Compilatore NILAB ST to IL - Simulatore

NILAB uPLC — Convertitore Structured Text to IL

Strumento online per programmare i motori lineari integrati NILAB (famiglia NLi) usando la sintassi IEC 61131-3 Structured Text. Lo strumento compila il sorgente ST nel formato Instruction List (IL) compatibile con il firmware NILAB µPLC (>= 5E41).

Accesso allo strumento: ni-lab.online → Strumenti → ST Converter

---

1. Panoramica

Il µPLC NILAB è una soft-PLC embedded che gira dentro ogni motore lineare integrato NLi. Esegue un programma scritto nel formato Instruction List (IL) — un linguaggio di basso livello, riga per riga, simile all'assembly. Scrivere IL direttamente è soggetto a errori e difficile da leggere. Il convertitore ST-to-IL ti permette di scrivere programmi in **Structured Text (ST)**, un linguaggio ad alto livello IEC 61131-3 simile a Pascal, e li compila automaticamente in IL.

Parametro	Valore
Firmware richiesto	>= 5E41
Max istruzioni	64
Ciclo di scan	6 ms (fisso)
Profondità stack	1 (niente AND/OR annidati in un IF)
Timer	2 timer hardware (R8028, R8029)

Fatti chiave sul µPLC NILAB:

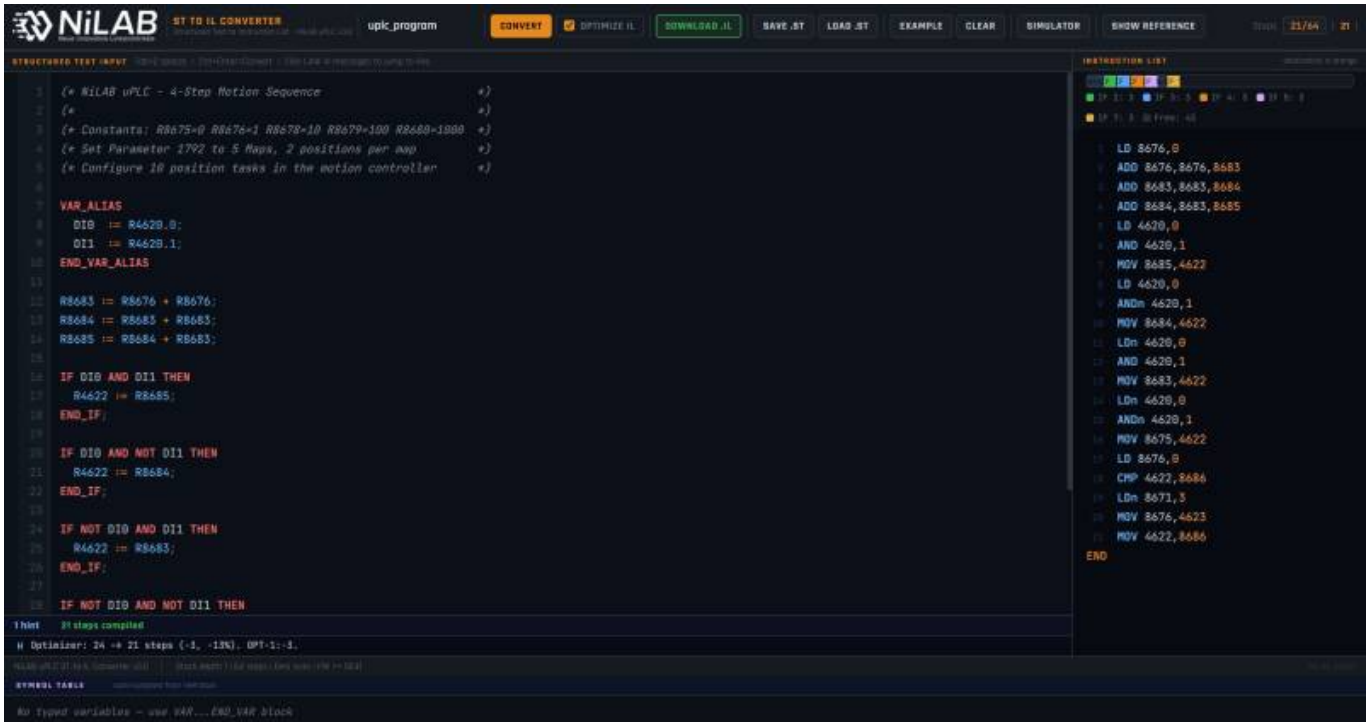


Figura 1: Interfaccia principale. Colonna sinistra: editor ST. Colonna destra: output IL generato. Pannelli inferiori: Symbol Table e IL Simulator (quando attivo).

