Guida del progettista

2013-11-15
Version 5.0.0

Questo manuale e' stato tradotto da Francesco Aria per uso personale e quindi non va inteso come documentazione ufficiale del progetto , si declina ogni responsabilita' per danni di ogni tipo derivanti dall'utilizzo di questa versione tradotta in lingua italiana.
[http://www.flussiliberi.it/]
Traduzione V1.0.0

---

2 Introduzione

Proview e' una soluzione moderna e potente per il controllo di processo. Contiene tutte le funzioni normalmente necesarie per il controllo sequenziale, la regolazione, acquisizione dati, comunicazione, supervisione, ecc.

La configurazione di un sistema Proview viene eseguita graficamente, rendendo l'adattamento dell'applicazione semplice, affidabile e flessibile. Proview e' un sistema distribuito, il che significa che il sistema e' composto da diversi computer, collegati tramite una rete. Attraverso la rete, i computer scambiano dati tra loro. In questo modo, ad esempio, i segnali di misurazione saranno noti su tutte le stazioni di processo e operatore di un sistema Proview.

Un sistema Proview e' definito con oggetti. Ogni oggetto rappresenta un'entita' fisica o astratta nel sistema. Gli oggetti sono strutturati in strutture gerarchiche ad albero,rendendo possibile descrivere il sistema in modo strutturato. La struttura ad albero impone anche una denominazione gerarchica di tutti gli oggetti. Scegliendo con cura i nomi durante la configurazione di un sistema, il nome completo di un oggetto identifichera' la sua funzione o ruolo nel sistema.

Le gerarchie sono divise in due gruppi; uno chiamato la configurazione dell'impianto, che descrive la vista logica di un sistema; e uno chiamato la configurazione del nodo, che descrive la vista fisica del sistema. La configurazione di queste due parti puo' essere eseguita in modo indipendente. Alla fine le due parti sono collegate l'una all'altra.

Per configurare un sistema si utilizza Proview Workbench. Il workbench comprende un database permanente e un numero di strumenti per aiutare a configurare gli oggetti necessari per definire il sistema. Dal workbench si crea un sistema eseguibile e una documentazione sul sistema.

Lo scopo di Proview e' di aiutarti a creare sistemi automatizzati. Supponiamo che tu abbia un processo che desideri controllare, Proview ti aiuta a creare il sistema di controllo per questo processo. Una volta creato il sistema, scoprirai di avere anche creato la documentazione per il sistema.

3 Panoramica

Poiche' questa e' una guida per i progettisti, iniziamo con una descrizione di cio' che implica il design di un sistema di controllo. La descrizione puo' anche funzionare come introduzione ai diversi concetti utilizzati in questa guida.

Il punto di partenza di un designer e', ovviamente, il processo che deve essere controllato dal sistema, ed il primo compito e' quello di conoscere il processo e capire il modo migliore per controllarlo: quali loop di controllo sono necessari, quali collegamenti, come vengono eseguiti l'avvio e l'arresto, in che modo operatori e manutentori devono lavorare con il sistema. Questo e' il riassunto in una specifica di progettazione.

Allo stesso tempo, devi considerare quali informazioni sul processo il sistema di controllo ha bisogno per svolgere il suo compito, cioe' quali sensori dovrebbero essere collocati nell'impianto. Il sistema di controllo deve anche influenzare il processo in vari modi, ad esempio con valvole e motori.
Questo ci porta alla stesura di una lista di segnali, che e' una lista di tutti i segnali di input e output del sistema.

A questo punto viene sollevata la questione su quale sistema di controllo utilizzare, e un'alternativa ovviamente e' Proview. E' inoltre necessario decidere quale sistema I/O utilizzare e come dividere la funzione tra le diverse stazioni di processo.

Sistemi di I/O

Il compito del sistema di I/O e' di portare i segnali dal processo al sistema di controllo e di inviare segnali di risposta per influenzare il processo. I segnali sono di solito digitali o analogici, ma ci sono anche altri tipi come numeri interi e contatori.
e' possibile scegliere tra un sistema di rack e schede collegato alla stazione di processo o I/O distribuiti, ad es. PROFIBUS.

Configurare il sistema

Quando e' il momento di iniziare a configurare il sistema, bisogna prima creare un nuovo progetto nell'Amministratore. L'amministratore e' uno strumento per creare l'ordine tra tutti i progetti, poiche' ce ne possono essere molti allo stesso tempo.

La configurazione di un sistema avviene principalmente creando oggetti in un database, il Workbench. Ci sono una grande quantita' di oggetti per configurare qualsiasi cosa, dai canali IO ai programmi PLC.

Il Manuale di riferimento degli oggetti di Proview contiene oltre 800 diversi tipi di oggetti.
Gli oggetti sono collocati in una struttura ad albero e si utilizza uno strumento chiamato Configurator per creare oggetti e navigare nell'albero degli oggetti.

L'albero degli oggetti e' diviso in due parti: la gerarchia Plant e la gerarchia Node.
La gerarchia Plant descrive le diverse funzioni dell'impianto e del processo, mentre la gerarchia del nodo descrive l'hardware del sistema di controllo, con computer, rack I/O e schede I/O.

Quando il sistema di controllo, in un secondo momento, viene avviato in runtime, una copia

dell'albero degli oggetti viene creata in un database in tempo reale, rtdb.
Il trasferimento da Workbench a rtdb viene eseguito con i cosiddetti loadfile, file generati dal Workbench e contenenti tutti i relativi oggetti.

Programma di controllo

Proview offre un linguaggio di programmazione grafico in cui sono programmate la logica, le sequenze Grafcet e i loop di controllo. Questo e' chiamato programma PLC.
Anche il programma PLC E' una parte dell'albero degli oggetti. E' configurato posizionando specifici oggetti programma, PlcPgm, nella gerarchia Plant.
Aprendo un PlcPgm si entra nell'Editor del Plc, in cui si utilizza la programmazione grafica.
I blocchi funzionali vengono creati e collegati in un flusso di segnali digitali e analogici, in cui i segnali di ingresso vengono recuperati sul lato sinistro, trasformati in diversi blocchi funzione e infine memorizzati nei segnali di uscita sul lato destro.

Un complemento al programma PLC sono i programmi applicativi, scritti nel linguaggio c, c ++ o java. Le applicazioni vengono scritte e avviate come programmi separati e collegate al database in tempo reale tramite un'API.

Simulazione

Il database in tempo reale, il programma PLC e le possibili applicazioni possono essere avviate facilmente nella stazione di sviluppo. Cio' rende possibile testare i programmi in connessione diretta con la programmazione. E' anche possibile scrivere programmi speciali di simulazione che leggono i segnali di uscita, elaborano l'influenza delle uscite sul processo simulato, calcolano i valori dei diversi sensori e copiano questi valori nei segnali di ingresso.

La configurazione e la programmazione del sistema e' quindi un processo in cui si passa dalla configurazione alla programmazione e alla verifica.
Il risultato sono programmi accuratamente testati e una messa in servizio rapida ed efficiente dell'impianto. Si ottengono anche programmi migliori che lavora in modo piu' approfondito attraverso le funzioni, perche' il feedback E' maggiore nel processo di creazione di quanto non implichi la costruzione di un sistema di controllo di processo.

Durante la simulazione e la messa in servizio E' di grande importanza avere accesso a strumenti che consentano di monitorare ed esaminare il sistema e localizzare rapidamente possibili malfunzionamenti. In Proview questo strumento E' chiamato Xtt.
Xtt contiene molte funzioni per esaminare il contenuto del database in tempo reale, per far scorrere il flusso del segnale, per registrare sequenze veloci o lente, ecc.

Interfaccia Operatore

Diversi gruppi di professionisti devono accedere al sistema, gli operatori che gestiscono l'impianto su base giornaliera, i manutentori che occasionalmente correggono alcuni malfunzionamenti, gli ingegneri di processo che richiedono vari dati di processo.
Tutti hanno le loro richieste sull'interfaccia di sistema.
Inoltre le limitazioni tra i vari gruppi di professionisti dovrebbe essere flessibile, operatori che sono sia operatori che manutentori e forse anche ingegneri di processo.
Cio' pone elevate esigenze in termini di funzionalita' e flessibilita' dell'interfaccia operatore.
E' possibile rapidamente e facilmente attraverso i cosiddetti metodi che vengono attivati da popupmen recuperare tutte le informazioni dei vari oggetti che sono memorizzati nel database in tempo reale o in diversi sistemi server.

Grafica di Processo

La grafica di processo E' costruita in un editor grafico. La grafica E' basata su vettori, il che rende tutti i grafici e i componenti scalabili in modo illimitato. I componenti hanno una dinamica preprogrammata per cambiare colore e forma, a seconda dei segnali nel database in tempo reale, o per rispondere ai clic del mouse e impostare i valori nel database. In ogni componente che E' sensibile al clic del mouse o all'input, E' possibile dichiarare l'accesso e consentire o impedire in modo selettivo agli utenti di influenzare il sistema. I grafici di processo possono anche essere esportati in java, oltre a essere visualizzati nell'ambiente operatore ordinario, il che li rende visualizzabili sul Web con tutte le dinamiche.

Supervisione

Se si verificano malfunzionamenti nel sistema, l'operatore deve essere avvisato. Questo viene fatto con oggetti speciali di supervisione che sono configurati nella gerarchia dell'impianto o nel programma PLC e che generano allarmi ed eventi.
Gli allarmi hanno quattro livelli di priorita': A, B, C e D e sono visualizzati all'operatore nell'elenco degli avvisi, nell'elenco degli eventi e nell'elenco degli eventi storici.

L'elenco degli allarmi contiene allarmi non riconosciuti (tacitati) e allarmi attivi. Normalmente un allarme deve essere riconosciuto prima che scompaia dalla lista.
Se lo stato di allarme E' attivo, l'allarme rimane nell'elenco finche' E' attivo.

Gli allarmi sono registrati anche nella lista degli eventi, che visualizza gli eventi in ordine cronologico.

La lista eventi storica E' un database, che registra anche eventi. Qui puoi cercare gli eventi, dichiarando i criteri come prioritari e parte del processo.

Se una parte dell'impianto viene spenta, E' possibile bloccare gli allarmi per evitare di distrarre l'operatore. Le parti dell'impianto bloccate vengono visualizzate nell'elenco Blocco.

Archivio dati

Spesso si desidera seguire il cambiamento di un segnale nel tempo, nella forma di una curva. In Proview ci sono tre tipi di funzioni per questo, DsTrend, DsFast e SevHist.

DsTrend E' una curva memorizzata nel database in tempo reale. Il valore di un segnale viene memorizzato continuamente con un intervallo di 1 secondo o piu'. Per ogni curva c'E' spazio per circa 500 campioni, quindi se si sceglie di memorizzare un nuovo campione ogni secondo, si ha una curva si tendenza del segnale di circa 8 minuti.

SevHist memorizza i segnali in modo simile in un database su disco, il che rende possibile memorizzare i valori per un periodo di tempo piu' lungo rispetto a DsTrend.

DsFast memorizza sequenze piu' rapide, in cui la memorizzazione viene avviata con una condizione di trigg e continua per un tempo specificato. Quando la sequenza E' finita, viene visualizzata come una curva.

4 Struttura del Database

Come abbiamo visto in precedenza, la parte principale della configurazione di un sistema Proview si svolge in un database, il Workbench. Nel Workbench si creano oggetti in una struttura ad albero e ogni oggetto presenta una determinata funzione nel sistema di controllo.
Proview e' orientato agli oggetti, quindi guardiamo un po' piu' da vicino cosa sia realmente un oggetto Proview.

4.1 Oggetti

Un oggetto consiste in una quantita' di dati, che in qualche modo definisce lo stato dell'oggetto o le proprieta' dell'oggetto. La quantita' di dati puo' essere molto semplice, come per And-gate, dove consiste in un valore booleano vero o falso. Tuttavia, un controller PID ha una quantita' di dati piu' complessa. Contiene guadagno, tempo di integrazione, uscita, forza, ecc. Consiste in un mix di valori digitali, analogici e interi. Alcuni valori sono configurati nell'ambiente di sviluppo e alcuni sono calcolati in runtime.

La quantita' di dati e' chiamata corpo dell'oggetto. Il corpo e' diviso in attributi, dove ogni attributo ha un nome ed un tipo. Il corpo di un And-gate e' costituito dall'attributo Status che e' di tipo Boolean, mentre il corpo di un controller PID e' composto da 47 attributi: ProcVal, SetVal, Bias, ForceVal ecc.

Tutti gli oggetti PID hanno la loro quantita' di dati strutturati nello stesso modo, tu dici che sono membri della stessa classe. Gli oggetti PID sono membri della classe PID e le And-gates sono membri della classe And. Una classe e' un tipo di modello di come appaiono gli oggetti appartenenti alla classe, ad esempio quali sono gli attributi e il nome e il tipo degli attributi.

Oltre a un corpo, un oggetto ha anche un'intestazione. Nell'intestazione si trova la classe e l'identita' dell'oggetto e anche la sua relazione con altri oggetti. Gli oggetti sono ordinati in una struttura ad albero e nell'intestazione ci sono collegamenti al genitore e ai fratelli piu' vicini dell'oggetto.

4.2 Volumi

Quando si configura un sistema e si creano oggetti, di solito si conosce a quale nodo appartengano gli oggetti in runtime. E' possibile raggruppare gli oggetti dopo il nodo a cui apparterranno, ma viene creato un raggruppamento piu' flessibile, quindi si raggruppano gli oggetti nei volumi.
Un volume e' un tipo di contenitore per oggetti. Il volume ha un nome e un'identita' e contiene un numero di oggetti ordinati in una struttura ad albero.

Esistono diversi tipi di volumi e il primo con cui si entra in contatto e' un volume di root.
Quando si configura un nodo, di solito si lavora in un volume di root. Ogni nodo e' connesso a un volume di root, ad esempio quando il nodo viene avviato in runtime, il volume di root e i relativi oggetti vengono caricati nel nodo. Di seguito e' riportata una descrizione dei diversi tipi di volumi.

RootVolume

Un volume di root contiene la radice dell'albero degli oggetti in un nodo. All'avvio, il nodo sta caricando il volume di root.

Un nodo e' connesso a un solo volume di root. Inoltre, un volume di root puo' essere caricato in piu' nodi. Quando una stazione di processo e' in esecuzione in produzione, lo stesso volume puo' essere contemporaneamente caricato in una stazione di sviluppo per la simulazione e un terzo nodo puo' eseguire il volume a scopi didattici. Tuttavia, e' necessario considerare che i nodi devono essere eseguiti in diversi bus di comunicazione.

SubVolume

Alcuni oggetti in un nodo possono essere collocati in un volume secondario. La ragione per dividere gli oggetti di un nodo in un volume di root e in uno o piu' sotto volumi potrebbe essere che un numero di persone deve configurare il nodo simultaneamente o che si pianifica di spostare parti del controllo di alcune parti di un impianto in un altro nodo dopo.

ClassVolume

La definizione di classi diverse risiede in un tipo speciale di volume, chiamato ClassVolume. Qui la descrizione di una classe e' costruita con oggetti che definiscono il nome della classe e gli attributi della classe.

Esistono due classi di volumi che vengono sempre inclusi in un sistema Proview, pwrs e pwrb. pwrs contiene le classi di sistema, principalmente le classi utilizzate nelle definizioni di classe.
pwrb contiene classi base, cioe' classi standard necessarie per costruire un processo o una stazione operatore.

DynamicVolume

Un volume dinamico contiene oggetti dinamici, ad esempio oggetti volatili creati in fase di runtime. Se nel sistema e' presente un modulo di pianificazione materiale, viene creato un oggetto per ogni materiale che viene elaborato nell'impianto.
Al termine dell'elaborazione, l'oggetto viene rimosso.

SystemVolume

Il volume di sistema e' un volume dinamico che risiede in ogni nodo e che conserva vari oggetti di sistema.

DirectoryVolume

Il volume della directory esiste solo nell'ambiente di sviluppo.
Qui sono configurati i volumi e i nodi del sistema.

Volume Identity

Ogni volume ha un'identita' univoca, che e' scritta con quattro numeri, separati da punti, ad es. "_V0.3.4.23". Il prefisso _V indica che si tratta di un'identita' di volume.
Per verificare che le identita' del volume siano univoche, esiste un elenco di volumi globale che contiene tutti i volumi. Prima di creare un progetto, i volumi del progetto devono essere registrati nell'elenco dei volumi.

4.3 Attributo

La quantita' di dati per un oggetto e' divisa in attributi.
Ogni attributo ha un nome e un tipo. Segue una descrizione dei tipi di attributi piu' comuni.

Boolean

Gli attributi digitali sono di tipo booleano. Il valore puo' essere true (1) o false (0) .

Float32

Gli attributi analogici sono di tipo Float32, ad esempio un float a 32 bit.

Int32

Gli attributi Integer sono di solito di tipo Int32, cioe' un numero intero a 32 bit.
Esistono anche un numero di altri tipi di interi, ad es. Int8, Int16, Int64, UInt8, UInt16, UInt32 e UInt64.

String

In un attributo di stringa viene memorizzata una stringa di caratteri. Esistono diversi tipi di stringhe con varie lunghezze, ad es. String8, String16, String40, String80, String256.

Time

Il tempo contiene un tempo assoluto, ad es. 1-MAR-2005 12: 35: 00.00.

DeltaTime

DeltaTime contiene un tempo di delta, ad es. 1: 12: 22,13 (1 ora, 12 minuti, 22,13 secondi).

Enum

Enum e' un tipo di enumerazione, utilizzato per scegliere un'opzione di diverse alternative. Puo' essere assegnato un valore intero, in una serie di valori interi, in cui ogni valore e' associato a un nome. Esistono, ad esempio, l'enumerazione ParityEnum che puo' avere i valori 0 (Nessuno), 1 (Dispari) o 2 (Pari).
Enum e' un tipo base e ParityEnum e' un tipo derivato.

Mask

Maschera viene utilizzata quando si sceglie uno o piu' di un certo numero di alternative. Le alternative sono rappresentate dai bit in un numero intero a 32 bit.

Un attributo puo' anche consistere in una struttura dati piu' complessa. Puo' essere una matrice con un numero specificato di elementi e potrebbe anche essere un altro oggetto, un cosiddetto oggetto attributo.

4.4 Class

Una classe e' una descrizione di come deve apparire un oggetto che e' un membro della classe. Un oggetto che appartiene alla classe e' chiamato un'istanza. La classe definisce come i dati delle istanze sono strutturati in attributi di vario tipo, o la rappresentazione

grafica degli oggetti nel PLCeditor o nell'ambiente operatore.

Ogni classe ha un oggetto modello, ad esempio un'istanza della classe che contiene valori predefiniti per gli attributi della classe.

Il basesystem di Proview contiene circa 1000 classi. Vedere il Manuale di riferimento dell'oggetto per una descrizione dettagliata. Il designer puo' anche creare le proprie classi all'interno di un progetto.

4.5 Albero degli oggetti

Gli oggetti in un volume sono ordinati in una struttura ad albero. Nel volume ci sono uno o piu' oggetti principali, ogni oggetto in alto puo' avere uno o piu' figli, che a loro volta possono avere figli, ecc.
Di solito nelle relazioni tra gli oggetti nell'albero si parla in termini di genitore, fratello, figlio, antenato e discendente.

4.6 Nome dell'oggetto

Ogni oggetto ha un nome che e' unico all'interno della sua famiglia di fratelli. L'oggetto ha anche un nome di percorso che e' unico nel mondo. Il nome del percorso include, oltre al nome dell'oggetto, il nome del volume e il nome di tutti gli antenati, ad es

```
VolTrafficCross1:TrafficCross1-ControlSignals-Reset
```

Se vuoi essere piu' specifico e indicare un attributo in un oggetto, aggiungi il nome dell'attributo al nome dell'oggetto interponendo un intervallo , ad es.

```
VolTrafficCross1:TrafficCross1-ControlSignals-Reset.ActualValue
```

Inoltre, un attributo puo' avere diversi segmenti, poiche' un attributo puo' essere costituito da un oggetto. I segmenti del nome dell'attributo sono separati da punti, ad es

```
VolTrafficCross1:Hydr-Valve.OpenSw.ActualValue
```

4.7 Montaggio

Una stazione operatore deve visualizzare i valori di segnali e attributi che risiedono nei volumi delle stazioni di processo. Cio' e' ottenuto da un meccanismo in cui una stazione operatore monta i volumi delle stazioni di processo nel proprio albero degli oggetti.
Un montaggio significa che si blocca un albero di oggetti di un altro volume nel proprio volume principale. Dove nell'albero i volumi sono appesi, e' configurato con un oggetto MountObject.
MountObject indica quale oggetto negli altri volumi montati. Il risultato e' che il MountObject viene visualizzato come oggetto montato, con l'albero dell'oggetto sottostante. Apparentemente sembra che gli oggetti appartengano al proprio volume di root, mentre in realta' risiedono in un altro nodo.

Se si utilizzano sottovolumi, devono anche essere montati in un volume di root per rendere disponibili gli oggetti.

Quando si scelgono punti di montaggio e nomi di punti di montaggio, e' opportuno farlo in modo che gli oggetti abbiano lo stesso nome di percorso in entrambi i volumi.

4.8 Identita' dell'oggetto

Un oggetto ha un'identita' che e' unica. E' costituito dall'identita' del volume e da un indice dell'oggetto che e' univoco all'interno del volume. Un'identita' dell'oggetto e' scritta ad esempio "_O0.3.4.23: 34" dove 0.3.4.23 e' l'identita' del volume e 34 l'indice dell'oggetto. Il prefisso _O indica che si tratta di un'identita' di oggetto.

5 Studio di un caso

In questo capitolo ti daremo un'idea di come viene creato un sistema Proview.
Il processo da controllare e' molto semplice in questo caso di studio
- un'intersezione con quattro semafori - ma ti dara' un'idea dei passi da compiere durante la creazione di un sistema Proview.

I semafori dovrebbero essere in grado di operare in due diverse modalita':

- Normale: i semafori eseguono il normale ciclo di rosso, giallo e verde.
- Flash: i semafori lampeggiano in giallo.

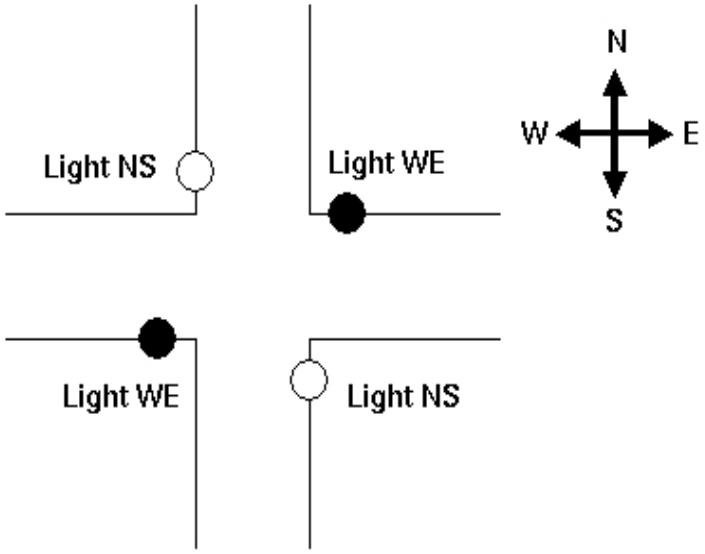


Fig Semafori in un incrocio

La modalita' operativa dei semafori viene decisa da un operatore tramite una postazione operatore o da un tecnico di manutenzione che influenza un interruttore. Il tecnico di manutenzione puo' cambiare modalita' solo se l'operatore ha acceso i semafori in modalita' assistenza.

La figura "Semafori, pannelli di controllo" mostra i diversi interruttori e indicatori necessari all'operatore e al tecnico di manutenzione rispettivamente per essere in grado di monitorare e controllare il sistema. Questi potrebbero essere realizzati con la grafica dell'impianto sulla postazione operatore o con l'hardware.

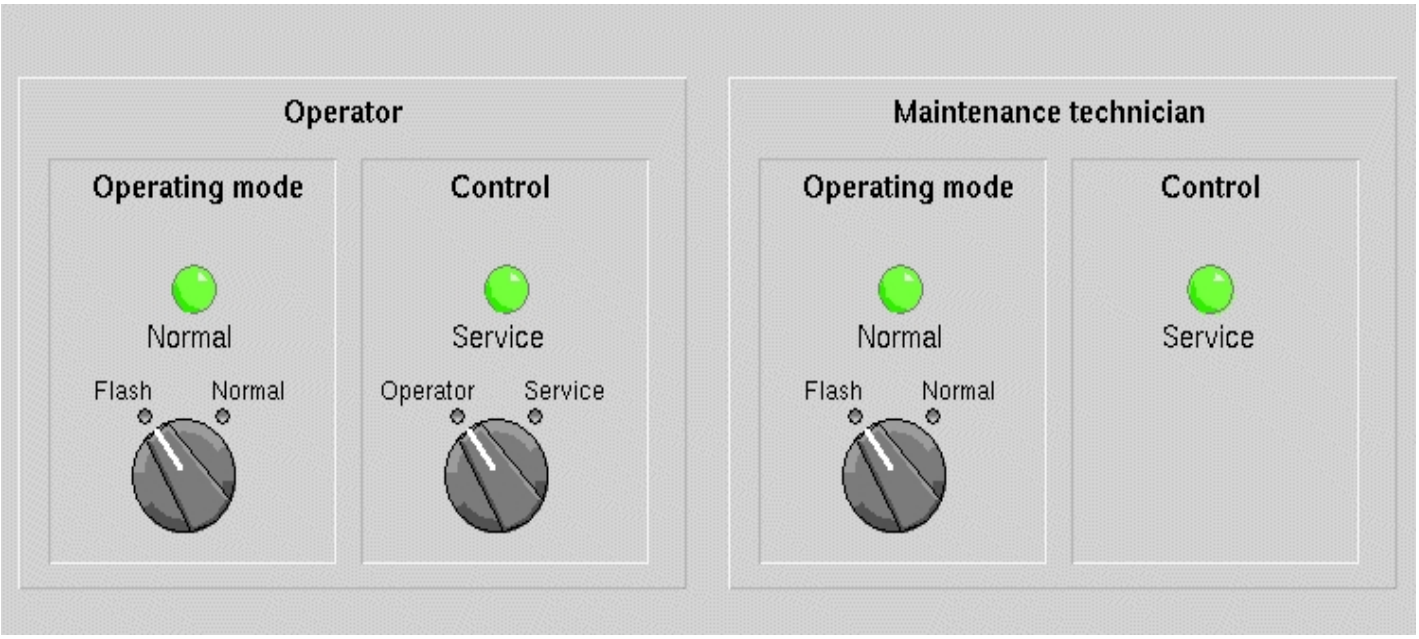


Fig Semafori, pannelli di controllo

5.1 Specifica di I/O

Iniziamo analizzando l'attivita' per decidere quale hardware utilizzare.

UScite Digitali

Abbiamo quattro semafori, ma i semafori nella stessa strada possono essere collegati in parallelo, il che significa che possiamo trattarli come due luci.

Tre uscite per luce: 2 * 3 = 6
Indicazione: modalita' operativa 1
Indicazione: controllo 1

Numero totale di uscite digitali: 8

Ingressi digitali

L'unico input digitale necessario e' per l'interruttore del tecnico di manutenzione.
L'operatore controlla il processo dal display del computer e cio' non richiede segnali fisici.

Interruttore: modalita' operativa 1

Numero totale di ingressi digitali: 1

I/O Analogici

Non sono necessari ingressi o uscite analogici per questo compito.

Specifica della stazione di processo

Quando abbiamo deciso l'I / O necessario, possiamo scegliere l'hardware. Noi scegliamo:

1 Linux PC con rack
1 scheda con 16 digital inputs
1 scheda con 16 digital outputs

Specifica della stazione operatore

1 Linux PC

Specifica della grafica dell'impianto

Abbiamo bisogno di un display da cui l'operatore possa controllare e sorvegliare i semafori.

5.2 Amministrazione

Per prima cosa dobbiamo registrare un nuovo volume, creare un progetto e, se necessario, creare nuovi utenti.
Per questo abbiamo bisogno di:

- Un nome per il progetto. Lo chiamiamo trafficcross1.
- Due volumi, uno per la stazione di processo e uno per la postazione operatore.
- Abbiamo bisogno di tre utenti: uno sviluppatore, un operatore e un tecnico di manutenzione.

Volumi, progetti e utenti sono registrati e creati nello strumento di amministrazione.

Registrare il volume

Per questo progetto, abbiamo bisogno di due volumi, uno per la stazione di processo e uno per la postazione operatore. Sono volumi di root potremo scegliere un'identita' di un volume inattivo nell'intervallo 0.1-254.1-254.1-254. Scegliamo 0.1.1.1 per la postazione operatore e 0.1.1.2 per la stazione di processo, e accediamo alla modalita' volume nell'Amministratore per registrare questi volumi.

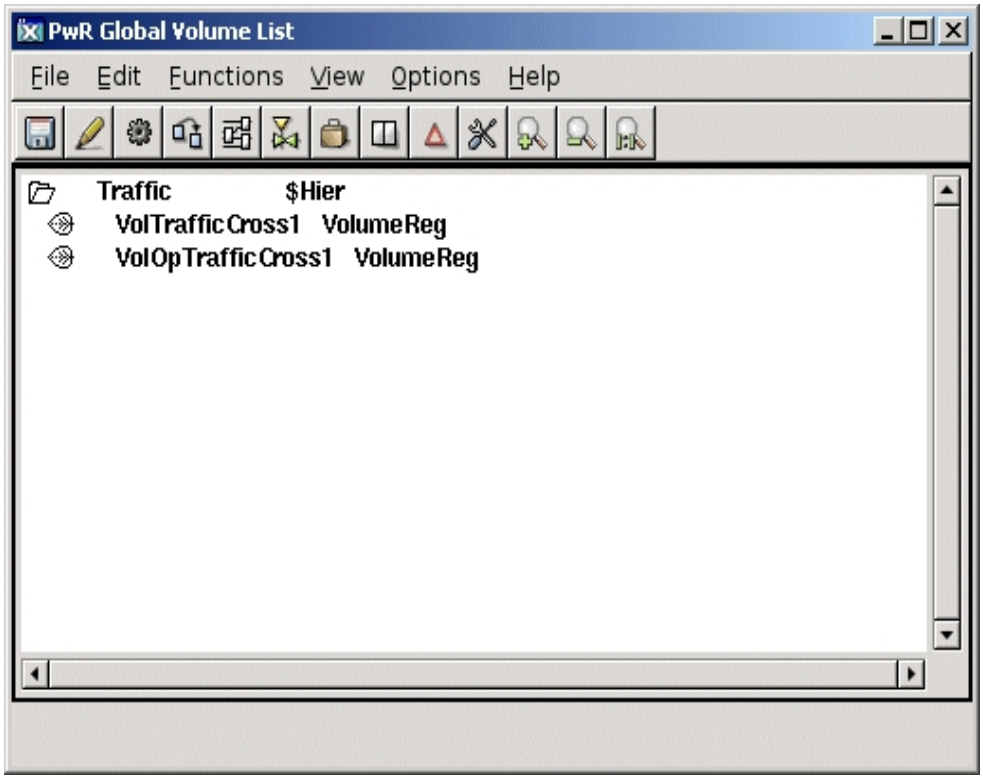


Fig Registrazione del volume

Creare gli utenti

Eric e' uno sviluppatore nel dipartimento del traffico, Carl e' un operatore. Entrambi sono coinvolti in tutti i progetti del dipartimento del traffico, quindi creiamo un gruppo di sistema comune per tutti i progetti e permettiamo loro di condividere gli utenti. Accettiamo i privilegi di sviluppatore e di sistema di Eric, i privilegi di operatore Carl e i privilegi di manutenzione di Lisa.

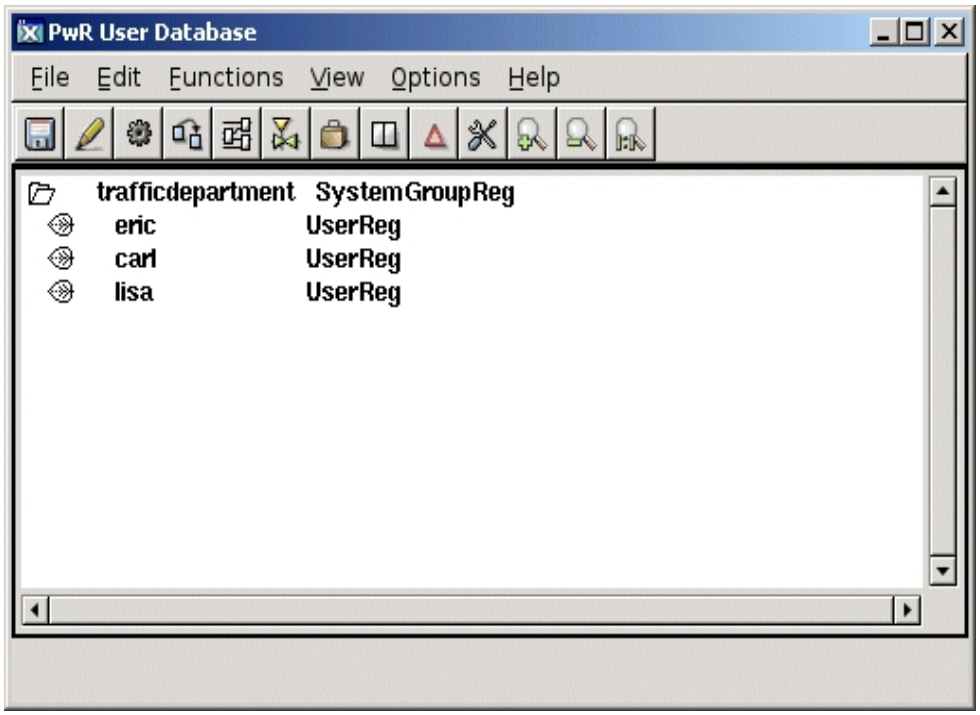


Fig Utenti creati

Creare il progetto

Creiamo il progetto con il nome di gerarchia 'traffic-trafficcross1'.

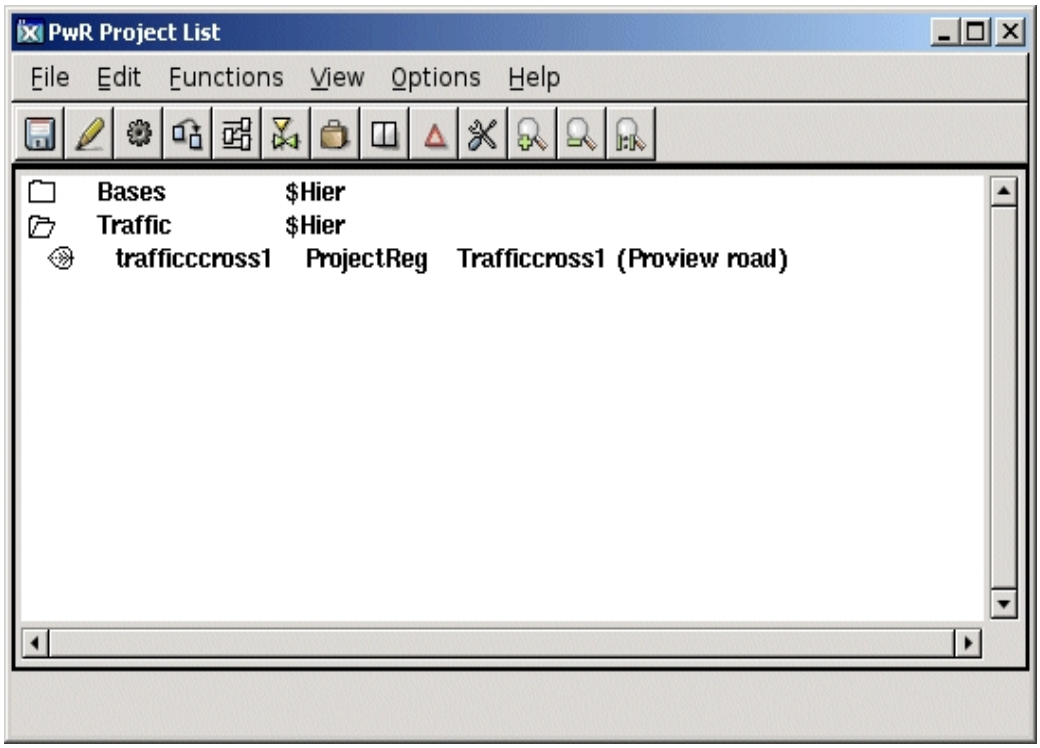


Fig Progetto creato

Configurare il progetto

Il progetto ha un volume di directory in cui sono configurati i nodi e i volumi del progetto.
Nella finestra di sinistra i volumi sono configurati con oggetti RootVolumeConfig.
Nella finestra di destra il processo e la stazione operatore sono configurati con oggetti NodeConfig. Gli oggetti NodeConfig si trovano sotto un oggetto BusConfig che indica in quale bus QCOM i nodi stanno comunicando.

Gli oggetti NodeConfig contiene

- Nodename
- indirizzo ip del nodo

Sotto ogni oggetto NodeConfig c'e' un oggetto RootVolumeLoad che indica il volume da caricare all'avvio di runtime.

Notare anche l'oggetto di sistema con l'attributo SystemGroup a cui e' assegnato il valore 'trafficdepartment'.
Questo garantisce agli utenti l'accesso erico, carl e lisa al progetto.

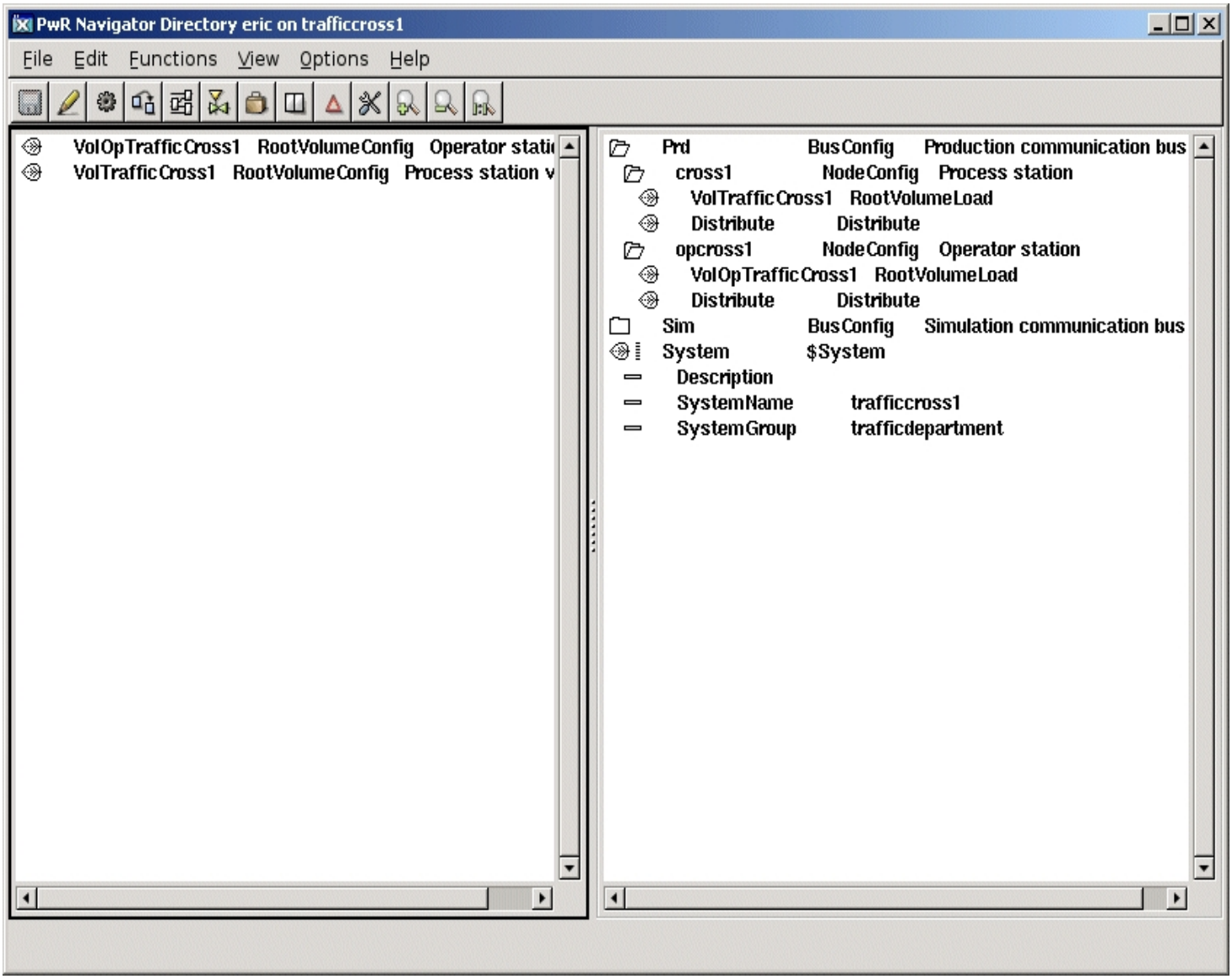


Fig II volume della directory

5.3 Configurazione dell'impianto

Una volta terminata la configurazione del progetto, il prossimo passo e' configurare l'impianto. La configurazione viene eseguita nell'Editor di configurazione.
L'impianto (Plant) e' una descrizione logica della realta', che deve essere controllata e supervisionata.

La stazione di processo

La maggior parte della configurazione viene eseguita nel volume della stazione di processo, VolTrafficCross1.
Questo perche' qui viene configurato tutto l'hardware fisico (I/O), tutti i segnali e i programmi PLC che funzionano con i segnali.

L'impianto e' strutturato gerarchicamente. Un esempio dei livelli dell'impianto puo' essere impianto, processo, parte di processo, componente e segnale. Questi sono i segnali logici rappresentati da oggetti di segnale che saranno collegati a canali fisici.

A volte puo' essere difficile configurare ciascun segnale in una fase iniziale, ma in ogni caso deve essere deciso in che modo i segnali devono essere raggruppati.

La figura seguente illustra come e' stato configurato l'impianto. Vediamo come i segnali sono stati configurati su diversi livelli e anche come i programmi PLC sono configurati nell'impianto.

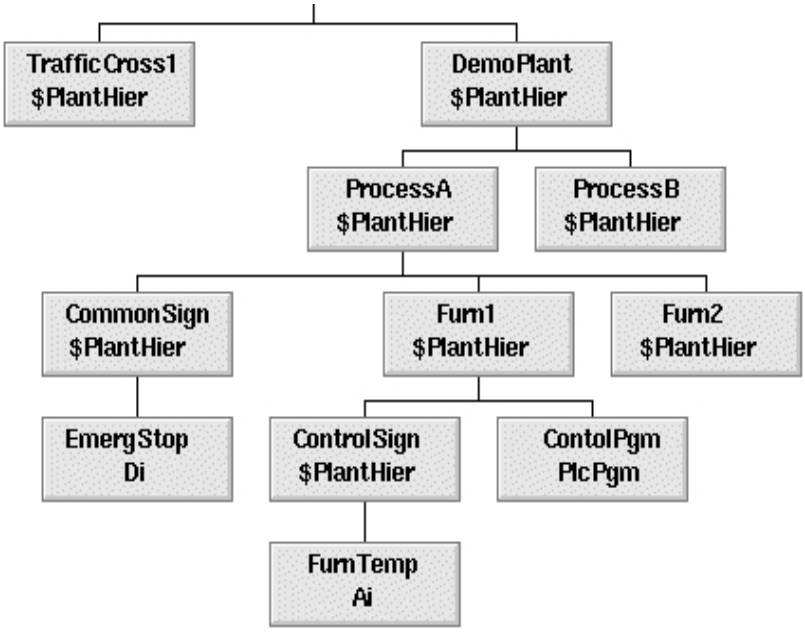


Fig Un esempio di configurazione di un impianto

Scegliamo di chiamare il nostro impianto TrafficCross1 e la seguente struttura:

- Due semafori, ciascuno composto da una lampada verde, una gialla e una rossa.
  Dal momento che le strade percorrono rispettivamente nord-sud e ovest-est, le chiamiamo TrafficLightNS e TrafficLightWE. Ogni lampada richiede un segnale.
  Questi sono segnali di uscita digitali e sono chiamati RedNS, RedWE, ecc.

- Un programma PLC per il controllo dei semafori.

- Un numero di segnali di controllo per selezionare la modalita' operativa e la funzione.
  Scegliamo di metterli in una cartella, ControlSignals.
  La tabella seguente mostra i segnali richiesti.

La figura mostra la configurazione dell'impianto risultante.

Scegliamo di chiamare il nostro impianto TrafficCross1 e la seguente struttura:

Nome Segnale	**Tipo Segnale**	**Funzione**
ServiceSwitch	Di	Un interruttore che il tecnico di manutenzione puo' influenzare per cambiare la modalita' operativa.
OperatorSwitch	Di	Un valore che l'operatore puo' influenzare per cambiare la modalita' operativa.
ServiceMode	Di	Un valore che l'operatore puo' influenzare per cambiare la funzione in modalita' assistenza.
ServiceModeInd	Do	Un segnale che mostra al tecnico di manutenzione che il programma e' in modalita' di servizio.
Mode	Dv	Indica se il programma e' in modalita' normale o lampeggiante.
ModeInd	Do	Indica se il programma e' in modalita' normale o lampeggiante.

Reset	Dv	Un valore che viene utilizzato per ripristinare il programma nella modalita' iniziale.

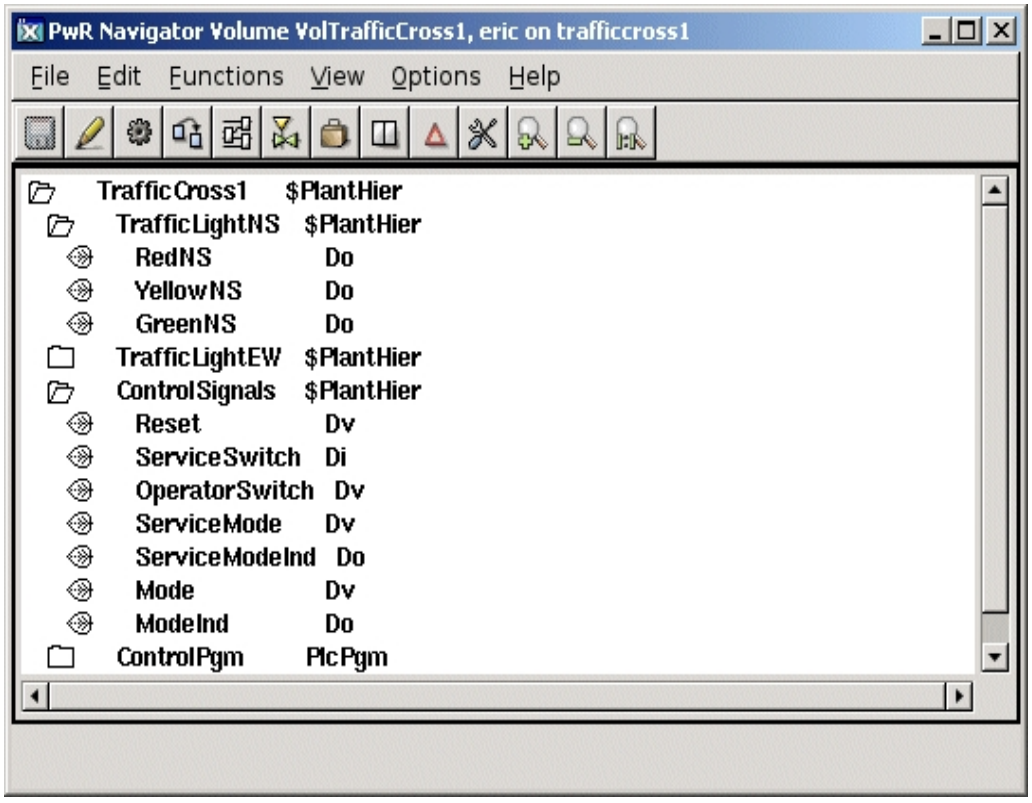


Fig La configurazione dell'impianto dell'incrocio

Come puoi vedere abbiamo un oggetto impianto al livello piu' alto, TrafficCross1 della classe $ PlantHier. Usiamo altri oggetti di classe $ PlantHier per raggruppare i nostri oggetti. Creiamo anche un oggetto, che definisce un programma PLC, l'oggetto ControlPgm della classe PlcPgm.

La postazione operatore

La configurazione della postazione operatore viene eseguita nel volume VolOpTrafficCross1. Nella parte Plant e' presente solo un oggetto mount, che rende disponibile la gerarchia dell'impianto del nodo di processo nella stazione operatore.
Abbiamo montato l'oggetto $ PlantHier piu' in alto, 'TrafficCross1' con un MountObject con lo stesso nome.

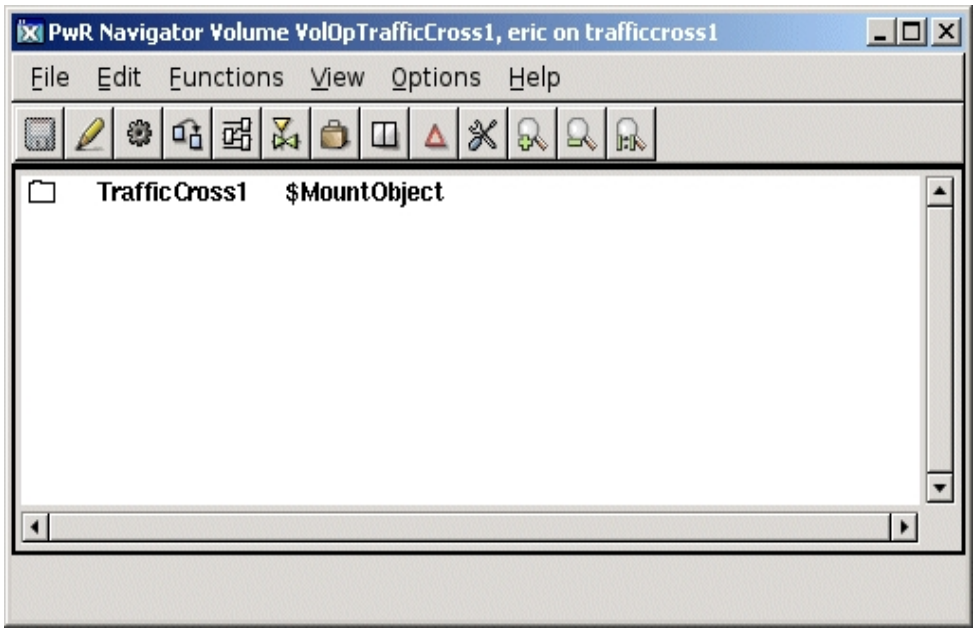


Fig La configurazione dell'impianto nel volume dell'operatore.

5.4 Configurazione del nodo

Una volta configurato l'impianto, andiamo avanti a configurare i nodi.

Stazione di Processo

In questo esempio, scegliamo di iniziare a configurare la stazione di processo. Chiamiamo la stazione di processo "cross1". E' consigliabile fornire i nomi descrittivi delle stazioni di processo. Nella gerarchia dei nodi creiamo un $NodeHier oggetto di tipo 'Nodes' e sotto, un oggetto $Node 'cross1' che configura il nodo.

Nella fase di analisi abbiamo deciso che la stazione di processo dovrebbe essere costituita dal seguente hardware:

- 1 Linux PC
- 1 rack con 16 slots
- 1 scheda con 16 inputs digital (Di channels).
- 1 scheda con 16 outputs digital(Do channels).

Il rack e le schede sono configurate in modo simile a quello che faresti fisicamente. Hai un nodo, posiziona il rack nel nodo, la scheda nel rack e i canali su ogni scheda.

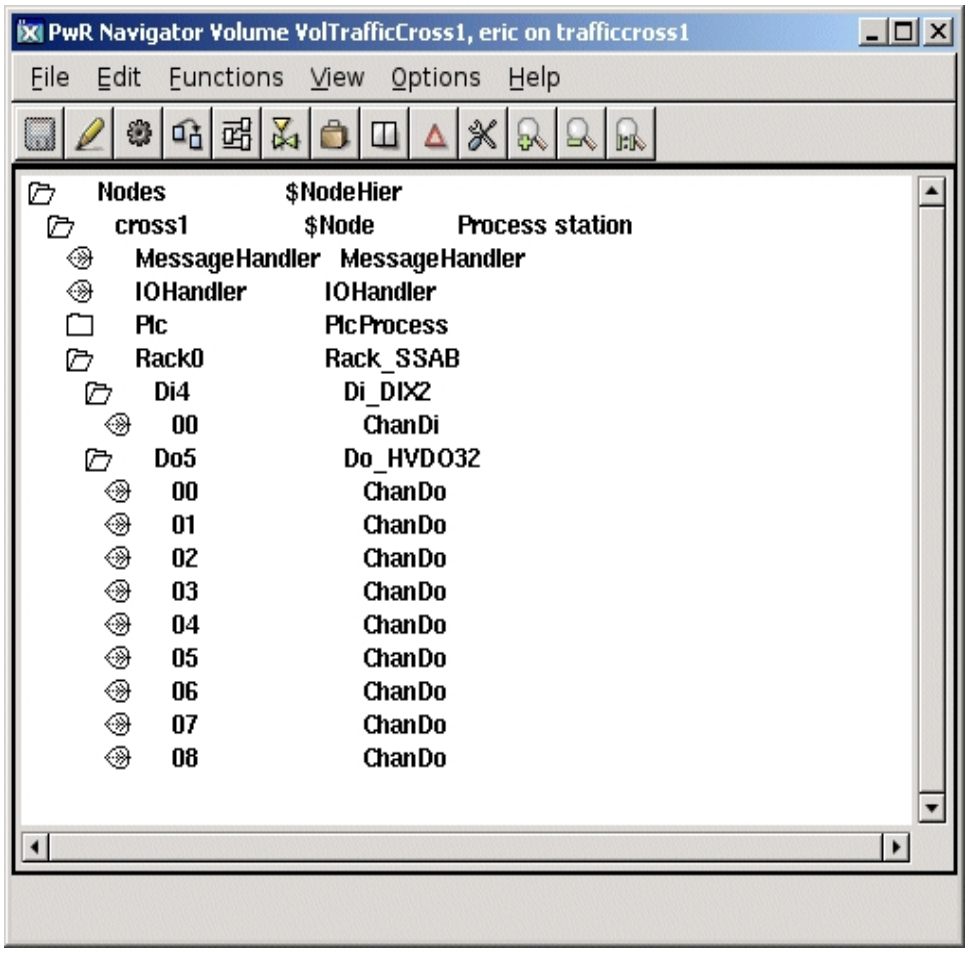


Fig Configurazione del nodo della Stazione di Processo

Configuriamo anche il programma PLC con un oggetto PlcProcess, e sotto questo, un oggetto PlcThread per ogni base temporale. Noi scegliamo una base dei tempi di 100 ms.

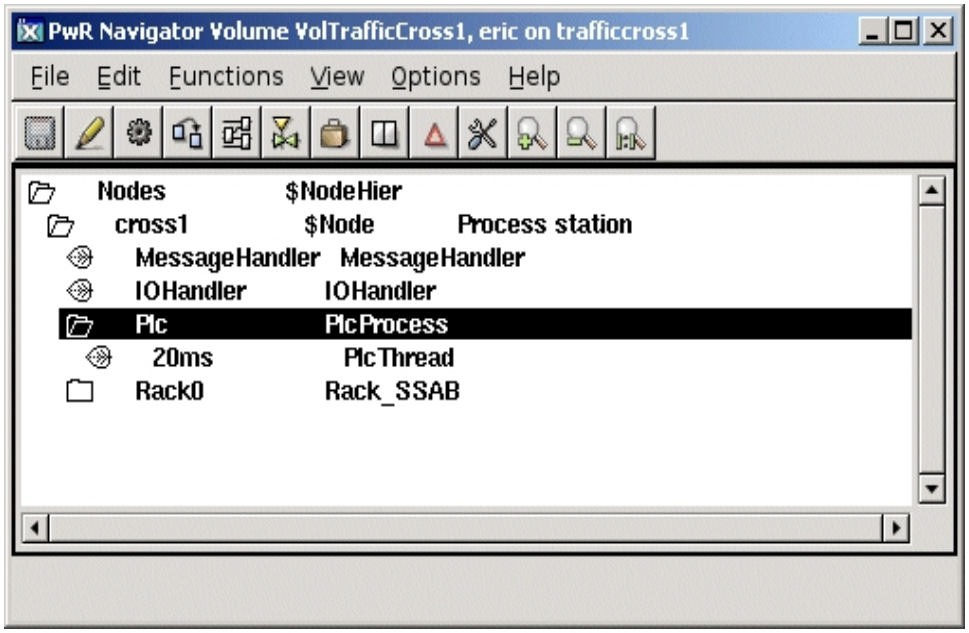


Fig Configurazione timebase del programma PLC

Ogni oggetto ha un numero di attributi che potrebbe essere necessario modificare.

Per capire come cambiare gli attributi, alcuni degli attributi nell'oggetto PlcProcess di seguito vengono modificati.

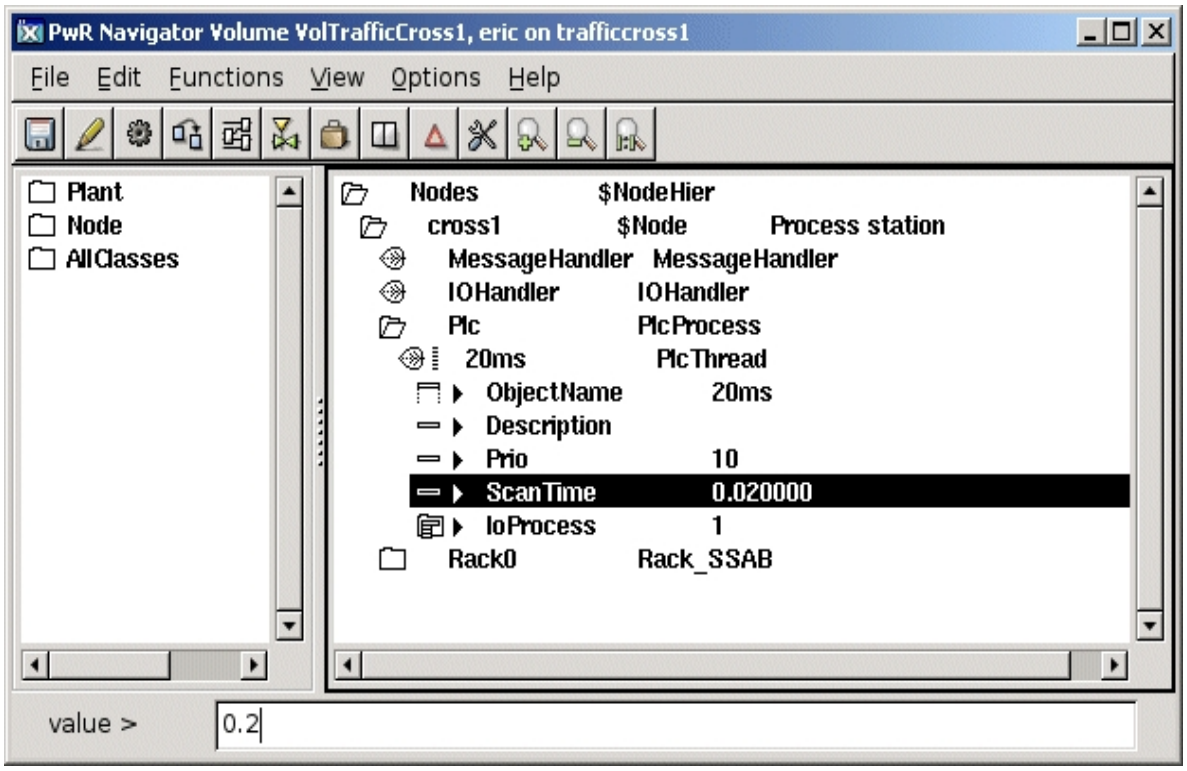


Fig Modifica del valore dell'attributo

Stazione Operatore

La gerarchia dei nodi della stazione operatore e' configurata nel volume VolOpTrafficCross1. Sotto l'oggetto nodo troviamo un oggetto OpPlace che definisce il posto operatore, e sotto questo un oggetto XttGraph per il grafico del processo della stazione operatore.

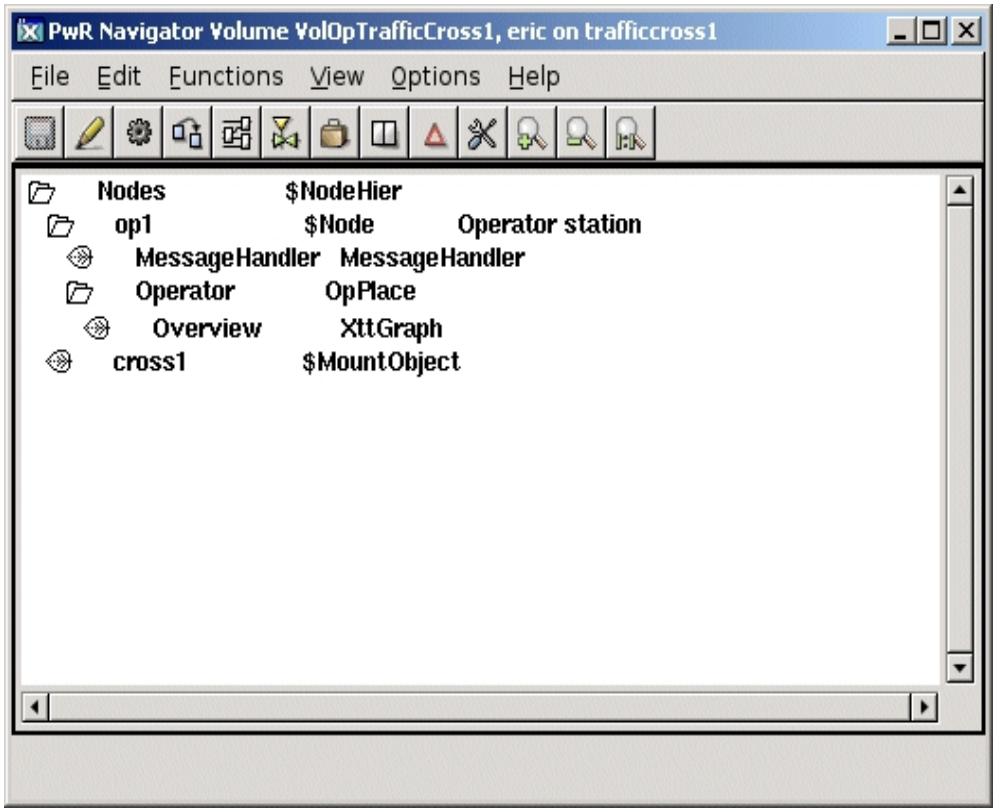


Fig La configurazione del nodo nel volume dell'operatore.

Collegamento dei canali ai segnali

Dopo aver configurato l'impianto e i nodi, e' il momento di collegare i segnali logici ai canali fisici. Ogni segnale logico nella configurazione dell'impianto deve essere collegato a un canale nella configurazione del nodo; un Di a un ChanDi, un Do a un ChanDo, un Ai a un ChanAi e un Ao a un ChanAo, ecc.

e' possibile vedere la connessione come una rappresentazione del cavo di rame tra i componenti nell'impianto e il canale nel rack I/O. Nella figura sottostante c'e' un cavo tra l'interruttore e il canale 0 nella Di-card. Poiche' il Di-signal ServiceSwitch rappresenta lo switch e Di-channel Di4-00 rappresenta il canale, dobbiamo creare una connessione tra questi due oggetti.

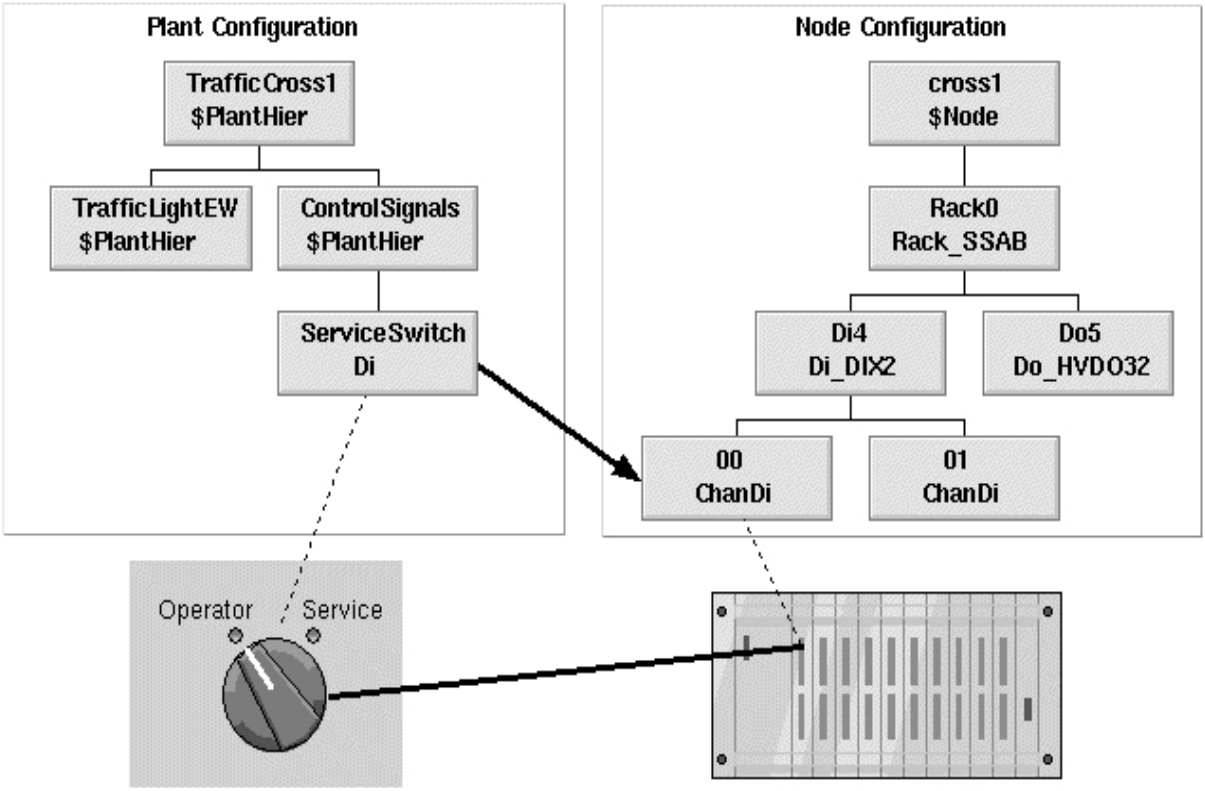


Fig Connessione tra un segnale e un canale

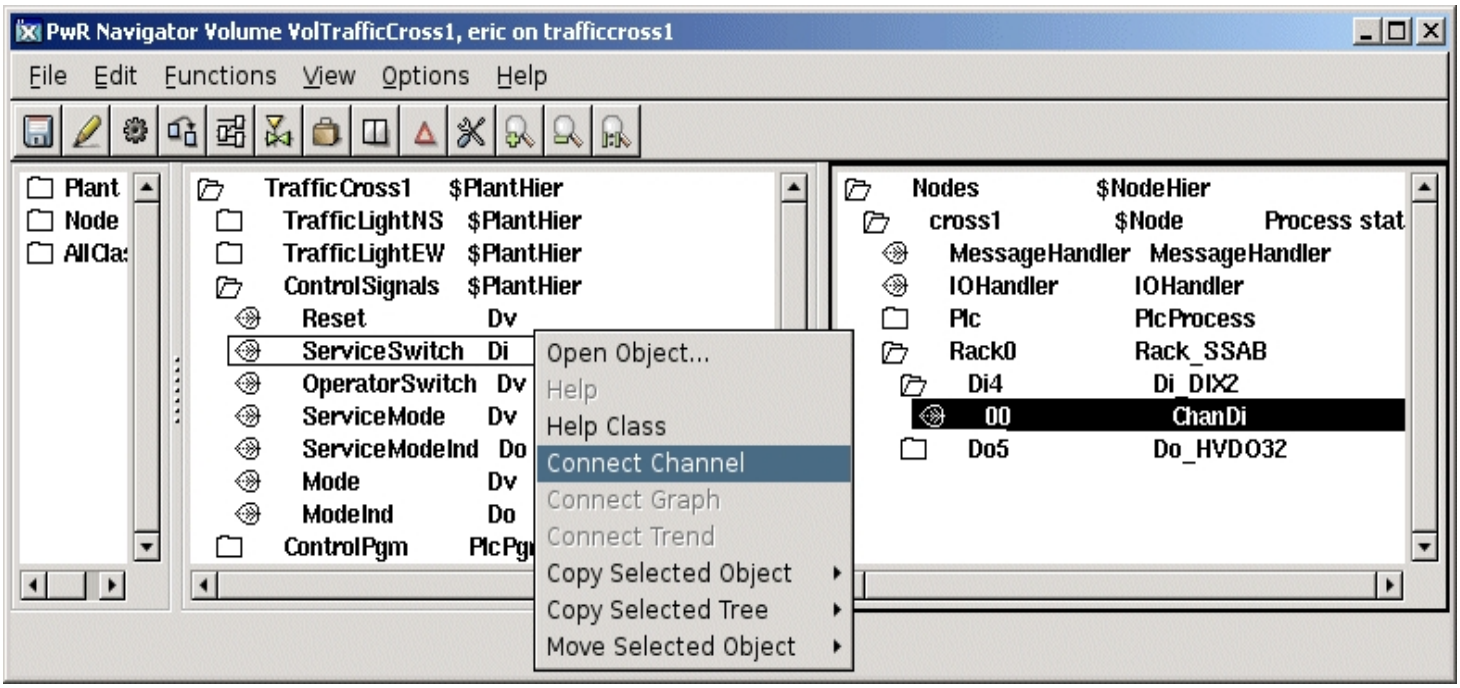


Fig Collegare un segnale a un canale

5.5 Programma PLC

Usiamo Graphical PLC Editor per creare programmi PLC.

Tuttavia, per prima cosa dobbiamo connettere l'oggetto PlcPgm a un oggetto PlcThread nella gerarchia dei nodi.
Questo indica in quale base temporale viene eseguito il programma PLC.

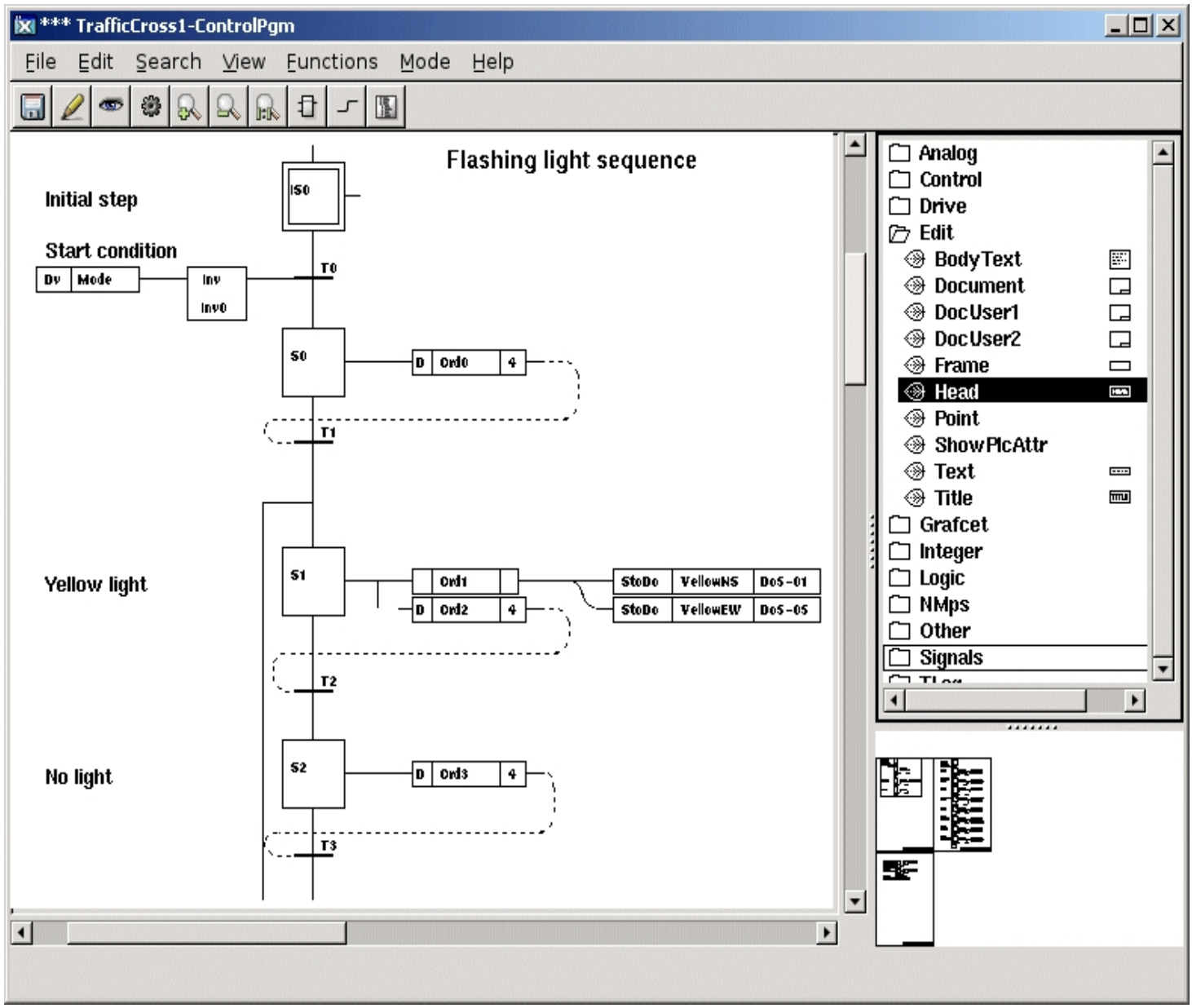


Fig L'editor grafico PLC

Useremo l'Editor PLC per creare un programma di controllo sequenziale per i semafori. Esistono due modi per risolvere il problema relativo alle due modalita' operative per un semaforo, normale e flash:

1. Utilizzare una sequenza Grafcet con rami condizionali, cioe' un ramo per la normale sequenza di modalita' operativa e uno per la sequenza della modalita' operativa flash.
2. Utilizzare due sequenze Grafcet separate con condizioni di partenza differenti.

Noi scegliamo di usare la seconda alternativa. Nel capitolo 4, Programmazione grafica PLC e' possibile trovare una descrizione piu' dettagliata sul Grafcet e sul controllo sequenziale.

I programmi Grafcet si basano sull'attivazione e disattivazione di un numero di passaggi in sequenza. Nelle sequenze lineari puo' essere attivo solo un passo alla volta. Ad ogni passo si lega un numero di ordini che devono essere eseguiti quando il passo e' attivo. Questo puo' essere ad es. per impostare (con un oggetto StoDo) un segnale di uscita digitale, che accende una lampada. I programmi PLC controllano quindi i segnali logici.

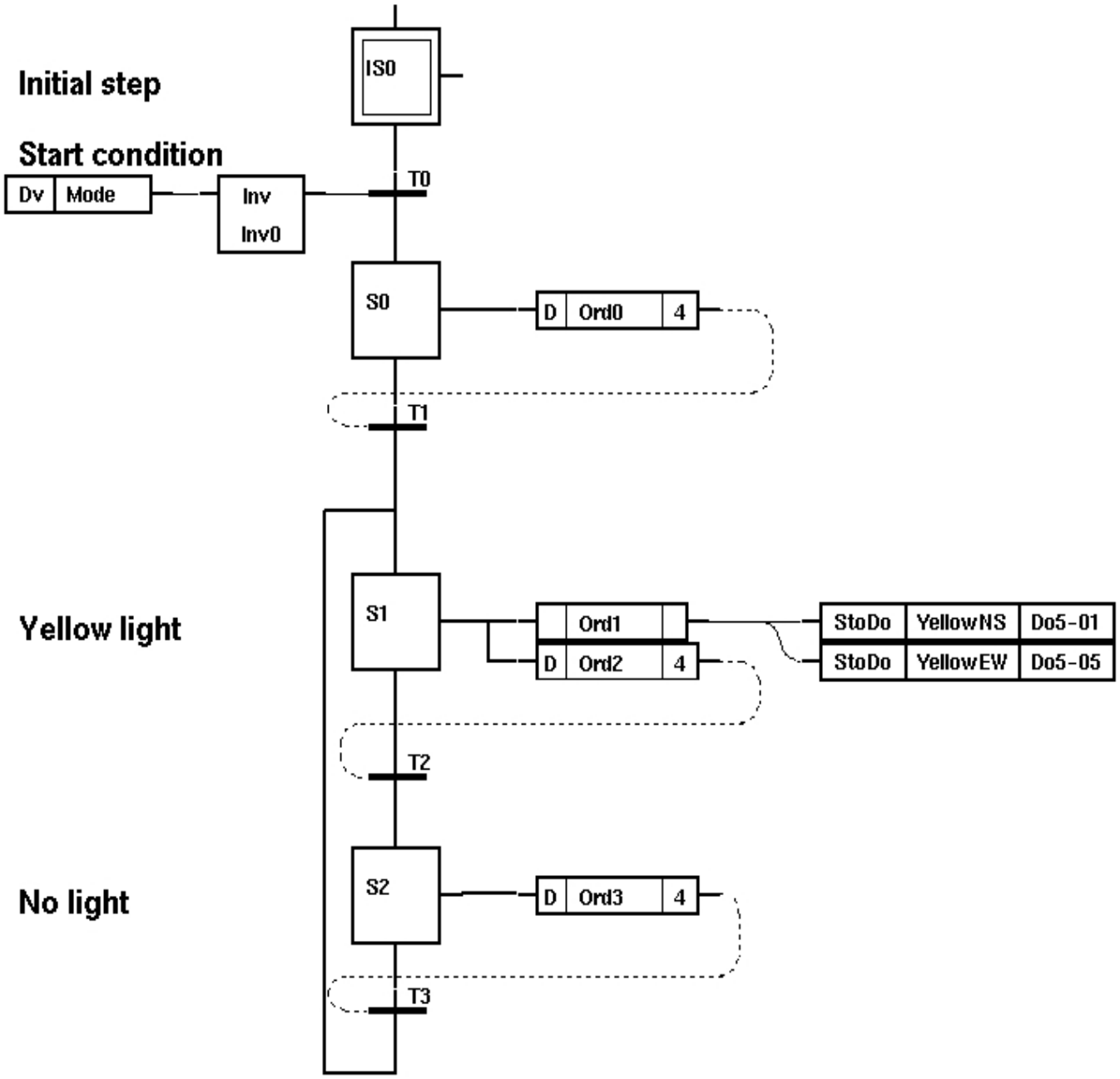


Fig La sequenza delle luci lampeggianti

Questa e' la sequenza che verra' eseguita quando si desidera che le luci lampeggino in giallo.

La condizione di avvio per questa sequenza e' invertita in relazione alla condizione di avvio per la normale sequenza della modalita' operativa. Cio' implica che le due sequenze non possono essere eseguite contemporaneamente.

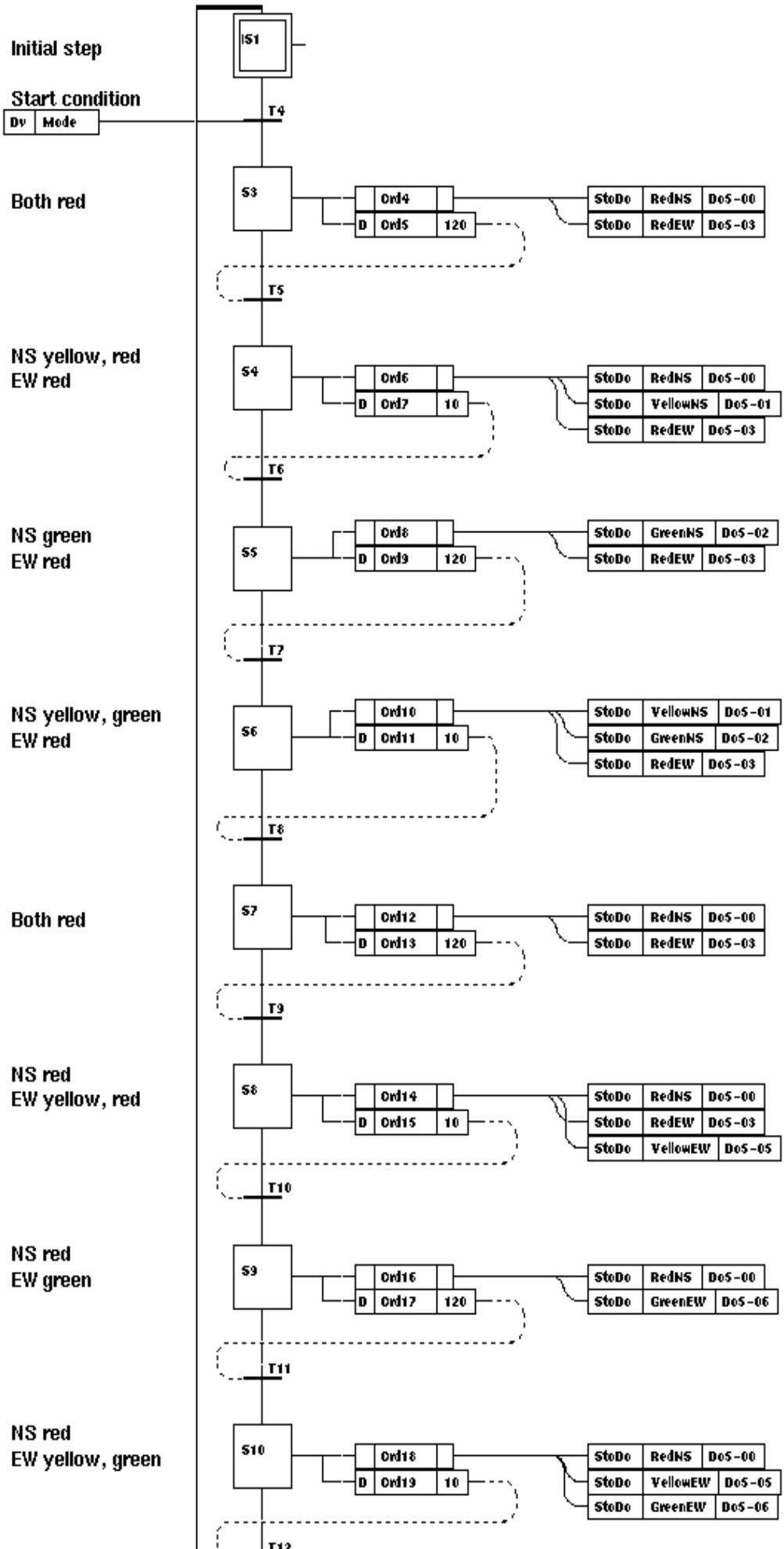

Initial step
S1
Start condition
Dv
Mode
T4
Both red
S3
Ord4
D
Ord5
120
StoDo
RedNS
Do5-00
StoDo
RedEW
Do5-03
T5
NS yellow, red
EW red
S4
Ord6
D
Ord7
10
StoDo
RedNS
Do5-00
StoDo
YellowNS
Do5-01
StoDo
RedEW
Do5-03
T6
NS green
EW red
S5
Ord8
D
Ord9
120
StoDo
GreenNS
Do5-02
StoDo
RedEW
Do5-03
T7
NS yellow, green
EW red
S6
Ord10
D
Ord11
10
StoDo
YellowNS
Do5-01
StoDo
GreenNS
Do5-02
StoDo
RedEW
Do5-03
T8
Both red
S7
Ord12
D
Ord13
120
StoDo
RedNS
Do5-00
StoDo
RedEW
Do5-03
T9
NS red
EW yellow, red
S8
Ord14
D
Ord15
10
StoDo
RedNS
Do5-00
StoDo
RedEW
Do5-03
StoDo
YellowEW
Do5-05
T10
NS red
EW green
S9
Ord16
D
Ord17
120
StoDo
RedNS
Do5-00
StoDo
GreenEW
Do5-06
T11
NS red
EW yellow, green
S10
Ord18
D
Ord19
10
StoDo
RedNS
Do5-00
StoDo
YellowEW
Do5-05
StoDo
GreenEW
Do5-06
T12

Fig La sequenza normale

Il programma per la normale modalita' operativa e' basato su un semaforo che segue la sequenza riportata di seguito :

	North-South	**West-East**
1	Rosso	Rosso
2	Rosso,Giallo	Rosso
3	Verde	Rosso
4	Giallo,Verde	Rosso
5	Rosso	Rosso
6	Rosso	Rosso,Giallo
7	Rosso	<Verde
8	Rosso	Giallo,Verde
9	Torna indietro al passo 1	

Il programma si avvia dal passo iniziale. Se la condizione di avvio e' soddisfatta, la fase S1 diventa attiva e le luci rosse si accendono. Dopo un certo tempo, il passo S1 diventera' inattivo e il passo S2 diventera' attivo, e verra' accesa anche una lampada gialla, e cosi' via.
Quando il passo S8 e' attivo per un certo tempo, verra' disattivato e il passo iniziale verra' nuovamente attivato.

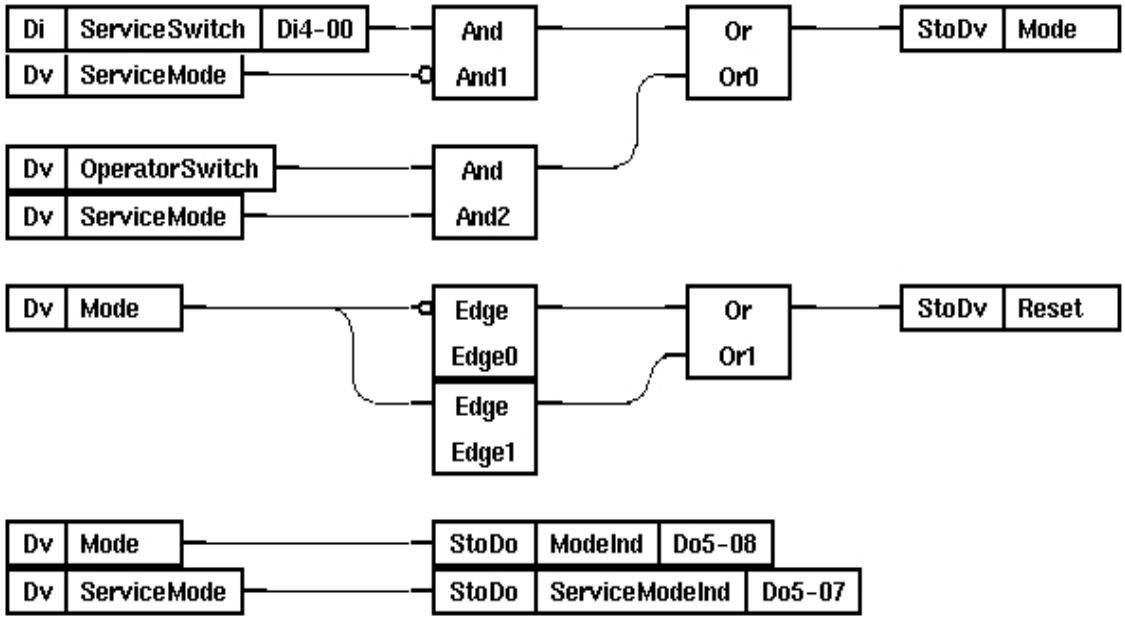


Fig Segnali di Trigger

Il programma sopra mostra la logica che controlla le diverse modalita' operative.

In alto a destra impostate il segnale Dv "Mode". Se questo e' impostato su 1 logico, verra' eseguita la sequenza per la normale modalita' operativa della luce, altrimenti verra' eseguita la sequenza per le luci lampeggianti.

Il segnale Dv "Reset" verra' impostato su 1 logico durante un ciclo di esecuzione quando la modalita' segnale cambia valore. Cio' implica che le due sequenze Grafcet ritorneranno al passo iniziale. La sequenza scelta verra' eseguita nuovamente quando Reset e' impostato su 0 logico.

I programmi PLC creati devono essere compilati prima di poter essere eseguiti su una stazione di processo.

5.6 Grafica dell'impianto

La grafica dell'impianto viene spesso utilizzata come interfaccia tra l'operatore e il processo. La grafica dell'impianto viene creata con l'editor di grafica per l'impianto.

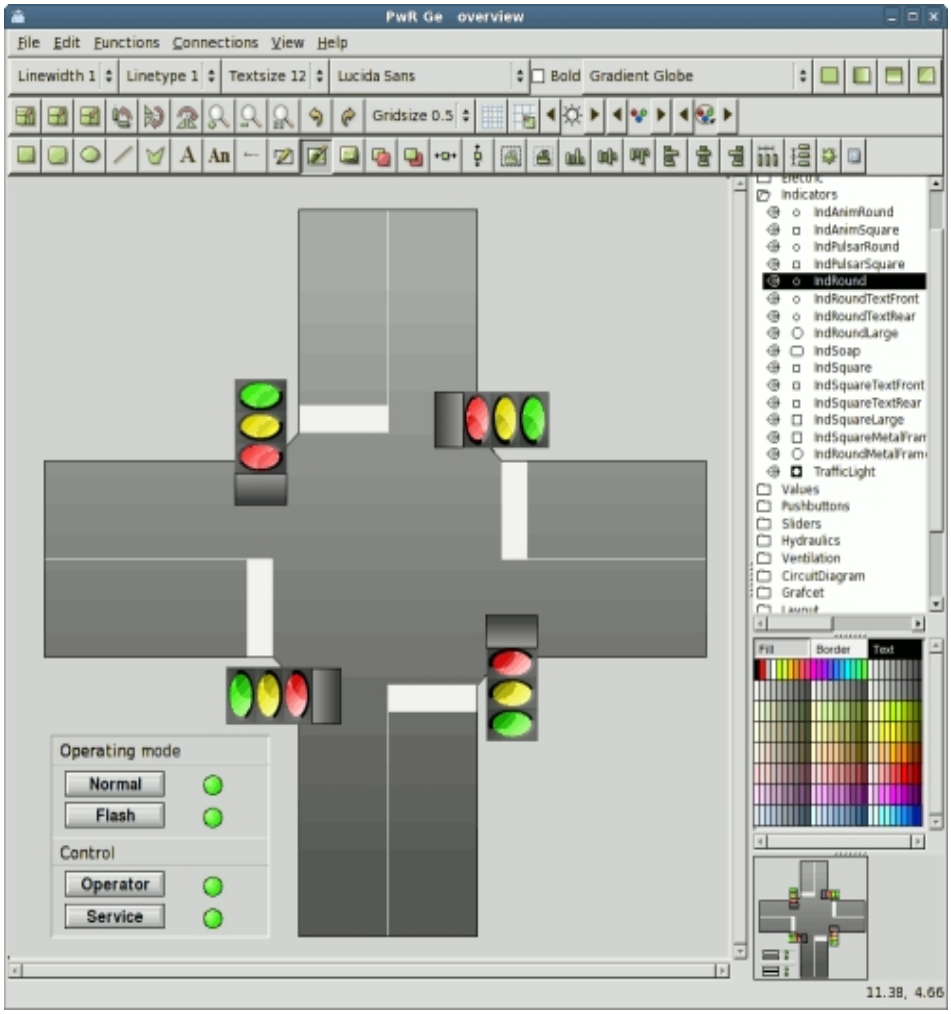


Fig L'editor di grafica per l'impianto

La grafica dell'impianto puo' contenere dinamiche, che sono collegate ai segnali logici, ad esempio :

- Testo che diventa visibile quando un segnale raggiunge un determinato valore
- Oggetti grafici che cambiano colore quando un segnale raggiunge un determinato valore
- Oggetti grafici che diventano invisibili quando un segnale raggiunge un determinato valore
- Oggetti grafici che si muovono in base al valore di un segnale

e' anche possibile inserire pulsanti nella grafica dell'impianto, tramite questi l'operatore puo' modificare i valori dei segnali digitali.
Per cambiare i segnali analogici si utilizza un campo di immissione.

Nel nostro esempio scegliamo di realizzare una grafica impianto, mostrando un incrocio stradale, dove i semafori (rosso, giallo e verde) sono dinamici come in figura.
Come creare la grafica dell'impianto e' descritto nel capitolo 5 Creazione di Plant Graphics.

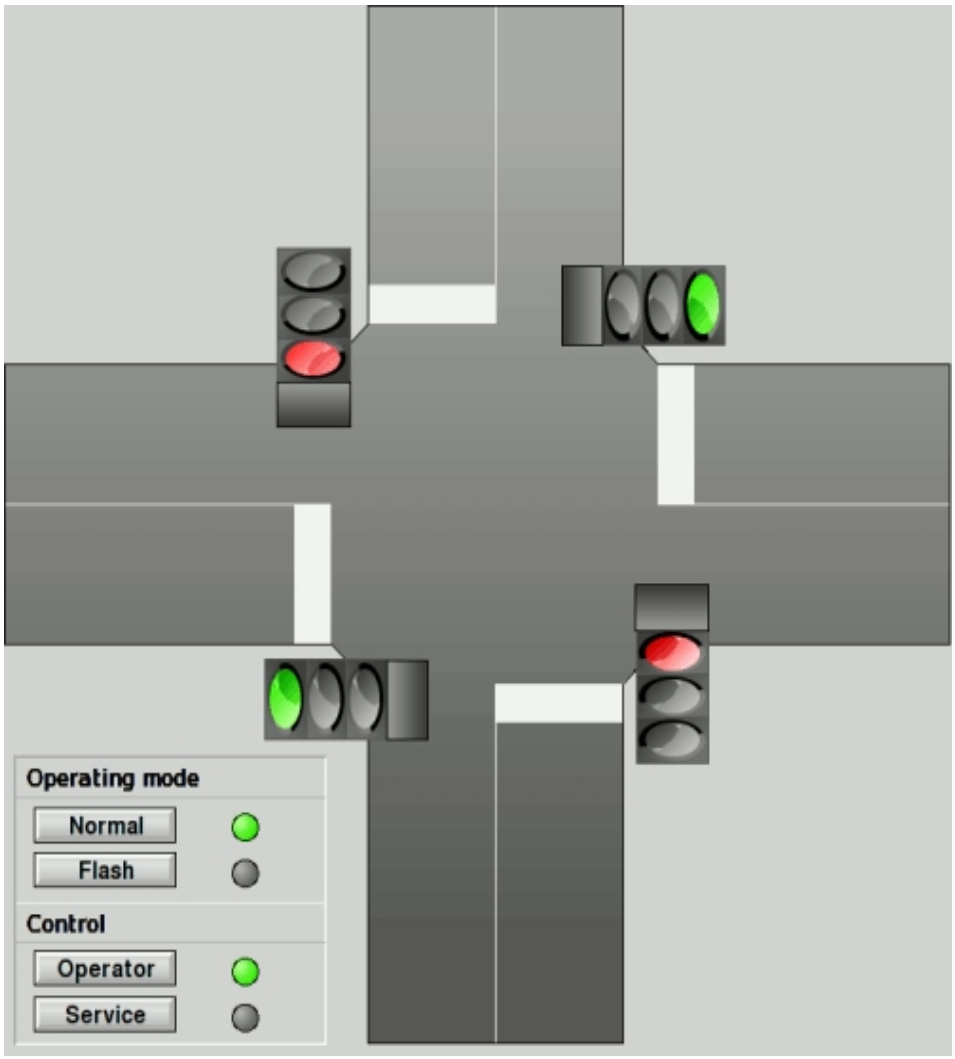


Fig Grafica impianto per l'incrocio

6 Crea un progetto

Installazione dell'ambiente di sviluppo

Prima di poter iniziare a lavorare con Proview e' necessario installare l'ambiente di sviluppo Proview. Sono disponibili numerosi pacchetti per diverse distribuzioni Linux, e se non esiste un pacchetto per la distribuzione desiderata e' possibile anche scaricare i sorgenti di Proview e compilare l'applicazione.

Il pacchetto di installazione e' denominato con il numero di versione nel nome, ad es. pwr47. Cio' rende possibile l'installazione di piu' versioni affiancate, il che e' un vantaggio quando si hanno molti progetti in produzione sviluppati su versioni diverse.

Troverete ulteriori informazioni sull'installazione nella pagina di Download su www.proview.se.

script di installazione

All'installazione viene creato l'utente Linux 'pwrp' con password 'pwrp'. Effettuando il login come pwrp puoi avviare Proview facendo clic sull'icona Proview.

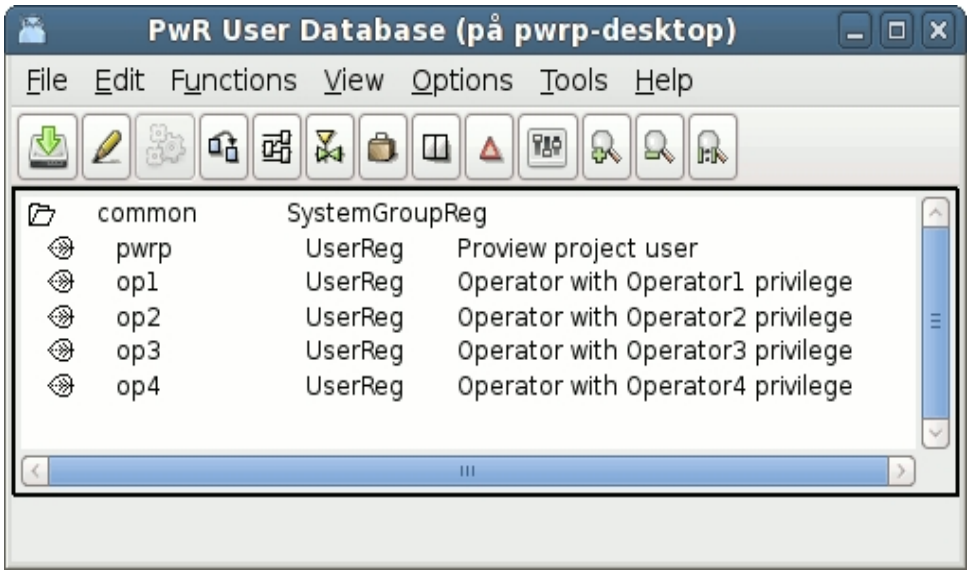
Se si desidera eseguire Proview come un altro utente, e' necessario avviare Proview all'accesso. Inserisci la seguente riga nel file .bashrc nella home directory

```
source /src/pwrp/adm/db/pwr_setup.sh
```

Se hai diversi utenti che lavora su progetti Proview in comune, devi assicurarti che abbiano accesso in scrittura ai file nei progetti. Un modo per ottenere cio' e' impostare umask su 002 e lasciare che tutti gli utenti abbiano lo stesso gruppo, ad es. pwrp.

utenti

Nel Caso Studiato prima, abbiamo imparato come creare utenti appartenenti a gruppi di sistemi diversi.
Questo e' descritto nel capitolo Amministrazione. All'istallazione del pacchetto di sviluppo, e' incluso un database utente contenente il gruppo di sistema 'commmon' con cinque utenti, pwrp, op1, op2, op3 e op4.
pwrp e' un utente di sviluppo e ha i privilegi di sistema in runtime. Gli utenti op sono operatori e hanno i privilegi di operatore in runtime. Questi utenti lavorano per molte applicazioni e per il momento ci accontenteremo di loro.



Utenti inclusi nell'installazione

Si noti che quando si compila Proview dai sorgenti, e' necessario creare gli utenti prima di creare un progetto.

Registra i volumi

Come abbiamo visto anche nel Caso di Studio, puoi registrare i volumi nel GlobalVolumeList. Questo puo' essere fatto in seguito, lasciando che la guida alla configurazione del progetto registri i volumi e recuperi la successiva identita' del volume libero.
Scegliamo in questa modalita' per il momento, ma se hai diverse stazioni di sviluppo indipendenti dovresti gestire la registrazione manualmente per evitare collisioni di identita' di volume. Se hai un impianto piu' grande con un sacco di volumi, ti consigliamo anche di assegnare manualmente le identita' del volume e raggrupparle in un modo adeguato.

Crea un progetto

Per creare un progetto, entra nell'elenco dei progetti dell'amministratore, facendo clic sull'icona Proview o tramite il comando

```
> pwra
```

pwra e' definito come 'wb -p pwrp pwrp', cioe' sta effettuando l'accesso come utente pwrp. Se hai definito altri utenti per lo sviluppo, devi ridefinire 'pwra' o usare direttamente il comando 'wb -p'.
wb accetta utente e password come argomenti.

Nell'elenco dei progetti si entra in modalita' modifica e si crea un oggetto ProjectReg al livello superiore o sotto un oggetto gerarchia. Nell'oggetto ProjectReg si specifica il nome del progetto, la versione Proview e il percorso per il progetto.
Quando si salva, vengono create le directory del progetto.

Come creare un progetto e' descritto nella Guida introduttiva. Si consiglia di passare attraverso le sezioni per creare il progetto e configurare il volume della directory prima di continuare.

7 Configurazione del volume della directory

Apri un progetto

Quando il progetto viene creato, si trova nella struttura del progetto amministratore.
Si apre un progetto attivando 'Open Project' nel menu popup per un oggetto ProjectReg.

Puoi anche usare il comando 'sdf' per attaccarsi ad un progetto.
sdf prende il nome del progetto come argomento

```
> sdf trafficcross1
```

Il volume della directory e' ora aperto dal comando

```
> pwrs
```

pwrs e' definito come 'wb pwrp pwrp', cioe' accede come utente pwrp.
Se hai definito altri utenti per lo sviluppo, devi ridefinire 'pwrs' o usare direttamente
il comando 'wb'.
wb accetta utente e password come argomenti (e anche volume come terzo argomento).

Se il volume e' vuoto, viene avviata una procedura guidata per aiutarci nella
configurazione. Per creare un progetto semplice con un nodo e un volume devi solo premere
il pulsante 'Avanti'.

La guida cerca i volumi registrati per il progetto. Se non ci sono volumi registrati
suggerisce volumi con nomi di volume adatti e identita' di volume libere e registra i volumi
se i suggerimenti sono approvati. La guida crea anche tutti gli oggetti di configurazione
nel volume della directory e inserisce dati appropriati in essi.

Se in seguito si espanderanno i sistemi con piu' nodi e volumi, e' utile avere una certa
conoscenza di come viene eseguita la configurazione, quindi descriviamo come configurare
manualmente il volume.

L'editor di configurazione

L'editor di configurazione mostra due finestre, e per DirectoryVolume, la sinistra mostra
la configurazione del volume e la destra la configurazione del nodo.

Configurare i Volumi

Per prima cosa configuriamo tutti i volumi root, sub-volumi e volumi di classe nel progetto.
Questo viene fatto nella finestra del volume nel volume della directory.
Iniziamo creando un oggetto RootVolumeConfig che configura un volume di root.

- Entra nella modalita' di modifica dal menu 'Edit/Edit mode'. Ora la tavolozza e' visibile a
  destra nella finestra e le mappe possono essere aperte con un clic sul simbolo della
  mappa o con un doppio clic sul testo.
- Apri la mappa del volume e seleziona la classe 'RootVolumeConfig'.
- Fare clic con MB2 nella finestra di configurazione del volume e l'oggetto viene creato.

- Seleziona l'oggetto e apri l'editor degli oggetti dal menu 'Functions/Open Object'.
- Seleziona ObjectName e attiva 'Functions/Change value' nel menu dell'editor degli oggetti.
- Inserisci il nome dell'oggetto. Il nome dovrebbe essere uguale al nome del volume.
- Chiudi l'editor degli oggetti.

Creare gli oggetti RootVolumeConfig per gli altri volumi root del progetto. Per i seguenti oggetti e' possibile controllare la posizione dell'oggetto. Se fai clic con MB2 sul nome di un oggetto, il nuovo oggetto sara' un fratello di questo oggetto.
Se fai clic sul simbolo della foglia o della mappa, l'oggetto sara' un bambino.

Anche i sottovolumi e volumiclasse sono configurati in modo simile con oggetti SubVolumeConfig e ClassVolumeConfig.

e' anche possibile visualizzare gli attributi di un oggetto direttamente nell'editor di configurazione:

- Premere SHIFT e fare clic MB1 sull'oggetto per aprire l'oggetto
- Selezionare un attributo e attivare Functions/Change per modificare un valore.

Configurare i nodi

Nella finestra di destra, i nodi del progetto vengono configurati. Raggruppate i nodi in base a quale bus QCOM comunicano. Creiamo due oggetti BusConfig, uno per i nodi di produzione ed uno per la simulazione. Nell'attributo BusNumber viene definito il numero di bus.

