



UNIVERSITÀ DEGLI STUDI DI PARMA

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA SPECIALISTICA IN INGEGNERIA INFORMATICA

---

SVILUPPO DI SOFTWARE DI CONTROLLO PER  
AUTOMAZIONE INDUSTRIALE SECONDO LO  
STANDARD OMAC

DEVELOPING CONTROL SOFTWARE FOR INDUSTRIAL  
AUTOMATION ACCORDING TO THE OMAC STANDARD

Relatore:

Chiar.mo Prof. STEFANO CASELLI

Correlatore:

Ing. OSCAR GERELLI

Tesi di Laurea di:

MICHELE PATTERA

ANNO ACCADEMICO 2009-2010

*Gli ultimi tempi di un ciclo che si chiude danno sempre modo di pensare al periodo trascorso, e si finisce quasi sempre per apprezzarli. Quello appena terminato è stato impegnativo e lungo, per quanto possa valere il mio punto di vista, ma non privo di soddisfazioni. Per questi motivi penso che sia giusto marcare il segno che ha lasciato, e questa pagina, l'ultima per tempo e la prima per importanza, ha questo, nemmeno troppo difficile, compito. Ci sono molte persone che hanno preso parte a tutto ciò, ed è a loro che mi sento di dire grazie.*

*Un primo ringraziamento va al prof. Caselli, alla competenza e alla passione che mette nel suo lavoro, sempre volto ad offrire e ad ottenere il meglio dai suoi studenti. E anche alla sua pipe, che ha sempre funzionato in maniera impeccabile, specialmente quando le scadenze erano prossime.*

*Il secondo ringraziamento è per Oscar, che definire disponibile sarebbe limitativo, per aver avuto, o essersi procurato, una risposta ad ogni domanda che gli ho posto.*

*Grazie alla mia famiglia, ai miei genitori che hanno reso possibile il raggiungimento di questo traguardo e a mio fratello, che si è spesso "sporcato le mani" dandosi da fare per aiutarmi quando ho avuto bisogno.*

*Grazie a chi negli ultimi tempi ha trascorso con me le giornate in quel posto del quale ormai conosciamo anche l'ultimo granello di polvere. Non per caso certo, ma perché sulla mia stessa barca e, a quanto mi è sembrato, ci siamo quasi divertiti :)*

*Infine vorrei ringraziare tutte le persone che hanno condiviso con me questi anni, chi in università e chi fuori, gli amici che ormai considero "di sempre" e quelli nuovi. Nelle prossime pagine c'è un pochino di me e, quindi, di voi. Grazie.*

*“... perché il tempo ci sfugge  
ma il segno del tempo rimane...”*

*Baustelle*

# Indice

<b>Introduzione</b>	<b>1</b>
<b>1 Supervisione ed interoperabilità</b>	<b>5</b>
1.1 La supervisione negli impianti di produzione . . . . .	5
1.2 OMAC Packaging Workgroup . . . . .	6
1.2.1 PackML . . . . .	7
1.2.2 PackTags . . . . .	16
1.3 Lo standard Weihenstephan . . . . .	21
1.3.1 Modello degli stati WS . . . . .	23
1.3.2 WS tags . . . . .	23
1.3.3 Rispetto delle linee guida OMAC . . . . .	25
<b>2 Requisiti e tecnologie</b>	<b>28</b>
2.1 Obiettivi . . . . .	28
2.2 Requisiti . . . . .	29
2.3 Strumenti di lavoro . . . . .	30
2.3.1 Simatic S7-300 . . . . .	31
2.3.2 Simatic STEP7 . . . . .	37
2.3.3 Simatic S7-PLCSIM . . . . .	41
2.3.4 Simatic HMI . . . . .	41
<b>3 Progettazione e realizzazione</b>	<b>46</b>
3.1 Approccio seguito . . . . .	46
3.2 Scelte progettuali . . . . .	47

---

3.2.1	Guida alla realizzazione Procter&Gamble . . . . .	48
3.2.2	Siemens OPL . . . . .	51
3.3	Scelte realizzative . . . . .	53
3.3.1	Struttura dati . . . . .	56
3.3.2	Funzioni di basso livello . . . . .	57
3.3.3	Funzionalità principali . . . . .	59
3.3.4	Supervisore . . . . .	64
<b>4</b>	<b>Sistema sperimentale</b>	<b>68</b>
4.1	Ambiente di test . . . . .	68
4.2	Risultati sperimentali . . . . .	71
4.3	Funzionamento . . . . .	75
<b>5</b>	<b>Conclusioni</b>	<b>79</b>
	<b>Bibliografia</b>	<b>80</b>

# Elenco delle figure

1.1	Modello degli stati PackML per il modo automatico . . . . .	13
1.2	Transizioni tra modi di funzionamento . . . . .	16
1.3	Stack ISO/OSI per WS . . . . .	22
1.4	Modello degli stati WS . . . . .	24
1.5	Mappatura degli stati WS su modello PackML . . . . .	27
2.1	Schema del sistema di simulazione da realizzare . . . . .	29
2.2	Aree di memoria delle CPU 31xC/31x . . . . .	32
2.3	Connettività delle CPU 31xC/31x . . . . .	34
2.4	Ciclo di scansione delle CPU 31xC/31x . . . . .	35
2.5	Ciclo di attuazione in tempo reale . . . . .	36
2.6	Simatic Manager . . . . .	37
2.7	Chiamate annidate in un programma STEP7 . . . . .	39
2.8	Interfaccia utente di S7-PLCSIM . . . . .	42
2.9	Dispositivi Simatic HMI . . . . .	43
2.10	Interfaccia utente di WinCC flexible 2008 . . . . .	45
3.1	Scomposizione modulare S88 dell'applicazione P&G . . . . .	50
3.2	Struttura della libreria Siemens LPMLV30 . . . . .	51
3.3	Schema concettuale della libreria realizzata . . . . .	56
3.4	Diagramma di flusso del blocco gestore di modo . . . . .	60
3.5	Logica interna del gestore di stato (diagramma KOP) . . . . .	62
3.6	Diagramma di flusso del blocco gestore dei tempi . . . . .	63

---

3.7	Diagramma di flusso del blocco gestore degli eventi . . . . .	64
3.8	Ambiente WinCC flexible, creazione dei collegamenti . . . . .	65
3.9	Ambiente WinCC flexible, definizione delle variabili . . . . .	66
3.10	Ambiente WinCC flexible, controlli grafici . . . . .	67
4.1	CPU 315-2 PN/DP . . . . .	70
4.2	Pannello HMI Simatic MP277 10" Touch . . . . .	71
4.3	Tempi di ciclo del sistema di test . . . . .	73
4.4	Utilizzo di memoria del sistema di test . . . . .	75
4.5	Interfaccia HMI, pagina principale . . . . .	76
4.6	Interfaccia HMI, pagina dei tempi . . . . .	77
4.7	Interfaccia HMI, pagina degli eventi . . . . .	78

# Elenco delle tabelle

1.1	Conformazioni di linea PackML . . . . .	8
1.2	Matrice di transizione di stato PackML per il modo automatico . . . . .	12
1.3	Modi macchina PackML . . . . .	14
1.4	Tipi di dato PackTags . . . . .	19
1.5	Tipi di dato WS . . . . .	25
3.1	Interfaccia dei blocchi bit_check, bit_set, bit_unset . . . . .	57
3.2	Interfaccia del blocco memcpy_s . . . . .	58
3.3	Interfaccia del blocco memset_s . . . . .	58
3.4	Interfaccia del blocco shift_up . . . . .	59
3.5	Interfaccia del blocco memadd_s . . . . .	59
3.6	Interfaccia del blocco PMLModeManager . . . . .	60
3.7	Interfaccia del blocco PMLStateManager . . . . .	62
3.8	Interfaccia del blocco PMLTimesManager . . . . .	63
3.9	Interfaccia del blocco PMLEventsManager . . . . .	64

# Introduzione

La progettazione e la successiva ingegnerizzazione di un moderno impianto industriale di produzione è un processo costituito da più fasi, svolte in sequenza, caratterizzate dal lavoro di diverse figure professionali che contribuiscono ognuna al completamento di un singolo passo. Maggiore è il numero dei cambiamenti che si rendono necessari durante le sopracitate fasi, maggiore sarà il numero di ripetizioni del percorso attraverso questa catena, in cui spesso alcune scelte ne condizionano altre. L'evoluzione a cui stiamo assistendo negli ambiti dell'informatica e dell'automazione fa sì che, nonostante i costi e contrariamente a quanto accadeva in passato, le decisioni di portare modifiche ai sistemi non siano guidate unicamente dalla necessità di correggere errori di progetto, ma sempre più dalla volontà di realizzare le caratteristiche di flessibilità ed interoperabilità che sono richieste a questo tipo di soluzioni.

Caratteristici di molti ambiti industriali moderni, tra i quali quelli di food&beverage e packaging, sono i cosiddetti processi "batch", ovvero quelli in cui le fasi della lavorazione dei prodotti sono eseguite in modo sequenziale ordinato ed in cui ogni fase può richiedere lo svolgimento di più passi elementari, eventualmente ripetuti, per essere completata. Il controllo dei processi batch deve soddisfare numerose funzionalità, oltre alle normali azioni di coordinamento delle operazioni da svolgere, quali ad esempio la gestione degli allarmi, delle situazioni di errore, delle ricette, delle pianificazioni. Il controllo di questo tipo di processi deve essere considerato un compito complesso, più che difficile. Un problema difficile è un problema che solitamente ammette una sola soluzione anche se matematicamente o scientificamente difficile da individuare, mentre un problema complesso è un problema difficile da manipolare, più impegnativo dal punto di vista pratico. Analogamente un processo batch è caratterizzato da numerosi

vincoli, dipendenze tra gli attori coinvolti, obiettivi mutuamente esclusivi e ammette molte possibili soluzioni: la strada che conduce alla soluzione di un problema complesso passa tipicamente per la ricorsiva scomposizione modulare in problemi di più basso livello, risolvibili in modo più agevole.

In passato la funzione primaria del software di controllo dei macchinari industriali era limitata allo scandire il ritmo delle operazioni da eseguire in base a schemi preimpostati. I moderni sistemi di controllo per macchinari industriali fanno pesante affidamento sul software e sui feedback sensoriali per offrire le funzionalità richieste potendo contare su potenze di calcolo e disponibilità di memoria che solo in tempi recenti è stato possibile ottenere: essi diventano a tutti gli effetti sistemi di elaborazione in cui si realizza la completa supervisione locale garantendo migliori prestazioni, sicurezza, maggiore autonomia funzionale e si generano grandi quantità di informazioni relative ai processi a cui sovrintendono. Sempre più spesso i macchinari che compongono una stessa linea di produzione devono rendere disponibili all'esterno queste molteplici ed eterogenee informazioni per garantire il corretto funzionamento della catena. Inoltre sono sempre più richiesti sistemi di controllo della produzione centralizzati (i cosiddetti Manufacturing Execution System o Line Information System) che, basandosi sulle informazioni ottenute dal campo, siano in grado di coordinare il funzionamento della linea e calcolarne gli indici di produttività. Tali indici, in genere denominati Key Performance Indicators, sono utili ad individuare punti critici del processo e quindi eventuali azioni correttive.

Un tipico impianto di produzione è ottenuto dall'integrazione di numerose macchine automatiche quali ad esempio, nell'ambito del food&beverage, soffiatrici, riempitrici, tappatrici, incartonatrici, etichettatrici, ecc. . . , nonché da una serie di trasporti in grado di gestire il flusso del prodotto - grezzo o finito - fra una macchina e la successiva. Un problema ricorrente nell'ambito delle linee di produzione è quello dell'integrazione tra macchinari. L'eterogeneità dei fornitori consente di ottimizzare i costi e le prestazioni delle singole macchine, ma è spesso causa di inefficienze dal punto di vista dei costi complessivi industriali, dei tempi di installazione e dell'armonizzazione delle interfac-

ce. I principali problemi non sono causati dal possibile utilizzo di differenti linguaggi di programmazione o diverse connessioni fisiche, ma dalla difficoltà di dover padroneggiare convenzioni e nomi fra loro non omogenei. Ulteriori problemi insorgono durante l'utilizzo, dal momento che gli operatori e gli utilizzatori possiedono un proprio background tecnico-culturale e necessitano di essere addestrati per interagire con ogni nuova interfaccia uomo macchina. È pertanto comprensibile come il funzionamento di un impianto coinvolga spesso molteplici attori e come sia importante che essi siano in grado di parlare una lingua comune: clienti finali, costruttori di macchine (OEM) e integratori di sistemi hanno interesse ad individuare delle terminologie, delle strutture di controllo e delle metodologie comuni di organizzazione dei programmi software, affinché sia per tutti più semplice ed efficiente lavorare insieme. Da più di qualche tempo, quasi una decina d'anni, queste problematiche hanno stimolato l'interesse delle parti coinvolte ed hanno portato alla costituzione di gruppi di lavoro nell'ambito di organizzazioni internazionali con l'obiettivo di delineare e condurre verso la standardizzazione un insieme di convenzioni che possano soddisfare le esigenze attuali. Ad OMAC Packaging Workgroup va attribuito lo standard PackML [1] che è supportato, diffuso e adottato principalmente nelle realtà industriali nordamericane, mentre lo standard Weihenstephan [2] sta emergendo nei settori packaging e food&beverage dell'industria europea.

Il lavoro descritto in questa tesi si colloca nell'ambito del Laboratorio di Ricerca Integrapack [3], nato dalla collaborazione tra Università, OCME s.r.l. e Promag s.r.l. con l'obiettivo di sviluppare un'isola altamente automatizzata per riempimento, confezionamento e stoccaggio di oli vegetali. Ad oggi, la possibilità di scambiare informazioni è sempre più ritenuta un valore fondamentale ed una parte centrale del progetto, oggetto di tesi, consiste nello sviluppo di un metodo di organizzazione a livello software e di un protocollo di comunicazione che, aderendo a standard industriali, possa allargare il ventaglio delle possibili applicazioni dell'isola, consentendo ad altri costruttori di integrare in modo semplice le proprie macchine all'interno dell'impianto a supervisione OCME.

La prima parte della tesi è dedicata alla presentazione sintetica dei documenti messi a disposizione dalle organizzazioni di standardizzazione per meglio comprendere vantaggi, punti critici, caratteristiche di flessibilità e possibilità di armonizzazione offerti dalle soluzioni proposte al fine di individuare le linee guida per lo sviluppo di sistemi compatibili.

La parte centrale riguarda la progettazione e realizzazione di una libreria software a supporto dell'integrazione dei meccanismi di comunicazione studiati nei dispositivi di controllo industriale.

Per ultimo, sarà presentato un sistema sperimentale realizzato al fine di effettuare test e valutazioni di carattere pratico in condizioni operative quanto più simili possibile alle attuali soluzioni industriali.

# Capitolo 1

## Supervisione ed interoperabilità

In questo capitolo sono illustrate le principali tematiche relative al ruolo del software nelle moderne linee produttive e ai compiti che i sistemi di controllo della produzione devono assolvere. Verranno presentati gli standard per l'organizzazione del software di controllo e per la comunicazione tra macchine industriali presi in considerazione nella fase di analisi.

### 1.1 La supervisione negli impianti di produzione

I sistemi di Production Data Acquisition (PDA) e Manufacturing Execution Systems (MES) sono largamente utilizzati negli impianti di confezionamento dell'industria alimentare. Il loro ruolo è quello di accrescere la precisione e la continuità nel flusso delle informazioni, rendere più trasparenti le varie fasi della produzione e fornire maggiori dati a supporto delle decisioni strategiche all'interno dell'azienda. Questi obiettivi dovrebbero condurre verso nuovi limiti di produttività e profitti, aumentando contemporaneamente il controllo sulla qualità dei prodotti finiti. Attualmente sistemi di questo tipo possono assolvere molti compiti in maniera automatica:

#### **Supervisione tecnica dell'impianto**

Visualizzazione delle quantità, dei tempi, di tutto ciò che il macchinario misura e dello stato in cui si trova, oltre che degli avvisi di allarmi e manutenzioni da effettuare.

### **Gestione della qualità**

Prevenzione dei guasti e identificazione precisa delle eventuali cause, al fine di preservare la qualità dei prodotti.

### **Preparazione delle informazioni relative ai processi produttivi**

Archiviazione dei dati di produzione, elaborazione e compressione dei dati, generazione di report, registrazione automatica dei flussi dei materiali in ingresso e uscita, creazione di statistiche di produzione, tempi, efficienza rispetto a specifici parametri ed utilizzo delle materie prime.

In passato, le industrie di confezionamento che hanno utilizzato sistemi PDA (tipicamente proprietari e personalizzati, investendo molto tempo e risorse per la loro realizzazione) non hanno raccolto i benefici che si aspettavano. Questi sistemi raccolgono e memorizzano enormi quantità di dati, che senza un'adeguata organizzazione e compressione possono facilmente diventare un flusso disordinato di informazioni difficile da valorizzare. Inoltre, molto spesso, non c'è integrazione con il sistema informativo aziendale con il risultato che gli impianti appaiono sempre più isolati all'interno della struttura aziendale rendendo l'integrazione molto difficile ed ostacolata da incompatibilità e dalla necessità di mantenere le informazioni distribuite (e quindi inutilmente ridondate). Di qui la necessità di razionalizzare e definire con precisione la natura e la quantità delle informazioni messe a disposizione di questi sistemi di controllo aziendale.

## **1.2 OMAC Packaging Workgroup**

La *Organization for Machine Automation and Control* (OMAC) è un'organizzazione mondiale che rappresenta soggetti impegnati nel campo dell'automazione a supporto della produzione industriale. Tra i suoi membri sono inclusi produttori, integratori, fornitori di tecnologia e clienti finali interessati allo sviluppo e all'implementazione di standard aperti per le applicazioni produttive. L'organizzazione opera tramite vari gruppi di lavoro, tra i quali vi è il *Packaging Workgroup* che si pone come obiettivo principale quello di migliorare gli standard per l'automazione di impianti di confezio-

namento ed imballaggio rendendo disponibili linee guida per l'implementazione. Il raggiungimento di questo obiettivo dovrebbe portare a maggiore flessibilità, migliore capacità produttiva e alla riduzione dei costi di integrazione tra sistemi di vari produttori; a tal fine sono stati creati sottogruppi, ciascuno dei quali approfondisce una particolare tematica.

### 1.2.1 PackML

Packaging Machinery Language (PackML) è stato, fin dall'inizio, orientato a proporsi come un insieme di linee guida per l'organizzazione del software di controllo di macchine automatiche industriali al fine di realizzare un "common look and feel" nelle modalità operative tra le diverse entità che collaborano alla realizzazione di un processo. Il documento finale prodotto dal team è consistente con lo standard ISA-88 ed è divenuto parte del technical report ISA-TR88.00.02-2008 [4]. L'approccio seguito è quello di dare una definizione rigorosa degli stati di funzionamento, al fine di poter costruire indicatori efficaci delle prestazioni della linea. Una definizione rigorosa degli stati richiede un modello di stato coerente e una robusta gestione delle transizioni tra le varie modalità di funzionamento, anch'esse definite nello standard: soltanto in questo modo è possibile ottenere uniformità rispetto a tutti quei meccanismi industriali che necessitano di conoscere in tempo reale ciò che avviene nelle varie fasi di un processo.

Nel seguito vengono presentati lo standard ed i principi sui quali esso si basa scegliendo di mantenere i termini inglesi per la nomenclatura dei componenti strutturali, ritenuti più idonei al fine di ottenere un riferimento diretto tra questo documento e quelli studiati. Le informazioni e le definizioni necessarie alla comprensione sono state ricavate dai documenti messi a disposizione dal team PackML [1, 5].

#### PackML line types

Per rappresentare le diverse tipologie di linee di confezionamento sono stati adottati quattro tipi di configurazioni possibili, ovvero i *Line Types* 1, 2, 3 e 4. In tabella 1.1 è schematizzato un riepilogo per delineare le caratteristiche di ogni tipo.

Line Type 1	Tutte le macchine sono autonome e reagiscono alle condizioni esterne utilizzando solamente le informazioni che provengono dalla sensoristica locale senza comunicare con gli altri macchinari che compongono l'impianto.
Line Type 2	Le macchine possono comunicare con le altre presenti nella linea; sono definiti due sottotipi funzionalmente identici: i macchinari all'interno di linee di tipo 2A comunicano tramite porte I/O analogiche o digitali, mentre nelle linee di tipo 2B i dati sono scambiati su reti, eventualmente su porte di I/O.
Line Type 3	Si tratta di una versione evoluta del tipo 2B, in cui è presente un server SCADA e/o un supervisore di linea. Le funzionalità aggiuntive riguardano la disponibilità di indicatori di performance, descrizioni dettagliate delle cause di allarmi, strumenti per la manutenzione e il debug, ecc. Inoltre i flussi di informazione provenienti da ogni singola linea possono essere connessi tra loro per visualizzare lo stato dell'intera fabbrica.
Line Type 4	Questo tipo di linea è integrata nel sistema informativo aziendale tramite il bus Enterprise Resource Planning (ERP); in questo modo, oltre a quanto già possibile nelle linee di tipo 3, gli ordini possono essere direttamente passati al reparto di produzione e lo stato di avanzamento di quest'ultima monitorato in modo automatico.

Tabella 1.1: Conformazioni di linea PackML

### PackML states

Uno stato macchina definisce in modo completo la condizione operativa attuale e il suo ciclo principale consiste generalmente in una sequenza di comandi e condizioni da verificare; ogni stato può, attraverso la sua logica o in risposta ad eventi esterni, condurre a ulteriori cicli di esecuzione al suo interno oppure abilitare transizioni verso altri stati. Gli stati sono disposti sequenzialmente secondo un ordine consistente con il modo di funzionamento corrente, sono predeterminati nel modello base ed in numero finito. Sono classificati in tre tipi principali:

- **Acting state** è uno stato che rappresenta una qualsiasi attività, come ad esempio

Starting oppure Holding. Implica l'esecuzione, singola o ripetuta, di azioni in ordine logico e definito per un tempo finito o fino a che una specifica condizione è raggiunta.

- **Wait state** è utilizzato per identificare uno stato in cui la macchina ha raggiunto un set di condizioni predefinite, come ad esempio Aborted, Suspended, Idle. In ognuno di questi stati le condizioni sono mantenute sino a che non si verifica una transizione verso un *acting state*.
- **Dual state** è definito come un *wait state* che causa un comportamento per cui la macchina appare come se si trovasse in un *acting state*. Rappresenta un tipo di stato in cui vi sono continue transizioni tra azione e attesa, circolarmente, secondo uno schema logico predefinito. Attualmente l'unico stato di questo tipo è Execute.

Secondo lo standard gli stati sono:

- **Clearing** Si ottiene in seguito ad un comando di *Clear*. In questo stato sono gestiti eventuali errori generati e presenti in Aborted, prima di procedere allo stato Stopped.
- **Stopped** Macchina accesa e ferma. Tutte le comunicazioni con altri sistemi sono funzionanti.
- **Resetting** Si ottiene in seguito ad un comando di *Reset* dallo stato Stopped. In questo stato sono tipicamente messi in tensione tutti i componenti della macchina in attesa di un comando di *Start*.
- **Idle** Reset completato. Mantiene le condizioni raggiunte durante lo stato Resetting.
- **Starting** Si ottiene in seguito ad un comando di *Start*. Esegue le operazioni necessarie all'avvio della macchina; in seguito al completamento dell'operazione viene effettuata la transizione verso lo stato Execute.

- **Suspending** Si ottiene in seguito ad un comando di *Suspend* dallo stato *Execute*. Questo stato è generalmente necessario per la transizione a *Suspended* e prepara la macchina per questo.
- **Suspended** La macchina potrebbe lavorare ad una velocità elevata, ma senza produrre nulla. Questo stato può essere raggiunto a partire da un comando generato dalla logica di macchina, e differisce da *Held* in quanto quest'ultimo è generalmente il risultato di una richiesta dell'operatore.
- **Unsuspending** Si ottiene in seguito ad un comando di *UnSuspend* dallo stato *Suspended*. È necessario per riprendere l'esecuzione del ciclo di produzione e provvede al ripristino delle velocità di setpoint.
- **Execute** Fintanto che la macchina esegue il ciclo di produzione, consumando materiali, si considera nello stato *Execute*. *Execute* è relativo al modo in cui ci si trova: se la macchina è in modo "Cleaning", allora lo stato *Execute* si riferisce all'azione della pulizia.
- **Stopping** Si ottiene in seguito ad un comando di *Stop*. Questo stato esegue le operazioni necessarie a condurre la macchina nello stato *Stop* in modo sicuro.
- **Aborting** Lo stato *Aborting* può essere raggiunto in qualsiasi momento in seguito ad un comando di *Abort* o all'occorrere di un errore. Questo stato esegue le operazioni necessarie a condurre la macchina nello stato *Aborted* in modo rapido e sicuro. L'utilizzo del comando di arresto di emergenza causa l'arresto della macchina ad opera dei suoi sistemi di sicurezza, ma deve altresì generare un comando di *Abort* per la transizione nello stato *Aborting*.
- **Aborted** Questo stato mantiene le informazioni sullo stato della macchina che hanno condotto al comando di *Abort*. In seguito ad un comando di *Stop* viene forzata la transizione verso lo stato *Stopped*.
- **Holding** Si ottiene in seguito ad un comando di *Hold* dallo stato *Execute*. La logica di *Holding* può essere utilizzata per condurre la macchina allo stato *Stop* in modo controllato oppure allo stato *Held*, caratteristico del modo in uso.

- **Held** Questo stato è generalmente utilizzato dall'operatore per fermare temporaneamente la macchina mentre vengono risolti problemi a monte o a valle.
- **Unholding** Si ottiene in seguito ad un comando *UnHold* dallo stato Held. È necessario per riprendere l'esecuzione del ciclo di produzione e provvede al ripristino delle velocità di setpoint.
- **Completing** Questo stato è generalmente il risultato di una transizione automatica dallo stato Execute. Rappresenta la situazione in cui l'operazione di Execute del modo corrente sta volgendo al termine e sono intraprese le azioni necessarie all'arresto della produzione.
- **Complete** La macchina ha terminato l'esecuzione dello stato Completing ed è in attesa di un comando *Stop* per avviare la transizione verso lo stato Stopped.

Osservando il diagramma degli stati per il modo automatico (figura 1.1), che mostra il più esteso ciclo di funzionamento possibile, si nota immediatamente come ogni transizione che inizia da un *Acting state* termina sempre in uno degli altri due tipi di stato in seguito ad un evento di *State complete*, abbreviato in SC nello schema. Tutte le possibili transizioni in corrispondenza delle relative condizioni scatenanti sono schematizzate nella matrice di transizione di stato rappresentata in tabella 1.2.

È evidente come il modello presentato si componga di molti stati e consenta numerose transizioni tra questi in risposta ad altrettanto numerosi eventi esterni. La complessità di questo modello deriva dalla necessità di voler soddisfare le esigenze di organizzazione di un insieme quanto più possibile esteso di macchinari industriali e stili di controllo adottati dagli sviluppatori. Per questi motivi il diagramma a stati di ogni specifica macchina sarà quasi sempre assai più semplice, comprendendo soltanto alcuni degli stati presenti nel modello generale di modo automatico.

### **PackML unit modes**

Uno *unit mode* si riferisce alla condizione attuale e determina come la macchina reagisce in risposta ai comandi ricevuti (coerentemente con l'operazione che si sta eseguendo, ovvero dipendentemente dallo stato in cui ci si trova) ed ha una propria catena

STATO	COMANDO									
	START	RESET	HOLD	UNHOLD	SUSPEND	UNSUSPEND	CLEAR	STOP	ABORT	SC
IDLE	Starting							Stopping	Aborting	
STARTING								Stopping	Aborting	Execute
EXECUTE			Holding		Suspending			Stopping	Aborting	Completing
COMPLETING								Stopping	Aborting	Complete
COMPLETE		Resetting						Stopping	Aborting	
RESETTING								Stopping	Aborting	Idle
HOLDING								Stopping	Aborting	Held
HELD				UnHolding				Stopping	Aborting	
UNHOLDING								Stopping	Aborting	Execute
SUSPENDING								Stopping	Aborting	Suspended
SUSPENDED					UnSuspending			Stopping	Aborting	
UNSUSPENDING								Stopping	Aborting	Execute
STOPPING								Stopping	Aborting	Stopped
STOPPED		Resetting						Aborting		
ABORTING										Aborted
ABORTED							Clearing			
CLEARING								Aborting		Stopped

Tabella 1.2: Matrice di transizione di stato PackML per il modo automatico



è sempre detto che sia possibile cambiare modo di funzionamento in qualsiasi istante a seguito di una richiesta, ma soltanto dopo la verifica di alcune condizioni all'interno della logica di programma. In tabella 1.3 sono descritti alcuni esempi di possibili modi macchina.

Automatic	È il modo utilizzato per il comune schema di produzione continuativa. La macchina esegue comandi inseriti dall'operatore o inviati da un sistema di supervisione.
Maintenance	È un modo utilizzato dal personale di supporto per mettere in funzione la macchina in modo indipendente dal resto della linea di produzione. Tipicamente serve per individuare guasti o eseguire dimostrazioni.
Manual	Fornisce il controllo diretto di ogni servomeccanismo o asse della macchina. Può essere presente o meno a seconda dei vincoli meccanici dell'impianto e può servire principalmente per verificare le operazioni che i meccanismi compiono, il sincronismo di certi assi, la validità di nuovi parametri, le velocità operative, ecc. . .

Tabella 1.3: Modi macchina PackML

### PackML procedural modes

Diversamente da quanto detto sopra, i *procedural modes* non fanno riferimento a modalità di funzionamento specifiche, ma descrivono come la logica di macchina sta operando. Queste modalità, dallo standard ANSI/ISA-88, sono Automatic, Semi-Automatic e Manual, e possono essere utilizzate dall'operatore per modificare il comportamento della logica di controllo durante il funzionamento. Ad esempio in modo logico semi-automatico può essere possibile eseguire i singoli step attraverso la catena degli stati di un particolare modo macchina. La differenza principale tra modi macchina e modi logici è da ricercarsi nel fatto che i primi descrivono il comportamento fisico, ovvero in base all'operazione che si sta eseguendo, mentre i secondi modificano il modo in cui la logica di controllo opera. Note queste premesse viene da sé che è sempre possibile passare da un modo logico ad un altro, indipendentemente dallo stato in cui si trova la macchina; questo avviene contrariamente alle transizioni tra modi

macchina, che tipicamente possono essere effettuati soltanto in stati di attesa o nel solo stato Stopped.

### **PackML unit mode manager**

Come discusso in precedenza, i macchinari industriali hanno diversi altri modi macchina oltre al ciclo automatico. Con la possibilità di avere più modi macchina, dalla versione 3.0 di PackML è stato necessario introdurre un meccanismo, detto *mode manager*, per permettere al progettista della logica di gestire correttamente le transizioni tra modi. Per i procedural modes tale meccanismo non è stato implementato come una routine separata e viene tipicamente integrato nella logica del singolo modo.

Lo scopo principale del gestore è quello di avere una supervisione logica di ciò che avviene a seguito dell'invio di un comando di cambio di modo, verificando la possibilità di soddisfare la richiesta, aggiornando lo stato della macchina mantenendolo consistente e restituendo una risposta, lasciando la logica del modo macchina al di fuori di questa operazione. Il gestore è una routine di alto livello che determina come e in quali stati la macchina può cambiare modo di funzionamento; solitamente è presente un solo gestore e la scelta dei punti di transizione è lasciata all'utente: tipicamente sono collocati in corrispondenza di stati di attesa abilitati in entrambi i modi coinvolti nella transizione oppure nello stato stopped.

In figura 1.2 sono schematizzate alcune richieste di transizioni tra modi che possono venire sottoposte al gestore il quale, in base ad una look-up table preimpostata, può catalogare ognuna di queste come sicura (soddisfacibile, esempio in verde) o non sicura (rischio inconsistenza, esempio in rosso).

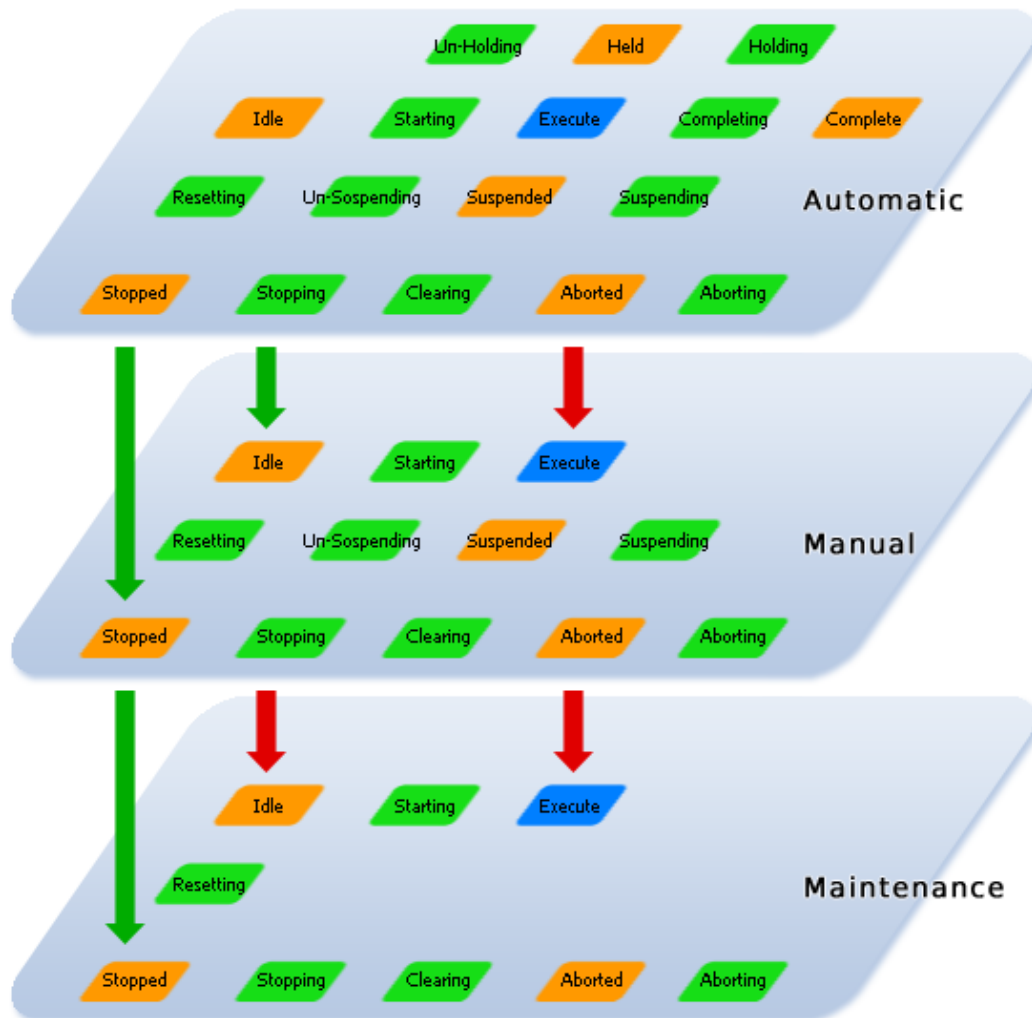


Figura 1.2: Transizioni tra modi di funzionamento. Esempi di richieste soddisfacenti (in verde) e non soddisfacenti (in rosso).

### 1.2.2 PackTags

La missione del team PackML non è limitata a quanto detto sopra, ma riguarda anche la definizione di convenzioni di naming per la comunicazioni tra macchinari nello scenario industriale. Infatti nelle specifiche dei tipi di linea è soltanto il tipo 1 a non prevedere scambio di informazioni verso l'esterno.

Come abbiamo visto precedentemente, il modello fornisce un set di stati macchina il cui scopo è quello di incasellare i comportamenti dei macchinari in uno schema comune e definirne le modalità di funzionamento. Questo terzo insieme di convenzioni ha come obiettivo la definizione dei cosiddetti “tag names”, che includono la struttura per i nomi identificativi dei vari dati (ovvero i tags) oltre ai tipi di dato primitivi con le loro definizioni, range ammissibili e le strutture da essi derivate. In questo modo è possibile leggere il set delle informazioni generate da un macchinario per valutare il contenuto dei singoli tags sia che si tratti di comunicazione orizzontale tra macchine, sia che riguardi una comunicazione verticale verso un sistema informativo di più alto livello.

Lo scambio di informazioni con sistemi di supervisione non è fine soltanto al controllo della produzione, ma anche ad ottenere indicatori utili alla gestione della linea in senso più ampio, ovvero per la valutazione delle prestazioni in relazione agli obiettivi industriali, per la pianificazione delle attività di manutenzione degli impianti, per supportare i responsabili nelle decisioni operative o anche semplicemente per il logging di eventi e valori per eventuali utilizzi futuri. Lo standard non fornisce metodi diretti per il calcolo delle performance, ma si pone come obiettivo quello di rendere disponibile la maggior quantità possibile di dati grezzi, lasciando all’utente la scelta delle metriche da implementare.

Il tempo di uptime e quello speso in ognuno degli stati macchina sono dati importanti per il calcolo degli indicatori di disponibilità del macchinario; lo standard descrive due metodi per fornire queste informazioni, il primo basato sul tempo trascorso conteggiato direttamente dal sistema logico di controllo e il secondo, di tipo event-driven, in cui il supervisore viene notificato ad ogni evento e, aggregando le informazioni ricevute, calcola il tempo speso. In base alle necessità o alle abitudini è possibile seguire indifferentemente uno dei due approcci, entrambi supportati dalle linee guida. Nel primo caso la logica di macchina tiene traccia del tempo trascorso in ogni stato ed in risposta ad una richiesta di un altro sistema può fornire i dati. Nel secondo caso è la logica di macchina che informa ad ogni cambiamento il supervisore tramite un messaggio asincrono: così facendo è necessario registrare gli eventi ed associare ad ognuno di essi un riferimento temporale per calcolare le statistiche in

postelaborazione.

### **Plug-and-Pack**

Le linee guida per i nomi dei tag sono utili per diverse ragioni e molte di queste concorrono al raggiungimento di un obiettivo più ampio, chiamato *Plug-and-Pack<sup>TM</sup>*. Il nome suggerisce immediatamente che si tratta di un sistema mirato a soddisfare le richieste di facilità di integrazione che provengono dai clienti, i quali necessitano di includere nei loro impianti macchinari sempre più eterogenei, anche di costruttori differenti. In questo modo i clienti finali si troverebbero in condizione di poter ridurre sensibilmente gran parte dei costi e delle risorse spese per l'integrazione, ottenendo allo stesso tempo omogeneità nei vari processi industriali da ora supervisionati in modo centralizzato ed efficiente.

Le linee guida sono state pensate per supportare macchinari facenti parte di tutti i tipi di linea che lo standard propone. Il metodo di comunicazione tra le varie piattaforme di controllo può variare a seconda di come il costruttore preferisce, ma in ogni caso deve essere in grado di permettere scambio di informazioni tanto con le altre macchine quanto con sistemi SCADA utilizzando tags predefiniti per l'indirizzamento. Generalmente i dati sono scambiati utilizzando standard di comunicazione basati su Ethernet, come OPC (OLE for process control).

### **Tag types**

L'insieme dei tag è diviso in tre categorie principali: controllo, stato e amministrazione. La prima categoria raccoglie tutto quanto necessario all'interfacciamento tra macchinari e supervisore di linea per invio di comandi o settaggio di parametri. I tags di stato rappresentando dati "prodotti" e sono quelli utili ai sistemi di più alto livello o per la visualizzazione su pannelli operatore. Per meglio comprendere la distinzione, si può considerare ad esempio il tag *PMLc.UnitMode*, che è utilizzato come comando per inviare una richiesta di cambio modo oppure *PMLs.UnitModeCurrent* che è il valore risultante dalla procedura di cambio modo ed è di sola lettura. La categoria dei tags amministrativi include tutti quelli che rappresentano informazioni specifiche relative

al macchinario, caratterizzati dal prefisso *PMLa*, normalmente usati per scambiare dati di performance (ad esempio metriche Overall Equipment Effectiveness, OEE) oppure notifiche di eventi macchina.

### Tag prefixes

Dal momento che PackTags potrebbe essere utilizzato in sistemi che già implementano altre convenzioni di naming per lo scambio dei dati, è necessario che ogni tag abbia un prefisso identificativo per essere distinto da eventuali altri con lo stesso nome. Tutti i control tag sono identificati dal prefisso *PMLc* e consentono accesso in lettura e scrittura, gli status tag utilizzano *PMLs* e consentono accesso in sola lettura, mentre i tags di amministrazione sono individuati da *PMLa* con accesso in sola lettura. È previsto l'utilizzo della notazione puntata e del mixed case per i nomi dei tag, analogamente a quanto si fa per tipi di dati strutturati.

### Data types

Spesso i sistemi di elaborazione sono più efficienti quando lavorano con tipi di dato nativi. Per esempio molti PLC permettono di operare su byte quando il core che effettua realmente i calcoli utilizza double word effettuando continue conversioni implicite: per questo motivo i tag utilizzano generalmente il tipo intero a 32 bit. In tabella 1.4 sono schematizzati e descritti i tipi di dato utilizzabili all'interno delle convenzioni PackTags.

Integer	32 bit, da 0 a 4294967295 senza segno
Real	32 bit IEEE-754 standard floating point (valore massimo 16,777,215)
Binary	Maschere di bit
String	ASCII null-terminated, 80 caratteri
Structure	Tipicamente utilizzato da sistemi di più alto livello

Tabella 1.4: Tipi di dato PackTags

Non sono presenti formati per date/time stamping in quanto non supportati. I riferimenti temporali sono gestiti dai sistemi di alto livello, al fine di evitare l'insor-

gere di problemi di sincronizzazione del tempo tra supervisione e logica di controllo macchina.

### 1.3 Lo standard Weihenstephan

Lo standard Weihenstephan è sviluppato dal dipartimento Food Packaging Technology della Technische Universität München (TUM) all'interno di un programma di ricerca svolto in collaborazione con le maggiori industrie tedesche del settore food&beverage. Si è imposto come lo standard di fatto del settore in Europa. È, infatti, possibile accreditarsi come membri presso l'organizzazione per poter sottoporre le dichiarazioni di conformità allo standard dei propri macchinari ed utilizzare il logo "WS Ready". Attualmente il lavoro di standardizzazione è completo e viene mantenuto tramite i soli aggiornamenti necessari; i documenti sono disponibili in forma di licenza personale per i soli membri.

Il Weihenstephan Standard, abbreviato WS, definisce un protocollo di comunicazione per la connessione tra macchinari industriali e sistemi di acquisizione dati di alto livello (PDA o MES). WS si propone di favorire un utilizzo di tipo plug-and-play per l'acquisizione dati da macchinari compatibili tramite un meccanismo di sincronizzazione automatico basato su file XML che contengono la descrizione dei parametri macchina. Inoltre lo standard fornisce istruzioni per l'analisi dei dati raccolti, template ed esempi per la generazione dei report e linee guida per garantire il funzionamento sicuro ed affidabile dei sistemi di acquisizione dati. In sostanza WS si propone come base per la realizzazione di sistemi PDA economici, funzionali e che favoriscano l'interoperabilità tra macchinari di diversi produttori nel settore dell'imbottigliamento e del confezionamento, specificando tutto quanto necessario per l'implementazione, dalla struttura di rete sino all'insieme minimo di dati che devono essere messi a disposizione dalla macchina e al modo in cui le informazioni devono essere scambiate.

Il documento di specifica in esame (WS versione 2005.05) è costituito da quattro parti riguardanti i principali ambiti oggetto della standardizzazione:

- Part 1: Physical Interface Specification [6] contiene la specifica dell'interfaccia fisica di comunicazione tra macchinari e sistemi PDA. Qui viene anche definita la struttura del file XML di descrizione macchina.
- Part 2: Content Specification of the Interface [7] definisce quali dati devono essere disponibili, in quale formato devono essere scambiati e le convenzioni di

naming dei tag.

- Part 3: Data Evaluation and Reporting contiene le funzioni per l’elaborazione dei dati e report di esempio da utilizzare come modello.
- Part 4: Inspection and Safe Operation è una guida per la configurazione e l’utilizzo dei sistemi PDA nei momenti successivi all’installazione, al fine di garantirne la sicurezza e l’affidabilità.

Nella prima parte WS fornisce la specifica dell’interfaccia di comunicazione da utilizzare; riferendosi alla pila ISO/OSI indica per ogni livello quale protocollo o interfaccia fisica o standard utilizzare, come schematizzato in figura 1.3.

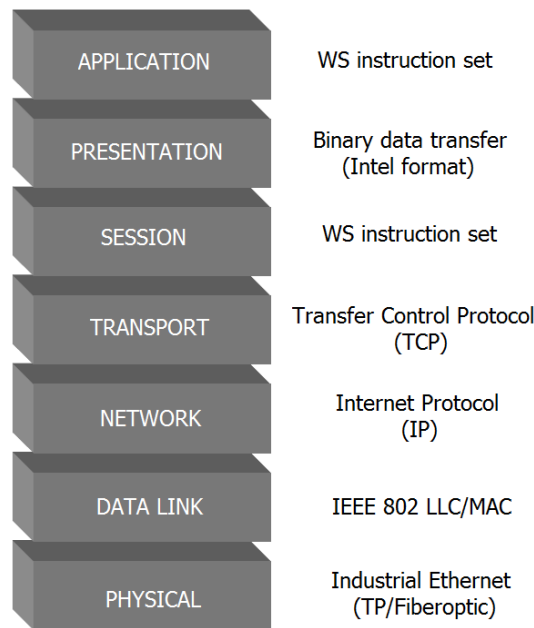


Figura 1.3: Stack ISO/OSI per WS

Per lo scambio delle informazioni è definito un set di istruzioni che realizza un protocollo client-server; ogni comunicazione viene incapsulata in un frame di dimensione costante in modo da semplificare al massimo l’utilizzo del protocollo su dispositivi PLC. Nel caso specifico dello scambio dati tra macchina e sistema PDA, quest’ultimo

agisce da client inviando comandi di lettura e scrittura di tags presenti nei controllori di macchina, che agiscono come server. È prevista l'eventuale presenza di un "master PLC" che operi come concentratore: in questo caso assolverà il compito di raccogliere dati dai PLC periferici e scambiare dati con il sistema PDA operando come client verso i primi e come server nelle comunicazioni verso il secondo.

Per rispettare le specifiche della parte 1, ogni macchina o sistema PDA deve soddisfare diversi requisiti. Oltre ad utilizzare una interfaccia di rete conforme, deve essere obbligatoriamente presente il supporto ad alcuni comandi per lettura e scrittura di base di parametri, mentre le opzioni più avanzate (trasferimenti multipli, lettura e scrittura di parametri stringa, trasferimento di file di configurazione) non sono necessarie per la certificazione. Ogni risposta ricevuta deve necessariamente contenere un codice che rappresenta eventuali errori generati nel tentativo di soddisfare la richiesta, oppure la conferma di corretta esecuzione del comando.

Siccome ogni dispositivo non ha le stesse funzionalità di un altro e non offre lo stesso identico set di dati, è necessario avere una descrizione di ciò che è disponibile. Questa descrizione è memorizzata in un file in formato XML (che deve obbligatoriamente chiamarsi "PDACONF.XML") e sfrutta a fondo le potenzialità di questo formato permettendo di estrarre le informazioni in esso contenute da qualsiasi combinazione applicazione/piattaforma utilizzata.

### **1.3.1 Modello degli stati WS**

Per descrivere la situazione operativa di un macchinario è necessario fare riferimento al concetto di stato. Durante il funzionamento si susseguono diverse configurazioni e il macchinario può ritrovarsi in uno stato qualsiasi tra quelli disponibili. WS utilizza un modello di stato che mette in rilievo le anomalie nel modo più diretto e specifico possibile, schematizzato in figura 1.4 con indicate le possibili transizioni.

### **1.3.2 WS tags**

Dal momento che i macchinari generano una grande quantità di informazioni durante il loro ciclo di funzionamento, è necessario che questi dati siano strutturati in modo



per citare qualche esempio, quello degli allarmi, piuttosto che quello dei parametri o quello degli stati macchina. In questo modo semplicemente conoscendo nome e id di un particolare dato è possibile associarlo immediatamente al tipo di informazione che esso veicola.

### Tipi di dato

I tags veicolano informazioni di vario genere e per questo lo standard mette a disposizione alcuni tipi di dato primitivo da utilizzare per la rappresentazione delle informazioni da scambiare a seconda della loro natura, senza eccedere nella varietà per mantenere comunque il protocollo semplice da implementare. In tabella 1.5 sono descritti i tipi supportati.

Unsigned32	32 bit, intero senza segno
Signed32	32 bit, intero con segno
Real	32 bit IEEE-754 standard floating point
Hex32	32 bit, esadecimale senza segno utilizzato per la rappresentazione compatta di maschere di bit
String16	stringa composta di caratteri UTF-8 o UTF-16

Tabella 1.5: Tipi di dato WS

Oltre al tipo, ogni dato è caratterizzato da un livello di accesso da parte di sistemi esterni. Ci sono dati che devono solamente essere letti, ovvero tutti quelli che riguardano lo stato della macchina (i contatori, gli allarmi, ecc. . . ), mentre per altri (principalmente comandi o impostazioni) è necessario poter scrivere. Sono previsti tre livelli di accesso: sola lettura (read only, R), sola scrittura (write only, W), lettura e scrittura (read and write, RW).

### 1.3.3 Rispetto delle linee guida OMAC

Per sua natura WS si candida come standard nell'ambito specifico dell'industria europea food&beverage proponendo soluzioni e funzionalità che mirano a soddisfare le

esigenze di coloro che operano in questi settori. Nonostante WS sia fortemente specializzato e fornisca direttive per tutto ciò che viene coinvolto nel processo industriale in senso più ampio, il lavoro del sottogruppo PackML di OMAC è comunque considerato di rilevante importanza nella stesura di WS, in quanto vi sono molti punti in comune. Nello specifico, la parte 2 di WS contiene una sezione che introduce procedure ed elementi per soddisfare le linee guida OMAC relative al modello degli stati e alle convenzioni di naming, nel tentativo di armonizzare i due standard in modo che i sistemi “WS Ready” possano essere considerati compatibili PackML.

Il modello degli stati WS, nonostante sia diverso da quello proposto da OMAC, ne rispetta le specifiche e tramite una operazione di mapping è possibile osservare le corrispondenze tra i due modelli in oggetto (figura 1.5). Analizzando lo schema appaiono alcune differenze:

- in Weihenstephan lo stato *Suspended* è suddiviso in sottostati per descrivere in modo più preciso le fasi di transizione da e verso lo stato *Operating*
- lo stato *Operating* corrisponde direttamente allo stato OMAC *Execute*
- lo stato *Aborted* è suddiviso in sottostati, allo stesso modo di *Suspended*, per meglio identificare le principali cause di guasto

Analogamente a quanto previsto per gli stati macchina, tutte le altre specifiche OMAC sono rispettate e in WS part 2 sono descritte le corrispondenze tra unit modes, procedural modes e tag names, anche se non tutto ciò che è previsto in PackML/PackTags è realizzato effettivamente in WS, dal momento che quest’ultimo implementa soltanto funzionalità utili ai sistemi PDA (ma è estensibile, se necessario!).

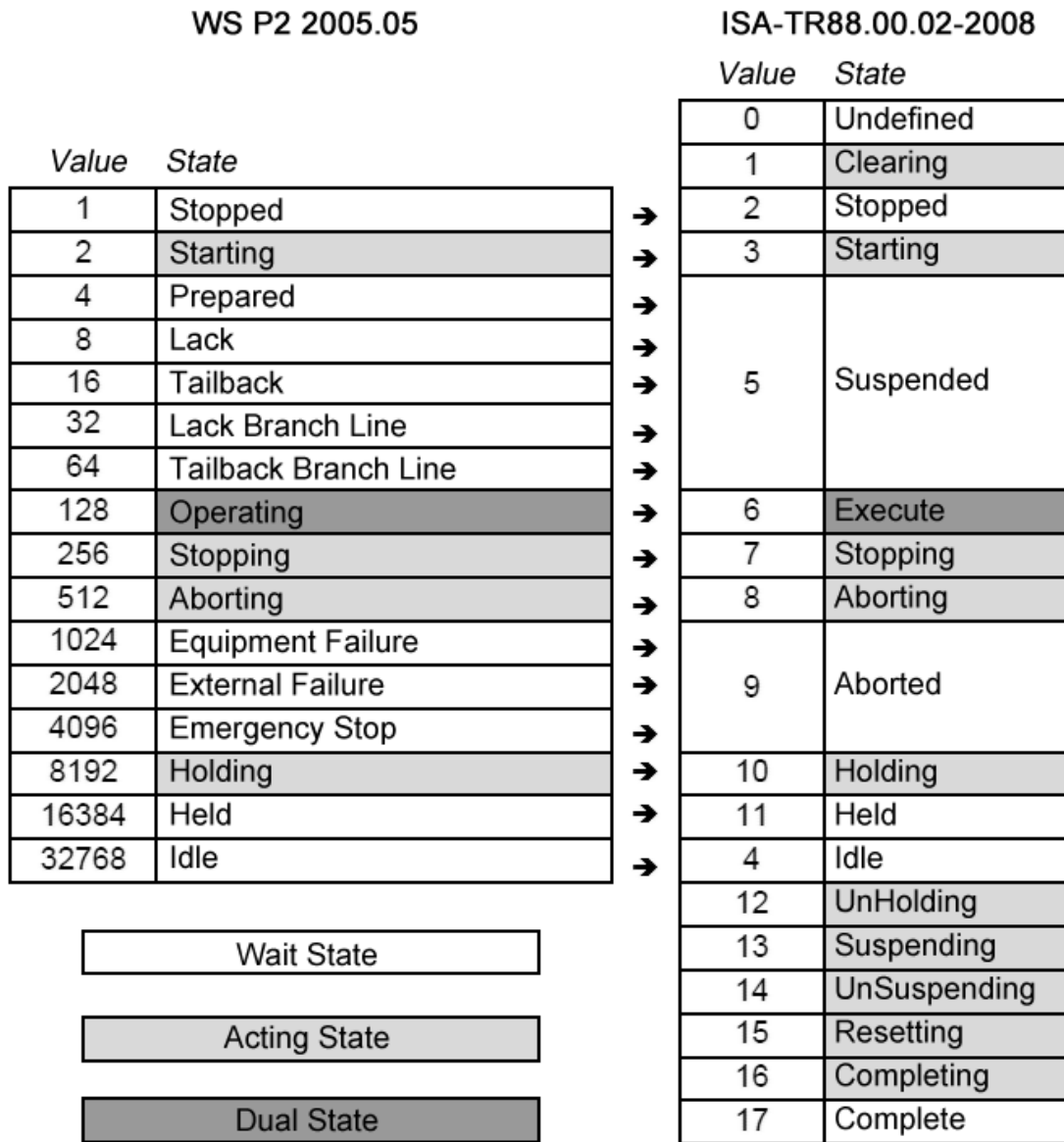


Figura 1.5: Mappatura degli stati WS su modello PackML

# Capitolo 2

## Requisiti e tecnologie

In questo capitolo sono illustrati gli obiettivi della tesi insieme ad una panoramica degli strumenti e delle tecnologie utilizzate, approfondendone gli aspetti che li rendono rilevanti dal punto di vista del supporto all'approccio al problema.

### 2.1 Obiettivi

Il lavoro descritto è finalizzato alla realizzazione di una libreria per un controllore industriale, Programmable Logic Controller (PLC), che permetta di sviluppare software nel rispetto delle specifiche dello standard proposto da OMAC. Successivamente, e al fine di poter valutare il funzionamento di un sistema “OMAC-compliant” basato sulla libreria sviluppata, sarà necessario simularne la condizione operativa in un contesto nel quale più attori, come ad esempio un'interfaccia uomo macchina (Human Machine Interface, HMI) ed eventualmente un sistema di raccolta dati di tipo SCADA o OPC possano scambiare informazioni con il controllore di una macchina automatica. Dopo un'adeguata fase di simulazione e collaudo la libreria potrà essere utilizzata per lo sviluppo di funzioni PDA e SCADA in macchine ed impianti reali.

Una schematizzazione del sistema di simulazione da realizzare è rappresentata in figura 2.1. In un sistema di questo tipo, ogni controllore dovrà eseguire il codice della libreria per simulare il modello di funzionamento OMAC, mentre il pannello

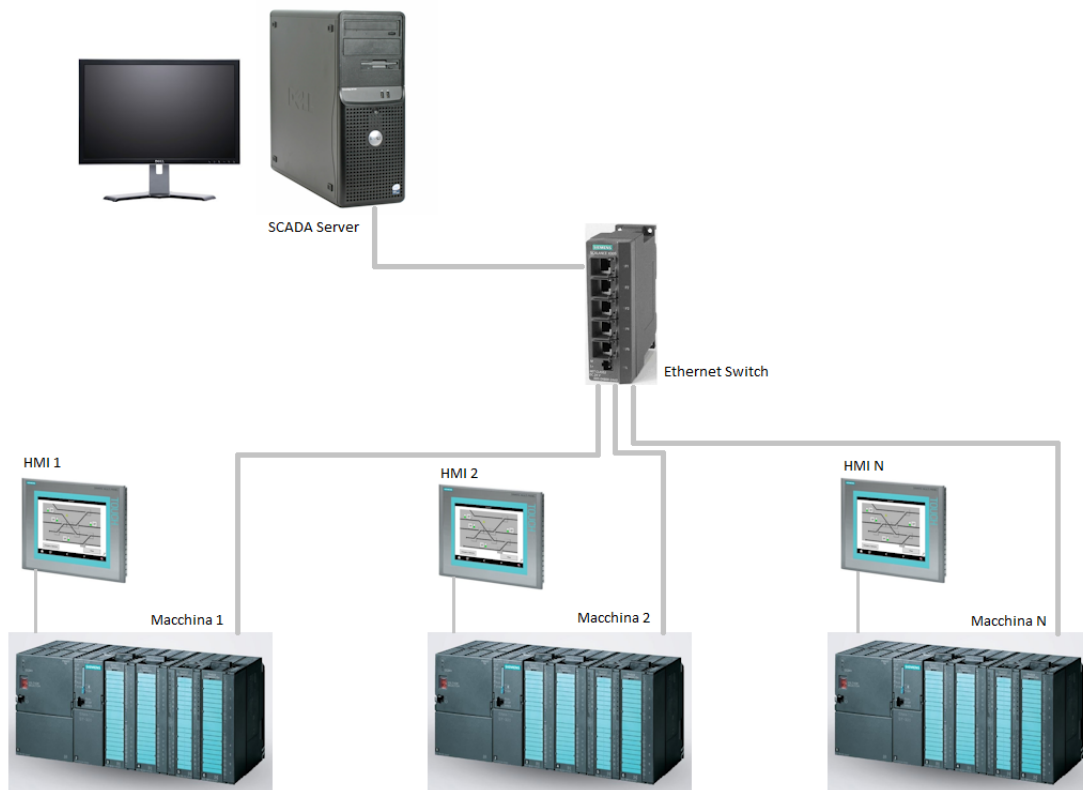


Figura 2.1: Schema del sistema di simulazione da realizzare

operatore ed il sistema di supervisione dovranno prelevare informazioni sullo stato della macchina ed inviare comandi secondo le convenzioni di comunicazione prescritte dallo standard.

## 2.2 Requisiti

I requisiti principali riguardano l'ambiente e gli strumenti di lavoro da utilizzare nello sviluppo. La libreria è destinata ad essere impiegata nell'ambito della programmazione di macchinari industriali basati su dispositivi Siemens Simatic, sia per la parte di controllo logico che per le HMI, ragion per cui questo lavoro dovrà realizzare una soluzione "OMAC-compliant" su piattaforma Siemens tramite gli strumenti software che vengono forniti a corredo. Un ulteriore requisito è quello di sviluppare il codice,

per quanto possibile, utilizzando diagrammi KOP, in modo da estendere la possibilità di comprendere ed utilizzare il codice al maggior numero di utenti possibile. Questa scelta è orientata al voler ottenere codice interpretabile e manutenibile anche da chi non possieda conoscenze di programmazione procedurale. Un ultimo requisito riguarda l'organizzazione del software, che dovrà essere composto da blocchi funzionali utilizzabili "a scatola chiusa" in qualsiasi altro contesto, basandosi solamente sulla loro interfaccia e senza conoscerne l'implementazione, necessitando al più di parametri di configurazione atti a modificarne il comportamento.

I requisiti sopra descritti appartengono tutti alla categoria dei "non funzionali", ovvero quella classe di requisiti che non impone nessun vincolo al comportamento o alle funzionalità del software e si contrappongono ai requisiti "funzionali" che in questo caso sono rappresentati dalle specifiche dello standard. L'unico requisito funzionale e quantitativo imposto dal contesto in cui questo tipo di sistemi opera riguarda il dover fornire una garanzia di esecuzione in un tempo deterministico, e in assoluto molto ristretto, dell'ordine del millisecondo. Questo vincolo nasce dalle caratteristiche dei processi da controllare in ambito industriale, nei quali la maggior parte delle decisioni deve essere intrapresa in base alle misurazioni di grandezze molto velocemente variabili. Vincoli progettuali di questo tipo, unitamente alla natura dell'hardware da programmare, che è pesantemente ottimizzato per l'esecuzione di operazioni logiche combinatorie in memoria di lavoro e scarsamente prestante per elaborazioni general purpose ed accessi alla memoria di massa, rendono critico il compito di realizzare software orientato ad elaborazioni di strutture dati complesse. Nella sezione 3.3 saranno descritti, fra le altre cose, gli accorgimenti pratici adottati per rendere il codice efficiente al fine di soddisfare i requisiti funzionali.

## 2.3 Strumenti di lavoro

In questa sezione viene presentata una breve panoramica delle soluzioni e degli strumenti di sviluppo Siemens utilizzati per la realizzazione del sistema. Sono principalmente prodotti commerciali rivolti a costruttori o integratori di sistemi di automazione,

supportati da pacchetti software dedicati alla scrittura dei programmi per i dispositivi e alla loro gestione e configurazione.

### 2.3.1 Simatic S7-300

La famiglia di controllori PLC Siemens Simatic S7-300 è specificamente progettata per essere parte di soluzioni di automazione industriale. Questo tipo di controllore è modulare e fisicamente adatto all'utilizzo in ambienti cosiddetti "ostili" per i dispositivi elettronici in genere, come ad esempio quadri elettrici industriali, in cui le interferenze elettromagnetiche e le sollecitazioni meccaniche possono costituire gravi problemi per il funzionamento delle apparecchiature elettroniche standard.

Il cuore del sistema è rappresentato dalla CPU. La famiglia Simatic S7-300 comprende una gamma di CPU [8] che si differenziano per capacità di elaborazione, disponibilità di memoria e presenza di interfacce di comunicazione. La possibilità di accoppiare moduli di I/O analogici e digitali rende questo prodotto ideale per il controllo di processi e macchinari industriali. Per sfruttare in modo corretto tutte le funzionalità della CPU ed ottenere le migliori prestazioni in termini di efficienza nell'esecuzione delle operazioni è necessario conoscerne, almeno in linea di principio, l'architettura ed il funzionamento, che saranno descritte di seguito.

#### Architettura

Dal punto di vista dell'architettura hardware, i moduli CPU sono caratterizzati dalla presenza di una unità di elaborazione e da tre aree di memoria separate (figura 2.2). In dettaglio:

- **Memoria di caricamento** Si trova nella Micro Memory Card (MMC) inserita nel dispositivo e corrisponde esattamente alle dimensioni della scheda. Viene utilizzata per la memorizzazione dei blocchi di codice e dei blocchi di dati nonché di tutti i blocchi che non sono rilevanti per l'esecuzione, come quelli contenenti i dati di progetto o i dati di sistema (configurazione, collegamenti, parametri delle unità, ecc.). Per sua natura si tratta di una memoria ritentiva.

- **Memoria di sistema** È integrata nella CPU e non può essere ampliata. Contiene l'area di Merker (caratteristica dei dispositivi Siemens e della quale si tratterà in seguito, nella sezione 2.3.2), i temporizzatori, i contatori, i registri immagine degli ingressi e delle uscite ed i dati locali dei blocchi. È possibile stabilire in fase di progettazione quali parti di questa memoria debbano essere ritentive.
- **Memoria di lavoro** La memoria di lavoro è integrata nella CPU e non può essere ampliata. Viene utilizzata esclusivamente per elaborare il codice ed i dati del programma utente. I blocchi in esecuzione caricati in questa memoria il cui attributo di ritenzione è selezionato vengono mantenuti anche in caso di riavviamento o mancanza di rete, altrimenti sono inizializzati con i valori presenti nella memoria di caricamento.

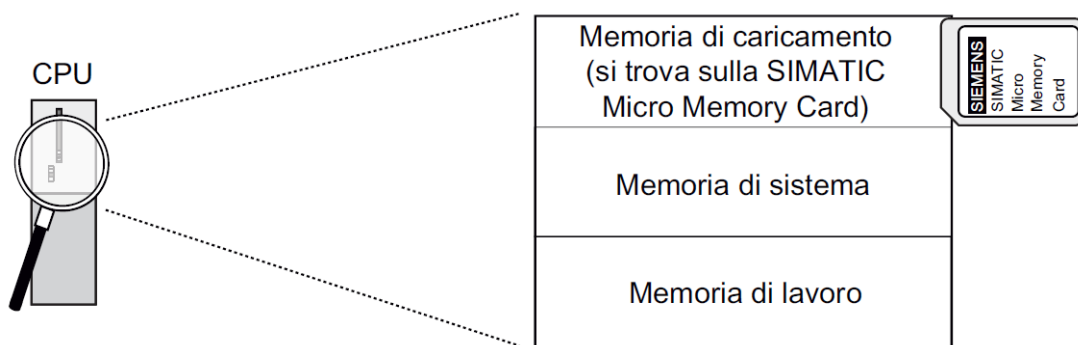


Figura 2.2: Aree di memoria delle CPU 31xC/31x

## Connettività

Per quanto riguarda la connettività, necessaria alla programmazione e configurazione dell'unità nonché alla comunicazione con altri dispositivi durante l'esecuzione di programma, ogni CPU dispone di almeno una interfaccia tra le seguenti:

- **Multi Point Interface** abbreviato MPI, è una interfaccia proprietaria sviluppata da Siemens per i PLC Simatic S7. Consente un'organizzazione della rete a bus Master/Slave e velocità di trasferimento fino a 187.5Kbps.

- **Process Field Bus Decentralized Peripherals** abbreviato Profibus DP, è una interfaccia standardizzata IEC [9, 10] il cui sviluppo è stato ispirato da MPI di Siemens, ed è principalmente utilizzata come bus di campo per la comunicazione tra controllore e periferiche da controllare come sensori ed attuatori. Consente velocità di trasferimento fino a 12Mbps su bus Master/Slave.
- **Profinet IO** è l'interfaccia più evoluta tra quelle offerte ed è, come la precedente, standardizzata IEC [9, 10]. Può essere considerata come una versione robusta, affidabile e deterministica del protocollo IEEE 802.3 e viene per questo comunemente definita Industrial Ethernet.

Soltanto i modelli di punta, caratterizzati dalla sigla PN/DP, offrono possibilità di interconnessione su rete Profinet, mentre un'interfaccia combinata MPI/Profibus DP è sempre disponibile, eventualmente anche con più di una porta fisica. Per questo motivo le CPU offrono funzionalità di routing tra le varie interfacce, al fine di consentire l'accoppiamento delle reti dei vari tipi, in modo da rendere agevole l'accesso a tutti i dispositivi anche nel caso in cui si disponga di una connessione fisica ad una sola delle periferiche in rete. In figura 2.3 è schematizzato un esempio di connettività con accoppiamento tra reti eterogenee.

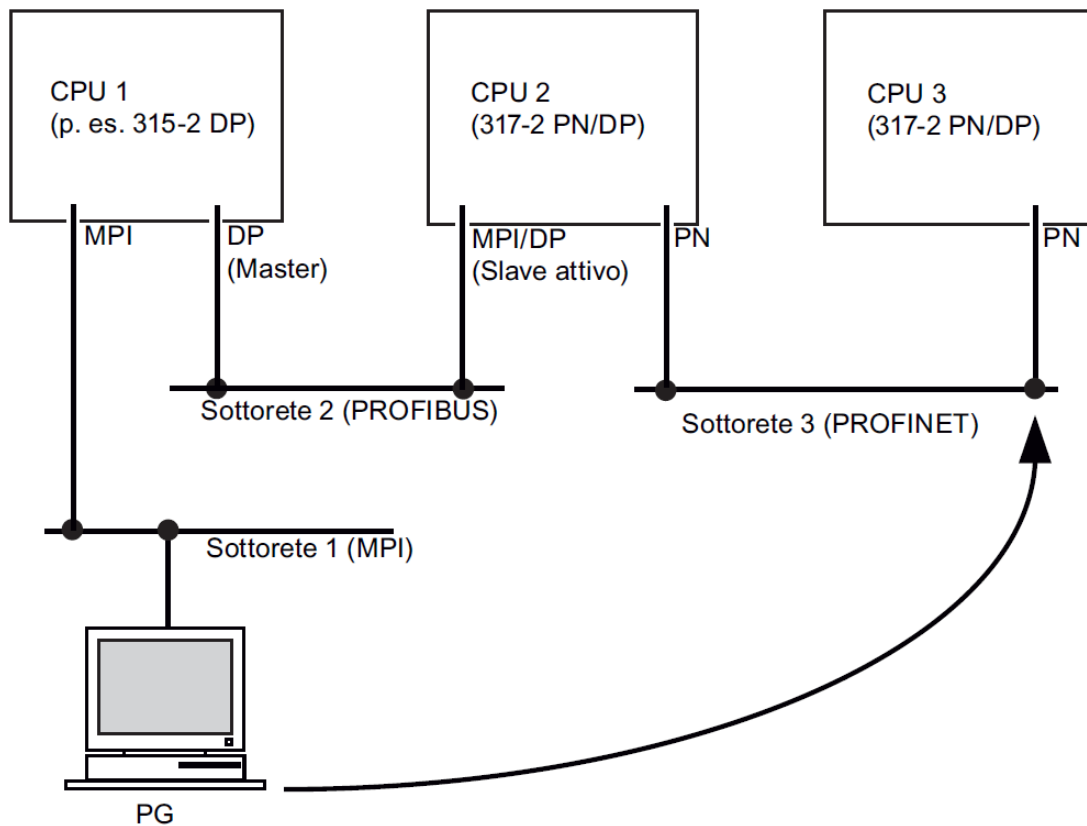


Figura 2.3: Connettività delle CPU 31xC/31x.

Esempio di accoppiamento di reti di diverso tipo sfruttando le funzionalità di routing tra le interfacce integrate nelle CPU.

### Funzionamento

Per quanto riguarda il funzionamento, il comportamento di un PLC durante l'esecuzione (stato di RUN) è di tipo ciclico e si svolge secondo quattro operazioni principali (figura 2.4) che compongono il cosiddetto "ciclo di scansione":

1. **Immagine Processo Uscite (IPU)** La CPU scrive i valori dell'immagine di processo delle uscite nelle unità di uscita.
2. **Immagine Processo Ingressi (IPI)** La CPU legge lo stato degli ingressi dalle unità degli ingressi e aggiorna l'immagine di processo degli ingressi.

3. **Programma utente** La CPU elabora il codice utente ed esegue le operazioni indicate nel programma in modo sequenziale.
4. **Punto di Controllo del Ciclo (PCC)** Alla fine di un ciclo il sistema operativo esegue le operazioni in attesa, come ad esempio il caricamento o la cancellazione di blocchi, oppure attende che sia trascorso il tempo impostato come minima durata del ciclo.

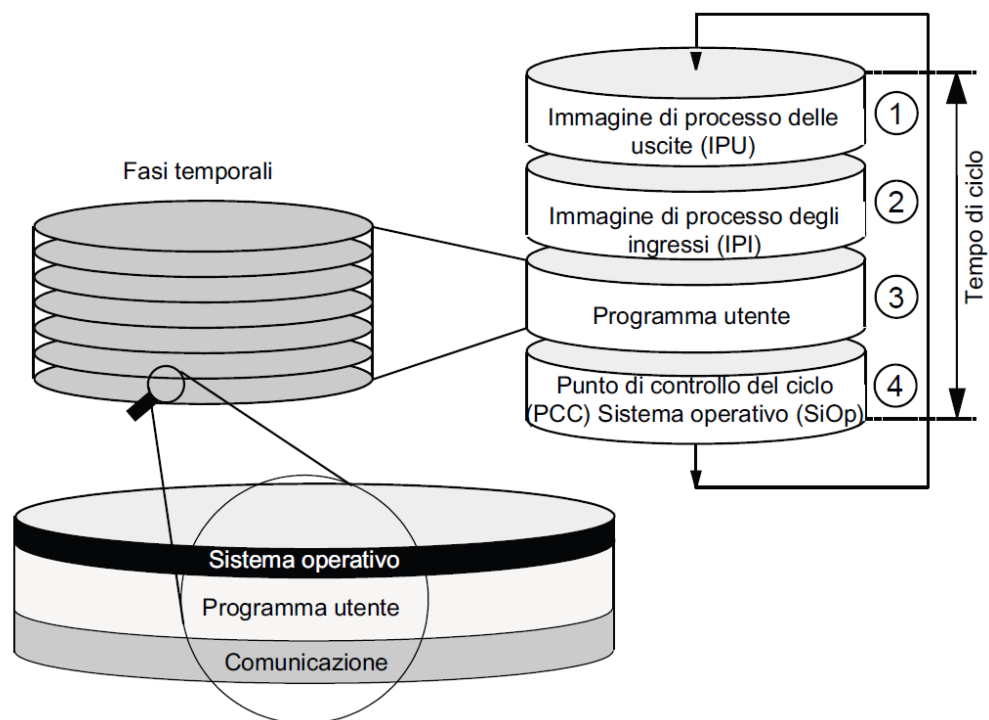


Figura 2.4: Ciclo di scansione delle CPU 31xC/31x

In questa successione ciclica di operazioni, tra le quali viene anche eseguito il codice utente, è importante definire la misura del tempo che intercorre tra l'inizio di un ciclo ed il successivo, detta "tempo di ciclo" o "tempo di scansione" (figura 2.4), poiché nelle applicazioni di controllo in tempo reale è necessario conoscere la durata dell'esecuzione del programma di controllo con garanzie deterministiche. Le CPU Simatic

calcolano il dato autonomamente e lo rendono disponibile in una locazione di memoria di sistema insieme ai limiti massimo e minimo raggiunti dall'ultimo avviamento.

Le operazioni 1 e 2 sono caratteristiche dei sistemi PLC e del modo in cui questi utilizzano le porte di I/O presenti nei moduli periferici e per questo motivo necessitano di un breve approfondimento. In questi sistemi il codice utente non può operare direttamente sui dispositivi periferici, ma deve utilizzare aree di memoria dedicate, dette immagine di processo degli ingressi (Eingang, E) ed immagine di processo delle uscite (Ausgang, A), che sono rispettivamente aggiornate dal sistema operativo sulla base degli ingressi e utilizzate dallo stesso per comandare le uscite, prima dell'esecuzione del programma. Lo svantaggio principale è che in questo modo non è possibile modificare lo stato di una porta di uscita durante l'esecuzione del codice, anche se fanno eccezione le uscite analogiche che sono trattate secondo una diversa politica (aggiornate in modo sincrono).

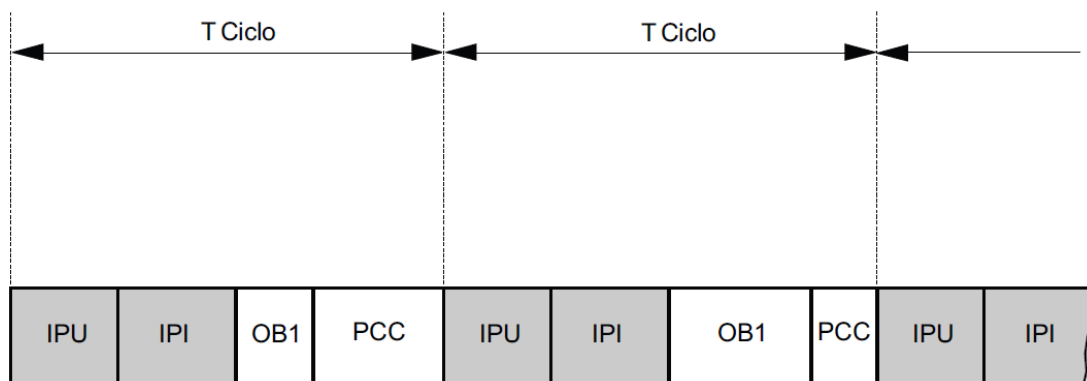


Figura 2.5: Ciclo di attuazione in tempo reale.

IPU e IPI sono elaborati in istanti fissi al variare del tempo di esecuzione del programma utente (OB1).

Dal punto di vista delle applicazioni industriali, invece, questa organizzazione si sposa perfettamente con le necessità ed i principi del controllo in tempo reale, in cui le garanzie di determinismo sono irrinunciabili. Si pensi ad esempio ai casi in cui sia necessario effettuare campionamenti o attuazioni sincrone: configurando opportunamente i parametri di ciclo della CPU, ed in particolare il tempo minimo di scansione, è possibile ottenere una schedulazione ciclica dei compiti in grado di soddisfare qualsia-

si necessità di controllo in tempo reale indipendentemente dalla variazione dei tempi di elaborazione del programma. In figura 2.5 è mostrato un esempio di schedulazione di questo tipo, in cui al variare del tempo di esecuzione del programma utente (OB1) il tempo di ciclo rimane costante al fine di elaborare IPU e IPI in istanti determinati.

### 2.3.2 Simatic STEP7

STEP 7 è il pacchetto software di base utilizzato per la gestione dei sistemi Simatic [11]. In questo ambiente è possibile progettare la configurazione hardware delle stazioni Simatic, scrivere il software per applicazioni di dimensioni o complessità anche elevate ed integrare in un unico progetto anche le componenti esterne del sistema, come le interfacce HMI. Il componente principale, Simatic Manager (figura 2.6), è il gestore dei progetti e consente l'accesso a tutte le funzionalità dell'ambiente da un'unica interfaccia.

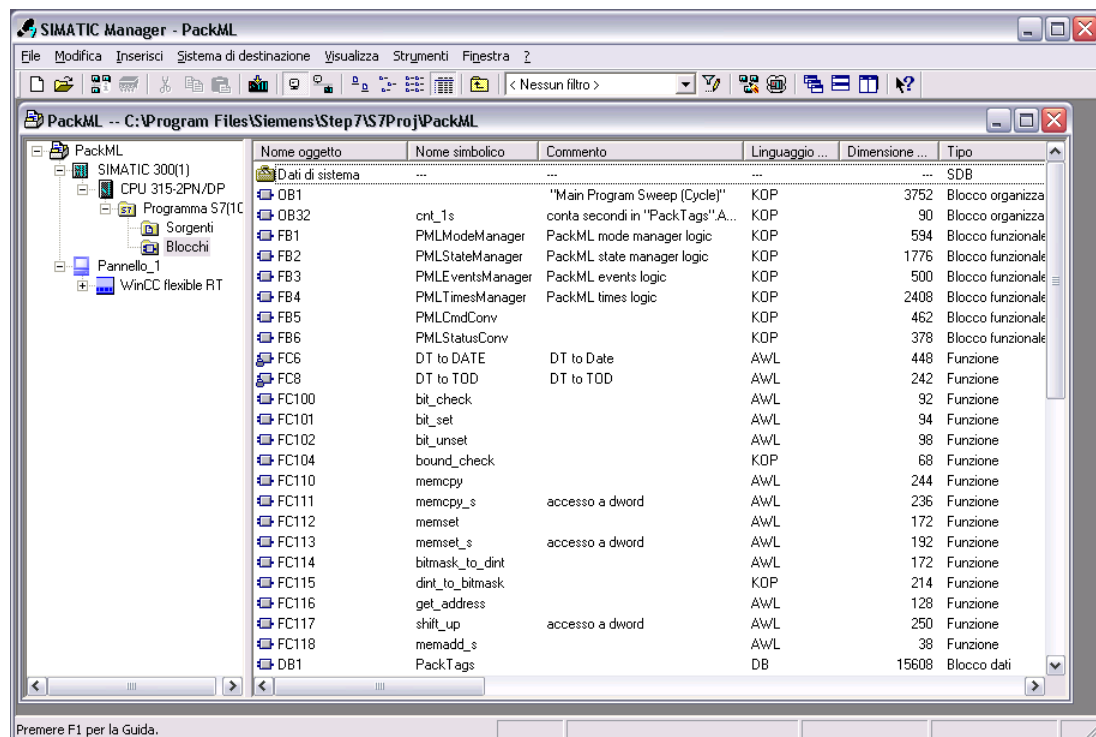


Figura 2.6: Simatic Manager

In ambiente STEP7, l'organizzazione del software è fatta secondo uno schema a blocchi, in cui ogni elemento può essere di diverso tipo e contenere definizione di dati o programmi utente. Le principali strutture sono:

- **Blocchi dati (DB)** Servono alla memorizzazione dei dati utente. Consentono di allocare memoria non volatile in modo statico e referenziabile simbolicamente.
- **Blocchi funzionali (FB)** Fanno parte dei blocchi programmabili e contengono codice utente. Un blocco di questo tipo dispone di un DB associato, detto “di istanza”, in cui sono appoggiati i parametri di ingresso, di uscita e statici. Un FB può, quindi, mantenere dati statici tra una chiamata e la successiva poiché al termine dell'elaborazione i dati memorizzati nel DB di istanza non vanno perduti, come invece accade alle variabili temporanee, memorizzate nello stack dei dati locali.
- **Funzioni (FC)** Fanno parte dei blocchi programmabili e contengono codice utente. A differenza dei FB, non hanno alcun DB di istanza e non possono quindi utilizzare dati statici o parametri di default, potendo contare esclusivamente su dati di ingresso e locali temporanei per le elaborazioni o, al più, sull'accesso a blocchi dati globali. Al termine dell'elaborazione dell'FC, i dati locali vanno perduti.
- **Blocchi Organizzativi (OB)** Rappresentano l'interfaccia tra il sistema operativo ed il programma utente. Vengono richiamati autonomamente dal dispositivo secondo un meccanismo di interruzione con priorità e sono utilizzati come punto di ingresso per l'esecuzione del codice; tipicamente i programmi richiedono elaborazioni cicliche di logiche di controllo e la maggior parte del codice risiede all'interno del blocco OB1 (o in altri blocchi da questo richiamati) che viene eseguito una volta per ciclo di scansione ed ha priorità minima. Una possibile struttura del programma utente è rappresentata in figura 2.7.

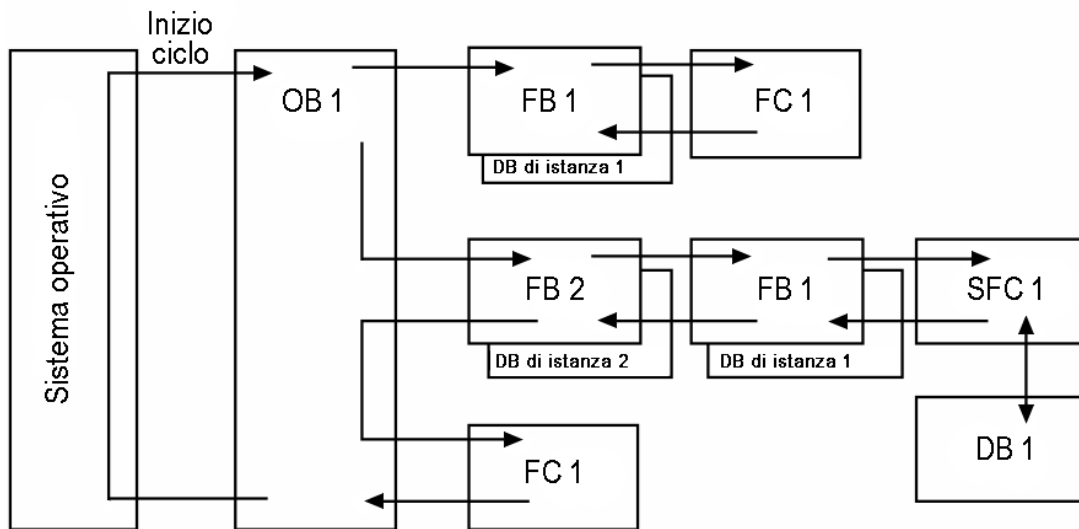


Figura 2.7: Chiamate annidate in un programma STEP7

Dopo aver descritto i blocchi DB, è necessario spendere qualche parola riguardo la memoria Merker (M), che risiede nell'area di memoria di sistema ed è presente in un quantitativo che può variare da qualche centinaio a qualche migliaio di KByte, a seconda del modello di CPU. Questa memoria è a disposizione dell'utente, ha ambito globale ed è di tipo statico, inoltre può essere resa ritentiva tramite la configurazione dei parametri di sistema. In caso di applicazioni che non richiedano dati strutturati, può offrire una valida alternativa all'uso dei DB poiché, anche se è spesso meno estesa della memoria di caricamento dove questi risiedono, garantisce migliori prestazioni dal punto di vista dei tempi di accesso. Anche trattandosi di valori dell'ordine del microsecondo, i tempi di accesso alla memoria M sono generalmente di circa la metà rispetto a quelli dei DB, ed in applicazioni sensibili potrebbero comunque rappresentare l'unica possibilità di ottenere l'efficienza desiderata.

Fatte le premesse relative agli aspetti architetturali di interesse per la programmazione, è possibile descrivere i metodi e gli strumenti per la scrittura dei programmi utente. Malgrado questi dispositivi possano essere ritenuti nei normali calcolatori elettronici, lo sviluppo dei linguaggi di programmazione ad essi dedicati ha seguito una via differente rispetto a quanto è avvenuto per le CPU general purpose. I PLC veniva-

no inizialmente programmati con una tecnica basata sugli schemi logici delle schede a relè e questo eliminava la necessità per i primi utilizzatori (prevalentemente tecnici elettricisti) di possedere nozioni di programmazione procedurale. In passato, inoltre, i produttori di tecnologia hanno favorito l'uso di strumenti di programmazione di tipo proprietario, ed in particolare ciascuno ha creato il proprio dialetto di programmazione (spesso basato su un linguaggio cosiddetto "a contatti" o sul linguaggio macchina). Questo fatto ha reso difficile l'acquisizione di competenze di programmazione universali e l'eventuale migrazione dei programmi tra PLC di diversi produttori, costituendo un problema per gli utilizzatori.

Con il passare del tempo le tecniche di sviluppo del software si sono ammodernate anche in questo campo e quindi il primo linguaggio a contatti è stato via via affiancato da linguaggi di programmazione più moderni ed efficienti. Recentemente, per risolvere il problema dell'interoperabilità, il comitato elettrotecnico internazionale, International Electrotechnical Commission (IEC), ha emanato lo standard IEC 61131-3 [12] che definisce alcuni linguaggi di programmazione per PLC. Di seguito sono elencati i linguaggi base rispondenti allo standard messi a disposizione da Siemens:

- **KOP (schema a contatti)** è un linguaggio di programmazione grafico e corrisponde al *Ladder Diagram* IEC. La sintassi delle istruzioni assomiglia ad uno schema di circuito. KOP consente all'utente di seguire in modo semplice il flusso dei segnali tra contatti, blocchi funzionali e bobine.
- **AWL (lista istruzioni)** è un linguaggio di programmazione testuale vicino al linguaggio macchina e corrisponde all'*Instruction List* IEC. Quando si crea un programma in AWL, le singole istruzioni corrispondono in larga misura alle operazioni con le quali la CPU elabora il programma.
- **FUP (schema funzionale)** è un linguaggio di programmazione grafico che rappresenta la logica mediante i box dell'algebra booleana e corrisponde al *Function Block Diagram* IEC. Esso consente inoltre di rappresentare funzioni complesse (ad es. le funzioni matematiche) direttamente in connessione con i box logici.

In ambiente STEP7, la programmazione può sempre essere effettuata utilizzando i linguaggi base, attraverso un editor dedicato. Altri linguaggi di programmazione, come il più evoluto S7-SCL che permette programmazione di più alto livello con una sintassi derivata dal pascal, sono disponibili in opzione e non utilizzati nel lavoro oggetto di tesi.

### **2.3.3 Simatic S7-PLCSIM**

L'ambiente STEP7 può integrare componenti opzionali, i cosiddetti "Engineering Tools", tra i quali il software S7-PLCSIM [13] che consente di eseguire e testare i programmi utente in un PLC simulato senza la necessità di essere collegati ad alcun componente hardware S7. Questo software mette a disposizione del programmatore un'interfaccia utente grafica (figura 2.8) per visualizzare e modificare le variabili dei programmi di controllo, eseguire il programma simulato nei modi di scansione a ciclo singolo o continuo e modificare il modo di funzionamento del controllore. È completamente trasparente a STEP7, è possibile caricare i blocchi di programma nel PLC simulato allo stesso modo in cui si utilizzerebbe un dispositivo reale fisicamente connesso ad una qualsiasi delle interfacce di comunicazione simulate e si può ad esempio utilizzare la tabella delle variabili per controllare la memoria. Sono disponibili quasi tutte le funzionalità del dispositivo reale, ad eccezione di quelle inderogabilmente legate all'hardware o alle porte di I/O che possono eventualmente non rispecchiare in modo esatto il comportamento di una vera CPU. Ad esempio la misurazione dei tempi di scansione, che dipendono fortemente dall'hardware, non è da ritenere affidabile se fatta in simulazione.

### **2.3.4 Simatic HMI**

Nel momento in cui i processi diventano sempre più complessi e le esigenze di funzionalità di macchine e impianti aumentano, l'operatore necessita della massima trasparenza. Tale trasparenza è ciò che viene offerto dalla HMI, che svolge molteplici compiti tra quelli descritti qui di seguito:

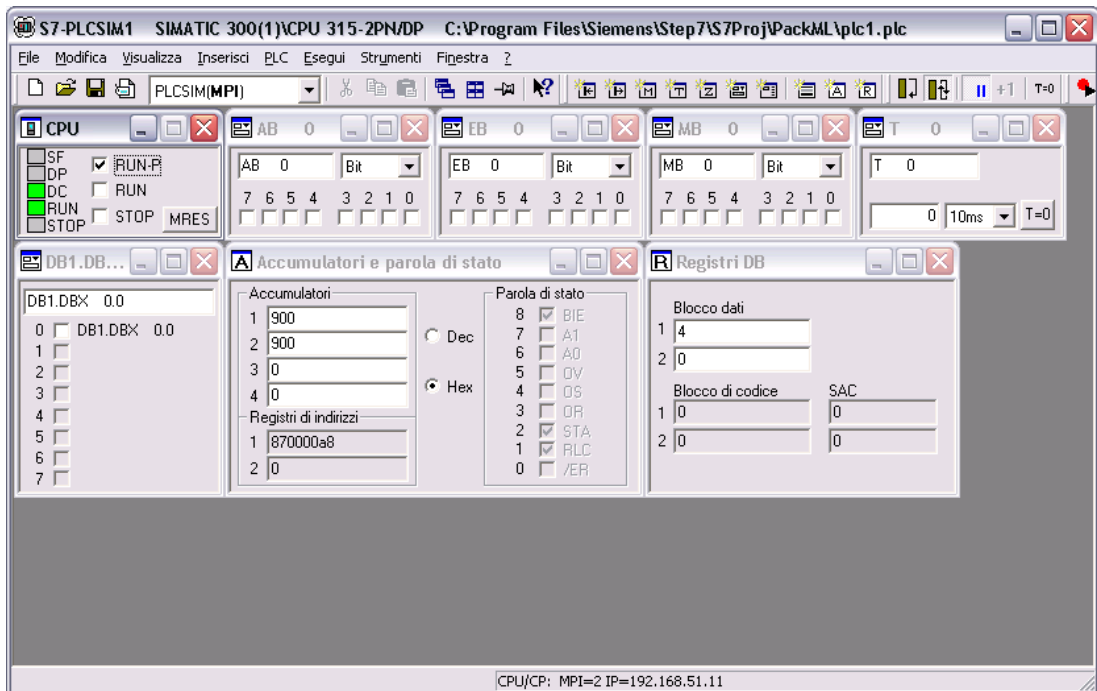


Figura 2.8: Interfaccia utente di S7-PLCSIM

- **Rappresentazione del processo** sul pannello operatore. Se ad esempio nel processo cambia uno stato, la visualizzazione sul pannello operatore viene in tal senso aggiornata.
- **Comando del processo** utilizzando l'interfaccia grafica. L'operatore può ad esempio indicare un valore di riferimento per il controllore o inviare un comando operativo.
- **Visualizzazione di eventi** di varia natura a cui possono essere associati allarmi di tipo acustico o visivo.
- **Archiviazione dati** nella memoria interna del sistema HMI. In tal modo è possibile documentare l'andamento del processo e avere accesso anche in un secondo tempo ai dati di produzione pregressi.
- **Gestione dei parametri macchina e di processo** che possono essere salvati nel database del sistema HMI. Tali parametri possono poi essere richiama-

ti e trasmessi al controllore per modificare il comportamento del macchinario controllato.

La soluzione Simatic HMI mette a disposizione una vasta gamma di possibilità per assolvere molteplici operazioni di servizio e supervisione. Per la realizzazione di interfacce HMI sono disponibili soluzioni hardware dedicate. I pannelli Simatic MP della serie 277 [14] sono progettati per l'uso in combinazione con PLC di vario tipo e di diversi produttori, tra i quali ovviamente i Simatic S7-300, o anche con sistemi di supervisione. I dispositivi sono basati su piattaforma software Microsoft Windows CE 5.0 e possiedono capacità di elaborazione e memoria locali, interfacce di tipo tattile o con tasti e display LCD ad elevato contrasto per la visualizzazione (figura 2.9).

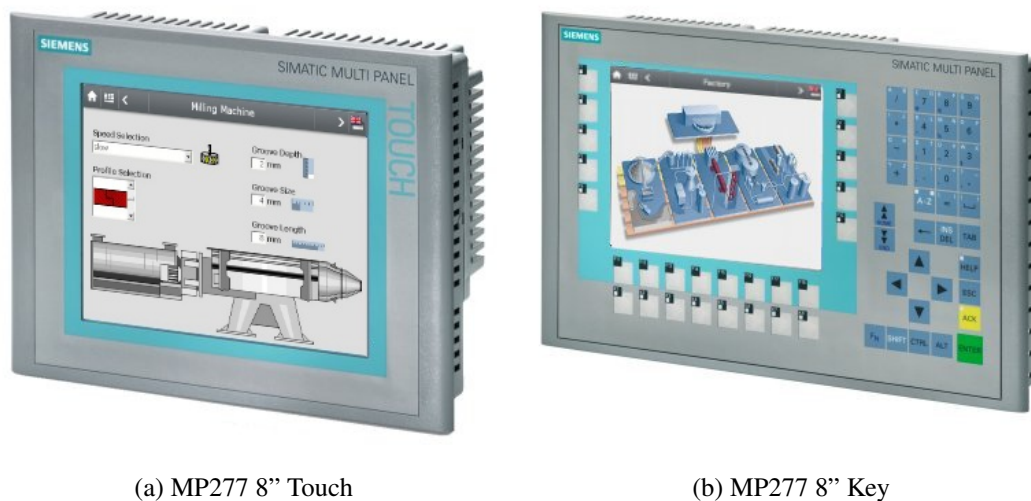


Figura 2.9: Dispositivi Simatic HMI

WinCC flexible [15] è l'ambiente di sviluppo per costruire e programmare interfacce HMI da eseguire su pannelli Simatic. Consente la creazione dei cosiddetti tags, ovvero variabili locali aggiornate ciclicamente con valori prelevati dalla memoria di un altro dispositivo collegato in rete; è anche possibile effettuare piccole elaborazioni locali tramite il linguaggio di scripting VBScript. Tramite un'interfaccia grafica (figura 2.10) permette di comporre le finestre delle HMI. È disponibile una libreria di controlli grafici e sono previsti meccanismi dinamici di visualizzazione e trasformazione

di questi, basati sui valori dei tag: in questo modo è possibile realizzare interfacce animate che riflettano lo stato dei sistemi controllati, evolvendo nel tempo.

A completamento della panoramica riguardo gli strumenti essenziali per lavorare con Simatic HMI, troviamo WinCC flexible Runtime [16]. Quest'ultimo componente è il framework di supporto per l'esecuzione di interfacce create con il software di progettazione WinCC flexible ed è organizzato secondo l'uso di finestre e controlli grafici. Ogni sistema sul quale si voglia eseguire un progetto WinCC flexible deve prevedere il componente di runtime, sia esso un pannello Simatic oppure un PC industriale opportunamente configurato. La caratteristica più interessante dal punto di vista dello sviluppo riguarda quindi il fatto che il runtime può essere utilizzato anche sullo stesso sistema in cui il progetto è stato creato, a fini di test e senza la necessità di essere fisicamente connessi ad hardware Simatic.

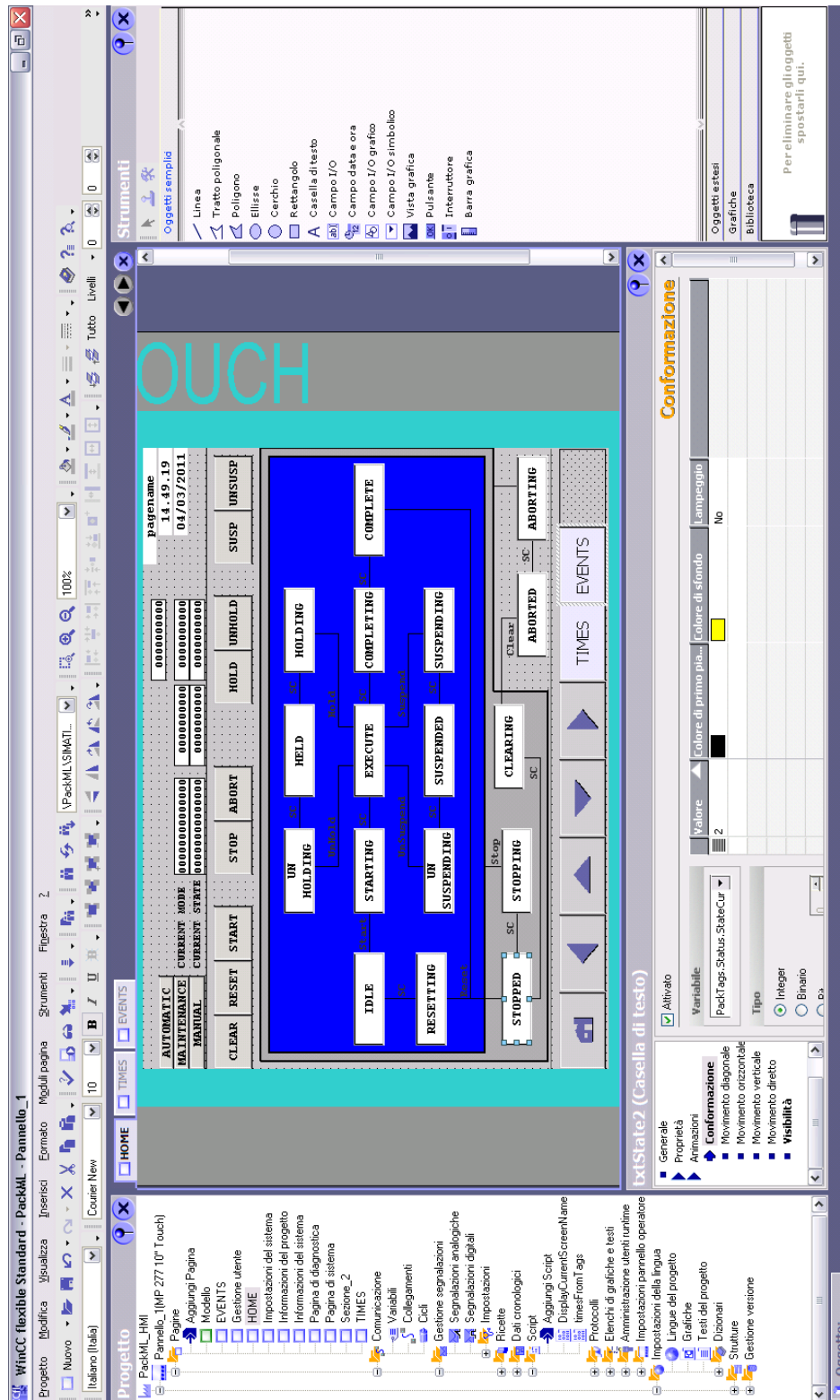


Figura 2.10: Interfaccia utente di WinCC flexible 2008

## Capitolo 3

# Progettazione e realizzazione

In questo capitolo è illustrato l'approccio al problema e le scelte che caratterizzano il sistema realizzato. Nelle sezioni appropriate saranno dettagliate le fasi progettuali e realizzative di tutti i componenti del sistema di test, descrivendo soluzioni concettuali e tecniche impiegate.

### 3.1 Approccio seguito

Lo sviluppo di un sistema come quello oggetto di questa tesi coinvolge più componenti ognuno dei quali deve soddisfare determinati requisiti e la cui realizzazione, specialmente in quei numerosi casi in cui è necessario interagire con gli altri, deve avvenire nel modo più razionale ed efficiente possibile. È quindi opportuno svolgere una fase progettuale rigorosa, volta allo studio ad alto livello del problema ed alla definizione di linee guida di sviluppo che permettano di affrontare in modo agevole le problematiche che si prevede possano emergere nella successiva fase di realizzazione.

Per quanto riguarda il software, è scontato che la progettazione e la scrittura debbano avvenire secondo quell'insieme di regole e convenzioni dell'ingegneria che sono comunemente indicate come "best practices". Seguendo questo modello di astrazione e scomposizione modulare si possono realizzare sistemi facili da mantenere ed estremamente scalabili a fronte di uno sforzo modesto quanto naturale per certi versi, mirato a mantenere un'alta coesione ed un basso accoppiamento tra i componenti.

Inoltre, si pone il problema della scelta dei metodi per la verifica della corretta funzionalità del sistema. Durante la fase di sviluppo deve essere prevista la possibilità di testare i singoli componenti definendone le modalità, eventualmente fornendo in ingresso insiemi di dati sintetici allo scopo di mettere in risalto tutti i possibili scenari di funzionamento.

Per ultimo, è necessario definire le metriche ed i dati che forniranno le misure di prestazione e di rispetto dei requisiti, nonché un componente che possa fornire un riscontro visivo del funzionamento del sistema.

## 3.2 Scelte progettuali

Lo standard proposto da OMAC offre una soluzione completa per lo sviluppo del software di controllo di macchine industriali fornendo specifiche ed indicazioni per ogni aspetto da considerare, dallo stile di programmazione alla logica a stati, fino all'organizzazione delle strutture dati. In questo modo lo standard mira a realizzare quello che viene definito come “common look and feel”, all'insegna dell'omologazione e dell'integrazione orizzontale e verticale tra i diversi attori che partecipano al processo industriale. Questo tipo di approccio, per sua natura, non offre soluzioni efficaci nei casi in cui si rende necessario adeguare software esistente allo standard; i tentativi di conversione verso il modello a stati PackML richiedono una profonda conoscenza di quanto si va a modificare e potrebbero causare problemi al codice, minandone la robustezza. Volendo comunque integrare macchinari con software non conforme all'interno di linee con supervisione “OMAC compliant”, è possibile limitarsi ad effettuare la mappatura delle informazioni da e verso le strutture dati prescritte dallo standard, consci di aver realizzato un approccio parziale e di aver speso tempo per la scrittura di codice difficilmente riutilizzabile in futuro.

Per quanto riguarda il lavoro di tesi, si realizzerà una libreria scrivendo nuovo software e verrà pertanto seguito un approccio completo, che include l'utilizzo di mode/state manager PackML e delle convenzioni di naming PackTags. In questo modo il codice sarà organizzato in maniera modulare fin dall'inizio, costruendo un modello riutilizzabile nella sua interezza o eventualmente nei singoli blocchi funzionali.

La realizzazione della parte software per PLC è un passaggio fondamentale nello sviluppo del sistema. Più della realizzazione stessa, è importante la definizione dei blocchi funzionali che comporranno il software, ognuno dei quali assolverà un compito ed eventualmente ne richiamerà al suo interno altri incaricati di eseguire funzionalità di più basso livello (primitive), all'insegna della programmazione modulare e del riuso del software. Dovranno essere definite le interfacce in modo funzionale, per permettere l'integrazione tra i blocchi, e le strutture dati per la comunicazione interna ed esterna, secondo quanto prescritto dallo standard. È facile capire come questo passaggio rappresenti una criticità nel processo di sviluppo del software e quanto sia importante produrre un modello efficiente, razionale e riutilizzabile. In ambito industriale, specialmente tra integratori ed OEM che sono i soggetti più spesso impegnati ad affrontare il problema della stesura dei software di controllo e dell'adeguamento dei propri prodotti agli standard internazionali, la tematica della progettazione razionale dovrebbe ricoprire un ruolo importante.

A dimostrazione dell'importanza di quanto appena detto, si riscontra un grande interesse anche tra fornitori di tecnologia e aziende leader che, oltre ad aderire a comitati e gruppi di lavoro per la standardizzazione, si adoperano per la diffusione delle innovazioni in un settore in cui chi realizza impianti è spesso restio a portare cambiamenti che non siano orientati al mercato o comunque alle richieste dei propri clienti. Dalla volontà di spingere nella direzione dell'innovazione e della diffusione dello standard OMAC nascono progetti mirati a fornire guide implementative, modelli software ed esempi di realizzazione. Alcuni di questi esempi sono stati analizzati durante la fase di progettazione e di seguito se ne darà una breve panoramica critica, sottolineando l'apporto che ciascuno di essi ha fornito al progetto sviluppato in questa tesi.

### **3.2.1 Guida alla realizzazione Procter&Gamble**

In tempi recenti, successivi alla standardizzazione di PackML in ISA-88, è stata resa disponibile una guida open source per la realizzazione della macchina a stati secondo l'interpretazione di P&G [17]; tale guida consiste in un insieme di documenti descrittivi della struttura dei blocchi funzionali del software ed è corredata da una progetto

di esempio in ambiente Rockwell RSLogix5000 per controllori Allen-Bradley. P&G ha sviluppato questa guida con l'intento di fornire un supporto agli sviluppatori nel realizzare software secondo le specifiche PackML seguendo i principi della scomposizione modulare S88 [18]. OMAC Packaging Workgroup ha adottato questo insieme di documentazione ed esempi come supporto ufficiale dandone visibilità sul sito istituzionale in modo gratuito per tutti gli utenti, ed incoraggia i fornitori di tecnologia allo sviluppo di software di esempio che segua la filosofia della guida P&G.

All'interno del pacchetto sono presenti alcune Add-On Instructions (AOI), contenenti le logiche PackML, che possono essere fundamentalmente interpretate come blocchi funzione parametrizzati e di conseguenza realmente utilizzate come traccia per lo sviluppo delle proprie routines. Nello specifico:

- **AOI.PackML** È il gestore dei modi e della macchina a stati. Dal punto di vista della logica interna è un buon esempio di implementazione realizzato in linguaggio a contatti, fin troppo chiaro (a scapito dell'efficienza) per certi aspetti. La sua interfaccia è complessa, poco razionale, ed utilizza molte variabili in ingresso ed in uscita che veicolano informazioni anche ridondanti. Ciò non è necessariamente un errore od uno svantaggio; generalmente potrebbe essere il risultato di una scelta realizzativa fine al poter lavorare con dati che permettano di avere un codice più efficiente, più semplice, più elegante o che comunque consentano di perseguire un vantaggio di qualche altro tipo. Nel caso di questa implementazione è semplicemente il risultato del voler minimizzare il numero di blocchi funzionali, integrando più funzioni e passaggi in ognuno, a spese della razionalità dell'interfaccia.
- **AOI.PackMLModeStateTimes** È il gestore dei tempi. L'interpretazione della sua logica non è immediata come per il blocco precedente; per quanto riguarda la sua struttura interna e l'interfaccia, valgono le stesse considerazioni fatte riguardo ad AOI.PackML. In questo caso, inoltre, viene fatto pesante uso di strutture dati locali statiche sulle quali sono inutilmente appoggiati calcoli intermedi senza riguardo all'uso razionale della memoria.

Entrando nel merito del documento e soprattutto dell'implementazione di esempio fornita a corredo, emerge subito come un'eventuale (e ragionevole) idea di farne un porting su piattaforma Siemens sia immediatamente da scartare. Il software è organizzato secondo i principi di scomposizione modulare e l'ambiente Rockwell facilita l'approccio secondo questo schema gerarchico che mette in corrispondenza le sezioni del codice con i moduli dell'impianto da controllare, funzionalità sfruttata massicciamente nel progetto di esempio il cui schema a blocchi secondo lo standard S88 è mostrato in figura 3.1. Inoltre la maggior parte delle logiche, anche quelle scritte con il linguaggio a contatti, utilizza meccanismi di indirizzamento indiretto non disponibili in STEP7 e ciò, unitamente alla possibilità di concatenare sequenze di operazioni anche complesse, rende impossibile creare software per apparati Siemens che approcci il problema allo stesso modo.

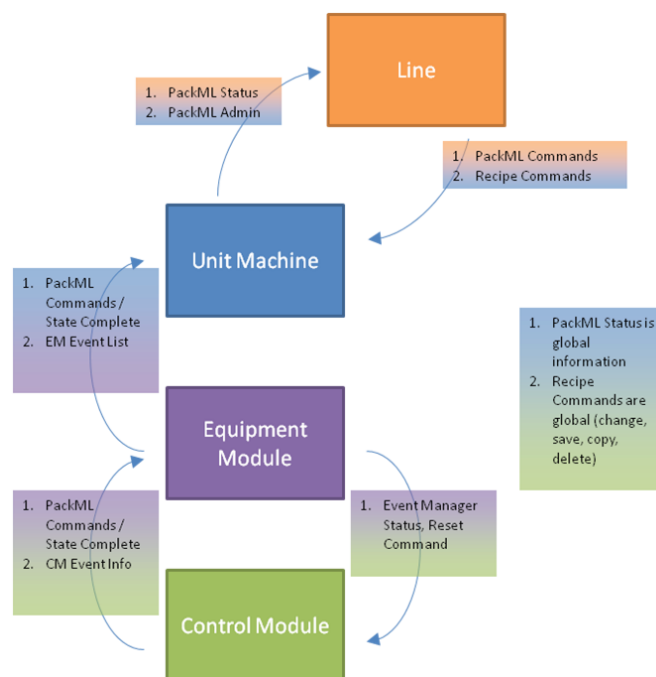


Figura 3.1: Scomposizione modulare S88 dell'applicazione P&G

### 3.2.2 Siemens OPL

Anche Siemens ha fatto proprio lo standard di OMAC utilizzandone le specifiche all'interno del pacchetto commerciale Optimized Packaging Line (OPL) [19], realizzando blocchi funzione parametrizzati e blocchi dati precompilati per l'ambiente STEP7. In questo caso la libreria software LPMLV30 e la documentazione fornita [20] sono gratuiti per i clienti, ma l'implementazione è protetta. A causa della protezione, non è possibile analizzare le logiche interne dei blocchi, ma è consentito soltanto l'utilizzo in un progetto software senza possibilità di modifica del codice o delle interfacce.

La libreria consiste di due moduli e di una interfaccia dati; lo schema logico secondo il quale i blocchi dovrebbero essere collegati per garantire il corretto flusso delle informazioni è rappresentato in figura 3.2.

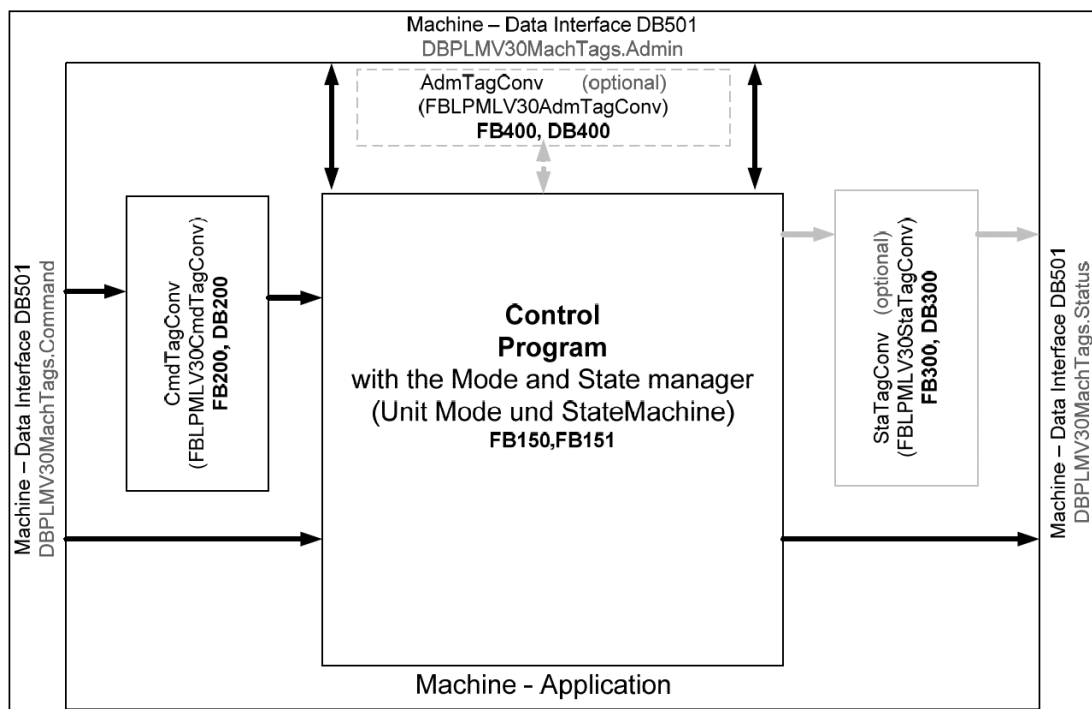


Figura 3.2: Struttura della libreria Siemens LPMLV30

Il primo modulo è utilizzato per la gestione di modi, stati e tempi secondo le specifiche dello standard ed è realizzato con i seguenti blocchi software:

- **FBLPMLV30ModeManager** È il gestore di modo e può gestire fino ad otto modi, di cui tre predefiniti e cinque personalizzabili. Deve essere richiamato ad ogni ciclo di scansione e richiede la conoscenza dello stato corrente per poter verificare le condizioni di transizione, che nel caso specifico possono avvenire solamente nello stato Stopped. È possibile abilitare una funzionalità di registrazione delle operazioni di cambio modo effettuate.
- **FBLPMLV30StateMachine** È il gestore di stato e si occupa di controllare la corretta esecuzione della macchina a stati PackML. Deve essere richiamato ad ogni ciclo di scansione per verificare la presenza di comandi ed effettuare le transizioni necessarie; richiede la conoscenza degli stati abilitati nel corrente modo di funzionamento. È possibile abilitare una funzionalità di registrazione delle operazioni di cambio stato effettuate.
- **FBLPMLV30PMLTimes** È il gestore dei tempi e si occupa dell'aggiornamento delle relative strutture PackTags. Richiede la conoscenza del modo e dello stato corrente.

Il secondo modulo è utilizzato per la manipolazione o la conversione dei formati delle variabili PackTags da e per le interfacce dei blocchi funzionali. È realizzato con i seguenti blocchi software:

- **FBLPMLV30CmdTagConv** Permette di decodificare i valori di comando *UnitMode*, *ProcMode* e *CntrlCmd* dalla struttura PackTags a segnali binari più agevolmente utilizzabili dai blocchi funzione.
- **FBLPMLV30AdmTagConv** Permette di scomporre la struttura *Parameter* in array al fine di rendere più agevole l'accesso ai i tag di amministrazione nella struttura PackTags. Il suo utilizzo è opzionale.
- **FBLPMLV30StaTagConv** Permette di scomporre la struttura *Parameter* in array al fine di rendere più agevole l'accesso ai i tag di stato nella struttura PackTags. Il suo utilizzo è opzionale.

L'interfaccia dati è contenuta in un DB globale *DBLPMLV30MachTags* in cui sono assemblati vari tipi di dato personalizzati secondo la struttura prescritta dallo standard.

Da un punto di vista strutturale, la libreria è progettata in modo razionale. I formati PackTags sono efficienti per lo scambio di informazioni tra sistemi di alto livello, ma più laboriosi da utilizzare nel ciclo di elaborazione di un PLC: l'idea di inserire strati di interfaccia tra i tags ed i blocchi funzionali, che si occupino di convertire i dati per l'elaborazione e riconvertire i risultati, è pienamente condivisibile. Meno efficaci sono, invece, le interfacce dei blocchi che, al fine di permettere un utilizzo di dati immediati, richiedono moltissimi parametri in ingresso e ne restituiscono altrettanti in uscita, col risultato di appesantire inutilmente i diagrammi in cui vengono richiamati.

### **3.3 Scelte realizzative**

Una parte consistente del lavoro riguarda la realizzazione di una libreria di supporto per software di controllo industriale conforme allo standard di OMAC ed i requisiti progettuali ne impongono lo sviluppo su piattaforma Siemens con controllori Simatic S7-300, descritta nella sezione 2.3.1. Non è, però, scontato che le scelte realizzative debbano rispecchiare quelle della libreria Siemens, anche perché quest'ultima è stata analizzata in un momento in cui lo sviluppo del sistema era già ad uno stato avanzato. Nonostante ciò, al momento dello studio della documentazione fornita è emerso come, nella pratica, vi fossero già alcune caratteristiche comuni mentre altre, ritenute utili, sono state integrate "in corso d'opera".

Una seconda parte, meno importante dal punto di vista progettuale, riguarda la messa a punto di un sistema in grado di raccogliere in remoto i dati prelevandoli dalla memoria del controllore, sfruttando meccanismi standardizzati, al fine di portarli in visualizzazione in un formato grafico che permetta di avere un riscontro visivo del funzionamento. In modo complementare, e per lo stesso motivo, questo sistema di supervisione dovrà anche consentire di inviare comandi al controllore. In relazione a questa tematica si è deciso di sfruttare le caratteristiche delle soluzioni Simatic HMI descritte nella sezione 2.3.4, che integrano meccanismi trasparenti per l'accesso alla

memoria del PLC e consentono di disegnare interfacce grafiche personalizzate con tempi di sviluppo estremamente contenuti.

In questa sezione sono descritte le fasi precedentemente introdotte ed i risultati di questi processi focalizzandosi principalmente sul lavoro svolto lato controllore, che presenta molti aspetti di cui è opportuno dare una panoramica e la cui realizzazione ha occupato la maggior parte del tempo.

Imponendo l'insieme delle informazioni da esportare e la precisa struttura della macchina a stati, lo standard, di fatto, richiede la presenza di funzionalità indipendenti (considerate atomiche dal punto di vista dell'utilizzatore) per la gestione degli aspetti operativi. Queste funzionalità, da realizzare tramite blocchi in ambiente STEP7, sono state individuate tra le seguenti:

- **Interfaccia PackTags** Si concretizza in una struttura dati globale dove vengono memorizzate tutte le informazioni richieste. I riferimenti devono essere noti per poter recuperare le informazioni dall'esterno.
- **Gestore dei modi** Si occupa di elaborare le richieste di cambio modo inviate tramite i tag di comando PackTags. Garantisce che le transizioni avvengano soltanto in corrispondenza di configurazioni predefinite (e sicure).
- **Gestore degli stati** Si occupa di verificare tutte le condizioni necessarie alla corretta evoluzione temporale della macchina a stati. Garantisce che questa si trovi sempre in uno stato consistente.
- **Gestore dei tempi** Mantiene aggiornata la tabella dei tempi coerentemente con modo e stato attuali di funzionamento.
- **Gestore degli eventi** Gestisce il latching delle notifiche di eventi generati dalla macchina verso la struttura PackTags.

Le interfacce dei blocchi saranno progettate per un utilizzo quanto più possibile generico e indipendente dall'applicazione oggetto del lavoro. Ogni blocco dovrà essere concepito per elaborare i parametri passati in ingresso restituendo i risultati in uscita,

senza mai accedere direttamente alla memoria, tranne in quei casi in cui i riferimenti (simbolici, in ogni caso!) siano forniti dal chiamante; anche l'utilizzo delle variabili statiche locali dovrà essere limitato allo stretto necessario, servendosene soltanto in quei casi in cui sia indispensabile conservare dati di lavoro tra una chiamata e la successiva. Seguendo questa filosofia strutturale, i parametri di configurazione statici richiesti dagli algoritmi non potranno risiedere nelle istanze dei blocchi: per questo motivo sarà predisposta una struttura dati globale nella quale inserire le configurazioni, il cui riferimento di memoria verrà passato di volta in volta richiamando quelle funzioni che ne hanno bisogno. In questo modo si potrà svincolare il blocco funzionale dal contesto in cui dovrà essere utilizzato e contemporaneamente se ne alleggerirà l'interfaccia, dovendo accettare, per quanto riguarda la configurazione, un solo dato di tipo puntatore piuttosto che una serie di dati statici immediati di vario tipo. Fatte queste premesse, è possibile delineare la struttura dell'applicazione di esempio che si andrà a realizzare, rappresentata in figura 3.3.

Dal punto di vista realizzativo è necessario considerare preliminarmente quegli aspetti che potrebbero limitare o rendere complicata la trasposizione pratica del progetto secondo l'approccio scelto. Nella precedente fase si è tenuto conto dei vincoli genericamente imposti dai linguaggi di programmazione per PLC, senza porre limitazioni progettuali specifiche relative alla piattaforma in uso. Durante la fase di realizzazione ci si è scontrati spesso con i limiti del linguaggio a contatti Siemens in ambito di indirizzamento indiretto e concatenazione di flussi di dati tra blocchi funzionali; questi problemi sono stati aggirati scrivendo un piccolo set di blocchi funzionali di utilità in linguaggio AWL che realizzano operazioni basilari di accesso indiretto alla memoria e che vengono richiamati in modo parametrico dai blocchi di più alto livello. La scelta di utilizzare il linguaggio AWL sarebbe comunque potuta rientrare tra quelle progettuali dal momento che, in genere, permette di scrivere programmi più efficienti rispetto al linguaggio a contatti; inoltre il codice dei blocchi PackML non è destinato ad essere maneggiato da collaudatori o utenti finali, per cui sarebbe anche vanificato il vantaggio della maggiore intellegibilità del linguaggio KOP.

Di seguito vengono descritti i singoli componenti della libreria, dalle funzioni di utilità ai blocchi funzionali, motivando le scelte delle interfacce e fornendo i dia-

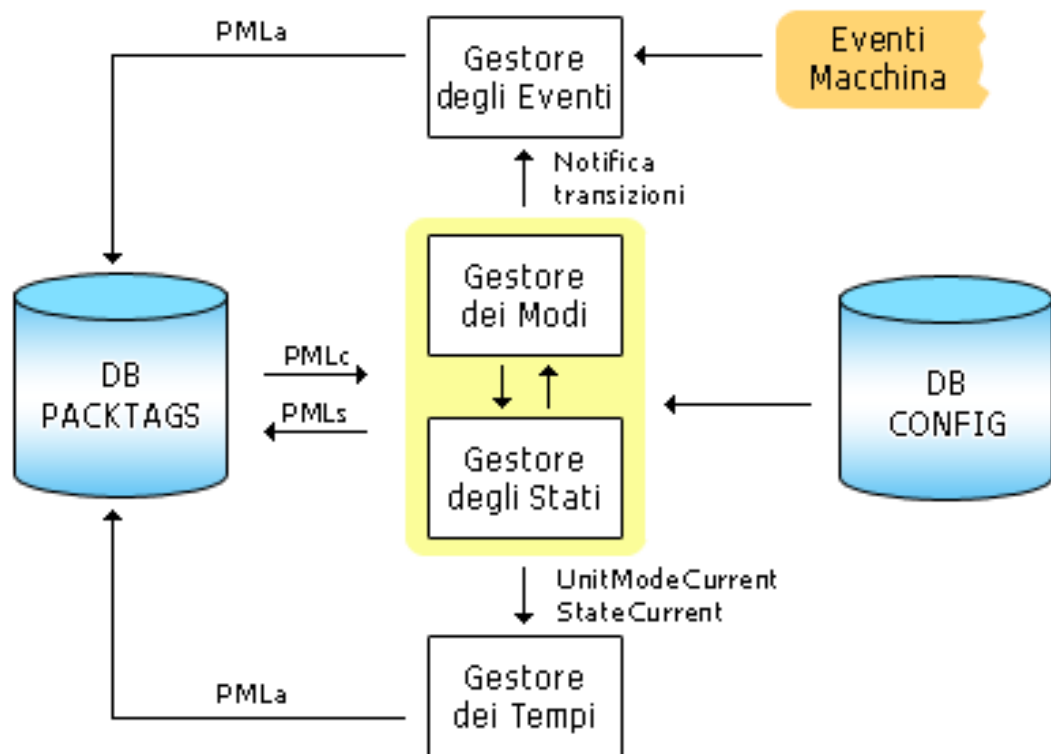


Figura 3.3: Schema concettuale della libreria realizzata.

Le logiche di gestione della macchina a stati utilizzano principalmente le informazioni presenti nei tags PMLc ed aggiornano i tags PMLs.

grammi di flusso delle logiche principali. Successivamente, si tratterà del sistema di supervisione realizzato.

### 3.3.1 Struttura dati

La struttura dati principale che contiene le informazioni PackTags è organizzata in modo gerarchico ed è basata su un insieme di User Data Type (UDT), ovvero tipi di dato personalizzato, che permettono di replicare esattamente quanto prescritto dallo standard. Per la realizzazione sono scelti i DB principalmente perché offrono la flessibilità richiesta in quanto risiedono in un'area di memoria globale, ritentiva ed accessibile in modo simbolico.

### 3.3.2 Funzioni di basso livello

Come introdotto precedentemente, sono necessarie alcune primitive di basso livello per l'accesso alla memoria al fine di superare le limitazioni del linguaggio KOP. Sono realizzate tramite blocchi FC, per lo più in AWL, e largamente utilizzate per la programmazione delle logiche che accedendo alle strutture PackTags. Di seguito le descrizioni di interfaccia e funzionamento delle principali.

#### **bit\_check, bit\_set, bit\_unset**

Questo set di funzioni permette di accedere ad un bit di memoria specificandone l'indirizzo come la somma tra una base e due offset, il primo inteso a 32 bit ed il secondo inteso come bit. Tramite questa interfaccia è possibile ad esempio controllare, impostare o resettare lo stato di un bit in uno degli elementi di una struttura (composta di elementi di dimensione double word, 32 bit) oppure puntare semplicemente ad un bit arbitrario in memoria.

Nome	Tipo	Area	Descrizione
BASE	Any	IN	Puntatore all'indirizzo iniziale
_DW	DInt	IN	Spiazzamento a 32 bit
_BIT	DInt	IN	Spiazzamento a bit
RETVAL	Bool	OUT	Stato del bit (solo per bit_check)

Tabella 3.1: Interfaccia dei blocchi bit\_check, bit\_set, bit\_unset

#### **memcpy\_s**

Questa funzione permette di copiare blocchi di memoria, specificando gli indirizzi iniziali della sorgente e della destinazione. È utilizzata quando è necessario indirizzare un preciso elemento all'interno di una struttura di cui si conosce l'indirizzo base, come nei casi in cui bisogna accedere all'elemento corrispondente al modo di funzionamento corrente all'interno di un array PackTags.

Nome	Tipo	Area	Descrizione
SRC	Any	IN	Puntatore alla struttura sorgente
SRC_INDEX	DInt	IN	Indice elemento sorgente
DST	Any	IN	Puntatore alla struttura di destinazione
DST_INDEX	DInt	IN	Indice elemento destinazione
LEN	Pointer	IN	Dimensione singolo elemento struttura

Tabella 3.2: Interfaccia del blocco memcpy\_s

**memset\_s**

Questa funzione permette di impostare tutti i bit di un blocco di memoria ad un valore scelto. È utilizzata per azzerare singoli elementi di strutture dati o l'intera struttura in blocco, come nel caso di una richiesta di cambio stato di funzionamento, che richiede il reset del tempo di stato corrente.

Nome	Tipo	Area	Descrizione
SRC	Any	IN	Puntatore alla struttura sorgente
SRC_INDEX	DInt	IN	Indice elemento sorgente
LEN	Pointer	IN	Dimensione singolo elemento struttura
VAL	Bool	IN	Valore con cui riempire l'area di memoria

Tabella 3.3: Interfaccia del blocco memset\_s

**shift\_up**

Questa funzione esegue lo shift di una struttura dati verso l'alto di N passi, sovrascrivendo le ultime N posizioni. È utilizzata per liberare la prima posizione di memoria nell'array degli eventi, per poterne inserire un nuovo in prima posizione.

Nome	Tipo	Area	Descrizione
SRC	Any	IN	Puntatore alla struttura sorgente
SRC_INDEX	DInt	IN	Indice elemento sorgente
N	DInt	IN	Numero dei passi di shift

Tabella 3.4: Interfaccia del blocco shift\_up

**memadd\_s**

Questa funzione permette di modificare il valore di un elemento di una struttura in memoria, sommando un addendo dato al valore attuale. È utilizzata dalla logica dei tempi per aggiornare i contatori della struttura PackTags per riferimento ed in maniera efficiente.

Nome	Tipo	Area	Descrizione
SRC	Any	IN	Puntatore alla struttura sorgente
SRC_INDEX	DInt	IN	Indice elemento sorgente
LEN	Pointer	IN	Dimensione singolo elemento struttura
ADD	DInt	IN	Valore da sommare all'elemento

Tabella 3.5: Interfaccia del blocco memadd\_s

**3.3.3 Funzionalità principali**

Le funzionalità principali che la libreria offre, individuate all'inizio della sezione, consentono il funzionamento della macchina a stati PackML su PLC Simatic S7-300. Sono realizzate tramite blocchi FB, per la maggior parte in linguaggio KOP, e fanno frequente utilizzo di richiami alle funzioni di utilità.

La realizzazione pratica di alcuni di questi blocchi ha subito l'influenza di scelte realizzative riscontrate nell'analisi della guida Procter&Gamble (sezione 3.2.1) e della libreria Siemens OPL (sezione 3.2.2). In particolare dalla prima si eredita l'organizzazione della struttura del gestore di stato e dalla seconda l'impostazione generale delle interfacce dei blocchi. Di seguito le descrizioni delle logiche dei singoli blocchi realizzati.

**Logica modi**

Questa logica è incaricata della gestione delle richieste di cambio modo, ed è realizzata nel blocco funzionale *PMLModeManager*, che deve essere richiamato ad ogni ciclo di scansione al fine di permettere l'aggiornamento della variabile statica contenente l'identificatore del modo corrente. In uscita viene fornito, di volta in volta, il valore attualizzato. Per mantenere la macchina in uno stato consistente le transizioni tra modi di funzionamento non sono sempre consentite e, in presenza di un comando di cambio modo, la logica contenuta in questo blocco deve verificare che la configurazione della macchina permetta di soddisfare la transizione richiesta. Il flusso di programma è rappresentato in figura 3.4.

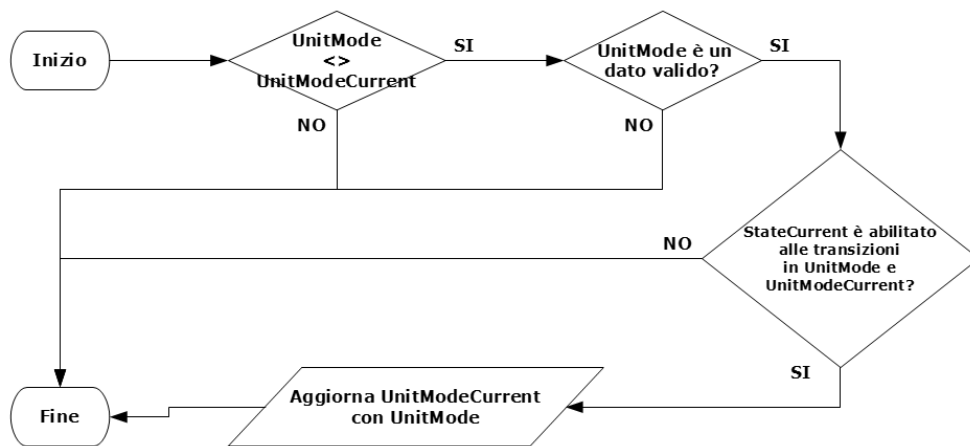


Figura 3.4: Diagramma di flusso del blocco gestore di modo

Nome	Tipo	Area	Descrizione
Cfg_ModeTransitions	Pointer	IN	Matrice di transizione di stato
StateCurrent	PMLstates	IN	Stato corrente
UnitMode	PMLmodes	IN	Modo richiesto
UnitModeChangeReq	BOOL	IN	Bit di latch richiesta
UnitModeCurrent	PMLmodes	OUT	Modo corrente
UnitModeRequested	DInt	OUT	Non implementato
UnitModeChangeInProgress	BOOL	OUT	Non implementato

Tabella 3.6: Interfaccia del blocco PMLModeManager

### Logica stati

Questa logica è incaricata della gestione di una macchina a stati finiti, in cui ogni stato è caratterizzato da un insieme di possibili transizioni attivate da diversi eventi che possono verificarsi, ed è realizzata nel blocco funzionale *PML-StateManager* che deve essere richiamato ad ogni ciclo di scansione al fine di permettere l'aggiornamento della variabile statica contenente l'identificatore di stato corrente. La logica mira a replicare il diagramma degli stati PackML per il modo automatico (figura 1.1) che mostra il più esteso ciclo di funzionamento possibile della macchina a stati. Vi sono due tipi principali di transizioni che possono occorrere: quelle da uno stato di attesa ad uno stato transiente (o duale) in seguito ad un comando, e quelle da uno stato transiente (o duale) al corrispondente stato di attesa finale in seguito alla conclusione delle operazioni. Dal momento che la logica di controllo ha una natura a stati, non è possibile mostrarne una panoramica del funzionamento tramite diagramma di flusso, come nel caso precedente; si cercherà quindi di dare una dimostrazione del comportamento tramite esempi pratici in corrispondenza di entrambi i tipi di transizione.

In figura 3.5a è rappresentato il diagramma KOP della logica dello stato di attesa Stopped. In questo caso il comando (CmdReset) che causerebbe una transizione al successivo stato di attesa (Idle) passando per il corrispondente stato transiente (Resetting), è accettato soltanto nel caso in cui lo stato finale (Idle) sia abilitato. Il controllo immediatamente successivo (rappresentato in figura 3.5b) è la logica dello stato Resetting. In questo caso il comando (CmdStateComplete) che causerebbe il completamento della transizione verso lo stato di attesa finale (Idle) è l'unico accettabile a meno che lo stato transiente (Resetting) sia disabilitato, caso in cui lo stato finale (Idle) verrebbe immediatamente raggiunto.

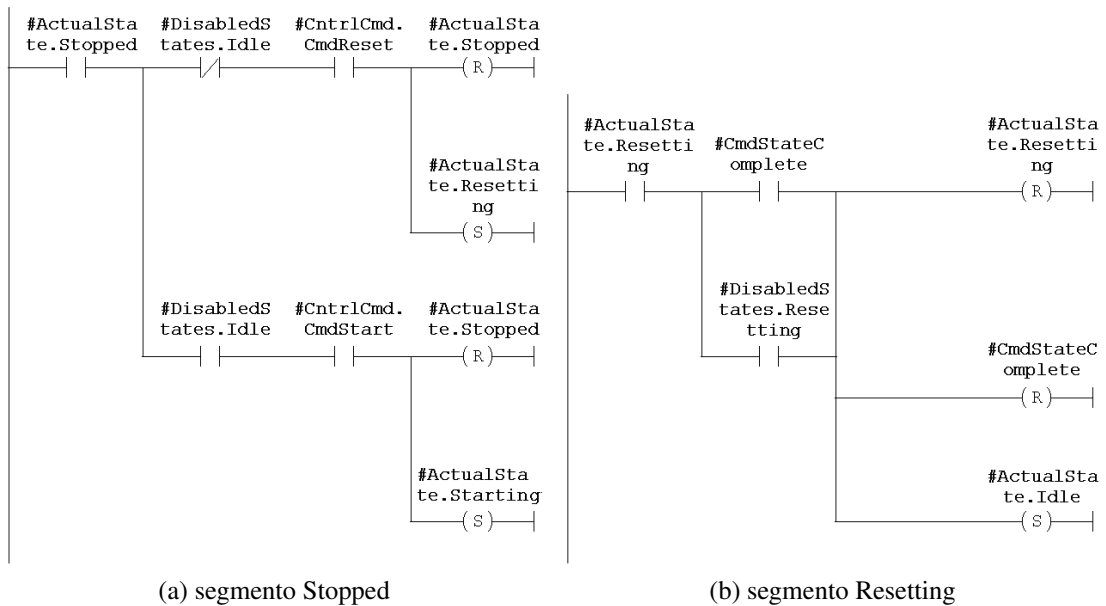


Figura 3.5: Logica interna del gestore di stato (diagramma KOP)

Nome	Tipo	Area	Descrizione
Cfg_DisableStates	Pointer	IN	Matrice degli stati disabilitati
UnitModeCurrent	PMLmodes	IN	Modo corrente
CntrlCmd	PMLstates	IN	Stato richiesto
CmdChangeReq	BOOL	IN	Bit di latch richiesta
StateCurrent	PMLstates	OUT	Stati corrente
StateRequested	DInt	OUT	Non implementato
StateChangeInProgress	BOOL	OUT	Non implementato

Tabella 3.7: Interfaccia del blocco PMLStateManager

### Logica tempi

Questa logica si occupa dell'aggiornamento della tabella dei tempi all'interno della struttura PackTags, ed è realizzata nel blocco funzionale *PMLTimesManager*. Tiene conto del tempo intercorso tra due chiamate successive ed utilizza questo valore per incrementare i contatori di modo e stato corrente; per questo

motivo dovrebbe essere richiamata ad ogni ciclo di scansione. Il flusso della logica è rappresentato in figura 3.6.

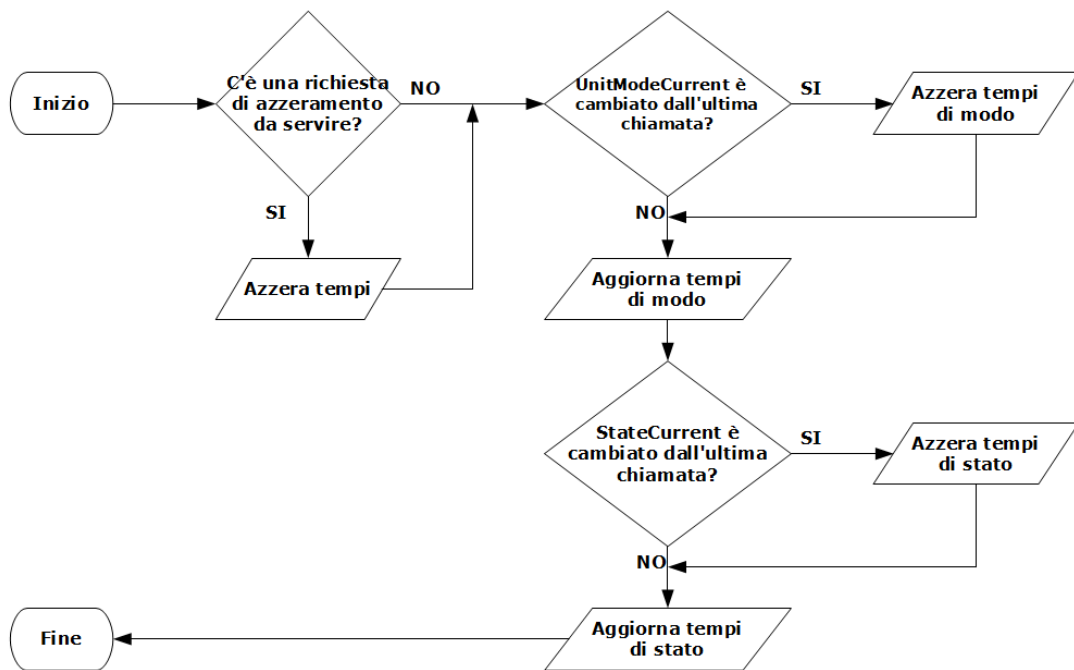


Figura 3.6: Diagramma di flusso del blocco gestore dei tempi

Nome	Tipo	Area	Descrizione
Cmd_ResetAllTimes	BOOL	IN	Reset di tutti i tempi
Cmd_ResetCurrentModeTimes	BOOL	IN	Reset dei soli tempi di modo corrente
AccTimeCurrent	DInt	IN	Tempo trascorso dall'accensione
UnitModeCurrent	DInt	IN	Modo corrente
StateCurrent	DInt	IN	Stato corrente
ModeCurrentTime	Pointer	IN	Array dei tempi di modo
StateCurrentTime	Pointer	IN	Array dei tempi cumulativi di modo
ModeCumulativeTime	Pointer	IN	Matrice dei tempi di stato
StateCumulativeTime	Pointer	IN	Matrice dei tempi cumulativi di stato

Tabella 3.8: Interfaccia del blocco PMLTimesManager

### Logica eventi

Questa logica viene richiamato quando è necessario inserire un evento nell'array dedicato all'interno della struttura PackTags, ed è realizzata nel blocco funzionale *PMLEventsManager*. Prepara la struttura di segnalazione compilando i campi ID, Value, Message e TimeEvent con i valori appropriati secondo lo standard ed includendo ora e data attuale dell'orologio interno del controllore.



Figura 3.7: Diagramma di flusso del blocco gestore degli eventi

Nome	Tipo	Area	Descrizione
AlarmArrayTag	Pointer	IN	Array degli allarmi
AlarmID	DInt	IN	ID dell'evento da accodare
AlarmValue	DInt	IN	Value dell'evento da accodare
NewAlarm	BOOL	IN	Bit di latch richiesta

Tabella 3.9: Interfaccia del blocco PMLEventsManager

### 3.3.4 Supervisore

Dal momento che il lavoro di tesi è orientato alla realizzazione di un sistema sperimentale basato sulla libreria realizzata, è stata predisposta un'interfaccia che consente di verificare il funzionamento del meccanismo di scambio dati PackTags. In questo senso è importante fornire un riscontro dello stato del sistema in base alle informazioni ottenute dai tags di stato, nonché permettere di modificarne il comportamento utilizzando i tags di comando e di amministrazione.

L'interfaccia, basata sulle soluzioni Simatic HMI, è progettata per l'utilizzo su di un pannello tattile ed è programmata utilizzando l'ambiente di sviluppo WinCC flexible. La fase realizzativa è stata guidata dalla volontà di replicare graficamente il modello degli stati PackML così come rappresentato nella documentazione ed in figura 1.1, per avere un riferimento visivo facilmente interpretabile e dal quale si potesse ricavare "a colpo d'occhio" lo stato operativo della macchina.

Utilizzando l'ambiente WinCC flexible per accedere alla memoria dei dispositivi connessi, sono stati innanzitutto configurati i *collegamenti* (figura 3.8). In questo modo, specificando l'indirizzo e la collocazione del PLC S7-300 del quale controllare le strutture dati, si possono indirizzare le aree di memoria tramite un meccanismo trasparente al programmatore. Successivamente alla creazione del collegamento, nella

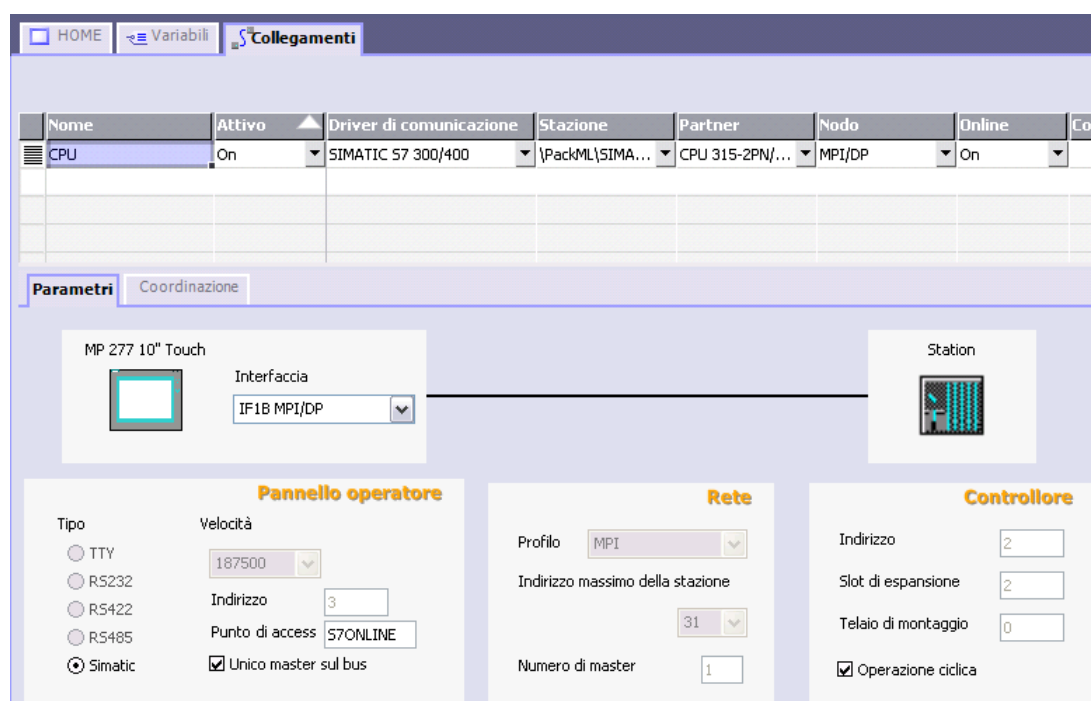


Figura 3.8: Ambiente WinCC flexible, creazione dei collegamenti

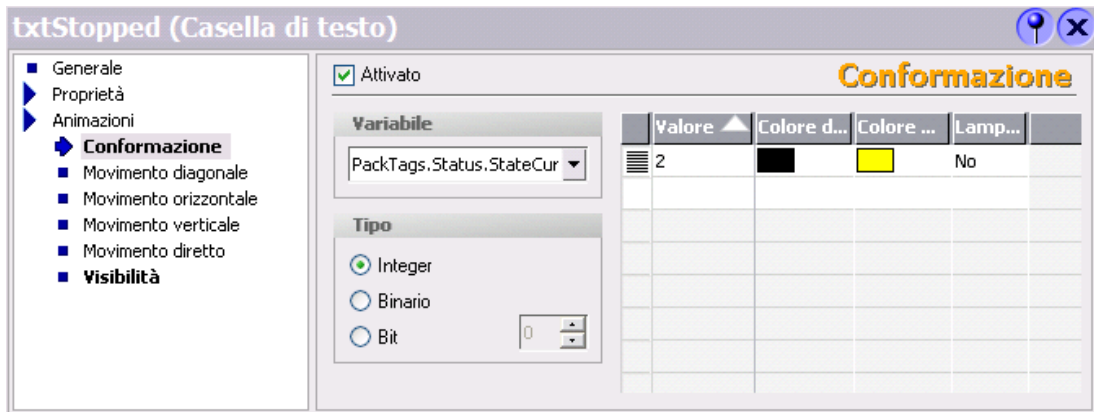
sezione *variabili* (figura 3.9), è stato definito un insieme di dati locali all'ambiente runtime, associati ognuno ad uno specifico dato della struttura PackTags. Le variabili locali sono associate direttamente ad un'area di memoria del dispositivo collegato,

e sono dette “tags”. Ciclicamente, ad intervalli programmabili, il contenuto dei tags viene aggiornato in base al valore della locazione di memoria puntata.

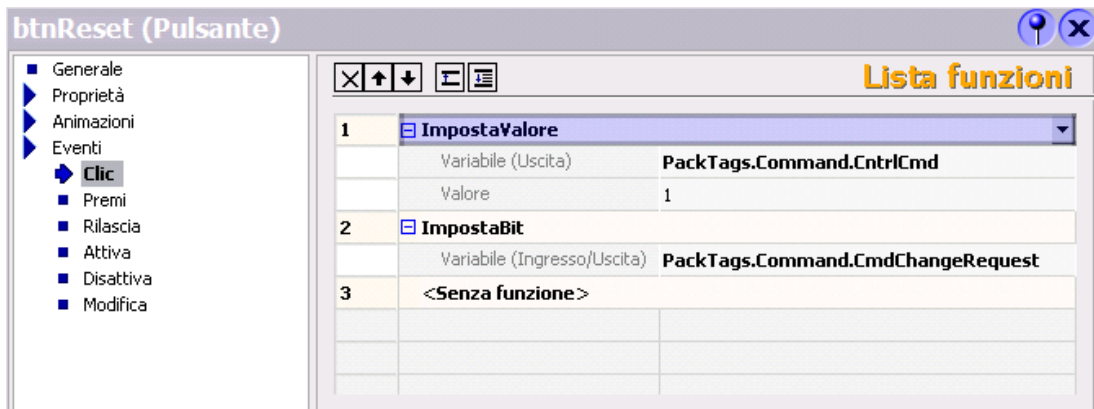
Nome	Collega...	Tipo di d...	Simbolo	Indirizzo	Elem...	Ciclo...
PackTags.Admin.Alarm[8].ID	CPU	DInt	ID	DB 1 DBD 812	1	1 s
PackTags.Admin.Alarm[8].Message	CPU	String	Message	DB 1 DBB 820	1	1 s
PackTags.Admin.Alarm[8].TimeEvent.AlmDate	CPU	Date	AlmDate	DB 1 DBW 854	1	1 s
PackTags.Admin.Alarm[8].TimeEvent.AlmTime	CPU	Time of day	AlmTime	DB 1 DBD 856	1	1 s
PackTags.Admin.Alarm[8].Value	CPU	DInt	Value	DB 1 DBD 816	1	1 s
PackTags.Admin.Alarm[9].ID	CPU	DInt	ID	DB 1 DBD 876	1	1 s
PackTags.Admin.Alarm[9].Message	CPU	String	Message	DB 1 DBB 884	1	1 s
PackTags.Admin.Alarm[9].TimeEvent.AlmDate	CPU	Date	AlmDate	DB 1 DBW 918	1	1 s
PackTags.Admin.Alarm[9].TimeEvent.AlmTime	CPU	Time of day	AlmTime	DB 1 DBD 920	1	1 s
PackTags.Admin.Alarm[9].Value	CPU	DInt	Value	DB 1 DBD 880	1	1 s
PackTags.Admin.ModeCumulativeTime[]	CPU	DInt	<Non definito>	DB 1 DBD [hmi_data.pModeCumulativeT]	32	1 s
PackTags.Admin.ModeCurrentTime[]	CPU	DInt	<Non definito>	DB 1 DBD [hmi_data.pModeCurrentT]	32	1 s
PackTags.Admin.StateCumulativeTime[]	CPU	DInt	<Non definito>	DB 1 DBD [hmi_data.pStateCumulativeT]	32	1 s
PackTags.Admin.StateCurrentTime[]	CPU	DInt	<Non definito>	DB 1 DBD [hmi_data.pStateCurrentT]	32	1 s
PackTags.Command.CmdChangeRequest	CPU	Bool	CmdChangeRequest	DB 1 DBX 10560.0	1	1 s
PackTags.Command.CntrlCmd	CPU	DInt	CntrlCmd	DB 1 DBD 10556	1	1 s
PackTags.Command.UnitMode	CPU	DInt	UnitMode	DB 1 DBD 10536	1	1 s
PackTags.Command.UnitModeChangeRequest	CPU	Bool	UnitModeChangeRequest	DB 1 DBX 10540.0	1	1 s
PackTags.Status.StateCurrent	CPU	DInt	StateCurrent	DB 1 DBD 13066	1	1 s
PackTags.Status.UnitModeCurrent	CPU	DInt	UnitModeCurrent	DB 1 DBD 13046	1	1 s

Figura 3.9: Ambiente WinCC flexible, definizione delle variabili

Una volta rese disponibili le informazioni PackTags in dati locali utilizzabili nel programma HMI, è possibile creare la parte grafica. I controlli grafici sono configurati principalmente per reagire alle variazioni dei dati al succedersi dei cicli di aggiornamento delle variabili modificando il loro aspetto (figura 3.10a). Alcuni controlli, tipicamente i pulsanti, effettuano azioni di scrittura di valori nella memoria collegata in seguito ad eventi dell’ambiente HMI, ad esempio il click (figura 3.10b).



(a) Animazioni - Modifica della conformazione



(b) Eventi - Azioni in risposta al click

Figura 3.10: Ambiente WinCC flexible, controlli grafici

# Capitolo 4

## Sistema sperimentale

Nei capitoli precedenti è stato illustrato il processo di progettazione e realizzazione di una libreria per lo sviluppo di software PLC secondo lo standard OMAC. Questo capitolo illustra la messa a punto di un sistema sperimentale, basato sulla libreria realizzata, che consenta di verificare il funzionamento della macchina a stati PackML e del meccanismo di scambio dati PackTags, nonché di valutare le prestazioni del codice secondo metriche che verranno successivamente definite. In ottica dimostrativa, inoltre, viene descritto l'utilizzo dell'interfaccia HMI appositamente sviluppata poiché in grado di fornire un riscontro dello stato del sistema in base alle informazioni ottenute tramite i tags di stato. Essa permetterà altresì di influenzarne il comportamento utilizzando i tags di comando e di amministrazione.

In questo capitolo viene presentato il sistema sperimentale per la verifica funzionale della libreria realizzata. Le diverse sezioni mostrano in dettaglio il funzionamento delle singole componenti.

### 4.1 Ambiente di test

Gli strumenti di sviluppo Siemens permettono di allestire un sistema completamente simulato in cui eseguire i programmi PLC. Durante lo sviluppo i test sono stati effettuati tramite il software S7-PLCSIM (presentato nella sezione 2.3.3), che consente di simulare il PLC in tutte le sue funzionalità: utilizzandolo in combinazione con

gli strumenti di debug di STEP7 è possibile eseguire codice e controllare il contenuto della memoria del dispositivo durante i cicli di scansione. Senza prescindere da quanto detto, è doveroso far notare come l'ambiente di simulazione sia in grado di fornire tutti i riscontri necessari dal punto di vista algoritmico e del funzionamento, anche se in questo caso la misura delle prestazioni di un sistema fisico rappresenta una caratterizzazione importante che il simulatore non può fornire.

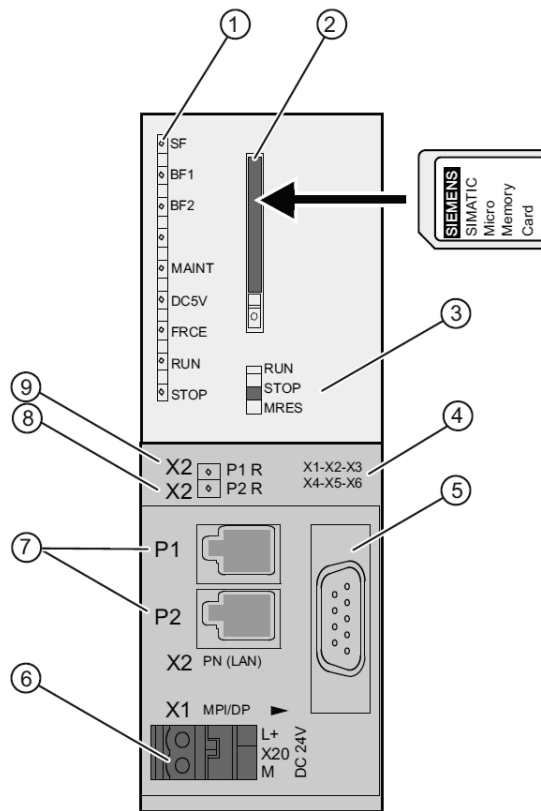
Visto l'ambito applicativo delle tecnologie studiate e della libreria realizzata, si rende necessaria la verifica del funzionamento del sistema su hardware reale. La configurazione utilizzata per le prove deve consentire di ottenere condizioni operative quanto più possibile simili al contesto di applicazione; pertanto la scelta dei componenti deve essere in linea con quelli tipicamente presenti nelle stazioni Simatic S7 che accompagnano i quadri elettrici dei macchinari industriali assemblati in azienda.

### **Configurazione PLC**

La stazione S7 di test, seppur minimale, è basata sulla CPU 315-2 PN/DP (figura 4.1). La CPU integra in un solo elemento la memoria di lavoro, la capacità di calcolo e le interfacce di comunicazione, quindi non sono necessari moduli aggiuntivi in quanto il programma dimostrativo si occupa solamente di inizializzare e far evolvere una macchina a stati PackML in base a comandi esterni o eventi temporizzati, senza operare su alcuna porta di I/O. L'esemplare specifico utilizzato è identificato dal numero di ordinazione *6ES7 315-2EH14-0A0B* e utilizza la versione *V3.1.1* del firmware di sistema operativo. La memoria di lavoro integrata è di 384KByte, mentre quella di caricamento è contenuta in una MMC da 512KByte.

### **Configurazione HMI**

Per quanto riguarda il dispositivo per l'interfaccia HMI, il modello scelto è il Simatic MP277 10" Touch (figura 4.2) con display da 10.4" a 64K colori e risoluzione VGA, che offre una buona area di visualizzazione e la connettività necessaria all'interfacciamento con il PLC utilizzato. Questo dispositivo è sostanzialmente un pc embedded che esegue Microsoft Windows CE 5.0 ed il framework runtime di WinCC flexible. In



**Cifra Descrizione**

- ① LED di stato e di errore
- ② Vano per la SIMATIC Micro Memory Card con pulsante di espulsione
- ③ Selettore dei modi operativi
- ④ Indirizzo MAC
- ⑤ 1. Interfaccia X1 (MPI/DP)
- ⑥ Connettore per l'alimentazione
- ⑦ 2. Interfaccia X2 (PN), con switch a 2 porte
- ⑧ Porta PROFINET 2  
Lo stato della porta 2 viene segnalato da un LED a due colori (verde/giallo):
  - LED verde: è disponibile un LINK a un partner
  - Il LED passa al giallo: traffico di dati attivo (RX/TX)
 R: porta per la configurazione di una topologia ad anello con ridondanza del mezzo
- ⑨ Porta PROFINET 1  
Lo stato della porta 1 viene segnalato da un LED a due colori (verde/giallo):
  - LED verde: è disponibile un LINK a un partner
  - Il LED passa al giallo: traffico di dati attivo (RX/TX)
 R: porta per la configurazione di una topologia ad anello con ridondanza del mezzo

Figura 4.1: CPU 315-2 PN/DP

questo sistema sperimentale il pannello MP277 non è fisicamente utilizzato e l'interfaccia viene eseguita su pc come già fatto durante lo sviluppo, in quanto il dispositivo non è stato messo a disposizione e comunque le sue prestazioni reali non sono oggetto di interesse per questa tesi.

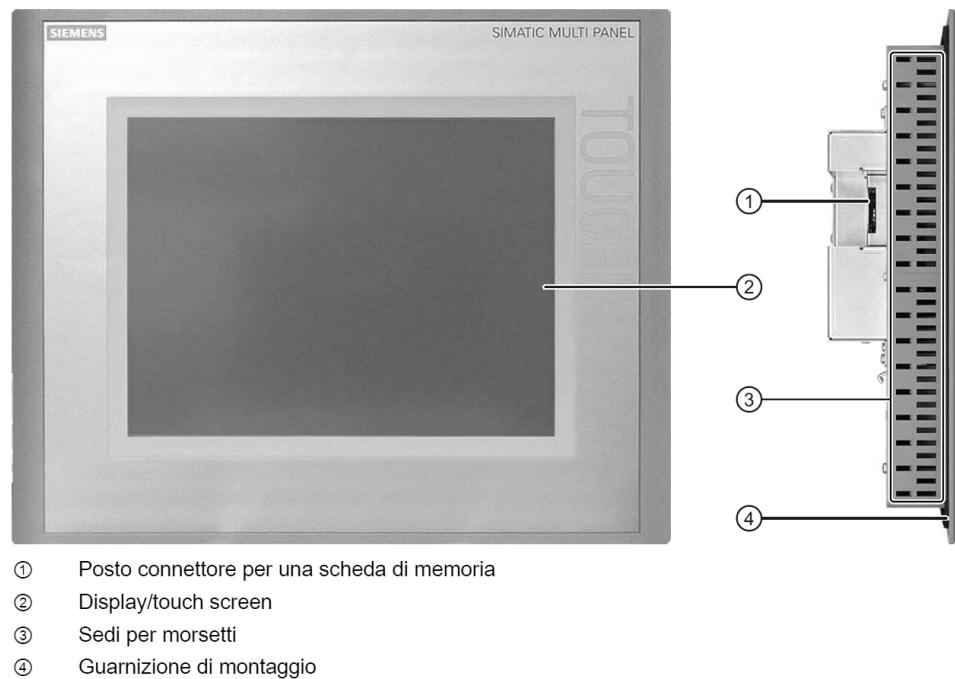


Figura 4.2: Pannello HMI Simatic MP277 10'' Touch

## 4.2 Risultati sperimentali

Per la lettura e l'interpretazione dei dati sperimentali è necessario definire delle metriche che consentano di misurare le prestazioni del sistema al fine di poter ottenere dati facilmente interpretabili che serviranno anche per la validazione del sistema rispetto ai requisiti. Queste metriche di valutazione sono essenzialmente legate all'efficienza nell'uso delle risorse del PLC, in particolare il tempo di ciclo e la memoria.

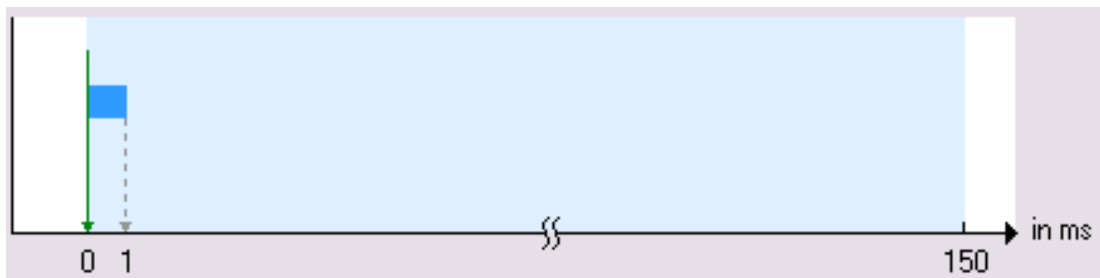
## Tempi di ciclo

Di rilevante importanza per la valutazione delle prestazioni è la misurazione dei tempi di esecuzione del codice, il cosiddetto processo di profiling. La conoscenza dei tempi di esecuzione è funzionale alla definizione dei limiti di garanzia che sono tanto necessari quanto ridotti nell'ambito industriale, in cui la maggior parte dei controlli di processo deve eseguire in tempo reale. Siccome l'ambiente di sviluppo Siemens non dispone di un profiler, si è proceduto alla valutazione del dato relativo al tempo di ciclo autonomamente calcolato dal PLC con risoluzione di un millisecondo. La risoluzione di questa misura non è così elevata da consentire un'analisi precisa del tempo impiegato per l'elaborazione di ogni singolo blocco nei diversi possibili scenari di esecuzione, ma permette comunque di ottenere un limite di tempo superiore utile per la verifica del rispetto dei requisiti. In figura 4.3 sono riportati i grafici delle misure effettuate: per ogni misura si riportano i tempi minimo e massimo rilevati in un periodo di funzionamento di 60 secondi. Il marker intermedio di colore rosso rappresenta l'ultimo dato rilevato ed è inutile ai fini della valutazione.

