نوع محصول و مدل : HMI دلتا	ناشر : شرکت فنی و مهندسی کامیاب مرام

عنوان : ماکرونویسی

مقدمه :

ماکرو قابلیت گسترده ای ست که شرکت دلتا برای افزایش کارایی برنامه های نوشته شده در HMI ، در اختیار کاربران قرار می دهد . در HMI های دلتا، انواع گوناگونی از دستورات ماکرو، همچون دستورات محاسباتی منطقی ، انتقال داده ، تبدیل داده ، مقایسه ، کنترلی ، تنظیمات بیتی، ارتباطات پورتCOM ، ترسیم و طراحی و...را برای کاربر فراهم نموده اند . در این فایل آموزشی محیط های ماکرو نویسی و دستورات ماکرو شرح داده شده است.

فهرست:

انواع ماکرو : 5

1-1. ON/OFF Macro : 7

1-2. Before Execute Macro : 8

1-3. After Execute Macro : 9

1-4. Screen Open Macro : 10

1-5. Screen Close Macro : 11

1-6. Screen Cycle Macro : 12

1-7. Sub-macro : 14

1-8. Initial Macro : 19

1-9. Background Macro : 20

1-10. Clock Macro : 22

1.پنجره نگارش برنامه ماکرو 25

2-1. نوار ابزار پنجره ماکرو نویسی : 26

دستورات ماکرو در Command : 29

مثال : 29

دستورات Arithmetic : 34

مثال : ADD 36

مثال : TAN 37

مثال : FDIV 38

دستورات Logical : 39

مثال : && 40

دستورات Data Transfer : 41

مثال : BMOV 42

مثال : MOV 43

مثال : MOV 44

دستورات Data Conversion : 45

مثال : FCNV 46

مثال : SPRINTF 47

مثال : TODWORD 50
مثال : TOHEX 51
دستورات ComparIson : 52
مثال : IF .... THEN GOTO 53
مثال : IF .... THEN CALL 54
مثال : IF 55
مثال : IF 55
مثال : ELSEIF 56
مثال : ELSEIF 57
مثال : ELSE 57
مثال : ELSE 58
مثال : ELSEIF 58
مثال : FCMP 59
مثال : Comparision 60
مثال : برنامه کاربردی Piping 61
مثال : برنامه کاربردی نمایش Gif 64
دستورات Flow Control : 66
مثال : GOTO & LABEL 67
مثال :FOR-NEXT 68
مثال : CALL 69
دستورات Bit Setting : 71
مثال : BITON 71
دستورات COM Port : 72
مثال : ارتباط HMI با برد میکرو 81
دستورات Drawing : 84
مثال : Circle 84
FileSlot : 85
دستورات File Access : 87
مثال : FileSlot Read/Write 87
دستورات Others : 90

مثال : دستور EXRCP16 95
مثال : IMRCP16 97
مثال : EXHISTORY 98
مثال : EXALARM 99
مثال : PLCDOWLOAD 100
مثال : 102
ماکرو در DOP100 : 104
OPENSCREEN : 105
مثال : OPENSCREEN 105
CLOSESUNSCREEN : 106
مثال : CLOSESUBSCREEN 106
VAR : 107
مثال : VAR 107
خطاهای ماکرو : 108
خطای کامپایل برنامه ماکرو : 108
خطای کامپایل برنامه : 109
خطا برنامه PLC برای دانلود با فرمت DVP یا ISP : 111

انواع ماکرو :

انواع محیط های ماکرو نویسی که در HMI های دلتا قرار داده شده است را در جدول زیر مشاهده می کنید.
در ادامه به شرح نحوه عملکرد و کاربرد آنها پرداخته شده است :

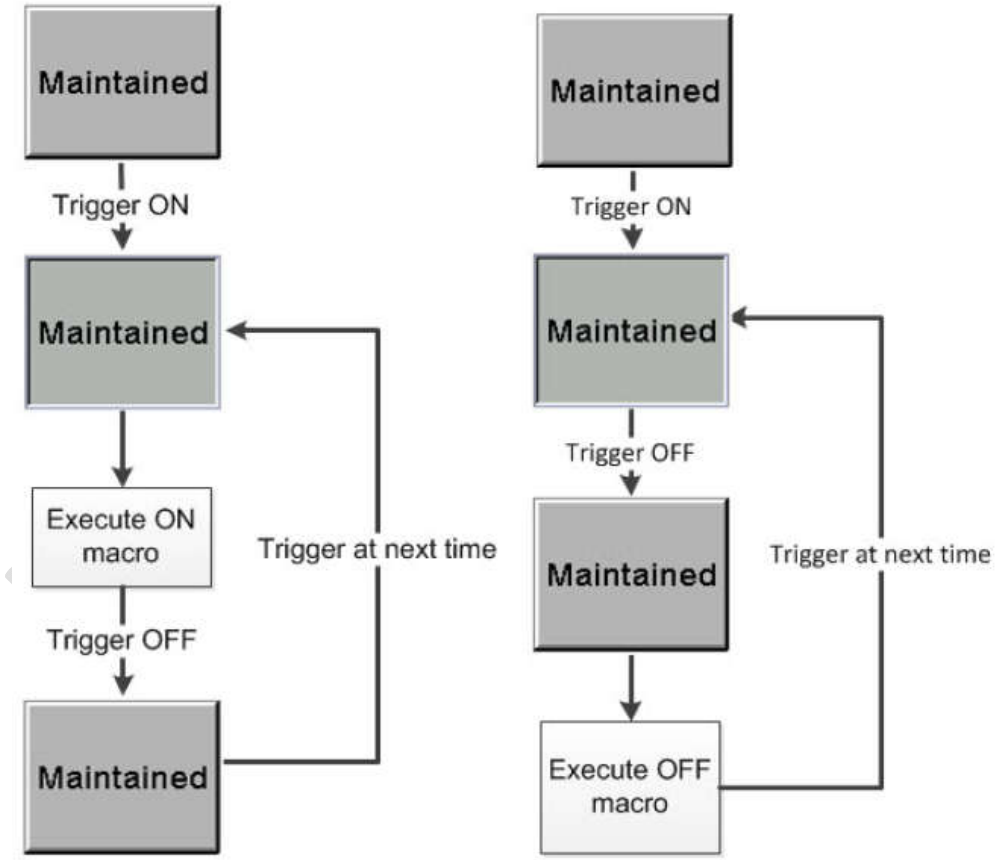
انواع ماکرو	ON Macro
	OFF Macro
	Before Execute Macro
	After Execute Macro
	Screen Open Macro
	Screen Close Macro
	Screen Cycle Macro
	Submacro
	Initial Macro
	Background Macro
	Clock Macro

انواع ماکرو و کاربرد آنها	
ON Macro	هر مرتبه یک بار بعد از روشن شدن فعال می شود و فقط برای کلید های ON Button، OFF Button ، Maintained Button ، Momentary Button قابل اجرا می باشد.
OFF Macro	هر مرتبه یک بار بعد از خاموش شدن فعال می شود و فقط برای کلید های ON Button، OFF Button ، Maintained Button ، Momentary Button قابل اجرا می باشد.
Before Execute Macro قبل از اجرا شدن ماکرو	زمانی که کاربر یکی از المان ها یا کلید های روی صفحه را فعال کند قبل از هر برنامه یا دستوری ، دستورات ماکرو اجرا خواهند شد. اگر وضعیت کلید یا المان تغییر نکند ، این دستورات اجرا نخواهند شد . برای تمامی ورودی ها و Botton ها قابل اجرا می باشد.
After Execute Macro بعد از اجرا شدن ماکرو	زمانی که کاربر یکی از المان ها یا کلید های روی صفحه را فعال کند بعد از هر برنامه یا دستوری ، دستورات ماکرو اجرا خواهند شد. اگر وضعیت کلید یا المان تغییر نکند ، این دستورات اجرا نخواهند شد . برای تمامی ورودی ها و Botton ها قابل اجرا می باشد.
Screen Open Macro	یک مرتبه بعد از باز شدن صفحه اجرا می شود.
Screen closed Macro	یک مرتبه بعد از بسته شدن صفحه اجرا می شود.
Screen Cycle Macro	دستورات آن سیکل وار ما بین اجرای دستورات Screen Open MACRO و Screen Close MACRO اجرا می شود .
Sub Macro	در HMI دلتا 512 عدد Sub MACRO قرار داده شده است. Sub MACRO مانند زیر برنامه ها یا بلاک ها در سایر زبان های برنامه نویسی عمل می کند . میتوانید دستورات و برنامه های تکراری را در آنها بنویسید و در صورت لزوم برنامه را فراخوانی کنید. نحوه برنامه نویسی Sub MACRO ها مانند سایر برنامه های ماکرو می باشد.
Initial Macro	تنها یک مرتبه بعد از Initial شدن وstartup ، HMI اجرا می شود.

Background Macro	این دستور به طور مداوم تا زمانی که HMI در حال عملکرد می باشد از خط اول تا اخر اجرا می شود و قابلیت تکرار برنامه را هم دارا می باشد . دستورات Background MACRO به صورت سیکل اجرا می شوند که هر سیکل متشکل از یک یا چندین خط دستور می باشد.
Clock Macro	به طور پیوسته در طول عملکرد HMI عمل خواهد کرد و تمامی دستورات از ابتدا تا انتها با هم اجرا خواهند شد.

1-1. ON/OFF Macro :

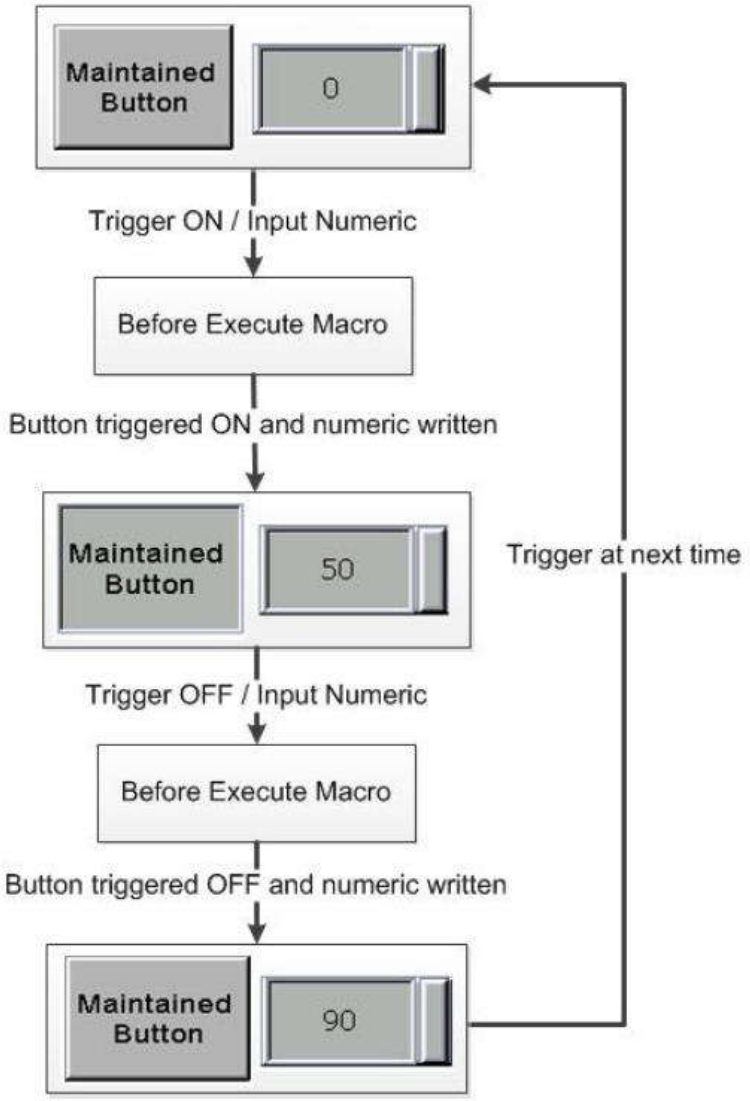
دستورات ON/OFF Macro در هر لبه اجرا می شوند و فقط برای کلید های ON/OFF Button ، Maintained Button ، Momentary Button قابل دسترسی هستند . اگرکلید روشن شود ، دستورات ON Macro اجرا خواهند شد و همچنین اگر کلید خاموش شود ، دستورات OFF Macro اجرا می شوند. فقط در صورتی که در کلید ها تغییر وضعیت به صورت ON یا OFF داشته باشیم دستورات ماکرو اجرا خواهند شد .



2-1. Before Execute Macro :

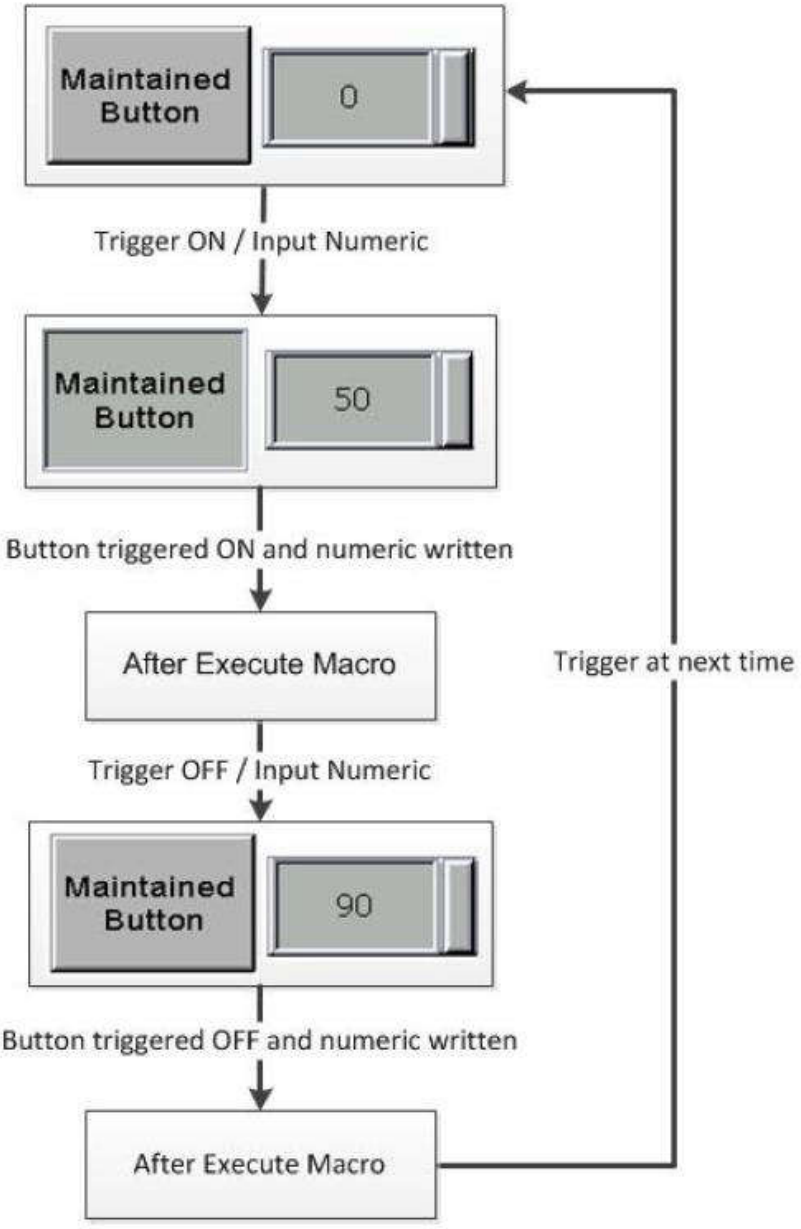
دستورات Before Execute Macro برای تمام کلیدها (Botton) و المان های ورودی قابل اجرا می باشد. با تغییر وضعیت کلید به حالت خاموش یا روشن و یا با تغییر مقدار المان ورودی ، ابتدا دستورات Before Execute Macro اجرا می شوند .

دستورات ON/OFF Macro و Before/After Execute Macro در قسمت Properties مربوط به باتن ها قابل دسترسی هستند.



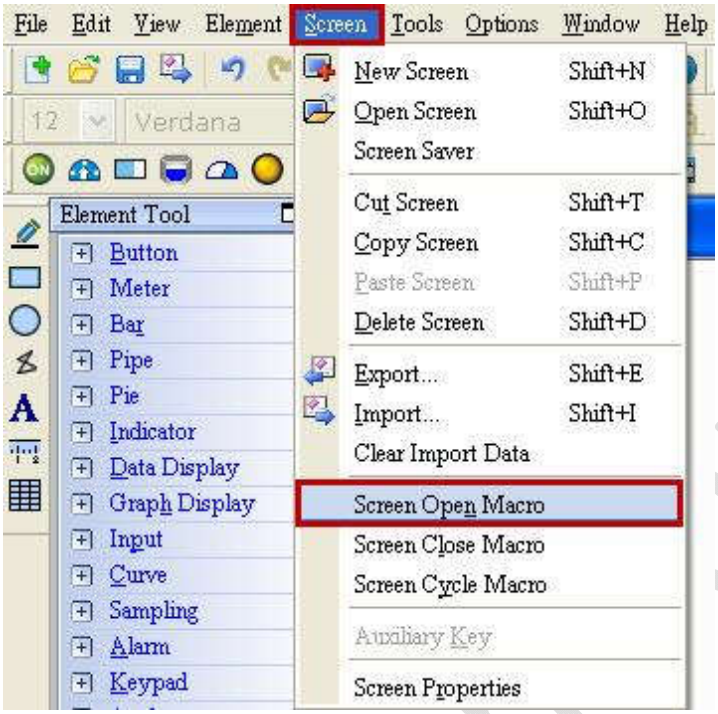
1-3. After Execute Macro :

دستورات After Execute Macro برای تمام کلیدها (Botton) و المان های ورودی قابل اجرا می باشد. با تغییر وضعیت کلید به خاموش یا روشن و یا با تغییر مقدار المان ورودی ، بعد از اعمال تغییرات ، دستورات After Execute Macro اجرا می شوند.

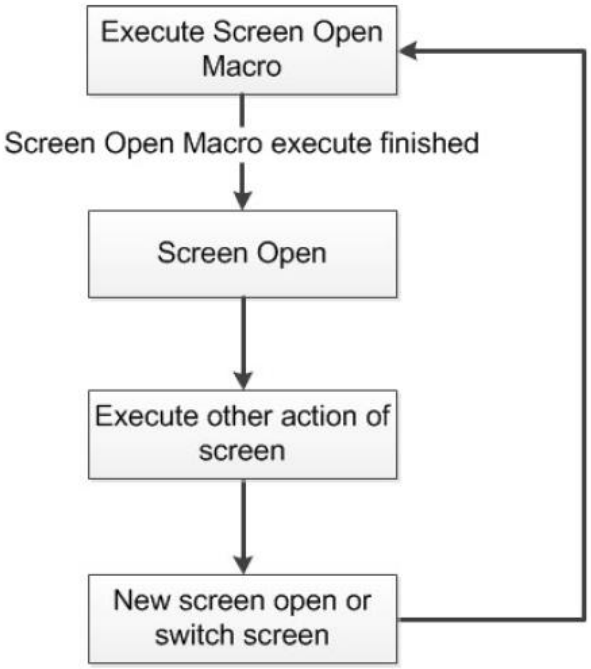


4-1. Screen Open Macro :

دستورات Screen Open Macro به محض باز شدن صفحه اجرا خواهند شد . جهت دسترسی به Screen Open Macro مطابق شکل زیر عمل کنید:

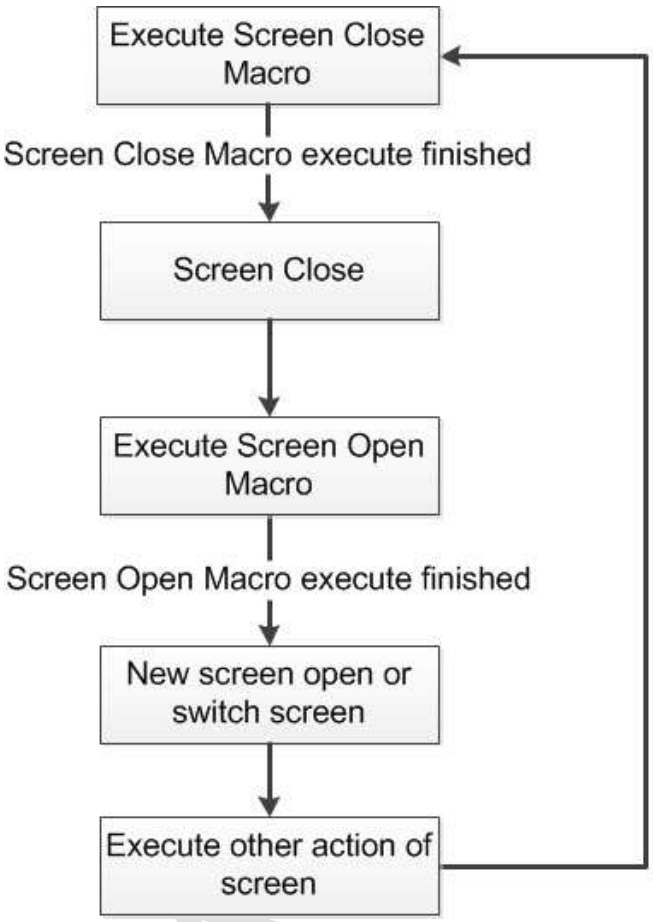


در زمان عملکرد HMI با هر بار وارد شدن به صفحه ای که برای آن Screen Open Macro نوشته اید، دستورات Screen Open Macro اجرا خواهند شد .



1-5. Screen Close Macro :

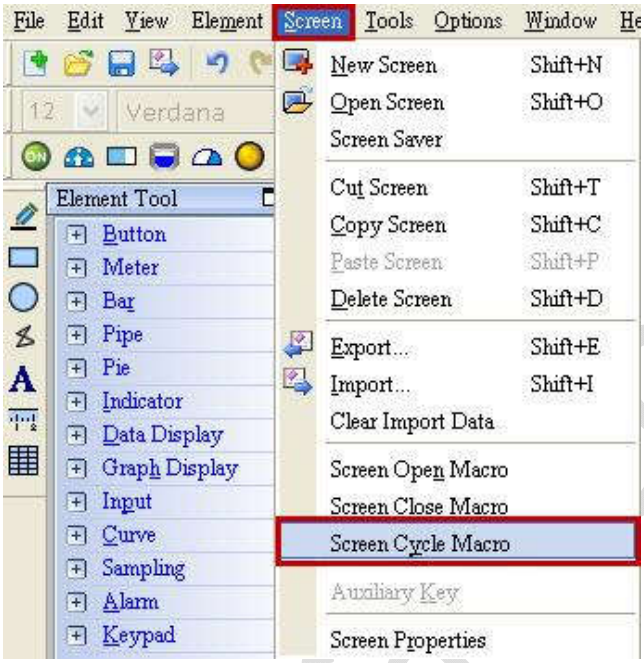
می توانید از طریق منو Screen وارد پنجره Screen Close Macro شده و دستورات مورد نظر را بنویسید.
این دستورات در هر مرتبه بسته شدن صفحه و وارد شدن به صفحه ای دیگر ، اجرا خواهند شد.



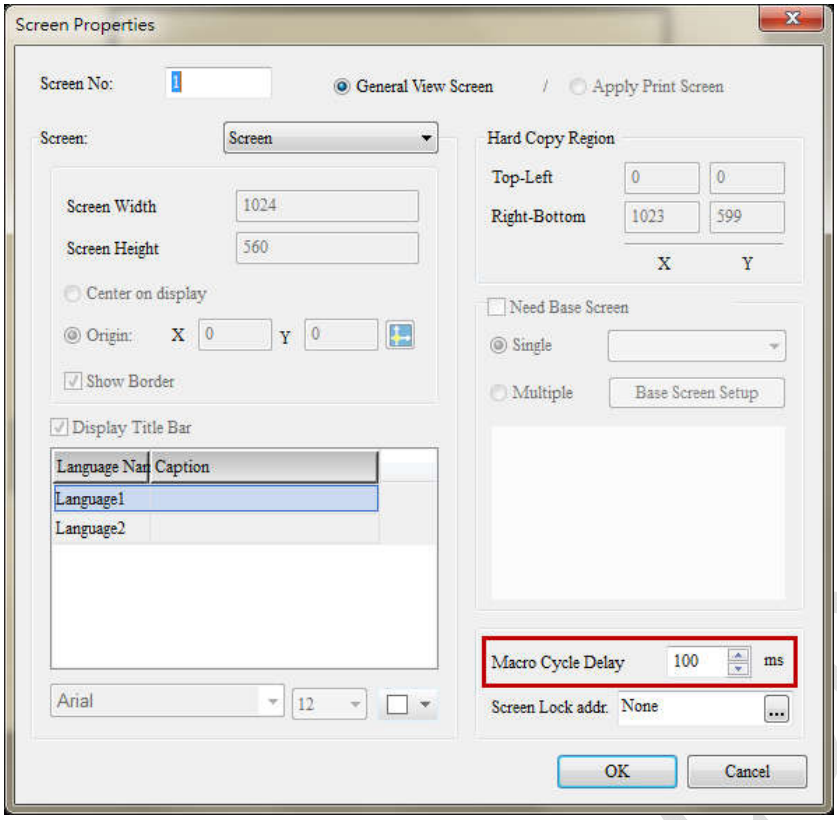
6-1. Screen Cycle Macro :

پنجره Screen Cycle Macro از طریق منو Screen قابل دسترسی می باشد . هر صفحه ای که در برنامه DOPSoft ایجاد می کنید دارای تمام محیط های ماکرونویسی Screen Open/Close/Cycle Macro می باشد.

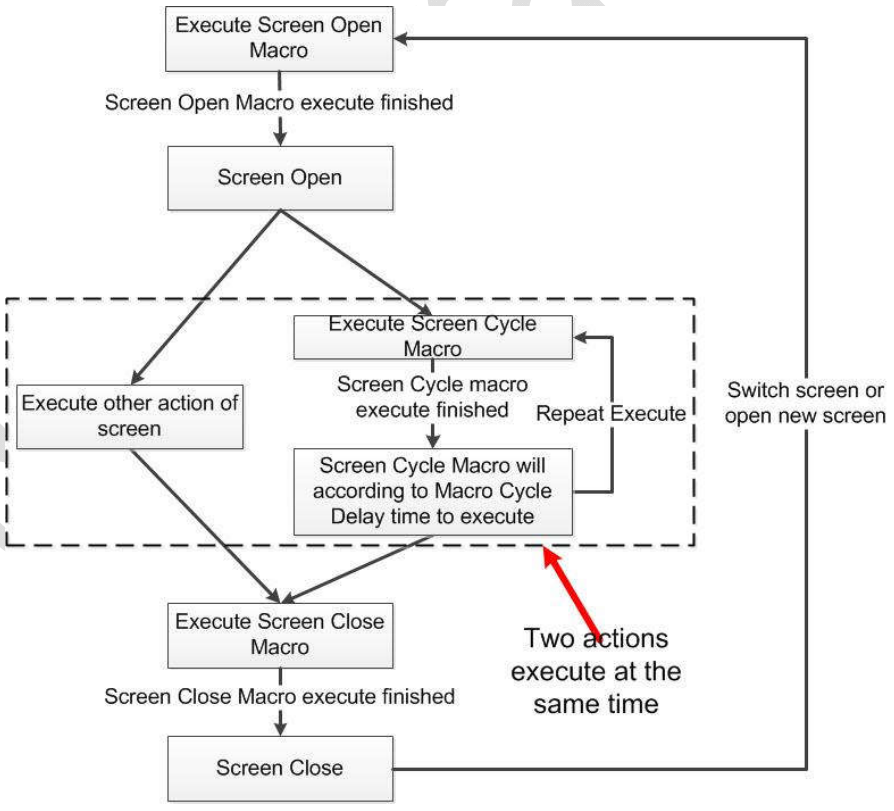
مطابق شکل زیر می تواند وارد پنجره Screen Cycle Macro شوید :



بر اساس مدت زمان تاخیر که در تنظیمات تعیین می کنید ، بعد از باز شدن صفحه ، دستورات اجرا می شوند. با دو بار کلیک کردن بر روی صفحه ، وارد پنجره Screen Properties شده و در قسمت Macro Cycle Time زمان تاخیر در اجرای دستورات Screen Cycle Macro را مشخص کنید . به طور پیش فرض مدت زمان تاخیر در اجرای Screen Cycle Macro ، 100ms می باشد.

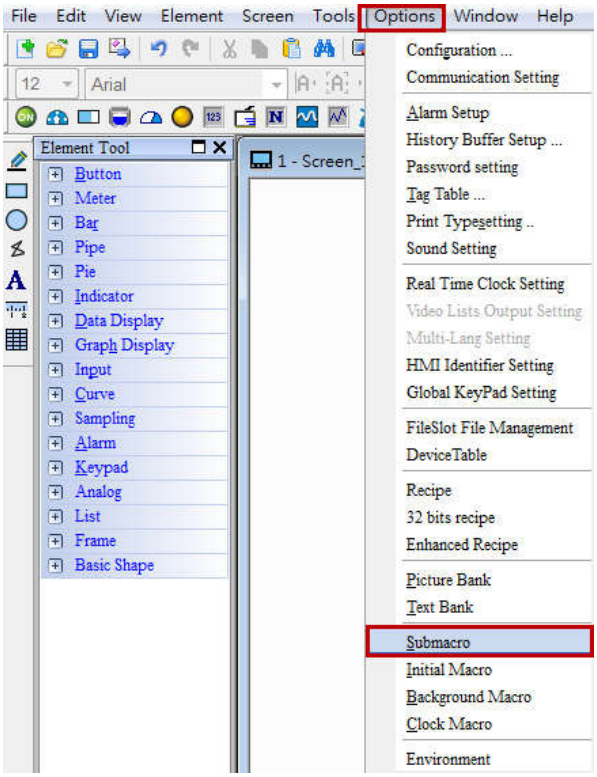


نحوه عملکرد و ترتیب اجرا شدن دستورات Screen Open/Close/Cycle Macro ، در شکل زیر نشان داده شده است :

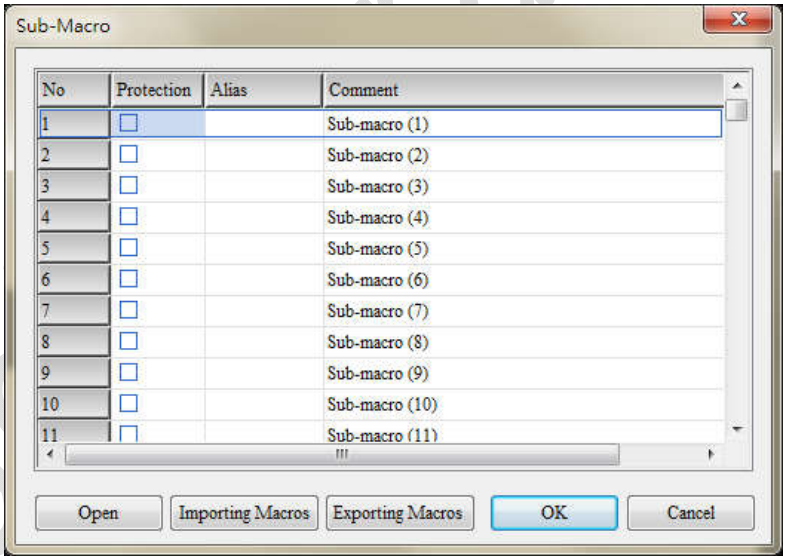


7-1. Sub-macro :

مسیر دسترسی به پنجره Sub-macro ، به صورت Option >> Sub-macro می باشد(مطابق شکل زیر).

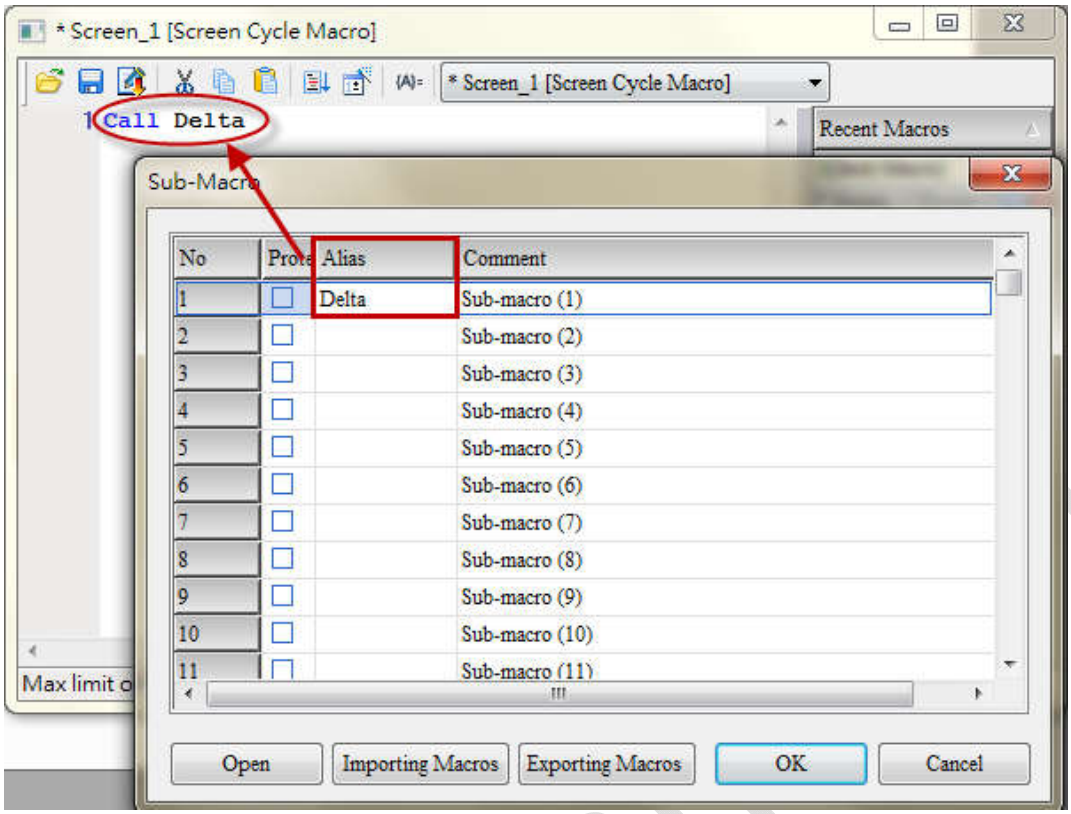


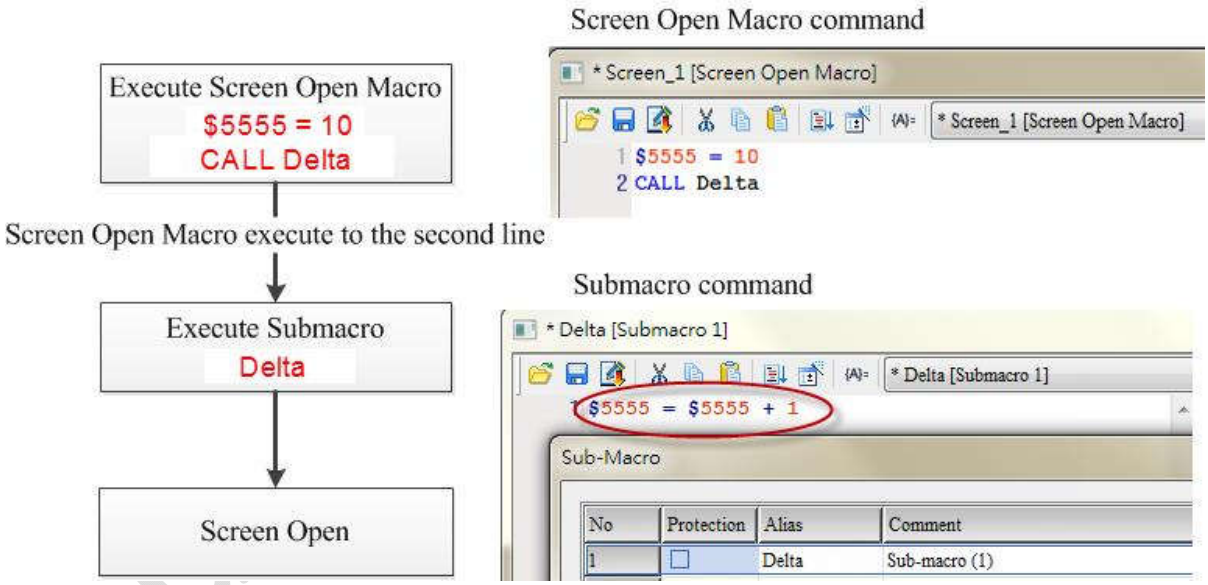
با انتخاب Sub-Macro پنجره ای به شکل زیر باز می شود:



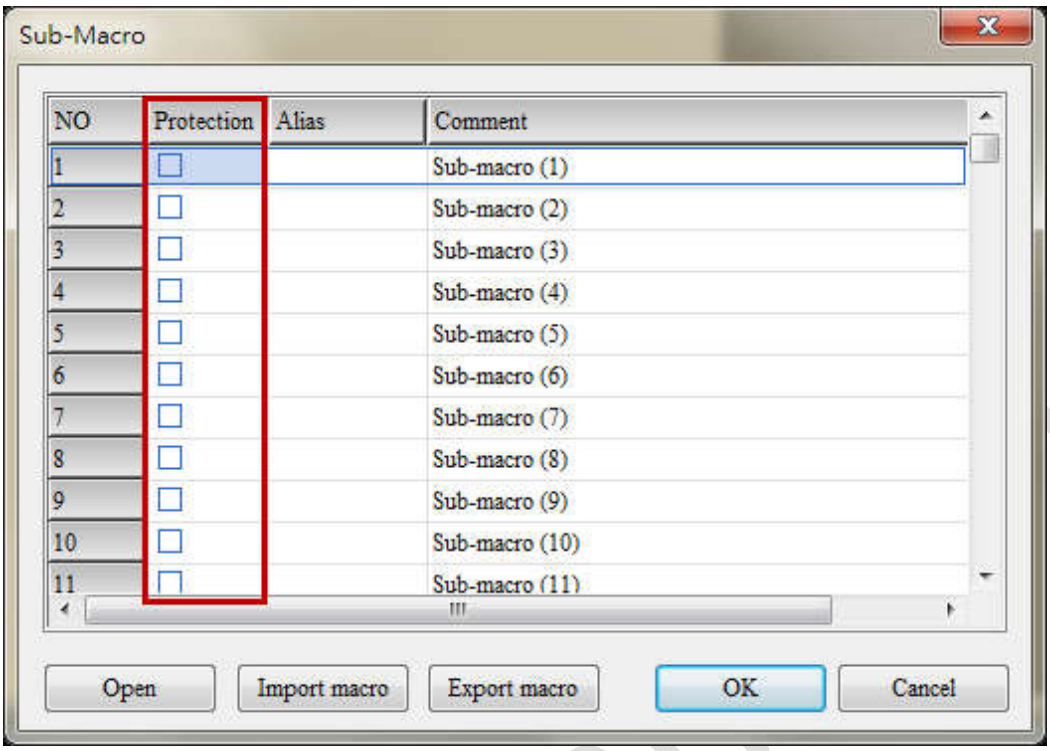
در این پنجره می توانید از 512، Sub-Macro استفاده کرده و برنامه های پر تکرار را در آنها بنویسید. برای فراخوانی Sub-Macro ها می توانید از شماره آنها استفاده کنید و یا در بخش Alias آنها را نام گذاری کنید، حداکثر می توانید از 64 کاراکتر برای نام گذاری Sub-Macro استفاده کنید.

