Sommario

QVIEW4.1

0.1 Introduzione

L'ambiente di sviluppo QVIEW (QMOVE VIEWer), il linguaggio QCL (QEM Control Language) ed il linguaggio LADDER (logica a contatti), sono gli strumenti necessari per lo sviluppo, il debugging e la modifica dei progetti dedicati al sistema QMOVE. Le potenzialità di questa combinazione di hardware e software, permettono la realizzazione di controlli per l'automazione industriale, con particolare riguardo alla gestione della movimentazione (elettrica, idraulica, CNC o ON / OFF).

0.1.1 LADDER

Il Ladder è un linguaggio di tipo logico a contatti il cui utilizzo risulterà familiare a chi si occupa già di programmazione di PLC. Chi invece si addentra per la prima volta nel mondo ladder dovrà provvedere ad integrare il presente manuale con un corso PLC generico.

0.1.2 QCL: QEM Control Language

Il QCL è un linguaggio di tipo strutturato la cui sintassi e i cui comandi risulteranno familiari a chi si occupa già di programmazione o che comunque ha avuto modo di fare esperienza con un linguaggio ad alto livello.
Chi invece si addentra per la prima volta nel mondo dei linguaggi strutturati potrà trovare in questo manuale un valido supporto di apprendimento.

0.1.3 QVIEW: QMOVE VIEWer

Come ogni linguaggio strutturato e compilato, anche il QCL ed il Ladder necessita di un editor per digitare le righe di codice e di un compilatore che le traduca in linguaggio macchina controllandone la sintassi; QVIEW è l'ambiente di sviluppo che, oltre a supportare il linguaggio QCL ed il linguaggio Ladder, permette di trasferire il programma nella CPU del sistema QMOVE, di monitorare e modificare tutte le variabili ed i parametri del programma ed eventualmente interromperne l'esecuzione per comprendere e correggere eventuali funzionamenti non desiderati (DEBUG).

0.1.4 Programma di apprendimento

Il manuale inizia con l'elaborazione di un semplice programma di prova - Firstapp.qmv - che aiuterà a comprendere le specifiche funzionalità del linguaggio QCL e Ladder; questo esempio sarà utile al lettore per familiarizzare con l'ambiente di sviluppo QVIEW e potrà essere ampliato man mano che verranno introdotti nuovi concetti sospendendo la lettura del manuale per mettere immediatamente in pratica la scrittura e la compilazione di nuove righe di codice. In seguito ci si addentrerà in maniera più rigorosa nella descrizione delle potenzialità e delle particolarità, sia del linguaggio QCL che del linguaggio Ladder che dell'ambiente QVIEW tramite una panoramica completa delle istruzioni.
Durante la lettura di questo volume si consiglia di sperimentare il più possibile quanto appreso, traducendo in pratica i numerosi esempi riportati. Per delucidazioni immediate sulla sintassi QCL e dei blocchi funzionali del Ladder si consiglia di consultare la guida on-line (accessibile dal menu principale di QVIEW).

0.1.4.1 Convenzioni di scrittura

Per la realizzazione degli esempi inseriti nel manuale sono state adottate delle convenzioni di scrittura del codice QCL; fare riferimento al capitolo dedicato.

0.1.4.2 I DEVICE

I device sono delle funzionalità messe a disposizione del programmatore. Essi svolgono le funzioni tipiche dell'automazione industriale, dalla lettura di un ingresso analogico, alla lettura di un segnale proveniente da un traduttore di posizione, fino ad eseguire dei veri e propri posizionamenti di assi.
Ogni device è dotato di un manuale specifico, si rimanda perciò alla documentazione relativa.

Lo scopo di questa pubblicazione è quello di fornire un corso di autoistruzione e uno strumento di consultazione, in modo da permettere al programmatore di sfruttare al meglio e secondo le sue esigenze le potenzialità del sistema QMOVE.

In questa sezione verrà introdotto l'ambiente di sviluppo (QVIEW) presentando un semplice esempio, da considerarsi come supporto all'introduzione ed alla sperimentazione pratica durante l'apprendimento.
Le fasi fondamentali che verranno sviluppate possono essere riassunte in:
- Collegamento del PC a Qmove.
- Apertura ed analisi del progetto di esempio.

- Descrizione e compilazione del progetto di esempio.
- trasferimento del progetto compilato nella CPU (Download).
- Monitoraggio del funzionamento.

0.1.5 Collegamento del PC a Qmove

Per iniziare, collegare il PC a Qmove (porta PROG) tramite il cavo in dotazione; alimentare il tutto. Impostare i parametri della porta di comunicazione ed aprire il collegamento.

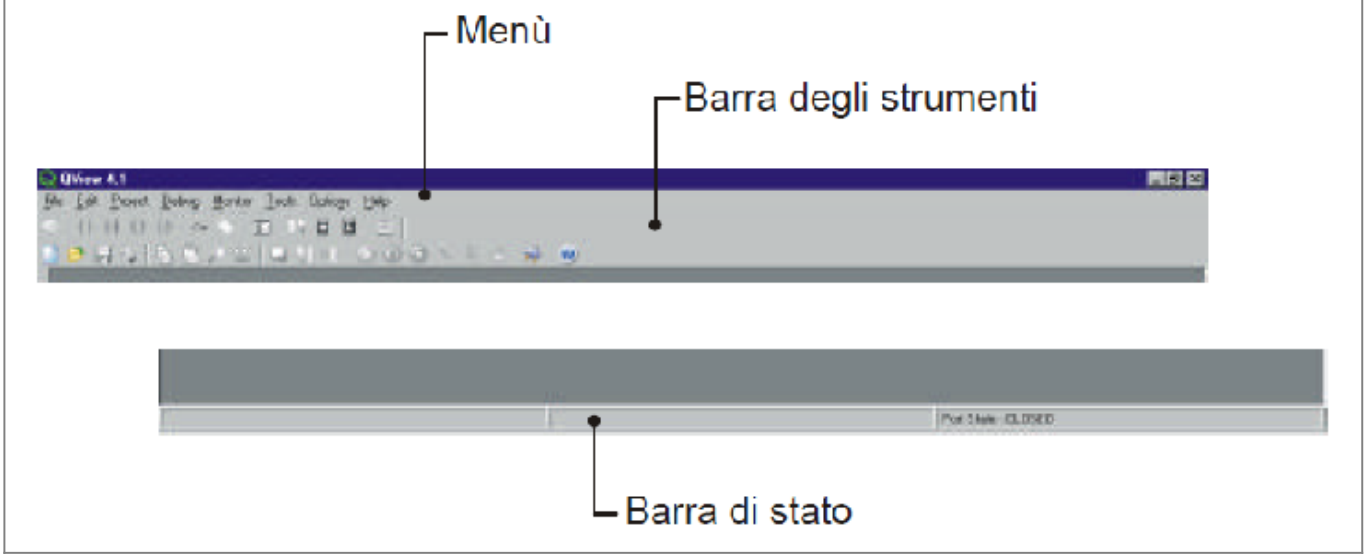
Fare riferimento al manuale di installazione e manutenzione del Qmove utilizzato per lo schema del cavo e per il settaggio degli switch.

0.1.6 Apertura ed analisi del progetto di esempio

Una volta installato il QVIEW 4.1 metterlo in esecuzione selezionando dal menù *Avvio > Qview 4.1*
La schermata iniziale si presenta come in figura 1.

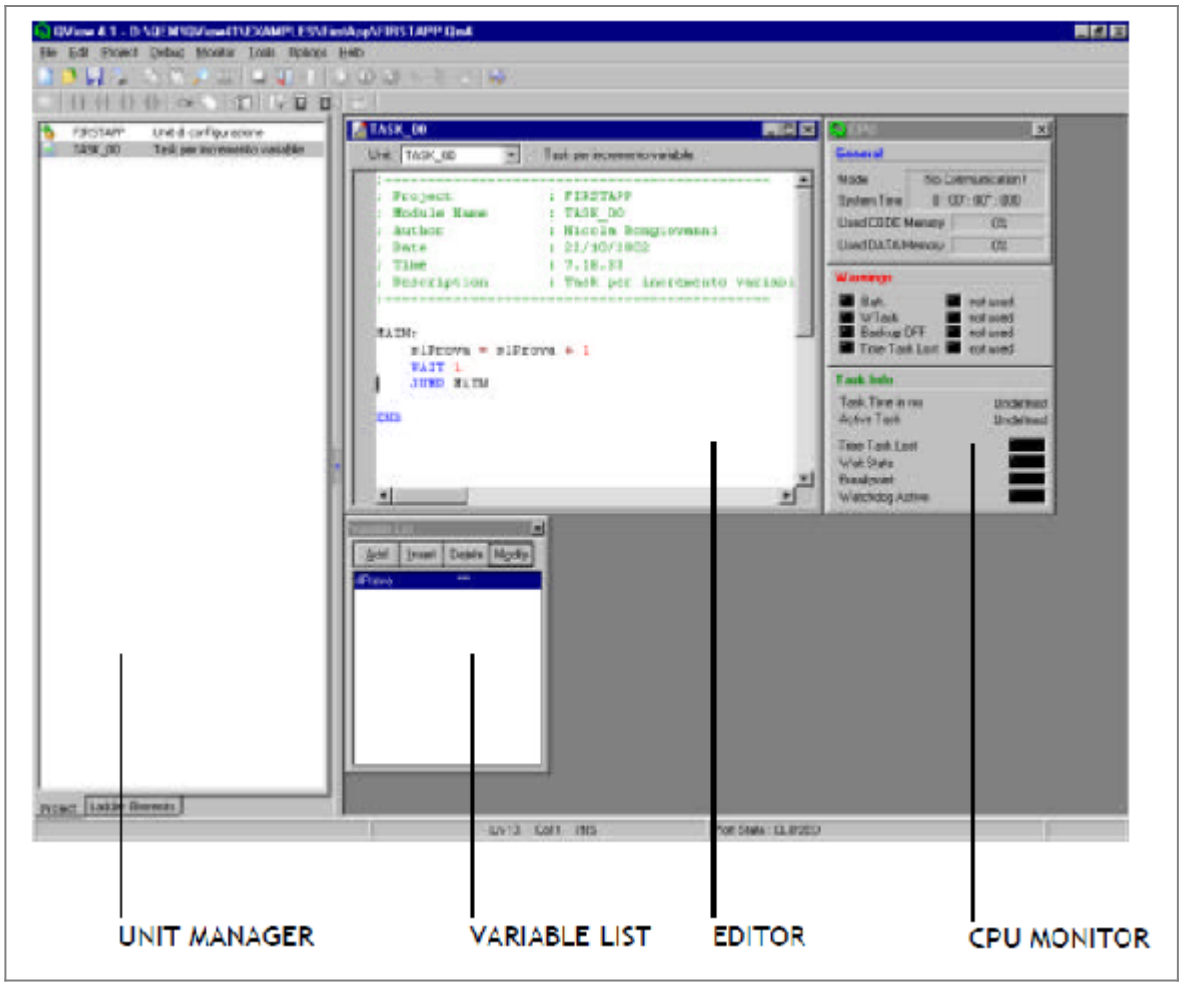
- Nella parte superiore della schermata trovano posto i vari menu da cui si possono compiere tutte le operazioni consentite.
- Immediatamente sotto ai menu si trova la Barra degli strumenti che offre la possibilità di accedere direttamente a tutte le operazioni di base.
- Nella parte inferiore della schermata è stata inserita una Barra di stato per il riepilogo delle informazioni principali.

Figura 1: barre di comando e di stato.



A questo punto si può creare un progetto nuovo oppure aprirne uno già esistente. Per aprire il progetto di esempio selezionare dal menu ***File > Open Project***; selezionare il file ***C:\…\QView41\Examples\FirstApp\Firstapp.qmv***. Successivamente all'apertura del progetto di esempio viene proposta una schermata come da figura 2.

Figura 2: schermata di apertura dell'esempio.



0.1.7 Finestra FIRSTAPP

Indice delle unità che compongono il progetto; la sua intestazione riporta il nome del progetto. L'icona a sinistra di ogni unità ha la funzione di permettere di capire “a colpo d'occhio”, il tipo ed il comportamento Runtime delle unità componenti il progetto. Per avere delle ulteriori informazioni riguardanti l'unità selezionata, è necessario accedere alla finestra Unit property.
- La prima è l’unità di configurazione (FIRSTAPP), contenente tutte le dichiarazioni delle variabili utilizzate nel progetto.
- La seconda unità (TASK_00) è un componente operativo dell’applicativo (viene chiamato modulo), nel quale viene scritto il codice QCL. In questo caso l’unità è di tipo normale.

0.1.8 Finestra CPU

Visualizza una serie di informazioni sullo stato della CPU del sistema QMOVE.

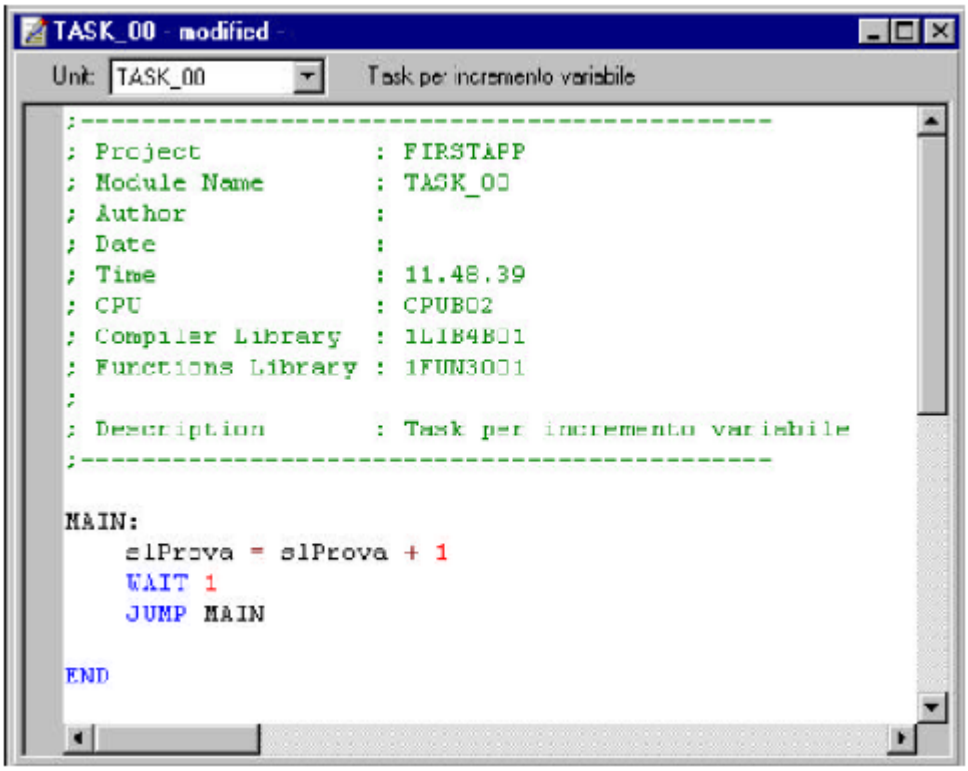
0.1.9 Finestra Variable List

Permette di creare una lista delle variabili da monitorare durante l’esecuzione del proprio progetto.

0.1.10 Finestra TASK_00

Editor di testo per la scrittura delle righe di codice. È possibile visualizzare tutte le unità che compongono il progetto, selezionandole dal menu a tendina presente nella parte superiore della finestra (Figura 3).

Figura 3: esempio finestra di editor.



L'assegnazione dei vari colori è utile per evitare alcuni errori di sintassi. Per esempio, se una parola di codice non è scritta correttamente, non apparirà del colore selezionato e quindi il programmatore ha la possibilità di accorgersi immediatamente dell'errore.

I colori possono essere configurati in base alle abitudini del programmatore.

L'editor di QVIEW 4.1 colora in maniera diversa le parti di codice a seconda della loro funzione.
Selezionando ***Options > Program Setup > QCL Editor*** vengono presentati i colori associati ad ogni parte di codice.

I colori di default sono:
Testo : ..... nero.
Sfondo : ..... bianco.
Program counter : ..... verde scuro.
Breakpoint : ..... rosso.
Bordo della finestra : ... grigio chiaro.
Parole chiave : ..... blu.
Operatori : ..... rosso scuro.
Commenti : ..... verde scuro.
Costanti : ..... rosso.

Dal menu Edit, oltre alle funzioni di taglia, copia ed incolla, possono essere richiamate quelle di ***Find***, ***Replace*** e ***Go To*** che facilitano rispettivamente la ricerca di una parola, la sostituzione di una parola con un'altra ed il salto alla riga voluta; questo menu contiene anche la funzione di ***Next Unit*** e ***Previous Unit*** per visualizzare nell'editor l'unità successiva e precedente. Una funzione molto utile in fase di stesura di un progetto è ***Next Selected Unit*** e ***Previous Selected Unit***. Questa funzione permette di spostarsi non tra tutte le unità ma solo tra quelle selezionate. Per selezionare una unità è sufficiente evidenziarla nella finestra indice e premere la barra spaziatrice (a sinistra dell'unità selezionata appare una L); per deselezionarla si deve premere ancora la barra spaziatrice in corrispondenza di quella unità.

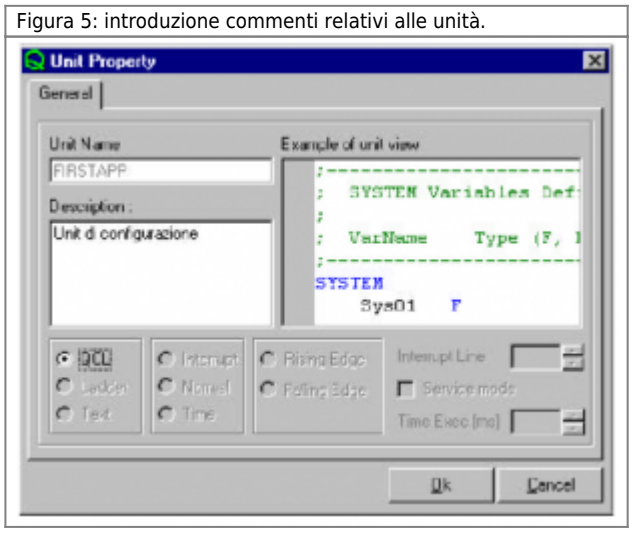
0.1.11 Descrizione e compilazione del progetto di esempio

Selezionando ***File > Project Informations***, vengono visualizzate le informazioni di base del progetto (Figura 4); I dati possono essere introdotti all'apertura di un nuovo progetto.

Figura 4: finestra informazioni di progetto.

In questa finestra sono presenti delle cartelle che suddividono le informazioni di progetto per argomento. Alcune informazioni sono contrassegnate con un asterisco per indicare che sono obbligatorie per una completa informazione sul progetto. Ad ogni unità che compone il progetto è possibile associare un commento che viene visualizzato nella lista delle unità in corrispondenza del relativo nome. Per inserire o modificare l'informazione relativa all'unità selezionata scegliere dal menu ***File > Unit Property*** (Figura 5).

Figura 5: introduzione commenti relativi alle unità.



0.1.12 Unità di configurazione FIRSTAPP

Le righe di codice che iniziano con il carattere ";" sono dei commenti e non hanno alcun effetto a livello esecutivo.

```
;---------------------------------------------------------------------
; Definizione variabili SYSTEM
;---------------------------------------------------------------------
SYSTEM
  slProva L ;Variabile prova di tipo LONG di gruppo SYSTEM
```

In QCL tutte le variabili devono essere dichiarate nel file di configurazione.

Con questa sintassi si dichiara la variabile slProva, di tipo Long e di gruppo *System*.
- *System* raggruppa le variabili che mantengono l'ultimo valore memorizzato prima dello spegnimento del sistema, riproponendolo all'accensione; questo genere di variabili vengono chiamate "ritentive".
- *Long* è un tipo di variabile che occupa in memoria due word di due byte ciascuna, quindi in tutto occupa 32 bit; può quindi contenere valori interi compresi tra –2147483648 e 2147483647.
- *slProva* è il nome assegnato alla variabile.

Altra operazione necessaria è la dichiarazione della composizione del sistema QMOVE, specificando quali sono le schede montate sul BUS. Per ogni slot occupato (1, 2, 3, 4, ...), deve essere inserito il codice identificativo della scheda.

```
;---------------------------------------------------------------------
; Configurazione BUS
```

```
;--------------------------------------------------------------------
BUS
  1 1CPUD 01
  2 . .
  3 . .
  4 . .
```

Le precedenti righe di codice servono per configurare un BUS sul quale è stata inserita la sola scheda CPUD 01 sullo slot 1 (gli altri 3 slot sono liberi).
Se viene utilizzata una scheda diversa dalla Q1-CPU-Dx01, fare riferimento alle descrizioni dell'unità di configurazione (sezione BUS).

0.1.13 Unità TASK_00

Le righe di codice che iniziano con il carattere ";" sono dei commenti e non hanno alcun effetto a livello esecutivo.

L'unità di esempio, in seguito all'istruzione:

```
slProva = slProva + 1
```

incremento di uno ad ogni ciclo il valore della variabile slProva (dichiarata nell'unità di configurazione).

```
MAIN:
  slProva = slProva + 1
  WAIT 1
  JUMP MAIN
END
```

0.1.14 Compilazione dell'esempio

L'insieme delle unità QCL deve essere compilato e trasferito alla CPU per essere eseguito. Per avviare la compilazione selezionare dal menu ***Project > Compile***. Al termine della compilazione viene proposta la finestra di figura 6 riportante i risultati della compilazione. Nel caso dell'esempio, la compilazione dovrebbe aver avuto buon fine e quindi appare il messaggio ***Project compiling OK!*** (compilazione del progetto eseguita correttamente).

Figura 6: finestra di compilazione.

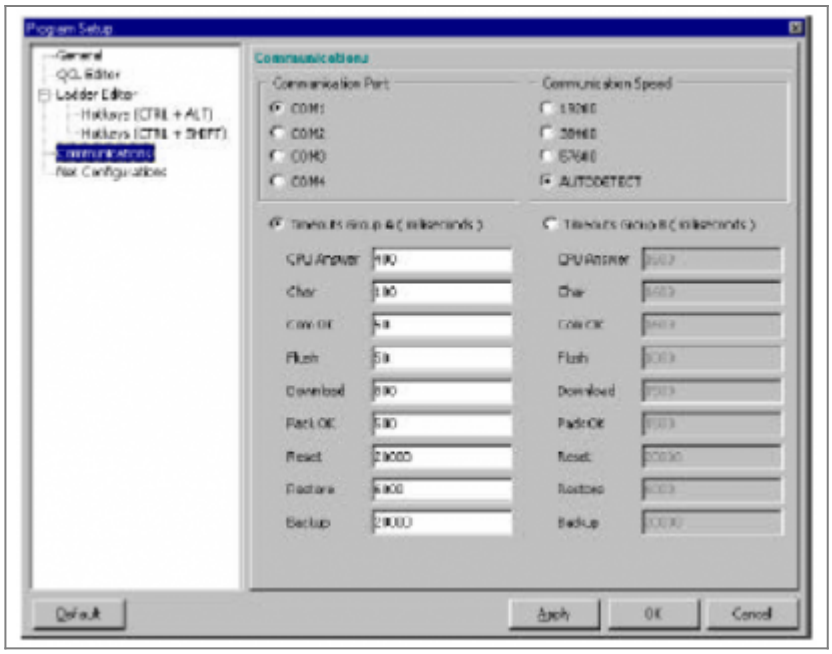
0.1.15 Trasferire il progetto compilato nella CPU (Download)

Per trasferire il progetto elaborato e compilato dal proprio PC all'interno della CPU del sistema QMOVE, è necessario stabilire una comunicazione seriale; per abilitare questa comunicazione bisogna compiere delle operazioni di settaggio sia sul PC che sulla CPU stessa.

0.1.16 Comunicazione seriale PC - QMOVE

La CPU dispone di 8 dip-switch per la configurazione della comunicazione seriale; per informazioni sulla loro localizzazione e numerazione fare riferimento al fascicolo "Caratteristiche firmware" della scheda utilizzata. Il settaggio del PC deve essere impostato da QVIEW, selezionando ***Options > Program setup > Communications*** (Figura 7). E' possibile selezionare la porta seriale del PC (COM) da utilizzare e la velocità di trasmissione (espressa in Baud); la velocità impostata deve essere la stessa che è stata settata sulla scheda CPU tramite gli appositi switch. È comunque possibile selezionare *AUTODETECT* che permette di rilevare automaticamente la velocità di trasmissione.

Figura 7: impostazione porta comunicazione seriale.



Per aprire la comunicazione seriale, selezionare ***Options > Open COM***. Nella barra di stato (finestra QVIEW - in basso a destra) vengono visualizzati i parametri della comunicazione seriale. Esempio: lo stato visualizzato potrebbe essere *Port State: OPEN – BIN1 - 38400* indicante che la comunicazione è aperta e funzionante con protocollo BIN1 e velocità 38400 baud.

0.1.17 Download del progetto

A questo punto è possibile scaricare il progetto nella CPU del sistema QMOVE, selezionando ***Project > Download***. Alla fine dell'operazione viene visualizzata la finestra di ***Status Download*** con il messaggio ***System READY*** per indicare che il download è andato a buon fine.

0.1.18 Monitoraggio del funzionamento

Il sistema ora è pronto per mettere in esecuzione l'applicativo appena scaricato; si noti che la finestra CPU presenta il messaggio *READY*.
Selezionando il comando ***Debug > Run***, l'applicativo scaricato viene messo in esecuzione; lo stato della CPU diventa *RUN* e si può osservare che il valore della variabile *slProva* (finestra ***Variable List***), viene incrementato molto velocemente.

0.2 Introduzione alla programmazione

0.2.1 Unità di configurazione

L'unità di configurazione è un componente fondamentale di un progetto per QMOVE; è unica nel progetto e si accompagna al resto delle unità componenti il progetto (che possono essere più di una).

L'unità di configurazione contiene le dichiarazioni di tutte le variabili e di tutte le costanti utilizzate nell'applicativo; inoltre viene anche definita la composizione hardware del sistema QMOVE adottato, specificando il tipo di scheda CPU, le schede intelligenti e le schede non intelligenti presenti sul rack. L'unità di configurazione deve essere editata tramite l'editor dell'ambiente QVIEW.

L'unità di configurazione è composta da una serie di sezioni a seconda del gruppo di variabili da dichiarare. Per la definizione specifica delle varie sezioni si rimanda ai paragrafi seguenti; ogni sezione è identificata da una parola chiave che indica l'inizio della sezione stessa all'interno dell'unità di configurazione.

Le parole chiavi sono: - CONST - SYSTEM - GLOBAL - ARRSYS - ARRGBL - TIMER - INPUT - OUTPUT - DATAGROUP - BUS - INTDEVICE (sezione inserita nel capitolo dedicato ai device) - EXTDEVICE (sezione inserita nel capitolo dedicato ai device)

Gli esempi di composizione dell'unità di configurazione vengono presentati nel resto del manuale al momento delle dichiarazioni delle variabili.

0.2.2 Tipi di variabili

QCL prevede cinque tipi principali di dato:

0.2.2.1 Flag

Il dato di tipo FLAG è utilizzato per la definizione di variabili booleane che hanno un range di valori compreso tra 0 e 1.
L'occupazione della memoria da parte delle variabili di questo tipo dipende dal loro numero.
La sintassi per la definizione di una variabile di tipo FLAG nell'unità di configurazione è la seguente:

```
<nome variabile> F
```

0.2.2.2 Byte

Il dato di tipo BYTE è utilizzato per la definizione di variabili che hanno un range di valori compreso tra -128 e +127; ogni variabile occupa un byte di memoria.
La sintassi per la definizione di una variabile BYTE nell'unità di configurazione è la seguente:

```
<nome variabile> B
```

0.2.2.3 Word

Il dato di tipo WORD è utilizzato per la definizione di variabili che hanno un range di valori compreso tra -32768 e +32767; ogni variabile occupa due byte di memoria.
La sintassi per la definizione di una variabile WORD nell'unità di configurazione è la seguente:

```
<nome variabile> W
```

0.2.2.4 Long

Il dato di tipo LONG è utilizzato per la definizione di variabili che hanno un range di valori compreso tra -2147483648 e +2147483647; ogni variabile occupa quattro byte di memoria. La sintassi per la definizione di una variabile LONG nell'unità di configurazione è la seguente:

```
<nome variabile> L
```

0.2.2.5 Single

Il dato di tipo SINGLE è utilizzato per la definizione di variabili reali che hanno un range di valori compreso tra -3.4 x 1038 e +3.4 x 1038; ogni variabile occupa quattro byte di memoria.
La sintassi per la definizione di una variabile SINGLE nell'unità di configurazione è la seguente:

```
<nome variabile> S
```

La precisione massima delle variabili SINGLE è di sette cifre (contando le cifre prima e dopo la virgola). Esempio:
1,234567 x 103 Precisione del millesimo; incremento minimo = 1 millesimo.
1,234567 x 104 Precisione del centesimo; incremento minimo = 1 centesimo.
1,234567 x 109 Precisione delle centinaia; incremento minimo = 100.

0.2.2.6 Tabella riassuntiva dei tipi di variabili utilizzabili

Assegnando ad una variabile un valore esterno al range consentito si verifica la condizione di overflow.

Tipo dato	Codice	Spazio occupato memoria (Bit)	Intervallo
FLAG	F	1	0 ÷ 1
BYTE	B	8	-128 ÷ 127
WORD	W	16	-32768 ÷ 32767
LONG	L	32	-2147483648 ÷ 2147483647
SINGLE	S	32	-3.4 x 10E38÷+3.4 x 10E38

0.2.3 Gli Identificatori

Il carattere punto e virgola (";") ha lo scopo di identificare una riga di commento. Tutto il testo inserito dopo questo carattere non verrà considerato codice QCL. Il carattere punto e virgola vale solo per la linea in cui è inserito.

Gli identificatori sono dei nomi tramite i quali è possibile fare dei riferimenti ad oggetti o ad etichette. Ogni identificatore è formato da uno o più caratteri alfanumerici tenendo presente che il primo deve essere una lettera. I caratteri alfanumerici possono essere riassunti in:
- dalla "A" alla "Z"
- dalla "a" alla "z"
- da "0" a "9"

- “_”

0.2.3.1 I nomi

I nomi vengono usati per agevolare l'identificazione di un oggetto all'interno del sistema QMOVE. Per oggetto si intende qualsiasi entità gestibile a livello di linguaggio e provvista di caratteristiche fisiche proprie come ad esempio: variabili, ingressi, uscite e devices interni od esterni. I nomi sono limitati ad una lunghezza massima di 12 caratteri ed essendo immagazzinati nella memoria del QMOVE incidono per un byte a carattere nell'occupazione di tale risorsa. In tutto il progetto non vi possono essere due oggetti con lo stesso nome.

0.2.4 Le costanti

Una costante è un carattere o una stringa di caratteri utilizzabile come valore in un applicativo. Le costanti vengono dichiarate nell’unità di configurazione e devono essere poste in seguito alla parola chiave “CONST”. Viene riportata la sintassi per la definizione di costanti:

```
;---------------------------------------------------------------------
; Definizione costanti
;---------------------------------------------------------------------
CONST
<nome costante> <valore>
```

dove:

CONST	Parola chiave che indica la definizione delle costanti.
<nome costante>	Nome della costante.
<valore>	valore associato alla costante.

0.2.4.1 Esempio: sviluppo Firstapp.qm4

Quanto descritto in questo esempio deve essere inserito nel progetto Firstapp.qm4 in modo da poter mettere immediatamente in pratica quanto appreso.

Nell’unità di configurazione (firstapp) aggiungiamo la sezione per la dichiarazione delle costanti:

```
;---------------------------------------------------------------------
; Definizione COSTANTI
;---------------------------------------------------------------------
CONST
  TM_SECONDO_1000 ;Costante con valore 1000 per il tempo di un secondo.
  DIM_ARRAY_7     ;Costante con valore 7 per la dimensione dell’array.
  DIM_PROG_10     ;Costante con valore 10 per il numero dei programmi nel datagoup.
  DIM_STEP_5      ;Costante con valore 5 per il numero degli step nel datagroup.
```

0.2.5 Le variabili SYSTEM

Le variabili SYSTEM mantengono il loro valore anche allo spegnimento del sistema.

Con il nome SYSTEM vengono raggruppate tutte le variabili ritentive di uso generico; possono essere di tipo FLAG, BYTE, WORD, LONG o SINGLE e sono accessibili sia in scrittura che in lettura. Vengono dichiarate nell’unità di configurazione e devono essere poste in seguito alla parola chiave “SYSTEM”. Viene riportata la sintassi per la definizione di variabili SYSTEM.

```
;---------------------------------------------------------------------
; Definizione variabili SYSTEM
;---------------------------------------------------------------------
SYSTEM
  <nome variabile>    <tipo>
```

dove:

SYSTEM	Parola chiave che indica la definizione di variabili SYSTEM.
<nome variabile>	Nome della variabile SYSTEM.
<tipo>	Tipo della variabile e può essere: F (FLAG) B (BYTE) W (WORD) L (LONG) S (SINGLE)

0.2.5.1 Esempio: sviluppo Firstapp.qm4

Quanto descritto in questo esempio deve essere inserito nel progetto Firstapp.qm4 in modo da poter mettere immediatamente in pratica quanto appreso.

Attualmente nell’unità firstapp di esempio è già presente la sezione SYSTEM (nella quale è dichiarata la variabile slProva); aggiungiamo le seguenti variabili:

```
;---------------------------------------------------------------------
; Definizione Variabili SYSTEM
;---------------------------------------------------------------------
SYSTEM
  slProva L    ;Variabile di prova dimensione LONG tipo SYSTEM.
  sbSecondi_B  ;Variabile per i secondi.
  sbMinuti_B   ;Variabile per i minuti.
  swOre W      ;Variabile per le ore.
```

0.2.6 Le variabili GLOBAL

Con il nome GLOBAL vengono raggruppate tutte le variabili non ritentive ad uso generico; hanno la caratteristica di venire azzerate ad ogni accensione del sistema; possono essere di tipo FLAG, BYTE, WORD, LONG e SINGLE e sono accessibili sia in scrittura che in lettura. Vengono dichiarate nell'unità di configurazione e devono essere poste in seguito alla parola chiave "GLOBAL". Viene riportata la sintassi per la definizione di variabili GLOBAL.

```
;---------------------------------------------------------------------
; Definizione variabili GLOBAL
;---------------------------------------------------------------------
GLOBAL
  <nome variabile>   <tipo>
```

dove:

GLOBAL	Parola chiave che indica per la definizione di variabili GLOBAL.
<nome variabile>	Nome della variabile GLOBAL.
<tipo>	Tipo della variabile e può essere: F (FLAG) B (BYTE) W (WORD) L (LONG) S (SINGLE)

Per esempi relativi alla sintassi per la definizione, fare riferimento a quelli indicati per le variabili SYSTEM.

Quanto descritto in questo esempio deve essere inserito nel progetto Firstapp.qm4 in modo da poter mettere immediatamente in pratica quanto appreso.

0.2.6.1 Esempio: sviluppo Firstapp.qm4

Nell'unità di configurazione (firstapp), in seguito alla sezione SYSTEM aggiungiamo la sezione per la dichiarazione del gruppo GLOBAL:

```
;---------------------------------------------------------------------
; Definizione Variabili GLOBAL
;---------------------------------------------------------------------
GLOBAL
  gfMioFlag      F     ;Variabile global di tipo Flag.
  gbTuoByte      B     ;Variabile global di tipo Byte.
  gwSuoWord      W     ;Variabile global di tipo Word.
  glNostroLong   L     ;Variabile global di tipo Long.
  gsVostroSing   S     ;Variabile global di tipo Single.
```

0.2.7 Utilizzo variabili SYSTEM e GLOBAL

Le variabili SYSTEM e GLOBAL possono essere utilizzate sia a destra che a sinistra di un'assegnazione o dentro un'espressione (indicandone soltanto il nome). Non è possibile indicizzare una variabile di questo tipo.
Esempi di scrittura ed utilizzo delle variabili system e global sono inseriti nello sviluppo del progetto ***Firstapp.qm4***.

0.2.8 Le variabili ARRAY SYSTEM

Una variabile Array System è un insieme di variabili ritentive dello stesso tipo, aventi la stessa dimensione ed alle quali è possibile accedere tramite un nome comune, riferendosi ad uno specifico elemento tramite un indice. Non sono previsti array di variabili tipo FLAG.
Come le SYSTEM sono variabili ritentive di uso generale e devono essere dichiarate di seguito alla parola chiave "ARRSYS".
Viene riportata la sintassi per la definizione di variabili ARRAY SYSTEM.

```
;---------------------------------------------------------------------
; Definizione variabili ARRAY SYSTEM
;---------------------------------------------------------------------
ARRSYS
  <nome variabile>   <tipo>   <numero_elementi>
```

dove:

GLOBAL	Parola chiave che per la definizione di variabili ARRAY SYSTEM.
<nome variabile>	Nome della variabile ARRAY SYSTEM.

<tipo>	Tipo della variabile e può essere: B (BYTE) W (WORD) L (LONG) S (SINGLE)
<numero_elementi>	Numero di elementi che compongono la variabile.

Il numero massimo di elementi consentiti in un array è di 65535. Non si possono indicare dimensioni negative o dimensioni con parte decimale. Come dimensione dell'array è possibile usare delle costanti già definite nella sezione CONST; se questa avesse valore decimale verrà troncato: ad esempio il valore 200.34 viene forzato a 200.

0.2.8.1 Esempio: sviluppo Firstapp.qm4

Quanto descritto in questo esempio deve essere inserito nel progetto Firstapp.qm4 in modo da poter mettere immediatamente in pratica quanto appreso.

Nell'unità di configurazione (firstapp), in seguito alla sezione GLOBAL aggiungiamo la sezione per la dichiarazione del gruppo ARRAY SYSTEM.

```
;---------------------------------------------------------------------
; Definizione variabili ARRAY SYSTEM
;---------------------------------------------------------------------
ARRSYS
  asbMioArray B 10         ;Dichiarazione di un array system di byte di dimensione 10.
  aslTuoArray L DIM_ARRAY  ;Dichiarazione di un array system di long di dimensione DIM_ARRAY.
```

Si osservi che nel secondo array si è usata una costante per la dichiarazione della dimensione.

0.2.9 Le variabili ARRAY GLOBAL

Una variabile Array Global è un insieme di variabili non ritentive dello stesso tipo, aventi la stessa dimensione ed alle quali è possibile accedere tramite un nome comune, riferendosi ad uno specifico elemento tramite un indice. Non sono previsti array di variabili tipo FLAG.
Come le GLOBAL sono variabili non ritentive di uso generale e devono essere dichiarate di seguito alla parola chiave "ARRGBL".
Viene riportata la sintassi per la definizione di variabili ARRAY GLOBAL:

```
;---------------------------------------------------------------------
; Definizione variabili ARRAY GLOBAL
;---------------------------------------------------------------------
ARRGBL
  <nome variabile>   <tipo>   <numero_elementi>
```

dove:

ARRGBL	Parola chiave per la definizione di variabili ARRAY GLOBAL.
<nome variabile>	Nome della variabile ARRAY GLOBAL.
<tipo>	Tipo della variabile e può essere: B (BYTE) W (WORD) L (LONG) S (SINGLE)
<numero_elementi>	Numero di elementi che compongono la variabile.

Per esempi relativi alla sintassi per la definizione, fare riferimento a quelli indicati per le variabili ARRAY SYSTEM.

0.2.9.1 Esempio: sviluppo Firstapp.qm4

Quanto descritto in questo esempio deve essere inserito nel progetto Firstapp.qm4 in modo da poter mettere immediatamente in pratica quanto appreso.

Nell'unità di configurazione (firstapp), in seguito alla sezione ARRAY SYSTEM aggiungiamo la sezione per la dichiarazione del gruppo ARRAY GLOBAL.

```
;---------------------------------------------------------------------
; Definizione variabili ARRAY GLOBAL
;---------------------------------------------------------------------
ARRGBL
arwMioArray W 15 ;Dichiarazione di un array global di byte di dimensione 15.
```

Un esempio della modalità di scrittura in un'array viene fatto in seguito nell'esempio relativo all'istruzione FOR / NEXT.

0.2.10 Utilizzo variabili ARRAY SYSTEM ed ARRAY GLOBAL

Le variabili Array possono essere utilizzate sia a destra che a sinistra di un'assegnazione o dentro un'espressione con la

seguente sintassi:

```
< Nome Array >[ i ]
```

dove “i” può essere un numero, una costante, una variabile (non di tipo Single) o un’espressione complessa. Gli indici di un array partono sempre da 1 e non sono ammessi valori superiori al numero massimo di elementi.

0.2.11 Le variabili TIMER

Sono variabili utilizzate per realizzare temporizzazioni alle quali può essere assegnato un valore intero (espresso in ms) che rappresenta il tempo che deve trascorrere (dal momento dell’assegnazione); in lettura è disponibile lo stato di “temporizzazione terminata” (1) o “temporizzazione attiva” (0).
Mediante l’istruzione *<nome variabile>:remain* è possibile leggere il valore del tempo rimanente prima della fine del timer.
Le variabili timer vengono dichiarate nell’unità di configurazione e devono essere poste in seguito alla parola chiave “TIMER”.
Viene riportata la sintassi per la definizione di variabili TIMER.

```
;--------------------------------------------------------------------
; Definizione variabili TIMER
;--------------------------------------------------------------------
TIMER
  <nome variabile>
```

dove:

TIMER	Parola chiave per la definizione di variabili TIMER.
<nome variabile>	Nome della variabile TIMER.

0.2.11.1 Esempio: sviluppo Firstapp.qm4

Quanto descritto in questo esempio deve essere inserito nel progetto Firstapp.qm4 in modo da poter mettere immediatamente in pratica quanto appreso.

Nell’unità di configurazione (firstapp), in seguito alla sezione ARRAY GLOBAL aggiungiamo la sezione per la dichiarazione del gruppo TIMER.

```
;--------------------------------------------------------------------
; Definizione TIMER
;--------------------------------------------------------------------
TIMER
  tTempo1   ;Identificativo del primo timer.
  tTempo2   ;Identificativo del secondo timer.
  tSecondi  ;Identificativo del timer per i secondi.
  tDelay    ;Identificativo del timer per un ritardo.
```

Un’esempio di uso della variabile timer viene presentato in seguito nell’esempio per l’istruzione IF / ELSE / ENDIF per la realizzazione di un semplice orologio.

0.2.12 Utilizzo variabili TIMER

Le variabili timer possono essere utilizzate sia a destra che a sinistra di un’assegnazione o dentro un’espressione indicando semplicemente il nome della variabile:

```
< Nome Timer >
```

Quando la variabile timer si trova a sinistra dell’assegnazione si definisce il caricamento di un valore sul timer (valore espresso in millisecondi):

```
tMioTimer = 1000  ;Imposta il timer tMioTimer a 1 secondo.
```

Quando la variabile timer si trova a destra dell’assegnazione o all’interno di una espressione ne viene letto lo stato (0 = Temporizzazione attiva, 1 = Temporizzazione terminata):

```
gfIsTimerEnd = tMioTimer ;Assegna alla variabile gfIsTimerEnd lo stato del timer.
```

oppure

```
IF(tMioTimer)  ;Se timer tMioTimer terminato esegue il blocco codice all’interno dell’IF.
  .....
  .....
ENDIF
```

È, inoltre, possibile leggere il valore del tempo rimanente prima dello scadere del timer (il valore restituito è espresso in millisecondi):

```
< Nome Timer >:remain
```

Esempio:

```
glTempoRiman = tMioTimer:remain
```

0.2.13 Le variabili INPUT ed OUTPUT

Sono tutte le variabili che fanno riferimento ad ingressi od uscite digitali. Vengono dichiarate nell'unità di configurazione; devono essere poste in seguito alle parole chiave "INPUT" per gli ingressi o "OUTPUT" per le uscite.
Viene riportata la sintassi per la definizione di variabili INPUT ed OUTPUT.

```
;---------------------------------------------------------------------
; Definizione variabili INPUT ed OUTPUT
;---------------------------------------------------------------------
INPUT
  <nome variabile>   <tipo>   <io address>
OUTPUT
  <nome variabile>   <tipo>   <io address>
```

Gli I/O address sono disponibili sulle schede tecniche hardware delle schede utilizzate.

dove:

INPUT	Parola chiave per la definizione di variabili INPUT.
OUTPUT	Parola chiave per la definizione di variabili OUTPUT.
<nome variabile>	Nome della variabile.
<tipo>	Tipo della variabile e può essere: F per (FLAG) B per (BYTE)
<io address>	Indirizzo dell'INPUT o OUTPUT così composto: *Slot number.name: Slot number* è il numero dello slot nel quale è posta la scheda con la risorsa hardware. *name*: è il nome che fa riferimento all'indirizzo fisico dell'I/O (definito nei riferimenti hardware).

