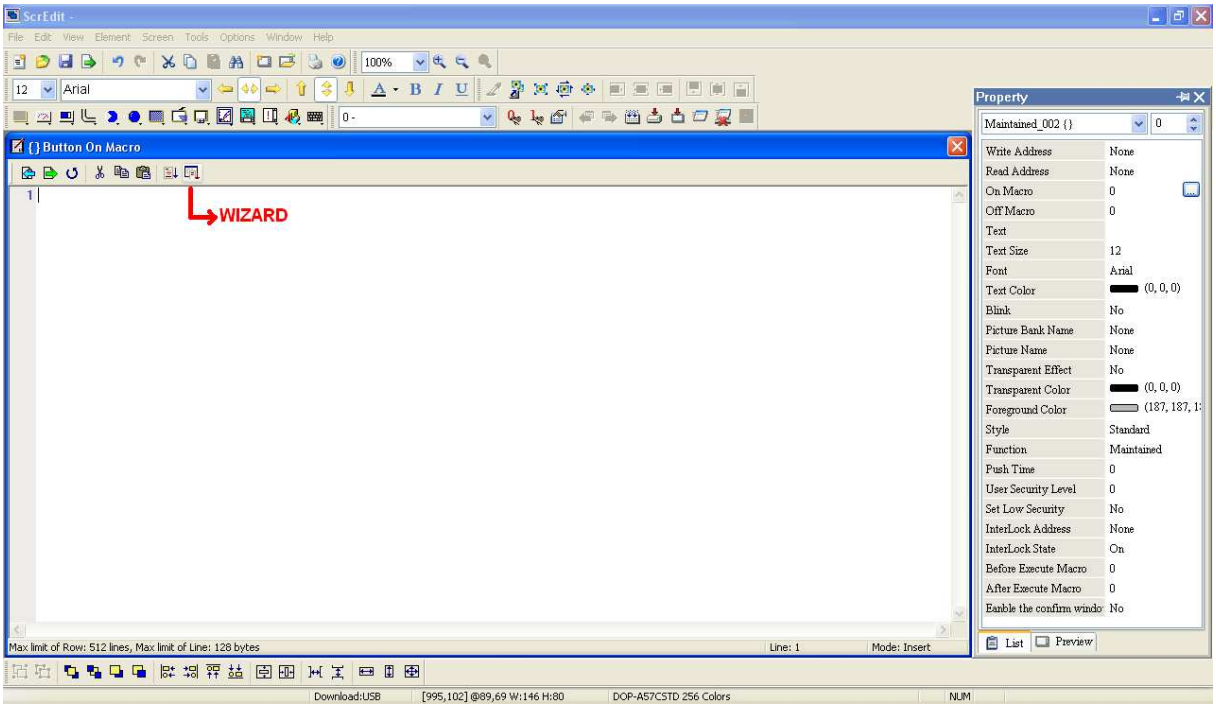
→ Bankground Macro HMI programı çalışırken arka planda her zaman çalışan makrodur. Makro son satırı işledikten sonra tekrar ilk satıra dönerek çalışmasına devam eder. Aynı anda başka makroların çalışması (Ör: Screen cycle Makro, clock makro) backgroung makronun çalışmasını etkilemez. Mesela 1 sayfada 25 tane nesne oluşturulmuş olsun ve aynı zamanda 3 satırlık backgroung makro çalışıyor olsun. Bu durumda HMI önce 25 nesnenin durumunu okur ardından background makro ilk satırını okur daha sonra 25 nesnenin durumunu tekrar okur ardından backgroung makro 2. Satırını okur. Kısaca HMI Backgroung makronun her satırını okumadan önce mevcut sayfadaki nesnelerin durumunu okur ardından background makro satırını okur. HMI okuma sırası aşağıdaki gibidir.

Ekran üzerindeki 25 nesneyi oku
Background makro 1. Satır oku
Ekran üzerindeki 25 nesneyi oku
Background makro 2. Satır oku
Ekran üzerindeki 25 nesneyi oku
Background makro 3. Satır oku

→ Programda her zaman çalışan sadece 1 tane clock makro vardır. Ve bu clock makronun son satırı işlendikten sonra tekrar ilk satırı işlemeye başlaması için gereken zaman Configuration → Standart penceresinden ayarlanır.

→ Sub makrolar subroutine çalışmaya benzer. Aynı makro satırlarının sürekli kullanılması gereken durumlarda bu satırlar sadece sub-makro içine yazılır ve ana makro programdan bu sub-makro CALL komutu ile çağırılır. Bu sayede kullanıcının etkili program yapması sağlanır. Değişiklik yapılması gerekiyorsa sadece sub-makro içinden değişiklik yapılması yeterli olur. Ayrıca sub-makroya isim verilerek içeriğinin hatırlanması kolaylaştırılır.

Makro programı yapılırken komutların doğru kullanıldığına emin olunmalıdır. (Özellikle loop (döngü) programları kullanılırken). Eğer kullanıcı makro programında sonsuz döngü oluştururs a (makro hiçbir zaman sona ermez) veya bazı koşullar gerçekleşmeden makronun çalışmayacağı programlar yazılır ise HMI anormal olarak çalışmaya başlar. Onun için kullanıcılar makro komutlarını simülasyon kullanarak test etmeli ve makro komutlarının tamamlanması sonucunda HMI nın çalışmasını test etmelidir.



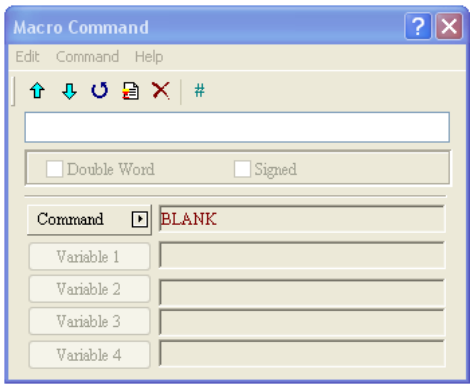
* Kullanıcılar menüden hangi makroda çalışacağını belirledikten sonra WIZARD butonuna bastığında Macro command pop-up penceresi açılır. Buradan Command butonuna basarak istediği makro komutunu seçebilir. Kullancılar isterse makro command pop-up penceresini açmadan makro komutlarını satırlara text olarakta girebilirler.

Makro penceresindeki butonların kullanımı ;

IMPORT : Daha önce kaydedilmiş makro programı açıp getitmek için kullanılır.
EXPORT : Mevcut makro programı kaydetmek için kullanılır.
UPDATE : Makro penceresinde yapılan değişiklikleri güncellemek için kullanılır. Eğer güncelleme yapılmadan makro penceresinden çıkılırsa program kullanıcıdan güncelleme onayı ister.
CUT : Makro penceresinde seçili olan satır(lar)ı başka bir yere yapıştırmak için keser.

COPY : Makro penceresinde seçili olan satırları başka bir yere yapıştırmak için kopyalar.
PASTE : CUT, COPY komutları ile seçilmiş satırları aynı program içinde (farklı makro pencereside olabilir) istenilen satırlara yapıştırır.
SYNTAX CHECK : Makro penceresinde tüm saturlarda yazılan komutları kontrol eder. Eğer yanlış yazılmış bir satır varsa kullanıcının hangi satırda yanlış yaptığını gösterir
WIZARD : Makro programını oluşturacak komutları yazmak için Macro command penceresini açar.