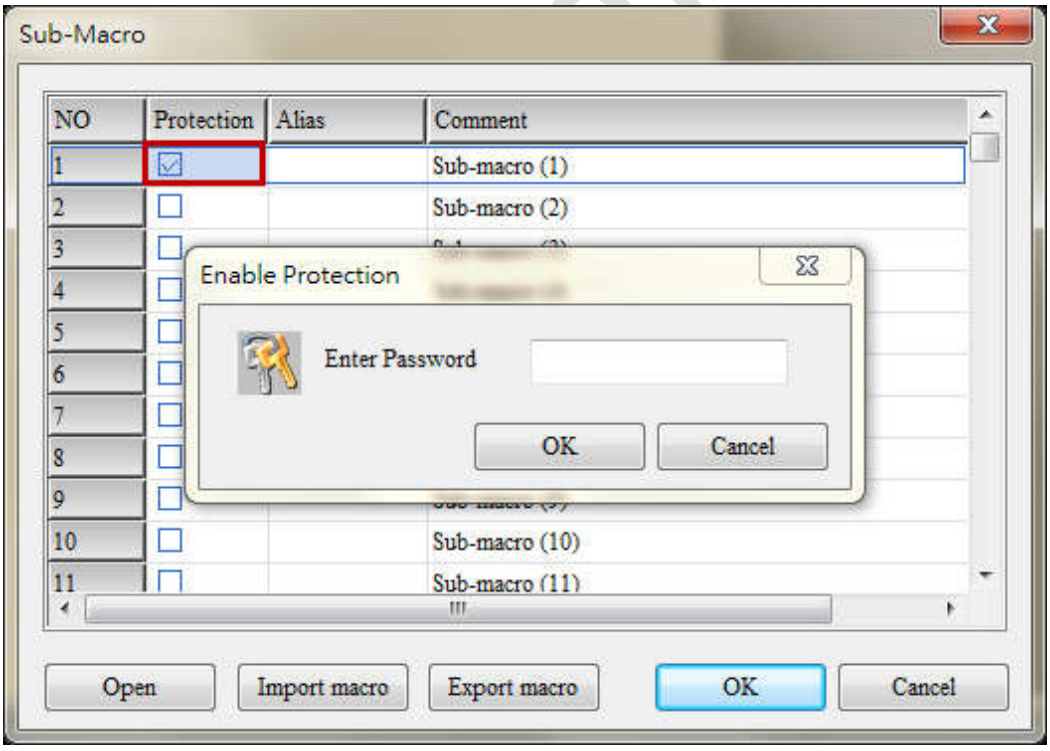
برای فراخوانی Sub-Macro داخل برنامه اصلی از دستور CALL استفاده کنید ، مطابق شکل زیر:





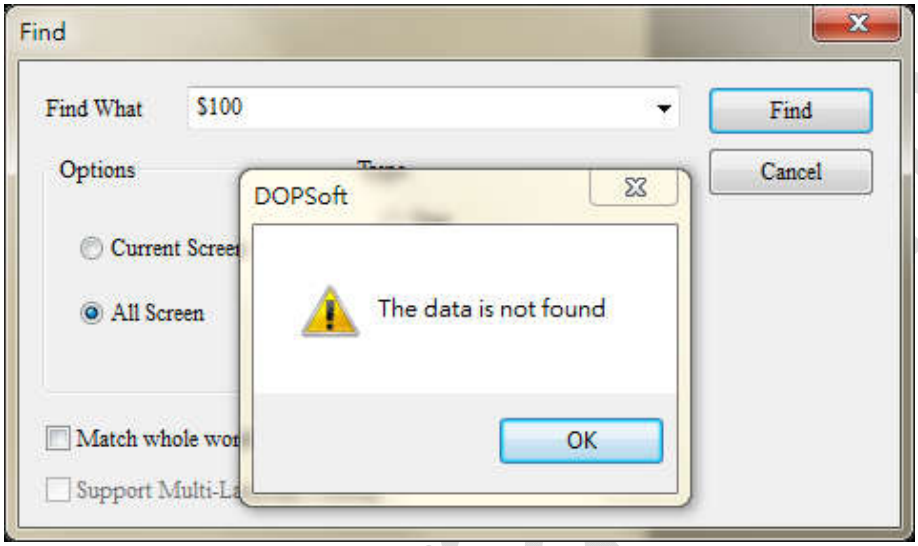
برای محافظت از برنامه می توانید برای هر Sub Macro یک Password تعیین کنید ، لذا باید قسمت Protection را فعال کرده و در پنجره نمایش داده شده به صورت زیر رمز مورد نظر را وارد نمایید.

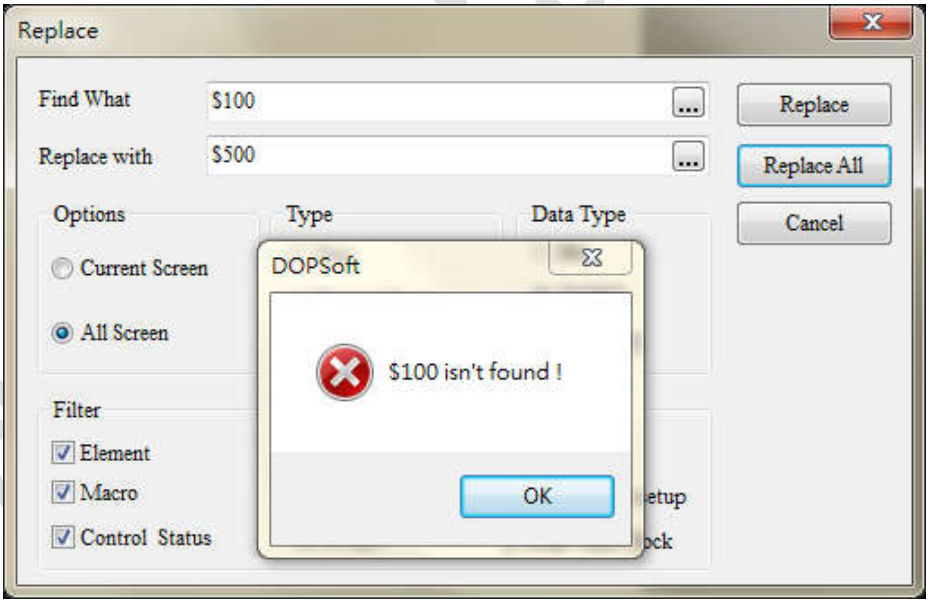




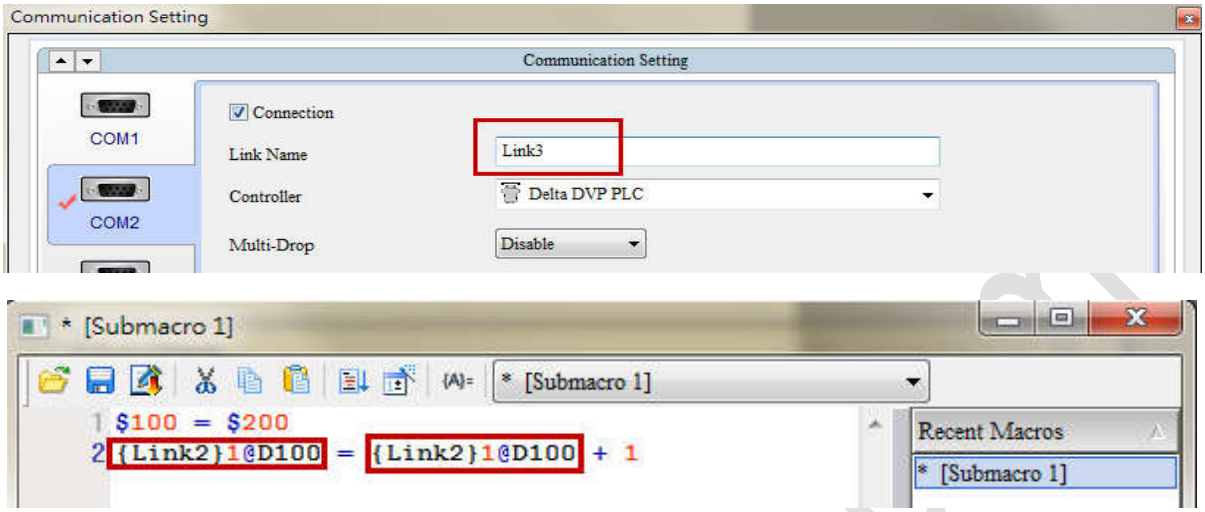
برای مشاهده مجدد و یا ویرایش برنامه Sub Macro باید رمز را وارد نمایید تا برنامه قابل دسترس شود. همچنین اگر بخواهید Password برنامه را غیر فعال کنید باید تیک گزینه Protection را غیر فعال کرده و مجددا رمز را وارد نمایید.

- ✓ قابلیت های Find و Replace که در منو Edite قرار دارند ، برای Sub Macro های حفاظت شده عمل نخواهند کرد . مثلا اگر آدرس $100 در برنامه Sub Macro حفاظت شده به کار رفته باشد، امکان جستجوی آن با دستور Find و یا جایگذاری آن با دستور Replace نمی باشد.

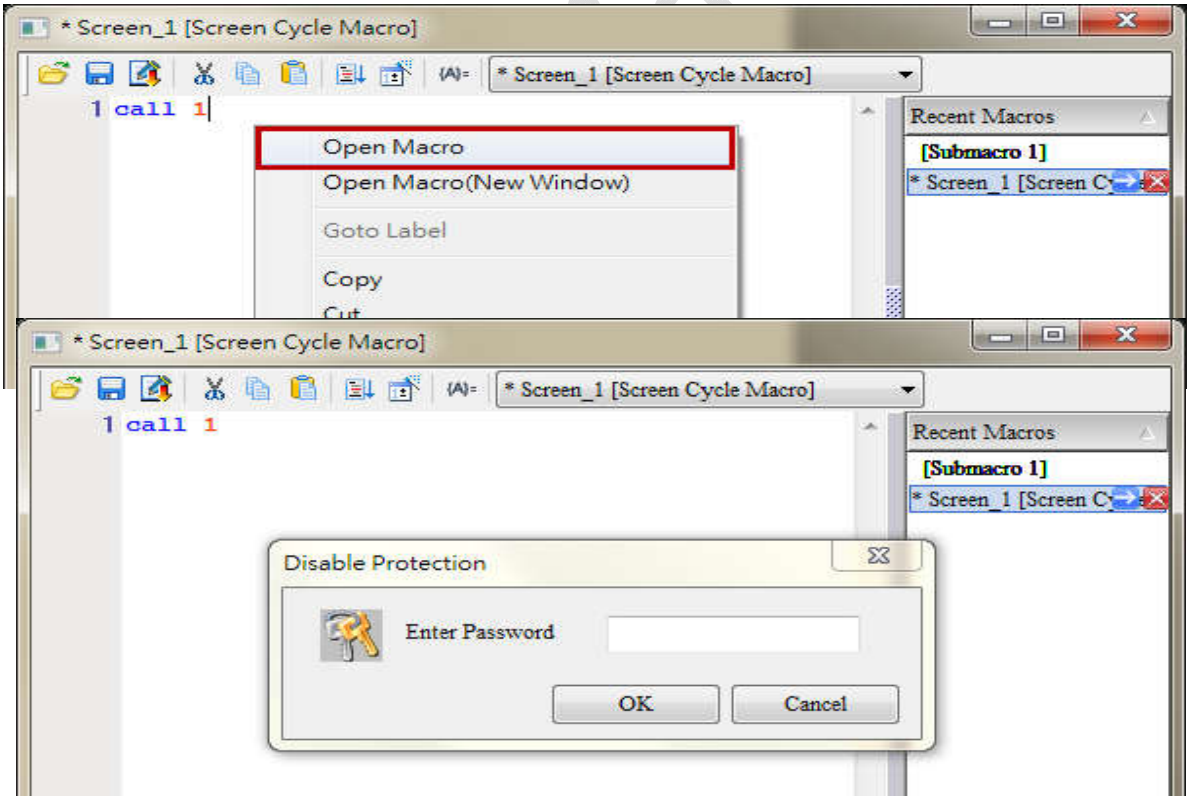




✓ همچنین اگر در پنجره Communication Setting پورت ارتباط و یا Station Number را، تغییر دهید امکان تغییر آدرس های بکار برده شده در Sub Macro محافظت به صورت اتوماتیک شده ، فراهم نمی باشد . مثلا اگر ارتباط HMI با PLC را از کام 2 به کام 3 تغییر دهید آدرس هایی که با Link2 در Sub Macro استفاده کرده اید به صورت خودکار به آدرس های Link3 تغییر نخواهند کرد.

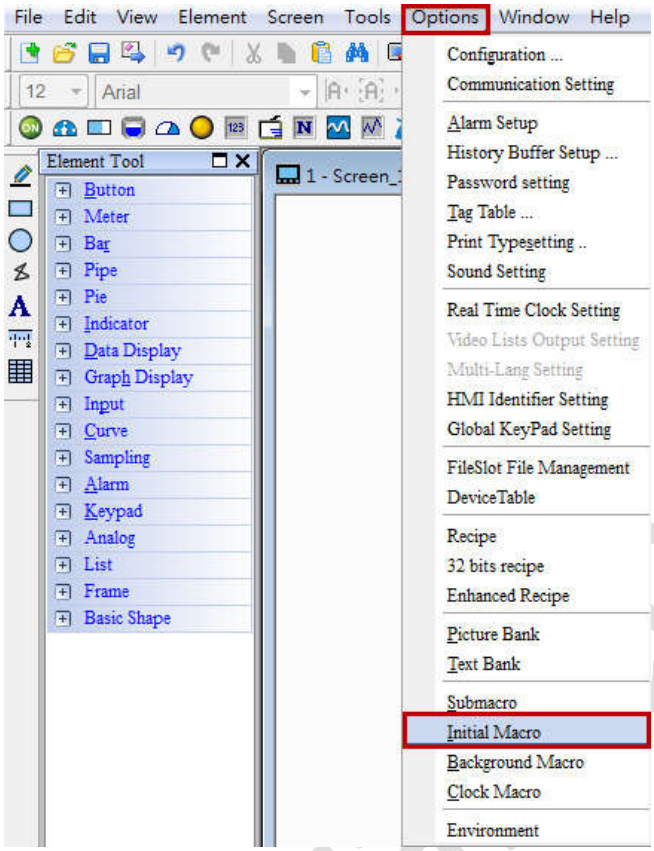


✓ در حین فراخوانی Sub Macro حفاظت شده در برنامه اصلی ، بعد از نوشتن دستور Call باید رمز مربوط به Sub Macro را وارد نمایید.

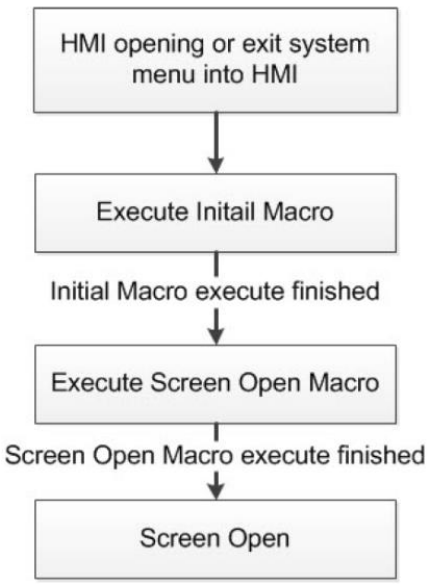


1-8. Initial Macro :

جهت دسترسی به Initial Macro وارد منو Option شده و گزینه Initial Macro را انتخاب کنید.

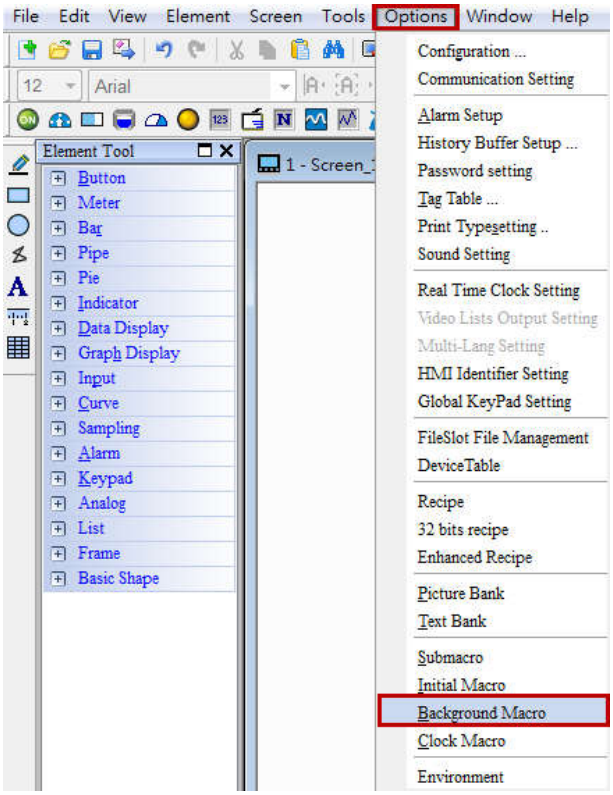


در این ماکرو می توانید Initialize یا مقدار دهی اولیه را انجام دهید. پس از شروع به کار HMI در Startup (Power ON) اولین دستورات ماکرو که اجرا می شوند ، دستورات Initial Macro خواهد بود. در شکل زیر ترتیب اجرای دستورات ماکروها نشان داده شده است:



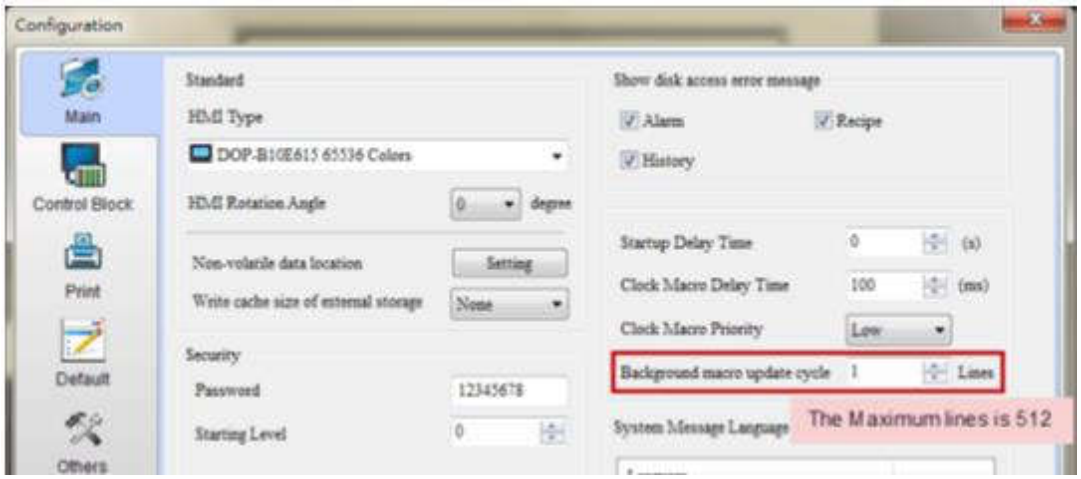
1-9. Background Macro :

جهت دست یابی به این ماکرو ، به صورت زیر عمل کنید :



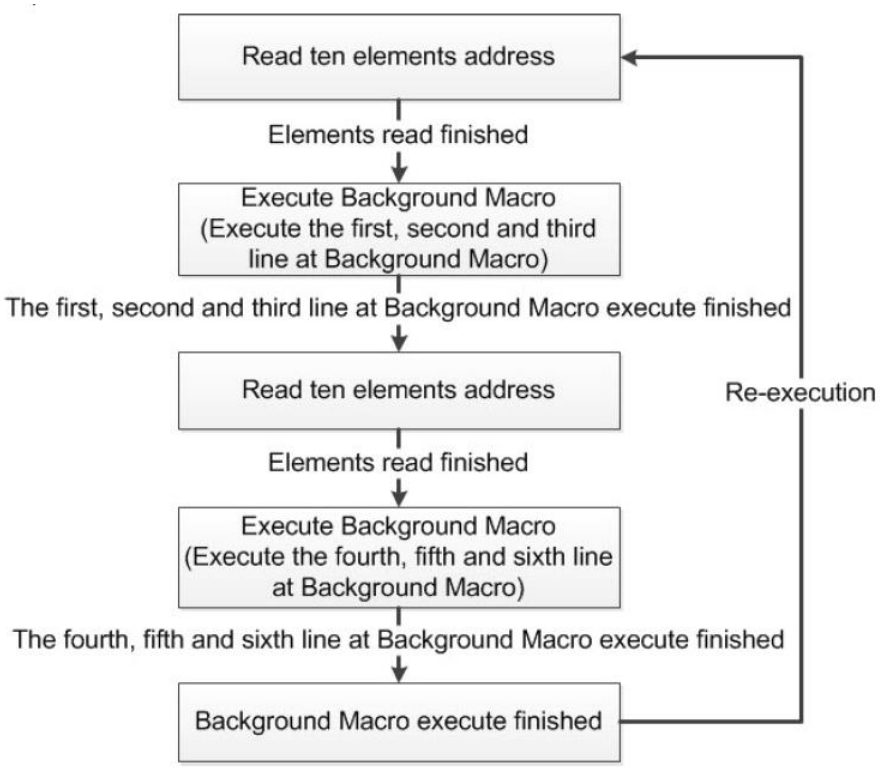
دستوراتی که در Background Macro نوشته می شود، به صورت پیوسته در طول عملکرد HMI قابل اجرا می باشند، و قابلیت تکرار برنامه بعد از اجرای آخرین خط دستور را دارند.

می توانید حداکثر 512 خط دستور بنویسید و با تعیین Background macro update cycle تعداد خطوط اجرا در هر سیکل را از 1 تا 512 خط مشخص کنید .



در اجرا شدن خطوط دستورات Background Macro ترتیب بسیار حائز اهمیت می باشد و خطوط برنامه به صورت یکجا و همزمان اجرا نمی شوند.

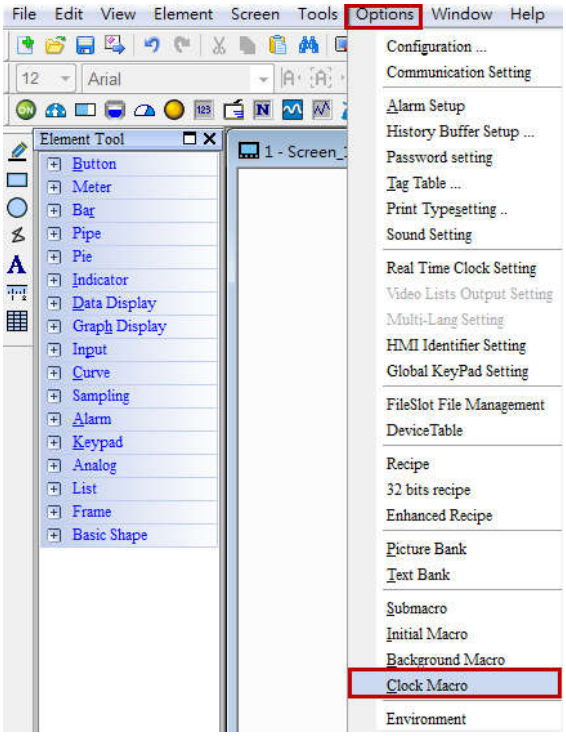
برای درک بهتر ، فرض کنید 10 المان روی صفحه HMI قرار داده شده و در Background Macro شش خط دستور نوشته شده باشد و مقدار خطوط Background Macro update Cycle را 3 مشخص شده باشد ، نحوه عملکرد و تاثیر Background Macro update Cycle در برنامه به صورت زیر می باشد :



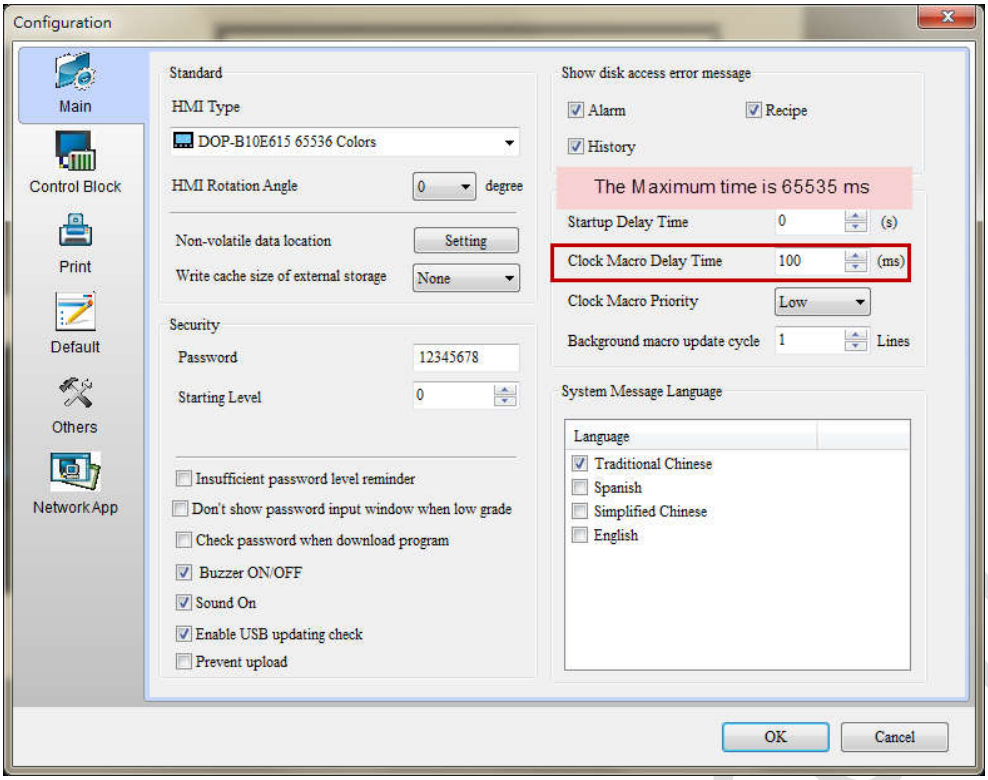
یعنی در هر Update Cycle فقط به تعداد خطوطی که در Background Macro update Cycle مشخص کرده باشیم ، از خطوط برنامه اجرا خواهند شد.

1-10. Clock Macro :

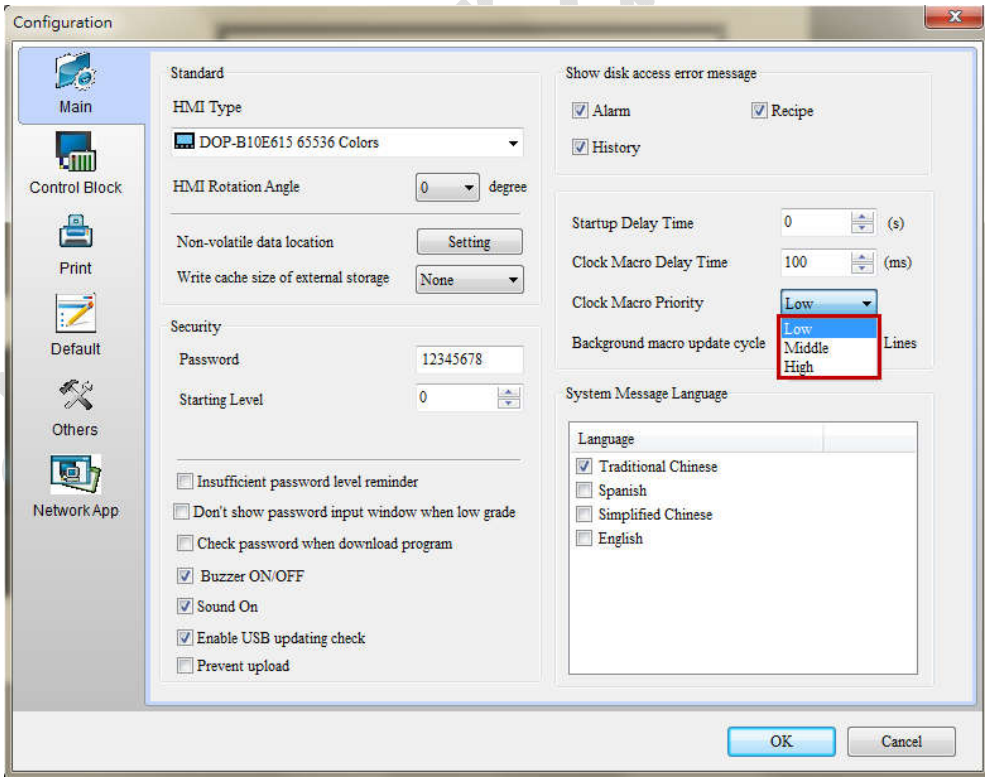
دستورات این ماکرو نیز به طور پیوسته در طول عملکرد HMI اجرا خواهند شد. جهت دسترسی به Clock Macro وارد منو Option شده و Clock Macro را انتخاب کنید ، مطابق شکل زیر:

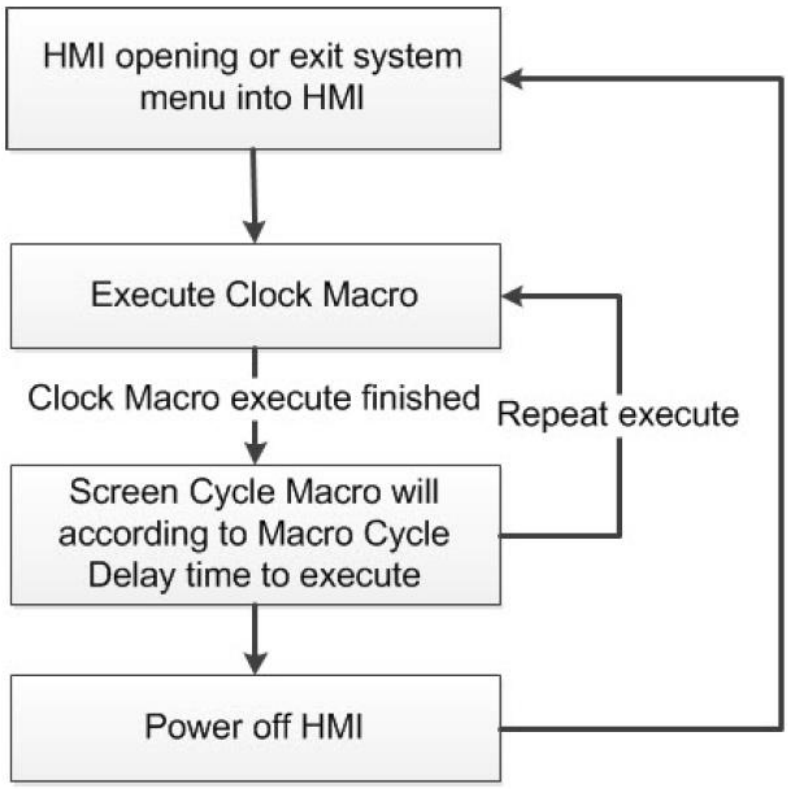


بر خلاف دستورات Background Macro ، دستورات clock Macro از خط ابتدایی تا انتهایی در هر سیکل اجرا می شوند و نیاز به تعیین خطوط برای اجرا شدن در هر سیکل را ندارد و مانند Screen Cycle Macro اجرای دستورات بر اساس زمان Clock Macro Delay Time صورت می گیرد .



همچنین می توانید برای Clock Macro الویت (Priority) را به صورت (High , Mediem , Low) تعیین کنید. با تعیین الویت زمان دقیق تاخیر (Delay) در اجرای برنامه مشخص می شود.

