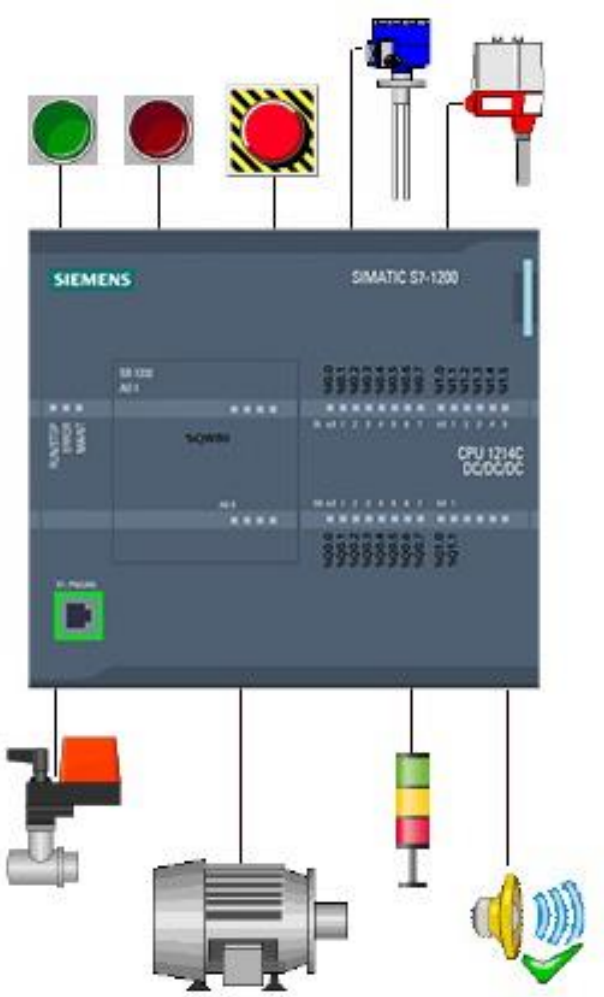
SISTEMI ELETTRONICI PER L'AUTOMAZIONE E L'INDUSTRIA (SEAI)

A.A. 2023-2024

IL PLC

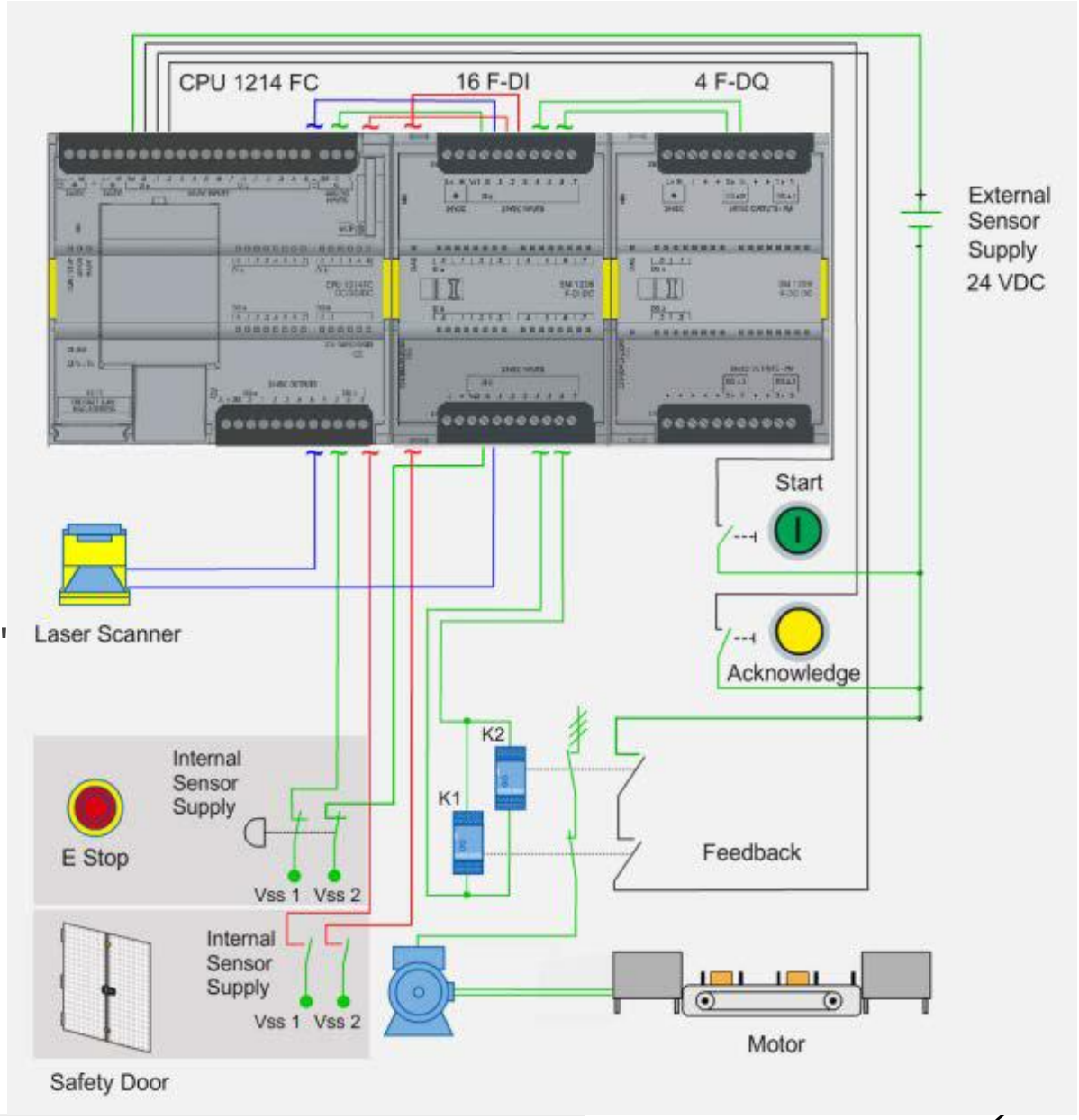
PLC: Collegamento ingressi e uscite a morsettiera