Come figli dell'oggetto BusConfig, vengono creati gli oggetti NodeConfig, uno per ogni processo e stazione operatore. Quando vengono creati gli oggetti NodeConfig, vengono anche creati alcuni oggetti aggiuntivi

- un oggetto RootVolumeLoad che indica il volume principale da caricare quando viene avviato l'ambiente di runtime su questo nodo. Il nome dell'oggetto dovrebbe essere uguale al nome del volume di root.
- un oggetto Distribute che configura quali file vengono copiati dall'ambiente di sviluppo al processo o alla stazione operatore.

Aprire l'oggetto NodeConfig e immettere nomenodo, sistema operativo e indirizzo IP.

Sotto l'oggetto BusConfig per il bus di simulazione e' opportuno posizionare un oggetto NodeConfig per la stazione di sviluppo, e al di sotto di questo, un RootVolumeLoad che indica il volume della stazione di processo con cui si lavorera' per primo. In questo modo e' possibile avviare il volume in runtime e testarlo nell'ambiente di sviluppo.
Indicare il nome, il sistema operativo e l'indirizzo IP della stazione di sviluppo nell'oggetto NodeConfig.

Oggetto di sistema
Crea anche un oggetto $System nella finestra di configurazione del nodo. L'oggetto di sistema ha gli attributi SystemName e SystemGroup.

- Il nome del sistema in questo stato e' spesso uguale al nome del progetto.
- L'attributo del gruppo di sistema rende il sistema membro di un gruppo di sistema nel database utente, che definisce gli utenti per il sistema. Una volta creato l'oggetto di sistema, e' necessario indicare un nome utente e una password validi quando si accede al workbench.

Salvataggio

Salva la sessione dal menu 'File/Salva'. Se la configurazione supera il controllo della

sintassi, ci verra' chiesto se si desidera creare i volumi configurati.
Rispondiamo Ok a questa domande e creiamo i volumi.

Se ora si apre la finestra di selezione del volume, 'File / Apri' nel menu, vengono visualizzati tutti i volumi configurati. Il prossimo passo e' configurare un RootVolume.

8 Configura il volume principale

Root Volume

Un volume di root viene aperto dalla finestra di selezione del volume. Seleziona il volume e clicca sul pulsante Ok. Questo avviera' l'editor di configurazione per il volume di root. Per quanto riguarda DirectoryVolume e' diviso in due finestre, ma questa volta, la finestra di sinistra mostra la configurazione dell'impianto e la destra la configurazione del nodo.

Configurazione dell'impianto

La configurazione dell'impianto descrive le diverse piante che puoi trovare in un sistema Proview. Una pianta e' una descrizione logica di ad es. un processo di produzione, funzioni, attrezzature, che deve essere controllato, supervisionato, ecc.
Vedi un esempio di configurazione di un impianto

Oggetto $PlantHier

L'oggetto principale nella gerarchia dell'impianto e' l'oggetto $ PlantHier.
Questo oggetto identifica la pianta o parti di essa.

L'oggetto $PlantHier viene utilizzato per raggruppare oggetti e per strutturare la pianta. Questo oggetto puo', ad esempio, essere utilizzato per raggruppare oggetti segnale.

Oggetti Signal

Gli oggetti segnale definiscono segnali logici, o punti, che rappresentano una quantita' o un valore da qualche parte nel processo; in contrasto con gli oggetti del canale che definiscono i segnali fisici. Gli oggetti di segnale sono generici, cioe' possono essere usati con qualsiasi sistema I / O.

Esistono alcune classi di segnali che non possono essere collegati al segnale hardware, ovvero gli oggetti Dv, Iv, Av e Sv (DigitalValue, IntegerValue, AnalogValue e StringValue). Questi oggetti vengono utilizzati per memorizzare rispettivamente valori logici, valori interi, numeri reali e stringhe.

Il valore effettivo del segnale e' definito dall'attributo ActualValue.

Attualmente sono disponibili i seguenti oggetti di segnale:

Ai	Input Analogico.
Ao	Output Analogico.
Av	Valore Analogico.
Ii	Input Intero.
Io	Output Intero.
Iv	Valore Intero.
Di	Input digitale.
Do	Output digitale.
Po	Output impulsivo digitale.
Dv	Valore Digitale.
Co	Counter Input COntatore.

Sv Valore Stringa.

Nota! Il programma PLC puo' leggere segnali posti su nodi remoti, ma non puo' scrivere su di essi.

Oggetto PlcPgm

L'oggetto PlcPgm definisce un programma PLC. E' possibile avere diversi programmi PLC in un impianto. Il seguente attributo deve avere un valore:

- Il ThreadObject indica il thread plc in cui viene eseguito il programma. Fa riferimento a un oggetto PlcThread nella configurazione del nodo.
- Se il programma contiene una sequenza Grafcet, e' necessario fornire ResetObject. Questo potra' essere un Dv, Di o Do che riporta la sequenza al suo stato iniziale.

Oggetto Backup

L'oggetto Backup viene utilizzato per indicare l'oggetto o l'attributo, per il quale verra' eseguito il backup. Viene anche indicato se la memorizzazione avverra' con tempo di ciclo veloce o lento.

MountObject

MountObject monta un oggetto in un altro volume. L'attributo Object specifica l'oggetto montato.

Configurazione Nodo

Il configuratore del nodo definisce i nodi del sistema PROVIEW/R. Viene dato un nome ai nodi e specificato il loro contenuto.

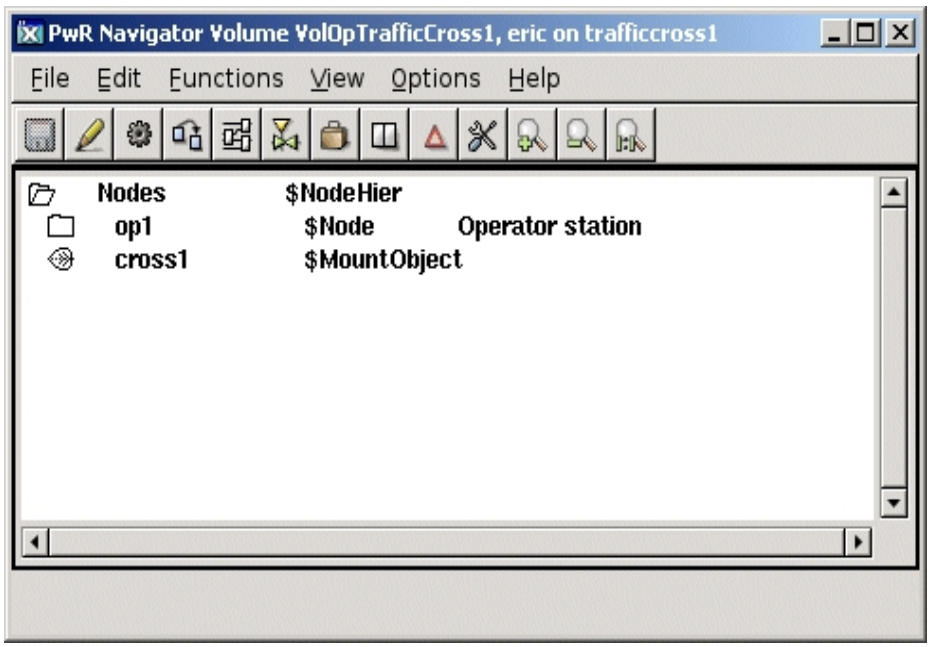


Fig Configuratore del Nodo

Oggetto $NodeHier

L'oggetto gerarchia dei nodi viene utilizzato per raggruppare oggetti nella configurazione del nodo. Questo oggetto e' della classe $NodeHier. L'oggetto puo' essere usato per raggruppare oggetti $Node o oggetti XttGraph.
Vedi $NodeHier nel Manuale di riferimento dell'oggetto

$Node

Per definire i nodi del sistema, si utilizzano oggetti nodo. L'oggetto nodo e' della classe $Node.
Quando viene creato l'oggetto nodo, viene creato un numero di oggetti server e operatore.

Vedi $Node nel manuale di riferimento dell'oggetto

Ogetti I/O

La configurazione del sistema I/O dipende dal tipo di sistema I/O che si utilizza. Proview ha un I/O modulare in grado di gestire diversi tipi di sistemi I/O: sistemi rack e card, sistemi bus distribuiti o sistemi connessi con alcune reti.

L'I/O modulare e' suddiviso in quattro livelli: agente, rack, scheda e canale.

Rack e Card System
Prenderemo il PSS9000 come esempio di I / O di rack e schede. Il sistema e' costituito da schede di input e output analogici e digitali montati in rack. Il rack e' collegato tramite un cavo bus a una scheda busconverter nel computer che converte il bus PSS9000 nel bus PCI del computer.

In questo caso, il livello dell'agente non viene utilizzato, quindi l'oggetto $Node funziona come un agente. Il livello del rack e' configurato con oggetti SSAB_Rack posizionati sotto l'oggetto $Node, uno per ogni rack nel sistema. Le schede sono configurate con oggetti sotto l'oggetto rack, che sono specifici per diversi tipi di IO card. Per PSS9000 ci sono oggetti di schede come Ai_Ai32uP, Ao_Ao4uP, Di_DIX2 e Do_DVDO32. Sotto un oggetto scheda, vengono posizionati gli oggetti del canale, uno per ciascun canale sulla scheda.

Comuni per i diversi sistemi I / O sono gli oggetti canale, che definiscono i canali di input o output di una scheda o di un modulo. Esistono diversi tipi di canali.

ChanDi	Input Digitale
ChanDo	Output Digitale
ChanAi	Input Analogico
ChanAit	Ingresso analogico con conversione del valore del segnale da una tabella
ChanAo	Output Analogico
ChanIi	Input Integer
ChanIo	Output Integer
ChanCo	Input Contatore

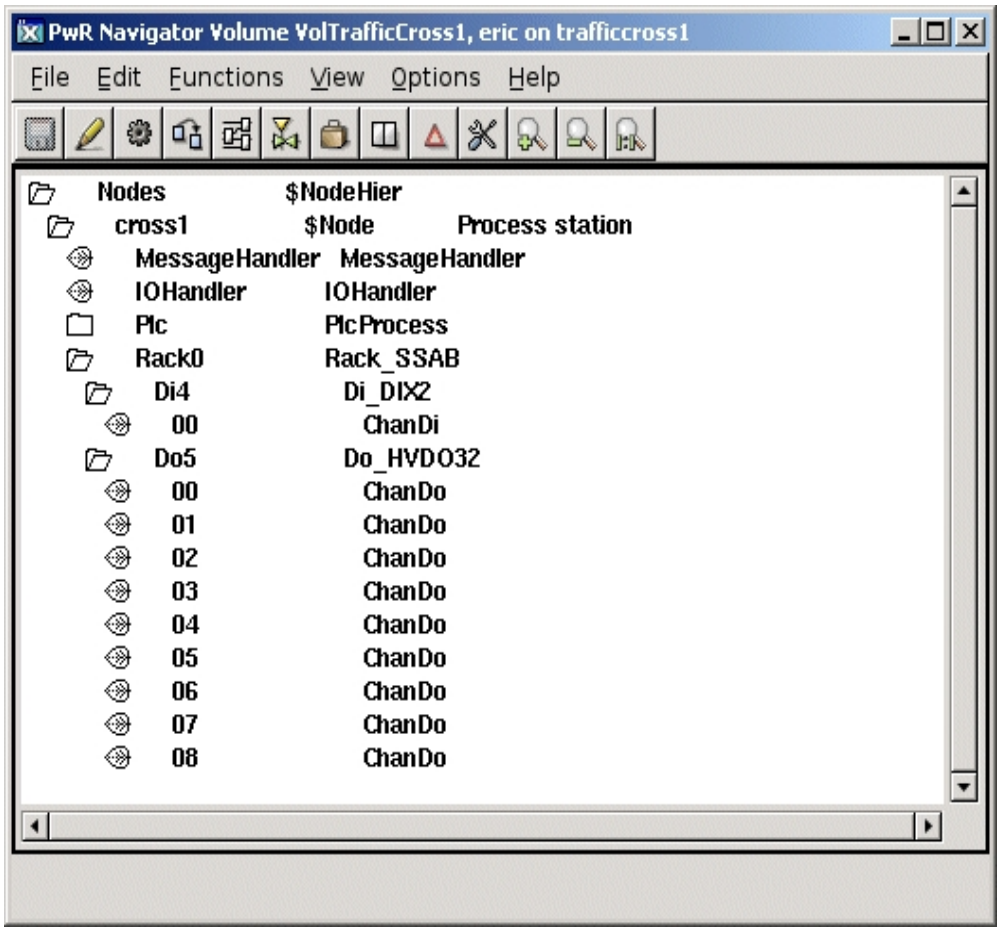


Fig Configurazione I/O

I/O Distribuiti

Come esempio di I/O distribuito scegliamo profibus. In questo caso, vengono utilizzati tutti e quattro i livelli.
Nel bus PCI del computer, c'e' una mastercard che comunica con un numero di slave sul circuito profibus. La mastercard e' configurata con una scheda Pb_Profiboard a livello di agente.
Al di sotto di questo, troviamo i diversi slave configurati con oggetti Pb_DP_Slave. Rappresentano il livello del rack. Sotto gli oggetti slave ci sono oggetti modulo di tipo Pb_Ai, Pb_Ao, Pb_Di, Pb_Do etc, che sono posti sul livello della scheda.
Sotto gli oggetti del modulo, infine, i canali sono configurati con gli oggetti canale ChanDi, ChanDo ecc.

Processo e thread per oggetti I/O

Gli oggetti I/O a livello scheda, spesso contengono gli attributi Process e ThreadObject. Quale processo gestira' la scheda viene definito nell'attributo Process.

La scheda puo' essere gestita dal programma PLC, cioe' lettura e scrittura sono sincronizzate con l'esecuzione del PLC. E' inoltre possibile specificare un thread nel PLC che deve gestire la scheda, vale a dire quale base dei tempi viene utilizzata per leggere o scrivere la scheda (l'attributo PlcThread).

La scheda puo' anche essere gestita dal processo rt_io, che di solito ha una priorita' inferiore rispetto al PLC e che non e' sincronizzato con il PLC. Alcuni tipi di inputcards analogici che richiedono un po' di tempo per la lettura sono gestite in modo vantaggioso da questo processo.

Puoi anche scrivere un'applicazione che gestisca la lettura e la scrittura delle schede. C'e' un'API per inizializzare, leggere e scrivere le schede. Cio' e' utile se la lettura e la

scrittura di una scheda deve essere sincronizzata con l'applicazione.

Oggetto MessageHandler

L'oggetto MessageHandler configura il processo server rt_emon, che gestisce gli oggetti di supervisione (DSup, ASup, CycleSup). Quando viene rilevato un evento dal server, viene inviato un messaggio alle unita' esterne che hanno interessi in questo specifico evento. Nell'oggetto e' indicato ad esempio il numero di eventi che sono memorizzati nel nodo. L'oggetto viene creato automaticamente sotto un oggetto $Node.
Vedere MessageHandler nel Manuale di riferimento dell'oggetto

Oggetto IOHandler

IOHandler configura le proprieta' per la gestione I/O.

- ReadWriteFlag specifica se indirizzare l'hardware fisico o meno.
- IOSimulFlag indica se utilizzare l'hardware o meno.
- La base dei tempi per il processo rt_io, ovvero il processo che gestisce i tipi piu' lenti di schede I/O che non sono adatte ad essere gestite dal PLC.

Nel sistema di produzione per una stazione di processo, ReadWriteFlag e' impostato su 1 e IOSimulFlag e' impostato su 0. Se si desidera simulare la stazione di processo, ad esempio sulla stazione di sviluppo, ReadWriteFlag e' impostato su 0 e IOSimulFlag e' impostato su 1.

L'oggetto IOHandler viene creato automaticamente, quando si crea un oggetto $Node.
Vedi IOHandler nel Manuale di riferimento dell'oggetto

Oggetto Backup_Conf - Oggetto di configurazione per il backup

A volte puo' essere preferibile avere un backup di un numero di oggetti nel sistema. In tal caso, posizionare un oggetto di configurazione di backup, Backup_Conf, sotto l'oggetto nodo. Il backup viene eseguito con due cicli diversi, uno veloce e uno lento.

Per indicare di quali oggetti/attributi eseguire il backup, utilizzare gli oggetti di backup.
Vedi la descrizione dell'oggetto Backup
Vedere Backup_Conf nel Manuale di riferimento dell'oggetto

Oggetto Operator Place

Per definire un posto operatore posizionare un oggetto della classe OpPlace sotto l'oggetto $Node.

I seguenti attributi devono essere compilati:

- UserName e' un nome di un utente Proview, definito nel UserDatabase. I privilegi dell'utente determinano l'accesso al sistema.
- MaxNoOfEvents indica il numero di eventi che l'elenco eventi dell'operatore puo' contenere contemporaneamente.
- EventSelectList indica le gerarchie degli oggetti nella Configurazione impianto, da cui l'operatore ricevera' gli eventi.

Se osserviamo la figura sottostante, che illustra l'impianto A, e assumiamo che vogliamo ricevere eventi solo dagli oggetti cerchiati, dichiariamo "A-C" come alternativa nell'elenco di selezione. Questa scelta significa che riceveremo eventi dall'oggetto supervisionato C e da tutti gli oggetti supervisionati che hanno C come loro padre.

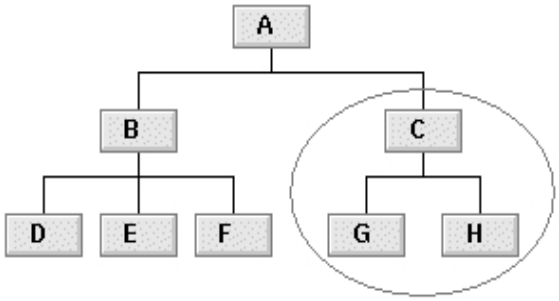


Fig SelectList esempio 1

Un altro esempio:

Guardiamo la figura sottostante, che illustra l'impianto TrafficCross1. Se desideri ricevere tutti gli eventi dall'impianto TrafficCross1, dichiari TrafficCross1 come alternativa.
TrafficCross1 gestisce due semafori, TrafficLightNS e TrafficLightWE. Diciamo che vogliamo eventi solo da TrafficLightNS. In tal caso, viene indicato "TrafficCross1-TrafficLightNS" anziche' TrafficCross1.

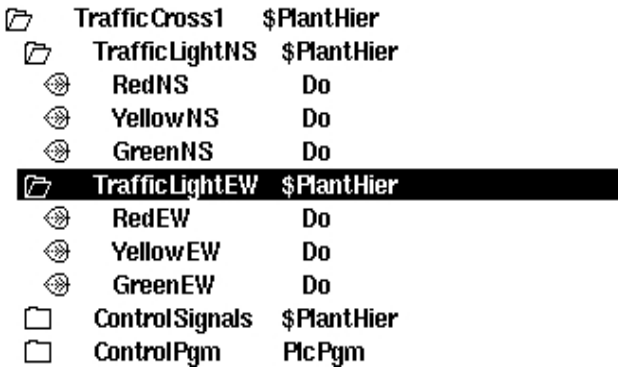


Fig EventSelectlist esempio 2

Se si desidera ricevere messaggi dall'oggetto CycleSup che supervisiona i thread plc, e' necessario specificare anche il nome della gerarchia dell'oggetto $Node.

In FastAvail si specifica il nome completo della gerarchia dell'oggetto XttGraph, che sara' possibile avviare dai pulsanti grafici della finestra operatore.
Puoi avere 0 - 25 pulsanti. I pulsanti non utilizzati diventano invisibili.

Vedi OpPlace nel Manuale di riferimento dell'oggetto

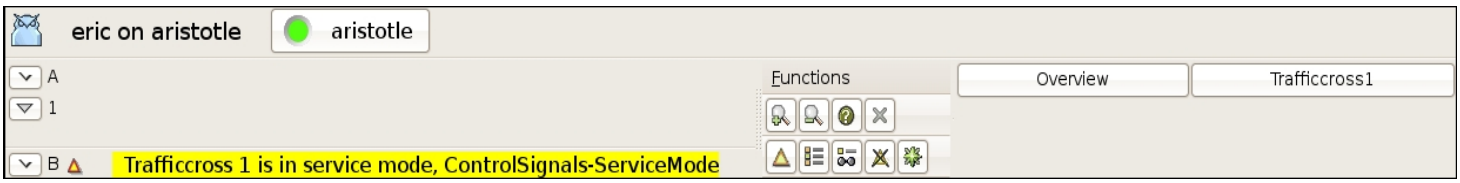


Fig Finestra dell'operatore

Gli oggetti grafici dell'impianto: l'oggetto XttGraph

Per poter mostrare la grafica dell'impianto che e' unica al progetto nella stazione operatore, e' necessario configurare gli oggetti XttGraph. Questi oggetti definiscono, ad esempio, i nomi dei file con la grafica dell'impianto. Gli oggetti sono indicati nell'attributo FastAvail dell'oggetto OpPlace e nell'attributo DefGraph che si trova in

$PlantHier e negli oggetti segnale.

Quando si fa riferimento all'oggetto in FastAvail, e' possibile utilizzare la possibilita' di eseguire un comando Xtt dall'oggetto XttGraph. In questo modo, e' possibile impostare un segnale da un pulsante nella finestra dell'operatore.

- Azione. Dichiara un grafico Ge da aprire o un comando Xtt da eseguire.

Vedi XttGraph nel Manuale di riferimento dell'oggetto

Oggetto Multiview

Multiview e' una finestra operatore organizzata come tabella in cui ogni cella puo' contenere un grafico, un trend, una curva storica, una lista di allarmi, una lista di eventi o un'altra finestra multview.
L'esempio seguente mostra una finestra multiview con una colonna e due righe. La prima cella contiene un'altra vista multipla con due colonne e tre righe e la seconda cella contiene una finestra di allarme. Le diverse finestre di allarme mostrano allarmi provenienti da diverse parti dell'impianto, specificati dagli oggetti AlarmView.
Una finestra multiview e' configurata da un oggetto XttMultiView. La matrice Action contiene le specifiche per ogni cella.

e' anche possibile scambiare un grafico o una curva in una cella con il comando 'set subwindow'.

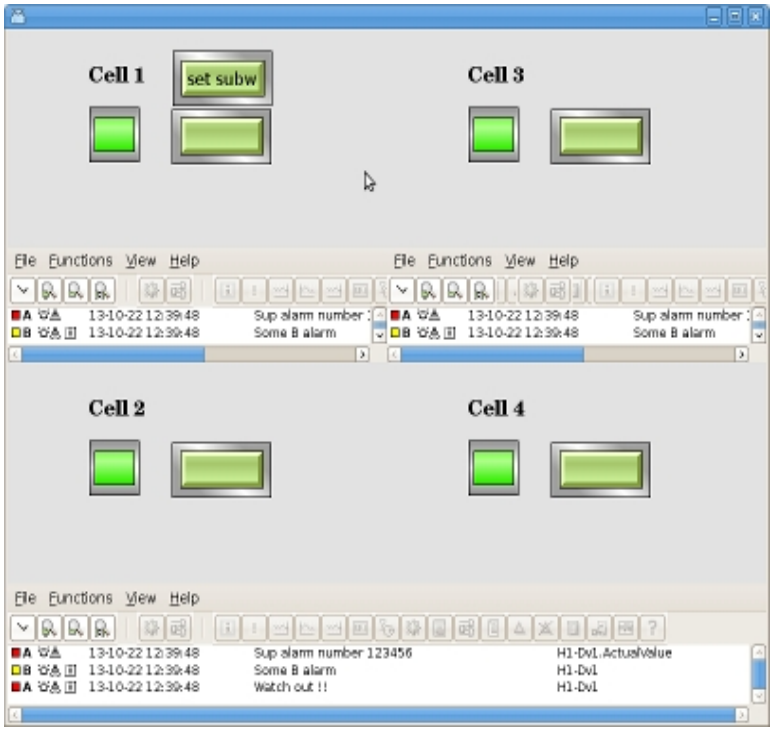


Fig Finestra Multiview

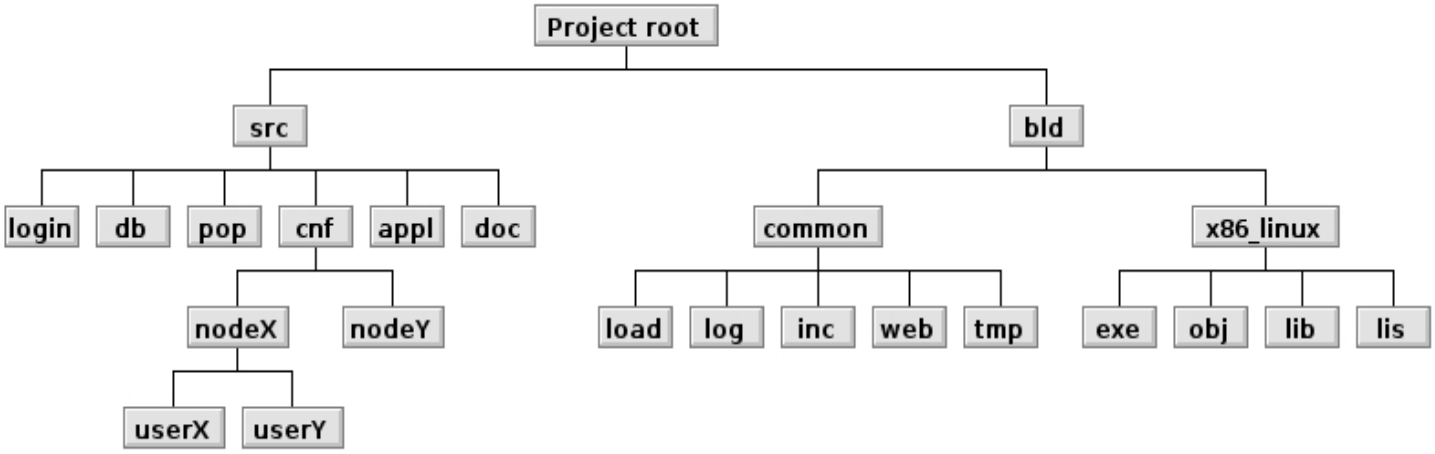
9 Navigazione del progetto

Questo capitolo descrive la struttura dei progetti e spiega lo scopo delle diverse directory. Il capitolo descrive anche alcuni file importanti che e' possibile utilizzare per configurare e controllare diverse cose nel progetto.

La struttura delle directory e' divisa in due rami, uno che contiene tutti i file sorgente del progetto e l'altro e' l'albero di compilazione in cui sono posizionati tutti i file prodotti.

9.1 Introduzione

La figura seguente mostra una struttura dell'albero delle directory di un progetto .



Questa struttura di directory e' stata modificata in Proview V4.7. Il motivo di cio' e' stata quella di rendere piu' chiaro quali siano le fonti e i file di configurazione del progetto e quali siano i contenuti generati.
Le versioni precedenti non lo rendevano chiaro e l'idea e' che tutto cio' che risiede nell'albero di compilazione puo' essere rigenerato dall'albero dei sorgenti.

Tutte le directory hanno una variabile di ambiente definita, quindi sara' facile raggiungere tutte le directory nel progetto. Queste variabili sono sempre definite come :

```
$pwrp_<directory>
```

Per esempio $pwrp_exe per la dire directory <project_root>/bld/x86_linux/exe.

9.2 L'albero dei sorgenti

L'albero dei sorgenti contiene tutti i file che sono i sorgenti e i file di configurazione dei progetti. Il livello superiore $pwrp_src contiene solo sottodirectory e nessun file sorgente o di configurazione. Di seguito sono descritti i contenuti e lo scopo delle sottodirectory.

9.2.1 $pwrp_login

Questa e' la directory in cui ti troverai quando ti sposti a un progetto con il comando 'sdf'.
Due file qui sono di interesse.

```
login.sh
sysinfo.txt
```

login.sh e' uno script che viene eseguito quando si accede al progetto. Puo' essere utilizzato per impostare variabili di ambiente specifiche del progetto e simili.

sysinfo.txt e' un file di testo che verra' stampato nella finestra del terminale quando si arriva al progetto. Informazioni su cosa sta succedendo o che cosa e' stato fatto nel progetto possono essere inserite qui.

9.2.2 $pwrp_db

Questa e' la directory in cui risiedono i database di tutti i volumi locali (incluso il volume di directory in cui e' configurato il progetto). Ogni database risiede nella sua sottodirectory. Questo e' valido se si sceglie di creare i propri database come database BerkleyDB. Se invece si sceglie di avere i database mysql, i database verranno creati sul server mysql.

In questa directory risiedono anche i file per le classi definite dall'utente, UserClassVolumes. Sono file di testo con wb_load file-end. Il volume della classe utente di solito ha un nome simile a RootVolume nel progetto che utilizza le classi. Se il tuo RootVolume e' denominato VolMyProject, ClassVolume si chiamera' CVolMyProject e il nome del file quindi sara':

```
cvolmyproject.wb_load
```

9.2.3 $pwrp_pop

Questa e' la directory in cui verranno archiviate le immagini sviluppate con il ge-editor (fileend * .pwg). I file completati devono essere copiati nella directory exe nella struttura ad albero ($ pwrp_exe). Anche gli xtt-helpfiles dovrebbero essere sviluppati qui e copiati in $Pwrp_exe-directory.
Le immagini che hanno un oggetto XttGraph corrispondente nella gerarchia dei nodi di un oggetto RootVolume verranno automaticamente copiate nella directory $ pwrp_exe quando viene creato quel volume.

9.2.4 $pwrp_appl

Questa e' la directory in cui e' necessario conservare i codici sorgente per le proprie applicazioni appartenenti al progetto e anche per i codici che si desidera collegare al programma plc.

Un file di interesse speciale che dovrebbe essere tenuto qui e' il
`ra_plc_user.h-file`
Questo file verra' incluso di default quando si compila il codice plc. E' tuttavia incluso da una directory nella struttura ad albero (<project_root>/bld/common/inc).

Si noti che ra_plc_user.h viene generato automaticamente su $ pwrp_inc quando viene compilato il primo PlcPgm. Se hai bisogno di modificare ra_plc_user.h, dovresti copiarlo in $ pwrp_applcode e conservare l'originale li'.

Tutti i file header situati in $ pwrp_appl o nelle sottodirectory e che dovrebbero essere inclusi con il programma plc devono essere copiati nella directory $pwrp_inc-directory (<project_root>/bld/common/inc).

Se si dispone di alcune funzioni di piccole dimensioni che si collegano solo con il programma plc e nient'altro, in genere si inserisce questo codice in un file chiamato:

```
ra_plc_user.c
```

9.2.5 $pwrp_doc

In questo direttorio viene messa la documentazione relativa al progetto. Questa puo' essere la tua documentazione impianto o, ad esempio, il DataSheet dei componenti esistenti nel tuo impianto che potresti voler distribuire alle stazioni operatore.

9.2.6 $pwrp_cnf

Questa directory contiene tutti i file di configurazione del tuo progetto. Alcuni file di configurazione sono comuni per l'intero progetto e sono posizionati qui. Altri file di configurazione sono specifici per ciascun nodo del progetto. Crea una sottodirectory qui per ogni nodo che ha specifici file di configurazione. A volte una configurazione e' unica per un utente specifico. In tal caso, creare una sottodirectory nella directory del nodo per quell'utente.

I file che dovresti tenere qui sono:

Configurazione dei tasti funzione globali.
`Rt_xtt`

Configurazione di menu e comandi rapidi in rt_xtt.
`xtt_setup.rtt_com`

File di avvio per Proview.
`ld_appl_<nodename>_<bus_no>.txt`

File per decidere quali librerie collegare con il programma plc.
`plc_<nodename>_<bus_no>_<plc_name>.opt`

File per controllare l'impostazione dei valori iniziali all'avvio di Proview.
`pwrp_alias.dat`

Tutti i file di cui sopra sono ulteriormente descritti di seguito.

9.3 L'albero di build

L'albero di build contiene tutti i file necessari durante la creazione del programma plc. Contiene anche tutti i file prodotti durante la compilazione.
I file necessari per la creazione del programma plc dovrebbero essere in questo albero, ma il master per tutti i file dovrebbe essere nell'albero dei sorgenti.
L'idea di base e' che dovrebbe essere possibile rimuovere completamente l'albero di build e rigenerarlo dall'albero dei sorgenti.

9.3.1 $pwrp_exe

Questa e' la directory in cui vengono creati i file exe per i programmi plc. Il programma plc e' chiamato:

```
plc_<nodename>_<bus_no>_<version_no>
```

Se hai delle tue applicazioni integrate nel progetto, anche questi file exe dovrebbero essere generati qui.

9.3.2 $pwrp_obj

Questa e' la directory in cui devono essere collocati tutti i file oggetto prodotti durante la compilazione.
I file oggetto per il programma plc vengono posizionati automaticamente qui.
Il file oggetto per un oggetto PlcPgm inserito in PlantHier prende il nome dall'identita' dell'oggetto di questo oggetto.

Anche i file oggetto per tutti gli altri codici dovrebbero essere posizionati qui.

9.3.3 $pwrp_lis

Questa e' la directory in cui si collocano i file di elenco prodotti durante la compilazione.

9.3.4 $pwrp_lib

Questa e' la directory in cui verranno create le librerie contenenti i file oggetto appartenenti a un determinato volume. Dovresti anche posizionare le tue librerie qui.

9.3.5 $pwrp_load

Questo e' il posto dove verranno creati i file di caricamento per i volumi dei progetti.
I file di caricamento sono denominati:

```
<volumename>.dbs
```

9.3.6 $pwrp_log

Questa directory contiene i file di registro prodotti durante la simulazione del progetto.
Il file di registro principale di Proview e' denominato

```
pwr_<nodename>.log
```

Se si riavvia la simulazione, la registrazione verra' aggiunta a questo file. Rimuoverlo se si vuole un file nuovo.

9.3.7 $pwrp_tmp

Questa directory contiene i file temporanei. Questi file verranno creati durante determinate operazioni.
Ad esempio, se si compila un programma plc in modalita' debug, qui verranno creati i file sorgente creati per questo programma.

9.3.8 $pwrp_inc

Questa directory contiene i file-include che verranno inclusi durante la creazione del programma plc.
Un file chiamato
`ra_plc_user.h`
sara' sempre cercato in questa directory.
Se hai altri file-include da includere, quindi includili in questo.
Il master per tutti questi file-include deve essere conservato nell'albero dei sorgenti e copiato qui.

Gli Header-files per le userclasse vengono creati qui quando si crea un classvolume. Se hai documentato le tue classi, verra' creato anche un file di aiuto. I file sono nominati:

```
pwr_<classvoumename>classes.h
pwr_<classvoumename>classes.hpp
<classvoumename>_xtthelp.dat
<classvolumename>.html
```

9.3.9 $pwrp_web

Questa directory contiene tutti i file per l'interfaccia web. Ad esempio xtt-helpfiles che generi come file html. Inoltre, se crei immagini java dai tuoi grafici ge, verranno creati qui.

9.4 File speciali

Tutti i file speciali che possono essere usati per diversi tipi di configurazione, o che sono di interesse diverso e si trovano da qualche parte nell'albero della directory del progetto, sono descritti qui. Sono tutti citati sopra in questo capitolo.

9.4.1 Rt_xtt

Questo file viene letto da rt_xtt quando viene avviato e il file viene cercato dalla directory in cui si avvia rt_xtt. Il file configura i tasti di scelta rapida per eseguire diversi tipi di comandi.

I comandi validi sono:

```
Command     // Questo eseguira' un comando xtt
SetDig      // Questo impostera' un segnale digitale su TRUE
ToggleDig   // Questo commutera' lo stato di un segnale digitale
ResetDig    // Questo resettera' un segnale digitale a FALSE
```

Per associare un tasto di scelta rapida a un comando, devi prima definire la chiave e quindi indicare il comando.
Ad esempio, per associare il tasto <ctrl> F5 a un comando che riconosce un allarme di tipo A:

```
Control <Key>F5: Command(event ack /prio=A)
```

Un tipico file Rt_xtt potrebbe essere simile a questo:

```
#
# Function key definition file
#
Control <Key>F5: Command(event ack /prio=A)
Control <Key>F6: Command(event ack /prio=NOA) # ack non A-alarms
Control <Key>F7: Command(show alarm)          # open alarm list
Control <Key>F8: Command(show event)          # open event list
# Below opens a graph defined by a XttGraph-object in the node hierarchy.
# The $Node-expression will be replaced by the node-object on this node.
# This makes the Rt_xtt-file work on different nodes.
Alt <Key>F12: Command(open graph /object=$Node-Pictures-rkt_overview)
# Below opens a graph defined by a XttGraph-object in the node hierarchy.
# The /focus-syntax sets focus on a object in the graph named NewPlate
Control <Key>F9: Command(open graph /object=*-Pictures-rkt_platepic/focus="NewPlate")
Control <Key>F10: Command(open graph /object=*-Bilder-rkt_cells/focus="Check_no")
# Below closes all open graphs except rkt_overview.
Control <Key>F11: Command(close all/except=rkt_overview)
Shift <Key>F1: SetDig(VWX-RKT-RB-DS-OnOffMan_1_2.ActualValue)
Shift <Key>F6: ResetDig(VWX-RKT-RI-RP-CalcPrePos.ActualValue)
Shift Control <Key>v: ToggleDig(VWX-RKT-COM-VWXSVR-BlockOrder_RKT.ActualValue)
```

9.4.2 xtt_setup.rtt_com

Questo file viene letto da rt_xtt quando viene avviato e il file viene cercato
dalla directory in cui si avvia rt_xtt. Il file configura l'aspetto di rt_xtt.
e' possibile creare i propri menu e creare voci che eseguono determinati comandi.

Tutti i comandi nel file seguono la sintassi del comando standard xtt. C'e' una
guida integrata in rt_xtt che spiega la maggior parte dei comandi.
Per visualizzare questo tipo di aiuto dare <ctrl>-b quando si e' in xtt apre la riga di
comando e qui scriviamo help. Navighiamo attraverso i comandi per capirli.

Quando si avvia rt_xtt, per impostazione predefinita sono disponibili alcuni menu in cui
il primo e' denominato Database.
Se, ad esempio, ti piacerebbe creare un menu in alto (prima), usa questo comando:

```
create item/text="Maintenance"/menu/dest=DataBase/before
```

Per creare un sottomenu (primo figlio) a questo, usa questo comando:

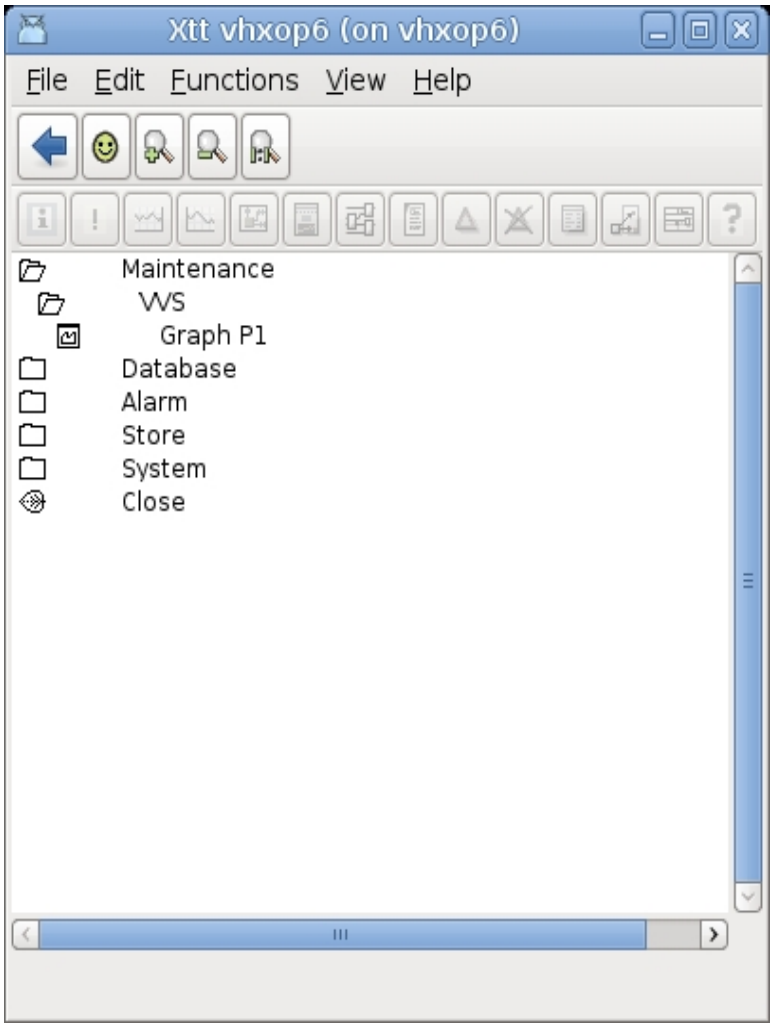
```
create item/text="VVS"/menu/dest=Maintenance/firstchild
```

Per creare una voce che esegue un comando sotto il menu VVS, utilizzare questo comando:

```
create item/text="Graph P1"/command="open graph/object=$Node-pics-h4_proc1"\
       /pixmap=graph/dest=Maintenance-VVS/lastchild
```

Il qualificatore pixmap definisce l'aspetto (icona) di questa voce. Senza questa qualifica
l'icona sara' una foglia. Il comando apre un grafico definito da un oggetto XttGraph nella
gerarchia dei nodi.

Il risultato sara' simile a questo:



Oltre a creare un menu puoi anche definire simboli (scorciatoie) che possono essere usati come comandi.
I simboli possono essere inseriti sulla riga di comando e il comando definito dal simbolo verra' eseguito.
Quanto segue definisce un simbolo 'h4' che aprira' un grafico:

```
define h4 "open graph /object=*-pics-h4_process1"
```

Nel file xtt_setup si crea un comando per riga. Le righe di commento iniziano con un punto esclamativo.

9.4.3 ld_appl_<node>_<bus_no>.txt

Questo file controlla quali applicazioni devono essere avviate quando si avvia il runtime di Proview. E' possibile aggiungere le proprie applicazioni e disattivare una o piu' delle applicazioni del kernel Proview.

Un tipico file ld_appl puo' avere questo aspetto:

```
# Startup file for PROVIEW/R
#
# id,   name,   load/noload run/norun,  file,   prio,   debug/nodebug,  "arg"
#pwr_neth,      , noload, norun, , 5, debug, ""
#pwr_plc,       , noload, norun, ,  , debug, ""
```

```
#pwr_alim,      , noload, norun, , 5, debug, ""
#pwr_emon,      , noload, norun, , 5, nodebug, ""
#pwr_tmon,      , noload, norun, , 5, debug, ""
#pwr_qmon,      , noload, norun, , 19, debug, ""
#pwr_nacp,      , noload, norun, , 5, debug, ""
#pwr_bck,       , noload, norun, , 5, debug, ""
#pwr_io,         , noload, norun, , 5, debug, ""
#pwr_linksup,   , noload, norun, , 5, debug, ""
#pwr_trend,     , noload, norun, , 5, debug, ""
#pwr_fast,       , noload, norun, , 5, debug, ""
#pwr_remh,       , noload, norun, , 5, debug, ""
pwr_remlog,       , noload, norun, , 5, debug, ""
#pwr_sysmon,       , noload, norun, , 5, debug, ""
#pwr_elog,       , noload, norun, , 5, debug, ""
pwr_webmon,       , noload, norun, , 5, debug, ""
pwr_webmonmh,       , noload, norun, , 5, debug, ""
pwr_webmonelog,       , noload, norun, , 5, debug, ""
#pwr_opc_server,       , noload, norun, , 5, debug, ""
#pwr_sevhistmon,       , noload, norun, , 5, debug, ""
#pwr_sev_server,       , noload, norun, , 5, debug, ""
#rs_nmps_bck, rs_nmps_bck, noload, run, rs_nmps_bck, 12, nodebug, ""
ra_utl_track, ra_utl_track, noload, run, ra_utl_track, 12, nodebug, ""
```

Il simbolo cancelletto (sharp) significa che le linee in cui compare sono commentate e quindi non eseguite. Quasi tutte le applicazioni del kernel Proview sono commentate finche' non si vorra' avviarle. Se si toglie il segno di hash, questa applicazione del kernel non verra' avviata. Per esempio in questo caso non ho un'interfaccia web, quindi non voglio che le applicazioni web inizino (pwr_webmon).

Se utilizzo Nmps-cells e voglio che il contenuto delle celle venga sottoposto a backup, tolgo il segno calcelletto su rs_nmps_bck.

Nell'ultima riga ho aggiunto un'applicazione prodotta da me stesso. Ho scelto la priorita' 12.
Non voglio che questa applicazione interferisca con le applicazioni del kernel che funzionano tra 17 e 19.

9.4.4 plc_<node>_<bus_no>_<plc_name>.opt

Nota! Questo file e' deprecato da V4.8.2. Al suo posto dovrebbe essere usato un oggetto BuildOptions.

Questo file (se esiste) verra' utilizzato come opzioni di collegamento quando creero' il programma plc.
Proview per collegamenti predefiniti rispetto ad alcune librerie e file di oggetti.
Se hai il tuo opt-file devi includerli. Un file di opt predefinito sara' simile a:

```
$pwr_obj/rt_io_user.o -lpwr_rt -lpwr_usbio_dummy -lpwr_usb_dummy -lpwr_pnak_dummy
 -lpwr_cifx_dummy -lpwr_nodave_dummy -lpwr_epl_dummy
```

Aggiungi le tue librerie a volonta'. La sintassi da utilizzare e' la sintassi per ld (Il linker GNU). Proview creera' un file di opzioni template .opt_template che puo' essere rinominato in .opt e utilizzato come modello.

9.4.5 pwrp_alias.dat

File per controllare l'impostazione dei valori iniziali all'avvio di Proview.

Esistono diversi modi per impostare i valori attraverso il file pwrp_alias.
Lo stesso file e' utilizzato per tutti i nodi nel progetto. Ogni riga nel file dovrebbe iniziare con la seguente espressione:

```
<nodename>_setval
```

I diversi modi di impostare le cose sono descritti di seguito:

1. Impostazione di un valore di attributo

```
<nodename>_setval <attribute_name> = <value>
```

Esempio:

```
bslds1_setval bsl-ds1-par-maxtemp.actualvalue = 70.0
```

L'uso della sintassi sopra descritta impostera' il valore prima che il backup sia caricato e prima che il programma plc sia avviato. Cio' significa che se viene eseguito il backup di un valore, il valore di backup sara' sempre valido.

Se invece vuoi davvero che l'impostazione in questo file abbia effetto, usa questa sintassi:

```
<nodename>_setvalp <attribute_name> = <value>
```

In questo caso, l'impostazione avra' effetto dopo il caricamento del file di backup e dell'avvio del programma plc.

2. Impostazione della modalita' di simulazione

L'impostazione della modalita' di simulazione significa che non verra' gestito alcun I/O fisico. E' possibile scrivere programmi di simulazione per impostare i valori corretti sull'ingresso I / O.
Aggiungi questa linea al file:

```
<nodename>_setval plcsim = yes
```

3. Impostare tutti i programmi plc in scan-off all'avvio

Per impostare tutti i programmi plc su scan-off, utilizzare questa riga:

```
<nodename>_setval plcscan = off
```

Attivare un programma plc trovando il corrispondente oggetto WindowPlc (da child a PlcPgm-object) e impostare l'attributo ScanOff su 0. Osservare che potrebbero esserci sottofinestre in questo programma che devono essere attivate.

9.4.6 /etc/proview.cnf

Un file di configurazione per la definizione di vari parametri sia in fase di sviluppo, runtime e ambiente di archiviazione. Nell'ambiente di sviluppo le definizioni sono valide per tutti i progetti sulla stazione.

Parametri nell'ambiente di sviluppo

mysqlSocket	socket mysql per volumi con database mysql. Valore predefinito "/var/run/mysqld/mysqld.sock".
mysqlServer	nodo server mysql.

defaultProjectRoot	Il percorso predefinito per i progetti per la creazione di nuovi progetti. Valore predefinito "/usr/local/pwrp".
defaultProductionBus	Il bus di produzione predefinito per la creazione di nuovi progetti. Valore predefinito 1.
defaultSimulationBus	Il bus di simulazione predefinito per la creazione di nuovi progetti. Valore predefinito 999.
defaultSystemGroup	Il gruppo di sistema predefinito per la creazione di nuovi progetti. Valore predefinito "Common".
defaultNodeHierRoot	Nome predefinito dell'oggetto di livello superiore nella gerarchia dei nodi per creare nuovi progetti. Valore predefinito "Nodes".
qcomBusId	IdentitÃ del bus QCom per la simulazione.

Parametri nell'ambiente di runtime

qcomBusId	Identita' QCom bus.
curveExportFile	Nome file predefinito per la funzione di esportazione della curva cronologica.
webDirectory	Directory in cui i file Web sono posizionati e accessibili dal server web.

Parametri nell'ambiente di archiviazione

sevDatabaseType	Tipo di Database, mysql o sqlite.
sevXttDefaultPriv	Privilegi predefiniti per l'accesso al database sev da sev_xtt.
sevMysqlEngine	Mysql engine for created tables, innodb or myisam.

10 Programmazione grafica PLC

Questo capitolo descrive come si creano i programmi PLC.

L'Editor

Si entra nell'editor di plc da un oggetto PlcPgm nella configurazione dell'impianto. Seleziona l'oggetto e attiva 'Functions/Open Program' nel menu. La prima volta che si apre il programma, troverai un documento vuoto. Il programma e' costituito da blocchi di funzioni e sequenze Grafcet.

La programmazione con i blocchi funzione viene eseguita in una rete orizzontale di nodi e connessioni da sinistra verso destra nel documento. I segnali o gli attributi vengono recuperati sul lato sinistro della rete e i valori vengono trasferiti tramite connessioni dai pin di uscita ai pin di input dei blocchi di funzioni.
I blocchi funzione operano sui valori e sul lato sinistro della rete, i valori sono memorizzati in segnali o attributi.

Le sequenze di Grafcet consistono in una rete verticale di nodi e connessioni. Uno stato viene trasferito tra i passaggi nella sequenza tramite le connessioni. Grafcet e blocchi funzionali possono interagire tra di loro ed essere combinati in una rete.

Modifica oggetti funzione

L'editor di plc e' composto da
- un'area di lavoro
- una tavolozza con oggetti grafcet e blocchi funzione e una tavolozza con collegamenti
- una finestra di navigazione, dalla quale e' possibile scorrere e ingrandire l'area di lavoro

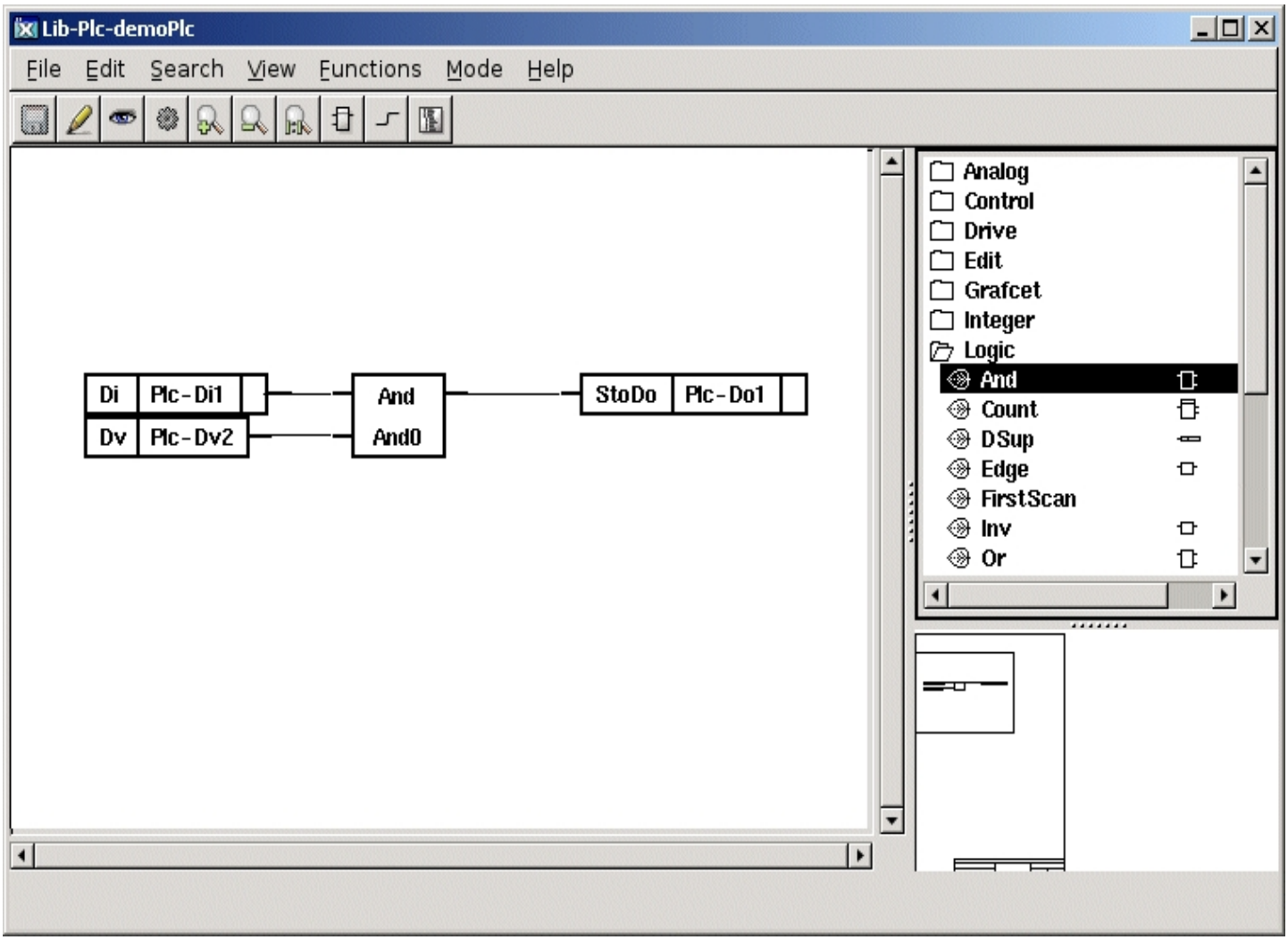


Fig L'Editor Plc

Un oggetto funzione viene creato selezionando una classe nella tavolozza e premendo MB2 nell'area di lavoro.

Modifica l'oggetto

L'oggetto viene modificato dall'editor di oggetti. Questo viene aperto selezionando l'oggetto ed attivando 'Functions/Open Objects' nel menu. I valori degli attributi dell'oggetto vengono modificati con 'Functions/Change value' nel menu dell'editor di oggetti o premendo il tasto freccia destra.
Se un input o output non viene utilizzato, puo' essere rimosso per mezzo di una casella di controllo. C'e' anche una casella di controllo che indica che il valore di un ingresso digitale deve essere invertito.

Collega oggetti funzione

Un pin di uscita e un pin di ingresso sono collegati da

- Posiziona il cursore sul pin o in un'area vicino al pin dell'oggetto funzione e poi premi MB2.
- Trascinare il cursore sull'altro pin o su un'area dell'oggetto funzione vicino al pin e rilasciare MB2.

Viene ora creata una connessione tra gli oggetti funzione.

Recuperare un valore di segnale

Il valore di un segnale Di viene recuperato con un oggetto GetDi. L'oggetto GetDi deve puntare a un segnale Di e questo viene fatto selezionando il segnale nella configurazione dell'impianto, quindi premere Ctrl e fare doppio clic sull'oggetto GetDi.
Il nome del segnale viene ora visualizzato nel disegno.
Segnali Dv, i segnali e gli attributi vengono recuperati allo stesso modo, con oggetti GetDv, GetDo e GetDp.

Il modo piu' semplice per creare un oggetto Get e' disegnare una connessione dal punto di input in cui l'oggetto Get deve essere connesso e rilasciarlo in uno spazio vuoto nell'area di lavoro. Viene ora creato un oggetto Get generico che verra' trasformato in un oggetto Get del tipo corretto quando viene specificato il segnale. Il segnale viene specificato come prima selezionando il segnale nella gerarchia dell'impianto e Ctrl/doppio clic sull'oggetto Get.

Memorizza un valore in un segnale

Il valore di un'uscita da un oggetto funzione e' memorizzato nel segnale Do utilizzando oggetti StoDo.
L'oggetto StoDo e' connesso a un segnale Do allo stesso modo degli oggetti Get. I segnali e gli attributi Dv sono memorizzati con oggetti StoDv e StoDp.

Nozioni di base sul Grafcet

Questa sezione offre una breve introduzione al Grafcet. Per una descrizione piu' dettagliata, leggere un manuale di riferimento sul Grafcet. Grafcet e' una norma o un metodo internazionale da utilizzare per il controllo sequenziale.

Un programma Grafcet e' composto da un certo numero di passi, e ad ogni passo sono connessi uno o piu' comandi, che verranno eseguiti quando il passo e' attivo.
Per passare da un passo all'altro, viene utilizzata una transizione. In ogni transizione si hanno condizioni di transizione e il passaggio da un passo al successivo puo' avvenire solo quando le condizioni di transizione sono state soddisfatte.

Singola sequenza diritta
Osserviamo la singola sequenza sottostante e assumiamo che il passo sia attivo, il che significa che verra' eseguito l'ordine connesso al passo iniziale. Questo ordine verra' eseguito fino a quando il passaggio iniziale diventa inattivo. Il passaggio 1 diventa attivo quando la condizione di transizione per la transizione 1 e' stata soddisfatta.
Quindi il passaggio iniziale diventa inattivo.

Un programma Grafcet e' sempre una sequenza chiusa.

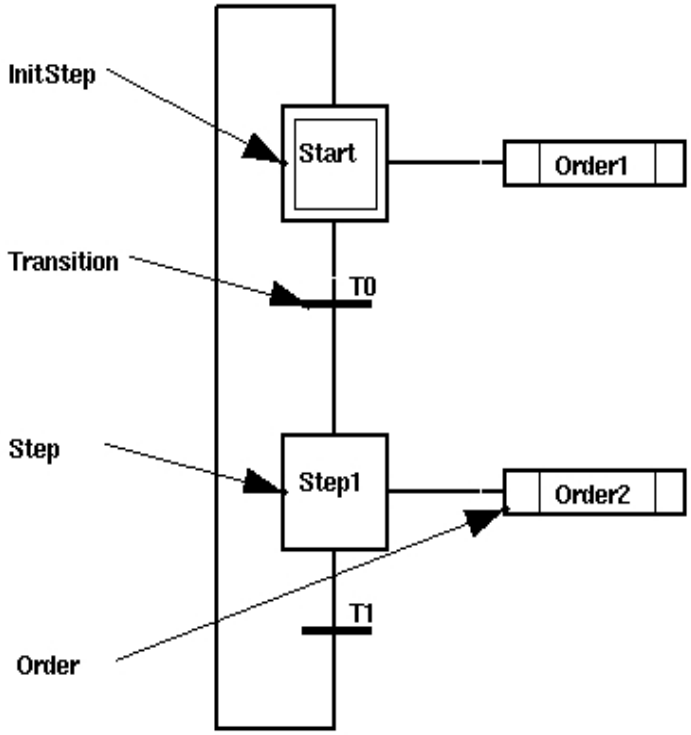


Fig A Simplice Sequenza Lineare in Grafcet

Sequenza divergente

Una sequenza diritta e' la variante piu' semplice delle sequenze. A volte potresti aver bisogno di rami alternativi nel tuo programma, ad esempio quando hai una macchina in grado di produrre tre prodotti diversi.
Nei punti in cui la produzione differisce, si introducono rami alternativi.

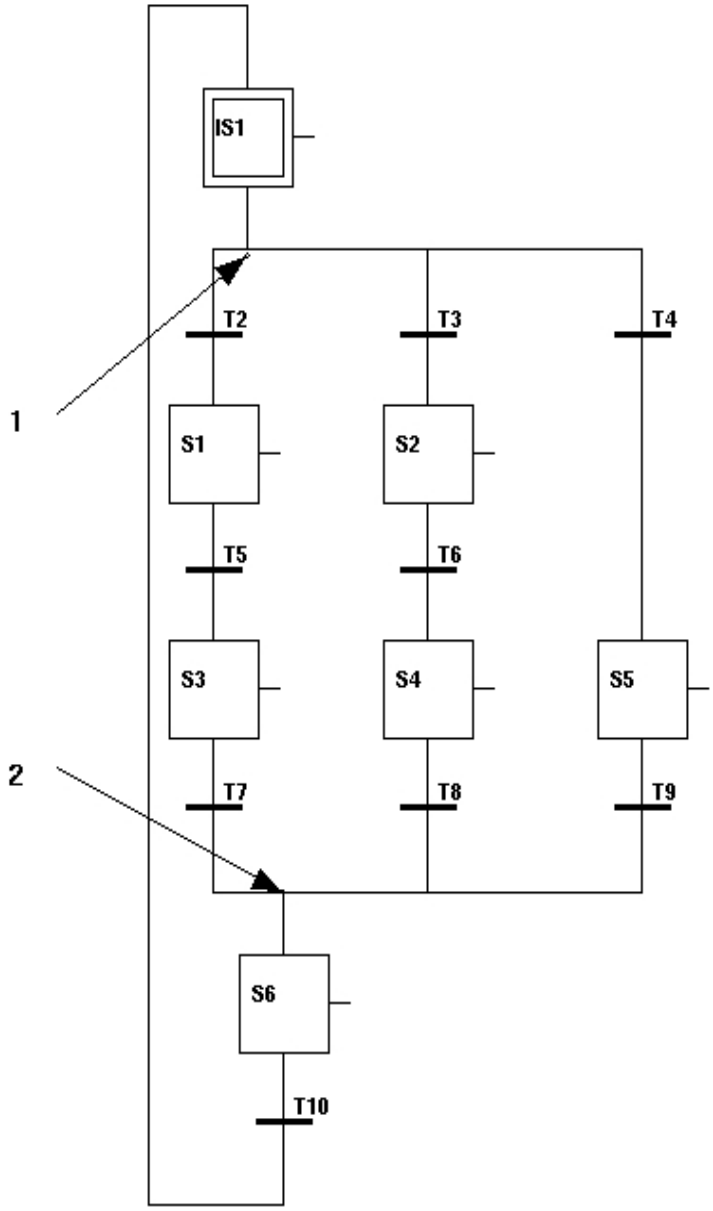


Fig Sequenza di selezione

L'esempio nella figura sopra mostra la sequenza di una macchina che puo' produrre i tre prodotti, Rosso, Verde e Blu. Nel punto di divergenza, il punto 1 nella figura, si sceglie il ramo desiderato in base al prodotto da produrre. I rami alternativi divergono da un passo, seguito da una condizione di transizione in ciascun ramo. E' compito dei costruttori verificare che solo una delle condizioni di transizione sia soddisfatta.
Se diverse transizioni sono soddisfatte, non e' definito quale sia selezionata.
Al punto 2 nella figura, i rami convergono in un passo comune.

Sequenze Parallele

A volte puo' essere necessario avviare contemporaneamente piu' procedure di lavoro parallele. Queste procedure operative devono essere indipendenti l'una dall'altra.
Per fare questo, vengono utilizzate sequenze parallele.

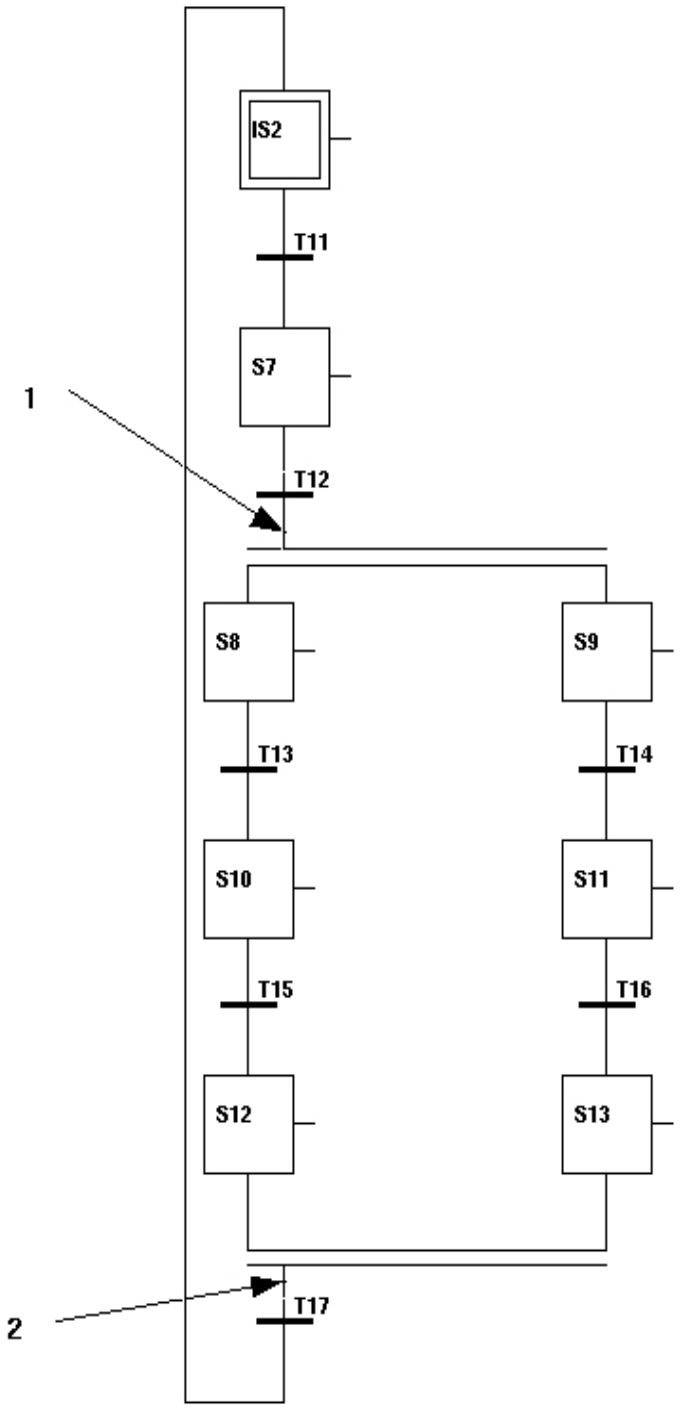
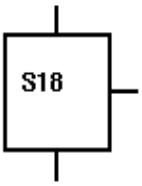


Fig Sequenze parallele

L'esempio nella figura sopra illustra la sequenza di due macchine che eseguono due fori contemporaneamente e indipendenti l'uno dall'altro. Quando la condizione di transizione prima della divergenza parallela (punto 1 nella figura) e' soddisfatta, l'attivita' viene spostata su entrambi i rami e le macchine iniziano la perforazione.
La perforazione viene eseguita indipendentemente l'una dall'altra.

I rami convergono in una condizione di transizione (punto 2 nella figura) e quando la perforazione e' completata in entrambe le macchine, cioe' sia S12 sia S13 sono attivi e la condizione di transizione T17 e' soddisfatta, l'attivita' viene spostata nella fase di inizializzazione IS2.

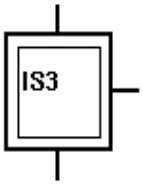
Passo



Un passo e' usato per descrivere uno stato nel processo. Quanto segue si applica ad un passo:

- Un passo puo' essere attivo o inattivo.
- Un attributo, Order, indica se il passo e' attivo o meno.
- e' possibile collegare uno o piu' ordini a un passo.
- Al passo puo' essere assegnato un nome a propria discrezione.

InitStep



In ogni sequenza e' necessario avere un passo iniziale (InitStep) che differisce da un passo normale nel modo seguente:

- Dovresti avere solo un passo iniziale in una sequenza.
- Quando il programma inizia la sua esecuzione, il passo iniziale e' attivo.
- e' sempre possibile attivare il passo iniziale impostando il segnale di reset.

Transizione - Trans

Come accennato in precedenza, la transizione (Trans) viene utilizzata per avviare una transizione tra un passaggio attivo e uno inattivo. Una condizione logica, ad esempio un segnale digitale, e' collegata a una transizione e determina quando avviene la transizione.

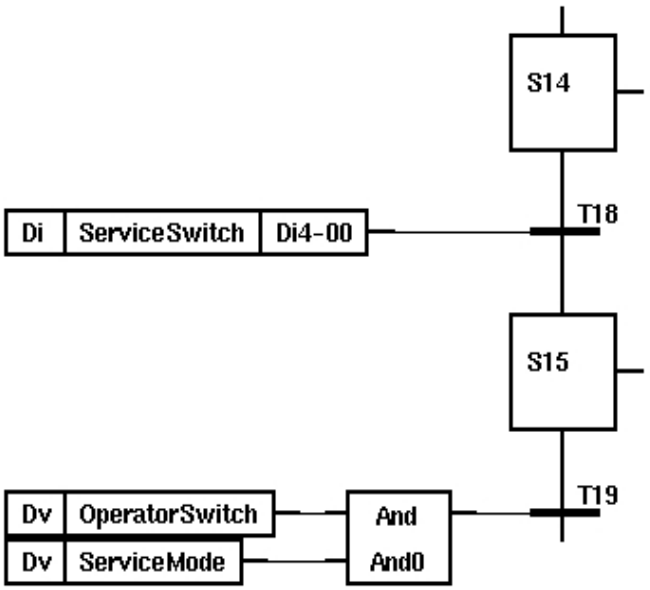


Fig Un esempio di transizione

Ordine

e' possibile collegare uno o piu' ordini ad ogni passo.

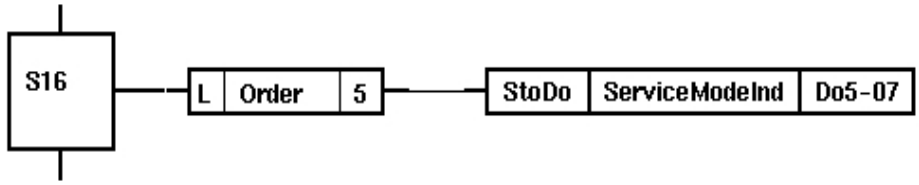


Fig Un esempio di ordine

Normalmente l'output e' attivo quando l'input e' attivo, ma per ogni ordine hai un numero di attributi, con il quale puoi influenzare la funzione dell'output:

- D Ritardo
- L Limite di tempo
- P Impulso
- C Conizionale
- S Memorizzato

Queste funzioni sono descritte in dettaglio nel Manuale di riferimento degli oggetti Proview. I principi sono che si indica il nome dell'attributo (lettere maiuscole) e il tempo se possibile tramite l'Editor oggetti. La figura seguente illustra come ritardare un ordine all'attivazione per 10 secondi.

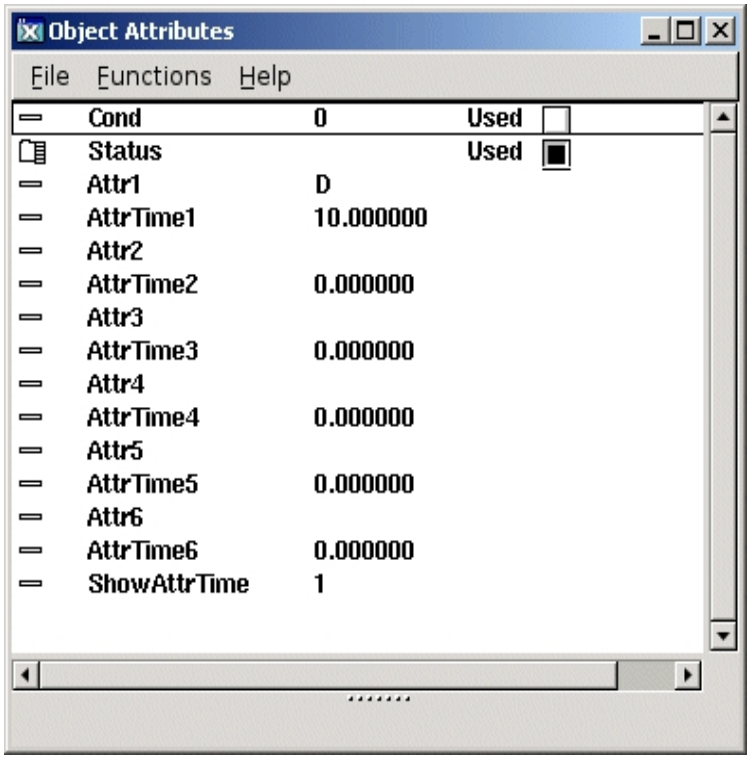


Fig Attributi DOrder

Gli attributi dell'ordine selezionati sono scritti nel simbolo dell'ordine.

La figura seguente illustra come e' possibile utilizzare un oggetto ordine con ritardo per rendere attivo un passo per un certo periodo.

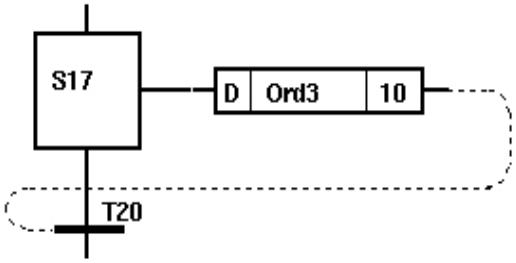


Fig Una transizione ritardata

Nota! e' necessario utilizzare una connessione ConFeedbackDigital per connettere l'oggetto dell'ordine ritardato con l'oggetto di transizione, altrimenti l'ordine di esecuzione sara' ambiguo.
Vedi Feedback Connection

SottoSequenze - SubStep

Quando si creano programmi Grafcet complessi, e' spesso opportuno utilizzare le sottofinestre e inserire in queste sottosequenze per ottenere un layout migliore del programma.

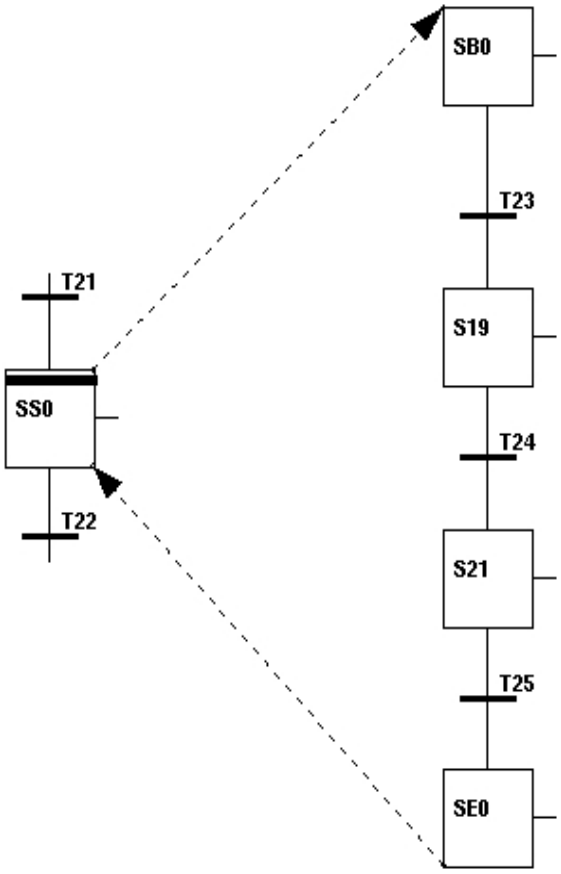


Fig Sottosequenze

La figura sopra mostra la sotto sequenza di un SubStep. Una sequenza secondaria inizia sempre con un oggetto SsBegin e termina con un oggetto SsEnd.
A sua volta una sottosequenza puo' contenere sottosequenze.

Costruire sequenze Grafcet

Le sequenze di Grafcet vengono create facilmente nell'editor plc iniziando con un oggetto Step o InitStep.
Disegnando una connessione dal punto di connessione inferiore del passo e rilasciando la connessione in uno spazio vuoto nell'area di lavoro, verra' creato un oggetto Trans collegato. Allo stesso modo gli oggetti ordine(comando) vengono creati dal punto di connessione destro del passo, e dall'oggetto Trans vengono creati nuovi oggetti Step dai punti di connessione superiore e inferiore.

Introduzione alla programmazione a blocchi funzione

Blocchi per recuperare e memorizzare valori

I blocchi per recuperare e memorizzare sono usati per leggere e scrivere valori. Ci sono blocchi di recupero e di memorizzazione per ogni tipo di segnale. Nella figura seguente viene visualizzato un numero di questi blocchi.
Si trovano nella cartella 'Signal' nella tavolozza.

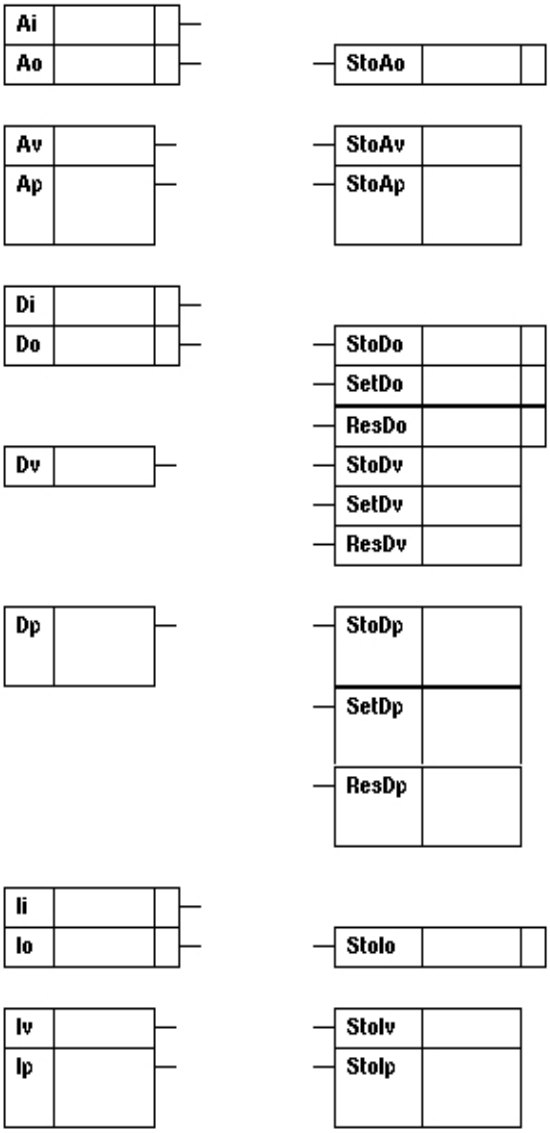


Fig Blocchi per recuperare e memorizzare valori

Per leggere i segnali si usano i blocchi GetAi, GetIi, GetDv o GetAo. Quando si desidera assegnare un valore ad un segnale, si utilizzano ad esempio StoAv, StoDo, SetDv o ResDo.

I valori digitali possono essere scritti in due modi:

- 'Sto' memorizza il valore di ingresso, cioe' se l'ingresso e' 1 il segnale diventa 1 e se l'ingresso e' zero il segnale diventa zero.
- 'Set' imposta il segnale su uno se l'input e' true, Res imposta il segnale su zero se l'input e' true. Ad esempio, se si imposta un segnale di uscita digitale con un oggetto SetDo, questo rimarra' impostato fino a quando non lo si ripristinera' con un oggetto ResDo.

Per leggere, assegnare rispettivamente valori attributo a attributi diversi da ActualValue, si utilizza quanto segue:

- attributi analogici, GetAp and StoAp
- attributi integer, GetIp and StoIp
- attributi digital, GetDp and StoDp, SetDp or ResDp
- attributi string, GetSp and StoSp
- attributi time, GetAtp, GetDtp, StoAtp and StoDtp

Blocchi logici

Sono disponibili numerosi oggetti per la programmazione logica, ad esempio And-gate (And), Or-gate (Or), inverter o timer. Per la programmazione logica vengono utilizzati i segnali digitali. Gli oggetti sono posizionati nella cartella Logic.

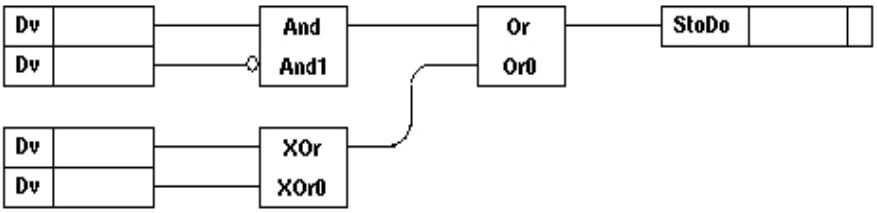
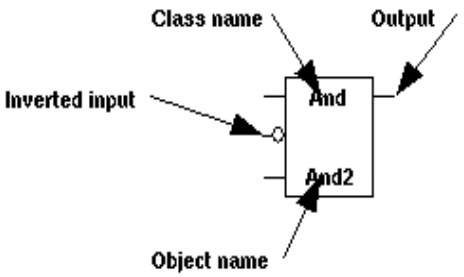


Fig Blocchi Logici

La figura in basso mostra un And-gate. Per questo oggetto vale quanto segue:

- Ingressi sulla sinistra
- Uscite sulla destra
- Nome della Classe scritto in alto
- Il nome dell'oggetto e' scritto in fondo (puo' essere modificato dall'utente)
- e' possibile utilizzare un numero variabile di input, il valore predefinito e' 2
- Gli ingressi possono essere invertiti, indicati da un anello sull'ingresso del simbolo

And-gate



Gli attributi di And-gate vengono modificati con l'Object Editor.

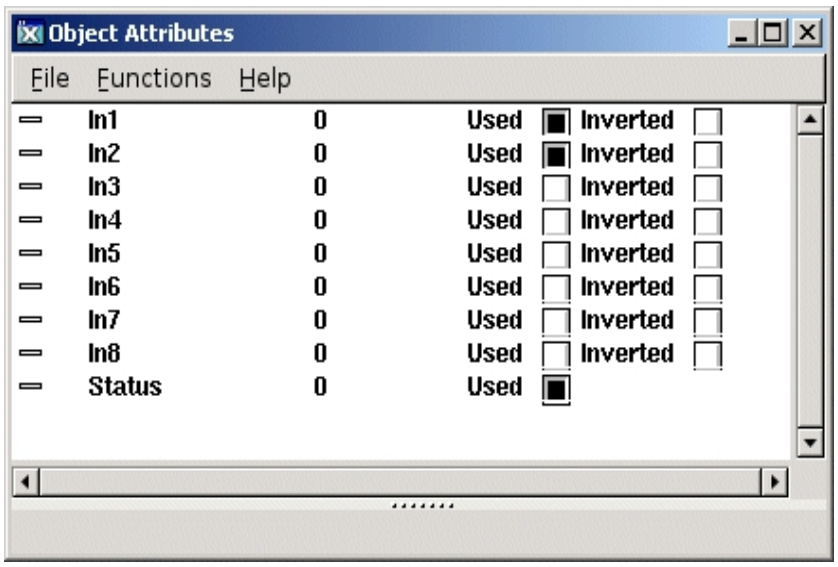


Fig Attributi di una And-gate

Gli altri oggetti nella cartella "Logic" hanno parametri simili, vedere Manuale di riferimento Oggetti Proview.

Blocchi di calcolo

La cartella "Analog" contiene un numero di oggetti per la gestione di segnali analogici, ad esempio filtri, blocchi di somma e integratori.

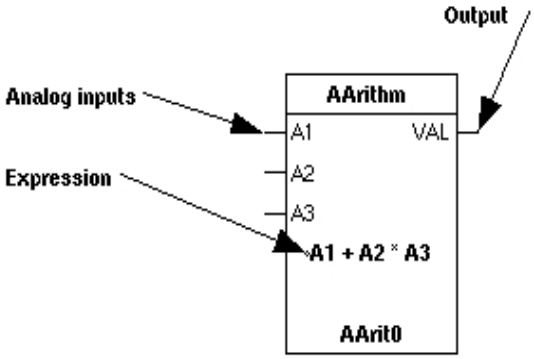


Fig Blocco di calcolo aritmetico

In questa guida non descriviamo la funzione degli oggetti, ma potrebbe essere utile commentare l'uso di blocchi aritmetici. I blocchi vengono utilizzati per il calcolo di espressioni definite dall'utente. Questi sono scritti nel linguaggio C.

Nella figura sottostante il blocco calcolera' l'espressione (A1 + A2 * A3) e dara' all'output questo valore. A1, A2 e A3 rappresentano valori analogici, ad esempio segnali che dovrebbero essere collegati agli ingressi dell'oggetto.
Quando si scrivono queste espressioni e' importante utilizzare lo spazio prima e dopo gli operatori, altrimenti l'espressione potrebbe essere interpretata erroneamente durante l'esecuzione.

L'espressione puo' contenere codice C avanzato con matrici e puntatori. Quando si scrive il codice per questi blocchi, e' necessario essere consapevoli che l'indicizzazione al di fuori degli array o puntatori errati potrebbero causare l'interruzione dell'esecuzione del programma plc.

Supervisione degli allarmi

In Proview e' possibile supervisionare i segnali analogici e digitali. La supervisione dei segnali analogici viene effettuata rispetto a un valore limite. Se il limite supervisionato viene superato, viene inviato un allarme al gestore dei messaggi, che a sua volta invia l'allarme all'unita' di uscita, ad es. una finestra di dialogo dell'operatore.

Vedere il Manuale di riferimento degli oggetti Proview per quanto riguarda gli attributi degli oggetti.

Supervisione dei segnali digitali

Per la supervisione di un segnale o attributo digitale, si utilizza l'oggetto DSup (Supervisione Digitale), che si trova nella cartella Logic.

Il segnale o l'attributo desiderato viene recuperato con un oggetto Get collegato all'oggetto DSup. Le uscite dei blocchi logici possono essere collegate direttamente all'oggetto DSup.

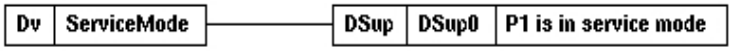


Fig Supervisione dei segnali digitali

La figura sopra illustra la supervisione di un segnale Dv e un attributo digitale.

Si ha anche un attributo dell'oggetto DSup, 'CtrlPosition', che indica se l'allarme verra' attivato quando il segnale/attributo supervisionato diventa vero o falso.

Supervisione dei segnali analogici

Per la supervisione di un segnale o attributo analogico, si utilizza l'oggetto ASUP (supervisione analogica), che si trova nella cartella "Analog" nella palette.

La supervisione avviene allo stesso modo degli oggetti DSup con l'eccezione che e' possibile scegliere se l'allarme sara' attivato quando il valore e' superiore o inferiore al limite di supervisione.

Ordine di esecuzione

L'ordine di esecuzione e' determinato dal modo in cui gli oggetti funzione sono accoppiati. Un oggetto il cui output e' collegato all'ingresso di un altro oggetto, viene eseguito prima dell'oggetto a cui e' collegato. Puoi visualizzare l'ordine di esecuzione attivando 'View/Show Execute Order' nel menu dell'editor di plc. Per oggetti e reti non accoppiati, l'ordine di esecuzione e' determinato dalla posizione.
L'oggetto piu' in alto viene eseguito per primo.

Se si desidera influenzare l'ordine di esecuzione tra due oggetti, e' possibile tracciare una speciale connessione di ordine di esecuzione tra gli oggetti. Questa viene scelto nella tavolozza delle connessioni e ha una freccia a un'estremita'. L'oggetto a cui punta la connessione verra' eseguito dopo l'oggetto da cui si emana la connessione.

Se una rete contiene una connessione di feedback, l'ordine di esecuzione non puo' essere determinato e l'errore 'Amigous execute order' viene segnalato. Quindi devi scambiare una connessione con una connessione di feedback, cioe' una connessione che non determina l'ordine di esecuzione. La connessione di feedback e' tratteggiata ed e' selezionata nel

pannello degli strumenti o nella tavolozza dei collegamenti.

Copia I/O

Se un segnale viene utilizzato in piu' punti in una rete di oggetti funzione, esiste la possibilita' che il valore del segnale venga modificato durante l'esecuzione, il che puo' causare blocchi e altri fenomeni difficili da prevedere. Pertanto viene utilizzato un meccanismo chiamato copia I/O. I valori per i segnali di tipo Ai, Ao, Av, Di, Do, Dv, Ii, Io, Iv, Co, Bo e Bi sono raccolti in oggetti area speciali. Prima che un thread plc inizi ad essere eseguito, viene eseguita una copia degli oggetti area e durante l'esecuzione tutte le letture vengono eseguite dalla copia, mentre le scritture vengono trasformate negli oggetti area originali. Questo assicura che questo tipo di fenomeni sia evitato, ma anche che si possano ottenere dei ritardi. Se viene impostato un valore di segnale e quindi letto nello stesso thread plc, la modifica non verra' registrata fino alla scansione successiva.

Compilare il plcpgm

Prima di iniziare a compilare, devi indicare su quale piattaforma (o piattaforme) deve essere eseguito il volume del plc. Aprire l'editor degli attributi del volume dal menu del navigatore: 'File/Volume Attributes' e accedere al Sistema operativo. Si noti che e' possibile scegliere piu' di un sistema operativo. Il volume puo' allo stesso tempo essere eseguito nel sistema di produzione, in un sistema di simulazione e in un sistema educativo, e i sistemi possono avere piattaforme diverse.

Il plcgpm viene compilato attivando 'File/Build' nel menu dell'editor di plc. Qualsiasi messaggio di avviso o di errore verra' visualizzato nella finestra del messaggio. Quando si costruisce il nodo, verra' compilato anche qualsiasi plcpgm nuovo o modificato.

11 Richiamo di funzioni dal programma plc

La programmazione con oggetti funzione nell'editor di plc ha i suoi limiti e alcune attivita' possono essere eseguite in modo molto piu' semplice e piacevole in codice C. La programmazione C puo' essere relizzata in CArithm e DataArihm dove e' possibile inserire una quantita' di codice C, ma il numero di caratteri e' limitato a 1023 (8191 per DataArithmL), e occasionalmente questo approccio puo' essere insufficiente. Allora avete due possibilita'; scrivere un'applicazione distaccata o chiamare una funzione C da un CArithm o da un DataArithm. Il vantaggio nel chiamare una funzione C e' che tutte le inizializzazioni e il collegamento a oggetti e attributi sono gestiti dal programma plc. L'esecuzione della funzione e' anche sincrona con l'esecuzione del thread plc che chiama la funzione.

Scrivere il codice

Il codice e' inserito in un file C, creato da qualche parte sotto $ pwrp_src. Creiamo il file$ pwrp_src/ra_myfunction.c e inseriamo la funzione MyFunction() che esegue alcuni semplici calcoli.

```
#include "pwr.h"
#include "ra_plc_user.h"

void MyFunction( pwr_tBoolean cond, pwr_tFloat32 in1, pwr_tFloat32 in2,
                 pwr_tFloat32 *out)
{
  if ( cond)
    *out = in1 * in2;
  else
    *out = in1 + in2;
}
```

Dichiarazione del prototipo

Nel file include ra_plc_user.h viene inserita una dichiarazione prototipo.

```
void MyFunction( pwr_tBoolean cond, pwr_tFloat32 in1, pwr_tFloat32 in2,
                 pwr_tFloat32 *out);
```

ra_plc_user.h e' incluso dal programma plc e la funzione puo' essere chiamata da un oggetto CArithm o DataArithm. Dovresti includere anche ra_plc_user.h nel codice funzione per assicurarti che il prototipo sia corretto.

ra_plc_user deve essere posto in $ pwrp_src e copiato in $ pwrp_inc, da dove viene incluso dal programma plc e dal codice della funzione.

Compilare il codice

Il file C e' compilato, ad esempio con make. Sotto viene mostrato un makefile, che compila ra_myfunction.cpp e inserisce il risultato, ra_myfunction.o su $pwrp_obj. Si noti che esiste anche una dipendenza da ra_plc_user.h, che fa si' che questo file venga copiato da $pwrp_src a $pwrp_inc.

```
ra_myfunction_top : ra_myfunction

include $(pwr_exe)/pwrp_rules.mk

ra_myfunction_modules : \
  $(pwrp_inc)/ra_plc_user.h \
  $(pwrp_obj)/ra_myfunction.o

ra_myfunction : ra_myfunction_modules
 @ echo "ra_myfunction built"

#
# Modules
#

$(pwrp_inc)/ra_plc_user.h : $(pwrp_src)/ra_plc_user.h

$(pwrp_obj)/ra_myfunction.o : $(pwrp_src)/ra_myfunction.c \
                              $(pwrp_inc)/ra_plc_user.h
```

Chiamata nel programma plc
La funzione e' chiamata da un CArithm o da DataArithm.

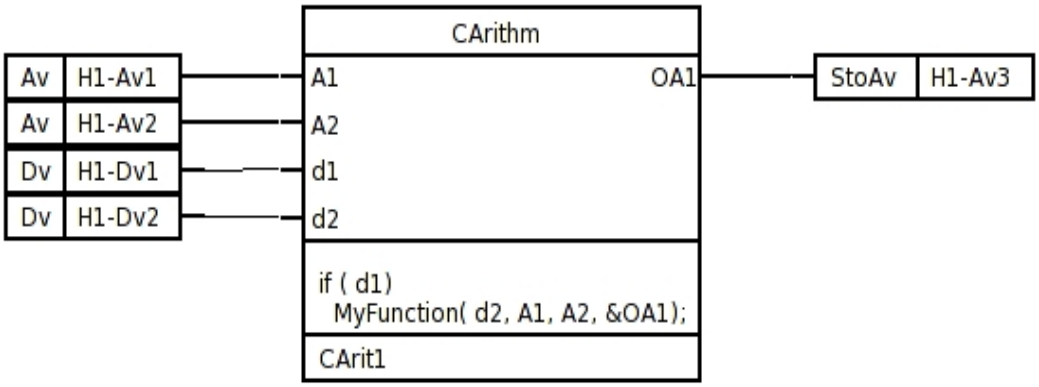


Fig Richiamo della Funzione da un programma PLC

Link del programma plc
Quando il codice sorgente della funzione e' stato compilato, e' stato creato il modulo oggetto $pwrp_obj/ra_myfunction.o. Questo deve essere aggiunto al comando link quando viene creato il programma plc, questo si ottiene creando un oggetto BuildOptions nel volume della directory sotto l'oggetto NodeConfig. Inserire il nome del modulo oggetto nell'array ObjectModules. Quando il volume della directory viene salvato, viene creato un file opt su $pwrp_exe che verra' incluso dal linker al momento della creazione del nodo.

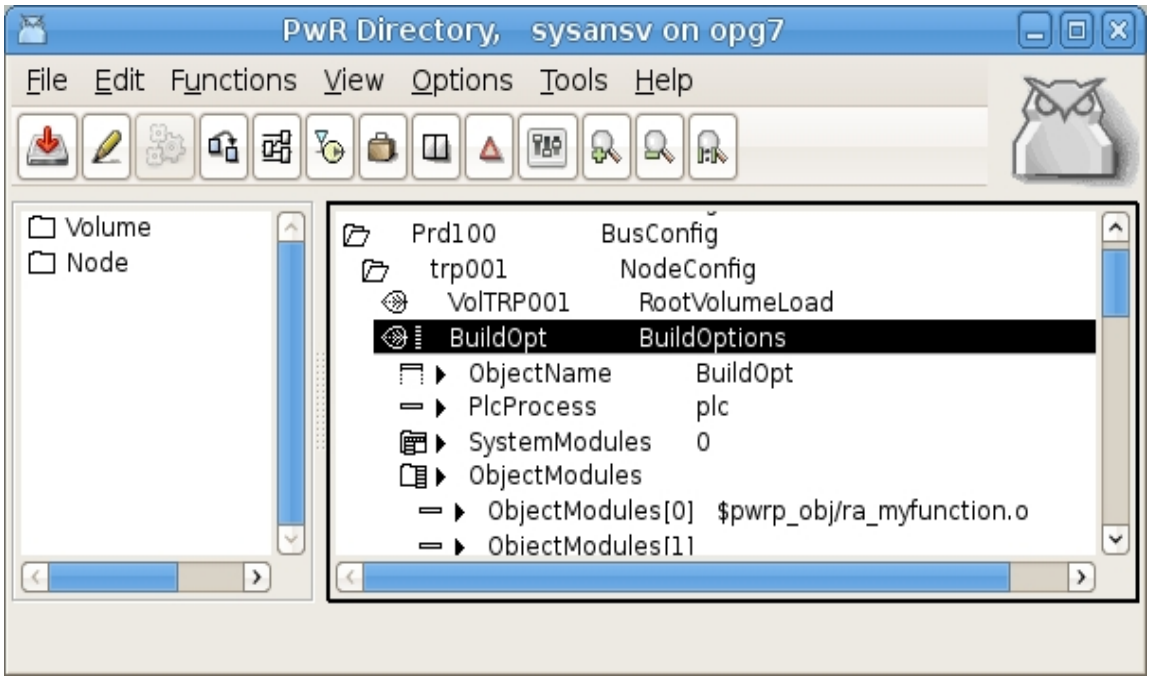


Fig Il modulo oggetto inserito in BuildOptions

Ora possiamo costruire il nodo e avviare il runtime Proview.

Eliminare gli errori (Debug)

Uno svantaggio quando si esce dalla programmazione grafica e si richiamano funzioni C e' che non e' piu' possibile utilizzare la traccia per eseguire il debug. Se si sospetta qualche errore nel codice funzione, e' necessario avviare occasionalmente il programma PLC in debug, impostare un punto di interruzione nella funzione e avanzare nel codice.

Per prima cosa devi compilare il programma plc con le funzionalita' debug, aprendo Options/Setting dal configuratore e attivando Build/Debug, e poi costruisci il nodo.

Dopo di cio', avvia il runtime di Proview e collega il debugger, gdb, al processo di plc avviando gdb con il pid per il processo. pid e' visualizzato da 'ps x'

```
> ps x
...
5473 pts/0    Sl     0:18 plc_mynode_0999_plc
```

dove 5473 e' pid per il processo di plc, e avviamo il debugger, impostiamo un breakpoint nella funzione e lasciamo che il programma continui l'esecuzione

```
> gdb -p 5473 plc_mynode_0999_plc
(gdb) b MyFunction
(gdb) c
```

Quando il programma entra nella funzione si ferma nel debugger, e possiamo avanzare a passi ed esaminare il contenuto delle variabili (x), ecc.

Se il programma plc viene terminato immediatamente dopo l'avvio, e' possibile riavviarlo in debug.

```
> gdb plc_mynode_0999_plc
```

Puoi anche uccidere il processo plc corrente e avviarne uno nuovo in debug.

```
> killall plc_mynode_0999_plc
> gdb plc_mynode_0999_plc
```

12 Componenti e Raggruppamenti

Questo capitolo tratta di come programmare con componenti e aggregati.

Un componente e' un ogetto, o un certo numero di oggetti, che gestisce un componente dell'impianto. Un componente puo' essere ad esempio una valvola, un contattore, un sensore di temperatura o un convertitore di frequenza. Poiche' questi componenti sono molto comuni ed esistono in molti e diversi tipi di impianto, e' un grande vantaggio se possiamo costruire un oggetto che contiene tutto cio' che e' necessario per controllare e supervisionare il componente e che sia cosi' generico da poter essere utilizzato nella maggior parte delle applicazioni.

Un componente in Proview puo' essere diviso in un numero di oggetti:
- un oggetto principale contenente dati di configurazione e dati necessari per supervisionare e utilizzare il componente.
  Contiene anche gli oggetti segnale per il componente.
- un oggetto funzione inserito nel programma plc e che contiene il codice per controllare il componente.
- un oggetto I/O che definisce una possibile comunicazione con, ad esempio, un modulo profibus.
- un oggetto simulato, utilizzato per testare e simulare il sistema.

Inoltre un oggetto grafico, documentazione, grafici ecc. Sono inclusi nel componente.

Un aggregato e' una parte piu' grande dell'impianto rispetto al componente e contiene un numero di componenti. Un aggregato puo' ad esempio essere un azionamento pompa, costituito da componenti pompa, motore, contattore e interruttore di sicurezza.
Per altri aspetti, l'aggregato e' costruito come un componente con oggetto principale, oggetto funzione, oggetto simulato, grafico oggetto, documentazione, ecc.

Orientamento all'oggetto
Proview e' un sistema orientato agli oggetti e componenti e aggregati sono un campo in cui vengono utilizzati i vantaggi dell'orientamento agli oggetti. Nei componenti, si puo' vedere come un oggetto e' costruito da altri oggetti e che un attributo, oltre ad essere un tipo semplice come float o booleano, puo' anche essere un oggetto, che a sua volta e' composto da altri oggetti. Un attributo che e' un oggetto e' chiamato un oggetto attributo.
Non e' del tutto analogo a un oggetto indipendente, in quanto privo di testa di oggetto e di identita' di un oggetto, ma a parte cio' contiene tutte le proprieta' di un oggetto indipendente in termini di metodi, grafici degli oggetti, ecc.

Un esempio di un oggetto attributo puo' essere visto nell'oggetto componente di un'elettrovalvola.
Qui tutti gli oggetti segnale, due oggetti Di per i finecorsa e un oggetto Do per il comando, sono posizionati internamente come oggetti attributi. Quindi, non dobbiamo creare questi segnali separatamente. Quando viene creato l'oggetto valvola, vengono creati anche i segnali per la valvola.
Un altro esempio di oggetto attributo e' l'aggregato motore che contiene gli oggetti componente per il convertitore di frequenza, l'interruttore di sicurezza, motore ecc. sotto forma di oggetti attributo.

Un'altra proprieta' importante nell'orientamento agli oggetti e' l'ereditarieta'.
Con l'ereditarieta' si puo' creare una sottoclasse derivata da una classe esistente, una superclasse. La sottoclasse eredita tutte le proprieta' della superclasse, ma ha anche la

possibilita' di estendere o modificare alcune proprieta'. Un esempio e' un componente per un sensore di temperatura che e' una sottoclasse di un oggetto sensore generale per sensori analoghi. L'unica differenza tra la classe del sensore di temperatura e la sua superclasse e' il grafico dell'oggetto, in cui il valore del sensore e' presentato sotto forma di un termometro anziche' di una barra. Un altro esempio e' un aggregato di pompe derivato da un aggregato motore. L'aggregato della pompa viene esteso da un oggetto attributo della pompa e presenta anche un grafico oggetto modificato che oltre al controllo motore visualizza anche una pompa.

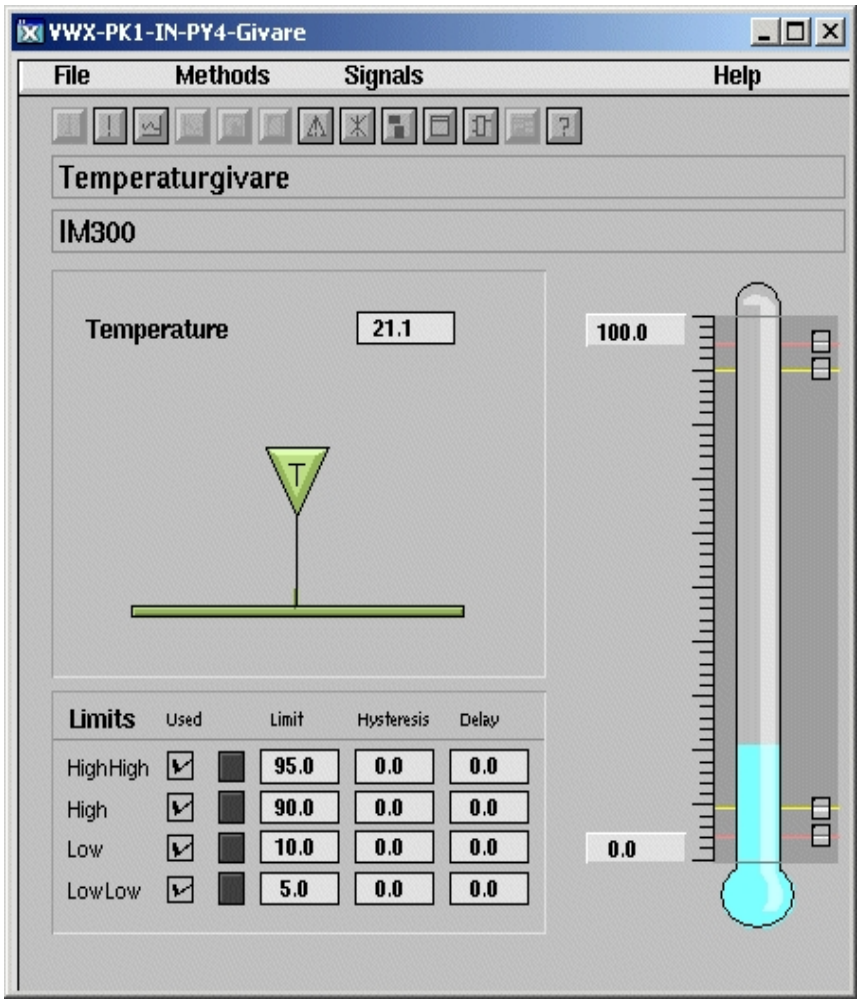


Fig Grafica dell'oggetto per la classe BaseTempSensor

Un'altra proprieta' che abbiamo introdotto e' la possibilita' di disabilitare gli attributi.
La ragione di cio' e' che gli oggetti componenti devono essere il piu' generici possibile, per essere in grado di gestire tutte le varianti del componente dell'impianto.
Una valvola a solenoide puo', per esempio, avere un finecorsa che indica la valvola aperta, ma esistono anche elettrovalvole con un finecorsa che indica la valvola chiusa, o valvole con entrambi gli interruttori o senza interruttori.
Naturalmente potremmo creare quattro diverse classi di componenti, una per ogni alternativa dei finecorsa, ma i problemi sorgono quando si inizia a creare aggregati dei componenti.
Il numero di varianti di un aggregato sara' presto ingestibile se si desidera coprire tutte le varianti dei componenti. Se, ad esempio, vogliamo creare un aggregato contenente quattro elettrovalvole, e ci sono quattro varianti di ogni valvola, ci saranno 64 varianti dell'aggregato.
Se vogliamo costruire un aggregato contenente quattro aggregati di valvole, il numero di varianti e' 4096. La soluzione e' costruire un componente valvolare che contiene entrambi gli interruttori, ma dove e' possibile disabilitarne uno o entrambi per poter gestire tutte e quattro le varianti dei finecorsa. In questo caso gli oggetti attributi di classe Di sono

disabilitati, il che significa che non sono visualizzati nel navigatore, sono ignorati dalla gestione I/O. Anche il codice per il componente della valvola e del grafico dell'oggetto viene preso in considerazione. La configurazione viene effettuata nel configuratore dal menu popup in cui e' possibile selezionare una configurazione tra le alternative in "ConfigureComponent".

