

LA NORMA IEC 1131-3: UN APPROCCIO FORMALE

PLC PROGRAMMING LANGUAGES: A FORMAL APPROACH

L. Baresi, S. Carmeli, A. Monti e M. Pezzè^(*)

Sommario

Questo articolo descrive un nuovo approccio per integrare le funzionalità standard di definizione, progettazione, generazione di codice e di editing con le capacità di modellizzazione e simulazione dell'impianto e delle sue interazioni con il controllo.

Le notazioni IEC 1131-3 (in particolare il **Funtion Block Diagram**) sono integrate con le equazioni che descrivono il comportamento dell'impianto e con un modello formale che consente l'analisi di proprietà temporali e funzionali.

Abstract

This paper introduces describes how to integrate standard editing and code generation functionalities with capabilities for modelling and simulating the plant and its interactions with the digital controller. The 1131-3 notations (in particular Functional Block Diagrams) are complemented with the equations that describe the behaviour of the plant and with an underlying formal model, which supports the analysis of functional and timing properties.

Introduzione

La standardizzazione gioca un ruolo chiave nell'ambito dello sviluppo industriale. L'annunciata compatibilità, la possibilità di scambio di dati e di integrazione tra i diversi tool, aumenta il livello di confidenza degli utenti, e incoraggia lo sviluppo di tool e metodologie congelando notazioni, tecniche e metodi.

La norma IEC 1131-3 [5] propone un insieme di notazioni per lo sviluppo di sistemi di controllo programmabili. Questa norma è oggi molto utilizzata in ambito industriale ed è supportata da strumenti commerciali. La maggior parte di questi strumenti consentono la definizione di modelli e la generazione di codice utilizzando le notazioni IEC 1131-3, ma non consentono l'analisi e la simulazione del sistema di controllo durante le prime fasi di sviluppo.

Questo articolo descrive come integrare le principali funzionalità di editing e di generazione di codice offerte dalla maggior parte degli strumenti conformi alla norma IEC 1131-3 presenti sul mercato, con le funzionalità di modellizzazione e di simulazione che consentono un'analisi dell'impianto controllato e delle sue interazioni con il sistema di controllo digitale. Attraverso opportune simulazioni e fasi di analisi della specifica integrata del controllore e del sistema controllato è possibile evidenziare sin dalle prime fasi di progetto

^{*}Dott. Ing. L. Baresi, dottore in ricerca, Prof. Mauro Pezzè, docente: Dipartimento di Elettronica e Informazione. Dott. Ing. S. Carmeli, dottoranda, Dott. Ing. A. Monti, ricercatore: Dipartimento di Elettrotecnica – Politecnico di Milano.

la presenza di errori funzionali altrimenti difficili da identificare. La possibilità di convalidare il sistema prima del termine del progetto si traduce, quindi, in una riduzione dei tempi e dei costi ad esso associati.

Le notazioni IEC 1131-3 (in particolare il Function Block Diagram) sono integrate con l'utilizzo delle equazioni differenziali che descrivono il comportamento dell'impianto e con il modello formale che supporta l'analisi della tempistica del sistema e delle sue funzionalità. L'articolo introduce l'approccio e descrive l'esperienza industriale all'interno del progetto ESPRIT INFORMA².

IEC Standard 1131-3

La norma IEC 1131-3 [5] definisce il modello software dei controllori programmabili e i linguaggi per programmarli. La norma propone 5 linguaggi di programmazione:

- **Instruction List (IL):** è un linguaggio testuale di basso livello con una struttura simile all'assembler. IL è un linguaggio adatto alla soluzione di semplici problemi e alla produzione di codice ottimizzato, ma non supporta la programmazione strutturata. Il linguaggio IL può essere interpretato direttamente da molti PLC conformi alla norma IEC1131-3. E' proprio per questo motivo che tale linguaggio è talvolta considerato *il linguaggio PLC* nel quale si possono tradurre tutti i linguaggi conformi alla norma IEC1131-3.
- **Structured Text (ST):** è un linguaggio procedurale di alto livello. ST mutua la propria sintassi dal Pascal. Il linguaggio ST aumenta i tipi di dati e supporta la programmazione strutturata. Talvolta considerato *il nuovo linguaggio di programmazione dei PLC*, è utile per gestire la complessità e la modularità dei controllori moderni.
- **Ladder Diagrams (LD):** è un linguaggio che può essere considerato l'evoluzione degli schemi elettrici. LD fornisce uno stile di programmazione mutuato dai circuiti elettrici ed elettronici. Tale linguaggio non è però adeguato alla complessità degli attuali controllori ed alla programmazione strutturata, ma è utile nel caso di sistemi già in uso.
- **Function Block Diagram (FBD):** è un linguaggio grafico simile a Analisi Strutturata [7]. I controllori sono modellati come flussi di dati e di segnali attraverso elementi di processo (*function blocks*). FBD trasforma la programmazione testuale (ST) nella connessione di blocchi predefiniti, migliorando così la modularità e il riutilizzo di codice.
- **Sequential Function Chart (SFC):** è un linguaggio grafico simile alle reti di Petri [8] e a SDL [9]. SFC è utilizzato per organizzare la struttura di un programma per PLC, ed è costituito da in insieme di passi e transizioni interconnesse tra loro attraverso connessioni dirette. Ai passi sono associate le azioni, alle transizioni sono associati i predicati (condizioni).

