ITIS "B. Pascal - P.le Macrelli, 100 - 47023 Cesena (FC) - tel. + 39 0547 22792 - fax +39 0547 24141
TDP 5^A ELETTRONICA TELECOMUNICAZIONI Prof, FERRARO GIUSEPPE Ass. Tecnico MARCO FERRETTI
Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

EXSYST 2013

PLC

SIEMENS

S7 1200

ITIS "B. Pascal - P.le Macrelli, 100 - 47023 Cesena (FC) - tel. + 39 0547 22792 - fax +39 0547 24141
TDP 5^A ELETTRONICA TELECOMUNICAZIONI Prof, FERRARO GIUSEPPE Ass. Tecnico MARCO FERRETTI
Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Eserciziario Tia Portal v. 11.1

INDICE

Sommario

Capitolo V : Operazioni di Confronto

Capitolo VI : Funzioni Matematiche

Capitolo VII : Operazioni di Trasferimento

Capitolo VIII : Operazioni di Conversione

Capitolo VIV : Controllo del Programma

Capitolo X : Esercizi Automazione Industriale

Capitolo XI : Combinazione Logica a Parola

Capitolo XII : Spostamento e Rotazione

Capitolo XIII : Data e Ora

Capitolo XIV : String + Char

Capitolo XV : Controllo del Programma

Capitolo XVI : Allarmi

Capitolo XVII : PID e Ctrl PWM

Capitolo XVIII : Gestione Hardware e Tempistica sui Counter

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Capitolo XIX : Ringraziamenti

Introduzione Generale

Farneti Alessandro
Benini Andrea
5AET 2013
(farneti.alessandro@gmail.com)
(andrea.benini1@gmail.com)

I PLC S71200 sono controllori con logica programmabile in grado di controllare un'ampia varietà di applicazioni.

Serve per la progettazione software e la programmazione semplice e rapida di rete.
Sono prodotti moderni, utilizzati nelle aziende di oggi.
L'unità centrale del nuovo PLC ha una vasta gamma di I/O e porte di comunicazioni in espansione, che permettono il loro ampliamento. unicazione tra il software, il PLC e l'HMI.

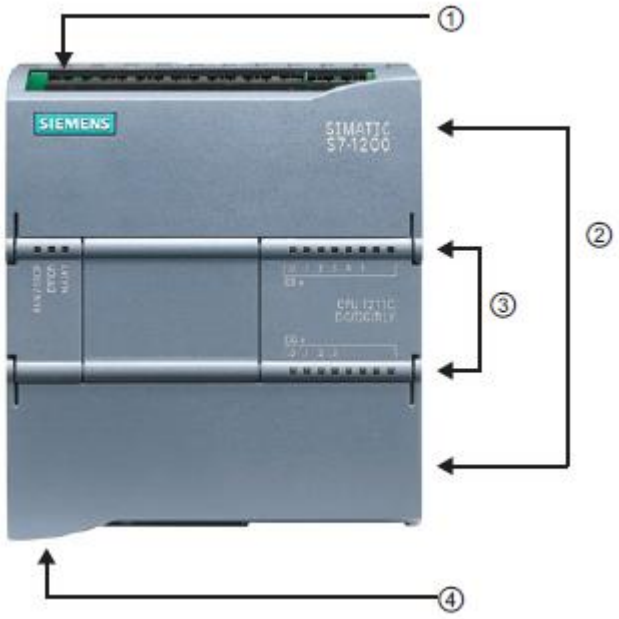


Fig.1: Il PLC e le sue parti principali.

① Connettore di alimentazione

② Morsettiera estraibile per il cablaggio utente (dietro i coperchi)

③ LED di stato per gli I/O onboard

④ Connettore PROFINET (in basso nella CPU)

Signal board:

Le SB consentono di aggiungere input ed output nel dispositivo.
Possono essere costituite da ingressi ed uscite analogici e digitali.

1) Led della SB.

2) Morsettiera per il cablaggio. (estraibile)

unicazione tra il software, il PLC e l'HMI.

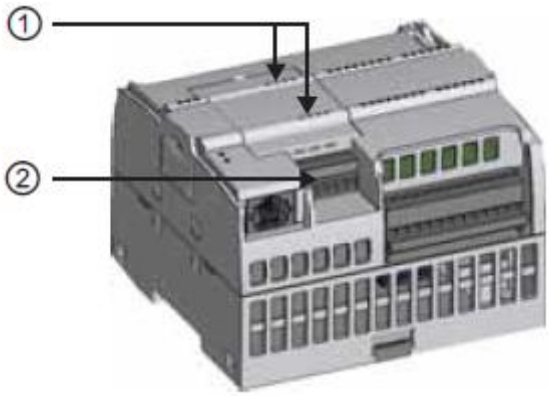
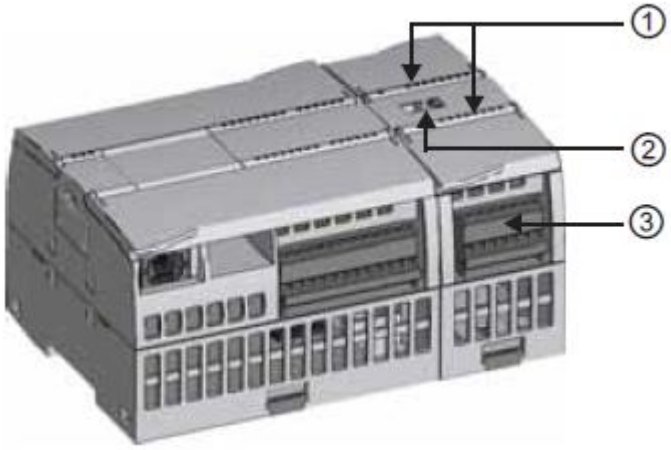


Fig.2: Signal board e locazioni dei led e della morsettiera nel PLC.

Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com



Input & Output:

1) Led di stato degli I/O.

2) Connettore di bus.

3) Morsettiera estraibile per il cablaggio.

Fig.3: Input e output del PLC.

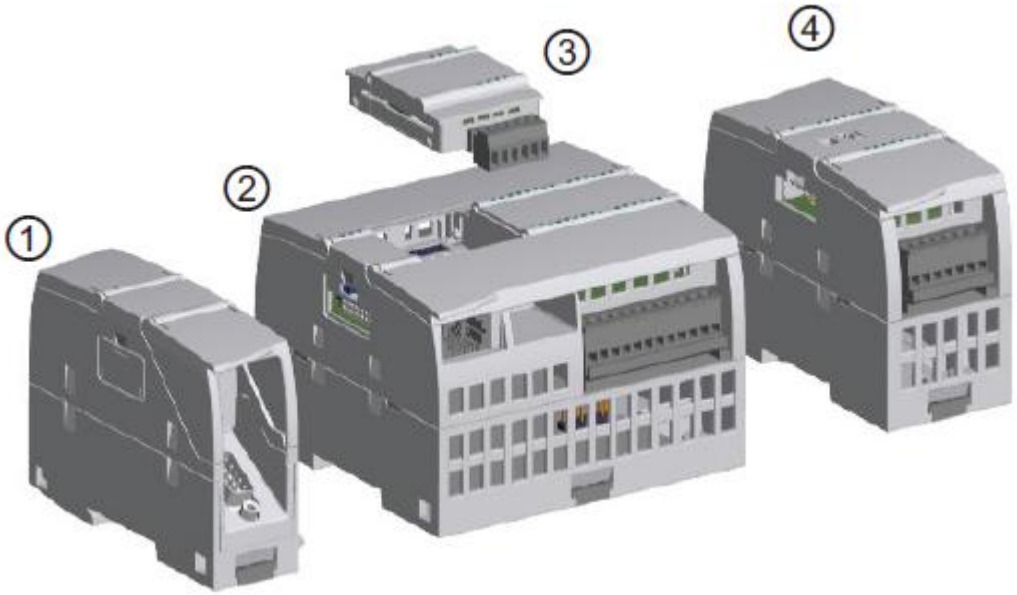


Fig.4: I moduli di comunicazione e per aggiungere I/O al PLC.

① Modulo di comunicazione (CM)
② Signal Board (SB)
③ CPU
④ Modulo di I/O (SM)

Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

Fig.5: Parti inerenti alla comunicazione del PLC.

Comunicazione:

Ci sono due tipologie di comunicazioni seriali : RS232 e RS485.

1) Led di stato della comunicazione.
(se ON comunicazione corretta)

2) Connettore di comunicazione.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Informazioni specifiche del plc

Farneti Alessandro 5AET 2013
(farneti.alessandro@gmail.com)

Il modello 1214 AC\DC è suddiviso in **3 memorie principali** : una memoria **utente** 50 Kybte, una memoria di **caricamento** 2 Mbyte ed infine una di **ritenzione**(area di memoria ram nella quale non vengano persi i dati, in presenza di una caduta di tensione. Se non si possiede la memoria di ritenzione si può verificare la perdita delle informazione : un esempio può essere nel caso del nastro trasportatore dove non ho una memoria di ritenzione, e si verifica una caduta di tensione, perdo tutte le informazioni. Queste memorie non sono molto elevate infatti hanno una dimensione contenuta ; il modello migliore della serie è l's300. Sono gestiti da un **real time clock,** l'orologio di sistema, che ha una batteria dedicata con ioni di litio in modo che non perde tempo nel caricamento.

-14 ingressi e 10 uscite per gli i\o digitali ; gli ingressi analogici presenti sono due.

Dimensione dell'immagine di processo : il programma del plc gira in continuazione, attiva - disattiva in ingressi. Tutto questo avviene in un tempo fisso detto **tempo di ciclo**. A seconda del plc che possiedo, questo tempo può essere regolabile. Ad esempio nel plc c'è un tempo ciclo di **1ms.**

Se il tempo di ciclo è veloce e devo misurare un encoder (15khz), in un ms non riesco a misurare tutte le pulsazioni per cui si necessita dei contatori veloci. Ci sono anche i contatori normali oltre a quelli veloci.

Un'altra memoria importante è la **memoria di tipo MEK,** ovvero quell' area di memoria ram dove vanno le variabile e i conti. Sul lato destro e sul lato sinistro del plc ci sono dei bus interni che consentono di ampliare gli i\o.

Moduli di comunicazioni : protocollo di comunicazione utilizzato è un **profinet.**

I contatori veloci sono complessivamente 6 : possono essere in monofase o in quadratura di fase. Il primo significa che ho un solo segnale, si parla quindi di encoder tachimetrici (arrivano un onda quadra in uscita proporzionale alla velocità dell'encoder) mentre quelli incrementali o ad inquadratura di fase possiede due segnali denominati rispettivamente A-B. (quest'ultimo serve per capire se mi muovo sul fronte a o b : se vedo prima il fronte B di A il sistema va in direzione opposta ovvero **control clock wise** e clock wise quando invece si vede prima A di B.)

La frequenza di questi contatori oscilla da 3 a 100 khz (monofase) e 3 a 80 khz (inquadratura di fase).Grazie all'encoder è permesso capire quanti periodi vengono fatti in un giro di 360°. In quelli bus ci sono 5000 impulsi giro oppure 3000.
Uscite a impulsi sono le uscite in frequenza, con relazione Ton\Toff costante. Una memory card siemantic, detta SD CARD, usata per l'aggiornamento del firmware.Tempo di ritenzione dell'orologio hardware.

Immettere una batteria alcune volte può essere un problema : questo viene gestito dal SuperCap. Ovvero condensatori elettrolitici di capacità estremamente elevate (10 F). Capacità molto alta e valori di tensioni di lavoro molto bassa. Quando c'è la tensione di rete, si carica, mentre quando va via si scarica. Quando si carica mantiene l'orario sul real time clock.

Il **MTBF (medium time between failure)** se viene sorpassato questo tempo, si va incontro all'effetto crancking, per cui si arriva alla rottura. Se in una batteria metto in parallelo un condensatore supercap, quest'ultima viene salvaguardata dal condensatore.

Le velocità di esecuzione delle operazioni matematiche è di 18 microsecondi. Mentre le operazioni booleane vengono svolte in 100 nano secondi.

TIPOLOGIE AREE DI MEMORIA

Area di memoria	Descrizione
I Immagine di processo degli ingressi	La CPU copia lo stato degli ingressi fisici nella memoria I all'inizio del ciclo di scansione. Per accedere direttamente agli ingressi fisici o forzarli aggiungere ":P" all'indirizzo o alla variabile (ad es. "Start:P" o I0.3:P).
Q Immagine di processo delle uscite	La CPU copia lo stato della memoria Q nelle uscite fisiche all'inizio del ciclo di scansione. Per accedere direttamente alle uscite fisiche o forzarle aggiungere ":P" all'indirizzo o alla variabile (ad es. "Stop:P" o Q0.3:P).
M Memoria di merker	Il programma utente legge e scrive i dati salvati nella memoria M. Qualsiasi blocco di codice può accedere alla memoria M. Gli indirizzi di questa memoria possono essere configurati in modo da mantenere i valori dei dati dopo un ciclo di spegnimento/riaccensione.
Memoria L "temporanea"	Quando viene richiamato un blocco di codice la CPU assegna la memoria (L) temporanea o locale che potrà essere utilizzata durante l'esecuzione del blocco. Al termine dell'esecuzione la CPU riassegna la memoria locale per l'esecuzione di altri blocchi di codice.
DB Blocco dati	I DB possono essere utilizzati per memorizzare diversi tipi di dati, tra cui lo stato intermedio di un'operazione o altri parametri di comando degli FB e strutture di dati per varie istruzioni, quali i temporizzatori e i contatori. È possibile specificare che il blocco dati sia di lettura/scrittura o di sola lettura. L'accesso alla memoria dei blocchi dati può essere effettuato a bit, byte, parola e doppia parola. Per i blocchi dati di lettura/scrittura è consentito l'accesso sia in lettura che in scrittura. Per i blocchi di sola lettura è consentito solo l'accesso in lettura.

I (immagine di processo degli ingressi): la CPU campiona gli ingressi (fisici) della periferia immediatamente prima dell'esecuzione dell'OB di ciclo di ogni ciclo di scansione e scrive i valori rilevati nell'immagine di processo degli ingressi. L'accesso all'immagine di processo degli ingressi può essere effettuato a bit, byte, parola e doppia parola. E' consentito l'accesso sia in scrittura che in lettura, ma generalmente gli ingressi dell'immagine vengono solo letti.

Bit	**I[indirizzo byte].[indirizzo bit]**	**I0.1**
Byte,parola o doppia parola	I[dimensione][indirizzo byte iniziale]	IB4,IW5 o ID12

Immagine di processo degli ingressi = OB (operation block). Bit uno del banco zero I0.1, possono essere di 1-14-16-23 bit per banco.

Aggiungendo ":P" all'indirizzo si fa in modo che gli ingressi digitali e analogici della CPU, dell'SB o dell'SM vengano letti immediatamente. La differenza tra un accesso mediante I_:P invece che l consiste nel fatto che i dati provengono direttamente dall'ingresso a cui si accede invece che dall'immagine di processo degli ingressi. L'accesso I_:P è considerato una "lettura diretta " perché i dati vengono prelevati direttamente dall'origine invece che dalla copia dell'ultima immagine di processo degli ingressi aggiornata.

Poiché gli ingressi fisici ricevono i loro valori direttamente dall'apparecchiatura da campo a cui sono collegati, non è consentito scrivervi. Ciò significa che gli accessi I_:P sono di sola lettura, diversamente dagli accessi l che possono essere di lettura o di scrittura.

Gli accessi I_:P sono inoltre limitati alla dimensione degli ingressi supportati da una singola CPU o modulo SB o SM, arrotondata al byte più vicino. Se, ad esempio gli ingressi 2 DI / 2 DQ di un SB sono stati configurati in modo da iniziare da l4.0, vi si può accedere con l4.0:P e l4.1:P oppure IB4. Gli accessi a l4.2:P – l4.7:P vengono comunque accettati ma non hanno senso perché non vengono utilizzati. Gli accessi a IW4:P e ID4:P non sono consentiti perché superano l'offset di byte associato all'SB.

Gli accessi con I_:P non influiscono sul valore corrispondente memorizzato nell'immagine di processo degli ingressi.

Bit	I[indirizzo byte].[indirizzo bit]	I0.1
Byte,parola o doppia parola	I[dimensione][indirizzo byte iniziale]	IB4,IW5 o ID12

Q (immagine di processo delle uscite): La CPU copia nelle uscite fisiche i valori memorizzati nell'immagine di processo delle uscite. L'accesso all'immagine di processo delle uscite può essere effettuato a bit, byte, parola e doppia parola. E' consentita l'acceso sia in lettura che in scrittura.

Bit	Q[indirizzo byte].[indirizzo bit]	Q1.1
Byte,parola o doppia parola	Q[dimensione].[indirizzo byte iniziale]	QB5,QW10 o QD40

Posso avere la necessità di attivare direttamente la uscite e questa cosa non è molto buona perché è meglio avere un tempo fisso. Se utilizza dei merker si può risolvere il problema : consente di leggerla perché il programma non è più originale.

Area merker : accedere informazione, bit per fare dei conti.

M (area dei merker): l'area dei merker (memoria M) può essere utilizzata sia per i relè di controllo che per i dati al fine di memorizzare lo stato intermedia di un'operazione o altre informazioni di comando. L'accesso all'area dei merker può essere effettuato a bit, byte parola e doppia parola. E' consentito l'accesso sia in lettura che in scrittura.

Bit	M[indirizzo byte].[indirizzo bit]	M26.7
Byte,parola o doppia parola	M[dimensione].[indirizzo byte iniziale]	MB20,MW30, MD50

Temp (memoria temporanea): La CPU assegna la memoria temporanea in base alla necessità. Quella per il blocco di codice viene assegnata all'avvio (nel caso degli OB) o al richiamo (nel caso delle FC o degli FB) del blocco. E' possibile che vengano assegnati a un blocco di codice gli stessi indirizzi di memoria temporanea precedentemente utilizzata da un'altra FC o un altro OB e FB. La CPU non inizializza la memoria temporanea durante l'assegnazione, per cui la memoria può contenere qualsiasi valore.

La memoria temporanea è simile alla memoria M con un'eccezione fondamentale: la memoria M è "globale" mentre la memoria L è "locale":

- Memoria M: qualsiasi OB, FC o FB può accedere ai dati di questa area di memoria, ovvero i dati sono disponibili globalmente per tutti gli elementi del programma utente
- Memoria temporanea : l'accesso ai dati di questa area è limitato all'OB, l'FC e l'FB che ha creato o dichiarato l'indirizzo di memoria temporanea. Gli indirizzi restano locali e non sono condiviso da blocchi di codice diversi, neppure se il blocco di codice ne richiama un altro. Ad esempio : quando un OB richiama un FC,

quest'ultima non può accedere alla memoria temporanea dell'OB da cui è stata richiamata.

La CPU mette a disposizione una memoria temporanea (locale) per ciascuna delle tre classi di priorità degli OB :

- 16 Kbyte per gli OB di avvio e di ciclo compresi gli FB e le FC associati
- 4 Kbyte per gli eventi di allarme standard compresi gli FB e le FC
- 4 Kbyte per gli eventi di allarme di errore compresi gli FB e le FC

L'accesso alla memoria temporanea può essere effettuato esclusivamente tramite indirizzamento simbolico.

Tipo di dati	**Dimensione (bit)**	**Campo**	**Esempio di costanti**
Bool	1	Da 0 a 1	TRUE,FALSE,0,1
Byte	8	Da 16#00 a 16#FF	16#12,16#AB
Word	16	Da 16#0000 a 16#FFFF	16#ABCD a 16#0001
DWord	32	Da 16#00000000 a 16#FFFFFFFF	16#02468ACE
Char	8	Da 16#00 a 16#FF	'a','t','@'
Sint	8	Da -128 a 127	123,-123
Int	16	Da -32.768 a 32.767	123,-123
DInt	32	Da -2.147.483.648 a 2.147.483.647	123,-123
USint	8	Da 0 a 256	123
Uint	16	Da 0 a 65.535	123
UDint	32	Da 0 a 4.294.967.295	123
Real	32	Da +/- 1,18 * 10^-38 a Da +/- 3,40 * 10^38	123.456.,3.4, -1.2E+12,3.4E-3
LReal	64	Da +/- 2,23 * 10^-308 a Da +/- 1,79 * 10^308	12345.123456789, -1.2E+12,3.4E-3
Time	32	T#24d_20h_31m_23s_648ms… T#24d_20h_31m_23s_648ms… Salvati come -2,147,483,648 ms… +2,147,483,647 ms	T#5m_30s 5#2d T#1d_2h_15m_30x_45 ms
String	Variabile	Da 0 a 254 caratteri in formato di byte	'ABC'

Tipi di variabili presenti : 16# con questa dicitura significa che il 16 è decimale. Il char è un numero intero compreso da 0 a 255. Singolo apice per un char e doppio apice per una stringa. Numero 8 bit con segno si chiama Sint 8, che va da -128 a +127. Usint(unsigned short integer) ovvero intero corto. LReal (long real fino a 10^38). Binary code to decimal che può avere un insieme di 4 bit.

DB (blocco dati): i DB possono essere utilizzati per memorizzare diversi tipi di dati, tra cui lo stato intermedio di un'operazione o altri parametri delle informazioni di comando per gli FB strutture di dati per varie istruzioni, quali i temporizzatori e i contatori. E' possibile specificare che il blocco di dati sia di lettura/scrittura o di sola lettura. L'accesso alla memoria dei blocchi può essere effettuato a bit, byte, parola e doppia parola. Per i blocchi dati di lettura/scrittura è consentito sia l'accesso sia in lettura che in scrittura. Per i blocchi di sola lettura è consentito solo l'accesso in lettura.

Bit	DB[numero blocco dati].DBX[indirizzo byte].[indirizzo bit]	DB1.DBX2.3

Il seguente formato numerico BCD è supportato dalle istruzioni di conversione nonostante non sia disponibile come tipo di dati.

Formato	**Dimensione(bit)**	**Campo numerico**	**Esempi di costanti**
BCD16	16	Da -999 a 999	123,-123
BCD32	32	Da -9999999 a 9999999	1234567,-1234567

E' possibile creare un array che contiene più elementi con tipo di dati semplice. Per creare gli array nell'editor delle variabili PLC.

Per creare un array nell'editor di interfaccia di un blocco, selezionare il tipo di dati "Array [lo … hi] of type", quindi indicare "lo" , "hi" e "type" nel seguente modo :

- lo – l'indice iniziale (più basso) dell'array
- hi – l'indice finale (più alto) dell'array
- type – un esempio di dati semplici, ad esempio BOOL, SINT, UDINIT

Vengono supportati gli indici negativi. Nella colonna nome editor di interfaccia del blocco è possibile attribuire un nome all'array. La seguente tabella mostra alcuni esempi di array così come verrebbero visualizzati nell'editor di interfaccia del blocco:

Nome	**Tipo di dati**	**Commento**
My_Bits	Array[1...10] of BOOL	Questo array contiene 10 valori booleani
My_Data	Array[-5...5] of SINT	Questo array contiene 11 valori SINT,compreso l'indice 0

L'array può essere dichiarato sia con indice positivo che con indice negativo.

- Array_name[i], dove i è l'indice desiderato.

Esempi di array in ingresso a un parametro così come verrebbero visualizzato nell'editor di programma :

- #My_Bits[3] – indirizza il terzo bit dell'array "My_Bits"
- #My_Data[-2] – indirizza il quarto SINT dell'array "My_Data"

Il carattere # viene inserito automaticamente dall'editor di programma

Lunghezza (byte)	Formato	Campo di valori	Esempio di valore immesso
12	Orologio e calendario (Anno-Mese Variabile:Ora:Minuto: Secondo.Nanosecondo)	Min.:DTL#1970-01-01-00:00:00.0 Max.:DTL#2554-12-31-23:59:59.999 999 999	DTL#2008-12-16-20:30:20.250

Ogni componente del DTL contiene un diverso tipo di dati e campo di valori. Il tipo di dati di un valore specificato deve essere uguale a quello dei relativi componenti.

Byte	Componente	Tipo di dati	Campo di valori
0 1	Anno	UINT	Da 1970 a 2554
2	Mese	USINT	Da 1 a 12
3	Giorno	USINT	Da 1 a 31
4	Giorno della settimana	USINT	Da 1(domenica) a 7 (sabato) Il giorno della settimana non viene considerato.
5	Ora	USINT	Da 0 a 23
6	Minuto	USINT	Da 0 59
7	Secondo	USINT	Da 0 a 59
8 9 10 11	Nanosecondi	UDINT	Da 0 a 999 999 999

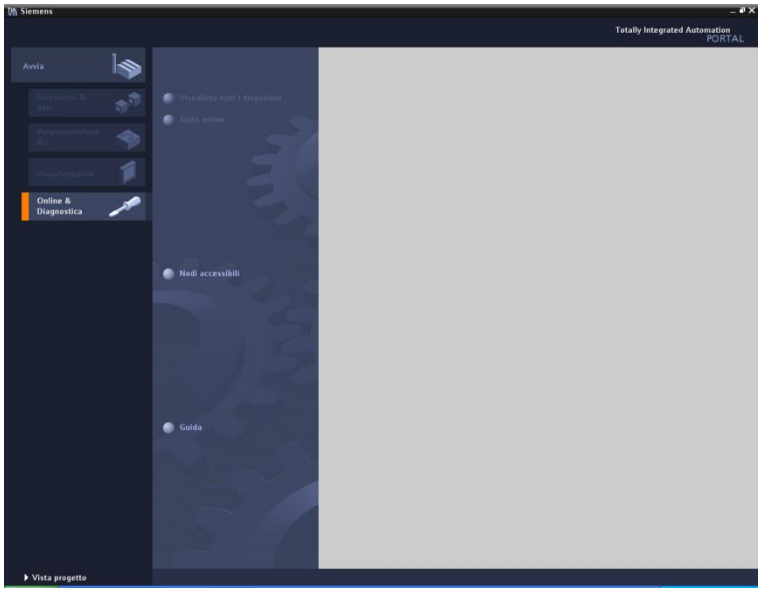
Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

Funzioni principali di Tia portal e accensione di un led

esperienza 1

Farneti Alessandro 5AET 2013
(farneti.alessandro@gmail.com)

Questa è la schermata iniziale del programma TIA portal, dove possiamo trovare le principali funzioni per programmare e per connettere il plc Siemens al pc.

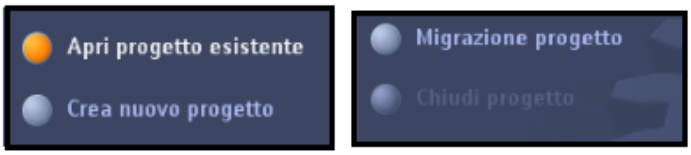


Per connettere il nostro plc al computer, dobbiamo utilizzare una scheda di rete al quale verrà collegato un cavo, che comunicherà direttamente col Siemens. Per questo motivo sono presenti delle schede che ci consentono di capire se la connessione è avvenuta o meno. Una di queste è "**Nodi Accessibili"** che ci consente di capire se il programma sta visualizzando una connessione in rete. Per trovare il nostro dispositivo bisogna cliccare su Nodi Accessibili.

Fig.6: Schermata iniziale del programma TIA Portal.

Per creare un progetto basta andare sulla scheda "**Avvia**" in altro a sinistra, e selezionare "**Crea un nuovo progetto"**.

1) 

2) 

La seguente immagine illustra il tipo di collegamento tra scheda di rete e plc che di dovrà ad andare impostare. Nel nostro caso, in alto a sinistra, nella tendina dove si richiede il "**Tipo di interfaccia PG \ PC**" bisogna impostare l'opzione "**PN/IE"**. In seguito bisogna impostare su quale scheda di rete si lavora : "**N**/C **Fast Ethernet PCI Real**". Una volta che abbiamo selezionato questi valori all'interno della tendina, non ci resta che cliccare su "**Visualizza**" per fare in modo che il computer vada a visualizzare le periferiche connesse. Quando si clicca su visualizza si va a fare una scansione dei nodi accessibili, al fine di trovare il dispositivo.

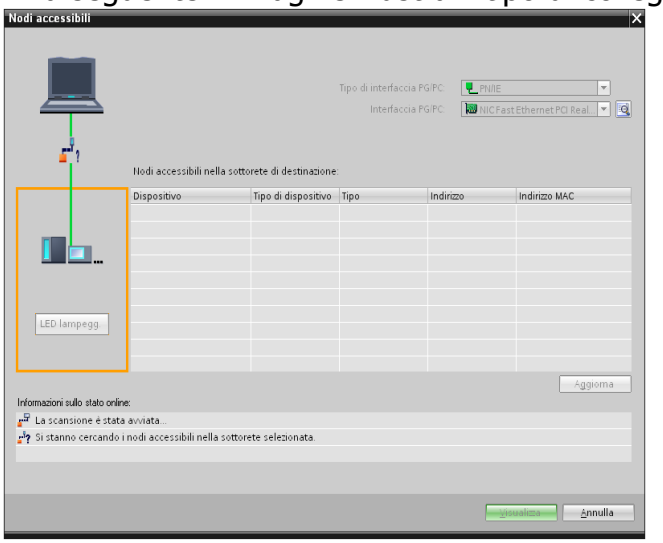


Fig.7: Collegamento tra il PLC e la scheda di rete in TIA.

"Led Lampegg." ci consente di capire se il Siemens è collegato correttamente; se viene premuto avremo il lampeggio dei led sul plc.

Se otteniamo questa schermata significa che il plc è stato trovato per cui ora bisogna andare ad impostare l'indirizzo IP di questa macchina e lo si fa facendo doppio click sul dispositivo trovato.

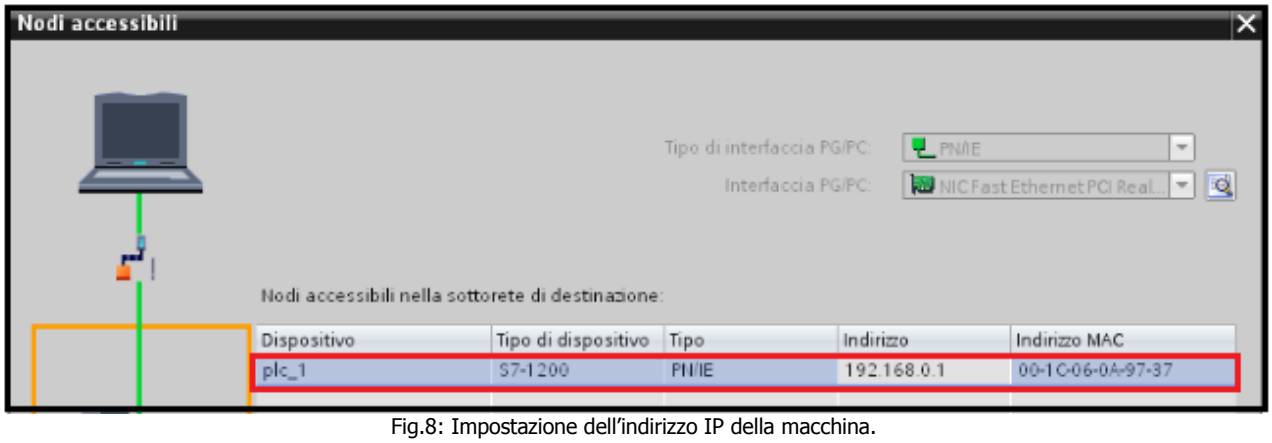


Fig.8: Impostazione dell'indirizzo IP della macchina.

Otterremo la finestra seguente dove andremo a scegliere l'indirizzo IP da attribuire la macchina.Per farlo dovremo andare a selezionare una voce all'interno del menù "**Navigazione del progetto"** nella scheda "**Dispositivi**".
Andremo a cliccare "**Online e Diagnostica**"

per immettere l'indirizzo IP all'interno del plc. Questa operazione deve essere effettuata in quanto il plc Siemens una volta collegato, è privo di indirizzo IP per cui bisogna procedere ad una assegnazione di tipo manuale.

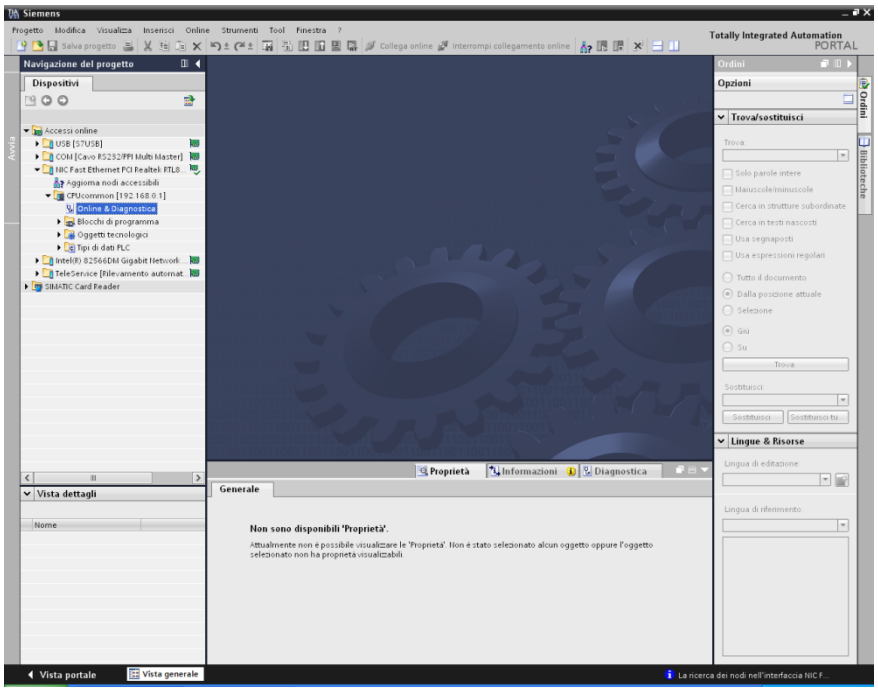
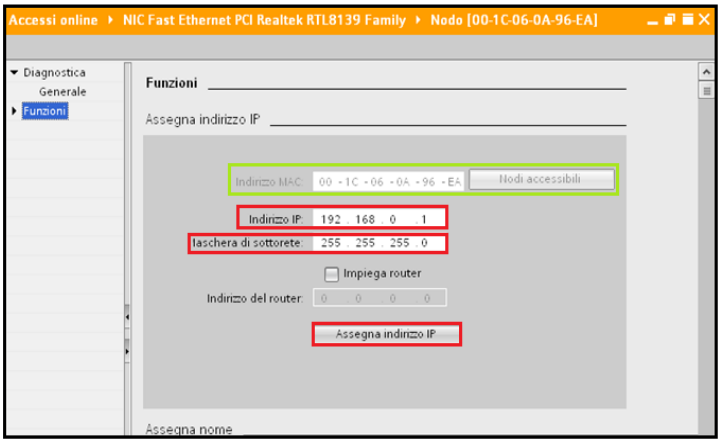


Fig.9: Finestra nella quale scegliamo l'indirizzo IP della macchina.



Si aprirà la seguente finestra nella quale è necessario inserire "**Indirizzo IP**" e "**Maschera di sottorete**", che stabiliranno un indirizzo MAC alla macchina collegata.
L'indirizzo IP da impostare è il seguente mentre la maschera di sottorete è la seguente

Fig.10: Finestra nella quale viene stabilito l'indirizzo MAC.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Per verificare che sia stato effettuato il collegamento, mandiamo un segnale attraverso un comando "**Ping"** da parte del pc. Si tratta di un segnale che viene mandato al plc, e che ritorna per informare che c'è collegamento. Per fare ciò dovremo scrivere "**command**" nella barra di windows, premendo semplicemente su " **Start** ", che può essere anche abbreviato con " **cmd** ". Apparirà una finestra di sfondo nero, in cui dovremmo andare a scrivere il l'indirizzo IP del plc. Ora bisogna scrivere "**ping 192.168.0.1**", e premere invio. Otterremmo la seguente schermata :

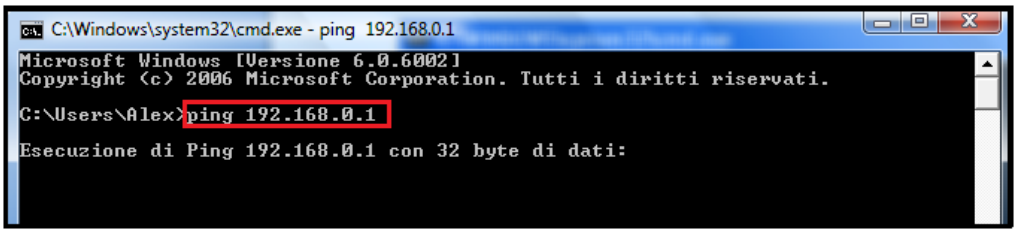


Fig.11: Verifica del collegamento tra PLC e computer.

Appena il pc troverà una comunicazione con il Siemens visualizzeremo i pacchetti di dati inviati tra pc e periferica, in un preciso tempo specificato. Avremmo un **tempo minimo, massimo e medio.**
Nel caso non sia presente questa schermata e visualizziamo il messaggio di " **Richiesta Scaduta** " significa che non esiste collegamento tra scheda di rete e plc, oppure le impostazioni che sono state immesse all'interno di " **Nodi Accessibili** " sono sbagliate.

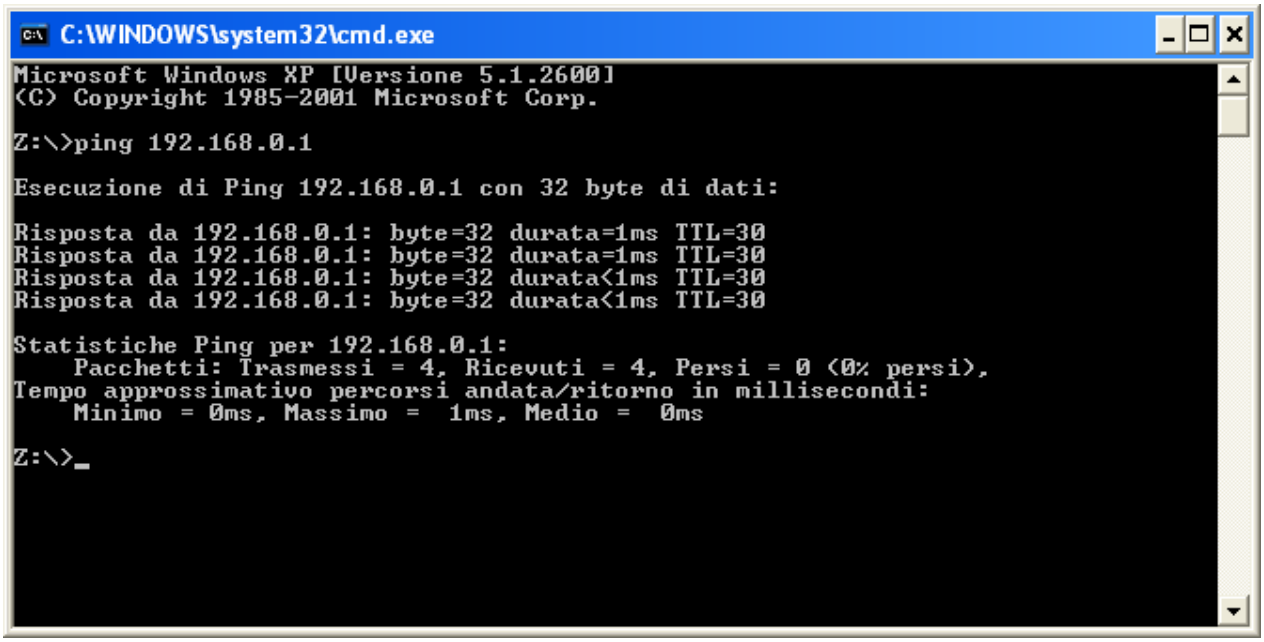
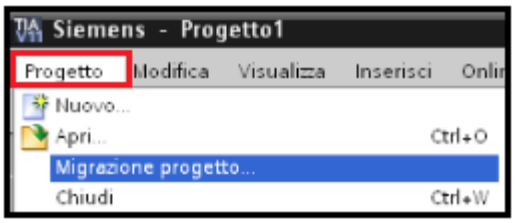


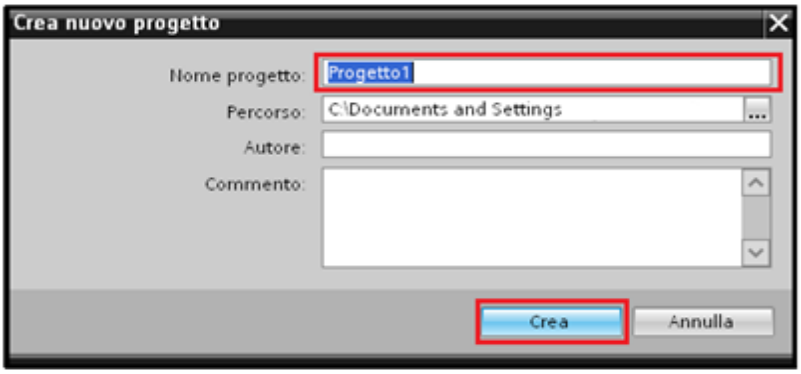
Fig.12: Schermata del corretto collegamento tra PLC e computer.

Ora che ci siamo assicurati di avere una comunicazione tra computer e plc, possiamo creare un nuovo progetto. Oltre alla procedura spiegata inizialmente, un nuovo progetto lo si può creare facendo in questo modo :



Col cursore si clicca su " Progetto " e si va selezionare " **nuovo** ". Oppure lo si può fare semplicemente cliccando sotto l'icona :

Fig.13: Selezione di un nuovo progetto.



Una volta clicca su nuovo progetto avremmo la seguente schermata dove dovremmo immettere il nome del progetto, che è necessario per poter inserire dispositivi e poter accedere al plc. Una volta nominato il progetto si andrà a cliccare su " **Crea** ".

Fig.14: Creazione del nuovo progetto.

A questo punto dovremmo andare ad aggiungere il dispositivo su cui andremo a lavorare per far ciò andiamo a cliccare su "**Aggiungi Dispositivo**" che si trova sulla tendina dei dispositivi a sinistra. Ci si aprirà una finestra come quella qui in basso a destra, nella quale dobbiamo andare a scegliere il nostro modello del plc : nel nostro caso selezioniamo "**CPU 1214C AC/DC**" > "**6ES7 214-1 BE30-0XB0**", versione V 2.2. Una volta terminato questo processo andremo a cliccare su ok.

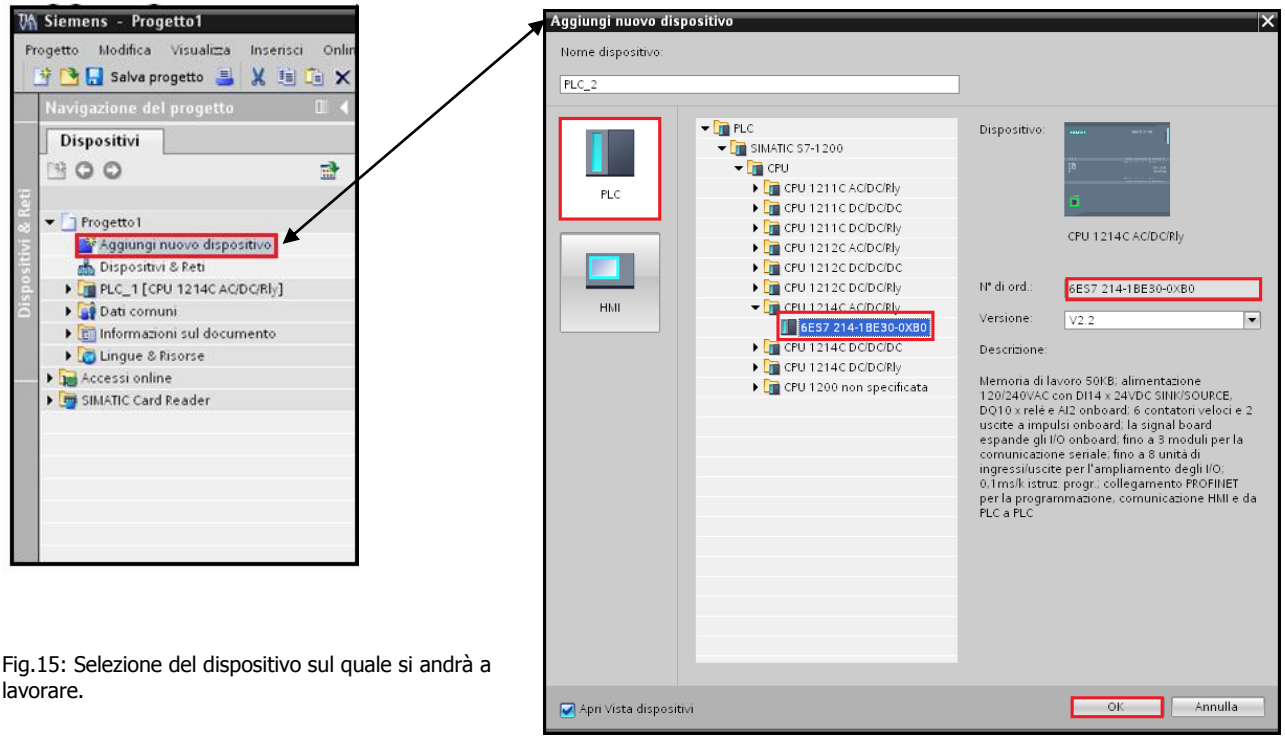


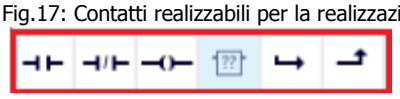
Fig.15: Selezione del dispositivo sul quale si andrà a lavorare.

Se il procedimento è andato a buon fine visualizzeremo questo tipo di modello sulla nostra schermata :
Ora non resta che scrivere il programma all'interno del main, per farlo abbiamo bisogno di andare nella schermata di programmazione.

Fig.16: Rappresentazione grafica del dispositivo sul quale lavoriamo.

Andiamo nella tendina a sinistra e clicchiamo "**Main**" che si trova nella scheda "**Blocchi di Programma**". A questo punto iniziamo a programmare con il linguaggio ladder : dobbiamo fare un semplice programma che vuole accendere un led grazie all'abilitazione di uno switch sul plc. Per far ciò usiamo i seguenti contatti :

Fig.17: Contatti realizzabili per la realizzazione del programma in ladder.



Quindi metteremo un ingresso, che nel Siemens viene identificato come "**I0.0**" e un uscita "**Q0.0**" che sta ad indicare la nostra lampadina. Questo passaggio va fatto trascinando i contatti sul foglio di programmazione.

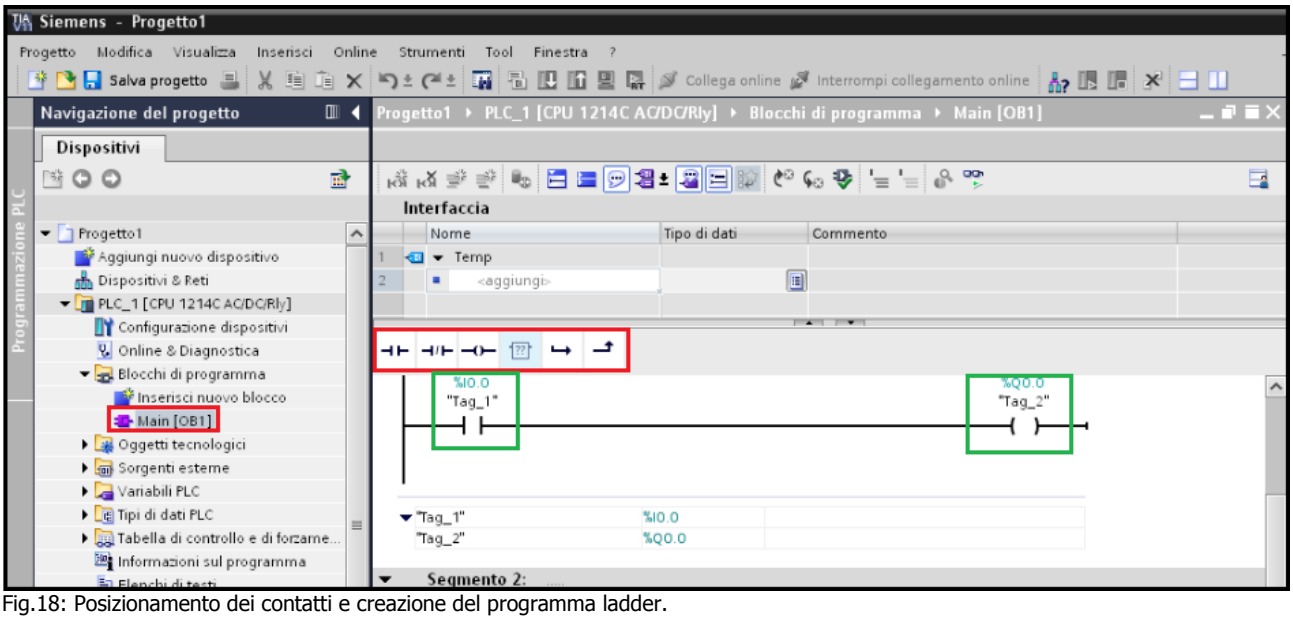
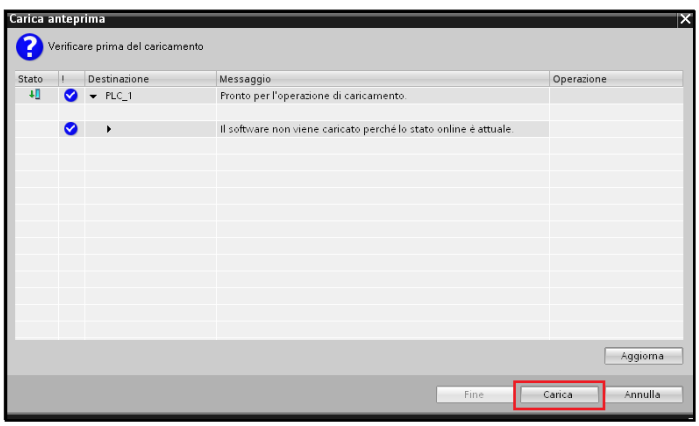


Fig.18: Posizionamento dei contatti e creazione del programma ladder.

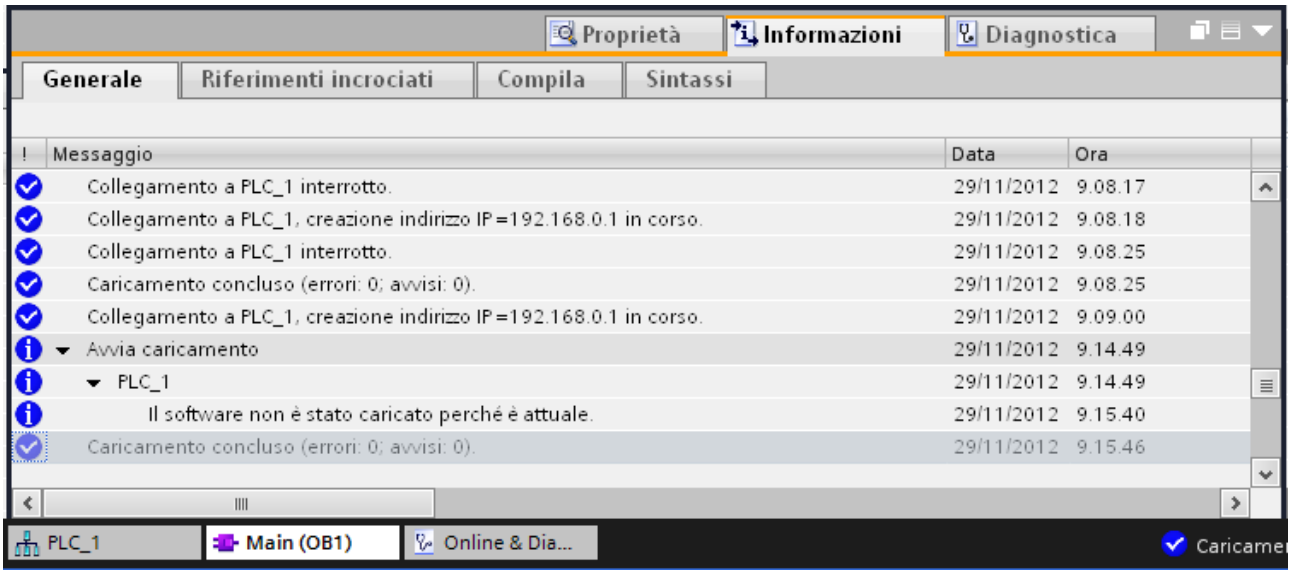


Per far si che il nostro programma funzioni, è necessario che sia caricato all'interno del plc per cui bisogna andare su "**Online**", scollegare il dispositivo e caricare il programma.
Questa operazione si fa cliccando su "**Carica**".

Fig.19: Schermata inerente al caricamento del programma nel PLC.

Una volta che avremmo cliccato su carica, otterremmo la seguente finestra :

Fig.20: Schermata delle informazioni generali riguardanti il caricamento del programma.



La programmazione è avvenuta a buon fine :

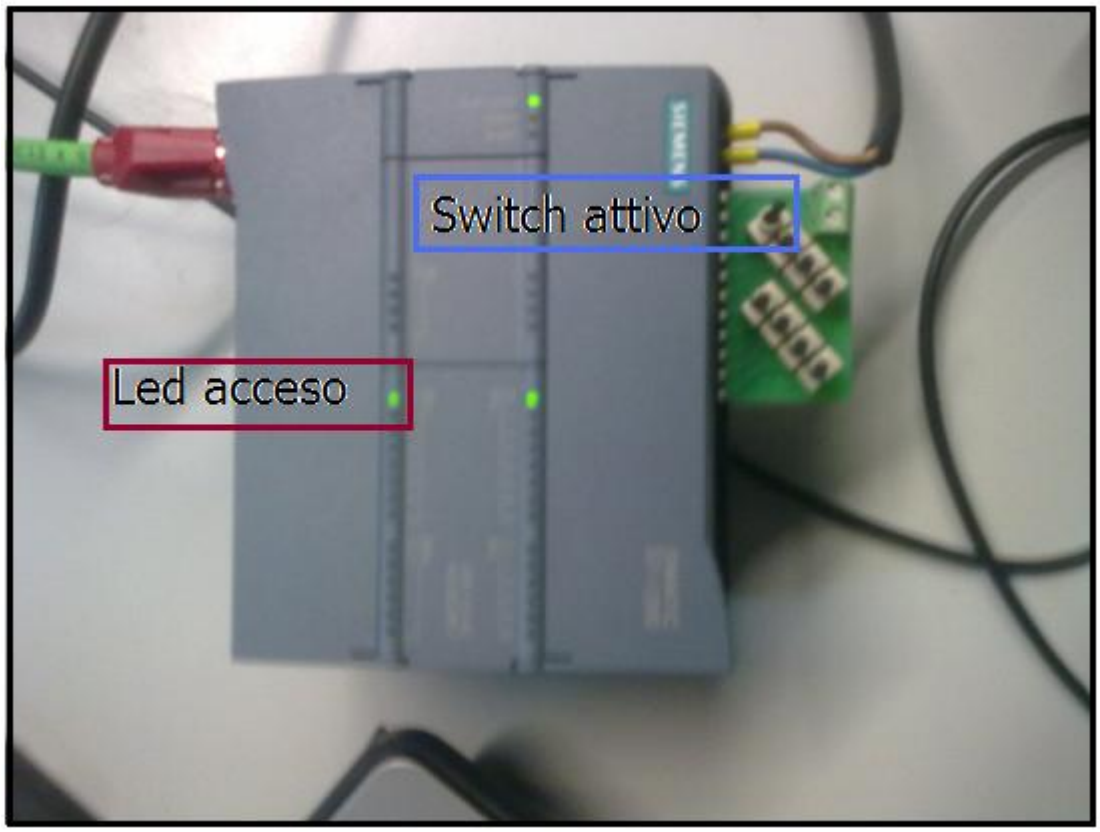


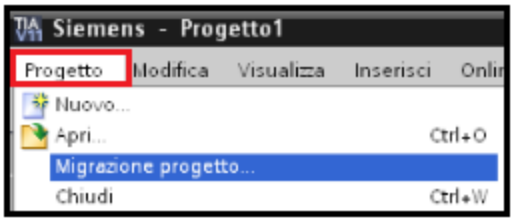
Fig.21: Corretto funzionamento del programma caricato nel PLC.

Relazione sugli operatori logici

esperienza 2

Farneti Alessandro 5AET 2013
(farneti.alessandro@gmail.com)

L'obiettivo di questa esperienza è realizzare i 7 operatori logici e testarli sul plc Siemens.



Per realizzare i vari operatori logici con TIA portal, dovremmo lavorare in un nuovo **progetto** come spiegato sopra (pagina 4).

Fig.22: Creazione del nuovo progetto.

A questo punto selezioniamo il dispositivo che si è scelto in precedenza :"**CPU 1214C AC/DC**" > "**6ES7 214-1 BE30-0XB0**" nella scheda dei dispositivi.

Da " **Navigazione del progetto** " iniziamo a programmare cliccando su " **Main** ".

Ci si aprirà la schermata dove saranno presente la barra degli strumenti che servirà per mettere le entrate e le uscite che fanno parte di ogni operatore logico.
Iniziamo con eseguire il primo operatore :

-AND LOGICO

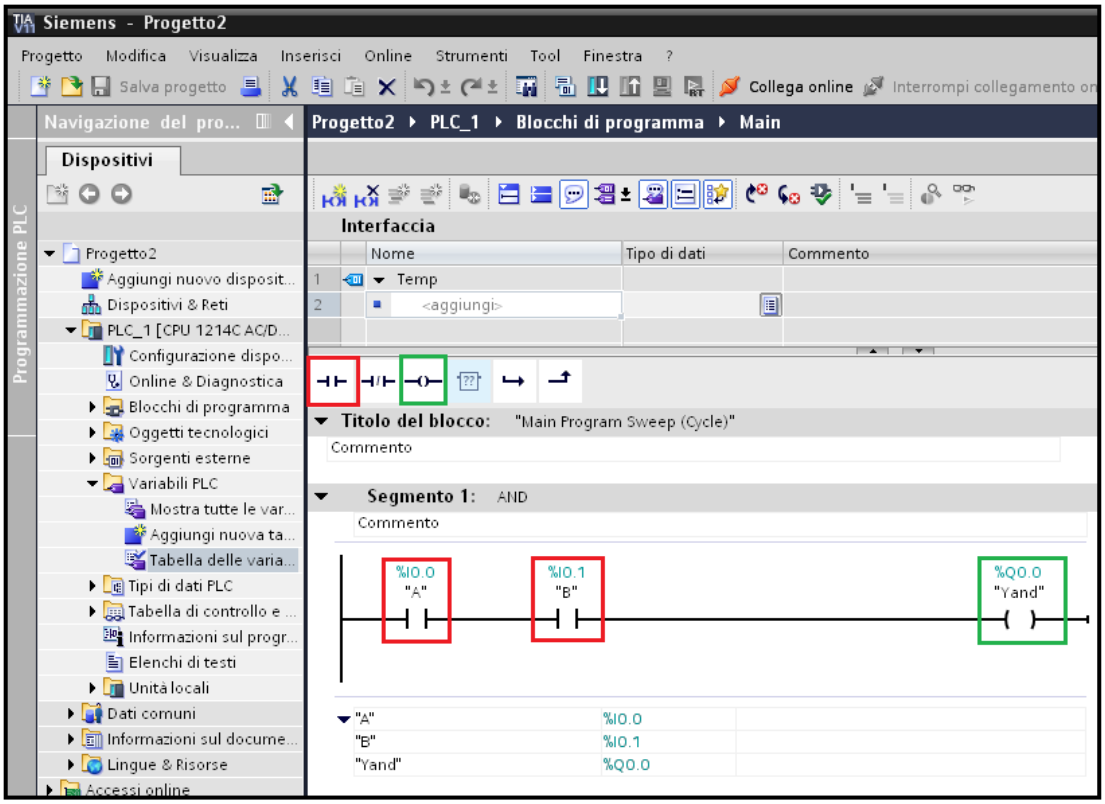
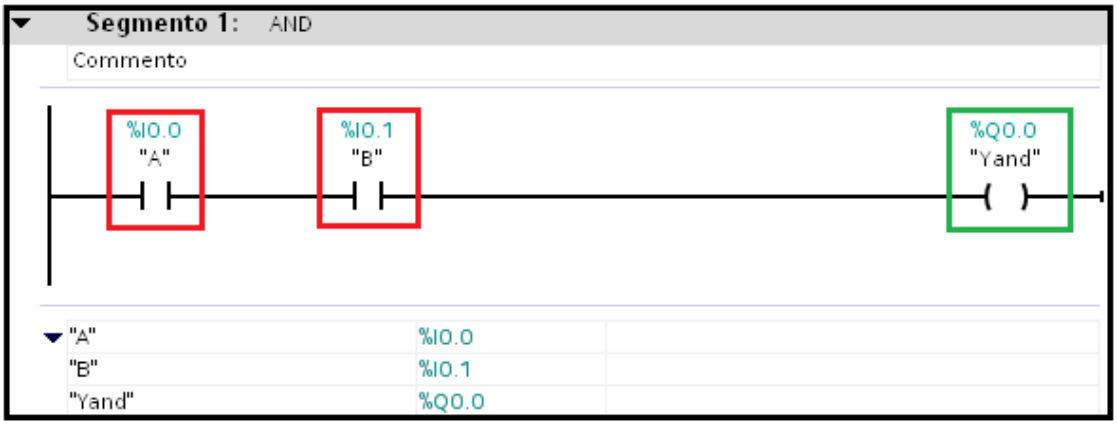


Fig.22: AND logico con programmazione ladder.

Fig.23: Nominazione del segmento contenente il programma in "AND".



Come indicato dalla foto, diamo un nome al nostro " **segmento 1** " in **AND**.
Per realizzarlo, abbiamo bisogno di due ingressi (evidenziati in rosso ),uno collegato all'altro, e infine di un'uscita (evidenziata in verde ).

Visto e considerato che durante la creazione di tutti gli operatori logici si dovrà usare più volte la barra di strumenti, si farà ricorso alle **variabili**, che dovremmo dichiarare in una finestra a parte. Nel momento in cui si crea un ingresso, quest'ultimo verrà nominato automaticamente col nome di " **Tag** " , non bisogna far altro che rinominarlo con altro nome, ad esempio " **A** ".

Per fare questo dobbiamo seguire il seguente percorso :
Navigazione del progetto > Dispositivi > Tabella delle Variabili
Dove c'è scritto " **Nome** " andremo a dichiarare la variabile col nome che vogliamo nel nostro caso chiamiamo un ingresso "**A**" e l'altro "**B**". Questo procedimento vale sia per gli ingressi che per le uscite.

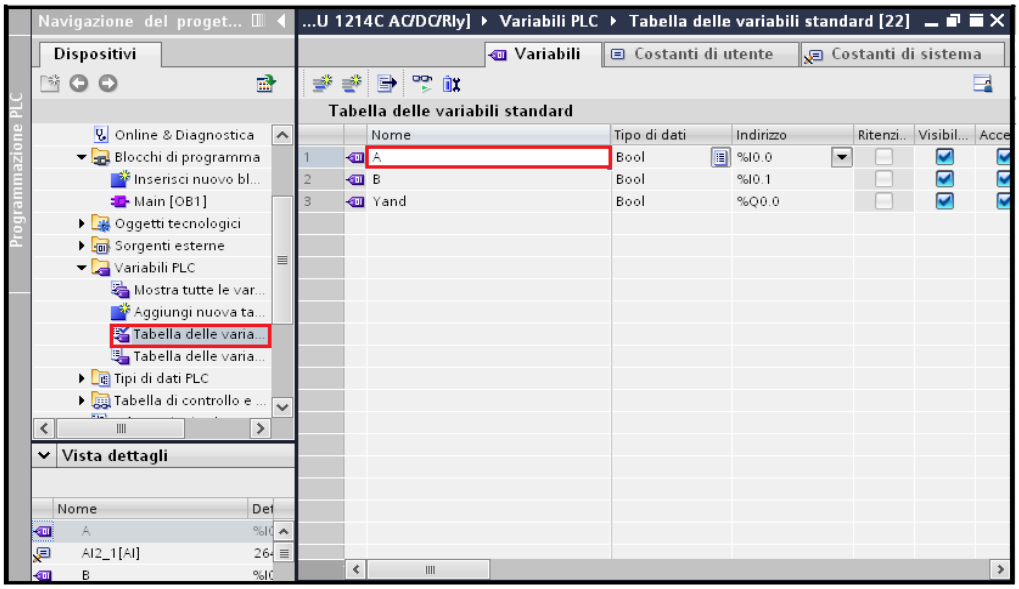


Fig.24: Tabella delle variabili.

N.B. questo procedimento non cambia l'indirizzo dell'ingresso o dell'uscita presa in considerazione ma SOLAMENTE la dichiarazione di ogni elemento.
La visualizzazione degli indirizzi ci è consentita dalla tabella che viene a crearsi subito dopo la realizzazione del segmento, nel caso dell' and avremo la seguente immagine :

-NAND LOGICO

Per realizzare il " **Nand** " la struttura del segmento cambia, bisogna procedere in questo modo :

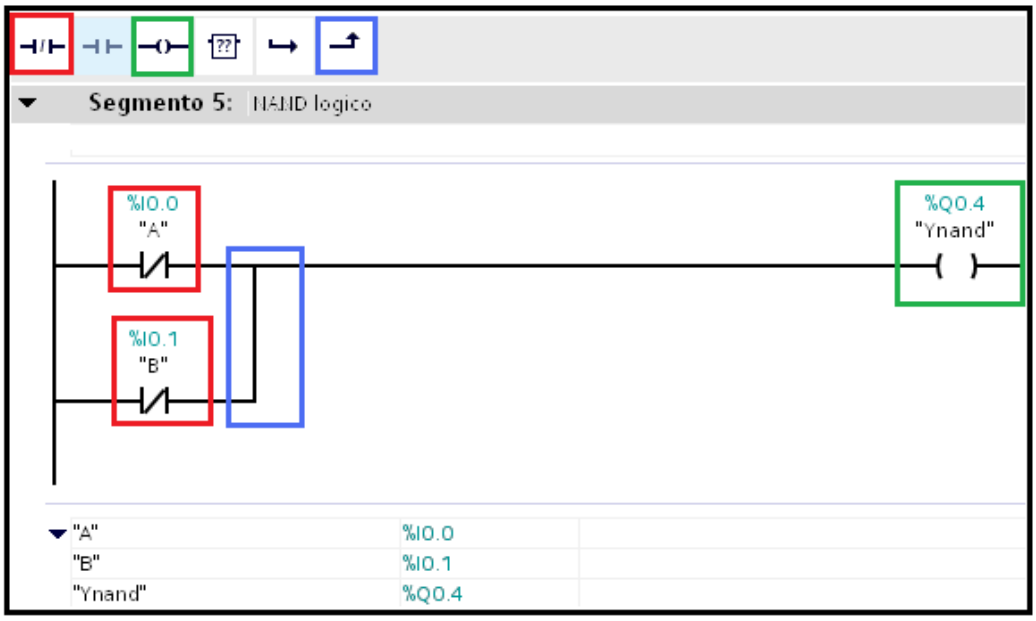


Fig.25: NAND logico con programmazione ladder.

Si inizia con fare i due ingressi negati " **A** " e " **B** " (evidenziati in rosso) che bisogna collegare, usando quindi la" **freccetta** " (evidenziata in blu); infine selezioniamo l'uscita " **Ynand** " (evidenziata in verde).

Per ogni operatore logico si andrà a dichiarare le variabili che andremo ad utilizzare quindi anche nel caso del nand ripeteremo il medesimo procedimento :

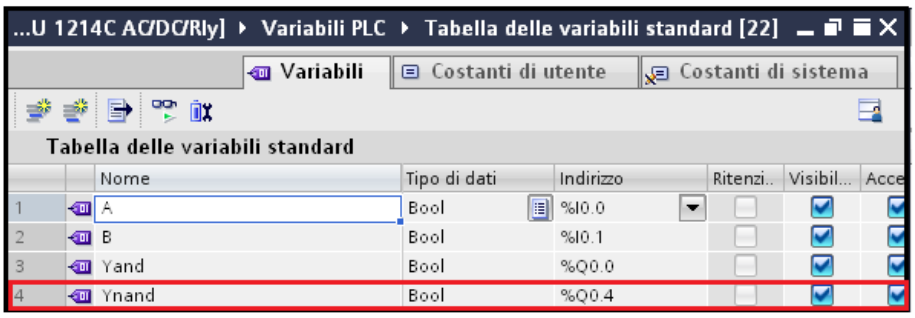


Fig.26: Dichiarazione variabili del programma NAND logico.

Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

-OR LOGICO

Per realizzare questo tipo di operatore logico non abbiamo bisogno di cambiare la struttura del segmento in quanto è simile a quella realizzata precedentemente nel nand. Quello che dobbiamo modificare sono gli ingressi, che non sono negati ma normali.

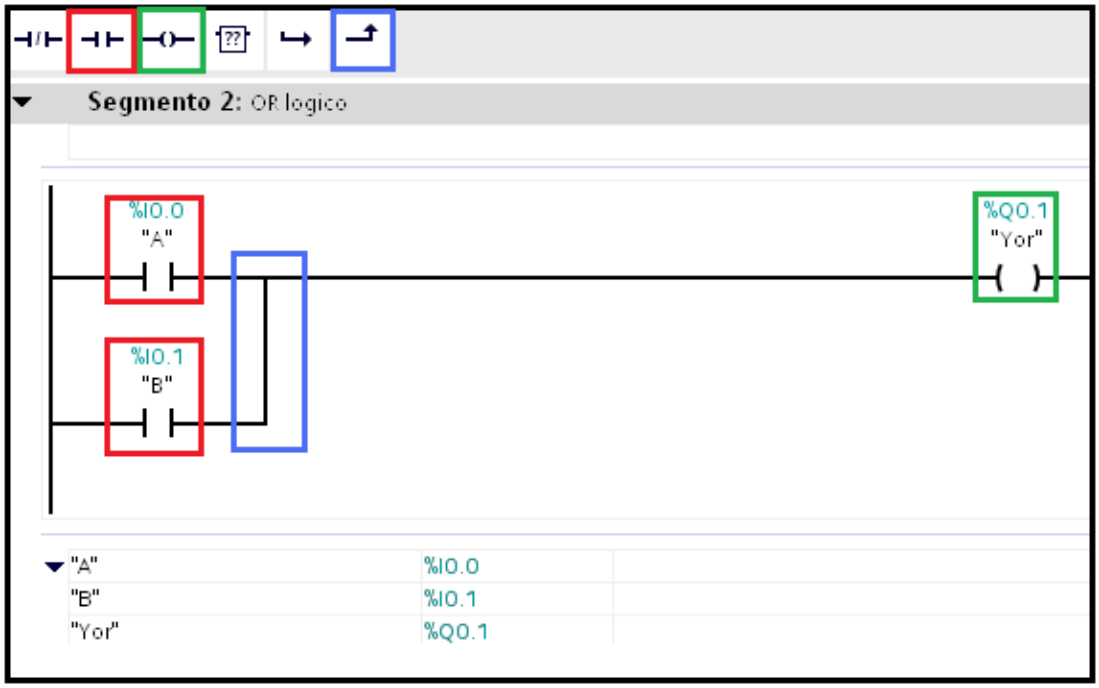


Fig.27: OR logico in programmazione ladder.

Si deve semplicemente cambiare gli ingressi negati con quelli normali, il resto rimane uguale. In questo caso sono gli ingressi " **A** " e " **B** " (evidenziati in rosso ⊣⊢). Ovviamente l'uscita prenderà il nome di " **Yor** " (evidenziata in verde ⊣○⊢) **.**

In seguito dovremmo dichiarare anche queste variabili :

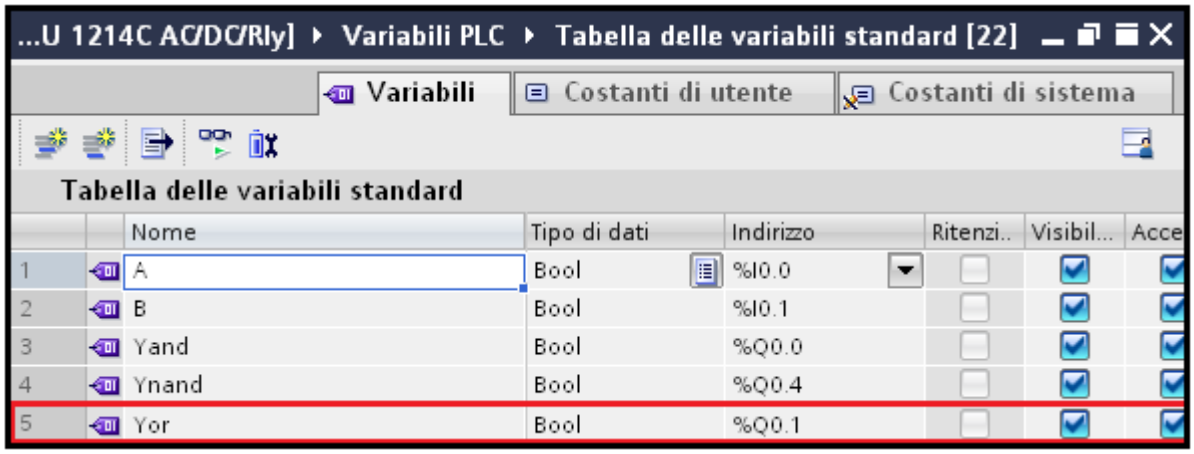


Fig.28: Dichiarazione variabili del programma OR logico.

-NOR LOGICO

Questo operatore si realizza semplicemente collegando in serie due ingressi normalmente chiusi " **A** " e " **B** (evidenziati in rosso) Con rispettiva uscita " **Ynor** ", (evidenziata in verde).

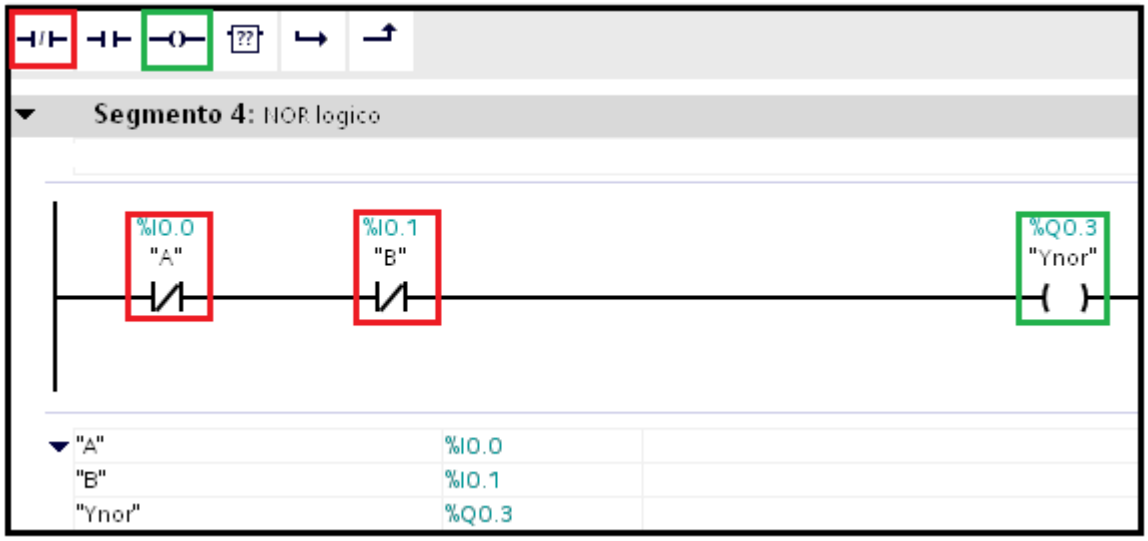


Fig.29: NOR logico in programmazione ladder.

Per ottenere la tabella soprastante dove vengono indicate le variabili dichiarate, bisogna andare nella tabella delle variabili e indicare il nome di ognuna; nel nostro caso avremo la seguente schermata :

Fig.30: Dichiarazione variabili del programma NOR logico.

-XOR e XNOR LOGICI

Il primo operatore " **Xor** " si realizza nel seguente modo :

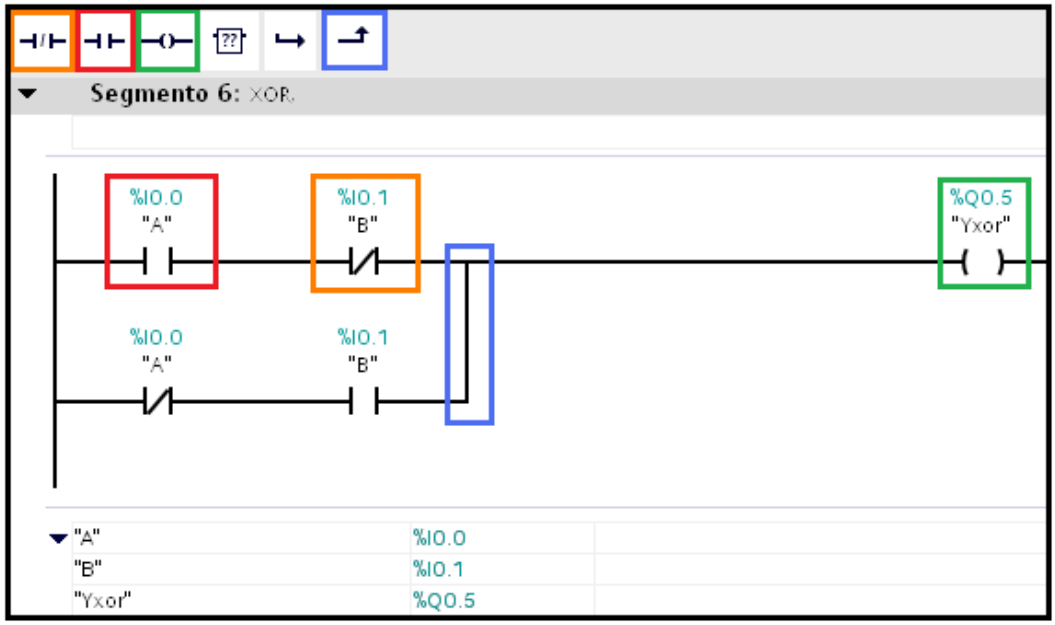


Fig.31: XOR logico in programmazione ladder.

Creiamo **4 ingressi** di cui : i primi due sopra devono essere uno normale e uno negato mentre quelli sotto il contrario. Essi comunicano grazie alla " **freccetta blu** " sopraindicata. L'uscita è " **Yxor** ".

Ingresso n.aperto ( ) ; Ingresso n.chiuso ( ) ; Freccetta ( ) ; Uscita ( ) .

Il secondo operatore " **Xnor** " si compone nel seguente modo :

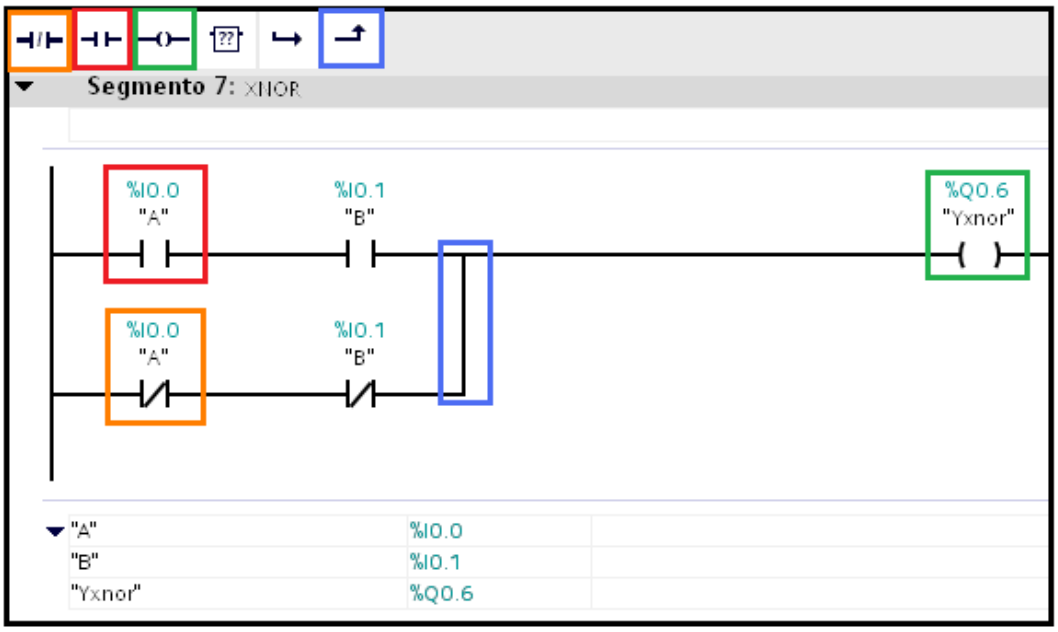
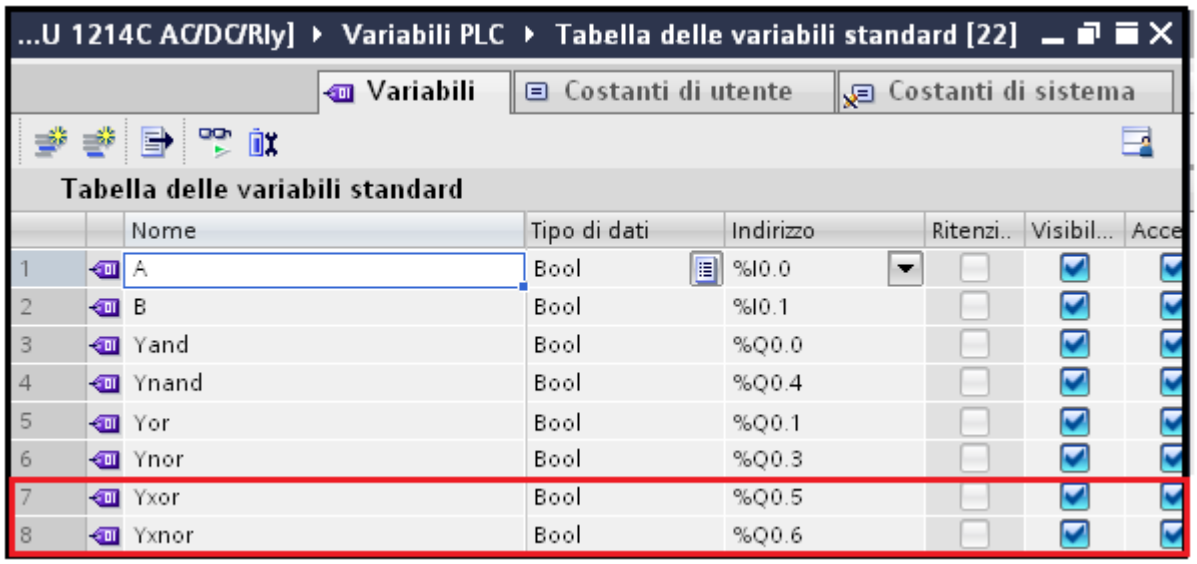


Fig.32: XNOR logico in programmazione ladder.

Anche l'**xnor** possiede **4 ingressi** come l'operatore precedente ma sono disposti differentemente : sopra troviamo 2 ingressi normali, mentre sotto troviamo due ingressi negati. Essi comunicano sempre grazie alla " **freccetta blu** " e il tutto termina con l'uscita " **Yxnor** ".

Ingresso n.aperto ( ) ; Ingresso n.chiuso ( ) ; Freccetta ( ) ; Uscita ( ) .

Entrambe le variabili degli operatori vanno dichiarate all'interno della tabella :



	Nome	Tipo di dati	Indirizzo	Ritenzi..	Visibil...	Acce
1	A	Bool	%I0.0			
2	B	Bool	%I0.1			
3	Yand	Bool	%Q0.0			
4	Ynand	Bool	%Q0.4			
5	Yor	Bool	%Q0.1			
6	Ynor	Bool	%Q0.3			
7	Yxor	Bool	%Q0.5			
8	Yxnor	Bool	%Q0.6			

Fig.33: Dichiarazione variabili sia per l'XOR che per l'XNOR.

-NOT LOGICO

Il not si realizza semplicemente prendendo un ingresso normalmente chiuso e collegandolo a un'uscita chiamata " **Ynot** ". Anche quest'ultimo si aggiunge nella lista delle variabili.

Fig.34: NOT logico in programmazione ladder.

	Nome	Tipo di dati	Indirizzo	Ritenzi..	Visibil...	Acce
1	A	Bool	%I0.0			
2	B	Bool	%I0.1			
3	Yand	Bool	%Q0.0			
4	Ynand	Bool	%Q0.4			
5	Yor	Bool	%Q0.1			
6	Ynor	Bool	%Q0.3			
7	Yxor	Bool	%Q0.5			
8	Yxnor	Bool	%Q0.6			
9	Ynot	Bool	%Q0.2			

Fig.35: Dichiarazione variabili per il NOT.

Relazione sui temporizzatori, TP

esperienza 3

Gridella Andrea
Benini Andrea
5AET 2013
(gridella.andrea@libero.it)
(andrea.benini1@gmail.com)

Il temporizzatore TP genera un impulso con una durata impostata in precedenza.
Come viene rappresentato questo temporizzatore nel programma TIA:

Fig.36: Blocco di istruzione del timer TP.

I temporizzatori TP, TON e TOF hanno gli stessi parametri di ingresso e di uscita.

Parametro	**Descrizione**	**Tipo di dati**	**Area di memoria**	**Descrizione**
IN	Input	BOOL	I,Q,M,D,L	Ingresso di avvio
PT	Input	TIME	I,Q,M,D,L o costante	Durata dell'impulso.Il valore del parametro PT deve essere positivo
Q	Output	BOOL	I,Q,M,D,L	Uscita dell'impulso
ET	Output	TIME	I,Q,M,D,L	Valore temporale attuale

Il parametro IN avvia e arresta i temporizzatori:
- La commutazione da 0 a 1 del parametro IN avvia i temporizzatori TP, TON e TONR.
- La commutazione da 1 a 0 del parametro IN avvia il temporizzatore TOF.

La seguente tabella illustra le conseguenze delle variazioni del valore dei parametri PT e IN.

Nel nostro caso del temporizzatore TP abbiamo che :

- la variazione di PT non ha alcuna conseguenza durante l'esecuzione del temporizzatore.
- la variazione di IN non ha alcuna conseguenza durante l'esecuzione del temporizzatore.

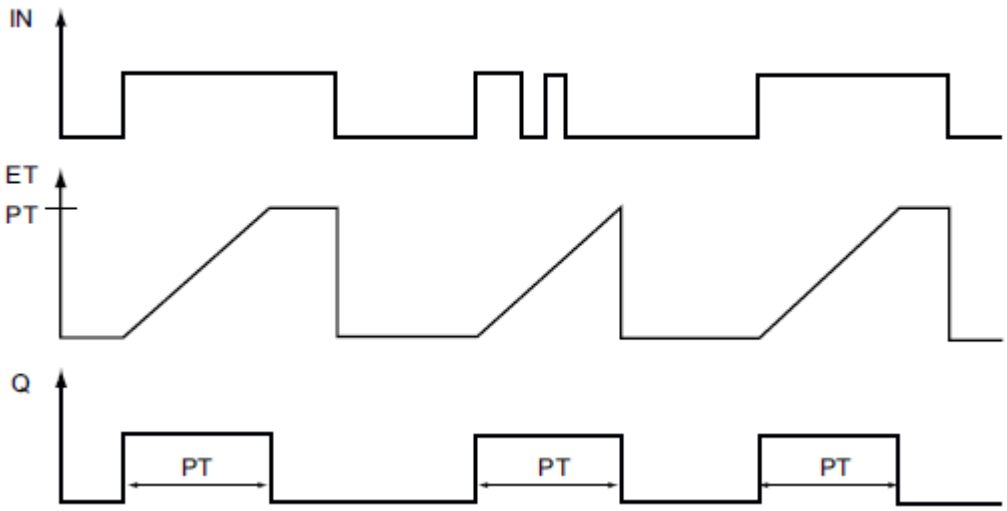


Fig.37: Diagramma del temporizzatore TP come impulso.

Il blocco TP è un temporizzatore che, dopo l'inserzione dell'ingresso, rende attiva l'uscita per un tempo reimpostato. Il blocco TP ha l'ingresso IN al quale colleghiamo uno switch per dare l'impulso iniziale che fa partire il conteggio, mentre all'ingresso PT scriviamo il tempo che ci serve. L'uscita che utilizziamo per controllare il corretto funzionamento del blocco e perciò dell'intero programma è Q, alla quale andiamo a collegare uno dei led a nostra disposizione. Riassumendo lo schema appena descritto è il seguente:

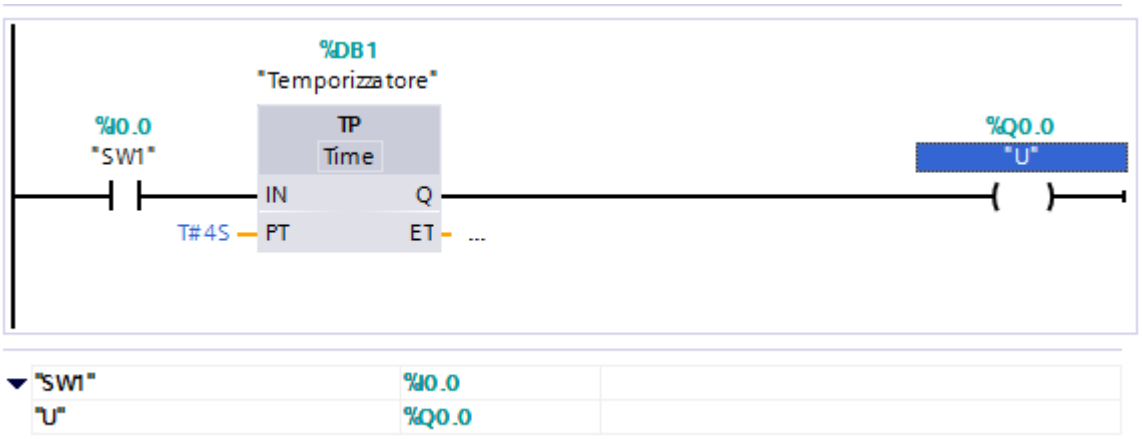


Fig.38: Esempio esercizio con il timer TP.

Questo è un programma molto semplice che realizza l'accensione temporizzata di un led, esempio col quale si comprende facilmente il funzionamento del blocco TP.
L'ingresso e l'uscita li abbiamo rispettivamente rinominati "SW1" e "U", in modo di rendere la comprensione dello schema piu' veloce ed intuitiva. La tabella delle variabili nelle quale sono stati modificati i nomi delle variabili è questa:

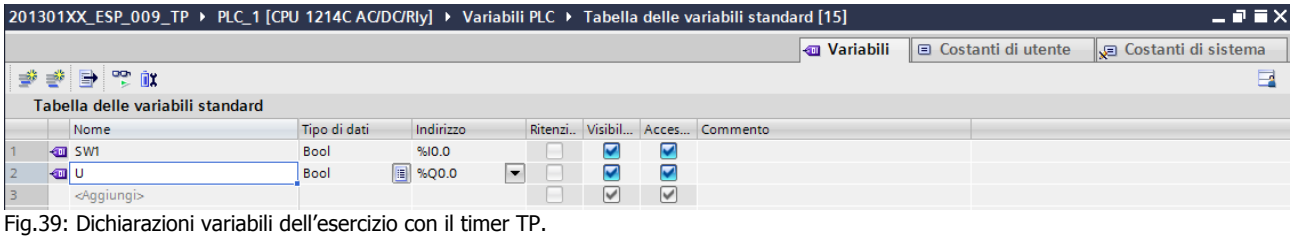


Fig.39: Dichiarazioni variabili dell'esercizio con il timer TP.

Relazione sui temporizzatori, TON

esperienza 4

Farneti Alessandro
Pablo Montemaggi
5AET 2013
(farneti.alessandro@gmail.com)
(pablo.Montemaggi@gmail.com)

Ora introdurremo un temporizzatore TON ovvero che introduce un ritardo con tempo stabilito dall'utente.
Una volta prestabilito il tempo, il timer conta il tempo quando l'ingresso di abilitazione è attivo.

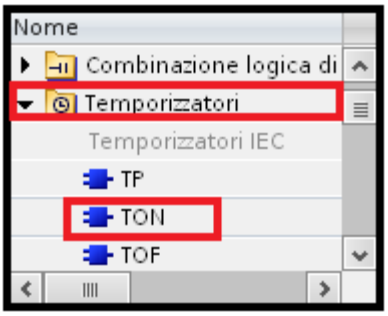


Fig.40: Locazione del timer TON.

Ovviamente va collegato ad un **ingresso** " **A** " normalmente aperto, che corrisponderà ad uno switch fisico e un uscita "**Yton**" che sarà il led.

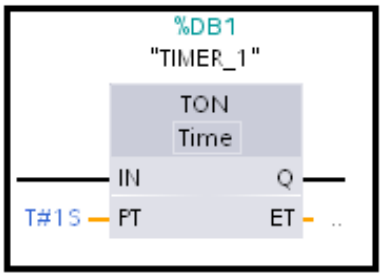


Fig.41: Blocco di istruzione TON.

Il timer ha un ingresso " **IN** " e un uscita " **Q** " il tempo può essere deciso cambiando il valore di " **PT** ".
Gli ingressi che possono allacciarsi con quest'ultimo anche essi fanno parte della tabella delle variabili.

Va collegato in questo modo :

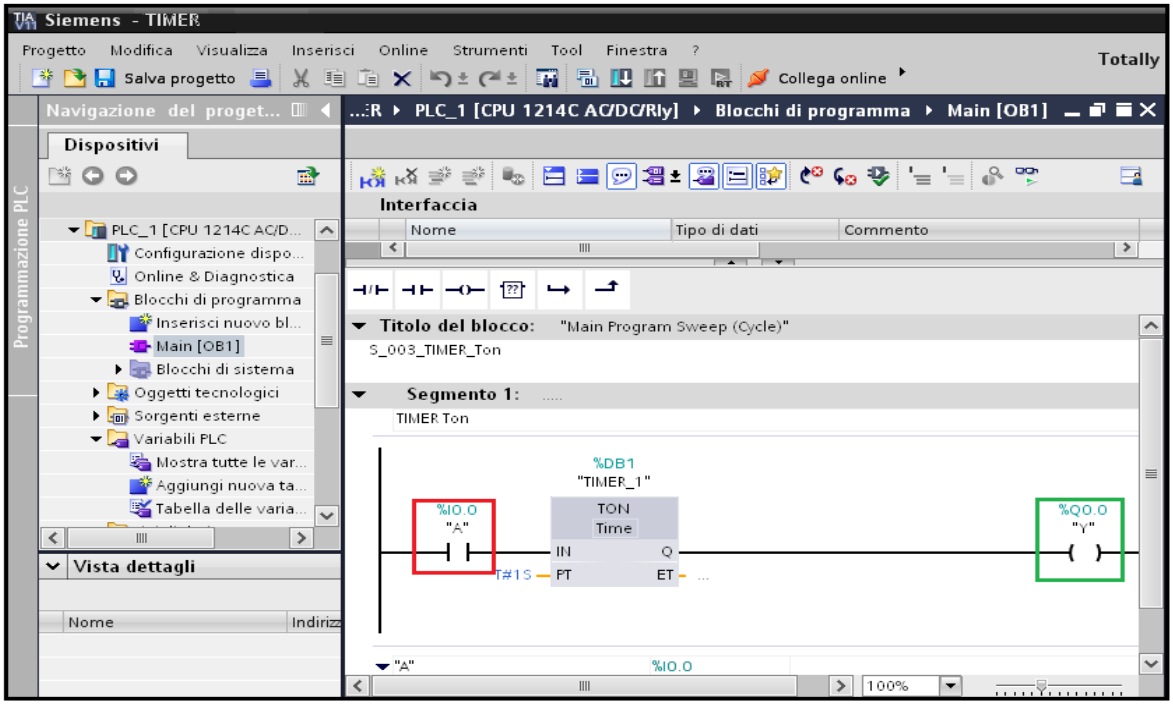


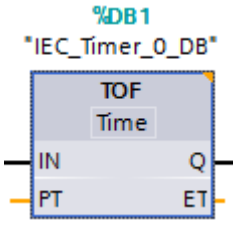
Fig.42: Esempio dell'utilizzo del timer TON in un esercizio.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Relazione sui temporizzatori, TOF

esperienza 5

Farneti Alessandro 5AET 2013
(farneti.alessandro@gmail.com)



Esiste anche un'altra modalità del temporizzatore, oltre a TON : vi è il temporizzatore TOF.

