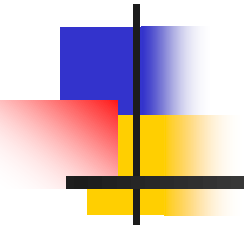


Il Linguaggio di Programmazione IEC 61131-3

Testo di Riferimento: R.W.Lewis, "Programming industrial control systems using IEC 1131-3", IEE Control Engineering Series 50.

Limiti "Storici" della Programmazione dei PLC

- Differenti linguaggi di programmazione
  - Stessi linguaggi ma differenti implementazioni
- Difficoltà nell'utilizzo di sub-routine
- Difficoltà nel produrre software riutilizzabile
- Limiti nella definizione di strutture dati più complesse

Lo Standard IEC 61131-3

- International Electro-technical Committee (IEC)
- Commissione Tecnica: TC65 "Industrial Process Measurement and Control"
  - Sottocommissione: SC65B "Devices"
    - ✓ Working Group: WG7 "Programmable Control Systems".

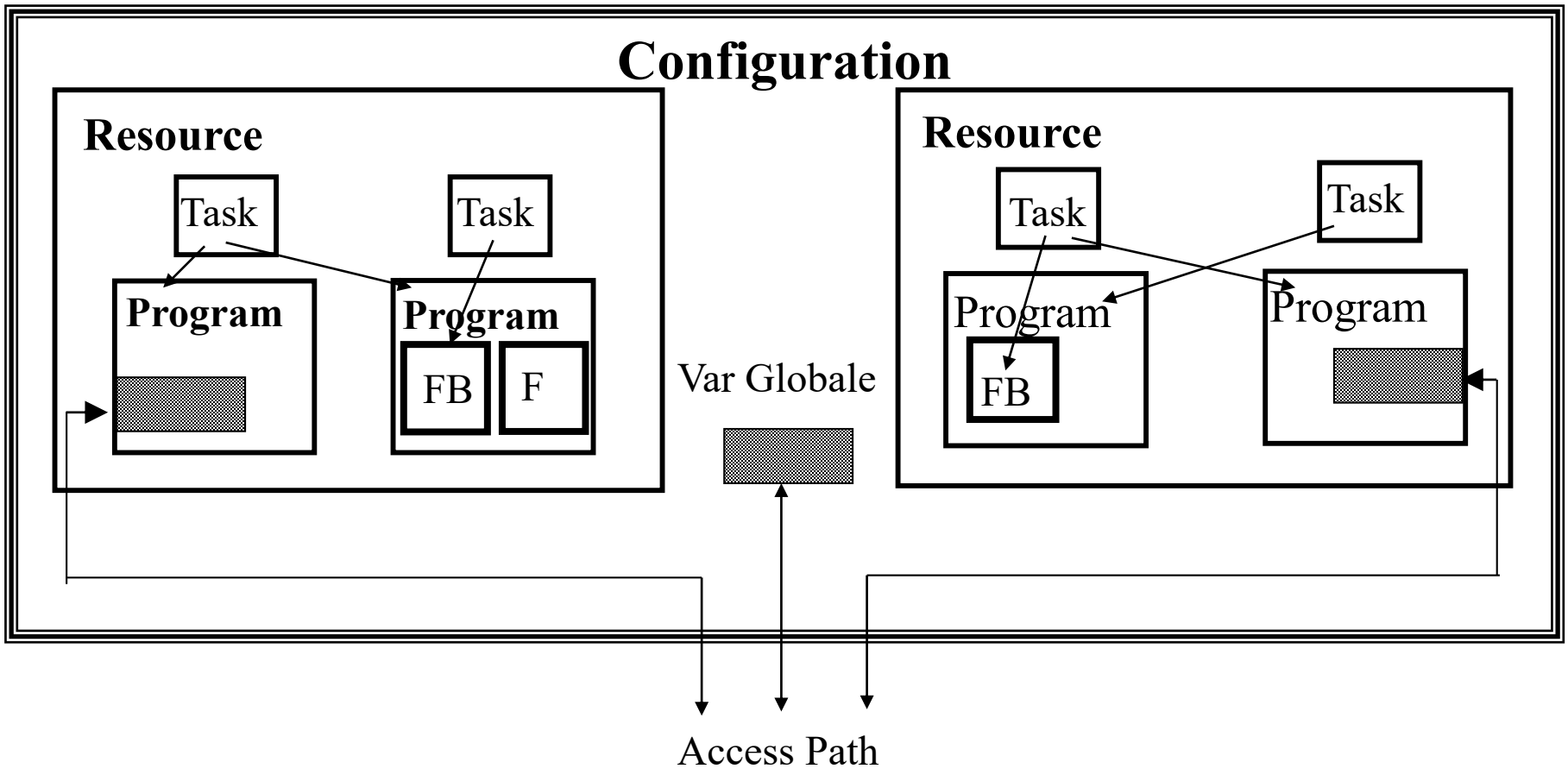
