# Diagrammi LADDER

Il LADDER è il linguaggio di programmazione (detto anche linguaggio a contatti) più diffuso per la programmazione dei PLC, i controllori a logica programmibile. Esso include nel suo formalismo una serie di istruzioni di base davvero semplici (quando la General Motors avanzò le specifiche di progetto per il linguaggio di programmazione dei PLC di quell'epoca si pensò a qualcosa di semplice e assai vicino alla preparazione informatica dei tecnici, quasi scarsa). Le istruzioni di base prevedono: il contatto normalmente aperto e quello normalmente chiuso, la bobina, timer e contatori. Tanto bastava per programmare i vecchi controllori a relè. In questo documento non mi occuperò delle istruzioni di base del LADDER, una descrizione completa può essere trovata su molti libri di testo che trattano di automazione. L'IEC (il comitato elettronico internazionale) formalizza esplicitamente nel documento 1131 i linguaggi da usare per la programmazione dei PLC (tra questi vi è anche il LADDER) e suggerisce solo una possibile implementazione delle istruzioni di base che quindi possono variare a seconda del costruttore. Tuttavia è possibile tracciare un insieme di istruzioni comuni ai vari costruttori che ci riporta all'insieme di istruzioni di base prima elencato. Preferisco, invece, mostrare alcuni diagrammi LADDER studiati a lezione ed altri da me pensati a scopo esercitativo. Ogni diagramma LADDER sarà, poi, accompagnato da un diagramma per le tempificazioni. Il diagramma delle tempificazioni è funzione dell'ingresso (oppure degli ingressi), quest'ultimo produce, in base alla configurazione del diagramma, una variazione dello stato attuale delle bobine di eccitazione. Pertanto, per ogni diagramma delle tempificazioni sono assegnati in maniera del tutto casuale degli ingressi (segnati con le lettere maisucola I:x) mentre le uscite (segnate con le lettere maisucole U:x oppure W:x per le variabili di appoggio) saranno funzione dei suddetti ingressi. Qualora sia possibile, oltre al diagramma LADDER, sarà presente anche una realizzazione di una rete logica che ne rappresenta lo schema fisico. Ciò permetterà di osservare la stretta relazione che esiste tra la logica di programmazione e quella realizzativa vera e propria. Una regola essenziale da tenere sempre presente nell'analisi di un diagramma LADDER riguarda il flusso di energia che vi può scorrere, il verso di quest'ultimo può infatti andare sempre e solo da sinistra verso destra. Infine, nell'analisi del diagramma occorre sempre tenere presente che i PLC eseguono ciclicamente il programma scritto in LADDER fintanto che il PLC è alimentato dalla tensione di rete (oppure ha energia propria a sufficienza per poterlo eseguire). Ogni linea orizzontale simboleggia una istruzione detta anche rung. Essa collega due linee verticali che in un diagramma LADDER simboleggiano, invece, l'alimentazione del PLC. Per quanto riguarda il ciclo eseguito dal PLC è sufficiente sapere che quest'ultimo si compone delle fasi di: lettura degli ingressi, elaborazione degli ingressi; scrittura degli ingressi. Il suddetto ciclo richiede, per essere completato, un determinato tempo di ciclo (qualche decina o cenitinaia di ms). Ciò significa che se l'ingresso acquisito in fase di lettura cambia durante le successive fasi del ciclo,

il PLC ne concepirà l'avvenuta variazione solo percorrendo il ciclo immediatamente successivo. Prima di analizzare qualche diagramma LADDER è utile osservare come la metodologia di progetto per le reti logiche sia strettamente collegata all'implementazione di un diagramma LADDER.

