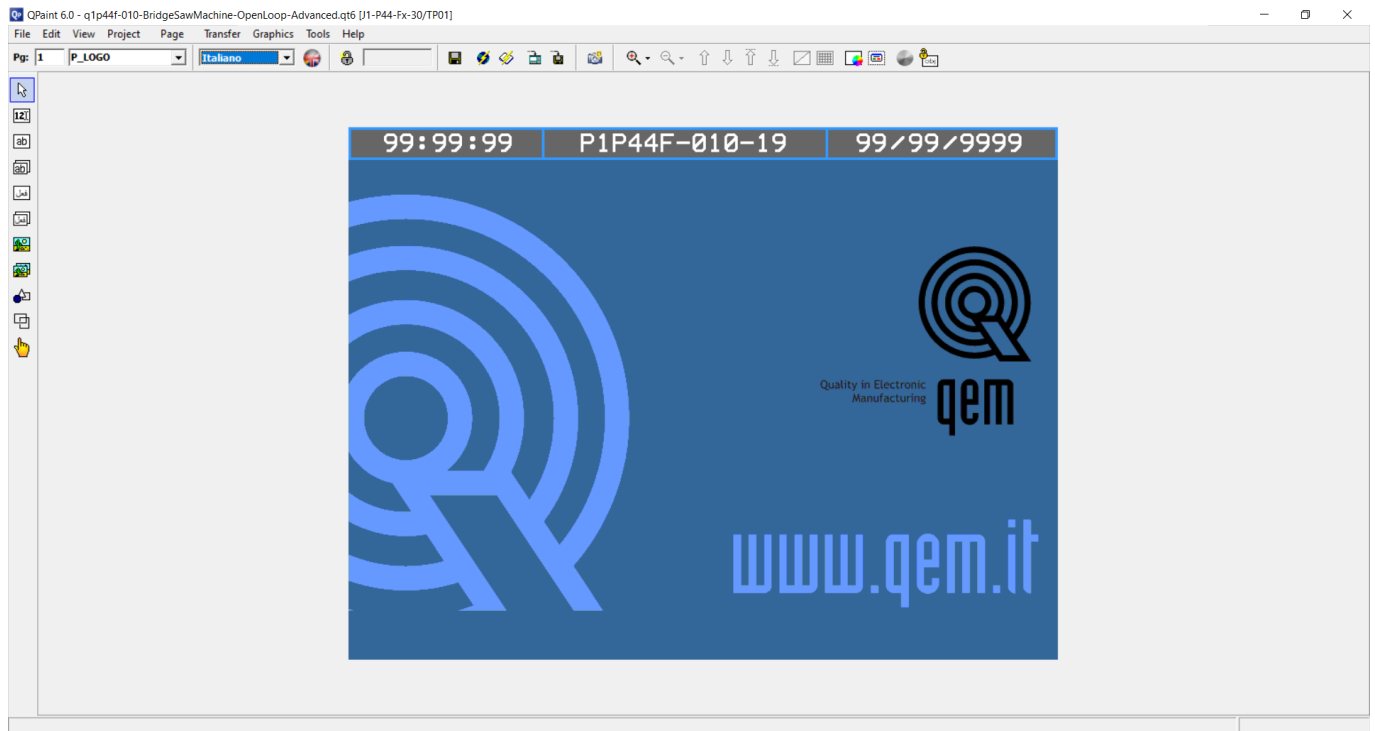


Sommario

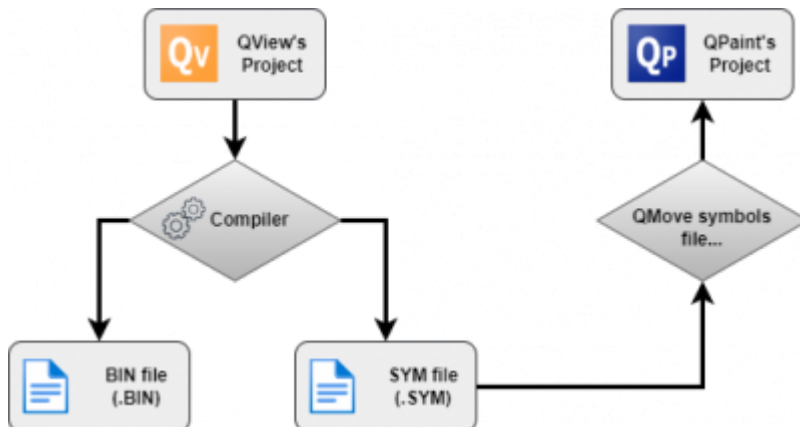
QPaint 6	3
0.1 Introduzione	3
0.2 Quick Start	3
0.2.1 Nuovo Progetto	3
0.2.2 Symbols file importing	4
0.2.3 Pagine	5
0.2.4 Oggetti	5
0.2.5 Eventi e Azioni	8
0.2.6 Variabili di terminale	9
1. Controllo da QView	9
1.1 Rilevare la pressione di un tasto	9
1.2 Capire quale pagina è visualizzata	11
1.3 Comandare un cambio pagina	11
1.4 Dataentry attivo	12
2. Esempi	13
2.1 Pulsante grafico	13
2.2 Gestire gli I/O nel terminale	19
2.3 Paradigmi di programmazione	21
2.3.1 Paradigmi di programmazione: senza device MMIQ2	21
2.3.2 Paradigmi di programmazione: con device MMIQ2	22
3. Guide	22
Vector image	22
Utilizzo	23
Creazione plot	24
Colori	24
Inserimento Figure	27
Riferimenti coordinate Touchscreen VS riferimenti Vector Image	27

QPaint 6



0.1 Introduzione

QPaint è un ambiente di sviluppo grafico per la programmazione di una interfaccia operatore QEM. In questo documento sono riportate le caratteristiche principali del programma QPaint. Durante la descrizione dell'ambiente QPaint si faranno molti riferimenti ai concetti dell'ambiente di sviluppo QView che è l'ambiente per sviluppare il software per l'automazione. Il progetto realizzato con QPaint può accedere a tutte le variabili, parametri e altre strutture dati dichiarate nel progetto realizzato con QView. Il meccanismo per “sincronizzare” il progetto QPaint con le variabili di QView è schematizzato nella figura seguente:



Al momento della compilazione del progetto QView, viene generato un file con lo stesso nome del progetto ed estensione “.sym”. Il progetto QPaint può utilizzare questo file per poter accedere a tutti i dati.

Il QPaint è un ambiente di sviluppo che permette di creare un'interfaccia operatore senza l'utilizzo di un linguaggio di programmazione, ma tramite un ambiente intuitivo che permette fin da subito di apprezzare la grafica del risultato finale.

0.2 Quick Start

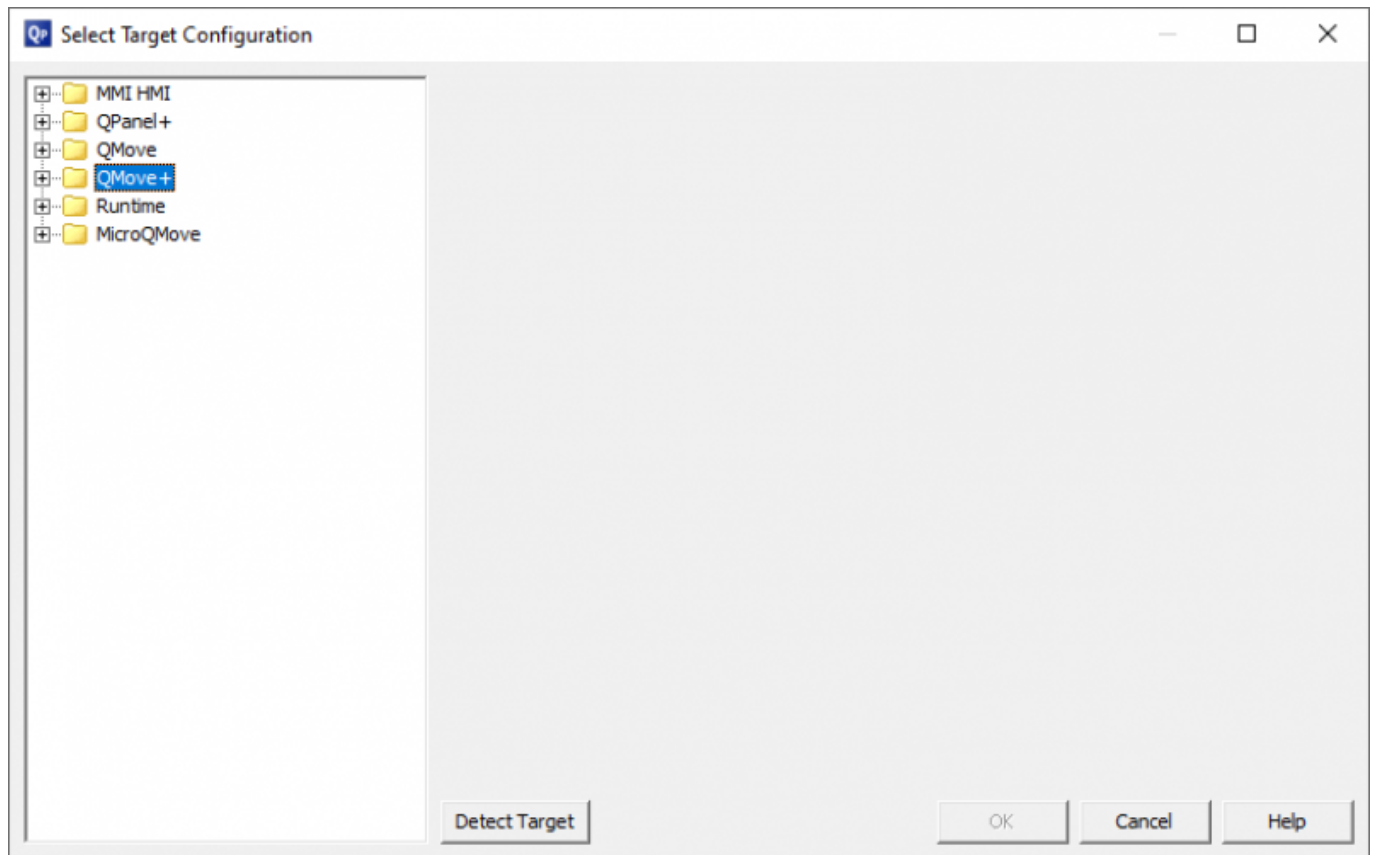
Diamo alcune informazioni fondamentali per un rapido avvio di un nuovo progetto QPaint.

0.2.1 Nuovo Progetto

Si deve selezionare

- File - New Project...

e il QPaint chiedere subito di specificare il modello del prodotto per cui si vuole realizzare il progetto:

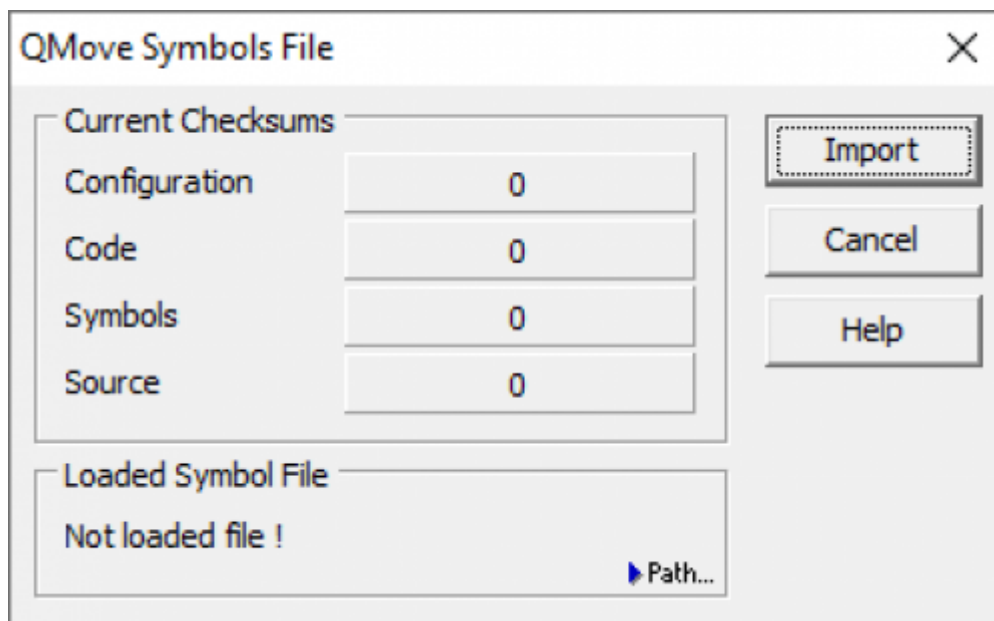


0.2.2 Symbols file importing

Una delle prime operazioni da fare è importare il file che permette di accedere ai simboli dichiarati nel progetto QView associato. Selezionare:

- Project - QMove Symbols File...

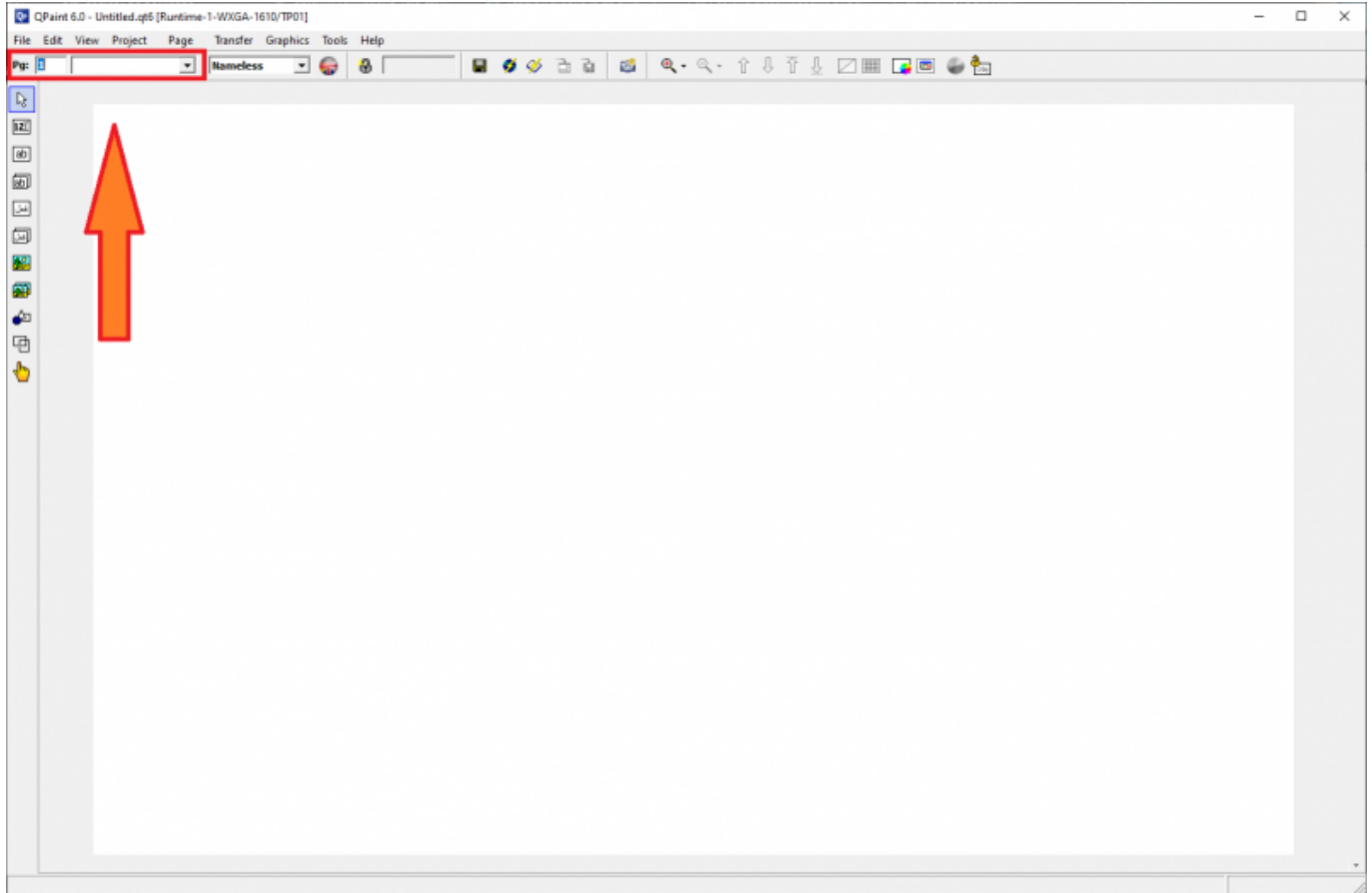
Appare la seguente finestra di dialogo



dopo avere premuto il bottone *Import*, si deve specificare il file dei simboli. Questo file viene generato da QView dopo la compilazione di un progetto. Esso ha lo stesso nome del progetto QView, ma con estensione “.sym”.

0.2.3 Pagine

Un progetto QPaint è composto da un certo numero di *Pagine* che sono le visualizzazioni che appaiono sul display e con cui l'operatore dovrà interagire. La lista delle pagine presenti è nell'angolo in alto a sinistra, esse possono essere identificate con un numero o un nome univoco:

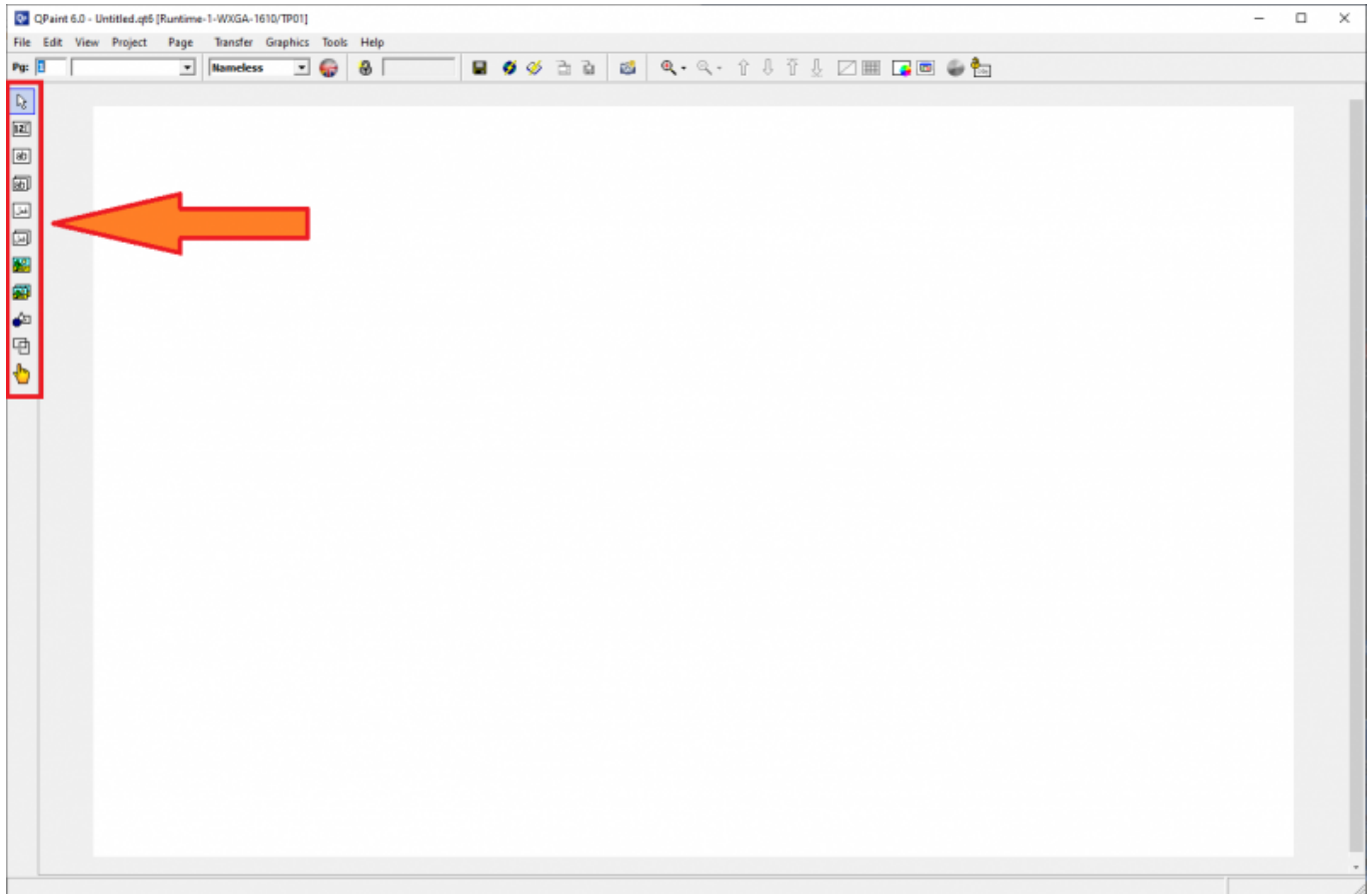


Esiste il menù *Page* per poter gestire le pagine del progetto:

- inserire,
- aggiungere,
- togliere,
- copiare,
- incollare,
- importare,
- esportare,
- impostare il colore dello sfondo,
- programmare gli eventi e le azioni di una pagina.

0.2.4 Oggetti

Nelle pagine è possibile inserire un certo numero di oggetti grafici per vari scopi. Per selezionare quale oggetto inserire nella pagina si usa la barra verticale indicata nella seguente figura:



0.2.4.1 Value Object



È l'oggetto che serve per visualizzare il valore di una variabile o di un parametro.

0.2.4.2 String Object



È l'oggetto che serve per scrivere del testo fisso sulla pagina.

0.2.4.3 ValString Object



Con questo oggetto è possibile far apparire un messaggio variabile dipendente dal valore di una variabile del progetto QView.

0.2.4.4 UniString Object



È un oggetto con la stessa funzionalità del *String Object* ma con la possibilità di usare i font installati sul PC ed in particolare i font Unicode che permettono l'uso di alfabeti diversi da quello latino: cirillico, greco, arabico, cinese, ...

0.2.4.5 UniValString Object



È un oggetto con la stessa funzionalità del *ValString Object* ma con la stessa caratteristica del *UniString Object*.

0.2.4.6 Image Object



Questo oggetto permette di inserire un'immagine nella pagina. L'immagine deve essere presente nella libreria delle immagini presente nel progetto. Per accedere alla libreria delle immagini selezionare

Graphics ➡ Image Manager

L'*Image Manager* serve per poter caricare le immagini nella libreria ed eventualmente fare alcune modifiche relative alle dimensioni.

0.2.4.7 VallImage Object



Questo oggetto permette di aggiungere sulla pagina una immagine variabile in funzione del valore di una variabile del progetto QView. Tutte le immagini utilizzate in un oggetto di questo tipo devono essere delle stesse dimensioni. Le immagini devono essere scelte dalla libreria delle immagini già inserite in *Image Manager*.

0.2.4.8 Vector Image Object



Questo oggetto permette di riservare un'area della pagina dedicata al disegno di forme elementari. Questo oggetto deve obbligatoriamente essere associato ad un array di word. Esistono delle [funzioni QCL \(QView\)](#) che forniscono delle primitive di disegno (linee, rettangoli, archi, cerchi,...).

[GUIDA ALL'UTILIZZO](#)

0.2.4.9 Box Object



È un oggetto che permette di realizzare delle cornici di forma rettangolare.

0.2.4.10 Touch Area



L'oggetto *Touch Area* crea un'area di tocco che permette di attivare degli eventi legati al tocco dello schermo da parte dell'operatore. Questo oggetto potrà essere utilizzato solamente nei terminali operatore dotati di touch screen. Gli eventi possono essere:

- **On Touch Press**, è il momento in cui il dito entra in contatto con lo schermo;

- **On Touch Release**, è il momento in cui il dito rilascia lo schermo,
- **On Touch Press Double**, è un doppio tocco ravvicinato nel tempo.

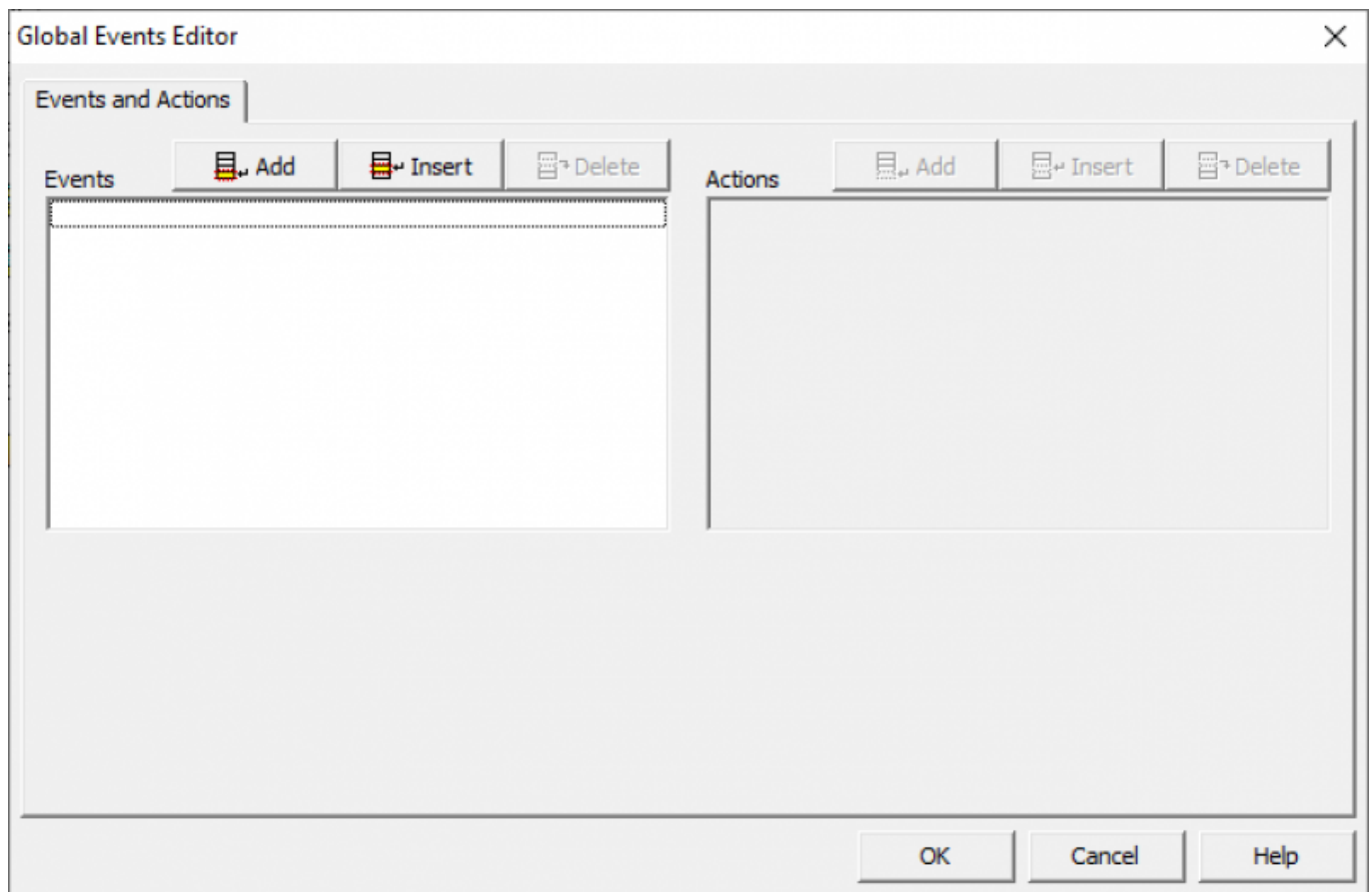
ad ognuno di questi eventi possono essere associate delle azioni. Le azioni sono di tipo generico e non legate al concetto di touch screen e quindi verranno descritte in seguito.

0.2.5 Eventi e Azioni

Oltre alla composizione della grafica, è possibile anche implementare nel progetto una semplice programmazione. Il principio è quello di avere a disposizione una serie di *eventi* da cui scegliere. Ad ogni evento io posso associare delle *azioni*. Inoltre posso programmare queste associazioni *evento-azioni* in modo che abbia effetto in tutte le pagine oppure solamente nella pagina visualizzata in quel momento. Per questo motivo esistono due contesti:

1. Project - Global Events Editor
2. Page - Page Events Editor

Entrambi gli *Editors* sono composti di due aree come mostrati in figura



A sinistra si inseriscono gli *eventi* e a destra si inseriscono le *azioni* associate all'evento selezionato a sinistra.

0.2.5.1 Eventi

La lista degli eventi possibili sono:

- **On Key**, evento che si ripete continuamente finché è premuto il tasto specificato;
- **On Press**, evento che si attiva alla pressione del tasto specificato;
- **On Release**, evento che si attiva al rilascio del tasto specificato;
- **On Always**, evento che si ripete continuamente;
- **On Page In**, evento che si attiva all'ingresso della pagina;
- **On Time**, evento che si attiva in un certo momento specificato da una data e un'ora;
- **On Change Var**, evento che si attiva quando la variabile specificata cambia valore;
- **On Var**, evento che si attiva quando la variabile specificata assume il valore specificato.

0.2.5.2 Azioni

Ad ogni *evento* è possibile associare più di una *azione*. Le *azioni* possibili sono:

- **Goto Page**, vai alla pagina specificata;
- **Next Page**, vai alla pagina successiva;
- **Previus Page**, vai alla pagina precedente;
- **Begin Data Entry**, attiva l'inserimento del dato;
- **Send Command**, invia un comando ad un device dichiarato nel progetto QView;
- **Set Variable**, assegna un valore ad una variabile del progetto QView;
- **Led On**, attiva il LED specificato (solitamente i LED sono associati ai tasti funzione);
- **Led Off**, disattiva il LED specificato;
- **Led Blink**, attiva il lampeggio del LED specificato;
- **Backup**, invia un comando di Backup (verificare se questo comando è supportato dall'hardware);
- **Restore**, invia un comando di Restore (verificare se questo comando è supportato dall'hardware).

0.2.6 Variabili di terminale

Ogni progetto QPaint contiene delle variabili sempre presenti che forniscono una serie di informazioni relative al terminale stesso. Queste variabili hanno già un nome prefissato, alcune sono in sola lettura e altre sono anche scrivibili. Facciamo un breve elenco delle variabili più importanti:

Nome	Tipo	Read/Write	Descrizione
\$KEY	L	R	Codice del tasto premuto. Ad ogni bit corrisponde un tasto della tastiera.
\$KEYF	L	R	Codice del tasto funzione premuto. Ad ogni bit corrisponde un tasto della tastiera.
\$KEYF2	L	R	Codice del tasto funzione premuto (Secondo gruppo). Ad ogni bit corrisponde un tasto della tastiera.
\$LED5	L	RW	Codice per attivare/disattivare i LED. Ad ogni bit corrisponde un LED.
\$LED52	L	RW	Codice per attivare/disattivare i LED (Secondo gruppo). Ad ogni bit corrisponde un LED.
\$HOUR, \$MIN, \$SEC, \$DAY, \$MONTH, \$YEAR	B, B, B, B, B, W	RW	Ora e data.
\$DATAENTRYON	F	R	Segnala che in questo momento è attivo l'inserimento di un valore
\$PAGE	L	R	Valore della pagina visualizzata in questo momento.

Ce ne sono molte altre ma queste sono le più utilizzate ed essenziali per poter sviluppare i progetti.

1. Controllo da QView

La gestione dell'interfaccia operatore può essere realizzata anche tramite uno scambio di informazioni con il progetto realizzato in QView. Le operazioni più importanti per la gestione di un'interfaccia operatore sono:

- rilevare la pressione di un tasto,
- capire che pagina è visualizzata in quel momento,
- comandare un cambio di una pagina,
- verificare se l'inserimento del dato (dataentry) è attivo (in questo caso non è possibile cambiare la pagina).

Spesso si preferisce gestire l'interfaccia operatore controllando direttamente le operazioni basilari dal progetto QView. Vediamo come questo sia possibile.

1.1 Rilevare la pressione di un tasto

Le variabili che permettono di conoscere se un tasto è premuto sono

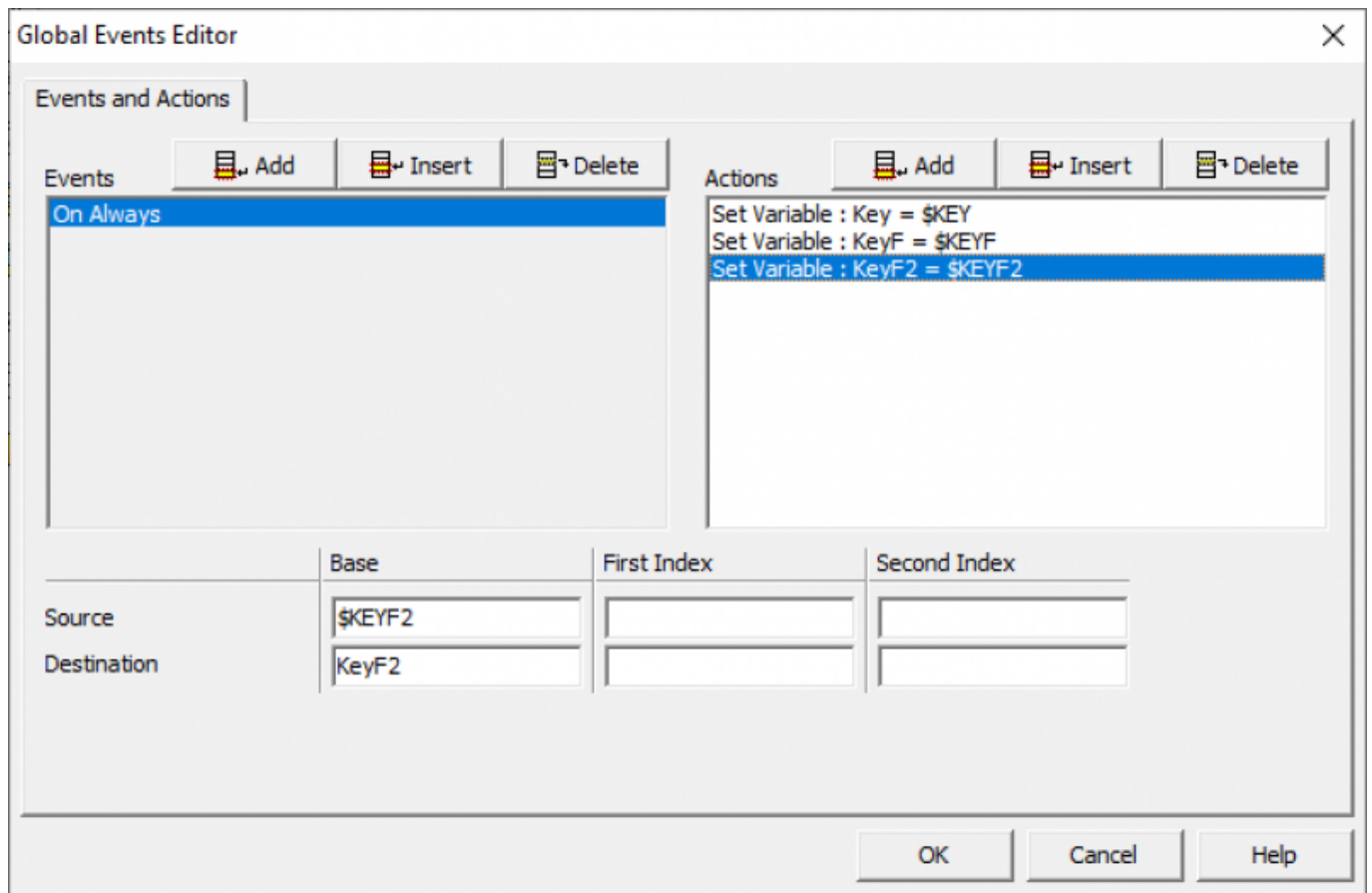
```
$KEY, $KEYF, $KEYF2.
```

Ogni bit di queste variabili è assegnato ad un tasto della tastiera. Queste variabili sono accessibili solo tramite QPaint e non sono visibili nel progetto QView. Per poter trasferire il valore di queste variabili al progetto QView è necessario prima di tutto predisporre tre variabili nel progetto QView:

```
GLOBAL
Key L ;This will be match to $KEY
KeyF L ;This will be match to $KEYF
KeyF2 L ;This will be match to $KEYF2
```

Nel progetto QPaint si seleziona:

- Project - Global Events Editor



Si deve inserire un evento *OnAlways* e associare ad esso tre azioni *SetVariable* come mostrato nella figura. In questo modo il progetto QPaint copierà continuamente (*OnAlways*) i valori delle sue tre variabili nelle variabili di QView. Nel progetto QView è possibile dichiarare le costanti relative alla pressione di ogni tasto:

```

CONST
KEY_1      268435456 ; "1"
KEY_2      1048576  ; "2"
KEY_3      4096     ; "3"
KEY_4      536870912 ; "4"
KEY_5      2097152  ; "5"
KEY_6      8192     ; "6"
KEY_7      1073741824 ; "7"
KEY_8      4194304  ; "8"
KEY_9      16384    ; "9"
KEY_0      8388608  ; "0"
KEY_CLR    -2147483648 ; "CLR"
KEY_ENTER  128      ; "ENTER"
KEY_HELP   64       ; "HELP"
KEY_DECP   32       ; "DECPT"
KEY_SIGN   16       ; "+/-"
KEY_ESC    33554432 ; "ESC"
KEY_UP     131072   ; "UP"
KEY_PGUP   512      ; "PGUP"
KEY_LEFT   67108864 ; "LEFT"
KEY_RIGHT  262144   ; "RIGHT"
KEY_NEXT   1024     ; "NEXT"
KEY_INS    134217728 ; "INS"
KEY_DOWN   524288   ; "DOWN"
KEY_PGDN   2048     ; "PGDN"

KEY_F1     33554432 ; "F1"
KEY_F2     67108864 ; "F2"
KEY_F3     134217728 ; "F3"
KEY_F4     268435456 ; "F4"
KEY_F5     536870912 ; "F5"
KEY_F6     131072   ; "F6"
KEY_F7     262144   ; "F7"
KEY_F8     524288   ; "F8"
KEY_F9     1048576  ; "F9"
KEY_F10    2097152  ; "F10"
KEY_F11    1        ; "F11"
KEY_F12    2        ; "F12"
KEY_F13    4        ; "F13"
KEY_F14    8        ; "F14"
KEY_F15    16       ; "F15"
KEY_F16    32       ; "F16"
KEY_F17    64       ; "F17"
KEY_F18    128      ; "F18"
KEY_F19    256      ; "F19"
KEY_F20    512      ; "F20"
KEY_F21    1024     ; "F21"
KEY_F22    2048     ; "F22"
KEY_F23    4096     ; "F23"
KEY_F24    8192     ; "F24"
KEY_F25    16384    ; "F25"
KEY_F26    32768    ; "F26"

```

Quindi il codice per eseguire una certa azione se viene premuto un tasto sarà

```
IF Key EQ KEY_1
;Put here the code to do when the operator press "1" key
ENDIF
```

1.2 Capire quale pagina è visualizzata

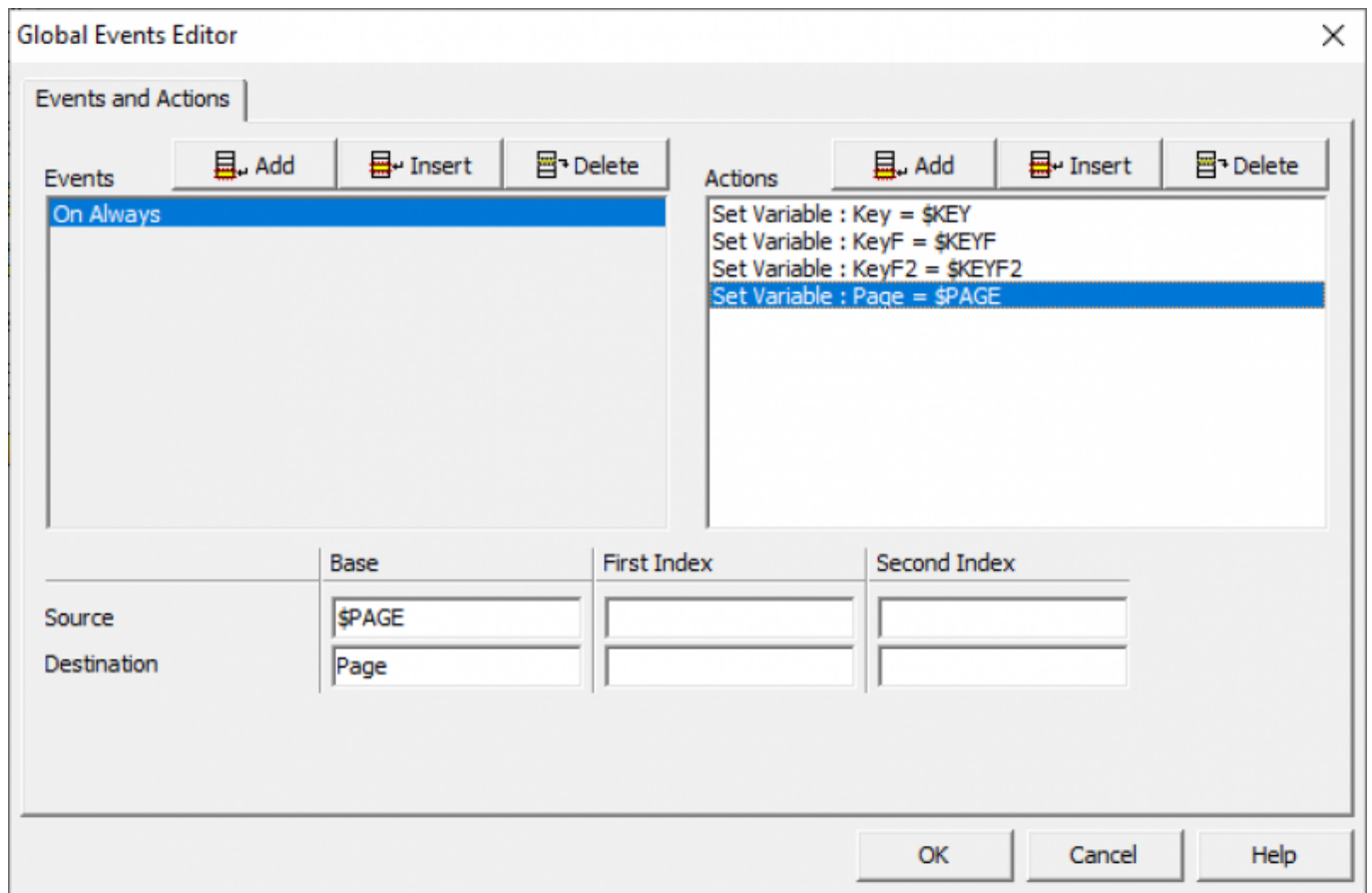
Analogamente al capitolo precedente, anche in questo caso devo predisporre una variabile nel progetto Qview per poter accogliere la variabile

```
$PAGE
```

del progetto QPaint. Quindi dichiaro:

```
GLOBAL
Page L ;This will be match to $PAGE
```

e poi aggiungo un'azione *SetVariable* associata all'evento *OnAlways*:



In questo modo posso eseguire operazioni diverse a seconda di quale pagina è visualizzata:

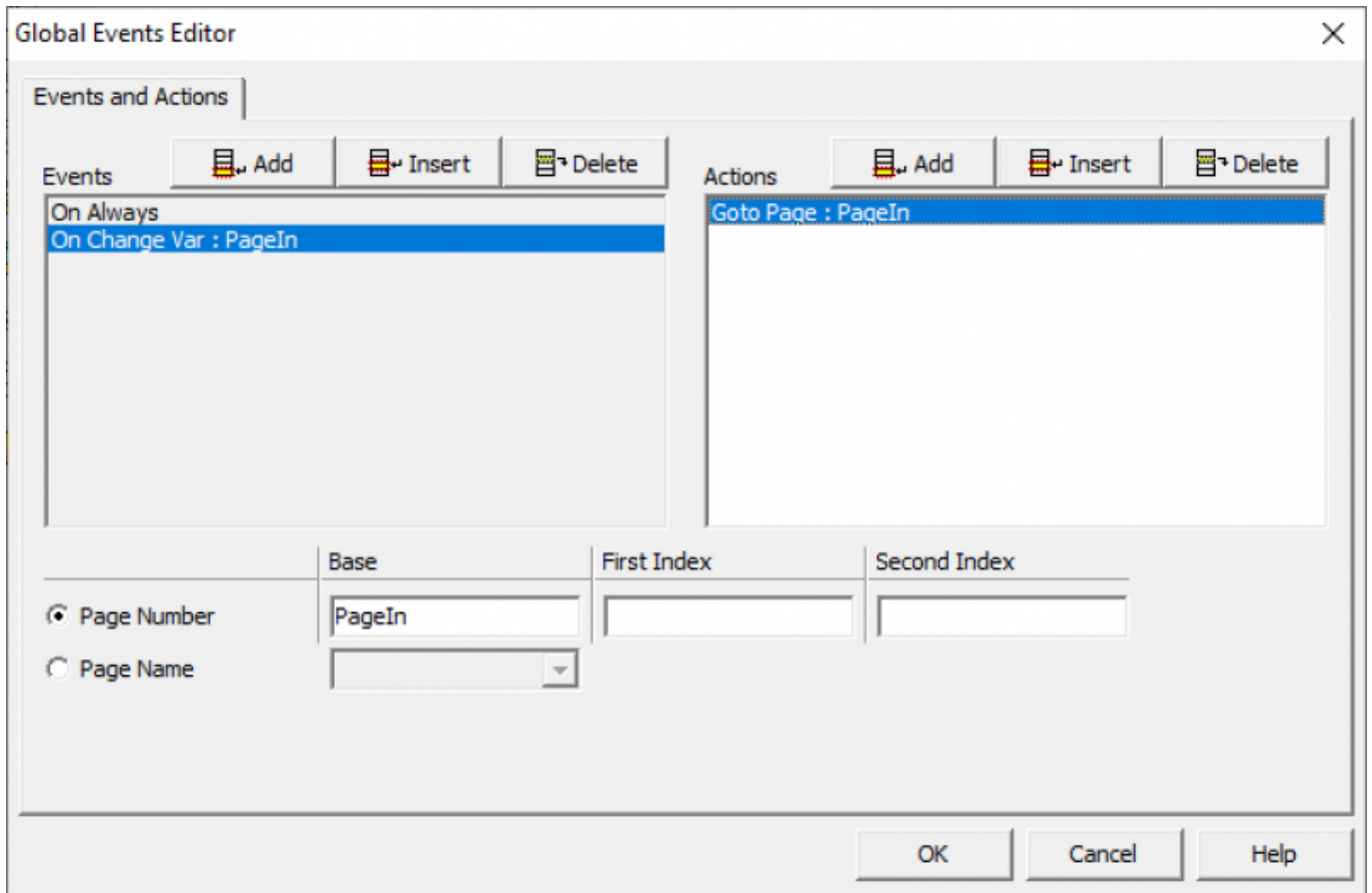
```
IF Page EQ MENU_PAGE
IF Key EQ KEY_1
;Put here the code to do when the operator press "1" key on the "MENU_PAGE" page
ENDIF
ENDIF
```

1.3 Comandare un cambio pagina

Per comandare un cambio pagina dell'interfaccia operatore tramite una sola riga di codice nel progetto QView, è necessario prima di tutto dichiarare una variabile QView :

```
GLOBAL
PageIn L ;This will be used to request the page change
```

è necessario poi aggiungere un evento *OnChangeVar* e associare ad esso un'azione *Goto Page* come in figura:



Quindi quando il valore della variabile *PageIn* viene cambiato in una riga di codice del progetto QView, viene attivata l'azione *Goto Page* che cambia la pagina e quindi viene visualizzata la pagina con indice il nuovo valore della variabile *PageIn*.

Il codice QCL sarà quindi:

```
IF Page EQ MENU PAGE
  IF Key EQ KEY_1
    PageIn = PAGE_1 ;Change page request: go to "PAGE_1" page
    WAIT(Page EQ PAGE_1) ;Wait the change page has done
  ENDF
ENDIF
```

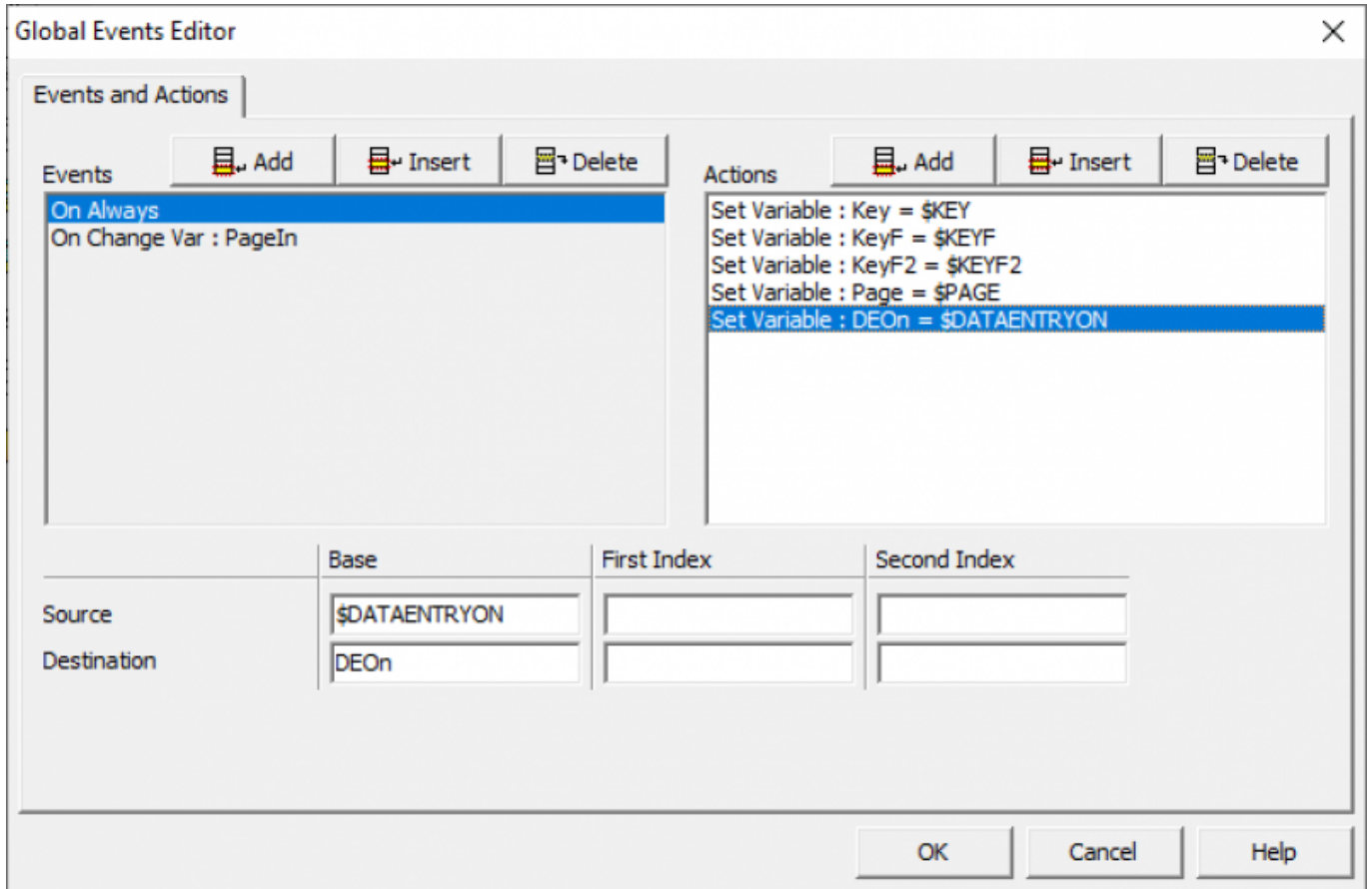
Come si può osservare, viene aggiunta una istruzione *WAIT* per attendere che effettivamente il cambio di pagina richiesto sia avvenuto prima di procedere con il resto delle linee di codice.

1.4 Dataentry attivo

Per conoscere lo stato dell'inserimento di un dato (Dataentry) si deve predisporre un'altra variabile nel progetto QVIEW:

```
GLOBAL
DEOn F ;This will match to $DATAENTRYON
```

aggiungere un ulteriore azione *SetVariable* associata all'evento *OnAlways*:



quindi se non posso cambiare pagine durante l'inserimento del dato, devo cambiare il codice in

```
IF Page EQ MENU_PAGE
  IF (Key EQ KEY_1) AND (DEOn EQ 0)
    PageIn = PAGE_1 ;Change page request: go to "PAGE_1" page
    WAIT (Page EQ PAGE_1) ;Wait the change page has done
  ENDF
ENDIF
```

2. Esempi

Forniamo degli esempi di programmazione del QPaint per le operazioni più diffuse. Spesso queste operazioni sono ottenute tramite la cooperazione tra la programmazione in QPaint e alcune righe di codice scritte in QView.

- Pulsante grafico.

2.1 Pulsante grafico

Un pulsante grafico è una immagine a forma di bottone che una volta toccata modifica la sua forma per dare l'impressione di



essere premuta. Per la costruzione di un pulsante grafico sono necessarie due immagini (tif, gif, jpg, pcx, bmp, ico):

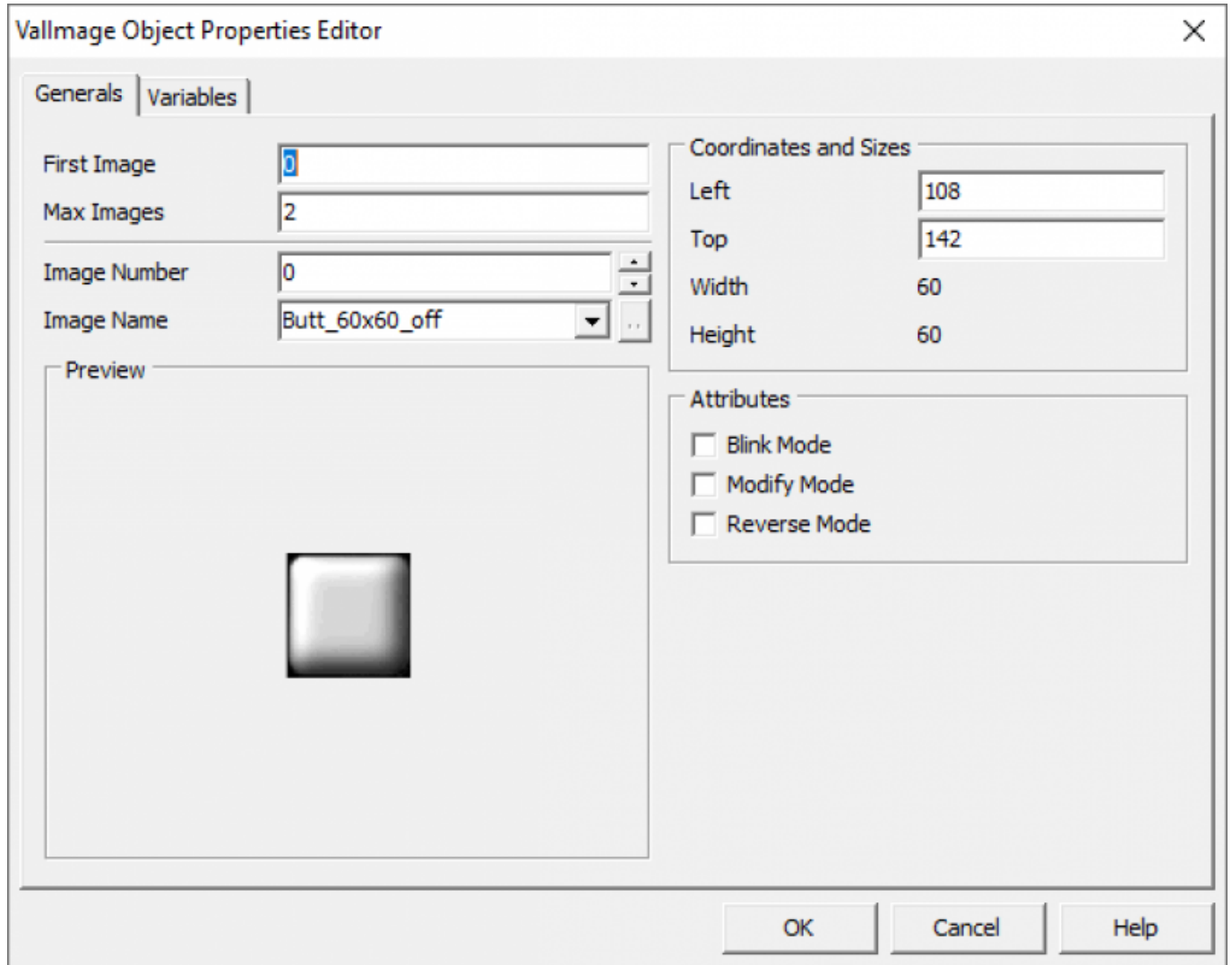
Queste immagini dovranno essere aggiunte al *Image Manager* del progetto QPaint 5 selezionando

- Grafica - Gestore delle immagini (Graphics - Image Manager)

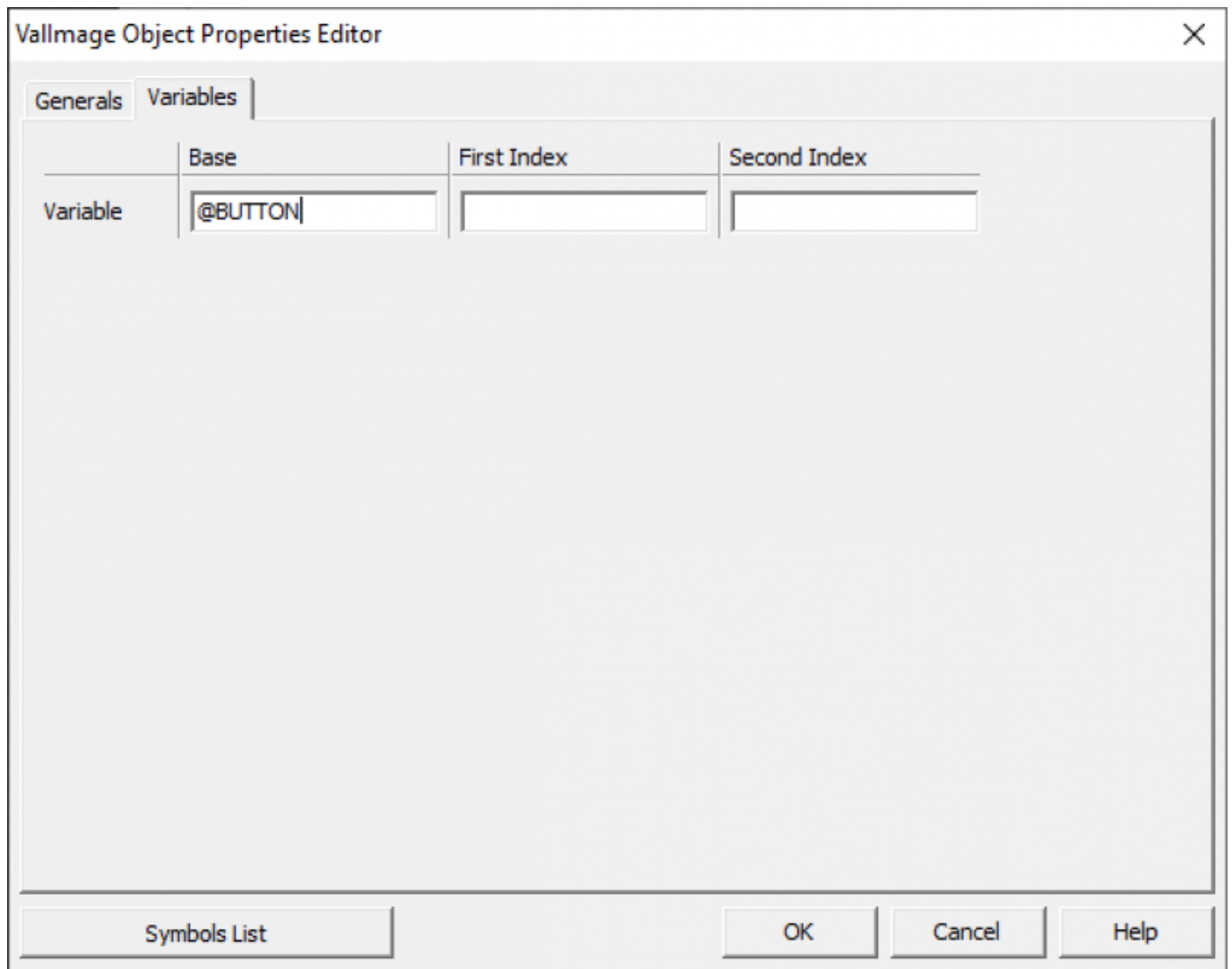
Per creare il bottone è necessario utilizzare due oggetti:

- un *VallImage* object e
- un *TouchArea* object.

Selezionare il *VallImage* e posizionare sulla pagina del progetto:



inserire come *Prima Immagine (First Image)* il valore 0 e come *Numero Immagini (Max Images)* il valore 2. Associare, quindi al valore 0 l'immagine *butt_60x60_off* e al valore 1 l'immagine *butt_60x60_on*. È necessario quindi associare a questo oggetto una variabile del terminale:

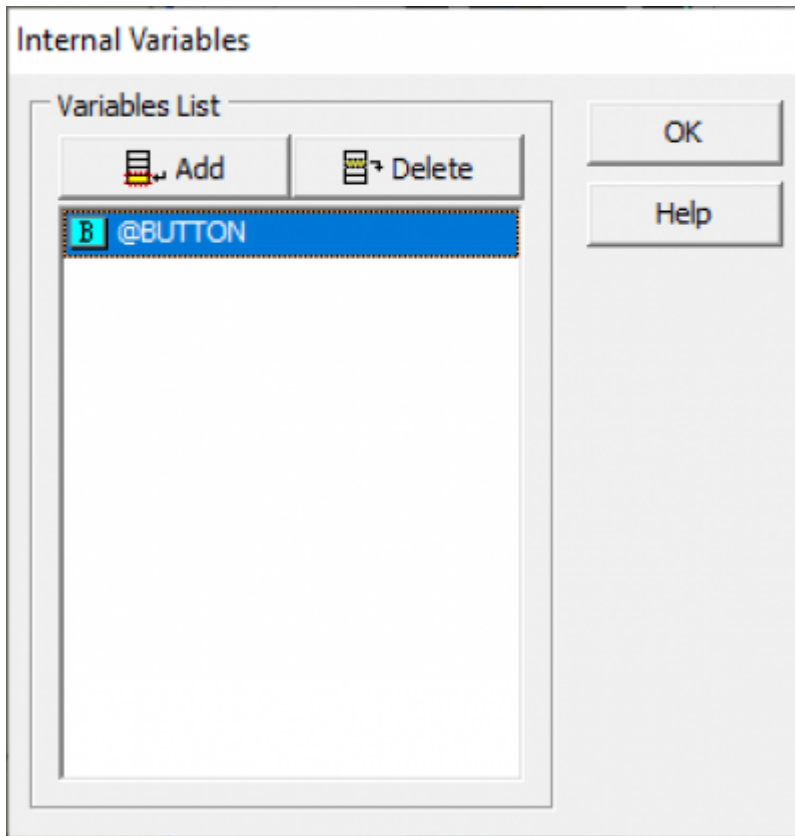


The image shows a dialog box titled "Vallimage Object Properties Editor" with a close button (X) in the top right corner. It has two tabs: "Generals" and "Variables". The "Variables" tab is selected. Inside the "Variables" tab, there is a table with four columns: "Variable", "Base", "First Index", and "Second Index". The "Variable" column contains the text "@BUTTON". The "Base", "First Index", and "Second Index" columns are empty. At the bottom of the dialog, there are four buttons: "Symbols List", "OK", "Cancel", and "Help".

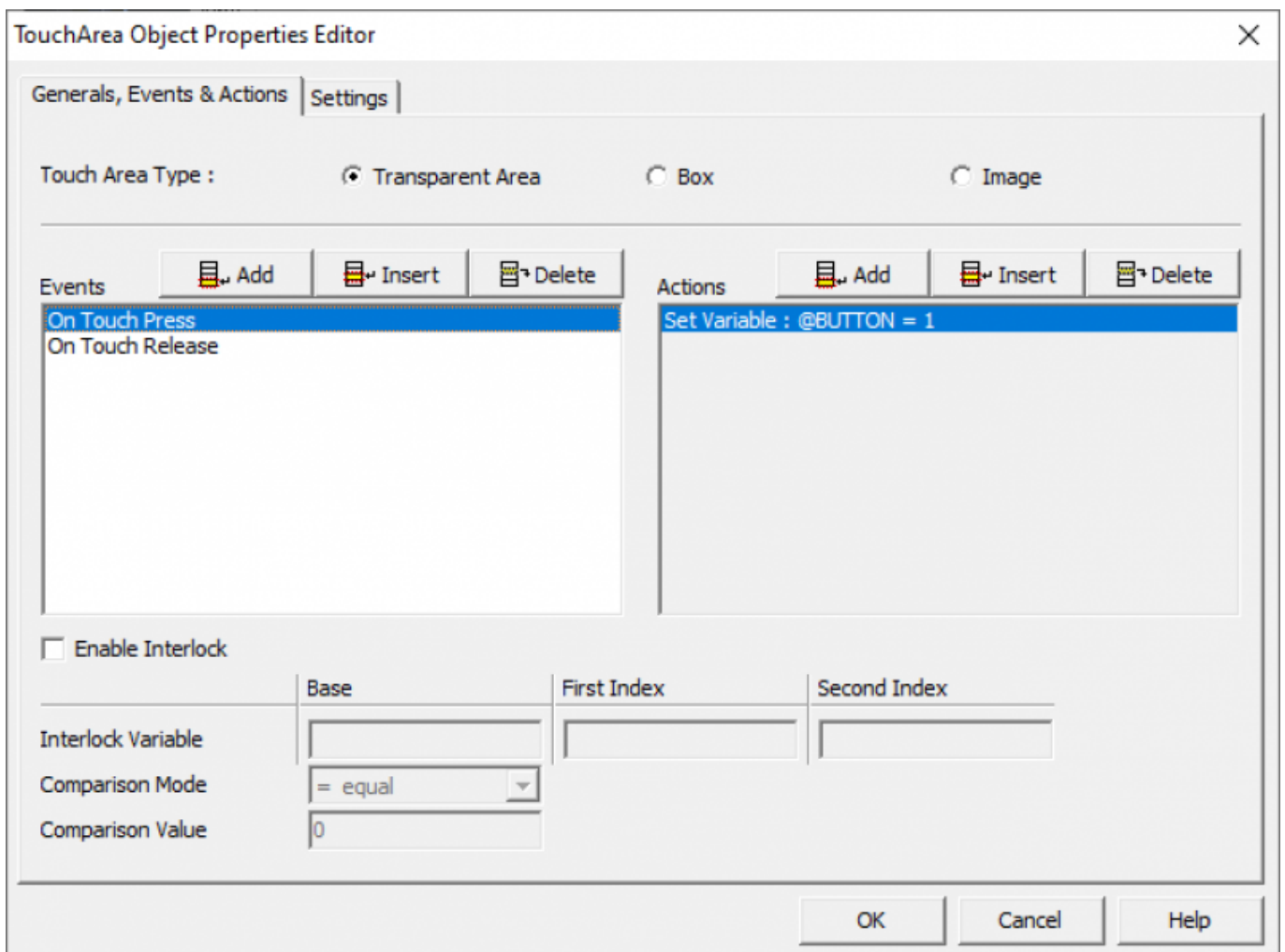
Variable	Base	First Index	Second Index
@BUTTON			

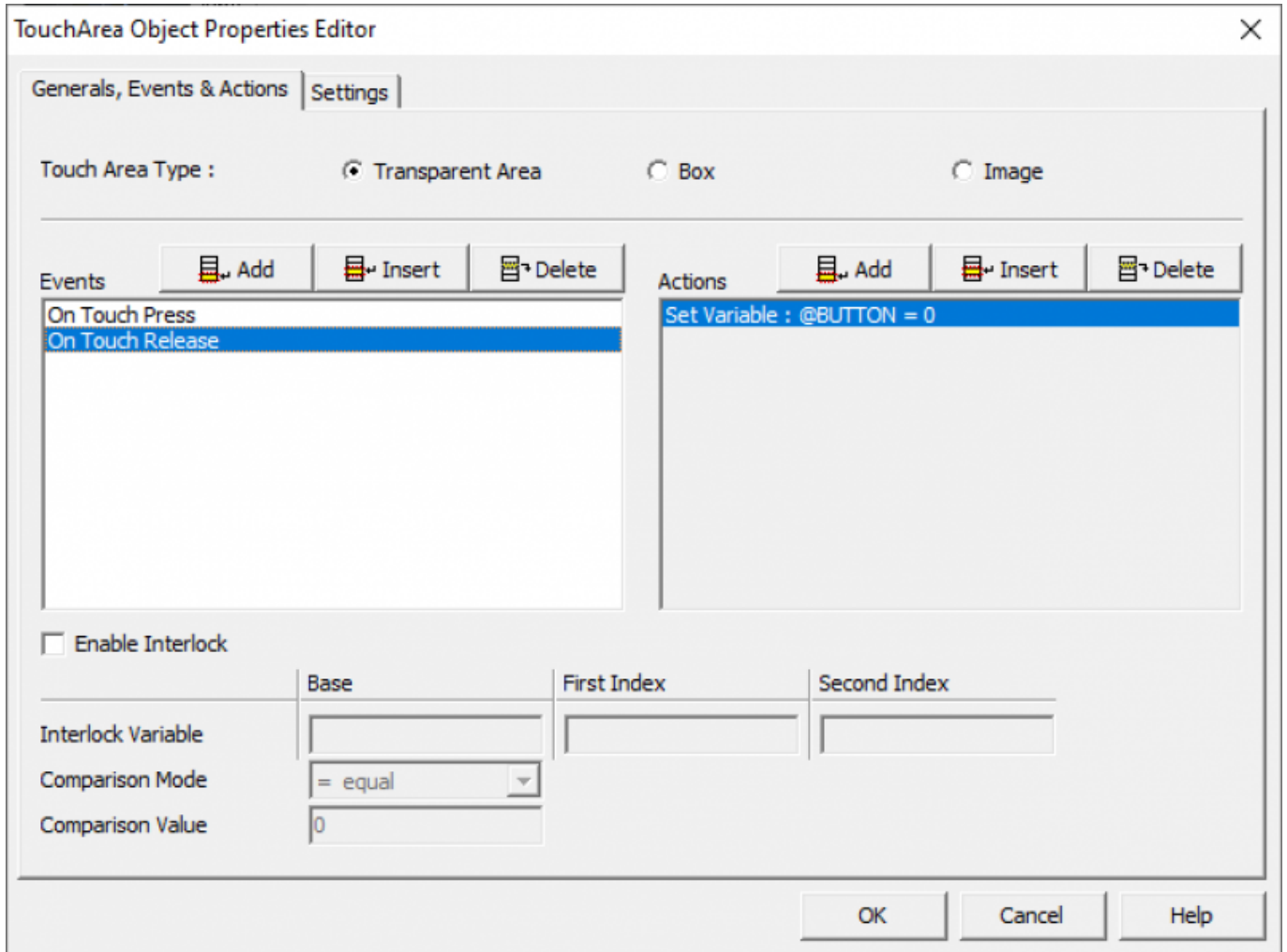
La variabile *@BUTTON* deve essere dichiarata in

- Progetto - Variabili Interne (Project - Internal variables)



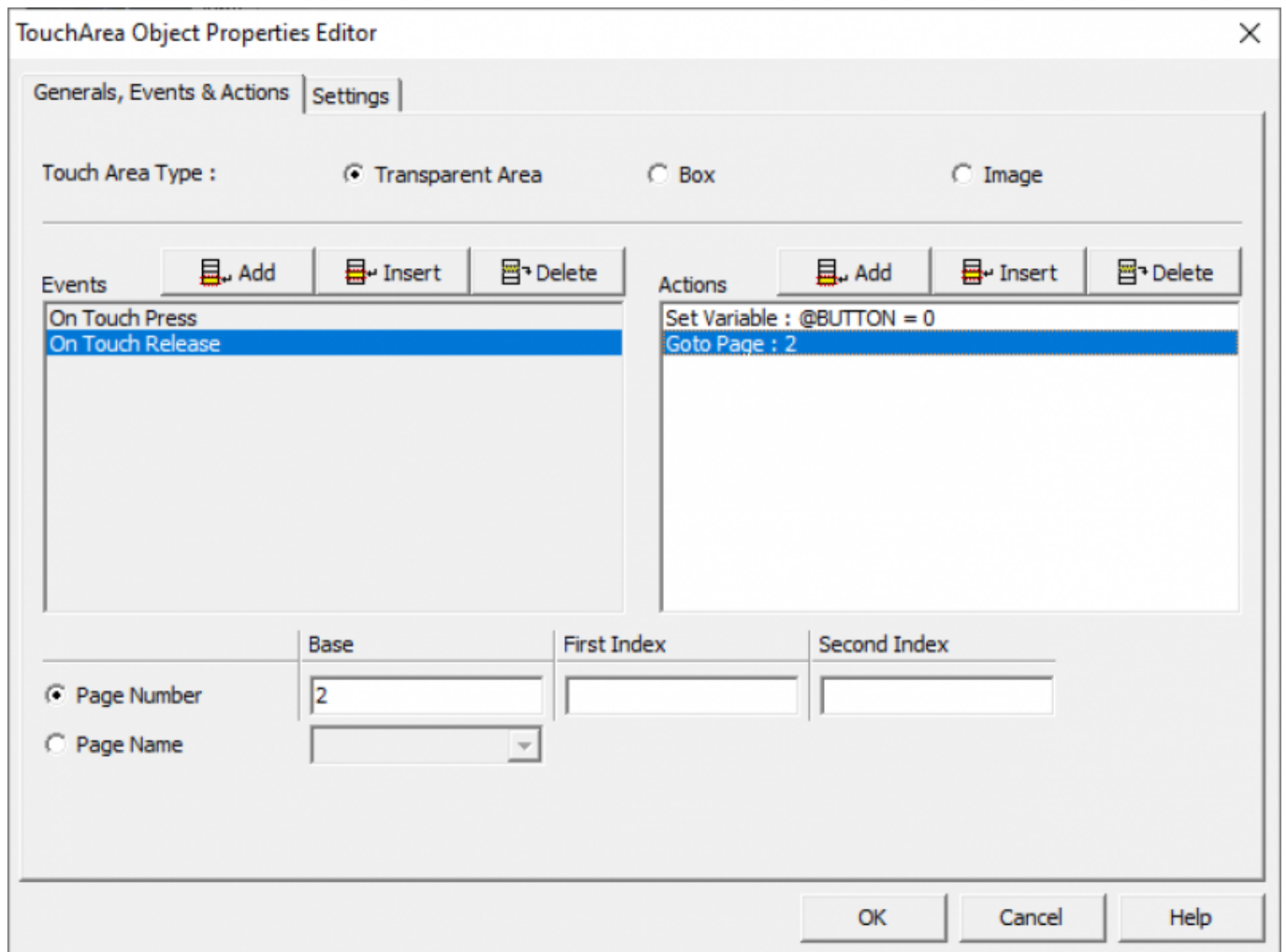
A questo punto è necessario aggiungere l'oggetto TouchArea sopra il VallImage e programmare il tocco e il rilascio in questo modo:



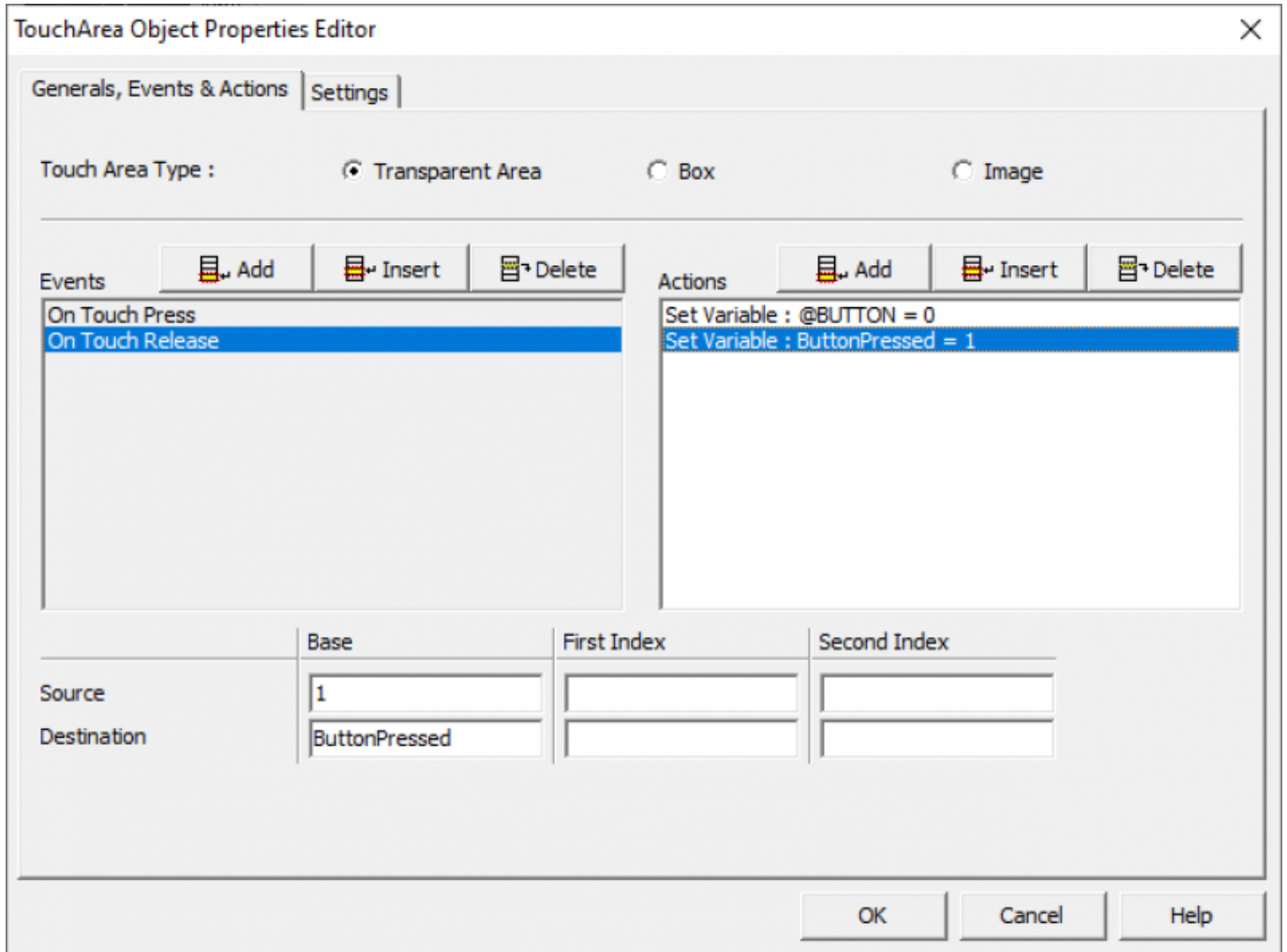


Quindi alla pressione dell'area di tocco viene assegnato il valore 1 alla variabile `@BUTTON` e quindi l'immagine cambierà in `butt_60x60_on`. Al rilascio dell'area di tocco viene assegnato 0 a `@BUTTON` e quindi l'immagine ritorna a `butt_60x60_off`.

A questo punto il pulsante grafico è fatto. Per l'utilizzo di questo pulsante è necessario associare alla pressione e/o al rilascio una azione aggiuntiva che permetta di utilizzare il bottone. Per esempio:



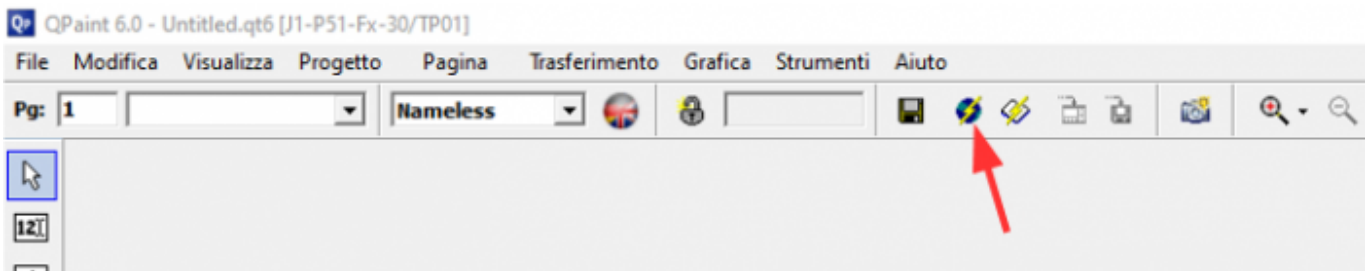
si vede che al rilascio del bottone, viene comandato il cambio pagina con l'azione *Go to page*. Altro utilizzo potrebbe essere quello di assegnare un valore ad una variabile di un progetto QView. Tale variabile poi potrà essere utilizzata nel progetto QView per eseguire altre operazioni.



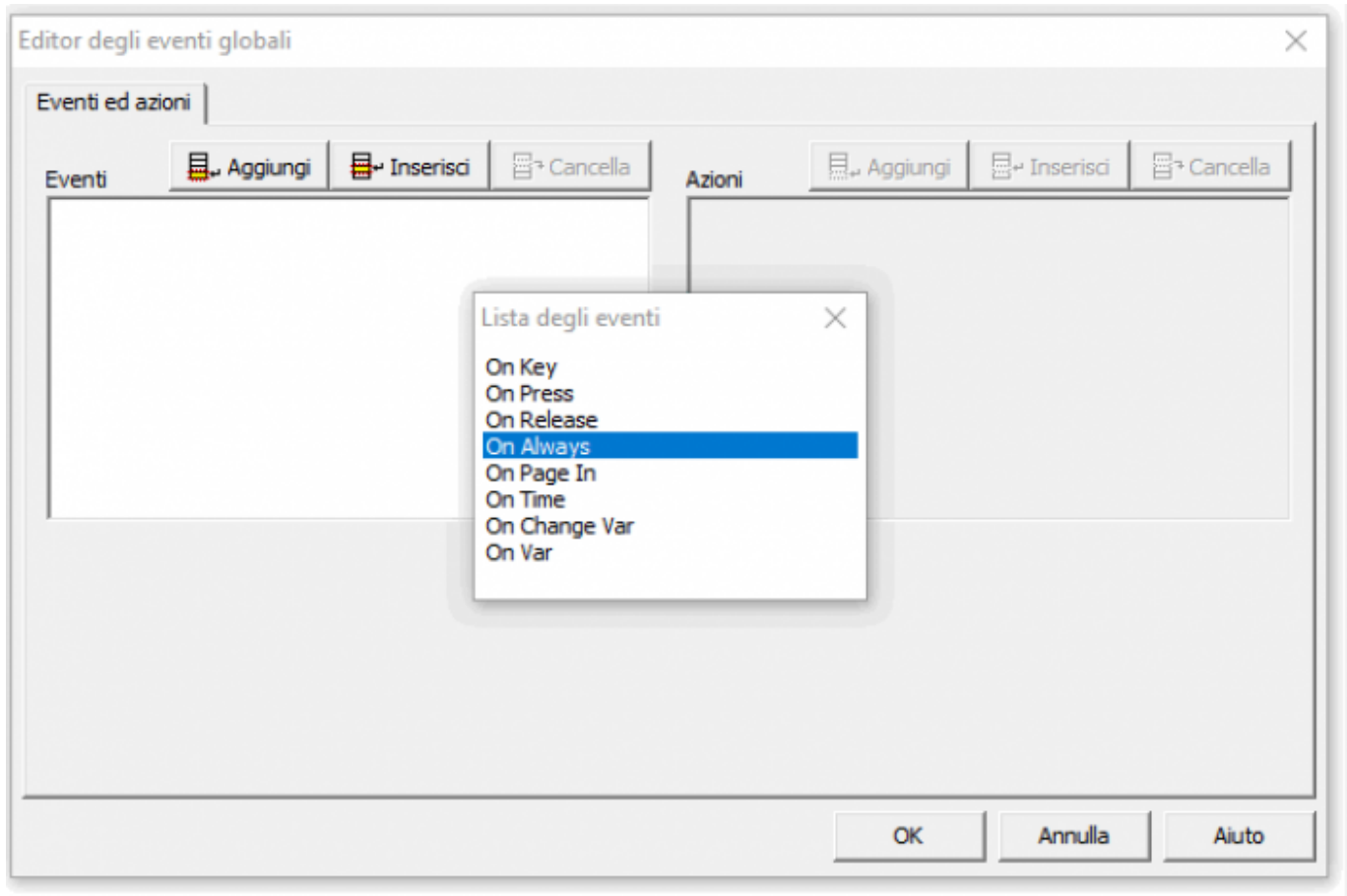
2.2 Gestire gli I/O nel terminale

Per gestire gli I/O nel terminale, è necessario aggiungere due azioni legate ad un evento globale “On Always” che copiano in continuazione i dati degli I/O nelle variabili QCL di appoggio.

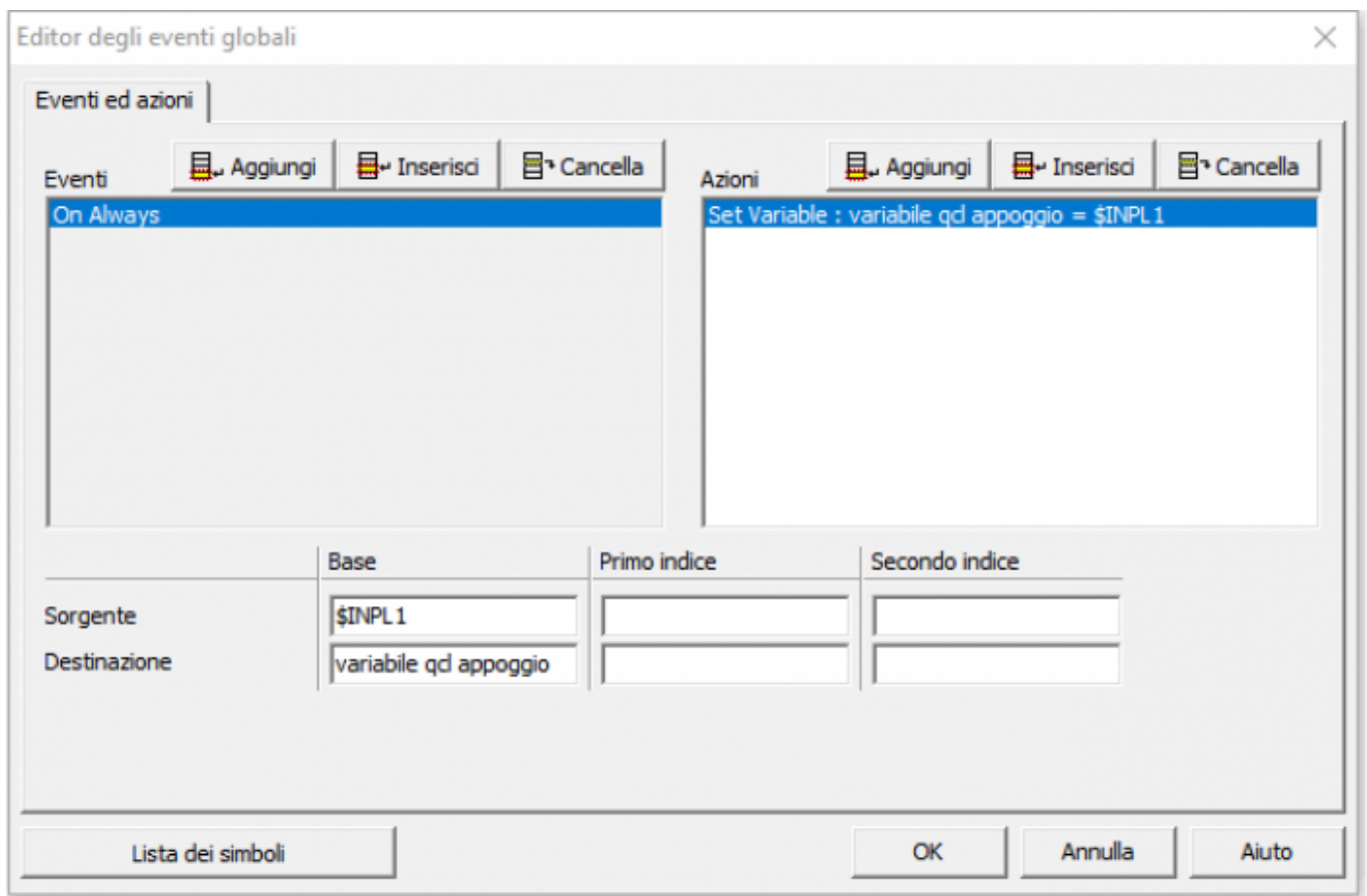
- Aprire il pannello *Eventi globali*



- Aggiungere un evento **OnAlways**

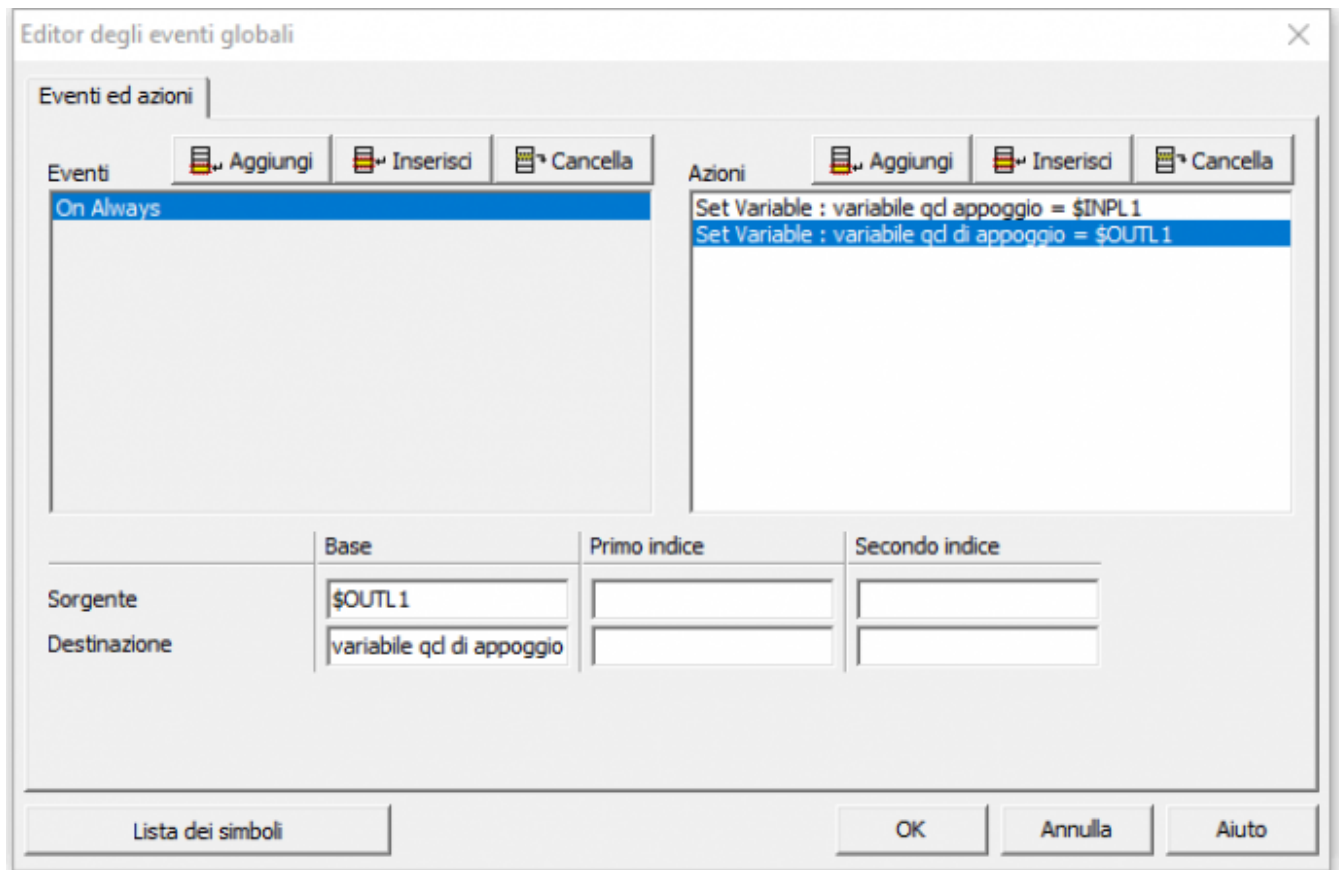


- Aggiungere un'azione **set variable** che copia la variabile `$INPL1` in una variabile QCL scelta. `$INPL1` rappresenta i valori binari dei primi 32 ingressi del terminale.



- Aggiungere un evento **set variable** che copia il valore di una variabile di appoggio QCL, nella variabile di

terminale \$OUTL1



2.3 Paradigmi di programmazione

Gli strumenti QEM si possono raggruppare in due macro categorie:

1. dispositivi non integrati (dispositivi PLC "Retroquadro" + dispositivi HMI)
2. dispositivi **integrati** (PLC + HMI)

I dispositivi integrati sono composti dal PLC ([Programmable Logic Controller](#)) e dall'HMI ([Human Machine Interface](#)). Vengono chiamati "integrati" perché solitamente PLC e HMI sono due dispositivi separati connessi tra loro con uno dei protocolli di comunicazione supportati.

Per ognuna di queste categorie c'è una paradigma di programmazione da seguire (la prima categoria può essere comunque programmata con entrambi i paradigmi):

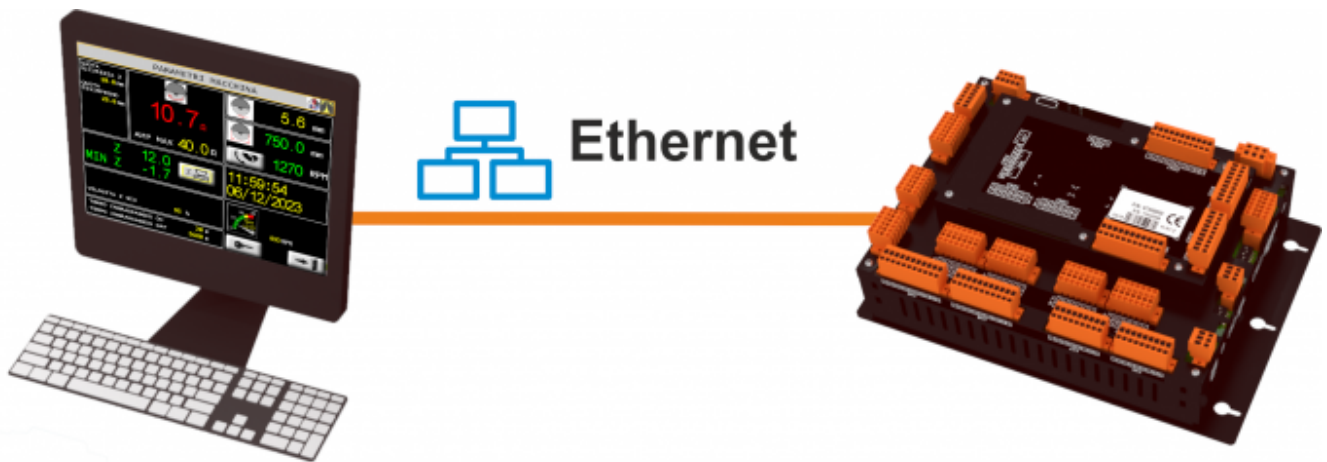
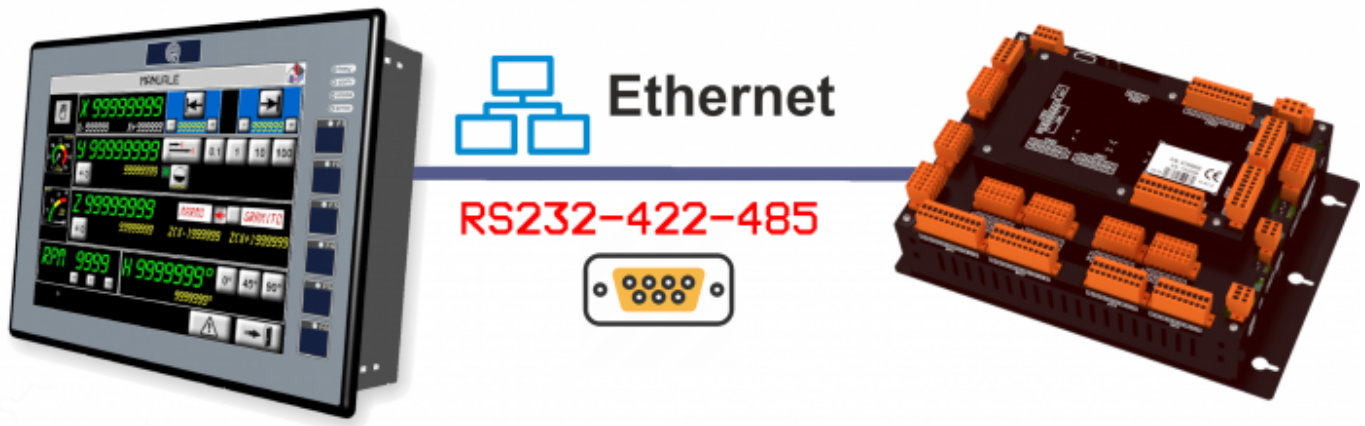
1. dispositivi non integrati ➡ senza l'uso del device MMIQ2 [sezione 2.3.1](#)
2. dispositivi integrati ➡ uso del device MMIQ2 [sezione 2.3.2](#)

Questi paradigmi di programmazione sono riferiti a come il software del PLC (software QCL) si interfaccia al software dell'HMI (display + touch).

I dispositivi integrati possono utilizzare il device [MMIQ2](#) per comandare i cambi pagina ed ottenere i valori come la pagina corrente.

2.3.1 Paradigmi di programmazione: senza device MMIQ2

Questo modo di programmare è quello in cui la logica del software QCL non dispone del device MMIQ2 per l'accesso al dispositivo HMI. Si usa con i dispositivi PLC non integrati sia HMI che PC che svolgono il ruolo di HMI come il nostro software [QPaint-Runtime-1](#).



2.3.2 Paradigmi di programmazione: con device MMIQ2

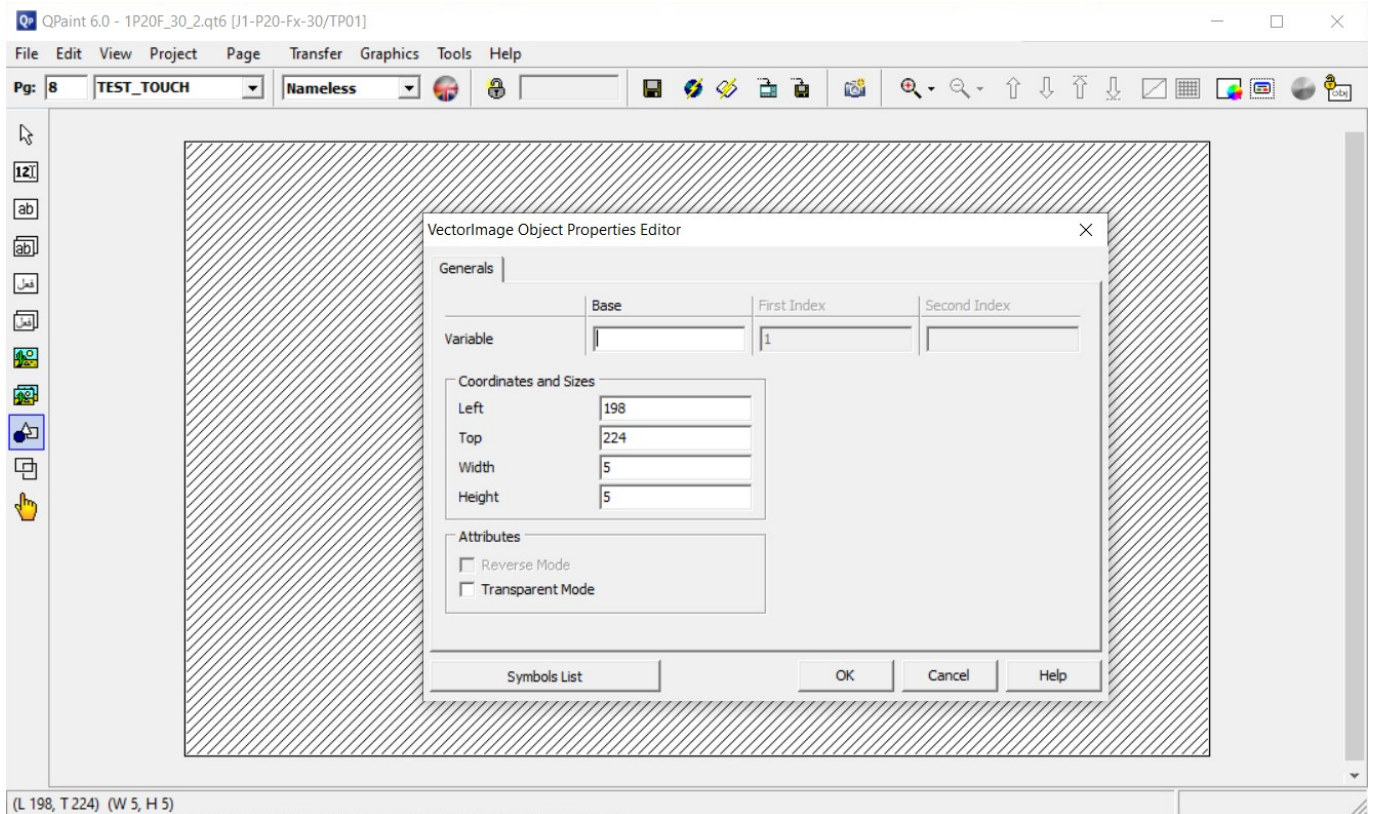


(spiegare che è il caso degli strumenti “integrati” → PLC+HMI)

3. Guide

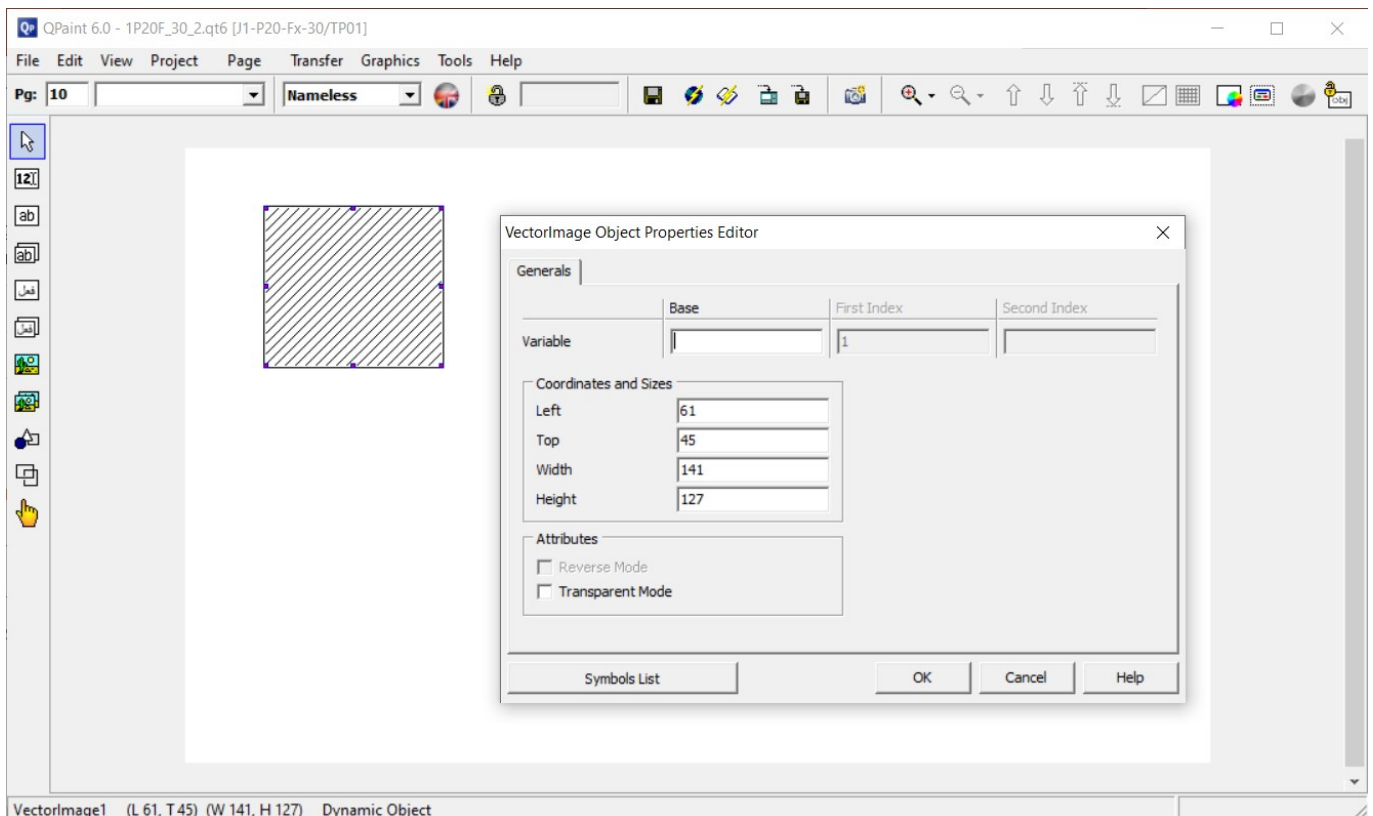
Vector image

Il Vector Image è un oggetto di tipo dinamico di QPaint, serve a disegnare sullo schermo figure piane.



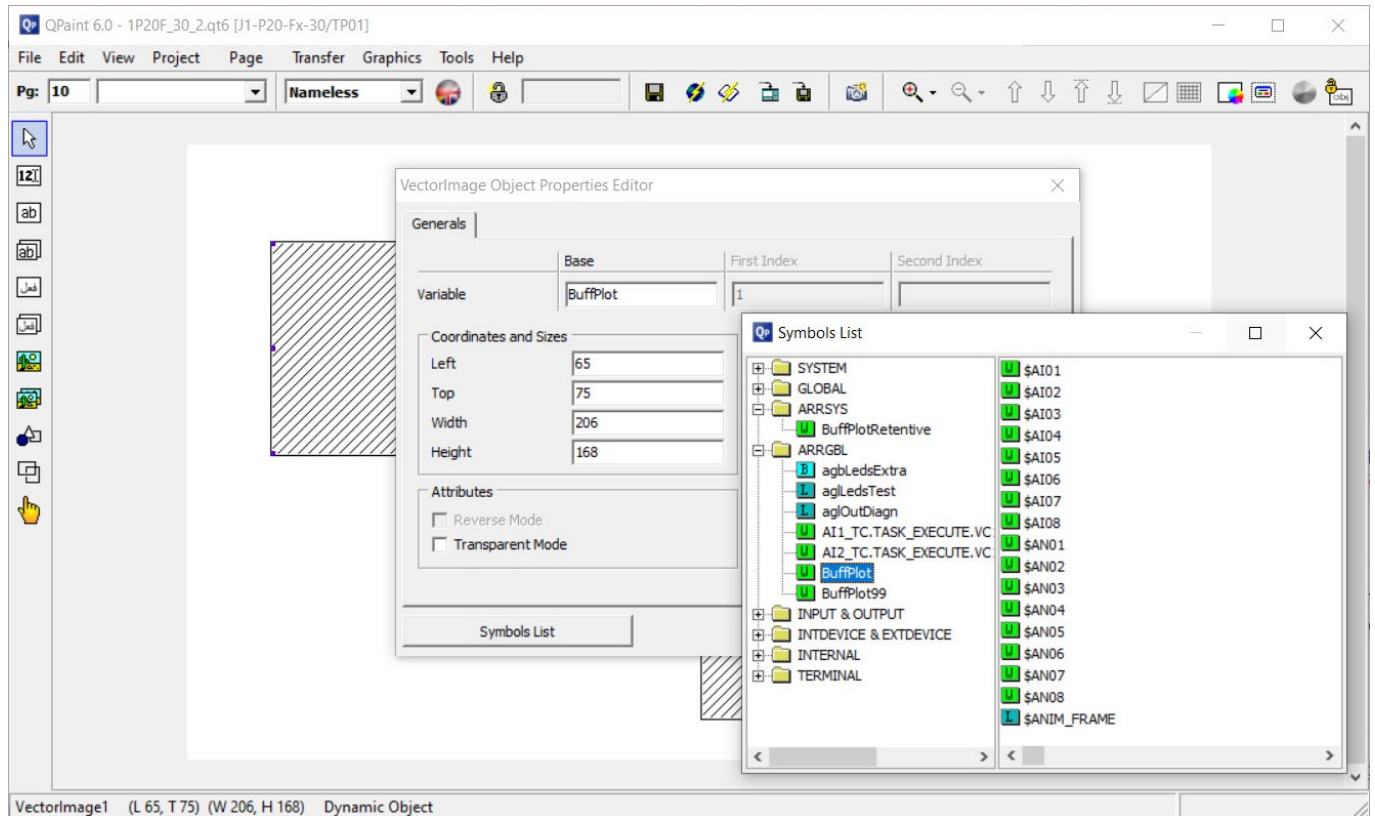
Utilizzo

Dal pannello a sinistra selezionare l'icona dell'oggetto Vector Image , tracciare sull'area dello schermo l'oggetto delle dimensioni desiderate (sarà possibile modificare le dimensioni e la posizione anche successivamente). Al rilascio apparirà l'oggetto e si aprirà il pannello delle proprietà.



Nel campo variabile occorre selezionare un array di tipo W (word) di dimensione da 100 a 65535 elementi definito in precedenza nel programma QView, per fare ciò cliccare su Symbols List e selezionare il simbolo (la variabile) corretta dal

ramo ARRGBL o nel caso di variabile ritentiva dal ramo ARRSYS.



Creazione plot

Per procedere al disegno occorre utilizzare alcune delle [funzioni QCL apposite](#). Tramite queste funzioni si eseguono le azioni di inizializzazione del buffer, settaggio del colore di background e disegno.

Di seguito un esempio:

```
;definizione costanti
CONST
BLACK          000
WHITE          001
BLUE           002
GREEN          003
CYAN           004
RED            005
MAGENTA       006
YELLOW        007

;definizione variabili
ARRGBL
BuffPlot      W    100

;svolgimento
BEGIN
VI10ClrErrorCode(BuffPlot)      ;Cancella eventuali errori segnalati
VI10InitBuffer(BuffPlot)        ;Inizializza il Buffer
VI10SetLayer(BuffPlot, 0)       ;Imposta il layer - zero è il layer di background
VI10SetBackground(BuffPlot, BLACK) ;Imposta il colore del background
VI10AddCls(BuffPlot)            ;Cancella l'area del Vector Image
VI10AddPen(BuffPlot, CYAN)      ;Imposta il colore del disegno
VI10DrawBuffer(BuffPlot)        ;Disegna il contenuto del buffer
END
```



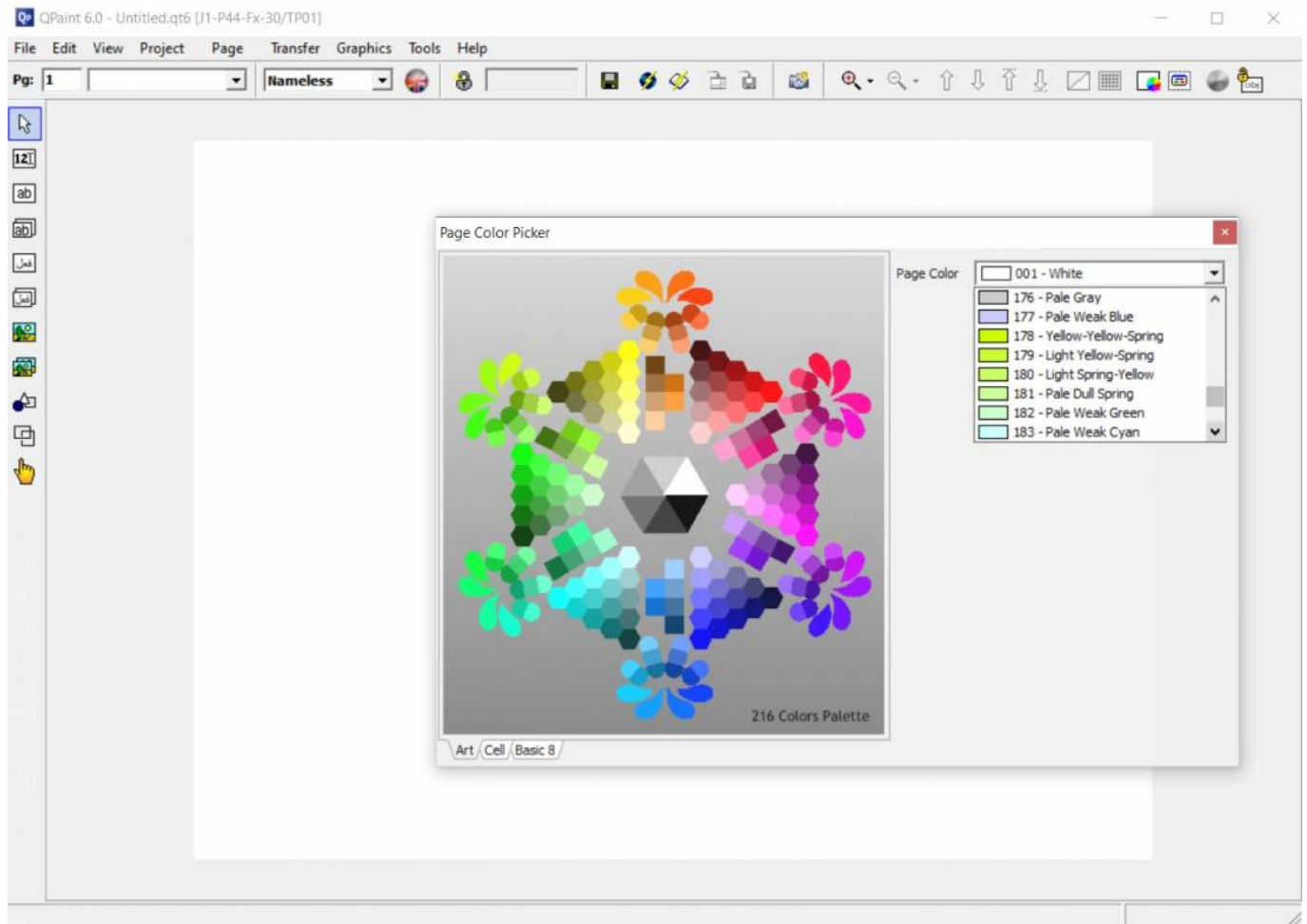
Da QView si può evidenziare il nome di una funzione di QCL e premere il tasto F1 per visualizzare la guida.

Colori

L'oggetto Vector Image gestisce 256 colori (da 0 a 255). Per vedere la tabella dei colori procedere come segue:

Cliccare il pulsante  Page Color, da qui si può scorrere il menù a tendina oppure cliccare su un colore per vederne il

codice.



Di seguito una UNIT QCL con all'interno tutte le costanti per gestire i colori.

COLOR.MOD

```

;===
; color constants support UNIT
;
CONST
;Colori per Vector Image
BLACK      0      OUT ; Black
WHITE      1      OUT ; White
BLUE       2      OUT ; Blue
GREEN      3      OUT ; Green
CYAN       4      OUT ; Cyan
RED        5      OUT ; Red
MAGENTA    6      OUT ; Magenta
YELLOW     7      OUT ; Yellow
OWB        8      OUT ; Obscure Weak Blue
ODB        9      OUT ; Obscure Dull Blue
DFB       10      OUT ; Dark Faded Blue
DHB       11      OUT ; Dark Hard Blue
OWG       12      OUT ; Obscure Weak Green
OWC       13      OUT ; Obscure Weak Cyan
ODA       14      OUT ; Obscure Dull Azure
DAB       15      OUT ; Dark Azure-Blue
DBA       16      OUT ; Dark Blue-Azure
BBA       17      OUT ; Blue-Blue-Azure
ODG       18      OUT ; Obscure Dull Green
ODT       19      OUT ; Obscure Dull Teal
ODC       20      OUT ; Obscure Dull Cyan
DAC       21      OUT ; Dark Azure-Cyan
DHA       22      OUT ; Dark Hard Azure
AAB       23      OUT ; Azure-Azure-Blue
DFG       24      OUT ; Dark Faded Green
DTG       25      OUT ; Dark Teal-Green
DTC       26      OUT ; Dark Teal-Cyan
DFC       27      OUT ; Dark Faded Cyan
DCA       28      OUT ; Dark Cyan-Azure
AAC       29      OUT ; Azure-Azure-Cyan
DHG       30      OUT ; Dark Hard Green
DGT       31      OUT ; Dark Green-Teal
DHT       32      OUT ; Dark Hard Teal
DCT       33      OUT ; Dark Cyan-Teal
DHC       34      OUT ; Dark Hard Cyan
CCA       35      OUT ; Cyan-Cyan-Azure
GGT       36      OUT ; Green-Green-Teal
TTG       37      OUT ; Teal-Teal-Green
TTC       38      OUT ; Teal-Teal-Cyan