Lo Standard IEC 61131-3

- lo standard è basato sulla programmazione grafica
- lo standard definisce 5 linguaggi:
  - Ladder, Sequential Function Chart (SFC), Instruction List, Function Block Diagram, Structured Text
- lo standard permette lo sviluppo di programmi "mischiando" diversi linguaggi
- lo standard permette approcci: top-down e bottom-up.
- un programma può essere decomposto in Program Organisation Unit (POU)

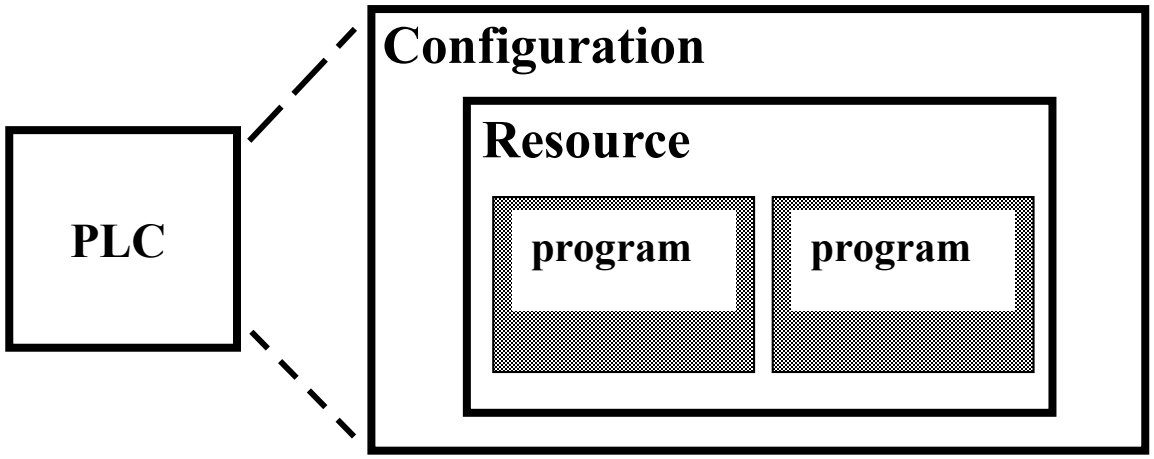
Lo Standard IEC 61131-3

- lo standard permette il pieno controllo dell'esecuzione di ciascun "sotto-programma" tramite l'assegnazione a task
- lo standard permette la definizione di strutture dati complesse: record, vettori.
- lo standard garantisce **in teoria** la portabilità del software.