Aynı program içinde farklı makrolar dahi olsa tüm makro satırları birbirine kopyalanıp Yapıştırılabilirken farklı programlarda bu yapılamaz. Eğer mevcut programda kullanmak için daha önce başka programda kullanılmış makroları kopyalamak istiyorsak ilk olarak önce yapılmış programda yapılmış makro programı EXPORT ile kaydedilmeli ardından mevcut programdan kaydedilen bu makro satırı IMPORT butonu ile çağırılmalıdır.



UP : Bir üst makro satırına geçmek için kullanılır.
DOWN : Bir alt makro satırına geçmek için kullanılır.
UPDATE: Mevcut makro satırını güncellemek için kullanılır.
INSERT : Makro yazmak için araya satır açmada kullanılır.
DELETE : Mevcut Makro satırını silmek için kullanılır.
COMMENT: Makro satırına açıklama yazmak için kullanılır. Satırın başına # işareti gelir. Bu satır HMI tarafından işlenmez. Sadece açıklama amaçlı kullanılır. Programda yazılımın bir komut satırını geçici olarak iptal etmek içinde kullanılabilir.
Kullanıcılar makro satırına makro komutlarını direkt girebildiği gibi isterse **COMMAND** butonunu kullanarak ta makro komutlarını girebilirler.

Makroların kolay kullanımı için Screen Editor kullanıcılara klavyeden makro girme imkanı sağlar. Ve program komutun doğru yazıldığını kontrol eder. Eğer yanlış bir komut satırı var ise kullanıcı uyarı mesajı ile uyarılır.

Makrolarda Data Çeşitleri

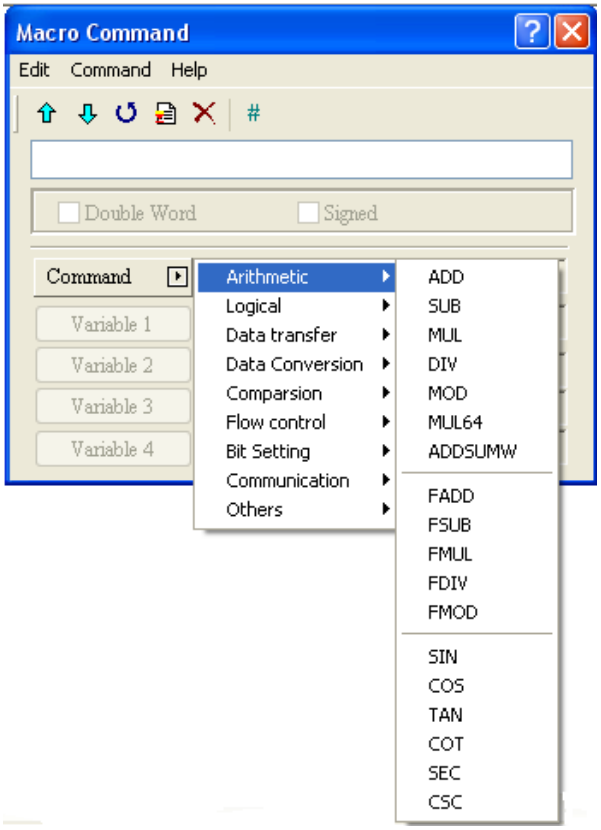
WORD : 16 bit data (b15-b0), 2 byte, Hex 0000-FFFF sayısına karşılık gelir
DWORD, DW : 32 bit data (b31 – b0), 2 word, Hex 00000000 – FFFFFFFF sayısına karşılık gelir
BYTE : 8 bit data (b7 – b0), 2 nibblei Hex 00 – FF sayısına karşılık gelir
SIGNED : İşaretli sayı. Pozitif sayıların önüne (+), negatif sayıların önüne ise (-) gelir.

→ Eğer makro komutundan sonra DW varsa bu komutun 32-bit data olduğunu DW yok ise 16-bit data olduğunu gösterir. Eğer kullanıcı DW aktif ettiyse 2 data kullanıyor demektir. Ör: komutta $0 kullanıyorsa komut $0 ve $1 beraber kullanır.

→ Eğer makro komutundan sonra SIGNED varsa bu işaretli sayı olduğunu yoksa işaretsiz sayı olduğunu gösterir.

$10=$13 + $15 → 16-bit komut, işaretsiz
$0=$2 + $4 (DW) → 32-bit komut, işaretsiz
$4=FADD($4, 1,9) (Signed, DW) → 32-bit komut işaretli

Makro Komutları (Aritmetik İşlemler)



1-) Tamsayı Çalışma Komutları

TOPLAMA(ADD) (+) → V1=V2 + V3 (V2 ve V3 datalarını toplar ve sonucu V1 datasına atar.)
$2 = $2 + 1 ($2 datasının değerini 1 arttırmak için kullanılan makro komut satırı)
$3 = $2 + $1 ($2 datasını $1 ile toplayıp sonucu $3 datasının içine atan makro komut satırı)

ÇIKARTMA (SUB)(-) → V1=V2 - V3 (V2 den V3 datasını çıkartır ve sonucu V1 datasına atar.)
$2 = $2 - 1 ($2 datasının değerini 1 azaltmak için kullanılan makro komut satırı)
$3 = $2 - $1 ($2 datasından $1 çıkartıp sonucu $3 datasının içine atan makro komut satırı)

ÇARPMA(MUL) (x) → V1=V2 x V3 (V2 ile V3 datasını çarpar ve sonucu V1 datasına atar.)
$2 = $2 * 1 ($2 datasını 2'ye katlamak için kullanılan makro komut satırı)
$3 = $2 * $1 ($2 datası ile $1 çarpılıp sonucu $3 datasının içine atan makro komut satırı)

BÖLME (DIV) (/) → V1=V2 / V3 (V2' yi V3 datasına böler ve sonucu V1 datasına atar.)
$2 = $2 / 2 ($2 datasını 2'ye bölmek için kullanılan makro komut satırı)
$3 = $2 / $1 ($2 datasını $1'e bölüp sonucu $3 datasının içine atan makro komut satırı)

İşlem sonucu WORD, DW, İşaretli veya işaretsiz olabilir. Sınırı aşan sonuçların sınırı aşan kısmı kaybolur. V1 değeri sadece dahili memory olabilirken V2 ve V3 sabit sayı veya dahili memory olabilir.

2-) Kesirli Sayı (Floating) Çalışma Komutları

TOPLAMA (FADD) → V1=FADD(V2,V3) (Signed DW)
$4= FADD ($4, 1.9) (Signed DW) ($4 datasının değerini 1.9 arttırmak için makro komut satırı)
$8= FADD ($4, $6) (Signed DW) ($4 ile $6 dataları toplamını $8 datasına yazan komut satırı)

ÇIKARTMA (FSUB) → V1=FSUB(V2,V3) (Signed DW)
$4= FSUB ($4, 1.9) (Signed DW) ($4 datasının değerini 1.9 azaltmak için makro komut satırı)
$8= FSUB ($4, $6) (Signed DW) ($4 den $6 çıkarıp sonucu $8 datasına yazan komut satırı)

ÇARPMA (FMUL) → V1=FMUL(V2,V3) (Signed DW)
$4= FMUL ($4, 1.5) (Signed DW) ($4 datasını 1.5 ile çarpmak için kullanılan makro satırı)
$8= FMUL ($4, $6) (Signed DW) ($4 ile $6'yı çarpıp sonucu $8 datasına yazan komut satırı)

BÖLME (FDIV) → V1=FDIV(V2,V3) (Signed DW)
$4= FDIV ($4, 4.3) (Signed DW) ($4 datasını 4.3'e bölmek için kullanılan makro satırı)
$8= FDIV ($4, $6) (Signed DW) ($4'ü $6'ya bölüp sonucu $8 datasına yazan komut satırı)

KALAN (FMOD) → V1=FMOD(V2,V3) veya V1=FMOD(V2,V3) (Signed DW)
$3= $2 % 5 ($2 nin 5'e bölümü sonucu kalan datayı $3 datasına atan makro komut satırı)
$8= FMOD ($4, 4) (Signed DW) ($4'ü 4'e bölüp kalanı $8 datasına atan makro komut satırı)

Kesirli (Floating) işlemler işaretli 32-bit data komutlardır. V1 sadece dahili memory olabilirken V2 ve V3 dahili memory ve sabit sayı olabilir.

