

UNIVERSITÀ DEGLI STUDI DI PADOVA

FACOLTÀ D'INGEGNERIA

**CORSO DI LAUREA IN INGEGNERIA MECCANICA E
MECCATRONICA – CURRICULUM MECCANICO**

TESI DI LAUREA

**PIANIFICAZIONE DEL MOTO PER MANIPOLATORI
CONTROLLATI DA SISTEMI BASATI SU WINDOWS CE**

**RELATORE:
Prof. GIOVANNI BOSCHETTI**

**LAUREANDO:
FRANCESCO PARMEGGIANI**

ANNO ACCADEMICO 2010-2011

Sommario

Introduzione	1
Capitolo 1	3
<i>Componenti Hardware</i>	3
Beckhoff CX1020	3
Alimentazione del sistema	6
Terminali EtherCAT	9
Beckhoff EL1252	12
Beckhoff EL2202	16
Beckhoff EL2252	18
Beckhoff EK1110	22
Capitolo 2	25
<i>Software</i>	25
Windows Embedded CE	25
Microsoft Visual Studio 2008 Professional©	28
<i>TwinCAT</i>	32
TwinCAT System Manager	33
Capitolo 3	45
<i>Legge di moto trapezoidale</i>	45
Capitolo 4	53
<i>Implementazione del codice</i>	53
Premessa: stati dell'azionamento	53
Interfaccia utente	56
Conclusioni	63
Bibliografia	65
Ringraziamenti	67
Appendice 1	69
Appendice 2	70
Appendice 3	71

Introduzione

La multinazionale Beckhoff realizza sistemi aperti per automazione con tecnologia di controllo basata su PC. La gamma di prodotti copre i principali settori dell'industria come PC industriali, componenti per bus di campo e I/O, Motion Control e software di automazione. Uno tra questi dispositivi (in particolare il modello Beckhoff CX1020), donato al Dipartimento di Tecnica e Gestione dei Sistemi Industriali dell'università di Padova dalla medesima azienda, è stato oggetto della tesi. Finalità principale, come attesta il titolo, consiste nella pianificazione del moto di un manipolatore sfruttando le potenzialità del sistema, moto di tipo trapezoidale cioè caratterizzato da tratti alternati ad accelerazione o velocità costante attraverso un bus di comunicazione di tipo CANopen. La possibilità d'interazione dell'utente con il sistema è data da un'interfaccia grafica sviluppata. L'obiettivo, quindi, è quello d'interfacciare il nostro computer industriale al drive in dotazione al quale sono collegati due motori: attraverso il software proprietario TwinCAT siamo in grado di gestire con semplicità la cosa permettendo di collegare le uscite fisiche direttamente a variabili logiche facilmente gestibili d'ambienti sviluppo software tra i quali si annovera il software Visual Studio © progettato da Microsoft.

Il primo e il secondo capitolo della tesi porteranno al lettore una breve descrizione del sistema globale hardware e software utilizzato oltre che le modalità con cui si è interfacciato il tutto per : il dispositivo Beckhoff CX1020, le sue diverse unità, il driver, i due motori, l'utilizzo di TwinCAT System manager, l'ambiente di sviluppo Visual Studio.

Il terzo si sofferma sulla descrizione della legge di moto studiata e analizzata, portandone i riferimenti analitici e la modalità con cui essa viene utilizzata nel progetto. Partendo da una definizione matematica, sono portate le relazioni principali oltre che la rappresentazione grafica la quale permette al lettore di dare un riscontro chiaro di tutto quello prima elencato.

Il quarto, il cuore dell'elaborato, porta al lettore l'intera implementazione del codice sotto forma di un file dll, interamente riportato e commentato, per lo sviluppo della legge di moto e di un'interfaccia grafica, la cui mutua comunicazione è fondamentale; entrambe sono state sviluppate con la piattaforma VISUAL STUDIO.

In coda si riportano alcune riflessioni circa l'intero sviluppo e le problematiche principali incontrate durante la programmazione.

Capitolo 1

Componenti Hardware

Beckhoff CX1020

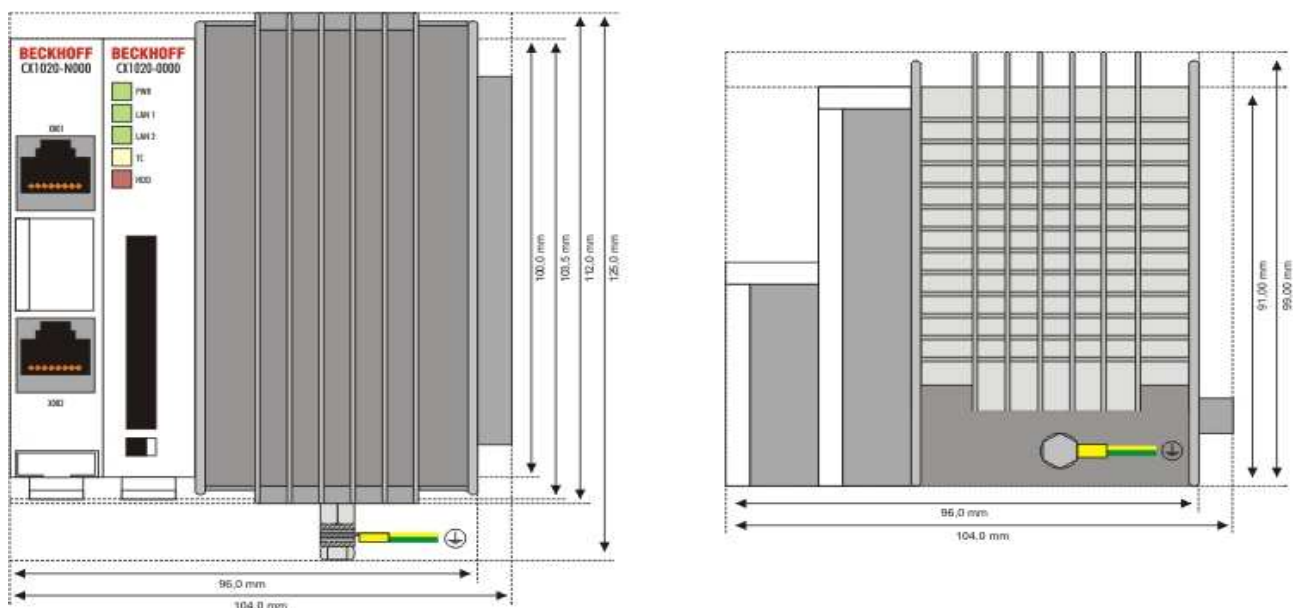


Fig.1.1 – Dispositivo Beckhoff

Con la serie CX dei computer Embedded, Beckhoff ha combinato la tecnologia del PC e I/O modulari per formare un'unità di alto livello. La CX1020 estende la famiglia di prodotti CX con una versione più performante dal punto di vista della CPU permettendo il collegamento diretto tra i terminal EtherCAT e i Bus Terminals.

Il CX1020 è equipaggiato con 1 GHz Intel® CPU caratterizzandosi come un dispositivo a risparmio energetico che opera con ultra-core a bassa tensione e a bassa potenza termica con caratteristiche di dissipazione di soli 7 W TDP (thermal design power). Di conseguenza, non è necessario l'utilizzo di una ventola nonostante il design molto compatto del computer embedded PC CX1020

Il Compact Flash è usato come supporto di memoria permettendo l'aumento del MTBF (Mean Time Between Failures) dell'intero sistema.

Sia il case che le modalità d'assemblaggio del CX1020 sono del tutto simili a quella del fratello minore CX1000: esso è inoltre caratterizzato da diversi componenti che

possono essere dall'utente. La configurazione più semplice è caratterizzata da un modulo CPU e da unità multifunzionali di alimentazione.

Il modulo base è equipaggiato come standard con due prese RJ-45 e un 3-port switch come standard.

Come il CX1000, il CX1020 può essere espanso con interfacce di sistema opzionali: un DVI-I (=DVI-D + VGA) output, due interfacce USB-2.0 e fino a quattro interfacce RS232 e audio sono disponibili. La funzione di quattro interfacce RS232 è di disaccoppiamento ottimale e può essere implementato come RS422/RS.

Il sistema operativo può essere scelto tra Windows CE o Windows Embedded Standard. Il software per automazione TwinCAT trasforma il sistema CX1020 in un potentissimo PLC oltre che in un sistema per il controllo del movimento il quale può essere gestito con o senza visualizzazione. A differenza del CX1010, il CX1020 può essere anche usato per interpolare movimenti degli assi con il software TwinCAT NC I.

Altre interfacce del sistema o fieldbus per la connessione possono essere aggiunti al modulo CPU base. Il modulo CPU richiede un'unità di alimentazione del tipo CX1100. Tutti i moduli fieldbus CX1500 e le unità d'alimentazione CX1100 dalla serie Cx possono essere usati in combinazione con il CX1020.

La tabella sotto riporta il significato del simbolo di ciascun prodotto:

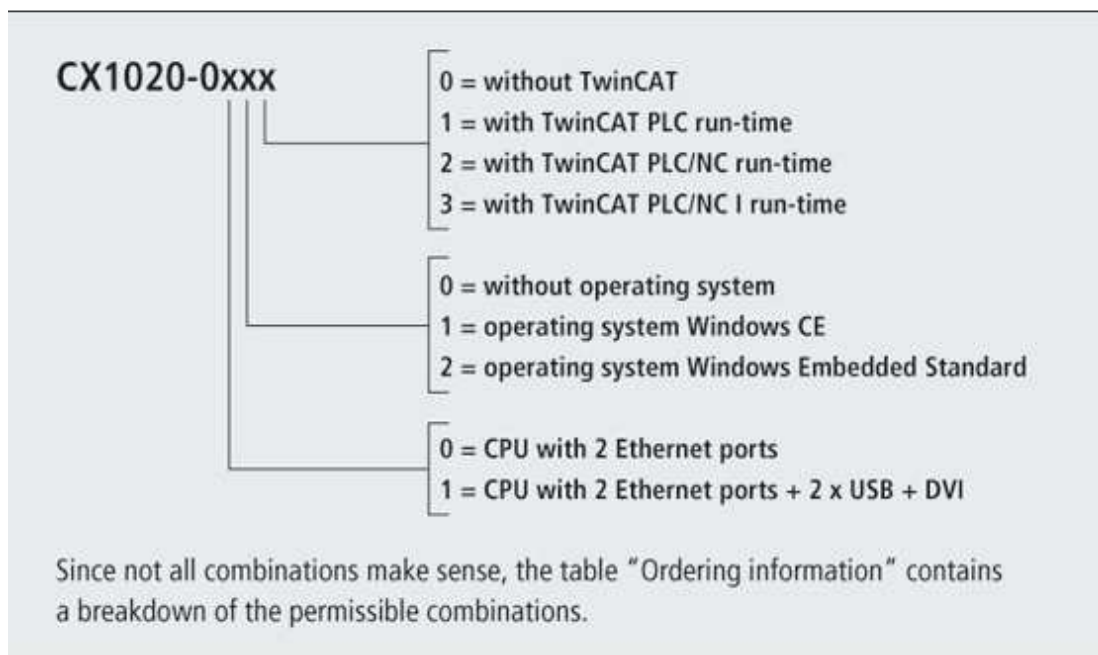


Fig.1.2 – Codice dispositivo CX1020

La programmazione può essere fatta in modo remoto tramite Ethercat: tale possibilità, adottata nel caso in cui il sistema sia dotato di "Windows CE.NET", permette di programmare il tutto tramite un computer fisso o portatile collegato al CX1020 tramite Ethernet (di rete o cavo crossover).

I programmi sono sviluppati sul portatile con una licenza standard del software TwinCAT per poi essere caricati sul dispositivo stesso.

La presenza di uno slot Compact Flash permette un'ulteriore possibilità di espandere la memoria nel caso in cui essa non fosse sufficiente permettendolo in modo agevole di sfilarla in caso di necessità.

Il CX 1020 è caratterizzato da una struttura modulare: infatti è costituito da vari moduli, ognuno con caratteristiche ben precise, con la possibilità di variare la configurazione dell'embedded PC a seconda delle esigenze di impiego.

Al modulo base possono essere aggiunti diversi altri tipi di moduli e terminali:

- Accoppiatori di Bus
- Terminali Bus
- Terminali EtherCAT
- Unità di alimentazione
- Altro (schede, switch, accessori)

Technical data	CX1020
Processor	Intel® Celeron® M ULV, 1 GHz clock frequency
Flash memory	64 MB Compact Flash card (optionally extendable)
Internal main memory	256 MB DDR RAM (expandable to 512 MB, 1 Gbyte)
Interfaces	2 x RJ 45 (Ethernet, internal switch)
Diagnostics LED	1 x power, 2 x LAN link/activity, TC status, 1 x flash access
Expansion slot	1 x Compact Flash type I+II insert with eject mechanism
Clock	internal battery-backed clock for time and date (battery exchangeable)
Operating system	Microsoft Windows CE or Microsoft Windows Embedded Standard
Control software	TwinCAT PLC run-time, NC PTP run-time, NC I run-time
System bus	16 bit ISA (PC/104)
Power supply	via system bus (through CX1100-xxxx power supply modules)
Max. power loss	11 W (including CX1020-N0xx system interfaces)
Dimensions (W x H x D)	96 mm x 112 mm x 98 mm
Weight	approx. 720 g
Operating/storage temperature	0...+50 °C/-25...+85 °C
Relative humidity	95 %, no condensation

Vibration/shock resistance	conforms to EN 60068-2-6/EN 60068-2-27/29
EMC immunity/emission	conforms to EN 61000-6-2/EN 61000-6-4
Protection class	IP 20

Alimentazione del sistema

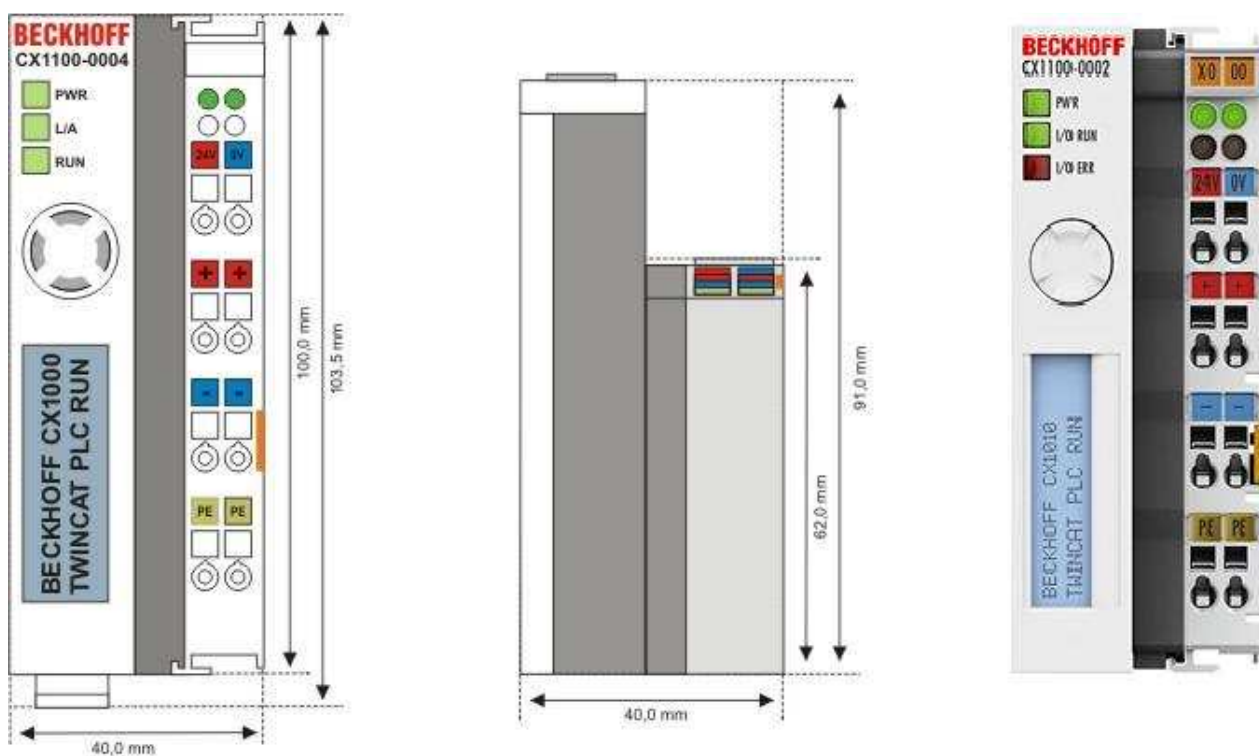


Fig.1.3 – CX1100-0004

L'alimentazione di tutti gli elementi del sistema è assicurata dal bus interno PC104, non richiedendo altre linee di alimentazione. I componenti del CX1100, però, offrono altre caratteristiche che vanno oltre alla singola funzionalità di alimentazione:

- La presenza di un display LCD con due linee di 16 caratteri permette la visualizzazione di messaggi di utente e di sistema.

