Ivan Iamoni (Ivan_Iamoni)

# L'INTERPRETAZIONE DEI DATI NEL PLC

25 January 2008

Presentazione

Chi frequenta il Forum di Electroportal ha avuto l'occasione di apprezzare la disponibilità dei suoi moderatori che si impegnano nel fornire risposte, conoscenze permettendo, ad ogni livello, al principiante come all'esperto in cerca di un'idea per superare un'incertezza od una difficoltà. A volte alcune domande sono molto generali e la risposta richiede un piccolo corso o comunque la stesura di un articolo. Come questo appunto che nasce proprio in tale modo. **Ivan Iamoni**, moderatore della sezione Automazione e strumentazione, esperto di applicazioni dei PLC che programma, installa ed insegna in ogni parte del mondo, spiega in questo suo lavoro la struttura dei dati all'interno di un controllore programmabile, in particolare del sistema Simatic. Egli tiene comunque a far notare che "*il documento e' stato inizialmente creato con lo scopo chiarire il principio del trattamento dei dati nei sistemi a microprocessore PLC, in riferimento ai prodotti Siemens per l'automazione, fermo restando che in generale, molti concetti sono correlati al campo dell'informatica, quindi indirizzabili anche ad una gamma di prodotti più vasta*".

**Introduzione**

Quando si arriva ad un punto della didattica inerente al PLC, superata la prima fase classica dello studio dell'apparecchio dal punto di vista della logica booleana elementare, dei diagrammi di flusso e contatti, e' obbligatorio fermarsi a riflettere su come il PLC, interpreta i dati al suo interno. Questo approfondimento aiuterà a comprendere meglio i successivi passi dell'apprendimento , che riguardano le funzioni avanzate, cioè:

- Funzioni matematiche con numeri interi
- Funzioni matematiche con numeri reali.
- Funzioni di scorrimento e rotazione.
- Funzioni di trasferimento.
- Funzioni di conversione.

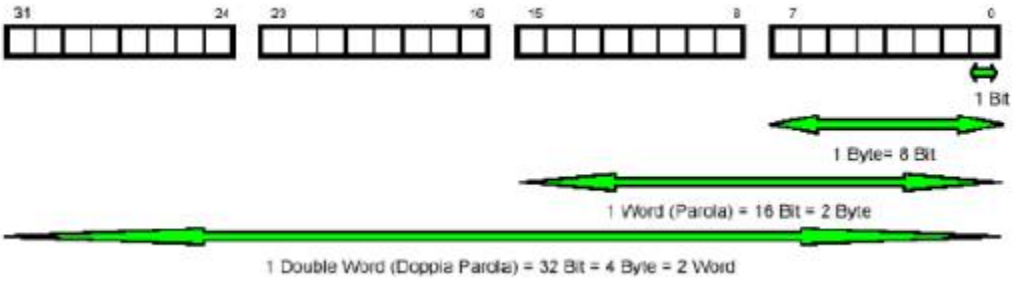
**Struttura della memoria dati , "area"**

**Fig.1-il BIT , la dimensione di base**


Il PLC immagazzina le informazioni attraverso un'area di memoria dati strutturata a bit, cioè l'elemento logico binario fondamentale: Il BIT consente di memorizzare solo 2 stati logici, quelli rappresentati nella Fig.1. Se volessimo memorizzare ad esempio un numero 3, dovremmo ricorrere alla associazione di più bit , che secondo la classica codifica binaria sarebbe: BIT 0 BIT 0 3 = 1 + 1 , in realtà con 2 bit, posso solo rappresentare un numero da 0 a 3. Comincia ad essere chiaro che ora, più complesso è il dato che devo memorizzare, maggiore sarà la quantità di BIT o memoria da mettere insieme. Ovviamente esistono delle dimensioni specifiche per ogni raggruppamento crescente di bit .

I raggruppamenti di Bit , si fanno con 8 bit alla volta, (dal bit 0 al bit 7 per chiarire) per potenze di 2. Il primo e più importante e il BYTE, perché? Agli albori dei computer, gli informatici, capirono che con 8 bit, cioè 128 combinazioni possibili si potevano rappresentare i numeri da 0 a 9, tutte le lettere dell'alfabeto, i segni di punteggiatura e dei simboli speciali, nasceva così il Byte che è stato poi preso internazionalmente come riferimento nelle unità di misura per la dimensione delle memorie dei computers , Kb=Kilobyte , Mb=Megabyte , Gb=Gigabyte, e così via.