La norma IEC1131-3 favorisce lo sviluppo di software ben strutturato. I blocchi funzionali sono il concetto chiave per rinforzare la modularità e la riusabilità dei componenti software: algoritmi ottimizzati e strategie di controllo possono essere infatti definiti con opportuni blocchi in modo da permetterne un'efficiente riusabilità. IL, ST, FBD e LD sono

² Il lavoro presentato in questo articolo è parzialmente finanziato dalla Comunità Europea nell'ambito del progetto ESPRIT INFORMA (EP23163). Il consorzio comprende IFAD (Danimarca), Ansaldo Ricerche (Italia), Politecnico di Milano (Italia), Odense Steelshipyard (Danimarca), Ansaldo Sistemi Industriali (Italia), e Scientific Software Group (Francia).

visti come possibili mezzi per definire “l’interno” dei blocchi funzionali, mentre SFC è utilizzato come *collante* per combinare le varie parti e organizzare la struttura del programma.

Approccio

Il progetto di controllori programmabili può essere migliorato utilizzando un approccio che integra le funzionalità standard di editing e di generazione di codice con le nuove funzionalità di simulazione ed analisi dei sistemi embedded (ibridi), dove il controllore digitale e la parte analogica sono strettamente legate. L’approccio descritto in questo articolo integra notazioni largamente utilizzate per definire un toolbox da utilizzarsi nella pratica industriale. Il controllore digitale è specificato utilizzando il linguaggio FBD³; l’impianto è modellizzato con Matlab/Simulink [6].

I modelli FBD sono tradotti automaticamente in reti di Petri di alto livello (HLTPN nel seguito), attraverso l’utilizzo di regole opportune. Tali regole definiscono le HLTPN che dal punto di vista funzionale risultano equivalenti ai blocchi FBD.

La creazione di nuovi blocchi FBD impone la definizione o della HLTPN o del codice C che ne specifica il comportamento. Il blocco così definito può essere utilizzato come se fosse un elemento di libreria. L’approccio descritto in [1,2] permette di costruire automaticamente la rete HLTPN corrispondente a ciascun modello FBD. Le reti HLTPN restano nascoste al progettista di sistemi di controllo e agiscono come motore formale. Le HLTPN consentono di:

- **Eseguire (simulare) modelli FBD.** Gli eventi di esecuzione delle HLTPN sono mostrati come visualizzazioni (animazioni) di blocchi FBD.
- **Eseguire il debugging di modelli FBD.** Il modello FBD può essere opportunamente controllato eseguendone il debugging.
- **Analizzare (convalidare) modelli FBD.** Le corrispondenti HLTPN sono analizzate sfruttando le tecniche e gli algoritmi solitamente utilizzati per le reti di Petri (per esempio l’analisi di raggiungibilità, la verifica del modello) e i risultati sono tradotti in visualizzazioni facilmente interpretabili da esperti FBD.
- **Generazione di codice da modelli FBD.** Le reti HLTPN costituiscono il passo intermedio che consente la generazione automatica del codice C corrispondente ad una specifica FBD.
- **Generazione di casi di test.** Le HLTPN sono utilizzate per la generazione di casi di test da utilizzarsi durante la fase di testing sulla CPU.

La figura 1 rappresenta i diversi componenti del toolbox INFORMA. L’intero sistema viene definito interamente utilizzando sia Simulink che l’editor FBD INFORMA. Quindi si utilizza Matlab per eseguire la parte analogica del sistema (attraverso la soluzione delle equazioni che descrivono il sistema nel dominio del tempo) e le HLTPN per eseguire i task.

La simulazione sia dell’impianto che del controllore è integrata in un unico ambiente in modo da evidenziare il comportamento dell’intero sistema. In figura 2 è raffigurato un esempio di simulazione di una specifica INFORMA in ambiente Simulink: il blocco *Plant* contiene la definizione del modello dell’impianto in termini di blocchi Simulink mentre il blocco *Control System* contiene la specifica del controllo in termini di diagramma FBD e di

³ I componenti con vincoli temporali non stringenti possono essere specificati utilizzando anche VDM++. In questo articolo non si considera il VDM++, per ulteriori approfondimenti si faccia riferimento a [3].

specifica VDM⁺⁺.

L'approccio INFORMA integra in un unico ambiente, standard internazionali o standard *de facto* (FBD, VDM⁺⁺) con strumenti largamente utilizzati in ambiente industriale (Matlab/Simulink), consentendo quindi di godere dei benefici derivanti dall'utilizzo di metodologie formali. La metodologia formale (HLTPN) rappresenta il cuore del toolbox, ma gli esperti del settore (ingegneri progettisti di sistemi di controllo) non devono interagire direttamente con questi. Si continuano ad utilizzare le medesime notazioni e tecniche di progetto, guadagnando i benefici legati alle metodologie formali pur ignorandone completamente l'esistenza anche durante le diverse fasi di progetto.

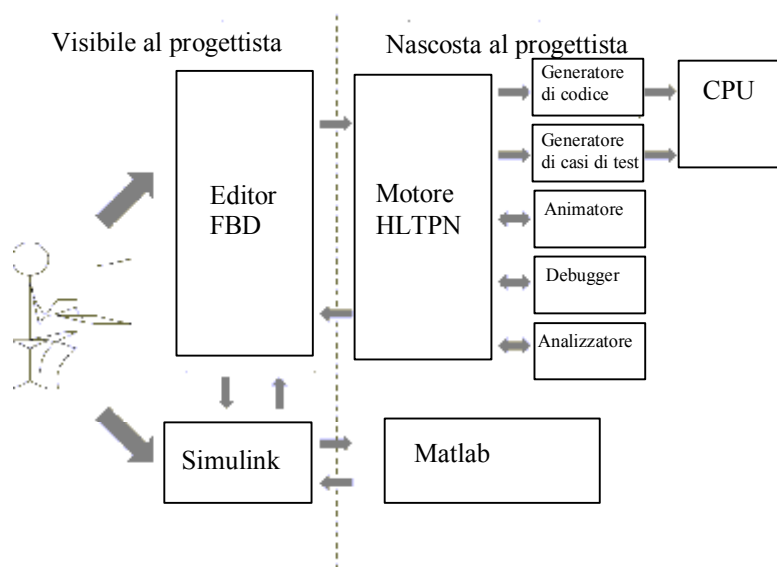


Figura 1: Visione ad alto livello dei componenti del toolbox INFORMA

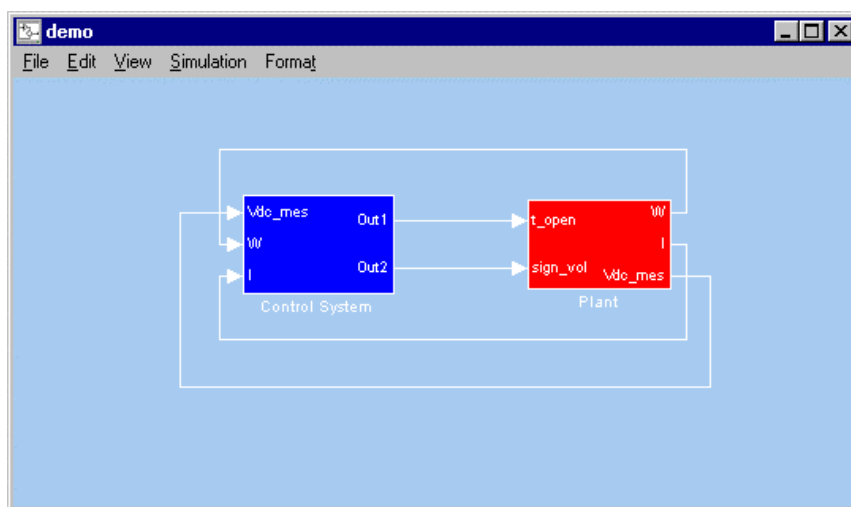


Figura 2: Un esempio di simulazione integrata in ambiente Simulink

Nei paragrafi che seguono si descrive l'editor PLC contenuto nel pacchetto INFORMA e si introduce un esempio di un'applicazione della metodologia ad un caso reale.

L'editor INFORMA ISO-PLC

Avviando il pacchetto INFORMA dalla finestra di comando di Matlab è possibile aprire l'editor ISO-PLC di INFORMA che consente all'utente di sviluppare rapidamente una nuova specifica utilizzando il linguaggio FBD. Dal punto di vista dell'utente i passi principali del progetto di un modello ISO-PLC sono i seguenti:

- apertura di un modello già esistente o creazione di uno nuovo;
- inserimento di nuovi blocchi presi dalla libreria ;
- definizione di nuovi blocchi, in particolare è possibile:
 - definire un blocco generico: l'utente può specificare le funzionalità del blocco utilizzando un linguaggio ad alto livello (ad esempio codice C), mentre l'aspetto grafico resta standardizzato;
 - definire un blocco personalizzato: l'utente può specificare sia le proprietà che le funzionalità del blocco agendo direttamente sulla specifica della rete di Petri;
 - definire un blocco di libreria: si può aggiungere un blocco personalizzato alle librerie e renderlo disponibile per un suo futuro utilizzo;
- salvataggio della specifica definita.

Nel seguito si introducono i principali elementi di editing insieme al loro ruolo sintattico.

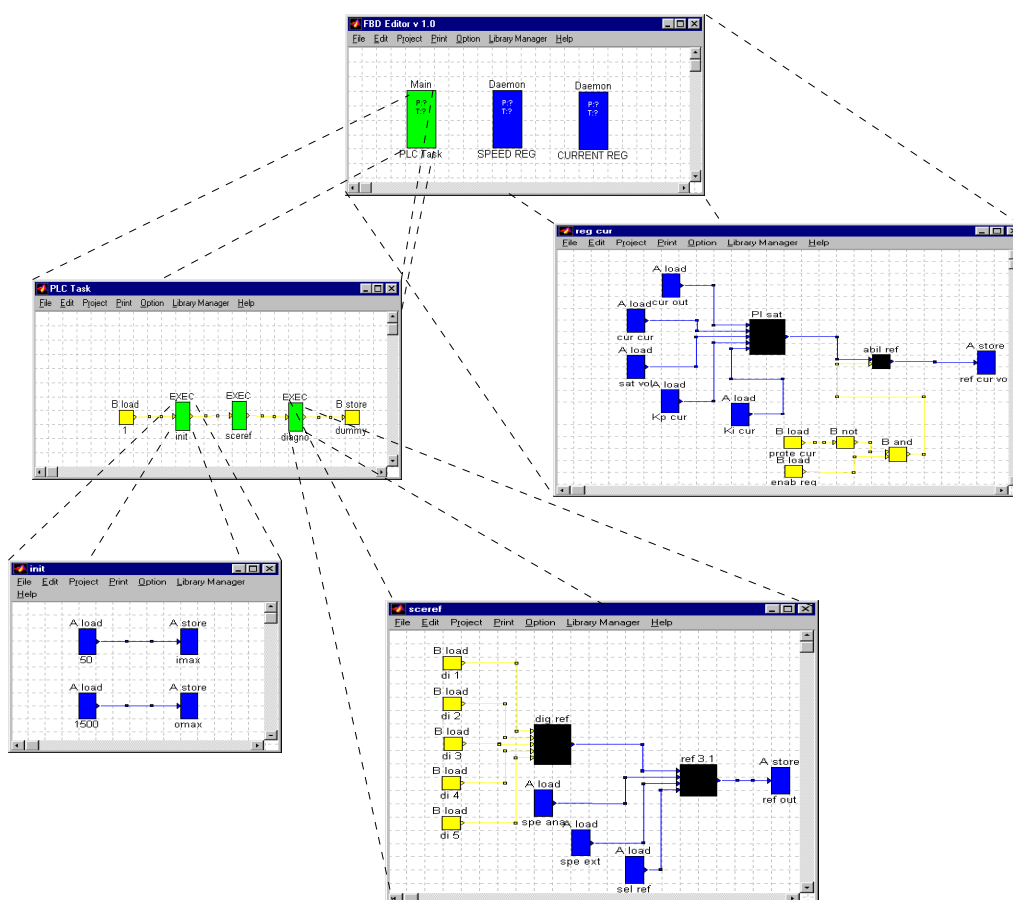


Figura 3: Tipica struttura gerarchica delle sessioni

Una SESSIONE è una finestra di lavoro nella quale è possibile disegnare diagrammi costituiti da blocchi. L'insieme delle sessioni sono strutturate in modo dinamico (si veda la figura 3)

Blocco

I blocchi sono gli elementi fondamentali delle notazioni. Ci sono tre tipi di blocchi (si veda la figura 4): blocchi *normali*, *task* e *sessioni*, tutti basati sulla medesima struttura.

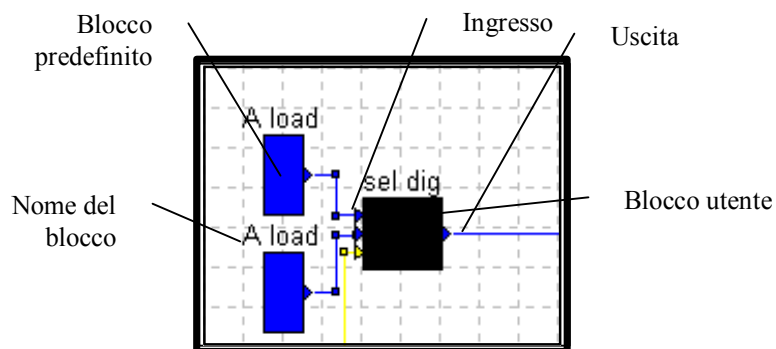


Figura 4: Struttura dei blocchi

I *blocchi normali* sono blocchi operazionali. Realizzano operazioni logiche o matematiche (ad esempio AND, +) oppure controllano gli ingressi/uscite (ad es. load, store). I *blocchi Task* descrivono l'architettura software in termini di strategia di scheduling. I *blocchi Session* rappresentano una nuova sessione.

I *Task* sono unità operative; mentre le sessioni descrivono graficamente il codice da eseguire. I task predefiniti sono i seguenti: *Main*, *Daemon*, *Interrupt*, e *Exception*.

Il *task main* è eseguito periodicamente. Ad ogni ciclo chiama le relative sessioni. Un task principale è sempre presente in una specifica e può essere interrotto solo da task a priorità più alta, ad esempio da un daemon task.

Un *daemon task* ha il livello di priorità massima ed è eseguito secondo una tempistica predefinita. Non è possibile assegnare la medesima priorità a più di un daemon task.

Un *task interrupt* modella un interrupt.

Un *task exception* prende il controllo della CPU su un'eccezione hardware. Può essere inserito un solo task exception.

Per ogni task si possono specificare più sessioni ottenendo una struttura ad albero.

Lo scambio di dati tra blocchi è rappresentato dalle linee rappresentate utilizzando il colore del tipo di dato scambiato.

Controllo di velocità di una macchina sincrona

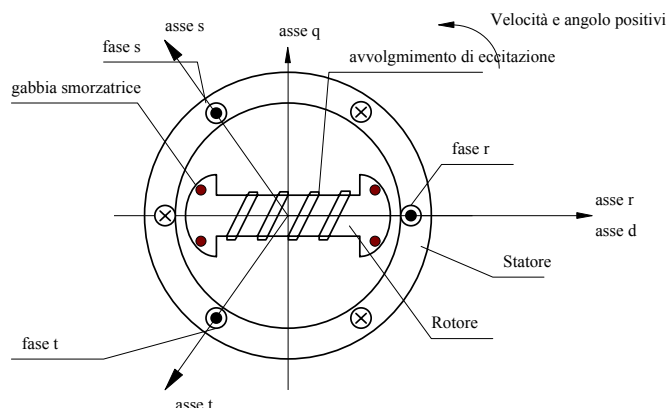


Figura 5: Sezione di una macchina sincrona

Dal punto di vista strutturale, una macchina sincrona è composta dalle seguenti parti (si veda figura 5):

- uno statore esterno con avvolgimenti trifasi che agiscono come indotto
- un rotore interno con avvolgimenti alimentati in corrente continua che agiscono come induttore. Il rotore presenta un avvolgimento cortocircuitato per limitare il transitorio derivante da rapide variazioni di carico (gabbia smorzatrice)

In applicazioni di controllo di velocità, la macchina sincrona è equipaggiata con un encoder che consente di conoscere la velocità e la posizione del rotore (vel_att); sono monitorate anche le correnti di statore (ir_att , is_att , it_att) e la corrente di eccitazione (ie_att). Il controllo di velocità della macchina viene realizzato modificando la tensione del cicloconvertitore (i.e. un convertitore diretto AC/AC) e la corrente di eccitazione nell'avvolgimento di rotore conoscendo la velocità e la posizione del rotore, le correnti e la coppia.