- Possibilità di collegare Bus Terminals o moduli Fieldbus Box permette di creare un sistema di controllo molto flessibile ed espandibile a una grande varietà di livelli di segnale.
- Segnali I/O locali sono collegati tramite l'alimentatore variabile CX1100- 004 ai terminali EtherCAT.
- Con CX110-0004 i dati I/O sono memorizzati direttamente nella memoria principale della CPU , non rendendosi più necessaria una DPRAM.
- L'unità CX1100-0004 di alimentazione per i terminali EtherCAT può essere solo collegata congiuntamente al modulo di base CX1020 CPU.

L'alimentatore CX1100-0004 offre un'interfaccia diretta tra il CX1020 e i terminali EtherCAT. La combinazione di CX1020 con EtherCAT e TwinCAT consente tempi di ciclo e di risposta minori di 1 millisecondo.

Il modulo CPU è disponibile in diverse varianti le quali consistono in:

- Memoria interna di configurazione: esistono tre diverse scelte tra 64 MB di flash/256 DDR RAM(standard), 512 MB di RAM oppure 1GB di RAM
- Interfaccia di configurazione del sistema: all'interfaccia Ethernet e alle due porte RJ 45, che vi sono già presenti, possono essere aggiunte un DVI e due interfacce USB.
- Sistema operativo: "Microsoft Windows CE.NET" o "Microsoft Windows XP Embedded"

Sotto è riportata la scheda tecnica dell'unità di alimentazione CX1100-000X

Technical data	CX1100-0001	CX1100-0002	CX1100-0003	CX1100-0004
Power supply	24 V DC (-15 %/+20 %)			
Dielectric strength	500 V (supply/internal electronics)			
E-bus connection	–	–	–	yes (adapter terminal)
K-bus connection	–	yes (adapter terminal)	yes (adapter terminal)	–
IP-Link connection	–	–	yes	–
Current supply E-bus	–	–	–	2 A
Current supply K-bus	–	2 A	2 A	–
Type of connection	1 x open style connector, 5-pin	spring-loaded technique (adapter terminal)	spring-loaded technique (adapter terminal)	spring-loaded technique (adapter terminal)

NOVRAM	8 kbytes			
Display	FSTN display 2 lines x 16 characters of text, illuminated			
I/O-DPRAM	–	4 kbytes	4 kbytes	–
Diagnostics LED	1 x PWR	1 x PWR, 1 x I/O Run, 1 x I/O Err	1 x PWR, 1 x I/O Run, 1 x I/O Err	1 x PWR, 1 x L/A, 1 x Run
Max. power consumption	2.5 W	3.5 W	4 W	3.5 W
Dimensions (W x H x D)	45 mm x 100 mm x 91 mm	39 mm x 100 mm x 91 mm	58 mm x 100 mm x 91 mm	39 mm x 100 mm x 91 mm
Weight	approx. 180 g	approx. 200 g	approx. 265 g	approx. 205 g
Operating/storage temperature	0...+55 °C/-25...+85 °C			
Relative humidity	95 %, no condensation			
Vibration/shock resistance	conforms to EN 60068-2-6/EN 60068-2-27/29			
EMC immunity/emission	conforms to EN 61000-6-2/EN 61000-6-4			
Protection class	IP 20			

Terminali EtherCAT

Il CX1020 Embedded PC è stato sviluppato per supportare e rendere ottimale l'integrazione con EtherCAT. I terminali I/O EtherCAT sono progettati in modo da permettere l'integrazione ottimale tra il dispositivo e ognuno di essi è fornito di guide per il montaggio che ne permettono un facile collegamento.

L'unità di alimentazione CX1100-0004 ha la funzione di accoppiatore tra i terminali EtherCAT e l'Embedded PC. I collegamenti elettrici vengono realizzati automaticamente montando i vari componenti, in particolare per ogni terminale sono presenti sei contatti a molla che consentono il trasferimento dei dati attraverso E-Bus e l'approvvigionamento di potenza (fino a 24 Volt) o per tensioni superiori tramite morsetti di alimentazione. I terminali a nostra disposizione sono:

Beckhoff EL1202

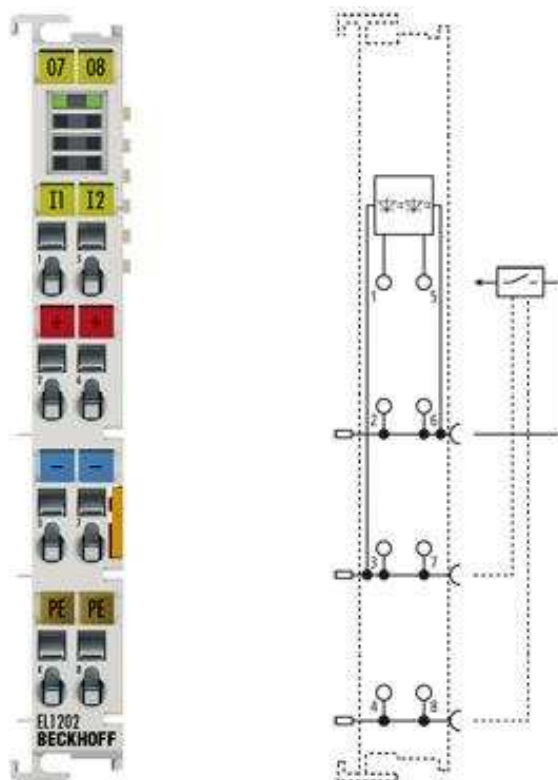


Fig.1.4 – EL1202

Il terminale di input digitale EL1202 è caratterizzato da due canali, con tensione di ingresso a 24 Volt DC, Ton / Toff 1 microsecondo.

Il terminale acquisisce i segnali di controllo in forma binaria dal livello di processo e li trasmette, in una forma isolate elettricamente, all'unità di automazione di più alto livello .

Adatto per i segnali particolarmente veloci grazie al suo basso ritardo di input, supporta, inoltre, la funzionalità a clock distribuiti, vale a dire che i dati possono essere controllati in modo sincrono con quelli degli altri dispositivi mantenendo comunque elevate le prestazioni: infatti, la precisione di tutto il sistema è minore di un 1 μ s.

Technical data	EL1202 ES1202
Number of inputs	2
Nominal voltage	24 V DC (-15 %/+20 %)
“0“ signal voltage	-3...+5 V (similar to EN 61131-2, type 3)
“1“ signal voltage	11...30 V (similar to EN 61131-2, type 3)
Input current	typ. 3 mA (similar to EN 61131-2, type 3)
Input delay TON/TOFF	< 1 μ s
Distributed clocks	yes
Current consumption power contacts	typ. 6 mA + load
Current consumption E-bus	typ. 110 mA
Electrical isolation	500 V (E-bus/field potential)
Bit width in the process image	2 inputs
Configuration	no address or configuration setting
Special features	DC can be activated, see documentation
Weight	approx. 55 g
Operating/storage temperature	0...+55 °C/-25...+85 °C
Relative humidity	95 %, no condensation

Vibration/shock resistance	conforms to EN 60068-2-6/EN 60068-2-27/29
EMC immunity/emission	conforms to EN 61000-6-2/EN 61000-6-4
Protect. class/installation pos.	IP 20/variable
Approvals	CE, UL, Ex

Beckhoff EL1252

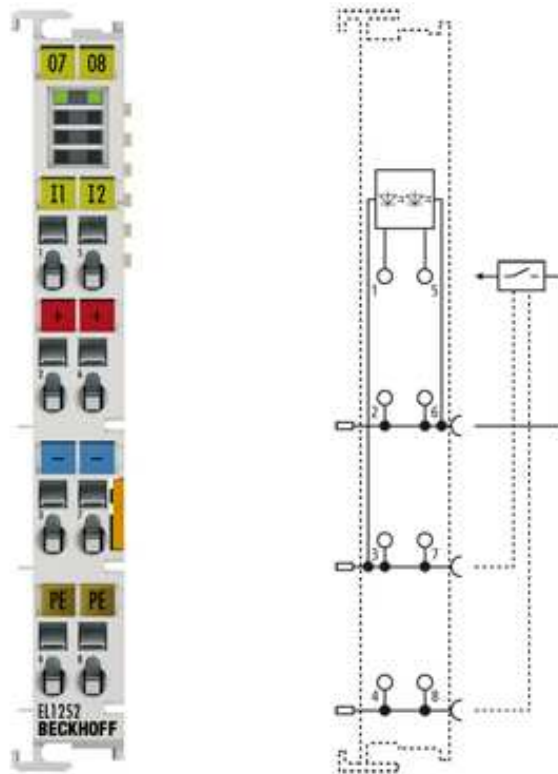


Fig.1.5 – EL1252

Il terminale di input digitale EL1252 acquisisce segnali di controllo binari dal livello di processo e li trasmette, in forma isolata elettricamente, al controllore. I segnali sono dotati di un timestamp che identifica il momento del cambio di fronte con una risoluzione di 1 ns. Questa tecnologia consente ai segnali di essere tracciati esattamente nel tempo e sincronizzati con il clock distribuiti in tutto il sistema.

Technical Data	EL1252 ES1252
Number of inputs	2
Nominal voltage	24 V DC (-15 %/+20 %)

“0“ signal voltage	-3...+5 V (similar to EN 61131-2, type 3)
“1“ signal voltage	11...30 V (similar to EN 61131-2, type 3)
Input current	typ. 3 mA (similar to EN 61131-2, type 3)
Resolution time stamp	1 ns (channel 0/1)
Precision of time stamp in the terminal	10 ns (+ input delay)
Distributed clock precision	<< 1 μ s
Input delay TON/TOFF	< 1 μ s
Time resolution signal	1 ns
Current consumption power contacts	typ. 6 mA + load
Current consumption E-bus	typ. 110 mA
Electrical isolation	500 V (E-bus/field potential)
Bit width in the process image	2 inputs + 36 byte time stamp
Special features	time stamp, latch last edge
Weight	approx. 55 g
Operating/storage temperature	0...+55 °C/-25...+85 °C
Relative humidity	95 %, no condensation
Vibration/shock resistance	conforms to EN 60068-2-6/EN 60068-2-27/29
EMC immunity/emission	conforms to EN 61000-6-2/EN 61000-6-4

Beckhoff EL1262

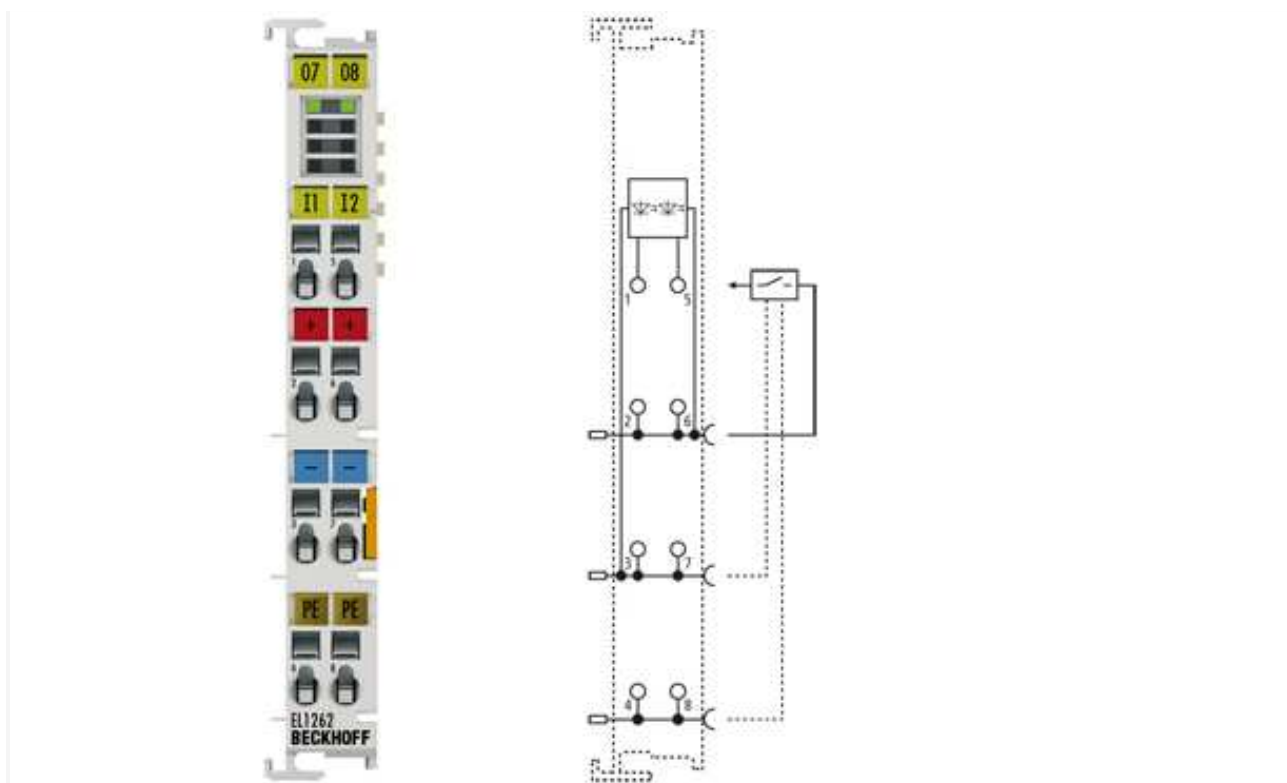


Fig.1.6 – EL1262

Il terminale di ingresso digitale EL1262 acquisisce segnali di controllo binari dal livello di processo e li trasmette, in forma isolata elettricamente, al controllore. I segnali possono essere sovracampionati con un valore intero N dando così l'opportunità di una risoluzione temporale N volte migliore di un normale terminale. Tutto ciò è possibile poiché il dispositivo supporta la funzionalità di clock distribuiti offerta dalla tecnologia.