Quest'ultimo consente di ritardare la disattivazione di un uscita per un dato periodo di tempo che l'ingresso è stato disattivato.
Il timer ha la stessa struttura del TON, e anche in questa modalità l'utente può decidere il tempo a piacere modificando il " **PT** " che in questo viene preimpostato di default a 999 ms.

Fig.42: Esempio dell'utilizzo del timer TON in un esercizio.

Esso necessita di un collegamento di uno ingresso negato, per cui si parte con un ingresso " **A** " normalmente aperto e subito dopo ne inseriamo uno " **B** " normalmente chiuso.

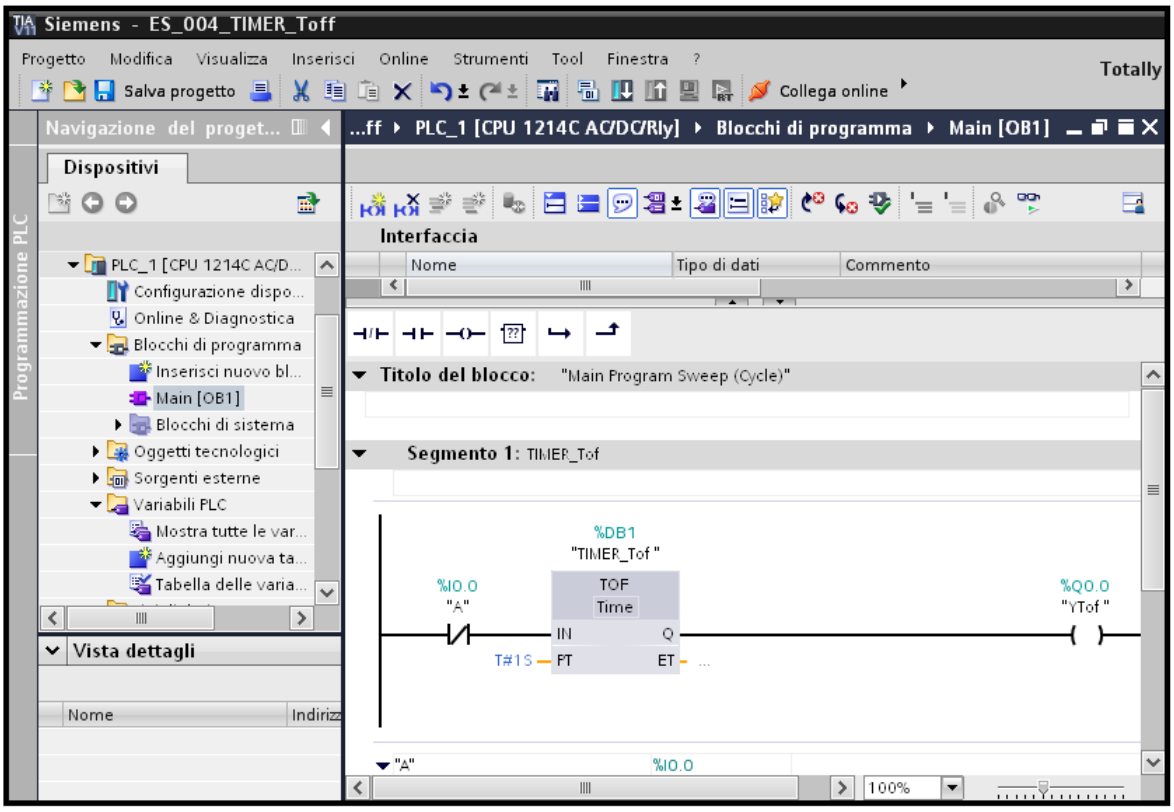


Fig.44: Esempio di utilizzo del timer TOF in un esercizio.

Relazione sui temporizzatori, Timer Ciclico

esperienza 6

Farneti Alessandro 5AET 2013
(farneti.alessandro@gmail.com)

Questa volta impostiamo due volte lo stesso TON in modo da formare un ciclo che accenda e spenga il nostro led ogni 999 ms.

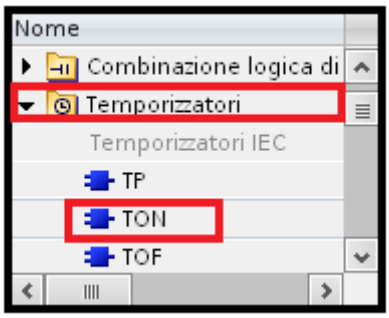
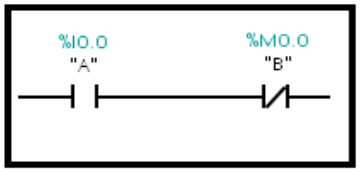


Fig.45: Locazione dei temporizzatori.

Per far ciò selezioniamo il temporizzatore in alto a destra. Una volta selezionato un il temporizzatore ton dobbiamo collegarlo rispettivamente a due ingressi :



Il primo ingresso serve per innescare il ciclo dei timer e il secondo serve per rendere bassa l'uscita.

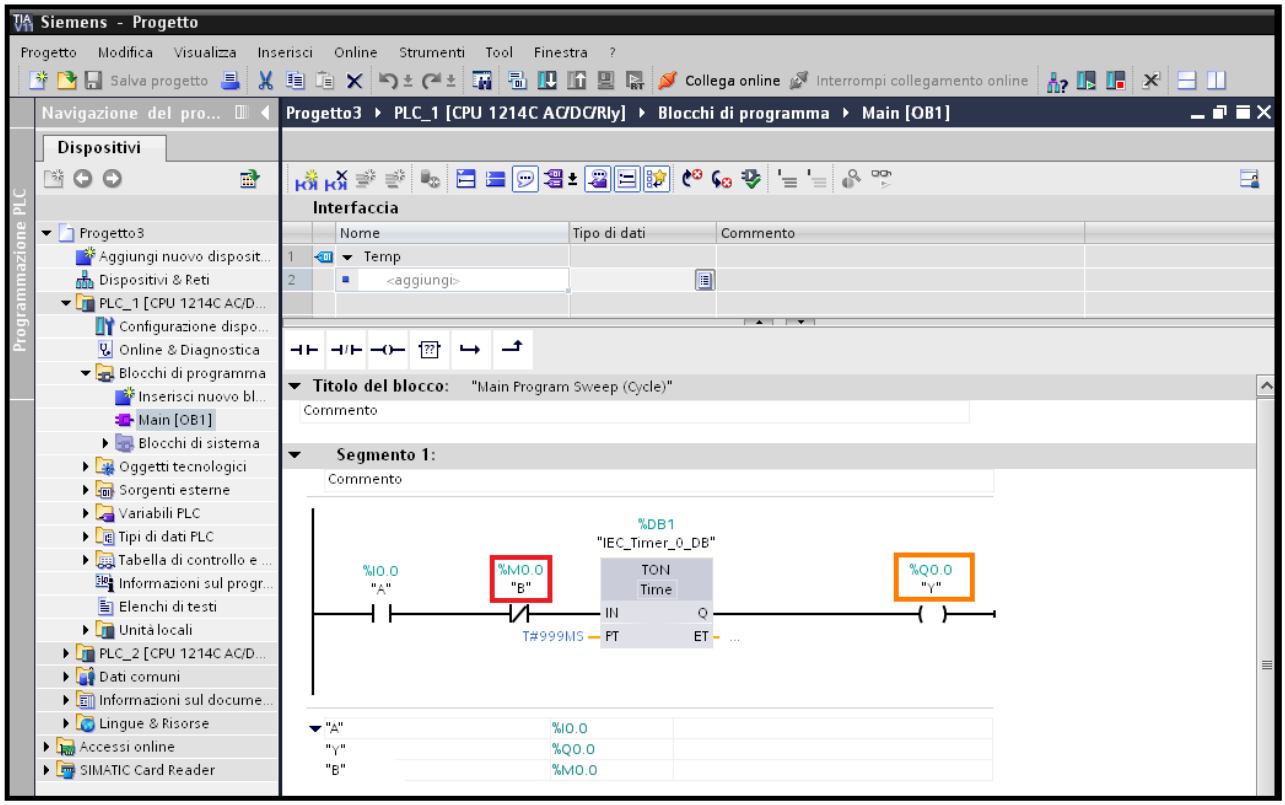


Fig.46: Esempio del Timer Ciclico.

Collegheremo l'uscita " **Y** " alla fine del timer in modo che possiamo portare il dato sul prossimo timer e innescare cosi il ciclo.

Si procede aprendo un altro segmento, con un nuovo ingresso con lo stesso nome dell'uscita e la si collega direttamente al timer.

L'uscita " **Y** " diventa un così un ingresso.
che si collegherà al prossimo timer che scatterà ogni 999 ms in quanto il valore **" PT "** è impostato su 999. T#999MS — PT

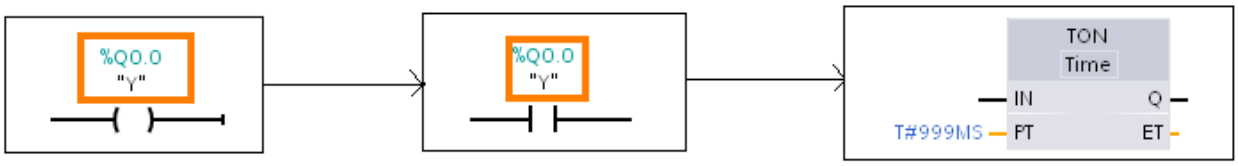


Fig.47: Esempio di utilizzo di un'uscita come ingresso.

L'uscita " **B** " del prossimo timer dovrà comunicare con l'ingresso nel segmento 1, in modo da rendere continuo il ciclo.

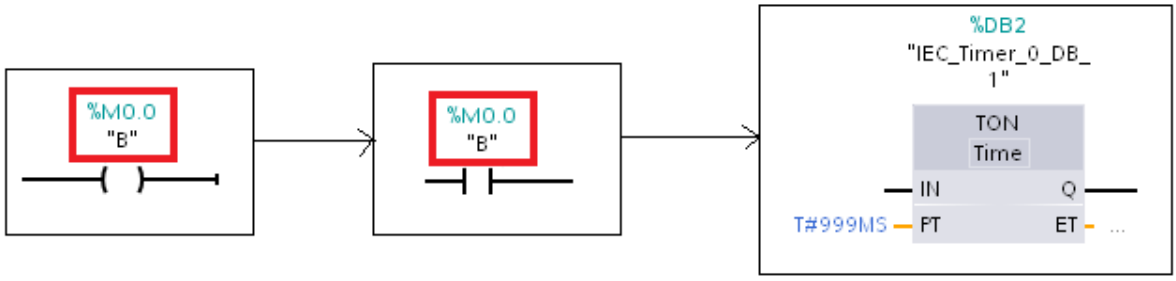


Fig.48: Altro esempio di utilizzo di un'uscita come ingresso.

Ecco come termina il timer ciclico :

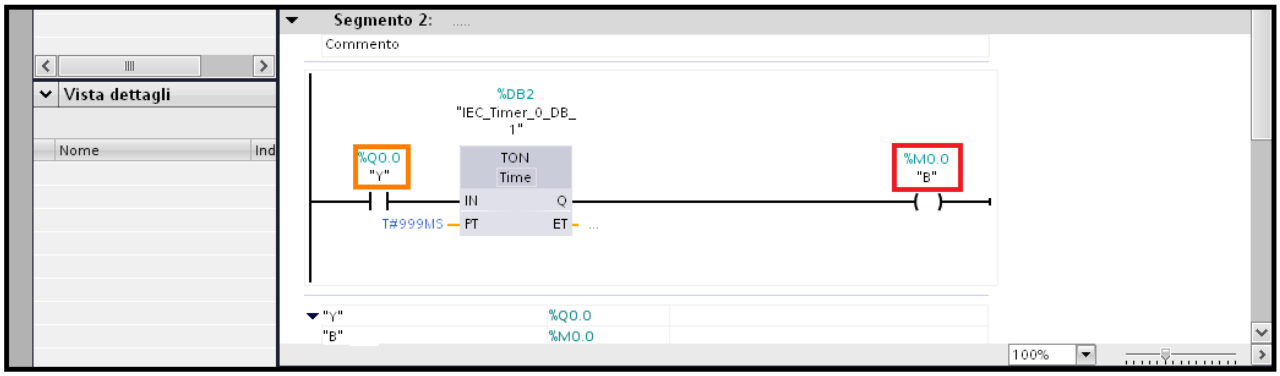


Fig.49: Termine dell'esempio di un Timer Ciclico.

Relazione sui temporizzatori, Counter

esperienza 7

Farneti Alessandro 5AET 2013
(farneti.alessandro@gmail.com)

In questa esperienza utilizziamo il contatore, una cosiddetta " **istruzione di base** "che consente di contare fino a un numero di volte impostato dall'utente ed attivare la luce.
Questa istruzione la si può trovare in altro a destra nella scheda:
istruzioni di base > contatori > CTU
Come è suggerito dalla descrizione, abbiamo un conteggio in avanti e se si clicca sulla freccia arancione CTU: Conteggio in avanti vengono date informazioni su come utilizzare e come collegare quest'ultima istruzione.

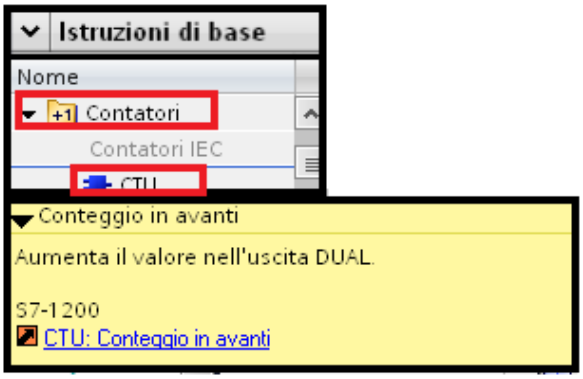
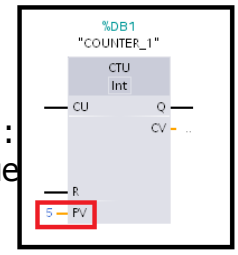


Fig.50: Locazione del contatore CTU.



Una volta selezionato il blocco del counter avremo la seguente figura da posizionare :
Bisogna collocarlo all'interno dello spazio bianco e agganciarlo due switch, ovvero due ingressi. Impostando S PV a 5 il contatore si attiverà dopo 5 ripetizioni.
Uno switch serve per le 5 ripetizioni e accendere la luce, l'altro per spegnere.

Fig.51: Blocco di istruzione del contatore CTU.

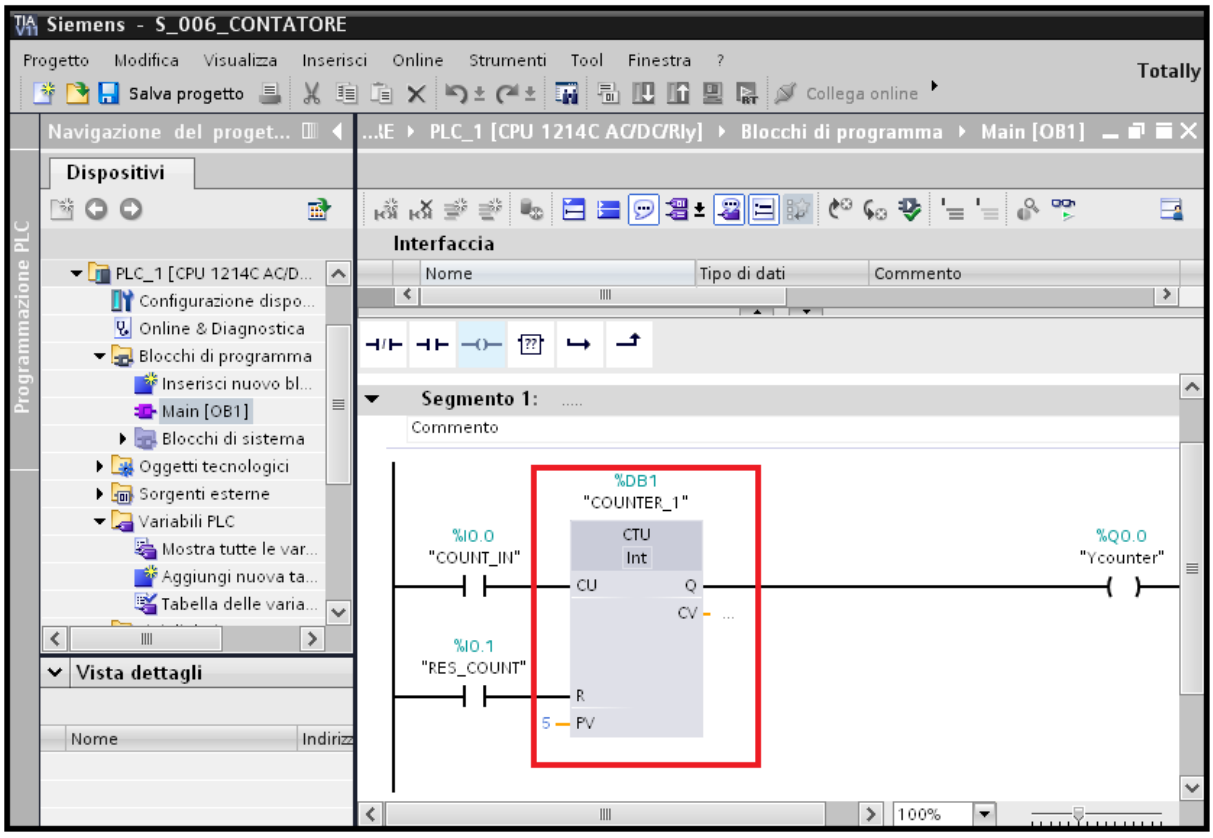


Fig.52: Esempio di utilizzo del contatore CTU.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

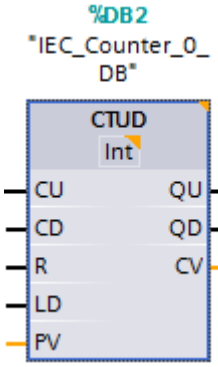
Temporizzatori, Counter CTUD

esperienza 8

Benini Andrea 5AET 2013
(andrea.benini01@gmail.com)

CTUD è un contatore di conteggio in avanti e all'indietro.

Parametri del blocco CTUD:



Parametro	Tipo di dati	Descrizione
CU, CD	Bool	Conta in avanti o indietrpìo di uno
R (CTU, CTUD)	Bool	Resetta a zero il valore di conteggio
LOAD (CTD, CTUD)	Bool	Carica il controllo per il valore preimpostato
PV	SInt, Int, DInt, USInt, UInt,UDInt	Valore di conteggio preimpostato
Q,QU		Vero se CV >= PV
QD		Vero se CV <= 0
CV		Valore di conteggio attuale

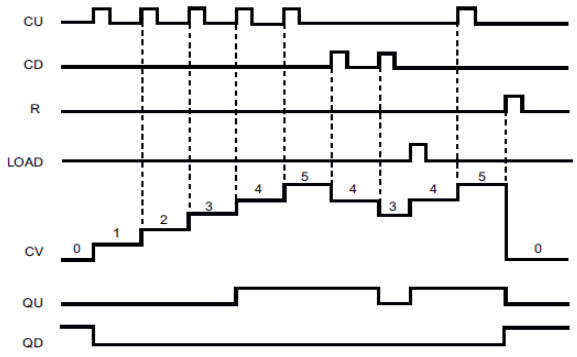


Fig.54: Temporizzazioni inerenti al contatore CTUD.

IL CTUD conta in avanti o indietro di 1 quando gli ingressi di conteggio in avanti (CU) o
all'indietro (CD) passano da 0 a 1.
Se il valore del parametro CV (valore di conteggio attuale) è uguale o maggiore del valore del parametro PV (valore preimpostato) il parametro di uscita del contatore QU = 1. Se il valore del parametro CV è inferiore o uguale a zero il parametro di uscita del contatore QD = 1. Se il valore del parametro LOAD cambia da 0 a 1, il valore del parametro PV (valore preimpostato) viene caricato nel contatore come nuovo CV (valore di conteggio attuale). Se il valore del parametro di reset R cambia da 0 a 1, il valore di conteggio attuale viene resettato a 0. La seguente figura mostra un diagramma di temporizzazione CTUD con un valore di conteggio costituito da un numero intero senza segno (dove PV = 4).

Se il valore del parametro CV (valore di conteggio attuale) è uguale o maggiore del valore del parametro PV (valore preimpostato) il parametro di uscita del contatore QU = 1. Se il valore del parametro CV è inferiore o uguale a zero il parametro di uscita del contatore
QD = 1. Se il valore del parametro LOAD cambia da 0 a 1, il valore del parametro PV (valore preimpostato) viene caricato nel contatore come nuovo CV (valore di conteggio attuale).
Se il valore del parametro di reset R cambia da 0 a 1, il valore di conteggio attuale viene resettato a 0.

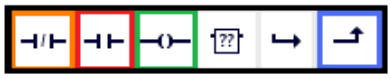
Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Marcia-Arresto di un Motore

esperienza 9

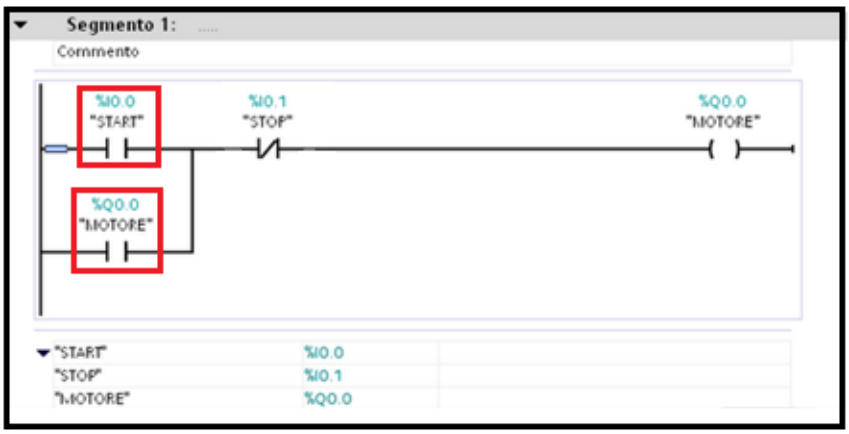
Farneti Alessandro 5AET 2013
(farneti.alessandro@gmail.com)

Si andrà a realizzare una programma che ci consentirà di fare marcia e arresto di motore. Per realizzare ciò, dovremmo disporre di 3 contatti, che vanno collegati nell'ordine che è nell'immagine qui sotto.



Andremo selezionare dalla "**barra degli strumenti**" il contatto aperto selezionato in rosso : 

Un ingresso "**I0.0**" verrà denominato "**Start**" come ingresso che attiverà il motore mentre l'altro ingresso "**Q0.0**" collegato parallelamente, verrà chiamato "**Motore**".



Mettiamo il primo contatto "**START**" nella prima fila mentre "**MOTORE**" va posto al di sotto. Nel momento in cui dovremmo collegare i due contatti si utilizza la "**freccia blu**" : 
In seguito andrà collegato un contatto chiuso.

Fig.55: Prima parte del programma Marcia-Arresto motore.

Chiameremo il contatto chiuso "**STOP**" con indirizzo "**I0.1**". Quest'ultimo è identificato con l'arancione nella barra degli strumenti :

Contatto chiuso :

Uscita "MOTORE" :

Fig.56: Seconda parte del programma Marcia-Arresto motore.

Ovviamente tutte le variabili andranno salvate nella "**Tabella Variabili**" :

Fig.57: Dichiarazione variabili dell'esercizio Marcia-Arresto motore.

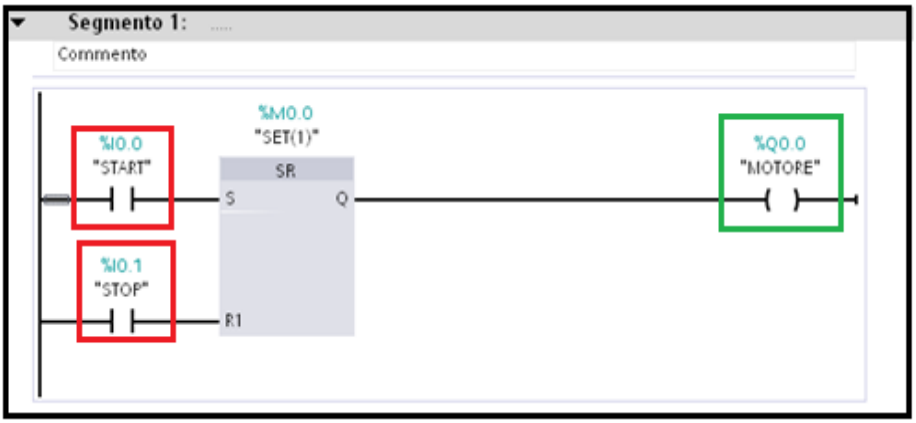
Marcia-Arresto di un Motore con Flip Flop

esperienza 10

Farneti Alessandro 5AET 2013
(farneti.alessandro@gmail.com)

Questa esperienza consiste nell'applicare una variante all'esercizio precedente, applicando al blocco di un "**flip flop di tipo SR**" al blocco del motore marcia-arresto.

Consiste in un blocco set/reset al quale è collegato un interruttore all'ingresso di SET ed uno a quello di RESET, in modo che con una semplice pressione di uno dei rispettivi interruttori si abbia la marcia e l'arresto del motore.



Questo schema si realizza attaccando **due contatti normalmente aperti** (evidenziati in rosso) ad un flip flop SR. Rispettivamente chiamate con "**I0.0**", "**START**" e con "**I0.1**", "**STOP**".

Fig.58: Prima parte dell'esercizio Marcia-Arresto motore con Flip-Flop.

In seguito si inserisce il blocco flip flop "**M0.0**" in questo modo. Il blocco si deve selezionare nella tendina in alto a destra nella quale sono posti i vari tipi di flip flop.

Nell'uscita Q con indirizzo "**Q0.0**"(evidenziata in verde), è collegato un led con il quale vediamo la corretta funzionalità del programma.

Le variabili sono state nominate in questo modo:

Variabili PLC

	Nome	Tabella delle varia...	Tipo di dati	Indirizzo	Ritenzi...	Visibil...	Acces...
1	START	Tabella delle va...	Bool	%I0.0		☑	☑
2	STOP	Tabella delle varia...	Bool	%I0.1		☑	☑
3	MOTORE	Tabella delle varia...	Bool	%Q0.0		☑	☑
4	SET(1)	Tabella delle varia...	Bool	%M0.0		☑	☑
5	<Aggiungi>					☑	☑

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Marcia Avanti-Indietro di un Motore

esperienza 11

Farneti Alessandro 5AET 2013
(farneti.alessandro@gmail.com)

In questa esperienza andremo a realizzare un motore marcia avanti/indietro di un motore con eventuale arresto. Il cambio di marcia può però solamente avvenire quando il motore è fermo.

Il programma è composto da **due segmenti** e, nel primo, mettiamo l'interruttore " START " in serie a un altro chiamato " MARCIA_AVANTI " ed entrambi vengono messi in parallelo ad un interruttore aperto chiamato " AVANTI ".

In seguito questo blocco viene posto in serie ad un interruttore normalmente chiuso chiamato " **STOP** ", ad un altro normalmente chiuso denominato " INDIETRO " e in serie all'uscita " AVANTI ".

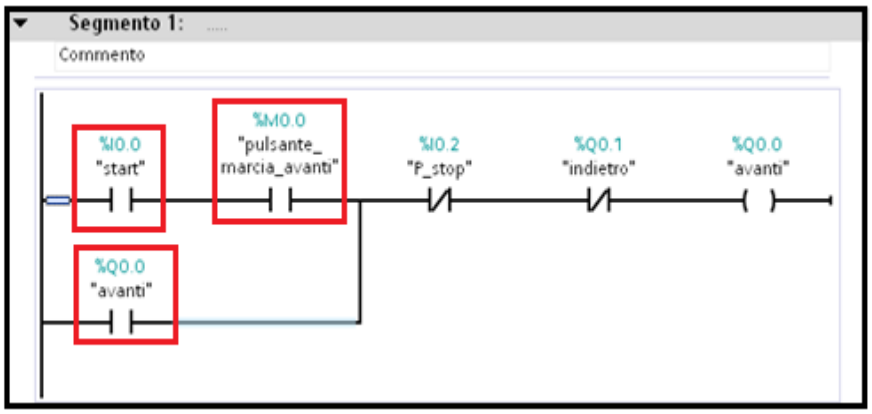


Fig.59: Seconda tipologia marcia-arresto.

Si parte mettendo **3 contatti normalmente aperti** collegati fra loro :

- con indirizzo "**I0.0**" abbiamo "**START**"
- con indirizzo "**Q0.0**" abbiamo "**AVANTI**"
- con indirizzo "**M0.0**" abbiamo "**Pulsante marcia avanti**"

Per collegare i tre contatti menzionati sopra, utilizziamo la "**freccia blu**" :
I contatti che seguono sono contatti normalmente chiusi.

Per selezionare i due sensi di marcia facciamo altri due segmenti con i quali selezioniamo cambiamo il senso. Questo passo necessita di un "**pulsante STOP**" e un "**selettore**" che andranno collegati all'uscita, dove abbiamo il "**pulsante marcia avanti**".

In questo caso mettiamo anche qua due contatti di cui uno è normalmente aperto mentre l'altro è il contrario.
- indirizzo "**10.2**" indica il pulsante stop
- indirizzo **"I0.3"** indica un preselettore.

Precedentemente verrà fatto un altro riquadro che consente di collegare due contatti normalmente chiusi il "**pulsante di marcia indietro**".

All'interno vengono collegati il "**P_STOP"** e il "**Selettore.**" L'uscita rimane il pulsante di marcia indietro.

Fig.60: Parte finale Marcia_arresto

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Contatori Veloci

esperienza 12

Carloni Lorenzo 5AET 2013
(lorenzo.carloni1@gmail.com)

Prima di iniziare dobbiamo settare alcune opzioni del PLC per poter leggere gli impulsi del nostro encoder con i contatori veloci interni al PLC.

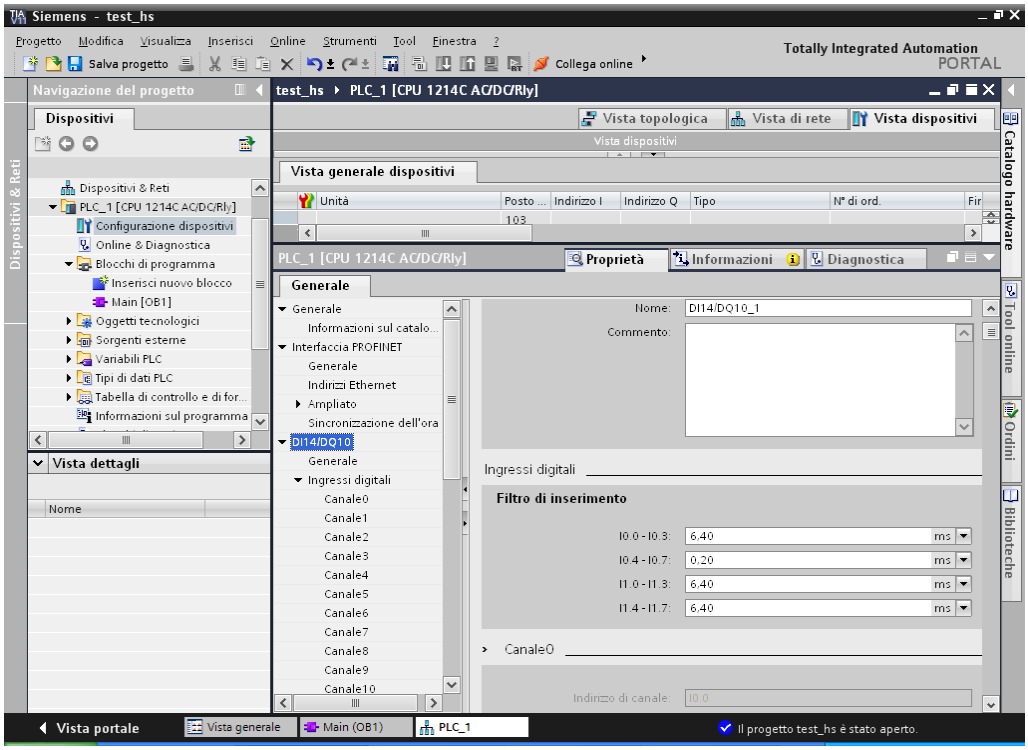


Fig.61: Finestra nella quale settare alcune opzioni per lavorare con i contatori veloci.

Per prima cosa impostiamo il filtro sul canale 4 al minimo per evitare che il nostro impulso venga filtrato come un disturbo.

Ora impostiamo il canale 4 per poter leggere gli impulsi.

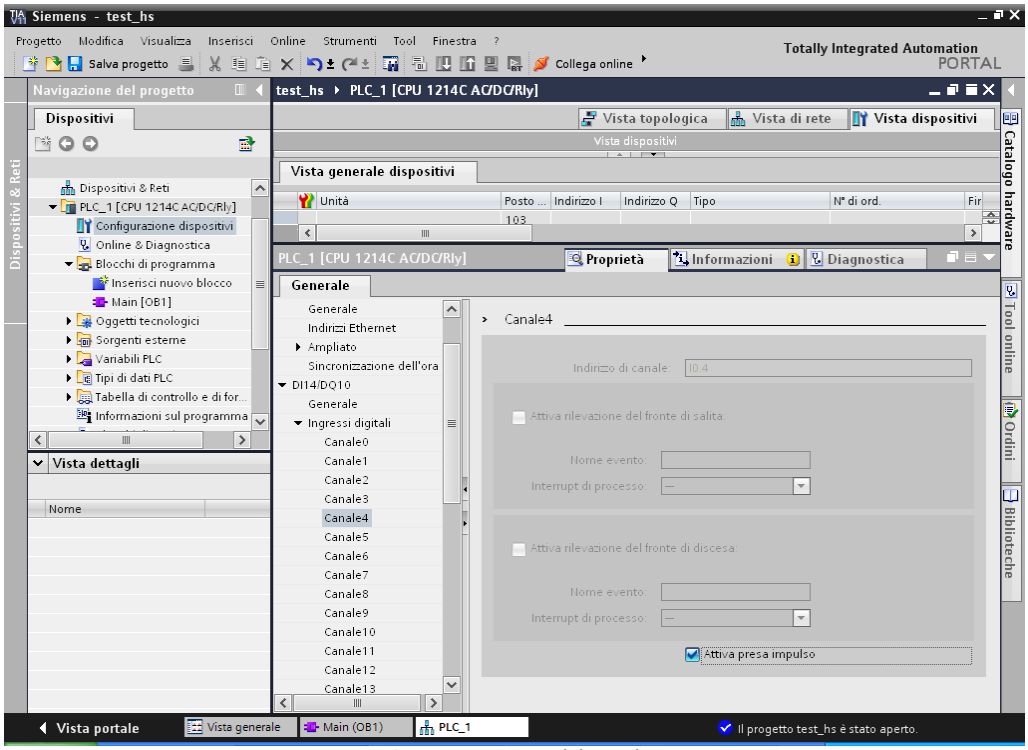


Fig.64: impostazione del canale 4.

Ora attiviamo l'HSC3 e lo impostiamo per il conteggio monofase con reset da programma.

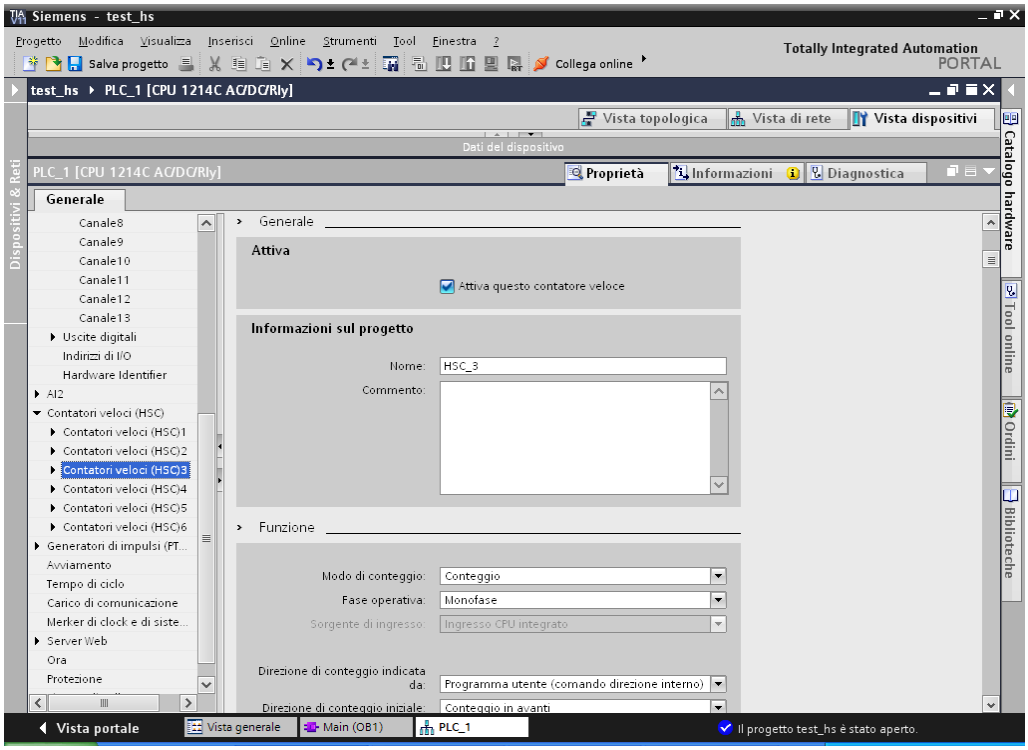


Fig.65: Attivazione e impostazione dell'HSC3 per il conteggio monofase.

Impostiamo i valori iniziale e di riferimento a zero in quanto non utilizzeremo il riferimento di conteggio dell'HSC che crea un interrupt ma andremo a leggere il conteggio all'interno del programma.

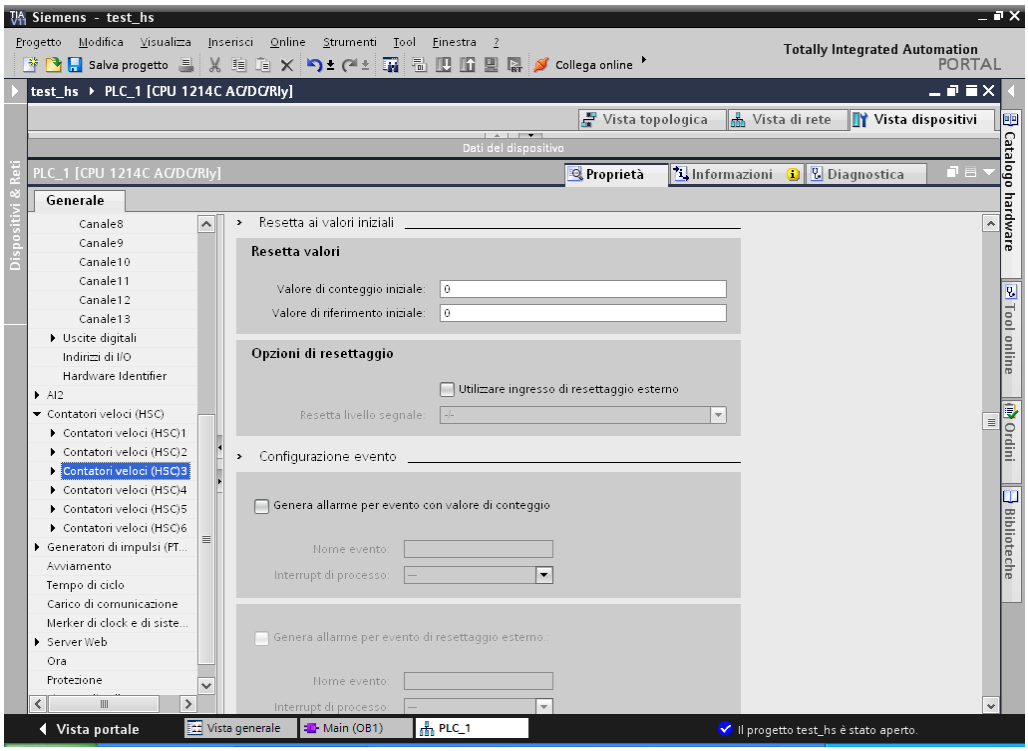


Fig.66: Impostazione dei valori iniziali e di riferimento a 0.

Qui possiamo leggere l'indirizzo del nostro ingresso (I0.4) l'indirizzo di memoria dove verrà salvato il conteggio (1008) e l'identificatore dell'HSC (260).

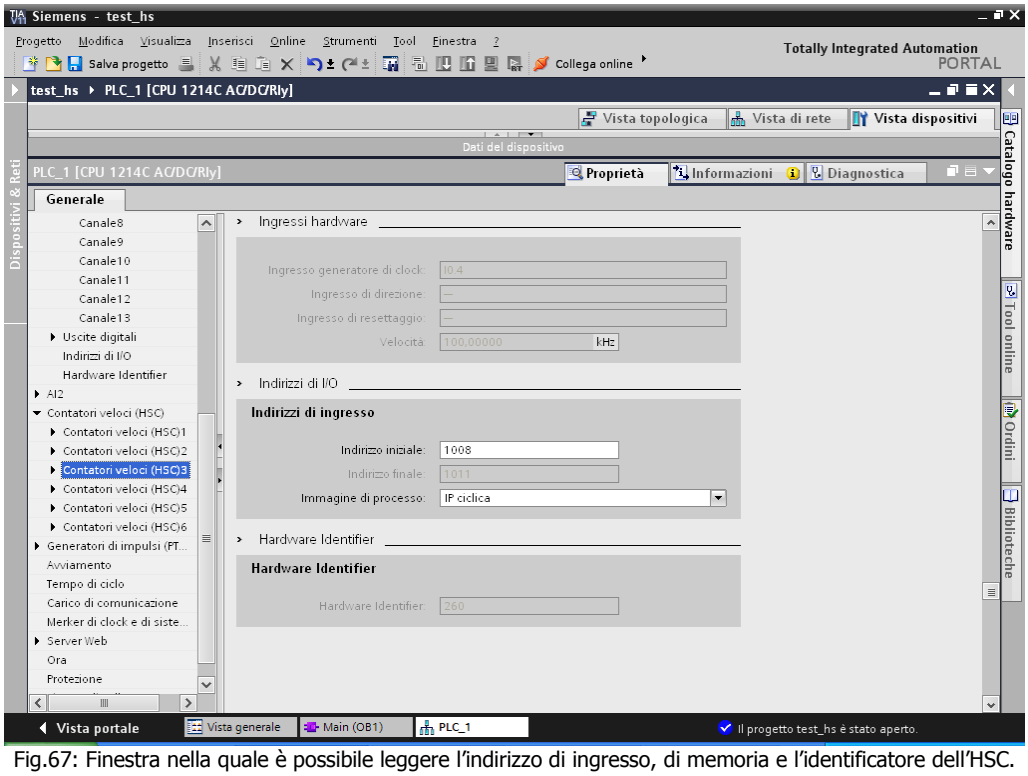


Fig.67: Finestra nella quale è possibile leggere l'indirizzo di ingresso, di memoria e l'identificatore dell'HSC.

N.B. il canale impostato (4) è relativo solamente all'HSC3.

Per poter usare altri contatori veloci dobbiamo applicare lo stesso procedimento ma al canale relativo all'ingresso dell'HSC

Attiviamo i vari merker di sistema che ci potranno tornare utili.

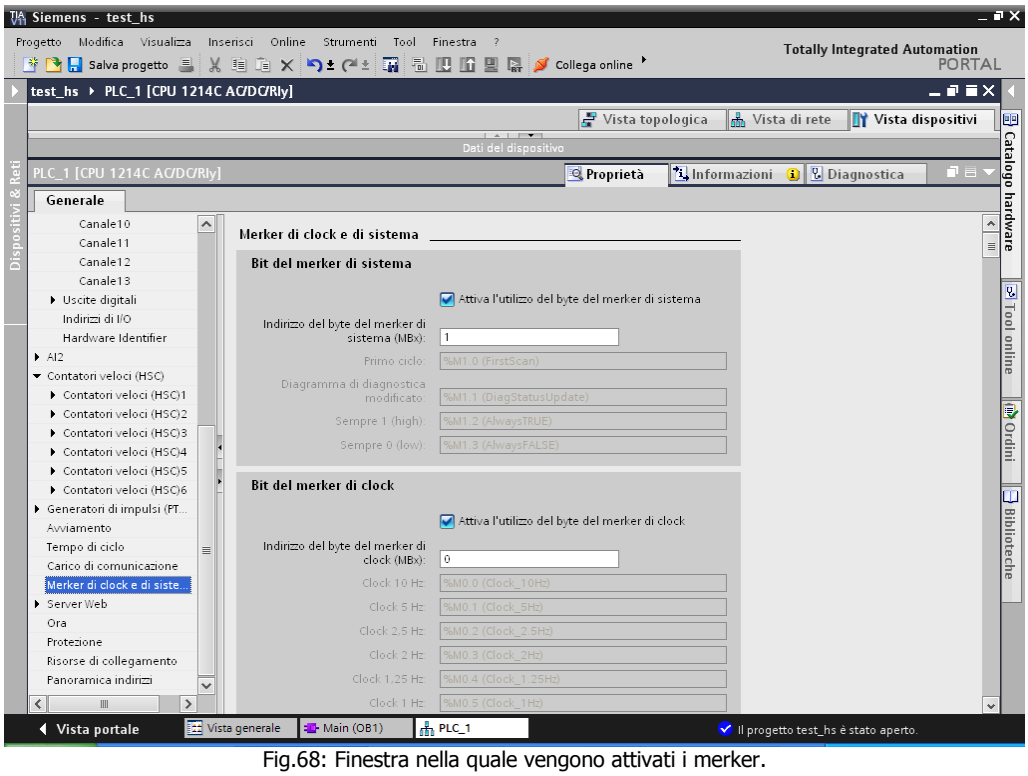


Fig.68: Finestra nella quale vengono attivati i merker.

Mettiamo un blocco di conversione per spostare il nostro dato del conteggio in una nostra variabile.

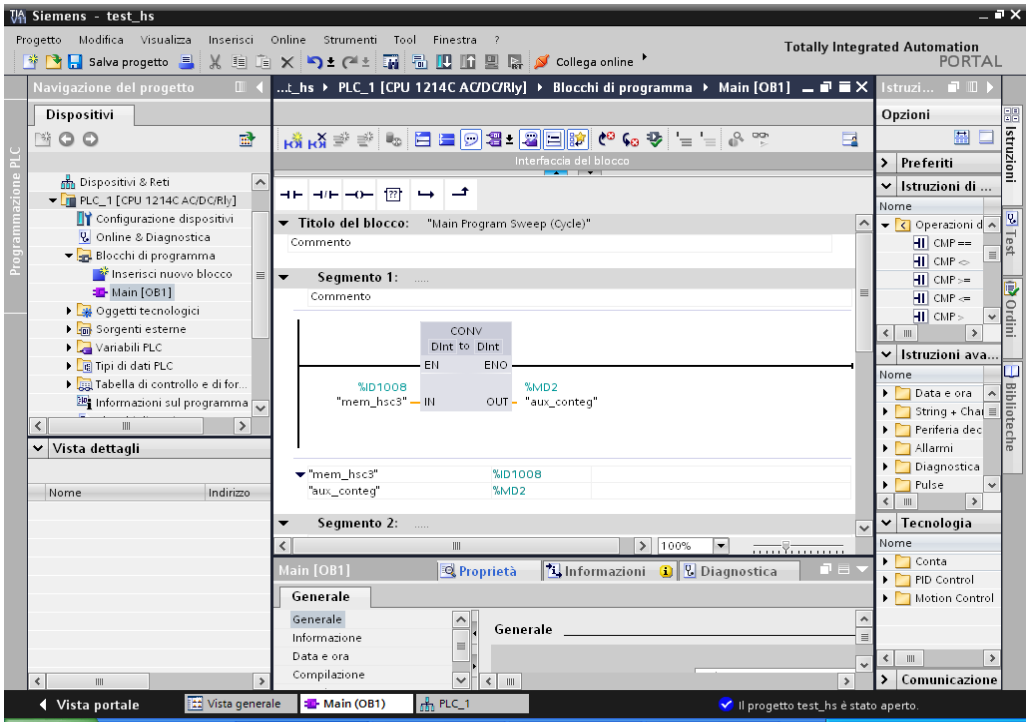


Fig.69: Blocco di conversione utilizzato nel programma.

Poi andiamo a confrontare la nostra variabile con un valore prefissato, nel caso venga raggiunto si accenderà l'uscita.
Utilizzo il merker **AlwaysOn** per mantenere il controllo sempre attivo.

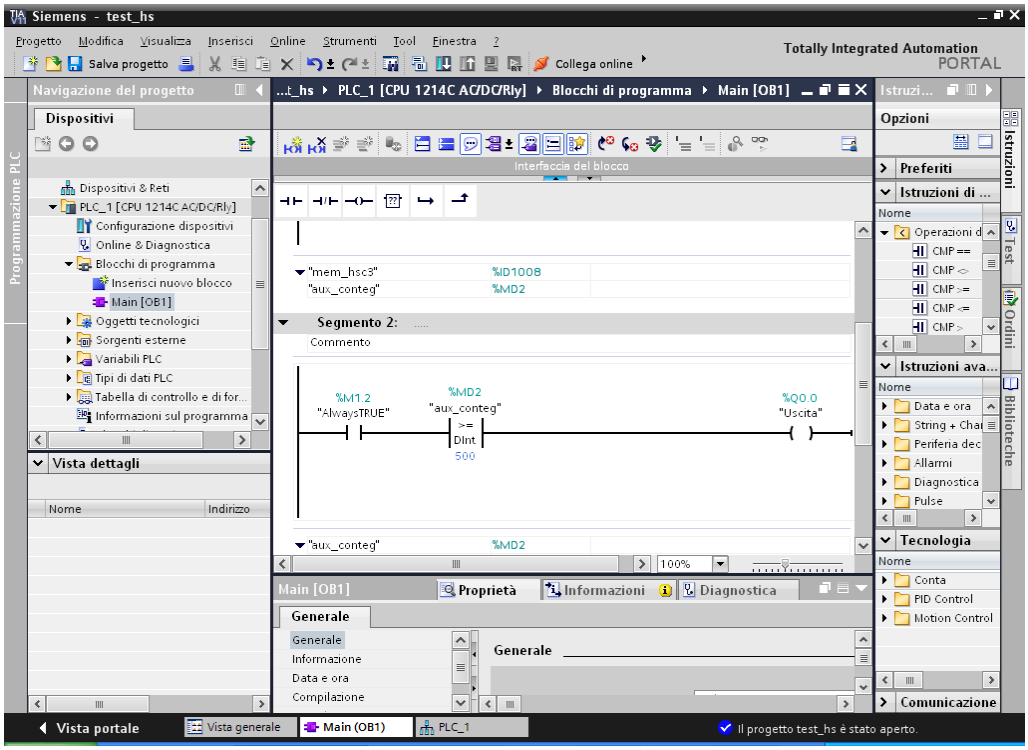


Fig.70: Parte di confronto del programma.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Blocco Ctrl Hsc

esperienza 13

Farneti Alessandro
Benini Andrea
5AET 2013
(farneti.alessandro@gmail.com)
(andrea.benini01@gmail.com)

Questa istruzione comanda i contatori veloci utilizzati per contare gli eventi che si verificano più rapidamente della velocità di esecuzione dell'OB. La velocità di conteggio delleistruzioni CTU, CTD e CTUD è limitata dalla velocità di esecuzione dell'OB in cui sono state in serite. Per esempio un uso tipico dei contatori veloci di questo tipo è il conteggio degli impulsi generati da un encoder per il comando della rotazione di un albero motore. Per salvare i propri dati ciascuna istruzione CTRL_HSC utilizza una struttura memorizzata in un blocco dati che viene assegnato quando si inserisce l'istruzione nell'editor.

Parametri del CTRL_HSC

Parametro	**Tipo di parametri**	**Tipo di dati**	**Descrizione**
HSC	IN	HW_HSC	Identificatore dell'HSC
DIR	IN	Bool	1=richiesta della nuova direzione
CV	IN	Bool	1=richiesta di impostare il nuovo valore di conteggio
RV	IN	Bool	1=richiesta di impostare il nuovo valore di riferimento
PERIOD	IN	Bool	1=richiesta di impostare il nuovo valore di periodo(solo per la modalità di misura della frequenza)
NEW_DIR	IN	Int	Nuova direzione: 1=in avanti -1=indietro
NEW_CV	IN	Dint	Nuovo valore di conteggio
NEW_RV	IN	DInt	Nuovo valoredi riferimento
NEW_PERIOD	IN	Int	Nuovo valore di periodo in secondi: .01,.1 o 1 (solo per la modalità di misura della frequenza)
BUSY	OUT	Bool	Funzione occupata
STATUS	OUT	WORD	Codice della condizione di occupazione

Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

I primi ingressi sulla sinistra sono le abilitazione per l'inserimento del nuovo valore corrente, del valore di riferimento, della direzione e periodo.
Gli ingressi con **NEW** invece sono i valori che verranno letti e inseriti una volta abilitati.
Sulla destra abbiamo **STATUS** che ci fornisce l'identificativo dell'errore se sopraggiunge e **BUSY** che ci indica se è occupato o meno. **EN0** è una propagazione dell'abilitazione.

Per poter utilizzare i contatori veloci nel programma è innanzitutto necessario configurarli in Configurazione dispositivi nelle impostazioni del progetto. Le impostazioni per la configurazione degli HSC consentono di selezionare i modi di conteggio, i collegamenti di I/O, l'assegnazione degli allarmi e il funzionamento come contatore veloce o come dispositivo di misura della frequenza degli impulsi.

esperienza 13
Carloni Lorenzo
Brighi Luca
5AET 2013
(lorenzo.carloni1@gmail.com)
(brighi.luca@gmail.com)

ITIS "B. Pascal" Cesena – 5AET2013

PROGETTO AUTOMAZIONE

Nastro trasportatore con stazioni di lavorazione

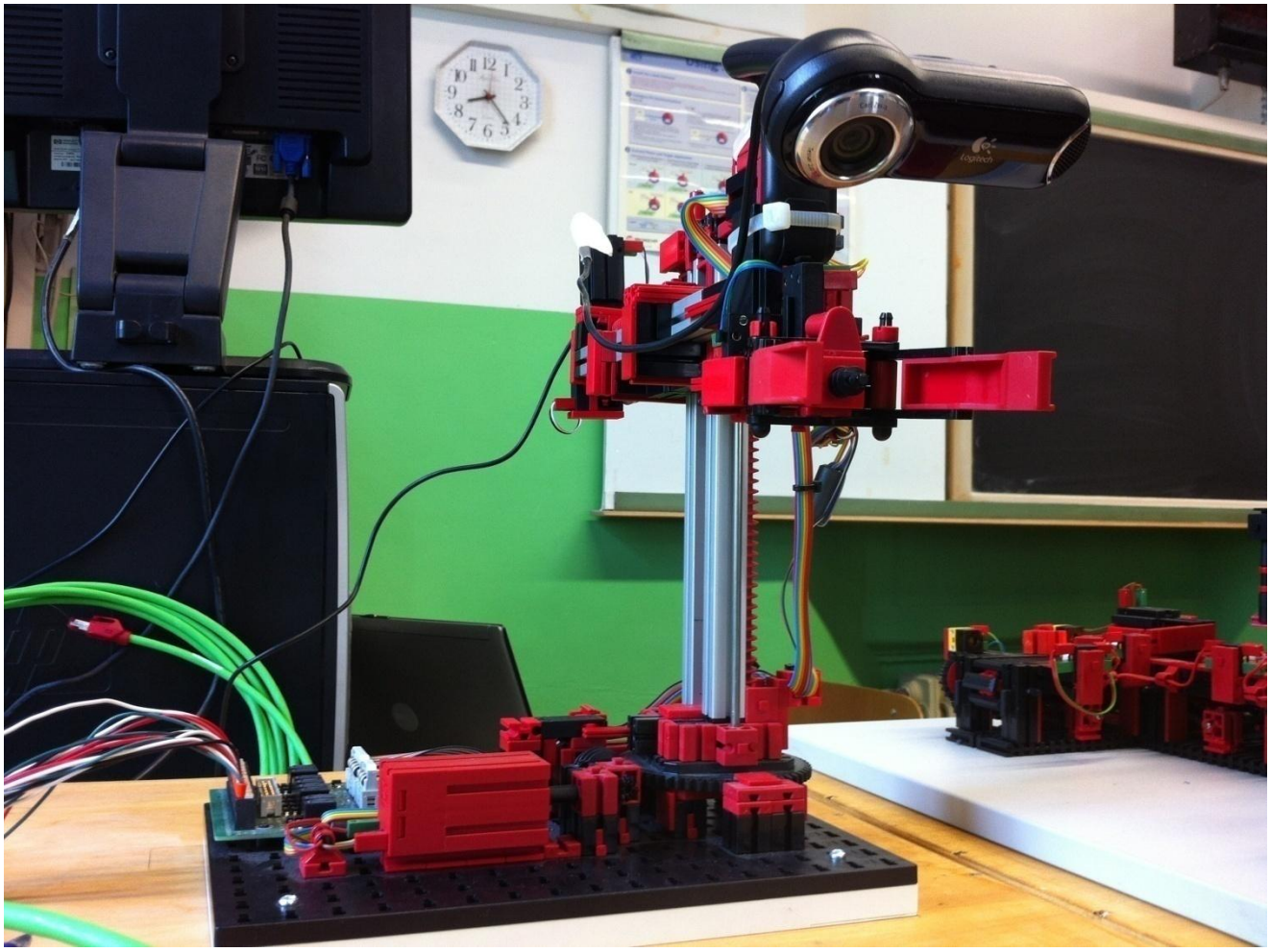


Fig.71: Nastro trasportatore.

Il modellino è un braccio robotico con 4 gradi di libertà, in quanto può muoversi verticalmente, orizzontalmente, ruotare e aprire e chiudere la pinza.
I motori per i movimenti verticali e rotativi sono associati ciascuno ad un encoder in quadratura di fase AB, i quali permettono di controllare la posizione del braccio.
Gli altri due motori sono invece dotati di un contatto che viene premuto durante la rotazione creando una serie di impulsi.
Entrambi i metodi di conteggio hanno la possibilità di essere resettati tramite un fine corsa posto alla fine di uno dei due spostamenti possibili di ogni motore, il quale indica lo zero dello spostamento: in alto per il verticale, senso orario per la rotazione, indietro per l'orizzontale e apertura per la pinza.
Il modellino necessita 24V di alimentazione per gli attuatori e per i sensori.
Ogni motore è bidirezionale e viene comandato da due contatti, uno per una direzione e uno per l'altra.
Dando il segnale logico "1" cioè i 24V viene attivata la direzione selezionata.
Abbiamo aggiunto una webcam collegabile via USB ad un PC per poter realizzare riprese in primo piano del movimento.

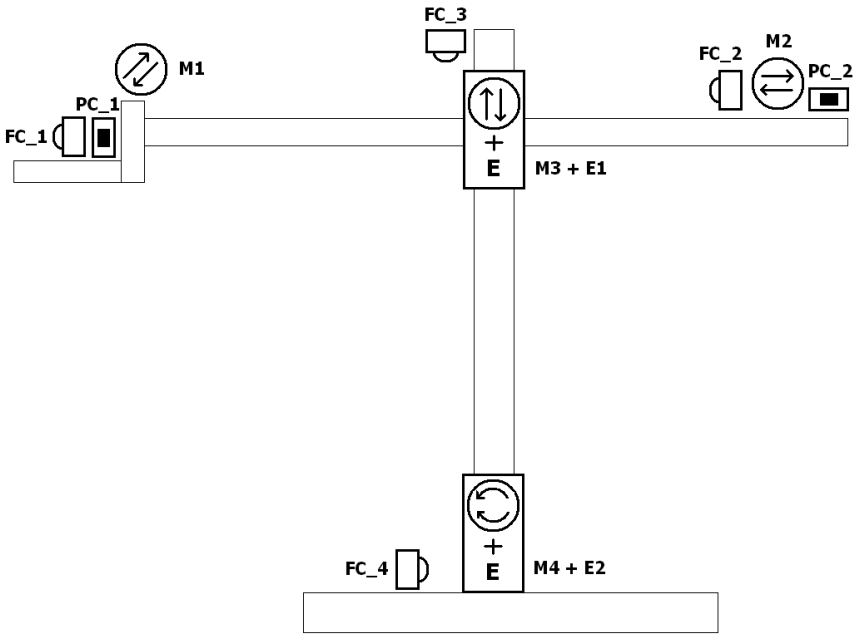


Fig.72: Descrizione modellino.

M1: Motore apertura/chiusura pinza
M2: Motore avanzamento/arretramento orizzontale
M3: Motore ascesa/discesa verticale
M4: Motore rotazione oraria/antioraria

PC_1: Contare di impulsi apertura/chiusura pinze
PC_2: Contatore impulsi avanzamento/arretramento orizzontale

FC_1: Finecorsa apertura pinza
FC_2: Finecorsa arretramento orizzontale
FC_3: Finecorsa ascesa verticale
FC_4: Finecorsa rotazione oraria

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

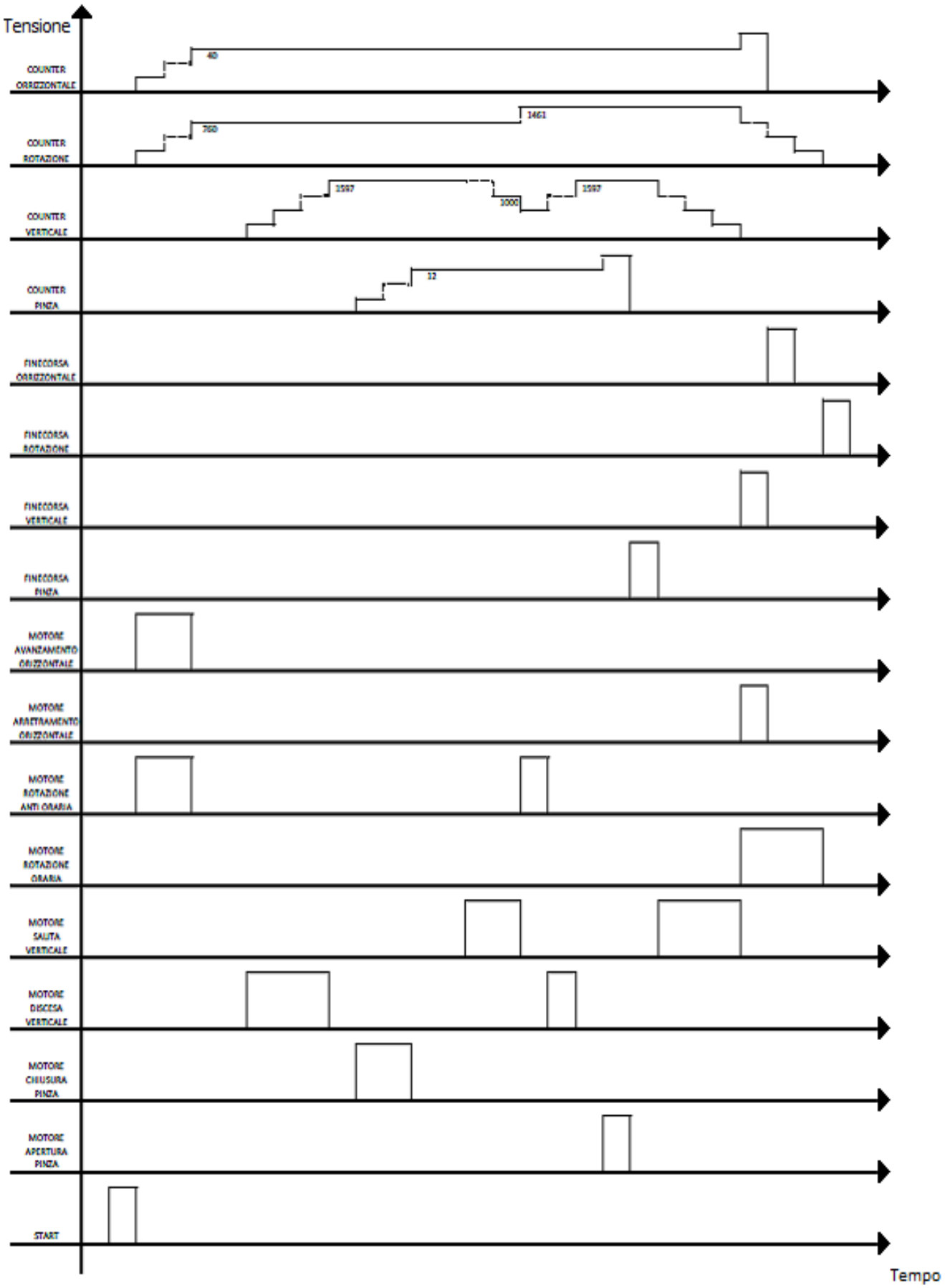


Fig.73: Temporizzazione delle varie parti del sistema.

MOTORI

M1: Motore che aziona la pinza del braccio, ad esso sono associati due contatti, uno per l'apertura (m_pinza_open – Q0.0) e uno per la chiusura della pinza (m_pinza_close – Q0.1).
Lo spostamento viene controllato tramite un contatto che genera degli impulsi, tramite il conteggio di essi possiamo conoscere la sua posizione.

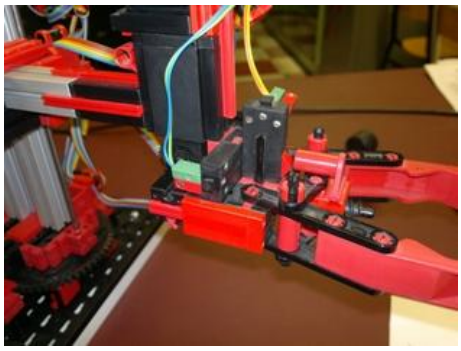


Fig.74: Motore M1.

M2: Motore che aziona il movimento avanti/indietro del braccio, ad esso sono associati due contatti, uno per l'avanzamento (m_orriz_forward – Q0.2) e uno per l'arretramento (m_orriz_back – Q0.3).
Lo spostamento viene controllato tramite un contatto che genera degli impulsi, tramite il conteggio di essi possiamo conoscere la sua posizione.

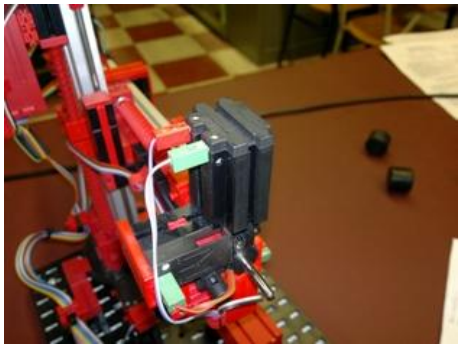
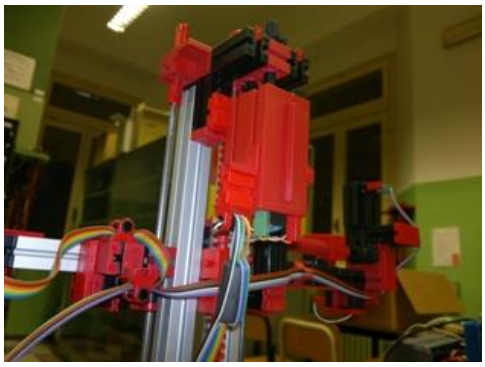


Fig.75: Motore M2.

MOTORI + ENCODER

M3: Motore che aziona il movimento su/giù del braccio, ad esso sono associati due contatti, uno per la discesa (m_vert_down – Q0.4) e uno per la salita (m_vert_up – Q0.5).
Ad esso è associato un encoder in quadratura di fase (AB).



E1: Encoder associato ad M3. Lo gestiamo attraverso il contatore veloce HSC1. Il reset avviene sul FC3 al termine di ogni esecuzione.
Grazie all'algoritmo del contatore possiamo cosi conoscere la posizione del nostro motore, in quanto il valore del conteggio verrà decrementato in salita e incrementato in discesa.

Fig.76: Motore ed encoder.

M4: Motore che aziona il movimento di rotazione del braccio, ad esso sono associati due contatti, uno per la rotazione oraria (m_turn_clock – Q0.6) e uno per la rotazione anti oraria (m_turn_ctrlclock – Q0.7).
Ad esso è associato un encoder in quadratura di fase (AB).

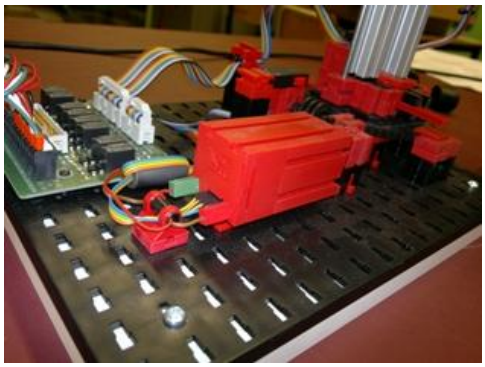
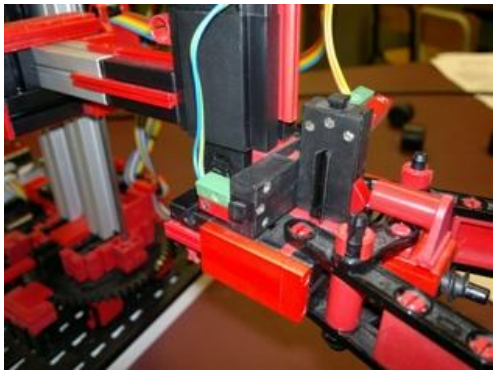


Fig.77: Motore M4.

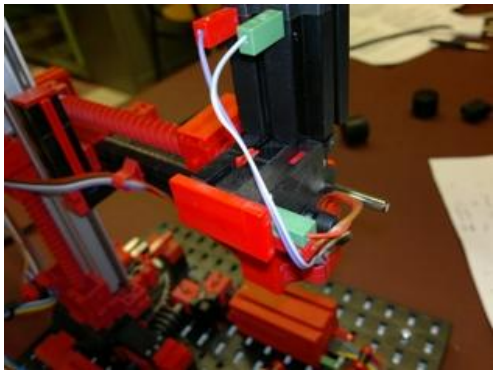
E2: Encoder associato ad M4. Lo gestiamo attraverso il contatore veloce HSC3. Il reset avviene sul FC3, in rotazione oraria, al termine di ogni esecuzione.
Grazie all'algoritmo del contatore possiamo cosi conoscere la posizione del nostro motore, in quanto il valore del conteggio verrà decrementato in senso orario e incrementato in senso anti orario.

CONTATORI AD IMPULSI



PC_1: Contatore di impulsi di M1, ad esso è associato il contatto "pulse_count_pinza" (I0.6), esso viene resettato tramite FC1 e ci permette di determinare di quanto la pinza si è chiusa per poter afferrare il nostro pezzo.

Fig.78: Contatore di impulsi M1.



PC_2: Contatore di impulsi di M2, ad esso è associato il contatto "pulse_count_orriz" (I1.1), esso viene resettato tramite FC2 e ci permette di determinare di quanto il braccio è avanzato orizzontalmente, per poterci portare esattamente sopra al pezzo.

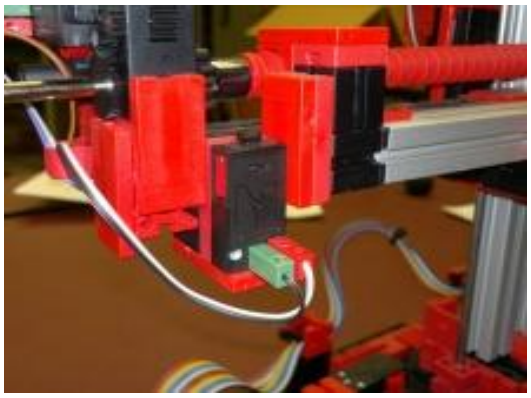
Fig.79: Contatore di impulsi di M2.

Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

FINECORSA

FC_1: Finecorsa di M1, serve per fermare M1 in apertura e resettare il contatore associato a questo movimento, ad esso è associato il contatto "fc_pinza" (I0.2).

Fig.80: Finecorsa di M1.



FC_2: Finecorsa di M2, serve per fermare M2 in arretramento e resettare il contatore associato a questo movimento, ad esso è associato il contatto "fc_orriz" (I1.0).

Fig.81: Finecorsa di M2.

FC_3: Finecorsa di M3 serve per fermare M3, in salita e resettare HSC1, ad esso è associato il contatto "fc_vert/reset" (I0.3).

Fig.82: Finecorsa di M3.

FC_4: Finecorsa di M4, serve per fermare M4 in rotazione oraria e resettare HSC3, ad esso è associato il contatto "fc_rot/reset" (I0.7).

Fig.83: Finecorsa di M4.

PROGRAMMA TIA

Il programma è gestito come una macchina a stati, nella prima parte ci sono tutte le uscite e le condizioni per le quali vengono attivate e resettate.
Nella seconda parte è invece gestito l'avanzamento di stato.

1° PARTE – GESTIONE USCITE

Attiviamo i contatori veloci HSC1 e HSC3 impostandoli per il conteggio in quadratura di fase, con resettaggio esterno.
Questo segmento controlla il motore di apertura della pinza.
Il quale dovrà partire all'avvio del PLC, in caso di RESET o nello STATO 6 se il FC non e premuto.
Il FC e l'unica condizione di arresto.

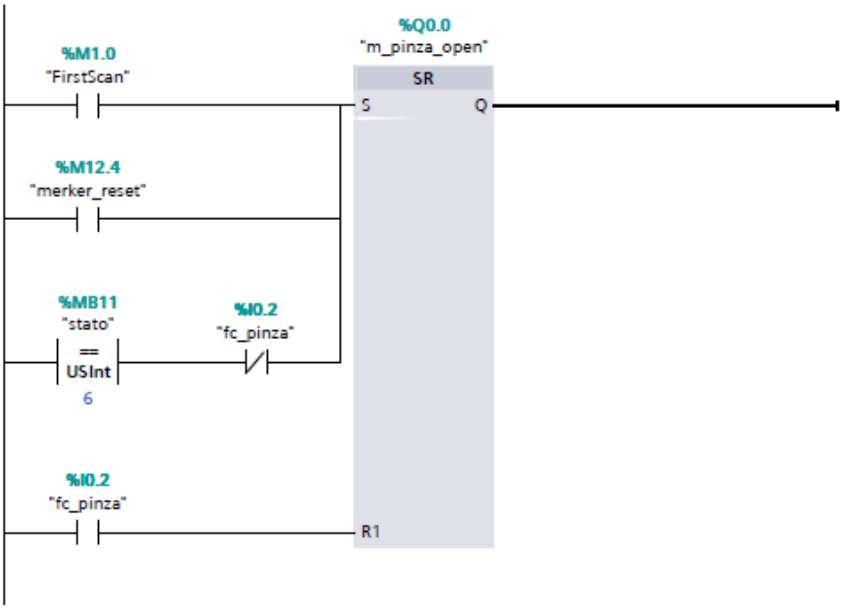


Fig.84: Parte della gestione delle uscite del programma in ladder.

Questo segmento controlla il motore di chiusura della pinza.
Il quale dovrà azionarsi solamente nello STATO 2 in una determinata posizione verticale e finché non raggiunge il valore di serraggio del pezzo.

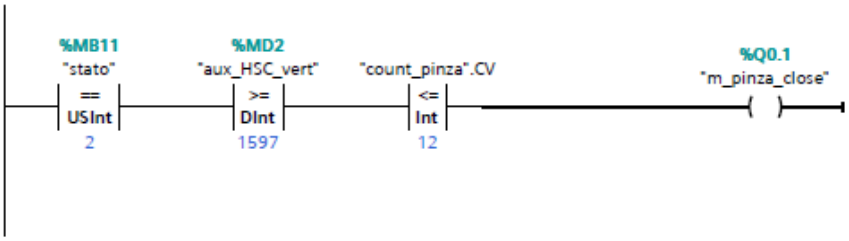
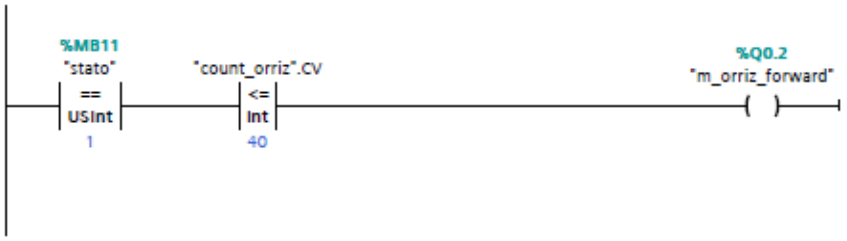


Fig.85: Parte che controlla la chiusura della pinza.

Questo segmento controlla il motore di avanzamento orizzontale.
Il quale dovrà azionarsi solamente nello STATO 1 fino al raggiungimento di un determinato valore di avanzamento.



Questo segmento controlla il motore di arretramento orizzontale.
Il quale dovrà partire all'avvio del PLC o in caso di RESET.
Il FC e l'unica condizione di arresto.

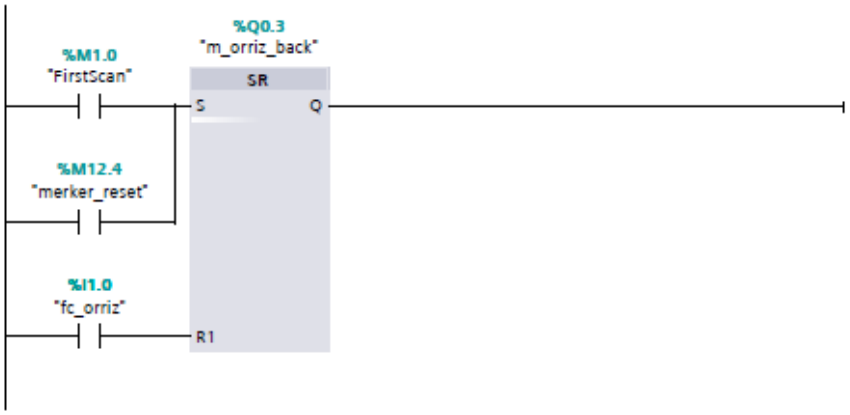


Fig.86: Parte che controlla il motore di arretramento orizzontale.

Questo segmento controlla il motore di discesa verticale.
Il quale dovrà partire nello STATO 2 e nello STATO 5 fino al raggiungimento di una determinata altezza.

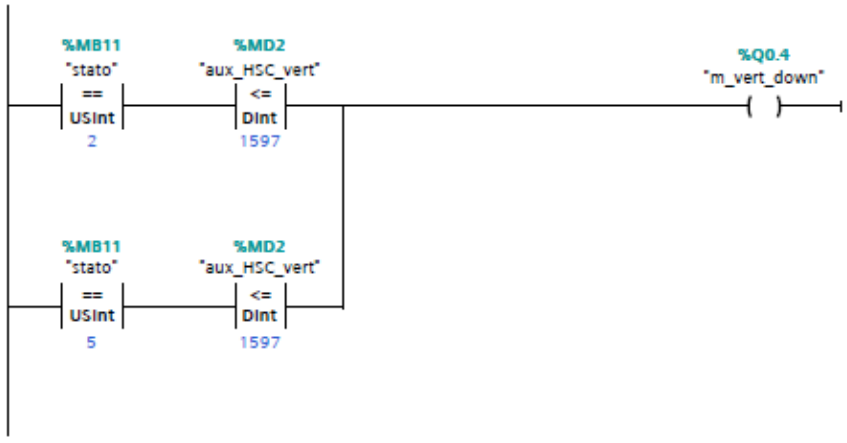


Fig.86: Parte che controlla il motore di arretramento orizzontale.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Questo segmento controlla il motore di salita verticale.
Il quale dovrà partire all'avvio del PLC, in caso di RESET, nello STATO 3 se si trova in una determinata posizione o nello STATO 7 se il FC non e premuto..
Il FC e il raggiungimento di una determinata altezza sono le condizioni di arresto.

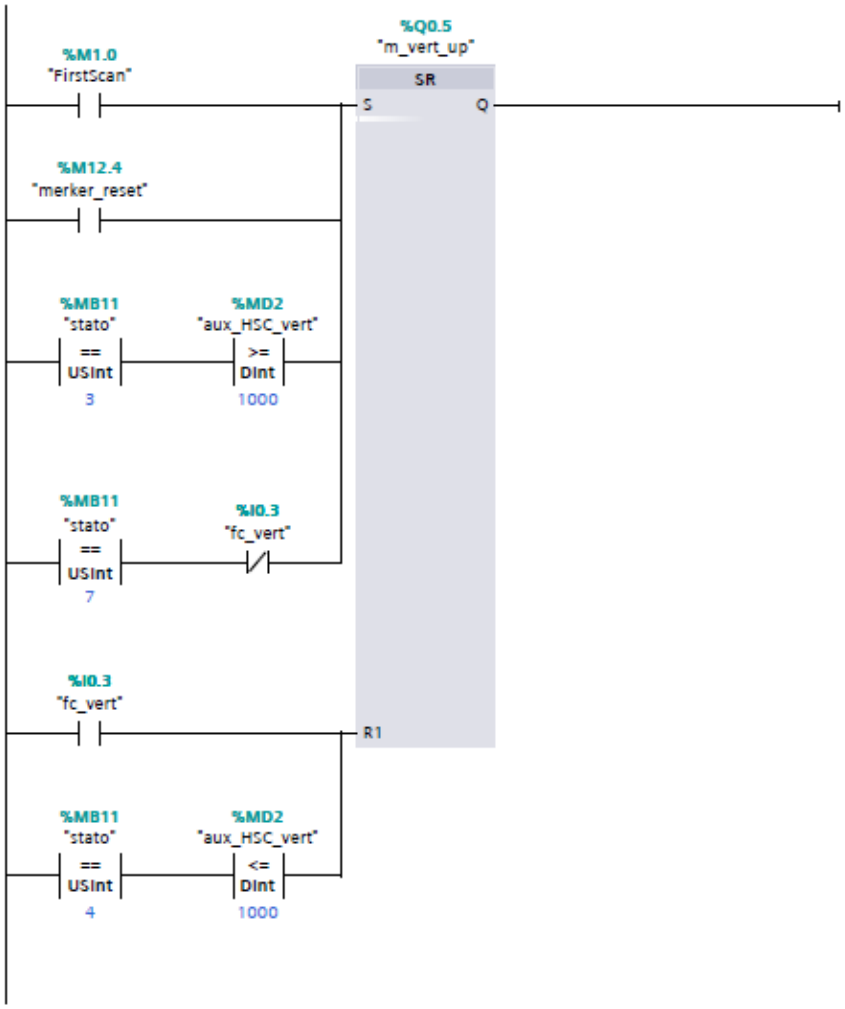
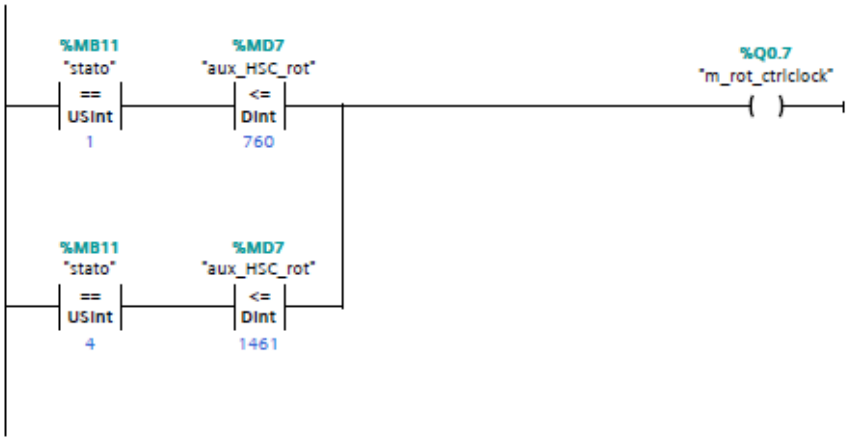


Fig.88: Parte che controlla il motore di salita verticale.

Questo segmento controlla il motore di rotazione oraria.
Il quale dovrà partire all'avvio del PLC o in caso di RESET.
Il FC e l'unica condizione di arresto.

Fig.89: Controlla il motore di rotazione oraria.

Questo segmento controlla il motore di rotazione oraria.
Il quale dovrà funzionare nello STATO 1 e nello STATO 4 fino al raggiungimento di due determinati valori.



Leggo l'area di memoria del conteggio dell'HSC1 e lo copio all'interno di una mia variabile di appoggio.

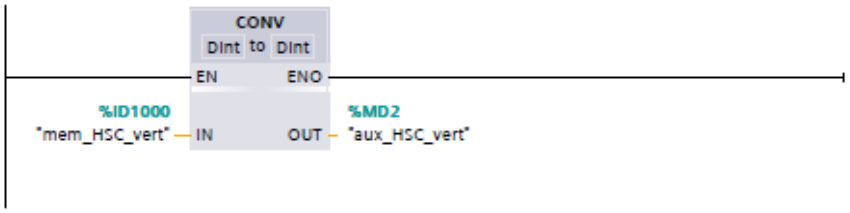


Fig.90: Parte di conversione che copia l'area di memoria dell'HSC1 in una variabile.

Leggo l'area di memoria del conteggio dell'HSC3 e lo copio all'interno di una mia variabile di appoggio.

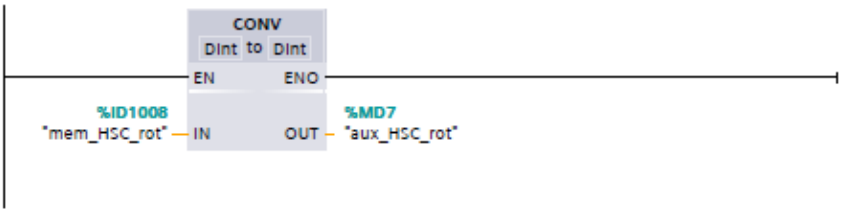


Fig.91: Parte di conversione che copia l'area di memoria dell'HSC3 in una variabile.

Conto gli impulsi relativi al conteggio orizzontale, resettabili tramite il relativo FC.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

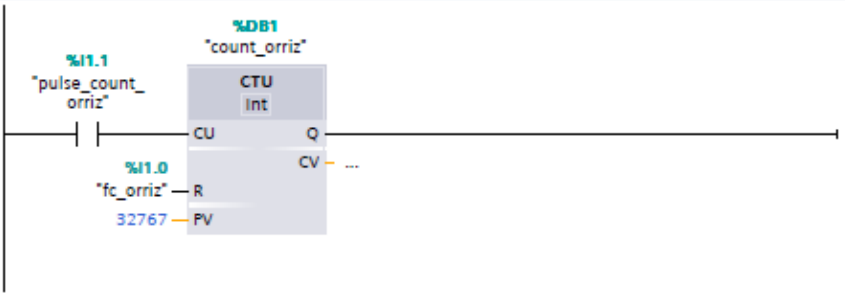


Fig.92: Parte che conta gli impulsi del conteggio orizzontale.

Conto gli impulsi relativi alla chiusura della pinza, resettabili tramite il relativo FC.

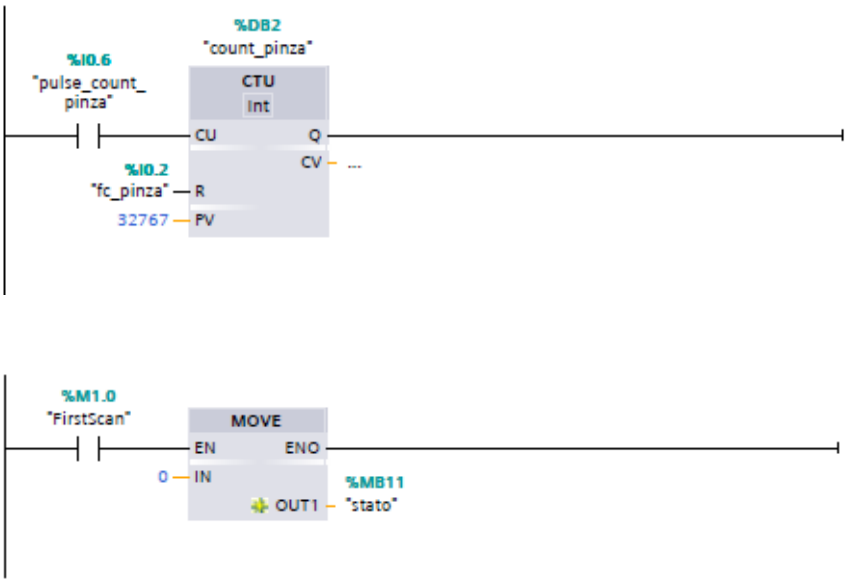


Fig.93: Parte che conta gli impulsi della chiusura della pinza.

2° PARTE – GESTIONE STATI

All'accensione del PLC posiziono all'interno della mia variabile STATO il valore 0.

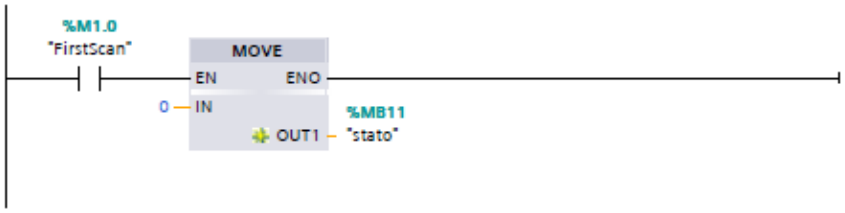


Fig.94:Parte che posiziona 0 all'interno della variabile stato.

Attendo l'impulso di abilitazione del PC per avanzare allo STATO 1.

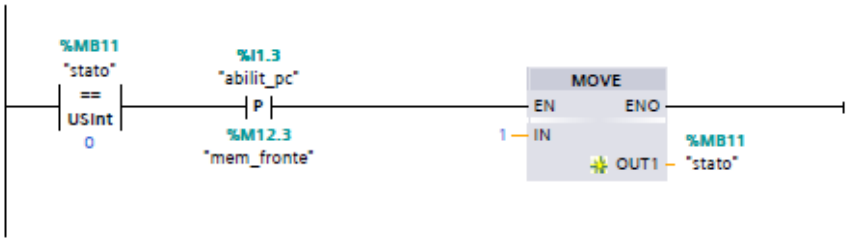


Fig.95: Parte che all'impulso di abilitazione fa si che si avanzi allo stato 1.

Attendo il raggiungimento di una determinata posizione orizzontale e di rotazione per avanzare allo STATO 2.

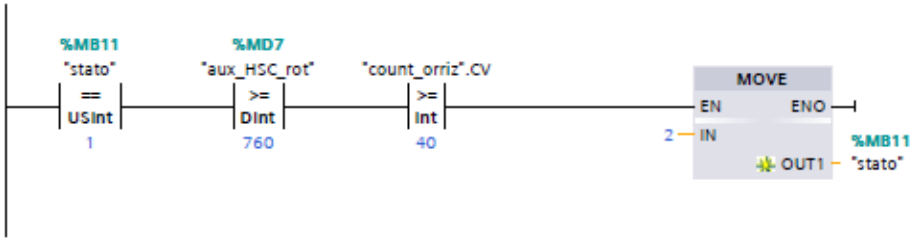


Fig.96: Parte che, arrivati in una data posizione e rotazione, fa si che si avanzi allo stato 2.

Attendo il raggiungimento di una determinata posizione verticale e della pinza per avanzare allo STATO 3.

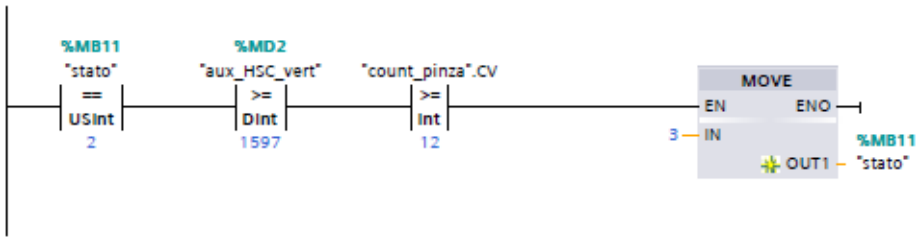


Fig.96: Parte che, arrivati in una data posizione e rotazione, fa si che si avanzi allo stato 2.

Attendo il completamento dello spostamento verticale per avanzare allo STATO 4.

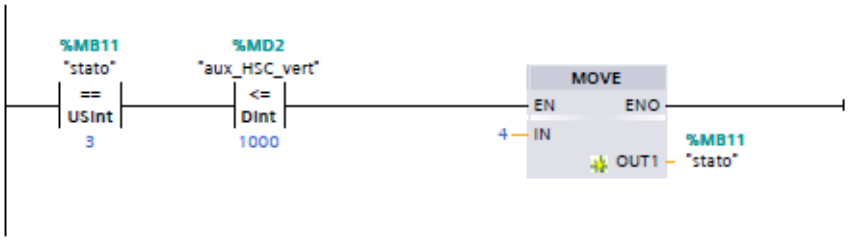


Fig.98: Parte che, alla fine dello spostamento verticale, fa si che si passi allo stato 4.

Attendo il completamento della rotazione per avanzare allo STATO 5.

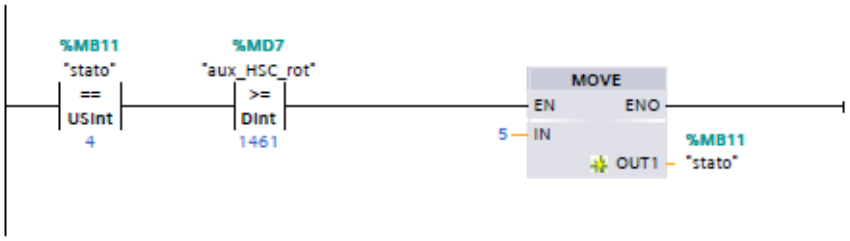


Fig.99: Parte che alla fine della rotazione fa avanzare allo stato 5.

Attendo il posizionamento verticale per avanzare allo STATO 6.

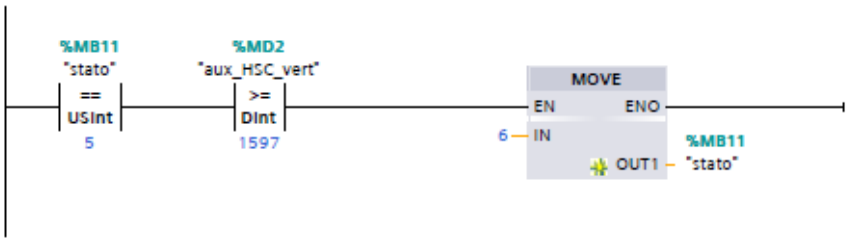


Fig.100: Parte che alla fine del posizionamento verticale fa avanzare allo stato 6.

Attendo l'apertura completa della pinza per avanzare allo STATO 7.

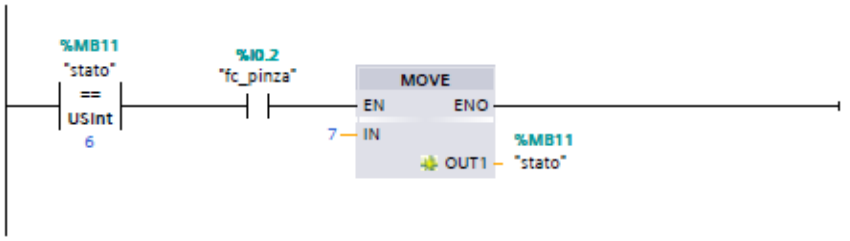


Fig.101: Parte che, all'apertura completa della pinza, fa avanzare allo stato 7.

Attendo la completa salita verticale per avanzare allo STATO 8.

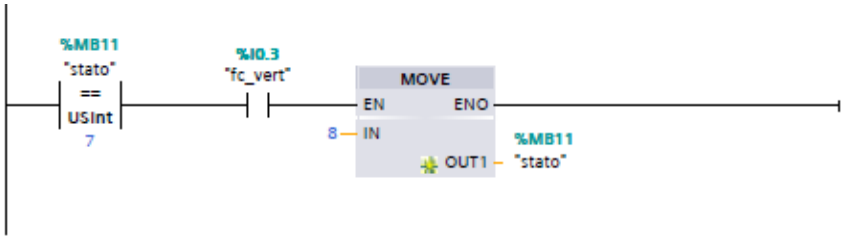


Fig.102: Parte che, alla completa salita verticale, fa avanzare allo stato 7.

Tengo attivo il merker di reset finche il braccio non si porta nella condizione iniziale attivando tutti i FC.
Raggiunta la posizione di partenza ritorno allo STATO 0.

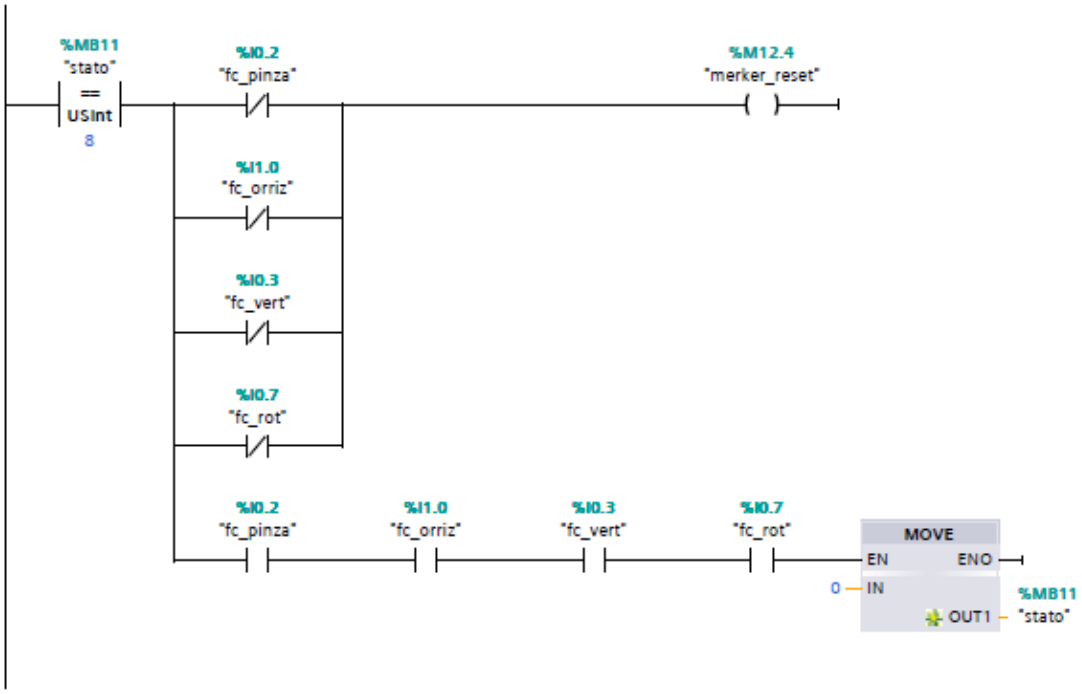
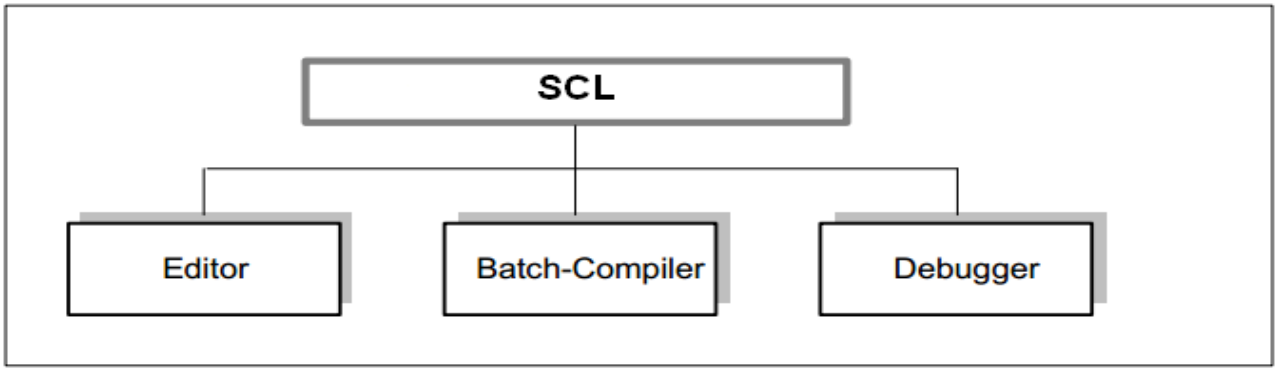


Fig.102: Parte che, alla completa salita verticale, fa avanzare allo stato 7.

SCL (Structured Control Language)

Teoria Linguaggio SCL

Andrea Benini
5AET 2013
(andrea.benini01@gmail.com)



Il linguaggio SCL è costituito da :

- un **Editor,** per programmare programmi composti di funzioni (FC), blocchi funzionali (FB), blocco organizzativi (OB), blocchi dati (DB) e tipi di dati definiti dall'utente (UDT). Facilitano il lavoro del programmatore numerose funzioni efficaci.

- un **Compilatore** per compilare ossia convertire in codice macchina MC7 il programma editato in precedenza. Il codice MC7 generato può essere eseguito su tutte le CPU del sistema di automazione S7-300/400 a partire dalla CPU 314.

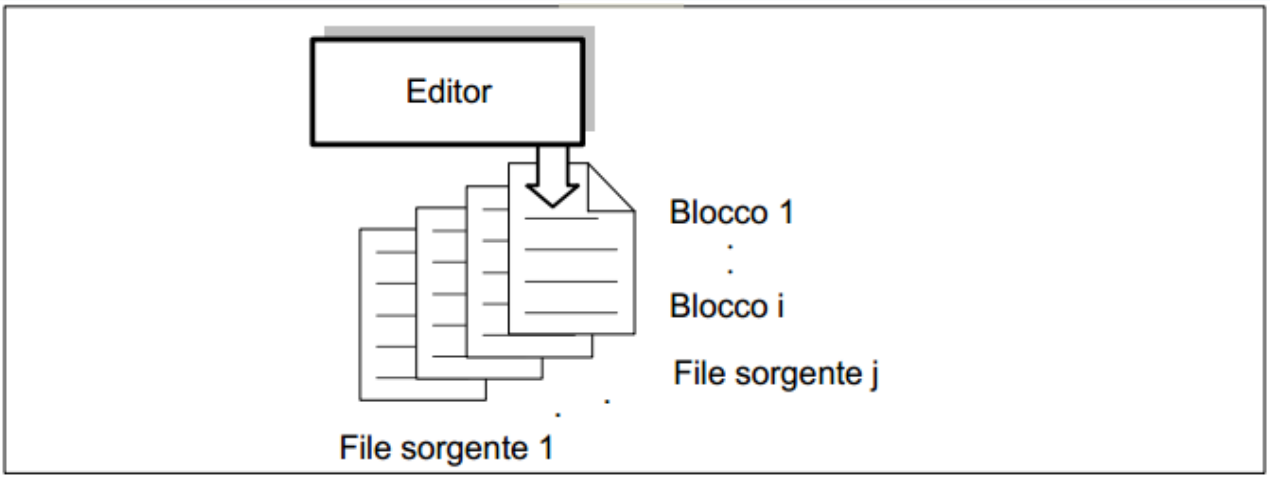
- un **Debugger**, che consente la ricerca di errori logici di programmazione in una compilazione esente da errori. La ricerca degli errori avviene nel linguaggio sorgente.

Nelle fasi di test e collaudo in **Online**, ha una minore velocità nel trovare errori di programmazione rispetto ai linguaggi di tipo grafico.
Risulta però essere molto potente, se dobbiamo utilizzare all'interno di un processo, formule complesse con relativa memorizzazione e gestione in Database multipli.

CARATTERISTICHE DEL LINGUAGGIO SCL :

Editor.
E' un editor di testi, che consente l'elaborazione di qualsiasi tipo di testo. Il compito principale realizzabile con un editor è la generazione e l'elaborazione di file sorgente per programmi STEP 7. In un file sorgente si possono programmare uno o più blocchi.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

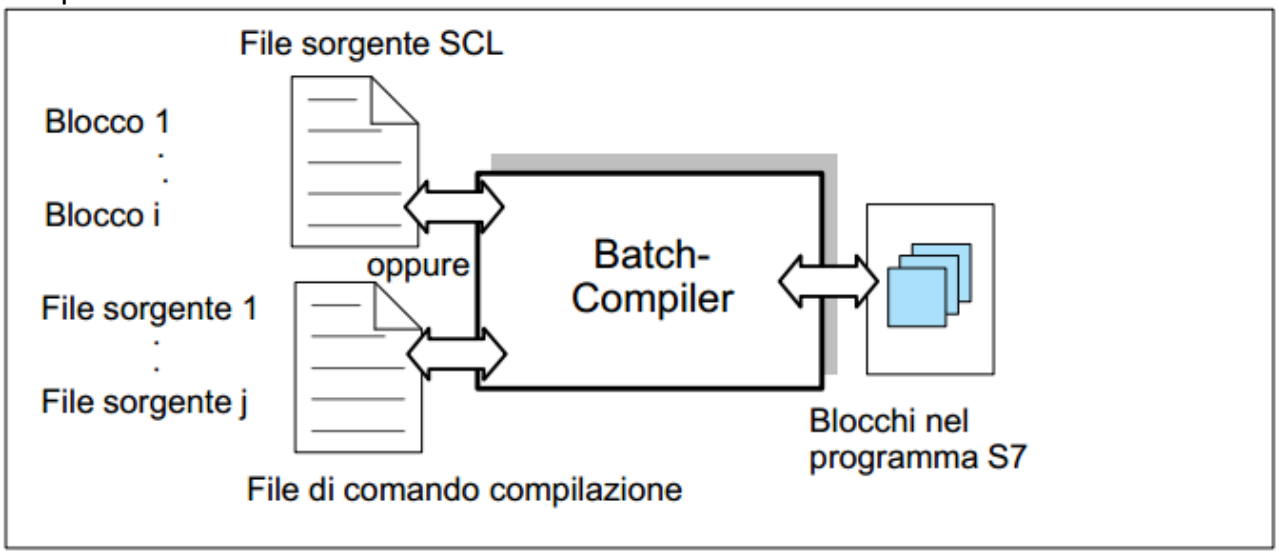


Funzioni che può svolgere l'Editor :

- Editare un intero file sorgente con uno o più blocchi.
- Editare un file di comando compilazione che consente di automatizzare la
- Compilazione di molti file sorgente.
- Utilizzo di funzioni supplementari, che consentono una pratica elaborazione del
- Testo sorgente, p. es. Cerca/Sostituisci.
- Impostare in modo ottimale l'editor in base alle proprie esigenze specifiche.

Compilatore.

Dopo aver creato i propri file sorgente con l'Editor SCL, essi devono essere compilati ossia convertiti in codice MC7.



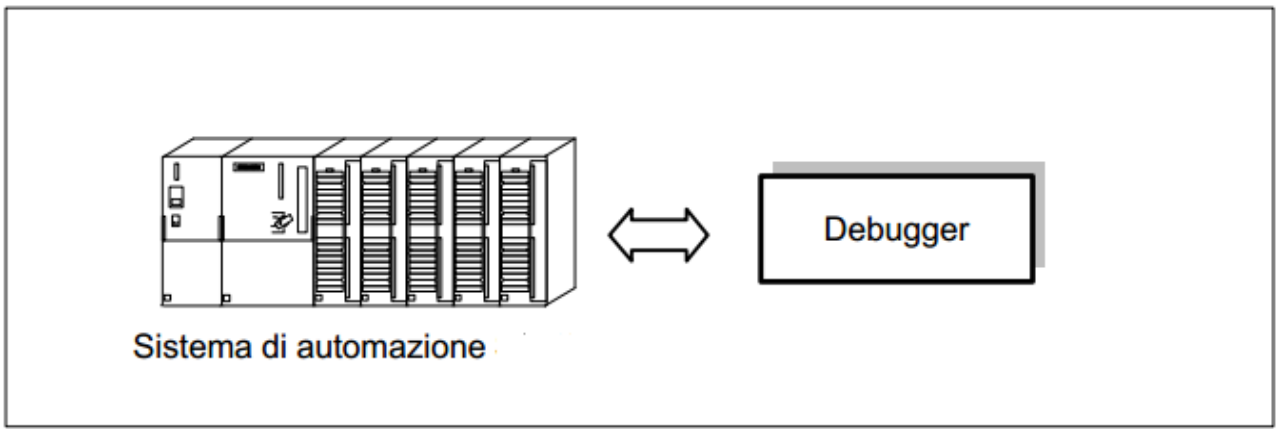
Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Funzioni che svolge il Compilatore:

- di compilare un file sorgente SCL con più blocchi in un solo lancio di compilazione.
- di compilare più sorgenti SCL mediante un file di comando compilazione contenente i nomi dei file sorgente.
- di impostare in modo ottimale il compilatore in base alle proprie esigenze specifiche.
- di visualizzare tutti gli errori e i messaggi di avviso che si verificano durante la compilazione.
- di localizzare con facilità i passaggi del testo sorgente contenente errori, eventualmente, come opzione, con descrizione dell'errore e indicazioni sul modo di eliminare l'errore.

Debugger.

Esso consente di controllare lo svolgimento di un programma nel PLC, per poter identificare eventuali errori logici.



SCL offre in tal senso due diverse tipologie di test:

- l'osservazione passo passo. Viene controllato l'intero svolgimento logico del programma. L'algoritmo del programma può essere eseguito istruzione per istruzione e in una finestra risultati si può osservare come vengono modificate le variabili elaborate.

- l'osservazione continua. Si può testare un intero gruppo di istruzioni nell'ambito di un blocco della sorgente. Durante lo svolgimento del test, i valori delle variabili ed i parametri vengono visualizzati in sequenza cronologica e – se possibile – aggiornati ciclicamente.

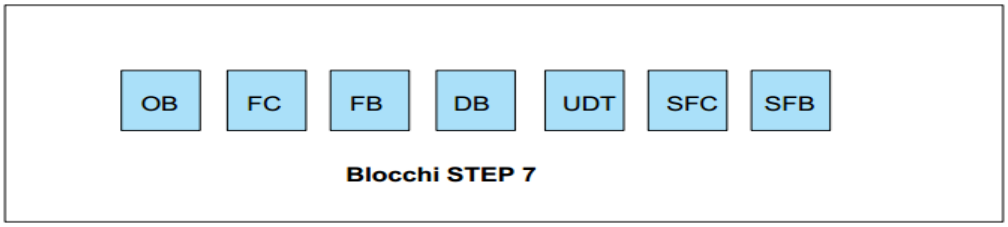
Vantaggi del linguaggio SCL.

Esso ha un linguaggio di programmazione avanzato.
SCL possiede però anche alcune proprietà per supportare la tecnica di programmazione strutturata:

- la struttura a blocchi di STEP 7
- blocchi predefiniti
- compatibilità con STEP 5

SCL è stato concepito in modo particolare per risolvere qualsiasi tipo di problema che può verificarsi nei progetti di automazione, per consentire all'utente di operare in STEP 7 con la massima efficacia in tutte le fasi del progetto.

Struttura a blocchi STEP 7 :



Abbreviazione	Tipo di blocco	Funzione
OB	Blocco organzzativo	Interafccia del sistema operativo e programma utente
FC	Funzione	Blocco con possibilità di trasferimento di parametri senza memoria
FB	Blocco funzionale	Blocco con possibilità di trasferimento di parametri con memoria
DB	Blocco dati	Blocco per l deposito di dati utente
UDT	Tipo di dati definito dall'utente	Blocco per il deposito di un tipo di dati definito dall'utente

Ci sono altri blocchi predefiniti che possono essere utilizzati per programmare funzioni di comunicazione:

Abbreviazione	Tipo di blocco	Funzione
SFC	Funzione di sistema	Proprietà analoghe a quelle di una funzione (FC)
SFB	Blocco funzionale di sistema	Proprietà analoghe a quelle di un blocco funzionale (FB

Montemaggi Pablo
Farneti Alessandro
5AET 2013
(pablo.montemaggi@gmai.com)
(farneti.alessandro@gmail.com)

Sintassi

A seconda del tipo di diramazione è possibile programmare le seguenti forme dell'istruzione:

- Diramazione tramite IF:

SCL

```
IF <Condizione> THEN <Istruzioni>
END_IF
```

Se la condizione è soddisfatta vengono eseguite le istruzioni programmate dopo THEN. Se la condizione non è soddisfatta, l'elaborazione del programma prosegue con la prima istruzione dopo END_IF.

- Diramazione tramite IF e ELSE:

SCL

```
IF <Condizione> THEN <Istruzioni1>
ELSE <Istruzioni0>
END_IF
```

Se la condizione è soddisfatta vengono eseguite le istruzioni programmate dopo THEN. Se la condizione non è soddisfatta vengono eseguite le istruzioni programmate dopo ELSE. Successivamente l'elaborazione del programma prosegue con la prima istruzione dopo END_IF.

- Diramazione tramite IF, ELSIF e ELSE:

SCL

```
IF <Condizione1> THEN <Istruzioni1>
ELSIF <Condizione2> THEN <Istruzione2>
ELSE <Istruzioni0>
END_IF;
```

Se la prima condizione (<Condizione1>) è soddisfatta vengono eseguite le istruzioni programmate dopo THEN (<Istruzioni1>). Elaborate le istruzioni, l'elaborazione del programma prosegue dopo END_IF.
Se la prima condizione non è soddisfatta viene verificata la seconda (<Condizione2>). Se la seconda condizione (<Condizione2>) è soddisfatta vengono eseguite le istruzioni <Istruzioni2> programmate dopo THEN. Elaborate le istruzioni, l'elaborazione del programma prosegue dopo END_IF.
Se nessuna delle condizioni è soddisfatta vengono eseguite le istruzioni (<Istruzioni0>) programmate dopo ELSE, quindi l'elaborazione del programma prosegue dopo END_IF.
All'interno dell'istruzione IF è possibile annidare qualsiasi combinazione di ELSIF e THEN.
La programmazione di un ramo ELSE è opzionale.

La sintassi dell'istruzione IF è costituita dalle seguenti parti:

Parte	Tipo di dati	Descrizione
<Condizione>	BOOL	Espressione che viene analizzata
<Istruzioni>	-	Instruzioni che vengono eseguite se la condizione è soddisfatta. Fanno eccezione le istruzioni programmate dopo ELSE, che vengono eseguite se non è soddisfatta nessuna condizione all'interno del loop di programma.

Esempio:
Il seguente esempio mostra il funzionamento dell'istruzione:

SCL

```
IF "Tag_1" = 1 THEN "Tag_Value" := 10;
ELSIF "Tag_2" = 1 THEN "Tag_Value" := 20;
ELSIF "Tag_3" = 1 THEN "Tag_Value" := 30;
ELSE "Tag_Value" := 0;
END_IF;
```

La seguente tabella mostra il funzionamento dell'istruzione in base a valori di operandi concreti:

Operando	Valore			
Tag_1	1	0	0	0
Tag_2	0	1	0	0
Tag_3	0	0	1	0
Tag_Value	10	20	30	0

Istruzioni principali
Le istruzioni **SCL** utilizzate per la programmazione dei blocchi si avvalgono di determinati tipi di dati per rilevare il valore di una funzione. Per alcune istruzioni **SCL** è possibile utilizzare solo un tipo di dati ben preciso. Per queste istruzioni non è consentito modificare il tipo di dati. La maggior parte delle istruzioni **SCL**, tuttavia, può funzionare con diversi tipi di dati. Queste istruzioni si suddividono nei tipi seguenti:

- Istruzioni per le quali il tipo di dati del valore della funzione è determinato dal tipo di dati dei parametri di ingresso. Questo caso riguarda la maggior parte delle istruzioni.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

- Istruzioni per le quali il tipo di dati è preimpostato. Questo caso riguarda le istruzioni che sono elencate nella tabella sottostante.

Nel caso del secondo gruppo è necessario modificare il tipo di dati preimpostato se non coincide con quello del parametro di ingresso utilizzato. Fondamentalmente è possibile modificare il tipo di dati con la sintassi seguente: _<Tipo di dati>

Istruzioni SCL con tipo di dati preimpostato

La tabella seguente mostra le istruzioni **SCL** con tipi di dati preimpostati:

Istruzione	**Tipo di dato preimpostato**
CEIL	DINT
DECO	DWORD
ENCO	INT
FLOOR	DINT
NORM_X	REAL
PEEK	BYTE
ROUND	DINT
SCALE_X	INT
TRUNC	DINT

ITT "B. Pascal" Cesena – 5AET2013

PROGETTO AUTOMAZIONE

Nastro trasportatore con stazioni di lavorazione

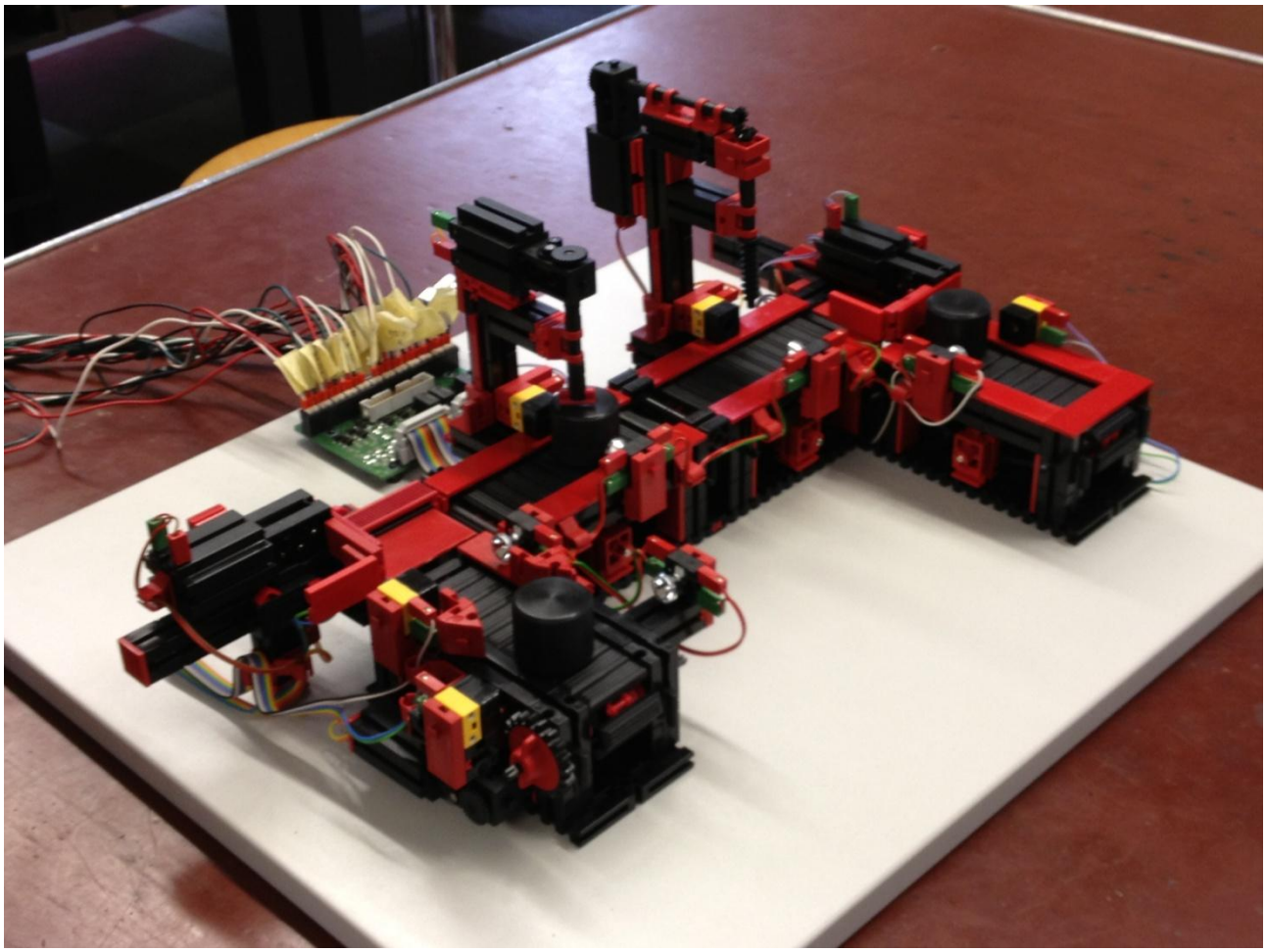


Fig.105: Nastro trasportatore con stazioni di lavorazione.

Il modellino rappresenta un processo industriale in cui il pezzo viene lavorato da macchine utensili e trasportato nelle varie fasi.
Il modellino comprende: 2 stazioni di lavorazione, una per la foratura e una per la fresatura, 4 nastri trasportatori e due carrelli che spingono i pezzi da un nastro all'altro.
In tutto sono presenti 8 motori DC, 6 a singola direzione e 2 bidirezionali.
Su ogni carrello sono presenti due finecorsa, uno anteriore e uno posteriore.
Lungo il percorso sono presenti 5 barriere luminose composte da una lampadina e un foto transistor.
Il modellino necessita 24V di alimentazione per gli attuatori e per i sensori.
Si può connettere tramite il connettore da 26 poli su due file oppure tramite la morsettiera.
Ha 9 uscite digitali che vanno da 0 a 24V. Nel caso dei finecorsa il livello logico alto (finecorsa premuto) corrisponde a 24V, invece nelle barriere luminose il livello logico alto (barriera ostruita) corrisponde ai 0V.
Tramite 10 ingressi digitali si controllano i motori, fornendo i 24V.
Nel caso dei motori bidirezionali è presente un contatto per la marcia avanti e uno per la marcia indietro. Per fargli cambiare direzione è necessario portare a 0 la direzione precedente prima di fornire i 24 V.

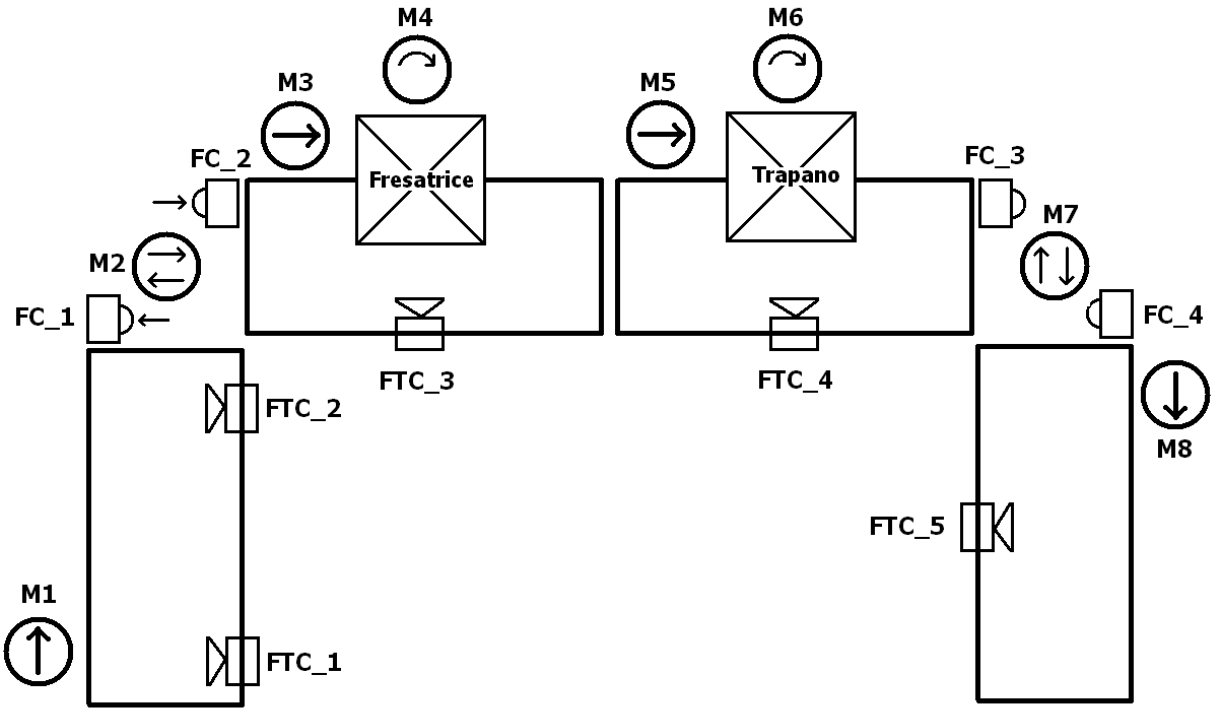


Fig.106: Descrizione del modellino del nastro trasportatore.

M1: Motore 1
M2: Motore 2
M3: Motore 3
M4: Motore 4
M5: Motore 5
M6: Motore 6
M7: Motore 7
M8: Motore 8

FTC_1: Fotocellula 1
FTC_2: Fotocellula 2
FTC_3: Fotocellula 3
FTC_4: Fotocellula 4
FTC_5: Fotocellula 5

FC_1: Finecorsa motore 2
FC_2: Finecorsa motore 2
FC_3: Finecorsa motore 7
FC_4: Finecorsa motore 7

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

MOTORI

M1: Gira e fa muovere il nastro 1 verso l'alto, secondo lo schema. Appena la FTC_1 viene oscurata M1 parte. Appena la FTC_2 viene oscurata il motore si blocca per dare il tempo a M2 di arretrare, appena FC_1 viene premuto M1 riparte fino a che il pezzo non si colloca sullo svincolo tra il nastro 1 e il nastro 2 (tempo prestabilito).

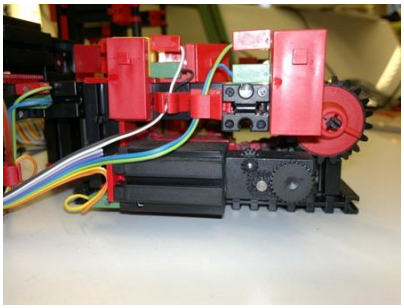


Fig.107A: Motore M1.

M2: Avanza e retrocede spingendo il pezzo da lavorare sul secondo nastro dallo svincolo tra il nastro 1 e il nastro 2.
FC_1 e FC_2 sono i finecorsa del motore.

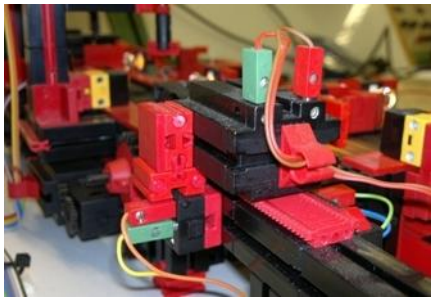


Fig.107B: Motore M2.

M3: Gira e fa muovere il nastro 2 verso destra ,secondo lo schema. Parte appena FC_2 viene premuto e appena FTC_3 viene oscurata il motore si blocca per il tempo in cui la fresatrice lavora. Poi riprende il suo moto fino a spingere il pezzo sul terzo nastro.

Fig.108: Motore M3.

M4: Girando aziona la fresatrice quando FTC_3 viene oscurata e lavora per un tempo prestabilito.

Fig.109: Motore M4.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

M5: Gira e fa muovere il nastro 3 verso destra secondo lo schema, parte quando M4 si ferma. Appena FTC_4 viene oscurata il motore si blocca per il tempo in cui il trapano lavora. Poi riprende il suo moto fino a che il pezzo non va sullo svincolo tra il nastro 3 e il nastro 4 (tempo prestabilito).

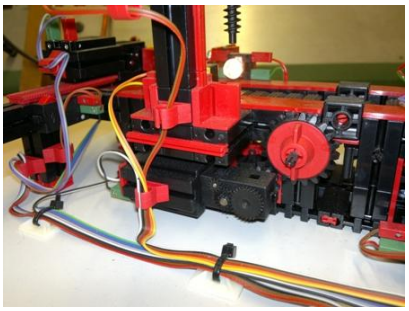


Fig.110: Motore M5.

M6: Girando aziona il trapano quando FCT_4 viene oscurata e lavora per un tempo prestabilito.

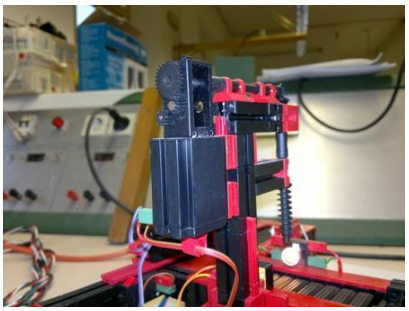


Fig.111: Motore M6.

M7: Il motore può avanzare o arretrare, spinge il pezzo da lavorare sul quarto nastro dallo svincolo tra il nastro 3 e il nastro 4.
FC_3 e FC_4 sono i finecorsa del motore.

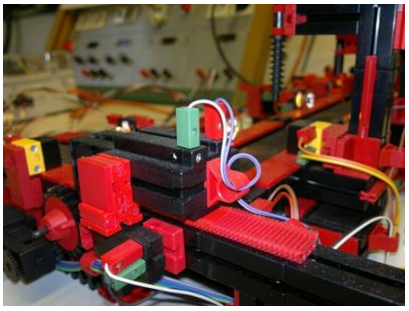


Fig.112: Motore M7.

M8: Gira in e muove il nastro 4 verso il basso, secondo lo schema. Parte appena il FC_4 viene premuto e gira fino a che FTC_5 viene oscurata.

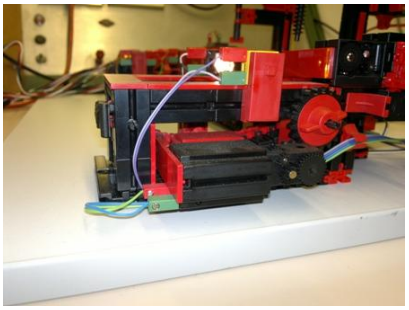
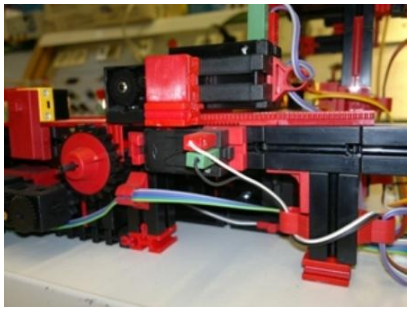


Fig.113: Motore M8.

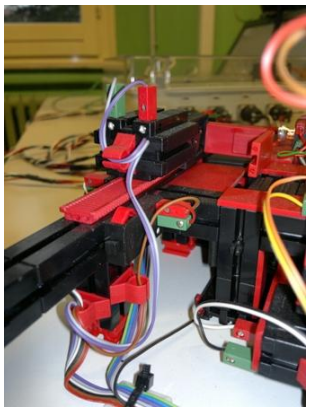
Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

FINECORSA



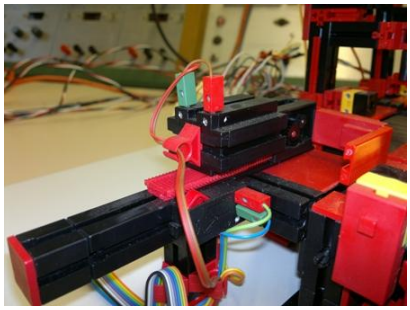
FC_1: Serve per non far uscire dalla guida M2, quando il motore arretra, appena FC_1 viene premuto M2 si blocca.

Fig.114: Finecorsa 1.



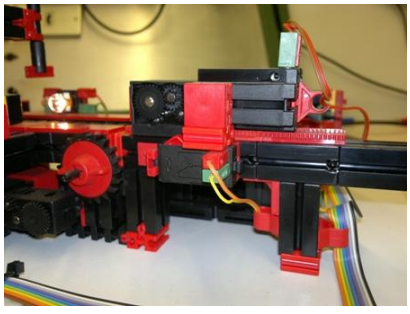
FC_2: Serve per non far uscire dalla guida M2, quando il motore avanza, appena FC_2 viene premuto M2 si blocca.

Fig.115: Finecorsa 2.



FC_3: Serve per non far uscire dalla guida M7, quando il motore retrocede, appena FC_3 viene premuto M7 si blocca.

Fig.116: Finecorsa 3.

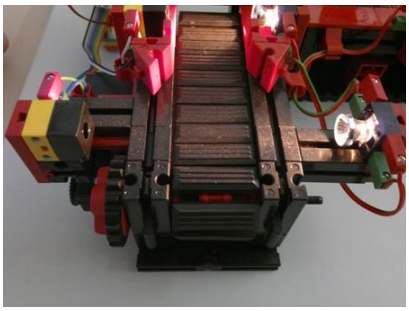


FC_4: Serve per non far uscire dalla guida M7, quando il motore avanza, appena FC_4 viene premuto M7 si blocca.

Fig.117: Finecorsa 4.

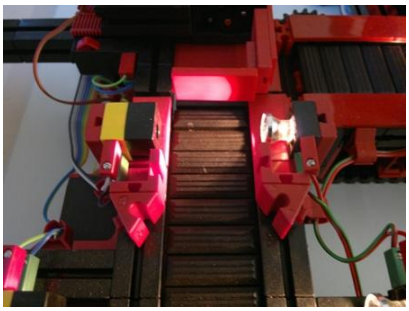
Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

FOTOCELLULE



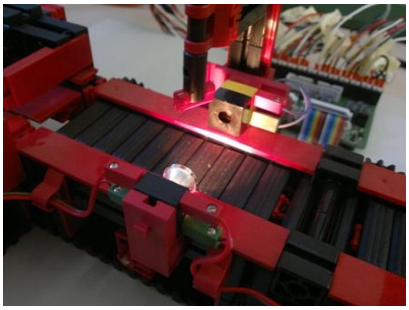
FTC_1: Fotocellula di carico, quando viene oscurata fa partire il primo nastro.

Fig.118: Fotocellula 1.



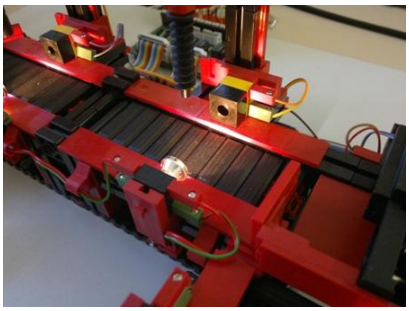
FTC_2: Serve per far bloccare M1 e dare il tempo a M2 di portarsi in posizione (tutto arretrato) per poi ripartire per un tempo prestabilito per posizionare il pezzo sullo svincolo.

Fig.119: Fotocellula 2.



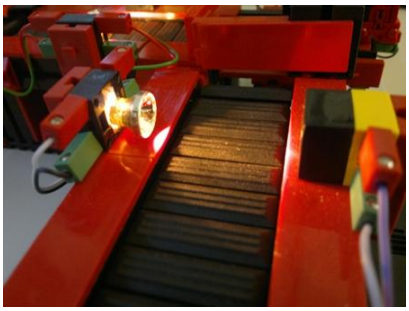
FTC_3: Serve per far fermare M3 appena il pezzo si trova sotto la fresatrice.

Fig.120: Fotocellula 3.



FTC_4: Serve per far fermare M5 appena il pezzo si trova sotto il trapano.

Fig.121: Fotocellula 4.

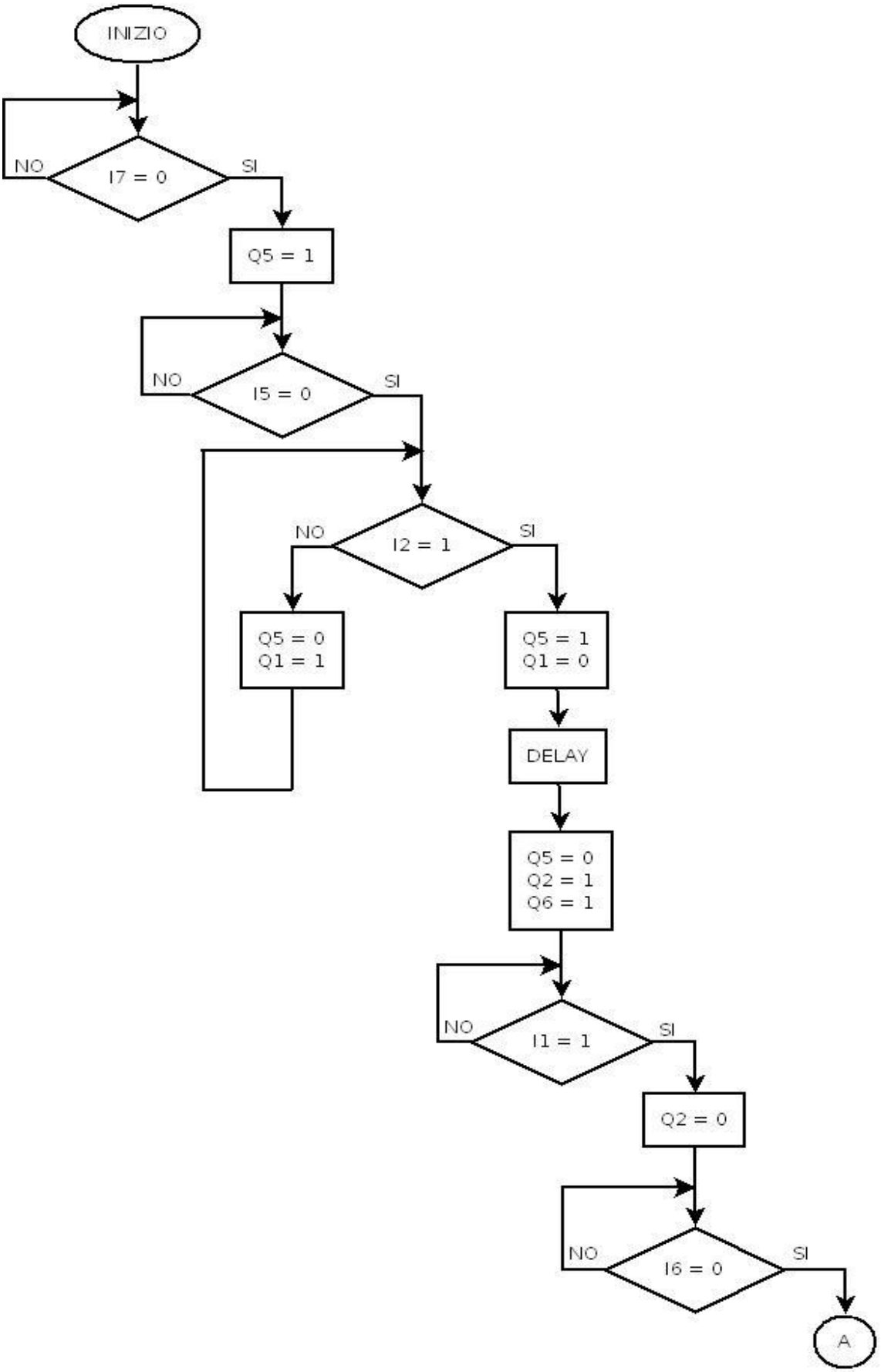


FTC_5: serve per far fermare M8 per portare il pezzo esattamente alla fine del nastro di lavorazione.

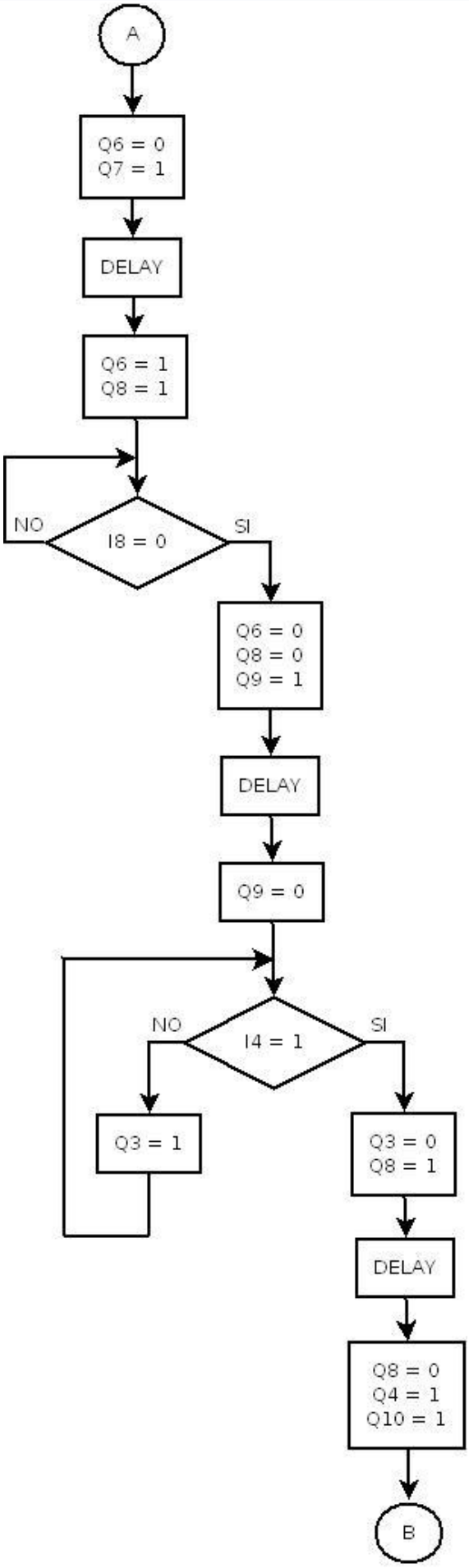
Fig.122: Fotocellula 5.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

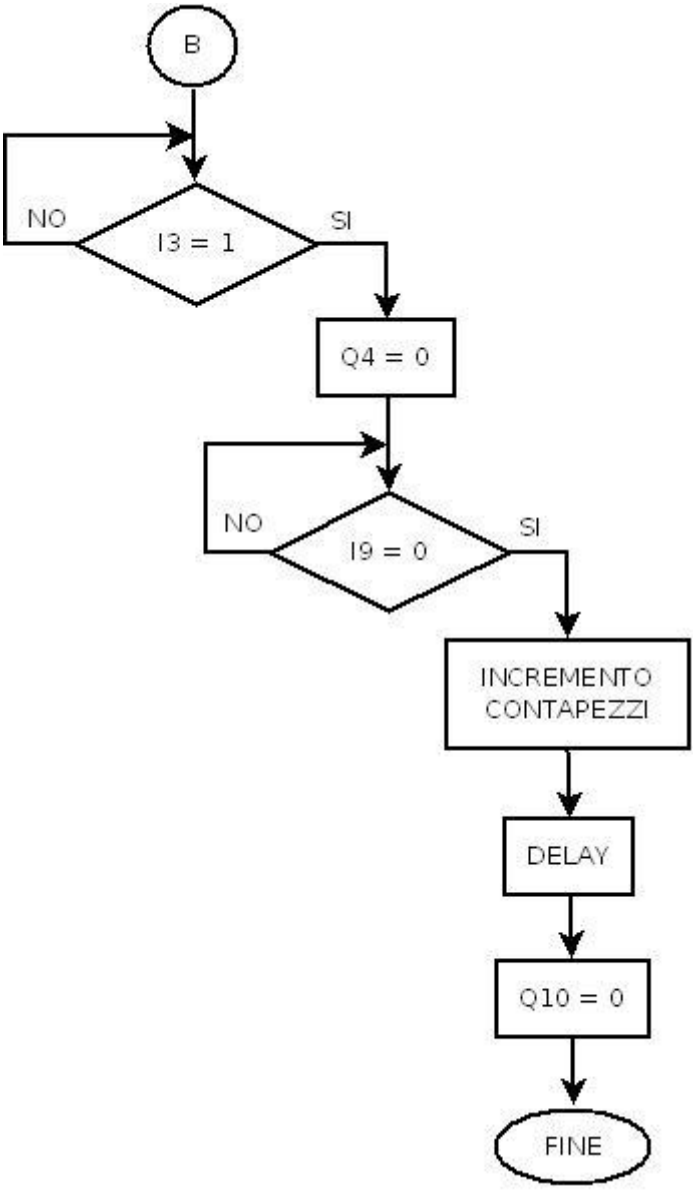
Questo è l'algoritmo di funzionamento del modellino:



Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com



Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com



Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

CICLO DI COMANDO:

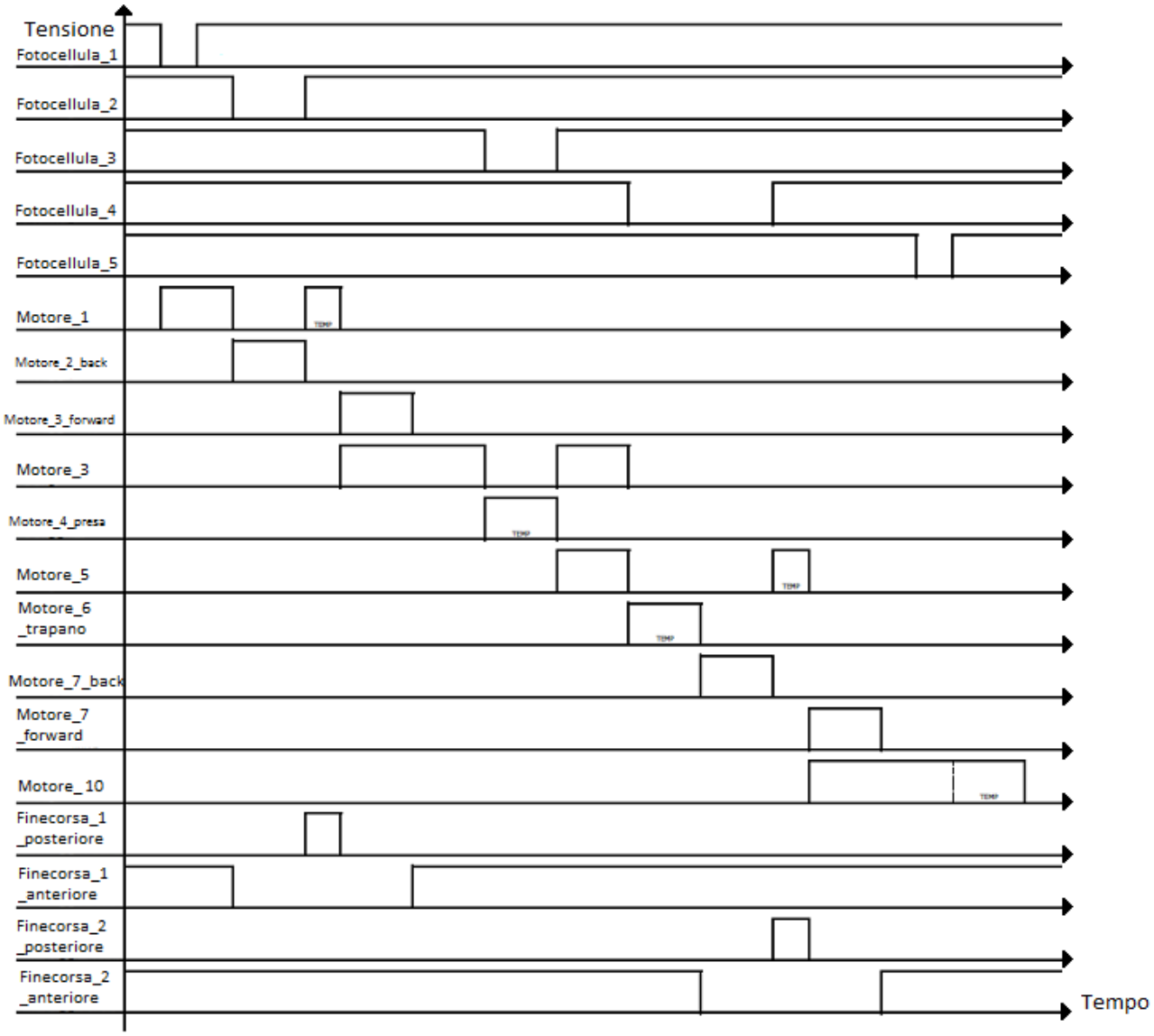


Fig.123: Cicli di comando del modellino.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

PROGRAMMA TIA:

La gestione può essere fatta in due differenti modi: seguendo una logica sequenziale, cioè a seconda di quali sensori sono attivi e quali erano attivi in precedenza attivo o meno le uscite, oppure come una macchina a stati.

NB: inizialmente il programma non svolge correttamente le funzioni assegnate in quanto **NON** vengono resettate le variabili, contatori e timer. Il resto del programma è correttamente funzionante.

GESTIONE SEQUENZIALE (UFFICIALE OLIMPIADI D'AUTOMAZIONE SIEMENS)

Appena caricato il pezzo avvio il motore, il quale lo porterà fino alla successiva fotocellula.

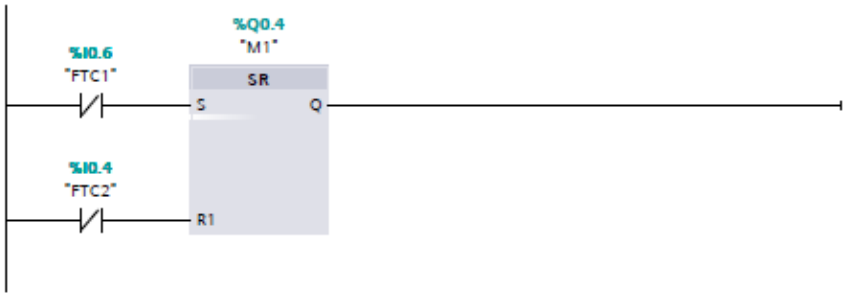


Fig.124: Parte che porta il pezzo fino alla fotocellula successiva.

Una volta che il pezzo raggiunge la FTC faccio arretrare il braccio di carico fino al fine corsa posteriore.

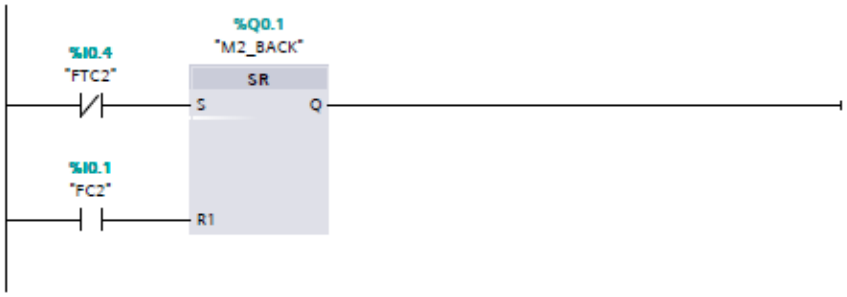


Fig.125: Parte che fa arretrare il braccio sino al finecorsa posteriore quando il pezzo arriva alla FTC.

Una volta che il braccio è in posizione attendo che esso venga caricato.

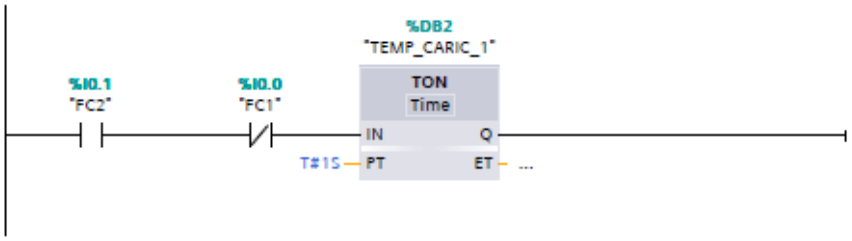


Fig.126: Parte che detta il tempo di carica del braccio.

Avvio il motore una volta che il braccio è arretrato completamente e attendo il tempo preimpostato.

Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

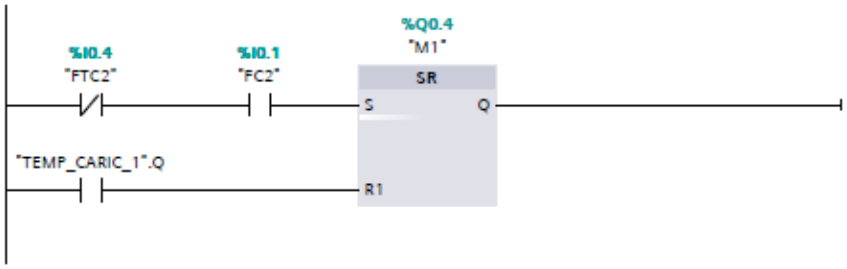
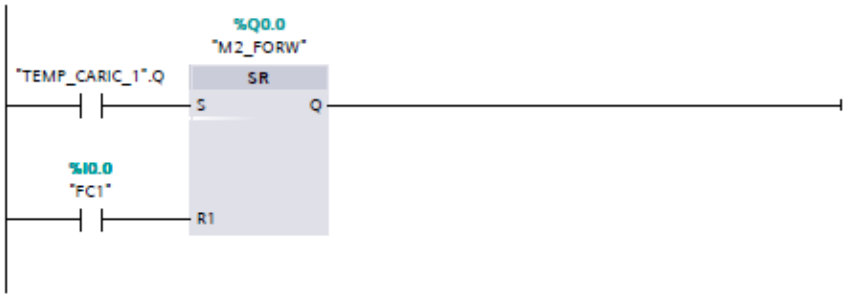


Fig.127: Parte che avvia il motore quando il braccio è arretrato completamente.

Dopo aver atteso il tempo preimpostato avvio il motore di avanzamento del braccio fino al raggiungimento del FC anteriore.



Mentre il braccio di carica avanza attivo il successivo nastro trasportatore, che mi trasporterà il pezzo fino alla fresa.

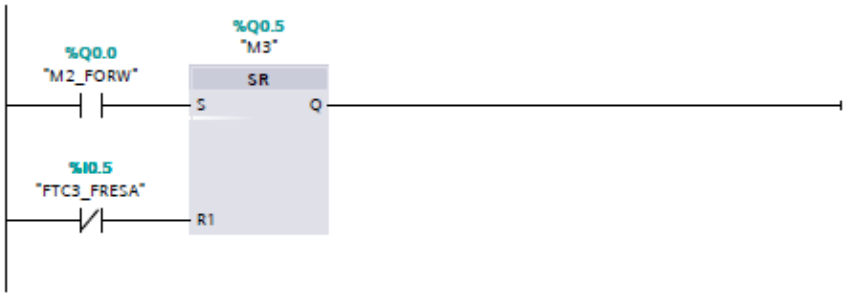


Fig.128: Parte che attiva il nastro mentre il braccio avanza.

Una volta posizionato il pezzo sotto la fresa inizia a contare il tempo di lavorazione.

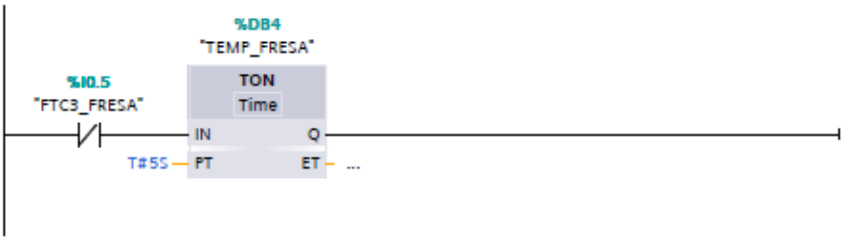


Fig.129: Tempo di lavorazione.

Raggiunta la posizione il pezzo viene lavorato da una fresa per il tempo preimpostato.

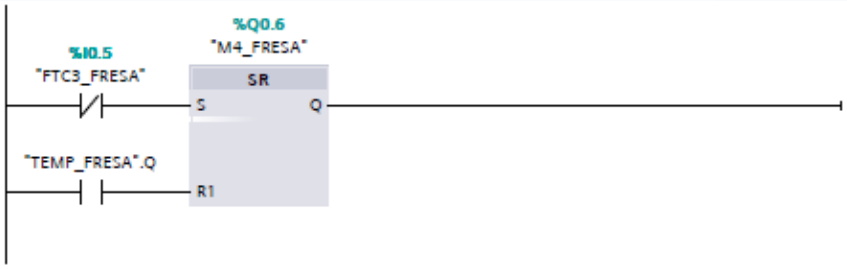


Fig.130: Settaggio lavorazione fresa.

Una volta finita la lavorazione della fresa il pezzo viene trasportato fin sotto al trapano da due nastri trasportatori.

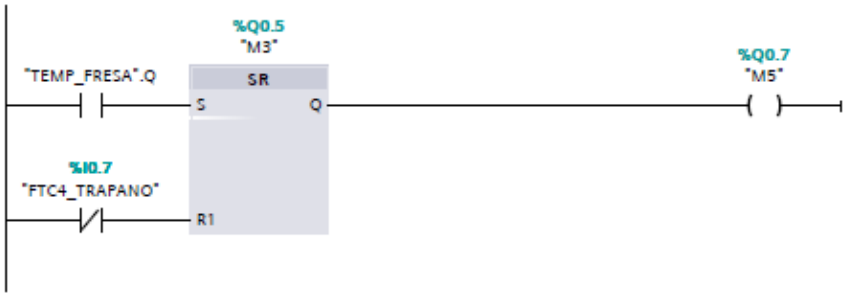


Fig.131: Parte che fa si che il pezzo sia trasportato da due nastri.

Una volta posizionato il pezzo sotto il trapano inizia a contare il tempo di lavorazione.

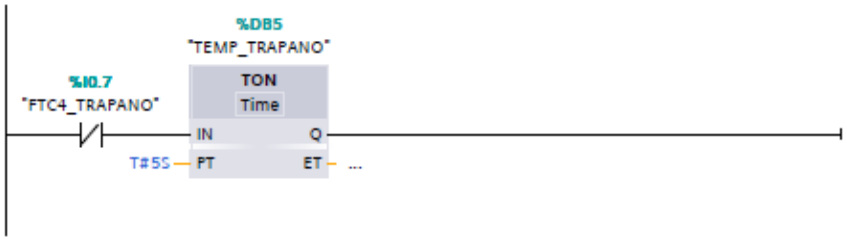


Fig.132: Tempo di lavorazione del trapano.

Raggiunta la posizione il pezzo viene lavorato da un trapano per il tempo reimpostato.

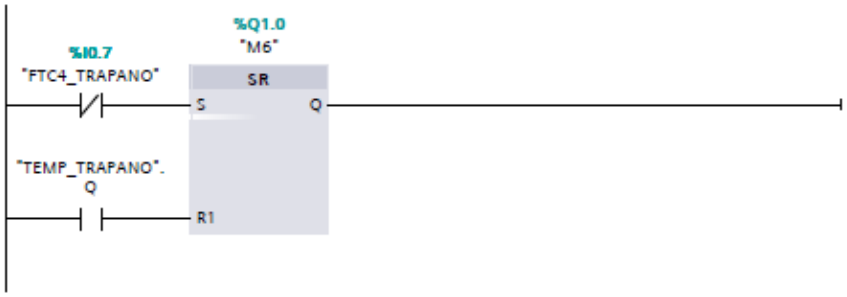


Fig.133: Il trapano lavora per il tempo reimpostato.

Una volta finita la lavorazione il braccio di scarico arretra fino al FC posteriore.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

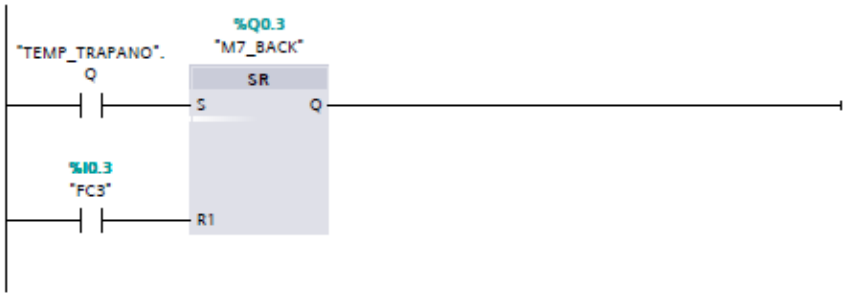


Fig.134: Parte che fa arretrare il braccio nella parte posteriore alla fine della lavorazione.

Quando il braccio raggiunge il FC posteriore inizio a contare il tempo di scarico pezzo.

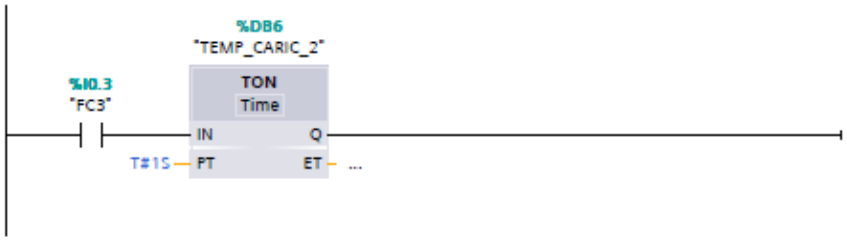


Fig.135: Tempo di scarico del pezzo.

Una volta raggiunto il FC posteriore attivo il nastro trasportatore sotto al trapano per trasportare il pezzo sul braccio di scarico per il tempo preimpostato.

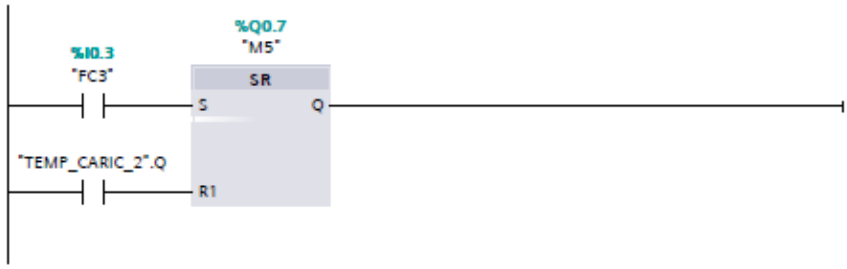


Fig.136: Attivazione nastro trasportatore.

Dopo aver atteso il tempo preimpostato avvio il motore di avanzamento del braccio di scarico, fino al FC anteriore.

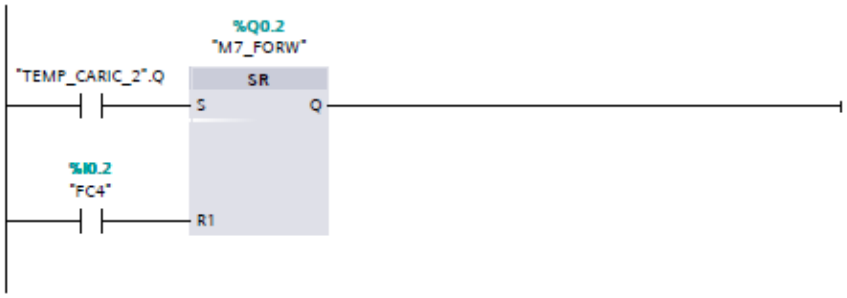


Fig.137: Determina il movimento del braccio di scarico fino alla fotocellula anteriore.

Quando leggo un fronte di salita sul FC anteriore del braccio e uno spegnimento dello stesso attivo il nastro trasportatore alla fine del processo produttivo per un determinato tempo.

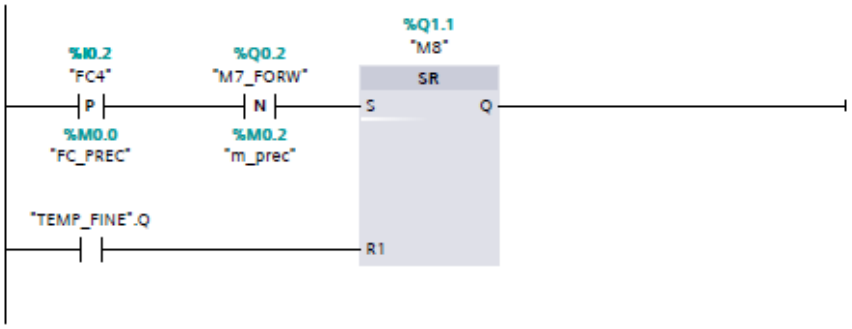


Fig.138: attivazione del nastro trasportatore.

Quando rilevo un passaggio davanti alla FTC alzo una flag che rimane attiva per un determinato lasso di tempo.

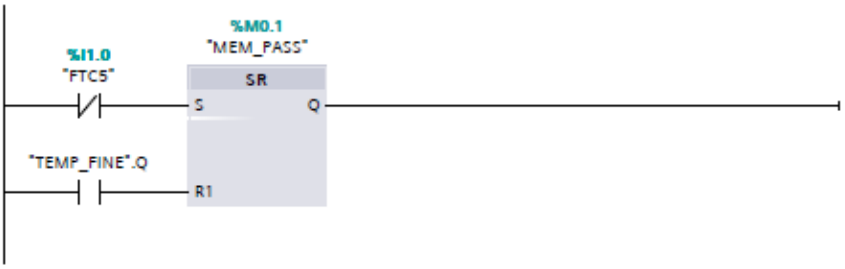


Fig.139: Alza una flag al passaggio del pezzo davanti alla fotocellula.

Quando si alza la flag inizio a contare fino ad un certo tempo preimpostato, una volta che scatta il timer si resetta.

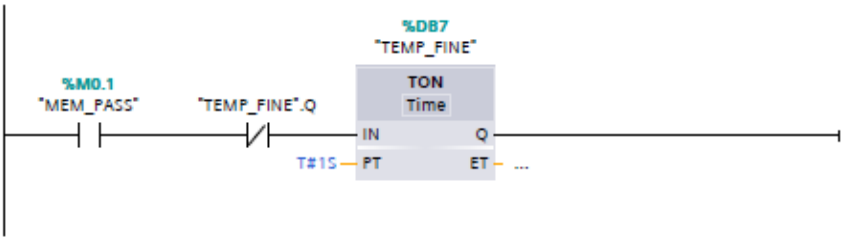


Fig.140: Il timer è attivato dalla flag e quando finisce di contare si resetta.

MACCHINA A STATI

All'interno del MAIN andiamo ad inizializzare la variabile STATO, mettendola a 0 al primo ciclo.

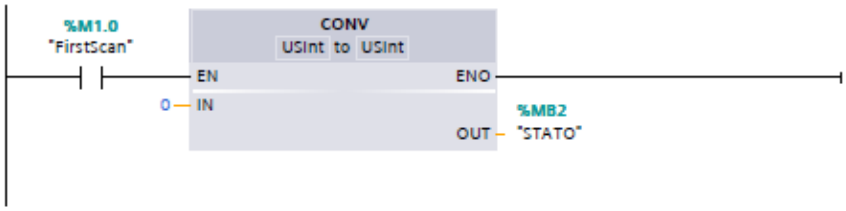


Fig.141: Mette a 0 la variabile stato al primo ciclo.

Utilizzando il contatto "AlwaysTRUE" precedentemente abilitato eseguiamo ad ogni ciclo il blocco FC1 il quale gestirà l'avanzamento e l'esecuzione degli stati.

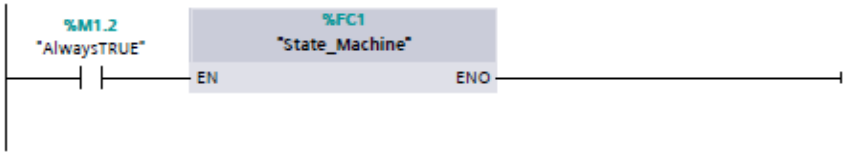


Fig.141: Mette a 0 la variabile stato al primo ciclo.

Il blocco FC1 è scritto in SCL.
Usiamo una case structure per gestire i vari valori di STATO, cioè i nostri stati.
Nel primo stato attendo che la fotocellula in ingresso venga oscurata per avanzare allo stato 1.

```
CASE "STATO" OF

  0: IF ("FTC1" = 0) THEN
     "STATO" := 1;
     END_IF;
```

Nello stato 1 eseguo la funzione relativa e attendo che si alzi la flag per passare allo stato successivo.

```
  1: "stato_1"();
     IF "flag_fine_stato" THEN
       "flag_fine_stato" := 0;
       "STATO" := 2;
     END_IF;
```

La funzione relativa allo stato 1 fa partire il motore del primo nastro se la prima fotocellula è oscurata e lo ferma dopo un determinato tempo.

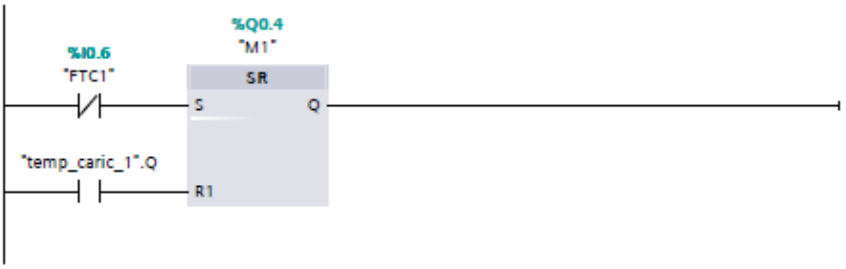


Fig.143: Fa partire il motore del primo nastro se la prima fotocellula è oscurata.

Il passaggio per la seconda fotocellula fa alzare una flag che memorizza il passaggio.

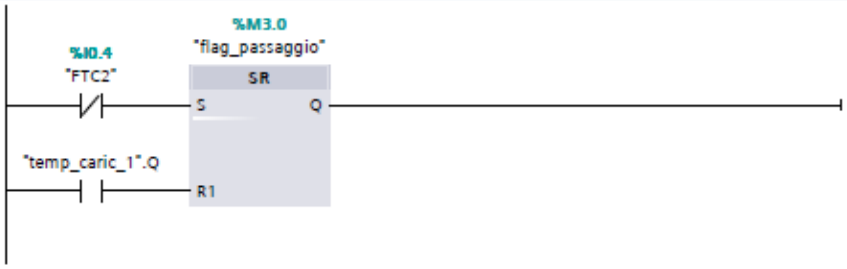


Fig.144: Permette di alzare la flag al passaggio dell'oggetto davanti alla seconda fotocellula.

Il timer scattando attiva la flag di fine stato.

Fig.144: Permette di alzare la flag al passaggio dell'oggetto davanti alla seconda fotocellula.

La flag di passaggio attiva il timer che inizia a contare.

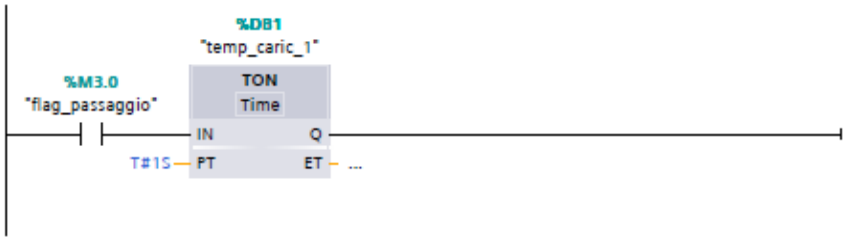


Fig.146: Il timer inizia a contare.

Nello stato 2 eseguo la relativa funzione e attendo che FC1 scatti per passare allo stato successivo.

```
2: "stato_2"();
  IF "FC1" THEN
    "STATO" := 3;
  END_IF;
```

Nella funzione relativa al secondo stato attivo il motore 2 dal finecorsa posteriore a quello anteriore.

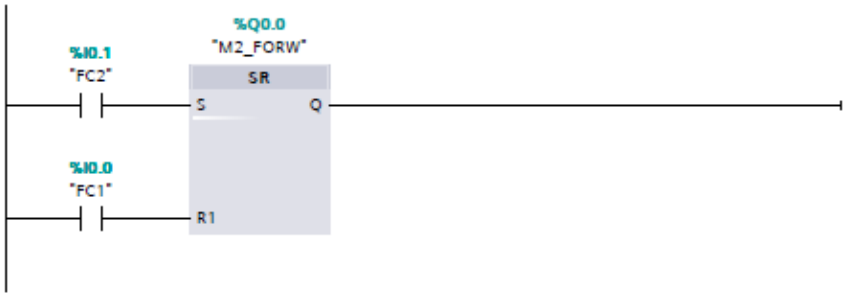


Fig.147: Il motore 2 viene attivato dal finecorsa posteriore a quello anteriore.

Nello stato 3 eseguo la relativa funzione e attendo che il FC2 scatti per passare allo stato successivo.

```
3: "stato_3"();
  IF "FC2" THEN
    "STATO" := 4;
  END_IF;
```

Nella funzione relativa al terzo stato faccio retrocedere il motore fino al finecorsa posteriore.

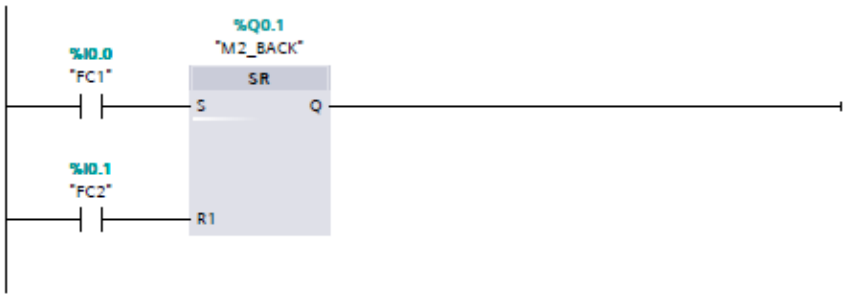


Fig.148: Il motore retrocede fino al finecorsa posteriore.

Nello stato 4 eseguo la relativa funzione e attendo che la fotocellula della fresa sia oscurata per passare allo stato successivo.

```
4: "stato_4"();
  IF ("FTC3_FRESA" = 0) THEN
    "STATO" := 5;
  END_IF;
```

Nella funzione relativa quarto stato attivo il secondo nastro finché la fotocellula della fresa non viene oscurata.

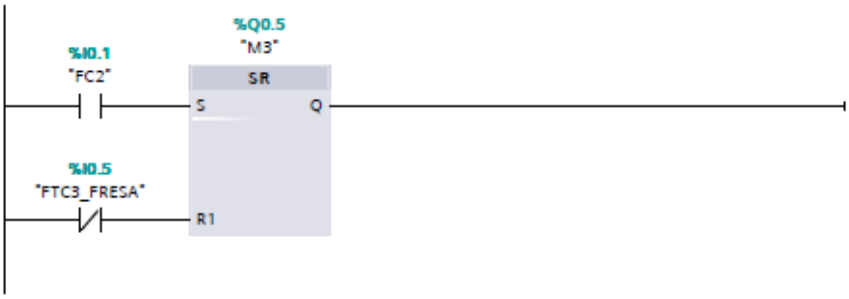


Fig.149A: Attivazione del secondo nastro fino all'oscuramento della fotocellula.

Nello stato 5 eseguo la relativa funzione e attendo che il timer relativo alla fresa finisca il conteggio per passare allo stato successivo.

```
5: "stato_5"();
  IF "temp_fresa".Q THEN
    "STATO" := 6;
  END_IF;
```

Nella funzione relativa al quinto stato se la fotocellula è attiva faccio scattare il timer che temporizza la fresa.

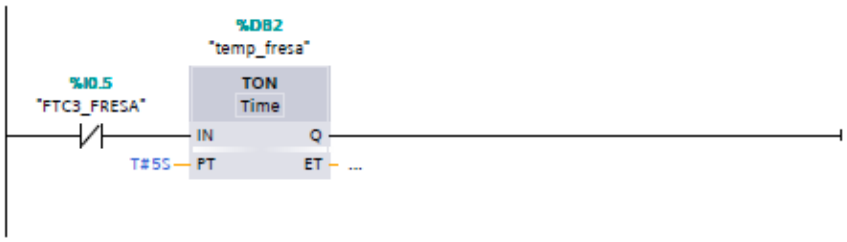


Fig.149B: Il timer scatta all'attivazione della fotocellula.

Finché il timer non scatta la fresa continua a funzionare.

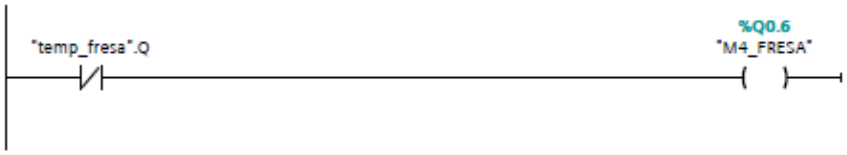


Fig.150: Funzionamento continuo della fresa.

Nello stato 6 eseguo la relativa funzione e attendo che la fotocellula del trapano sia oscurata per passare allo stato successivo.

```
6: "stato_6"();
  IF ("FTC4_TRAPANO" = 0) THEN
    "STATO" := 7;
  END_IF;
```

Nella funzione relativa al sesto stato avvio il secondo e il terzo nastro finché la fotocellula del trapano non scatta.

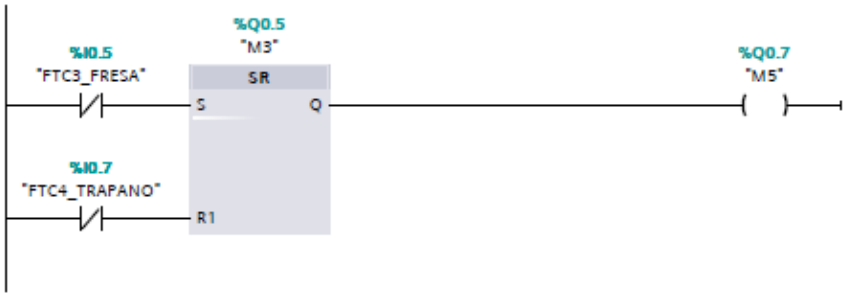


Fig.151: Avvio del secondo e terzo nastro.

Nello stato 7 eseguo la relativa funzione e attendo che il timer relativo al trapano finisca il conteggio per passare allo stato successivo.

```
7: "stato_7"();
  IF "temp_trapano".Q THEN
    "STATO" := 8;
  END_IF;
```

Nella funzione relativa al settimo stato attivo il timer relativo alla temporizzazione del trapano nel momento in cui la fotocellula viene oscurata.

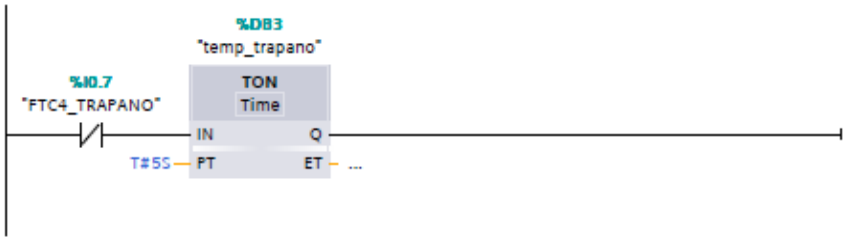


Fig.152: Attivazione Timer del trapano.

Finché il conteggio continua il trapano gira.

Fig.153: Funzionamento continuo del trapano.

Nello stato 8 eseguo la relativa funzione e attendo che la flag di fine stato si alzi per passare allo stato successivo.

```
8: "stato_8"();
  IF "flag_fine_stato_2" THEN
    "flag_fine_stato_2" := 0;
    "STATO" := 9;
  END_IF;
```

Nella funzione relativa all'ottavo stato attivo l'ultimo nastro finché non scatta il timer.

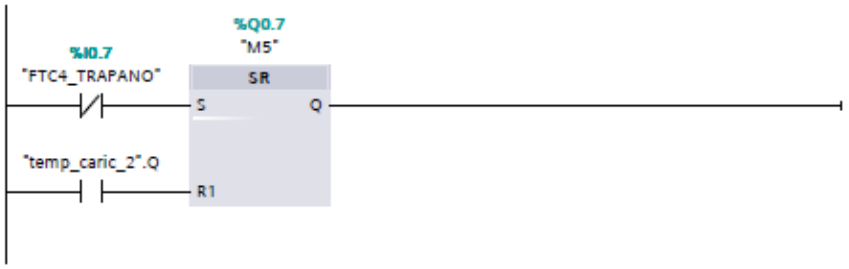


Fig.154: Attivazione ultimo nastro.

La flag di passaggio abilita il timer che agisce su un merker.

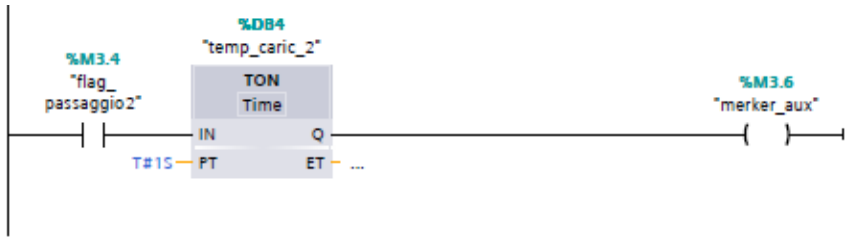


Fig.155: Abilitazione del timer attraverso la flag.

Il merker abilita una flag di fine stato.

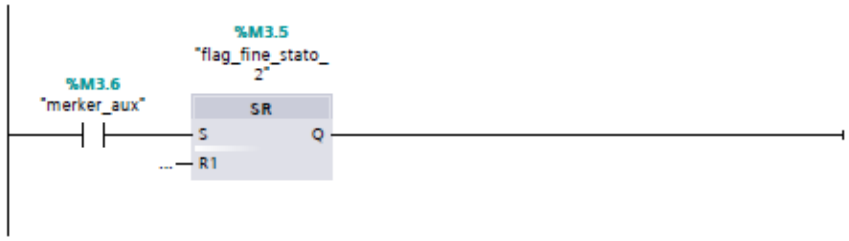


Fig.156: Abilitazione della flag da parte del merker.

La flag di passaggio memorizza il passaggio dalla fotocellula del trapano.

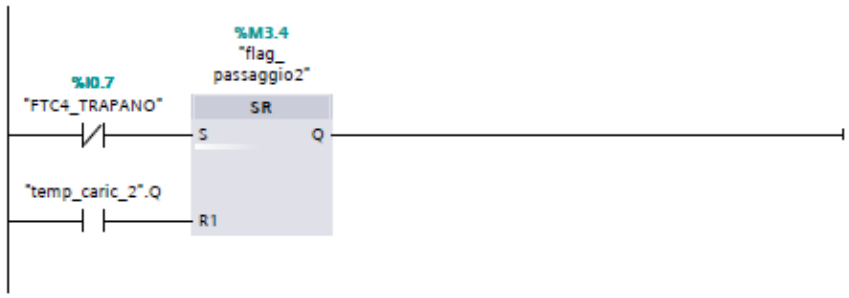


Fig.157: La flag di passaggio memorizza il passaggio per la fotocellula del trapano.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Nello stato 9 eseguo la relativa funzione e attendo che FC4 scatti per passare allo stato successivo.

```
9: "stato_9"();
  IF "FC4" THEN
    "STATO" := 10;
  END_IF;
```

Nella funzione relativa al nono stato faccio andare avanti il motore fino al finecorsa anteriore.

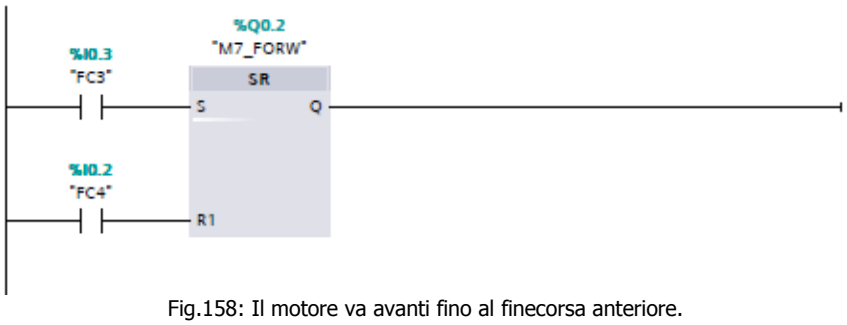


Fig.158: Il motore va avanti fino al finecorsa anteriore.

Nello stato 10 eseguo la relativa funzione e attendo che FC3 scatti per passare allo stato successivo.

```
10: "stato_10"();
  IF "FC3" THEN
    "STATO" := 11;
  END_IF;
```

Nella funzione relativa al decimo stato faccio retrocedere il motore fino al finecorsa posteriore.

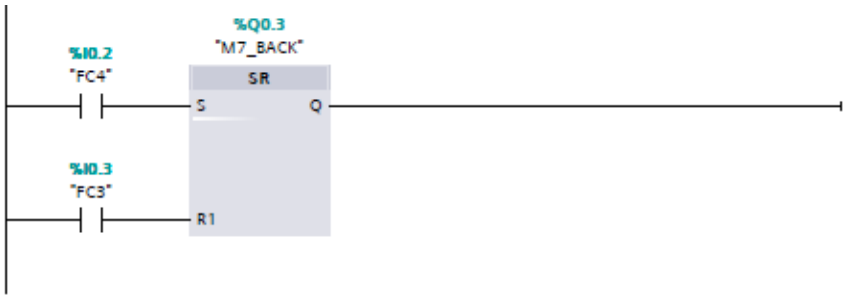


Fig.159: Il motore retrocede sino al finecorsa posteriore.

Nello stato 11 eseguo la relativa funzione e attendo che il timer relativo allo scarico del pezzo finisca il conteggio per passare allo stato successivo.

```
11: "stato_11"();
  IF "temp_scarico".Q THEN
    "STATO" := 12;
  END_IF;
```

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Nella funzione relativa all'undicesimo stato attivo un timer che temporizza lo scarico nel momento in cui il finecorsa viene premuto.

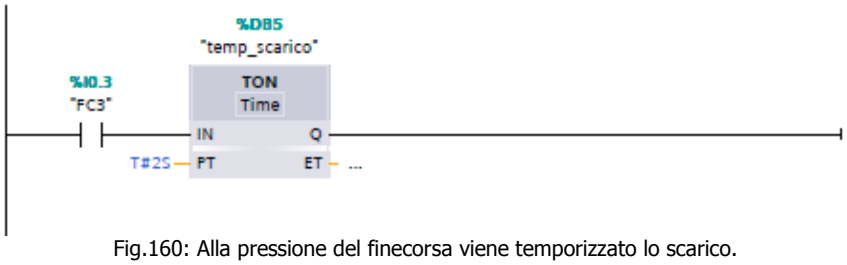


Fig.160: Alla pressione del finecorsa viene temporizzato lo scarico.

Attivo il nastro finché il temporizzatore non scatta.

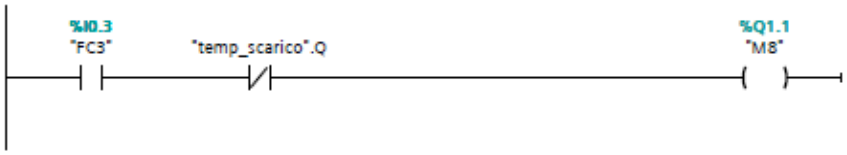


Fig.161: Attivazione continua del nastro fino a un certo evento.

Nell'ultimo stato resetto tutte le variabili e i conteggi per poi tornare allo stato iniziale.

```
12: "flag_passaggio" := 0;
    "flag_fine_stato" := 0;
    "m_prec" := 0;
    "m_prec2" := 0;
    "flag_passaggio2" := 0;
    "flag_fine_stato_2" := 0;
    "merker_aux" := 0;
    "temp_caric_1".ET := 0;
    "temp_caric_2".ET := 0;
    "temp_fresa".ET := 0;
    "temp_scarico".ET := 0;
    "temp_trapano".ET := 0;
    "STATO" := 0;
```

Considero eventuali valori differenti che mi riportano allo stato precedente.

```
ELSE
  "STATO" := 0;

END_CASE;
```

L'Scl è il linguaggio strutturato di Siemens (ST per la norma 61131-3), e viene utilizzato per la programmazione dei PLC .
E' un linguaggio molto simile al Pascal, di alto livello e indicato per svolgere compiti di automazione complessi come:

- La gestione dei dati;
- L'ottimizzazione del processo;

- La gestione delle ricette;
- Lo svolgimento di compiti matematici e statistici;

FUNZIONE SCL: Case

Descrizione:
L'istruzione "Diramazione multipla" consente di elaborare una tra diverse sequenze di istruzioni in funzione del valore di un'espressione numerica.
Il valore dell'istruzione deve essere un numero intero. Durante l'esecuzione dell'istruzione il valore dell'espressione viene confrontato con quelli di diverse costanti. Se il valore dell'espressione corrisponde a quello di una costante, vengono eseguite le istruzioni programmate direttamente dopo questa costante. Le costanti possono assumere i valori seguenti:

- Un numero intero (ad es. 5)
- Un campo di numeri interi (ad es. 15..20)
- Un'enumerazione di numeri interi e campi (ad es. 10,11,15..20)

Sintassi:

Per l'istruzione "Diramazione multipla" viene utilizzata la seguente sintassi:

CASE <Espressione> OF
<Costante1>: <Istruzioni1>
<Costante2>: <Istruzioni2>
<CostanteX>: <IstruzioniX>; // X >=3
ELSE <Istruzioni0>
END_CASE

La sintassi dell'istruzione è costituita dalle seguenti parti:

Parte/parametro	Tipo di dati	Descrizione
<Espressione>	Numeri interi	Valore che viene confrontato con i valori delle costanti programmati.
<Costante>	Numeri interi	Valori delle costanti che costituiscono la condizione per eseguire una sequenza di istruzioni. Le costanti possono assumere i valori seguenti: • Un numero intero (ad es. 5) • Un campo di numeri interi (ad es. 15..20) • Un'enumerazione di numeri interi e campi (ad es. 10,11,15..20)
<Istruzione>	-	Istruzioni a scelta che vengono eseguite se il valore dell'espressione corrisponde al valore di una costante. Fanno eccezione le istruzioni programmate dopo ELSE, che vengono eseguite se i valori non coincidono.

Per maggiori informazioni sui tipi di dati validi consultare la sezione "Vedere anche".
Se il valore dell'espressione corrisponde a quello della prima costante (<Costante1>), vengono eseguite le istruzioni (<Istruzioni1>) programmate direttamente dopo la prima costante. Successivamente l'elaborazione del programma prosegue dopo END_CASE.
Se il valore dell'espressione non corrisponde a quello della prima costante (<Costante1>), esso viene confrontato con il valore della costante programmata successiva. In questo modo l'istruzione CASE viene eseguita fino a trovare una corrispondenza tra i valori. Se il valore dell'espressione non corrisponde a nessuno dei valori delle costanti programmati, vengono eseguite le istruzioni (<Istruzioni0>) programmate dopo ELSE. ELSE è una parte opzionale della sintassi e può essere eliminato.
L'istruzione CASE può anche essere annidata sostituendo un blocco dell'istruzione con CASE . END_CASE segna la fine dell'istruzione CASE.

Esempio:
Il seguente esempio mostra il funzionamento dell'istruzione:

SCL

```
CASE "Tag_Value" OF
0 : "Tag_1" := 1;
1,3,5 : "Tag_2" :=1;
6..10 : "Tag_3" := 1;
16,17,20..25 : "Tag_4" := 1;
ELSE "Tag_5" := 1;
END_CASE;
```

La seguente tabella mostra il funzionamento dell'istruzione in base a valori di operandi concreti:

Operando	**Valori**				
Tag_Value	0	1, 3, 5	6, 7, 8, 9, 10	16, 17, 20, 21, 22, 23, 24, 25	2
Tag_1	1	-	-	-	-
Tag_2	-	1	-	-	-
Tag_3	-	-	1	-	-
Tag_4	-	-	-	1	-
Tag_5	-	-	-	-	1
1: l'operando viene impostato allo stato di segnale "1". -: lo stato di segnale dell'operando rimane invariato.					