**Fig.2 , i raggruppamenti di BIT , dimensioni delle aree di memoria.**




Nella figura 2 , i gruppi si fermano intenzionalmente alla doppia parola, cioè 32 bit. Ovviamente vi sono gruppi maggiori, salendo di livello, vi sono gli ARRAY e i dati di tipo STRUCT. La scelta di fermare il conto a 32 bit , è in funzione del fatto che è che qui si parla dei dati semplici. Nel gergo dei programmatori, la word si dice anche PAROLA e la Dword, DOPPIA PAROLA , letteralmente tradotto dall'Inglese. Per convenzione , nel PLC Siemens la numerazione dei bit, inizia sempre dal bit 0,

proseguendo verso sinistra sino al settimo ed ultimo degli otto bit. Dopo aver chiarito questo concetto di struttura della memoria dati, vediamo che tipo di informazioni vi si possono memorizzare dentro. La dimensione ovvero il numero di bit, limita il valore che può essere contenuto.

Nella figura 3, sono illustrati tutti i formati dei dati semplici disponibili per ogni dimensione specifica di area. Nella seconda colonna si trova il numero di bit necessari, massimo 32. Nella terza colonna vi sono le opzioni di formato consentite in funzione del numero di bit. Nella quarta colonna vengono riportati il valore massimo e minimo ammissibile. Nell'ultima colonna vi è un esempio della corretta sintassi nel programma del PLC.

**FIG.3 Tabella dei Tipi di Dati Semplici (Dal manuale di Automazione Simatic S7-300)**

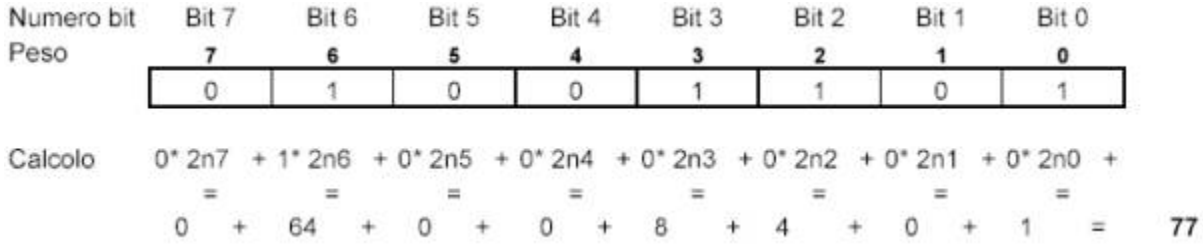
| Tipo e descrizione | Grandezza in bit | Opzioni di formato | Area e rappresentazione dei numeri (dal valore minore a quello maggiore) | Esempio |
|---|---|---|---|---|
| BOOL (Bit) | 1 | Testo booleano | TRUE/FALSE | TRUE |
| BYTE (Byte) | 8 | Esadeci-male | da B16#0 a B16#FF | L B#16#10<br>L byte#16#10 |
| WORD (Parola) | 16 | Cifra binaria | da 2#0 a 2#1111_1111_1111_1111 | 2#0000_0000_0001_0000 |
| | | Esadeci-male | da W#16#0 a W#16#FFFF | L W#16#1000<br>L word16#1000 |
| | | BCD | da C#0 a C#999 | L C#998 |
| | | Numero decimale senza segno | da B#(0,0) a B#(255,255) | L B#(10,20)<br>L byte#(10,20) |
| DWORD (Doppia parola) | 32 | Cifra binaria | da 2#0 a 2#1111_1111_1111_1111 1111_1111_1111_1111 | 2#1000_0001_0001_1000_ 1011_1011_0111_1111 |
| | | Numero esadeci-male | da DW#16#0000_0000 a DW#16#FFFF_FFFF | L DW#16#00A2_1234<br>L dword#16#00A2_1234 |
| | | Numero decimale senza segno | da B#(0,0,0,0) a B#(255,255,255,255) | L B#(1, 14, 100, 120)<br>L byte#(1,14,100,120) |
| INT (Numero intero) | 16 | Numero decimale con segno | da 32768 a 32767 | L 1 |
| DINT (Numero intero, 32 bit) | 32 | Numero decimale con segno | da L#-2147483648 a L#2147483647 | L L#1 |
| REAL (Numero in virgola mobile) | 32 | IEEE Numero in virgola mobile | Limite superiore: ±3.402823e+38<br>Limite inferiore: ±1.175 495e-38 | L 1.234567e+13 |
| S5TIME (Tempo SIMATIC) | 16 | Tempo S7 a intervalli di 10 ms (valore di default) | da S5T#0H_0M_0S_10MS a S5T#2H_46M_30S_0MS e S5T#0H_0M_0S_0MS | L S5T#0H_1M_0S_0MS<br>L S5TIME#0H_1H_1M_ 0S_0MS |
| TIME (Tempo IEC) | 32 | Tempo IEC a intervalli di 1 ms, numero intero con segno | da -T#24D_20H_31M_23S_ 648MS a T#24D_20H_31M_23S_ 647MS | L T#0D_1H_1M_0S_0MS<br>L TIME#0D_1H_1M_0S_ 0MS |
| DATE (Data IEC) | 16 | Data IEC in intevalli di 1 giorno | da D#1990-1-1 a D#2168-12-31 | L D#1994-3-15<br>L DATE#1994-3-15 |
| TIME_OF_DAY (Ora) | 32 | tempo ad intervalli di 1 ms | da TOD#0:0:0.0 a TOD#23:59:59.999 | L TOD#1:10:3.3<br>L TIME_OF_DAY#1:10:3.3 |
| CHAR | 8 | Simbolo ASCII | 'A','B', ecc. | L 'E' |