Technical data	EL1262 ES1262
Connection technology	4-wire
Specification	similar to EN 61131-2, type 3, "0": -3...5 V DC, "1": 11...30 V DC, typ. 3 mA input current

Number of inputs	2
Nominal voltage	24 V DC (-15 %/+20 %)
“0” signal voltage	-3...+5 V (EN 61131-2, type 1)
“1” signal voltage	11...30 V (EN 61131-2, type 1)
Input current	typ. 3 mA
Input filter	typ. < 1 μ s
Oversampling factor	n = integer multiple of the cycle time, 1...1,000, see documentation
Precision of time stamp in the terminal	10 ns (+ input delay)
Distributed clock precision	<< 1 μ s
Sampling rate	max. 1 Msample/s
Distributed clocks	yes
Time resolution signal	\geq 1 μ s, adjustable
Current consumption power contacts	typ. 20 mA + load
Current consumption E-bus	typ. 70 mA
Electrical isolation	500 V (E-bus/field potential)
Bit width in the process image	n x 2 inputs + 64 bit CycleCounter/latch
Special features	oversampling
Weight	approx. 60 g
Operating/storage temperature	0...+55 $^{\circ}$ C/-25...+85 $^{\circ}$ C
Relative humidity	95 %, no condensation
Vibration/shock resistance	conforms to EN 60068-2-6/EN 60068-2-27/29

Beckhoff EL2202

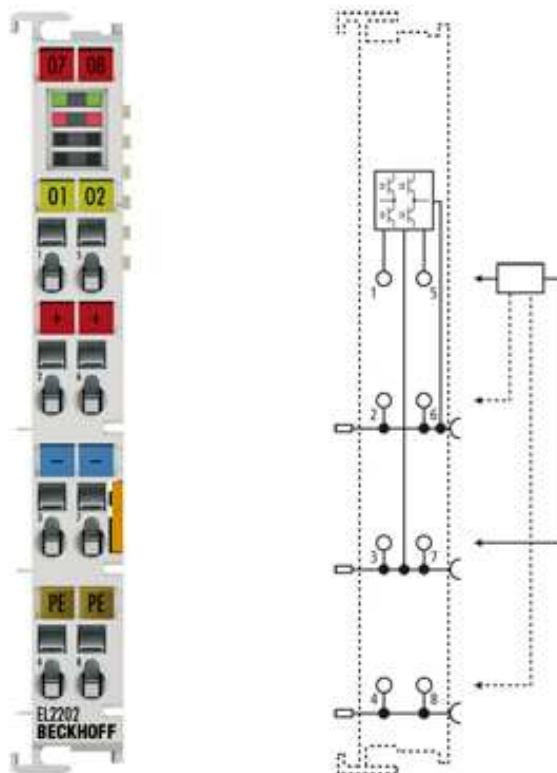


Fig.1.7 – EL2202

Il terminale di output digitale EL2202 collega i segnali in uscita dall'unità di controllo agli attuatori di processo che devono essere controllato. Questo terminale beneficia di un piccolissimo ritardo in uscita che lo rende adatto ad applicazioni che richiedono un'elevata velocità. Supporta la funzionalità a clock distribuiti. Il terminale, inoltre, è dotato di un push/pull che permette di collegare in uscita 24 V, 0 V oppure un'elevata impedenza.

Technical data	EL2202 ES2202
Connection technology	4-wire
Number of outputs	2
Rated load voltage	24 V DC (-15 %/+20 %)

Load type	ohmic, inductive, lamp load
Distributed clocks	– (EL2202-0100 yes, see documentation)
Distributed clock precision	$\ll 1 \mu\text{s}$
Max. output current	0.5 A (short-circuit-proof in push operation) per channel
Short circuit current	typ. $< 1.5 \text{ A}$
Reverse voltage protection	yes
Breaking energy	$< 150 \text{ mJ/channel}$
Switching times	typ. TON: $< 1 \mu\text{s}$, typ. TOFF: $< 1 \mu\text{s}$
Output stage	push-pull, high-ohmic
Current consumption E-bus	typ. 130 mA
Electrical isolation	500 V (E-bus/field potential)
Current consumption power contacts	typ. 30 mA + load
Bit width in the process image	2 outputs, 2 bit enable tri-state
Configuration	no address or configuration setting
Special features	Can be converted to DC version EL2202-0100. Outputs can be connected in high-resistance mode.
Weight	approx. 55 g
Operating/storage temperature	0...+55 °C/-25...+85 °C
Relative humidity	95 %, no condensation
Vibration/shock resistance	conforms to EN 60068-2-6/EN 60068-2-27/29
EMC immunity/emission	conforms to EN 61000-6-2/EN 61000-6-4
Protect. class/installation pos.	IP 20/variable

Beckhoff EL2252

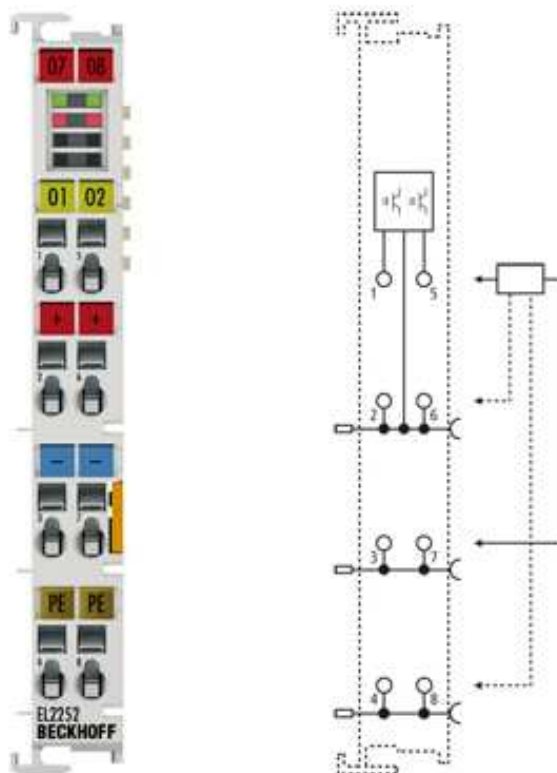


Fig.1.8 – EL2252

Il terminale di output digitale EL2252 collega i segnali in uscita dall'unità di controllo agli attuatori di processo che deve essere controllato. Questo terminale beneficia di un piccolissimo ritardo in uscita che lo rende adatto ad applicazioni che richiedono un'elevata velocità. Supporta la funzionalità a clock distribuiti. Deve essere specificato il tempo in cui le due uscite devono commutare: tale informazione, sotto forma di sequenza di bit, viene trasferito come dato di processo. Di conseguenza, il terminale attende fino all'istante in cui avviene la commutazione prima di prendere in considerazione altri segnali.

Technical data	EL2252 ES2252
Connection technology	4-wire
Number of outputs	2
Rated load voltage	24 V DC (-15 %/+20 %)
Load type	ohmic, inductive, lamp load

Resolution time stamp	1 ns
Precision of time stamp in the terminal	10 ns (+ output circuit delay)
Distributed clocks	yes
Distributed clock precision	$\ll 1 \mu\text{s}$
Output delay through 24 V power section	typ. $< 1 \mu\text{s}$
Max. output current	0.5 A (short-circuit-proof) per channel
Short circuit current	typ. $< 1.5 \text{ A}$
Reverse voltage protection	yes
Current limitation	typ. 1.5 A
Breaking energy	$< 150 \text{ mJ/channel}$
Switching times	typ. TON: $< 1 \mu\text{s}$, typ. TOFF: $< 1 \mu\text{s}$
Output stage	push-pull
Current consumption E-bus	typ. 130 mA
Electrical isolation	500 V (E-bus/field potential)
Current consumption power contacts	typ. 30 mA + load
Bit width in the process image	8 bit output (ch. 1 + ch. 2), 9 byte time stamp
Special features	Outputs can be connected in high-resistance mode, short-circuit-proof.
Weight	approx. 60 g
Operating/storage temperature	0...+55 °C/-25...+85 °C
Relative humidity	95 %, no condensation
Vibration/shock resistance	conforms to EN 60068-2-6/EN 60068-2-27/29

Beckhoff EL6751

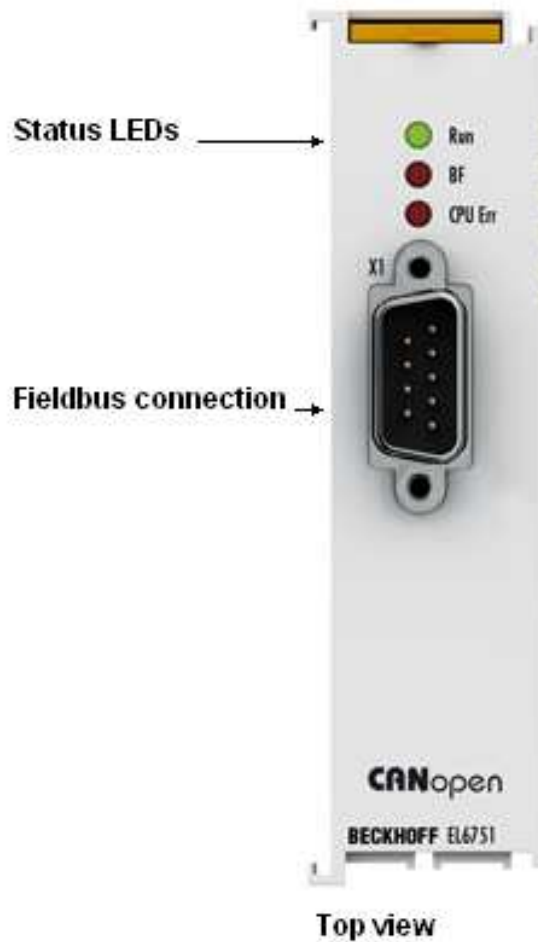


Fig.1.8 – EL6751

Il terminale CANopen EL6751 , che può funzionare sia da master che da slave, consente l'integrazione dei vari dispositivi CANopen.

Il terminale ha una potente implementazione del protocollo con molte caratteristiche:

- Tutti i tipi di comunicazione CANopen sono supportati
- Sincronizzazione con il PC
- Potenti interfacce e strumenti di diagnostica
- Possibilità di monitorare le funzionalità e di visualizzare online lo stato del bus

Technical data	EL6751
Technology	CANopen master terminal
Fieldbus	CANopen
Number of fieldbus channels	1
Data transfer rates	10, 20, 50, 100, 125, 250, 500, 800, 1,000 kbaud
Interfaces	D-sub connector, 9-pin according to CANopen specification, galvanically decoupled
Communication	CANopen network master and CANopen manager, optionally CANopen slave
Hardware diagnosis	status LEDs
Current consumption E-bus	typ. 300 mA
Special features	status LEDs, CANopen network master, CANopen Manager, supports RAW-CAN
Weight	approx. 70 g
Operating/storage temperature	0...+55 °C/-25...+85 °C
Relative humidity	95 %, no condensation
Vibration/shock resistance	conforms to EN 60068-2-6/EN 60068-2-27/29
EMC immunity/emission	conforms to EN 61000-6-2/EN 61000-6-4
Protect. class/installation pos.	IP 20/variable
Approvals	CE, UL, Ex

Beckhoff EK1110

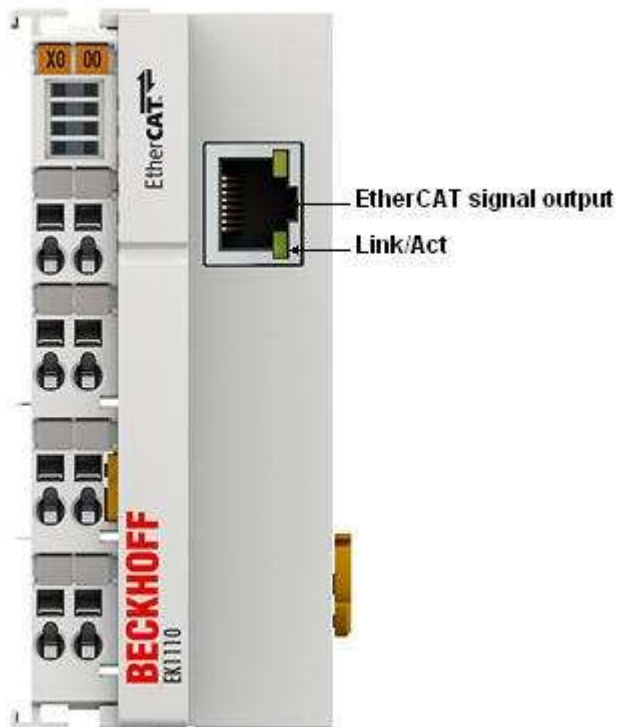


Fig.1.9 – EK1110

L'unità Beckhoff EK1110 consente di collegare un cavo di Ethernet con connettore RJ 45 al sistema, permettendo di prolungare la rete EtherCAT per mezzo di un filo elettrico isolato fino a 100 m.

Technical data	EK1110
Task within EtherCAT system	conversion of the E-bus signals to 100BASE-TX Ethernet for extension of the EtherCAT network
Data transfer medium	Ethernet/EtherCAT cable (min. CAT 5), shielded
Distance between stations	100 m (100BASE-TX)

Protocol	any EtherCAT protocol
Delay	approx. 1 μ s
Data transfer rates	100 Mbaud
Configuration	not required
Bus interface	1 x RJ 45
Power supply	from E-bus
Current consumption E-bus	typ. 125 mA
Electrical isolation	500 V (supply voltage/Ethernet)
Weight	approx. 50 g
Operating/storage temperature	0...+55 °C/-25...+85 °C
Relative humidity	95 %, no condensation
Vibration/shock resistance	conforms to EN 60068-2-6/EN 60068-2-27/29
EMC immunity/emission	conforms to EN 61000-6-2/EN 61000-6-4
Protect. class/installation pos.	IP 20/variable
Approvals	CE, UL, Ex

Capitolo 2

Software

Windows Embedded CE



Fig. 2.0 – Logo Windows Embedded

Windows Embedded offre una suite completa di sistemi operativi e strumenti che possono aiutare i produttori di dispositivi a ridurre i tempi di commercializzazione e a migliorare le prestazioni dei dispositivi connessi.

Windows Embedded CE è un sistema operativo e una piattaforma di sviluppo che offre un sistema operativo a 32 bit, real-time e modulare, un kernel unificato e potenti strumenti di sviluppo embedded, progettato per dispositivi intelligenti, che consentono alle organizzazioni di connettersi all' infrastruttura IT, di migliorare l'efficienza del personale e di aumentare la soddisfazione dei clienti. I prodotti Windows Embedded sono stati utilizzati per realizzare migliaia di dispositivi embedded, dalle apparecchiature a ultrasuoni portatili ai dispositivi GPS, dai bancomat ai dispositivi che alimentano grandi macchinari da costruzione. Attraverso una gamma completa di funzionalità, strumenti di facile utilizzo, kit di valutazione gratuiti e l'accesso a una vasta rete di assistenza, Windows Embedded consente di ridurre i tempi di commercializzazione e di contenere i costi legati allo sviluppo embedded.

Windows Embedded offre una suite completa di sistemi operativi e strumenti che possono aiutare i produttori di dispositivi a ridurre i tempi di commercializzazione e a migliorare le prestazioni dei dispositivi connessi.

I principali clienti includono OEM, grandi imprese e provider di servizi che scelgono Windows Embedded per una perfetta integrazione con l'infrastruttura Microsoft e i servizi che includono:

- Gestione, sicurezza e identità: aggiornamenti, upgrade, manutenzione remota, provisioning

- Sincronizzazione dei dati: device-cloud, device-PC, device-device
- Profilo di utilizzo
- Servizi di posizione
- Servizi pubblicitari
- Applicazioni line-of-business e Business Intelligence
- Segnali del device: accesso a dati relativi ai servizi e alle funzionalità del dispositivo

Windows Embedded CE, a differenza di molti altri sistemi operativi disponibili, è stato creato in maniera tale da non essere vincolato ad una specifica architettura del processore o implementazione hardware: l'unica limitazione consta nell'utilizzo di un processore a 32 bit.

La versione più recente, Windows Embedded CE 6.0 supporta quattro architetture di processore (ARM, MIPS, SH4, e x86) e un numero considerevole di implementazioni offerte dai vari produttori di processori.

Quest'ultima versione del sistema operativo Microsoft supporta vari sottoinsiemi di Microsoft Win32 API e diverse interfacce di programmazione. Come tutti i sistemi embedded che utilizzano OS Microsoft, anche il CX1020 di Beckhoff è pre-configurato con sistema operativo

Windows CE 6.0.

Tale sistema operativo offre allo sviluppatore una vasta gamma di opportunità, supporti e di tecnologie, garantendo sistemi rapidi per lo sviluppo di applicazioni quali:

- AYGShell API, che assicura la compatibilità con le applicazioni Windows Mobile.
- .NET Compact Framework 2.0 e 3.5, tra cui Active Template Library (ATL), Microsoft Foundation Classes (MFC), Windows Template Library (WTL), Standard Template Library (STL), ActiveSync, Exchange Server Client, Global Positioning System (GPS) driver, Speech API 5.0, Windows Messenger, Pocket Outlook Object Model (POOM), Extensible Markup Language (XML) e Microsoft SQL Server Compact 3.5.
- Simple Network Management Protocol (SNMP).
- 3,9 milioni di righe di codice sorgente, il 100 per cento del codice sorgente del kernel.
- Production Quality OAL (PQOAL), un set di librerie e il codice sorgente per la creazione dell'OAL.