Connessione "logica"



Connessione Funzionale

- PLC alimentato
- sensori alimentati
- attuatori alimentati
- connessioni "sicure"



Immagini da Siemens

PLC: Hardware

PLC ad architettura compatta (es. Siemens S7-1200)
Posso aggiungere moduli opzione

PLC ad architettura modulare (es. Siemens S7-1500)
CPU e moduli per segnali di ingresso o uscita, digitali o analogici
Architettura molto flessibile che salva gli investimenti

- **Aggiungo moduli all'esistente**
- **Cambio CPU, mantengo moduli**
- **Cambio moduli, mantengo il resto**

Approfondimenti sui PLC
https://www.siemens.com/global/en/home/products/automation/systems/industrial/plc.html

Periferia (es. Siemens et200sp)
E' una struttura modulare "slave"
Può fare elaborazione asservita ad un PLC (o PC)
https://www.youtube.com/watch?v=FhSJhlQn3Cs

PLC: CPU e moduli

L'hardware di un PLC si basa almeno su tre elementi

- **Alimentatore (spesso integrato nella CPU)**
- **CPU**
- **Moduli di I/O (Input/Output)**

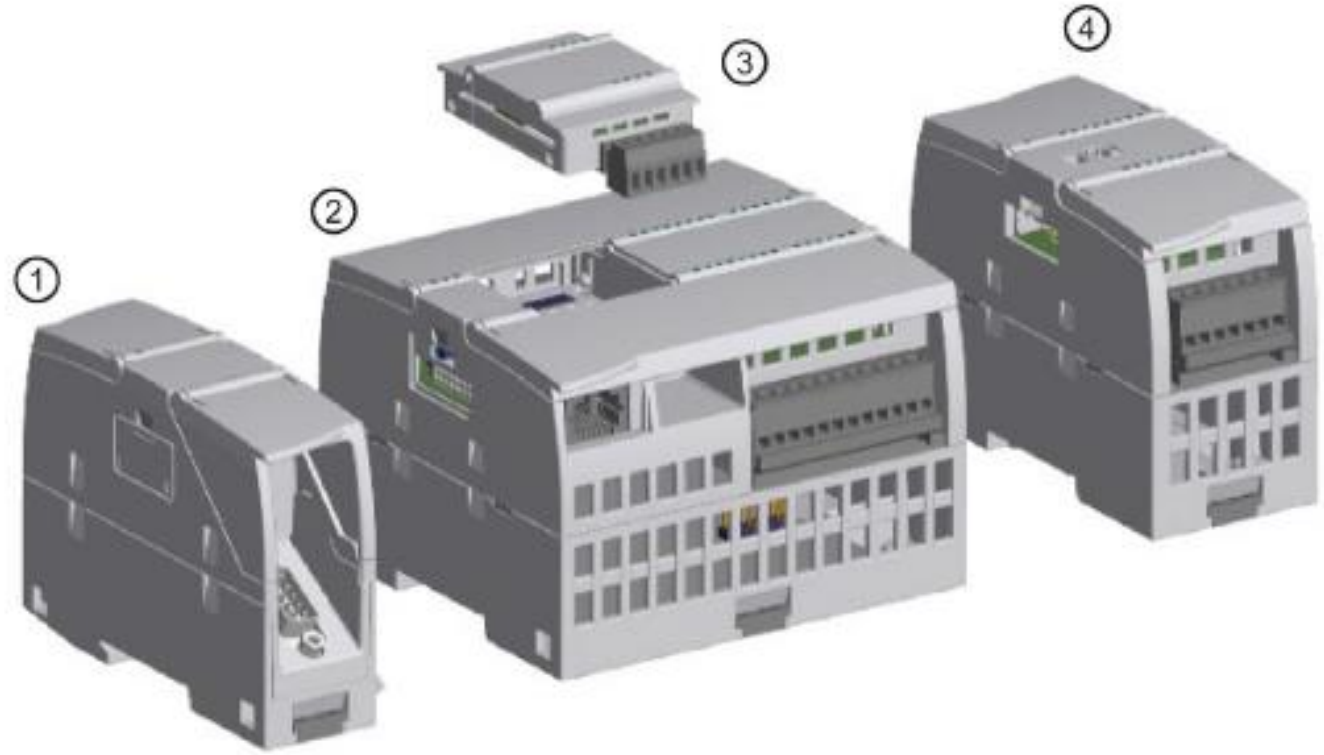
Per un PLC compatto i tre elementi possono essere integrati