---

2. Layout dell'interfaccia

Lo strumento è diviso in tre aree principali:

① **Toolbar** — campo nome progetto, pulsante Convert, Download IL, Salva/Carica file ST, checkbox Optimize, toggle Simulator, Mostra/Nascondi pannello Reference.

② **ST Editor (colonna sinistra, ~65% larghezza)** — Scrivi qui il tuo programma Structured Text. Caratteristiche:

Voce
Evidenziazione sintassi (parole chiave in rosso, registri in blu, nomi tipo in viola, commenti in grigio)
Numeri riga con marcatori errore nella gutter (rosso `!` = errore, giallo `W` = avviso)
Tasto Tab inserisce 2 spazi
Ctrl+Invio attiva la conversione

③ **IL Output (colonna destra, ~35% larghezza)** — L'Instruction List compilato appare qui dopo aver premuto Convert. I numeri di passo sono mostrati in grigio a sinistra; il registro di destinazione delle istruzioni matematiche è evidenziato in arancione.

④ **Message Panel** — Situato sotto l'editor ST. Mostra errori di compilazione (rosso), avvisi (giallo) e suggerimenti dell'ottimizzatore (blu). Clicca qualsiasi tag posizione L## per saltare a quella riga nell'editor.

⑤ **Symbol Table** — Appare in fondo alla pagina quando è presente un blocco VAR. Mostra tutte le variabili dichiarate con i loro indirizzi di registro assegnati automaticamente.

⑥ **IL Simulator** — Appare sotto la Symbol Table quando il pulsante Simulator è attivato. Consente esecuzione passo-passo o continua dell'IL compilato.

⑦ **Reference Panel** — Nascosto di default. Premi *Mostra Reference* nella toolbar per aprire una barra laterale di quick-reference della sintassi.

---

3. Riferimento linguaggio ST

3.1 Blocco VAR_ALIAS

Usa VAR_ALIAS per dare nomi leggibili a registri hardware e indirizzi di bit. Queste sono semplici sostituzioni di testo — nessun registro viene assegnato automaticamente.

```
\
VAR_ALIAS\
DI0  := R4620.0;    (* Digital Input 0, bit 0 di R4620 *)\
DI1  := R4620.1;    (* Digital Input 1, bit 1 di R4620 *)\
ACTV := R8684.1;    (* flag sequenza attiva             *)\
TIMER1 := R8028;    (* registro Timer 1                 *)\
END_VAR_ALIAS\
```

3.2 Blocco VAR (variabili tipizzate)

Usa VAR per dichiarare variabili tipizzate. Il compilatore **assegna automaticamente** il prossimo registro libero dal pool di allocazione — non devi conoscere gli indirizzi dei registri.

```
\VAR
counter : INT  := 0;   (* assegnato automaticamente a R8683         *)
enable  : BOOL;        (* assegnato automaticamente a R8681.0       *)
done    : BOOL := TRUE;(* assegnato automaticamente a R8681.1, init *)
T1      : TON;         (* assegnato automaticamente a Timer1 R8028  *)
edge1   : R_TRIG;      (* assegnati automaticamente 2 slot BOOL     *)
\END_VAR
```

Tipi supportati:

Tipo	Pool registri IL	Max	Valore iniziale
BOOL	R8681 bit 0-15, poi R8682 bit 0-15	32	0/1/TRUE/FALSE
INT	R8683, R8684, R8685, R8686, R8687, R8688	6	0, 1, 10, 100, 1000
DWORD	stesso pool di INT	6	uguale
TON	R8028 (Timer1), R8029 (Timer2)	2	-
CTD	R8028 (Timer1), R8029 (Timer2)	2	-
CTU	uno slot INT per valore contatore	16	-
R_TRIG	2 slot BOOL (memoria + uscita Q)	16	-

F_TRIG	2 slot BOOL (memoria + uscita Q)	16	-

> **Nota:** Le costanti R8675–R8680 sono di sola lettura e non possono essere usate come destinazioni di scrittura. Il compilatore lo impone con un errore.

3.3 Costanti di registro

I seguenti registri costanti sono sempre disponibili senza dichiarazione:

Registro	Valore	Esempio di utilizzo
R8675	0	Azzera un registro a zero
R8676	1	Incrementa, booleano TRUE
R8678	10	Moltiplica o confronta
R8679	100	Valore ricarica timer (600 ms)
R8680	1000	Costante grande

I letterali interi 0, 1, 10, 100 e 1000 sono mappati automaticamente a questi registri nel compilatore. Qualsiasi altro valore letterale produrrà un errore di compilazione — carica prima il valore in un registro utente.