Sıralı Toplama Çalışma Komutu

SIRALI TOPLAMA (ADDSUMW) → V1=ADDSUMW(V2,V3)
$0= ADDSUMW ($2, 4) ($2'den itibaren 4 data $2,$3,$4,$5 toplayıp $0'a yazan komut satırı)

3-)Trigonometrik Fonksiyon Çalışma Komutları

SINUS FONKSİYONU (SIN) → V1=SIN (V2) (Signed DW)
$0=SIN (50) (Signed DW) → $0=0,766

COSINUS FONKSİYONU (COS) → V1=COS (V2) (Signed DW)
$0=COS (50) (Signed DW) → $0=0,643

TANGENT FONKSİYONU (TAN) → V1=TAN (V2) (Signed DW)
$0=TAN (50) (Signed DW) → $0=1,192

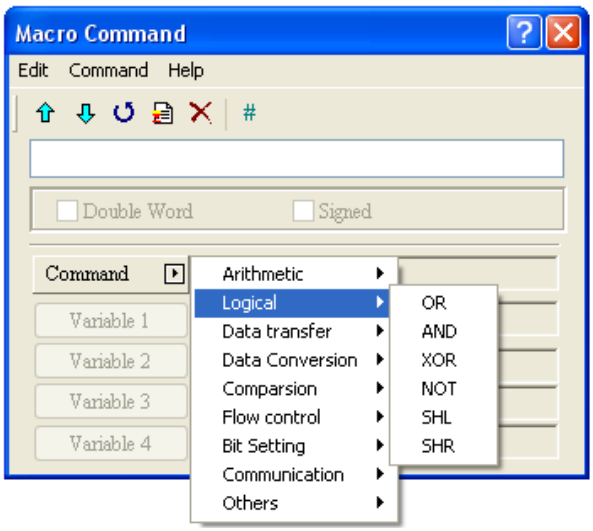
COTANGENT FONKSİYONU (COT) → V1=COT (V2) (Signed DW)
$0=COT (50) (Signed DW) → $0=0,839

SECAND FONKSİYONU (SEC) → V1=SEC (V2) (Signed DW)
$0=SEC (50) (Signed DW) → $0=1,556

COSECAND FONKSİYONU (CSC) → V1=CSC (V2) (Signed DW)
$0=COSEC (50) (Signed DW) → $0=1,305

Trigonometrik fonksiyon çalışma komutları işaretli 32-bit data komutlardır. V1 sadece dahili memory olabilirken V2 ve V3 dahili memory ve sabit sayı olabilir.

Makro Komutları (Mantık İşlemler)



MANTIK ÇALIŞMA KOMUTLARI

BIT OR (I) → V1=V2 I V3 (V2 ve V3 datalarını OR işlemine tabi tutar sonucu V1 datasına atar.)
ÖR: $2=F000H, $4=0F00H → $6=$2 I $4 → $6=FF00H olur.

BIT AND (&&) → V1=V2 &&V3 (V2,V3 datalarının AND işlemini yapar, sonucu V1 datasına atar.) ÖR: $2=F000H, $4=0F00H → $6=$2&& $4 → $6=0000H olur.

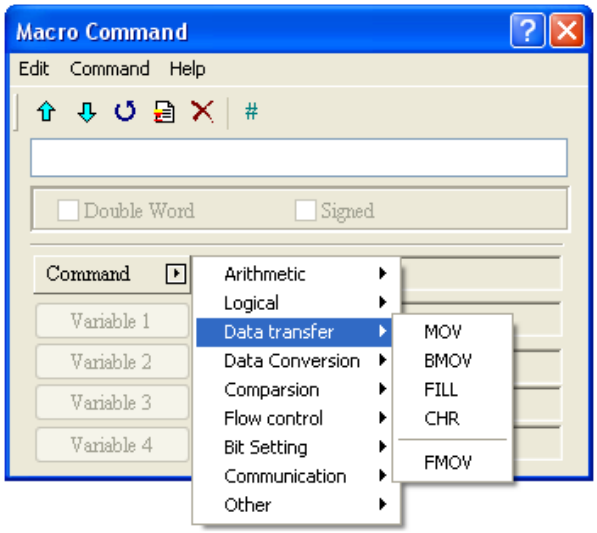
BIT XOR (^) → V1=V2 ^ V3 (V2,V3 datalarının XOR işlemini yapar, sonucu V1 datasına atar.)
ÖR: $2=F100H, $4=0F00H → $6=$2 ^ $4 → $6=FE00H olur.

BIT NOT (NOT) → V1= NOT V2 (V2 datasının NOT işlemini yapar, sonucu V1 datasına atar.)
ÖR: $2=F100H → $4= NOT $2 → $4=0EFFH olur.

<< (BIT Sola Kayma) → V1= V2 << V3 (V2 datasını V3 bit sola kaydırır, sonucu V1'e atar.)
ÖR: $2=F100H → $1 = $2 << 4 → $1=1000H olur. (V2 WORD/DWORD olabilir).

>> (BIT Sağa Kayma) → V1= V2 >> V3 (V2 datasını V3 bit sağa kaydırır, sonucu V1'e atar.)
ÖR: $2=F100H → $1 = $2 >> 4 → $1=0F10H olur. (V2 WORD/DWORD olabilir).

Makro Komutları (Data Transfer)



DATA TRANSFER KOMUTLARI

MOVE (=) → V1 = V2 (V2 datasını V1'e transfer eder. V1 PLC datası veya ekran datası olabilir.
$0 = 4 → $0 datasının içine 4 transfer eder.
$4 = $2 → $2 datası $4 datasına transfer eder.

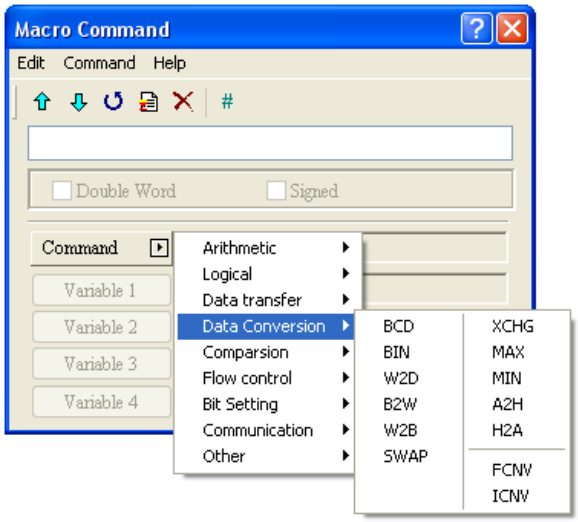
BMOV → BMOV (V1, V2,V3) (V2'den itibaren V3 datayı V1'den itibaren V3 dataya yazar.)
BMOV ($10, $1, 4) → $1'den itibaren 4 datayı ($1,$2,$3,$4) $10'dan itibaren 4 dataya ($10,$11,$12,$13) transfer eder.