In un PLC compatto posso avere ulteriori moduli di I/O

L'alimentatore serve per far funzionare il PLC e per alimentare i segnali di ingresso e uscita

PLC: CPU e moduli

Immagini da s71200_system_manual_it-IT_it-IT.pdf



① Modulo di comunicazione (CM) o processore di comunicazione (CP) (Pagina 1596)

② CPU (CPU 1211C (Pagina 1425), CPU 1212C (Pagina 1439), CPU 1214C (Pagina 1453), CPU 1215C (Pagina 1469), CPU 1217C (Pagina 1486))

③ Signal board (SB) (SB digitale (Pagina 1565), SB analogica (Pagina 1577)), scheda di comunicazione (CB) (Pagina 1606) o scheda di batteria (BB) CPU (CPU 1211C, CPU 1212C, CPU 1214C, CPU 1215C, CPU 1217C) (Pagina 1594)

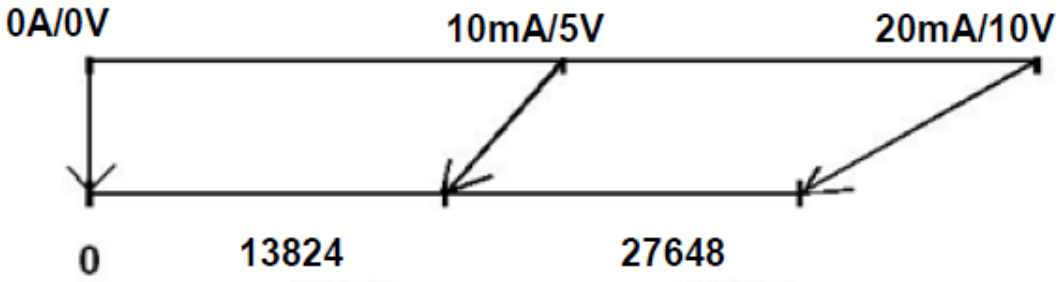
④ Modulo di I/O (SM) (SM digitale (Pagina 1503), SM analogico (Pagina 1522), SM per termocoppie (Pagina 1538), SM RTD (Pagina 1544), SM tecnologico) (Pagina 1551)

PLC: moduli di I/O (Signal Module)

I Signal Module possono essere:

- Segnali digitali (ingressi o uscite Bool)
- Segnali analogici (ingressi o uscite)
- Segnali per sensori speciali (es. termocoppie)
- Funzioni "tecnologiche"

Immagine da documentazione Siemens



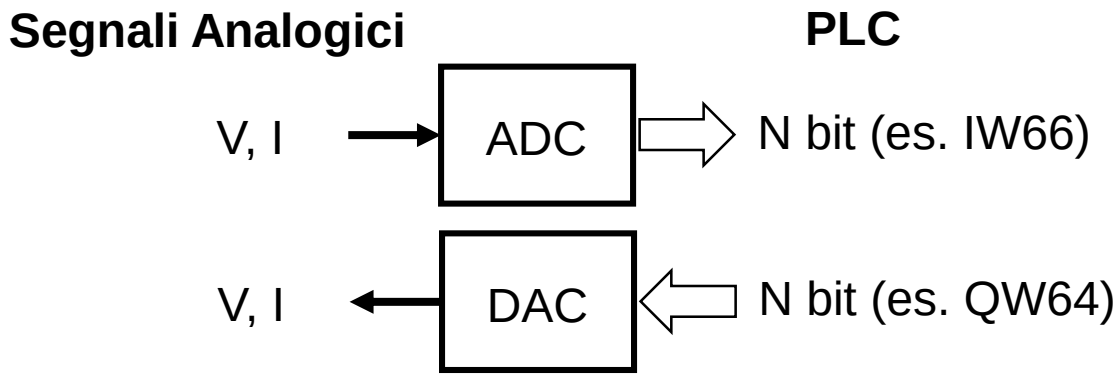
Segnali analogici

- Ingresso tra 0 e 10 V -> lettura tra 0 e 27648
- Uscita tra 0 e 20 mA -> scrittura tra 0 e 27648
- Campo di overshoot (/undershoot) -> lettura tra 27649 e 32511
- Campo di overflow (/underflow) -> lettura tra 32511 e 32768 ($=2^{15-1}$)