Modello Software dello Standard IEC 61131-3

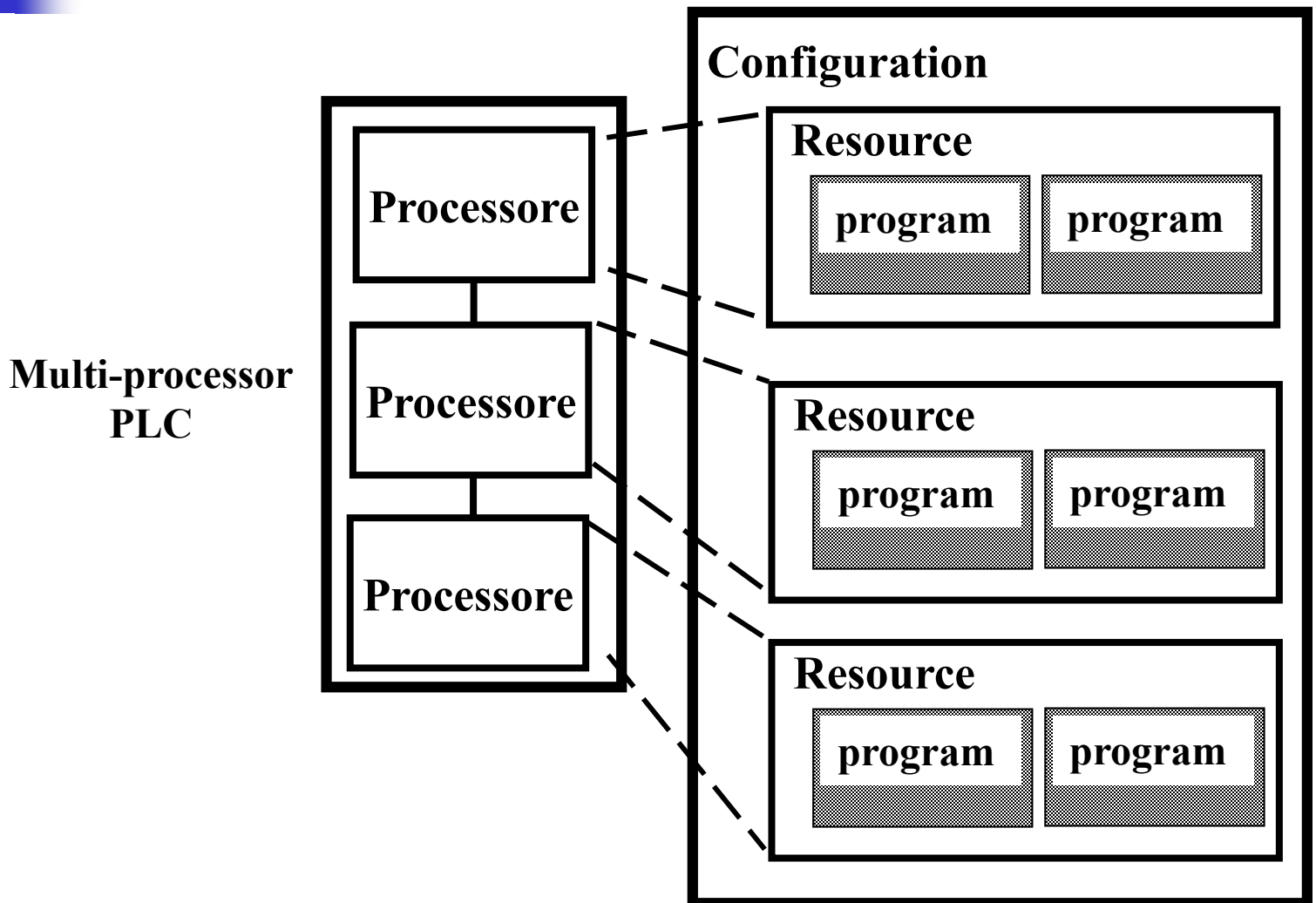


Corrispondenza tra Modello Software e i Sistemi Reali

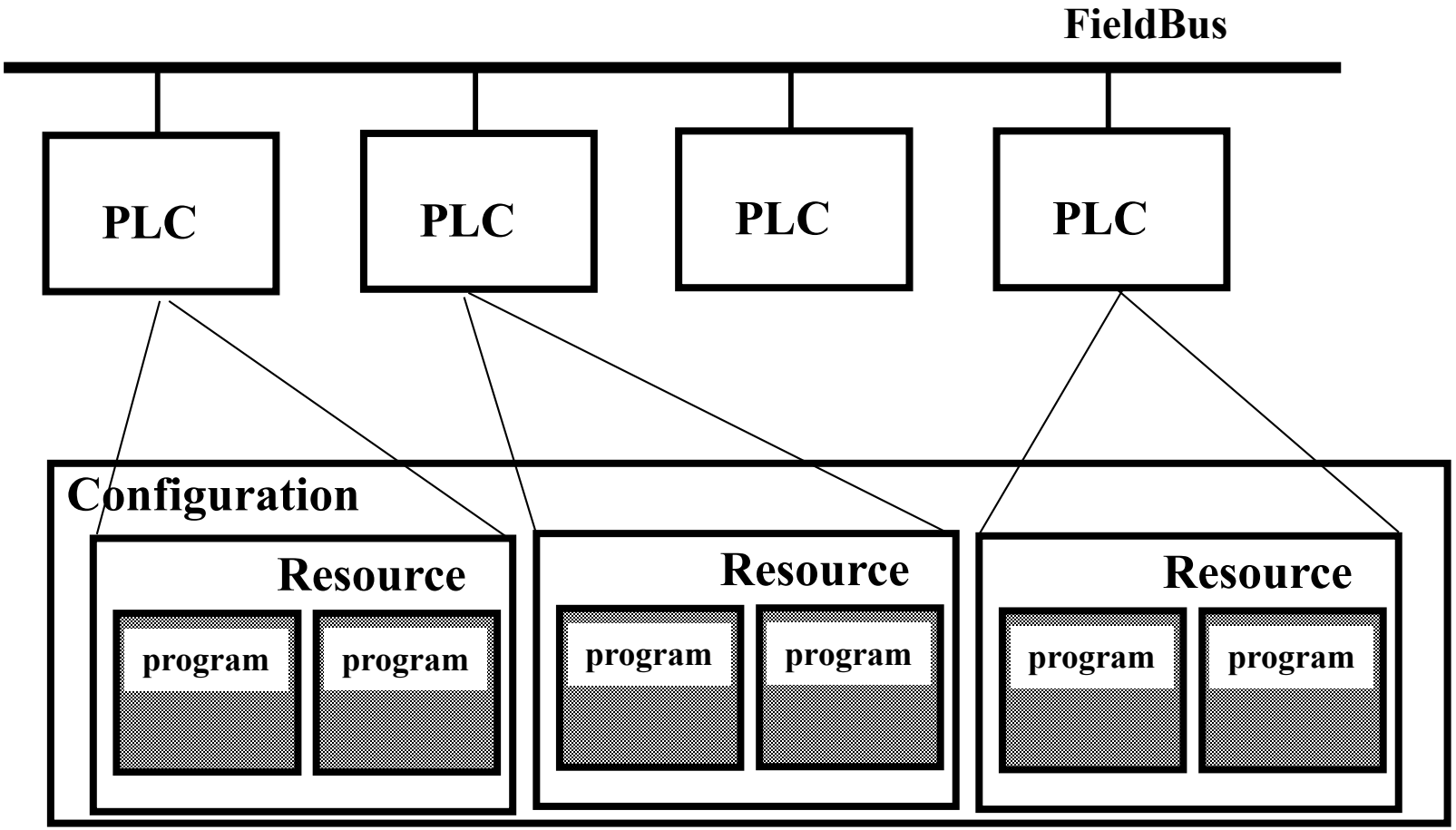


- Singolo Processore:
  - se il processore non supporta il multi-tasking, il numero di programmi è unitario
    - molto raro

Corrispondenza tra Modello Software e i Sistemi Reali



Corrispondenza tra Modello Software e i Sistemi Reali



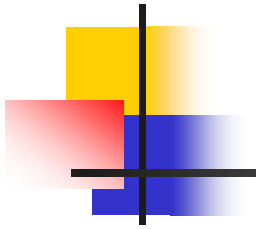
Program Organisation Unit (POU)

- POUs: program, function blocks, function
  - Che differenza c'è tra function blocks e function ?
- La definizione di un POU permette il suo utilizzo un numero di volte illimitato: chiamate di funzioni e istanze di Function Block
  - Ciascuna istanza di un Function Block condivide lo stesso codice, ma ha la sua area privata di memoria
- **Non è ammessa alcuna ricorsione nelle POUs**

Elementi in Comune tra i 5 Linguaggi

- Identificatori
  - Un identificatore può essere costituito da una sequenza di lettere e numeri purché siano soddisfatte le seguenti condizioni:
    - ✓il primo carattere non sia un numero
    - ✓non ci siano più di due caratteri "_" consecutivi
    - ✓non vi siano spazi
  - Lo standard impone che almeno i primi 6 caratteri debbano differenziare due identificatori
- Keywords
  - Lo standard definisce un set di keywords (ad esempio VAR, VAR_EXTERNAL, VAR_ACCESS). Si deve evitare l'uso di identificatori uguali alle keywords.