FILL → FILL (V1, V2,V3) (V1'den itibaren V3 datay2 V2 datasını transfer eder.)
FILL ($0, $5, 4) → $0'dan itibaren 4 dataya ($0,$1,$2,$3) $2 datasının içindeki değeri transfer eder. Eğer $5=2 ise $0=2, $1=2, $2=2, $3=2 olur.

CHR → CHR (V1, "V2") V2'de yazılı olan texti ASCII'ye çevirir V1'e atar. Maksimum uzunluk 128 word olabilir. 1 tane V2 datası 2 tane V1 datası işgal eder. Fazla gelen karakterler sıra ile 1 sonraki dataya geçer. (V1 + 1, V1 + 2….). Sıra ile düşük bit ve yüksek bit ASCII'ye çevrilir.
CHR ($1, "AB10") → $1 = 4241H ve $2 = 3031H olur.

FMOV→ V1=FMOV (V2) (Signed DW) (Kesirli V2 datasını V1 'e transfer eder).
$0=FMOV(44.3) (Signed DW) → 44.3 kesirli sayıyı $0 içine transfer eder.

Makro Komutları (Data Dönüştürme)



DATA TRANSFER KOMUTLARI

BCD → V1=BCD (V2) (V2'de yazılı olan BIN datayı BCD (HEX) 'ye çevirir sonucu V1'e atar).
$5 = BCD ($4) ve $4=5564 ise $5=5564H olur.

BIN → V1=BIN (V2) (V2'de yazılı olan BCD (HEX) datayı BIN'e çevirir sonucu V1'e atar).
$5 = BIN ($4) ve $4=5564H ise $5=5564 olur.

W2D → V1 = W2D (V2) (V2'deki decimal WORD datayı DWORD'e dönüştürür V1 içine yazar).
$7=W2D($4) (SIGNED) ve $4=-10 → $7=-10 olur. (32-bit olacağı için $7 ve $8 birlikte olur).

B2W → V1 = B2W (V2,V3) (V2'den itibaren V3 byte'ı V1'den itibaren V3 word'e dönüştürür. Her bir byte değeri ayrı ayrı word'e dönüştürülür. Word'lerdeki yüksek değerlikli byte'lar "0" olur. Önce düşük byte daha sonra yüksek byte worde dönüştürülür).
$20=B2W($10,4) ve $10=AB12H,$11=34CDH → $20=12H, $21=ABH, $22=CDH, $23=34H olur.

DATA TRANSFER KOMUTLARI-2

W2B → V1 = W2B (V2,V3) (V2'den itibaren V3 word'ün düşük byte değerini V1'den itibaren word'lere atar. a yüksek byte değeri dikkate alınmaz.
$20=W2B($10,4) ve $10=1234H, $11=5678H, $12=ABH, $13=CDH → $20=7834H, $21=CDABH olur.

SWAP → SWAP (V1,V2,V3) (V2'den itibaren V3 tane word'ün düşük byte değeri ile yüksek byte değerini değiştirip V1'den itibaren V3 tane word'lere atar.)
SWAP ($2, $11, 1) ve $11=1234H → $2=3412H olur.

XCHG → XCHG (V1,V2,V3) (DW) (V2'den itibaren V3 tane word'ün değeri ile V1'den itibaren V3 tane word'ün değerini birbirleri ile yer değiştirir.
XCHG ($2, $11, 1) ve $11=1234H, $2=5678H → $11=5678H, $2=1234H olur.

MAX → V1 = MAX (V2,V3) (V2 ve V3 datalarından değeri yüksek olan datayı V1 datasına atar).
$0=MAX($1,$2) ve $1=2, $2=10 → $0=10 olur.

MIN → V1 = MIN (V2,V3) (V2 ve V3 datalarından değeri düşük olan datayı V1 datasına atar).
$0=MIN($1,$2) ve $1=2, $2=10 → $0=2 olur.

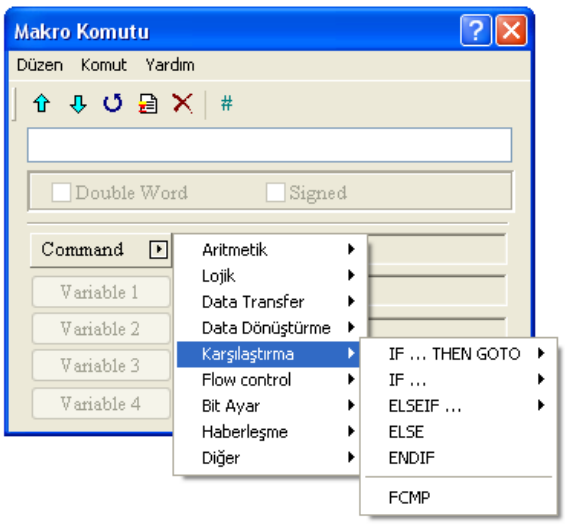
A2H → V1 = A2H (V2) (V2'den sonraki 4 ASCII datayı V1'de hex tamsayıya çevirir).
$1=A2H($10) ve $10=0034H, $11=0033H, $12=0036H, $13=0038H → $1=4368H olur.

H2A → V1 = H2A (V2) (V2 Hex datanın her bir basamağını V1'den itibaren 4 dataya ASCII olarak dönüştürür).
$10=H2A($2) ve $2=1234H ise $10=0031H, $11=0032H, $12=0033H, $13=0034H olur.

FCNV → V1 = FCNV (V2) (Signed DW) (V2 datasındaki tamsayıyı kesirli (noktalı) sayıya çevirip V1 datasına atar).
$2=FCNV($0) (Signed DW) ve $0=100 ise $2=100.0 olur.

ICNV → V1 = FCNV (V2) (Signed DW) (V2 datasındaki kesirli (noktalı) sayıyı tamsayıya çevirip V1 datasına atar).
$2=FMOV(100.5) Signed DW
$0=ICNV($2) (Signed DW) → İlk satırda $2=100.5 olur 2.satırda $0=100 olur.

Makro Komutları (Karşılaştırma Komutları)



KARŞILAŞTIRMA KOMUTLARI

IF....THEN GOTO .....
Komut : IF koşul THEN GOTO hedef
IF koşulu sağlanıyorsa program GOTO belirtilen hedefte çalışmasına devam eder.
Komutta kullanılabilecek KOŞULLAR,
V1 ve **V2** değerleri ekran dahili memory veya sabit sayı olmalıdır.
V1 == V2 → V1 değeri V2 'ye eşitse
V1 != V2 →V1 değeri V2 'ye eşit değilse
V1 > V2 →V1 değeri V2 'den büyükse
V1 >= V2 →V1 değeri V2 'den büyük eşitse
V1 < V2 →V1 değeri V2 'den küçükse
V1 <= V2 →V1 değeri V2 'den küçük eşitse
V1 && V2 == 0 →V1 ile V2 AND işlemi sonucu "0" ise
V1 && V2 != 0 →V1 ile V2 AND işlemi sonucu "0" değilse
V1 == ON →V1 ON ise
V1 == OFF →V1 OFF ise

Örnek :

IF $2>=10 THEN GOTO LABEL 1
…… ($2<10)
……
……
LABEL 1 ($2>=10)
……
……
……

* Eğer $2 datasının değeri 10'dan büyük veya eşitse program çalışmasına Label 1 satırından itibaren devam eder. $2 değeri 10 değerinden düşük olunca program taramaya bir alt satırdan devam eder ve tarama sırası geldiği LABEL 1 ve altındaki komutları tarar.