Funzioni tecnologiche

- Master verso sensori numerici (es. IO-Link Master)
- Funzione di pesatura, motion,… (interfaccia sensori, attuatori e firmware)

PLC: segnali analogici



I Segnali analogici possono essere:

- **In tensione V (0-10V)**
- **In corrente I (0-20mA oppure 4-20mA)**

Nel PLC c'è un convertitore Analogico/Digitale (ADC) che ci permette di leggere una word (word, 16 bit = 15 bit + segno, da – 32768 a +32767) che è in relazione di proporzionalità diretta con il segnale in ingresso Vin

- **Ingresso Vin tra 0 e 10 V -> lettura tra 0 e 27648 (es. Potenziometro IW66 nel PLC S7-1215)**
- **IW66/27648 = Vin/10V quindi IW66 = 27648·Vin/10V e Vin = 10V·IW66/27648**

Nel PLC c'è un convertitore Digitale/Analogico (DAC) che ci permette di scrivere una word (word, 16 bit = 15 bit + segno, da – 32768 a +32767) che è in relazione di proporzionalità diretta con il segnale in uscita Vout

- **Uscita Iout tra 0 e 20mA -> scrittura tra 0 e 27648 (es. Led rosso QW64 nel PLC S7-1215)**
- **QW64/27648 = Iout/20mA e quindi QW64 = 27648·Iout/20mA e Iout = 20mA·QW64/27648**

PLC: CPU

Ci sono vari modelli di CPU (in laboratorio abbiamo 1215C)

Caratteristica		CPU 1211C	CPU 1212C	CPU 1214C	CPU 1215C	CPU 1217C
Dimensioni di ingombro (mm)		90 x 100 x 75		110 x 100 x 75	130 x 100 x 75	150 x 100 x 75
Memoria utente	Lavoro	50 Kbyte	75 Kbyte	100 Kbyte	125 Kbyte	150 Kbyte
	Carico	1 Mbyte	2 Mbyte	4 Mbyte		
	Ritenzione	10 Kbyte				
I/O on-board locali	Digitale	6 ingressi/ 4 uscite	8 ingressi/ 6 uscite	14 ingressi/ 10 uscite		
	Analogico	2 ingressi			2 ingressi/2 uscite	
Dimensione dell'immagine di processo	Ingressi (I)	1024 byte				
	Uscite (Q)	1024 byte				
Memoria di merker (M)		4096 byte		8192 byte		
Ampliamento con modulo di I/O (SM)		Nessuno	2	8		
Signal Board (SB), scheda di batteria (BB) o scheda di comunicazione (CB)		1				
Modulo di comunicazione (CM) (ampliamento sul lato sinistro)		3				

PLC: CPU

Cosa sono i contatori veloci? Non bastano i contatori software?

I contatori software leggono un evento al ciclo (Tciclo~ms -> Fmax<kHz)

I contatori veloci sono contatori hardware che non lavorano sulle immagini di processo ma sui segnali reali (es. encoder = sensore che si monta sui motori e che fornisce N impulsi ad ogni giro -N~1000-)

1ms = 12500 operazioni booleane = oltre 400 operazioni su reali

Caratteristica		CPU 1211C	CPU 1212C	CPU 1214C	CPU 1215C	CPU 1217C
Contatori veloci	Totale	Fino a 6 configurati per l'uso di qualsiasi ingresso integrato o SB				
	1 MHz	-				Ib.2 ... Ib.5
	100/180 kHz	Ia.0 ... Ia.5				
	30/120 kHz	--	Ia.6 ... Ia.7	Ia.6 ... Ib.5		Ia.6 ... Ib.1
	200 kHz [3]					
Velocità di esecuzione operazioni matematiche con numeri reali		2,3 µs/istruzione				
Velocità di esecuzione operazioni booleane		0,08 µs/istruzione				

PLC: CPU, memorie

La memoria di caricamento (carico) è non volatile (flash, SD card) e contiene il programma utente, i dati e la configurazione.

