

### مثال : اجرای دستور جمع توسط C-Action

پس از تعریف سه عدد تگ داخلی بصورت Unsigned 32 bit با نامهای f1 ، f2 و fsum تنظیمات زیر را انجام داده و کد C-Action مربوطه را به تحریک موس اتصال می دهیم .

**Object Properties**

Properties	Events	Action
Rectangle	Mouse Click	⚡
	Keyboard	
	Press Left	⚡
	Release Left	⚡
	Release Right	⚡

```

#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpsz...
{
int f1,f2,f3;

f1=GetTagWord("f1");
f2=GetTagWord("f2");
f3=f1+f2;
SetTagWord("fsum",f3);

// WINCC:TAGNAME_SECTION_START
// syntax: #define TagNameInAction "DMTagname"
// next TagID : 1
// WINCC:TAGNAME_SECTION_END
    
```



**مثال : برنامه متوسط گیری از سه عدد توسط C-Action**

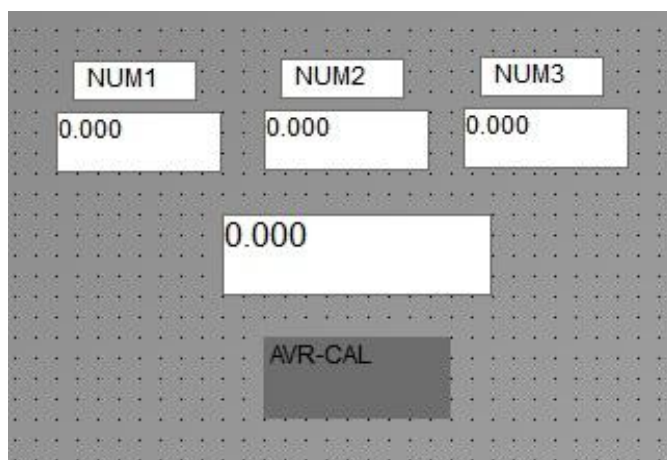
پس از تعریف چهار عدد تگ داخلی بصورت Float 32 bit با نامهای NUM1 الی NUM3 و AVR ، در C-Action مربوط به تحریک کلید چپ یک شی مستطیل که به منظور فرمان انجام متوسط گیری در نظر گرفته شده است ، دستورات زیر تایپ شده و سپس با قراردادن چهار عدد ابزار I/O Filed برای نمایش چهار تگ فوق و مقداردهی در محیط RunTime برنامه را چک می نمایم.

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName, char*
lpszPropertyName, UINT nFlags, int x, int y)
{
float val1,val2,val3,avr_tag;
val1=GetTagFloat("NUM1"); //Return-Type: float
val2=GetTagFloat("NUM2"); //Return-Type: float
val3=GetTagFloat("NUM3"); //Return-Type: float

avr_tag=((val1+val2+val3)/3);
SetTagFloat("AVR",avr_tag); //Return-Type: BOOL

// WINCC:TAGNAME_SECTION_START
// syntax: #define TagNameInAction "DMTagname"
// next TagID : 1
// WINCC:TAGNAME_SECTION_END