In figura 6 il riferimento di velocità (vel_rif) e il valore effettivamente monitorato della velocità di rotore (vel_att) sono confrontate e la loro differenza costituisce l'ingresso del controllore di velocità che è di tipo PI. Il segnale in uscita dal controllore di velocità è proporzionale alla coppia elettromagnetica prodotta dalla macchina sincrona (C_e), e così si ottiene il riferimento di coppia (C_{e_ref}). Quest'ultimo viene diviso per il modulo del vettore flusso magnetizzante (ψ_att) per ottenere il valore di riferimento della componente di corrente di statore che produce la coppia iy_rif .

La velocità di rotore monitorata è l'ingresso del blocco di indebolimento di campo che per velocità inferiori a quella base fornisce un valore di flusso di riferimento costante (ψ_rif); sopra la velocità base il flusso è ridotto. Il vettore flusso di riferimento (ψ_rif) viene confrontato con il valore attuale del flusso magnetizzante (ψ_att) e la loro differenza è l'ingresso del regolatore di flusso dinamico, che è di tipo PI.

Il controllore di flusso mantiene il flusso magnetizzante al valore di preset, indipendentemente dal carico. Per assicurare un utilizzo ottimale del motore, il valore del flusso di riferimento (ψ_rif) è molto elevato durante l'avviamento. L'uscita del regolatore dinamico di flusso è il valore di riferimento della componente di corrente storica che produce flusso magnetizzante (ix_att).

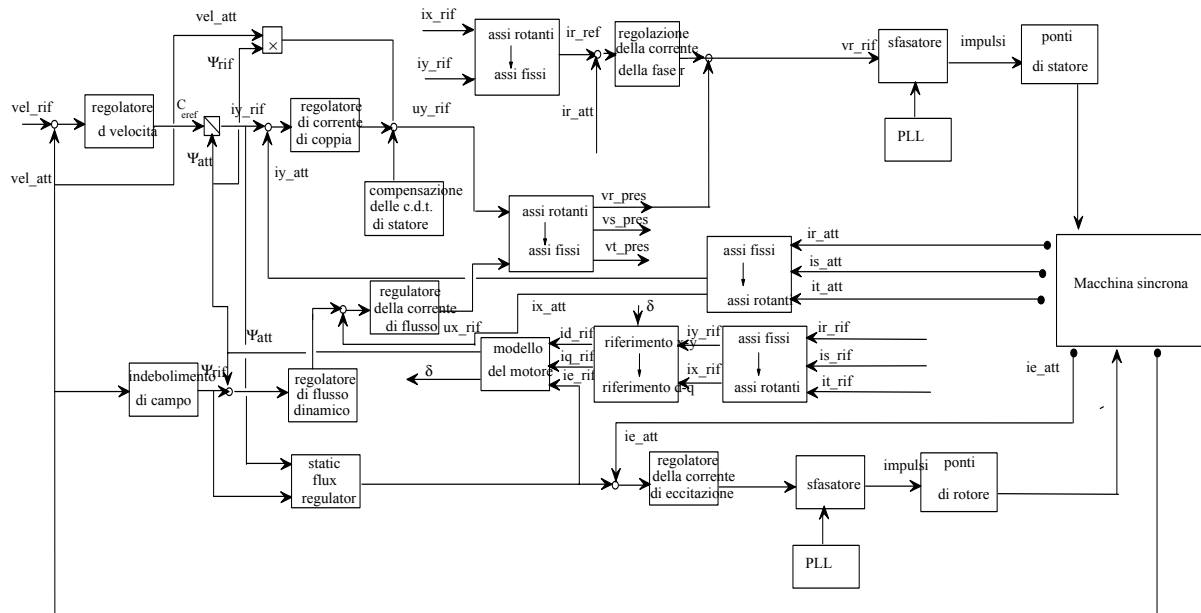


Figura 6: Schema di controllo di una macchina asincrona

Si osservi che ix_att è la componente di asse diretto della corrente statorica in un particolare sistema di riferimento, ed ha la stessa direzione del vettore flusso di magnetizzazione, mentre la componente iy_att è la componente di corrente che produce coppia.

Utilizzando l'inverso della matrice di trasformazione, le due componenti dei riferimenti della corrente statorica (ix_rif , iy_rif) vengono trasformate nei valori di fase, tale trasformazione è indicata in figura 6 con il seguente simbolo *assi rotanti* \rightarrow *assi fissi*. Le correnti di riferimento di fase di statore sono confrontate con i rispettivi valori attuali e la differenza costituisce l'ingresso dei controllori di corrente statorica, che sono di tipo PI. I segnali di uscita di tali controllori vengono utilizzati per generare gli impulsi di accensione del cicloconvertitore che alimenta la macchina.