```

CCT	39	OUT	Cyan-Cyan-Teal
OWR	40	OUT	Obscure Weak Red
OWM	41	OUT	Obscure Weak Magenta
ODV	42	OUT	Obscure Dull Violet
DVB	43	OUT	Dark Violet-Blue
DBV	44	OUT	Dark Blue-Violet
BBV	45	OUT	Blue-Blue-Violet
OWY	46	OUT	Obscure Weak Yellow
OG	47	OUT	Obscure Gray
DWB	48	OUT	Dark Weak Blue
DOB	49	OUT	Dark Dull Blue
MFB	50	OUT	Medium Faded Blue
LHB	51	OUT	Light Hard Blue
ODS	52	OUT	Obscure Dull Spring
DWG	53	OUT	Dark Weak Green
DWC	54	OUT	Dark Weak Cyan
DDA	55	OUT	Dark Dull Azure
MAB	56	OUT	Medium Azure-Blue
LBA	57	OUT	Light Blue-Azure
DSG	58	OUT	Dark Spring-Green
DDG	59	OUT	Dark Dull Green
DDT	60	OUT	Dark Dull Teal
DDC	61	OUT	Dark Dull Cyan
MAC	62	OUT	Medium Azure-Cyan
LHA	63	OUT	Light Hard Azure
DGS	64	OUT	Dark Green-Spring
MFG	65	OUT	Medium Faded Green
MTG	66	OUT	Medium Teal-Green
MTC	67	OUT	Medium Teal-Cyan
MFC	68	OUT	Medium Faded Cyan
LCA	69	OUT	Light Cyan-Azure
GGS	70	OUT	Green-Green-Spring
LHG	71	OUT	Light Hard Green
LGT	72	OUT	Light Green-Teal
LHT	73	OUT	Light Hard Teal
LCT	74	OUT	Light Cyan-Teal
LHC	75	OUT	Light Hard Cyan
ODR	76	OUT	Obscure Dull Red
ODP	77	OUT	Obscure Dull Pink
ODM	78	OUT	Obscure Dull Magenta
DVM	79	OUT	Dark Violet-Magenta
DHV	80	OUT	Dark Hard Violet
VVB	81	OUT	Violet-Violet-Blue
ODO	82	OUT	Obscure Dull Orange
DWR	83	OUT	Dark Weak Red
DWM	84	OUT	Dark Weak Magenta
DDV	85	OUT	Dark Dull Violet
MVB	86	OUT	Medium Violet-Blue
LBV	87	OUT	Light Blue-Violet
ODY	88	OUT	Obscure Dull Yellow
DWY	89	OUT	Dark Weak Yellow
DG	90	OUT	Dark Gray
MWB	91	OUT	Medium Weak Blue
LDB	92	OUT	Light Dull Blue
LFB	93	OUT	Light Faded Blue
DSY	94	OUT	Dark Spring-Yellow
DDS	95	OUT	Dark Dull Spring
MWG	96	OUT	Medium Weak Green
MWC	97	OUT	Medium Weak Cyan
LDA	98	OUT	Light Dull Azure
LAB	99	OUT	Light Azure-Blue
DHS	100	OUT	Dark Hard Spring
MSG	101	OUT	Medium Spring-Green
LDG	102	OUT	Light Dull Green
LDT	103	OUT	Light Dull Teal
LDC	104	OUT	Light Dull Cyan
LAC	105	OUT	Light Azure-Cyan
SSG	106	OUT	Spring-Spring-Green
LGS	107	OUT	Light Green-Spring
LFG	108	OUT	Light Faded Green
LTG	109	OUT	Light Teal-Green
LTC	110	OUT	Light Teal-Cyan
LFC	111	OUT	Light Faded Cyan
DFR	112	OUT	Dark Faded Red
DPR	113	OUT	Dark Pink-Red
DPM	114	OUT	Dark Pink-Magenta
DFM	115	OUT	Dark Faded Magenta
DMV	116	OUT	Dark Magenta-Violet
VVM	117	OUT	Violet-Violet-Magenta
DOR	118	OUT	Dark Orange-Red
DDR	119	OUT	Dark Dull Red
DDP	120	OUT	Dark Dull Pink
DDM	121	OUT	Dark Dull Magenta
MVM	122	OUT	Medium Violet-Magenta
LHV	123	OUT	Light Hard Violet
DOY	124	OUT	Dark Orange-Yellow
DDO	125	OUT	Dark Dull Orange
MWR	126	OUT	Medium Weak Red
MWM	127	OUT	Medium Weak Magenta
LDV	128	OUT	Light Dull Violet
LBV	129	OUT	Light Violet-Blue
DFY	130	OUT	Dark Faded Yellow
DDY	131	OUT	Dark Dull Yellow
MWY	132	OUT	Medium Weak Yellow
LG	133	OUT	Light Gray
LWB	134	OUT	Light Weak Blue
PDB	135	OUT	Pale Dull Blue
DYS	136	OUT	Dark Yellow-Spring
MSY	137	OUT	Medium Spring-Yellow
LDS	138	OUT	Light Dull Spring
LWG	139	OUT	Light Weak Green
LWC	140	OUT	Light Weak Cyan
PDA	141	OUT	Pale Dull Azure
SSY	142	OUT	Spring-Spring-Yellow
LHS	143	OUT	Light Hard Spring
LSG	144	OUT	Light Spring-Green
PDG	145	OUT	Pale Dull Green
PDT	146	OUT	Pale Dull Teal
PDC	147	OUT	Pale Dull Cyan
DHR	148	OUT	Dark Hard Red
DRP	149	OUT	Dark Red-Pink
DHP	150	OUT	Dark Hard Pink
DMP	151	OUT	Dark Magenta-Pink
DHM	152	OUT	Dark Hard Magenta
MMV	153	OUT	Magenta-Magenta-Violet
DRO	154	OUT	Dark Red-Orange
MFR	155	OUT	Medium Faded Red
MPR	156	OUT	Medium Pink-Red
MPM	157	OUT	Medium Pink-Magenta
MFV	158	OUT	Medium Faded Magenta
LMV	159	OUT	Light Magenta-Violet
DHO	160	OUT	Dark Hard Orange
MOR	161	OUT	Medium Orange-Red
LDR	162	OUT	Light Dull Red
LDP	163	OUT	Light Dull Pink
LDM	164	OUT	Light Dull Magenta

LVM	165	OUT	;	Light Violet-Magenta
DYO	166	OUT	;	Dark Yellow-Orange
MOY	167	OUT	;	Medium Orange-Yellow
LDO	168	OUT	;	Light Dull Orange
LWR	169	OUT	;	Light Weak Red
LWM	170	OUT	;	Light Weak Magenta
PDV	171	OUT	;	Pale Dull Violet
DHY	172	OUT	;	Dark Hard Yellow
MFY	173	OUT	;	Medium Faded Yellow
LDY	174	OUT	;	Light Dull Yellow
LWY	175	OUT	;	Light Weak Yellow
PG	176	OUT	;	Pale Gray
PWB	177	OUT	;	Pale Weak Blue
YYS	178	OUT	;	Yellow-Yellow-Spring
LYS	179	OUT	;	Light Yellow-Spring
LSY	180	OUT	;	Light Spring-Yellow
PDS	181	OUT	;	Pale Dull Spring
PWG	182	OUT	;	Pale Weak Green
PWC	183	OUT	;	Pale Weak Cyan
RRP	184	OUT	;	Red-Red-Pink
PPR	185	OUT	;	Pink-Pink-Red
PPM	186	OUT	;	Pink-Pink-Magenta
MMP	187	OUT	;	Magenta-Magenta-Pink
RRO	188	OUT	;	Red-Red-Orange
LHR	189	OUT	;	Light Hard Red
LRP	190	OUT	;	Light Red-Pink
LHP	191	OUT	;	Light Hard Pink
LMP	192	OUT	;	Light Magenta-Pink
LHM	193	OUT	;	Light Hard Magenta
OOR	194	OUT	;	Orange-Orange-Red
LRO	195	OUT	;	Light Red-Orange
LFR	196	OUT	;	Light Faded Red
LPR	197	OUT	;	Light Pink-Red
LPM	198	OUT	;	Light Pink-Magenta
LFM	199	OUT	;	Light Faded Magenta
Ooy	200	OUT	;	Orange-Orange-Yellow
LHO	201	OUT	;	Light Hard Orange
LOR	202	OUT	;	Light Orange-Red
PDR	203	OUT	;	Pale Dull Red
PDP	204	OUT	;	Pale Dull Pink
PDM	205	OUT	;	Pale Dull Magenta
YYO	206	OUT	;	Yellow-Yellow-Orange
LYO	207	OUT	;	Light Yellow-Orange
LOY	208	OUT	;	Light Orange-Yellow
PDO	209	OUT	;	Pale Dull Orange
PWR	210	OUT	;	Pale Weak Red
PWM	211	OUT	;	Pale Weak Magenta
LHY	212	OUT	;	Light Hard Yellow
LFY	213	OUT	;	Light Faded Yellow
PDY	214	OUT	;	Pale Dull Yellow
PWY	215	OUT	;	Pale Weak Yellow
GRAY	235	OUT	;	Gray
; main entry point				
DO NOT TOUCH				
BEGIN				
WHILE TRUE				
SUSPEND				
ENDWHILE				
END				

Per utilizzare un colore all'interno di una delle funzioni dell'oggetto Vector Image fare riferimento alla UNIT COLOR come nell'esempio seguente:

```

MAIN:
;...
VI10AddPen (awBuffer, COLOR.RED) ;Aggiunge il comando PEN - seleziona il colore rosso
;...
WAIT 1
JUMP MAIN
END

```

Inserimento Figure

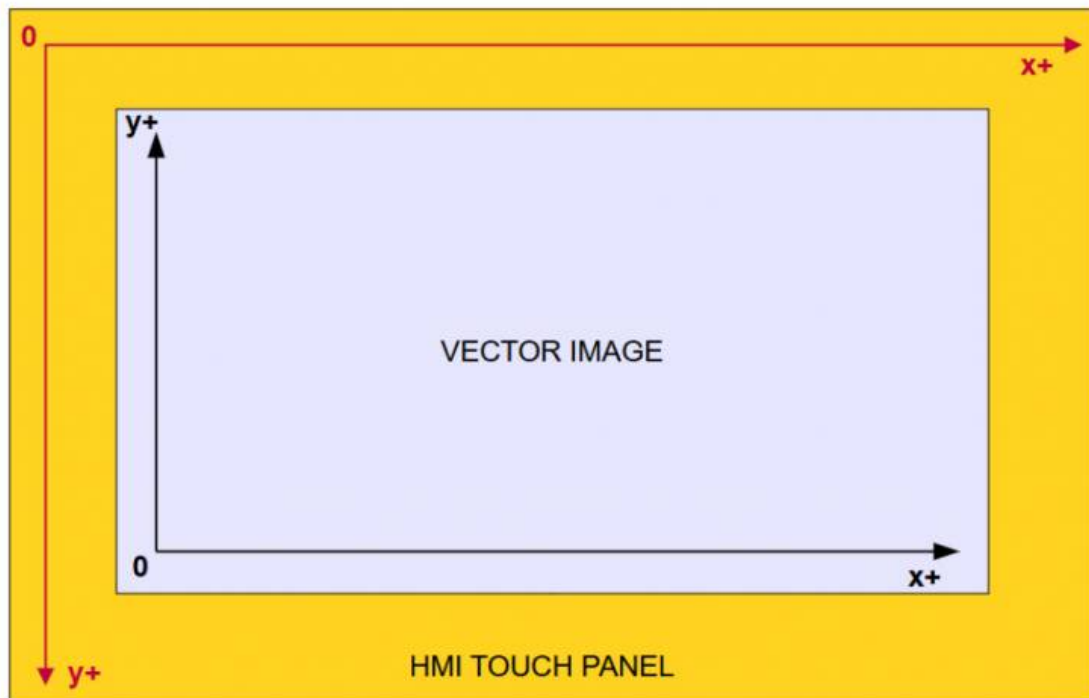


Sezione in costruzione.

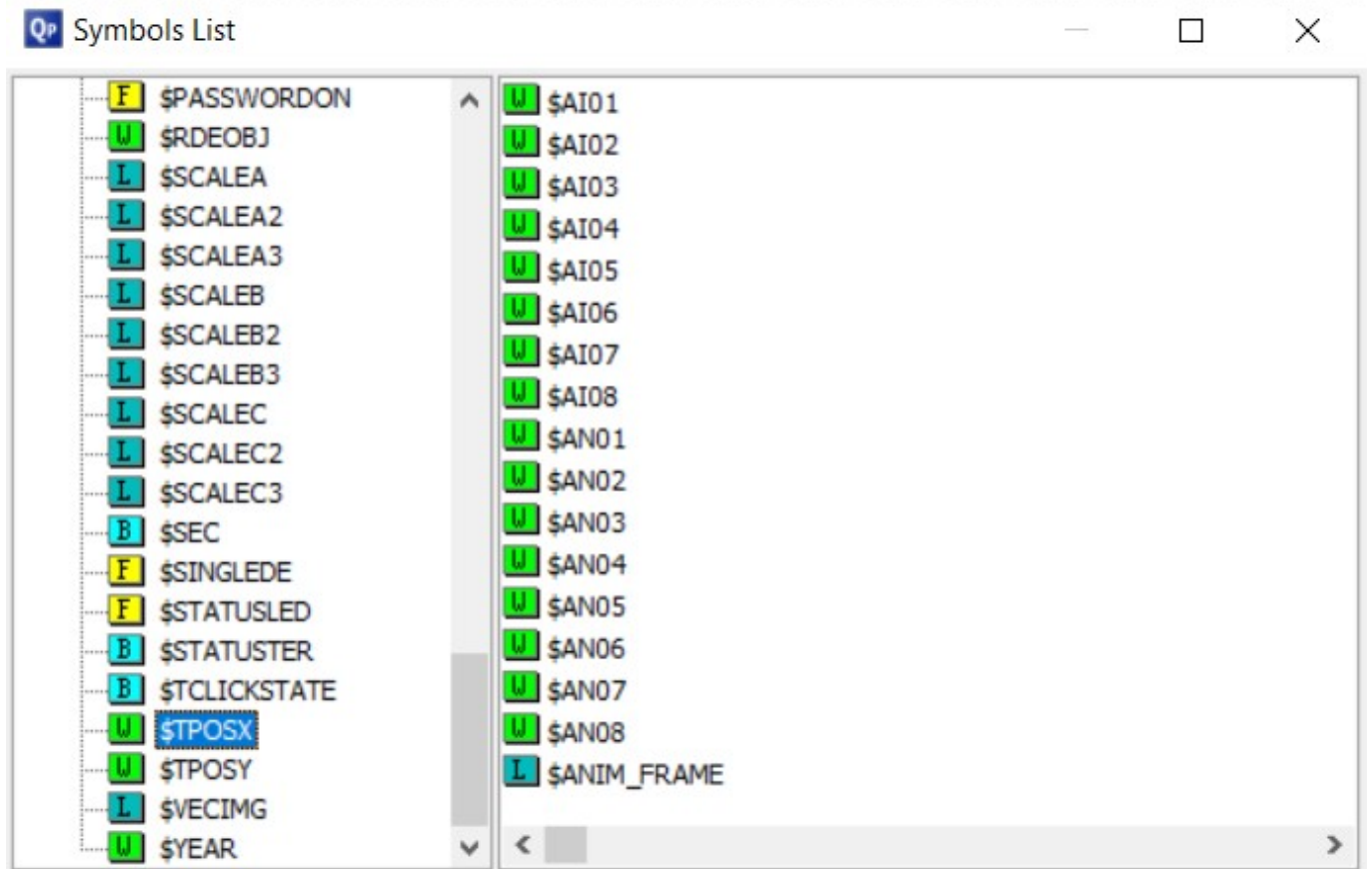
Riferimenti coordinate Touchscreen VS riferimenti Vector Image

L'oggetto Vector Image (figura grigia) ha un sistema di coordinate coerente col piano cartesiano.

Il touchscreen dell'HMI (figura gialla) ha un sistema di coordinate come mostrato in figura.

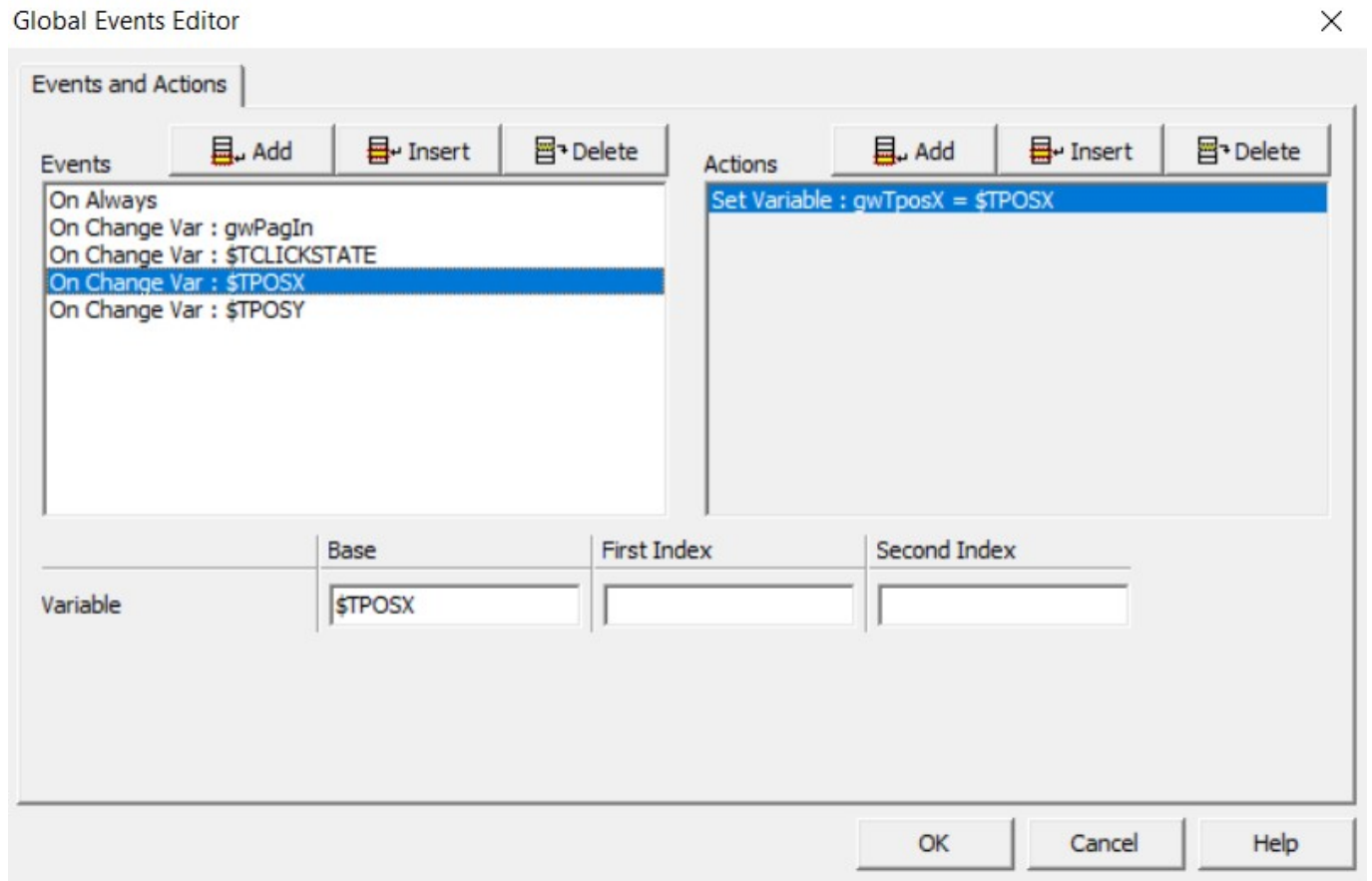


Per accedere alle coordinate del touchscreen si leggono le variabili del terminale \$TP0SX e \$TP0SY



Per utilizzare il contenuto di queste variabili si può creare un evento (globale o di pagina) che copia il valore di queste variabili

in una variabile (anche detta simbolo) del QCL. Nella figura sottostante un esempio.



Documento generato automaticamente da **Qem Wiki** - <https://wiki.qem.it/>

Il contenuto wiki è costantemente aggiornato dal team di sviluppo, è quindi possibile che la versione online contenga informazioni più recenti di questo documento.