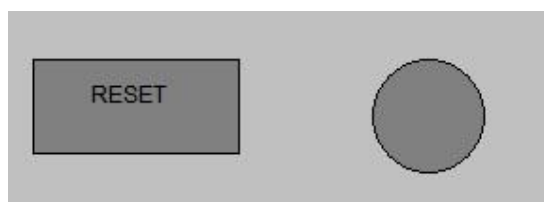
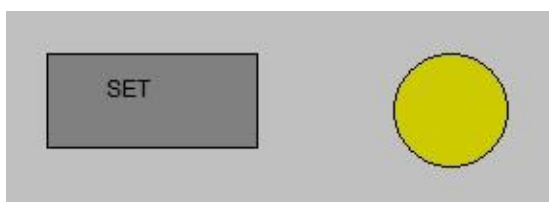
// WINCC:PICNAME_SECTION_START
// syntax: #define PicNameInAction "PictureName"
// next PicID : 1
// WINCC:PICNAME_SECTION_END
}
```



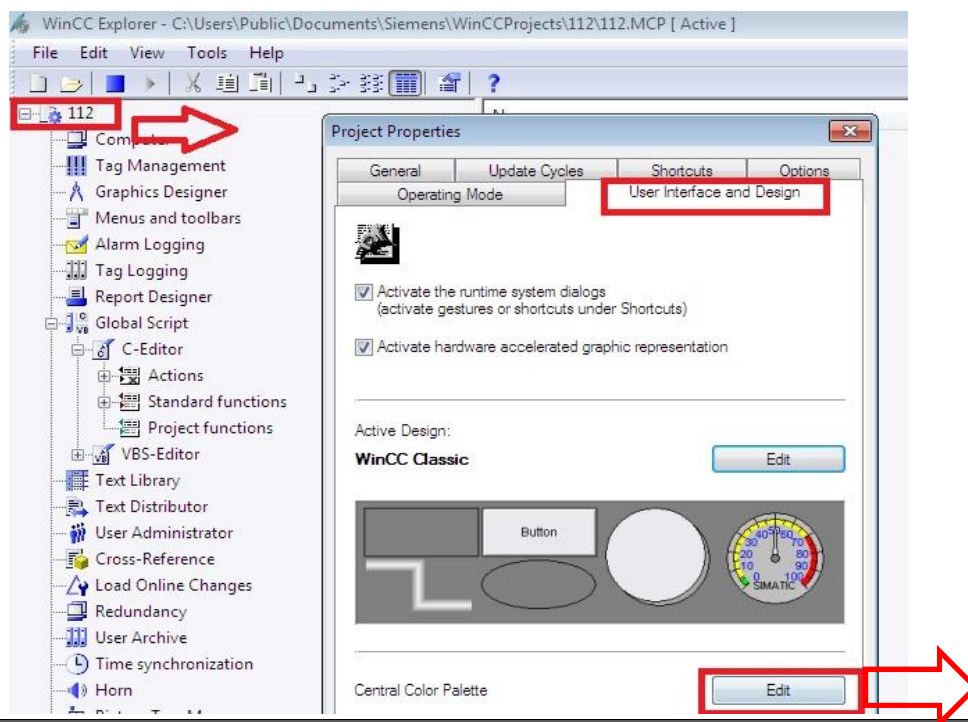
**مثال : ساخت Push-Button بصورت Momumentry توسط C-Action**

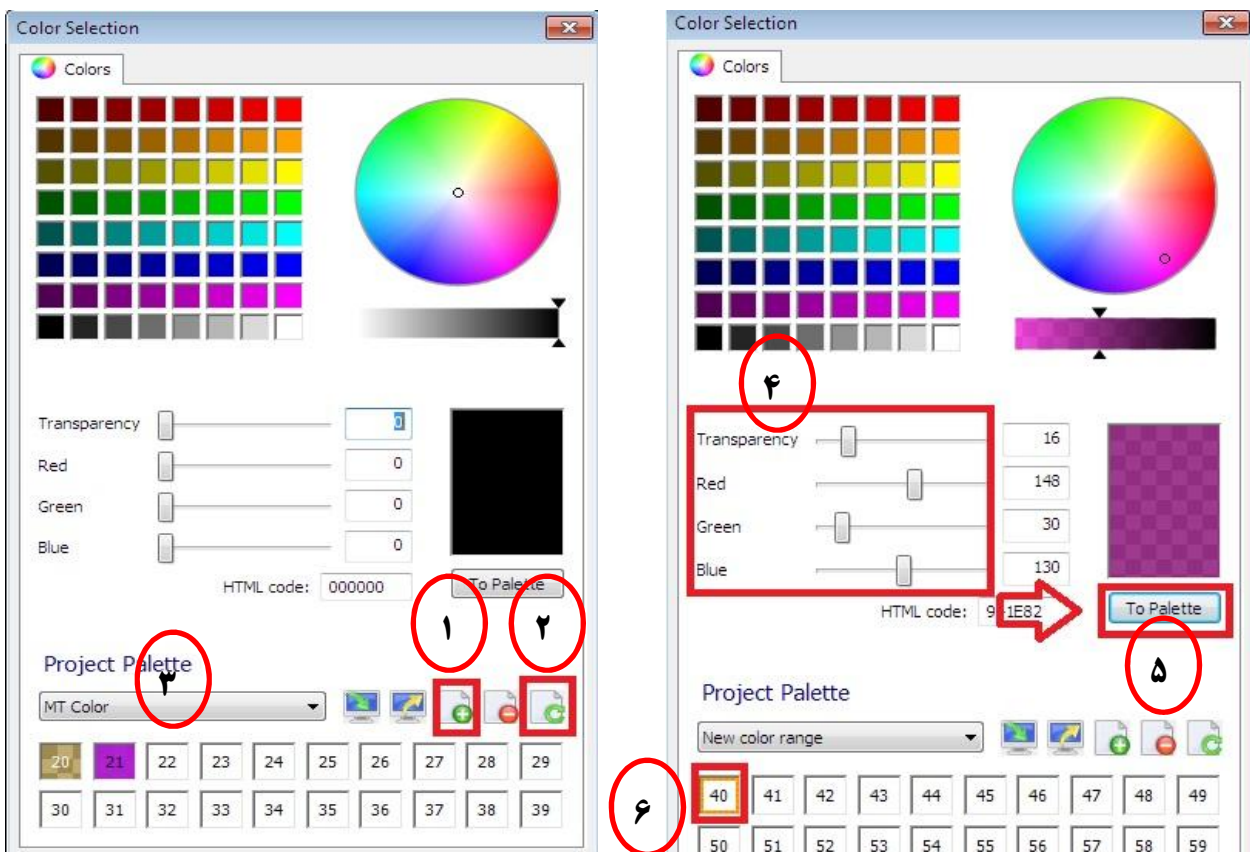
ابتدا یک عدد تگ بصورت Bool تعریف نموده و سپس در C-Action مربوط به تحریک Mouse بر روی یک ابزار مستطیل کد زیر را وارد می کنیم. در صورتی که کلید تحریک شده و مقدار تگ برابر یک باشد ، مقدار صفر جایگزین شده و در حالت عکس و در صورتی که مقدار تگ برابر یک باشد ، مقدار صفر جایگزین می شود. در این صورت کلید مورد نظر ساخته شده است .