Dispone delle tecnologie di comunicazione:

- Transmission Control Protocol/Internet Protocol (TCP/IP), IPv4, IPv6, Network Driver Interface Specification (NDIS) 5.1, Winsock 2.2, Internet Protocol security (IPsec) v4.
- Personal area network (PAN), local area network (LAN), wide area network (WAN), Bluetooth, 802.11.
- SOAP, Object EXchange (OBEX), Lightweight Directory Access Protocol (LDAP) client, Remote Desktop Protocol (RDP).
- VoIP, real-time communications (RTC), Session Initiation Protocol (SIP).

- Radio Interface Layer (RIL), support for Short Message Service (SMS), Wireless Application Protocol (WAP), support for Subscriber Identity Module (SIM) cards.
- Remote API (RAPI) and RAPI2, Point-to-Point Protocol over Ethernet (PPPoE), Telephony Application Programming Interface (TAPI), virtual private network (VPN).

Fornisce inoltre tecnologie Server-side e diverse applicazioni multimediali come DirectDraw, DirectShow, Direct3D, Windows Media Player, Windows Media Audio (WMA), MP3, Internet Explorer e un interfaccia DVD.

Microsoft Windows Embedded CE 6.0 è un sistema operativo real-time (ogni processo ha una scadenza temporale prefissata), che supporta il multitasking e gira su architetture di processori diversi, tra cui ARM, MIPS, x86 e SH4, operando nello spazio di indirizzamento virtuale di 4 gigabyte (GB). Il kernel di sistema utilizza i 2 GB di memoria virtuale superiori, mentre i rimanenti vengono utilizzati per il processo del utente attivo.

Windows Embedded CE 6.0 supporta fino a 32000 processi utente ma tale numero effettivo viene limitato dalle risorse di sistema.

Il kernel di Windows Embedded CE 6.0 interagisce con l'hardware attraverso la OAL(OEM Adaptation Layer), che nasconde l'implementazione del processore ed esegue l'inizializzazione

del hardware. La costruzione del sistema operativo è basata sulla memoria virtuale, che viene fornita dall'OS in modo flessibile ed efficace per gestire le limitate risorse di memoria fisica..

Per costruire applicazioni per Windows Embedded CE esistono due diverse tipologie di codice: quello nativo (native code) oppure "gestito" (managed code). Le applicazioni con "native code" possono essere costruite come sottoprogetti del progetto operativo oppure come singoli progetti e necessitano di un SDK. Le applicazioni con "managed code" possono essere costruite solo come applicazioni separate che non hanno bisogno di un SDK ma richiedono invece l'ambiente di esecuzione del dispositivo.

Per sviluppare delle applicazioni per Windows CE abbiamo bisogno di utilizzare un ambiente di sviluppo (ad esempio Visual Studio) e di compilare il codice su una workstation con determinate le seguenti caratteristiche minime di sistema:

- Microsoft Windows 2000 Professional con Service Pack 4 oppure Windows XP Professional con Service Pack 2 o superiori.
- Processore minimo da 933MHz (2 GHz raccomandato).
- RAM minimum da 512MB (1 GB raccomandato).
- 18 GB di spazio libero sul disco per l'installazione.
- 1 GB di spazio libero sul disco di sistema.

Microsoft Visual Studio 2008 Professional©



Fig. 2.2 – Logo Windows Visual Studio

Visual Studio è lo strumento di sviluppo Microsoft di punta per lo sviluppo su sistemi Microsoft e ambiente principale di tutta la linea di software per sviluppatori sul .NET Framework.

Nato negli anni '90 come editor per sviluppatori Visual Basic, ha visto la sua evoluzione con il passare degli anni passando dalla versione per sviluppo in C++ (1993) alle versioni 2002 e 2003 che supportavano le prime versioni del .NET Framework (1.0 e 1.1).

Nel 2005 poi, è stata rilasciata una nuova versione del prodotto in corrispondenza alla diffusione della versione 2.0 del .NET Framework, che ha segnato grossi cambiamenti architetturali su tutto l'ambiente di programmazione.

A due anni di distanza della versione 2005, Microsoft lancia una nuova versione dell'IDE, anche questa volta in parallelo con il rilascio della nuova edizione del .NET Framework: la 3.5. Nasce così Visual Studio 2008 (noto nelle fasi di sviluppo sotto il nome di "Orcas").

Sia il .NET Framework sia VS2008 presentano molte novità, in grado di aumentare la produttività degli sviluppatori e la qualità dei prodotti, grazie alle revisioni sul Framework e alle numerose semplificazioni a livello visuale.

Troviamo strumenti in grado di supportarci nella costruzione di nuove applicazioni e servizi, basati sulle più recenti tecnologie Microsoft. Possiamo sviluppare ad esempio:

- applicazioni Web (ASP.NET 3.5), e applicazioni Ajax-enabled
- servizi (Web services) e client WCF ,
- flussi di lavoro (Windows Workflow Foundation),
- applicazioni desktop di impatto con WPF, nei diversi linguaggi di programmazione contemplati dal .NET Framework (da VB.NET a C++)
- componenti per Office 2003-2007 (tramite VSTO).

Il vantaggio è sempre quello di avere tutto integrato in un ambiente unico con funzionalità di collaborazione per lo sviluppo di progetti in team.

Si ricorda in particolare il linguaggio di programmazione Visual c# presente all'interno del pacchetto Visual studio: C# è un linguaggio di programmazione appositamente progettato per la compilazione di un'ampia gamma di applicazioni per la piattaforma .NET Framework. Si tratta di un linguaggio semplice, potente, indipendente dai tipi e orientato a oggetti. In C# sono state introdotte diverse innovazioni che facilitano lo sviluppo rapido di applicazioni, mantenendo al tempo stesso l'espressività e l'eleganza tipiche dei linguaggi di tipo C.

Visual C# è un'implementazione del linguaggio C# realizzata da Microsoft. Come supporto per Visual Studio, in Visual C# vengono forniti un editor di codice completo, un compilatore, modelli di progetto, strumenti di progettazione, procedure guidate per la creazione di codice, un debugger semplice e potente e altri strumenti. La libreria di classi .NET Framework fornisce l'accesso a numerosi servizi del sistema operativo e ad altre classi utili e accuratamente progettate che consentono di velocizzare in modo significativo il ciclo di sviluppo.

Versioni

La versione attuale del prodotto è la 9.0, mentre per il .NET Framework siamo alla versione 3.5; al suo interno anche il compilatore di C# segna la versione 3.5, mentre quello di VB.NET la 9.0; i rispettivi linguaggi di sviluppo sono infatti presentati come C# 3.0 e VB 9, ASP.NET è 3.5. Inoltre si può notare l'inserimento di librerie già esistenti in precedenza come plug-in esterni, come ad esempio il pacchetto AJAX e o come il SQL Server Database Publishing Wizard.

Molti invece risultano i componenti visuali aggiunti, sia per quanto riguarda lo sviluppo Web, sia per lo sviluppo di applicazioni Windows; in sostanza per supportare tutte le novità presenti all'interno del .NET Framework 3.5. Sono stati creati ex-novo dei designer grafici per la creazione e la modifica di pagine Web, per lo sviluppo di applicazioni WPF e per la creazione del modello ad oggetti per i nostri software mappato direttamente sulle tabelle del database correlato.

Inoltre, troviamo nuovi wizard per la migrazione dei progetti e una comoda procedura guidata per l'importazione delle impostazioni da Visual Studio 2005.

Come nell'edizione 2005 sono disponibili sia versioni a pagamento (Professional Edition, Team Suite, Team Foundation Server e Test Load Agent), sia versioni Express totalmente gratuite, che possono essere liberamente scaricate ed usate.

Visual Studio .NET offre un certo numero di designer visuali per il supporto allo sviluppo con le principali tecnologie legate al .NET Framework. I principali sono:

- **Windows Form designer** - per la creazione di applicazioni windows form in maniera visuale;
- **WPF designer** - per la creazione di applicazioni Windows Presentation Foundation (designer aggiunto nella versione 2008 dell'IDE);
- **ASP.NET designer** - un editor WYSIWYG per la costruzione di Web form (nei capitoli successivi vedremo in dettaglio le funzionalità legate a questo particolare designer e allo sviluppo di applicazioni Web ASP.NET);

- **CSS designer** - editor per la creazione e la manutenzione di fogli di stile CSS;
- **Component designer** - editor per lo sviluppo di componenti lato server;
- **XML designer** - editor per la generazione e la manutenzione di file XML e di file in formato derivato dall'XML, come fogli di stile XSLT, schemi XSD. Questo editor viene arricchito di ulteriori funzionalità visuali per la costruzione di DataSet per le nostre applicazioni;
- **Workflow designer** - editor per la costruzione in maniera grafica di flusso di lavoro basati sul framework di Windows Workflow Foundation.
- **Database designer** - editor che permette la creazione di tabelle, viste, funzioni e stored procedures in database SQL Express 2005.
- **Object-Relational designer** - editor per la costruzione del mapping tra gli oggetti dell'applicazione e gli oggetti del database, detto Linq to SQL (funzionalità aggiunta nella versione 2008 a fronte della nascita del linguaggio Linq, di cui vedremo le principali funzionalità nei capitoli successivi).
- **Resource designer** - editor per la manutenzione dei file di risorse (.resx) legati ai propri progetti.
- **Settings designer** - editor per la creazione e la gestione di proprietà chiave-valore da utilizzare all'interno delle proprie applicazioni Windows Form o WPF.

In generale la parte centrale dell'IDE è dedicata all'editing e fornisce funzionalità comuni a tutti i designer sia testuali che visuali; prima fra tutte risulta l'intellisense. Tramite questo meccanismo è possibile essere supportati durante la stesura del codice delle proprie applicazioni da un menu contestuale che in base alla posizione corrente presenta all'utente le possibili soluzioni da attuare.

Questo strumento risulta a dir poco fondamentale, in quanto evita la lettura continua della documentazione delle classi del .NET Framework da utilizzare, proponendo la lista completa di tipi, proprietà, metodi ed eventi per le classi dichiarate all'interno dell'applicazione e per quelle degli assembly esterni referenziati al progetto.

Interessante è anche la possibilità di inserire dei frammenti di codice sorgente preconfezionati (Code Snippets) per snellire le operazioni più ripetitive. Esistono snippet per cicli, dichiarazioni di proprietà, e costrutti come try e catch, using, etc, che vengono richiamati digitando la parola chiave associata e premendo "Tab" sulla tastiera.

In aggiunta, ogni sviluppatore può creare i propri snippet di codice ed utilizzarli all'interno delle proprie applicazioni. L'elenco degli snippet è visualizzabile scegliendo l'opzione "Code Snippets Manager" presente all'interno del menu "Tools" (o digitando lo shortcut Ctrl+K+B).

Sono state pensate altre due funzioni molto utili (nel menu contestuale che appare cliccando col tasto destro sull'editor dei contenuti):

- **Refactor** - che permette all'utente di modificare i nomi in modo consistente tra le diverse parti del progetto, di estrarre un frammento di codice e formare un nuovo metodo, incapsulare campi e rimuovere o riordinare i parametri in un metodo.
- **Surround with** - che permette all'utente di selezionare un blocco particolare di codice e circondarlo da un particolare statement (scelto dalla lista che propone il menu stesso) come un blocco try e catch, una direttiva using, cicli e blocchi condizionali.

Le finestre degli strumenti sono poste intorno alla parte centrale e forniscono allo sviluppatore informazioni sul contesto corrente, permettono un rapido accesso ai file della soluzione o al database, o ancora forniscono informazioni sulle classi utilizzate. Possiamo spostare queste finestre a piacimento e ottenere la nostra configurazione ideale, inoltre possiamo attivarle o disattivarle attraverso combinazioni di tasti.

Requisiti di Sistema

REQUISITI SOFTWARE

Visual Studio 2008 può essere installato nei seguenti sistemi operativi:

- Windows XP (x86) con Service Pack 3 - tutte le edizioni eccetto Starter Edition
- Windows Vista (x86 e x64) con Service Pack 1 - tutte le edizioni eccetto Starter Edition
- Windows 7 (x86 e x64)
- Windows Server 2003 (x86 e x64) con Service Pack 2
- Windows Server 2003 R2 (x86 e x64)
- Windows Server 2008 (x86 e x64) con Service Pack 2
- Windows Server 2008 R2 (x64)

Architetture supportate:

- 32-Bit (x86)
- 64-Bit (x64)

REQUISITI HARDWARE

- Computer dotato di un processore con una velocità minima di 1,6 GHz
- 1024 MB RAM
- 3 GB di spazio libero su disco rigido
- Unità disco rigido da 5400 giri al minuto
- Scheda video con DirectX 9 e risoluzione minima da 1280 x 1024
- Unità DVD-ROM

TwinCAT

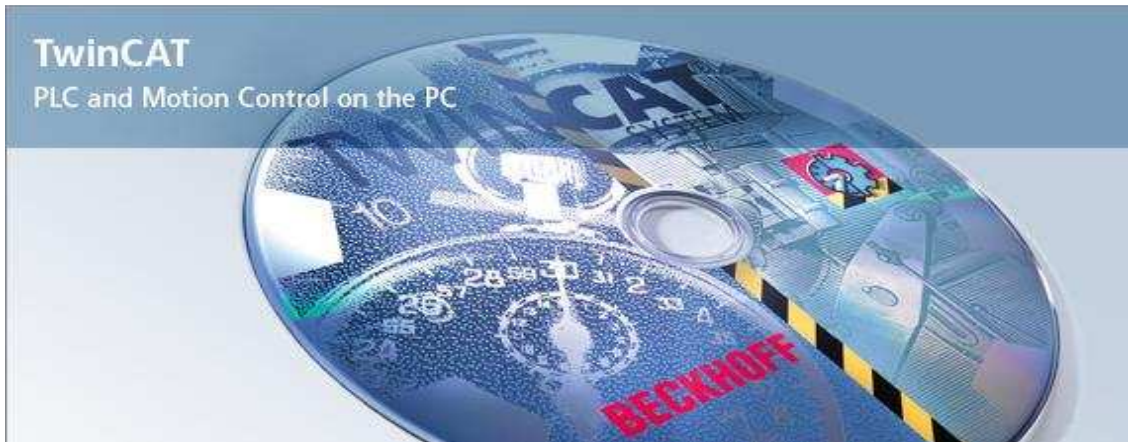


Fig. 2.3 – Logo TwinCAT

Il software TwinCAT (The Windows Control and Automation Technology) permette di convertire ogni tipo di pc compatibile, in particolare PC industriali e PC embedded Beckhoff, in un controllore "real-time". L'architettura TwinCAT consiste in un sistema "run-time" che esegue programmi di controllo in "real-time" disponendo servizi di programmazione, diagnosi e configurazione, che consentono lo scambio di informazioni con programmi Microsoft per la visualizzazione dei dati o per l'esecuzione stessa dei comandi.

TwinCAT sostituisce i convenzionali controllori PLC e NC / CNC e dispositivi che funzionano con:

- hardware PC compatibile
- sistemi di programmazione e di run-time opzionalmente insieme su un PC o separati
- embedded IEC 61131-3 software per PLC, software NC e software CNC in WindowsNT/2000/XP, Windows 7, NT / XP Embedded, CE
- collegamento a tutti i comuni fieldbus
- interfacce utente per comunicazione dati e gli altri programmi per mezzo di standard aperti Microsoft (OPC, OCX, DLL, ecc)

TwinCAT è caratterizzato da sistemi run-time che eseguono programmi di controllo in tempo reale e da ambienti di sviluppo per la programmazione, diagnostica e configurazione. Tutti i programmi Windows, per esempio, come quelli di visualizzazione o Office, possono accedere ai dati tramite interfacce TwinCAT.

Il pacchetto software TwinCAT è formato da vari sottoprogrammi specifici per varie applicazioni:

- TwinCAT PLC : permette la scrittura di programmi multi-PLC su PC in accordo con lo standard IEC 61131-3.
- TwinCat NC PTP: permette il posizionamento degli assi con tecnica "point-to-point".
- TwinCAT NC I : permette il posizionamento degli assi con tecnica di interpolazione in tre dimensioni.
- TwinCAT CNC : offre soluzioni CNC ad alte prestazioni per problemi complessi.
- TwinCAT I/O : connette gli ingressi e le uscite con i programmi windows.