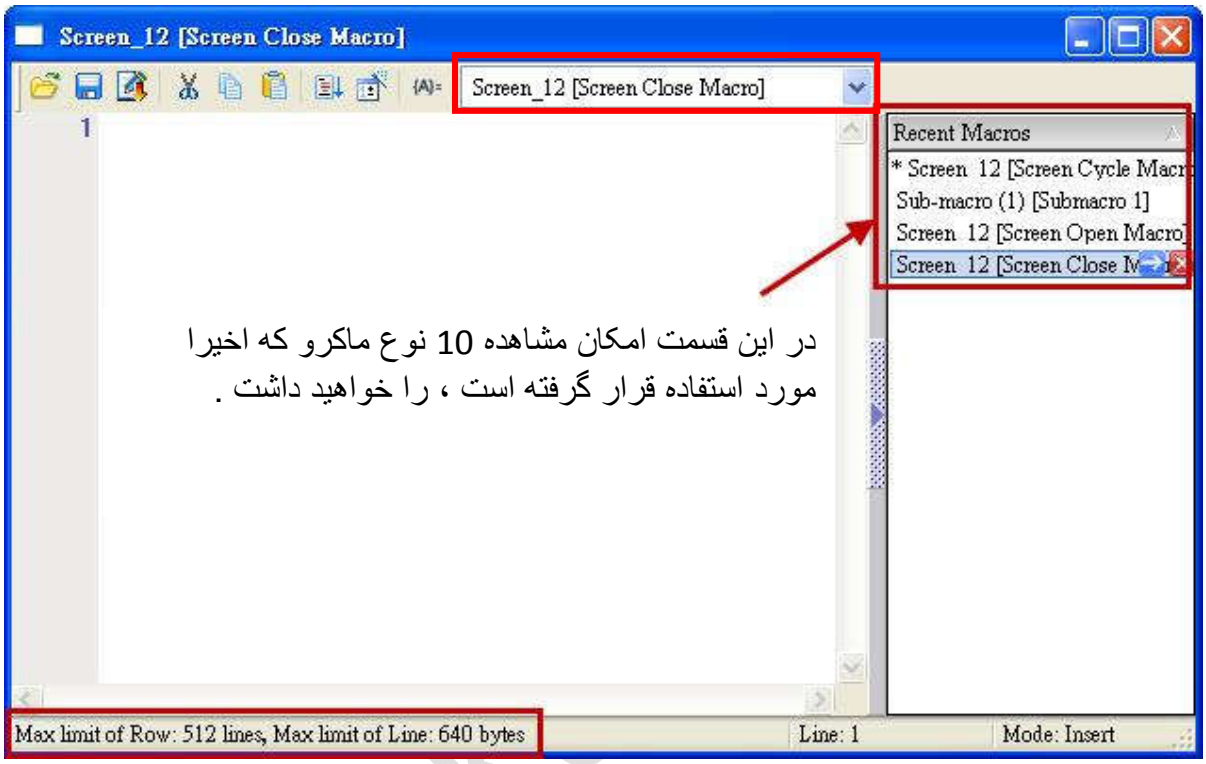




فلو چارت اجرای برنامه Clock Macro

1. پنجره نگارش برنامه ماکرو

بعد از انتخاب نوع ماکرو پنجره ای مانند شکل زیر نشان داده خواهد شد . برای هر نوع ماکرو می توانید 512 خط دستور بنویسید و در هر خط دستور می توانید حد اکثر 640 بایت کاراکتر قرار دهید . جهت ویرایش ماکرو های نوشته شده به همین پنجره مراجعه کنید و با انتخاب صحیح نوع ماکرو وارد برنامه شده و آن را ویرایش کنید.



2-1. نوار ابزار پنجره ماکرو نویسی :

در ادامه به شرح گزینه ها و دستورات نوار ابزار پنجره نگارش ماکرو خواهیم پرداخت :

- Open :

گشودن برنامه های ماکرو با فرمت *.txt و *.mro

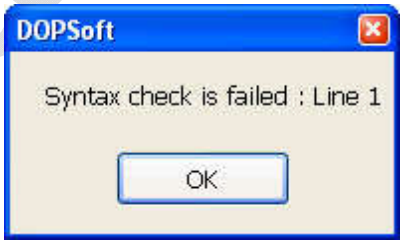
- Save :

ذخیره سازی برنامه ها ماکرو جهت استفاده مجدد با فرمت *.txt

- Download :

دانلود برنامه نوشته شده یا ویرایش شده در پروژه

با انتخاب گزینه Yes ، علاوه بر به روز رسانی برنامه های ماکرو دستورات نوشته شده را بررسی می شوند و در صورت صحیح نبودن دستورات پیام زیر مشاهده خواهد شد :

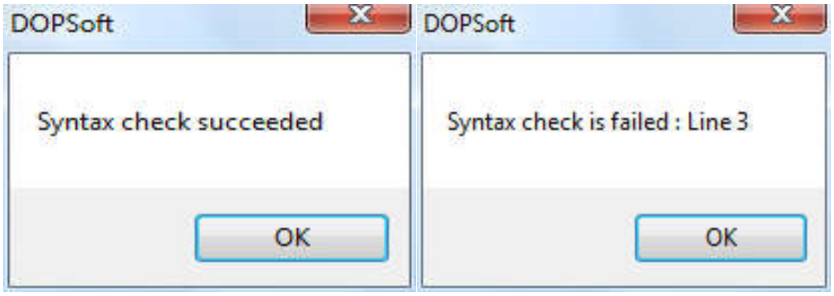


- Cut & Copy & Paste :

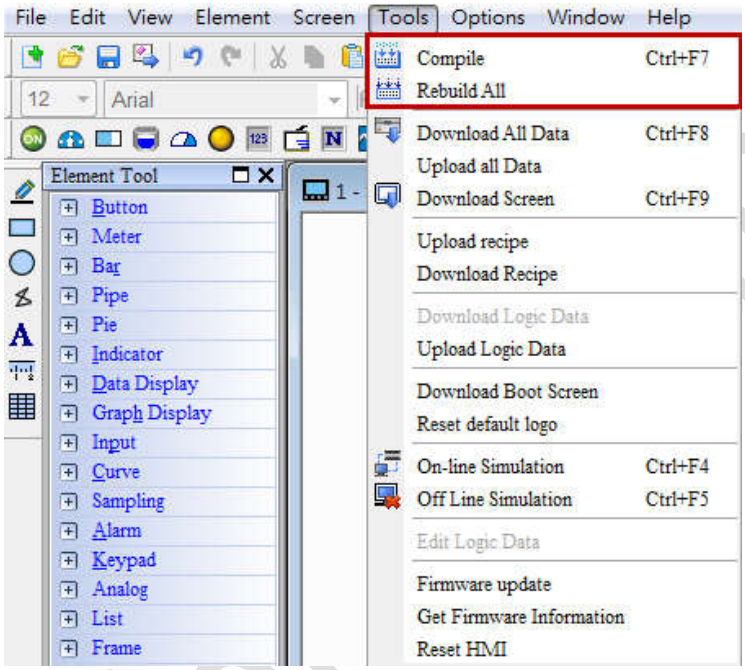
دستورات ویرایش ، جهت بریدن یا کپی کردن برنامه

- Syntax Check :

بررسی صحت برنامه نویسی دستورات

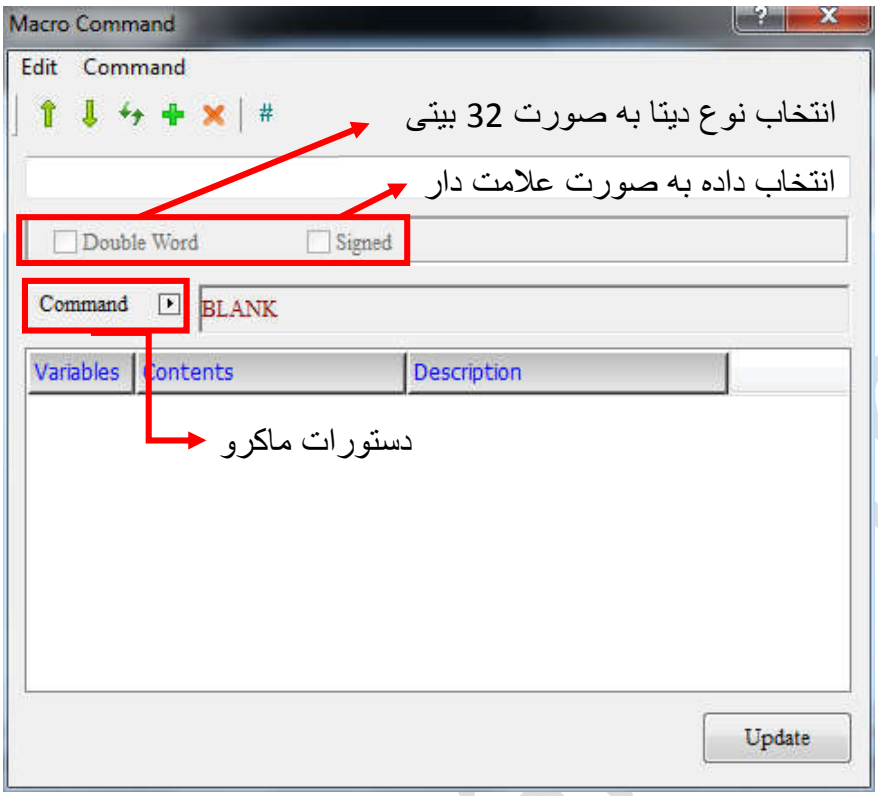


روش دیگر برای بررسی صحت برنامه ماکرو ، کامپایل کردن کل برنامه نوشته شده در DOPSoft می باشد. برای دسترسی به این گزینه وارد منو Tools شده و Compile یا Rebuild All را انتخاب کنید ، مانند شکل زیر :



- Macro Wizard :

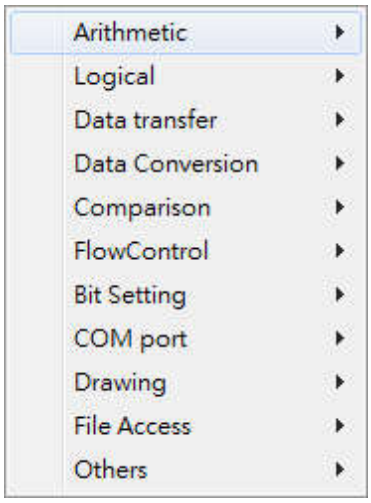
پنجره برنامه نویسی برای استفاده آسان تر از دستورات



Unsigned	Unsigned number
Signed	Signed number
WORD	16 bit data
DW（DOUBLE WORD, DWORD）	32 bit data

دیتا در ماکرو نویسی می تواند به صورت 16 بیتی (Word) یا 32 بیتی (Double Word) باشد و می تواند از نوع علامت دار (Signed) یا بدون علامت (Unsigned) باشد.

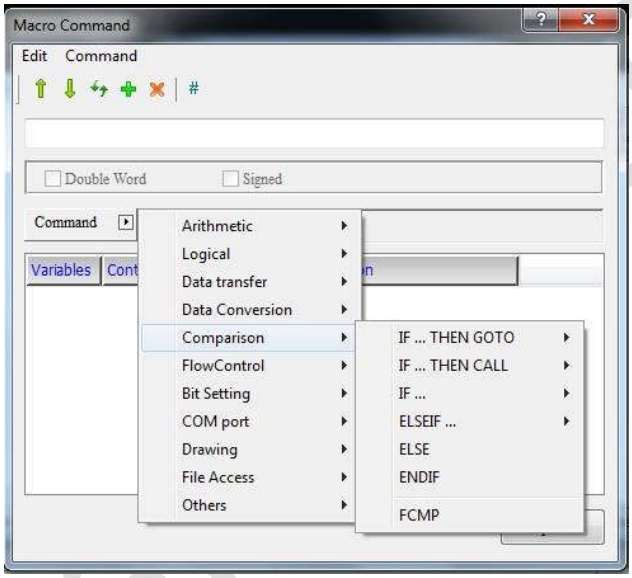
دستورات ماکرو در Command :

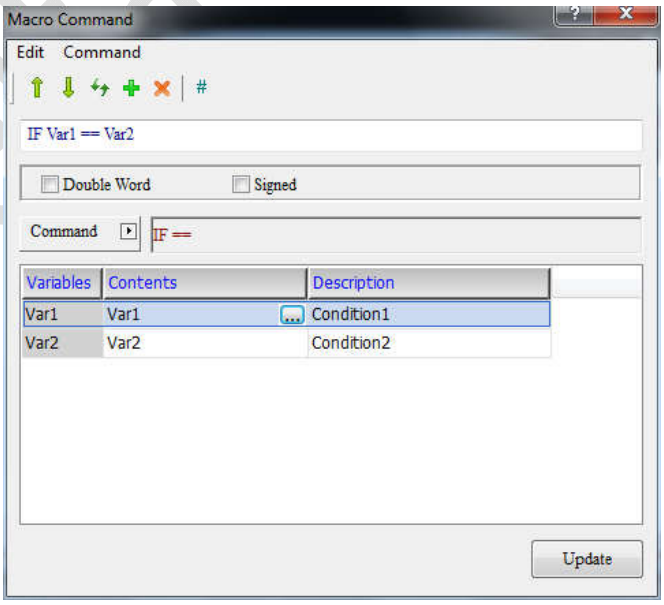


با انتخاب هر یک از دستورات موجود در بخش Command ، متغییر ها و پارا متر های مورد نیاز برای دستور نشان داده می شود که متناسب با برنامه مورد نظرتان آن را تکمیل کنید.

مثال :

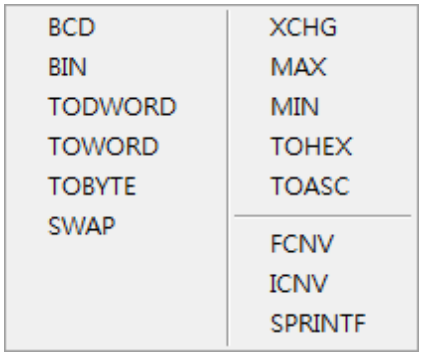
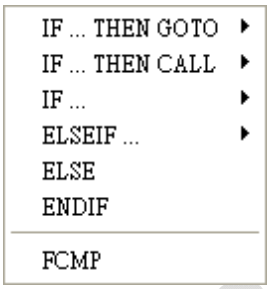
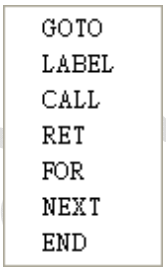
در این مثال از دستور If استفاده شده است :



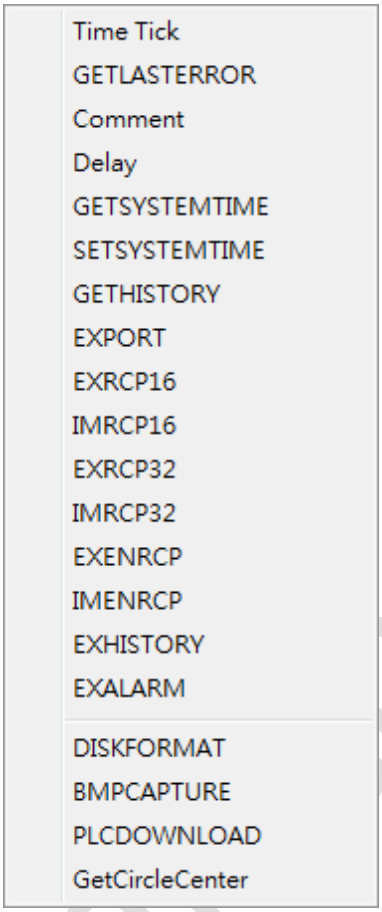


دستورات در پنجره COMMAND به 11 دسته کلی تقسیم می شود ، که در جدول زیر معرفی شده اند :

Arithmatic	دستورات ریاضی برای دیتاهای Integer وFloating + - * / % MUL64 ADDSUMW FADD FSUB FMUL FDIV FMOD SIN COS TAN COT SEC CSC
Logical	دستورات منطقی | && ^ NOT << >>
Data Transfer	دستورات انتقال و جابجایی داده MOV BMOV ArrayCopy FILL FILLASC STRCAT FMOV

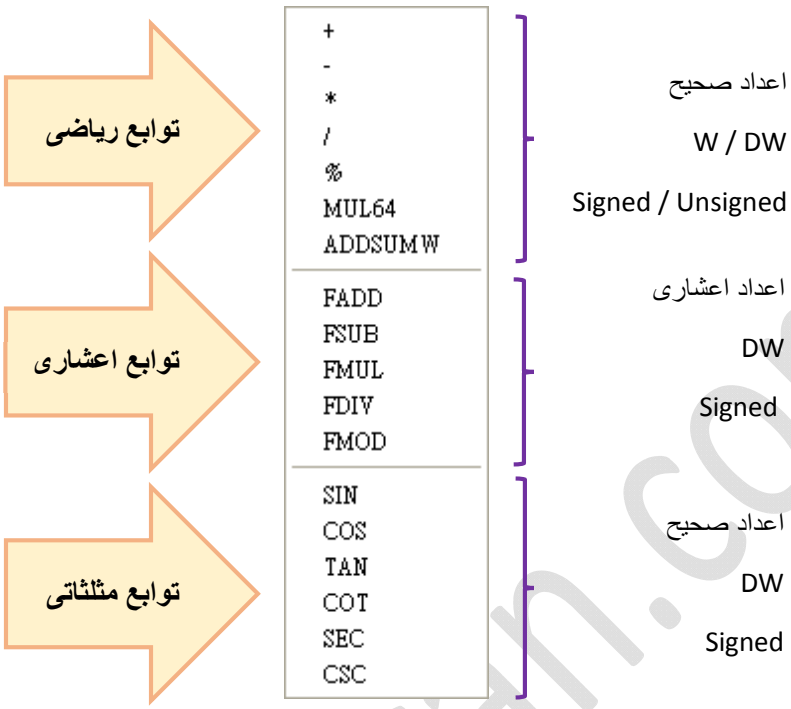
Data Conversion	دستورات تبدیل و تغییر نوع داده 
Comparision	دستورات شرطی 
Flow Control	دستورات کنترل برنامه و ایجاد یا فراخوانی ریز برنامه ها 
Bit Setting	دستورات بیتی 

COM Port	دستورات کنترل و تنظیم COM port INITCOM ADDSUM XORSUM PUTCHARS GETCHARS SELECTCOM CLEARCOMBUFFER CHRCHKSUM LOCKCOM UNLOCKCOM STATIONON STATIONOFF IPON IPOFF
Drawing	دستورات ترسیمی RECTANGLE LINE POINT CIRCLE
File Access	دستورات دسترسی به فرجه ها ****** FileSlotRead FileSlotWrite FileSlotRemove FileSlotGetLength FileSlotExport FileSlotImport

Others	دستورات تنظیم زمان و EXPORT و IMPORT و سایر دستورات 

دستورات Arithmetic :

دستورات ریاضی از جمله جمع ، تفریق ، ضرب ، تقسیم و توابع مثلثاتی . در ادامه دستوراتی که نیاز به توضیح دارند معرفی شده اند .



دستورات اعداد صحیح :

+ : جمع

- : تفاضل

* : ضرب

/ : خارج قسمت تقسیم دو متغییر (Var2 و Var3) را در Var1 ذخیره می کند .

% : باقی مانده تقسیم دو متغییر (Var2 و Var3) را در Var1 نمایش می دهد .

MUL64 : ضرب اعداد به صورت 16 بیتی و 32 بیتی ، می تواند تنیجه ضرب را به صورت 64 بیتی (ضرب دو عدد 32 بیتی) نمایش دهد . (Var1 می تواند تا 4 رجیستر را اشغال کند)

ADDSUM : مجموع چند رجیستر را محاسبه می کند .

دستورات اعشاری :

FADD : جمع اعداد اعشاری

FSUB : تفاضل اعداد اعشاری

FMULL : ضرب اعداد اعشاری

FDIV : این دستور خارج قسمت تقسیم دو متغییر اعشاری را نشان می دهد .

FMOD : باقی مانده تقسیم دو عدد اعشاری را در Var1 نمایش می دهد .

دستورات مثلثاتی :

SIN : سینوس یک زاویه بر حسب درجه را محاسبه می کند و در Var1 ذخیره می کند .

COS : کسینوس یک زاویه بر حسب درجه را محاسبه می کند و در Var1 ذخیره می کند .

TAN : تانژانت یک زاویه بر حسب درجه را محاسبه می کند و در Var1 ذخیره می کند .

COT : کتانژانت یک زاویه بر حسب درجه را محاسبه می کند و در Var1 ذخیره می کند .

SEC : سکانت (1/Cosα) یک زاویه بر حسب درجه را محاسبه می کند و در Var1 ذخیره می کند .

SCS : کسکانت (1/Sinα) یک زاویه بر حسب درجه را محاسبه می کند و در Var1 ذخیره می کند .

در توابع مثلثاتی ورودی به صورت دسیمال و علامت دار می باشد (Signed) و خروجی به صورت اعشاری یا Floating می باشد .

مثال : ADD

جمع دو متغییر :

هر سه متغییر از نوع اعداد صحیح می باشند.

Variables	Contents	Description
Var1	Var1	Sum
Var2	Var2	Addend
Var3	Var3	Augend

Var1 = Var2 + Var3

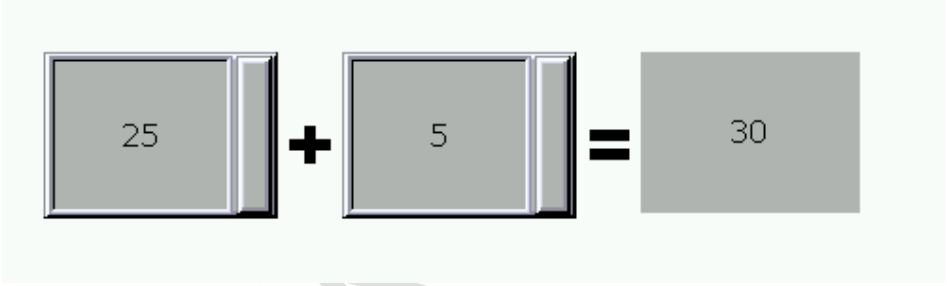
نتیجه دو متغییر در آدرسی که به Var1 اختصاص می دهید ، ذخیره می شود.

متغییر دوم (یک رجیستر یا عدد ثابت)

متغییر اول (یک رجیستر یا یک عدد ثابت)

$0	Sum
$4	Addend
$8	Augend

```
$0 = $4 + $8
```



مثال : TAN

توابع مثلثاتی ، محاسبه تانژانت یک زاویه

Variables	Contents	Description
Var1	Var1	Tangent
Var2	Var2	Angle

اعداد اعشاری و از نوع Double Word به صورت علامت دار Signed می باشند.

Var1 = Tangent (Var2)

متغییر دوم (DW-Signed)

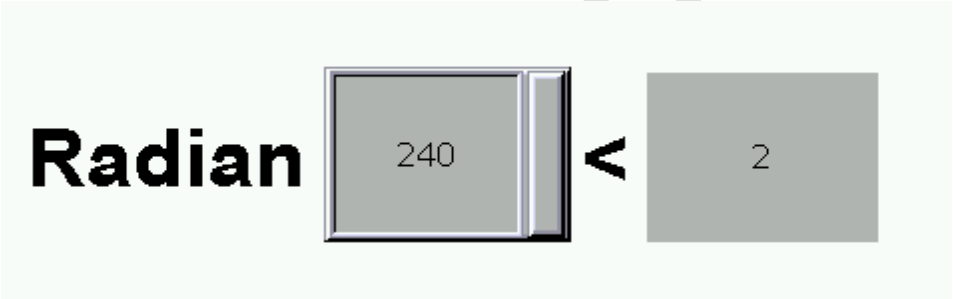
از نوع اعداد صحیح بر حسب رادیان

تانژانت زاویه Var2 در Var1 ذخیره می شود.

نتیجه محاسبات اعشاری خواهد بود.

$10	Tangent
$12	Angle

```
$10 = TAN($12) (Signed DW)
```



مثال : FDIV

محاسبات اعشاری ، تقسیم دو عدد اعشاری

Variables	Contents	Description
Var1	Var1	Diffrence
Var2	Var2	Subtrahend
Var3	Var3	Minuend

Var1 = FDIV (Var2 , Var3)

خارج قسمت تقسیم دو متغییر Var2 و Var3 در Var1 ذخیره می شود.

متغییر دوم و سوم (DW-Signed)

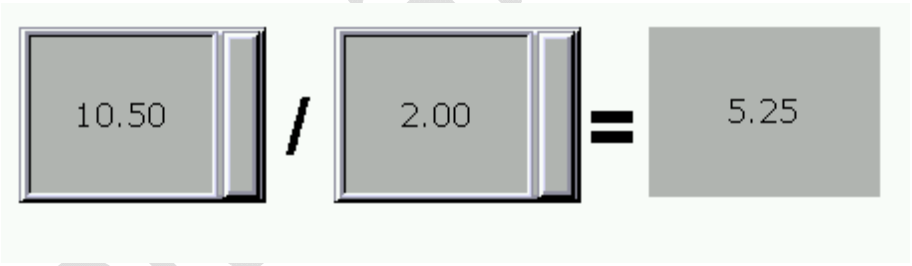
از نوع اعداد اعشاری علامت دار

Var2 : مقسوم

Var3 : مقسوم علیه

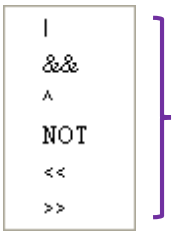
$40	Quotient
$32	Dividend
$34	Divisor

```
$40 = FDIV($32, $34) (Signed DW)
```



دستورات Logical :

عملگرهای منطقی OR ، AND ، NOT و ...



W / DW

Unsigned

| : نتیجه OR منطقی دو متغییر

Bitwise OR Operation characteristic	
Operator	**Result**
0 OR 0	0
0 OR 1	1
1 OR 0	1
1 OR 1	1

&& : نتیجه AND منطقی دو متغییر

Bitwise AND Operation characteristic	
Operator	**Result**
0 OR 0	0
0 OR 1	0
1 OR 0	0
1 OR 1	1

^ : نتیجه XOR منطقی دو متغییر

Bitwise XOR Operation characteristic	
Operator	**Result**
0 OR 0	0
0 OR 1	1
1 OR 0	1
1 OR 1	0

NOT : نتیجه NOT منطقی یک متغییر

Bitwise NOT Operation characteristic	
Operator	**Result**
NOT 0	1
NOT 1	0

>> : این تابع بیت های کم ارزش یک رجیستر را به سمت بیت های با ارزش شیفت می دهد (شیف به سمت چپ). تعداد بیت برای شیفت در Var3 تنظیم می شود .

<< : این تابع بیت های با ارزش یک رجیستر را به سمت بیت های کم ارزش شیفت می دهد (شیفت به سمت راست). تعداد بیت برای شیفت در Var3 تنظیم می شود .

مثال : &&

AND منطقی دو متغییر با هم

Variables	Contents	Description
Var1	Var1	Result
Var2	Var2	Logical Variable
Var3	Var3	Logical Variable

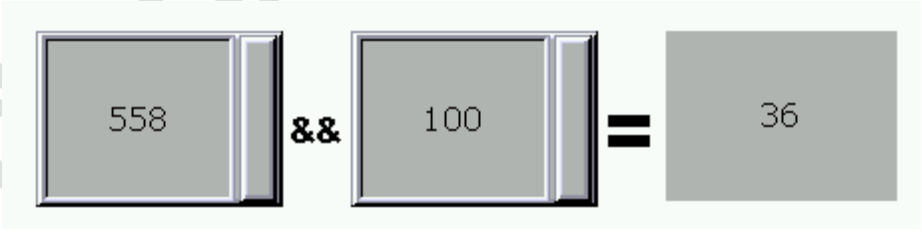
Var1 = Var2 && Var3

نتیجه AND منطقی متغییر Var2 و Var3 در Var1 ذخیره می شود.

متغییر دوم و سوم (W/DW-Unsigned) از نوع اعداد صحیح بدون علامت

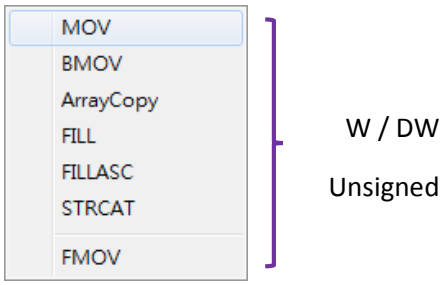
$100	Result
$101	Logical Variable
$102	Logical Variable

```
$100 = $101 && $102
```



دستورات Data Transfer :

دستورات جابجای یک یا تعدادی رجیستر



MOV : برای انتقال دیتا یک یا دو رجیستر به یک یا دو رجیستر دیگر مورد استفاده قرار می گیرد . دیتا می تواند به صورت Signed یا Unsigned باشد .

BMOV : برای انتقال دیتا به صورت 16 بیتی یا یک رجیستری به چندین رجیستر به طور همزمان استفاده می شود .

ArrayCopy : برای انتقال تعدادی رجسیتر یا یک Array به تعدادی رجیستر دیگر یا یک Array دیگر مورد استفاده قرار می گیرد .

FILL : دیتا Var2 را در رشته ای که آدرس آن در Var1 قرار داده شده است ذخیره می کند . طول دیتا در این دستور در Var3 مشخص می شود .

FILLASC : این دستور دیتا یک رشته را به کد اسکی هگز کاراکتر ها تبدیل می کند .

STRCAT : دیتا یک رشته را که آدرس آن در Var3 و طول آن در Var4 مشخص می شود را به یک رشته دیگر که آدرس آن در Var2 مشخص شده ، منتقل می کند . Var1 وضعیت عملکرد دستور را مشخص می کند . در صورتی که در عملکرد دستور مشکلی نباشد عدد یک در Var مشخص می شود .

FMOV : برای جابجایی دیتا به صورت اعشاری مورد استفاده قرار می گیرد . در دستورات قبلی دیتا به صورت اعداد صحیح می باشد .

مثال : BMOV

جابجایی تعدادی رجیستر با استفاده از دستور BMOV

Variables	Contents	Description
Var1	Var1	Destination
Var2	Var2	Source
Var3	Var3	Length

BMOV (Var1 , Var2 , Var3)

Var1 متغییر اول که باید به Var2 منتقل شود.

Var1 : مبدا و Var2 : مقصد

تعداد رجیستر های که باید منتقل شوند را در Var3 مشخص کنید.

می تواند عدد ثابت (Constant) یا متغییر باشد.

در این مثال سه رجیستر $50 و $51 و$52 به ترتیب به سه رجیستر $60 و $61 و $62 منتقل می شوند .

$50	Destination
$60	Source
3	Length

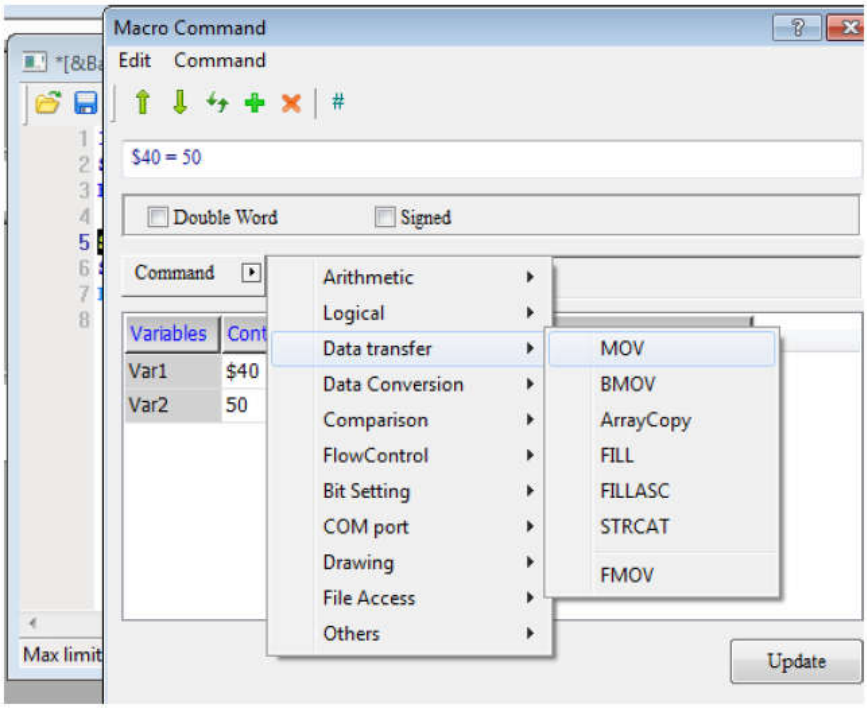
```
BMOV($50, $60, 3)
```

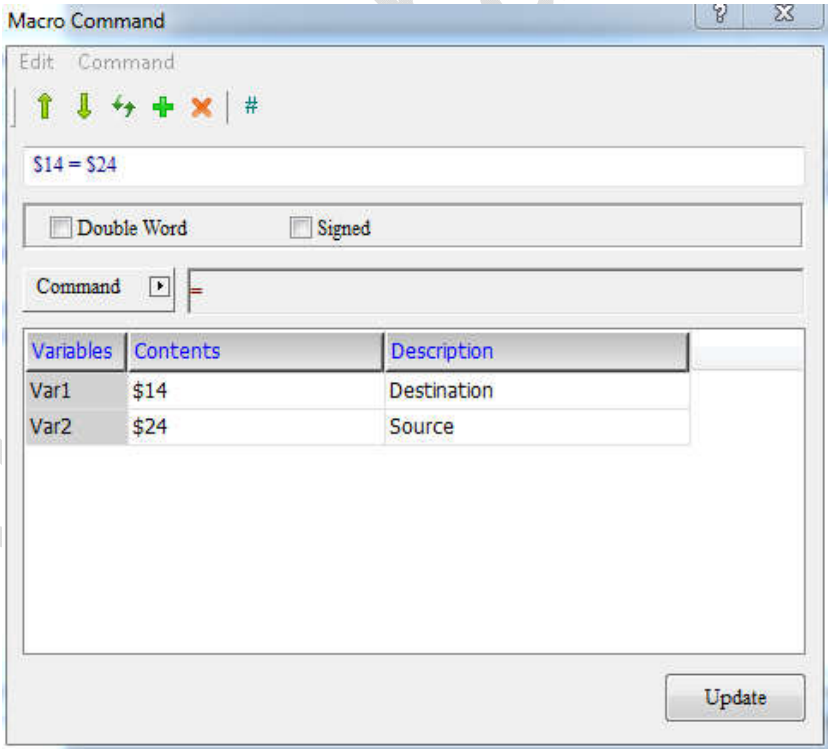
مثال : MOV

از این دستور برای جابجایی دیتا استفاده می شود .