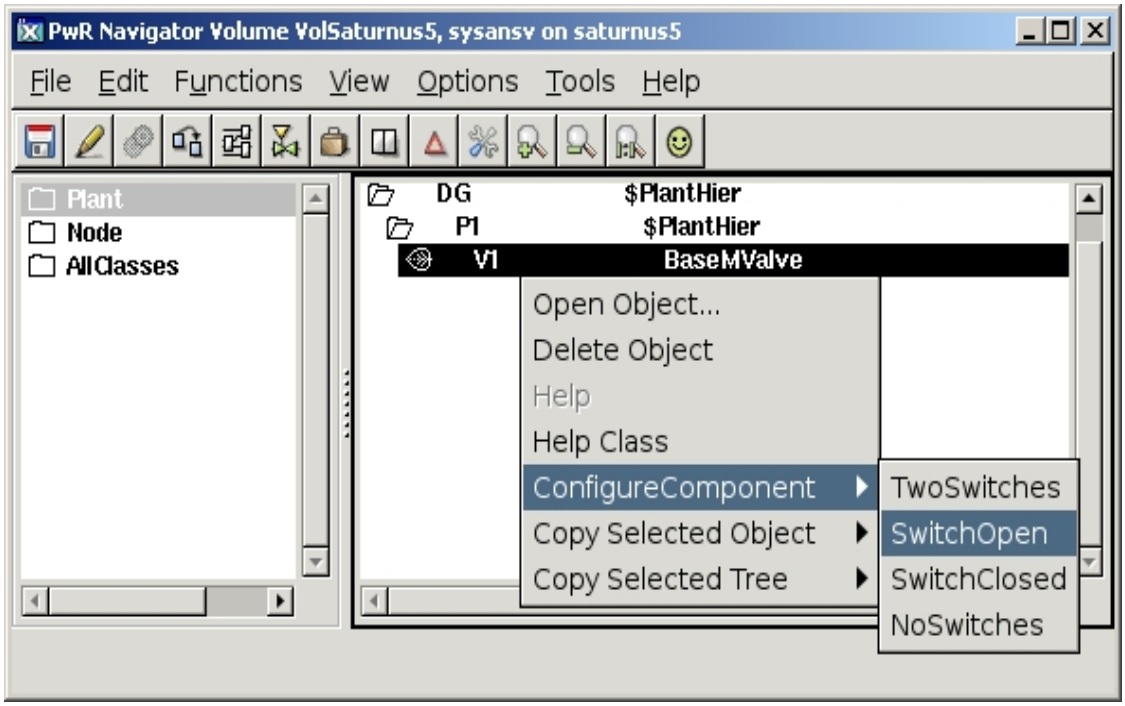


Fig Il metodo ConfigureComponent per un BaseMValve.

Basecomponents

Proview contiene un numero di componenti e oggetti aggregati per componenti comuni di impianto, ad es. sensori di temperatura, sensori di pressione, pressostati, elettrovalvole, filtri, motori e azionamenti di ventilatori.
Questi sono raccolti nel Classvolume BaseComponent. Un basecomponent puo' essere usato direttamente, e questo e' probabilmente il modo piu' comune di usarli, ma l'idea e' che voi dai basecomponents creiate librerie e classi per i componenti specifici che state usando nel vostro impianto.

Per le elettrovalvole ci sono le Baseclass BaseMValve. Se hai un'elettrovalvola di tipo Durholt 100.103, crei una sottoclasse con BaseMValve come superclasse,Durholt_Valve_100_103 e inserisci la configurazione valida per questa valvola. Inoltre, aggiungi un collegamento a un foglio dati e inserisci l'attributo Specifica, che consente di identificare e ordinare parti di ricambio alla valvola. Quando si utilizza un oggetto Durholt_Valve_100_103 non e' necessario eseguire molte configurazioni e adattamenti poiche' cio' e' gia' stato fatto nella classe. In questo modo, gli archivi componenti possono essere costruiti per i tipi di componenti che si utilizzano nel proprio impianto.

Un problema sorge quando si utilizzano gli aggregati. Un aggregato contiene basecomponents dal volume BaseComponent e se ci sono sottoclassi specifiche per un componente, si desidera utilizzarle. La soluzione e' la funzione Cast(fondere). Un basecomponent in un aggregato puo' essere 'castato' in una sottoclasse, dato che la sottoclasse non e' estesa con nuovi attributi.
Il casting significa che il componente recupera valori iniziali, configurazioni, metodi, oggetto grafico ecc dalla sottoclasse, cioe' in tutte le situazioni agisce come sottoclasse viene castato a. Il cast viene eseguito dal menu a comparsa nel configuratore, in cui l'alternativa "Cast" ottiene un elenco di tutte le sottoclassi disponibili.

Selezionando una sottoclasse, il componente viene castato(fuso) a questa.

Pressostato

Diamo un'occhiata ad un componente relativamente semplice, un pressostato, per esaminare come e' costruito e come configurarlo. Per i pressostati c'e' il componente base BasePressureSwitch, che e' una sottoclasse di BaseSupSwitch.
Poiche' temperatura, pressione e interruttori di soglia sono abbastanza simili, hanno una superclasse comune. BaseSupSwitch ha anche una superclasse, Component, che e' comune a tutte le classi di componenti.
La dipendenza della classe per la classe del pressostato puo' essere scritta

```
Component-BaseSubSwitch-BasePressureSwitch
```

La Classe Component

Il componente contiene gli attributi Description, Specification, HelpTopic, DataSheet, CircuitDiagram, Photo e Foto che sono quindi presenti in tutti i componenti.
Nella Description c'e' posto per una breve descrizione, in Specification si inserisce la specifica del modello, le altre sono usate per configurare i metodi corrispondenti nell'ambiente operatore.

BaseSubSwitch
Dalla superclasse BaseSupSwitch vengono ereditati gli attributi Switch, AlarmStatus, AlarmText, Delay, SupDisabled e PlcConnect.

- L'interruttore e' un oggetto Di per il pressostato. Dovrebbe essere connesso a un oggetto canale nella gerarchia dei nodi.
- AlarmStatus mostra lo stato di allarme in runtime.
- AlarmText contiene il testo di allarme per l'allarme inviato allo stato di allarme. Il testo di allarme ha il valore predefinito "Pressostato", ma puo' essere modificato in un altro testo.
  Si noti che se il testo predefinito viene mantenuto, questo verra' tradotto se viene selezionata un'altra lingua. Se viene sostituito da un altro testo, la traduzione fallira'.
- Delay e' il ritardo di intervento dell'allarme in secondi, il valore predefinito e' 0.
- SupDisabled indica che l'allarme e' disabilitato.
- PlcConnect e' un collegamento all'oggetto funzione nel codice plc.

All'oggetto principale BaseSupSwitch c'e' un oggetto funzione corrispondente, BaseSupSwitchFo, anch'esso ereditato dalla sottoclasse BasePressureSwitch.

BasePressureSwitch
BasePressureSwitch non ha attributi oltre a quelli ereditati dalle superclassi.
Le proprieta' uniche in BasePressureSwitch nel simbolo grafico, Components/BaseComponent/PressureSwitch e il grafico dell'oggetto in cui il simbolo dell'interruttore include una P per la pressione.

Configurazione
Apriamo il configuratore e creiamo l'oggetto principale, BasePressureSwitch, nella gerarchia dell'impianto. In un PlcPgm adatto inseriamo un oggetto funzione BaseSupSwitchFo e colleghiamolo all'oggetto principale con la funzione connect.
Selezioniamo l'oggetto principale e facciamo clic con Shift/Doppioclick MB1 sull'oggetto funzione. L'oggetto funzione contiene il codice per il componente, che per un BaseSubSwitch e' un allarme che viene inviato quando si perde il segnale dell'interruttore.

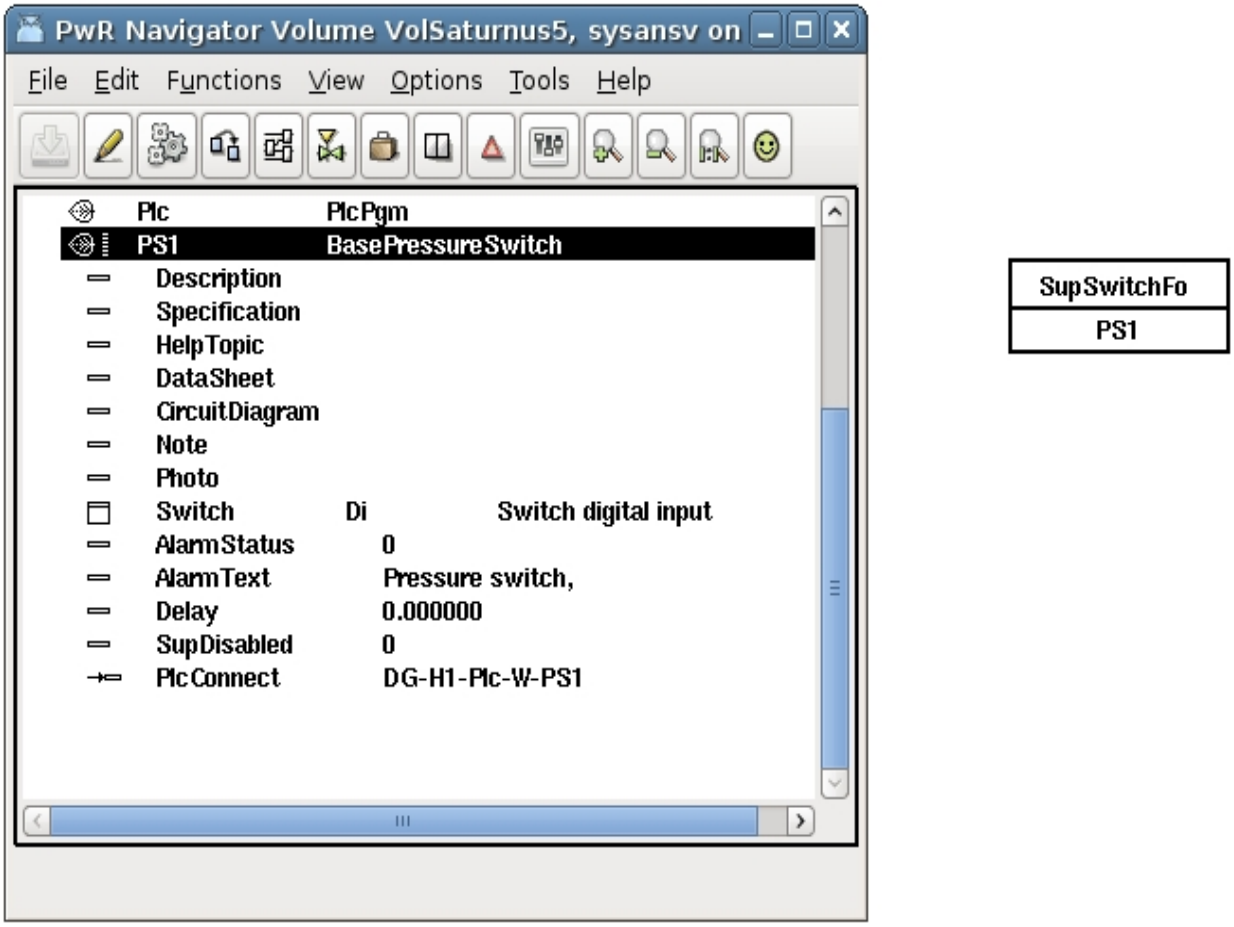


Fig Oggetto principale con oggetto funzione corrispondente

Il pressostato deve essere visualizzato in un grafico Ge. Apriamo il grafico Ge e recuperiamo il sottografo BaseComponent/SupSwitch dalla palette. Il sottografo e' adattato ad un oggetto BaseSubSwitch e tutto cio' che dobbiamo fare e' collegarlo all'oggetto principale. Selezioniamo l'oggetto principale nella gerarchia dell'impianto e facciamo clic con Shift/Doppioclick MB1 sul sottografo.

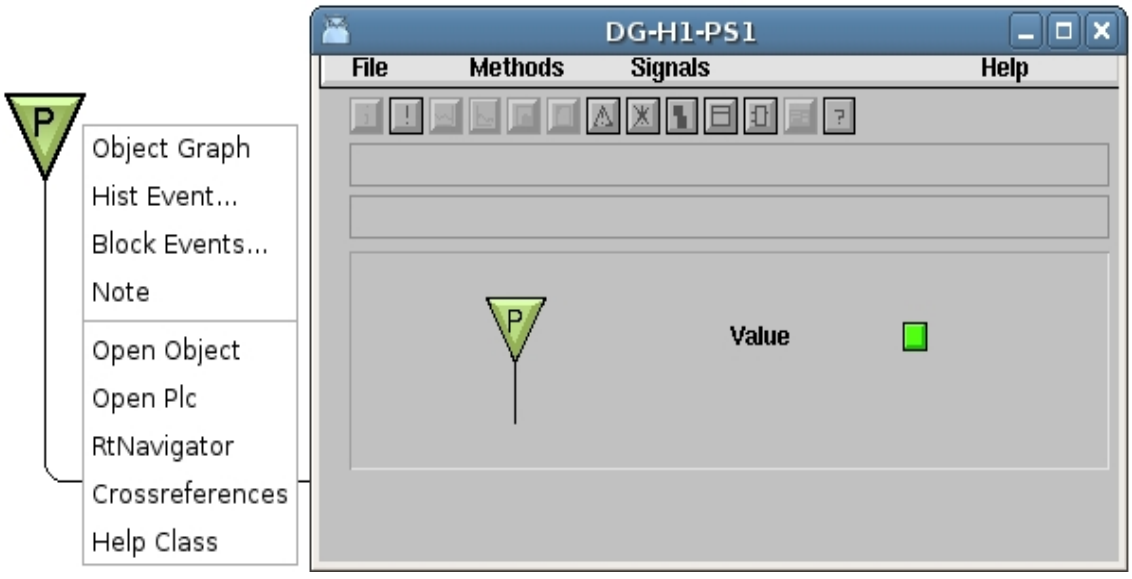


Fig Simbolo grafico, menu popup con metodi e oggetto grafico

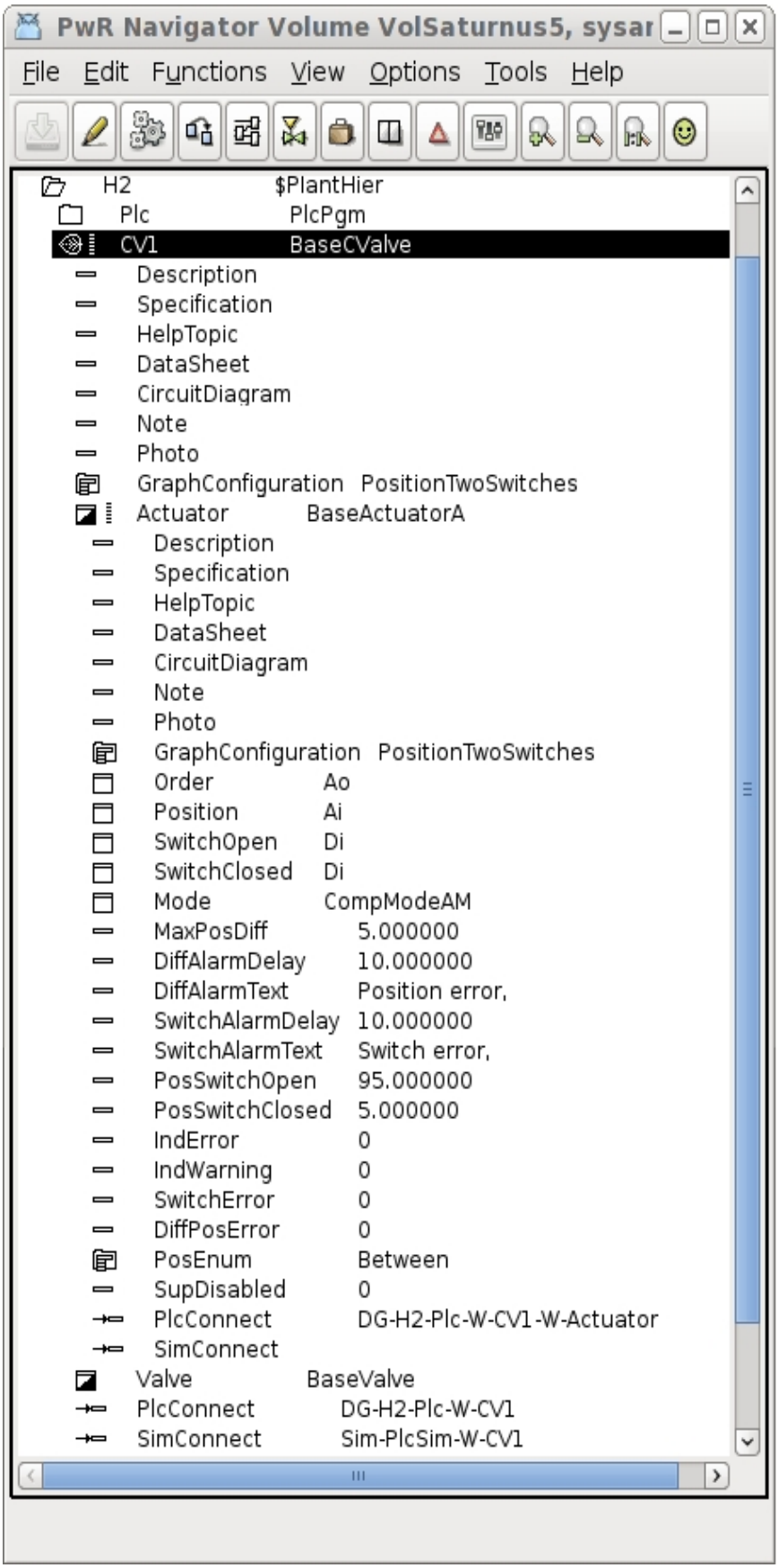
Abbiamo ora realizzato un componente funzionante. Naturalmente, possiamo anche continuare a configurare gli attributi del metodo con testi di aiuto e collegamenti a foto, schemi circuitali e schede tecniche.

Valvola di controllo

Diamo un'occhiata piu' da vicino a un componente un po 'piu' complicato, BaseCValve, che manovra una valvola di controllo. In contrasto con il pressureswitch di cui sopra, devi anche usare ConfigureComponent per configurare l'oggetto, e c'e' anche un oggetto simulato che viene utilizzato per testare il componente.

Supponiamo ora di avere una valvola di controllo, che e' controllata da un'uscita analogica, e che restituisce la posizione della valvola in un ingresso analogico.
Qui possiamo usare un BaseCValve. Ha l'uscita analogica "Order" e l'ingresso analogico "Position", cioe' i segnali che di cui abbiamo bisogno. Oltre a questi, ci sono due segnali di ingresso digitali per interruttore aperto e interruttore chiuso, ma questi possono essere disabilitati dalla configurazione.

Configurazione
Collochiamo l'oggetto principale BaseCValve nella gerarchia dell'impianto e l'oggetto funzione BaseCValveFo in un PlcPgm, e li colleghiamo insieme per mezzo della funzione Connect. Il functionobject ha un pin di input di comando che colleghiamo a un oggetto PID. Dobbiamo anche dichiarare che la nostra valvola non ha alcun interruttore, e lo facciamo attivando 'ConfigureComponent/PositionNoSwitches' nel menu popup per l'oggetto principale. Quando apriamo l'oggetto principale, e in questo l'oggetto Attuatore, troveremo un segnale di posizione, ma nessun segnale di input per gli interruttori.



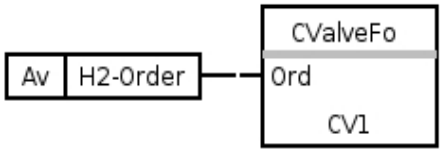


Fig Oggetto principale con oggetto funzione

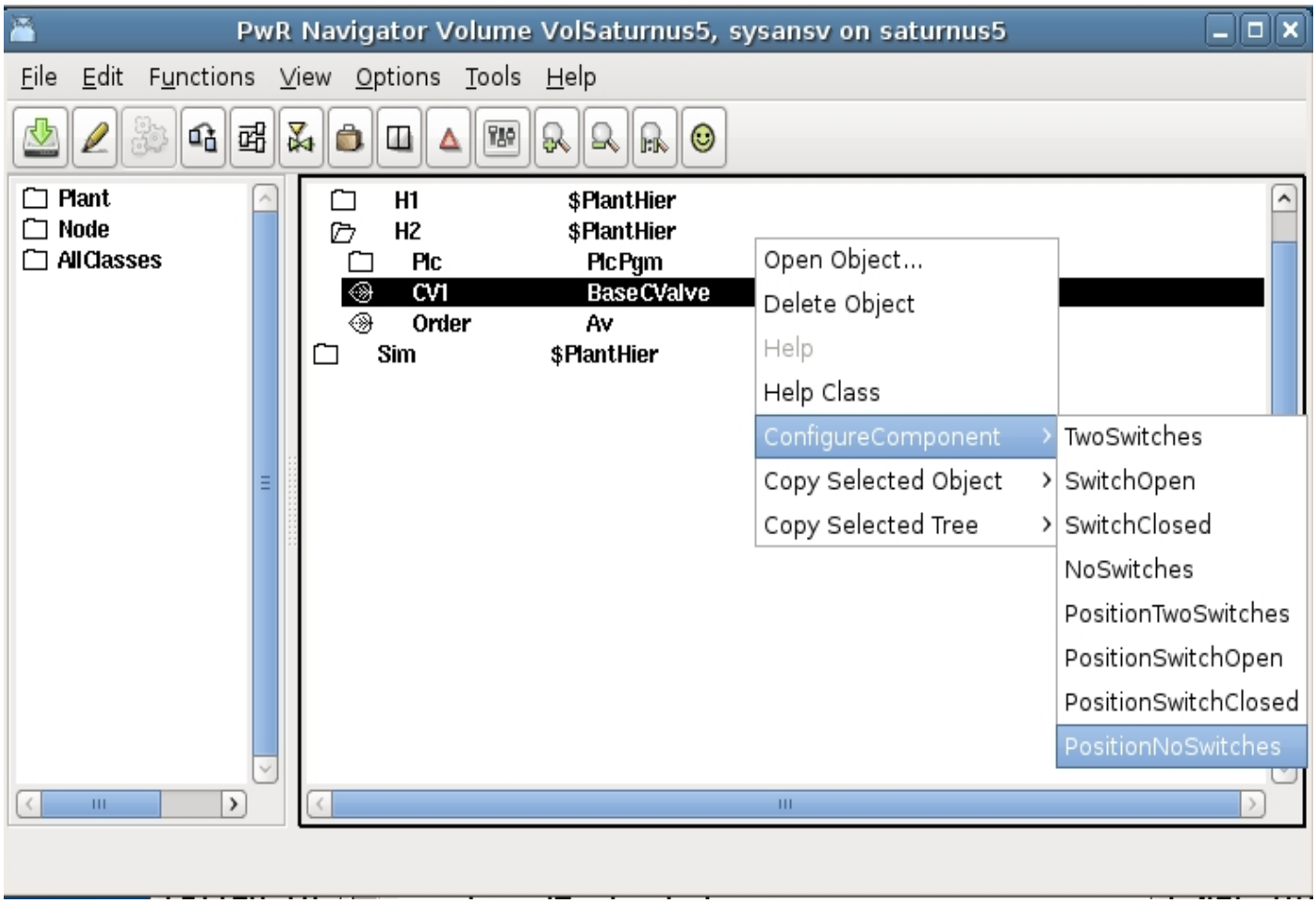


Fig Configurazione dell'oggetto principale con ConfigureComponent/PositionNoSwiches

Nell'HMI posizioniamo il sottogruppo BaseComponent/CValve in un grafico Ge e lo colleghiamo all'oggetto principale.

Vogliamo anche essere in grado di simulare la valvola, per vedere che funzioni come vogliamo. Per la simulazione c'e' l'oggetto funzione BaseCValveSim che inseriamo in un PlcPgm specifico per la simulazione, poiche' questo PlcPgm non deve essere eseguito nel sistema di produzione.
L'oggetto funzione e' connesso all'oggetto principale con la funzione Connect.

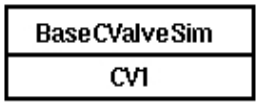


Fig Oggetto di simulazione per la valvola di controllo

La configurazione e' finita e dopo aver costruito il nodo di simulazione possiamo testare il sistema ed esaminare il risultato.

Azionamento pompa

Il prossimo esempio e' un aggregato, un azionamento pompa con un convertitore di frequenza che comunica tramite Profibus con il protocollo PPO3. Vedremo come un oggetto componente nell'aggregato, oltre al solito oggetto principale, oggetto funzione e oggetto di

simulazione, contenga anche oggetti I/O per il recupero e l'invio di dati tramite Profibus.

La classe che usiamo e' BaseFcPPO3PumpAggr e la classe di dipendenza per questa classe e'

```
Aggregate-BaseFcPPO3MotorAggr-BaseFcPPO3PumpAggr
```

Tutti gli aggregati hanno la superclasse Aggregate che corrisponde alla classe Component per i componenti. La prossima superclasse, BaseFcPPO3MotorAggr contiene tutte le funzionalita' per il controllo. La classe per le pompe BaseFcPPO3PumpAggr estende l'aggregato del motore con un oggetto pompa che rappresenta la pompa meccanica, ma questo non contiene alcun segnale o alcuna funzionalita' aggiuntiva. L'aggregato della pompa ha anche il suo oggetto grafico e il suo simbolo grafico.

Configurazione
L'oggetto principale BaseFcPPO3PumpAggr viene inserito nella gerarchia dell'impianto e l'oggetto funzione, ereditato dall'aggregazione motore, BaseFcPPO3MotorAggrFo, viene inserito in un PlcPgm e sono collegati tra loro dalla funzione Connetti.

L'oggetto principale ha non meno di 24 diverse alternative di configurazione tra cui scegliere, a seconda di quali dei componenti Fusibile, Contattore, SafetySwitch, StartLock e CircuitBreaker sono presenti nella costruzione. Nel nostro caso abbiamo solo un contattore e un convertitore di frequenza e scegliamo L'alternativa CoFc nel ConfigureComponent.

Alcuni componenti in un aggregato possono avere le proprie configurazioni. In questo caso il contattore e il motore possono essere configurati individualmente. Il nostro contattore ha due segnali, un Do per il comando e un Di per il feedback, e questo vale per la configurazione di default, cioe' non dobbiamo cambiarlo. Il motore, tuttavia, ha un interruttore di temperatura, quindi selezioniamo il componente del motore e attiviamo ConfigureComponent/TempSwitch nel menu popup.

Il prossimo passo e' connettere gli oggetti del segnale agli oggetti del canale.
Il contattore ha un Do di comando e un Di per feedback che deve essere collegato ai canali adatti nella gerarchia dei nodi. Il motore ha un interruttore termico in forma di Di che dovrebbe anche essere collegato. Il convertitore di frequenza collega quattro segnali, StatusWordSW (Ii), ActSpeed (Ai), ControlWordCW (Io) e RefSpeed (Ao). Questi segnali vengono scambiati con il convertitore di frequenza via Profibus con il protocollo PPO3.
Esiste un oggetto modulo Profibus specifico per PPO3, BaseVcPPO3PbModule, che contiene i segnali per la comunicazione PPO3. Il modulo e' configurato dal configuratore Profibus (consultare Guida al sistema I / O) ed e' collegato al componente FrequencyConverter nell'aggregato della pompa.
Poiche' l'oggetto componente e l'oggetto modulo sono adattati l'uno all'altro, non e' necessario connettere ogni segnale, ma al contrario si collega il componente al modulo. Selezionando l'oggetto modulo e attivando ConnectIo nel menu popup per il componente FrequenceConverter, viene stabilita la connessione.

Mettiamo anche il sottografo Component/BaseComponents/FccPPO3PumpAggr in un grafico di panoramica e lo colleghiamo all'oggetto principale. Inoltre, posizioniamo l'oggetto di simulazione BaseFcPPO3MotorAggrSim in uno specifico PlcPgm di simulazione e lo colleghiamo all'oggetto principale.

Modalita'

L'aggregato della pompa contiene un oggetto Mode in cui si configura da dove viene controllata la pompa. Si potranno avere le modalita' Auto, Manual o Local ovvero :

- Auto: la pompa e' controllata del programma plc.
- Manual: la pompa e' controllata dall'operatore per mezo dell'oggeto grafico.
- Local: la pompa e' controllata da una pulpito.

Nell'oggetto modalita' e' possibile configurare le modalita' applicabili alla pompa in questione.

12.1 Caso di studio sui Componenti

Processi
Il processo che controlleremo e' mostrato nella figura "Controllo di livello". L'acqua viene pompata da un bacino idrico ad un serbatoio. Il nostro compito e' controllare il livello nel serbatoio. A nostra disposizione abbiamo un sensore di livello e una valvola di controllo. Nel sistema c'e' anche un tubo di ritorno con un'elettrovalvola sempre aperta durante il controllo.

Notiamo che il serbatoio e' alto 1 metro, che si riflettera' in vari valori min e max nella configurazione.

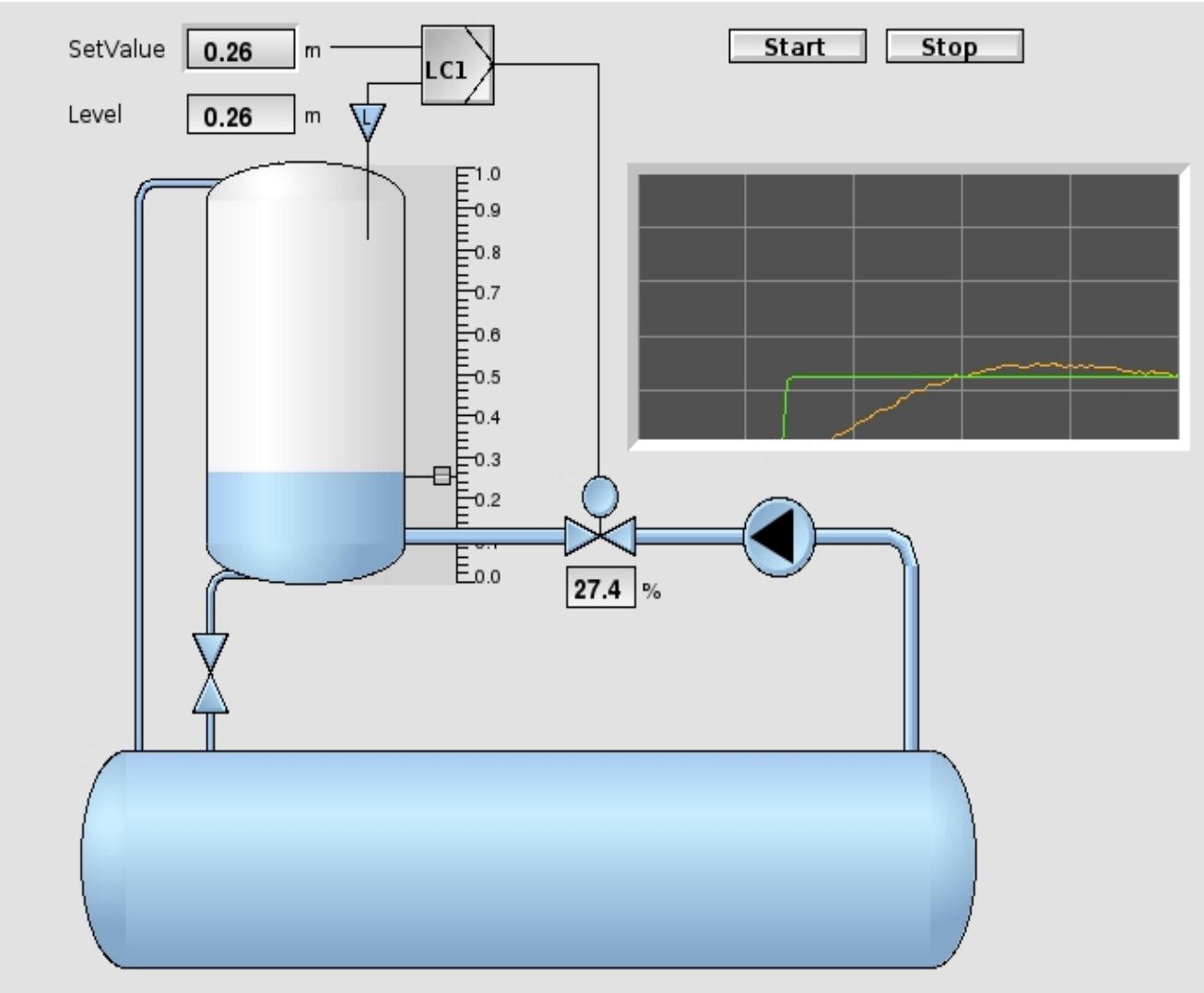


Fig Controllo di livello

Possiamo identificare i seguenti componenti e aggregati:
- una pompa azionata da un contattore, con interruttore, contattore, rele' di sovraccarico e interruttore di sicurezza
- una valvola di controllo con due interruttori di limite per aperto e chiuso

- un sensore di livello
- un'elettrovalvola con due finecorsa per valvola aperta e chiusa

Il seguente oggetto componente corrisponde ai componenti nell'impianto:

- Pompa	BasePumpAggr
- Valvola di controllo	BaseCValve
- Sensore di livello	BaseLevelSensor
- Elettrovalvola	BaseMValve

Configurazione nella gerarchia dell'impianto

I componenti vengono creati nella gerarchia LevelControl. Sotto di questa posizioniamo un PlcPgm 'Plc' che conterra' il codice per il controllo e un PlcPgm 'Simulate' che conterra' il codice necessario per la simulazione e il test del programma. Creiamo anche tre oggetti Dv per avviare, arrestare e resettare il programma di controllo, Start, Stop e Reset.

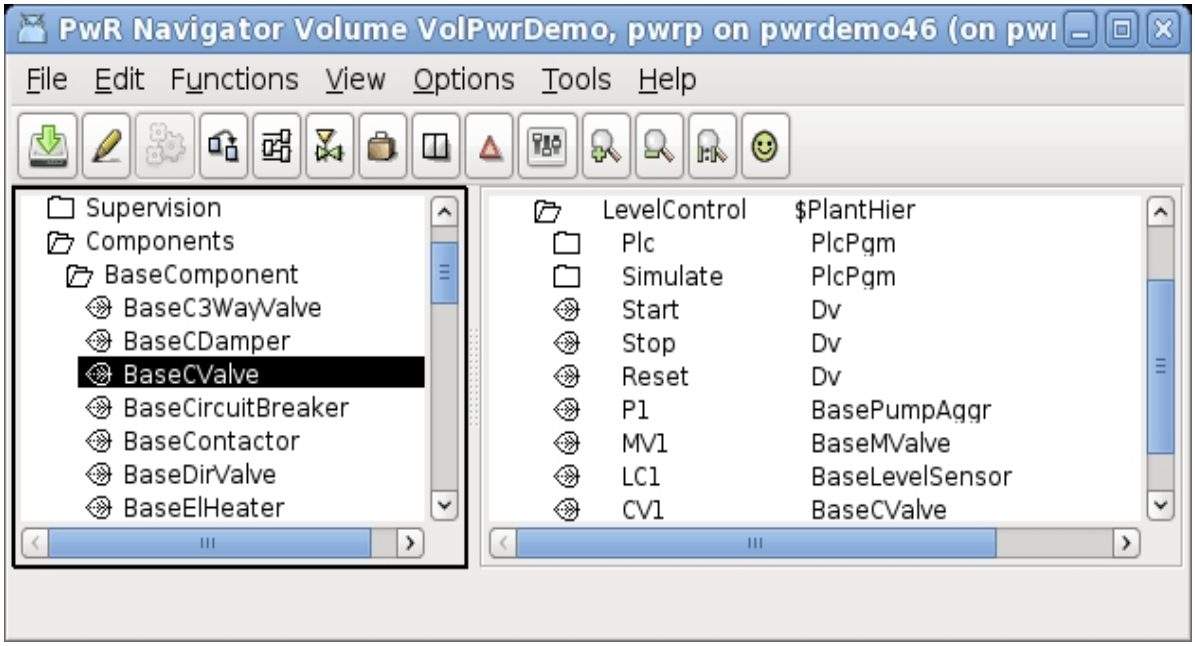


Fig Configurazione impianto

Pompa

L'azionamento della pompa e' composto da
- un interruttore con un Di
- un contattore con un Do per il comando e un Di per il feedback
- un interruttore di sicurezza con un Di
- un motore senza segnali

Per la pompa, viene creato un oggetto BasePumpAggr con il nome P1.

L'alternativa ConfigureComponent che corrisponde alla costruzione e' CbCoOrSs (CircuitBreaker, Contactor, Order, SafetySwitch) selezioniamo P1 e attiviamo questa alternativa in ConfigureComponent dal menu popup.

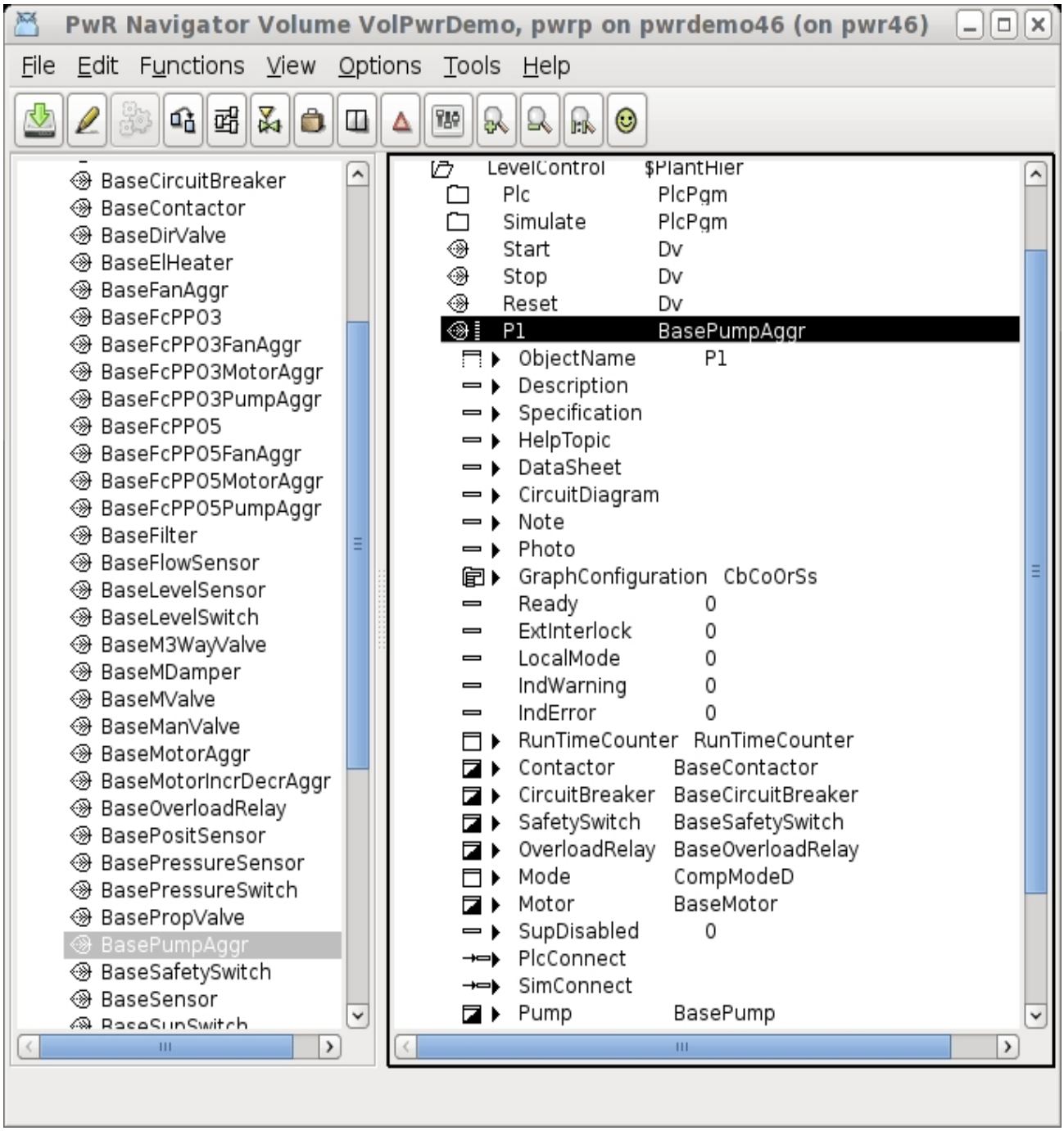


Fig Configurazione pompa

I componenti Contactor e Motor hanno le loro configurazioni, e per loro dobbiamo anche attivare ConfigureComponent. Selezioniamo Contactor e selezioniamo ConfigureComponent/OrderFeedback in quanto abbiamo un segnale per il comando e uno per il feedback. Per l'oggetto motore attiviamo ConfigureComponent/NoTempSwitchOrSensor, in quanto il motore non ha alcun segnale.

Tutti i segnali nell'aggregato della pompa devono essere collegati agli oggetti del canale nella gerarchia dei nodi. Troviamo i seguenti oggetti segnale e li colleghiamo ai canali adatti:

P1.Contactor.Order	Do comando del contattore
P1.Contactor.Feedback	Di feedback del contattore (segnale di ritorno)
P1.CircuitBreaker.NotTripped	Di interruttore non scattato

P1.SafetySwitch.On	Di interruttore di sicurezza acceso
P1.OverloadRelay.Overload	Di rele' di sovraccarico scattato

Valvola di controllo
La valvola di controllo ha un attuatore che e' controllato da un segnale di uscita analogico e interruttori di fine corsa per valvola aperta e chiusa.

Creiamo un oggetto BaseCValve con il nome CV1. L'alternativa in ConfigureComponent che corrisponde alla nostra costruzione e' TwoSwitches.

I seguenti segnali sono collegati a canali appropriati nella gerarchia dei nodi:

CV1.Actuator.Order	Ao per il comando
CV1.Actuator.SwitchOpen	Di finecorsa di posizione aperta
CV1.Actuator.SwitchClosed	Di finecorsa di posizione chiusa

Sensore di livello
Per il sensore di livello creiamo un oggetto BaseLevelSensor con il nome LC1. Questo non ha alcun metodo ConfigureComponent, ma ci sono alcune altre proprieta' da configurare.

L'oggetto sensore ha limiti di allarme per HighHigh, High, Low e Low Low e questi sono indicati negli attributi LC1.LimitHH.Limit, LC1.LimitH.Limit, LC1.LimitL.Limit e LC1.LimitLL.Limit. Impostiamo i valori limite su 0,95; 0,90; 0,10 e 0,05. Impostiamo anche il limite superiore per la presentazione del valore in LC1.Value.PresMaxLimit a 1.
Cio' influira' sull'intervallo delle barre e del grafico.

Elettrovalvola
L'elettrovalvola e' controllata da un'uscita digitale e ha una retroazione sotto forma di ingressi digitali per finecorsa aperto e finecorsa chiuso.

Per l'elettrovalvola creiamo un oggetto BaseMValve con il nome MV1.

In ConfigureComponent attiviamo TwoSwitches che corrisponde alla configurazione corrente con entrambi i finecorsa attivi.

I segnali che devono essere collegati ai canali nella nodehierarchia sono:

MV1.Order	Do per il comando di apertura della valvola.
MV1.SwitchOpen	Di per il finecorsa di aperto.
MV1.SwitchClosed	Di per il finecorsa di chiuso.

Programma PLC

Il passo successivo e' scrivere il programma plc per il controllo di livello, in cui verranno inseriti gli oggetti funzione per le componenti:

- BaseMotorAggrFo per la pompa P1
- BaseCValveFo per la valvola di controllo CV1
- BaseMValveFo per l'elettrovalvola MV1
- BaseSensorFo per il sensore di livello LC1

Creiamo gli oggetti funzione e li colleghiamo ai loro oggetti principali, selezionando l'oggetto principale e attivando Connetti nel menu popup per l'oggetto funzione.

Useremo anche un controller PID per controllare il livello nel serbatoio. Il controllore

ricevera' il valore dal sensore di livello come valore di processo e stabilira' il valore di uscita per la valvola di controllo. L'afflusso al serbatoio verra' quindi regolato per raggiungere il livello desiderato.
Il controller viene creato con functionobjects Mode e PID.

Il programma e' costruito attorno a una sequenza Grafcet con quattro passaggi.
Vedi Fig Programma Plc

1. La fase iniziale IS0 e' la posizione di riposo quando la pompa viene spenta e tutte le valvole sono chiuse.

2. Quando Start Dv e' settato da un pulsante nel grafico dell'operatore, il passaggio S0 viene attivato.
   Qui la pompa viene avviata siccome il comando Ord0 e' collegato all'ingresso startpin dell'oggetto funzionale per la pompa P1. All'avvio della pompa, viene settato l'outputpin dell'oggetto pompa e l'attivita' viene trasferita al passo successivo S1.
   Si noti che il comando Ord0 continua ad essere attivo, cioe' la pompa viene accesa fino al reset eseguito nel passo S2.

   Il valore errore dell'outputpin Err dell'oggetto pompa e' impostato nel Reset Dv.
   Il reset e' indicato come ResetObject nell'oggetto PlcPgm e tutte le sequenze nel PlcPgm verranno resettate quando viene attivato il Reset, ovvero la sequenza tornera' alla posizione iniziale e la pompa e il controllo verranno disattivati.

3. S1 e' la fase di lavoro, in cui il controller e' attivo, controllando il livello nel serbatoio.
   Finche' S1 non e' attivo, viene attivato l'input di forzatura dell'oggetto Mode e il controller ha 0 come valore out. Quando il passo e' attivo, il controller preleva il valore di processo dall'uscita dell'oggetto sensore LC1 ed il setpoint viene impostato nell'oggetto modalita' LC1_Mode.SetVal dal grafico dell'operatore. L'uscita del controller e' collegata al punto di input di comando della valvola di controllo.
   L'ordine Ord1 e' anche collegato all'ingresso del pin di comando dell'elettrovalvola, che apre la valvola.

4. La fase S1 e' attiva fino a quando Stop Dv non viene impostato da un pulsante nel grafico dell'operatore.
   Quando il passo viene abbandonato, il controller e' nuovamente forzato a zero e la valvola di controllo e' chiusa.
   Anche l'elettrovalvola viene chiusa. Il passo S2 e' attivo per un momento, resettando il comando Ord0, arrestando cosi' la pompa.
   Quindi la sequenza ritorna nella posizione di riposo IS0

La sequenza Grafcet richiede anche un oggetto di reset, il Reset Dv, da inserire nell'attributo ResetObject di PlcPgm.

L'oggetto modalita' LC1_Mode e il controller LC1_PID richiedono una configurazione aggiuntiva.
Nell'oggetto modalita'
- OpMode = Auto, per avviare il controller in modalita' automatica
- MaxSet = 1, il valore nominale massimo e' l'altezza del serbatoio, 1 m
- SetMaxShow = 1, anche l'altezza del serbatoio
- SetEngUnit = m, metri
- PidObjDid = LevelControl-Plc-W-LC1_PID, il nome dell'oggetto PID

Nell'oggetto PID
- PidAlg = PID
- Gain = 100
- IntTime = 10
- MaxGain = 200

- SetMaxShow = 1
- SetEngUnit = m
- ModeObjDid = LevelControl-Plc-W-LC1_Mode, il nome dell'oggetto modalita'

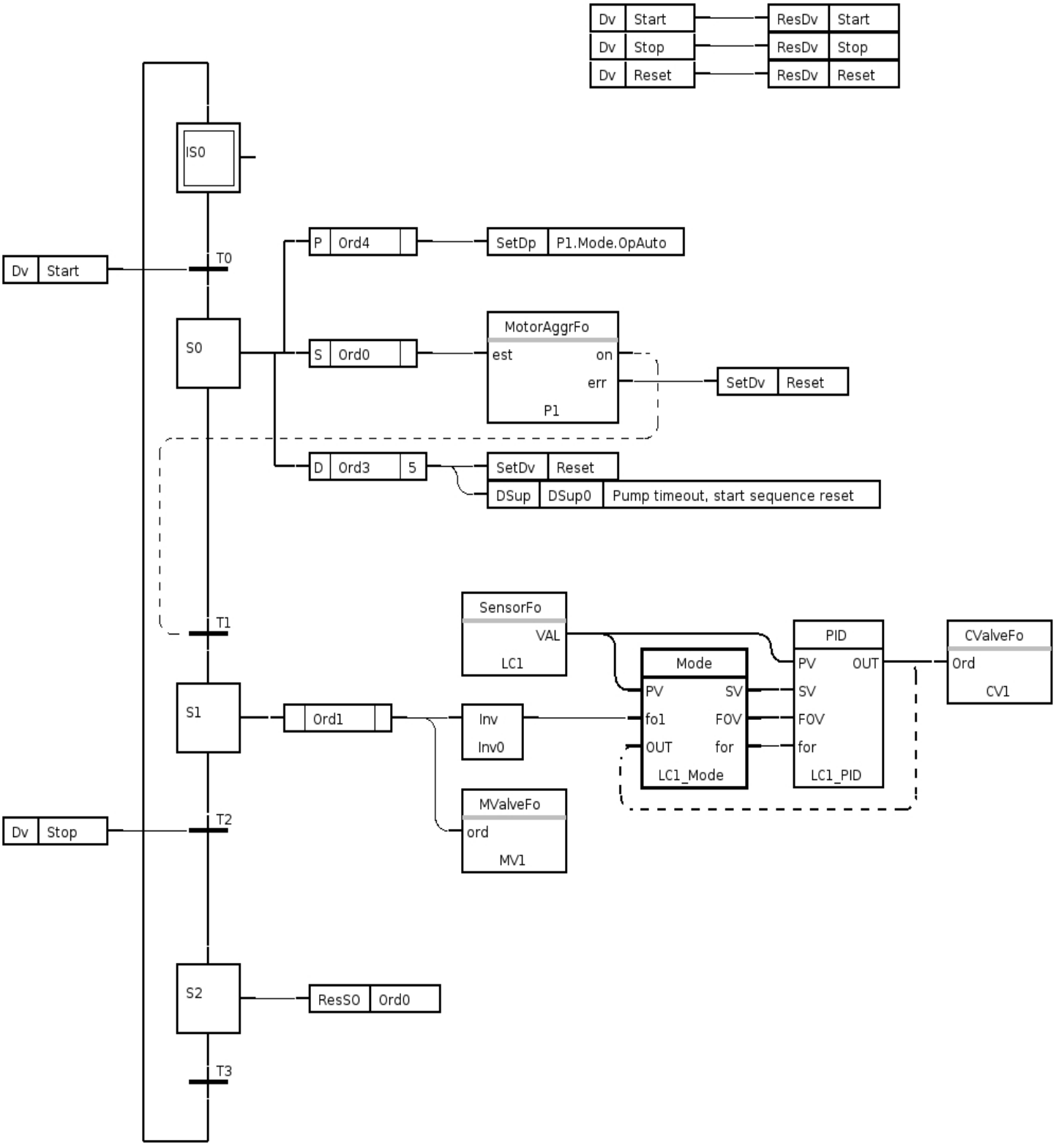


Fig Programma Plc

Programma di Simulazione del processo

Creiamo il programma di simulazione per poter testare tutte le funzioni del programma, la gestione degli allarmi e i grafici degli operatori prima della messa in servizio.
Puoi anche usarlo per scopi educativi e dimostrazione.

Il codice di simulazione viene inserito in un programma separato 'Simulate', che non dovrebbe essere eseguito nel sistema di produzione. In questo programma, vengono creati gli oggetti di simulazione per i componenti:

- BaseMotorAggrSim per la pompa P1
- BaseCValveSim per la valvola di controllo CV1
- BaseMValveSim per l'elettrovalvola MV1
- BaseSensorSim per il sensore di livello LC1

Gli oggetti funzione sono collegati ai loro oggetti principali, selezionando l'oggetto principale e attivando Connect nel menu popup dell'oggetto funzione.

Gli oggetti simulati per la pompa, la valvola di controllo e l'elettrovalvola non hanno pin di ingresso o di uscita, lavorano esclusivamente sui dati dell'oggetto principale dove leggono i segnali di uscita e impostano valori adatti nei segnali di ingresso.
Gli oggetti simulati hanno anche un oggetto grafico, dal quale e' possibile influenzare la simulazione e causare diversi errori al fine di verificare che gli errori siano gestiti in modo corretto e che l'operatore sia informato tramite i grafici di processo e gli llarmi.

L'oggetto simulato per il sensore di livello, tuttavia, ha un pin di input, e per questo dobbiamo calcolare un livello simulato nel serbatoio. Un cambiamento del livello e' determinato dal flusso in ingresso meno il flusso in uscita diviso per l'area del serbatoio. Se si assume che il flusso di ingresso sia proporzionale al valore di comando dell'uscita della valvola di controllo (CV1.Actuator.Order), e sottrae il flusso di uscita attraverso l'elettrovalvola quando questa e' aperta. La variazione del flusso viene accumulata nell'uscita OA1 del CArithm, che viene inviata all'oggetto simulato del sensore di livello LC1. In LC1, un segnale di rumore viene sommato al segnale per ottenere un comportamento piu' realistico. Cio' si ottiene impostando LC1.RandomCurve su 1 e LC1.RandomAmplitude su 0.01.

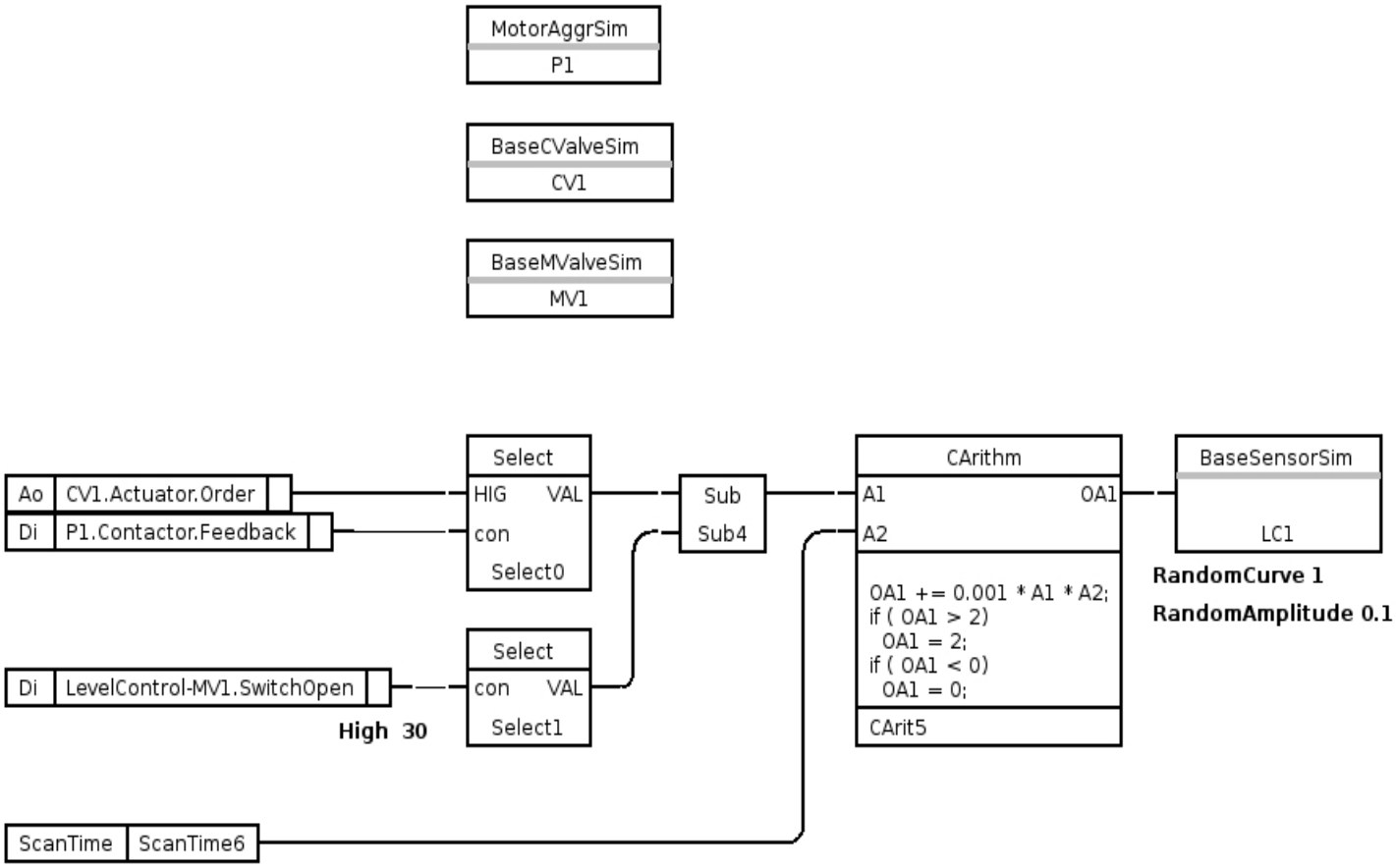


Fig Programma di simulazione

Grafica di processo.

La grafica di processo per il controllo del livello viene disegnata in Ge. Troviamo i simboli grafici per i componenti nella palette Ge:

- Component/BaseComponent/PumpAggr per la pompa
- Component/BaseComponent/CValve per la valvola di controllo
- Component/BaseComponent/MValve per l'elettrovalvola
- Component/BaseComponent/LevelSensor per il sensore di livello

I simboli hanno l'HostObject dinamico, il che significa che hanno una dinamica preprogrammata che e' connessa a diversi attributi nell'oggetto. E' sufficiente inserire il nome oggetto dell'oggetto principale in HostObject.Object oppure collegarli selezionando l'oggetto principale e fare clic con DoubleClick Ctrl /MB1 sul simbolo.

La dinamica predefinita non include l'apertura del grafico dell'oggetto quando si fa clic sul simbolo.
Aggiungiamo questa funzione aprendo l'editor degli attributi per il simbolo e aggiungendo OpenGraph in action (se non viene specificato OpenGraph.GraphObject, il grafico dell'oggetto viene aperto).

Assembliamo anche un serbatoio, da un rettangolo e due mezze ellissi, su cui impostiamo le proprieta' di riempimento e sfumatura ed anche il gruppo. Sul gruppo viene impostato il FillLevel dinamico e FillLevel.Attribute e' collegato al valore del sensore di livello,

LevelControl-LC1.Value.ActualValue.
I valori minimo e massimo per FillLevel sono impostati su 0 e 1.

A sinistra del serbatoio viene posizionato uno slider da cui e' possibile impostare il setpoint del livello.
Questo e' formato da uno Slider/SliderBackground3 e uno Slider/Slider3. Il cursore e' collegato al setpoint nell'oggetto modalita' LC1_Mode, cioe' LevelControl-Plc-W-LC1_Mode.SetVal.
I valori minimo e massimo per il cursore sono impostati su 0 e 1.

Per il setpoint, viene creato anche un campo di input 'SetValue', che e' collegato allo stesso valore del cursore precedente. Il valore di processo del livello viene visualizzato nel campo "Level", che e' collegato al valore del sensore di livello.

Il simbolo del controller viene recuperato dalla tavolozza Process/PID_Controller.
Questo simbolo ha una dinamica di default, ma vogliamo che il grafico dell'oggetto per l'oggetto modalita' sia aperto quando si fa clic sullo stesso, quindi aggiungiamo un comando all'azione come sotto riportato

```
open graph /class/instance=LevelControl-Plc-W-LC1_Mode
```

I pulsanti Start e Stop sono di tipo Buttons/SmallButton. SmallButton ha ToggleDig come azione predefinita, ma a noi servirebbe SetDig, poiche' i segnali Start e Stop vengono resettati dal programma plc.
Rimuoviamo quindi Eredita (Inherit) in Azioni per evitare ToggleDig e aggiungiamo SetDig e poi connettiamoli a Start e Stop Dv.

La curva di tendenza (grafico) viene recuperata dalla palette da Analog/Trend. I valori di processo e i valori impostati devono essere visualizzati nel grafico, vale a dire
- Trend.Attribute1 e' impostato a LevelControl-LC1.Value.ActualValue (il valore del sensore di livello).
- Trend.Attribute2 e' impostato a LevelControl-Plc-W-LC1_Mode.SetVal (il valore di impostazione dell'oggetto mode).

Infine disegniamo alcuni tubi e linee e la pagina grafica e' finita.

Valorizziamo anche gli attributi File/Graph e inseriamo le coordinate per gli angoli in alto a sinistra e in basso a destra in x0, y0 e x1, y1.
DoubleBuffered e' impostato su 1 e MB3Azione su PopupMenu.

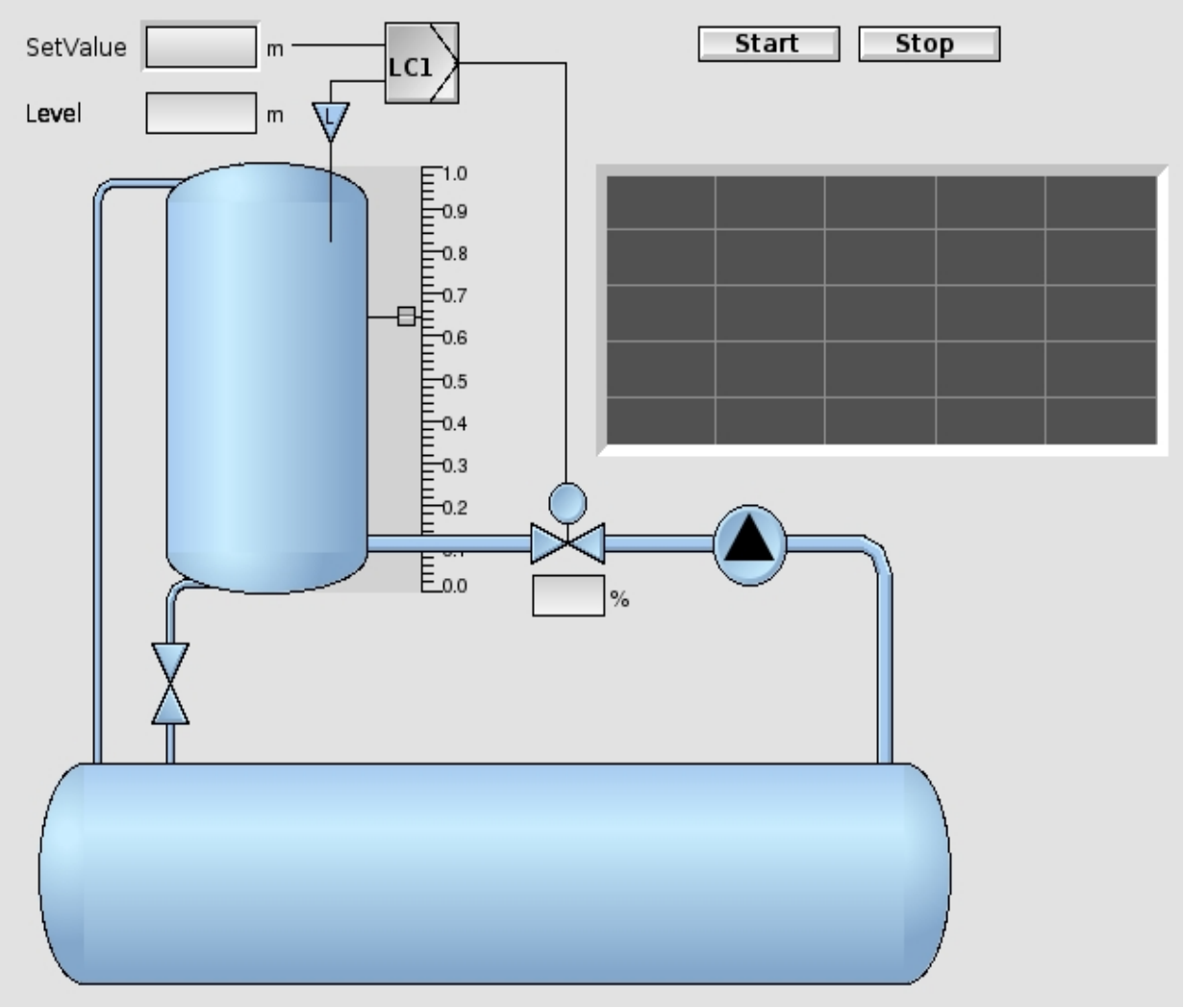


Fig Editor Grafico

Simulazione

Per vedere il risultato del nostro sforzo di programmazione, iniziamo la simulazione.

Quando apriamo il grafico, entrambe le valvole sono colorate in bianco, il che indica che sono chiuse.
La pompa non e' avviata, contrassegnata con un colore grigio e il triangolo nel simbolo della pompa non punta nella direzione del flusso. Il serbatoio e' di colore bianco, cioe' e' vuoto e il sensore di livello lampeggia in rosso quando il livello si trova al di sotto del livello di LowLow nell'oggetto del sensore di livello.

Premendo il pulsante di avvio, lasciamo la fase di riposo nella sequenza Grafcet e attiviamo il passo che avvia la pompa (S0). Quando la pompa viene avviata, viene colorata in blu e il triangolo punta nella direzione del flusso. In questa fase, anche l'elettrovalvola viene aperta e colorata in blu. Quando la pompa e' stata avviata, la sequenza passa alla fase operativa S1.

Impostiamo un valore di set con il cursore a circa 0,3 e si spera che il controller inizi a funzionare. Alla fine, sono necessari alcuni aggiustamenti dei parametri del controller e il grafico del controller viene aperto facendo clic sul simbolo del controller, che aprira' il grafico per l'oggetto Mode. In questo, facciamo clic sul pulsante

PID per aprire il grafico dell'oggetto PID.
Qui possiamo regolare il guadagno (Kp) e il tempo di integrazione (Ti).

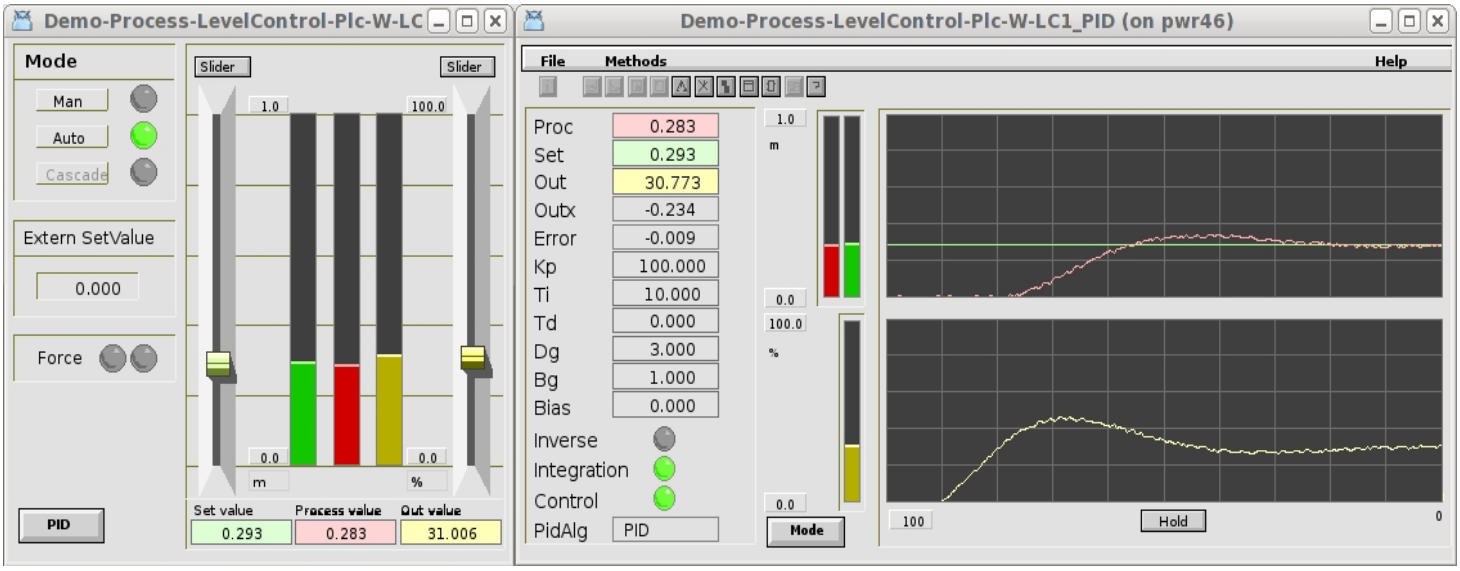


Fig Grafico degli oggetti mode e PID.

Diamo un'occhiata a cosa possiamo fare con i componenti nella pagina grafica del processo.

Sensore di livello
Se si fa clic con il tasto destro sul sensore di livello, viene aperto un menu a comparsa con i metodi definiti per il sensore. Con OpenPlc si apre la traccia del plc per l'oggetto funzione del componente. Con RtNavigator l'oggetto viene visualizzato e visualizzato nel navigatore, con Trend viene visualizzata una curva di tendenza per il livello e con OpenGraph viene aperto il grafico dell'oggetto.
Il grafico dell'oggetto puo' anche essere aperto facendo clic sul simbolo.

La parte superiore delle pagine grafiche degli oggetti, componenti e aggregati, hanno tutte un aspetto simile.
C'e' un menu in cui sotto "Methods" puoi attivare i metodi del componente. Sotto 'Signals' puoi vedere i segnali nel componente e aprire il loro grafico. Per gli aggregati puoi anche vedere i componenti e aprire il grafico degli oggetti 'Components'. C'e' anche una barra degli strumenti con pulsanti per i metodi e due campi di testo che visualizzano Descrizione e Specifiche del componente. Nella riga piu' bassa del grafico viene visualizzato il messaggio Nota se tale messaggio e' stato inserito (con il metodo Note).

Inoltre, il livello viene visualizzato come un numero e con una barra e vengono visualizzati anche i limiti di allarme. I limiti di allarme possono essere regolati con i cursori e abilitati o disabilitati dalle caselle di controllo.

Nell'angolo in alto a destra del grafico c'e' un pulsante contrassegnato da una "S".
e' visibile solo in modalita' di simulazione (cioe' IOSimulFlag nell'oggetto IOHandler e' impostato) e apre il grafico di simulazione. Dal grafico simulato, puoi influenzare il segnale simulato. Abbiamo gia' configurato Random con ampiezza 0,01, ma puoi anche aggiungere una curva sinusoidale o una curva a dente di sega al segnale.

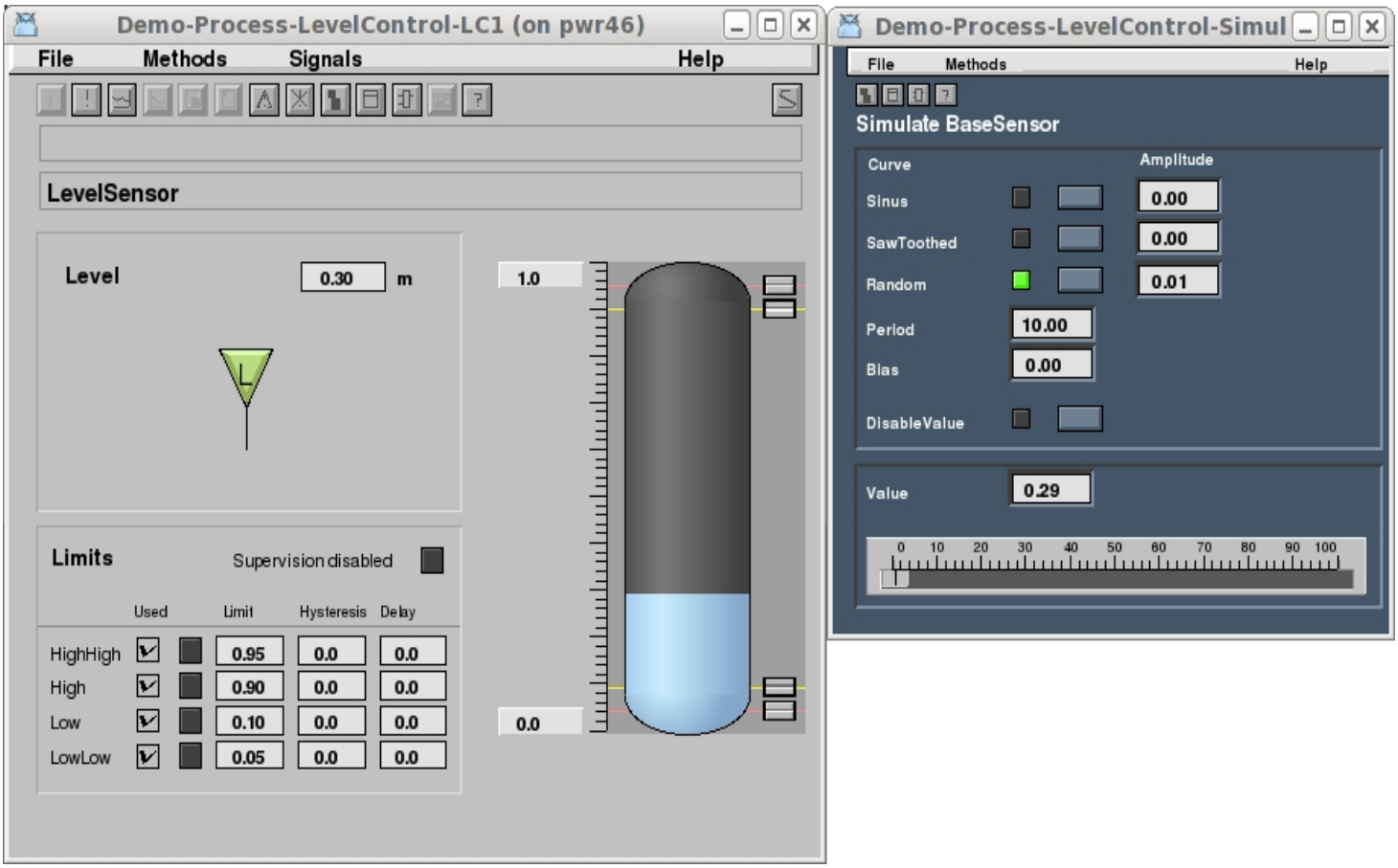


Fig Pagina grafica per il sensore di livello e finestra di simulazione

Valvola di controllo

La grafica dell'oggetto per la valvola di controllo ha indicatori per i finecorsa e mostra l'uscita di comando della valvola come una barra.

e' possibile forzare la valvola in modalita' manuale, ovvero la posizione della valvola viene ora regolata con il cursore "Manual" invece di essere comandata dal segnale di uscita del controller.
Il circuito di controllo e' ora fuori servizio e il flusso dell'acqua e' regolato dal cursore.

Dalla grafica di simulazione e' possibile ad esempio influenzare la simulazione dei finecorsa.
Se l'ordine e' 0 e l'interruttore chiuso non e' interessato, si riceve un allarme di fine corsa e un simbolo rosso lampeggiante. Durante la simulazione gli oggetti simulati impostano i valori corretti nei finecorsa , ma questo puo' essere ignorato dalla grafica di simulazione. Premendo 'Manual Control', invece, l'interruttore viene controllato dalla grafica e, azzerando il finecorsa, e' possibile controllare che la supervisione del finecora funzioni.

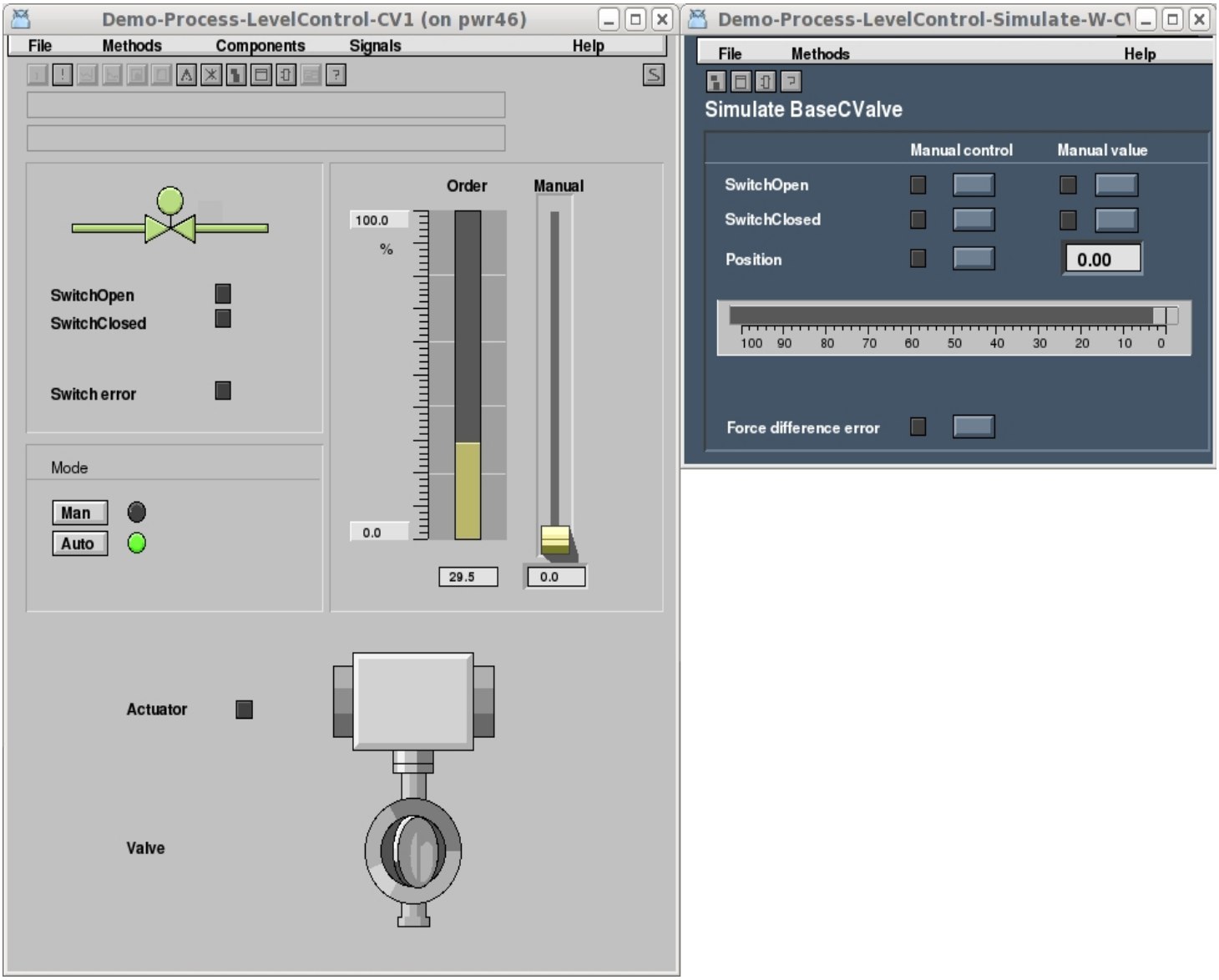


Fig Pagina grafica per la valvola di controllo e di simulazione

Elettrovalvola

La pagina grafica per l'elettrovalvola visualizza i finecorsa e il segnale di comado con gli indicatori.
La valvola passa al controllo manuale facendo clic sul pulsante Man e ora puo' essere manovrata dai pulsanti Apri e Chiudi.

Dal grafico di simulazione e' possibile, come per la valvola di controllo, influenzare la simulazione dei finecorsa e attivare un allarme per errore di commutazione.

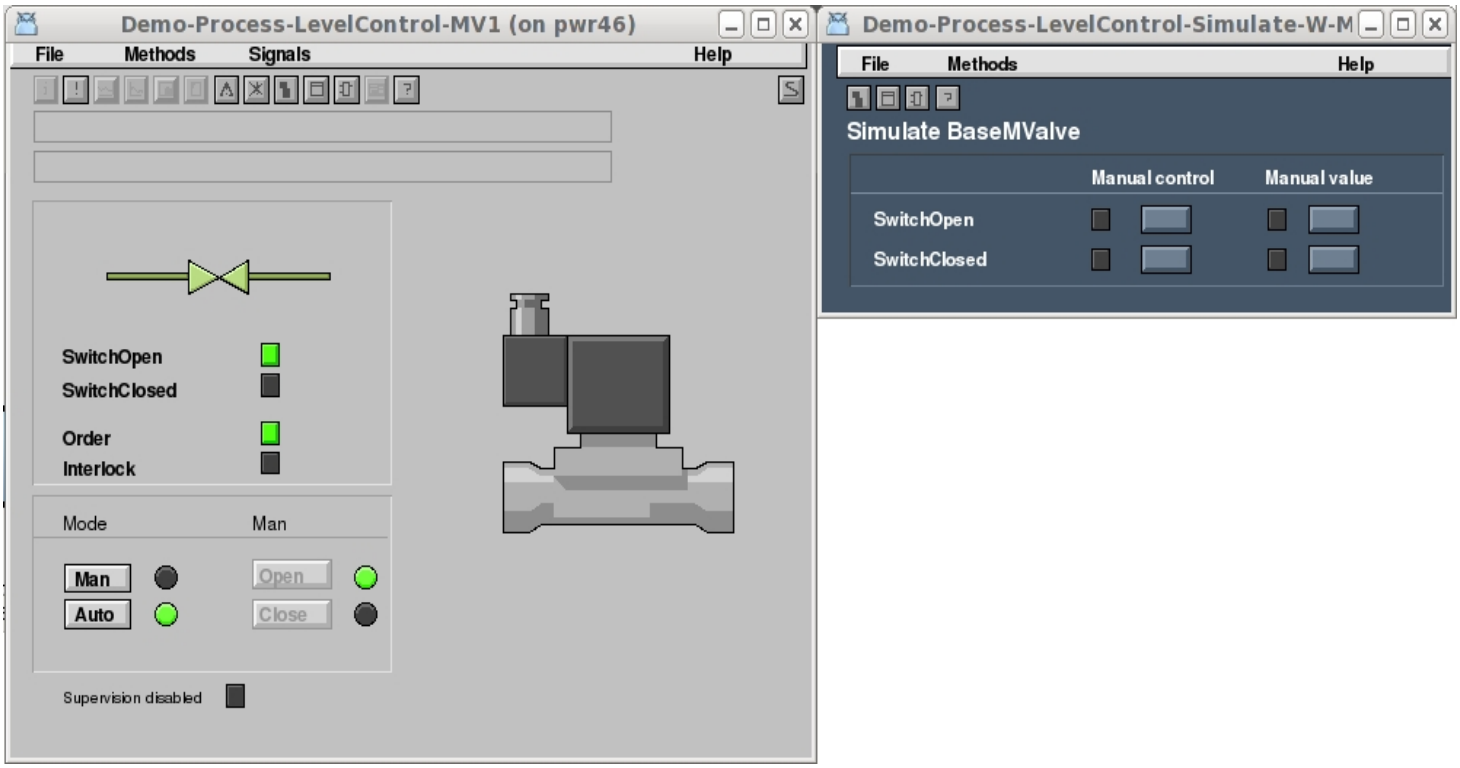


Fig Pagina grafica per elettrovalvola e simulazione

Pompa

La pagina grafica dell'oggetto pompa mostra un disegno schematico dei componenti nella pompa e anche un indicatore di stato per ciascun componente. Facendo clic su un componente, si apre la pagina grafica del componente selezionato.

Con il pulsante 'Man' e' possibile passare alla modalita' manuale e avviare e arrestare la pompa dai pulsanti Start e Stop .

Dalla pagina grafica di simulazione e' possibile simulare diversi eventi.
- 'SafetySwich on' simula che qualcuno abbia attivato l'interruttore di sicurezza.
  Questo fa spegnere la pompa e il simbolo della pompa diventa colore giallo.
  Anche il pin di uscita Err dell'oggetto funzionale e' impostato.
  Poiche' questo e' collegato al Reset Dv, la sequenza viene ripristinata e il controllo viene disattivato.
- 'CircuitBreaker tripped' simula l'intervento dell'interruttore automatico.
- 'Contacor feeback lost' simula che il contatto feedback del contattore non arrivi.
- 'OverloadRelay tripped' simula che la protezione di sovraccarico e' scattata.

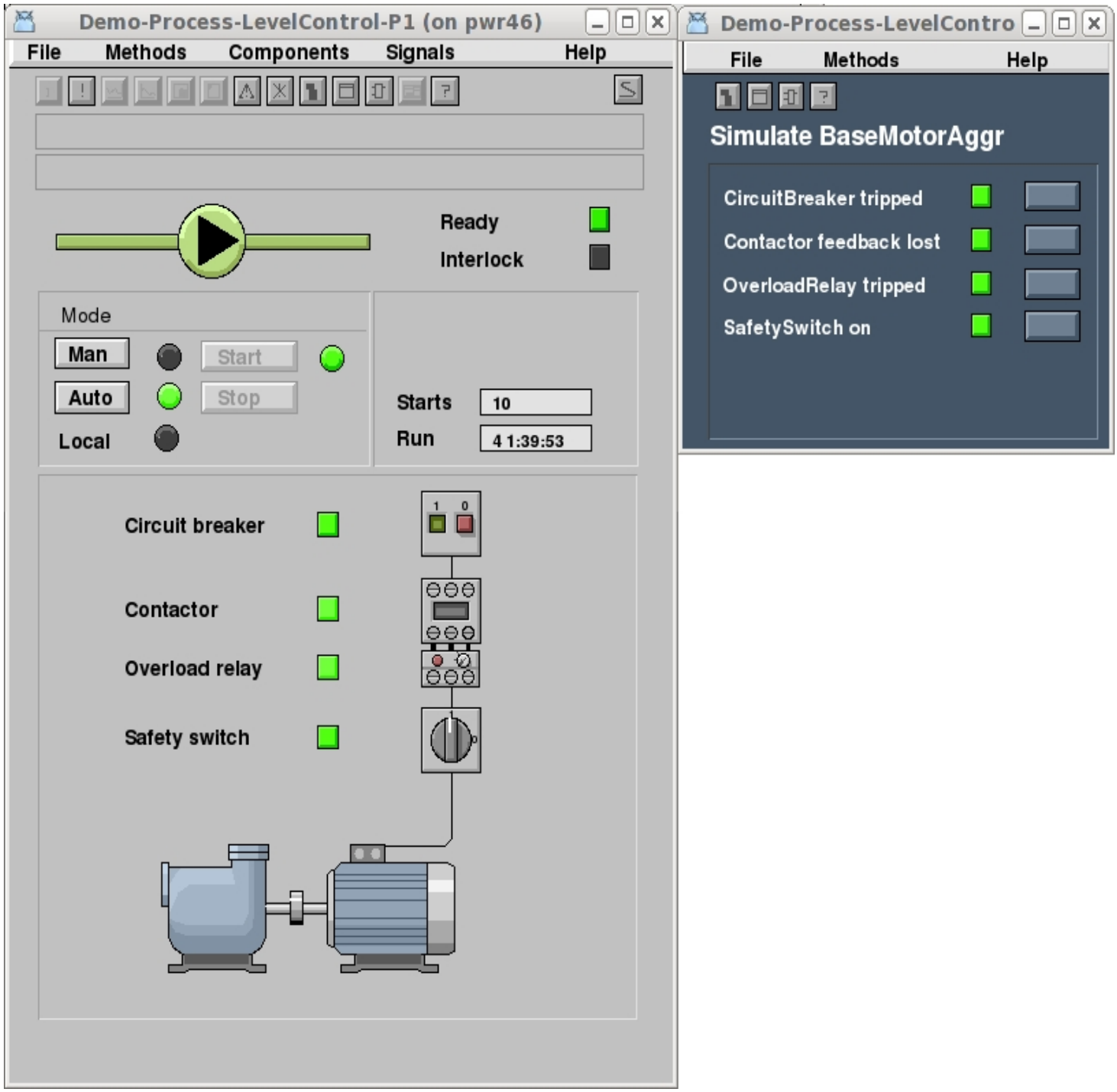


Fig Pagina grafica per la pompa e la simulazione

13 Comunicazione

13.1 Comunicazione interna

La comunicazione interna in Proview invia informazioni su volumi, oggetti, allarmi, eventi, dati cronologici, ecc. tra processi e nodi. Esistono tre diversi protocolli, per la gestione degli allarmi, la gestione della rete e i dati storici, tutti basati su Qcom.

Qcom

Qcom e' un bus di comunicazione che invia messaggi in una coda e collega tra loro i vari processi. I nodi che devono comunicare tra di loro devono condividere il bus QCom e un nodo puo' solo collegarsi ad un solo bus. Il bus e' configurato nell'oggetto BusConfig dichiarando l'identita' del bus che e' un valore compreso tra 1 e 999.

Bisogna anche configurare quali nodi devono comunicare tra loro. Con la configurazione predefinita, tutti i nodi nello stesso progetto comunicheranno e i nodi aggiuntivi di altri progetti verranno configurati con oggetti FriendNodeConfig. Ad esempio questi possono essere le stazioni di processo montate dalle stazioni operatore.

Tutti i messaggi Qcom vengono inviati con richiesta di conferma. Se la conferma non arriva, il messaggio viene inviato nuovamente con timeout doppio. Dopo un certo numero di rinvii senza risposta, il collegamento viene considerato inattivo. A seconda del tipo di rete e della velocita' di trasmissione, potrebbe essere necessario regolare i timeout. Questo puo' essere fatto con gli attributi ResendTime negli oggetti NodeConfig e FriendNodeConfig.

Vedere il documento QCOM per ulteriori informazioni su Qcom.

Gestore di rete

Il gestore di rete invia informazioni su volumi e oggetti tra i nodi, ad esempio i volumi in un nodo, il genitore, i figli oi fratelli di un oggetto o il contenuto di un oggetto o attributo. Anche gli abbonamenti vengono impostati tramite il gestore di rete, ovvero i dati vengono inviati ciclicamente da un nodo a un altro, di solito dalle stazioni di processo alle stazioni operatore.
I processi per la gestione della rete sono rt_neth, rt_neth_acp e rt_tmon.

Gestore di eventi

Il gestore eventi analizza tutti gli oggetti di supervisione in un nodo e invia allarmi ed eventi alle unita' esterne, ad esempio agli elenchi di allarmi e eventi nell'ambiente dell'operatore.
Mandano riconoscimenti indietro al gestore di eventi. Esattamente quali allarmi e eventi inviati a un'unita' esterna sono configurati nell'elenco di selezione. L'elenco di selezione

per l'ambiente operatore si trova nell'oggetto OpPlace. Vengono inviati solo eventi e allarmi sotto le gerarchie indicate.
Il processo di gestione degli eventi, rt_emon, e' configurato con un oggetto MessageHandler.

Archiviazione di dati storici

L'archiviazione dei dati storici significa che i dati vengono inviati dalle stazioni di processo e memorizzati in un database in un server di archiviazione. Il processo del server rt_sevhistmon nella stazione di processo analizza tutti gli attributi contrassegnati per la memorizzazione, recupera i valori corrispondenti e li invia al processo sev_server che li archivia nel database. Quando le curve dei dati storici devono essere visualizzate, viene inviata una richiesta dall'ambiente operatore e una selezione adeguata di punti viene recuperata dal database e inviata all'ambiente operatore.
Vedi il capitolo Memorizzazione dei dati

Comunicazione Web e app

L'interfaccia web e l'applicazione Andriod recuperano le informazioni dal database in tempo reale di Proview tramite i processi del server rt_webmon, rt_webmonmh e rt_webmonelog.
Questa comunicazione e' configurata dall'oggetto WebHandler.

Server di stato

Il monitor di runtime e il centro di supervisione prelevano le informazioni dal server di stato.
Questa comunicazione e' basata su http e soap ed e' configurata con l'oggetto StatusServerConfig.

13.2 Remote

Il concetto di Remote in Proview e' un modo per standardizzare i metodi di comunicazione con altri sistemi.
Descrive un numero di programmi di trasporto e oggetti Proview utilizzati per implementare una varieta' di protocolli di comunicazione diversi e per gestire i messaggi in entrata e in uscita.
Remote e' progettato per utilizzare diversi protocolli di trasporto come TCP/IP o BEA Message Queue e diversi supporti hardware come Ethernet o linee seriali.
Lo scopo principale di Remote e' fornire al programmatore un'interfaccia per la comunicazione.

13.2.1 Introduzione

Esistono classi e oggetti diversi utilizzati per gestire la comunicazione.

RemoteConfig

Necessario per qualsiasi comunicazione remota.
Senza questo non viene avviato alcun remotehandler.

Posiziona un oggetto nella gerarchia dei nodi.

RemNode

Definisce un collegamento di qualche tipo ad un nodo remoto su un protocollo specifico.
Sono supportati diversi protocolli ed esiste una classe specifica per ciascun protocollo.
I protocolli supportati sono:

TCP/IP
UDP/IP
MQ (Coda messaggi BEA)
ALCM (un vecchio protocollo digitale, supportato per ragioni storiche)
Serial
Modbus/RTU Seriale
3964R (protocollo seriale Siemens)
Webspere MQ

La configurazione di ciascun protocollo e' descritta piu' avanti.
Posiziona oggetti RemNode sotto l'oggetto RemoteConfig.

RemTrans

Classe generica che definisce un messaggio specifico verso o da un nodo remoto specifico su un protocollo specifico. Dovrebbe essere posizionato sotto l'oggetto RemNode.
La dimensione di un messaggio da inviare e' specificata nell'oggetto RemTrans. I dati da inviare risiedono tuttavia in un buffer che e' configurato come un figlio per l'oggetto remtrans. Quando un messaggio deve essere inviato, i dati della lunghezza specificata nell'oggetto remtrans vengono recuperati dal buffer.
A volte tuttavia un'intestazione di qualche lunghezza viene aggiunta al messaggio.

Buffer

Definizione dell'area di invio e/o ricezione dati per ciascun messaggio. Esiste in diverse dimensioni.
Dovrebbe essere collocato sotto un oggetto RemTrans. Il buffer deve essere almeno della dimensione del messaggio che verra' ricevuto o inviato. Se la dimensione non e' sufficiente, il messaggio verra' tagliato alla fine.

RemTransSend

Oggetto funzione utilizzato in un programma plc per l'invio di messaggi.

RemTransRcv

Oggetto funzione utilizzato in un programma plc per ricevere messaggi.

Registrazione delle transazioni

Le transazioni remote possono essere registrate su file di testo. L'estensione della registrazione e' configurata nell'attributo LoggLevel dell'oggetto RemTrans.
Il file di registro e' configurato con un oggetto LoggConfig.

13.2.2 Protocolli

Quale protocollo utilizzare e' definito dal tipo di oggetto-Remnode che si configura.
Per ogni oggetto Remnode configurato viene avviato un lavoro di trasporto. Cioe', un programma per gestire il protocollo specifico viene avviato come processo. Questo

processo gestira' tutti gli oggetti RemTrans che sono configurati come figli sull'oggetto Remnode.

13.2.2.1 UDP

UDP usa la comunicazione socket senza connessione (datagramma), rispetto a TCP, che e' un protocollo connesso.
Nell'oggetto RemnodeUDP si specifica il nome e l'indirizzo IP del nodo da comunicare, nonche' i numeri di porta per entrambe le estremita'. Il numero di porta locale deve essere univoco sul nodo per non entrare in conflitto con altre comunicazioni.

Come impostazione predefinita, tutti i messaggi vengono inviati con un'intestazione speciale che non e' inclusa nel buffer dei dati dell'utente.
Questa intestazione viene aggiunta all'inizio del messaggio. Lo scopo dell'intestazione e' fornire informazioni sul messaggio inviato. Cio' aiuta a identificare quale tipo di messaggio viene ricevuto e in quale buffer verranno decompressi i dati.
L'intestazione assomiglia a:

```
char       RemId1;  /* STX (Hex 02) */
char       RemId2;  /* ETB (Hex 0F) nel messaggio di dati senza conferma
                       ENQ (Hex 05) nel messaggio di dati con conferma
                       ACK (Hex 06) nel messaggio di conferma */
short int  Length;  /* Numero di byte nel messaggio che include questa
                       intestazione */
short int  MessId1; /* Identita' del messaggio parte 1 */
short int  MessId2; /* Identita' del messaggio parte 2 */
```

Tutti gli interi nell'intestazione saranno inviati come valori big endian, il che significa il byte piu' significativo e' il primo nel datagramma. Sara' responsabilita' dell'utente la conversione dei dati , se vuole che gli interi siano inviati con big endian.
Intel (x86), VAX e Alpha utilizzano tutti little endian! Per inviare messaggi senza intestazioni, l'attributo DisableHeader deve essere impostato su TRUE. Quando si comunica tra due sistemi Proview, l'intestazione deve essere mantenuta. MessId1 e MessId2 sono recuperati dagli attributi RemTrans.Address [0] e RemTrans.Address [1].
Attraverso l'intestazione e' anche possibile richiedere il riconoscimento di un messaggio inviato. Se non ci sono conferme, il messaggio verra' inviato nuovamente con una ciclicita' specificata dall'attributo RetransmitTime nell'oggetto remtrans.

Poiche' UDP / IP e' un protocollo senza connessione, esiste la possibilita' di guardare la c onnessione utilizzando messaggi keepalive.
Questa funzionalita' viene impostata tramite l'attributo UseKeepalive.

Invio di messaggi

Il trasporto inviera' un messaggio alla porta remota, composto da header + data.
MessId nell'intestazione e' preso da RemTrans.Address [0,1], byte commutato per inviare come big endian.
Se MaxBuffer in remtrans-object> 0, il messaggio viene inviato con tipo "want acknowledge" e viene memorizzato nella coda di ritrasmissione per il remnode. Quando viene ricevuto un messaggio di conferma corrispondente, il messaggio viene cancellato dalla coda di ritrasmissione. Questo viene fatto automaticamente dal processo di trasporto.

Ricezione di messaggi

Quando riceviamo un buffer, prima controlliamo l'intestazione per vedere che questo sia un messaggio RemTrans corretto.
Quindi cerchiamo RemTrans.Address[0,1] che corrisponde al MessId byte-switched.

Se l'oggetto dati per questo messaggio e' abbastanza grande da contenere il messaggio, il messaggio verra' archiviato e verra' impostato il flag DataValid.
Se il Remnode e' contrassegnato per essere utilizzato senza intestazione
(set attributo DisableHeader), verra' ricercato un RemTrans contrassegnato come un ricevente remtrans con un buffer sufficientemente grande.

13.2.2.2 TCP

RemnodeTCP e' configurato in modo molto simile a RemnodeUDP.
La grande (unica) differenza e' che TCP e' un protocollo connesso che agisce in modo client/server.
Pertanto, e' necessario connettersi a un socket remoto (comportarsi come un client) o attendere una connessione (agire come un server).
Quandosi agisce come un server, verra' accettato un solo client.
L'attributo ConnectionMode nell'oggetto remnode definisce se sei un client o un server.
Impostarlo su zero (predefinito) significa che si e' client e impostarlo su uno
significa server.

13.2.2.3 MQ

RemnodeMQ e' un trasporto per l'invio di messaggi su BEA Message Queue (BMQ).
Richiede l'installazione di BEA Message Queue sul nodo. Questa coda di messaggi e' utile per il recapito sicuro di messaggi a un nodo remoto anche se non e' attivo al momento dell'invio del messaggio.
Viceversa, i messaggi verranno consegnati in modo sicuro.

Questa documentazione si aspetta che tu abbia una conoscenza di base sulla coda messaggi BEA. La comunicazione viene eseguito su un bus specifico.
Ogni nodo ha un numero di gruppo e puo' comunicare solo ad altri gruppi sullo stesso bus.
Su ciascun gruppo possono essere configurate diverse code.

Per poter avviare questo trasporto, ovviamente, e' necessario che il software Message Queue sia in esecuzione.
Hai anche bisogno di impostare alcune variabili d'ambiente.
Queste sono:

```
DMQ_BUS_ID
DMQ_GROUP_ID
DMQ_GROUPNAME
```

Nell'oggetto RemnodeMQ nell'attributo MyQueue si configura su quale coda BMQ volgiamo ricevere i messaggi .
Bisogna inoltre configurare il numero di gruppo e la coda dei nodi remoti da impostare negli attributi TargetGroup e TargetQueue.

Invio di messaggi

Analogamente a UDP e TCP-trasporti RemTrans.Address[0,1] sono utilizzati per identificare il messaggio.
Address[0] rappresenta la classe messaggio e Address[1] rappresenta il tipo di messaggio (secondo la nomenclatura BMQ).
Address[2,3] viene utilizzato per definire quale tipo di modalita' di consegna (Address[2]) deve essere utilizzato e quale azione deve essere intrapresa quando un messaggio non puo' essere consegnato (Address[3]).

Le possibili modalita' di consegna sono:

```
PDEL_MODE_WF_SAF 25
```

```
PDEL_MODE_WF_DQF 26
PDEL_MODE_WF_NET 27
PDEL_MODE_WF_RCM 28
PDEL_MODE_WF_MEM 29
PDEL_MODE_AK_SAF 30
PDEL_MODE_AK_DQF 31
PDEL_MODE_AK_NET 32
PDEL_MODE_AK_RCM 33
PDEL_MODE_AK_MEM 34
PDEL_MODE_NN_SAF 35
PDEL_MODE_NN_DQF 36
PDEL_MODE_NN_NET 37
PDEL_MODE_NN_RCM 38
PDEL_MODE_NN_MEM 39
PDEL_MODE_WF_DEQ 40
PDEL_MODE_AK_DEQ 41
PDEL_MODE_WF_CONF 42
PDEL_MODE_AK_CONF 43
PDEL_MODE_WF_ACK 44
PDEL_MODE_AK_ACK 45
```

e le possibili azioni sono:

PDEL_UMA_RTS 1
PDEL_UMA_DLJ 2
PDEL_UMA_DLQ 3
PDEL_UMA_SAF 4
PDEL_UMA_DISC 5
PDEL_UMA_DISCL 6

Non tutte le combinazioni sono possibili (consultare la documentazione della coda dei messaggi BEA per ulteriori informazioni).

Le combinazioni consigliate sono,

per la consegna sicura del messaggio
Address[2] = 26
Address[3] = 4

e per scartare il messaggio se non puo' essere consegnato
Address[2] = 39
Address[3] = 5

Se Address[2,3] sono entrambi impostati su zero, verra' utilizzata un'impostazione predefinita. L'impostazione predefinita e' scartare il messaggio se non puo' essere consegnato.

Ricezione di messaggi

Address[0,1] sono usati per identificare il messaggio. Address[0] rappresenta la classe del messaggio e Address[1] rappresenta il tipo di messaggio (secondo la nomenclatura BMQ).

13.2.2.4 Seriale

RemoteSerial e' un tentativo di generalizzare l'uso di un semplice protocollo di comunicazione seriale.
E' utile quando abbiamo un invio unidirezionale di messaggi da alcune apparecchiature al

sistema di controllo.
E' possibile specificare fino a otto caratteri di terminazione nell'attributo TermChar [0-7]. Questi caratteri di terminazione vengono utilizzati per rilevare la fine di un messaggio ricevuto (se un carattere corrisponde a uno dei caratteri di terminazione).
Specificare anche le impostazioni per il collegamento seriale, ovvero DevName, Speed, Parity (0 = none, 1 = dispari, 2 = pari), StopBits e DataBits.
Per esempio potrebbe essere /dev/ttyS0, 9600, 0, 1, 8.

13.2.2.5 3964-R

3964R e' un protocollo di comunicazione seriale semplice sviluppato da Siemens.

Si specificano le impostazioni per il collegamento seriale come con RemnodeSerial, tranne per il fatto che non ci sono bit di stp da specificare. e' inoltre necessario specificare il timeout del carattere (il tempo massimo tra i caratteri ricevuti) nell'attributo CharTimeout.
L'attributo AckTimeout specifica il tempo di attesa per una risposta.