La versione base del programma è formata da applicazioni di configurazione (TwinCAT System Manager, TwinCAT System Control) e dal sottoprogramma TwinCAT I/O, che non richiedono licenza Beckhoff. Tale sarebbe indispensabile per usufruire degli altri software che forniscono applicazioni di controllo più avanzate.

TwinCAT System Manager



Fig. 2.4 – Logo TwinCAT System Manager

TwinCAT System Manager è lo strumento di configurazione centrale del sistema TwinCAT: è proprio qui che gli ingressi e le uscite delle task del software e quelle fisiche dei fieldbus collegati sono gestiti. Le informazioni di I / O delle singole task vengono lette e inserite in TwinCAT System Manager.

Da qui, i fieldbus installati e i relativi moduli sono a loro volta descritti. Gli input logici e fisici e gli output sono assegnati l'uno all'altro, collegando le variabili del task del software e le variabili di bus di campo: in e output di ogni task possono essere scambiati, in modo ciclico, con output e input di un'altra task mantenendo, comunque, l'integrità dei dati.

Le più piccole unità collegabili sono le variabili le quali sono sempre e in modo preciso assegnate all'immagine di processo, il tutto in maniera precisa un'immagine di processo: TwinCAT System Manager produce assegnamenti tra le task e i dispositivi fieldbus e tra le task stesse nel caso in cui queste abbiano variabili mutuamente collegate. Mentre si definiscono le task del sistema non è necessario conoscere il tipo e la struttura di quest'ultime o dei fieldbus usati in quanto la programmazione è effettuata con l'aiuto delle variabili logiche locali definite e collegate con altre dal software TwinCAT System Manager: quest'ultimo collega le variabili logiche con le altre variabili, corrispondenti agli input e output fisici sul fieldbus o a variabili logiche di altre task.

TwinCAT System Manager supporta tutti i fieldbus commercializzati e qualche altra interfaccia per Pc come:

- Beckhoff Lightbus
- Profibus DP, MC (DP-V2)
- Interbus
- CANopen
- SERCOS
- DeviceNet
- Ethernet
- EtherCAT
- Beckhoff Real-Time Ethernet
- PC printer port (8 inputs and 8 outputs on TTL basis)
- Serial Bus Coupler BK8100 to COM
- USB Bus Coupler BK9500 and USB Control Panel interface (CPx8xx)
- Dual-ported memory interface (DPRAM) for PC cards
- NOVRAM (Non-Volatile RAM)
- System Management Bus (SMB) for monitoring of PC hardware like fans, temperatures, etc...

Configurazione di TwinCAT System Manager

Per aprire il software si deve cliccare il bottone “start” di Windows e scegliere, alla voce “TwinCAT System”, “TwinCAT System Manager” oppure sulla barra delle applicazioni, cliccare sopra alla corrispondente icona e selezionare “System Manager” dal Pop-up mostrato qui a fianco.

Nella finestra principale di TwinCAT System Manager (v2.11), sotto riportata, si noti il campo con lo sfondo rosso nella parte bassa destra del riquadro: esso segnala che il System Manager è collegato ad un sistema remoto mentre il testo in questo box definisce il nome di tale sistema.

Il doppio click apre la finestra di dialogo "Choose Target System" tramite cui si riesce ad importare, in modo quasi del tutto automatico, l'intera configurazione del sistema.



Fig. 2.4 – Finestra Popup

L'attuale nome del sistema è mostrato nella barra di titolo in cima alla finestra, Se il campo nell'angolo in basso a destra è grigio come il resto del frame della finestra significa che il System Manager è collegato al sistema target locale.

Il campo con lo sfondo blu segnala che il sistema remote è attualmente in "Config Mode" (modalità di configurazione) ma può diventare a sua volta verde ("Running") o giallo (connection establishment / Timeout) il quale indica lo stato del collegamento tra software e hardware. La finestra a sfondo blu segnala che il sistema remoto è attualmente in "Config Mode" ma può anche passare ad un differente stato e, riferendoci alla figura riportata, potrebbe tranquillamente passare, per esempio, allo stato "Running".

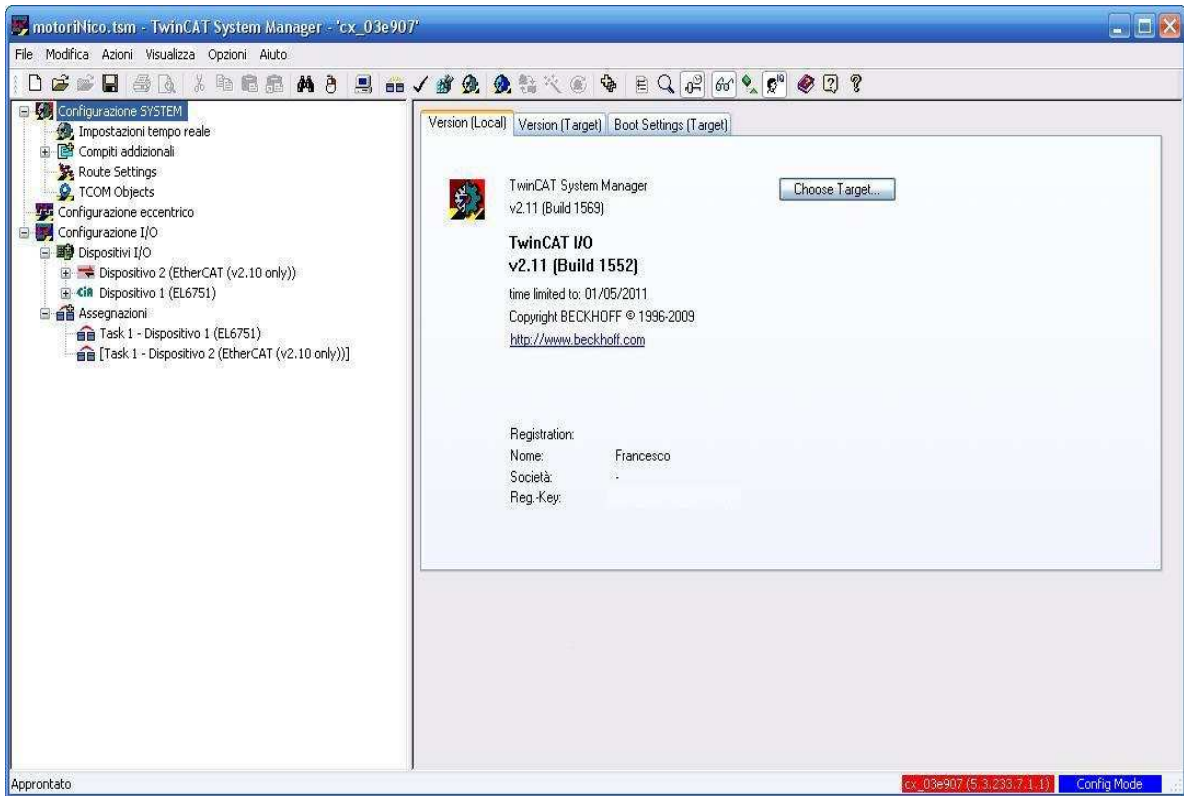



Fig. 2.5 – Finestra principale


Si osservi l'immagine allegata:





Fig. 2.6 – Barra degli strumenti

Molte delle opzioni mostrate sono già note al lettore in quanto presenti in innumerevoli programmi presenti in ambiente Windows. Ci soffermiamo in particolar modo su alcuni command button:

- **Set/Reset TwinCAT to Run Mode** 

Fa partire o riavviare il sistema locale TwinCAT con la configurazione correntemente attivata
Questa funzionalità è anche possibile attraverso l'interfaccia di comando senza aprire il System Manger.
- **Set/Reset TwinCAT to Config Mode** 

Fa partire o riavviare la configurazione in remoto di TwinCAT

- Reload Devices 
Ricerca la configurazione I/O del sistema target, mostrandola nel System Manager
- Choose Target System 
Con questo oggetto si è in grado di selezionare un appropriato sistema target dove bisogna fare la configurazione. Tale possibilità è selezionabile, inoltre, nel menù “Actions” nel main form del programma.

Configurazione SYSTEM

Alla voce “Configurazione SYSTEM” possono essere introdotte le varie configurazioni per il Real Time oltre che create le diverse task definite dall’utente: i.e. per gli output o per leggere le variabili I/O da linguaggi di programmazioni di alto livello come Visual Basic, Visual C++, C# .NET o Visual Basic.NET .

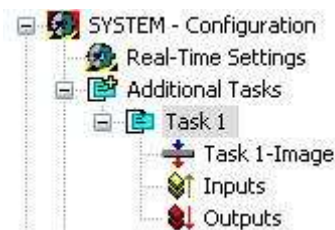


Fig. 2.7 – Finestra task

Aprendo il menù a tendina (immagine a fianco) e facendo click con il mouse su “Additional Tasks”, appaiono le seguenti opzioni:

- Append Task
Aggiunge nuove task.
- Import Task
Integra nel sistema esistente le task create ed esportate precedentemente

- Paste
Inserisce task aggiuntive dalla clipboard
- Paste with Links
Inserisce task addizionali con link alle variabili dalla clipboard

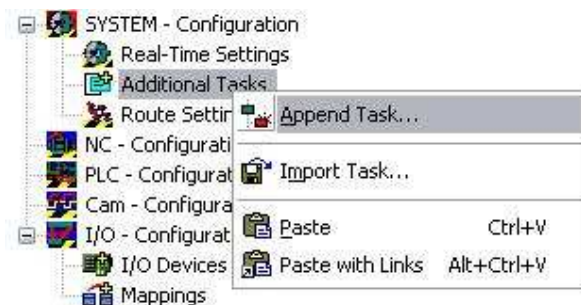


Fig. 2.8 – Opzioni task

Nel momento in cui si decida di aggiungere una task , la seguente finestra di dialogo appare:



Fig. 2.9 – Finestra di dialogo

A questo punto bisogna sceglierne e, nel caso in cui servissero ulteriori righe per descriverlo al meglio, è dato a disposizione uno spazio di commento.

La nuova Task, apparsa nell'”albero” delle diverse scelte, automaticamente contiene i sotto menù “Process Image”, “Inputs” e “Outputs”.

Un click con il tasto destro del mouse sulla specifica task mostra il seguente menù :

- **Export Task**
Esporta le configurazioni del task con sotto elementi e i diversi collegamenti in un file con il suffisso *.tce.
- **Export Header File**
Esporta le variabili di input output della task in un header file di C/C++
- **Cut/ Copy**
Copia le task sulla clipboard rimuovendolo/copiandole dalla configurazione corrente.
- **Disabled**
Esclude la corrente task dall'interazione con TwinCAT system.



Fig. 2.10 - Opzioni task

Configurazione delle variabili

Le variabili di In e output possono essere aggiunte all'immagine di processo : queste possono essere collegate con i vari dispositivi I/O o con variabili di altre task.

Per aggiungere una variabile basta andare nelle sotto opzioni del menu Input e Output, fare click con il tasto destro facendo apparire il menù a fianco riportato dal quale, selezionando "Insert Variable" ottiene la seguente finestra di dialogo:



Fig. 2.11 – Aggiunta variabile

La voce "Name", definisce il nome della variabile,"Comment" un opzionale commento su di essa,"Start Address" specifica l'indirizzo della variabile nell'immagine di processo della Task e "Multiple" definisce quante variabili dello stesso tipo possono essere create e aggiunte con indirizzi sequenziali.

In "Variable Type" è riportata la lista di tutti i tipi di dati riconoscibili da TwinCAT

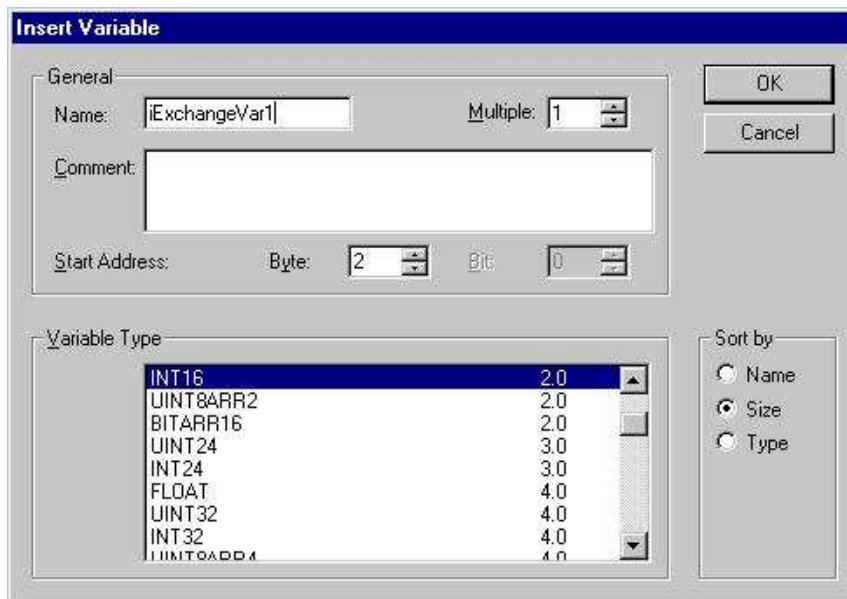


Fig. 2.12 – Tipi di variabili

System Manager, dalla quale è possibile scegliere quello più appropriato per la nostra variabile.

I/O-Configuration

La configurazione I/O è un'importante parte di TwinCAT System Manager in quanto, dopo aver configurato le differenti task e impostato le diverse variabili nel System Manager, l'hardware (di solito un bus di campo con moduli I/O) è configurato sotto questa opzione: basta far con il tasto destro del mouse su I/O "Devices" per mostrare il seguente menù.

Facendo click con il tasto destro del mouse su "I/O Devices" otteniamo le seguenti opzioni:

- Append device:

Aprire la finestra di selezione per i diversi fieldbus supportati o per altro hardware.

- Import Device
Integra le configurazioni I/O create ed esportare precedentemente nel progetto di System Manager.



- Scan devices

Fig. 2.13 – I/O configurazione

Cerca i dispositivi supportati mostrandone, successivamente, nella lista ad albero sotto la voce "I/O Devices". Per questa funzione è richiesta la modalità "Config Mode"

Dopo aver configurato il tutto, aprendo la lista ad albero sotto “I/O Devices”, si ritroveranno lo status e le informazioni di controllo del dispositivo selezionato: quelle che appaiono sono le variabili di input e output che possono essere collegate. Le diverse variabili di input o output possono essere collegate quando sono selezionate o nel menu di testo o di dialogo mostrato nella parte destra: alcune delle opzioni selezionabili dal menù di testo sono disponibili nel momento in cui la configurazione è attiva.

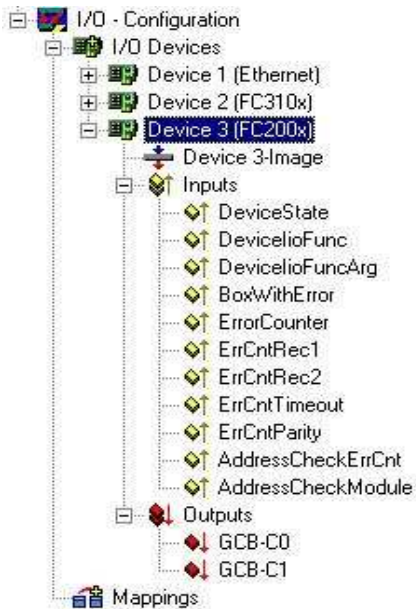


Fig. 2.14 - Device

Attraverso l’opzione “Change Link”, siamo in grado di aprire il “Selection Dialog” per impostare la variabile che deve essere linkata, mentre con “Clear Link(s)” possiamo rimuovere un collegamento (il quale è indicato con una piccola freccia alla sinistra del simbolo di variabile). Il comando “Insert Variable” aggiunge ulteriori variabili alla task mentre “Delete” cancella la variabile desiderata dalla lista. “Add to Watch” aggiunge la variabile selezionata alla “Watch Window” attraverso la quale è possibile una continua osservazione della variabile corrente. Per disabilitare tale possibilità basta selezionare “Remove from Watch”. Infine “Create Linked Variable in” permette di aggiungere le variabili collegate ad un esistente progetto PLC.