KARŞILAŞTIRMA KOMUTLARI

Komut : IFB V1 == (ON I OFF) THEN GOTO LABEL Hedef
Eğer V1 bitinin durumu ON veya OFF ise program çalışmasına hedefte belirtilen LABEL satırından itibaren devam eder. V1 biti PLC adresidir.

Örnek :

IFB 1@M0 == ON THEN GO TO LABEL 1
…… (M0 = ON)
……
……
LABEL 1 (M0 = OFF)
……
……
……

* Eğer PLC'nin M0 biti ON ise program çalışmasına LABEL 1 satırından itibaren devam eder.

KARŞILAŞTIRMA KOMUTLARI

Komut : IF V1 == V2 THEN CALL Hedef Sub-Makro No
Eğer V1 datası V2 ye eşit ise ilgili Sub-makroyu çalıştırma komut satırı.

Örnek :

IF $2 = 10 THEN CALL 1

* Eğer $2 = 10 ise program 1 numaralı sub makroyu çalıştırır. Sub- makroyu çalıştırdıktan sonra normal çalışmasına geri döner. Sub- Makronun bittiğini RET komutundan anlar. Onun için hazırlanan her sub-makronun sonuna RET komutu yazılmalıdır.

KARŞILAŞTIRMA KOMUTLARI

Komut : IF ….. ELSE ….. ENDIF

IF Koşul 1
Durum 1
ELSE IF Koşul 2
Durum 2
ELSE
Durum 3
ENDIF

Çoklu durumların mantık olarak tespiti için kullanılır. Eğer koşul 1 sağlanırsa Durum 1 gerçekleşir. Eğer koşul 1 yoksa program koşul 2'yi tarar. Eğer koşul 2 sağlanırsa Durum 2 gerçekleşir. Eğer Koşul 1 ve Koşul 2 sağlanamazsa (ELSE) Durum 3 gerçekleşir. IF komutları ENDIF ile sonlandırılmalıdır.

Örnek :

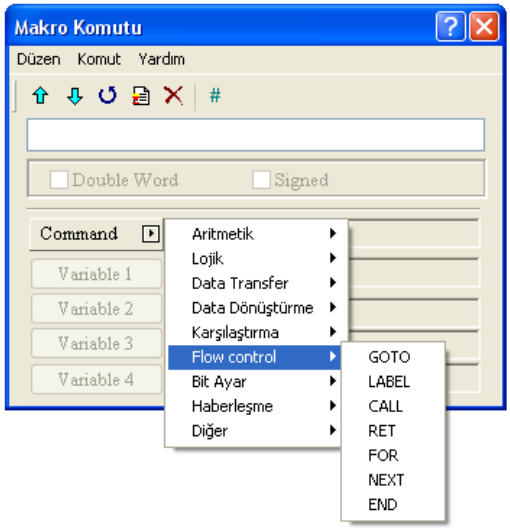
IF $1 == 10
$1 = $1 +1
ELSEIF $1 == 20
$1 = $1 +2
ELSE
$1 = $1 +15
ENDIF

Eğer $1 datasının değeri 10 ise $1 datasına 1 ekleyen (IF), eğer $1 datasının değeri 20 ise $1 datasına 2 ekleyen (ELSEIF) eğer $1 datasını değeri 10 ve 20' den farklı bir değerse $1 datasına 15 ekleyen (ELSE) makro program. IF komut satırı ENDIF ile sonlandırılmalıdır. Çoklu durum karşılaştırmaları IF ile başlar, aradaki karşılaştırmalar ELSEIF ile devam eder, son karşılaştırma ELSE komutu ile karşılaştırma sonlandırılır ve ENDIF ile IF makro sonlandırılır.

KARŞILAŞTIRMA KOMUTLARI

ENDIF → IF ile başlayan komutlar ENDIF ile sonlandırılmalıdır. Yukarıdaki örneği inceleyiniz.

Makro Komutları (Akış Kontrol Komutları)



AKIŞ KONTROL KOMUTLARI

GO TO → Program Kayıtsız şartsız komutta belirtilen hedef LABEL satırına gider.
Komut : GO TO LABEL V1
Program V1 de belirtilen LABEL satırına gider.

LABEL → GO TO komutu ile gidilecek komut satırını belirler. Aynı LABEL numarası 2 kere kullanılamaz.
ÖR: LABEL 5

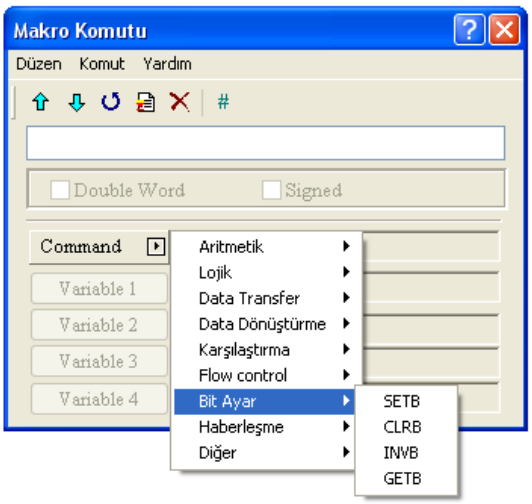
AKIŞ KONTROL KOMUTLARI

CALL…RET → Sub-Makro çalıştırma komutları.
Komut : CALL V1

Bu komut ile V1 datasından çalıştırılmak istenen sub-makro belirlenir. Sub-Makro numarası 1-512 arasında olabilir. V1 datası Dahili memory ve sabit sayı olabilir. CALL V1 ile V1 nolu sub-makro çağırılır Sub-makro tamamlandıktan sonra (RET) program CALL komutundan bir sonraki satırdan çalışmasına devam eder. Sub-makronun tamamlanabilmesi için RET komutu ile sonlandırılması gerekir. Kullanıcılar Sub-makrolar isim verebilirler. Fakat V1 datası sayı olmalıdır. Kullanıcılar Sub-makro içinden farklı bir sub-makro çalıştırabilirler. Fakat bu en fazla 6 adet olabilir.

FOR ….NEXT → Program döngüsü oluşturma
Komut: FOR V1
KOŞUL.
NEXT
Çoklu seviye kontrol için kullanılır. Maksimum 3 seviyeye kadar kullanılabilir.