In modo simile, il riferimento della corrente di campo viene confrontato con il suo valore attuale e la loro differenza è l'ingresso del regolatore della corrente di eccitazione. Quest'ultimo fornisce i segnali necessari ad un ponte raddrizzatore controllato.

Tale struttura di controllo può essere descritta in termini di insiemi di task composti da opportune sessioni.

La finestra principale dell'editor INFORMA e l'insieme dei task che costituiscono la struttura di controllo sono rappresentati in figura 7.

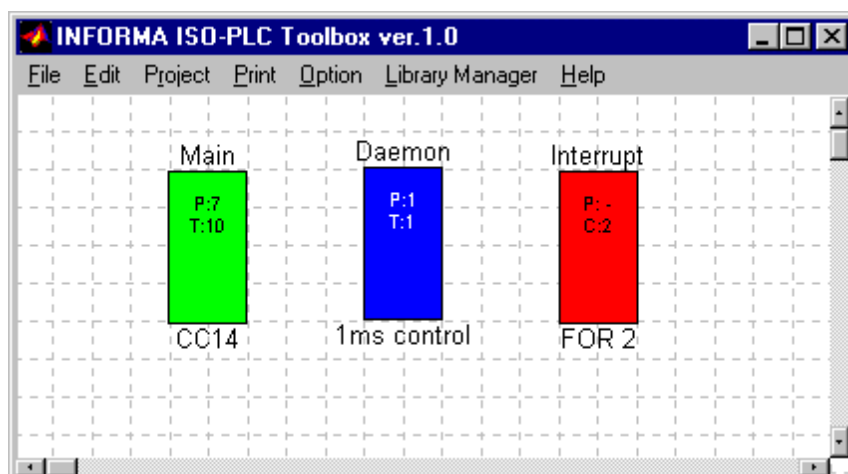


Figura 7: Principale finestra di specifica dell'esempio

Cliccando due volte sull'icona di un task si aprono le relative sessioni consentendo di esplorare la struttura gerarchica della specifica. In questa specifica è possibile riconoscere:

- il task principale (*CC14*) con un periodo pari a 10 ms
- un daemon task periodico (*1 ms control*) con un ciclo di 1 ms
- una routine di interrupt (*FOR 2*).

Si consideri ora la routine di interrupt. La sessione che la realizza è in illustrata in figura 8.

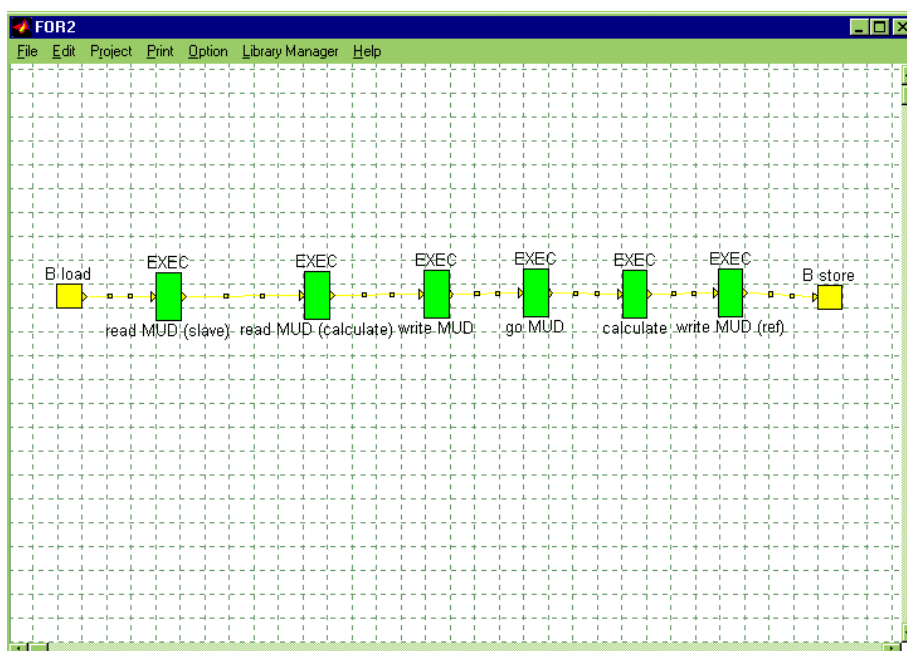


Figura 8: Sessioni contenute nel task FOR2

Questa sessione è descritta dalla sequenza di sei sessioni eseguite una dopo l'altra. Esplorendo la sessione *motor model* si ottiene il diagramma di figura 9: in tale sessione sono stati utilizzati sia blocchi standard che blocchi utente per definire la logica di controllo per la ricostruzione del flusso e per il suo controllo.