Dopo aver illustrato con completezza il funzionamento generico del System Manager e delle modalità di creazione e associazione delle diverse variabili in gioco, si passa ad illustrare il nostro caso specifico, riportando uno screenshot della configurazione personale del sistema:

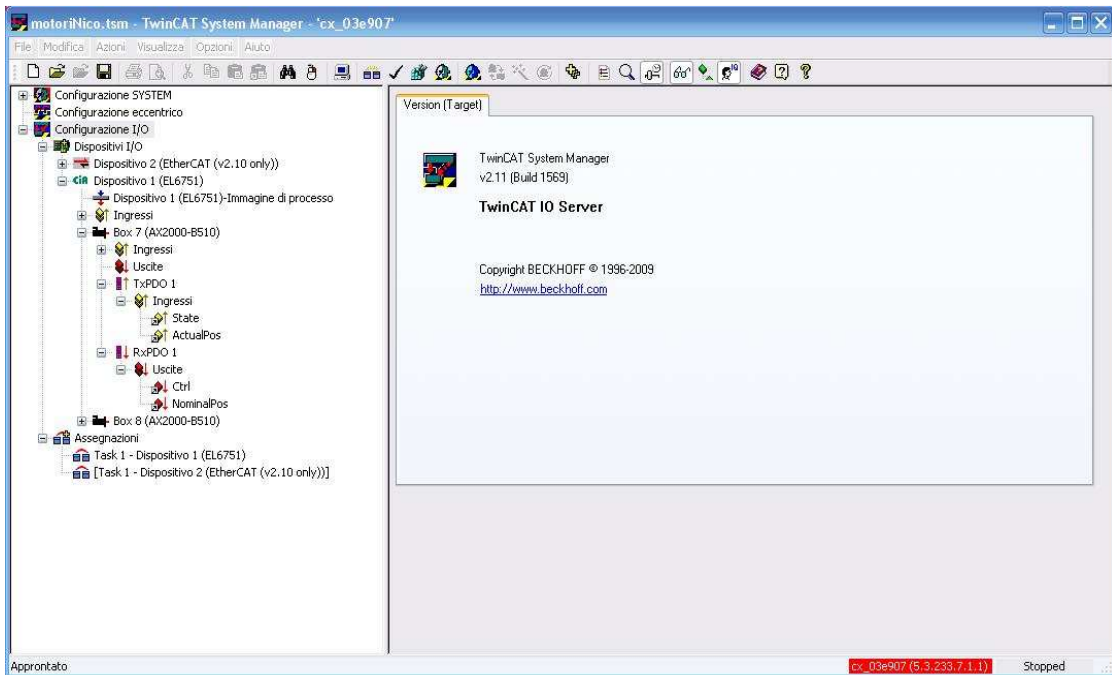


Fig. 2.15 – Finestra principale

Si noti la piccola icona a forma di motore che sta ad indicare come System Manager abbia riconosciuto in modo ottimale il nostro sistema, mostrandone le possibili variabili collegabili, suddividendole con il colore giallo gli input (“State”, “ActualPos”) e con il rosso gli output (“Ctrl”, “NominalPos”) che andremo a vedere tra breve. Nella seconda finestra sono invece riportati i collegamenti delle variabili nell’immagine di processo:

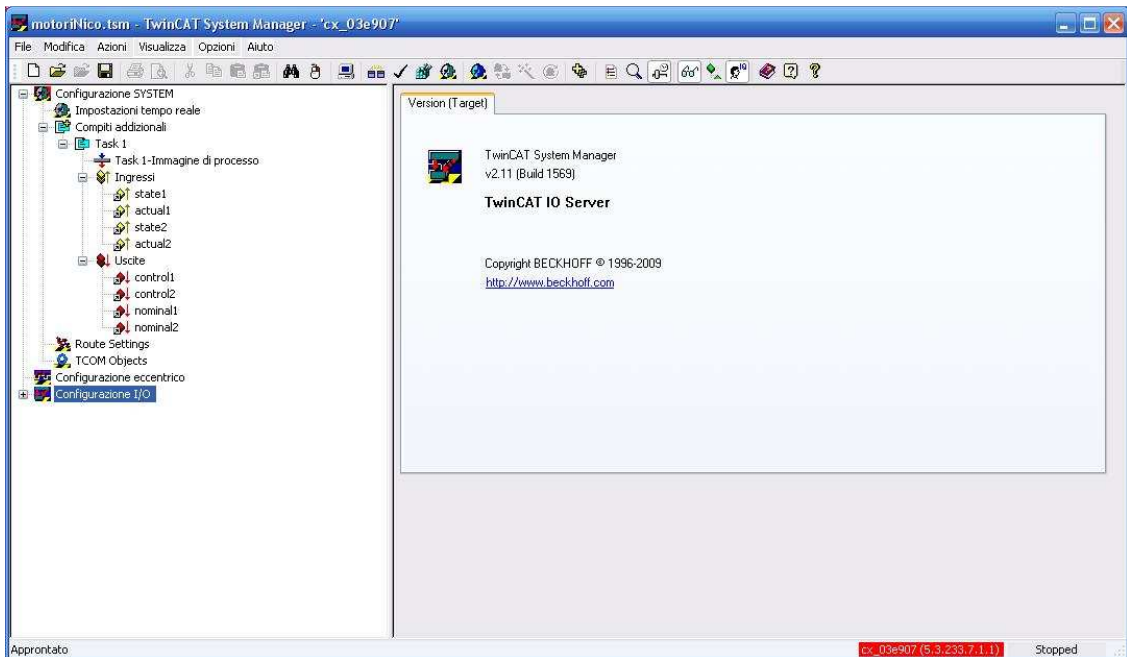


Fig. 2.16 – Finestra principale, seconda veduta

Come prima detto, i colori sono specifici per la tipologia di variabile I/O (i valori 1 o 2 stanno ad indicare parametri specifici per il primo o il secondo asse). Viene illustrata

come appare il form nel momento in cui una variabile sia collegata: in questo caso, la variabile “nominal1” è collegata con la variabile “NominalPos” del Box7.

The screenshot shows a configuration window for a variable. The 'Variabile' tab is selected. The variable name is 'nominal1', its type is 'DINT', and its group is 'Uscite'. The size is set to 4.0, the address is 4 (0x4), and the user ID is 0. The 'Collegato a' (Linked to) dropdown menu is open, showing the selected variable: 'NominalPos . Uscite . RxPDO 1 . Box 7 (AX2000-B510) . Dispositivo 1 (EL6)'. The 'ADS Info' field at the bottom indicates the port (301), group (0xF030), offset (0x4), and length (4).

Fig. 2.17 – Collegamento variabili

Fase successiva è quella della creazione dell’headers file che da la possibilità di sfruttare le variabili appena create all’interno del programma Visual Studio©.

Capitolo 3

Legge di moto trapezoidale

Al fine di pianificare il moto si decide di generare un profilo di posizione lineare raccordato all'inizio e alla fine della traiettoria con tratti parabolici. Tale profilo di velocità ha il tipico andamento trapezoidale come sotto illustrato:

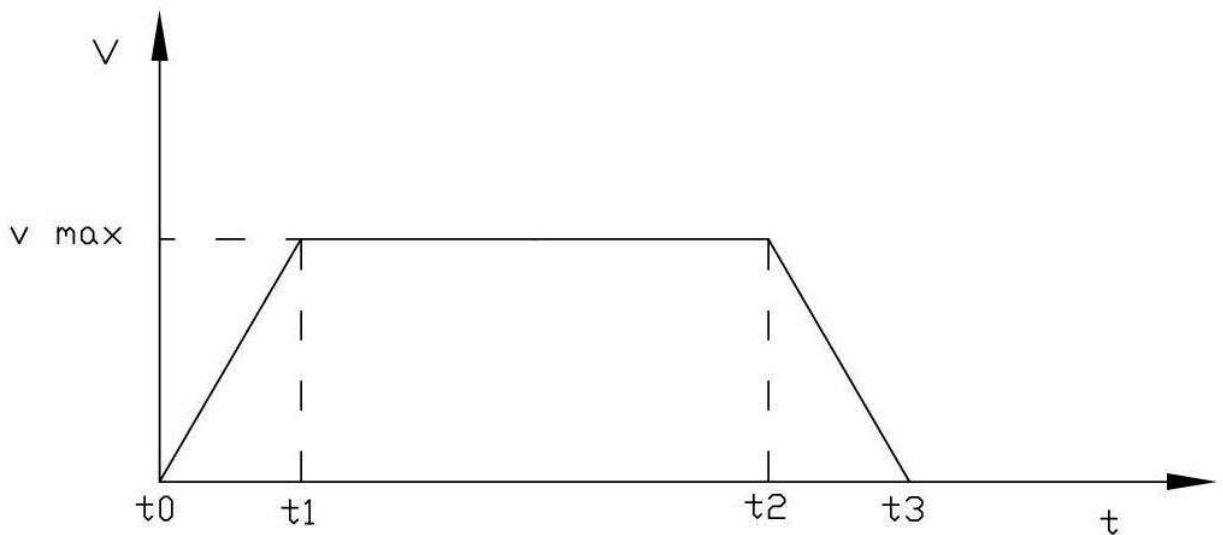


Fig. 3.1 – Legge trapezoidale

Fissati i valori della velocità massima (v_{max}) e di accelerazione massima (a_{max}), si possono immediatamente distinguere tre tratti distinti:

- Tratto compreso tra t_0 e t_1

Tale è caratterizzato dall'essere un moto uniformemente accelerato e, pertanto, le relazioni che lo governano sono le seguenti:

$$a = \text{cost}$$

$$v = v_0 + a(t - t_0)$$

Dalla sua integrazione otteniamo che

$$s = s_0 + v_0(t - t_0) + \frac{1}{2}(t - t_0)^2 a$$

dove

" s " è lo distanza, " s_0 " la distanza iniziale, " v_0 " la velocità iniziale, " a " è l'accelerazione e " t " il tempo.

Poiché s_0, v_0 sono nulli, l'equazioni si riducono a

$$v = a(t - t_0) \quad (1)$$

e

$$s = \frac{1}{2} a(t - t_0)^2$$

Per $t = t_1$ raggiungiamo il valore di velocità massima pari a :

$$v_{max} = a(t_1 - t_0)$$

la distanza invece

$$s_1 = \frac{1}{2} a(t_1 - t_0)^2$$

- Tratto compreso tra t_1 e t_2

In questo caso il moto è uniforme, per esso valgono le seguenti relazioni:

$$v = v_{max} = cost$$

dalla sua integrazione otteniamo che

$$s = s_0 + v(t - t_1) \quad (2)$$

In questo caso s_0 non è nullo ma vale s_1 calcolato prima. Per $t = t_2$ otteniamo la distanza totale compiuta a partire dal riferimento t_0 :

$$s_2 = \frac{1}{2} (t_1 - t_0)^2 a + v(t_2 - t_1)$$

- Tratto compreso tra t_2 e t_3

Si noti come, per simmetria costruttiva,

$$t_1 - t_0 = t_3 - t_2 \quad (3)$$

Il tratto finale deve portare il valore della velocità finale a zero, di conseguenza il tratto sarà uniformemente decelerato, in equazioni:

$$acc = -a = cost$$

e

$$v = v_{max} - a(t - t_2)$$

Dalla sua integrazione otteniamo che

$$s = s_3 - \frac{1}{2}a(t - t_3)^2$$

dove “s” è la distanza, “s₃” la distanza finale definita dall’utente, “v_{max}” la velocità massima precedentemente definita e “t” il tempo.

Per $t = t_3$ abbiamo :

$$v = 0$$

e

$$s = s_3$$

$$s = \frac{1}{2}a(t - t_0)^2$$

Riassumendo, raggruppiamo i tre rami della nostra spezzata in un sistema riassuntivo:

$$s(t) = \begin{cases} \frac{1}{2}a(t - t_0)^2 & t_0 \leq t \leq t_1 \\ v_{max}(t - t_1) + \frac{1}{2}a(t_1 - t_0)^2 & t_0 \leq t \leq t_1 \\ s_3 - \frac{1}{2}a(t - t_3)^2 & t_2 \leq t \leq t_3 \end{cases} \quad (4)$$

Si noti come la condizione limite affinché tale legge di moto sia possibile è che la distanza impostata sia maggiore di un valore identificato con il nome di spazio limite: per s_3 uguale a tale valore il segmento che identifica il tratto a velocità costante degenera in un punto e ciò che otteniamo è un triangolo.

Con riferimento all'immagine sotto portata, tale distanza è facilmente identificabile se si considera (3): infatti fissati a_{max} , v_{max} e (1) otteniamo

$$t_{lim} = (t_1 - t_0) = \frac{v_{max}}{a_{max}}$$

e, quindi,

$$t_2 - t_0 = 2t_{lim}$$

$$s_{3,lim} = 2 \frac{1}{2} a_{max} (t_{lim})^2 = \frac{v_{max}^2}{a_{max}}$$

Relazione immediatamente data se si considera l'area del triangolo con base $2t_{lim}$ e altezza v_{max} .

Il sistema di equazioni permane a meno del tratto a velocità costante:

$$s(t) = \begin{cases} \frac{1}{2} a (t - t_0)^2 & t_0 \leq t \leq t_1 \\ s_3 - \frac{1}{2} a (t - t_2)^2 & t_1 < t \leq t_2 \end{cases} \quad (5)$$

In figura è riportato il triangolo degenero

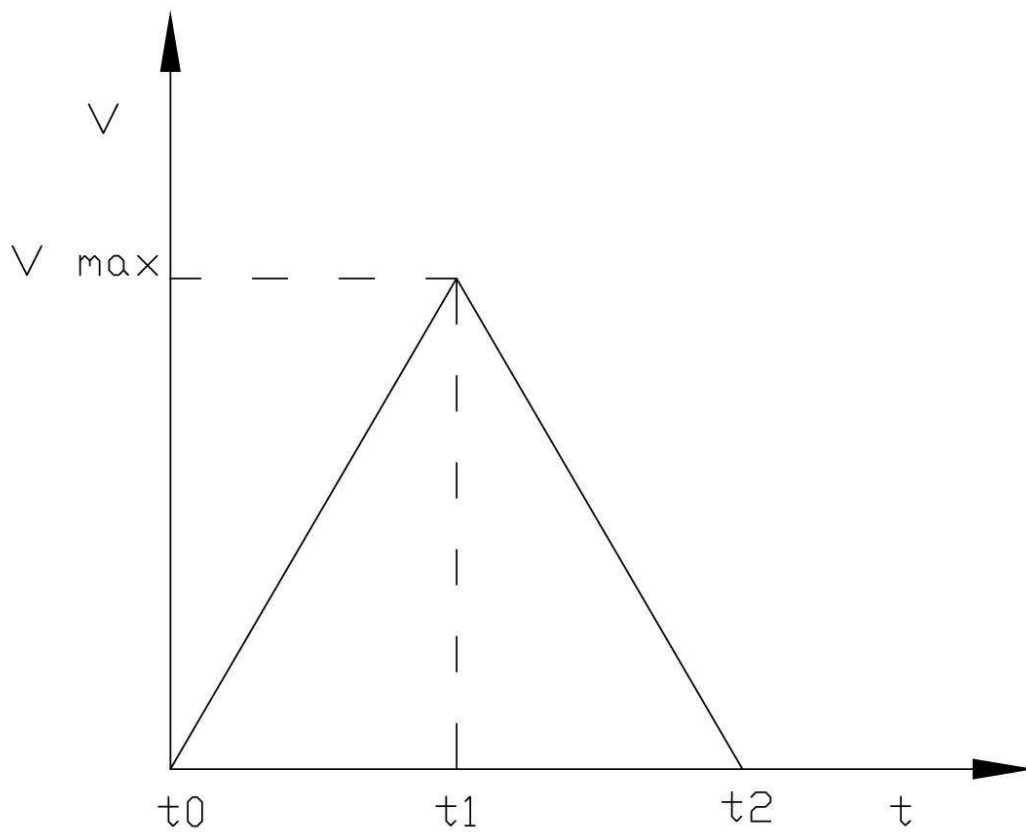


Fig. 3.2 – Legge triangolare

Nel caso finale in cui risulta che il valore di s_3 impostato dall'utente sia inferiore del valore limite, quello che si ottiene, tenuto $a_{max} = cost$, è una serie di triangoli degeneri con altezza (cioè velocità massima del caso) " h " compresa tra il valore 0 e quello del triangolo degenere sopra citato cioè v_{max} . Tutto ciò è facilmente comprensibile guardando la figura allegata:

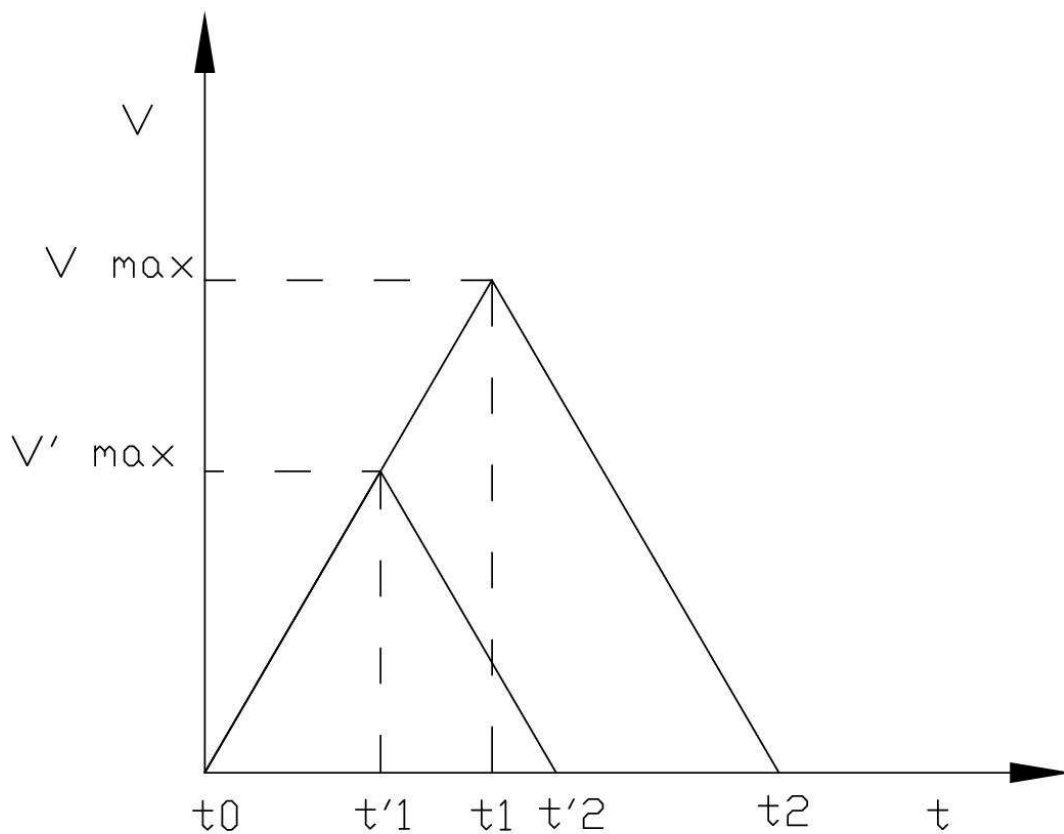


Fig. 3.3 – Casi limiti

Definito s_3' come il valore della distanza inferiore a quella limite e v'_{max} il nuovo valore della velocità massima, si ottiene:

$$v'_{max} = \sqrt{s_3' a_{max}} \quad (6)$$

avendo considerato l'area del triangolo di base $t_2 - t_0$, altezza v'_{max} e le relazioni sul moto uniformemente accelerato sopra esposte.

Conoscendo (6) si ricava

$$t'_1 - t_0 = \frac{v'_{max}}{a_{max}}$$

e

$$t'_2 - t_0 = 2 \frac{v'_{max}}{a_{max}}$$

da cui si ricava il sistema di equazioni così modificato:

$$s(t) = \begin{cases} \frac{1}{2}a(t - t_0)^2 & t_0 \leq t \leq t'_1 \\ s_3 - \frac{1}{2}a(t - t'_2)^2 & t'_1 < t \leq t'_2 \end{cases} \quad (7)$$

Riassumendo i tre diversi casi otteniamo:

- $s_3 > s_{3,lim}$ legge trapezoidale, sistema di riferimento : (4)
- $s_3 = s_{3,lim}$ legge triangolare limite, sistema di riferimento : (5)
- $s_3 < s_{3,lim}$ legge triangolare, sistema di riferimento : (7)

Capitolo 4

Implementazione del codice

Dopo aver dato le basi teoriche sul software e sull'hardware utilizzato, oltre che aver illustrato in maniera approfondita la modalità di configurazione con System Manager e la creazione delle relative variabili, si giunge allo sviluppo del tutto: il fine è quello di processare il valore immesso dall'utente e pianificare tale distanza in un moto trapezoidale. Il programma, quindi, si suddivide in due parti:

- una *.dll scritta nel linguaggio c++ all'interno della quale sono definite le modalità di suddivisione del moto e le relative funzioni operanti in campo real time : tale parte comprende l'assegnazione delle diverse variabili, processate secondo un algoritmo definito e le variabili di output vengono poi scritte esternamente
- un'interfaccia utente che permette la visualizzazione dello stato del mio driver (possibile attraverso uno schema logico fornito) oltre che la possibilità di definire la distanza che il mio asse deve compiere. Il linguaggio utilizzato è il Visual c#.

Premessa: stati dell'azionamento

La figura sopra riportata rappresenta i vari stati dell'azionamento usato, mentre quella sotto la loro descrizione:

Stato	Descrizione
NOT READY TO SWITCH ON	L'azionamento è alimentato con alimentazione ausiliaria. È in fase di inizializzazione oppure sta eseguendo un autotest. L'azionamento è disabilitato.
SWITCH ON DISABLED	L'inizializzazione dell'azionamento è conclusa. I parametri dell'azionamento sono stati reimposti e possono essere modificati. L'azionamento è disabilitato.
READY TO SWITCH ON	È possibile alimentare l'azionamento con l'alimentazione principale. I parametri dell'azionamento possono essere modificati. L'azionamento è disabilitato.
SWITCHED ON	L'azionamento è alimentato con alimentazione principale. Il servo convertitore è pronto. I parametri dell'azionamento possono essere modificati. L'azionamento è disabilitato.
OPERATION ENABLED	Non è stato rilevato alcun malfunzionamento. L'azionamento è abilitato ed il motore alimentato. I parametri dell'azionamento possono essere modificati.
QUICK STOP ACTIVE	È intervenuta la funzione arresto di emergenza. I parametri dell'azionamento possono essere modificati. L'azionamento è abilitato ed il motore alimentato.
FAULT REACTION ACTIVE	Si è verificato un errore (fault) sull'azionamento. È intervenuta la funzione arresto di emergenza. I parametri dell'azionamento possono essere modificati. L'azionamento è abilitato ed il motore alimentato.
FAULT	Si è verificato un errore (fault) sull'azionamento. I parametri dell'azionamento possono essere modificati. L'azionamento è disabilitato. L'alimentazione principale può essere accesa o spenta a seconda del tipo di applicazione.

Fig. 4.1 – Stati azionamento - 1

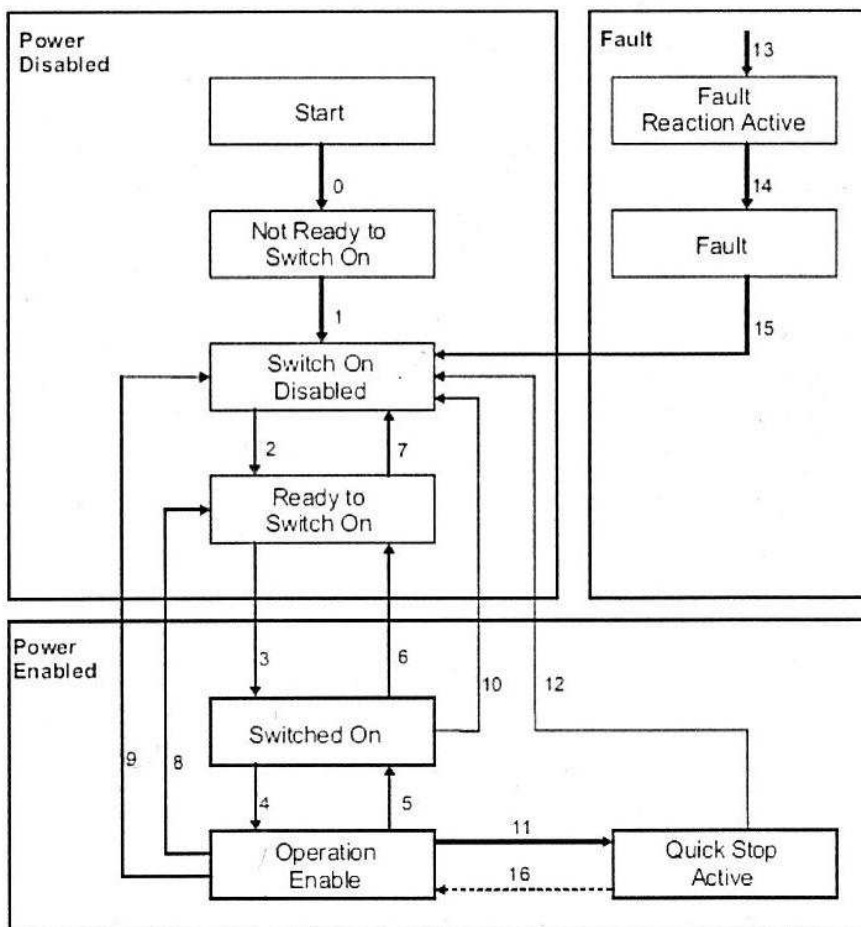


Fig. 4.2 – Stati azionamento - 2

Questa premessa è necessaria per capire le modalità usate per interagire direttamente con il nostro sistema e infatti per interagire con esso si usano un insieme di bit, chiamati “Controlword”, tra i quali è incluso lo stato, i modi operativi o altre funzioni che dipendono dal particolare operatore. Tali sono:

Funzione	Manufacturer specific	reserved	Halt	Fault reset	Operation mode specific	Enable operation	Quick stop	Enable voltage	Switch on
bit	15 - 11	10 - 9	8	7	6 - 4	3	2	1	0

I bit da 0 a 3 e il 7 definiscono il comando da inviare alla macchina per cambiare lo stato interno secondo la seguente ulteriore tabella (i bit segnati con una “x” non sono rilevanti per le transizioni):

Comando	Controlword	Transizioni
Shutdown	xxxx xxxx xxxx x110	2,6,8
Switch On	xxxx xxxx xxxx x111	3
Disable Voltage	xxxx xxxx xxxx xx0x	7,9,10,12

Quick Stop	xxxx xxxx xxxx x01x	7,10,11
Disable Operation	xxxx xxxx xxxx 0111	5
Enable Operation	xxxx xxxx xxxx 1111	4,16
Fault Reset	xxxx xxxx 1xxx xxxx	15

Con lo “Statusword” siamo invece in grado di leggere lo stato corrente oltre che ad altre informazioni sullo stato del dispositivo. In modo analogo si propone una tabella per lo Statusword circa i comandi e il significato dei diversi bit:

Statusword	Commando
xxxx xxxx x0xx 0000	Not ready to switch on
xxxx xxxx x1xx 0000	Switch On disabled
xxxx xxxx x01x 0001	Ready to switch on
xxxx xxxx x01x 0011	Switched on
xxxx xxxx x01x 0111	Operation enabled
xxxx xxxx x00x 0111	Quick stop activated
xxxx xxxx x0xx 1111	Fault reaction activated
xxxx xxxx x0xx 1000	Fault

Bit	Descrizione
0	Ready to switch on
1	Switched on
2	Operation enabled
3	Fault
4	Voltage enabled
5	Quick stop
6	Switch On disabled
7	Warning
8	Manufacturer specific
9	Remote
10	Target reached
11	Internal limit active
12 - 13	Operation mode specific
14 - 15	Manufacturer specific

Interfaccia utente

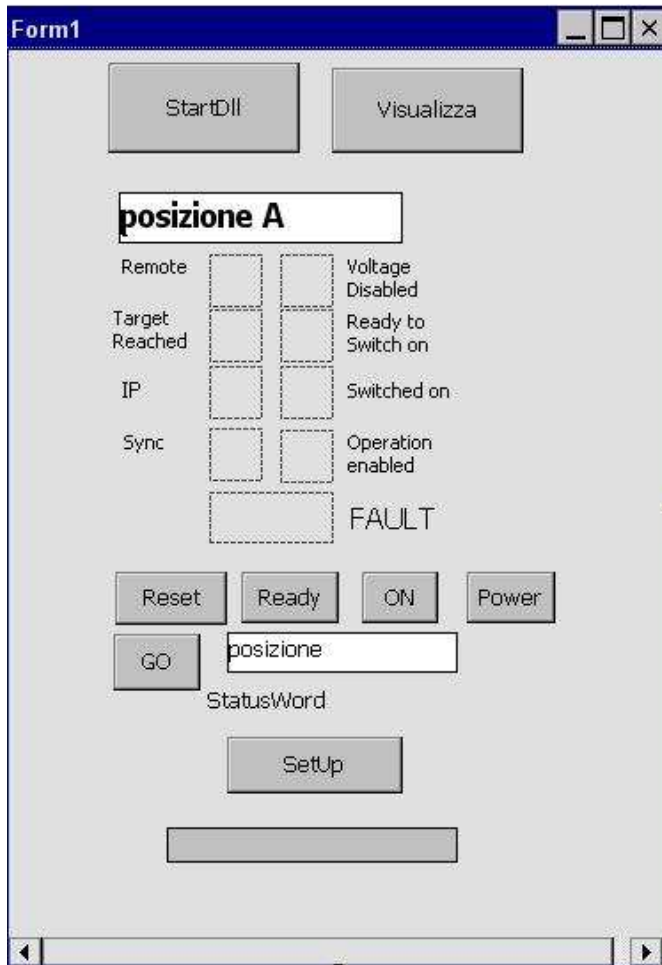


Fig. 4.3 – Interfaccia utente

Viene riportata l'interfaccia utente con il quale si è in grado di interagire direttamente con la dll.

Riferendoci all'immagine, si osservi il punto (1) dove sono collocati due command button la cui funzione è proprio quella di far partire l'interazione tra l'intero hardware e il pc (il software System Manager deve essere stato precedentemente azionato e il sistema impostato sullo stato "Running"): all'evento "Click" del tasto "Visualizza" il restante comparto di bottoni viene attivato mentre nel textBox appare la posizione letta dall'encoder del motore.

I vari stati indicati sono i medesimi citati nel paragrafo superiore e al fine d'identificarli si opera nel seguente modo:

la funzione “valori”, a seconda del parametro “type” mi assegna alla variabile “var” o la posizione (type = 1) o lo stato il cui valore, in decimale convertito in binario, viene letto nella tabella dello Statusword (type = 2).

Sotto è riportata la funziona presa dalla dll.

```
DL1_API int valori(int type, int num, double &val)
{
    switch (type)
    {
        case 1:
            val = actPos[num];
            break;

        case 2:
            val = status[num];
            break;
    }
    return 0;
}
```

```
double posizione0 = 0;
WCThreadIntf.valori(1, 0, ref posizione0);
textBox1.Text = posizione0.ToString();
```

La posizione viene letta e scritta all’interno della casella di testo mentre lo status viene assegnato ad una variabile e convertito da binario: il metodo è molto semplice e consiste nel fare il cast da “double” a “int”, calcolare il resto del valore letto il quale corrisponde al valore 0 o 1; lo si divide a sua volta per due e a tale si ricalcola il resto, in modo iterativo fino alla totale conversione. Il tutto viene poi incanalato in due array da cui, semplicemente facendo un riferimento alla tabella dei bit per lo Statusword, riesco a capire lo stato del mio drive (vedi appendice 1).