**Formato nei dati semplici**

Nella precedente tabella, si fa riferimento a diversi tipi di dati, e diversi **formati**. Soffermiamoci un attimo sulla definizione di formato. Nella nostra realtà quotidiana siamo abituati a rappresentare i numeri e le lettere dell'alfabeto nel modo che ci hanno sempre insegnato "1.2.43 A,B,F", mentre per farli "digerire" al PLC , vengono convertirli in formati specifici , questa operazione si chiama oltretutto "codifica".

**Binario** , è il formato, o codifica più nota, ed è la più importante funziona in questo modo, ad ogni bit partendo da destra viene associato un peso, ovvero un valore crescente. In ogni locazione in cui si trova il valore logico 1 si prende questo e lo si moltiplica per 2 elevato il peso relativo alla posizione in cui il bit è ad 1. Nel sistema Simatic , il numero Binario è preceduto dalla dichiarazione tipo : **2#**

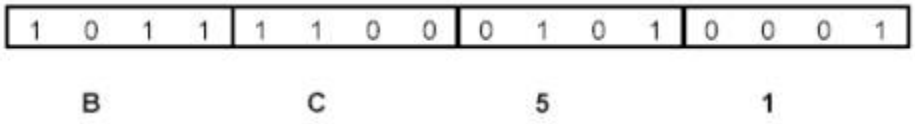
**Fig. 4 Esempio di rappresentazione del numero 77 in codice binario in un byte:**




**Esadecimale** è composto da una numerazione che va da 0 a 9 , più le prime sei lettere dell'alfabeto (maiuscole), totale sedici simboli , da cui deriva suo nome. Nel sistema Simatic, il numero esadecimale è preceduto dalla dichiarazione tipo : **16#**

Esempio di numerazione Esadecimale : 0_1_2_3_4_5_6_7_8_9_A_B_C_D_E_F. La sua introduzione, deriva dalla necessità di raffigurare più simboli possibili con solo 4 bit (16 possibilità), quindi si ha un dato raggruppato in quattro bit per volta, dove ogni gruppo contiene un singolo numero o lettera .

**Fig. 5 Esempio di rappresentazione del numero intero 48209 in esadecimale in una word:**



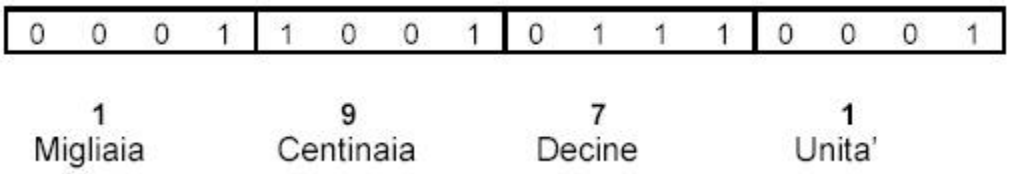

L'interpretazione del singolo raggruppamento di 4 bit segue le stesse regole del codice binario. Il formato esadecimale è utilizzabile in tutti i tipi di aree.

**BCD** è un tipo di formato, in cui ogni singolo gruppo di quattro bit binari può rappresentare un numero decimale da 0 a 9. Il nome è l'abbreviazione di **B**inary **C**ode to **D**ecimal. Il sistema BCD, nasce nei primi circuiti logici digitali, privi di

microprocessore, per la rappresentazione di numeri su display a 7 segmenti , a cui ogni singolo gruppo di 4 bit, corrispondeva un singolo numero visualizzato dal display. Nel sistema Simatic , il numero BCD è preceduto dalla dichiarazione tipo : **B#**

L'interpretazione del singolo raggruppamento di 4 bit segue le stesse regole del codice binario con il solo limite che il numero rappresentabile più alto è il 9. L'ordine dei gruppi di 4 bit a partire da sinistra, definisce i multipli X10 della cifra

**Fig. 6 Esempio di rappresentazione del numero BCD 1971 in una word:**




Nel sistema operativo del PLC, come anche nel BIOS dei PC, i dati riguardanti data e ora corrente sono memorizzati in formato BCD. Inoltre, è bene sottolineare che gli elementi di programmazione come i timers, che sono residenti nella memoria di sistema , vogliono anch'essi l'impostazione del tempo in formato BCD.

**Il rappresentazione dei Dati "semplici"**

Adeguandoci ad esigenze diverse, i dati semplici possono essere di **rappresentazioni** diverse. Diciamo subito che tra **BOOL** ,**BYTE** , **WORD** , **DWORD**, di cui abbiamo già parlato prima, la differenza sostanziale sta nella quantità di Bit. **INT**, è un numero decimale intero senza virgola contenuto in una word (16 bit) , con segno, positivo o negativo, Es. +1234 oppure –546. In questo formato sono ammessi solo numeri interi.

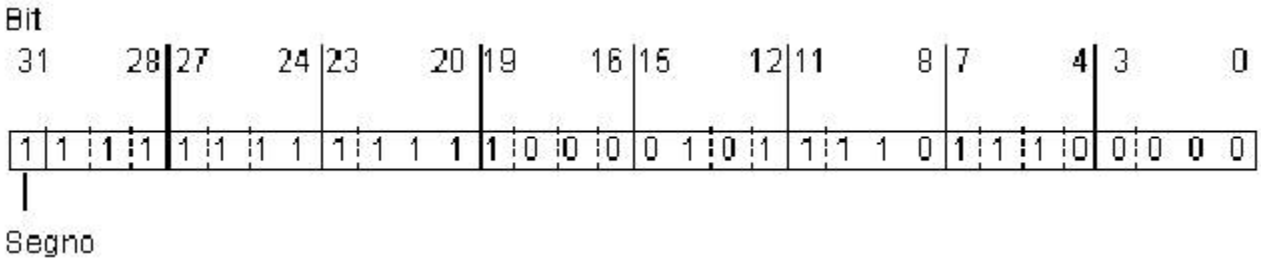
**Fig. 7 Esempio di rappresentazione del numero INT +44 in una word in codifica binaria :**


Siccome il segno +/- , non è rappresentabile attraverso la codifica dei bit, si è scelto arbitrariamente di rappresentarlo tramite l'ultimo bit a sinistra della parola. Il sedicesimo bit a sinistra nella parola è il segno, che si assume positivo, quando il bit è FALSE=0 e negativo quando il bit è TRUE=1. La necessità di "sacrificare" un BIT per indicare il segno del valore numerico, riduce come ovvio la capienza della word

che sarebbe massimo 65535, infatti si possono memorizzare solo valori compresi tra +27648 cioè tutti i BIT a zero tranne il bit di segno ad 1, ed il limite negativo – 32767, con tutti i bit ad 1 segno compreso.

Per via della sua struttura il dato **INT** e' memorizzabile solo in una **WORD. DINT** e' il fratello maggiore di INT, in termini di limite numerico essendo un numero intero a 32 bit. Anch'esso è un numero con segno +/-.Viene usato quando il dato oltrepassa in positivo o negativo il limite massimo di INT , in quanto il suo range è notevolmente maggiore, MAX. +2.147.483.647. e MIN +2.147.483.648

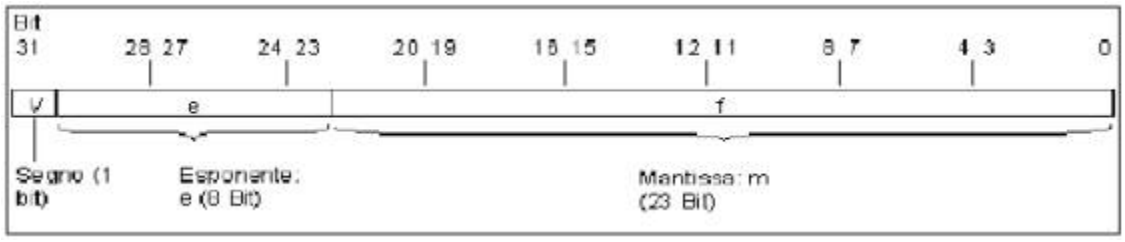
**Fig. 8 Esempio di rappresentazione del numero DINT in una Dword in codifica binaria :**




Il trentaduesimo ed ultimo bit a sinistra nella parola è il segno, che si assume positivo, quando il bit è FALSE=0 e negativo quando il bit è TRUE=1. Per via della sua struttura e dimensione il dato **DINT** è memorizzabile solo in una **DWORD. REAL**, sono i numeri che nella terminologia del programmatore chiamiamo "reali", mentre come viene descritto nella norma ANSI/IEEE , sono numeri in "virgola mobile" o in inglese più comunemente "Floating Point".

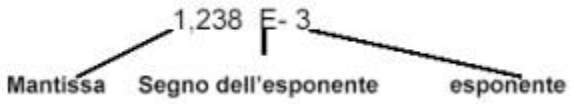
Ogni singolo numero reale e fatto da 3 distinti componenti, che messi insieme occupano lo spazio di 32 bit (doppia parola).

**Fig. 9 Composizione di un numero reale , Floating Point.**

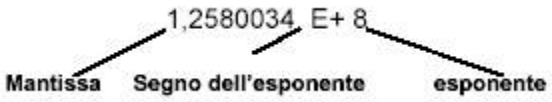



Mantissa, lunghezza 23 bit, è il numero, in seguito vi è l'esponente, che raggiunge 8 bit di lunghezza, ed in fine un bit che interpreta il segno. Quando questo bit del segno è=0 FALSE , il segno è positivo , negativo se invece il bit è TRUE. Il segno negativo, identifica subito che il numero è inferiore a 1, mentre il segno positivo, identifica che il numero è superiore a 1.

Prendiamo ad esempio, il numero 0,001238, in annotazione reale, si traduce in :



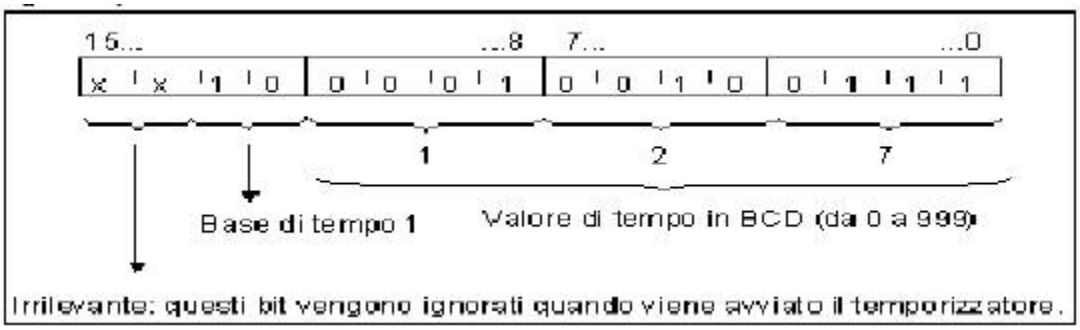

Altro esempio , il numero 125800034 , in annotazione reale , si traduce in :




Il ricorso ai numeri reali è necessario quando si effettuano calcoli con operazioni matematiche con numeri inferiori a zero o con valori elevatissimi . Il formato REAL , anche quando non è indispensabile garantisce una migliore precisione del calcolo e del risultato, a discapito però della maggior memoria usata, oppure quando il calcolo comporta l'uso di numeri inferiori allo zero, sia positivi che negativi.

**S5TIME** , è il formato classico del valore di impostazione dei timer nelle CPU Siemens, deriva dai vecchi sistemi a PLC , e viene tuttora usato. Il dato semplice occupa una DWORD e' memorizzato in formato BCD. Nel sistema Simatic , il dato S5TIME e' preceduto dalla dichiarazione tipo : **S5T#**

**Fig.10 Composizione del dato S5TIME**




Si vede dalla Fig. 5 che e' composto da 3 BYTE che rappresentano di valore di impostazione , ed un BYTE che contiene la base dei tempi. Poi vi sono 2 BIT inutilizzati , è una cosa tipica dei sistemi BCD, che spesso occupano più' memoria del necessario.

**Fig.11 Interpretazione dei BIT della base dei tempi.**

| BASE DI TEMPO | CODICE BINARIO | |
|---|---|---|
| | BIT n.13 | Bit n. 12 |
| 10 millisecond. | 0 | 0 |
| 100 millisecond. | 0 | 1 |
| 1 secondo | 1 | 0 |
| 10 secondi | 1 | 1 |

La base dei tempi specifica il valore di impostazione , come viene decrementato. Ad esempio nella programmazione, la sintassi è:

- **L S5T# 4 s** , carica nel registro il valore del timer pari a 4 secondi.
- **L S5T# 1s,50ms** , carica nel registro il valore del timer pari a 1,05 secondi.

Nella impostazione di tempi straordinariamente lunghi, si può usare anche la seguente sintassi:

**L S5T# aH_ bbM_ccS_dddMS**

dove **a** sono le ore , **bb** il numero di minuti, **cc** i secondi, **ddd** i millisecondi.

Essendo i timers residenti nella memoria di sistema del PLC, per ovvi motivi di precisione che non deve essere influenzata dalla lunghezza del programma nella memoria RAM, il dato deve essere per forza in BCD.

**DATE** è il tipo di dati della data memorizzata nel PLC. Corrisponde alla data corrente aggiornata ciclicamente ad intervalli singoli di 1giorno. Il suo formato, corrisponde alla norma IEC. Nel sistema Simatic, il DATE è preceduto dalla dichiarazione tipo : **D#**

**Fig.12 Composizione del dato DATE**


Essendo anche la data residente nella memoria di sistema del PLC, per gli stessi motivi del dato S5T# il dato deve essere per forza in BCD ed occupa una word (16 bit).

**TIME_OF_DAY** è l'ora del giorno memorizzato nel PLC.

Corrisponde all'ora regolarmente aggiornata ad intervalli di 1 ms. Nel sistema Simatic , il TIME_OF_DAY è preceduto dalla dichiarazione tipo : **TOD#**

**Fig.13 Strutura del dato Time Of Day**

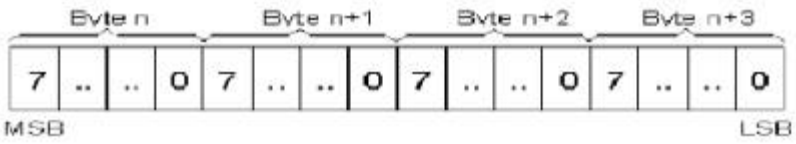



**Fig.14 Composizione del dato Time Of Day**


Essendo anche l'orologio residente nella memoria di sistema del PLC, per gli stessi motivi del dato S5T# il dato deve essere per forza in BCD ed occupa una doppia word (32 bit). Nello scrivere l'ora, si può anche omettere, di aggiornare il valore dei millisecondi.

**CHAR**, è il tipo di dato corrispondente ai caratteri della tabella ASCII, come ad esempio A, B, C ecc. Ogni dato CHAR, occupa un singolo Byte. Il numero memorizzato nel Byte in formato CHAR, ha un suo corrispettivo simbolo nella tabella ASCII standard. Il ricorso al formato CHAR, si fa quando si compongono stringhe di dati non numerici , come ad esempio se vogliamo memorizzare del PLC il nome "Francesco", ma essendo la stringa un tipo di dato "complesso", non proseguo oltre, almeno per ora.