I messaggi verranno inviati direttamente senza alcuna intestazione. ACK, NAK, DLE e BCC sono gestiti secondo il protocollo 3964R.

Ricezione di messaggi

Puo' esserci un solo oggetto RemTrans per i messaggi di ricezione a causa della mancanza di intestazione in questo protocollo.
Ogni messaggio ricevuto verra' inserito nel primo oggetto RemTrans trovato sotto l'oggetto RemNode.
Se l'oggetto dati per questo messaggio e' abbastanza grande da contenere il messaggio, il messaggio verra' archiviato e verra' impostato il flag DataValid.

Invio di messaggi

Il trasporto inviera' un messaggio 3964R alla seriale senza aggiungere alcuna intestazione.
Se non si ha contatto con l'altro nodo, il messaggio verra' memorizzato nel buffer se ci sono ancora buffer liberi per questo messaggio.

13.2.2.6 Modbus seriale

Il formato di MODBUS implementato e' RTU. Per l'identificazione dei messaggi utilizziamo i campi noti come indirizzo slave e codice funzione nell'intestazione MODBUS.
RemTrans.Address [0] e [1] definiscono questi campi nell'ambiente Proview.
Vedere i documenti delle specifiche MODBUS per ulteriori informazioni.

Modbus Serial non e' ancora implementato come sistema I / O in Proview.
E' necessario configurare personalmente i messaggi utilizzando RemnodeModbus e specificando RemTrans-object per le varie operazioni che si desidera eseguire.
Modbus funziona in modalita' richiesta/risposta, quindi per ogni operazione che si desidera eseguire si specifica un oggetto RemTrans per l'invio e uno per la ricezione.
Ad eccezione dell'intestazione Modbus del messaggio e della gestione checksum, e' necessario specificare il contenuto del messaggio da inviare nel buffer di invio.
Allo stesso modo e' necessario decodificare da se' il contenuto di un messaggio ricevuto.

Modbus TCP e' implementato come sistema I/O in Proview. Vedere ulteriori informazioni a riguardo nel documento "Guida ai sistemi I/O".
Con Modbus TCP non ti devi preoccupare della codifica dei messaggi.

Ricezione di messaggi

Quando riceviamo un buffer, verifichiamo che RemTrans.Address[0] e [1] corrispondano ai campi indirizzo slave e codice funzione nell'intestazione del messaggio.

Se l'oggetto dati per questo messaggio e' abbastanza grande da contenere il messaggio, il messaggio verra' archiviato e verra' impostato il flag DataValid.

Invio di messaggi

I messaggi verranno inviati utilizzando i contenuti di RemTrans.Address[0] e [1] come indirizzo slave e codice funzione nell'intestazione del messaggio.

13.2.2.7 Websphear MQ

RemnodeWMQ e' un trasporto per l'invio di messaggi su WebSphear Message Queue (WMQ). Richiede l'installazione di Websphear MQ sul proprio nodo.

RemnodeWMQ configura le comunicazioni attraverso una coda messaggi usando Websphere MQ come client.
Tutto cio' che riguarda Server, canale, gestore code e code per la connessione e' configurato nell'oggetto RemnodeWMQ.

Le configurazioni per i diversi messaggi sono configurate nell'oggetto RemtTrans che definisce ciascun messaggio in entrata e in uscita.
Per ciascun messaggio RemTrans e' possibile configurare quanto segue:

TransName	una stringa che definisce il MsgId (identita' del messaggio del messaggio).
Address[0]	definisce se il messaggio deve essere inviato come messaggio permanente o no. 1 significa che il messaggio verra' inviato come messaggio persistente. 0 no.

13.2.3 Un esempio

Per mostrare come lavorare con le classi che sono brevemente descritte sopra inizieremo con un piccolo esempio. Le classi sono descritte in maggior dettaglio di seguito.

Nel nostro esempio abbiamo un sistema Proview che comunica con un altro nodo via UDP/IP. Invieremo alcuni messaggi in entrambe le direzioni. Il mio nodo e' chiamato 'dumle' e il nodo remoto e' chiamato 'asterix'.

I messaggi che invieremo sono:

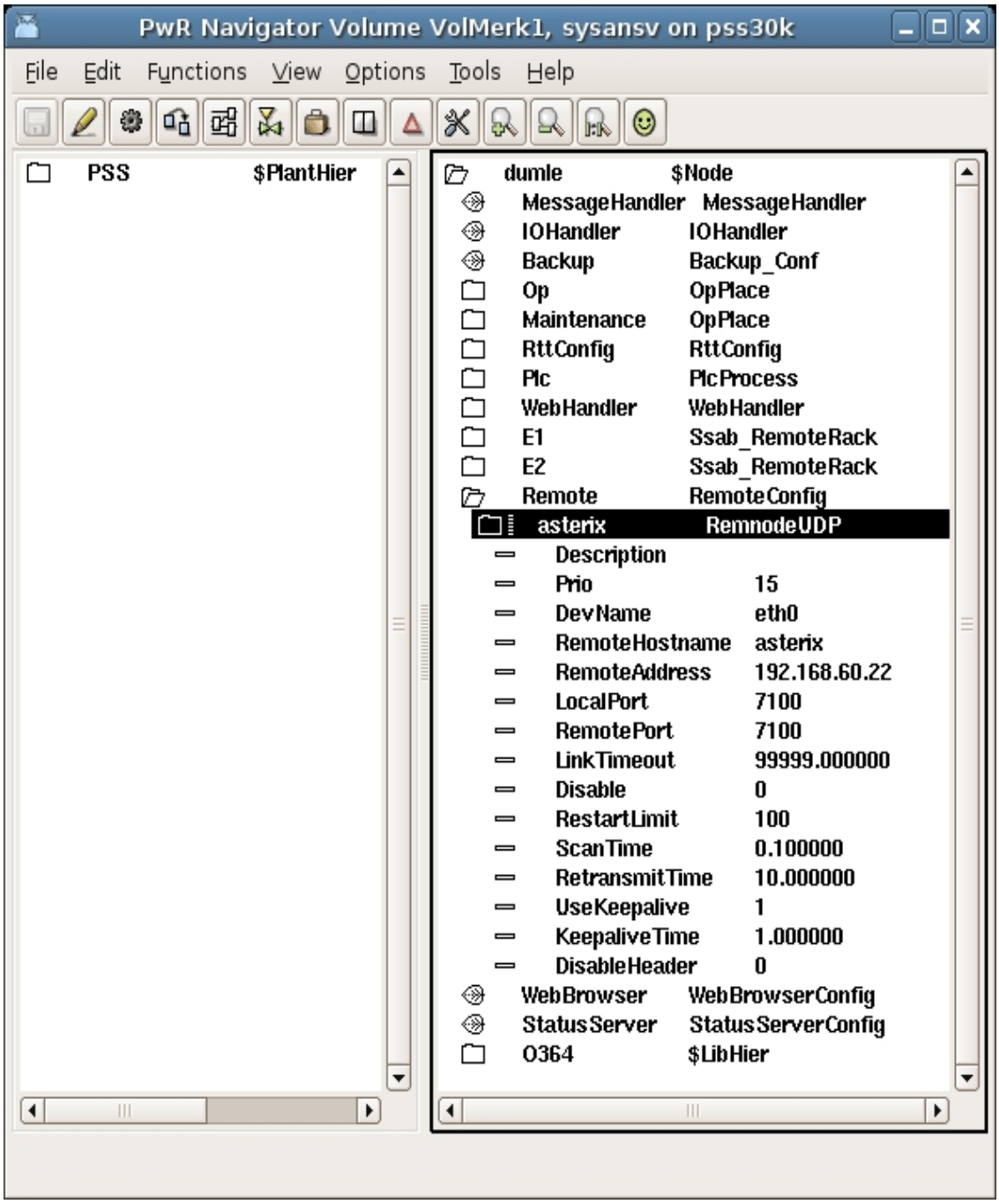
d_a_RequestData	4 Byte
d_a_Report	20 Byte

ed i messaggi che riceveremo sono:

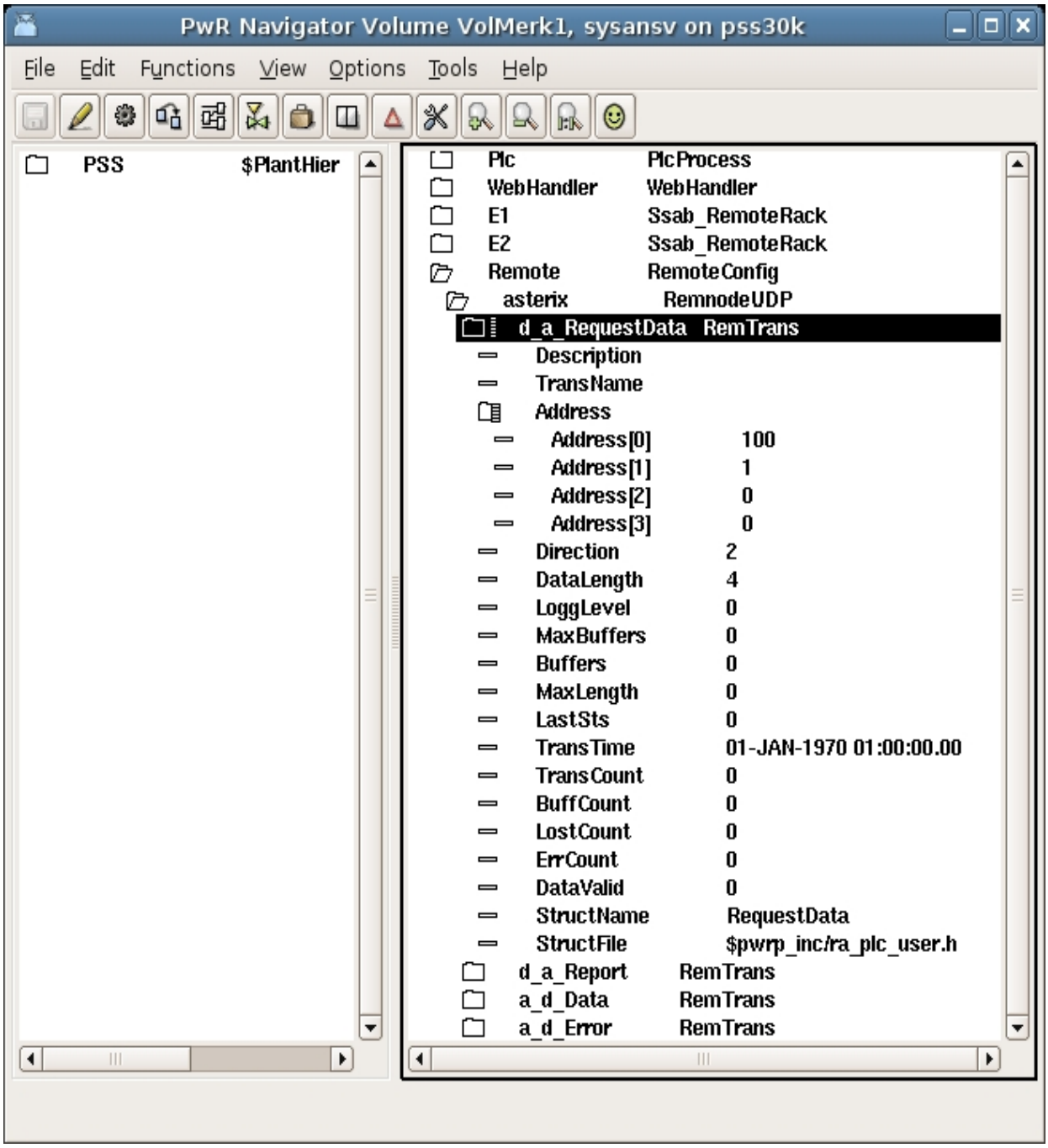
a_d_Data	365 Byte (as an answer to the d_a_RequestData-message)
a_d_Error	10 Byte

La configurazione nella gerarchia dei nodi e' simile a questa.

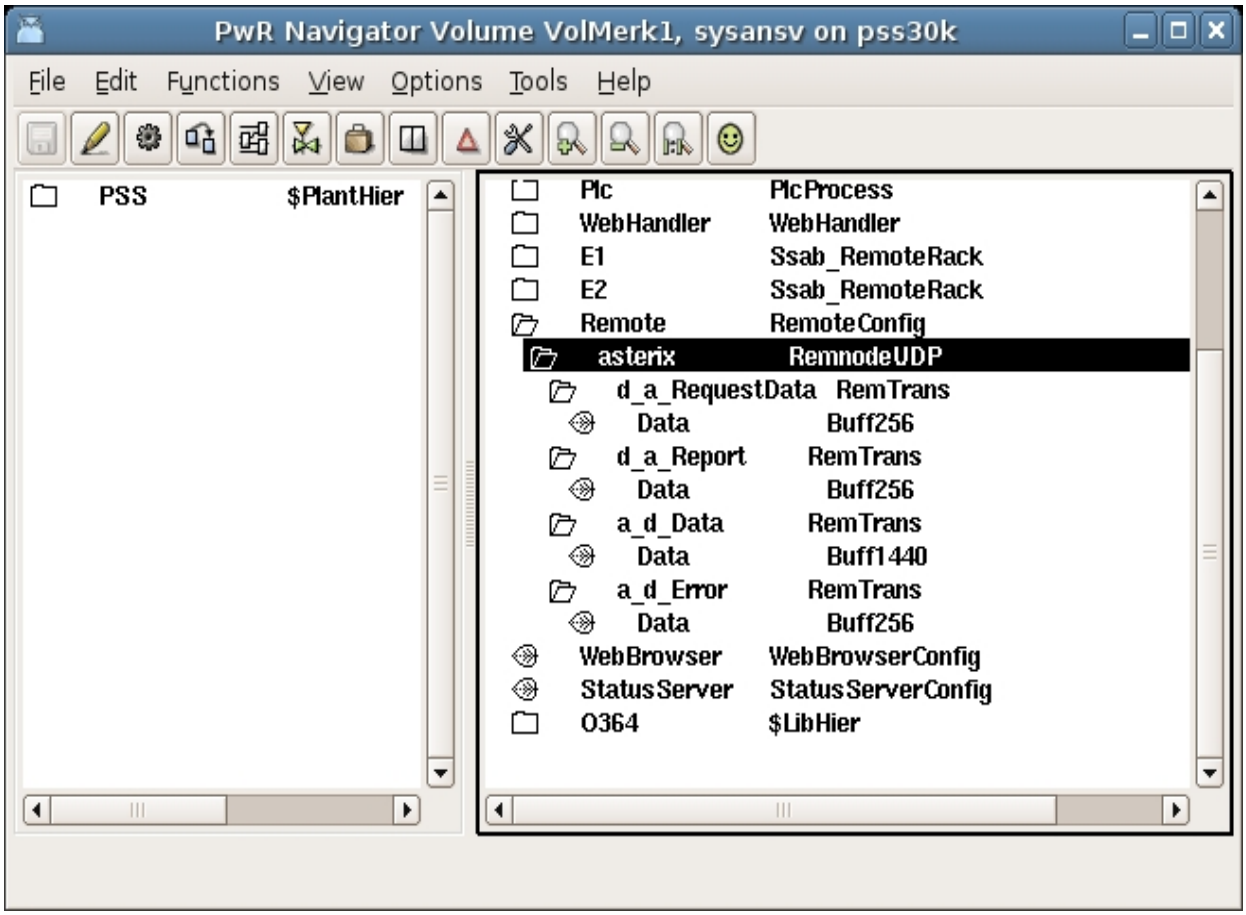
L'oggetto RemoteConfig deve essere li. Ho aggiunto un oggetto RemnodeUDP sotto questo indirizzo configurato e nodename cosi come i numeri di porta su cui comunicare.



Sotto l'oggetto RemnodeUDP ho aggiunto quattro oggetti RemTrans, uno per ogni messaggio. Negli oggetti remtrans ho configurato la direzione (invia o ricevi) e numerato gli indirizzi in modo da poter distinguere tra i messaggi e impostare le dimensioni sui messaggi di invio.



Sotto gli oggetti remtrans ho inserito i buffer dei dati.
Per i messaggi piu' piccoli ho scelto il buffer piccolo Buff256.
Il Data-message e' piu' grande e quindi per questo messaggio ho scelto Buff1440-buffer .



Le strutture dati

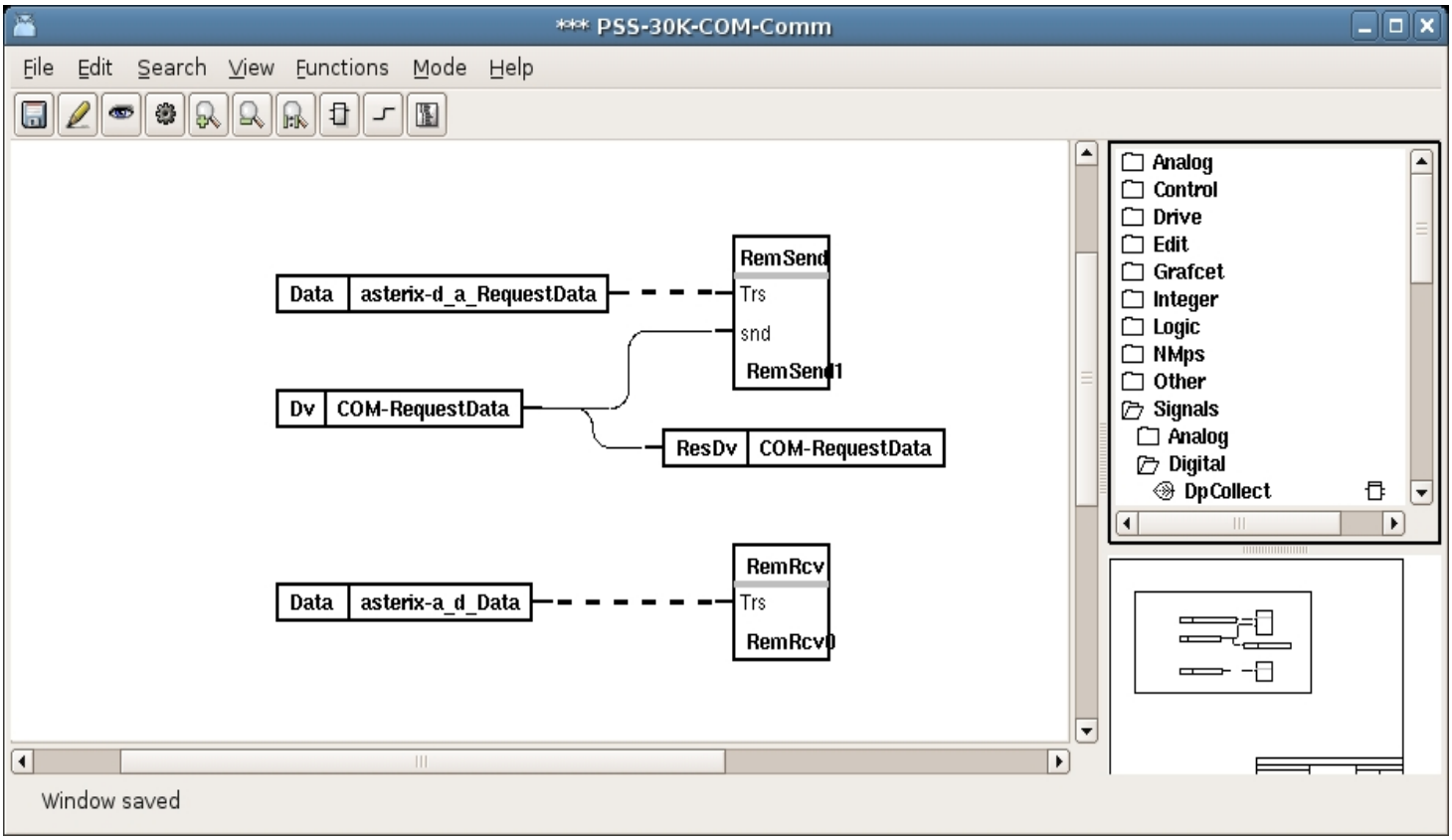
Le strutture dati per i messaggi sono definite nel file ra_plc_user.h in $ pwrp_inc-directory.
Questo file viene automaticamente incluso quando si compila il codice plc.
Le strutture sono simili a:

```
typedef struct {
  pwr_tUInt32  Id;
} d_a_RequestData;

typedef struct {
  pwr_tUInt32   Id;
  pwr_tFloat32  data_1;
  pwr_tInt32    data_2;
  pwr_tInt32    data_3;
  pwr_tInt32    data_4;
} d_a_Report;

typedef struct {
  pwr_tUInt32   Id;
  pwr_tFloat32  data_1;
  ...
  ...
  pwr_tInt32    data_xx;
} a_d_Data;
```

```
typedef struct {
  pwr_tUInt32   Id;
  pwr_tInt32    func_no;
  pwr_tInt16    err_code;
} a_d_Data;
```

Il codice plc

Ho un programma plc chiamato Comm. In questo programma ho inserito un oggetto RemTransSend e un oggetto RemTransRcv. Questi oggetti si trovano sotto la gerarchia "Other" nella palette dell'editor di plc.
All'oggetto RemTransSend ho collegato il RemTrans che voglio inviare; in questo caso il messaggio d_a_RequestData. Il messaggio verra' inviato quando e' impostato il segnale Dv "RequestData".
Allo stesso modo ho collegato il messaggio a_d_Data all'oggetto RemTransRcv, che sara' la risposta alla mia richiesta.

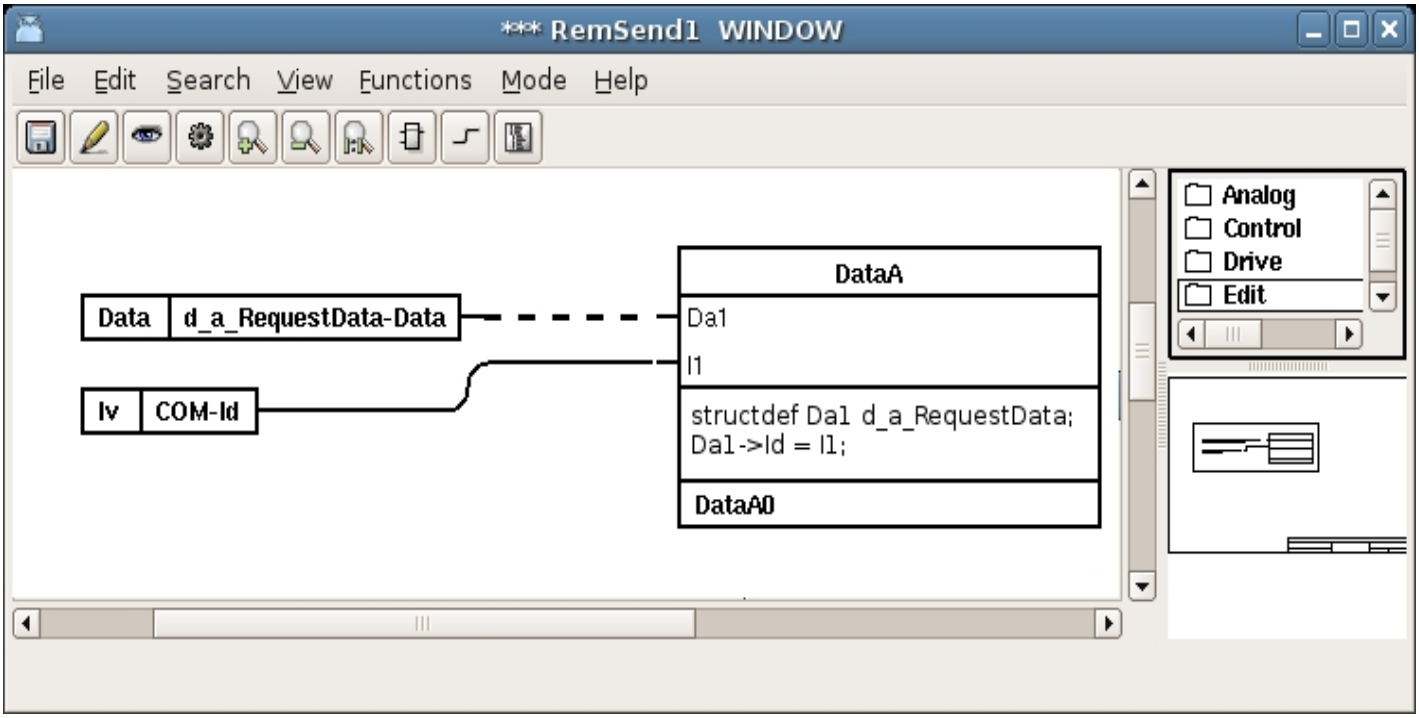


Sia il RemTransSend che gli oggetti RemTransRcv hanno una sottofinestra.
Per RemTransSend questa sottofinestra verra' eseguita quando c'e' un fronte sul snd-pin.
Quando la sottofinestra e' stata eseguita, viene impostato l'attributo DataValid dell'oggetto RemTrans collegato.
Il lavoro di trasporto per questo Remnode invia il messaggio e imposta il valore DataValid-flag su zero.

Per RemTransRcv la sottofinestra verra' eseguita quando viene impostato l'attributo DataValid nell'oggetto RemTrans connesso.
Quando il lavoro di trasporto per questo Remnode riceve un messaggio, riempie il buffer di dati con i dati ricevuti e quindi imposta il flag DataValid.
Dopo l'esecuzione il flag verra' ripristinato.

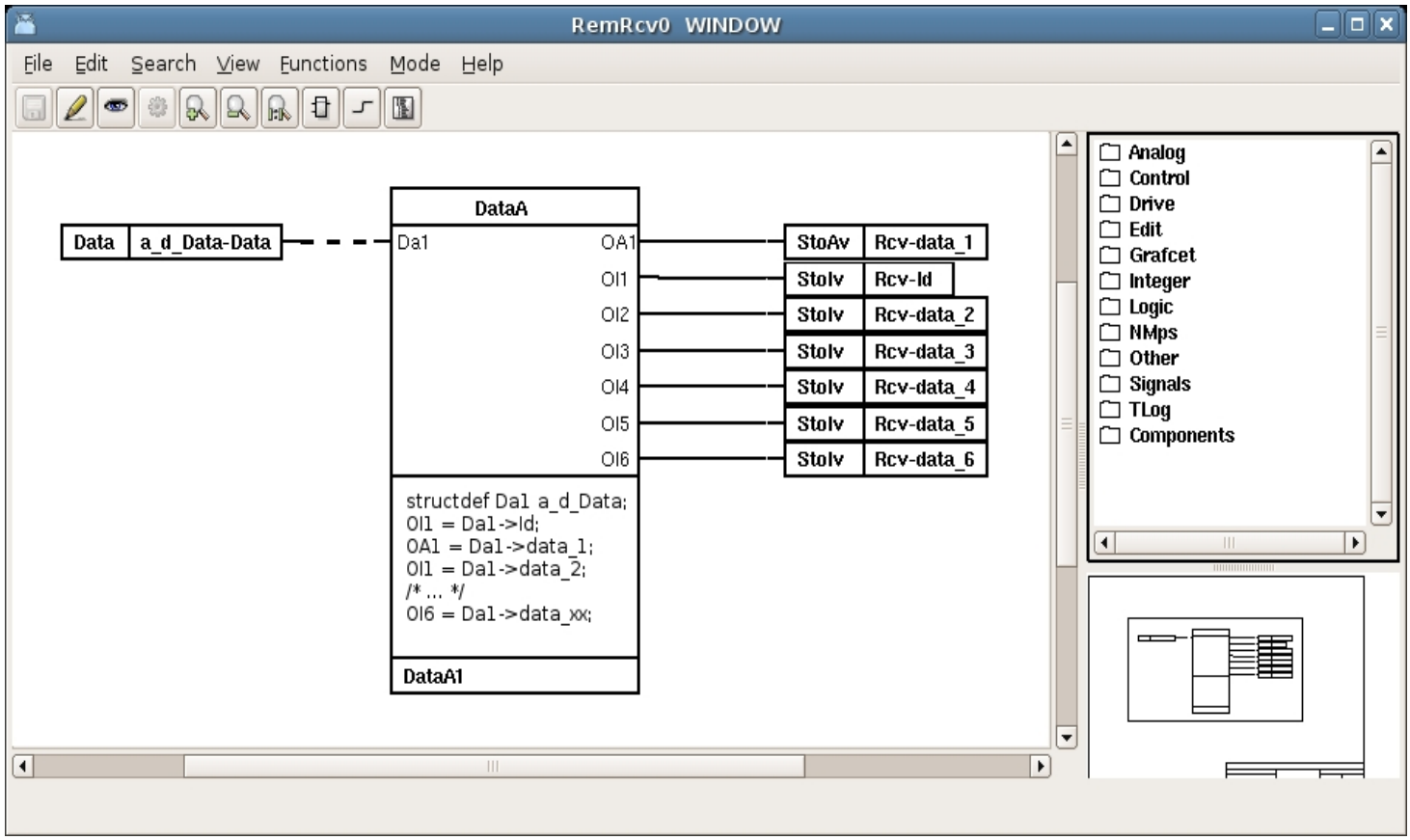
Invia sottofinestra

Nella sottofinestra di invio si inseriscono i dati nel buffer di invio. Il buffer di invio per il messaggio da inviare e' connesso a un DataArithm.
La speciale sintassi 'structdef' consente a Da1 di essere un puntatore a una struttura d_a_RequestData.



Ricevi sottofinestra

Nella sottofinestra di ricezione scompattiamo i dati ricevuti nel buffer di ricezione.
Il buffer di ricezione e' connesso a un DataArithm. Ancora una volta usiamo la dichiarazione classdef per eseguire un 'cast' (conversione di tipo) del puntatore Da1.
Decomprimiamo i dati sui pin di uscita di DataArithm. Se un DataArithm non e' sufficiente per decomprimere i parametri, aggiungiamo semplicemente DataArithm e continuiamo nello stesso modo.


RemRcv0 WINDOW
File Edit Search View Functions Mode Help
Data
a_d_Data-Data
DataA
Da1
OA1
OI1
OI2
OI3
OI4
OI5
OI6
StoAv
Rcv-data_1
StoIv
Rcv-Id
StoIv
Rcv-data_2
StoIv
Rcv-data_3
StoIv
Rcv-data_4
StoIv
Rcv-data_5
StoIv
Rcv-data_6
structdef Da1 a_d_Data;
OI1 = Da1->Id;
OA1 = Da1->data_1;
OI1 = Da1->data_2;
/* ... */
OI6 = Da1->data_xx;
DataA1
Analog
Control
Drive
Edit
Grafcet
Integer
Logic
NMps
Other
Signals
TLog
Components

14 Archivio dati

Esistono tre diversi tipi di archiviazione dei dati in Proview, trend, curve veloci e archiviazione dei dati storici.
I trend memorizzano i dati ciclici durante un periodo di tempo piu' breve nel database in tempo reale.
Le curve veloci vengono attivate da alcuni eventi e memorizzano i dati per un periodo di tempo dopo l'evento trigg e in alcuni casi anche prima.
La memorizzazione storica viene effettuata su disco in un database relazionare ed e' in grado di memorizzare ciclicamente i dati per diversi anni.

There are three different types of data storage in Proview, trends, fast curves and historical data storage. Trends are storing data cyclic during a shorter period of time in the realtime database. Fast curves are triggered by some event and stores data for a period of time after the trigg event, and in some cases also before. The historical storage is made on disk in a relation database and is able to cyclically store data in several years.

14.1 Trends

Esistono due tipi di trends
- DsTrend con un buffer di dati interno in grado di memorizzare circa 500 campioni con un tempo di scansione minimo di 1 s.
- DsTrendCurve, con dimensione del buffer configurabile e con un tempo di scansione minimo di 20 ms.

14.1.1 DsTrend

L'oggetto DsTrend ha un buffer di dati interno di 1912 byte che puo' ad esempio memorizzare 478 campioni di tipo Float32. Sono possibili anche altri tipi di dati. La configurazione e' un po 'strana perche' il buffer e' diviso in due parti, ma normalmente e' sufficiente indicare l'attributo che deve essere memorizzato in DataName e possibilmente inserire un valore in Multiplo per indicare il tempo di scansione.

Vedi anche DsTrend in Object Reference Manual.

14.1.2 DsTrendCurve

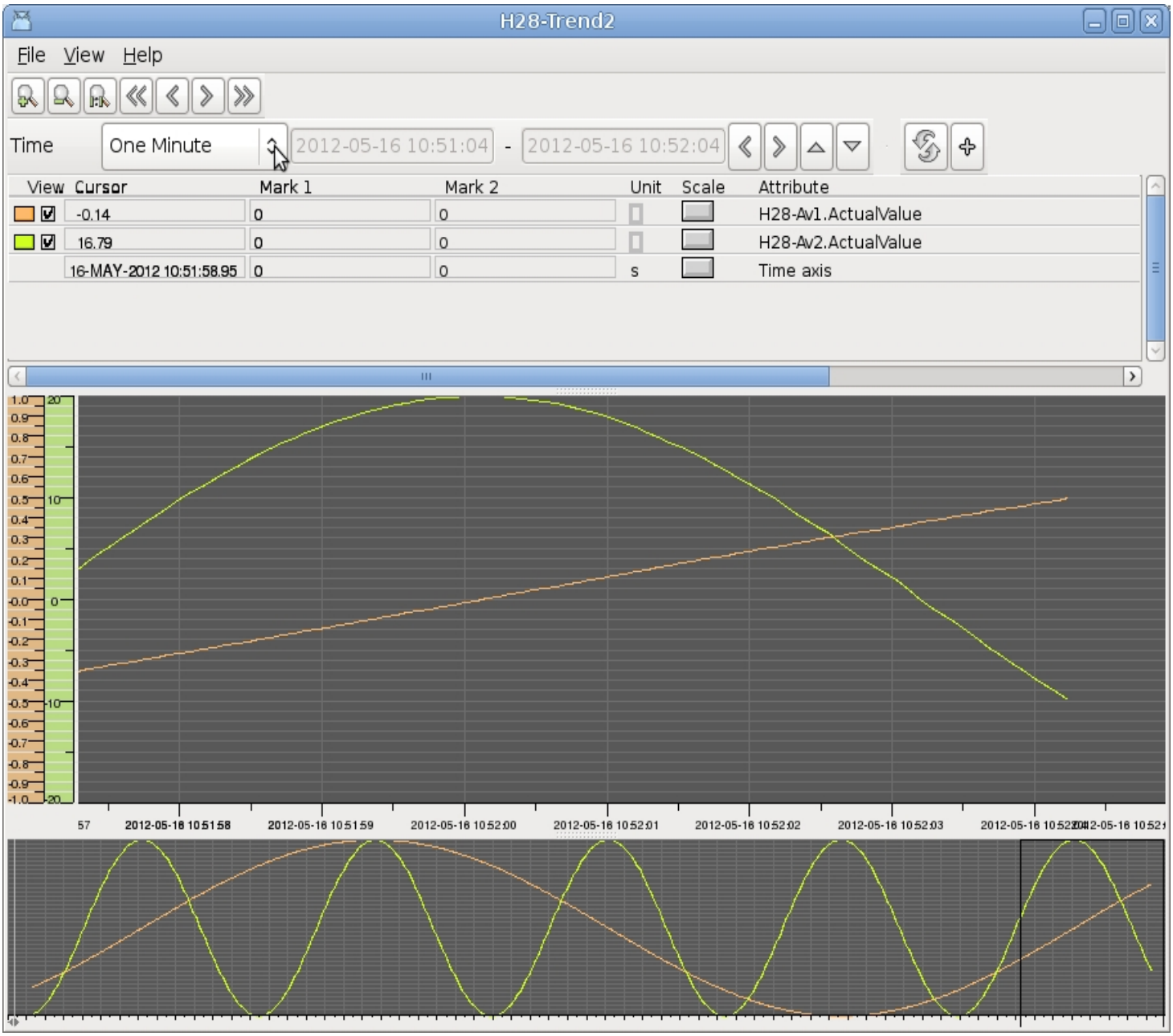


Fig Trend configurato con DsTrendCurve

Gli oggetti DsTrendCurve memorizzano le curve con oggetti buffer esterni, dove la dimensione degli oggetti Buffer limita il numero di campioni che possono essere memorizzati.
Un oggetto CricBuff100k, come nell'esempio seguente, puo' contenere 25000 campioni.
La memorizzazione puo' essere eseguita con scantime fino a 20 ms.
Abbiamo a disposizione anche una funzione di istantanea in cui la curva di tendenza attuale viene congelata e visualizzata in una finestra dei grafici dove puo' essere analizzata in dettaglio.
L'istantanea della curva puo' essere archiviata in file e aperta in un'occasione successiva.

La curva e' configurata con un oggetto DsTrendCure. Fino a 10 attributi diversi vengono memorizzati e possono essere specificati nell'array Attribute.
Per ogni attributo, un oggetto buffer di tipo CircBuffer viene creato e dichiarato nell'attributo Buffer.
La dimensione dell'oggetto buffer deve essere adattata al numero di campioni nella curva.
E' anche possibile specificare un buffer temporale, ma cio' e' necessario solo se si desidera

utilizzare la funzione snapshot.
La risoluzione temporale determina la dimensione del buffer temporale. Per una risoluzione di 1 s, vengono utilizzati 4 byte per campione e per una risoluzione di 1 ns sono necessari 8 byte.

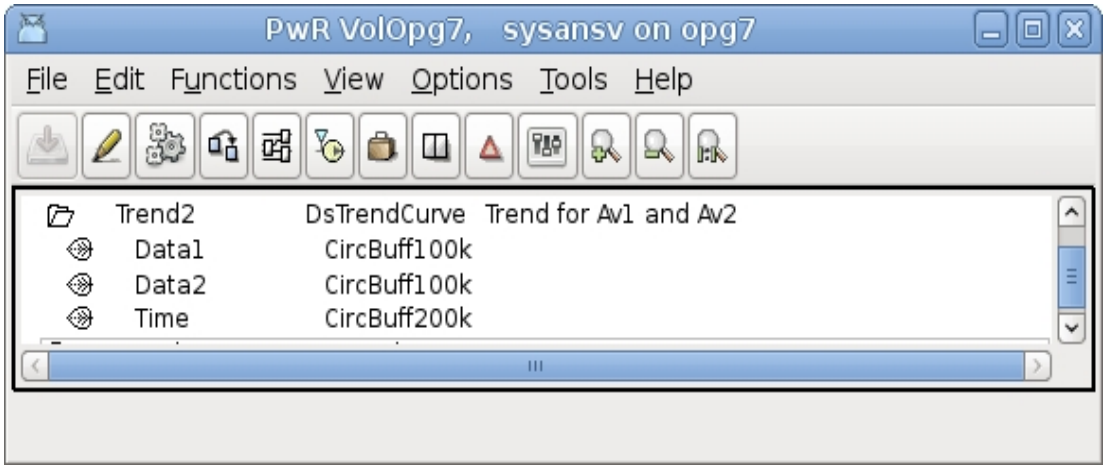


Fig Configurazione di una DsTrendCurve con buffer di oggetti

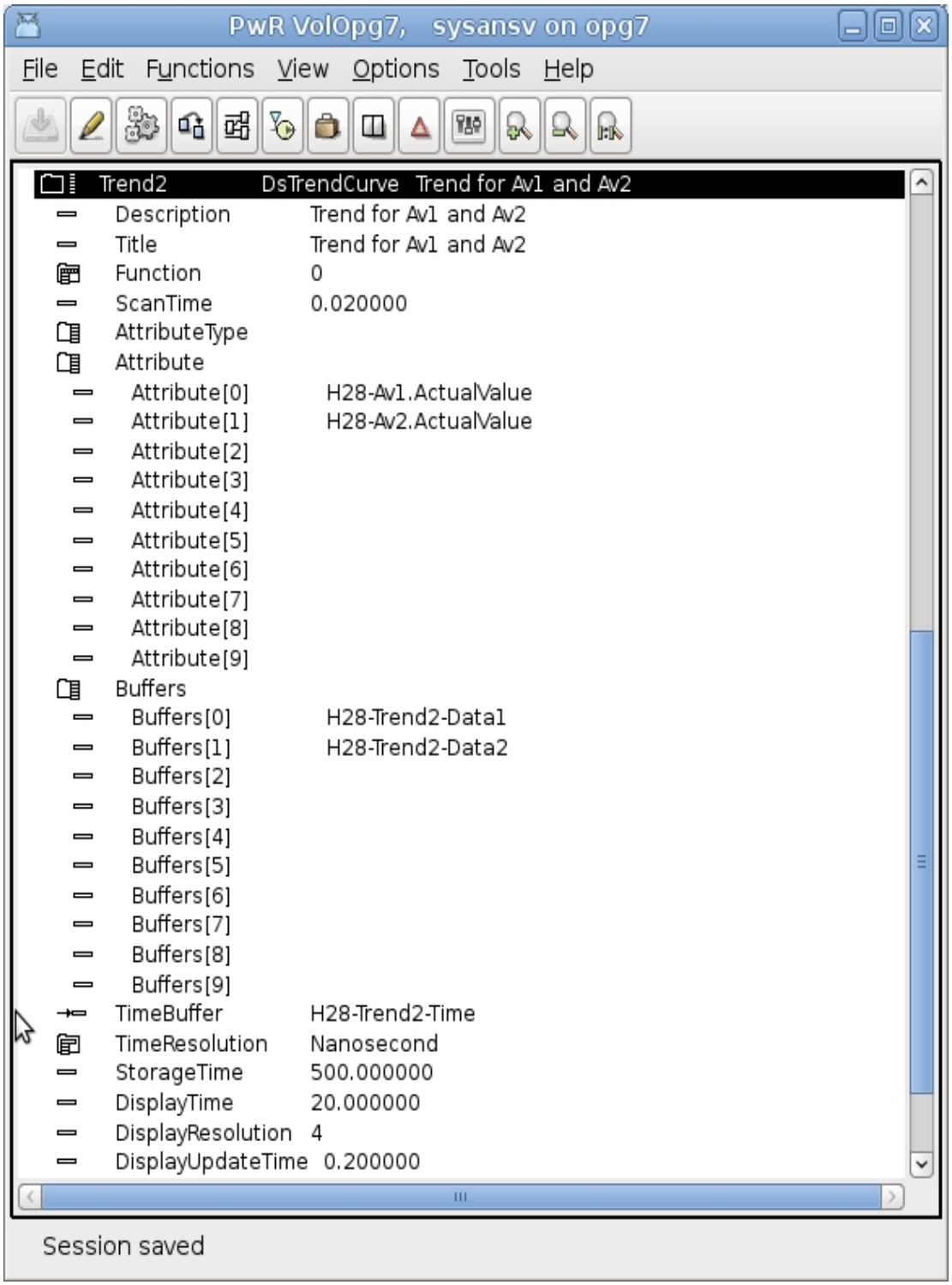


Fig Configurazione di DsTrendCurve

Istantanea (Snapshot) significa che si esegue una copia dei dati e la si visualizza in una finestra separata.
E' possibile configurare la curva per mostrare solo una parte limitata dei dati memorizzati e questi dati vengono archiviati per un periodo di tempo piu' lungo e anche con una risoluzione piu' elevata.
Nella finestra dell'istantanea e' quindi possibile andare piu' indietro nel tempo o ingrandire e aumentare la risoluzione della curva.

Gli attributi DisplayTime e DisplayResolution nell'oggetto DsTrendCurve vengono utilizzati

per specificare quala parte dei dati viene visualizzata nella finestra del trend.
Nell'esempio sopra il tempo di memorizzazione totale e' di 500 secondi, mentre solo gli ultimi 20 s (DisplayTime) sono mostrati nella finestra del trend.
DisplayResolution e' impostato su 4, che indica che la visualizzazione avviene ogni quarto campioni.
Poiche' il tempo di scansione e' di 20 ms, viene visualizzato un valore ogni 80 ms e in totale 20 s/0,08 s = 250 campioni vengono visualizzati.
I dati totali contengono 500 s / 0,02 s = 25000 campioni che sono quindi disponibili nella finestra dell'istantanea.

Vedi anche DsTrendCurve nel Manuale di riferimento dell'oggetto.

14.2 Curve veloci

Una curva veloce viene attivata da un determinato evento e quindi memorizza i dati per un determinato periodo di tempo rispetto a quelli visualizzati in una finestra curva.
L'evento trigger puo' essere un segnale digitale che va alto, un segnale analogico che raggiunge un valore limite o un trigger manuale.
La curva veloce puo' essere configurata per memorizzare continuamente i dati in modo che anche i dati possano essere visualizzati prima del punto di trigger.

Una curva veloce e' configurata con un oggetto DsFastCurve. Fino a 10 attributi possono essere gestiti da un oggetto DsFastCurve e gli attributi sono indicati nell'array Attribute.
Per ciascun attributo, un oggetto buffer deve essere creato e indicato nell'array Buffers.
Anche un oggetto buffer per il tempo dovrebbe essere creato e dichiarato nell'attributo TimeBuffer.
La dimensione degli oggetti buffer deve essere adattata alla quantita' di dati che deve essere memorizzata.

Una curva veloce puo' essere visualizzata in una normale finestra per curve, che viene aperta ad esempio dalla voce Fast nel menu popup. Puo' anche essere visualizzata in un grafico Ge con un componente FastCurve.

Vedi anche DsFastCurve nel Manuale di riferimento agli oggetti.

14.3 Archiviazione di dati storici

Sev e' l'ambiente di archiviazione, in cui i dati storici sono archiviati in un database.
E' un complemento agli altri ambienti in Proview, allo sviluppo, al runtime e all'ambiente operatore.
Sev contiene processi server che gestiscono il recupero e l'archiviazione di dati storici.
Sev puo' essere installato come unita' separata su una stazione di archiviazione, ma e' anche incluso nel pacchetto runtime e puo' essere avviato nell'ambiente di runtime.

Stazione di stoccaggio

Una stazione di archiviazione e' un nodo del server in cui e' installato il pacchetto sev (pwrsev). Questo nodo comunica tramite Qcom con i nodi client e riceve i dati storici memorizzati in un database MySQL o SQLite. I dati storici possono essere gestiti e visualizzati sulla stazione di memoria dal programma sev_xtt, oppure possono essere inviati ai nodi client e visualizzati da rt_xtt nell'ambiente operatore.

Configurazione

La configurazione di una stazione di archiviazione viene effettuata nella Directory Volume, creando un oggetto SevNodeConfig sotto un oggetto BusConfig.
Il nodo puo' essere inserito in un progetto con stazioni di processo e operatore, oppure puo' essere collocato in un progetto separato con la sola stazione di archiviazione.
Se la stazione di memoria deve comunicare con i nodi client in altri progetti, e' necessario configurarli come nodi amici, con oggetti FriendNodeConfig.
La stazione di memoria deve anche essere configurata come nodo amico sui nodi client.
L'attributo Connection in FriendNodeConfig deve essere impostato su QcomOnly.
Questo dovrebbe essere fatto negli oggetti FriendNodeConfig in entrambi i lati.

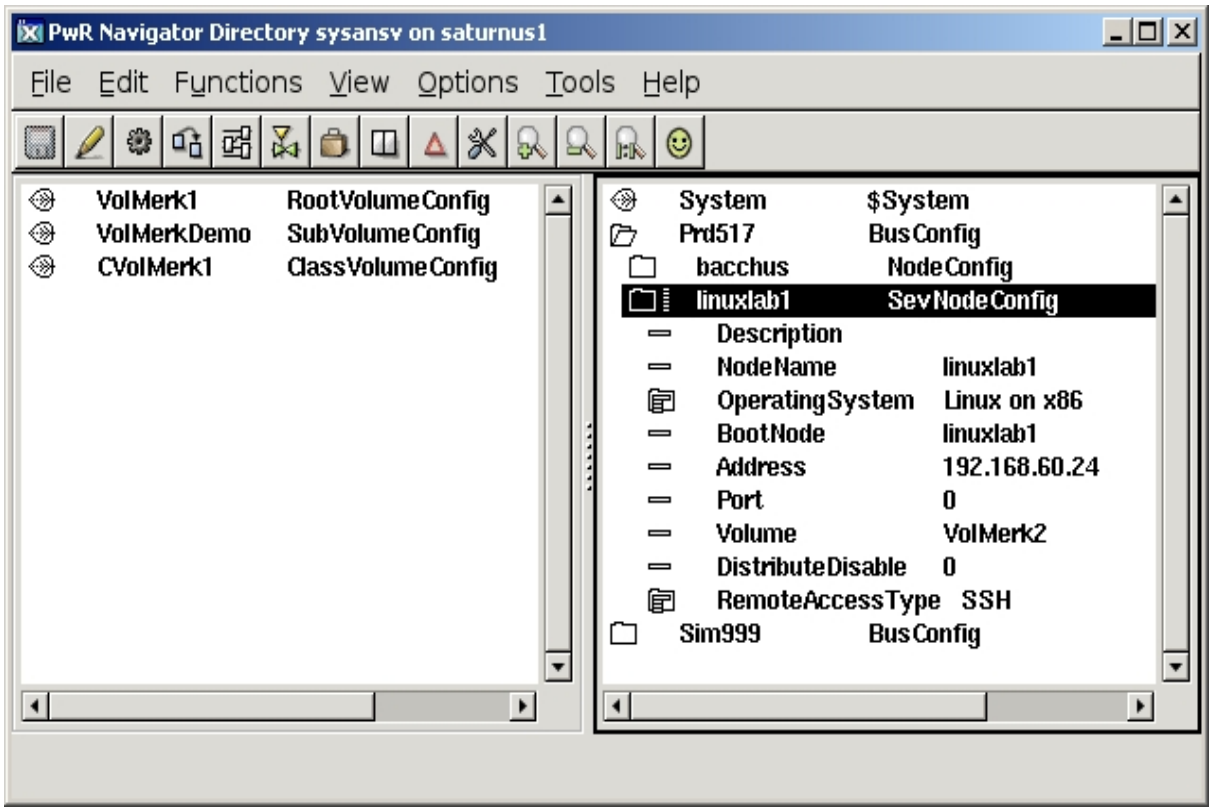


Fig Configurazione di una stazione di archiviazione

Archiviazione dei dati storici

La memorizzazione dei dati storici e' suddivisa in un processo server, sev_server, che gestisce l'archiviazione dei dati storici e un processo client, rt_sevhistmon, che raccoglie i dati sul nodo client e li invia al server. Il processo del server fa parte dell'ambiente di archiviazione e il client elabora una parte dell'ambiente di runtime.
Il processo del server e' anche incluso nell'ambiente di runtime, il che rende possibile creare una stazione di processo/storage combinata.

Configurazione

SevHist

Gli attributi che devono essere memorizzati nel database storico sono configurati con oggetti SevHist.
Gli oggetti SevHist sono posizionati in un volume root o sub e l'attributo e' dichiarato in

Attribute nell'oggetto SevHist.
L'oggetto SevHist viene normalmente posizionato sotto l'oggetto che deve essere memorizzato e se questo oggetto ha un attributo ActualValue, questo viene automaticamente inserito nell'oggetto SevHist.

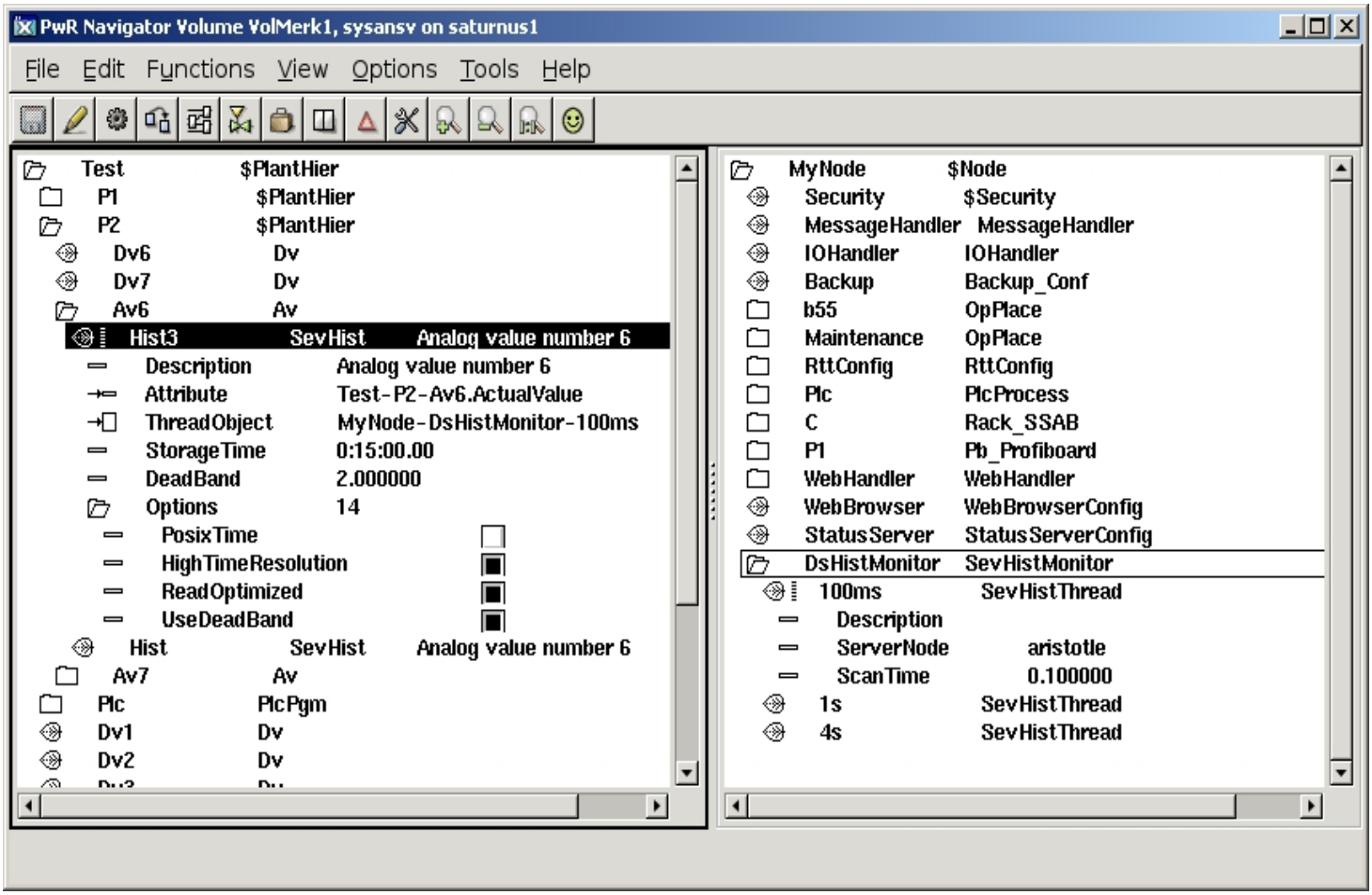


Fig Configurazione in un volume root degli oggetti SevHist, DsHistMonitor e SevHistThread

SevHistObject

SevHistObject e' simile a SevHist ma memorizzera' tutti gli attributi in un oggetto nel database.
Tutti gli attributi verranno salvati in una tabella con lo stesso tempo.

SevHistMonitor

Il processo client, rt_sevhistmon, e' configurato con un oggetto SevHistMonitor nella gerarchia dei nodi.
rt_sevhistmon analizza gli oggetti SevHist e SevHistObject e invia i dati storici al server sev.

SevHistThread

Gli oggetti SevHistThread sono posizionati sotto l'oggetto SevHistMonitor per configurare basi temporali e nodi server nel client. Ogni oggetto SevHist e SevHistObject e' connesso a un oggetto SevHistThread che determina a quale nodo del server e quanto spesso vengono inviati i dati.

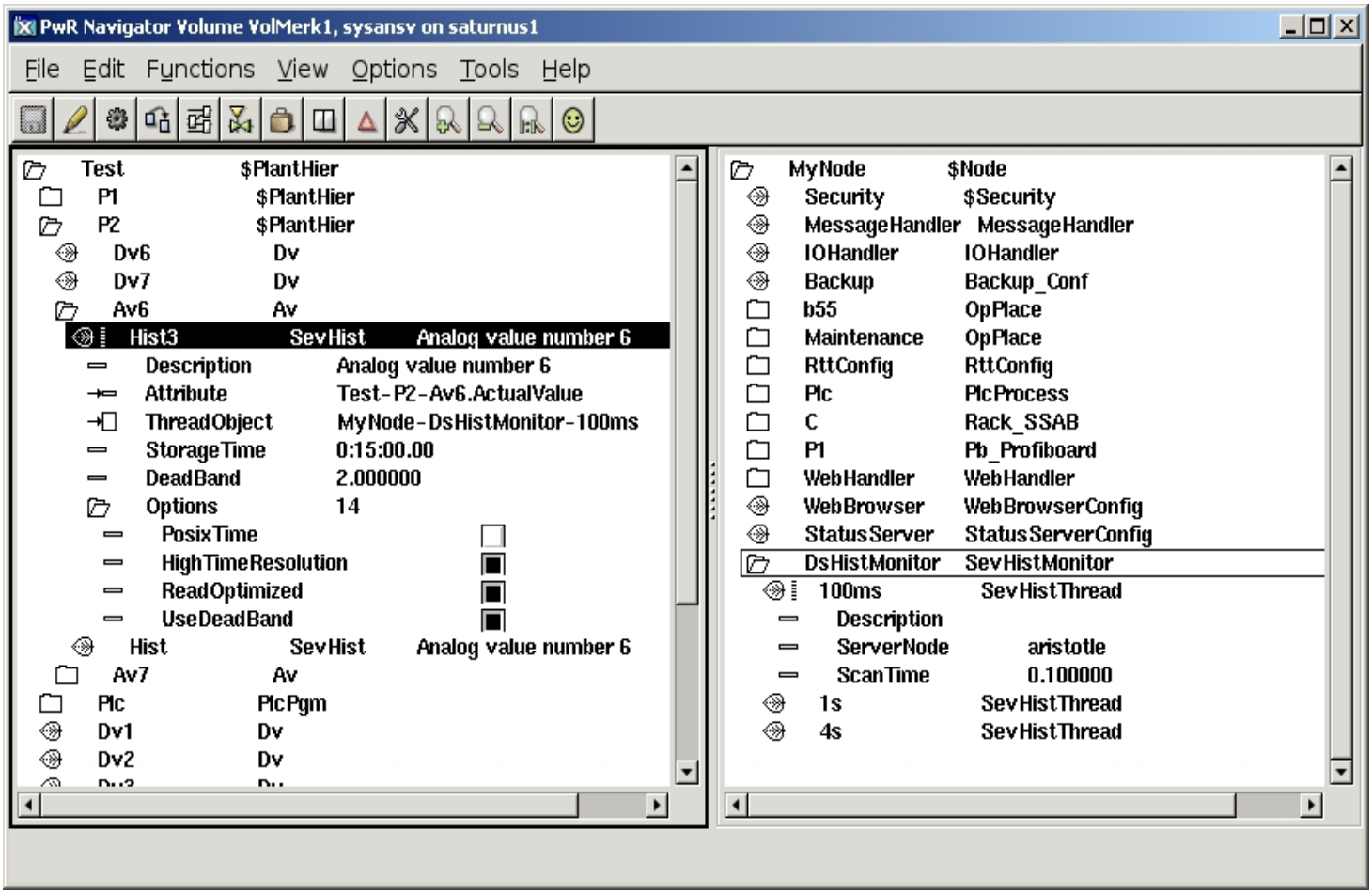


Fig Configuration in a rootvolume with SevHist, SevHistMonitor and SevHistThread objects

Server process

Il Server process, sev_server, gestisce il database che archivia i dati storici e comunica con i client che forniscono dati o richiedono dati, ad esempio, visualizzati in una finestra curva.
Il server puo' essere avviato nell'ambiente di runtime ed e' configurato da un oggetto SevServer nella gerarchia dei nodi. Nell'ambiente di archiviazione, il processo del server viene avviato automaticamente all'avvio di Proview.

Database

MySQL o SQLite possono essere scelti come database nel sev server.
La scelta viene effettuata nell'oggetto SevServer, se viene avviato il runtime Proview o nel file /etc/proview.cnf con il parametro 'sevDatabaseType' che puo' avere il valore 'sqlite' o 'mysql'.

```
sevDatabaseType sqlite
```

MySQL e' piu' testato e dovrebbe essere usato per i database piu' grandi.

sev_xtt

Sev_xtt e' uno strumento per navigare e visualizzare i dati nel database storico.
Viene avviato sul nodo del server e puo' solo visualizzare i dati in questo nodo.
All'inizio, un elenco di tutti gli attributi memorizzati viene recuperato e visualizzato in una struttura ad albero.
Aprendo un oggetto (un attributo memorizzato) vengono visualizzate le proprieta' dell'elemento.

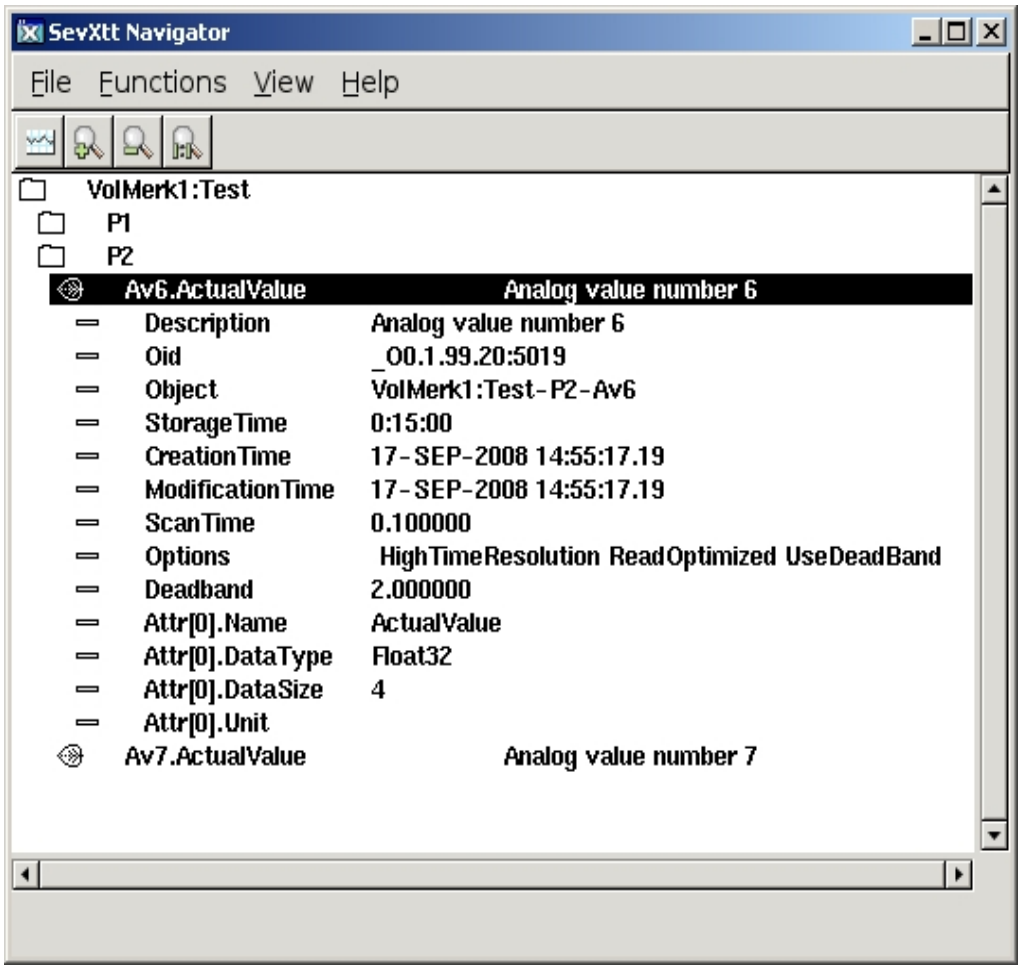


Fig Un elemento visualizzato in Sev_xtt

Se si seleziona una voce e si attiva il pulsante della curva, i valori memorizzati vengono recuperati e visualizzati in una finestra curva.
Quando l'elemento viene aperto, vengono recuperati circa 500 valori dagli ultimi dati memorizzati.
e' quindi possibile ingrandire o ridurre con una risoluzione superiore o inferiore e andare piu' indietro nel tempo.

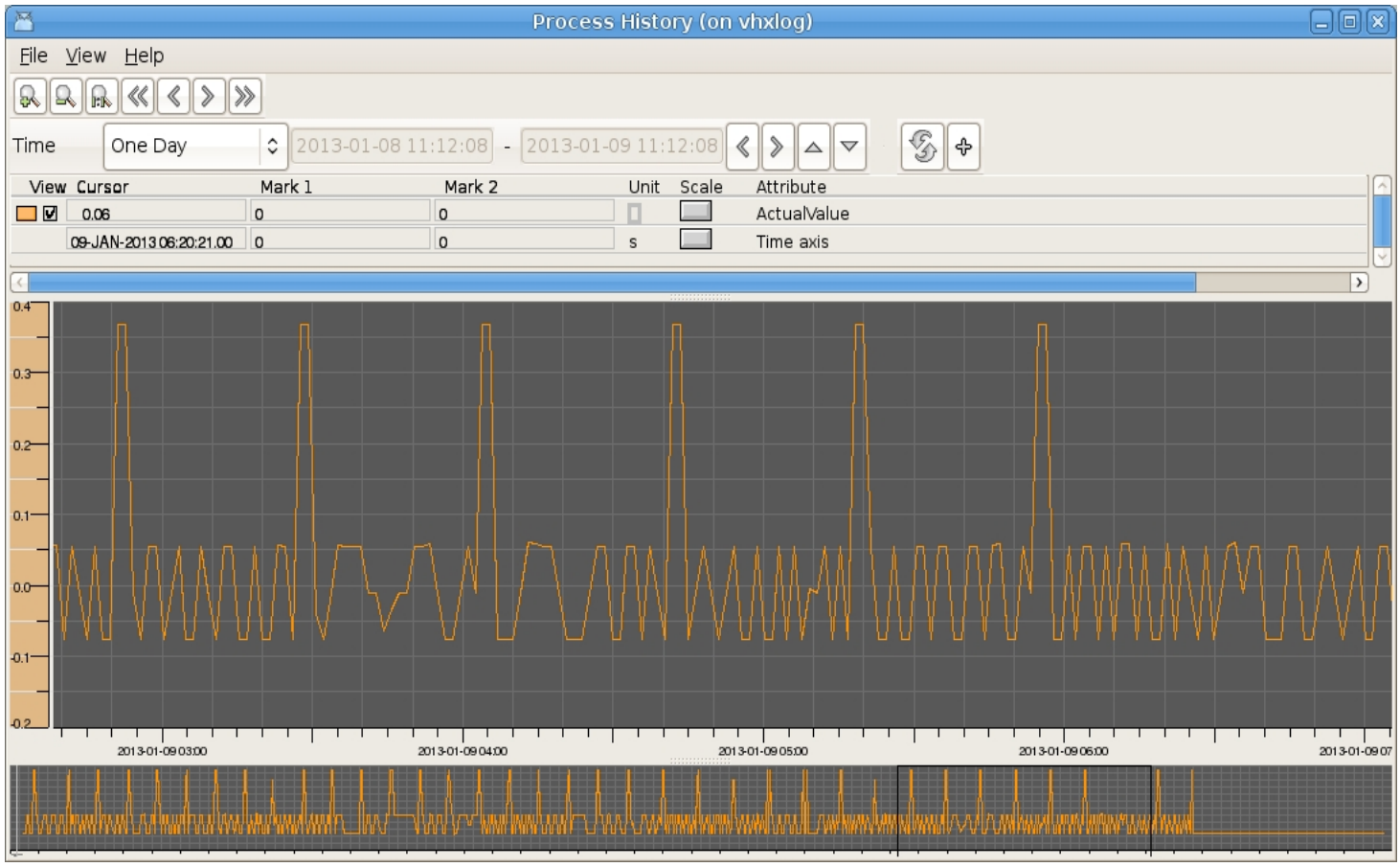


Fig Finestra della curva in sev_xtt

I vecchi elementi possono essere rimossi dal database da sev_xtt selezionando l'elemento e attivando 'Function/Delete Item' nel menu. Solo gli utenti con il privilegio SevAdmin sono autorizzati a farlo.

Accesso

Esistono due privilegi che controllano l'accesso in sev_xtt:
SevRead e' necessario per visualizzare il contenuto del database di archiviazione e SevAdmin deve influire sul database di archiviazione (al punto di poter eliminare gli elementi di archiviazione).
Esistono due modi per ottenere questi privilegi:

1. Accedere come utente a cui sono concessi i privilegi SevRead o SevAdmin.

2. Impostare un privilegio predefinito per sev_xtt in /etc/proview.cnf. Il parametro sevXttDefaultPriv puo' avere i valori Read, Admin o None. Se il valore e' Read, sev_xtt avra' il privilegio SevRead come predefinito, se il valore e' Admin, sev_xtt avra' il privilegio SevAdmin come predefinito e se il valore e' None, e' richiesto un login con nome utente e password in sev_xtt.

rt_xtt

I dati storici possono anche essere visualizzati da rt_xtt nell'ambiente operatore.
Per oggetti che hanno un oggetto SevHist come figlio, una voce di menu 'History' viene aggiunta al menu a comparsa.

Il metodo history visualizza la stessa finestra curva di sev_xtt.

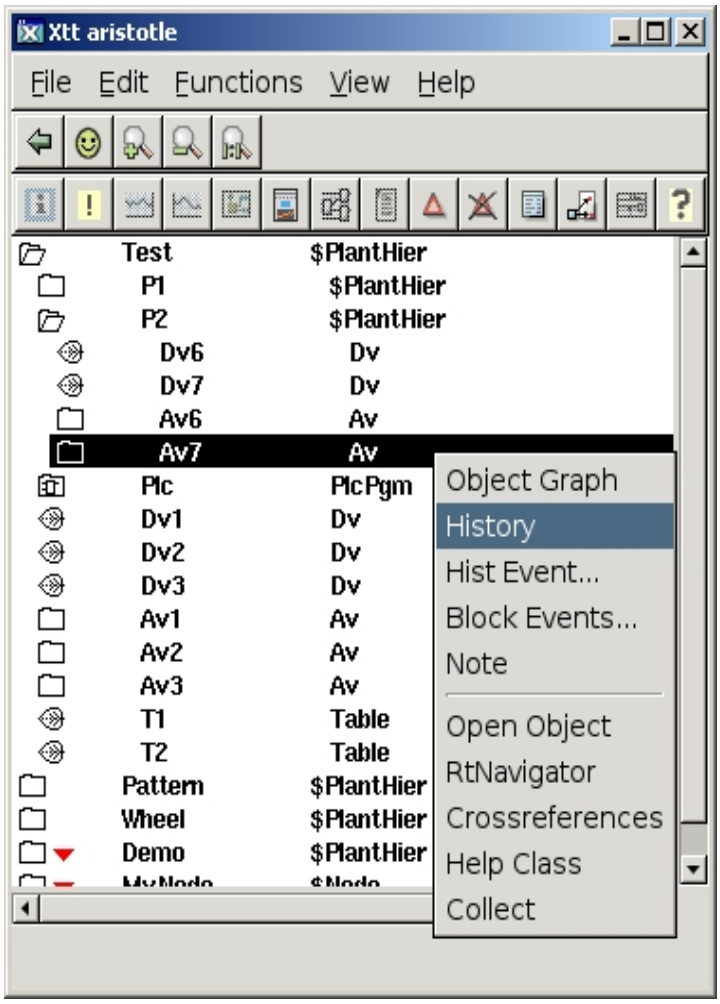


Fig Metodo History nell'ambiete operatore

Installazione di una stazione di archiviazione

MySQL

Si consiglia di utilizzare il motore database innodb. Specificare in /etc/proview.cnf che mysql deve essere usato e che le tabelle devono essere create con engine innodb

```
sevDatabaseType mysql
sevMysqlEngine innodb
```

Installate mysql-server sulla stazione di archiviazione, e aggiungete l'utente mysql chiamato pwrp

```
> mysql
mysql> grant all privileges on *.* to pwrp@localhost;
```

Ci sono molti parametri per mettere a punto mysql. Un suggerimento e' di aumentare le dimensioni del pool innodb aggiungendo queste righe in /etc/mysql/my.cnf

```
innodb_buffer_pool_size=1G
innodb_additional_mem_pool_size=20M
```

Per i database di grandi dimensioni, e' necessario aumentare la dimensione del file di registro e l'opzione per usare un file di log per tabella fornira' maggiori informazioni e maggior controllo dello spazio su disco.

```
innodb_log_file_size=128M
innodb_log_buffer_size=16M
innodb_file_per_table
```

Per il calcolo dello spazio su disco per innodb sono necessari circa 47 byte per valore memorizzato.
La memorizzazione di un segnale con un ciclo di 10 s in un anno richiede circa
6 * 60 * 24 * 365 * 47 byte = 148 Mb.
L'opzione UseDeadBand nell'oggetto SevHist ridurra' questo valore.

SQLite

Installare libsqlite3-0 sulla stazione di archiviazione.

pwrsev

Installare il pacchetto pwrsev, distribuire e quindi avviare l'ambiente di archiviazione con

```
> pwrsev start
```

Il carico di una stazione di archiviazione dipende principalmente dal numero di elementi memorizzati, dai tempi di scansione e dalle prestazioni dell'unita' disco.
Il carico viene continuamente calcolato dal sev_server e memorizzato nel database nella tabella sev_stat. Di seguito e' riportato un esempio di come leggere la tabella delle statistiche in un database mysql, in cui il nome del progetto e' opg7.

```
>mysql -u pwrp
mysql> use pwrp__opg7;
mysql> select * from set_stat;
| 7.24149 | 7.90368 | 907.713 | 908.8 | 32321 | 0 | 1 | 0 |
1 row in set (0.00 sec)
```

I primi quattro valori mostrano
1. Il carico corrente, ovvero il tempo di esecuzione del database in percentuale.
2. Carico medio.
3. Numero di elementi di dati memorizzati al secondo, valore corrente.
4. Numero di elementi di dati memorizzati al secondo, valore medio.

Il carico medio dovrebbe essere ben al di sotto del 100%.

15 Programmazione delle applicazioni

Questo capitolo tratta di come scrivere programmi applicativi, ad es. Programmi in c,c++ o java, che lavoreranno in Proview.
Si presume che il lettore abbia una conoscenza di base nel linguaggio di programmazione c.

In molte applicazioni Proview, la codifica dell'applicazione nell'editor plc con la programmazione di oggetti funzione produce risultati eccellenti.
Tuttavia, ci sono applicazioni che con la programmazione grafica saranno inutilmente complesse, ad esempio modelli avanzati, gestione di database e pianificazione dei materiali.
In questo caso si scrive un programma applicativo in c,c++ o java, che si collega al database realtime, rtdb.
Il programma legge i dati di input da rtdb, fa i suoi calcoli e imposta i dati su rtdb, dove i dati vengono ulteriormente elaborati dal programma plc, inviati al sistema I/O e visualizzati nei grafici dell'operatore.

Ci concentreremo su c/c++, poiche' questo e' il linguaggio di programmazione piu' comune nella programmazione delle applicazioni e ha anche la maggior parte delle funzionalita'.

Le interfacce utilizzate sono descritte nel Manuale di riferimento del programmatore (PRM).

15.1 Collegarsi al database e gestire oggetti e dati

Iniziamo scrivendo una semplice aplicazione C++ che si collega al database realtime e si collega ad alcuni oggetti.

Il file cpp deve essere creato sulla directory $pwrp_src o su una sua sottodirectory.
Creiamo la directory $pwrp-src/myappl e modifichiamo il file ra_myappl.cpp.

Datatypes

Nel file include pwr.h sono definiti i tipi di dati di base in Proview. Il piu' comune e' pwr_tBoolean per i segnali digitali e pwr_tFloat32 per i segnali analoghi, ma esistono anche tipi C per tutti gli altri tipi di dati Proview,
ad es. pwr_tInt32, pwr_tUInt32, pwr_tString80 ecc.

Inizializzazione Gdh

Il database viene collegato con una chiamata a gdh_Init() che accetta come argomento una stringa identificativa per l'applicazione. Innanzitutto includiamo pwr.h che contiene i tipi di base in Proview e rt_gdh.h che contiene l'API nel database.

```
#include "pwr.h"
#include "rt_gdh.h"

int main() {
```

```
  pwr_tStatus sts;

  sts = gdh_Init( "ra_myappl");
  if ( EVEN(sts)) {
    cout << "gdh_Init failure " << sts << endl;
    exit(0);
  }
}
```

La funzione restituisce una variabile di stato di tipo pwr_tStatus.
Uno stato pari implica che qualcosa ha generato un errore, uno stato dispari che la chiamata e' terminata con successo.
Lo stato puo' essere tradotto in una stringa che fornisce ulteriori informazioni sull'errore. Questo e' ottenuto con l'interfaccia errh che viene descritta piu' avanti.

Leggi e scrivi i valori degli attributi

Se vogliamo leggere o scrivere un attributo dell'oggetto possiamo usare le funzioni gdh_SetObjectInfo () e gdh_GetObjectInfo ().

Una lettura e scrittura del Dv H1-H2-Start puo' assomigliare a quato riportato di seguito.
Si noti che il valore del Dv viene recuperato dall'attributo ActualValue.

```
pwr_tBoolean value;

sts = gdh_GetObjectInfo( "H1-H2-Start.ActualValue", &value, sizeof(value));
if (ODD(sts)) {
  value = !value;
  sts = gdh_SetObjectInfo( "H1-H2-Start.ActualValue", &value, sizeof(value));
}
```

Link diretto agli attributi

I programmi applicativi vengono spesso inseriti in un ciclo infinito, controllando gli attributi nel database e reagendo a determinati cambiamenti.
In questo caso e' preferbile adottare il collegamento diretto all'attributo, cioe' ottenere un puntatore. Questo viene fatto da gdh_RefObjectInfo().
Nell'esempio seguente il programma e' suddiviso in una funzione init() che collega direttamente gli attributi, una funzione scan() contenente le funzioni di supervisione e controllo e una funzione close() che rimuove i collegamenti diretti.

```
class ra_myappl {
  pwr_tBoolean *start_ptr;
  pwr_tRefId dlid;
 public:
  ra_myappl() {}
  void init();
  void scan();
  void close();
 };

void ra_myappl::init()
{
```

```
  sts = gdh_RefObectInfo( "H1-H2-Start.ActualValue", &start_ptr, &dlid,
                          sizeof(*start_ptr));
  if ( EVEN(sts)) exit(0);
}

void ra_myappl::scan()
{
  for (;;) {
    if ( *start_ptr) {
      // Do something...
      cout << "Starting" << endl;

      *start_ptr = 0;

    }
    sleep(1);
  }
}

void ra_myappl::close()
{
  gdh_UnrefObjectInfo( &dlid);
}
```

Nella funzione init() il puntatore start_ptr e' impostato per puntare al valore del Dv H1-H2-Start nel database.

Avvertimento

Nota che i puntatori nel linguaggio C richiedono cautela. Se si utilizzano puntatori aritmetici o indici di array, e' facile indicare la posizione errata nel database e quindi scrivere nella posizione errata.
Cio' puo' causare errori per i quali e' molto difficile trovare la fonte.

Collegamento diretto agli oggetti

gdh_RefObjectInfo () puo' generare, oltre che il collegamento diretto ai singoli attributi, anche il collegamento diretto agli oggetti e agli oggetti attributi.

Supponiamo di impostare i punti in una curva e di visualizzare la curva in un grafico.
Abbiamo il link diretto all'oggetto H1-H2-Curve della classe XyCurve.
Il file include pwr_baseclasses.hpp contiene una classe c++, pwr_Class_XyCurve, per l'oggetto.

```
#include <math.h>
#include "pwr.h"
#include "pwr_baseclasses.hpp"
#include "rt_gdh.h"

class ra_myappl {
  pwr_Class_XyCurve *curve_ptr;
  pwr_tRefId dlid;
 public:
  ra_myappl() {}
  void init();
```

```
  void scan();
  void close();
 };

void ra_myappl::init()
{
  pwr_tStatus sts;
  pwr_tOName name = "H1-H2-Curve";

  // Connect to database
  sts = gdh_Init( "ra_myappl");
  if ( EVEN(sts)) exit(0);

  // Direct link to curve object
  sts = gdh_RefObjectInfo( name, (void **)&curve_ptr, &dlid, sizeof(*curve_ptr));
  if ( EVEN(sts)) exit(0);
}

void ra_myappl::scan()
{
  for ( unsigned int i = 0;;i++) {
    if ( i % 5 == 0) {
      // Calculate x and y coordinates for a sine curve every fifth second
      for ( int j = 0; j < 100; j++) {
        curve_ptr->XValue[j] = j;
        curve_ptr->YValue[j] = 50 + 50 * sin( 2.0 *  M_PI * (j + i) / 100);
      }
      // Indicate new curve to graph
      curve_ptr->Update = 1;
    }
    else if ( i % 5 == 2)
      curve_ptr->Update = 0;
    sleep(1);
    if ( i > 360)
      i = 0;
  }
}

void ra_myappl::close()
{
  gdh_UnrefObjectInfo( dlid);
}


int main()
{
  ra_myappl myappl;

  myappl.init();
  myappl.scan();
  myappl.close();
}
```

Il programma e' compilato e collegato (lincato) con

```
> g++ -g -c  ra_myappl.cpp -o $pwrp_obj/ra_myappl.o -I$pwr_inc -DOS_LINUX=1
      -DOS=linux -DHW_X86=1 -DHW=x86
> g++ -g -o $pwrp_exe/ra_myappl $pwrp_obj/ra_myappl.o $pwr_obj/pwr_msg_rt.o
      -L$pwr_lib -lpwr_rt -lpwr_co -lpwr_msg_dummy -lrt
```

Piu' avanti vedremo come utilizzare make per la compilazione ed il link.

Quando apriamo il grafico dell'oggetto per l'oggetto Curva H1-H2, possiamo studiarne il risultato.

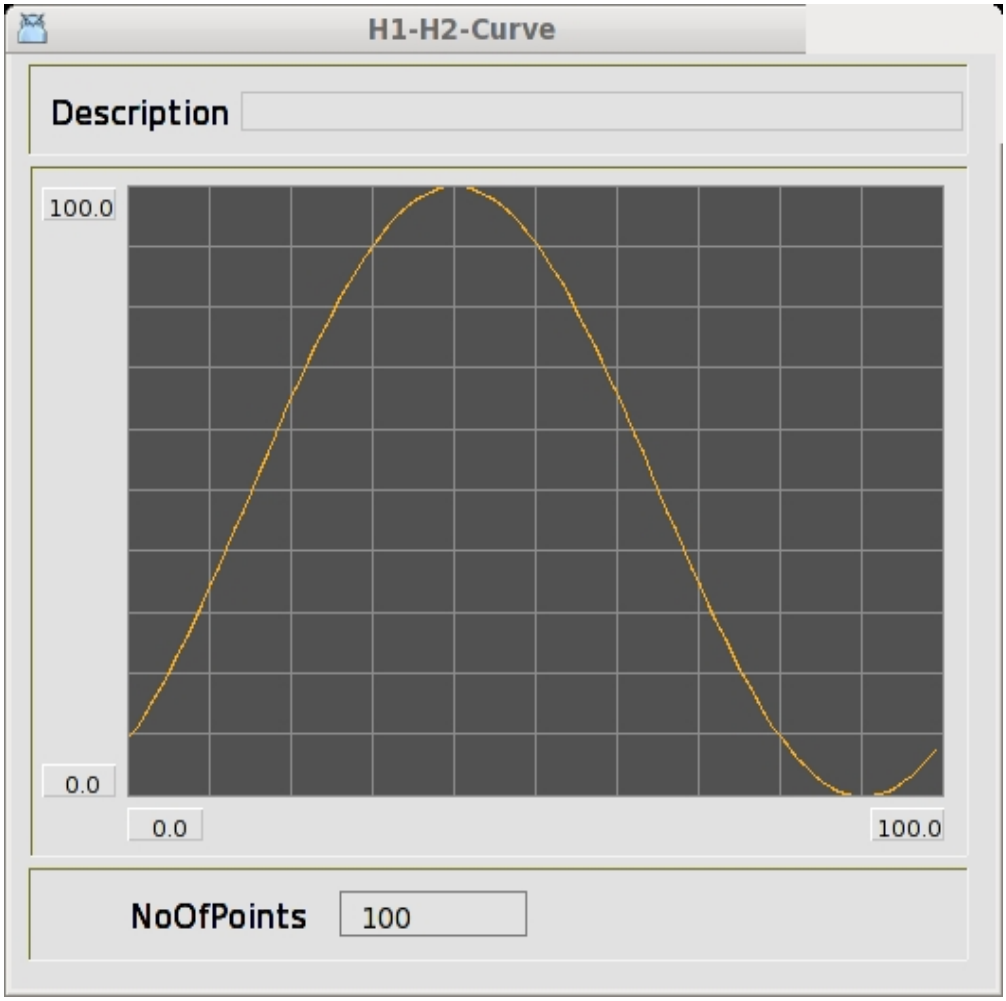


Fig Oggetto grafico per l'ogetto curva.

15.2 Console di log

Log nella console di log

Console di log

Il log della console contiene i messaggi di log dei processi di sistema.
Se c'e' qualcosa in errore nel sistema, dovresti cercare nel log della console ed esaminare se qualche processo ha registrato messaggi di errore.

Il log degli errori e' un file di testo su $pwrp_log, pwr_'nodename'.log, che puo' essere aperto anche in rt_xtt da System / SystemMessages.
Le registrazioni hanno cinque livelli di gravita', fatali, errori, avvisi, informazioni e successo.
Fatale e errore sono colorati di rosso, giallo di avvertimento, informazioni e successo di colore verde.

Anche le applicazioni possono scrivere sul log della console.
Prima ci si collega al log della console con errh_Init(), quindi si possono scrivere messaggi con severita' diversa utilizzando errh_Fatal(), errh_Error(), errh_Warning(), errh_Info() e errh_Success(),

La funzione errh_Init() viene chiamata prima di gdh_Init() e ha come argomento un nome dell'applicazione e un indice dell'applicazione fornito come errh_eAnix_Appl1, errh_eAnix_Appl2 ecc.
Ogni applicazione dovrebbe avere un indice di applicazione univoco all'interno del nodo.

```
#include "rt_errh.h"

sts = errh_Init( "ra_myappl", errh_eAnix_Appl1);
```

Alle funzioni di log si invia la stringa che deve essere scritta nel log, ad es.

errh_Error("Something went wrong");

La stringa puo' anche funzionare come un'istruzione di formattazione contenente %s per formattare le stringhe,%f per float e %d per intero, vedere printf per maggiori informazioni.

```
errh_Error( "Number is to high: %d", n);
```

Il formato %m converte un codice di stato nel testo corrispondente

```
catch ( co_error e) {
  errh_Error( "Error status: %m", e.sts());
}
```

Stato dell'applicazione

Ogni applicazione ha una parola di stato nell'oggetto $ Node. Si trova nell'attributo ProcStatus[] nell'elemento applicationindex + 20.
Lo stato dovrebbe riflettere la condizione dell'applicazione e viene impostato dall'applicazione stessa dalla funzione errh_SetStatus ().

```
errh_SetStatus( PWR__ARUN);
```

PWR__ARUN e' definito in rt_pwr_msg.h e lincato al testo "Application running".

Altri codici di stato utili sono

PWR__APPLSTARTUP "Application starting up" (info)
PWR__APPLRESTART "Application restaring" (info)
PWR__APPLTERM   "Application terminated" (fatal)

Nell'oggetto nodo esiste anche un SystemStatus che e' un tipo di somma di tutto lo stato del server e dei processi dell'applicazione.

Lo stato del server o dell'applicazione piu' grave viene inserito in systemstatus.

Watchdog

Un'applicazione che ha chiamato errt_Init() e' supervisionata dal sistema. Dovrebbe chiamare ciclicamente aproc_TimeStamp, oppure lo stato dell'applicazione e' impostato su "Process timeout" (fatale).
Il timeout per le applicazioni e' 5 s.

Oggetto dell'applicazione

e' possibile creare un oggetto applicazione per le applicazioni. Viene posizionato nella gerarchia dei nodi sotto l'oggetto $ Node ed e' di classe Application.

L'oggetto dell'applicazione e' registrato dalla funzione aproc_RegisterObject () che ha come oggetto l'identita' dell'oggetto per l'oggetto.

```
pwr_tObjid aoid;
pwr_tOName name = "Nodes-MyNode-ra_myappl";

sts = gdh_NameToObjid( name, &aoid);
if (EVEN(sts)) throw co_error(sts);

sts = aproc_RegisterObject( aoid);
if (EVEN(sts)) throw co_error(sts);
```

Grafico di stato

L'applicazione viene visualizzata nel grafico di stato del nodo se l'applicazione ha allegato errh e l'oggetto applicazione e' registrato. Verra' mostrato sotto "Applicazione" sulla riga corrispondente all'indice dell'applicazione. Nel grafico vengono visualizzati lo stato dell'applicazione e il messaggio di registro ultimo/piu' grave.

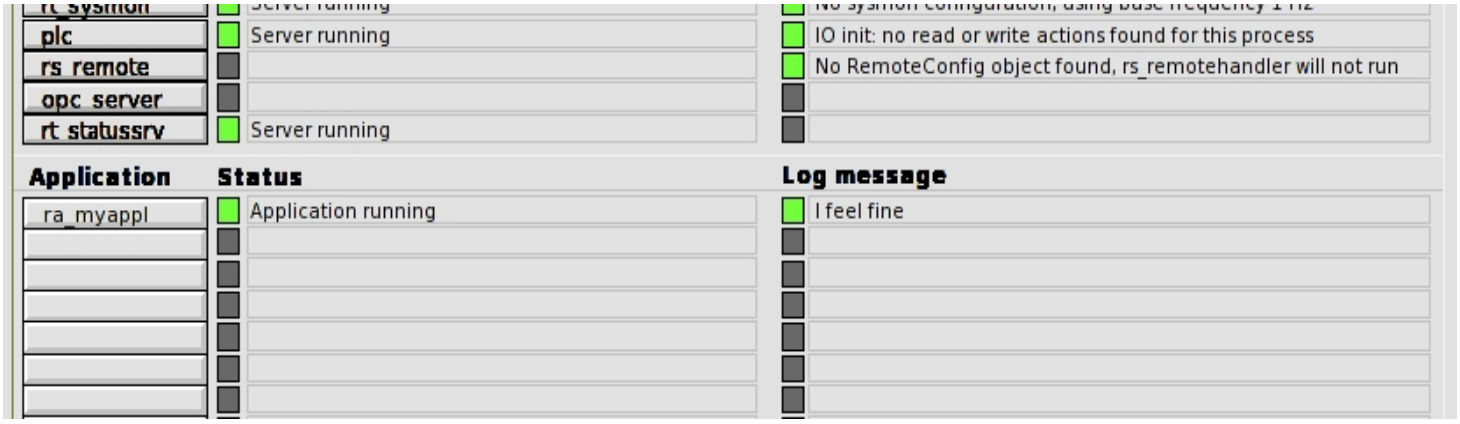


Fig Dettagli per il grafico di stato che mostra lo stato e il messaggio di registro per l'applicazione.

Se il processo e' interrotto, lo stato e' impostato su timeout. Cio' influenzera' anche lo stato del sistema.

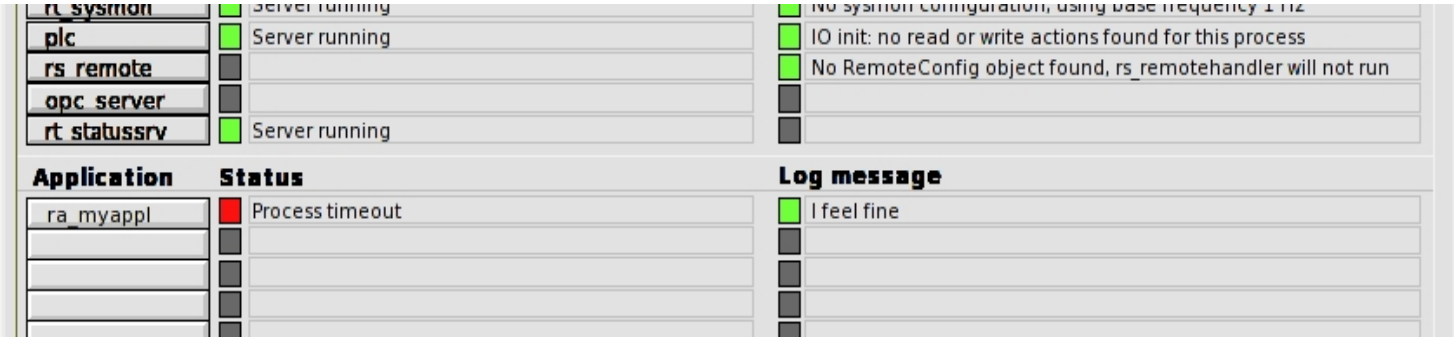


Fig L'applicazione e stata fermata.

Esempio

Nell'esempio abbiamo esteso il precedente programma con la curva xy, e abbiamo inserito le chiamate per impostare lo stato dell'applicazione, accedere al log della console e registrare l'oggetto dell'applicazione.

```
#include <math.h>
#include <iostream>
#include "pwr.h"
#include "pwr_baseclasses.hpp"
#include "rt_gdh.h"
#include "rt_errh.h"
#include "rt_aproc.h"
#include "rt_pwr_msg.h"
#include "co_error.h"

class ra_myappl {
  pwr_Class_XyCurve *curve_ptr;
  pwr_tRefId dlid;
 public:
  ra_myappl() {}
  void init();
  void scan();
  void close();
};

void ra_myappl::init()
{
  pwr_tStatus sts;
  pwr_tOName name = "H1-H2-Curve";
  pwr_tObjid aoid;

  // Init errh with anix 1
  sts = errh_Init( "ra_myappl", errh_eAnix_appl1);
  if ( EVEN(sts)) throw co_error(sts);

  // Write message to consolelog and set application status
  errh_Info( "I feel fine");
  errh_SetStatus( PWR__APPLSTARTUP);

  // Connect to database
  sts = gdh_Init( "ra_myappl");
  if ( EVEN(sts)) throw co_error(sts);
```

```
  // Register application object
  sts = gdh_NameToObjid( "Nodes-Saturnus7-ra_myappl", &aoid);
  if ( EVEN(sts)) throw co_error(sts);

  aproc_RegisterObject( aoid);

  // Directlink to curve object
  sts = gdh_RefObjectInfo( name, (void **)&curve_ptr, &dlid, sizeof(*curve_ptr));
  if ( EVEN(sts)) throw co_error(sts);

  errh_SetStatus( PWR__ARUN);
}

void ra_myappl::scan()
{
  for ( unsigned int i = 0;;i++) {
    // Notify that we are still alive
    aproc_TimeStamp();

    if ( i % 5 == 0) {
      for ( int j = 0; j < 100; j++) {
        curve_ptr->XValue[j] = j;
        curve_ptr->YValue[j] = 50 + 50 * sin( 2.0 *  M_PI * (j + i) / 100);
      }
      curve_ptr->Update = 1;
    }
    else if ( i % 5 == 2)
      curve_ptr->Update = 0;
    sleep(1);
    if ( i > 360)
      i = 0;
  }
}

void ra_myappl::close()
{
  gdh_UnrefObjectInfo( dlid);
}


int main()
{
  ra_myappl myappl;

  try {
    myappl.init();
  }
  catch ( co_error e) {
    errh_Fatal( "ra_myappl terminated, %m", e.sts());
    errh_SetStatus( PWR__APPLTERM);
    exit(0);
  }
  myappl.scan();
```

```
  myappl.close();
}
```

15.3 Avvio dell'applicazione

Un'applicazione che deve essere avviata all'avvio del runtime di Proview viene inserita nel file dell'applicazione.
Questo si trova su $ pwrp_load ed e' chiamato ld_appl_'nodename '_' qbus'.txt, ad es.

```
$pwrp_load/ld_appl_mynode_999.txt
```

Nel file si inserisce una riga per ogni applicazione che deve essere avviata

```
# id       name      [no]load [no]run file       prio  [no]debug  "arg"
ra_myappl, ra_myappl, noload,  run,   ra_myappl, 12,   nodebug,   ""
```

15.4 Ricezione di eventi di sistema

Proview trasmette messaggi quando vengono succedono determinati eventi, ad es. quando si procede ad un riavvio software o quando l'ambiente di runtime viene arrestato.
Un'applicazione puo' ascoltare questi messaggi, ad esempio per terminare quando Proview viene terminato. I messaggi sono ricevuti da Qcom.
Per ricevere questi messaggi si crea una coda Qcom per la nostra applicazione e si associa questa coda alla coda che invia i messaggi.

```
#include "rt_qcom.h"
#include "rt_ini_event.h"
#include "rt_qcom_msg.h"

qcom_sQid qid = qcom_cNQid;
qcom_sQid qini;
qcom_sQattr qAttr;

if ( !qcom_Init(&sts, 0, "ra_myappl")) {
  throw co_error(sts);

// Create a queue to receive stop and restart events
qAttr.type = qcom_eQtype_private;
qAttr.quota = 100;
if ( !qcom_CreateQ(&sts, &qid, &qAttr, "events"))
  throw co_error(sts);

// Bind to init event queue
qini = qcom_cQini;
if ( !qcom_Bind(&sts, qid, &qini))
  throw co_error(sts);
```

Ad ogni scansione si legge la coda con qcom_Get() per vedere se sono arrivati nuovi messaggi.

E' inoltre possibile utilizzare il timeout in qcom_Get() per attendere la scansione successiva.
Nell'esempio seguente, viene gestito l'evento terminate, ma anche gli eventi oldPlcStop e swapDone che indicano l'inizio e la fine di un riavvio.
Bisogna fare questa operaozione solo se si desidera che l'applicazione scopra i nuovi oggetti di nuove configurazioni dopo un soft restart.

```
int tmo = 1000;
char mp[2000];
qcom_sGet get;
int swap = 0;

for (;;) {
  get.maxSize = sizeof(mp);
  get.data = mp;
  qcom_Get( &sts, &qid, &get, tmo);
  if (sts == QCOM__TMO || sts == QCOM__QEMPTY) {
    if ( !swap)
      // Do the normal thing
      scan();
  }
  else {
    // Ini event received
    ini_mEvent  new_event;
    qcom_sEvent *ep = (qcom_sEvent*) get.data;

    new_event.m  = ep->mask;
    if (new_event.b.oldPlcStop && !swap) {
      errh_SetStatus( PWR__APPLRESTART);
      swap = 1;
      close();
    } else if (new_event.b.swapDone && swap) {
      swap = 0;
      open();
      errh_SetStatus( PWR__ARUN);
    } else if (new_event.b.terminate) {
      exit(0);
    }
  }
}
```

Se sei interessato solo a fermare il processo quando Proview viene rimosso, c'e' un modo piu' semplice per ucciderlo. Puoi mettere un file di script, pwrp_stop.sh, su $pwrp_exe dove uccidi il processo.

```
killall ra_myappl
```

15.5 Classe base per applicazioni rt_appl

La classe di base rt_appl contiene molte delle inizializzazioni e la supervisione degli eventi descritti sopra. Generando una sottoclasse di rt_appl non devi scrivere alcun codice per gestire gli eventi, viene fatto tutto da rt_appl.

rt_appl contiene tre funzioni virtuali che devono essere implementate dalla sottoclasse, open(), close() e scan().
open() viene utilizzato durante l'inizializzazione per dirigere il collegamento ad attributi e oggetti ,
scan() viene richiamato ciclicamente con cycletime fornito, e in close() si rimuovono i collegamenti diretti.

rt_appl gestisce questo:

- Inizializzazione di gdh, errh e qcom
- Impostazione delleo stato applicazione all'avvio e riavvio
- Gestione degli eventi per soft restart e terminazione
- Timestamp per evitare il timeout

Questo esempio mostra l'applicazione ra_appl sottoclasse di rt_appl.

```
class ra_appl : public rt_appl {
public:
 ra_appl() : rt_appl( "ra_appl", errh_eAnix_appl1) {}
 void open();
 void close();
 void scan();
};

void ra_appl::open()
{
 // Link to database objects
}

void ra_appl::close()
{
 // Unlink to database objects
}

void ra_appl::scan()
{
 // Do something
}

int main()
{
 ra_appl appl;

 appl.init();
 appl.register_appl( "Nodes-MyNode-MyAppl");

 appl.mainloop();
}
```

15.6 Invio di allarmi e messaggi

Da un'applicazione e' possibile inviare allarmi e messaggi alla lista degli allarmi e alla lista degli eventi dell'operatore. Innanzitutto e' necessario connettersi al monitor degli eventi con mh_ApplConnect() che accetta l'identita' dell'oggetto per l'oggetto applicazione come primo argomento.
Per le sottoclassi della classe rt_appl, questa identita' viene recuperata con apploid ().

```
#include "rt_mh_appl.h"

pwr_tUInt32 num;
pwr_tOid aoid = apploid();
sts = mh_ApplConnect( aoid, mh_mApplFlags(0), "", mh_eEvent_Info, mh_eEventPrio_A,
   mh_mEventFlags_Bell, "", &num);
  if (EVEN(sts)) throw co_error(sts);
```

Quindi possiamo inviare allarmi con mh_ApplMessage ().

```
mh_sApplMessage msg;
pwr_tUInt32 msgid;

memset( &msg, 0, sizeof(msg));
msg.EventFlags = mh_mEventFlags(mh_mEventFlags_Returned |
      mh_mEventFlags_NoObject |
      mh_mEventFlags_Bell);
time_GetTime( &msg.EventTime);
strcpy( msg.EventName, "Message from ra_myappl");
strcpy( msg.EventText, "I'm up and running now !");
msg.EventType = mh_eEvent_Alarm;
msg.EventPrio = mh_eEventPrio_B;

sts = mh_ApplMessage( &msgid, &msg);
if (EVEN(sts)) throw co_error(sts);
```

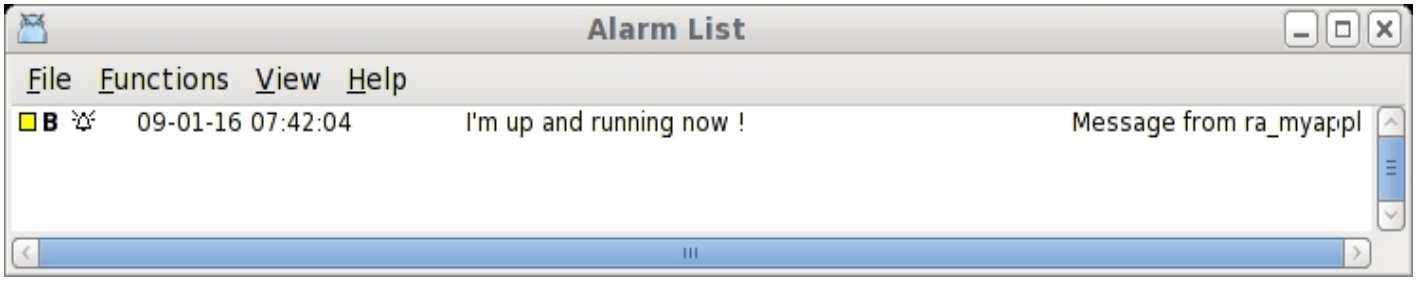


Fig The alarm in the alarmlist.

15.7 Comunicare con altri processi

Il protocollo Proview per la comunicazione tra processi puo' essere utilizzato anche dalle applicazioni.
La comunicazione puo' essere

- tra processi nello stesso nodo
- tra processi in diversi nodi che appartengono allo stesso progetto

- tra i processi nei nodi che appartengono a progetti diversi, se il progetto ha lo stesso bus Qcom. In questo caso i nodi devono essere configurati dagli oggetti FriendNodeConfig dove Connection e' impostato su QcomOnly.

Maggiori informazioni su Qcom nella Guida di riferimento Qcom.