Per il passaggio dei diversi stati si utilizzano i relativi bottoni che, basandosi sulle tabelle di Controlword, inviano un codice decimale al drive relativo ad un determinato e ben preciso stato: per esempio, nel caso in cui mi trovassi nello stato “READY TO SWITCH ON”, corrispondente allo Statusword binario xxxx xxxx x01x 0001 , dando un valore decimale “7” (vedi tabella Controlword) passerei allo stato ”SWITCHED ON”. Tutto ciò corrisponde al seguente codice:

```
private void button5_Click(object sender, EventArgs e)
{
    WCETHreadIntf.setPar(1, 0, 7);
}
```

Dove la funzione “setPar” permette di impostare o un parametro o una posizione target desiderata

```
DLL1_API int setPar(int type, int num, double val)
{
    switch (type)
    {
        case 1:
            control[num] = (short)val;
            break;
        case 2:
            targetPos[num] = (long)val;
            break;
    }
    return 0;
}
```

In modo analogo lavorano i bottoni “Reset”, “Ready” e “Power” a cui, naturalmente, corrispondono specifici comandi. Premendoli in successione e verificando l’adeguato raggiungimento dello stato desiderato, si può passare all’invio vero e proprio di una posizione target, la cui immissione è possibile tramite la casella di testo in basso. Alla pressione del tasto “GO”, il dato viene letto e l’informazione viene gestita basandosi sull’algoritmo esposto in precedenza: in primis, bisogna comprendere se la distanza (differenza tra posizione iniziale e finale) porta alla rotazione in senso orario o antiorario e ciò viene gestito nella parte iniziale di inizializzazione delle variabili (appendice 2)

Nella parte iniziale del codice vengono lette le diverse entrate per poi essere assegnate a determinati vettori. La variabile boolean “init” è di fondamentale importanza in quanto consente l’assegnazione effettiva di tutti i diversi parametri; il suo valore è pari a “true” nel caso in cui preme il pulsante “GO” (vedi funzione “setPar”) e “false” alla fine dell’assegnazione.

```

di TIME
    if (init == true)
    {
        //questa assegnazione non sarebbe necessaria a meno
        TIME = 0;
    }

```

```

        dist_lim = 0; // distanza limite che mi definisce il
caso in cui ho legge trapezoidale o triangolare
        dist_vel_cost = 0;
        dist_sign = 0;
        distanza_vet[0] = 0;
        addendo[0] = 0;
        addendo_2 = 0;
        startPos[0] = actPos[0];
        //calcolo
        distanza_2 = targetPos[0] - startPos[0];

        distanza_vet[0] = targetPos[0] - startPos[0];

        // lavoro con il vettore
        distanza = distanza_vet[0];

```

Per cercare di discriminare il caso in cui sto la mia distanza è positiva (il motore gira in senso orario) o negativa (il motore gira in senso antiorario) di una funzione segno che va a moltiplicare il valore di velocità e accelerazione considerata per fare in modo che il segno contrario non vada ad alterare la bontà dell’algoritmo sviluppato: infatti, se avessi una distanza negativa ma il segno della velocità positiva, avrei un moto uniformemente acc/dec al posto di uno dec/acc.

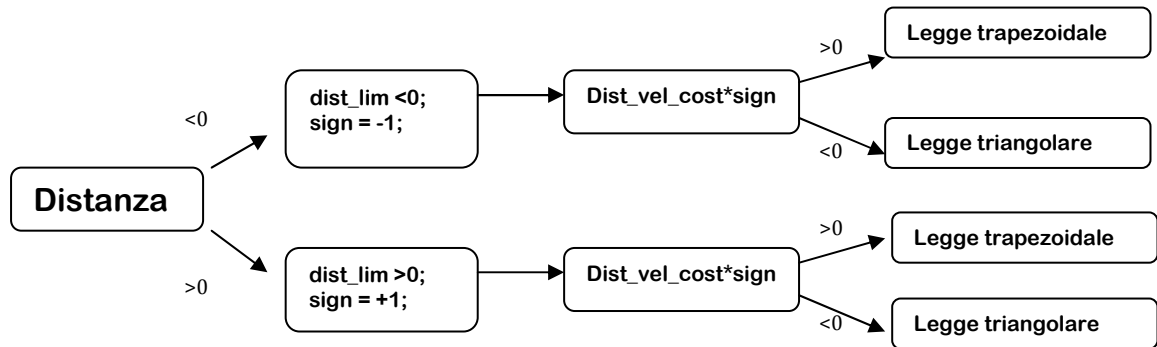
```

if (distanza_vet[0] <0)
{
    sign = -1;
}
else
{
    sign = +1;
}

v_max = sign*fabs(v_max);
a_max = sign*fabs(a_max);

```

La variabile “dist_vel_cost” mi discrimina il caso in cui ho legge triangolare o trapezoidale: per una distanza totale minore o uguale di tale valore, mi trovo nel caso triangolare. Moltiplico inoltre il valore della distanza così calcolato per il segno per fare in modo che la mia funzione non cada in fallo nel momento in cui ho a che fare con distanza negative: infatti, se io considerassi un caso di distanza positiva e maggiore di quella limite, allora il valore di dist_vel_cost sarà positiva;se invece avessimo una distanza negativa, dist_lim sarebbe a sua volta negativa ,quindi dist_vel_cost sarebbe negativo. Moltiplicando per “sign” avremmo di nuovo un valore positivo. In questo modo riesco con facilità a discriminare i due casi. In modo schematico:



Indicato con “`dist_sign = sign*dist_vel_cost`”, sotto è riportato il relativo codice:

```

dist_lim = (long)(v_max*v_max/a_max);
//calcolo tratto a velocità costante
dist_vel_cost = distanza_vet[0] - dist_lim;
dist_sign = sign*dist_vel_cost;
TIME = 0;//inizializzo la variabile tempo
T0 = TIME;

if(dist_sign>=0)
{
    //tempo del tratto a velocità costante
    TIME_vel_cost = dist_vel_cost/v_max;
    //per il caso limite e legge trapezoidale
    T1 = T0 + v_max/a_max;
    T2 = T1 + TIME_vel_cost;//tempo fine tratto vel
    costante

    T3 = T2 + v_max/a_max;//tempo finale
}

else
{
    v_max_lim = sign*sqrt(distanza_vet[0]*a_max);
    T1 = T0 + (v_max_lim/a_max);
    T2 = T1;
    T3 = T1 + (v_max_lim/a_max);
}

init = false;
}
  
```


Sono definite quindi le diverse variabili di tempo necessarie per attuare in modo efficace la nostra legge cercata: nel caso di legge trapezoidale le variabili T0,T1,T2 e T3. Nel caso di legge triangolare, riferendoci alla trattazione iniziale, dobbiamo definire una nuova velocità massima a parità di accelerazione e tale è definita come “v_max_lim”. L’eguagliare T2 con T3 in quest’ultimo caso verrà compreso poco più avanti. A seguito di questa fase, la variabile booleana “init” verrà definita “false”.

Inizializzato l’intero sistema di variabili e i diversi valori temporali di riferimento, ricordando che ci si trova all’interno di un ciclo while sempre vero finché la variabile “!g_bFinish == true“, si incrementa la variabile temporanea TIME e, a seconda del suo valore, sappiamo che il sistema è nel tratto ad accelerazione, velocità o decelerazione costante: ricordando la parte teorica precedentemente definita, abbiamo che:

$$\text{tratto} \begin{cases} \text{ad accelerazione costante,} & T_0 \leq \text{TIME} \leq T_1 \\ \text{a velocità costante} & , \quad T_1 < \text{TIME} < T_2 \\ \text{a decelerazione costante,} & T_2 \leq \text{TIME} \leq T_3 \end{cases}$$

Implementato nel seguente codice:

- Tratto uniformemente accelerato

```

if (TIME >= T0 && TIME <= T1)
{
    refPos[0] = (long)(startPos[0] + 0.5*a_max*(TIME-
T0)*(TIME-T0));
}

```

- Tratto a velocità uniforme

La necessità di mettere , nel caso di legge triangolare, T2 = T3, sta nella possibilità di incorporare tutto il ciclo in poche e semplici righe: infatti, come si vede dal codice, dall’uguaglianza di tali valori la parte a velocità costante viene saltata dal non rispetto delle condizioni del ciclo “if” cosa che, naturalmente, doveva esser tale.

```

else if (TIME>T1 && TIME < T2)
{
refPos[0] = (long)(startPos[0] + 0.5*a_max*(T1-
T0)*(T1-T0) + v_max*(TIME-T1));
}

```

- Tratto uniformemente decelerato

```

else if (TIME>=T2 && TIME<=T3)
{
refPos[0] = (long)(targetPos[0] - 0.5*a_max*(T3-
TIME)*(T3-TIME));
}

TIME = TIME+0.005

```

La variabile “TIME” viene incrementata e, calcolato punto per punto, scrivo le uscite che saranno inviate al nostro drive al motore.

```

pTlmsOut->control1 = control[0];
pTlmsOut->control2 = control[1];
pTlmsOut->nominal1 = refPos[0];
pTlmsOut->nominal2 = refPos[1];
TCatIoOutputUpdate(TASK_1_PORTNUMBER);
}
return 0;
}

```

Conclusioni

Il progetto, che aveva inizialmente mire più ampie, purtroppo, a seguito di diversi problemi riscontrati durante lo sviluppo, è stato leggermente ridimensionato. Al di fuori di questo inconveniente, lo sviluppo è proseguito senza grandi intoppi.

Il sistema Beckhoff risulta essere un potente dispositivo per il controllo real-time di ultima generazione a seguito delle sue grandi prestazioni oltre che dell'affidabilità, della flessibilità e dei bassi costi della tecnologia applicata. Inoltre, grazie all'ausilio del software in dotazione TwinCAT System Manager, superato l'ostacolo iniziale circa la comprensione del funzionamento (ambiente, a mio avviso, poco user friendly), si scopre la facilità con cui è possibile far riconoscere al nostro PC le diverse periferiche che possono essere aggiunte. Inoltre la creazione e l'associazione delle diverse variabili è qualcosa di veramente veloce e pratico. L'utilizzo del pacchetto Microsoft VISUAL STUDIO© è stato a nostra discrezione, quindi non esiste alcun vincolo circa l'ambiente di sviluppo software, a patto che se ne utilizzi uno compatibile.

La procedura di test e di debug è molto facilitata durante l'intero sviluppo: infatti, sono messi a disposizione nel sistema di sviluppo, diversi strumenti che permettono all'utente stesso di testare la versione, non ancora definitiva, del suo software, mostrandone, passo per passo, le varie fasi dei diversi cicli, il valore assunto dalle variabili: in parole povere, tutte le diverse istanze sono monitorate in modo continuo permettendo, nei casi più critici (e non sono rari) un totale controllo della situazione e intervenendo per risolvere li eventuali problemi.

Bibliografia

- Guida in linea TwinCAT System
- BECKHOFF New Automation Technology (<http://www.beckhoff.it>)
- Microsoft VISUAL STUDIO © (<http://msdn.microsoft.com>)
- Microsoft Windows Embedded
(<http://www.microsoft.com/windowembedded>)

Ringraziamenti

Giunto alla fine di questo lavoro desidero ringraziare ed esprimere la mia riconoscenza nei confronti di tutte le persone che, in modi diversi, mi sono state vicine, incoraggiandomi nei momenti un po' più difficili durante i miei studi ma anche durante la realizzazione e la stesura di questa tesi.

I miei più sentiti ringraziamenti vanno a:

- Ch.mo Prof. Ing. Giovanni Boschetti il quale, oltre avermi dato la possibilità di sviluppare questa tipologia molto interessante di tesi, ringrazio in particolare per la sua disponibilità e la sua pazienza. Il suo aiuto è stato fondamentale e molto importante durante l'intero sviluppo.
- Ai ragazzi del laboratorio con cui ho passato diverse ore durante lo sviluppo dell'intero sistema.
- Alla mia famiglia, senza il sostegno della quale non avrei potuto raggiungere questa meta.
- A Gloria che mi ha sostenuto in ogni momento durante questo percorso di laurea oltre che nel periodo abbastanza difficile della tesi.
- Agli amici dell'università e non, ma anche a tutti coloro con cui mi hanno accompagnato durante questi tre anni di università. Grazie anche dei momenti più semplici, di quelle quattro parole scambiate che, nei momenti più difficili, risultavano di ristoro. Un particolare ringraziamento va a Rodolfo, da cui ho ricevuto vari stimoli per il continuamento dei miei studi, e a William, amico d'infanzia e dello sport.

Appendice 1

```
double posizionel = 0;
WCEThreadIntf.valori(1, 1, ref posizionel);
textBox2.Text = posizionel.ToString();
//inizializzo i due vettori
double[] S1 = new double[4];
double[] S = new double[5];
double SS1 = 0;

//leggo il valore e lo assegno nella variabile SS1
WCEThreadIntf.valori(2, 0, ref SS1);

// Ready to Switch on 0
S[0] = (SS1 % 2);
();
SS1 = SS1 / 2;

S[1] = (Math.Ceiling(SS1 % 2)); // Switched On 1
SS1 = SS1 / 2;

S[2] = (Math.Ceiling(SS1 % 2)); // Operation Enable 2
SS1 = SS1 / 2;

S[3] = (Math.Ceiling(SS1 % 2)); // fault 3
SS1 = SS1 / 2;

S[1] = (Math.Ceiling(SS1 % 2)); // Voltage enable 4
SS1 = SS1 / 2;
//SS1 % 2    _5
SS1 = SS1 / 2;
//SS1 % 2    _6
SS1 = SS1 / 2;
//SS1 % 2    _7
SS1 = SS1 / 2;
//SS1 % 2    _8
SS1 = SS1 / 2;
S1[0] = SS1 % 2; // remote_9;
SS1 = SS1 / 2;
S1[1] = SS1 % 2; // targetR_10;
SS1 = SS1 / 2;
//SS1 % 2    _11
SS1 = SS1 / 2;
S1[2] = SS1 % 2; // interp_12
SS1 = SS1 / 2;
//SS1 % 2    _13
SS1 = SS1 / 2;
//SS1 % 2    _14
SS1 = SS1 / 2;
S1[3] = SS1 % 2; // Sync _15
```

Appendice 2

```
short status[2] = {0,0}; //vettore relativo allo stato
//unsigned long actPos[2] = {0,0};
long actPos[2] = {0,0}; //vettore relativo alla posizione nel momento
di lettura
short control[2] = {0,0}; //vettore relativo al controllo
//unsigned long refPos[2] = {0,0};
long refPos[2] = {0,0}; //posizione calcolata e scritta all'uscita
//long refPos[2] = {0,0};
//unsigned long targetPos[2]= {0,0};
long targetPos[2]= {0,0}; //vettore relativo alla posizione di arrivo
long distanza_vet[2] = {0,0};
long addendo[2] = {0,0};
long prova[2] = {0,0};

//unsigned long startPos[2] = {0,0};
long startPos[2] = {0,0}; //vettore relativo alla posizione iniziale

double TIME=0; // tempo assoluto definito esternamente
double TIME_vel_cost=0;

double v_max=100; //100 con velocità alte il motore gira in loop..
double a_max=10; //10
double v_max_lim=0;
int sign = 1;
int contatore = 1;
double addendo_2;

//variabili varie la cui definizione si comprenderà meglio
successivamente
long distanza =0;
long distanza_2 = 0;
long dist_lim = 0; // distanza limite che mi definisce il caso in cui
ho legge trapez o triangolare
long dist_vel_cost = 0;
long dist_sign = 0;
long tratto_1 = 0;

double T0;
double T1;
double T2;
double T3;
double factor;
double factor_2;

bool init;
```

Appendice 3

```
DWORD WINAPI mioth(LPVOID lpParameter)
{

    CeSetThreadPriority(GetCurrentThread(),0); // Thread a Priorità
    Massima
    CeSetThreadQuantum(GetCurrentThread(), 0); // Thread senza
    interruzioni
    //inizializzo il contatore

    while(!g_bFinish)// il ciclo parte subito acceso il programma
    {

        WaitForSingleObject(g_hEvent1,INFINITE); // INFINITE

        // Leggo gli ingressi e li assegno alle relative variabili
        TCatIoInputUpdate(TASK_1_PORTNUMBER);
        status[0] = pTlmsIn->state1;
        status[1] = pTlmsIn->state2;
        actPos[0] = pTlmsIn->actual1;
        actPos[1] = pTlmsIn->actual2;
    }
}
```