Elementi in Comune tra i 5 Linguaggi



- Commenti
  - Vengono messi tra (* *)
- Tipi di dati predefiniti:
  - Interi, Reali, Time, Date, String, Boolean
- Tipi di dati derivati:
  - Record (Struct), Enumerativi, Range, Vettori

Tipo TIME

IEC data type	Description	bits
TIME	time duration	implementation dependent

- Valori e/o Costanti:
  - tramite gli operatori d, h, m, s, ms preceduti dall'operatore T# o TIME#
  - Esempi:
    - ✓T#12d3h2s
    - ✓T#3s56ms
    - ✓TIME#6d_10m
    - ✓TIME#16d5h3m4s

Tipi DATES

IEC data type	description	bits
DATE	**calendar date**	**implementation dependent**

- Valori e/o Costanti:
  - formato anno-mese-giorno preceduti dall'operatore D# o DATE#
  - Esempi:
    - ✓ D#1994-06-10 o DATE#1994-06-10

Tipi TIMES of DAY

IEC data type	description	bits
TIME_OF_DAY o TOD	**time of day**	**implementation dependent**
DATE_AND_TIME o DT	**date and time of day**	**implementation dependent**

- Valori e/o Costanti:
  - TOD: formato ora:minuti:secondi.centesimi preceduti dall'operatore TOD# o TIME_OF_DAY#
  - Esempio:
    - ✓ TOD#23:59:34.56 o TIME_OF_DAY#23:59:34.56
  - DT: formato anno-mese-giorno-ora:minuti:secondi.centesimi preceduti dall' operatore DT# o DATE_AND_TIME#
  - Esempio:
    - ✓ DT#1993-04-12-15:36:55.40 o DATE_AND_TIME#1993-04-12-15:36:55.40