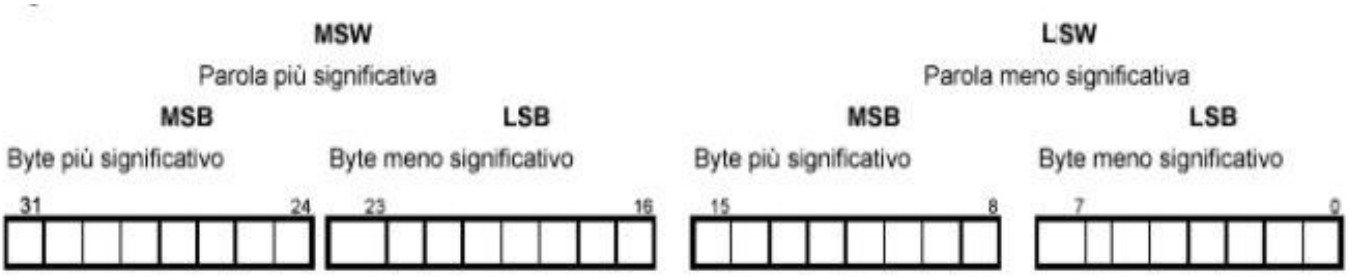
**Byte "piu' significativo e "meno significativo"**

Una piccola precisazione su come i diversi tipi di dati dal BYTE in su vengono scritti nei loro corrispettivi formati , nella memoria del PLC. Qui entra in gioco la definizione **LSB** ed **MSB** e l'ordine di scrittura. I sistemi SIMATIC, sono per definizione "Big Endian" , che identifica quali sono il Byte più e meno significativi , per dare una regola nel tipo di accesso in lettura e scrittura.

- **MSB** e' l'abbreviazione Inglese di "Most Significant Byte" , "Byte più significativo"
- **LSB** e' l'abbreviazione Inglese di "Less Significant Byte" , "Byte meno significativo"
- **MSW** e' l'abbreviazione Inglese di "Most Significant Word" , "Parola più significativa"
- **LSW** e' l'abbreviazione Inglese di "Less Significant Word" , "Parola meno significativa"

Di regola MSB , ed LSB nel PLC Siemens sono ordinati in questo modo:

**Fig.15 Ordinamento MSB , LSB**




Non tutti i sistemi di automazione sono "Big Endian", ve ne sono molti che sono "Little Endian" in cui praticamente l'ordine di scrittura dai Byte è alla rovescia. Nel dettaglio di ogni singolo sistema l'ordine dei bit e la dimensione o il significato del dato non cambia . Allo scopo del programmatore, cioè creare un programma, in pochissimi casi si tiene conto di questo aspetto della memoria. Discorso diverso è quando due dispositivi di diverso tipo, come un PLC ed un pannello operatore si scambiano le informazioni tra loro. In questo caso, se i due dispositivi, non usano la stessa metodologia di scrittura, vi possono essere degli errori gravi di interpretazione dei dati da e verso il pannello se non si tiene conto di questo dettaglio. La differenza tra "Big Endian" e "Little Endian" , è solo nell'architettura del microprocessore usato nel PLC.

**Conclusione**

Si è parlato in questo corso, di formato e rappresentazione dei "dati semplici". I dati semplici, dal Byte alla Doppia Word, sono usati oltre che nelle operazioni di logica booleana, nello sviluppo dei calcoli tramite il programma. A completamento possiamo dire che ad un dato si associa sempre un formato specifico a seconda di nostri gusti e delle nostre esigenze. Fatto questo, i dati vengono assegnati come "operando" nelle formule di calcolo. Una formula o funzione matematica qualsiasi ha bisogno 2 operandi. Bisogna sempre tenere presente che gli operandi di una singola operazione devono necessariamente essere "omogenei" nella loro rappresentazione e formato. Ad esempio, la somma tra il numero in un byte e quello in una word non è consentito. Oppure la moltiplicazione di un numero intero 16 BIT con uno in virgola mobile, non è consentita. Oppure ancora estrarre la radice quadrata da un numero intero a 16 BIT non è consentito, e via dicendo. Solitamente, errori del genere di sintassi o errata assegnazione degli operandi, vengono subito visualizzati dal compilatore dell'editor che si usa per creare il programma. Per ovviare a questi inconvenienti, esiste un set di operazioni, chiamato , "conversione", che permette di trasformare dati in formati o rappresentazioni diverse , per adattarli all'esigenza specifica del calcolo.