$14 = $24 انتقال دیتا از $24 به $14

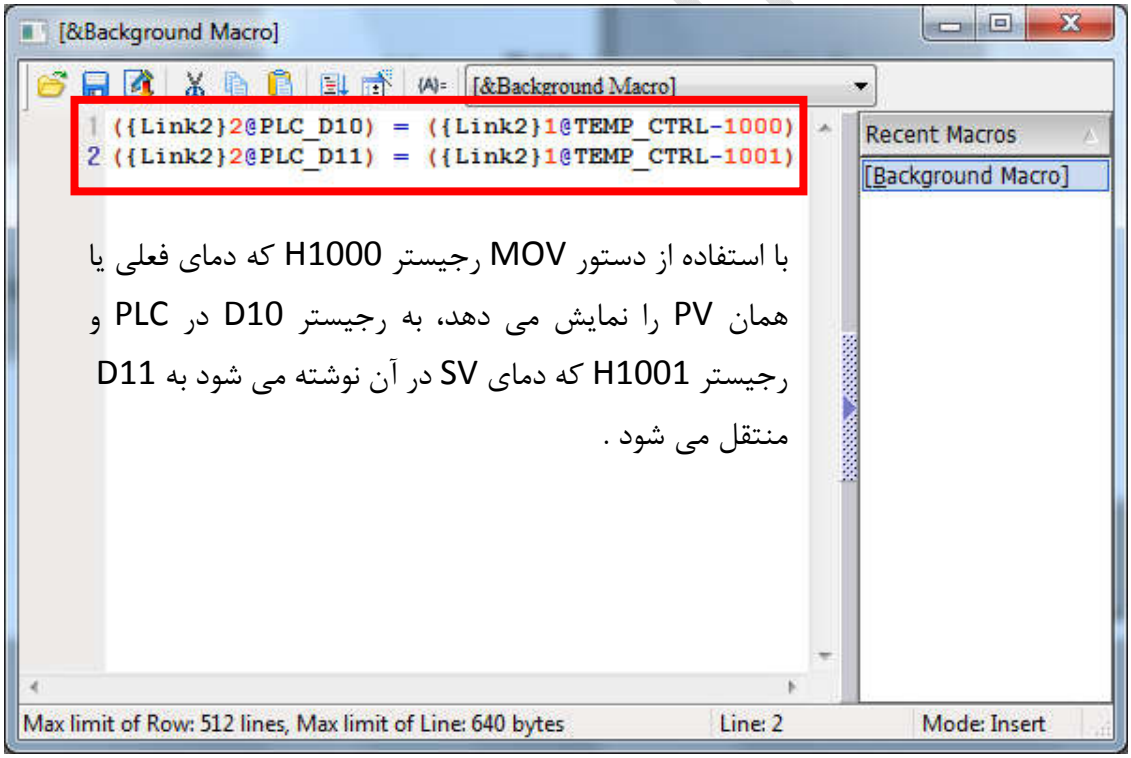
$12 = 0 محتوا $12 ، صفر می شود





مثال : MOV

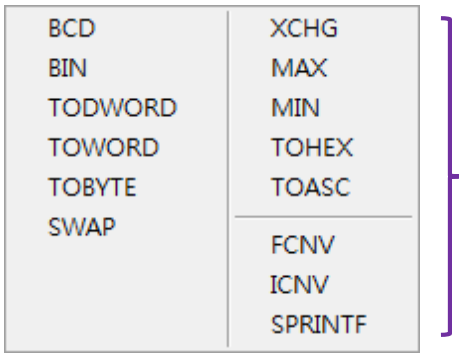
فرض کنید یک PLC و یک کنترلر دما DTC1000 را در شبکه مدباس RS485 دارید ، در این شبکه HMI را به عنوان Master انتخاب کنید . برای انتقال دیتا DTC1000 به PLC از دستور MOV در ماکرو استفاده می شود . به این طریق شما دیتا DTC1000 را از طریق ماکرو نویسی و به واسطه HMI به PLC منتقل می کنید و می توانید از این اطلاعات در برنامه PLC استفاده کنید .



با استفاده از دستور MOV رجیستر H1000 که دمای فعلی یا همان PV را نمایش می دهد، به رجیستر D10 در PLC و رجیستر H1001 که دمای SV در آن نوشته می شود به D11 منتقل می شود .

دستورات Data Conversion :

دستورات تبدیل اعداد 16 بیتی به 32 بیتی و تبدیل فرمت های هگز ، اسکی ، اعشاری ، باینری ، BCD و...



BCD : دستور تبدیل دیتا دسیمال به BCD

BIN : دستور تبدیل دیتا BCD به دسیمال

TODWORD : دستور تبدیل یک Word به دو Word (DW)

TOWORD : دستور تبدیل یک بایت به ورد

TOBYTE : دستور تبدیل ورد به بایت

SWAP : جای بایت کم ارزش (Low byte) و پر ارزش (High byte) را در یک Word عوض می کند.

XCHG : جابجایی محتوا یک یا چند رجیستر با یکدیگر (طول دیتا قابل تنظیم است)

MAX : محتوا دو متغییر به صورت Word یا Double Word را بررسی کرده ماکزیمم مقدار آنها را در Var3 ذخیره می کند .

MIN : محتوا دو متغییر به صورت Word یا Double Word را بررسی کرده مینیمم مقدار آنها را در Var3 ذخیره می کند .

TOHEX : تبدیل کاراکتر اسکی به عدد هگز

TOASC : تبدیل کد هگز کاراکتر به کد دسیمال کاراکتر

FCNV : تبدیل عدد Integer به اعشاری

ICNV : تبدیل اعداد اعشاری به Integer

SPRINTF : دستور دریافت و نمایش رشته (String)

مثال : FCNV

Variables	Contents	Description
Var1	Var1	Floating
Var2	Var2	Integer

تبدیل اعداد Integer به اعداد اعشاری با تابع FCNV

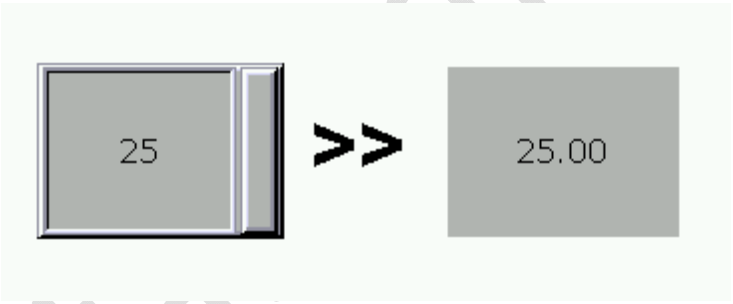
Var1 = FCNV (Var2) (Signed DW)

Var1 : نتیجه تبدیل و عدد به صورت اعشاری

Signed/DW – Integer عدد

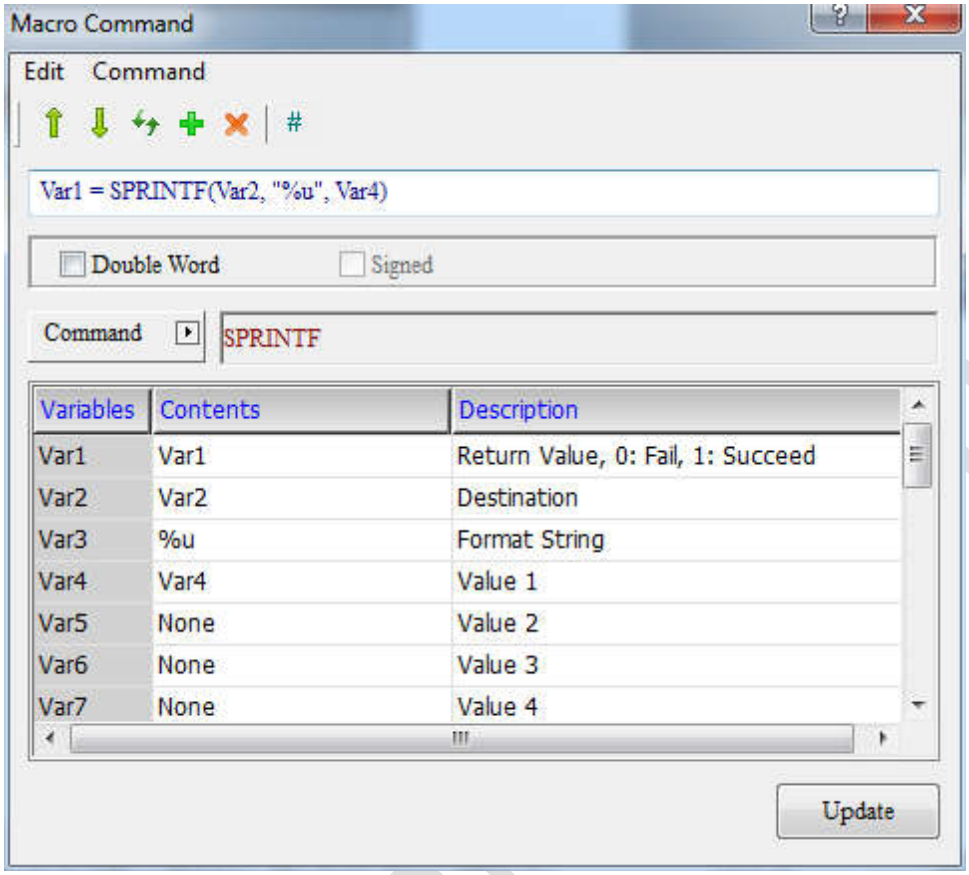
$80	Floating
$82	Integer

```
$80 = FCNV($82) (Signed DW)
```



مثال : SPRINTF

از دستور SPRINTF برای دریافت رشته (String) استفاده می شود . هر رشته می تواند شامل اعداد ، حروف و علائم باشد .



Var1 : این متغییر وضعیت عملکرد دستور را مشخص می کند و در صورتی که دریافت رشته با موفقیت انجام شده باشد مقدار یک در این متغییر ذخیره می شود ، در غیر این صورت مقدار یک را نمایش می دهد .

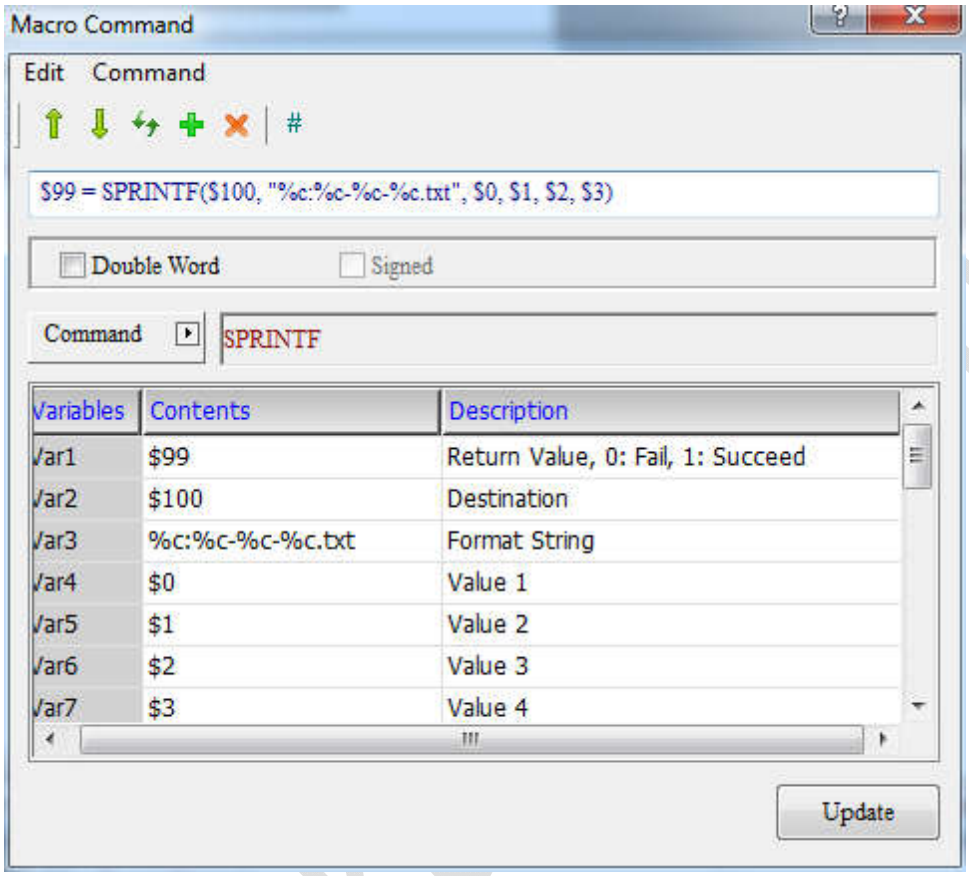
Var2 : آدرسی که می خواهید نتیجه نهایی رشته در آن ذخیره شود را در این قسمت وارد نمایید .

Var3 : فرمت نمایش رشته را در این قسمت تعیین کنید . منظور از فرمت دیتا نوع دیتا و نمایش آن می باشد . دیتا می تواند به صورت حروف یا اعداد همراه با علائم اختصاری استفاده شود . برای حروف و برخی از علائم اختصاری باید کد اسکی آن را وارد نمایید .

در شکل نشان داده شده در صفحه بعد کد اسکی مختص هر کاراکتر را به صورت دسیمال و هگز (هگزا دسیمال) مشاهده می کنید .

Var4 تا Var24 : به این متغییر ها آدرس ورودی را بدهید . هر حرف یا عدد یا علامت اختصاری که با کد اسکی مشخص می شود در یک رجیستر قرار می گیرد .

برای مثال در صفحه Background Macro از بخش Data Conversion دستور SPRINTF را انتخاب کنید و تنظیمات را به صورت نشان داده شده در شکل زیر انجام دهید :

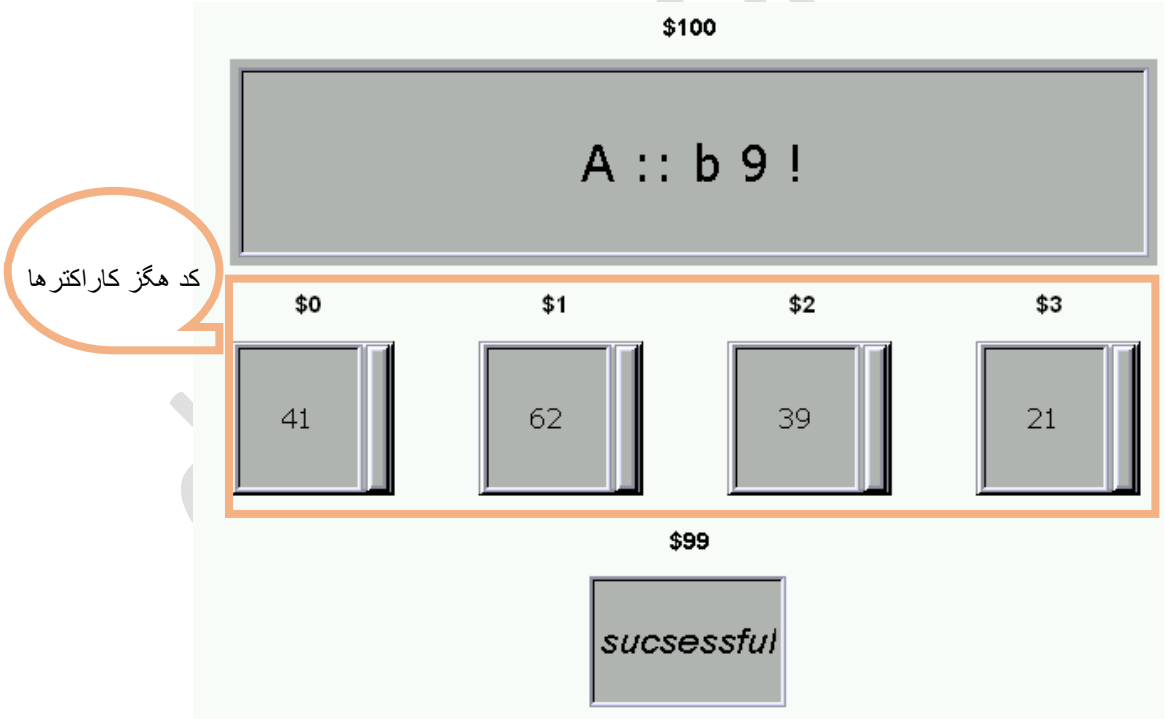


برای Var1 که وضعیت دستور را نمایش می دهد می توانید از کلید استفاده کنید و یا یک Numeric Display قرار دهید تا مقدار $99 را نمایش دهد و یا اینکه از Multistate Indicator برای نمایش دو وضعیت صفر و یک استفاده کنید . در این مثال از یک کلید Maintaind استفاده شده است . در رجیستر $100 مقدار نهایی رشته ، بعد از دریافت ورودی ها ذخیره می شود و برای نمایش آن از یک دستور Character Display برای مشاهده نتیجه نهایی استفاده کنید . آدرس های $0 تا $3 آدرس رجیستر های ورودی می باشند که به ازا هر رجیستر یک کاراکتر با وارد کردن کد اسکی کاراکتر ، وارد رشته می شود. برای وارد نمودن هر کاراکتر ، با استفاده از دستور Numeric Entry کد دسیمال یا هگز مربوط به کاراکتر را وارد نمایید . برای وارد کردن کد دسیمال فرمت دستور را به صورت دسیمال و برای وارد کردن کد هگز فرمت دستور را به صورت هگزا دسیمال انتخاب کنید . در جدول زیر کد اسکی هر کاراکتر را به دو صورت دسیمال و هگز مشاهده می کنید :

Dec	Hex	Char
32	20	sp
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

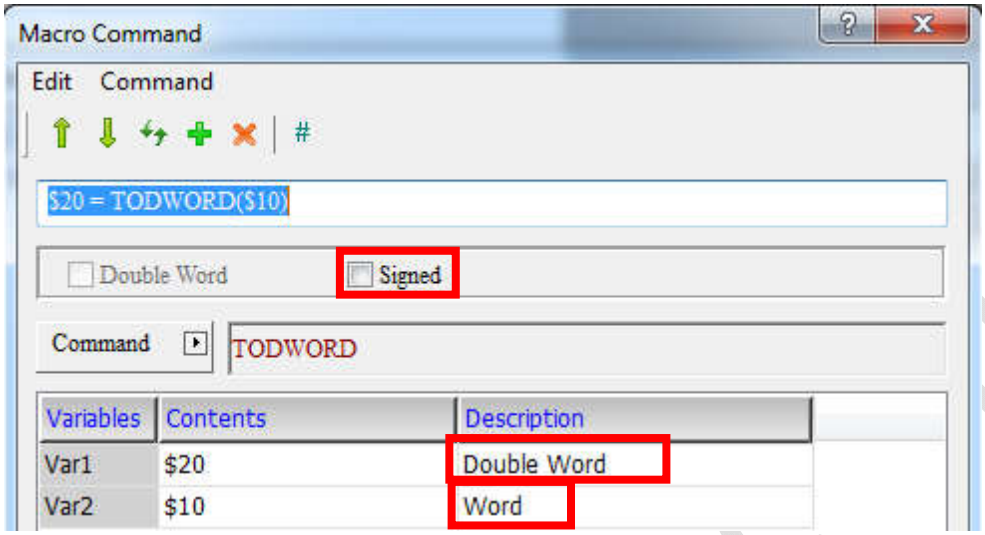
Dec	Hex	Char
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

Dec	Hex	Char
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	⌂



مثال : TODWORD

دستور TOWORD دیتا Word را به Double Word تبدیل می کند . کافیست در Var2 آدرس رجیستر مورد نظر را وارد نمایید تا به صورت 32 بیتی (DW) در Var1 ذخیره شود.



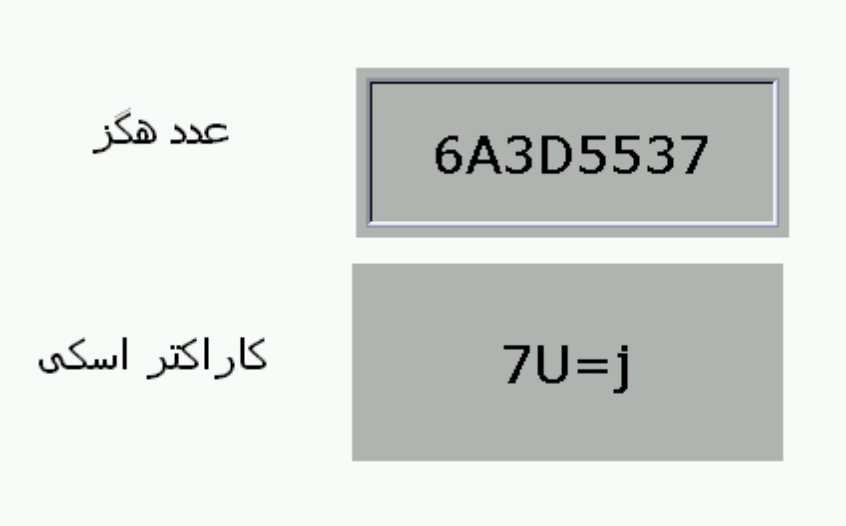

```
$20 = TODWORD($10)
```

مثال : TOHEX

دیتا فقط می تواند به صورت Word باشد .

Command	TOHEX	
Variables	Contents	Description
Var1	$10	HEX Value
Var2	$12	Start Address

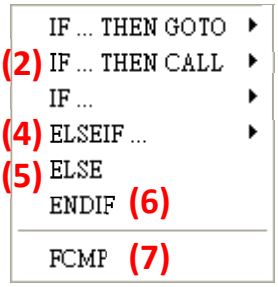
```
$10 = TOHEX($12)
```



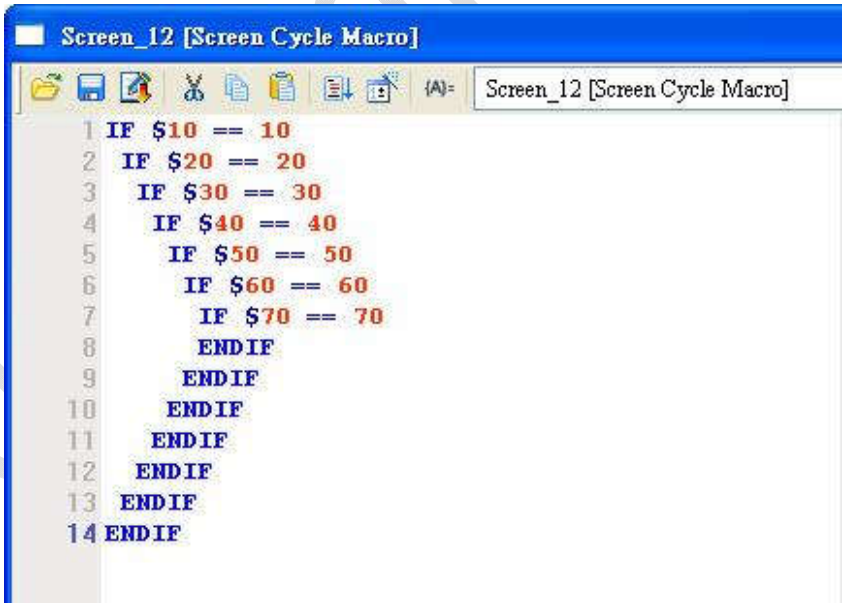
دستورات Comparlson :

دستورات مقایسه ای برای قرار دادن شرط در برنامه های ماکرو و مقایسه دو متغییر با یکدیگر

1. در صورت بر قراری شرط وارد Lable شده و دستورات آنرا اجرا می کند.
2. در صورت بر قراری شرط به Submacro مراجعه می کند.
3. دستورات شرطی به صورت IF,ENDIF ، فقط در صورت برقراری شرط دستورات مورد نظر را اجرا می کند.
4. دستورات شرطی به صورت IF,ELSEIF ، در صورت برقرار نبودن شرط دستورات دیگری را اجرا می کند.
5. زمانی که شرط IF یا ELSEIF برقرار نباشند ، دستورات ELSE اجرا خواهد شد.
6. پایان حلقه شرط
7. شرط برای مقایسه دو عدد اعشاری



در استفاده از حلقه های تو در تو ، حداکثر می توانید از 7 دستور شرط استفاده کنید ، مانند شکل زیر:



مثال : IF .... THEN GOTO

در این مثال از دستور IF > استفاده شده است.

IF ==	IF AND == 0
IF !=	IF AND != 0
IF >	IF == ON
IF >=	IF == OFF
IF <	IFB == ON
IF <=	IFB == OFF

Variables	Contents	Description
Var1	Var1	Condition1
Var2	Var2	Condition2
Var3	Var3	Label Name

W / DW

Signed / Unsigned

IF Var1 > Var2 THEN GOTO LABEL (Var3)

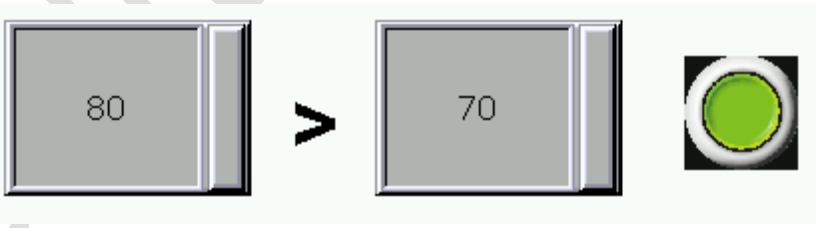
متغییر اول

متغییر دوم

نام LABEL

$90	Condition1
$91	Condition2
1	Label Name

```
IF $90 > $91 THEN GOTO LABEL 1
LABEL 1
BITON $85.0
ENDIF
```



مثال : IF .... THEN CALL

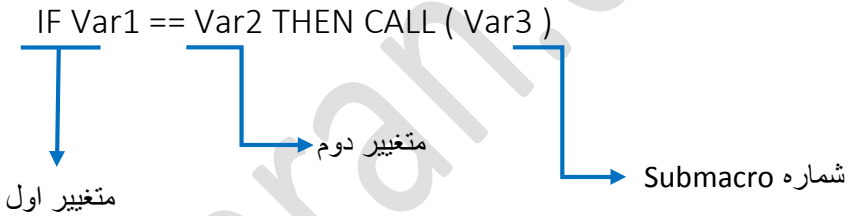
دستور مورد استفاده : IF == CALL

```
IF == CALL
IF != CALL
IF > CALL
IF >= CALL
IF < CALL
IF <= CALL
IF AND == 0 CALL
IF AND != 0 CALL
IF == ON CALL
IF == OFF CALL
```

Variables	Contents	Description
Var1	Var1	Condition1
Var2	Var2	Condition2
Var3	Var3	Call Sub-Macro

W / DW

Signed / Unsigned



$92	Condition1
$93	Condition2
1	Call Sub-Macro

```
IF $92 == $93 THEN CALL 1
ENDIF
```

در صورت برقراری شرط دستوراتی که در Submacro نوشته شده است ، اجرا خواهد شد.

مثال : IF

دستورات شرط برای مقایسه یک متغییر با متغییری دیگر و یا مقایسه

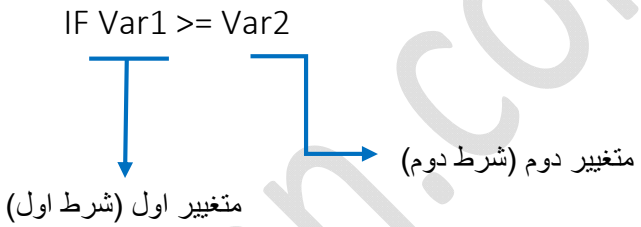
با عدد ثابت دستور مورد استفاده : IF >=

```
IF ==
IF !=
IF >
IF >=
IF <
IF <=
IF AND ==0
IF AND !=0
IF ==ON
IF ==OFF
```

Variables	Contents	Description
Var1	Var1	Condition1
Var2	Var2	Condition2

W / DW

Signed / Unsigned



$10	Condition1
$11	Condition2

```
IF $10 >= $11
$12 = $12 + 1
ENDIF
```

این برنامه با مقایسه دو متغییر در صورت بر قراری شرط مقدار یک رجیستر را یک واحد افزایش می دهد.

مثال : IF

در صورت روشن شدن کلید $150.0 ، بیت $60.0 یک می شود .

IF $150.0==on شرط

Biton $60.0 عملکرد

Endif پایان شرط

مثال : ELSEIF

از این دستور برای ایجاد شرط در راستای یک دستور IF استفاده می شود ،

اگر شرط دستور IF برقرار نباشد ، شرط دستور ELSEIF بررسی خواهد شد .

دستور مورد استفاده : IF ==

```
ELSEIF ==
ELSEIF !=
ELSEIF >
ELSEIF >=
ELSEIF <
ELSEIF <=
ELSEIF AND ==0
ELSEIF AND !=0
ELSEIF ==ON
ELSEIF ==OFF
```

Variables	Contents	Description
Var1	Var1	Condition1
Var2	Var2	Condition2

W / DW

Signed / Unsigned

IF Var1 == Var2

متغییر اول (شرط اول)

متغییر دوم (شرط دوم)

$0	Condition1
10	Condition2

```
IF $10 >= $11
$12 = $12 + 1
ELSEIF $0 == 10
ENDIF
```

برای برنامه نوشته شده در مثال قبل یک شرط دیگر با استفاده از دستور ELSEIF قرار داده شده است ، در صورتی که شرط دستور IF بر قرار باشد ،حلقه خاتمه می یابد و به خط اول برنامه برگشته و شرط را مجددا بررسی می کند . اما اگر شرط IF برقرار نباشد ، برنامه به خط بعد رفته و شرط دستور ELSEIF را بررسی می کند .

* دستور ELSEIF در پایان شرط به دستور ENDIF نیاز ندارد .

مثال : ELSEIF

اگر کلید $150.0 روشن شود .بیت $60.0 (عملکرد 1) فعال می شود ولی در صورت روشن نبودن کلید $150.0 بیت $60.1 (عملکرد 2) روشن خواهد شد .

IF $150.0==on شرط1

Biton $60.0 عملکرد 1

elseIF $150.0==off شرط2

Biton $60.1 عملکرد 2

Endif پایان شرط

مثال : ELSE

در این مثال از دستور IF && استفاده شده است این دستور نتیجه AND دو حافظه را بررسی می کند و در صورتی که نتیجه صفر شود ، شرط برقرار بوده و باقی دستورات را اجرا می کند .

```
IF ($10 && $20) == 0
BITON $0.0
ELSE
BITOFF $0.0
ENDIF
```

مثلا اگر مقادیر صفر را به هر دو حافظه بدهید و نتیجه AND صفر خواهد شد و بیت $0.0 روشن می شود ولی اگر عدد یک را به $10 بدهید و عدد 3 را به $20 ، در نتیجه حاثل AND یک شده و بیت $0.0 خاموش می شود .

مثال : ELSE

در صورتی که در یک حلقه برنامه شرطی هیچ کدام از دستورات IF و ELSEIF بر قرار نباشند می توان از دستور ELSE استفاده کرده و یک شرط دیگر در برنامه قرار دهید . این دستور هیچ متغییری نمی پذیرد ، فقط کافی ست در خط بعد از دستور ، برنامه مورد نظرتان را بنویسید . دستور ELSE نیز مانند دستور ELSEIF نیاز به دستور ENDIF ندارد ولی حتما باید در یک حلقه IF قرار بگیرد .

```
IF $10 >= $11
$12 = $12 + 1
ELSEIF $0 == 10
ELSE
BITOFF $100.0
ENDIF
```

مثال : ELSEIF

IF $150==12 شرط

Biton $60.0 **عملکرد 1**

Else

Biton $60.1 **عملکرد 2**

Endif پایان شرط

مثال : FCMP

دستور مقایسه دو شرط با مقادیر اعشاری

Variables	Contents	Description
Var1	Var1	Return Value, 0: =, 1: >, 2: <
Var2	Var2	Condition1
Var3	Var3	Condition2

DW

Signed

Var1 = FCMP (Var2 , Var3)

مقایسه متغییر اول و دوم

نتیجه دستور مقایسه در رجیستر اختصاص داده شده به Var1 ذخیره می شود ، اگر مقار این متغییر صفر باشد نشانه تساوی ، اگر 1 باشد Var2 > Var3 و اگر مقدار آن 2 باشد Var2 < Var3 می باشد .

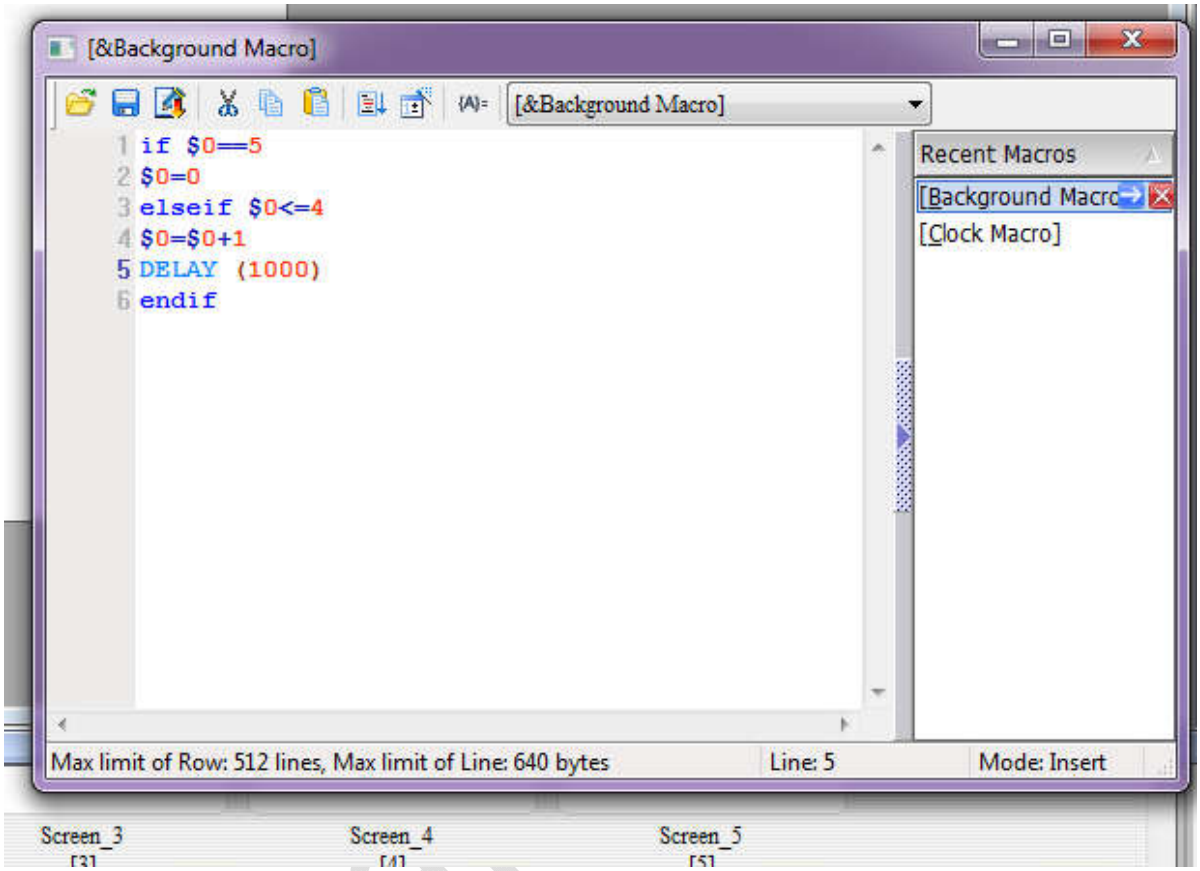
$0	Return Value, 0: =, 1: >, 2: <
$1	Condition1
$3	Condition2

```
$0 = FCMP($1, $3) (Signed DW)
```

مثال : Comparision

در مثال زیر اگر مقدار رجیستر $0 برابر با 5 باشد ، محتوا رجیستر صفر می شود و در صورتی که محتوا رجیستر $0 مساوی یا کوچکتر از 4 باشد مقدار رجیستر $0 یک واحد افزایش می یابد وبعد از 1 ثانیه تاخیر برنامه دوباره به خط اول برگشته و محتوا رجیستر $0 را چک می کند.

می توانید با برنامه نویسی مانند شکل زیر ، یک سیکل تکرار با تعداد دفعات تکرار مشخص بسازید .



مثال : برنامه کاربردی Piping

در این مثال با استفاده از ماکرو نویسی برای المان های Pipe ، با زدن یک کلید که در واقع Valve اصلی است ، سیال در مسیر لوله ها جریان پیدا کرده و وارد مخزن می شود .

```
IF {Link2}1@X1 == ON
$10 = 2
$11 = 1
{Link2}1@D10 = {Link2}1@D10 + 10
DELAY(500)
ENDIF
IF {Link2}1@D10 == 700
BITOFF {Link2}1@X1
ENDIF
IF {Link2}1@X1 == Off
$10 = 0
$11 = 0
ENDIF
```

آدرسی که به لوله عمودی شکل (Pipe7) اختصاص داده شده است ، $10 و آدرس لوله افقی (Pipe6) $11 می باشد . برای آنکه بتوانید جهت حرکت سیال در لوله ها را مشخص کنید باید به آدرس اختصاص داده شده در دستور مقادیر 1 یا 2 را بدهید .