La memoria di lavoro è volatile (RAM) ed è utilizzata per velocizzare l'esecuzione del programma e l'elaborazione dei dati

La memoria a ritenzione consente di salvare in Flash parte della memoria di lavoro (es. Merker da MB0 a MBx)

La dimensione dell'immagine di processo limita il numero di moduli

Caratteristica		CPU 1211C	CPU 1212C	CPU 1214C	CPU 1215C	CPU 1217C
Dimensioni di ingombro (mm)		90 x 100 x 75		110 x 100 x 75	130 x 100 x 75	150 x 100 x 75
Memoria utente	Lavoro	50 Kbyte	75 Kbyte	100 Kbyte	125 Kbyte	150 Kbyte
	Carico	1 Mbyte	2 Mbyte	4 Mbyte		
	Ritenzione	10 Kbyte				
Dimensione dell'immagine di processo	Ingressi (I)	1024 byte				
	Uscite (Q)	1024 byte				
Memoria di merker (M)		4096 byte		8192 byte		

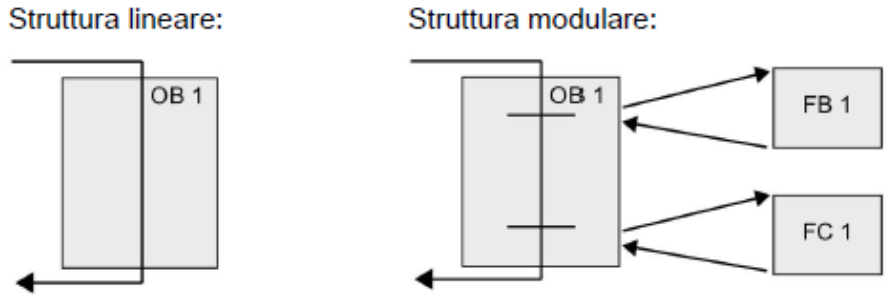
PLC: CPU, memorie

Il forzamento permette di accedere direttamente a ingressi e uscite senza passare dalle immagini di processo

Area di memoria	Descrizione	Forza-mento	Ritenzione
I Immagine di processo degli ingressi I_:P [1] (ingresso fisico)	Viene copiata dagli ingressi fisici all'inizio del ciclo di scansione	No	No
	Lettura diretta degli ingressi fisici della CPU e degli SB ed SM	Sì	No
Q Immagine di processo delle uscite Q_:P [1] (uscita fisica)	Copiata nelle uscite fisiche all'inizio del ciclo di scansione	No	No
	Scrittura diretta nelle uscite fisiche della CPU e degli SB ed SM	Sì	No
M Memoria di merker	Memoria di comando e di dati	No	Sì (opzionale)
L Memoria temporanea	Dati temporanei per un blocco, locali nel bloc-co specifico	No	No
DB Blocco dati	Memoria di dati e, nel caso degli FB, anche memoria per i parametri	No	Sì (opzionale)

PLC: CPU e programma utente

La CPU supporta più tipi di blocchi di codice o dati, in accordo allo standard IEC 61131:



- **OB (blocchi organizzativi)**
  - **OB1 inserito nel ciclo di scansione**
  - **OB100 attivato al reset del PLC (da STOP a RUN)**
  - **OB30 intervento periodico (interrupt)**
  - **… Ci sono anche OB relativi a guasti (es. estrazione moduli) o a errori (es. divisione per 0)**
- **FC (funzioni)**
  - **Sottoprogrammi richiamati all'interno degli OB che non istanziano blocchi dati (che non hanno memoria, ad esempio radice quadrata)**
- **FB (blocchi funzionali)**
  - **Sottoprogrammi richiamati all'interno degli OB che istanziano blocchi dati (che hanno memoria propria, ad esempio i timer, che sono FB di sistema)**
- **DB (blocchi dati)**
  - **Aree di memoria dedicate ai dati (es. l'area di memoria che corrisponde ad un counter)**

PLC: Tecniche di programmazione, prog. "base"

1. Partire da un programma esistente da i seguenti vantaggi

- **La configurazione dell'hardware è già fatta**
- **La tabella dei simboli è già impostata**
- **Alcune funzioni "base", come i filtri software all'inserzione (>>1 ms) sui segnali o come la predisposizione di variabili o uscite è già presente**

Nel nostro PLC la configurazione dell'HW prevede:

- **Filtri hardware (~ 1ms) sui segnali digitali (es. attendi che la commutazione sia stabile per oltre 1 ms e attivala sulle IPI)**
- **Filtri hardware sui segnali analogici (attenuazione dell'effetto del rumore)**

Utilizzare il programma "base" invece del programma "vuoto"

PLC: Tecniche di programmazione, prog. "base"

Il programma "base" si usa nel seguente modo

- Configurare le costanti tra le quali la maschera ingressi, i tempi dei filtri software all'inserzione e il valore iniziale delle uscite ed eventuali altre costanti di interesse
- Dare nomi appropriati agli ingressi di interesse (M2.x-M3.x), ai fronti positivi degli ingressi di interesse (M4.x-M5.x), ai fronti negativi degli ingressi di interesse (M6.x-M7.x), alle uscite di interesse (Q0.x-Q1.x), alle uscite del ciclo precedente di interesse (M10.x-M11.x) e a eventuali altre variabili a partire da M20.x