*Funzioni logiche come prodotti di somme*

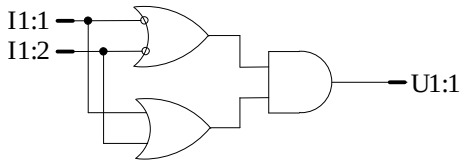
E' possibile tradurre la tabella delle verità di una funzione logica interpretando le sue righe nulle (che nell'esempio saranno contrassegnate in grassetto) come prodotti di somme. Ad esempio, sia data la seguente tabella delle verità:

| I1:1 | I1:2 | U1:1 |
|---|---|---|
| **0** | **0** | **0** |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| **1** | **1** | **0** |

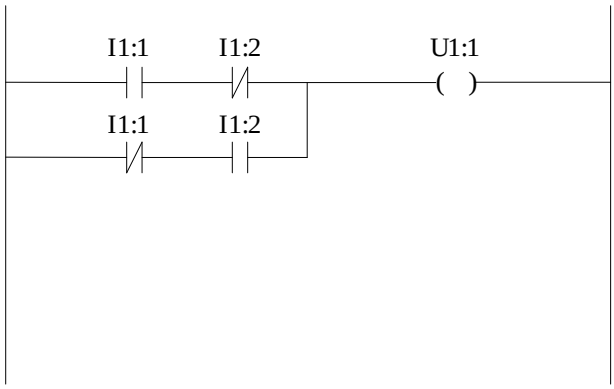
La funzione logica che descrive l'uscita `U1:1` è:

$$U1:1 = \left(\overline{I1:1} + \overline{I1:2}\right)\left(I1:1 + I1:2\right)$$

Una rete logica che modella la suddetta uscita è invece la seguente:




Il diagramma LADDER, invece, è:




che potrebbe essere interpretato come un'istruzione del tipo:

```
if(I1:1==1 && !I1:2==1)||(!I1:1==1 && I1:2==1)
      I1:1=1;
```

Osservando la tabella della verità scopriamo (se ancora non lo abbiamo fatto) che essa descrive una funzione logica XOR.

*Funzioni logiche come somma di prodotti*

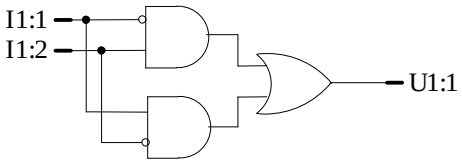
E' possibile, inoltre, tradurre la tabella delle verità di una funzione logica interpretando le sue righe non nulle (che nell'esempio saranno contrassegnate in grassetto) come somme di prodotti. Ad esempio, sia data la seguente tabella delle verità:

| I1:1 | I1:2 | U1:1 |
|---|---|---|
| 0 | 0 | 0 |
| **0** | **1** | **1** |
| **1** | **0** | **1** |
| 1 | 1 | 0 |

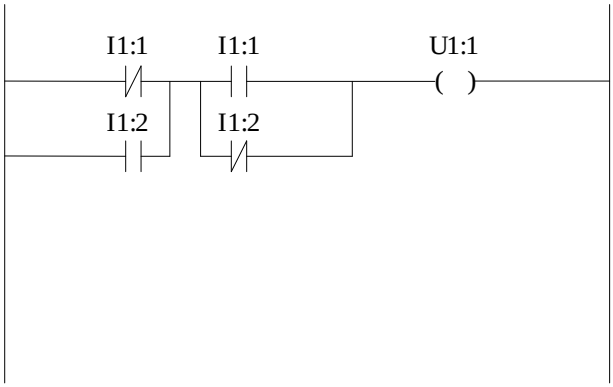
La funzione logica che descrive l'uscita `U1:1` è:

$$U1:1 = \left(\overline{I1:1} \cdot I1:2\right) + \left(I1:1 \cdot \overline{I1:2}\right)$$

Una rete logica che modella la suddetta uscita è invece la seguente:




Il diagramma LADDER, invece, è:



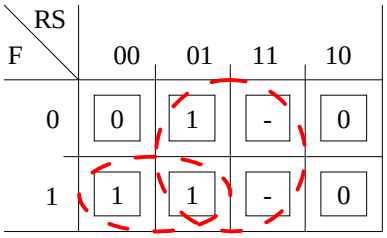

che potrebbe essere interpretato come un'istruzione del tipo:

```
if(I1:1==1 || !I1:2==1)&&(!I1:1==1 || I1:2==1)
      I1:1=1;
```

Osservando la tabella della verità scopriamo (se ancora non lo abbiamo fatto) che essa descrive una funzione logica XOR.