Tipi Boolean e String of Bit

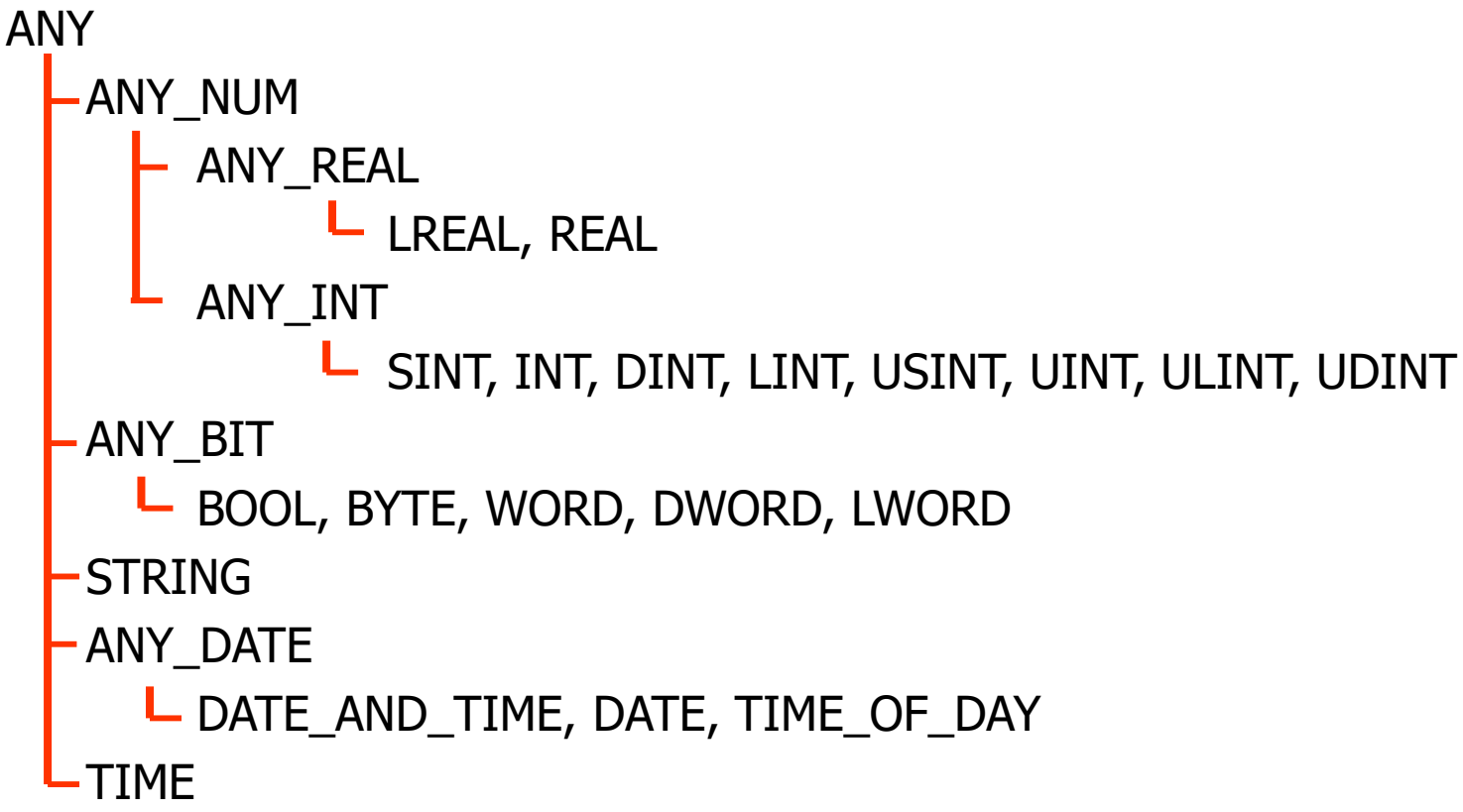
IEC data type	Description
BOOL	Boolean

IEC data type	description	bits
BYTE	bit string 8 bits	8
WORD	bit string 16 bits	16
DWORD	bit string 32 bits	32
LWORD	bit string 64 bits	64

Valori Iniziali di Default

- Interi: 0
- Reali: 0.0
- Time:0d0h0m0s
- Date:0001-01-01
- Stringhe: ‘’
- Boolean: FALSE
- **Nota Bene:** quando viene definita una variabile di un certo tipo, il suo valore iniziale può essere ridefinito

Overloading



Definizione di Variabili

- Variabili Locali:
  - dichiarate in un POU (programma, function block, funzione)
- Variabili Globali :
  - dichiarate in un Program, Resource e in una Configuration
- Parametri Formali di Ingresso, Uscita, Ingresso/Uscita di un POU:
  - Function block, funzione
- Variabili con Riferimento Diretto
  - E' possibile non utilizzare simboli di variabile, ma fare riferimento diretto a locazioni di memoria, ingressi e uscite.

Variabili con Riferimento Diretto

- Aree in cui è divisa la memoria:
  - **Input memory location (I).** E' relativa alla memoria in cui vengono copiati gli ingressi relativi a ciascun canale di ingresso. Ogni indirizzo della memoria corrisponde ad un particolare canale di ingresso (Immagine Processi Ingresso).
  - **Output memory location (Q).** E' relativa alla memoria in cui vengono copiate le uscite relative a ciascun canale di uscita. Ogni indirizzo della memoria corrisponde ad un particolare canale di uscita (Immagine Processi Uscita).
  - **Internal memory location (M).** E' relativa alla memoria in cui vengono memorizzati i risultati intermedi della computazione
    - M=Merker

Variabili con Riferimento Diretto

Sintassi variabili

- simbolo %
- uno tra i simboli: **I** (Input memory location), **Q** (Output memory location), **M** (Merker memory location)
- uno tra i simboli: **X** (bit), **B** (byte), **W** (word 16 bit), **D** (Double Word 32 bit), **L** (Long word 64 bit)
- Indirizzo (vedi lucido successivo)