Makro Komutları (Bit Ayar Komutları)



BİT AYAR KOMUTLARI

SETB → İstenilen biti ON yapma.
Komut : SETB V1
V1 bitini ON yapar.
Örnek : $1 datasının 5. Bitini ON yapmak istediğimizde,
$1 = 0000000000000000B iken
SETB $1.5 yapıldıktan sonra
$1 = 0000000000010000B olur

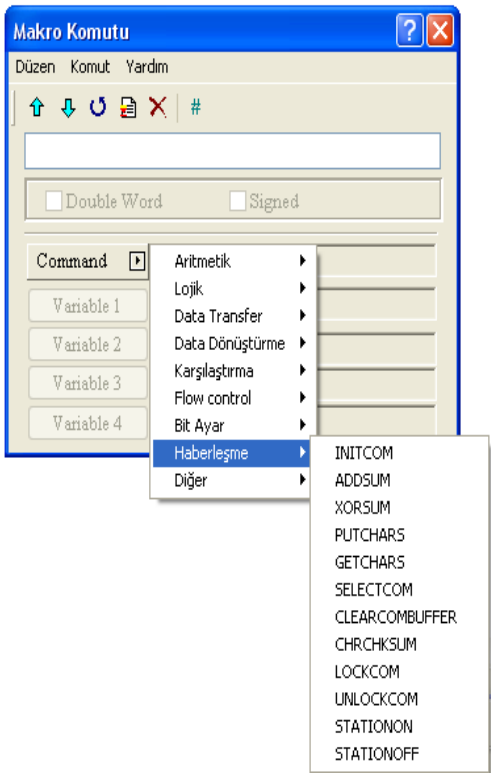
CLRB → İstenilen biti OFF yapma.
Komut : CLRB V1
V1 bitini OFF yapar.
Örnek : $2 datasının 15. Bitini OFF yapmak istediğimizde,
$2 = 1111111111111111B iken
CLRB $2.15 yapıldıktan sonra
$2 = 0111111111111111B olur

BİT AYAR KOMUTLARI

INVB → İstenilen biti OFF ise ON veya ON ise OFF yapma (tersleme).
Komut : INVB V1
V1 bitinin değerini ON ise OFF veya OFF ise ON yapar (tersler).
Örnek : $3 datasının 0. Bitini terslemek istediğimizde,
$3 = 1111111111111111B iken
INVB $3.0 yapıldıktan sonra
$3 = 1111111111111110B olur

GETB → İstenilen bitin durumunu alma (ON/OFF).
Komut : V1 = GETB V2
V2 bitinin ON/OFF değerini alır V1 bitine kopyalar.
Örnek : $0 datasının 3. Bitinin durumunu alıp $10 datasının 5. Bitine kopyalamak için,
$0 = 0000000000000100B iken
$10 = 0
$10.5 = GETB $0.3 yapıldıktan sonra
$10 = 0000000000010000B olur

Makro Komutları (Haberleşme Komutları)



HABERLEŞME KOMUTLARI

INITCOM → Haberleşmeye başlamak ve protokolü ayarlamak için başlangıç COM Port setup ayarı. 7 tane parametre ayarlanır.

Komut : $10 = INITCOM (0, 0, 0, 0, 0, 6, 0)
Haberleşme protokolüni $10 datasına yazar.

Parametre 1 : COM Port Seçimi (COM1, COM2)
Parametre 2 : Arabirim (interface) Seçimi (RS232, RS422, RS485)
Parametre 3 : Data Bit Seçimi (7-bit, 8-bit)
Parametre 4 : Parity Seçimi (None, Odd, Even)
Parametre 5 : Stop Bit Seçimi (1-bit, 2-bit)
Parametre 6 : Baud Rate Seçimi (300, 600, 900, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 115200 bps)
Parametre 7 : Akış Kontrol Seçimi (Yok, CTS/RTS, DTR/DSR, Xon/Xoff akıl Kontrol)

HABERLEŞME KOMUTLARI

ADDSUM → Checksum hesaplamada için ADD (Toplama) işlemi yapar.
Komut : V1 = ADDSUM(V2,V3)
V1 hesaplama sonucunun kaydedildiği data.
V2 Hesaplamanın başlangıç adresi
V3 Hesaplama data uzunluğu

XORSUM → Checksum hesaplamada için XOR işlemi yapar.
Komut : V1 = XORSUM(V2,V3)
V1 hesaplama sonucunun kaydedildiği data.
V2 Hesaplamanın başlangıç adresi
V3 Hesaplama data uzunluğu

PUTCHARS → COM Porttan karakter gönderilmesini sağlar.
Komut : V1 = PUTCHARS(V2,V3,V4)
V1 Haberleşme sonucunu gösterir. Eğer haberleşme başarılı olduysa değeri (1), başarısız olduysa V1 değeri (0) olur.
V2 Seri porttan gönderilecek datanın başlangıç adresi
V3 Seri porttan gönderilecek datanın uzunluğu
V4 Maksimum izin verilen haberleşme zamanı (ms)

GETCHARS → COM Porttan karakter alınmasını sağlar.
Komut : V1 = GETCHARS(V2,V3,V4)
V1 Haberleşme sonucunu gösterir. Eğer haberleşme başarılı olduysa değeri (1), başarısız olduysa V1 değeri (0) olur.
V2 Seri porttan alınan datanın kaydedileceği başlangıç adresi
V3 Seri porttan alınan datanın uzunluğu
V4 Maksimum izin verilen haberleşme zamanı (ms)

HABERLEŞME KOMUTLARI

SELECTCOM →Haberleşme yapılacak COM Port seçimi
Komut : SELECTCOM (V1)
V1 parametresinin değeri 0 ise COM1 1ise COM2 portu haberleşme için seçilmiş olur.

Haberleşme makroları kullanılırken seçilen COM Port sistemin kullandığı COM pot ile aynı olamaz. Tüm haberleşme komutları bu komut ile seçilen COM port ile yapılır. Bundan dolayı farklı makrolar birbirlerini desteklemediğinden dolayı makrolar arası çakışma olmaz.

CLEARCOMBUFFER → COM Port Buffer siler
Komut : CLEARCOMBUFFER (V1,V2)
V1 Haberleşme Portu seçimi 0 ise (COM1) ve 1 ise (COM2)
V2 Beffer tipi 0 ise Receiving (Alıcı) Buffer Alanı 1 ise Sending (Gönderici) Buffer Alanı

ÖRNEK : COM2 portunun Receiving Buffer Alanını silmek için aşağıdaki komut kullanılır.
CLEARCOM BUFFER(1,0)

CHRCHKSUM → Yazı ve karakterlerin data uzunluğu ve checksum hesaplar.
Komut: V1=CHRCHKSUM(V2,V3,V4)
V1 dahili datasının içinde V2 datasında bulunan karakter veya yazı uzunluğu kaydedilir.
V2 karakter yada yazı dizisi
V3 hesaplanan (V2 parametresindeki) checksum değerinin kaydedildiği data.
V4 hesaplanan checksum uzunluğu (V3 datasının)i 1 ise byte 2 ise word olduğu gösterir.

ÖRNEK: "24" yazıcının data uzunluğu ve checksum hesaplaması.
Yazının içindeki tüm karakterler ASCII'ye çevrildikten sonra toplanır. 2 değeri ASCII'de 32 ve 4 değeri ASCII'de 34'e karşılık gelir.
$0= CHRCHKSUM("24", $10, 2) komut işlemi sonrasında $0=4 ve $10=66H olur.

LOCKCOM → COM Port kapatma komutudur.
Komut: V1 = LOCKCOM(V2, V3)
V1
V2
V3

UNLOCKCOM → COM Port açma komutudur.
Komut: UNLOCKCOM(V1)
V1

STATIONON → İstasyon aktif etme.
Komut: STATIONON(V1, V2)
V1 : COM PORT (0: COM1, 1:COM2, 2:COM3)
V2 : İstasyon Numarası

STATIONOFF → İstasyon pasif etme.
Komut: STATIONOFF(V1, V2)
V1 : COM PORT (0: COM1, 1:COM2, 2:COM3)
V2 : İstasyon Numarası

HABERLEŞME MAKRO ÖRNEKLERİ

Örnek : PLC

Y0 üzerinde ON/OFF Makro Ayarlama
Y0 biti ON (Set) yapma makro satırı ve haberleşme komutları aşağıdadır.