15.8 Recuperare dati da una stazione di archiviazione

I dati memorizzati in una stazione di archiviazione Proview possono essere recuperati dall'interfaccia client sevcli.
Innanzitutto si avvia sevcli con sevcli_init() e si specifica da quale stazione di memoria si desidera recuperare i dati con sevcli_set_servernode().

```
sevcli_tCtx sevctx;
char server_node[40] = "MyStorageStation";

if ( !sevcli_init( &sts, &sevctx))
  throw co_error(sts);

if ( !sevcli_set_servernode( &sts, sevctx, server_node))
  throw co_error(sts);
```

Quindi puoi recuperare i dati con sevcli_get_itemdata (). I dati sono identificati dall'identita' dell'oggetto e dal nome dell'attributo. Bisogna indicare anche l'intervallo di tempo per i dati che devono essere recuperati e il numero massimo di punti.

```
pwr_tTime *time_buf;
void *value_buf;
pwr_tTime from = pwr_cNTime;
pwr_tTime to = pwr_cNTime;
int rows;
pwr_eType vtype;
unsigned int vsize;
pwr_tOName name = "H1-H2-Temperature";
pwr_tOName aname = "ActualValue";
pwr_tObjid oid;
char timstr[40];

sts = gdh_NameToObjid( name, &oid);
if (EVEN(sts)) throw co_error(sts);

if ( !sevcli_get_itemdata( &sts, sevctx, oid, aname, from, to, 1000,
                           &time_buf, &value_buf, &rows, &vtype, &vsize))
  throw co_error(sts);

for ( int i = 0; i < rows; i++) {
  time_AtoAscii( &time_buf[i], time_eFormat_DateAndTime, timstr, sizeof(timstr));

  cout << timstr << " " << ((pwr_tFloat32 *)value_buf)[i] << endl;
}

free( time_buf);
free( value_buf);
```

Infine si chiama sevcli_close() per disconnettere il nodo del server.

```
sevcli_close( &sts, sevctx);
```

15.9 Gestione degli I/O

Se un'applicazione richiede dati I/O veloci e sincronizzati, puo' funzionare direttamente in antagonismo al sistema I/O e chiamare le routine di I/O per leggere e scrivere gli I/O in autonomia.

L'inizializzazione viene eseguita con la funzione io_init(), a cui viene fornito un argomento di processo.
Il processo identifica le unita' I/O (agente, rack o scheda) gestite da un processo specifico.
Ogni oggetto I/O ha un attributo Process e se questo corrisponde al processo inviato come argomento a io_init(), l'unita' sara' gestita dall'applicazione.
Se una scheda viene gestita da un'applicazione, anche il rack e l'agente della scheda devono essere gestiti dall'applicazione.

Poiche' l'attributo Process e' una maschera di bit, un'unita' puo' essere gestita da diversi processi impostando diversi bit nella maschera. Se, ad esempio, si hanno diverse schede in un rack e alcune schede devono essere gestite dal processo plc e alcune da un'applicazione, l'unita' rack deve essere gestita sia dal plc che dall'applicazione.
Se funziona, per gestire un'unita' da piu' processi, dipende da come vengono scritti i metodi I/O per l'unita'.
Ad esempio per Profibus, non e' possibile dividere la gestione degli slave in processi diversi.

```
#include rt_io_base.h

io_tCtx io_ctx;

sts = io_init( io_mProcess_User, pwr_cNOid, &io_ctx, 0, scantime);
if ( EVEN(sts)) {
  errh_Error( "Io init error: %m", sts);
  throw co_error(sts);
}
```

La lettura viene eseguita con io_read() che legge i dati dalle unita' I/O e inserisce i dati nei segnali collegati all'unita'.
L'applicazione preferibilmente si collega direttamente a questi segnali, e anche ai segnali delle unita' di uscita.
Le unita' di output sono scritte con la funzione io_write().

```
sts = io_read( io_ctx);

sts = io_write( io_ctx);
```

15.10 Costruire un'applicazione

Un'applicazione c++ deve essere compilata e lincata, e puoi usare make per farlo. Proview contiene un file di regole, $pwr_exe/pwrp_rules.mk, che contiene regole per la compilazione.

Un makefile per l'applicazione ra_myappl nella directory $pwrp_src/myappl puo' assomigliare a questo ($pwrp_src/myappl/makefile):

```
ra_myappl_top : ra_myappl

include $(pwr_exe)/pwrp_rules.mk

ra_myappl_modules : \
  $(pwrp_obj)/ra_myappl.o \
  $(pwrp_exe)/ra_myappl

ra_myappl : ra_myappl_modules
 @ echo "ra_myappl built"

#
# Modules
#

$(pwrp_obj)/ra_myappl.o : $(pwrp_src)/myappl/ra_myappl.cpp \
                          $(pwrp_src)/myappl/ra_myappl.h

$(pwrp_exe)/ra_myappl : $(pwrp_obj)/ra_myappl.o
 @ echo "Link $(tname)"
 @ $(ldxx) $(linkflags) -o $(target) $(source) -lpwr_rt -lpwr_co \
  -lpwr_msg_dummy -lrpcsvc -lpthread -lm -lrt
```

Il makefile viene eseguito posizionando sulla directory e scrivendo make

```
make
```

e' anche possibile inserire il comando build nell'oggetto Application dell'applicazione nell'attributo BuildCmd.
In questo caso il comando di build e' :

```
make --directory $pwrp_src/myappl -f makefile
```

Questo comando viene eseguito quando il nodo viene creato dal configuratore. Questo e' un modo per garantire che tutte le applicazioni vengano aggiornate quando il nodo viene creato.

15.11 Applicazioni Java

Alcune API esistono anche per java nella forma delle classi Gdh, Errh e Qcom.
Di seguito e' riportato un esempio di un'applicazione java che collega il database in tempo reale e legge e scrive un attributo.

```
import jpwr.rt.*;

public class MyJappl {
  public MyJappl() {
```

```
    Gdh gdh = new Gdh( null);

    CdhrBoolean rb = gdh.getObjectInfoBoolean( "H1-H2-Start.ActualValue");

    PwrtStatus rsts = gdh.setObjectInfo( "H1-H1-Start.ActualValue",
         !rb.value);
  }

  //Main method
  public static void main(String[] args) {
    new MyJappl();
  }
}
```

Per compilare ed eseguire devi inserire $pwr_lib/pwr_rt.jar e la directory di lavoro in CLASSPATH, e $pwr_exe in LD_LIBRARY_PATH

```
> export CLASSPATH=$pwr_lib/pwr_rt.jar:$pwrp_src/myjappl
> export LD_LIBRARY_PATH=$pwr_exe
```

Compilare con

```
> javac MyJappl.java
```

ed eseguire con

```
> java MyJappl
```

Per l'avvio automatico dell'applicazione, creare uno shell script che esporta CLASSPATH e LD_LIBRARY_PATH e avvia l'applicazione java.
Lo script deve essere inserito nel appl-file allo stesso modo di un'applicazione c.

16 Creazione della grafica di processo

Questo capitolo descrive come si creano le grafiche di processo.

La grafica di processo viene disegnata e configurata nell'editor di ge.

L'editor Ge

Ge viene aperto dal menu nel navigatore: 'Functions/Open Ge'. Consiste di

- un pannello degli strumenti
- un'area di lavoro
- una tavolozza dei sottografici
- una tavolozza di colori
- una finestra che mostra la gerarchia dell'impianto
- una finestra di navigazione

Immagine di sfondo

Un'immagine di sfondo viene disegnata con oggetti di base come rettangoli, cerchi, linee, polilinee e testo.
Questi si trovano nel pannello degli strumenti. Creare un oggetto base attivando il pulsante nel pannello degli strumenti e trascinando o facendo clic su MB1 nell'area di lavoro. Se l'oggetto di base deve essere riempito, selezionare l'oggetto e attivare il riempimento nel pannello degli strumenti. Cambia il colore di riempimento selezionando l'oggetto e fai clic sul colore desiderato nella tavolozza dei colori.
Modificare il colore del bordo facendo clic con MB2 nella tavolozza dei colori e il colore del testo facendo clic su Maiusc/clic su MB1.

Sottografi

Un sottografo e' un componente grafico, ad es. una valvola, un motore, un pulsante.
Per creare un sottografo, selezionare un sottografo nella tavolozza del sottografo e fare clic su MB2 nell'area di lavoro.

gruppi

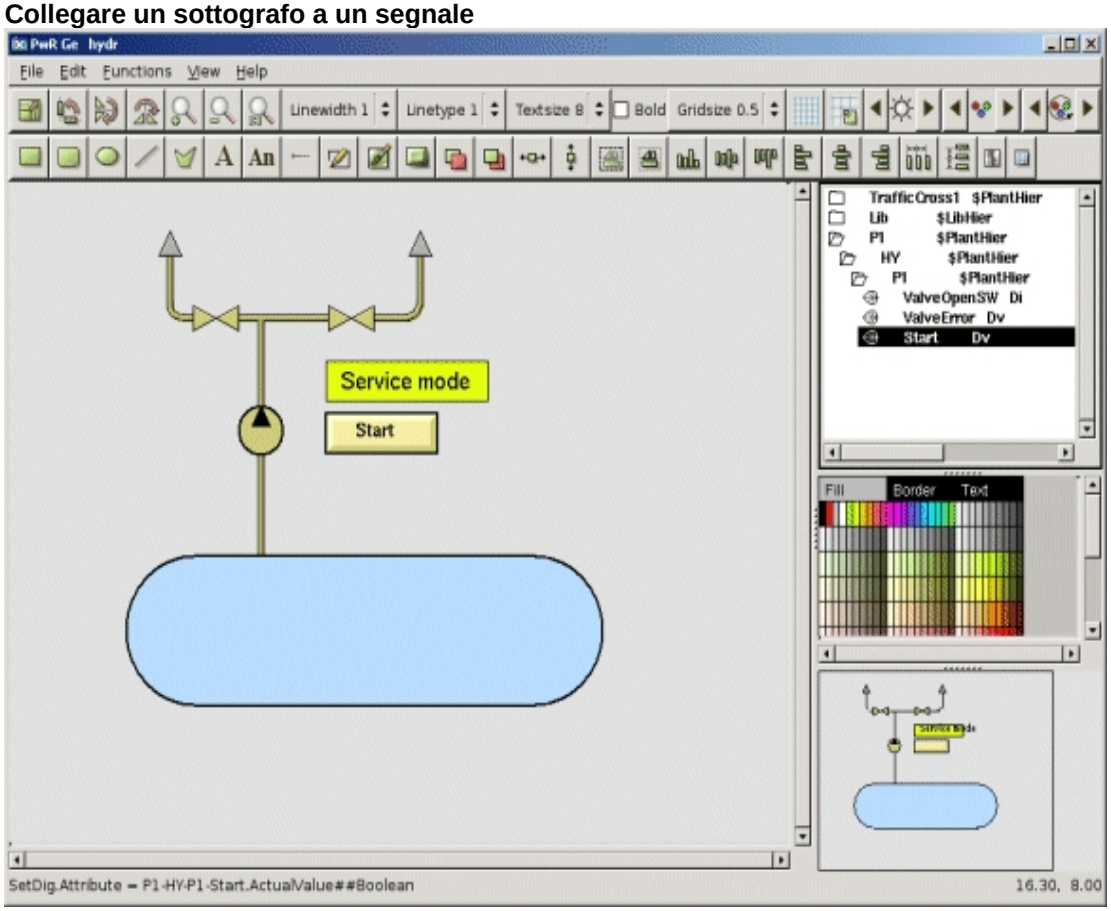
Oggetti base e sottografi possono essere raggruppati selezionandoli e attivando 'Functions/Group' nel menu.

Dinamica

Sottografi e gruppi hanno proprieta' dinamiche, cioe' possono essere collegati ai segnali nel database di runtime e cambiare colore, posizione o forma a seconda dei valori dei segnali.
Un sottografo spesso ha un comportamento dinamico predefinito, ad esempio un indicatore si sposta tra due colori.
Devi solo collegare l'indicatore a un segnale digitale per farlo funzionare.
Questo viene fatto selezionando un segnale nella finestra della gerarchia dell'impianto e facendo clic sulla valvola con Ctrl/DoubleClick MB1.

Un pulsante ha una proprieta' di azione; che setta, resetta o commuta un segnale nel database.
Un pulsante con un'azione set viene creato selezionando un PulsanteSet nella tavolozza dei sottografi e facendo clic MB2 nell'area di lavoro.
Il segnale che deve essere impostato e' collegato come sopra, selezionando il segnale e facendo clic con Ctrl/DoubleClick MB1 sul pulsante.
Nell'editor di oggetti e' possibile assegnare un testo di pulsante.

Collegare un sottografo a un segnale



Un buGroups ha anche una proprieta' dinamica, cioe' puo' cambiare colore, spostarsi o eseguire un'azione.
Non hanno alcuna azione predefinita o colore predefinito, come i sottografi.
Devi assegnare questo per ogni gruppo.

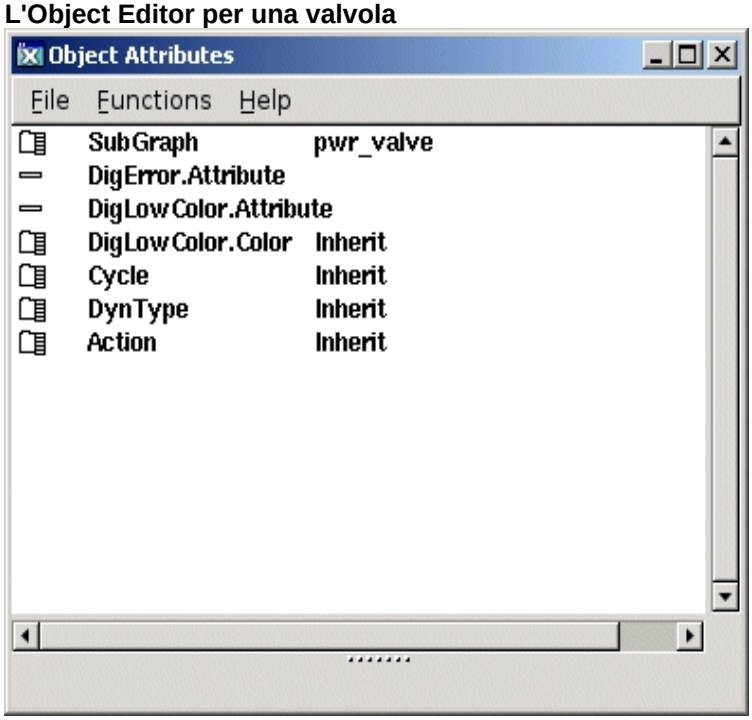
L'editor di oggetti

Oggetti di base, oggetti del sottografo e gruppi hanno proprieta', che vengono modificate dall'editor di oggetti.
L'editor di oggetti viene aperto selezionando l'oggetto e attivando 'Function/Object attributes' nel menu. Aprendo l'editor di oggetti per il pulsante di cui sopra, e' possibile ad esempio inserire il testo che viene visualizzato nel pulsante nell'attributo 'Text'.

Se un sottografo ha una dinamica piu' avanzata, ad esempio lo spostamento tra piu' colori,e' spesso necessario collegarlo a piu' segnali. Se apri l'editor di oggetti per una valvola, vedi che puo' essere collegato a due attributi, "DigError.Attribute" e "DigLowColor.Attribute". L'attributo DigError indica che qualcosa non va, e se questo segnale e' vero, la valvola e' colorata in rosso. L'attributo DigLowColor e' collegato all'interruttore aperto della valvola.

Se questo segnale e' falso, la valvola e' colorata con il colore indicato in "DigLowColor.Color". Se il segnale e' true, mantiene il colore indicato nell'editor.
I segnali dei due attributi vengono inseriti selezionando rispettivamente ciascun segnale nella gerarchia dell'impianto e facendo clic con Ctrl/Doubleclick MB1 sulla riga dell'attributo dell'attributo.
Il colore 'DigLowColor' viene indicato aprendo l'attributo e selezionando uno dei 300 colori. I colori hanno lo stesso ordine della tavolozza dei colori e con un po 'di pratica possono essere identificati dal nome.

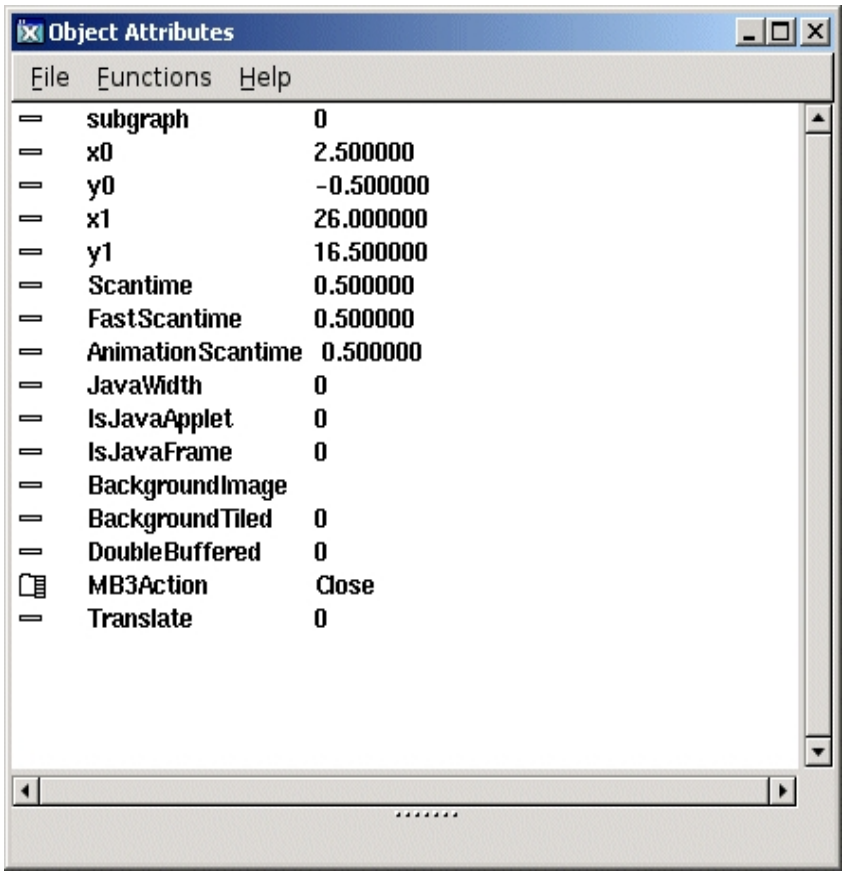
L'Object Editor per una valvola



Bordi del grafico

L'area di disegno in Ge e' illimitata in ogni direzione, quindi prima di salvare il grafico, devi indicare i bordi del grafico. Aprire gli attributi del grafico con gli attributi 'File/Graph' nel menu.
Misurare le coordinate dell'angolo in alto a destra e inserire come x0 e y0, quindi misurare le coordinate dell'angolo in basso a sinistra e inserire come x1 e y1.
La misurazione viene eseguita posizionando il cursore in posizione e leggendo le coordinate nella riga del messaggio.

Attributi della grafica



Configurazione nel workbench

L'oggetto XttGraph

Ad ogni grafica impianto appartiene un oggetto XttGraph. Questo oggetto e' in genere un figlio dell'oggetto OpPlace (posto oggetto operatore) per il nodo, sul quale verranno visualizzati i grafici.
E' necessario creare un oggetto XttGraph per ciascun nodo, sul quale verranno visualizzati i grafici.
Tuttavia, e' necessario disporre di un solo file grafico.
I seguenti attributi nell'oggetto grafico devono essere impostati su valori appropriati:

- Azione, il nome del file pwg con il tipo di file, ad esempio "hydr.pwg"
- Titolo, il titolo della finestra del grafico
- ButtonText, testo del pulsante nella finestra dell'operatore

L'oggetto XttGraph contiene altri attributi che ad es. ti aiuta a personalizzare la posizione e le dimensioni della grafica dell'impianto. Questi attributi sono descritti in dettaglio nel Manuale di riferimento degli oggetti Proview.
Vedi XttGraph nel Manuale di riferimento agli oggetti

17 Interfaccia operatore Java

Oltre all'ambiente operatore ordinario in X windows, Proview contiene anche un ambiente operatore scritto in java. Il vantaggio di java e' che e' eseguibile in un browser Web e su molte piattaforme. Lo svantaggio e' che e' piu' lento e ha una funzionalita' limitata.

Installazione di java

Stazione di sviluppo

Per poter esportare i grafici dei processi su java, e' necessario installare Java Development Kit, jdk. Scarica un jdk adatto da java.sun.com e definisci la variabile di ambiente jdk nella directory in cui jdk e' scompattato, ad es. /usr/local/jdk1.6.0_10, e metti $jdk/bin in PATH. Definisci anche jdk_home in $jdk/bin.

```
export jdk=/usr/local/jdk1.6.0_10
export PATH=$PATH:$jdk/bin
export jdk_home=$jdk/bin
```

Stazioni di processo e operatore

Sulle stazioni di processo e operatore e' necessario installare Java Runtime Environment, jre. Scarica da java.sun.com e scompatta sotto /usr/local. Anche qui dovrebbero essere definiti jdk e jdk_home (questo e' fatto da /etc/pwrp_profile)

```
export jdk=/usr/local/jre1.6.0_10
export PATH=$PATH:$jdk/bin
export jdk_home=$jdk/bin
```

Panoramica

Web browser

L'ambiente java puo' essere avviato in un browser Web, inserendo un URL nella homepage del progetto.
Le informazioni su oggetti, allarmi, ecc. vengono recuperate da tre processi server che devono essere avviati sul nodo da cui viene aperta la homepage, rt_webmon, rt_webmonmh e rt_webmonelog. La configurazione di un oggetto WebHandler nella gerarchia dei nodi e' necessaria per avviare i processi del server.

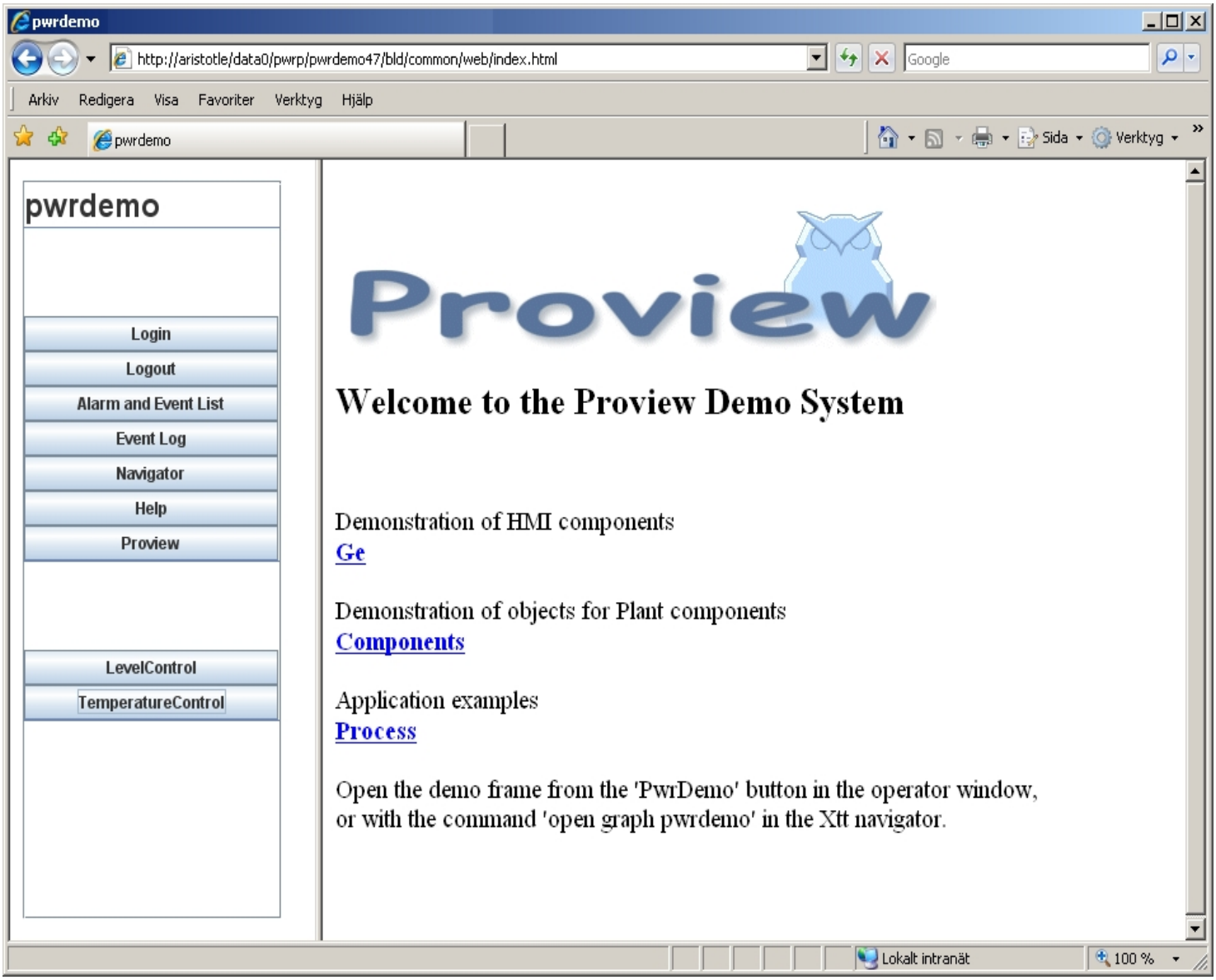


Fig Homepage per un progetto

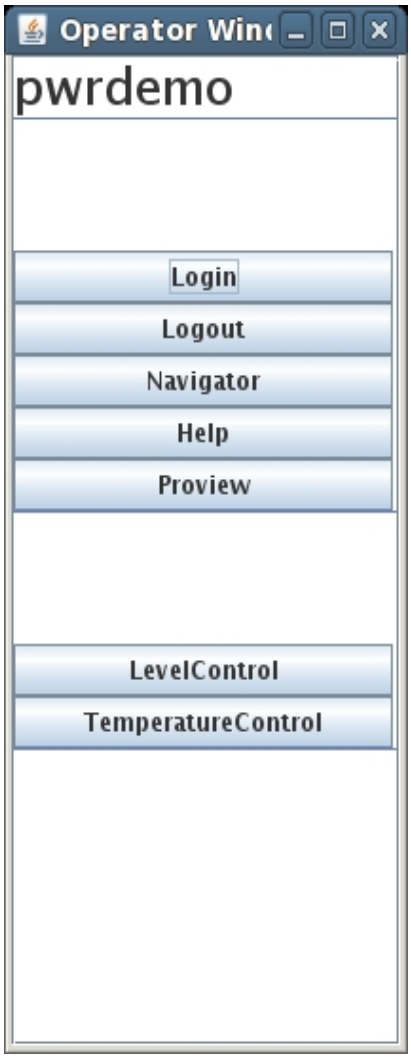
Locale

L'ambiente java puo' anche essere avviato localmente sul nodo su cui e' avviato il runtime Proview. I processi del server per l'interfaccia web non devono essere avviati, le informazioni vengono recuperate direttamente dal database in tempo reale.

L'ambiente java locale contiene un menu, ma non una cornice destra come la pagina web. E' possibile visualizzare solo il grafico Ge esportato come frame java, non le applet java. Le funzioni per la visualizzazione di allarmi ed eventi e la ricerca nel registro eventi non sono implementate nell'ambiente locale.

L'ambiente operatore viene avviato con il comando

```
> jpwr_opwind.sh
```



Menu dell'operatore nell'ambiente locale

Configurazione

Configurazione dei processi del server

Se e' presente un oggetto WebHandler, vengono avviati i processi del server per l'interfaccia Web.

Configurazione del menu dell'operatore

La configurazione del menu dell'operatore viene effettuata nella gerarchia dei nodi con gli attributi nell'oggetto OpPlaceWeb e con l'oggetto WebGraph e WebLink sotto questo oggetto.

OpPlaceWeb

Nell'oggetto OpPlaceWeb sono configurati i pulsanti di menu per le funzioni standard.

Abilita lingua
Visualizza un pulsante per la selezione della lingua nel menu. La selezione della lingua deve essere fatta prima che vengano aperte varie finestre. Le finestre gia' create non sono

interessate da una nuova selezione della lingua.

EnableLogin
Visualizza i pulsanti per il login e il logout nel menu. Il pulsante di accesso consente di accedere come utente Proview con privilegi specificati che definiscono gli utenti l'accesso al sistema.

EnableAlarmList
Visualizza un pulsante nel menu per aprire una finestra con elenchi di allarmi e eventi.

EnableEventLog
Visualizza un pulsante nel menu per cercare gli eventi nel registro eventi.

EnableNavigator
Visualizza un pulsante nel menu per aprire il navigatore.

WebGraph
L'oggetto WebGraph configura i pulsanti nel menu per aprire la grafica di processo. Gli oggetti WebGraph sono posizionati sotto l'oggetto OpPlaceWeb.

Un oggetto WebGraph apre un grafico Ge come un frame Java, cioe' in una finestra separata. Nell'editor di ge, devi specificare che il grafico deve essere esportato come un frame java impostando IsJavaFrame su 1 in Attributi grafico.
Se il grafico deve essere aperto come applet, viene utilizzato un oggetto WebLink, vedi sotto.

Nell'attributo Nome viene indicato il nome della classe java per il grafico.
Normalmente questo e' lo stesso del nome del grafico, ma lettera di blocco nel primo carattere.
E' possibile esportare il grafico con un altro nome di classe, selezionando 'Export As'.
Questo deve essere fatto quando il graphname contiene caratteri locali che non sono validi in un nome di classe java, o se il grafename inizia con una cifra.

Nell'attributo Testo, viene indicato il testo del pulsante.

Collegamento web
Un oggetto WebLink configura un pulsante nel menu per aprire un URL arbitrario. In WebTarget viene indicato dove viene inserita la nuova pagina, in un frame separato o nel frame destra o nel frame principale (sia il menu che la cornice destra).

Aprire un grafico Ge come applet Java
Un oggetto WebLink puo' anche essere utilizzato per aprire un grafico Ge come applet, ad esempio nella cornice destra.
Nell'editor di ge si specifica il grafico da esportare come frame java impostando IsJavaApplet su 1 in GraphAttributes.
Viene generata una classe java per l'applet e anche un file html per aprire l'applet.
Il file html ha lo stesso nome del grafico e viene creato su $ pwrp_web, ad es. mygraph.html.
Dichiarando questo file nell'attributo URL dell'oggetto WebLink e impostando WebTarget su RightFrame, il grafico verra' visualizzato nel frame di destra quando viene attivato il pulsante.
Si noti che un grafico che viene aperto come applet riceve i privilegi dall'attributo DefaultWebPriv dell'oggetto Security.
Non e' influenzato da un possibile login.

Configurazione del frame destro

La home page di un progetto e' composta da un menu e un frame destro.
Come impostazione predefinita, il testo della guida di xtt per il progetto viene visualizzato nel riquadro destro. Questo viene convertito in html quando il nodo viene

creato. Se si desidera un'altra pagina iniziale, e' possibile specificare un URL nell'attributo StartURL dell'oggetto OpPlaceWeb.
Se, ad esempio, si desidera visualizzare un grafico Ge, si esporta il grafico come applet java e si inserisce il file html per il grafico nell'attributo StartURL.

LoadArchives

Se devono essere visualizzati i grafici Ge di altri progetti, gli archivi java per questi progetti devono essere disponibili per il browser web. Le classi java generate quando i grafici vengono esportati nell'editor di ge, sono collocati in un archivio con il nome pwrp_'systemname'_web.jar.
L'archivio del presente progetto viene sempre caricato, ma se anche altri archivi devono essere caricati, devono essere dichiarati nell'attributo LoadArchive dell'oggetto OpPlaceWeb. Gli archivi devono anche essere copiati in $ pwrp_web.

Privilegi

Privilegi predefiniti
I privilegi di default sono i privilegi che vengono concessi ad un utente che non ha effettuato il login. Sono indicati nell'attributo DefaultWebPriv dell'oggetto Security.
0 significa che un utente non connesso ha solo accesso alla finestra di accesso. Normalmente si imposta RtRead, che non consente agli utenti che hanno effettuato l'accesso di accedere, ma di non modificare i valori.

Login
Per poter accedere, la funzione di login deve essere abilitata nell'oggetto OpPlaceWeb. Facendo clic sul pulsante Login nella finestra dell'operatore, l'utente puo' accedere con nome utente e password. Il nome utente deve essere registrato nel gruppo di sistema indicato nell'attributo WebSystemGroup dell'oggetto Security. Dopo un accesso riuscito, i privilegi dell'utente determinano l'accesso al sistema.

La password viene decifrata dal client, cioe' nessuna password non crittografata viene inviata in rete.

Processi del server

Tre processi server sono avviati all'avvio del runtime Proview, rt_webmon, rt_webmonmh e rt_webmonelog.

rt_webmon
rt_webmon apre una porta da cui i client web possono recuperare informazioni sugli oggetti nel database e impostare le sottoscrizioni.

rt_webmonmh
rt_webmonmh fornisce un client Web con allarmi ed eventi. L'interfaccia web contiene una finestra con allarmi e liste di eventi che recuperano le informazioni da questo processo.
La selezione di allarmi ed eventi e' determinata da EventSelectList nell'oggetto WebHandler.

rt_webmonelog
rt_webmonelog esegue ricerche nel database del registro eventi.

Funzioni

Questa sezione descrive in modo piu' dettagliato come le varie funzioni nell'ambiente web e

java sono configurate e generate .

Homepage

La home page e' configurata dall'oggetto OpPlaceWeb e generata dal metodo build dell'oggetto WebHandler. Consiste di un numero di file html e jar creati nella directory $ pwrp_web.

La pagina iniziale, index.html, e' composta da due frame. La cornice di sinistra apre il file html per il menu che avvia un'applet che si collega al server Proview e crea il menu.
Il riquadro destro visualizza l'URL indicato nella attributo StartURL dell'oggetto OpPlaceWeb, o se questo attributo e' vuoto, la pagina iniziale per l'aiuto XTT.

Ge grafici

I grafici ge che devono essere visualizzati devono essere esportati come frame java o applet java.
Oggi normalmente vengono utilizzati i frame java.

Un frame java viene visualizzato in una finestra separata. E' scalabile e si adatta alle dimensioni della finestra.
La possibilita' di influenzare i campi di input e i pulsanti nel grafico e' determinata dai privilegi dell'utente connesso.
Per poter esportare un grafico come un frame java, IsJavaFrame deve essere impostato in GraphAttributes nell'editor di ge.

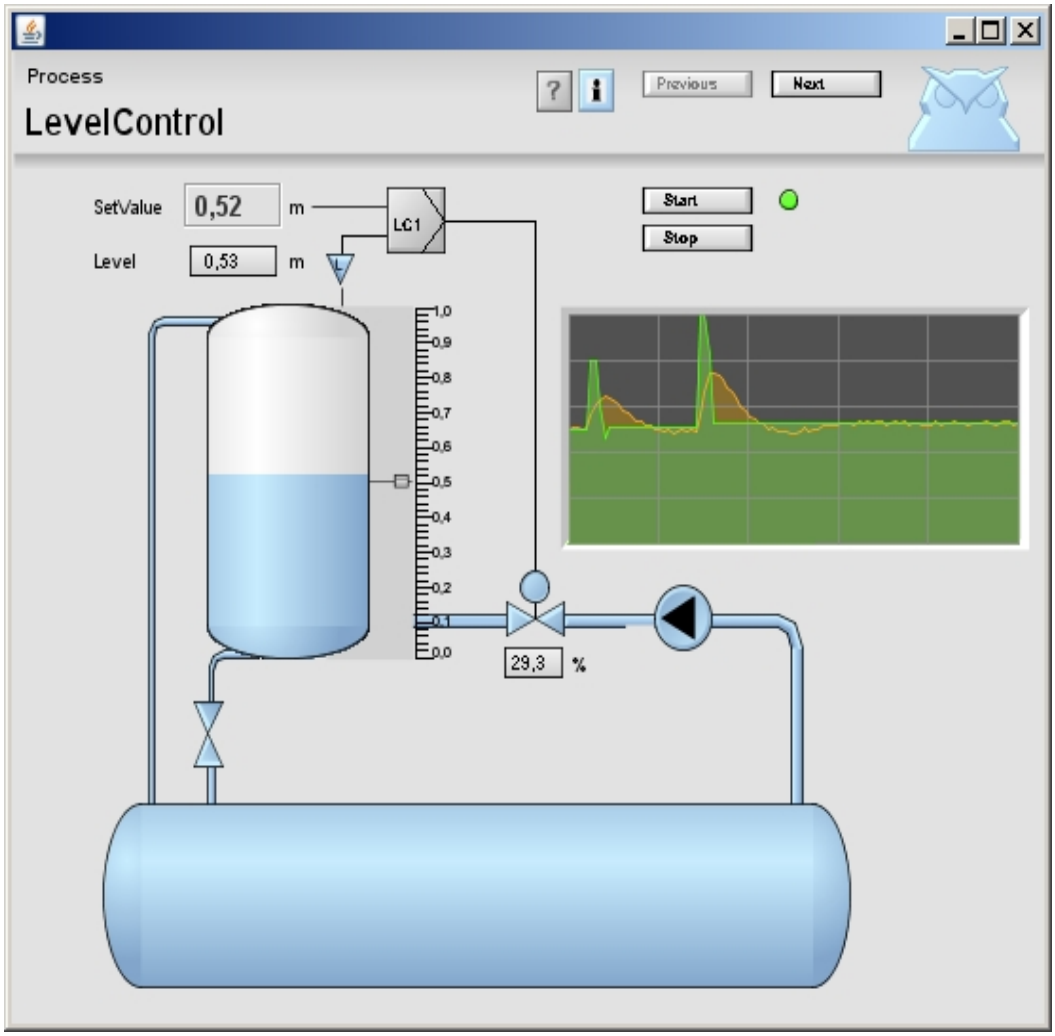


Fig Un grafico aperto come un frame

Un'applet java viene aperta da un file html. Non e' scalabile e i privilegi vengono recuperati da DefaultWebPriv dell'oggetto Security e non sono interessati all'utente connesso.
Le applet Java non possono essere visualizzate nell'ambiente java locale. Per poter esportare un grafico come applet java, IsJavaApplet deve essere impostato in GraphAttributes nell'editor di ge.

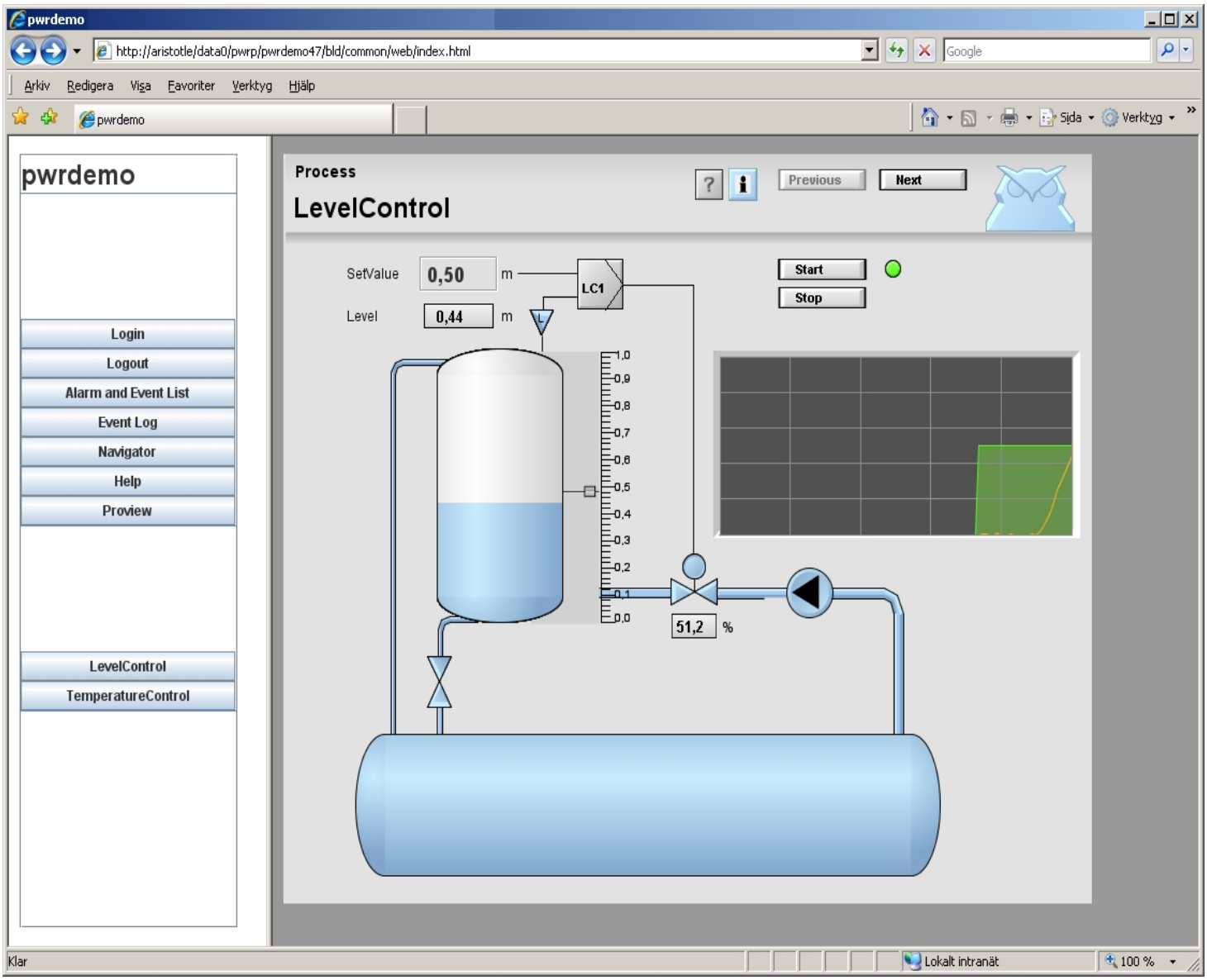


Fig Un grafico aperto come applet.

Dovresti anche indicare la dimensione in pixel del grafico Java generato JavaWidth in GraphAttributes.

Un grafico viene esportato attivando File/Export/Java nel menu dell'editor di Ge.
Anche il metodo di costruzione per un oggetto WebGraph esportera' un grafico, poichÃ© e' stato salvato dopo l'ultima esportazione.
Durante l'esportazione, viene creato un file java su $pwrp_pop che viene compilato e inserito in due archivi, $pwrp_web/pwrp_'systemname'_web.jar e $pwrp_lib/pwrp_'systemname.jar. L'archivio su $ pwrp_web viene utilizzato dall'interfaccia Web e l'archivio su $pwrp_load dall'ambiente java locale.
Per un frame java, il grafico viene esportato come una classe java con lo stesso nome del

grafico, ma con una lettera di blocco sul primo carattere. my_graph e' ad esempio esportato come My_graph.
Si noti che i caratteri locali non sono consentiti nei nomi delle classi java e non possono iniziare con una cifra. In questi casi il grafico puo' essere esportato da Export/Java come con un nome di classe java consentito. Salvando il grafico dopo l'esportazione, il nome java viene memorizzato.
Altrimenti deve essere dichiarato per ogni esportazione.

Per le applet java, il suffisso _A viene aggiunto al nome della classe java, ad es. My_graph_A. Inoltre viene generato un file html, $pwrp_web/my_graph.html, che viene utilizzato per avviare l'applet.
Ovviamente l'applet puo' anche essere avviata da altri file html insieme ad altre applet che terminano gli elementi grafici nella stessa pagina.

L'esportazione java ha alcune limitazioni. Nell'editor di Ge e' possibile posizionare oggetti di testo sopra i sottografi, ma in java il sottografo sara' in primo piano.
Alcuni comandi xtt utilizzati nei pulsanti di comando non sono disponibili nell'ambiente web. C'e' anche un limite al numero di caratteri in una funzione java, che puo' essere raggiunta in grafici di grandi dimensioni.

Elenco allarmi ed eventi

L'elenco degli allarmi e degli eventi richiede che EventSelectList nell'oggetto WebHandler sia compilato con gerarchie da cui visualizzare gli allarmi e gli eventi.

I testi di Aiuto

I testi di aiuto per il progetto nel file xtt_help.dat, vengono esportati in file html su $pwrp_web mediante il metodo di compilazione dell'oggetto OpPlaceWeb. La pagina iniziale per i testi di aiuto e' $pwrp_web/xtt_help_index.html. Questo e' mostrato come predefinito nella cornice destra della home page se non e' specificato nessun altro URL nell'attributo StartURL di OpPlaceWeb.

Se il testo della guida contiene tag immagine, i file png o gif devono essere copiati in $pwrp_web.

Plc

Per poter aprire i programmi plc nell'ambiente web, i file .flw su $pwrp_load devono essere copiati in $pwrp_web.

Langua

La lingua predefinita e' l'inglese. Altre lingue possono essere dichiarate nell'attributo Language dell'oggetto OpPlaceWeb oppure essere selezionate nella finestra di dialogo di selezione della lingua, se configurata.
Se e' necessaria una lingua diversa dall'inglese, i file di lingua per questa lingua e per l'inglese devono essere copiati da $pwrp_exe a $pwrp_web. Ad esempio, per rendere disponibile lo svedese, vengono create le directory $pwrp_web/en_us e $pwrp_web/sv_se, e i file $pwr_exe/en_us/xtt_lng.dat vengono copiati in $pwrp_web/en_us/ e $pwr_exe/sv_se/xtt_lng.dat sono copiati in $pwrp_web/sv_se/.

Archivio

I seguenti archivi dovrebbero essere disponibili su $ pwrp_web: pwr_rt_client.jar, pwr_jop.jar, pwr_jopc.jar, pwr_bcomp.jar, pwr_bcompfc.jar e pwr_abb.jar. Nell'ambiente di sviluppo questi devono essere copiati da $ pwr_lib, in runtime cio' viene eseguito durante l'installazione del pacchetto pwrrt.

Webserver

Per aprire una home page del progetto in un browser Web, e' necessario avviare un server Web sul nodo, ad esempio apache.

Solo alcune directory scelte possono essere aperte dal server web, ad es. /Var/www. O puoi spostare il contenuto di $pwrp_web in una directory aperta, oppure puoi aprire $pwrp_web.
In apache questo e' specificato nel file /etc/apache2/apache2.conf aggiungendo

```
Alias /pwrp_web/ /usr/pwrp/myproj/bld/common/web/

<Directory /usr/pwrp/myproj/bld/common/web>
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

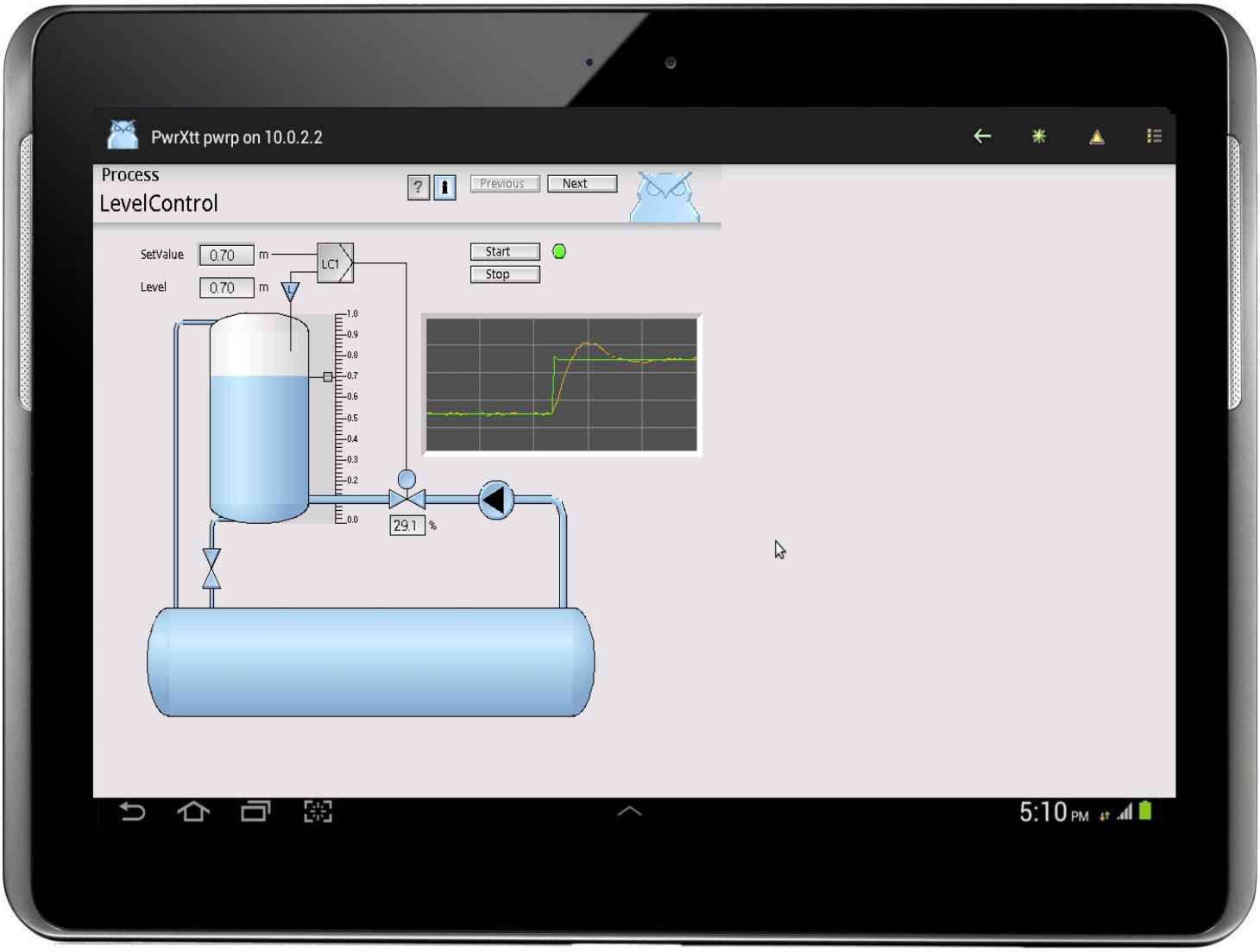
Un progetto, myproj, viene aperto con la directory root/usr/pwrp/myproj, con l'alias pwrp_web.
Se il nodo e' mynode, l'URL per aprire l'ambiente web e'

```
http://mynode/pwrp_web/index.html
```

L'aggiunta corrispondente ad apache2.conf in un processo o stazione operatore, che viene aggiunta all'installazione del pacchetto pwrrt, sarebbe

```
Alias /pwrp_web/ /pwrp/common/web/

<Directory /pwrp/common/web>
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

18 Applicazione Android



Applicazione Android

L'applicazione Android puo' essere installata su telefoni cellulari e tablet con sistema operativo Android. L'applicazione si connette a un processo o stazione operatore e puo' mostrare

- il runtime navigator.
- pagine grafiche di processo.
- plc trace.
- liste alarmi e eventi.
- liste incrociate.

Per installare l'app, scarica PwrXtt.apk dalla pagina di download di Proview su Source Forge e installa sul tuo dispositivo.

Configurazione

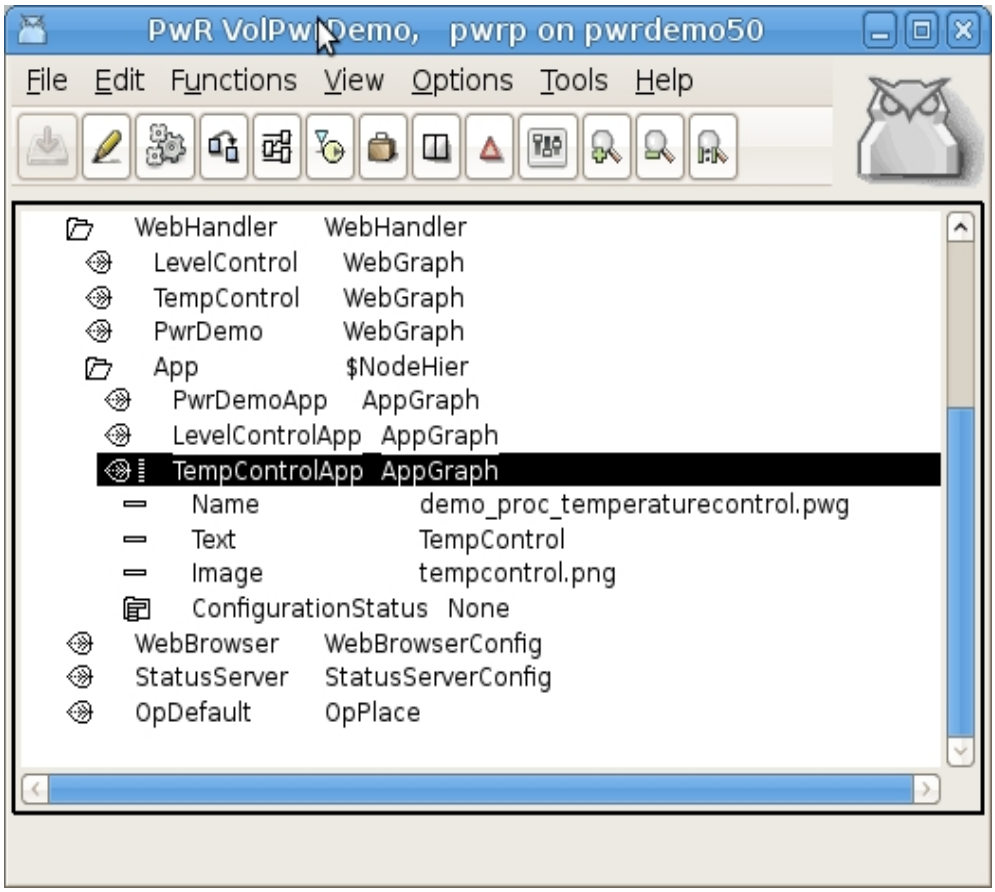
L'app utilizza gli stessi processi server dell'interfaccia Web, rt_webmon e rt_webmonmh, configurati dall'oggetto WebHandler. Vedere il capitolo Ambiente operatore Java per maggiori informazioni.

Start menu

All'avvio dell'applicazione viene visualizzato un menu di avvio o, se questo non e' configurato, viene visualizzato il navigatore di runtime.

Il menu di avvio e' configurato con un oggetto OpPlaceApp e al di sotto di questo vengono posizionati gli oggetti AppGraph e AppLink. L'AppGraph viene utilizzato per aprire i grafici Ge e l'AppLink e' un collegamento a una pagina Web o un documento.

Nell'oggetto AppGraph, viene specificato il nome del file pwg, un file immagine per l'icona e un testo che viene scritto sotto l'icona. Nell'oggetto AppLink, viene specificato un URL, insieme al file immagine e al testo.



Configurazione dello start menu

Fig Start menu

Distribuzione

L'attributo AppUseWebDir nell'oggetto OpPlaceApp specifica se tutti i file vengono recuperati dalla directory $pwrp_web, o se i file com vengono recuperati da $pwrp_exe e $pwrp_load. Se AppUseWebDir e' impostato, l'app funziona in modo analogo all'interfaccia Web e i file pwg e flw devono essere copiati in $pwrp_web.
Tabella sui file che vengono utilizzati dall'applicazione. Se AppUseWebDir e' impostato, questi file devono essere distribuiti su $pwrp_web.

Funzione	Filetype	Directory
Ge graphs	pwg	$pwrp_exe
Plc trace	flw	$pwrp_load
Crossreferenser	rtt_crr*.dat	$pwrp_load
Helpfiles	html	$pwrp_web
Icons	png, gif etc	$pwrp_exe

Web server

L'app legge i file attraverso un server web, ad esempio apache, che deve essere avviato. Per quanto riguarda l'interfaccia web, esiste un alias pwrp_web definito, che punta a $pwrp_web. Per apache questo e' specificato nel file /etc/apache2/apache2.conf.

```
Alias /pwrp_web/ /pwrp/common/web/

<Directory /pwrp/common/web>
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

Se AppUserWebDir e' zero, anche gli alias per pwrp_exe e pwrp_load devono essere definiti. In questo caso i file non devono essere copiati in $pwrp_web, ma vengono letti da $pwrp_load o $pwrp_exe dove risiedono normalmente. i file flw e crossreferences vengono letti da $pwrp_load e altri file, pwg, png, ecc, da $pwrp_exe.

Nota che la directory pwrp_exe dipende dalla piattaforma.

```
Alias /pwrp_load/ /pwrp/common/load/

<Directory /pwrp/common/load>
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

Alias /pwrp_exe/ /pwrp/x86_linux/exe/

<Directory /pwrp/x86_linux/exe>
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

Per visualizzare i testi di aiuto di Proview, deve esserci anche un alias per pwr_doc che punta a $pwr_doc. Normalmente questo viene aggiunto all'installazione del pacchetto pwrrt.

```
Alias /pwr_doc/ /usr/pwrrt/doc/

<Directory /usr/pwrrt/doc>
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

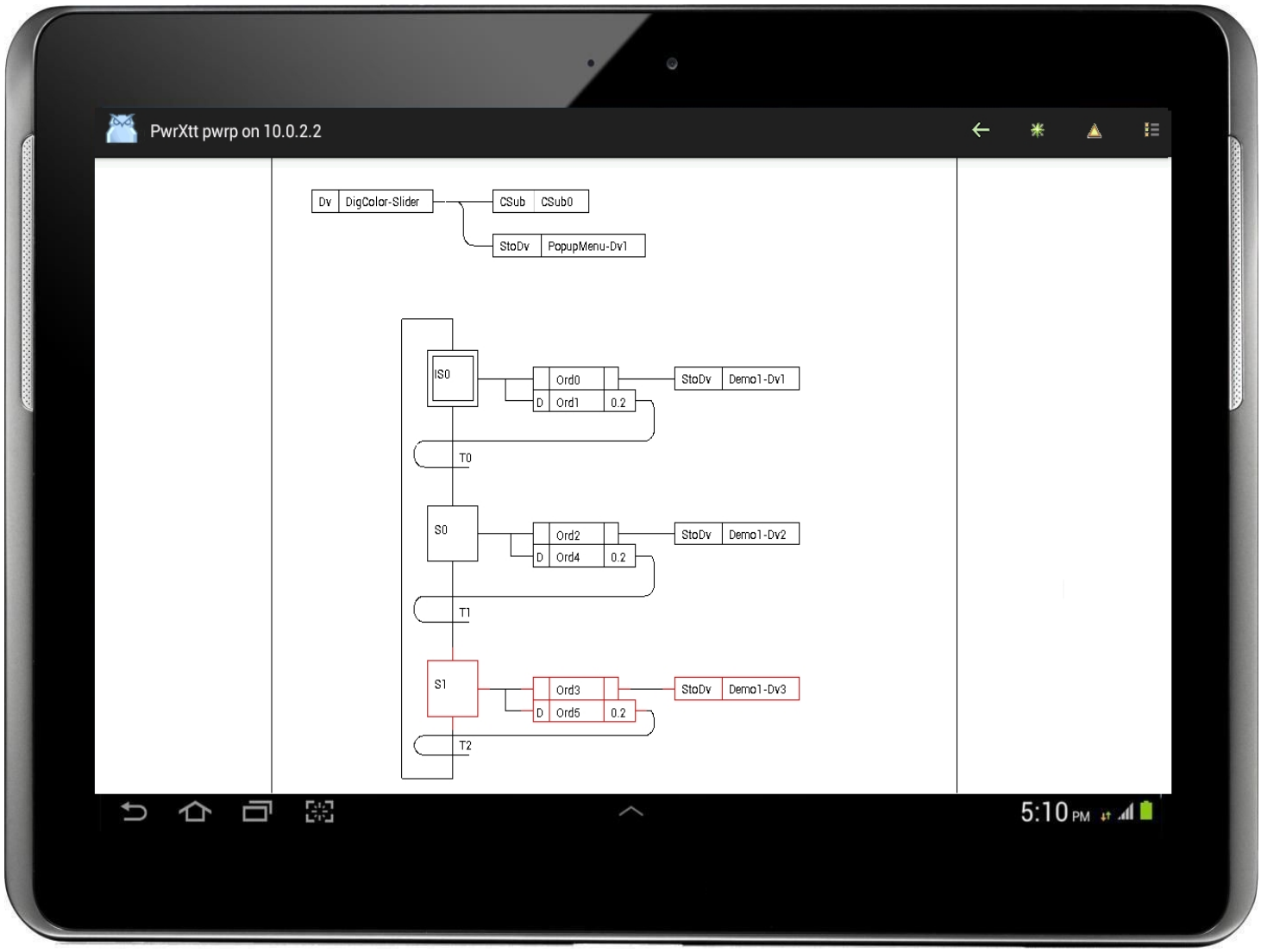


Fig La traccia del Plc legge flw-filer da $pwrp_load

19 Esecuzione e test di un sistema Proview

Nei capitoli precedenti abbiamo descritto come configurare un sistema PROVIEW/R, come creare programmi PLC e come creare grafica di impianto.
Ora e' il momento di eseguire e testare il sistema.

Questo capitolo mostra come:

- Costruire il sistema.
- avviare l'ambiente di simulazione
  ed il monitor di runtime
- distribuire
- Avviare l'ambiente di runtime

Controllo della sintassi

Alcune classi hanno un metodo di controllo della sintassi, controllando cosi' che l'oggetto sia configurato correttamente.
I metodi degli oggetti dei segnali controllano ad esempio che siano collegati a un canale e il metodo per PlcPgm controlla che sia collegato a un oggetto thread plc, ecc.
I metodi di sintassi catturano una grande quantita' di errori, ma non garantiscono che tutto funzionera' alla perfezione.

I metodi di sintassi vengono eseguiti da 'Functions/Syntax Check' nel menu Configuratore.

19.1 Compilazione

Prima di avere un sistema avviabile per un nodo, bisogna compilare il nodo.
Cio' significa che bisogna generare tutto quello che serve in un ambiente di runtime, ad esempio, un file di avvio che specifica quali volumi devono essere caricati, loadfile per i volumi contenenti informazioni sugli oggetti nel volume, un eseguibile per il programma plc ecc.

La compilazione avviene attivando 'Functions/Build Node' nel menu Configuratore o con il pulsante corrispondente nel pannello degli strumenti. Se nel progetto sono stati configurati diversi nodi, e' necessario selezionare un nodo dall'elenco dei nodi visualizzato.

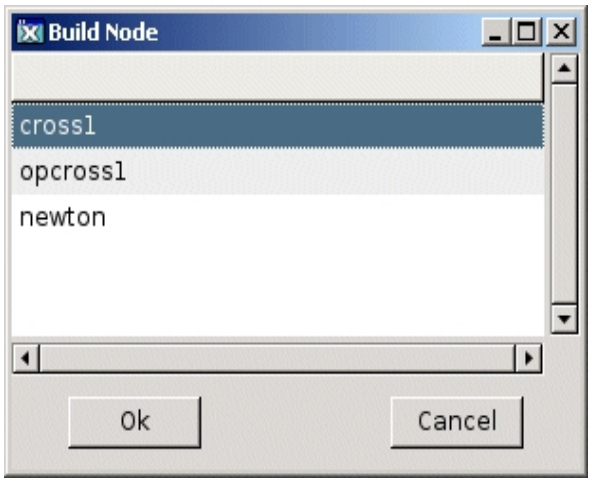


Fig Selezione di un nodo

Anche la compilazione e' divisa in metodi per classi diverse. Il metodo di compilazione per un PlcPgm genera codice per le finestre modificate e compila questo codice.
Il metodo di compialzione per un volume crea un file di caricamento per il volume e chiama i metodi di costruzione per tutti gli oggetti nel volume.
Il metodo di compilazione per un nodo chiama il metodo build per il volume, crea un file di boot per il nodo e collega l'eseguibile di plc.
I metodi di compilazione per un oggetto possono essere chiamati selezionando l'oggetto e attivando 'Functions/Build Object' nel menu e il metodo di compialzione per il volume corrente attivando 'Functions/Build Volume'.

Di seguito una descrizione di alcuni metodi di compilazione.

XttGraph	Copia il file .pwg per il grafico da $pwrp_pop a $pwrp_exe.
WebGraph	Esporta il grafico in java, compila e inserisce in un archivio.
OpPlaceWeb	Genera file html per la home page del nodo e converte i file help xtt al formato web.
PlcPgm	Genera codice-C che viene compilato dal compilatore C.
RootVolume	Chiama il metodo di compilazione per tutti gli oggetti nel volume e crea un file di caricamento con informazioni sugli oggetti nel volume. Crea anche file crossreference se specificato in Opzioni.
ClassVolume	Genera gli include i file con le strutture C per le classi del volume, e crea un file di caricamento con le definizioni di tipo e classe.
Node	Chiama il metodo di compilazione per il volume principale, se il volume e' disponibile. Crea un file di boot con informazioni su quali volumi devono essere caricati e collega i file programma plc del nodo.

Normalmente il metodo di compilazione verifica innanzitutto se qualcosa e' stato modificato e esegue la build solo se trova una modifica.
In alcuni casi, si desidera forzare una build e quindi impostare "Force" nella colonna Build nelle opzioni aperte da 'Options/Settings' nel menu. E' anche possibile contrassegnare che si desidera creare i file di crossreference su build, o che la compilazione e il collegamento del plc devono essere eseguiti con debug.

Se un nodo contiene sottovolumi o volumi condivisi, questi devono essere creati da 'Build Volume' prima di creare il nodo.

19.2 Simulare

Simulare significa avviare il sistema di una stazione di processo o stazione operatore sulla stazione di sviluppo.
Questo e' un modo veloce per testare programmi e processare la grafica durante la fase di sviluppo.
Inoltre, quando un sistema e' in produzione, e' possibile testare le modifiche del sistema prima di scaricarle nel sistema di produzione.

Durante la simulazione, i dati di input del processo devono essere simulati.
Questo viene fatto creando un PlcPgm che legge i dati di output nel processo, ad esempio un valore Ao e da questi valori impostati agli oggetti Ai e Di. A questo scopo ci sono oggetti StoDi e StoAi speciali, usati solo in simulazione.
Devi assicurarti che il programma di simulazione venga eseguito solo in simulazione, ad es. impostando l'attributo ScanOff dell'oggetto PlcWindow se IOSimulflag nell'oggetto IOHandler e' 1.

Per essere in grado di simulare sulla stazione di sviluppo, deve essere configurato nel volume della directory.
La simulazione viene eseguita su un bus QCom separato, configurato da un oggetto BusConfig sul livello superiore nella finestra destra del volume della directory.
In questo si specifica l'identita' del bus, ad es. 999. Sotto l'oggetto BusConfig si posiziona un oggetto NodeConfig per la stazione di sviluppo e si compila il nome del nodo e l'indirizzo IP.
e' possibile utilizzare l'indirizzo di loopback 127.0.0.1 finchÃ© non si prevede di comunicare con altri nodi.
Devi anche specificare quale volume vuoi simulare impostando il nome dell'oggetto RootVolumeLoad sotto l'oggetto NodeConfig, al nome del volume.

Si noti che la guida alla configurazione per il volume della directory normalmente crea un bus simulato e un nodo simulato.

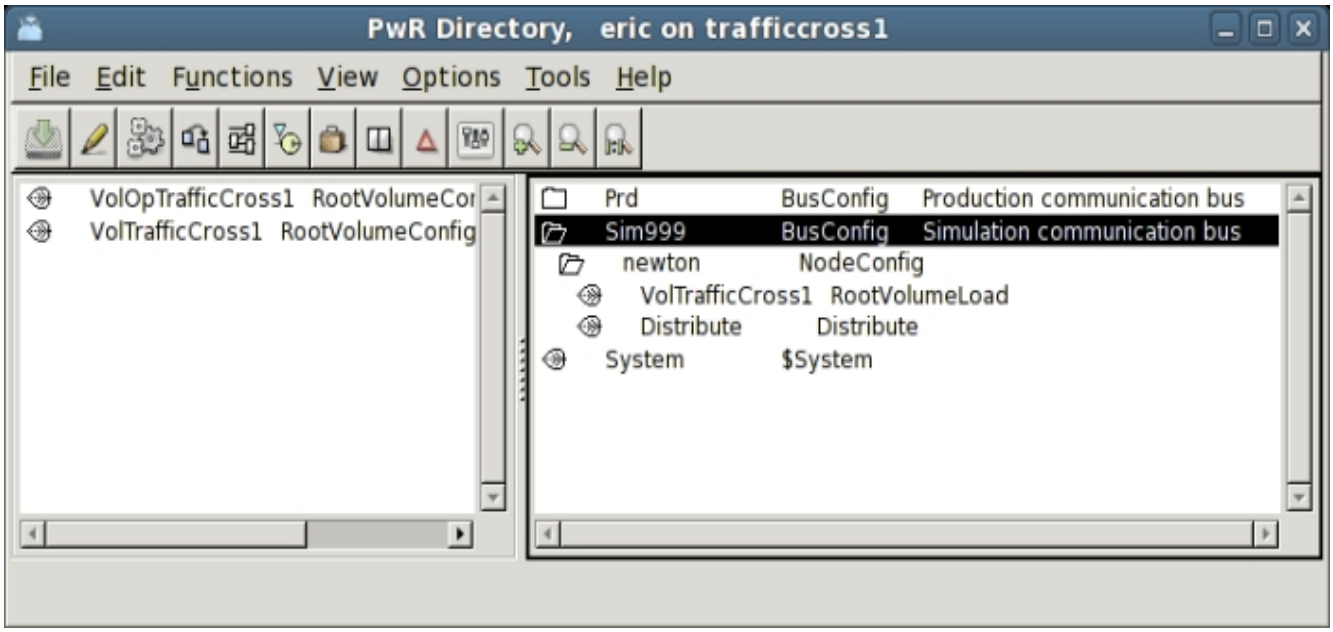


Fig Volume di directory con configurazione della stazione di sviluppo newton

Quando la stazione di sviluppo e' configurata, si costruisce aprendo il configuratore per il volume che deve essere simulato e si attiva 'Build Node' e si seleziona la stazione di sviluppo nell'elenco dei nodi visualizzato.

Prima di avviare il runtime, e' necessario definire la variabile di ambiente PWR_BUS_ID sull'identita' del bus di simulazione. Un valore predefinito per PWR_BUS_ID e' impostato nel file /etc/proview.cnf, parametro QcomBusId. Con il comando

```
> echo $PWR_BUS_ID
```

controlli l'identita' del bus e con il comando

```
> export PWR_BUS_ID=999
```

tu assegni un altro valore. E' possibile inserire questo comando, ad esempio, in $pwrp_login/login.sh.

Ora e' possibile avviare il runtime Proview con

```
> rt_ini &
```

e fermarlo con

```
> . pwr_stop.sh
```

Se Proview non si avvia, e' possibile aggiungere -i al comando start per visualizzare i messaggi di errore.

```
> rt_ini -i
```

Nota che devi sempre resettare eseguendo '. pwr_stop.sh 'prima di un nuovo tentativo di avvio.

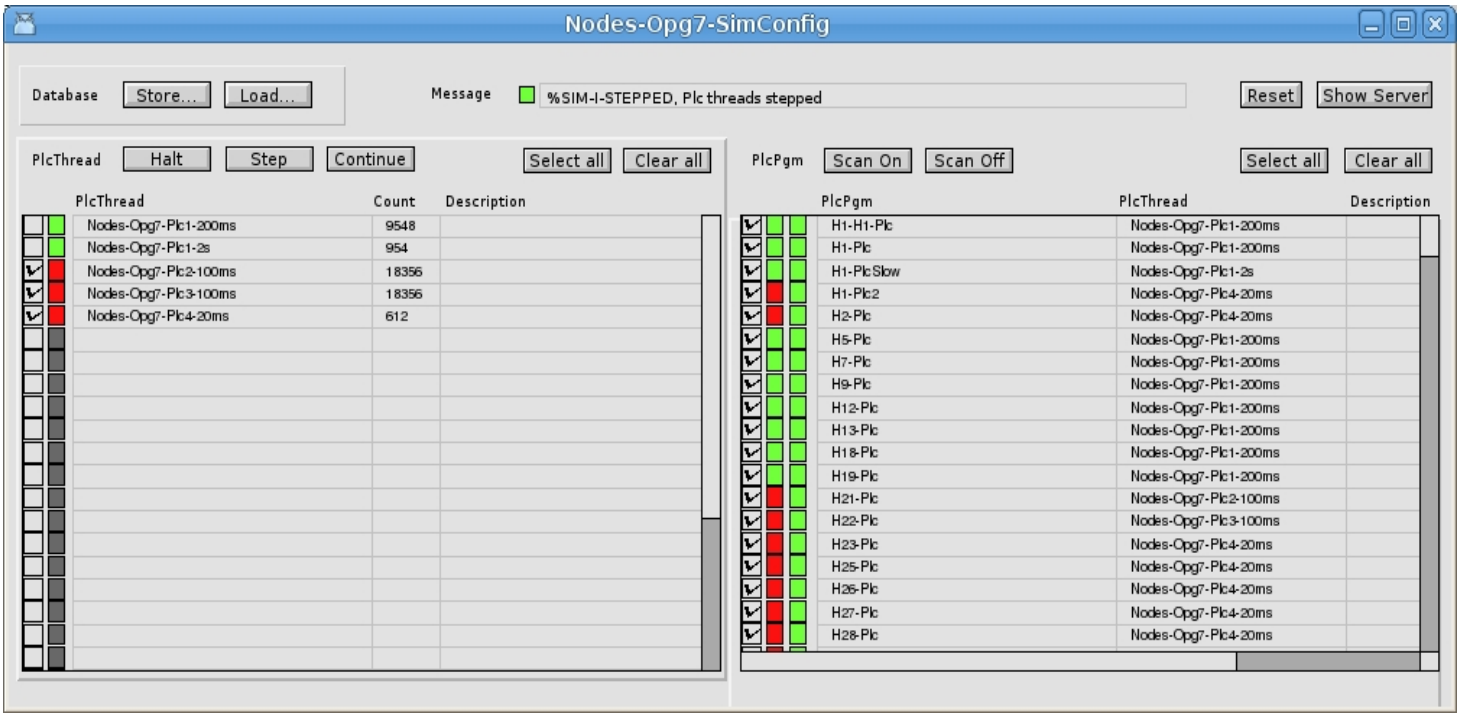
Quando l'ambiente di runtime e' in esecuzione, e' possibile esplorare il sistema avviando il navigatore di runtime

```
> rt_xtt
```

E' inoltre possibile utilizzare il monitor di runtime per avviare l'ambiente di runtime.
Vedi sotto.

19.2.1 Simula server

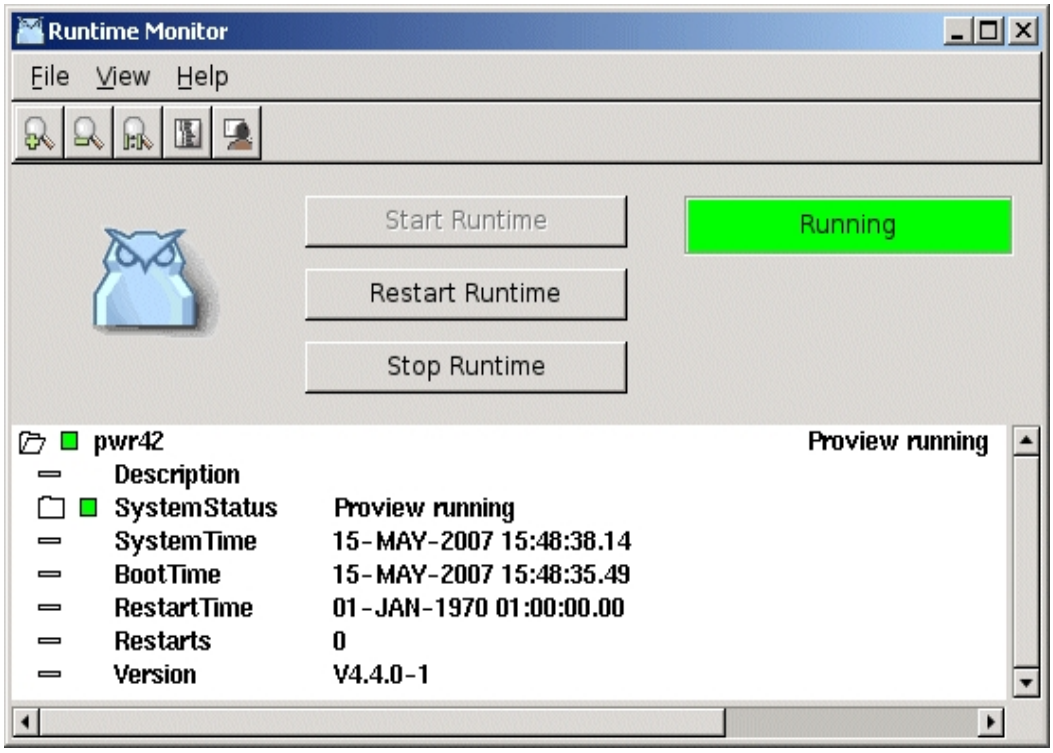
E' possibile avviare un server simulato per controllare l'esecuzione del programma plc e archiviare il file



19.3 Runtime Monitor

Spesso si desidera avviare l'ambiente di runtime sul nodo di sviluppo, ad esempio se e' stata apportata una modifica nel sistema che si desidera testare, prima di inviarlo al sistema di produzione.

Runtime Monitor viene utilizzato per avviare e arrestare l'ambiente di runtime sulla stazione di sviluppo.



Per avviare l'ambiente di runtime sulla stazione di sviluppo, e' necessario soddisfare i seguenti requisiti

- il nodo deve essere configurato con un oggetto NodeConfig nel volume del progetto.

- il bus di comunicazione corretto deve essere impostato. Per fare cio' si imposta la variabile di ambiente bash PWR_BUS_ID sul bus dichiarato nell'oggetto BusConfig nel volume del progetto, ad esempio

```
export PWR_BUS_ID=999
```

Il monitor di runtime richiede anche un oggetto StatusServerConfig da configurare sotto l'oggetto $Node nel volume da avviare.

Il Runtime Monitor viene avviato da Tools / Runtime Monitor nel menu.
Ci sono pulsanti per avviare e arrestare l'ambiente di runtime
('Start Runtime' e 'Stop Runtime').
Nel riquadro colorato viene visualizzato lo stato dell'ambiente runtime
('In esecuzione' o 'Giu").
Il colore indica lo stato del sistema, rosso per stato errore, giallo per avviso e verde per OK.

Il pulsante 'Restart Runtime' esegue un riavvio graduale e puo' essere utilizzato se il runtime e' gia' stato avviato.

19.4 Stazioni di processo e operatore

In questa sezione vengono illustrati i passaggi per avviare Proview su una stazione di processo o operatore.

- configurazione del nodo nel volume della directory
- compilare il nodo
- installazione del pacchetto runtime
- distribuire al nodo
- avviare l'ambiente di runtime sul nodo

Configurazione del volume della directory

Il nodo deve essere configurato nel volume della directory da un oggetto NodeConfig. Questo e' posto sotto un oggetto BusConfig che contiene l'identita' del bus per il bus di produzione.

Nell'oggetto NodeConfig sono configurati il nome del nodo e l'indirizzo IP, e sotto l'oggetto NodeConfig c'e' un oggetto RootVolumeLoad che indica quale volume di root deve essere avviato sul nodo.

Il nodo viene creato dal configuratore di rootvolume attivando 'Functions/Build Node' e selezionando la stazione di processo nell'elenco dei nodi (se e' presente un solo nodo configurato, la lista dei nodi non viene visualizzata).

Installazione del pacchetto runtime

Il pacchetto runtime viene installato sul nodo processo/operatore installando il pacchetto pwrrt della distribuzione Linux installata sul nodo. Leggi la guida all'installazione nella pagina di Download su www.proview.se per maggiori informazioni.

19.4.1 Distribuire

Distribuire, significa raccogliere in un pacchetto i file che vengono creati al momento della compilazione del nodo e che sono necessari per eseguire l'ambiente runtime. Il pacchetto viene copiato nella stazione di processo o nella stazione operatore e li' decompresso.

I file che faranno parte del pacchetto, sono configurati nel volume di directory con un oggetto Distribute sotto l'oggetto NodeConfig del nodo. L'oggetto Distribute contiene l'attributo Componenti in cui vengono specificati i tipi di componenti o file selezionati. Se ci sono componenti specificati che non vengono generati, un messaggio di avviso sara' generato alla distribuzione.

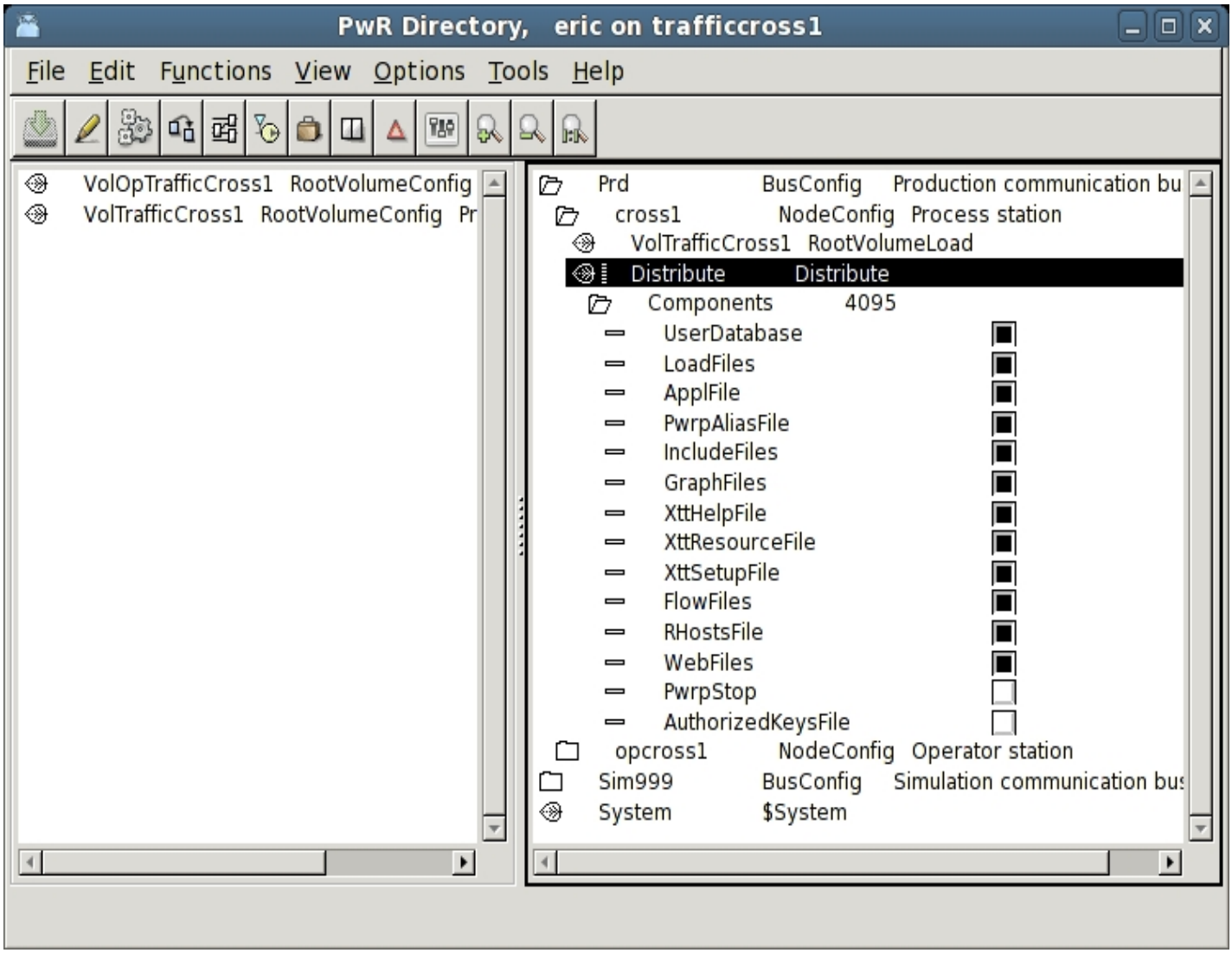


Fig L'oggetto Distribuisci

Se ci sono altri file, ad es. programmi applicativi, che devono essere una parte del pacchetto, si aggiunge un oggetto di tipo ApplDistribute sotto l'oggetto Distribuisci. Nell'oggetto ApplDistribute e' possibile indicare quali file devono essere aggiunti (e' consentita la specifica con caratteri jolly) e dove devono essere copiati.

Tutti i file necessari in fase di runtime dovrebbero far parte del pacchetto.
E' importante che un pacchetto rappresenti una versione completa del sistema, rendendo possibile il ripristino dell'ambiente di runtime se, ad esempio, si desidera tornare a una versione precedente.
Se si verifica un arresto anomalo del disco, e' anche importante poter ripristinare il sistema su un nuovo disco senza copiare e modificare manualmente i file.

La distribuzione viene eseguita dal distributore che viene aperto da 'Functions/Distribute' nel menu Configuratore. Seleziona il nodo che vuoi distribuire e seleziona 'Functions/Distribute' nel menu Distributore.

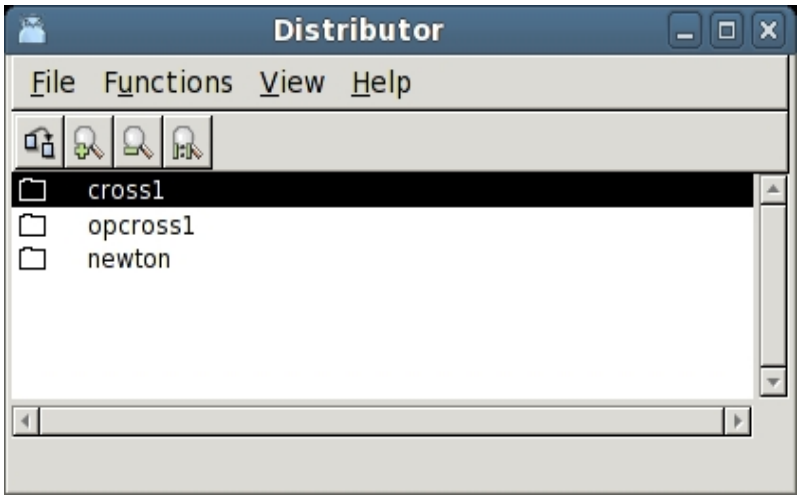


Fig Distribut\F6ren

Il distributore ora raccogliera' i file specificati in un pacchetto e copiera' il pacchetto sull'utente 'pwrp' sul nodo di processo con ssh. ssh richiede una password e questa deve essere digitata due volte nella finestra del terminale da cui viene avviato il configuratore. All'installazione l'utente 'pwrp' ha la password 'pwrp', ma potrebbe essere stata modificata per motivi di sicurezza.

Se non si dispone di un connessione di rete con la stazione di processo, e' possibile creare un pacchetto e spostarlo nella stazione di processo, ad esempio con una chiavetta USB. Seleziona il nodo nel distributore e attiva 'Functions/Create Package'.
Il pacchetto e' memorizzato su $pwrp_load con il nome pwrp_pkg_'nodename '_' version'.tgz, ad es. pwrp_pkg_cross1_0002.tgz. Sulla stazione di processo si compatta il pacchetto con lo script pwr_pkg.sh -i che prende come argomento il nome del pacchetto.

```
> pwr_pkg.sh -i pwrp_pkg_cross1_0002.tgz
```

Ripristina una versione precedente

A volte una modifica del sistema non funziona come previsto e si desidera ripristinare una versione precedente.
Questo viene fatto facilmente con pwr_pkg.sh. I pacchetti sono memorizzati nella directory home dell'utente pwrp, /home/pwrp.
Trovando il pacchetto per la versione precedente e avviando pwr_pkg.sh -i con questo pacchetto, si ripristina la versione.

```
> pwr_pkg.sh -i pwrp_pkg_cross1_0001.tgz
```

19.4.2 Identita' del bus

L'identita' del bus QCom per il nodo e' impostata nel file /etc/proview.cnf.

```
#
# Configuration file for Proview
#
# Default QCOM Bus Id
#
qcomBusId              999
#
# Web directory
```

```
#
webDirectory            /var/www
```

Questo valore deve corrispondere all'ID bus configurato per il nodo nel volume della directory.
L'id del bus corrente e' memorizzato nella variabile di ambiente $PWR_BUS_ID.

19.4.3 Avvio dell'ambiente di runtime

L'ambiente di runtime su una stazione di processo o stazione operatore viene avviato con il comando

```
> pwr start
```

e arrestrato con

```
> pwr stop
```

o

```
> pwr kill
```

Il comando 'pwr stop' richiede che tutti i processi siano vivi, mentre 'pwr kill' cancella
tutto in tutte le situazioni.

Se proview runtime non si avvia correttamente, e' possibile avviarlo con il comando

```
> rt_ini -i
```

per visualizzare le registrazioni della console nella finestra del terminale.
Ripristinate sempre con 'pwr kill' prima di un nuovo tentativo di avvio.

20 Il configuratore

Il configuratore viene utilizzato per navigare e configurare il Workbench (ambiente di lavoro).
Il configuratore visualizza gli oggetti in un volume. Gli oggetti sono solitamente separati in due finestre, una sinistra e una destra, e il modo in cui viene eseguita la separazione dipende dal tipo di volumi gestiti.

- Per i volumi root e i sottovolumi, la gerarchia dell'impianto viene visualizzata nella finestra a sinistra e la gerarchia dei nodi a destra.

- Per il volume di directory, i volumi vengono visualizzate nella finestra di sinistra e i bus e i nodi nella finestra di destra.

- Per i volumi delle classi, le classi vengono visualizzate nella finestra di sinistra e i tipi nella destra.

Da 'View/TwoWindow' e' possibile scegliere se visualizzare due finestre o una sola.
Se viene visualizzata una sola finestra, ogni volta che si attiva 'TwoWindow' verra' visualizzata la finestra superiore, altrimenti sara' la finestra inferiore.

Da 'Edit/Edit mode' si entra in modalita' modifica, e una palette con varie classi viene visualizzata a sinistra.
Ora puoi creare nuovi oggetti, spostare oggetti, cambiare i valori degli attributi, ecc.

Rappresentazione del volume

I volumi sono memorizzati in vari formati, in un database, in un file di caricamento o in un file di testo.
Il configuratore puo' visualizzare un volume in tutti questi formati e ha quattro diverse rappresentazioni di volumi:

- db, un database. I volumi e i sottovolumi sono creati e modificati in un database. Prima di poter avviare l'ambiente di runtime, i file di caricamento vengono generati dai volumi. I file di caricamento vengono letti all'avvio del runtime.
  La rappresentazione db e' modificabile.

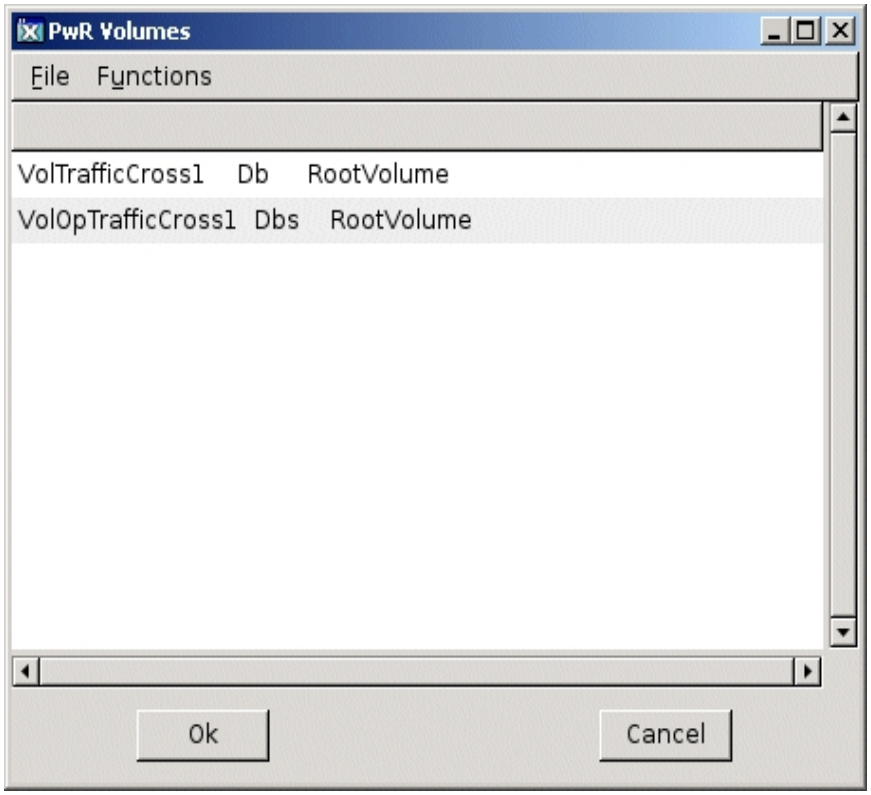
- wbl, un file di testo con estensione .wb_load. I volumi di classe sono archiviati come wbl e i volumi root e sub possono essere scaricati in un file wbl, ad esempio durante l'aggiornamento e successivamente ricaricati. La rappresentazione wbl e' modificabile dal nodo. Quando si modifica un volume di una classe, si importa la rappresentazione wbl in un mem-represention, quindi lo si salva di nuovo come wbl.

- dbs, un file di caricamento. Da rootvolumes, sottovolume e classivolumi, nella rappresentazione db e wbl, i file di caricamento sono certificati e utilizzati nell'ambiente di runtime. Il configuratore legge anche i file dbs del classvolumes per essere in grado di interpretare le classi e i file dbs della radice e dei sottovolumi per essere in grado di tradurre i riferimenti a oggetti esterni.
  La rappresentazione dbs non e' modificabile.

- mem, un volume che il configuratore conserva internamente in memoria. I buffer di copia/incolla sono composti da mem-volume. Il classeditor importa il volume della classe, che in origine e' un wbl, in un volume mem, poichÃ© la rappresentazione mem e' modificabile.

Come vediamo sopra, lo stesso volume puo' esistere sia come database sia come file di caricamento. Quando si avvia il configuratore, si specifica un volume come argomento. Per questo volume, il database e' aperto, cioe' e' rappresentato come db, per gli altri volumi nel progetto, i file di caricamento sono aperti, cioe' sono rappresentati come dbs.
Cio' rende possibile visualizzare gli altri volumi nel progetto e risolvere i riferimenti ad essi, ma non sono modificabili.
Se il database del volume e' bloccato, poichÃ© qualcun altro l'ha aperto, viene visualizzato un messaggio di errore e il file di caricamento viene aperto al posto del database.

Nella figura seguente viene visualizzato l'elenco dei volumi, che viene aperto da 'File/Open' nel menu.
Mostra tutti i volumi aperti dal configuratore. Possiamo vedere che il database per il volume di root VolTrafficCross1 e' aperto, mentre l'altro volume di root, VolOpTrafficCross1 e' aperto come un file di caricamento. Anche i volumi delle classi vengono aperti come file di caricamento.



Se nessun volume viene fornito come argomento all'avvio del configuratore, viene aperto il database del volume della directory e gli altri volumi vengono aperti come volumi-dbs.

Navigare

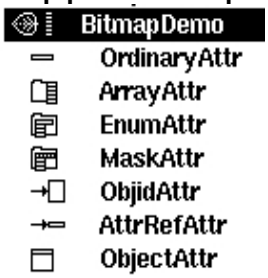
Gli oggetti del volume corrente vengono visualizzati nel configuratore. Gli oggetti sono ordinati in una struttura ad albero e gli oggetti con figli vengono visualizzati con una mappa mentre gli oggetti senza figli con una foglia.
Per ogni oggetto il nome dell'oggetto, la classe e la descrizione possibile vengono visualizzati come predefiniti (la descrizione viene recuperata dall'attributo Descrizione nell'oggetto).

Facendo clic con MB1 su una mappa, la mappa viene aperta e vengono visualizzati i figli

dell'oggetto.
Se la mappa e' gia' aperta, sara' chiusa. Puoi anche aprire una mappa con un doppio clic in qualsiasi punto della riga dell'oggetto.

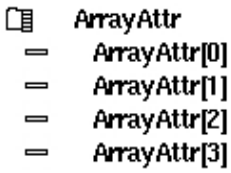
Se volete vedere il contenuto di un oggetto, fare clic con Shift/Click MB1 sulla mappa o foglia, o Shift/Doubleclick MB1 punto qualsiasi nella riga dell'oggetto. Ora vengono visualizzati gli attributi dell'oggetto, insieme al valore di ciascun attributo.
Gli attributi sono contrassegnati da varie icone dipendenti dal tipo.

Bitmap per diversi tipi di attributi



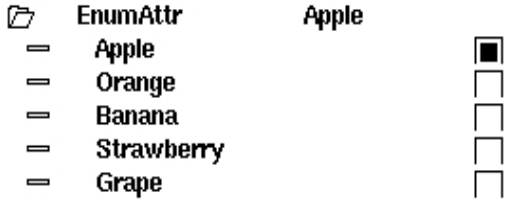
- Un attributo ordinario e' contrassegnato da un rettangolo lungo e stretto.

- Una matrice e' contrassegnata da una mappa e una pila di attributi. L'array viene aperto con un clic MB1 sulla mappa o doppio clic ovunque sulla riga dell'attributo. Ora vengono visualizzati gli elementi dell'array.

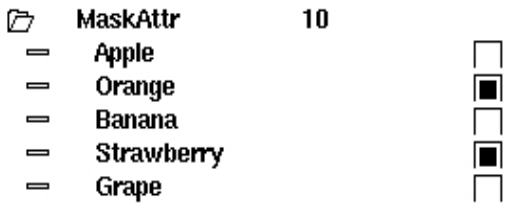


- Un attributo che fa riferimento ad un altro attributo o oggetto, cioe' di tipo Objid o AttrRef, e' contrassegnato da una freccia che punta a un quadrato.

- I tipi enumerazione, Enum, sono contrassegnati con la mappa am e alcuni rettangoli lunghi e stretti. Facendo clic MB1 sulla mappa vengono visualizzate le diverse alternative dell'enumerazione. Le alternative sono visualizzate con caselle di controllo e l'alternativa scelta e' contrassegnata. Si puo' anche fare doppio clic MB1 nella riga degli attributi per visualizzare le alternative.



- I tipi maschera, Mask, sono contrassegnati in modo simile ad Enum, e vengono visualizzati i diversi bit facendo clic MB1 sulla mappa o doppio clic MB1 nella riga degli attributi.



- Gli oggetti attributi, ovvero gli attributi che contengono la struttura dati di un oggetto, sono contrassegnati da un quadrato con una doppia linea nella parte superiore. L'oggetto attributo viene aperto facendo clic MB1 sul quadrato o DoubleClick MB1 nella riga dell'attributo.

Un oggetto o attributo e' selezionato con clic MB1 nella riga oggetto/attributo (non nella mappa o foglia). Con Shift/MB1 puoi selezionare diversi oggetti. Con Drag MB1 puoi anche selezionare diversi oggetti.

Da un punto di vista ergonomico, e' spesso meglio navigare utilizzando la tastiera. Si utilizzano principalmente i tasti freccia. Per prima cosa bisogna impostare l'input focus sulla finestra, facendo clic su di esso. Il fuoco di input tra la finestra sinistra e quella destra viene spostato con TAB.

Con Freccia Su/Freccia Giu' puoi selezionare un oggetto. Se l'oggetto ha figli, apri i figli con Freccia Destra e chiudi con Freccia Sinistra. Il contenuto dell'oggetto, cioe' gli attributi, viene visualizzato con Shift/Freccia Destra e chiuso con Freccia Sinistra.

Un attributo che e' un array, enum, maschera o oggetto attributo, viene aperto da Freccia Destra e chiuso da Freccia Sinistra.

Quando ti senti a casa nell'albero degli oggetti, puoi impostarti come "advanced user". Funzione aggiuntive vengono date ai tasti freccia. Freccia Destra su un oggetto, visualizza ad esempio gli attributi dell'oggetto, se non ha figli. Se ha figli, bisogna usare Shift/Freccia Destra come prima.

Editing

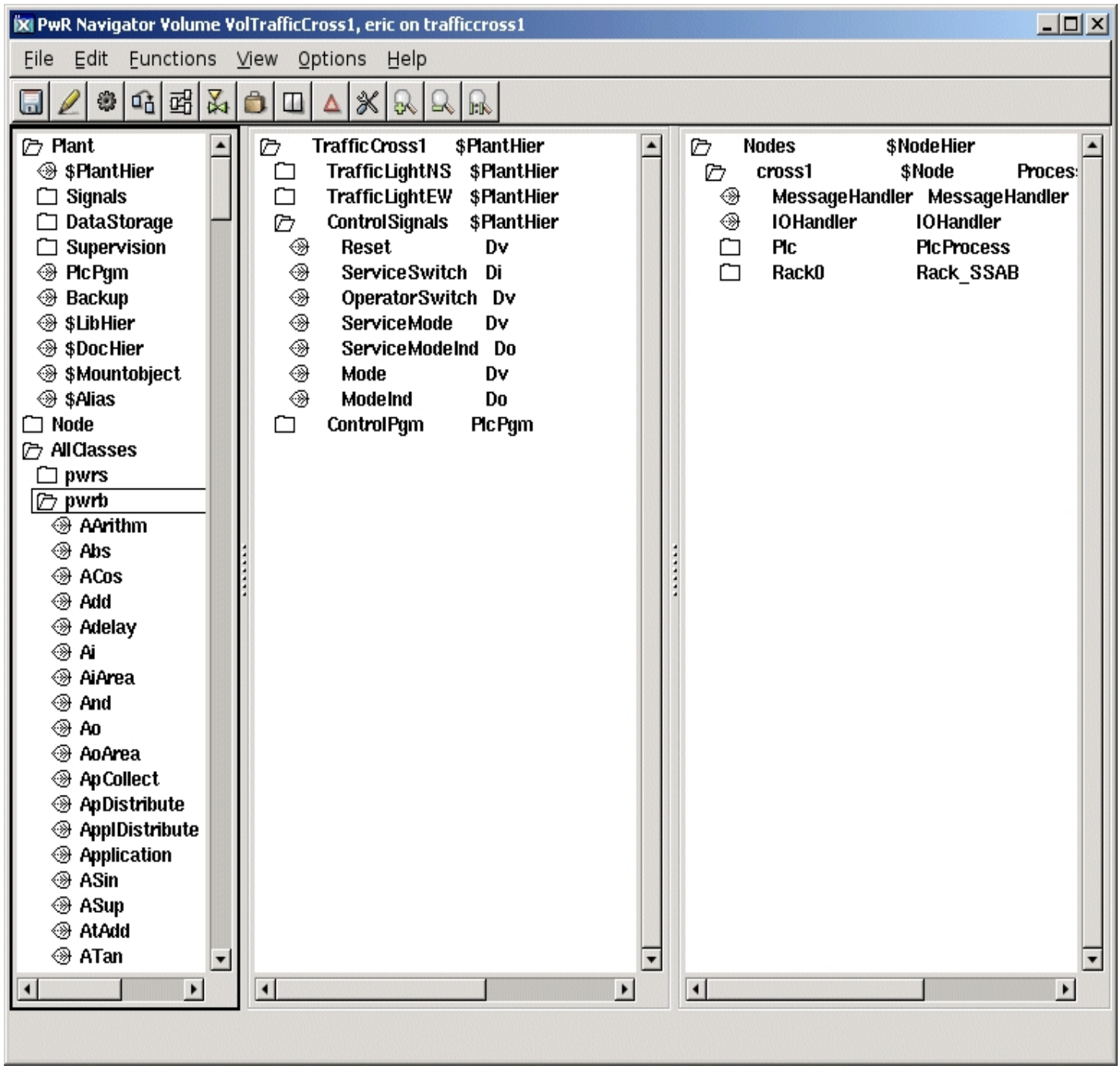
Quando si modifica un volume, si creano nuovi oggetti, si copiano oggetti, si rimuovono oggetti e si modificano i valori degli attributi.

Crea un oggetto

Si crea un oggetto selezionando la classe dell'oggetto nella tavolozza. La tavolozza e' divisa nelle cartelle Plant, Node e AllClasses. In Plant si trovano le classi piu' comuni nella gerarchia dell'impianto, sotto il nodo piu' comune nella gerarchia dei nodi. Se la classe non e' stata trovata qui, tutte le classi sono disponibili in AllClasses.
Qui sono elencati tutti i volumi delle classi e sotto ciascun volume le classi del volume. Successivamente, si fa clic con il tasto centrale del mouse sul futuro fratello o genitore del nuovo oggetto. Se si fa clic sulla mappa/foglia nell'oggetto di destinazione, il nuovo oggetto viene posizionato come primo figlio, se si fa clic a destra della mappa/foglia, viene posizionato come fratello.

Puoi anche creare un oggetto dal menu popup. Seleziona una classe nella tavolozza e apri il menu popup facendo clic MB3 sull'oggetto di destinazione. Attiva 'Create Object' e scegli dove posizionare il nuovo oggetto, relativamente alla destinazione, prima, dopo o come primo o ultimo figlio.

Il Configuratore in modalita' modifica



Elimina un oggetto

Un oggetto viene cancellato dal menu popup. Fai clic MB3 sull'oggetto e attiva "Elimina oggetto".

Sposta un oggetto

Puoi anche spostare un oggetto dal menu popup, ma e' spesso piu' facile usare il pulsante centrale del mouse: seleziona l'oggetto che deve essere spostato e fai clic con il pulsante centrale sull'oggetto di destinazione.
Se si fa clic sulla mappa/foglia sull'oggetto di destinazione, l'oggetto viene posizionato come primo figlio o come fratello.

Nota! Evita di usare Taglia/Incolla per spostare un oggetto. Questo creera' una copia dell'oggetto con una nuova identita' , e i riferimenti all'oggetto potrebbero andare persi. Puoi usare il comando paste/keepoid per mantenere l'identita'.

Copia un oggetto

Puoi copiare un oggetto con copia/incolla o dal menu popup.

- copia incolla. Seleziona l'oggetto o gli oggetti da copiare e attiva 'Edit/Copy' (Ctrl/C) nel menu. Gli oggetti selezionati vengono ora copiati in un buffer per la copia. Seleziona un oggetto di destinazione e attiva 'Edit/Paste' (Ctrl/V). Gli oggetti nel buffer per copia vengono ora posizionati come fratelli sugli oggetti di destinazione. Se invece si attiva 'Edit/Paste Into' (Maiusc + Ctrl/V) i nuovi oggetti vengono posizionati come figli nell'oggetto di destinazione. Se gli oggetti copiati hanno figli, i figli vengono copiati anche da copia/incolla.

- dal menu popup. Seleziona l'oggetto o gli oggetti da copiare, apri il menu popup dall'oggetto di destinazione e attiva 'Copy selected object(s)'. Ora devi scegliere dove posizionare i nuovi oggetti, rispetto all'oggetto di destinazione, come primo o ultimo figlio o come fratello precedente o successivo. Se gli oggetti copiati hanno ascendenti, e devono anche essere copiati, si attiva invece 'Copy selected Tree(s)'.

Cambia il nome dell'oggetto

Il nome di un oggetto viene modificato selezionando l'oggetto e attivando 'Edit/Rename' (Ctrl / N) nel menu. Un campo di input viene aperto nella regione inferiore del configuratore, dove viene inserito il nuovo nome. Un nome oggetto puo' avere un massimo di 31 caratteri.

Puoi anche cambiare il nome visualizzando gli attributi dell'oggetto. In modalita' modifica, il nome dell'oggetto viene visualizzato sopra gli attributi e viene modificato allo stesso modo di un attributo.

Cambia un valore di attributo

Selezionare l'attributo da modificare e attivare 'Functions/Change value' (Ctrl/Q) nel menu. Immettere il nuovo valore nel campo di immissione. Se si desidera interrompere l'immissione, si attiva nuovamente 'Change value'.

Non tutti gli attributi sono modificabili. Dipende dalla funzione dell'attributo, se deve essere assegnato un valore nell'ambiente di sviluppo o meno. Gli attributi modificabili sono contrassegnati da una freccia.

Puoi anche modificare il valore di un attributo dall'editor di oggetti, aperto dal menu popup (Open Object). Un attributo di tipo testo multilinea puo' essere modificato solo dall'editor di oggetti.

Come 'utente avanzato' puoi aprire il campo di input con 'Freccia Destra', come alternativa piu' veloce a 'Change value'.

File di simboli

Il symbolfile e' un file di comando che viene eseguito all'avvio del configuratore.

Puo' contenere definizioni di simboli e altri comandi di configurazione.
Il nome file predefinito del symbolfile e' $ pwrp_login/wtt_symbols.pwr_com.
Ecco alcuni esempi di comandi utili.

Collegamento a qualche parte nella gerarchia del database:

```
define rb9 "show children /name=hql-rb9"
```

20.1 L'editor di oggetti

La quantita' di dati di un oggetto e' divisa in attributi. L'Editor oggetti visualizza gli attributi di un oggetto e il valore di ciascun attributo. Se si e' in modalita' di modifica, e' anche possibile modificare i valori degli attributi.

Gli attributi sono visualizzati nello stesso modo del configuratore, la differenza principale e' che sono visualizzati in una finestra separata.

Navigare

Anche la navigazione e l'assegnazione dei valori vengono eseguiti nello stesso modo del configuratore.

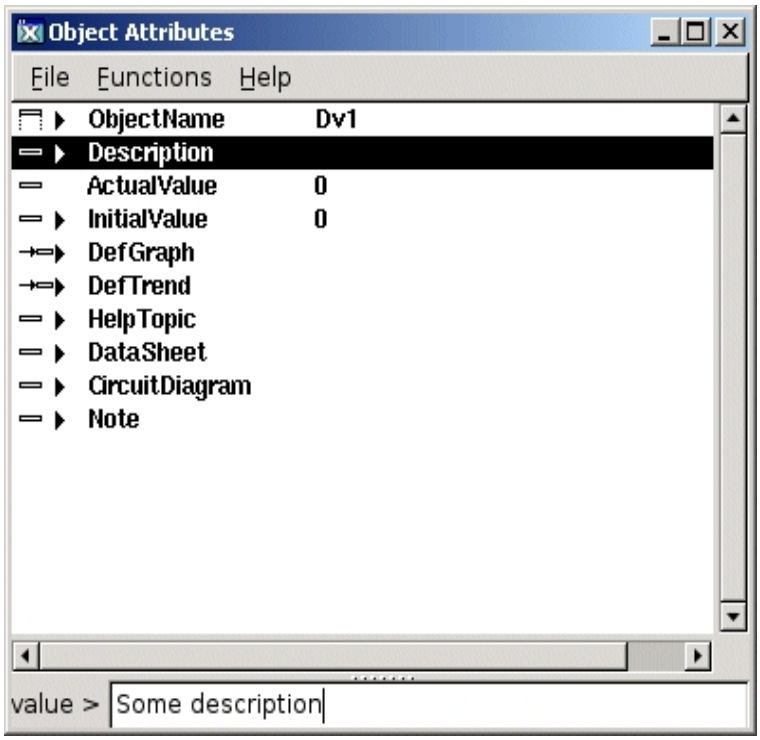
Avvio

L'Object Editor e' aperto dal Configuratore o dall'editor del Plc. Attiva "OpenObject" nel menu popup di un oggetto, oppure seleziona l'oggetto e attiva 'Functions/Open Object' nel menu. Dall'editor del Plc puoi anche avviare l'editor di oggetti facendo doppio clic sull'oggetto. Se l'editor di Configuration/Plc e' in modalita' di modifica, l'Object Editor viene aperto a sua volta in modalita' di modifica.

Menu

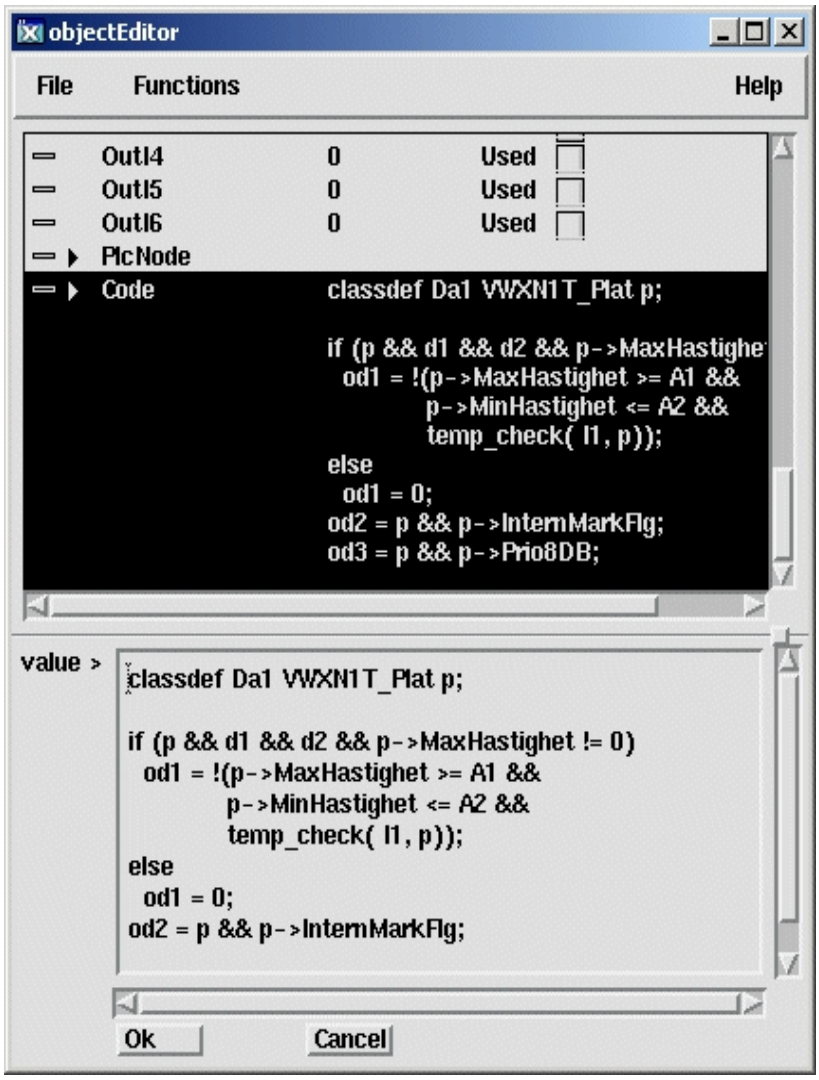
File/Close	chiudi l'editor di oggetti.
Functions/Change value	apri il campo di immissione per l'attributo selezionato. Questo e' consentito solo in modalita' modifica.
Functions/Close change value	Chiudere il campo di immissione.

Editor di oggetti



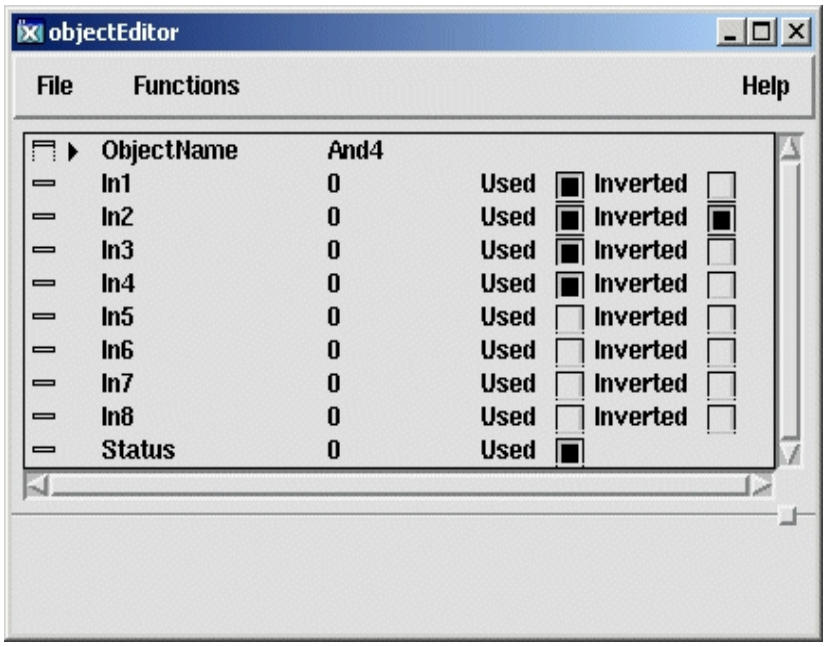
L'Object Editor ha un campo di input per inserire testi multilinea. In questo caso non e' possibile terminare l'input con 'Invio', come per i testi a riga singola. Fai clic sul pulsante "Ok" o attiva 'Functions/Close change value' (Ctrl/T) per terminare.

Un testo multilinea



L'Object Editor per oggetti plc ha funzioni per stabilire quali ingressi o uscite devono essere visualizzati nel blocco funzione. E' anche possibile scegliere se gli ingressi digitali devono essere invertiti.
Questo viene scelto con caselle di controllo rispettivamente per ciascun attributo ('Used' and 'Inverted'). La casella di controllo per 'Used' puo' anche essere modificata dalla tastiera con 'Shift/Freccia Destra' e la casella di controllo 'Inverted' puo' essere cambiata con 'Shift/Freccia Sinistra'

Oggetto Plc con caselle di controllo



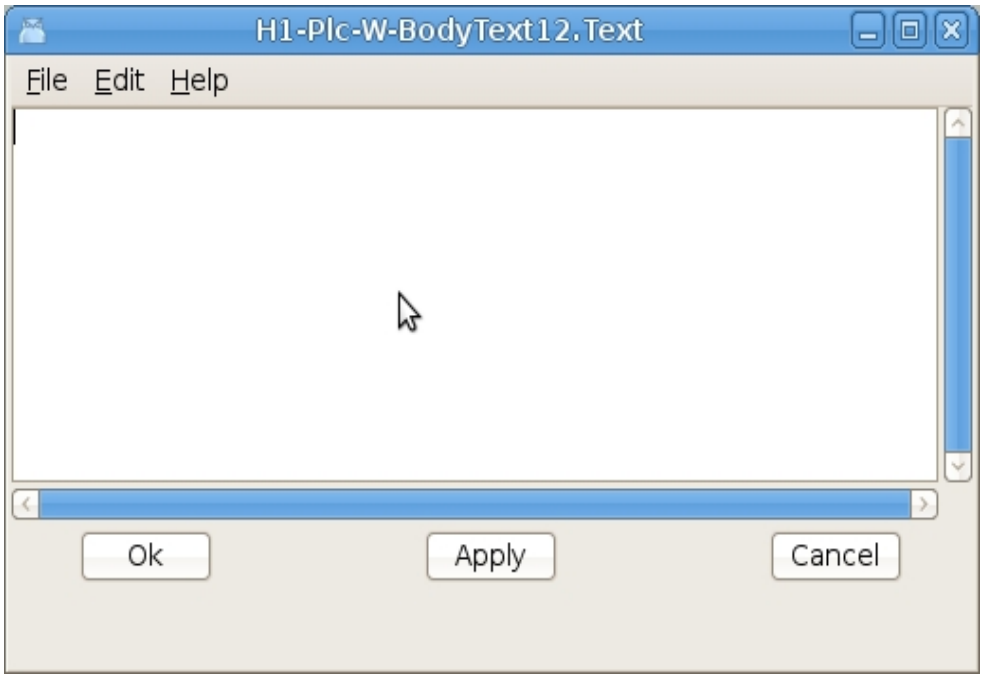
20.2 Object Text Editor

Per gli attributi di testo negli oggetti come BodyText, HelpText, CArithm
e DataArithm e' disponibile un editor di testo speciale. L'editor si apre facendo clic
destro sull'oggetto nell'editor plc e attivando o EditText o EditCode nel menu a comparsa.

Menu

File/Close	chiudi l'editor di testo.
File/Save	Salva il testo nell'attributo dell'oggetto. Si noti che il testo non e' stato salvato nel database fino a quando la sessione dell'editor del plc non viene salvata.
Edit/Copy	Copia il testo selezionato negli appunti.
Edit/Cut	Taglia il testo selezionato.
Edit/Paste	Incolla il testo dagli Appunti nella posizione corrente del cursore.

Object Text Editor



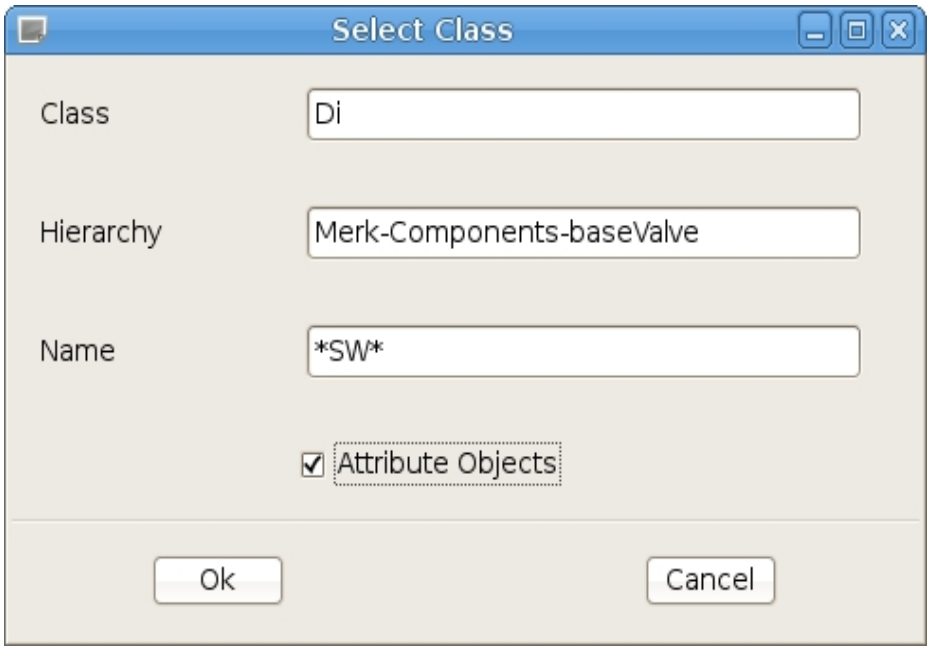
20.3 L'editor di fogli di calcolo

L'editor di fogli di calcolo viene utilizzato per visualizzare o configurare diversi oggetti della stessa classe contemporaneamente. Gli oggetti per una determinata classe, sotto un oggetto specificato nell'albero degli oggetti, sono visualizzati in una tabella nell'editor. Nella tabella vengono visualizzati anche i valori di un attributo negli oggetti e puoi spostarti facilmente tra diversi attributi.

L'editor di fogli di calcolo viene aperto dal configuratore: 'Functions/Spreadsheet' nel menu. Se il configuratore e' in modalita' di modifica, l'Editor foglio di calcolo viene aperto anche in modalita' modifica.

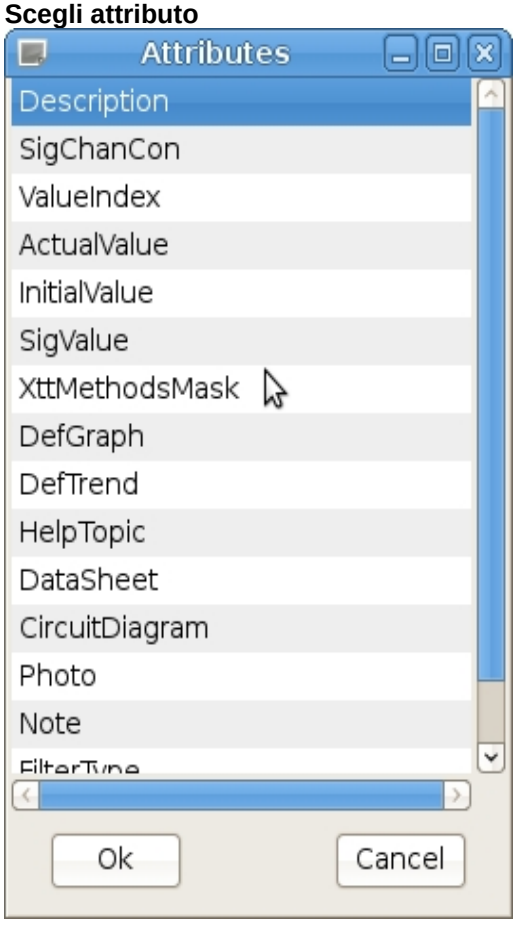
Quando viene avviato l'editor del foglio di calcolo, devi prima indicare quali oggetti devono essere visualizzati, cioe' a quale classe appartengono e sotto quale gerarchia sono posizionati. Questo viene fatto attivando 'File/Select Class' nel menu. Immettere classe, gerarchia e stato se devono essere visualizzati oggetti attributo, ovvero oggetti che si trovano come attributi in altri oggetti.

Scegli la classe e la gerarchia



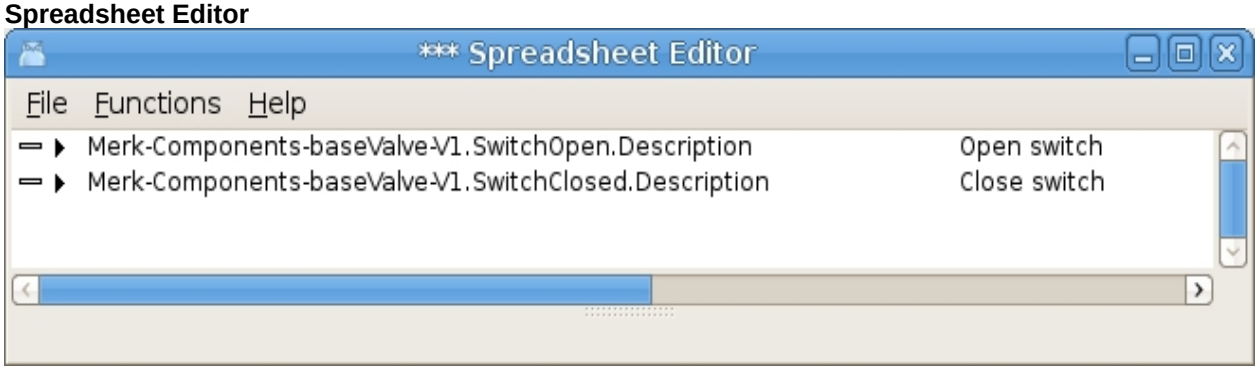
Dopo questo, scegli quale attributo deve essere visualizzato. Seleziona un attributo nell'elenco degli attributi e fai clic su "OK" o fai doppio clic su un attributo.

Scegli attributo



Il risultato e' mostrato nella figura sottostante. Qui e' stato scelto l'attributo 'Description'. Puoi facilmente visualizzare gli altri attributi nell'oggetto attivando 'File/Next Attribute' (Ctrl / N) e 'File/Previous Attribute' nel menu.

Spreadsheet Editor



Menu

File/Select Class	Indica classe e gerarchia per gli oggetti che devono essere visualizzati.
File/Select Attribute	Indica quale attributo deve essere visualizzato.
File/Next Attribute	Mostra l'attributo successivo per l'oggetto nella tabella.
File/Previous Attribute	Mostra l'attributo precedente per gli oggetti nella tabella.
File/Print	Stampare la tabella.
File/Close	Chiudi l'editor di fogli di calcolo.
Functions/Change value	Aprire un campo di input per l'oggetto selezionato.
Functions/Close change value	Chiudere il campo di immissione.

20.4 Finestra di aiuto

La finestra di aiuto e' usata per visualizzare e navigare nei testi di aiuto. I testi di aiuto possono essere vari manuali e guide forniti con Proview o testi di aiuto scritti dal costruttore per descrivere l'impianto e fornire assistenza agli operatori.

20.5 Finestra dei messaggi.

La finestra dei messaggi visualizza i messaggi di Proview che vengono emessi in varie operazioni. I messaggi possono avere cinque livelli di gravita', contrassegnati con colori diversi:

S	Successo	verde
I	Informazione	verde
W	Attenzione	giallo
E	Errore	rosso
F	Fatale	rosso

Se viene visualizzata una freccia davanti al messaggio, il messaggio contiene un collegamento a un oggetto.
Facendo clic sulla freccia, viene visualizzato l'oggetto.

20.6 Utilita'

La finestra delle utilita' e' un'interfaccia grafica per diversi comandi del configuratore.
Per ulteriori informazioni sui comandi, vedere il capitolo Comandi.

20.7 Utilita' di backup

L'utilita' di backup analizza un file di backup di runtime. Rende possibile

- ispezionare il contenuto del file di backup.
- vedere la differenza tra due file di backup archiviati e tempi diversi.
- vedere la differenza tra il file di backup e i valori corrispondenti nel database di sviluppo.
- Trasferire i valori selezionati dal file di backup nel database di sviluppo.

La funzione di backup nell'ambiente di runtime Proview memorizza i valori degli oggetti e degli attributi specificati con gli oggetti di backup nel file di backup.
Viene letto all'avvio di Proview e i valori precedentemente memorizzati vengono inseriti nel database in tempo reale.

Visualizza il contenuto di un file di backup

Aprire l'utilita' di backup dal configuratore del volume di root del nodo da cui viene recuperato il file di backup. L'utilita' di backup viene aperta con il comando 'backup show'.
Attiva File/Open nel menu e seleziona un file di backup.
Il file di backup puo' essere una copia del file di backup corrente o una copia da una precedente occorrenza.
Tutti gli attributi e i loro valori sono visualizzati nella finestra di backup.

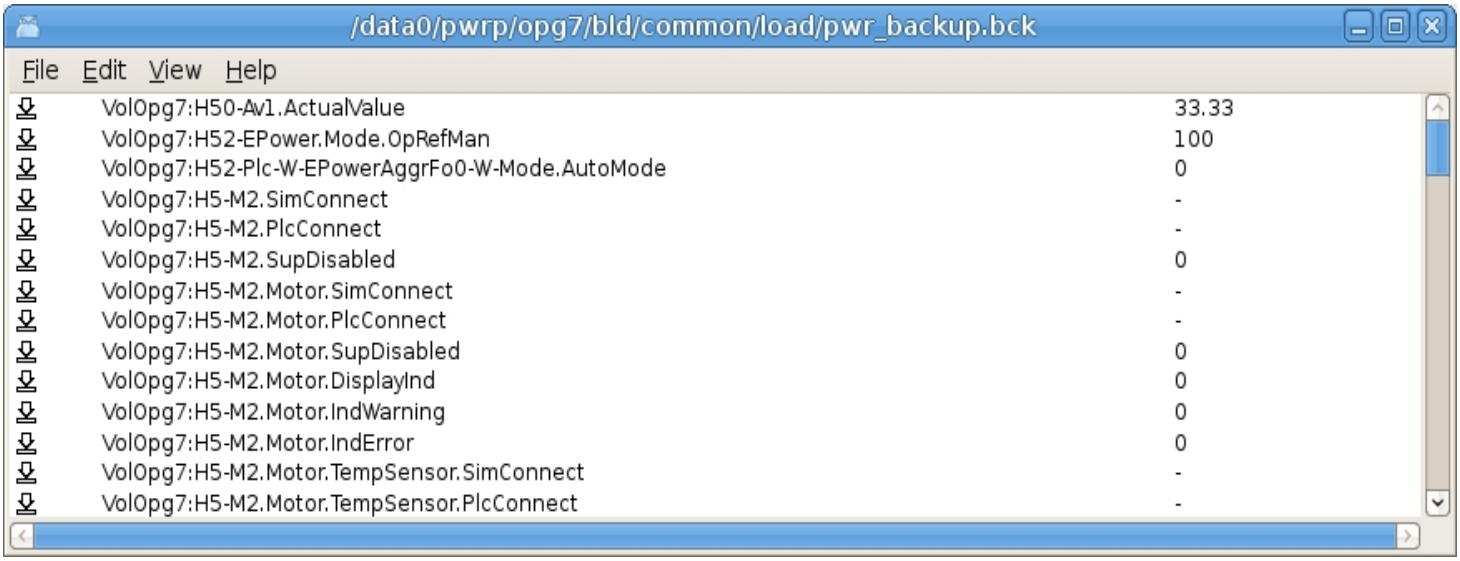


Fig Mostra il file di backup

Confronta due file di backup

Aprire il primo file di backup per la visualizzazione come descritto sopra.
Quindi attivare File/Compare file di backup nel menu e selezionare l'altro file di backup.
Gli attributi con i valori che differiscono tra i file ora vengono visualizzati.

Confronta un file di backup con il database di sviluppo

Aprire il file di backup per la visualizzazione, quindi attivare File/Compare database nel menu.
Gli attributi con i valori che differiscono tra il file di backup e il database di sviluppo ora sono visualizzati.

Trasferire i valori da un file di backup al database di sviluppo

Immettere la modalita' di modifica nel configuratore e quindi aprire l'utilita' di backup.
Aprire il file di backup per la visualizzazione, quindi confrontarlo con i valori nel database di sviluppo da File/Compare Database nel menu.
Gli attributi con i valori che differiscono tra il file di backup e il database ora vengono visualizzati con le caselle di controllo.
Controllare i valori che si desidera trasferire e attivare File/Transfer nel database nel menu.
Infine attiva Salva nel configuratore per memorizzare le modifiche.

20.8 Costruisci directory

La finestra Crea directory visualizza le directory con azioni di compilazione configurate. La casella di controllo per ogni directory mostra se la directory sara' costruita o meno, cioe' se qualcosa deve essere aggiornato nella directory.

Nota! Se un file viene memorizzato dopo l'apertura della finestra Build Directories, e' necessario premere il pulsante di aggiornamento per mostrare lo stato corretto.
In caso contrario, il file memorizzato non verra' aggiornato quando viene creata la directory.

Le cartelle di directory possono essere aperte per visualizzare i file, i makefile o gli script che devono essere aggiornati. Attivando 'Edit/Show all' nel menu vengono visualizzati anche i file e i makefile che non devono essere aggiornati.

Vengono eseguite solo le directory e i file selezionati. e' possibile creare singole directory o file deselezionando le caselle di controllo per altre directory e file.

Premendo il pulsante 'Build Directories' nella barra degli strumenti viene eseguita la build.

Le directory e le azioni di compilazione sono configurate nel volume della directory.

20.9 Costruzione(compilazione) esportazione e importazione

La finestra Esporta Build mostra i file che devono essere esportati in altri progetti o moduli. Ad esempio, i file comuni tra i progetti possono essere file dbs per volumi di classe o h-file per le transazioni tra i nodi. Normalmente i file vengono esportati da un progetto in una directory comune, da cui successivamente vengono importati da altri progetti. Non e' consigliabile importare o esportare direttamente da o ad altri progetti.

Premendo il pulsante "Esporta file" nella barra degli strumenti viene eseguita l'esportazione.

La finestra Build Import visualizza i file che devono essere importati da altri progetti e moduli.

Premendo il pulsante 'Import file' nella barra degli strumenti viene eseguita l'importazione.

21 L'editor Plc

Nell'Editor di Plc si creano programmi di plc in un linguaggio di programmazione grafico.

La programmazione con blocchi funzione viene eseguita in una rete orizzontale di nodi e connessioni da sinistra a destra nel documento. I segnali o gli attributi vengono recuperati sul lato sinistro della rete e i valori vengono trasferiti tramite connessioni dai pin di uscita ai pin di input dei blocchi di funzioni.
I blocchi funzione operano sui valori e sul lato sinistro della rete, i valori sono memorizzati in segnali o attributi.

Le sequenze di Grafcet consistono in una rete verticale di nodi e connessioni. Uno stato viene trasferito tra i passaggi nella sequenza tramite le connessioni. Grafcet e blocchi funzionali possono interagire tra loro e essere combinati in una rete.

Inizio

L'editor di Plc viene aperto dal configuratore. Seleziona un oggetto di classe PlcPgm e attiva 'Functions/Open Program' (Ctrl/L) nel menu, o attiva 'Open Program' nel menu popup per l'oggetto PlcPgm. Il configuratore non dovrebbe essere in modalita' di modifica.

Modalita' di lavoro

L'editor del Plc puo' essere in una delle quattro diverse modalita': Visualizza, Modifica, Traccia e Simula. La modalita' e' selezionata in "Mode" nel menu.

Visualizza
In Visualizza e' possibile guardare il programma, ma non creare o modificare oggetti.
Le alternative di menu per le funzioni di modifica sono disattivate.

Modifica
Se disponi di privilegi di modifica, puoi accedere alla modalita' di modifica.
In questa modalita' e' possibile creare e modificare oggetti.

Traccia e simula
Se si desidera tracciare il programma si entra nella modalita' traccia. Cio' richiede che l'ambiente runtime di Proview sia avviato nella stazione di sviluppo. Simula funziona come traccia, ma puoi anche impostare valori sui segnali.

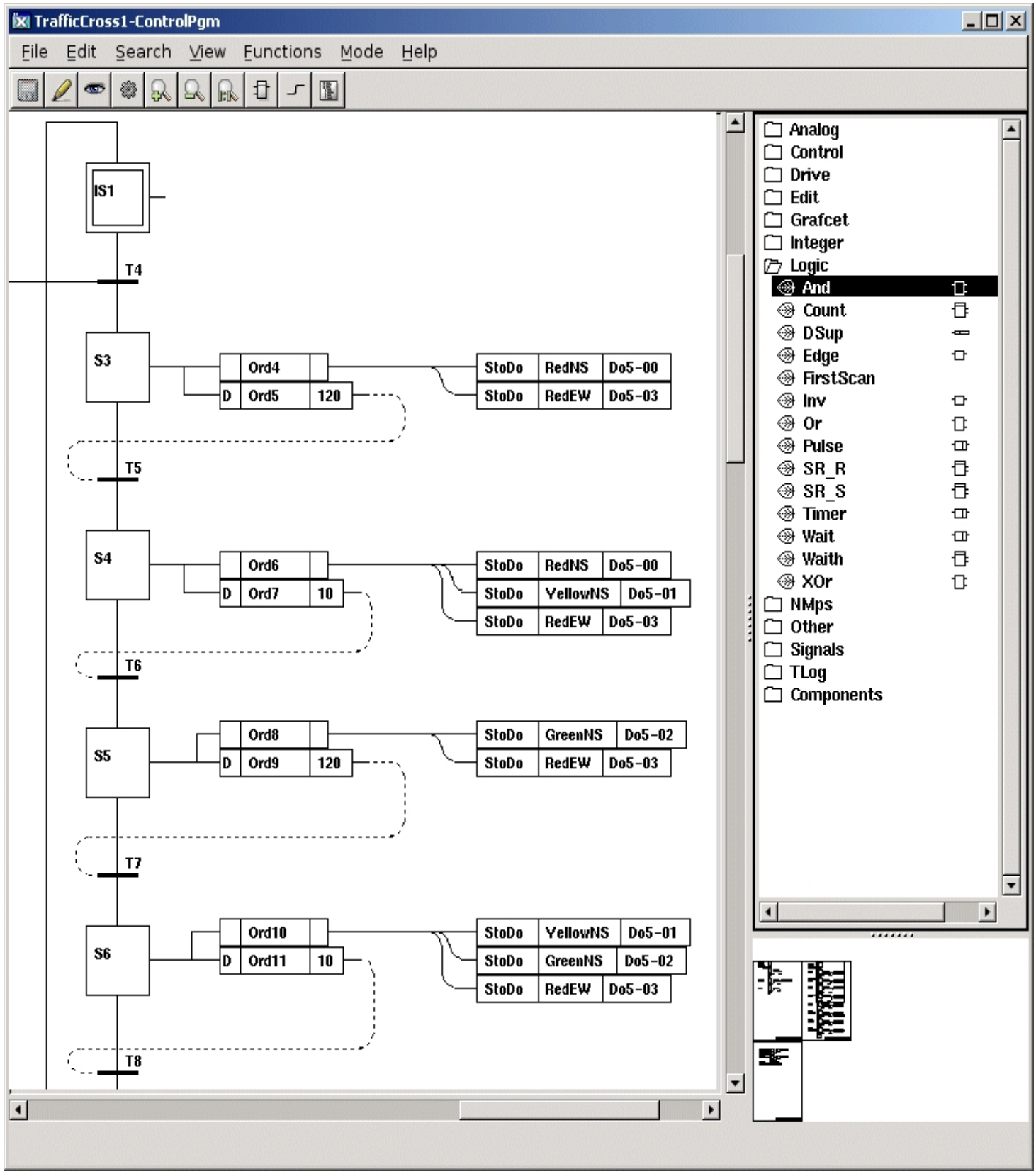
La traccia e' piu' semplice e veloce eseguita da Xtt. Ti consigliamo quindi di utilizzare PlcTrace in Xtt.

Modifica

L'editor del Plc e' composto da

- un'area di lavoro
- due tavolozze, una per gli oggetti funzione e una per le connessioni (e' visibile solo una palette alla volta).
- una finestra di navigazione, dalla quale e' possibile scorrere e ingrandire l'area di lavoro.

L'editor Plc



Le tavolozze

La tavolozza degli oggetti

Quando si avvia l'editor del Plc, viene visualizzata la tavolozza degli oggetti funzione.

Quando si crea un blocco funzione nell'area di lavoro, si sceglie una classe nella tavolozza.

La tavolozza delle connessioni
Quando si creano connessioni tra oggetti, l'editor sceglie un tipo di connessione adatto. Sebbene, in alcuni casi, il costruttore debba influenzare la scelta del tipo di connessione. Questo viene fatto nella tavolozza delle connessioni che viene visualizzata attivando 'View/Palette/Connection' nel menu. Quando la palette e' chiusa, attivando 'View/Palette/Object' o 'View/Palette/Plant', l'editor e' nuovamente responsabile della scelta del tipo di connessione.

Gerarchia dell'impianto
E' possibile visualizzare la gerarchia dell'impianto attivando 'View/Palette/Plant' nel menu. Quando si collegano oggetti funzione ai segnali, ad esempio quando si recuperano valori di segnale, e' possibile indicare quale segnale deve essere prelevato. E' inoltre possibile selezionare il segnale nel configuratore, che in molti casi e' un'alternativa piu' agevole.

Finestra di navigazione

In basso a sinistra c'e' una visione del programma in scala ridotta. La parte dell'area di lavoro visualizzata nella finestra principale e' contrassegnata da un rettangolo.
Spostando il rettangolo (trascina MB1) si scorre la finestra principale.
Puoi anche ingrandire con trascina MB2.

Oggetti funzionali

Crea oggetto

Per creare oggetti l'editor deve essere in modalita' modifica. Attiva la modalita' di modifica da 'Mode/Edit' nel menu.

Per creare un oggetto, selezionare una classe nella tavolozza e fare clic con MB2 (per MB si intende il pulsante centrale del mouse) nell'area di lavoro.

Modificare un oggetto

Un oggetto viene creato con determinati valori predefiniti. Cio' si applica anche a quali input e output sono visualizzati nell'editor di plc e possono essere collegati ad altri oggetti. Se un valore deve essere modificato, l'editor viene aperto per l'oggetto da modificare. L'editor degli oggetti viene aperto nei seguenti modi:

- doppio click sull'oggetto
- activare 'Open Object' nel menu a comparsa per l'oggetto.
- selezionare un oggetto e attivare 'Functions/Open object' nel menu.

Dall'editor di oggetti e' possibile modificare i valori di vari attributi. Gli attributi per un oggetto plc sono separati in attributi di input, attributi interni e attributi di output.

Ingressi
Il valore di un attributo di input viene recuperato da un altro blocco funzione, tramite una connessione.
L'attributo viene visualizzato nel blocco funzione come pin di input. In alcuni casi l'input non viene utilizzato, un and-gate ha per esempio 8 ingressi ma spesso solo due di essi vengono utilizzati. Questo e' controllato dalla casella di controllo 'Used' nell'editor degli oggetti. Se 'Used' e' segnato, gli attributi sono visualizzati con un pin di input, altrimenti sono nascosti.

Ad alcuni attributi di input, in particolare di tipo analogico, puo' essere assegnato un valore nell'editor di oggetti. Se 'Used' non e' segnato per l'attributo, viene utilizzato il valore assegnato. Tuttavia, se 'Used' e' contrassegnato, il valore viene recuperato dall'output a cui e' collegato l'attributo. Questo e' ad esempio la funzione per i valori limite "Min" e "Max" in un oggetto Limit. E' possibile scegliere se recuperare il valore da un altro blocco funzione o assegnare un valore. L'assegnazione funziona in runtime come valore iniziale, che in seguito puo' essere modificato in vari modi.

Alcuni ingressi digitali possono essere invertiti. Per farlo, contrassegna la casella di controllo 'Inverted' nell'editor degli oggetti. Nel blocco funzione questo viene visualizzato con un cerchio sul pin di input.

Attributi interni
Gli attributi interni possono contenere valori di configurazione assegnati nell'ambiente di sviluppo o valori calcolati in runtime. Quest'ultimo tipo non e' modificabile, e forse non e' nemmeno visibile nell'ambiente di sviluppo.

Uscite
Il valore di un attributo di uscita viene trasferito a un input tramite una connessione. Come per un input, puoi scegliere se visualizzare un pin di output o meno con la casella di controllo 'Used' nell'editor di oggetti.

Seleziona un oggetto

Gli oggetti sono selezionati nei seguenti modi

- click MB1 sull'oggetto.
- Shift/Click MB1 aggiunge l'oggetto alla lista degli oggetti selezionati, o lo rimuove se l'oggetto e' gia' selezionato
- trascinando con MB1 puoi selezionare uno o piu' oggetti. Gli oggetti che hanno una parte all'interno del rettangolo di selezione verranno selezionati.
- premendo il tasto Shift e trascinando con MB1 aggiungi gli oggetti che saranno nel rettangolo di selezione alla selectlist.

Gli oggetti selezionati sono disegnati con il colore rosso.

Sposta oggetti

Un singolo oggetto viene spostato posizionando il cursore su di esso e trascinandolo con MB1. Diversi oggetti vengono spostati selezionandoli e trascinando uno degli oggetti con MB1.

Connessioni

Creare connessioni

Un pin di uscita e un pin di ingresso vengono collegati nel modo seguente

- posizionare il cursore sul pin o in un'area dell'oggetto funzione vicino al pin e premere MB2 (il pulsante centrale).
- trascina il cursore sull'altro pin o in un'area dell'oggetto funzione vicino al pin e rilasciare MB2.

Una connessione viene ora creata tra gli oggetti.

Due ingressi sono collegati allo stesso modo, ma alcuni degli ingressi collegati devono

essere collegati a un'uscita e da questa uscita il valore viene prelevato su tutti gli ingressi collegati in parallelo.

Tipi di dati

I valori che vengono trasferiti tra oggetti diversi tramite le connessioni possono essere valori digitali, analogici, interi o stringa. Gli input e gli output collegati devono essere dello stesso tipo. Se sono di tipo diverso, devi utilizzare un oggetto che converta i valori tra i tipi, ad esempio AtoI o ItoA. Questi oggetti di conversione si trovano nella palette 'Signals/Conversion' .

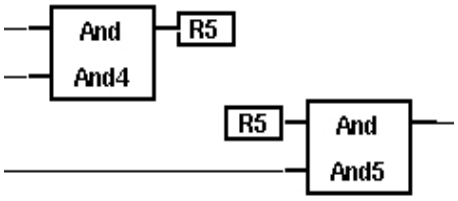
Le connessioni analogiche e integer sono contrassegnate da linee leggermente piu' spesse e le connessioni digitali con linee piu' sottili.

Inoltre c'e' un tipo di connessione per il trasferimento di un riferimento a un oggetto. Questi collegamenti sono disegnati con una linea spessa e tratteggiata.

Connessioni di riferimento

Se l'editor ha difficolta' a trovare un percorso per la connessione tra il pin di input e output, perchÃ© ci sono troppi oggetti tra i due pin o perchÃ© risiedono in documenti diversi, le connessioni vengono disegnate come connessioni di riferimento.
Le connessioni di riferimento possono anche essere disegnate attivando 'View/Reference connection' nel menu.

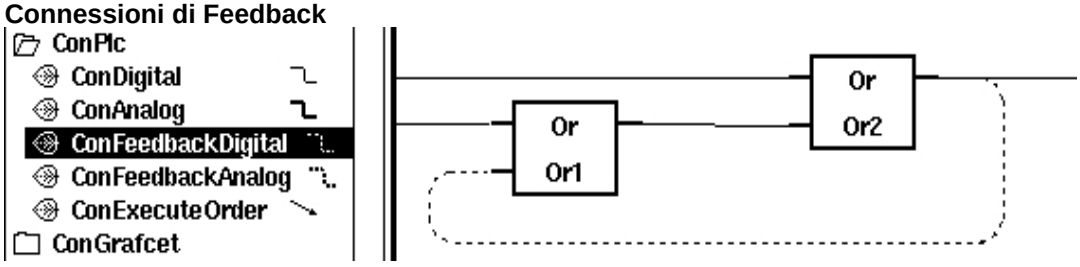
Connessioni di riferimento



Ordine in cui vengono eseguite le funzioni

Oltre al trasferimento di un valore di segnale, le connessioni determinano anche l'ordine di esecuzione tra diversi blocchi funzione. Se due oggetti sono connessi tramite un'uscita e un ingresso, normalmente l'oggetto di uscita deve essere eseguito prima dell'oggetto di input. Ma a volte e' necessario un feedback in rete, e quindi si affronta un loop di ordini di esecuzione. Per determinare l'ordine di esecuzione e' necessario specificare il feedback con una connessione di tipo ConFeedbackDigital o ConFeedbackAnalog. Questi sono selezionati nella tavolozza delle connessioni, visualizzata attivando 'View/Palette/Connection' nel menu. Sotto la cartella 'ConPlc' puoi trovare le connessioni di feedback.
Sono disegnate con linee tratteggiate.

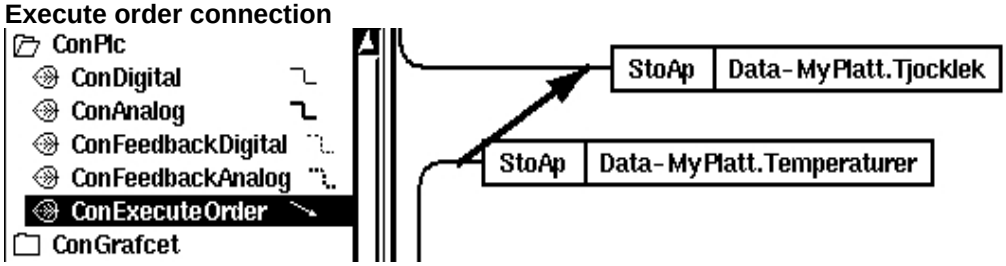
Connessioni di Feedback



Here you can also find the connection type 'ConExecuteOrder'. In some cases you want to control the execute order between to function blocks, though they are not connected to each other.

Then you can draw a ConExecuteOrder between them (between which input or output doesn't matter). The connection is to be drawn from the object that is to execute first, to the object that is to execute last. In the figure below, the storage of the attribute 'Temperaturer' is done before the storage of the attribute 'Tjocklek'.

Execute order connection



Recupera e archivia i valori dei segnali

Recupera il segnale e i valori degli attributi

Nella parte sinistra della rete dei blocchi funzione, vengono acquisiti i valori dei segnali e degli attributi.
Il recupero viene eseguito da oggetti come GetDi, GetDo, GetDv, GetIi ecc. Il recupero dei valori degli attributi viene eseguito da GetDp, GetIp, GetAp e GetSp.
Questi oggetti si trovano nella cartella 'Signals' nella tavolozza. Quando viene creato un oggetto di questo tipo, devi indicare quale segnale o quale attributo deve essere recuperato.
Il modo piu' semplice per farlo e' selezionare il segnale/attributo nel configuratore e fare un Ctrl/Doppio click MB1 sull'oggetto.
Il segnale/attributo viene quindi visualizzato nel blocco funzione e, se il segnale e' un segnale di ingresso, viene anche visualizzato il canale del segnale.

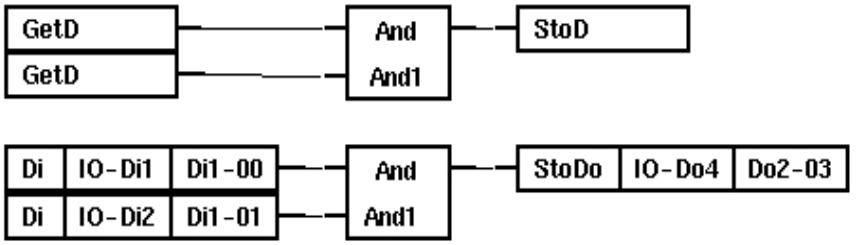
C'e' un modo piu' veloce per creare questi oggetti. Se si disegna una connessione da un pin di input in un oggetto funzione e la si rilascia in uno spazio vuoto nell'area di lavoro, viene creato un oggetto Get generico con il tipo di dati dell'ingresso, ovvero GetDgeneric, GetIgeneric, GetGeneric o a GetSgeneric. Quando si specifica il segnale o l'attributo, l'oggetto Get deve essere recuperato, l'oggetto Get generico viene convertito in un oggetto Get del tipo corretto per il segnale o l'attributo. Se si sceglie un Dv nel configuratore, un GetDgeneric verra' convertito in un GetDv quando si fa un Ctrl/Doppio click su MB1.

Memorizza il segnale e i valori degli attributi

Nella parte destra della rete i valori calcolati sono memorizzati in segnali e attributi.
L'archiviazione viene eseguita da oggetti come StoDo, StoDv, StoDp, StoIo ecc.
Il metodo per specificare il segnale o l'attributo da connettere e' lo stesso degli oggetti Get, cioe' selezionando il segnale/attributo nel configuratore e facendo un Ctrl/Douppio click MB1 sull'oggetto.

Se si disegna una connessione da un pin di uscita in un blocco funzione, viene creato un oggetto Sto generico, che viene convertito in un oggetto Sto di tipo adatto quando connesso ad un segnale o attributo. Se si desidera memorizzare valori con Set o Reset (ad esempio SetDo o ResDo), non e' possibile utilizzare questo metodo.
Devi creare gli oggetti dalla tavolozza.

Oggetti generici Get e Sto



Sottofinestre

Alcuni oggetti contengono sottofinestre, ad es. CSub, SubStep, Trans, Order. Un oggetto con una sottofinestra e' contrassegnato da una spessa linea grigia da qualche parte nel blocco funzione. Una sottofinestra viene aperta in diversi modi:

- selezionando l'oggetto e attivando 'Function/Subwindow' nel menu.
- attivando 'Subwindow' nel menu popup dell'oggetto.
- facendo clic sull'oggetto con Shift/Doppio click MB1.

Per creare una nuova sottofinestra si agisce nel modo seguente (il fatto che una sola sessione di modifica alla volta possa essere aperta, lo rende un po' complicato)

- crea l'oggetto che contiene la sottofinestra
- salva
- apri la sottofinestra
- lascia la modalita' di modifica nella finestra principale
- entra nella modalita' di modifica nella sottofinestra

Controlla l'ordine di esecuzione

Normalmente non e' necessario considerare l'ordine di esecuzione dei diversi blocchi funzione in una finestra. PoichÃ© i segnali sono copie degli I/O, ovvero ad ogni scansione del programma plc, viene eseguita una copia di tutti i valori di segnale prima dell'esecuzione e quindi questa copia non cambia durante l'esecuzione.
La memorizzazione ed il recupero dei valori dei segnali non saranno influenzati dall'ordine di esecuzione tra singoli oggetti di memorizzazione o di recupero.

Tuttavia, se si memorizza e si recupera il valore di un attributo, che non e' gestito dall'immagine (copia) I/O, l'ordine di esecuzione potrebbe essere importante per la funzione.

L'ordine di esecuzione e' determinato dalle connessioni tra i blocchi funzione. Le connessioni comuni sono sia il trasferimento del segnale che la determinazione dell'esecutore. Se si effettua un feedback, e' necessario scegliere un tipo di connessione che trasferisca il segnale, ma non determinare l'esecuzione dell'ordine.
Le diverse connessioni di feedback sono di questo tipo. Inoltre, esiste una connessione che determina l'esecuzione ma non il trasferimento del segnale del nodo, ConExecuteOrder. Con questo e' possibile controllare l'ordine di esecuzione tra diversi blocchi funzione senza trasferire alcun valore di segnale.

L'ordine di esecuzione per i blocchi funzione in una finestra PLC viene visualizzato con 'View/Show execute order' nel menu. Il numero visualizzato per ciascun blocco funzione indica l'ordine in cui vengono eseguiti. L'oggetto senza un numero non contiene alcun codice eseguibile.

L'ordine di esecuzione tra diversi PlcPgm e' controllato dall'attributo executeOrder nell'oggetto PlcPgm. L'ordine di esecuzione determina l'ordine all'interno di un thread. Valori inferiori di ExecuteOrder vengono eseguiti prima di quelli piu' alti.

Compilare

Prima che una finestra PLC possa essere eseguita, e' necessario compilare. Allo stesso tempo, viene eseguito un controllo della sintassi del codice plc. Se la sintassi non e' corretta, viene visualizzato un messaggio nella finestra dei messaggi. Il messaggio di errore puo' essere di tipo Error o Warning. Error indica un errore grave a cui e' necessario prestare attenzione. Facendo clic sulla freccia davanti al messaggio ,nella finestra dei messaggi, l'oggetto errato viene visualizzato nell'editor del plc.

Dopo il controllo della sintassi, il codice C viene generato e inviato al compilatore C.
Se esiste un oggetto con C-code definito dall'utente, ad esempio CArithm o DataArithm, il compilatore C puo' trovare errori che segnalara' nella finestra del terminale.
Quindi visualizza sempre la finestra del terminale per controllare che la compilazione abbia avuto successo.

La compilazione viene evviata dalla voce menu 'File/Build'.

Se si desidera verificare la sintassi senza generare alcun codice, si attiva 'File/Syntax'.
Il compilatore c non viene attivato, quindi gli errori del codice c definito dall'utente non vengono rilevati.

Taglia e incolla

L'editor del plc contiene un buffer per le operazioni di incolla. Il buffer e' comune a tutte le finestre, il che rende possibile copiare tra finestre diverse. Con le funzioni 'Edit/Copy' e 'Edit/Cut' nel menu, gli oggetti selezionati vengono copiati nel buffer (Taglia li rimuove anche dall'area di lavoro). La funzione 'Edit/Paste' copia il buffer nell'area di lavoro.
Gli oggetti copiati vengono ora spostati utilizzando il cursore e li si posiziona nella posizione voluta facendo clic su MB1 per bloccarli.

Taglia, Copia e Incolla puo' anche essere attivato dalla tastiera con Ctrl/X, Ctrl/C e Ctrl/V.

Oggetti speciali del Plc

Di seguito sono descritti un numero di oggetti che hanno funzioni speciali nel programma plc.

Documento
L'oggetto documento viene utilizzato per dividere il codice in pagine, quando si stampa il codice. Quando apri una nuova finestra, questa contiene un oggetto documento. Dall'editor degli oggetti e' possibile modificare la dimensione del documento ed inserire la firma ed il numero di pagina. Altre informazioni nell'intestazione del documento vengono compilate automaticamente. L'oggetto documento si trova nella cartella 'Edit' nella tavolozza degli oggetti.

ShowPlcAttr
ShowPlcAttr puo' essere utilizzato come estensione dell'intestazione del documento.
In questo oggetto vengono visualizzate le informazioni su volume, scantime e reset dell'oggetto per le sequenze Grafcet.

Head, Title, Text e BodyText
Questi oggetti sono usati per scrivere testo informativo nel documento. Head, Title e Text contengono testi su linea singola di diverse dimensioni con un massimo di 79 caratteri. Bodytext contiene un testo multilinea con un massimo di 1023 caratteri.
Questi oggetti si trovano sotto 'Edit' nella palette.

Point
L'oggetto Point e' un punto di connessione libero utilizzato per diramare una connessione o per controllare il layout di una connessione. Point si trova sotto 'Edit' nel menu.

Grafcet
Le sequenze Grafcet sono costruite con oggetti Grafcet specifici come InitStep, Step, Trans e Order.
Le connessioni tra gli oggetti seguono regole specifiche. I pin verticali in un oggetto Step sono ad esempio collegati a oggetti Trans e il pin orizzontale e' collegato a un oggetto order. Ecco un esempio di come creare una sequenza Grafcet.

Inizia creando un oggetto InitStep. Disegna una connessione dal pin inferiore e rilasciala nell'area di lavoro sotto l'oggetto InitStep. Ora viene creato un oggetto Trans, che e' connesso all'oggetto InitStep. Disegna una connessione dal pin inferiore dell'oggetto Trans e rilascialo nello spazio di lavoro sotto l'oggetto Trans. Qui viene creato un oggetto Step.
Se si disegna una connessione dal pin inferiore degli oggetti Step, viene creato un altro oggetto Trans.
Se si desidera un ramo della sequenza, si disegna una connessione aggiuntiva dal pin inferiore dell'oggetto Step. Ora viene creata una divergenza con le specifiche connessioni StepDiv.
Se si crea nello stesso modo un ramo da un oggetto Trans, disegnando su due connessioni dal pin inferiore, viene creato un ramo parallelo, con le connessioni TransDiv contrassegnate da doppie linee.
Se si disegna una connessione dal piedino orizzontale di un passo, viene creato un oggetto Ordine e cosi' via.
Come puoi vedere questo e' un modo veloce per costruire sequenze complesse.

SCANTIME
ScanTime traccia il tempo reale di esecuzione, cioe' il tempo trascorso dall'ultimo giro di programma.

FirstScan
FirstScan e' true il primo giro dell'esecuzione di plc dopo l'avvio di Proview.
E' anche true dopo un soft restart.

Menu

File/Save	Salva
File/Print/Documents	Stampa tutti i documenti
File/Print/Overview	Stampa una panoramica
File/Print/Selected documents	Stampa i documenti selezionati
File/Syntax	Eseguire un controllo della sintassi del codice
File/Build	Compilare il programma
File/Plc Attributes	Aprire l'editor degli oggetti per l'oggetto PlcPgm
File/Delete Window	Elimina la finestra di plc
File/Save Trace	Salva traceobjects
File/Restore Trace	Ripristina traceobjects salvati in precedenza

File/Close	Chiudi la finestra
Edit/Undo Delete	Annulla l'ultima azione di cancellazione
Edit/Undo Select	Reimposta la lista di selezione
Edit/Cut	Taglia gli oggetti selezionati
Edit/Copy	Copia l'oggetto selezionato nel buffer
Edit/Paste	Copia il buffer nell'area di lavoro
Edit/Connect	Connetti l'oggetto selezionato al segnale o all'attributo selezionato nel configuratore
Edit/Delete	Elimina oggetti selezionati
Edit/Change Text	Cambia il testo nell'oggetto di testo selezionato
Edit/Expand Object	Espandi l'oggetto selezionato
Edit/Compress Object	Comprimi l'oggetto selezionato
Search/Object	Cerca un nome oggetto
Search/String	Cerca una stringa
Search/Next	Cerca ulteriormente con la stessa stringa
View/Palette/Object	Mostra la tavolozza degli oggetti delle funzioni
View/Palette/Connection	Mostra la tavolozza di connessione
View/Palette/Plant	Mostra la gerarchia impianto
View/Reference connections	Crea connessioni come connessioni di riferimento
View/Grid Size	Imposta la dimensione della griglia
View/Show Grid	Mostra la griglia
View/Zoom/In	Ingrandire
View/Zoom/Out	Zoom indietro
View/Zoom/Reset	Ripristina il fattore di zoom originale
View/Show Execute Order	Mostra ordine di esecuzione per gli oggetti delle funzioni
View/Redraw	Ridisegna le connessioni e ridisegna la finestra
Functions/Open Object	Aprire l'editor di oggetti per l'oggetto selezionato
Functions/Subwindow	Apri la sottofinestra per l'oggetto selezionato
Mode/View	Modalita' di visualizzazione
Mode/Edit	Modalita' Modifica
Mode/Trace	Modalita' traccia
Mode/Simulate	Modalita' simulazione

Funzioni del mouse

Area di lavoro

Click MB1	Seleziona un oggetto. Fare clic in uno spazio vuoto ripristinera' l'elenco di selezione
Shift/Click MB1	Aggiungi oggetto all'elenco di selezione
DoppioClick MB1	Apri l'editor di oggetti
Shift+Ctrl/DoppioClick MB1	Copia nel buffer. Clicca su un oggetto copia l'oggetto, clicca nello spazio vuoto copia gli oggetti seleziona
Trascina MB1	Su un oggetto: sposta oggetto o sposta oggetti selezionati Nello spazio vuoto: seleziona gli oggetti all'interno del rettangolo di selezione
Shift/Trascina MB1	Aggiungi oggetti all'interno del rettangolo contrassegnato all'elenco di selezione
Click MB2	Crea oggetto
DoppioClick MB2	Elimina. Fare clic sull'oggetto elimina l'oggetto, fare clic

	nello spazio vuoto cancella tutti gli oggetti selezionati
Shift+Ctrl/Click MB2	Incolla. Copia il buffer nell'area di lavoro
Shift+Ctrl/DoppioClick MB2	Taglio. Facendo clic su un oggetto elimina l'oggetto, fare clic in uno spazio vuoto cancella gli oggetti selezionati. Gli oggetti eliminati sono salvati nel buffer
Press MB3	Popup menu

Navigation window

Drag MB1	Scorri l'area di lavoro
Drag MB2	Zoom area di lavoro

22 File di aiuto

I testi di aiuto vengono visualizzati nella finestra della guida che puo' essere aperta dal configuratore e dall'ambiente operatore. I testi di aiuto sono scritti in un file $pwrp_exe/xtt_help.dat.
I testi di aiuto sono suddivisi in argomenti e ogni argomento ha una chiave, questa chiave viene specificata quando viene visualizzato il testo della guida per aprire l'argomento.
I collegamenti nel testo della guida rimandano ad altri argomenti, questo rende possibile navigare nei vari contesti di aiuto.

L'argomento 'index' e' l'argomento radice che viene visualizzato da diverse utilita'

- 'Help/Project' nel menu del configuratore.
- 'Help/Project' nel navigatore del runtime.
- Il pulsante 'Help' nella finestra operatore.

Argomenti specifici della guida possono essere aperti dai grafici Ge usando pulsanti (actiontype Help) o dal menu a comparsa per un oggetto nell'ambiente operatore (method 'Help').

22.1 Conversione

Il testo della guida puo' essere convertito in formato html, PDF e PostScript. Quando convertito in html, ogni argomento viene convertito in una pagina html. Quando vengono convertiti in PDF e PoscScript, sono disponibili numerosi tag aggiuntivi per creare un documento del testo della guida con capitoli e intestazioni.

La conversione viene eseguita con il comando (programma) 'co_convert'.

Conversion to html

Un file di aiuto viene convertito in html con il comando

```
co_convert -f [-d outputdirectory] 'helpfile'
```

Esempio
```
co_convert -f -d $pwrp_web $pwrp_exe/xtt_help.dat
```

Conversione in postscript

Un file di aiuto viene convertito in PostScript con il comando

```
co_convert -n [-d outputdirectory] 'helpfile'
```

Esempio
```
co_convert -n -d $pwrp_lis $pwrp_exe/xtt_help.dat
```

Conversione in PDF

Un file di aiuto viene convertito in PDF con il comando

```
co_convert -f [-d outputdirectory] 'helpfile'
```

Esempio

```
co_convert -f -d $pwrp_lis $pwrp_exe/xtt_help.dat
```

22.2 Codifica

La codifica predefinita del file della guida e' ISO 8859-1. UTF-8 puo' essere specificato con un'istruzione di codifica sulla prima riga del file di help, ad es

```
Coding:UTF-8
```

I valori supportati per la codifica sono UTF-8 and ISO8859-1.

22.3 Sintassi

Esistono diversi tag che influenzano la ricerca e la conversione del file di aiuto.

topic	Definisce il testo della guida per un argomento
bookmark	Definisce una posizione all'interno di un argomento
link	Link a un argomento o a un URL
index	Elenco di argomenti
h1	Intestazione 1
h2	Intestazione 2
b	Testo grassetto
c	Codice
t	Tabulatore
hr	Linea orizzontale
include	Includi altri file di aiuto

Tag PDF e PostScript

I seguenti tag vengono utilizzati per formattare i testi della guida quando vengono convertiti in PDF e PostScript

chapter	Dividere argomenti in capitoli
headerlevel	Aumentare o diminuire il livello dell'intestazione
pagebreak	Nuova pagina
option	Opzioni
style	Specifica lo stile del testo

Titolo pagina e informazioni sul documento

Esempi

22.3.1 Argomento

<topic>

<topic> inizia un argomento e dovrebbe essere collocato nella prima posizione di una linea. Il topic-tag dovrebbe essere seguito dalla chiave che la funzione di guida cerchera'. Tutte le seguenti linee fino a ul </topic> tag verranno visualizzate come testo dell'argomento.

<topic> 'key'

</topic>
Fine di un'argomento. </topic> dovrebbe essere collocato nella prima posizione di una riga.

Esempio
<topic> avvio motore
Il motore verra' avviato da ...
</topic>

Il comando

```
wtt> help avvio motore
```

mostrera' il testo di questo argomento.

22.3.2 Segnalibro

<bookmark>
Bookmark e' una linea all'interno di un argomento che puo' essere trovata da un tag di collegamento o dal qualificatore /bookmark nel comando help. Il tag del segnalibro deve essere posizionato alla fine della riga e dovrebbe essere seguito da un nome.

'del testo' <bookmark> 'nome'

Esempio
Questo e' un segnaibro. <bookmark> primo_motore

Il comando

```
wtt> help avvio motore/bookmark=primo_motore
```

mostrera' il testo dell'argomento e scorrera' fino al segnalibro.

22.3.3 Collegamento

<link>
Il tag <link> e' un collegamento ad un'altro argomento di aiuto. Il tag <link> deve essere piazzato alla fine della linea. Quando viene attivata la linea del collegamento, verra' visualizzato l'argomento del collegamento. Il tag link dovrebbe essere seguito dall'argomento e puo' anche essere seguito da un segnalibro e dal file di help in cui risiede l'argomento, separati da una virgola. Se una linea contiene un collegamento, verra' contrassegnata con una freccia.

'del testo' <link> 'topic'[,'bookmark'][,'helpfile']

Esempio
Collegamento a primo motore <link> visualizza motore, primo_motore

22.3.4 Indice

<index>
Il tag <index> e' un collegamento speciale che visualizza un indice del file di aiuto, questo e' una lista di tutti gli argomenti in ordine alfabetico.

'del testo' <index>

22.3.5 Header1

<h1>
Il tag <h1> mostrera' una linea come intestazione con dimensioni di testo piu' grandi.
Il tag deve essere posizionato all'inizio della riga. Una linea di intestazione
non puo' contenere alcun link

<h1>'testo di intestazione'

Esemplio
<h1>Questa e' un'intestazione h1
sara' visualizzato come

Questa e' un'intestazione h1

22.3.6 Header2

<h2>
Il tag <h2> mostrera' una linea di intestazione con testo in grassetto circondato
da linee grigie. Il tag deve essere posizionato all'inizio della riga.
Una riga di intestazione non puo' contenere alcun collegamento.

Esempio
<h2>Questa e' un'intestazione h2
sara' visualizzato come

Questa e' un'intestazione h2

22.3.7 Grassetto

<b>
Il tag <b> mostrera' una linea con testo in grassetto.
Il tag deve essere posizionato all'inizio della riga.

Esempio
<b>Questa e' una linea in grassetto
sara' visualizzato come
Questa e' una linea in grassetto

22.3.8 Codice

<c>
Il tag <c> visualizzera' una riga con il carattere Courier utilizzato per il codice.
Il tag deve essere posizionato all'inizio della riga.

Esempio
<c>for (i = 0; i < 10; i++)
sara' visualizzato come
`for ( i = 0; i < 10; i++)`

22.3.9 Tab

<t>
Il tag <t> rende possibile scrivere in colonne. Solo tre colonne
(two <t> tags) sono ammessi.

Esempio
Col1 <t> Col2 <t> Col3
sara' visualizzato come
Col1 Col2 Col3

22.3.10 Linea orizzontale

<hr>
Il tag <hr> mostrera' una linea orizzontale.
Il tag deve essere posizionato all'inizio della riga.

Esempio
<hr>
sara' visualizzato come

22.3.11 Include

<include>
Include un'altro file fi aiuto. Il tag <include> non deve essere insertito
all'interno di un tag argomento.

<include> 'filename'

22.3.12 Capitolo

<chapter>
Questo tag divide gli argomenti in capitoli. Un capitolo inizia con <chapter> e finisce con
</chapter>. Il titolo del primo argomento nel capitolo sara' l'intestazione del capitolo.

</chapter>
Termina un capitolo.

Esempio
<chapter>
<topic>
Introduzione
...
</topic>
</chapter>

22.3.13 Livello di intestazione

Divide gli argomenti in un capitolo in livelli di intestazione.

<headerlevel>
Aumenta il livello dell'intestazione

</headerlevel>
Riduce il livello di intestazione

22.3.14 Interruzione di pagina

<pagebreak>
Forza l'interruzione della pagina

22.3.15 Opzioni

<option>
L'opzione puo' avere i seguenti valori

printdisable	Ignora i tag e il testo fino al prossimo 'printenable' per i file PDF e PostScript. Normalmente utilizzato per collegamenti che non hanno alcun utilizzo nei PDF e PostScript.
printenable	Resetta il 'printdisable'.

Esempio
<option> disable
Del testo
...
<option> enable

22.3.16 Stile

<style>
Specifica che un argomento deve essere scritto in uno stile specifico.
Styles

function	Stile utilizzato per funzioni e comandi. Grande titolo e interruzione di pagina dopo ogni argomento.

Esempio
<topic> MiaFunzione <style> function
...
</topic>

22.3.17 Pagina del titolo e informazioni sul documento

La pagina del titolo e la pagina per le informazioni sul documento possono essere create con due argomenti speciali.

__DocumentTitlePage

Argomento per frontespizio. E' posto come primo in un file di aiuto.

<topic> __DocumentTitlePage
...
</topic>

__DocumentInfoPage

Argomento per informazioni sul documento, ad es. Copyright.
Viene posizionato dopo il frontespizio.

<topic> __DocumentInfoPage
...
</topic>

22.3.18 Esempio di file di aiuto

<topic> helpfile_example
Start and stop of engines.

Engine 1 <link> helpfile_example, bm_engine_1
Engine 2 <link> helpfile_example, bm_engine_2
Characteristics <link> helpfile_example, bm_char

<h1>Engine 1 <bookmark> bm_engine_1
Start engine one by pressing the start button.
Stop engine one by pressing the stop button.

<h1>Engine 2 <bookmark> bm_engine_2
Start engine two by pressing the start button.
Stop engine two by pressing the stop button.

<h2>Characteristics <bookmark> bm_char

<b><t>Engine1 <t>Engine2
Max speed <t> 3200 <t> 5400
Max current <t> 130 <t> 120
</topic>

Questa e' l'aspetto di questo esempio

22.3.18.1 Start and stop of engines.

Engine 1
Engine 2
Characteristics

Engine 1

Start engine one by pressing the start button.
Stop engine one by pressing the stop button.

Engine 2

Start engine two by pressing the start button.
Stop engine two by pressing the stop button.

Characteristics

	Engine1	**Engine2**
Max speed	3200	5400
Max current	130	120

23 Utenti

Questo capitolo descrive come creare un utente in Proview e come concedere privilegi e accesso per l'utente.

La crescente disponibilita' del sistema Proview per diversi tipi di utenti, ad esempio tramite la intranet, ha comportato l'aumento delle richieste di poter limitare la capacita' dei vari utenti di influenzare il sistema.
Proview contiene un database utente, in cui si definiscono gli utenti per i diversi sistemi e in cui si ha la possibilita' di raggruppare i sistemi con utenti comuni. Il database e' progettato per far fronte alle richieste di aumentare il controllo degli accessi e, allo stesso tempo, per mantenere l'amministrazione a un livello basso.

23.1 Database utente

Il database utente e' popolato da gruppi di sistema e utenti. Quando viene avviata un'utilita' Proview, ad esempio l'ambiente operatore o l'ambiente di sviluppo, viene verificato che l'utente esista nel database e che i privilegi dell'utente siano registrati. I privilegi determinano cio' che un utente e' autorizzato a fare nel sistema.

Systemgroup

Il concetto systemgroup viene introdotto per non dover definire ogni sistema nel database. Invece si definiscono i gruppi di sistema e si collegano i numeri di sistemi a ciascun gruppo di sistema.
Questi sistemi condivideranno gli utenti.

Il database e' costituito da una gerarchia di gruppi di sistemi. La gerarchia ha due funzioni, per descrivere la connessione tra i diversi gruppi di sistemi e per introdurre l'eredita' tra i gruppi di sistemi. I gruppi di sistemi piu' in basso nella gerarchia possono ereditare attributi e utenti dai gruppi di sistemi piu' in alto nella gerarchia.

Se un gruppo di sistema ereditera' gli utenti o no, e' determinato dall'attributo UserInherit. Se l'attributo e' impostato, il systemgroup ereditera' tutti gli utenti dal suo gruppo utente principale.
Anche gli utenti che il genitore ha ereditato dal suo genitore sono ereditati.
Un gruppo di sistema puo' sovrascrivere un utente ereditato definendo il nome utente nel proprio gruppo di sistema.

A un gruppo di sistemi viene fatto riferimento con 'path'-name nella gerarchia, in cui i nomi sono separati da punti, ad es. 'ssab.hql.se1', dove ssab e' il gruppo radice e se1 il livello piu' basso nella gerarchia.

Un sistema Proview e' connesso a un gruppo di sistema dichiarando il gruppo di sistema nell'oggetto System. Se il systemgroup non e' presente nel database degli utenti, sebbene sia un genitore o un antenato, si suppone che il systemgroup erediti gli utenti dall'antenato.

Attributi

Attributo	Descrizione
UserInherit	Il systemgroup eredita gli utenti dal suo systemgroup padre, anche gli utenti che il genitore ha ereditato.

Utenti

A user is characterized by a username, a password and a set of privileges. A user is also connected to a systemgroup.

I privilegi definiscono cio' che un utente e' autorizzato a fare in Proview. Alcuni privilegi influenzano l'accesso per apportare modifiche dalle utilita' di Proview, ad es. il navigatore o plc-editor, alcuni riguardano la costruzione della grafica degli operatori, per controllare quali campi di input e pulsanti possono essere usati da un utente.

Un nome utente puo' essere collegato a piu' gruppi di sistemi, ma dal punto di vista del database, sono utenti diversi, con password e privilegi unici. Hanno solo lo stesso nome utente.

Privilegi

Privilegio	Descrizione
RtRead	Accesso in lettura a rtdb. Privilegi predefiniti per l'utente che non ha effettuato l'accesso
RtWrite	Accesso in scrittura a rtdb. Permette all'utente di modificare rtdb dalla modalita' xtt e simulazione e trace
System	Privilegio per system manager
Maintenance	Privilegio per tecnico di manutenzione
Process	Privilegio per tecnico di processo
Instrument	Privilegio per tecnico strumentista
Operator1	Privilegio per operatore
Operator2	Privilegio per operatore
Operator3	Privilegio per operatore
Operator4	Privilegio per operatore
Operator5	Privilegio per operatore
Operator6	Privilegio per operatore
Operator7	Privilegio per operatore
Operator8	Privilegio per operatore
Operator9	Privilegio per operatore
Operator10	Privilegio per operatore
DevRead	Privilegio di lettura per il workbench
DevPlc	Privilegio di scrittura nell'editor plc
DevConfig	Privilegio di scrittura del configuratore
DevClass	Privilegio di scrittura nell'editor di classi (non ancora implementato)

23.2 Esempio

Proview user database V1.0.0

```
ssab
. . . . . . sysansv    System DevRead DevPlc DevConfig (14680068)
. . . . . . skiftel    Maintenance DevRead (2097160)
. . . . . . 55         Operator1 (64)
. hql                  UserInherit
```

```
. . . . . . anna       RtWrite Operator4 (514)
. . bl2
. . . . . . anna       Operator4 (512)
. . bl1                UserInherit
. . . . . . 55         Operator1 (64)
. . . . . . carlgustav Operator8 (8192)
. hst
. . . . . . magnus     Operator1 (64)
. . rlb                UserInherit
. . . . . . amanda     Operator4 (512)
```

Guarda l'esempio sopra. Questo e' un elenco di un database utente. A sinistra, vengono visualizzati i gruppi di sistema e il numero di punti indica il loro livello nella gerarchia. Nella stessa riga viene scritto l'attributo del gruppo di sistema. Sotto ogni gruppo di sistema, vengono trovati i suoi utenti con privilegi. Pertanto ssab systemgroup ha gli utenti sysansv, skiftel e 55.

Il systemgroup sasb.hql.bl1 ha l'attributo UserInherit, che ha come risultato che eredita gli utenti dal suo genitore. Anche il padre ssab.hql ha UserInherit, cioe' ssab.hql.bl1 eredita anche da ssab. Gli utenti di ssab.hql.bl1 sono quindi, sysansv, skiftel, anna 55 e carlgustav. Qui l'utente 55 di sasb.hql.bl1 sovrascrive l'utente 55 di ssab.

Il systemgroup ssab.hql.bl2 non ha UserInherit e ha solo l'utente anna.

Il systemgroup ssab.hst.rlb ha UserInherit e eredita dal suo padre ssab.hst. Tuttavia, questo non ha UserInherit e non ha ereditato dal suo padre ssab. Gli utenti di ssab.hst.rlb sono quindi amanda e magnus.

Ad un sistema con il systemgroup sandviken.hql verra' negato l'accesso perchÃ© manca il systemgroup e tutti i suoi antenati.

Un sistema con systemgroup ssab.vwx.n2 ereditera' gli utenti dal systemgroup ssab , cioe' sysansv, skifel e 55. Tutti i gruppi di sistemi non devono essere presenti nel database, l'esistenza di un antenato e' sufficiente. Si suppone che quelli non trovati abbiano l'attributo UserInherit.

23.3 Login

Questa sezione descrive come funzionano il Login ed il controllo di accesso nei diversi ambienti Proview.

Ambiente di sviluppo

All'avvio del configuratore, viene aperta una finestra di accesso in cui e' possibile indicare nome utente e password. Puoi anche dare il nome utente e la password come argomenti al workbench se vuoi evitare la procedura di login. Per aprire il configuratore, e' necessario il privilegio DevRead e per accedere alla modalita' di modifica e' necessario DevWrite. Per modificare nell'editor di plc, hai bisogno di DevPlc.

Ambiente operatore

Quando l'ambiente operatore viene avviato con un argomento OpPlace, l'utente viene recuperato dall'attributo UserName nell'oggetto utente corrispondente.
Per effettuare modifiche nel database dal navigatore di runtime, e' richiesto il privilegio RtWrite. Nella grafica di processo ci sono pulsanti, cursori, ecc. dai quali si influenza il database. Questi oggetti hanno un attributo di accesso, che determina quali sono i privilegi necessari per attivare l'oggetto. Questi privilegi sono associati ai privilegi degli utenti e, se non gli viene concesso nessuno privilegio, gli viene negato l'accesso.

Dal navigatore di runtime, e' possibile con il comando login / logout, accedere come un altro utente e quindi modificare i propri privilegi.

Accesso Web

Per la grafica di processo sul web e' disponibile un frame di accesso speciale che puo' essere aggiunto al menu di avvio della home page del progetto. Questo e' abilitato nell'attributo OpPlaceWeb.EnableLogin. Il frame di accesso controlla nome utente e password.

E' possibile specificare un gruppo di sistema specifico per gli utenti Web nell'attributo WebSystemGroup nell'oggetto $Security. Il gruppo predefinito e' common.web, che non esiste nel database utente del modello. Viene quindi utilizzato il primo gruppo principale esistente, comune, e gli utenti di questo gruppo hanno accesso al Web. Se si desidera limitare l'accesso Web, creare il gruppo common.web senza UserInherit o creare un altro gruppo e inserirlo nell'oggetto $Security. Crea anche gli utenti appropriati nel gruppo.

Modifica il database utente

Il database utente viene modificato dall'amministratore o dal configuratore. E' aperto da File/Open/UserDatabase nel menu.
Vedi capitolo Amministrazione

Puoi anche modificare il database con i comandi in pwr_user,
Man kan \E4ven modificare il database con i comandi in pwr_user, vedi pwr_user nel capitolo Tools.

24 L'editor di classi

Questa sezione descrive come creare nuove classi in Proview.
Esistono diversi casi in cui potresti prendere in considerazione la creazione di una nuova classe.

Oggetti Data

Si desidera memorizzare i dati in una struttura dati, ad esempio per accedere facilmente ai dati dalle applicazioni.
Puoi anche creare oggetti dati che descrivono il materiale che passa attraverso un'impianto, dove l'oggetto dati contiene proprieta' per un materiale, ad es. lunghezza, larghezza ecc.
Il materiale puo' essere spostato tra le celle NMps per indicare la posizione di un materiale nell'impianto.

Oggetto funzione Plc

Un oggetto funzione utilizzato nella programmazione di plc consiste in una classe che definisce i pin di input e output dell'oggetto funzione e i possibili attributi interni.
Questo tipo di oggetti comprende anche codici che vengono eseguiti dal programma plc.
E' possibile scegliere di creare il codice come codice plc o codice c.

Componenti

Un oggetto componente riflette un componente nell'impianto ed e' spesso diviso in due o tre classi diverse, un oggetto principale, un oggetto funzione e un oggetto bus, possibilmente anche un oggetto simulato. L'oggetto principale e' collocato nella gerarchia dell'impianto e contiene i segnali che sono collegati al componente, oltre ad altri dati di configurazione.
Un oggetto funzione, inserito in un programma plc, e' collegato all'oggetto principale e lavora in parte con i dati provenienti dai propri input e output e in parte con segnali e altri parametri nell'oggetto principale. Se lo scambio di segnali avviene tramite Profibus, si puo' anche creare un oggetto modulo speciale che contiene oggetti canale per i dati trasportati sul Profibus. E' sufficiente creare una connessione tra l'oggetto principale e l'oggetto modulo per connettere tutti i segnali e i canali nel componente. L'oggetto di simulazione e' un oggetto funzione, che e' collegato all'oggetto principale e che simula il componente quando il sistema viene eseguito in modalita' simulazione.

Sottoclassi di componenti

Proview contiene una serie di classi di base per valvole, motori ecc. Questi sono progettati in modo generale per coprire un gran numero di componenti. Spesso, si crea una sottoclasse adattata a un componente specifico e che, ad esempio, contiene un collegamento a un foglio dati, un testo della guida ecc. Per questo componente. Creando una sottoclasse di un basecomponent erediti tutti i metodi e gli attributi da questo, ma hai anche la possibilita' di aumentare la funzionalita' con piu' attributi e piu' codice plc.

Aggregati

Un aggregato riflette una parte dell'impianto che contiene un numero di componenti.
In questo caso, e' possibile creare una classe aggregata che contiene i diversi componenti in forma di oggetti attributo. Nell'aggregato, c'e' anche un oggetto funzione, che chiama

gli oggetti funzioni per i componenti presenti. Gli aggregati possono anche contenere altri aggregati e dare origine a strutture di oggetti piuttosto estese. In linea di principio, e' possibile costruire un impianto come un singolo oggetto, ma in pratica e' opportuno mantenere la struttura dell'oggetto a un livello piuttosto basso. Soprattutto quando si hanno diversi aggregati identici di cui si beneficia creando un oggetto aggregato di una parte di impianto.

24.1 Struttura Database

Oggetto

Nel capitolo Struttura del database c'e' una descrizione di come vengono costruiti gli oggetti. Ora c'e' motivo di andare un po' oltre nell'argomento.

Un oggetto consiste in una testa (head) dell'oggetto ed un corpo dell'oggetto. L'oggetto head contiene informazioni sul nome dell'oggetto, classe e relazione con altri oggetti.
Il corpo dell'oggetto contiene i dati dell'oggetto.

Testata dell'oggetto

Un oggetto ha un nome con una dimensione massima di 31 caratteri che viene memorizzato nell'intestazione dell'oggetto.

Nell'intestazione dell'oggetto c'e' anche un collegamento alla descrizione della classe dell'oggetto. La descrizione della classe contiene informazioni su come interpretare i dati dell'oggetto, come e' suddiviso in diversi attributi e il tipo di attributi. Troverai anche i metodi che funzionano sull'oggetto.

Un oggetto e' posto in una struttura ad albero e la testa dell'oggetto contiene un puntatore ai parenti piu' stretti: padre, fratello all'indietro, fratello in avanti e primo figlio.

La struttura di una testata di oggetto e' comune a tutti i tipi di oggetti.

Corpo dell'oggetto

Un oggetto puo' avere due corpi, un corpo che contiene i dati necessari in runtime.
Puo' anche contenere un corpo aggiuntivo con dati che esistono solo nell'ambiente di sviluppo.

Un corpo e' diviso in attributi che contengono dati di un tipo specifico, ad esempio un booleano, un float32 o un int32. Ma un attributo puo' anche avere un tipo di dati piu' complesso, come una matrice o una classe.

RtBody
RtBody e' il corpo che esiste nel database di runtime. Il corpo e' anche presente nell'ambiente di sviluppo, per rendere possibile impostare valori per diversi attributi nel corpo.

DevBody
Alcuni oggetti hanno anche un DevBody, un corpo che esiste solo nel database di sviluppo e che non e' caricato nel database di runtime. Questo corpo e' usato principalmente da oggetti plc, dove per esempio il devbody contiene dati grafici per l'editor di plc.

24.2 Descrizione della classe

Il layout di un corpo dell'oggetto e' descritto nella descrizione della classe dell'oggetto. Qui trovi anche metodi e altre proprieta' della classe. La descrizione della classe e' costruita con specifici oggetti di definizione di classe che risiedono in un volume di classe. Il volume della classe ha una sintassi rigorosa su come creare le descrizioni delle classi. Segue una presentazione dei diversi oggetti che fanno parte della descrizione della classe.

Volume di classe

Le descrizioni delle classi risiedono in un tipo specifico di volume, un volume di classe. Questi possono contenere due gerarchie, una gerarchia con descrizioni di classi e una con descrizioni di tipi.

$ClassHier

Le descrizioni delle classi si trovano nell'oggetto radice "Class" di tipo $ ClassHier. Sotto l'oggetto $ClassHier, gli oggetti $ ClassDef definiscono le classi nel volume.

$ClassDef

Un oggetto $ClassDef con i suoi discendenti, descrive una classe. Il nome dell'oggetto fornisce il nome della classe. Sotto il $ClassDef, possono essere localizzati i seguenti oggetti

- un oggetto $ObjBodyDef, 'RtBody', che descrive il corpo del runtime.
- un oggetto $ObjBodyDef, 'DevBody', che descrive il corpo nell'ambiente di sviluppo.
- un oggetto Template, ovvero un oggetto della classe corrente che contiene valori predefiniti per oggetti di istanza della classe.
- uno o piu' oggetti del corpo che contengono dati per una funzione specifica.
- un oggetto PlcTemplate, che puo' essere aperto dall'editor di plc e che contiene il codice plc per la classe.
- oggetti del menu che definiscono il menu popup nel navigatore, configuratore e xtt.
- oggetti metodo che si collegano a metodi chiamati ad esempio quando gli oggetti vengono creati o spostati nell'ambiente di sviluppo.

$ObjBodyDef

Un oggetto $ObjBodyDef puo' avere il nome 'RtBody', quindi descrivere il corpo runtime o il nome 'DevBody' e descrivere il corpo dello sviluppo. L'attributo 'StructName' contiene il nome della c-struct della classe nel file incluso che viene generato per il volume.
Sotto l'oggetto $ObjBodyDef, si trova un oggetto attributo per ogni attributo nel corpo dell'oggetto. Gli oggetti $Attribute sono utilizzati per oggetti dati e $Input, $Intern e $Output per oggetti funzione plc.

$Attribute

Un oggetto $ Attribute descrive un attributo in un corpo. L'attributo puo' essere del seguente tipo:

- un tipo di base, ad es. Boolean, Float32, Time, Int16.
- un tipo derivato, ad es. String80, Text1024, URL.
- una matrice di un tipo di base o di un tipo derivato.
- un'altra classe.
- un'array di classe.

- un puntatore rtdb, cioe' un puntatore che puo' essere interpretato da tutti i processi.
- un puntatore privato, cioe' un puntatore che e' valido su un singolo processo.

Il tipo e' indicato nell'attributo "TypeRef". Nell'attributo 'Flag' si specifica se l'oggetto descrive una matrice, un puntatore, una classe ecc. Se l'oggetto descrive una matrice, il numero di elementi e' indicato in 'Elements'.

$Input

$Input descrive un pin di input in un oggetto funzioni nel programma plc. L'input puo' essere di tipo Boolean, Float32, Int32, String80, Time, DeltaTime o di tipo datatype (puntatore a Float32).
$Input da' origine a un attributo con due elementi, un elemento del tipo dichiarato e un elemento con un puntatore al tipo indicato. Se l'input e' connesso, il puntatore punta all'attributo output collegato, se il puntatore non e' collegato punta al suo primo elemento, dove e' possibile specificare un valore per l'input.

L'attributo 'PgmName' indica il nome dell'attributo in c-struct e 'GraphName' la stringa di testo che viene scritta nell'oggetto funzione sul pin di input.

$Intern

Definisce un attributo interno in un oggetto funzione, cioe' un attributo che non e' nÃ© un input nÃ© un output.

$Output

$Output descrive un pin di output in un oggetto funzione. Gli stessi tipi di dati sono validi sia per $Output che per $Input.

$Buffer

$ Buffer specifica un attributo che contiene dati di una certa dimensione che solo una singola funzione e' in grado di interpretare. I dati sono descritti da una classe, ma non sono visualizzabili per esempio xtt. PlcNode, che si trova in tutti gli oggetti plc, e' un esempio di $ Buffer. Qui trovi le informazioni grafiche che interessano solo l'editor di plc.

Classe body

Una classe puo' contenere un oggetto body di classe. L'oggetto body class contiene dati che sono comuni a tutte le istanze della classe. Un esempio di oggetto body class e' $GraphPlcNode che risiede in tutte le classi plc. $GraphPlcNode contiene dati per la generazione del codice ed il layout grafico dell'oggetto funzione.

Menu

Gli oggetti menu vengono utilizzati per definire i menu popup per gli oggetti nell'ambiente di sviluppo e nell'ambiente operatore. $Menu definisce un menu popup nell'ambiente di sviluppo e $RtMenu nell'ambiente operatore. Sotto l'oggetto del menu, le alternative di menu sono definite dagli oggetti $MenuButton e dai sottomenu con gli oggetti $MenuCascade. Gli oggetti del menu sono posizionati sotto l'oggetto $ClassDef.

L'oggetto menu chiama i metodi, ad esempio le funzioni c collegate all'ambiente di sviluppo o dell'operatore. Per il momento non c'e' possibilita' di farlo da un progetto. Questo deve essere fatto dal codice sorgente Proview.

$Menu

Gli oggetti $Menu descrivono i menu a comparsa nell'ambiente di sviluppo. Il nome dell'oggetto specifica la funzione, la prima parte indica lo strumento (Navigator/Configurator). Le ultime cinque lettere indicano durante quali condizioni il menu e' presente, a seconda di quali oggetti sono selezionati o puntati.

```
char 1: P sta per punto, cioe' l'oggetto a cui punta il puntatore.
char 2: indica cosa dovrebbe essere puntato: 'o' un oggetto, 'a' un attributo,
        'c' una classe nella tavolozza.
char 3: 's' sta per selezionato, cioe' l'oggetto che e' selezionato.
char 4: indica quale dovrebbe essere l'oggetto selezionato: 'o' un oggetto,
        'a' un attibuto, 'c' una classe nella tavolozza.
char 5: indica se selezionato e puntato dovrebbe essere lo stesso oggetto:
        's' stesso oggetto, 'n' differenti oggetti.
```

Esempio ConfiguratorPosos: 'Po' il puntatore punta su un oggetto, 'so' e' selezionato un oggetto, 's' l'oggetto al quale punta il puntatore e l'oggetto selezionato e' lo stesso oggetto.

$RtMenu

Gli oggetti del menu che descrivono i menu a comparsa nell'ambiente operatore.

$MenuButton

Definisce un'alternativa di menu in un menu a comparsa.

24.3 Tipo descrizione

Le descrizioni dei tipi, come descrizioni di classi, risiedono in un volume di classe. Sono collocati in una gerarchia separata sotto un oggetto $ TypeHier. I tipi sono divisi in due categorie, tipi base e tipi derivati.

Tipi di base

I tipi di base sono definiti nel volume di sistema pwrs. Esempi di tipi di base sono Boolean, Float32, Int32, String, Enum e Mask.

Tipi derivati

I tipi derivati possono essere definiti in qualsiasi classevolume. Sono costituiti da
- matrici di tipi di base, ad es. String80.
- tipi di enumerazione, Enum, con stringhe di caratteri definite per vari valori.
- bitmasks, Mask, con stringhe definite per i vari bit.

$TypeHier
Le descrizioni dei tipi sono poste sotto l'oggetto radice "Type" della classe $TypeHier. L'oggetto $TypeHier ha oggetti $Type e $TypeDef come figli.

$Type
Descrizione di un tipo di base. Questo oggetto e' riservato al volume di sistema pwrs.

$TypeDef

Descrizione di un tipo derivato. L'attributo 'TypeRef' contiene il tipo di base. Gli usi piu' comuni sono stringhe e testi con dimensioni specifiche, tipi di enumerazione e maschere di bit.

Per definire un tipo di enumerazione, il tipo di base dovrebbe essere $Enum. Sotto l'oggetto $TypeDef, i testi per valori diversi sono definiti con oggetti $Value. Quando deve essere visualizzato il valore di un attributo del tipo, viene visualizzato il testo corrispondente al valore. Quando viene assegnato un valore all'attributo, i diversi testi vengono visualizzati con le caselle di controllo e si deve selezionare un'alternativa.

Per definire maschere di bit viene utilizzato il tipo di base $Mask. Sotto l'oggetto $TypeDef, i testi sono definiti per bit diversi da oggetti $Bit. Quando viene assegnato un valore all'attributo, i testi vengono visualizzati con caselle di controllo, come per i tipi di enumerazione. Per le maschere di bit e' possibile scegliere diverse alternative.

$Value
Definisce un valore in un tipo di enumerazione. Il valore corrisponde a un testo, che viene visualizzato nella configurazione e in xtt quando l'attributo e' aperto. Nel file di inclusione per il volume, viene creata una dichiarazione enum che puo' essere utilizzata nel codice c.

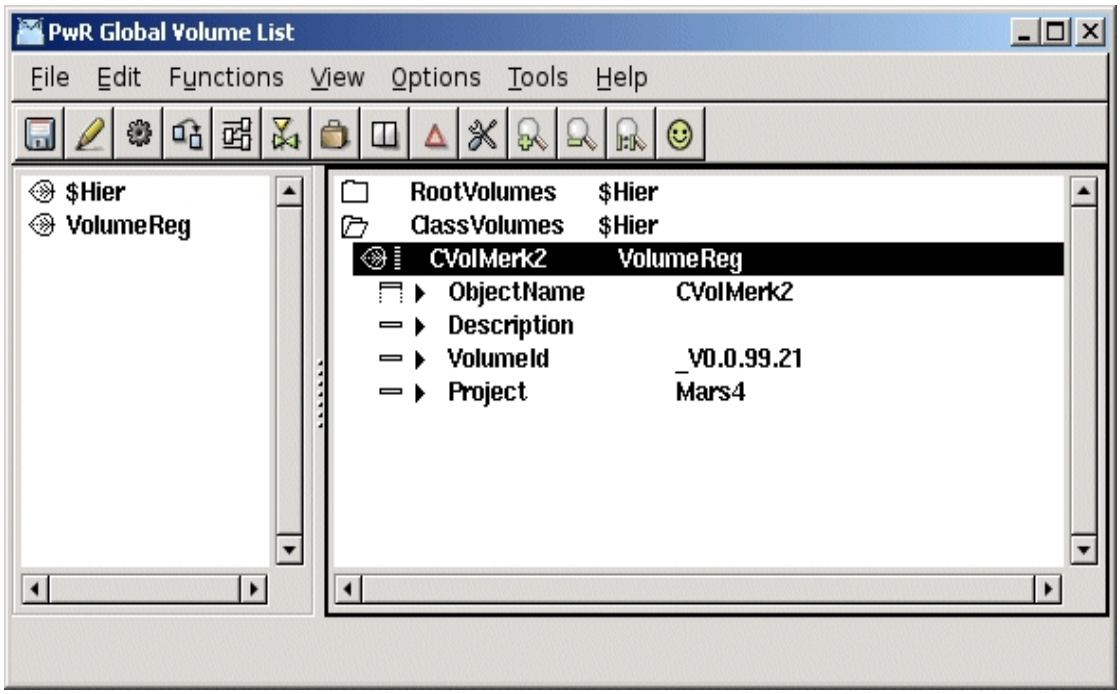
$Bit
Definisce un bit in una maschera di bit. Il bit corrisponde a un testo che viene visualizzato nel configuratore e in xtt quando viene aperto un attributo del tipo. Nel file di inclusione per il volume, viene creata una dichiarazione enum che puo' essere utilizzata nel codice c.

24.4 Creare classi

24.4.1 Crea un volume di classe

Gli oggetti classdefinition risiedono in un volume di classi e prima il classvolume deve essere registrato e creato.

La registrazione viene effettuata nell'elenco globale dei volumi che viene aperto da File/Open/GlobalVolumeList nel menu del navigatore. Qui si crea un oggetto VolumeReg con il nome del volume e l'identita' del volume appropriati. L'identita' del volume per i volumi di classe utente deve essere nell'intervallo 0.0.2-249.1-254. Utilizzare preferibilmente il prefisso CVol nel nome per indicare che si tratta di un volume di classe.
Indicare anche il progetto corrente.

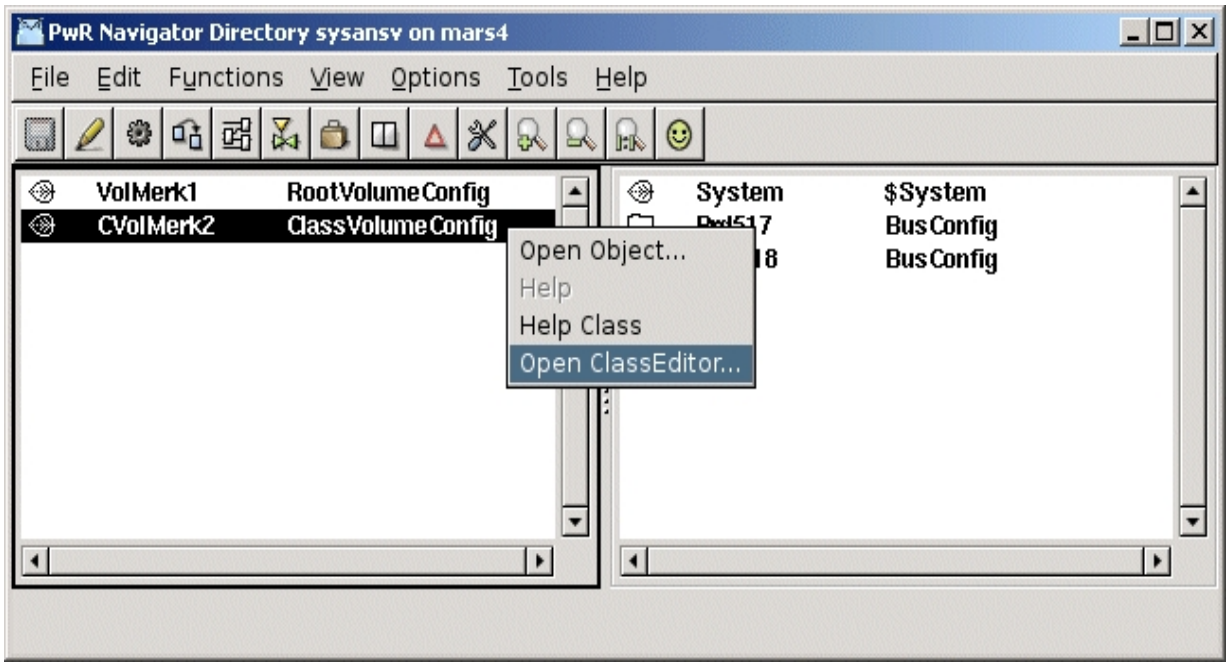


Registrazione del volume della classe in GlobalVolumeList

Il passo successivo e' configurare il volume della classe nel volume della directory del progetto, con un oggetto ClassVolumeConfig. Aprire il volume della directory con

```
$ pwrs
```

e creare un oggetto ClassVolumeConfig nella finestra di sinistra. L'oggetto dovrebbe avere lo stesso nome del classvolume. Dopo aver salvato e lasciato la modalita' di modifica, il classvolume puo' essere aperto facendo clic con il tasto destro sull'oggetto ClassVolumeConfig e attivando 'Open ClassEditor...'.



Configurazione del classvolume nel volume della directory

Ora viene aperto l'Editor di classi, in cui e' possibile creare oggetti classdefinition.

Entrando in modalita' di modifica, viene visualizzata una tavolozza con le classi di descrizione della classe e del tipo utilizzate per definire una classe o un tipo.

i inizia creando un oggetto di tipo $ClassHier al livello piu' alto. Questo otterra' automaticamente il nome "Class". Sotto gli oggetti $ClassHier, vengono creati oggetti $ClassDef per ogni classe che deve essere definita.

24.4.2 Classi di dati

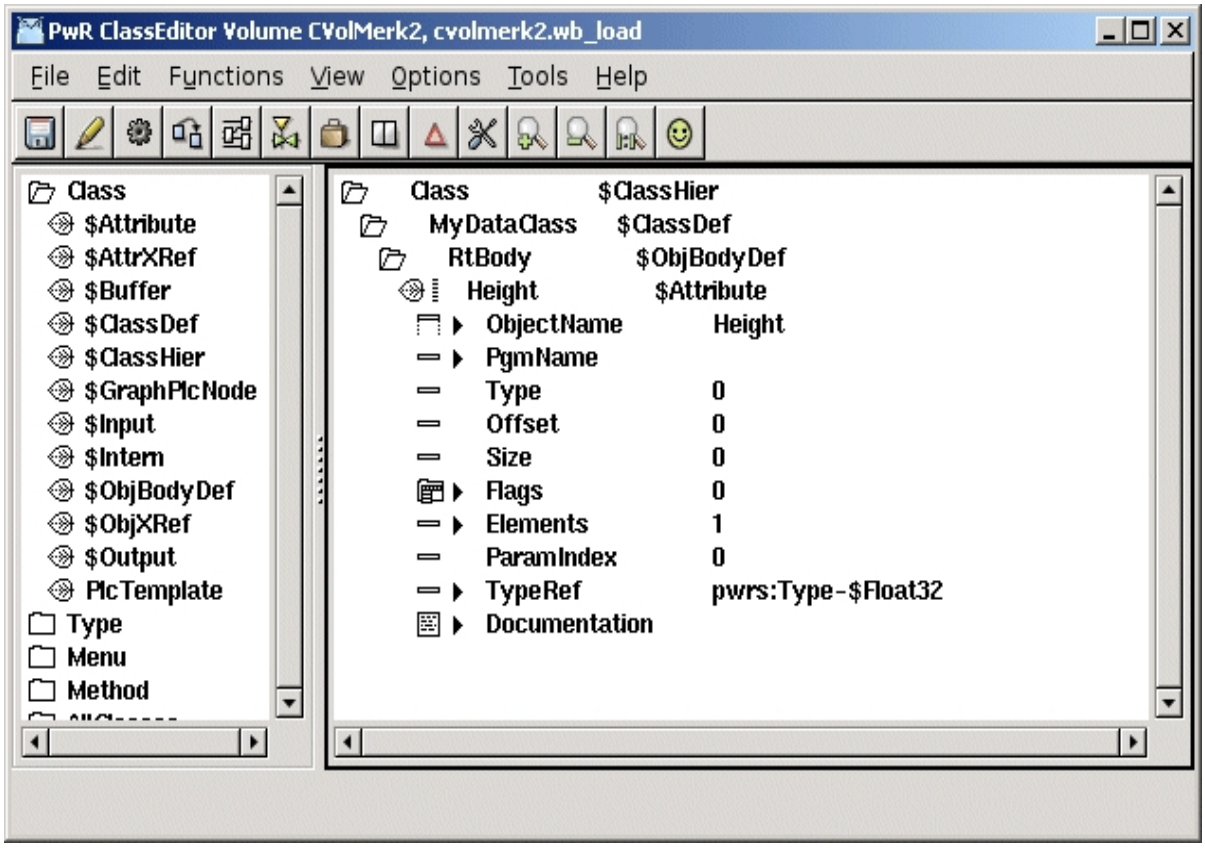
Le classi BData sono le classi piu' elementari e solitamente vengono utilizzate per memorizzare i dati. Le classi sono costituite da un RtBody con attributi.

Per creare una classe, inserisci un oggetto $ ClassDef sotto l'oggetto 'Class'.
Il nome dell'oggetto indica il nome della classe.

Sotto l'oggetto $ ClassDef si crea un oggetto $ ObjBodyDef che ottiene automaticamente il nome RtBody.

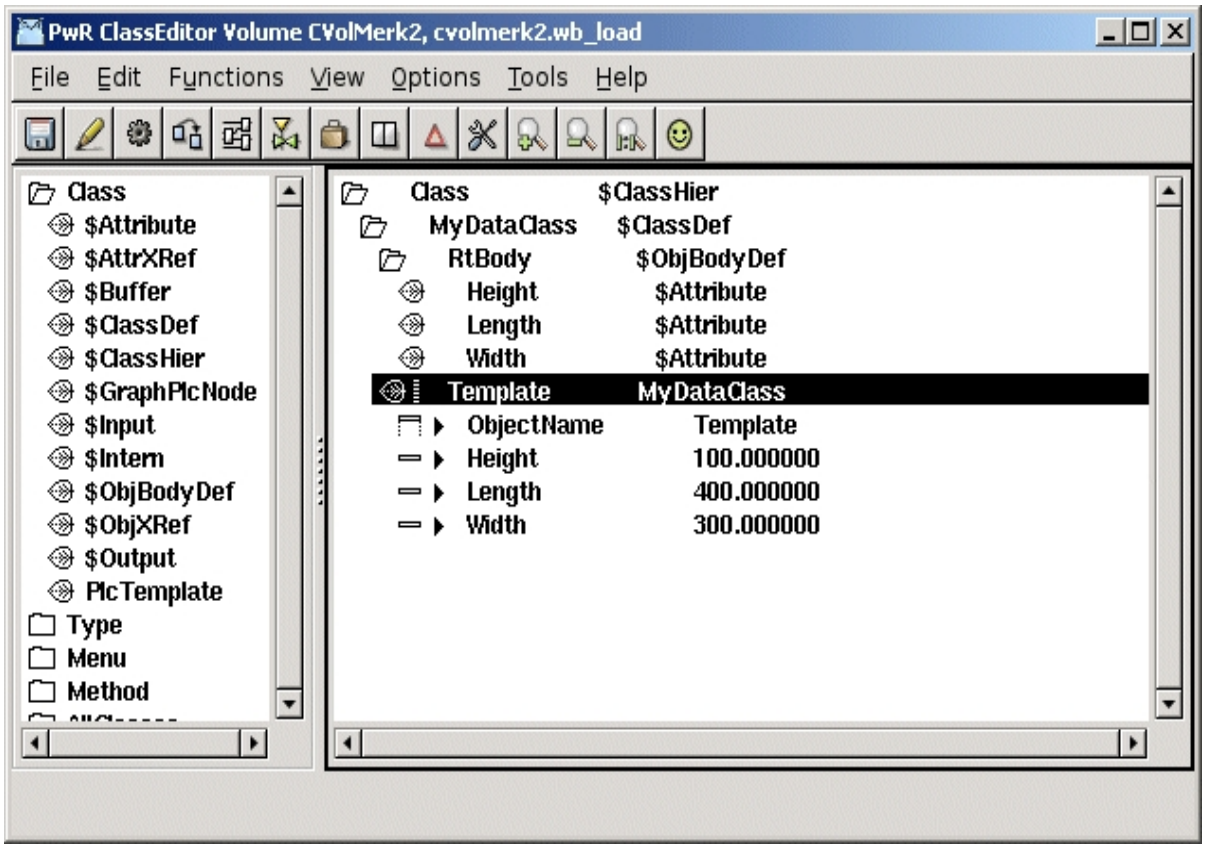
Sotto l'oggetto RtBody, viene creato un oggetto $Attribute, che definisce un attributo nella classe. Il nome dell'oggetto $Attribute indica il nome dell'attributo. Nell'oggetto dovresti dichiararlo nell'oggetto attributo:

- il tipo di attributo e' specificato in TypeRef, ad esempio un numero intero a 32 bit e' indicato con pwrs: Type- $ Int32, un float a 32 bit con pwrs: Type- $ Float32 e un booleano con pwrs: Type- $ Boolean. In realta' e' il nome completo di un oggetto di definizione del tipo che viene inserito. Vedere il Manuale di riferimento dell'oggetto, pwrs/Types, quali tipi sono definiti.
- se il nome dell'attributo contiene caratteri nazionali, in PgmName devi indicare un nome senza caratteri nazionali, in modo che sia accettato dal compilatore c.



Definizione di un attributo

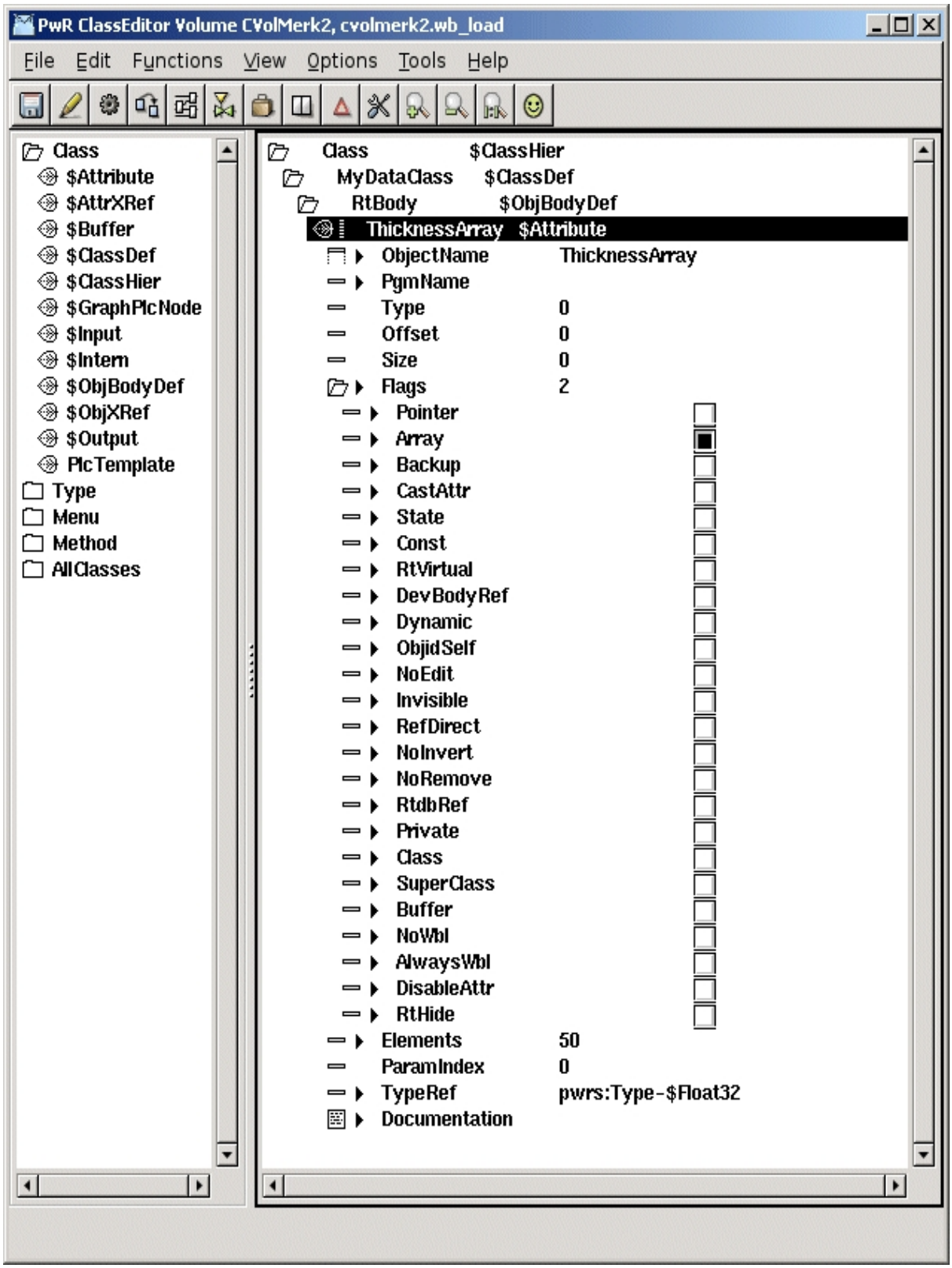
Quando si salva, un oggetto istanza della classe corrente con il nome Template, viene creato sotto l'oggetto $ClassDef. Qui puoi vedere il layout della classe e anche impostare i valori dei template per gli attributi. Quando vengono create altre istanze della classe, vengono create come una copia dell'oggetto Template.



Oggetto modello con valori predefiniti

Arrays

Un attributo array e' definito con un oggetto $Attribute, come altri attributi.
Qui si imposta il bit Array nei Flags e si specifica il numero di elementi della matrice in Elements.



Definizione di un attributo array con 50 elementi

Puntatori

Esistono due tipi di attributi puntatore

- puntatori relativi che possono essere utilizzati da diversi processi. Il valore e' impostato con la funzione gdh_StoreRtdbPointer () e convertito in un puntatore

assoluto con gdh_TranslateRtdbPointer (). Imposta il bit del puntatore in flag.
Nota! La dimensione dell'elemento puntato dal puntatore deve essere impostata in Size (in byte).

- puntatori assoluti. Questi possono essere impostati e utilizzati solo da un singolo processo. Imposta il bit del puntatore e il bit privato in flag.

Attribute objects

Oggetti attributi

Il termine oggetti attributo si riferisce ad attributi che sono descritti da una struttura di dati. Si utilizzano questi oggetti quando si desidera raccogliere dati in una mappa o quando la struttura di dati viene ripetuta e in questo caso si crea una matrice di oggetti di attributo.

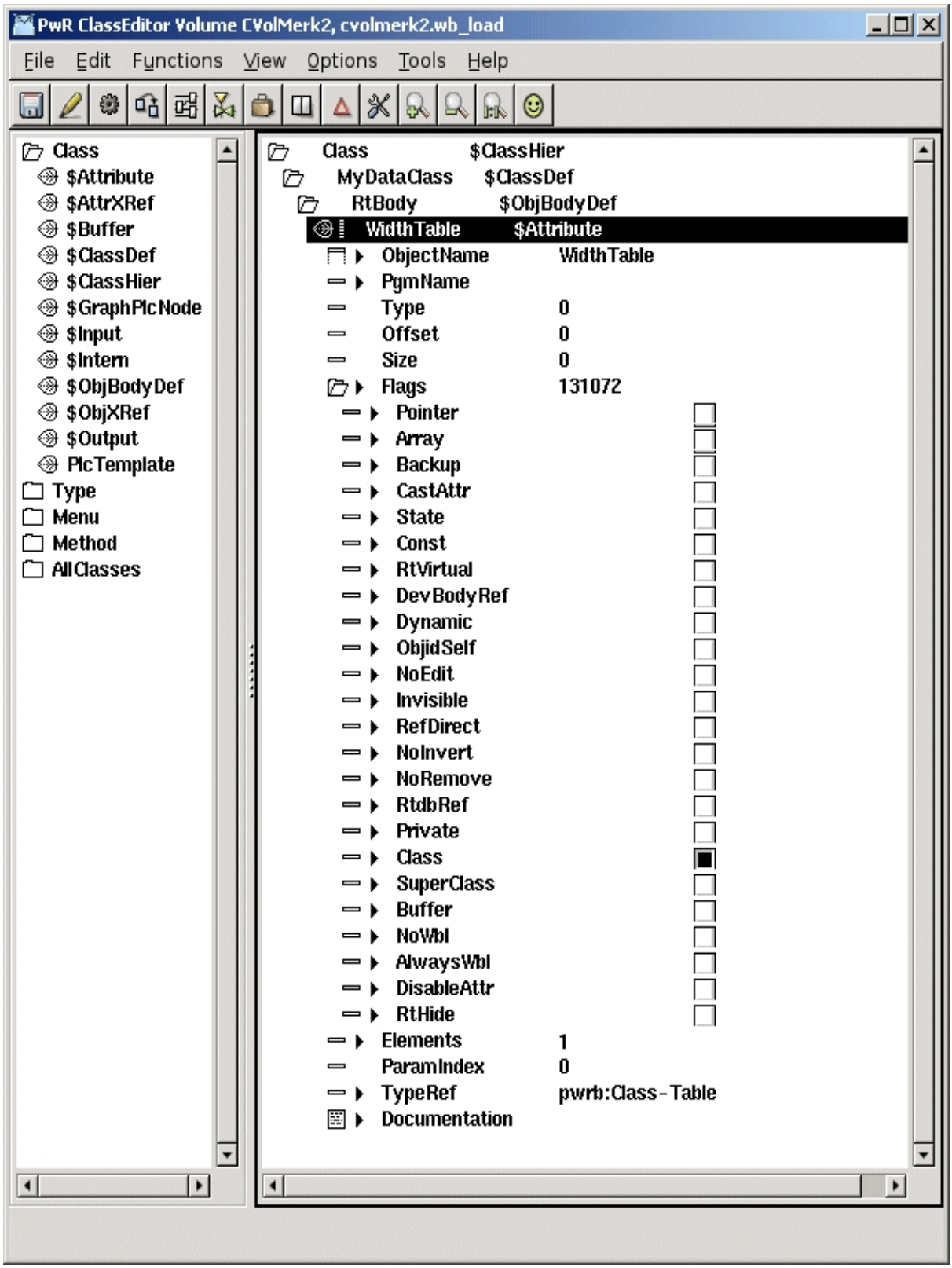
La struttura dati dell'attributo e' definita in una classe separata. La classe dovrebbe contenere solo un corpo runtime e non puo' avere un corpo di sviluppo.

L'oggetto attributo e' definito da un oggetto $Attribute. In TypeRef viene dichiarata la classe che descrive la struttura dati, e in Flags viene impostato il bit Class.

E' inoltre possibile creare un array, impostando il bit Array in Flags e il numero di elementi in Elements.

Gli oggetti attributi possono anche contenere attributi che sono oggetti attributo.
Il numero di livelli e' limitato a 20 e il nome dell'attributo totale e' limitato a 255 caratteri.

Un attributo in un oggetto attributo e' delimitato con punti, cioe' l'attributo Descrizione nell'oggetto attributo Pump nell'oggetto o, viene indicato con il nome 'o.Pump.Description'. Se Pump e' anche una matrice di pumpobjects, il nome dell'attributo Description nel primo oggetto pump e' 'o.Pump [0] .Description'.



Definizione di un oggetto attributo della classe Tabella

Sotto classi

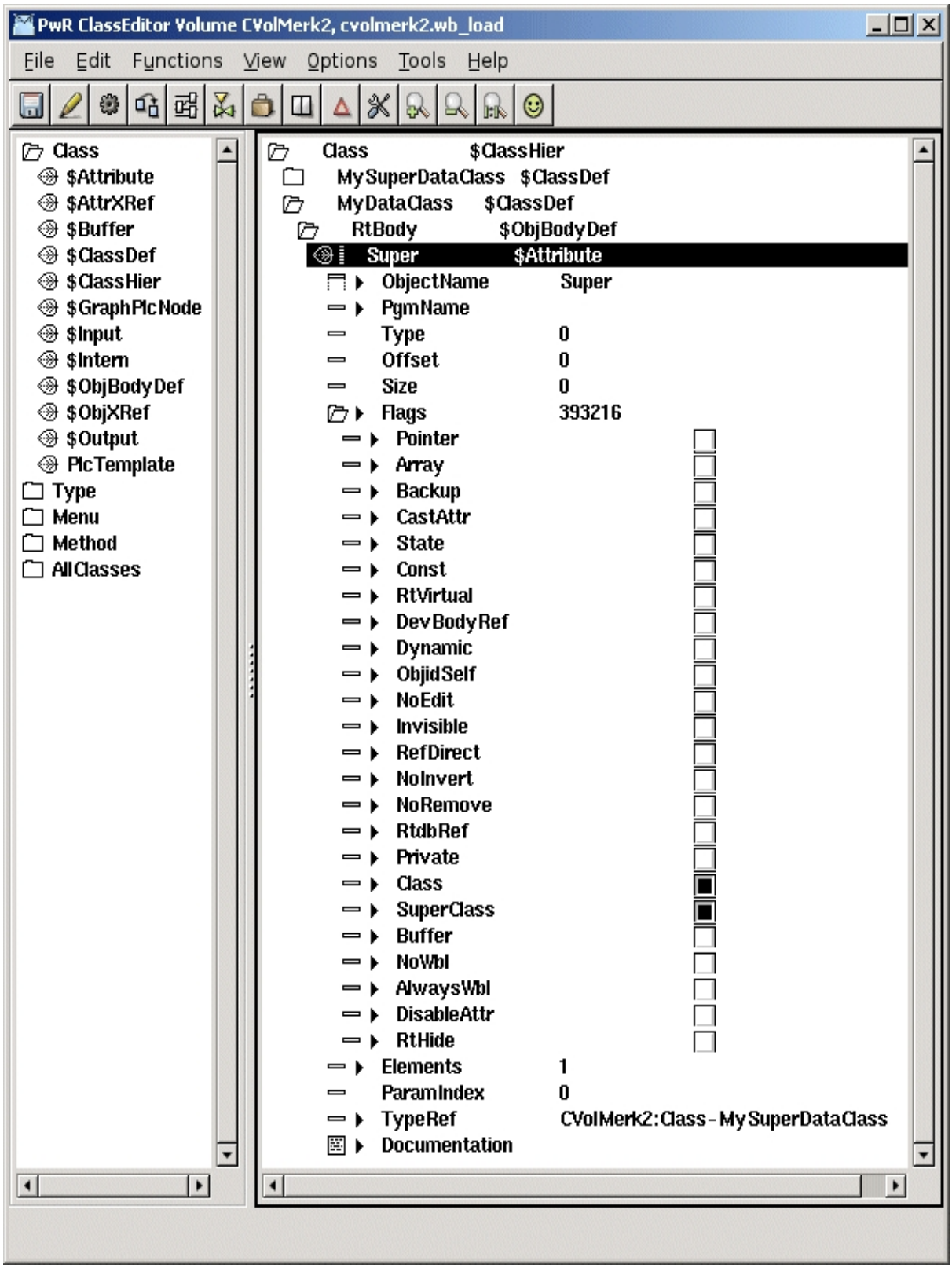
Puoi anche definire una classe come sottoclasse a un'altra classe. La sottoclasse ereditera'

attributi e metodi dall'altra classe, che e' chiamata la super classe.

Una sottoclasse viene definita nominando il primo oggetto $ Attribute nella classe a 'Super' e impostando i bit Class e SuperClass in Flags. La superclasse e' dichiarata in TypeRef.

Tutti gli attributi che esistono nella superclasse saranno anche presenti nella sottoclasse. La sottoclasse puo' anche avere attributi aggiuntivi che vengono definiti come al solito da oggetti $Attribute.

Una superclasse puo' contenere solo un corpo runtime, non un corpo di sviluppo.



Gli attributi Super rendono MyDataClass una sottoclasse di MySuperDataClass

24.4.3 Classi di oggetti funzione

Gli oggetti funzione vengono utilizzati nell'editor di plc per programmare il plc.

Un oggetto funzione e' descritto da una classe, solitamente un po' piu' complessa di una classe dati. Definisce, oltre alla struttura dati, il layout grafico con input e output e il codice che deve essere generato per il programma plc.

Il codice puo' essere definito mediante c-code o mediante programmazione grafica nell'editor del plc.

24.4.3.1 Funzione oggetto con codice c

La classe dell'oggetto funzione e' definita da un oggetto $ClassDef sotto l'oggetto 'Class'. Assegna un nome all'oggetto e attiva "Configure-CCodeFo" dal menu popup dell'oggetto. Ora vengono creati

- un oggetto RtBody.
- un oggetto DevBody con un oggetto PlcNode che definisce un buffer per le informazioni grafiche nelle istanze.
- un oggetto GraphPlcNode che contiene informazioni per la generazione di grafici e codice per la classe.

Il prossimo passo e' definire gli attributi della classe. Gli attributi sono suddivisi in input, attributi interni e output.

Inputs
Gli attributi di input definiscono i pin di input dell'oggetto funzione, cioe' i valori che vengono prelevati dai pin di uscita di altri oggetti funzione. Gli input sono definiti da oggetti $Input che sono posizionati sotto l'oggetto RtBody.

In TypeRef viene indicato il tipo di dati dell'ingresso. I tipi di dati validi per un input sono pwrs: Type-Float32, pwrs: Type-Int32 e pwrs: Type-String80.

In GraphName viene indicato il testo sul pin di input nell'oggetto funzione. Normalmente si usano da 2 a 4 caratteri, lettere maiuscole per i segnali analogici, minuscole per il digitale e maiuscole del primo carattere per altri tipi di segnale.

Un attributo di input in un oggetto istanza, contiene sia un puntatore all'output a cui e' connesso, sia un valore che puo' essere dichiarato. E' possibile scegliere se utilizzare il pin di input e collegare un'uscita o impostare un valore con una casella di controllo (utilizzata). Se si sceglie di non contrassegnare Usato, il pin di input non viene visualizzato nell'oggetto funzione. Nell'oggetto Template, e' possibile impostare i valori predefiniti per l'input, che verranno utilizzati quando l'input non e' connesso.

Attributi Interni
Gli attributi interni sono attributi che non sono input o output. Possono essere utilizzati per i valori calcolati che devono essere memorizzati nell'oggetto o per i valori utilizzati per configurare l'oggetto.

Tutti i tipi di dati comuni sono validi per gli attributi interni.

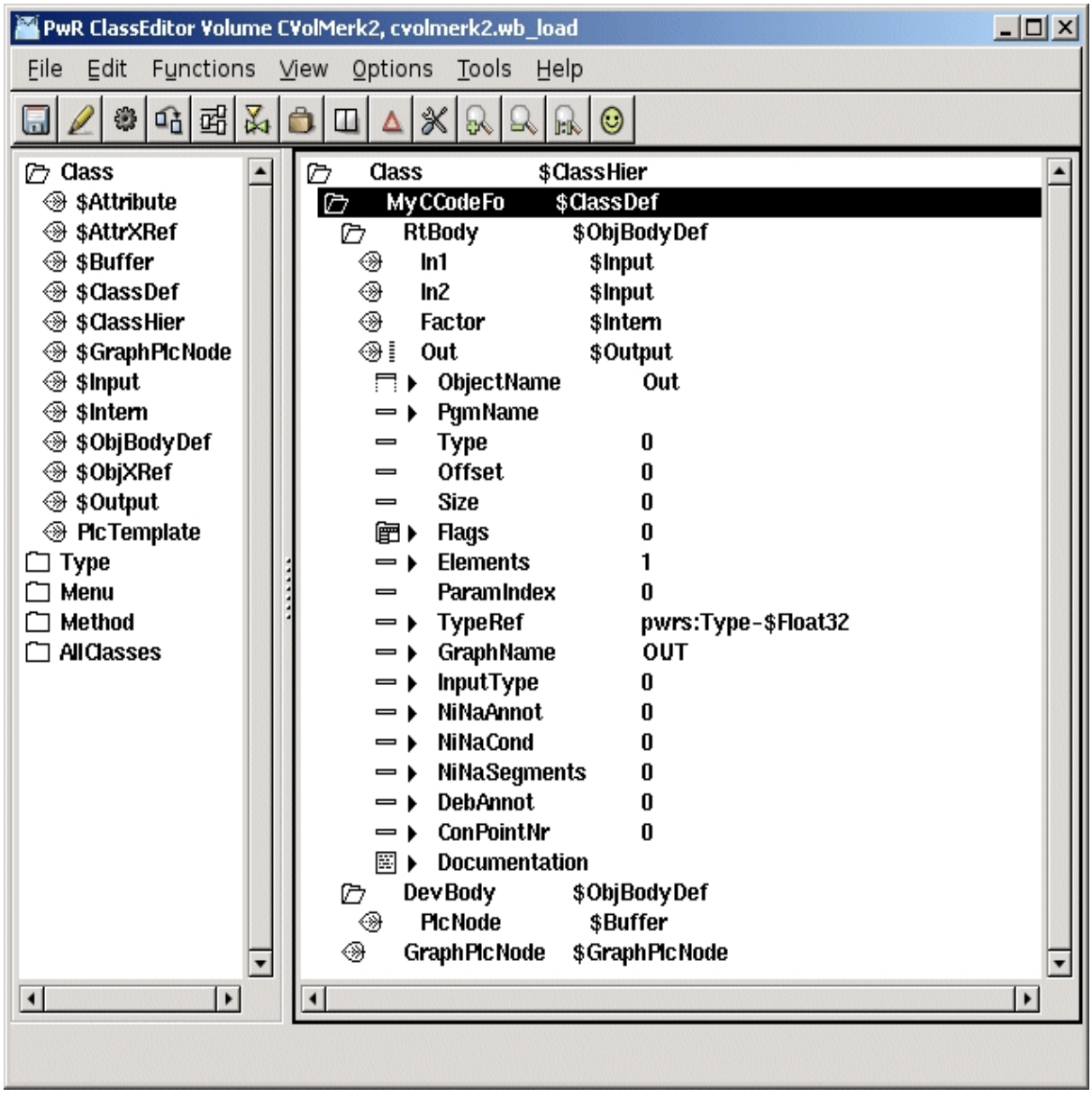
Outputs
Gli attributi di output definiscono i pin di uscita dell'oggetto funzione, cioe' i valori memorizzati nell'oggetto che possono essere recuperati dagli input di altri oggetti funzione. Gli output sono definiti da oggetti $Outputi che sono posizionati sotto l'oggetto RtBody.

Il tipo di dati per l'output e' dichiarato in TypeRef. Come per $Input i tipi Boolean, Float32, Int32 e String80 possono essere dichiarati, e in GraphName viene indicato il testo per i pin di output nell'oggetto function.

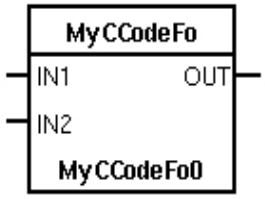
Nota !
$Input, $Intern e $Output devono essere posizionati in questo ordine sotto un RtBody:
$Input per primi, poi $Intern e poi $Output.

Valori di Default
I valori predefiniti degli attributi possono essere impostati nell'oggetto Modello (Template).

Se si desidera indicare quali input e output devono essere visualizzati come predefiniti, esiste una maschera nell'oggetto GraphPlcNode che controlla questo aspetto, la default_mask. I bit in default_mask[0] corrispondono agli attributi di input e bit in default_mask[1] agli attributi di output. Se il bit che corrisponde a uno specifico input o output e' impostato, questo verra' visualizzato come predefinito.



Oggetto funzione con due input, un attributo interno e un output



L'oggetto funzione per la classe

Codice
Quando viene creato il volume della classe, viene generato un file h con una struttura c per la classe. Il nome della struttura e'

```
pwr_sClass_'StructName'
```

dove StructName viene prelevato dall'attributo StructName in RtBody. Come impostazione predefinita, e' uguale al nome della classe, ma, ad esempio, se il nome della classe contiene caratteri nazionali, e' possibile specificare un altro nome.

Di seguito viene visualizzato un esempio della struttura per la classe MyFo. MyFo contiene due input In1 e In2, un fattore di attributo interno e un output Out, tutti di tipo Float32.

```
typedef struct {
  pwr_tFloat32           *In1P;
  pwr_tFloat32           In1;
  pwr_tFloat32           *In2P;
  pwr_tFloat32           In2;
  pwr_tFloat32           Factor;
  pwr_tFloat32           Out;
} pwr_sClass_MyFo;
```

Si noti che ogni input e' costituito da due elementi, un puntatore con il suffisso 'P' e un elemento a cui e' possibile assegnare un valore se l'input non e' connesso. Se l'input e' connesso, l'elemento puntatore puntera' all'output a cui e' collegato, altrimenti puntera' all'elemento value. Pertanto, nel codice c, e' necessario utilizzare l'elemento puntatore per recuperare il valore dell'input.

Il codice per la classe e' una funzione con questo aspetto

```
void 'StructName'_exec( plc_sThread *tp,
                        pwr_sClass_'StructName' *o) {
}
```

Nel codice, i dati vengono recuperati dagli input e i valori calcolati vengono inseriti negli output. Anche gli attributi interni possono essere utilizzati per memorizzare informazioni alla scansione successiva o per recuperare i dati di configurazione.

Nell'esempio di codice seguente In1 e In2 sono input, Factor e' un attributo interno e Out un'output.

```
  o->Out = o->Factor * (*o->In1P + *o->In2P);
```

Si noti che l'elemento puntatore per gli ingressi In1 e In2 viene utilizzato nel codice.

Dovresti anche aggiungere la dichiarazione del prototipo della funzione exec in ra_plc_user.h

```
void 'StructName'_exec( plc_sThread *tp,
                        pwr_sClass_'StructName' *o);
```

Il modulo del codice c e' compilato e collegato al programma plc. Cio' richiede che un file di collegamento venga inserito nella directory $pwrp_exe. Il file e' chiamato plc_'nodename '_' busnumber '_' plcname'.opt, ad es. plc_mynode_0999_plc.opt. Il contenuto del file viene inserito nel linker, ld, e qui si aggiungono i moduli del codice plc. Nell'esempio seguente questi moduli sono collocati nell'archivio $ pwrp_lib / libpwrp.a

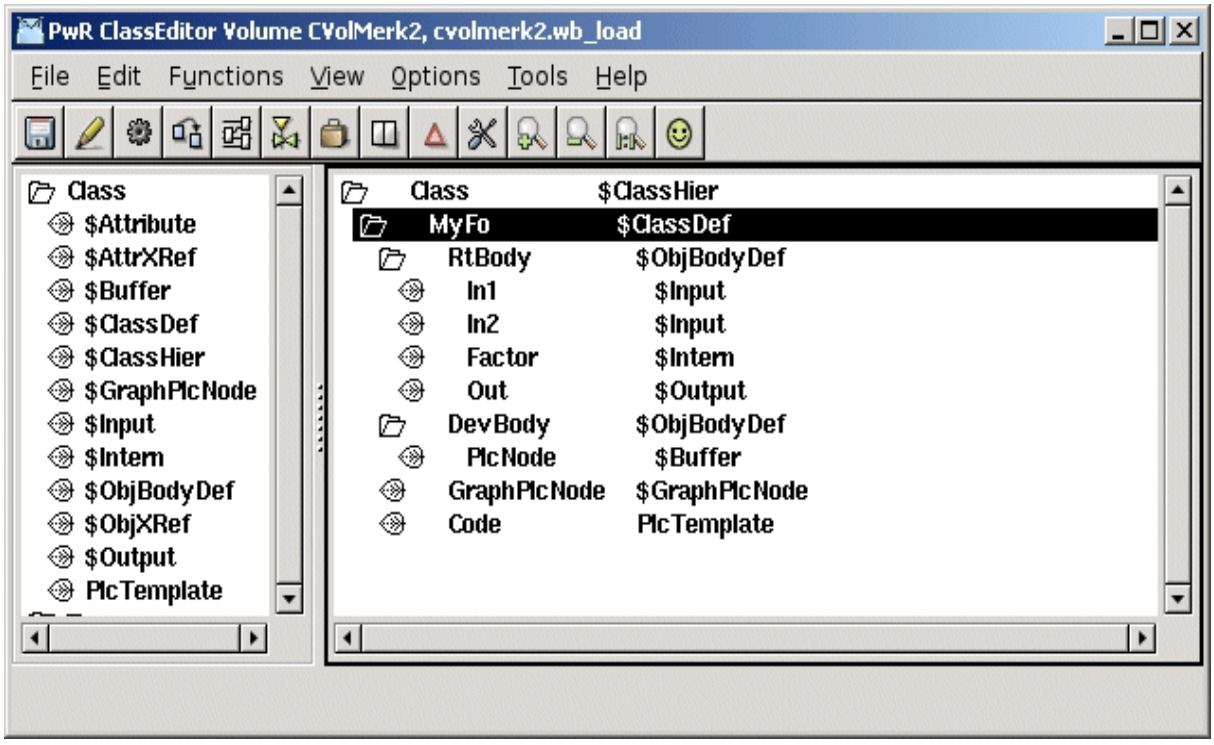
```
$pwr_obj/rt_io_user.o -lpwrp -lpwr_rt -lpwr_usbio_dummy -lpwr_usb_dummy
-lpwr_pnak_dummy -lpwr_cifx_dummy -lpwr_nodave_dummy -lpwr_epl_dummy
```

24.4.3.2 Oggetto funzione con codice plc

Un oggetto funzione, in cui il codice e' scritto in plc-code nell'editor del plc, e' definito in modo simile all'oggetto function con il c-code come prima riportato.

La classe functionobject e' definita da un oggetto $ClassDef sotto l'oggetto 'Class'. Assegna un nome all'oggetto e attiva Configure-Fo nel menu popup dell'oggetto. Ora, oltre agli oggetti creati per il functionobject c-code, viene creato anche un oggetto Code della classe PlcTemplate. Questo oggetto puo' essere aperto con l'editor del plc per definire il codice della classe.

Gli input, gli attributi interni e gli output nell'oggetto funzione sono definiti allo stesso modo degli oggetti funzione c-code, con gli attributi $Input, $Intern e $Output.



Definizione di un oggetto funzione con codice plc.

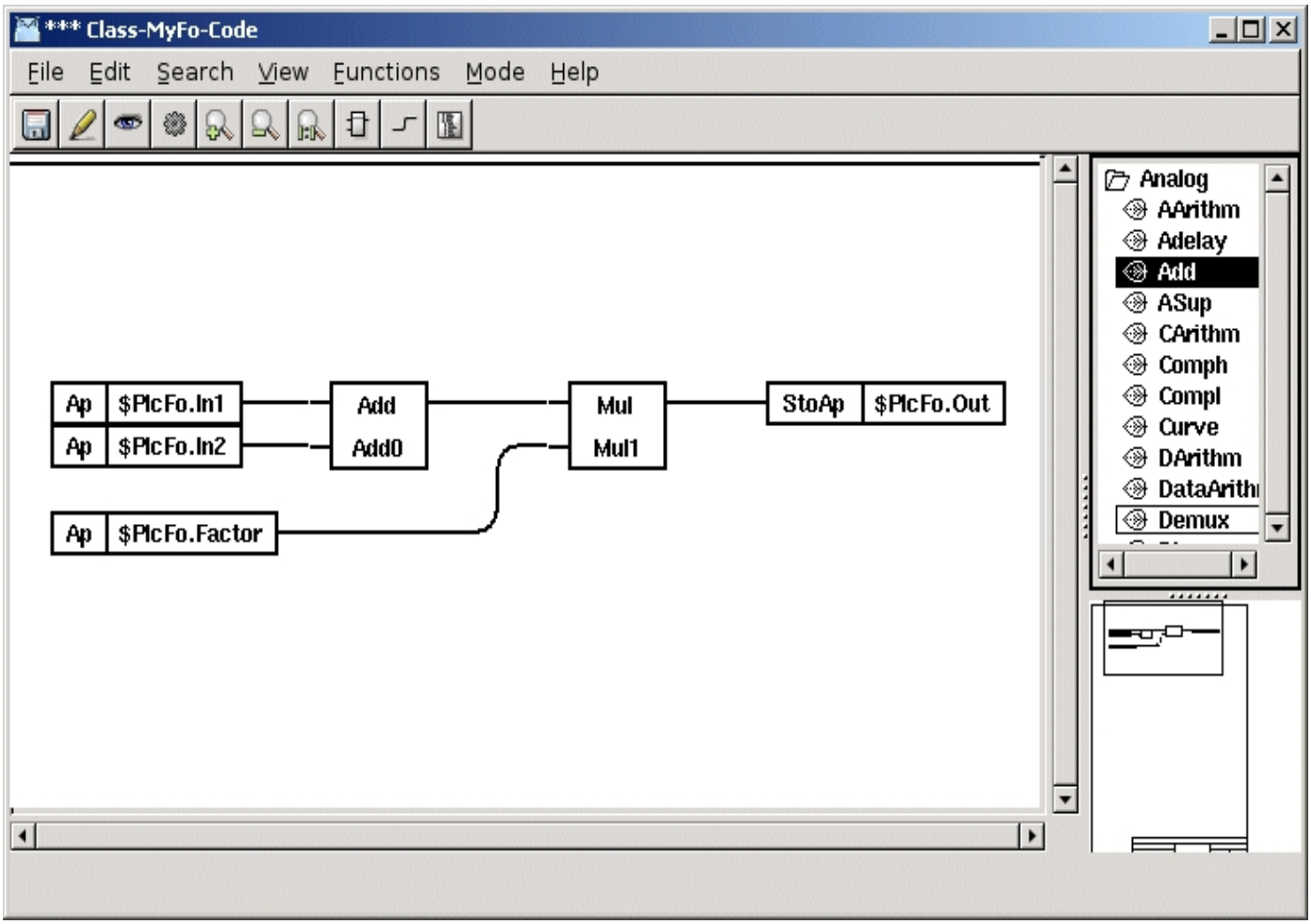
Codice

Attivando 'Open Program...' nel menu popup dell'oggetto Codice, viene aperto l'editor del plc. Qui il codice e' scritto con la programmazione degli oggetti funzione. Il codice viene creato come un normale programma, ma in questo caso bisogna anche recuperare i valori dall'attributo

input e intern, e per impostare i valori sugli output.

I valori di input, attributi interni e anche output, vengono recuperati nel codice con oggetti GetDp, GetIp, GetAp o GetSp. Connetti gli oggetti agli attributi nella classe selezionando l'attributo nell'oggetto Template per la classe e attiva il metodo 'Connect' per l'oggetto Get. Un riferimento simbolico $PlcFo viene inserito nell'oggetto Get. Questo sara' in seguito scambiato con un riferimento all'istanza corrente, quando il codice per l'istanza e' compilato.

I valori calcolati sono memorizzati nelle uscite o negli attributi interni con StoDp, StoIp ecc. Questi sono collegati agli attributi allo stesso modo degli input, selezionando gli attributi nell'oggetto Template e attivando 'Connect'.



Esempio di codice plc per un oggetto funzione

Il codice modello nell'oggetto Codice non deve essere compilato o creato. Quando un oggetto istanza viene compilato per la prima volta, il codice del modello viene copiato nell'istanza.

Quando il codice del template viene modificato, il codice delle istanze verra' aggiornato alla successiva compilazione (il volume contenente le istanze deve essere prima aggiornato con UpdateClasses).

24.4.4 Classi I/O

Gli oggetti I / O sono gli oggetti gestiti dalla gestione I/O in Proview. Sono suddivisi in oggetti Agent, Rack, Card e Channel. Quando si adattano nuovi sistemi I/O a Proview, e' necessario creare nuove classi di tipi Agent, Rack e Card. Gli oggetti I/O sono definiti

da un oggetto $ClassDef in cui il bit IoAgent, IoRack o IoCard e' impostato in Flag.

Una guida piu' dettagliata su come creare oggetti I/O si trova nella Guida al sistema I/O.

24.4.5 Componenti

Un componente e' un oggetto o un numero di oggetti che gestisce un componente dell'impianto. Potrebbe essere una valvola, un motore, un convertitore di frequenza, ecc. L'idea alla base del concetto di componente e' che creando un oggetto (o un numero di oggetti) si ottengono tutte la funzionalita' necessarie per controllare il componente: un oggetto contenente dati e segnali , un oggetto funzione con codice per controllare il componente, un oggetto grafico per HMI, un oggetto di simulazione, oggetti I/O per configurare la comunicazione del bus, ecc.

Un componente puo' includere le seguenti parti
- un oggetto principale
- un oggetto funzione
- un oggetto di simulazione
- uno o piu' oggetti bus I/O
- grafica dell'oggetto per l'oggetto principale
- grafica dell'oggetto per l'oggetto di simulazione
- simbolo grafico per l'oggetto principale

24.4.5.1 Oggetto principale

L'oggetto principale contiene tutti i dati necessari per configurare e fare calcoli. L'oggetto viene posizionato nella gerarchia dell'impianto, come singolo oggetto o come parte di un aggregato.

Spesso la classe BaseComponent:Component e' usata come super classe in una classe di componenti. Contiene un numero di attributi come Descrizione, Specifica, DataSheet ecc.

Tutti i segnali di input e output che sono collegati al componente devono essere collocati nell'oggetto principale. Gli oggetti Di, Ii, Ai, Do, Io, Ao o Co vengono inseriti come oggetti attributo. Quando si creano istanze del componente, i segnali devono essere collegati agli oggetti del canale. Per profibus, ad esempio, e' possibile creare un oggetto modulo, che contiene i canali, e preconnettere i segnali nell'oggetto principale a questi canali. Per ogni istanza, non e' quindi necessario connettere singolarmente ogni canale, ma e' possibile creare una singola connessione tra l'oggetto principale e l'oggetto modulo.

Attributi speciali
PlcConnect
Se c'e' un codice che deve essere creato dal programma plc, si crea un oggetto funzione per la classe. Questo deve essere connesso all'oggetto principale, e questa connessione e' memorizzata in un attributo con il nome 'PlcConnect' di tipo pwrs: Type-$AttrRef.

SimConnect
Se c'e' un oggetto di simulazione, questo e' collegato all'oggetto principale da un attributo 'SimConnect' di tipo pwrs:Type-AttrRef.

IoConnect
Se esiste un oggetto modulo I / O, questo e' collegato con un attributo 'IoConnect' di tipo pwrs:Type-AttrRef. L'attributo e' gestito dal metodo IoConnect.

IoStatus
Se si desidera recuperare lo stato dall'oggetto modulo I/O, si crea l'attributo 'IoStatus'

di tipo pwrs:Type-$Status e si imposta il bit Pointer in Flags. Devi anche impostare 4 in Size (per i puntatori relativi la dimensione di cio' a cui punta il puntatore, deve essere specificata in Size).

All'attributo verra' assegnato un puntatore all'attributo Status del modulo I/O in runtime quando viene inizializzata la gestione I/O. L'attributo Status e' di tipo Status e puo' essere visualizzato ad esempio in un oggetto grafico con il tipo dinamico StatusColor.
Se si desidera utilizzare IoStatus nel codice plc per l'oggetto, e' necessario considerare che l'attributo e' un puntatore e recuperare il valore con GetIpPtr.

Reset Sequenza
E' possibile utilizzare le sequenze Grafcet in un componente. Una differenza rispetto a una sequenza ordinaria e' che l'oggetto di reset dovrebbe essere definito come un attributo della classe Dv nell'oggetto principale, con il nome "SequenceReset". SubSteps non puo' essere utilizzato nella sequenza.

GraphConfiguration
GraphConfiguration e' di tipo Enum e viene utilizzato per decidere quale oggetto grafico deve essere aperto per l'istanza corrente. E' usato dal metodo 'ConfigureComponent' (vedi sotto).

DisableAttr
La funzione DisableAttr consente di disabilitare un attributo in un'istanza. Se un attributo e' disabilitato, non sara' visualizzato nel navigatore o nell'editor di oggetti. Se l'attributo disabilitato e' un segnale, verra' ignorato dalla gestione I/O.

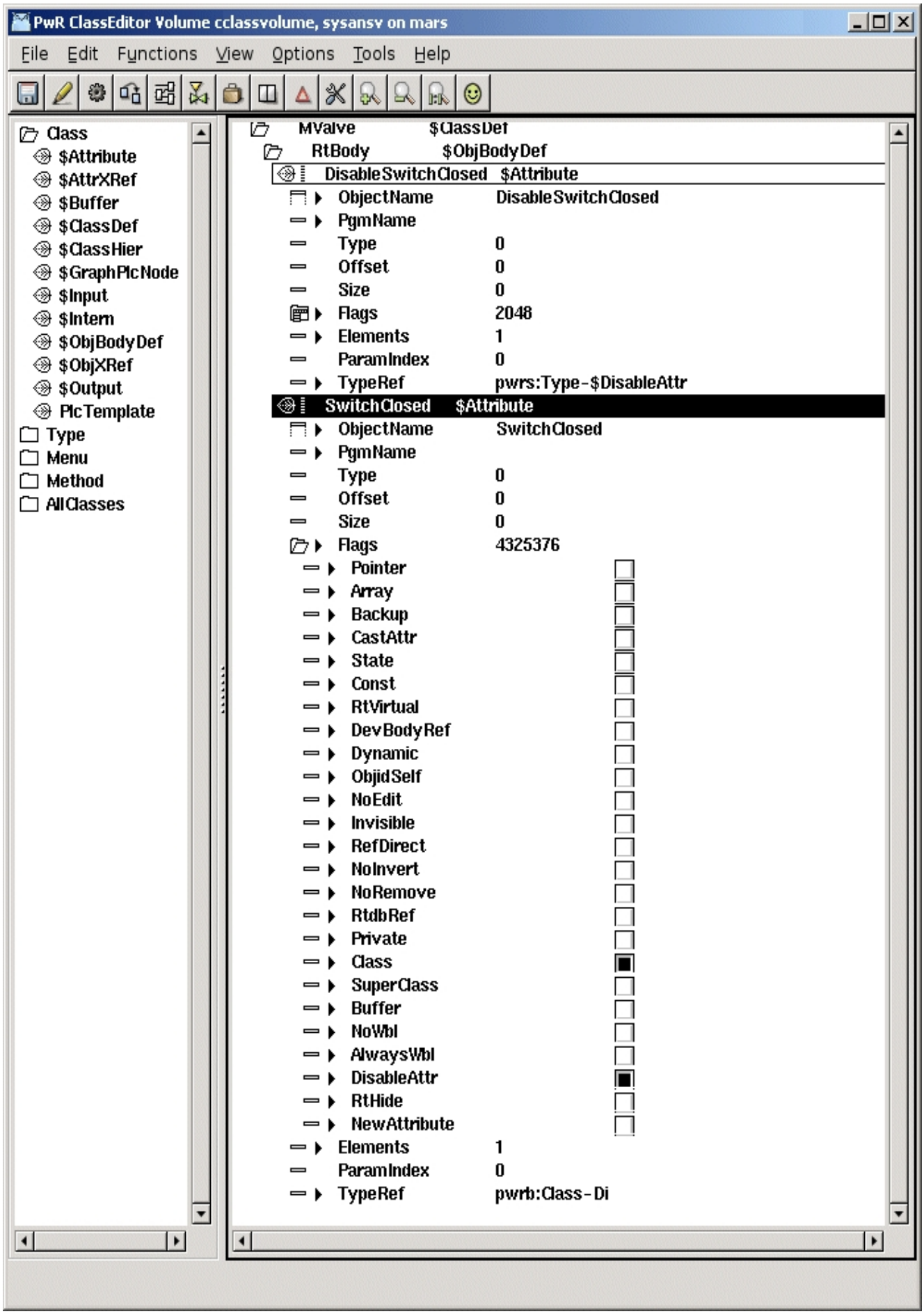
La funzione di disabilitazione viene utilizzata per componenti che possono esistere in diverse configurazioni.
Ad esempio, una valvola a solenoide puo' avere un sensore che indica quando la valvola e' aperta e uno che indica quando la valvola e' chiusa. In totale esistono sono quattro possibili configurazioni per l'elettrovalvola:

- senza sensori
- con sesore di posizione aperta
- con sesore di posizione chiusa
- con entrambi i sensori di posizione aperta e chiusa

E' possibile creare quattro diverse classi di elettrovalvole, ma si verifichera' un problema durante la creazione di aggregati degli oggetti della valvola. Un aggregato, contenente un oggetto valvola, deve anche esistere in quattro varianti e se l'aggregato contiene due oggetti valvola, ci devono essere 16 varianti. Usando la funzione DisableAttr sugli attributi switch possiamo creare una classe di elettrovalvola che copre tutte e quattro le configurazioni e puo' anche essere utilizzata in classi aggregate.

DisableAttr per un attributo viene configurato nel modo seguente.
- il bit DisableAttr in Flags e' impostato per l'attributo.
- prima dell'attributo, viene inserito un attributo di tipo pwrs:Type-$DisableAttr, con lo stesso nome dell'attributo, ma con il prefisso 'Disable'. Il bit Invisible in Flags deve essere impostato per l'attributo DisableAttr.

Esempio
Nella classe di elettrovalvole di cui sopra, l'interruttore chiuso e' rappresentato dall'attributo SwitchClosed che e' un segnale digitale di tipo pwrb:Class-Di. Immediatamente sopra l'attributo viene posizionato un attributo con il nome 'DisableSwitchClosed' di tipo pwrs:Type-$DisableAttr. Per questo attributo viene impostato il bit Invisibile in Flags e per l'attributo SwitchClosed viene impostato il bit DisableAttr in Flags.


PwR ClassEditor Volume cclassvolume, sysansv on mars
File Edit Functions View Options Tools Help
Class
$Attribute
$AttrXRef
$Buffer
$ClassDef
$ClassHier
$GraphPlcNode
$Input
$Intern
$ObjBodyDef
$ObjXRef
$Output
PlcTemplate
Type
Menu
Method
AllClasses
MValve $ClassDef
RtBody $ObjBodyDef
DisableSwitchClosed $Attribute
ObjectName DisableSwitchClosed
PgmName
Type 0
Offset 0
Size 0
Flags 2048
Elements 1
ParamIndex 0
TypeRef pwrs:Type-$DisableAttr
SwitchClosed $Attribute
ObjectName SwitchClosed
PgmName
Type 0
Offset 0
Size 0
Flags 4325376
Pointer
Array
Backup
CastAttr
State
Const
RtVirtual
DevBodyRef
Dynamic
ObjidSelf
NoEdit
Invisible
RefDirect
NoInvert
NoRemove
RtdbRef
Private
Class
SuperClass
Buffer
NoWbl
AlwaysWbl
DisableAttr
RtHide
NewAttribute
Elements 1
ParamIndex 0
TypeRef pwrb:Class-Di

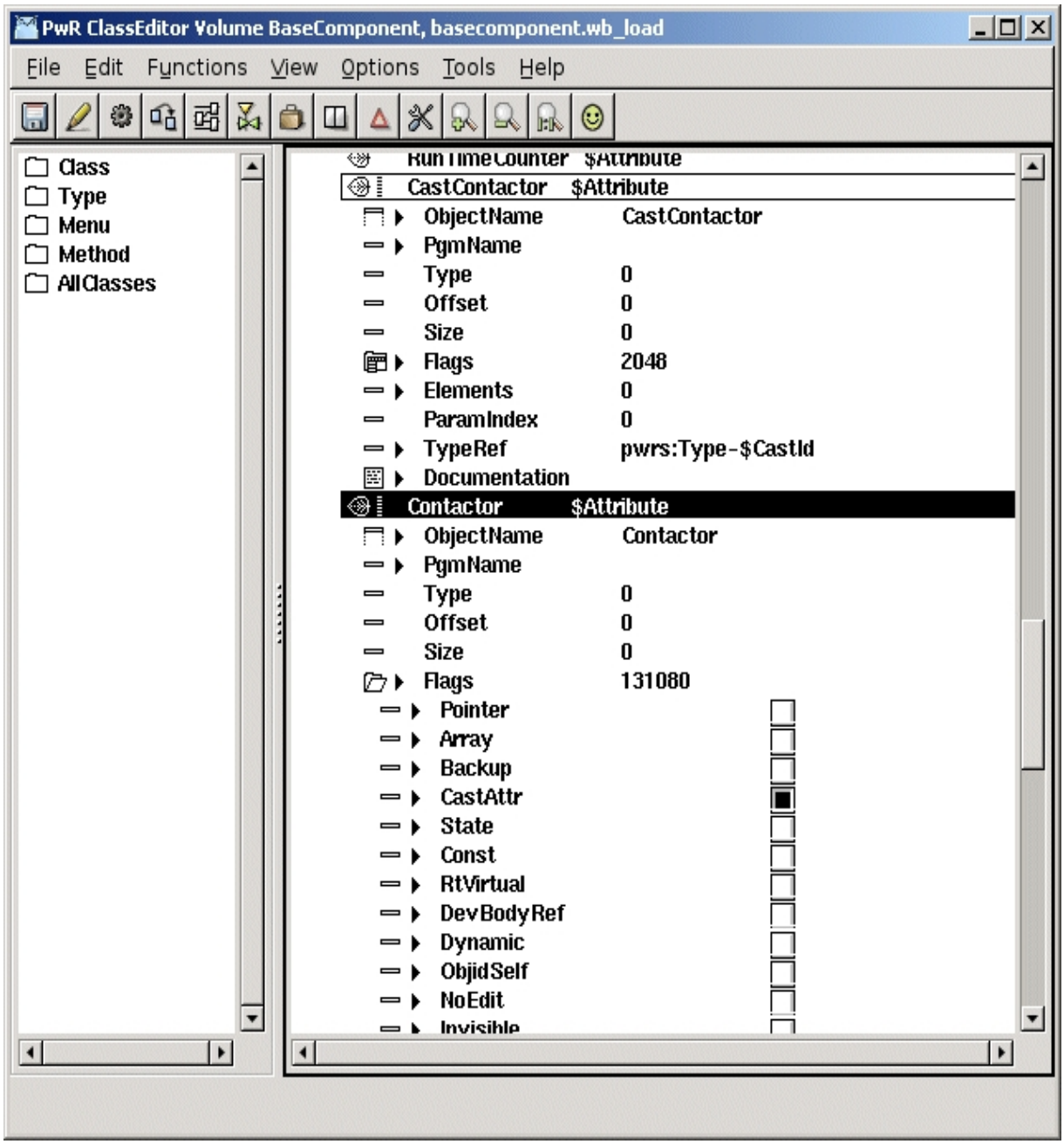
Attributo con funzione disabilita

Cast

Le classi di componenti sono spesso costruite in un modo relativamente flessibile per minimizzare il numero di varianti. Spesso si crea una classe di base che utilizza la funzione DisableAttr per essere in grado di coprire un numero di diverse configurazioni. Nell'esempio sopra una classe dell'elettrovalvola puo' coprire quattro diverse configurazioni impostando DisableAttr sui segnali di posizione. Si possono anche creare sottoclassi adattate a valvole specifiche. Ad esempio, un Durholt 100.103 non contiene alcun interruttore e viene creata una sottoclasse in cui entrambi gli switch sono disabilitati nell'oggetto Template.
E' inoltre possibile impostare altri adattamenti nell'oggetto Modello come il collegamento ad un foglio dati.
Il risultato e' una sottoclasse che puo' essere utilizzata per le valvole Durholt senza alcuna configurazione per ogni istanza.

Se ora costruiamo un aggregato generale, contenente una valvola a solenoide, e vogliamo essere in grado di utilizzare le sottoclassi esistenti per l'elettrovalvola, usiamo la funzione Cast. Con la funzione Cast, un oggetto attributo puo' essere castato come sottoclasse della classe originale, dato che la sottoclasse ha la stessa dimensione. Quando viene castato un oggetto attributo, i valori predefiniti e quindi le configurazioni vengono prelevati dalla sottoclasse. Anche il nome della classe e i metodi sono recuperati dalla sottoclasse.

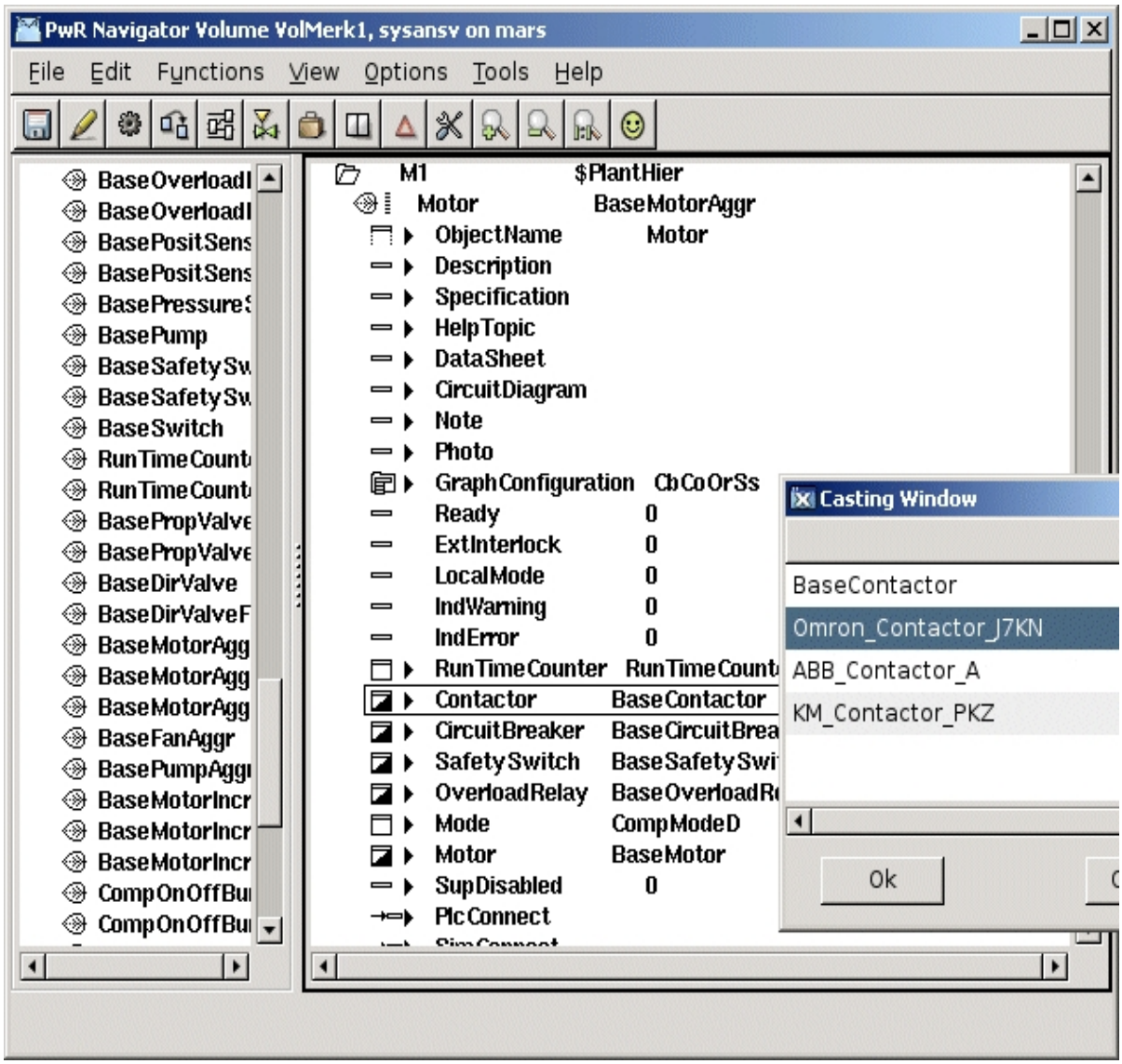
La funzione di cast per un attributo viene inserita nel seguente modo:
- Il bit CastAttr in Flags e' impostato per l'attributo.
- Prima dell'attributo,viene inserito un attributo di tipo pwrs:Type-$CastId con lo stesso nome dell'attributo, ma con il prefisso 'Cast'. Il bit Invisible in Flags deve essere impostato per l'attributo cast.



Contattore con attributo cast

Il casting di un'istanza viene eseguito attivando il metodo "Cast" nel menu popup dell'attributo. Viene visualizzata una lista con la classe di base e tutte le sottoclassi, in cui e' possibile selezionare una classe di cast appropriata.

Se un attributo ha entrambi gli attributi cast e disable, l'attributo cast deve essere posizionato prima dell'attributo disable.



Casting di un'istanza

Metodi

Metodo ConfigureComponent

Spesso ci sono diverse varianti di un componente. Nell'esempio con l'elettrovalvola di cui sopra, sono state trovate quattro diverse varianti dipendenti dalla configurazione degli interruttori. Per semplificare la configurazione degli utenti del componente, e' possibile definire il metodo 'ConfigureComponent'.

Il metodo ConfigureComponent consente di impostare Disabilita su uno o un numero di attributi da un'alternativa di menu nel menu popup e di selezionare un oggetto grafico adattato alla configurazione corrente.

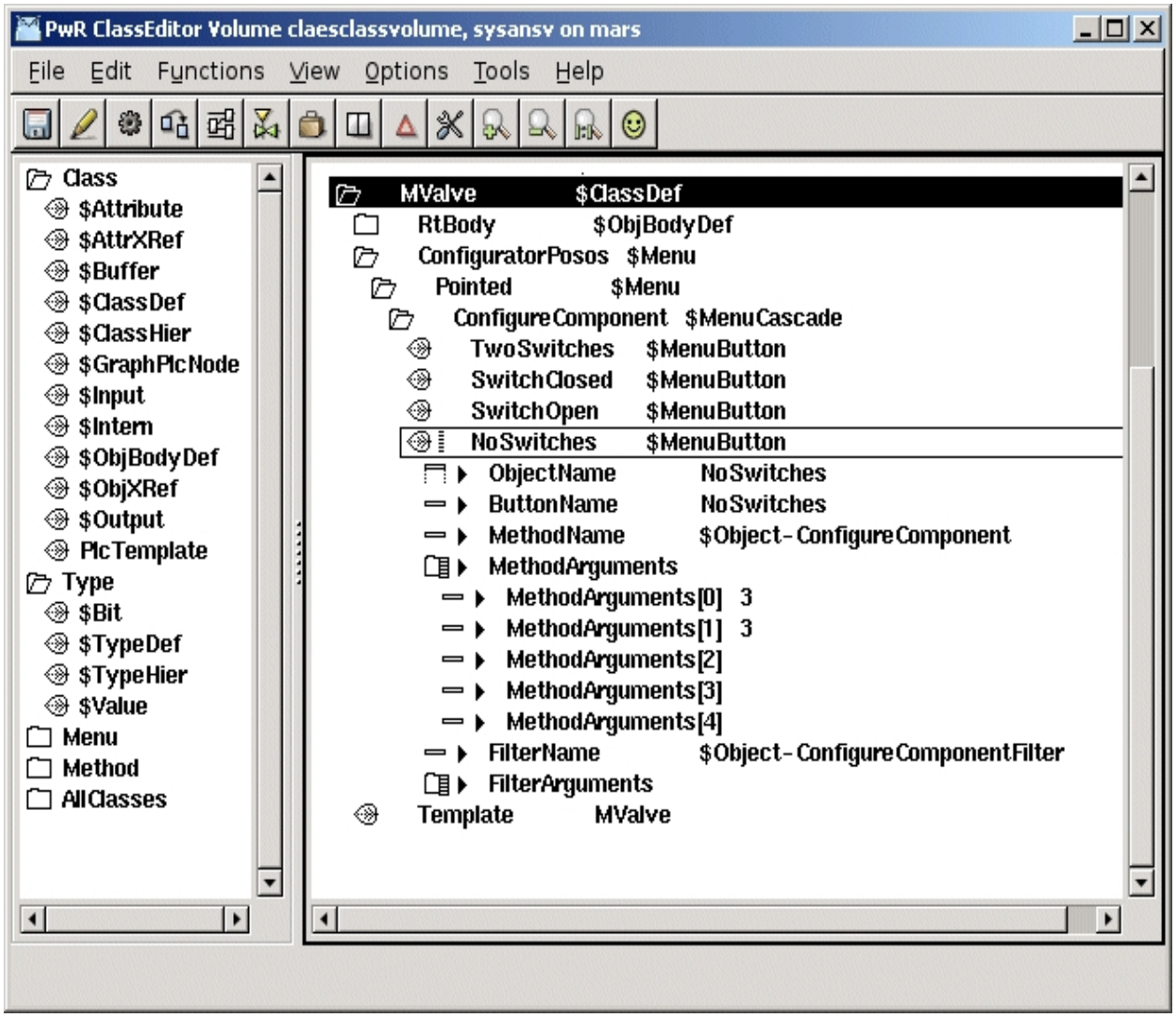
Menu

Le alternative di menu per ConfigureComponent sono definite dagli oggetti di menu. Sotto l'oggetto $ClassDef, viene posizionato un oggetto $Menu con nome 'ConfiguratorPosos', che rende il menu visibile in modalita' di modifica quando l'oggetto viene puntato e selezionato. Sotto questo, viene posizionato ancora un altro oggetto $Menu con il nome 'Pointed', e sotto questo un oggetto $MenuCascade con il nome 'ConfigureComponent'. L'attributo ButtonName e' impostato come ConfigureComponent per questo oggetto.

Al di sotto di questo, viene posizionato finalmente un oggetto $MenuButton per ogni alternativa di configurazione.
Il nome e' preferibilmente impostato sul nome dell'alternativa di configurazione e viene anche inserito nell'attributo ButtonName. Nell'attributo MethodName viene inserito '$ Object-ConfigureComponent' e nell'attributo FilterName '$ Object-ConfigureComponentFilter'. Dovresti anche inserire argomenti nel metodo in MethodArguments. MethodArguments[0] contiene una maschera di bit, che decide quali attributi verranno disabilitati nel menu corrente delle alternative. Ogni attributo, che e' possibile disabilitare, e' rappresentato da un bit e l'ordine bit corrisponde all'ordine dell'attributo nell'oggetto. MethodArguments[1] contiene la rappresentazione grafica, vedi sotto.

Se guardiamo all'elettrovalvola, abbiamo due attributi che possono essere disabilitati, SwitchClosed e SwitchOpen. Nella maschera di bit in MethodArguments[0] SwitchClosed corrisponde al primo bit e SwitchOpen al secondo, vale a dire se il primo bit e' impostato, SwitchClosed sara' disabilitato e se il secondo bit e' impostato, SwitchOpen e' disabilitato.
Le quattro alternative di configurazione TwoSwitches, SwitchClosed, SwitchOpen e NoSwitches corrispondono alle seguenti maschere

```
TwoSwitches   0   (entrambi SwitchOpen e SwitchClosed sono presenti)
SwitchClosed  2   (SwitchOpen e' disabilitato)
SwitchOpen    1   (SwitchClosed e' disabilitato)
NoSwitches    3   (entrambi SwitchOpen e SwitchClosed sono disabilitati)
```



Configurazione degli attributi del componente
Se si disattiva un attributo che e' un componente che contiene segnali, anche i segnali nel componente devono essere disabilitati. La gestione I/O controlla solo se il singolo segnale e' disabilitato e non guarda verso l'alto a livelli piu' alti. Per disabilitare un segnale in un attributo componente, si aggiunge una virgola e il nome del componente seguito dalla maschera di disabilitazione che e' valida per il componente in MethodArguments[0]. Ad esempio in un oggetto in cui i componenti Contactor e CircuitBreaker sono disabilitati MethodArguments[0] puo' contenere

```
3, Contactor 1, CircuitBreaker 1
```

dove '3' e' la maschera di disabilitazione dell'oggetto (che disabilita gli attributi Contactor e CircuitBreaker), e 'Contactor 1' comporta la disabilitazione di un attributo di segnale in Contactor e 'CircuitBreaker 1' disabilita un segnale in CircuitBreaker.

C'e' anche un'altra sintassi con parantesi che consente piu' di due livelli. In questo esempio l'oggetto sopra, Motore, e' una parte di un aggregato piu' grande.

```
(5 (Motor 3 (Contactor 1, CircuitBreaker 1), Temp 1))
```

Attributi del componente con configurazione individuale

Quando il metodo ConfigureComponent e' attivato, Disable viene rimosso da tutti gli attributi del componente, per ripristinare qualsiasi configurazione precedente. A volte ci sono attributi del componente che non fanno parte della configurazione dell'oggetto, ma devono essere configurati singolarmente. Questi componenti non devono essere reimpostati dal metodo ConfigureComponent e devono essere dichiarati in MethodArguments[2] con una virgola come delimitatore. Nell'esempio seguente, gli attributi del componente Motor e Contactor devono essere configurati con i propri metodi ConfigureComponent e non influenzati dal metodo ConfigureComponent dell'oggetto. In MethodArguments[2] e' indicato

```
Motor, Contactor
```

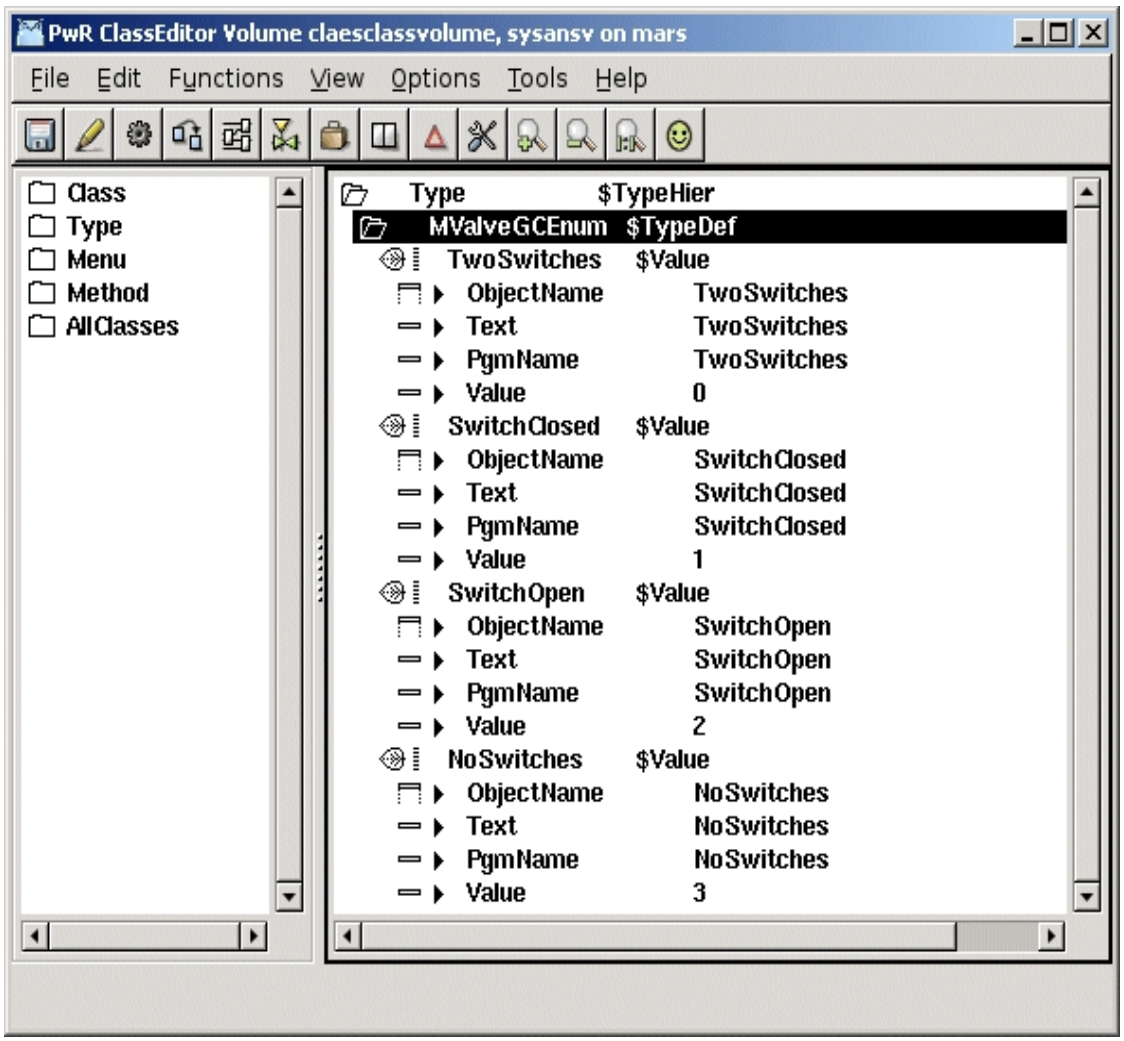
Grafico dell'oggetto
Quando si disegna il grafico dell'oggetto per il componente, e' necessario considerare le diverse configurazioni. Se le differenze tra le configurazioni sono piccole, puoi utilizzare la dinamica Invisibile. Se le differenze sono maggiori, potrebbe essere piu' comodo disegnare grafici separati per le configurazioni. Quindi si inserisce un attributo nell'oggetto principale con il nome GraphConfiguration di tipo Enum. E' comune creare un tipo di enumerazione specifico con le alternative di configurazione. Se GraphConfiguration e' 0, viene utilizzato il grafico standard, altrimenti il valore in GraphConfiguration viene impostato come suffisso al grafo.

Nell'esempio con l'elettrovalvola, MValve, creiamo un tipo di enum, MValveGCEnum, e definiamo i valori

```
TwoSwitches  0
SwitchClosed 1
SwitchOpen   2
NoSwitches   3
```

Per la configurazione TwoSwitches, con valore 0, disegniamo un oggetto grafico con nome mvalve.pwg.
Per SwitchClosed, con valore 1, nominiamo il grafico mvalve1.pwg, per SwichOpen mvalve2.pwg e per NoSwitches mvalve3.pwg.

Indichiamo anche il valore di enumerazione in MethodArguments[1] nell'oggetto $MenuButton per la configurazione corrente. Cio' implica che GraphConfiguration verra' impostato su questo valore quando e' attiva l'alternativa del menu corrente.



Tipo di enumerazione per GraphConfiguration

24.4.5.2 Functionobject

Il functionobject e' l'interfaccia del componente nel programma plc. Definisce input e output che possono essere collegati ad altri oggetti funzionali nell'editor di plc. A differenza di un oggetto funzionale ordinario, il codice funziona anche con i dati nell'oggetto principale.

Il codice puo' essere scritto in plc-code o c-code.

plc-code
Se si desidera mantenere il codice dell'oggetto funzione visibile e se e' necessario eseguire PlcTrace nel codice, e' opportuno utilizzare un oggetto funzione con codice-plc.

Creare un oggetto $ ClassDef e denominare l'oggetto. Preferibilmente usa lo stesso nome dell'oggetto principale seguito dal suffisso "Fo", ad es. MyComponentFo. Quindi attivare Configure-ConnectedFo nel menu popup.

Sotto RtBody viene creato un attributo PlcConnect di tipo AttrRef, che conterra' un

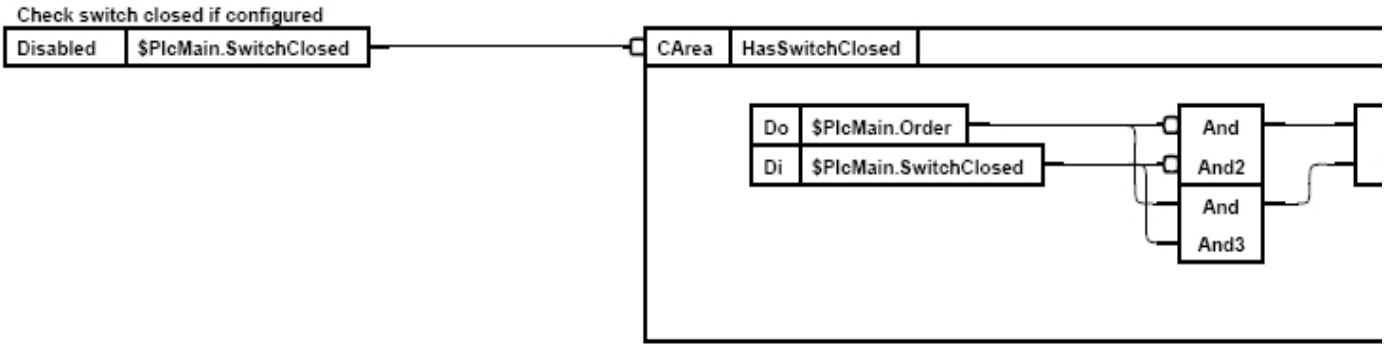
collegamento all'oggetto principale, quando un'istanza e' connessa nell'editor di plc.

Configura ingressi e uscite con oggetti $Input e $Output sotto l'oggetto RtBody. E' inoltre possibile creare oggetti $Intern, ma questo tipo di dati viene solitamente archiviato nell'oggetto principale.
Si noti che l'ordine degli oggetti attributo deve essere $Input, $Intern, $Output.

Il codice viene creato aprendo l'editor plc per l'oggetto Codice. Nel codice, si recuperano i valori da un input, selezionando l'attributo di input nell'oggetto template per il functionobject nel navigatore e si attiva la funzione connect. L'output e' memorizzato in modo simile. Quando i dati devono essere recuperati o archiviati nell'oggetto principale, selezionare l'attributo nell'oggetto modello dell'oggetto principale. I riferimenti all'oggetto funzione sono visualizzati nel codice plc con il simbolo $PlcFo e i riferimenti all'oggetto principale con il simbolo $PlcMain.

Se l'oggetto contiene componenti, l'oggetto funzione di questi componenti viene inserito nel codice plc.

Se hai DisableAttr su segnali o altri attributi, questo deve essere gestito con l'esecuzione condizionale nel codice. Un segnale disabilitato non deve essere letto o scritto nel codice.
Si utilizza l'oggetto Disabilitato sotto la mappa Other, per valutare se un attributo e' disabilitato o meno.
Questo puo' quindi essere collegato a un oggetto CArea che gestisce l'esecuzione condizionale.



Esecuzione delle condizioni con Disabilitazioni e CArea

c-code
Un oggetto funzione con codice c e' configurato con un oggetto $ ClassDef. Assegna un nome all'oggetto e quindi attiva Configure-ConnectedCCodeFo nel menu popup.

Sotto RtBody, vengono creati due attributi, PlcConnect di tipo AttrRef e PlcConnectP che e' un puntatore. In PlcConnect, il riferimento all'oggetto principale viene memorizzato, quando un'istanza e' connessa nell'editor di plc. Quando il programma plc viene inizializzato in runtime, si recupera, con l'aiuto del riferimento, un puntatore all'oggetto principale.
Il puntatore e' memorizzato in PlcConnectP.
Questo viene fatto nel codice c, che e' separato in una funzione init che viene eseguita durante l'inizializzazione del programma plc, e una funzione exec che viene eseguita ad ogni scansione. Per l'oggetto funzione MyComponentFo con l'ingresso In1 e In2 e l'uscita Out2, il codice e'

```
void MyComponentFo_init( pwr_sClass_MyComponentFo  *o)
{
  pwr_tDlid dlid;
  pwr_tStatus sts;

  sts = gdh_DLRefObjectInfoAttrref( &o->PlcConnect,
```

```
        (void **)&o->PlcConnectP, &dlid);
  if ( EVEN(sts))
    o->PlcConnectP = 0;
}

void MyComponentFo_exec( plc_sThread      *tp,
                         pwr_sClass_MyComponentFo *o)
{
  pwr_sClass_MyComponent *co = (pwr_sClass_MyComponent *) o->PlcConnectP;

  if ( !co)
    return;

  o->Out = co->Value = co->Factor * (*o->In1P + *o->In2P);
}
```

24.4.5.3 Oggetto di simulazione

Un oggetto di simulazione viene utilizzato per simulare il processo, sia in condizioni normali che quando si verificano errori diversi. L'oggetto di simulazione legge i segnali di uscita (Do, Ao, Io) nell'oggetto principale e imposta i valori sui segnali di ingresso (Di, Ai, Ii, Co). L'oggetto e' un oggetto funzione che puo' contenere attributi di input e output, ma in genere questi sono mancanti e l'oggetto sta lavorando con i dati nell'oggetto principale e con gli attributi interni che configurano la simulazione e attivano varie condizioni di errore. L'oggetto di simulazione ha spesso un oggetto grafico che viene aperto dal metodo Simula dell'oggetto principale.

Un oggetto di simulazione e' collegato all'oggetto principale tramite un metodo di connessione, allo stesso modo di un oggetto di funzione ordinario. Ma la classe di simulazione ha un altro metodo di connessione rispetto alla classe Fo.
L'oggetto principale dovrebbe contenere l'attributo 'SimConnect' di tipo pwrs: Type-$AttrRef, in cui il metodo di connessione memorizzera' l'identita' per l'oggetto di simulazione quando un oggetto di istanza principale e un oggetto di istanza di simulazione sono connessi.

Una classe di simulazione viene creata allo stesso modo di una classe functionobject e puo' essere scritta in c oppure in plc-code. La classe e' preferibilmente denominata con lo stesso nome della classe principale, seguita dal suffisso 'Sim'.

Creare un oggetto $ ClassDef e denominare l'oggetto. Quindi attivare Configure-ConnectedFo o Configure-ConnectedCCodeFo. Aggiungere eventuali attributi $Input, $Intern e $Output e scrivere il codice in linguaggio plc o c-code. Modificare il metodo di connessione in GraphPlcNode su 26.

La grafica degli oggetti per gli oggetti di simulazione viene spesso disegnata con sfondo blu scuro e testo bianco per essere facilmente riconoscinile dalla grafica di altri oggetti.
Nota che gli attributi nell'oggetto principale possono essere referenziati dalla notazione "&", ad es. &($object.PlcConnect).IndError##Boolean.

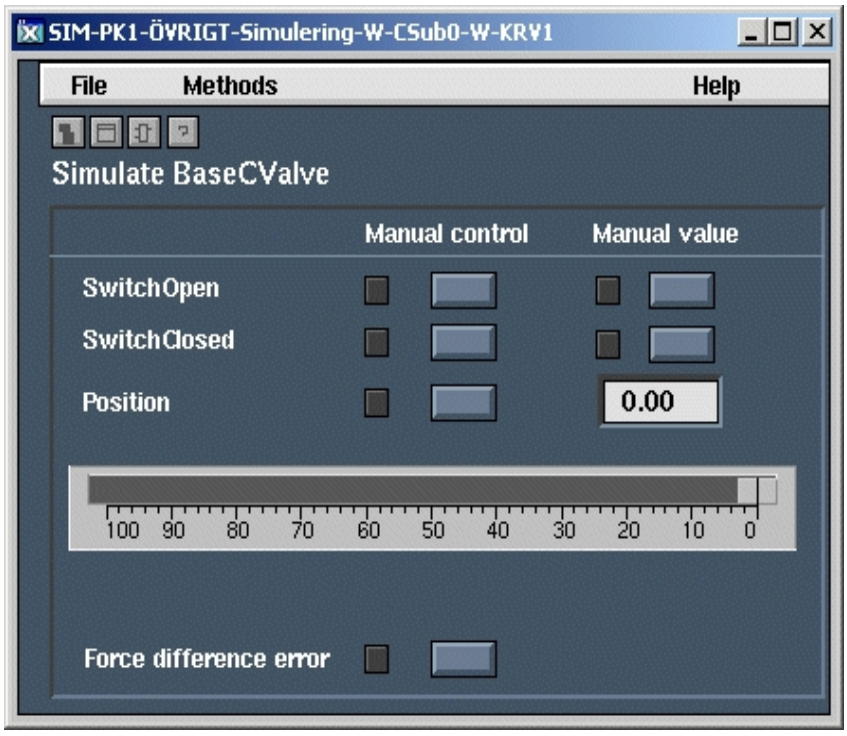
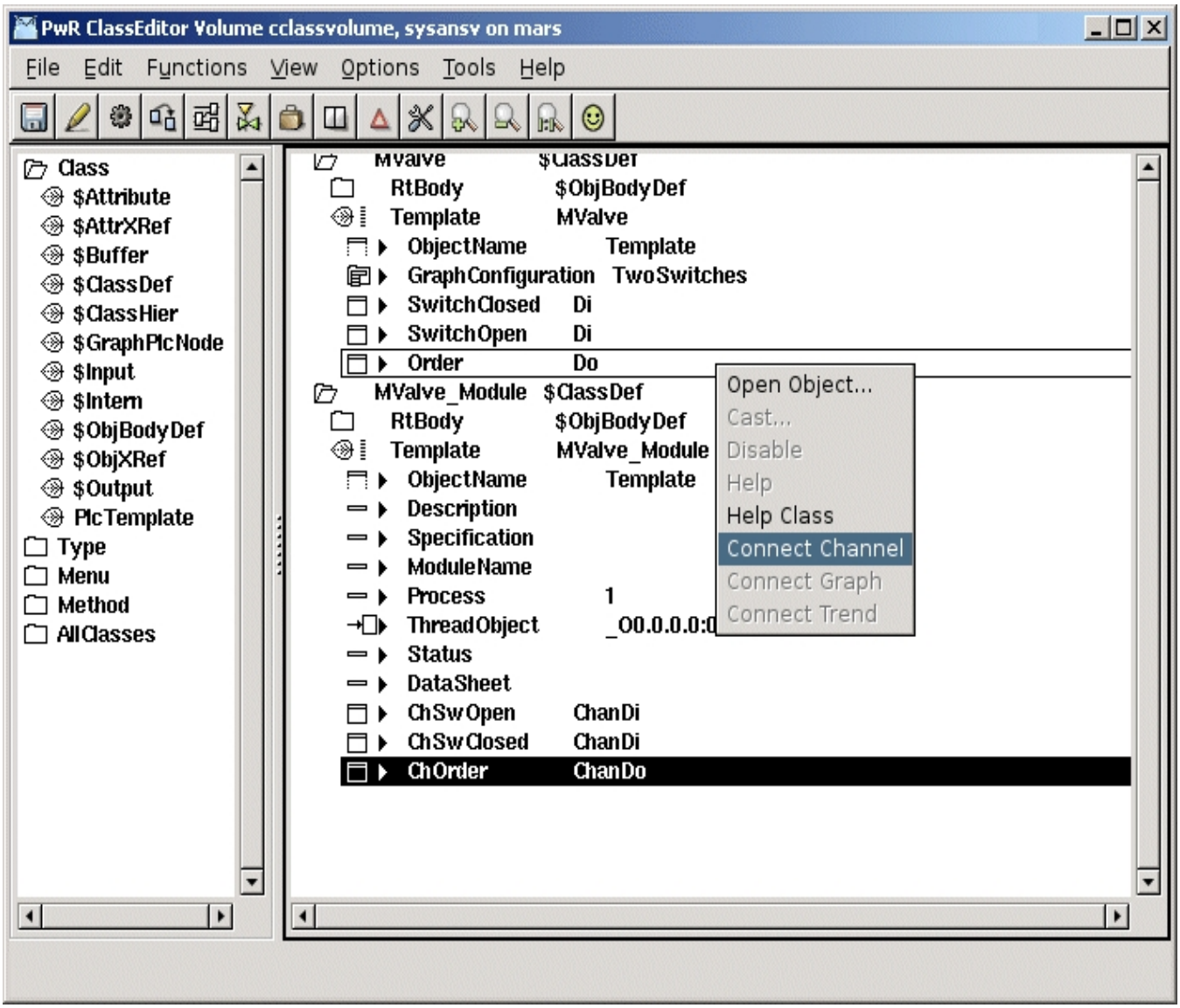


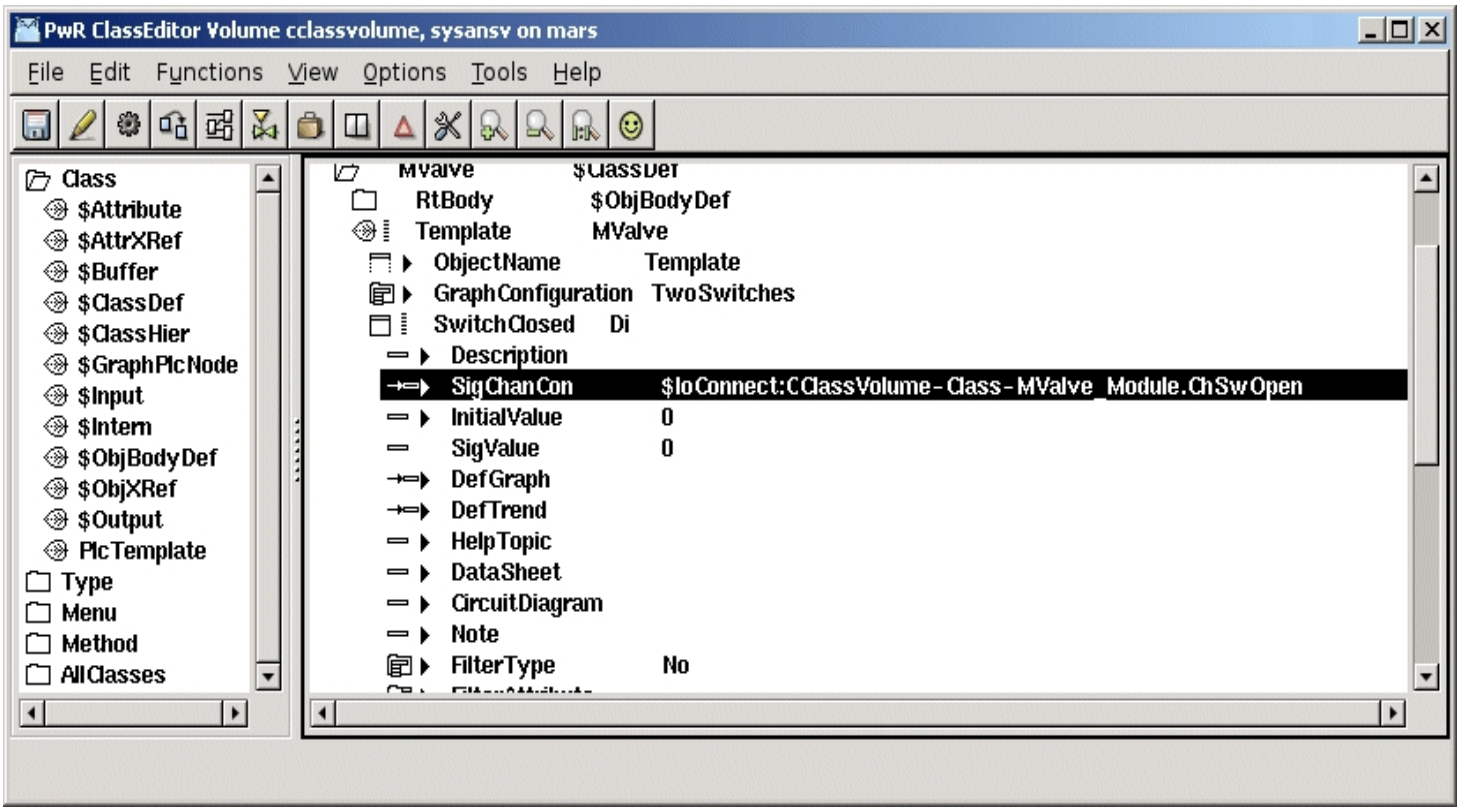
Grafico oggetto per un oggetto di simulazione

24.4.5.4 Oggetto I/O-module

Un oggetto principale per un componente puo' contenere oggetti segnale di tipo Ai, Ai, Ii, Ao, Do, Io e Co. I segnali di un'istanza devono essere collegati agli oggetti del canale nella gerarchia dei nodi. Ad esempio, in Profibus e' possibile creare oggetti modulo, in cui i canali vengono adattati alla dataarea che viene ricevuta e inviata sul bus. Se i segnali di un componente sono gestiti da un oggetto modulo, e' possibile memorizzare riferimenti simbolici agli oggetti canale negli oggetti segnale.
Quindi e' sufficiente eseguire una IoConnection tra il componente e l'oggetto modulo, non e' necessario collegare separatamente ciascun segnale. I riferimenti simbolici sono memorizzati nell'oggetto modello dell'oggetto modello del componente, collegando i segnali nell'oggetto modello ai canali nell'oggetto modello del modulo I/O.
I riferimenti simbolici sono di tipo $IoConnect e vengono convertiti in riferimenti reali durante l'inizializzazione della gestione I/O in runtime.



Un segnale e' collegato a un canale in un modulo I/O.



Riferimento simbolico all'oggetto canale

24.4.5.5 Grafica dell'oggetto

Il grafico dell'oggetto (oggetto di rappresentazione grafica del componente) ha lo stesso nome del componente, ma con lettere minuscole. Per le classi nel sistema di base Proview si aggiunge il prefisso 'pwr_c_'. I grafici sono modificati normalmente in Ge. Nella dinamica si scambia il nome dell'oggetto con '$object'. Il grafico degli oggetti per gli oggetti nel sistema di base sono stati disegnati seguendo le seguenti linee guida.

Menu
Dovrebbe esserci un menu con i menu a tendina File, Methods, Signals and Help.

Il menu File dovrebbe avere le seguenti voci

```
Print   Command     print graph/class/inst=$object
Close   CloseGraph
```

I metodi dovrebbero avere le seguenti voci

```
Help            Invisible $cmd(check method/method="Help"/object=$object)
                Command   call method/method="Help"/object=$object
Note            Invisible $cmd(check method/method="Note"/object=$object)
                Command   call method/method="Note"/object=$object
Trend           Invisible $cmd(check method/method="Trend"/object=$object)
                Command   call method/method="Trend"/object=$object
Fast            Invisible $cmd(check method/method="Fast"/object=$object)
                Command   call method/method="Fast"/object=$object
Help            Invisible $cmd(check method/method="Photo"/object=$object)
                Command   call method/method="Photo"/object=$object
DataSheet       Invisible $cmd(check method/method="DataSheet"/object=$object)
```

```
                Command   call method/method="DataSheet"/object=$object
Hist Event...   Invisible $cmd(check method/method="Hist Event..."/object=$object)
                Command   call method/method="Hist Event..."/object=$object
Block Events..  Invisible $cmd(check method/method="Block Events..."/object=$object)
                Command   call method/method="Block Events..."/object=$object
RtNavigator     Invisible $cmd(check method/method="RtNavigator"/object=$object)
                Command   call method/method="RtNavigator"/object=$object
Open Object     Invisible $cmd(check method/method="Open Object"/object=$object)
                Command   call method/method="Open Object"/object=$object
Open Plc        Invisible $cmd(check method/method="Open Plc"/object=$object)
                Command   call method/method="Open Plc"/object=$object
Circuit Diagram Invisible $cmd(check method/method="Circuit Diagram"/object=$object)
                Command   call method/method="Circuit Diagram"/object=$object
HelpClass       Invisible $cmd(check method/method="HelpClass"/object=$object)
                Command   call method/method="HelpClass"/object=$object
```

I segnali dovrebbero contenere tutti i segnali nel componente e aprire la grafica dell'oggetto per ciascun segnale.

Esempio

```
SwitchOpen Di    Command     open graph/class /inst=$object.SwitchOpen
SwitchClosed Di  Command     open graph/class /inst=$object.SwitchClosed
Order Do         Command     open graph/class /inst=$object.Order
```

Il menu Guida dovrebbe contenere Guida e HelpClass

```
Help             Command     call method/method="Help"/object=$object
HelpClass        Command     call method/method="HelpClass"/object=$object
```

Toolbar
La barra degli strumenti contiene pulsanti per i metodi. Le dinamiche sono le stesse del menu metodi sopra. A destra, c'e' anche un pulsante per la grafica dell'oggetto dell'oggetto simulato. Questo ha la dinamica

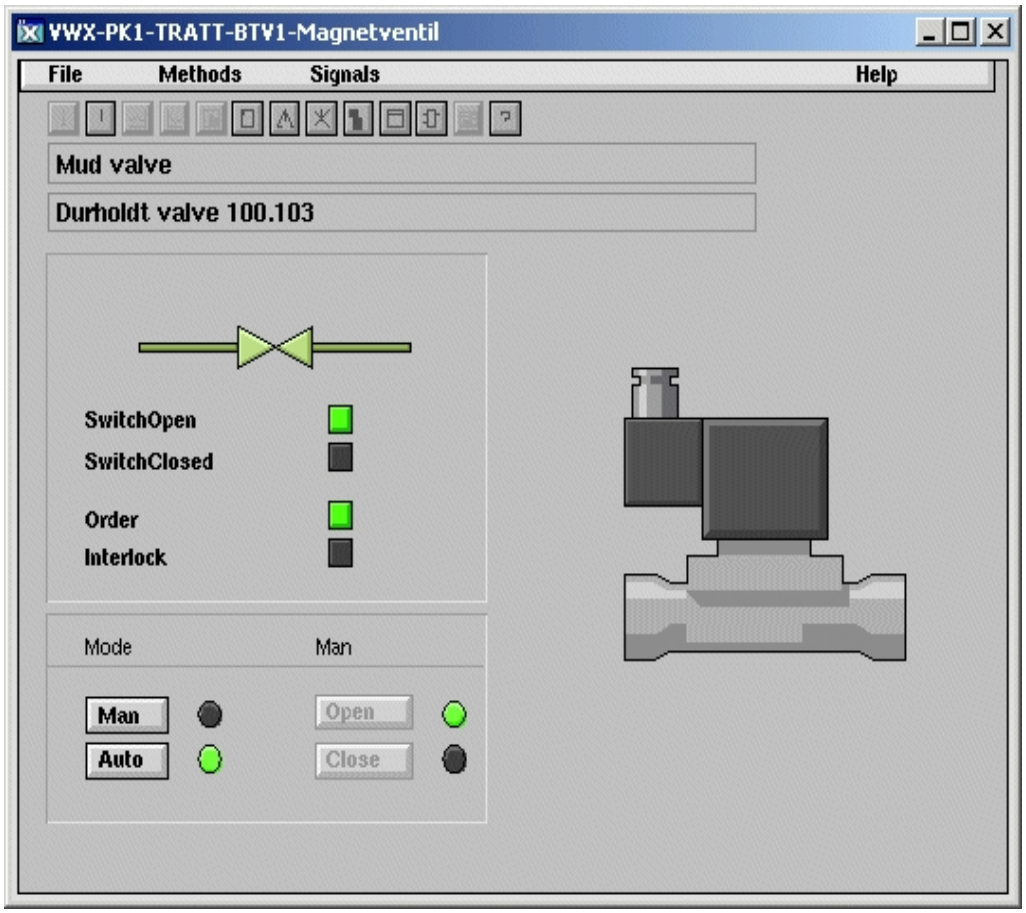
```
Invisible   $cmd(check method/method="Simulate"/object=$object)
Command     call method/method="Simulate"/object=$object
```

Sotto i pulsanti del metodo ci sono due campi di testo che visualizzano gli attributi Descrizione e Specifica nel componente con la dinamica

```
Description     Value.Attribute $object.Description##String80
                Value.Format    %s
Specification   Value.Attribute $object.Specification##String80
                Value.Format    %s
```

Nella riga piÃ¹ bassa del grafico, vengono visualizzati tutti i messaggi di Notes, con un pulsante per modificare o rimuovere il messaggio.

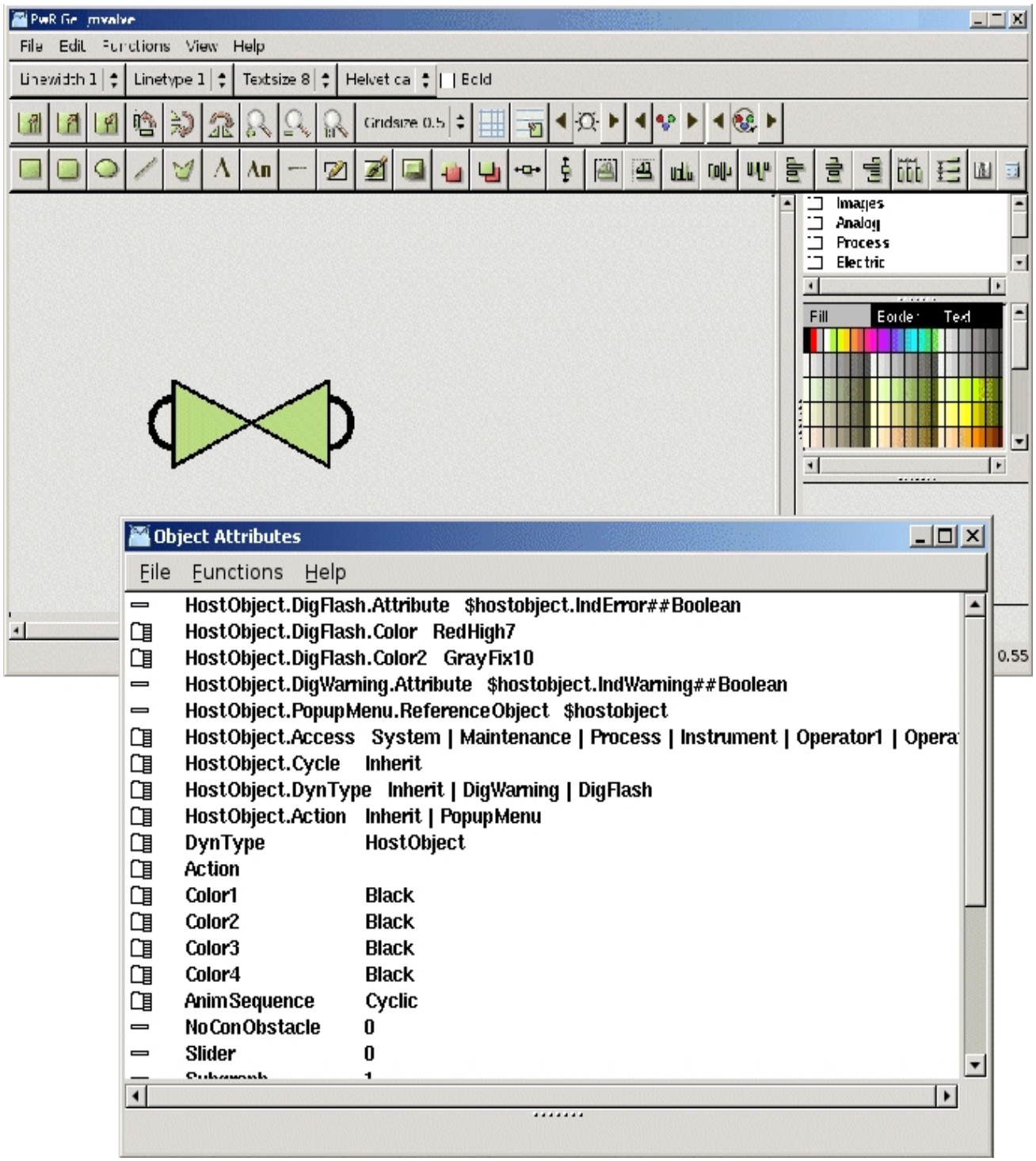
```
Notes button    Invisible       $object.Note##String80
                Command         call method/method="Note"/object=$object
Notes text      Value.Attribute $object.Note##String80
                Value.Format    %s
```



Object graph

24.4.5.6 Graphic symbol

Il simbolo grafico e' disegnato in Ge e dato l'hostObject dinamico predefinito. Nell'oggetto host dinamico, tipi diversi di dinamica sono connesse agli attributi nell'oggetto. Il nome oggetto viene scambiato con '$hostobject' nella dinamica. Spesso si creano gli attributi IndWarning e INDError nell'oggetto principale e si colora il simbolo di giallo o rosso, o rosso lampeggiante, quando questi sono impostati.



Il simbolo grafico viene disegnato in Ge con la dinamica predefinita di HostObject.

24.5 Costruisci il volume della classe

Quando si crea il volume della classe, vengono creati un file di caricamento e due file struct.

Loadfile
Il file di caricamento e' archiviato in $ pwrp_load e ha lo stesso nome del volume, con lettere minuscole.
Il filetype e' .dbs, ad esempio il file di caricamento per il volume CVolMerk1 e' $pwrp_load/cvolmerk1.dbs.

L'ora in cui viene creato il file di caricamento e' archiviata nel file. Inoltre, viene archiviata la versione di altri volumi di classe da cui dipende il file di caricamento. All'avvio del runtime, viene verificato che le versioni correnti nel sistema coincidano con le versioni registrate nel file di caricamento.
Se una qualsiasi versione non coincide, viene visualizzato il messaggio 'Version mismatch' e l'avvio viene interrotto.

e' possibile visualizzare la versione di un file di caricamento e le versioni dei volumi dipendenti con wb_ldlist.

```
$ wb_ldlist $pwrp_load/cvolmerk1.dbs
Volume     CVolMerk1       21-DEC-2007 13:52:05.22 (1198241525,227130443) 25364
VolRef     CVolMerk1       21-DEC-2007 13:52:05.22 (1198241525,227130443) 25364
VolRef     pwrs            12-DEC-2007 08:35:06.98 (1197444906,983976467) 1
VolRef     pwrb            12-DEC-2007 08:35:09.93 (1197444909,930182284) 2
VolRef     BaseComponent   12-DEC-2007 08:35:26.92 (1197444926,926679004) 10
```

Structfiles
Quando si crea un volume di classe, vengono generati due file di inclusione, un file .h e un file .hpp.

Se il volume della classe contiene oggetti funzioni o classi utilizzate negli oggetti CArithm o DataArithm, e' necessario includere il file .h in $ pwrp_inc/ra_plc_user.h.

Classi di aggiornamento
Quando viene creato il volume della classe, e' necessario aggiornare le classi nel volume principale o secondario nel progetto. L'aggiornamento e' attivato nel configuratore per il volume root o sub, dal menu con 'Function/Update Classes'. Se una classe viene modificata, gli oggetti di istanza della classe vengono adattati alla classe modificata. Verranno inoltre aggiornati tutti i riferimenti alle istanze della classe.

24.6 Documentazione delle classi

Per gli oggetti $ClassDef e $Attribute, esiste un blocco di documentazione, che viene compilato dall'editor di oggetti. Il blocco della documentazione, insieme alla descrizione della classe, viene utilizzato quando la documentazione della classe viene generata in formato xtthelp o html quando viene creato il volume della classe.

Il blocco della documentazione per l'oggetto $ ClassDef dovrebbe contenere una descrizione della classe e il blocco della documentazione per l'oggetto $ Attribute una descrizione dell'attributo.

24.6.1 Generare file di aiuto Xtt

I file di aiuto per xtt sono generati con il comando

```
co_convert -xv -d $pwrp_exe/ $pwrp_db/userclasses.wb_load
```

Il comando genera un file di aiuto $pwrp_exe/'volumename'_xtthelp.dat, ed e' corretto inserire un collegamento al file nel file di aiuto xtt per il progetto $pwrp_exe/xtt_help.dat:

Esempio per il volume di classe cvoltank

<topic> index

...

User classes<link>cvoltank,"",$pwrp_exe/cvoltank_xtthelp.dat

</topic>

...

<include> $pwrp_exe/cvoltank_xtthelp.dat

24.6.2 Generare documentazione html

i file html sono generati dal comando

```
co_convert -wv -d $pwrp_web/ $pwrp_db/userclasses.wb_load
```

Il comando genera, tra gli altri, il file $pwrp_web/'volumename'_index.html che contiene la pagina iniziale per la documentazione della classe. Questo, insieme agli altri file ($pwrp_web/'volumename'_*. Html) dovrebbe essere copiato in una directory appropriata del server web.

Un collegamento alla documentazione viene effettuato con un oggetto WebLink che punta all'URL "volumename'_index.html.

Se vuoi essere in grado di mostrare la struttura per le classi, converti il file h con co_convert

```
co_convert -cv $pwrp_web/ $pwrp_inc/'volymsnamn'classes.h
```

Se vuoi anche essere in grado di visualizzare il codice di oggetti plc, devi aggiungere i tag aref nel file c o h del codice e convertirlo con

```
co_convert -sv -d $pwrp_web/ 'filename'
```

24.6.3 ClassDef

Esempio

```
@Author Homer Simpson
@Version 1.0
@Codice ra_plc_user.c
@Riepilogo Breve descrizione di questa classe
Descrizione di
questa classe.
```

```
Vedi anche
@link Example plat.html
@classlink AnotherPlate cvolvhxn2r_anotherplate.html
```

Tags

Author	Autore o descrizione della classe
Version	Versione della classe
Code	File che contiene il codice per la classe
Summary	Summario
Link	Link arbitrario
Classlink	Link a un'altra classe

wb_load sintassi

24.6.3.1 @Author

Author. Optional.

Sintassi

```
@Author 'nome dell'autore'
```

24.6.3.2 @Version

Version. Optional.

Sintassi

```
@Version 'version number'
```

24.6.3.3 @Code

Per le classi con codice plc e' possibile specificare il nome del file c. Opzionale.
Anche il file c deve essere convertito dal comando: co_convert -c -d $pwrp_web/'filename'

Sintassi

```
@Code 'filename'
```

24.6.3.4 @Summary

Breve descrizione in una riga. Opzionale.
Questo e' mostrato nel file di indice nel file di aiuto xtt. Non usato in html.

Sintassi

```
@Summary 'text'
```

24.6.3.5 @Link

Un link a un URL arbitrario. Viene visualizzato solo nella documentazione html, non in Xtt.
Il link deve essere inserito dopo la descrizione della classe.

Sintassi

```
@Link 'URL'
```

24.6.3.6 @Classlink

Un link a un'altra classe. Questo link funziona sia in html che in xtt.
Il link deve essere inserito dopo la descrizione della classe.

Sintassi

```
@Classlink 'html-filename'
```

24.6.3.7 wb_load syntax

La documentazione di una classe e' scritta sopra la riga $ClassDef.