Variabili con Riferimento Diretto

- indirizzo di memoria:
  - Generalmente l'unità di base è il byte
  - ***Ingressi/Uscite:***
    - **Moduli di I/O: tipicamente a byte caratterizzati da un numero di modulo** definito nella configuration
      - Indirizzo di modulo di I/O:
        - **numero_modulo** (byte iniziale) oppure
        - **numero_modulo.numero_bit**
    - **Memoria interna**:
      - Indirizzo di memoria:
        - **numero byte** (byte iniziale) oppure
        - **numero byte.numero bit**

Variabili con Riferimento Diretto

Esempi:

- %IX0.0 oppure %I0.0 (1 bit)
- %Q3.2 (1 bit)
- %IB0.0 oppure %IB0 (1 byte)
- %IB0.3 (1 byte)
- %QW0 (2 byte)
- %M0.0 (1 bit)
- %MB10 (1 byte)
- %MB20.1 (1 byte)
- %MW36 (2 byte)
- %MD40 (4 byte)

Attributi di Variabili

- RETAIN. Tale attributo permette alla variabile di riassumere il valore precedente ad una interruzione, quando il PLC viene nuovamente fatto ripartire (**WARM Start-Partenza a Caldo**)

```
VAR RETAIN
        Speed: REAL;
END_VAR;
```

Attributi di Variabili

- CONSTANT. L'attributo specifica che il valore attribuito alla variabile non può essere modificato. **Questo attributo non può essere usato per variabili esterne.**

```
VAR CONSTANT
        Speed: REAL:=12.3;
END_VAR;
```

- AT. Permette di attribuire ad una variabile simbolica, un indirizzo di memoria (I,Q,M) determinato.

```
VAR
        Status  AT  %IX0.0: BOOL;
END_VAR;
```

Partenze (Start) di un PLC

- **COLD (Freddo)**
  - Durante una partenza COLD tutte le variabili sono inizializzate a valori di default o a quelli ridefiniti dall'utente
- **WARM (Caldo)**
  - Durante una partenza WARM solo le variabili (compresi timers e contatori) NON-RETENTIVE (attributo RETAIN non presente) sono inizializzate ai valori di default o a quelli ridefiniti dall'utente.
  - Le variabili (compresi timers e contatori) con attributo RETAIN non vengono inizializzate ma continuano ad assumere l'ultimo valore precedente alla WARM start
- **HOT**
  - Durante una partenza HOT nessuna variabile viene inizializzata

Task nello Standard IEC 61131

- Concetti fondamentali della schedulazione di processi:
  - Un processo può trovarsi nello stato di pronto, di attesa o di esecuzione
  - Un processo nello stato di pronto viene posto in esecuzione in base alla politica di scheduling del S.O.
  - E' possibile assegnare una priorità ai processi in modo da aiutare il S.O. nella scelta del processo da porre in esecuzione tra i processi pronti

Task nello Standard IEC 61131

- **Task nello standard IEC 61131-3:**
  - Ad ogni Program e Function Block viene associato un task.
  - Ha il compito di "risvegliare" un processo (Program o Function Block) in attesa ponendolo nello stato di **pronto**
  - Offre all'utente differenti tipologie di controllo
  - Un programma senza task associato ha la più bassa priorità e viene posto in stato di pronto appena termina.

Task nello Standard IEC 61131

- Esistono tre tipi di tasks:
  - Cyclic tasks: sono attivati ad intervalli temporali
  - System (or Error) tasks: sono attivati all'occorrenza di un evento di sistema (o di un errore di sistema)
  - Event (or Interrupt) tasks: sono attivati all'occorrenza di certi eventi programmabili, ad esempio se una variabile ha raggiunto un certo valore o al sopraggiungere di un interrupt

Task nello Standard IEC 61131

- **La dichiarazione dei task è caratterizzata dai seguenti parametri:**
  - Task Cyclic
    - ✓ Definizione dell'intervallo in ms. Si noti che il task può essere eseguito anche dopo intervalli superiori all'intervallo specificato, in dipendenza del S.O.
  - Task Sistema/Interrupt
    - ✓ Definizione dell'evento di sistema o di interrupt (strettamente legato al PLC)
  - Priorità.
    - Viene assegnata una priorità al task, generalmente in ordine decrescente (0 la più alta). Il S.O. usa la priorità nella scelta tra processi pronti.