ASCII → :01050500FF00F6\r\n
HEX → 3A 30 31 30 35 30 35 30 30 46 46 30 30 46 36 0D OA

ON Makro

$0=INITCOM(0,0,0,2,0,6,0)	Haberleşme ayarı COM1, RS232, 9600, 7, E, 1 ayarlanır.
SELECTCOM(0)	COM 1 portu seçilir.
CHR($11, "01050500FF00F6")	Komut satırı $11 datasından itibaren ASCII koda çevrilir.
$10=3A00H	Başlangıç biti ayarı. HMI komutu göndereceği zaman yüksek byte ile düşük byte'ın yeri değişir. Başlangıç karakterini 3A göndermek için 3A00 gönderilmelidir.
$18=0A0DH	Sonuç (END) biti ayarı. HMI komutu göndereceği zaman yüksek byte ile düşük byte'ın yeri değişir. Başlangıç karakterini 3A göndermek için 0A0D gönderilmelidir.
$50=PUTCHARS($10, 18, 500)	$10 datasından itibaren 18 byte'ı 500 ms zaman aşımı ayarı ile göndermeye başlar.
$51=GETCHARS($100, 18, 500)	Haberleşme sonrası PLC cevabını $100'den itibaren 18 byte 500 ms zaman aşımı ayarı ile almaya başlar.

Y0 biti OFF (Reset) yapma makro satırı ve haberleşme komutları aşağıdadır.

ASCII → :010505000000F5\r\n
HEX → 3A 30 31 30 35 30 35 30 30 30 30 30 30 46 35 0D OA

OFF Makro

$0=INITCOM(0,0,0,2,0,6,0)	Haberleşme ayarı COM1, RS232, 9600, 7, E, 1 ayarlanır.
SELECTCOM(0)	COM 1 portu seçilir.
CHR($21, "010505000000F5")	Komut satırı $21 datasından itibaren ASCII koda çevrilir.
$20=3A00H	Başlangıç biti ayarı. HMI komutu göndereceği zaman yüksek byte ile düşük byte'ın yeri değişir. Başlangıç karakterini 3A göndermek için 3A00 gönderilmelidir.
$28=0A0DH	Sonuç (END) biti ayarı. HMI komutu göndereceği zaman yüksek byte ile düşük byte'ın yeri değişir. Başlangıç karakterini 3A göndermek için 0A0D gönderilmelidir.
$50=PUTCHARS($20, 18, 500)	$20 datasından itibaren 18 byte'ı 500 ms zaman aşımı ayarı ile göndermeye başlar.
$51=GETCHARS($110, 18, 500)	Haberleşme sonrası PLC cevabını $110'den itibaren 18 byte 500 ms zaman aşımı ayarı ile almaya başlar.

Örnek : DTB

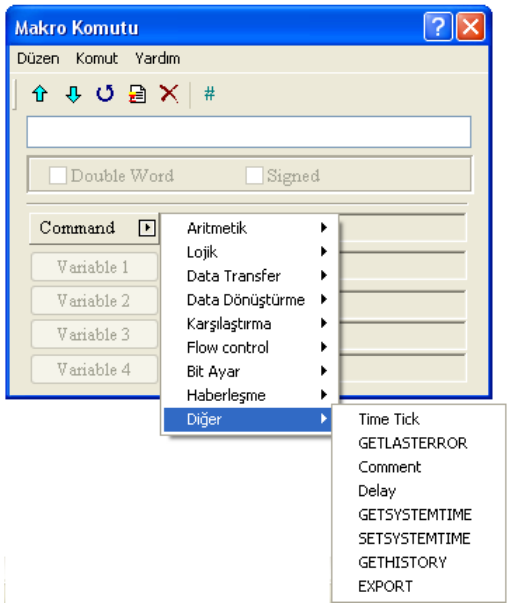
DTB sıcaklık kontrol ünitesinin set değerini (1001H) 20 derece ayarlamak için on makroya aşağıdaki makro komutları yazılmalıdır.

ASCII → :0106100100C820\r\n
HEX → 3A 30 31 30 36 31 30 30 31 30 30 43 38 32 30 0D OA

ON Macro

$0=INITCOM(1,2,0,2,0,6,0)	Haberleşme ayarı COM2, RS485, 9600, 7, E, 1 ayarlanır.
SELECTCOM(1)	COM 2 portu seçilir.
CHR($1002,"0106100100C820")	Komut satırı $1002 datasından itibaren ASCII koda çevrilir.
$100=3A00H	Başlangıç biti ayarı. HMI komutu göndereceği zaman yüksek byte ile düşük byte'ın yeri değişir. Başlangıç karakterini 3A göndermek için 3A00 gönderilmelidir.
$1009=0A0DH	Sonuç (END) biti ayarı. HMI komutu göndereceği zaman yüksek byte ile düşük byte'ın yeri değişir. Başlangıç karakterini 3A göndermek için 0A0D gönderilmelidir.
$50=PUTCHARS($1001,20,500)	$1001 datasından itibaren 20 byte'ı 500 ms zaman aşımı ayarı ile göndermeye başlar.
$51=GETCHARS($1121,20,500)	Haberleşme sonrası PLC cevabını $1121'den itibaren 20 byte 500 ms zaman aşımı ayarı ile almaya başlar.
$1050 = B2W($1125, 6)	Gelen ASCII kodu çözmek ve $1125'ten itibaren peşpeşe 6 byte datayı 6 WORD dataya çevrilir.
$1060 = A2H($1051)	Sonra, $1052'den sonra datalarkahi Hex datayı dönüştürür ve sonucu decimal formatta gösterir.

Makro Komutları (Diğer Komutlar)



DİĞER KOMUTLAR

TIMETICK
V1 = TIMETICK → Sistem ilk başlangıcından şu ana kadar geçen zamanı V1 datasına kaydeder.

GETLASTERROR
V1 = GETLASTERROR → Son Hata değerini V1 datasında gösterir. Eğer hiçbir hata yoksa "0" gösterir.

#V1 → Makro satırını okunabilir makro yapar. Bu komut makro fonksiyonunun çalışmasını etkilemez. Herhangi bir makro satırının önüne # işareti konarak makro satırı sadece okunabilir hale gelir. Makro fonksiyonun tekrar çalışması için önündeki # işaretini kaldırmak yeterli olur. Makro komutu çalışır hale gelir.

DİĞER KOMUTLAR

DELAY
Delay V1 → Sistem zamanı gecikmesi sağlar. Karmaşık sistemlerde sistem gecikme problemi olabilir. Sistem yoğun olduğu zaman gecikme zamanı arttırılarak sistemin delay komutunda ayarlanan süre kadar beklemesini sağlar. Daha sonra çalışmasına devam eder. Fakat çok yüksek girilen gecikme zamanları HMI'nın yavaş cevap vermesine sebep olabilir. Gecikme zamanı birimi ms'dir.

GETSYSTEMTIME
V1 = GETSYSTEMTIME → Sistem zamanını alır. Sistem bilgilerini V1 adresinden itibaren 7 dataya kaydeder.
V1 YIL
V1 + 1 AY
V1 + 2 GÜN
V1 + 3 HAFTA
V1 + 4 SAAT
V1 + 5 DAKİKA
V1 + 6 SANİYE

Ör: Sistem zamanı 2009/01/05 Pazartesi 09:26:25 ise $1 = GETSYSTEMTIME yapılırsa,
$1 = 2009, $2 =1, $3 = 5, $4 = 1, $5 = 9, $6 = 26, $7 = 25 olur.