```
!
!/**
!  @Author Homer Simpson
!  @Version 1.0
!  @Code ra_plc_user.c
!  @Summary Brief description of this class
!  Description of
!  this class.
!
!  See also
!  @link Example plat.html
!  @classlink AnotherPlate cvolvhxn2r_anotherplate.html
!*/
!
Object        Plat    $ClassDef 1
```

!/**

Inizio di un blocco di documentazione.
Tutto il testo tra!/** e !*/ sara' scritto come una descrizione della classe.

!*/

Fine di un blocco di documentazione.

24.6.4 Attributo

Esempio

```
@Summary Lunghezza della piastra
Una descrizione piu' dettagliata
dell'attributo Lunghezza...
```

@Summary

Breve descrizione in una riga. Se esiste un @Summary, questo testo viene inserito nella tabella degli attributi nel file html. Se non c'e' un riassunto, viene scritta l'intera descrizione.
Non utilizzato in xtt.

wb_load sintassi

24.6.4.1 Sintassi wb_load

La documentazione di un attributo e' scritta sopra la riga $Attribute, $Input, $Output o $Intern.

```
      !/**
```

```
!  @Summary Plate length
!  A more detailed description
!  of the attribute Length...
!*/
Object    Length $Attribute 3
  Body             SysBody
    Attr           TypeRef = "pwrs:Type-$Float32"
  EndBody
EndObject
```

!/**

Inizio di un blocco di documentazione.
Tutto il testo tra !/** e !*/ sara' scritto come descrizione dell'attributo.

!*/

Fine di un blocco di documentazione.

24.6.5 Sintassi per i file c- e h-

Se si desidera utilizzare i collegamenti ai file c- e h-, anche questi devono essere convertiti in html. C'e' anche una funzione per aggiungere segnalibri.

Lo structfile per le classi viene generato automaticamente con i segnalibri.

```
/**
   MyPlcClass

   Descrizione per la classe che non e' visualizzata
   da nessuna parte ma nel codice.

   @aref MyPlcClass MyPlcClass
*/
void MyPlcClass_exec(...)
```

@aref

@aref deve essere inserito all'interno di un /*_* ... */ blocco. All'interno del blocco ci possono essere anche commenti che non sono gestiti dal convertitore.

Sintassi

```
@aref 'bookmark' 'text'
```

25 Amministrazione

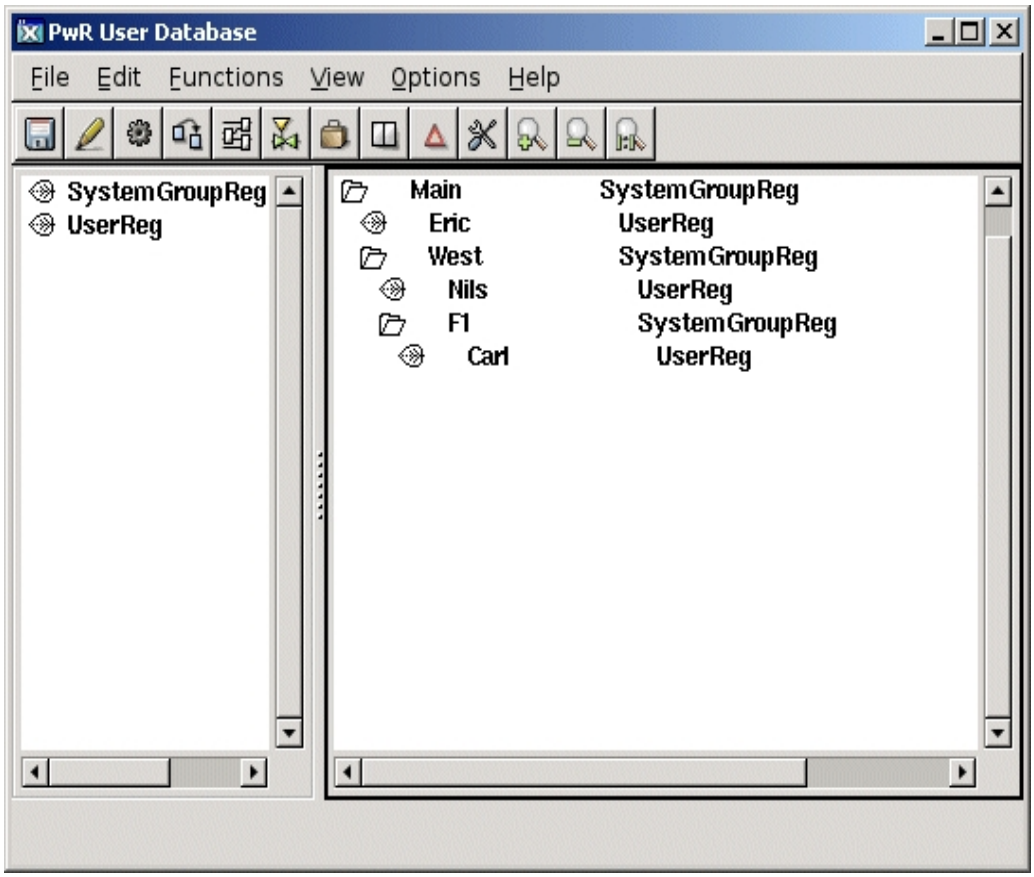
25.1 Utenti

Per accedere all'ambiente di sviluppo e runtime di Proview e' necessario effettuare il login con nome utente e password. Gli utenti sono registrati nel database degli utenti e qui sono concessi i privilegi che stabiliscono l'autorizzazione degli utenti ad apportare modifiche al sistema.

I sistemi che condividono gli stessi utenti sono raggruppati in un gruppo di sistemi e gli utenti di questo gruppo sono definiti. E' inoltre possibile creare una gerarchia di gruppi di sistemi in cui i gruppi figli ereditano gli utenti del proprio genitore e possono essere definiti utenti aggiuntivi per ogni gruppo figlio.

Un sistema e' connesso a un gruppo di sistemi dall'attributo SystemGroup nell'oggetto $System. La notazione per un gruppo di sistemi in una gerarchia e' il nome del gruppo separato da un punto, ad esempio "Main.West.F1".

Nell'esempio seguente, Eric e' responsabile di tutti i sistemi nello stabilimento ed e' definito al livello piu' alto nella gerarchia. Nils sta lavorando con il lato ovest dello stabilimento ed e' definito nel gruppo di sistemi "Ovest". Infine, Carl lavora con i sistemi nella parte F1 dello stabilimento. Tutti i gruppi di sistema hanno l'attributo UserInherit, che afferma che un gruppo figlio eredita tutti gli utenti del genitore.



Utenti e gruppi di sistema sono creati dall'interfaccia di Amministrazione:

- Avviare l'interfaccia di amministrazione con il comando 'pwra'
- Inserire UserDatabase dal menu 'File/Open/UserDatabase'.
- Accedi inserendo il comando di login. Apri il prompt di accesso dal menu 'Functions/Command' e inserisci 'login / adm' sulla riga di comando. Se e' presente l'amministratore di sistema, e' necessario aggiungere nome utente e password a un utente definito nel gruppo di sistema amministratore.
- Inserisci la modalita' di modifica dal menu 'Edit/Edit mode'.
  I systemgroup e gli utenti sono rappresentati dall'oggetto delle classi SystemGroupReg e UserReg, che vengono visualizzati nella palette a sinistra. Un oggetto viene creato selezionando una classe nella tavolozza. Successivamente, si fa clic con il tasto centrale del mouse sul fratello o genitore futuro sul nuovo oggetto. Se si fa clic sulla mappa/foglia nell'oggetto di destinazione, il nuovo oggetto viene posizionato come primo figlio, se si fa clic a destra della mappa/foglia, viene posizionato come fratello.
- Crea un gruppo di sistemi selezionando 'SystemGroupReg' nella palette e fai clic con MB2 (il pulsante centrale del mouse) nella finestra a destra. Aprire l'oggetto SystemGroupReg e immettere l'attributo nome ant per il gruppo di sistema. Inserisci il nome completo della gerarchia, ad es. 'Main.West'.
- Creare un utente selezionando 'UserReg' nella palette e fare clic con MB2 sulla mappa/foglia dell'oggetto SystemGroupReg di cui UserReg dovrebbe essere figlio. Apri l'oggetto e inserisci nome utente, password e privilegi per l'utente.
- Salvare.
- Effettuare il Logout con il comando 'logout'.

Il database utenti risiede nella directory $pwra_db.

25.2 Registra volumi

Tutti i volumi in una rete devono avere un nome di volume univoco e un'identita' di volume. Per garantire questo, tutti i volumi sono registrati in un volume globale.

La registrazione viene effettuata dall'amministratore:

- Avvia l'interfaccia di amministrazione con il comando 'pwra'
- Entrare in modalita' volume dal menu 'File/Open/GlobalVolumeList'.
- Accedi come amministratore.
- Accedi alla modalita' di modifica dal menu 'Edit/Edit mode'.
  I volumi sono registrati da oggetti della classe VolumeReg, che vengono visualizzati nella palette a sinistra. Nella tavolozza, c'e' anche la classe $Hier, che puo' essere usata per ordinare gli oggetti VolumeReg in una struttura ad albero.
- Creare un oggetto VolumeReg, aprire l'oggetto e immettere volumename (uguale a objectname), volumeidentity e project.
- Salva.
- Esci con il comando 'logout'.

Nome Volume

Il nome del volume, un nome univoco con un massimo di 31 caratteri.

Identita' volume

L'identita' volume e' una parola di 32 bit specificato nella forma v1.v2.v3.v4 dove v1, v2, v3 e v4 sono numeri nell'intervallo 0-255. A seconda della classe del volume, i numeri possono essere scelti a intervalli separati.

RootVolumes	0. 1-254. 1-254. 1-254
User ClassVolumes	0. 0. 2-254. 1-254

La DirectoryVolume ha sempre l'identita' 254.254.254.253

25.3 Crea un progetto

Un progetto e' un numero di nodi e volumi che condividono lo stesso ambiente di sviluppo. Di solito consiste in alcune stazioni di processo e un paio di stazioni operatore che controllano una parte dell'impianto, ma non ci sono restrizioni nella dimensione di un progetto. E' possibile scegliere di avere ciascun nodo nel proprio progetto o tutti i nodi nello stesso progetto.

- Tutti i nodi di un progetto (sullo stesso bus) hanno un collegamento nethandler tra loro.
- Tutti i volumi e i nodi condividono lo stesso albero delle directory.
- Tutti i nodi devono essere aggiornati alle nuove versioni di Proview allo stesso tempo.

Una dimensione comune e' 1-10 nodi in un progetto. Troppi nodi aumenteranno il sovraccarico della comunicazione e renderanno piu' difficile l'aggiornamento del progetto.

Crea il progetto nell'interfaccia di amministrazione:

- Avvia l'interfaccia di amministrazione con il comando 'pwra'.
- L'elenco dei progetti viene mostrato come predefinito all'avvio dell'amministratore.

Puo' anche essere aperto dal menu (File/Open/ProjectList).
- Accedi come amministratore.
- Entra nella modalita' di modifica dal menu'Edit/Edit mode'.
  I progetti sono rappresentati da oggetti di classe ProjectReg, che vengono visualizzati nella palette a sinistra. Gli oggetti $ Hier possono essere usati per ordinare gli oggetti ProjectReg in una struttura ad albero.
- Creare un oggetto ProjectReg ed inserire nome del progetto, versione base, percorso e descrizione.
- Il progetto viene creato quando si salva. Per prima cosa bisogna confermare le modifiche.
- Salva e disconnetti.

Nome del progetto

Un progetto ha un nome di progetto che identifica il progetto nell'ambiente di sviluppo. e' simile al nome di sistema che identifica il progetto nell'ambiente di runtime, ma non deve essere lo stesso. In realta' un sistema puo' avere diversi progetti nell'ambiente di sviluppo. Quando si aggiorna o si effettua una modifica importante del sistema, e' consigliabile prendere una copia del progetto e mantenere disponibile la versione attualmente in esecuzione del sistema per piccole correzioni. La copia viene quindi creata con un nuovo nome di progetto, sebbene abbia lo stesso nome di sistema.

Base

Proview e' un sistema multiversione, cioe' versioni diverse di Proview possono essere installate nello stesso ambiente di sviluppo e progetti di diverse versioni di Proview possono coesistere nello stesso ambiente di sviluppo. Un progetto punta a una base Proview, ad es. V3.4, V4.0, e quando si crea un progetto e' necessario scegliere su quale base deve puntare il progetto.

Path (percorso)

Il progetto consiste in un albero di directory in cui sono archiviati database, file di sorgenti, archivi ecc.
Il percorso e' la directory principale di questo albero.

26 Strumenti

pwrc	Interfaccia di comando al database di sviluppo.
wb_ge	Editor GE
co_help	Finestra di aiuto.
pwr_user	Interfaccia a riga di comando per database utente.
wb_ldlist	Verifica le versioni di loadfile.

26.1 pwrc

Interfaccia di comando al database di sviluppo.

```
pwrc        Proview workbench commands

Argomenti:
  -v 'volume'    Load volume 'volume'.
  -h             Print usage.

  Altri argomenti sono trattati come un comando e passati al parser del comando
  Se un comando viene fornito come argomento, il comando verra' eseguito ed il
  programma e' quindi terminato.
  Se non viene dato alcun comando, pwrc richiedera' un comando.

Esempi:
  $ pwrc -v MyVolume
  pwrc>

  $ pwrc -a show volume
  directory     Attached Db  $DirectoryVolume 254.254.254.253
  MyVolume               Db  $RootVolume        0.1.99.20
```

26.2 co_help Finestra di aiuto

Apre un file di aiuto Proview. Di default il file di aiuto del progetto viene aperto.

```
>co_help

Uso:

  co_help [-t 'topic'] [-s 'sourcefile'] [-b 'bookmark']

Argomenti:

  -t    Argomento della guida, predefinito 'index'
  -s    File sorgente di aiuto
  -b    Segnalibro
```

```
-l    Lingua, e.g sv_se
-c    Apri la guida di configurazione
-d    Apri la Guida del designer
-g    Aprire Manuale di riferimento Ge
-o    Apri la guida per l'operatore
```

26.3 wb_ge Ge editor

Avvia l'editor di ge senza aprire un database.
Utile quando desideri velocizzare l'apertura di un grafico Ge, ma non effettuerai nessuna connessione al database.

```
> wb_ge ['pwg-file']
```

26.4 pwr_user

Si utilizza pwr_user per creare systemgroup e utenti nel database utente.
La configurazione viene eseguita con i comandi.

pwr_user viene avviato dal prompt dei comandi.

Di seguito e' riportata una descrizione dei diversi comandi disponibili per creare, modificare ed elencare gruppi di sistema e utenti.

add group	Aggiungi un gruppo di sistema
add user	Aggiungi un utente
get	Ottieni un utente
list	Elenca i gruppi di sistema e gli utenti
load	Carica l'ultimo database salvato
modify group	Modifica un gruppo di sistema
modify user	Modifica un utente
remove group	Rimuovere un gruppo di sistema
remove user	Rimuovi un utente
save	Salva
su	Accedi come super utente

26.4.1 add

add group
add user

26.4.1.1 add group

Crea un gruppo di sistema

```
pwr_user> add group 'name' [/nouserinherit]
```

/nouserinherit	L'attributo UserInherit non e' impostato per il gruppo di sistema. Come impostazione predefinita e' impostato UserInherit.

26.4.1.2 add user

Crea un utente.

pwr_user> add user 'name' /group= /password= [/privilege=]
[/rtread][/rtwrite][/system][/maintenance][/process]
[/instrument][/operator1][/operator2]...[oper10][/devread]
[/devplc][/devconfig][/devclass]

/group	Systemgroup dell'utente
/password	Password dell'utente
/privilege	Privilegi se questo e' fornito come maschera, cioe' un valore intero
/rtread	L'utente ha diritto a RtRead
/rtwrite	L'utente ha diritto a RtWrite
/system	L'utente ha diritto a System
/maintenance	L'utente ha diritto a Maintenance
/process	L'utente ha diritto a Process
/operator1	L'utente ha diritto a Operator1
...	
/operator9	L'utente ha diritto a Operator9
/operator10	L'utente ha diritto a Operator10
/devread	L'utente ha diritto a DevRead
/devplc	L'utente ha diritto a DevPlc
/devconfig	L'utente ha diritto a DevConfig
/devclass	L'utente ha diritto a DevClass

26.4.2 get

Recupera un utente con un algoritmo utilizzato in runtime.

pwr_user> get 'username' /group= /password=

26.4.3 list

Elenca systemgroup e utenti.

pwr_user> list

26.4.4 load

Carica l'ultimo database salvato e ripristina la sessione corrente.

26.4.5 modifica

modify group
modify user

26.4.5.1 modify group

Modifica un gruppo di sistema.

```
pwr_user> modify group 'name' /[no]userinherit
```

/userinherit	Imposta l'attributo UserInherit che indica che il systemgroup eredita utenti dal suo genitore nella gerarchia di systemgroup. Negato con /nouserinherit

26.4.5.2 modify user

Modifica un utente.

```
pwr_user> modify user 'name' /group= [/password=][/privilege=]
 [/rtread][/rtwrite][/system][/maintenance][/process]
     [/instrument][/operator1][/operator2]...[oper10][/devread]
     [/devplc][/devconfig][/devclass]
```

/group	Systemgroup dell'utente
/password	Password dell'utente
/privilege	Privilegi se questi sono forniti come maschera, cioe' un valore intero
/rtread	L'utente ha diritto a RtRead
/rtwrite	L'utente ha diritto a RtWrite
/system	L'utente ha diritto a System
/maintenance	L'utente ha diritto a Maintenance
/process	L'utente ha diritto a Process
/operator1	L'utente ha diritto a Operator1
...	
/operator9	L'utente ha diritto a Operator9
/operator10	L'utente ha diritto a Operator10
/devread	L'utente ha diritto a DevRead
/devplc	L'utente ha diritto a DevPlc
/devconfig	L'utente ha diritto a DevConfig
/devclass	L'utente ha diritto a DevClass

26.4.6 remove

remove group
remove user

26.4.6.1 remove group

Rimuovere un gruppo di sistema.

```
pwr_user> remove group 'name'
```

26.4.6.2 remove user

Rimuovi un utente.

```
pwr_user> remove user 'name' /group=
```

26.4.7 save

Salva la sessione corrente.

```
pwr_user> save
```

26.4.8 su

Accesso come super utente. Come superutente puoi visionare le password degli utenti quando elenchi il database.
su richiede password.

```
pwr_user> su 'password'
```

26.5 wb_ldlist

Mostra la versione dei volumi nei file dbs.
Utilizzato per indagare sulla mancata corrispondenza della versione.

> wb_ldlist <dbs-file>

Esempio

```
wb_ldlist $pwrp_load/volpwrdemo.dbs
Volume     VolPwrDemo      27-MAR-2014 11:06:48.67  0.254.254.200   RootVolume
VolRef     VolPwrDemo      27-MAR-2014 11:06:48.67  0.254.254.200   RootVolume
VolRef     pwrs            14-FEB-2014 16:57:21.19  0.0.0.1         ClassVolume
VolRef     pwrb            14-FEB-2014 16:57:24.82  0.0.0.2         ClassVolume
VolRef     BaseComponent   14-FEB-2014 16:57:52.68  0.0.0.10        ClassVolume
VolRef     OtherManufacturer 14-FEB-2014 16:58:22.73  0.0.250.1     ClassVolume
VolRef     ABB             14-FEB-2014 16:58:10.44  0.0.250.2       ClassVolume
VolRef     Profibus        14-FEB-2014 16:57:34.40  0.0.250.7       ClassVolume
VolRef     Inor            14-FEB-2014 16:58:16.18  0.0.250.8       ClassVolume
VolRef     OtherIO         14-FEB-2014 16:57:55.31  0.0.250.10      ClassVolume
```

27 OPC

Proview ha implementato il protocollo OPC XML / DA per lo scambio di dati con altri software di automazione. Per maggiori informazioni su OPC vedi www.opcfoundation.org.

27.1 OPC XML/DA Server

Un server OPC XML / DA e' un servizio Web da cui un client OPC XML / DA puo' ottenere informazioni su un sistema Proview. Un client opc puo', ad esempio, esplorare la gerarchia degli oggetti, leggere e scrivere il valore dell'attributo e iscriversi agli attributi.

Il server opc implementa anche il protocollo http e non e' connesso a un server web.
Il numero di porta di opc_server e' impostato su 80, l'URI per il servizio web e' sul nodo 'mynode' e'

```
http://mynode
```

Se e' presente un server Web, questo normalmente ha assegnato la porta 80 e un'altra porta deve essere scelta per opc_ever. Se viene scelta la porta 8080, l'URI sara'

```
http://mynode:8080
```

Navigazione

La navigazione OPC XML / DA supporta rami e articoli. L'articolo contiene un valore, mentre il ramo e' un componente gerarchico senza valore. Non c'e' alcun supporto per gli oggetti, quindi un oggetto deve essere implementato come un ramo, mentre un attributo e' implementato come un elemento. Anche gli array sono implementati come rami, mentre gli elementi dell'array (che non e' una classe) sono implementati come item.

Threads

Se il client opc utilizza gli attributi HoldTime e WaitTime nella richiesta SubscriptionPolledRefresh, il server opc deve essere multi-thread, ovvero, per ogni richiesta, viene creato un nuovo thread. Se HoldTime e WaitTime non vengono utilizzati (come nel client Opiew di Proview), tutte le richieste possono essere gestite in un singolo thread, il che richiede meno tempo. I multithread o meno sono configurati nell'oggetto di configurazione per il server opc. Il valore predefinito e' 'IfNeeded' che attiva il multithreading per un client se vengono rilevati HoldTime o WaitTime.

Accesso client

Per accedere a un server opView di Proview, l'indirizzo IP del client deve essere configurato nell'oggetto di configurazione per il server opc. Qui puoi anche scegliere se il client ha accesso a ReadWrite o ReadOnly, dove ReadOnly consente al client di leggere e

sottoscrivere i valori degli attributi, mentre ReadWrite consente anche di scrivere valori di attributo.

Buffering di abbonamenti

Il server non supporta il buffering delle sottoscrizioni.

Configurazione

Il server opc e' configurato con un oggetto Opc_ServerConfig posizionato nella gerarchia del nodo. L'oggetto di configurazione causera' l'avvio di un processo del server (opc_server) all'avvio di Proview.

27.1.1 OPC XML/DA Client

Il client OpView di Proview e' implementato come un volume extern, che viene montato da qualche parte nell'albero degli oggetti del volume di root. Sotto l'oggetto mount, i rami e gli elementi del server vengono visualizzati con oggetti opc speciali. Un oggetto Opc_Hier rappresenta un ramo e Opc_Int un oggetto con un valore intero, Opc_Boolean un oggetto con un valore booleano ecc. Se un oggetto item viene aperto, i valori dell'oggetto vengono visualizzati in un attributo Value e alcune altre proprieta' come description, lowEU, highEU, engineeringUnit, lowIR e highIR vengono visualizzati. Quando l'oggetto viene aperto, viene avviata una sottoscrizione e il valore viene continuamente aggiornato. Per gli oggetti interi e float c'e' anche un grafico dell'oggetto che mostra una tendenza del valore.

Con il client OPC si puo'

- sfoglia i rami e gli oggetti in Xtt e mostra ed imposta i valori degli oggetti.
- sottoscrive i valori degli articoli e visualizzali in un grafico Ge.
- recupera i valori degli elementi nella logica di plc e scrive anche valori sugli elementi.

Il client opc richiede che la navigazione dei nomi sia implementata nel server opc.

Ge

Un valore dell'oggetto puo' essere visualizzato in un grafico Ge usando il nome nel volume esterno. Ad esempio, se l'oggetto di montaggio per il volume extern e' "Ext-P1" e il nome locale dell'elemento e'

```
/P1/Signals/Ai22
```

il nome del segnale in Ge sara' probabilmente (dipende dalla funzione di navigazione del server)

```
Ext-P1-P1-Signals-Ai22.Value##Float32
```

presumendo che sia un tipo di dati float.

Plc

I valori degli oggetti possono anche essere gestiti nel programma plc, usando gli oggetti GetExt ... e CStoExt ....
Gli oggetti normalmente utilizzati per ottenere e archiviare gli attributi GetDp, GetAp, StoDp, StoAp, ecc. Non possono essere utilizzati, poichÃ© richiedono che il nome di riferimento sia noto nell'ambiente di sviluppo, il che non e' il caso per la maggior parte

dei volumi extern. Negli oggetti Ext, il riferimento e' fatto con una stringa di nome, rendendo possibile inserire il nome dell'oggetto. Per ottenere il valore dell'elemento nell'esempio precedente, e' necessario utilizzare un oggetto GetExtFloat32 e il nome dell'oggetto deve essere

```
Ext-P1-P1-Signals-Ai22.Value
```

Per memorizzare un valore in un elemento, diciamo / P1 / Segnali / Ao5, si utilizza un CStoExtFoat32. Questo oggetto crea una memoria condizionale e solo su un fronte positivo della condizione. Confrontare con CStoAp, in cui e' memorizzato il valore, purchÃ© la condizione sia vera. Il nome di riferimento nell'oggetto CStoExtFloat32 in questo caso dovrebbe essere

```
Ext-P1-P1-Signals-Ao5.Value
```

Processo client

Per ogni connessione client-server opc deve essere avviato un processo client. L'eseguibile per questo processo e' opc_provider che ha gli argomenti

1. URL del server Opc
2. ID del volume esterno
3. Nome del volume esterno
4. Identita' del server (opzionale, impostazione predefinita 200)

Configurazione

Registrare ExternVolume

Registrare il volume esterno in GlobalVolumeList con un nome e un'identita' del volume.

File di applicazione

Aggiungi una riga nel file dell'applicazione per avviare opc_provider. Ecco un esempio per un client opc che si connette al server opc 'http://servernode:8080'. L'externvolume registrato ha il nome MyOpcVolume con l'id del volume 0.1.99.55

```
opc_provider, opc_provider, noload, run, opc_provider, 9, nodebug,
   http://servernode:8080 0.1.99.55 MyOpcVolume
```

Se i valori degli articoli vengono recuperati in plc, la priorita' deve essere impostata su 4 (sesto argomento).

Montare l'oggetto

Creare un oggetto di montaggio nella gerarchia dell'impianto del volume principale e inserire l'oggetto objid dell'oggetto volume nell'outternvolume nell'attributo Object. Nell'esempio sopra questo objid e' _O0.1.99.55: 0.

Suggerimento

Il file dell'applicazione risiede su $pwrp_load e ha il nome

```
$pwrp_load/ld_appl_'nodename'_'busnumber'.txt
```

dove nodename e' il nome del nodo e busnumber il numero del bus qcom. Se il nodo e' "mynode" e il numero di bus e' 507, il nome del file sara'

```
$pwrp_load/ld_appl_mynode_507.txt
```

28 Comandi

build	**Costruisci nodo, volume o oggetto**
check classes	**Controlla se qualche classe ha bisogno di aggiornamenti**
close graph	**Chiudi un grafico Ge**
compile	**Compilare plcpgm**
configure card	**Configura un oggetto scheda**
connect	**Collegare il segnale e il canale**
copy	**Copia gli alberi degli oggetti selezionati**
copy object	**Copia un oggetto**
create bootfiles	**Crea file di boot**
create crossreferencefiles	**Creare crossreferencefiles**
create flowfiles	**Creare file di flusso per traccia plc**
create loadfiles	**Crea file di caricamento**
create object	**Crea un oggetto**
create structfiles	**Crea file struct**
create volume	**Crea un volume**
cut	**Taglia oggetti**
define	**Definisci un simbolo**
delete object	**Elimina un oggetto**
delete tree	**Elimina un albero degli oggetti**
delete volume	**Cancella un volume**
disconnect	**Disconnetti segnale e canale**
display	**Mostra una finestra**
distribute	**Distribuire all'operatore o alla stazione di processo**
edit	**Imposta la modalita' di modifica**
exit	**Chiudi wtt**
help	**Mostra aiuto**
generate web	**Genera pagine web**
list channels	**Elenca canali**
list descriptor	**Elenco dal descrittore dell'elenco**
list hierarchy	**Elenca la gerarchia**
list plcpgm	**Elenca plcprogram**
list signals	**Elenca i segnali**
login	**Login utente**
logout	**Disconnessione utente**
move object	**Sposta un oggetto**
new buffer	**Crea un nuovo buffer**
one	**Una finestra**
open buffer	**Apre la finestra di selezione del buffer**
open graph	**Apri un grafico Ge**
paste	**Incolla il buffer**
print	**Stampa plcpgm**
redraw	**Ridisegna plcpgm**
release subwindow	**Continua l'esecuzione con il grafico nell'oggetto finestra.**
revert	**Ripristina sessione**
save	**Salva sessione**
search	**Ricerca**
set advanceduser	**Imposta utente avanzato**

set alltoplevel	**Mostra tutti gli oggetti di livello superiore**
set attribute	**Imposta gli attributi dell'oggetto**
set db	**Imposta il database**
set inputfocus	**Imposta l'input focus sulla finestra**
set showalias	**Mostra il nome dell'alias**
set showattrref	**Mostra i riferimenti agli attributi**
set showattrxref	**Visualizza x-references degli attributi**
set showclass	**Mostra la classe dell'oggetto**
set showdescription	**Mostra descrizione**
set showobjref	**Mostra riferimenti a oggetti**
set showobjxref	**Mostra x-references oggetto**
set subwindow	**Apri un grafico in un oggetto finestra**
set template	**Imposta i valori del modello per gli oggetti**
set verify	**Verifica degli script**
set window	**Imposta la larghezza e l'altezza della finestra**
setup	**Impostazione Wtt**
show children	**Mostra i figli di un oggetto**
show license	**Mostra i termini della licenza**
show object	**Mostra un oggetto**
show objid	**Mostra identita' dell'oggetto**
show script	**Mostra scriptfiles**
show symbol	**Mostra un simbolo**
show user	**Mostra l'utente corrente**
show version	**Mostra la versione di wtt**
show volumes	**Mostra tutti i volumi nel database**
sort	**Ordina i figli di un oggetto**
two	**Due finestre**
update classes	**Aggiorna classi**
wb dump	**Scarica gli oggetti nel file di testo**
wb load	**Carica oggetti dal file di testo**

Symbols

related subjects

script

28.1 Comando build

Chiama il metodo di compilazione per un nodo, un volume o un oggetto.

wtt> build node /name= [/force][/debug][/manual][/crossreference]
wtt> build volume /name= [/force][/debug][/manual][/crossreference]
wtt> build object /name= [/force][/debug][/manual][/crossreference]

/name	Nome del nodo, nome del volume o nome dell'oggetto.
/force	Non controllare le dipendenze, costruisci tutto.
/debug	Costruisci con il debug, cioe' compila e collega con debug.
/manual	Costruisci solo l'oggetto specificato.
/crossreferences	Crea file crossreference. Valido per volumi di costruzione.

28.2 Comando check classes

Controlla se qualche classe ha bisogno di aggiornamenti.

wtt> check classes

28.3 Comando close graph

Chiudi un grafico Ge.

wtt> close graph /file=

/file	Name of the Ge graph.

28.4 Comando compile

Compila i programmi plc.

Se nessuna gerarchia, plcpgm o finestra e' specificata, il plcpgm selezionato sara' compilato

wtt> compile [/debug]
wtt> compile /plcpgm= [/debug]
wtt> compile /window= [/debug]
wtt> compile /hierarchy= [/debug][/modified][/from_plcpgm=]
wtt> compile /volume= [/debug][/modified][/from_plcpgm=]
wtt> compile /allvolumes [/debug][/modified][/from_plcpgm=]

/plcpgm	Nome dell'oggetto plcpgm che verra' compilato.
/window	Nome dell'oggetto plcwindow che verra' compilato.
/hierarchy	Verranno compilati tutti i plcpgm nella gerarchia.
/volume	Nome del volume Verranno compilati tutti i plcpgm nel volume.
/allvolumes	Verranno compilati tutti i plcpgm in tutti i volumi nel database.
/debug	Compilare con il debug.
/modified	Saranno compilati solo i plcwindows modificati.

28.5 Comando configure card

Creare una scheda con i canali.

wtt> configure card /rack= /cardname= /channelname= /chanidentity= /chandescription= /table=

/rack	Nome dell'oggetto rack a cui apparterra' la scheda.
/cardname	Nome della scheda. Ultimo segmento del nome.
/channelname	Nome del canale Ultimo segmento del nome. Un '#' sara' sostituito con il numero del canale. Per esempio /chan=di33## dara' i nomi dei canali di3301, di3302... Se c'e' piu' di un canale il nome del canale deve contenere un segno '#' .
/chanidentity	Identita' del canale Sara' inserito nell'attributo- Identita' del canale.
/chandescripton	Descrizione del canale Sara' inserito nell'attributo- descrizione del canale.

28.6 Comando connect

Connette un segnale e un canale.

wtt> connect /source= /destination= [/reconnect]

/source	Un segnale o un oggetto canale.
/destination	Un segnale o un oggetto canale.
/reconnect	Se la fonte o la destinazione e' gia' connessa sara' prima disconnessa.

28.7 Comando copy

Copia gli alberi degli oggetti selezionati per incollare il buffer.

wtt> copy [/keepreferences] [/ignore_errors]

/keepreferences	Mantieni riferimenti a oggetti al di fuori degli alberi copiati. Di default questi riferimenti verranno azzerati.
/ignore_errors	Prova a completare la copia nonostante gli errori rilevati.

28.8 Comando copy object

Copia un oggetto o un albero di oggetti.

wtt> copy object /source= /destination= /name= [/hierarchy]
[/first] [/last] [/after] [/before]

/source	L'oggetto che verra' copiato.
/destination	Il genitore o fratello all'oggetto creato.
/name	Il nome dell'oggetto creato. Ultimo segmento
/hierarchy	Se l'oggetto di origine ha figli, l'albero figlio sara' anche copiato.
/first	L'oggetto sara' inserito come primo figlio dell'oggetto di destinazione.
/last	L'oggetto verra' inserito come ultimo figlio dell'oggetto di destinazione.
/after	L'oggetto verra' inserito come fratello dopo l'oggetto di destinazione.
/before	L'oggetto verra' inserito come fratello prima dell'oggetto di destinazione.

28.9 Comando create bootfiles

Crea nuovi file di boot.

wtt> create bootfiles /nodeconfig= [/debug]
wtt> create bootfiles /allnodes [/debug]

/nodeconfig	Il nome dell'oggetto NodeConfig del nodo per il quale verra' creato il file nodo.
/all	Crea bootfile per tutti i nodi nel progetto.
/debug	Collega il plcprogram con il debug.

28.10 Comando create crossreferencefiles

Creare file per la visualizzazione di riferimenti incrociati in xtt e rtt per il volume attuale.

wtt> create crossreferencefiles

28.11 Comando create flowfiles

Creare i file di flusso per la traccia di plc.
Il layout delle finestre PLC sono memorizzati in file di flusso e utilizzati
nella traccia di plc.

wtt> create flowfiles /plcpgm=
wtt> create flowfiles /hier=
wtt> create flowfiles /all

Comando per creare file di flusso dal modello plcpgm in un volume di classe
wtt> create flowfiles /template/plcpgm=
wtt> create flowfiles /template/hier=Class

/all	Crea file di flusso per tutti i programmi plc nel volume (non puo' essere usato nei volumi delle classi, usare invece /hier=Class).
/plcpgm	Crea file di flusso per l'oggetto PlcPgm specificato.
/hierarchy	Crea file di flusso per tutti gli oggetti PlcPgm sotto la gerarchia specificata.
/template	Crea file di flusso per programmi PlcTemplate in un volume di classe.

28.12 Comando create loadfiles

Crea loadfile per un volume.

wtt> create loadfile /volume=
wtt> create loadfile [/class] [/all]

/volume	Crea file di caricamento per un volume specifico.
/all	Crea file di caricamento per tutti i volumi di root nel database.
/class	Crea file di caricamento per tutti i volumi di classe nel database.

28.13 Comando create object

Crea un oggetto.

wtt> create object /destination= /name= /class=
[/first] [/last] [/after] [/before]

/destination	La destinazione del nuovo oggetto. La posizione del nuovo oggetto sara' figlio o fratello relativamente all'oggetto di destinazione.
/name	Nome del nuovo oggetto. Ultimo segmento
/class	Classe del nuovo oggetto.
/first	L'oggetto sara' inserito come primo figlio dell'oggetto di destinazione.
/last	L'oggetto verra' inserito come ultimo figlio dell'oggetto di destinazione.
/after	L'oggetto verra' inserito come fratello dopo l'oggetto di destinazione.
/before	L'oggetto verra' inserito come fratello prima dell'oggetto di destinazione.

28.14 Comando create structfiles

Crea C include-files per le classi in un volume di classe.

wtt> create structfiles [/files=]

/files	Nome del file wb_load. Nome predefinito $pwrp_db/userclasses.wb_load

28.15 Comando cut

Copia gli alberi degli oggetti selezionati nel buffer e rimuovere gli oggetti dal volume corrente.

wtt> cut [/keepreferences]

/keepreferences	Mantieni riferimenti a oggetti al di fuori degli alberi copiati. Di default questi riferimenti verranno azzerati.

28.16 Comando define

Definisci un simbolo.

wtt> define 'symbolname' 'text'

argomenti correlati

simbolo
mostra il simbolo
file del simbolo

28.17 Comando delete object

Elimina un oggetto.

wtt> delete object /name= [/noconfirm] [/nolog]

/name	Nome dell'oggetto
/noconfirm	Elimina senza confermare.
/nolog	L'operazione non verra' registrata sul dispositivo di output.

28.18 Comando delete tree

Elimina un albero degli oggetti.

wtt> delete tree /name= [/noconfirm] [/nolog]

/name	L'oggetto radice dell'albero.
/noconfirm	Elimina senza confermare.
/nolog	L'operazione non verra' registrata sul dispositivo di output.

28.19 Comando disconnect

Disconnetti un segnale o un canale.

wtt> disconnect /source=

/source	Un segnale o un oggetto canale.

28.20 Comando display

Visualizza la finestra gerarchia dei nodi o degli impianti (w1 o w2).

wtt> display w1
wtt> display w2

28.21 Comando distribute

Distribuire i file alla stazione operatore o di processo.
Crea un pacchetto di distribuzione, copia il pacchetto
nella stazione e decomprime il pacchetto.

wtt> distribute /node= [/package] [/file=]

/node	Nodo a cui distribuire.
/package	Crea solo pacchetto. Il pacchetto viene creato ma non copiato.
/file	Nome del file del pacchetto. Copia un pacchetto esistente senza creare un nuovo pacchetto.

28.22 Comando edit

Entra o lascia la modalita' di modifica.

wtt> edit
wtt> noedit

28.23 Comand exit

Chiude wtt.

wtt> exit

28.24 Comando help

Mostra le informazioni di aiuto per un argomento.
Le informazioni della guida verranno ricercate in un file della guida. Il file puo' essere il file di aiuto di base, il file di aiuto del progetto o un altro file di aiuto.

Se non viene fornito alcun file di aiuto, il soggetto verra' cercato nei file di aiuto base e di progetto .

wtt> help 'subject'
wtt> help 'subject' /helpfile=

/helpfile	Un file della guida che contiene informazioni sull'oggetto della guida.

argomenti correlati

helpfile

28.25 Comando generate web

Genera file html per pagine Web configurate da oggetti Web nella gerarchia dei nodi del volume corrente.

wtt> generate web

28.26 Comando list

Stampa un elenco di oggetti e attributi.

Gli elenchi verranno inviati a una coda di stampa specificata dal simbolo PWR_FOE_PRINT.

wtt> list descriptor /descriptor=
wtt> list channels [/node=]
wtt> list signals [/hierarchy=]
wtt> list plcpgm [/plcpgm=] [/hierarchy=]
wtt> list hierarchy [/hierarchy=]

28.27 Comando list channels

Elenca schede e canali.

wtt> list channels [/node=] [/volume=] [/allvolumes] [output=]

/node	Oggetto $Node.
/volume	Elenca oggetti in questo volume.
/allvolume	Elenca oggetti in tutti i volumi.
/output	File di uscita. Se viene fornito il file di output, l'elenco non verra' inviato alla stampante.

28.28 Comando list descriptor

Stampa un elenco descritto da un oggetto ListDescriptor.

wtt> list descriptor /descriptor=

/descriptor ListDescriptor object.

28.29 Comando list hierarchy

Elenco degli oggetti PlantHier e NodeHier.

wtt> list hierarchy [/hierarchy=] [/volume=] [/allvolumes] [output=]

/hierarchy	Oggetto gerarchia.
/volume	Elenca oggetti in questo volume.
/allvolume	Elenca oggetti in tutti i volumi.
/output	File di uscita. Se viene fornito il file di output, l'elenco non verra' inviato alla stampante.

28.30 Comando list plcpgm

Elenco di oggetti PlcPgm.

wtt> list plcpgm [/hierarchy=] [plcpgm=] [/volume=] [/allvolumes] [output=]

/plcpgm	Oggetto Plcpgm.
/hierarchy	Oggetto gerarchia.
/volume	Elenca oggetti in questo volume.
/allvolume	Elenca oggetti in tutti i volumi.
/output	File di uscita. Se viene fornito il file di output, l'elenco non verra' inviato alla stampante.

28.31 Comando list signals

Elenco di segnali e riferimenti incrociati ai segnali.

wtt> list signals [/hierarchy=] [/volume=] [/allvolumes] [output=]

/hierarchy	Oggetto gerarchia.
/volume	Elenca oggetti in questo volume.
/allvolume	Elenca oggetti in tutti i volumi.
/output	File di uscita. Se viene fornito il file di output, l'elenco non verra' inviato alla stampante.

28.32 Comando login

Accedi con username e password. I privilegi dell'utente verranno recuperati
dal database dell'utente e influenzeranno l'accesso al sistema.

wtt> login 'username' 'password'

Se si desidera creare o modificare un progetto, un utente o registrare
un volume, si accede come amministratore con il qualificatore
/administrator. Devi specificare un utente nel systemgroup 'system'.
Se questo systemgroup non esiste, nome utente e password non sono richiesti.

wtt> login /administrator 'username' 'password'

argomenti correlati

logout
mostra utente

28.33 Comando logout

Esegui il logout di un utente e torna all'utente originale.

wtt> logout

argomenti correlati

login

28.34 Comando move object

Sposta un oggetto.

wtt> move object /source= /destination= [/rename=] [/first] [/last] [/after] [/before]
wtt> move object /source= /rename=

/source	Nome dell'oggetto da spostare.
/destination	Il genitore o fratello dell'oggetto dopo lo spostamento.
/rename	Nuovo nome oggetto, se il nome dell'oggetto deve essere cambiato. Ultimo segmento Se non viene fornita alcuna destinazione, l'oggetto sara' rinominato, non spostato.
/first	L'oggetto sara' inserito come primo figlio dell'oggetto di destinazione.
/last	L'oggetto verra' inserito come ultimo figlio dell'oggetto di destinazione
/after	L'oggetto verra' inserito come fratello dopo l'oggetto di destinazione
/before	L'oggetto verra' inserito come fratello prima dell'oggetto di destinazione

28.35 Comando new buffer

Crea un nuovo buffer vuoto.

wtt> new buffer /name=

/name Nome del buffer

28.36 Comando one

Mostra una finestra. Viene mantenuta la finestra che attualmente possiede il focus di input.

wtt> one

28.37 Comando open buffer

Aprire la finestra di selezione del buffer.

wtt> open buffer

28.38 Comando open graph

Apri un grafico Ge.
Se e' selezionato modale, l'esecuzione dello script continua
quando il grafico viene chiuso.

wtt> open graph /file= /modal

/file	Nome del grafico Ge.
/modal	Modale

28.39 Comando paste

Incolla l'oggetto dall'ultima copia oppure esegue l'operazione taglia nel volume corrente. Con l'opzione buffer, e' possibile incollare un vecchio buffer.

wtt> paste [/keepoid] [/buffer=]

/keepoid	Conservare le identita' dell'oggetto, se possibile.
/buffer	Nome del buffer che dovrebbe essere incollato. Di default viene usato l'ultimo buffer.
/into	Copia gli oggetti root del buffer paste come figlio dell'oggetto selezionato.
/toplevel	Copia gli oggetti radice del buffer sul livello superiore. Deve essere usato quando si copia su un volume vuoto.

28.40 Comando print

Stampa documenti plc.

wtt> print /plcpgm= [/nodocument] [/nooverview]
wtt> print /hierarchy= [/nodocument] [/nooverview]

/plcpgm	Stampa i documenti di un plcpgm.
/hierarchy	Oggetto gerarchia. Tutti i plc nella gerarchia vengono stampati.
/nodocument	I documenti plc non verranno stampati.
/nooverview	La panoramica della finestra di plc non verra' stampata.
/pdf	Stampa in file pdf.
/all	Stampa tutti i plcpgms.

28.41 Comando redraw

Ridisegna il codice plc.

wtt> redraw /all
wtt> redraw /hierarchy=
wtt> redraw /plcpgm=

/plcpgm	Ridisegna un plcpgm.
/hierarchy	Oggetto gerarchia. Tutti i plc nella gerarchia saranno ridisegnati.
/all	Ridisegna tutti i plcpgms.

28.42 Comando release subwindow

Continua l'esecuzione di uno script che ha aperto un grafico
in un oggetto finestra tramite il comando 'set subwindow' o
la funzione 'SetSubwindow' con modale selezionato.
Il comando di rilascio deve essere eseguito da un pulsante
nel grafico con il comando actiontype.

wtt> release subwindow 'graph'

graph	Nome del grafico principale.

28.43 Comando revert

Ripristina sessione.

wtt> revert

28.44 Comando save

Salva sessione.

wtt> save

28.45 Comando search

Cerca un nome oggetto o una stringa.

wtt> search 'object'
wtt> search /regularexpression 'expression'
wtt> search /next

28.46 Comando set advanceduser

Imposta o resetta utente avanzato.

wtt> set advanceduser
wtt> set noadvanceduser

argomenti correlati

advanced user

28.47 Comando set alltoplevel

Mostra tutti gli oggetti radice nel database, non solo gli oggetti radice definiti per la gerarchia dell'impianto o la gerarchia dei nodi.

wtt> set alltoplevel
wtt> set noalltoplevel

28.48 Comando set attribute

Imposta un valore su un attributo.
Gli oggetti sono selezionati dai qualificatori di nome, classe e gerarchia.

wtt> set attribute /attribute= [/value=] [/name=] [/class=] [/hierarchy=]
[/noconfirm] [/nolog] [/output] [/noterminal]

/attribute	Nome dell'attributo
/value	Valore da inserire nell'attributo. Se non viene fornito alcun valore verra' posta una domanda per ogni oggetto.
/class	Seleziona l'oggetto di questa classe.
/hierarchy	Verranno selezionati solo i successori di questo oggetto.
/noconfirm	Nessuna richiesta di conferma viene emessa.
/nolog	L'operazione non viene registrata sul dispositivo di output.
/output	File di uscita.
/noterminal	Le operazioni non verranno registrate nel terminale.

28.49 Comando set db

Connettiti al database con l'ID fornito.
Questo non ha alcun effetto se un database e' gia' aperto.

wtt> set db /dbid=

/dbid	Identita' del database

28.50 Comando set inputfocus

Impostare l'input focus sull'impianto o sulla finestra della gerarchia dei nodi (w1 o w2).

wtt> set inputfocus w1
wtt> set inputfocus w2

28.51 Comando set showalias

Mostra il nome alias degli oggetti nella gerarchia di impianto e nodo.

wtt> set showalias
wtt> set noshowalias

28.52 Comando set showattrref

Visualizza il numero di referenze dei collegamenti all'attributo
degli oggetti nella gerarchia di impianto e nodo.

wtt> set showattrref
wtt> set noshowattrref

28.53 Comand set showattrxref

Visualizza il numero di x-references degli attributi connessi
degli oggetti nella gerarchia di impianto e nodo.

wtt> set showattrxref
wtt> set noshowattrxref

28.54 Comando set showclass

Mostra la classe dell'oggetto nella gerarchia di impianto e nodo.

wtt> set showclass
wtt> set noshowclass

28.55 Comando set showdescription

Mostra la descrizione degli oggetti nella gerarchia di impianto e nodo.

wtt> set showdescription
wtt> set noshowdescription

28.56 Comando set showobjref

Visualizza il numero di riferimenti oggetto collegati degli oggetti nella gerarchia di impianto e nodo.

wtt> set showobjref
wtt> set noshowobjref

28.57 Comando set showobjxref

Visualizza il numero di x-references degli oggetti connessi degli oggetti nella gerarchia di impianto e nodo.

wtt> set showobjxref
wtt> set noshowobjxref

28.58 Comando set subwindow

Aprire un grafico in un oggetto finestra in un grafico precedentemente
aperto o scambiare un grafico in una cella a vista multipla.

wtt> set subwindow 'graph' /name= /source=
wtt> set subwindow 'multiview-object' /name= /source= [/continue]

/name	Nome dell'oggetto finestra.
/source	Nome del grafico che deve essere aperto nell'oggetto finestra o nella cella di visualizzazione multipla.
/continue	Puo' essere utilizzato se il comando viene eseguito da un pulsante Ge con altre dinamiche da eseguire. Non dovrebbe essere usato se il grafico con il pulsante stesso viene scambiato.

28.59 Comando set template

Impostare i valori di modello per alcuni attributi che influenzano
il layout nel plceditor.

wtt> set template [/signalobjectseg=] [/sigchanconseg=] [/shosigchancon=]
[/shodetecttext=]

/signalobjectseg	Numero di segmenti del nome del segnale che saranno visualizzati negli oggetti 'Get' e 'Set' nel plc-editor.
/sigchanconseg	Numero di segmenti del nome del canale che saranno visualizzati negli oggetti 'Get' e 'Set' nel plc-editor.
/shosigchancon	Mostra il nome del canale negli oggetti 'Get' e 'Set' nel plc-editor.
/shodetecttext	Mostra il testo rilevato negli oggetti ASup e DSup nel plc-editor.

28.60 Comando set verify

Mostra tutte le linee eseguite durante l'esecuzione di uno script.

wtt> set verify
wtt> set noverify

28.61 Comando set window

Imposta la larghezza e l'altezza della finestra.

wtt> set window /width= /height=

/width	larghezza in pixel.
/height	altezza in pixel.

28.62 Comando set volume

set volume e' un comendo obsoleto.

28.63 Configurazione Wtt

Setup delle proprieta' wtt

DefaultDirectory	Directory predefinita per file di comandi.
SymbolFilename	File di simboli
Verify	Verifica l'esecuzione del file di comandi.
AdvancedUser	L'utente e' avanzato.
AllToplevel	Mostra tutti gli oggetti di livello superiore.
Bypass	Ignora alcune restrizioni di modifica.

28.64 Comando show children

Visualizza un oggetto e i figli

wtt> show children /name=

/name Nome dell'oggetto genitore.

28.65 Comando show license

Mostra i termini della licenza.

wtt> show license

28.66 Comando show object

Elenca oggetti.

wt> show object [/name=] [/hierarchy=] [/class=] [/volume=] [/allvolumes]
[/parameter=] [/full] [/output=] [/noterminal]
wt> show object /objid=

/name	Nome dell'oggetto. I caratteri jolly sono supportati.
/hierarchy	Oggetto gerarchia. Solo l'oggetto nella gerarchia sara' selezionato.
/class	Verranno selezionati solo gli oggetti di questa classe.
/volume	Nome del volume
/allvolumes	La ricerca di oggetti verra' eseguita in tutti i volumi.
/parameter	Elenca il valore di un attributo per gli oggetti selezionati.
/full	Mostra il contenuto degli oggetti. Gli attribuiti che differiscono dal valore del modello verranno visualizzati.
/output	File di uscita.
/noterminal	L'output non verra' scritto sul terminale.
/objid	Mostra oggetto per uno specifico objid.

28.67 Comando show objid

Mostra l'objid di un oggetto.
Se il nome viene omesso, viene mostrata l'objid
dell'oggetto selezionato corrente.

wtt> show objid [/name=]

/name	Object name.

28.68 Comando show script

Fornisce un elenco di file di script.
Il carattere jolly asterisco (*) puo' essere utilizzato per cercare i file.

wtt> show script ['scriptspec']

28.69 Comando show symbol

Mostra un simbolo o tutti i simboli

wtt> show symbol 'symbol' **Mostra simbolo 'symbol'**
wtt> show symbol **Mostra tutti i simboli**

argomenti correlati

define
symbol

28.70 Comando show version

Mostra la versione di wtt

wtt> show version

28.71 Comando show volumes

Mostra tutti i volumi nel database.

wtt> show volumes

28.72 Comando sort

Ordina i figli di un oggetto in ordine alfabetico o in ordine di classe.
Se non viene fornito un genitore, i figli degli oggetti selezionati verranno ordinati.

wtt> sort /parent= [/class] [/signals]

/parent	Genitore degli oggetti che verranno ordinati.
/class	Ordina in ordine di classe.
/signals	Ordina gli oggetti signal e plcpgm in ordine di classe, e altri oggetti in ordine alfabetico.

28.73 Comando two

Mostra due finestre. Vengono visualizzate sia la pianta che la finestra gerarchia dei nodi.

wtt> two

28.74 Comando update classes

Aggiorna le classi nel volume allegato.

wtt> update classes

28.75 Comando wb dump

Scarica il volume o una parte del volume in un file di testo.

wtt> wb dump /output= [/hierarchy=]

/hierarchy	Oggetto gerarchia. L'oggetto e il suo albero figlio verra' scritto in un file di testo
/output	File di uscita.
/nofocode	Non scrivere codice plc per oggetti funzionali con codice modello Cio' ridurra' la dimensione del dumpfile. Il nuovo codice verra' copiato quando il plc sara' compilato.
/keepname	Scrivi riferimenti esterni per nome invece che per stringa identita'.
/noindex	Non scrivere indice oggetto nel dumpfile.

28.76 Comando wb load

Carica il database o da wb_load-file o dbs-file.

wtt> wb load /loadfile=

/loadfile	Nome del file Puo' essere di tipo .wb_load, .wb_dmp or .dbs.
/noindex	Ignora gli indici oggetto nel dumpfile e crea nuove identita' oggetto.

28.77 Symbol

Un simbolo wtt puo' essere utilizzato come comando breve o come sostituzione di stringhe in un comando. Se il simbolo viene utilizzato come sostituzione di stringa, il nome-simbolo deve essere racchiuso tra virgolette.

I simboli sono creati con il comando define.
I comandi define possono essere eseguiti dal symbolfile.

Esempio di simbolo utilizzato come comando breve.

```
wtt> define p1 "show child/name=hql-hvk-pumpar-pump1"
wtt> p1
```

Esempio di simbolo utilizzato come sostituzione di stringhe

```
wtt> define p1 hql-hvk-pumpar-StartPump1
wtt> open trace 'p1'
```

argomenti correlati

define
show symbol
symbolfile

29 Wtt script

Esegui script

Tipi di dati e dichiarazioni

Tipi di dati
Conversioni di tipi di dati
Dichiarazioni variabili
Operatori

Dichiarazioni

main-endmain
function-endfunction
if-else-endif
while-endwhile
for-endfor
break
continue
goto
include

Funzioni Input/output

ask()
printf()
say()
scanf()

Funzioni di gestione dei file

fclose()
felement()
fgets()
fopen()
fprintf()
fscanf()
translate_filename()

Funzioni stringa

edit()
element()
extract()
sprintf()
strchr()
strrchr()
strlen()
strstr()
toupper()
tolower()

Funzioni del database

GetAttribute()
GetChild()
GetParent()
GetNextSibling()
GetNextVolume()
GetClassList()
GetNextObject()
GetClassListAttrRef()
GetNextAttrRef()
GetTemplateObject()
GetNextTemplateAttrRef()
GetObjectClass()
GetNodeObject()
GetRootList()
GetVolumeClass()
GetVolumeList()
SetAttribute()
CreateObject()
RenameObject()
MoveObject()
InLib()
OpenPlcPgm()
CreatePlcObject()
ClosePlcObject()
CreatePlcConnection()
SetPlcObjectAttr()
PlcConnect()

Funzioni di sistema

exit()
system()
time()
verify()

Funzioni varie

GetProjectName()
CheckSystemGroup()
CutObjectName()
MessageError()
MessageInfo()
GetCurrentText()
GetCurrentObject()
GetCurrentVolume()
IsW1()
IsW2()
EditMode()
MessageDialog()
ConfirmDialog()
ContinueDialog()
PromptDialog()
OpenGraph()
CloseGraph()
SetSubwindow()

GetIoDeviceData()
SetIoDeviceData()
GetVersion()
get_pwr_config()
get_node_name()
EVEN()
ODD()

wtt-commands

wtt-commands

29.1 Esegui uno script

Un file di script verra' eseguito dalla riga di comando con il comando

wtt> @'filename'

29.2 Tipi di dati

I tipi di dati sono float, int e string.

int	integer value.
float	32-bit float value.
string	80 character string (null terminated).

Esistono tre diverse tabelle in cui e' possibile dichiarare una variabile:
locale, globale ed esterna. Una variabile locale e' nota all'interno di una
funzione, un globale e' noto in tutte le funzioni di un file (con include-files),
un esterno e' noto per tutti i file eseguiti in una sessione.

29.3 Conversioni di tipi di dati

Se un'espressione e' costituita da variabili e funzioni di diversi tipi di dati,
le variabili verranno convertite con la di la seguente precedenza stringa , float, int.
Se due operandi in un'espressione sono di tipo int e float, il risultato sara' float
Se due operandi sono di tipo float e string, o int e string, il risultato sara' string.
In una assegnazione il valore di un'espressione verra' convertito nel tipo della variabile
di assegnazione, anche se il risultato e' una stringa e la variabile e' di tipo float o int.

Esempio

```
string str;
int    i = 35;
str = "Luthor" + i;
```
Il valore in str sara' "Luthor35".

```
float   f;
```

```
string  str = "3.14";
int     i = 159;
f = str + i;
```
Il valore in f sara' 3.14159.

29.4 Dichiarazioni variabili

Una variabile deve essere dichiarata prima di essere utilizzata.
Una dichiarazione consiste di
- la tabella (globale o extern, se locale la tabella e' soppressa)
- il tipo di dati (int, float o stringa)
- il nome della variabile (maiuscole e minuscole)
- Se array, numero di elementi
- segno di uguale seguito da un valore di init, se omesso il valore di init e' zero o Â Â null-stringa
- punto e virgola

Una variabile extern dovrebbe essere cancellata (dall'istruzione delete).

Esempio

```
int             i;
float           flow = 33.4;
string          str = "Hello";
extern int      jakob[20];
global float    ferdinand = 1234;
...
delete jakob[20];
```

29.5 Operatori

Gli operatori hanno la stessa funzione che nel linguaccio C, con alcune limitazioni. Tutti gli operatori non sono implementati. Alcuni operatori (+, =, ==) possono anche operare su variabili stringa. La precedenza degli operatori e' simile a c.

Operator	Description	Datatypes
+	somma	int, float, string
-	sottrazione	int, float
*	moltiplicazione	int, float
/	divisione	int, float
++	incrementa, solo postfix.	int, float
--	decrement, solo postfix	int, float
>>	bits shift-destro	int
<<	bits shift-sinistro	int
<	minore di	int, float
>	magggiore di	int, float
<=	minore uguale	int, float
>=	maggiore uguale	int, float
==	uguale	int, float, string
!=	diverso	int, float, string
&	bit a bit and	int
|	bit a bit or	int

&&	and logico	int
||	or logico	int
!	not logico	int
=	assegnazione	int, float, string
+=	somma e assegna	int, float
-=	sottrai e assegna	int, float
&=	and logico e assegna	int
|=	or logico e assegna	int

29.6 Dichiarazioni di script

main-endmain	Funzione principale.
function-endfunction	Dichiarazione di funzione
if-else-endif	Esecuzione condizionale.
while-endwhile	While loop.
for-endfor	For loop.
break	Termine del loop while o for.
continue	Continua del loop while o for.
goto	Salta all'etichetta.
include	Include di uno script file.

29.6.1 main-endmain

Le istruzioni main e endmain controllano dove inizia e si ferma l'esecuzione
Se non vengono trovate le istruzioni main e endmain, l'esecuzione iniziera'
all'inizio del file e si fermera' alla fine.

Esempio

```
main()
  int a;

  a = p1 + 5;
  printf( "a = %d", a);
endmain
```

29.6.2 function-endfunction

Una dichiarazione di funzione consiste di
- il tipo di dati del valore restituito per la funzione
- il nome della funzione
- un elenco argomenti delimitato da virgola e circondato da parentesi. L'elenco degli argomenti deve includere una dichiarazione di tipo e un nome per ogni argomento.

Gli argomenti forniti dal chiamante verranno convertiti nel tipo del tipo dichiarato nell'elenco degli argomenti. Se un argomento viene modificato all'interno della funzione, il nuovo valore verra' trasferito al chiamante. In questo modo e' possibile restituire altri valori oltre al valore di ritorno della funzione. Una funzione puo' contenere una o piu' istruzioni di ritorno valore. L'istruzione return consegnera' l'esecuzione al chiamante e restituira' il valore fornito.

Esempio

```
function float calculate_flow(float a, float b)
  float c;
  c = a + b;
  return c;
endfunction

...
flow = korr * calculate_flow( v, 35.2);
```

29.6.3 if-else-endif

Le linee tra un'istruzione if-endif saranno eseguite se l'espressione nell'istruzione if e' vera. L'espressione dovrebbe essere circondata da parentesi. Se viene trovata un'altra istruzione tra if e endif, le righe tra else e endif verranno eseguite se if e' false.

Esempio

```
if ( i < 10 && i > 5)
  a = b + c;
endif

if ( i < 10)
  a = b + c;
else
  a = b - c;
endif
```

29.6.4 while-endwhile

Le linee tra un'istruzione while-endwhile verranno eseguite fintanto che l'espressione nell'istruzione while e' vera. L'espressione dovrebbe essere circondata da parentesi.

Esempio

```
while ( i < 10)
  i++;
endwhile
```

29.6.5 for-endfor

Le linee tra un'istruzione for-endfor verranno eseguite fintanto che l'espressione centrale nell'istruzione for e' vera. L'espressione for consiste di tre espressioni, delimitate da punto e virgola e circondate da parentesi. La prima espressione verra' eseguita prima del primo ciclo, la terza verra' eseguita dopo ogni ciclo, la parte centrale viene eseguita prima di ogni ciclo e, se e' vera, viene eseguito un altro ciclo, se falso il ciclo viene lasciato.

Esempio

```
for ( i = 0; i < 10; i++)
  a += b;
endfor
```

29.6.6 break

Un'istruzione break cerchera' la prossima istruzione endwhile o endfor per continuare l'esecuzione alla riga successiva.

Esempio

```
for ( i = 0; i < 10; i++)
  a += b;
  if ( a > 100)
    break;
endfor
```

29.6.7 continue

A continue statement will search for the previous while or for statement continue the loop execution.
Un'istruzione continue cerca la precedente istruzione while o for e continua l'esecuzione del ciclo.

Esempio

```
for ( i = 0; i < 10; i++)
  b = my_function(i);
  if ( b > 100)
    continue;
  a += b;
endfor
```

29.6.8 goto

Un goto fara' saltare l'esecuzione a una riga definita dall'etichetta.
La riga dell'etichetta e' terminata con due punti.

Esempio

```
  b = attribute("MOTOR-ON.ActualValue", sts);
  if (!sts)
    goto some_error;
  ...
some_error:
  say("Something went wrong!");
```

29.6.9 include

Un comando include include file contenente funzioni con l'istruzione #include. L'estensione del file predefinita e' '.pwr_com'

Esempio

```
#include <my_functions>
```

29.7 Funzioni Input/Output

Funzione	Descrizione
ask	Stampa una domanda e legge una risposta.
printf	Stampa formattata
say	Stampa un testo
scanf	Lettura formattata.

29.7.1 ask()

int ask(string question, (arbitrary type) reply)

Descrizione

Richiede l'input con la stringa fornita.
Restituisce il numero di token letti, 1 o 0.

Argomenti

string	question	Richiesta.
arbitrary type	reply	Risposta inserita Puo' essere int, float o string.

Esempio

```
 string reply;

 ask( "Do you want to continue? [y/n] ", reply);
 if ( reply != "y")
   exit();
 endif
```

29.7.2 printf()

int printf(string format [, (arbitrary type) arg1, (arbitrary type) arg2])

Descrizione

Stampa formattata in sintassi-C. Formato argomento e non, uno o due argomenti valore. Restituisce il numero di caratteri stampati.

Argomenti

string	format	Format.
arbitrary type	arg1	Argomento valore. Opzionale. Puo' essere int, float o string.
arbitrary type	arg2	Argomento valore. Opzionale. Puo' essere int, float o string.

Esempio

```
printf( "Watch out!");
printf( "a = %d", a);
printf( "a = %d and str = %s", a, str);
```

29.7.3 say()

int say(string text)

Descrizione

Stampa una stringa.

Argomenti

string	text	Text to print.

Esempio

```
say( "Three quarks for Muster Mark!");
```

29.7.4 scanf()

int scanf(string format , (arbitrary type) arg1)

Descrizione

Input formattato. Sintassi-C.
Restituisce il numero di caratteri letti.

Argomenti

string	format	Format.
arbitrary type	arg1	Argomento valore. Opzionale. Puo' essere int, float o string.

Esempio

```
scanf( "%d", i);
```

29.8 Funzioni Input/Output

Funzione	Descrizione
fclose	Chiude un file
felement	Estrai un elemento dall'ultima riga di lettura.
fgets	Leggi una riga da un file.
fopen	Apri un file.
fprintf	Scrittura formattato su file.
fscanf	Lettura formattata dal file.
translate_filename	Sostituisci le variabili d'ambiente in un nome file.

29.8.1 fclose()

int fclose(int file)

Description

Chiude un file aperto.

Argomenti

int	file	file-id fornto da fopen.

Esempio

```
int infile;
infile = fopen("some_file.txt","r");
...
fclose( infile);
```

29.8.2 felement()

string felement(int number, string delimiter)

Descrizione
Estrae un elemento da una stringa di elementi letti da un file con la funzione fgets(). felement() puo' essere usato a favore di element() quando la stringa di lettura e' piu' grande della dimensione della stringa 256. felement() puo' analizzare linee fino a 1023 caratteri.

Argomenti

int	number	numero degli elementi.
string	delimiter	carattere delimitatore.

Returns

string	The extracted element.

Esempio

```
string elem1;
int file;
string line;

file = fopen( "my_file.txt", "r");
while( fgets( line, file))
  elem1 = felement( 1, " ");
endwhile
```

29.8.3 fgets()

int fgets(string str, int file)

Descrizione

Reads a line from a specified file.
Returns zero if end of file.

Argumento

string	str	Leggi linea. Restituita.
int	file	File restituito da fopen.

Esempio

```
file = fopen("some_file.txt","r");
while( fgets( str, file))
  say( str);
endwhile
fclose( file);
```

29.8.4 fopen()

int fopen(string filespec, string mode)

Descrizione

Apre un file per leggere o scrivere.
Restituisce un identificatore di file. Se il file non puo' essere aperto, viene restituito zero.

Argomenti

string	filespec	Nome del file
string	mode	Modalita' di accesso

Returns

int	Identificatore file o zero su errore.

Esempio

```
int infile;
int outfile;

infile = fopen("some_file.txt","r");
outfile = fopen("another_file.txt","w");
...
fclose( infile);
fclose( outfile);
```

29.8.5 fprintf()

int fprintf(int file, string format [, (arbitrary type) arg1,
 (arbitrary type) arg2])

Descrizione

Stampa formattata su file. Sintassi-C. Formato argomento e non, uno o due argomenti. Restituisce il numero di caratteri stampati.

Argomenti

int	file	ID file restituito da fopen.
string	format	Formato.
arbitrary type	arg1	Valore argomento. Opzionale. Puo' essere int, float o string.
arbitrary type	arg2	Valore argomento. Opzionale. Puo' essere int, float o string.

Esempio

```
int outfile;
outfile = fopen( "my_file.txt", "w");
if (!outfile)
  exit();
fprintf( outfile, "Some text");
fprintf( outfile, "a = %d", a);
fclose( outfile);
```

29.8.6 fscanf()

int fscanf(int file, string format , (arbitrary type) arg1)

Descrizione

Lettura formattata dal file. Sintassi-C.
Restituisce il numero di caratteri letti.

Argomenti

int	file	File id.
string	format	Formato.
arbitrary type	arg1	Valore argomento. Restituito. Puo' essere int, float o string.

Esempio

```
int file;
int i;

file = fopen( "my_file.txt", "r");
if (file)
  fscanf( file, "%d", i);
  fclose( file);
endif
```

29.8.7 translate_filename()

string translate_filename(string fname)

Descrizione

Sostituisci le variabili d'ambiente nel nome del file.

Argomenti

string	fname	Un nome file.

Restituisce

string	Stringa con variabili env espanse.

Esempio

```
string fname1 = "$pwrp_db/a.wb_load";
string fname2;
fname2 = translate_filename( fname1);
```

29.9 Funzioni String

Funzione	Descrizione
edit	Rimuovi spazi e tabulati superflui.
element	Estrai un elemento da una stringa.
extract	Estrai una sottostringa da una stringa.
sprintf	Stampa formattata su una stringa.
strchr	Restituisce la prima occorrenza di un carattere in una stringa.
strrchr	Restituisce l'ultima occorrenza di un carattere in una stringa.
strlen	Calcola la lunghezza di una stringa.
strstr	Restituisce la prima occorrenza di una sottostringa in una stringa.
tolower	Converti stringa in minuscolo.
toupper	Converti la stringa in maiuscolo.

29.9.1 edit()

string edit(string str)

Descrizione

Rimuove gli spazi iniziali e finali e e sostituisce piu' tabulatori e spazi con un singolo spazio. Restituisce la stringa modificata.

Argomenti

string	str	stringa da editare.

Esempio

```
collapsed_str = edit(str);
```

29.9.2 element()

string element(int number, string delimiter, string str)

Descrizione

Estrae un elemento da una stringa di elementi.
Restituisce l'elemento estratto.

Argomenti

int	number	il numero dell'elemento.
string	delimiter	carattere delimitatore.
string	str	stringa di elementi.

Esempio

```
string str = "mary, lisa, anna, john";
string elem1;
elem1 = element( 1, ",", str);
```

29.9.3 extract()

string extract(int start, int length, string str)

Descrizione

Estrae i caratteri specificati dalla stringa specificata.
Restituisce i caratteri estratti come una stringa.

Argomenti

int	start	Posizione iniziale del primo carattere. Il primo carattere ha la posizione 1.
int	length	numero di caratteri da estrarre.
string	str	stringa da cui estrarre i caratteri.

Esempio

```
extracted_str = extract( 5, 7, str);
```

29.9.4 sprintf()

int sprintf(string str, string format [, (arbitrary type) arg1, (arbitrary type) arg2])

Descrizione

Stampa formattata da buffer. Sintassi-C. Formato argomento e non, uno o due argomenti. Restituisce il numero di caratteri stampati.

Argomenti

string	str	Stringa su cui stampare.
string	format	Formato.
arbitrary type	arg1	Valore argomento. Optionale. Puo essere int, float o string.
arbitrary type	arg2	Valore argomento. Optionale. Puo essere int, float o string.

Esempio

```
string str;
int items;

sprintf( str, "Number of items: %d", items);
```

29.9.5 strchr()

int strchr(string str, string c)

Descrizione

Restituisce la prima occorrenza di un carattere in una stringa.

Argomenti

string	str	Stringa in cui cercare
string	c	Carattere da cercare.

Restituisce

int	Indice per la prima occorrenza di carattere. Il primo carattere ha indice 1. Restituisce zero se il carattere non viene trovato.

Esempio

```
string str = "index.html";
int idx;

idx = strchr( str, ".");
```

29.9.6 strrchr()

int strrchr(string str, string c)

Descrizione

Restituisce l'ultima occorrenza di un carattere in una stringa.

Argomenti

string	str	Stringa in cui cercare
string	c	Carattere da cercare.

Returns

int	Indice per l'ultima occorrenza di carattere. Il primo carattere ha indice 1. Restituisce zero se il carattere non viene trovato.

Esempio

```
string str = "/usr/local/pwrrt";
int idx;

idx = strrchr( str, "/");
```

29.9.7 strlen()

int strlen(string str, string c)

Descrizione

Calcola la lunghezza di una stringa.

Argomenti

string	str	Stringa per calcolare la lunghezza per.

Restituisce

int	Lunghezza della stringa.

Esempio

```
string str = "/usr/local/pwrrt";
int len;

len = strlen( str);
```

29.9.8 strchr()

int strstr(string str, string substr)

Descrizione

Restituisce la prima occorrenza di una sottostringa in una stringa.

Argomenti

string	str	Stringa in cui cercare
string	substr	Sotto stringa da cercare.

Returns

int	Indice per l'ultima occorrenza sotto stringa. Il primo carattere ha indice 1. Restituisce zero se la sotto stringa non viene trovata.

Esempio

```
string str = "index.html";
int idx;

idx = strstr( str, ".html");
```

29.9.9 toupper()

string toupper(string str)

Descrizione

Converti la stringa in maiuscolo.

Argomenti

string	str	Stringa da convertire.

Restituisce

string	Stringa in maiuscolo.

Esempio

```
string str1 = "Buster Wilson";
string str2;
str2 = toupper( str);
```

29.9.10 tolower()

string tolower(string str)

Descrizione

Converti la stringa in minuscolo.

Argomenti

string	str	Stringa da convertire.

Returns

string	Stringa in minuscolo.

Esempio

```
string str1 = "Buster Wilson";
string str2;
str2 = tolower( str);
```

29.10 Funzioni di sistema

Funzione	Descrizione
exit	Esce dallo script.
system	Esegue comandi di shell.
time	Ottiene l'orqa si sistema.
verify	Stampa le linee eseguite.

29.10.1 exit()

int exit()

Descrizione

Termona l'esecuzione del file.

Esempio

```
exit();
```

29.10.2 system()

int system(string cmd)

Descrizione

Esegue un comando di shell.

Argomenti

string	cmd	Comando di Shell da eseguire.

Returns

int	Il valore di ritorno e' -1 su errore o restituisce lo stato del comando in caso contrario.

Esempio

```
string cmd;

cmd = "firefox http://www.proview.se";
system( cmd);
```

29.10.3 time()

string time()

Descrizione

Restituisce l'ora corrente nel formato stringa.

Eaempio

```
string t;
t = time();
```

29.10.4 verify()

int verify([int mode])

Descrizione

Imposta o mostra la modalita' di verifica. Se la verifica e' attiva,
tutte le linee eseguite verranno visualizzate sullo schermo.
Restituisce la modalita' di verifica corrente.

Argomenti

int	mode	verifica attiva (1) o disattivata (0). Opzionale.

Esempio

```
verify(1);
```

29.11 Funzioni Database

Funzione	Descrizione
GetAttribute()	Ottieni il valore dell'attributo.
GetChild()	Ottieni oggetto figlio.
GetParent()	Ottieni oggetto padre.
GetNextSibling()	Ottieni oggetto fratello.
GetNextVolume()	Ottieni il prossimo volume.
GetClassList()	Ottieni la prima istanza di una classe.
GetNextObject()	Ottieni la prossima istanza di una classe.
GetClassListAttrRef()	Ottieni la prima istanza di una classe, inclusi oggetti attributi.
GetNextAttrRef()	Ottieni l'istanza successiva di una classe, inclusi oggetti attributi.
GetTemplateObject()	Ottieni l'oggetto modello per una classe.
GetNextTemplateAttrRef()	Ottieni l'istanza successiva in un oggetto modello.
GetObjectClass()	Ottieni la classe di un oggetto.
GetNodeObject()	Ottieni oggetto nodo.
GetRootList()	Ottieni il primo oggetto nell'elenco radice.
GetVolumeClass()	Ottenere classe di un volume.
GetVolumeList()	Ottieni il primo volume.
SetAttribute()	Imposta il valore dell'attributo.
CreateObject()	Crea un oggetto.
RenameObject()	Cambia il nome di un oggetto.
MoveObject()	Sposta un oggetto.
InLib()	Controlla se un oggetto e' in un $LibHier.
OpenPlcPgm()	Apri un PlcPgm.
ClosePlcPgm()	Chiudi un PlcPgm.
CreatePlcObject()	Crea un oggetto plc.
CreatePlcConnection()	Crea una connessione PLC.
SetPlcObjectAttr()	Imposta l'attributo in un oggetto plc.
PlcConnect()	Connetti un oggetto plc.

29.11.1 GetAttribute()

(variable type) GetAttribute(string name [, int status])

Descrizione

Ottieni il valore dell'attributo specificato. Il tipo restituito dipende
dal tipo di attributo. L'attributo verra' convertito in int, float o stringa.

Argomenti

string	name	nome dell'attributo da prelevare.
int	status	stato di funzionamento. Restituito. Se zero, l'attributo non puÃ² essere recuperato. Opzionale.

Esempio

```
int alarm;
int sts;

alarm = GetAttribute("Roller-Motor-Alarm.ActualValue");
on = GetAttribute("Roller-Motor-On.ActualValue", sts);
if ( !sts)
  say("Could not find motor on attribute!");
```

29.11.2 GetChild()

string GetChild(string name)

Descrizione

Ottiene il primo figlio di un oggetto. I prossimi figli possono essere recuperati con GetNextSibling(). Restituisce il nome del figlio.
Se non esiste alcun figlio, viene restituita una stringa nulla.

Argomenti

string	name	nome dell'oggetto.

Esempio

```
string child;

child = GetChild("Roller-Motor");
```

29.11.3 GetParent()

string GetParent(string name)

Descrizione

Ottieni il genitore di un oggetto.
Restituisce il nome del genitore. Se non esiste un genitore,
viene restituita una stringa nulla.

Argomenti

string	name	nome dell'oggetto.

Esempio

```
string parent;

parent = GetChild("Roller-Motor");
```

29.11.4 GetNextSibling()

string GetNextSibling(string name)

Descrizione

Ottieni il prossimo fratello di un oggetto.
Restituisce il nome del fratello. Se nessun fratello successivo
esiste, viene restituita una stringa nulla.

Argomenti

string	name	nome dell'oggetto.

Esempio

```
string name;
int not_first;

name = GetChild("Rt");
not_first = 0;
while ( name != "")
  if ( !not_first)
    create menu/title="The Rt objects"/text="'name'"/object="'name'"
  else
    add menu/text="'name'"/object="'name'"
  endif
  not_first = 1;
  name = GetNextSibling(nname);
endwhile
if ( !not_first )
  MessageError("No objects found");
```

29.11.5 GetClassList()

string GetClassList(string class)

Descrizione

Ottieni il primo oggetto di una classe specificata. Il prossimo oggetto della classe puÃ² essere recuperato con GetNextObject (). Restituisce il nome del primo oggetto. Se non esiste alcuna istanza della classe, viene restituita una stringa nulla.

Argomenti

string	name	nome della classe.

Esempio

```
string name;

name = GetClassList("Dv");
```

29.11.6 GetNextObject()

string GetNextObject(string name)

Descrizione

Ottieni l'oggetto successivo in una lista di classe.
Restituisce il nome dell'oggetto. Se non esiste alcun oggetto successivo, viene restituita una stringa nulla.

Argomenti

string	name	nome dell'oggetto.

Esempio

```
string name;

name = GetClassList("Di");
while ( name != "")
  printf("Di object found: %s", name);
  name = GetNextObject(name);
endwhile
```

29.11.7 GetClassListAttrRef()

string GetClassListAttrRef(string class)

Descrizione

Ottiene il primo oggetto o attributo oggetto di una classe specificata.
Il prossimo oggetto o attributo oggetto della classe puÃ² essere recuperato con GetNextAttrRef(). Restituisce il nome del primo oggetto.
Se non esistono istanze o oggetti attributo della classe, viene restituita una stringa nulla.

Argomenti

string	name	nome della classe.

Esempio

```
string name;

name = GetClassListAttrRef("Dv");
```

29.11.8 GetNextAttrRef()

string GetNextAttrRef(string class, string name)

Descrizione

Ottiene il prossimo oggetto o oggetto attributo in una classlist.
Restituisce il nome dell'oggetto o dell'oggetto attributo. Se non
esiste alcun oggetto successivo, viene restituita una stringa nulla.

Argomenti

string	class	nome della classe.
string	name	nome dell'oggetto o dell'oggetto dell'attributo.

Esempio

```
string name;

name = GetClassListAttrRef("Di");
while ( name != "")
  printf("Di object found: %s", name);
  name = GetNextAttrRef( "Di", name);
endwhile
```

29.11.9 GetTemplateObject()

string GetTemplateObject(string class)

Descrizione

Ottieni l'oggetto modello di una classe specificata.

Argomenti

string	name	nome della classe.

Esempio

```
string name;

name = GetTemplateObject("Dv");
```

29.11.10 GetNextTemplateAttrRef()

string GetNextTemplateAttrRef(string class, string name)

Descrizione

Ottieni il prossimo oggetto attributo della classe specificata in un oggetto modello. Restituisce il nome dell'oggetto attributo. Se non esiste alcun oggetto successivo, viene restituita una stringa nulla.
Il primo oggetto modello viene recuperato con GetTemplateObject ().

Argomenti

string	class	nome della classe.
string	name	nome dell'oggetto modello o dell'oggetto attributo modello.

Esempio

```
string name;

name = GetTemplateObject("Di");
while ( name != "")
  printf("Di object found: %s", name);
  name = GetNextTemplateAttrRef( "Di", name);
endwhile
```

29.11.11 GetObjectClass()

string GetObjectClass(string name)

Descrizione

Ottieni la classe di un oggetto.
Restituisce il nome della classe.

Argomenti

string	name	nome dell'oggetto.

Esempio

```
string class;

class = GetObjectClass("Motor-Enable");
```

29.11.12 GetNodeObject()

string GetNodeObject()

Descrizione

Ottieni l'oggetto nodo.
Restituisce il nome dell'oggetto nodo.

Esempio

```
string node;
node = GetNodeObject();
```

29.11.13 GetRootList()

string GetRootList()

Descrizione

Ottieni il primo oggetto nell'elenco radice.
Restituisce il nome dell'oggetto root. L'oggetto successivo
nell'elenco radice puÃ² essere recuperato con GetNextSibling ().

Esempio

```
string name;

name = GetRootList();
while( name != "")
  printf( "Root object found: %s", name);
  name = GetNextSibling(name);
endwhile
```

29.11.14 GetNextVolume()

string GetNextVolume(string name)

Descrizione

Ottieni il prossimo volume. Il primo volume viene recuperato widh GetVolumeList(). Restituisce il nome del volume. Se non c'e' un volume successivo, viene restituita una stringa nulla.

Argomenti

string	name	nome del volume.

29.11.15 GetVolumeClass()

string GetVolumeClass(string name)

Descrizione

Ottiene la classe di un volume.
Restituisce il nome della classe.

Argomenti

string	name	nome del volume.

Esempio

```
string class;

class = GetVolumeClass("CVolVKVDKR");
```

29.11.16 GetVolumeList()

string GetVolumeList()

Descrizione

Ottieni il primo volume nella volumelist.
Restituisce il nome del volume. Il prossimo volume verra' recuperato con GetNextVolume().

Esempio

```
string name;

name = GetVolumeList();
while( name != "")
  printf( "Volume found: %s", name);
  name = GetNextVolume(name);
endwhile
```

29.11.17 SetAttribute()

int SetAttribute(string name, (arbitrary type) value)

Descrizione

Imposta il valore di un attributo.
L'attributo e' specificato con l'oggetto completo
ed il nome dell'attributo.
Restituisce lo stato dell'operazione.

Argomenti

string	name	nome attributo.
arbitrary type	value	valore attributo.

Esempio

```
SetAttribute( "Pump-V1-Switch.Description", "Valve switch open");
```

29.11.18 CreateObject()

int CreateObject(string name, string class, string destination, int destcode)

Descrizione

Crea un oggetto.
Restituisce lo stato dell'operazione.

Argument

string	name	nome dell'oggetto. Senza percorso
string	class	classe dell'oggetto.
string	destination	oggetto di destinazione. Un padre o un fratello all'oggetto.
int	destcode	codice di destinazione 1 primo figlio, 2 ultimo figlio, 3 dopo, 4 prima.

Esempio

```
CreateObject( "Temperature", "BaseTempSensor", "Pump-V1", 2);
```

29.11.19 RenameObject()

int RenameObject(string name, string newname)

Descrizione

Cambia il nome di un oggetto.
Restituisce lo stato dell'operazione.

Argomenti

string	name	vecchio nome con percorso.
string	newname	nuovo nome senza percorso.

Esempio

```
RenameObject( "H1-Zon1-Temp2", "PT2");
```

29.11.20 MoveObject()

int MoveObject(string name, string destination, int destcode)

Descrizione

Sposta un oggetto.
Restituisce lo stato dell'operazione.

Argomenti

string	name	nome con percorso.
string	destination	oggetto di destinazione. Un padre o un fratello all'oggetto nella nuova posizione.
int	destcode	codice di destinazione 1 primo figlio, 2 ultimo figlio, 3 dopo, 4 prima.

Esempio

```
MoveObject( "H1-Zon1-Temp2", "H1-Zon2", 1);
```

29.11.21 InLib()

int InLib(string name)

Descrizione

Controlla se un oggetto si trova in una gerarchia di librerie.
Restituisce 1 se si trova in una gerarchia di librerie, altrimenti 0.

Argument

string	name	nome oggetto con percorso.

Returns

int	1 se l'oggetto si trova in una libreria gerarchia, altrimenti 0.

Esempio

```
if ( !InLib( "H1-Motor"))
  ...
endif
```

29.11.22 OpenPlcPgm()

int OpenPlcPgm(string name)

Descrizione

Aprire un programma plc e una sessione di modifica di plc.
Restituisce lo stato dell'operazione.
Non dovrebbero esserci operazioni non salvate quando viene
chiamata questa funzione.
Solo un programma puÃ² essere aperto contemporaneamente.

La sessione di editing di plc dovrebbe essere chiusa con una
chiamata a ClosePlcPgm(). Nella sessione di modifica del plc
devono essere utilizzate solo le funzioni plc CreatePlcObject(),
CreatePlcConnection(), SetPlcObjectAttr() e PlcConnect() per
manipolare e creare oggetti. L'accesso per la sessione precedente
e' temporaneo impostato su readonly quando la modifica di plc e' attiva.

Argomenti

string	name	nome dell'oggetto PlcPgm.

Esempio

```
OpenPlcPgm( "Pump-V1-Control");
...
ClosePlcPgm();
```

29.11.23 ClosePlcPgm()

int ClosePlcPgm()

Descrizione

Chiude una sessione di editing di plc.

Esempio

```
OpenPlcPgm( "Pump-V1-Control");
...
ClosePlcPgm();
```

29.11.24 CreatePlcObject()

int CreatePlcObject(string name, string class, float x, float y [, string destination,
int inputmask, int outputmask, int invertmask])

Descrizione

Crea un oggetto funzione plc in una sessione di modifica di plc.
La funzione puÃ² essere utilizzata solo in una sessione di modifica di plc
avviata con una chiamata a OpenPlcPgm().

L'oggetto e' posizionato sulle coordinate x e y. Se la destinazione, un
oggetto documento, viene fornita, le coordinate sono relative a questo
oggetto, altrimenti sono assolute.
Se le maschere vengono lasciate fuori vengono utilizzate le maschere di default.

Argomenti

string	name	Nome dell'oggetto Senza percorso
string	class	Classe dell'oggetto
float	x	coordinate x .
float	y	coordinate y .
string	destination	Opzionale. Un oggetto documento. Se fornite le coordinate sono relative a questo oggetto.
int	inputmask	Opzionale. Maschera in cui i bit indicano i pin di input visibili nel blocco funzione.
int	outputmask	Opzionale. Maschera in cui i bit indicano i pin di output visibili nel blocco funzione.
int	invmask	Opzionale. Maschera in cui i bit indicano i pin di input invertiti.

Esempio

```
OpenPlcPgm( "Pump-V1-Control");
CreatePlcObject( "Document0", "Document", 1.2, 0.0);
CreatePlcObject( "And0", "And", 0.3, 0.1, "Document0", 15, 1, 3);
CreatePlcObject( "V1", "BaseCValve", 0.3, 0.4, "Document0");
ClosePlcPgm();
```

29.11.25 CreatePlcConnection()

int CreatePlcConnection(string source, string sourceattr, string dest, string destattr [,
int feedback])

Descrizione

Crea una connessione PLC tra l'oggetto funzione sorgente e l'oggetto funzione destinazione.

La funzione puÃ² essere utilizzata solo in una sessione di modifica di plc avviata con una chiamata a OpenPlcPgm ().

I pin a cui connettersi sono specificati dai nomi degli attributi dei pin. Se deve essere creata una connessione di feedback tratteggiata, questo puÃ² essere indicato con l'argomento di feedback opzionale.

Argomenti

string	source	Nome dell'oggetto di origine. Senza percorso
string	sourceattr	Attributo per il pin sull'oggetto sorgente.
string	dest	Nome dell'oggetto di destinazione. Senza percorso
string	destattr	Attributo per il pin sull'oggetto di destinazione.
int	feedback	Opzionale. Se 1 viene creata una connessione di feedback tratteggiata.

Esempio

```
OpenPlcPgm( "Pump-V1-Control");
...
CreatePlcObject( "And0", "And", 0.3, 0.1, "Document0", 15, 1, 3);
CreatePlcObject( "And1", "And", 0.6, 0.1, "Document0", 15, 1, 3);
CreatePlcConnection( "And0", "Status", "And1", "In1");
# Feedback connection
CreatePlcConnection( "And1", "Status", "And0", "In4", 1);
ClosePlcPgm();
```

29.11.26 PlcConnect()

int PlcConnect(string plcobject, string connectobject)

Descrizione

Attiva la funzione di connessione PLC, ad esempio per collegare oggetti funzione Get e Sto a segnali e attributi nella gerarchia dell'impianto.

La funzione puÃ² essere utilizzata solo in una sessione di modifica di plc avviata con una chiamata a OpenPlcPgm().

Argomenti

string	plcobject	Nome, senza percorso, dell'oggetto funzione plc che dovrebbe essere connesso.
string	connectobject	Nome dell'oggetto o attributo a cui connettersi. Nome completo con percorso.

Esempio

```
OpenPlcPgm( "Pump-V1-Control");
...
CreatePlcObject( "GetDv0", "GetDv", 0.3, 0.1, "Document0");
PlcConnect( "GetDv0", "Pump-V1-Active");
...
ClosePlcPgm();
```

29.11.27 SetPlcObjectAttr()

int SetPlcObjectAttr(string attribute, (arbitrary type) value)

Descrizione

Imposta un valore in un oggetto plc.

La funzione puÃ² essere utilizzata solo in una sessione di modifica di plc avviata con una chiamata a OpenPlcPgm().

Argomenti

string	name	nome attributo. Senza percorso
arbitrary type	value	valore dell'attributo.

Esempio

```
OpenPlcPgm( "Pump-V1-Control");
CreatePlcObject( "Document0", "Document", 1.3, 0.0);
SetPlcObjectAttr( "Document0.DocumentOrientation", 1);
SetPlcObjectAttr( "Document0.DocumentSize", 3);
...
ClosePlcPgm();
```

29.12 Funzioni varie

Funzione	Descrizione
GetProjectName()	Ottieni il nome del progetto.
CheckSystemGroup()	Verifica che il gruppo di sistema esista.
CutObjectName()	Tagliare il nome di un oggetto.
MessageError()	Stampa un messaggio di errore.
MessageInfo()	Stampa un messaggio informativo.
GetCurrentText()	Ottieni il testo dell'articolo corrente.
GetCurrentObject()	Ottieni l'oggetto associato all'articolo corrente.
GetCurrentVolume()	Ottieni il volume allegato.
IsW1()	Controlla se la gerarchia dell'impianto ha il focus.
IsW2()	Controlla se la gerarchia del nodo ha il focus.
EditMode()	Controlla se in modalita' modifica.
MessageDialog()	Apri una finestra di dialogo.
ConfirmDialog()	Apri una finestra di conferma.
ContinueDialog()	Apri una finestra di dialogo Continua / Esci.
PromptDialog()	Apri una finestra di dialogo di input.
OpenGraph()	Apri un grafico Ge.
CloseGraph()	Chiudi un grafico Ge.
SetSubwindow()	Apri un grafico in un oggetto finestra.
GetIoDeviceData()	Ottieni dati dalla configurazione IO.
SetIoDeviceData()	Imposta i dati nella configurazione IO.
GetVersion()	Ottieni la versione di Proview.
get_pwr_config()	Ottieni valori di configurazione.
get_node_name()	Ottieni il nome del nodo.
EVEN()	Controlla se il valore e' pari.
ODD()	Controlla se il valore e' dispari.

29.12.1 GetProjectName()

string GetProjectName()

Descrizione

Ottieni il nome del progetto.
Restituisce il nome del progetto.

Esempio

```
string name;

name = GetProjectName();
```

29.12.2 CheckSystemGroup()

int CheckSystemGroup()

Descrizione

Controlla se esiste un system group.
Restituisce 1 se il gruppo di sistema esiste, altrimenti 0.

Esempio

```
if ( !CheckSystemGroup( "MyGroup"))
  return;
endif
```

29.12.3 CutObjectName()

string CutObjectName(string name, int segments)

Descrizione

Taglia i primi segmenti di un nome oggetto.
Restituisce gli ultimi segmenti di un nome oggetto. Il numero di segmenti rimanenti e' specificato dal secondo argomento

Argomenti

string	name	Nome del percorso dell'oggetto.
int	segments	Numero di segmenti che dovrebbero essere lasciati.

Esempio

```
string path_name;
string object_name;

path_name = GetChild("Rt-Motor");
object_name = CutObjectName( path_name, 1);
```

29.12.4 MessageError()

string MessageError(string message)

Descrizione

Stampa un messaggio di errore sullo schermo.

Esempio

```
MessageError("Something went wrong");
```

29.12.5 MessageInfo()

string MessageInfo(string message)

Descrizione

Stampa un messaggio informativo rtt sullo schermo.

Esempio

```
MessageInfo("Everything is all right so far");
```

29.12.6 GetCurrentText()

string GetCurrentText()

Descrizione

Ottieni il testo della voce di menu corrente o del campo di aggiornamento.

Esempio

```
string text;

text = GetCurrentText();
```

29.12.7 GetCurrentObject()

string GetCurrentObject()

Descrizione

Ottieni l'oggetto associato alla voce di menu corrente.
Se nessun oggetto e' associato, viene restituita una stringa nulla.

Esempio

```
string object;

object = GetCurrentObject();
```

29.12.8 GetCurrentVolume()

string GetCurrentVolume()

Descrizione

Ottieni il volume allegato.
Se non e' associato alcun volume, viene restituita una stringa nulla.

Esempio

```
string current_volume;

current_volume = GetCurrentVolume();
set volume/volume=SomeOtherVolume
...
set volume/volume='current_volume'
```

29.12.9 IsW1()

int IsW1()

Descrizione

Restituisce 1 se la finestra corrente focalizzata in wtt e' la finestra Gerarchia impianto.
Altrimenti restituisce 0.

29.12.10 IsW2()

int IsW2()

Descrizione

Restituisce 1 se la finestra corrente focalizzata in wtt e' la finestra della gerarchia del nodo. Altrimenti restituisce 0.

29.12.11 EditMode()

int EditMode()

Descrizione

Restituisce 1 se wtt e' in modalita' di modifica.
Altrimenti restituisce 0.

29.12.12 MessageDialog()

MessageDialog(string title, string text)

Descrizione

Mostra una finestra di dialogo.

Argomenti

string	title	Titolo.
string	text	Testo del Messaggio.

Esempio

```
MessageDialog( "Message", "This is a message");
```

29.12.13 ConfirmDialog()

int ConfirmDialog(string title, string text [, int cancel])

Descrizione

Mostra una finestra di dialogo di conferma.
Restituisce 1 se si preme il pulsante si, 0 se il pulsante di no e' premuto.
Se viene aggiunto il terzo argomento (annulla), viene visualizzato un pulsante di annullamento.
Se il pulsante di cancellazione e' stato premuto o se la finestra di dialogo e' chiusa, l'argomento cancel e' impostato su 1.

Argomenti

string	title	Titolo.
string	text	Testo da confermare
int	cancel	Opzionale. Viene visualizzato un pulsante Annulla. Annulla e' impostato su 1 se il pulsante Annulla e' premuto, o se la finestra di dialogo e' stata chiusa.

Esempio 1

```
if ( ! ConfirmDialog( "Confirm", "Do you really want to..."))
  printf( "Yes is pressed\ ");
else
  printf( "No is pressed\ ");
endif
```

Esempio 2

```
int cancel;
int sts;

sts =  ConfirmDialog( "Confirm", "Do you really want to...", cancel);
if ( cancel)
  printf("Cancel is pressed\ );
  exit();
endif

if ( sts)
  printf( "Yes is pressed\ ");
else
  printf( "No is pressed\ ");
endif
```

29.12.14 ContinueDialog()

ContinueDialog(string title, string text)

Descrizione

Visualizza una finestra di dialogo con i pulsanti "Continua" e "Esci".
Restituisce 1 se viene premuto continua, 0 se viene premuto quit.

Argomenti

string	title	Titolo.
string	text	Testo del messaggio.

Esempio

```
if ( ! ContinueDialog( "Message", "This script will...");
  exit();
endif
```

29.12.15 PromptDialog()

int PromptDialog(string title, string text, string value)

Descrizione

Mostra una finestra di dialogo di prompt che richiede un valore di input.
Restituisce 1 se si preme il pulsante si', 0 se si preme il pulsante
Annulla, o se la finestra di dialogo e' chiusa.

Argomenti

string	title	Titolo.
string	text	Valore del testo.
string	value	Contiene il valore inserito.

Esempio

```
string name;

if ( PromptDialog( "Name", "Enter name", name))
  printf( "Name : '%s'\ ", name);
else
  printf( "Cancel...\ ");
endif
```

29.12.16 OpenGraph()

int OpenGraph(string name, int modal)

Descrizione

Apri un grafico Ge.
Se e' selezionato modale, l'esecuzione dello script continua
quando il grafico e' chiuso.

Argomenti

string	name	Graph name.
int	modal	Modale.

Esempio

```
  OpenGraph( "pwr_wizard_frame", 0);
```

29.12.17 CloseGraph()

int CloseGraph(string name)

Descrizione

Chiudi un grafico Ge.

Argomenti

string	name	Nome del grafico.

Esempio

```
CloseGraph( "pwr_wizard_frame");
```

29.12.18 SetSubwindow()

int SetSubwindow(string name, string windowname, string source, int modal)

Descrizione

Aprire un grafico Ge in un oggetto finestra in un grafico aperto in precedenza.
Se e' selezionato modal, l'esecuzione dello script viene continuata quando il comando 'release subwindow' viene eseguito da un pulsante nel grafico.

Argument

string	name	Nome del grafico principale.
string	windowname	Nome dell'oggetto finestra in cui il grafico sorgente deve essere aperto.
string	source	Nome del grafico che deve essere aperto nell'oggetto della finestra.
int	modal	Modale.

Esempio

```
SetSubwindow( "pwr_wizard_frame", "Window1", "MyGraph", 1);
```

29.12.19 GetIoDeviceData()

int GetIoDeviceData(string object, string parameter, string value)

Descrizione

Ottieni dati dalla configurazione IO.
Restituisce lo stato dell'operazione, dispari se ha successo e pari per l'errore.

Questo comando utilizza metodi e deve essere eseguito in pwrs. Non funziona in pwrc.

I parametri supportati per i dispositivi Profinet sono

NetworkSettings-DeviceName
NetworkSettings-IP Address
NetworkSettings-Subnet Mask
NetworkSettings-MAC Address
NetworkSettings-SendClock
NetworkSettings-ReductionRatio
NetworkSettings-Phase
NetworkSettings-API

Argomenti

string	object	Nome oggetto dispositivo.
string	parameter	Nome del parametro per i dati.
string	value	Valore del parametro restituito.

Esempio

```
sts = SetIoDeviceData( "Nodes-MyNode-PN-D1", "NetworkSettings-DeviceName",
                       "ET200M-D1");
```

29.12.20 SetIoDeviceData()

int SetIoDeviceData(string object, string parameter, string value)

Descrizione

Impostare i dati sulla configurazione IO.
Restituisce lo stato dell'operazione, dispari se ha successo e pari per l'errore.

Questo comando utilizza metodi e deve essere eseguito in pwrs. Non funziona in pwrc.

I parametri supportati per i dispositivi Profinet sono

NetworkSettings-DeviceName
NetworkSettings-IP Address
NetworkSettings-Subnet Mask
NetworkSettings-MAC Address
NetworkSettings-SendClock
NetworkSettings-ReductionRatio
NetworkSettings-Phase
NetworkSettings-API

Argument

string	object	Nome oggetto dispositivo.
string	parameter	Nome del parametro per i dati.
string	value	Valore da impostare.

Esempio

```
sts = SetIoDeviceData( "Nodes-MyNode-PN-D1", "NetworkSettings-DeviceName",
                       "ET200M-D1");
```

29.12.21 GetVersion()

int GetVersion()

Descrizione

Ottieni la versione Proview per la versione corrente.
Restituisce un valore intero che e' 10000 * maggiore + 100 * minore + rilascio.
Ad esempio, V5.3.1 restituisce 50301.

Esempio

```
if ( GetVersion() > 50300)
  # Only for versions larger than V5.3.0
  ...
endif
```

29.12.22 get_pwr_config()

string get_pwr_config(string name)

Descrizione

Ottieni il valore di una variabile di configurazione.
I valori di configurazione sono impostati in /etc/proview.cnf.
Restituisce il valore della variabile di configurazione.

Esempio

```
group = get_pwr_config( "defaultSystemGroup");
```

29.12.23 get_node_name()

string get_node_name()

Descrizione

Ottieni il nome host per il nodo corrente.
Restituisce il nome host.

Esempio

```
name = get_node_name();
```

29.12.24 EVEN()

int EVEN(int sts)

Descrizione

Controllare se un numero intero e' pari.
Restituisce 1 se pari e 0 se dispari.

Esempio

```
sts = SetAttribute( "Pump-V1-Switch.Description", "Valve switch open");
if ( EVEN(sts))
  printf("Couldn't set attribute\ ");
endif
```

29.12.25 ODD()

int ODD(int sts)

Descrizione

Il controllo se un numero intero e' dispari.
Restituisce 1 se dispari e 0 se pari.

Esempio

```
sts = SetAttribute( "Pump-V1-Switch.Description", "Valve switch open");
if ( ODD(sts))
  printf("Set operation successful\ ");
endif
```

29.13 Comandi Wtt

Tutti i comandi wtt sono disponibili nel codice dello script. Una riga di comando wtt NON dovrebbe terminare con un punto e virgola. Le variabili possono essere sostituite nella riga di comando circondandole con apostrofi.

Esempio

```
string name = "PUMP-VALVE-Open";
string value = "The valve is open";
set attribute/name='name'/attr="Description"/value='value'
```

Esempio

```
string name;
string parname;
int j;
int i;
for ( i = 0; i < 3; i++)
  parname = "vkv-test-obj" + (i+1);
  create obj/name='parname'
  for ( j = 0; j < 3; j++)
    name = parname + "-obj" + (j+1);
    create obj/name='name'
  endfor
endfor
```