3.4 IF / ELSE / END_IF

```
\
IF condition THEN\
statement(s);\
ELSE                (* opzionale *)\
statement(s);\
END_IF;\
```

Operatori di condizione:

Operatore	Istruzione IL	Note
AND	AND / ANDn	Contatto in serie
OR	OR / ORn	Contatto in parallelo
NOT	LDn / ANDn / ORn	Inverte il termine seguente

Miscelare AND e OR è supportato con valutazione strettamente da sinistra a destra (nessuna precedenza operatori): * `A AND B OR C` → IL: `LD A, AND B, OR C` → risultato: *(A AND B) OR C* * `A OR B AND C` → IL: `LD A, OR B, AND C` → risultato: *(A OR B) AND C* * Un **avviso** viene mostrato quando AND e OR sono miscelati nella stessa condizione.

> **Limite profondità stack = 1:** Ogni condizione IF usa lo stack a livello singolo. Non puoi usare `(A AND B) OR (C AND D)` in un IF — dividilo in due blocchi IF o usa un flag BOOL intermedio.

3.5 Istruzioni di bit

```
SET(R4621.0);           (* latch bit ON  *)\
RESET(R4621.0);         (* latch bit OFF *)\
```

```
R4621.0 := 1;           (* equivalente a SET  *)\
R4621.0 := 0;           (* equivalente a RESET *)\
```

3.6 Matematica di registro

```
(* Quattro operazioni aritmetiche *)
Rc := Ra + Rb;          (* IL: ADD Ra, Rb, Rc *)
Rc := Ra - Rb;          (* IL: SUB Ra, Rb, Rc *)
Rc := Ra * Rb;          (* IL: MUL Ra, Rb, Rc *)
Rc := Ra / Rb;          (* IL: DIV Ra, Rb, Rc — divisione intera *)

(* Copia *)
Rb := Ra;               (* IL: MOV Ra, Rb *)

(* Confronto — imposta flag in R8671 *)
CMP(Ra, Rb);
```

Flag risultato CMP in R8671:

Bit	Flag	Significato
.0	T1	Timer1 (R8028) scaduto
.1	T2	Timer2 (R8029) scaduto
.2	LT	Pa < Pb
.3	EQ	Pa = Pb
.4	GT	Pa > Pb

3.7 Blocchi funzione

I blocchi funzione vengono chiamati per nome con assegnazioni di porte con nome. Sono espansi inline in istruzioni IL.

R_TRIG — Rilevatore fronte di salita

```
\
VAR\
edge1 : R_TRIG;\
END_VAR\
VAR_ALIAS\
DI1 := R4620.1;\
END_VAR_ALIAS

edge1(CLK := DI1);

(* edge1.Q è TRUE per esattamente uno scan sul fronte di salita di DI1 *)\
IF edge1.Q AND NOT ACTV THEN\
SET(ACTV);\
```

```
END_IF;\
```

IL generato (4 istruzioni): `LD / ANDn / OUT / LD / OUT`

F_TRIG — Rilevatore fronte di discesa

Uguale a R_TRIG ma scatta sul fronte di **discesa**. Porta: `CLK`.

TON — Timer di ritardo all'accensione

```
\VAR
T1 : TON;
END_VAR

T1(IN := ACTV, PT := 100);
(* PT=100 scan × 6ms = 600ms di ritardo *)
(* T1 scaduto quando R8671.0 = 1      *)

IF R8671.0 AND ACTV \THEN
(* azione timer scaduto *)
R8028 := R8679;    (* ricarica timer *)
END_IF;
```

CTU — Contatore in su

```
\VAR
C1 : CTU;
END_VAR

C1(CU := pulse_bit, PV := 5);
(* C1.Q = TRUE quando il conteggio >= 5 *)
```

---

4. Ottimizzatore IL

Abilita la checkbox **Optimize IL** prima di premere Convert per attivare l'ottimizzatore. Esegue fino a 12 passaggi iterativi e applica quattro trasformazioni:

OPT-1 — Hoisting della condizione (risparmio maggiore)

Ogni istruzione dentro un blocco IF normalmente ripete la condizione completa. L'ottimizzatore rileva condizioni ripetute consecutive e le emette solo una volta, mantenendo lo stack attivo su più azioni.

```
Prima:   LD 8671.2  AND 8671,0  ADD 4622,8683,4622   (3+1 = 4 istruzioni)
```

```
        LD 8671.2  AND 8671,0  MOV 8679,8028         (3+1 = 4 istruzioni)

Dopo:    LD 8671.2  AND 8671,0  ADD 4622,8683,4622    (3+1+1 = 5 istruzioni)
                                MOV 8679,\8028
```

Per un IF con N istruzioni e una condizione di lunghezza C, il risparmio è `(N-1) × C` istruzioni.

OPT-2 — IF/ELSE SET/RES → OUT

`IF A THEN SET(Q); ELSE RESET(Q); END_IF;` è semanticamente identico a `OUT Q` — il bit riceve esattamente il valore dello stack.

```
Prima:   LD 4620.1   SET 8684,0      (2 istruzioni)
        LDn 4620.1  RES 8684,0      (2 istruzioni)

Dopo:    LD 4620.1   OUT 8684,0      (2 istruzioni totali)
```

OPT-3 — Eliminazione MOV morto

`MOV Pa, Pa` (copiare registro su se stesso) viene rimosso silenziosamente.

OPT-4 — Eliminazione doppio NOT

`LDn + OUTn` → `LD + OUT` (la doppia negazione si cancella).

Risparmi tipici: 30–45% su programmi con più blocchi IF. Il pannello messaggi mostra una ripartizione: es. *Optimizer: 38 → 22 passi (-42%). OPT-1:-14, OPT-2:-2. Passaggi: 2.*

---

5. Symbol Table

La Symbol Table appare automaticamente in fondo alla pagina quando il programma contiene un blocco VAR. Mostra:

Il badge del conteggio (es. *3 vars*) mostra quante variabili tipizzate sono state dichiarate.

Usa la Symbol Table per sapere esattamente quale registro corrisponde a ogni variabile quando si monitora il motore con MODBUS o NILAB Starter.

---

6. IL Simulator

L'IL Simulator esegue il programma IL compilato ciclo per ciclo direttamente nel browser, consentendo la verifica funzionale senza collegarsi all'hardware.

Attivare il Simulator

> Se modifichi il programma ST e premi di nuovo Convert mentre il Simulator è attivo, l'IL viene ricaricato e la simulazione si reimposta automaticamente.

Pannelli del Simulator

Input R4620 — Tre pulsanti toggle (IN0, IN1, IN2) che rappresentano i bit 0-2 del registro ingressi digitali R4620. Clicca per commutare il bit. Evidenziazione verde = bit è 1.

Output R4621 — Quattro LED che mostrano i bit 0-2 di R4621 (uscite digitali) e il bit 0 di R4096 (abilitazione drive). Arancione = attivo.

R4622 — Mostra il valore corrente del registro di uscita utente R4622 (indice tabella motion controller, comando posizione, ecc.).

Flags R8671 — Pillole indicatore per ogni bit di flag. Verde = attivo: * T1, T2 — timer scaduto * LT, EQ, GT — risultato dell'ultima istruzione CMP

Controlli: * **Reset** — Ricarica IL dal pannello di output e inizializza tutti i registri * **Step** — Esegue un ciclo di scan completo (o premi **Spazio** quando in pausa) * **Run / Pause** — Esecuzione continua alla velocità selezionata * **Slow / Med / Fast / 6ms** — Intervallo di scan: 1000ms / 300ms / 80ms / 6ms (velocità hardware reale)

Registers — Un elenco di tutti i registri referenziati nell'IL compilato, mostrati come campi modificabili. I valori si aggiornano a ogni ciclo di scan. Puoi anche digitare un nuovo valore per sovrascrivere un registro (utile per iniettare un valore di posizione o simulare una scrittura MODBUS).

Simulazione timer

Entrambi i timer hardware (R8028, R8029) vengono decrementati di 1 all'inizio di ogni ciclo di scan. I flag T1/T2 in R8671 vengono impostati quando il timer raggiunge zero. Per simulare un timer di 600ms:

---

7. Salva, Carica e Download

Pulsante	Azione
CONVERT	Compila il programma ST. Scorciatoia: Ctrl+Invio
DOWNLOAD .IL	Salva l'IL compilato come file di testo `.il`. Bloccato se ci sono errori
SAVE .ST	Salva il sorgente ST corrente come file .st
LOAD .ST	Apre un file .st salvato in precedenza nell'editor. Attiva la conversione automatica
OPTIMIZE IL	checkbox — abilita prima di Convert per attivare l'ottimizzatore IL a 4 passaggi
SIMULATOR	Toggle del pannello IL Simulator
SHOW REFERENCE	Toggle della barra laterale di riferimento sintassi
PROJECT NAME FIELD	Imposta il nome file usato dai pulsanti Download/Save

8. Caricare l'IL nel motore

Dopo aver scaricato il file `.il`:

> La dimensione massima del programma è **64 istruzioni** (incluso l'obbligatorio END). Il contatore passi nella toolbar mostra l'utilizzo corrente e diventa giallo sopra 48 passi e rosso sopra 64.