Un'applicazione interessante degli ingressi e delle uscite digitali è quello del raggruppamento degli stessi in un unico identificatore. Questo identificatore è analogo ad una variabile di otto bit dove ogni ingresso o uscita digitale rappresenta un bit.
Se per esempio abbiamo una terza scheda MIX montata sullo slot 3 sono possibile le ulteriori dichiarazioni:
Nella sezione INPUT

```
ibIngresso B 3.INPB     ;8 ingressi digitali formano un unico ingresso di dimensione 1 byte.
```

Nella sezione OUTPUT

```
obUscita B 3.OUTB       ;8 uscite digitali formano un'unica uscita di dimensione 1 byte.
```

Se si dispone di schede con più di otto ingressi o uscite digitali (per esempio D24, I24 oppure O24), è possibile raggrupparli in gruppi di otto modificando la dichiarazione:
Nella sezione INPUT

```
ibIngresso1 B 3.INPB1   ;Primo gruppo di otto (1÷8) ingressi
                        ;digitali raggruppati in un byte.
ibIngresso2 B 3.INPB2   ;Secondo gruppo di otto (9÷16)
                        ;ingressi digitali raggruppati in
                        ;un byte.
```

Nella sezione OUTPUT

```
obUscita1 B 3.OUTB1     ;Primo gruppo di otto (1÷8) uscite
                        ;digitali raggruppate in un byte.
obUscita2 B 3.OUTB2     ;Secondo gruppo di otto (9÷16) uscite
                        ;digitali raggruppate in un byte.
```

0.2.13.1 Esempio: sviluppo Firstapp.qm4

Quanto descritto in questo esempio deve essere inserito nel progetto Firstapp.qm4 in modo da poter mettere immediatamente in pratica quanto appreso.

Per inserire la sezione relativa alla dichiarazione del gruppo di INPUT e di OUTPUT è necessario disporre di una scheda che metta a disposizione un insieme di ingressi e di uscite digitali. In questo esempio faremo riferimento ad una scheda MIX, la quale dispone di otto ingressi e otto uscite digitali identificabili nei manuali delle schede tecniche con X.INP01 … X.INP08 e X.OUT01 … X.OUT08, rispettivamente, dove X è il numero dello slot in cui è installata la scheda. Nell'unità di configurazione (firstapp), in seguito alla sezione TIMER aggiungiamo la sezione per la dichiarazione delle variabili INPUT ed OUTPUT.

```
;---------------------------------------------------------------------
; Definizione degli INPUT
;---------------------------------------------------------------------
INPUT
  ifSetUpOre F 2.INP01   ;Primo ingresso digitale della scheda
                         ;nello slot 2.
  ifSetUpMin F 2.INP02   ;Secondo ingresso digitale della scheda
                         ;nello slot 2.
  ifAbilU1 F 2.INP03     ;Terzo ingresso digitale della scheda
```

```
                        ;nello slot 2.
;---------------------------------------------------------------------
; Definizione degli OUTPUT
;---------------------------------------------------------------------
OUTPUT
  ofUscita1 F 2.OUT01    ;Prima uscita digitale della scheda
                         ;nello slot 2.
  ofUscita2 F 2.OUT02    ;Seconda uscita digitale della scheda
                         ;nello slot 2.
```

0.2.14 Le variabili DATAGROUP

Le variabili Datagroup sono una particolare struttura di dati. Quando si dichiara un datagroup, si organizza una parte della memoria come una tabella formata da righe e colonne. Le colonne sono chiamate “programmi”, mentre le righe “passi” (in inglese “step”).
Ogni programma (colonna) contiene due tipologie di variabili:
- Statiche.
- Indicizzate.

Le statiche sono delle variabili che possono assumere un valore diverso a seconda del programma (colonna) a cui si fa riferimento. Nella dichiarazione, ognuna di queste variabili è identificata con un unico nome quindi, per poter far riferimento ai diversi valori che può assumere, si deve utilizzare un metodo di indicizzazione. Per esempio, per far riferimento alla variabile “dslVeMa” del programma (colonna) 5, viene adottato questo metodo:
dslVeMa[5]

Le indicizzate sono variabili che possono assumere un valore diverso a seconda del programma e del passo (riga) a cui si fa riferimento. Nella dichiarazione, ognuna di queste variabili è identificata con un unico nome quindi, per potersi riferire ai diversi valori che può assumere, si deve utilizzare un metodo di indicizzazione. Per esempio, per far riferimento alla variabile “ddwLuPe” del programma (colonna) 5 e del passo (riga) 3, viene adottato questo metodo:
ddwLuPe[5,3]

Graficamente il datagroup si può rappresentare in questo modo:

_	**Prog.1**	**Prog. 2**	**Prog. 3**	**Prog. 4**	**Prog. 5**	_
_	dslVeMa[1]	dslVeMa[2]	dslVeMa[3]	dslVeMa[4]	dslVeMa[5]	**Statiche**
Step 1	ddwLuPe[1,1]	_	_	_	_	**Indicizzate**
Step 2	ddwLuPe[1,2]	_	_	_	_	
Step 3	ddwLuPe[1,3]	_	_	_	ddwLuPe[5,3]	
Step 4	_	_	_	_	ddwLuPe[5,4]	
Step 5	_	_	_	_	ddwLuPe[5,5]	

Nelle due sezioni, statiche e indicizzate, del datagroup è possibile dichiarare più di una variabile.

Viene riportata la sintassi per la definizione di variabili DATAGROUP:

```
;---------------------------------------------------------------------
; Definizione del DataGroup
;---------------------------------------------------------------------
DATAGROUP
  <nome DataGroup>
;Definizione del numero programmi
  DATAPROGRAM
    <numero programmi>
;Definizione variabili statiche
    <nome variabile> <tipo>
    <nome variabile> <tipo>
    <nome variabile> <tipo>
;Definizione del numero di passi
  STEP
    <numero passi>
;Definisce le variabili indicizzate di ciascun passo
    <nome variabile> <tipo>
    <nome variabile> <tipo>
    <nome variabile> <tipo>
```

dove:

DATAGROUP	Parola chiave per la definizione di un DataGroup.
<nome DataGroup	Nome associato al DataGroup.
DATAPROGRAM	Parola chiave per la definizione delle variabili statiche nel DataGroup.
<num. programmi>	Numero di programmi (DataProgram) di cui è composto il DataGroup.
<nome variabile>	Nome della variabile statica del DataGroup.
<tipo>	Tipo della variabile statica e può essere: F (FLAG) B (BYTE) W (WORD) L (LONG) S (SINGLE)
STEP	Parola chiave per la definizione delle variabili indicizzate nel DataGroup.
<numero passi>	Numero di passi (Step) di cui è composto il DataGroup.
<nome variabile>	Nome della variabile indicizzata del DataGroup.

<tipo>	Tipo della variabile indicizzata e può essere: F (FLAG) B (BYTE) W (WORD) L (LONG) S (SINGLE)

La definizione del DATAGROUP è composta di 3 parti:
- una relativa alla definizione del nome del datagroup.
- una relativa all'impostazione del numero programmi e delle variabili statiche (inizia con la parola chiave DATAPROGRAM).
- una relativa all'impostazione del numero di passi di programma e delle variabili indicizzate (inizia con la parola chiave STEP).

Il nome datagroup segue tutte le regole generali finora incontrate per la sintassi dei nomi variabili.

Il numero programmi va scritto in forma numerica, oppure tramite l'ausilio di costanti e deve essere diverso da zero; il numero massimo di programmi è 65534.

Il numero passi va scritto in forma numerica, oppure tramite l'ausilio di costanti e deve essere diverso da zero. Il numero massimo di passi è 65534.

La sottosezione DATAPROGRAM è obbligatoria, mentre quella STEP è opzionale.

Non è possibile dichiarare una sezione STEP senza dichiarare almeno una variabile indicizzata. È possibile dichiarare una sezione STEP senza dichiarare una sezione DATAPROGRAM. Non è possibile dichiarare una sezione DATAPROGRAM senza dichiarare almeno una variabile statica. È possibile dichiarare una sezione DATAPROGRAM senza dichiarare una sezione STEP. In un DATAGROUP tutte le variabili, sia statiche che indicizzate, sono ritentive.

Per calcolare l'occupazione in memoria totale del DATAGROUP si deve tener presente che ogni variabile inserita nel datagroup occupata 4 byte (qualsiasi sia il tipo scelto per la variabile stessa). Quindi l'occupazione in byte è pari a:

(N.Programmi x N.Variabili statiche x 4) + (N.Programmi x N.Passi x N.Variabili indicizzate x 4).

0.2.14.1 Esempio: sviluppo Firstapp.qm4

Quanto descritto in questo esempio deve essere inserito nel progetto Firstapp.qm4 in modo da poter mettere immediatamente in pratica quanto appreso.

Nell'unità di configurazione (firstapp), in seguito alla sezione di dichiarazione degli INPUT ed OUTPUT inseriamo le dichiarazioni del gruppo DATAGROUP:

```
;---------------------------------------------------------------------
; Definizione del DATAGROUP
;---------------------------------------------------------------------
DATAGROUP
  dMioDataGrp           ;Nome identificativo del
                        ;Datagroup.
  DATAPROGRAM
    DIM_PROG            ;Numero dei programmi.
    ;--------------Definizione variabili statiche------------------
    dsfStat1 F          ;variabile statica di tipo Flag.
    dswStat2 W          ;Variabile statica di tipo Word.
  STEP
    DIM_STEP            ; numero di passi per programma
    ;----------Definisce le variabili di ciascun passo-------------
    ddbDin1 B           ;Variabile dinamica di tipo Byte.
    ddlDin2 L           ;Variabile dinamica di tipo long.
```

0.2.15 Utilizzo variabili DATAGROUP

Una variabile DataGroup (static o index), può essere utilizzata sia a destra che a sinistra di un'assegnazione o dentro un'espressione.

Sintassi nel caso di variabile statiche:

```
< Nome variabile Static > < [ num_prog ] >
```

Sintassi nel caso di variabile indicizzate:

```
< Nome variabile Index > < [ num_prog, num_step ] >
```

num_prog e *num_step* possono essere un numero (non SINGLE), una costante, una variabile o un'espressione complessa. Nel caso di numero o di costante, viene eseguito il controllo durante la compilazione che l'indice non superi la dimensione massima dichiarata in configurazione (rispettivamente per il numero programmi e il numero step); gli altri indici - *num_prog e num_step* - partono (in valore) da uno.

Un esempio di utilizzo dei datagroups viene presentato nell'esempio dell'istruzione FOR / NEXT.

0.2.16 Sezione BUS

La sezione BUS nell'unità di configurazione è indispensabile per dichiarare quali siano le dotazioni hardware che il programmatore ha a disposizione.
In questa sezione si dovrà indicare quale è la CPU utilizzata e inserita nello slot 1 e quali siano le altre schede negli altri slots. Ogni scheda è individuata da una parola chiave che ne identifica tipologia di hardware e, nel caso di schede intelligenti, versione del firmware; le parole chiavi sono reperibili nei manuali installazione e manutenzione dell'hardware utilizzato.

0.2.16.1 Esempio: sviluppo Firstapp.qm4

Nell'unità di configurazione (firstapp), in seguito alla sezione di dichiarazione dei DATAGROUP inseriamo le dichiarazioni relative alla configurazione del BUS:

```
;---------------------------------------------------------------------
; Configurazione BUS
;---------------------------------------------------------------------
BUS
  1 1CPUB  02     ;Scheda CPUB-02 installata (slot 1).
  2 1MIXA  00     ;Scheda MIX installata (slot 2).
  3 . .           ;Lo slot 3 è il primo libero.
```

Dopo aver configurato in questo modo il BUS è possibile compilare il progetto. Se ci sono errori nella compilazione verrà visualizzato un messaggio di errore che darà una descrizione sommaria del tipo di errore, indicando il file, riga e colonna in cui è localizzato.
Una volta compilato ed eseguito correttamente il download del progetto è possibile visualizzare, con i comandi del menu monitor, tutte le variabili introdotte, compresi gli elementi degli arrays e le variabili dei datagroup; con un doppio clic sul nome della variabile appare una piccola finestra per l'inserzione del valore.
Immettendo un valore diverso da zero in una variabile global, quando si mette in stato di stop la CPU questa variabile perde il valore impostato e si azzera. Ora abbiamo a disposizione una serie di variabili su cui sperimentare le istruzioni QCL descritte in seguito.

Se non si ha a disposizione l'hardware indicato nella configurazione del bus, dovranno essere inserite le parole chiave relative alle schede in possesso. È chiaro che per il corretto funzionamento dell'esempio Firstapp.qm4 è necessario che nello slot 2 venga inserito un hardware che metta a disposizione ingressi ed uscite digitali.

0.3 Simboli Speciali

Vi sono dei simboli che sono dichiarabili esclusivamente nello slot 1 (CPU) e che servono per leggere gli ingressi in interruzione e per i device simulati. Di seguito è riportata una tabella riepilogativa per questi simboli:

Simbolo	Tipo segnale	Dimensione	Accesso
INT01	Input	Flag	Read
INT02	Input	Flag	Read
INT03	Input	Flag	Read
INT04	Input	Flag	Read
INT05	Input	Flag	Read
INT06	Input	Flag	Read
INT07	Input	Flag	Read
INT08	Input	Flag	Read
CNT01	Conteggio	Word	Read
CNT02	Conteggio	Word	Read
CNT03	Conteggio	Word	Read
CNT04	Conteggio	Word	Read
CNT05	Conteggio	Word	Read
AN01	DAC	Word	Write
AN02	DAC	Word	Write
AN03	DAC	Word	Write
AN04	DAC	Word	Write
AN05	DAC	Word	Write

Con il simbolo speciale INTnn è possibile dichiarare un ingresso fittizio che legge la linea di interrupt in ingresso alla CPU. Un esempio di utilizzo di questo simbolo speciale è:

```
;---------------------------------------------------------------------
; Definizione variabili INPUT ed OUTPUT
;---------------------------------------------------------------------
INPUT
   ifInter01     F     1.INT01
   ifInter02     F     1.INT02
   ifInter05     F     1.INT05
   ifInter06     F     1.INT06
```

Gli altri simboli della tabella sono utilizzati per creare dei device simulati. Utilizzando i simboli CNTnn in una dichiarazione di un normale device il conteggio che il device acquisisce non è reale, ma viene simulato dalla CPU stessa. Se per esempio vengono dichiarati i seguenti device (si veda dichiarazione dei device nei relativi manuali):

```
;-------------------------------------------
; INTDEVICE Declaration
;-------------------------------------------
INTDEVICE
eaSim EANPOS     2     1.CNT01   X   X.X   X.X
cnSim COUNTER3   4     1.CNT01   X   X.X   X.X
```

Il device EANPOS, che è un posizionatore analogico, genererà un profilo ideale della posizione nel tempo e il device COUNTER3 leggerà semplicemente la posizione di un asse dall'ingresso di conteggio virtuale 1.CNT01.
Se inoltre il device EANPOS viene dichiarato nel seguente modo:

```
;-------------------------------------------
; INTDEVICE Declaration
;-------------------------------------------
INTDEVICE
eaSim EANPOS     2     1.CNT01   X   X.X   1.AN01
```

la CPU genera, oltre al profilo di posizione nel tempo, anche l'andamento di una tensione virtuale nel tempo per realizzare tale posizionamento. L'andamento della tensione nel tempo si può leggere da un apposito parametro del device EANPOS.

0.4 Istruzioni QCL

Il QCL (QEM Control Language) è un linguaggio nato appositamente per la programmazione del sistema QMOVE. Le caratteristiche principali del QCL sono la semplicità (poche ma potenti istruzioni) la facilità d'uso (somiglianza con il linguaggio BASIC) e l'orientamento verso l'automazione industriale grazie alle istruzioni appositamente studiate per il controllo assi.

0.4.1 Gli operatori del QCL

Nel linguaggio QCL vengono messi a disposizione tutti gli operatori elementari per la manipolazione dei dati.

0.4.1.1 Operatore di assegnamento

L'operatore di assegnamento può essere utilizzato all'interno di una qualunque espressione e può agire su tutte le variabili, sia elementari che appartenenti a DataGroup o a parametri Device. La forma generica di un'istruzione di assegnamento è la seguente:
nome variabile = <espressione>
Se la dimensione della variabile "nome variabile" è inferiore alla dimensione del risultato dell'espressione il valore viene troncato, secondo le regole della conversione di tipo (vedi appendice dedicata).

0.4.1.2 Operatori aritmetici

Negli esempi il valore 1 indica semplicemente che la variabile è diversa da 0.

Gli operatori aritmetici sono somma (+), sottrazione (-), moltiplicazione (*), divisione (/) ed il resto della divisione (%). Questi operatori agiscono su tutti i tipi di dato e sulle loro combinazioni (compresi i bit).
Esempi

```
sbVar01 = slVar02 + swVar03
slVarxy = glVarA * glVarB
sl01 = ss02 % ss03
```

0.4.1.3 Operatori logici

Gli operatori logici (AND, OR e NOT) possono agire su tutte le variabili tranne quelle a singola precisione. La loro funzione è riassunta dalla seguente tabella.

a b	a AND b	a OR b	NOT a
0 0	0	0	1
0 1	0	1	1
1 0	0	1	0
1 1	1	1	0

Esempi

```
glBit01 = glBit02 AND gfBit03
glBitA = NOT glBitB
```

0.4.1.4 Operatori binari

Gli operatori binari (ANDB, ORB, NOTB, XORB) agiscono su tutti i tipi di variabili compresi i SINGLE (previa conversione automatica in intero). Questi operatori realizzano la stessa tabella di verità che regola le equivalenti operazioni logiche, con la differenza che agiscono sui singoli bit. La tabella di verità dell'operatore OR esclusivo (XOR) è la seguente.

$(a)_{10}$	$(b)_{10}$	$(a)_2$	$(b)_2$	a ANDB b	a ORB b	NOTBa	a XORB b
3	2	011	010	010	011	100	001
7	5	111	101	101	111	000	010

Esempio

```
swVar01 = gwVar02 ANDB gwVar03
```

0.4.1.5 Operatori relazionali

Permettono di valutare le relazioni che intercorrono tra i valori. Il linguaggio QCL definisce cinque tipi di operatori relazionali.

EQ	(equal)	uguale	=
LT	(lower than)	minore	<
LE	(lower equal)	minore uguale	?
GT	(greater than)	maggiore	>
GE	(greater equal)	maggiore uguale	?

Esempio

```
gfBit01 = glCont GE glAsse01
```

0.4.1.6 Gli operatori di livello di precedenza

Sono le parentesi rotonde "()" che permettono di modificare il livello di precedenza degli operatori sopradescritti.
Esempio AA = [(A + B)/C] - (D x E)

```
gwVarAA = ((gwVarA + gwVarB) / glVarC) - (swVarD * glVarE)
```

0.4.1.7 Livelli di priorità degli operatori del QCL

Il livello di priorità degli operatori QCL ricalca quello di tutti i linguaggi di programmazione; Le priorità sono riassunte in tabella.

Priorità massima ( ) Funzioni matematiche e Funzioni trigonometriche NEG NOT NOTB * / % + - LT GT LE GE EQ AND ANDB OR ORB XORB = **Priorità minima**

0.4.2 Istruzioni e strutture per il controllo del flusso

Sono tutte quelle istruzioni che permettono di modificare il flusso di esecuzione.

0.4.2.1 IF / ELSE / ENDIF

È la tipica istruzione condizionale; se la condizione è vera, valore booleano 1, viene eseguito il blocco istruzioni 1 altrimenti viene eseguito il blocco istruzioni 2. Un blocco istruzioni può essere costituito da un insieme di istruzioni ma anche da un'istruzione sola; inoltre la sezione ELSE è opzionale.

```
IF <condizione>
<blocco istruzioni 1>
[ELSE
<blocco istruzioni 2>]
ENDIF
```

0.4.2.2 Esempio: sviluppo Firstapp.qm4

Modifichiamo il task_00 in questo modo:

```
MAIN:
IF tTempo                   ;Se il timer è scaduto (all'inizio dell'esecuzione è sempre scaduto).
tTempo = TM_SECONDO         ;Ricarica il timer il quale comincia subito a scaricarsi.
slProva = slProva + 1       ;Incrementa di 1 la varibile slProva.
ENDIF
```

```
  WAIT 1
  JUMP MAIN
END
```

In questo esempio la variabile *slProva* viene incrementata di uno ogni volta che scade il timer il quale, ogni volta viene ricaricato con il valore della costante TM_SECONDO (1000 ms = 1 s). Si osservi che se si mette in STOP l'applicativo e poi lo si fa ripartire, il valore di *slProva* non si azzera. Questa è la caratteristica di ritentività delle variabili SYSTEM già accennata.

Esempio di ciclo IF nidificato
Modifichiamo l'esempio appena introdotto nel modo seguente:

```
MAIN:
;--------------Gestione di un orologio di sistema-------------------
IF tSecondi                     ;Se il timer e' terminato.
  tSecondi = TM_SECONDO         ;Ricarica il timer.
  IF sbSecondi LT 59            ;Se i secondi sono minori di 59.
    sbSecondi = sbSecondi + 1   ;Incremento secondi.
  ELSE                          ;Altrimenti.
    sbSecondi = 0               ;Azzero i secondi.
    IF sbMinuti LT 59           ;Se i minuti sono minori di 59.
      sbMinuti = sbMinuti + 1   ;Incremento i minuti.
    ELSE                        ;Altrimenti.
        sbMinuti = 0            ;Azzero i minuti.
        IF swOre LT 167         ;Se le ore sono minori di 167.
          swOre = swOre + 1     ;Incremento le ore.
  ELSE
          swOre = 0             ;È passato una settimana e azzero le ore.
        ENDIF
ENDIF
  ENDIF
ENDIF
  WAIT 1
  JUMP MAIN
END
```

Abbiamo realizzato così un orologio di sistema con tre variabili system: una per i secondi, una per i minuti e una per le ore. Si è utilizzata una variabile di tipo Byte per i secondi e per i minuti dato che il valore che esse assumono non esce mai dal range [-128, 127], mentre per la variabile swOre si è utilizzato una variabile di tipo Word dato che può arrivare fino al valore 167, all'interno del range [-32768, 32767], che è il numero di ore in una settimana.

0.4.2.3 FOR / NEXT

È un ciclo iterativo che serve normalmente per eseguire un certo numero di volte una o un blocco di istruzioni. È possibile uscire dal ciclo FOR-NEXT prima che diventi falsa la condizione di ciclo tramite l'istruzione BREAK; quando quest'ultima viene incontrata viene eseguito un salto alla prima istruzione successiva al NEXT.

FOR (<inizializzazione contatore>, <condizione di ciclo >, <incremento contatore>)
<blocco istruzioni >
NEXT

dove:

<inizializzazione contatore>	è una espressione eseguita all'avvio del ciclo e quindi una sola volta.
<condizione di ciclo >	è una espressione condizionale che, se falsa, determina l'uscita dal ciclo, cioè fa eseguire un salto alla prima istruzione successiva al NEXT.
<incremento contatore>	è un valore numerico che viene sommato al contatore al termine di ogni esecuzione del <blocco istruzioni>.

0.4.2.4 Esempio: sviluppo Firstapp.qm4

Si aggiunge al task_00 il seguente ciclo di FOR/NEXT:

```
MAIN:
;----------------Gestione di un orologio di sistema-----------------
IF ....
.......
ENDIF
;---------------------------------------------------------------------
;----Caricamento automatico di alcuni valori in array e datagroup---F O R
(gbTuoByte = 1, gbTuoByte LE 10, 1)         ;Per gbByte da 1 fino a 10 con incremento 1.
arwMioArray[gbTuoByte] = POW(3, gbTuoByte)  ; All'elemento gbTuoByte-esimo dell'array arwMioByte viene assegnato il valore 3
elevato alla gbTuoByte.
  dswStat2[gbTuoByte] = 1                   ;Inserisce 1 in tutte le variabili statiche.
  ddbDin1[gbTuoByte,1] = 2                  ;Inserisce 2 in tutte le variabili dinamiche degli step 1 di ogni programma. NEXT
  WAIT 1
  JUMP MAIN
END
```

Con il ciclo FOR / NEXT visto si riempie l'array global *arwMioArray* delle prime 10 potenze di 3. Da qview è possibile vedere i valori introdotti nell'array selezionando la voce ***Monitor > Arrays***. La finestra che appare è la lista di tutti gli array presenti nel progetto. Scegliere ***arwMioArray*** e appare una finestra in cui è rappresentata la lista di tutti i valori presenti nell'array. Si osservi che l'ultimo valore, corrispondente a gbTuoByte = 10, è un valore non esatto dato che 310=59049 (non -6487 come viene visualizzato) che non può essere contenuto in una variabile di tipo word e quindi si verifica un *overflow*.
Per vedere i valori contenuti nel datagroup si segue lo stesso metodo solo che si seleziona ***Monitor > Data Group***.

0.4.2.5 WHILE / ENDWHILE

È un ciclo iterativo che serve per eseguire una o un blocco di istruzioni fintantoché non si verifica una condizione.
È possibile uscire dal ciclo WHILE-ENDWHILE prima che diventi falsa la condizione di ciclo tramite l'istruzione BREAK; quando quest'ultima viene incontrata viene eseguito un salto alla prima istruzione successiva al ENDWHILE.

WHILE (<condizione di ciclo >) <blocco istruzioni > ENDWHILE

dove:

<condizione di ciclo >	è una espressione condizionale che, se falsa, determina l'uscita dal ciclo, cioè fa eseguire un salto alla prima istruzione successiva al 'ENDWHILE'.

0.4.2.6 Esempio: sviluppo Firstapp.qm4

Questi due cicli hanno lo scopo di poter impostare il valore delle variabili sbMinuti e swOre tenendo attivati due diversi ingressi ifSetUpMin e ifSetUpOre, rispettivamente. Si osservi che all'interno di ogni while è stato inserito un timer di ritardo, tDelay, per fare in modo che l'incremento della variabile sia più lento per dare modo all'utente di disattivare l'ingresso quando la variabile assume il valore voluto. Naturalmente il valore da assegnare ai minuti o alle ore è limitato e quindi viene aggiunto un IF all'interno del while per controllare questo limite.

Si aggiungono al task_00 i due cicli WHILE / ENDWHILE:

```
MAIN:
;---------------Gestione di un orologio di sistema-----------------
IF ...
.....
ENDIF
;----Caricamento automatico di alcuni valori in array e datagroup---FOR ...
.......
NEXT
;--------------Impostazione delle ore dell'orologio ----------------
WHILE (ifSetUpOre)             ;Finchè l'ingresso è attivo.
  IF tDelay                    ;Se il tempo di ritardo è scaduto.
    tDelay = 500               ;Ricarico il tempo di ritardo.
    swOre = swOre + 1          ;Incremento il contatore delle ore.
    IF swOre GE 168            ;Se il numero delle ore supera 167.
      swOre = 0                ;Azzera il contaore.
    ENDIF
  ENDIF
  WAIT 1
ENDWHILE
;--------------Impostazione dei minuti dell'orologio ---------------
WHILE (ifSetUpMin)             ;Finchè l'ingresso è attivo.
  IF tDelay                    ;Se il tempo di ritardo è scaduto.
    tDelay = 500               ;Ricarico il tempo di ritardo.
    sbMinuti = sbMinuti + 1    ;Incremento il contatore dei minuti.
IF sbMinuti GE 60              ;Se il numero dei minuti supera 59.
      sbMinuti = 0             ;Azzera il contaminuti.
  ENDIF
  ENDIF
  WAIT 1
ENDWHILE
WAIT 1
  JUMP MAIN
END
```

0.4.2.7 WAIT

Attende il verificarsi di una condizione o di un evento. L'istruzione WAIT è una istruzione molto importante poichè quando viene eseguita e la <condizione> è falsa, l'esecuzione delle istruzioni passa al task successivo. È comunque necessario fare una precisazione in questo contesto; infatti l'esecuzione dell'istruzione WAIT comporta comunque la cessione del controllo al task successivo quando viene incontrata la prima volta, senza che la condizione sia controllata. Quando il controllo del flusso istruzioni ritorna al task in questione viene rieseguita l'istruzione WAIT e quindi controllata la condizione. Per maggiori informazioni fare riferimento al capitolo dedicato al multitasking. La sintassi è:

```
WAIT <condizione>
```

0.4.2.8 JUMP

Esegue un salto incondizionato all'istruzione locata all'etichetta specificata. La sintassi è:

```
JUMP <etichetta>
```

Esempio
L'esempio classico per l'istruzione JUMP è quello che viene ripetuto in ogni task per saltare dalla fine del codice all'inizio, cioè all'etichetta MAIN. Questa istruzione per il controllo di flusso può essere utilizzata anche in altri punti del task. JUMP è un salto incondizionato, si può renderlo condizionato in combinazione con un IF con lo scopo di deviare il flusso di codice in base al verificarsi di particolari condizioni.

0.4.2.9 ETICHETTE

Tramite le etichette è possibile localizzare determinati punti nella sequenza delle istruzioni dell'applicativo. Vengono utilizzate per modificare il flusso di esecuzione delle istruzioni (comando JUMP del QCL). Quando viene inserita un'etichetta nelle istruzioni è necessario che sia seguita dai due punti ":", mentre se si fa riferimento all'etichetta stessa si deve scrivere il nome senza i due punti.
Esempio jump incondizionato

```
Etichetta1:                   ;Riga di destinazione del jump.
  <istruzione>
  <istruzione>
  ..................
JUMP Etichetta1               ;Istruzione di ritorno all'etichetta.
```

Esempio jump condizionato

```
Etichetta1:                   ;Riga di destinazione del jump.
  <istruzione>
  IF <condizione>             ;Se la condizione è vera.
    JUMP Etichetta 1          ;Istruzione di ritorno all'etichetta.
  ENDIF
  ...................
```

0.4.2.10 SUBROUTINE

Quando, all'interno di uno stesso modulo, vi è una parte di codice ripetuta più volte è utile definire una subroutine. L'esecuzione del codice in essa contenuto avverrà ogni qualvolta viene incontrata una istruzione di chiamata a subroutine (CALL). La subroutine viene identificata da un nome preceduto dalla parola chiave SUB e deve terminare con la parola chiave ENDSUB. Vedi anche istruzione CALL.

0.4.2.11 CALL

Call to Subroutine: esegue un salto incondizionato alla prima istruzione della subroutine identificata dal nome. Al termine della subroutine (identificata da ENDSUB) il flusso del programma continua dalla istruzione successiva alla CALL.

```
CALL <nome_subroutine>
```

0.4.2.12 Esempio: sviluppo Firstapp.qm4

Quanto descritto in questo esempio deve essere inserito nel progetto Firstapp.qm4 in modo da poter mettere immediatamente in pratica quanto appreso.
Queste righe di codice vanno poste oltre la parola chiave END, cioè oltre la fine del task.

Introduciamo una subroutine per calcolare la lunghezza di una diagonale di un quadrato data la lunghezza di un lato e viene inserito dopo l' END del Task_00.

L'esempio a seguire illustra il richiamo della subroutine:

```
MAIN:
;---------------Gestione di un orologio di sistema-----------------
IF ...
.....
ENDIF
;----Caricamento automatico di alcuni valori in array e datagroup---FOR ...
.......
NEXT
;---------------Impostazione delle ore dell'orologio---------------
WHILE ...
.........
ENDWHILE
;---------------Impostazione dei minuti dell'orologio--------------
WHILE...
.......
ENDWHILE
;---------------------Chiamata alla subroutine---------------------
glNostroLong = 4           ;Lato del quadrato.
CALL CALC_DIAG             ;Chiamata alla subroutine di calcolo
                           ;della diagonale.
  WAIT 1
  JUMP MAIN
END

;-------------Subroutine di calcolo della diagonale----------------
SUB CALC_DIAG
gsVostroSing = SQRT(2 * POW(glNostrolong,2))
ENDSUB
```

0.4.2.13 NOP

No operation; questa istruzione non ha nessun effetto sul programma e può essere utilizzata per introdurre dei ritardi.

0.4.3 Istruzioni dedicate alle uscite digitali

Sono le istruzioni che permettono di attivare o disattivare un'uscita digitale. L'attivazione e la disattivazione possono essere

eseguite anche tramite l'operatore di assegnamento (<nome uscita> = 1) ma le istruzioni dedicate sono più veloci.

0.4.3.1 SETOUT

Attiva una uscita digitale. La sintassi è la seguente:

```
SETOUT <nome uscita>
```

0.4.3.2 RESOUT

Disattiva una uscita digitale. La sintassi è la seguente:

```
RESOUT <nome uscita>
```

0.4.3.3 Esempio: sviluppo Firstapp.qm4

Si può osservare il commutare dell'uscita dall'accendersi e spegnersi del primo LED in alto a destra sul pannello frontale della scheda MIX. L'uscita ofUscita1 viene attivata con il comando SETOUT quando la variabile sbSecondi assume il valore 20 solo se l'ingresso ifAbilU1 e attivo. Tale uscita viene poi resettata quando sbSecondi assume il valore 40.

Introduciamo un esempio in cui vengono cambiate di stato un'uscita a seconda del valore di una variabile:

```
MAIN:
;---------------Gestione di un orologio di sistema-----------------
IF ...
......
ENDIF
;----Caricamento automatico di alcuni valori in array e datagroup---FOR ...
.......
NEXT
;---------------Impostazione delle ore dell'orologio---------------
WHILE ...
.........
ENDWHILE
;---------------Impostazione dei minuti dell'orologio--------------
WHILE...
........
ENDWHILE
;----------------------Chiamata alla subroutine--------------------
...
;----------------------Set o reset dell'uscita 1-------------------
IF sbSecondi EQ 20 AND ifAbilU1   ;Se sbSecondi è uguale a 20 e se ho
                                  ;l'abilitazione da ingresso.
  SETOUT ofUscita1                ;Mette in ON l'uscita.
ENDIF
IF sbSecondi EQ 40                ;Se slSecondi è uguale a 40.
  RESOUT ofUscita1                ;Mette in OFF l'uscita.
ENDIF

WAIT 1
  JUMP MAIN
END
```

0.4.4 Istruzioni di sistema

Sono le istruzioni che permettono il controllo dell'esecuzione dei tasks (unità contenenti il codice QCL o LADDER) del programma utente.

0.4.4.1 T_SUSPEND

Esegue la sospensione dell'esecuzione del task di programma specificato.
Tale istruzione può essere utilizzata da un task per bloccare l'esecuzione di un altro task oppure per bloccare l'esecuzione di se stesso. In questo caso bisogna ricordare che il task in questione, essendo sospeso, non può più riattivarsi ma tale operazione deve essere eseguita da uno degli altri task. Una particolare variabile di sistema (QMOVE:is_suspend) contiene il valore 1 se il task è sospeso. La sintassi è:
T_SUSPEND <nome_task>

0.4.4.2 T_RESUME

Esegue il ripristino dell'esecuzione delle istruzioni di un task sospeso. Tale istruzione fa riprendere l'esecuzione di un task sospeso dall'istruzione successiva a quella eseguita al momento della sospensione; T_RESUME produce degli effetti solamente se il task <nome_task> era sospeso. La sintassi è:
T_RESUME <nome_task>

0.4.4.3 T_RESTART

Esegue una “ripartenza” dell'esecuzione di un task di programma dalla prima istruzione. Tale istruzione non altera lo stato di sospensione del task in questione; ciò significa che se viene eseguita una T_RESTART in riferimento ad un task sospeso, quest'ultimo si predispone per l'esecuzione della prima istruzione ma rimane tuttavia sospeso fino alla sua attivazione con l'istruzione T_RESUME. La sintassi è:
T_RESTART <nome_task>

Per comprendere meglio il significato di queste instruzioni di sistema si veda il capitolo “Multitasking”.

0.4.5 Variabili di sistema

Il sistema QMOVE mette a disposizione delle variabili legate al sistema operativo interno; i nomi di queste variabili sono predefiniti, non possono cioè essere cambiati dall'utente; ciascuna di queste variabili deve essere preceduta dalla parola chiave QMOVE.

0.4.5.1 is_suspend

Fornisce lo stato di un task: 0 = task in running; 1 = task sospeso. La sintassi è:
QMOVE:is_suspend Nome task
Esempio:

```
QMOVE:is_suspend Task_00
```

A partire dalla build 022 del Qview 4.1 il compilatore supporta anche le seguenti variabili di sistema.

0.4.5.2 w_dog_task

Watchdog task. Indica che un task QCL è stato eseguito per più di 200 ms. È la stessa informazione che compare nel pannello “Monitor - CPU” alla voce “Watchdog Active”.
La sintassi è:
QMOVE:w_dog_task

0.4.5.3 time_task_lost

Task lost. Indica che un task a tempo ha perso un evento. È la stessa informazione che compare nel pannellino CPU alla voce “Time task Lost”.
La sintassi è:
QMOVE:time_task_lost

0.4.5.4 battery_low

Indica che la batteria è ha un livello di carica basso. Il led BATT lampeggia.
La sintassi è:
QMOVE:battery_low

0.4.5.5 battery_down

Indica che la batteria è scarica. Il led BATT è acceso.
La sintassi è:
QMOVE:battery_down

Esistono altre variabili di sistema che dipendono dal firmware della CPU utilizzata. Per accedere a queste variabili si utilizza la sintassi:
QMOVE:sys001
QMOVE:sys002
…
QMOVE:sys016
Il significato di queste variabili deve essere ricercato nella documentazione firmware della CPU.

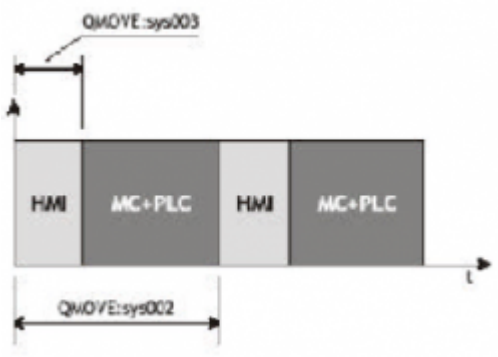
ESEMPIO:
Le seguenti tre variabili di sistema sono utilizzabili nel caso si stia utilizzando la versione integrata del Qmove (codice di ordinazione J1-255Dx).

Nel motion controller J1-255, un'unica CPU esegue sia il programma di controllo della macchina (Motion Controller + PLC) sviluppato con Qview, che il programma di interfaccia operatore (HMI) sviluppato con Qpaint. Per regolare in modo appropriato la suddivisione del tempo tra questi due compiti sono state predisposte 3 variabili di sistema, visualizzabili e modificabili dal programmatore nel programma applicativo sviluppato con Qview.
Esse sono:
QMOVE:sys001 Switch type change task CPU – Term (tipo Long)
QMOVE:sys002 Period rate (tipo Long)
QMOVE:sys003 Duty cycle rate (tipo Long)

Di seguito si descriveranno prima le due variabili QMOVE:sys002 e QMOVE:sys003 che sono legate tra di loro e quindi si concluderà il capitolo con la descrizione della variabile QMOVE:sys001.

0.4.5.6 QMOVE:sys002 (Period rate) e QMOVE:sys003 Duty cycle rate

Con la variabile QMOVE:sys002 si fissa il tempo che viene messo a disposizione della CPU per eseguire sia il programma MC+PLC (o parte di esso) che il programma HMI (o parte di esso). Questo tempo viene espresso in percentuale di 300 ms. Quindi se QMOVE:sys002 = 50% significa che si fissa una base tempo di 150ms.
La variabile QMOVE:sys003 serve per dividere il tempo fissato con QMOVE:sys002 in due parti: una destinata all'esecuzione del programma HMI e la rimanente per il programma MC+PLC. La variabile QMOVE:sys003 è espressa in percentuale del tempo espresso con la variabile QMOVE:sys002. Quindi, continuando l'esempio di prima, se Qmove:sys003 = 25% significa che nel Period rate di 150 ms ne sono previsti 37.5 ms per eseguire il programma HMI. Il tutto si può rappresentare con il seguente grafico temporale:
QMOVE:sys003



come si può osservare più piccola è la variabile QMOVE:sys002, più rapida sarà la risposta dell'interfaccia HMI nel senso che il programma MC+PLC verrà interrotto molte volte per gestire, per esempio, la visualizzazione sul display. D'altro canto questo potrebbe provocare dei rallentamenti dell'esecuzione del programma MC+PLC.
Inoltre si può osservare che variando la variabile QMOVE:sys003 si può regolare la quantità di risorse di calcolo da destinare alla gestione dell'HMI all'interno del periodo e quindi è possibile potenziare le prestazioni in velocità della parte HMI o della parte MC+PLC a seconda se QMOVE:sys003 assume valori alti o bassi, rispettivamente.
Per la variabile QMOVE:sys002 sono valori compresi tra 3 e 100%.
Per la variabile QMOVE:sys003 sono valori compresi tra 3 e 90%.

La sintassi di utilizzo di queste variabili è:
QMOVE:sys002
QMOVE:sys003

Esempio di utilizzo:
Qmove:sys002 = 50
Qmove:sys003 = 25

0.4.5.7 QMOVE:sys001 (Switch type change task CPU – Term)

Questa variabile permette di impostare una regolazione delle variabili QMOVE:sys002 e QMOVE:sys003 in modo automatico o manuale.
Può assumere i seguenti valori:

1: impostazione manuale delle QMOVE:sys002 e QMOVE:sys003 senza blocco delle unità speciali. Le impostazioni delle variabili sono demandate al programmatore il quale può decidere di impostarle a seconda delle prestazioni che vuole ottenere.
Con questa impostazione le unità speciali a interrupt e a tempo definite dal programmatore rimangono ad alta priorità e non possono essere interrotte o bloccate dall'esecuzione del programma HMI.

2: impostazione manuale delle QMOVE:sys002 e QMOVE:sys003 con blocco delle unità speciali. Le impostazioni delle variabili sono demandate al programmatore il quale può decidere di impostarle a seconda delle prestazioni che vuole ottenere.
Con questa impostazione le unità speciali a interrupt e a tempo definite dal programmatore possono essere interrotte o bloccate dall'esecuzione del programma HMI.

3: calcolo automatico delle QMOVE:sys002 e QMOVE:sys003 senza blocco delle unità speciali (valore di default). Con questa impostazione il sistema fa in modo che la variabile QMOVE:sys002 sia il più possibile intorno al valore 66% (200ms). Il valore della variabile QMOVE:sys003 varia a seconda della complessità della pagina da visualizzare. In una pagina con molti oggetti dinamici questa variabile può assumere valori alti, mentre in una pagina con pochi oggetti dinamici essa assume valori bassi.
Con questa impostazione le unità speciali a interrupt e a tempo definite dal programmatore rimangono ad alta priorità e non possono essere interrotte o bloccate dall'esecuzione del programma HMI.