  NOTA: se si utilizza il simulatore, dare nomi appropriati anche a I0.x-I1.x
- Verificare/modificare il programma di Startup (OB100) che viene eseguito una sola volta (non in modo ciclico) quando il PLC passa da Stop a Run
- Completare il Main (OB1) secondo le specifiche funzionali

PLC: Tecniche di programmazione, I/Q/M

In un programma per PLC ci sono sempre

1. **Ingressi I (nel S7-1215 digitali I0.x-I1.x; analogici IW64 e IW66)**

   Gli ingressi si leggono attraverso le immagini di processo IPI. Se utilizziamo Prog_Base si leggono attraverso memorie. Gli ingressi non si possono scrivere!

2. **Uscite Q (nel S7-1215 digitali Q0.x-Q1.x; analogici QW64 e QW66)**

   Le uscite si scrivono attraverso le immagini di processo IPU e possono essere rilette. Se utilizziamo Prog_Base si leggono attraverso memorie, che contengono il valore reale del ciclo precedente

In un programma per PLC ci sono spesso

1. **Costanti** **E' un modo comodo per definire valori Es. tempo_max_motore3 = T#2s**
2. **Variabili M (nel S7-1215 organizzata a byte, dove x=indirizzo byte)**

   Area M nel S7-1215, che può contenere bit MY.x, byte MBx, word MWx, ecc. Le variabili di memoria si definiscono nella tabella delle variabili e si scrivono e si leggono negli OB/FC

3. **Blocchi dati DB (gruppi di variabili –es. Timer- non analizzati in questo corso)**

PLC: Tecniche di programmazione, analisi

Date le specifiche funzionali

1. **Identificare ingressi e uscite**
2. **ripartire il problema in più sottoproblemi distinguendo tra sottoproblemi indipendenti e dipendenti dalle uscite di altri sottoproblemi**

Partendo dai sottoproblemi indipendenti

- **Analizzare se si tratta di una funzione di assegnazione o di Set-Reset**
  - **Assegnazione: l'uscita del sottoproblema viene modificata ad ogni ciclo di programma. Scrivere la funzione mediante tabella della verità**
  - **Set-Reset: ci sono casi nei quali l'uscita non subisce modifiche (non viene posta a 1 –set- e non viene posta a 0 –reset-). Analizzare il sottoproblema in logica Set-Reset**
  - **NOTA: se vi sono pulsanti, che non hanno un'azione permanente ma impulsiva e da memorizzare, normalmente si usano logiche Set-Reset (S-R)**

PLC: Tecniche di programmazione, logiche S-R

Logiche Set-Reset (S-R)

- **Capire se si tratta di una logica con prevalenza al Set al Reset o alla memorizzazione interrogandosi, in caso di dubbio, se sia una condizione più sicura quella con la funzione a 1 (set prevalente) o a 0 (reset prevalente)**
- Identificare condizioni che attivano la funzione **prevalente** (es. il set, ossia il forzamento a 1 dell'uscita del sottoproblema) anche mediante tabella della verità.
  - Nota: la funzione prevalente spesso si realizza mediante OR (è sufficiente avere una condizione tra tante per agire secondo la funzione prevalente)
- Identificare condizioni che attivano la funzione **non prevalente** (es. il reset, ossia il forzamento a 0 dell'uscita del sottoproblema) anche con la tabella della verità.
  - Nota: la funzione non prevalente spesso si realizza mediante AND (è necessario avere tante condizioni contemporaneamente per agire secondo la funzione non prevalente)
  - Se ci sono **pulsanti** interrogarli con il fronte di salita per evitare azioni non prevalenti (che tolgono dalla sicurezza) in caso di pulsante incastrato
  - La funzione non prevalente viene spesso "interbloccata" dalla funzione prevalente (es. se Reset non prevalente si avrebbe Res1&Res2&Res3&!Set1&!Set2…)

PLC: Tecniche di programmazione, sicurezza

Stati di un segnale booleano S (S_k = valore di S al ciclo k)

S_k	S_{k-1}	Stato
0	0	Livello fisso a 0 (questa condizione si individua con -|/|-)
0	1	Fronte di discesa (questa condizione si individua con -|N|-)
1	0	Fronte di salita (questa condizione si individua con -|P|-)
1	1	Livello fisso a 1 (questa condizione si individua con -| |-)