9. Limiti e vincoli noti

Vincolo	Dettagli
Max istruzioni	64 passi (incluso END)
Profondità stack	1 — nessun AND/OR annidato in una condizione IF
No loop FOR / WHILE	Usa invece sequenze basate su timer o logica srotolata
No IF annidato	Usa bit di flag (variabili BOOL) per stati complessi
No aritmetica nelle condizioni	Valuta con CMP prima, poi testa i bit flag R8671
Letterali costanti	Solo 0, 1, 10, 100, 1000 sono mappati a registri costanti
MAX variabili BOOL	32 (bit 0-15 di R8681 e R8682)
MAX variabili INT	6 (R8683 a R8688)
MAX timer (TON/CTD)	2 (R8028 e R8029)
Intervallo indice bit	solo 0 a 15
Registri di sola lettura	R8675–R8680 non possono essere destinazioni di scrittura

10. Esempio completo — Movimento a 4 passi con 2 ingressi digitali

Questo programma seleziona uno di quattro voci della tabella di movimento in base allo stato di due ingressi digitali. È tipicamente usato con il motion controller NILAB configurato per 5 mappe con 2 posizioni per mappa (parametro 1792 = 5).

```
\
(* NILAB uPLC - Sequenza di movimento a 4 passi con 2 Ingressi Digitali  *)\
(*                                                                       *)\
(* Parametro 1792 = 5 Mappe, 2 posizioni per mappa                       *)\
(* Motion controller: 10 task di posizione configurati                     *)

VAR_ALIAS\
DI0  := R4620.0;    (* Digital Input 0 *)\
DI1  := R4620.1;    (* Digital Input 1 *)\
END_VAR_ALIAS

(* Pre-calcola le costanti dell'indice tabella *)\
```

```
R8683 := R8676 + R8676;   (* R8683 = 2 *)\
R8684 := R8683 + R8683;   (* R8684 = 4 *)\
R8685 := R8684 + R8683;   (* R8685 = 6 *)

(* Seleziona l'indice tabella in base alla combinazione DI0 e DI1 *)\
IF DI0 AND DI1 THEN         (* DI0=1, DI1=1 → indice 6 *)\
R4622 := R8685;\
END_IF;

IF DI0 AND NOT DI1 THEN     (* DI0=1, DI1=0 → indice 4 *)\
R4622 := R8684;\
END_IF;

IF NOT DI0 AND DI1 THEN     (* DI0=0, DI1=1 → indice 2 *)\
R4622 := R8683;\
END_IF;

IF NOT DI0 AND NOT DI1 THEN (* DI0=0, DI1=0 → indice 0 *)\
R4622 := R8675;\
END_IF;

(* Scrivi l'indice tabella solo quando cambia *)\
CMP(R4622, R8686);\
IF NOT R8671.3 THEN         (* flag EQ = 0 significa valore cambiato *)\
R4623 := R8676;           (* segnale: nuovo indice tabella attivo  *)\
R8686 := R4622;           (* salva indice corrente come riferimento *)\
END_IF;\
```

Tabella di verità:

DI0	DI1	R4622 (indice tabella) - R4623 deve essere impostato a 1	Posizione di movimento
0	0	0	Task di posizione 0 e 1
0	1	2	Task di posizione 2 e 3
1	0	4	Task di posizione 4 e 5
1	1	6	Task di posizione 6 e 7

Come testare con il Simulator: - Premi Convert (con Optimize IL abilitato: 23 istruzioni → ~16) - Premi Simulator → Reset - Commuta i pulsanti IN0 e IN1 — osserva R4622 che cambia nel pannello registri - Il flag EQ (R8671.3) dovrebbe essere 0 quando il valore cambia e 1 quando rimane uguale - R4623 cambia a 1 solo nello scan quando l'indice tabella cambia

Vedi anche:

Voce
Riferimento set istruzioni uPLC
Programmi di esempio uPLC

From:
https://www.nilab.at/dokuwiki/ - **NiLAB GmbH Knowledgebase**

Permanent link:
https://www.nilab.at/dokuwiki/doku.php?id=it:nilab_st_converter:start

Last update: **2026/09/22 17:58**