SETSYSTEMTIME
V1 = SETSYSTEMTIME → Sistem zamanını ayarlar. Sistem bilgilerini V1 adresinden itibaren 7 datada ayarlar.
V1 YIL
V1 + 1 AY
V1 + 2 GÜN
V1 + 3 HAFTA
V1 + 4 SAAT
V1 + 5 DAKİKA
V1 + 6 SANİYE

Ör: Sistem zamanı 2009/01/05 Pazartesi 09:26:25 ayarlanacak ise $1 = SETSYSTEMTIME yapılır ve $1 = 2009, $2 =1, $3 = 5, $4 = 1, $5 = 9, $6 = 26, $7 = 25 olarak ayarlanır.

GETHISTORY
V1 = GETHISTORY (V2, V3, V4, V5, V6) → History datası alır

COMMENT
→ İşareti ile Macro yazılımı içerisine açıklamalar yazılabilir.

EXPORT
EXPORT(V1) → Dışa aktarma komutudur.
V1 = 0 → Dataları SMC Karta yazdır.
V2 = 1 → Dataları USB Flash Belleğe yazdır.
V3 = 2 → Dataları Printer'a yazdır.

HATA MESAJLARI

Compile yaparken, kullanıcının kolaylıkla bulabilmesi için hatalar çıkış (output) penceresinde görüntülenir. Bazı hatalar dikkatsizlikten bazıları ise komut parametre eksikliklerinden olabilir. Kısa programlarda hata kolay bulunabilirken uzun makro programda zor olabilir. Kullanıcıya yardımcı olmak amacıyla HMI yazılımı compile yapıldıktan sonra eğer varsa hata mesajlarını gösterir.

Programlama sırasında oluşan hata mesajları,

Code – 100 LABEL bulunamadı → GO TO komutuna karşılık tanımlı LABEL bulunamdı.
Code – 101 Recursion (Özyineleme) oluştu → Bu hata mesajı gelende SUB-MACRO'nun CALL komutu ile kendisini yinelemesi durumunda meydana gelir. Doğrudan veya dolaylı olarak çağırıldığı önemli değildir. GO TO veya FOR komutu kullanılması tavsiye edilir.
Code – 102 3'den fazle FOR sıralı döngüsü → FOR döngüsü3 'den fazla iç içe oluşturulamaz. Böyle durumlarda GO TO ve IF komutları kullanılması önerilir.
Code – 103 Alt Makro yok → CALL komutu ile çağırılan SUB-MACRO'nun programda tanımlanmamış olduğunu kullanıcıya hatırlatır.
Code – 104 NEXT sayısı FOR sayısından az → Programda FOR sayısı il NEXT sayısı aynı olmalıdır. NEXT sayısının az olduğunu kullanıcıya hatırlatır.
Code – 105 FOR sayısı NEXT sayısından az → Programda FOR sayısı il NEXT sayısı aynı olmalıdır. FOR sayısının az olduğunu kullanıcıya hatırlatır.
Code – 106 Tekrar kullanılan LABEL Komutları → Aynı LABEL komutu birden fazla kullanılmaz. Eğer kullanılırsa programda sorun çıkabilir. Bu durumda kullanıcı bu mesaj ile uyarılır.
Code – 107 Makro içinde RET Komutu var → RET komutu Ana makro programında değil SUB-MACRO programında Ana makro programına dönmek için programın sonunda kullanılır. Ana makro programında makroyu sonlandırmak için RET yerine END kullanılmalıdır.

HMI Makro Hata Mesajları

Kullanıcılar makrodan hata mesajlarını okuyabilirler. Bir hata olduğunda ve kullanıcı hata mesajını okumadan önce doğru komutu girdiğinde hata mesajı nın üzerine yazılır. Tüm makrolar işlenirken, herhangi bir makronun hata mesajı başka bir makrodan etkilenmez.

Code – 10 GO TO Error → Bu mesaj makroda GO TO hatası olduğunu gösterir.
Code – 11 Stack Overflow → Bu mesaj makro kapasitesinin dolduğunu gösterir. Bu durum çok fazla SUB-MACRO olduğundan veya aynı anda çok fazla makro çalıştırıldığından olabilir.
Code – 12 CALL Empty Sub-Macro → Bu bir CALL – SUM MACRO hatasıdır. CALL tarafından çağırılan SUB-MACRO boş olmamalıdır.
Code – 13 Data Read Error → Data okuma hatası. Bazen memory data okuma hatasından kaynaklanabildiği gibi çoğu zaman kontrol ünitesi data okuma hatasından kaynaklanır.
Code – 14 Data Write Error → Data yazma hatası. Bazen memory data yazma hatasından kaynaklanabildiği gibi çoğu zaman kontrol ünitesi data yazma hatasından kaynaklanır.
Code – 15 Divisor is 0 → Bölme komutu gerçekleştirirken bölenin "0" olduğunu gösterir.

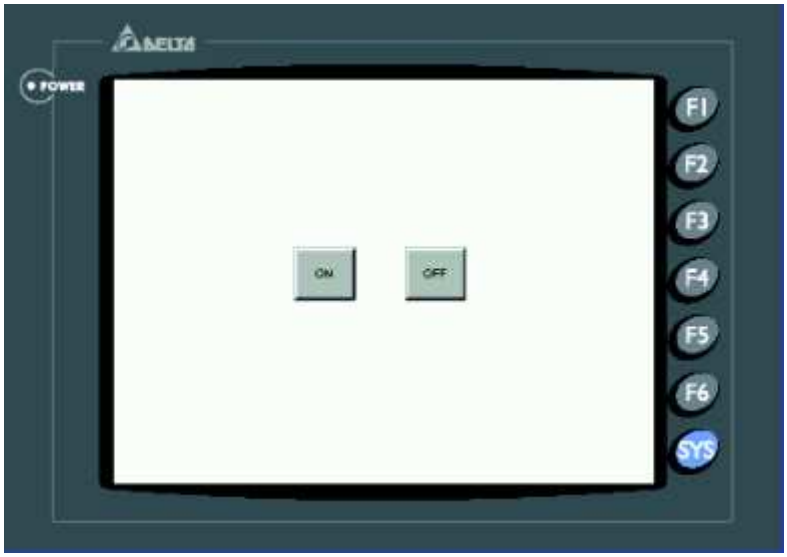
ÖRNEK : MAKRO KOMUTLARI ile E SERİSİ DELTA SÜRÜCÜYÜ RUN/STOP Yapma

E-Serisi Sürücü'yü RS485 üzerinden haberleştirmek için sürücüde yapılması gereken ayarlar.

1. 09.00=2 Haberleşme Adresi 2 seçilir. (1 – 254 arası ayarlanabilir.)
2. 09.01=3 Haberleşme Hızı 38400 seçilir.
3. 09.04=0 HaberleşmeProtokolü 7-N-1 seçilir.
4. 02.00=3 1. Ana frekans komutu kaynağı RS-485 seçilir.
5. 02.01=4 1. Ana çalışma komutu kaynağı RS-485 seçilir.

DELTA HMI'yı E serisi sürücü ile Habarleştirmek için yapılması gereken ayarlar.

1. Ana kontrol cihazı DELTA CONTROLLER ASCII seçilir
2. Haberleşme protokol ayarı COM-2, RS-485,38400,7,N,1,HMI 0, PLC 1 seçilir.
3. Aşağıdaki şekilde görüldüğü gibi ON ve OFF butonları yapılır. (Set Buton) Adresleri 10.0 ve 11.0 ayarlanır.



4. ON butonunun ON makrosuna 2@INVERTER-2000=12 yazılır
5. OFF butonunu ON makrosuna 2@INVERTER-2000=1 yazılır.