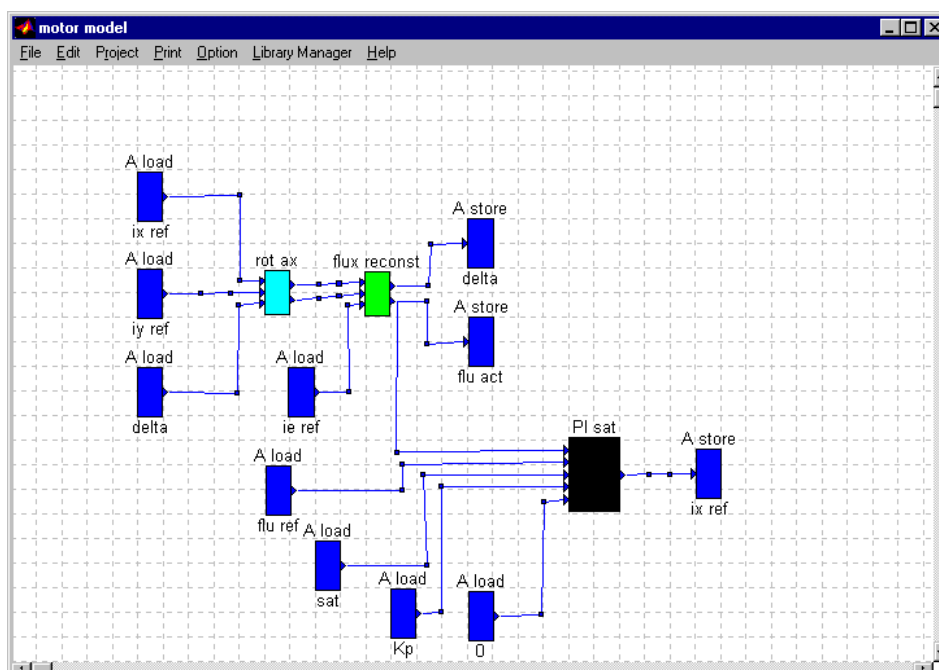


Figura 9: La sessione "motor model"

Conclusioni

L'articolo presenta le caratteristiche principali dell'ambiente INFORMA PLC insieme alla loro relazione con la norma IEC 1131-3. Si sono evidenziati i vantaggi che derivano dall'utilizzo di un ambiente integrato per l'editing, la simulazione, la verifica di controllori integrati e dalla specifica dell'impianto attraverso un semplice esempio.

Ringraziamenti

Gli autori ringraziano tutti i partner coinvolti nel progetto INFORMA ed in particolare la dottoressa Carla Penno (Ansaldo Ricerche) e il dottor Ezio Cosatto (Ansaldo Sistemi Industriali) per il loro aiuto nella descrizione dello studio del caso riportato.

Bibliografia

- [1] L. Baresi. *Personalizzazione Formale di Notazioni Grafiche*. Tesi di dottorato, Dipartimento di Elettronica e Informazione – Politecnico di Milano (1997)
- [2] L. Baresi, A. Orso, and M. Pezzè. Introducing Formal Methods in Industrial Practice. In *Proceedings of the 19th International Conference on Software engineering*, pp 56-66. ACM Press (1997)
- [3] E. H. Durr and N. Plat. *VDM++ Language Reference Manual*. Rapporto tecnico, IFAD – The Institute of Applied Computer Science (1995)
- [4] C. Ghezzi, D. Mandrioli, S. Morasca and M. Pezzè. A Unified High-Level Petri Net Model for Time-Critical Systems. *IEEE Transaction on Software Engineering*, **17**(2): 160-172 (1991)
- [5] IEC. Part 3: Programming Languages, IEC 1131-3. Technical Report, International Electrotechnical Commission - Geneva (1993)

- [6] The MathWorks Inc. Matlab 5.2 URL: <http://www.mathworks.com/products/matlab/> (1998)
- [7] T. De Marco. *Structured Analysis and System Specification*. Prentice Hall (1978)
- [8] T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, **77**(4): 541-580 (1989)
- [9] O. Faergemand and A. Olsen. Introduction to SDL-92. *Computer Networks and ISDN Systems*, **26**:1143-1167 (1994)