ورودی Read Address	Pipe6	Pipe7
مقدار یک (بیت1 فعال شود)	حرکت از راست به چپ	حرکت از پائین به بالا
مقدار دو (بیت2 فعال شود)	حرکت از چپ به راست	حرکت از بالا به پائین

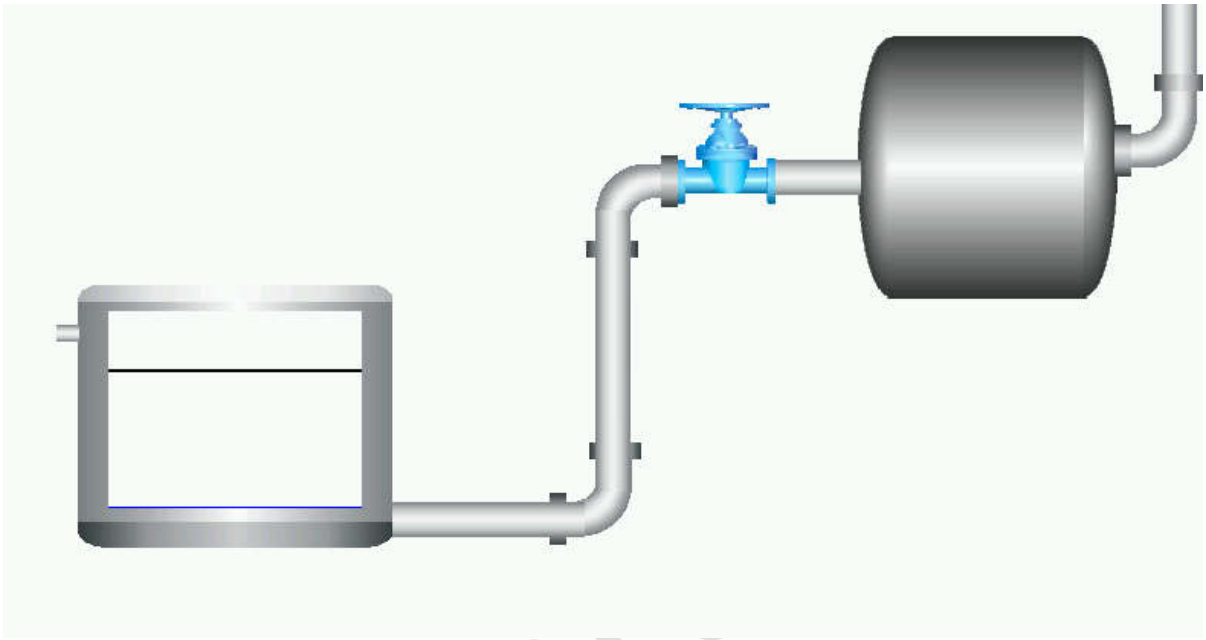
طبق شرط IF اگر کلید X1 که همان شیر می باشد ، زده شود (IF {Link2}1@X1 == ON) ، جهت حرکت لوله عمودی از بالا به پائین ($10 = 2) و جهت حرکت لوله افقی ($11 = 1) از راست به چپ تعیین می شود ، سپس متناسب با حرکت سیال در لوله ها مخزن پر می شود ({Link2}1@D10 = {Link2}1@D10 + 2). در ادامه یک شرط قرار داده شده است که اگر مخزن تا حجم 700 واحد پر شد ، شیر به صورت اتوماتیک بسته شود :

```
IF {Link2}1@D10 == 700
BITOFF {Link2}1@X1
ENDIF
```

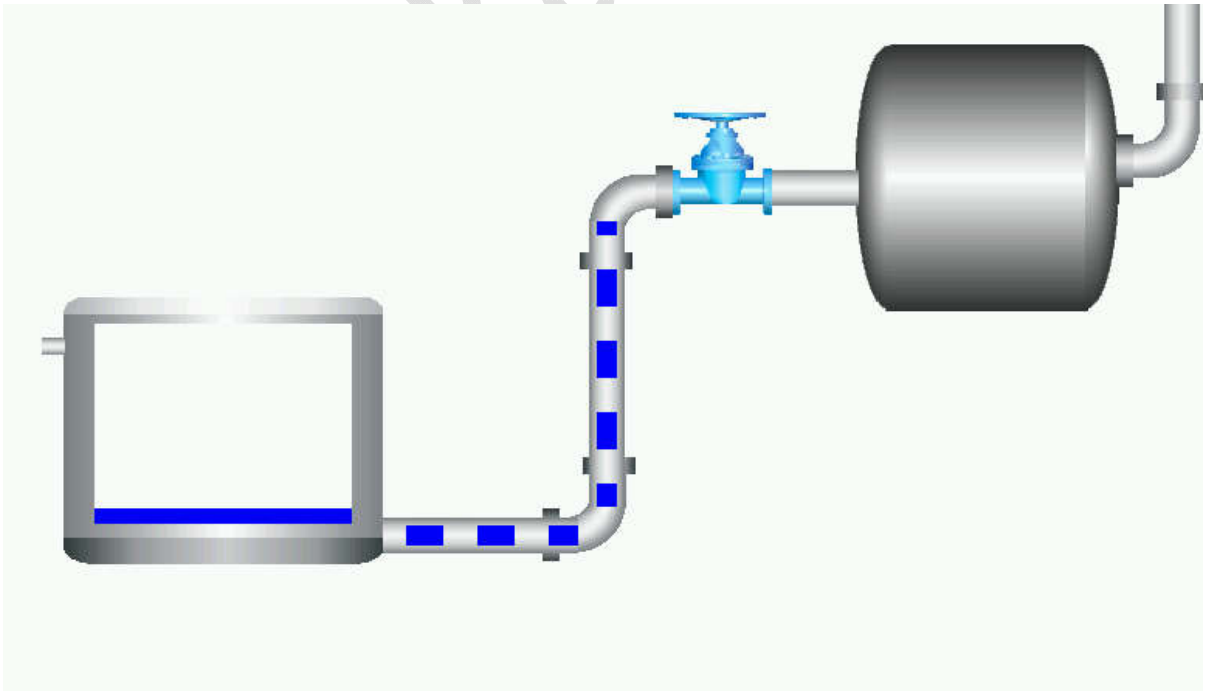
برای آنکه سیال در زمان بسته شدن شیر ، جریان نداشته باشد . شرط بسته بودن شیر ، به صورت زیر نوشته شده است :

```
IF {Link2}1@Y1 == Off
$10 = 0
$11 = 0
ENDIF
```

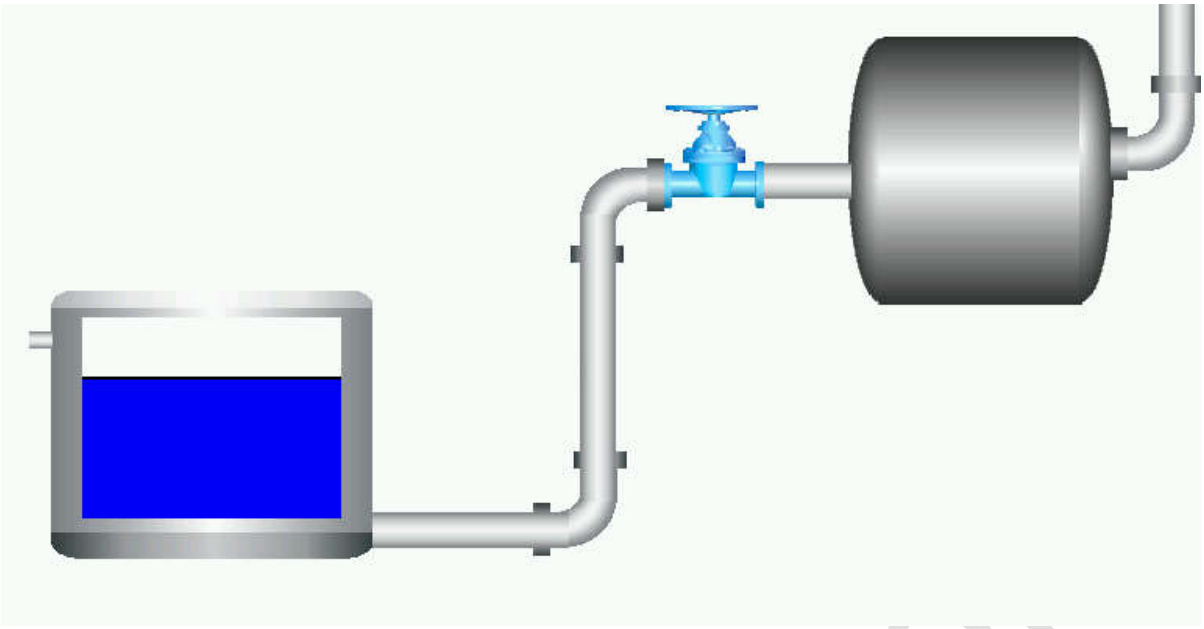
شیر بسته باشد :



شیر باز باشد :



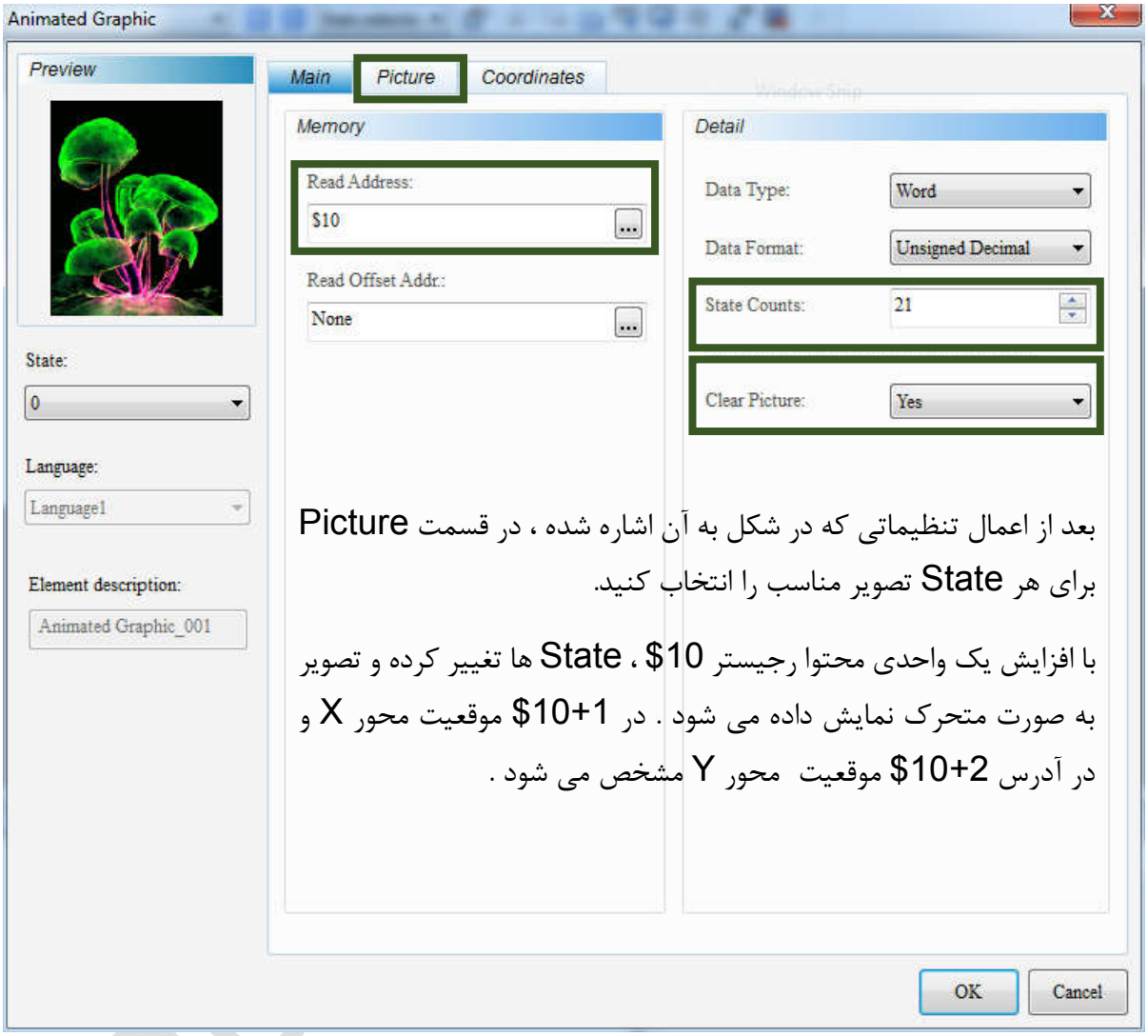
مخزن به حد مورد نظر رسیده باشد و شیر به صورت اتوماتیک بسته شود :



حال اگر بخواهید به سیستم مخزن یک شیر تخلیه

مثال : برنامه کاربردی نمایش Gif

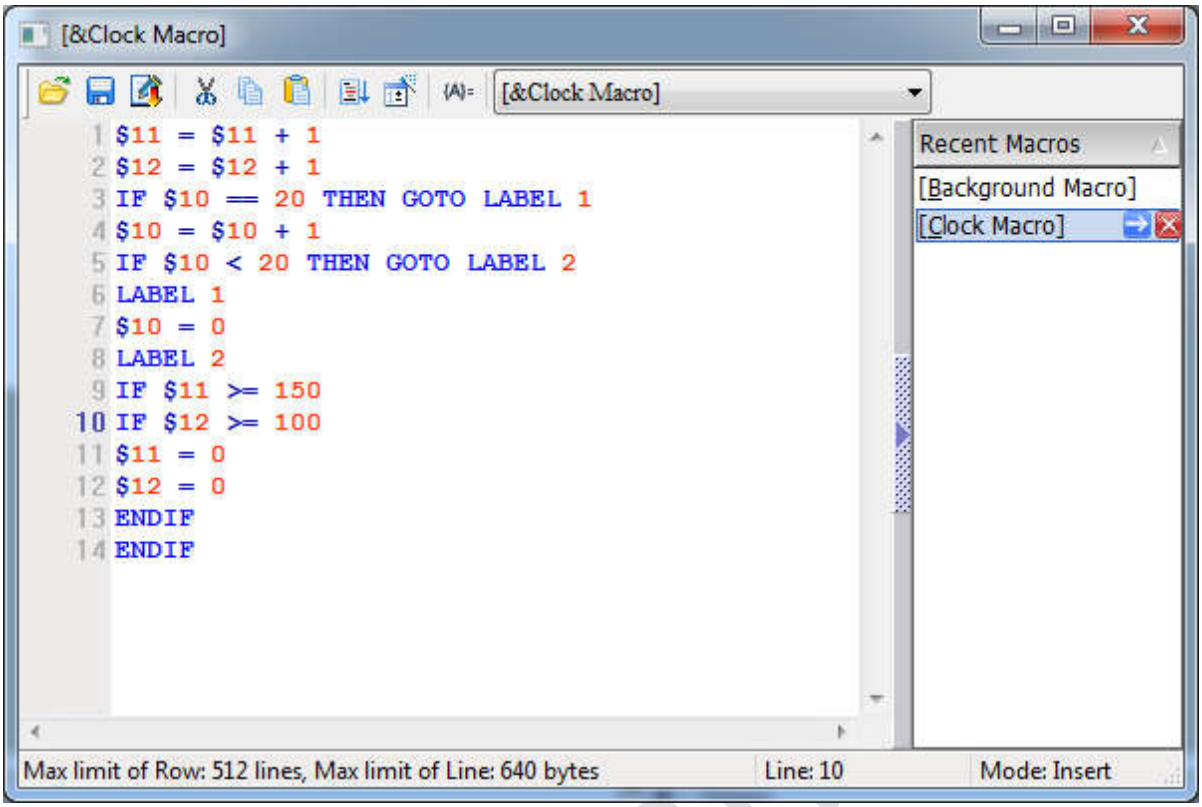
برای نمایش تصاویر متحرک با فرمت Gif. و یا حرکت یک تصویر در HMI از دستور Animated Graphic استفاده می شود . در مرحله اول باید Gif را به Picture Bank بیفزایید (مانند افزودن سایر تصاویر) . بعد از آن هر Gif تبدیل به مجموعه ای از تصاویر می شود که حالات مختلف را نمایش می دهد . برای دستورAnimated Graphic به تعداد عکس هایی که از فایل گیف در Picture Bank ایجاد شده است State تعریف کنید . تنظیمات لازم برای المان در شکل زیر نمایش داده شده است :



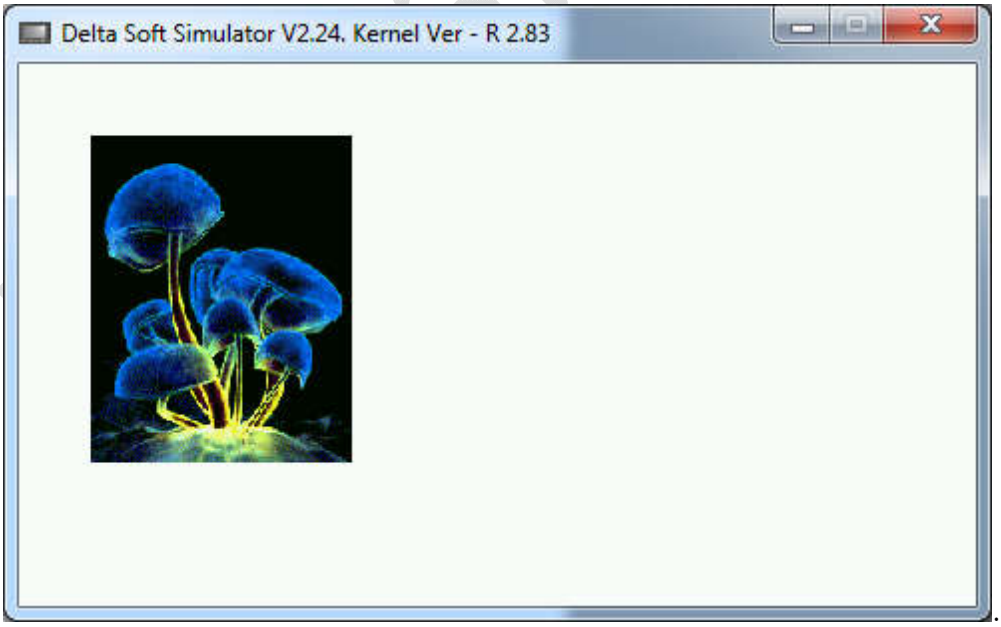
بعد از اعمال تنظیماتی که در شکل به آن اشاره شده ، در قسمت Picture برای هر State تصویر مناسب را انتخاب کنید.

با افزایش یک واحدی محتوا رجیستر $10 ، State ها تغییر کرده و تصویر به صورت متحرک نمایش داده می شود . در $10+1 موقعیت محور X و در آدرس $10+2 موقعیت محور Y مشخص می شود .

حال به برنامه نویسی در محیط ماکرو برای تغییر State ها و تغییر موقعیت X و Y نیاز دارید . در این مثال برنامه به صورت زیر نوشته شده است :



در این برنامه همزمان با تغییر موقعیت فیزیکی تصویر در صفحه ، State ها و تصویر آن نیز تغییر می کند.



دستورات Flow Control :

دستورات Flow Control شامل شش دستور GOTO ، LABEL ، CALL ، RET ، FOR ، NEXT و END برای کنترل الگوریتم و ایجاد حلقه برنامه استفاده می شود .

```
GOTO
LABEL
CALL
RET
FOR
NEXT
END
```

GOTO : دستور پرش به یک LABEL

LABEL : از این دستور برای ایجاد یک لیبل که شامل دستورات خاصی می باشد ، استفاده می شود .

CALL : دستور فراخوانی Sub MACRO

RET : دستور خروج از برنامه Sub MACRO

FOR / NEXT : FOR دستور ایجاد حلقه تکرار با تعداد مشخص می باشد و NEXT دستور پایان حلقه FOR

END : دستور پایان ، در هر قسمتی از برنامه که استفاده شود ، دستورات بعدی اجرا نخواهند شد .

مثال : GOTO & LABEL

در مثال از دستور GOTO و LABEL استفاده شده است :

```
IF {Link2}1@X0 == ON
GOTO LABEL 1
ENDIF
#LABEL 1 :
LABEL 1
{Link2}1@D10 = {Link2}1@D10 + 1
```

در صورت برقراری شرط و روشن بودن کلید X0 برنامه به LABEL 1 ، پرش می کند و دستورات نوشته شده در LABEL 1 اجرا می شوند .

* با استفاده از # در ابتدای خط برنامه می توانید کامنت یا توضیحات مورد نیاز در برنامه را بنویسید .

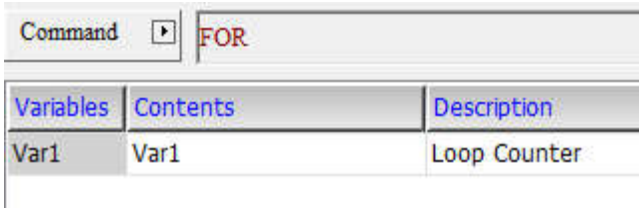
در دستور LABEL نیز باید برای لیبل نام انتخاب کنید و نام LABEL مورد نظر را در دستور GOTO برای پرش به دستورات آن لیبل وارد نمایید .

* نام لیبل فقط می تواند شامل اعداد باشد .

مثال :FOR-NEXT

حلقه FOR-NEXT :

ابتدای حلقه تکرار با دستور FOR آغاز می شود ، که باید برای آن تعداد دفعات تکرار حلقه ذکر شود ، در ادامه دستورات مورد نظر که می خواهید در هر بار اجرا شدن حلقه ، اجرا شوند را بنویسید و با دستور NEXT به حلقه FOR پایان دهید .



متغییر VAR1 تعداد دفعات تکرار حلقه را تعیین می کند و می تواند به صورت عدد ثابت یا متغییر باشد .

```
FOR 5
{Link2}1@D100 = {Link2}1@D100 + 1
NEXT
```

در برنامه نوشته شده ، در هر مرتبه تکرار حلقه یک واحد به رجیستر D100 افزوده می شود و بعد از 5 مرتبه تکرار شدن حلقه ، در مجموع 5 واحد به رجیستر D100 اضافه خواهد شد .

مثال : CALL

از دستور CALL برای فراخوانی Sub Macro ها در برنامه اصلی استفاده می شود . برنامه های روتین و پر تکرار را می توانید در SUB Macro ها بنویسید و در صورت لزوم آن ها را فراخوانی کنید .

در این مثال در Sub Macro1 ، برنامه ساده ای نوشته شده و در برنامه اصلی ماکرو فراخوانی شده است .

برنامه نویسی Sub Macro ها هم مانند برنامه نویسی در محیط های دیگر ماکرو می باشد . در Sub Macro 1 برنامه زیر را بنویسید که در آن هر یک ثانیه یک واحد به رجیستر D10 افزوده می شود .

```
{Link2}1@D10 = {Link2}1@D10 + 1
DELAY(1000)
```

سپس باید برنامه Sub Macro را در برنامه اصلی فراخوانی نمایید . در این مثال برنامه اصلی در محیط Background Macro نوشته شده است . نحوه فراخوانی Sub Macro در برنامه اصلی به صورت زیر و با استفاده از دستور CALL می باشد :

```
CALL 1
IF {Link2}1@D10 == 10
{Link2}1@D10 = 0
ENDIF
```

در این برنامه بعد از فراخوانی Sub Macro با دستور CALL یک شرط IF قرار داده شده است که اگر مقدار رجیستر D10 به عدد 10 رسید ، مقدار آن را صفر کند تا شمارش مجددا آغاز شود . چون برنامه در Background MACRO نوشته شده است ، برنامه به طور خودکار به خط اول بازگشته و مجددا تکرار می شود .

اگر در صفحه HMI یک المان Numeric Display ایجاد کنید . می توانید مقادیر رجیستر D10 را مشاهده کنید .

* از دستور RET برای خروج از برنامه SUB Macro و برگشت به برنامه اصلی استفاده می شود . این دستور که در SUB Macro نوشته می شود ، دستورات ماکرو را از خط بعد از SUB Macro ، در برنامه اصلی آغاز می کند .

دستورات Bit Setting :

در این بخش چهار دستور برای اعمال در بیت ها وجود دارد که با استفاده از آنها می توان مقدار یک بیت را صفر (OFF) ، یک (ON) یا معکوس کرد و یا مقدار بیت را نمایش داد .ورودی متغییر VAR1 تنها می تواند آدرس یک بیت را بپذیرد .

```
BITON
BITOFF
BITNOT
GETB
```

BITON : یک بیت را روشن یا یک می کند

BITOFF : یک بیت را خاموش یا صفر می کند

BITNOT : مقدار بیت را معکوس می کند . صفر را به یک و یک را به صفر تبدیل می کند .

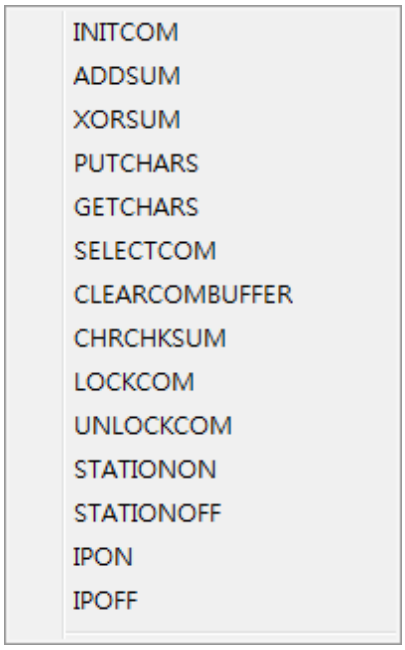
GETB : این دستور یک بیت از یک رجیستر را دریافت می کند و در Var1 ذخیره می کند .

مثال : BITON

در برنامه نوشته شده در شکل زیر در صورتی که محتوای رجیستر D20 بزرگتر یا مساوی عدد 100 باشد ، بیت X4 روشن خواهد شد و در صورت عدم برقراری شرط ، بیت X4 روشن نخواهد شد .

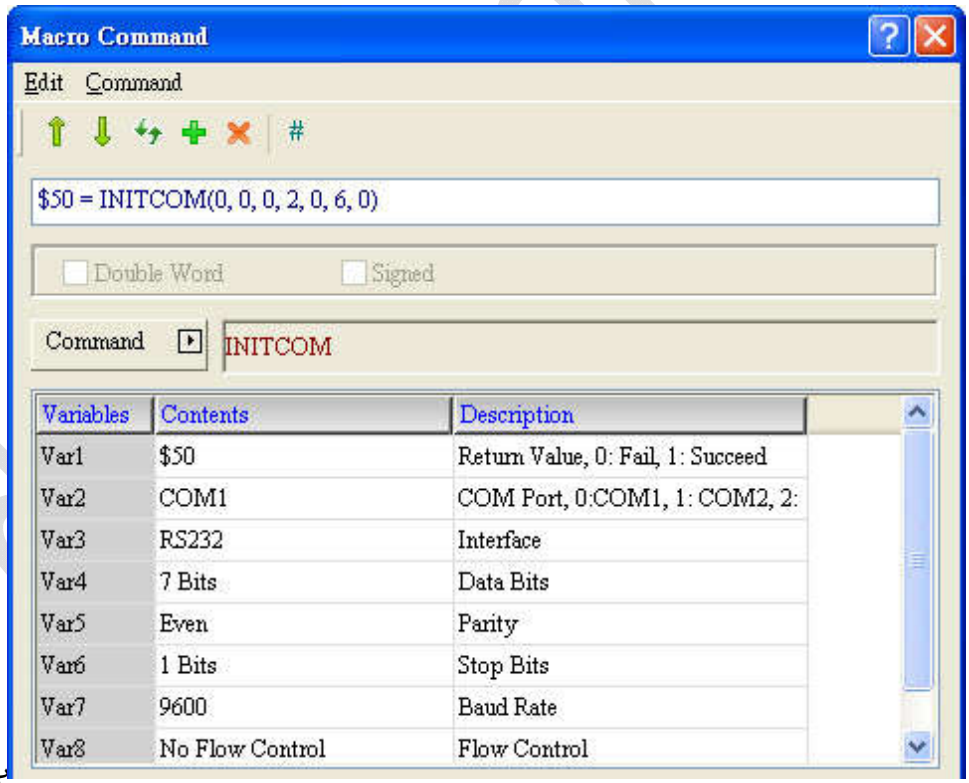
```
IF {Link2}1@D20 >= 100
BITON ({Link2}1@X4)
ENDIF
```

دستورات COM Port :



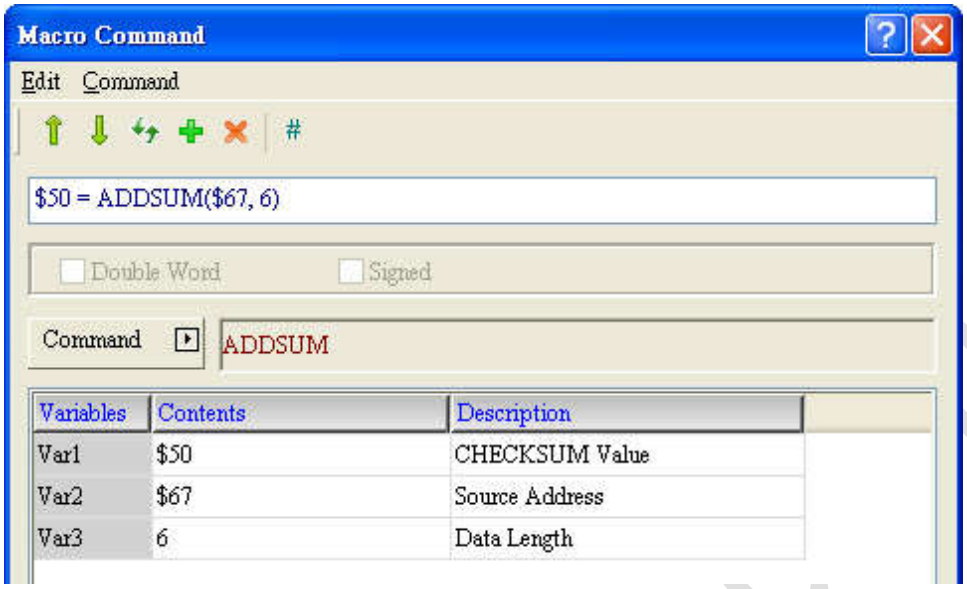
از این دستورات برای اعمال تنظیمات بر روی پورت های HMI استفاده می شود . در ادامه به شرح این دستورات پرداخته خواهد شد.

INITCOM (Initialization COM) : تنظیمات شبکه بر روی پورت های COM1 ، COM2 و COM3، که شامل Buad rate ، طول دیتا ، بیت استاپ و ... می باشد .

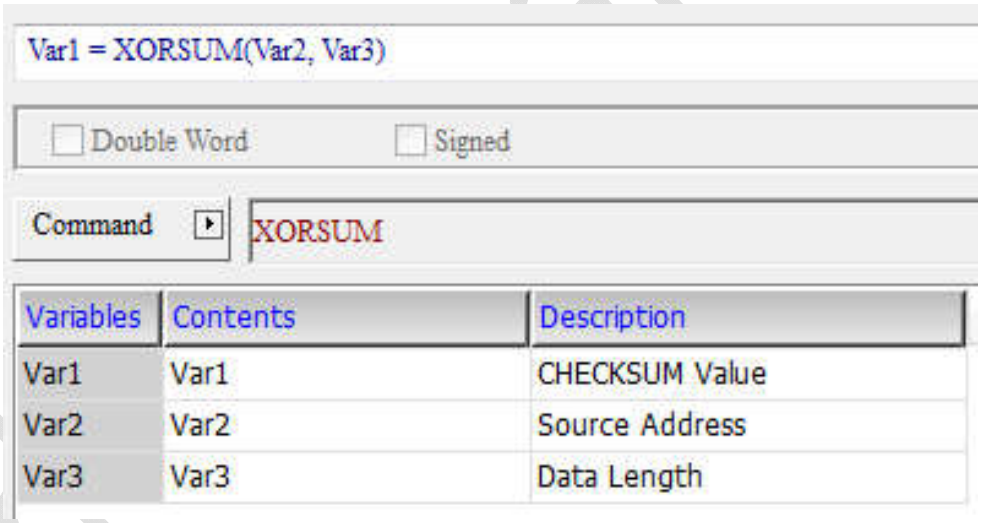


پ

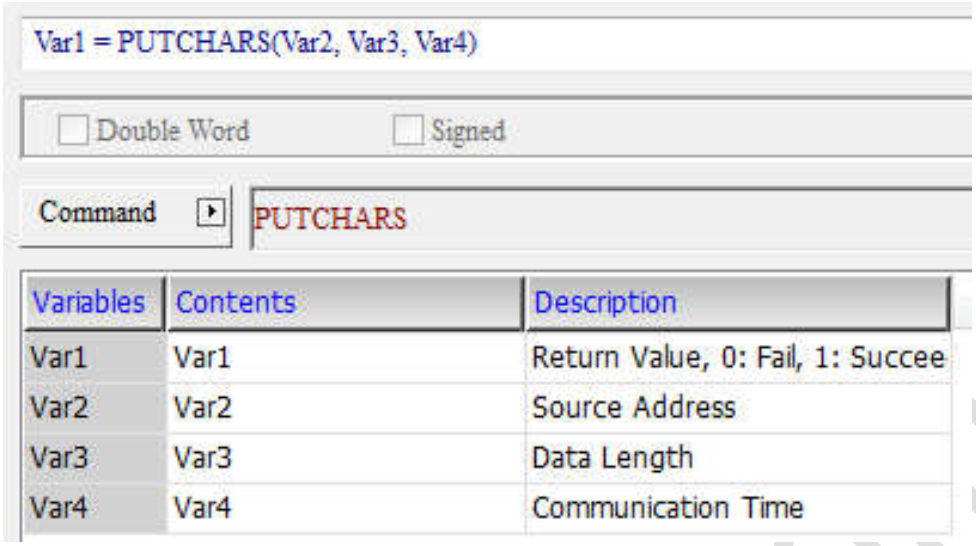
ADDSUM : این دستور مجموع چند رجیستر را در یک رجیستر ذخیره می سازد . Data Length در واقع تعداد بایت ها می باشد برای مثال در شکل زیر Data Length برابر 6 می باشد و تعداد رجیستر ها 2÷6=3 رجیستر می باشد .



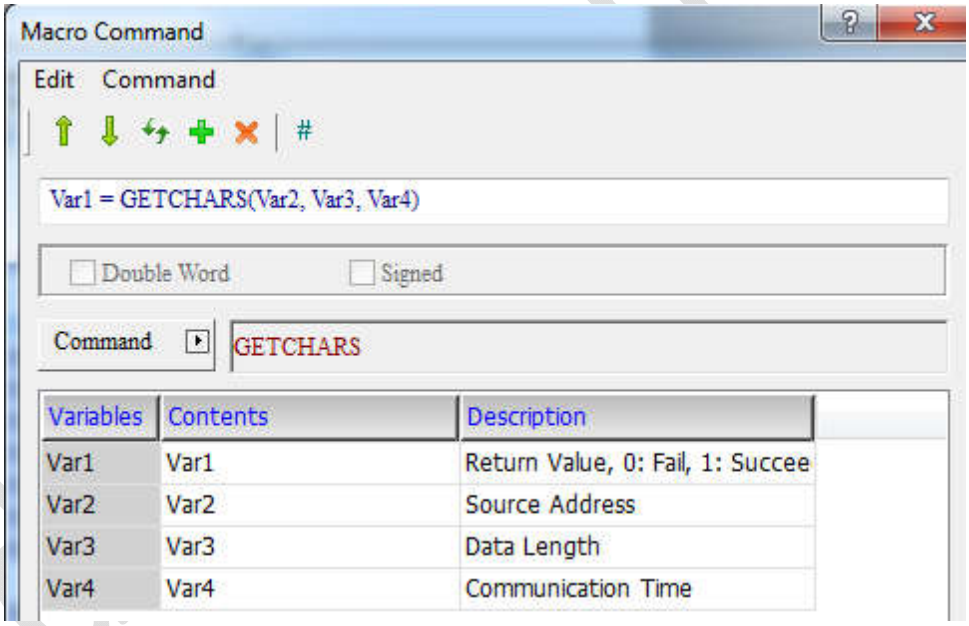
XORSUM : مجموع عملیات XOR بر روی چند رجیستر را ذخیره می کند .