- **L'interrogazione con il fronte "indebolisce" l'azione che si ha solo in caso di commutazione del segnale (si usa per il segnali –es. pulsanti- che allontanano dalla situazione di sicurezza)**
- **L'interrogazione con il livello potrebbe generare azione anche in caso di guasto (è a 0 perché manca alimentazione o è rotto oppure va bene ed è a 0). Si usa per segnali che portano in sicurezza**

PLC: Tecniche di programmazione, sicurezza

Interblocchi

- **Unilaterali (prevalenza ad una delle due condizioni)**

  Una condizione A deve prevalere su un'altra condizione B, per cui la "blocca"

  Se A allora…

  Se B&!A allora…

- **Bilaterali (nessuna prevalenza o prevalenza allo stato di memoria)**

  Non c'è una condizione che prevale sull'altra ma le condizioni (es. A, B) sono mutuamente esclusive

  Se A&!B allora…

  Se B&!A allora…

PLC: Tecniche di programmazione, logiche S-R

Come decidere se un'uscita o una variabile F sia da gestire in logica di assegnazione o in logica Set-Reset (S-R)?

Ci sono cicli nei quali F non cambia? Se si → Logiche S-R

NOTA: l'assegnazione può essere gestita in S-R (A->Set; !A->Reset) ma S-R non sempre può essere gestita in assegnazione (con le bobine)

Assegnazione → **0**, **1**

Logiche S-R → **0**, **Memoria**, **1**

Prevalenza delle logiche S-R → **0 (Set "bloccato" da &!<Reset>)**, **Memoria (interblocco bilaterale o nulla)**, **1 (Reset "bloccato" da &!<Set>)**

PLC: Tecniche di programmazione, diagnostica

Variabili diagnostiche o di semplificazione del programma

- **Es. logiche Set-Reset**

  La condizione prevalente viene portata alla bobina di set o di reset e assegnata ad una variabile (es. M20.0). In questo modo &!M20.0 realizza l'interblocco di sicurezza sulla condizione non prevalente

- **Es. variabili intermedie tra le condizioni e l'uscita**

  In questo modo capisco meglio perché l'uscita non da il risultato voluto

- **Es. variabili che interrogano il livello invece del fronte**

  L'interrogazione del fronte diventa vera solo per un ciclo (circa 1 ms) e quindi non è visibile a livello di diagnostica

Segmenti semplici che seguono il costrutto

Se <condizione> allora <operazione> (una o più in parallelo)

PLC: Tecniche di programmazione, timer

Usare un solo tipo di timer (es. TON) e concentrarsi a capire il funzionamento di quello

- **Il TON inizia a contare quando l'ingresso IN commuta da 0 a 1 e si resetta se IN=0 o mediante una specifica istruzione**
- **L'uscita Q del TON va a 1 quando il valore di conteggio ET è ET>PT**

L'uso dei timer deve essere ordinato

- **Il timer deve essere "dichiarato" solo in un segmento dove si fissa la costante di tempo PT e la condizione di ingresso IN**
- **Il timer può essere usato (interrogando la variabile booleana Q o la variabile intera ET) nei segmenti di norma successivi a quello di dichiarazione**
- **Il timer può essere resettato mediante l'istruzione -|RT|- nei segmenti di norma successivi ai segmenti di uso**

TIMER: dichiarazione → uso → reset

PLC: Tecniche di programmazione, linguaggi

Esiste un modello (Norme IEC 61131) di definizione del PLC

- **il PLC è un sistema elettronico a funzionamento digitale destinato all'uso in ambito industriale per l'implementazione di funzioni logiche, di sequenziamento, di temporizzazione, di conteggio e calcolo aritmetico**
- **utilizza una memoria programmabile per l'archiviazione interna di istruzioni organizzate in un funzionamento ripetitivo (ciclico)**
- **controlla, mediante segnali di ingresso ed uscita sia digitali che analogici, vari tipi di sistemi semplici e/o complessi**
- **riceve informazioni dai sensori e da sistemi computerizzati, la elabora e fornisce informazioni ad attuatori e a sistemi computerizzati**
- **La programmazione del PLC è organizzata a blocchi (OB), ciascuno attivato da un suo "task" (dal reset, dal ciclo di scansione, da un evento –es. ingresso digitale che passa da 0 a 1-, da un transitorio di alimentazione, ogni To dove To è un tempo programmabile, da un guasto,…). Gli OB sono contenitori di istruzioni, FC e FB (sottoprogrammi con o senza DB propri)**

PLC: Tecniche di programmazione, linguaggi

Esiste un modello (Norme IEC 61131) di definizione del PLC

Le norme IEC 61131-3 prevedono 5 linguaggi di programmazione dei quali 3 grafici:

- **LD Ladder Diagram (KOP nei PLC Siemens)**
  - trasposizione informatica degli schemi a relè
- **FBD Functional Block Diagram (FUP nei PLC Siemens)**
  - trasposizione informatica degli schemi elettronici dove si evidenziano i segnali e i blocchi di elaborazione
- **SCF Sequential Function Chart (GRAPH nei PLC Siemens)**
  - Formalismo grafico per rappresentare operazioni logiche sequenziali (prima questo, poi quello, poi se … quello,..)

e 2 testuali:

- **IL Instruction List (IL nei PLC Siemens, obsoleto)**
  - Linguaggio di programmazione di basso livello del tipo Accumulatore ⇔ Accumulatore.Operazione.Operando per cui una semplice operazione come C=A&B diventa 1.Acc<=A 2.Acc<=Acc.AND.B 3.Store Acc in C
- **ST Structured Text (SCL nei PLC Siemens)**
  - Linguaggio di programmazione strutturato tipo C (con una sintassi simile al Pascal) che ha sostituito IL

PLC: Tecniche di programmazione, Soft-PLC

Grazie alle norme IEC 61131-3 è possibile realizzare software che permettono di programmare diverse piattaforme hardware (HW):

- **PC (o.s. Windows oppure Linux)**
- **Piattaforme embedded con o.s. (es. Rasperry Pi)**
- **Piattaforme embedded senza o.s. (es. Arduino)**

Soft-PLC:

- **Un Soft-PLC è comunemente definito come un sistema basato su piattaforma PC-compatible e un ambiente di sviluppo (IDE) che permette di programmarlo in accordo a IEC 61131-3**

I progetti sui Soft-PLC:

- **https://plcopen.org Dal 1992 il riferimento sullo std. IEC 61131**
- **https://www.openplcproject.com/ dedicato a specifiche piattaforme HW**
  - **https://www.youtube.com/watch?v=xpTBpFHyluw (FBD)**
- **https://www.controllino.com/ un PLC basato su HW e SW open-source**

PLC: Soft-PLC e Virtual PLC

Hannover Messe (19/04/2023) primo PLC virtuale (Simatic S7-1500V) per piattaforma Industrial Operation X (Siemens Industrial Edge)

- Tecnologia a container con Market di APP
- Potenti egde device (CPU multi-core) x più APP, tra le quali il PLC (licenze)
- Convergenza OT (focus su devices e operazioni) e IT (focus sui dati)

Soft-PLC:

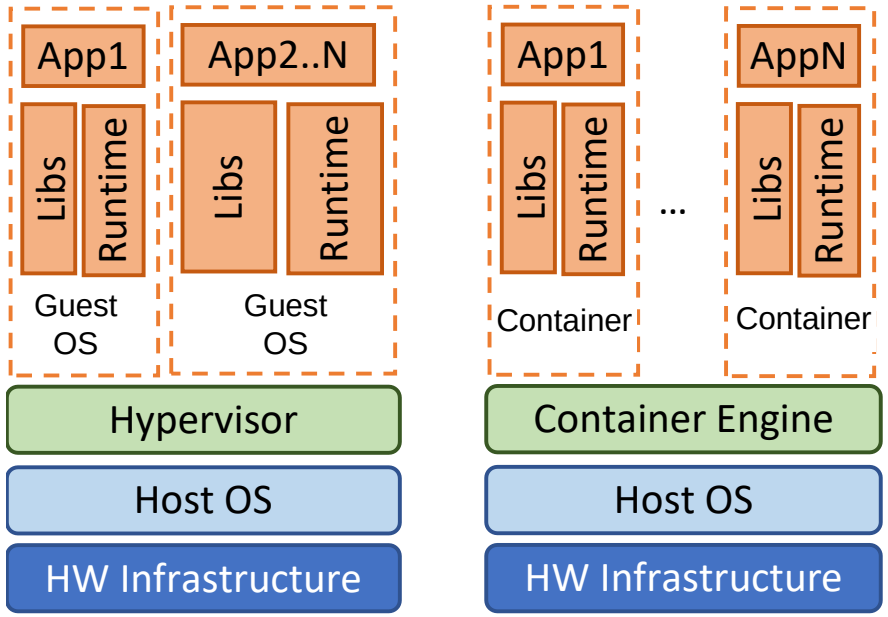
- Invece di un hardware proprietario si usa un PLC, quasi interamente dedicato a fare il PLC
- Possono coesistere altre applicazioni asservite al PLC (es. NON due PLC)

Virtual PLC:

- E' software containerizzato (es. Docker) che non impatta sulle altre APP
- Posso avere più PLC su uno stesso edge device
- I/O su periferia come per Soft-PLC

PLC: Virtual PLC

Containerizzazione è ~ virtualizzazione ma usa meno risorse



Virtualizzazione

Containerizzazione (Es. Docker, Balena OS)

Industrial Edge Device (Simatic IPC227E -Intel Celeron N2930-)

2 virtual PLCs, SCADA, altre APPS containerizzate (esempio)