4: calcolo automatico delle QMOVE:sys002 e QMOVE:sys003 con blocco delle unità speciali. Con questa impostazione il sistema fa in modo che la variabile QMOVE:sys002 sia il più possibile intorno al valore 66% (200ms). Il valore della variabile QMOVE:sys003 varia a seconda della complessità della pagina da visualizzare. In una pagina con molti oggetti dinamici questa variabile può assumere valori alti, mentre in una pagina con pochi oggetti dinamici essa assume valori bassi.
Con questa impostazione le unità speciali a interrupt e a tempo definite dal programmatore possono essere interrotte o bloccate dall'esecuzione del programma HMI. La sintassi di utilizzo di questa variabile è:
QMOVE:sys001
Esempio di utilizzo:
Qmove:sys001 = 1

0.4.6 Funzioni matematiche

Elevamento a potenza

Esegue l'elevamento alla potenza della base. La sintassi è:
POW(base, esponente).

Esempio: gsMaxVal = 2^{gwNbit}

```
gsMaxVal = POW(2,gwNbit)
```

0.4.6.1 Radice quadrata

Calcola la radice quadrata dell'argomento dato. La sintassi è:
SQRT(radicando)
Esempio: gsIpot = ?(lato12 + lato22)

```
gsIpot = SQRT(POW(gslato1,2)+POW(gslato2,2))
```

0.4.6.2 Logaritmo naturale

Calcola il logaritmo naturale dell'argomento dato. La sintassi è:
LN(val)
Esempio: gsValue = ln gsMaximum

```
gsValue = LN(gsMaximun)
```

0.4.6.3 Esponenziale

Esegue l'elevamento a potenza del numero di Nepero. La sintassi è:
EXP(esponente)
Esempio: gsA = e2

```
gsA =EXP(2)
```

0.4.6.4 Valore assoluto

Ritorna il valore assoluto dell'argomento. La sintassi è:
ABS(Value)
Esempio: glModulo = I glValore I

```
glModulo = ABS(glValore)
```

0.4.7 Funzioni trigonometriche

0.4.7.1 Seno

Calcola il seno di un angolo espresso in radianti. La sintassi è:
SIN(angle)
Esempio: gsYPos = ssRadius x SIN gsalpha

```
gsYPos = ssRadius * SIN(gsalpha)
```

0.4.7.2 Coseno

Calcola il coseno di un angolo espresso in radianti. La sintassi è:
COS(angle)
Esempio: gsXPos = ssRadius x COS gsalpha

```
gsXPos = ssRadius * COS(gsalpha)
```

0.4.7.3 Tangente

Calcola la tangente di un angolo espresso in radianti. La sintassi è:
TAN(angle)
Esempio: gsMyVal = TAN gsalpha

```
gsMyVal = TAN(gsalpha)
```

0.4.7.4 Cotangente

Calcola la cotangente di un angolo espresso in radianti. La sintassi è:
COT(angle)
Esempio: gsMyVal = COTG gsalpha

```
gsMyVal = COTG(gsalpha)
```

0.4.7.5 Arcoseno

Calcola l'angolo, espresso in radianti, il cui seno è uguale all'argomento. La sintassi è:
ASIN(arc)
Esempio: gsAngolo = ASIN Arco

```
gsAngolo = ASIN(Arco)
```

0.4.7.6 Arcocoseno

Calcola l'angolo, espresso in radianti, il cui coseno è uguale all'argomento. La sintassi è:
ACOS(arc)
Esempio: gsAngolo = ACOS gsArco

```
gsAngolo = ACOS(gsArco)
```

0.4.7.7 Arcotangente

Calcola l'angolo, esresso in radianti, la cui tangente è uguale all'argomento. La sintassi è:
ATAN(arc)
Esempio: gsMyVal = ATAN Arco

```
gsMyVal = ATAN(Arco)
```

0.4.7.8 Avvertenza

In generale, nelle funzioni trigonometriche, gli angoli vengono espressi in radianti. Essendo p un numero irrazionale (non finito) e potendolo rappresentare con una precisione di 6 cifre dopo il punto decimale, si introduce una approssimazione nei calcoli trigonometrici.
Altro limite è dato dai calcoli trigonometrici che danno come risultato un numero infinito, che non può sicuramente essere rappresentato con un numero floating point a singola precisione (7 cifre). Per esempio il risultato della tangente di p/2 è un numero molto grande in modulo con segno negativo, risultato che evidentemente si allontana da +¥ (risultato corretto).

ESEMPIO: tang p/2 = tang 1.570796371 = -22877332 ;
arctg -22877332 = -1.570796371 = - p/2 (!!)

Non è possibile calcolare la tangente di ?/2 con la formula:
tang p/2 = (sen p/2) / (cos p/2)
in quanto viene eseguita una divisione per zero.

0.5 Funzioni di libreria QCL

Una funzione QCL è una parte di codice che permette di risolvere particolari problemi, di eseguire elaborazioni dati, di fornire specifiche funzionalità ad un applicativo QMove. Per poter utilizzare una funzione QCL in un codice task è sufficiente richiamarla passandole gli argomenti necessari, come si trova descritto in seguito.
Questo modo di procedere comporta alcuni vantaggi: - Si trovano parti di codice già scritte, collaudate, che risolvono in maniera ottimizzata i piú frequenti problemi che si affrontano nella stesura di un applicativo.
- Offrono la possibilità di riutilizzare quante volte si vuole la stessa funzione, senza doversi preoccupare di riscriverla ogni volta.
- Esiste la possibilità di aggiornare semplicemente la libreria man mano che questa viene dotata di nuove funzioni, mantenendo una assoluta compatibilità con gli applicativi già sviluppati.
La lista delle funzioni QCL disponibili ed la relativa modalità di utilizzo è disponibile nel menù ***Help > Functions info***, richiamabile in qualsiasi momento dall'ambiente di sviluppo.

Se indichiamo con FuncQCL01(...) una qualsiasi funzione della libreria, la sintassi corretta per richiamarla è la seguente:

```
FuncQcl01(...)
```

Non restituendo valori, le funzioni non possono essere utilizzate a destra di un assegnazione o come parametri in un istruzione IF, FOR o WHILE.

```
slType01 = FuncQCL01(...)                       !ERRORE!
IF (FuncQCl01(...) AND ...)                     !ERRORE!
FOR (sInt01 = 1, sInt01 LT FuncQCL01(...), ...) !ERRORE!
WHILE (FuncQCL01(...) LT 24)                    !ERRORE!
```

Anche se una funzione QCL non restituisce alcun valore, è tuttavia possibile che una funzione QCL imposti dei valori a parametri passati come argomento e quindi in pratica è come se si avessero piú valori di ritorno.

Vediamo alcuni semplici esempi:

AC10AvergArr (Myarray, val_medio, ok_calc)

La funzione calcola il valore medio di un array passato come argomento. In questo caso *MyArray* è un parametro di ingresso e si vuol calcolarne il valore medio.
val_medio e *ok_calc* sono parametri di uscita e sono rispettivamente il valore medio calcolato dalla funzione e un flag indicante il calcolo completato. Quando *ok_calc* assume valore 1 significa che la funzione ha terminato i suoi calcoli e quindi è possibile recuperare tale valore leggendo la variabile *val_medio*.

Per ciascuna funzione è poi indicato in quale parte dell'unità chiamarla; infatti vi sono funzioni, che proprio per le operazioni che devono svolgere, vanno chiamate in una parte ciclica del codice, mentre alcune altre possono essere chiamate anche in una parte non ciclica del codice. Con parte ciclica si intende la parte del codice QCL che viene eseguita ad ogni scansione del programma.

Con parte non ciclica si intende la parte di codice che non viene eseguita ciclicamente perchè mancano determinate condizioni (per esempio se il codice è contenuto in una istruzione IF la cui condizione non è vera).
L'indicazione sul "dove" richiamare una funzione è riportato nella descrizione di ciascuna funzione (***Help > Functions info***).
Un'altra caratteristica da tenere presente per alcune funzioni è il fatto che, quando sono necessarie operazioni che impiegano molto tempo, la funzione stessa si preoccupa di eseguire un'istruzione di WAIT al suo interno, ogni 180 millisecondi. Questo per impedire il "blocco" dell'esecuzione del task QCL dovuto a compiti particolarmente gravosi. Solitamente in funzioni di questo tipo è presente un argomento di tipo flag che puó essere monitorato per verificare quando la funzione ha terminato il suo compito.
Nell'help della funzione utilizzata viene indicato chiaramente il tipo degli argomenti che la funzione si aspetta. Se questo tipo non viene rispettato dal programmatore viene generato un errore in fase di compilazione del progetto.

0.5.1 Note per l'utente

Il programmatore non deve dichiarare nel file di configurazione le variabili utilizzate nelle funzioni delle librerie.

1) L'utilizzo delle funzioni puó portare ad una modifica del file simboli, e quindi la necessità di riallinearli nel terminale operatore (se utilizzato).
2) Le variabili interne utilizzate dalle funzioni QCL assumono nomi che iniziano con

```
ZZ_ZZqem_ nnn
```

dove con nnn si indica un numero interno di riferimento. L'uso di variabili con tale nome da parte del programmatore è sconsigliato, in quanto potrebbero comportare un malfunzionamento imprevedibile.
3) Le funzioni sono suddivise in più livelli. Le funzioni di livello 0 (zero) sono installate al momento dell'installazione del Qview. Le altre funzioni di livello superiore devono essere installate appositamente tramite ***Tools > Upgrade QCL Functions Library….*** I file di upgrade per le funzioni di livello superiore devono essere richiesti direttamente a QEM srl. In ***Help - Functions*** info è presente comunque l'help completo per tutte le funzioni realizzate al momento dell'installazione del Qview.

0.5.2 Funzioni utente

Il programmatore ha la possibilità di creare delle funzioni proprie le quali si collocano nella categoria delle funzioni utente. Le motivazioni che potrebbero portare un programmatore a realizzare una funzione sono le seguenti:
1) Realizzare delle parti di codice parametriche che vengono richiamate quando serve e che rendono il programma più leggibile e ordinato;
2) Realizzare delle parti di codice parametriche che possono essere riutilizzate in altri programmi;
3) Criptare delle parti di codice che sono contenute nella funzione. Il programmatore può generare queste funzioni, diffonderne l'utilizzo, ma mantenere per se il codice sorgente.

Si veda in appendice un apposito capitolo che descrive come generare le funzioni utente.

0.5.2.1 Esempio di utilizzo delle funzioni QCL

Supponiamo di utilizzare la funzione di calcolo della media aritmetica dei valori contenuti in un array (AC10AvergArr), e di averlo dichiarato nel file di configurazione nel modo seguente:

```
GLOBAL
  glAverage L           ; Variabile per contenere il risultato della funzione
  gfDone F              ; Flag per indicare che la funzione ha completato il calcolo
ARRSYS
  aslMyArray 10 L       ; Array System di 10 elementi di tipo long
```

In un task viene richiamata la funzione in questo modo:

```
MAIN:
  IF NOT gfDone
    AC10AvergArr(aslMyArray, glAverage, gfDone)
  ENDIF
  WAIT 1
  JUMP MAIN
END
```

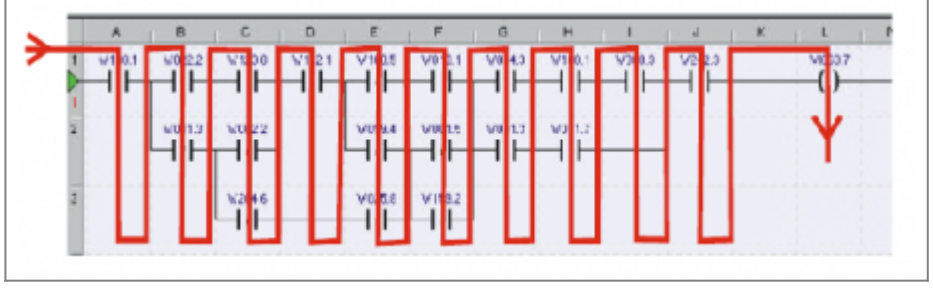
0.6 Editor Ladder

Il Ladder (logica a contatti) è un linguaggio nato per la programmazione dei PLC, per cui è molto performante per la gestione di ingressi uscite e piccole sequenze.
Si dà per scontato che l'utilizzatore conosca il linguaggio Ladder IEC1131; in caso contrario si consiglia di integrare il presente manuale con un corso PLC generico.

0.6.1 L'esecuzione del rung

Nel linguaggio LADDER bisogna tener presente che i vari elementi sono analizzati, all'interno del rung, partendo dall'angolo in alto a sinistra e spostandosi dall'alto in basso e da sinistra a destra (Figura 1).

Figura 1: esecuzione del rung.

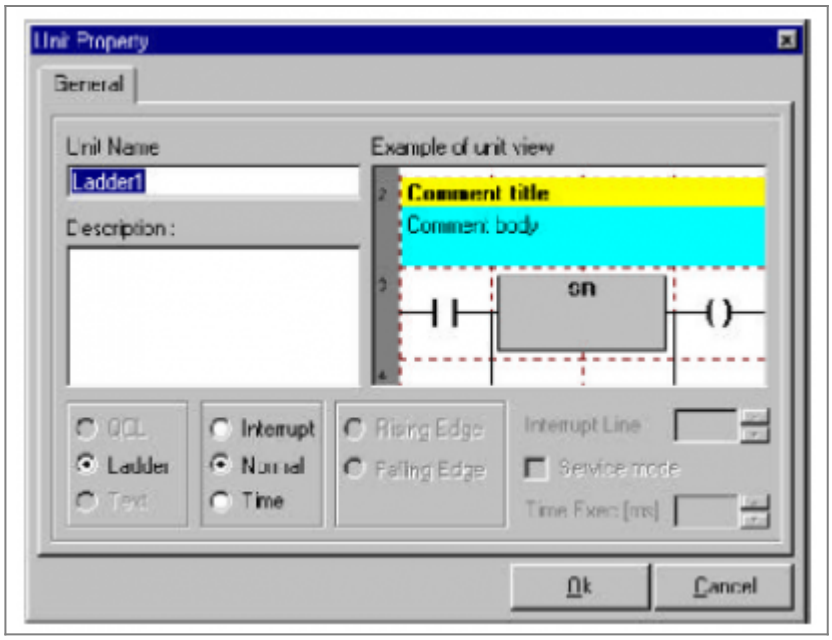


Nel caso si inserisca un elemento che occupa più celle (ad esempio un contatore), la cella che fa da riferimento per lo svolgimento del rung è quella in alto a sinistra.
Il nome delle variabili per il programma Ladder sono dichiarate nella unità di configurazione e vanno dichiarate come descritto nella sezione QCL. Le stesse variabili possono essere utilizzate sia in Ladder che in QCL anche contemporaneamente.
In seguito si guida l'utente all'inserimento di un elemento nella rete LADDER utilizzando le voci del menù predisposte.

0.6.2 Come inserire un elemento

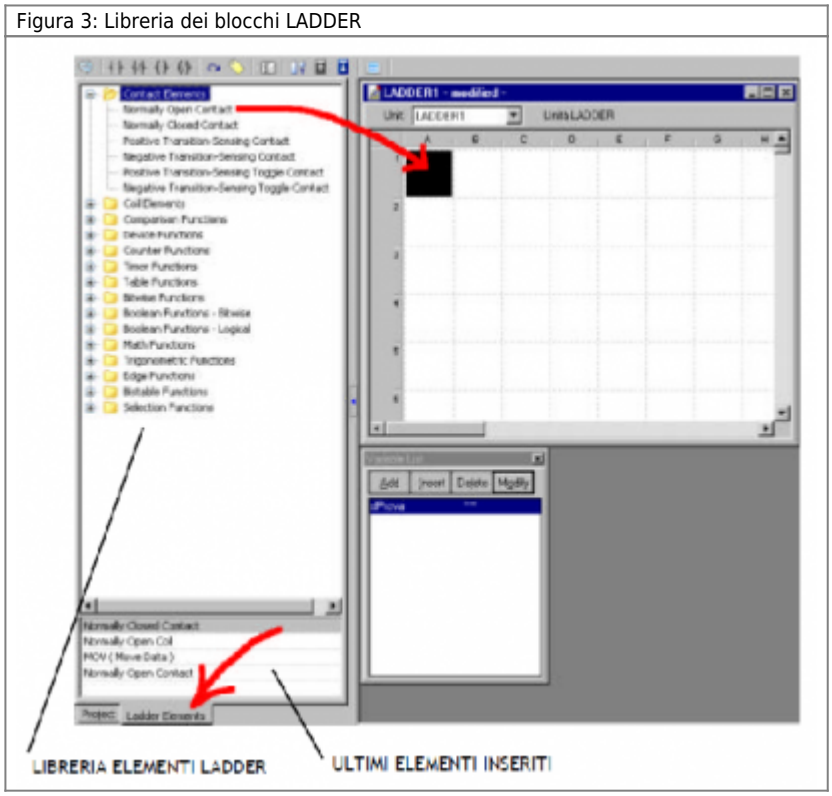
Spieghiamo con un esempio come inserire un elemento in una rete LADDER. Utilizziamo per il nostro scopo sempre il progetto Firstapp.qm4. Innanzitutto si deve inserire una nuova unità LADDER selezionando ***File > Add unit > Ladder unit***. Comparirà la finestra di Figura 2.

Figura 2: inserimento di nuova unità LADDER.



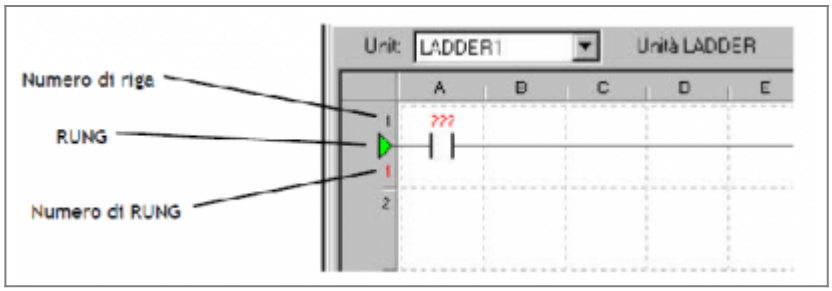
Una volta confermato con *Ok* una nuova unità verrà ad aggiungersi nella finestra **UNIT MANAGER**. Con un doppio click su questa nuova unità LADDER si aprirà l'editor LADDER. A questo punto è possibile inserire il primo elemento della nostra rete LADDER utilizzando la "libreria degli elementi LADDER" come mostrato in figura 3.

Figura 3: Libreria dei blocchi LADDER



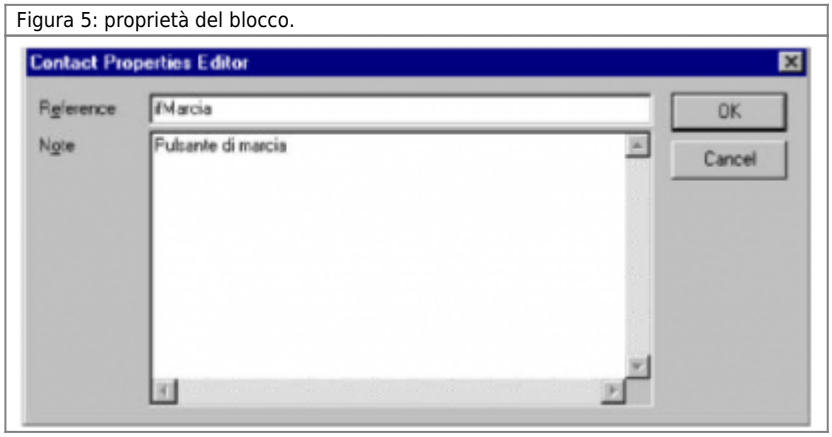
A questo punto è possibile scegliere tra i vari elementi messi a disposizione e posizionarli sulla griglia dell'editor. L'operazione si può fare sia trascinando l'elemento sulla posizione voluta, sia facendo un doppio click sull'elemento stesso. In questo caso l'elemento viene posizionato nella casella dell'editor evidenziata in quel momento.
E' possibile selezionare l'elemento ladder da inserire scegliendolo anche dalla lista degli "ultimi elementi inseriti".
L'elemento ladder deve essere inserito in una rete esistente. Se la rete non esiste bisogna crearla inserendo un nuovo "rung".
Per inserire un nuovo "rung" selezionare il menù ***Edit > New Rung***.
Se supponiamo di voler inserire il contatto normalmente aperto si deve selezionare Normally Open Contact e sull'editor appare il simbolo del contatto (Figura 4).

Figura 4: contatto normalmente aperto inserito in una rete LADDER.



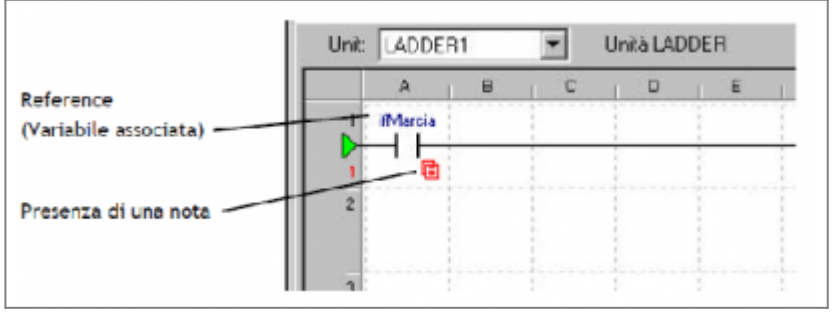
La libreria degli elementi LADDER è formata da una serie di categorie in cui sono suddivisi i vari elementi disponibili per comporre la rete LADDER. Nel seguito faremo una veloce carrellata su queste categorie. Il programmatore tenga presente che la maggior parte degli elementi sono elementi di tipo standard (IEC1131) e tutti sono dotati di un help in linea richiamabile direttamente da Qview (selezionando l'elemento e premendo F1).
L'elemento inserito nella figura 4 non ha associata nessuna variabile. Per associare una variabile all'elemento inserito si deve evidenziare l'elemento e selezionare il menù ***Edit > Element Properties***. Apparirà la finestra di figura 5.

Figura 5: proprietà del blocco.



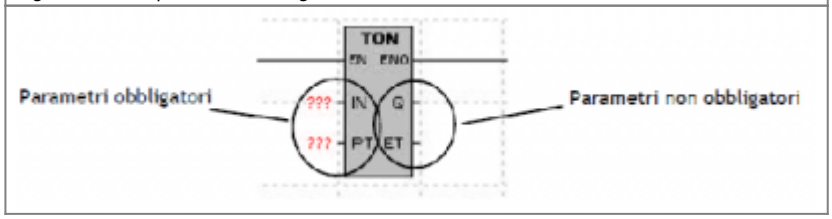
Il campo "Note" serve per associare all'elemento delle notazioni utili per il programmatore. Una volta compilata la finestra come in figura 5, la rete LADDER appare come in figura 6.

Figura 6: proprietà del blocco.



La finestra delle proprietà dell'elemento permette di inserire tutte le variabili di cui l'elemento necessita. Alcune variabili però non sono obbligatorie. Per esempio in figura 7 si osserva che alcuni parametri dell'elemento **TON (On Delay)** non sono obbligatori e questo è indicato dal fatto che non presentato i "???" (tre punti interrogativi).

Figura 7: TON: parametri obbligatori e non.



0.6.3 Categorie degli elementi

Le categorie degli elementi LADDER disponibili per comporre la rete LADDER sono:

Contact Elements:	elementi di contatto;
Coil Elements:	elementi con bobine di contatto;
Comparison Function:	confronto tra variabili;

Device Functions:	funzioni relative ai device;
Counter Functions:	elementi contatori;
Timer Functions:	temporizzatori;
Table functions:	funzioni su array;
Bitwise Functions:	operatori sul singolo bit;
Boolean Functions - Bitwise:	operatori binari a bit;
Boolean Functions - Logical:	operatori binari su variabili;
Math Functions:	operatori matematici;
Trigonometric Functions:	operatori trigonometrici;
Edge Functions:	Funzioni di trigger;
Bistable Functions:	Funzioni di set/reset;
Selection Functions:	selettori, multiplexer, limitatori.

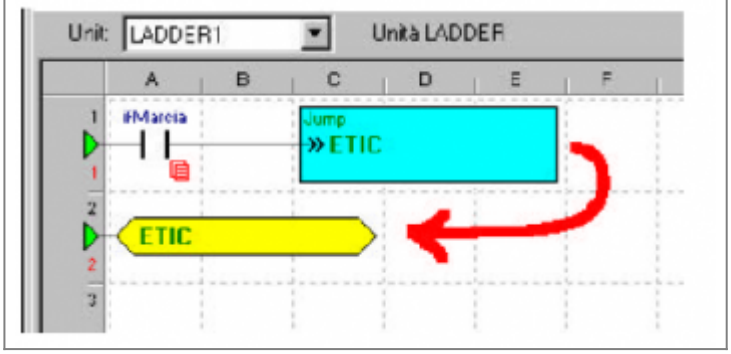
0.6.4 Commenti

Nella rete LADDER è possibile utilizzare una riga di caselle per poter inserire dei commenti. Per inserire un commento si deve selezionare del menù la voce ***Edit > New Element > Comment***.
Ogni commento può essere composto da un titolo e da un testo.

0.6.5 Jump e Label

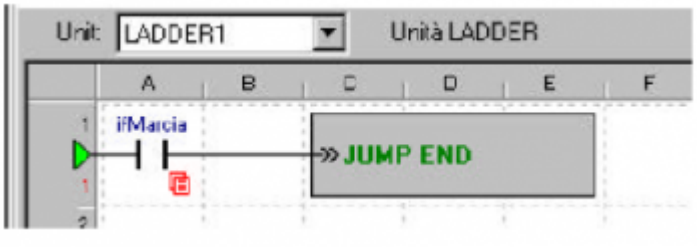
Nella struttura Ladder è prevista la possibilità di saltare ad una label in modo che il programma possa saltare delle parti gravose di programma quando non è neccessario eseguirle. Per inserire l'elemento di jump si deve selezionare dal menù la voce ***Edit > New Element > Jump***. Le proprietà di questo elemento consistono nel specificare l'etichetta associata al rung a cui saltare. Per inserire l'etichetta a cui saltare si deve agire al solito modo selezionando dal menù la voce ***Edit > New Element > Label***. Naturalmente non ci possono essere etichette (label) con lo stesso nome nella stessa unità (Figura 8).

Figura 8: Elemento di salto (JUMP) alla etichetta (LABEL) ETIC.



E' possibile inserire anche l'elemento "Jump end" per saltare alla fine dell'unià direttamente senza bisogno di etichette (Figura 9).

Figura 9: Elemento di salto alla fine (JUMP END).



0.6.6 Movimento delle righe LADDER

Nella struttura Ladder è prevista la possibilità di muovere parti di programma verso l'alto e verso il basso, in modo da consentire al programmatore di inserire linee di programma non previste. Per muovere una linea si deve selezionare un rung presente e selezionare tramite le voci del menù ***Edit > Move Rows Up*** o ***Edit > Move Rows Down***.
Una volta terminato il task ladder, esiste la possibilità di compattare le linee di programma in modo di non lasciare spazi vuoti con il comando di ***Edit > Compact Rows***.

0.6.7 Drag & Drop degli elementi LADDER

Dalla parola stessa si intuisce che il comando serve a trascinare (drag) e posizionare (drop) uno o piú elementi ladder tramite

mouse dopo averli opportunamente selezionati. Selezionando uno o piú elementi ladder e tenendo premuto il pulsante di sinistra del mouse sulla selezione, la selezione cambia colore ed il cursore del mouse cambia forma. È possibile in queste condizioni spostare la selezione. Durante lo spostamento della selezione, compare, nella status bar in basso a sinistra il messaggio: “Placing mode for drag and drop action”. Come per il comando *Paste* si possono presentare le seguenti situazioni:

- Drag & Drop eseguito in un'area libera da altri elementi ladder:
Una volta piazzato il riquadro, se si rilascia il pulsante del mouse, il riquadro colorato sparisce e l'elemento o gli elementi selezionati vengono inseriti in editor.

- Drag & Drop eseguito sopra o nell'area di altri elementi ladder:
come precedentemente descritto, una volta piazzato il riquadro, se si rilascia il pulsante del mouse, il riquadro colorato sparisce e viene visualizzato un box di messaggio che chiede di confermare o annullare l'operazione. Confermando l'operazione l'elemento o gli elementi vengono inseriti nella rete sottostante operando automaticamente tutti i collegamenti necessari per permettere una corretta compilazione del risultato finale.

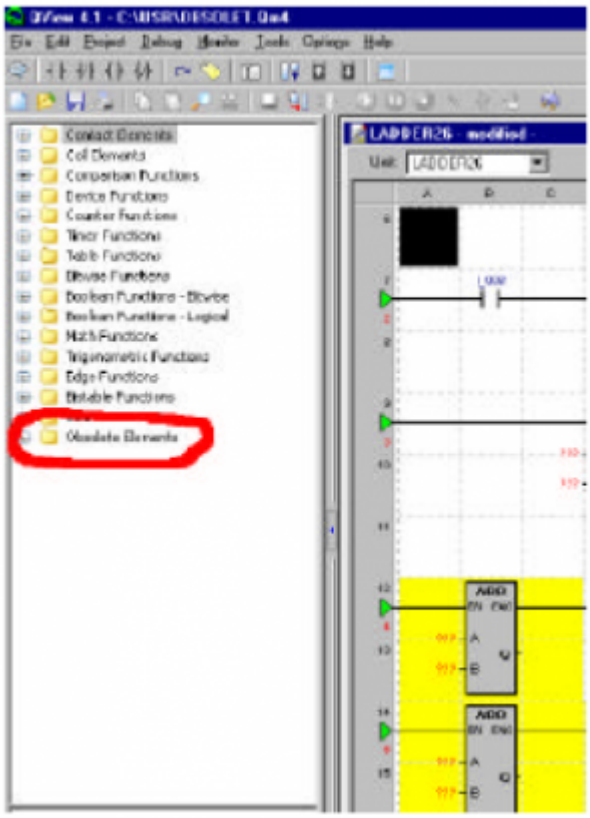
0.6.8 Elementi LADDER obsoleti

Gli **“Obsolete elements”**, sono elementi ladder utilizzati dai programmatori, ma che sono stati successivamente dichiarati obsoleti da QEM in qualità di fornitore e manutentore della libreria degli elementi ladder.

Generalmente, un elemento ladder puó venire marcato come obsoleto se è stato sostituito da un'equivalente elemento piú aggiornato, al quale per esempio, è stata aumentata l'efficienza del codice interno.

L'elemento obsoleto è sempre rimpiazzabile da un'equivalente elemento della libreria ladder in uso. In un progetto che utilizza degli elementi LADDER obsoleti essi sono riconoscibili dal fatto che vengono visualizzato con un colore di backgroud giallo (per default). La presenza di elementi obsoleti nel programma è facilemente constatabile dalla presenza della voce ***Edit > New Elements > Obsolete Elements*** e dalla cartella “Obsolete Elements” nella libreria degli elelenti (Figura 10).

Figura 10: Cartella “Obslete Elements”



Nelle impostazioni ***Options > Program Setup… > Ladder Editor*** è possibile modificare le impostazioni di default relative agli elementi ladder obsoleti.

0.6.9 Sostituzione degli elementi obsoleti

La funzionalità di sostituzione degli elementi obsoleti (**Sobstitute Obsolete element**) è stata inserita per operare la sostituzione di un elemento obsoleto inserito nella rete ladder, con un corrispondente elemento sostitutivo.
La funzionalità è attivata dalla voce di menù ***Edit > Sobstitute Obsolete Element***. Questa funzione è attivata se il cursore dell’editor ladder è posizionato sopra un'elemento obsoleto sostituibile.

Una volta sostituiti, gli ex elementi obsoleti cambiano il loro colore di background uniformandosi con gli elementi non obsoleti. Utilizzando la voce di menù **E*dit > Sobstitute All Obsolete Elements*** si ha la possibilità, in una sola passata, di sostituire su tutto il progetto (tutte le unità ladder) di tutti gli elementi ladder obsoleti. Una volta avviata questa procedura si ricercano inizialmente gli elementi ladder obsoleti, appare il messaggio:

Searches for obsolete elements in progress…

Questo messaggio viene seguito da:

No obsolete elements to substitute

se non ci sono elementi obsoleti, oppure

There are obsolete ladder elements…

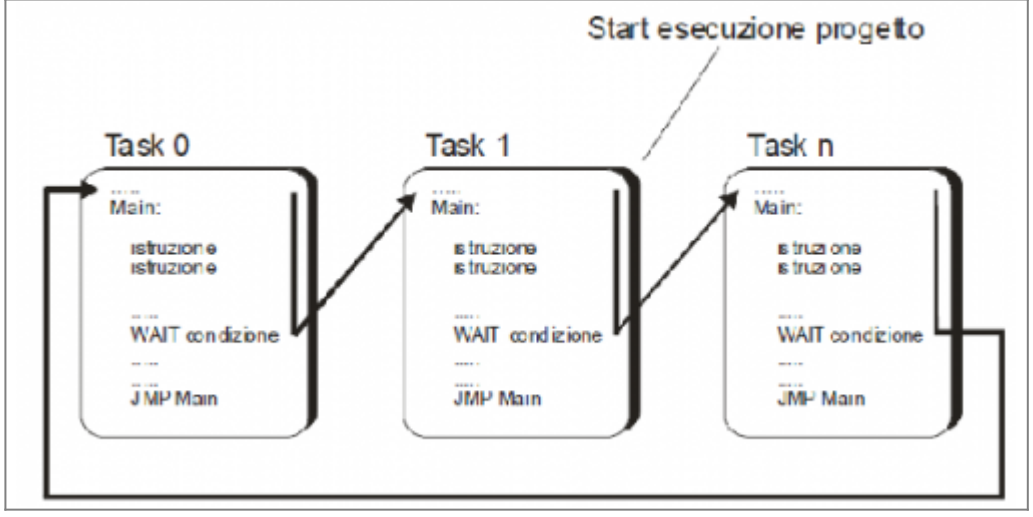
A questo punto è possibile scegliere di avviare la sostituzione completa o di abbandonare l'operazione. Una volta completata la sostituzione appare il messaggio:

All obsolete elements are substituted!

0.7 Multitasking

Il multitasking è la capacità di un sistema di gestire l'esecuzione contemporanea di diversi compiti.
Un progetto realizzato in QCL e ladder è formato da un insieme di unità o "tasks" (in italiano *compiti*, appunto). Il multitasking implementato nel sistema QMOVE è chiamato di tipo "cooperativo", nel senso che l'esecuzione delle istruzioni passa da un task ad un altro solamente quando, nel task in esecuzione, viene incontrata una particolare istruzione (WAIT < condizione >) se si tratta di un task in QCL oppure al termine del task se si tratta di un task ladder, che cede il controllo al task successivo. Il disimpegno delle risorse della CPU, quindi, dipende dal modulo che le sta utilizzando.
Si schematizza il cosiddetto "ciclo di tasks" nella figura 1. Se nel progetto esiste un task QCL che non contiene un'istruzione WAIT quando comincia a essere eseguito non "passa la mano" a nessun altro modulo e quindi il resto del progetto rimane "congelato" allo stato in cui si trovava. Al momento dell'accensione l'ordine con cui vengono eseguiti i moduli presenti è quello dichiarato nella finestra indice in QVIEW solo che il primo task eseguito è il secondo in lista poi si procede fino all'ultimo e quindi si esegue per ultimo il primo in lista.

Figura 1: esecuzione task (condizione iniziale).



Il termine "device" identifica una categoria di dispositivi software atti a svolgere attività di supporto o controllo (più o meno complesse), semplificando operazioni e procedure proprie dell'automazione industriale.
Per esempio, un device può gestire un posizionatore di tipo CNC con uscita analogica +/-10V, rendendo disponibili tutte le funzioni, i comandi ed i parametri necessari alla corretta gestione dei posizionamenti (ricerca preset, comandi per la movimentazione manuale, parametri di velocità, tempo di inversione, ...).
I device dispongono di proprie funzioni, variabili, comandi e parametri; l'utente ha la possibilità di configurarli ed inserirli all'interno del proprio progetto, in modo che uno o più device diventino parte integrante dell'applicativo sviluppato.
I devices vengono identificati in due diverse categorie: device interni e device esterni. Il firmware dei device interni risiede nella CPU; il firmware dei device esterni risiede nelle schede intelligenti aggiuntive alla CPU. L'accesso a parametri o funzioni e l'esecuzione dei comandi vengono gestite dal linguaggio QCL senza alcuna distinzione di sintassi tra device interni ed esterni.

I parametri sono delle variabili che permettono di configurare il funzionamento del device (ad esempio la velocità di posizionamento, il conteggio, ecc.):
<nome del device>:parametro

I comandi fanno eseguire al device determinate funzioni (ad esempio START, STOP, ecc.):

COMANDO <nome del device>

La lista dei device interni disponibili è inserita nei manuali d'installazione e manutenzione relativi alla CPU utilizzata.

0.7.1 Dichiarazione dei Device

La sintassi per la dichiarazione dei device viene trattata approfonditamente nella documentazione dei device.

L'utilizzo di un device deve essere dichiarato all'interno del file di configurazione. La sintassi della dichiarazione varia a seconda del device (interno o esterno).

0.7.1.1 Dichiarazione device interni

Per la dichiarazione del device interno, nel file di configurazione deve essere inserita la parola chiave INTDEVICE.
Per ulteriori informazioni fare riferimento alla documentazione relativa ai device interni.

0.7.1.2 Dichiarazione device esterni

Per la dichiarazione del device esterno, nel file di configurazione deve essere inserita la parola chiave EXTDEVICE.
Per ulteriori informazioni fare riferimento alla documentazione relativa ai device esterni.

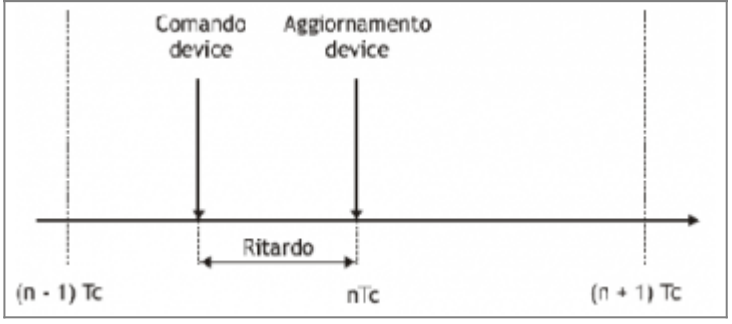
0.7.2 Utilizzo dei devices interni

In questo capitolo si daranno delle informazioni preliminari per un corretto utilizzo dei devices interni. È indispensabile comunque completare ciò che segue con la documentazione specifica fornita per ognuno dei devices.

0.7.2.1 Tempo di campionamento

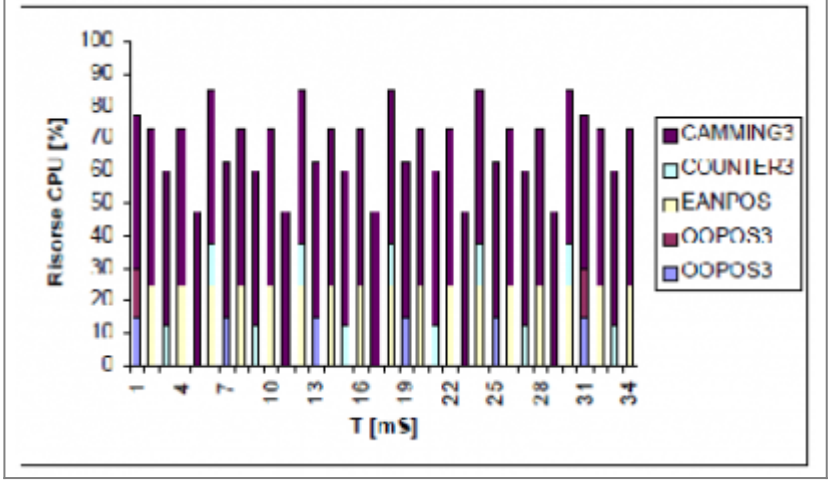
Come abbiamo già detto, nel capitolo dedicato al multitasking, una delle caratteristiche peculiari dei devices è il tempo di campionamento (Tc) che stabilisce ogni quanto tempo viene eseguita la gestione del device da parte della CPU.
Vediamo quali sono i criteri generici nella scelta dei tempi di campionamento. Solitamente un tempo piccolo permette al device di venire eseguito molto frequentemente e quindi di poter reagire velocemente alle azioni esterne (es.: miglior regolazione degli assi). Durante l'esecuzione del codice QCL un accesso in scrittura al device o un comando viene processato dopo un tempo massimo pari al tempo di campionamento. Questo fa sì che più piccolo è Tc, minore è il ritardo di esecuzione del device. Nel grafico di figura 1 si dà una esemplificazione di quanto detto.

Figura 1: ritardo aggiornamento device.



La scelta dei tempi di campionamento per i device utilizzati nel progetto che si sta realizzando deve essere fatta con le opportune attenzioni. Una domanda che il lettore si sarà posto è la seguente: "Perchè non posso scegliere come tempi di campionamento il valore minimo consentito per ognuno dei device?". In effetti in tal modo le prestazioni sarebbero le massime. Questa scelta però non è sempre possibile.
La CPU riserva una parte della sua risorsa di calcolo per la gestione dei devices utilizzati nel progetto. Ogni device ad ogni campionamento impegna una parte di questa risorsa. Per determinare e stimare l'utilizzo della risorsa consideriamo che la CPU ad ogni millisecondo mette a disposizione una risorsa pari al 100%. Nella documentazione relativa ai firmware si puo' trovare la percentuale di risorse che ogni device utilizza in un istante di campionamento. La CPU può gestire nello stesso campionamento più devices e la risorsa occupata complessivamente è la somma delle percentuali relative ad ogni devices. Nella scelta dei tempi bisogna evitare che in qualche campionamento la risorsa complessiva superi il valore 100%. Automaticamente la CPU sfasa l'esecuzione dei device per evitare di superare il limite massimo della risorsa di calcolo disponibile.
Il grafico di figura 2 dà una esemplificazione di quanto detto. Nell'esempio sono stati installati: un device OOPOS3 con tc=6msec., un OOPOS3 con tc=30msec., un device EANPOS con tc=2msec. un COUNTER3 con tc=3msec ed un CAMMING3 con tc=1msec.

Figura 2: impegno risorse CPU per gestione device interni.



0.7.2.2 Comandi consecutivi e loro priorità

Un comando ad un device non viene elaborato subito dalla CPU ma al successivo tempo di campionamento, senza per questo che la sequenza delle successive istruzioni QCL venga interrotta. Per questo motivo il device potrà trovarsi a processare nello stesso istante di campionamento più comandi, ed in questo caso la gestione del device non terrà conto della sequenza con cui tali comandi sono stati dati ma li processerà secondo un ordine interno di priorità. Tale priorita è specificata nella documentazione relativa ad ogni device.
Quando si vuole assicurare la sequenza con cui i comandi sono stati inseriti nella stesura del codice QCL, bisogna porre delle istruzioni di WAIT e come condizione uno stato del device. In tal modo l'istruzione WAIT attende l'esecuzione del comando prima di eseguire l'istruzione successiva.

0.7.2.3 Comandi complementari

Esistono dei comandi che sono tra loro complementari e cioè che producono ognuno l'effetto inverso dell'altro. Se ad un campionamento il device si trova ad eseguire alcuni comandi fra loro complementari, risulta avere effetto solamente l'ultimo della sequenza. Anche in questo caso per assicurare l'esecuzione dei due comandi fare riferimento al paragrafo "Comandi consecutivi e loro proprietà".

0.7.3 Ritentività parametri Intdevice e Extdevice

Quando si utilizza nell'applicativo un device vi sono solitamente alcuni parametri la cui programmazione non è richiesta perchè riguardano funzionalità del device non interessanti per l'applicazione. In alcuni casi il valore zero su questi parametri è necessario proprio per disabilitare una funzionalità. È proprio su questi parametri che bisogna porre attenzione: esiste una importante differenza su come devono essere trattati questi parametri nel caso di device interni o device esterni. Nei device interni i parametri vengono azzerati con il comando reset del menu Debug nel programma QVIEW. Così l'utente dopo il primo download dell'applicativo, programma solo i parametri interessanti, i rimanenti sicuramente sono tutti a valore zero.
I devices esterni invece non hanno un particolare comando per azzerare i parametri. L'applicativo deve essere realizzato scrivendo il corretto valore di tutti i parametri necessari al funzionamento del device senza prevedere che i parametri siano tutti a valore zero la prima volta che la scheda intelligente viene installata.

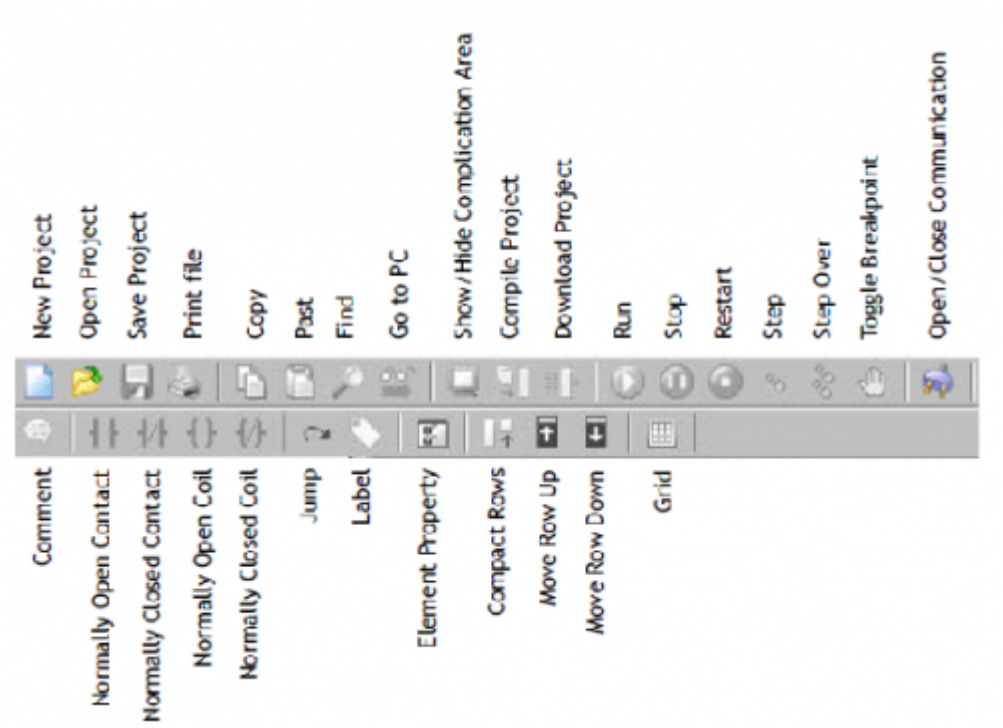
0.8 Interfaccia Qview

Le note inserite fianco delle descrizioni definiscono le condizioni necessarie per l'utilizzo del comando descritto.

In questo capitolo verranno descritti i vari menu e comandi di QVIEW. Per l'approfondimento di alcune funzionalità proprie dell'ambiente Windows (Apri, Salva, ...) si rimanda alla documentazione del sistema operativo.

0.8.1 Barra degli strumenti

La barra degli strumenti è composta da una serie di icone che ripropongono le funzionalità dei comandi principali (o di uso frequente).



0.8.2 Barra di stato

La barra di stato è composta da quattro sezioni distinte.
La prima sezione a sinistra è dedicata alla visualizzazione di messaggi sullo stato della CPU (Backup, Restore, ...). All'interno di una finestra di editor, cliccando con il tasto destro del mouse su una variabile, nella sezione in oggetto viene visualizzato il valore della variabile. La seconda sezione da sinistra, relativa all'editor di testo, visualizza la posizione del cursore (riga e colonna) e la modalità di scrittura (INS = inserimento di testo e OVR = sovrascrittura testo) (Figura 1).
La terza sezione da sinistra visualizza lo stato della porta seriale di comunicazione tra PC e CPU: stato della porta (OPEN o CLOSE), protocollo di comunicazione in uso e velocità di trasmissione (Figura 2).
La quarta sezione più a destra visualizza i messaggi "Match OK" o "No Match" quando la comunocazione seriale e attiva e il progetto aperto è lo stesso di quello presente nella CPU oppure e diverso, rispettivamente.

Figura 1: seconda sezione da sinistra: posizione cursore.

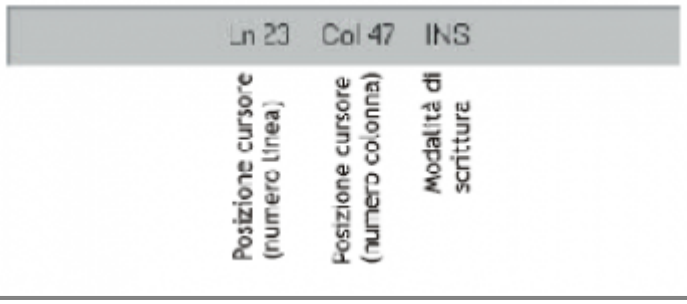
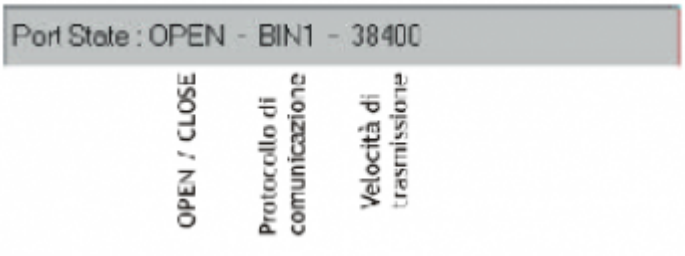


Figura 2: terza sezione da sinistra: stato e parametri porta di comunicazione seriale.



0.8.3 Menu File

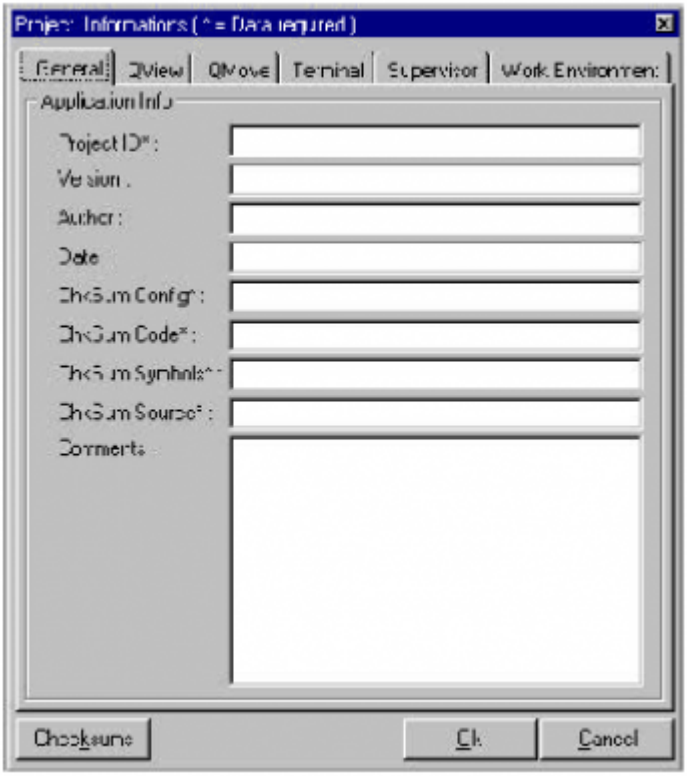
Contiene i comandi per la gestione dei progetti e dei files che li compongono.

0.8.3.1 New Project

Permette di creare un nuovo progetto. Inizialmente viene richiesto il nome da assegnare al progetto (Figura 1).
Viene quindi aperta automaticamente la finestra “Project Information” (Figura 2) per inserire i dati del progetto che si intende realizzare. Questa tabella ha lo scopo di raccogliere una serie di informazioni, essa può essere compilata anche in un secondo momento (in ogni caso non è obbligatorio compilarla).

Figura 1: inserimento del nome del progetto.

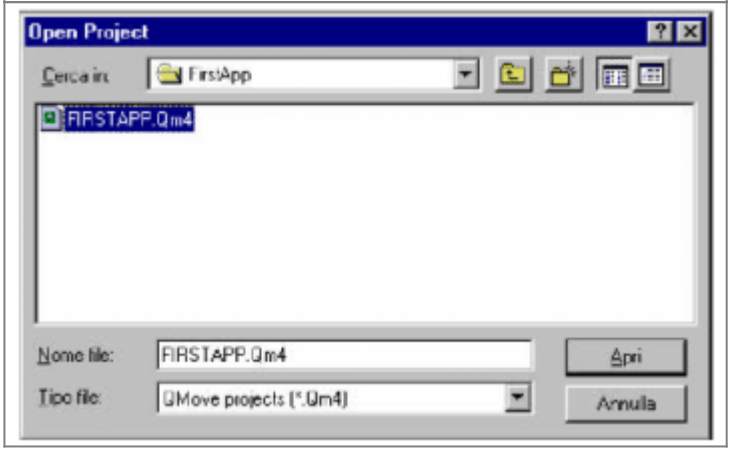
Figura 2: finestra con le informazioni di progetto.



0.8.3.2 Open Project

Permette di aprire un progetto esistente (Figura 3).

Figura 3: apertura di un progetto.



E’ possibile aprire i progetti realizzati con versioni precedenti di Qview selezionando le diverse estensioni da “Tipo file”.

0.8.3.3 Save Project

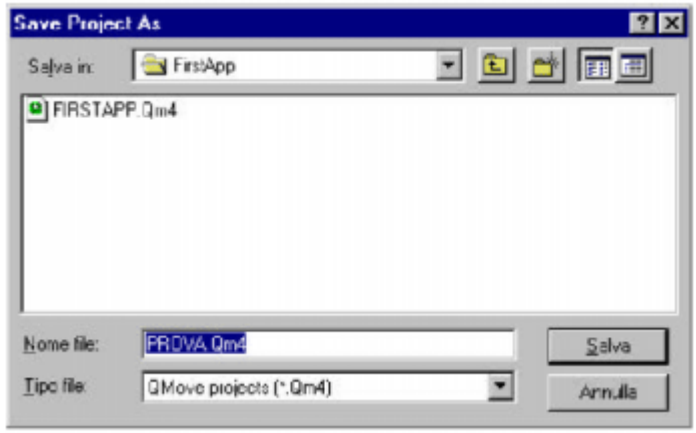
I comandi di salvataggio del progetto (Save Project e Save Project As …) sono disponibili solo con un progetto in uso.

Permette di salvare le modifiche apportate al progetto in uso.

0.8.3.4 Save Project As

Permette di salvare una copia del progetto in uso (comprese le eventuali modifiche apportate) (Figura 4).

Figura 4: salvataggio di un progetto con un altro nome.



=== - Close Project === - Permette di chiudere il progetto in uso.

0.8.3.5 Add Unit

Permette di aggiungere una nuova unità al progetto in uso.
Le nuove unità possono essere:
- Configuration Unit: inserisce l'unità di configurazione.
- QCL Unit: inserisce una nuova unità (task) in linguaggio QCL
- Ladder Unit: inserisce una nuova unità (task) in linguaggio Ladder
- Document Unit: inserisce una nuova unità document (documento di testo per la raccolta di commenti, specifiche, note …).
Questa nuova unità verrà aggiunta in coda alla lista delle unità già presenti.

0.8.3.6 Insert Unit

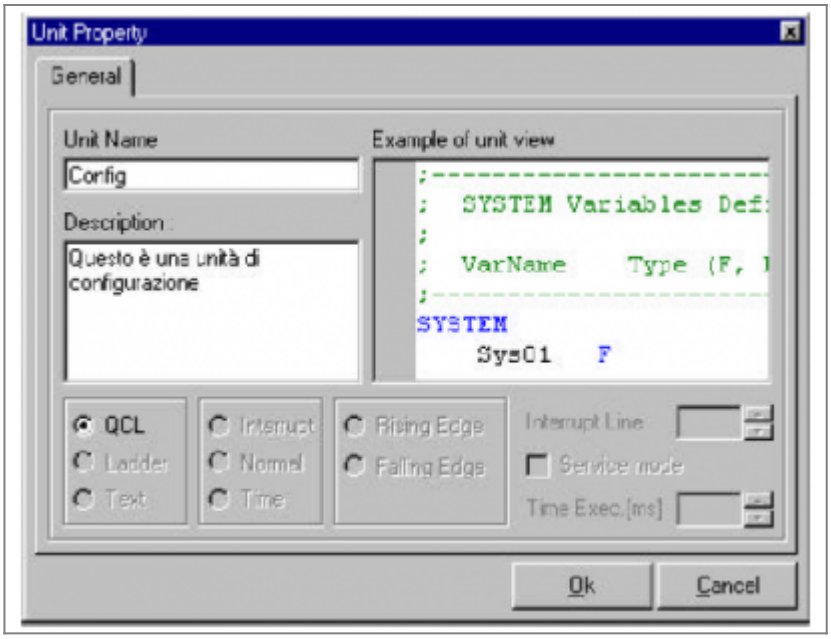
Permette di inserire una nuova unità al progetto in uso.\ Le nuove unità possono essere:\ - Configuration Unit: inserisce l'unità di configurazione.\ - QCL Unit: inserisce una nuova unità (task) in linguaggio QCL\ - Ladder Unit: inserisce una nuova unità (task) in linguaggio Ladder\ - Document Unit: inserisce una nuova unità document (documento di testo per la raccolta di commenti, specifiche, note …).

Questa nuova unità verrà inserita nella lista appena sopra alla unità selezionata.

0.8.3.7 Creazione di un'unità di configurazione

Ogni volta che viene creato una nuova unità di configurazione con i comandi "Add unit" o "Insert unit", appare la finestra "Unit property" (Figura 5).

Figura 5: intestazione dell'unità di configurazione.



In questa finestra è possibile assegnare un nome all'unità e una breve descrizione dell'unità. Una volta confermato con ***Ok***, l'unità di configurazione viene aggiunta alla lista delle unità e viene aperta la finestra di editor per iniziare a modificarla.
In un progetto deve esistere una sola unità di configurazione.

0.8.3.8 Creazione di un'unità QCL o LADDER

Ogni volta che viene creato una nuova unità QCL o LADDER con i comandi "Add unit" o "Insert unit", appare la finestra "Unit property" (Figura 6, 7).

Figura 6: intestazione delle unità QCL.

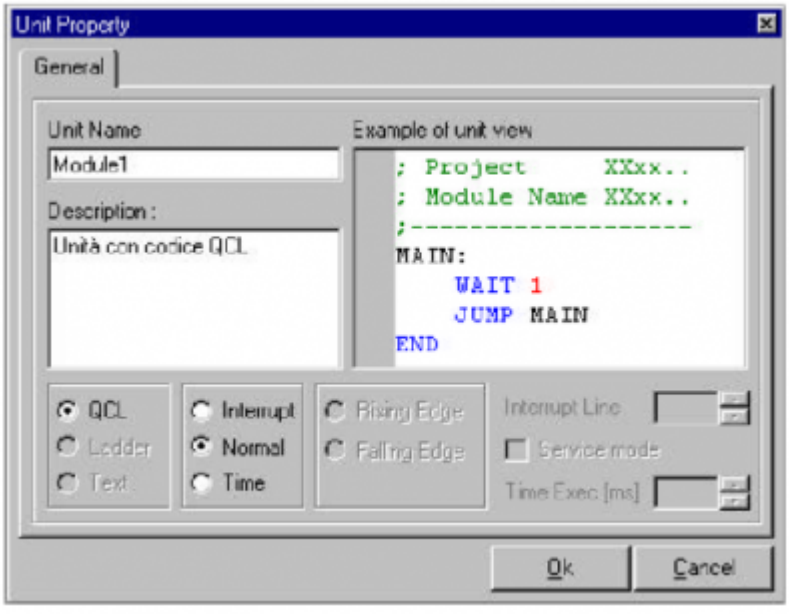
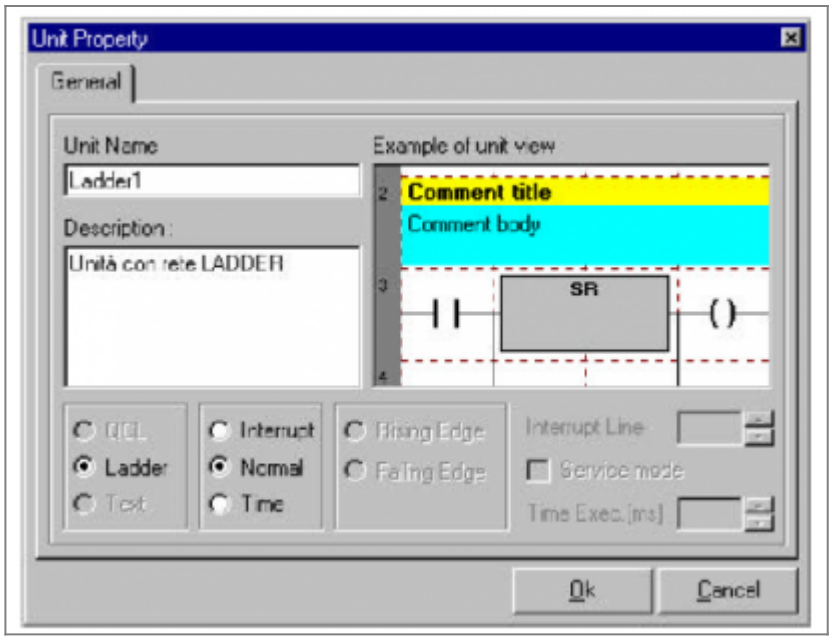


Figura 7: intestazione delle unità LADDER.



In questa finestra è possibile assegnare un nome all'unità e una breve descrizione dell'unità.
Una volta confermato con ***Ok***, l'unità QCL o LADDER viene aggiunta alla lista delle unità e viene aperta la finestra di editor per iniziare a modificarla.
Nelle finestre "Unit property" è possibile specificare se l'unità (QCL o LADDER) è una unità **normale**; **in interrupt**: si deve specificare il fronte (**Rising Edge**: fronte di salita, **Falling Edge**: fronte di discesa) dell'impulso che avvia l'interrupt e la linea di interrupt utilizzata (**Interrupt Line**);
a tempo: si deve specificare ogni quanto tempo si deve ripetere il codice inserito nell'unità (**Time Exec.**).

0.8.3.9 Creazione di un'unità di documentazione

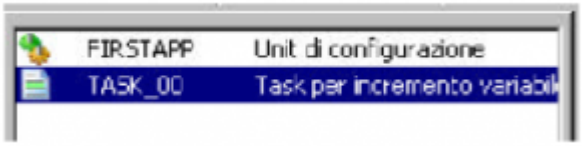
Le unità di documentazione sono utilizzate dal programmatore per scrivere appunti, note, particolari di funzionamento. E' possibile utilizzare queste unità solamente in modalità di testo.

0.8.3.10 Remove Unit

Il comando è disponibili solo con un progetto in uso.

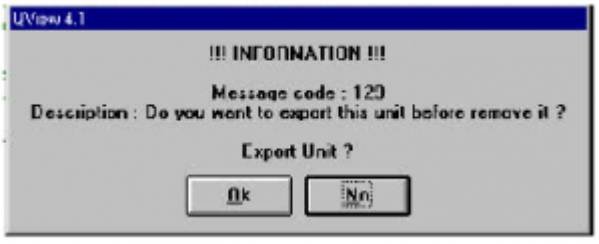
Permette di rimuovere l'unità selezionata.

Figura 8: selezione del file da rimuovere.



Prima dell'eliminazione dell'unità, il programma chiede se si vuole esportare l'unità prima di rimuoverla; in caso contrario, l'unità verrà persa.

Figura 9: richiesta di esportazione unità.

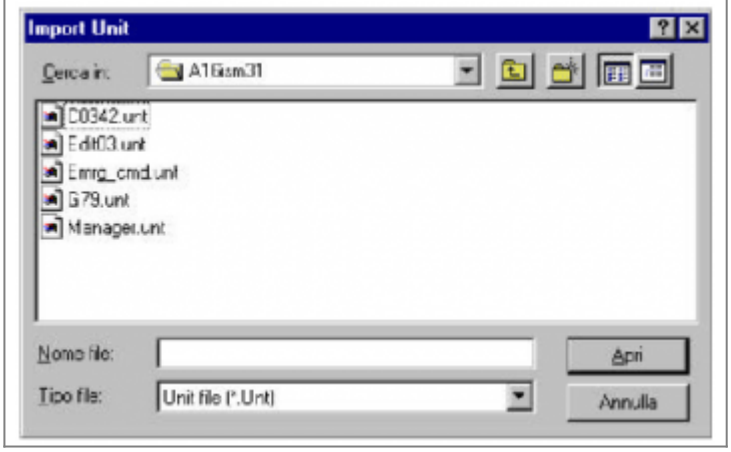


0.8.3.11 Import Unit

Il comando è disponibili solo con un progetto in uso.

Permette di importare un unità precedentemente esportata anche da un altro progetto (Figura 10).

Figura 10: Importazione di un unità.



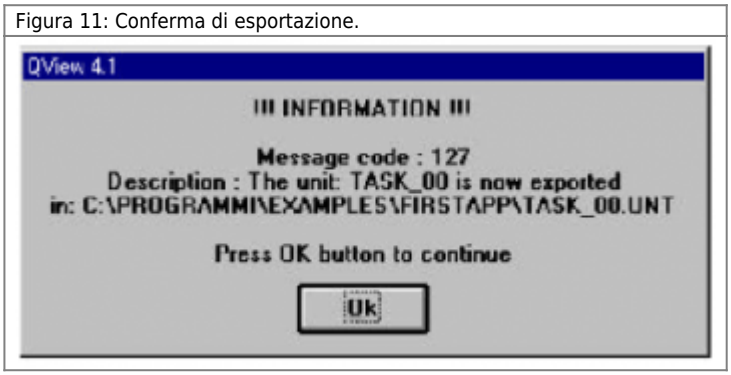
E' possibile selezionare l'alternativa "Add unit..." o "Insert unit..." per importare l'unità in coda alla lista o sopra l'unità selezionata in quel momento.

0.8.3.12 Export Unit

Il comando è disponibili solo con un progetto in uso.

Permette di esportare una copia dell'unità di progetto selezionata, per poterla trasferire ad un altro progetto. La copia originale rimarrà nel progetto stesso. L'unità verrà esportata nella stessa directory del progetto aperto. L'unità verrà esportata come un file con estensione unt.

Figura 11: Conferma di esportazione.

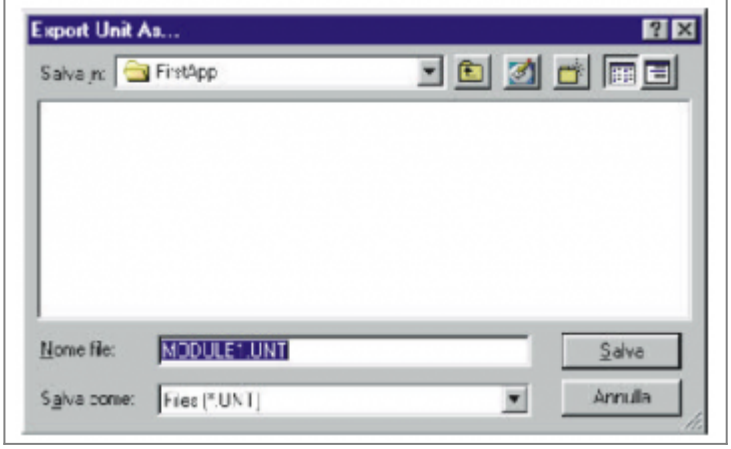


0.8.3.13 Export Unit As

Il comando è disponibili solo con un progetto in uso.

Permette di esportare una copia dell'unità di progetto selezionata rinominando l'unità. La copia originale rimarrà nel progetto stesso.

Figura 12: richiesta di esportazione unità con rinomina del nome.



0.8.3.14 Unit Property

Il comando è disponibili solo con un progetto in uso.

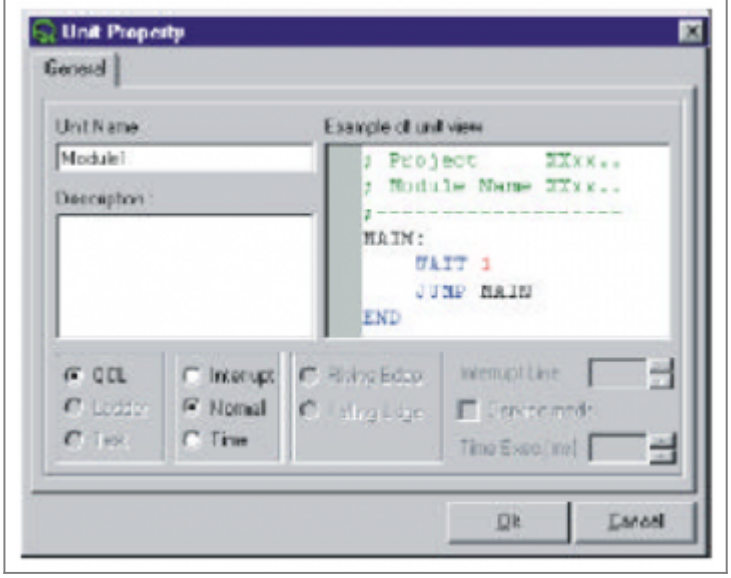
Permette di modificare le proprietà dell'unità selezionata. In particolare è possibile impostare il comportamento Runtime della unità in questione, cioè la modalità di esecuzione dell'unità una volta che verrà trasferita nella CPU del Qmove. Le unità possono avere i seguenti comportamenti:

Normal Fino al limite massimo consentito (totale massimo di unità: 65535).
Conferisce alla unità un comportamento Runtime ciclico (vedere il capitolo relativo al multitasking).

Interrupt Massimo tre unità per progetto.
Conferisce alla unità un comportamanto Runtime determinato da eventi esterni all'apparecchiatura (linee di interrupt hardware). Si puó determinare se l'esecuzione della unità sarà attivata sul fronte di salita o discesa del segnale entrante ed il numero della linea hardware ad essa collegata, impostabile da 1 a 8.

Time Massimo sette unità per progetto.
Conferisce alla unità un comportamento Runtime determinato dallo scadere di un tempo prefissato, interno all'apparecchiatura. Il predetto tempo è regolabile in un range da 1 a 999 millisecondi. Nell' inserimento di una nuova unità a tempo, il tempo proposto è di 100ms.

Figura 13: modifica delle proprietà di un'unità.



Aprendo la finestra "Unit property" (Figura 13) è possibile rinominare un unità precedentemente introdotta cambiando il suo Unit Name.

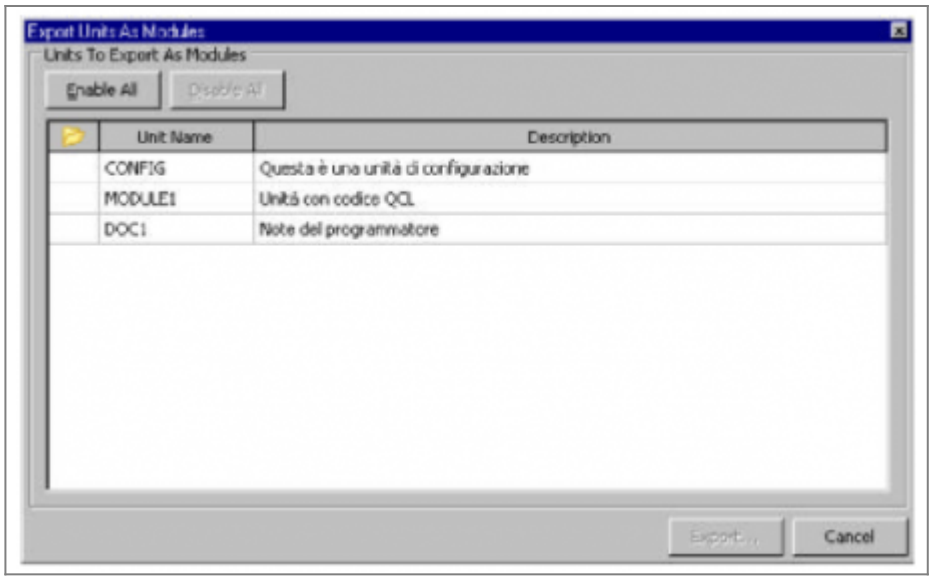
0.8.3.15 Import Module As Unit

Permette di importare un'unità realizzata con le vecchie versioni di Qview (*.mod) nel progetto oppure esportata da un'altro progetto realizzato con Qview 4. E' possibile aggiungere l'unità in coda alla lista (Add Module as unit) oppure inserirla in una posizione intermedia sopra a quella selezionata (Insert Module as unit).

0.8.3.16 Export Unit as Module

Permette di esportare una o più unità in un formato compatibile con le precedenti versioni del Qview (moduli per Qview 2.x e Qview 3.x). Una volta selezionata questa funzione viene visualizzata la finestra di figura 14 dove è possibile selezionare quali unità esportare (vengono elencate solo le unità non LADDER).

Figura 14: esportazione delle unità come moduli.



Per selezionare l'unità da esportare bisogna eseguire un doppio-click sull'unità stella.

0.8.3.17 Export Symbols File

Permette di esportare il file simboli dal progetto per poter riallineare i simboli nel terminale.

Figura 15: esportazione simboli.

0.8.3.18 Export Symbols File As ...

Permette di esportare il file simboli dal progetto, rinominandoli, per poter riallineare i simboli nel terminale.

Figura 16: esportazione simboli con rinomina.

0.8.3.19 Export Binary File & Export Binary File As ...

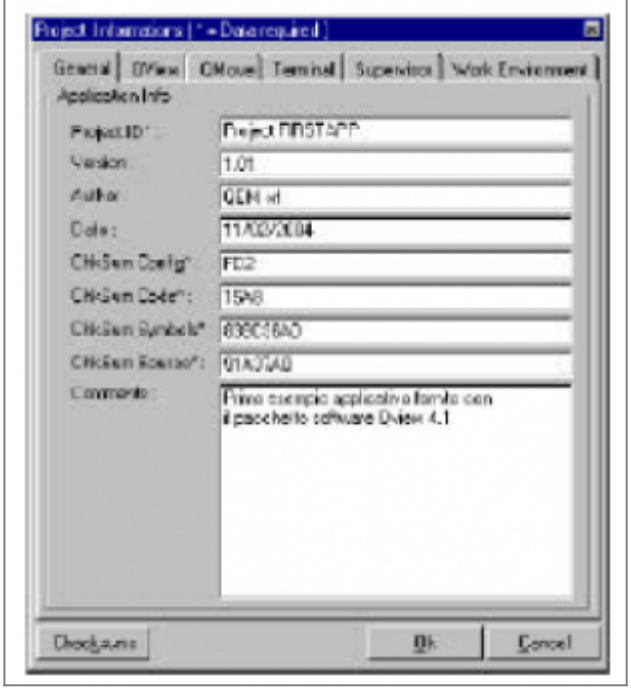
Questa funzionalità è disponibile se la compilazione del progetto è andata a buon fine. Esso permette di esportare il file binario (risultato della compilazione) utile per poter eseguire il download dell'applicativo nella CPU senza l'utilizo della comunicazione seriale (per esempio trasferendolo alla CPU attraverso una Multi Media Card). E' possibile esportare il file binario con lo stesso

nome del progetto (**Export Binary File**) oppure cambiando il suo nome (**Export Binary File As…**).

0.8.3.20 Project Informations ...

Permette di visualizzare una finestra (Figura 20) per l'inserimento delle informazioni specifiche del progetto.

Figura 17: visualizzazione informazioni sul progetto in uso.



In questa finestra sono presenti delle cartelle che suddividono le informazioni di progetto per argomento. Alcune informazioni sono contrassegnate con un asterisco per indicare che sono obbligatorie per una completa informazione sul progetto. Se alcune di queste informazioni obbligatorie non vengono inserite, periodicamente appariranno dei messaggi che avviseranno il programmatore di questa mancanza (per disabilitare questi avvisi si veda il capitolo *Interfaccia* ***Qview - Menu Options - Program Setup***). In questa finestra esiste il tasto "Checksums" che inserisce in tabella i codici checksum del progetto automaticamente.

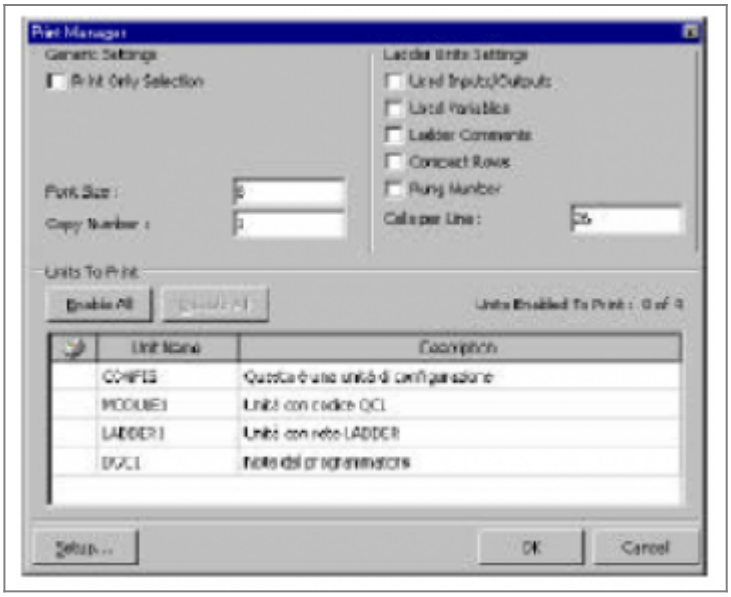
0.8.3.21 Print

Il comando è disponibili solo selezionando una finestra di editor.

Permette di accedere ad un finestra "Print Manager" tramite la quale è possibile stampare una parte o tutto il progetto con alcune possibilità di configurazione della stampa stessa. In figura 18 viene mostrata questa finestra dove nel riquadro basso appare la lista delle unità che compongono il progetto. Selezionando queste unità con il mouse è possibile decidere quali stampare. Nel riquadro in alto a sinistra si possono impostare dei settaggi generici sulla dimensione del font e sul numero di copie da eseguire.
Nel riquadro in alto a destra è possibile configurare la stampa dei task LADDER dove si può specificare:
- stampa della lista degli ingressi e uscite utilizzati nel task LADDER;
- stampa delle variabili utilizzate nel task LADDER;
- stampa dei commenti inseriti nel task LADDER;
- stampa con compattamento delle righe LADDER eliminando gli spazi vuoti;
- stampa dei numeri dei rung;
- numero di celle per linea da stampare.

Inoltre in alto a sinistra è possibile selezionare la stampa solo della parte dell'unità evidenziata.

Figura 18: selezione modalità di stampa.



0.8.3.22 Richiamo ultimi progetti aperti

È possibile richiamare direttamente gli ultimi 4 file aperti (Figura 19).

Figura 19: apertura progetti recenti.

0.8.3.23 Exit

Chiude QVIEW chiedendo il salvataggio delle eventuali modifiche apportate al progetto in uso.

0.8.4 Menu Edit

I comandi a seguire sono disponibili solamente quando è visualizzata un'unità con l'apposito editor.

0.8.4.1 New Element

Permette di inserire un nuovo elemento nella rete Ladder (deve essere visualizzata un'unità LADDER).

0.8.4.2 New Rung

Permette di inserire un nuovo rung nella rete Ladder (deve essere visualizzata un'unità LADDER).

0.8.4.3 Delete Rung

Permette di eliminare il rung selezionato nella rete Ladder (deve essere visualizzata un'unità LADDER).

0.8.4.4 Toggle Link

Permette di collegare in verticale due elementi ladder. Il collegamento viene fatto sempre verso il basso e nella parte sinistra della cella selezionata (deve essere visualizzata un'unità LADDER).

0.8.4.5 Substitute Obsolete Element & Substitute All Obsolete Elements...

Permette di sostituire l’elemento o tutti gli elementi obsoleti. Per una descrizione più completa si veda il capitolo “Editor LADDER - Elementi LADDER obsoleti” (deve essere visualizzata un’unità LADDER).

0.8.4.6 Element Properties...

Permette di modificare le variabili utilizzate all’interno dell’elemento LADDER selezionato (deve essere visualizzata un’unità LADDER).

Undo

Permette di cancellare l’ultima modifica fatta.

Redo

Permette di annullare il comando di Undo e reinserire la modifica eliminata.

0.8.4.7 Cut - Copy - Paste - Delete

Per tagliare, copiare, incollare ed eliminare del testo o degli elementi.

0.8.4.8 Select All

Permette di selezionare tutto il testo o gli elementi contenuti in una finestra di editor.

0.8.4.9 Compact Rows

Permette di compattare le righe di codice Ladder nel minor numero di celle possibili eliminando le righe vuote (deve essere visualizzata un’unità LADDER).

0.8.4.10 Move Rows Up

Permette di spostare in alto una riga di codice ladder (deve essere visualizzata un’unità LADDER).

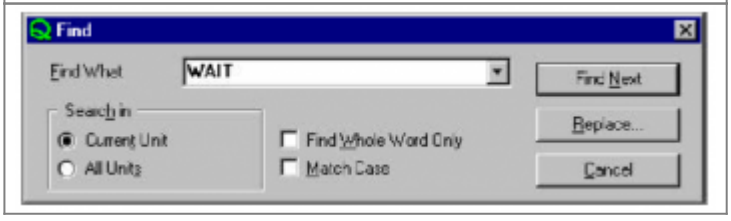
0.8.4.11 Move Rows Down

Permette di muovere in basso una linea di codice Ladder (deve essere visualizzata un’unità LADDER).

0.8.4.12 Find

Ricerca la parola inserita in funzione dei criteri di ricerca selezionati (Figura 20).
Criteri di ricerca
- Search in Current Unit: ricerca solo nell’unità selezionata.
- Search in All Units: ricerca in tutte le unità del progetto.
- Find Whole Word Only: cerca solo parole intere
- Match Case: cerca parole come digitato, rispettando maiuscole e minuscole.

Figura 20: finestra inserimento criteri di ricerca.



0.8.4.13 Find Next

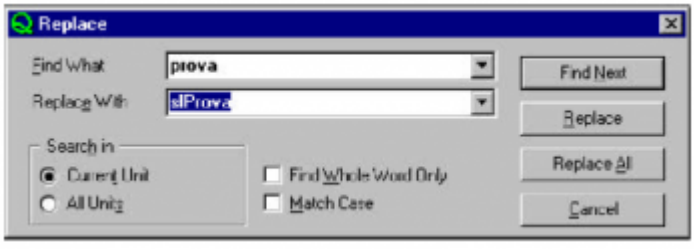
Una volta trovato un termine (comando Find), permette di continuare la ricerca dello stesso termine sul resto del documento (o sulle altre unità), mantenendo inalterati i criteri di ricerca.

0.8.4.14 Replace

Ricerca la parola inserita (Find What) con i criteri di ricerca impostati e la sostituisce con la nuova parola (Replace With)(Figura 21).
Criteri di ricerca
- Search in Current Unit: ricerca solo nel file selezionato;
- Search in All Units: ricerca in tutti i file del progetto;
- Find Whole Word Only: cerca solo parole intere;
- Match Case: cerca parole come digitato, rispettando maiuscole e minuscole.

Criteri di sostituzione
- Find Next: non esegue nessuna sostituzione ma inizia la ricerca della parola (Find Wath) nel restante del documento (o negli altri documenti);
- Replace: sostituisce la parola cercata (Find Wath) con la nuova parola (Replace With);
- Replace All: sostituisce automaticamente tutte le parole (Find Wath) dell'intero documento (o di tutti i documenti).

Figura 21: finestra inserimento criteri per ricerca e sostituzione.



0.8.4.15 Go to ...

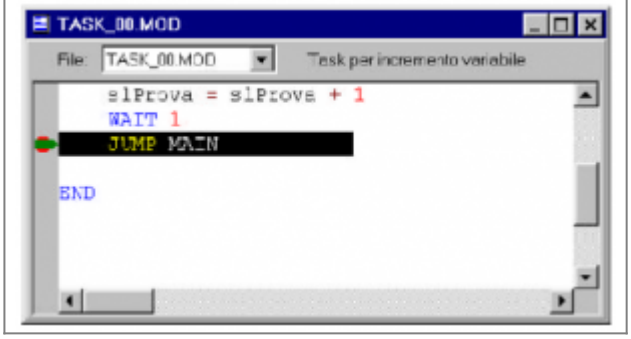
Sposta il cursore al numero di riga inserito (Figura 22).

Figura 22: inserimento numero di riga per spostamento cursore.

0.8.4.16 Go to PC

Dal momento in cui l'esecuzione del progetto viene interrotta (per esempio per l'inserimento di un breakpoint), questo comando visualizza la riga di programma alla quale si è verificata l'interruzione. In figura 23, il comando Go to PC visualizza l'interruzione per l'inserimento di un breakpoint.

Figura 23: esecuzione programma interrotta da breackpoint.



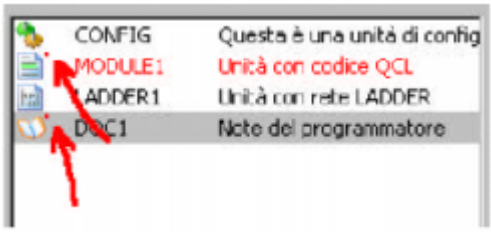
0.8.4.17 Next Unit / Previous Unit

Permette di spostarsi tra le unità del progetto.

0.8.4.18 Next Selected Unit / Previous Selected Unit

Quando fosse necessario spostarsi tra alcune unità del progetto ignorandone alcuni altri, selezionare uno per volta i file da visualizzare e premere la barra spaziatrice. A fianco dei file selezionati compare un punto rosso (Figura 24).
Con i comandi Next e Previous Selected Unit è possibile scorrere in avanti ed indietro solamente tra i file selezionati.
Per rimuovere la marcatura, selezionare il file e ripremere la barra spaziatrice; il punto rosso scompare.

Figura 24: selezione unità da analizzare.



0.8.5 Menu: Project

Contiene i comandi la gestione dei progetti e dei dati contenuti nella CPU.

0.8.5.1 Compile

Questo comando è disponibile dopo l'apertura di un progetto.

Converte il progetto in uso in un formato interpretabile dalla CPU. Il progetto potrà essere scaricato nella CPU solo se la compilazione si è conclusa senza errori (Figura 25).

Figura 25: finestra compilazione progetto.

Nel caso di compilazione del progetto conclusa con errori, è possibile visualizzare sull'editor la riga di codice contenente l'errore facendo un doppio click con il mouse sulla segnalazione fornita dalla finestra di compilazione.

0.8.5.2 Force Compile

La funzionalità ha lo scopo di indurre la compilazione forzata di tutto il progetto indipendentemente che esso sia già stato compilato o meno.

0.8.5.3 Ladder Network Checking

Permette eseguire un check della rete Ladder. Sul desktop appare la finestra del risultato della compilazione eseguita (Figura 26).

Figura 26: finestra compilazione rete ladder.