Operazioni di Confronto, IN_RANGE

esperienza 15

Gridella Andrea 5AET 2013
(gridella.andrea@libero.it)

Con il blocco istruzione "Valore compreso nel campo", piu' semplicemente chiamato IN_RANGE, si ha la possibilità di determinare se un certo valore posto all'ingresso VAL si trovi entro un campo di valori delimitato da altri due valori posti sugli ingressi MAX e MIN. Questo blocco istruzione confronta il dato posto all'ingresso VAL con gli altri due dati corrispondenti agli ingressi MAX e MIN, scrivendo poi il risultato nell'uscita del box. Se si ha come risultato MIN ≤ VAL ≤ MAX allora si ottiene in uscita il segnale a livello logico "1", altrimenti se VAL < MIN o VAL > MAX il livello logico del segnale in uscita è "0".
Un aspetto molto importante per la corretta funzionalità del blocco e dell'intero sistema interessa i valori da confrontare, infatti questi devono avere lo stesso tipo di dati e le possibili famiglie di dati da utilizzare negli ingressi precedentemente descritti sono le seguenti:

Parametro	Dichiarazione	Tipo di dati	Area di memoria	Descrizione
Ingresso del box	Input	BOOL	I, Q, M, D, L	Risultato della combinaizone logica precedente
MIN	Input	Numeri interi, numeri in virgola mobile	I, Q, M, D, L, o costante	Limite inferiore del campo di valori
VAL	Input	Numeri interi, numeri in virgola mobile	I, Q, M, D, L, o costante	Valore di confronto
MAX	Input	Numeri interi, numeri in virgola mobile	I, Q, M, D, L, o costante	Limite superiore del campo di valori
Uscita del box	Output	BOOL	I, Q, M, D, L	Risultato del confronto

Un semplice esempio che possa rendere il concetto piu' chiaro è il seguente:

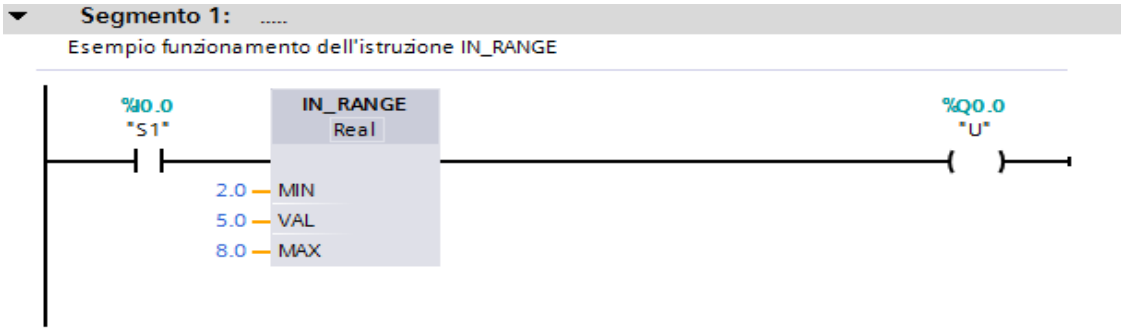


Fig.167 :Esempio che spiega il funzionamento blocco IN_RANGE(operazione di confronto)

In questo programma alla pressione di S1 si attiva il blocco istruzione IN_RANGE, il quale fa il confronto tra i valori posti agli ingressi MIN, MAX e VAL . Da come si può vedere i valori dati rispettano il caso MIN ≤ VAL ≤ MAX, perciò viene fornito in uscita lo stato segnale "1" che fa accendere la luce "U".

Operazioni di Confronto, OUT_RANGE

esperienza 16

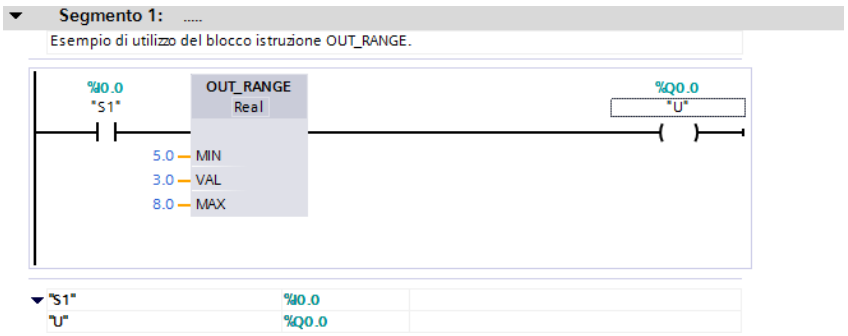
Gridella Andrea 5AET 2013
(gridella.andrea@libero.it)

Il blocco istruzione "Valore fuori campo" permette di controllare se un valore posto all'ingresso VAL si trova fuori da un determinato campo delimitato dagli altri due ingressi MIN e MAX. Il funzionamento di questo blocco consiste nel confrontare il valore corrispondente all'ingresso VAL con quelli degli altri due ingressi-limiti, andando poi a scrivere il risultato nell'uscita del box. Se il valore all'ingresso VAL soddisfa il confronto MIN > VAL o VAL > MAX all'uscita si ha il segnale a livello logico "1", al contrario in uscita si otterrebbe un segnale a livello logico "0"(in questo caso l'istruzione OUT_RANGE non viene elaborata).
I valori agli ingressi MIN,VAL e MAX devono avere dati dello stesso tipo e le varie tipologie di dato(con le aree si memoria corrispondenti) da poter attribuire ai 3 ingressi sono le seguenti :

Parametro	Dichiarazione	Tipo di dati	Area di memoria	Descrizione
Ingresso del box	Input	BOOL	I, Q, M, D, L	Risultato della combinaizone logica precedente
MIN	Input	Numeri interi, numeri in virgola mobile	I, Q, M, D, L, o costante	Limite inferiore del campo di valori
VAL	Input	Numeri interi, numeri in virgola mobile	I, Q, M, D, L, o costante	Valore di confronto
MAX	Input	Numeri interi, numeri in virgola mobile	I, Q, M, D, L, o costante	Limite superiore del campo di valori
Uscita del box	Output	BOOL	I, Q, M, D, L	Risultato del confronto

Per rendere piu' chiaro il funzionamento di questo blocco istruzione è mostrato qui di seguito un esempio:

Fig.168 :esercizio con blocco OUT_RANGE(Operazione di confronto)



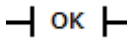
In questo esempio alla pressione dello switch S1 si attiva il blocco istruzione OUT_RANGE, il quale fa il confronto tra i valori messi agli ingressi MIN, VAL e MAX (numeri reali). In questo esercizio il valore all'ingresso VAL (3.0) soddisfa il confronto MIN (5.0) > VAL e perciò in uscita passa un segnale a livello logico alto che va ad accendere l'uscita "U".

Operazioni di Confronto, OK

esperienza 17

Gridella Andrea 5AET 2013
(gridella.andrea@libero.it)

Con l'istruzione OK o anche chiamata "Verifica validità" permette all'utente di verificare se un numero inserito sia o no un numero in virgola mobile valido. Il controllo, se all'ingresso dell'istruzione vi è un segnale a livello logico alto (1), avviene ad ogni ciclo del sistema in modo da controllare in ogni momento i valori (questo perché l'utente potrebbe inserire valori in qualsiasi periodo di funzionamento del sistema).
L'uscita dell'istruzione è a livello logico alto solo se dal controllo è stata rilevata la validità del valore in virgola mobile e se il segnale all'ingresso dell'istruzione è a 1. In tutti gli altri casi invece il funzionamento on avviene in modo corretto e quindi l'uscita dell'istruzione è a 0. Il simbolo dell'istruzione OK è il seguente:



L'utilizzo di questa istruzione può anche essere associato a quello di altre istruzioni infatti, se si collega l'istruzione OK a qualsiasi meccanismo con ingresso EN, quest'ultimo viene attivato solo se il controllo verifica la validità del valore in virgola mobile.
Di seguito è mostrato un semplice esempio per capire meglio il funzionamento di tale istruzione:

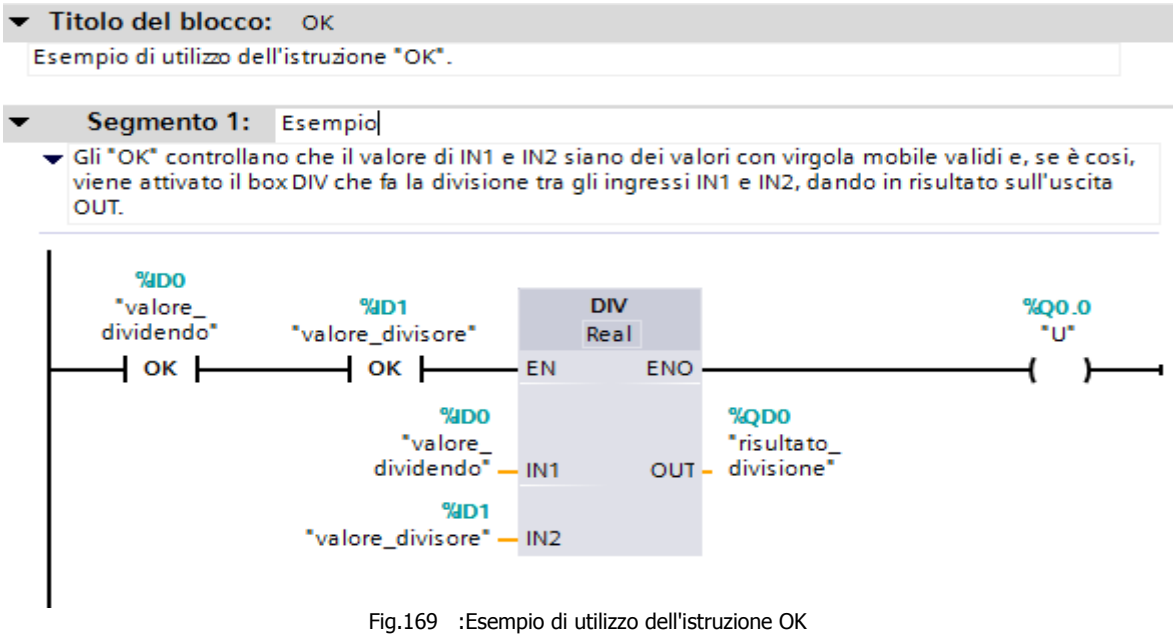


Fig.169 :Esempio di utilizzo dell'istruzione OK

In questo semplice esempio i due blocchi istruzione "OK" che sono all'ingresso EN della funzione DIV vanno a controllare i valori posti agli ingressi IN1 e IN2. Se con tale controllo non si riscontrano errori viene attivata l'istruzione DIV, la quale fa la divisione tra i valori agli ingressi IN1 e IN2, rendendo poi leggibile il risultato della divisione sull'uscita OUT dell'istruzione DIV. Se il funzionamento è avvenuto senza errori all'uscita OUT vi è un segnale a livello logico alto (1) che "accende" l'uscita "U".

Operazioni di confronto, NOT OK

esperienza 18
Montemaggi Pablo 5AET 2013
(pablo.montemaggi@gmail.com)

L'istruzione NOT_OK, o verifica invalidità, controlla se il valore di un operando è un numero in virgola mobile valido. A ogni ciclo di programma viene avviata l'interrogazione se lo stato del segnale dell'ingresso dell'istruzione è 1.

L'uscita dell'istruzione ha lo stato del segnale 1 solamente se al momento dell'interrogazione l'operando è un numero in virgola mobile non valido e l'ingresso dell'istruzione presenta un valore di segnale uguale a 1.

L'esempio seguente mostra come è possibile utilizzare la funzione NOT_OK.

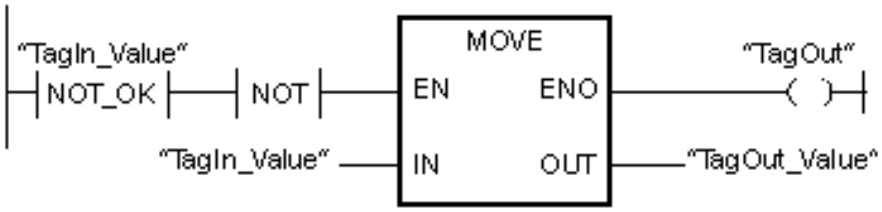


Fig.170 :Esempio di utilizzo dell'istruzione NOT_OK

Se il valore dell'operando "TagIn_Value" è un numero in virgola mobile non valido, l'istruzione MOVE non viene eseguita in quanto è presente il NOT.
L'uscita "TagOut" viene resettata sullo stato del segnale 0.

Funzioni Matematiche, ADD

esperienza 19
Della Chiesa Enrico 5AET 2013
(enrico.dellachiesa@gmail.com)

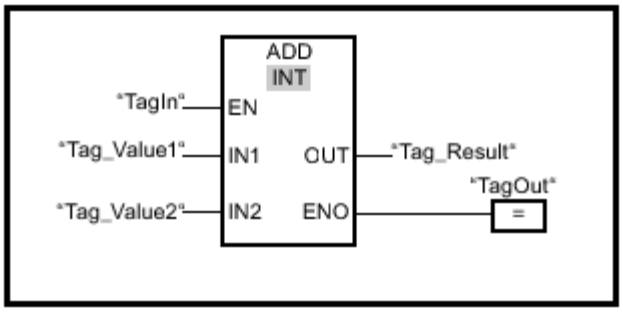


Fig.171 :Blocco ADD(funzione matematica)

L'istruzione "Somma" addiziona il valore dell'ingresso IN1 con quello dell'ingresso IN2 e legge la somma nell'uscita OUT ((OUT = IN1+IN2)).

Il box dell'istruzione contiene nello stato di base almeno due ingressi (IN1 e IN2). Il numero degli ingressi è ampliabile. Gli ingressi inseriti vengono numerati in ordine crescente nel box. Durante l'esecuzione dell'istruzione vengono sommati i valori di tutti i parametri degli ingressi disponibili. La somma viene salvata nell'uscita OUT.

L'istruzione viene eseguita solo se lo stato di segnale nell'ingresso di abilitazione EN è "1". Se l'istruzione viene eseguita senza errori, anche l'uscita ENO fornisce lo stato di segnale "1".

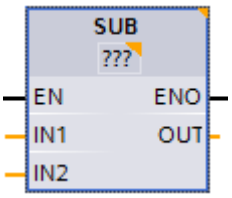
L'uscita di abilitazione ENO fornisce lo stato di segnale "0" se viene soddisfatta una delle seguenti condizioni:

- L'ingresso EN fornisce lo stato di segnale "0".
- Il risultato dell'istruzione non è compreso nel campo ammesso per il tipo di dati indicato nell'uscita OUT.
- Un numero in virgola mobile ha un valore non valido.

Funzioni Matematiche, SUB

esperienza 20

Gridella Andrea 5AET 2013
(gridella.andrea@libero.it)



Con il blocco istruzione SUB (sottrazione) si ha la possibilità di fare una sottrazione, sottraendo il valore dell'ingresso IN2 da quello dell'ingresso IN1 e di leggere il risultato nell'uscita OUT. L'attivazione del blocco SUB e il suo conseguente funzionamento può avvenire solo se nell'ingresso di abilitazione EN arriva un segnale a livello logico alto ("1") e, se non vi è alcun errore durante il funzionamento, anche l'uscita ENO è a livello logico alto.I valori attribuiti agli ingressi devono avere la stessa tipologia di dato dell'uscita OUT e nella tabella qui di seguito sono mostrati i tipi di dato che è possibile utilizzare:

Parametro	**Dichiarazione**	**Tipo di dati**	**Area di memoria**	**Descrizione**
EN	Input	BOOL	I, Q, M, D, L	Ingresso abilitazione
ENO	Output	BOOL	I, Q, M, D, L	Uscita abilitazione
IN1	Input	Numeri interi, numeri in virgola mobile	I, Q, M, D, L, o costante	Minuendo
IN2	Input	Numeri interi, numeri in virgola mobile	I, Q, M, D, L, o costante	Sottraendo
OUT	Output	Numeri interi, numeri in virgola mobile	I, Q, M, D, L	Differenza

Per chiarire il concetto appena spiegato qui di seguito vi è un esempio che mostra il funzionamento dell'istruzione SUB:

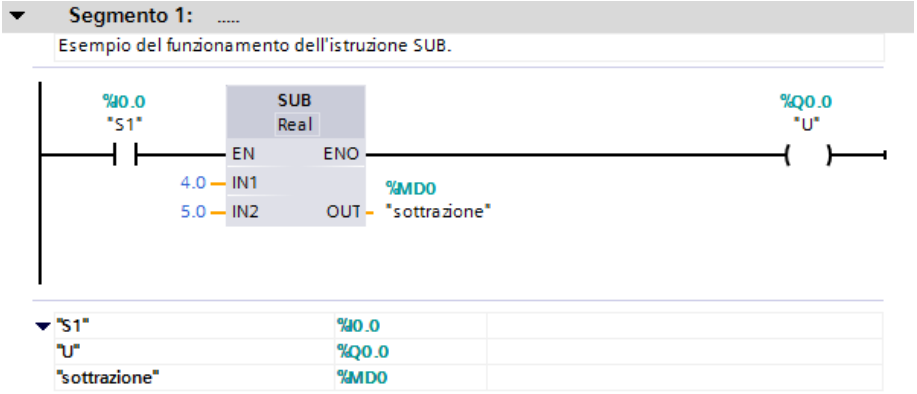


Fig.172 :Esempio che spiega il funzionamento istruzione SUB

Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

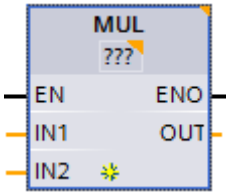
In questo semplice esempio alla pressione dello switch S1 si attiva il blocco istruzione SUB, il quale sottrae il valore di IN2 a quello di IN1, ottenendo il risultato all'uscita OUT. In questo esempio sono state utilizzate delle variabili reali, perciò in uscita si ha un valore reale. Se la sottrazione si svolge in modo corretto all'uscita ENO arriva un segnale a livello logico alto che "accende" l'uscita U.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Funzioni Matematiche, MUL

esperienza 21

Gridella Andrea 5AET 2013
(gridella.andrea@libero.it)



Con il blocco istruzione si ha la possibilità di moltiplicare il valore dell'ingresso IN1 con quello dell'ingresso IN2 e di leggere il risultato nell'uscita OUT del blocco. Si possono moltiplicare anche piu' di due valori, infatti il numero degli ingressi del blocco può essere aumentato. L'istruzione MUL funziona soltanto se all'ingresso di abilitazione EN arriva un segnale di livello logico "1" e, se durante il funzionamento non vi sono errori, anche all'uscita ENO arriva un segnale a livello logico alto.
I valori di ingresso devono avere la stessa tipologia di dato di quello di uscita e i vari tipi di dato che si possono utilizzare sono mostrati nella seguente tabella:

Parametro	**Dichiarazione**	**Tipo di dati**	**Area di memoria**	**Descrizione**
EN	Input	BOOL	I, Q, M, D, L	Ingresso abilitazione
ENO	Output	BOOL	I, Q, M, D, L	Uscita abilitazione
IN1	Input	Numeri interi, numeri in virgola mobile	I, Q, M, D, L, o costante	Moltiplicatore
IN2	Input	Numeri interi, numeri in virgola mobile	I, Q, M, D, L, o costante	Moltiplicando
INn	Input	Numeri interi, numeri in virgola mobile	I, Q, M, D, L, o costante	Valori ingresso opzionali che possono essere moltiplicati
OUT	Output	Numeri interi, numeri in virgola mobile	I, Q, M, D, L	Prodotto

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Per rendere maggiormente comprensibile questo tipo di istruzione qui di seguito è mostrato un esempio:

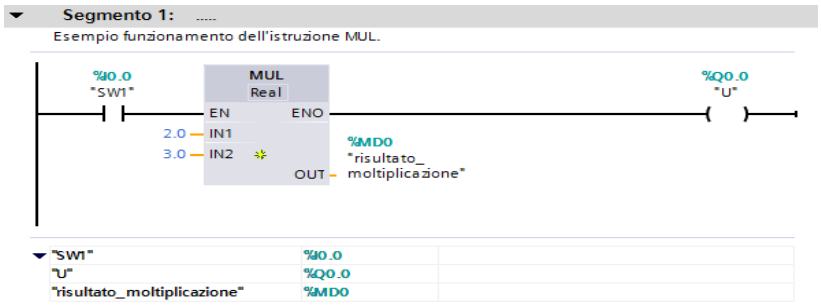


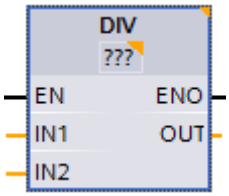
Fig.173 :Esempio che spiega il funzionamento istruzione MUL

In questo sistema alla pressione dello switch SW1 si ha l'attivazione dell'istruzione MUL, la quale moltiplica i valori posti agli ingressi IN1 (2.0) e IN2 (3.0). Il risultato della moltiplicazione si può leggere nell'uscita OUT (risultato_moltiplicazione) e, se il processo di moltiplicazione è avvenuto senza errori, all'uscita ENO arriverà un segnale a livello logico alto che attiva l'uscita ENO, "accendendo" l'uscita U.

Funzioni Matematiche, DIV

esperienza 22

Gridella Andrea 5AET 2013
(gridella.andrea@libero.it)



L'istruzione DIV o "Dividi" viene utilizzata per fare una divisione, dividendo il valore posto all'ingresso IN1 con quello all'ingresso IN2 e avendo poi la possibilità di leggere il risultato sull'uscita OUT del box. Il blocco di funzione DIV funziona solo se all'ingresso EN riceve un impulso di livello logico alto (1) e eventuali errori durante il suo funzionamento possono essere causati dalla diversità di tipologia di dati delle variabili utilizzate (le varie tipologie di dati che si possono utilizzare sono: SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal) o da un numero in virgola mobile che ha un numero non valido.
Inizialmente il blocco di istruzione DIV si presenta con due soli ingressi, nei quali non vi è scritto alcun valore:
I valori scritti agli ingressi "IN1" e "IN2" devono avere la stessa tipologia di dato della variabile che si pone all'uscita "OUT".
Per chiarire il funzionamento dell'istruzione DIV di seguito è mostrato un semplice esempio:

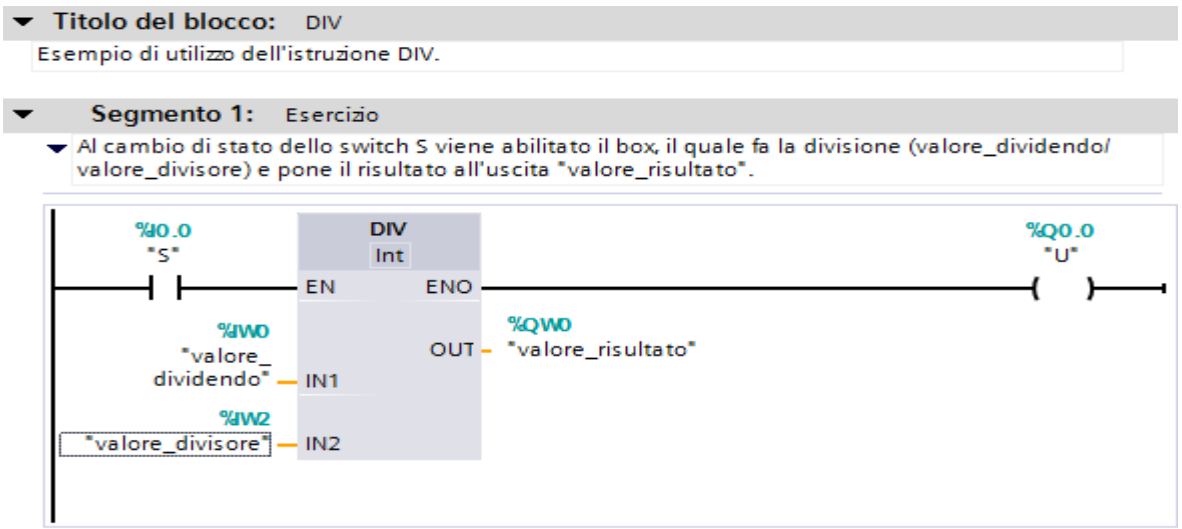


Fig.174 :Esempio che spiega il funzionamento istruzione DIV

In questo semplice esercizio con il cambio di stato da "0" a "1" dello switch "S" si attiva il box MIN, il quale rileva i segnali posti sui 2 e divide il "valore_dividendo" per il "valore_divisore", andando poi a scrivere il risultato della divisione sull'uscita "OUT" rendendolo perciò anche leggibile nella variabile "valore_risultato". Se il funzionamento avviene in modo corretto viene accesa l'uscita "U" posta sull'output ENO del blocco di funzione DIV.

Funzioni Matematiche, MOD

esperienza 23
Gridella Andrea 5AET 2013
(gridella.andrea@libero.it)

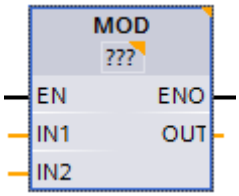


Fig.175 :Blocco MOD

L'istruzione MOD o "Rileva il resto della divisione" viene utilizzata per fare una divisione, dividendo il valore posto all'ingresso IN1 con quello all'ingresso IN2 e avendo poi la possibilità di leggere il resto della divisione fatta sull'uscita OUT del box. Il blocco di funzione MOD funziona solo se all'ingresso EN riceve un impulso di livello logico alto (1) e eventuali errori durante il suo funzionamento possono essere causati dalla diversità di tipologia di dati delle variabili utilizzate (le varie tipologie di dati che si possono utilizzare sono: SInt, Int, DInt, USInt, UInt, UDInt) o nel caso in cui lo stato del segnale all'ingresso EN è 0, caso in cui l'uscita ENO viene resettata.
Inizialmente il blocco di istruzione DIV si presenta con due soli ingressi, nei quali non vi è scritto alcun valore:

I valori scritti agli ingressi "IN1" e "IN2" devono avere la stessa tipologia di dato della variabile che si pone all'uscita "OUT".
Per chiarire il funzionamento dell'istruzione MOD di seguito è mostrato un semplice esempio:

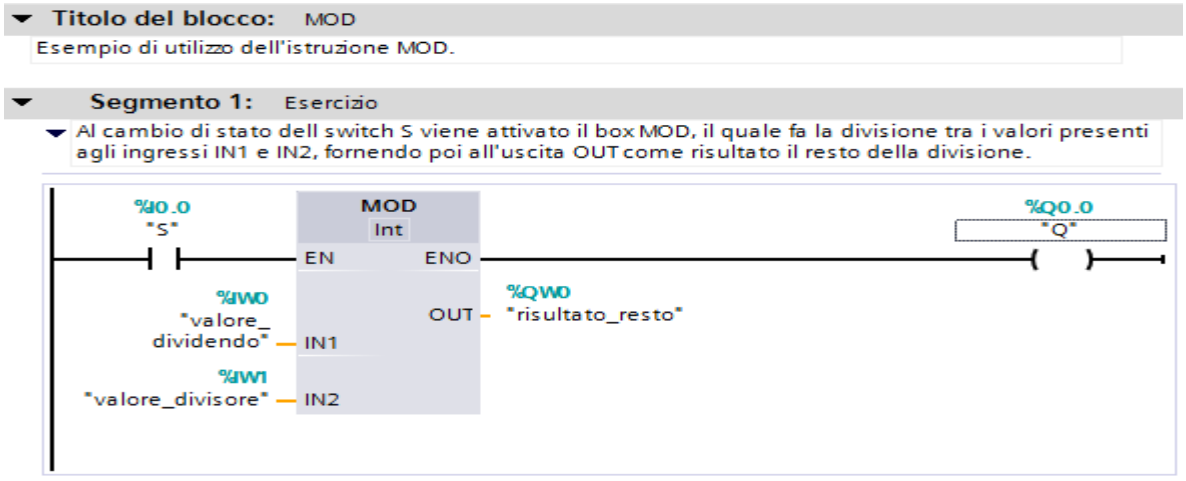


Fig.176 :Esempio che spiega il funzionamento istruzione MOD

In questo semplice esercizio con il cambio di stato da "0" a "1" dello switch "S" si attiva il box MOD, il quale rileva i segnali posti sui 2 ingressi e divide il "valore_dividendo" per il "valore_divisore", andando poi a scrivere il resto della divisione come risultato sull'uscita "OUT" rendendolo perciò anche leggibile nella variabile "valore_risultato". Se il funzionamento avviene in modo corretto viene accesa l'uscita "U" posta sull'output ENO del blocco di funzione MOD

Funzioni Matematiche, NEG

esperienza 24

Gridella Andrea 5AET 2013
(gridella.andrea@libero.it)

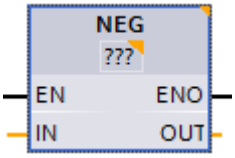


Fig.177 :Blocco NEG

L'istruzione NEG o "Crea complemento a due" permette all'utente di poter cambiare il segno di un valore ponendo questo all'ingresso IN del box e andando a leggere il valore con segno opposto all'uscita OUT del blocco di istruzione. Questo perciò significa che se mettiamo all'ingresso in un valore con segno negativo otterremo all'uscita OUT lo stesso valore con segno però decisivo, e viceversa. Come in molte altre istruzioni descritte precedentemente il blocco di istruzione è in grado di eseguire il proprio compito solo se all'ingresso EN gli arriva un segnale con livello logico alto (1). Se il funzionamento avviene in modo corretto all'uscita OUT si avrà un segnale a livello logico alto.
Il blocco di istruzione NEG è il seguente:
Durante il funzionamento potrebbero esserci problemi derivanti da un numero in virgola mobile che ha un valore non valido oppure da una diversa tipologia di dati tra ingresso ed uscita. Questi problemi causano la non "accensione" dell'uscita ENO.
Vi sono varie tipologie di dato che è possibile utilizzare (Int, DInt, SInt, Real, LReal) e, come già detto prima, se la tipologia di dato dell'ingresso è diversa da quella dell'uscita si ha un'errore nel funzionamento.
Per comprendere meglio il funzionamento dell'istruzione NEG qui di seguito vi è un semplice esempio:

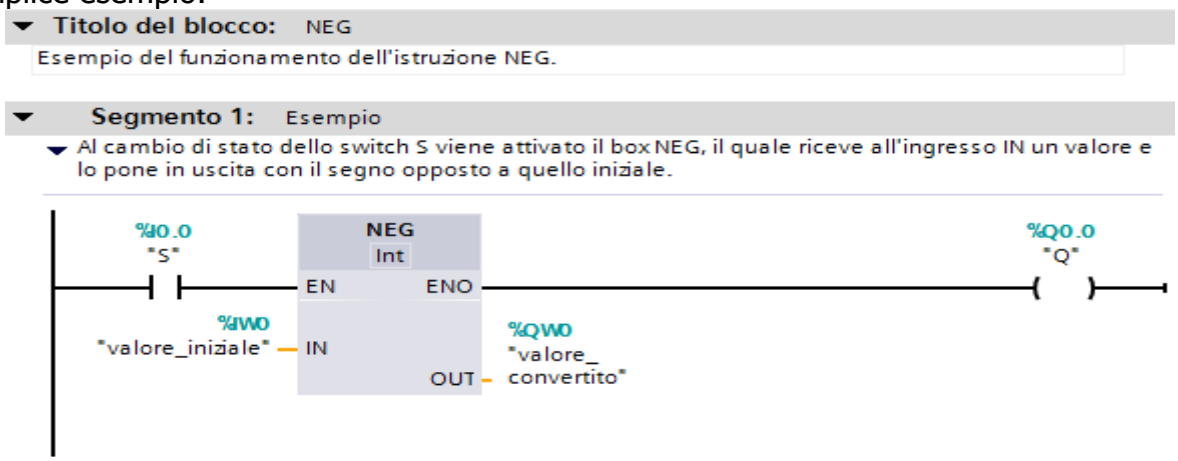


Fig.178 :Esempio che spiega il funzionamento dell'istruzione NEG

In questo semplice esempio alla pressione dello switch "S" si attiva il box NEG, il quale riceve all'ingresso IN un valore rappresentato dalla variabile "valore_iniziale" e lo converte in uno stesso valore con segno però opposto denominato come "valore_convertito".

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Funzioni Matematiche, INC

esperienza 25

Gridella Andrea 5AET 2013
(gridella.andrea@libero.it)

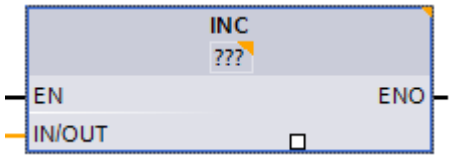


Fig.179 :Blocco INC

L'istruzione INC o "Incrementa" permette all'utente di poter inserire all'ingresso IN/OUT un valore e di ottenere il successivo valore superiore avendo poi l'opportunità di leggere il risultato ottenuto. Come in molte altre istruzioni descritte precedentemente il blocco di istruzione è in grado di eseguire il proprio compito solo se all'ingresso EN gli arriva un segnale con livello logico alto (1). Se il funzionamento avviene in modo corretto all'uscita OUT si avrà un segnale a livello logico alto. Il blocco di istruzione INC è il seguente:

Durante il funzionamento potrebbero esserci problemi derivanti da una diversa tipologia di dati tra ingresso ed uscita. Questi problemi causano la non "accensione" dell'uscita ENO. Vi sono varie tipologie di dato che è possibile utilizzare (Int, DInt, SInt, UDInt, USInt, UInt) e, come già detto prima, se la tipologia di dato dell'ingresso è diversa da quella dell'uscita si ha un'errore nel funzionamento.
Per comprendere meglio il funzionamento dell'istruzione INC qui di seguito vi è un semplice esempio:

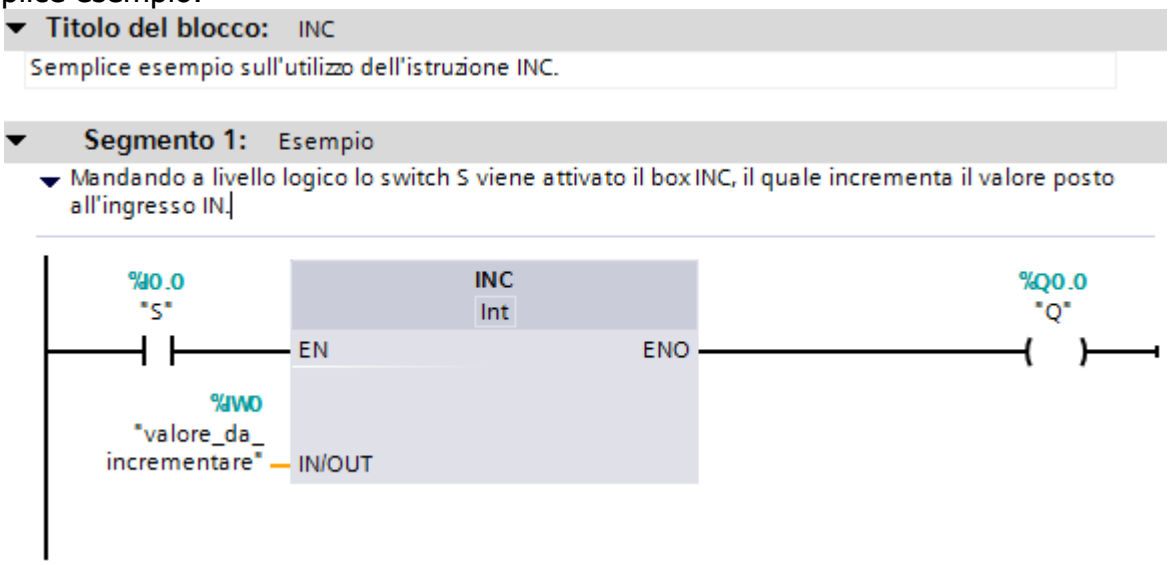


Fig.180 :Esempio che spiega il funzionamento istruzione INC

In questo semplice esempio alla pressione dello switch "S" si attiva il box INC, il quale riceve all'ingresso IN un valore rappresentato dalla variabile "valore_da_incrementare" e lo incrementa poi al valore successivo, rendendo poi il valore ottenuto leggibile all'utente. Col funzionamento del box INC si "accende" anche l'uscita Q collegata all'output OUT.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Funzioni Matematiche, ABS

esperienza 26

Gridella Andrea 5AET 2013
(gridella.andrea@libero.it)

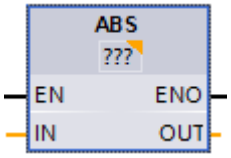


Fig.181 :Blocco ABS

Con l'istruzione ABS o "Genera valore assoluto" si ha la possibilità di rilevare e calcolare la grandezza assoluta di un valore inviato all'ingresso IN, ottenendo poi il risultato dell'operazione sull'uscita OUT in modo che possa essere facilmente letto. Il blocco di funzione esegue tale funzione solamente se viene inviato un impulso a livello logico alto (1) all'ingresso EN dell"istruzione. Perciò nel caso in cui ciò non avvenisse l'istruzione ABS è inattiva. Inoltre si potrebbe avere un funzionamento scorretto del blocco ABS nel caso in cui un numero con virgola mobile abbia un numero non valido.

Come già detto all'ingresso IN dell'istruzione viene inserito un valore che, dopo essere stato calcolato, verrà poi posto all'uscita OUT in modo che possa essere letto. Il valore posto all'ingresso IN deve essere della stessa tipologia della variabile posta all'uscita OUT del blocco di istruzione e i vari tipi di dati che si possono utilizzare sono: Real, SInt, Int, DInt, LReal.
Di seguito è mostrato un semplice esempio per comprendere meglio il funzionamento dell'istruzione in oggetto:

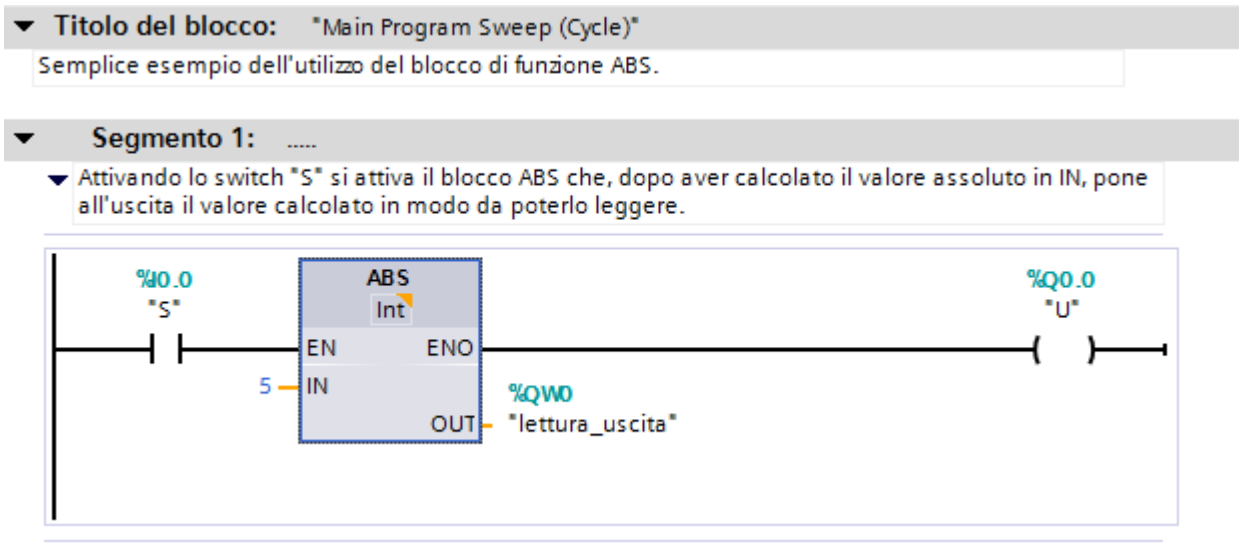


Fig.182 :Esempio che spiega il funzionamento dell' istruzione ABS

In questo esempio è stato posto lo switch "S" all'ingresso EN dell'istruzione ABS per attivare quest'ultima, abilitando il blocco di funzione e andando ad "accendere" l'uscita "U" posta sull'output ENO. Inoltre all'ingresso IN è stato messo il valore intero "5" che, dopo essere stato riconosciuto e calcolato dal blocco di funzione, verrà poi reso leggibile dalla variabile "lettura_uscita" posta appunto sull'OUT dell'istruzione ABS.

Funzioni Matematiche, MIN

esperienza 27
Gridella Andrea 5AET 2013
(gridella.andrea@libero.it)

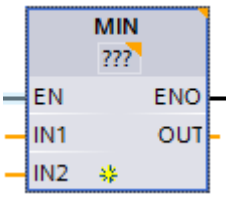


Fig.183 :Blocco MIN

L'istruzione MIN o "Rileva valore minimo" viene utilizzata per rilevare degli ingressi e trovare tra loro quello con valore minimo, andandolo poi a scrivere all'uscita OUT in modo che possa anche essere letto facilmente. Il numero degli ingressi può essere aumentato, arrivando ad un massimo di 100 ingressi (numerati in modo crescente nel blocco di istruzione). Il blocco di funzione MIN funziona solo se all'ingresso EN riceve un impulso di livello logico alto (1) e eventuali errori durante il suo funzionamento possono essere causati dalla diversità di tipologia di dati delle variabili utilizzate (le varie tipologie di dati che si possono utilizzare sono: SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal) o da un numero in virgola mobile che ha un numero non valido.
Inizialmente il blocco di istruzione MIN si presenta con due soli ingressi, nei quali non vi è scritto alcun valore:
I valori scritti agli ingressi "IN1" e "IN2" devono avere la stessa tipologia di dato della variabile che si pone all'uscita "OUT". Inoltre il numero di ingressi, come già detto precedentemente, può essere aumentato premendo i tasto destro del mouse sul blocco e andando in "Inserisci ingresso".
Per chiarire il funzionamento dell'istruzione MIN di seguito è mostrato un semplice esempio:

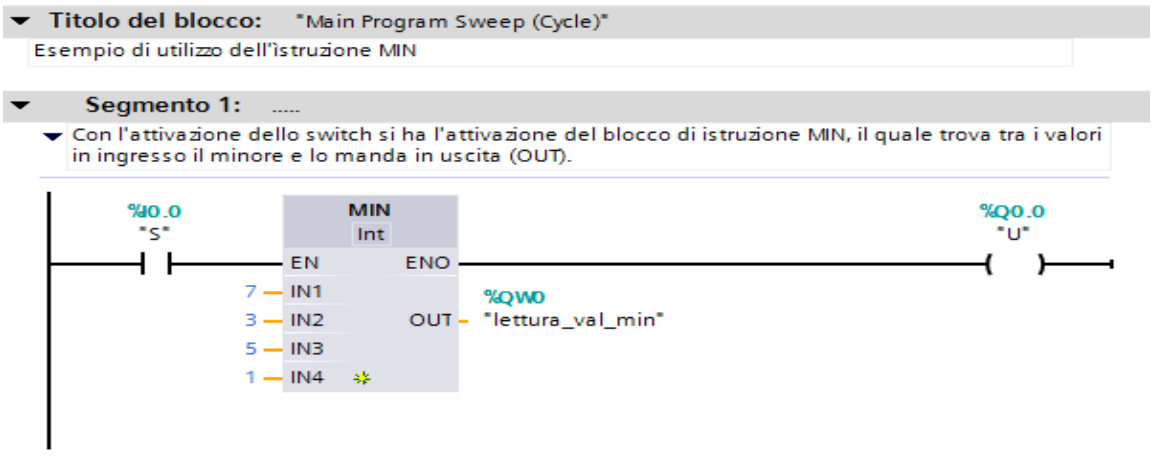
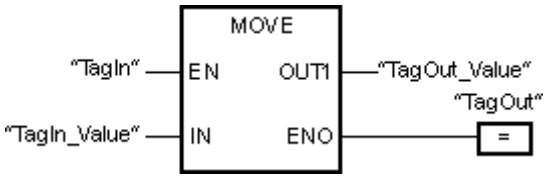


Fig.184 :Esercizio con l'istruzione MIN

In questo semplice esercizio con il cambio di stato da "0" a "1" dello switch "S" si attiva il box MIN, il quale rileva i segnali posti sui 4 ingressi e trova il minore tra loro, andandolo a scrivere poi sull'uscita "OUT" rendendolo perciò anche leggibile. Se il funzionamento avviene in modo corretto viene accesa l'uscita "U" posta sull'output ENO del blocco di funzione MIN.

Operazioni di Trasferimento, MOVE BLK

Fig.185 :Esempio Ladder dell'istruzione MOVE BLK



Questa istruzione consente di copiare il valore dell'operando "TagIn_Value" in "TagOut_Value". L'istruzione viene eseguita solo se il valore di ingresso in "EN" è 1, in questo caso anche l'uscita "ENO" si porta alta.

I tipi di operandi che sono consentiti in ingresso e in uscita sono elencati in questa tabella: Quest'altra tabella invece mostra i parametri attribuiti agli ingressi e le uscite del blocco:

Sorgente (IN)	**Destinazione (OUT1)**
BYTE	BYTE, WORD, DWORD, INT, DINT, TIME, DATE, TOD, CHAR
WORD	BYTE, WORD, DWORD, INT, DINT, TIME, S5TIME, DATE, TOD, CHAR
DWORD	BYTE, WORD, DWORD, INT, DINT, REAL, TIME, DATE, TOD, CHAR
INT	BYTE, WORD, DWORD, INT, DINT, TIME, DATE, TOD
DINT	
REAL	DWORD, REAL
TIME	BYTE, WORD, DWORD, INT, DINT, TIME
S5TIME	WORD, S5TIME
DATE	BYTE, WORD, DWORD, INT, DINT, DATE
TOD	BYTE, WORD, DWORD, INT, DINT, TOD
CHAR	BYTE, WORD, DWORD, CHAR

Parametri

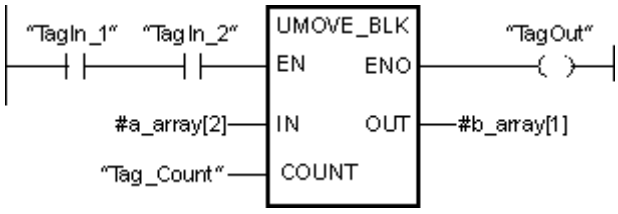
La seguente tabella mostra i parametri dell'istruzione "Copia area":

Parametro	**Dichiarazione**	**Tipo di dati**	**Area di memoria**	**Descrizione**
EN	Input	BOOL	I, Q, M, T, C, D, L	Ingressi di abilitazione
ENO	Output	BOOL	I, Q, M, D, L	Uscita di abilitazione
IN	Input	Stringhe di bit, numeri interi, numeri in virgola mobile, temporizzatori, DATE, TOD, CHAR	I, Q, M, D, L, o costante	Valori sorgente
OUT1	Output	Stringhe di bit, numeri interi, numeri in virgola mobile, temporizzatori, DATE, TOD, CHAR	I, Q, M, D, L	Indirizzi di destinazione

Operazioni di Trasferimento, UMOVE BLK

esperienza 29
Della Chiesa Enrico 5AET 2013
(enrico.dellachiesa@gmail.com)

Fig.186 :Esempio Ladder dell'istruzione UMOVE BLK



L'istruzione UMOVE_BLK consente di copiare un area di memoria collegata all'ingresso "IN" ad un'altra in uscita "OUT" senza interruzioni dovute al programma o ai cicli macchina.

Per questo ci potrebbero essere dei ritardi di eventuali allarmi della CPU.
Il numero di valori da copiare è definito a seconda del valore di ingresso di "COUNT".
La copia avviene solo se l'ingresso "EN" è alto, se il dato viene copiato correttamente anche l'uscita "ENO" va a 1.

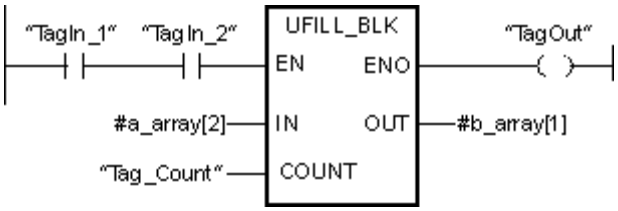
Questa tabella mostra le caratteristiche delle entrate/uscite del blocco:

Parametro	Dichiarazione	Tipo di dati	Area di memoria	Descrizione
EN	Input	BOOL	I, Q, M, D, L	Ingresso di abilitazione
ENO	Output	BOOL	I, Q, M, D, L	Uscita di abilitazione
IN	Input	Numeri binari, numeri interi, numeri in virgola mobile, temporizzatori, DATE e CHAR come elementi di una struttura ARRAY	D, L	Primo elemento dell'area sorgente che viene copiato
COUNT	Input	UINT	I, Q, M, D, L o costante	Numero di elementi che vengono copiati dall'area sorgente in quella di destinazione
OUT	Output	Numeri binari, numeri interi, numeri in virgola mobile, temporizzatori, DATE e CHAR come elementi di una struttura ARRAY	D, L	Primo elemento dell'area di destinazione nel quale vengono copiati i contenuti dell'area sorgente

Operazioni di Trasferimento, FILL BLK

esperienza 30

Della Chiesa Enrico 5AET 2013
(enrico.dellachiesa@gmail.com)



L'istruzione UFILL_BLK consente di inserire nell'aria di memoria "b" il valore dell'area di memoria "a" senza interruzioni di tipo hardware/software. Per questo ci potrebbero essere dei ritardi di eventuali allarmi della CPU.
I valori vengono inseriti nell'area di destinazione a partire dall'indirizzo indicato nell'uscita "OUT". L'inserimento viene compiuto tante volte quanto è il valore all'ingresso "COUNT". Il blocco è abilitato dall'ingresso "EN", che deve essere uguale ad 1; se l'inserimento viene compiuto senza la presenza di errori anche l'uscita ENO viene settata alta.
Ecco la tabella con le caratteristiche degli ingressi e delle uscite:

Parametro	**Dichiarazioni**	**Tipo di dati**	**Area di memoria**	**Descrizione**
EN	Input	BOOL	I,Q,M,D,L	Ingresso di abilitazione
EN0	Output	BOOL	I,Q,M,D,L	Uscita di abilitazione
IN	Input	Numeri binari ,numeri interi,numeri in virgola mobile,temporizzatori,DATE e CHAR come elementi di una struttuta ARRAY	D,L o costante	Elemento che viene inserito nell'area di destinazione
COUNT	Input	UINT	I,Q,M,D,L o costante	Numero di esecuzione della copia
OUT	Output	Numeri binari ,numeri interi,numeri in virgola mobile,temporizzatori,DATE e CHAR come elementi di una struttuta ARRAY	D,L	Indirizzo nell'area di destinazione dal quale inizia l'inserimento

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Operazioni di Trasferimento, UFILL BLK

esperienza 31
Della Chiesa Enrico 5AET 2013
(enrico.dellachiesa@gmail.com)

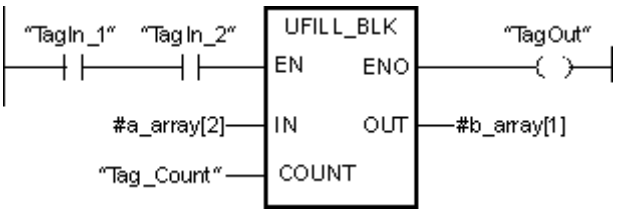


Fig.187 :Esempio Ladder dell'istruzione UFILL BLK

L'istruzione UFILL_BLK consente di inserire nell'aria di memoria "b" il valore dell'area di memoria "a" senza interruzioni di tipo hardware/software. Per questo ci potrebbero essere dei ritardi di eventuali allarmi della
CPU. I valori vengono inseriti nell'area di
destinazione a partire dall'indirizzo indicato nell'uscita "OUT". L'inserimento viene compiuto tante volte quanto è il valore all'ingresso "COUNT". Il blocco è abilitato dall'ingresso "EN", che deve essere uguale ad 1; se l'inserimento viene compiuto senza la presenza di errori anche l'uscita ENO viene settata alta.

Ecco la tabella con le caratteristiche degli ingressi e delle uscite:

Parametro	**Dichiarazioni**	**Tipo di dati**	**Area di memoria**	**Descrizione**
EN	Input	BOOL	I,Q,M,D,L	Ingresso di abilitazione
EN0	Output	BOOL	I,Q,M,D,L	Uscita di abilitazione
IN	Input	Numeri binari ,numeri interi,numeri in virgola mobile,temporizzatori,DATE e CHAR come elementi di una struttuta ARRAY	D,L o costante	Elemento che viene inserito nell'area di destinazione
COUNT	Input	UINT	I,Q,M,D,L o costante	Numero di esecuzione della copia
OUT	Output	Numeri binari ,numeri interi,numeri in virgola mobile,temporizzatori,DATE e CHAR come elementi di una struttuta ARRAY	D,L	Indirizzo nell'area di destinazione dal quale inizia l'inserimento

Operazioni di Conversione, CONVERT

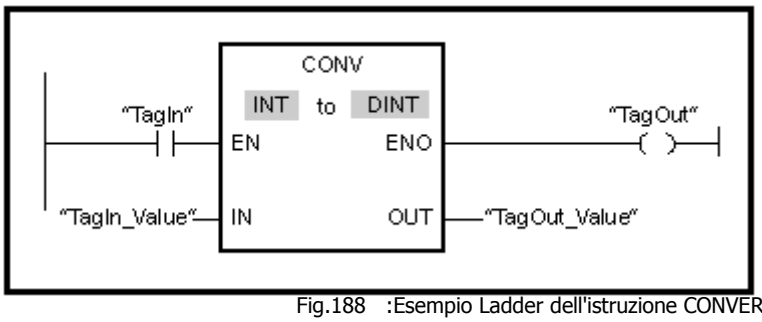


Fig.188 :Esempio Ladder dell'istruzione CONVERT

esperienza 32
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

L'istruzione "Convert" legge il contenuto del parametro IN e lo converte in funzione dei tipi di dati selezionati nel box. Il valore convertito viene emesso nell'uscita OUT.

La conversione viene avviata solo se lo stato di segnale nell'ingresso di abilitazione EN è "1". Se non si verificano errori durante l'esecuzione anche l'uscita ENO ha lo stato di segnale "1".

L'uscita di abilitazione ENO fornisce lo stato di segnale "0" se viene soddisfatta una delle seguenti condizioni:

- L'ingresso "EN" ha lo stato di segnale "0".
- Si verificano errori durante l'elaborazione, ad es. un overflow.
- Nell'ingresso IN è indicato un operando di tipo BYTE, WORD o DWORD nel quale il bit con il valore più alto è impostato. Nell'uscita OUT è indicato un numero intero con segno (SINT, INT, DINT) che ha la stessa lunghezza di bit dell'operando nell'ingresso IN.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Operazioni di Conversione, ROUND

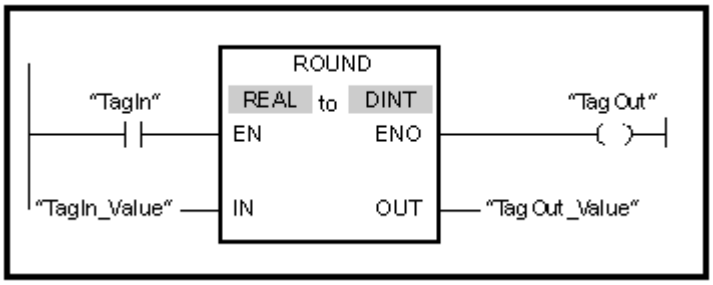


Fig.189 :Esempio con il blocco Round

esperienza 33
Carloni Lorenzo
Matteucci Thomas
5AET 2013
(lorenzo.carloni1@gmail.com)
(thomas.matteucci@gmail.com)

L'istruzione "Round" consente di arrotondare il valore nell'ingresso IN al successivo numero intero. L'istruzione interpreta il valore nell'ingresso IN come valore in virgola mobile e lo converte nel numero intero di tipo di dati DINT. Il risultato dell'istruzione viene emesso nell'uscita OUT, nella quale può essere letto.

N.B. Se il valore di ingresso si trova tra un numero pari e uno dispari, viene scelto il numero pari.

L'istruzione viene eseguita solo se lo stato di segnale nell'ingresso di abilitazione EN è "1". Se l'istruzione viene eseguita senza errori, anche l'uscita di abilitazione ENO fornisce lo stato di segnale "1".

L'uscita di abilitazione ENO fornisce lo stato di segnale "0" se viene soddisfatta una delle seguenti condizioni:

- L'ingresso "EN" ha lo stato di segnale "0".
- Si verificano errori durante l'elaborazione, ad es. un overflow.

Operazione di Conversione, TRUNC

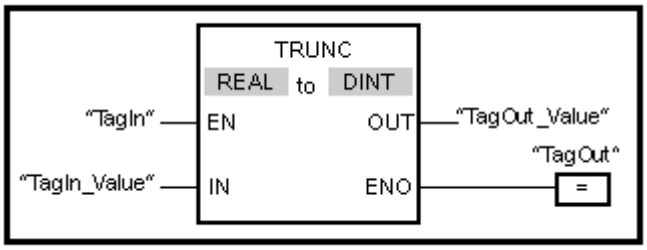


Fig.190 :Esempio con il blocco TRUNC

esperienza 34
Carloni Lorenzo
Matteucci Thomas
5AET 2013
(lorenzo.carloni1@gmail.com)
(thomas.matteucci@gmail.com)

L'istruzione "Trunc" consente di generare un numero intero senza arrotondamento dal valore nell'ingresso IN. Il valore nell'ingresso IN viene interpretato come numero in virgola mobile.

L'istruzione seleziona solo la parte intera del numero in virgola mobile e la emette nell'uscita OUT senza decimali.

L'istruzione viene eseguita solo se lo stato di segnale nell'ingresso di abilitazione EN è "1". Se l'istruzione viene eseguita senza errori, anche l'uscita di abilitazione ENO fornisce lo stato di segnale "1".

L'uscita di abilitazione ENO fornisce lo stato di segnale "0" se viene soddisfatta una delle seguenti condizioni:

- L'ingresso "EN" ha lo stato di segnale "0".
- Si verificano errori durante l'elaborazione, ad es. un overflow.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Operazione di Conversione, CEIL

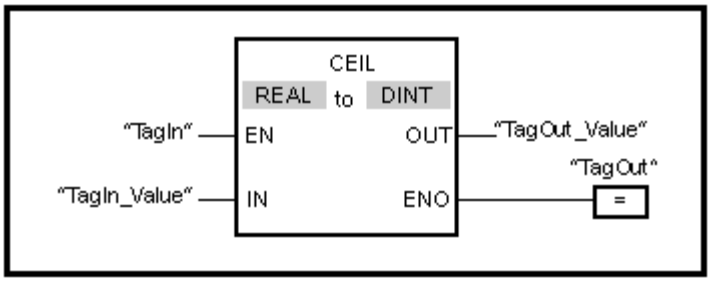


Fig.191 :Esempio con il blocco CEIL

esperienza 35
Carloni Lorenzo
Matteucci Thomas
5AET 2013
(lorenzo.carloni1@gmail.com)
(thomas.matteucci@gmail.com)

L'istruzione "Ceil" arrotonda il valore nell'ingresso IN al successivo numero intero superiore.

L'istruzione interpreta il valore nell'ingresso IN come valore in virgola mobile e lo converte nel numero intero superiore più vicino.

Il risultato dell'istruzione viene emesso nell'uscita OUT, nella quale può essere letto. Il valore di uscita può essere maggiore o uguale al valore di ingresso.

L'istruzione viene eseguita solo se lo stato di segnale nell'ingresso di abilitazione EN è "1". Se l'istruzione viene eseguita senza errori, anche l'uscita di abilitazione ENO fornisce lo stato di segnale "1".

L'uscita di abilitazione ENO fornisce lo stato di segnale "0" se viene soddisfatta una delle seguenti condizioni:

- L'ingresso "EN" ha lo stato di segnale "0".
- Si verificano errori durante l'elaborazione, ad es. un overflow.

Operazione di Conversione, FLOOR

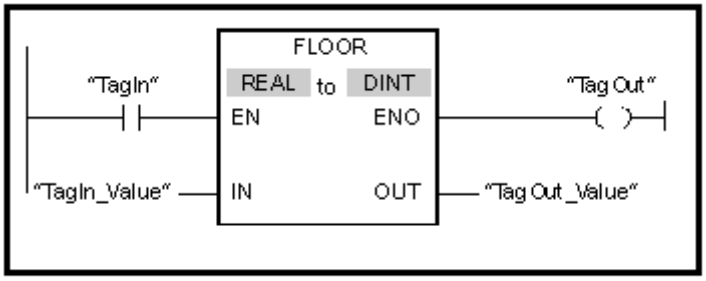


Fig.192 :Esempio con il blocco FLOOR

esperienza 36

Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

L'istruzione "Floor" arrotonda il valore nell'ingresso IN al successivo numero intero inferiore.

L'istruzione interpreta il valore nell'ingresso IN come valore in virgola mobile e lo converte nel numero intero inferiore.

Il risultato dell'istruzione viene emesso nell'uscita OUT, nella quale può essere letto. Il valore di uscita può essere minore o uguale al valore di ingresso.

L'istruzione viene eseguita solo se lo stato di segnale nell'ingresso di abilitazione EN è "1". Se l'istruzione viene eseguita senza errori, anche l'uscita di abilitazione ENO fornisce lo stato di segnale "1".

L'uscita di abilitazione ENO fornisce lo stato di segnale "0" se viene soddisfatta una delle seguenti condizioni:

- L'ingresso "EN" ha lo stato di segnale "0".
- Si verificano errori durante l'elaborazione, ad es. un overflow.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Operazione di Conversione, SCALE_X

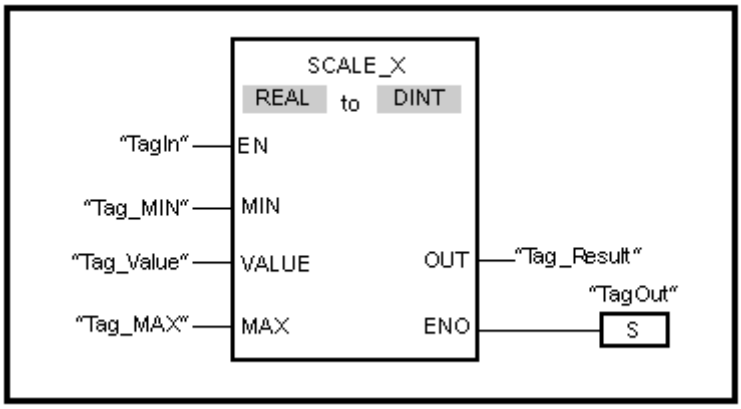


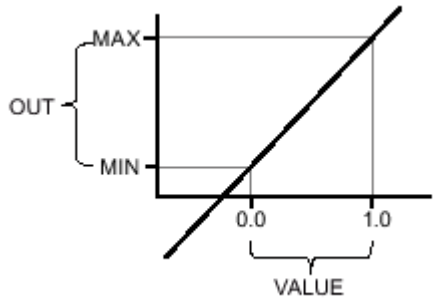
Fig.193 :Esempio con il blocco SCALE_X

esperienza 37
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

L'istruzione "Scale_x" riporta in scala il valore nell'ingresso VALUE rappresentandolo in un determinato campo di valori.

Durante l'esecuzione dell'istruzione "Scale_x" il valore in virgola mobile nell'ingresso VALUE viene riportato in scala nel campo di valori definito con i parametri MIN e MAX.

Fig.194 : rappresentazione in scala del numero intero che viene salvato nell'uscita OUT.



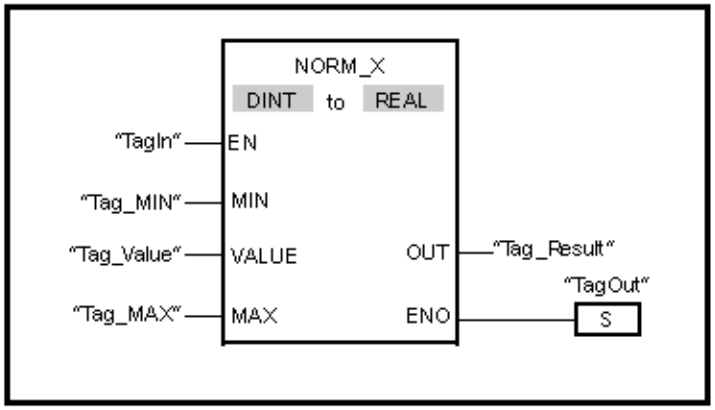
Il risultato della rappresentazione in scala è un numero intero che viene salvato nell'uscita OUT.

L'istruzione viene eseguita solo se lo stato di segnale nell'ingresso di abilitazione EN è "1". Se l'istruzione viene eseguita senza errori, anche l'uscita di abilitazione ENO fornisce lo stato di segnale "1".

L'uscita di abilitazione ENO fornisce lo stato di segnale "0" se viene soddisfatta una delle seguenti condizioni:

- L'ingresso "EN" ha lo stato di segnale "0".
- Il valore nell'ingresso MIN è maggiore o uguale a quello dell'ingresso MAX.
- Il valore di uno specifico numero in virgola mobile non è compreso nel campo dei numeri normalizzati secondo IEEE-754.
- Si verifica un overflow.
- Il valore nell'ingresso VALUE è NaN (Not a number = risultato di un'operazione di calcolo non valida).

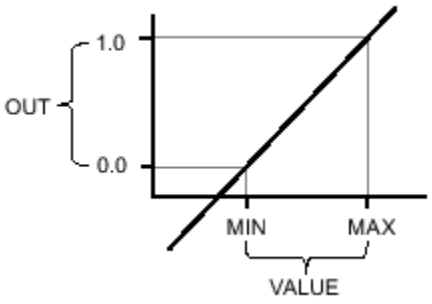
Operazione di Conversione, NORM_X



esperienza 38
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

L'istruzione "Norm_x" normalizza il valore della variabile nell'ingresso VALUE rappresentandolo su una scala lineare.

Con i parametri MIN e MAX si definiscono i limiti del campo di valori che viene rappresentato nella scala. A seconda della posizione del valore da normalizzare in questo campo di valori, il risultato viene calcolato e salvato nell'uscita OUT come numero in virgola mobile.



Se il valore da normalizzare è uguale al valore dell'ingresso MIN, l'uscita OUT fornisce il valore "0.0". Se il valore da normalizzare assume il valore dell'ingresso MAX, il valore nell'uscita OUT è "1.0".

L'istruzione viene eseguita solo se lo stato di segnale nell'ingresso di abilitazione EN è "1". Se l'istruzione viene eseguita senza errori, anche l'uscita di abilitazione ENO fornisce lo stato di segnale "1".

L'uscita di abilitazione ENO fornisce lo stato di segnale "0" se viene soddisfatta una delle seguenti condizioni:

- L'ingresso "EN" ha lo stato di segnale "0".
- Il valore nell'ingresso MIN è maggiore o uguale a quello dell'ingresso MAX.
- Il valore di uno specifico numero in virgola mobile non è compreso nel campo dei numeri normalizzati secondo IEEE-754.
- Il valore nell'ingresso VALUE è NaN (risultato di un'operazione di calcolo non valida).

Controllo del Programma, JMP

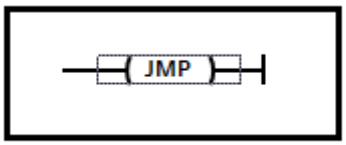


Fig.195 :Blocco Jump

esperienza 39
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

L'istruzione "Jmp" consente di interrompere l'elaborazione lineare del programma e di proseguire in un altro segmento. Il segmento di destinazione deve essere identificato da un'etichetta di salto (LABEL) il cui nome va indicato nel segnaposto sopra l'istruzione e deve essere contenuta nello stesso blocco in cui viene eseguita l'istruzione. La sua denominazione può essere assegnata una volta sola nel blocco.

Se il risultato logico combinatorio (RLO) nell'ingresso dell'istruzione è "1", viene effettuato un salto al segmento contrassegnato dall'etichetta indicata. Il salto può essere effettuato verso segmenti con numero sia superiore che inferiore.

Se la condizione nell'ingresso dell'istruzione non viene soddisfatta (RLO = 0) l'elaborazione del programma prosegue nel segmento successivo

(qui va l'immagine della spiegazione) vedi Tia

L'istruzione "Salta se RLO = 1" viene eseguita se l'operando "TagIn_1" ha lo stato di segnale "1". L'elaborazione lineare del programma viene interrotta e prosegue nel segmento 3 identificato dall'etichetta di salto CAS1. Se l'ingresso "TagIn_3" ha lo stato di segnale "1" l'uscita "TagOut_3" viene resettata.

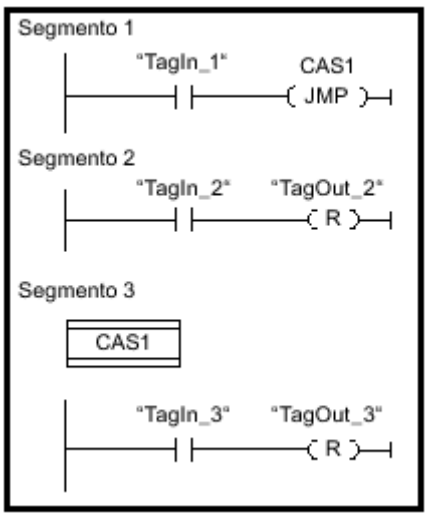


Fig.196 :Esempio applicativo del blocco Jump

Controllo del Programma, JMPN

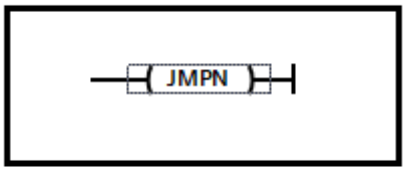


Fig.197 :Blocco JMPN

esperienza 40
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

L'istruzione "JMPN" consente di interrompere l'elaborazione lineare del programma e di proseguire in un altro segmento se il risultato logico combinatorio nell'ingresso dell'istruzione è "0". Il segmento di destinazione deve essere identificato da un'etichetta di salto (LABEL) il cui nome va indicato nel segnaposto sopra l'istruzione e deve essere contenuta nello stesso blocco in cui viene eseguita l'istruzione. La sua denominazione può essere assegnata una volta sola nel blocco.

Se il risultato logico combinatorio (RLO) nell'ingresso dell'istruzione è "0", viene effettuato un salto al segmento contrassegnato dall'etichetta indicata. Il salto può essere effettuato verso segmenti con numero sia superiore che inferiore.

Se il risultato logico combinatorio nell'ingresso dell'istruzione è "1", l'elaborazione del programma prosegue nel segmento successivo.

(qui va l'immagine della spiegazione) vedi Tia

L'istruzione "Salta se RLO = 0" viene eseguita se l'operando "TagIn_1" fornisce lo stato di segnale "0". L'elaborazione lineare del programma viene interrotta e prosegue nel segmento 3 identificato dall'etichetta di salto CAS1. Se l'ingresso "TagIn_3" ha lo stato di segnale "1" l'uscita "TagOut_3" viene resettata.

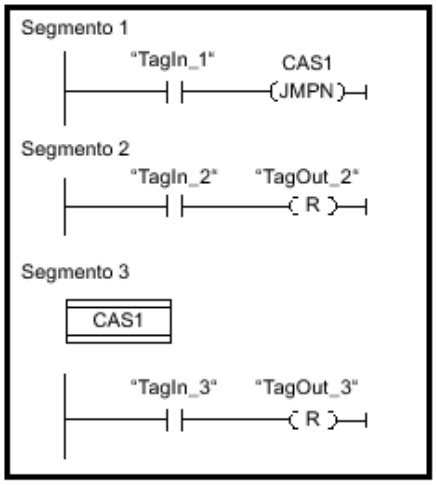


Fig.198 :Esempio applicativo blocco JMPN

Controllo del Programma, RET

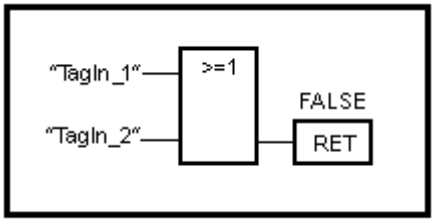


Fig.199: Blocco RET

esperienza 41
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

L'istruzione "Salta indietro" consente di concludere l'esecuzione di un blocco richiamato. Esistono tre modi di eseguire questa operazione:

- **Senza richiamo dell'istruzione "Salta indietro"**
  Il blocco viene chiuso al termine dell'esecuzione dell'ultimo segmento. L'ENO della funzione di richiamo viene impostato allo stato di segnale "1".

- **Richiamo dell'istruzione "Salta indietro" preceduta da un'operazione logica combinatoria (vedere esempio)**
  Il blocco viene chiuso se il collegamento sinistro ha lo stato di segnale "1". L'ENO della funzione di richiamo corrisponde all'operando.

- **Richiamo dell'istruzione "Salta indietro" non preceduta da un'operazione logica combinatoria**
  Il blocco viene chiuso. L'ENO della funzione di richiamo corrisponde all'operando.

Se il risultato logico combinatorio (RLO) nell'ingresso dell'istruzione "Salta indietro" è "1" l'elaborazione del programma nel blocco richiamato viene interrotta e proseguita nel blocco (ad es. nell'OB) richiamante dopo la funzione di richiamo. Lo stato (ENO) della funzione di richiamo viene determinato dal parametro dell'istruzione. che può assumere i valori seguenti:

- RLO
  La combinazione logica viene messa in uscita
- TRUE/FALSE
  Posso scegliere se mettere in uscita "1" (TRUE) o "0" (FALSE)
- <Operando>
  In uscita avrò il valore scelto in precedenza. (I, Q, M, D, L)

(qui va l'immagine della spiegazione) vedi Tia

L'istruzione "Salta indietro" viene eseguita se l'operando "TagIn" ha lo stato di segnale "1". L'elaborazione del programma viene interrotta nel blocco richiamato e proseguita in quello richiamante. L'uscita "ENO" della funzione di richiamo viene resettata sullo stato di segnale "0".

Controllo del Programma, RE TRIGR

esperienza 42

Gridella Andrea 5AET 2013
(gridella.andrea@libero.it)

Con questo blocco si ha la possibilità, riavviando il watchdog del tempo di ciclo, di aumentare il tempo massimo che può trascorrere prima che il timer di controllo del tempo di ciclo induca un errore. Questo blocco è molto semplice da utilizzare ed è costituito da un solo ingresso ed una uscita, ai quali per dimostrazione abbiamo collegato rispettivamente uno switch(S1) e un'uscita generica(U). Un semplice esempio di come vengano collegati ingressi ed uscite a questo blocco è il seguente:

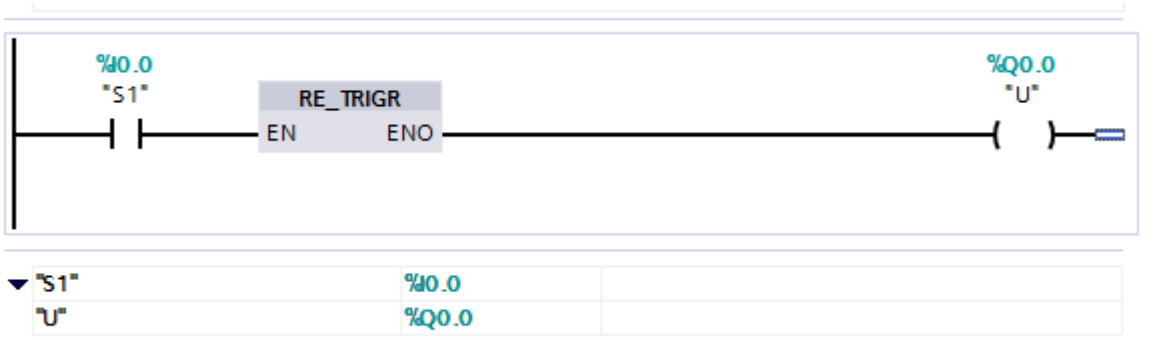


Fig.200 :Esercizio Ladder con blocco RE_TRIGR

L'ingresso e l'uscita li abbiamo rinominati in S1 e U in modo che il lettore capisca in modo immediato il funzionamento e, per farlo, siamo andati nella Tabella delle variabili e abbiamo modificato i nomi delle varibili:

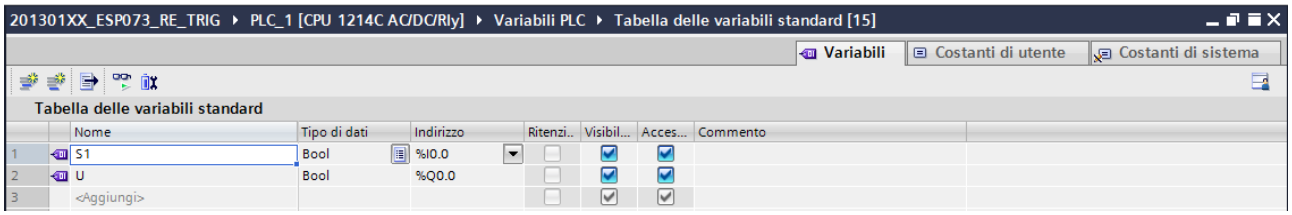


Fig.203 :Variabili usate nell'esercizio

Controllo del Programma, STP

esperienza 43
Gridella Andrea 5AET 2013
(gridella.andrea@libero.it)

Con questo blocco di funzione si ferma il funzionamento del PLC, fermando conseguentemente l'esecuzione del programma e gli aggiornamenti fisici dell'immagine di processo.

Questo blocco ha un solo ingresso al quale possiamo collegare uscite di altri blocchi o merker in modo che lo stop del PLC avvenga come conseguenza ad eventi precedenti.

Il blocco ha anche una sola uscita alla quale, per esempio, possiamo collegare un led di uscita che ci mostri il momento in cui il PLC viene fermato. Un semplice esempio di come avvengano i collegamenti per questo blocco è il seguente:

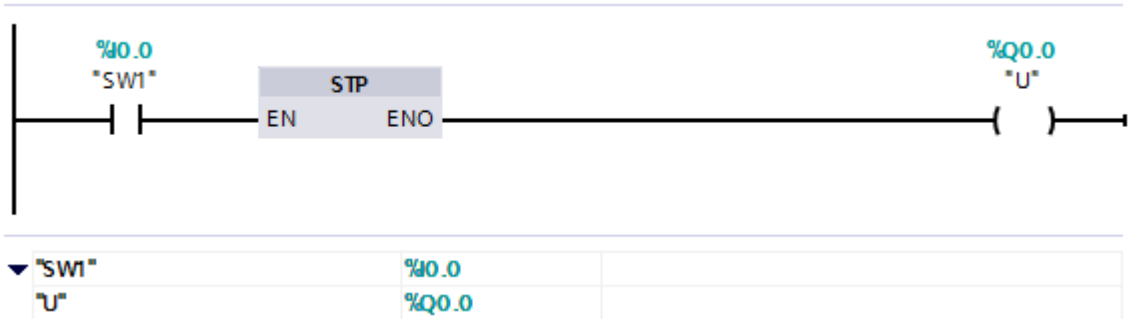


Fig.202 :Esercizio con blocco STP

Anche in questo esempio abbiamo rinominato ingresso ed uscita rispettivamente in SW1 e U con la finalità di semplificare l'immediata comprensione dello schema. La tabella delle variabili è la seguente:

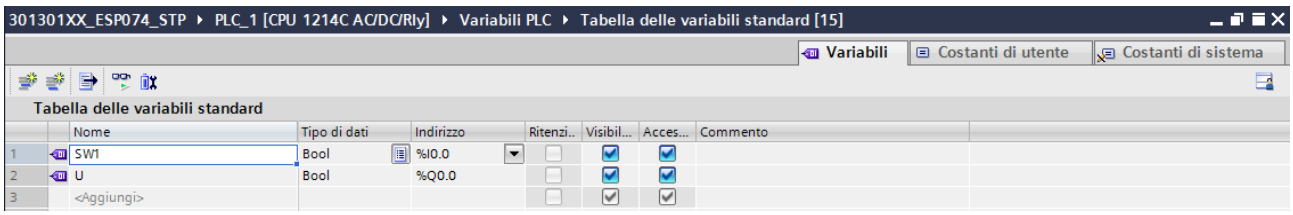


Fig.203 :Variabili usate nell'esercizio

Esercizi sull'automazione industriale

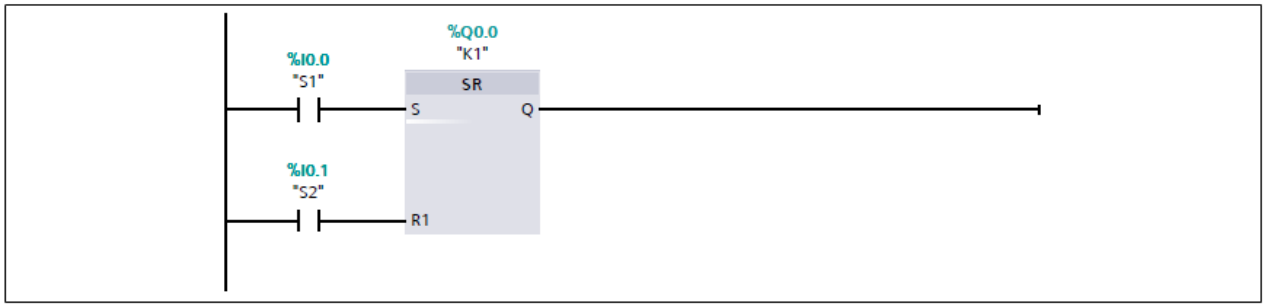
esperienza 44

Farneti Alessandro 5AET 2013
(farneti.alessandro@gmail.com)

Esercizio 1

Realizzare il seguente ciclo di comando:

1) Se si preme S1 si eccita K1
2) Se si preme S2 si eccita K2

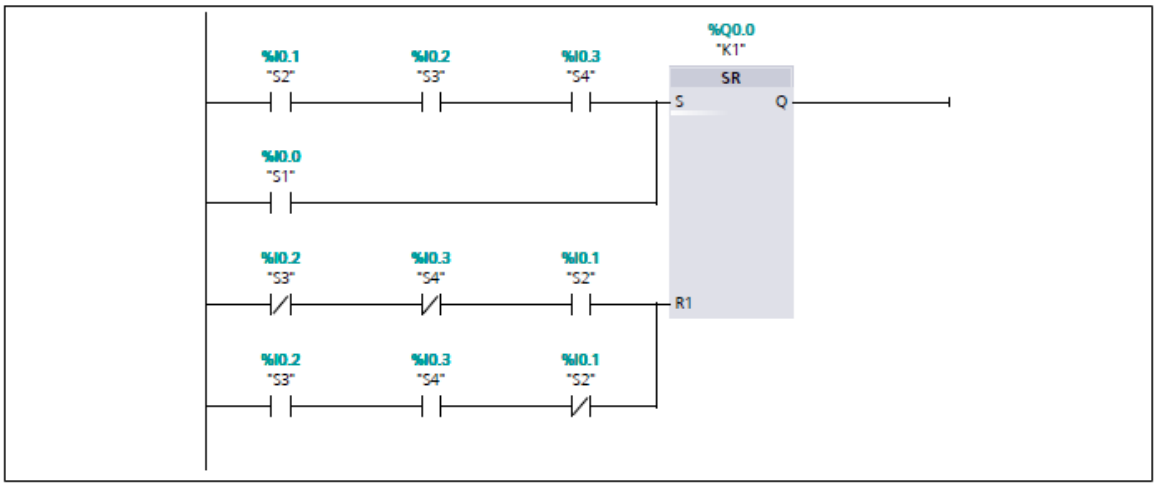


Simbolo	Indirizzo	Tipo	Commento
"K1"	%Q0.0	Bool	
"S2"	%I0.1	Bool	
"S1"	%I0.0	Bool	

Esercizio 2

Realizzare il seguente ciclo di comando:

1) Se si preme S1 si eccita K1
2) Se si preme S2 di alt si diseccita K1
3) Solo premendo contemporaneamente S3 e S4, K1 si eccita e diseccita sempre con S2



Simbolo	Indirizzo	Tipo	Commento
"S2"	%I0.1	Bool	
"S3"	%I0.2	Bool	
"S4"	%I0.3	Bool	
"S1"	%I0.0	Bool	
"K1"	%Q0.0	Bool	

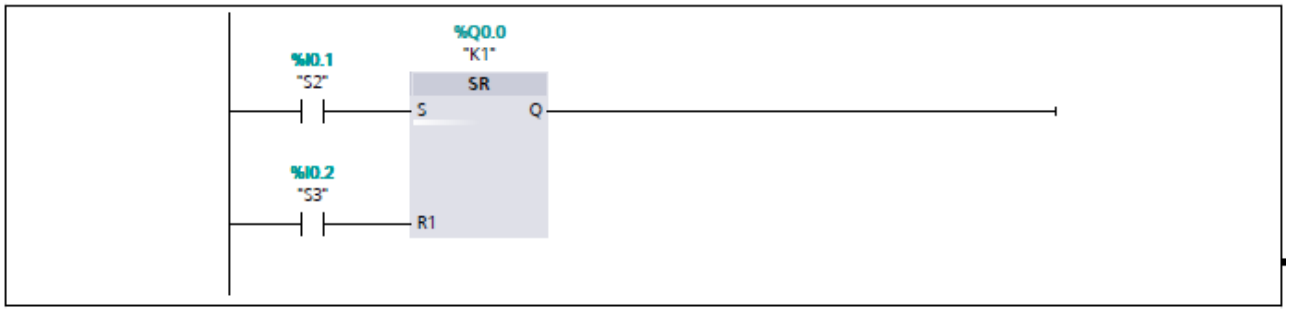
Fig.204 :Esercizio 2 sull'automazione industriale

Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

Esercizio 3

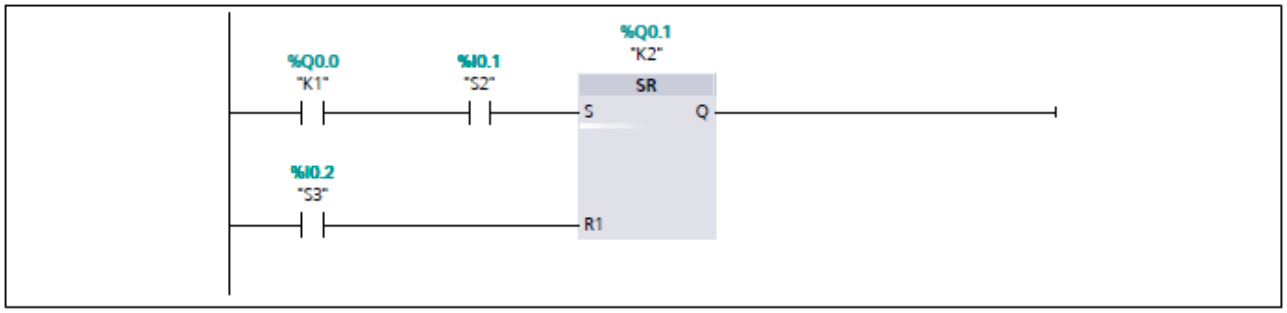
Realizzare il seguente ciclo di comando:

1) K1 si possa eccitare sempre
2) K2 si possa eccitare sempre
3) K1 e K2 si possono diseccitare sempre con S3 di alt



Simbolo	Indirizzo	Tipo	Commento
"S2"	%I0.1	Bool	
"S3"	%I0.2	Bool	
"K1"	%Q0.0	Bool	

Segmento 2:



Simbolo	Indirizzo	Tipo	Commento
"S2"	%I0.1	Bool	
"S3"	%I0.2	Bool	

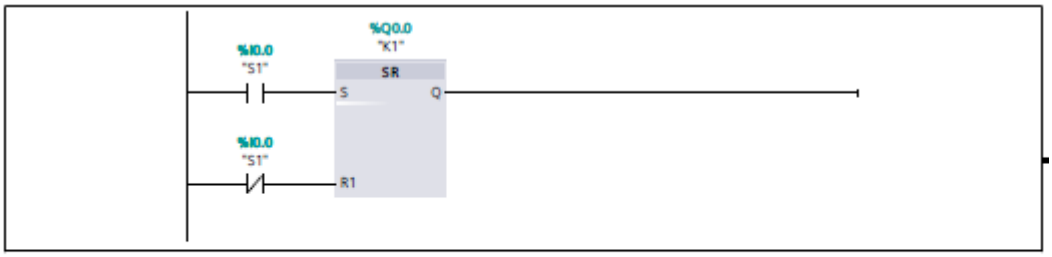
Fig.205 :Esercizio 3 sull'automazione industriale

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Esercizio 4

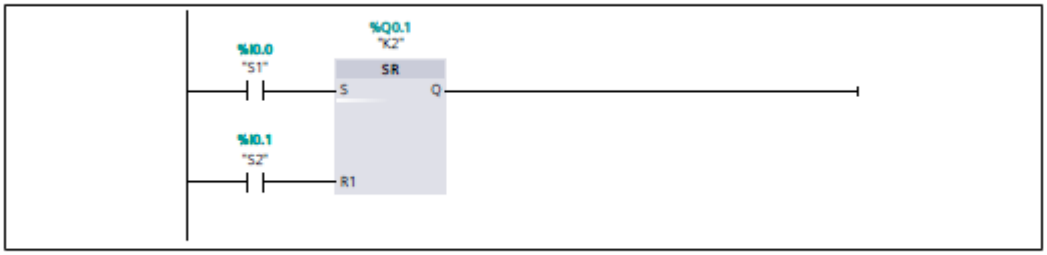
Realizzare il seguente ciclo di comando:

1) Se si preme S1 si eccitano K1 e K2
2) Se si lascia S1 si diseccita K1 e rimane K2 eccitato
3) Se si preme S2 si diseccita K2



Simbolo	Indirizzo	Tipo	Commento
"K1"	%Q0.0	Bool	
"S1"	%I0.0	Bool	

Segmento 2:



Simbolo	Indirizzo	Tipo	Commento
"K2"	%Q0.1	Bool	

Simbolo	Indirizzo	Tipo	Commento
"S1"	%I0.0	Bool	
"S2"	%I0.1	Bool	

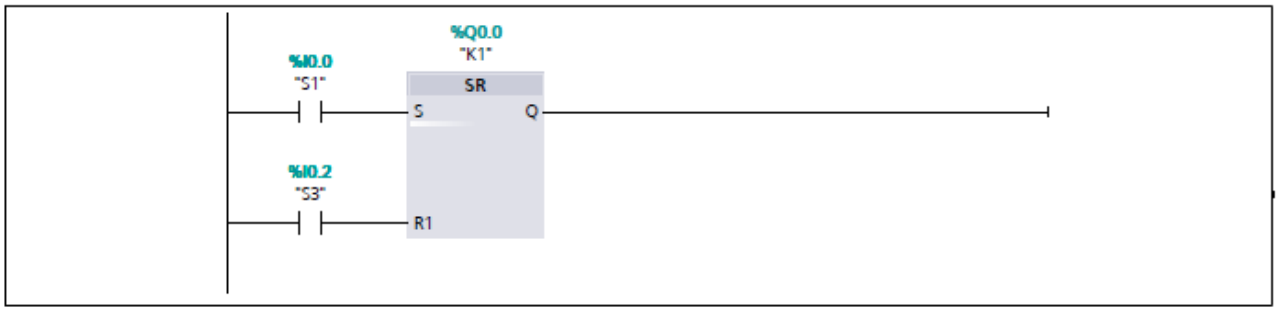
Fig.206 :Esercizio 4 sull'automazione industriale

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Esercizio 5

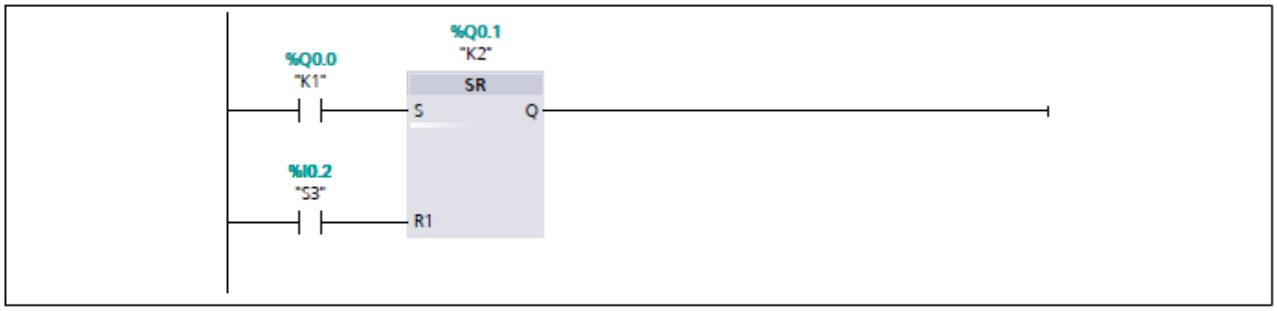
Realizzare il seguente ciclo di comando:

1) K1 si possa eccitare sempre
2) K2 si possa eccitare ad impulsi con K1 diseccitato
3) K2 si eccita e rimane eccitato se K1 è eccitato
4) Diseccitando K1 si diseccita anche K2
5) Con un pulsante di alt si deve diseccitare tutto



Simbolo	Indirizzo	Tipo	Commento
"S1"	%I0.0	Bool	
"S3"	%I0.2	Bool	
"K1"	%Q0.0	Bool	

Segmento 2:



Simbolo	Indirizzo	Tipo	Commento
"S3"	%I0.2	Bool	
"K1"	%Q0.0	Bool	
"K2"	%Q0.1	Bool	

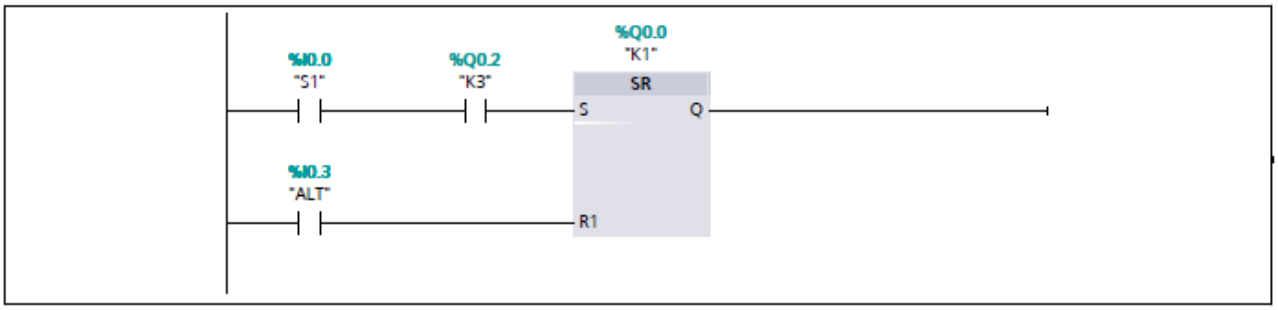
Fig.207 :Esercizio 5 sull'automazione industriale

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Esercizio 6

Realizzare il seguente ciclo di comando:

1) K1 si eccita con K2
2) K2 si può eccitare solo se K3 è eccitato
3) K2 si può eccitare se K1 è diseccitato
4) Con un pulsante di alt si deve diseccitare tutto

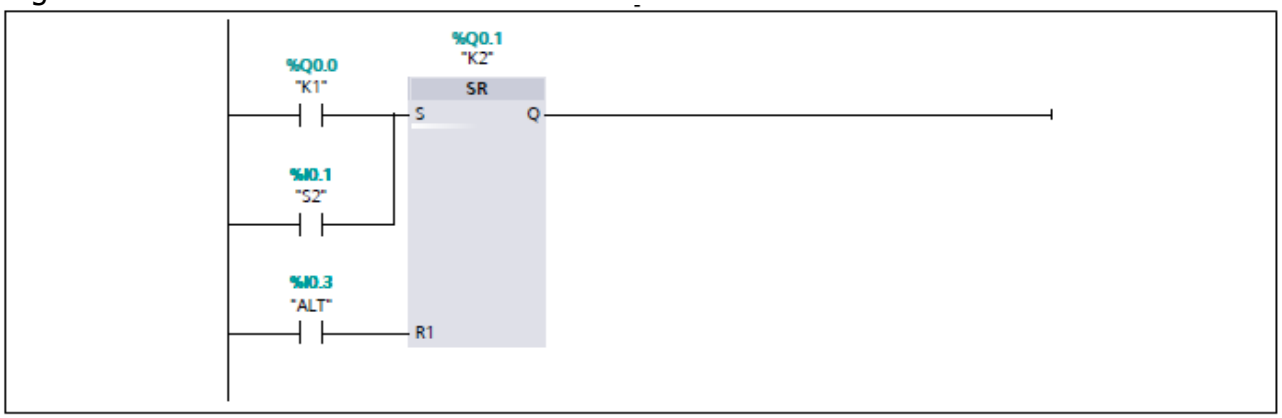


Simbolo	Indirizzo	Tipo	Commento
"K1"	%Q0.0	Bool	
"ALT"	%I0.3	Bool	
"S1"	%I0.0	Bool	
"K3"	%Q0.2	Bool	