PUTCHARS : دستور ارسال دیتا ، این دستور ، در کنار دستورات INITCOM و SELECTCOM به کار می رود .



GETCHARS : دستور دریافت دیتا ، این دستور ، در کنار دستورات INITCOM و SELECTCOM به کار می رود .



SELECTCOM : دستور برای انتخاب پورت COM ، این دستور در کنار دستور INITCOM مورد استفاده قرار می گیرد .

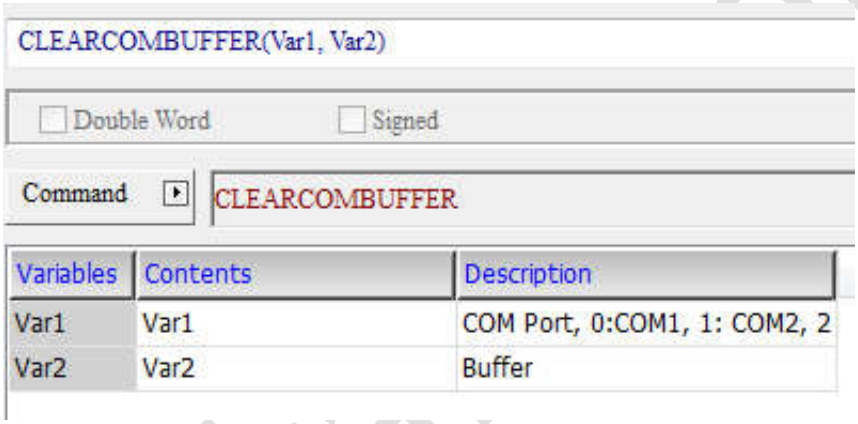

SELECTCOM(Var1)

Double Word Signed

Command SELECTCOM

Variables	Contents	Description
Var1	Var1	COM Port, 0:COM1, 1: COM2, 2

CLEARCOMBUFFER : بافر ارسال یا دریافت دیتا را با مشخص کردن پورت ، پاک سازی می کند ، متغییر های این دستور فقط می توانند اعداد ثابت را شامل شوند .

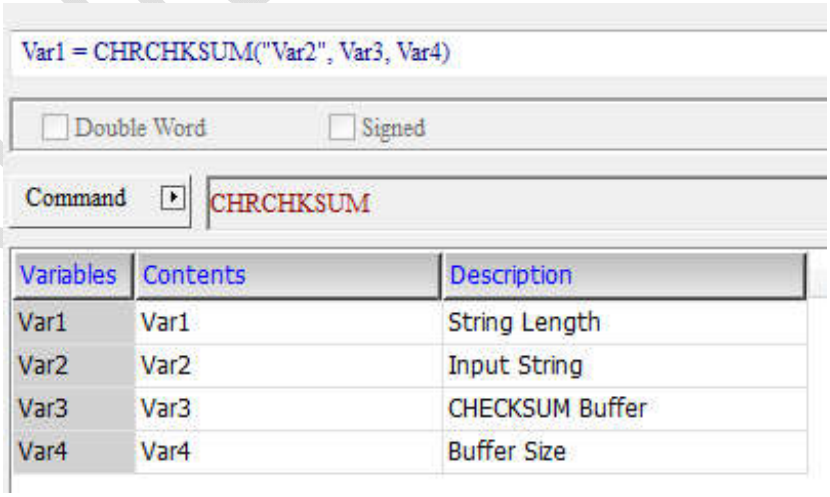

CLEARCOMBUFFER(Var1, Var2)

Double Word Signed

Command CLEARCOMBUFFER

Variables	Contents	Description
Var1	Var1	COM Port, 0:COM1, 1: COM2, 2
Var2	Var2	Buffer

CHECKSUM : این دستور با دریافت یک رشته ، مجموع آن را محاسبه کرده و در بافر CHECKSUM ذخیره می کند .

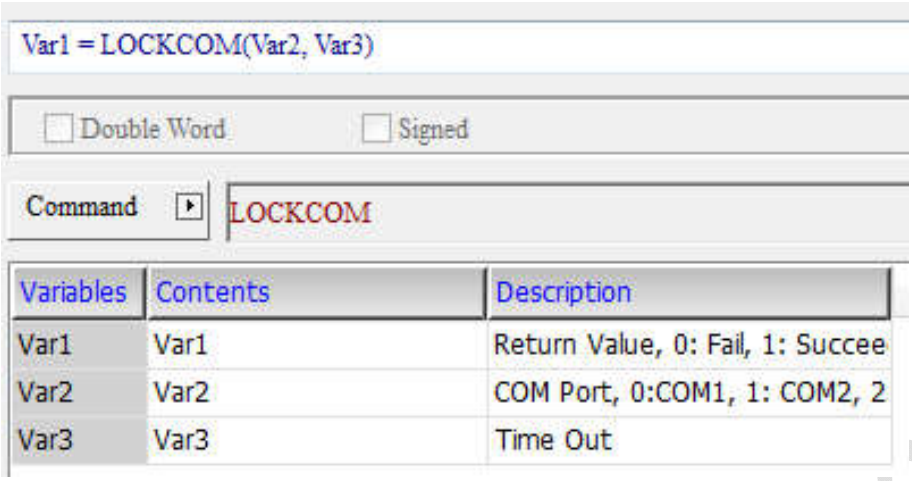

Var1 = CHRCHKSUM("Var2", Var3, Var4)

Double Word Signed

Command CHRCHKSUM

Variables	Contents	Description
Var1	Var1	String Length
Var2	Var2	Input String
Var3	Var3	CHECKSUM Buffer
Var4	Var4	Buffer Size

LOCKCOM / UNLOCKCOM : دستور قفل کردن پورت یا باز کردن قفل پورت .

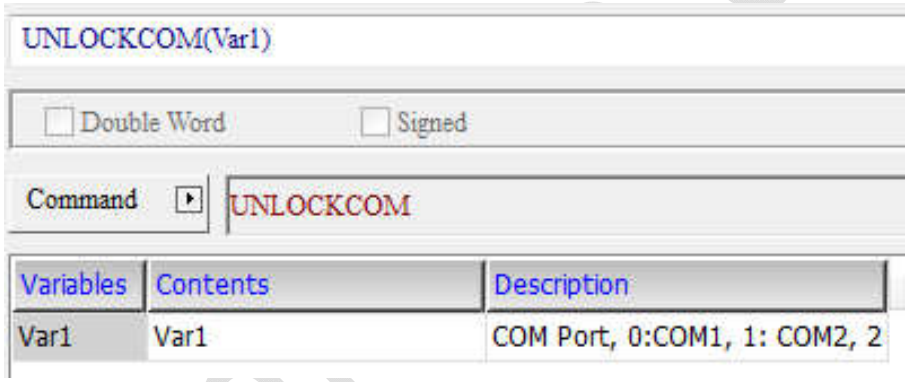

Var1 = LOCKCOM(Var2, Var3)

Double Word Signed

Command LOCKCOM

Variables	Contents	Description
Var1	Var1	Return Value, 0: Fail, 1: Succee
Var2	Var2	COM Port, 0:COM1, 1: COM2, 2
Var3	Var3	Time Out

اگر Var3 را مساوی صفر قرار دهید به این معنی می باشد که دستور LOCKCOM به طور مداوم و بدون در نظر گرفتن زمان TIMEOUT اجرا می شود . یعنی دو دستور LOCKCOM در برنامه ماکرو به طور همزمان اجرا می شوند و به همین علت HMI قادر به پاسخگویی نسبت به دستور ماکرو نخواهد بود .


UNLOCKCOM(Var1)

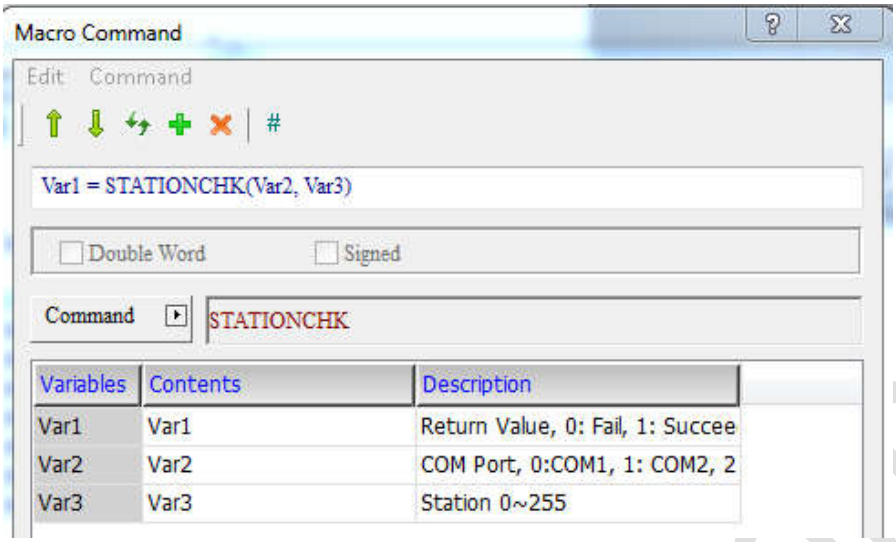
Double Word Signed

Command UNLOCKCOM

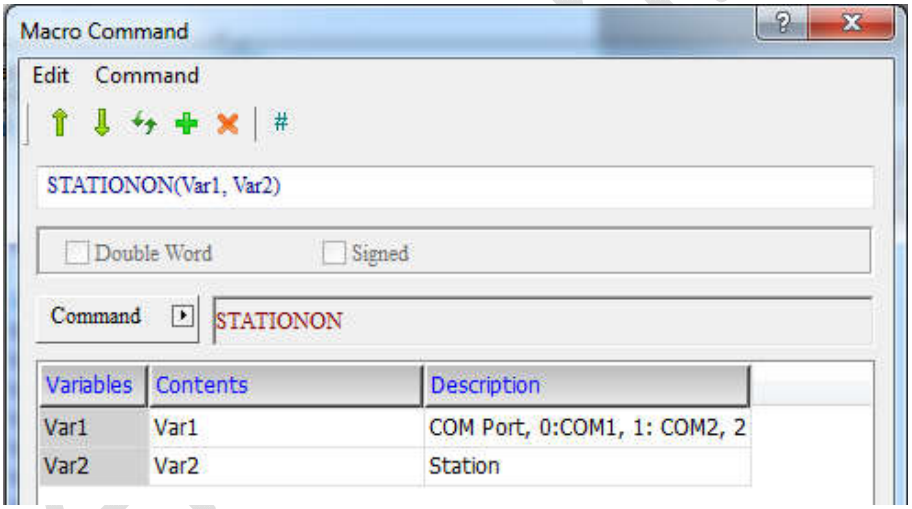
Variables	Contents	Description
Var1	Var1	COM Port, 0:COM1, 1: COM2, 2

زمانی که دستورات COMPORT برای تنظیمات شبکه و ارتباط در چند محیط ماکرو نویسی نوشته شوند ، مثلا OPEN SCREEN MACRO ، ON/OFF MSCRO ، CLOSE SCREEN MACRO ، SCREEN CYCLE MACRO ، خطا رخ خواهد داد . بنابراین بهتر است دستورات LOCKCOM / UNLOCKCOM قبل و بعد از دستورات شبکه نوشته شوند تا در اجرا دستورات برنامه ، وقفه ایجاد نشود .

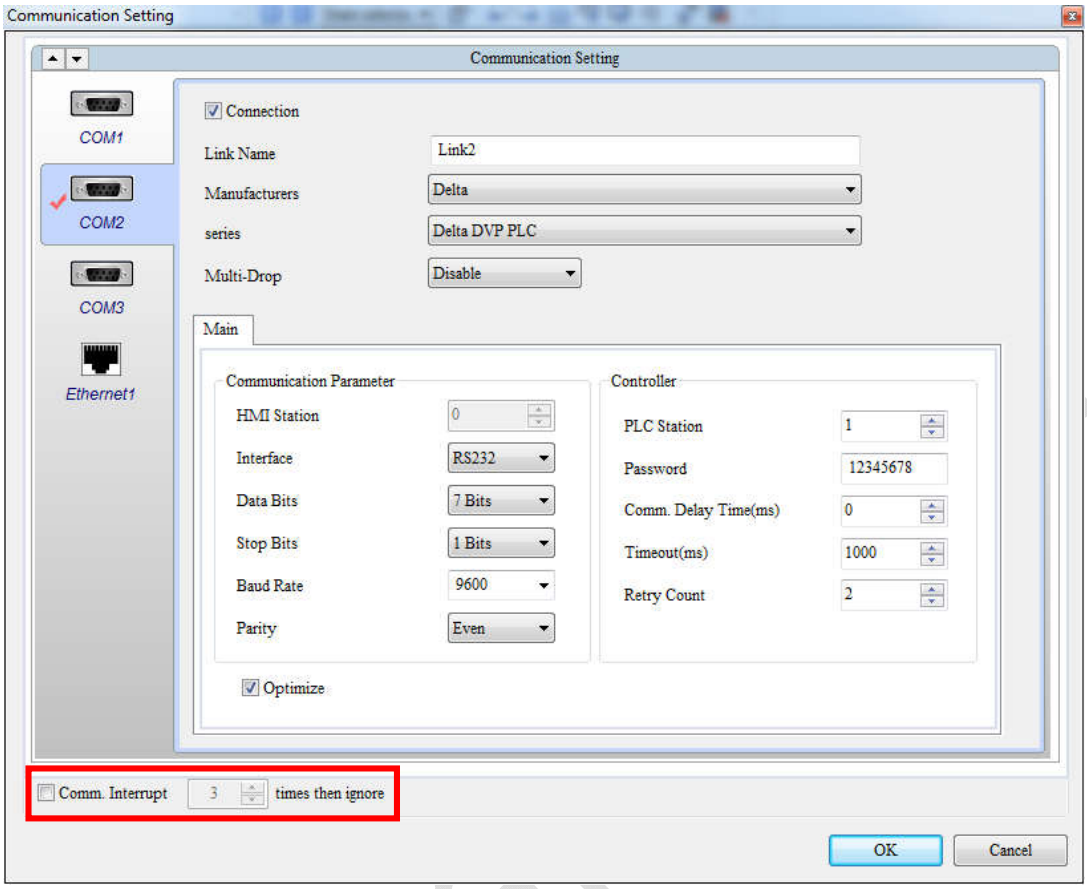
STATIONCHECK : این دستور برقراری یا عدم برقراری ارتباط پورت ها را با توجه به STATION NUMBER دیوایس ها ، بررسی می کند .



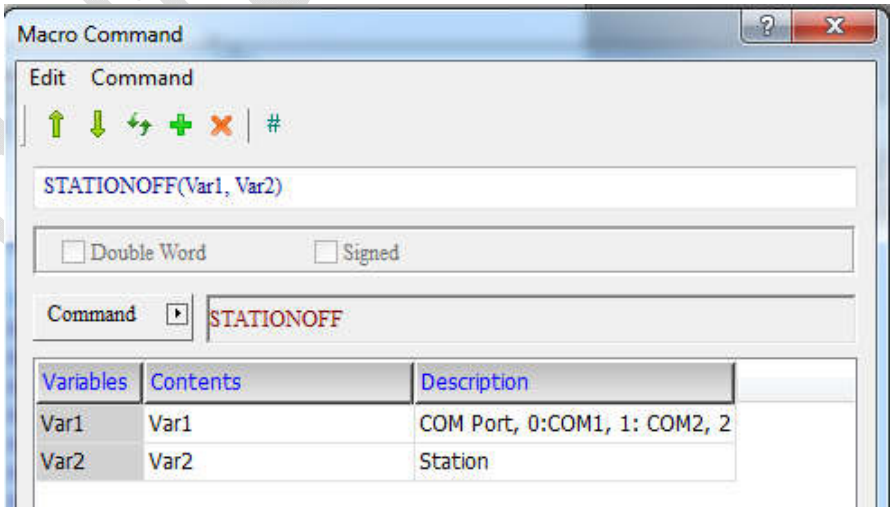
STATIONON : این دستور با دریافت شماره STATION تجهیز و شماره پورت ارتباط ، STATION مورد نظر را فعال می کند .



* در صورتی که گزینه Comm.Interupt ----- Then Ignore فعال باشد این دستور عمل نخواهد کرد ، برای غیر فعال کردن این گزینه وارد منو Option شده و گزینه Communication Setting را انتخاب کنید . گزینه مورد نظر را مانند شکل زیر غیر فعال نمایید .

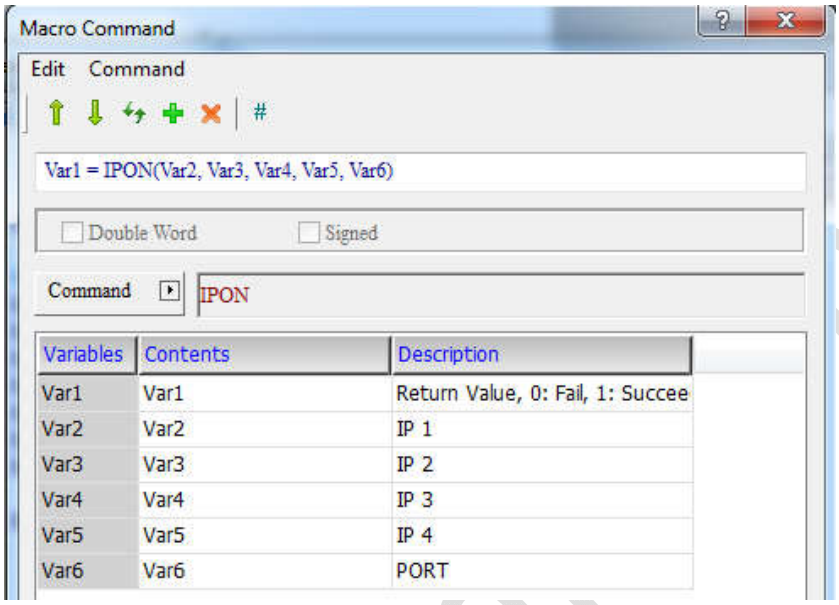


STATIONOFF : این دستور با دریافت شماره STATION تجهیز و شماره پورت ارتباط ، STATION مورد نظر را غیر فعال می کند .

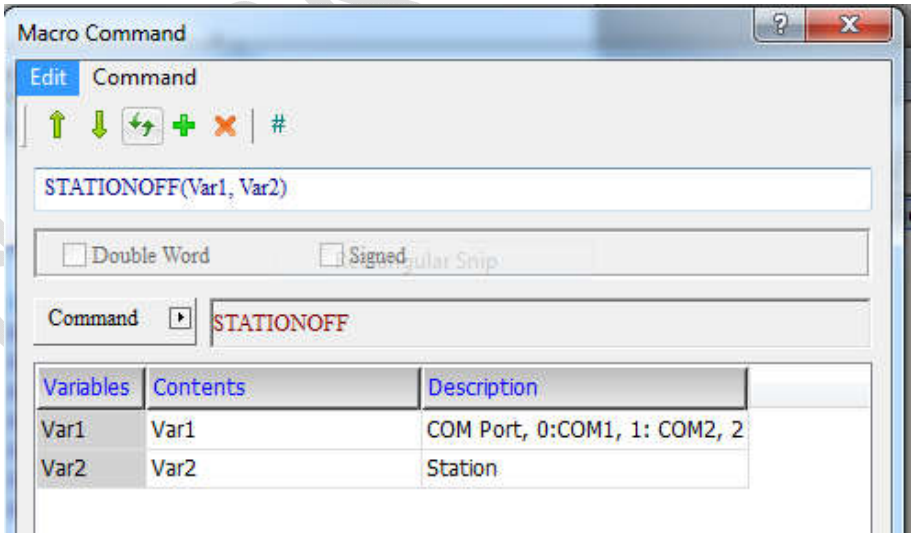


* جهت اجرا شدن دستور ، مانند آنچه در قسمت قبل توضیح داده شده است ، گزینه Comm.Interupt ----- Then Ignore را غیر فعال کنید .

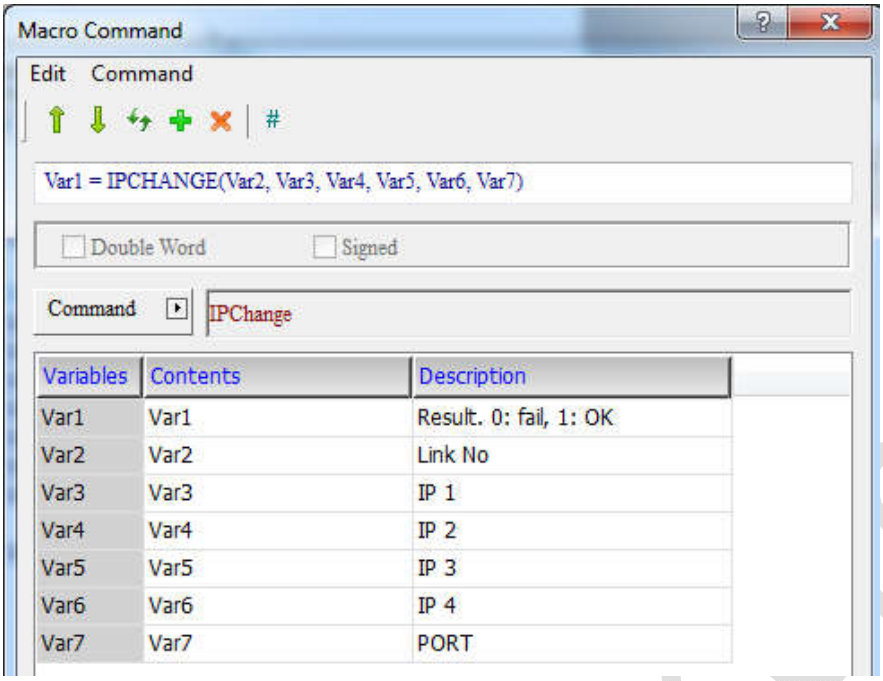
IPON (IP Address Activate) : این دستور IP آدرس را فعال می کند . جهت استفاده ار این دستور باید در تنظیمات Communication Setting ابتدا پورت LAN را فعال نمایید و گزینه Comm.Interupt ----- Then Ignore را غیر فعال نمایید .

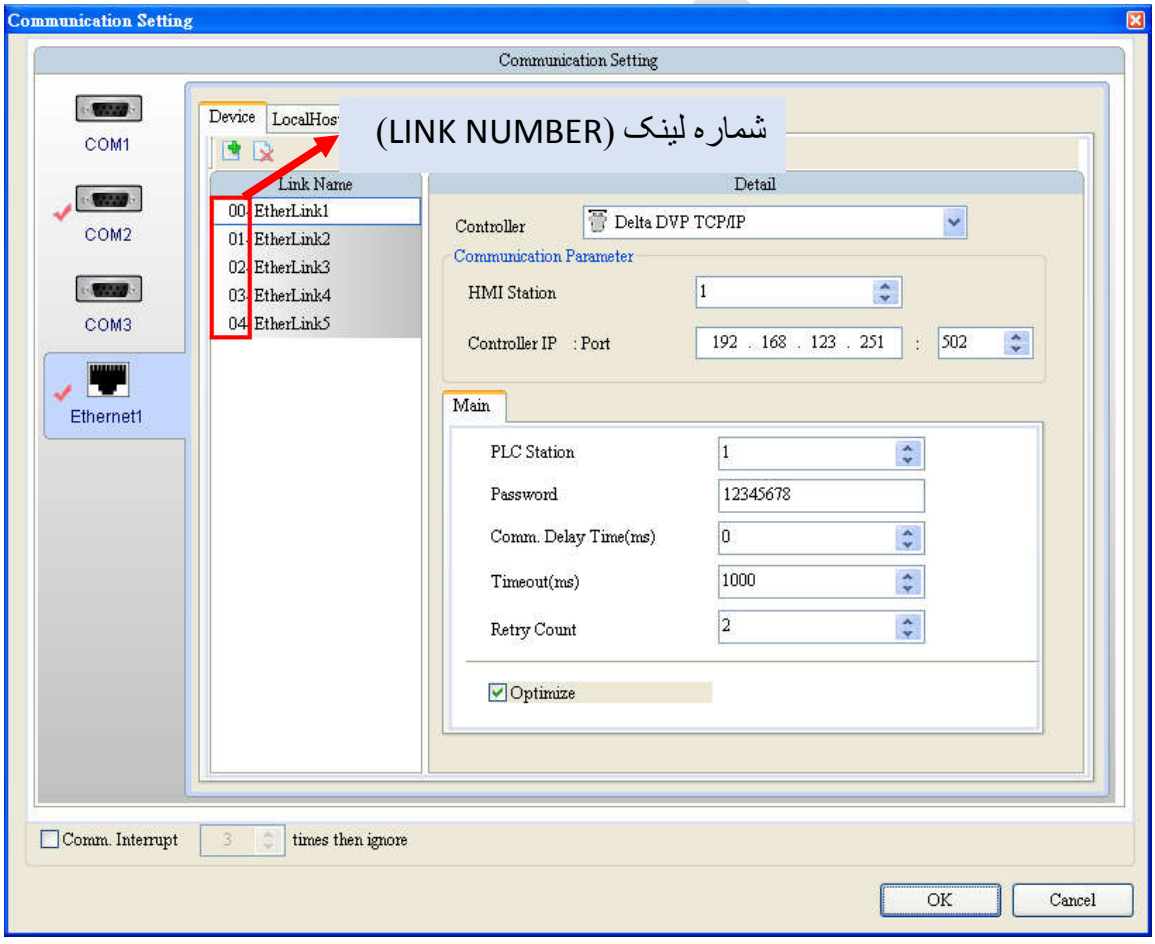


IPOFF (Disable IP address) : این دستور IP آدرس را فعال می کند . جهت استفاده از این دستور مانند مثال قبل باید در تنظیمات Communication Setting ابتدا پورت LAN را فعال نمایید و گزینه Comm.Interupt ----- Then Ignore را غیر فعال نمایید .



IPCHANGE : با استفاده از این دستور می توانید IP را تغییر دهید .

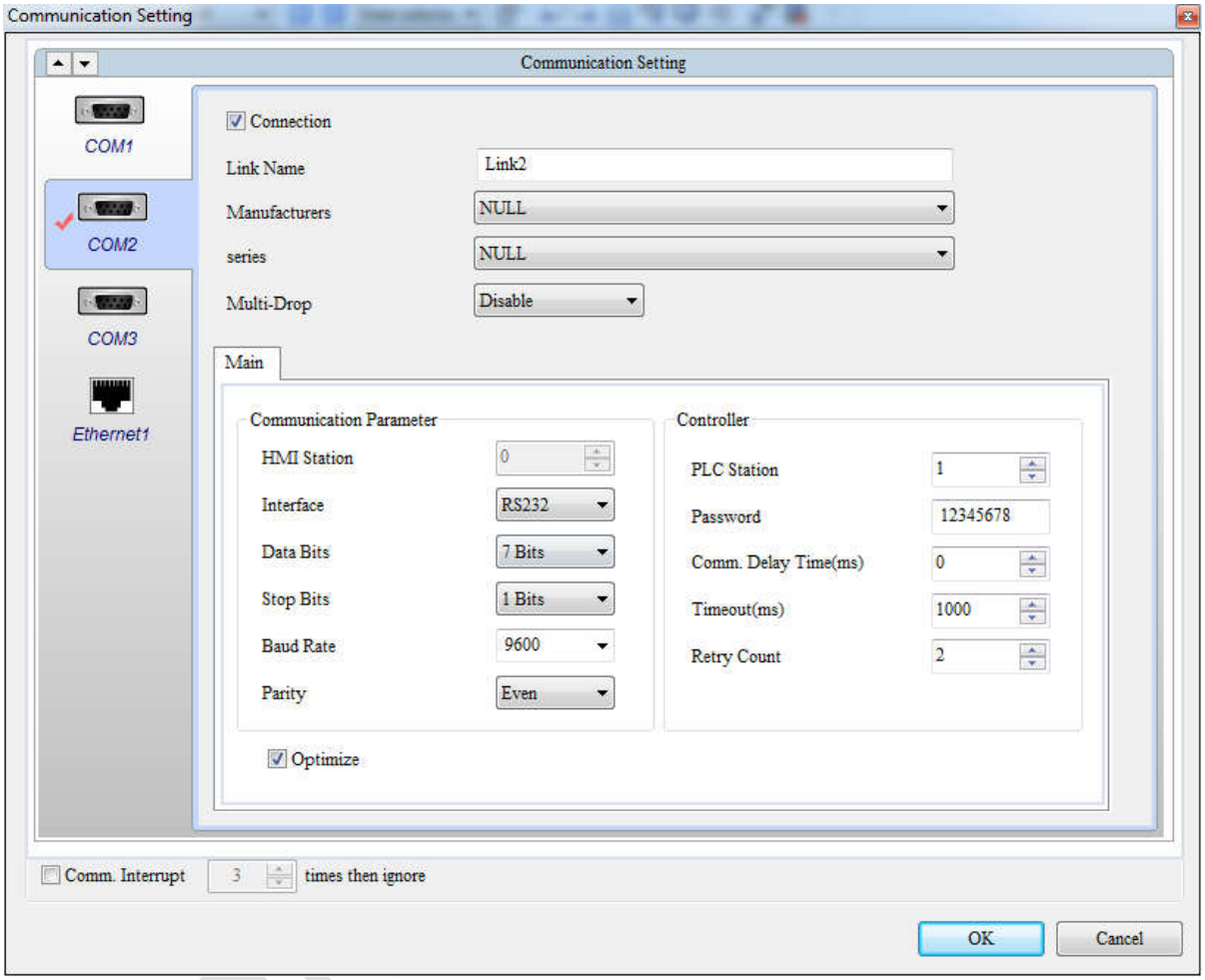




مثال : ارتباط HMI با برد میکرو

برای مثال در صورتی که بخواهید یک تجهیز خارجی مثلا برد میکرو را با استفاده از یک شبکه ارتباطی که پارامترهای آن به صورت User Defined (تعریف شده توسط کاربر) مشخص می شوند ، به HMI متصل کنید باید به صورت زیر عمل نمایید :

1. ابتدا وارد Communication Setting شده و در قسمت Manufactures گزینه NULL را انتخاب نمایید تا شبکه ارتباطی به صورت User Defined تعریف شود ، مانند شکل زیر :

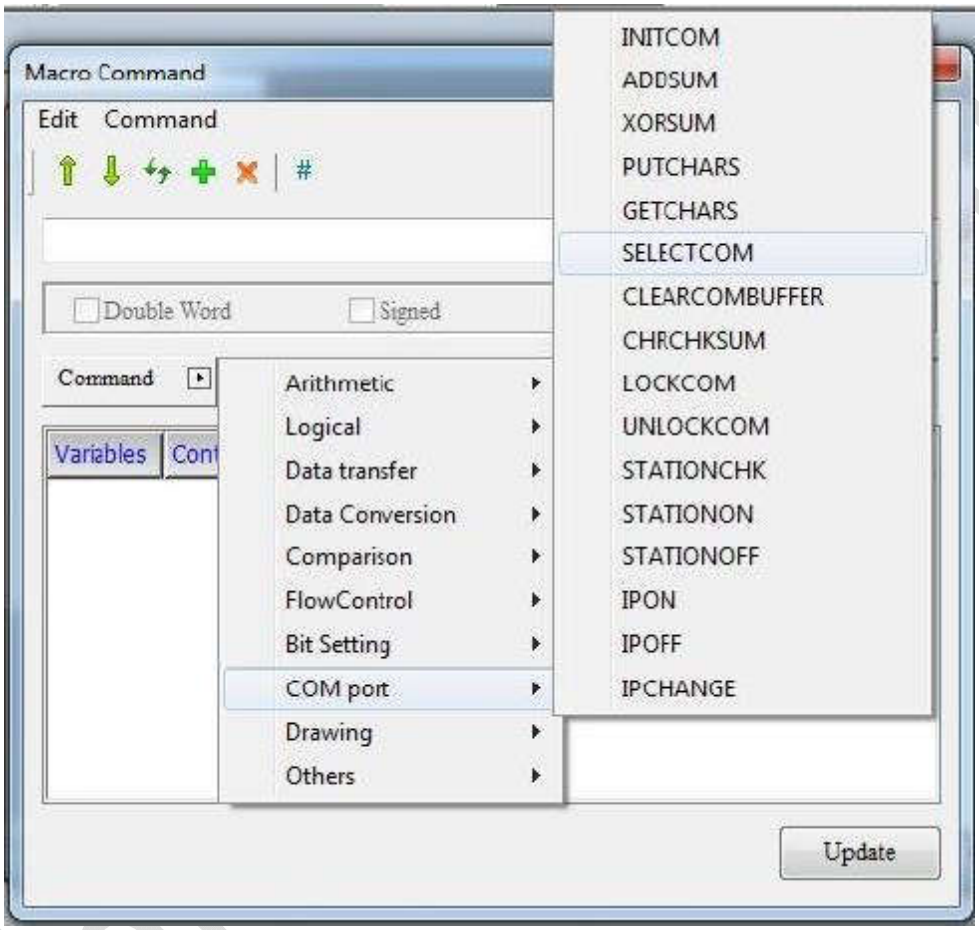


2. سپس وارد یکی از برنامه های ماکرو شوید ، مثلا Background MACRO یا Cycle MACRO .
در صفحه Macro Command وارد قسمت COM port شده و با استفاده از دستورات ابتدا مشخصات شبکه و پورت را وارد نمایید و بعد از آن دستورات ارسال و دریافت دیتا را بنویسید .

INITCOM : دستور تنظیم فرمت شبکه (Baud rate ، شماره پورت ، تعداد بیت های دیتا و ...)

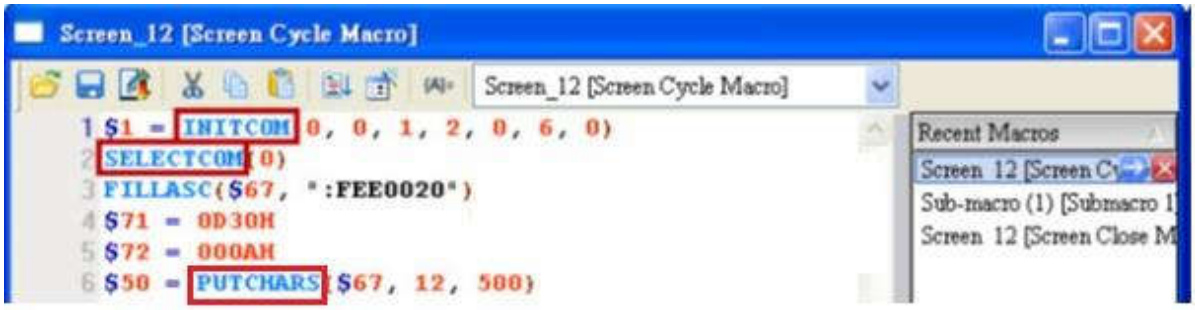
SELECTCOM : دستور تعیین پورت

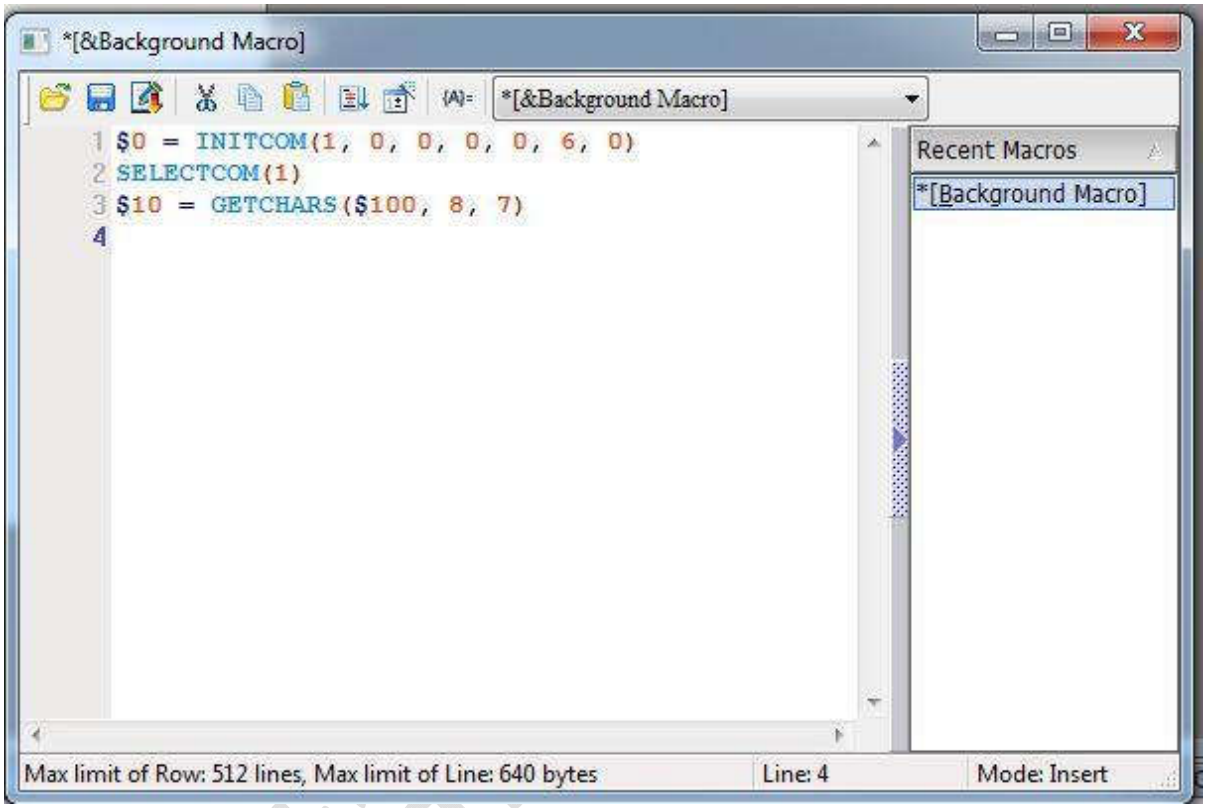
برای دریافت دیتا دستور GETCHARS و برای ارسال دیتا دستور PUTCHARS را استفاده می شود .



* در این شبکه دیتا باید به صورت کد هگز نوشته شود .

در شکل زیر یک نمونه برنامه از ماکرو نویسی شبکه قرار داده شده است :





دستورات Drawing :

دستورات ترسیم به وسیله دریافت مختصات

RECTANGLE	ترسیم مستطیل
LINE	ترسیم خط
POINT	ترسیم نقطه
CIRCLE	

مثال : Circle

ترسیم بیضی :

```
CIRCLE ($1)
```

CIRCLE (Var1)

Var1 = مختصات محور X مرکز بیضی

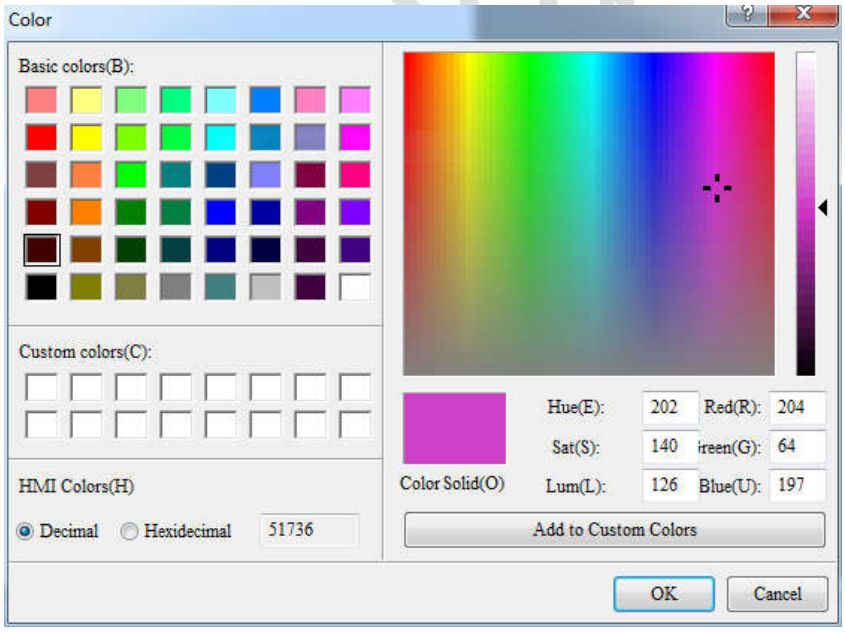
Var1 + 1 = مختصات محور Y مرکز بیضی

Var1 + 2 = طول بیضی (قطر بزرگ)

Var1 + 3 = عرض بیضی (قطر کوچک)

Var1 + 4 = رنگ بیضی

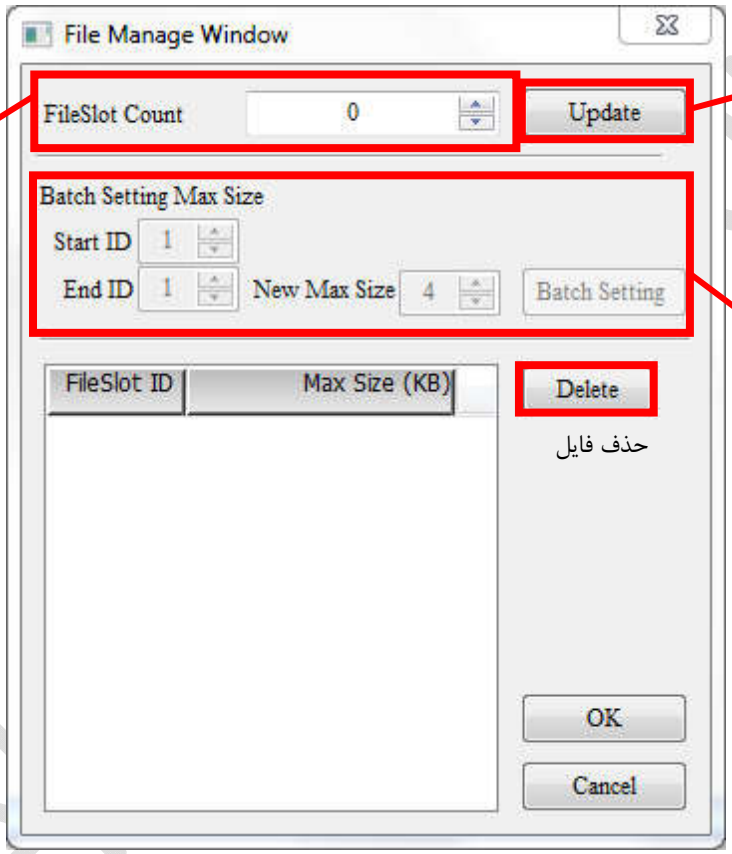
Var1 فقط می تواند یک رجیستر از حافظه داخلی HMI باشد .کد مربوط به رنگ ها را می توانید از جدول Color بدست آورید .



: FileSlot

جهت ذخیره سازی دیتا با طول و محتوا مشخص از FileSlot ها استفاده می شود . برای مثال ، در یک کارخانه محصولات متنوعی تولید می شود و هر محصول دارای مشخصات مخصوص به خود می باشد . برای آنکه بتوان به راحتی خط تولید هر محصول را برنامه ریزی کرد ، می توان از FileSlot ها استفاده کرده و مشخصات هر محصول را در رجیسترهای اختصاص داده شده به فایل ذخیره کرده و در مواقع مورد نیاز از دیتا استفاده شود .

فایل اسلات ها در حافظه Flash ROM ، HMI ذخیره می شوند . برای استفاده از Fileslot ، ابتدا باید فایل مورد نظر را ایجاد کنید . بدین منظور وارد منو Option شده و گزینه FileSlot File Managment را انتخاب کنید ، پنجره ای مانند شکل زیر گشوده خواهد شد :



تعداد کلی FileSlot ها

ماکزیمم تعداد FileSlot، 1024 عدد می باشد .

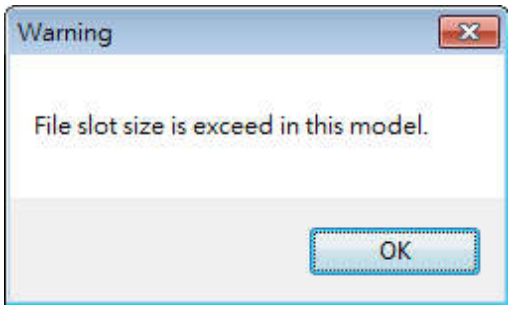
پس ار انتخاب تعداد فایل ها با استفاده از گزینه Update ، فایل ها ایجاد می شوند .

به صورت پیش فرض حجم هر فایل 4KB در نظر گرفته شده است ، اما با تعیین StartID و EndID می توانید حجم فایل اسلات هایی که رنج ID آنها در بازه Start تا End قرار می گیرد را تغییر دهید . حجم مورد نظر را در قسمت New Max Size نوشته با کلیک بر روی گزینه Batch Setting آن را استخراج نمایید .

حذف فایل

فایل اسلات می تواند ماکزیمم تا 102400 KB را اشغال نماید و حداقل ظرفیت هر فایل اسلات 4 KB می باشد . پس از اعمال تنظیمات مورد نظر بر روی گزینه OK کلیک کنید تا فایل ها ایجاد شود .

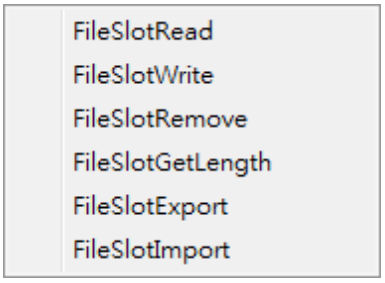
اگر حجم فایلی که ایجاد کرده اید با ظرفیت حافظه Flash ROM ، HMIمغایرت داشته باشد با پیام نشان داده شده در شکل زیر مواجه خواهید شد و فایل ها ذخیره نخواهند شد .



* برای خواندن دیتا ذخیره شده در فایل ها از دستور ماکرو FileSlotRead استفاده می شود و اگر بخواهید از فایل های ذخیره شده بر روی فلش مموری یا SDcsrd استفاده کنید باید دیتای آن ها را با دستور FileSlotImport استخراج نمایید .
* برای نوشتن دیتا در فایل ها از دستور ماکرو FileSlotWrite و برای ذخیره سازی آن ها بر روی فلش مموری یا SDcard از دستور ماکرو FileSlotExtract استفاده می شود .
* با ریست کردن یا Format کردن HMI و یا با استفاده از دستور ماکرو FileSlotRemove می توانید فایل ها را پاک کنید .
* با کپی کردن فایل ها از فلش بر روی HMI و یا بر عکس دیتا فایل ها تغییر نخواهد کرد .

دستورات File Access :

این دستورات شامل دستوراتی چون خواندن ، نوشتن ، ذخیره سازی فایل ها و ... می باشد .



خواندن مقادیر فایل اسلات ها

نوشتن در فایل اسلات ها

پاک کردن فایل اسلات

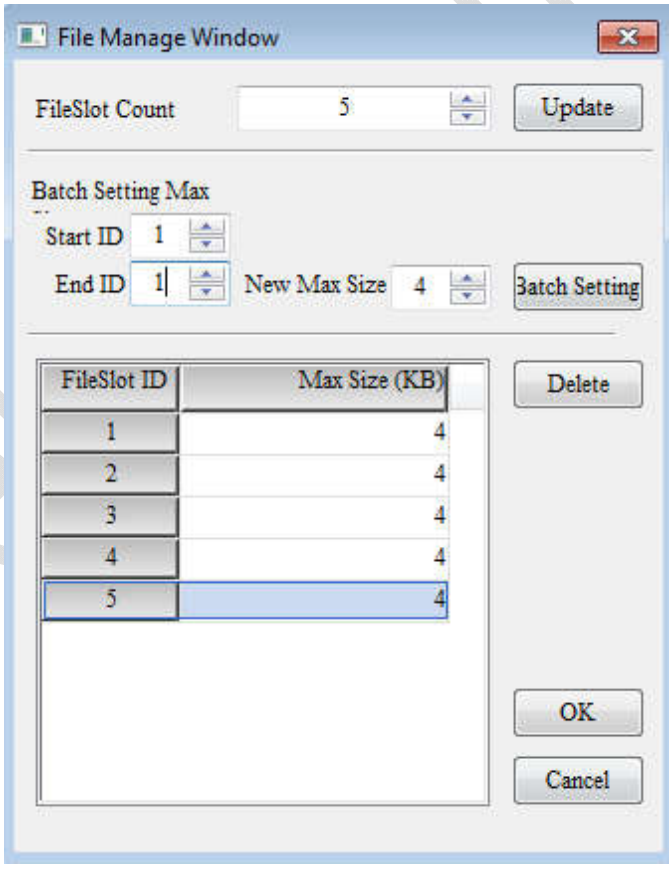
دریافت طول فایل اسلات

ذخیره سازی فایل اسلات بر روی فاش یا SDcard

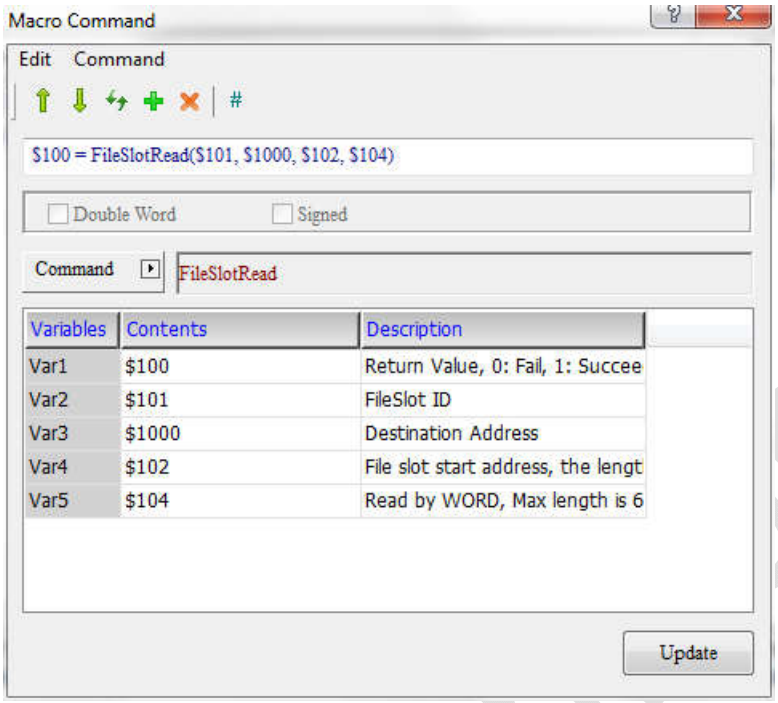
وارد کردن فایل اسلات در برنامه از فلش یا SDcard

مثال : FileSlot Read/Write

وارد پنجره FileSlot File Managment شده و با اعمال تنظیمات نشان داده شده در شکل زیر فایل اسلات را بسازید :



پنجره Background MACRO باز کنید و از منوی File Access دستور FileSlotRead را انتخاب کرده و مانند شکل زیر آدرس دهی نمایید.



Var1 : این متغییر دو مقدار 0 یا 1 را می پذیرد و نشان دهنده موفقیت در انجام عملیات (1) و یا عدم موفقیت (0) می باشد .

Var2 : ID مربوط به FileSlot مورد نظر برای خواندن یا نوشتن دیتا را در این قسمت وارد نمایید.

Var3 : آدرس شروع رجیسترهایی که قرار است دیتای فایل ها در آن ها نوشته شوند را در این متغییر بنویسید .

Var4 : آدرس شروع فایل اسلات هایی که دیتا آنها خوانده و در رجیسترهای مقصد نوشته می شوند را در این متغییر وارد کنید.

Var4 : طول دیتاهایی که قرار است خوانده یا نوشته شوند را در این قسمت مشخص کنید.

در پروژه ای که ایجاد کرده اید دو کلید Maintained ایجاد کنید و آدرس های $10.0 و $10.1 را به آن ها اختصاص دهید سپس وارد Properties کلید ها شوید در بخش Details و در قسمت On MACRO برنامه های زیر را بنویسید تا امکان خواندن و نوشتن FileSlot ها را داشته باشید.

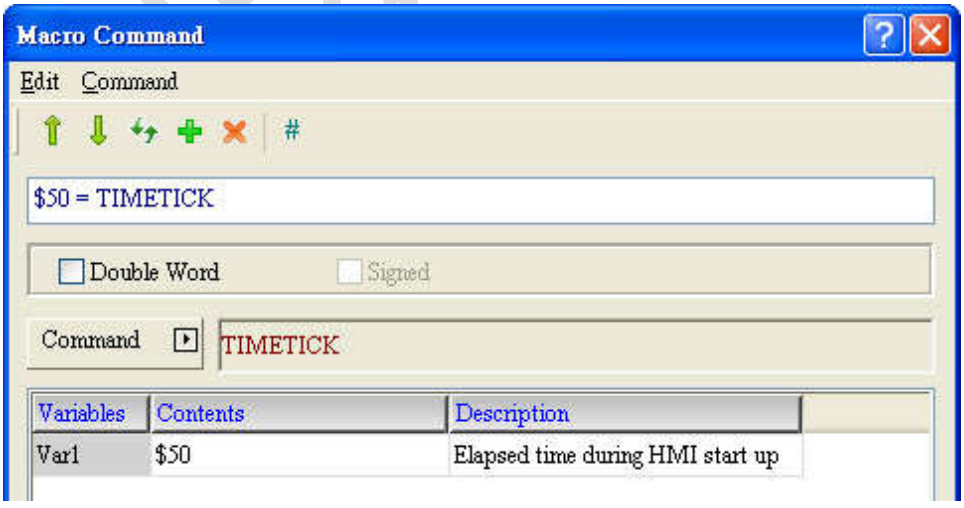
با وارد کردن شماره ID مربوط به FILESLOT و مشخص نمودن طول دیتا و فشردن کلید READ/WRITE می توانید دیتا را خوانده یا بنویسید .

دستورات Others :

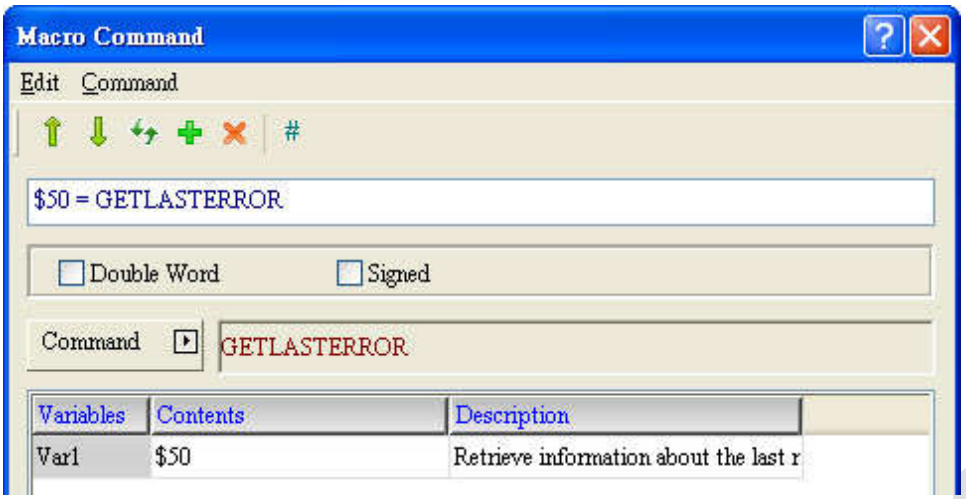
این دسته از دستورات شامل دستورات متنوعی مثل دستور Delay ، دستور استخراج Recipe ها ، تنظیم زمان و تاریخ داخلی HMI و ... می باشد.

1	Time Tick
2	GETLASTERROR
3	Comment
4	Delay
5	GETSYSTEMTIME
6	SETSYSTEMTIME
7	GETHISTORY
8	EXPORT
9	EXRCP16
	IMRCP16
	EXRCP32
	IMRCP32
	EXENRCP
	IMENRCP
10	EXHISTORY
11	EXALARM
12	DISKFORMAT
13	BMPCAPTURE
14	PLCDOWNLOAD
15	GetCircleCenter

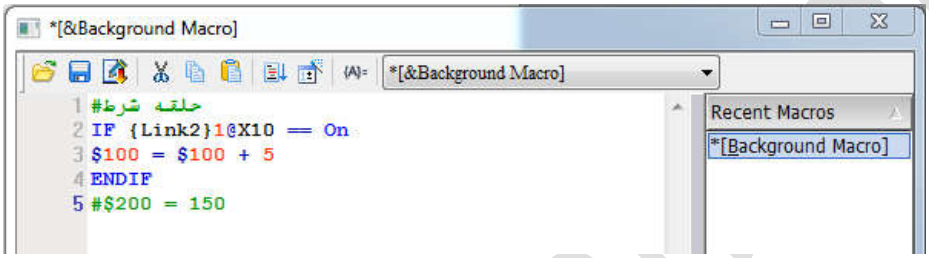
1. محاسبه مدت زمان Start Up و ذخیره در Var1



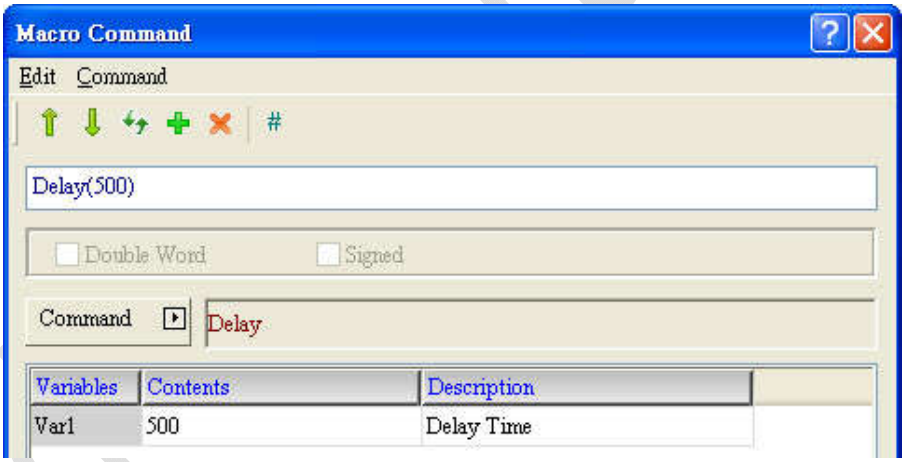
2. کد آخرین خطای سیستم را در Var1 نشان می دهد . (W/DW – Signed/UnSigned)



3. این دستور در ابتدای خط برنامه با قرار دادن # ، آن را به کامنت تبدیل می کند .

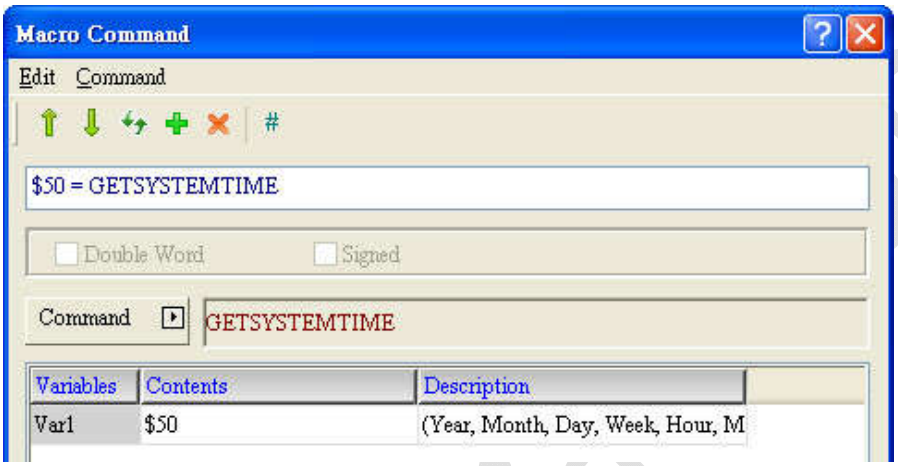


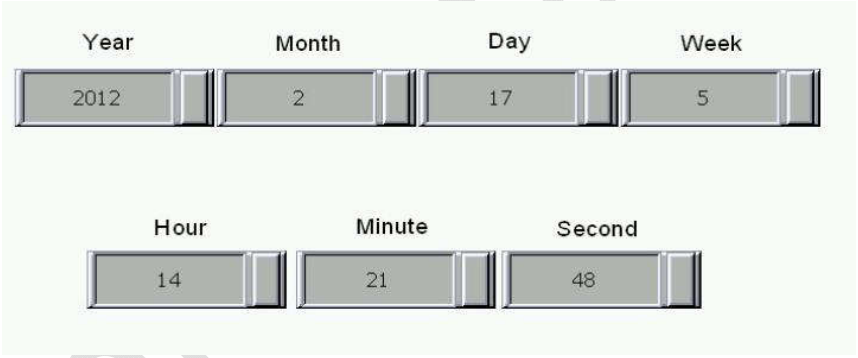
4. دستور وقفه ، زمان 0 تا 65767 میلی ثانیه را در Var1 قرار دهید .



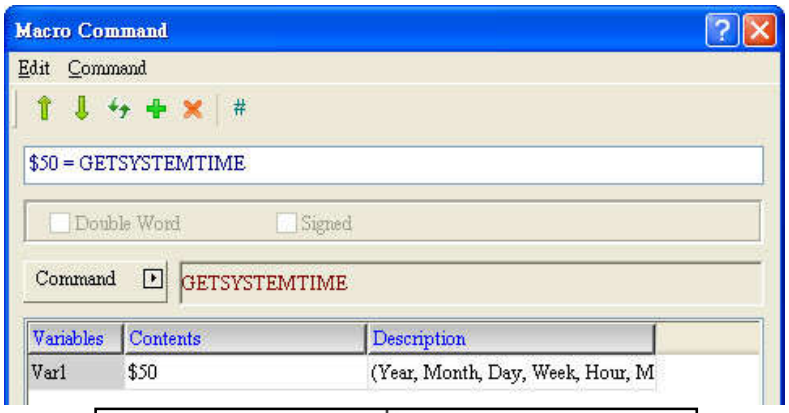
5. این دستور 7 رجیستر را اشغال کرده و زمان و تاریخ HMI را نشان می دهد . آدرس اولین رجیستر را در Var1 وارد کنید .

Var 1	Year
Var 1 + 1	Month
Var 1 + 2	Day
Var 1 + 3	Week
Var 1 + 4	Hour
Var 1 + 5	Minute
Var 1 + 6	Second

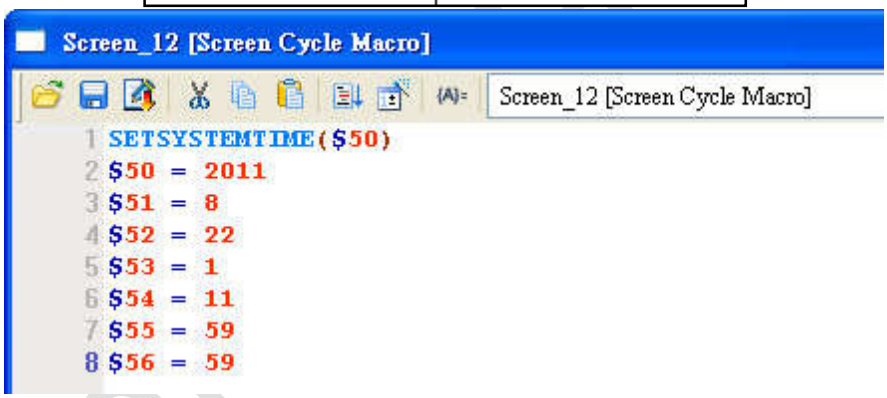


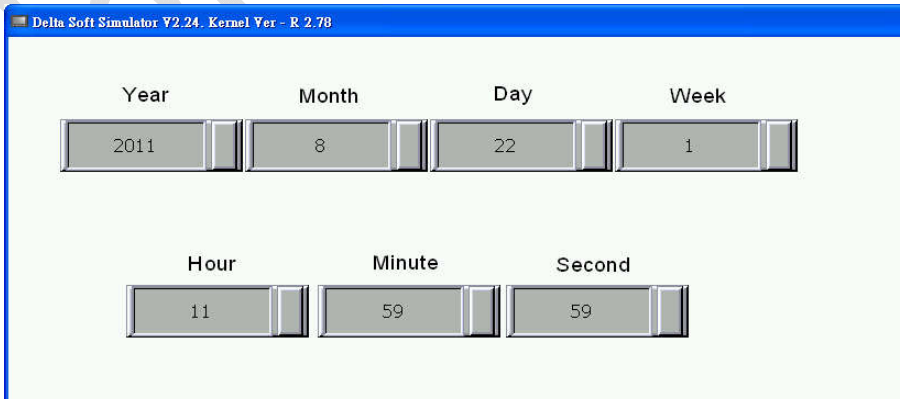


6. دستور تنظیم ساعت و تاریخ HMI . زمان و تاریخ مورد نظر را در 7 رجیستر با آدرس های پشت سر هم و با ترتیب نشان داده شده در شکل زیر ذخیره کنید و آدرس اولین رجیستر را در Var1 قرار دهید .

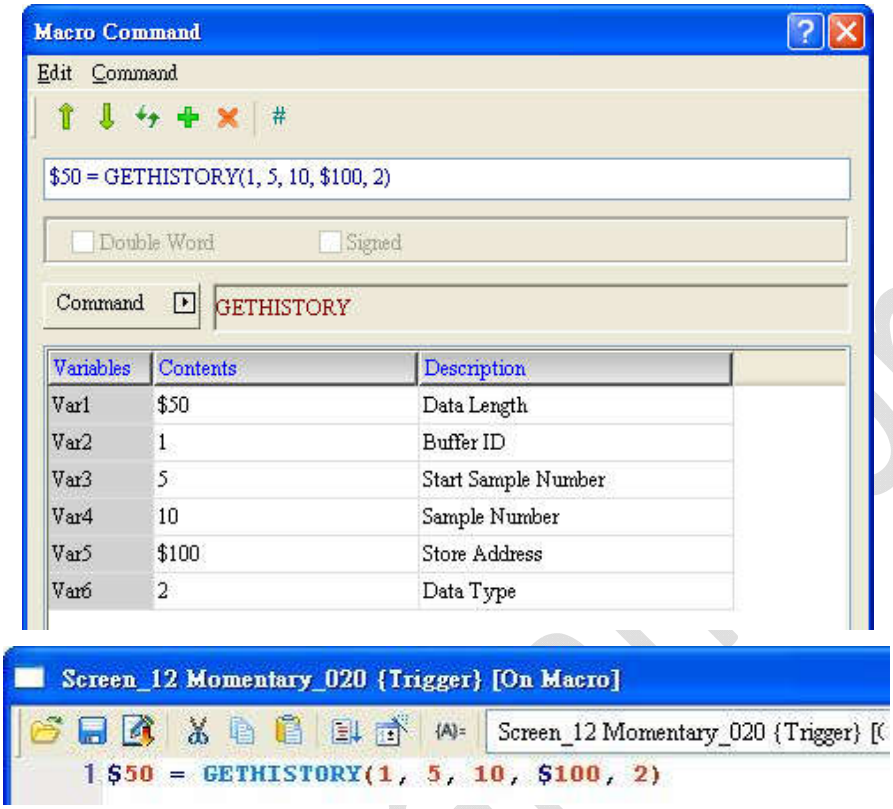


Var 1	Year
Var 1 + 1	Month
Var 1 + 2	Day
Var 1 + 3	Week
Var 1 + 4	Hour
Var 1 + 5	Minute
Var 1 + 6	Second

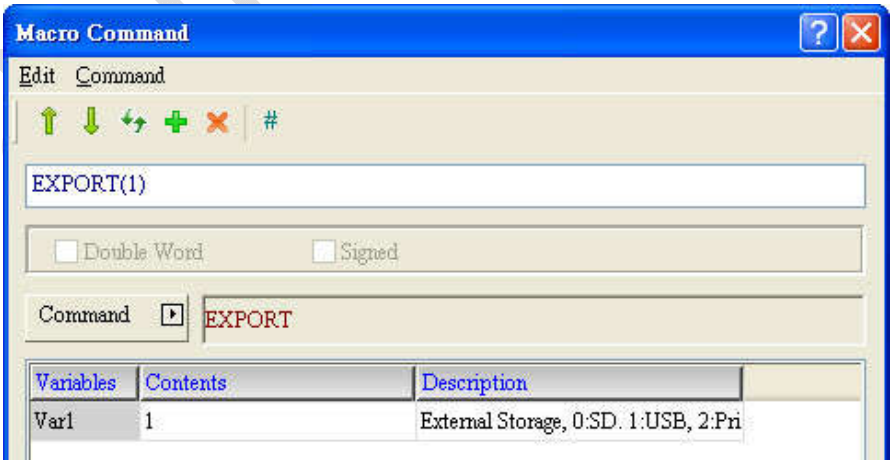




7. دستور ایجاد بافر جهت Data Logging . آدرس بافر در Var5 ، طول دیتا در Var1 ، شماره ID در Var2 ، تعداد نمونه گیری Var3 (Sample Number) ، شماره Sample برای شروع ذخیره سازی دیتا و نوع ذخیره سازی دیتا را در Var6 مشخص کنید . اگر مقدار 0 را در Var6 قرار دهید . فقط دیتا ذخیره می شود برای ذخیره سازی همزمان دیتا و زمان نمونه گیری مقدار 2 را در Var6 وارد نمایید .



8. این دستور به فلش مموری ، SDcard یا پرینتر یک گزارش تحویل می دهد . اگر عدد صفر را وارد نمایید ذخیره سازی در SDcard انجام می شود . مقدار یک برای فلش مموری و مقدار 2 برای پرینتر استفاده می شود .