0.8.5.4 View compilation result

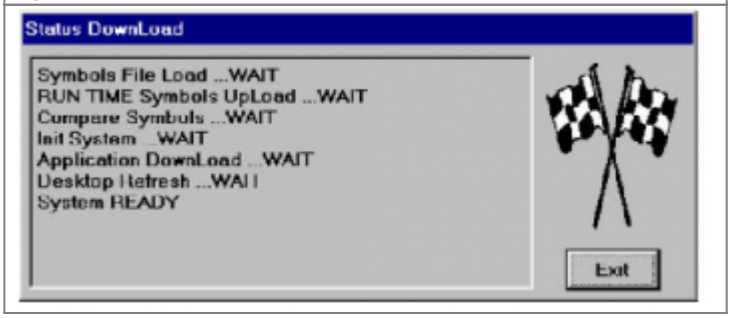
Permette di visualizzare o di togliere dal desktop la finestra del risulatato dell'ultima compilazione eseguita (Figura 24).

0.8.5.5 Download

Questo comando è disponibile solo dopo l'attivazione della comunicazione seriale PC - QMOVE.

Permette di scaricare nella CPU il progetto compilato. Le varie fasi del download vengono visualizzate in una finestra dedicata (Figura 27).

Figura 27: esito download.



0.8.5.6 Backup data

Questo comando è disponibile solo dopo aver eseguito il download dell'applicativo.

L'applicativo che l'utente trasferisce nella CPU è costituito da informazioni non mutabili ed informazioni mutabili.

Informazioni non mutabili Sono le informazioni dell'applicativo che non subiscono variazioni, quali istruzioni QCL, simboli utilizzati nel progetto, titolo applicativo, ...
Questi dati sono memorizzati in flash memory.

Informazioni mutabili Sono le informazioni che indicano situazioni di funzionamento dell'applicativo quali aree dati per devices interni, valore variabili, array system, contenuto datagroup e dati il cui valore viene modificato durante il funzionamento (tutti i dati applicativo). Questi dati sono memorizzati in memoria tamponata.

Il backup è un comando che consente di creare una copia di sicurezza di tutti i dati mutabili, registrandola all'interno della flash-memory. L'utilità di questa operazione deve essere vista nella possibilità di ripristinare tutta la parametrizzazione esistente in un determinato momento. Poichè i valori vengono copiati in flash-memory, esiste la massima sicurezza nel dispositivo di memorizzazione (Figura 28).
Condizioni che permettono l'esecuzione del comando di backup:
- CPU in stato di READY.
- Applicazione che non utilizzi ram per un valore superiore al limite di backup; la quantità di ram disponibile viene definita al momento dell'acquisto scegliendo tra i tagli di memoria disponibili.
- La somma dello spazio occupato dai dati mutabili e non mutabili non deve superare la dimensione della flash-memory.
Se il backup, eseguito da terminale operatore, impiega un tempo maggiore del timeout del terminale, interviene l'errore di timeout error nella comunicazione seriale tra CPU e terminale.

Figura 28: richiesta conferma procedura di backup.

0.8.5.7 Restore data

Questo comando è disponibile solo dopo aver eseguito il download dell'applicativo.

L'applicativo che l'utente richiama dalla CPU è costituito da informazioni non mutabili ed informazioni mutabili.

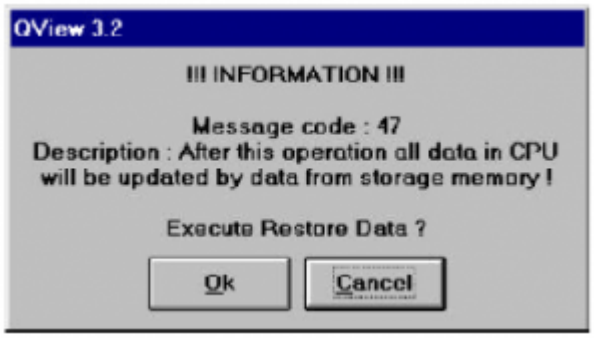
Informazioni non mutabili Sono le informazioni dell'applicativo che non subiscono variazioni, quali istruzioni QCL, simboli utilizzati nel progetto, titolo applicativo, ...
Questi dati sono memorizzati in flash memory.

Informazioni mutabili Sono le informazioni che indicano situazioni di funzionamento dell'applicativo quali aree dati per

devices interni, valore variabili, array system, contenuto datagroup e dati il cui valore viene modificato durante il funzionamento (tutti i dati applicativo).
Questi dati sono memorizzati in memoria tamponata.

Il comando restore consente di ripristinare tutte le informazioni mutabili con quelle presenti al momento del backup (Figura 29). Il contenuto del backup viene cancellato durante la procedura di download; infatti non ha alcun senso copiare i valori dei dati mutabili di un applicativo su un applicativo diverso. Condizioni che permettono l'esecuzione del comando di restore: - Deve essere stato fatto precedentemente un backup.
- CPU in stato di READY o ERROR.

Figura 29: richiesta conferma procedura di restore.

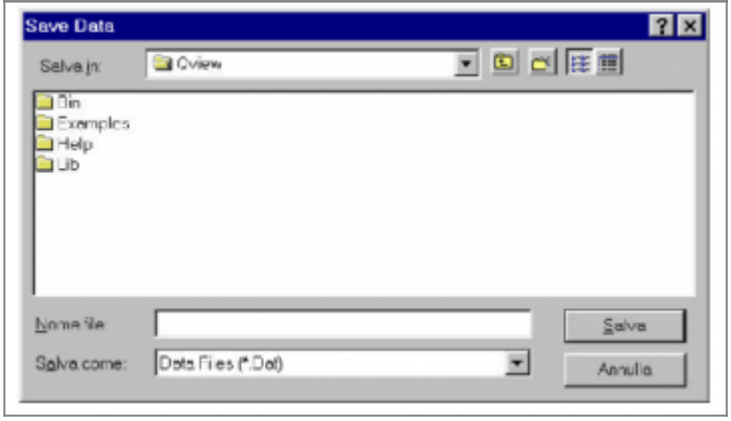


0.8.5.8 Save Data...

Questo comando è disponibile solo dopo aver eseguito il download dell'applicativo.

Salva i dati presenti nella CPU (valori delle variabili ritentive) in un file .DAT; è possibile definire la directory di destinazione del file (Figura 30).

Figura 30: salvataggio dati CPU su file.

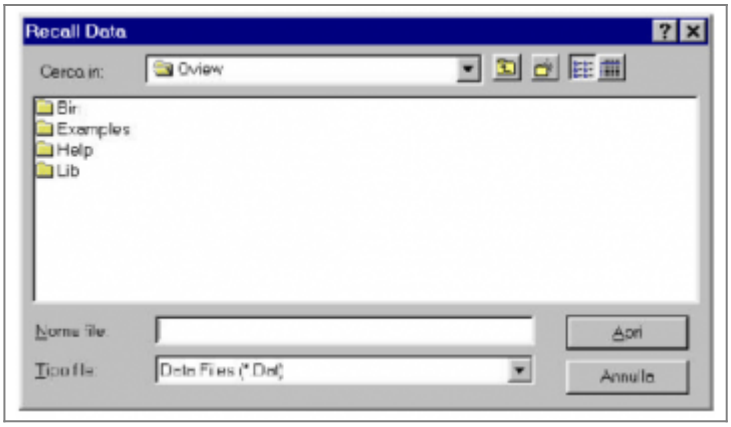


0.8.5.9 Recall Data...

Questo comando è disponibile solo dopo aver eseguito il download dell'applicativo.

Sostituisce i valori delle variabili ritentive presenti nella CPU con i valori delle stesse variabili archiviate in un file .DAT (Figura 31). La sostituzione interessa solamente le variabili in comune all'applicativo ed al file .DAT; se, per esempio, nella CPU sono state aggiunte delle variabili successivamente all'archiviazione dei dati, queste non saranno interessate dalla sostituzione.

Figura 31: sostituzione dati CPU con file .DAT

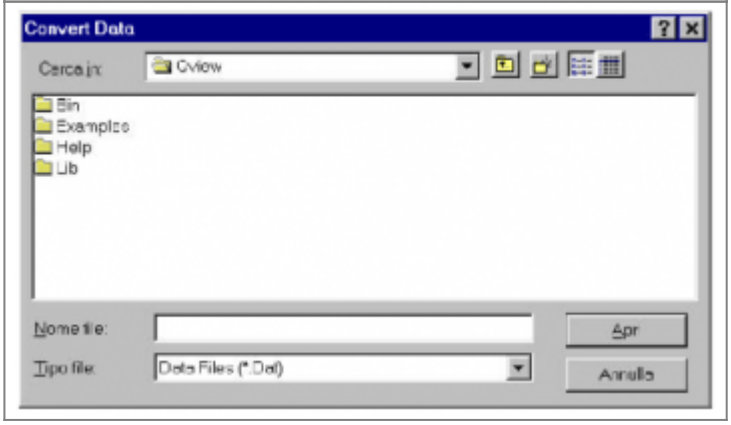


0.8.5.10 Convert Data...

Questo comando è sempre disponibile.

Converte il file .DAT selezionato in un file di testo (.TXT) contenente il nome delle variabili ed il relativo valore. Il file di testo viene generato all'interno della directory che contiene il file .DAT e gli viene assegnato lo stesso nome del file .DAT (Figura 32).

Figura 32: conversione file dati.



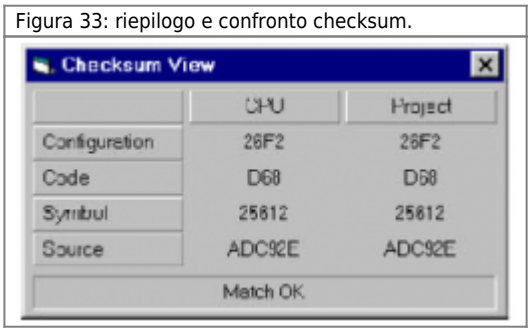
0.8.5.11 Checksum View

Questo comando è disponibile dopo l'apertura di un progetto.

Confronta i checksums del progetto in uso con quelli dell'applicativo scaricato nella CPU (Figura 33), se sono gli stessi vuol dire che la CPU contiene il progetto su cui sto lavorando. Le eventuali diversità vengono segnalate con i valori di checksum in rosso. Vengono confrontati i seguenti tipi di dati:
- Configuration: configurazione memoria utilizzata (dimensione DataGroup, Array, ...).
- Code: codice QCL generato dalla compilazione.
- Symbol: simboli delle variabili utilizzate. Dipende dall'elenco delle variabili e dal loro tipo.
- Source: contenuto delle unità

ATTENZIONE! La compilazione dello stesso progetto con due Qview 4.1 di build diverse garantisce le stesse funzionalità, ma non viene garantito che i codici checksum si mantengano uguali.

Figura 33: riepilogo e confronto checksum.



La colonna CPU visualizza i valori presenti in CPU e rappresentanti il suo applicativo.
La colonna Project viene aggiornata all'apertura del progetto (se precedentemente compilato) e dopo ogni compilazione.

0.8.5.12 Project ID View

Questo comando è disponibile solo dopo l'attivazione della comunicazione seriale PC - QMOVE.

Visualizza il nome assegnato al progetto. Questo nome è stato definito nella finestra Project Information. Durante il download il Project ID viene scaricato nella CPU con l'applicativo ed è poi visualizzabile tramite il comando Project ID View (Figura 34).

Figura 34: visualizzazione stringa progetto.

0.8.6 Menu Debug

Contiene i comandi per l'esecuzione dell'applicativo scaricato nella CPU.

0.8.6.1 Run

Questo comando è disponibile dopo un download.

Mette in esecuzione l'applicativo scaricato nella CPU..

0.8.6.2 Stop

Questo comando è disponibile dopo un run.

Interrompe l'esecuzione dell'applicativo residente nella CPU (i device continuano comunque a funzionare).

0.8.6.3 Restart

Questo comando è disponibile dopo uno start o uno stop.

L'esecuzione dell'applicativo viene interrotta e al RUN successivo il programma riparte dal la prima unità.

0.8.6.4 Reset

Questo comando è disponibile solo dopo l'attivazione della comunicazione seriale PC - QMOVE.

Cancella l'applicativo residente nella CPU. La cancellazione dei dati è definitiva; viene quindi richiesta la conferma al reset dell'applicativo (vedi figura 35).

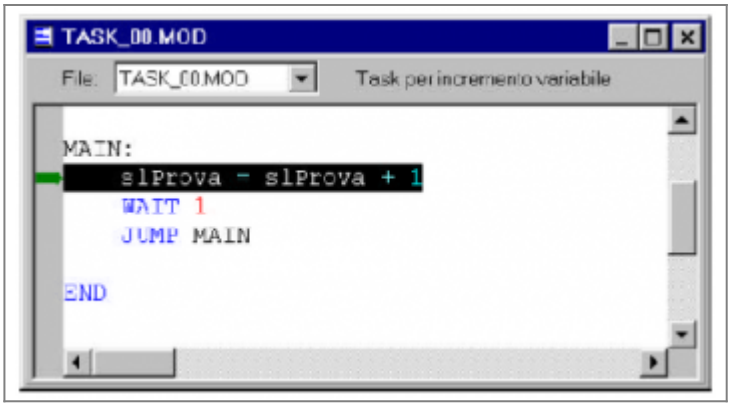
Figura 35: richiesta conferma procedura di reset progetto.

0.8.6.5 Step

Questo comando è disponibile solo dopo un download e selezionando la finestra di editor testo.

Questo comando permette di eseguire il progetto in uso un passo alla volta; ad ogni step l'esecuzione del flusso di programma avanza di un passo. Raggiunta l'istruzione di wait, la finestra di editor visualizza l'unità successivo permettendo in questo modo l'esecuzione del programma un passo alla volta su tutte le unità che compongono il progetto.
Una freccia verde sulla sinistra delle finestra di editor identifica la riga di programma che è stata eseguita (Figura 36). Nel caso di unità LADDER sarà possibile eseguire un rung ad ogni passo.

Figura 36: esecuzione programma in modalità Step.



0.8.6.6 Step Over

Questo comando è disponibile solo dopo un download e selezionando la finestra di editor testo.

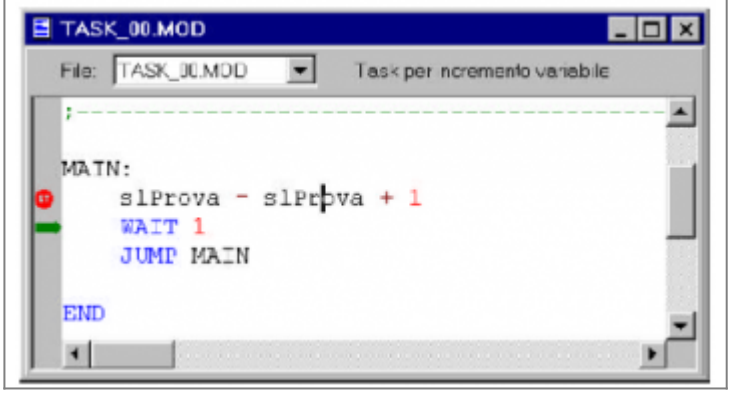
Questo comando permette di eseguire il progetto in uso un'istruzione alla volta; ad ogni step l'esecuzione del flusso di programma avanza di un'istruzione. Raggiunta l'istruzione di wait, la finestra di editor continua a visualizzare l'unità in fase di debug; l'esecuzione del programma continua automaticamente sulle altre unità fino a ritornare a quella in uso riprendendo l'esecuzione passo a passo.
Una freccia verde sulla sinistra delle finestra di editor identifica la riga di programma che è stata eseguita (Figura 37). Nel caso di un'unità LADDER sarà possibile eseguire un rung ad ogni passo.

0.8.6.7 Toggle Breakpoint

Questo comando è disponibile solo dopo un download e selezionando la finestra di editor testo.
E' possibile inserire un massimo di 5 breakpoint. Vengono resettati allo spegnimento della CPU e con il download del progetto.

Questo comando permette di inserire un breakpoint ovvero di identificare una riga di codice, raggiunta la quale, l'esecuzione del programma si interrompe. E' possibile definire più punti di breakpoint. Selezionando una riga di codice sulla quale è stato inserito un breakpoint, il comando breakpoint elimina il breakpoint. L'inserimento del breakpoint viene visualizzato sulla sinistra della finestra di editor con un contrassegno rosso all'interno del quale viene visualizzata la sigla "ST" (Figura 37). E' possibile inserire un breakpoint anche in un'unità LADDER in corrispondenza di uno dei rung.

Figura 37: inserimento brackpoint.



0.8.6.8 Clear All

Questo comando è disponibile solo dopo un download.

Questo comando elimina da tutti i task tutti i breakpoint inseriti.

0.8.6.9 Watchpoint

Questo comando è disponibile solo dopo un download.
E' possibile inserire un massimo di 5 watchpoint. Vengono resettati allo spegnimento della CPU e con il download del progetto.

Questo comando permette di interrompere l'esecuzione del progetto in uso alla soddisfazione di una particolare condizione; si può quindi parlare di breakpoint condizionato. Selezionando il comando watchpoint viene proposta la finestra di figura 38 che visualizza le condizioni di interruzione del programma che sono state impostate.
- Add: permette di aggiungere una condizione di arresto del programma (Figura 39).
- Delete: elimina la condizione selezionata.

- Reset: elimina tutte le condizioni di arresto impostate.

Figura 38: elenco watchpoint.

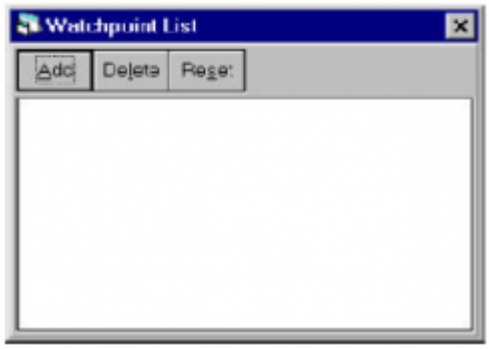
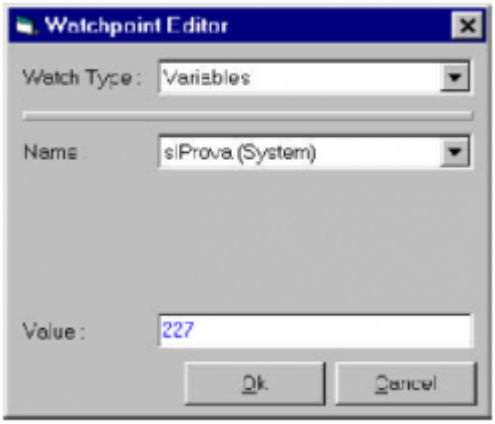


Figura 39: impostazione parametri di watchpoint.



La finestra di figura 39, visualizzata alla pressione del tasto Add (Figura 38), permette di impostare la condizione di arresto del programma.
- Watch Type: selezione la tipologia di variabili (Variables, I/O, Arrays, Data Groups, Devices).
- Name: permette di selezionare la variabile appartenente al gruppo definito nella casella a scorrimento Watch Type.
- Value: permette di inserire il valore della variabile selezionata al quale l'esecuzione del programma si deve arrestare.

0.8.7 Menu Monitor

Contiene i comandi per la diagnostica del sistema QMOVE e dell'esecuzione di progetti.

0.8.7.1 Variables

Questo comando è disponibile dopo l'apertura di un progetto esistente.

La finestra variabili permette di vedere il valore delle variabili in modo dinamico (Figura 40).
- Add: permette di aggiungere alla lista una o piú variabili, dichiarate nel progetto, selezionandole dalla finestra dedicata in cui appare un elenco, in ordine alfabetico, delle variabili (Figura 41). La variabile viene aggiunta come ultimo dato della lista.
- Insert: permette di aggiungere alla lista una o piú variabili, dichiarate nel progetto, selezionandole dalla finestra dedicata in cui appare un elenco, in ordine alfabetico, delle variabili (Figura 41). La variabile viene aggiunta immediatamente sopra la variabile selezionata.
- Delete: elimina dalla lista la variabile selezionata.
- Modify: permette di modificare il valore di una variabile senza aggiungerlo alla lista (figure 41 e 42).

Figura 40: lista delle variabili e valori.

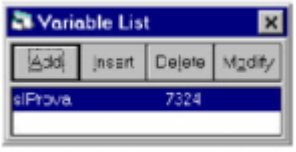


Figura 41: elenco in ordine alfabetico delle variabili da aggiungere o modificare.

Figura 42: modifica del valore della variabile selezionata.

0.8.7.2 I/O

Questo comando è disponibile dopo l'apertura di un progetto esistente.

Al pari della finestra variabili, la finestra I/O permette di vedere e modificare lo stato degli ingressi/uscite. Per la funzionalità dei tasti Add, Insert, Delete e Modify fare riferimento a quanto descritto nella finestra variabili.

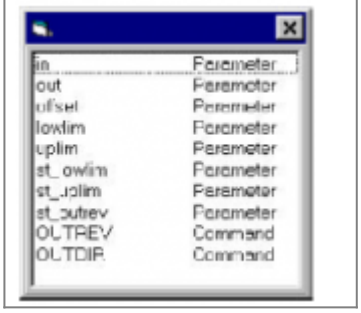
0.8.7.3 Devices

Questo comando è disponibile dopo l'apertura di un progetto esistente.

La finestra device visualizza tutti i device interni dichiarati nel file di configurazione; vengono visualizzati il nome assegnato al device ed il tipo di device (Figura 43).

Figura 43: elenco device.

Figura 44: elenco parametri device.



Selezionando un device (con il tasto ENTER o con doppio clic) viene proposta una finestra (Figura 44) all'interno della quale è possibile visualizzare i dati del device; con il tasto Add è viene richiamata una finestra contenente i parametri ed i comandi propri del device selezionato. Selezionando un parametro è possibile modificarlo.

0.8.7.4 Data Groups...

Questo comando è disponibile dopo l'apertura di un progetto esistente.

La finestra Data Groups (Figura 45) visualizza l'elenco dei data groups utilizzati; selezionandone uno viene proposta la finestra di figura 46.

Figura 45: elenco data groups.

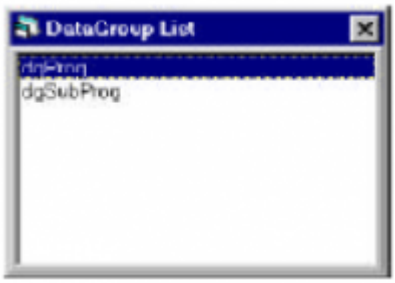
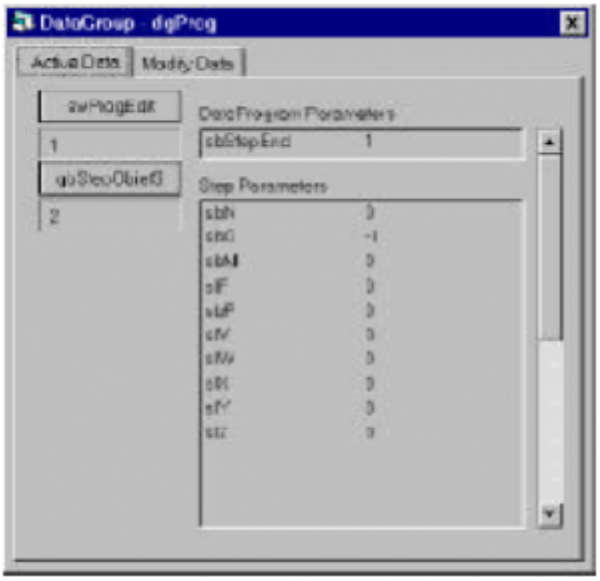


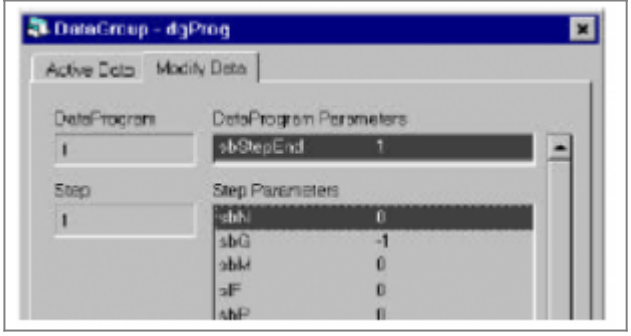
Figura 46: visualizzazione dati data groups.



- Il pulsante (swProgEdit di figura 46) permette di selezionare la variabile dedicata alla definizione del programma del data group da mettere in uso.
- Il pulsante (gbStepObietS di figura 46) permette di selezionare la variabile dedicata alla definizione del passo di programma da mettere in uso.
- La finestra DataProgram Parameters visualizza le variabili static associate al programma in uso (variabili il cui valore è fisso per ogni passo di programma); la finestra Step Parameters visualizza tutte le variabili index associate al passo in uso (variabili il cui valore può essere diverso di passo in passo).

Selezionando la cartella Modify Data, viene proposta la finestra di figura 47; i dati visualizzati sono gli stessi della cartella Active Data ma è possibile modificarli (selezionare il valore da modificare, introdurre il nuovo valore e confermare con ENTER).

Figura 47: modifica dati data groups.



0.8.7.5 Arrays

Questo comando è disponibile dopo l’apertura di un progetto esistente.

La finestra Array List (Figura 48) visualizza l’elenco degli Array utilizzati; selezionandone uno viene proposta la finestra di figura 49.

Figura 48: elenco arrays

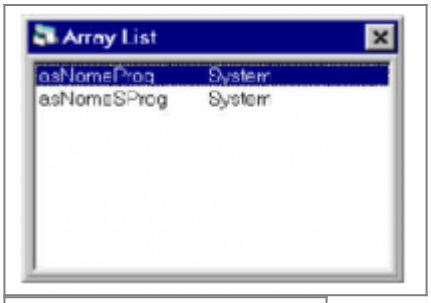


Figura 49: modifica array.

Dalla finestra di figura 49 è pssibile modificare un elemento dell'array.

0.8.7.6 Timers

Questo comando è disponibile dopo l'apertura di un progetto esistente.

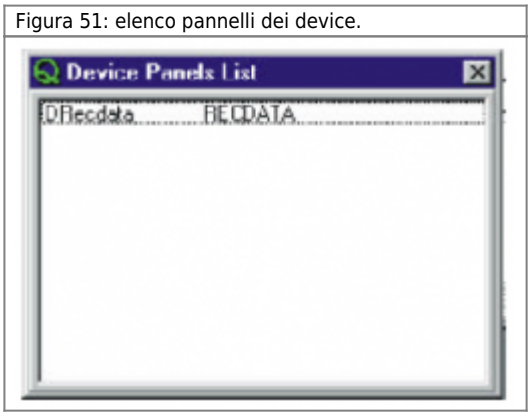
La finestra Timers List (Figura 50) visualizza l'elenco dei timers utilizzati. Per ogni timer viene visualizzato il nome assegnato, il tempo impostato ed il tempo (1) rimanente prima dello scadere del timer (remain).

Figura 50: elenco timers.

0.8.7.7 Devices Panels

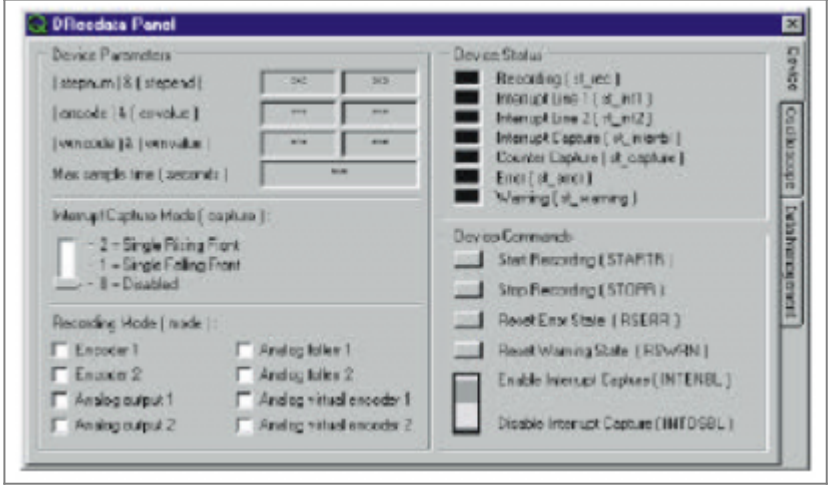
L'ambiente Qview prevede che per i devices ci sia, oltre che alla lista di stati parametri comandi, un pannello che permetta all'utilizzatore un semplice approcio con il devices stesso. vengono visualizzati il nome assegnato al device ed il tipo di device (Figura 51).

Figura 51: elenco pannelli dei device.



Selezionando un device (con il tasto ENTER o con doppio clic) viene proposta una finestra (Figura 52) all'interno della quale è possibile visualizzare i dati del device; per una spiegazione dei devices panel, fare riferimento all'apposito help.

Figura 52: esempio di un pannello.

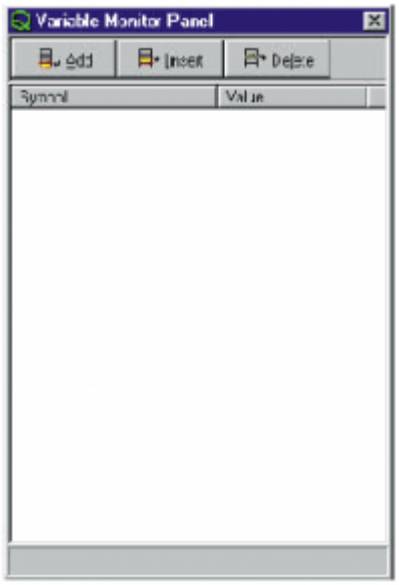


0.8.8 General Panels

0.8.8.1 Variabile monitor

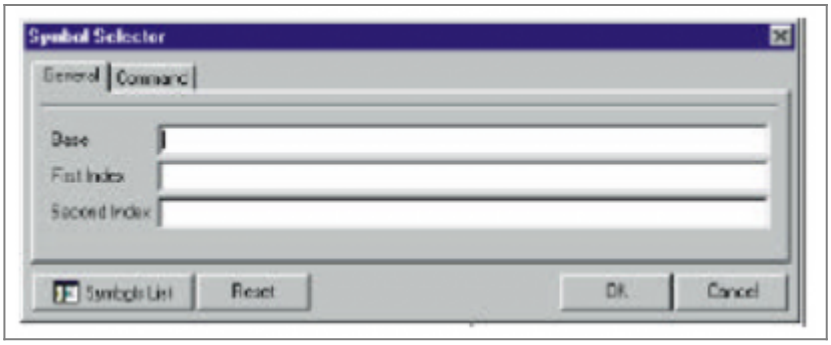
L'ambiente Qview 4 mette a disposizione un pannello generale dove vi è la possibilità di inserire variabili, parametri di devices diversi, ingressi, uscite, etc. (Figura 53).

Figura 53: pannello variabili generale.



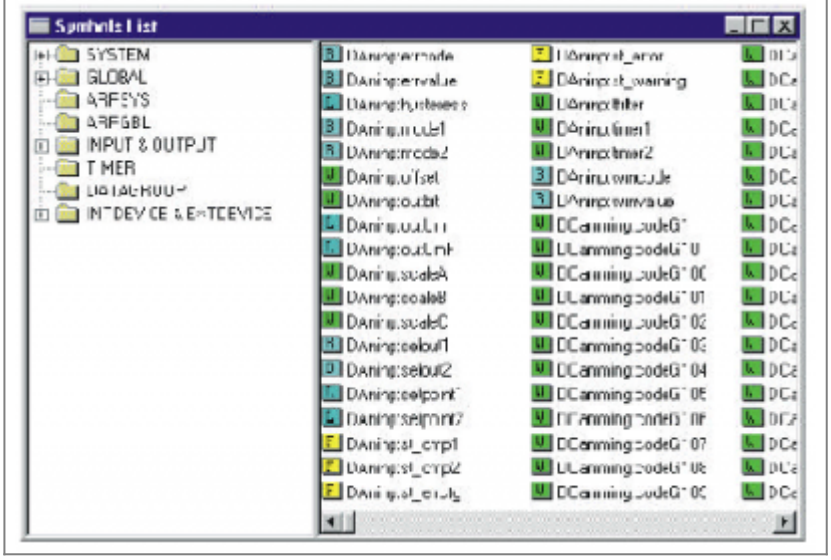
- Add: permette di aggiungere alla lista una o piú variabili, dichiarate nel progetto, inserendone il nome nella finestra dedicata (Figura 54). La variabile viene aggiunta come ultimo dato della lista.
- Insert: permette di aggiungere alla lista una o piú variabili, dichiarate nel progetto, inserendone il nome nella finestra dedicata (Figura 54). La variabile viene aggiunta immediatamente sopra la variabile selezionata.
- Delete: elimina dalla lista la variabile selezionata.

Figura 54: finestra per l'inserimento del simbolo nella lista.



Selezionando il pulsante di Symbol list è possibile visualizzare la lista dei simboli del progetto (Figura 55). Selezionando il simbolo viene inserito automaticamente nella lista.

Figura 55: pannello lista variabili.



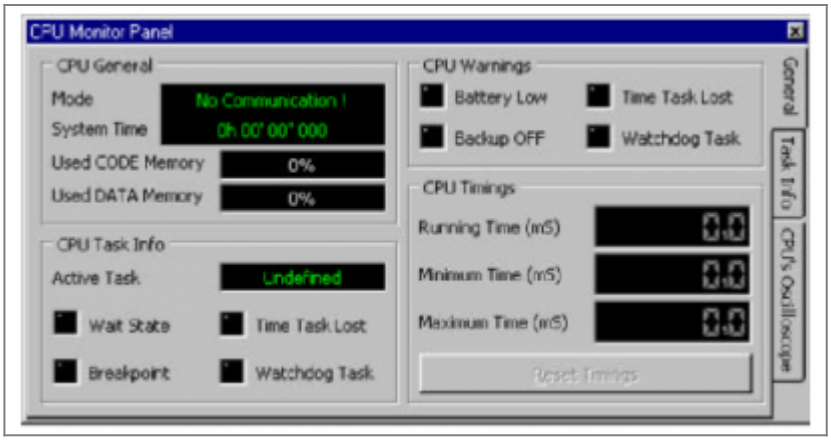
0.8.8.2 CPU MONITOR

La presente funzionalità integra e completa la caratteristica MONITOR CPU presente su tutti i Qview.
A questa nuova funzionalità vi si accede tramite “MONITOR - GENERAL PANEL - CPU MONITOR”. La finestra CPU Monitor Panel si compone di tre sezioni:
- GENERAL
- TASK INFO
- CPU’s oscilloscope

0.8.8.3 General

Si trovano nella pagina principale tutti gli stessi dati presenti nel menù Monitor Cpu che già conosciamo, completati però da MINIMUM TIME e MAXIMUM TIME, che a livello firmware danno informazioni, sul minimo e massimo tempo, impiegato per elaborare il programma generato (ciclo task).
Vi è inoltre un tasto di Reset Timing il quale azzera i due valori ed il Running Time riabilitando il conteggio del tempo di elaborazione.

Figura 56: CPU general panel.



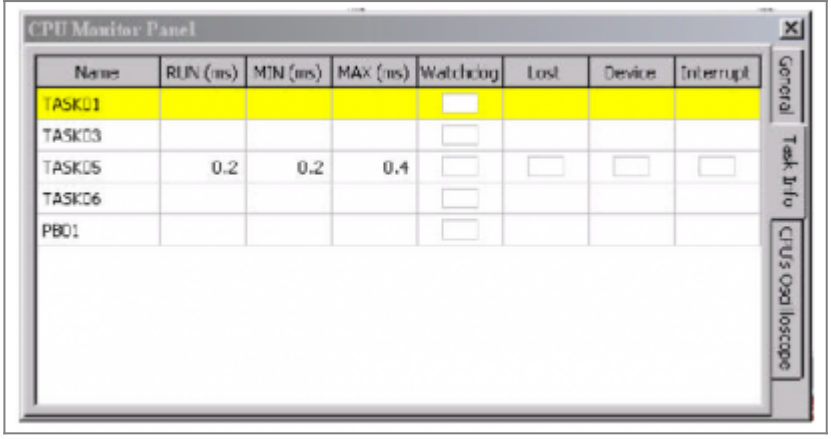
Task info

In questo menù possono essere controllati tutti i task (unità) elaborati dal programma, e possono essere ricavate informazioni di vario genere. Tutti i task possono dare infomazioni sul tempo di esecuzione minimo, massimo ed attuale.

Per i task normali si possono avere inoltre informazioni relative a:
WATCHDOG:Indica che il task è stato eseguito per più di 200ms. E' la stessa informazione che compare nel pannellino CPU alla voce "Watchdog active".

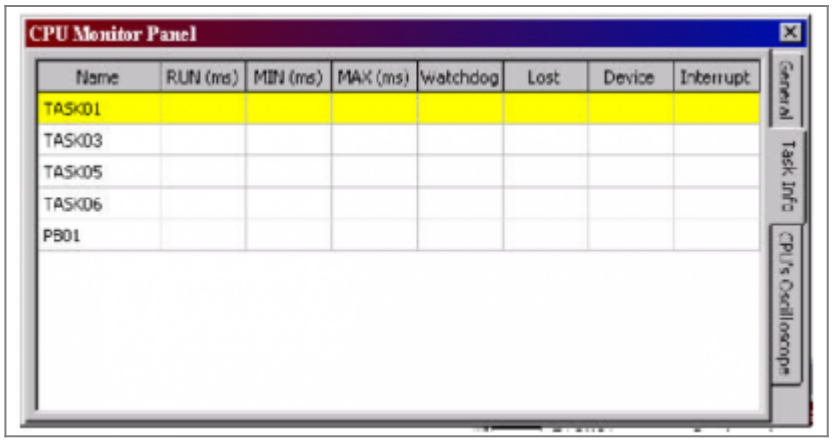
Per tutti gli altri task si possono avere informazioni relative a :
WATCHDOG: Indica che il task speciale è stato eseguito per più di 200ms. E' la stessa informazio ne che compare nel pannellino CPU alla voce " Watchdog active " (Figura 57).
TASK LOST: Indica che il task a tempo ha perso un evento. E' la stessa informazione che compare nel pannellino CPU alla voce "Time task Lost" (Figura 57).
DEVICE ACCESS: Indica che il task ha eseguito un accesso a device e che questo non era aggiornato.
Questa particolarità è identica ai "Wait forzati" dei task normali ma in questi task speciali il task rimane in attesa che il device si aggiorni nel prossimo tempo di campionamento (Figura 57).
INTERRUPT: Indica che un task a tempo è stato interrotto da un task in interrupt (Figura 57).

Figura 57: CPU general panel.



Tutte le informazioni della videata precedente (Figura 57) possono essere visualizzate solo se i check sum del programma in CPU e su PC corrispondono, altrimenti, nella videata di Task Info verranno visualizzati solamente i task di programma senza alcun'altra informazione (Figura 58).

Figura 58: CPU Task info neutra.

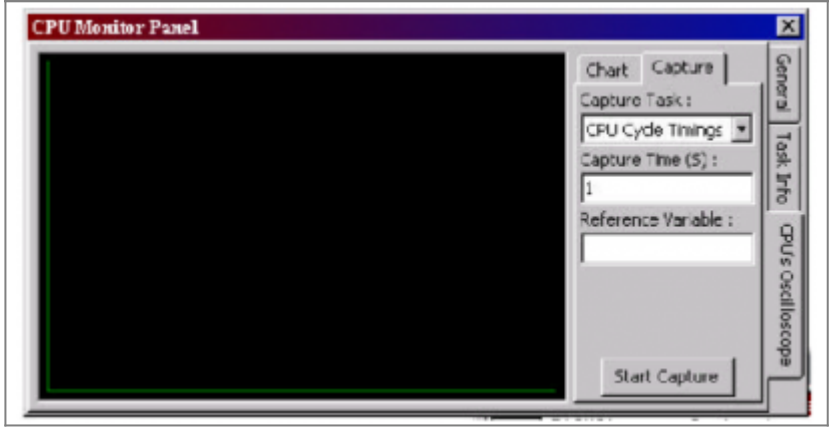


Nel caso di errori, i vari campi assumeranno il colore rosso, in caso contrario, i campi rimarranno di colore bianco.

CPU's Oscilloscope

L'oscilloscopio permette di visionare nel tempo cio' che accade alla CPU (tempo massimo di campionamento 9999Sec). (Figura 59).

Figura 59: CPU Oscilloscopio.



La velocità di acquisizione è determinata da tre fattori:
- Velocità della seriale
- Tipo di PC
- Quanti dati vogliamo controllare

Esempio:
Con un PC Pentium 200MHz, velocità di seriale massima, sono stati ottenuti 7msec. quale tempo di scansione.

Se si vuole fare un'acquisizione si abilita il Capture e la scansione inizia continuando per il tempo impostato, comunque non superiore ai 9999 sec.
Quando si lancia una acquisizione tutto il resto del Qview viene bloccato proprio per non togliere risorse a questa funzionalità.
L'acquisizione può essere associata ad una variabile qualunque, tranne un datagroup, e l'acquisizione nel tempo può essere abilitata anche con macchina in funzione senza pregiudicarne il funzionamento in quanto il controllo non grava sull'elaborazione.
L'acquisizione può essere fermata in qualunque momento tramite la pressione del tasto Backspace.

Impostazioni per la memorizzazione:

CAPTURE TASK: Questo parametro determina in quale campo agire per effettuare la misurazione:
- Su tutti i task , se impostato come CPU CICLE TIMINGS , in questo modo possiamo vedere il tempo di ciclo di tutto il programma (Min. Max. Run). - Su un task singolo, se impostato come task a tempo o ad interrupt, in questo modo possiamo vedere il tempo di ciclo del singolo task seleziona to (Min. Max. Run). Naturalmente se il task selezionato risulta essere un task normale il tempo di scansione che verrà visualizzato sara' quello di tutto il programma, questo particolare caso comunque viene evidenziato con l'ausilio di un messaggio a terminale (Figura 60).

Figura 60: CPU Oscilloscopio Capture Task.

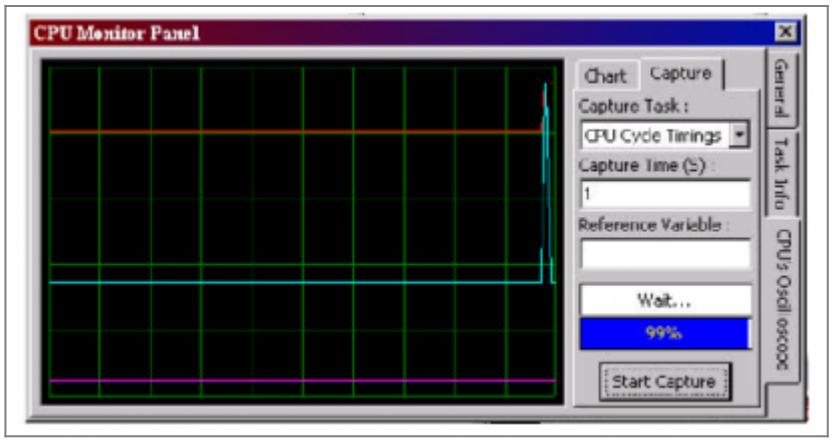
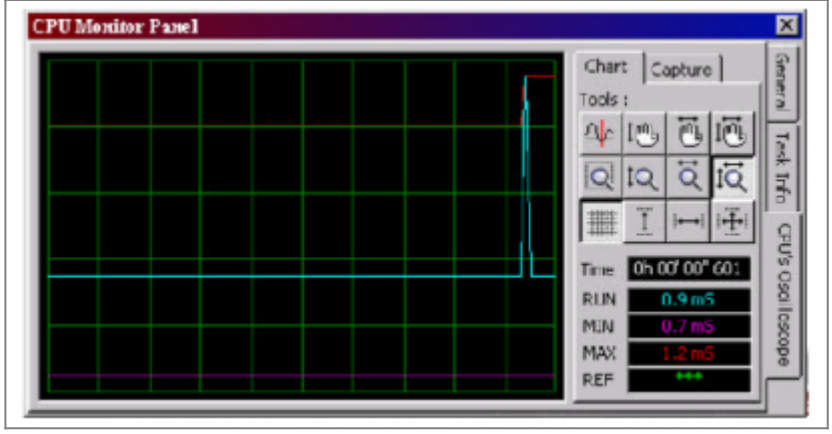


CHART: Questa finestra serve per impostare al meglio le caratteristiche grafiche di visualizzazione ed i riferimenti relativi alle misurazioni effettuate (Figura 61).