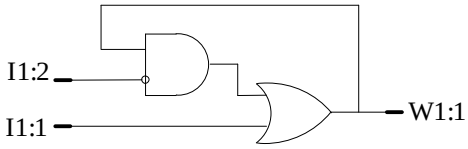
*Flip flop RS*

Il flip flop RS è un elemento di memoria bistabile ossia capace di memorizzare due valori logici. Progetteremo in questo esempio una rete logica che realizzi il comportamento di un flip flop RS, quindi disporremo le istruzioni base del LADDER per effettuarne una sua simulazione:

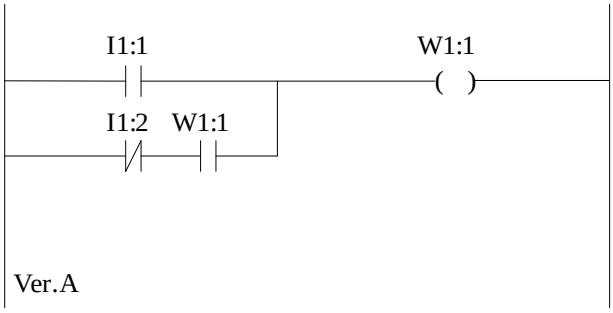



$$F = S + \overline{R}F$$

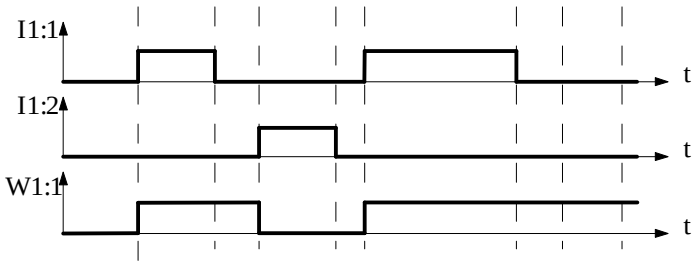
Nell'ipotesi in cui S=I1:1, R=I1:2 ed F=W1:1 si ha la seguente rete logica:




che corrisponde alla prima versione (Ver.A) del flip flop RS che qui mostriamo:




Una possibile temporizzazione della Ver.A del flip flop RS è la seguente:



Un altro modo per realizzare un flip flop RS in LADDER è il seguente (Ver.B vista a lezione che corrisponde all'approccio basato su una funziona logica ricavata da somme di prodotti):

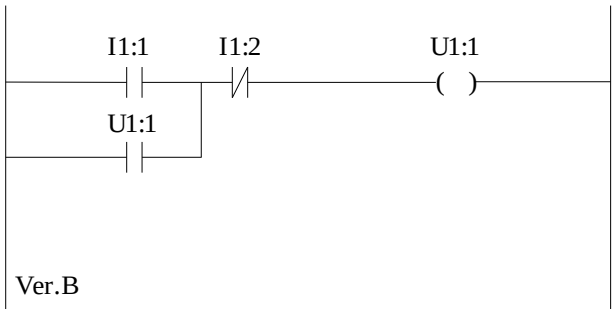
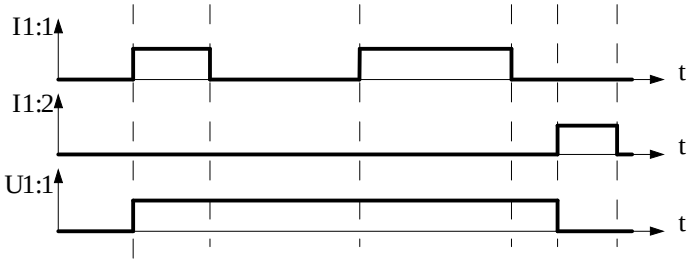



Diagramma delle tempificazioni:



*Riconoscitore di fronte di salita*
Questo diagramma LADDER riconosce il fronte di salita di un segnale di ingresso (I1:1):