```
if(GetTagBit("valve_tag")==1)
SetTagBit("valve_tag",0); //Return-Type: BOOL
else
SetTagBit("valve_tag",1); //Return-Type: BOOL
```

**مثال : تغییر رنگ پس زمینه توسط برنامه نویسی با کد C**

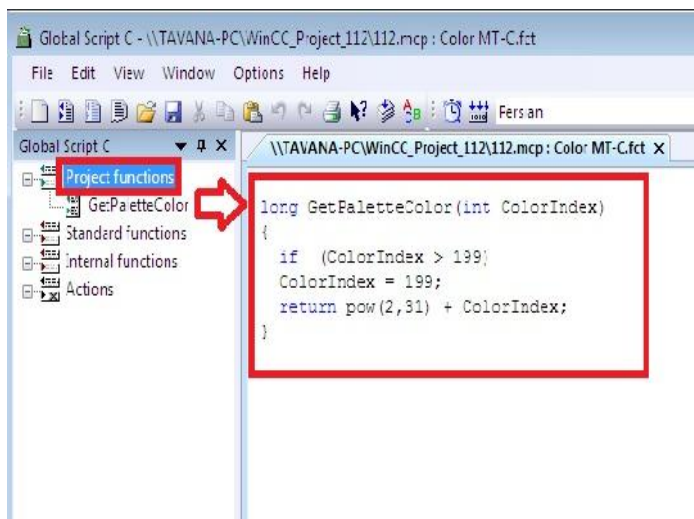
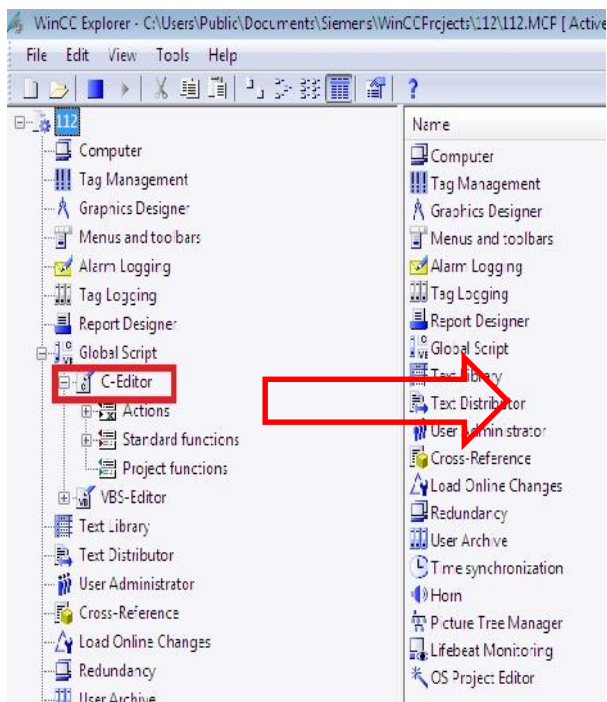
در این مثال می خواهیم در محیط RunTime با تحریک یک کلید ، رنگ پس زمینه صفحه اصلی تغییر یابد  
۱- ایجاد یک پالت رنگ جدید و تعریف رنگ های دلخواه :





بر اساس الگوی فوق تعدادی کد رنگ ایجاد نموده و هر یک را در خانه ای از پالت تنظیم نمایید.

در صفحه اصلی WinCC برنامه C-Editor را باز نمایید و پس از ایجاد یک Project Function جدید ، کد زیر را با رعایت حروف کوچک و بزرگ در آن بنویسید .



توجه : شماره هر مربع که رنگی در آن ریخته می شود در ادامه به عنوان شماره رنگ مورد نظر در دستورات استفاده می شود . البته بصورت پیش فرض خود نرم افزار تعدادی از شماره های اول این خانه را با رنگ های پیش فرض پر نموده است که اگر بخواهیم از آنها استفاده کنیم نیازی به طی مراحل فوق نمی باشد .

```
long GetPaletteColor(int ColorIndex)
{
    if (ColorIndex > 199)
        ColorIndex = 199;
    return pow(2,31) + ColorIndex;
}
```

مجموعه دستورات فوق از پالت رنگ ، یک کد رنگ را دریافت نموده و کد رنگ مربوطه را باز می گرداند .

## ۲- تنظیمات محیط گرافیک

در محیط گرافیک ابتدا در مشخصه صفحه پروژه تنظیمات Global Color Scheme را غیر فعال نمایید. سپس یک شی مستطیل در صفحه قرار داده و در تحریک موس آن یک کد Action تعریف نموده و دستور زیر را در آن بنویسید . **دقت** **Main** نام صفحه اصلی است که می خواهیم رنگ پس زمینه آن تعویض شود و عدد یک شماره رنگ مورد نظر از پالت رنگ ها است .

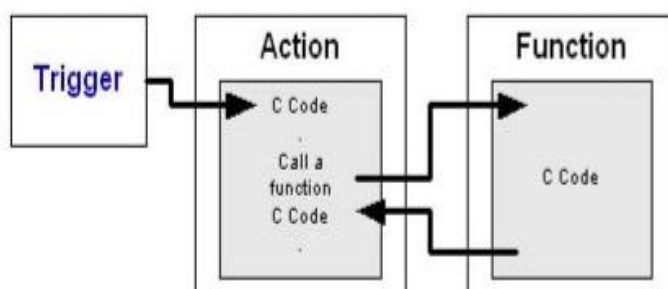
```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
lpszPropertyName)
{
    SetBackColor(lpszPictureName, "Main", GetPaletteColor(1));
}
```

توجه در صورتی که در این دستور نام مربوط به هر شی دیگر را جایگزین نموده و برای آن شی نیز Global Color Scheme را غیر فعال کنیم همین شرایط انجام شده و رنگ پس زمینه تغییر می کند. دستور فوق دستور SetBackColor یعنی تغییر رنگ پس زمینه می باشد . در این دستور از یک فانکشن که قبلا در محیط برنامه نویسی C ساخته بودیم ( GetPaletteColor(1) ) برای فراخوانی رنگ از پالت استفاده کرده ایم .

در ادامه توضیحاتی در خصوص نحوه ایجاد فانکشن و سپس فراخوانی آن در Action ذکر می گردد.



ساختار کد نویسی C در WinCC به این نحو می باشد که می توان بصورت مستقیم یک C-Action در محیط گرافیک ایجاد نمود و آنرا به یک تریگر ( به عنوان مثال تحریک موس بر روی یک شیء ) متصل نمود . البته تریگر ها می تواند در انواع مختلف دیگری از جمله تغییرات مقدار یک تگ ، اتمام یک زمان سنجی و... نیز تعریف گردد . همچنین می توان با ایجاد یک فانکشن ، آنرا در داخل C-Action فراخوانی نمود .



فانکشن ها در سه دسته قابل تعریف می باشند :

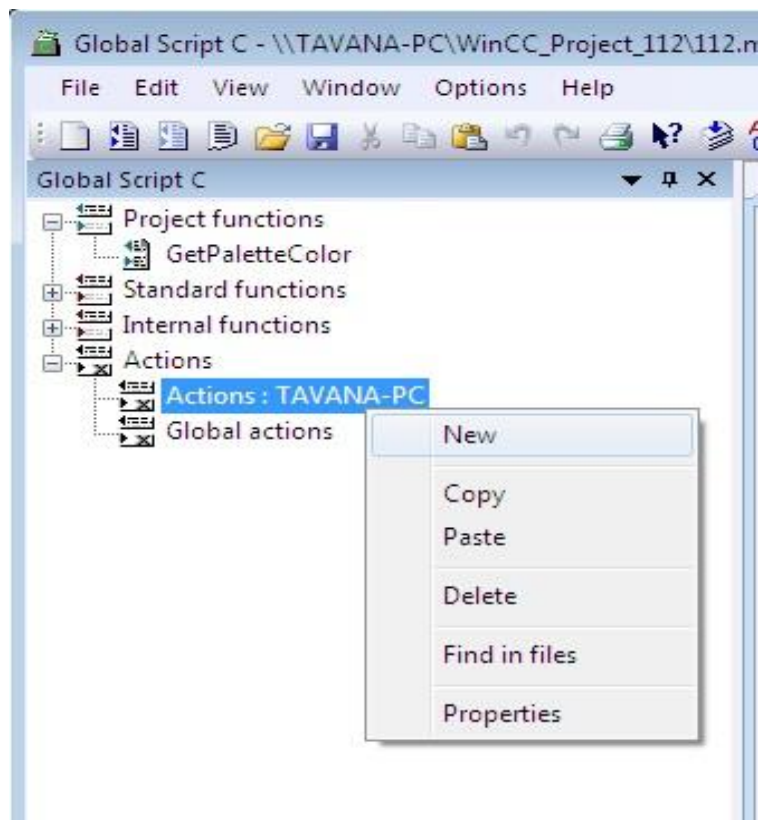
### Project Function و Internal Function – Standard Function

که دو دسته اول توسط شرکت سازنده تعریف و ایجاد شده و کاربر می تواند بدون امکان ویرایش آنها را در برنامه خود و بدون نیاز به تریگر استفاده نماید . نوع سوم یا Project Function ها توسط کاربر ایجاد و ویرایش می شوند .

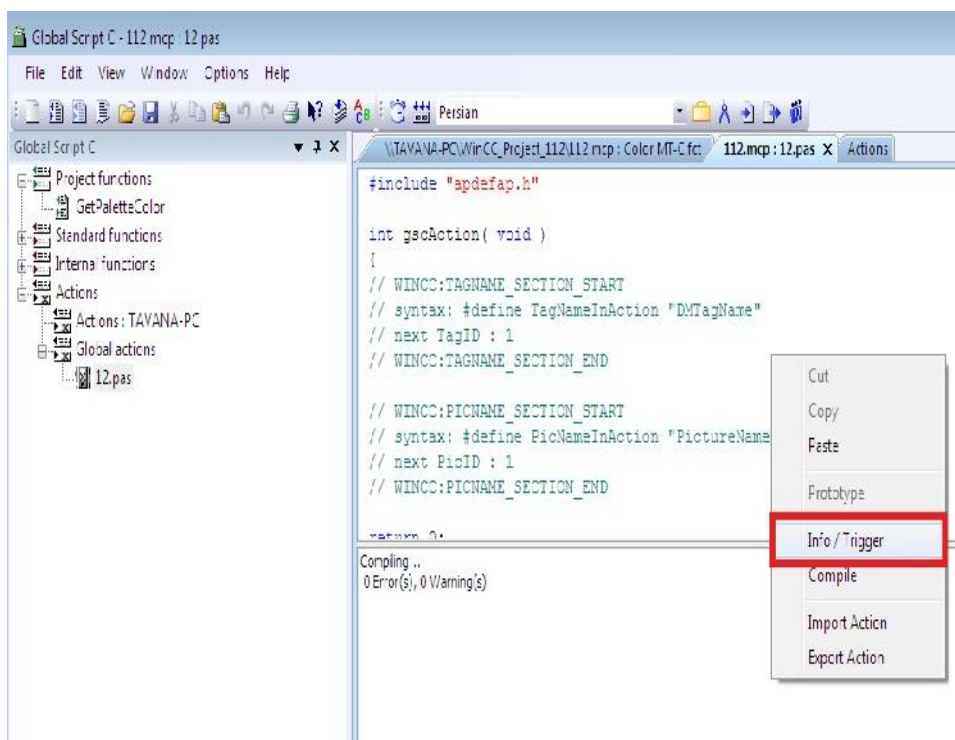
برای ایجاد یک فانکشن همانگونه که در مثال فوق توضیح داده شد می توان در محیط Global C-editor یک فانکشن جدید ایجاد نموده و پس از کدنویسی آنرا ذخیره نمود . در این حال این فانکشن همواره وجود داشته و در صورت لزوم می توان آنرا مورد استفاده قرار داد .

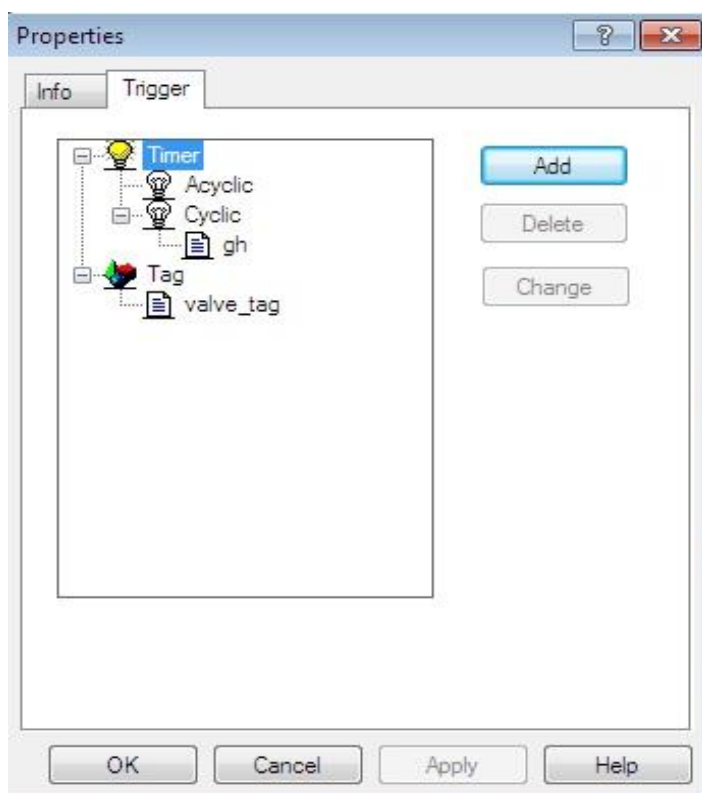
جهت ایجاد Action هم می توان همانند شکل زیر بصورت Global یا LOCAL ( فقط در یک Client ) نسبت به تعریف Action اقدام نمود . البته در محیط گرافیک نیز می توان همزمان تریگر و Action را مطابق با توضیحات مثالهای قبل ایجاد کرد .





جهت اتصال یک تریگر به یک Action می توان در محیط برنامه نویسی یک Action کلیک راست نمود و گزینه info/Trigger را انتخاب نمود و سپس با توجه به نیاز یک تریگر تعریف نمود.





در تنظیمات فوق در حالت سیکلی در یک بازه زمانی بر اساس الگوی تعریف شده Action اجرا می شود . در حالت Acyclic در یک زمان مشخص شده فقط یکبار اجرا می شود و در حالت Tag می توان برای یک Action یک یا چند تگ تعریف نموده که در صورت تغییر در آنها Action اجرا گردد .

### مثال : ذخیره سازی اطلاعات ورود / خروج کاربران با بهره گیری از کد C

با توجه به اینکه با هر ورود و خروج کاربران ( Log In / Log out ) ، تگ داخلی سیستمی به نام @CurrentUserName تغییر مقدار می دهد ، لذا می توان از این تغییرات این تگ به عنوان تریگر نمودن یک Action با کد دستورات زیر بهره گرفت . این Action با توجه به وضعیت Log In یا Log Out یک تگ بیتی متناظر را SET می کند . لذا دو عدد تگ بیتی داخلی نیز به نامهای LoginBit و LogoutBit تعریف می نمایم .

با باز نمودن C-Editor با کلیک راست بر روی قسمت Global Action یک فایل جدید ایجاد نموده و کد زیر را در داخل تابع می نویسیم . در این تابع دو متغیر جهت ذخیره سازی نام کاربری قبل و نام کاربری جدید در نظر گرفته شده است که اطلاعات خود را از تگ های سیستمی مربوطه دریافت می نمایند . در مقایسه انجام





شده ، در صورتی که طول کاراکتر در نام کاربری جدید مخالف صفر باشد ، تگ Logout برابر صفر شده و در تگ Log In یک لبه بالارونده ایجاد می شود . در غیر این صورت در تگ Logout یک لبه بالارونده ایجاد می شود .

```
#include "apdefap.h"

int gscAction( void )
{

#pragma code ("kernel32.dll")
void Sleep (int a);
#pragma code ()
char *NewUser, *OldUser;
OldUser=GetTagChar ("@OldUser");

if (strlen(NewUser)!=0)
{
SetTagBit ("LogoutBit",0);
SetTagBit ("LoginBit",0);

Sleep(1000);

SetTagBit ("LoginBit",1);
SetTagChar ("@OldUser",NewUser);
}
else
{

SetTagBit ("LoginBit",0);

Sleep(1000);

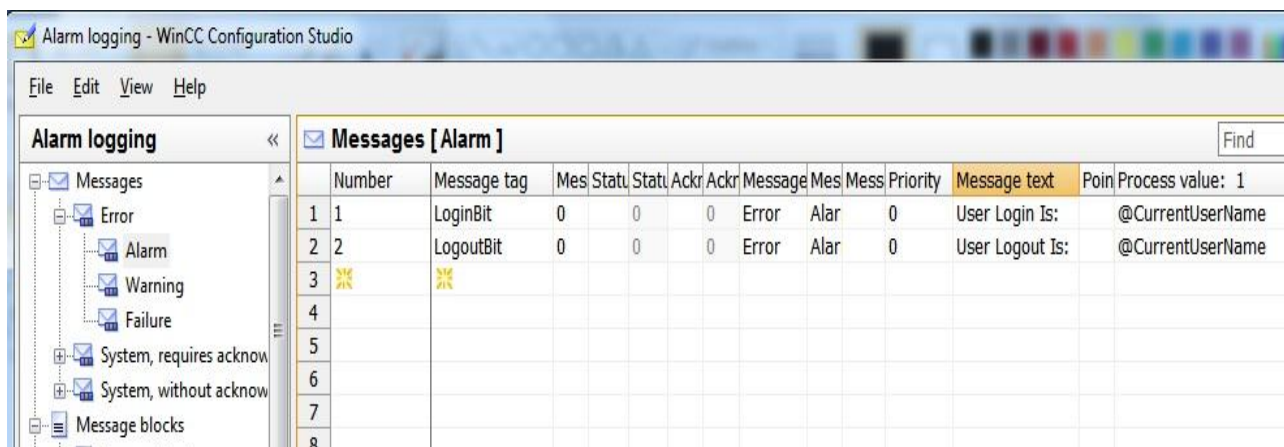
SetTagBit ("LogoutBit",1);
}

return 0;
```

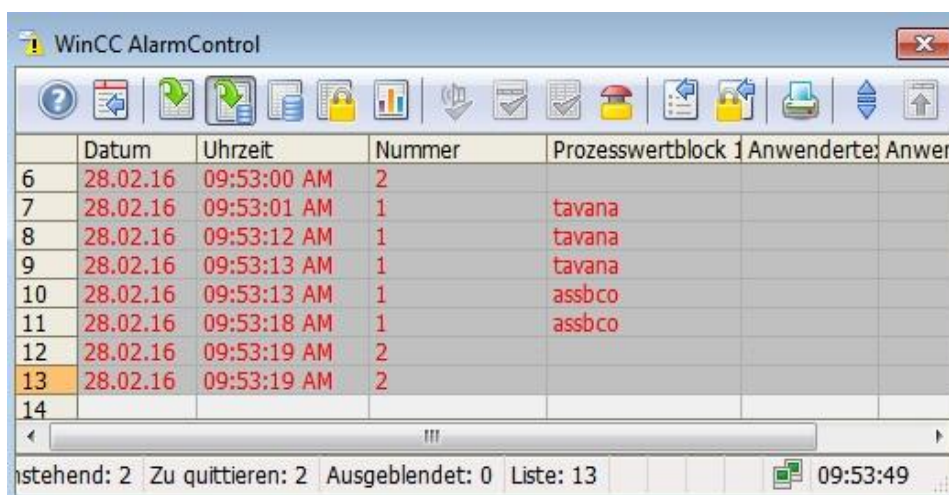
پس از نوشتن دستورات فوق ، در محلی خالی از صفحه کد نویسی کلیک راست نموده و جهت اعمال یک تریگر ، گزینه Info/Trigger را انتخاب می کنیم . در صفحه ای مطابق با شکل زیر تریگر از نوع تگ را به تگ سیستمی نام کاربری (@CurrentUserName) متصل می کنیم تا در صورت تغییر وضعیت این تگ ،



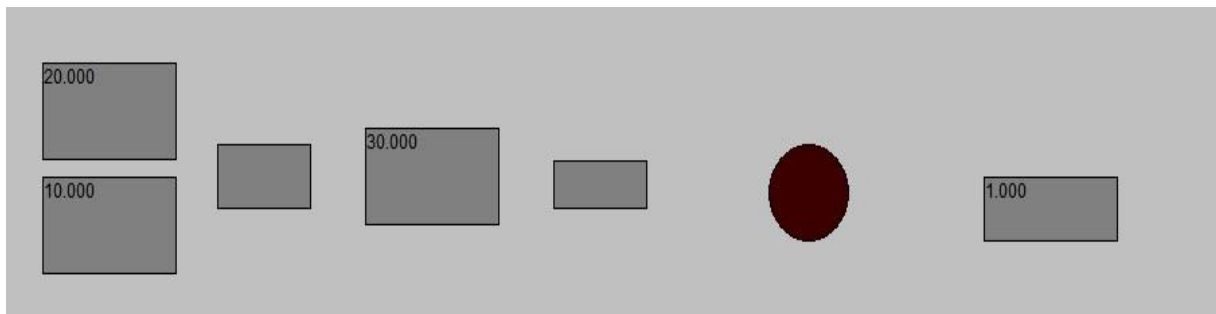
Action نوشته شده فراخوانی شود . در مرحله بعد Action را ذخیره نمود و سپس جهت نمایش اطلاعات مربوط به ورود/ خروج کاربران ، از Alarm Logging استفاده می کنیم. پس از باز نمودن محیط آلام ، جهت نمایش نام کاربر ، ابتدا از قسمت Message Block به Process value کادر Process value اضافه می نمایم. سپس در قسمت آلام ها مطابق با شکل زیر دو عدد آلام را به تگ های مربوطه متصل نموده و در کادر Process value نیز نام کاربر جاری را ذکر می کنیم.



در محیط Runtime پس از هر ورود/ خروج آلامی با ذکر تاریخ و زمان و نام کاربر مربوطه ایجاد می گردد .



مثال : برنامه ای بنویسید که ابتدا دو عدد را از اپراتور دریافت نموده و سپس با استفاده از C-Action مربوط به تحریک یک کلید حاصل جمع آنها را محاسبه نماید . سپس با تحریک کلید دوم ، ابتدا لامپی را روشن نموده و با شمارش معکوس زمان به میزان عدد حاصل جمع مرحله قبل لامپ را خاموش نماید .



Tag Management - WinCC Configuration Studio

File Edit View Help

**Tag Management**

- Internal tags
  - SIMATIC S7 Protocol Suite
    - MPI
    - PROFIBUS
    - Industrial Ethernet
    - Slot PLC
    - TCP/IP

**Tags [ Internal tags ]**

Name	Data type	Length	Format ada
13 @TLGRT_AVERAGE_TAGS_PER_SECOND	Floating-point number 64-bit IEEE	8	
14 @TLGRT_SIZEOF_NLL_INPUT_QUEUE	Floating-point number 64-bit IEEE	8	
15 @TLGRT_SIZEOF_NOTIFY_QUEUE	Floating-point number 64-bit IEEE	8	
16 @TLGRT_TAGS_PER_SECOND	Floating-point number 64-bit IEEE	8	
17 cat	Text reference	4	
18 dt	Unsigned 16-bit value	2	
19 language_eng	Text reference	4	
20 languaqe_per	Text reference	4	
21 sum_var	Unsigned 16-bit value	2	
22 x1_var	Unsigned 16-bit value	2	
23 x2_var	Unsigned 16-bit value	2	
24 k_var	Unsigned 16-bit value	2	
25 bit_lamp_color	Binary Tag	1	
26			
27			
28			
29			

Tag Management

- Alarm logging
- Tag Logging
- Text Library
- User Administrator

Edit Action

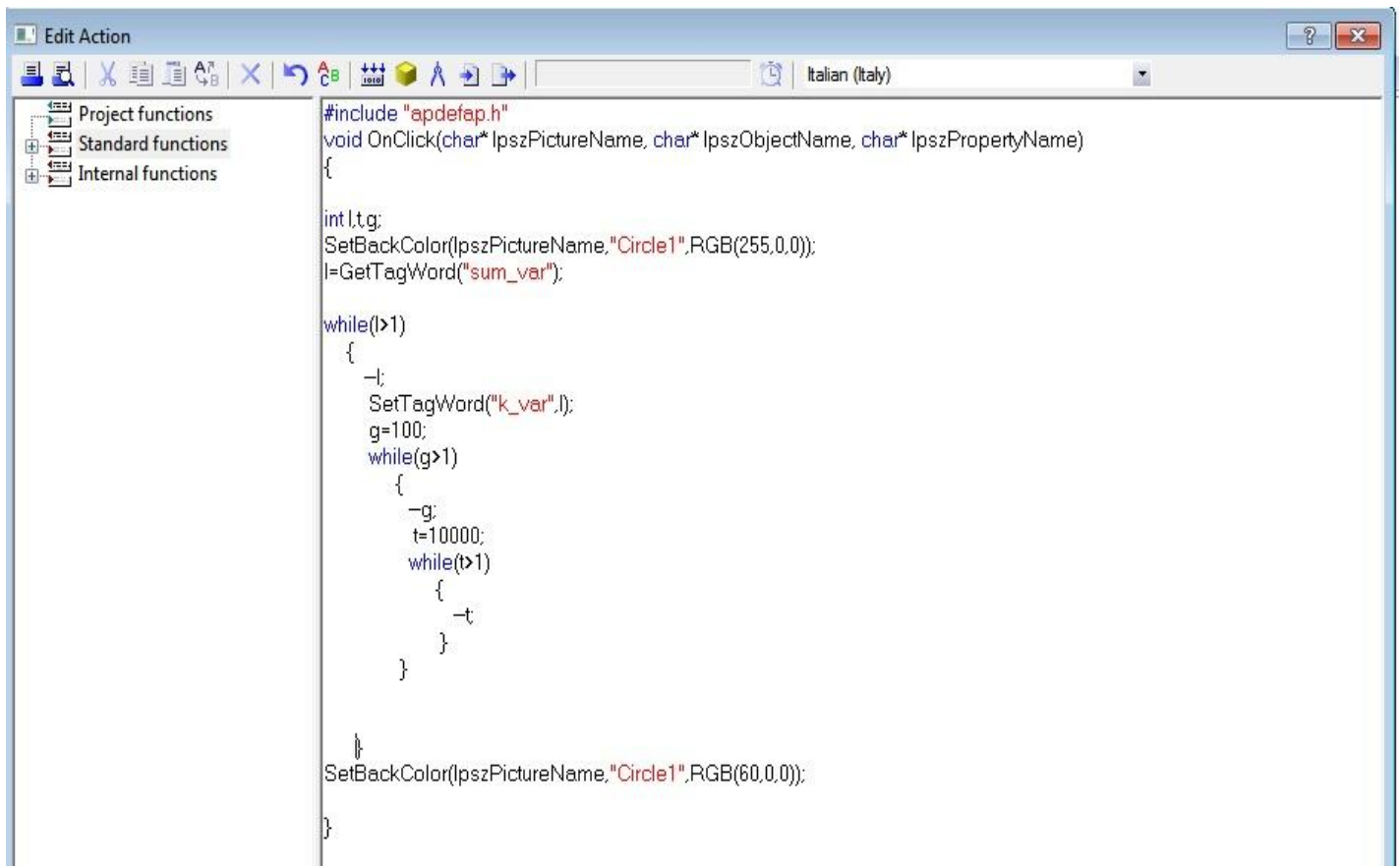
Italian (Italy)

- Project functions
- Standard functions
- Internal functions

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName)
{
    int x1,x2,sum;
    x1= GetTagWord("x1_var");
    x2= GetTagWord("x2_var");

    sum=x1+x2;
    SetTagWord ("sum_var",sum);
}
```





```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName)
{
    int l,t,g;
    SetBackColor(lpszPictureName,"Circle1",RGB(255,0,0));
    l=GetTagWord("sum_var");

    while(l>1)
    {
        -l;
        SetTagWord("k_var",l);
        g=100;
        while(g>1)
        {
            -g;
            t=10000;
            while(t>1)
            {
                -t;
            }
        }
    }

    SetBackColor(lpszPictureName,"Circle1",RGB(60,0,0));
}
```