Figura 61: CPU Oscilloscopio Chart.



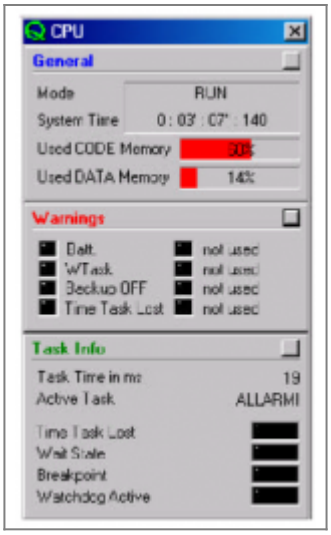
Per conoscere il significato dei vari tasti basta posizionarsi sopra al tasto interessato con il cursore del mouse e comparirà la descrizione specifica.
Oltre alle misure delle tempistiche dei task è possibile monitorare l'andamento nel tempo di una variabile di riferimento che si puo specificare nella casella "Reference variable" nella sezione Capture.
Se la variabile non esiste compare un messaggio di segnalazione, ma la registrazione viene compiuta ugualmente senza, naturalmente, la variabile.

0.8.8.4 CPU

Questo comando è disponibile dopo l'apertura di QVIEW.

La finestra CPU (Figura 62) monitorizza lo stato della CPU.
- Mode: visualizza lo stato della CPU (RUN, STOP, ... Err). in caso di errore viene visualizzato anche il tipo di errore alla scheda firmware della CPU per ulteriori dettagli.
- System Time: visualizza il tempo reale di funzionamento dell'applicativo dal momento dell'accensione o restart della CPU.
- Used CODE Memory: visualizza la percentuale di memoria impegnata dal codice.
- Used DATA Memory: visualizza la percentuale di memoria impegnata dai dati.
- Batt: segnala lo stato di batteria tampone interna scarica.
- WTask: segnala che vengono impiegati più di 200 ms per eseguire un task. Questa segnalazione viene mantenuta fino allo spegnimento della CPU. Il progetto continua a funzionare correttamente, tuttavia la segnalazione indica un'errata programmazione QCL.
- SBackup OFF: segnala l'impossibilità di eseguire il backup (ad esempio perchè viene impegnata una percentuale elevata di DATA Memory).
- Time Task Lost: segnala che nell'esecuzione del programma non è stato eseguito una o più volte un task a tempo.
- Task Time in ms: tempo di esecuzione dei task.
- Active Task: task in uso.
- Time Task Lost: segnala che nell'esecuzione del programma non è stato eseguito una o più volte un task a tempo.
- Wait State: segnala il raggiungimento dell'istruzione di wait (all'interno del task in uso).
- Breakpoint: segnala che il flusso del task attivo ha raggiunto un breackpoint e si è interrotto.
- Step by Step: segnala che il task in uso è bloccato per l'esecuzione passo-passo.
- Watchdog Active: segnala che il task attivo ha causato la segnalazione di Watchdog.

Figura 62: monitor CPU.

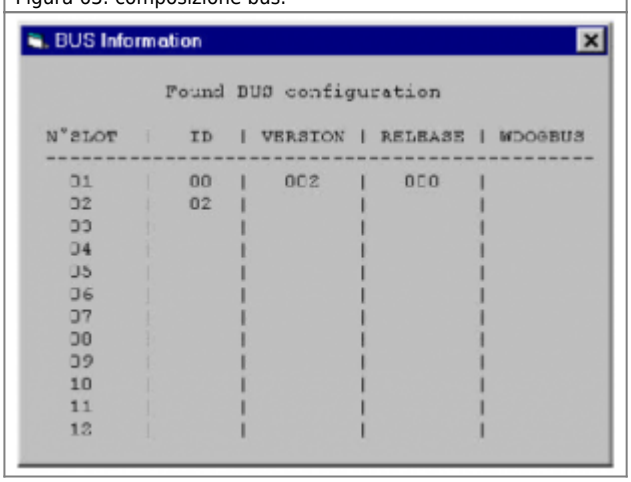


0.8.8.5 BUS

Questo comando è disponibile dopo l'apertura di un progetto esistente e l'apertura della porta di comunicazione seriale PC - QMOVE.

La finestra BUS Information (Figura 63) visualizza la composizione hardware e le versioni/release dei firmware. Queste informazioni vengono acquisite dalla CPU tra il BUS; per questo motivo sono disponibili solamente se la comunicazione è attiva.
- N° SLOT: posizione nello slot.
- ID: identifica il tipo di scheda installata.
- VERSION: indica la versione del firmware installato sulla scheda intelligente.
- RELEASE: indica la release del firmware installato sulla scheda intelligente.
- WDOGBUS: segnala le eventuali anomalie di funzionamento delle schede.

Figura 63: composizione bus.



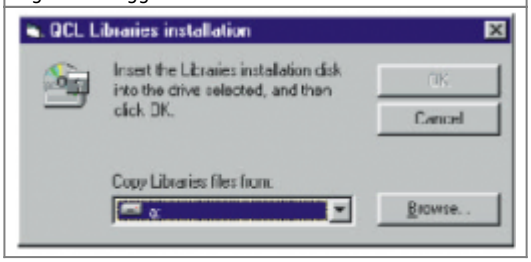
0.8.9 Menu Tools

0.8.9.1 Upgrade QCL Card Library

Questo comando è sempre disponibile.

Il comando Upgrade QCL Card Library permette di aggiornare le librerie del linguaggio QCL (Figura 64).

Figura 64: aggiornamento librerie.

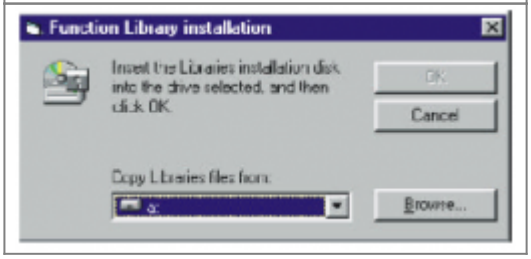


0.8.9.2 Upgrade QCL Functions Library

Questo comando è sempre disponibile.

Il comando Upgrade Functions Library permette di aggiornare le librerie di funzioni QCL (Figura 65).

Figura 65: aggiornamento librerie delle funzioni QCL.

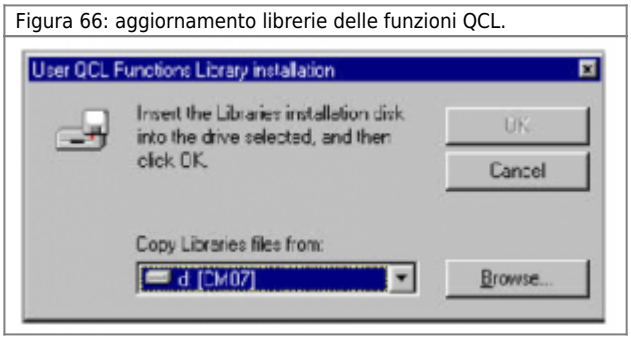


0.8.9.3 Upgrade User QCL Functions Library

Questo comando è sempre disponibile.

Il comando Upgrade User Functions Library permette di aggiornare le librerie di funzioni QCL create del programmatore (Figura 66). Per maggiori informazioni sulle funzioni create dal programmatore si veda il capitolo “Funzioni di libreria QCL”.

Figura 66: aggiornamento librerie delle funzioni QCL.

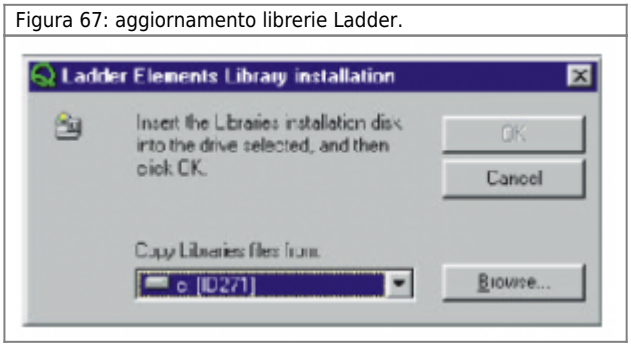


0.8.9.4 Upgrade Ladder Elements Library

Questo comando è sempre disponibile.

Il comando Upgrade ladder elements permette di aggiornare le librerie degli elementi ladder (Figura 67).

Figura 67: aggiornamento librerie Ladder.



0.8.10 Menu Options

I comandi del menu Options permettono di personalizzare alcune funzionalità di QVIEW.

0.8.10.1 Open COM / Close COM

Permette di aprire e chiudere la porta di comunicazione seriale PC - QMOVE.

0.8.10.2 Program Setup ...

Questo comando è sempre disponibile.

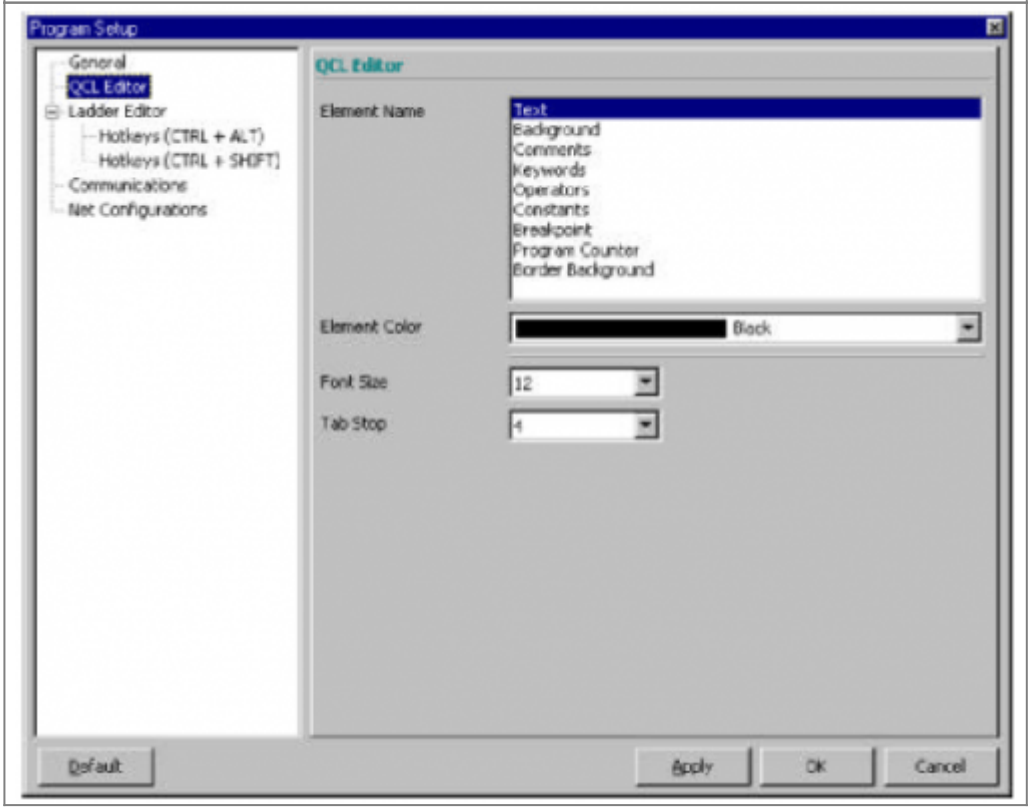
Viene presentata una finestra composta da sei diverse cartelle.

Cartella QCL Editor

Questo comando è sempre disponibile.

Permette di personalizzare l'interfaccia grafica di QVIEW, assegnando dimensioni e colori diversi in funzione del testo QCL da editare (Figura 68).

Figura 68: personalizzazione editor QCL.

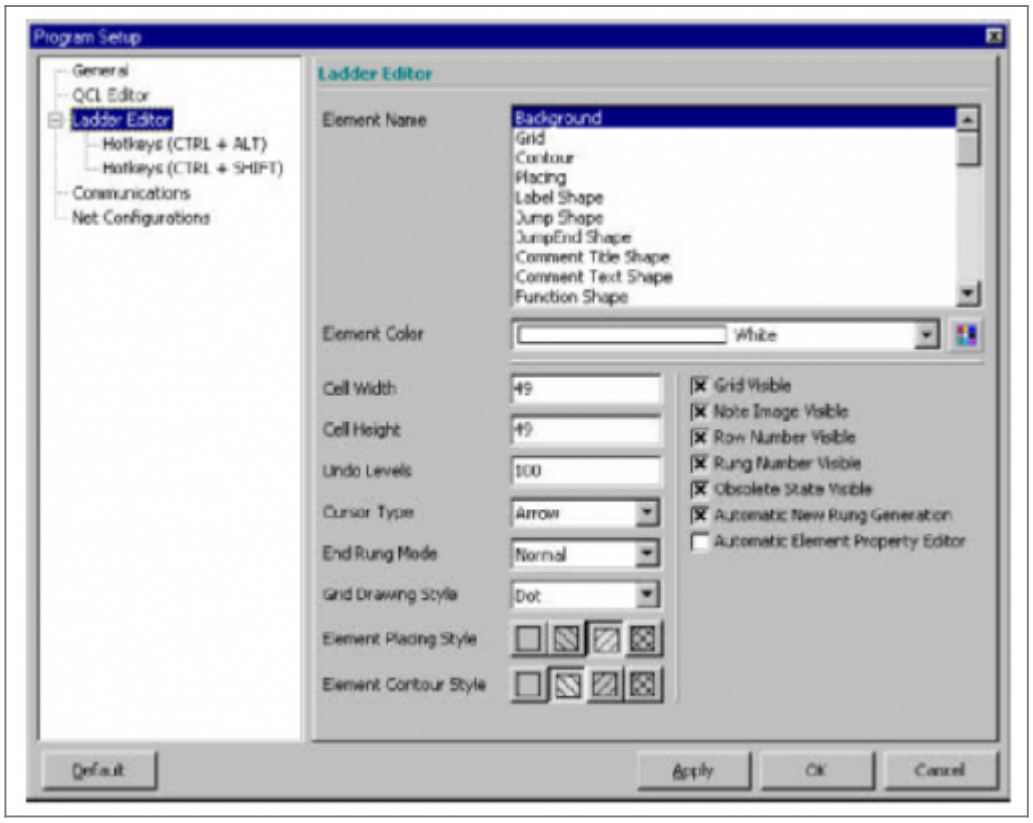


- Font Size: definisce le dimensione del font utilizzato nella finestra di editor.
- Tab Stop: definisce la lunghezza della tabulazione.
- Text Color: definisce il colore del testo (codice QCL).
- Element Color: permette di specificare il colore dell'elemento selezionato.
- Element Name: permette di selezionare gli elementi dell'editor per specificarne il colore. Gli elementi selezionabili sono:
- Text: testo presente nell'editor.
- Background: sfondo della finestra di editor.
- Comments: commentipresenti nel programma.
- Keywords: parole chiave.
- Operators: operatori.
- Constant: costanti.
- Breakpoint: simbolo di breakpoint.
- Program Counter: puntatore di programma (visibile solamente in condizioni di stop).
- Border Background: bordo della finestra di editor.

Cartella Ladder Editor

Permette di personalizzare l'interfaccia grafica di QVIEW, assegnando dimensioni e colori diversi agli elementi che conpongono l'editor LADDER (Figura 69).

Figura 69: personalizzazione editor Ladder.

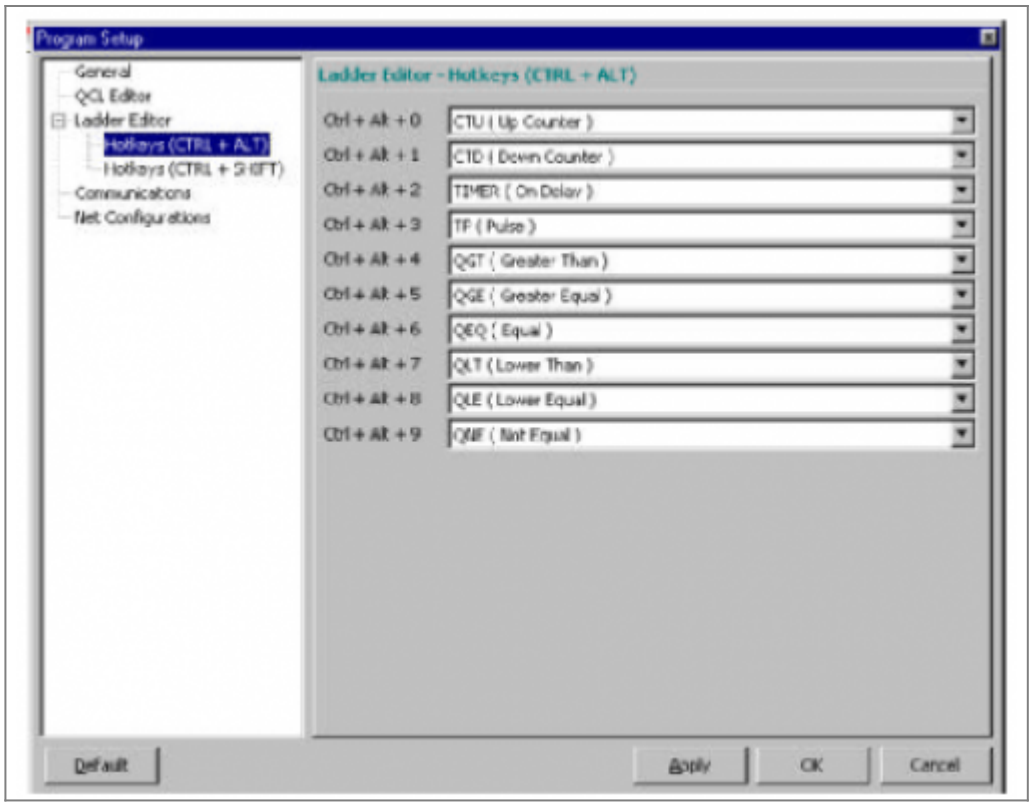


- Cell Width: definisce la larghezza delle caselle dell'editor.
- Cell Height: definisce l'altezza delle caselle dell'editor.
- Undo Levels: numero massimo di UNDO possibili.
- Cursor type: scelta del tipo di cursore.
- End Rung Mode: scelta del tipo di fine rung.
- Grid Drawing Style: scelta del tipo di tratto utilizzato per disegnare la griglia.
- Element Placing Style: grafica utilizzata come sfondo degli elementi LADDER selezionati.
- Element Contour Style: grafica utilizzata come sfondo dell'area dell'editor selezionata.
- Grid Visible: seleziona se rendere visibile la griglia.
- Note image Visible: seleziona se rendere visibili il simbolo che indica la presenza di una nota associata ad un elemento LADDER.
- Row Number visible: seleziona se rendere visibile i numeri di riga.
- Rung Number visible: seleziona se rendere visibile i numeri di rung.
- Obsolete State Visible: seleziona se indicare gli elementi LADDER obsoleti (cambiandone i lcolore di sfondo).
- Automatic New Rung Generation: seleziona se rendere possibile la generazione automatica dei rung al momento dell'inserimento di un nuovo elemento LADDER.
- Automatic Element Property Editor: seleziona se far apparire automaticamente la finestra con le proprietà dell'elemento LADDER una volta che viene inserito.
- Color: definisce il colore dello sfondo della finestra di editor.

- Element Color: permette di specificare il colore dell'elemento selezionato.
- Element Name: permette di selezionare gli elementi dell'editor per specificarne il colore.
Gli elementi selezionabili sono elencati in questa finestra.

In questa sezione del "Program Setup" è possibile associare ad una serie di HotKeys (Tasti Caldi) l'inserimento degli elementi LADDER più utilizzati. Nella Figura 70 è possibile vedere che è possibile associare l'inserimento di 10 elementi LADDER alla combinazione dei tasti CTRL + ALT + 0…9 e altri 10 a CTRL + SHIFT + 0…9. Esiste già un associamento per default, ma il programmatore potrà modificarlo.

Figura 70: associazione ai tasti caldi.

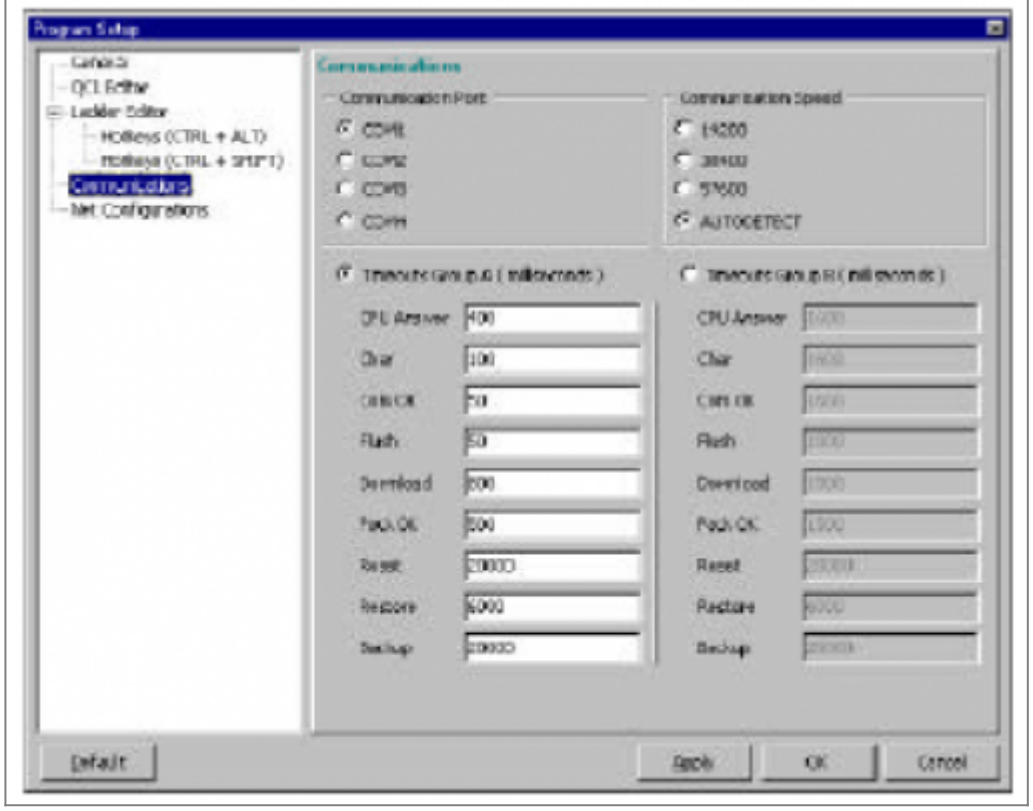


Cartella Communication

Per utilizzare le porte COM3 e COM4 è necessario adottare degli IRQ diversi da quelli utilizzati per altri dispositivi hardware (mouse, COM1, COM2, ...).

Permette di configurare la porta di comunicazione seriale (Figura 71).

Figura 71: impostazione parametri porta di comunicazione seriale.



- COM Port: permette di selezionare la porta di comunicazione seriale da utilizzare. - Communication Speed: definisce la velocità di trasmissione della comunicazione seriale; selezionando l'opzione autodetect, QVIEW rileva automaticamente la velocità di trasmissione da adottare.

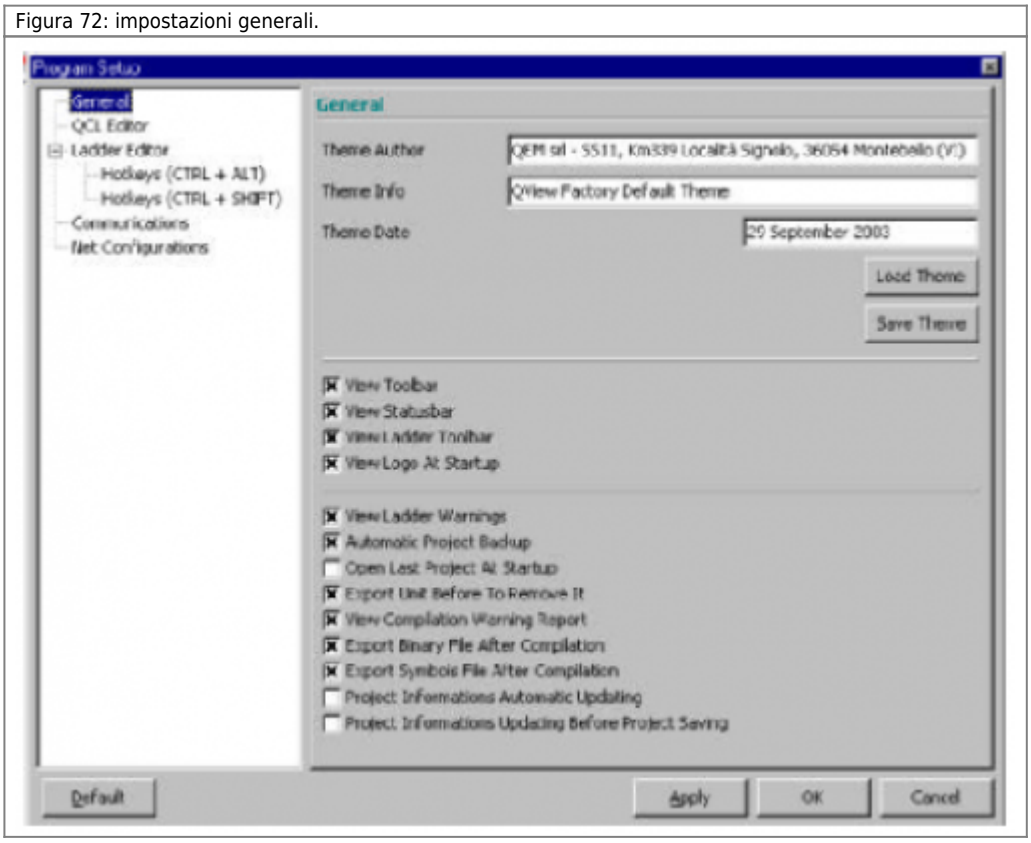
E’ possibile selezionare tra due gruppi di impostazione dei tempi della comunicazione seriale (Timeouts Group A e Timeouts Group B):

- CPU Answer: timeout sulla risposta del terminale.
- Char: timeout tra carattere e carattere.
- Com Ok: timeout sulla verifica di connessione seriale.
- Flush: timeout su invio richiesta al terminale.
- Download: timeout su conclusione procedura di download al terminale.
- Pack Ok: timeout ricezione pacchetto informazioni.
- Reset: timeout per risposta della CPU al comando di reset.
- Restore: timeout per risposta della CPU al comando di restore.
- Backup: timeout per risposta della CPU al comando di backup.

Cartella General

Permette di personalizzare l’ambiente di sviluppo (Figura 72).

Figura 72: impostazioni generali.



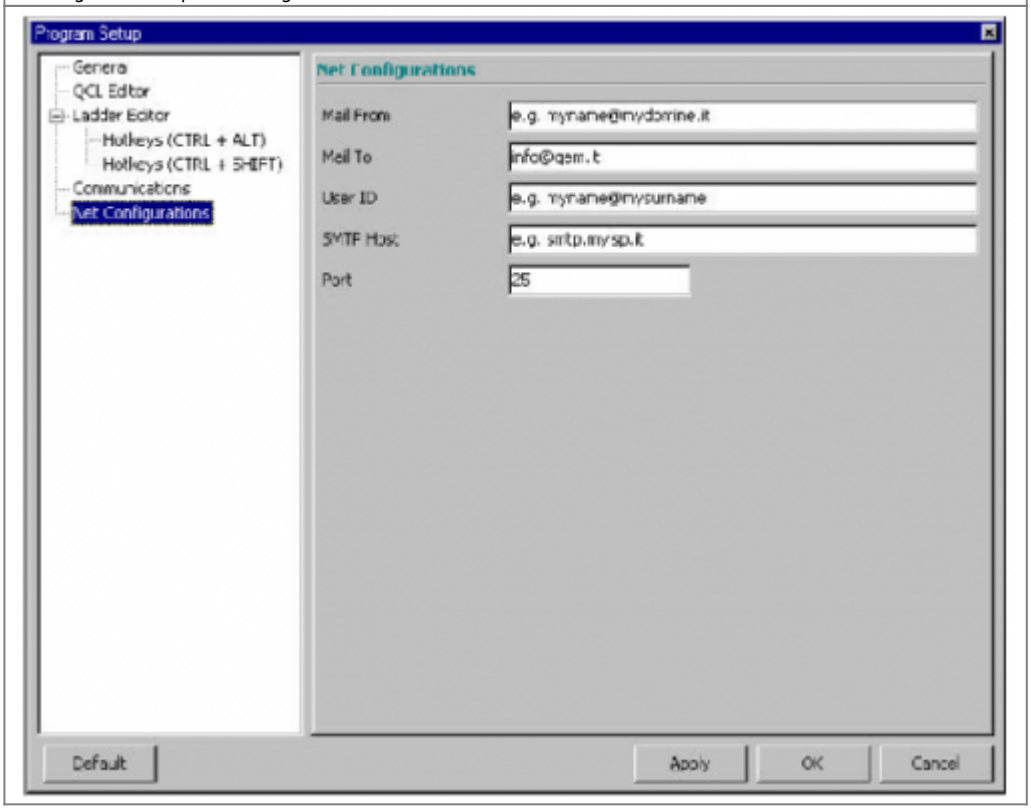
- View Toolbar: abilita la visualizzazione della toolbar del progetto.
- View Status Bar: abilita la visualizzazione della barra di stato.
- View Ladder Toolbar: abilita la visualizzazione della toolbar del ladder.
- View Logo At Startup: abilita la visualizzazione del logo Qview 4.1 allo startup del programma.
- View Ladder Warnings: abilita la visualizzazione dei messaggi di warning nella finestra con i risultati della compilazione.
- Automatic Project Backup: abilita l’esecuzione di una copia di backup del progetto ogni volta che viene salvato.
- Open the last project at startup: abilita l’apertura dell’ultimo progetto salvato ad ogni apertura del Qview.
- Export unit before remove it: abilita la richiesta di esportazione dell’unità prima della cancellazione della stessa.
- View Compilation Warning report: abilita la visualizzazione dei messagi di warning durante la compilazione.
- Export Binary file After Compilation: abilita l’esportazione automatica del file binario risultato della compilazione.
- Export symbol file after compiling: abilita l’esportazione automatica del file simboli ad ogni compilazione andata a buon fine. Il file simboli verra' esportato nella stessa directory del file di progetto con lo stesso nome del file di progetto. Se il progetto è nuovo e non ancora salvato su disco, Qview genererà una finestra di dialogo per chiedere dove ubicare il file simboli.
- Automatic Project Information update: se selezionato appare un messaggio in cui si chiede all’utente se le informazioni di progetto contenute in “Project information” sono state aggiornate.
Questo messaggio appare se l’applicativo non viene salvato da almeno un’ora.
- Project Information update before project saving: se selezionato, ogni volta che si salva il progetto, appare un messaggio in cui si avvisa che i dati obbligatori (segnalati con un asterisco) da inserire in “Project information” non sono completi.

In questa finestra è possibile salvare le impostazioni in un file e specificare anche alcuni dati relativi all’autore.

Cartella Net Configurations

Permette di impostare i dati necessari ad inviare un'email direttamente da Qview4 senza l'ausilio di un mail client esterno (Figura 73).

Vedi figura 73: impostazioni generali.



- Mail From...: in questo campo è possibile impostare l'indirizzo da dove la mail dovra' partire. Tale indirizzo potrebbe essere il proprio indirizzo email.
- Mail To...: è l'indirizzo di chi riceverà la mail. Solitamente va impostato come support@qem.it, ma è possibile impostare liberamente anche altri indirizzi email.
- User ID: ID Utente necessario per effettuare il collegamento al provider di servizi internet (ISP).
- SMTP Host: è l'indirizzo del server di posta elettronica.
- Port: è la porta IP di comunicazione. Generalmente il valore 25 è corretto per la stragrande maggioranza dei casi.

0.8.11 Menu Help

Questo comando è sempre disponibile.

Dal menu help è possibile richiamare gli help in linea e alcune informazioni su QVIEW.

0.8.11.1 Contents

Richiama l’help in linea relativo al linguaggio di programmazione.

0.8.11.2 Qview4 Help

Consultare il manuale del Qview 4.1 nella sezione “Sviluppo applicativi” presente nel CDQEM.

0.8.11.3 QCL Language guide

Richiama l’help in linea per la programmazione in QCL.

0.8.11.4 Ladder Functions info

Richiama l’help in linea relativo agli elementi LADDER.

0.8.11.5 Functions info

Richiama l’help in linea relativo alle funzioni QCL.

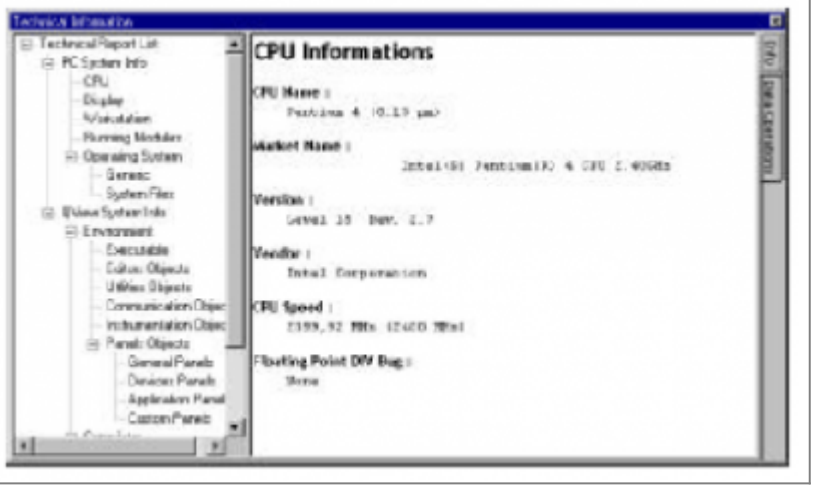
0.8.11.6 User Functions info

Richiama l’help in linea relativo alle funzioni QCL realizzate dall’utente.

0.8.11.7 Technical Info

Per accederte a Technical Info bata selezionare: TECHNICAL INFO. Viene visualizzata la finestra di figura 74 dove vengono raccolte tutte le informazioni del Personal Computer nel quale è istallato il Qview, complete di impostazioni e setup particolari , vengono inoltre registrate le DLL utilizzate per permettere di avere il massimo numero di informazioni possibili.
Una volta raccolte tutte queste informazioni, è possibile trasferirle in modo automatico su un documento Word pronte per essere inviate per Email (Figura 76), oppure si può eseguire l’invio di queste informazioni via email in modo automatico direttamente da Qview 4 (Data Operations).

Figura 74: informazioni relative ai software e hardware installati.



Le informazioni sono suddivise in tre categorie.
- PC System Info: raccoglie tutte le informazioni Hardware e Software del Computer con il quale si stà lavorando.
- Qview System Info: raccoglie tutte le informazioni relative al programma Qview che si stà utilizzando.
- Project Information Info: raccoglie tutte le informazioni relative al progetto aperto in Qview.

0.8.11.8 About Qview

Viene visualizzata la versione di QVIEW e gli estremi per contattare la QEM srl.

0.9 Debug

Come è stato anticipato all’inizio di questo manuale, QVIEW è un indispensabile supporto per la programmazione in QCL e LADDER, sia per la stesura e la compilazione del codice, sia per il debugging del progetto realizzato.
Cos’è il debugging? Solitamente il tempo impiegato per la realizzazione di un progetto si suddivide in maniera equa tra il tempo per la programmazione e il tempo per la correzione degli errori. Il debugging è l’insieme di tutte le operazioni che permettono di rilevare questi errori che causano funzionamenti non desiderati. Gli strumenti che il QCL e LADDER mette a disposizione sono i seguenti:
- Esecuzione passo-passo del progetto (Step).
- Esecuzione passo-passo del singolo modulo (Step over).
- Inserimento di breakpoint.
- Watchpoint.

0.9.0.1 Esecuzione passo-passo

Una volta compilato e fatto il download, è possibile mettere in esecuzione l'applicativo scaricato selezionando il comando **Run** dal menu **Debug**; è anche possibile eseguire un'istruzione per volta, selezionando il comando **Step** dal menu **Debug**.
Ad ogni comando di **Step**, si può osservare lo scorrere di una istruzione; sull'editor viene evidenziata la riga di codice in esecuzione con una freccia sul bordo sinistro della finestra. Durante la scansione del codice, si può osservare l'effettivo flusso delle istruzioni ed il valore delle variabili.
Il comando **Step** esegue fedelmente la successione delle istruzioni QCL e LADDER e, la visualizzazione sull'editor, si sposta di modulo in modulo quando incontra un'istruzione di WAIT. Nel caso di tasks scritti in LADDER l'esecuzione passo-passo consiste nel eseguire il codice un rung per ogni passo.

0.9.0.2 Esecuzione passo-passo del singolo modulo

Il comando **Debug > Step Over** ha lo stesso effetto del comando **Step**; la successione delle istruzioni visualizzate si limita però a quelle dell'unità visualizzata sull'editor. E' un comando utile per la scansionare il codice contenuto in una sola delle unità componenti il progetto.

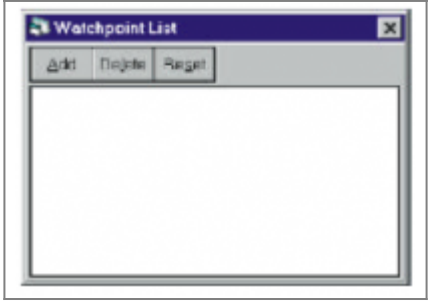
0.9.0.3 Inserimento di breakpoint

Il breakpoint è un punto preciso del programma in cui l'esecuzione del progetto si deve interrompere. Per fissare un breakpoint, posizionare con il cursore nel punto dell'editor in cui si desidera bloccare il programma e selezionare il comando **Debug > Toggle Breakpoint**; in corrispondenza dell'introduzione del breakpoint viene visualizzato il simbolo  . Con comando di RUN il programma verrà eseguito fino all'istruzione marcata dal breakpoint (esclusa) e quindi si bloccherà; la CPU assumerà lo stato di STOP. Nella finestra CPU si accenderà il LED Breakpoint per indicare l'intervento di un breakpoint. Sull'editor la riga interessata dal breakpoint viene indicata con il simbolo  . Con un nuovo RUN il programma verrà eseguito ancora una volta fino ad incontrare di nuovo il breakpoint. Questa funzionalità è molto utile per osservare lo stato delle variabili in un certo punto del codice che non sarebbe possibile apprezzare in fase di esecuzione normale. Inoltre il sistema del breakpoint può essere usato per vedere se una parte di codice viene eseguita oppure no. E' possibile posizionare un numero massimo di 7 breakpoint contemporaneamente lungo il codice. Nel caso ci siano più di un breakpoint per visualizzare velocemente il punto del codice su cui si è bloccato il programma si può usare l'icona  (Go To PC).

0.9.0.4 Watchpoint

Il watchpoint è un breakpoint condizionato dal valore di una variabile, di un ingresso o di un'uscita digitale, di un array, di un Data Group o di un parametro di un device.
La domanda a cui risponde il watchpoint è: in che punto del programma un parametro o una variabile fissata acquisisce il valore impostato?
Selezionando **Debug > WatchPoint**, appare la finestra ***Watchpoint List*** (Figura 1). Questa finestra contiene il parametro su cui impostare il watchpoint e il valore a cui deve "scattare".

Figura 1:elenco watchpoint.



Con il tasto ***Add*** è possibile scegliere nome e tipo di parametro da usare per il watchpoint.
Una volta introdotte queste informazioni, digitare il valore al quale il watchpoint si deve attivare per bloccare l'esecuzione del programma.

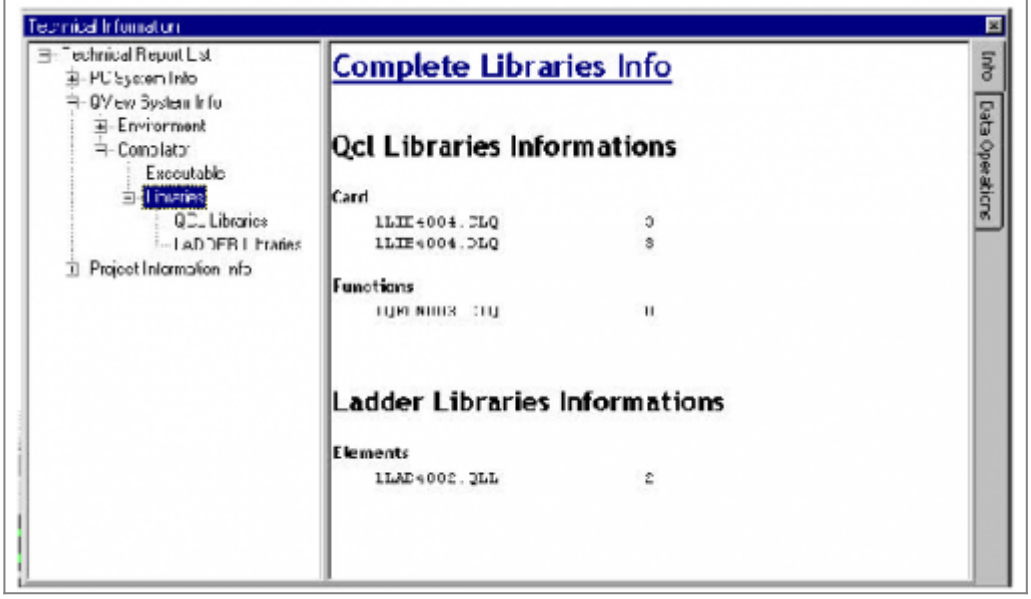
Esempio
Nella finestra ***Watchpoint List*** premere il tasto ***Add***. A questo punto appare una finestra ***Watchpoint Editor***; selezionare variables nella lista ***Watch Type***; nella lista ***Name*** indicare per esempio la variabile ***gfPariDisp***. Il programma si bloccherà quando la variabile assumerà questo valore; introdurre il valore 1. Dopo aver dato il RUN all'applicativo esso si bloccherà subito dopo l'assegnazione di *gfPariDisp* = 1

0.10 Le librerie QCL

Le librerie QCL sono una componente fondamentale per il compilatore QCL, mettendo a disposizione - in fase di compilazione - una serie di informazioni riguardanti i device e l'hardware. Tali informazioni sono necessarie per sapere quali parametri utilizza un device interno, le loro caratteristiche e la loro collocazione in memoria; quali comandi sono disponibili per quel determinato device e la sua sintassi di configurazione. Le librerie contengono le stesse informazioni anche per i device esterni gestiti da altre schede intelligenti che non siano la CPU; le librerie forniscono informazioni al compilatore per quanto riguarda la gestione delle schede non intelligenti.

Al momento della realizzazione di un nuovo progetto è quindi necessario verificare che la versione di QVIEW installata contenga le librerie necessarie per la gestione dei devices e dell'hardware da utilizzare (fare riferimento alle schede tecniche hardware). Selezionare la voce *Technical Info* dal menu *Help*; Appare la finestra di figura 1.

Figura 1: componenti software installati.



Alla riga LIBRARY viene specificato l'identificatore della libreria in uso (nell'esempio 1LIB3004). Le informazioni relative alla libreria da utilizzare per ogni tipo di scheda intelligente sono specificate, una volta per tutti i device gestiti dalla stessa scheda, nella documentazione firmware relativa, mentre per le schede non intelligenti bisogna far riferimento alla documentazione hardware.
Nel caso non si sia in possesso delle librerie richieste è necessario fare l'upgrade delle stesse selezionando Upgrade Library... dal menu Tools e seguire le istruzioni proposte nelle finestre che appariranno.

0.10.1 Accesso ai device (WAIT forzato)

Esiste una diversità tra l'esecuzione del codice scritto dall'utente e l'esecuzione delle funzionalità dei devices: il primo viene eseguito in modo sequenziale mentre i secondi vengono aggiornati ad intervalli di tempo regolari (a campionamento).
Nella normale esecuzione, il sistema operativo esegue questi due processi indipendentemente. Esiste però una condizione in cui i due processi si devono sincronizzare: una istruzione QCL di "lettura-parametro device" successiva ad una scrittura nel device stesso.
Per "scrittura nel device" si intende, oltre all'assegnazione di un valore ad uno dei suoi parametri, anche un qualsiasi comando verso il device. Quando si esegue una "scrittura nel device", tale scrittura viene eseguita all'istante di campionamento del device immediatamente successivo.
La lettura di un parametro di un device deve avvenire quando il device è perfettamente aggiornato. Quindi dal momento della scrittura non possono avvenire letture del device fino al successivo istante di campionamento; quindi, se dal codice QCL si richiede per esempio di leggere il valore di una variabile, questa richiesta viene rimandata al ciclo di task successivo eseguendo un cambio di task forzato: WAIT forzato. Questo per dare modo al device di essere aggiornato. Questo funzionamento assicura una lettura sempre aggiornata dei parametri device; inoltre evita che la CPU rimanga in attesa dell'aggiornamento su un task senza eseguire il codice degli altri task.
Esempio di codice che non funziona correttamente perchè non si è tenuto conto del WAIT forzato:
Unità 01:

```
MAIN:
  IF NOT ifStop            ;Se l'ingresso ifStop è 0.
    STOP AnAsse            ;Comando di STOP al device ANPOS AnAsse (scrittura nel device).
  ENDIF
  WAIT 1
  JUMP MAIN
END
```

Unità 02:

```
MAIN:
  IF AnAsse:st_still        ;Se l'asse è fermo AnAsse:st_still è 1 (lettura del device).
    ofLampada = 1           ;Setta l'uscita a 1.
  ENDIF
```

```
  WAIT 1
  JUMP MAIN
END
```

Il modulo 01 continua fare scritture nel device; è praticamente impossibile che alla lettura del parametro st_still il device sia aggiornato. Viene eseguito un cambio di task forzato ritornando al modulo 01 il quale ripete continuamente la scrittura impedendo l'aggiornamento del device. Per rendere funzionale il codice dell'esempio precedente si dovrà correggere nel modo seguente:

Unità 01:

```
MAIN:
  IF (NOT ifStop) AND (NOT AnAsse:st_still)     ;Se l'ingresso ifStop è 0 e l'asse non è fermo
    STOP AnAsse                                 ;Comando di STOP al device ANPOS AnAsse (scrittura nel device).
  ENDIF
  WAIT 1
  JUMP MAIN
END
```

Unità 02:

```
MAIN:
  IF AnAsse:st_still                            ;Se l'asse è fermo AnAsse:st_still è 1 (lettura del device).
    ofLampada = 1                               ;Setta l'uscita a 1.
  ENDIF
  WAIT 1
  JUMP MAIN
END
```

Una volta eseguito il comando STOP lo stato del device st_still va a 1 e quindi il comando non viene più ripetuto e il WAIT forzato non accade. In generale si deve evitare di inviare comandi ai device in modo continuo quando non è necessario.

0.11 Appendice A: Limitazioni QCL

0.11.0.1 Numero massimo di etichette

Durante la compilazione di alcune istruzioni QCL il compilatore genera internamente delle etichette che vengono utilizzate per successive elaborazioni. Esiste un limite di 999 etichette generabili per ogni unità, superato il quale non è possibile compilare il file sorgente. Per conoscere il numero di etichette interne generate bisogna sapere che le istruzioni IF, ELSE, CALL, SUB, o label (es. MAIN) generano una etichetta, mentre le istruzioni FOR e WHILE ne generano due. L'unica soluzione è quella di eliminare dal sorgente alcune delle istruzioni viste sopra e rientrare nei limiti ammessi.

0.11.0.2 Numero massimo di operandi annidati

Durante la compilazione di espressioni il massimo numero di operandi annidati è 6.
Ad esempio una istruzione di questo tipo è ammessa:
Variabile = 1+(1+(1+(1+(1+1))))
mentre questa provoca un errore:
Variabile = 1+(1+(1+(1+(1+(1+1)))))

0.11.0.3 Numero massimo di programmi

Il numero massimo di programmi e passi ammessi in un datagroup è di 65534. Il compilatore segnala errore se tale limite viene superato.

0.11.0.4 Dimensione massima di un Array

La dimensione massima di un array (sia ARRSYS che ARRGBL) è di 65535 elementi. Il compilatore segnala errore se tale limite viene superato.

0.11.0.5 Ciclo FOR

Il passo di incremento in un ciclo FOR deve essere numerico. Non è possibile usare variabili o espressioni.

0.11.0.6 Datagroup

Nella dichiarazione di un datagroup la sottosezione DATAPROGRAM è obbligatoria.

0.12 Appendice B: Conversione e promozione di tipo

0.12.1 Conversione di tipo

È importante ricordare che una conversione dal tipo intero (Flag, Byte Word o Long) al tipo Single non corrisponde ad un aumento di precisione ma cambia solamente il formato in cui il valore è rappresentato.

Per espressione si intende un insieme di operatori, costanti e variabili che, risolti, definiscono un valore numerico risultante. Il QCL mette a disposizione l'operatore di assegnamento "=" nella forma generale:

variabile = espressione
Non sono supportate le assegnazioni multiple tipo *variabile = variabile = espressione*
Per quanto riguarda l'istruzione di assegnamento, la regola di conversione è semplice: il QCL converte il valore alla destra del segno uguale nel tipo del dato sinistro.

Esempio:
Consideriamo le seguenti variabili:

```
SYSTEM
  sfFlag   F
  sbByte   B
  swWord   W
  slLong   L
  ssSingle S
```

sbByte = swWord
Viene eliminato il primo byte della variabile swWord assegnando a sbByte solo il byte meno significativo. Se il valore di swWord è compreso tra 127 e -128 i due valori risultano uguali e non viene eseguito alcun troncamento. Se il valore di swWord è esterno ai valori precedentemente considerati il valore di sbByte riflette solamente il valore del byte meno significativo di swWord.

ssSingle = slLong
Il valore di slLong viene convertito nel formato reale a singola precisione.

sfFlag = ssSingle
La variabile sfFlag viene assegnata al valore 1 se ssSingle rappresenta un valore diverso da zero.

0.12.2 Promozione di tipo

Quando in una espressione sono utilizzati dati di tipo differente, il compilatore QCL li converte tutti nello stesso tipo e, in particolare, nel tipo della dimensione che occupa più memoria, secondo quella che nei linguaggi viene definita *promozione di tipo*. Dopo che il compilatore ha applicato queste regole di conversione ogni coppia di operandi risulta della stessa dimensione che sarà anche la dimensione del risultato.

Ad esempio:

```
Variabile = (sbByte*sfFlag) + (swWord / sbByte) - (ssSingle+sfFlag)
```

Prima il compilatore converte sfFlag in BYTE e calcola il valore della moltiplicazione, poi il secondo sbByte in WORD e calcola il valore della divisione, poi sfFlag in SINGLE e calcola il valore della somma; il risultato di sbByte*sfFlag viene convertito in Word e ne calcola il valore. Quindi, questo risultato viene promosso a SINGLE per eseguire la sottrazione con il risultato di ssSingle+sfFlag.

Per quanto riguarda le costanti nelle espressioni, vengono convertite sempre nel tipo intero (FLAG, BYTE, WORD o LONG) della dimensione più indicata per contenere il valore (se nella costante non vi è il punto decimale). Se la costante contiene il punto decimale, viene convertita in tipo SINGLE.

Esempio:

```
  Variabile = swWord / sbByte
```

con swWord = 5 e sbByte = 2
Variabile risulta di tipo WORD con valore 2 perdendo così la parte decimale.

Se si riscrive l'espressione come:

```
  Variabile = (swWord * 1.0) / sbByte
```

con swWord = 5 e sbByte = 2
Variabile risulta di tipo SINGLE con valore 2,5. Questo perchè il prodotto della variabile swWord con una costante con parte decimale provoca la conversione del risultato in SINGLE.

0.13 Appendice C: Convenzioni di scrittura

Vengono proposte le convenzioni di scrittura del codice che sono state adottate dalla QEM. Queste convenzioni non sono regole sintattiche la cui inosservanza provoca degli errori in fase di compilazione; si possono considerare come stile di programmazione per evitare errori banali che causano inutili perdite di tempo.

I punti interessati da queste convenzioni sono:
- Nominare le costanti
- Nominare le variabili
- Nominare i device
- Indentazione nei cicli annidati
- Ordine e nome da dare alle unità

0.13.1 Nominare le costanti

Le costanti sono nomi ai quali è associato un ben definito valore numerico e sono dichiarate nel file di configurazione sotto la parole chiave CONST.
I nomi delle costanti sono composte da 2 campi separati da underscore (_) e le lettere che li compongono devono essere tutte maiuscole:

XXX_YYYYYYYY
XXX = Oggetto a cui fa riferimento (3 caratteri)
YYYYYYYY = Nome mnemonico (1÷8 caratteri)

Per la prima parte (XXX) possono essere utilizzate le seguenti parole:
TM = Timer.
KEY = Numero di tasto per pannellino.
PAG = Numero di pagina per pannellino.
MSG = Messaggio per pannellino.
DIM = Dimensione (per array o datagroup).
GEN = Costante di carattere generale.

Esempio
Una costante per contenere la dimensione di un array di quote può essere: DIM_QUOTE.

0.13.2 Nominare le variabili

Le variabili sono tutti i tipi di dato che possono cambiare il loro valore nel corso dell'esecuzione dell'applicativo (ad esclusione dei parametri Devices).
Il nome di una variabile è composto da 4 campi :

xyZZkkkkkkkk x = Gruppo di appartenenza della variabile (1 carattere o 2 per array) y = Dimensione della variabile (1 carattere) ZZ = Oggetto di riferimento (2 carattere) kkkkkkkk = Nome mnemonico (1÷8 caratteri)

0.13.3 Gruppo di appartenenza della variabile

Determina il tipo di variabile; i gruppi possibili sono:
s = System.
g = Global.
t = Timer.
d = Datagroup.
ds = Variabili statiche di datagroup.
dd = Variabili dinamiche di datagroup.
i = Input.
o = Output.
as = Array System.
ag = Array Global.

0.13.4 Dimensione della variabile

Determina il size della variabile; le dimensioni possibili sono:
f = Flag.
b = Byte.
w = Word.
l = Long.

s = Single.

0.13.5 Oggetto di riferimento

Area facoltativa; se utilizzata fa riferimento all'oggetto per cui viene usata. Gli oggetti standard sono:
PU = Pulsantiera , pulpito.
FT = Fotocellula.
FC = Fine Corsa.
AB = Abilitazioni.
EV = Elettrovalvole.
TL = Teleruttore.
CO = Comandi.
TR = Terminale. TM = Timer. ST = Stato. GE = Generale.

0.13.6 Nome mnemonico

Nome della variabile. Il nome deve essere scelto in modo che ricordi la funzionalità della variabile; se questa parte è composta da più nomi diversi, conviene che ogni iniziale di nome sia maiuscola.

Esempio
Esempio di una variabile di gruppo input, di tipo flag che rappresenta un pulsante di Start:

```
ifPUStart
```

Esempio di una variabile di gruppo global, di tipo byte che rappresenta un Comando di avanti manuale ad un device:

```
gbCOAvAsse
```

Esempio di una variabile di gruppo system, di tipo single, relativa ad un valore di velocità calcolato per l'avanzamento di un asse:

```
ssVelAvanAss               ;Il campo ZZ è stato omesso.
```

0.13.7 File di configurazione

Non esiste un preciso ordine per l'inserimento delle varie sezioni. Le varie sezioni dell'unità di esempio Firstapp sono state inserite automaticamente al momento dell'inserimento dell'unità con un determinato ordine, relativo a delle convenzioni interne QEM; nulla vieta che questo ordine possa essere cambiato.

0.13.8 Nominare i device

Anche i device interni ed esterni devono essere nominati. Si è adottata una convenzione del tipo:

XxYyyyyyyyyy
Xx = Tipo di device
yyyyyyyyy = Nome mnemonico device

Tipo device interni
Da = DAC
Ea = EANPOS
Oo = OOPOS3
Cn = COUNTER3
Ai = ANINP2
Cm = CAMMING3 e CAMMIN4
Rc = RECDATA

Alcuni tipi di device esterni
Md = MODBUS01
Pr = PROFI01

Esempio
un device interno DAC per fornire un riferimento analogico ad un motore di una pompa, si potrebbe chiamare:

```
DaMotorePom
```

0.13.9 Indentazione dei cicli annidati

E’ buona norma applicare l’indentazione dei cicli di iterazione (IF, FOR, WHILE). Per spiegare cos’è l’indentazione facciamo uso di un esempio.

Codice non indentato:

```
IF slProva % 2
gfPariDisp = 1             ;slProva è dispari.
ELSE
gfPariDisp = 0             ;slProva è pari.
ENDIF
```

Codice indentato:

```
IF slProva % 2
    gfPariDisp = 1         ;slProva è dispari.
ELSE
    gfPariDisp = 0         ;slProva è pari.
ENDIF
```

Quando si comincia un ciclo di iterazione, indentare significa spostare il codice al suo interno di un TAB in modo che sia chiaro, come si può osservare dall’esempio, quale è il codice interno al ciclo e quando il ciclo è concluso.
Nel caso di più cicli annidati si sposta il codice di un ulteriore TAB. Per esempio:

```
WHILE <condizione>
tab® IF <condizione>
tab® tab® . . .
          ; codice
          . . .
      ENDIF
ENDWHILE
```

0.13.10 Ordine e nome da assegnare ai moduli

Si utilizza una convenzione anche per i nomi da assegnare alle unità. Le unità possono essere chiamate con un nome qualsiasi ma viene adottata la convenzione di chiamarli

TASK_xx.MOD
xx è un numero che dà un’indicazione di massima della funzionalità del modulo.

xx = funzione.
00 = Gestione allarmi.
01 = Gestione ingressi e uscite digitali.
da 02 a 09 = Altri moduli.
10 e 11 = Gestione interfaccia utente-macchina.
da 12 in poi = Altri moduli.

E’ buona norma assegnare ad ogni unità una descrizione sintetica in modo da capire subito la loro funzione.

0.14 Appendice D: Parole chiave

In questo capitolo vengono riassunte le parole chiave del QCL.

ABS	valore assoluto
ACOS	arcocoseno
AND	AND logico
ANDB	And logico bit a bit
ARRGBL	sezione file di configurazione
ARRSYS	sezione file di configurazione
ASIN	arcoseno
ATAN	arcotangente
B	byte
BREAK	break
BUS	sezione file di configurazione
CALL	chiamata subroutine
CONST	sezione file di configurazione
COS	coseno
COT	cotangente
DATAGROUP	sezione file di configurazione
DATAPROGRAM	sezione file di configurazione
ELSE	“altrimenti” nell'istruzione IF
END	fine task
ENDIF	fine istruzione IF
ENDSUB	fine subroutine

ENDWHILE	fine while
EQ	uguale
EXP	esponenziale
EXTDEVICE	sezione file di configurazione
F	flag
FOR	istruzione FOR
FPROG	istruzione FPROG
FSTEP	istruzione FSTEP
GE	
GLOBAL	sezione file di configurazione
GT	maggiore
IF	istruzione IF
INPUT	sezione file di configurazione
INTDEVICE	sezione file di configurazione
JUMP	istruzione JUMP
LE	minore o uguale
LN	logaritmo naturale
LT	minore
NEG	negazione (inversione del segno o complemento a 2)
NEXT	istruzione NEXT
NOP	istruzione NOP
NOT	negazione (complemento a 1)
NOTB	negazione bit a bit (complemento a 1)
OR	OR logico
ORB	OR logico bit a bit
OUTPUT	sezione file di configurazione
POW	potenza
RESOUT	reset uscite
S	singola precisione
SETOUT	setta uscita
SIN	seno
SQRT	radice quadrata
STEP	sezione file di configurazione
SUB	subroutine
SYSTEM	sezione file di configurazione
T_RESTART	istruzione di restart
T_RESUME	istruzione di resume
T_SUSPEND	istruzione di sospensione
TAN	tangente
TIMER	sezione file di configurazione
W	word
WAIT	istruzione di wait
WHILE	istruzione di while
XORB	OR esclusivo bit a bit

F1	Contenuti (Contents)
F2	-
F3	Prossima ricerca (Find Next)
F4	Prossima unità (Next Unit)
F5	Run
F6	Stop
F7	Restart
F8	Step
F9	Toggle breakpoint
F11	Vai a PC (Go to PC)
F12	Prossima unità selezionata (Next selected unit)
SHIFT + F2	Functions Info
SHIFT + F4	Unità precedente (Previous Unit)
SHIFT + F5	Move Rows Up (Editor LADDER)
SHIFT + F6	Move Rows Down (Editor LADDER)
SHIFT + F8	Step Over
SHIFT + F9	Clear All
SHIFT + F12	Unità selezionata precedente (Previous selected unit)
CTRL + A	Redo
CTRL + C	Copia
CTRL + E	Proprietà dell'elemento LADDER (Element Properties...)

CTRL + F	Trova (Find)
CTRL + G	Vai a (Go to)
CTRL + K	Compila (Compile)
CTRL + L	Download
CTRL + N	Salva progetto con altro nome (Save project As...)
CTRL + P	Stampa (Print)
CTRL + R	Sostituisci (Replace)
CTRL + S	Salva progetto (Save project)
CTRL + T	Ladder Network Checking
CTRL + V	Incolla
CTRL + X	Taglia
CTRL + Z	Undo
CTRL + F1	Ladder Function Info
CTRL + F2	Functions Info
CTRL + F3	Risultati della compilazione (View compilation results)

0.14.1 File *.qm4: file di progetto

Viene creato da QVIEW dopo l'esecuzione del comando *New Project*, e non è gestibile dal programmatore. È un file unico per ogni applicazione nel quale sono elencati in successione il file di configurazione e i files sorgente.

0.14.2 File derivante dalla compilazione

0.14.2.1 File *.sym

E’ un file simboli utilizzato per la gestione dell'interfaccia operatore (terminale).

0.14.2.2 File *.bin

E’ un file risultato della compilazioneche può essere utilizzato per essere trasferito nella CPU senza utilizzare la comunicazione seriale (per esempio con Multi Media Card).

0.15 Appendice G: Compatibilità con le versioni precedenti

L’ambiente di sviluppo è pienamente compatibile con le versioni e release precedenti.
Nel Qview 4 è stata aggiunta la modalità di programmazione grafica LADDER che segue gli standard IEC1131. Il linguaggio di programmazione strutturato QCL rimane pienamente compatibile con le versioni e release precedenti del Qview. Il Qview 4.x non può essere utilizzato in alternativa al Qview 3.x dato che il Qview 4.x deve essere utilizzato per CPU di tipo “D”, mentre il Qview 3.x può essere utilizzato con CPU di tipo “B” e “C”.

0.15.0.1 Il file di progetto

Il progetto generato dal Qview 4, non è piú costituito da una collezione di files ASCII, disposti in una o piú directory, ma da un unico file di tipo binario. La scelta di racchiudere il progetto all'interno di un unico file binario è stata fatta per i seguenti motivi :
- Comodità di trasporto del progetto.
- Possibilità di nascondere a terzi la struttura interna del progetto (criptatura).
- Eliminazione della dipendenza del progetto dal percorso di salvataggio su disco.
- Futura possibilità di comprimere (comprimere) il progetto con conseguente riduzione dello spazio occupato sul supporto di salvataggio.
- Maggiore facilità di trasmettere il progetto a terzi (via internet come allegato).

Si è inoltre cambiata l'estensione del progetto generato, che è passata da “.QMV” a “.QM4” in modo da essere distinguibile dai files dai Qview di versione precedente.

0.15.0.2 Composizione del progetto

Essendo il progetto composto da un unico file, non si puó piú parlare di “files” di progetto, ma di units (o unità) che compongono il progetto. Il progetto potrà quindi essere composto da:
- Una Unit di configurazione.
- Una o piú units di codice QCL.
- Una o piú units di codice ladder.

- Una o piú units di documentazione.

Il massimo numero totale di units che è possibile introdurre in un progetto è di 65535.

NOTE I linguaggio QCL4 è compatibile con le versioni QCL3, QCL2 e QCL1. Nel QCL3 e QCL4 si nota una sostanziale miglioria nella dichiarazione e gestione dei DATAGROUP rispetto ai QCL2 e QCL1. In ogni caso i compilatori QCL4 e QCL3 interpretano correttamente entrambe le dichiarazioni e i modi di utilizzo di questa struttura dati.

0.15.1 Dichiarazione dei Datagroup

Nelle versioni QCL1 e QCL2 era obbligatorio definire almeno un task proprietario ed eventualmente i task associati; l'accesso alle variabili del datagroup era permesso solamente a tali tasks. La seguente è la sintassi per la definizione dei datagroup nelle versioni QCL2 e QCL3:

```
;---------------------------------------------------------------------
; Dichiarazione del DataGroup nelle versioni QCL1 e QCL2
;---------------------------------------------------------------------
DATAGROUP
  <nome DataGroup>   <task_proprietario> [task_associato]
  [task_associato]   [task_associato]

;-- Definisce il numero programmi ----------------------------------
  DATAPROGRAM
    <numero programmi>

;-- Definizione variabili statiche ---------------------------------
    <nome variabile>   <tipo>
    <nome variabile>   <tipo>
    <nome variabile>   <tipo>

;-- Definisce il numero di passi -----------------------------------
  STEP
    <numero passi>

;-- Definisce le variabili di ciascun passo ------------------------
    <nome variabile>   <tipo>
    <nome variabile>   <tipo>
    <nome variabile>   <tipo>
```

0.15.2 Utilizzo dei DATAGROUP

Le istruzioni FPROG e FSTEP del QCL1 e QCL2 sono state soppiantate dall'uso di variabili che contengono il valore del programma e del passo in elaborazione.

```
;---------------------------------------------------------------------
; Utilizzo del DataGroup nelle versioni QCL1 e QCL2
;---------------------------------------------------------------------

FPROG sbProg
FSTEP sbStep
StaticVar = 10
IndexVarVar = 100
```

0.16 Appendice H: Segnalazioni di errore

Per semplificare l'utilizzo di Qview, è stato introdotta l'interfaccia message (Figura 1) con la quale il programma comunica all'utente il presentarsi di particolari eventi funzionali.

0.16.0.1 Evento informativo

Serve a comunicare o richiedere all'utente determinate informazioni prima di continuare con l'operazione in esecuzione.

0.16.0.2 Evento di pericolo

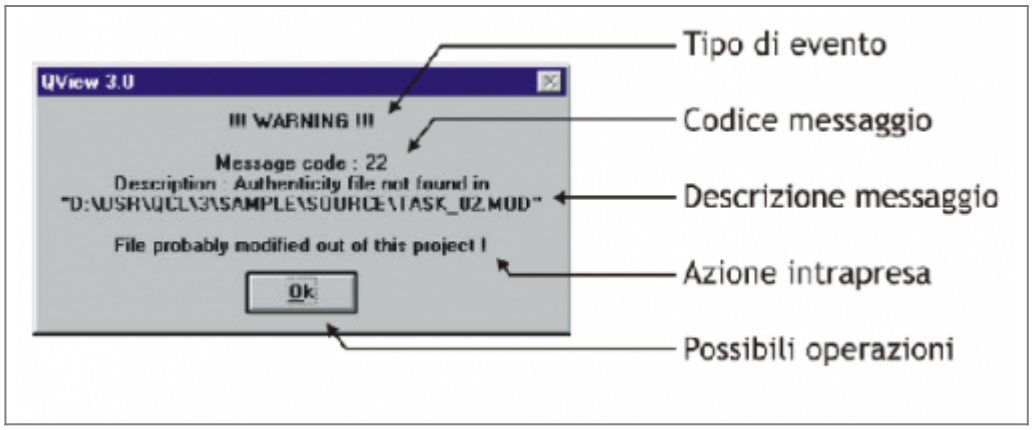
Indica all'utente che il programma ha riscontrato un'anomalia e fornisce tutte le informazioni utili alla comprensione della stessa.

0.16.0.3 Errore fatale

Indica all'utente che il programma ha riscontrato un grave errore funzionale il quale non permette un corretto funzionamento del programma e ne precede quindi la conclusione forzata.

0.16.1 Interfaccia Message

Figura 1



0.16.1.1 Tipo evento (TEV).

Descrive il tipo di evento intervenuto. Può valere:
!!! INFORMATION !!! per un evento informativo.
!!! WARNING !!! per un evento di pericolo.
!!! FATAL ERROR !!! per un evento di errore fatale.

0.16.1.2 Codice messaggio (CME)

Contiene il numero identificativo del messaggio intervenuto. Tramite questo numero è possibile accedere alla documentazione approfondita dell'evento presente in questa appendice.

0.16.1.3 Descrizione messaggio (DME)

Descrive in modo sintetico l'evento intervenuto.

0.16.1.4 Azione intrapresa (AZI)

Viene descritta l'azione intrapresa di risposta all'evento intervenuto. Può contenere anche una estensione alla descrizione messaggio.

0.16.1.5 Operazioni possibili (OPE)

Sono disponibili uno o più bottoni per le scelte dell'utente.

0.16.1.6 01] Errore durante la lettura del file di configurazione.

TEV: !!! FATAL ERROR !!!
CME: Message code : 1
DME: Description : Error in reading configuration file
AZI: The application will be closed !
OPE: Ok, per confermare ed uscire da Qview.
Questo evento puó presentarsi solo all'avvio del programma Qview e nel caso di:
- Errore nel controllo del percorso del file di configurazione.
- Errore nel controllo dell'esistenza del file di configurazione.
- Errore nell'apertura del file di configurazione.
- Errore nella lettura del file di configurazione.
- Errore nella chiusura del file di configurazione.

Nel caso l'evento continui a presentarsi, impedendo l'uso di Qview, uscire da Windows e riavviare il computer ed eventualmente al persistere dell'errore reinstallare il programma dopo aver opportunamente rimosso la precedente versione.

0.16.1.7 02] Errore durante la scrittura del file di configurazione.

TEV: !!! FATAL ERROR !!!
CME: Message code : 2

DME: Description : Error in writing configuration file
AZI: The application will be closed !
OPE: Ok, per confermare ed uscire da Qview.
Questo evento può presentarsi solo alla chiusura del programma Qview e nel caso di:
- Errore nel controllo del percorso del file di configurazione.
- Errore nell'apertura del file di configurazione.
- Errore nella scrittura del file di configurazione.
- Errore nella chiusura del file di configurazione.

0.16.1.8 03] File di configurazione desktop inesistente o non trovato.

TEV: !!! INFORMATION !!!
CME: Message code : 3
DME: Description : Configuration file not found
AZI: Default values will be assumed !
OPE: Ok, per confermare l'operazione e far assumere i valori di default al desktop.
Questo evento può presentarsi solo all'avvio del programma Qview e nel caso in cui il file di configurazione non venga trovato.
I valori assunti per default sono:
- Dimensione Font = 9.
- Numero caratteri Tab = 4.
- Colore testo Editor = NERO.
- Colore sfondo Editor = BIANCO.
- Colore simbolo PC Counter = VERDE SCURO.
- Colore simbolo Breakpoint = ROSSO CHIARO.
- Colore bordo sinistro Editor = GRIGIO CHIARO.
- Porta di comunicazione seriale = COM1.
- Velocità di comunicazione seriale = 19200 BAUD.

0.16.1.9 04] Disco non pronto.

TEV: !!! WARNING !!!
CME: Message code : 4
DME: Description : Drive "nome_disco:" not ready
OPE: Retry, per ritentare l'accesso al disco.
OPE: Cancel, per annullare l'operazione indicando l'indisponibilità del disco.
Questo evento si presenta ogni qual volta Qview tenta di accedere ad un disco di sistema che non è pronto come ad esempio un floppy drive senza dischetto inserito (es: lettura di un progetto da floppy).

0.16.1.10 05] Errore di sincronismo software.

TEV: !!! FATAL ERROR !!!
CME: Message code : 5
DME: Description : Software synchronous error
AZI: The application will be closed !
OPE: Ok, per confermare ed uscire da Qview.
Questo evento può presentarsi in caso di errori nei sincronismi interni di esecuzione del programma.
Nel caso l'evento continui a presentarsi, impedendo l'uso di Qview, uscire da Windows e riavviare il computer ed eventualmente al persistere dell'errore reinstallare il programma dopo aver opportunamente rimosso la precedente versione.

0.16.1.11 06] Errore durante la creazione di un oggetto.

TEV: !!! FATAL ERROR !!!
CME: Message code : 6
DME: Description : Error in object closing
AZI: The application will be closed !
OPE: Ok, per confermare ed uscire da Qview.
Questo evento può presentarsi in caso di errori nella creazione degli oggetti interni al programma.
Nel caso l'evento continui a presentarsi, impedendo l'uso di Qview, uscire da Windows e riavviare il computer ed eventualmente al persistere dell'errore reinstallare il programma dopo aver opportunamente rimosso la precedente versione.

0.16.1.12 07] Errore durante la chiusura di un oggetto.

TEV: !!! FATAL ERROR !!!

CME: Message code : 7
DME: Description : Error in object creation
AZI: The application will be closed !
OPE: Ok, per confermare ed uscire da Qview.
Questo evento può presentarsi in caso di errori nella creazione degli oggetti interni al programma.
Nel caso l'evento continui a presentarsi, impedendo l'uso di Qview, uscire da Windows e riavviare il computer ed eventualmente al persistere dell'errore reinstallare il programma dopo aver opportunamente rimosso la precedente versione.

0.16.1.13 08] Argomenti linea di comando non validi

TEV: !!! Warning !!!
CME: Message code : 8
DME: Description : Arguments not valid
AZI: Command line not executed !
OPE: Ok per confermare e proseguire con Qview
L'avvertimento si presenta quando si è fatto partire Qview con un comando che non riconosce.
Controllare il tipo di comando specificato nella linea di comando ed eventualmente correggerlo.

0.16.1.14 09] Errore durante il recupero dati

TEV: !!! Warning !!!
CME: Message code : 9
DME: Description : Data recover error
AZI: Entered data will be lost !
OPE: Ok per confermare e proseguire con Qview
Il problema si manifesta durante l'utilizzo delle funzionalità di Watch. Provare a chiudere Qview e riaprirlo. Se il problema dovesse ancora manifestarsi, contattare l'assistenza tecnica.

0.16.1.15 10] Errore durante il recupero dei simboli

TEV: !!! Warning !!!
CME: Message code : 10
DME: Description : Symbols recover error
AZI: The application will be closed !
OPE: Ok, per confermare ed uscire da Qview.
Il messaggio può comparire durante le operazioni con la I/O list, specialmente nella fase di aggiunta o cancellazione di variabili dalla lista. Riaprire il progetto, ricompilarlo e trasferirlo in CPU. Se il problema dovesse ripresentarsi, chiamare il servizio assistenza.

0.16.1.16 11] Dati errati o nessun Watchpoint disponibile

TEV: !!! Warning !!!
CME: Message code: 11
DME: Description: Invalid data or no more Watchpoint avaiable for this data type AZI: The Watchpoint will not be set !
OPE: OK per confermare e proseguire con Qview.

Il messaggio si manifesta durante le operazioni di impostazione del Watchpoint. Ricompilare e ritrasferire il progetto. Se il problema si manifesta nuovamente, chiamare il servizio assistenza.

0.16.1.17 12] Errore nella procedura Save Data

TEV: !!! Warning !!!
CME: Message code : 12
DME: Description : Error in Save Data procedure
AZI: Save Data failed !
OPE: Nessuna operazione
Messaggio attualmente non supportato.

0.16.1.18 13] Errore nella procedura Recall Data

TEV: !!! Warning !!!
CME: Message code : 13

DME: Description : Error in Recall Data procedure
AZI: Recall Data failed !
OPE: Nessuna operazione
Messaggio attualmente non supportato.

0.16.1.19 14] Dati introdotti non validi

TEV: !!! INFORMATION !!!
CME: Message code : 14
DME: Description : Invalid data in one or more fields
AZI: Check and retype wrong fields !
OPE: Ok per eliminare il messaggio e passare di nuovo alla finestra File Property.
Il messaggio compare quando si lasciano vuoti uno o più campi della finestra File Property.
Se il messaggio continua ad apparire, controllare che non ci siano campi vuoti di introduzione dati.

0.16.1.20 15] Dati introdotti insufficienti

TEV: !!! INFORMATION !!!
CME: Message code : 15
DME: Description : Missing data in one or more fields
AZI: Check and type missing fields !
OPE: OK per confermare e proseguire con Qview.
Il messaggio si manifesta nell'introduzione dati durante le operazioni di Watchpoint. Controllare l'esattezza dei dati introdotti e/o reintrodurre i dati.

0.16.1.21 16] Errore durante le operazioni sui file temporanei

TEV: !!! FATAL ERROR !!!
CME: Message code : 16
DME: Description : Error in temporary file operation
AZI: The application will be closed !\ OPE: Ok, per confermare ed uscire da Qview.
Questo errore si manifesta quando Qview non riesce a creare, ad espletare correttamente tutte le funzioni di gestione delle directory temporanee. Nel caso l'errore continui a presentarsi, accertarsi che il disco fisso non sia pieno oppure che altri programmi non facciano riferimento a qualche file contenuto nelle directory temporanee create da Qview.

0.16.1.22 17] Errore durante la creazione del file

TEV: !!! WARNING !!!
CME: Message code : 17
DME: Description : Error in creation file
AZI: File will not be created !
OPE: OK per confermare e proseguire con Qview.
Questo errore si manifesta quando Qview non riesce ad operare correttamente con i files di progetto. Verificare se il disco rigido non può più contenere files. Provare a riavviare Qview e se l'errore persiste, provare a riavviare Windows.

0.16.1.23 18] Errore durante l'aggiunta del file di progetto

TEV: !!! WARNING !!!
CME: Message code : 18
DME: Description : Error in creation file
AZI: File will not be added !
OPE: OK per confermare e proseguire con Qview.
Questo errore si manifesta quando Qview non riesce ad aggiungere un file di progetto. Controllare se il file o i files di progetto in questione sono presenti nella directory di progetto oppure se sono danneggiati.

0.16.1.24 19] Errore durante la lettura file

TEV: !!! WARNING !!!
CME: Message code : 18
DME: Description : Error in reading file
AZI: File will not be read !

OPE: OK per confermare e proseguire con Qview.
Questo errore si manifesta quando Qview non riesce a leggere le informazioni contenute in un file di progetto. Controllare se il file o i files di progetto in questione sono danneggiati.
Provare a chiudere e riaprire Qview.

0.16.1.25 20] Errore durante la lettura del prossimo file

TEV: !!! WARNING !!!
CME: Message code : 20
DME: Description : Error in reading next file
AZI: File will not be read !
OPE: OK per confermare e proseguire con Qview.
L'errore si manifesta quando Qview non riesce a leggere le informazioni contenute in un file di progetto che è in procinto di aprire con il comando Next File. Controllare se il file o i files di progetto in questione sono danneggiati o se sono mancanti.
Provare a ricaricare il progetto.

0.16.1.26 21] Errore durante la scrittura del file

TEV: !!! WARNING !!!
CME: Message code : 21
DME: Description : Error in writing file
AZI: File will not be written !
OPE: OK per confermare e proseguire con Qview.
L'errore si manifesta quando ci sono problemi ai file della directory temporanea oppure il progetto non è stato caricato in modo corretto. Ricaricare il progetto e se il problema persiste, chiudere e riaprire Qview.

0.16.1.27 22] Errore di autenticità del file

TEV: !!! WARNING !!!
CME: Message code : 22
DME: Description : Authenticity file not found in
AZI: File probably modified out of this project !
OPE: OK per confermare e proseguire con Qview.
Il messaggio si manifesta durante l'apertura del progetto. Controllare se il progetto è stato modificato con un'editor di testo senza l'ausilio di Qview.

0.16.1.28 23] Trovati caratteri non validi nel file

TEV: !!! WARNING !!!
CME: Message code : 23
DME: Description : Not valid chars found in file
AZI: Not valid chars replaced with '?' !
OPE: OK per confermare e proseguire con Qview.
Il messaggio si manifesta durante l'apertura del progetto oppure passando ad altro task all'interno di Qview. Il Controllare l'integrità dei file di progetto. Se si ha una copia di backup, sovrascrivere il progetto.

0.16.1.29 24] Errore durante l'apertura della porta di comunicazione seriale

TEV: !!! WARNING !!!
CME: Message code : 24
DME: Description : COM port open error
AZI: The communication will not be opened !
OPE: OK per confermare e proseguire con Qview.
Il messaggio si manifesta dopo aver tentato di aprire la comunicazione con il Qmove. Controllare che il cavo di comunicazione seriale sia collegato al PC e/o al Qmove. Qualora il problema persistesse, controllare se il cavo di comunicazione è correttamente costruito o se le porte seriali del PC e/o del Qmove siano perfettamente funzionarti.

0.16.1.30 25] Errore durante la chiusura della porta di comunicazione seriale

TEV: !!! WARNING !!!
CME: Message code : 25

DME: Description : COM port close error
AZI: The communication will not be closed !
OPE: OK per confermare e proseguire con Qview.
Il messaggio si manifesta durante la chiusura della porta di comunicazione seriale.

0.16.1.31 26] File di servizio del progetto dichiarati ma non disponibili

TEV: !!! FATAL ERROR !!!
CME: Message code : 26
DME: Description : Service project files declared but unavailable
AZI: The application will be closed !
OPE: Ok per confermare ed uscire da Qview.
Il messaggio può comparire durante la fase di apertura di un progetto. Il file .SYS puó essere danneggiato. Provare a riaprire il progetto. Se il messaggio compare ancora, ricostruire il progetto manualmente partendo dai files sorgente a disposizione.

0.16.1.32 27] File di servizio del progetto non trovati

TEV: !!! INFORMATION !!!
CME: Message code : 27
DME: Description : Service project files not found
AZI: Compile project action necessary !
OPE: OK per confermare e proseguire con Qview.
Il messaggio si manifesta durante l'apertura del progetto. Eseguire una nuova compilazione e salvataggio del progetto.

0.16.1.33 28] File di servizio del progetto non trovati

TEV: !!! FATAL ERROR !!!
CME: Message code : 28
DME: Description : Error in saving service project files .. RUN, SYM, DBG files will not be correctly saved
AZI: The application will be closed !
OPE: OK per confermare ed uscire da Qview.
Il messaggio si manifesta durante l'apertura, la chiusura o il salvataggio di un progetto. Controllare che nella directory del progetto siano presenti i file con estensione specificata nel messaggio. Riaprire il progetto, compilarlo e salvarlo. Se il messaggio persiste, costruire un nuovo progetto con gli stessi task del precedente.

0.16.1.34 29] Errore checksum dei files di

TEV: !!! WARNING !!!
CME: Message code : 29
DME: Description : Checksum error in service project files
AZI: Compile project action necessary !
OPE: OK per confermare e proseguire con Qview.
Il messaggio si manifesta durante l'apertura del progetto qualora qualche file di servizio abbia subito dei cambiamenti al di fuori dell'ambiente Qview. E' necessario ricompilare e salvare il progetto.

0.16.1.35 30] Compilatore non trovato

TEV: !!! WARNING !!!
CME: Message code : 30
DME: Description : QCL Compiler not found
AZI: Compilation is not possible !
OPE: OK per confermare e proseguire con Qview.
Il messaggio si manifesta quando si tenta di compilare il progetto e il compilatore QCLC30.EXE non è presente nella directory Qview32/Bin. Reinstallare QView32.

0.16.1.36 31] Errore durante le fasi di compilazione

TEV: !!! WARNING !!!
CME: Message code : 31
DME: Description : Error during compilation
AZI: Compilation is not possible ! ... Check disk write protection !

OPE: OK per confermare e proseguire con Qview.
Il messaggio si manifesta quando si tenta di compilare il progetto. Salvare, riaprire il progetto e ricompilare.

0.16.1.37 32] Errore durante l'apertura del progetto

TEV: !!! WARNING !!!
CME: Message code : 32
DME: Description : Error in opening project
AZI: The project will not be opened !
OPE: OK per confermare e proseguire con Qview.
Il messaggio si manifesta quando si tenta di aprire il progetto. Le cause che producono l'errore sono svariate e legate alle informazioni contenute nel progetto. Controllare l'esistenza della directory del progetto o se quest'ultimo è stato spostato.

0.16.1.38 33] Errore durante la scrittura del progetto

TEV: !!! WARNING !!!
CME: Message code : 33
DME: Description : Error in writing project
AZI: Then project will not be written !
OPE: OK per confermare e proseguire con Qview.
Il messaggio può comparire durante le fasi di salvataggio di un nuovo progetto. Controllare se il disco fisso è troppo pieno. Provare a riavviare Qview. Se il messaggio continua ad apparire, contattare il servizio assistenza tecnica.

0.16.1.39 34] Progetto realizzato con QView di versioni precedenti

TEV: !!! INFORMATION !!!
CME: Message code : 34
DME: Description : This project is coded with another QView version.
AZI: Only project's file will be loaded !
OPE: OK per confermare e proseguire con Qview.
Il messaggio si verifica quando viene caricato un progetto realizzato con una precedente versione di Qview. Lo scopo del messaggio e' solo quello di avvertire circa la natura del progetto aperto. Il progetto necessita di essere ricompilato e salvato.

0.16.1.40 35] Progetto realizzato con vecchia release di Qview

TEV: !!! INFORMATION !!!
CME: Message code : 35
DME: Description : This project refers to an older Qview version or release
AZI: Only project's file will be loaded !
OPE: OK per confermare e proseguire con Qview.
Il messaggio si verifica quando viene caricato un progetto realizzato con una vecchia versione di Qview. Il Per rendere pienamente compatibile il progetto con Qview32, eseguire una compilazione e salvare subito dopo.

0.16.1.41 36] Versione progetto incompatibile

TEV: !!! WARNING !!!
CME: Message code : 36
DME: Description : Incompatible Qview version
AZI: The project will not be opened !
OPE: OK per confermare e proseguire con Qview.
Il messaggio compare quando si tenta di caricare un progetto realizzato con versioni successive di Qview. Riaprire il progetto con una versione più recente di Qview.

0.16.1.42 37] Progetto corrotto

TEV: !!! WARNING !!!
CME: Message code : 37
DME: Description : File project corrupted
AZI: The project will not be opened !
OPE: OK per confermare e proseguire con Qview.
Il messaggio compare quando si tenta di caricare un progetto che non è stato precedentemente salvato correttamente o

inopportunamente modificato manualmente. Reperire il backup del progetto e sostituirlo al posto di quello danneggiato.

0.16.1.43 38] Errore durante l’ispezione del progetto

TEV: !!! FATAL ERROR !!!
CME: Message code : 38
DME: Description : Error during project inspection
AZI: The application will be closed !
OPE: Ok per confermare ed uscire Qview.
Il messaggio compare se Qview incontra problemi nel reperimento delle informazioni dei dati di progetto durante le operazioni di chiusura dello stesso. Riaprire Qview con lo stesso progetto e vedere se i l problema si manifesta. Se il problema si manifestasse ancora, provare ad aprire un progetto diverso e verificare che il progetto precedente non sia stato danneggiato.

0.16.1.44 39] Salva le modifiche di

TEV: Non specificato
CME: Message code : 39
DME: Description : Save changes to
AZI: Non specificato
OPE: OK per confermare e proseguire con Qview.
OPE: NO per non eseguire l’azione e proseguire con Qview
OPE: ANNULLA per annullare le operazioni.
Il messaggio compare quando si esce da Qview o si apre un nuovo progetto con il vecchio progetto ancora caricato contenenti parti modificate da salvare.

0.16.1.45 40] Testo in ricerca non trovato

TEV: !!! INFORMATION !!!
CME: Message code : 40
DME: Description : The searched for text not been found !
AZI: Non specificato
OPE: Ok per confermare e proseguire con Qview
Il messaggio compare quando si sta eseguendo un’operazione di ricerca di testo. Con questo messaggio Qview indica che la parola cercata nell’area specificata, non esiste.

0.16.1.46 41] Occorrenze testo rimpiazzate

TEV: !!! INFORMATION !!!
CME: Message code : 41
DME: Description : “ ..... occurrences replaced ! “
AZI: Non specificato
OPE: Ok per confermare e proseguire con Qview
Il messaggio compare alla fine dell’operazione di replace all e sta ad indicare quante parole sono state trovate e rimpiazzate.

0.16.1.47 42] Errore durante il recupero dei dati dalla CPU

TEV: !!! WARNING !!!
CME: Message code : 42
DME: Description : CPU data recover error
AZI: Required data will not be recovered !
OPE: Ok per confermare e proseguire con Qview
Il messaggio può apparire qualora si scegliesse di visualizzare le informazioni del bus o l’ID di progetto. Verificare il funzionamento della comunicazione seriale. Se il problema continua a manifestarsi, provare a sostituire la CPU. Contattare il servizio di assistenza tecnica.

0.16.1.48 43] Reset CPU

TEV: !!! WARNING !!!
CME: Message code : 43
DME: Description : After this operation all data and program in CPU will be irrimediably lost!
AZI: Execute Reset ?

OPE: Ok per confermare e proseguire con Qview
Il messaggio compare quando si esegue un'operazione di reset sulla CPU tramite il comando Reset.

0.16.1.49 44] Operazione di Reset CPU fallita

TEV: !!! WARNING !!!
CME: Message code : 44
DME: Description : CPU Reset aborted
AZI: Required operation not complete !
OPE: Ok per confermare e proseguire con Qview
Il messaggio compare quando si è tentato di eseguire un Reset CPU, ma l'operazione non è stata portata a termine con successo. Controllare lo stato della comunicazione seriale. Se il problema persiste, provare a spegnere e riaccendere il sistema Qmove.

0.16.1.50 45] Backup Data

TEV: !!! INFORMATION !!!
CME: Message code : 45
DME: Description : After this operation all data in CPU will be saved in storage memory !
AZI: Execute Backup Data ?
OPE: Ok per confermare e proseguire con Qview
Il messaggio compare quando si esegue il backup dei dati contenuti nella CPU del Qmove.

0.16.1.51 46] Operazione di Backup Data fallita

TEV: !!! WARNING !!!
CME: Message code : 46
DME: Description : Backup Data aborted
AZI: Required operation not complete !
OPE: Ok per confermare e proseguire con Qview Il messaggio compare quando si è tentato di eseguire il backup dei dati contenuti nella CPU ma l'operazione non è andata a buon fine. Ritentare l'operazione e/o controllare lo stato della comunicazione seriale. Se il messaggio continua ad apparire, contattare il servizio assistenza.

0.16.1.52 47] Restore Data

TEV: !!! INFORMATION !!!
CME: Message code : 47
DME: Description : After this operation all data in CPU will be updated by data from storage memory !
AZI: Execute Restore Data ?
OPE: Ok per confermare proseguire con Qview
Il messaggio compare quando si esegue un comando di restore data

0.16.1.53 48] Operazione di Restore Data fallita

TEV: !!! WARNING !!!
CME: Message code : 48
DME: Description : Restore Data aborted
AZI: Required operation not complete !
OPE: Ok per confermare e proseguire con Qview
Il messaggio compare quando si esegue un comando di restore data e l'operazione non ha avuto successo. Ritentare l'operazione e/o controllare lo stato della comunicazione seriale.

0.16.1.54 49] Operazione di Save Data fallita

TEV: !!! WARNING !!!
CME: Message code : 49
DME: Description : Save Data aborted
AZI: Required operation not complete !
OPE: Ok per confermare e proseguire con Qview
Il messaggio compare quando si esegue un comando di save data e l'operazione non ha successo. Controllare se ci sono problemi al disco fisso del proprio computer e/o verificare lo stato della comunicazione seriale.

0.16.1.55 50] Operazioni di Recall Data fallita

TEV: !!! WARNING !!!
CME: Message code : 50
DME: Description : Recall Data aborted
AZI: Required operation not complete !
OPE: Ok per confermare e proseguire con Qview
Il messaggio compare quando si esegue un comando di recall data e l'operazione non ha successo. Verificare se ci sono problemi al disco fisso del computer e/o alla comunicazione seriale con la CPU.

0.16.1.56 51] Errore durante la lettura dei Breakpoint

TEV: !!! WARNING !!!
CME: Message code : 51
DME: Description : Recover Breakpoint error
AZI: Required operation not complete !
OPE: Ok per confermare e proseguire con Qview
Il messaggio può comparire durante la fase di impostazione o cancellazione di breakpoint. Ricompilare e riscaricare il progetto. Controllare lo stato della comunicazione seriale.

0.16.1.57 52] Errore durante l'impostazione del Breakpoint

TEV: !!! WARNING !!!
CME: Message code : 52
DME: Description : No Breakpoint allowed here
AZI: Required operation not complete !
OPE: Ok per confermare e proseguire con Qview
Il messaggio compare quando si tenta d'impostare un breakpoint in un punto del codice dove non ha senso impostarlo. Spostare l'introduzione del break point in un punto consentito.

0.16.1.58 53] Errore durante la rimozione del Breakpoint

TEV: !!! WARNING !!!
CME: Message code : 53
DME: Description : Clear Breakpoint error
AZI: Required operation not complete !
OPE: Ok per confermare e proseguire con Qview
Il messaggio può comparire quando si tenta di eliminare un breakpoint precedente impostato. Ricompilare e scaricare nuovamente il progetto. Se il problema continua a manifestarsi, verificare la comunicazione seriale e/o provare a sostituire la CPU.

0.16.1.59 54] Non sono disponibili altri Breakpoint

TEV: !!! WARNING !!!
CME: Message code : 54
DME: Description : No more Breakpoints available
AZI: Required operation not complete !
OPE: Ok per confermare e proseguire con Qview
Il messaggio compare quando si tenta d'impostare più di cinque breakpoint consentiti.

0.16.1.60 55] Errore in operazioni di SubClass

TEV: !!! FATAL ERROR !!!
CME: Message code : 55
DME: Description : Software SubClass error
AZI: The application will be closed !
OPE: Ok per uscire da Qview
Provare a ripartire con il Qview e se l'errore si manifesta ancora, uscire da Windows e ripartire.

0.16.1.61 56] Errore durante la procedura di stampa

TEV: !!! WARNING !!!
CME: Message code : 56
DME: Description : Print error
AZI: Required operation not complete !
OPE: Ok per confermare e proseguire con Qview
Controllare che la stampante sia correttamente collegata, installata ed alimentata. Vedere il manuale della propria stampante.

0.16.1.62 57] Operazione di Convert Data fallita

TEV: !!! WARNING !!!
CME: Message code : 57
DME: Description : Convert Data aborted
AZI: Required operation not complete !
OPE: Ok per confermare e proseguire con Qview
Il messaggio potrebbe comparire mentre si tenta di eseguire un'operazione di Convert Data. Controllare la presenza su disco fisso del file .DAT. Provare ad uscire e riavviare Qview.

0.16.1.63 58] Errore durante la lettura del file precedente

TEV: !!! WARNING !!!
CME: Message code : 58
DME: Description : Error in reading previous file
AZI: File will not be read !
OPE: Ok per confermare e proseguire con Qview
Il messaggio compare dopo aver tentato di ritornare al file precedente tramite il comando di Previus file. Provare a chiudere e riaprire Qview.

0.16.1.64 59] Errore durante l'aggiornamento della libreria

TEV: !!! WARNING !!!
CME: Message code : 59
DME: Description : Error in upgrading library
AZI: Required operation not complete !
OPE: Ok per confermare e proseguire con Qview
Il messaggio compare se ci sono problemi nelle operazioni di aggiornamento delle librerie. Accertarsi di aver inserito il floppy disk di aggiornamento librerie nel drive A. Accertarsi che il floppy sia effettivamente quello di aggiornamento delle librerie. Verificare che il floppy introdotto non contenga files di aggiornamento danneggiati.

0.16.1.65 60] Disco di aggiornamento della libreria incompatibile

TEV: !!! WARNING !!!
CME: Message code : 60
DME: Description : Incompatible Upgrade Library disk
AZI: Required operation not complete !
OPE: Ok per confermare e proseguire con Qview.
Messaggio non attualmente utilizzato

0.16.1.66 61] Conferma aggiornamento libreria

TEV: !!! INFORMATION !!!
CME: Message code : 61
DME: Description : Libraries updating ! Old libraries are :..... New libraries are :
AZI: Are you sure to upgrade libraries ?
OPE: Ok per confermare l'operazione e proseguire.
OPE: Cancel per annullare l'operazione e proseguire con Qview.
Il messaggio chiede di confermare se si desidera proseguire nelle operazioni di upgrade delle librerie.

0.16.1.67 62] Riavvio del programma Qview

TEV: !!! INFORMATION !!!
CME: Message code : 62

DME: Description : Restart all Qview active programs to complete library upgrading !
AZI: Non specificato
OPE: Non specificato
Messaggio attualmente non utilizzato specificato.

0.16.1.68 63] Errore informazioni registry

TEV: !!! WARNING !!!
CME: Message code : 63
DME: Description : Error in registry informations
AZI: Required operation not complete !
OPE: Ok per confermare e proseguire con Qview.
Il messaggio può comparire sia in fase di apertura che di chiusura di Qview. Reinstallare Qview.

0.16.1.69 64] Errore nel caricamento del file di linguaggio

TEV: !!! WARNING !!!
CME: Message code : 64
DME: Description : Error in language file operations, no enviroment language defined AZI: Updating library required
OPE: Ok per confermare e proseguire con Qview.
Il messaggio compare quando non è presente il file di linguaggio o quest'ultimo è danneggiato.
E' richiesto un'aggiornamento delle librerie oppure una reinstallazione per ripristinare di Qview per ripristinare o correggere il file di linguaggio.

0.16.1.70 65] Salvataggio progetto

TEV: !!! INFORMATION !!!
CME: Message code : 65
DME: Description : Will be saved object files only.
AZI: Press OK button to continue\\
OPE: Ok per confermare e proseguire con Qview. Il messaggio compare all' inizio dell'operazione "Save Project As", e serve ad avvisare l'utente che con questo comando, saranno salvati solamente i files oggetto:
.DBG, .RUN, .SYM, .BIN.

0.16.1.71 66] Copia di tutti i file di progetto

TEV: !!! INFORMATION !!!
CME: Message code : 66
DME: Description : Copy of all project files terminate correctly
AZI: Press OK button to continue
OPE: Ok per confermare e proseguire con Qview.
Il messaggio compare alla fine dell'operazione "Copy All Project Files To...", indicando all'utente che l'operazione si è conclusa ed è andata a buon fine.

0.16.1.72 67] Errore nella compilazione

TEV: !!! ERROR !!!
CME: Message code : 67
DME: Description : Error in compilation Task information
AZI: Required operation not complete !
OPE: Ok per confermare e proseguire con Qview.
Il messaggio compare se si verificano errori nei processi di compilazione a livello di codice eseguibile di QView. Se compare, si deve ritentare la compilazione e se il problema persiste, chiudere e riaprire QView.

0.16.1.73 68] Errore di calcolo checksum interno

TEV: !!! FATAL ERROR !!!
CME: Message code : 68
DME: Description : Error in control checksum operations
AZI: The application will be closed !
OPE: Ok per uscire da Qview.

Il messaggio compare se si verificano errori nel processo interno di calcolo dei checksums. Riaprire il QView e lavorare normalmente. Se il problema persiste, chiamare il fornitore del programma.

0.17 Appendice I: Errori durante il download

Durante la fase di download applicativo possono verificarsi degli errori per i quali non è possibile concludere l'operazione correttamente. Al presentarsi di uno di questi errori si devono controllare i cavi di collegamento seriale e ritentare l'operazione. Se si dovessero ripresentare contattate il fornitore del sistema.

0.17.0.1 1] Not found .RUN file for download

Non si riesce a creare il file nella directory temp di Windows necessario per acquisire i simboli dalla CPU.
Non si riesce a creare il file nella directory temp di Windows necessario per acquisire i dati dalla CPU.
Non è stato trovato il file per il download

0.17.0.2 2] Errore di Framing

Durante la trasmissione/ricezione seriale si verifica un errore di framing, cioè un errore che riguarda la lunghezza della parola, il numero sbagliato su bit di dati o stop. E' segnalato direttamente dal sistema.

0.17.0.3 3] Errore di Overrun

Durante la trasmissione/ricezione seriale si verifica un errore di overrun, che avviene quando i caratteri ricevuti, e non ancora processati, sono stati sovrascritti da caratteri appena giunti; in pratica c'è un overflow nel buffer di ricezione. E' segnalato direttamente dal sistema.

0.17.0.4 4] Errore di Parity

Durante la trasmissione/ricezione seriale si verifica un errore di parity, cioè non vi è un match nella parità del pacchetto trasmesso. E' segnalato direttamente dal sistema.

0.17.0.5 5] Error in serial communication

E' scaduto un timeout di ricezione da CPU.

0.17.0.6 6] Error reading data from Qmove

E' stato ricevuto un carattere '?' dalla CPU.
E' stata ricevuta una stringa più corta di quella prevista.

0.17.0.7 7] Error in checksum control

E' stato verificato un checksum diverso da quello previsto.

0.17.0.8 8] No terminator character in QMOVE answer string

Manca il terminatore di stringa nella risposta (per protocollo ASCII).

0.17.0.9 9] Error in opening .SQP file

E' stato rilevato un errore nell'apertura del file simboli recuperato dalla CPU.

0.17.0.10 10] File creation not possible. Verify your media support.

C’è stato un errore nella creazione di un file necessario al download.

0.17.0.11 11] Error in opening .DQP file

E’ stato rilevato un errore nell’apertura del file dati recuperato dalla CPU.

0.17.0.12 12] Empty file .DQP

Il file dati recuperato dalla CPU è vuoto.

0.17.0.13 13] Serial port not open

La porta seriale non è stata aperta.

0.17.0.14 14] Invalid file. Download is not possible.

Si sta tentando di scaricare un file non valido per il download.

0.17.0.15 15] DownLoad Error

E’ stato riscontrato un generico errore durante il download.

0.17.0.16 16] Error in CPU type or version

Il tipo o la versione della CPU QMove non coincide con quella impostata nel file di downlaod.

0.17.0.17 17] DownLoad Error QMOS release

E’ stato verificato un errore tra la release della CPU QMove e quella dichiarata nel file di download.

0.17.0.18 18] DownLoad Error Memory Full

Memoria QMove danneggiata.

0.17.0.19 19] DownLoad Error Out Of Memory

Memoria QMove piena: applicativo o dati troppo grandi.

0.17.0.20 20] Error Downloading data

E’ stato riscontrato dalla CPU un errore nella struttura del file dati trasferito.

0.17.0.21 21] Error Downloading config

E’ stato riscontrato dalla CPU un errore nella struttura della configurazione trasferita.

0.17.0.22 22] Error Downloading symbols

E’ stato riscontrato dalla CPU un errore nella struttura dei simboli trasferiti.

0.18 Appendice L: Errori durante la compilazione Ladder

Il presente documento riporta tutti gli errori e warning che possono avvenire durante la fase di compilazione LADDER.

0.18.0.1 1] Warning Link Out Of Rung Scope

Presente quando viene rilevato un link verticale fuori dall'area di appartenenza di un rung.

0.18.0.2 2] Warning Contact Out Of Rung Scope

Presente quando viene rilevato un contatto fuori dall'area di appartenenza di un rung.

0.18.0.3 3] Warning Function Out Of Rung Scope

Presente quando viene rilevata una funzione fuori dall'area di appartenenza di un rung.

0.18.0.4 4] Warning Coil Out Of Rung Scope

Presente quando viene rilevata una bobina fuori dall'area di appartenenza di un rung.

0.18.0.5 5] Warning Jump Out Of Rung Scope

Presente quando viene rilevata un'istruzione di salto fuori dall'area di apparteneza di un rung.

0.18.0.6 6] Warning Label Without Rung

Presente quando viene rilevato che è presente un'etichetta senza un rung.

0.18.0.7 7] Warning Empty Rung

Presente quando viene rilevato un rung senza nessun elemento ladder connesso.

0.18.0.8 8] Warning Unused Link In An Empty Rung

Presente quando viene rilevato un link inutilizzato in un rung vuoto.

0.18.0.9 9] Error Unbalanced Vertical Link

Presente quando all'interno di un rung viene rilevata la presenza di un link verticale non connesso in uno dei suoi due apici (superiore o inferiore).

0.18.0.10 10] Error Unconnected Vertical Link

Presente quando all'interno di un rung viene rilevata la presenza di un link verticale non connesso.

0.18.0.11 11] Error Unconnected Contact

Presente quando all'interno di un rung viene rilevata la presenza di un contatto non connesso.

0.18.0.12 12] Error Unconnected Function

Presente quando all'interno di un rung viene rilevata la presenza di una funzione non connessa.

0.18.0.13 13] Error Unconnected Coil

Presente quando all'interno di un rung viene rilevata la presenza di una bobina non connessa.

0.18.0.14 14] Error Unconnected Jump

Presente quando all'interno di un rung viene rilevata la presenza di un'istruzione di salto non connessa.

0.18.0.15 15] Error Label Without Name

Presente quando viene rilevata un'etichetta senza nome.

0.18.0.16 16] Error Jump Without Name

Presente quando viene rilevata un'istruzione di salto senza nome.

0.18.0.17 17] Error Empty Reference

Presente quando viene rilevata una variabile di riferimento vuota.

0.18.0.18 18] Error Empty First Index

Presente quando nella variabile di riferimento viene riscontrato che l'elemento First Index non è stato definito.

0.18.0.19 19] Error Empty Second Index

Presente quando nella variabile di riferimento viene riscontrato che l'elemento Second Index non è stato definito.

0.18.0.20 20] Error Invalid Reference

Presente quando viene rilevata una variabile di riferimento non valida.

0.18.0.21 21] Error Unknowed Base Symbol

Presente quando nella variabile di riferimento viene riscontrato che l'elemento Base non è tra i simboli validi.

0.18.0.22 22] Error Unknowed First Index Symbol

Presente quando nella variabile di riferimento viene riscontrato che l'elemento First Index non è tra i simboli validi.

0.18.0.23 23] Error Unknowed Second Index Symbol

Presente quando nella variabile di riferimento viene riscontrato che l'elemento Second Index non è tra i simboli validi.

0.18.0.24 24] Error Invalid Base Data Type

Presente quando nella variabile di riferimento viene riscontrato che l'elemento Base ha un tipo di dato non valido. I tipi di dato sono : FLAG (F), BYTE (B), WORD (W), LONG (L) e SINGLE (S).

0.18.0.25 25] Error Invalid Base Data Group

Presente quando nella variabile di riferimento viene riscontrato che l'elemento Base ha un gruppo di dato non valido.

0.18.0.26 26] Error Invalid Base Data Access

Presente quando nella variabile di riferimento viene riscontrato che l'elemento Base ha un tipo di dato non valido.

0.18.0.27 27] Error Invalid First Index Data Type

Presente quando nella variabile di riferimento viene riscontrato che l'elemento First Index ha un tipo di dato non valido. I tipi di dato sono : FLAG (F), BYTE (B), WORD (W), LONG (L) e SINGLE (S).

0.18.0.28 28] Error Invalid First Index Data Group

Presente quando nella variabile di riferimento viene riscontrato che l'elemento First Index ha un gruppo di dato non valido.

0.18.0.29 29] Error Invalid Second Index Data Type

Presente quando nella variabile di riferimento viene riscontrato che l'elemento Second Index ha un tipo di dato non valido. I tipi di dato sono : FLAG (F), BYTE (B), WORD (W), LONG (L) e SINGLE (S).

0.18.0.30 30] Error Invalid Second Index Data Group

Presente quando nella variabile di riferimento viene riscontrato che l'elemento First Index ha un gruppo di dato non valido.

0.18.0.31 31] Error Invalid Reference Data Indexation

Presente quando nella variabile di riferimento viene riscontrata una indicizzazione errata.

0.18.0.32 32] Error Invalid Reference Required Command

Presente quando nella variabile di riferimento viene riscontrato qualcosa diverso da un comando device.

0.18.0.33 33] Error Invalid Reference Required Array Name

Presente quando nella variabile di riferimento viene riscontrato qualcosa diverso da un nome array.

0.18.0.34 34] Error Invalid Reference Required Timer Name

Presente quando nella variabile di riferimento viene riscontrato qualcosa diverso dal nome di un timer.

0.18.0.35 35] Error Invalid Reference Required Static Name

Presente quando nella variabile di riferimento viene riscontrato qualcosa diverso dal nome di una variabile STATIC.

0.18.0.36 36] Error Invalid Reference Required Index Name

Presente quando nella variabile di riferimento viene riscontrato qualcosa diverso dal nome di una variabile INDEX.

0.18.0.37 37] Error Invalid Reference Required Device Name

Presente quando nella variabile di riferimento viene riscontrato qualcosa diverso da un nome device.

0.18.0.38 38] Error Invalid Reference Required Device Paramenter

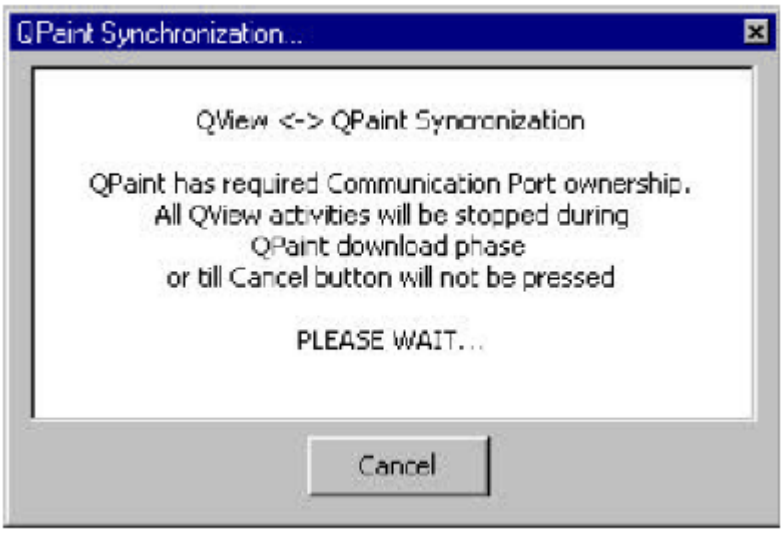
Presente quando nella variabile di riferimento viene riscontrato qualcosa diverso da un parametro device.

0.18.0.39 39] Error Invalid Reference Required Device Command

Presente quando nella variabile di riferimento viene riscontrato qualcosa diverso da un comando device.

0.19 Appendice M: Qpaint syncronization

Una volta trasferito il progetto realizzato con QView 4.1 è possibile avviare il trasferimento di un eventuale progetto realizzato con QPaint 3.0 senza chiudere preventivamente la comunicazione seriale aperta con Qview 4.1. Durante questa operazione il Qview 4.1 segnalerà il seguente messaggio:



0.20 Appendice N: Creare le funzioni utente

È stata introdotta una speciale libreria di funzioni QCL programmate e gestite direttamente dal programmatore. Questo tipo di libreria, è del tutto simile alla libreria di funzioni che viene distribuita con il programma.
Una volta creata la libreria, essa deve essere importata nel Qview 4.1 per poter utilizzare le funzioni che contiene. A tale proposito, è stata aggiunta la voce ***Tools > Upgrade User Qcl Functions Library…***, che serve ad installare ed aggiornare la libreria utente.

Come per tutte le altre librerie del sistema, anche la libreria delle funzioni QCL Utente, ha bisogno di essere installata e/o aggiornata. Per fare ció, occorre creare un insieme di file simile a quello usato per le altre librerie installate. Riportiamo un elenco di quali file sono necessari:

- Upgufunc.lst : File che configura l'installazione.
- 1Ufun003.CIQ : File di libreria.
- UQCLFunc.hlp : File di help.

Vediamo ora nel dettaglio il contenuto e la costruzione dei files riportati sopra.

0.20.1 File “Upgufunc.lst”

È il file che serve al sistema di installazione (aggiornamento) delle librerie di Qview4.1 per installare (aggiornare) la libreria. Esso è del tutto simile ai files “.Ini” di Windows.

Segue una breve descrizione delle sue parti necessarie:

- [Upgrade Files] Sezione del file
- Build=0 Build della libreria (Esempio)
- LibName=1Ufun003 Nome della libreria (Esempio)
- CIQ=1Ufun003.CIQ Nome completo del file di libreria (Esempio)
- HLP=UQCLFunc.hlp Nome completo del file di help associato alla libreria (Es.)

Per esempio un file “Upgufunc.lst” potrebbe essere il seguente:

```
[Upgrade Files]
```

```
Build=1
LibName=1Ufun003
CIQ=1Ufun003.CIQ
HLP=USER_FUN.HLP
```

0.20.1.1 File di libreria funzioni ladder utente "1Ufun003.CIQ"

Il file “1Ufun003.CIQ” dell’esempio precedente viene generato tramite un apposito programma “Decfun20” che viene installato con Qview 4.1. Il programmatore scrive le funzioni in un UNICO file di testo che poi nomina dandogli l’estensione *.qcl, per esempio “Funzioni.qcl”. Una volta avviato il programma Decfun20, apparirà la finestra di figura 1.

Figura 1. Finestra Decfun20

Come si vedrà si deve realizzare un unico file contenente tutte le funzioni utente, questo file è un file di testo che può essere suddiviso in livelli. Il programmmatore, inserendo la propria funzione in questo file, può decidere in che livello inserirla. Al momento della crazione del file CIQ si deve specificare il livello delle funzioni. Impostando un numero da 0 a 1 verranno aggiunte nella libreria tutte le funzioni presenti in questo livello e nei precedenti livelli.
Il programmatore può anche scegliere di realizzare più file sorgenti QCL e da ognuno creare dei file CIQ. Per utilizzare una funzione il programmatore dovrà assicurarsi di aver integrato nel Qview 4.1 la libreria giusta.

0.20.2 Come si scrive una funzione

Il programmatore che vuole sviluppare le proprie funzioni deve scrivere un file di testo ASCII in cui si troveranno le definizioni delle funzioni implementate dalla libreria ed il loro codice. La struttura di ciascuna funzione è la seguente (In grassetto sono riportate le parole chiave da utilizzare, il resto fa parte dell’esempio):

```
FUNCTION
MY_FUNC_NAME
ARGS
  SYSTEM inp01 L
  GLOBAL out01 L
  ARRSYS arrs01 B
LOCALS
  local01 L 10
  local02 arrs01
  local03 B
  local04 L
CODE
  local01[ 2 ] = local02[ 3 ] + local03

  FOR ( inp01 = 2, inp01 LT 24, 1 )
    out01 = out01 + 1
  NEXT

  FOR ( local04 = 1, local04 LT arrs01:dim, 1 )
    local02[ local04 ] = arrs01[ local04 ]
  NEXT
ENDCODE
```

Vediamo ora, sezione per sezione, una breve spiegazione di ciò che va scritto.

Per la scrittura del file di libreria funzioni QCL si consiglia caldamente di utilizzare un editor ASCII generico (non utilizzare WORD).

L’ultima linea di un file di libreria va sempre lasciata vuota. Se non si segue questa regola la libreria risulterà inutilizzabile.

Prima di iniziare il corpo di una funzione bisogna sempre mettere una linea o più di commento per descrivere il funzionamento della funzione. Questo non è tanto una forma di stile ma una richiesta esplicita del Function Expander il quale non accetterà una libreria fatta diversamente.

Dopo la descrizione della funzione bisogna sempre inserire una linea vuota. Se non si segue questa regola la libreria risulterà inutilizzabile.

La sezione FUNCTION seguente serve ad identificare la presenza del nome della funzione. Il nome della funzione segue le stesse indicazioni per i nomi di variabili QCL, con la differenza che non c'è la limitazione dei 12 caratteri massimi. Il nome della funzione, anche se la cosa ovviamente non è vincolante, dovrebbe rispecchiare quanto offerto dalla funzione stessa. Tra la parola riservata FUNCTION e il nome della funzione non vi devono essere linee vuote.

La sezione ARGS contiene la descrizione degli argomenti che verranno passati alla funzione. Come si vede è simile alla definizione di una variabile QCL. Possono essere usati tutti i tipi di dato ad eccezione dei device esterni. Segue ora un esempio di come definire i vari tipi di variabili:

Per le variabili normali:

<tipo variabile><spazio><nome variabile><spazio><dimensione variabile>

SYSTEM sys01 L
GLOBAL gbl01 W
ARRSYS arrs01 B
ARRGBL arrg01 S
DATAPROGRAM arg01 L
STEP arg02 L
INPUT inp01 F
OUTPUT out01 F

Per i device interni:
INTDEVICE<spazio><nome device><tipo device interno>

INTDEVICE dev01 OOPOS2

Questa sezione deve essere scritta correttamente perchè vi sono le informazioni che servono per il controllo sui tipi da parte del compilatore. Tra la parola riservata ARGS e la dichiarazione degli argomenti non vi devono essere linee vuote, come non ve ne devono essere tra le varie parametrizzazioni. All'interno di una dichiarazione è permesso solamente un carattere di spazio come separatore tra i vari parametri.

La sezione LOCALS contiene le variabili usate localmente dalla funzione. Non occorre indicare alcun tipo, perchè il compilatore le ridefinisce come GLOBAL cioè non ritentive. Nel caso si volessero definire degli array locali ciò sarà possibile in due modi:

1) come la definizione standard cioè
<nome array><spazio>< tipo array><spazio><dimensione array> (esempio: pippo B 23)

2) sarà possibile " duplicare " le caratteristiche di un array passato come argomento. Come esempio si può vedere la dichiarazione della variabile locale "local02" che avrà le stesse caratteristiche dell'array "arrs01" (comprese le dimensioni).

Per poter usare gli array ARRSYS & ARRGBL, le variabili static e le index di un DATAGROUP sarebbe necessario anche passare la dimensione di tali strutture, in quanto al momento in cui si crea la libreria non si sa a priori quale parametro verrà passato quando verrà chiamata la funzione stessa. Per questo sono stati introdotti qualcosa di simile ai parametri device in modo che si possano recuperare tali informazioni. Così per gli array basterà scrivere nel codice <array:dim> che il compilatore sostituirà nel codice macchina la dimensione di quell'array. Lo stesso dicasi per le static con <static:nprog> e per le index, in cui sarà possibile utilizzare <index:nprog> e <index:nstep>. Usando queste nuove funzionalità sarà possibile scrivere codice QCL che non è legato alle variabili dell'applicativo che userà la funzione, ma che verrà adattato al momento della compilazione. Una cosa importante è che il risultato delle funzionalità che restituiscono le dimensioni degli array e dei datagroup non supera il valore 32767. Segue ora un esempio di come definire i vari tipi di variabili locali:

Per le variabili normali:

<nome variabile><spazio><dimensione variabile>

local01 L local02 W

Per gli array:

<nome variabile><spazio><dimensione variabile><spazio><dimensione array>

local03 B 10

<nome variabile><spazio><nome array da cui copiare informazioni>

local04 arrSetpoint

Per i timers:

<nome timer>

local05

Tra la parola riservata LOCALS e la dichiarazione delle variabili locali non vi devono essere linee vuote, come non ve ne devono essere tra le varie parametrizzazioni. All'interno di una dichiarazione è permesso solamente un carattere di spazio come separatore tra i vari parametri.

La sezione CODE contiene infine il codice della funzione, che sarà copiato nel file (nascosto) che verrà successivamente compilato.

È stata implementata, e quindi riconosciuta da parte del Function Expander, la possibilità di definire, in maniera facoltativa, dei

prototipi diversi per le funzioni. Cioè è possibile definire diverse liste di argomenti e variabili locali a fronte dello stesso nome e dello stesso codice QCL. Questo permette di evitare la duplicazione di codice necessario per poter definire funzioni con lo stesso nome ma con argomenti diversi, nel tipo e nel numero. Queste sono le due implementazioni possibile per una semplice funzione di Blink.

```
;-----------------------------------------------
; Blink
;
; Blink(OUTPUT out01 F, CONST time01 L)
;-----------------------------------------------

FUNCTION
  Blink
ARGS
  OUTPUT out01 F
  CONST time01 L
LOCALS
  tmLamp
CODE
  IF ( tmLamp AND out01 )
    RESOUT out01
    tmLamp = time01
  ENDIF
  IF ( tmLamp AND NOT out01 AND time01 GT 0 )
    SETOUT out01
    tmLamp = time01
  ENDIF
ENDCODE

;-----------------------------------------------
; Blink
;
; Blink(SYSTEM out01 F, CONST time01 L)
;-----------------------------------------------

FUNCTION
  Blink
ARGS
  SYSTEM out01 F
  CONST time01 L
LOCALS
  tmLamp
CODE
  IF ( tmLamp AND out01 )
    out01 = 0
    tmLamp = time01
  ENDIF
  IF ( tmLamp AND NOT out01 AND time01 GT 0 )
    out01 = 1
    tmLamp = time01
  ENDIF
ENDCODE
```

Quindi si devono definire 2 funzioni diverse a causa dei diversi tipi di argomento anche se poi il codice della funzione è perfettamente identico. Vediamo invece adesso come funziona la seconda sintassi:

```
;-----------------------------------------------
; Blink
;
; Blink(OUTPUT out01 F, CONST time01 L)
; Blink(SYSTEM out01 F, CONST time01 L)
;-----------------------------------------------

FUNCTION
  Blink
ARGS
  OUTPUT out01 F
  CONST time01 L
#
  SYSTEM out01 F
  CONST time01 L
LOCALS
  tmLamp
CODE
  IF ( tmLamp AND out01 )
    RESOUT out01
    tmLamp = time01
  ENDIF
  IF ( tmLamp AND NOT out01 AND time01 GT 0 )
    SETOUT out01
    tmLamp = time01
  ENDIF
ENDCODE
```

Come si vede è sufficiente inserire una riga con il carattere '#' per indicare al compilatore che c'è un nuovo prototipo. Il compilatore poi al suo interno espande tali prototipi e crea comunque un oggetto per ogni funzione, in modo che sia completamente autonomo, con i suoi argomenti, le sue variabili locali, il suo codice. Il trucco serve solo per rendere più agevole la scrittura delle funzioni nel file QCL. La stessa cosa può essere fatta anche per le variabili locali perchè può rendersi necessario avere variabili interne di dimensioni diverse in base agli argomenti del prototipo. Questo serve di solito per le funzioni che trattano array. Segue ora un esempio:

```
;-----------------------------------------------
; SortArrayDown
;-----------------------------------------------

FUNCTION
  SortArrayDown
ARGS
  ARRSYS arr S
  GLOBAL found F
LOCALS
  bound L
  loct L
  locj L
  temp S
  time1
CODE
  bound = arr:dim
  loct = 1

    WHILE ( loct )
      loct = 1
```

```
      FOR ( locj=1, locj LT bound, 1 )
        IF ( arr[ locj ] LT arr[ locj+1 ])
          temp = arr[ locj ]
          arr[ locj ] = arr[ locj+1 ]
          arr[ locj+1 ] = temp
          loct = locj
        ENDIF
        IF time1
          WAIT 1
          time1 = 180
        ENDIF
      NEXT

      IF ( loct EQ 1 )
        BREAK
      ELSE
        bound = loct
      ENDIF
    ENDWHILE
      found = 1
ENDCODE

;-----------------------------------------------
; SortArrayDown
;-----------------------------------------------

FUNCTION
  SortArrayDown
ARGS
  ARRSYS arr L
  GLOBAL found F
LOCALS
  bound L
  loct L
  locj L
  temp L
  time1
CODE
  bound = arr:dim
  loct = 1

  WHILE ( loct )
    loct = 1
    FOR ( locj=1, locj LT bound, 1 )
      IF (arr[ locj ] LT arr[ locj+1 ])
        temp = arr[ locj ]
        arr[ locj ] = arr[ locj+1 ]
        arr[ locj+1 ] = temp
        loct = locj
      ENDIF
      IF time1
        WAIT 1
        time1 = 180
      ENDIF
    NEXT

    IF ( loct EQ 1 )
      BREAK
    ELSE
      bound = loct
    ENDIF
  ENDWHILE
  found = 1
ENDCODE
```

Con la seconda sintassi la definizione della funzione diventa:

```
;-----------------------------------------------
; SortArrayDown
;-----------------------------------------------
FUNCTION
  SortArrayDown
ARGS
  ARRSYS arr S
  GLOBAL found F
#
  ARRSYS arr L
  GLOBAL found F
LOCALS
  bound L
  loct L
  locj L
  time1
$
  temp S
$
  temp L
CODE
  bound = arr:dim
  loct = 1

  WHILE ( loct )
    loct = 1
    FOR ( locj=1, locj LT bound, 1 )
      IF ( arr[ locj ] LT arr[ locj+1 ])
        temp = arr[ locj ]
        arr[ locj ] = arr[ locj+1 ]
        arr[ locj+1 ] = temp
        loct = locj
      ENDIF
      IF time1
        WAIT 1
        time1 = 180
      ENDIF
    NEXT

    IF ( loct EQ 1 )
      BREAK
    ELSE
      bound = loct
    ENDIF
  ENDWHILE
  found = 1
ENDCODE
```

In questo modo posso definire un gruppo di variabili locali comuni, e indicare quali variabili rendere specifiche per i vari prototipi. Basta separare le varie locals con il carattere '$'. ATTENZIONE NOTA IMPORTANTE: le variabili locali vengono associate in ordine di inserimento con i vari prototipi; cioè la prima variabile locale va con primo prototipo e così via per le altre. Se c'è un errore nell'inserimento delle variabili locali, l'associazione segue gli errori.

0.20.3 LIVELLI DI FUNZIONI

Come è stato detto precedentemente le funzini scritte possono essere suddivise in livelli. Al momento della codificazione del file QCL, con le funzioni scritte, tramite il programma Decfun20 viene chiesto fino a quale livello codificare, le funzioni appartenenti a livelli superiori non verranno codificate.
Per creare i livelli bisogna inserire la parola chiave LEVEL seguita dal numero del livello. Per esempio:

```
LEVEL 0
;-----------------------------------------------
; Foo00
;-----------------------------------------------

FUNCTION
  Foo00
ARGS
  GLOBAL found F
LOCALS
  bound L
CODE
  bound = 1
  found = 1
ENDCODE

LEVEL 1
;-----------------------------------------------
; Foo01
;-----------------------------------------------

FUNCTION
  Foo01
ARGS
  GLOBAL found F
LOCALS
  bound L
CODE
  bound = 1
  found = 1
ENDCODE

LEVEL 2
;-----------------------------------------------
; Foo02
;-----------------------------------------------

FUNCTION
  Foo02
ARGS
  GLOBAL found F
LOCALS
  bound L
CODE
  bound = 1
  found = 1
ENDCODE

...
```

l'etichetta LEVEL N è la stringa aggiuntiva che indica che da quel punto in poi è attiva la codifica (se rispettato il livello) fino al LEVEL successivo che determina una nuova analisi. Se non viene incontrato un altro LEVEL, resta sempre attiva l'impostazione dell'ultimo LEVEL incontrato; se ⇐ parametro di riga di comando si codifica tutto, se > non si codifica più niente.

0.20.3.1 Note per il programmatore

Nella scelta dei nomi delle variabili argomento o locali bisogna fare attenzione a non utilizzare stringhe che corrispondano all'inizio di altri nomi. Per esempio l'utilizzo dei nomi *sotto*
e
sottoscala
può dare dei risultati impredicibili al momento della compilazione.

È possibile scrivere cicli FOR e/o WHILE all'interno di funzioni QCL.

È possibile richiamare funzioni QCL contenenti cicli FOR e/o WHILE dall'interno di cicli FOR e/o WHILE di codice QCL scritto dall'utente.

È possibile scrivere un FOR, un WHILE o una linea vuota o commentata come prima istruzione della parte CODE.

Gli argomenti funzione vanno definiti nella apposita sezione ARGS; essi indicano il tipo e l'ordine con cui vanno passati i parametri reali da parte di chi richiama la funzione nel proprio codice. Il compilatore effettua un controllo sul tipo dell'argomento passato, indicando in caso di incongruenza un errore nella riga di chiamata della funzione stessa.

È possibile definire funzioni diverse con lo stesso nome, ma con parametri diversi; il compilatore va a sostituire il codice corretto verificando in base al numero e ai tipi degli argomenti quale sia la funzione corrispondente. Questo funzionamento è detto polimorfismo delle funzioni.

Ogni funzione potrà definire ed utilizzare alcune variabili locali (F, B, W, L, S, Array tutte solo di tipo GLOBAL) che in fase di pre-compilazione verranno rinominate con nomi codificati dal compilatore in modo che ogni chiamata a funzione abbia il proprio contesto di dati. Queste variabili locali appariranno nelle liste variabili (cioè nel file SYM), ma non nel file CNF, ed avranno un nome formato da una parte fissa (ZZ_ZZqem_) ed una parte numerica incrementata per ogni variabile. Si mettono a disposizione 3 cifre per un totale quindi di 1000 variabili locali complessive.

Si potranno definire Array come variabili locali, ma occorrerà definirne la dimensione, come normalmente accade nel file CNF. Non si potranno usare costanti come dimensione perchè tali valori potrebbero essere modificati dall'utente e questo potrebbe creare problemi alla funzione di libreria, che deve essere sufficiente a sè stessa.

Se una stessa funzione viene chiamata all'interno dello stesso task il codice viene sostituito tutte le volte.

In una funzione QCL non esiste il concetto di valore di ritorno diretto presente in alcune delle funzioni standard del linguaggio QCL come SIN e SQRT. Per ottenere un valore di ritorno da una funzione QCL è necessario avare un argomento della funzione che funge da risultato. Questa è anche una soluzione semplice da realizzare. Per esempio, una funzione che calcola la somma potrà essere chiamata così: SUM(x, y, z) in cui x e y sono i due addendi, mentre in z si troverà il risultato. Con questo modo di procedere non sarà possibile usare una funzione direttamente all'interno di espressioni o cicli come nel caso:
WHILE (CAPTURE_RISE_UP(bit) AND ...)
...
ENDWHILE

Questo modo di procedere ha però il vantaggio di poter avere più valori di ritorno da una funzione oppure di impostare più variabili usando la stessa funzione.

Non sarà possibile effettuare il debug di una funzione QCL, per il solito motivo che il codice non appare nel task scritto dall'utente, ma soltanto in quello passato al compilatore. Tuttavia sarà possibile impostare un breakpoint nella riga contenente la funzione o comunque eseguire uno step a step, passando dalla funzione. Se il PCounter in QView si trova sulla riga contenente la chiamata della funzione un comando di step (F8) provocherà l'esecuzione della funzione, come deve essere.

Non è possibile scrivere funzioni di libreria QCL ricorsive come non è possibile richiamare una funzione di libreria QCL da un'altra funzione di libreria QCL.