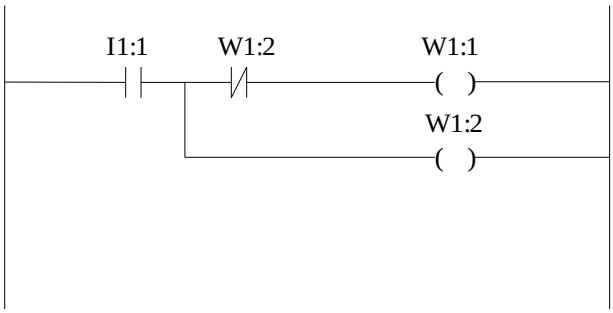
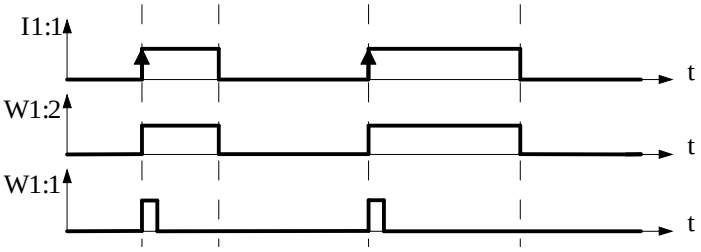



Diagramma delle tempificazioni:




Per comprendere meglio il funzionamento del diagramma gli impulsi sui fronti di salita sono stati accentuati. La variabile W1:2 di appoggio è una copia dell'ingresso: quando I1:1 commuta la bobina della suddetta variabile ne segue le evoluzioni. Nei primi cicli del diagramma non accade nulla finchè l'ingresso è immutato. Non appena il valore logico della variabile di ingresso I1:1 diventa alto il primo rung del diagramma è in grado di energizzare la bobina W1:1 come tra l'altro mostra il diagramma delle tempificazioni. Il ciclo del PLC procede ed analizza adesso il secondo rung: nulla di nuovo, la bobina W1:2 segue come già anticipato l'ingresso I1:1. Al successivo ciclo il PLC legge in ingresso un valora logico ancora alto di I1:1, nonostante ciò il primo rung del diagramma adesso non può energizzare più la bobina W1:1. Pertanato l'ampiezza dell'impulso è pari al tempo di ciclo del PLC! La variabile W1:1 assume un valore logico basso mentre il rung successivo copia ancora una volta in uscita, su W1:2, la variabile di ingresso. Quando la variabile di ingresso I1:1 assume un valore logico basso la variabile di appoggio W1:2 ne segue le evoluzioni mentre l'uscita W1:1 è immutata fino al prossimo fronte di salita.

*Riconoscitore di fronte di discesa*
Questo diagramma LADDER riconosce il fronte di discesa di un segnale di ingresso (I1:2):

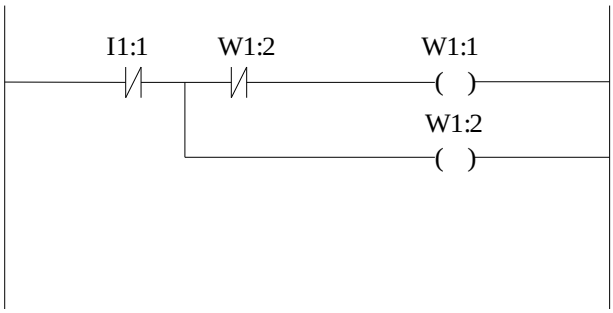
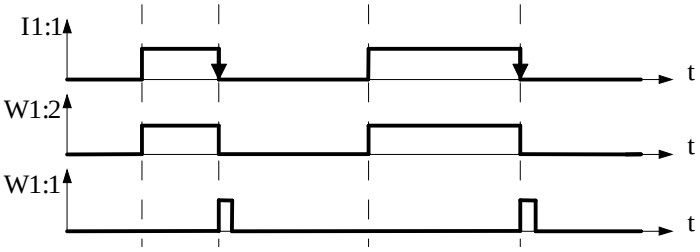


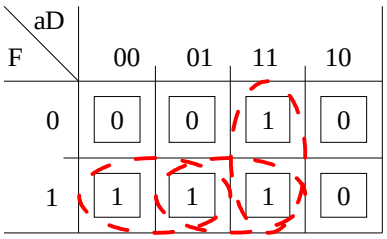

Diagramma delle tempificazioni:




Il funzionamento è duale al diagramma precedente.