Fig.208 :Esercizio 6(segmento 1) sull'automazione industriale

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Segmento 2:



Simbolo	Indirizzo	Tipo	Commento
"K1"	%Q0.0	Bool	
"K2"	%Q0.1	Bool	
"S2"	%I0.1	Bool	
"ALT"	%I0.3	Bool	

Segmento 3:

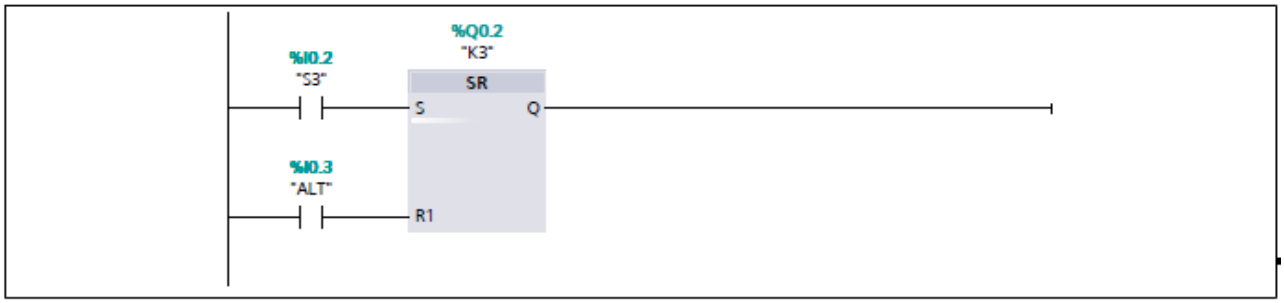


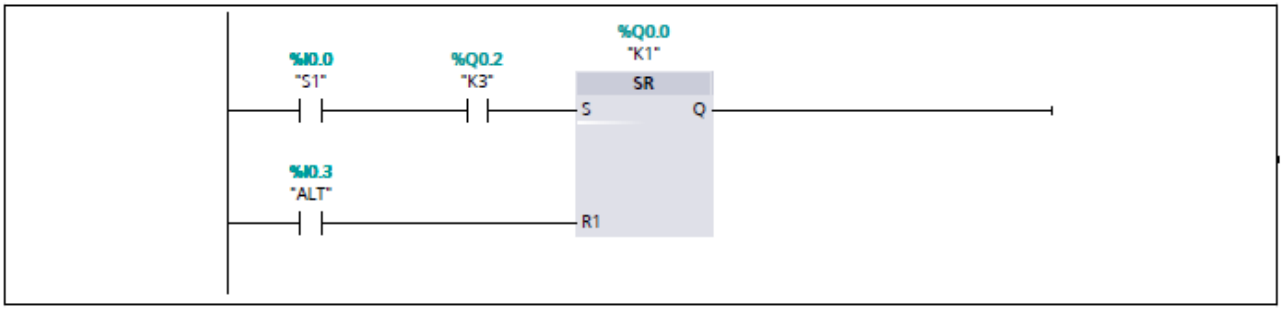
Fig.209 :Esercizio 6(segmento 2 e 3) sull'automazione industriale

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Esercizio 7

Realizare il seguente ciclo di comando:

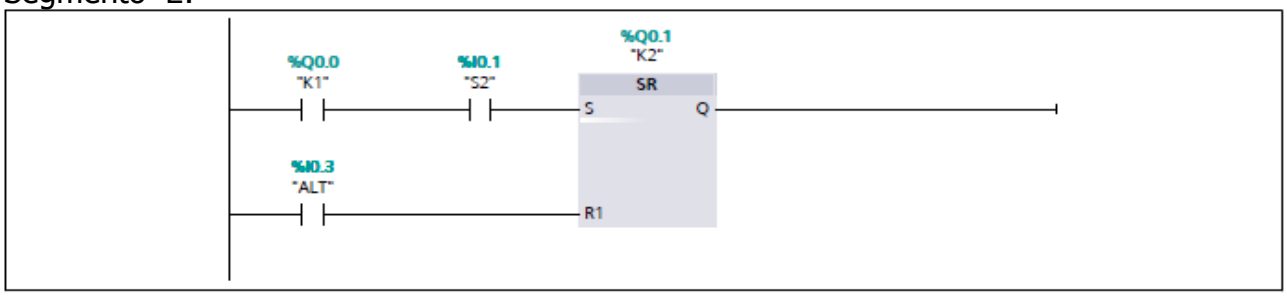
1) K1 si può eccitare solo se K3 è diseccitato
2) K2 si può eccitare solo se K1 è eccitato
3) K3 si può eccitare solo se K1 è diseccitato
4) Con un pulsante di alt si deve diseccitare tutto



Simbolo	Indirizzo	Tipo	Commento
"K1"	%Q0.0	Bool	
"ALT"	%I0.3	Bool	
"S1"	%I0.0	Bool	
"K3"	%Q0.2	Bool	

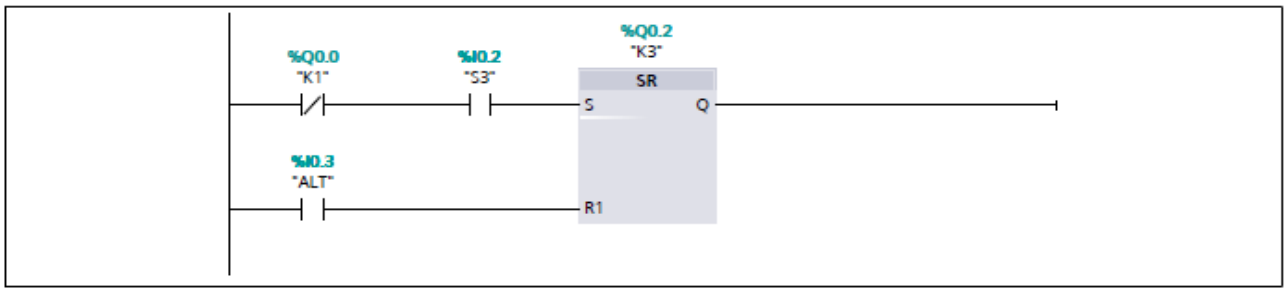
Fig.210 :Esercizio 7(segmento 1) sull'automazione industriale

Segmento 2:



Simbolo	Indirizzo	Tipo	Commento
"K1"	%Q0.0	Bool	
"ALT"	%I0.3	Bool	
"S2"	%I0.1	Bool	
"K2"	%Q0.1	Bool	

Segmento 3:



Simbolo	Indirizzo	Tipo	Commento
"K1"	%Q0.0	Bool	
"ALT"	%I0.3	Bool	
"K3"	%Q0.2	Bool	
"S3"	%I0.2	Bool	

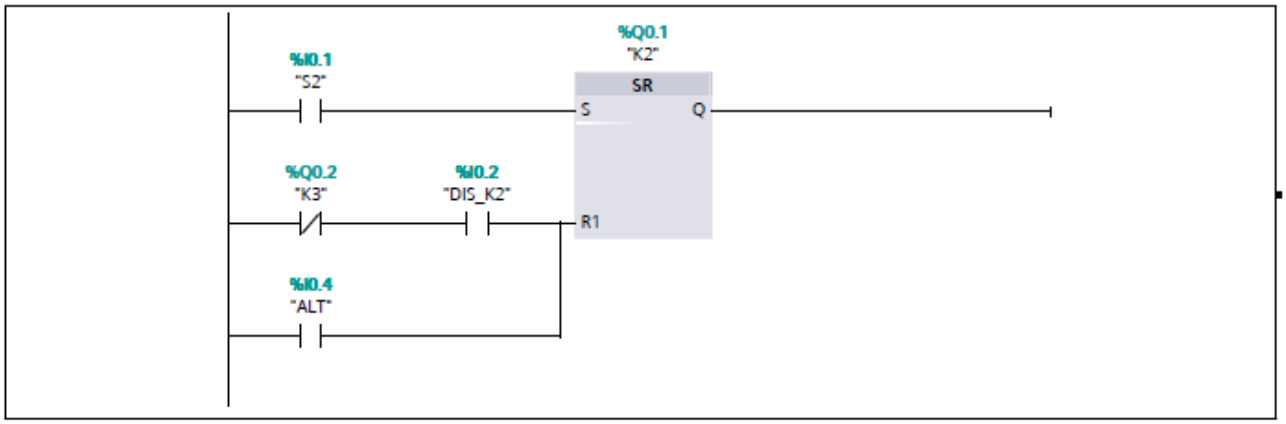
Fig.211 :Esercizio 7(segmento 2 e 3) sull'automazione industriale

Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

Esercizio 8

Realizzare il seguente ciclo di comando:

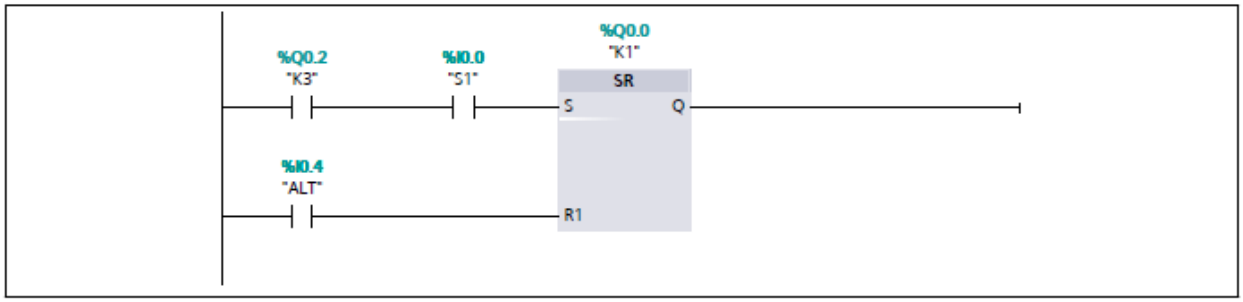
1) Con S1 si può eccitare K2 che eccita K3
2) K2 si può eccitare solo se K2 è eccitato
3) K2 si può diseccitare solo se K1 è diseccitato
4) K3 si può diseccitare solo se K2 è diseccitato
5) Con un pulsante di alt si può diseccitare tutto



Simbolo	Indirizzo	Tipo	Commento
"S2"	%I0.1	Bool	
"K2"	%Q0.1	Bool	
"K3"	%Q0.2	Bool	
"DIS_K2"	%I0.2	Bool	
"ALT"	%I0.4	Bool	

Fig.212 :Esercizio 8(segmento 1) sull'automazione industriale

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com



Simbolo	Indirizzo	Tipo	Commento
"K3"	%Q0.2	Bool	
"ALT"	%I0.4	Bool	
"S1"	%I0.0	Bool	
"K1"	%Q0.0	Bool	

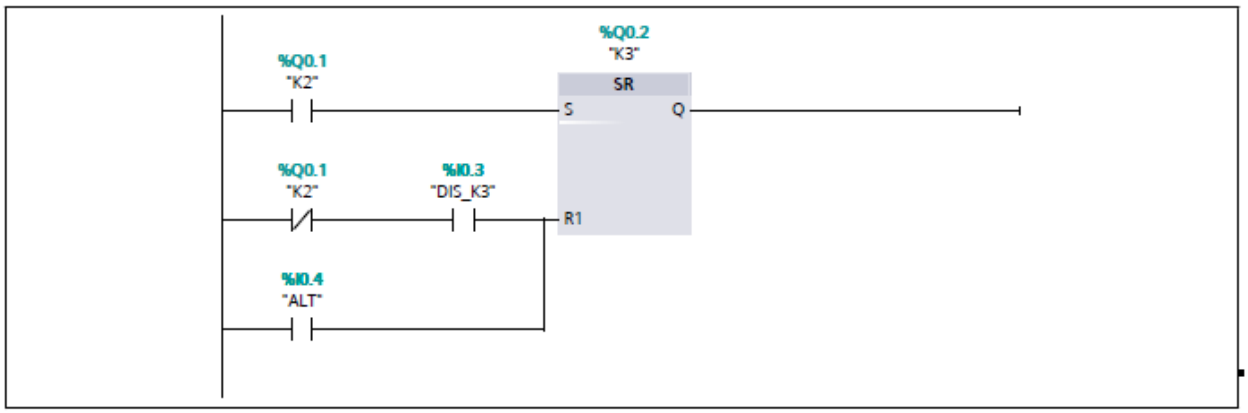


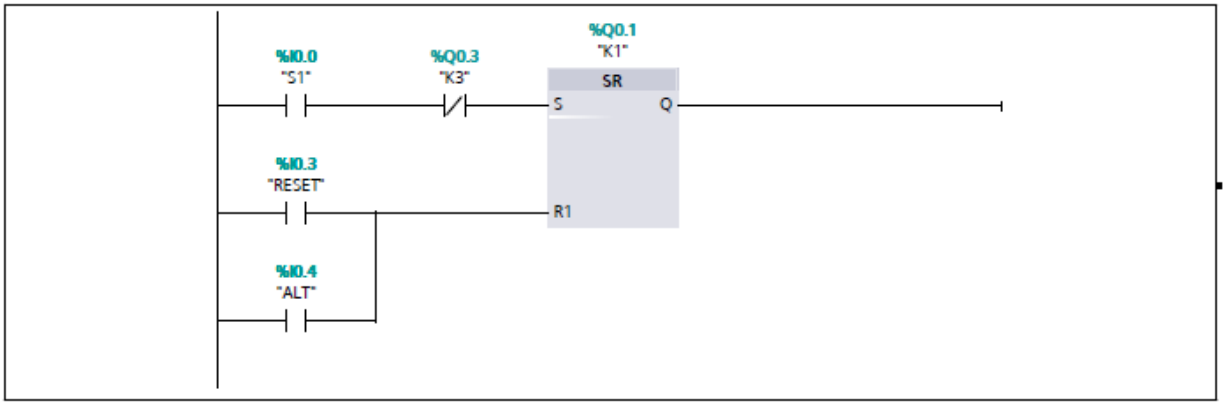
Fig.213 :Esercizio 8(segmento 2 e 3) sull'automazione industriale

Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

ESERCIZIO 9

Realizzare il seguente ciclo di comando:

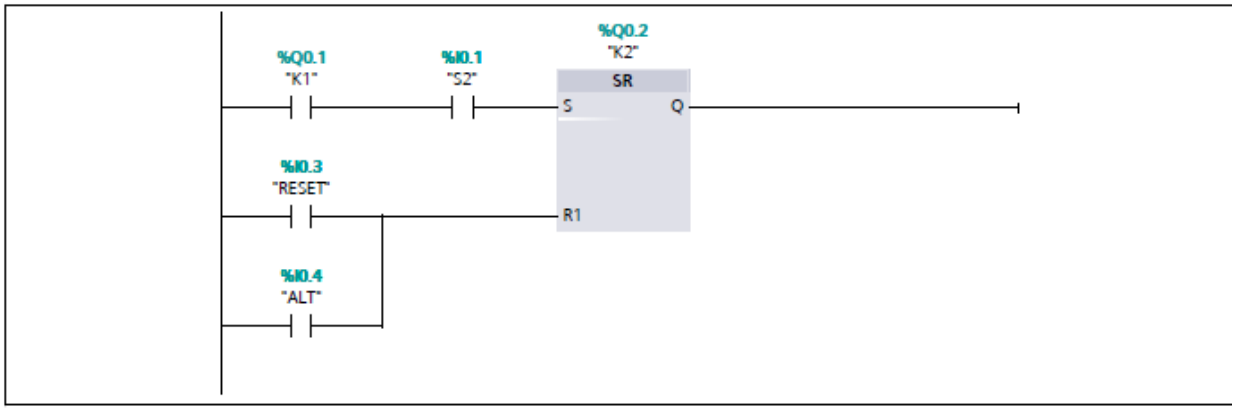
1) K1 si può eccitare solo se K3 è diseccitato
2) K2 si può eccitare solo se K1 è eccitato
3) K3 si può diseccitare solo se K1 è diseccitato
4) Diseccitando K1 si deve diseccitare K2
5) Con un pulsante di alt generale si può diseccitare tutto



Simbolo	Indirizzo	Tipo	Commento
"K3"	%Q0.3	Bool	
"K1"	%Q0.1	Bool	
"RESET"	%I0.3	Bool	
"ALT"	%I0.4	Bool	
"S1"	%I0.0	Bool	

Fig.214 :Esercizio 9(segmento 1) sull'automazione industriale

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com



Simbolo	Indirizzo	Tipo	Commento
"K1"	%Q0.1	Bool	
"RESET"	%I0.3	Bool	
"ALT"	%I0.4	Bool	
"K2"	%Q0.2	Bool	
"S2"	%I0.1	Bool	

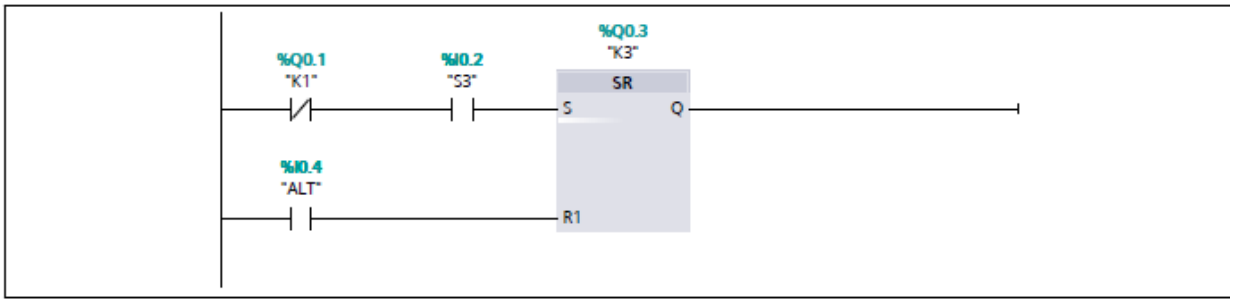


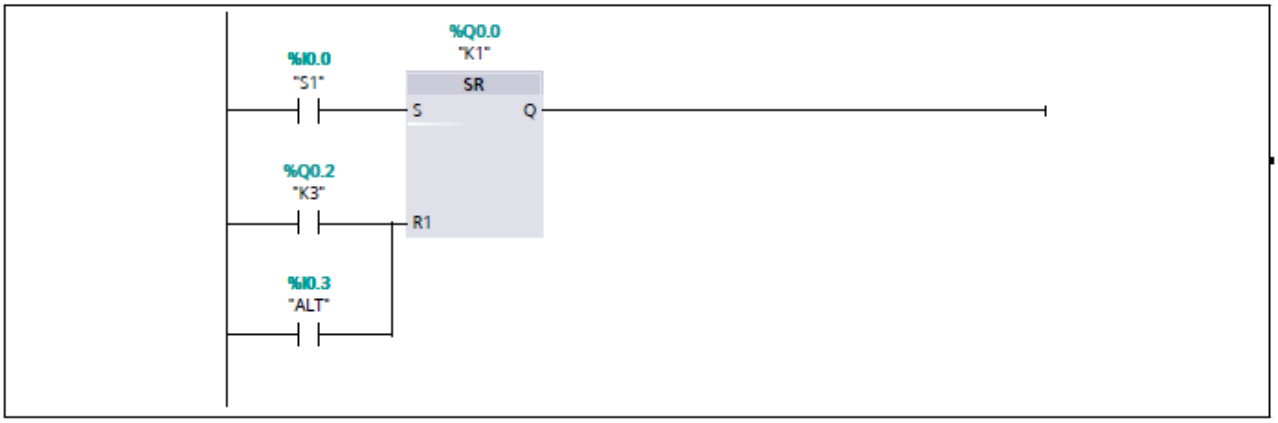
Fig. :Esercizio 9(segmento 2 e 3) sull'automazione industriale

Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

Esercizio 10

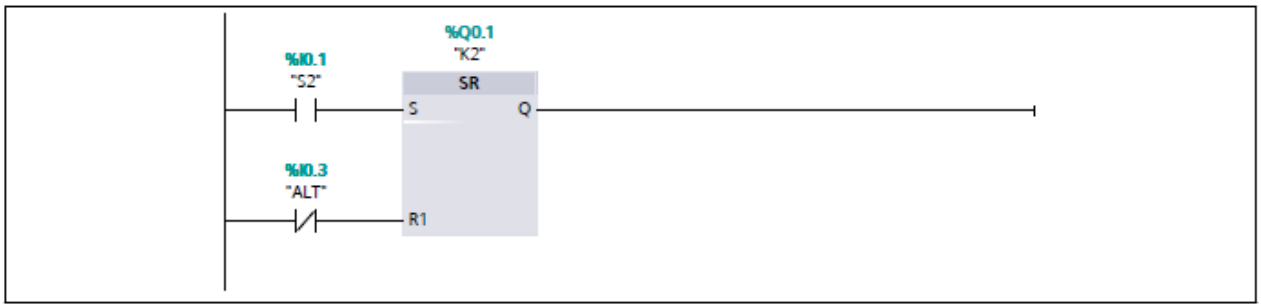
Realizzare il seguente ciclo di comando:

1) K1 e K2 si possono eccitare sempre
2) K3 si può eccitare solo se K1 e K2 sono eccitati
3) Quando si eccita K3 si deve diseccitare K1
4) Con un pulsante di alt si può diseccitare tutto



Simbolo	Indirizzo	Tipo	Commento
"K1"	%Q0.0	Bool	
"ALT"	%I0.3	Bool	
"S1"	%I0.0	Bool	
"K3"	%Q0.2	Bool	

Fig.215 :Esercizio 10(segmento 1) sull'automazione industriale



Simbolo	Indirizzo	Tipo	Commento
"ALT"	%I0.3	Bool	
"K2"	%Q0.1	Bool	
"S2"	%I0.1	Bool	

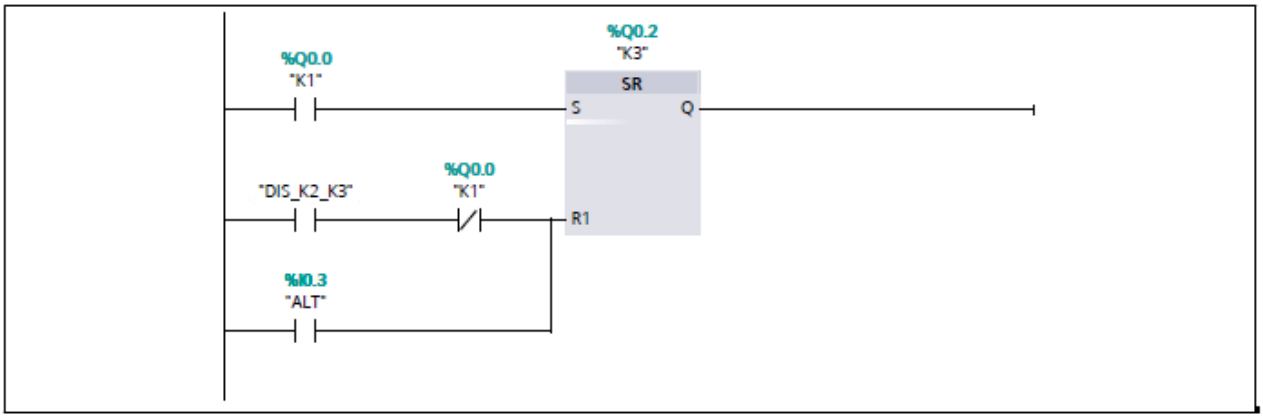


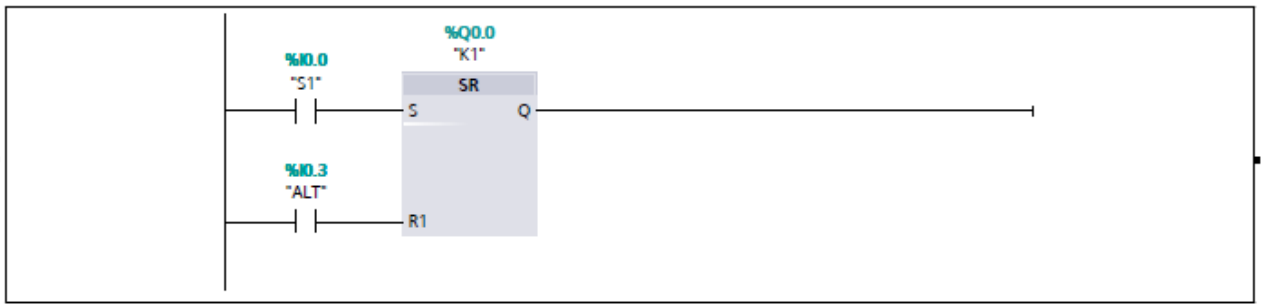
Fig.216 :Esercizio 10(segmento 2 e 3) sull'automazione industriale

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Esercizio 11

Realizzare il seguente ciclo di comando:

1) K1 si può eccitare sempre
2) K3 si può eccitare solo dopo l'eccitazione di K1
3) K2 si eccita automaticamente con K3
4) Con un pulsante di alt si può diseccitare tutto



Simbolo	Indirizzo	Tipo	Commento
"S1"	%I0.0	Bool	
"K1"	%Q0.0	Bool	
"ALT"	%I0.3	Bool	

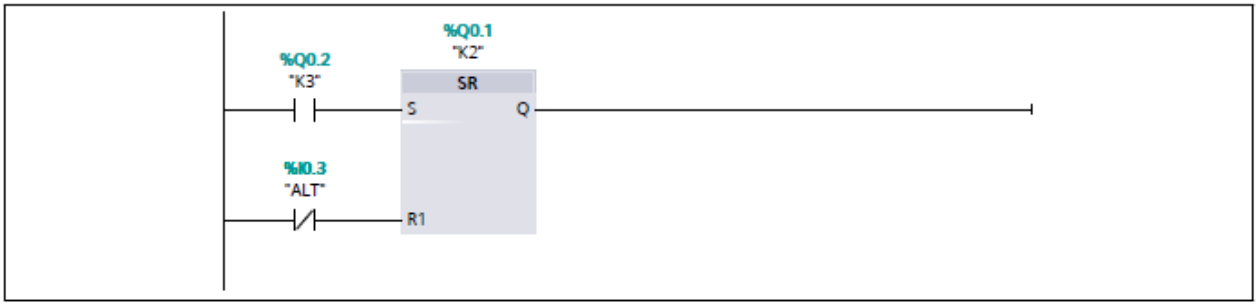
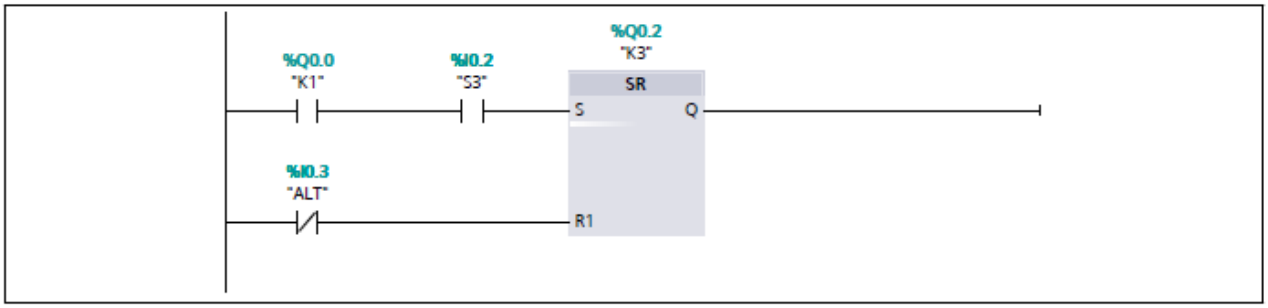


Fig.217 :Esercizio 11(segmento 1 e 2) sull'automazione industriale

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Simbolo	Indirizzo	Tipo	Commento
"K2"	%Q0.1	Bool	
"ALT"	%I0.3	Bool	
"K3"	%Q0.2	Bool	



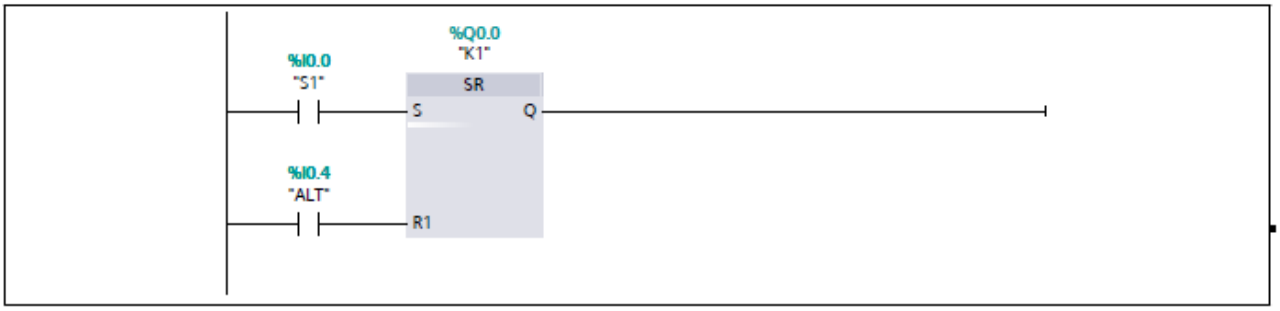
Simbolo	Indirizzo	Tipo	Commento
"K1"	%Q0.0	Bool	
"S3"	%I0.2	Bool	
"ALT"	%I0.3	Bool	
"K3"	%Q0.2	Bool	

Fig.218 :Esercizio 11(segmento 3) sull'automazione industriale

Esercizio 12

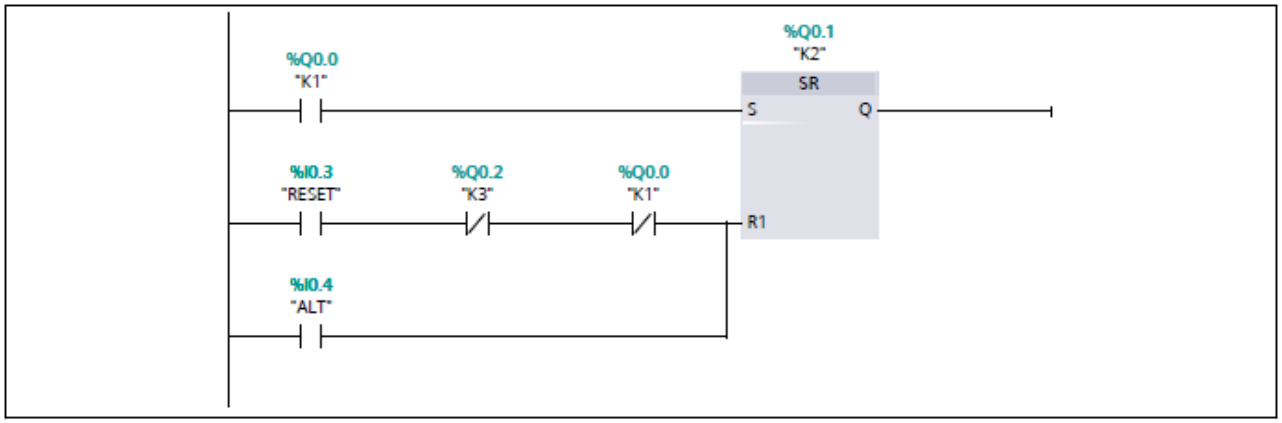
Realizzare il seguente ciclo di comando:

1) K2 e K3 si eccitano automaticamente quado viene eccitato K1
2) K3 si può diseccitare solo se K1 è diseccitato
3) K2 si può diseccitare solo se K1 e K3 sono diseccitati
4) Con un pulsante di alt si diseccita tutto



Simbolo	Indirizzo	Tipo	Commento
"S1"	%I0.0	Bool	
"ALT"	%I0.4	Bool	
"K1"	%Q0.0	Bool	

Fig.219 :Esercizio 12(segmento 1) sull'automazione industriale



Simbolo	Indirizzo	Tipo	Commento
"ALT"	%I0.4	Bool	
"K1"	%Q0.0	Bool	
"K2"	%Q0.1	Bool	
"RESET"	%I0.3	Bool	
"K3"	%Q0.2	Bool	

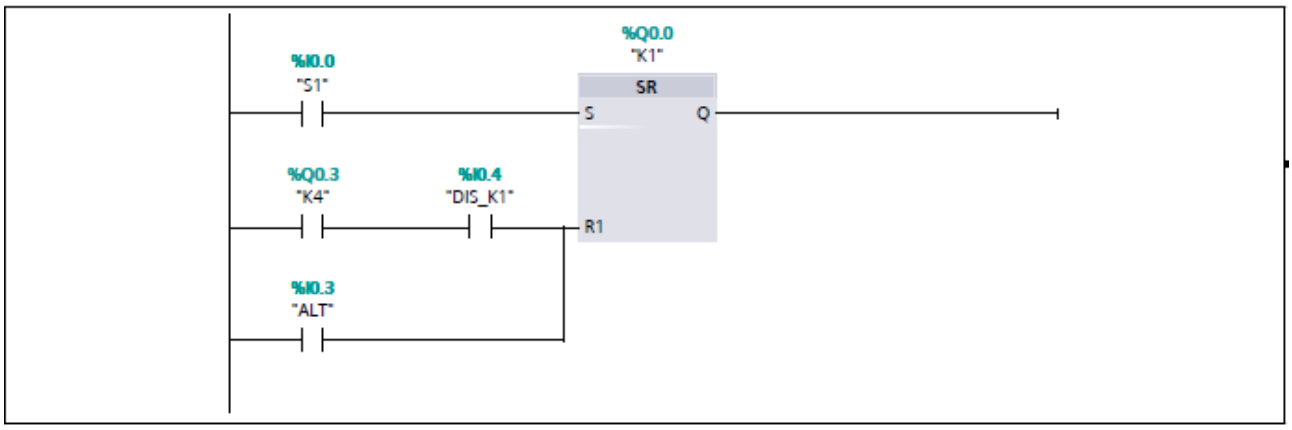
Fig.220 :Esercizio 12(segmento 2) sull'automazione industriale

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Esercizio 13

Realizzare il seguente ciclo di comando:

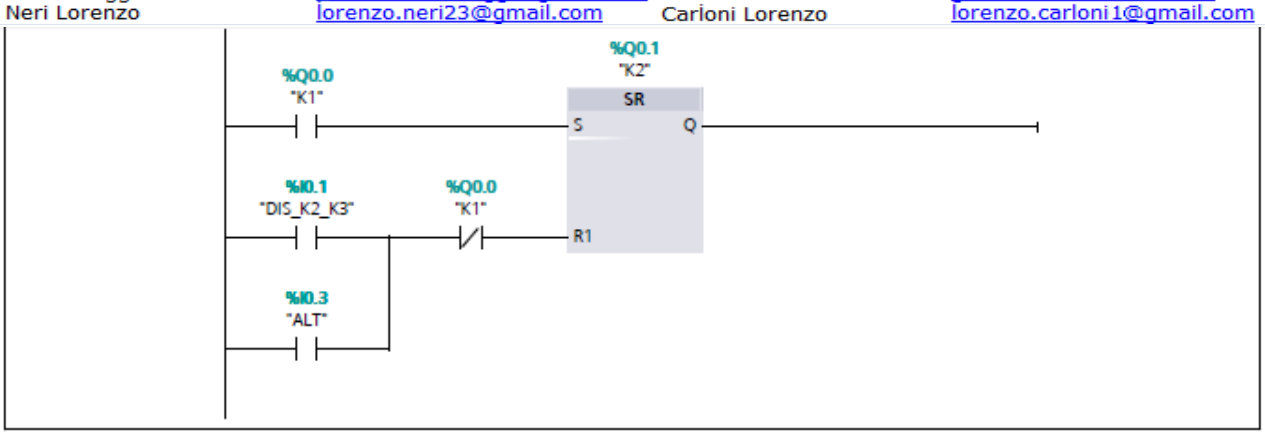
1) K1 si può eccitare sempre
2) K2, K3, K4 si eccitano automaticamente quanndo si eccita K1
3) K2 e K3 si possono diseccitare (contemporaneamente) se K1 è diseccitato
4) K1 si può diseccitare solo se K4 è eccitato
5) Con un pulsante di alt generale si può diseccitare tutto



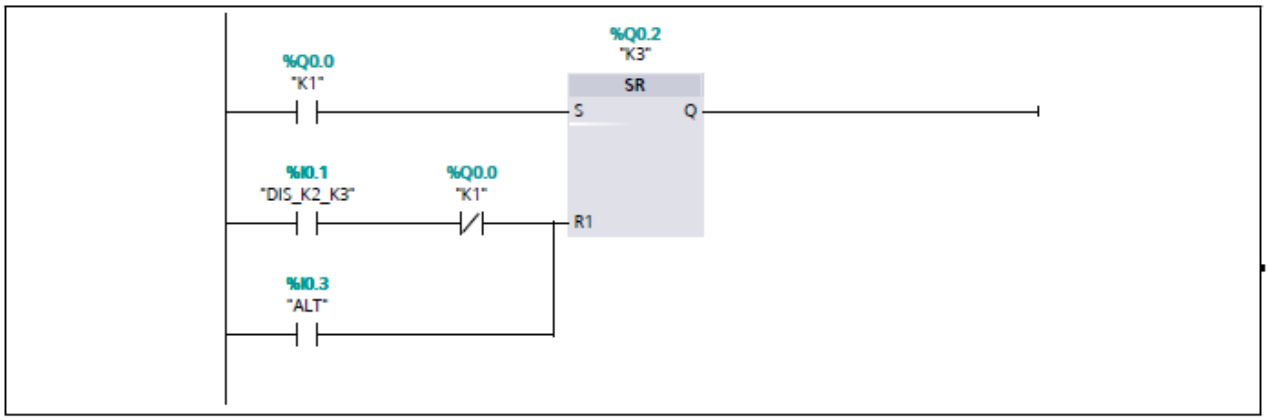
Simbolo	Indirizzo	Tipo	Commento
"K1"	%Q0.0	Bool	
"ALT"	%I0.3	Bool	
"S1"	%I0.0	Bool	
"K4"	%Q0.3	Bool	
"DIS_K1"	%I0.4	Bool	

Fig.221 :Esercizio 13(segmento 1) sull'automazione industriale

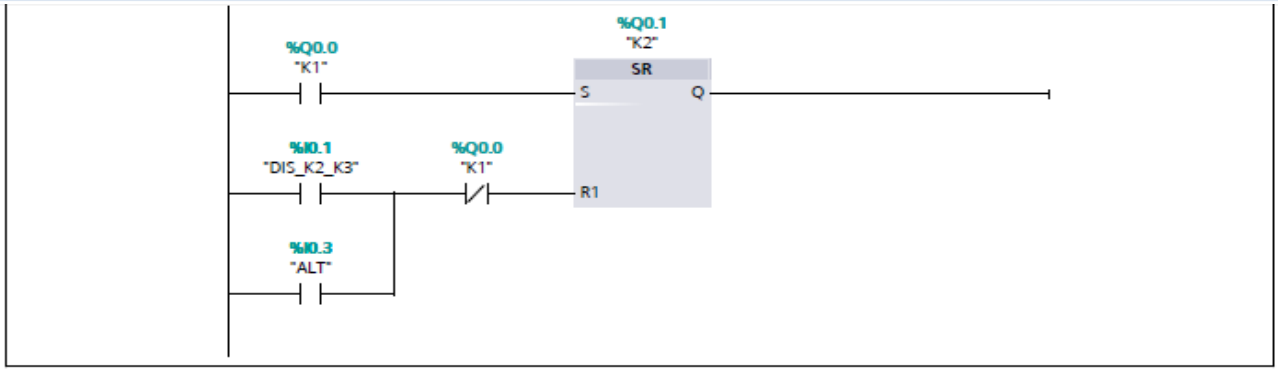
Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com



Simbolo	Indirizzo	Tipo	Commento
"K1"	%Q0.0	Bool	
"DIS_K2_K3"	%I0.1	Bool	
"ALT"	%I0.3	Bool	
"K2"	%Q0.1	Bool	



Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com



Simbolo	Indirizzo	Tipo	Commento
"K1"	%Q0.0	Bool	
"DIS_K2_K3"	%I0.1	Bool	
"ALT"	%I0.3	Bool	
"K2"	%Q0.1	Bool	

Segmento 3:

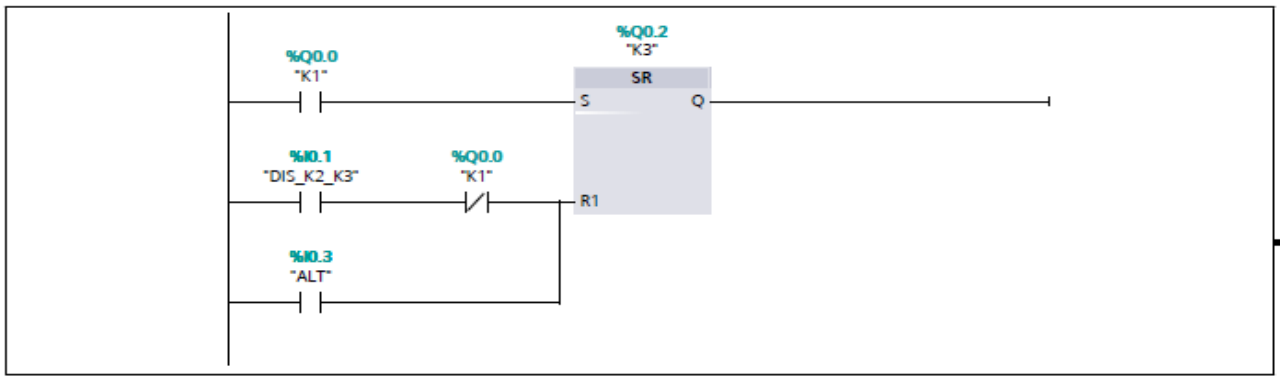


Fig.222 :Esercizio 13(segmento 2 e 3) sull'automazione industriale

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Esercizio 30

Montemaggi Pablo
5AET 2013
(pablo.montemaggi@gmail.com)

Realizzare il seguente ciclo di comando:

1) Premendo il pulsante S1 si eccita K1
2) Se K1 è eccitato, premendo S2 si deve eccitare K2
3) Rilasciando S2, dopo 5 secondi, si deve eccitare K3 e diseccitare K1
4) Automaticamente dopo 2 secondi si eccita K4 per un secondo e per un secondo si diseccita
5) Dopo 5 eccitazioni e diseccitazioni il ciclo si ferma
6) Premendo il pulsante S3 si deve diseccitare tutto in qualsiasi momento

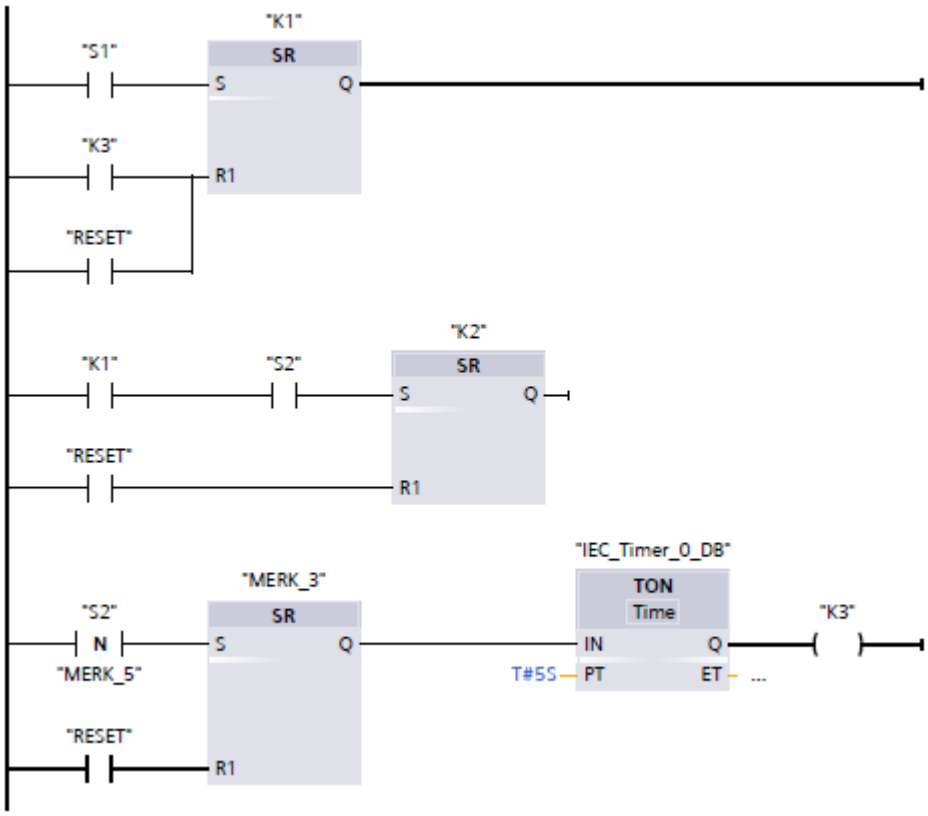


Fig.223 :Esercizio 30(parte 1) sull'automazione industriale

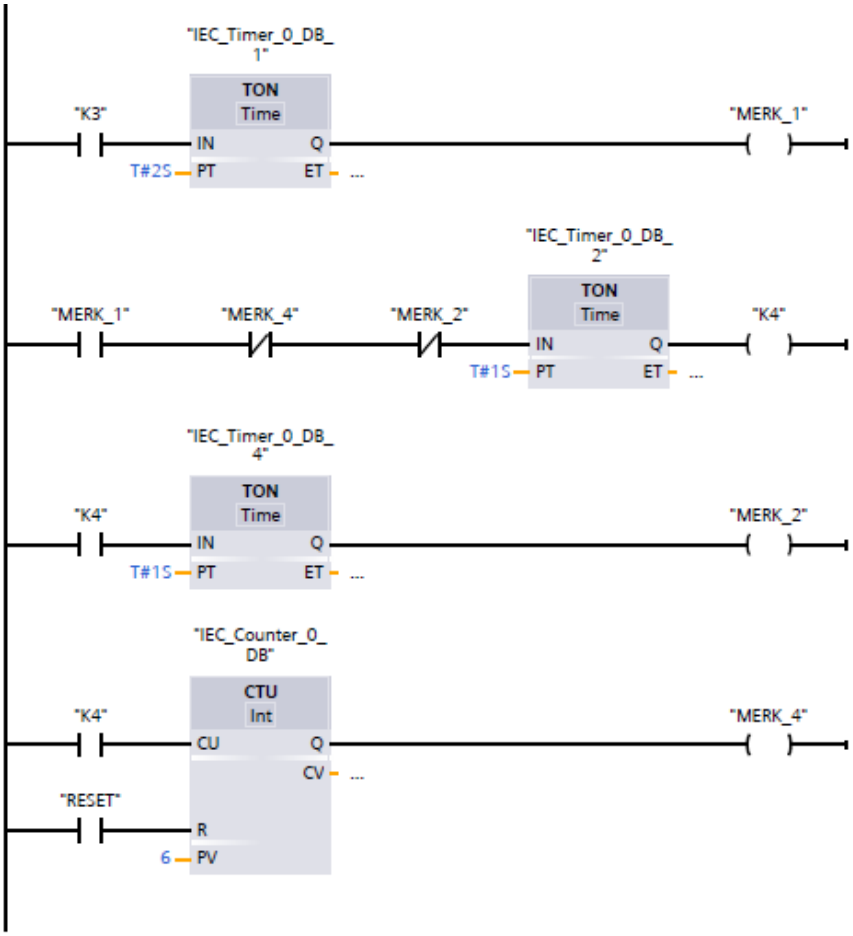


Fig.224 :Esercizio 30(parte 2) sull'automazione industriale

Simbolo	Indirizzo	Tipo	Commento
"K2"	%Q0.1	Bool	
"K3"	%Q0.2	Bool	
"MERK_1"	%M0.1	Bool	
"K4"	%Q0.3	Bool	
T#1S	T#1S	Time	
"MERK_2"	%M0.2	Bool	
"RESET"	%I0.2	Bool	
"IEC_Counter_0_DB"	%DB4	IEC_Counter	
"IEC_Timer_0_DB_1"	%DB2	IEC_Timer	
"IEC_Timer_0_DB_4"	%DB6	IEC_Timer	
"IEC_Timer_0_DB"	%DB1	IEC_Timer	
T#5S	T#5S	Time	
6	6	Int	
"S2"	%I0.1	Bool	
"MERK_5"	%M0.6	Bool	
"MERK_3"	%M0.3	Bool	
"MERK_4"	%M0.7	Bool	
T#2S	T#2S	Time	
"IEC_Timer_0_DB_2"	%DB3	IEC_Timer	

Combinazione logica a parola, AND / OR / XOR

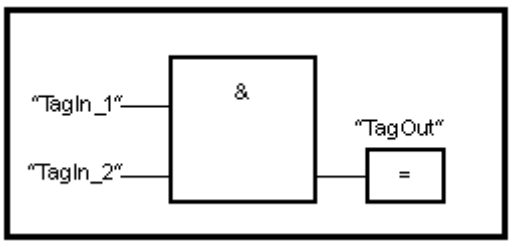


Fig.226 :Blocco AND/OR/XOR

esperienza 45
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

Le istruzioni "AND", "OR" e "XOR" consentono di combinare il valore dell'ingresso IN1 con quello dell'ingresso IN2 e di leggere il risultato nell'uscita OUT.

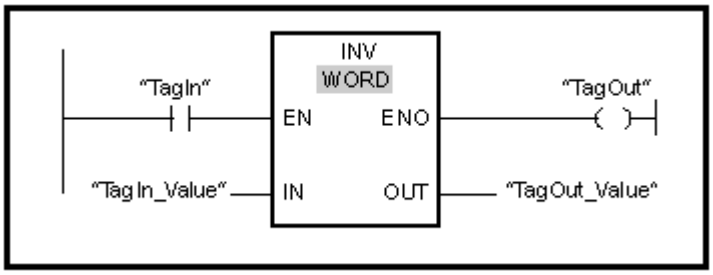
Durante l'esecuzione dell'istruzione il bit 0 del valore nell'ingresso IN1 viene combinato con il bit 0 del valore nell'ingresso IN2. Il risultato viene memorizzato nel bit 0 dell'uscita OUT. La stessa combinazione logica viene eseguita per tutti gli ulteriori bit del valore indicato.

Il numero di ingressi del box dell'istruzione può essere aumentato. Gli ingressi inseriti vengono numerati in ordine progressivo nel box. Quando viene eseguita l'istruzione vengono combinati i valori di tutti i parametri di ingresso disponibili. Il risultato viene salvato nell'uscita OUT.

L'istruzione viene eseguita solo se lo stato di segnale nell'ingresso di abilitazione EN è "1". In questo caso anche l'uscita ENO ha lo stato di segnale "1".

Se lo stato di segnale dell'ingresso di abilitazione EN è "0", anche l'uscita di abilitazione ENO ha lo stato di segnale "0".

Combinazione logica a parola, INV



esperienza 46
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

Fig.227 :Blocco INV

L'istruzione "INV" consente di invertire lo stato del segnale dei bit nell'ingresso IN.

Durante l'esecuzione dell'istruzione il valore nell'ingresso IN viene combinato tramite OR esclusivo con una maschera esadecimale (W#16#FFFF per numeri a 16 bit o DW#16#FFFF FFFF per numeri a 32 bit). Lo stato di segnale dei singoli bit viene così invertito ed emesso nell'uscita OUT.

L'istruzione viene eseguita solo se lo stato di segnale nell'ingresso di abilitazione EN è "1". In questo caso anche l'uscita ENO ha lo stato di segnale "1".

Se lo stato di segnale dell'ingresso di abilitazione EN è "0", anche l'uscita di abilitazione ENO ha lo stato di segnale "0".

Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

Combinazione logica a parola, ENCO

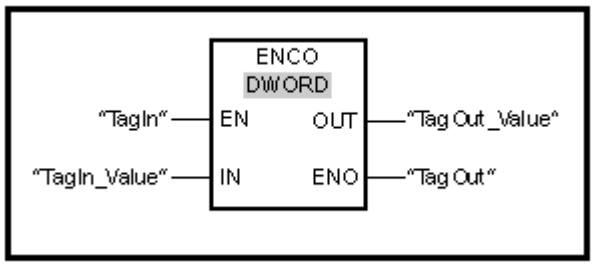


Fig.228 :Blocco ENCO

esperienza 47
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

L'istruzione "Enco" consente di leggere il numero del bit impostato sul valore più basso nel valore di ingresso e di emetterlo nell'uscita OUT.

L'istruzione "Enco" viene avviata solo se lo stato di segnale nell'ingresso di abilitazione EN è "1". Se non si verificano errori durante l'esecuzione anche l'uscita ENO ha lo stato di segnale "1".

Se lo stato di segnale dell'ingresso di abilitazione EN è "0", anche l'uscita di abilitazione ENO ha lo stato di segnale "0".

Esempio:
Se EN viene settato ad "1", leggo il valore messo in ingresso (1000010010011100). In questo caso il valore che leggero in uscita sarà "2", in quanto il primo "1" nella mia variabile occupa la 2° posizione.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Combinazione logica a parola, DECO

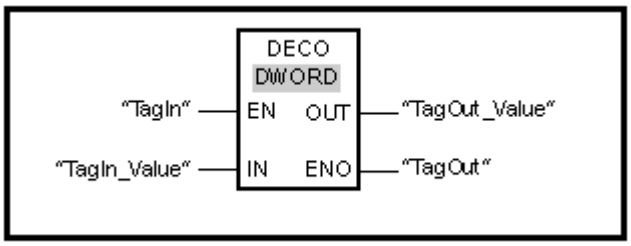


Fig.229 :Blocco DECO

esperienza 48
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

L'istruzione "Deco" consente di impostare nel valore di uscita un bit predefinito dal valore di ingresso.

L'istruzione "Deco" legge il valore nell'ingresso IN e imposta nel valore di uscita il bit la cui posizione corrisponde al valore letto. Gli ulteriori bit del valore di uscita vengono sostituiti con degli zeri. Se il valore nell'ingresso IN è maggiore di 31, viene eseguita un'istruzione modulo 32.

È possibile selezionare il tipo di dati dell'uscita, il quale può essere I, Q, M, D o L.

L'istruzione "Deco" viene avviata solo se lo stato di segnale nell'ingresso di abilitazione EN è "1". Se non si verificano errori durante l'esecuzione anche l'uscita ENO ha lo stato di segnale "1".

Se lo stato di segnale dell'ingresso di abilitazione EN è "0", anche l'uscita di abilitazione ENO ha lo stato di segnale "0".

Esempio:
Se EN viene settato ad "1", leggo il valore messo in ingresso (4). In questo caso il valore che leggero in uscita sarà "00010000", l'unico bit settato sarà quello della posizione indicata, tutti gli altri saranno resettati.

Combinazione logica a parola, SEL

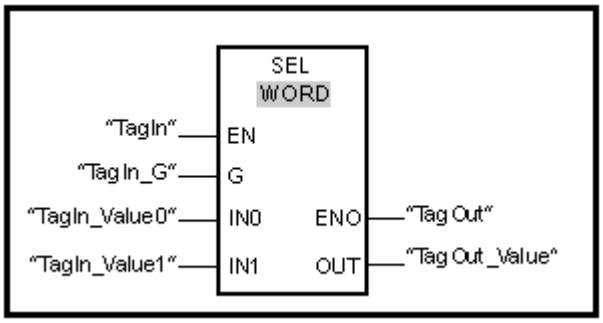


Fig.230 :Blocco SEL (Select)

esperienza 49
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

L'istruzione "Sel" sceglie in funzione di un interruttore (ingresso G) uno degli ingressi IN0 o IN1 e ne copia il contenuto nel parametro OUT.

Se l'ingresso G fornisce lo stato di segnale "0" viene copiato il valore nell'ingresso IN0. Se l'ingresso G ha lo stato di segnale "1", il valore dell'ingresso IN1 viene copiato nell'uscita OUT.

L'esecuzione dell'istruzione presuppone che l'ingresso di abilitazione EN presenti lo stato del segnale "1" e che le variabili abbiano lo stesso tipo di dati in tutti i parametri. Se non si verificano errori durante l'esecuzione anche l'uscita di abilitazione ENO ha lo stato di segnale "1".

L'uscita di abilitazione ENO viene resettata se l'ingresso di abilitazione EN ha lo stato di segnale "0" o se si verificano errori nel corso dell'esecuzione dell'istruzione.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Combinazione logica a parola, MUX

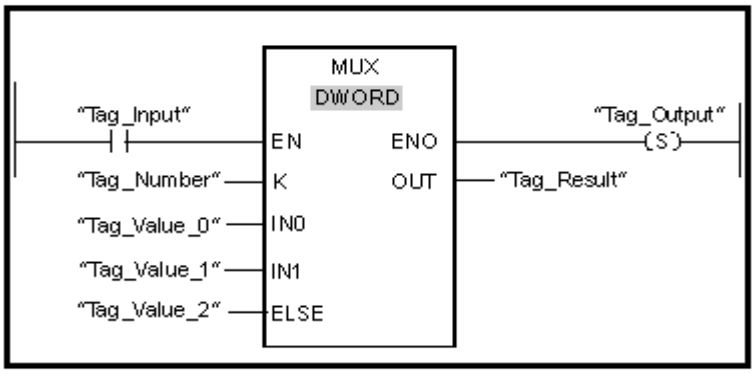


Fig. 231 :Blocco MUX

esperienza 50
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

L'istruzione "Mux" consente di copiare nell'uscita OUT il contenuto di un ingresso selezionato.

Il numero degli ingressi selezionabili del box dell'istruzione può essere ampliato. Gli ingressi vengono numerati automaticamente nel box. La numerazione inizia da IN0 e prosegue in ordine crescente con ogni nuovo ingresso. Con il parametro K si determina l'ingresso di cui copiare il contenuto nell'uscita OUT. Se il valore del parametro K è maggiore del numero degli ingressi disponibili, il contenuto del parametro ELSE viene copiato nell'uscita OUT e viene assegnato all'uscita di abilitazione ENO lo stato di segnale "0".

L'istruzione "Mux" può essere eseguita solo se le variabili di tutti gli ingressi e dell'uscita OUT hanno lo stesso tipo di dati. Fa eccezione il parametro K per il quale si possono indicare solo numeri interi.

L'istruzione viene eseguita solo se lo stato di segnale nell'ingresso di abilitazione EN è "1". Se non si verificano errori durante l'esecuzione anche l'uscita ENO ha lo stato di segnale "1".

L'uscita di abilitazione "ENO" viene resettata se viene soddisfatta una delle seguenti condizioni:

- L'ingresso di abilitazione "EN" ha lo stato di segnale "0".
- Il valore del parametro K è maggiore del numero di ingressi disponibili.
- Si verificano errori durante l'esecuzione dell'istruzione.

Combinazione logica a parola, DEMUX

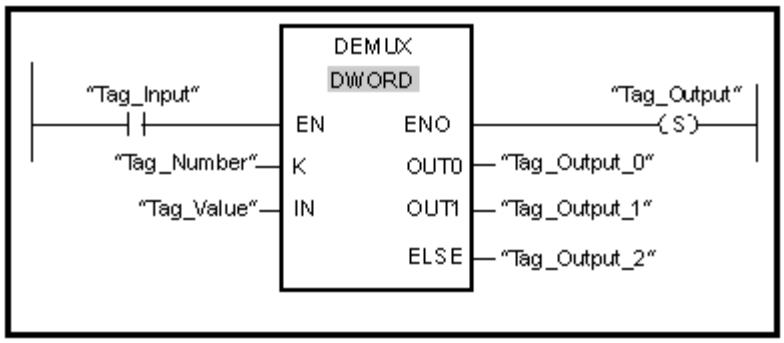


Fig.232 :Blocco DEMUX

esperienza 51
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

L'istruzione "Demux" consente di copiare il valore messo in ingresso in un uscita selezionata.

L'istruzione "Demux" è esattamente identica, l'unica differenza è che ho un solo ingresso e molteplici uscite selezionabili. Il parametro "K" indica in quale uscita verrà scritto il valore messo in ingresso, nel caso sia più grande del numero di uscite disponibili verrà copiato nell'uscita "ELSE". Per il resto le regole da rispettare e il principio di funzionamento è il medesimo.

L'istruzione viene eseguita solo se lo stato di segnale nell'ingresso di abilitazione EN è "1". Se non si verificano errori durante l'esecuzione anche l'uscita ENO ha lo stato di segnale "1".

L'uscita di abilitazione "ENO" viene resettata se viene soddisfatta una delle seguenti condizioni:

- L'ingresso di abilitazione "EN" ha lo stato di segnale "0".
- Il valore del parametro K è maggiore del numero di ingressi disponibili.
- Si verificano errori durante l'esecuzione dell'istruzione.

Spostamento e rotazione, SHR / SHL

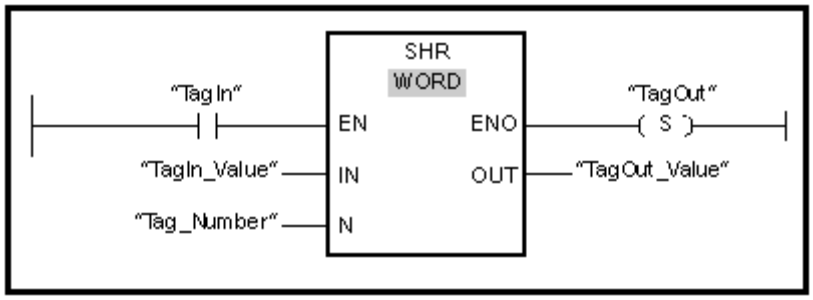


Fig.233 :Blocco SHR(Shift Right)/SHL(Shift Left)

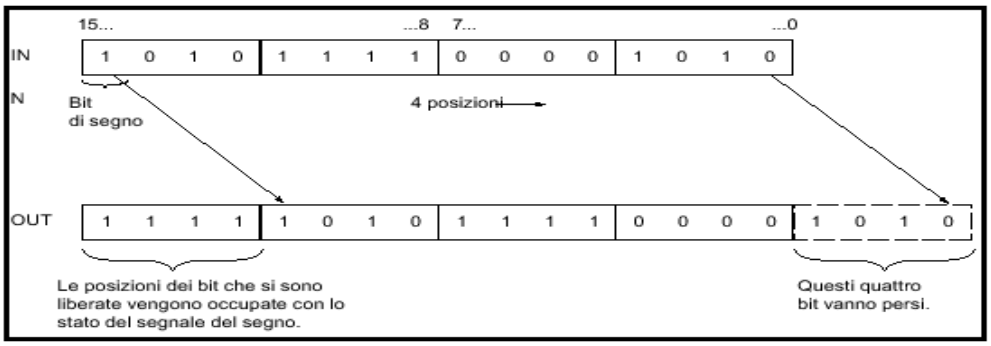
esperienza 52
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

Le istruzioni "SHR" e "SHL" consentono di spostare a destra o a sinistra il contenuto dell'operando nell'ingresso IN bit per bit e di leggere il risultato nell'uscita OUT. Con il parametro N si definisce il numero di posizioni di bit di cui viene spostato il valore indicato.

Se il valore del parametro N ha lo stato di segnale "0", il valore dell'ingresso IN viene copiato nell'operando dell'uscita OUT.

Se il valore nel parametro N è maggiore del numero di posizioni di bit disponibili, il valore dell'operando nell'ingresso IN viene spostato verso destra/sinistra per il numero di posizioni di bit disponibili.Nel caso dei valori senza segno, le posizioni dei bit che si liberano con lo spostamento nel campo sinistro/destro dell'operando vengono occupate da zeri.

In caso di spostamento verso destra se il valore indicato è dotato di segno, le posizioni dei bit libere vengono occupate con lo stato del segnale del bit del segno.La seguente figura mostra come il contenuto di un operando con tipo di dati Integer viene spostata verso destra di quattro posizioni di bit:



Perché sia possibile eseguire le istruzioni è necessario che l'ingresso di abilitazione EN abbia lo stato di segnale "1". In questo caso anche l'uscita di abilitazione ENO ha lo stato di segnale "1".Se lo stato di segnale dell'ingresso di abilitazione EN è "0", anche l'uscita di abilitazione ENO ha lo stato di segnale "0".

Spostamento e rotazione, ROR / ROL

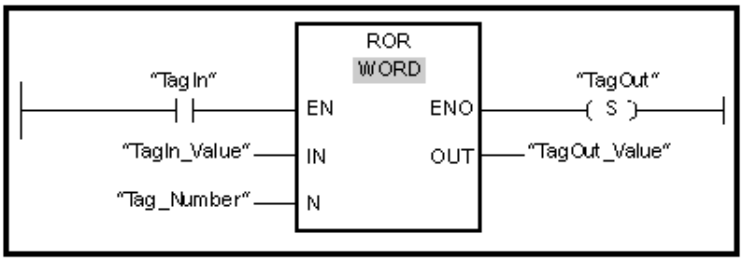


Fig.235 :Blocco ROR(Rotate Right)/ROL(Rotate Left)

esperienza 53
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

Le istruzioni "ROR" e "ROL" consentono di far ruotare a destra o sinistra il contenuto dell'operando nell'ingresso IN bit per bit e di leggere il risultato nell'uscita OUT.

Con il parametro N si definisce il numero di posizioni di bit di cui viene fatto ruotare il valore indicato. Le posizioni di bit che si liberano con la rotazione vengono occupate dalle posizioni dei bit spostati.

Se il valore del parametro N ha lo stato di segnale "0", il valore dell'ingresso IN viene copiato nell'operando dell'uscita OUT.

Se il valore nel parametro N è maggiore del numero di posizioni di bit disponibili, il valore dell'operando nell'ingresso IN viene fatto ugualmente ruotare per il numero di posizioni di bit indicati.

La seguente figura mostra come il contenuto di un operando con tipo di dati DWORD venga fatto ruotare verso destra di tre posizioni di bit:

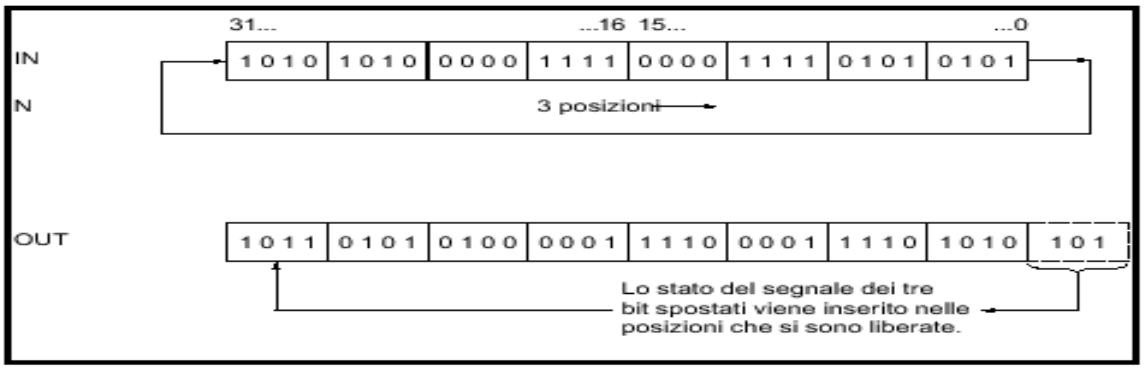


Fig.236 :Operando con tipo di dati DWORD ruotato a destra di 3 posizioni

Perché sia possibile eseguire le istruzioni è necessario che l'ingresso di abilitazione EN abbia lo stato di segnale "1". In questo caso anche l'uscita di abilitazione ENO ha lo stato di segnale "1".Se lo stato di segnale dell'ingresso di abilitazione EN è "0", anche l'uscita di abilitazione ENO ha lo stato di segnale "0".

Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

Data e ora, T_CONV

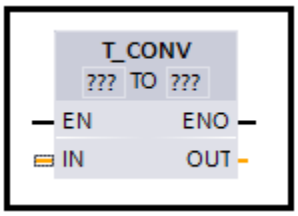


Fig.237 :Blocco T_CONV(Convert)

esperienza 54
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

Questa istruzione consente di convertire il tipo di dati nel parametro di ingresso IN nel tipo di dati che viene emesso nell'uscita OUT .

I box dell'istruzione dell'ingresso e dell'uscita permettono di selezionare i formati di dati per la conversione.

Parametri	**Dichiarazioni**	**Tipo di dati**	**Area di memoria**	**Descrizione**
IN	Input	TIME,DINT	I,Q,M,D,L o costante	Valore da convertire
OUT	Return	TIME,DINT	I,Q,M,D,	Risultato dela conversione

Data e ora, T_ADD

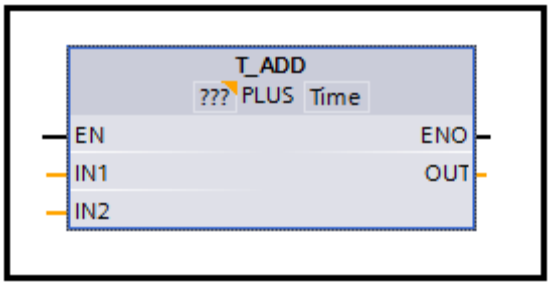


Fig.238 :Blocco T_ADD

esperienza 55
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

Questa istruzione consente di sommare il valore di tempo indicato nel parametro di ingresso IN1 a quello indicato nel parametro di ingresso IN2.

Il risultato può essere letto nel parametro di uscita OUT. È possibile sommare i seguenti formati:

- Somma di una durata (TIME) con un'altra (TIME). Il risultato può essere emesso in una variabile con formato TIME.
- Somma di una durata (TIME) con una data e ora (DTL). Il risultato può essere emesso in una variabile con formato DTL.

I formati dei valori del parametro di ingresso IN1e del parametro di uscita OUT possono essere determinati selezionando i tipi di dati dell'ingresso e dell'uscita dell'istruzione. Nel parametro di ingresso IN2 si possono indicare solo valori di tempo con formato TIME.

Data e ora, T_SUB

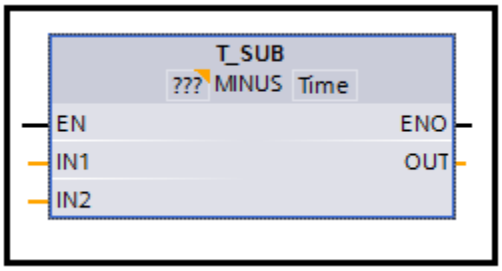


Fig.239 :T_SUB(Subtract)

esperienza 56
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

Questa istruzione consente di sottrarre il valore di tempo indicato nel parametro di ingresso IN2 da quello indicato nel parametro di ingresso IN1.

La differenza può essere letta nel parametro di uscita OUT. È possibile sottrarre i seguenti formati:

- Sottrazione di una durata (TIME) da un'altra (TIME). Il risultato può essere emesso in una variabile del tipo di dati TIME.
- Sottrazione di una durata (TIME) da una data e ora (DTL). Il risultato può essere emesso in una variabile del tipo di dati DTL.

I formati dei valori del parametro di ingresso IN1 e del parametro di uscita OUT possono essere determinati selezionando i tipi di dati del parametro di ingresso e del parametro di uscita dell'istruzione. Nel parametro di ingresso IN2 si possono indicare solo valori di tempo con formato TIME.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Data e ora, T_DIFF

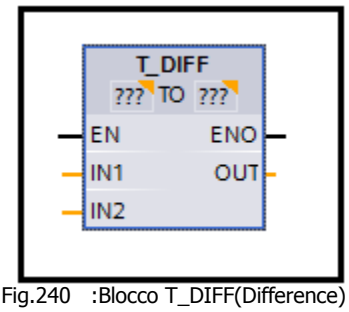


Fig.240 :Blocco T_DIFF(Difference)

esperienza 57
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

Questa istruzione consente di sottrarre il valore di tempo indicato nel parametro di ingresso IN2 da quello indicato nel parametro di ingresso IN1.

La differenza tra questo blocco e il precedente è che il risultato viene emesso nel parametro di uscita OUT nel formato **TIME**. Nei parametri di ingresso IN1 e IN2 si possono indicare **solo valori con formato DTL**. Questo blocco esegue una sottrazione fra due date, dando in uscita un valore che indica il lasso di tempo.

Se il valore di tempo indicato nel parametro di ingresso IN2 è maggiore di quello indicato nel parametro di ingresso IN1 , il risultato emesso nel parametro di uscita OUT sarà costituito da un valore negativo.

Data e ora, WR_SYS_T

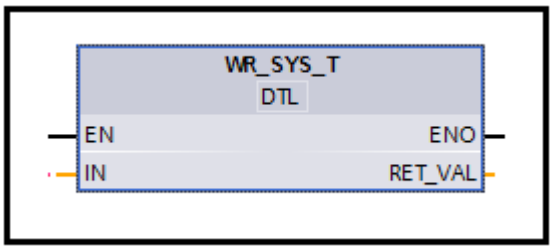


Fig.241 :Blocco Write_System_Time

esperienza 58
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

Questa istruzione consente di impostare la data e l'ora dell'orologio della CPU.

La data e l'ora vanno specificate in formato DTL nel parametro di ingresso IN dell'istruzione. Leggendo il parametro di uscita RET_VAL è possibile rilevare se si sono verificati errori durante l'esecuzione dell'istruzione.

L'istruzione "WR_SYS_T" non può essere utilizzata per trasferire i dati relativi al fuso orario o all'ora legale.

Parametro RET_VAL:

Codice dell'errore (W#16#....)	Descrizione
0000	Nessun errore
8081	Anno non valido
8082	Mese non valido
8083	Giorno non valido
8084	Indicazione delle ore non valida
8085	Indicazione dei minuti non valida
8086	Indicazione dei secondi non valida
8087	Indicazione dei nanosecondi non valida
80B0	Guasto dell'orologio hardware

Data e ora, RD_SYS_T

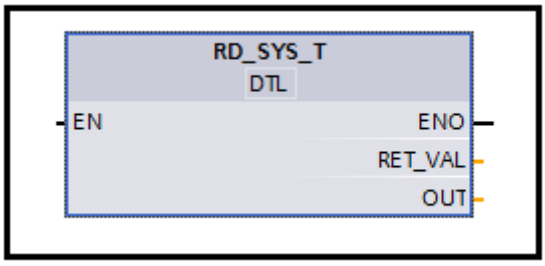


Fig.242 :Blocco Read_System_Time

esperienza 59
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

Questa istruzione consente di leggere la data e l'ora attuali dell'orologio della CPU.

I dati letti vengono emessi in formato DTL nel parametro di uscita OUT dell'istruzione. Il valore emesso non contiene indicazioni sul fuso orario o l'ora legale. Leggendo l'uscita RET_VAL è possibile rilevare se si sono verificati errori durante l'esecuzione dell'istruzione.

Parametro RET_VAL:

Codice dell'errore (W#16#....)	**Descrizione**
0000	Nessun errore
8222	Il risultato non è compreso entro il campo di valori ammesso
8223	Non è possibile memorizzare il risultato con il tipo di dati indicato

Data e ora, RD_LOC_T

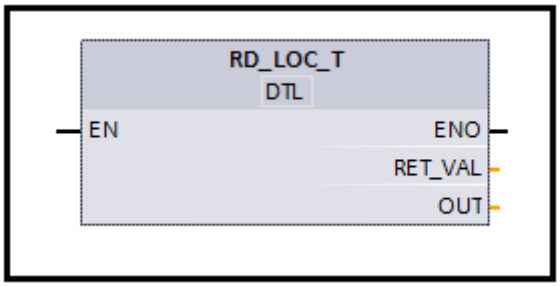


Fig.243 :Blocco Read_Local_Time

esperienza 60
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

Questa istruzione consente di leggere l'ora locale corrente dell'orologio della CPU e di emetterla in formato DTL nell'uscita OUT.

Per l'emissione dell'ora locale vengono utilizzati i dati relativi al fuso orario e all'inizio dell'ora legale e dell'ora solare configurati per l'orologio della CPU.

Parametro RET_VAL:

Codice dell'errore (W#16#....)	Descrizione
0000	Nessun errore
0001	Nessun errore. L'ora locale viene emessa come ora legale.
8080	L'ora locale non è leggibile.

String + Char, S_CONV

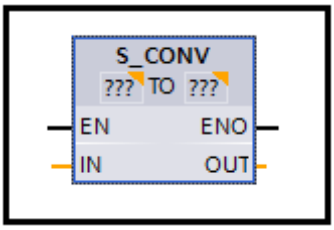


Fig.244 :Blocco String + Char,S_CONV

esperienza 61
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

Questa istruzione consente di convertire il valore dell'ingresso IN nel formato di dati indicato nell'uscita OUT.

Sono possibili i seguenti tipi di conversione:

- Conversione di una stringa di caratteri (STRING) in un valore numerico

La conversione viene effettuata per tutti i caratteri della stringa indicata nel parametro di ingresso IN. Sono ammesse le cifre da "0" a "9", il punto decimale e i caratteri più e meno. Il primo carattere della sequenza può essere costituito da una cifra valida o da un segno. Gli spazi iniziali e gli esponenti vengono ignorati.
La conversione dei caratteri può essere interrotta dalla presenza di caratteri non validi. Il formato della conversione può essere determinato selezionando il tipo di dati per il parametro di uscita OUT.

- Conversione di un valore numerico in una stringa di caratteri (STRING)

Il formato del valore numerico da convertire può essere determinato selezionando il tipo di dati per l'ingresso IN. Nell'uscita OUT si deve indicare una variabile valida con tipo di dati STRING. La lunghezza della stringa di caratteri dopo la conversione dipende dal valore dell'ingresso IN. Il risultato della conversione viene salvato a partire dal terzo byte della stringa di caratteri. Nel primo byte della stringa di caratteri viene rilevata la lunghezza massima della stringa mentre nel secondo quella effettiva. I valori numerici positivi vengono emessi senza segno.

- Copia di una stringa

Se nel parametro di ingresso e di uscita dell'istruzione si immette il tipo di dati STRING, la stringa viene copiata dall'ingresso IN all'uscita OUT. Se la lunghezza effettiva della stringa di caratteri nell'ingresso IN supera la lunghezza massima della stringa nell'uscita OUT, viene copiata la massima parte di stringa di IN che può essere contenuta dalla stringa di OUT.

String + Char, STRG_VAL

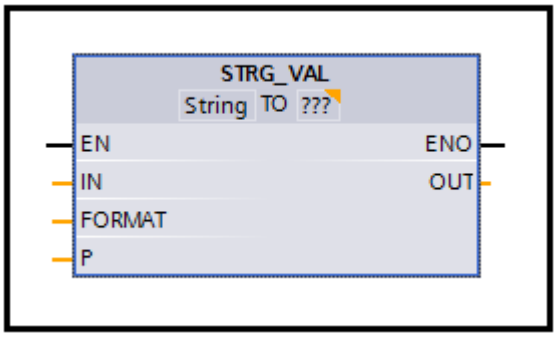


Fig.245 :Blocco String + Char,STRG_VAL

esperienza 62
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

L'istruzione "STRG_VAL" consente di convertire una stringa di caratteri numerici nella corrispondente rappresentazione in numeri interi o in numeri a virgola mobile:

- La stringa di caratteri da convertire va indicata nel parametro di ingresso IN.
- Il formato del valore ottenuto può essere determinato selezionando il tipo di dati per il parametro di uscita OUT. Il risultato può essere letto nel parametro di uscita OUT.

I caratteri ammessi per la conversione sono le cifre da "0" a "9", il punto decimale, la virgola decimale, le notazioni "E" ed "e" e i segni più e meno. Se si immettono caratteri non validi la conversione viene interrotta.

Il parametro FORMAT consente di impostare in che modo dovranno essere interpretati i caratteri della stringa. Nel parametro FORMAT si possono indicare solo variabili con tipo di dati USINT.

La seguente tabella indica i possibili valori del parametro FORMAT e il relativo significato:

Valore (W#16#....)	**Notazione**	**Rappresentazione decimale**
0000	Frazione decimale	"."
0001		","
0002	Esponenziale	"."
0003		","
0004 - FFFF	Valori non validi	

La conversione inizia dal carattere di cui è stata indicata la posizione nel parametro P. Se, ad esempio, nel parametro P è stato indicato il valore "1", la conversione viene effettuata a partire dal primo carattere della stringa di caratteri indicata. Il valore "0" o un valore maggiore della lunghezza della stringa non è valido.

La seguente tabella riporta alcuni esempi di conversione di una sequenza di caratteri in valore numerico:

IN (STRING)	FORMAT (W#16#....)	OUT (tipo di dati)	OUT (valore)	Stato ENO
'123'	0000	INT/DINT	123	1
'-00456'	0000	INT/DINT	-456	1
'123.45'	0000	INT/DINT	123	1
'+2345'	0000	INT/DINT	2345	1
'00123AB'	0000	INT/DINT	123	1
'123'	0000	REAL	123.0	1
'-00456'	0001	REAL	-456.0	1
'+00456'	0001	REAL	456.0	1
'123.45'	0000	REAL	123.45	1
'123.45'	0001	REAL	12345.0	1
'123,45'	0000	REAL	12345.0	1
'123,45'	0001	REAL	123.45	1
'.00123AB'	0001	REAL	123.0	1
'1.23e-4'	0000	REAL	1.23	1
'1.23E-4'	0000	REAL	1.23	1
'1.23E-4'	0002	REAL	1.23E-4	1
'12,345.67'	0000	REAL	12345.67	1
'12,345.67'	0001	REAL	12.345	1
'3.4e39'	0002	REAL	W#16#7F800000	1
'-3.4e39'	0002	REAL	W#16#FF800000	1
'1.1754943e-38'	0002	REAL	0.0	1
'12345'	-/-	SINT	0	0
'A123'	-/-	-/-	0	0
''	-/-	-/-	0	0
'++123'	-/-	-/-	0	0

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

String + Char, VAL_STRG

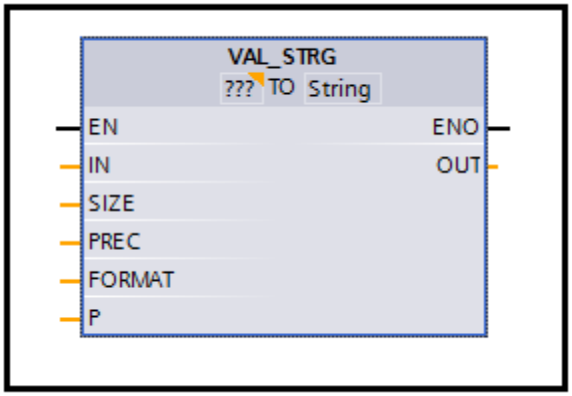


Fig.246 :Blocco String + Char,VAL_STRG

esperienza 63
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

Questa istruzione consente di convertire un valore numerico in una stringa di caratteri.

Il valore da convertire va indicato nel parametro di ingresso IN. Il formato del valore numerico può essere determinato selezionando il tipo di dati.

Il risultato della conversione può essere letto nel parametro di uscita OUT.

I caratteri ammessi per la conversione sono le cifre da "0" a "9", il punto decimale, la virgola decimale, le notazioni "E" ed "e" e i segni più e meno. Se si immettono caratteri non validi la conversione viene interrotta.

Il parametro P consente di indicare a partire da quale carattere della stringa debba essere scritto il risultato. Se, ad esempio, nel parametro P è stato indicato il valore "2", il valore convertito viene memorizzato a partire dal secondo carattere della stringa.

Il parametro SIZE consente di stabilire quanti caratteri della stringa vengono scritti. Il conteggio inizia dal carattere indicato nel parametro P. Se il valore di uscita è più breve della lunghezza indicata, il risultato viene scritto nella sequenza allineato a destra. Le posizioni vuote vengono riempite con caratteri di spaziatura.

Il parametro FORMAT consente di indicare come interpretare il valore numerico durante la conversione e come scriverlo nella stringa di caratteri. Nel parametro FORMAT si possono indicare solo variabili con tipo di dati USINT.

La seguente tabella indica i possibili valori del parametro FORMAT e il relativo significato:

Valore (W#16#....)	Notazione	Segno	Rappresentazione decimale
0000	Frazione decimale	"-"	"."
0001			","
0002	Esponenziale		"."
0003			","
0004	Frazione decimale	"+" e "-"	"."
0005			","
0006	Esponenziale		"."
0007			","
0008 - FFFF	Valori non validi		

Il parametro PREC definisce il numero di cifre decimali per la conversione dei numeri in virgola mobile. Viene supportata al massimo una precisione di 7 cifre per i valori numerici con tipo di dati REAL. Se il valore da convertire è un numero intero, il parametro PREC consente di stabilire dove viene posizionato il punto decimale.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

String + Char, LEN

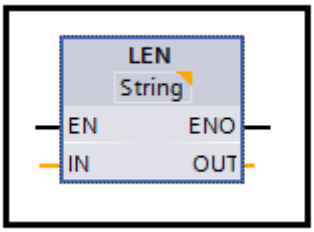


Fig. 247 :Blocco String+Char,LEN

esperienza 64
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

Le variabili di tipo STRING contengono due lunghezze: la lunghezza massima e la lunghezza attuale (corrispondente al numero di caratteri attualmente validi). La lunghezza massima della stringa viene specificata tra parentesi quadre nella parola chiave STRING di ciascuna variabile. La lunghezza attuale rappresenta il numero di posizioni effettivamente occupate dai caratteri. La lunghezza attuale è inferiore o uguale a quella massima. Il numero di byte occupati da una sequenza è il doppio della lunghezza massima.

Questa istruzione consente di leggere nel parametro di ingresso IN la lunghezza attuale della stringa indicata e di specificarla sotto forma di valore numerico nel parametro di uscita OUT.

Le stringhe vuote ('') hanno lunghezza zero. Se si verificano errori durante l'esecuzione dell'istruzione viene emessa una stringa vuota.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

String + Char, CONCAT

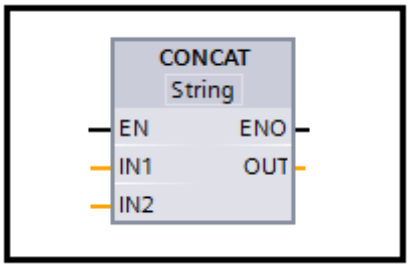


Fig.248 :Blocco String+ Char,CONCAT

esperienza 65
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

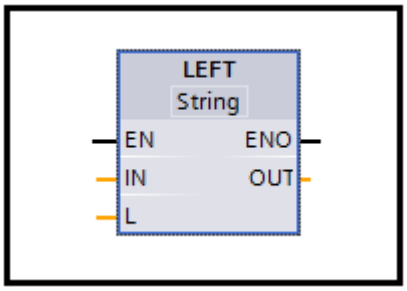
Questa istruzione consente di collegare la stringa di caratteri indicata nel parametro di ingresso IN1 con quella indicata nel parametro di ingresso IN2. Il risultato viene emesso nel parametro di uscita OUT in formato STRING.

Se la stringa di caratteri del risultato è più lunga della variabile indicata nel parametro di uscita OUT, viene limitata alla lunghezza disponibile.

Se si verificano errori durante l'esecuzione dell'istruzione e il parametro di uscita OUT è editabile, viene emessa una stringa vuota.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

String + Char, LEFT



esperienza 66
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

Fig.249 :Blocco String + Char,LEFT

Questa istruzione consente di estrarre una parte della stringa di caratteri nel parametro di ingresso IN a partire dal suo primo carattere. Il numero di caratteri da estrarre va definito nel parametro L. I caratteri estratti vengono emessi nel parametro di uscita OUT in formato STRING.

Se il numero di caratteri da estrarre supera la lunghezza attuale della stringa, il parametro di uscita OUT restituisce come risultato la stringa di ingresso. Se il parametro L ha valore "0" o se il valore di ingresso è una stringa vuota, viene restituita una stringa vuota. Se il parametro L ha un valore negativo, viene restituita una stringa vuota.

Se si verificano errori durante l'esecuzione dell'istruzione e il parametro di uscita OUT è editabile, viene emessa una stringa vuota.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

String + Char, RIGHT

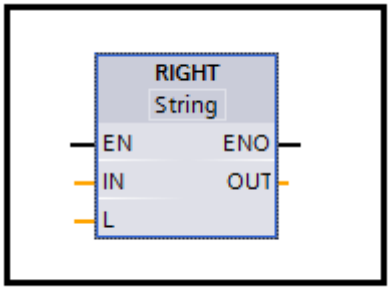


Fig.250 :Blocco String+Char,RIGHT

esperienza 67
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

Questa istruzione consente di estrarre gli ultimi caratteri L di una stringa di caratteri nel parametro di ingresso IN. Il numero di caratteri da estrarre va definito nel parametro L. I caratteri estratti vengono emessi nel parametro di uscita OUT in formato STRING.

Se il numero di caratteri da estrarre supera la lunghezza attuale della stringa, il parametro di uscita OUT restituisce come risultato la stringa di ingresso. Se il parametro L ha valore "0" o se il valore di ingresso è una stringa vuota, viene restituita una stringa vuota. Se il parametro L ha un valore negativo, viene restituita una stringa vuota.

Se si verificano errori durante l'esecuzione dell'istruzione e il parametro di uscita OUT è editabile, viene emessa una stringa vuota.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

String + Char, MID

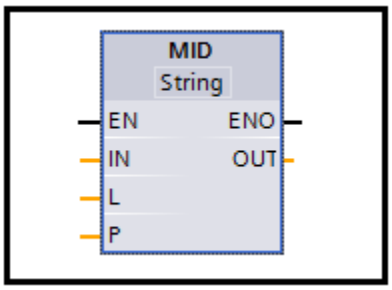


Fig.251 :Blocco String+ Char,MID

esperienza 68
Carloni Lorenzo
5AET 2013
(lorenzo.carloni1@gmail.com)

Questa istruzione consente di estrarre una parte della stringa di caratteri nel parametro di ingresso IN. Il parametro P consente di stabilire la posizione del primo carattere da estrarre. La lunghezza della stringa di caratteri da estrarre va definita con il parametro L. La parte di stringa estratta viene emessa nel parametro di uscita OUT.

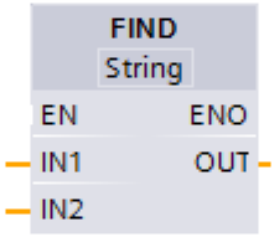
Se il numero di caratteri da estrarre supera la lunghezza attuale della stringa del parametro di ingresso IN, viene emessa la parte di stringa che va dalla posizione indicata in P fino alla fine.

Se la posizione indicata nel parametro P non rientra nella lunghezza attuale della stringa nel parametro di ingresso IN, nel parametro di uscita OUT viene emessa una stringa vuota.

Se il valore del parametro P o L è pari a zero o negativo, nel parametro di uscita OUT viene emessa una stringa vuota.

Se si verificano errori durante l'esecuzione dell'istruzione e il parametro di uscita OUT è editabile, viene emessa una stringa vuota.

String + Char, Find



esperienza 69

Severi Michele
Brighi Nicolas
5AET 2013
(michele.severi@gmail.com)
(nicolas.brighi2@gmail.com)

Fig.252 :Blocco FIND

Questa istruzione consente di cercare un particolare carattere o una particolare sequenza nella stringa di caratteri del parametro di ingresso IN1.

Il valore da cercare va indicato nel parametro di ingresso IN2. La ricerca viene effettuata da sinistra a destra.

Nel parametro di uscita OUT viene emessa la posizione del primo elemento trovato. Se la ricerca non dà alcun risultato il parametro di uscita OUT emette il valore "0".
Se si verificano errori durante l'esecuzione dell'istruzione viene emessa una stringa vuota.

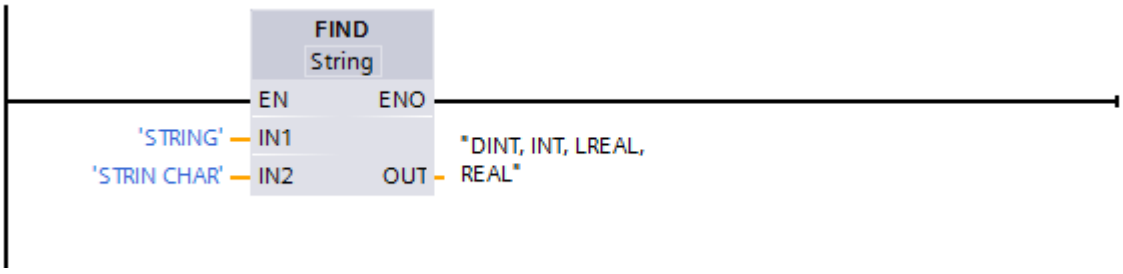


Fig.253 :Esercizio con blocco FIND

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

String + Char, Insert

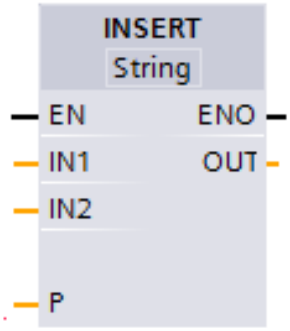


Fig.254 :Blocco INSERT

esperienza 70
Severi Michele
Brighi Nicolas
5AET 2013
(michele.severi@gmail.com)
(nicolas.brighi2@gmail.com)

Questa istruzione consente di inserire la stringa di caratteri indicata nel parametro di ingresso IN2 in quella indicata nel parametro di ingresso IN1. Con il parametro P si definisce la posizione del carattere a partire dal quale viene effettuato l'inserimento. Il risultato viene emesso nel parametro di uscita OUT nel formato STRING.

Durante l'esecuzione di quest'istruzione è importante osservare le seguenti regole:

Se il valore del parametro P supera la lunghezza attuale della stringa di caratteri del parametro di ingresso IN1, la stringa del parametro di ingresso IN2 viene agganciata a quella del parametro di ingresso IN1.

Se il valore del parametro P è pari a zero, l'uscita OUT emette la stringa nel parametro IN2 seguita dalla stringa nel parametro IN1.

Se il parametro P ha un valore negativo, l'uscita OUT emette una stringa vuota.

Se la stringa di caratteri del risultato è più lunga della variabile indicata nel parametro di uscita OUT, viene limitata alla lunghezza disponibile.

Questa istruzione consente di cercare un particolare carattere o una particolare sequenza nella stringa di caratteri del parametro di ingresso IN1.

Il valore da cercare va indicato nel parametro di ingresso IN2. La ricerca viene effettuata da sinistra a destra.

Nel parametro di uscita OUT viene emessa la posizione del primo elemento trovato. Se la ricerca non dà alcun risultato il parametro di uscita OUT emette il valore "0".

Se si verificano errori durante l'esecuzione dell'istruzione viene emessa una stringa vuota.

String + Char, Delete

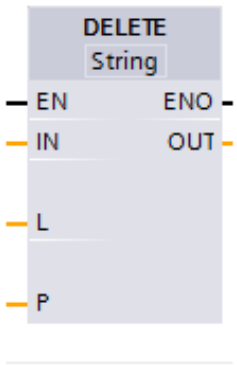


Fig.255 :Blocco DELETE

esperienza 71

Severi Michele
Brighi Nicolas
5AET 2013
(michele.severi@gmail.com)
(nicolas.brighi2@gmail.com)

Questa istruzione consente di cancellare una parte della stringa di caratteri nel parametro di ingresso IN. La posizione del primo carattere da cancellare va definita con il parametro P. Nel parametro L va indicato il numero di caratteri da cancellare. La parte rimanente della stringa viene emessa nel parametro di uscita OUT in formato STRING.

Durante l'esecuzione di quest'istruzione è importante osservare le seguenti regole:

Se il valore del parametro P è minore o uguale a zero, il parametro di uscita OUT emette una stringa di caratteri vuota.

Se il valore del parametro P è superiore alla lunghezza attuale della stringa nell'ingresso IN, viene restituita la stringa di ingresso nel parametro di uscita OUT.

Se il valore nel parametro L è pari a zero, nel parametro di uscita OUT viene restituita la stringa di ingresso.

Se il numero di caratteri da cancellare nel parametro L è superiore alla lunghezza della stringa nel parametro di ingresso IN, viene emessa una stringa vuota.

Se il parametro L ha un valore negativo, viene emessa una stringa vuota.

Se si verificano errori durante l'esecuzione dell'istruzione e il parametro di uscita OUT è editabile, viene emessa una stringa vuota.

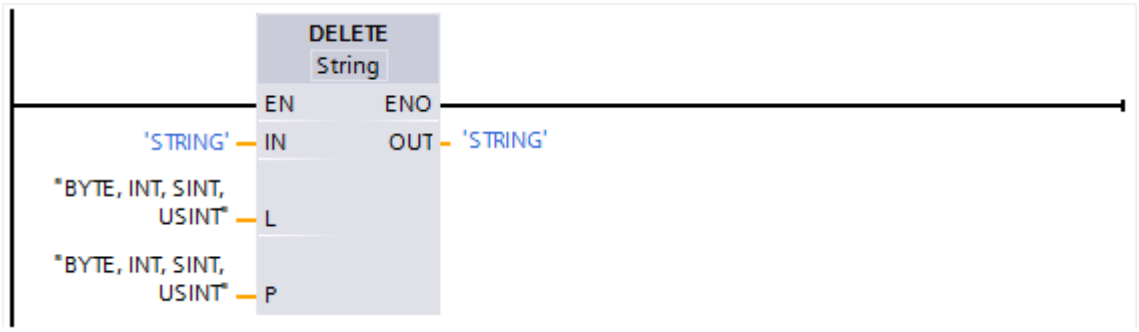
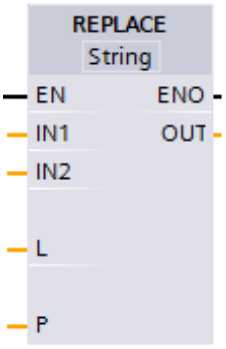


Fig.256 :Esercizio con blocco DELETE

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

String + Char, Replace



esperienza 72
Severi Michele
Brighi Nicolas
5AET 2013
(michele.severi@gmail.com)
(nicolas.brighi2@gmail.com)

Fig.257 :Blocco REPLACE

Questa istruzione consente di sostituire una parte della stringa di caratteri dell'ingresso IN1 con la stringa del parametro di ingresso IN2. La posizione del primo carattere da sostituire va definita con il parametro P. Il numero di caratteri da sostituire va indicato nel parametro L. Il risultato viene emesso nel parametro di uscita OUT in formato STRING.
Durante l'esecuzione di quest'istruzione è importante osservare le seguenti regole:
Se il valore del parametro P è minore o uguale a zero, il parametro di uscita OUT emette una stringa di caratteri vuota.
Se il valore del parametro L è minore di zero, il parametro di uscita OUT emette una stringa di caratteri vuota.
Se il valore del parametro P supera la lunghezza attuale della stringa di caratteri del parametro di ingresso IN1, nel parametro di uscita OUT viene scritto il contenuto della stringa del parametro IN1.
Se P è pari a 1, la stringa dell'ingresso IN1 viene sostituita a partire dal primo carattere (compreso).
Se il valore del parametro P supera la lunghezza attuale della stringa di caratteri del parametro di ingresso IN1, la stringa del parametro di ingresso IN2 viene agganciata a quella del parametro di ingresso IN1.
Se la stringa di caratteri del risultato è più lunga della variabile indicata nel parametro di uscita OUT, viene limitata alla lunghezza disponibile.

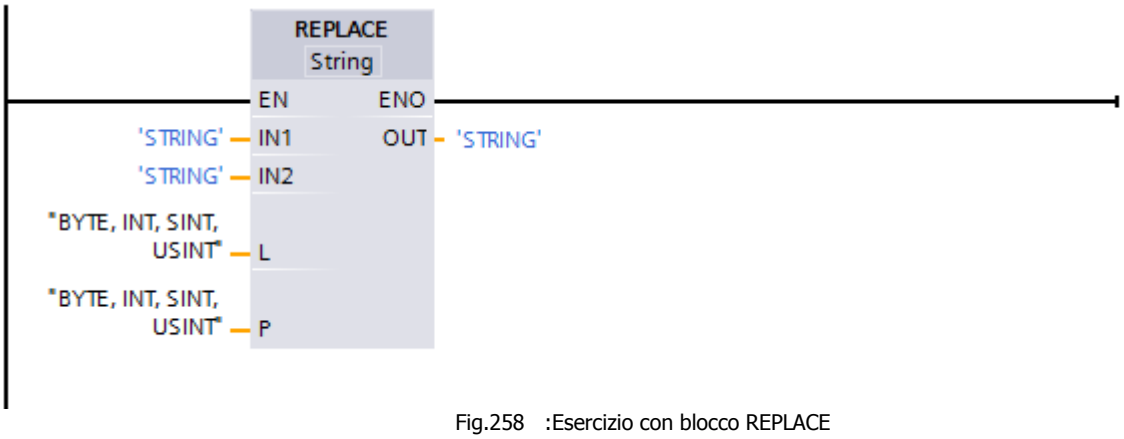


Fig.258 :Esercizio con blocco REPLACE

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Controllo del programma, Get_Error

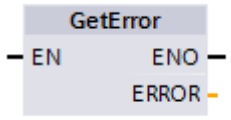


Fig.259 :Blocco GET_ERROR

esperienza 75
Severi Michele
Brighi Nicolas
5AET 2013
(michele.severi@gmail.com)
(nicolas.brighi2@gmail.com)

L'istruzione "Interroga errori localmente" consente di sapere se si sono verificati errori in un blocco. Se il sistema segnala che si sono verificati errori durante l'elaborazione di un blocco, memorizza un'informazione dettagliata sul primo errore nell'operando dell'uscita ERROR. Nell'uscita ERROR si possono indicare solo operandi con tipo di dati di sistema "ErrorStruct". "ErrorStruct" specifica con precisione la struttura in cui vengono memorizzate le informazioni sull'errore. Utilizzando altre istruzioni è possibile analizzare tale struttura e programmare una reazione. Una volta eliminato il primo errore l'istruzione fornisce informazioni sull'errore successivo.

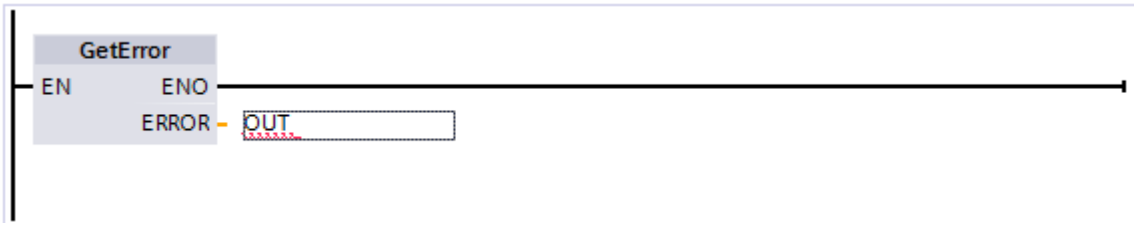


Fig:260 :Applicazione del blocco Get_Error

Esempio:
Il seguente esempio illustra il funzionamento dell'istruzione:

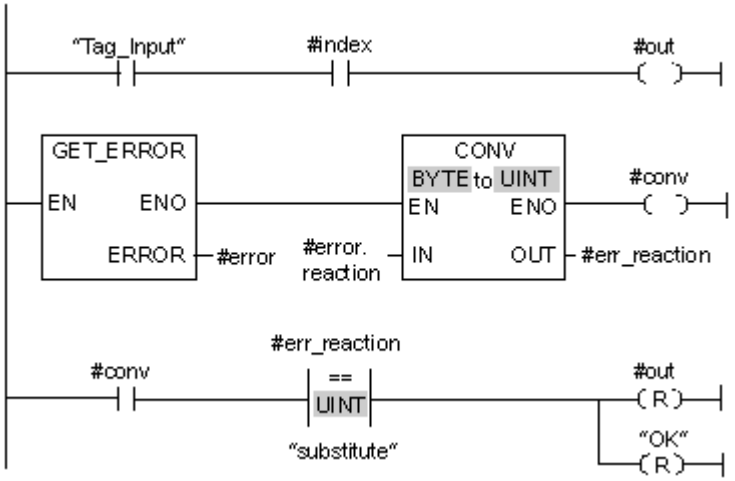


Fig:261 :Esempio illustrativo per il funzionamento del blocco GET_ERROR

Quando si verifica un errore l'istruzione "Interroga errori localmente" fornisce la relativa informazione alla struttura locale "#error" dell'uscita ERROR. L'informazione fornita viene convertita e analizzata mediante l'istruzione di confronto "Uguale". Come primo valore di confronto, all'istruzione viene assegnata l'informazione relativa al tipo di errore. Come secondo valore di confronto viene indicato nell'operando "substitute" il valore "1". Se si tratta di un errore di lettura, la condizione dell'istruzione di confronto viene soddisfatta. e le uscite "#out" e "OK" vengono resettate.

Controllo del programma, Get_Error_ID

esperienza 76
Severi Michele
Brighi Nicolas
5AET 2013
(michele.severi@gmail.com)
(nicolas.brighi2@gmail.com)

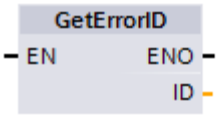


Fig.262 :Blocco Get_Error_ID

L'istruzione "Interroga ID di errore localmente" consente di sapere se si sono verificati errori in un blocco. Se il sistema segnala la presenza di errori durante l'elaborazione del blocco, l'ID del primo errore viene memorizzato nella variabile dell'uscita ID. Nell'uscita ID si possono indicare solo operandi con tipo di dati WORD. Dopo che è stato eliminato il primo errore l'istruzione fornisce l'ID dell'errore successivo.

L'uscita dell'istruzione "Interroga ID di errore localmente" viene impostata solo se l'ingresso dell'istruzione ha lo stato di segnale "1" e se è presente un'informazione di errore. Se una di queste condizioni non viene soddisfatta l'istruzione "Interroga ID di errore localmente" non influisce in alcun modo sulla successiva elaborazione del programma.

L'istruzione "Interroga ID di errore localmente" può essere utilizzata anche per inoltrare al blocco richiamante un messaggio sullo stato di errore. In questo caso la si dovrà inserire nell'ultimo segmento del blocco richiamato.

Nota

L'istruzione "Interroga ID di errore localmente" attiva la gestione locale degli errori all'interno del blocco. Se nel codice di programma di un blocco è stata inserita l'istruzione "Interroga ID di errore localmente", le reazioni del sistema agli errori impostate in precedenza vengono ignorate.

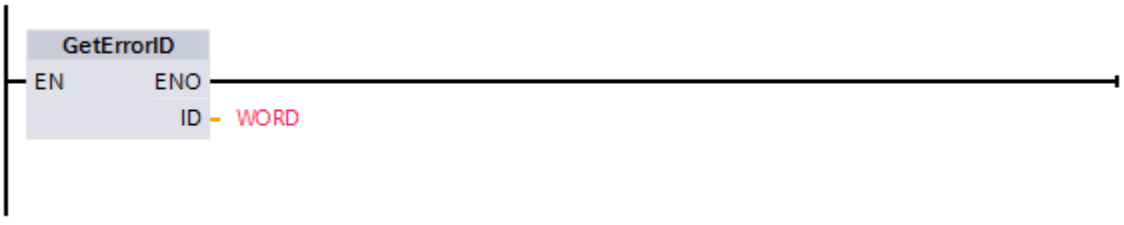


Fig:263 :Applicazione blocco Get_Error_ID

Allarmi, Attach

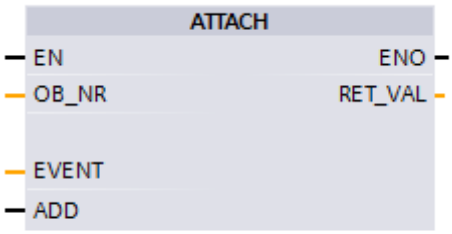


Fig.264 :Blocco ATTACH

esperienza 77
Severi Michele
Brighi Nicolas
5AET 2013
(michele.severi@gmail.com)
(nicolas.brighi2@gmail.com)

Questa istruzione consente di assegnare un blocco organizzativo (OB) a un evento.

Nel parametro OB_NR si indica la denominazione simbolica o numerica del blocco organizzativo a cui viene assegnato l'evento indicato nel parametro EVENT.

Se l'evento indicato nel parametro EVENT si verifica dopo che l'istruzione "ATTACH" è stata eseguita senza errori, il blocco organizzativo indicato nel parametro OB_NR viene richiamato e ne viene eseguito il programma.

Il parametro ADD consente di stabilire se le assegnazioni del blocco organizzativo ad altri eventi effettuate finora devono essere eliminate o mantenute. Se ADD ha valore "0" le assegnazioni disponibili vengono sostituite con quella attuale.

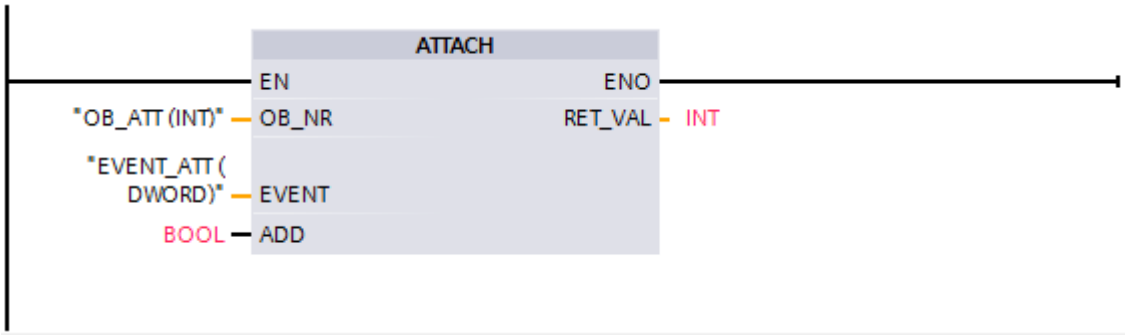
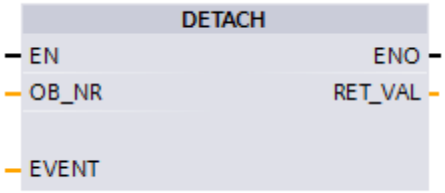


Fig.265 :Applicazione blocco Attach

Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

Allarmi, Detach



esperienza 78
Severi Michele
Brighi Nicolas
5AET 2013
(michele.severi@gmail.com)
(nicolas.brighi2@gmail.com)

Fig.266 :Blocco DETACH

Questa istruzione consente di annullare durante l'esecuzione l'assegnazione attuale di un blocco organizzativo a uno o più eventi.

Se è stato selezionato un singolo evento, l'assegnazione dell'OB allo stesso viene annullata. Tutte le altre assegnazioni momentaneamente esistenti vengono mantenute. È possibile selezionare un singolo evento dalla casella di riepilogo del segnaposto dell'operando nel parametro EVENT.

Se non è stato selezionato alcun evento, tutte le assegnazioni del blocco organizzativo agli eventi momentaneamente esistenti vengono annullate.

Nel parametro OB_NR si indica la denominazione simbolica o numerica del blocco organizzativo di cui viene annullata l'assegnazione all'evento indicato nel parametro EVENT.

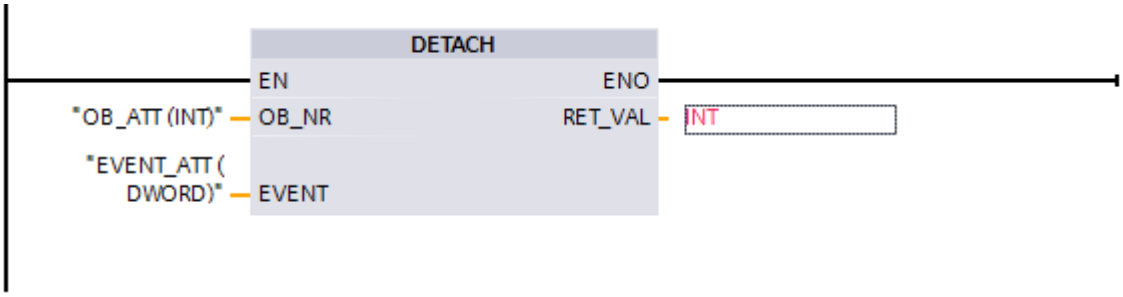


Fig.267 :Applicazione blocco Detach

PID Control, Pid_Compact

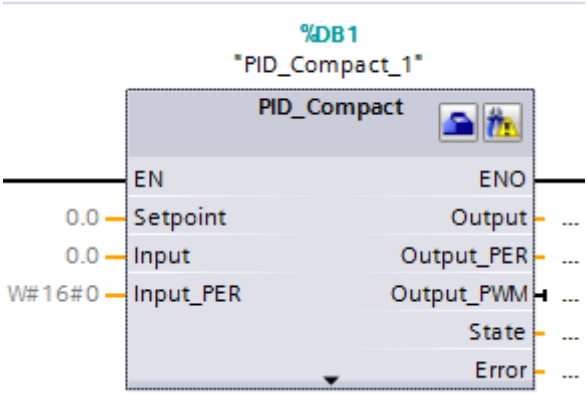


Fig.268 :Blocco PID_COMPACT

esperienza 79
Severi Michele
Brighi Nicolas
5AET 2013
(michele.severi@gmail.com)
(nicolas.brighi2@gmail.com)

L'istruzione PID_Compact mette a disposizione un regolatore PID con ottimizzazione integrata per il funzionamento automatico e manuale.

L'istruzione PID_Compact viene richiamata ad intervalli costanti entro il tempo di ciclo dell'OB richiamante (di preferenza in un OB di schedulazione orologio).

Quando si avvia la CPU, PID_Compact viene avviato nell'ultimo modo di funzionamento attivo. Per lasciare PID_ Compact nel modo di funzionamento "Inattivo" impostare sb_RunModeByStartup = FALSE.

Idealmente il tempo di campionamento dovrebbe corrispondere al tempo di ciclo dell'OB richiamante. L'istruzione PID_Compact misura l'intervallo tra due richiami il quale corrisponde al tempo di campionamento attuale. Ad ogni commutazione del modo di funzionamento e al primo avvio viene generato il valore medio dei primi 10 tempi di campionamento. Se il tempo di campionamento attuale si discosta troppo dal valore medio si verifica un errore (Error = 0800 hex) e PID_Compact passa al modo di funzionamento "Inattivo".

Poiché il sistema regolato richiede un certo tempo per reagire a una modifica del valore di uscita, è utile non calcolare questo valore in ogni ciclo. Il tempo di campionamento dell'algoritmo PID è il tempo che trascorre tra due calcoli del valore di uscita. Viene determinato durante l'ottimizzazione e arrotondato ad un multiplo del tempo di ciclo. Tutte le altre funzioni di PID_Compact vengono eseguite ad ogni richiamo.

Quando si verificano degli errori, vengono emessi nel parametro Error e PID_Compact passa al modo di funzionamento "Inattivo". Il parametro Reset resetta nuovamente gli errori.

CTRL_PWM

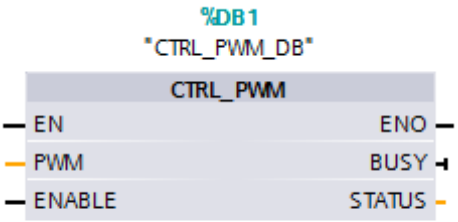


Fig.269 :Blocco CTRL_PWM

esperienza 80
Severi Michele
Brighi Nicolas
5AET 2013
(michele.severi@gmail.com)
(nicolas.brighi2@gmail.com)

L'istruzione "CTRL_PWM" consente di attivare e disattivare tramite il software un generatore di impulsi supportato dalla CPU.

Nota

Un generatore di impulsi si parametrizza esclusivamente nella Configurazione dispositivi e non tramite l'istruzione "CTRL_PWM". Una parametrizzazione che deve essere effettiva nella CPU è di conseguenza possibile soltanto nello stato di funzionamento STOP della stessa.

L'identificativo HW del generatore di impulsi che si intende controllare con questa istruzione deve essere specificato sull'ingresso PWM. Una premessa necessaria per l'esecuzione corretta dell'istruzione è che il generatore di impulsi indicato sia stato abilitato nella configurazione hardware.

Nell'ingresso PWM si possono indicare solo variabili con tipo di dati HW_PWM. La lunghezza del tipo di dati hardware HW_PWM è di una WORD.

Il generatore di impulsi viene attivato quando è impostato il bit dell'ingresso ENABLE dell'istruzione. Se ENABLE presenta il valore TRUE, il generatore crea impulsi caratterizzati dalle proprietà definite nella Configurazione dispositivi. Quando il bit nell'ingresso ENABLE viene resettato o se la CPU commuta in STOP, il generatore di impulsi viene disattivato e non vengono più generati impulsi.

L'istruzione "CTRL_PWM " viene eseguita solo se lo stato di segnale nell'ingresso EN è "1".

Poiché l'S7-1200 attiva il generatore di impulsi quando viene eseguita l'istruzione CTRL_PWM, in questo sistema il parametro BUSY ha sempre il valore FALSE.

L'uscita di abilitazione ENO viene impostata solo se il segnale nell'ingresso di abilitazione EN è "1" e se non si sono verificati errori durante l'esecuzione dell'istruzione.

Nota

Utilizzo della tabella di forzamento con PAM e PTO

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Il forzamento degli ingressi e delle uscite digitali utilizzati per PAM e PTO non è possibile. Gli ingressi e le uscite digitali che sono stati assegnati con la configurazione dispositivi possono essere comandati solo mediante la tabella di forzamento o la tabella di controllo.

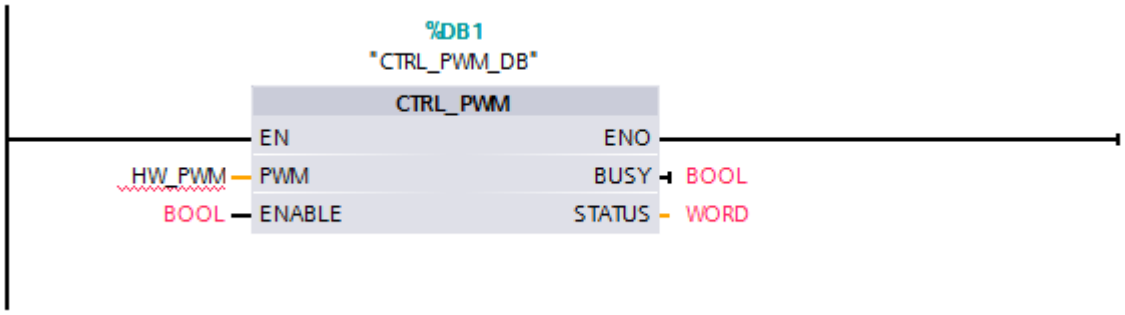


Fig. 270 :Applicazione blocco CTRL_PWM

Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

Web Server

esperienza 69

Carloni Lorenzo
Morigi Marco
5AET 2013
(lorenzo.carloni1@gmail.com)
(marco.morigi26@gmail.com)

Il PLC mette a disposizione un **WEB SERVER** il quale è accessibile tramite l'indirizzo IP del PLC se opportunamente configurato.
Per verificare il funzionamento, inseriamo questo semplice programma nel PLC che corrisponde ad un set-reset.

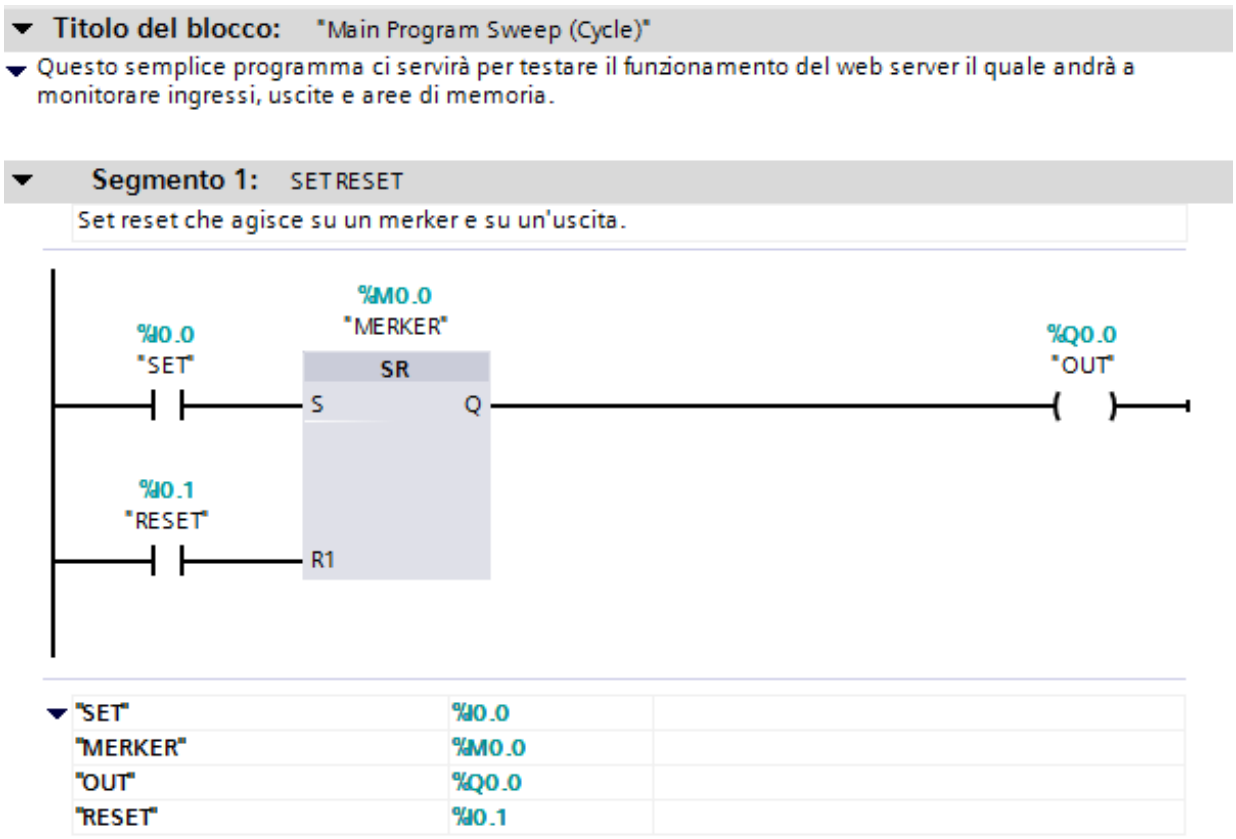


Fig.271 :Monitoraggio ingressi,uscite,merker tramite Web Server

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Ora attiviamo il WEB SERVER cliccando su configurazione dispositivo e selezionando la tendina "Proprietà" tra quelle che compariranno.
Dall'elenco sulla sinistra selezioniamo "Server Web".
Al suo interno mettiamo il tick sull'opzione "Abilita Web Server su quest'unità".

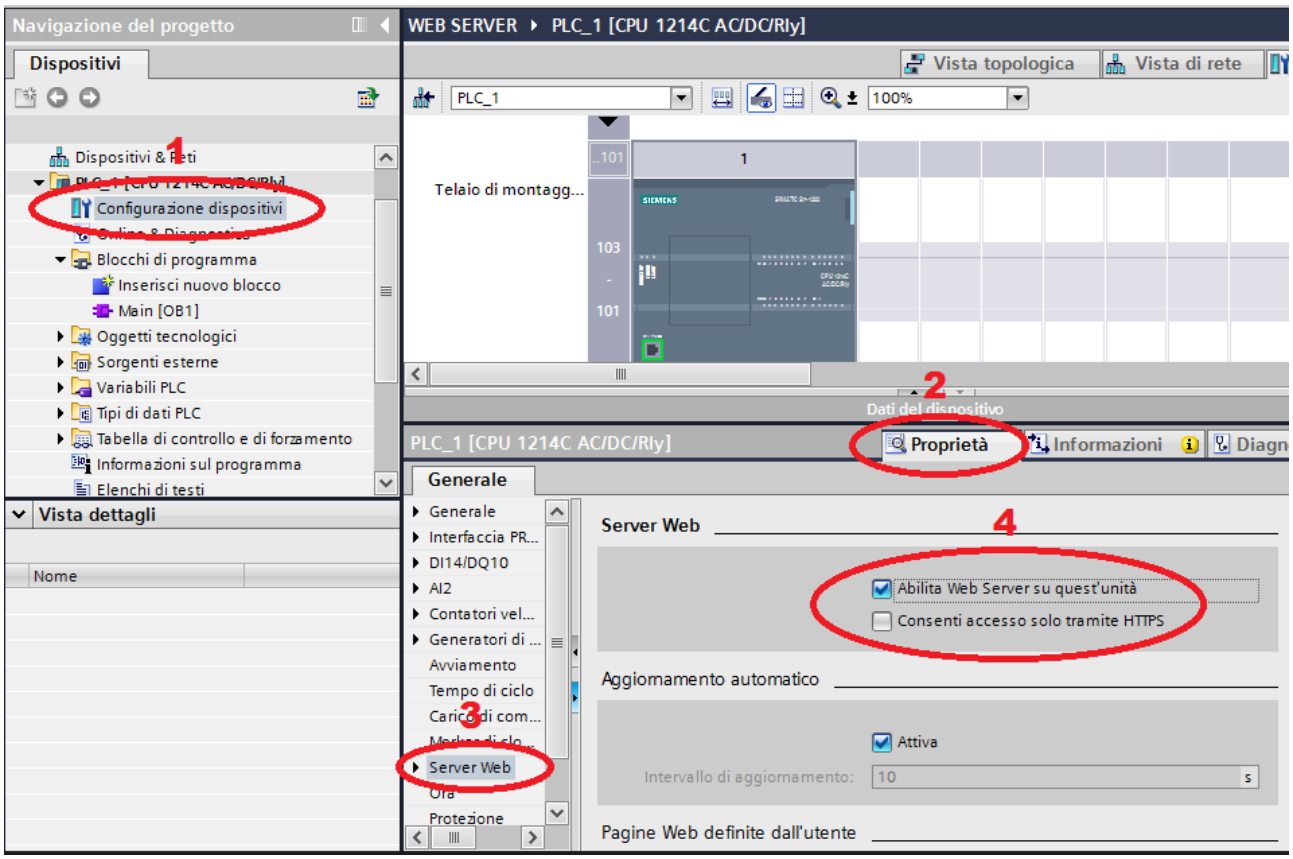


Fig.272 :Configurazione del Web Server(come si attiva)

Sempre all'interno di "Proprietà" selezioniamo "Protezione".
Qui possiamo scegliere come regolare l'accesso al PLC: possiamo lasciarlo senza protezione, mettere la protezione solo in scrittura o sia in scrittura che in lettura.
Nel caso inseriamo una protezione dobbiamo definire una password.

N.B. La password è Caps Sensitive, occhio alle maiuscole!
La password non è formattabile, attenzione a non perderla!

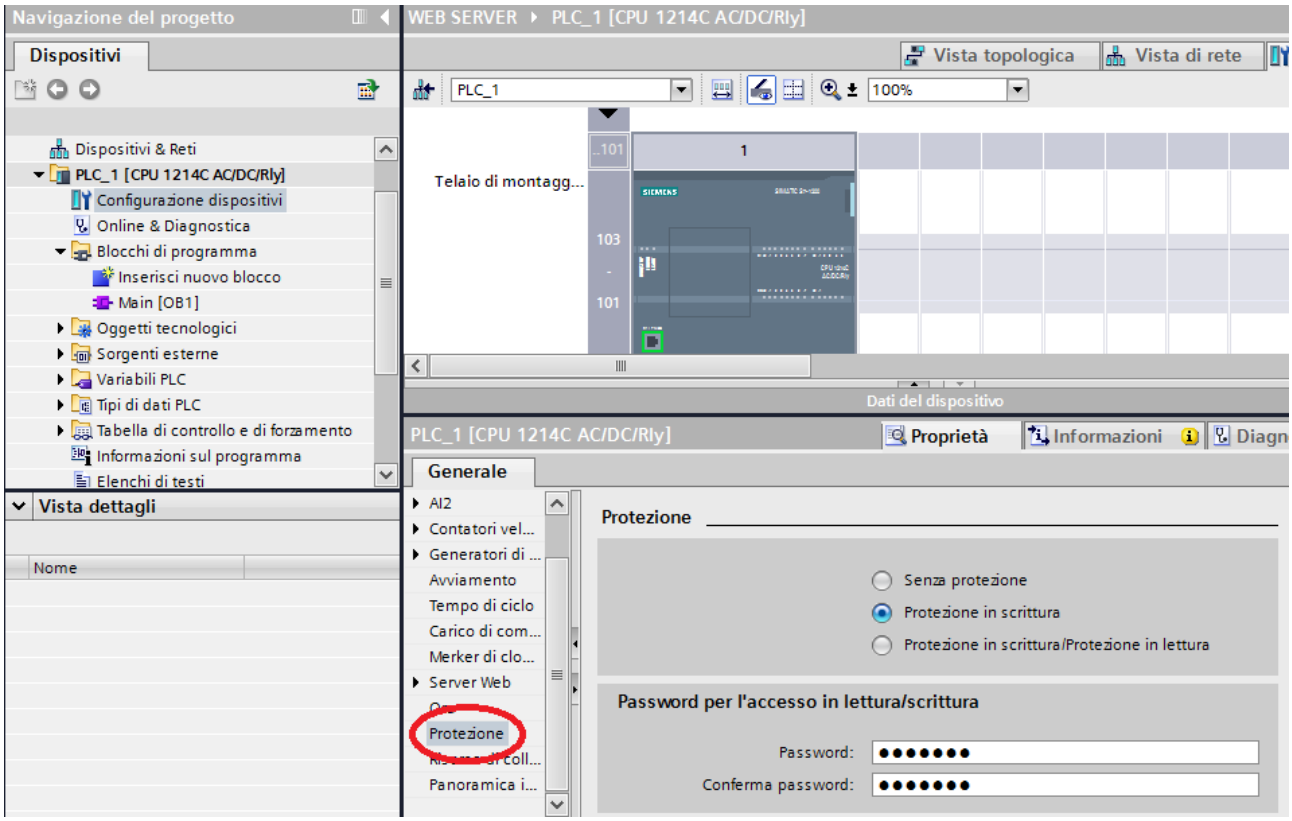


Fig.273 :Protezione,Pasword per accedere alla lettura/scrittura

Per sapere l'indirizzo del nostro PLC bisogna andare in "Online & Diagnostica" e selezionare "Interfaccia PROFINET".
Qui troveremo il nostro indirizzo IP, tramite il quale accederemo all'HTML del PLC.

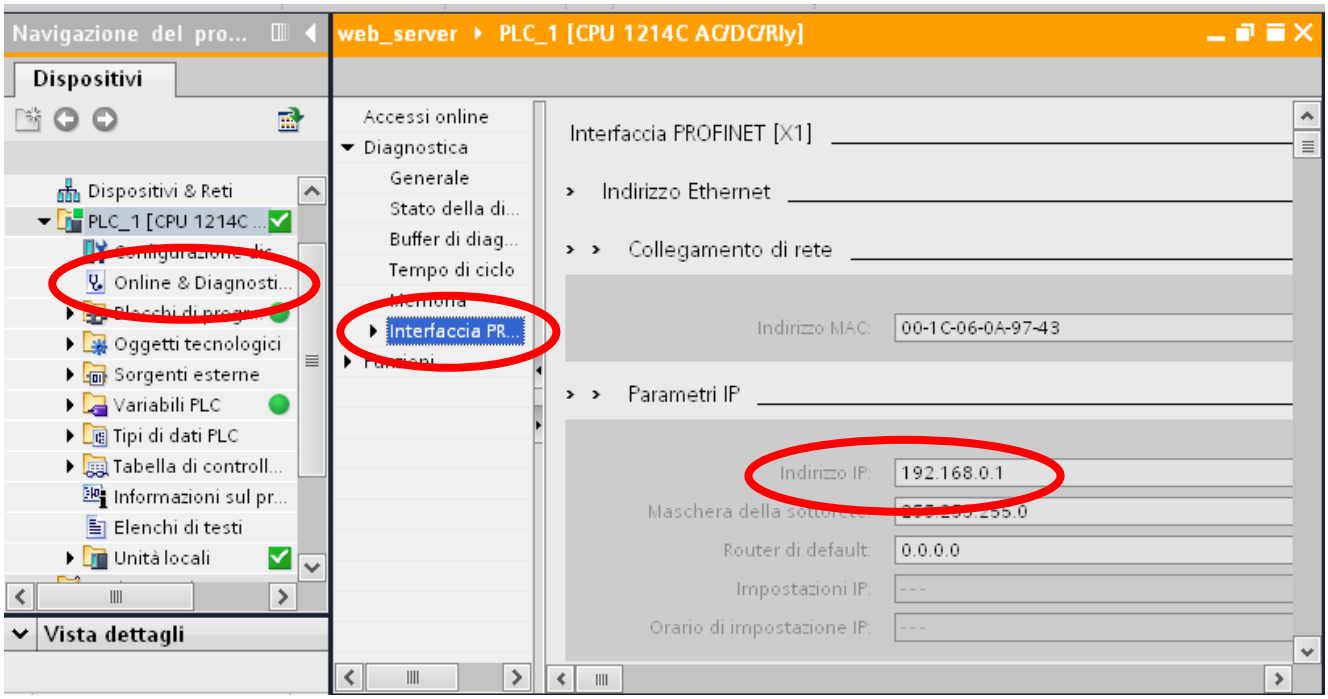


Fig.274 :Interfaccia PROFINET e indirizzo IP del PLC

Ora apriamo il nostro browser e inseriamo l'IP nella barra dell'indirizzo WEB.

Fig.275 :Browser PLC

Ora dobbiamo installare i certificati per poter utilizzare l'autenticazione tramite Password. In alto a destra troviamo "download certificate". Cliccandoci sopra ci fa scaricare un file, il nostro certificato. Lo salviamo in una directory facilmente raggiungibile.
Cliccandoci sopra due volte si installa automaticamente il certificato.

Se il browser blocca ancora la connessione perché non affidabile, bisogna seguire questa procedura:

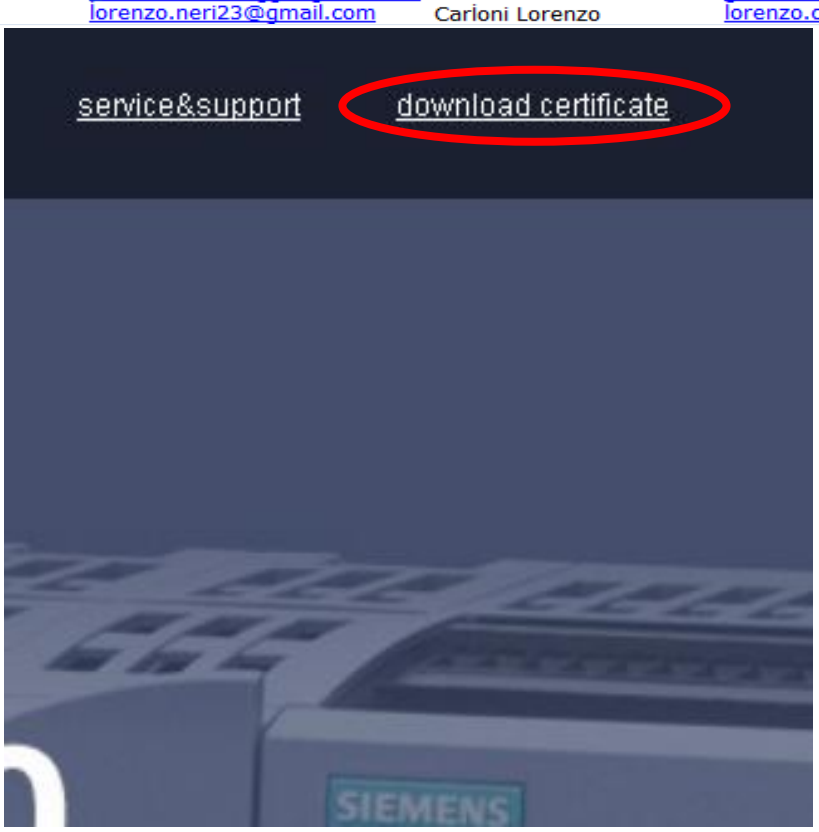


Fig.276 :Procedura in caso di blocco connessione

PER MOZILLA FIREFOX (v. 20.0.1):

Andiamo in "Strumenti" → "Opzioni" → "Avanzate" → "Encriptyon".
Qui apriamo "Visualizza certificati".

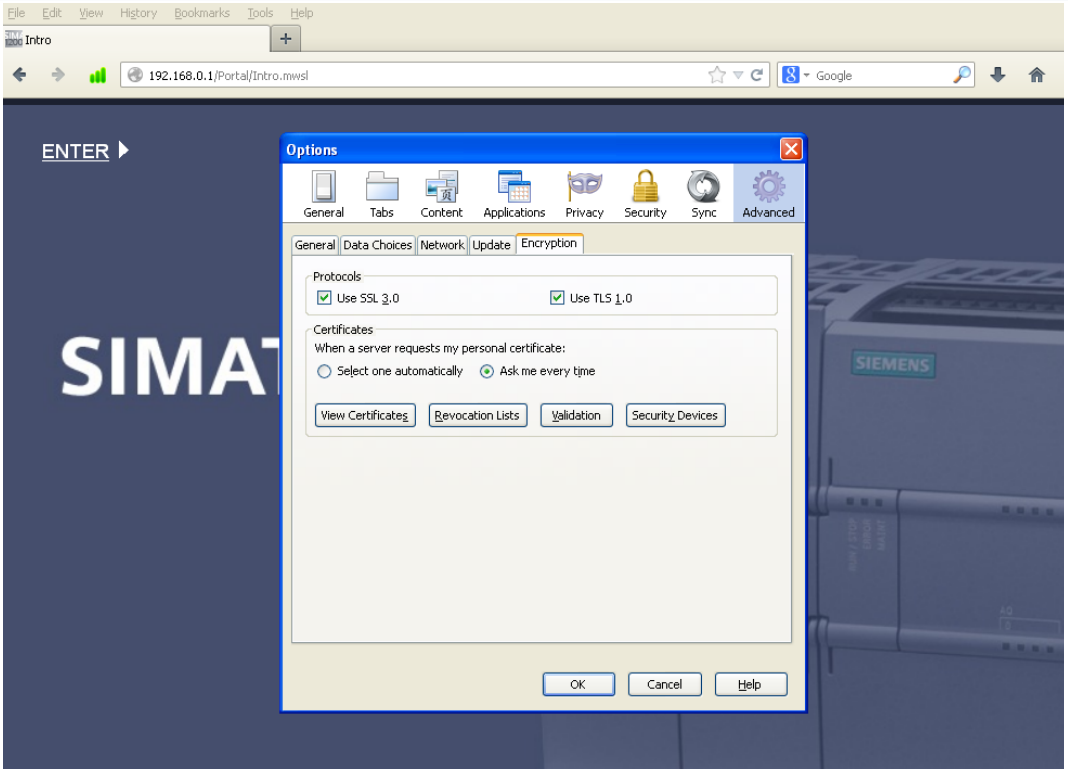


Fig.277 :Procedure per Mozzilla Firefox

Clicchiamo "Importa", selezioniamo il nostro File e lo installiamo.

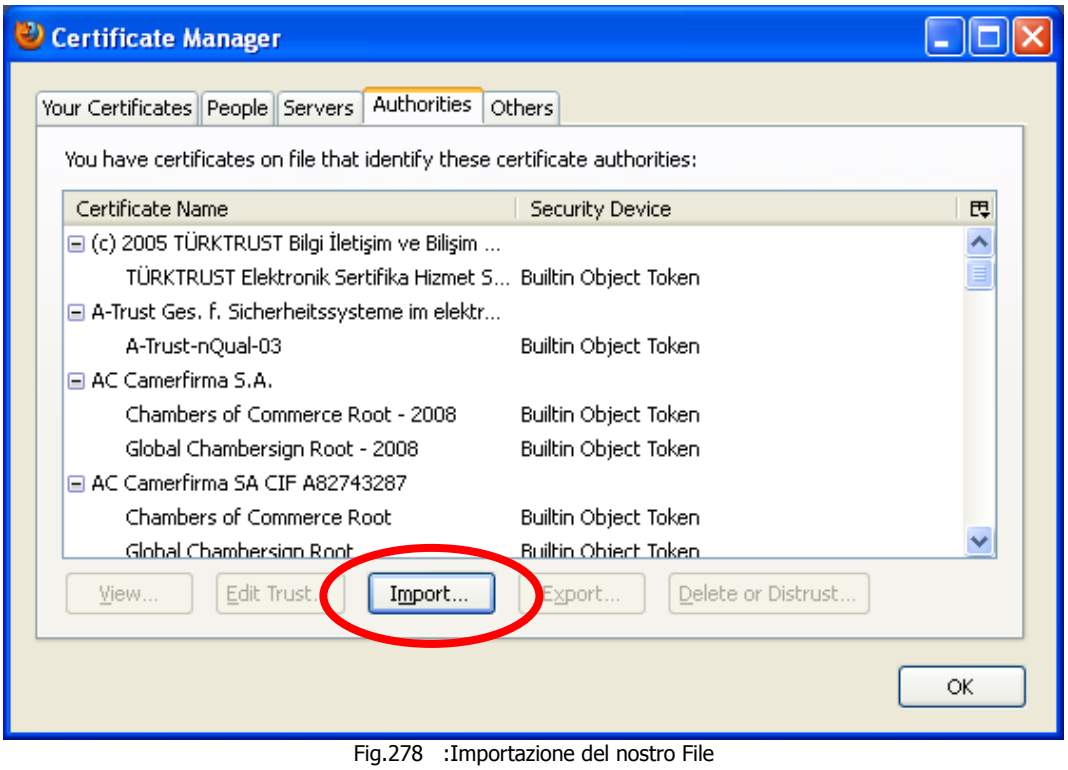


Fig.278 :Importazione del nostro File

PER INTERNET EXPLORER (v. 8):

Andiamo in "Strumenti" → "Opzioni Internet" → "Contenuto".

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

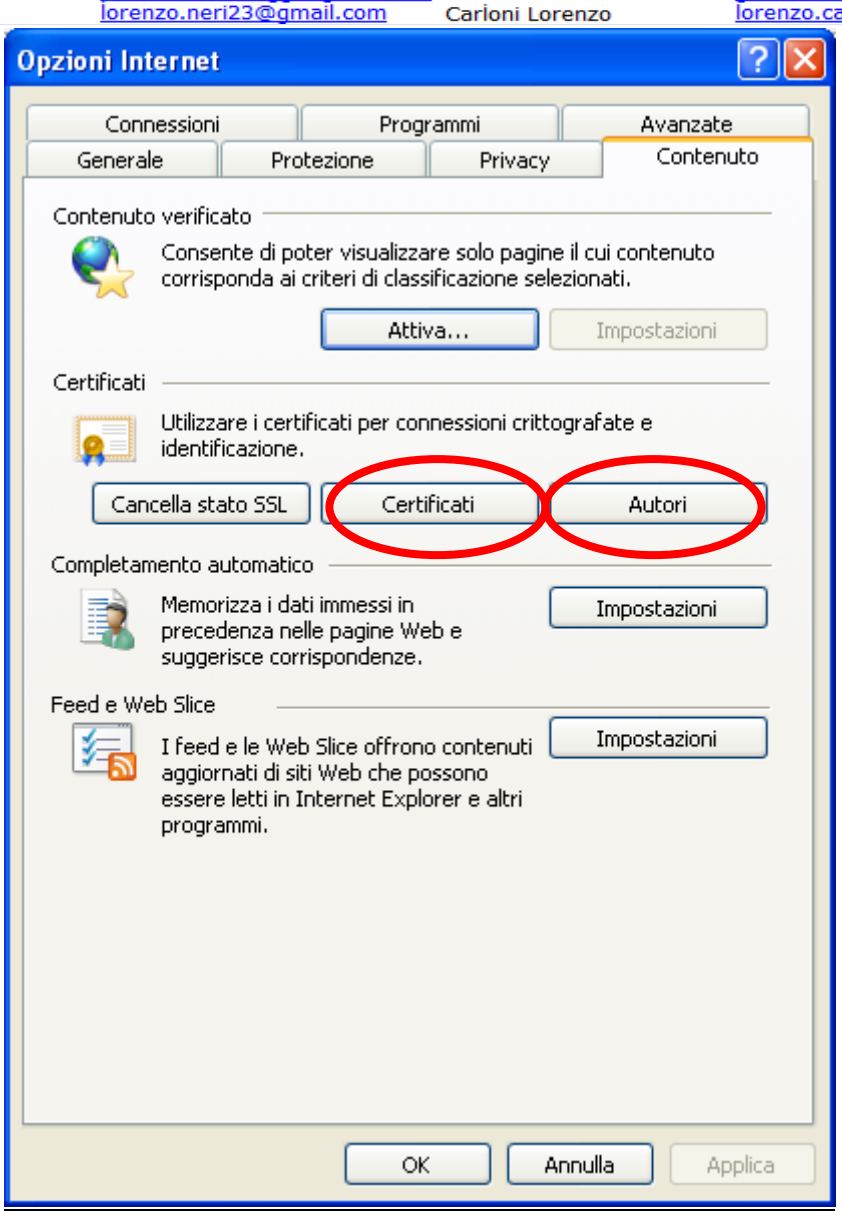


Fig.279 :Procedure per Internet Explorere

Qui apriamo "Certificati" e clicchiamo su "Importa".
Partirà una procedura guidata che ci chiederà di selezionare il percorso del nostro certificato.
Alla fine della procedura selezioniamo "Autori" nella schermata riportata qui sopra e ripetiamo lo stesso procedimento.

PER GOOGLE CHROME (v. 26):

Apriamo il menù a tendina sulla destra e selezioniamo "Impostazioni".
In fondo visualizziamo le opzioni avanzate.
Clicchiamo su "Gestisci certificati.." sotto "HTTPS/SSL".

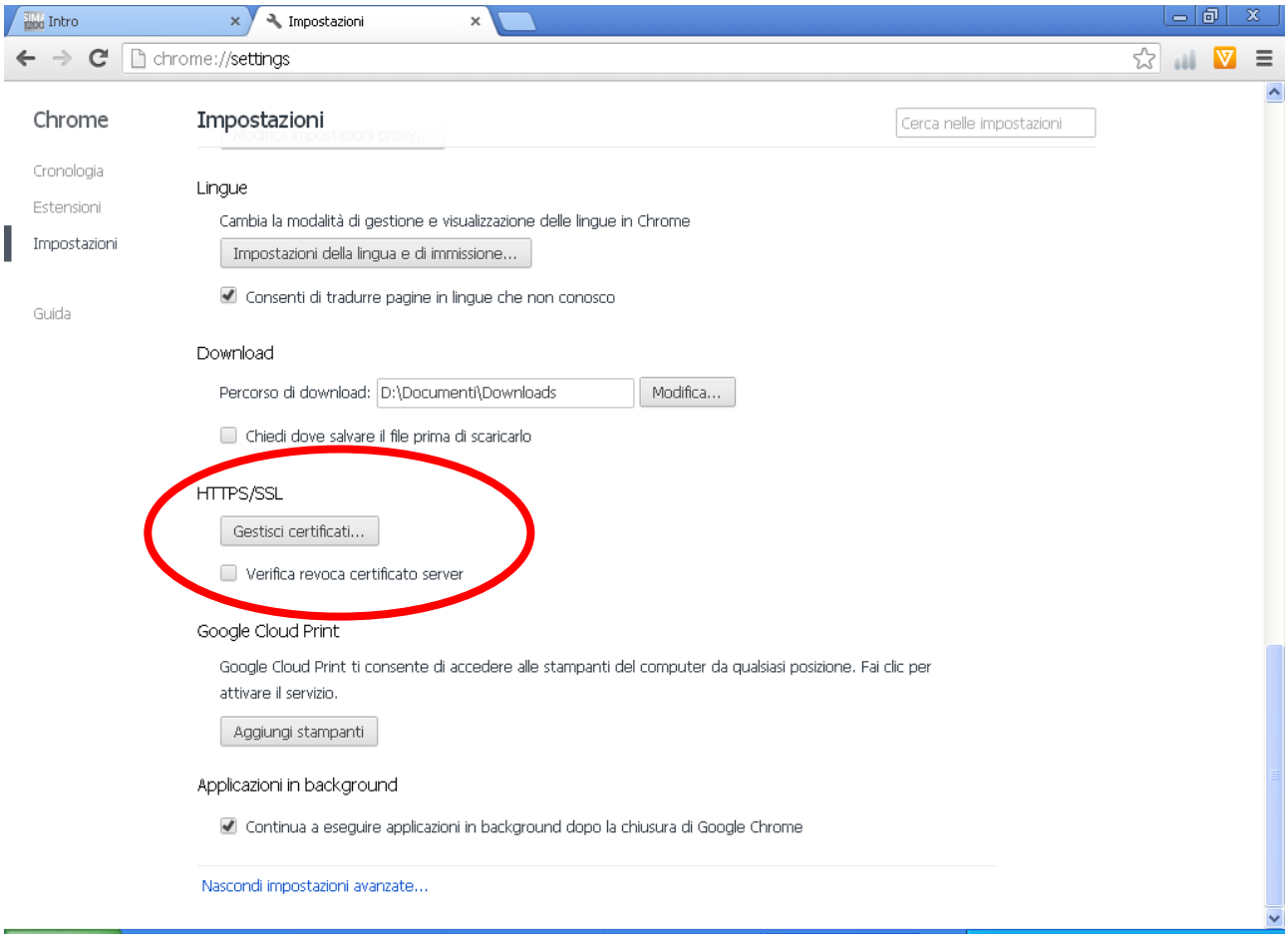


Fig.280 :Procedure per Google Chrome

Nella finestra che si apre clicchiamo su "Importa" e si aprirà una procedura guidata che ci chiederà il percorso del nostro certificato.

PER OPERA (v. 12.15):

Premiamo Ctrl+F12 per aprire la finestra "Preferenze", qui andiamo in "Sicurezza" e poi "Gestione dei certificati".
Qui selezioniamo la tendina "Autorità" e poi clicchiamo su "Importa".
Cerchiamo il nostro certificato e clicchiamo su OK.

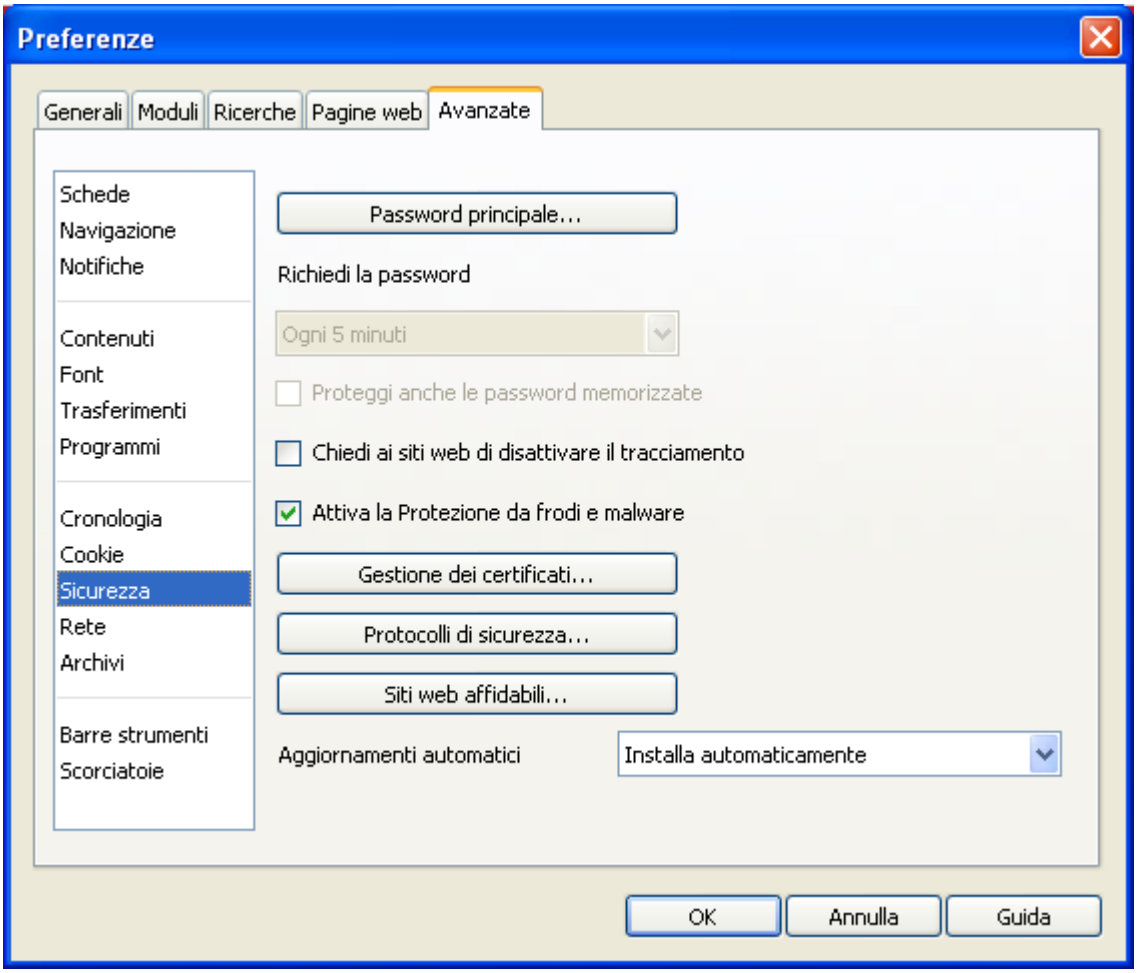


Fig.281 :Procedure per Opera

Ora clicchiamo su "ENTER" per accedere al PLC.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

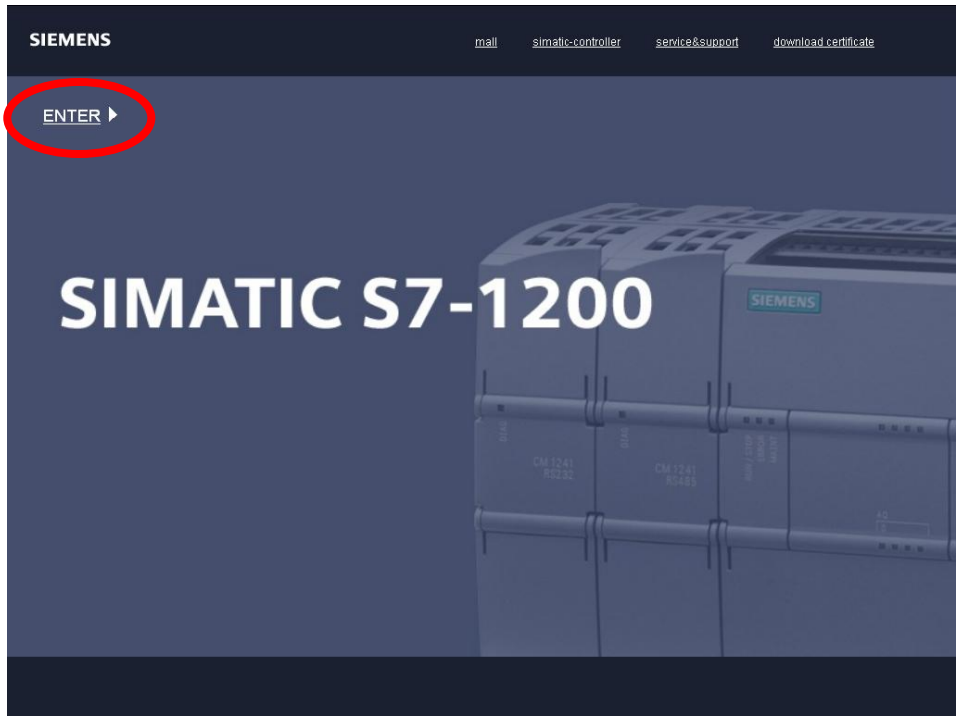


Fig.282 :Accesso al PLC

In alto a destra possiamo accedere, nel caso che abbiamo impostato delle limitazioni tramite password. Il nome utente sarà "admin" e la password quella scelta da noi.

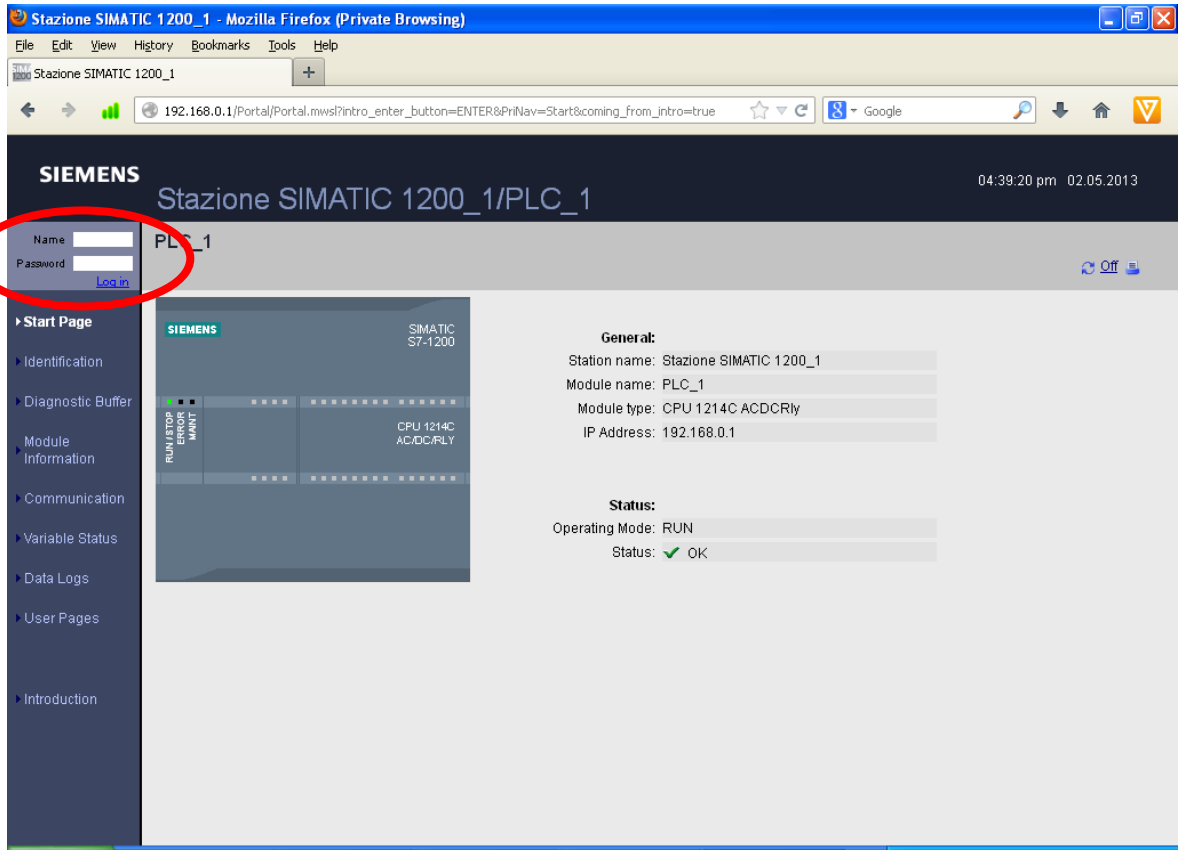


Fig.283 :Nome utente e password

Nella prima schermata "Start Page", possiamo visualizzare le informazioni principali del nostro PLC e lo stato di funzionamento.
Se loggati possiamo far lampeggiare i led e cambiare lo stato del PLC.

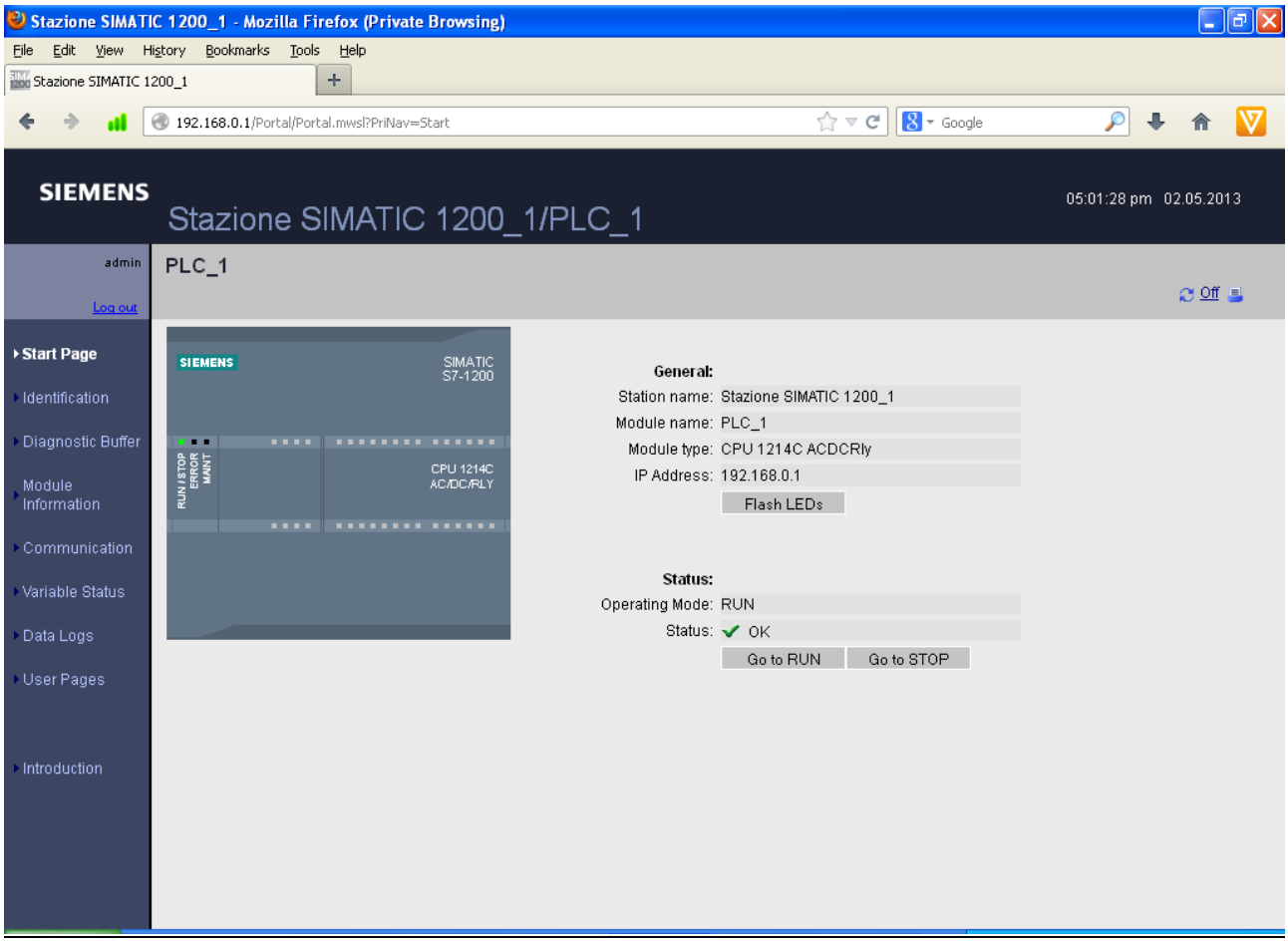


Fig.284 :Informazioni e stato funzionamento PLC

Nella pagina "identification" vengono visualizzati tutti i codici che identificano il nostro PLC: il codice seriale, il codice dell'hardware e del firmware e le loro versioni.

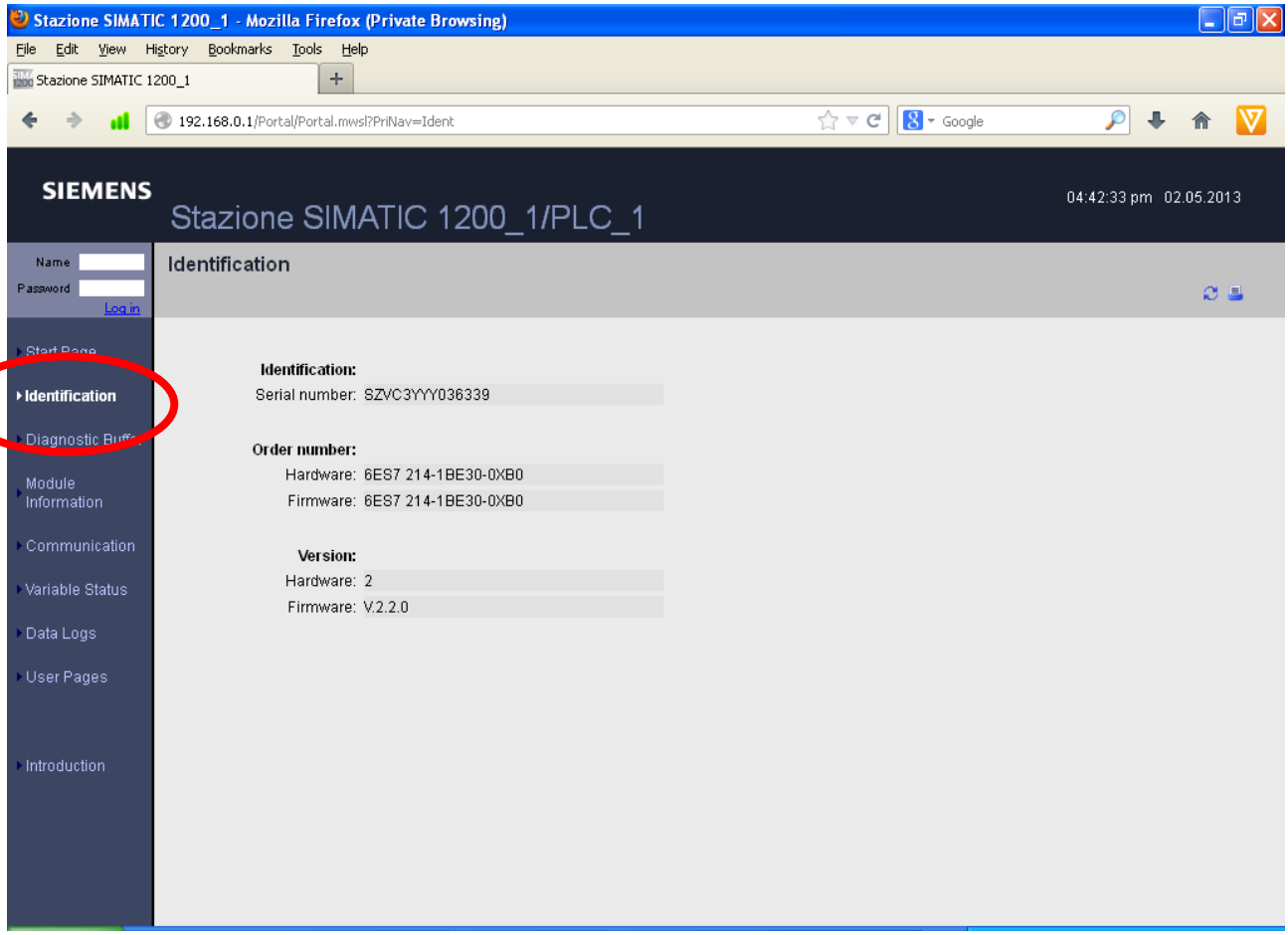


Fig.285 :Identification PLC

In "Diagnostic Buffer" viene visualizzato lo storico del nostro PLC, con data e ora.

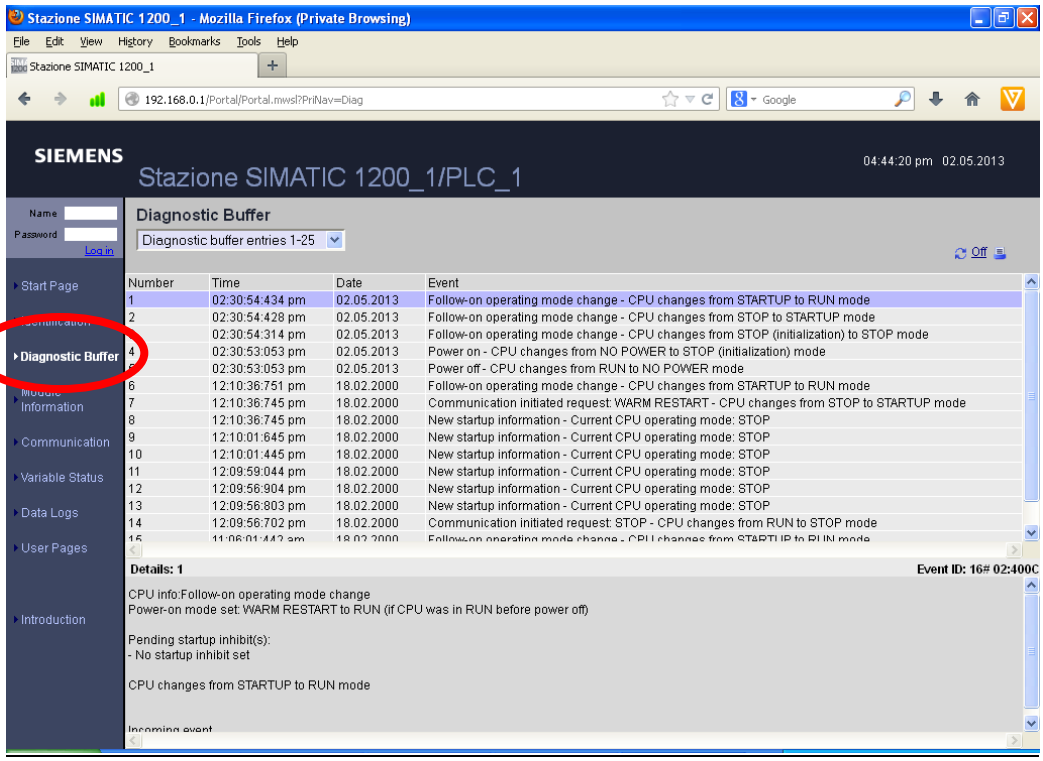


Fig.286 :Diagnostic Buffer del PLC

In "Module Information" viene visualizzato il PLC e tutto ciò che è collegato ad esso e il loro stato.

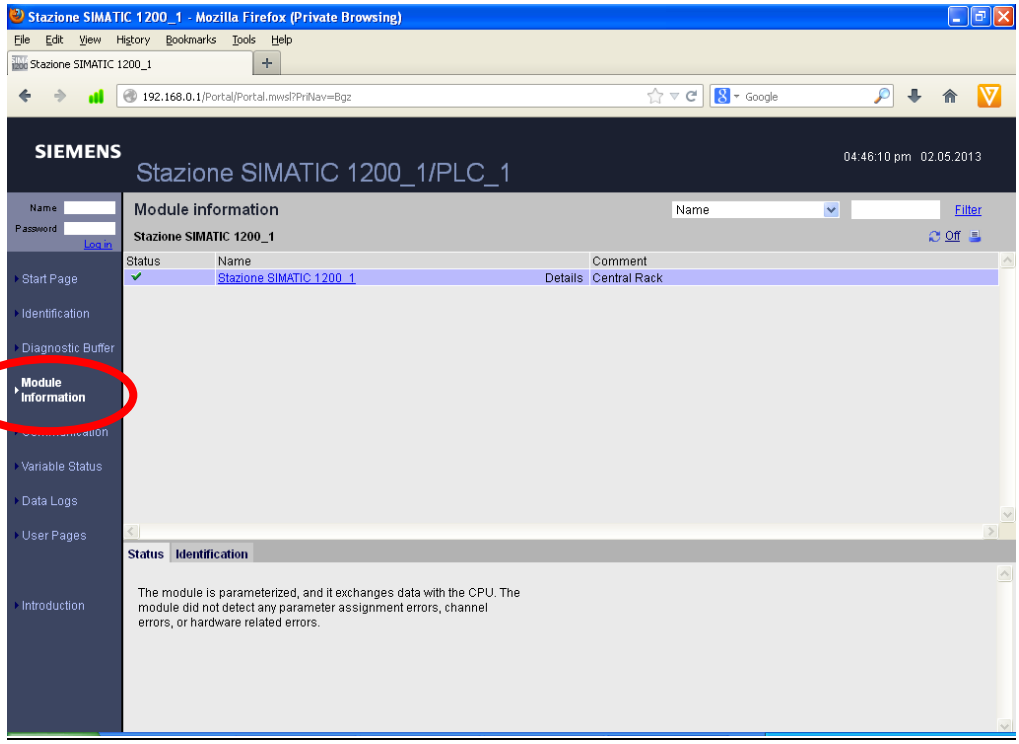


Fig.287 :Module information del PLC

In "Comunication" ottengo informazione sulla connessione alla rete, posso leggere qui il MAC Address, il nome del PLC, i parametri IP e le proprietà fisiche della connessione.
In "Statistics" vengono mostrate le informazioni riguardanti i pacchetti scambiati.

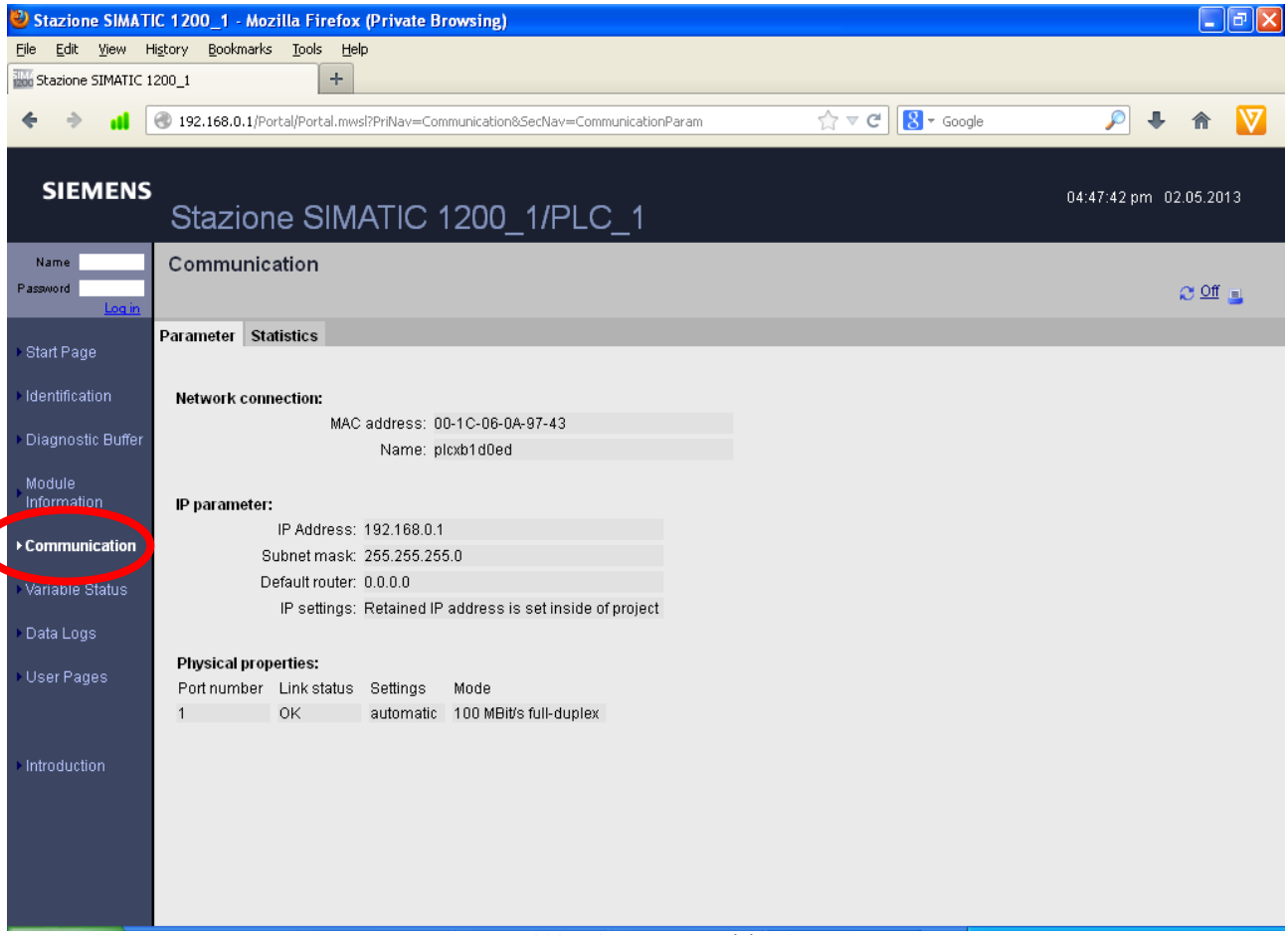


Fig.288 :Comunication del PLC

In "Variable Status" posso scegliere quali variabili visualizzare, le posso dichiarare tramite l'indirizzo (I0.0 e M0.0) o tramite l'identificativo da me assegnato (RESET e OUT).
Dal di qui posso scegliere anche in che formato mostrarle tramite il menù a tendina.
Se loggato posso anche andarle a scrivere e modificarne lo stato.

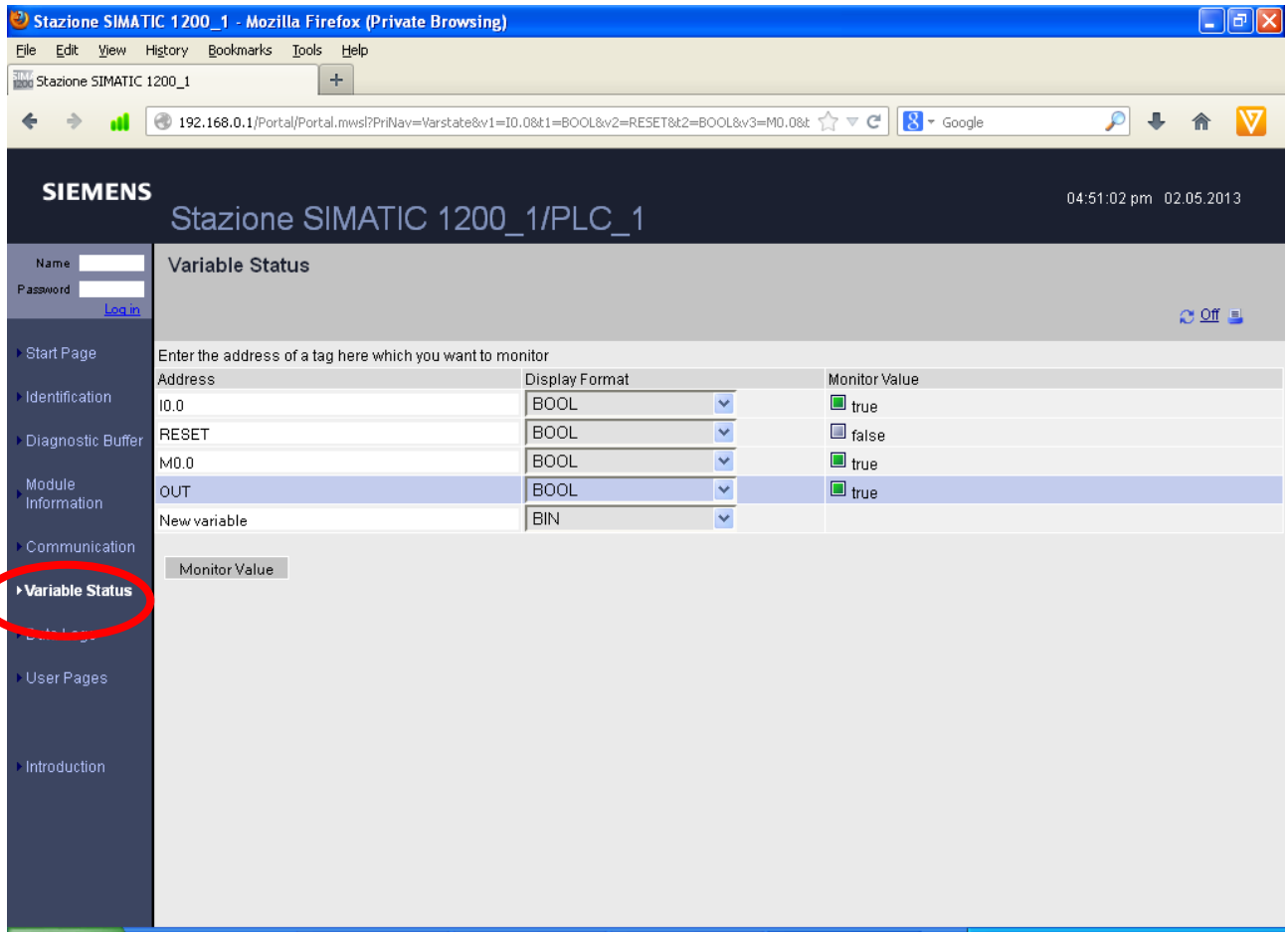


Fig.289 :Variable Status del PLC

In "Data Logs" vengono visualizzate delle informazioni relative al PLC.

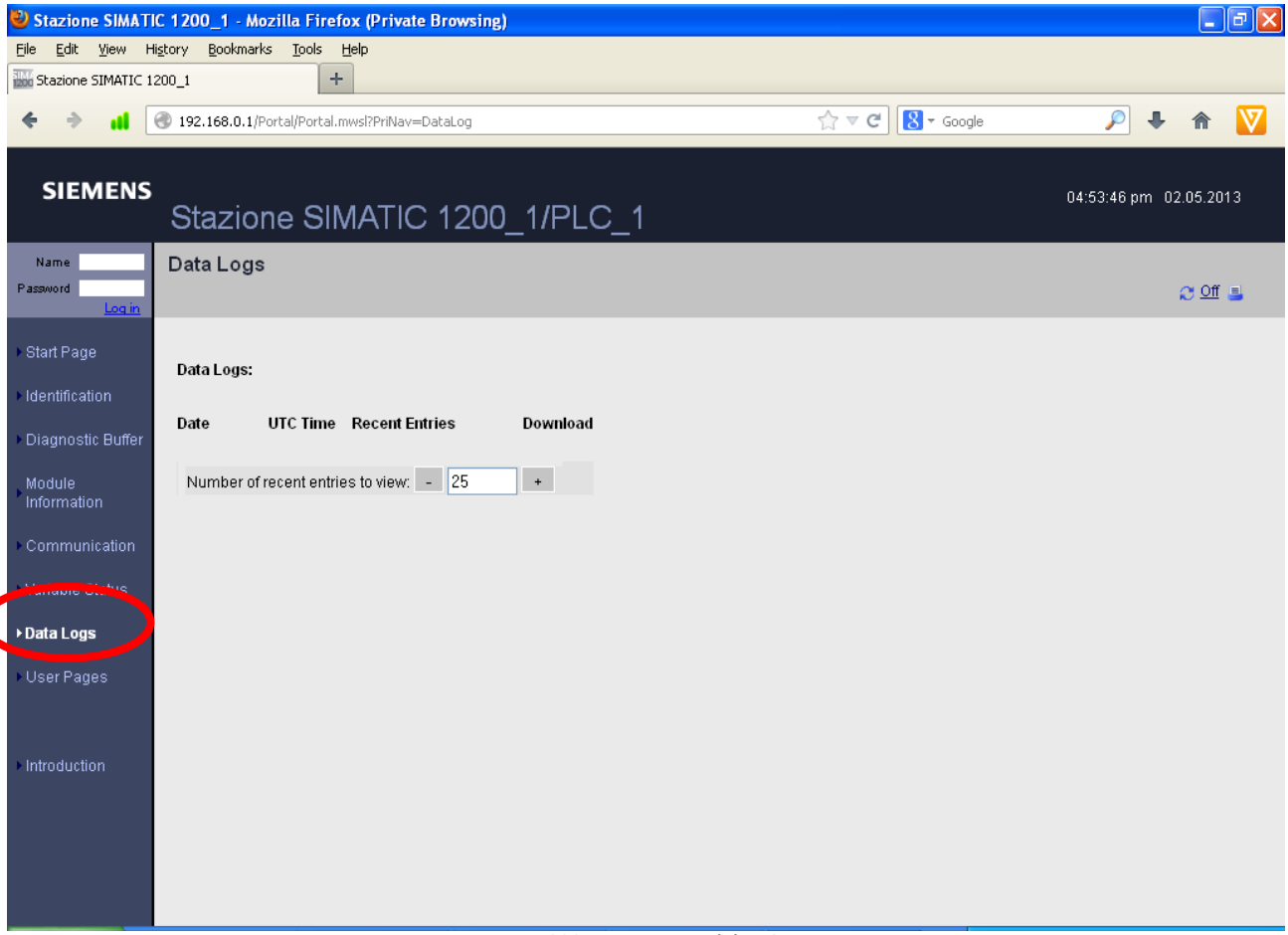


Fig.290 :Data Logs del PLC

In "User Pages" posso accedere alla mia pagina HTML nel caso l'abbia caricata in precedenza nel PLC e con"Introdution" torno alla pagina iniziale.

Gestione orologio HARDWARE

esperienza 70
Marco Morigi
5AET 2013
(marco.morigi26@gmail.com)

INSERIMENTO DATA

Questo blocco a seguito di una attivazione "**I0.1**" consente di inserire la data nel plc facendo riferimento al database attuale.

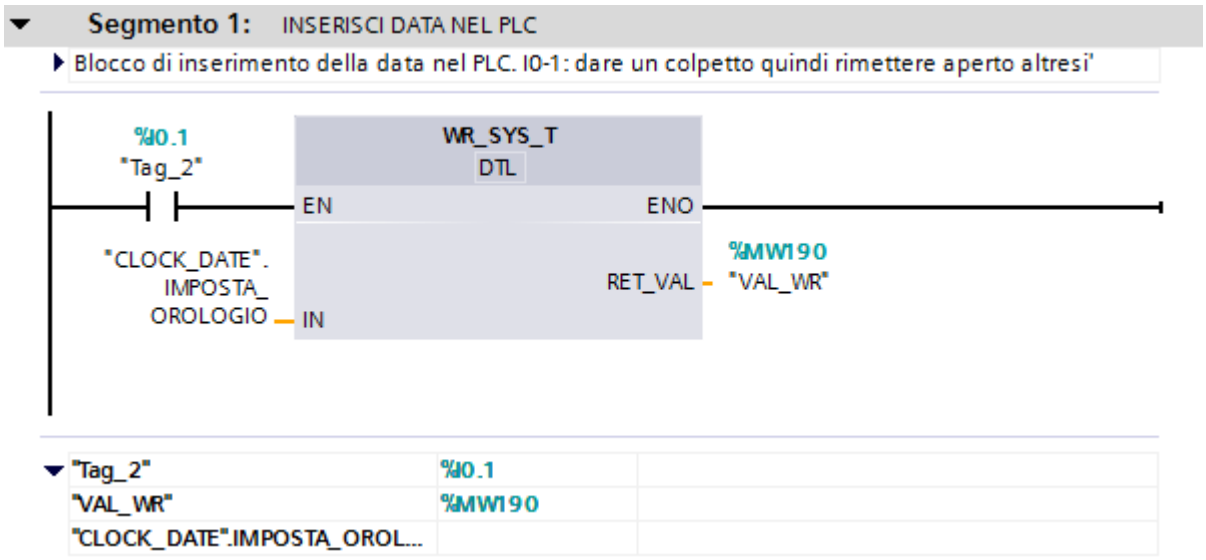


Fig.291 :Gestione Orologio Hardware (parte 1)

Con il seguente blocco si andrà a leggere la temporizzazione dal sistema ogni qualvolta riceve l'impulso dal contatto "**I0.0**".

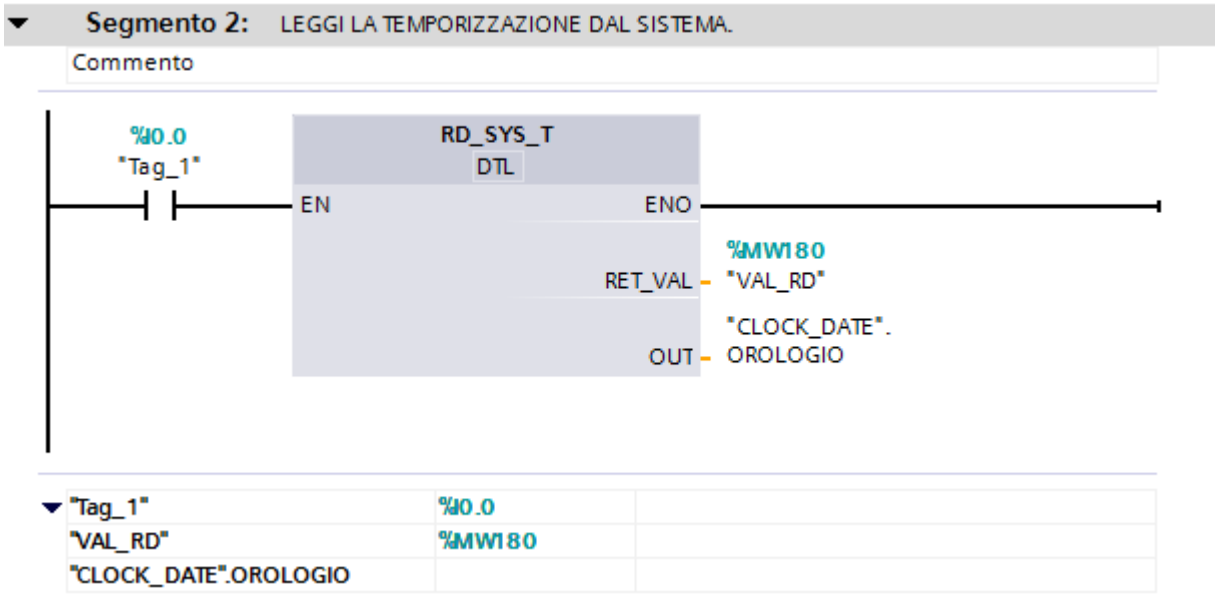


Fig.292 :Gestione Orologio Hardware (parte 2)

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

LEGGI L'ANNO

Quando il blocco riceve l'impulso da "**I0.2**", andiamo a leggere il DB in cui è salvata la temporizzazione e preleviamo l'anno che poi verrà convertito in numero per far dei confronti.

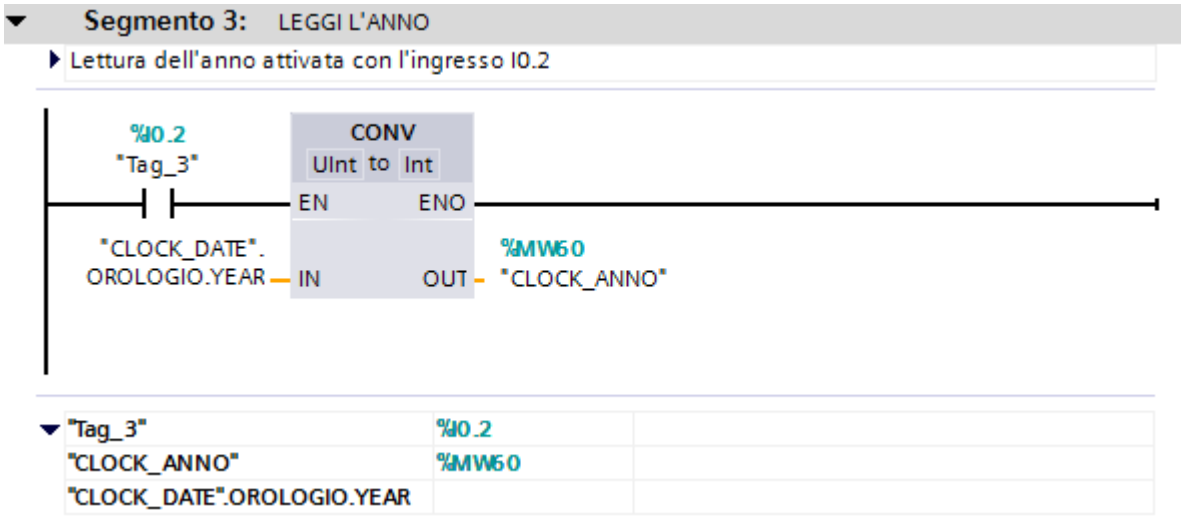


Fig.293 :Gestione Orologio Hardware (parte 3)

LEGGI IL MESE

Quando il blocco riceve l'impulso da "**I0.2**", andiamo a leggere il DB in cui è salvata la temporizzazione e preleviamo il mese che poi verrà convertito in numero per far dei confronti.

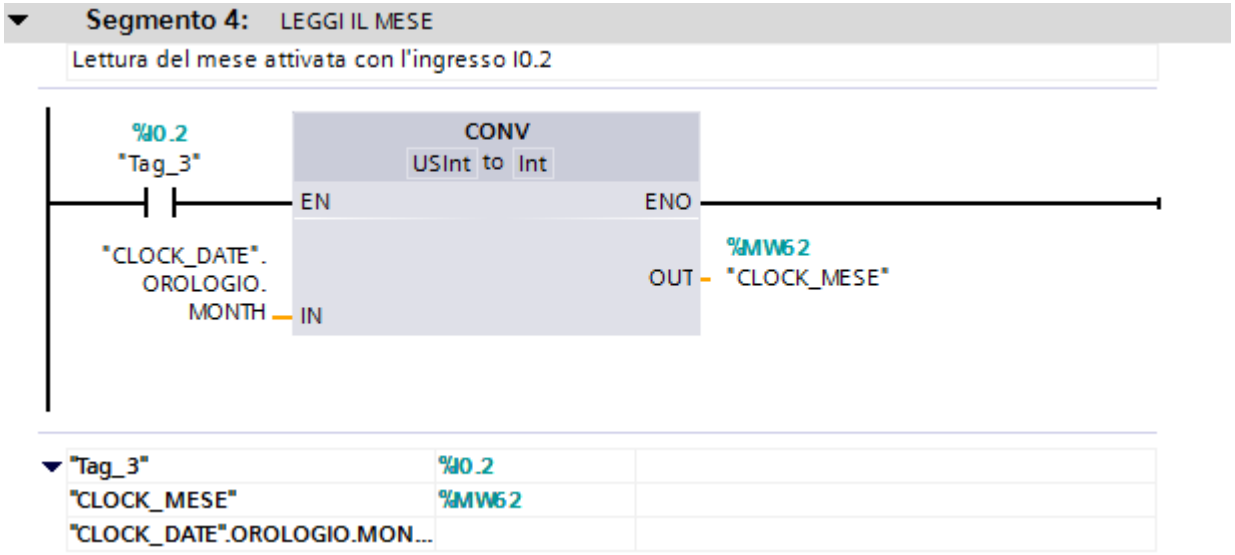


Fig.294 :Gestione Orologio Hardware (parte 4)

LEGGI IL GIORNO

Quando il blocco riceve l'impulso da "**I0.2**", andiamo a leggere il DB in cui è salvata la temporizzazione e preleviamo il giorno che poi verrà convertito in numero per far dei confronti.

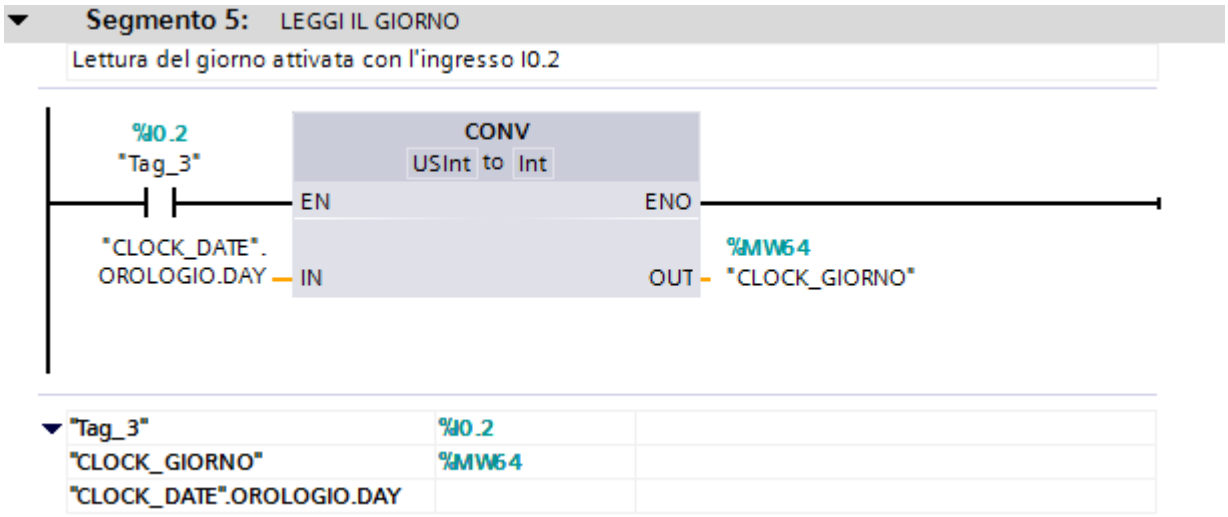


Fig.295 :Gestione Orologio Hardware (parte 5)

LEGGI L'ORA

Quando il blocco riceve l'impulso da "**I0.2**", andiamo a leggere il DB in cui è salvata la temporizzazione e preleviamo l'ora che poi verrà convertito in numero per far dei confronti.

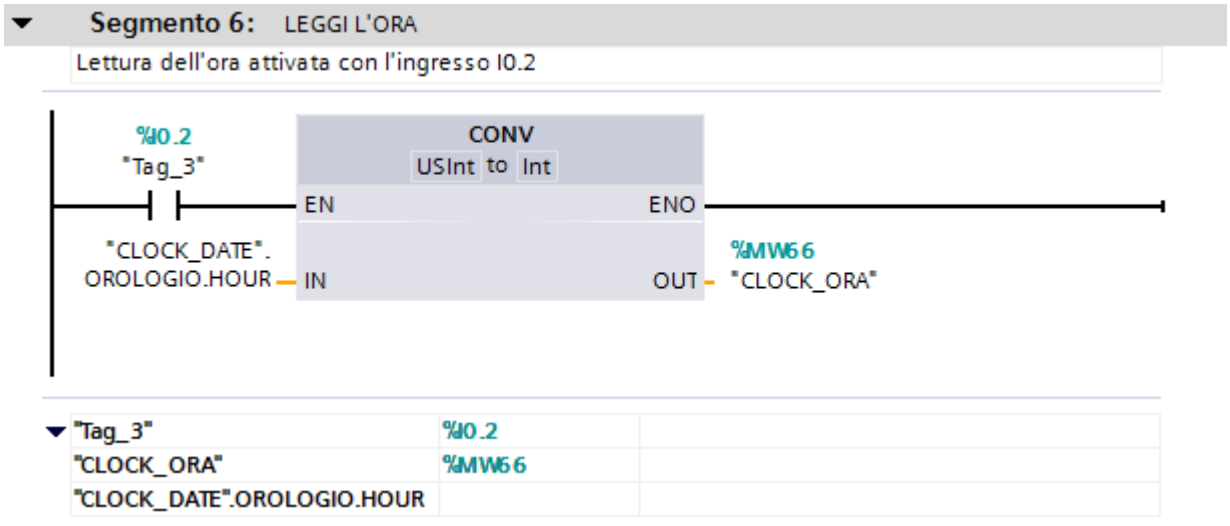


Fig.296 :Gestione Orologio Hardware (parte 6)

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

LEGGI IL MINUTO

Quando il blocco riceve l'impulso da "**I0.2**", andiamo a leggere il DB in cui è salvata la temporizzazione e preleviamo il minuto che poi verrà convertito in numero per far dei confronti.

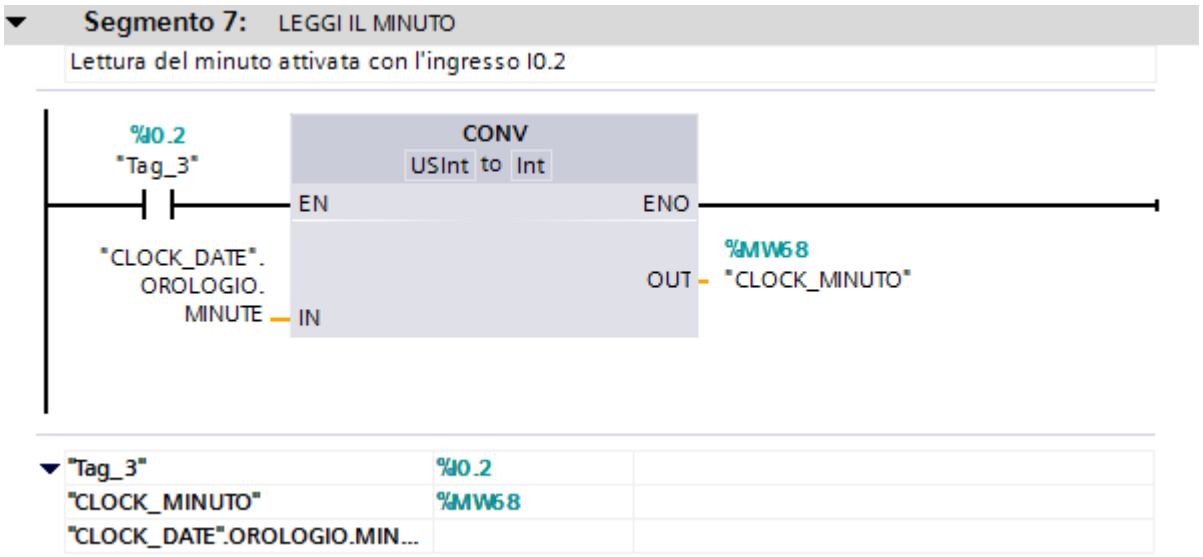


Fig.297 :Gestione Orologio Hardware (parte 7)

LEGGI IL SECONDO

Quando il blocco riceve l'impulso da "**I0.2**", andiamo a leggere il DB in cui è salvata la temporizzazione e preleviamo il secondo che poi verrà convertito in numero per far dei confronti.

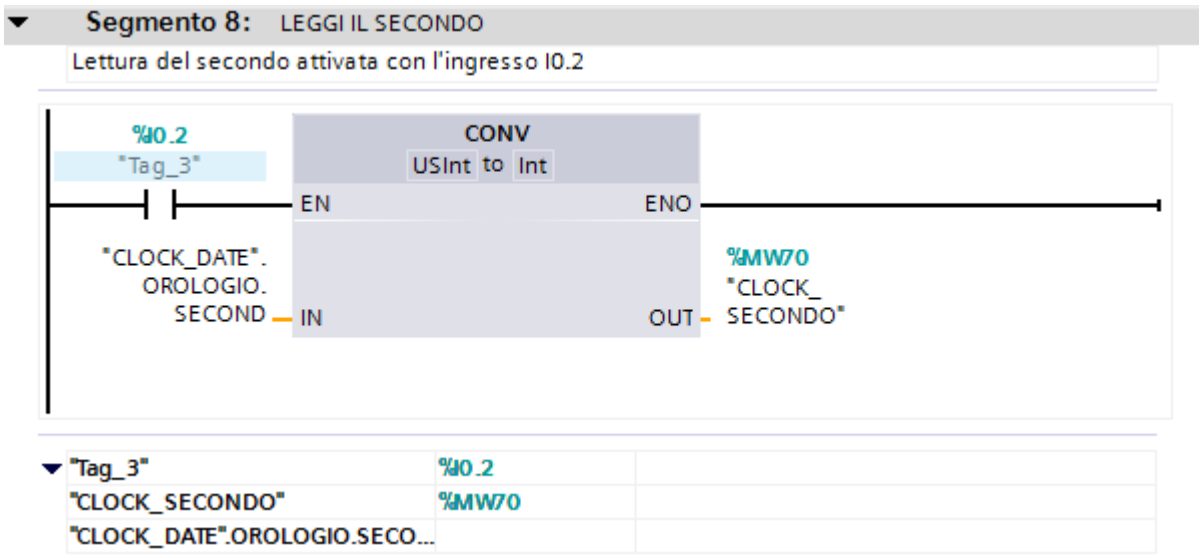


Fig.298 :Gestione Orologio Hardware (parte 8)

Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

LEGGI LA SETTIMANA

Quando il blocco riceve l'impulso da "**I0.2**", andiamo a leggere il DB in cui è salvata la temporizzazione e preleviamo la settimana che poi verrà convertito in numero per far dei confronti.
(settimana inglese che definisce la domenica come primo giorno della settimana e il sabato come l'ultimo)

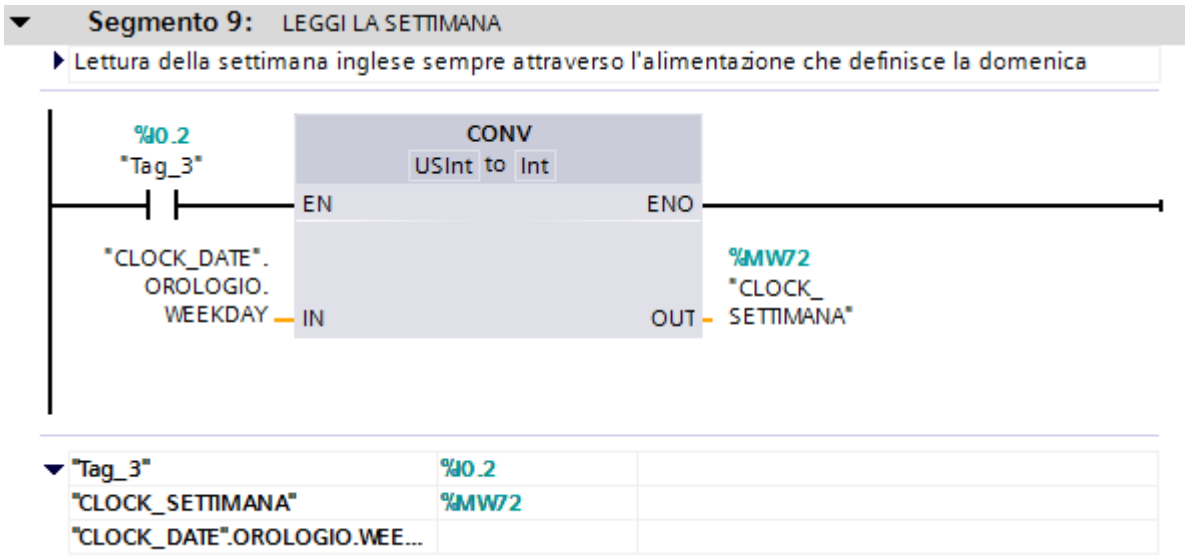


Fig.299 :Gestione Orologio Hardware (parte 9)

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Gestione fotocopiatrici e PC

esperienza 71

Marco Morigi
5AET 2013
(marco.morigi26@gmail.com)

OBBITTIVO:
Una serie di apparecchiature elettroniche va controllata nell'accensione e nello spegnimento.
ORE DI NON UTILIZZO: dalle 19.30 alle 7.30 del mattino successivo.
GIORNI DI UTILIZZO: da LUNEDI' a VENERDI'.
MANUALE: possibilita' di controllare in modo manuale l'accensione e spegnimento delle macchine.
I0.0: interruttore di funzionamento automatico
I0.1: interruttore di funzionamento manuale
Q0.0: rele' controllo impianto
M0.0: merker ausiliario di azionamento

Inseriamo la data nel PLC e subito dopo andiamo a leggere tutti i valori corrispondenti a: ANNO, MESE, GIORNO, ORA, MINUTO, SECONDO e SETTIMANA.
Questi valori andranno poi convertiti per poter fare le dovute operazioni di confronto.
NB: Impostiamo la settimana da configurare come quella inglese quindi il primo giorno della settimana corrisponde alla domenica mentre l'ultimo al sabato.

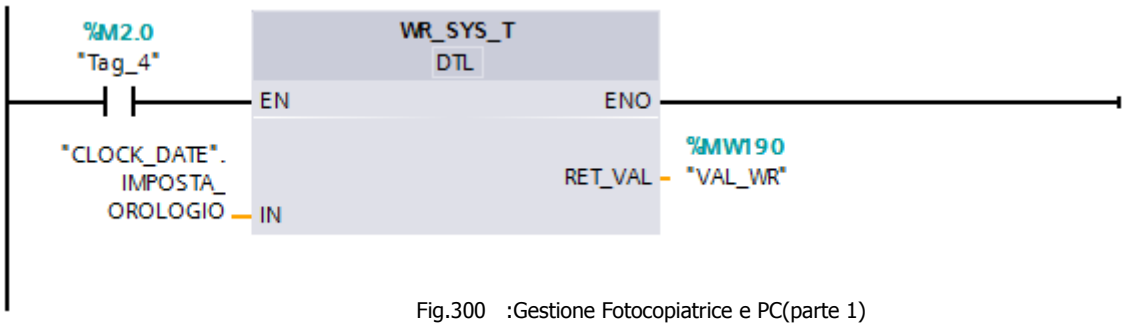


Fig.300 :Gestione Fotocopiatrice e PC(parte 1)

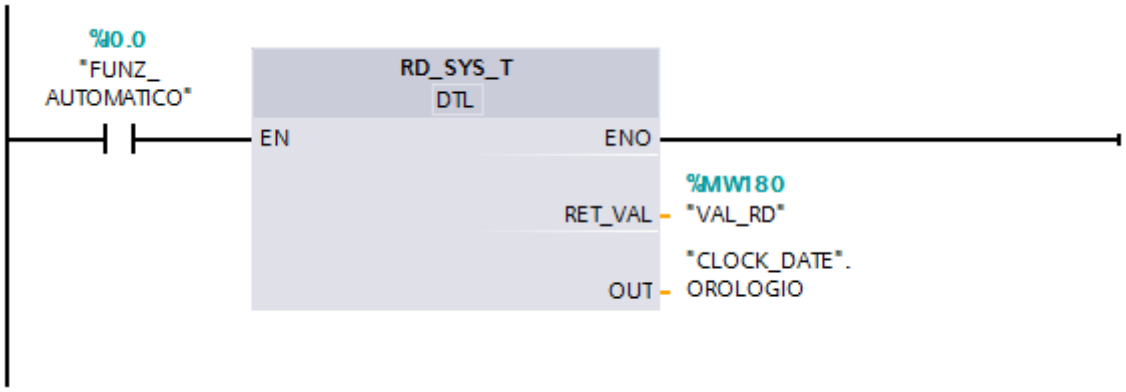


Fig.301 :Gestione Fotocopiatrice e PC(parte 2)

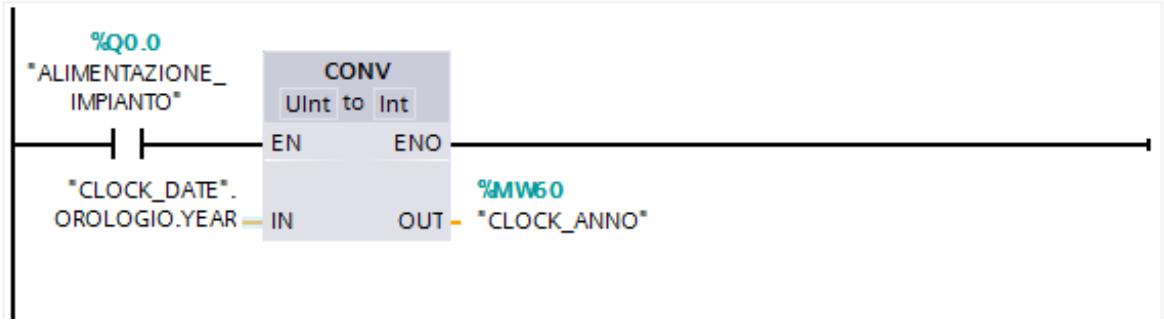


Fig.302 :Gestione Fotocopiatrice e PC(parte 3)

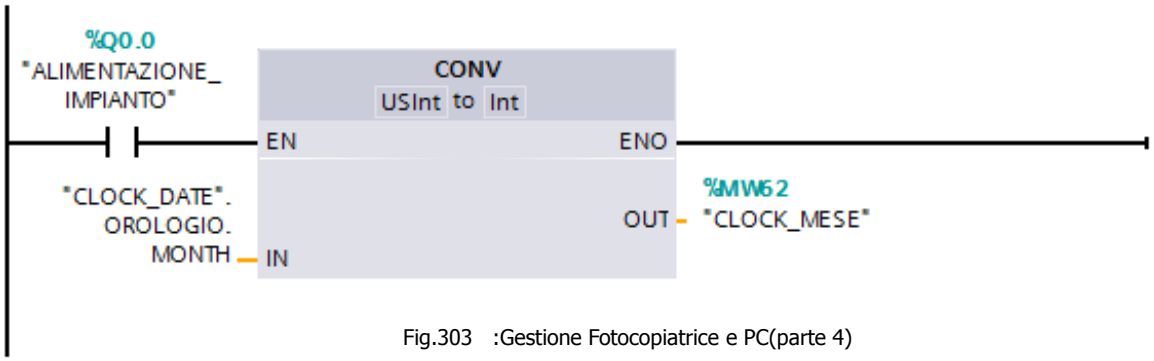
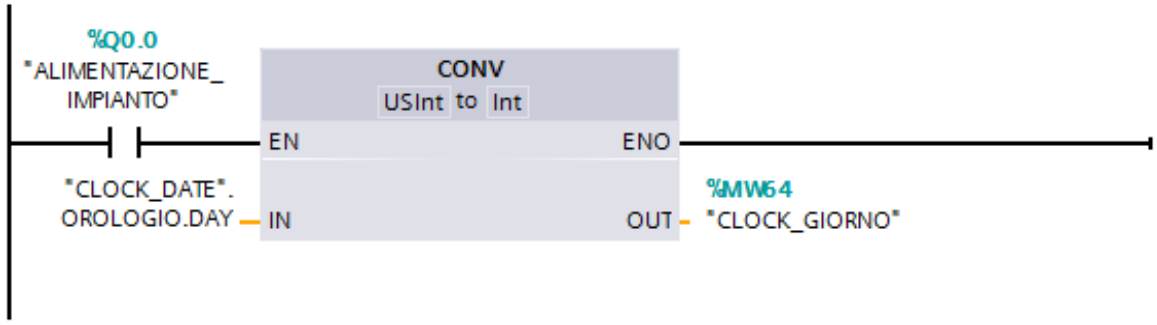
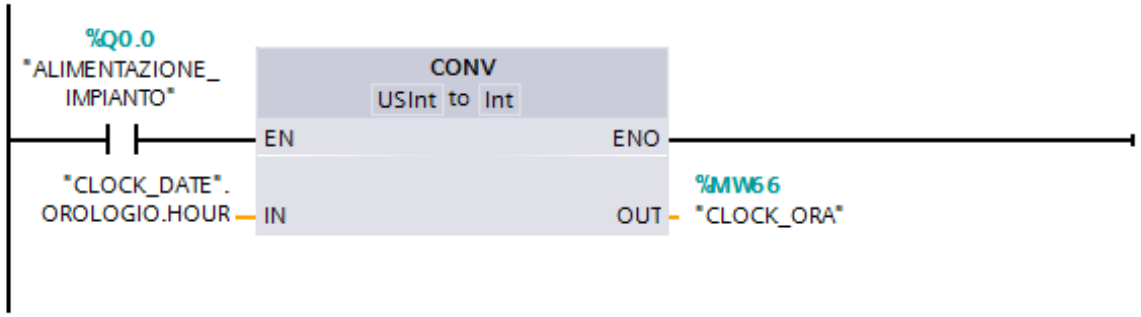


Fig.303 :Gestione Fotocopiatrice e PC(parte 4)





Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

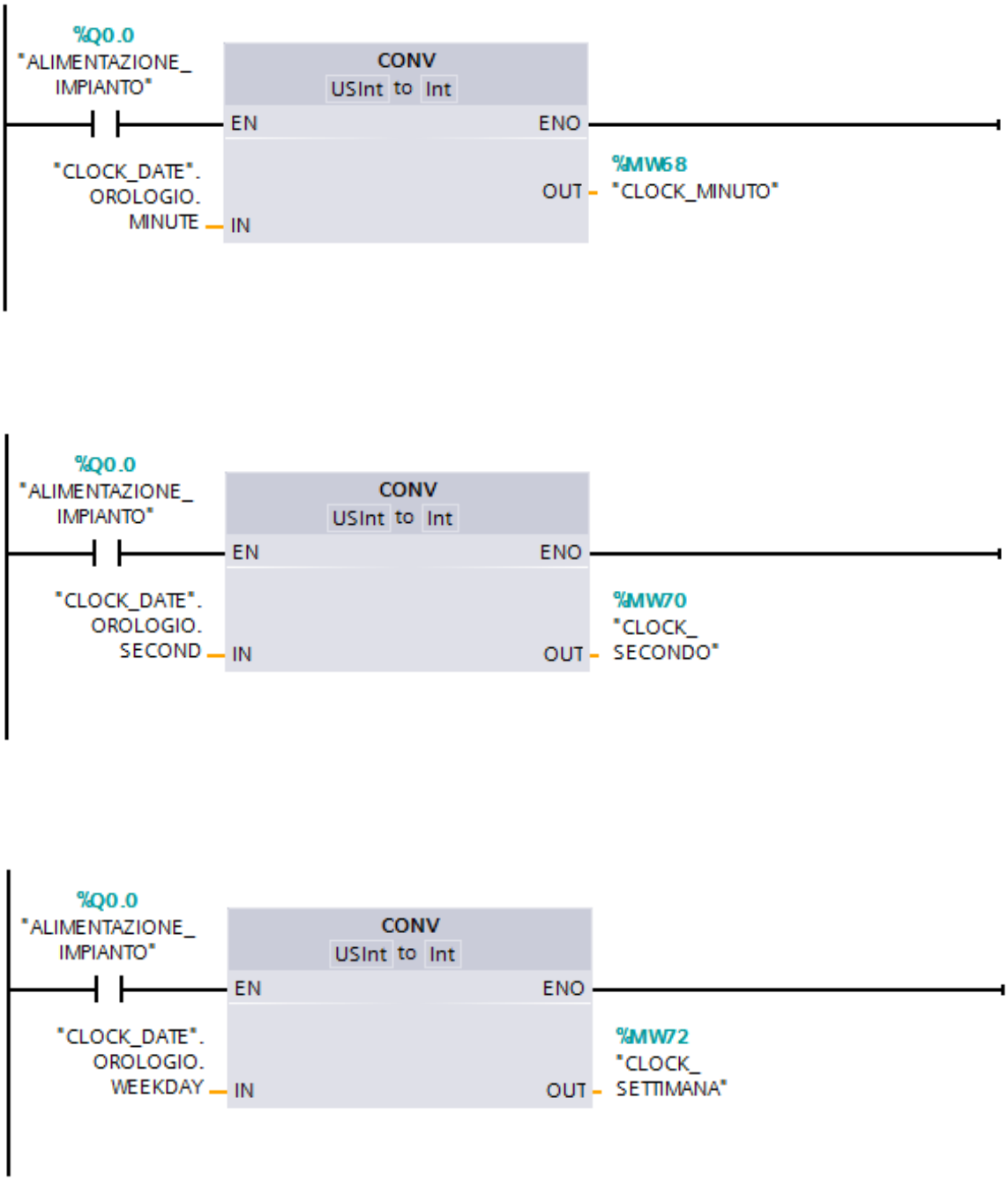


Fig.304 :Gestione Fotocopiatrice e PC(parte 5)

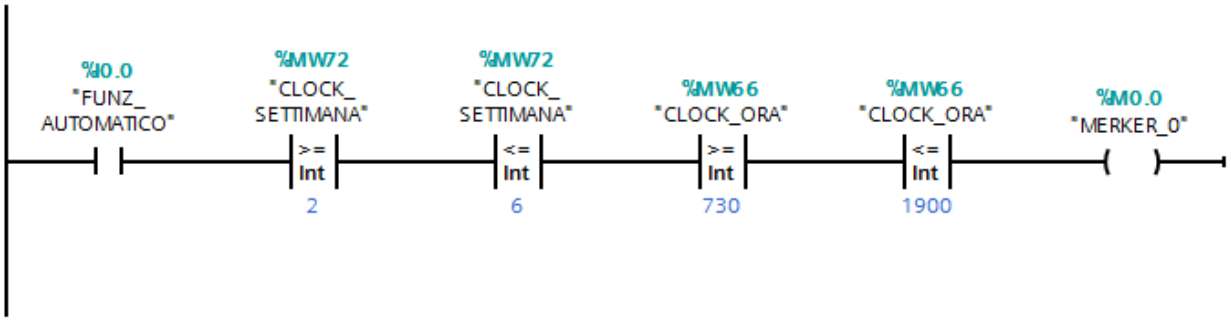
Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

Sapendo che la nostra stampante lavora sia in modo automatico che in modo manuale, il primo contatto ci permette di determinare che la stampante lavora in MODO AUTOMATICO**.**

Una volta che è stato impostato lo switch, e tutti i controlli sono "veri", cioè ci troviamo tra il lunedì e il venerdì, e ci si trova nell'arco di orario compreso tra le 7.30 e le 19.00, allora si attiverà l'uscita "M0.0".
A sua volta l'abilitazione del merker, insieme a quella del funzionamento automatico, ci permette di attivare l'uscita "Q0.0" che corrisponde all'alimentazione delle fotocopiatrici e dei PC.
Si può notare che è possibile anche alimentare l'impianto fuori dagli orari di lavoro, attivando semplicemente il pulsante "FUNZIONE MANUALE".

Segmento 10: GESTIONE DEL FUNZIONAMENTO AUTOMATICO.

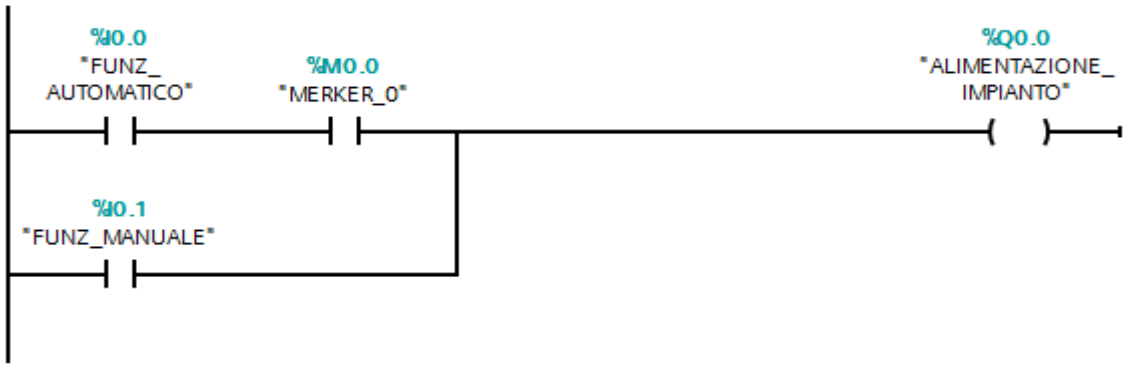
gestione del funzionamento automatico della stampante nei giorni lavorativi (da lunedì a venerdì)



"FUNZ_AUTOMATICO"	%I0.0
"CLOCK_ORA"	%MW66
"CLOCK_SETTIMANA"	%MW72
"MERKER_0"	%M0.0

Segmento 11: GESTIONE DELL'ALIMENTAZIONE.

Con questo segmento possiamo far lavorare in modo automatico la nostra stampante solo se ci



"FUNZ_AUTOMATICO"	%I0.0
"ALIMENTAZIONE_IMPIANTO"	%Q0.0
"MERKER_0"	%M0.0
"FUNZ_MANUALE"	%I0.1

ITIS "B. Pascal - P.le Macrelli, 100 - 47023 Cesena (FC) - tel. + 39 0547 22792 - fax +39 0547 24141
TDP 5^A ELETTRONICA TELECOMUNICAZIONI Prof, FERRARO GIUSEPPE Ass. Tecnico MARCO FERRETTI
Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Gestione illuminazione esterna

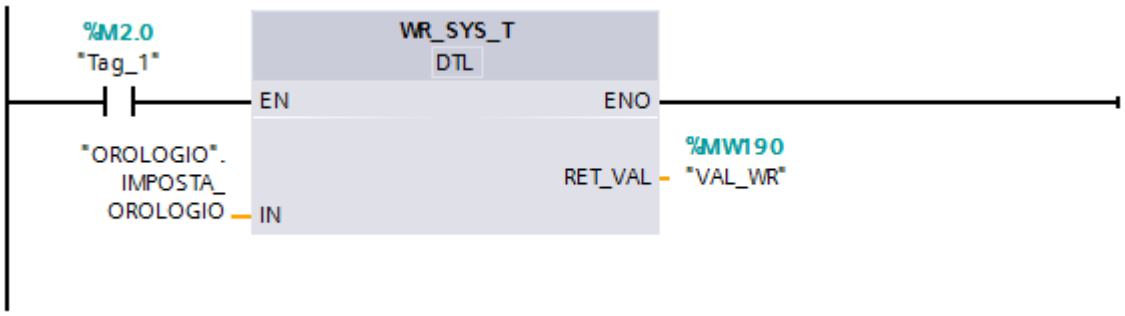
esperienza 72

Marco Morigi
5AET 2013
(marco.morigi26@gmail.com)

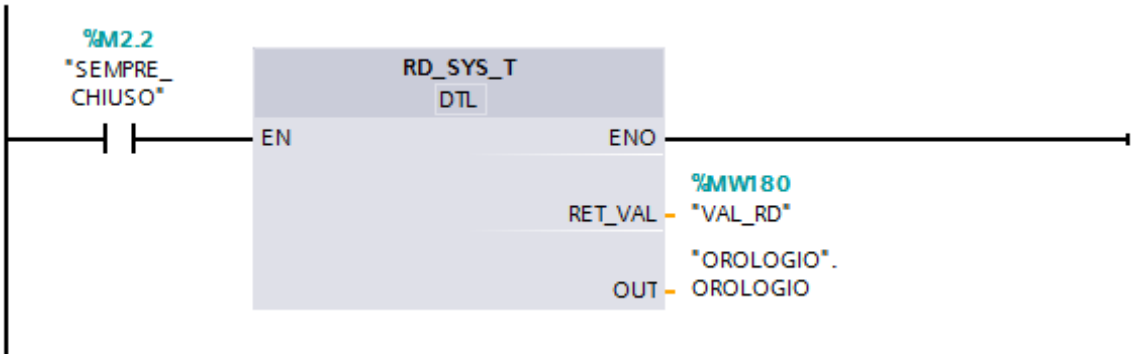
OBIETTIVO:
Si deve controllare l'impianto di illuminazione esterna di una civile abitazione.
La regolazione puo' essere sia MANUALE che AUTOMATICA (o una o l'altra, ovviamente).
Ci sono DUE gruppi di luci: GRUPPO PRINCIPALE e GRUPPO SECONDARIO .
Il GRUPPO PRINCIPALE si attiva in automatico dalle ore 6.00 alle ore 24.00 utilzizando un interruttore CREPUSCOLARE.
In automatico il GRUPPO SECONDARIO funziona dalle 6.00 alle 8.00 del mattino e dalle 19.00 alle 24.00 SE E SOLO SE intervengono CONTEMPORANEAMENTE sia il CREPUSCOLARE che il sensore di PRESENZA
Nel caso di controllo MANUALE entrambi i gruppi di luce si accendono e l'intervento e' condizionato solo dall'interruttore preposto per tale funzione.

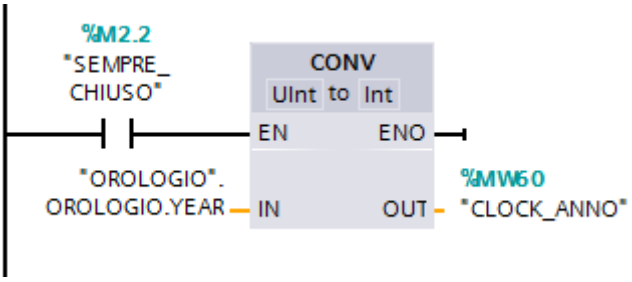
I0.0: CREPUSCOLARE
I0.1: PRESENZA
I0.2: AUTOMATICO
I0.3: MANUALE
Q0.0: GRUPPO PRINCIPALE
Q0.1: GRUPPO SECONDARIO
SM0.0: merker di primo ciclo

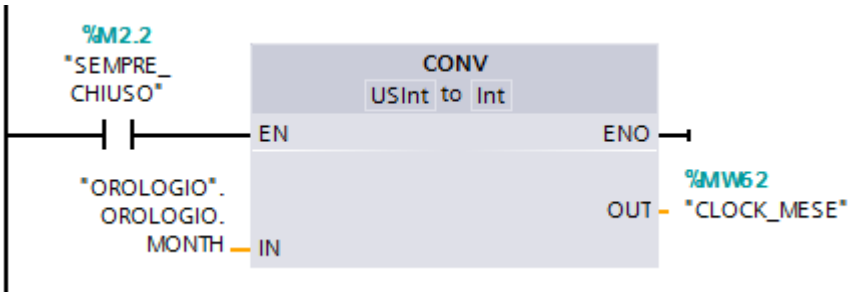
Inseriamo la data nel PLC e subito dopo andiamo a leggere tutti i valori corrispondenti a: ANNO, MESE, GIORNO, ORA, MINUTO, SECONDO e SETTIMANA.
Questi valori andranno poi convertiti per poter fare le dovute operazioni di confronto.
NB: Impostiamo la settimana da configurare come quella inglese quindi il primo giorno della settimana corrisponde alla domenica mentre l'ultimo al sabato.

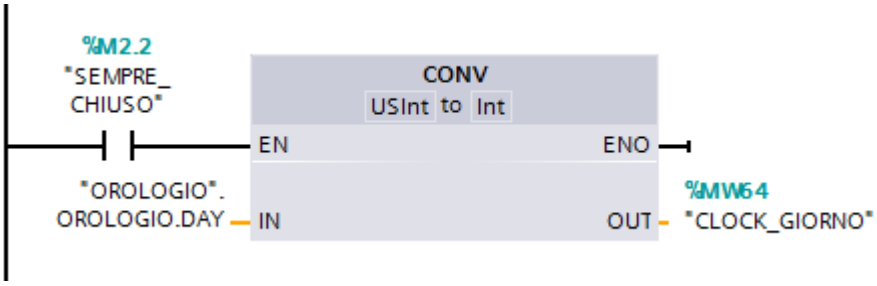


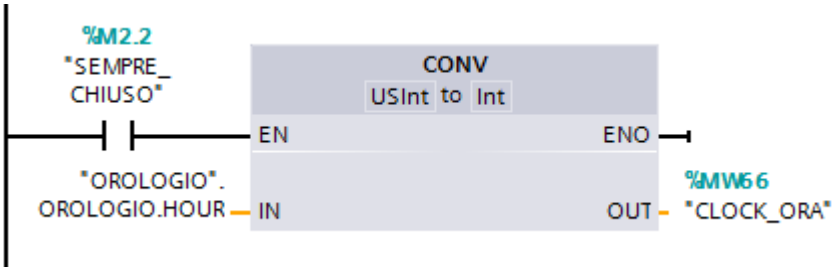
Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com











Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

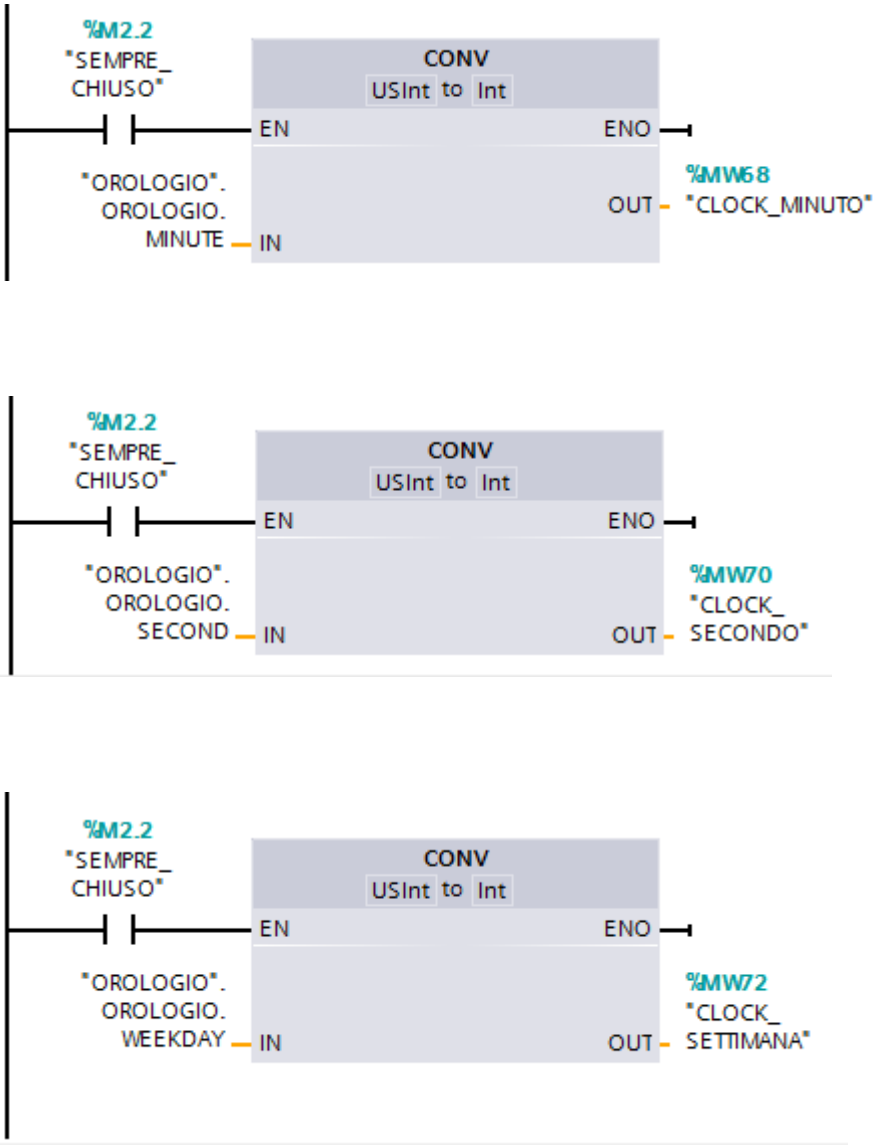


Fig.306 :Gestione Illuminazione esterna(Parte 1)

Con questo segmento possiamo far lavorare in MODO AUTOMATICO il gruppo principale di luci esterne solo se l'interruttore crepuscolare è attivo e se ci troviamo in un orario compreso fra le 6.00 e le 24.00 (come da consegna).
Possiamo anche alimentare il gruppo principale con il FUNZIONAMENTO MANUALE.

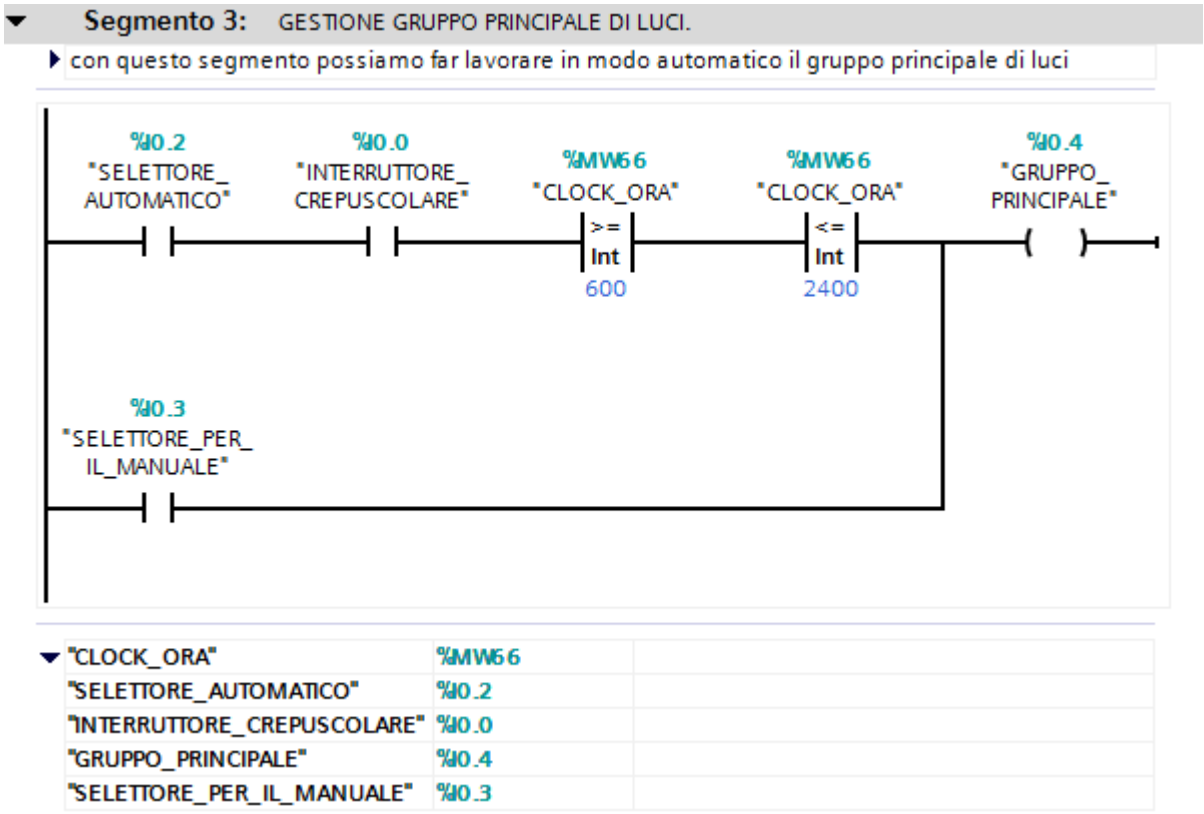


Fig.307 :Gestione Illuminazione esterna(Parte 2)

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Con quest'altro segmento possiamo far lavorare in MODO AUTOMATICO il gruppo secondario di luci solo se l'interruttore crepuscolare e il sensore di presenza sono attivi contemporaneamente.
Se così fosse il gruppo di luci si accenderebbe se ci trovassimo in un orario compreso fra le 6.00 e le 8.00 di mattina o fra le 19.00 e le 24.00 di sera.
Il gruppo secondario di luci, come anche il primario, si può attivare in qualsiasi momento attivando il selettore per il FUNZIONAMENTO in modalità MANUALE.

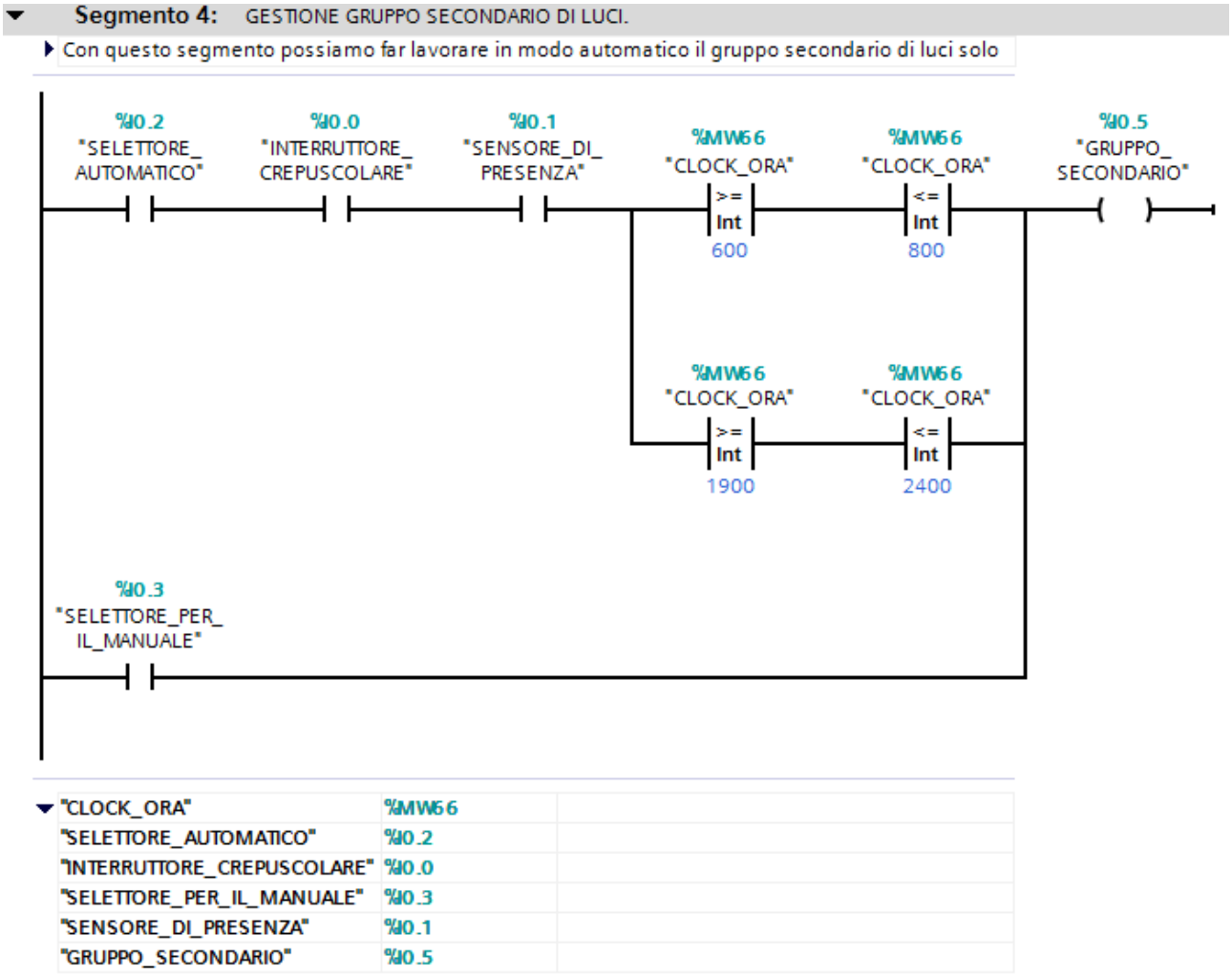


Fig.308 :Gestione Illuminazione esterna(Parte 3)

Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

Gestione caldaia

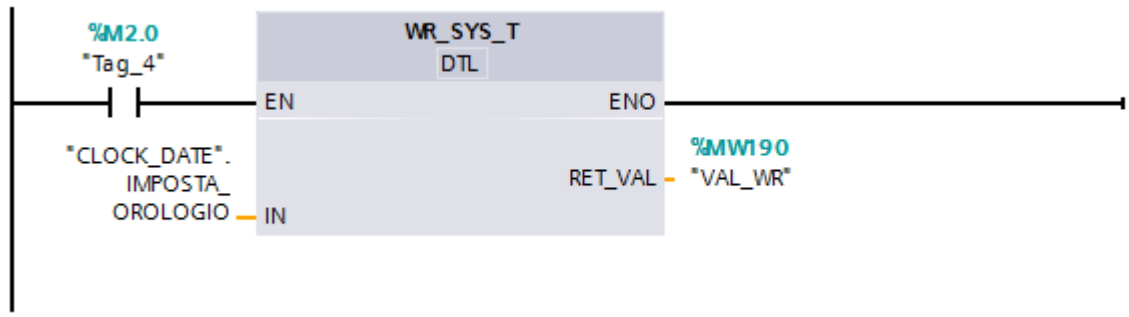
esperienza 73

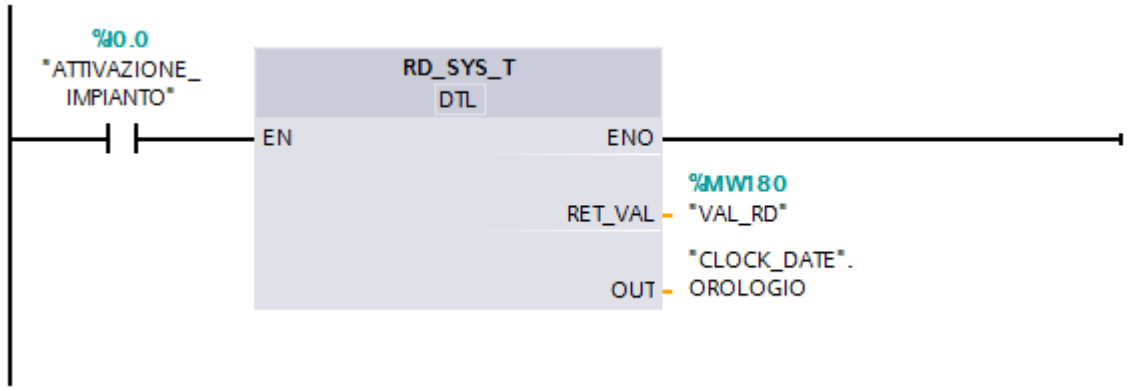
Marco Morigi
5AET 2013
(marco.morigi26@gmail.com)

OBIETTIVO:
Si deve controllare il funzionamento di un impianto di riscaldamento per uso civile abitazione. Serve un PLC con funzione di orologio hardware.
Ci sono diverse fasce a seconda del giorno della settimana.
Da prevedere la possibilità di avere diversa gestione a seconda del mese di operatività del sistema.
INVERNO (NOVEMBRE-MARZO)
DOMENICA: se la temperatura misurata e' inferiore a quella impostata allora la caldaia funziona dalle 7.00 alle 21.00
Nei giorni feriali: dalle 7.00 alle 10.00 e dalle 14.00 alle 20.00

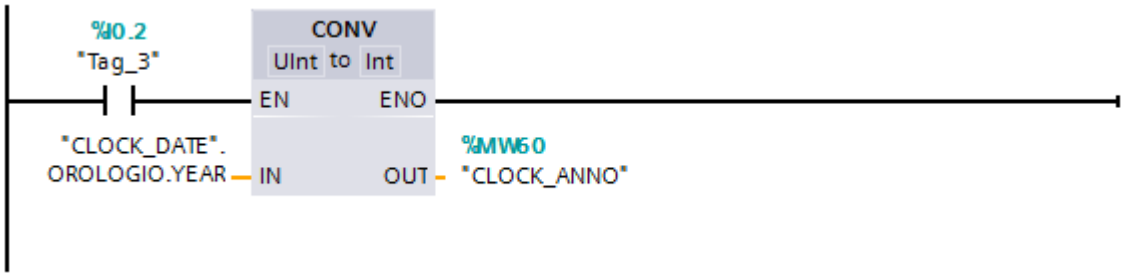
Q0.1: comanda la caldaia
i0.0: INTERRUTTORE ATTIVAZIONE IMPIANTO
I0.1: SEGNALE CHE INDICA, SE 1, CHE LA TEMPERATURA DELL'AMBIENTE HA RAGGIUNTO IL VALORE IMPOSTATO

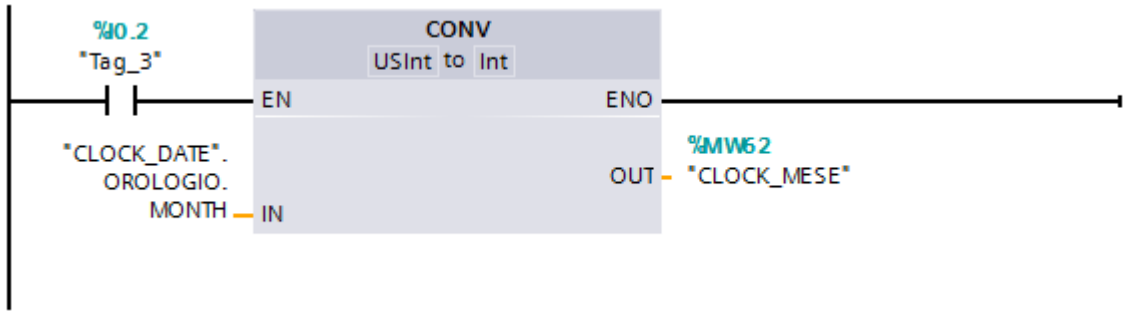
Inseriamo la data nel PLC e subito dopo andiamo a leggere tutti i valori corrispondenti a: ANNO, MESE, GIORNO, ORA, MINUTO, SECONDO e SETTIMANA.
Questi valori andranno poi convertiti per poter fare le dovute operazioni di confronto.
NB: Impostiamo la settimana da configurare come quella inglese quindi il primo giorno della settimana corrisponde alla domenica mentre l'ultimo al sabato.

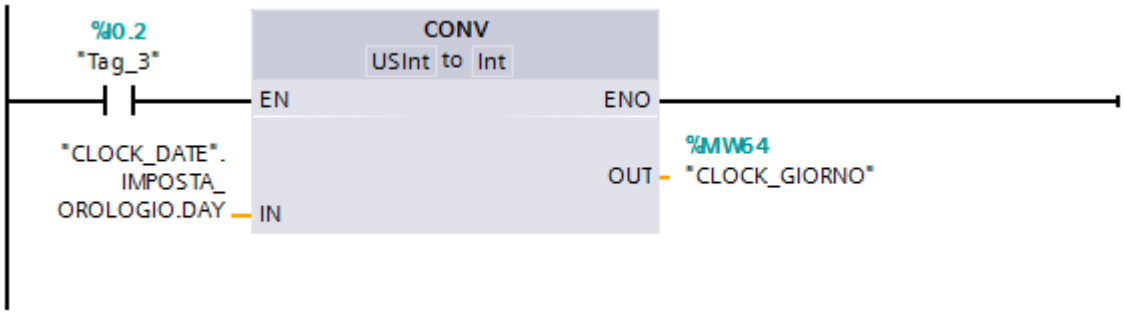


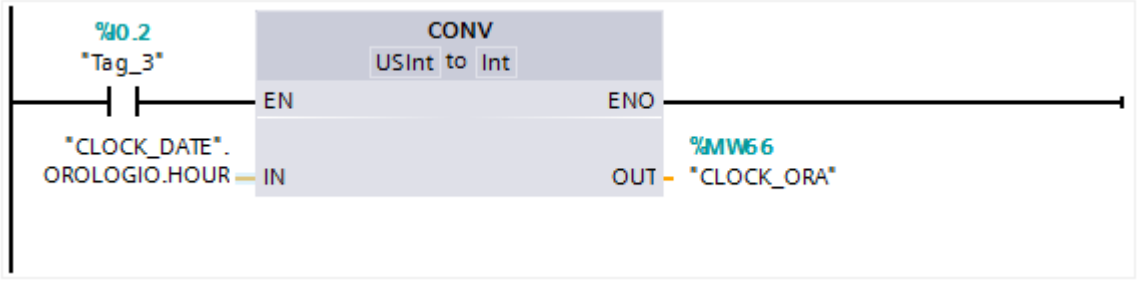


Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com









Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

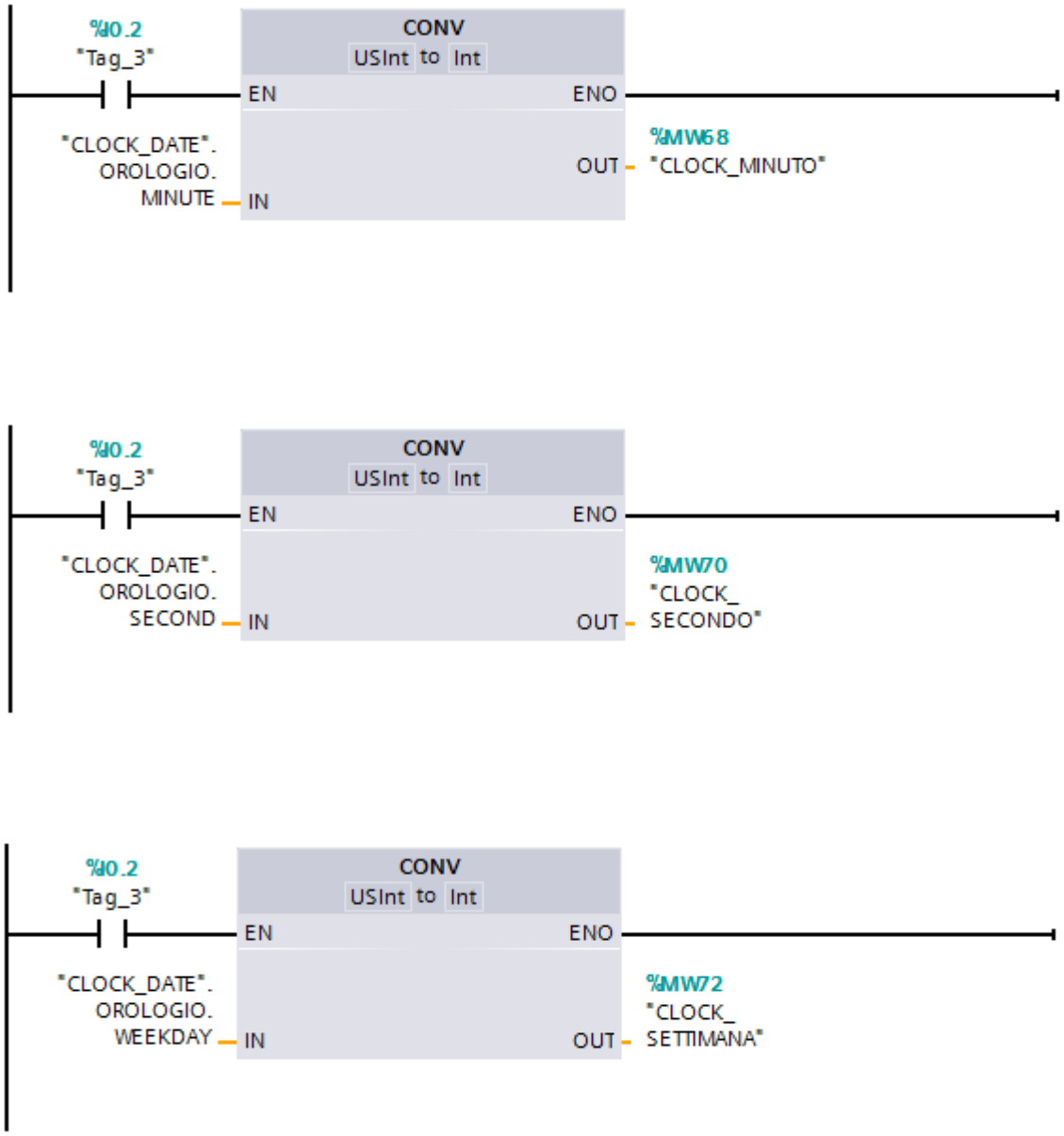
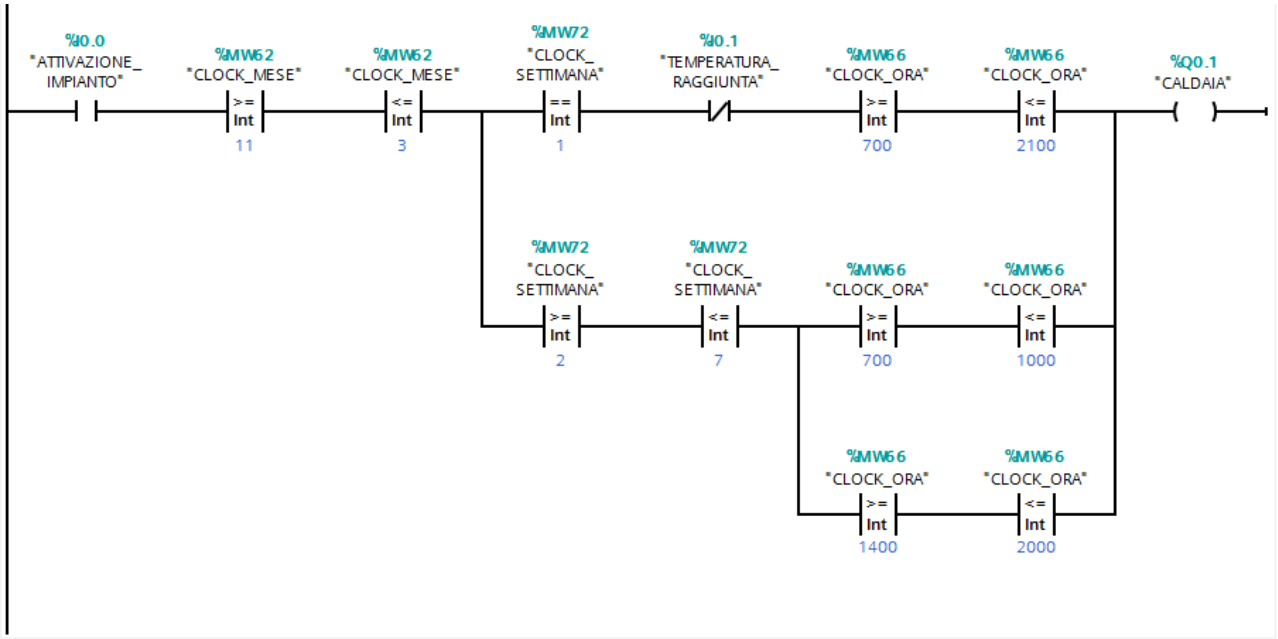


Fig.309 :Gestione caldaia(parte 1)

Come da consegna, se l'impianto viene attivato possiamo vedere le diverse condizioni sviluppate in sequenza.
Possiamo trovare il controllo del periodo mensile (da novembre a marzo), il controllo sul giorno settimanale (differenziando la domenica dagli altri giorni) e il controllo sull'orario.
Da notare che di domenica la caldaia funziona solo se la temperatura misurata e' inferiore a quella impostata, infatti il controllo "I0.1" serve per avvisare l'utente che la temperatura dell'ambiente ha raggiunto il valore che era stato impostato e quindi interrompe il funzionamento della caldaia.



"ATTIVAZIONE_IMPIANTO"	%I0.0
"CLOCK_MESE"	%MW62
"CLOCK_ORA"	%MW66
"CLOCK_SETTIMANA"	%MW72
"TEMPERATURA_RAGGIUNTA"	%I0.1
"CALDAIA"	%Q0.1

Fig.310 :Gestione caldaia(parte 2)

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Gestione apertura tende

esperienza 74

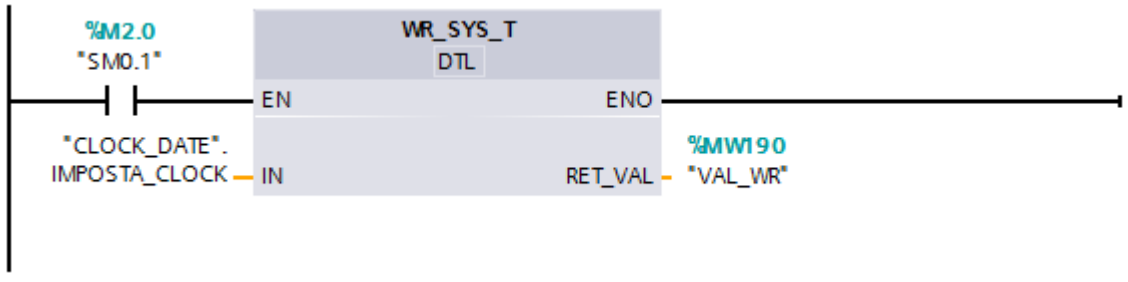
Marco Morigi
5AET 2013
(marco.morigi26@gmail.com)

OBIETTIVO:
Si vuole automatizzare l'apertura delle tende in una civile abitazione.
Le tende devono essere aperte dalle ore 7.00 del lunedì..venerdì mattina, dalle ore 8 del sabato mattina, dalle ore 10 della domenica mattina-
Per la chiusura automatica delle tende si utilizza un interruttore crepuscolare che risulta ON in condizioni di crepuscolo mentre e' OFF in condizioni di luce solare.
Si prevede anche un funzionamento MANUALE del motore in modo che in condizioni di emergenza o necessita si possa intervenire manualmente sull'azionamento delle tende.
Ci sono due finecorsa: in apertura della tenda e in chiusura della tenda. I due finecorsa sono NORMALMENTE CHIUSI e quando si raggiunge la fine della corsa si aprono.

I0.0: interruttore crepuscolare
I0.1: pulsante di apertura manuale
I0.2: pulsante di chiusura manuale
I0.3: finecorsa delle tende in apertura
I0.4: finecorsa delle tende chiuse
I0.5: interruttore di funzionamento automatico/manuale
Q0.0: rele' di apertura tende
Q0.1: rele' di chiusura tende

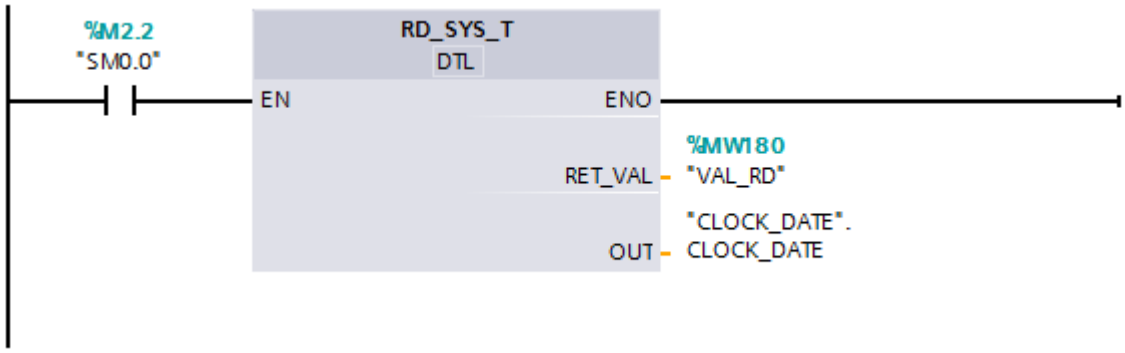
Inseriamo la data nel PLC e subito dopo andiamo a leggere tutti i valori corrispondenti a: ANNO, MESE, GIORNO, ORA, MINUTO, SECONDO e SETTIMANA.
Questi valori andranno poi convertiti per poter fare le dovute operazioni di confronto.
NB: Impostiamo la settimana da configurare come quella inglese quindi il primo giorno della settimana corrisponde alla domenica mentre l'ultimo al sabato.

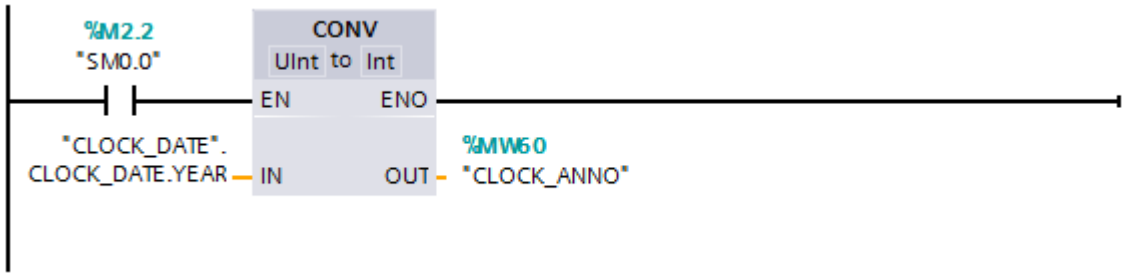
M2.0 e' un MERKER che va a 1 solamente nel primo ciclo di esecuzione del programma.

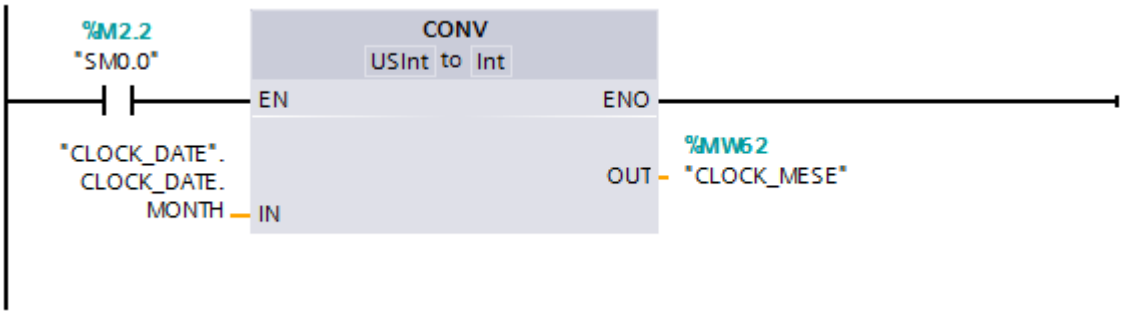


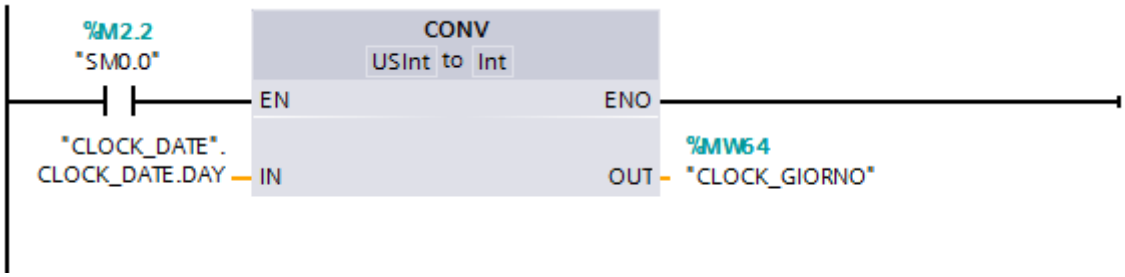
Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

M2.2: e' un merker di sistema che e' sempre a 1









Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

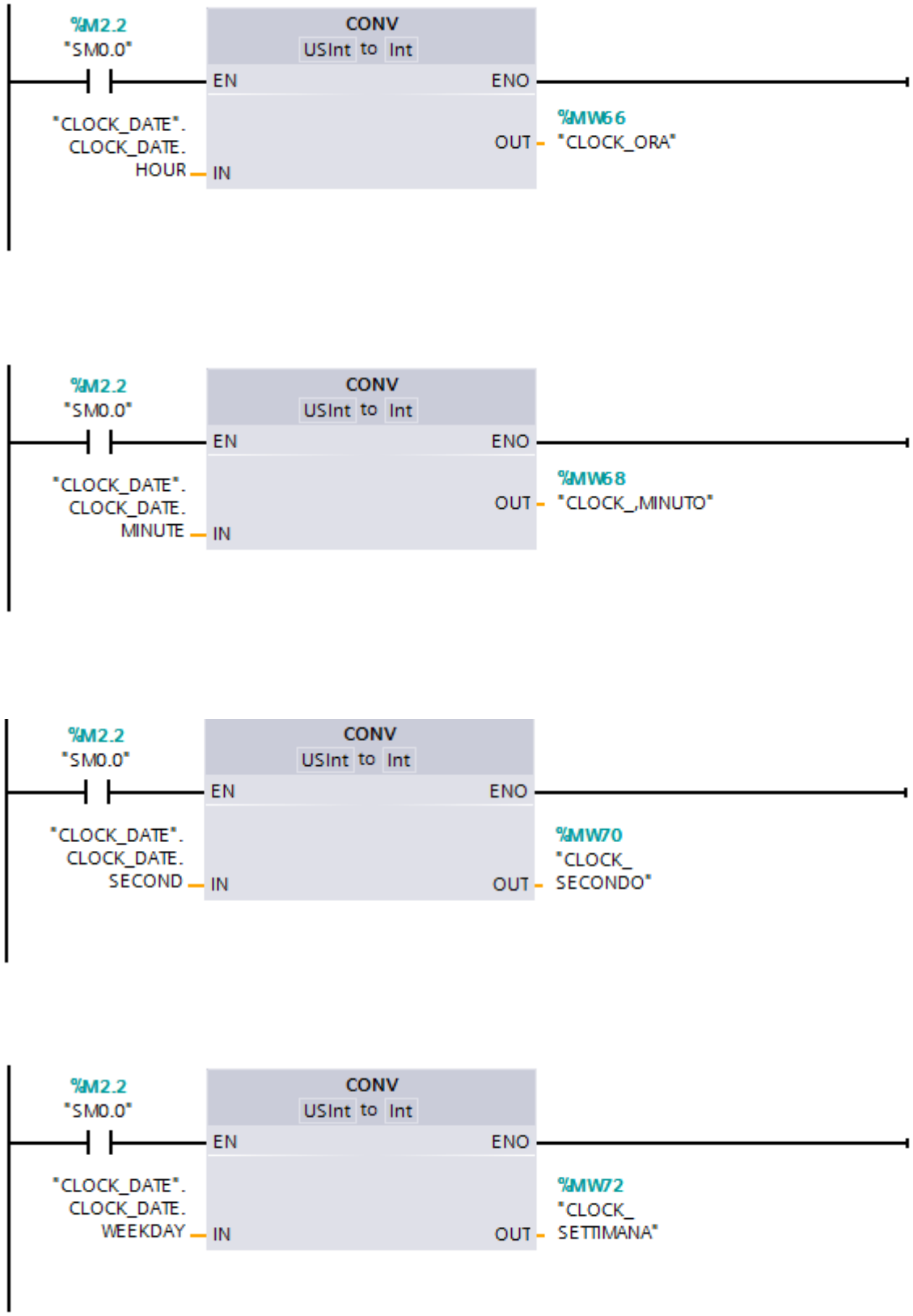


Fig.311 :Gestione apertura tende(parte 1)

Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

Con questo segmento si gestisce l'apertura delle tende, come da consegna, in modo automatico e con orari e giorni stabiliti.
Infatti si può notare che quando il contatto del funzionamento MANUALE_AUTOMATICO è attivo, andiamo a controllare in quale giorno della settimana ci troviamo con un blocco di confronto riferito a "CLOCK_SETTIMANA".
Il blocco "IN RANGE" da in uscita un 1 solo se ci troviamo all'interno del range impostato, ovvero 2..6 (da lunedì a venerdì secondo l'attribuzione dei giorni della settimana inglese).
Dopo aver controllato il giorno della settimana, andiamo controllare l'orario seguendo i le tempistiche richieste dalla consegna (nel nostro caso: lun..ven 7.00, sab 8.00, dom 10.00).
Quando anche questi secondi confronti sono veri, viene abilitato il merker "M0.0" che servirà successivamente per abilitare il motore che apre le tende.

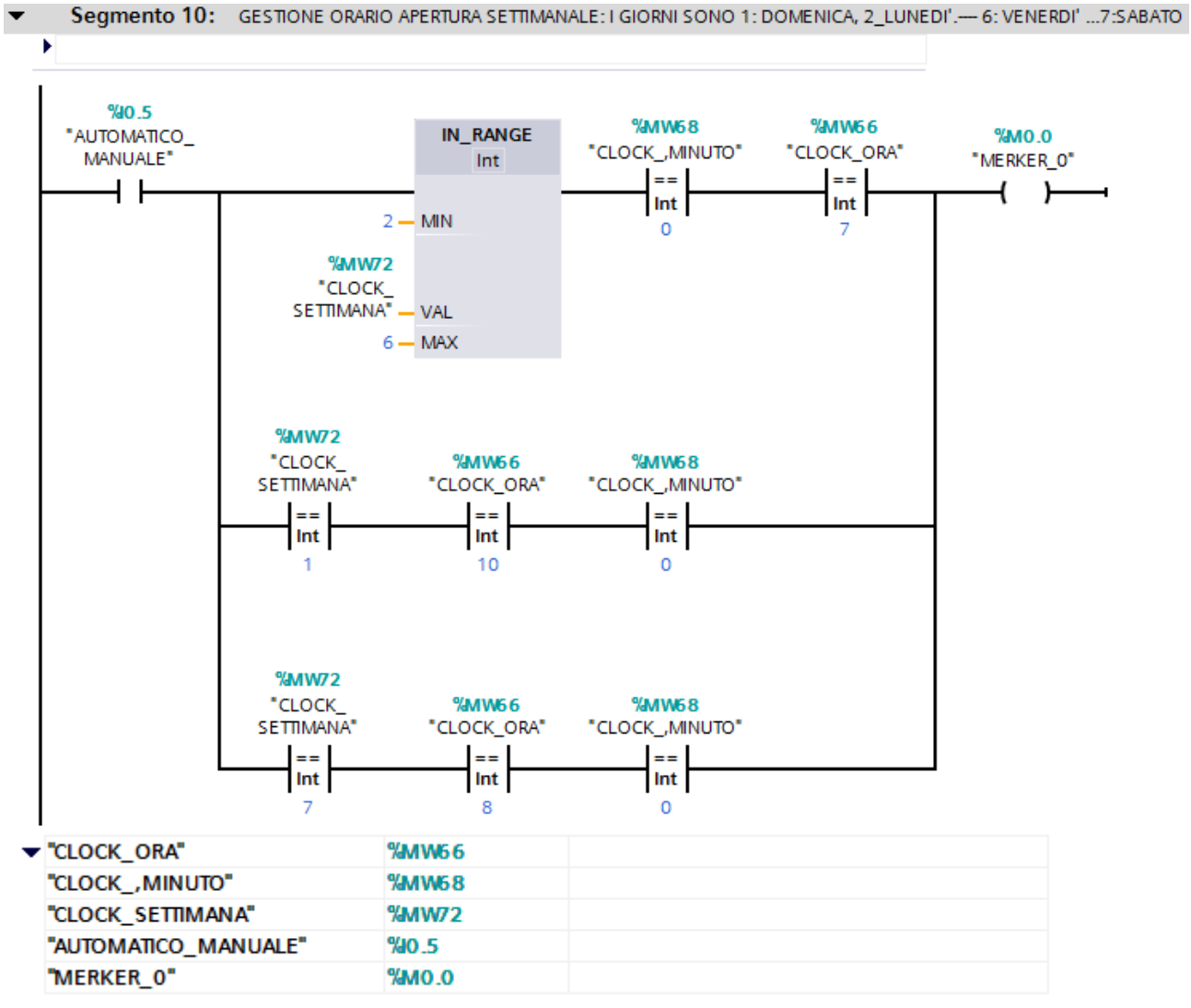


Fig.312 :Gestione apertura tende(parte 2)

Nel caso di FUNZIONAMENTO AUTOMATICO, se siamo in presenza del crepuscolo, viene settato l'ingresso "I0.0" che a sua volta attiverà il merker "M0.3".
Quest'ultimo verrà usato per la CHIUSURA AUTOMATICA della tenda.

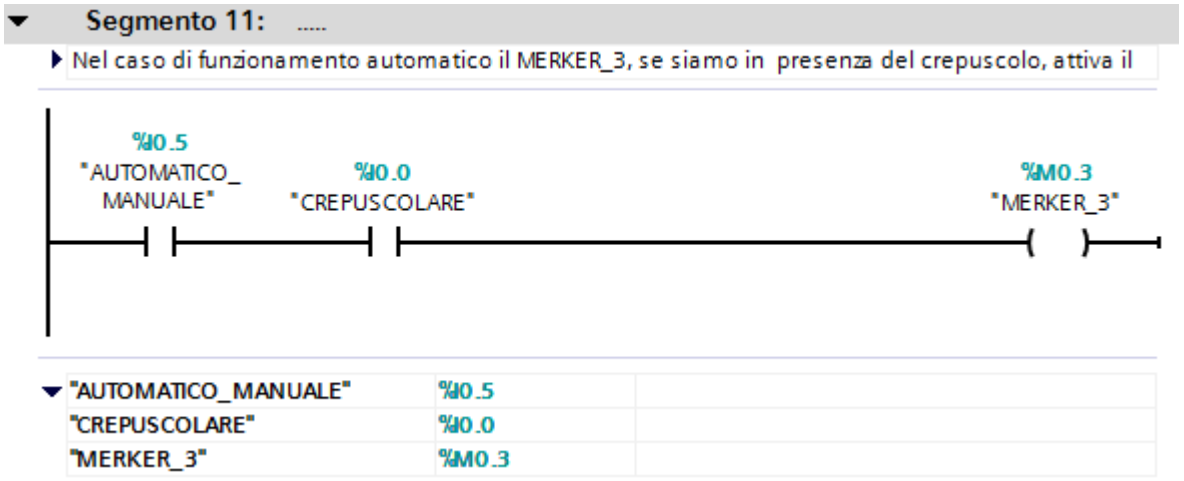


Fig.313 :Gestione apertura tende(parte 3)

Nel caso di FUNZIONAMENTO MANUALE, il merker "M0.1" viene abilitato.
Se si vuole chiudere la tenda in MODO MANUALE, è necessario che l'ingresso "I0.5" sia a zero, quindi vuol dire che dobbiamo essere nello stato MANUALE.
Il merker sarà usato per stabilire se si decide di chiudere la tenda oppure no.

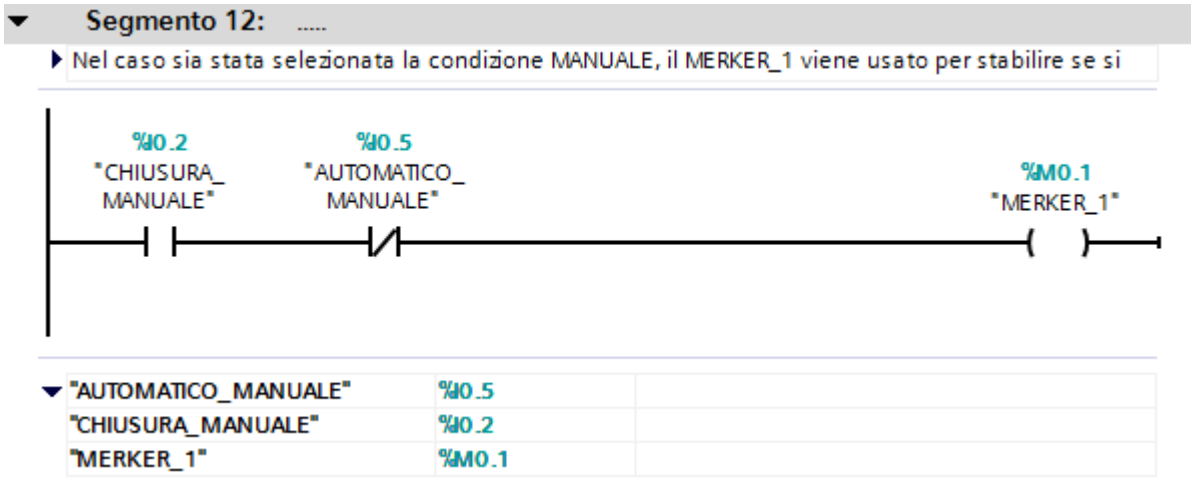


Fig.314 :Gestione apertura tende(parte 4)

Nel caso di FUNZIONAMENTO MANUALE, il merker "M0.2" viene abilitato se si vuole eseguire un'apertura della tenda in MODO MANUALE, cioè in un orario non gestito dal FUNZIONAMENTO AUTOMATICO.
Il merker sarà usato per stabilire se si decide di aprire la tenda.

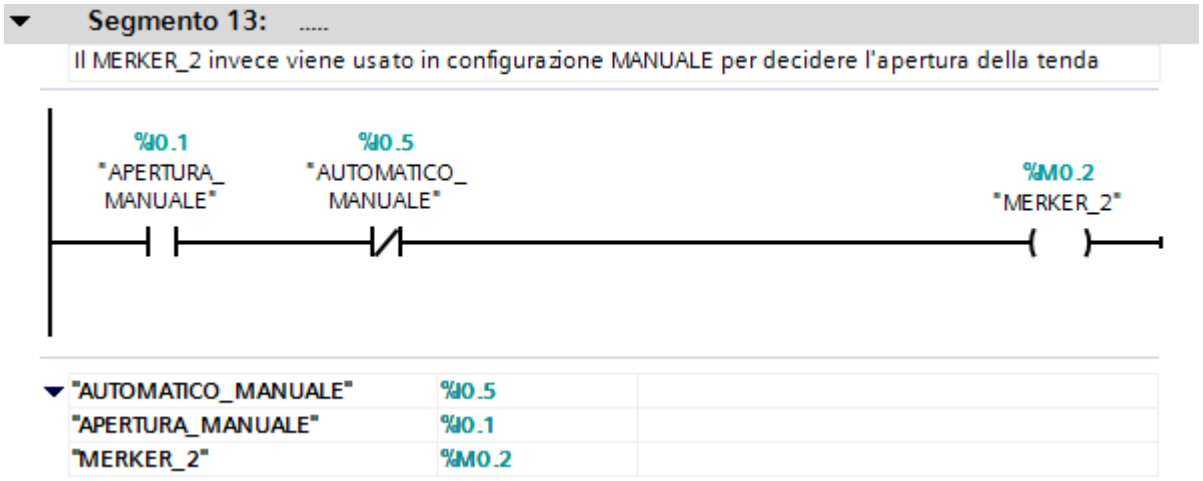


Fig.315 :Gestione apertura tende(parte 4)

Con questo segmento gestiamo il motore che apre le tende.
Se "M0.0" (merker per l'apertura secondo il FUNZIONAMENTO AUTOMATICO) è attivo, il motore per l'apertura delle tende verrà attivato.
Molto importante è il controllo sul finecorsa per l'apertura delle tende "I0.3"; esso è un contatto normalmente chiuso e commuta quando la tenda è completamente aperta.
Quando il contatto si apre, il motore non viene più alimentato, evitando così la rottura di eventuali parti.
Anche per "M0.2" (merker per l'apertura secondo il FUNZIONAMENTO MANUALE) il funzionamento è lo stesso.

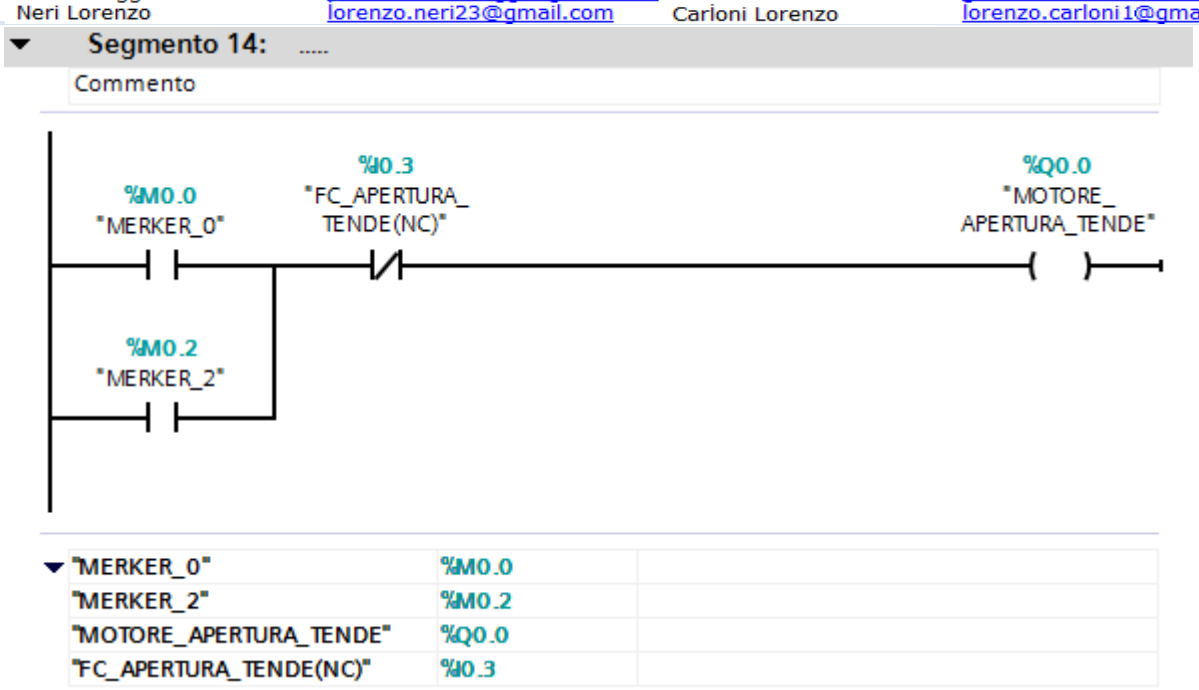


Fig.316 :Gestione apertura tende(parte 5)

Con questo segmento gestiamo il motore che chiude le tende.
Se "M0.3" (merker per la chiusura secondo il FUNZIONAMENTO AUTOMATICO) è attivo, il motore per la chiusura delle tende verrà attivato.
Molto importante è il controllo sul finecorsa per la chiusura delle tende "I0.4"; esso è un contatto normalmente chiuso e commuta quando la tenda è completamente chiusa.
Quando il contatto si apre, il motore non viene più alimentato, evitando così la rottura di eventuali parti.
Anche per "M0.1" (merker per la chiusura secondo il FUNZIONAMENTO MANUALE) il funzionamento è lo stesso.

Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

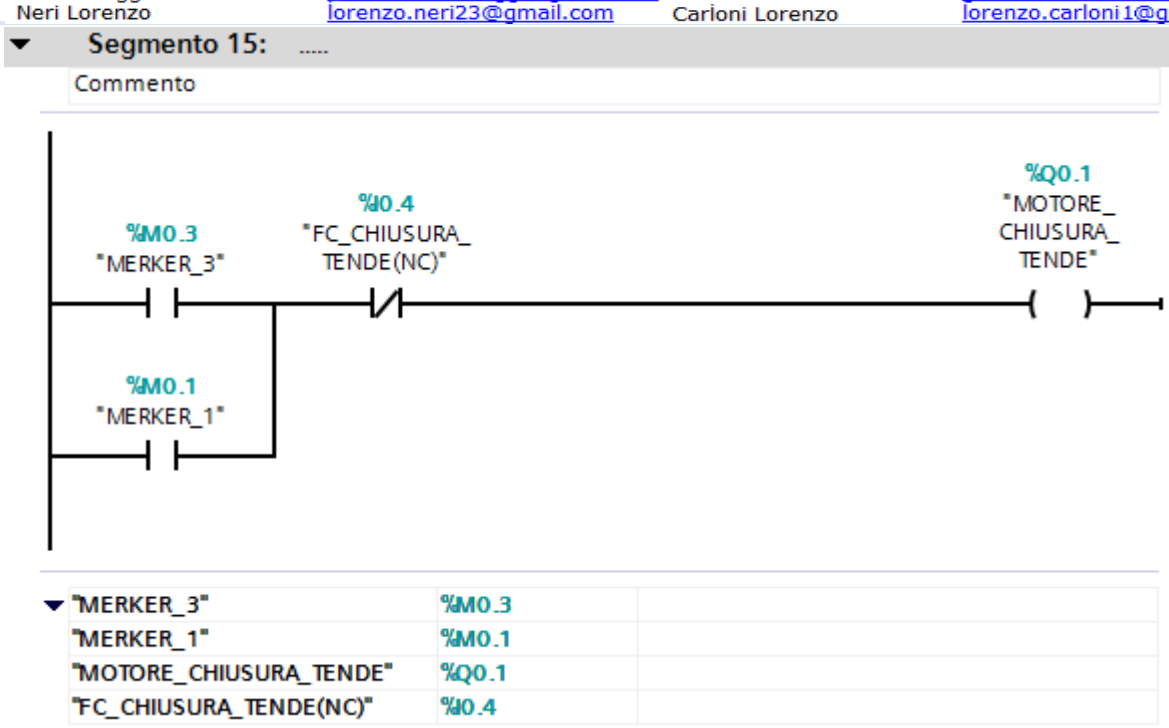


Fig.317 :Gestione apertura tende(parte 6)

Ù

ITIS "B. Pascal - P.le Macrelli, 100 - 47023 Cesena (FC) - tel. + 39 0547 22792 - fax +39 0547 24141
TDP 5^A ELETTRONICA TELECOMUNICAZIONI Prof, FERRARO GIUSEPPE Ass. Tecnico MARCO FERRETTI
Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Gestione analogica

esperienza 75

Lorenzo Carloni
Marco Morigi
5AET 2013
(lorenzo.carloni1@gmail.com)
(marco.morigi26@gmail.com)

Con questo programma andremo a gestire ingressi e uscite ANALOGICHE.
Durante la creazione del progetto, dopo aver scelto il dispositivo, andiamo ad abilitare il componente aggiuntivo "SB 1232 AQ" andando nella sezione "Dispositivi & Reti" del proprramma.
All'interno di "Vista dispositivi" possiamo inserire il nostro modulino andando a prelevarlo dal catalogo.
Come si può notare dall'immagine sottostante all interno della cartella "Signal board" troviamo il nostro componente "6ES7 232-4HA30-0XB0", il quale comparirà automaticamente nelle configurazioni.
Per evitare interferenze o letture errate, è consigliato collegare la massa dell'ingresso e dell'uscita in comune con quella del segnale da prelevare e con quella del circuito del segnale in uscita.
E' sconsigliato collegarle con la massa comune del plc.

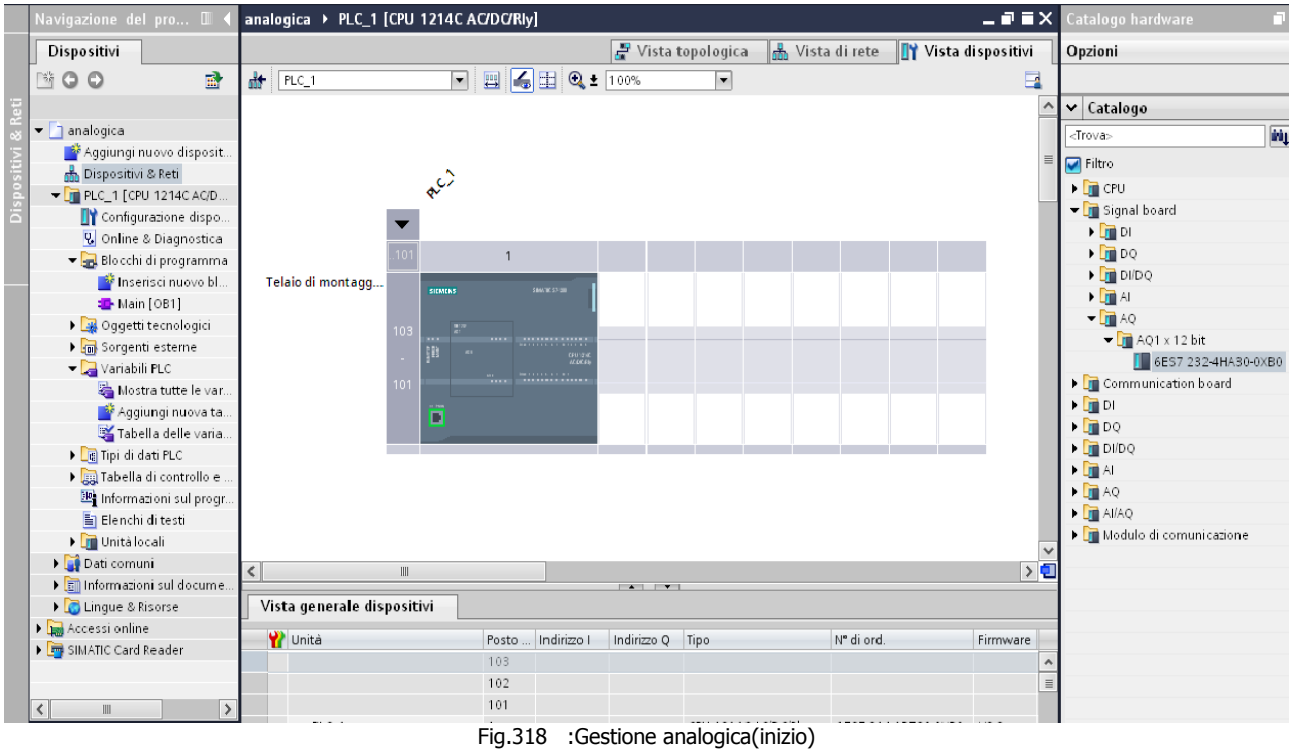


Fig.318 :Gestione analogica(inizio)

Gli ingressi analogici hanno un range di lavoro in tensione 0-10 V con una risoluzione di 10 bit.
L'uscita analogica, fornita dal modulino, ha un range di lavoro ±10 V e 0→20 mA, con una risoluzione di 12 bit per la tensione e 11 bit per la corrente.

Il PLC legge l'ingresso analogico e lo salva in un'area di memoria facendo automaticamente la conversione, viceversa, per l'uscita, leggerà un'area di memoria per poi produrre un segnale analogico.

Dentro le proprietà del nostro PLC, sotto la voce "AI2" troviamo le impostazioni degli ingressi analogici.
Qui possiamo vedere l'indirizzo nel quale verrà salvata la conversione.

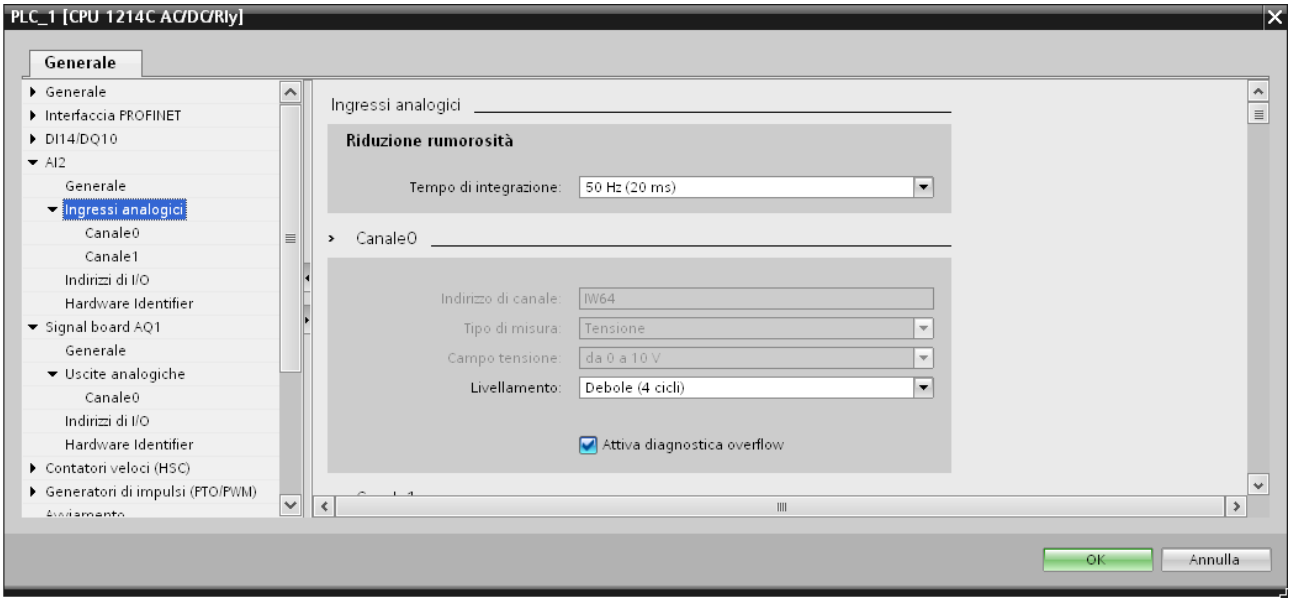


Fig.319 :Impostazioni ingressi analogici

Questo è l'indirizzo identificativo dell' hardware che si occupa dell'acquisizione analogica.

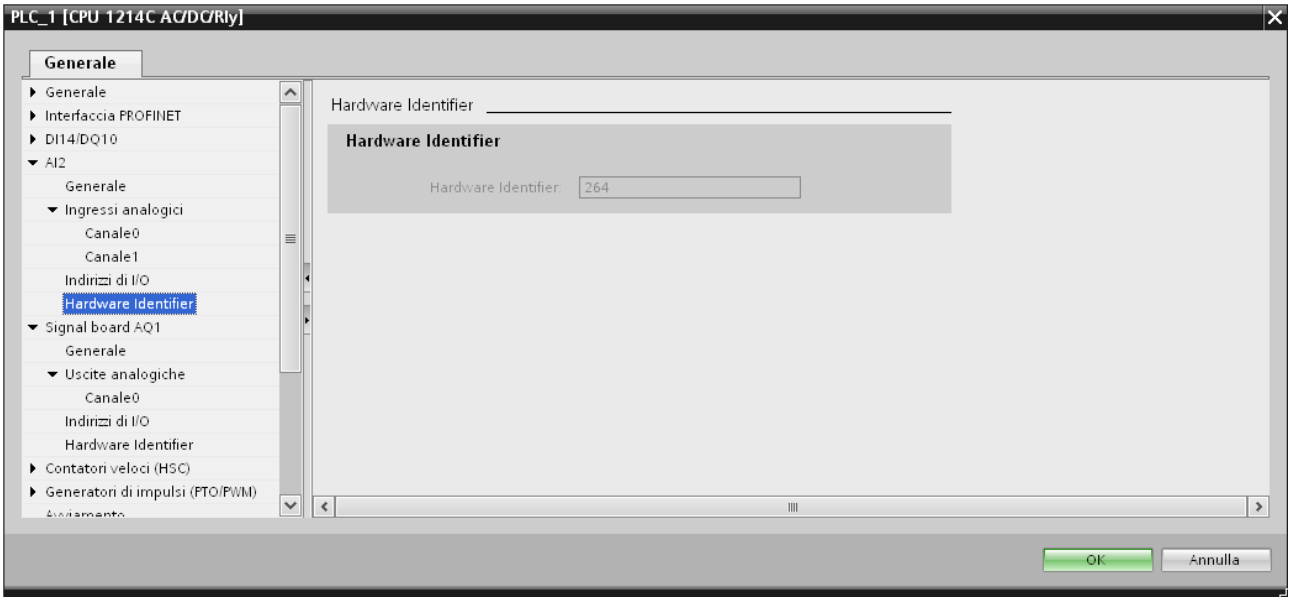


Fig.320 :Identificativo Hardware per l'acquisizione analogica

Da questa finestra possiamo scegliere: il valore da dare in uscita in caso di STOP, se uscire in tensione o in corrente e l'indirizzo in cui salvare il valore.

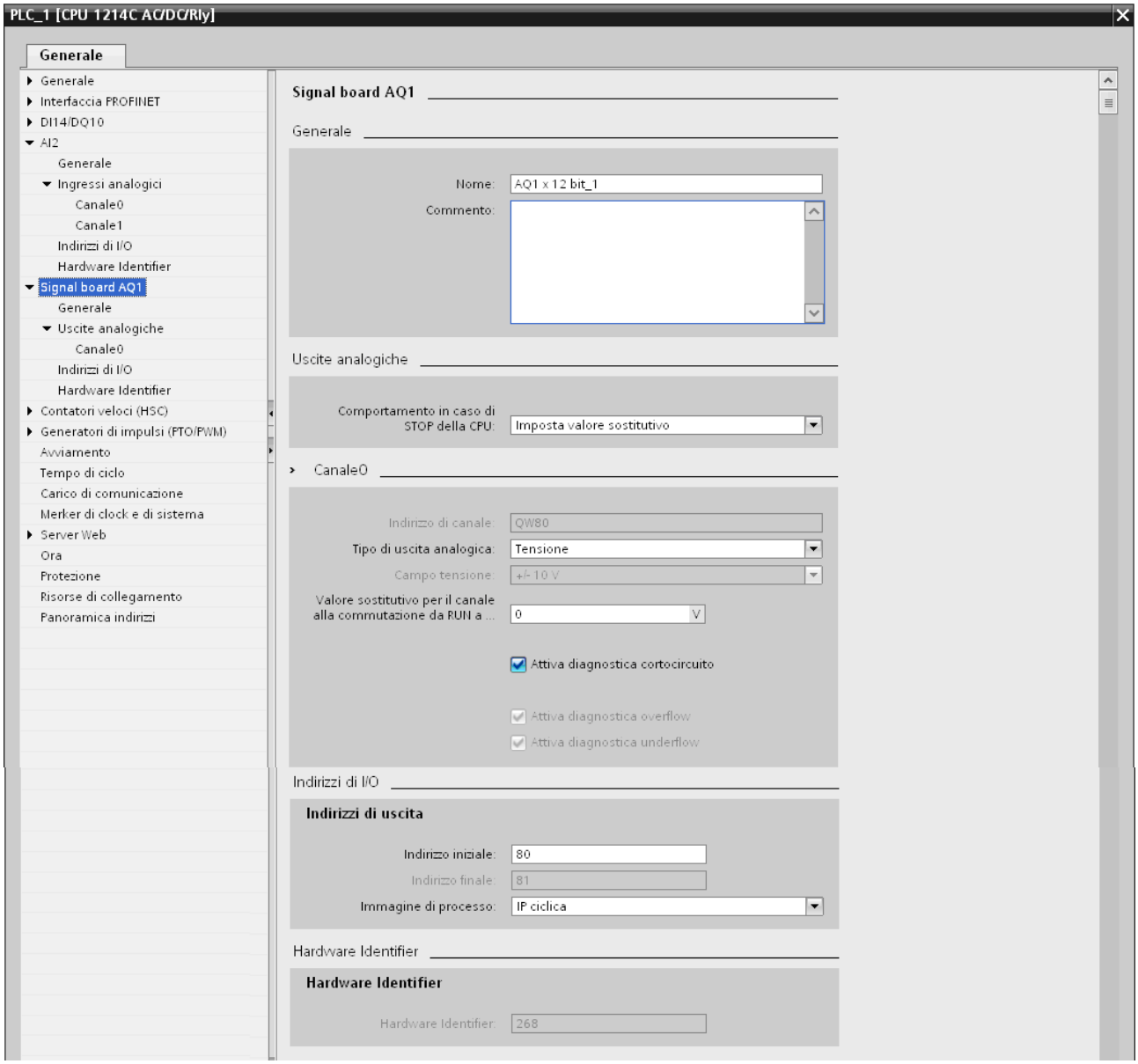


Fig.321 :Signal Board

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Preleviamo il valore digitale dall'area di memoria in cui è stata salvata la conversione e lo spostiamo nell'area di memoria da cui verrà prelevato per essere convertito in un segnale analogico.

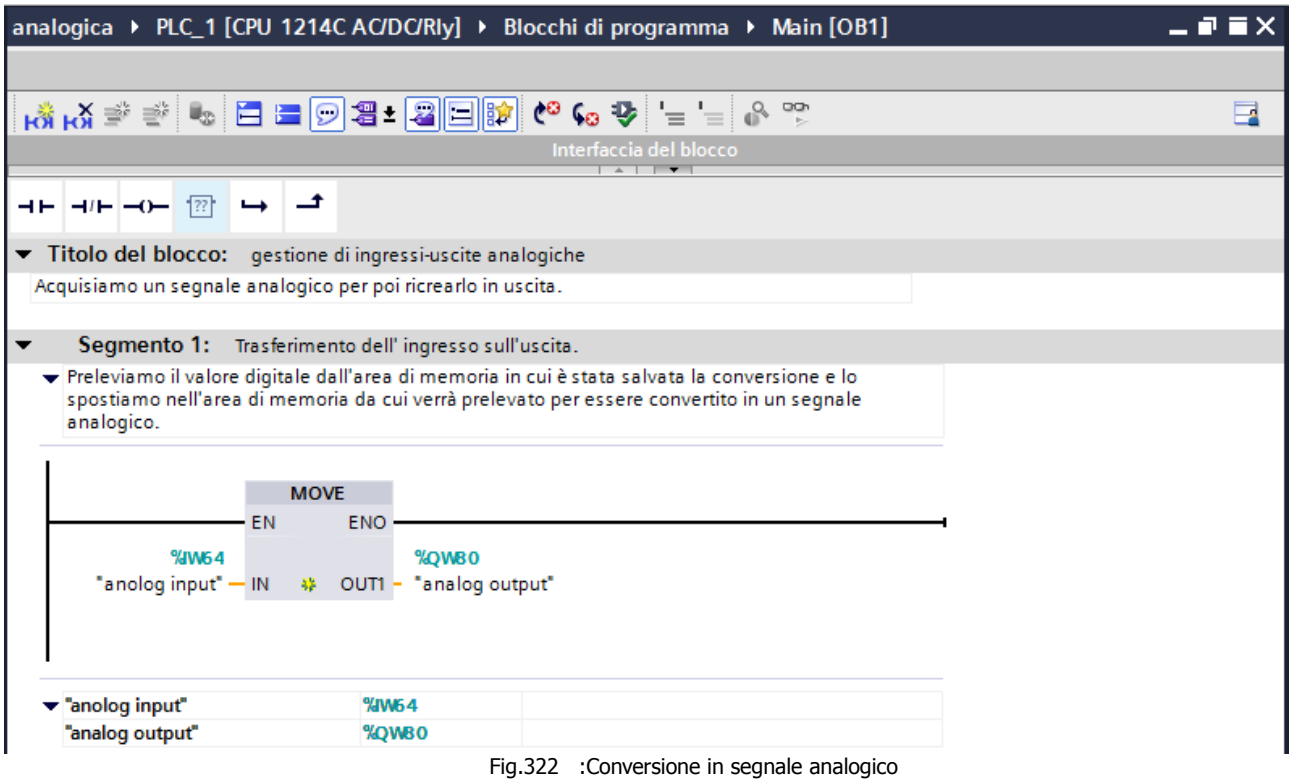


Fig.322 :Conversione in segnale analogico

Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

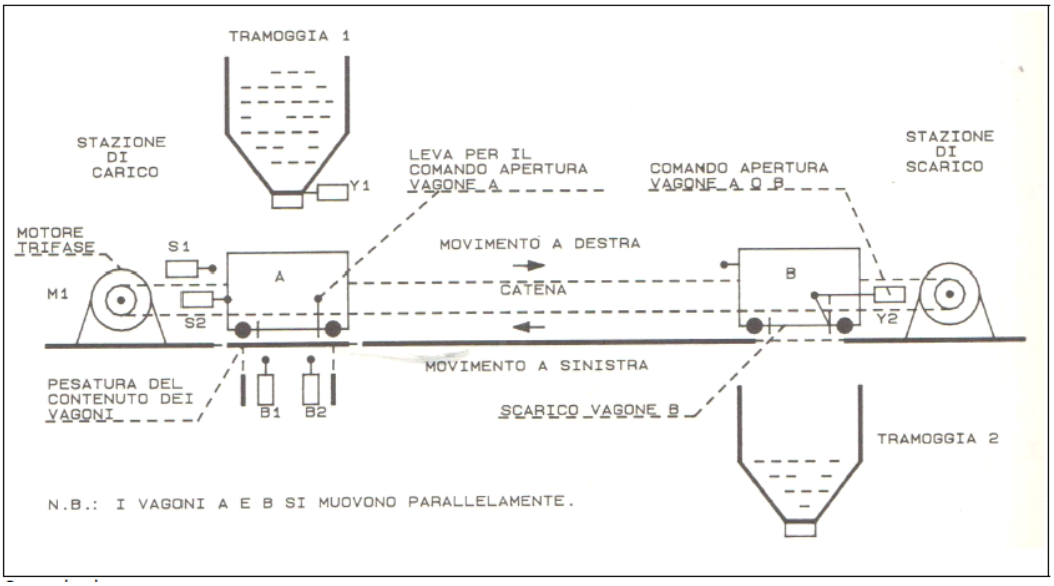
Gestione Tramoggia

esperienza 75A

Benini Andrea
DellaChiesa Enrico
5AET 2013
(andrea.benini01@gmail.com)
(enrico.dellachiesa@gmail.com)

TESTO ESERCIZIO:

L'impianto deve realizzare l'automazione per il carico di un vagone nella stazione di carico e il contemporaneo scarico dell'altro nella stazione di scarico.
Due elettrovalvole, Y1 e Y2, consentono il caricamento di uno con il contenuto della tramoggia 1 e lo scarico dell'altro nella tramoggia 2.
I due vagoni si muovono parallelamente su due binari, mediante una catena azionata da un motore elettrico trifase M1.
Nella stazione di carico vengono previsti dei finecorsa, S1 e S2, che rilevano rispettivamente la presenza del vagone B o del vagone A nella posizione di carico; la quantità di materiale da caricare viene definito dal dinamometro B1 per il vagone A e dal dinamometro B2 per il vagone B.
Quando i vagoni sono stati caricati della quantità dovuta, il cilindro pneumatico azionato dall'elettrovalvola Y1 chiude lo scarico e la fase di caricamento s'interrompe.
Durante la fase di carico (esempio vagone A), l'altro vagone (vagone B) si deve scaricare nella stazione di scarico nella tramoggia 2; ciò avviene mediante un cilindro pneumatico azionato dall'elettrovalvola Y2, che agisce sul sistema di scarico del vagone.
Dopo tale fase temporizzata (il tempo deve essere diverso per i due vagoni) , l'elettrovalvola Y2 interrompe il processo di scarico.
Il motore M1 allora sposta verso la stazione di scarico il vagone A e il vagone B verso la stazione di carico.
Il ciclo a questo punto si ripete, sino a quando non viene premuto il pulsante di ALT che deve concludere il ciclo riportando il vagone A nella stazione di carico.
Si prevede inoltre un pulsante di arresto di emergenza che blocca il ciclo in qualsiasi momento.



Sigla	Dispositivo
Y1	elettrovalvola ON/OFF che regola il caricamento del vagone e che apre/chiude lo scarico della TRAMOGGIA 1
Y2	elettrovalvola ON/OFF che regola e si usa per lo scarico del vagone in TRAMOGGIA 2
B1	dinamometro (ON/OFF) che definisce la fine carica del VAGONE A
B2	dinamometro (ON/OFF) che definisce la fine carica del VAGONE B
S1	finecorsa di presenza vagone B in posizione di carico
S2	finecorsa di presenza vagone A in posizione di carico
M1	motore che muove una catena dedicata allo spostamento dei due vagoni
P_ALT	Pulsante di arresto
P_ EMERG	Pulsante di emergenza
RESET_ALRM	Pulsante di arresto allarme

FUNZIONAMENTO :

Inizialmente bisogna accertarsi della posizione dei vagoni A e B.
Se la posizione non risulta corretta avremo lo stato di allarme mentre se è corretta, ovvero il vagone A è in posizione di carica (finecorsa S2 attivo) e il vagone B è in posizione di scarica, inizia il ciclo di lavorazione.
Dato che il finecorsa S2 è attivo, il vagone A si trova sotto la tramoggia 1 e inizia la fase di carica mediante l'apertura della valvola Y1. Contemporaneamente il vagone B, trovandosi in posizione di scarica e quindi sulla tramoggia 2, mediante la valvola Y2 viene scaricato.
Le elettrovalvole Y1 e Y2 rimangono attive per un determinato tempo, diverso nei due casi: il tempo di carica risulterà maggiore rispetto quello di scarica, in quanto non abbiamo sensori che indicano lo stato del vagone in fase di scarica.
I dinamometri B1 e B2 determinano la fine della fase di carica di entrambi i vagoni: il dinamometro B1 riguarda il peso del vagone A mentre il dinamometro B2 riguarda il peso del vagone B.
Una volta che il dinamometro ha misurato il peso corretto del vagone, l'elettrovalvola Y1 viene disattivata e a sua volta anche l'elettrovalvola Y2.
A questo punto viene attivato il motore M1 che effettua lo scambio di posizione dei due vagoni tramite una catena. Ricordiamo che i due vagoni si muovono in modo alternato e parallelo.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Il ciclo di lavorazione continua a ripetersi. Se tali condizioni non si verificano, oppure si presentano altri problemi all'interno del processo di lavorazione, scattano l'allarme e la sirena: bisogna prevedere un pulsante di emergenza con lampada che attivi anche una sirena. La lampada deve lampeggiare con un Ton=200mS Toff=200mS. La sirena e' ON/OFF e si attiva per conto suo.
Per disattivare il sistema di allarme abbiamo un sistema con due pulsanti a quadro, distanti fra loro 1,5m che premuti contemporaneamente.

Grafico di rappresentazione dei cicli di comando :

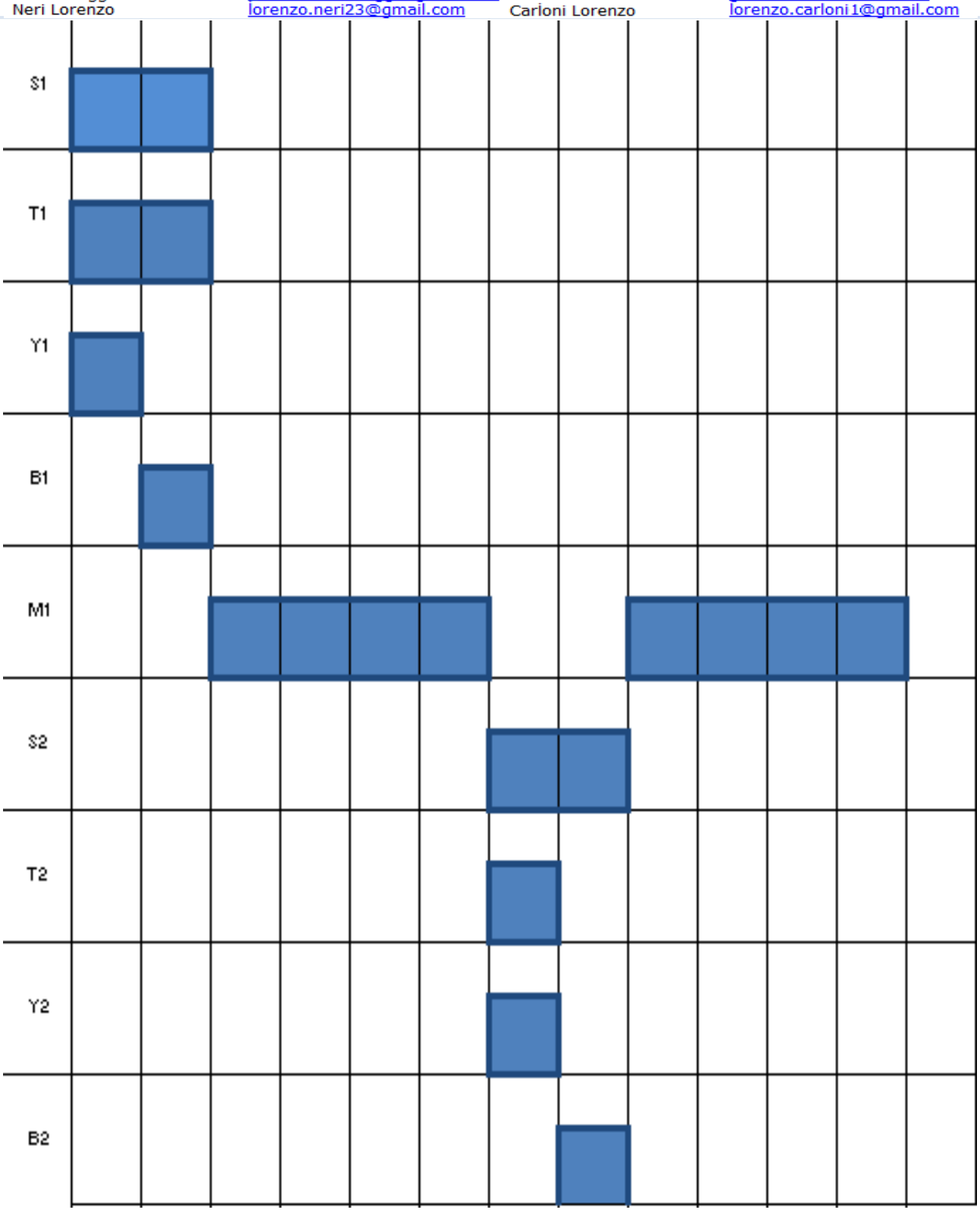
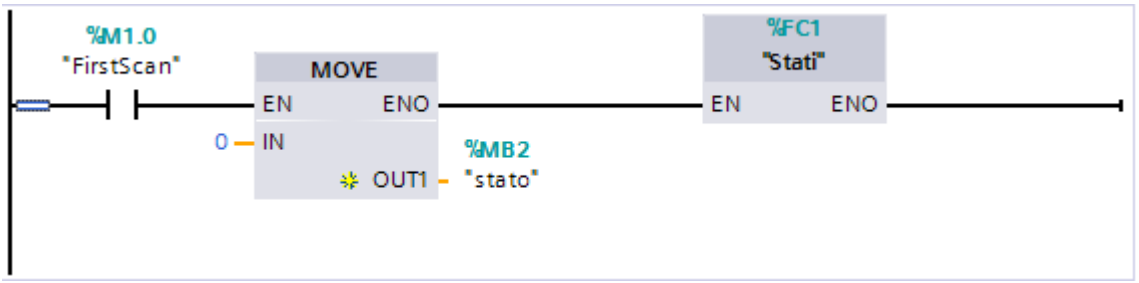


Fig. 324 :Cicli di comando gestione tramoggia

Ovviamente il pulsante di ALT non viene utilizzato in questa tempistica perché è la tempistica corretta. Se si verificassero condizioni diverse a queste, scatterebbe l'allarme e la sirena e in quel caso si userebbe il pulsante di ALT.

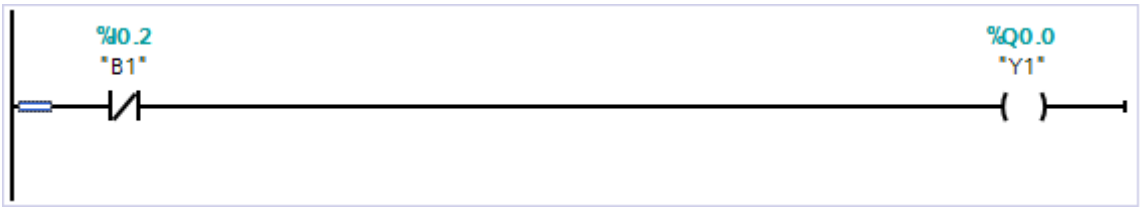
Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

MAIN



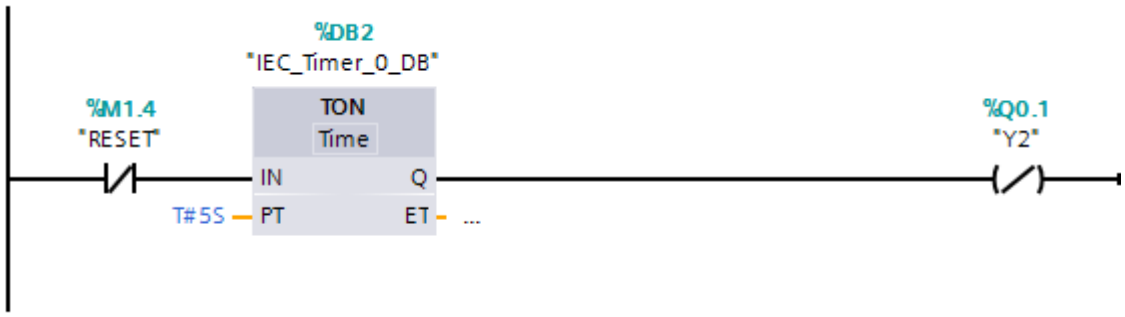
Stato S0

Carico vagone A

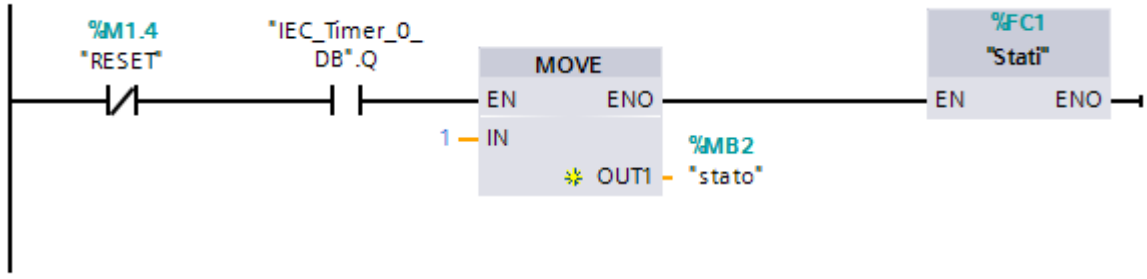


Scarico vagone B

5s è il tempo di scarica > del tempo di carica.

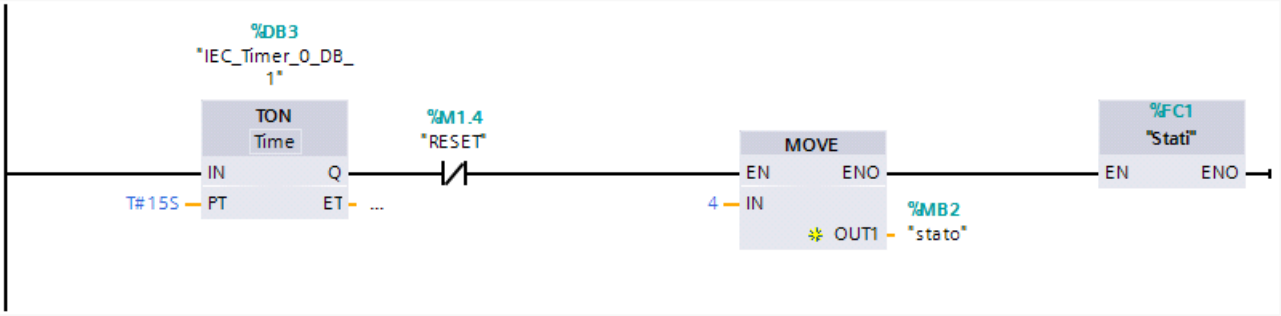


Passaggio di stato

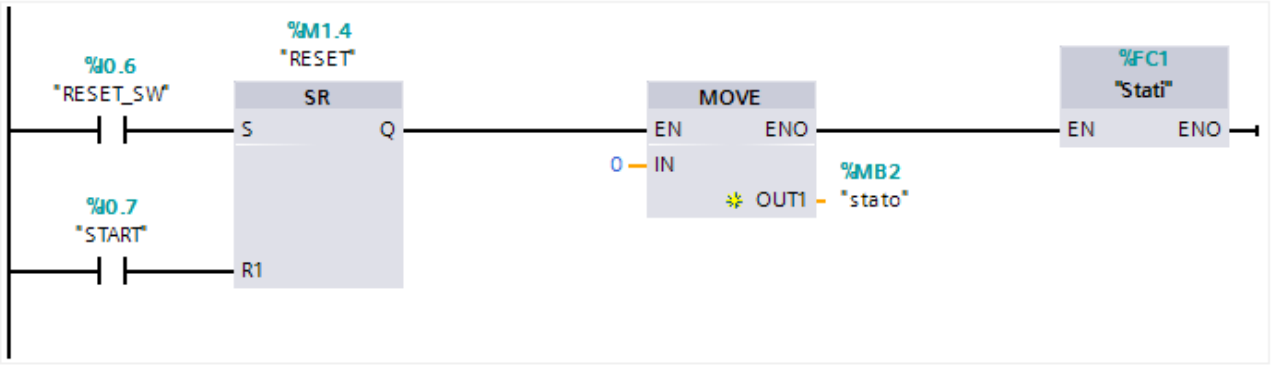


Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Allarme?
Se resto 15s in questo stato scatta l'allarme.

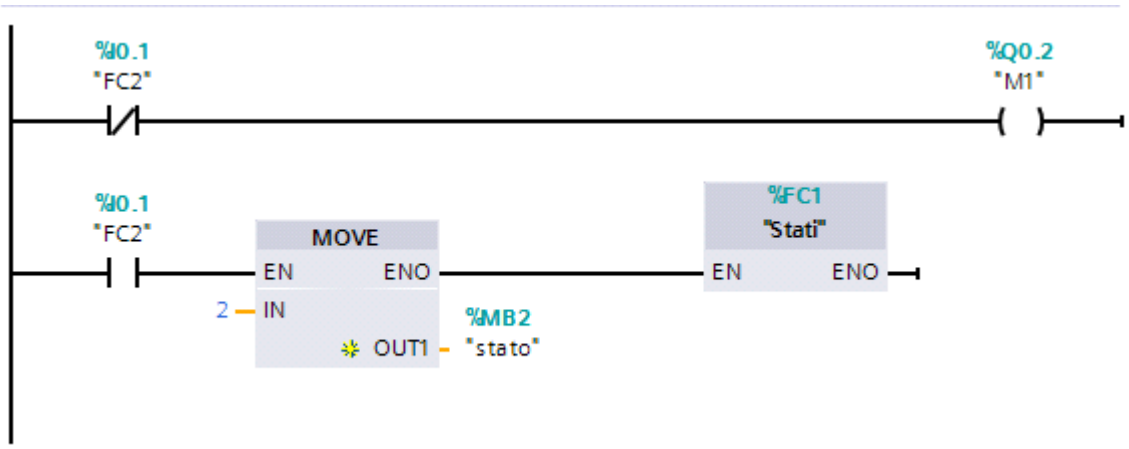


Reset
Se RESET viene premuto il programma si deve fermare nello stato 0.

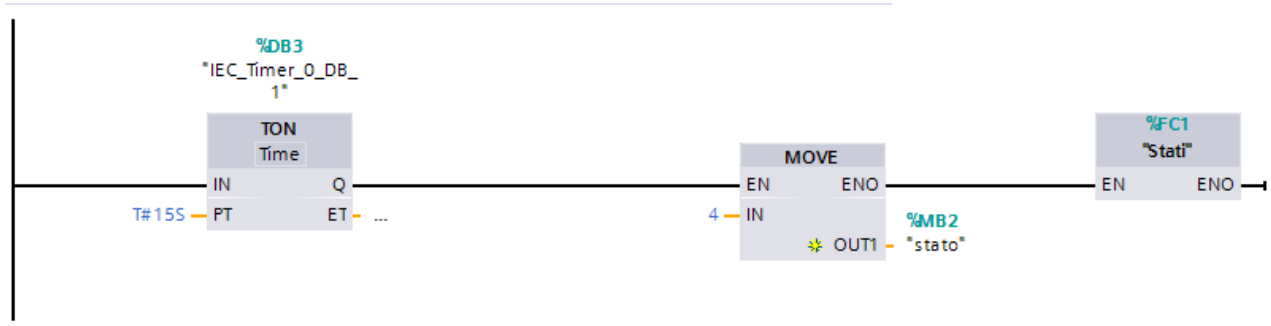


Stato S1

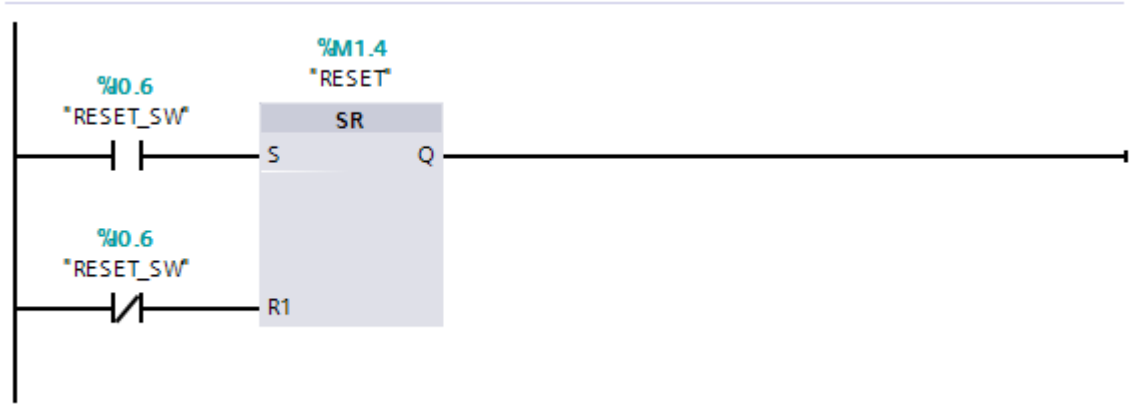
Il vagone B si muove verso la tramoggia di carico fino a quando non incontra il finecorsa.



ALLARME?

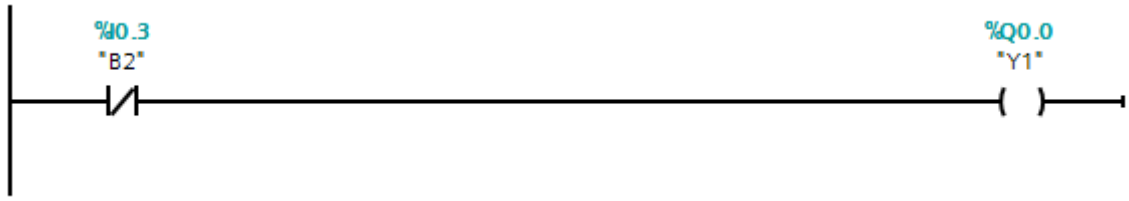


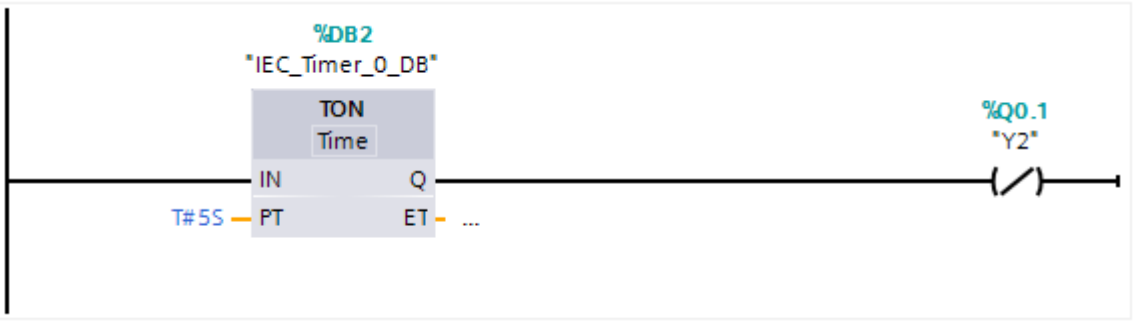
Il RESET può scattare in ogni stato. Quando scatta il programma gira fino a tornare allo stato 0, poi si ferma.

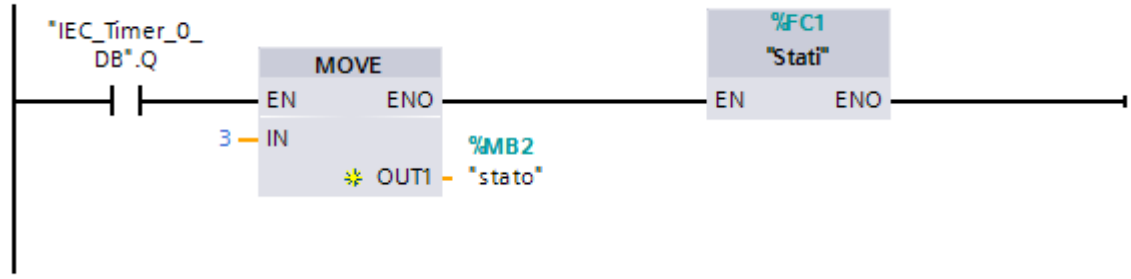


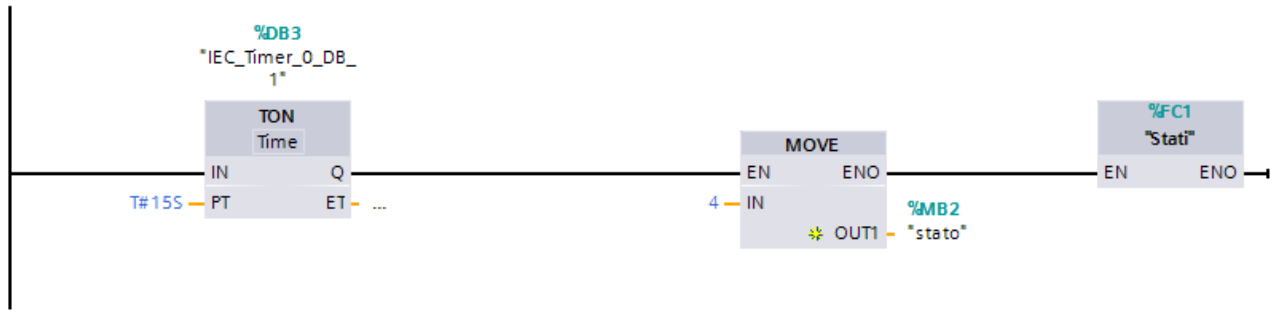
Stato S2

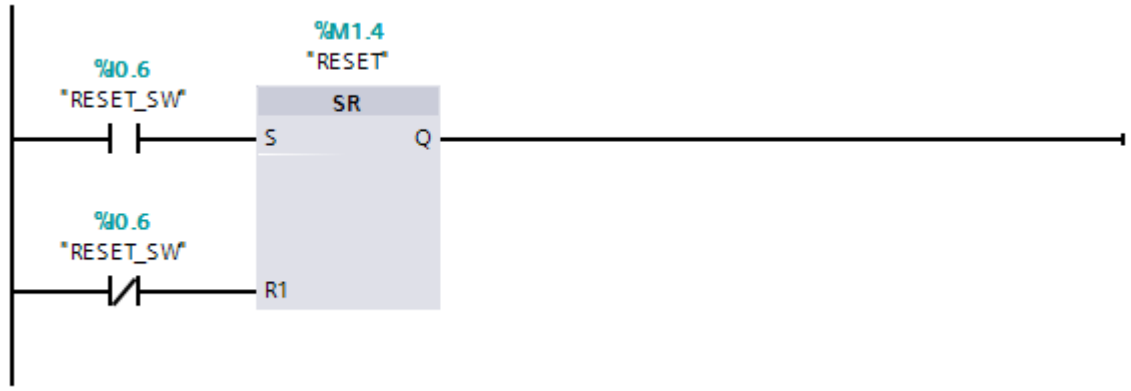
CARICO VAGONE B E SCARICO VAGONE A
È S0, cambia solo il dinamometro B2 al posto del B1.





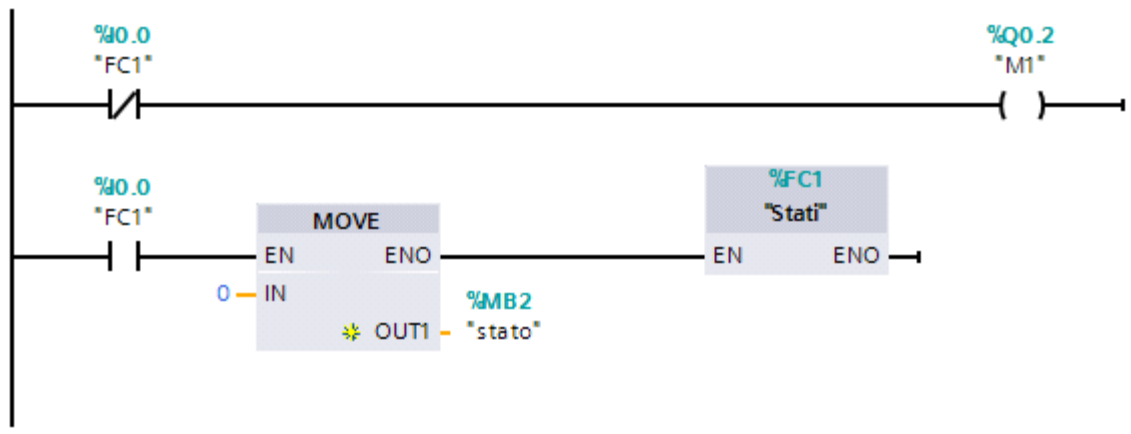




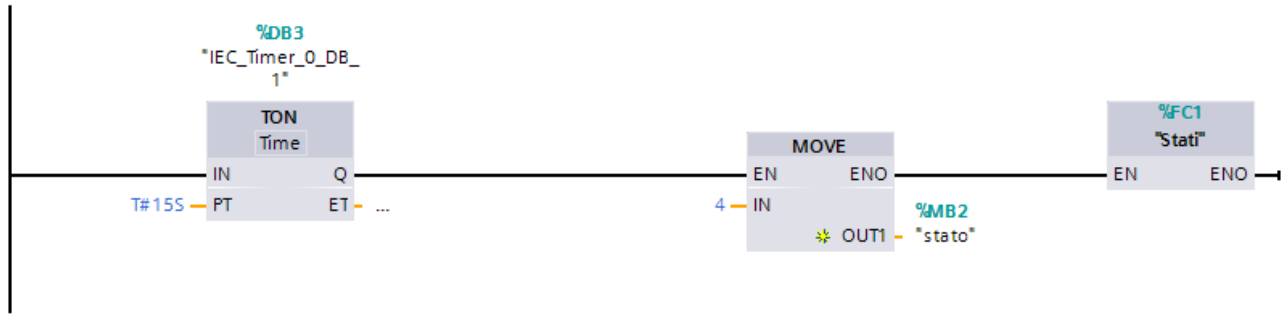


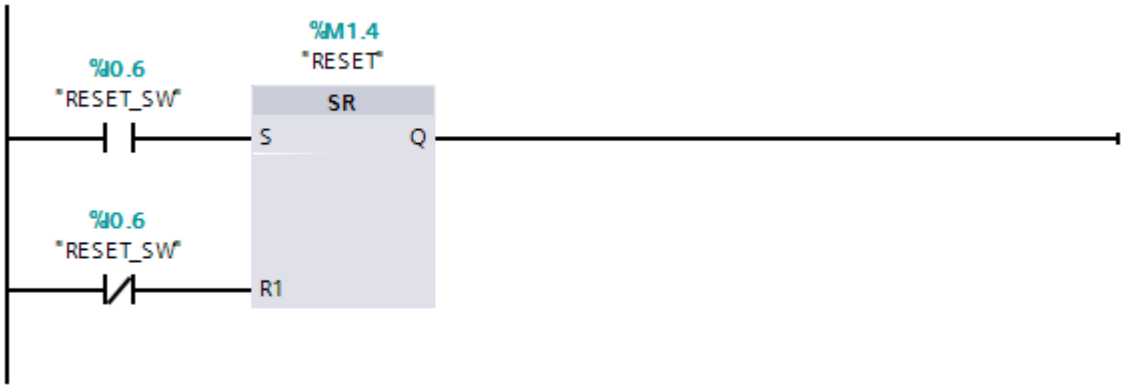
Stato S3

MOVIMENTO DEI VAGONI CHE TORNANO A S0



Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

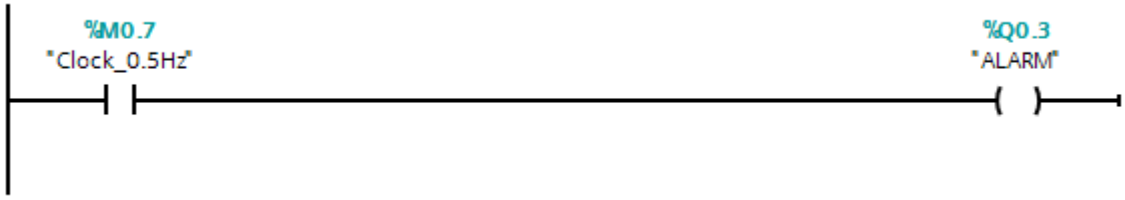




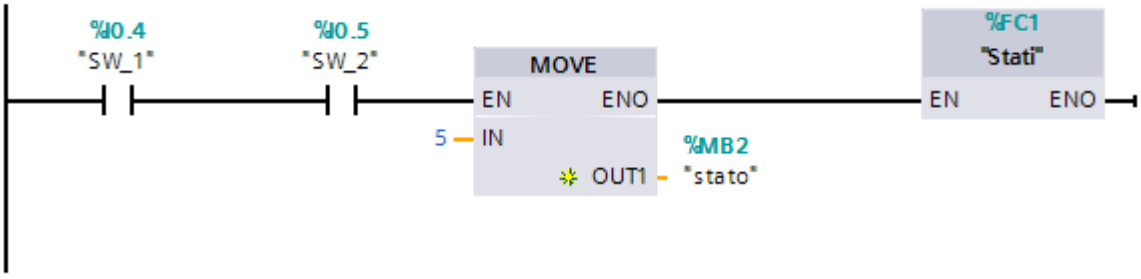
Stato S4

ALLARME

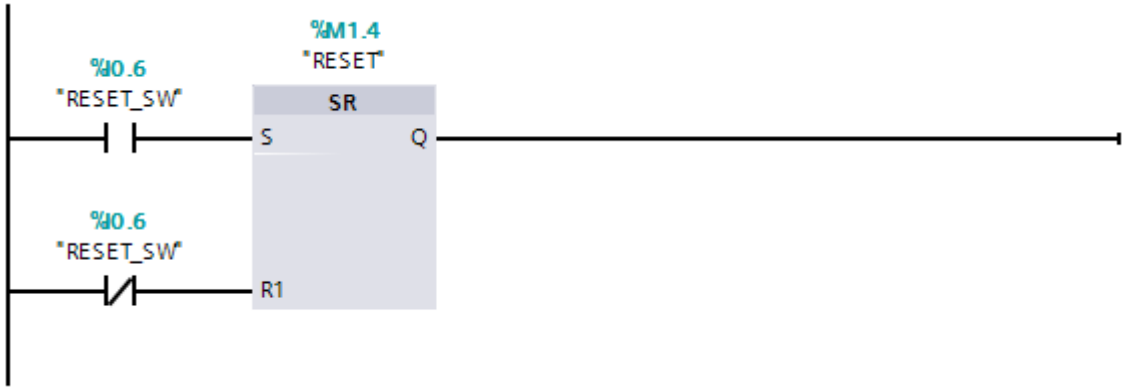
LUCE LAMPEGGIANTE a 1S ON e 1S OFF



SE PREMO CONTEMPORANEAMENTE SW_1 SW_2 termino l'allarme.



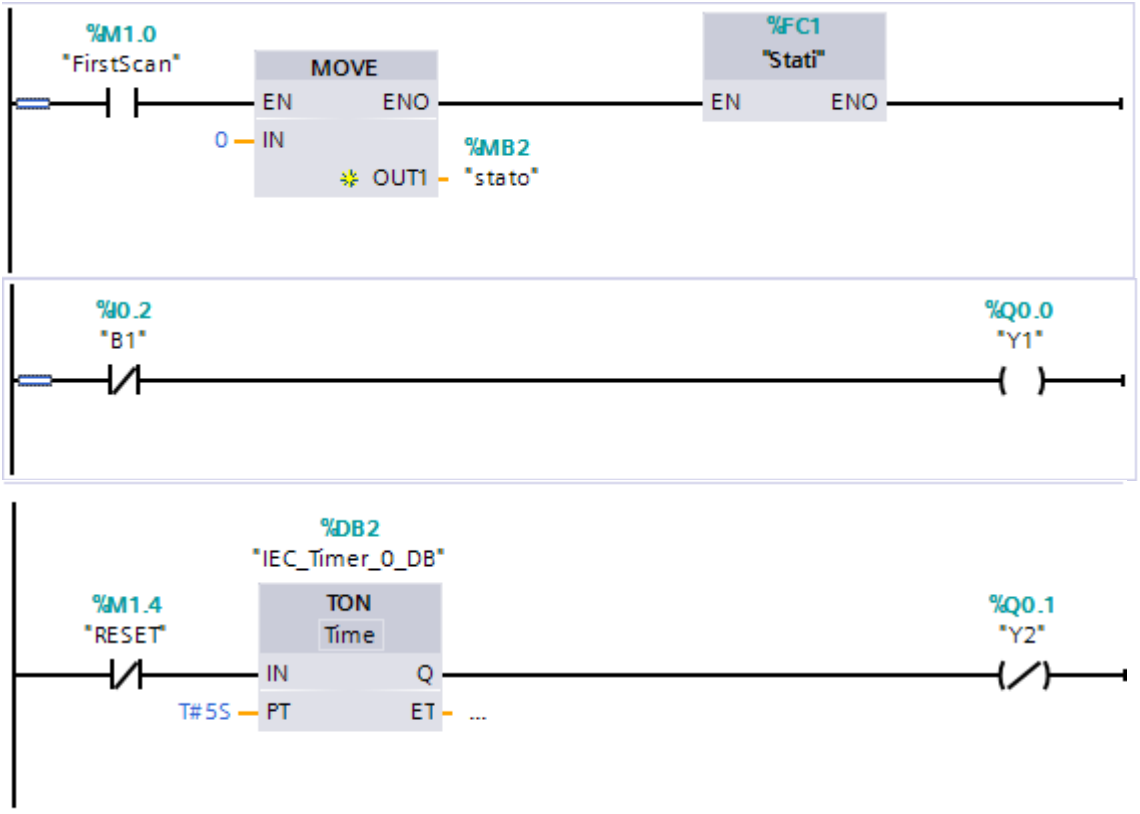
Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com



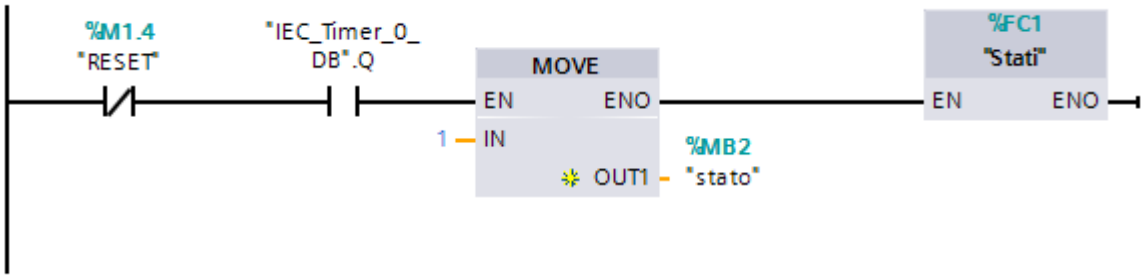
MOVIMENTO DEI VAGONI CHE TORNANO A S0

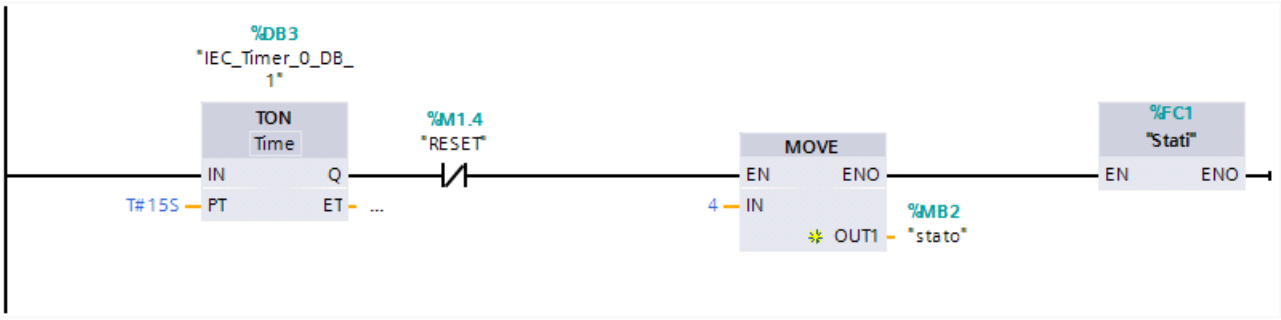
ALLARME
LUCE LAMPEGGIANTE a 1S ON e 1S OFF
SE PREMO CONTEMPORANEAMENTE SW_1 SW_2 termino l'allarme.
Vado in stato 5, uno stato di manutenzione.

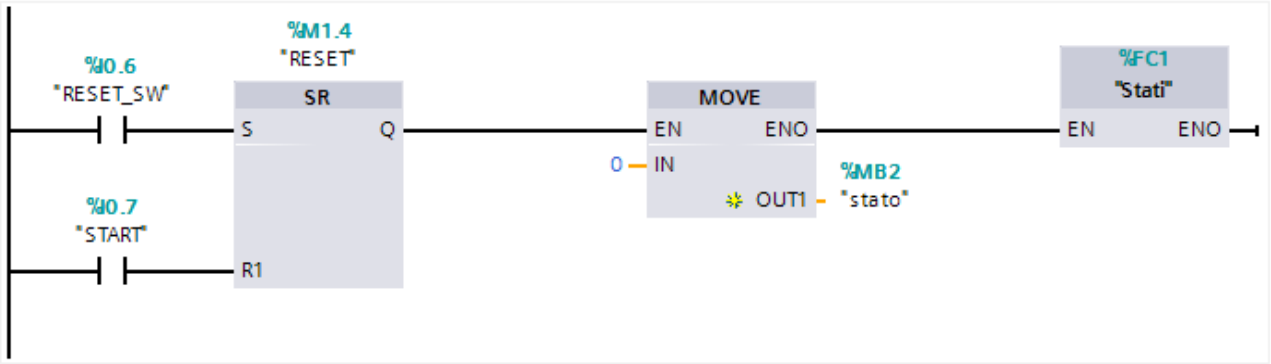
Stato di manutenzione: quando premo i due pulsanti contemporaneamente vado in RESET.

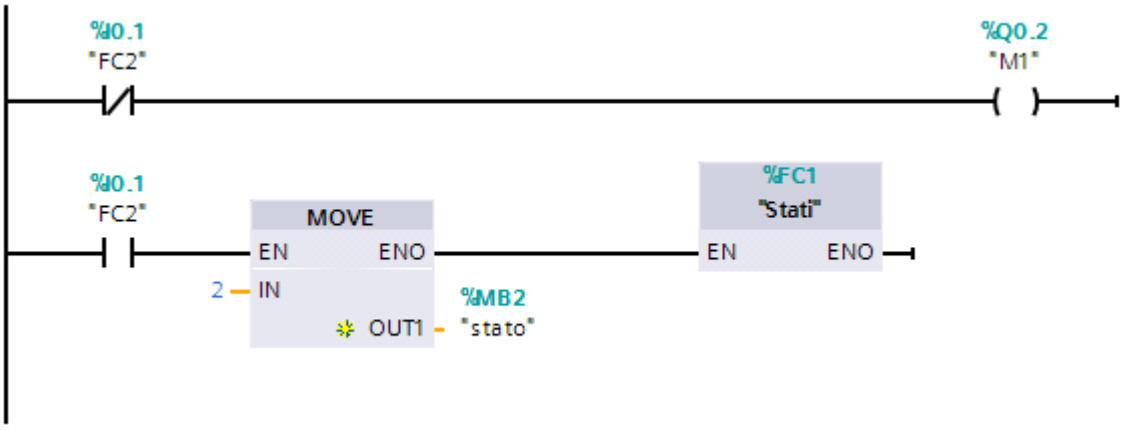


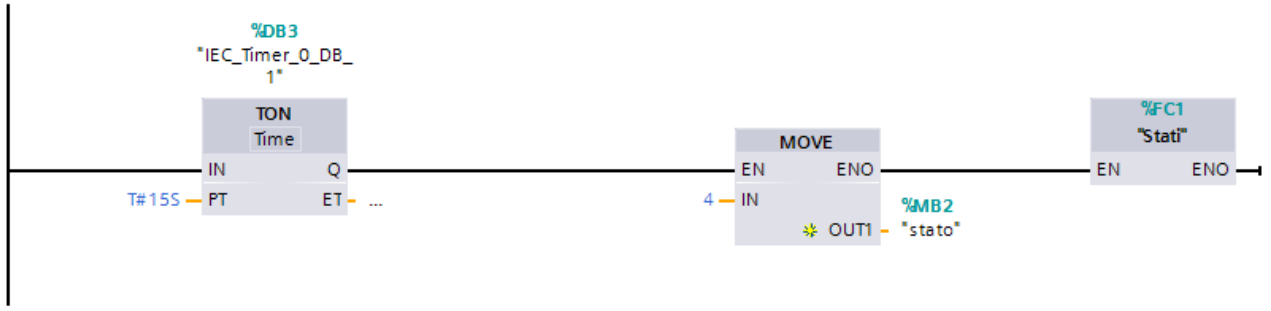
Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com



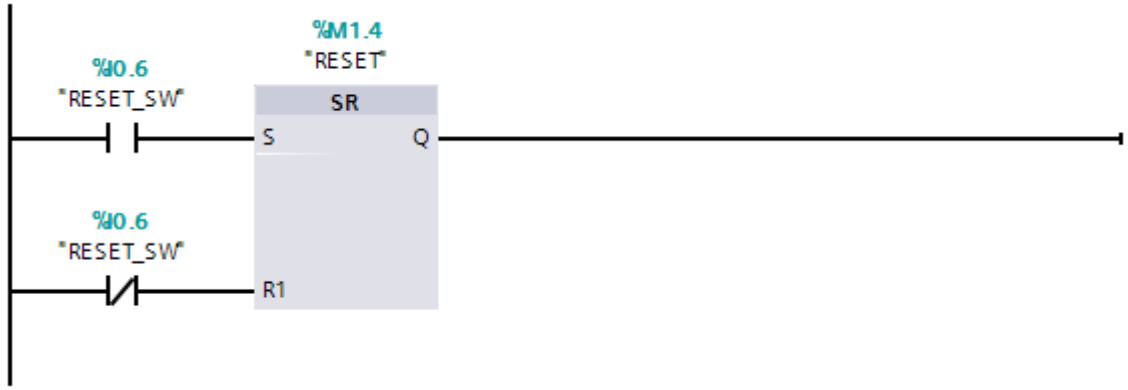


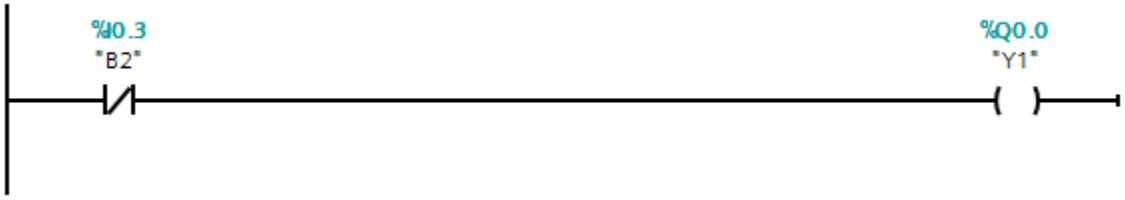


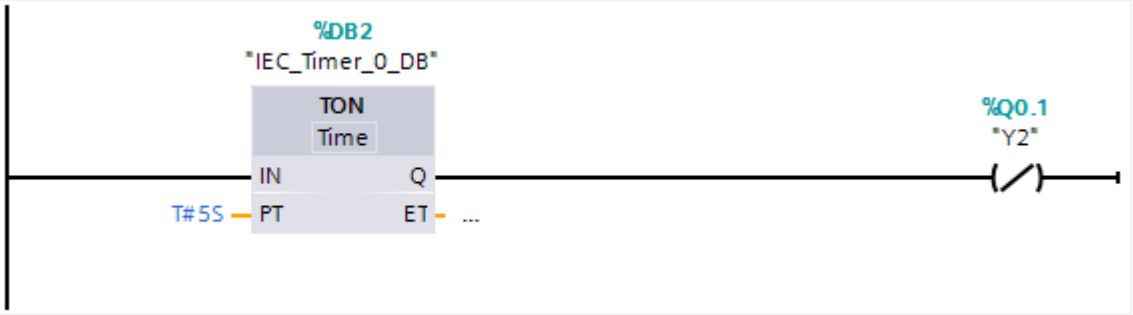


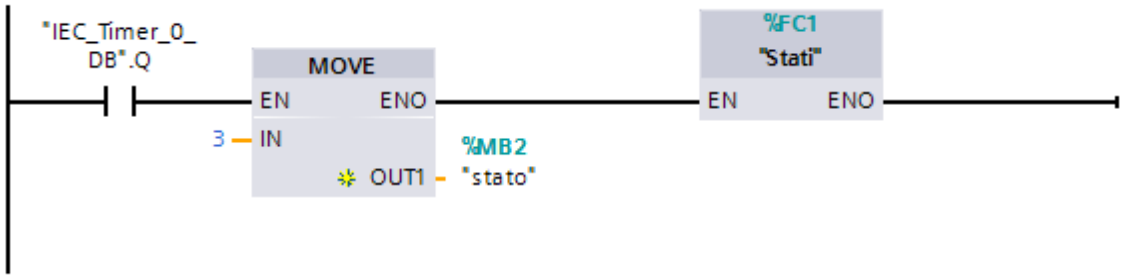


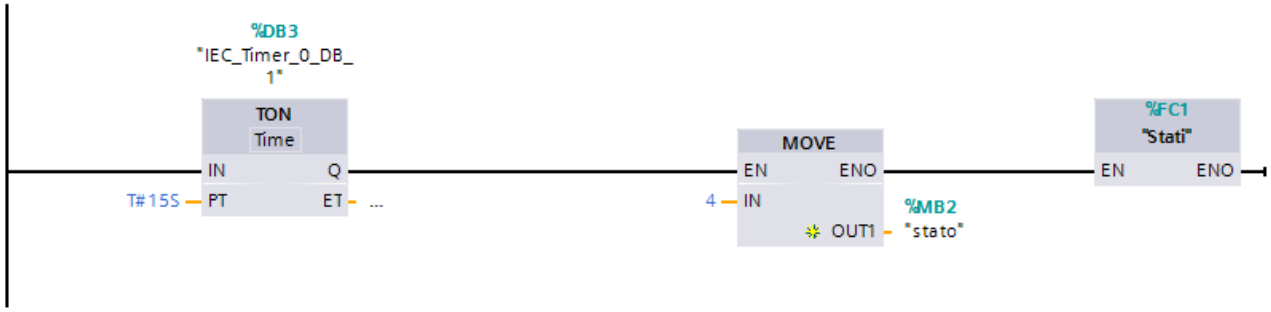
Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com



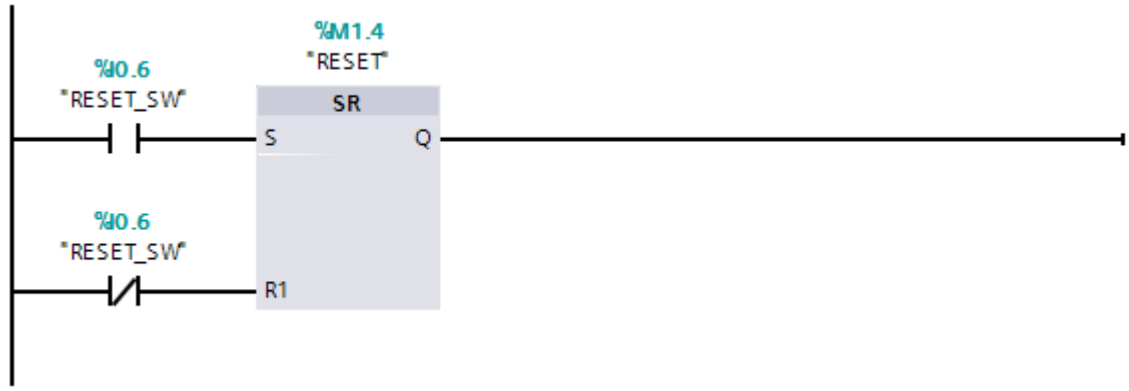


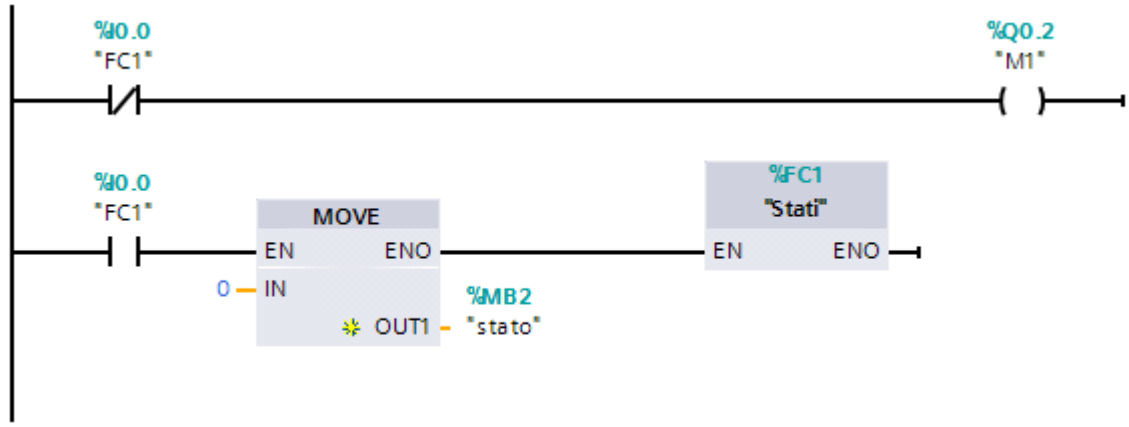


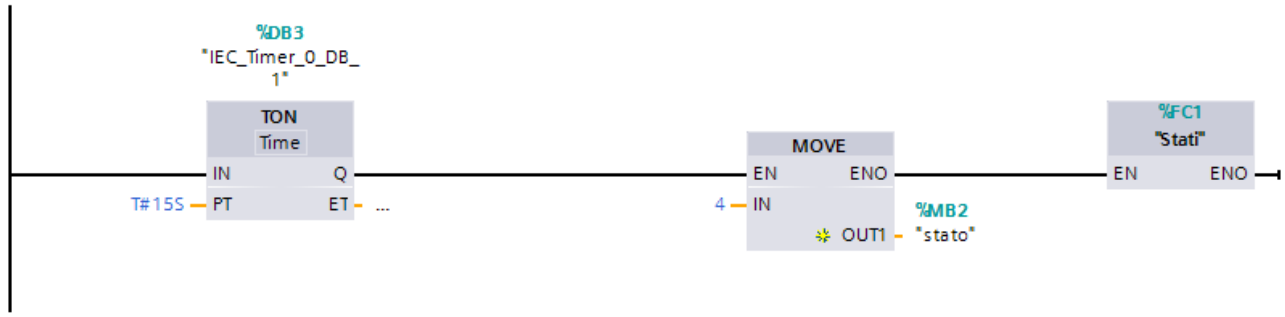


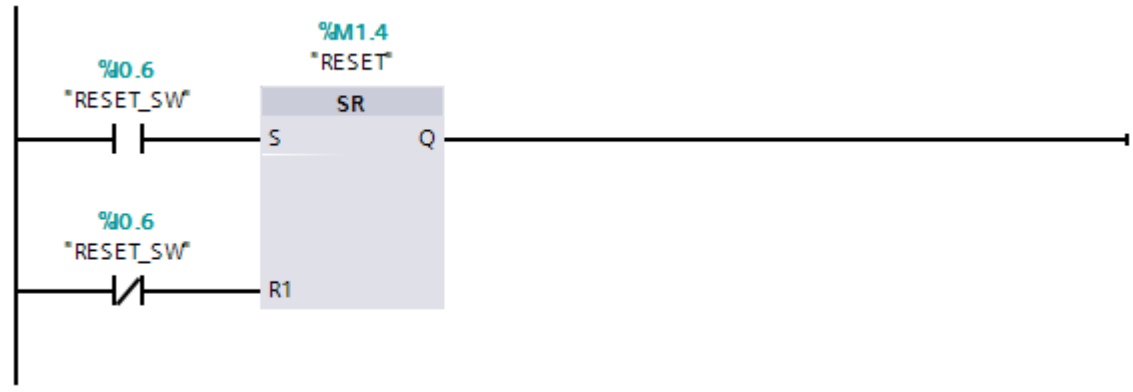


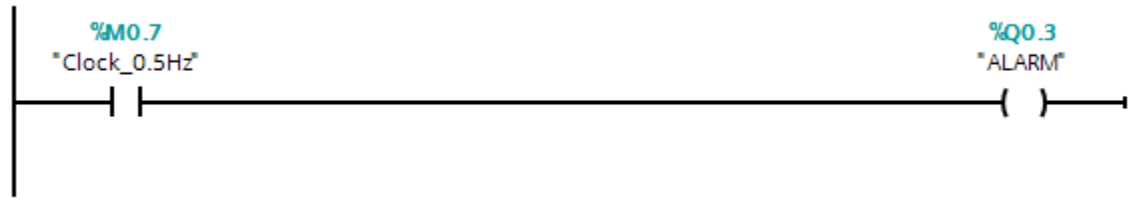
Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com











Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

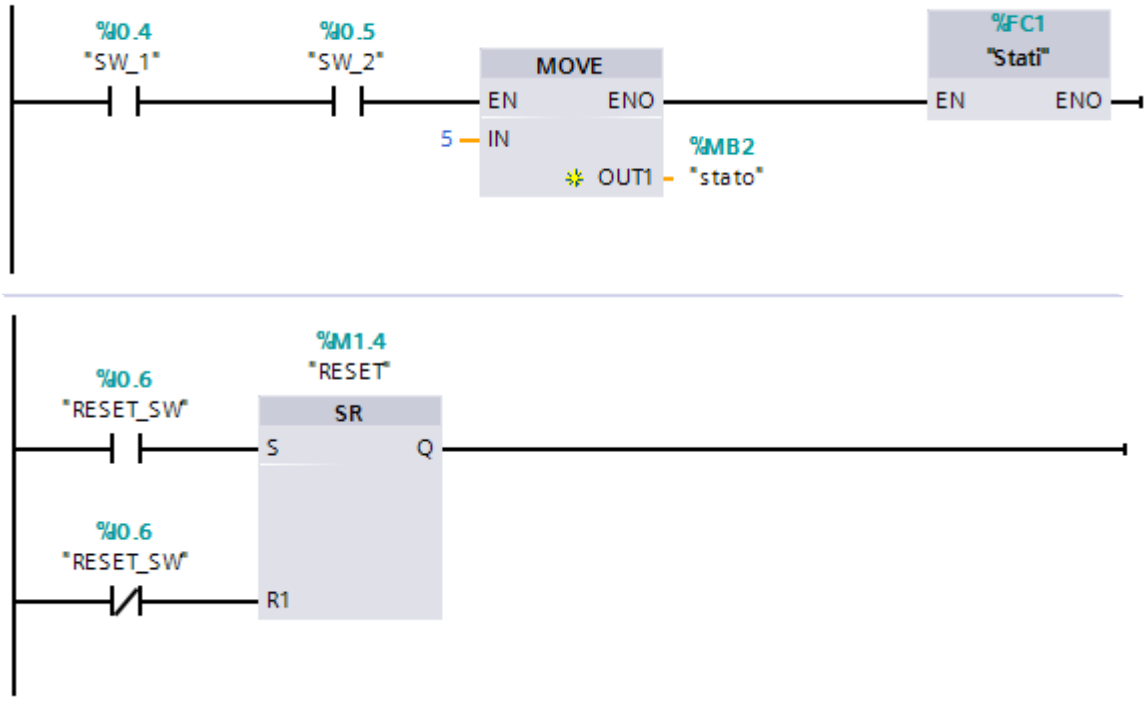


Fig.325 :Programma LADDER per gestione tramoggia

ITIS "B. Pascal - P.le Macrelli, 100 - 47023 Cesena (FC) - tel. + 39 0547 22792 - fax +39 0547 24141
TDP 5^A ELETTRONICA TELECOMUNICAZIONI Prof, FERRARO GIUSEPPE Ass. Tecnico MARCO FERRETTI
Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Gestione tempistica generale di un counter

esperienza 75

Montemaggi Pablo 5AET 2013
(pablo.montemaggi@gmail.com)

TESTO ESERCIZIO:

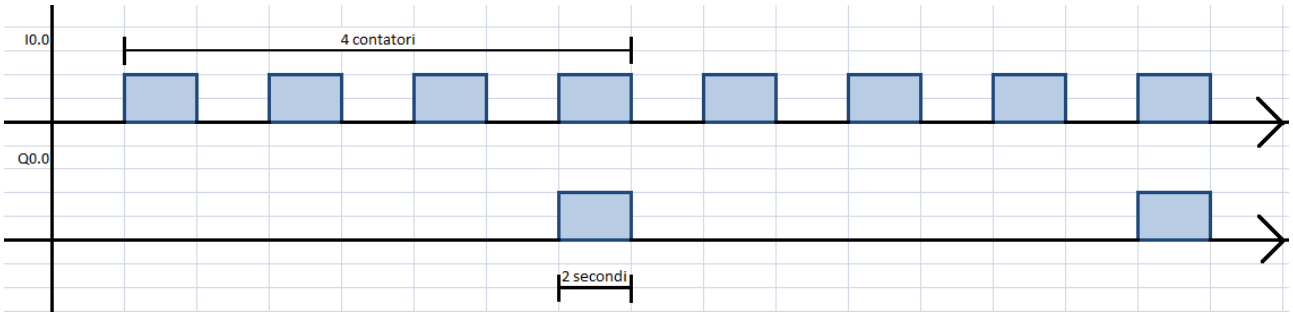
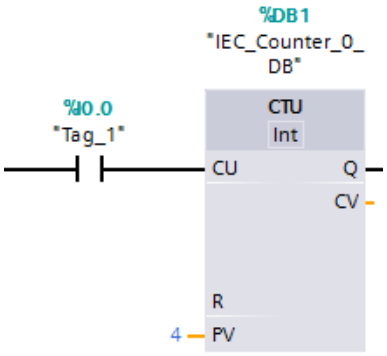


Fig.326 :Tempistica

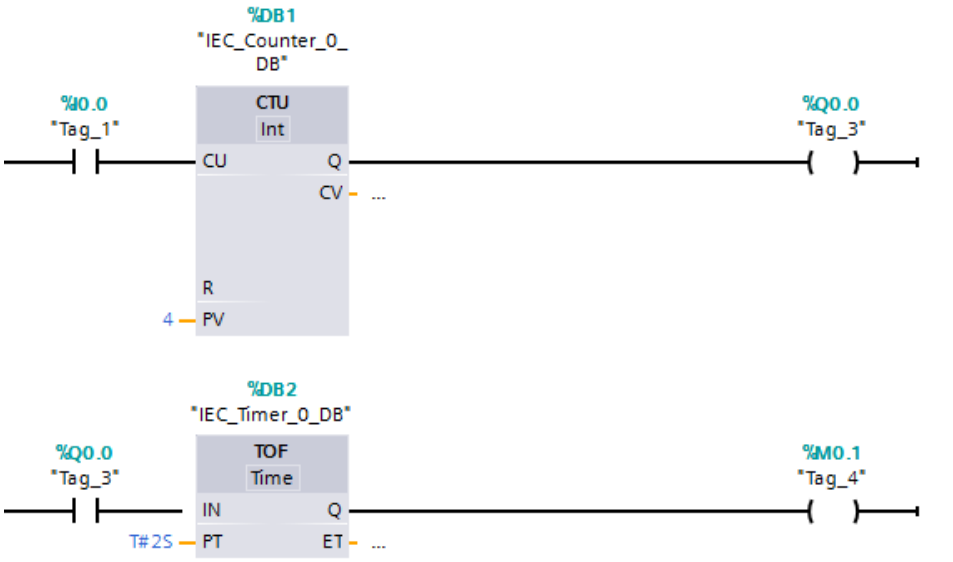
Al 4° impulso parte un timer che dura 2 secondi poi disattiva la porta Q0.0. Eventuali impulsi di conteggio mentre Q0.0 è uguale a 1 non vanno presi in considerazione. Deve essere possibile la ripetibilità dell'esercizio.

RISOLUZIONE ESERCIZIO: L'ingresso I0.0 è l'ingresso che ci permette di incrementare il valore letto dal contatore.



Quando il numero degli impulsi arriva a 4 il COUNTER attiva l'uscita Q0.0 la quale, immediatamente, attiva il timer TOF.

Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com



Questo, dopo due secondi, mette a 1 il merker, il quale andrà a resettare il counter per poter ripetere il procedimento.

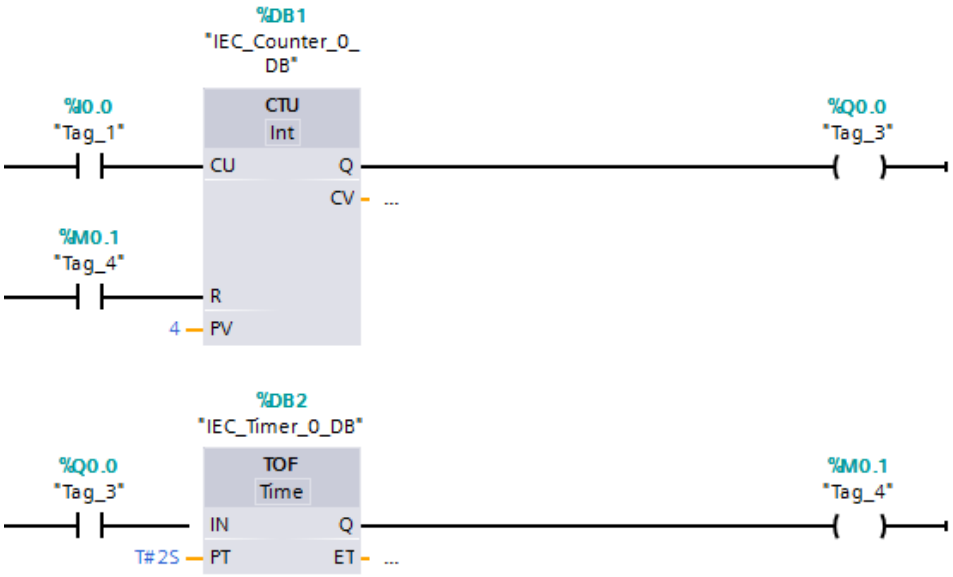


Fig.327 :Risoluzione esercizio

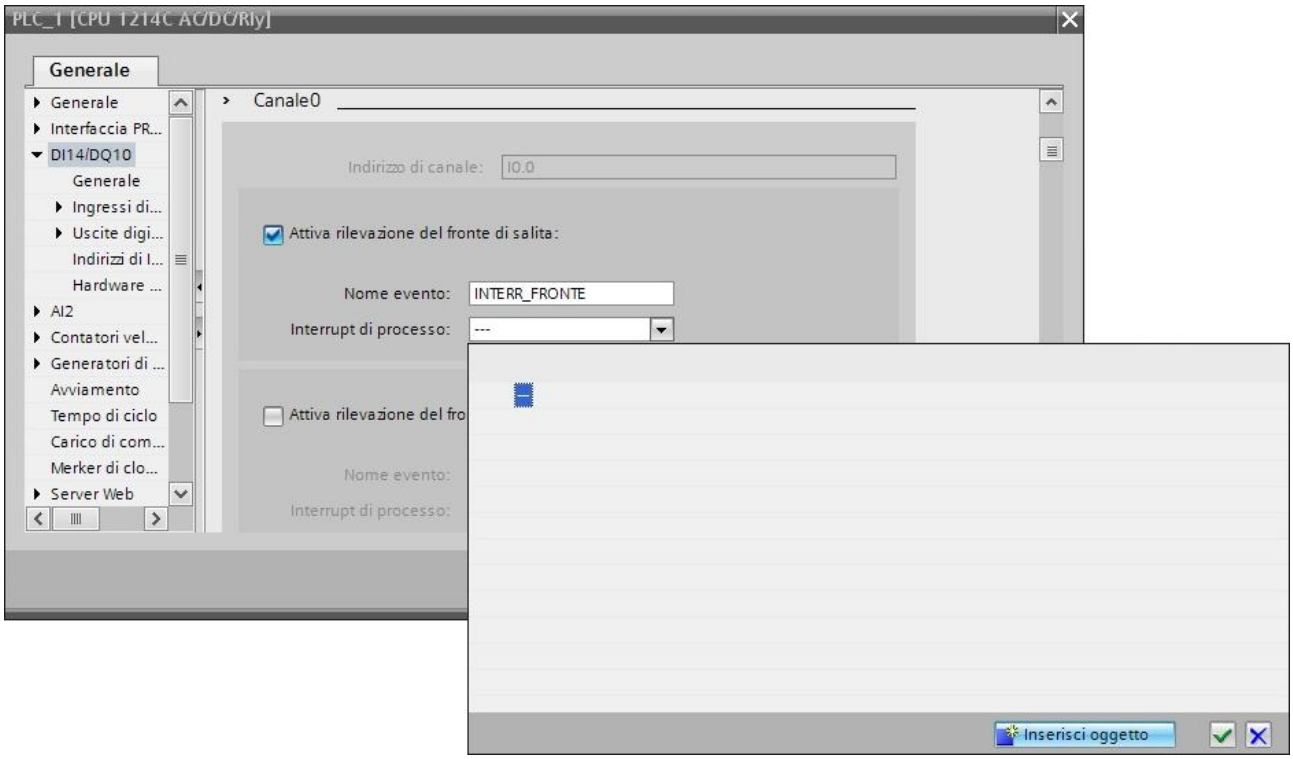
Farneti Alessandro farneti.alessandro@gmail.com Della Chiesa Enrico enrico.dellachiesa@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com Gridella Andrea gridella.andrea@libero.it
Neri Lorenzo lorenzo.neri23@gmail.com Carloni Lorenzo lorenzo.carloni1@gmail.com

Gestione Interrupt Hardware

esperienza 76

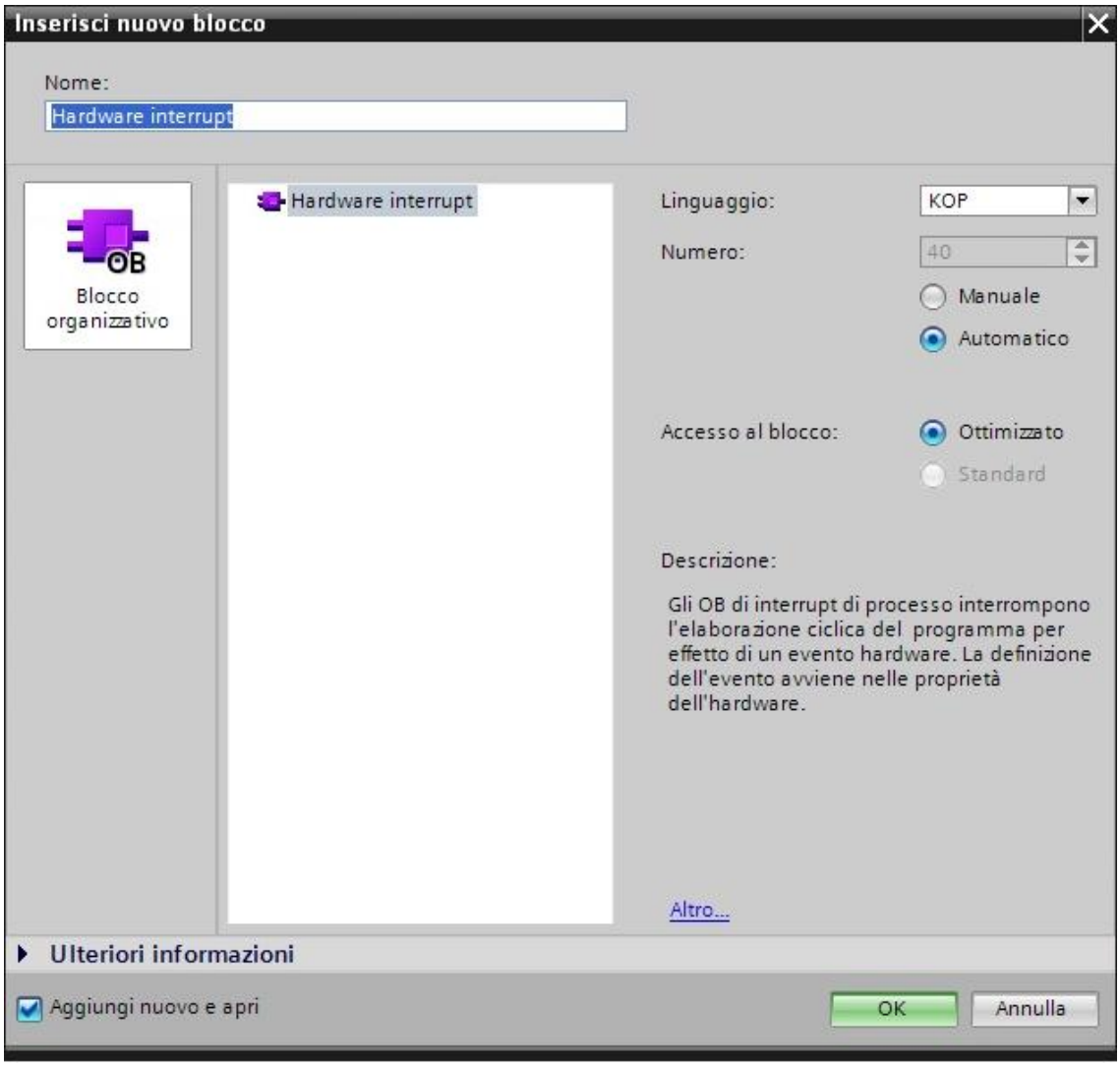
Montemaggi Pablo 5AET 2013
Carloni Lorenzo 5AET 2013
(pablo.montemaggi@gmail.com)
(lorenzo.carloni1@gmail.com)

E' possibile programmare un blocco di interrupt il quale viene eseguito come conseguenza ad uno stimolo hardware.
Andiamo ora in proprietà, sotto la voce relativa all'ingresso I0.0 e attiviamo la rilevazione del fronte di salita.

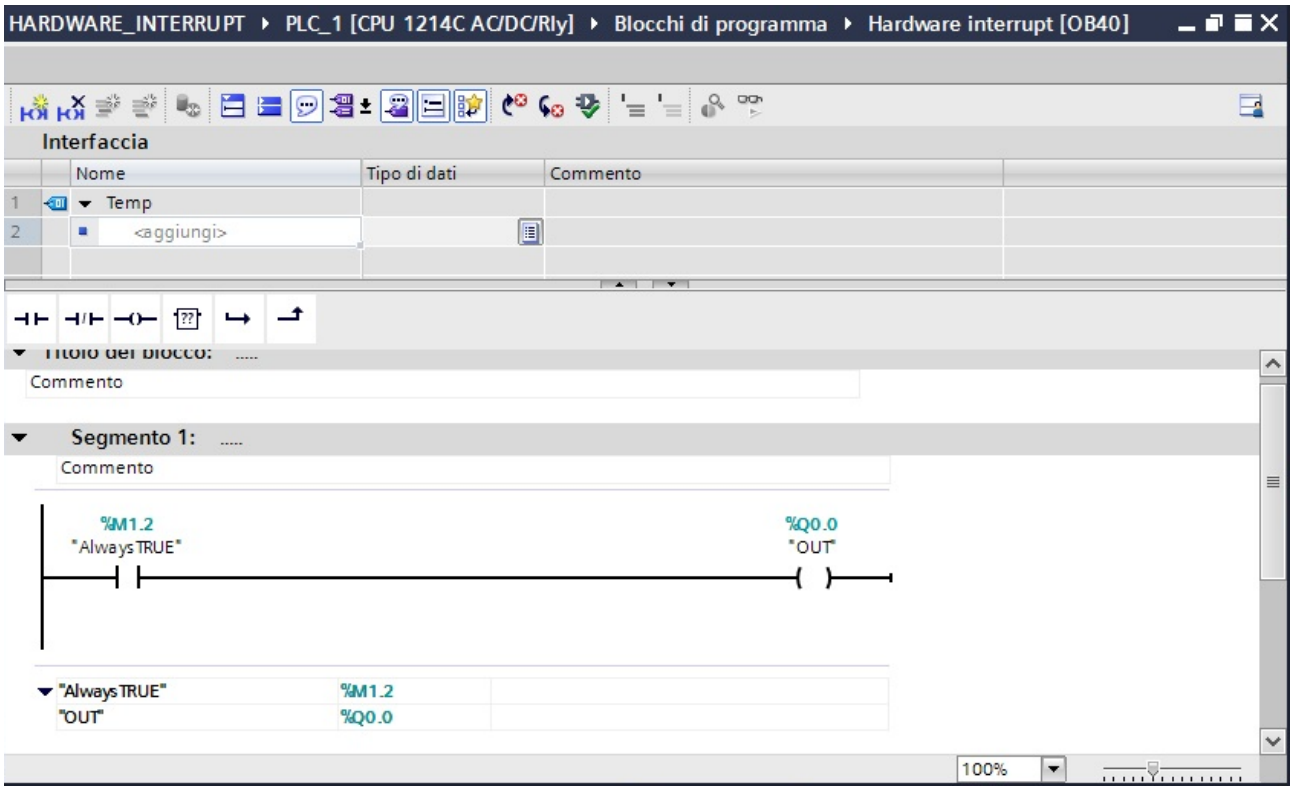


Farneti Alessandro farneti.alessandro@gmail.com
Montemaggi Pablo pablo.montemaggi@gmail.com
Neri Lorenzo lorenzo.neri23@gmail.com
Della Chiesa Enrico enrico.dellachiesa@gmail.com
Gridella Andrea gridella.andrea@libero.it
Carloni Lorenzo lorenzo.carloni1@gmail.com

Cliccando su inserisci oggetto, creiamo il blocco di interrupt il quale verrà eseguito come conseguenza all'evento. Dando la conferma si aprirà automaticamente.



All'interno del blocco di interrupt andremo ad accendere un'uscita.
Questo blocco verrà eseguito solamente durante l'interrupt.



Bibliografia

IL PLC - DALLA TEORIA AL LABORATORIO GIOVANNI PIRRAGLIA YOUCANPRINT EDITORE

SCHEMARIO DI IMPIANTI ELETTRICI CIVILI INDUSTRIALI PNEUMATICA, ELETTROPLNEUMATICA, PLC - SECONDA EDIZIONE - M.BAREZZI EDITORE CUSL "A.RUBLEV"

Ringraziamenti

Ing. FERRARO GIUSEPPE – Docente Scolastico
Ing. BAMBINI – Righi Elettroservizi
MARCO FERRETTI – Assistente tecnico
MIRIA PRACUCCI – Ufficio Tecnico
ANTONELLA – Ufficio Tecnico
Dott. SERGI CARMELO per l'acquisto dei PLC SIEMENS
Dott. POSTIGLIONE FRANCESCO - Dirigente scolastico attuale

Ing. VARGIU Giampiero - SIEMENS
ALOI Bruno - SIEMENS EDUCATION