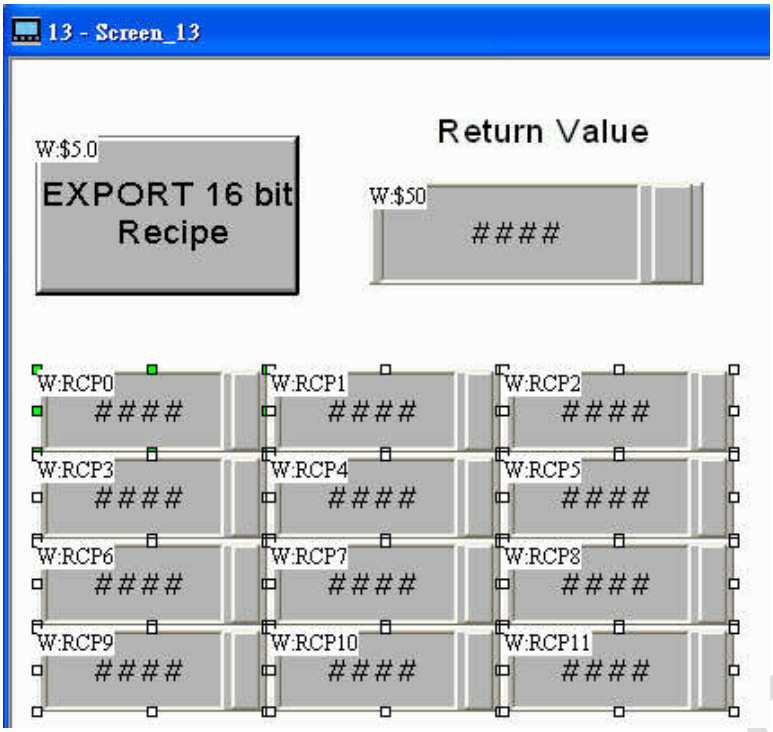


دستورات ذخیره سازی یا استخراج Recipe ها . در Var2 نام فایل را قرار دهید و محل ذخیره سازی دیتا و یا استخراج فایل رسیپی را در Var3 تعیین نمایید . برای اطمینان از صحت عملکرد (مقدار یک) یا عدم عملکرد (مقدار یک) از Var1 استفاده می شود . دستورات EXRCP16/32 برای ذخیره کردن جداول و دستورات IMRCP16/32 برای وارد کردن جداول از فلش یاSDcard استفاده می شود . اگر از SDcard استفاده می کنید مقدار 3 را در var3 بریزید و اگر از فلش مموری استفاده می کنید مقدار 2 را به Var3 انتقال دهید . دستورات IMENRCP و EXENRCP نیز مانند دستورات 16 بیتی و 32 بیتی عمل می کنند و برای Recipe های Enhanced مورد استفاده قرار می گیرد .

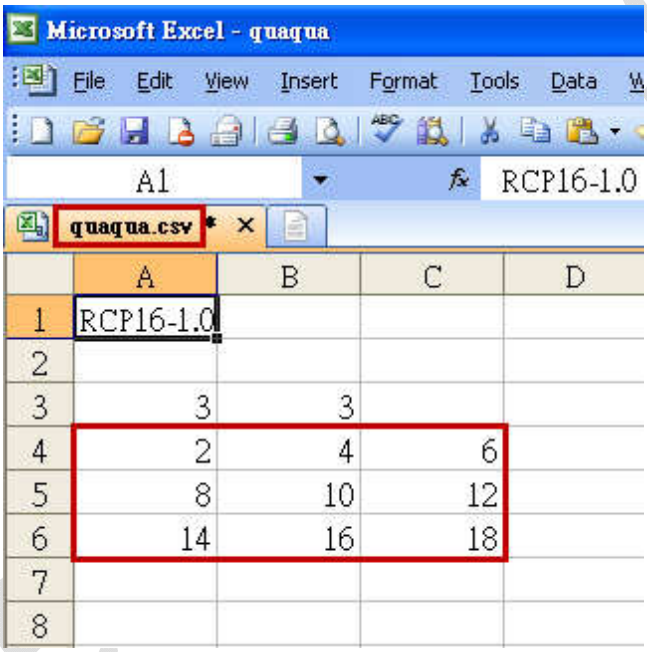
مثال : دستور EXRCP16

1. برای ذخیره سازی فایل Recipe ها بر روی فلش نیاز به نوشتن برنامه نشان داده شده در شکل زیر را دارید . با دستور FILLASC می توانید اسم فایل را به صورت کد اسکی بنویسید و در Var2 ذخیره کنید .

2. در صفحه HMI مانند شکل زیر از دستور Numeric Entry برای وارد کردن دیتا در Recipe ها استفاده کنید و یک کلید با آدرس $5.0 برای فرمان ذخیره سازی دیتا استفاده کنید .

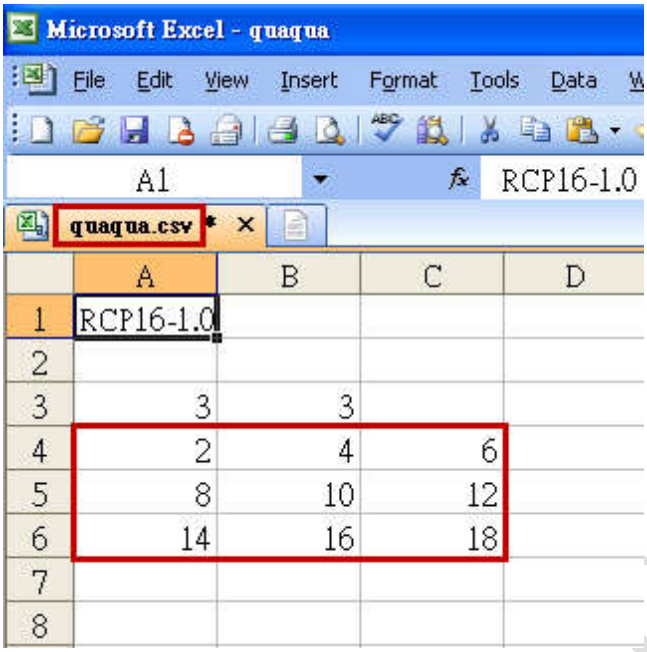


3. فایلی که بر روی فلش مموری ذخیره می شود ، به صورت فایل اکسل می باشد (مانند شکل زیر).



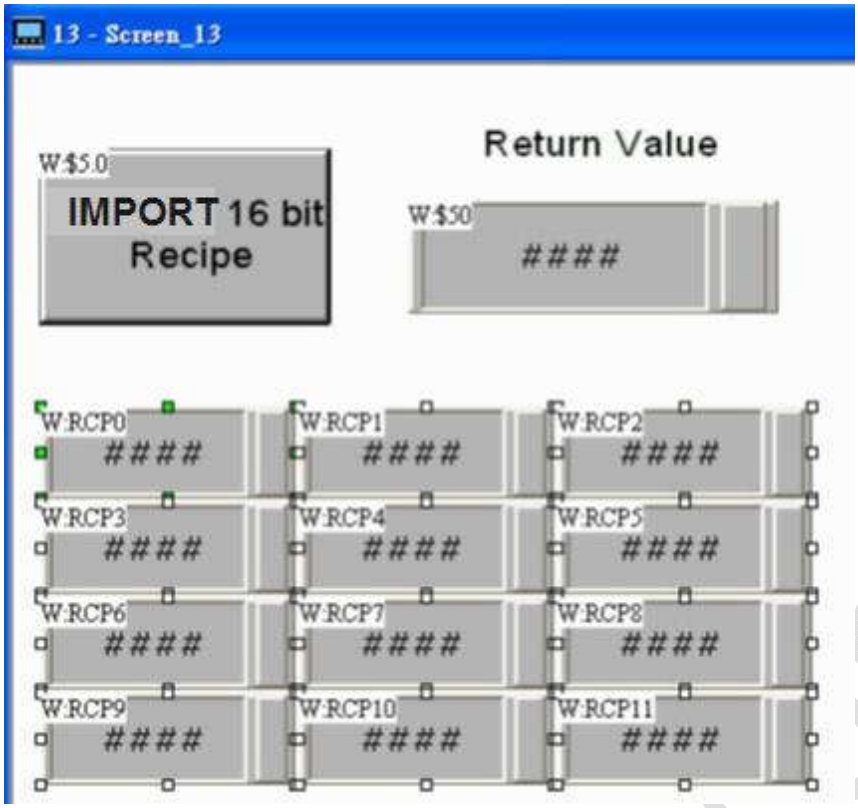
مثال : IMRCP16

1. ابتدا باید یک فایل اکسل برای وارد کردن جدول Recipe از روی فلش مموری یا SDcard تهیه کنید .



2. سپس مانند شکل زیر ، برنامه ماکرو برای وارد کردن جدول Recipe به برنامه HMI را بنویسید .

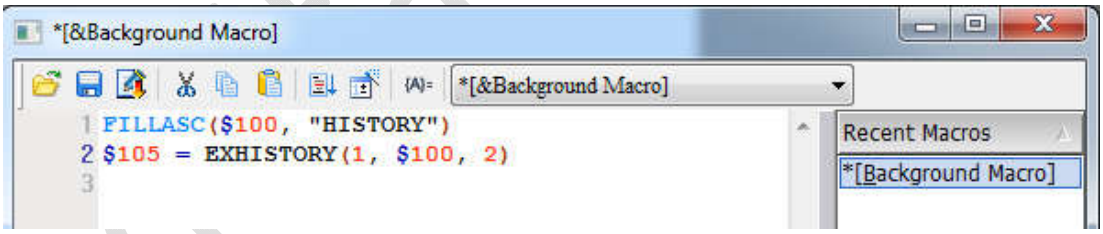
3. سپس برای مشاهده مقادیر Recipe ها از دستور Numeric Entry متناسب با تعداد سطر ها و ستون ها استفاده نمایید . برای اعمال دستور وارد کردن اطلاعات ذخیره شده بر روی فلش مموری یا SDcard ، از یک کلید با آدرس $50.0 (مطابق برنامه ماکرو) استفاده کنید .



9. دستور ذخیره کردن دیتا Historical بر روی فلش مموری یا SDcard . شماره ID بافر در Var2 و اسم فایل در Var3 و محل ذخیره سازی دیتا در Var4 قرار می گیرد . Var1 موفقیت یا عدم موفقیت در ذخیره سازی دیتا را نمایش می دهد . برای استفاده از فلش مموری عدد 2 را به Var4 انتقال دهید و برای SDcard عدد 3 را در Var4 بریزید .

مثال : EXHISTORY

فایلی با نام HISTORY از بافر با ID شماره یک ، بر روی فلش ذخیره خواهد شد .

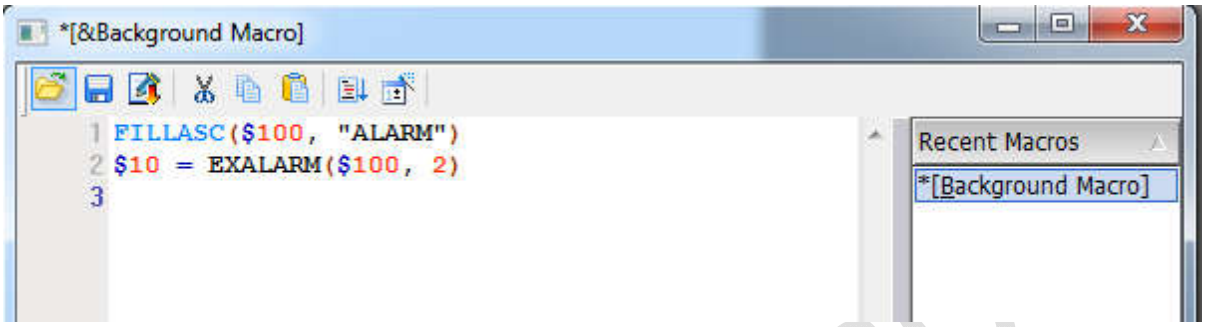


از دستور FILLASC برای وارد کردن نام فایل استفاده شده است .

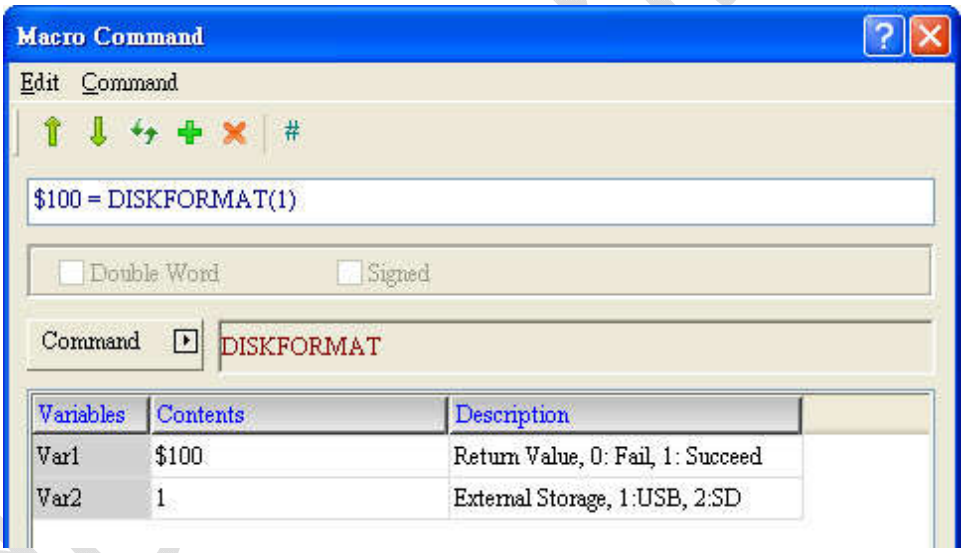
10.دستور ذخیره سازی دیتا آلارم ها بر روی فلش مموری یا SDcard . مانند دستورات قبل اسم فایل در Var2 و محل ذخیره سازی در Var3 مشخص می شود . نتیجه عملکرد دستور را می توانید در Var1 مشاهده نمایید .

مثال : EXALARM

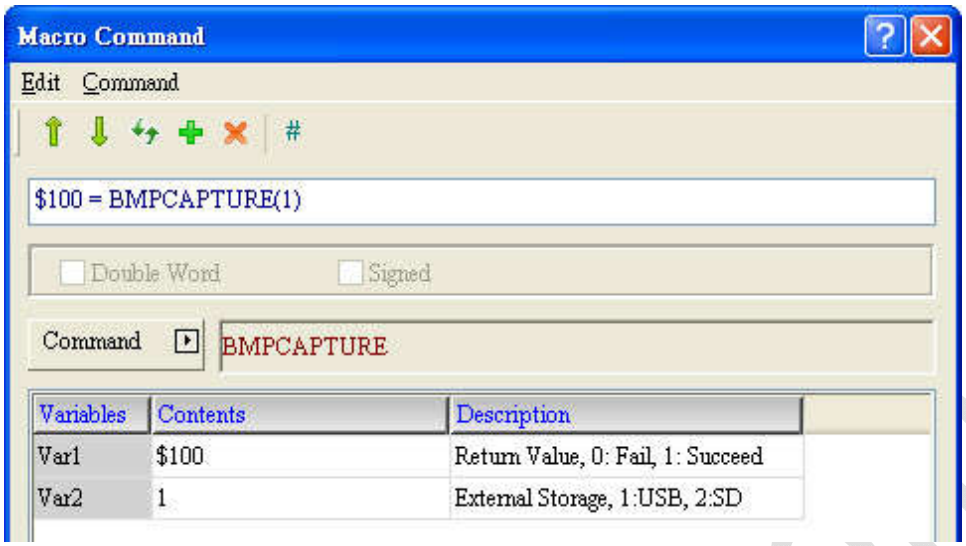
در این برنامه فایلی با نام ALARM بر روی فلش ذخیره خواهد شد . از دستور FILLASC برای وارد کردن نام فایل استفاده شده است .



11.با استفاده از این دستور می توانید فلش مموری یا SDcard را فرمت کنید . برای فرمت کردن SDcard عدد 1 و برای فرمت کردن فلش مموری عدد 2 را در Var2 قرار دهید .



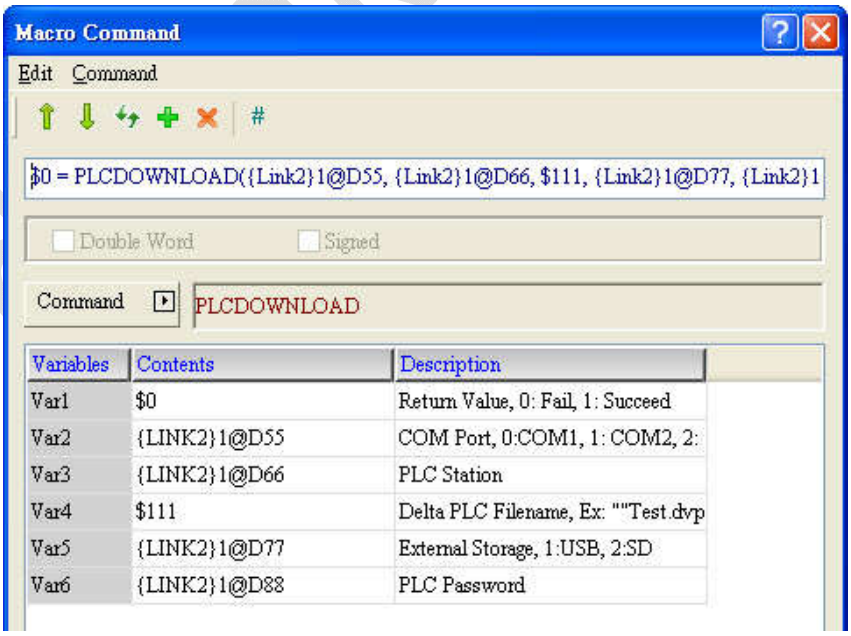
12.این دستور از صفحه HMI عکس گرفته و بر روی فلش یا SDcard ذخیره می کند . محل ذخیره سازی عکس را در Var2 و با مقدار 1 (SDcard) یا مقدار 2 (فلش مموری) مشخص نمایید .



13.با استفاده از این دستور می توانید از طریق HMI ، PLC را برنامه ریزی کنید . پورت ارتباط HMI و PLC را در Var2 ، Station Number پی ال سی را در Var3 مشخص کنید . همچنین نام فایل برنامه در Var4 ، محل قرار گرفتن فایل در Var5 و پسورد PLC (در صورتی که بر روی PLC پسورد گذاشته شده باشد) در Var6 مشخص می شود . برای مشخص کردن نام فایل برنامه از دستور FILLASC استفاده کنید .

مثال : PLCDOWLOAD

ابتدا متناسب با توضیحات داده شده پارامترهای دستور را وارد کنید .



برنامه ماکرو به صورت زیر خواهد بود .

* در دستوراتی که در آنها نیاز به وارد کردن نام می باشد ، باید نام را با کد اسکی هر کاراکتر وارد نمایید . کد اسکی می تواند به صورت هگز یا دسیمال باشد . برای همین منظور از دستور FILLASC برای بدست آوردن کد هگز کاراکترها استفاده شده است .

14. این دستور از جمله دستورات ریاضی بوده و با داشتن سه نقطه بر روی محیط دایره مرکز دایره را محاسبه می کند . با توجه به آدرسی که در Var2 وارد می کنید مختصات سه نقطه از دایره به صورت نشان داده شده در شکل زیر وارد می شود . ورودی این دستور به صورت DW می باشد .

مختصات نقطه اول	X1	Var2
	Y1	Var2+2
مختصات نقطه دوم	X2	Var2+4
	Y2	Var2+6
مختصات نقطه سوم	X3	Var2+8
	Y3	Var2+10

در متغیر Var3 ، نتیجه دستور ذخیره می شود . در Var1 هم می توانید از اجرا موفق یا ناموفق دستور با اطلاع شوید .

مختصات مرکز دایره	X4	Var3
	Y4	Var3+2

مثال :

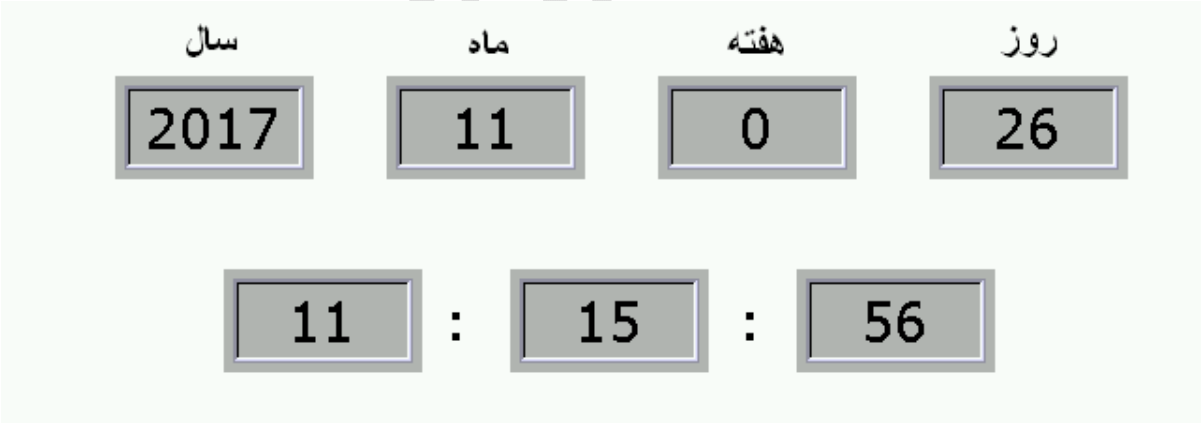
تنظیم زمان در HMI

1- دریافت و نمایش زمان :

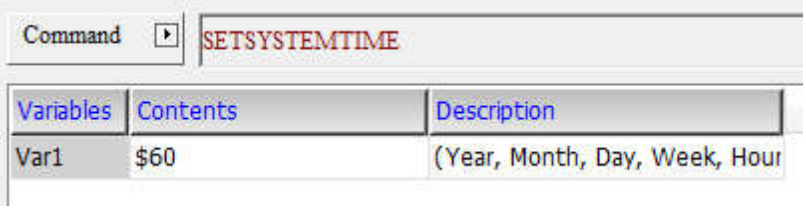

```
$50 = GETSYSTEMTIME
```

این دستور 7 رجیستر را اشغال می کند تا زمان و تاریخ HMI را نمایش بدهد . در صفحه HMI از دستور Numeric Display برای نمایش رجیستر های $50 تا $56 استفاده کنید .

Var1	سال
Var1+1	ماه
Var1+2	روز
Var1+3	هفته
Var1+4	ساعت
Var1+5	دقیقه
Var1+6	ثانیه



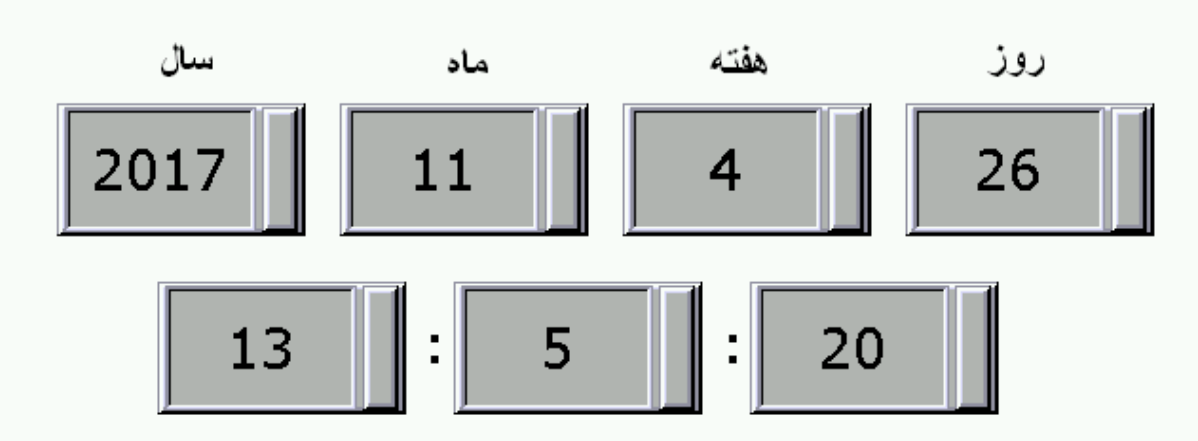
2- اعمال مقادیر دلخواه بر روی HMI :




```
SETSYSTEMTIME($60)
```

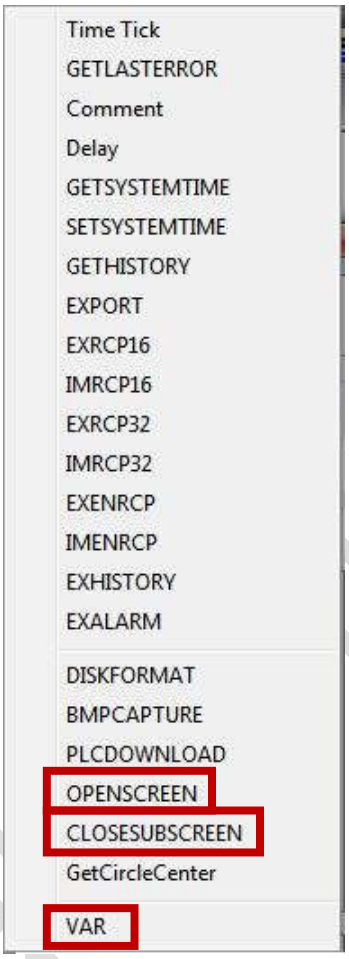
مانند مثال قبل ، این دستور نیز 7 رجیستر را اشغال می کند و ترتیب آنها مانند جدول گنجانده شده در مثال قبل می باشد . برای تغییر مقادیر تاریخ و ساعت می توانید به صورت شکل زیر ماکرو نویسی انجام دهید و یا با استفاده از دستور Numeric Entry مقادیر را وارد نمایید .

```
SETSYSTEMTIME($60)
$60 = 2017
$61 = 11
$62 = 26
$63 = 4
$64 = 13
$65 = 5
$66 = 20
```



ماکرو در DOP100 :

در HMI های سری DOP100 تمامی دستورات گفته شده نیز مورد استفاده قرار می گیرند و علاوه بر آنها دو دستور کاربردی برای کنترل اسکرین ها و یک دستور بسیار کار آمد برای تعریف متغییر در برنامه های HMI قرار داده شده است . در ادامه به شرح این دستورات پرداخته خواهد شد .



OPENSCREEN :

از این دستور برای باز کردن صفحات HMI استفاده می شود .

مثال : OPENSCREEN

در این مثال با زدن کلید M10 صفحه 4 گشوده می شود .

```
IF {Link2}1@M10 == ON
OPENSCREEN(4)
ENDIF
```

CLOSESUNSCREEN :

از این دستور برای بستن SUB Screen ها در برنامه HMI استفاده می شود . از این دستور برای کنترل SUB Screen ها از جمله کیبورد های Custom Defined یا منو ها استفاده می شود .

مثال : CLOSESUBSCREEN

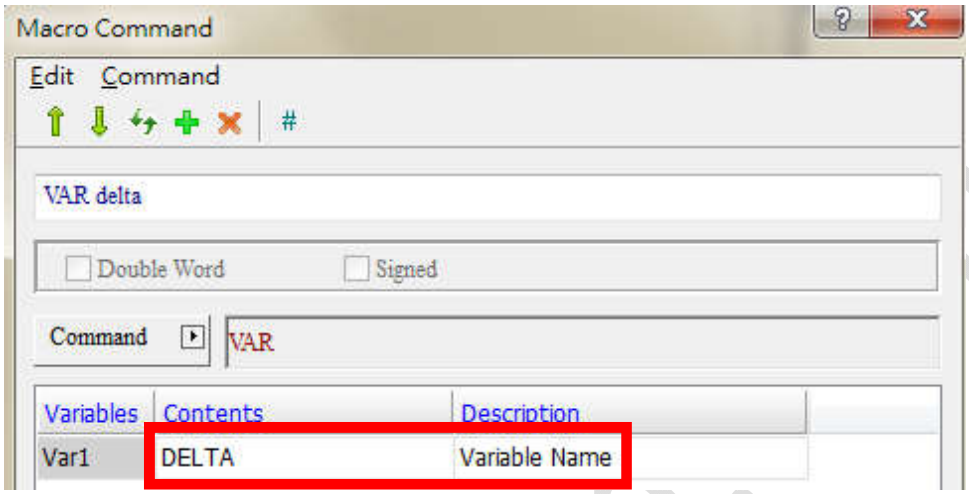
در برنامه نوشته شده در این مثال در صورتی که مقدار رجیستر D20 مساوی 100 باشد . پنجره SUB Screen فعال شماره 10 ، بسته خواهد شد .

```
IF {Link2}1@D20 == 100
CLOSESUBSCREEN(10)
ENDIF
```

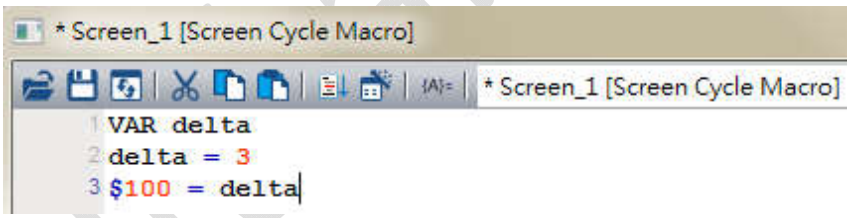
VAR :

در برنامه های ماکرو ، گاهی نیاز به تعریف یک متغییر یا Variable می باشد . Var متغییری است که هیچ آدرسی نمی پذیرد فقط باید برای آن یک نام تعیین کنید .

مثال : VAR



در برنامه زیر یک مثال ساده از نحوه استفاده VAR نوشته شده است . در این مثال مقدار متغییر DELTA در رجیستر $100 قرار داده می شود .



خطاهای ماکرو :

خطای کامپایل برنامه ماکرو :

این خطا ها در حین بررسی SYNTAX ها رخ می دهند :

کد خطا	شرح خطا
-100	LABEL برای دستور GOTO وجود ندارد .
-101	این خطا در Sub MACRO ها رخ می دهد و زمانی رخ می دهد که Sub MACRO در خودش فراخوانی شده باشد .
-102	بیش از 10 مرتبه از دستور FOR/NEXT استفاده شده است . می توانید برای کاهش تعداد دفعات استفاده از این دستو ، از دیتور IF یا GOTO استفاده کنید .
-103	این خطا زمانی رخ می دهد که در برنامه اصلی Sub MACRO با نام مشخص فراخوانی شده باشد ولی برنامه آن Sub MACRO نوشته نشده است . برای مثال ، فرض کنید در برنامه اصلی CALL 5 نوشته شده باشد یعنی Sub MACRO با نام 5 فراخوانی شود ولی Sub MACRO با نام 5 نوشته نشده باشد .
-104	تعداد دستورات NEXT کمتر از تعداد دستورات FOR می باشد . دستور NEXT دستور پایان دهنده هر مرتبه اجرای دستور FOR می باشد بنابراین به ازاء هر دستور FOR باید یک مرتبه دستور NEXT نوشته شود و بالعکس.
-105	تعداد دستورات FOR کمتر از تعداد دستورات NEXT می باشد . دستور NEXT دستور پایان دهنده هر مرتبه اجرای دستور FOR می باشد بنابراین به ازاء هر دستور FOR باید یک مرتبه دستور NEXT نوشته شود و بالعکس.
-106	نشان دهنده وجود دو LABEL هم نام در یک برنامه ماکرو می باشد .
-107	در برنامه اصلی از دستور RET استفاده شده است . دستور RET برای Sub MACRO استفاده می شود . برای خارج شدن از Sub MACRO و بازگشت به برنامه اصلی می باشد در برنامه اصلی ماکرو برای پایان دادن به دستورات از دستور END استفاده می شود .

خطای کامپایل برنامه :

این خطا ها هنگام COMPILE برنامه HMI رخ می دهند :

کد خطا	شرح خطا
-10	خطای دستور GOTO – دستور GOTO بررسی شود .
-11	خطا حافظه برای ماکرو – این خطا ممکن است به دلیل استفاده تعداد زیادی از Sub MACRO ها رخ دهد و یا به دلیل اجرای همزمان دستورات چند ماکرو رخ دهد .
-12	Sub MACRO نوشته نشده است – این خطا به دلیل فراخوانی یک Sub MACRO که نوشته نشده است رخ می دهد .
-13	خطا خواندن دیتا از یک دیوایس دیگر – این خطا در طی فرآیند خواندن دیتا های یک دیوایس دیگر مثلا PLC رخ می دهد . ممکن است محتوا دیتا خوانده شده غلط باشد .
-14	خطا نوشتن دیتا در یک دیوایس دیگر – این خطا در طی فرآیند نوشتن دیتا در یک دیوایس دیگر مثلا PLC رخ می دهد . ممکن است محتوا دیتا نوشته شده غلط باشد .
-15	مقسوم علیه صفر است – در دستور تقسیم ، مقسوم علیه صفر قرار داده شده است .
-16	خطا دستور BCD – دستور BCD نوشته شده را اصلاح کنید .
-17	خطا دستور تبدیل ASCII به HEX – دستورنوشته شده را اصلاح کنید .
-18	خطا دستور NEXT – زمانی رخ می دهد که ماکرو خطا در اجرای دستور NEXT داشته باشد .
-19	خطا دستور انتقال کاراکتر – زمانی رخ می دهد که اجرای دستور جابجایی رشته ای از کاراکتر ها (FILLASC) خطا داشته باشد .
-20	خطا دستور BIN – دستور ماکرو BIN را اصلاح کنید .
-21	خطا دیتا Sub MACRO – زمانی رخ خواهد داد که دیتا Sub MACRO در زمان فراخوانی خطا داشته باشد .
-22	خطا دستور FOR – زمانی رخ می دهد که ماکرو خطا در اجرای دستور FOR داشته باشد .
-23	خطا INITIAL - زمانی رخ می دهد که از دستور INITCOM استفاده شده باشد .
-24	خطای حافظه – زمانی رخ می دهد که حافظه اختصاص یافته HMI برای برنامه ماکرو کافی نباشد
-25	خطا COM Port – این خطا زمانی رخ خواهد داد که دستورات COM port به درستی اجرا نشوند .

-26	خطا پورت – زمان اجرای دستورات مربوط به PORT رخ می دهد .
-27	خطا خواندن مقادیر – زمانی رخ می دهد که مقادیر خارج از رنج مجاز دیتا باشند .
-28	خطا در اجرای دستورات IF ، ELSE و ENDIF .
-29	خطا در اندازه ضخامت قلم ، هنگام اجرای دستورات ترسیم .
-30	خطا دیتا HISTORY – این خطا در دستور GETHISTORY رخ می دهد .
-31	خطا در اجرا دستور EXPORT .
-32	این خطا در دستورات EXPORT و DISKFORMAT رخ می دهد و نشان دهنده ایراد در دیوایس خارجی (فلش مموری یا SDcad) می باشد .
-33	خطا Print – خطا در چاپ .
-34	خطا سرریز در اجرا دستور IF ، ELSE ، ENDIF .
-35	خطا در وارد کردن پسورد – در زمان اجرا دستورات تائید پسورد ماکرو رخ خواهد داد .
-36	خطا قفل شدن پسورد – زمانی رخ خواهد داد که پسورد وارد شده در ماکرو از حد مجاز خارج باشد .
-37	خطا شناسایی ID پسورد .
-38	Syntax ERROR – زمان اجرا دستور PLCDOWNLOAD رخ خواهد داد .
-39	خطا ارتباط – زمانی که ارتباط PLC برای دانلود برنامه بر روی آن برقرار نباشد یا با مشکل روبرو شود .

خطا برنامه PLC برای دانلود با فرمت DVP یا ISP :

کد خطا	شرح خطا
-40	خطا اسم فایل هنگام باز کردن فایل برنامه با استفاده از دستور ماکرو .
-41	ورژن برنامه برای بازکردن فایل برنامه با دستور ماکرو ساپورت نمی شود .
-42	خطا باز کردن فایل برنامه با استفاده از دستور ماکرو .
-43	خطا دسترسی به فایل برنامه با استفاده از دستور ماکرو .
-44	خواندن فایل برنامه با دستور ماکرو با مشکل مواجه شده است .
-45	خطا در باز کردن فایل برنامه – زمانی که با دستور ماکرو می خواهید فایل برنامه را باز کنید ، اگر با این خطا مواجه شوید یعنی ماکرو نمی تواند محتوا برنامه را برای باز کردن جابجا کند .
-46	نوشتن فایل برنامه با دستور ماکرو با مشکل مواجه شده است .
-47	پاک کردن فایل برنامه با دستور ماکرو با مشکل مواجه شده است .
-48	خطا تغییر نام فایل برنامه .
-49	خطا طول دیتا فایل برنامه .
-50	خطا در محتوا فایل برنامه .