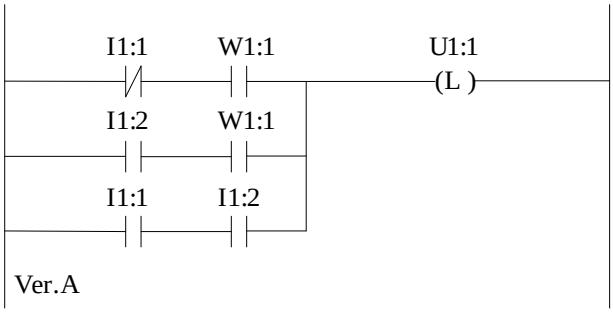
*Flip flop D*

Il flip flop D latch è un altro elemento di memoria assai comune nelle reti logiche. Il flip flop D latch è talvolta detto flip flop abilitato poichè commuta il bit in ingresso solo se l'elemento è abilitato.

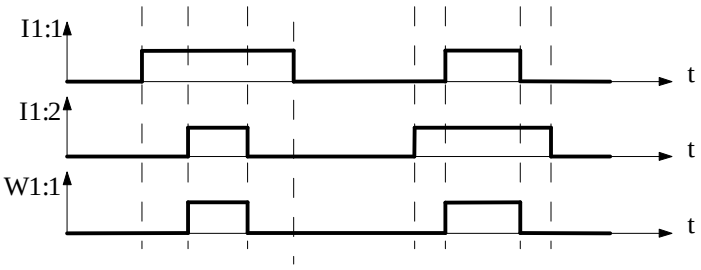



$$F = \bar{a}F + DF + aD$$

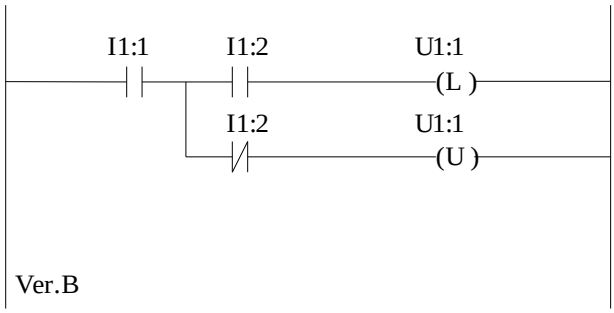
E' possibile, quindi, realizzare il flip flop D latch come somma di prodotti (Ver.A) in cui I1:1=a, I1:2=D e WI:1=F::



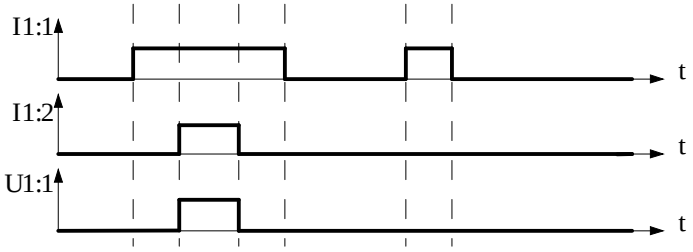

Un esempio di tempificazione dei segnali:




Osservazione: il secondo rung del diagramma (Ver.A) è ridondante, esso solitamente è aggiunto in fase di progettazione della rete logica mediante mappa di Karnough per evitare propblemi relativi alle alee.
Ed ancora, è possibile seguire una diversa strategia (Ver.B) per la realizzazione di un flip flop D. Questa ci permette, tra l'altro, di osservare il comportamento di una particolare istruzione del LADDER: la bobina latch il cui simobolo è --(L)--. Essa mantiene alto il segnale di uscita anche se il relativo segnale di ingresso, poi, diventa basso. Solo l'istruzione di unlatch, il cui simbolo è --(U)--, può azzerarne lo stato.