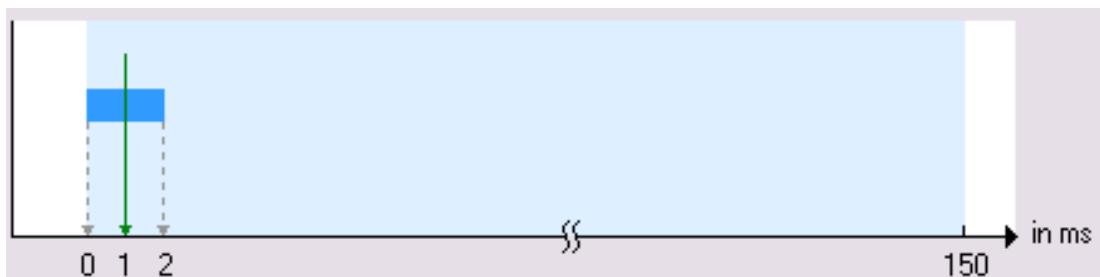
Vista la scarsa risoluzione a disposizione si è fatta una prima misurazione “a vuoto” eseguendo un programma in cui il blocco OB1 non contiene nessuna istruzione o richiamo. In figura 4.3a si osserva come il tempo impiegato per elaborare un programma senza istruzioni oscilla tra  $0\text{ ms}$  ed  $1\text{ ms}$  quando nella realtà è di pochi  $\mu\text{s}$ , secondo i dati dichiarati dal produttore.

Proseguendo nell'analisi si passa ad analizzare il tempo richiesto per l'esecuzione dei soli blocchi dei gestori di modo e stato (figura 4.3b) che si dimostrano sostenibili dalla CPU dal momento che vengono elaborati in un tempo variabile tra  $1\text{ ms}$  e  $2\text{ ms}$ .

La misura del tempo di esecuzione dell'applicazione completa (figura 4.3c), che comprende l'esecuzione di tutti i blocchi della libreria con la gestione dei comandi remoti, è leggermente più lenta rispetto alla semplice gestione della macchina a stati. L'allungamento del tempo di ciclo è dovuto ad operazioni sequenziali effettuate dai blocchi di gestione dei tempi e degli eventi. Il primo deve, ad ogni cambio modo, azzerare parte della tabella dei tempi, mentre il secondo esegue lo scorrimento verso l'alto di un elemento della struttura degli eventi nel momento in cui ne viene notificato



(a) esecuzione "a vuoto"



(b) esecuzione dei soli gestori di modo e stato



(c) esecuzione completa con gestione dei tempi e accodamento eventi



(d) evento di reset totale dei tempi

Figura 4.3: Tempi di ciclo del sistema di test.

Per ogni figura sono rappresentati i tempi minimo e massimo rilevati in 60 secondi di funzionamento, in corrispondenza di diverse configurazioni di esecuzione.

un qualsiasi nuovo. Entrambe le operazioni implicano un discreto quantitativo di operazioni sequenziali di accesso alla memoria. Il tempo misurato, si mantiene comunque entro i  $3\text{ ms}$  ed è ancora accettabile.

L'ultimo test è analogo al precedente con in aggiunta l'elaborazione del comando di azzeramento dei tempi (figura 4.3d) che, anche se non previsto dallo standard, può comunque essere utilizzato dalle routine di inizializzazione. In questo caso si tratta di azzerare completamente i due array (32 elementi ognuno) e le due matrici (1024 elementi ognuna) dei tempi presenti in PackTags, per un totale di 2112 valori di tipo DInt, ovvero 8.25 KByte di memoria. La CPU lavora con accumulatori a 32 bit e per questo le routines che accedono alla memoria sono tutte programmate in modo da scrivere e leggere a blocchi di questa dimensione. Al lato pratico l'operazione di azzeramento si traduce nella scrittura di un singolo elemento per volta nelle strutture dei tempi, rendendo necessarie 2112 operazioni di accesso a DB tramite indirizzamento indiretto per completare l'operazione. Il tempo di accesso dichiarato dal produttore per questo tipo di operazioni è di  $0.1\ \mu\text{s}$  [21] a cui si aggiunge l'overhead della preparazione e dell'esecuzione del ciclo di accesso alla memoria, che porta al tempo massimo di  $7\text{ ms}$ .

## Memoria utilizzata

Un secondo parametro di valutazione riguarda l'utilizzo della memoria di lavoro, in quanto il PLC ne possiede quantitativi relativamente ridotti rispetto a sistemi general purpose, in linea con le prestazioni che offre e con la sua destinazione d'uso. È importante che la libreria utilizzi un ridotto quantitativo di memoria di lavoro, in quanto in un'ottica applicativa la stessa CPU dovrà elaborare anche le logiche di macchina e per questo motivo non è possibile che le risorse siano esaurite dalla sola libreria. Per quanto riguarda la memoria di caricamento la struttura dati imposta dallo standard offre pochi gradi di libertà, ma in questo caso i limiti sono meno stringenti poiché la CPU può accettare MMC fino ad una dimensione massima di 8 MByte. In figura 4.4 è rappresentata graficamente l'occupazione di memoria della CPU dopo il caricamento dei blocchi della libreria; si può notare come l'uso di questa risorsa sia relativamente

contenuto rispetto alla dotazione della CPU a disposizione, lasciando sufficiente spazio per le logiche di controllo macchina.

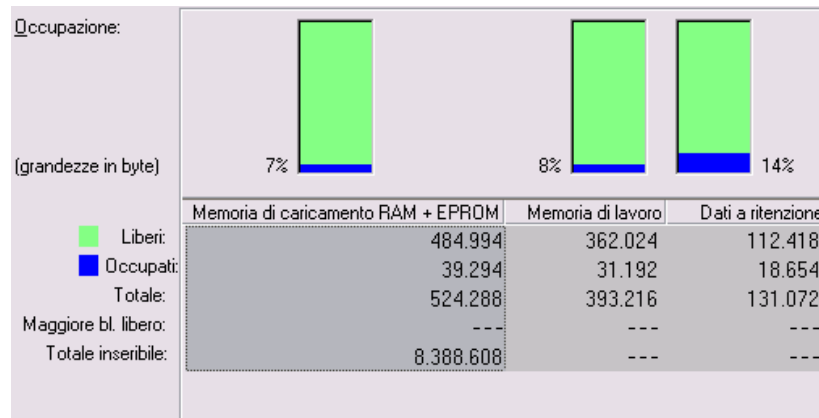


Figura 4.4: Utilizzo di memoria del sistema di test

### 4.3 Funzionamento

Ponendo il selettore di funzionamento del PLC nello stato di RUN è possibile trasferire ed eseguire il programma. Una volta in esecuzione, il sistema è utilizzabile tramite l'interfaccia HMI, che permette di visualizzare i dati forniti dal PLC ed inviare comandi. Nelle figure 4.5, 4.6 e 4.7 sono mostrate e brevemente descritte le pagine dell'interfaccia HMI.

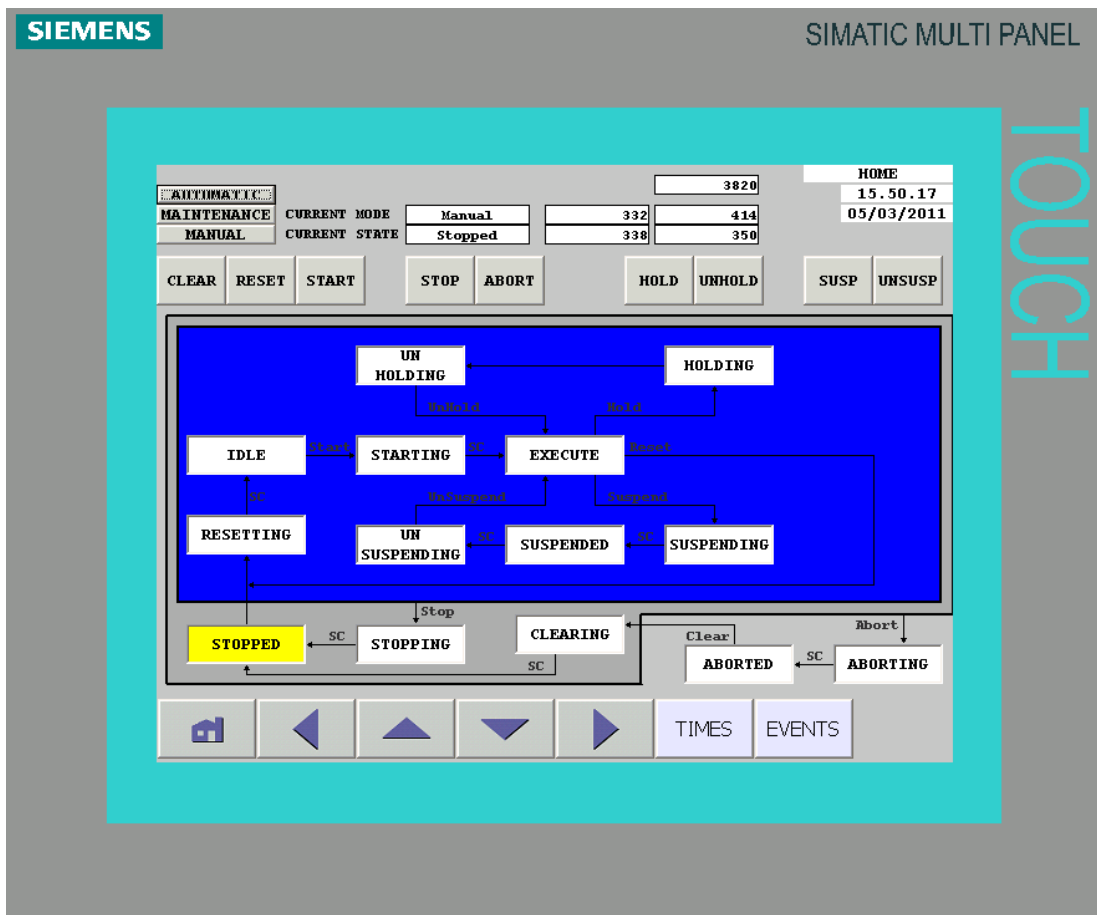


Figura 4.5: Interfaccia HMI, pagina principale.

Evidenzia in un diagramma lo stato attuale della macchina e consente di inviare i comandi *Clear*, *Reset*, *Start*, *Stop*, *Abort*, *Hold*, *UnHold*, *Suspend* e *Unsuspend* o passare alle pagine dei tempi e degli eventi.

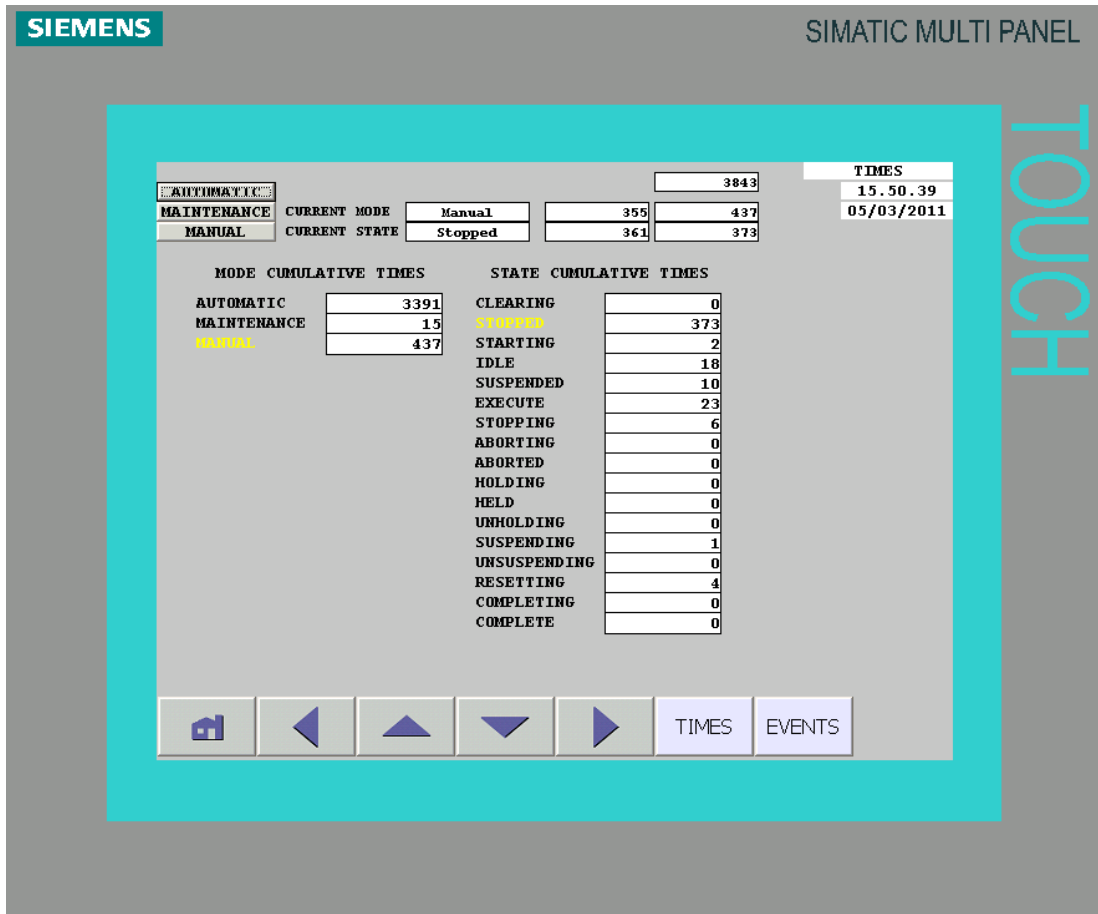


Figura 4.6: Interfaccia HMI, pagina dei tempi.

Mostra i valori della tabella dei tempi relativi al modo corrente, evidenziando lo stato ed il modo attualmente in aggiornamento.

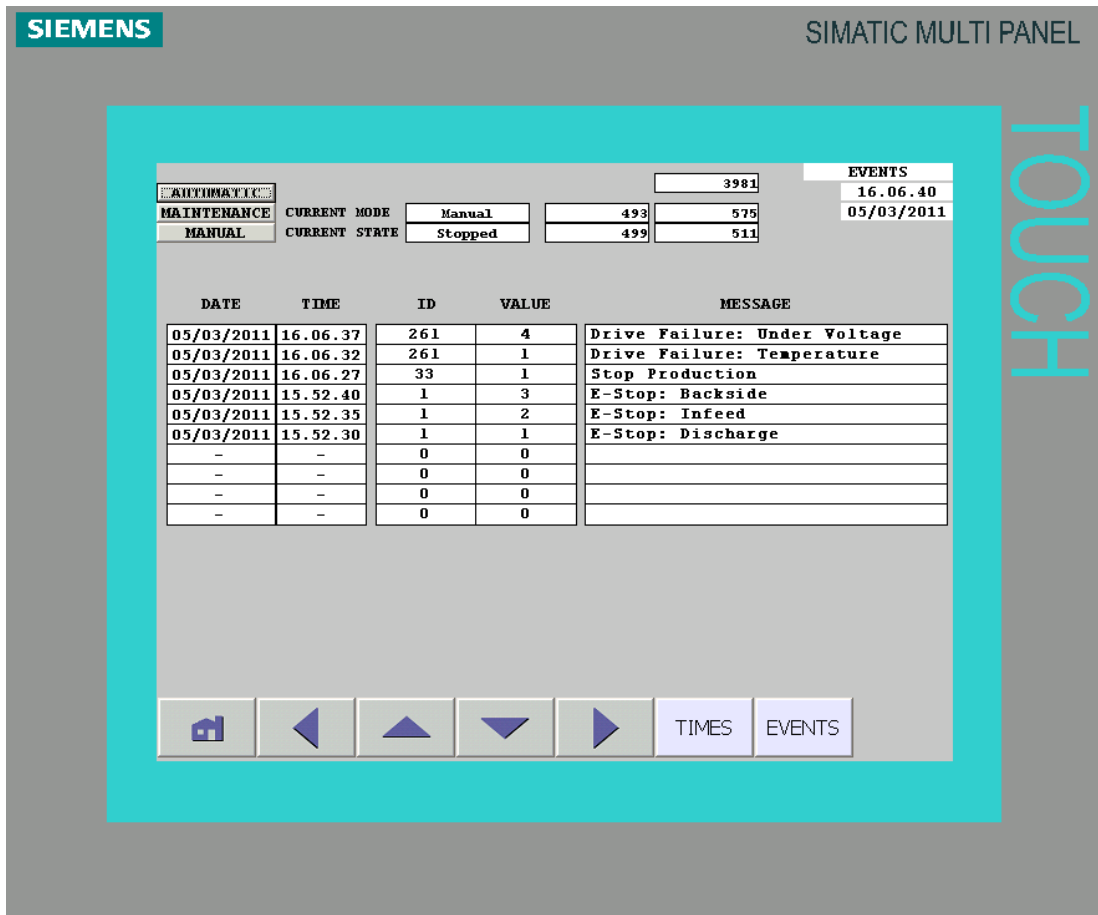


Figura 4.7: Interfaccia HMI, pagina degli eventi.

Mostra l'array degli eventi macchina notificati dal blocco di gestione degli eventi lato PLC.

## Capitolo 5

### Conclusioni

Le argomentazioni che hanno ispirato il lavoro descritto in questa tesi sono il risultato e la testimonianza della rivoluzione attualmente in corso nel settore dell'automazione industriale per quanto riguarda l'approccio al controllo ed alla coordinazione delle linee di produzione. Anche se l'origine dei processi che hanno condotto allo stato attuale è da collocare alcuni anni indietro nel tempo, è soltanto in questo periodo storico che una vista d'insieme tra le realtà più innovative mostra un nuovo modello di organizzazione degli impianti, che sta sensibilizzando anche le realtà meno attente a queste tematiche.

Dal punto di vista del contesto accademico in cui questo lavoro si è svolto, non si può fare a meno di notare come il processo di evoluzione degli impianti, da meccanici ed elettronici prima, passando per gli attuali sistemi di controllo basati su software, per arrivare ai sistemi mecatronici in cui l'integrazione e l'interdipendenza tra le varie discipline svolgono un ruolo chiave, abbia molti punti di contatto con le tecnologie del settore ICT in senso più ampio, anche se non ne condivide le modalità. A questo riguardo il settore informatico, per sua natura, è generalmente orientato a sostenere e sospendere il cambiamento in modo diretto, piuttosto che seguirlo o subirlo, anche a causa dell'interesse collettivo e della scarsa presenza di vincoli realizzativi e di impiego. L'ambito industriale, per contro, non può permettersi di adottare soluzioni che non trovino appoggio tra i clienti-utilizzatori e che non godano della fiducia o non soddisfino le aspettative di questi ultimi.

Allo stesso modo in cui i sistemi informatici richiedono la presenza di tecnologie e meccanismi di mediazione per la coordinazione delle nuove realtà complesse, eterogenee e distribuite, questa tesi ha dato un contributo alla nascente necessità di collaborazione tra sistemi di controllo industriale nei più innovativi impianti produttivi. Il lavoro descritto riguarda la realizzazione di una libreria per l'organizzazione del software di controllo industriale secondo le specifiche OMAC, che si propongono come modello per favorire l'interoperabilità tra le diverse entità che possono rendere disponibili informazioni di processo che spesso sono veicolate secondo protocolli proprietari.

Il sistema realizzato mostra come sia possibile mantenere aggiornata una struttura dati complessa su un controllore industriale non espressamente progettato per lavorare con frequenti accessi a dati strutturati in memoria di massa, senza apprezzabili aggravii prestazionali. In modo complementare è stato mostrato un possibile esempio di raccolta ed utilizzo di questi dati tramite una interfaccia uomo macchina. Il ventaglio delle applicazioni realizzabili a partire dalla libreria progettata è ampio, ma principalmente orientato alla messa a punto di sistemi di supervisione, anch'essi conformi allo standard, che possano raccogliere ed elaborare dati da fonti eterogenee per attuare politiche di coordinazione centralizzate.

Il naturale sviluppo di questa tesi dovrebbe riguardare lo studio e la progettazione di un sistema SCADA che possa essere in grado di aggregare le informazioni provenienti dai macchinari che compongono una intera linea di produzione e coordinare le loro operazioni. Dal punto di vista realizzativo, invece, è possibile approfondire gli aspetti legati al miglioramento delle prestazioni, per fare in modo che l'utilizzo della libreria raggiunga un grado di trasparenza nell'utilizzo delle risorse dei dispositivi ancora maggiore di quello attuale. A questo scopo è possibile ricercare margini di miglioramento sfruttando i blocchi funzionali forniti dal produttore che promettono maggiore efficienza dal punto di vista dei tempi di esecuzione delle operazioni di copia con accesso indiretto alla memoria.

# Bibliografia

- [1] PackML subcommittee. *Packaging Machine Language V3.0 Mode & States Definition Document*. OMAC Packaging Working Group, June 2006.
- [2] Chair of Food Packaging Technology. Weihenstephan standards for production data acquisition in bottling and packaging plants, general part. Technical report, Technische Universität München, 2010. Versione 2005.03.
- [3] <http://www.integrapack.it>.
- [4] ISA88 committee. Machine and unit states: An implementation example of ISA-88. Technical Report ISA-TR88.00.02-2008, International Society of Automation, August 2008.
- [5] PackML subcommittee. *Tag Naming Guidelines (PackML V3.0)*. OMAC Packaging Working Group, July 2006.
- [6] Chair of Food Packaging Technology. Weihenstephan standards for production data acquisition in bottling and packaging plants, part 1: Physical interface specifications. Technical report, Technische Universität München, 2010. Version 2005.05.
- [7] Chair of Food Packaging Technology. Weihenstephan standards for production data acquisition in bottling and packaging plants, part 2: Content specification of the interface. Technical report, Technische Universität München, 2010. Version 2005.05.
- [8] Siemens AG. *Simatic S7-300, CPU 31xC e CPU 31x, Dati tecnici*, June 2010.

- 
- [9] Subcommittee 65C. Industrial communication networks - fieldbus specifications. Technical Report IEC 61158, International Electrotechnical Commission.
- [10] Subcommittee 65C. Industrial communication networks - profiles. Technical Report IEC 61784, International Electrotechnical Commission.
- [11] Siemens AG. *Simatic, Programmazione con STEP 7, Manuale*, May 2010.
- [12] Subcommittee 65B. Programmable controllers – part 3: Programming languages. Technical Report IEC 61131-3, International Electrotechnical Commission, 2003.
- [13] Siemens AG. *Simatic Engineering Tools, S7-PLCSIM V5.4 incl. SP3, Manuale d'uso*, March 2009.
- [14] Siemens AG. *Simatic HMI, Pannello operatore MP 277 (WinCC flexible), Istruzioni operative*, March 2009.
- [15] Siemens AG. *Simatic HMI, WinCC flexible 2008 Compact / Standard / Advanced, Manuale utente*, July 2008.
- [16] Siemens AG. *Simatic HMI, WinCC flexible 2008 Runtime, Manuale utente*, July 2008.
- [17] Procter & Gamble. *PackML Implementation Guide*, 2009.
- [18] S88 committee. Batch control part 1: Models and terminology. Technical Report ISA-88.01-1995, International Society of Automation, 1995.
- [19] Siemens AG. *Optimized Packaging Line, OPL Overview, Manuale di sistema*, March 2010.
- [20] Siemens AG. *LPMLV30 for Simatic S7, Manuale utente*, May 2010.
- [21] Siemens AG. *Lista operazioni S7-300*, May 2010.