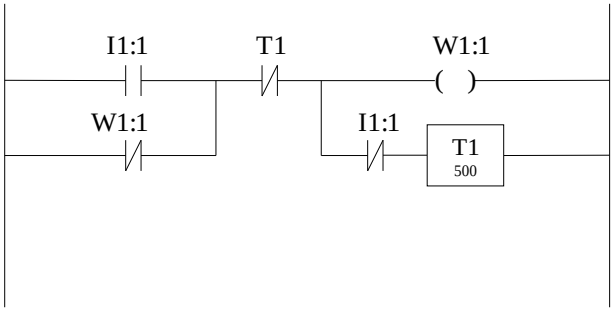



Un esempio di tempificazione dei segnali:

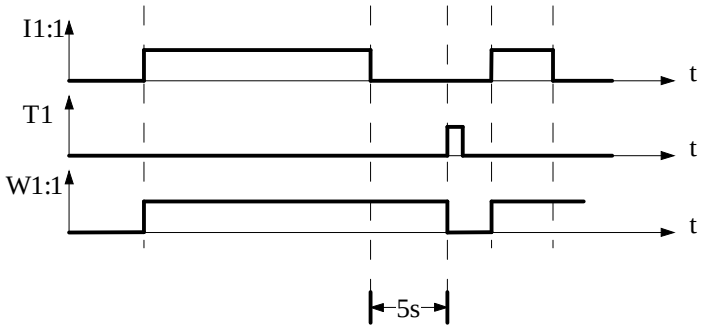



Dove I1:1 è il segnale di abilitazione e I1:2 è il dato da acquisire. Finquanto chè il segnale di abilitazione si tiene basso nessun dato in ingresso sarà memorizzato. Non appena I1:1 assume un valore logico alto il segnale I2:2 può essere memorizzato. Nel secondo rung, per I1:1=1 ed I1:2=0, la bobina di uscita U1:1 subisce un unlatch è conterrà pertanto il valore logico basso. Sotto le suddette condizioni il primo rung, infatti, non può energizzare la bobina U1:1. Solo quando I1:1=1 ed I1:2=1 il primo rung può essere portato a termine energizzando la bobina U1:1 con una operazione di latch sicchè se pure I1:1 dovesse diventare basso lo stato della bobina I1:1 rimane invariato. Quindi se I1:1=1 ed I1:2 ridiventa basso l'unica azione possibile è quella di unlatch che azzera la variabile U1:1.

*Ritardo di spegnimento*
Abilitando l'ingresso I1:1 l'uscita U1:1 (che si mantiene alta anche se I1:1 cambia valore logico) ha una durata (e quindi ritardo di spegnimento) di 5s dopo che lo stato logico di I1:1 è variato (immaginate il classico sistema elettrico per far rimanere accesa l'illuminazione condominiale, per un certo numero di secondi, dopo che un pulsante principale è stato azionato).



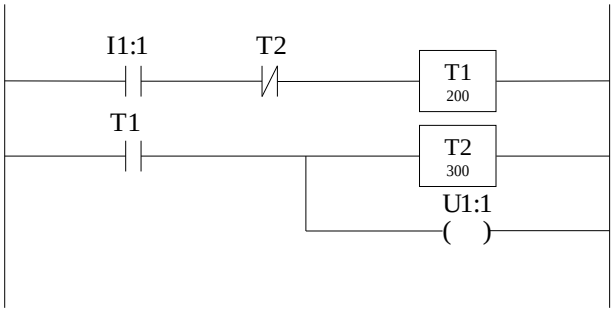

Un esempio di tempificazione dei segnali:



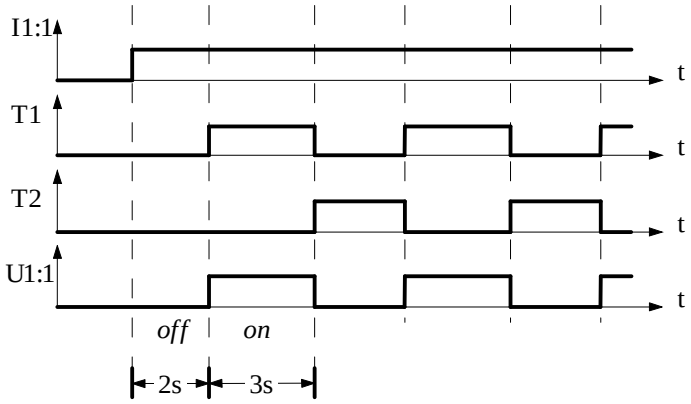

Il timer T1 viene avviato non appena l'ingresso I1:1 passa allo stato logico 0 (la variabile di appogio, per questo motivo, ne memorizza il valore) sicchè l'energia può fluitr da sinistra verso destra attraverso il percorso !W1:1;!T1;!I1:1;Timer T1 (con il simbolo ! si è indicata la negazione logica dei segnali). Quando il conteggio è avviato la variabile di appoggio W1:1 viene fissata al suo valore logico alto ed è effettivamente azzerata solo quando il timer T1 raggiunge il valore di conteggio (500 cicli=5s).
Dunque la variabile W1:1 potrebbe simboleggiare una luce che rimane accesa 5s dopo aver rilasciato il relativo tasto per l'accensione.

*Oscillatore ad onda quadra*

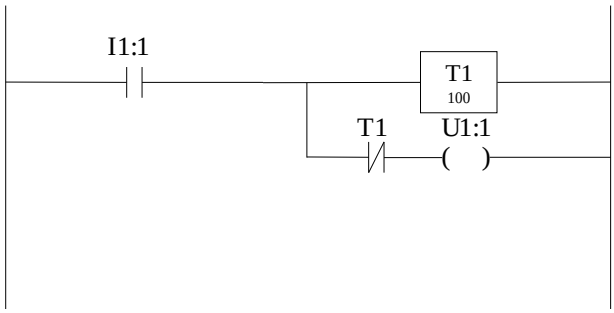
Questo diagramma realizza, a partire da un segnale di ingresso I1:1 fissato al valore logico alto, un segnale ad onda quadra in cui i periodi di on/off sono stabiliti dal valore di conteggio dei rispettivi timer:



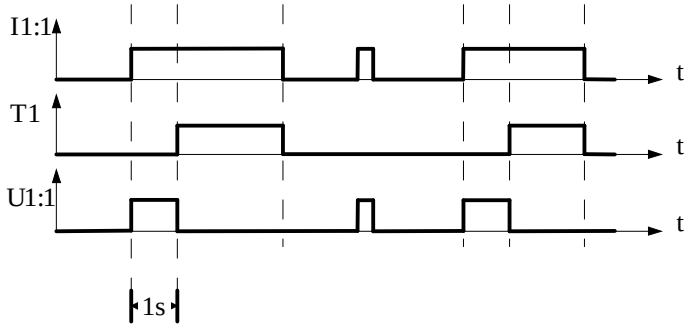

Un esempio di tempificazione dei segnali:



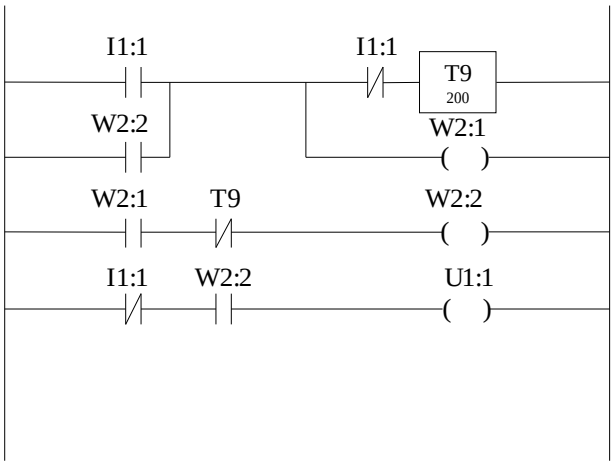
*Impulso all'accensione*
Quando è presente un certo ingresso si vuole avere un impylso in uscita della durata di un secondo:




Un esempio di tempificazione dei segnali:



*Impulso allo spegnimento*
Quando è presente un certo ingresso che poi termina si vuole avere un impulso di 2 secondi in uscita sopo la fine dell'ingresso:




Un esempio di tempificazione dei segnali: