



Programmazione IEC 61131 su SlimLine

ELSIST S.r.l.
Sistemi in elettronica

Via G. Brodolini, 15 (Z.I.)
15033 CASALE M.TO
ITALY

Internet: <http://www.elsist.it>
Email: elsist@elsist.it

TEL. (39)-0142-451987
FAX (39)-0142-451988

INDICE

1 - SlimLine	6
1.1 - Risorse del sistema.....	7
2 - Architettura memoria.....	8
2.1 - Memoria di backup (Retain).....	9
2.2 - Accesso alla memoria.....	10
3 - Definizione tipo dati.....	11
3.1 - FILEP, file pointer.....	11
3.2 - SYSSERIALMODE, modo comunicazione porta seriale.....	11
3.3 - SYSCANMESSAGE, messaggio CAN.....	11
4 - Variabili di sistema.....	12
4.1 - Variabili sola lettura (System variables).....	13
4.2 - Variabili sola lettura (Report variables).....	14
4.3 - Variabili lettura e scrittura (System variables).....	15
5 - Definizioni dati.....	16
5.1 - Variable types, definizione tipo variabili.....	16
5.2 - Task ID definition, identificatore di task PLC.....	16
5.3 - TermIO, definizioni per terminale di I/O.....	16
5.4 - Serial mode definition, definizioni modo seriale.....	17
6 - Protocollo modbus.....	18
6.1 - Accesso variabili da modbus.....	18
6.1.1 - Lettura variabili da modbus.....	18
6.1.2 - Scrittura variabili da modbus.....	18
6.2 - Accesso Real time clock da modbus.....	19
6.2.1 - Lettura RTC da modbus.....	19
6.2.2 - Scrittura RTC da modbus.....	20
6.3 - Accesso Epoch time da modbus.....	20
6.3.1 - Lettura Epoch time da modbus.....	20
6.3.2 - Scrittura Epoch time da modbus.....	20
7 - Funzioni ed FB.....	21
- Funzioni.....	21
- Function Blocks.....	21
7.1 - Funzioni ed FB embedded.....	21
7.2 - Librerie.....	22
7.2.1 - Import libreria.....	22
7.2.2 - Link a libreria.....	23
7.2.3 - Import oggetti.....	24
7.2.4 - Considerazioni su link a libreria e su import oggetti.....	25
7.3 - Protezione funzioni e blocchi funzione.....	26
7.4 - Funzioni ed FB per gestione Flip/Flop.....	27

7.4.1 - F_TRIG, Falling edge trigger.....	27
7.4.2 - R_TRIG, Raising edge trigger.....	28
7.4.3 - RS, Reset/Set flip flop.....	29
7.4.4 - SR, Set/Reset flip flop.....	30
7.5 - Funzioni ed FB per gestione timers.....	31
7.5.1 - eTOF, Timer Off.....	31
7.5.2 - eTON, Timer On.....	32
7.5.3 - eTP, Timer pulse.....	33
7.6 - Funzioni ed FB per gestione counters.....	34
7.6.1 - CTD, Counter Down.....	34
7.6.2 - CTU, Counter Up.....	36
7.6.3 - CTUD, Counter Up/Down.....	37
7.7 - Funzioni ed FB per conversione dati.....	39
7.7.1 - VBitTest, Variable bit test.....	39
7.7.2 - VBitSet, Variable bit set.....	40
7.7.3 - BitToByte, Bit to byte conversion.....	41
7.7.4 - ByteToBit, Byte to bit conversion.....	43
7.7.5 - ByteToWord, Byte to word conversion.....	44
7.7.6 - WordToByte, Word to byte conversion.....	45
7.7.7 - DoubleToWord, Double to word conversion.....	46
7.7.8 - WordToDouble, Word to double conversion.....	47
7.7.9 - ToLower, Uppercase to lowercase letter conversion.....	48
7.7.10 - ToUpper, Lowercase to uppercase letter conversion.....	49
7.8 - Funzioni ed FB di utilità sistema.....	50
7.8.1 - SysGetSysTime, get system time.....	50
7.8.1.1Calcolo timeout.....	50
7.8.1.2Semplice cronometro.....	51
7.8.2 - SysSetTaskLpTime, set task loop time.....	52
7.8.3 - SysGetRandom, get random number.....	53
7.8.4 - SysGetLastError, get last error.....	54
7.8.5 - SysPCodeAccept, accepts the protection code.....	55
7.8.6 - SysGetCrc, get CRC value.....	56
7.8.7 - SysMAlloc, Memory allocation.....	58
7.9 - Funzioni ed FB per gestione Data/Ora.....	59
7.9.1 - SysETimeToDate, epoch time to date conversion.....	59
7.9.2 - SysDateToETime, date to epoch time conversion.....	61
7.10 - Funzioni ed FB per gestione terminale di I/O.....	63
7.10.1 - Sysfopen, file open.....	63
7.10.2 - Sysfclose, file close.....	65
7.10.3 - Sysfgetc, get character from file.....	66
7.10.4 - Sysfputc, put character to file.....	67
7.10.5 - Sysfread, read data from file.....	69
7.10.6 - Sysfwrite, write data to file.....	70
7.10.7 - SysGetlChars, get input available characters from file.....	71
7.10.8 - SysGetOSpace, get output available space on file.....	72
7.10.9 - SysGetRxBSize, get file Rx input buffer size.....	73
7.10.10 - SysGetTxBSize, get file Tx output buffer size.....	74
7.10.11 - SysFIBfClear, file input buffer clear.....	75
7.10.12 - SysFOBfClear, file output buffer clear.....	76

7.10.13 - SysFOBfFlush, file output buffer flush.....	77
7.10.14 - SysVarprintf, variable print to file.....	78
7.11 - File system.....	79
7.11.1 - Sysremove, file remove.....	80
7.11.2 - Sysrename, file rename.....	81
7.11.3 - Sysfilelength, file length.....	82
7.11.4 - Sysfseek, file seek.....	83
7.11.4.1 Scrittura file di log.....	85
7.12 - Funzioni ed FB per gestione porta seriale.....	86
7.12.1 - SysGetSerialMode, get serial mode.....	87
7.12.2 - SysSetSerialMode, set serial mode.....	89
7.12.3 - SetSMode, Set serial mode.....	90
7.12.4 - SysGetSerialCTS, get serial CTS signal status.....	92
7.12.5 - SysSetSerialDTR, set DTR signal status.....	93
7.13 - Funzioni ed FB per gestione CAN bus.....	94
7.13.1 - SysIsCANRxTxAv, checks if CAN Rx or Tx is available.....	94
7.13.2 - SysCANRxMsg, receives a CAN message.....	95
7.13.3 - SysCANTxMsg, transmit a CAN message.....	96
7.14 - Funzioni ed FB per gestione stringhe.....	97
7.14.1 - eLEN, string length.....	97
7.14.2 - eFIND, string find.....	98
7.14.3 - MemSet, memory set.....	99
7.14.4 - SysVarsnprintf, variable print to string.....	100
7.14.5 - SysVarsscanf, extracts values from string.....	102
7.15 - Funzioni ed FB per gestione moduli periferici.....	103
7.15.1 - SysPhrInfos, get infos from peripheral modules.....	103
7.15.2 - SysGetPhrDI, get peripheral digital input.....	104
7.15.3 - SysSetPhrDO, set peripheral digital output.....	107
7.15.4 - SysGetAnInp, get analog input.....	110
7.15.5 - SysSetAnOut, set analog output.....	113
7.15.6 - SysGetCounter, get counter.....	115
7.15.7 - SysGetEncoder, get encoder input.....	117
7.15.8 - SysPhrVRd, read variable from peripheral module.....	119
7.15.9 - SysPhrVWr, write variable to peripheral module.....	120
7.15.10 - CPUModuleIO, CPU module I/O management.....	121
7.16 - Funzioni ed FB di utilità generale.....	122
7.16.1 - DB100AddOffset, returns DB100 address offset.....	122
7.16.2 - BLINK, blink command.....	123
7.16.3 - BlinkValue, blink out value.....	124
7.16.4 - ModbusRTUMaster, modbus Rtu master.....	125
7.16.5 - MDBRTUMASTER, modbus Rtu master.....	127
7.16.6 - ModbusRTUSlave, modbus Rtu slave.....	129
7.16.7 - ONOFFCYCLE, on/off cycle with random times.....	132
7.16.8 - PIDMng, PID management.....	134
7.16.9 - PWMOut, PWM output management.....	136
7.16.10 - SysDMXMng, DMX management.....	137
7.16.11 - IOEncoder, incremental encoder over I/O.....	139
7.16.12 - GetISO1155Crc, calculate CRC according ISO1155.....	140
7.16.13 - IODataExchange, exchange data by using logic I/O.....	141

7.16.14 - Average, value average.....	143
7.17 - Protocollo DLMS, o IEC 62056-21.....	144
7.17.1 - IEC62056_21Rd, IEC62056-21 protocol read.....	145
7.18 - Funzioni ed FB gestione modem (eModemLib).....	147
7.18.1 - ModemCore, modem core management.....	148
7.18.2 - ModemSMSReceive, receive a SMS message.....	150
7.18.3 - ModemSMSRxCmd, receive a SMS command.....	151
7.18.4 - ModemSMSSend, send a SMS message.....	152
7.19 - Funzioni ed FB gestione One-Wire (ePLC1WireLib).....	153
7.19.1 - OWireMng, One-Wire management.....	154
7.19.2 - OWRdIdentifier, One-Wire read ROM identifier.....	155
7.19.3 - OWRdTemperature, One-Wire read temperature.....	157
7.19.4 - OWRdHumidity, One-Wire read humidity.....	159
7.20 - Funzioni ed FB gestione networking.....	161
7.20.1 - SysIPReach, IP address is reachable.....	162
7.20.2 - SysSkListen, Socket listen.....	163
7.20.3 - SysUDPSktRcv, UDP socket receive.....	165
7.20.4 - SysUDPSktSend, UDP socket send.....	166
7.20.5 - UDPDataTxfer, UDP data transfer.....	168
7.21 - Funzioni ed FB supporto prodotti Hw Group (eHwGSpLib).....	170
7.21.1 - STESnmpAcq, STE termometer acquisition over SNMP.....	171
7.21.2 - HTempBox, HTemp Box2-485 acquisition.....	173
7.22 - Funzioni ed FB supporto protocollo NMEA (eNMEALib).....	175
7.22.1 - NMEASInterface, NMEA system interface.....	176
7.22.2 - GLLSentence, Geographic Position sentence.....	177
7.22.3 - MWVSentence, Wind Speed and Angle sentence.....	179
7.23 - Funzioni ed FB supporto inverter Power One (ePowerOneLib).....	180
7.23.1 - AuroraDSPMeasure, Aurora measure request to DSP.....	181
7.23.2 - AuroraCEnergy, Aurora cumulated energy reading.....	184
7.23.3 - CRCPolynomial, CRC polinomial calculation.....	186
8 - Tips and tricks.....	187
8.1 - Swap variabile DWORD.....	187
8.2 - Swap variabile WORD.....	187
8.3 - Swap variabile BYTE.....	188
8.4 - Espandere DWORD in 32 BOOL.....	189
8.5 - Comprimere 32 BOOL in DWORD.....	190
8.6 - Definire caratteri ascii non stampabili.....	191
8.7 - Rx/Tx dati su stream.....	192
9 - Appendici.....	193
9.1 - Tabella istruzioni IL.....	193
9.2 - Operatori linguaggio ST.....	194
9.3 - Statements linguaggio ST.....	195
9.4 - Conversione tipo dati.....	196
9.5 - Errori di esecuzione.....	197
9.6 - Tabella codici Ascii.....	199
9.6.1 - Tabella codici ASCII standard.....	199
9.6.2 - Tabella codici ASCII estesi.....	200
10 - Esempi di programmazione.....	201
10.1 - Biblioteca esempi.....	201
10.2 - Definizioni I/O logici negli esempi.....	202

10.3 - FB gestione lettura/scrittura registri.....	203
10.3.1 - Utilizzo FB Lettura/Scrittura registri.....	205
10.4 - FB gestione ciclo On/Off con tempi random.....	206
10.4.1 - Utilizzo FB ciclo On/Off con tempi random.....	207
10.5 - Blocco funzione MiniDMX.....	208
10.5.1 - Programma DMXUse.....	209

1 SlimLine

Il modulo CPU ARM7 SlimLine è l'unità centrale base di questa famiglia di controllori programmabili (PLC). Il dispositivo consente la gestione dei moduli di espansione I/O SlimLine attraverso il bus di espansione I2C™ High-Speed, ed incorpora altresì un alimentatore ad alta efficienza, dimensionato per il collegamento di fino a 16 moduli di espansione.

Il modulo è ingegnerizzato in un compatto contenitore in PC/ABS autoestinguente UL94 V-0 e riciclabile; esso è direttamente innestabile su guide DIN secondo gli standard EN50022/IEC60715 ed accetta tensioni di alimentazione nel range 10-30Vdc. A livello hardware questa CPU è dotata di:

- 1 Porta Ethernet 10/100 BaseT RJ45 (Solo versioni Full)
- 2 Porte COM RS232
- 1 porta MiniUSB
- 1 Bus di campo RS485(ModBus) o CAN bus (Solo versioni Full)
- 2 Input digitali isolati
- 2 Output digitali isolati
- 1 slot micro-SD card (Solo versioni Full)
- 1 bus di espansione I2C High-speed

Grazie all'utilizzo di memorie non volatili in tecnologia FRAM (RAM ferroelettriche), è stato possibile una riduzione sensibile del size della batteria al litio, minimizzando l'impatto ambientale a fine vita del prodotto. La tecnologia FRAM garantisce inoltre un numero di cicli di scrittura praticamente illimitati, migliorando quindi l'affidabilità del sistema, ed un bassissimo consumo di energia anche nelle fasi di scrittura.

SlimLine è programmabile nei 5 linguaggi previsti dalla norma IEC61131-3, attraverso il tool di sviluppo LogicLab, scaricabile ed utilizzabile gratuitamente e senza limitazioni.



1.1 Risorse del sistema

Esistono modi diversi per accedere alle risorse del sistema.

Gli **I/O logici** presenti sul modulo CPU e sui moduli di estensione sono gestiti tramite le funzioni [SysGetPhrDI](#) e [SysGetPhrDO](#).

Gli I/O logici dei moduli di estensione sono automaticamente gestiti in immagine di processo dal sistema operativo che provvede a trasferire lo stato di tutti gli ingressi logici nella immagine degli ingressi in memoria di sistema ed a trasferire il valore presente nella immagine delle uscite dalla memoria di sistema alle uscite logiche.

Quindi testando lo stato della immagine di memoria degli ingressi logici si testa lo stato del relativo punto di ingresso (Esempio **IX0.0** corrisponde all'ingresso 0 del modulo 0, **IX1.5** corrisponde all'ingresso 5 del modulo 1).

Scrivendo lo stato nella immagine di memoria delle uscite logiche si setta lo stato del relativo punto di uscita (Esempio **QX0.0** corrisponde all'uscita 0 del modulo 0, **QX1.5** corrisponde all'uscita 5 del modulo 1).

Gli **I/O analogici** presenti sui moduli di estensione sono gestiti tramite le funzioni [SysGetAnInp](#) e [SysSetAnOut](#)

I **contatori** sul modulo CPU e sui moduli di estensione sono acquisiti dal blocco funzione [SysGetCounter](#).

Gli ingressi **encoder** presenti sui moduli di estensione sono acquisiti dal blocco funzione [SysGetEncoder](#).

Per accedere alle **porte seriali** presenti sul modulo CPU occorre utilizzare la funzione [Sysfopen](#) definendo il nome della porta da utilizzare. Esistono moduli di estensione che sono provvisti di porte seriali, l'accesso a queste porte è esattamente uguale a quello delle porte presenti sul modulo CPU, occorre utilizzare la funzione [Sysfopen](#) definendo il nome della porta da utilizzare. Si utilizza la definizione **PCOMx.y** dove con **x** si indica l'indirizzo del modulo e con **y** il numero di porta presente sul modulo. (Esempio **PCOM0.0** definisce la porta 0 presente sul modulo 0, **PCOM1.2** definisce la porta 2 presente sul modulo 1).

COM0	Porta seriale RS232 (Su modulo CPU)
COM1	Porta seriale RS232 (Su modulo CPU)
COM2	Porta seriale RS485 (Su modulo CPU)
PCOMx.y	Porta seriale x sul modulo di estensione y . (PCOM0.2 definisce porta 0 sul modulo 2).

Il **CAN bus** viene gestito dalle funzioni [SysCANRxMsg](#) e [SysCANTxMsg](#).

2 Architettura memoria

La memoria del sistema è così suddivisa:

DB	Dimensione	Descrizione
IX0	32 Bytes	Ingressi logici modulo 00 (R)
IX1	32 Bytes	Ingressi logici modulo 01 (R)
IX2	32 Bytes	Ingressi logici modulo 02 (R)
IX3	32 Bytes	Ingressi logici modulo 03 (R)
IX4	32 Bytes	Ingressi logici modulo 04 (R)
IX5	32 Bytes	Ingressi logici modulo 05 (R)
IX6	32 Bytes	Ingressi logici modulo 06 (R)
IX7	32 Bytes	Ingressi logici modulo 07 (R)
IX8	32 Bytes	Ingressi logici modulo 08 (R)
IX9	32 Bytes	Ingressi logici modulo 09 (R)
IX10	32 Bytes	Ingressi logici modulo 10 (R)
IX11	32 Bytes	Ingressi logici modulo 11 (R)
IX12	32 Bytes	Ingressi logici modulo 12 (R)
IX13	32 Bytes	Ingressi logici modulo 13 (R)
IX14	32 Bytes	Ingressi logici modulo 14 (R)
IX15	32 Bytes	Ingressi logici modulo 15 (R)
QX0	32 Bytes	Uscite logiche modulo 00 (R/W)
QX1	32 Bytes	Uscite logiche modulo 01 (R/W)
QX2	32 Bytes	Uscite logiche modulo 02 (R/W)
QX3	32 Bytes	Uscite logiche modulo 03 (R/W)
QX4	32 Bytes	Uscite logiche modulo 04 (R/W)
QX5	32 Bytes	Uscite logiche modulo 05 (R/W)
QX6	32 Bytes	Uscite logiche modulo 06 (R/W)
QX7	32 Bytes	Uscite logiche modulo 07 (R/W)
QX8	32 Bytes	Uscite logiche modulo 08 (R/W)
QX9	32 Bytes	Uscite logiche modulo 09 (R/W)
QX10	32 Bytes	Uscite logiche modulo 10 (R/W)
QX11	32 Bytes	Uscite logiche modulo 11 (R/W)
QX12	32 Bytes	Uscite logiche modulo 12 (R/W)
QX13	32 Bytes	Uscite logiche modulo 13 (R/W)
QX14	32 Bytes	Uscite logiche modulo 14 (R/W)
QX15	32 Bytes	Uscite logiche modulo 15 (R/W)
MX0	512 Bytes	Variabili di sistema sola lettura (R)
MX1	512 Bytes	Variabili di sistema lettura/scrittura (R/W)
MX100	4096 Bytes	Memoria utente (R/W). Da indirizzo 2048 a 4095 i dati sono ritentivi.

2.1 Memoria di backup (Retain)

SlimLine dispone di 2048 bytes di memoria ritentiva nell'area memoria utente **MX100** ed ulteriori 2000 bytes di memoria ritentiva a disposizione utente per allocare variabili mnemoniche.

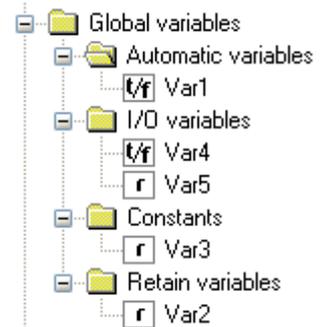
Le variabili allocate nella memoria utente **MX100** da indirizzo 2048 ad indirizzo 4095 sono ritentive, mantengono il loro valore anche allo spegnimento del sistema.

Qualsiasi variabile mnemonica a cui viene attribuito l'attributo **RETAIN**, manterrà il suo valore anche allo spegnimento del sistema. Da quanto detto precedentemente l'area totale allocabile per le variabili **RETAIN** è di 2000 bytes.

	Name	Type	Address	Group	Array	Init value	Attribute	Description
1	Var1	BOOL	Auto		No	FALSE	..	Sample variable 1
2	Var2	REAL	Auto		No	0	RETAIN	Sample variable 2
3	Var3	REAL	Auto		No	12.5	CONSTANT	Sample variable 3
4	Var4	BOOL	%MD100.0		No	FALSE	..	Sample variable 4
5	Var5	REAL	%MD100.2048		No	0	..	Sample variable 5

Come si vede dalla foto la variabile **Var2** è dichiarata con l'attributo **RETAIN** e manterrà il suo valore anche allo spegnimento del sistema. La variabile **Var5** allocata nella memoria utente **MD100.2048** pur essendo ritentiva non necessita dell'attributo **RETAIN** in quanto è implicito dalla sua allocazione.

Nella finestra di navigazione progetto, tutte le variabili globali sono suddivise in base alla loro definizione, e come si nota nella cartella delle variabili ritentive figureranno solo le variabili mnemoniche **Var2** e non le variabili allocate nella memoria utente **Var5** pur essendo anch'essa di tipo ritentivo.



2.2 Accesso alla memoria

IX: Immagine di processo ingressi logici

SlimLine esegue la lettura degli ingressi logici all'inizio di ogni loop di esecuzione programma. E' possibile accedere a quest'area utilizzando variabili di tipo **BOOL**, ogni indirizzo rappresenta lo stato booleano del relativo ingresso logico. L'indirizzo **IX0.0**, rappresenta lo stato dell'ingresso 0 del modulo 0, l'indirizzo **IX5.12**, rappresenta lo stato dell'ingresso 12 del modulo 5.

QX: Immagine di processo uscite logiche

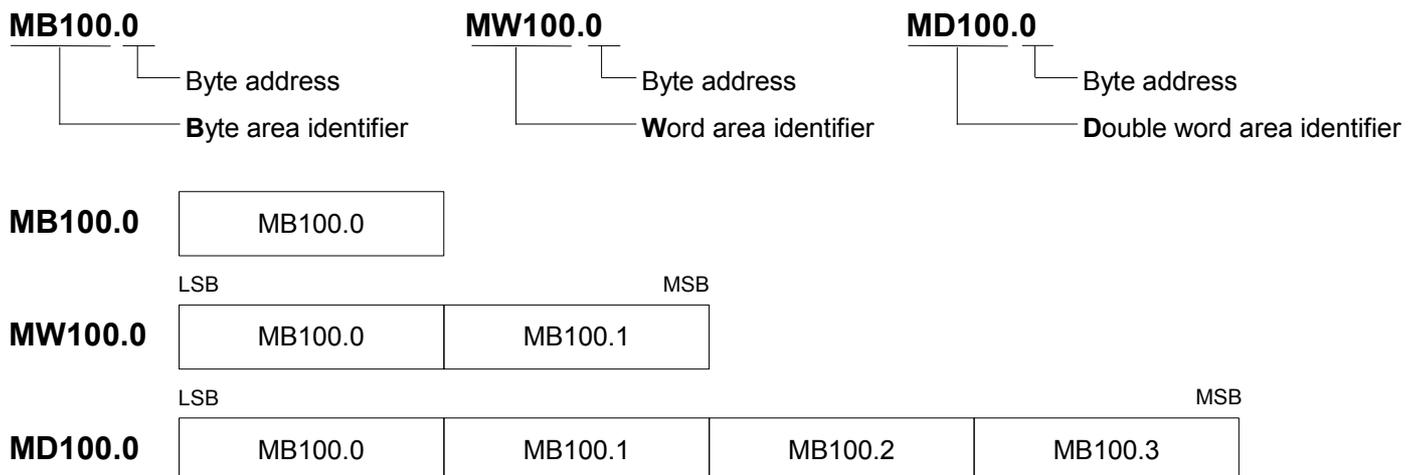
SlimLine esegue la scrittura delle uscite logiche alla fine di ogni loop di esecuzione programma. E' possibile accedere a quest'area utilizzando variabili di tipo **BOOL**, ogni indirizzo rappresenta lo stato booleano della relativa uscita logica. L'indirizzo **QX0.0**, rappresenta lo stato dell'uscita 0 del modulo 0, l'indirizzo **QX5.12**, rappresenta lo stato dell'uscita 12 del modulo 5.

MX: Area di memoria

A queste aree è possibile accedere utilizzando tutti i tipi di variabili definiti. Siccome tutte le variabili utilizzano la stessa area di memoria, occorre prestare attenzione alla dimensione in bytes del tipo definito per evitare sovrapposizioni di indirizzo.

Ad esempio una variabile **DWORD** allocata ad indirizzo **MX100.10** utilizzerà anche lo spazio di memoria **MX100.11**, **MX100.12** ed **MX100.13**. Quindi allocando una variabile **BYTE** all'indirizzo **MX100.11** si andrebbe ad occupare uno spazio di memoria già utilizzato dalla variabile precedente.

E' comunque possibile allocare variabili sovrapponendone l'indirizzo, esempio allocare due variabili **BYTE** sugli stessi indirizzi di una variabile **WORD** per andarne a considerare la parte MSB od LSB. Oppure allocare due variabili **WORD** sugli stessi indirizzi di una variabile **DWORD** per andarne a considerare la parte MSW od LSW. Riporto una semplice tabella esplicativa.



Attenzione! SlimLine è basato su architettura ARM e questo tipo di architettura assume che:

Le variabili a 16 bits, **WORD**, **INT**, **UINT** siano allocate in memoria ad indirizzi **divisibili per 2**. Quindi una variabile a 16 bits potrà essere allocata ad esempio ad indirizzo MW100.32 ma non ad indirizzo MW100.33.

Le variabili a 32 bits **DWORD**, **DINT**, **UDINT**, **REAL** siano allocate ad indirizzi **divisibili per 4**. Quindi una variabile a 32 bits potrà essere allocata ad esempio ad indirizzo MD100.32 ma non ad indirizzo MD100.33, MD100.34, MD100.35.

3 Definizione tipo dati

Oltre ai dati standard definiti dalla normativa IEC61131 sono stati definiti altri tipi di dato che possono essere utilizzati nel programma PLC.

3.1 FILEP, file pointer

Questo tipo di dati è utilizzato dalle funzioni che eseguono accesso alle risorse di I/O del sistema, una variabile di tipo **FILEP** punta ad una risorsa utilizzata per effettuare la lettura e/o scrittura di dati. Un esempio di file pointer è il puntatore ad una porta seriale od un file su disco.

3.2 SYSSERIALMODE, modo comunicazione porta seriale

Questo tipo di dati è utilizzato dalle funzioni che eseguono lettura ed impostazione modo di comunicazione su porta seriale. Il tipo dati contiene tutte le informazioni per caratterizzare la comunicazione sulla porta seriale.

Name	Type	Description
Baudrate	UDINT	Valore di baud rate porta seriale (da 300 a 115200 baud)
Parity	STRING[1]	Tipo di parità, valori possibili "E" pari, "O" dispari, "N" nessuna.
DataBits	USINT	Numero di bit frame dato, valori possibili 7, 8.
StopBits	USINT	Numero di bit di stop, valori possibili 1, 2.
DTRManagement	USINT	Modo di gestione del segnale DTR sulla porta seriale, vedi Serial mode definition .
DTRComplement	BOOL	FALSE: DTR normale, TRUE: DTR complementato.
EchoFlush	BOOL	FALSE: I dati trasmessi sono ritornati in ricezione. TRUE: I dati trasmessi sono ignorati.
DTROnTime	UINT	Tempo di attesa trasmissione caratteri su porta seriale dopo attivazione segnale DTR (mS). Questo parametro assume significato solo se DTR management è impostato nel modo DTR_AUTO_W_TIMES , vedi Serial mode definition .
DTROffTime	UINT	Tempo di attesa dopo trasmissione ultimo dato e disattivazione segnale DTR (mS). Questo parametro assume significato solo se DTR management è impostato nel modo DTR_AUTO_W_TIMES , vedi Serial mode definition .

3.3 SYSCANMESSAGE, messaggio CAN

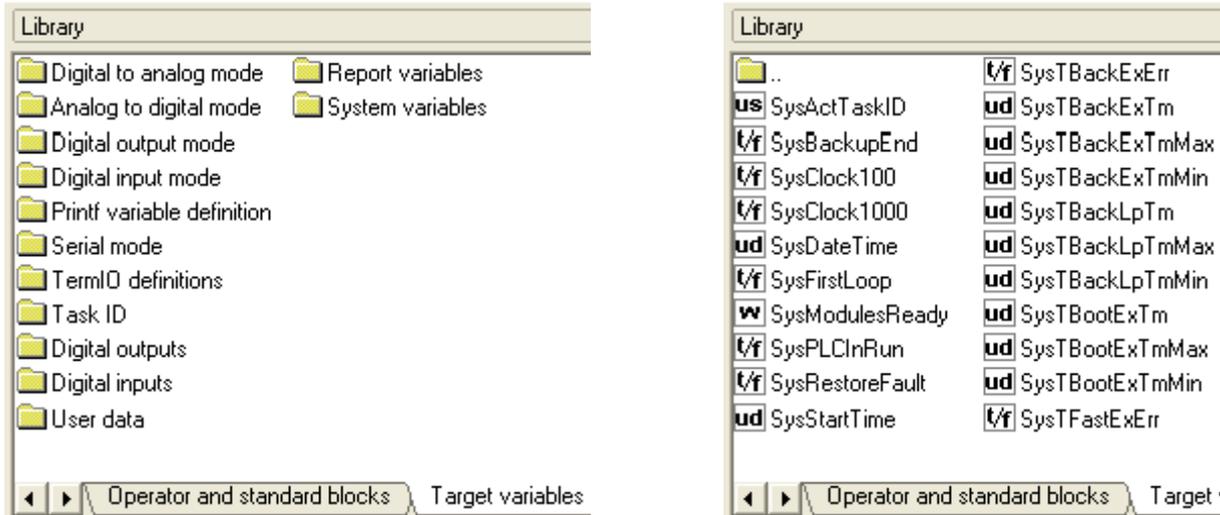
Questo tipo di dati è utilizzato dalle funzioni che gestiscono il controller CAN. La struttura definisce il formato di un messaggio CAN.

Name	Type	Description
RmReq	BOOL	FALSE:Data frame, TRUE:Remote request.
Length	USINT	Lunghezza record dati da 0 a 8 bytes.
MsgID	UDINT	Message ID, 11 o 29 bit di identificativo messaggio. Il bit 31 è il bit di FF.
Data	ARRAY[0..7] OF USINT	Array dati messaggio

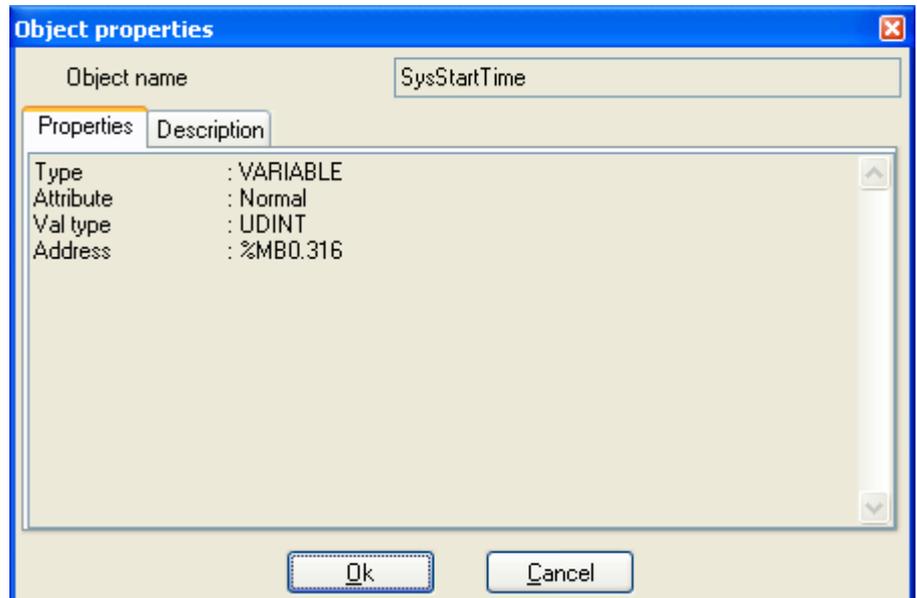
4 Variabili di sistema

Con l'ambiente di sviluppo LogicLab vengono pubblicate variabili di sistema che possono essere referenziate nel programma per accedere ad informazioni sul sistema. Le variabili sono visualizzate da LogicLab nella finestra delle librerie.

Se la finestra non è visualizzata, occorre abilitarne la visualizzazione dalla voce di menù **View** → **Tool windows** → **Library**. Attivando il Tab **Target variables** verrà visualizzato un elenco con tutte le variabili pubblicate suddivise in cartelle. Un doppio click del mouse sulla cartella **System variables** si aprirà la cartella visualizzando tutte le variabili pubblicate (Vedi foto a destra).



Agendo con il tasto destro del mouse su ogni singola variabile è possibile visualizzarne la finestra delle proprietà in cui è indicato il tipo e l'indirizzo di allocazione, così come nella figura a lato.



4.1 Variabili sola lettura (System variables)

Queste variabili di sola lettura presenti nella cartella **System variables**, ritornano informazioni sul sistema. Il programma utente può utilizzare le variabili a piacere ma non può modificarne il valore.

Name	Type	Description
SysClock100	BOOL	Clock lampeggiante con duty cycle di 100mS.
SysClock1000	BOOL	Clock lampeggiante con duty cycle di 100mS.
SysBackupEnd	BOOL	Attiva per un loop di programma al termine di un ciclo di backup dati.
SysRestoreFault	BOOL	Si attiva alla accensione se i dati di backup sono in errore. Tutti i dati sono azzerati.
SysPLCInRun	BOOL	Sempre attiva.
SysFirstLoop	BOOL	Attiva per un loop alla prima esecuzione di ogni task PLC.
SysLLabCn	BOOL	Attiva se l'ambiente di sviluppo LogicLab è connesso al sistema.
SysActTaskID	USINT	Numero di identificazione della task in corso, vedi tipi definiti .
SysModulesReady	UINT	Ogni bit della variabile se attivo indica la presenza del modulo connesso al bus SlimLine.
SysTBackLpTm	UDINT	Tempo di loop attuale della task di background PLC (uS).
SysTBackLpTmMin	UDINT	Tempo di loop minimo della task di background PLC (uS). E' possibile inicializzarne il valore settando il bit SysTimeInit .
SysTBackLpTmMax	UDINT	Tempo di loop massimo della task di background PLC (uS). E' possibile inicializzarne il valore settando il bit SysTimeInit .
SysTBootExTm	UDINT	Tempo di esecuzione attuale della task di boot PLC (uS).
SysTBootExTmMin	UDINT	Tempo di esecuzione minimo della task di boot PLC (uS). E' possibile inicializzarne il valore settando il bit SysTimeInit .
SysTBootExTm	UDINT	Tempo di esecuzione massimo della task di boot PLC (uS). E' possibile inicializzarne il valore settando il bit SysTimeInit .
SysTFastExTm	UDINT	Tempo di esecuzione attuale della task fast PLC (uS).
SysTFastExTmMin	UDINT	Tempo di esecuzione minimo della task fast PLC (uS). E' possibile inicializzarne il valore settando il bit SysTimeInit .
SysTFastExTmMax	UDINT	Tempo di esecuzione massimo della task fast PLC (uS). E' possibile inicializzarne il valore settando il bit SysTimeInit .
SysTSlowExTm	UDINT	Tempo di esecuzione attuale della task slow (uS).
SysTSlowExTmMin	UDINT	Tempo di esecuzione minimo della task slow (uS). E' possibile inicializzarne il valore settando il bit SysTimeInit .
SysTSlowExTmMax	UDINT	Tempo di esecuzione massimo della task slow (uS). E' possibile inicializzarne il valore settando il bit SysTimeInit .
SysTBackExTm	UDINT	Tempo di esecuzione attuale della task background (uS).
SysTBackExTmMin	UDINT	Tempo di esecuzione minimo della task background (uS). E' possibile inicializzarne il valore settando il bit SysTimeInit .
SysTBackExTmMax	UDINT	Tempo di esecuzione massimo della task background (uS). E' possibile inicializzarne il valore settando il bit SysTimeInit .
SysStartTime	UDINT	Data ed ora di start esecuzione programma PLC (Epoch time).
SysTime	UDINT	Tempo di sistema si incrementa ad ogni 1 mS raggiunto il valore massimo si reinializza.
SysTFastLpTm	UDINT	Tempo di loop della task fast PLC. Il tempo è impostabile tramite la funzione SysSetTaskLpTime .
SysTSlowLpTm	UDINT	Tempo di loop della task slow PLC. Il tempo è impostabile tramite la funzione SysSetTaskLpTime .
SysAppID	UDINT	ID applicazione, è un numero univoco che identifica il programma utente attualmente in esecuzione sul sistema.
SysMfcCode	UDINT	Codice manufacturer, questo codice v� richiesto con il prodotto. Se non definito � ritornato il codice 0.
SysCustomerCode	UDINT	Codice customer, questo codice pu� essere impostato dall'utente che ha accesso come amministratore al sistema. Se non definito � ritornato il codice 0.

4.2 Variabili sola lettura (Report variables)

Queste variabili di sola lettura presenti nella cartella **Report variables**, ritornano informazioni di report sul funzionamento del sistema. Il programma utente può utilizzare le variabili a piacere ma non può modificarne il valore.

Name	Type	Description
ErCode	UDINT	Numero dell'errore di esecuzione programma.
SysErTime	UDINT	Data ed ora in cui l'errore è avvenuto (Epoch time).
ErActTaskID	USINT	Numero di identificazione della task in cui l'errore si è verificato.
ErInfos	STRING[32]	Stringa ascii con informazioni aggiuntive sull'errore.

4.3 Variabili lettura e scrittura (System variables)

Queste variabili lettura e scrittura presenti nella cartella **System variables**, permettono di leggere e scrivere da programma utente informazioni sul funzionamento del sistema. Il programma utente può utilizzare le variabili a piacere ed anche modificarne il valore.

Name	Type	Description
SysTimeInit	BOOL	Attivata da programma utente o da debug permette di inizializzare il calcolo dei tempi di loop e di esecuzione delle task PLC. La variabile viene automaticamente resettata dal sistema.
SysDateTime	UDINT	Data ed ora di sistema (Epoch time). Modificando il valore verrà automaticamente aggiornato anche il real time clock.
SysLastError	UDINT	Last error, ritorna il valore dell'ultimo errore verificatosi nella esecuzione di una funzione o di un blocco funzione, elenco errori .

5 Definizioni dati

Oltre alle cartelle delle variabili di sistema sono presenti anche cartelle con identificatori di tipo dati che permettono di identificare in modo unifico un dato di sistema.

5.1 Variable types, definizione tipo variabili

La definizione del tipo di variabili è utilizzata nella funzione di stampa formattata [SysVarfprintf](#). Per ogni tipo di variabile è definito un valore USINT che lo identifica, il valore è indicato anche con definizioni che si possono trovare nella cartella **Variable types definition**.

Value	Define	Type	Description
10	BOOL_TYPE	USINT	Variabile booleana (BOOL), 1 bit può assumere solo significato FALSE o TRUE.
20	BYTE_TYPE	USINT	Variabile byte (BYTE) 8 bits senza segno, range da 0x00 a 0xFF.
21	SINT_TYPE	USINT	Variabile byte (SINT) 8 bits con segno, range da 0x00 a 0xFF.
22	USINT_TYPE	USINT	Variabile byte (USINT) 8 bits senza segno, range da 0x00 a 0xFF.
30	WORD_TYPE	USINT	Variabile word (WORD) 16 bits senza segno, range da 0x0000 a 0xFFFF.
31	INT_TYPE	USINT	Variabile word (INT) 16 bits con segno, range da 0x0000 a 0xFFFF.
32	UINT_TYPE	USINT	Variabile word (UINT) 16 bits senza segno, range da 0x0000 a 0xFFFF.
40	DWORD_TYPE	USINT	Variabile double word (DWORD) 32 bits senza segno, range da 0x00000000 a 0xFFFFFFFF.
41	DINT_TYPE	USINT	Variabile double word (DINT) 32 bits con segno, range da 0x00000000 a 0xFFFFFFFF.
42	UDINT_TYPE	USINT	Variabile double word (UDINT) 32 bits senza segno, range da 0x00000000 a 0xFFFFFFFF.
43	REAL_TYPE	USINT	Variabile floating (REAL) 32 bits con segno, range da 0x00000000 a 0xFFFFFFFF.
50	STRING_TYPE	USINT	Variabile stringa (STRING).

5.2 Task ID definition, identificatore di task PLC

Le task PLC sono identificate da un identificativo di tipo **USINT**, il valore della task corrente in esecuzione può essere letto dal programma utente dalla variabile [SysActTaskID](#). L'ID di task è indicato anche con definizioni che si possono trovare nella cartella **Task ID**.

Value	Define	Type	Description
0	ID_TASK_BOOT	USINT	Identifica la task di boot PLC. Questa task viene eseguita solo al primo loop di esecuzione programma utente.
1	ID_TASK_BACK	USINT	Identifica la task di background. Questa task è eseguita in background alle task slow e fast. Il tempo di loop di questa task non è fisso ma dipende dal carico di lavoro della CPU nella esecuzione delle altre tasks.
2	ID_TASK_SLOW	USINT	Identifica la task slow. Questa task è eseguita con un tempo di loop fisso definito con la funzione SysSetTaskLpTime . Di default il tempo è fissato a 10 mS.
3	ID_TASK_FAST	USINT	Identifica la task fast. Questa task è eseguita con un tempo di loop fisso definito con la funzione SysSetTaskLpTime . Di default il tempo è fissato a 1 mS.

5.3 TermIO, definizioni per terminale di I/O

Nella gestione dei terminali di I/O sono utilizzate delle definizioni che si possono trovare nella cartella **TermIO definition**.

Value	Define	Type	Description
0	NULL	FILEP	Identifica un puntatore vuoto. Utilizzato come ritorno da alcune funzioni in caso di errore.
-1	EOF	INT	Identifica il fine file. Utilizzato come valore di ritorno da alcune funzioni in caso di errore.

5.4 Serial mode definition, definizioni modo seriale

Nella gestione dei terminali di I/O sono utilizzate delle definizioni che si possono trovare nella cartella **Serial mode definition**.

Value	Define	Type	Description
0	DTR_OFF	USINT	Valore membro DTRManagement del dato SYSSERIALMODE , indica segnale DTR sempre in condizione di off.
1	DTR_ON	USINT	Valore membro DTRManagement del dato SYSSERIALMODE , indica segnale DTR sempre in condizione di on.
2	DTR_AUTO_WO_TIMES	USINT	Valore membro DTRManagement del dato SYSSERIALMODE , indica segnale DTR in funzionamento automatico senza interposizione di tempi.
3	DTR_AUTO_W_TIMES	USINT	Valore membro DTRManagement del dato SYSSERIALMODE , indica segnale DTR in funzionamento automatico con interposizione di tempi.

6 Protocollo modbus

Il modbus è un protocollo di comunicazione seriale diventato uno standard de facto nella comunicazione di tipo industriale, ed è ora il protocollo di connessione più diffuso fra i dispositivi elettronici industriali. E' un protocollo di tipo richiesta/risposta ed offre dei servizi specificati da function codes.

SlimLine supporta il protocollo modbus Rtu sulle porte seriali, e modbus Over IP su connessione ethernet su porta **502**. Il protocollo modbus Rtu sulla porta seriale ha come parametri di comunicazione di default **115200, 8, e**, e l'indirizzo di nodo sia su porta seriale che su TCP/IP è **1**.

6.1 Accesso variabili da modbus

Le funzioni del protocollo accedono tutte alla memoria utente **MX100**, le funzioni supportate sono:

Code	Function	Tipo oggetto	Tipo accesso	Range indirizzo
01h	Read coil status	Bit singolo	Read	40000-44095 (<i>Note 1</i>)
02h	Read input status	Bit singolo	Read	40000-44095 (<i>Note 1</i>)
03h	Read holding registers	Word (16 Bit)	Read	40000-42047 (<i>Note 2</i>)
04h	Read input registers	Word (16 Bit)	Read	40000-42047 (<i>Note 2</i>)
05h	Force single coil	Bit singolo	Write	40000-44095 (<i>Note 1</i>)
06h	Preset single register	Word (16 Bit)	Write	40000-42047 (<i>Note 2</i>)
10h	Preset multiple registers	Word (16 Bit)	Write	40000-42047 (<i>Note 2</i>)

Note 1) Nelle funzioni che accedono al bit singolo (In realtà ogni bit equivale ad un byte di memoria) si utilizza nel comando l'indirizzo della variabile, quindi dovendo accedere alla locazione **MX100.50** utilizzeremo come indirizzo il valore **40050**.

Note 2) Nelle funzioni che accedono ai registri (16 Bits) occorre considerare l'indirizzo della variabile diviso per 2, quindi dovendo raggiungere da modbus la locazione **MX100.50** utilizzeremo come indirizzo il valore **40025**.

6.1.1 Lettura variabili da modbus

Per la lettura delle variabili si utilizza il comando **Read holding registers** (Codice **0x03**). Ipotizzando di dover accedere in lettura ad una variabile **DWORD** allocata in memoria all'indirizzo **MX100.64** calcoleremo l'indirizzo di lettura nel modo:

$((\text{Indirizzo variabile}/2)+\text{Offset})-1 \rightarrow ((64/2)+40000)-1=40031 \rightarrow 0x9C5F$

Essendo una variabile **DWORD** dovremo leggere 2 registri consecutivi a partire dal suo indirizzo di allocazione, ipotizzando che il valore della variabile sia **0x12345678** avremo.

Frames modbus RTU

Stringa di comando: **01 03 9C 5F 00 02 DA 49**

Stringa di risposta: **01 03 04 56 78 12 34 66 D5**

Frames modbus TCP/IP

Stringa di comando: **00 00 00 00 00 06 01 03 9C 5F 00 02**

Stringa di risposta: **00 00 00 00 00 07 01 03 04 56 78 12 34**

La rappresentazione dei dati in SlimLine è nel formato **Little-Endian**, la numerazione inizia dal byte meno significativo per finire col più significativo, quindi come si nota dalla stringa di risposta il valore della variabile a 32 bits **0x12345678** viene ritornato suddiviso in due registri a 16 bits con i valori **0x5678**, **0x1234**.

6.1.2 Scrittura variabili da modbus

Per la scrittura delle variabili si utilizza il comando **Preset multiple registers** (Codice **0x10**). Ipotizzando di dover accedere in scrittura ad una variabile **DWORD** allocata in memoria all'indirizzo **MX100.64** calcoleremo l'indirizzo di scrittura nel modo:

$((\text{Indirizzo variabile}/2)+\text{Offset})-1 \rightarrow ((64/2)+40000)-1=40031 \rightarrow 0x9C5F$

Essendo una variabile **DWORD** dovremo scrivere 2 registri consecutivi a partire dal suo indirizzo di allocazione, ipotizzando di dover scrivere nella variabile il valore **0x12345678** avremo.

Frames modbus RTU

Stringa di comando: **01 10 9C 5F 00 02 04 56 78 12 34 D3 33**

Stringa di risposta: **01 10 9C 5F 00 02 5F 8A**

Frames modbus TCP/IP

Stringa di comando: **00 00 00 00 00 0B 01 10 9C 5F 00 02 04 56 78 12 34**

Stringa di risposta: **00 00 00 00 00 06 01 10 9C 5F 00 02**

La rappresentazione dei dati in SlimLine è nel formato **Little-Endian**, la numerazione inizia dal byte meno significativo per finire col più significativo, quindi come si nota dalla stringa di comando il valore da scrivere a 32 bits **0x12345678** viene definito suddiviso in due registri a 16 bits con i valori **0x5678**, **0x1234**.

6.2 Accesso Real time clock da modbus

E' possibile accedere al real time clock utilizzando i comandi modbus di accesso ai registri le funzioni supportate sono:

Code	Function	Tipo oggetto	Tipo accesso	Range indirizzo
03h	Read holding registers	Word (16 Bit)	Read	100-105 (150 Epoch time)
04h	Read input registers	Word (16 Bit)	Read	100-105 (150 Epoch time)
06h	Preset single register	Word (16 Bit)	Write	100-105 (150 Epoch time)
10h	Preset multiple registers	Word (16 Bit)	Write	100-105 (150 Epoch time)

I registri (16 Bits) del real time clock sono allocati in locazioni consecutive a partire dall'indirizzo modbus 100. I registri contengono il valore attuale del real time clock e scrivendo un nuovo valore il real time clock verrà automaticamente aggiornato.

Address	Register	Note
100	Second	Valore secondi (Range da 0 a 59)
101	Minute	Valore minuti (Range da 0 a 59)
102	Hour	Valore ora (Range da 0 a 23)
103	Day	Valore giorno (Range da 1 a 31)
104	Month	Valore mese (Range da 1 a 12)
105	Year	Valore anno (Range da 1900 a 2037)

6.2.1 Lettura RTC da modbus

Per la lettura dei registri del real time clock si utilizza il comando **Read holding registers** (Codice **0x03**). Dovremo leggere 6 registri consecutivi a partire dall'indirizzo di allocazione, l'indirizzamento di modbus prevede un offset di 1, quindi **99 (0x0063)**.

Frames modbus RTU

Stringa di comando: **01 03 00 63 00 06 35 D6**

Stringa di risposta: **01 03 0C 00 1E 00 30 00 0B 00 1D 00 09 07 DA A2 32**

Frames modbus TCP/IP

Stringa di comando: **00 00 00 00 00 06 01 03 00 63 00 06**

Stringa di risposta: **00 00 00 00 00 0F 01 03 0C 00 1E 00 30 00 0B 00 1D 00 09 07 DA**

Come si vede dalla risposta il valore è:

Secondi: 30 (**0x001E**)

Minuti: 48 (**0x0030**)

Ora: 11 (**0x000B**)

Giorno: 29 (**0x001D**)

Mese: 9 (**0x0009**)

Anno: 2010 (**0x07DA**)

6.2.2 Scrittura RTC da modbus

Per la scrittura dei registri del real time clock si utilizza il comando **Preset multiple registers** (Codice **0x10**). Dovremo scrivere 6 registri consecutivi a partire dall'indirizzo di allocazione, l'indirizzamento di modbus prevede un offset di 1, quindi **99 (0x0063)**. Ipotizziamo di dover impostare nel real time clock i valori:

Secondi: 30 (**0x001E**)
 Minuti: 48 (**0x0030**)
 Ora: 11 (**0x000B**)
 Giorno: 29 (**0x001D**)
 Mese: 9 (**0x0009**)
 Anno: 2010 (**0x07DA**)

Frames modbus RTU

Stringa di comando: **01 10 00 63 00 06 08 00 1E 00 30 00 0B 00 1D 00 09 07 DA 5D C8**
 Stringa di risposta: **01 10 00 63 00 06 B0 15**

Frames modbus TCP/IP

Stringa di comando: **00 00 00 00 00 13 01 10 00 63 00 06 08 00 1E 00 30 00 0B 00 1D 00 09 07 DA**
 Stringa di risposta: **00 00 00 00 00 06 01 10 00 63 00 06**

6.3 Accesso Epoch time da modbus

E' allocato anche un registro a 32 bits per il valore di data/ora in Epoch time, l'accesso a questo registro in lettura e/o scrittura va sempre effettuato usando due registri a 16 bits.

Address	Register	Note
150	Epoch time	Epoch time

6.3.1 Lettura Epoch time da modbus

Per la lettura dell'epoch time si utilizza il comando **Read holding registers** (Codice **0x03**). Dovremo leggere 2 registri consecutivi a partire dall'indirizzo di allocazione, l'indirizzamento di modbus prevede un offset di 1, quindi **149 (0x0095)**.

Frames modbus RTU

Stringa di comando: **01 03 00 95 00 02 D4 27**
 Stringa di risposta: **01 03 04 30 B5 4C A3 90 6C**

Frames modbus TCP/IP

Stringa di comando: **00 00 00 00 00 06 01 03 00 95 00 02**
 Stringa di risposta: **00 00 00 00 00 07 01 03 04 30 B5 4C A3**

Come si vede dalla risposta il valore è: **0x4CA330B5** → **1285763253** → **GMT: Wed, 29 Sep 2010 12:27:33 UTC**.

6.3.2 Scrittura Epoch time da modbus

Per la scrittura dell'epoch time si utilizza il comando **Preset multiple registers** (Codice **0x10**). Dovremo scrivere 2 registri consecutivi a partire dall'indirizzo di allocazione, l'indirizzamento di modbus prevede un offset di 1, quindi **149 (0x0095)**. Ipotizziamo di dover impostare il valore:

GMT: Wed, 29 Sep 2010 12:27:33 UTC → **1285763253** → **0x4CA330B5**

Frames modbus RTU

Stringa di comando: **01 10 00 95 00 02 04 30 B5 4C A3 50 A3**
 Stringa di risposta: **01 10 00 95 00 02 51 E4**

Frames modbus TCP/IP

Stringa di comando: **00 00 00 00 00 0B 01 10 00 95 00 02 04 30 B5 4C A3**
 Stringa di risposta: **00 00 00 00 00 06 01 10 00 95 00 02**

7 Funzioni ed FB

Funzioni

Le funzioni hanno numero di variabili in ingresso variabile e sempre una sola variabile in uscita. Per utilizzarle basta inserirle nei programmi LD ed FBD e connetterle alle variabili. Nei programmi IL devono essere chiamate con l'istruzione CAL, nei programmi ST basta indicarne il nome per essere eseguite.

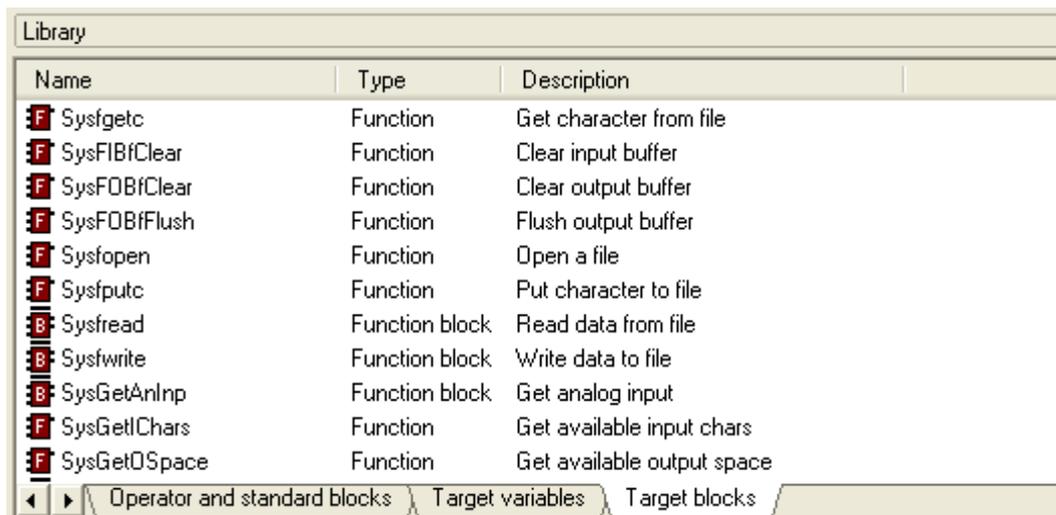
Function Blocks

Le FB a differenza delle funzioni allocano nel programma una variabile che contiene tutte le variabili di input e di output gestite dal blocco funzione. Per utilizzarle basta inserirle nei programmi LD ed FBD e connetterle alle variabili. Nei programmi IL devono essere chiamate con l'istruzione CAL, nei programmi ST basta indicarne il nome per essere eseguite.

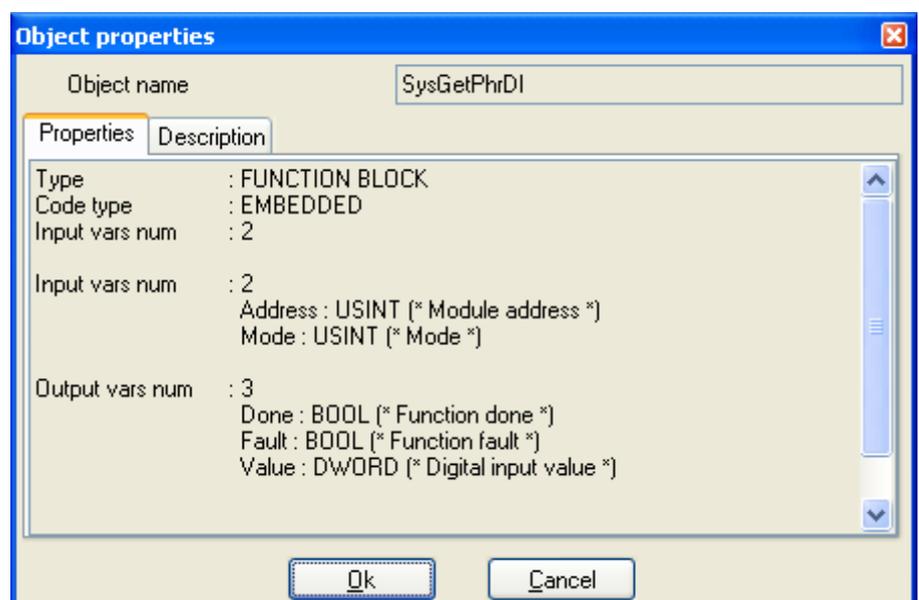
7.1 Funzioni ed FB embedded

Con l'ambiente di sviluppo LogicLab vengono fornite funzioni e blocchi funzione (FB) embedded che permettono di accedere alle risorse hardware e software del sistema **Slim line**. Le funzioni e le FB embedded sono visualizzate da LogicLab nella finestra delle librerie.

Se la finestra non è visualizzata, occorre abilitarne la visualizzazione dalla voce di menù **View** → **Tool windows** → **Library**. Attivando il Tab **Target blocks** verrà visualizzato un elenco con tutte le funzioni (Indicate con **F**) ed i blocchi funzione (Indicati con **B**) embedded.



Agendo con il tasto destro del mouse su ogni singola funzione o blocco funzione è possibile visualizzarne la finestra delle proprietà in cui sono indicate le variabili in ingresso ed il ritorno delle funzioni, mentre per i blocchi funzioni sono indicate le variabili in ingresso ed in uscita, così come nella figura a lato.



7.2 Librerie

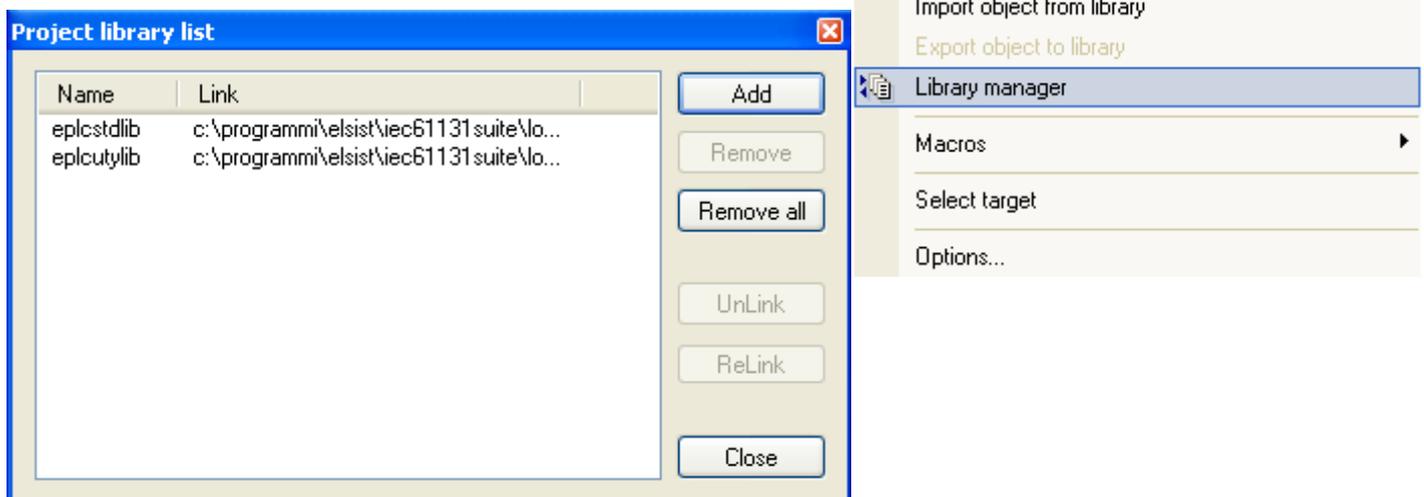
Le aggiunte alle funzioni ed FB embedded sono fornite dalle librerie che contengono una serie di funzioni e blocchi funzioni che possono essere utilizzati nel proprio programma. Le librerie fornite con LogicLab si trovano nella directory di installazione programma **Programmi\Elsist\IEC61131Suite\LogicLab2p0\Libraries**, ma è possibile anche utilizzare librerie fornite successivamente o di cui si è eseguito il download dal sito. Esistono due possibilità per utilizzare le librerie.

Import libreria: In questo modo vengono importati nel proprio programma tutti gli oggetti presenti nella libreria, gli oggetti possono così essere utilizzati nel programma. Questa soluzione la controindicazione quella di aumentare la dimensione del file di progetto LogicLab (*.ppjs), in quanto deve contenere oltre al proprio programma anche tutti gli oggetti della libreria importata. **Il programma eseguibile generato conterrà comunque solo gli oggetti utilizzati.**

Import oggetti: In questo modo è possibile importare da una libreria solo gli oggetti (Funzioni, FB, ecc) che interessano, i quali diverranno parte integrante del proprio progetto.

7.2.1 Import libreria

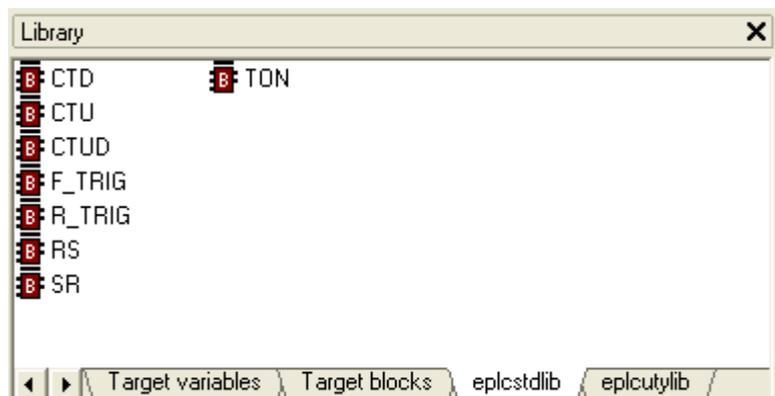
Con questa modalità vengono importati tutti gli oggetti presenti nella libreria. Per importare nel proprio programma l'intera libreria dal menù selezionare la voce **Project** → **Library manager** si aprirà una finestra come quella sottoriportata.



Agendo sul tasto **Add** si aprirà una finestra di browser del disco. Scegliere la directory dove si trova la libreria, e selezionare i files di libreria da importare.

Agendo sul tasto **Close**, nella finestra **Library** di LogicLab (**Ctrl-L**) verranno visualizzati dei tabs aggiuntivi, uno per ogni libreria importata.

Basta trascinare l'oggetto desiderato nel proprio progetto per poterlo utilizzare.



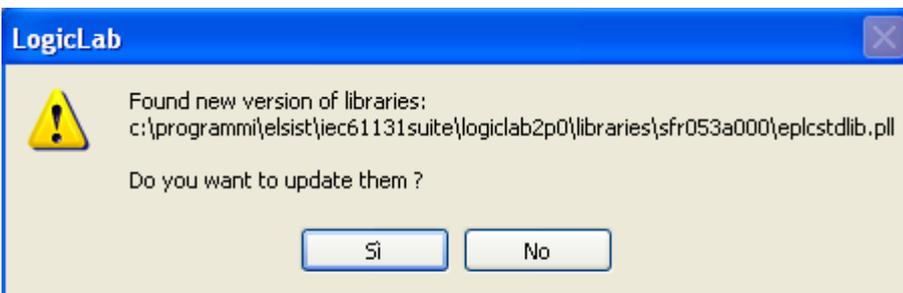
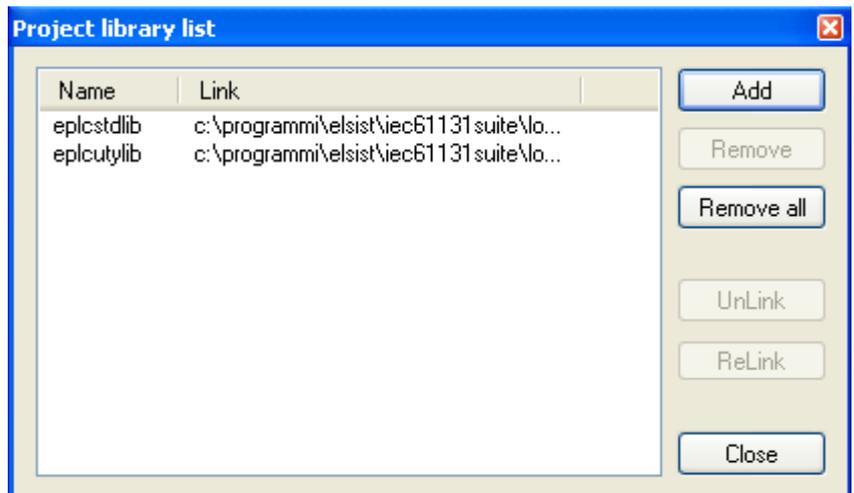
7.2.2 Link a libreria

Eseguendo l'import delle librerie nel proprio progetto come indicato al capitolo precedente tutti gli oggetti della libreria importata vengono trasferiti nel proprio file di progetto (*.ppjs), ma viene comunque mantenuto un link alla libreria di origine come si vede dalla finestra a lato.

Questo permette nel caso la libreria sorgente venga modificata con una versione più recente di effettuare l'update automatico della nuova libreria nel proprio progetto.

Se la libreria sorgente non è più presente oppure è stata spostata dalla posizione da dove è stata importata, LogicLab non eseguirà più il controllo senza segnalare errori.

Tramite il menù **Project** → **Library manager** che apre la finestra a lato, come si vede, è possibile selezionare le varie librerie e con il tasto **UnLink** rimuovere il link oppure con il tasto **ReLink** eseguire un link alla nuova posizione dove si trova la libreria.



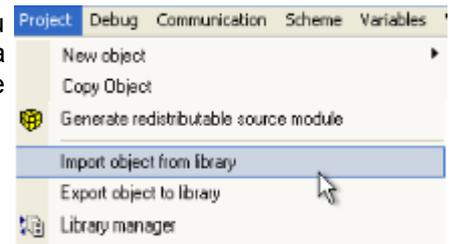
Aperto il progetto, LogicLab controlla tutte le librerie importate e nel caso una o più librerie sorgenti siano più recenti delle versioni importate viene visualizzato un messaggio di avvertimento che chiede conferma se eseguire oppure no l'update delle librerie.

Eseguendo l'update tutti gli oggetti della libreria importata presenti nel proprio

progetto vengono sovrascritti con gli oggetti presenti nella libreria sorgente ed eventuali nuovi oggetti sono automaticamente importati.

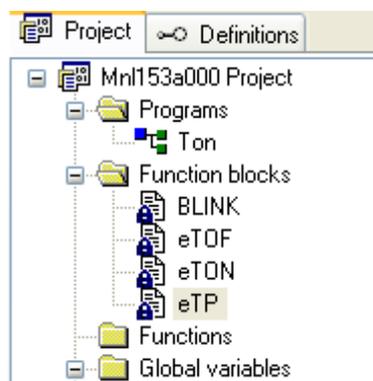
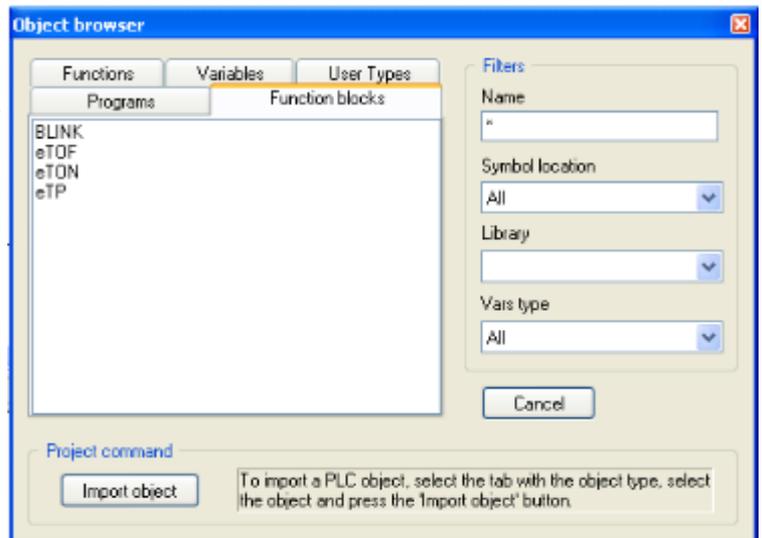
7.2.3 Import oggetti

Per importare nel proprio programma oggetti dalle librerie occorre nel menù selezionare la voce **Project** → **Import object from library**. Si aprirà una finestra di browser del disco. Scegliere la directory dove si trova la libreria, e selezionare il file di libreria da cui si desidera importare gli oggetti.



Si aprirà la finestra di **Object browser** che permetterà di visualizzare tutti gli oggetti presenti nella libreria, selezionando i vari tabulatori presenti è possibile visualizzare tutti gli oggetti della libreria ordinati per nome.

Con un click del mouse si evidenzia l'oggetto o gli oggetti desiderati. Con il tasto **Import object** gli oggetti selezionati verranno inclusi nel programma.



Come si nota dalla foto a destra alcuni oggetti appaiono visualizzati con un simbolo di lucchetto, questo sta ad indicare che sono oggetti protetti, cioè non possono essere modificati. Una volta importati nel proprio programma gli oggetti resteranno inclusi nel programma stesso e sarà possibile utilizzarli su qualsiasi PC anche se non si dispone della libreria originale.

7.2.4 Considerazioni su link a libreria e su import oggetti

Come visto nei paragrafi precedenti per utilizzare funzioni e/o blocchi funzione di libreria è possibile usare due diversi metodi, importare solo l'oggetto desiderato oppure tutta la libreria nel proprio progetto.

In entrambi i casi l'oggetto verrà incluso nel proprio progetto, in questo modo si è sicuri che anche nel futuro con versioni successive di libreria sarà sempre possibile ricompilare il progetto utilizzando l'oggetto con il quale si era sviluppato e testato.

Nel caso si desideri sostituire l'oggetto con una versione più recente dello stesso si userà un diverso approccio in funzione del fatto che l'oggetto sia presente in una libreria linkata oppure sia stato importato.

Libreria linkata

Come visto precedentemente, le librerie linkate mantengono un riferimento alla libreria di origine, nel percorso di memorizzazione nella distribuzione di LogicLab le librerie sono incluse in directories il cui nome rappresenta la versione. In questo modo potranno essere distribuite versioni successive di libreria, ma il progetto alla sua riapertura farà sempre il controllo con la versione originale senza eseguire l'upgrade automatico.

Per effettuare l'upgrade di un oggetto di una libreria linkata occorre eseguire un **ReLink** alla nuova versione della libreria. **Attenzione!** Questa operazione aggiornerà tutti gli oggetti presenti nella libreria.

Oggetto importato

Nel caso di oggetto importato, per effettuare l'upgrade, basterà rimuovere l'oggetto attuale dal progetto ed eseguire un import dello stesso oggetto dalla nuova versione della libreria.

Conclusioni

In generale si consiglia di non eseguire il link della libreria ma di includere i singoli oggetti nel proprio progetto, questo permette una più semplice gestione degli upgrades.

Alcune librerie contengono una serie di oggetti (Funzioni e blocchi funzione) che sono di vasto impiego, in questo caso è consigliabile sempre linkare queste librerie. Ecco l'elenco delle librerie che si consiglia di linkare al progetto:

Libreria	Codice	Descrizione
ePLCStdLib	SFR053*000	Libreria standard IEC61131, contiene funzioni e blocchi funzione definiti dalla normativa IEC61131 e non presenti nella libreria embedded del prodotto.
ePLCAuxLib	SFR058*000	Libreria ausiliaria, contiene funzioni e blocchi funzione di varia utilità.

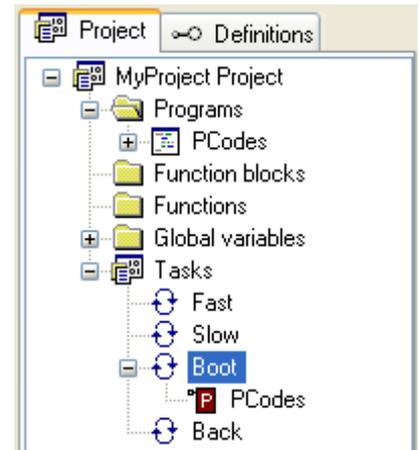
7.3 Protezione funzioni e blocchi funzione

Alcune funzioni e/o blocchi funzione di libreria possono essere protetti da un codice che deve essere ordinato separatamente. Per abilitarne l'esecuzione occorre sbloccarle definendone il codice (Stringa alfabetica di 18 caratteri) con la funzione **SysPCodeAccept**.

La funzione va eseguita una sola volta passando il codice di protezione, se il codice è corretto la funzione ritorna **TRUE** e la relativa funzione sarà sprotetta fino al prossimo riavvio del programma. E' possibile eseguire più chiamate alla funzione una per ogni codice di protezione da definire.

Il consiglio è di inserire le varie chiamate alla funzione in un programma che verrà eseguito nella task di boot quindi prima di ogni chiamata ad altri programmi, garantendo lo sblocco delle funzioni desiderate.

A lato si può vedere come in un progetto il programma di definizione codici di protezione **PCodes** sia definito nella esecuzione della task di boot. Di seguito riportiamo il codice sorgente del programma **PCodes** realizzato in linguaggio ST. Naturalmente i codici riportati sono di fantasia pertanto se eseguito la funzione **SysPCodeAccept** ritornerà sempre **FALSE**.



Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	CodesOk	BOOL	Auto	[0..2]	3(0)	..	Protection codes ok

Programma ST

```
(* Check the protection codes. *)

CodesOk[0]:=SysPCodeAccept('abcdefghijklmnopqr'); (* Protection code ok (Function 1) *)
CodesOk[1]:=SysPCodeAccept('rqponmlkjihgfedcba'); (* Protection code ok (Function 2) *)
CodesOk[2]:=SysPCodeAccept('abcdefghijklmnopqihgfedcba'); (* Protection code ok (Function 3) *)

(* [End of file] *)
```

Normalmente le funzioni ed i blocchi funzioni protetti possono funzionare in modo demo per un certo periodo di tempo dalla loro prima esecuzione dopo l'accensione del sistema. Terminato il tempo di prova termina il funzionamento e viene generato un errore che è rilevabile con la funzione **SysGetLastError**.

7.4 Funzioni ed FB per gestione Flip/Flop

7.4.1 F_TRIG, Falling edge trigger

Type	Library	Version
FB	ePLCStdLib	SFR053A000

Questo blocco funzione attiva l'uscita **Q** per un loop di programma sul fronte di disattivazione dell'ingresso di clock **CLK**.



CLK (BOOL) Clock, sul fronte di disattivazione del segnale, viene attivata l'uscita Q per un loop di programma.

Q (BOOL) Uscita, attiva per un loop di programma sul fronte di disattivazione dell'ingresso di clock CLK.

Esempi

Sul fronte di disattivazione dell'ingresso digitale **Di00M00** viene attivata per un loop di programma l'uscita digitale **Do00M00**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBF_TRIG	F_TRIG	Auto	No	0	..	F_TRIG (Falling edge trigger function block)

Esempio LD (Ptp115a100)



Esempio IL

```

CAL FBF_TRIG (* Call the F_TRIG function block *)

LD Di00M00
ST FBF_TRIG.CLK (* Transfer the digital input to the function block clock *)

LD FBF_TRIG.Q
ST Do00M00 (* On the falling edge of digital input the output is set *)
    
```

Esempio ST

```

FBF_TRIG(); (* Call the F_TRIG function block *)

FBF_TRIG.CLK:=Di00M00; (* Transfer the digital input to the function block clock *)
Do00M00:=FBF_TRIG.Q; (* On the falling edge of digital input the output is set *)
    
```

7.4.2 R_TRIG, Raising edge trigger

Type	Library	Version
FB	ePLCStdLib	SFR053A000

Questo blocco funzione attiva l'uscita **Q** per un loop di programma sul fronte di attivazione dell'ingresso di clock **CLK**.



CLK (BOOL) Clock, sul fronte di attivazione del segnale, viene attivata l'uscita Q per un loop di programma.

Q (BOOL) Uscita, attiva per un loop di programma sul fronte di attivazione dell'ingresso di clock CLK.

Esempi

Sul fronte di attivazione dell'ingresso digitale **Di00M00** viene attivata per un loop di programma l'uscita digitale **Do00M00**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBR_TRIG	R_TRIG	Auto	No	0	..	R_TRIG (Raising edge trigger function block)

Esempio LD (Ptp115a100)



Esempio IL

```

CAL FBR_TRIG (* Call the R_TRIG function block *)

LD Di00M00
ST FBR_TRIG.CLK (* Transfer the digital input to the function block clock *)

LD FBR_TRIG.Q
ST Do00M00 (* On the raising edge of digital input the output is set *)
    
```

Esempio ST

```

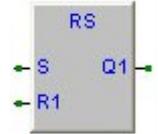
FBR_TRIG(); (* Call the R_TRIG function block *)

FBR_TRIG.CLK:=Di00M00; (* On the raising edge of digital input the counter count up *)
Do00M00:=FBR_TRIG.Q; (* On the raising edge of digital input the output is set *)
    
```

Type	Library	Version
FB	ePLCStdLib	SFR053A000

7.4.3 RS, Reset/Set flip flop

Questo blocco funzione su attivazione del comando di set **S** attiva l'uscita **Q1** che rimane attiva anche quando il comando viene disattivato. Per disattivare l'uscita occorre attivare il comando di reset **R1**.



Il comando di reset R1 è prioritario sul comando di set S.

S (BOOL) Set, su attivazione del segnale, viene attivata l'uscita Q1 che rimane attiva anche quando il comando viene disattivato.

R1 (BOOL) Reset, su attivazione del segnale, viene disattivata l'uscita Q1 è prioritario sul comando di set S.

Q1 (BOOL) Uscita, si attiva e disattiva in funzione dei comandi di S set e R1 reset.

Esempi

Sulla attivazione dell'ingresso digitale **Di00M00** viene attivata l'uscita digitale **Do00M00** che rimane attiva anche quando l'ingresso digitale **Di00M00** viene disattivato. Per disattivare l'uscita digitale **Do00M00** occorre attivare l'ingresso digitale **Di01M00**.

Nota! L'ingresso digitale **Di01M00** ha la priorità sull'ingresso digitale **Di00M00**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBRs	RS	Auto	No	0	..	RS (Reset/Set function block)

Esempio LD (Ptp115a100)



Esempio IL

```

CAL FBRs (* Call the RSG function block *)

LD Di00M00
ST FBRs.S (* Transfer the digital input to the set command *)

LD Di01M00
ST FBRs.R1 (* Transfer the digital input to the reset command *)

LD FBRs.Q1
ST Do00M00 (* The function block output is copied to digital output *)
    
```

Esempio ST

```

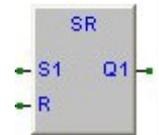
FBRs(); (* Call the RS function block *)

FBRs.S:=Di00M00; (* Transfer the digital input to the set command *)
FBRs.R1:=Di01M00; (* Transfer the digital input to the reset command *)
Do00M00:=FBRs.Q1; (* The function block output is copied to digital output *)
    
```

Type	Library	Version
FB	ePLCStdLib	SFR053A000

7.4.4 SR, Set/Reset flip flop

Questo blocco funzione su attivazione del comando di set **S1** attiva l'uscita **Q1** che rimane attiva anche quando il comando di set viene disattivato. Per disattivare l'uscita occorre attivare il comando di reset **R**.



Il comando di set S1 è prioritario sul comando di reset R.

S1 (BOOL) Set, su attivazione del segnale, viene attivata l'uscita Q1 che rimane attiva anche quando il comando viene disattivato. Il comando è prioritario sul comando di reset R.

R (BOOL) Reset, su attivazione del segnale, viene disattivata l'uscita Q1.

Q1 (BOOL) Uscita, si attiva e disattiva in funzione dei comandi di S set e R reset.

Esempi

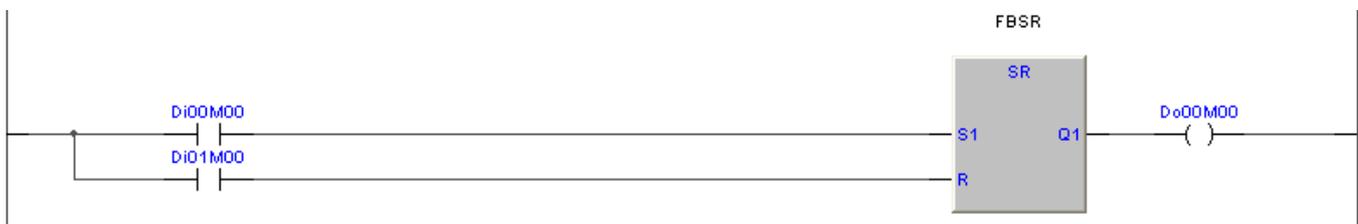
Sulla attivazione dell'ingresso digitale **Di00M00** viene attivata l'uscita digitale **Do00M00** che rimane attiva anche quando l'ingresso digitale **Di00M00** viene disattivato. Per disattivare l'uscita digitale **Do00M00** occorre attivare l'ingresso digitale **Di01M00**.

Nota! L'ingresso digitale **Di00M00** ha la priorità sull'ingresso digitale **Di01M00**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBSR	SR	Auto	No	0	..	SR (Set/Reset function block)

Esempio LD (Ptp115a100)



Esempio IL

```

CAL FBSR (* Call the SR function block *)

LD Di00M00
ST FBSR.S1 (* Transfer the digital input to the set command *)

LD Di01M00
ST FBSR.R (* Transfer the digital input to the reset command *)

LD FBSR.Q1
ST Do00M00 (* The function block output is copied to digital output *)
    
```

Esempio ST

```

FBSR(); (* Call the SR function block *)

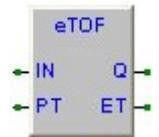
FBSR.S1:=Di00M00; (* Transfer the digital input to the set command *)
FBSR.R:=Di01M00; (* Transfer the digital input to the reset command *)
Do00M00:=FBSR.Q1; (* The function block output is copied to digital output *)
    
```

7.5 Funzioni ed FB per gestione timers

7.5.1 eTOF, Timer Off

Type	Library	Version
FB	ePLCStdLib	SFR053A000

Questo blocco funzione esegue la temporizzazione sulla disattivazione, attivando l'ingresso **IN** l'uscita **Q** si attiva immediatamente ed il tempo in uscita **ET** si azzerava. Disattivando l'ingresso **IN** inizia il conteggio e dopo il tempo definito **PT** espresso in mS, si disattiva l'uscita **Q**. Sulla uscita **ET** viene ritornato il tempo trascorso dalla disattivazione dell'ingresso espresso in mS.



- IN** (BOOL) Ingresso timer, attivandolo l'uscita **Q** si attiva immediatamente ed il tempo in uscita **ET** si azzerava. Disattivandolo inizia il conteggio e dopo il tempo definito in **PT**, si disattiva l'uscita **Q**.
- PT** (UDINT) Preset tempo, definisce il tempo di ritardo dalla disattivazione dell'ingresso **IN** alla disattivazione dell'uscita **Q**, espresso in mS.
- Q** (BOOL) Uscita timer, si attiva su attivazione ingresso **IN**, e si disattiva dopo il tempo definito in **PT** dalla disattivazione dell'ingresso **IN**.
- ET** (UDINT) Tempo timer, si azzerava su attivazione ingresso **IN** ed inizia conteggio da disattivazione ingresso **IN**. Raggiunto tempo impostato in **PT** si arresta conteggio, espresso in mS.

Esempi

Il timer è pre-settato a 1 secondo (1000 mS). Attivando l'ingresso digitale **Di00M00** si attiva immediatamente l'uscita digitale **Do00M00** ed il valore di tempo nella variabile **VarOut** è azzerato.

Disattivando l'ingresso digitale **Di00M00** il timer inizia il conteggio del tempo, il valore di tempo trascorso dalla disattivazione è trasferito nella variabile **VarOut**. Trascorso il tempo l'uscita digitale **Do00M00** si disattiva.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBeTOF	eTOF	Auto	No	0	..	eTOF (Timer Off function block)
2	OutValue	UDINT	Auto	No	0	..	Output value

Esempio LD (Ptp115a100)



Esempio IL

```

CAL FBeTOF (* Call the eTOF function block *)

LD Di00M00
ST FBeTOF.IN (* Transfer the digital input to timer input *)

LD 1000
ST FBeTOF.PT (* Set the delay time *)

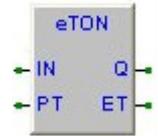
LD FBeTOF.Q
ST Do00M00 (* When time is passed the digital output is set *)

LD FBeTOF.ET
ST OutValue (* The counting time is copied to variable *)
    
```

7.5.2 eTON, Timer On

Type	Library	Version
FB	ePLCStdLib	SFR053A000

Questo blocco funzione esegue la temporizzazione sulla attivazione, attivando l'ingresso **IN** inizia il conteggio e dopo il tempo definito **PT** espresso in mS, si attiva l'uscita **Q**. Sulla uscita **ET** viene ritornato il tempo trascorso dalla attivazione dell'ingresso espresso in mS. Disattivando l'ingresso **IN** l'uscita **Q** si disattiva istantaneamente ed il valore di tempo su uscita **ET** si azzerava.



- IN** (BOOL) Ingresso timer, attivandolo inizia il conteggio e dopo il tempo definito in PT, si attiva l'uscita Q. Disattivandolo l'uscita Q si disattiva immediatamente ed il tempo in uscita ET si azzerava.
- PT** (UDINT) Preset tempo, definisce il tempo di ritardo dalla attivazione dell'ingresso IN alla attivazione dell'uscita Q, espresso in mS.
- Q** (BOOL) Uscita timer, si attiva dopo il tempo definito in PT dalla attivazione dell'ingresso IN e si disattiva su disattivazione ingresso IN.
- ET** (UDINT) Tempo timer, inizia conteggio da attivazione ingresso IN, raggiunto tempo impostato in PT si arresta conteggio. Si azzerava su disattivazione ingresso IN, espresso in mS.

Esempi

Su attivazione dell'ingresso digitale **Di00M00** dopo 1 Sec (1000 mS) viene attivata l'uscita digitale **Do00M00**. Disattivando l'ingresso digitale **Di00M00** l'uscita digitale **Do00M00** si disattiva immediatamente.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBeTON	eTON	Auto	No	0	..	eTON (Timer On function block)
2	OutValue	UDINT	Auto	No	0	..	Output value

Esempio LD (Ptp115a100)



Esempio IL

```

CAL FBeTON (* Call the eTON function block *)
LD Di00M00
ST FBeTON.IN (* Transfer the digital input to timer input *)

LD 1000
ST FBeTON.PT (* Set the delay time *)

LD FBeTON.Q
ST Do00M00 (* When time is passed the digital output is set *)

LD FBeTON.ET
ST OutValue (* The counting time is copied to variable *)
    
```

Esempio ST

```

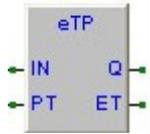
FBeTON(PT:=1000); (* Call the eTON function block *)

FBeTON.IN:=Di00M00; (* Transfer the digital input to timer input *)
OutValue:=FBeTON.ET; (* The counting time is copied to variable *)
Do00M00:=F_eTON.Q; (* When time is passed the digital output is set *)
    
```

Type	Library	Version
FB	ePLCStdLib	SFR053A000

7.5.3 eTP, Timer pulse

Questo blocco funzione esegue la temporizzazione su impulso di attivazione, attivando l'ingresso **IN** l'uscita **Q** si attiva, in uscita **ET** è ritornato il tempo trascorso (in mS) dall'impulso di attivazione. Raggiunto il tempo impostato **PT** (in mS), indipendentemente dallo stato dell'ingresso **IN**, l'uscita **Q** si azzerà, mentre il tempo in uscita su **ET** si azzerà solo se ingresso **IN** non è più attivo.



- IN** (BOOL) Ingresso timer, attivandolo si attiva l'uscita Q ed inizia il conteggio, dopo il tempo definito in PT indipendentemente dallo stato dell'ingresso IN, l'uscita Q si azzerà.
- PT** (UDINT) Preset tempo, definisce il tempo di attivazione dell'uscita Q, espresso in mS.
- Q** (BOOL) Uscita timer, si attiva all'attivazione dell'ingresso IN per il tempo definito in PT.
- ET** (UDINT) Tempo timer, inizia conteggio da attivazione ingresso IN, raggiunto tempo impostato in PT si arresta conteggio, espresso in mS.

Esempi

Il timer è pre-settato a 5 secondi (5000 mS). Attivando l'ingresso digitale **Di00M00** si attiva immediatamente l'uscita digitale **Do00M00** ed il valore di tempo nella variabile **VarOut** inizia il conteggio. Raggiunto il tempo definito 5 secondi, l'uscita **Do00M00** si azzerà mentre il valore di tempo nella variabile **VarOut** rimane bloccato sul valore di preset (5000 mS) sino alla disattivazione dell'ingresso **Di00M00**.

Disattivando l'ingresso digitale **Di00M00** durante la temporizzazione non si hanno ripercussioni nè sullo stato dell'uscita **Do00M00**, nè sul valore della variabile **VarOut**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBeTP	eTP	Auto	No	0	..	eTP (Timer pulse function block)
2	OutValue	UDINT	Auto	No	0	..	Output value

Esempio LD (Ptp115a100)



Esempio IL

```

CAL FBeTP (* Call the eTP function block *)

LD Di00M00
ST FBeTP.IN (* Transfer the digital input to timer input *)

LD 5000
ST FBeTP.PT (* Set the delay time *)

LD FBeTP.Q
ST Do00M00 (* When time is passed the digital output is set *)

LD FBeTP.ET
ST OutValue (* The counting time is copied to variable *)
    
```

Esempio ST

```

FBeTP(PT:=5000); (* Call the eTP function block *)

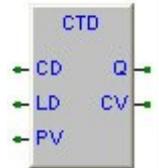
FBeTP.IN:=Di00M00; (* Transfer the digital input to timer input *)
OutValue:=FBeTP.ET; (* The counting time is copied to variable *)
Do00M00:=FBeTP.Q; (* When time is passed the digital output is set *)
    
```

7.6 Funzioni ed FB per gestione counters

7.6.1 CTD, Counter Down

Type	Library	Version
FB	ePLCStdLib	SFR053A000

Questo blocco funzione esegue la gestione di un counter in decremento. Agendo sull'ingresso di load **LD** è possibile in qualsiasi momento trasferire il valore di preset definito su **PV** nel counter **CV**. Ad ogni fronte di attivazione dell'ingresso **CD**, il valore del counter **CV** viene decrementato, quando il valore raggiunge 0, l'uscita **Q** viene settata ed il conteggio si arresta. Solo agendo sull'ingresso di load **LD** è possibile premettere il counter e fare ripartire un nuovo conteggio.



L'ingresso di load LD è prioritario sull'ingresso di decremento CD.

- CD** (BOOL) Comando decremento counter, ad ogni fronte attivazione il valore del counter CV si decrementa.
- LD** (BOOL) Comando di load, attivando l'ingresso il valore di preset PV, viene trasferito nel valore del counter CV.
- PV** (INT) Valore di preset, attivando l'ingresso di load LD, viene trasferito nel valore del counter CV.
- Q** (BOOL) Uscita counter, attiva se il valore del counter CV raggiunge il valore 0.
- CV** (INT) Valore counter, valore di conteggio counter, quando raggiunge il valore 0, l'uscita Q si attiva ed il counter non si decrementa più.

Esempi

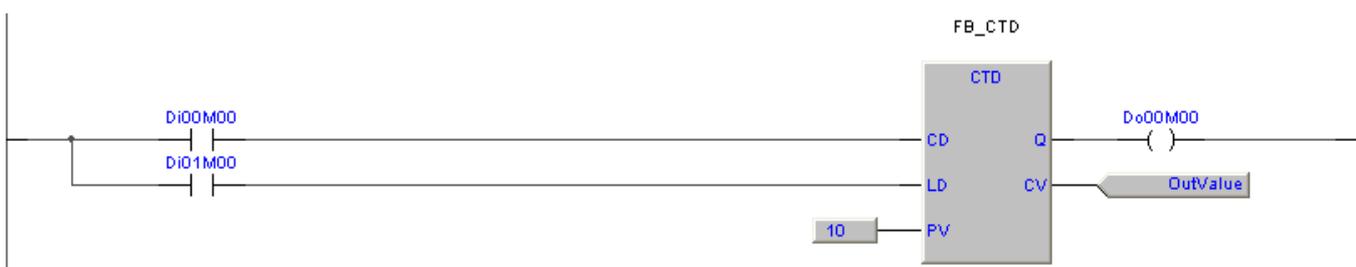
Il counter è premettuto a 10 ed il suo valore in uscita CV è copiato nella variabile **OutValue**. Attivando l'ingresso digitale **Di01M00** il counter viene premettuto ed il suo valore in uscita CV è posto a 10, resettando anche l'uscita Q.

Sul fronte di attivazione dell'ingresso digitale **Di00M00** il counter è decrementato di 1, quando il valore di conteggio si azzerava, il conteggio si arresta e viene attivata l'uscita del counter Q che attiva l'uscita digitale **Do00M00**. Per fare ripartire il conteggio occorre attivare l'ingresso digitale **Di01M00** che premette il counter.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FB_CTD	CTD	Auto	No	0	..	CTD (Counter Down function block)
2	OutValue	INT	Auto	No	0	..	Output value

Esempio LD (Ptp115a100)



Esempio IL

```

CAL FB_CTD (* Call the CTD function block *)
LD 10
ST FB_CTD.PV (* Preset value *)

LD Di00M01
ST FB_CTD.CD (* On the raising edge of digital input the counter count down *)

LD Di01M00
ST FB_CTD.LD (* If the digital input is set the PV value is loaded *)

LD FB_CTD.Q
ST Do00M00 (* If the counter value is 0 the digital output is set *)

LD FB_CTD.CV
ST OutValue (* The counter value is copied to the variable *)
    
```

Esempio ST

```

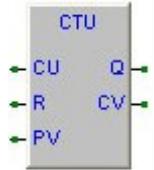
FB_CTD(PV:=10); (* Call the CTD function block and preset counter *)

FB_CTD.CD:=Di00M00; (* On the raising edge of digital input the counter count down *)
FB_CTD.LD:=Di01M00; (* If the digital input is set the PV value is loaded *)
Do00M00:=FB_CTD.Q; (* If the counter value is 0 the digital output is set *)
OutValue:=FB_CTD.CV; (* The counter value is copied to the variable *)
    
```

7.6.2 CTU, Counter Up

Type	Library	Version
FB	ePLCStdLib	SFR053A000

Questo blocco funzione esegue la gestione di un counter in incremento. Agendo sull'ingresso di reset **R** è possibile in qualsiasi momento azzerare il valore del counter **CV**. Ad ogni fronte di attivazione dell'ingresso **CU**, il valore del counter **CV** viene incrementato. Quando il valore del counter **CV** raggiunge il valore di preset, definito su **PV**, l'uscita **Q** viene settata ed il conteggio si arresta. Solo agendo sull'ingresso di reset **R** si potrà resettare il counter e fare ripartire un nuovo conteggio.



L'ingresso di reset R è prioritario sull'ingresso di incremento CU.

- CU** (BOOL) Comando incremento counter, ad ogni fronte attivazione il valore del counter CV si incrementa.
- R** (BOOL) Comando di reset, attivando l'ingresso il valore del counter CV si resetta.
- PV** (INT) Valore di preset, quando il valore del counter CV raggiunge questo valore l'uscita Q si attiva ed il counter non si incrementa più.
- Q** (BOOL) Uscita counter, attiva se il valore del counter CV raggiunge il valore definito in preset PV.
- CV** (INT) Valore counter, valore di conteggio counter, quando raggiunge il valore di preset PV, l'uscita Q si attiva ed il counter non si incrementa più.

Esempi

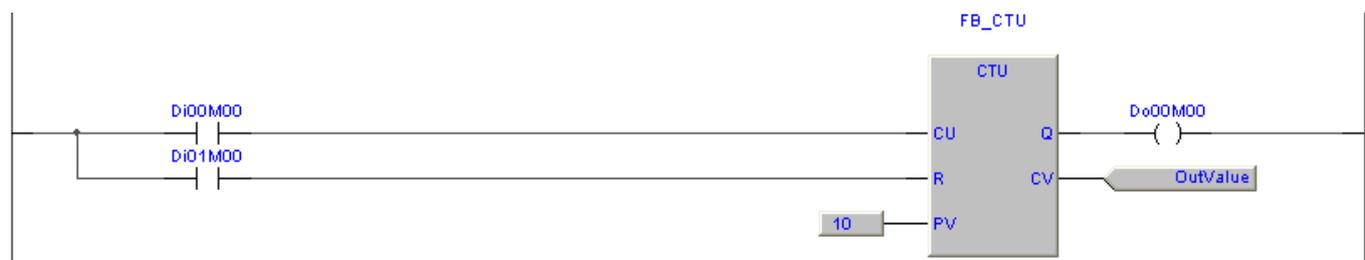
Il counter è presetto a 10 ed il suo valore in uscita CV è copiato nella variabile **OutValue**. Attivando l'ingresso digitale **Di01M00** il counter viene resettato ed il suo valore in uscita CV è posto a 0, resettando anche l'uscita Q.

Sul fronte di attivazione dell'ingresso digitale **Di00M00** il counter è incrementato di 1, quando il valore di conteggio raggiunge il valore di preset, il conteggio si arresta e viene attivata l'uscita del counter Q che attiva l'uscita digitale **Do00M00**. Per fare ripartire il conteggio occorre attivare l'ingresso digitale **Di01M00** che resetta il counter.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FB_CTU	CTU	Auto	No	0	..	CTU (Counter Up function block)
2	OutValue	INT	Auto	No	0	..	Output value

Esempio LD (Ptp115a100)



Esempio IL

```

CAL FBCTU (* Call the CTU function block *)
LD 10
ST FBCTU.PV (* Preset counter *)

LD Di00M00
ST FB_CTU.CU (* On the raising edge of digital input the counter count up *)

LD Di01M00
ST FB_CTU.R (* If the digital input is set the counter is reset *)

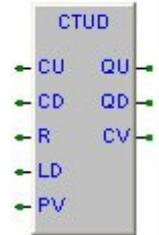
LD FBCTU.Q
ST Do00M00 (* If the counter value has reached the preset the digital output is set *)

LD FB_CTU.CV
ST OutValue (* The counter value is copied to the variable *)
    
```

Type	Library	Version
FB	ePLCStdLib	SFR053A000

7.6.3 CTUD, Counter Up/Down

Questo blocco funzione esegue la gestione di un counter in incremento e decremento. Agendo sull'ingresso di reset **R** è possibile in qualsiasi momento azzerare il valore del counter **CV**. Agendo sull'ingresso di load **LD** è possibile in qualsiasi momento trasferire il valore di preset definito su **PV** nel counter **CV**.



Ad ogni fronte di attivazione dell'ingresso **CU**, il valore del counter **CV** viene incrementato. Quando il valore del counter **CV** raggiunge il valore di preset, definito su **PV**, l'uscita **Q** viene settata ed il conteggio si arresta. Solo agendo sull'ingresso di reset **R** si potrà resettare il counter e fare ripartire un nuovo conteggio.

Ad ogni fronte di attivazione dell'ingresso **CD**, il valore del counter **CV** viene decrementato, quando il valore raggiunge 0, l'uscita **Q** viene settata ed il conteggio si arresta. Solo agendo sull'ingresso di load **LD** è possibile presetare il counter e fare ripartire un nuovo conteggio.

- CU** (BOOL) Comando incremento counter, ad ogni fronte attivazione il valore del counter CV si incrementa
- CD** (BOOL) Comando decremento counter, ad ogni fronte attivazione il valore del counter CV si decrementa.
- R** (BOOL) Comando di reset, attivando l'ingresso il valore del counter CV si resetta.
- LD** (BOOL) Comando di load, attivando l'ingresso il valore di preset PV, viene trasferito nel valore del counter CV.
- PV** (INT) Valore di preset, quando il valore del counter CV raggiunge questo valore l'uscita Q si attiva ed il counter non si incrementa più.
- QU** (BOOL) Uscita counter up, attiva se il valore del counter CV raggiunge il valore definito in preset PV.
- QD** (BOOL) Uscita counter down, attiva se il valore del counter CV raggiunge il valore 0.
- CV** (INT) Valore counter, valore di conteggio counter, quando raggiunge il valore di preset PV, l'uscita Q si attiva ed il counter non si incrementa più.

Esempi

Il counter è presetato a 10 ed il suo valore in uscita CV è copiato nella variabile **VarOut**. Attivando l'ingresso digitale **Di02M00** il counter viene resettato ed il suo valore in uscita CV è posto a 0, resettando anche l'uscita QU. Attivando l'ingresso digitale **Di03M00** il counter viene presetato ed il suo valore in uscita CV è posto a 10, resettando anche l'uscita QD.

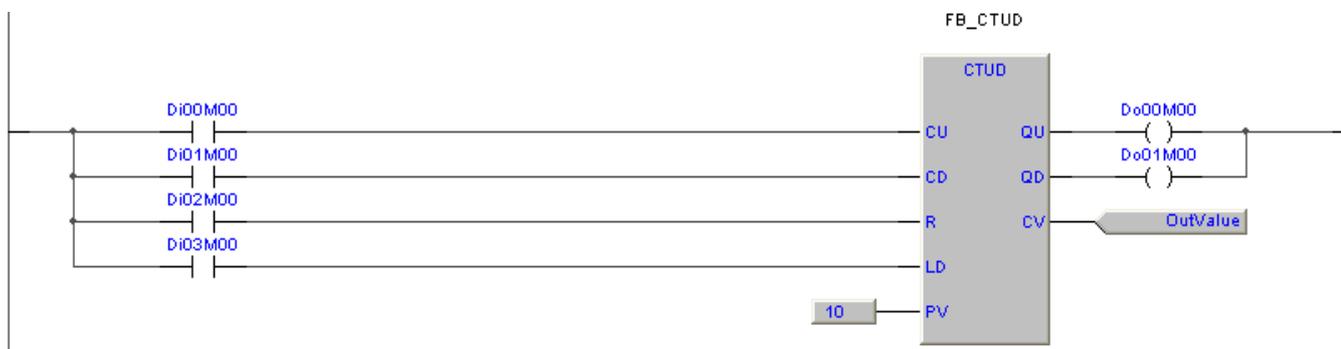
Sul fronte di attivazione dell'ingresso digitale **Di00M00** il counter è incrementato di 1, quando il valore di conteggio raggiunge il valore di preset, il conteggio si arresta e viene attivata l'uscita del counter QU che attiva l'uscita digitale **Do00M00**. Per fare ripartire il conteggio occorre attivare l'ingresso digitale **Di02M00** che resetta il counter.

Sul fronte di attivazione dell'ingresso digitale **Di01M00** il counter è decrementato di 1, quando il valore di conteggio si azzerà, il conteggio si arresta e viene attivata l'uscita del counter QD che attiva l'uscita digitale **Do01M00**. Per fare ripartire il conteggio occorre attivare l'ingresso digitale **Di03M00** che presetta il counter.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FB_CTUD	CTUD	Auto	No	0	..	CTUD (Counter Up/Down function block)
2	OutValue	INT	Auto	No	0	..	Output value

Esempio LD (Ptp115a100)



Esempio IL

```

CAL FB_CTUD (* Call the CTUD function block *)
LD 10
ST FB_CTUD.PV (* Preset value *)

LD Di00M00
ST FB_CTUD.CU (* On the raising edge of digital input the counter count up *)

LD Di01M00
ST FB_CTUD.CD (* On the raising edge of digital input the counter count down *)

LD Di02M00
ST FB_CTUD.R (* If the digital input is set the counter is reset *)

LD Di03M00
ST FB_CTUD.LD (* If the digital input is set the PV value is loaded *)

LD FB_CTUD.QU
ST Do00M00 (* If the counter value has reached the preset the digital output is set *)

LD FB_CTUD.QD
ST Do01M00 (* If the counter value is 0 the digital output is set *)

LD FB_CTUD.CV
ST OutValue (* The counter value is copied to the variable *)
    
```

Esempio ST

```

FB_CTUD(PV:=10); (* Call the CTD function block and preset counter *)

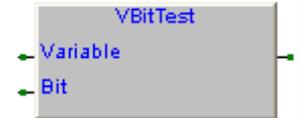
FB_CTUD.CU:=Di00M00; (* On the raising edge of digital input the counter count up *)
FB_CTUD.CD:=Di01M00; (* On the raising edge of digital input the counter count down *)
FB_CTUD.R:=Di02M00; (* If the digital input is set the counter is reset *)
FB_CTUD.LD:=Di03M00; (* If the digital input is set the PV value is loaded *)
Do00M00:=FB_CTUD.QU; (* If the counter value has reached the preset the digital output is set *)
Do01M00:=FB_CTUD.QD; (* If the counter value is 0 the digital output is set *)
OutValue:=FBCTUD.CV; (* The counter value is copied to the variable *)
    
```

7.7 Funzioni ed FB per conversione dati

7.7.1 VBitTest, Variable bit test

Type	Library	Version
Function	ePLCAuxLib	SFR058A000

Questa funzione esegue il test di un bit in una variabile.



Parametri funzione:

Variable (UDINT) Variabile in cui testare il bit.

Bit (USINT) Numero del bit da testare (Range da 0 a 31).

La funzione ritorna:

(BOOL) Stato bit indicato.

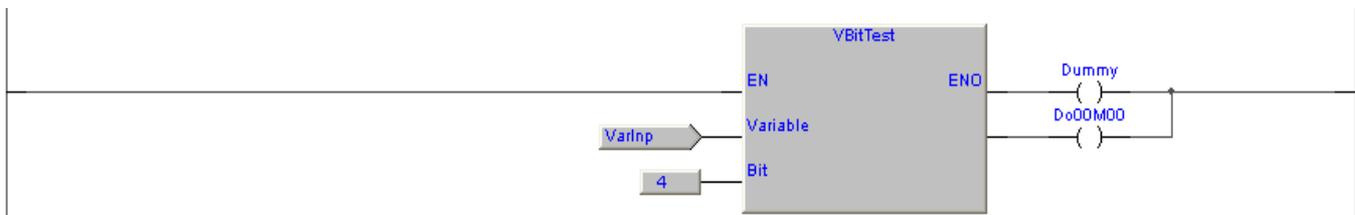
Esempi

Lo stato del bit 4 della variabile **VarInp** viene trasferito sull'uscita digitale **Do00M00**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable
2	VarInp	UDINT	Auto	No	0	..	Variable input

Esempio LD



Esempio IL

```
LD VarInp (* Variable input *)
VBitTest 4 (* Variable bit test *)
ST Do00M00 (* Transfer bit status to digital output *)
```

Esempio ST

```
Do00M00:=VBitTest(VarInp, 4); (* Variable bit test *)
```

Type	Library	Version
Function	ePLCAuxLib	SFR058A000

7.7.2 VBitSet, Variable bit set

Questa funzione esegue il set di un bit in una variabile.



Parametri funzione:

- Value** (BOOL) Valore bit da settare.
- Variable** (UDINT) Variabile in cui settare il bit.
- Bit** (USINT) Numero del bit da settare (Range da 0 a 31).

La funzione ritorna:

- (UDINT) Valore variabile dopo il set del bit.

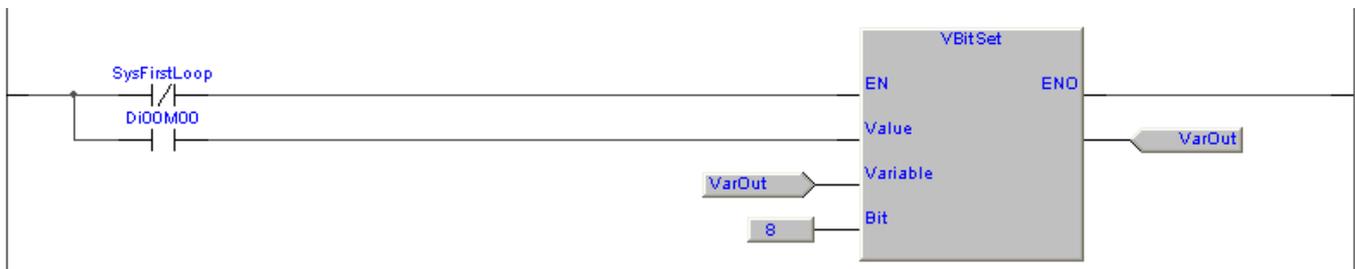
Esempi

Lo stato dell'ingresso digitale **Di00M00** è trasferito nel bit 8 della variabile **VarOut**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	VarOut	UDINT	Auto	No	0	..	Variable output

Esempio LD



Esempio IL

```
LD Di00M00 (* Variable input *)
VBitSet VarOut, 8 (* Variable bit set *)
ST VarOut (* Transfer result to variable *)
```

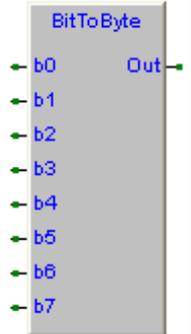
Esempio ST

```
VarOut:=VBitSet(Di00M00, VarOut, 8); (* Variable bit set *)
```

Type	Library	Version
FB	ePLCUtyLib	SFR054A000

7.7.3 BitToByte, Bit to byte conversion

Questo blocco funzione permette di convertire 8 variabili **BOOL** in una variabile **BYTE**.



- b0** (BOOL) Bit 0 del byte di **Out**.
- ...
- b7** (BOOL) Bit 7 del byte di **Out**.
- Out** (BYTE) Risultato conversione ingressi a bit.

Esempi

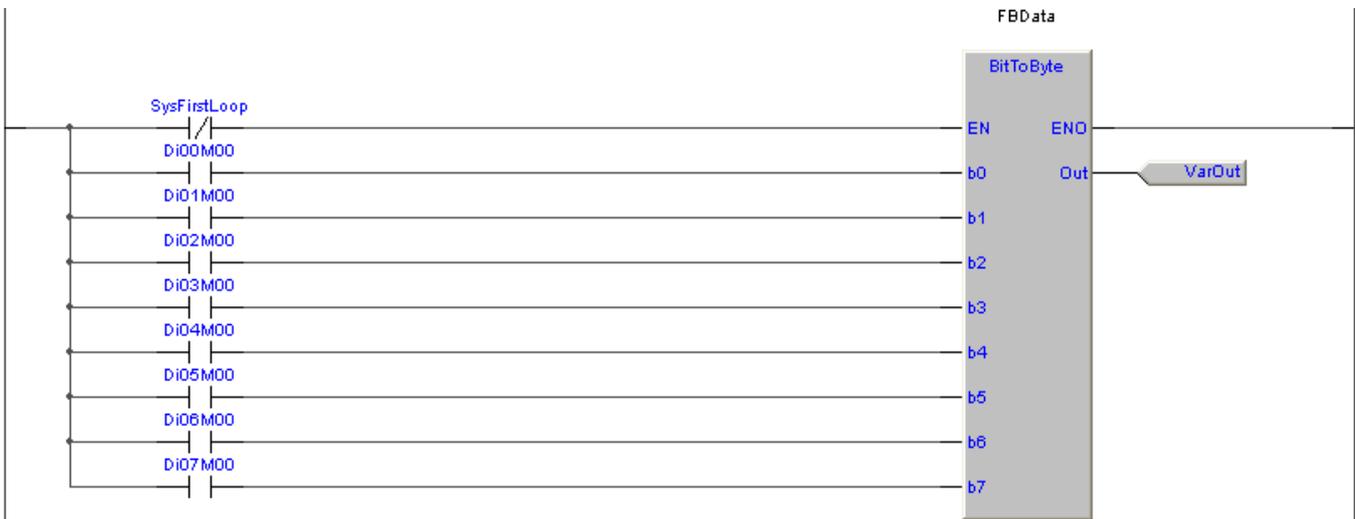
Gli 8 ingressi digitali del modulo 0 sono trasferiti nella variabile **VarOut**. Attivando il solo ingresso digitale **Di00M00** la variabile **VarOut** assumerà valore 1, attivando il solo ingresso digitale **Di01M00** la variabile **VarOut** assumerà valore 2, e così via fino all'ingresso **Di07M00** attivando il quale la variabile **VarOut** assumerà valore 128. Attivando più ingressi contemporaneamente la variabile **VarOut** assumerà valore pari alla somma degli ingressi attivati.

Per semplicità negli esempi IL e ST non vengono riportati tutti i bits.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBData	BitToByte	Auto	No	0	..	FB Bit to byte data
2	VarOut	UDINT	Auto	No	0	..	Variable output

Esempio LD (Ptp114a100)



Esempio IL (Ptp114a100)

```

LD Di00M00
ST FBData.b0 (* Transfer digital input to input bit *)

LD Di07M00
ST FBData.b7 (* Transfer digital input to input bit *)

CAL FBData (* Call the BitToByte function block *)

LD FBData.Out
ST VarOut (* Transfer the result to variable *)
    
```

Esempio ST

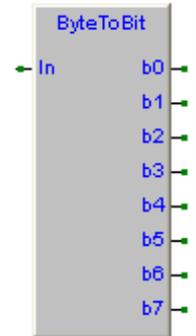
```
FBData.b0:=Di00M00; (* Transfer digital input to input bit *)  
FBData.b7:=Di07M00; (* Transfer digital input to input bit *)  
  
FBData(); (* Call the BitToByte function block *)  
  
VarOut:=FBData.Out; (* Transfer the result to variable *)
```

Type	Library	Version
FB	ePLCUtyLib	SFR054A000

7.7.4 ByteToBit, Byte to bit conversion

Questo blocco funzione permette di convertire una variabile **BYTE** in 8 variabili **BOOL**.

- In** (BYTE) Valore byte da convertire
- b0** (BOOL) Bit 0 di *In*.
- ...
- b7** (BOOL) Bit 7 di *In*.



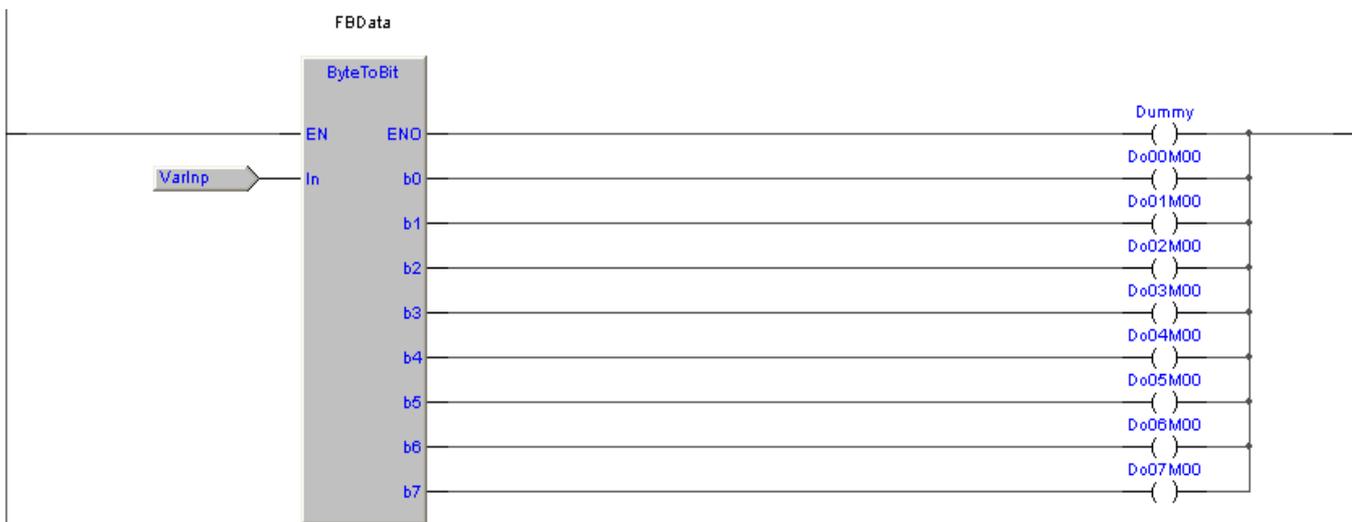
Esempi

Lo stato del bit 0 della variabile **VarInp** viene trasferito sull'uscita digitale **Do00M00** lo stato del bit 1 della variabile **VarInp** viene trasferito sull'uscita digitale **Do01M00** e così via fino allo stato del bit 7 della variabile **VarInp** viene trasferito sull'uscita digitale **Do07M00**. Per semplicità negli esempi IL e ST non vengono riportati tutti i bits.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	VarInp	USINT	Auto	No	0	..	Variable input
2	FBData	ByteToBit	Auto	No	0	..	FB Byte to bit data
3	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable

Esempio LD (Ptp114a100)



Esempio IL (Ptp114a100)

```
LD VarInp
ST FBData.In (* Transfer the variable to input *)

CAL FBData (* Call the ByteToBit function block *)

LD FBData.b0
ST Di00M00 (* Transfer output bit to digital output *)
```

Esempio ST (Ptp114a100)

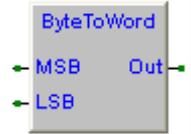
```
FBData(In:=VarInp); (* Call the ByteToBit function block *)

Do00M00:=FBData.b0; (* Transfer output bit to digital output *)
Do01M00:=FBData.b1; (* Transfer output bit to digital output *)
```

Type	Library	Version
FB	ePLCUtyLib	SFR054A000

7.7.5 ByteToWorld, Byte to word conversion

Questo blocco funzione permette di convertire due variabili **BYTE** in una variabile **WORD**.



MSB_(BYTE) MSB del valore in uscita **Out**

LSB_(BYTE) LSB del valore in uscita **Out**

Out_(WORD) Valore in uscita

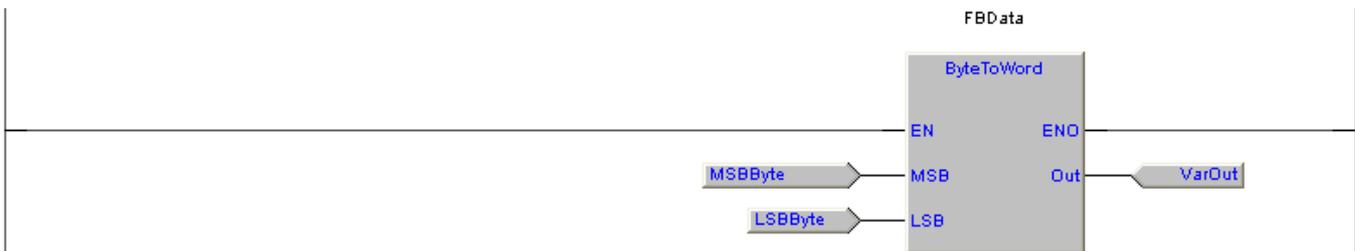
Esempi

Le due variabili **MSBByte** e **LSBByte** sono uniti nella variabile **VarOut** in uscita.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBData	ByteToWorld	Auto	No	0	..	FB Byte to word data
2	LSBByte	BYTE	Auto	No	0	..	Valore byte MSB
3	MSBByte	BYTE	Auto	No	0	..	Valore byte LSB
4	VarOut	WORD	Auto	No	0	..	Variable output

Esempio LD (Ptp114a100)



Esempio IL (Ptp114a100)

```

LD MSBByte
ST FBData.MSB (* Transfer the MSB variable to input *)

LD LSBByte
ST FBData.LSB (* Transfer the LSB variable to input *)

CAL FBData (* Call the ByteToWorld function block *)

LD FBData.Out
ST VarOut (* Transfer output to variable *)
    
```

Esempio ST (Ptp114a100)

```

FBData.MSB:=MSBByte; (* Transfer the MSB variable to input *)
FBData.LSB:=LSBByte; (* Transfer the LSB variable to input *)

FBData(); (* Call the ByteToWorld function block *)

VarOut:=FBData.Out; (* Transfer output to variable *)
    
```

Type	Library	Version
FB	ePLCUtyLib	SFR054A000

7.7.6 WordToByte, Word to byte conversion

Questo blocco funzione permette di convertire una variabile **WORD** in due variabili **BYTE**.



- IN** (WORD) Variabile da convertire.
- MSB** (BYTE) MSB del valore in ingresso.
- LSB** (BYTE) LSB del valore in ingresso.

Esempi

Le variabile **VarInp** è divisa nelle due variabili **MSBByte** e **LSBByte**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	LSBByte	BYTE	Auto	No	0	..	Valore byte MSB
2	MSBByte	BYTE	Auto	No	0	..	Valore byte LSB
3	VarInp	WORD	Auto	No	0	..	Variable input
4	FBData	WordToByte	Auto	No	0	..	FB Word to byte data

Esempio LD (Ptp114a100)



Esempio IL (Ptp114a100)

```
LD VarInp
ST FBData.In (* Transfer the variable to input *)

CAL FBData (* Call the WordToByte function block *)

LD FBData.MSB
ST MSBByte (* Transfer the MSB output to variable *)

LD FBData.LSB
ST LSBByte (* Transfer the LSB output to variable *)
```

Esempio ST (Ptp114a100)

```
FBData.In:=VarInp; (* Transfer the variable to input *)

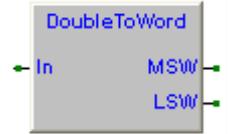
FBData(); (* Call the WordToByte function block *)

MSBByte:=FBData.MSB; (* Transfer the MSB output to variable *)
LSBByte:=FBData.LSB; (* Transfer the LSB output to variable *)
```

7.7.7 DoubleToWorld, Double to word conversion

Type	Library	Version
FB	ePLCUtyLib	SFR054A000

Questo blocco funzione permette di convertire una variabile DWORD in due variabili WORD.



- IN** _(DWORD) Variabile da convertire.
- MSW** _(WORD) MSW del valore in ingresso.
- LSW** _(WORD) LSW del valore in ingresso.

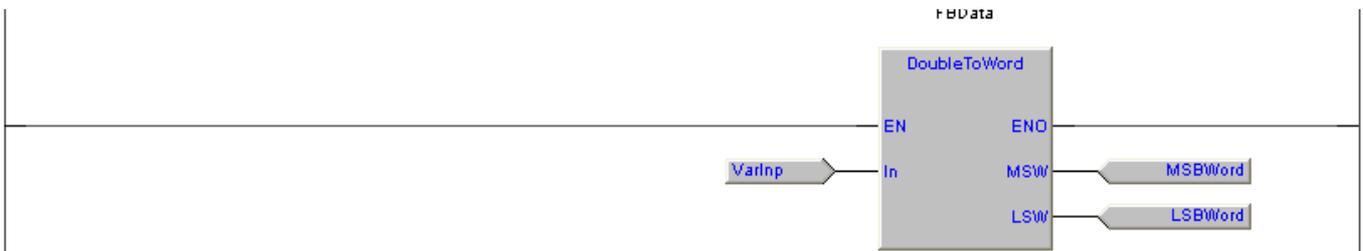
Esempi

Le variabile **VarInp** è divisa nelle due variabili **MSBWord** e **LSBWord**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	LSBWord	WORD	Auto	No	0	..	Valore word MSB
2	MSBWord	WORD	Auto	No	0	..	Valore word LSB
3	VarInp	DWORD	Auto	No	0	..	Variable input
4	FBData	DoubleToWor	Auto	No	0	..	FB Word to byte data

Esempio LD (Ptp114a100)



Esempio IL (Ptp114a100)

```

LD VarInp
ST FBData.In (* Transfer the variable to input *)

CAL FBData (* Call the DoubleToWorld function block *)

LD FBData.MSW
ST MSBWord (* Transfer the MSW output to variable *)

LD FBData.LSW
ST LSBWord (* Transfer the LSW output to variable *)
    
```

Esempio ST (Ptp114a100)

```

FBData.In:=VarInp; (* Transfer the variable to input *)

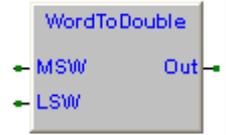
FBData(); (* Call the DoubleToWorld function block *)

MSBWord:=FBData.MSW; (* Transfer the MSW output to variable *)
MSBWord:=FBData.LSW; (* Transfer the LSW output to variable *)
    
```

7.7.8 WordToDouble, Word to double conversion

Type	Library	Version
FB	ePLCUtyLib	SFR054A000

Questo blocco funzione permette di convertire due variabili **WORD** in una variabile **DWORD**.



MSW_(WORD) MSB del valore in uscita **Out**

LSW_(WORD) LSB del valore in uscita **Out**

Out_(DWORD) Valore in uscita

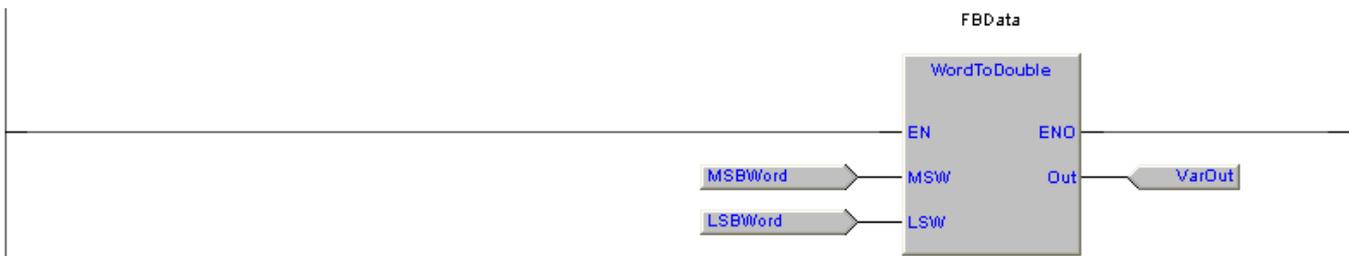
Esempi

Le due variabili **MSBWord** e **LSBWord** sono uniti nella variabile **VarOut** in uscita.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	LSBWord	WORD	Auto	No	0	..	Valore word MSB
2	MSBWord	WORD	Auto	No	0	..	Valore word LSB
3	VarOut	DWORD	Auto	No	0	..	Variable output
4	FBData	WordToDouble	Auto	No	0	..	FB Word to double data

Esempio LD (Ptp114a100)



Esempio IL (Ptp114a100)

```

LD MSBWord
ST FBData.MSW (* Transfer the MSW variable to input *)

LD LSBWord
ST FBData.LSW (* Transfer the LSW variable to input *)

CAL FBData (* Call the WordToDouble function block *)

LD FBData.Out
ST VarOut (* Transfer output to variable *)
    
```

Esempio ST (Ptp114a100)

```

FBData.MSW:=MSBWord; (* Transfer the MSW variable to input *)
FBData.LSW:=LSBWord; (* Transfer the LSW variable to input *)

FBData(); (* Call the WordToDouble function block *)

VarOut:=FBData.Out; (* Transfer output to variable *)
    
```

Type	Library	Version
Function	ePLCAuxLib	SFR058A000

7.7.9 ToLower, Uppercase to lowercase letter conversion

Questa funzione converte un carattere dal formato maiuscolo nel corrispondente carattere in formato minuscolo.



Parametri funzione:

Char (USINT) Carattere da convertire.

La funzione ritorna:

(USINT) Carattere nel formato lowercase.

Esempi

La variabile **Upper** viene convertita nel corrispondente valore lowercase e trasferita in **Lower**. Il valore di inizializzazione 16#41 che corrisponde alla lettera **A**, viene convertito nel valore 16#61 che corrisponde alla lettera **a**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Upper	USINT	Auto	No	16#41	..	Uppercase letter
2	Lower	USINT	Auto	No	0	..	Lowercase letter

Esempio LD



Esempio IL

```
LD Upper (* Uppercase letter *)
  ToLower (* Uppercase to lowercase letter conversion *)
ST Lower (* Lowercase letter *)
```

Esempio ST

```
Lower:=ToLower(Upper); (* Uppercase to lowercase letter conversion *)
```

Type	Library	Version
Function	ePLCAuxLib	SFR058A000

7.7.10 ToUpper, Lowercase to uppercase letter conversion

Questa funzione converte un carattere dal formato minuscolo nel corrispondente carattere in formato maiuscolo.



Parametri funzione:

Char (USINT) Carattere da convertire.

La funzione ritorna:

(USINT) Carattere nel formato uppercase.

Esempi

La variabile **Lower** viene convertita nel corrispondente valore uppercase e trasferita in **Upper**. Il valore di inizializzazione 16#61 che corrisponde alla lettera **a**, viene convertito nel valore 16#41 che corrisponde alla lettera **A**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Lower	USINT	Auto	No	16#61	..	Lowercase letter
2	Upper	USINT	Auto	No	0	..	Uppercase letter

Esempio LD



Esempio IL

```
LD Lower (* Lowercase letter *)
ToUpper (* Lowercase to uppercase letter conversion *)
ST Upper (* Lowercase letter *)
```

Esempio ST

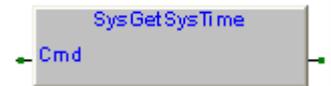
```
Upper:=ToUpper(Lower); (* Lowercase to uppercase letter conversion *)
```

7.8 Funzioni ed FB di utilità sistema

Type	Library	Version
Function	Embedded	3.0

7.8.1 SysGetSysTime, get system time

Questa funzione ritorna il tempo di sistema espresso in uS. E' possibile definire con il valore di **Cmd** se si vuole avere il tempo di sistema attuale (**Cmd:=TRUE**) oppure quello memorizzato con la precedente esecuzione della funzione (**Cmd:=FALSE**).



Parametri funzione:

Cmd (BOOL) Indica il valore di tempo che deve essere ritornato.
TRUE: Viene salvato e ritornato il valore attuale di tempo.
FALSE: Viene ritornato il tempo salvato dalla precedente chiamata con **Cmd:=TRUE**.

La funzione ritorna:

(UDINT) Tempo di sistema espresso in uS.

Esempi

Viene calcolato il tempo in cui l'ingresso digitale **Di00M00** rimane nella condizione di attivo.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	StartTime	UDINT	Auto	No	0	..	Input raising time reference (uS)
2	SetTime	UDINT	Auto	No	0	..	Input set time (uS)
3	Pulse	BOOL	Auto	No	FALSE	..	Pulse flag

Esempio ST (Ptp116a100)

```
(* Check if input is activated. *)
IF (Di00M00 <> Pulse) THEN
  Pulse:=Di00M00; (* Pulse flag *)

  (* On input raising edge relate time is saved. *)
  IF (Di00M00) THEN StartTime:=SysGetSysTime(TRUE); END_IF;

  (* On input falling edge the set time is calculated. *)
  IF (NOT(Di00M00)) THEN SetTime:=SysGetSysTime(TRUE)-StartTime; END_IF;
END_IF;
```

7.8.1.1 Calcolo timeout

Essendo il valore di tempo di sistema ritornato dalla funzione un numero **UDINT** che si incrementa ogni uS, ed al valore massimo esegue overflow a zero, non è possibile effettuare comparazioni dirette con il tempo di riferimento ma occorre sempre eseguire la differenza.

Nel seguente esempio viene attivato **Timeout** se l'ingresso **Di00M00** rimane attivo per più di un secondo.

```
IF NOT(Di00M00) THEN TimeBf:=SysGetSysTime(TRUE);
ELSE IF ((SysGetSysTime(TRUE)-TimeBf) >= 1000000) THEN Timeout:=TRUE; END_IF;
END_IF;
```

Lo stesso esempio scritto in questo modo non funziona correttamente.

```
IF NOT(Di00M00) THEN TimeBf:=SysGetSysTime(TRUE);
ELSE IF (SysGetSysTime(TRUE) >= (TimeBf+1000000)) THEN Timeout:=TRUE; END_IF;
END_IF;
```

7.8.1.2 Semplice cronometro

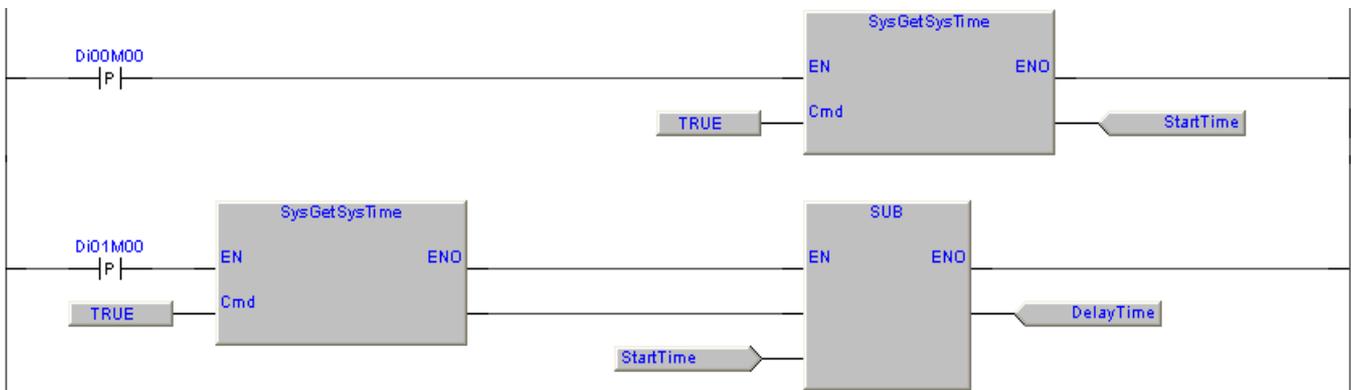
Questo esempio realizza un semplice cronometro per misurare il tempo che intercorre tra un comando di start ed il comando di stop. Utilizzando ad esempio due fotocellule una sulla linea di start ed una sulla linea di stop di un percorso è possibile calcolare il tempo di percorrenza espresso in uSec.

Attivando l'ingresso di start **Di00M00** viene salvato il tempo di sistema allo start nella variabile **StartTime**, attivando l'ingresso di stop **Di01M00** viene calcolato il tempo trascorso tra il tempo salvato allo start ed il tempo nel momento di stop. Il tempo calcolato è salvato nella variabile **DelayTime**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	StartTime	UDINT	Auto	No	0	..	Start time (uS)
2	DelayTime	UDINT	Auto	No	0	..	Delay time (uS)

Esempio LD (Ptp119a000)



Type	Library	Version
Function	Embedded	3.0

7.8.2 SysSetTaskLpTime, set task loop time

Questa funzione permette di impostare il tempo di esecuzione delle tasks PLC. Esistono due tasks eseguite a tempo la task slow **ID_TASK_SLOW** e la task fast **ID_TASK_FAST**, ad ognuna di queste task può essere assegnato un tempo di esecuzione.



Se il tempo impostato non è compreso nel range definito o se il rapporto tra i tempi di esecuzione della task fast rispetto alla slow non sono coerenti la funzione non modifica i tempi di esecuzione e ritorna **FALSE**. Di seguito sono riportati i range di tempo definibili per le varie tasks.

ID_TASK_FAST Range da 100 uS a 10 mS

ID_TASK_SLOW Range da 1 a 100 mS

Parametri funzione:

TaskID (USINT) Identifica la task a cui si vuole definire il tempo di esecuzione secondo le definizioni in [Task ID](#).

Time (UDINT) Indica il valore di tempo di esecuzione task espresso in uS.

La funzione ritorna:

(BOOL) **TRUE:** Se funzione eseguita correttamente
FALSE: In caso di errore esecuzione, esempio parametri errati.

Esempi

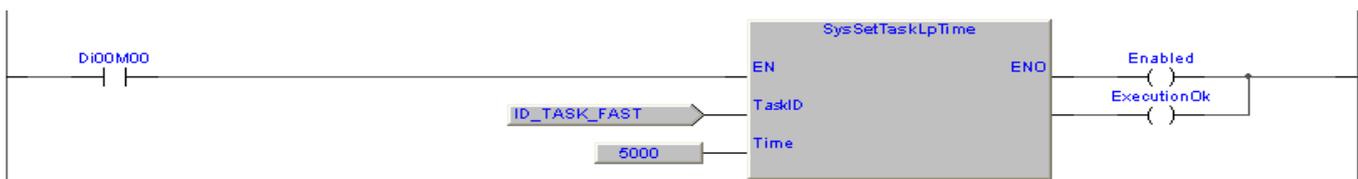
Attivando l'ingresso **Di00M00** viene impostato un tempo di esecuzione di 5 ms per la task PlcFast.

Attenzione! Per aumentare i tempi di esecuzione delle tasks dal valore di default occorre eseguire la funzione nella task di boot.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Enabled	BOOL	Auto	No	FALSE	..	Function enabled
2	ExecutionOk	BOOL	Auto	No	FALSE	..	Function execution ok

Esempio LD (Ptp116a100)



Esempio ST

```

Enabled:=Di00M00; (* Function enabled *)

IF Di00M00 THEN
    ExecutionOk:=SysSetTaskLpTime(TaskID:=ID_TASK_FAST, Time:=5000); (* Function execution ok *)
END_IF;
    
```

Type	Library	Version
Function	Embedded	3.0

7.8.3 SysGetRandom, get random number

Questa funzione ritorna un numero random compreso tra 0.0 e 1.0. E' possibile definire con il valore di **Cmd** se si vuole avere un nuovo numero random (**Cmd:=TRUE**) oppure quello memorizzato con la precedente esecuzione della funzione (**Cmd:=FALSE**).



Parametri funzione:

Cmd (BOOL) Indica il numero random ritornato.
TRUE: Viene salvato e ritornato un nuovo numero random.
FALSE: Viene ritornato il numero salvato dalla precedente chiamata con **Cmd:=TRUE**.

La funzione ritorna:

(REAL) Un numero random compreso nel range da 0.0 a 1.0.

Esempi

Attivando l'ingresso digitale **Di00M00** viene inviato sulla porta seriale **COM0** una sequenza di 10 numeri random.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	RandomNr	UINT	Auto	No	0	..	Random number
2	i	USINT	Auto	No	0	..	Auxiliary counter
3	NrOfChars	INT	Auto	No	0	..	Number of printed chars
4	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer
5	Pulse	BOOL	Auto	No	FALSE	..	Pulse flag

Esempio ST (Ptp116a100)

```
(* Here the COM0 port is opened in read/write. *)
IF (Fp = NULL) THEN
  Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
END_IF;

(* Check if input is activated. *)
F (Di00M00 <> Pulse) THEN
  Pulse:=Di00M00; (* Pulse flag *)

  (* On input raising edge print out 10 random numbers. *)
  IF (Di00M00) THEN
    FOR i:=0 TO (9) BY 1 DO
      RandomNr:=TO_UINT(SysGetRandom(TRUE)*1000.0); (* Random number *)
      NrOfChars:=SysVarfprintf(Fp, 'Rn:%03d$r$n', UINT_TYPE, ADR(RandomNr));
    END_FOR;
  END_IF;
END_IF;
```

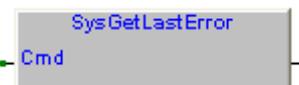
Collegando un terminale seriale alla porta **COM0** impostato a **115200,e,8,1** vedremo un elenco del tipo:

```
Rn:437
Rn:488
Rn:898
...
Rn:261
Rn:944
```

7.8.4 SysGetLastError, get last error

Type	Library	Version
Function	Embedded	5.0

Questa funzione ritorna il numero dell'ultimo errore rilevato da una funzione e/o da un blocco funzione ([Elenco errori](#)). Occorre eseguire la funzione su abilitazione del bit di fault in uscita dalla funzione e/o blocco funzione da controllare. E' possibile definire con il valore di **Cmd** se si vuole avere il valore attuale dell'ultimo errore (**Cmd:=TRUE**) oppure quello memorizzato con la precedente esecuzione della funzione (**Cmd:=FALSE**).



Parametri funzione:

- Cmd** (BOOL) Indica il numero di errore ritornato.
TRUE: Viene ritornato l'ultimo valore di errore.
FALSE: Viene ritornato il numero salvato dalla precedente chiamata con **Cmd:=TRUE**.

La funzione ritorna:

(UDINT) Il numero dell'ultimo errore rilevato

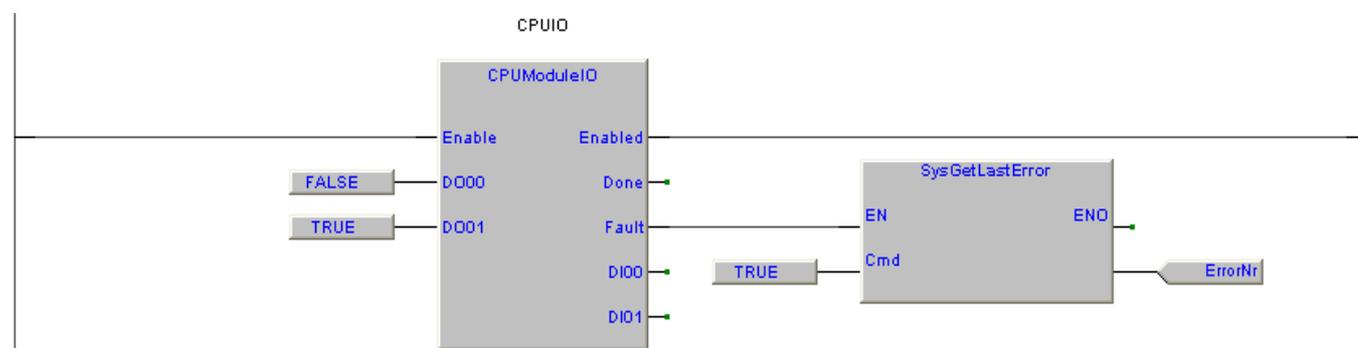
Esempi

Viene salvato l'eventuale errore durante l'esecuzione del blocco funzione **CPUModuleIO**. In caso di errore il numero di errore è trasferito nella variabile **ErrorNr**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	CPUIO	CPUModuleIO.Auto	No	No	0	..	I/O on CPU module
2	ErrorNr	UDINT	Auto	No	0	..	Error number

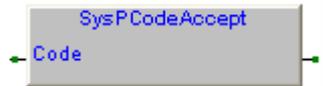
Esempio LD



Type	Library	Version
Function	Embedded	4.0

7.8.5 SysPCodeAccept, accepts the protection code

Alcune funzioni di programma e/o blocchi funzione possono essere protetti da un codice che deve essere ordinato separatamente. Per abilitare l'esecuzione della funzione e/o del blocco funzione occorre sbloccarlo definendone il codice con questa funzione.



La funzione controlla il codice fornito e ritorna **TRUE** se codice accettato. Vedere capitolo [Protezione funzioni e blocchi funzione](#) per ulteriori informazioni.

Parametri funzione:

Code (STRING[20]) Codice di protezione.

La funzione ritorna:

(BOOL) **TRUE**: Codice verificato ID relativo sbloccato. **FALSE**: Codice non verificato.

Codici di errore

In caso di errore la funzione torna **FALSE** e con [SysGetLastError](#) è possibile rilevare il codice di errore.

9991100 Lunghezza **Code** non corretta.

9991110 ÷ 5 Codice definito in **Code** non corretto.

9991200 Non vi è più spazio per eseguire ulteriori funzioni protette.

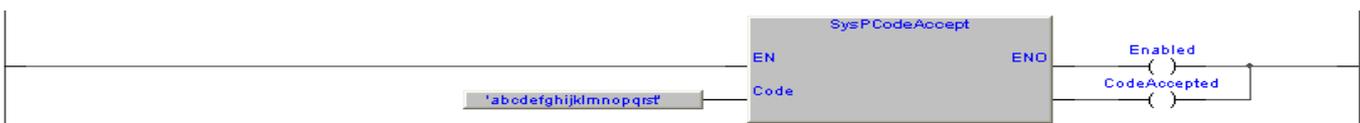
Esempi

E' riportato un semplice programma che esegue il controllo sul codice di sblocco "abcdefghijklmnoqrst". Se il codice è corretto viene attivata la variabile **CodeAccepted**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	CodeAccepted	BOOL	Auto	No	FALSE	..	Protection code accepted
2	Enabled	BOOL	Auto	No	FALSE	..	Function enabled

Esempio LD (Ptp116a100)



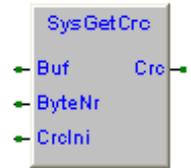
Esempio ST

```
(* Check the protection code. *)
CodeAccepted:=SysPCodeAccept('abcdefghijklmnoqrst'); (* Protection code accepted *)
```

Type	Library	Version
FB	Embedded	3.0

7.8.6 SysGetCrc, get CRC value

Questa funzione esegue il calcolo del CRC **Cyclic Redundancy Check**, (Controllo Ciclico di Ridondanza) su di un'area dati. Il calcolo è effettuato secondo le specifiche richieste dal protocollo **modbus Rtu**



Occorre passare alla funzione l'indirizzo del buffer di memoria **Buf** ed il numero di bytes **ByteNr** su cui eseguire il calcolo del CRC.

- Buf** (@USINT) Indirizzo dell'area di memoria su cui eseguire il calcolo del CRC.
- ByteNr** (UINT) Numero di bytes su cui eseguire il calcolo del CRC a partire dall'indirizzo definito in **Buf**.
- CRCIni** (UINT) Valore di inizializzazione del CRC da calcolare.
- CRC** (UINT) Valore CRC calcolato.

Codici di errore

In caso di errore con **SysGetLastError** è possibile rilevare il codice di errore.

- 9978050 Errore allocazione blocco funzione.
- 9978070 Errore versione blocco funzione.

Esempi

Viene calcolato il CRC di un frame modbus Rtu per il comando di lettura registri **Read holding registers**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Frame	USINT	Auto	[0..9]	10(0)	..	Frame array
2	RegsAddress	UINT	Auto	No	0	..	Registers address
3	NrOfRegs	USINT	Auto	No	0	..	Number of registers
4	GetCRC	SysGetCrc	Auto	No	0	..	Get CRC
5	CRCValue	UINT	Auto	No	0	..	CRC value

Esempio ST (Ptp116a100)

```
(* ----- *)
(* Calculate CRC of a modbus Rtu frame for command "Read holding registers". *)
(* +---+---+---+---+---+---+ *)
(* |Nd|03|Addr |NumR |CRC| *)
(* +---+---+---+---+---+---+ *)
(* ----- *)
(* Define the registers address and the number of registers to read. *)

RegsAddress:=16#0120; (* Registers address *)
NrOfRegs:=8; (* Number of registers *)

(* Prepare the command frame. *)

Frame[0]:=1; (* Node address *)
Frame[1]:=3; (* Function code (16#03) *)
Frame[2]:=TO_USINT(RegsAddress/256); (* MSB registers address *)
Frame[3]:=TO_USINT(RegsAddress&255); (* LSB registers address *)
Frame[4]:=0; (* MSB number of registers to read *)
Frame[5]:=NrOfRegs; (* LSB number of registers to read *)

(* Calculate the frame CRC. *)

GetCRC.Buf:=ADR(Frame[0]); (* Buffer address *)
GetCRC.ByteNr:=6; (* Byte number *)
GetCRC.CrcIni:=16#FFFF; (* CRC ini value *)
GetCRC(); (* Calculate CRC *)
CRCValue:=GetCRC.Crc; (* CRC value *)
Frame[6]:=TO_USINT(CRCValue/256); (* MSB of CRC value *)
Frame[7]:=TO_USINT(CRCValue&255); (* LSB of CRC value *)
```

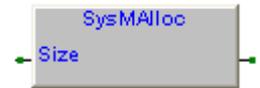
Il valore del CRC del frame modbus Rtu è 16#443A, l'intero frame è visibile ponendo nella finestra di watch la variabile **Frame** così come riportato nella figura.

Frame	Value	Type
-	-	USINT[]
[0]	16#01	USINT
[1]	16#03	USINT
[2]	16#01	USINT
[3]	16#20	USINT
[4]	16#00	USINT
[5]	16#08	USINT
[6]	16#44	USINT
[7]	16#3A	USINT
[8]	16#00	USINT
[9]	16#00	USINT

Type	Library	Version
Function	Embedded	6.0

7.8.7 SysMAlloc, Memory allocation

Questa funzione esegue l'allocazione di uno spazio di memoria della dimensione in byte definita da parametro **Size**. La funzione ritorna il puntatore allo spazio di memoria allocato.



La memoria è allocata nella memoria di sistema e quindi non utilizza la memoria a disposizione del programma utente. Nel caso in cui non vi sia spazio in memoria per l'allocazione del buffer definito, la funzione ritorna **0**.

Parametri funzione:

Size (UDINT) Dimensione in bytes dell'area da allocare.

La funzione ritorna:

(@USINT) Indirizzo allocazione buffer. **NULL** se non vi è spazio per allocare il buffer.

Esempi

Su fronte attivazione ingresso **Di00M00** viene incrementata la variabile **Counter** e la stampa del suo valore trasferita nell'array **StringOut**. Il valore presente in **StringOut** viene poi inviato sulla porta seriale **COM0**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Pulse	BOOL	Auto	No	FALSE	..	Pulse flag
2	Ch	INT	Auto	No	0	..	Written character
3	i	INT	Auto	No	0	..	Auxiliary counter
4	NrOfChars	INT	Auto	No	0	..	Number of printed chars
5	Counter	UDINT	Auto	No	0	..	Counter
6	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer
7	StringOut	@USINT	Auto	No	0	..	String output pointer

Esempio ST (Ptp116a300)

```
(* ----- *)
(* "SysMAlloc" example *)
(* ----- *)
(* Here at first program execution loop allocate memory and open COM. *)

IF (SysFirstLoop) THEN
    StringOut:=SysMAlloc(16); (* String output pointer *)
    Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
END_IF;

IF ((StringOut = 0) OR (Fp = 0)) THEN RETURN; END_IF;

(* On input raising edge the counter value is printed. *)

IF (Di00M00 <> Pulse) THEN
    Pulse:=Di00M00; (* Pulse flag *)

    IF (Di00M00) THEN
        Counter:=Counter+1; (* Counter *)
        NrOfChars:=SysVarsprintf(StringOut, 32, 'Counter:%04d$r$n', UDINT_TYPE, ADR(Counter));

        FOR i:=0 TO NrOfChars DO
            Ch:=Sysfputc(TO_INT(@StringOut), Fp); (* Written character *)
            StringOut:=StringOut+1; (* String output pointer *)
        END_FOR;
    END_IF;
END_IF;

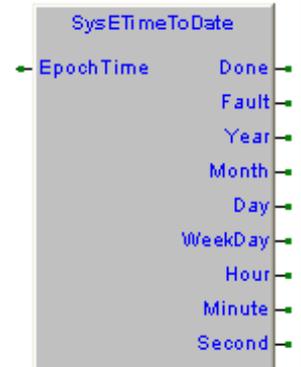
(* [End of file] *)
```

7.9 Funzioni ed FB per gestione Data/Ora

7.9.1 SysETimeToDate, epoch time to date conversion

Type	Library	Version
FB	Embedded	3.0

Questo blocco funzione esegue la conversione della data espressa in epoch time. Occorre fornire al blocco funzione la data espressa nel formato epoch ime come presente nella variabile di sistema **SysDateTime**, in uscita dal blocco funzione avremo i valori di data espressi nel formato Giorno/Mese/Anno ed Ora:Minuti:Secondi.



- EpochTime** (UDINT) Occorre specificare la data espressa in epoch time.
- Done** (BOOL) Attivato al termine della conversione.
- Fault** (BOOL) Errore di conversione, viene attivato in caso di errore nella conversione.
- Year** (UINT) Ritorna il valore di anno (Range da 1970 a 2099)
- Month** (USINT) Ritorna il valore di mese dell'anno (Range da 1 a 12)
- Day** (USINT) Ritorna il valore di giorno del mese (Range da 1 a 31)
- WeekDay** (USINT) Ritorna il valore di giorno della settimana (Range da 0 a 6)
0: Domenica, 1:Lunedì, 2:Martedì, 3:Mercoledì, 4:Giovedì, 5:Venerdì, 6:Sabato
- Hour** (USINT) Ritorna il valore di ora (Range da 0 a 23)
- Minute** (USINT) Ritorna il valore di minuti (Range da 0 a 59)
- Second** (USINT) Ritorna il valore di secondi (Range da 0 a 59)
- Done** (BOOL) Attivato al termine della conversione.

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con **SysGetLastError** è possibile rilevare il codice di errore.

- 9986050 Errore allocazione blocco funzione.
- 9986060 Errore versione blocco funzione.

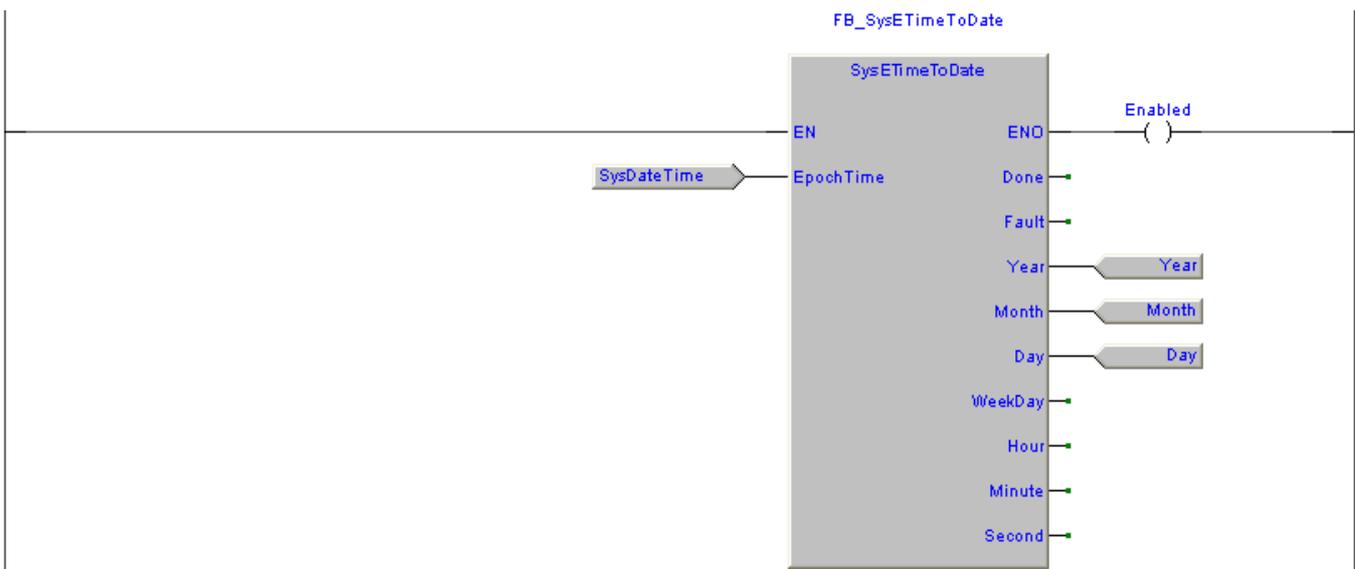
Esempi

Viene convertito il valore di data ed ora espresso in epoch time dalla variabile **SysDateTime** e viene ritornato il valore di anno, mese e giorno nelle tre variabili definite.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FB_SysETimeToDate	SysETimeToDate	Auto	No	0	..	FB SysETimeToDate data
2	Year	UINT	Auto	No	0	..	Year
3	Month	USINT	Auto	No	0	..	Month
4	Day	USINT	Auto	No	0	..	Day

Esempio LD (Ptp116a100)



Esempio IL (Ptp116a100)

```
(* Transfer system date e time to FB input variable. *)
LD SysDateTime
ST FB_SysETimeToDate.EpochTime

CAL FB_SysETimeToDate (* Call the SysETimeToDate function block *)

(* Transfer the FB output variables to program variables. *)

LD FB_SysETimeToDate.Year
ST Year

LD FB_SysETimeToDate.Month
ST Month

LD FB_SysETimeToDate.Day
ST Day
```

Esempio ST (Ptp116a100)

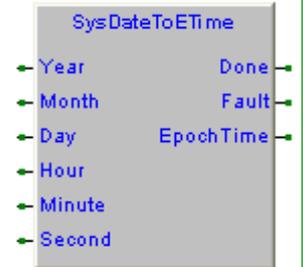
```
(* Here FB SysETimeToDate is executed and variables copied. *)

FB_SysETimeToDate.EpochTime:=SysDateTime;
FB_SysETimeToDate();
Year:=FB_SysETimeToDate.Year; (* Year *)
Month:=FB_SysETimeToDate.Month; (* Month *)
Day:=FB_SysETimeToDate.Day; (* Day *)
```

7.9.2 SysDateToETime, date to epoch time conversion

Type	Library	Version
FB	Embedded	3.0

Questo blocco funzione esegue la conversione della data espressa in epoch time. Occorre fornire al blocco funzione la data espressa nel formato epoch ime come presente nella variabile di sistema **SysDateTime**, in uscita dal blocco funzione avremo i valori di data espressi nel formato Giorno/Mese/Anno ed Ora:Minuti:Secondi.



- Year** (UINT) Definisce il valore di anno (Range da 1970 a 2099)
- Month** (USINT) Definisce il valore di mese dell'anno (Range da 1 a 12)
- Day** (USINT) Definisce il valore di goiorno del mese (Range da 1 a 31)
- Hour** (USINT) Definisce il valore di ora (Range da 0 a 23)
- Minute** (USINT) Definisce il valore di minuti (Range da 0 a 59)
- Second** (USINT) Definisce il valore di secondi (Range da 0 a 59)
- Done** (BOOL) Attivato al termine della conversione.
- Fault** (BOOL) Errore di conversione, viene attivato in caso di errore nella conversione.
- EpochTime** (UDINT) Ritorna data espressa in epoch time.

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con **SysGetLastError** è possibile rilevare il codice di errore.

- 9987050 Errore allocazione blocco funzione.
- 9987060 Errore versione blocco funzione.
- 9987200 Errore durante l'esecuzione del blocco funzione.

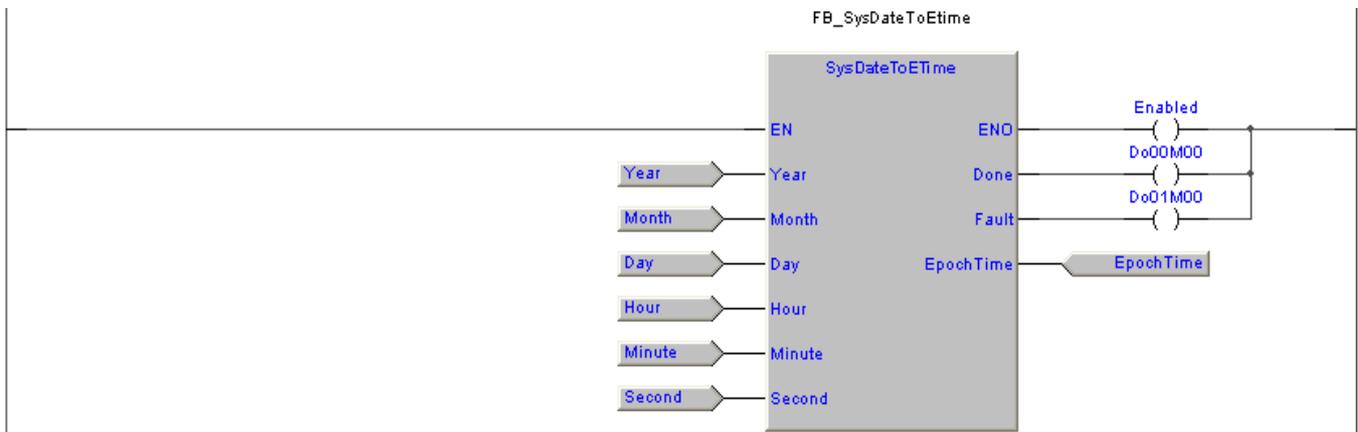
Esempi

Viene convertito il valore di data ed ora in epoch time. Esempio definendo il valore di data 9/4/2010 e ora 14:20:15 avremo in uscita il valore 1270822815.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FB_SysDateToEtime	SysDateToETime	Auto	No	0	..	FB SysDateToETime data
2	Hour	USINT	Auto	No	0	..	Day
3	Minute	USINT	Auto	No	0	..	Month
4	Second	USINT	Auto	No	0	..	Year
5	Day	USINT	Auto	No	0	..	Day
6	Month	USINT	Auto	No	0	..	Month
7	Year	UINT	Auto	No	0	..	Year
8	EpochTime	UDINT	Auto	No	0	..	Epoch time

Esempio LD (Ptp116a100)



Esempio IL (Ptp116a100)

```
(* Transfer date e time to FB input variable. *)

LD Year
ST FB_SysDateToEtime.Year

LD Month
ST FB_SysDateToEtime.Month

LD Day
ST FB_SysDateToEtime.Day

CAL FB_SysDateToEtime (* Call the SysDateToEtime function block *)

(* Transfer the FB output variables to program variables. *)

LD FB_SysDateToEtime.EpochTime
ST EpochTime
```

Esempio ST (Ptp116a100)

```
(* Here FB SysDateToEtime is executed and variables copied. *)

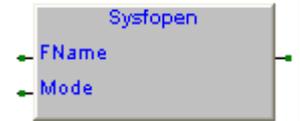
FB_SysDateToEtime.Year:=Year;
FB_SysDateToEtime.Month:=Month;
FB_SysDateToEtime.Day:=Day;
FB_SysDateToEtime(); (* Call the SysDateToEtime function block *)
EpochTime:=FB_SysDateToEtime.EpochTime; (* Epoch time *)
```

7.10 Funzioni ed FB per gestione terminale di I/O

Type	Library	Version
Function	Embedded	3.0

7.10.1 Sysfopen, file open

Questa funzione permette l'apertura del collegamento tra la risorsa indicata dal parametro **FName**, ed un flusso di dati **stream** da impiegare nelle successive chiamate alle funzioni di I/O. La funzione ritorna il pointer alla risorsa.



Se la risorsa indicata è già aperta oppure il nome della risorsa è errato, la funzione ritorna **NULL**. Se si sta aprendo un file su disco, accertarsi che il disco sia formattato.

Parametri funzione:

FName (STRING[20]) E' il nome della risorsa da utilizzare.

Name	Resource
COM0	Serial port COM0
COM1	Serial port COM1
COM2	Serial port COM2
PCOMx.y	Porta seriale y su modulo periferico con indirizzo x
UDPSKT	UDP socket
TCP SKT	TCP socket
pathname	Percorso completo comprensivo del nome file (es.: 'Storage/myFile.txt')

Mode (STRING[4]) Indica il modo in cui la risorsa è aperta: r=read; w=write; a=append. Per le porte seriali definire 'rw'. Per creare un file su disco, occorre eseguire l'apertura in 'w' o 'a'. L'apertura in 'w' su un file esistente, provoca la cancellazione del suo contenuto. L'apertura in 'a', salvaguarda il suo contenuto.

La funzione ritorna:

(FILEP) Pointer alla risorsa.
NULL: In caso di errore.

Codici di errore

In caso di errore la funzione torna con **NULL** e con [SysGetLastError](#) è possibile rilevare il codice di errore.

- 9996100 Nome risorsa **FName** ha lunghezza errata.
- 9996110 Nome risorsa **FName** ha lunghezza errata.
- 9996200~2 Impossibile utilizzare porta da programma utente.

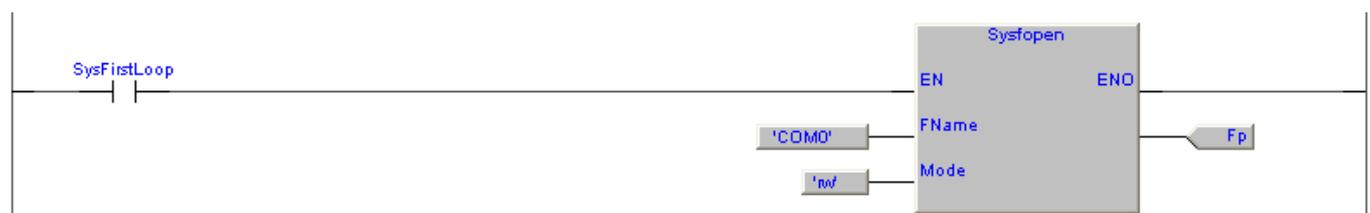
Esempi

Viene aperta la porta seriale in read/write.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer

Esempio LD



Esempio ST (Ptp116a100)

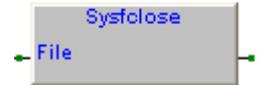
```
IF (Fp = NULL) THEN
```

```
Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)  
END_IF;
```

7.10.2 Sysfclose, file close

Type	Library	Version
Function	Embedded	3.0

Questa funzione permette la chiusura del collegamento alla risorsa indicata dal parametro **File**, precedentemente aperto dalla funzione [Sysfopen](#).



In caso di errore chiusura, la funzione ritorna **EOF**.

Parametri funzione:

File (FILEP) Flusso dati **stream** ritornato dalla funzione **Sysfopen**.

La funzione ritorna:

(INT) **0**: Se esecuzione corretta.
EOF: In caso di errore.

Codici di errore

In caso di errore la funzione torna con **EOF** e con [SysGetLastError](#) è possibile rilevare il codice di errore.

- 9973100 Terminale di I/O usato in task fast o slow.
- 9973200 Errore nella chiusura della risorsa.

Esempi

Viene aperta e successivamente chiusa la porta seriale **COM0**. Se la porta è correttamente aperta viene attivata l'uscita **Do00M00**. Se la porta è correttamente chiusa viene attivata l'uscita **Do01M00**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer

Esempio ST (Ptp116a100)

```
(* Here the COM0 port is opened in read/write. *)

IF (Fp = NULL) THEN
  Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
  Do00M00:=(Fp <> NULL); (* Output is set if port is opened *)
END_IF;

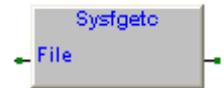
(* Here the COM0 port is closed. *)

IF (Fp <> NULL) THEN
  Do01M00:=(Sysfclose(Fp) <> EOF); (* Output is set if port is closed *)
END_IF;
```

Type	Library	Version
Function	Embedded	3.0

7.10.3 Sysfgetc, get character from file

Questa funzione ritorna un carattere dal flusso dati **stream** indicato dal parametro **File**, precedentemente aperto dalla funzione [Sysfopen](#).



La funzione ritorna il carattere letto dallo stream. In caso di errore o se nessun dato dallo stream, la funzione ritorna **EOF**. Per essere certi che vi siano caratteri dallo stream è possibile utilizzare la funzione [SysGetIChars](#) che ne ritorna il numero.

Parametri funzione:

File (FILEP) Flusso dati **stream** ritornato dalla funzione **Sysfopen**.

La funzione ritorna:

(INT) Carattere letto dal flusso di dati.
EOF: In caso di errore o se nessun dato dallo stream.

Codici di errore

In caso di errore la funzione torna con **EOF** e con [SysGetLastError](#) è possibile rilevare il codice di errore.

9972100 Terminale di I/O usato in task fast o slow.

Esempi

Viene eseguita apertura porta seriale **COM0** e controllato se caratteri disponibili dalla porta. Se almeno un carattere è disponibile ne viene eseguita lettura ed il carattere letto è trasferito nella variabile **Ch**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer
2	Ch	INT	Auto	No	0	..	Character read

Esempio ST (Ptp116a100)

```
(* Here the COM0 port is opened in read/write. *)
IF (Fp = NULL) THEN
  Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
END_IF;

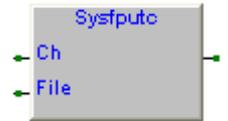
(* Here check if a character is available from port and read it. *)
IF (Fp <> NULL) THEN
  IF (TO_BOOL(SysGetIChars(Fp))) THEN
    Ch:=Sysfgetc(Fp); (* Get input character *)
  END_IF;
END_IF;
```

Type	Library	Version
Function	Embedded	3.0

7.10.4 Sysfputc, put character to file

Questa funzione invia un carattere sul flusso dati **stream** indicato dal parametro **File**, precedentemente aperto dalla funzione [Sysfopen](#).

La funzione ritorna il carattere scritto sullo stream. In caso di errore o se lo stream non accetta il dato, la funzione ritorna EOF. Per essere certi che vi sia spazio sullo stream per accettare il carattere, è possibile utilizzare la funzione [SysGetOSpace](#) che ritorna lo spazio disponibile.



Parametri funzione:

- Ch** (INT) Carattere da inviare sul flusso dati.
- File** (FILEP) Flusso dati **stream** ritornato dalla funzione **Sysfopen**.

La funzione ritorna:

(INT) Carattere scritto sul flusso di dati. **EOF**: In caso di errore o se lo stream non accetta il dato.

Codici di errore

In caso di errore la funzione torna con **EOF** e con [SysGetLastError](#) è possibile rilevare il codice di errore.

9971100 Terminale di I/O usato in task fast o slow.

Esempi

E' riportato un semplice programma che esegue l'eco dei caratteri ricevuti dalla porta seriale **COM0**. Viene eseguita apertura porta seriale **COM0** e controllato se caratteri disponibili dalla porta. Se almeno un carattere è disponibile ne viene eseguita lettura e successiva ritrasmissione.

Definizione variabili (Ptp116a100)

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer
2	Ch	INT	Auto	No	0	..	Character read

Esempio ST

```
(* Here the COM0 port is opened in read/write. *)

IF (Fp = NULL) THEN
    Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
END_IF;

(* Here execute the received characters echo. *)

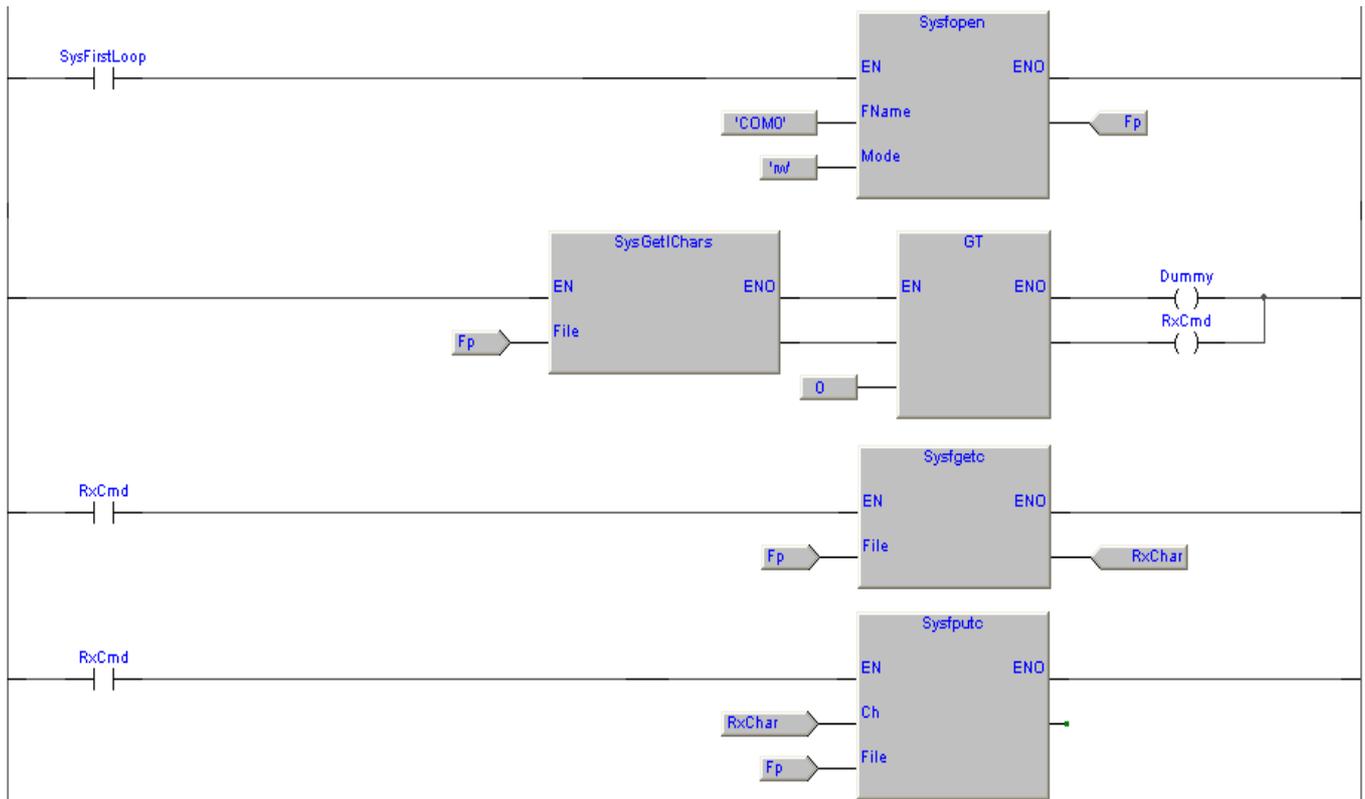
IF (Fp <> NULL) THEN
    IF (TO_BOOL(SysGetIChars(Fp)) AND (TO_BOOL(SysGetOSpace(Fp))) THEN
        Ch:=Sysfgetc(Fp); (* Get input character *)
        Ch:=Sysfputc(Ch, Fp); (* Put input character *)
    END_IF;
END_IF;
```

Utilizzando le funzioni di gestione terminale di I/O è possibile realizzare un semplice programma che esegue l'echo del carattere ricevuto dalla porta seriale **COM0**. La porta viene aperta con il modo impostato di default (115200, e, 8, 1), se è ricevuto un carattere dalla porta si attiva **RxCmd** ed il carattere ricevuto è copiato in **RxChar**. La ricezione di un carattere provoca la trasmissione del carattere ricevuto.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer
2	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable
3	RxChar	INT	Auto	No	0	..	Received char
4	RxCmd	BOOL	Auto	No	FALSE	..	Rx command

Esempio LD (Ptp119a000)



Type	Library	Version
Function	Embedded	3.0

7.10.5 Sysfread, read data from file

Questa funzione esegue la lettura di un numero definito di stringhe di lunghezza definita dal flusso dati **stream** indicato dal parametro **File**, precedentemente aperto dalla funzione [Sysfopen](#).

La funzione ritorna il numero di stringhe dati lette. Se nello **stream** non ci sono abbastanza stringhe da soddisfare i parametri, viene ritornato un numero minore di stringhe lette rispetto al valore definito.



Parametri funzione:

- Buf** (@STRING) Indirizzo della stringa dove trasferire le stringhe lette.
- Size** (INT) Lunghezza in caratteri delle stringhe da leggere.
- Count** (INT) Numero di stringhe da leggere.
- File** (FILEP) Flusso dati **stream** ritornato dalla funzione [Sysfopen](#).

La funzione ritorna:

- (INT) Numero di stringhe lette, se il valore ritornato è minore di **Count**, significa che non vi erano abbastanza dati nello stream.

Codici di errore

In caso di errore la funzione torna con **0** e con [SysGetLastError](#) è possibile rilevare il codice di errore.

9970100 Terminale di I/O usato in task fast o slow.

Esempi

Vengono attesi almeno 5 caratteri ricevuti dalla porta seriale e quando ricevuti viene letta una stringa di 5 caratteri (5 stringhe di 1 carattere), la stringa letta è trasferita nella variabile **RxString**. La stringa letta viene poi ritrasmessa sulla porta seriale, notare come anche nella trasmissione è trasmessa una stringa di 5 caratteri (1 stringa di 5 caratteri).

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer
2	RxString	STRING	Auto	[10]	0	..	Received string
3	RxChars	INT	Auto	No	0	..	Received characters
4	TxChars	INT	Auto	No	0	..	Transmitted characters

Esempio ST

```
(* Here the COM0 port is opened in read/write. *)

IF (Fp = NULL) THEN
    Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
END_IF;

(* Here wait until at least 5 chars are received and echoes them. *)

IF (Fp <> NULL) THEN
    IF (SysGetIChars(Fp) >= 5) THEN
        RxChars:=Sysfread(ADR(RxString), 1, 5, Fp); (* Received characters *)
        TxChars:=Sysfread(ADR(RxString), 5, 1, Fp); (* Received characters *)
    END_IF;
END_IF;
```

Type	Library	Version
Function	Embedded	3.0

7.10.6 Sysfwrite, write data to file

Questa funzione esegue la scrittura di un numero definito di stringhe di lunghezza definita nel flusso dati **stream** indicato dal parametro **File**, precedentemente aperto dalla funzione [Sysfopen](#).

La funzione ritorna il numero di stringhe dati scritte. Se nello stream non c'è abbastanza spazio per contenere il numero di stringhe definito, viene ritornato un numero minore di stringhe scritte rispetto al valore definito.



Parametri funzione:

- Buf** (@STRING) Indirizzo della stringa da scrivere.
- Size** (INT) Lunghezza in caratteri delle stringhe da scrivere.
- Count** (INT) Numero di stringhe da scrivere.
- File** (FILEP) Flusso dati **stream** ritornato dalla funzione [Sysfopen](#).

La funzione ritorna:

- (INT) Numero di stringhe scritte, se valore ritornato minore di **Count**, non vi era abbastanza spazio nello stream.

Codici di errore

In caso di errore la funzione torna con **0** e con [SysGetLastError](#) è possibile rilevare il codice di errore.

9969100 Terminale di I/O usato in task fast o slow.

Esempi

Vengono attesi almeno 5 caratteri ricevuti dalla porta seriale e quando ricevuti viene letta una stringa di 5 caratteri (5 stringhe di 1 carattere), la stringa letta è trasferita nella variabile **RxString**. La stringa letta viene poi ritrasmessa sulla porta seriale, notare come anche nella trasmissione è trasmessa una stringa di 5 caratteri (1 stringa di 5 caratteri).

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer
2	RxString	STRING	Auto	[10]	0	..	Received string
3	RxChars	INT	Auto	No	0	..	Received characters
4	TxChars	INT	Auto	No	0	..	Transmitted characters

Esempio ST

```
(* Here the COM0 port is opened in read/write. *)
IF (Fp = NULL) THEN
  Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
END_IF;

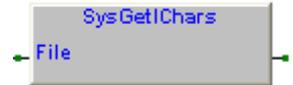
(* Here wait until at least 5 chars are received and echoes them. *)

IF (Fp <> NULL) THEN
  IF (SysGetIChars(Fp) >= 5) THEN
    RxChars:=Sysfread(ADR(RxString), 1, 5, Fp); (* Received characters *)
    TxChars:=Sysfwrite(ADR(RxString), 5, 1, Fp); (* Transmitted characters *)
  END_IF;
END_IF;
```

7.10.7 SysGetIChars, get input available characters from file

Type	Library	Version
Function	Embedded	3.0

Questa funzione ritorna il numero di caratteri disponibili per la lettura dal flusso dati **stream** indicato dal parametro **File**, precedentemente aperto dalla funzione [Sysfopen](#).



Se il valore ritornato è diverso da **0** i caratteri potranno essere letti con la funzione [Sysfgetc](#).

Parametri funzione:

File (FILEP) Flusso dati **stream** ritornato dalla funzione [Sysfopen](#).

La funzione ritorna:

(INT) Numero di caratteri disponibili dal flusso dati.

Codici di errore

In caso di errore la funzione torna con **0** e con [SysGetLastError](#) è possibile rilevare il codice di errore.

9968100 Terminale di I/O usato in task fast o slow.

Esempi

E' riportato un semplice programma che esegue l'eco dei caratteri ricevuti dalla porta seriale **COM0**. Viene eseguita apertura porta seriale **COM0** e controllato se caratteri disponibili dalla porta. Se almeno un carattere è disponibile ne viene eseguita lettura e successiva ritrasmissione.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer
2	Ch	INT	Auto	No	0	..	Character read

Esempio ST

```
(* Here the COM0 port is opened in read/write. *)

IF (Fp = NULL) THEN
    Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
END_IF;

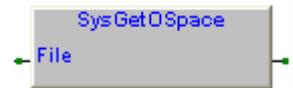
(* Here execute the received characters echo. *)

IF (Fp <> NULL) THEN
    IF (TO_BOOL(SysGetIChars(Fp))) AND (TO_BOOL(SysGetOSpace(Fp))) THEN
        Ch:=Sysfgetc(Fp); (* Get input character *)
        Ch:=Sysfputc(Ch, Fp); (* Put input character *)
    END_IF;
END_IF;
```

7.10.8 SysGetOSpace, get output available space on file

Type	Library	Version
Function	Embedded	3.0

Questa funzione ritorna lo spazio disponibile per la scrittura dati sul flusso dati **stream** indicato dal parametro **File**, precedentemente aperto dalla funzione **Sysfopen**.



Se il valore ritornato è diverso da **0** i caratteri potranno essere scritti con la funzione **Sysfputc**.

Parametri funzione:

File (FILEP) Flusso dati **stream** ritornato dalla funzione **Sysfopen**.

La funzione ritorna:

(INT) Spazio disponibile sul flusso dati per trasferire caratteri.
Se buffer vuoto viene ritornata la dimensione del buffer di trasmissione.

Codici di errore

In caso di errore la funzione torna con **0** e con **SysGetLastError** è possibile rilevare il codice di errore.

9967100 Terminale di I/O usato in task fast o slow.

Esempi

E' riportato un semplice programma che esegue l'eco dei caratteri ricevuti dalla porta seriale **COM0**. Viene eseguita apertura porta seriale **COM0** e controllato se caratteri disponibili dalla porta. Se almeno un carattere è disponibile ne viene eseguita lettura e successiva ritrasmissione.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer
2	Ch	INT	Auto	No	0	..	Character read

Esempio ST

```
(* Here the COM0 port is opened in read/write. *)
IF (Fp = NULL) THEN
  Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
END_IF;

(* Here execute the received characters echo. *)

IF (Fp <> NULL) THEN
  IF (TO_BOOL(SysGetIChars(Fp))) AND (TO_BOOL(SysGetOSpace(Fp))) THEN
    Ch:=Sysfgetc(Fp); (* Get input character *)
    Ch:=Sysfputc(Ch, Fp); (* Put input character *)
  END_IF;
END_IF;
```

7.10.9 SysGetRxBSize, get file Rx input buffer size

Type	Library	Version
Function	Embedded	5.0

Questa funzione ritorna la dimensione del buffer di input (Ricezione) sul flusso dati **stream** indicato dal parametro **File**, precedentemente aperto dalla funzione **Sysfopen**.



Parametri funzione:

File (FILEP) Flusso dati **stream** ritornato dalla funzione **Sysfopen**.

La funzione ritorna:

(UDINT) Dimensione buffer di input espressa in numero di caratteri (Bytes).

Codici di errore

In caso di errore la funzione torna con **0** e con **SysGetLastError** è possibile rilevare il codice di errore.

9966100 Terminale di I/O usato in task fast o slow.

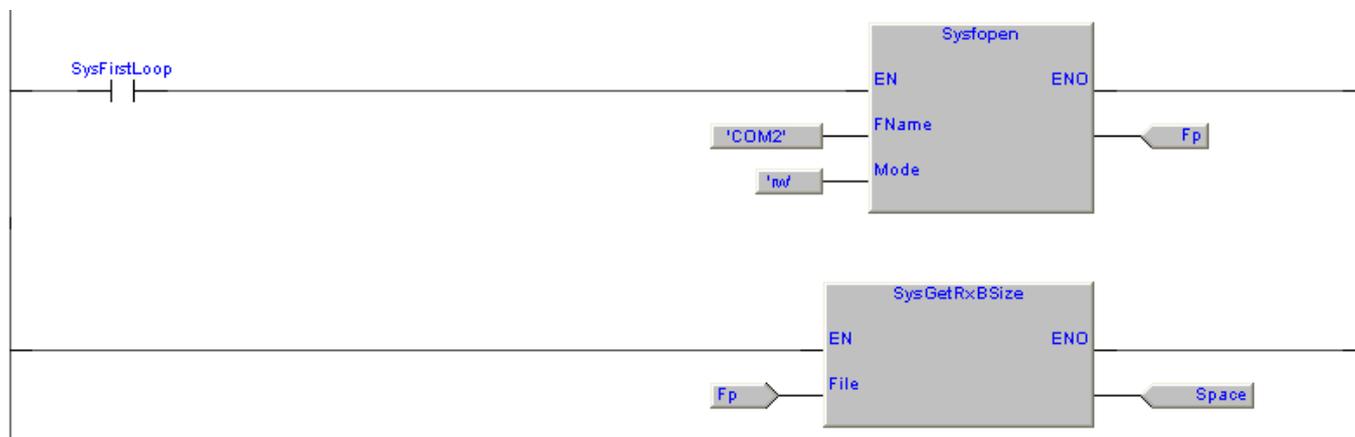
Esempi

E' riportato un semplice programma che ritorna la dimensione del buffer di input (Ricezione) della porta seriale **COM2**. Il valore ritornato espresso in numero di caratteri (Bytes), è trasferito nella variabile **Space**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer
2	Space	UDINT	Auto	No	0	..	Rx buffer space (Nr of Chars)

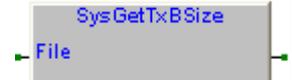
Esempio LD



7.10.10 SysGetTxBSize, get file Tx output buffer size

Type	Library	Version
Function	Embedded	5.0

Questa funzione ritorna la dimensione del buffer di output (Trasmissione) sul flusso dati **stream** indicato dal parametro **File**, precedentemente aperto dalla funzione **Sysfopen**.



Parametri funzione:

File (FILEP) Flusso dati **stream** ritornato dalla funzione **Sysfopen**.

La funzione ritorna:

(UDINT) Dimensione buffer di output espressa in numero di caratteri (Bytes).

Codici di errore

In caso di errore la funzione torna con **0** e con **SysGetLastError** è possibile rilevare il codice di errore.

9965100 Terminale di I/O usato in task fast o slow.

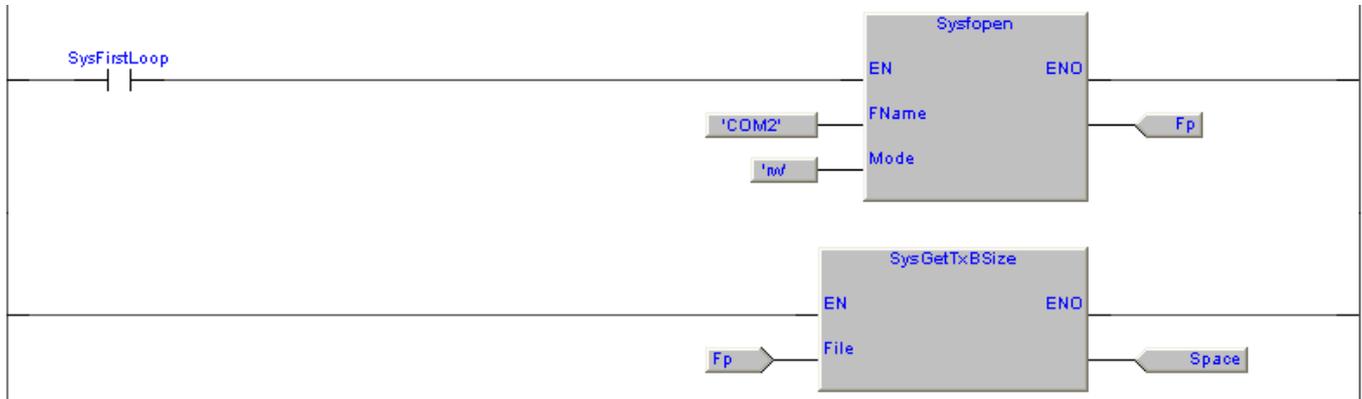
Esempi

E' riportato un semplice programma che ritorna la dimensione del buffer di output (Trasmissione) della porta seriale **COM2**. Il valore ritornato espresso in numero di caratteri (Bytes), è trasferito nella variabile **Space**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer
2	Space	UDINT	Auto	No	0	..	Rx buffer space (Nr of Chars)

Esempio LD



7.10.11 SysFIBfClear, file input buffer clear

Type	Library	Version
Function	Embedded	3.0

Questa funzione elimina tutti i caratteri in lettura presenti sul flusso dati **stream** indicato dal parametro **File**, precedentemente aperto dalla funzione **Sysfopen**.



La funzione ritorna **FALSE** in caso di errore.

Parametri funzione:

File (FILEP) Flusso dati **stream** ritornato dalla funzione **Sysfopen**.

La funzione ritorna:

(BOOL) **FALSE**: Errore esecuzione.
TRUE: Funzione eseguita correttamente.

Codici di errore

In caso di errore la funzione torna **FALSE** e con **SysGetLastError** è possibile rilevare il codice di errore.

9964100 Terminale di I/O usato in task fast o slow.

Esempi

Se è attivo l'ingresso **Di00M00** tutti i caratteri in ingresso dalla porta seriale saranno cancellati e l'uscita **Do00M00** attivata.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer

Esempio ST

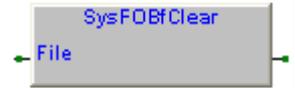
```
(* Here the COM0 port is opened in read/write. *)
IF (Fp = NULL) THEN
    Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
END_IF;

(* If the input is active the input buffer is cleared. *)
IF (Fp <> NULL) THEN
    IF (Di00M00) THEN Do00M00:=SysFIBfClear(Fp); END_IF;
END_IF;
```

Type	Library	Version
Function	Embedded	3.0

7.10.12 SysFOBfClear, file output buffer clear

Questa funzione elimina tutti i caratteri in uscita presenti sul flusso dati **stream** indicato dal parametro **File**, precedentemente aperto dalla funzione **Sysfopen**.



La funzione ritorna **FALSE** in caso di errore.

Parametri funzione:

File (FILEP) Flusso dati **stream** ritornato dalla funzione **Sysfopen**.

La funzione ritorna:

(BOOL) **FALSE**: Errore esecuzione.
TRUE: Funzione eseguita correttamente.

Codici di errore

In caso di errore la funzione torna **FALSE** e con **SysGetLastError** è possibile rilevare il codice di errore.

9963100 Terminale di I/O usato in task fast o slow.

Esempi

Se è attivo l'ingresso **Di00M00** tutti i caratteri in uscita dalla porta seriale saranno cancellati e l'uscita **Do00M00** attivata.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer

Esempio ST

```
(* Here the COM0 port is opened in read/write. *)
IF (Fp = NULL) THEN
    Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
END_IF;

(* If the input is active the ouput buffer is cleared. *)
IF (Fp <> NULL) THEN
    IF (Di00M00) THEN Do00M00:=SysFOBfClear(Fp); END_IF;
END_IF;
```

7.10.13 SysFOBfFlush, file output buffer flush

Type	Library	Version
Function	Embedded	3.0

Questa funzione forza l'uscita immediata dei caratteri presenti sul flusso dati **stream** indicato dal parametro **File**, precedentemente aperto dalla funzione **Sysfopen**, sulla risorsa connessa.



La funzione ritorna **FALSE** in caso di errore.

Parametri funzione:

File (FILEP) Flusso dati **stream** ritornato dalla funzione **Sysfopen**.

La funzione ritorna:

(BOOL) **FALSE**: Errore esecuzione.
TRUE: Funzione eseguita correttamente.

Codici di errore

In caso di errore la funzione torna **FALSE** e con **SysGetLastError** è possibile rilevare il codice di errore.

9962100 Terminale di I/O usato in task fast o slow.

Esempi

Se è attivo l'ingresso **Di00M00** tutti i caratteri presenti nel buffer di uscita della porta seriale saranno trasmessi e l'uscita **Do00M00** attivata.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer

Esempio ST

```
(* Here the COM0 port is opened in read/write. *)
IF (Fp = NULL) THEN
    Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
END_IF;

(* If the input is active the ouput buffer is cleared. *)
IF (Fp <> NULL) THEN
    IF (Di00M00) THEN Do00M00:=SysFOBfFlush(Fp); END_IF;
END_IF;
```

Type	Library	Version
Function	Embedded	3.0

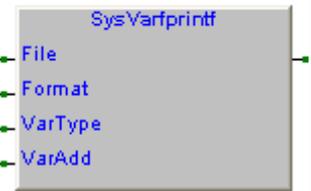
7.10.14 SysVarfprintf, variable print to file

Questa funzione esegue la stampa formattata di una variabile sullo stream collegato al parametro **File**, precedentemente aperto dalla funzione **Sysfopen**.

La stringa **Format** specifica il formato con il quale stampare la variabile. Mentre in **VarType** è indicato il tipo di variabile ed in **VarAdd** il suo indirizzo.

La funzione ritorna il numero di caratteri trasferiti nello stream, **EOF** in caso di errore.

Parametri funzione:



File (FILEP) Flusso dati **stream** ritornato dalla funzione **Sysfopen**.

Format (STRING[80]) Ha due tipi di argomenti, i caratteri ordinari che vengono copiati nello stream di uscita, e le specifiche di conversione, contraddistinte dal simbolo percentuale (%) e da un carattere che specifica il formato con il quale stampare la variabile definita.

VarType (USINT) Tipo variabile, come indicato nella tabella [Variable types definition](#).

VarAdd (UDINT) Indirizzo variabile.

La funzione ritorna:

(INT) Numero di caratteri trasferiti nello stream. **EOF**: Errore esecuzione.

Codici di errore

In caso di errore la funzione torna con **EOF** e con [SysGetLastError](#) è possibile rilevare il codice di errore.

9998010 Valore di **File** non definito.

9998100 Terminale di I/O usato in task fast o slow.

9998200 Tipo variabile non gestito, controllare **VarType**.

Esempi

Su fronte attivazione ingresso **Di00M00** viene incrementata la variabile **Counter** ed il suo valore inviato sulla porta seriale **COM0**. Nella variabile **NrOfChars** viene caricato il numero di caratteri inviati in uscita sulla porta seriale.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer
2	Pulse	BOOL	Auto	No	FALSE	..	Pulse flag
3	Counter	UDINT	Auto	No	0	..	Counter
4	NrOfChars	INT	Auto	No	0	..	

Esempio ST

```
(* Here the COM0 port is opened in read/write. *)
IF (Fp = NULL) THEN Fp:=Sysfopen('COM0', 'rw'); END_IF;
IF (Fp <> NULL) THEN
  IF (Di00M00 <> Pulse) THEN
    Pulse:=Di00M00; (* Pulse flag *)
  IF (Di00M00) THEN
    Counter:=Counter+1; (* Counter *)
    NrOfChars:=SysVarfprintf(Fp, 'Counter:%04d$r$n', UDINT_TYPE, ADR(Counter));
  END_IF;
  END_IF;
END_IF;
```

7.11 File system

Le “CPU SlimLine ARM7” a partire dal firmware versione **SFW167C100**, possono gestire il file system. In tali CPU esistono due directories predefinite:

Storage: Directory allocata sulla memoria EEPROM presente su SlimLine (Tutte le versioni).

SDCard: Directory allocata sul chip SD Card che deve essere inserito nell'apposito connettore.

Per le operazioni di formattazione del file system si rimanda al manuale utente, il file system è raggiungibile da FTP, quindi utilizzando un client FTP è possibile creare nuovi files, cancellare files esistenti, rinominare files esistenti, leggere e scrivere dati nei files.

Type	Library	Version
Function	Embedded	7.0

7.11.1 Sysremove, file remove

Questa funzione esegue la rimozione (cancellazione) di un file. In **Name** occorre definire il nome del file da eliminare specificando l'intero percorso (Esempio Storage/File.txt).



Se l'operazione di rimozione va a buon fine la funzione ritorna **TRUE**, in caso di errore viene ritornato **FALSE**.

Parametri funzione:

Name (STRING[32]) Nome del file da cancellare.

La funzione ritorna:

(BOOL)
FALSE: Errore esecuzione.
TRUE: Ok esecuzione.

Codici di errore

In caso di errore la funzione torna **FALSE** e con [SysGetLastError](#) è possibile rilevare il codice di errore.

9961100 Funzione richiamata in task fast o slow.

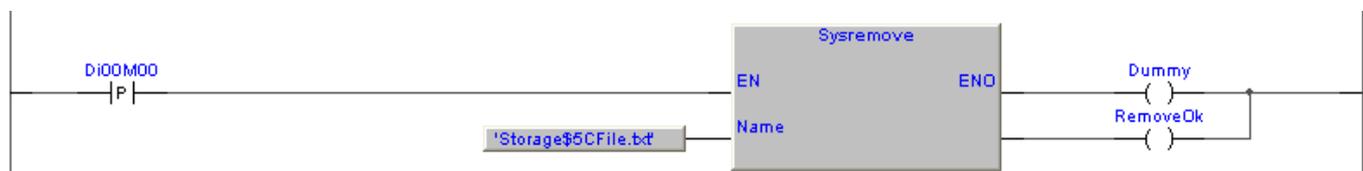
Esempi

Sul fronte di attivazione dell'ingresso digitale **Di00M00** viene eliminato il file **File.txt** presente nella directory **Storage**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	RemoveOk	BOOL	Auto	No	FALSE	..	File remove Ok
2	Dummy	BOOL	Auto	No	FALSE	..	Dummy

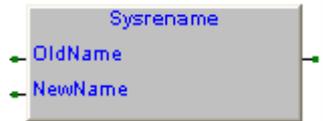
Esempio LD



Type	Library	Version
Function	Embedded	7.0

7.11.2 Sysrename, file rename

Questa funzione esegue la cambiamento del nome di un file. In **OldName** occorre definire il nome del file da rinominare specificando l'intero percorso (Esempio Storage/OldFile.txt), in **NewName** occorre definire il nuovo nome del file specificando l'intero percorso (Esempio Storage/NewFile.txt).



Se l'operazione di rinomina va a buon fine la funzione ritorna **TRUE**, in caso di errore viene ritornato **FALSE**.

Parametri funzione:

OldName (STRING[32]) Nome del file da rinominare.

NewName (STRING[32]) Nuovo nome da dare al file.

La funzione ritorna:

(BOOL) **FALSE**: Errore esecuzione.
TRUE: Ok esecuzione.

Codici di errore

In caso di errore la funzione torna **FALSE** e con [SysGetLastError](#) è possibile rilevare il codice di errore.

9960100 Funzione richiamata in task fast o slow.

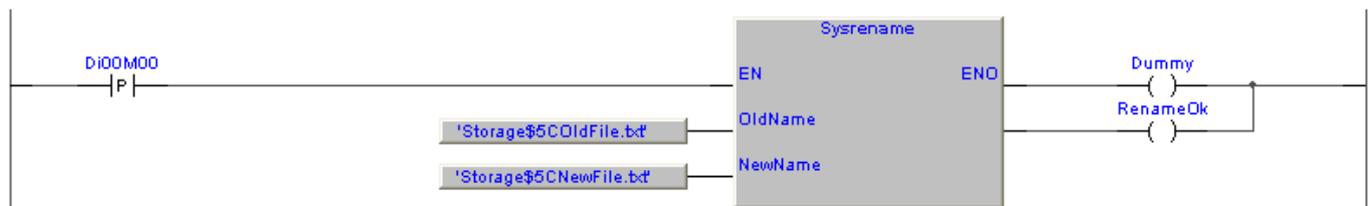
Esempi

Sul fronte di attivazione dell'ingresso digitale **Di00M00** viene rinominato il file **OldFile.txt** presente nella directory **Storage**. Il file assumerà il nuovo nome di **NewFile.txt**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Dummy	BOOL	Auto	No	FALSE	..	Dummy
2	RenameOk	BOOL	Auto	No	FALSE	..	File rename Ok

Esempio LD



Type	Library	Version
Function	Embedded	7.0

7.11.3 Sysfilelength, file length

Questa funzione ritorna la lunghezza in bytes di un file. In **Name** occorre definire il nome del file di cui si vuole conoscere la lunghezza specificando l'intero percorso (Esempio Storage/File.txt).



Se il file indicato non è presente, la funzione ritorna **-1**.

Parametri funzione:

Name (STRING[32]) Nome del file di cui si vuole conoscere la lunghezza.

La funzione ritorna:

(DINT) Lunghezza file (Bytes). **EOF** se file non presente.

Codici di errore

In caso di errore la funzione torna **EOF** e con [SysGetLastError](#) è possibile rilevare il codice di errore.

9959100 Funzione richiamata in task fast o slow.

Esempi

Sul fronte di attivazione dell'ingresso digitale **Di00M00** viene ritornata la lunghezza del file **File.txt** presente nella directory **Storage**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FileSize	DINT	Auto	No	0	..	File size

Esempio LD



7.11.4 Sysfseek, file seek

Type	Library	Version
Function	Embedded	7.0

Questa funzione permette di cambiare l'indicatore di posizione dello stream collegato al parametro **File**, precedentemente aperto dalla funzione **Sysfopen**.

Offset specifica il numero di bytes dall'origine dove andrebbe posizionato l'indicatore di posizione.
Origin specifica la posizione di origine rispetto alla quale spostare l'indicatore di posizione.



La funzione ritorna il valore attuale dell'indicatore di posizione. In caso di errore di posizionamento, l'indicatore di posizione rimane inalterato e la funzione ritorna **EOF**.

Parametri funzione:

- File** (FILEP) Flusso dati **stream** ritornato dalla funzione **Sysfopen**.
- Offset** (DINT) Numero di bytes dall'origine dove posizionare l'indicatore di posizione
- Origin** (INT) Occorre specificare la posizione di origine secondo la tabella sottoriportata.

Value	Mode	Acquisition
0	ID_SEEK_SET	Inizio del file
1	ID_SEEK_CUR	Posizione corrente file
2	ID_SEEK_END	Fine del file

La funzione ritorna:

- (DINT) Valore attuale dell'indicatore di posizione. **EOF** se errore.

Codici di errore

In caso di errore la funzione torna **EOF** e con **SysGetLastError** è possibile rilevare il codice di errore.

9958100 Funzione richiamata in task fast o slow.

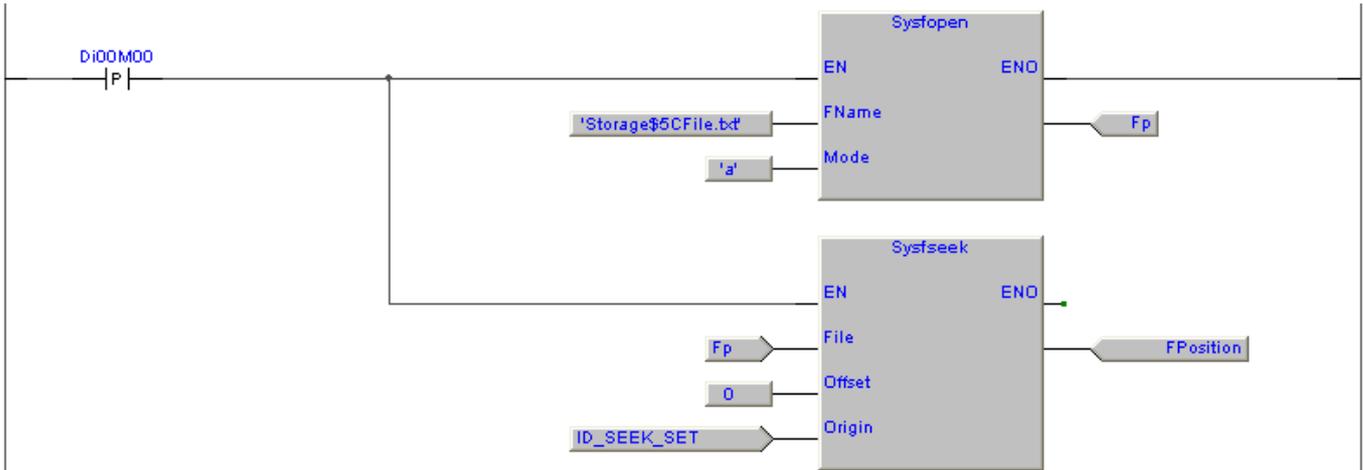
Esempi

Sul fronte di attivazione dell'ingresso **Di00M00** viene posizionato l'indicatore di posizione all'inizio del file **File.txt** presente nella directory **Storage**.

Definizione variabili

	Name	Type	Address	Array	Init value	filelength	Attribute	Description
1	Fp	FILEP	Auto	No	0	..		File pointer
2	FPosition	DINT	Auto	No	0	..		File position

Esempio LD



7.11.4.1 Scrittura file di log

Ecco un semplice programma che scrive nel file **Log.txt** presente nella directory **Storage**, un record di log ad ogni secondo. Il record è scritto nel formato CSV e quindi facilmente importabile in Excel. Vengono scritti 50 records di log consecutivi, terminato il ciclo di scrittura si riprende dall'inizio del file sovrascrivendo i records già scritti. Il file di log creato sarà del tipo:

```
00;15;00;15;00000015;
01;15;00;16;00000005;
02;15;00;17;00000004;
03;15;00;18;00000004;
04;15;00;19;00000005;
```

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Pulse	BOOL	Auto	No	FALSE	..	Time base pulse
2	NrOfLog	USINT	Auto	No	0	..	Number of log stored
3	i	INT	Auto	No	0	..	Auxiliary counter
4	LogRecord	STRING	Auto	[32]		..	Log record
5	Fp	FILEP	Auto	No	0	..	File pointer
6	DateTime	SysETimeToDate	Auto	No	0	..	Date/Time conversion

Esempio ST

(* Check if a second is passed. *)

```
IF (SysClock1000 = Pulse) THEN RETURN; END_IF;
Pulse:=SysClock1000; (* Time base pulse *)
```

(* Open the file in "append" mode. *)

```
Fp:=Sysfopen('Storage/Log.txt', 'a'); (* File pointer *)
IF (Fp = NULL) THEN RETURN; END_IF;
```

(* Create the log record. *)

```
DateTime(EpochTime:=SysDateTime); (* Date time conversion *)
i:=SysVarsnprintf(ADR(LogRecord), 4, '%02d;', USINT_TYPE, ADR(NrOfLog));
i:=SysVarsnprintf(ADR(LogRecord[3]), 4, '%02d;', USINT_TYPE, ADR(DateTime.Hour));
i:=SysVarsnprintf(ADR(LogRecord[6]), 4, '%02d;', USINT_TYPE, ADR(DateTime.Minute));
i:=SysVarsnprintf(ADR(LogRecord[9]), 4, '%02d;', USINT_TYPE, ADR(DateTime.Second));
i:=SysVarsnprintf(ADR(LogRecord[12]), 12, '%08ld;$r$n', UDINT_TYPE, ADR(SysTBackExTm));
```

(* Set the file pointer on right position. *)

```
i:=Sysfseek(Fp, TO_DINT(NrOfLog*23), ID_SEEK_SET);
NrOfLog:=NrOfLog+1; (* Number of log stored *)
IF (NrOfLog >= 50) THEN NrOfLog:=0; END_IF;
```

```
i:=Sysfwrite(ADR(LogRecord), 23, 1, Fp); (* Write to file *)
i:=Sysfclose(Fp); (* Close file *)
```

7.12 Funzioni ed FB per gestione porta seriale

I sistemi SlimLine dispongono di più porte seriali in base alla versione del prodotto. Le porte seriali vengono identificate con un nome del tipo **COMx** dove al posto della **x** vi è il numero di porta seriale.

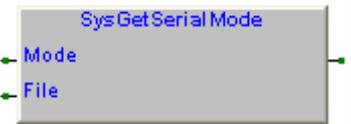
Per la corrispondenza tra il numero di porta ed il connettore fisico ad essa connesso, si rimanda al manuale hardware del prodotto.

Type	Library	Version
Function	Embedded	3.0

7.12.1 SysGetSerialMode, get serial mode

Questa funzione ritorna il modo di comunicazione impostato sulla porta seriale connessa al parametro **File**, precedentemente aperto dalla funzione **Sysfopen**.

Nel parametro **Mode** occorre definire l'indirizzo della struttura **SYSSERIALMODE** in cui dovrà essere trasferito il modo seriale attualmente impostato sulla porta. La funzione ritorna **FALSE** in caso di errore.



Parametri funzione:

Mode (@SYSSERIALMODE) Indirizzo struttura **SYSSERIALMODE** in cui trasferire il modo letto.

File (FILEP) Flusso dati **stream** ritornato dalla funzione **Sysfopen**.

La funzione ritorna:

(BOOL) **FALSE**: Errore esecuzione. **TRUE**: Funzione eseguita correttamente.

Codici di errore

In caso di errore la funzione torna **FALSE** e con **SysGetLastError** è possibile rilevare il codice di errore.

9995010 Valore di **File** non definito.

9995020 Indirizzo struttura **SYSSERIALMODE** non corretto, verificare **Mode**.

9995100 ÷ 1 Errore esecuzione funzione.

Esempi

Su fronte attivazione ingresso **Di00M00** viene salvato il modo impostato sulla porta seriale **COM0** nella variabile **Sm** e viene attivata l'uscita **Do00M00**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer
2	Sm	SYSSERIALMODE	Auto	No	0	..	Serial mode data struct
3	Pulse	BOOL	Auto	No	FALSE	..	Pulse flag

Esempio ST

```
(* Here the COM0 port is opened in read/write. *)

IF (Fp = NULL) THEN
  Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
END_IF;

(* Check if the COM0 port is open. *)

IF (Fp <> NULL) THEN

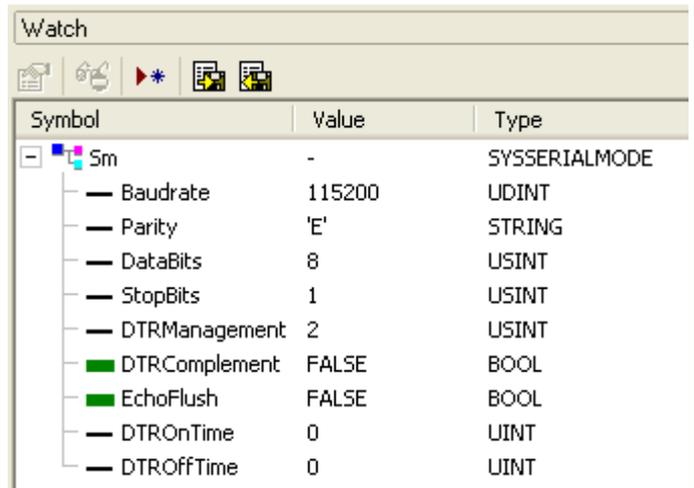
  (* Check if input is activated. *)

  IF (Di00M00 <> Pulse) THEN
    Pulse:=Di00M00; (* Pulse flag *)

    (* On input raising edge the serial mode is read. *)

    IF (Di00M00) THEN
      Do00M00:=SysGetSerialMode(ADR(Sm), Fp);
    END_IF;
  END_IF;
END_IF;
```

Mettendo in watch la variabile **Sm** di tipo **SYSSERIALMODE** possiamo vedere i valori di tutti i suoi membri come riportato nella figura a lato. In questo caso è visualizzata la configurazione di default **115200, e, 8, 1**.



Symbol	Value	Type
Sm	-	SYSSERIALMODE
Baudrate	115200	UDINT
Parity	'E'	STRING
DataBits	8	USINT
StopBits	1	USINT
DTRManagement	2	USINT
DTRComplement	FALSE	BOOL
EchoFlush	FALSE	BOOL
DTROnTime	0	UINT
DTROffTime	0	UINT

Type	Library	Version
Function	Embedded	3.0

7.12.2 SysSetSerialMode, set serial mode

Questa funzione imposta il modo di comunicazione definito nella struttura **SYSSERIALMODE** sulla porta seriale connessa al parametro **File**, precedentemente aperto dalla funzione **Sysfopen**.



Nel parametro **Mode** occorre definire l'indirizzo della struttura **SYSSERIALMODE** in cui è definito il modo seriale da impostare sulla porta. La funzione ritorna **FALSE** in caso di errore.

Parametri funzione:

- Mode** (@SYSSERIALMODE) Indirizzo struttura **SYSSERIALMODE** con il modo da impostare.
- File** (FILEP) Flusso dati **stream** ritornato dalla funzione **Sysfopen**.

La funzione ritorna:

(BOOL) **FALSE**: Errore esecuzione. **TRUE**: Funzione eseguita correttamente.

Codici di errore

In caso di errore la funzione torna **FALSE** e con **SysGetLastError** è possibile rilevare il codice di errore.

- 9994010 Valore di **File** non definito.
- 9994020 Indirizzo struttura **SYSSERIALMODE** non corretto, verificare **Mode**.
- 9994050 Errore valore di baud rate.
- 9994051 Errore valore di parità.
- 9994052 Errore valore bit di dato.
- 9994053 Errore valore bit di stop.
- 9995100 Errore esecuzione funzione.

Esempi

Su fronte attivazione ingresso **Di00M00** viene salvato il modo impostato sulla porta seriale **COM0** nella variabile **Sm** e viene attivata l'uscita **Do00M00**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer
2	Sm	SYSSERIALMODE	Auto	No	0	..	Serial mode data struct
3	Pulse	BOOL	Auto	No	FALSE	..	Pulse flag

Esempio ST

```
(* On input raising edge the serial mode is changed. *)

IF (Di00M00 <> Pulse) THEN
    Pulse:=Di00M00; (* Pulse flag *)

    IF (Di00M00) THEN
        IF (Fp = NULL) THEN Fp:=Sysfopen('COM0', 'rw'); END_IF;
        Do00M00:=SysGetSerialMode(ADR(Sm), Fp);
        Sm.Baudrate:=19200;
        Sm.Parity:='N';
        Sm.DTRManagement:=DTR_AUTO_WO_TIMES;
        Do01M00:=SysSetSerialMode(ADR(Sm), Fp);
    END_IF;
END_IF;
```

Type	Library	Version
FB	PLCUtyLib	SFR054A000

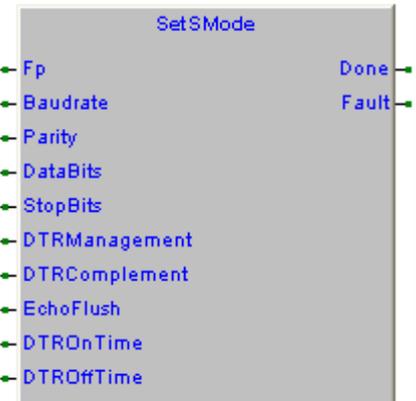
7.12.3 SetSMode, Set serial mode

Questo blocco funzione esegue impostazione parametri di comunicazione porta seriale. Occorre passare al blocco funzione il puntatore al file di porta seriale su cui eseguire l'impostazione dei parametri di comunicazione così come viene ritornato dalla funzione di apertura file [Sysfopen](#). Per ulteriori chiarimenti sui vari parametri fare riferimento alla definizione modo comunicazione porta seriale [SYSSERIALMODE](#).

In caso di errore esecuzione (Esempio definizione di un parametro fuori range) il valore di impostazione della porta seriale non viene modificato e si attiverà l'uscita **Fault**.

Attivando il modo di funzionamento **DTR_AUTO_W_TIMES** su **DTRManagement** è possibile con il parametro **DTROnTime**, definire un tempo di attesa dopo attivazione segnale DTR prima della trasmissione dei dati su seriale. Con il parametro **DTROffTime** è possibile definire un tempo di attesa dopo la trasmissione dell'ultimo dato prima della disattivazione del segnale DTR.

Queste temporizzazioni sulla gestione del segnale DTR permettono di gestire in modo automatico apparecchiature radiomodem, su cui deve essere attivato il comando di trasmissione prima di eseguire l'invio dei dati.



- Fp** (FILEP) Puntatore al file di porta seriale come ritornato da funzione [Sysfopen](#).
- Baudrate** (UDINT) Valore di baud rate porta seriale (da 300 a 115200 baud)
- Parity** (STRING[1]) Tipo di parità, valori possibili "E" pari, "O" dispari, "N" nessuna.
- DataBits** (USINT) Numero di bit frame dato, valori possibili 7, 8.
- StopBits** (USINT) Numero di bit di stop, valori possibili 1, 2.
- DTRManagement** (USINT) Modo di gestione del segnale DTR sulla porta seriale, [vedi definizione](#).
- DTRComplement** (BOOL) **FALSE**: DTR normale, **TRUE**: DTR complementato.
- EchoFlush** (BOOL) **FALSE**: I dati trasmessi sono ritornati in ricezione. **TRUE**: I dati trasmessi sono ignorati.
- DTROnTime** (UINT) Tempo di attesa dopo attivazione segnale DTR prima di trasmissione caratteri (mS).
- DTROffTime** (UINT) Tempo di attesa dopo trasmissione ultimo dato prima e disattivazione segnale DTR (mS).
- Done** (BOOL) Impostazione modo eseguito correttamente.
- Fault** (BOOL) Errore su esecuzione blocco funzione

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con [SysGetLastError](#) è possibile rilevare il codice di errore.

- 10006010 Lettura modo corrente in errore la funzione [SysGetSerialMode](#) è ritornata con errore probabilmente non è stato indicato correttamente il file pointer **Fp**.
- 10006020 Impostazione modo in errore la funzione [SysSetSerialMode](#) è ritornata con errore probabilmente uno dei parametri definiti non è nel range corretto.

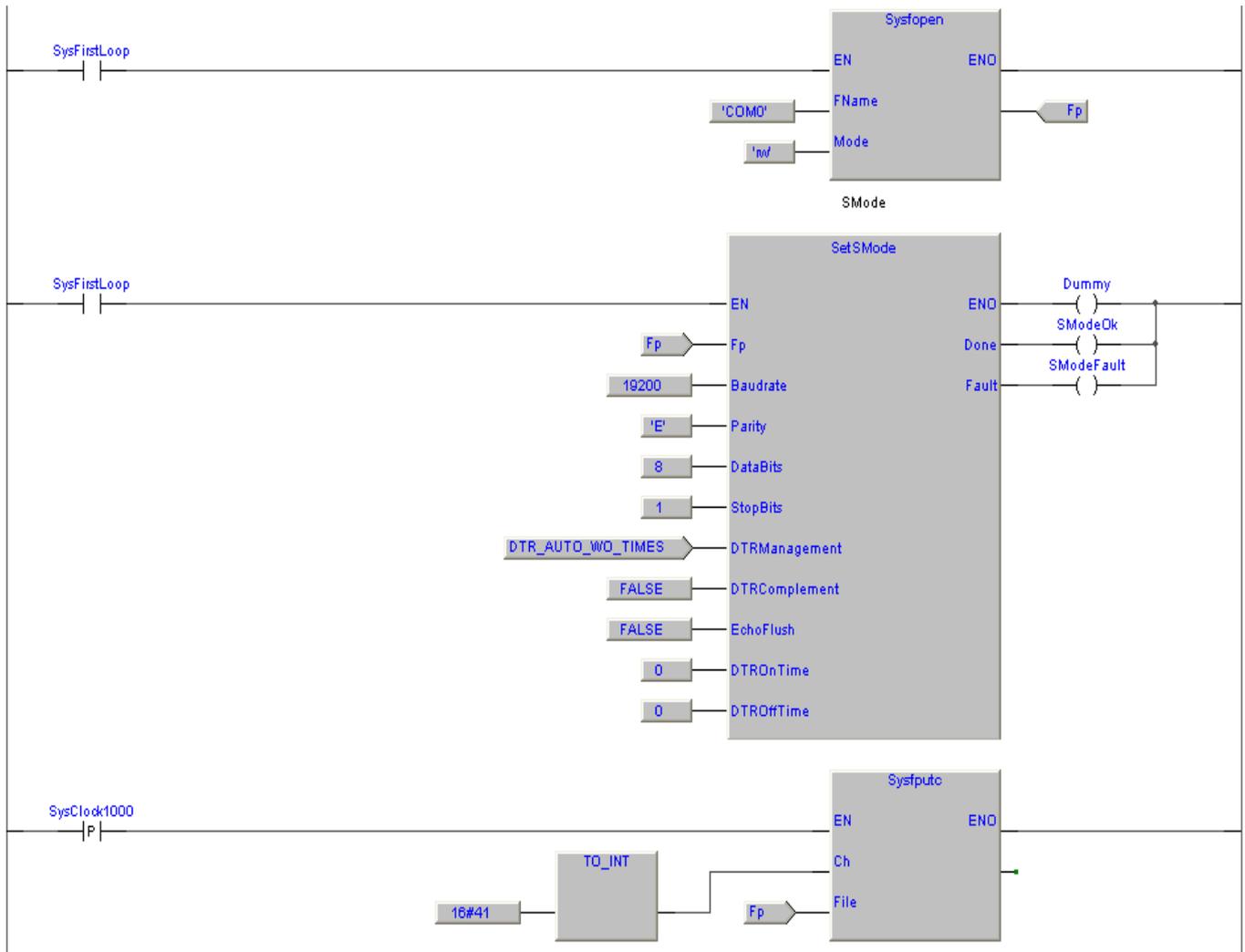
Esempi

Al primo loop di esecuzione programma viene aperta la porta seriale **COM0**, viene impostato il modo di comunicazione 19200, e, 8, 1. Ogni secondo viene inviato sulla porta seriale il carattere 'A' codice ascii 16#41.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer
2	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable
3	SMode	SetSMode	Auto	No	0	..	Set serial mode
4	SModeOk	BOOL	Auto	No	FALSE	..	Set serial mode ok
5	SModeFault	BOOL	Auto	No	FALSE	..	Set serial mode fault

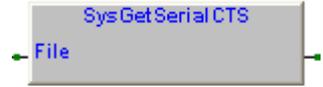
Esempio LD (Ptp119a000)



7.12.4 SysGetSerialCTS, get serial CTS signal status

Type	Library	Version
Function	Embedded	5.0

Questa funzione ritorna lo stato del segnale **CTS** della porta seriale connessa al parametro **File**, precedentemente aperto dalla funzione **Sysfopen**.



Parametri funzione:

File (FILEP) Flusso dati **stream** ritornato dalla funzione **Sysfopen**.

La funzione ritorna:

(BOOL) **FALSE**: Segnale CTS non attivo. **TRUE**: Segnale CTS attivo.

Codici di errore

In caso di errore la funzione torna **FALSE** e con **SysGetLastError** è possibile rilevare il codice di errore.

9993010 Valore di **File** non definito.

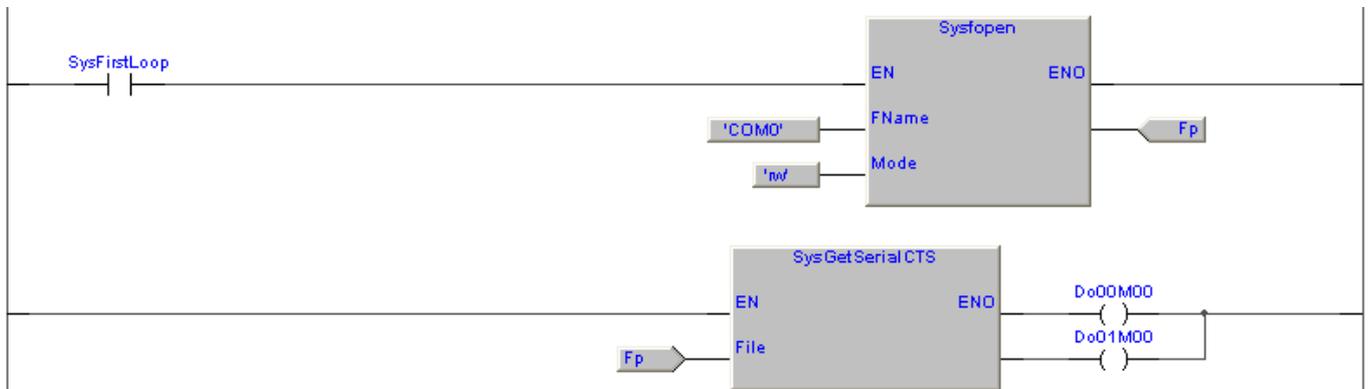
Esempi

Lo stato del segnale **CTS** della porta seriale **COM0** è appoggiato sull'uscita **Do01M00**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer

Esempio LD



7.12.5 SysSetSerialDTR, set DTR signal status

Type	Library	Version
Function	Embedded	5.0

Questa funzione imposta lo stato del segnale **DTR** della porta seriale connessa al parametro **File**, precedentemente aperto dalla funzione **Sysfopen**.

Per poter gestire il segnale DTR occorre avere definito sulla porta seriale il valore **DTR_OFF** nella variabile **DTRManagement** nella struttura **SYSSERIALMODE**.



Parametri funzione:

- Status** (BOOL) Stato segnale DTR su porta seriale
- File** (FILEP) Flusso dati **stream** ritornato dalla funzione **Sysfopen**.

La funzione ritorna:

(BOOL) **FALSE**: Errore esecuzione. **TRUE**: Ok esecuzione.

Codici di errore

In caso di errore la funzione torna **FALSE** e con **SysGetLastError** è possibile rilevare il codice di errore.

9992010 Valore di **File** non definito.

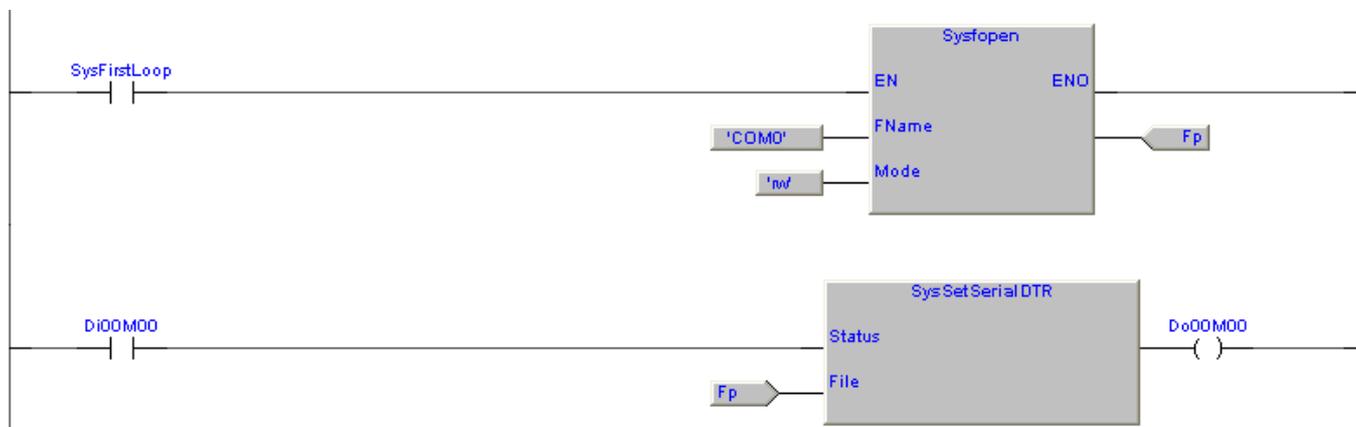
Esempi

Su lo stato dell'ingresso **Di00M00** viene trasferito sul segnale DTR della porta seriale **COM0**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer

Esempio LD



7.13 Funzioni ed FB per gestione CAN bus

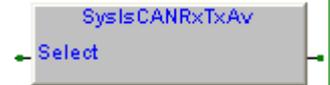
7.13.1 SysIsCANRxTxAv, checks if CAN Rx or Tx is available

Type	Library	Version
Function	Embedded	3.0

Questa funzione controlla:

Select:=FALSE: Se vi è almeno un messaggio nel buffer di ricezione CAN.

Select:=TRUE: Se vi è spazio per un messaggio nel buffer di trasmissione CAN.



La funzione ritorna TRUE se la condizione selezionata è vera.

Parametri funzione:

Select (BOOL) **FALSE:** Se vi è almeno un messaggio nel buffer di ricezione CAN.

TRUE: Se vi è spazio per un messaggio nel buffer di trasmissione CAN.

La funzione ritorna:

(BOOL) **TRUE:** Condizione selezionata è vera.

Esempi

E' riportato un semplice programma che esegue il controllo se un messaggio CAN è stato ricevuto, ne esegue la ricezione ed invia in uscita sulla porta seriale **COM0** la struttura del messaggio ricevuto.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer
2	CANMsg	SYSCANMESSAGE	Auto	No	0	..	CAN message
3	NrOfChars	INT	Auto	No	0	..	Number of printed chars

Esempio ST

```
(* Here the COM0 port is opened in read/write. *)
IF (Fp = NULL) THEN
    Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
END_IF;

(* Here check if a CAN message is available and receive it. *)

IF (SysIsCANRxTxAv(FALSE)) THEN
    IF (SysCANRxMsg(16#00000000, 16#00000000, ADR(CANMsg))) THEN

        NrOfChars:=SysVarfprintf(Fp, 'Length:%04d$r$n', USINT_TYPE, ADR(CANMsg.Length));
        NrOfChars:=SysVarfprintf(Fp, 'MsgID:%04d$r$n', UDINT_TYPE, ADR(CANMsg.MsgID));
        NrOfChars:=SysVarfprintf(Fp, 'Data[0]:%02X$r$n', UDINT_TYPE, ADR(CANMsg.Data[0]));

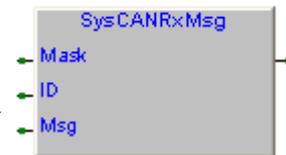
    END_IF;
END_IF;
```

7.13.2 SysCANRxMsg, receives a CAN message

Type	Library	Version
Function	Embedded	3.0

Questa funzione riceve un messaggio CAN e lo trasferisce nella variabile il cui indirizzo è definito in **Msg**. E' possibile definire un **Mask** ed un **ID** per ricevere i soli messaggi CAN desiderati.

La funzione ricerca nello stack dei messaggi un messaggio il cui ID posto in AND con **Mask** coincide con **ID** messo in AND con **Mask**. La funzione ritorna TRUE se messaggio ricevuto.



Parametri funzione:

- Mask** (UDINT) Codice maschera ID messaggio.
- ID** (UDINT) ID check ID messaggio.
- Msg** (@SYSCANMESSAGE) Indirizzo buffer messaggio ricevuto.

La funzione ritorna:

(BOOL) **TRUE**: Messaggio ricevuto.

Esempi

E' riportato un semplice programma che esegue la ricezione di qualsiasi messaggio CAN ed esegue l'invio in uscita sulla porta seriale **COM0** della struttura del messaggio ricevuto.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer
2	CANMsg	SYSCANMESSAGE	Auto	No	0	..	CAN message
3	NrOfChars	INT	Auto	No	0	..	Number of printed chars

Esempio ST

```
(* Here the COM0 port is opened in read/write. *)

IF (Fp = NULL) THEN
    Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
END_IF;

(* Here receive a CAN message. *) *)

IF (SysCANRxMsg(16#3FFFFFFF, 16#00000000, ADR(CANMsg))) THEN

    NrOfChars:=SysVarfprintf(Fp, 'Length:%04d$r$n', USINT_TYPE, ADR(CANMsg.Length));
    NrOfChars:=SysVarfprintf(Fp, 'MsgID:%04d$r$n', UDINT_TYPE, ADR(CANMsg.MsgID));
    NrOfChars:=SysVarfprintf(Fp, 'Data[0]:%02X$r$n', UDINT_TYPE, ADR(CANMsg.Data[0]));

END_IF;
```

Type	Library	Version
Function	Embedded	3.0

7.13.3 SysCANTxMsg, transmit a CAN message

Questa funzione trasmette un messaggio CAN, occorre creare il messaggio e poi passarne alla funzione l'indirizzo in **Msg**.



La funzione ritorna TRUE se messaggio trasmesso.

Parametri funzione:

Msg (@SYSCANMESSAGE) Indirizzo buffer messaggio da trasmettere.

La funzione ritorna:

(BOOL) **TRUE:** Messaggio trasmesso.

Esempi

E' riportato un semplice programma che esegue la trasmissione di un messaggio CAN.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	CANMsg	SYSCANMESSAGE	Auto	No	0	..	CAN message
2	TxOk	BOOL	Auto	No	FALSE	..	Transmission Ok

Esempio ST

```
(* Here check if there is a space in Tx buffer and send a CAN message. *)

IF (SysIsCANRxTxAv(TRUE)) THEN
    CANMsg.RmReq:=FALSE; (* eFALSE:Data frame, eTRUE:Remote request *)
    CANMsg.Length:=2; (* Data length *)
    CANMsg.MsgID:=16#00000000; (* Message ID (FF:Bit 31) (11 or 29 Bit) *)
    CANMsg.Data[0]:=16#01; (* Message data *)
    CANMsg.Data[1]:=16#00; (* Message data *)
    TxOk:=SysCANTxMsg(ADR(CANMsg)); (* Transmission Ok *)
END_IF;
```

7.14 Funzioni ed FB per gestione stringhe

7.14.1 eLEN, string length

Type	Library	Version
Function	ePLCStdLib	SFR053A200

Questa funzione ritorna la lunghezza (Espressa in numero di caratteri) della stringa definita in **Str**.



Parametri funzione:

Str (@USINT) Pointer alla stringa di cui calcolare lunghezza.

La funzione ritorna:

(INT) Numero di caratteri della stringa.

Esempi

Viene calcolata la lunghezza della stringa **'Hello!'** ed il numero di caratteri che compongono la stringa viene trasferito nella variabile **Length**. Il risultato del calcolo è **6**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Length	INT	Auto	No	0	..	String length

Esempio LD



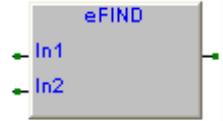
7.14.2 eFIND, string find

Type	Library	Version
Function	ePLCStdLib	SFR053A200

Questa funzione cerca la posizione del carattere di inizio della prima apparizione di **In2** in **In1**. Se nessuna apparizione viene trovata, la funzione ritorna **0**.

Se la stringa **In2** è trovata nella stringa **In1**, viene ritornata la posizione dove si trova.

Esempio: **eFIND(In1:='abcd', In2:='bc')**. Ha come risultato **2**.



Parametri funzione:

In1 (@USINT) Pointer alla stringa dove effettuare la ricerca.

In1 (@USINT) Pointer alla stringa da ricercare.

La funzione ritorna:

(INT) Posizione dove la stringa **In2** è stata trovata. **0** se stringa non trovata.

Esempi

Viene eseguita la ricerca della stringa **'lo'** nella stringa **'Hello world!'**. La posizione trovata è 4 e viene trasferita nella variabile **Position**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Position	INT	Auto	No	0	..	StrToFind position
2	StrSource	STRING	Auto	[32]		..	String were looking
3	StrToFind	STRING	Auto	[32]		..	String to find

Esempio ST

```
(* Find the position where is StrToFind in StrSource. *)

StrSource:='Hello world!';
StrToFind:='lo';

Position:=eFIND(ADR(StrSource), ADR(StrToFind));
```

7.14.3 MemSet, memory set

Type	Library	Version
Function	ePLCAuxLib	SFR058A000

Questa funzione trasferisce in **Buffer** il valore definito in **Value** per il numero di bytes definito in **Size**.

La funzione ritorna il numero definito in **Size**.



Parametri funzione:

- Buffer** (@USINT) Pointer al buffer di memoria dove trasferire **Value**.
- Value** (USINT) Valore da trasferire nel buffer di memoria.
- Size** (UINT) Numero di volte in cui **Value** è trasferito in **Buffer**.

La funzione ritorna:

- (UINT) Valore definito in **Size**.

Esempi

Viene eseguito l'azzeramento di tutti i bytes della stringa **StrBuffer** scrivendo il valore **0** in tutta la sua lunghezza.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	StrBuffer	STRING	Auto	[32]		..	String buffer
2	RetVal	UINT	Auto	No	0	..	Return value

Esempio ST (Ptp116a100)

```
(* The 'StrBuffer' variable is set to '0'. *)
RetVal:=MemSet(ADR(StrBuffer), 0, 32); (* Return value *)
```

Type	Library	Version
Function	Embedded	4.0

7.14.4 SysVarsnprintf, variable print to string

Questa funzione trasferisce in **String** la stampa formattata di una variabile. Il valore stampato ritornato nella variabile stringa non può superare la lunghezza definita in **Size**.

La stringa **Format** specifica il formato con il quale stampare la variabile. Mentre in **VarType** è indicato il tipo di variabile ed in **VarAdd** il suo indirizzo.

La funzione ritorna il numero di caratteri trasferiti nella variabile **String**. EOF in caso di errore.



Parametri funzione:

- String** (@USINT) Pointer all'array dove deve essere trasferito il risultato della stampa.
- Size** (UINT) Numero di caratteri da trasferire nella variabile **String**. Il numero definito è comprensivo del codice di fine stringa '\0'. Se la lunghezza della stringa di output supera il limite di **Size** byte, viene troncata al numero di byte indicato.
- Format** (STRING[80]) Ha due tipi di argomenti, i caratteri ordinari che vengono copiati nella variabile **String** di uscita, e le specifiche di conversione, contraddistinte dal simbolo percentuale (%) e da un carattere che specifica il formato con il quale stampare la variabile definita.
- VarType** (USINT) Tipo variabile, come indicato nella tabella [Variable types definition](#).
- VarAdd** (UDINT) Indirizzo variabile.

La funzione ritorna:

- (INT) Numero di caratteri comprensivo codice fine stringa '\0' trasferiti in variabile **String**.
- EOF**: Errore esecuzione.

Codici di errore

In caso di errore la funzione torna con **EOF** e con [SysGetLastError](#) è possibile rilevare il codice di errore.

- 9997100 Tipo variabile non gestito, controllare **VarType**.
- 9997200 Il valore di **Size** limita la formattazione della stringa in uscita.

Esempi

Su fronte attivazione ingresso **Di00M00** viene incrementata la variabile **Counter** e la stampa del suo valore trasferita nell'array **StringOut**. Il valore presente in **StringOut** viene poi inviato sulla porta seriale **COM0**. Nella variabile **NrOfChars** viene caricato il numero di caratteri trasferiti in **StringOut** ed inviati in uscita sulla porta seriale.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer
2	Pulse	BOOL	Auto	No	FALSE	..	Pulse flag
3	Counter	UDINT	Auto	No	0	..	Counter
4	NrOfChars	INT	Auto	No	0	..	Number of printed chars
5	StringOut	USINT	Auto	[0..31]	32(0)	..	String output
6	i	INT	Auto	No	0	..	Auxiliary counter
7	Ch	INT	Auto	No	0	..	Character written

Esempio ST (Ptp116a100)

```
(* Here the COM0 port is opened in read/write. *)

IF (Fp = NULL) THEN
    Fp:=SysFopen('COM0', 'rw'); (* Port COM0 file pointer *)
END_IF;

(* Check if the COM0 port is open. *)

IF (Fp <> NULL) THEN
```

```
(* Check if input is activated. *)
IF (Di00M00 <> Pulse) THEN
    Pulse:=Di00M00; (* Pulse flag *)

    (* On input raising edge the counter value is printed. *)

    IF (Di00M00) THEN
        Counter:=Counter+1; (* Counter *)
        NrOfChars:=SysVarsnprintf(ADR(StringOut), 32, 'Counter:%04d$r$n', UDINT_TYPE, ADR(Counter));

        (* Copy the printed result to serial port. *)

        FOR i:=0 TO NrOfChars DO Ch:=Sysfputc(TO_INT(StringOut[i]), Fp); END_FOR;
    END_IF;
END_IF;
END_IF;
```

In questo esempio viene eseguito il merge tra le stampe del valore di due variabili. Eeguire il merge può essere molto utile per avere un'unica stringa contenente la stampa del valore di più variabili.

Mettendo in debug la variabile **Result** vedremo la stringa **Var[0]:12 Var[1]:34**. Avendo bloccato la stampa a 12 caratteri il valore di **Var[0]** sarà stampato correttamente fino ad un massimo di 4 cifre (7 caratteri stringa, 4 caratteri valore più terminatore stringa '\0'). Per valori di **Var[0]** superiori a 9999 non saranno più stampate le cifre meno significative.

Il valore di **Var[1]** sarà stampato immediatamente dopo il valore di **Var[0]**. Da notare l'offset decrementato di 1 per sovrascrivere il terminatore stringa '\0'. Per garantire la stampa di un massimo di 4 anche per **Var[1]** è stato definito un **Size** di 13, la stringa di testo inizia infatti con un carattere di space per separarla dalla stampa del valore della variabile precedente.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	NrOfChars	INT	Auto	No	0	..	Number of printed chars
2	Var	UDINT	Auto	[0..1]	12,34	..	Variables
3	Result	STRING	Auto	[32]		..	

Esempio ST

```
(* ----- *)
(* EXECUTE A VARIABLES PRINT MERGE *)
(* ----- *)
(* Print the variable values, merging them into a single string. *)

NrOfChars:=SysVarsnprintf(ADR(Result), 12, 'Var[0]:%d', UDINT_TYPE, ADR(Var[0]));
NrOfChars:=SysVarsnprintf(ADR(Result[NrOfChars-1]), 13, '$20Var[1]:%d', UDINT_TYPE, ADR(Var[1]));

(* [End of file] *)
```

Type	Library	Version
Function	Embedded	4.0

7.14.5 SysVarsscanf, extracts values from string

Questa funzione legge la stringa **String** e ne interpreta il contenuto basandosi sul parametro **Format**.

La stringa **Format** specifica il formato con il quale interpretare la variabile, in **VarType** è indicato il tipo di variabile ed in **VarAdd** il suo indirizzo.

La funzione ritorna TRUE se valore variabile trovato, in caso contrario FALSE.

Parametri funzione:

String (@USINT) Pointer alla stringa da leggere.

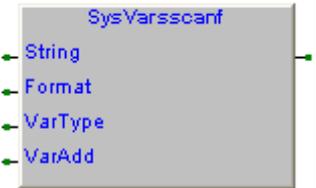
Format (STRING[80]) Ha due tipi di argomenti, i caratteri ordinari che vengono copiati nella variabile **String** di uscita, e le specifiche di conversione, contraddistinte dal simbolo percentuale (%) e da un carattere che specifica il formato con il quale stampare la variabile definita.

VarType (USINT) Tipo variabile, come indicato nella tabella [Variable types definition](#).

VarAdd (UDINT) Indirizzo variabile.

La funzione ritorna:

(BOOL) **TRUE:** Valore variabile acquisito.



Codici di errore

In caso di errore la funzione torna con **FALSE** e con [SysGetLastError](#) è possibile rilevare il codice di errore.

9999100 Tipo variabile non gestito, controllare **VarType**.

Esempi

E' riportato un programma che esegue la lettura di una stringa **InputString** valorizzata con il testo **Value:123**. Su fronte attivazione ingresso **Di00M00** sono eseguite tre diverse funzioni **SysVarsscanf** tutte sulla stringa **InputString** ma con diverse definizioni di **Format**. Le prime due hanno esito positivo e le variabili **Variable[0]** e **Variable[1]** saranno valorizzate con il valore **123**. La terza avrà esito negativo, la variabile **Variable[2]** sarà azzerata.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Pulse	BOOL	Auto	No	FALSE	..	Pulse flag
2	Variable	UDINT	Auto	[0..2]	3(0)	..	Variable
3	InputString	STRING	Auto	[32]	Value:123	..	Input string
4	Result	BOOL	Auto	[0..2]	3(0)	..	Function result

Esempio ST (Ptp116a200)

```
(* Check if input is activated. *)

IF (Di00M00 <> Pulse) THEN
    Pulse:=Di00M00; (* Pulse flag *)

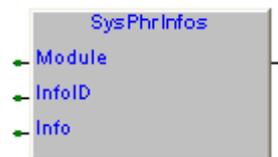
    IF (Di00M00) THEN
        Result[0]:=SysVarsscanf(ADR(InputString), 'Value:%d', UDINT_TYPE, ADR(Variable[0]));
        Result[1]:=SysVarsscanf(ADR(InputString)+6, '%d', UDINT_TYPE, ADR(Variable[1]));
        Result[2]:=SysVarsscanf(ADR(InputString), '%d', UDINT_TYPE, ADR(Variable[2]));
    END_IF;
END_IF;
```

7.15 Funzioni ed FB per gestione moduli periferici

7.15.1 SysPhrInfos, get infos from peripheral modules

Type	Library	Version
Funzione	Embedded	5.0

Questa funzione esegue l'acquisizione di informazioni dai moduli periferici. Viene trasferita nella variabile stringa il cui indirizzo è passato in **Info**, l'informazione indicata da **InfoID** relativa al modulo indicato in **Module**.



La funzione ritorna **TRUE** se correttamente eseguita, in caso contrario **FALSE**.

Parametri funzione:

Module (USINT) Occorre specificare l'indirizzo di modulo da cui eseguire l'acquisizione (Range da 0 a 15). Il valore 0 indica il primo modulo di estensione, 1 il secondo e così di seguito.

InfoID (USINT) Occorre specificare l'ID della informazione richiesta.

Value	Acquisition
0	Ritorna codice prodotto
1	Ritorna codice programma

Info (STRING[10]) Indirizzo variabile dove trasferire l'informazione.

La funzione ritorna:

(BOOL) **FALSE:** Errore esecuzione. **TRUE:** Funzione eseguita correttamente.

Codici di errore

In caso di errore la funzione torna con **FALSE** e con [SysGetLastError](#) è possibile rilevare il codice di errore.

9990100 Il modulo indirizzato in **Module** non è presente.

9990110 Il valore di **InfoID** non è corretto.

9990200 L'informazione richiesta non è supportata dal modulo.

9990210 Errore durante la richiesta della informazione dal modulo.

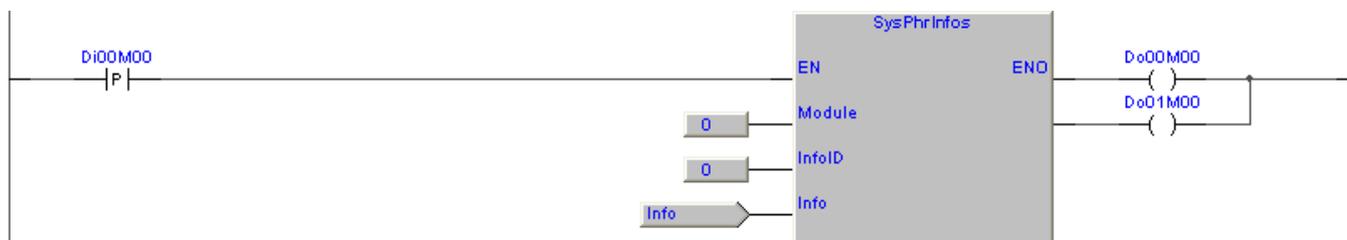
Esempi

E' riportato un programma che esegue la lettura del codice prodotto dal modulo di estensione con indirizzo 0. Il codice ritornato è trasferito nella variabile **Info**. Il ritorno della funzione è trasferito sull'uscita **Do01M00**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Info	STRING	Auto	[10]		..	Info returned from module

Esempio LD



Type	Library	Version
FB	Embedded	3.0

7.15.2 SysGetPhrDI, get peripheral digital input

Questo blocco funzione esegue l'acquisizione degli ingressi digitali dai moduli periferici. Il blocco funzione ritorna lo stato degli ingressi digitali dal modulo indicato in **Address** in base al comando di **Mode** definito.

Per acquisire gli ingressi digitali presenti sul modulo CPU occorre definire **Address** 255 e **Mode** DI_8_LL. Si raccomanda di utilizzare in alternativa il blocco funzione [CPUModuleIO](#).



Address (USINT) Occorre specificare l'indirizzo di modulo da cui eseguire l'acquisizione degli ingressi digitali (Range da 0 a 255). Il valore 0 indica il primo modulo di estensione, 1 il secondo e così di seguito. L'indirizzo 255 indica il modulo CPU.

Mode (USINT) Occorre specificare il modo di acquisizione ingressi digitali secondo la tabella sottoriportata.

Value	Mode	Acquisition
1	DI_8_LL	Read 0-7 input mode
2	DI_8_L	Read 8-15 input mode
3	DI_8_M	Read 16-23 input mode
4	DI_8_MM	Read 24-31 input mode
5	DI_16_L	Read 0-15 input mode
6	DI_16_M	Read 16-31 input mode
7	DI_32	Read 0-31 input mode

Done (BOOL) Dato acquisito, viene attivato se acquisizione ingressi digitali terminata.

Fault (BOOL) Errore di acquisizione, viene attivato in caso di errore nella sequenza di acquisizione.

Value (UDINT) Ritorna lo stato degli ingressi digitali acquisiti.

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con [SysGetLastError](#) è possibile rilevare il codice di errore.

9985050 Errore allocazione blocco funzione.

9985070 Errore versione blocco funzione.

9985100 Il modulo indirizzato in **Address** non è presente.

9985110~8 Il modo acquisizione definito in **Mode** non è corretto.

9985200~6 Errore durante l'esecuzione della lettura ingressi dal modulo periferico.

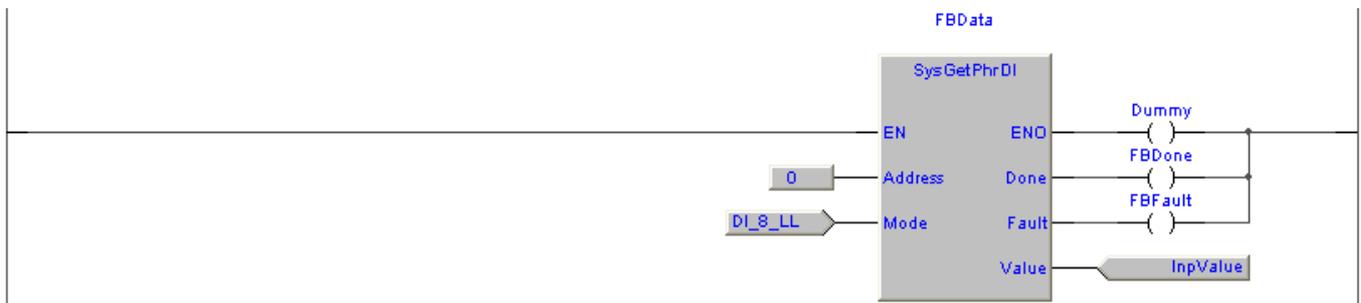
Esempi

Viene acquisito lo stato degli 8 ingressi bassi (Da Inp 0 a Inp 7) del modulo di con indirizzo 0. Se dato valido viene attivata la variabile **FBDone** se errore di acquisizione viene attivata la variabile **FBFault**. Il valore acquisito nel range da 0x00 a 0xFF è trasferito nella variabile **InpValue**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBDData	SysGetPhrDI	Auto	No	0	..	FB SysGetPhrDI data
2	InpValue	UDINT	Auto	No	0	..	Digital input value
3	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable
4	FBDone	BOOL	Auto	No	FALSE	..	FB done
5	FBFault	BOOL	Auto	No	FALSE	..	FB fault

Esempio LD (Ptp116a100)



Esempio IL (Ptp116a100)

(* Read Inp 0 to 7 from module with address 0. *)

```
LD 0
ST FBDData.Address
```

```
LD DI_8_LL
ST FBDData.Mode
```

```
CAL FBDData
```

```
LD FBDData.Value
ST InpValue
```

Esempio ST (Ptp116a100)

(* Read Inp 0 to 7 from module with address 0. *)

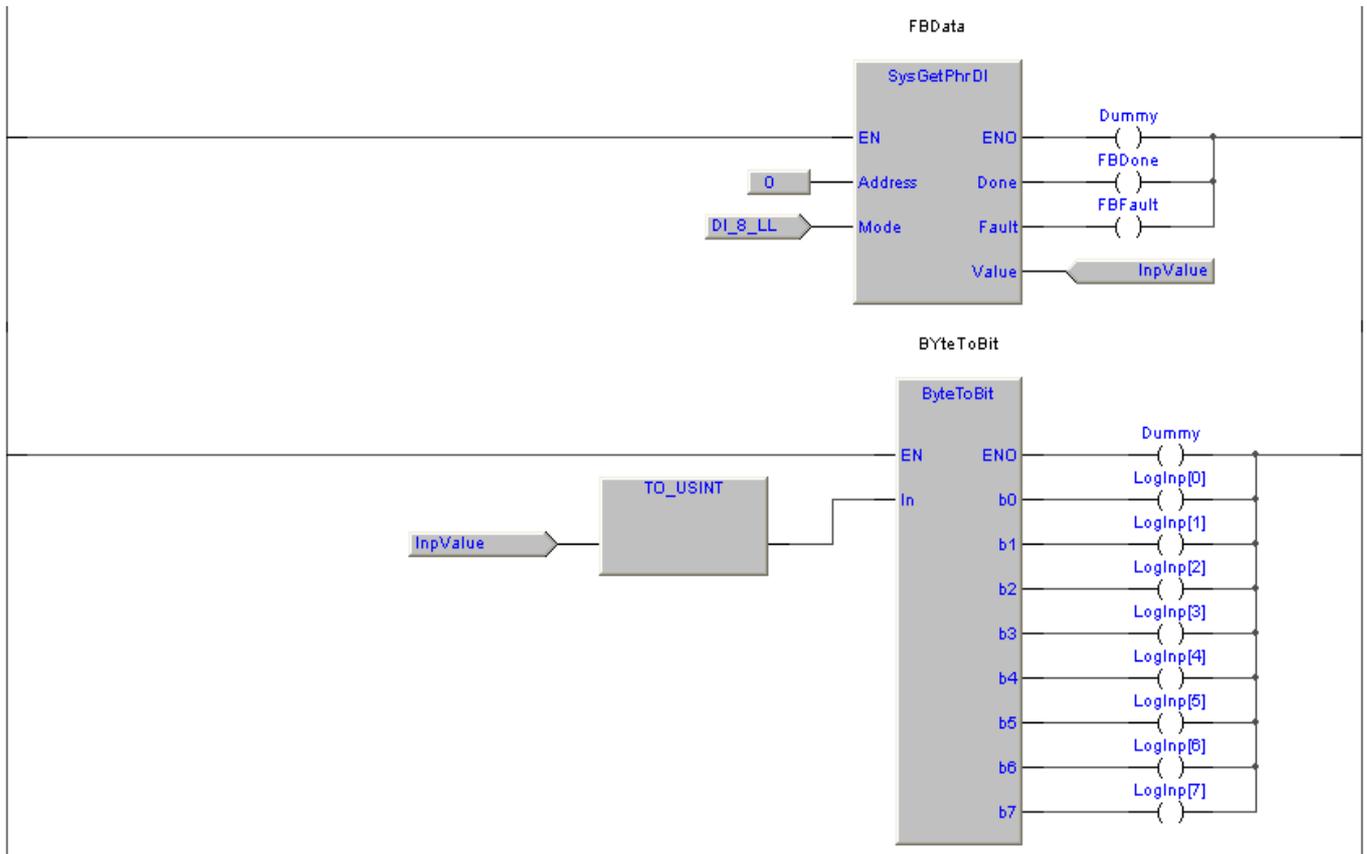
```
FBDData.Address:=0;
FBDData.Mode:=DI_8_LL;
FBDData(); (* Execute FB *)
InpValue:=FBDData.Value; (* Digital input value *)
```

Questo esempio è una evoluzione dell'esempio precedente. Utilizzando il blocco funzione **ByteToBit** lo stato degli 8 ingressi bassi (Da Inp 0 a Inp 7) del modulo con indirizzo 0, è appoggiato su di un array di BOOL.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBDData	SysGetPhrDI	Auto	No	0	..	FB SysGetPhrDI data
2	InpValue	UDINT	Auto	No	0	..	Digital input value
3	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable
4	FBDone	BOOL	Auto	No	FALSE	..	FB done
5	FBFault	BOOL	Auto	No	FALSE	..	FB fault
6	BYteToBit	ByteToBit	Auto	No	0	..	FB ByteToBit
7	LogInp	BOOL	Auto	[0..7]	8(0)	..	Logic inputs

Esempio LD (Ptp119a000)



Type	Library	Version
FB	Embedded	3.0

7.15.3 SysSetPhrDO, set peripheral digital output

Questo blocco funzione esegue impostazione uscite digitali sui moduli periferici indirizzati in base al comando di **Mode** definito.

Per gestire le uscite digitali presenti sul modulo CPU occorre definire **Address** 255 e **Mode** DO_8_LL. Si raccomanda di utilizzare in alternativa il blocco funzione [CPUModuleIO](#).



Address (USINT) Occorre specificare l'indirizzo del modulo su cui eseguire l'impostazione delle uscite digitali (Range da 0 a 255). Il valore 0 indica il primo modulo di estensione, 1 il secondo e così di seguito. L'indirizzo 255 indica il modulo CPU.

Mode (USINT) Occorre specificare il modo di gestione uscite digitali secondo la tabella sottoriportata.

Mode	Define	Acquisition
1	DO_8_LL	Write 0-7 output mode
2	DO_8_L	Write 8-15 output mode
3	DO_8_M	Write 16-23 output mode
4	DO_8_MM	Write 24-31 output mode
5	DO_16_L	Write 0-15 output mode
6	DO_16_M	Write 16-31 output mode
7	DO_32	Write 0-31 output mode

Value (UDINT) Impostare il valore da trasferire sulle uscite digitali.

Done (BOOL) Dato acquisito, viene attivato per un loop al termine della acquisizione ingressi digitali.

Fault (BOOL) Errore di acquisizione, viene attivato in caso di errore nella sequenza di acquisizione.

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con [SysGetLastError](#) è possibile rilevare il codice di errore.

9984050 Errore allocazione blocco funzione.

9984060 Terminato spazio memoria rilocabile, non è possibile eseguire l'FB.

9984070 Errore versione blocco funzione.

9984100 Il modulo indirizzato in **Address** non è presente.

9984110~8 Il modo gestione definito in **Mode** non è corretto.

9984200~6 Errore durante l'esecuzione gestione uscite digitali sul modulo periferico.

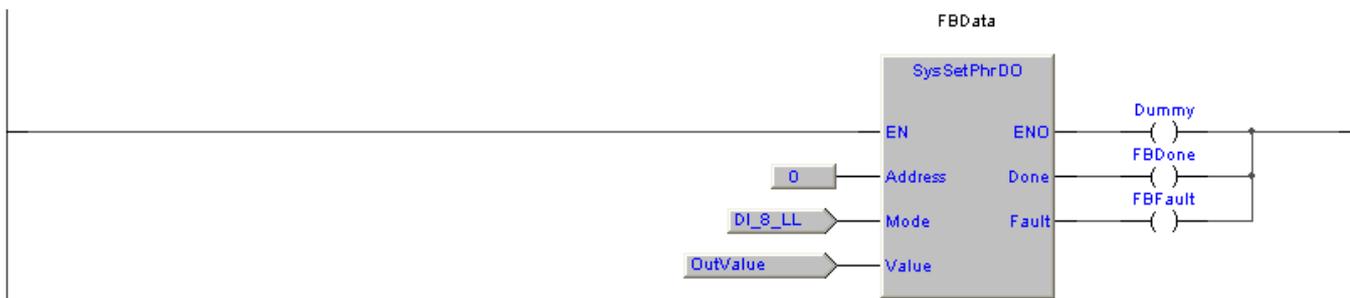
Esempi

Viene trasferito il valore della variabile **OutValue** sulle 8 uscite basse (Da Out 0 a Out 7) del modulo con indirizzo 0.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBData	SysSetPhrDO	Auto	No	0	..	FB SysSetPhrDO data
2	OutValue	UDINT	Auto	No	0	..	Digital output value
3	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable
4	FBDone	BOOL	Auto	No	FALSE	..	FB done
5	FBFault	BOOL	Auto	No	FALSE	..	FB fault

Esempio LD (Ptp116a100)



Esempio IL (Ptp116a100)

(* Manage digital outputs Out 0 to Out 7 on module with address 0. *)

```
LD 0
ST FBData.Address

LD DO_8_LL
ST FBData.Mode

LD OutValue
LD FBData.Value

CAL FBData
```

Esempio ST (Ptp116a100)

(* Manage digital outputs Out 0 to Out 7 on module with address 0. *)

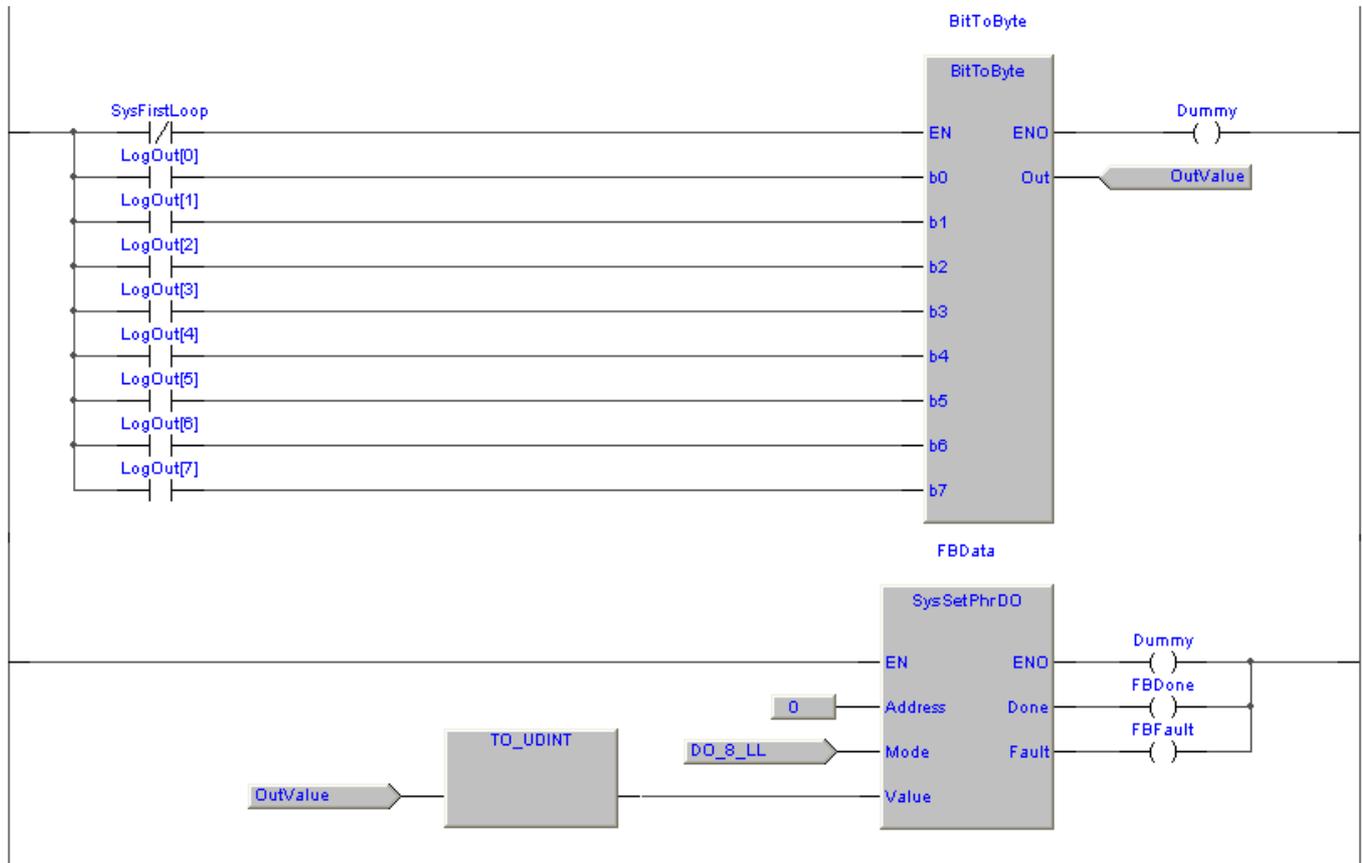
```
FBData.Address:=0;
FBData.Mode:=DO_8_LL;
FBData.Value:=OutValue; (* Digital output value *)
FBData(); (* Execute FB *)
```

Questo esempio è una evoluzione dell'esempio precedente. Utilizzando il blocco funzione **BitToByte** un array di 8 BOOL è trasferito in uscita sulle 8 uscite basse (Da Out 0 a Out 7) del modulo con indirizzo 0.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBDData	SysSetPhrDO	Auto	No	0	..	FB SysSetPhrDO data
2	OutValue	UDINT	Auto	No	0	..	Digital output value
3	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable
4	FBDone	BOOL	Auto	No	FALSE	..	FB done
5	FBFault	BOOL	Auto	No	FALSE	..	FB fault
6	BitToByte	BitToByte	Auto	No	0	..	FB BitToByte
7	LogOut	BOOL	Auto	[0..7]	8(0)	..	Logic inputs

Esempio LD (Ptp119a000)



Type	Library	Version
FB	Embedded	3.0

7.15.4 SysGetAnInp, get analog input

Questo blocco funzione esegue l'acquisizione dell'ingresso analogico dal modulo di acquisizione. Il blocco funzione gestisce vari modi di acquisizione in funzione del modulo analogico a cui fa riferimento.



Address (USINT) Occorre specificare l'indirizzo del modulo da cui eseguire l'acquisizione analogica (Range da 0x00 a 0x0F). Il valore 0x00 indica il primo modulo di estensione, 0x01 il secondo e così di seguito.

Channel (USINT) Occorre specificare l'indirizzo del canale sul modulo (Range da 0x00 a 0x0F). Se viene settato un indirizzo di canale non presente, si interrompe l'esecuzione e viene settato il bit di **Fault**.

Mode (USINT) Occorre specificare il modo di acquisizione analogica secondo la tabella sottoriportata.

Mode	Define	Acquisition
0	AD_IDLE	Idle mode
1	AD_VOLT_0_125_COMMON	Voltage from 0 to 1.25 V (Common mode)
2	AD_VOLT_0_10_COMMON	Voltage from 0 to 10 V (Common mode)
3	AD_CURR_0_20_COMMON	Current from 0 to 20 mA (Common mode)
4	AD_CURR_4_20_COMMON	Current from 4 to 20 mA (Common mode)
5	AD_VOLT_0_1_COMMON	Voltage from 0 to 1 V (Common mode)
7	AD_VOLT_0_1_DIFFER	Voltage from 0 to 1 V (Differential mode)
8	AD_VOLT_0_125_DIFFER	Voltage from 0 to 1.25 V (Differential mode)
9	AD_VOLT_0_10_DIFFER	Voltage from 0 to 10 V (Differential mode)
10	AD_PT100_DIFFER	Pt100 sensor (Differential mode)
11	AD_PT1000_DIFFER	Pt1000 sensor (Differential mode)
12	AD_NI1000_DIFFER	Ni1000 sensor (Differential mode)
32	AD_PT100_DIN_43760	Pt100 DIN_43760 standard
33	AD_PT100_AMERICAN	Pt100 American standard
34	AD_PT100_ITS_90	Pt100 ITS-90 standard
35	AD_PT100_IEC_60751	Pt100 IEC-60751 standard
40	AD_PT1000_DIN_43760	Pt1000 DIN_43760 standard
41	AD_PT1000_AMERICAN	Pt1000 American standard
42	AD_PT1000_ITS_90	Pt1000 ITS-90 standard
43	AD_PT1000_IEC_60751	Pt1000 IEC-60751 standard
48	AD_NI1000_DIN_43760	Ni1000 DIN_43760 standard
64	AD_THERMOCOUPLE_B	Thermocouple B type
65	AD_THERMOCOUPLE_E	Thermocouple E type
66	AD_THERMOCOUPLE_J	Thermocouple J type
67	AD_THERMOCOUPLE_K	Thermocouple K type
68	AD_THERMOCOUPLE_N	Thermocouple N type
69	AD_THERMOCOUPLE_R	Thermocouple R type
70	AD_THERMOCOUPLE_S	Thermocouple S type
71	AD_THERMOCOUPLE_T	Thermocouple T type

Done (BOOL) Dato analogico acquisito, viene attivato al termine della acquisizione analogica.

Fault (BOOL) Errore di acquisizione, viene attivato in caso di errore nella sequenza di acquisizione.

Value (REAL) Ritorna il valore di acquisizione espresso nella unità definita dal modo di acquisizione. Potrebbe essere un valore NaN (Not A Number) ad indicare un problema nell'acquisizione, tipicamente sensore rotto.

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con [SysGetLastError](#) è possibile rilevare il codice di errore.

9983050 Errore allocazione blocco funzione.

9983060 Terminato spazio memoria rilocabile, non è possibile eseguire l'FB.

9983070 Errore versione blocco funzione.

9983080 Impossibile inizializzare il modulo.

9983100 Il modulo indirizzato in **Address** non è presente.

9983110~1 Il modulo indirizzato non supporta i comandi acquisizione analogica.

9983150 Il valore ritornato dal modulo analogico non è corretto.

9983200 Il modo di acquisizione definito in **Mode** non è gestito dal modulo.

9983210 Errore nella acquisizione analogica dal modulo.

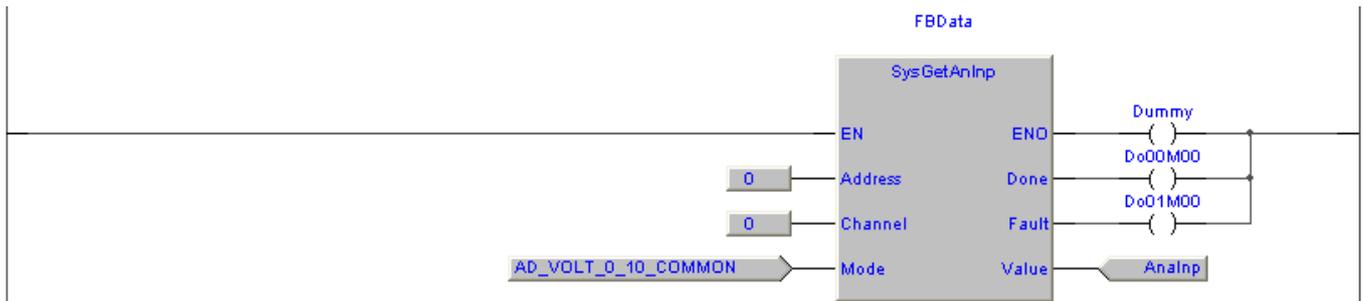
Esempi

Viene eseguita l'acquisizione analogica dal canale 0 del modulo 0 in modo 0-10 Volt. Se dato valido viene attivata l'uscita digitale **Do00M00** se errore di conversione viene attivata l'uscita digitale **Do01M00**. Il dato analogico acquisito nel range da 0.000 a 9.999 è trasferito nella variabile **AnaInp**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBData	SysGetAnInp	Auto	No	0	..	FB SysGetPhrDI data
2	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable
3	AnaInp	REAL	Auto	No	0	..	Analog value (Volts)

Esempio LD (Ptp116a100)



Esempio IL

```
(* Acquires analog input 0 from module. *)

LD 0
ST FB_SysGetAnInp.Address (* Set module address *)

LD 0
ST FB_SysGetAnInp.Channel (* Set channel *)

LD AD_VOLT_0_10_COMMON
ST FB_SysGetAnInp.Mode (* Set acquisition mode *)

CAL FBData (* Call the SysGetAnInp function block *)

LD FB_SysGetAnInp.Done
ST Do00M00 (* The output is active if data is acquired *)

LD FB_SysGetAnInp.Fault
ST Do01M00 (* The output is active if execution fault *)

LD FB_SysGetAnInp.Value
ST AnaInp (* Store the acquired value *)
```

Esempio ST

```
(* Acquires analog input 0 from module. *)

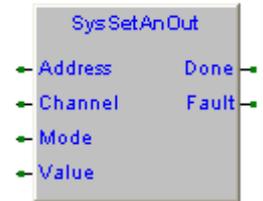
FB_SysGetAnInp(Address:=0, Channel:=0, Mode:=AD_VOLT_0_10_COMMON); (* Call the SysGetAnInp FB *)

Do00M00:=FB_SysGetAnInp.Done; (* The output is active if data is acquired *)
Do01M00:=FB_SysGetAnInp.Fault; (* The output is active if execution fault *)
VarReal:=FB_SysGetAnInp.Value; (* Store the acquired value *)
```

Type	Library	Version
FB	Embedded	3.0

7.15.5 SysSetAnOut, set analog output

Questo blocco funzione esegue il set del valore sull'uscita analogica sul modulo di uscita. Il blocco funzione gestisce vari modi di uscita in funzione del modulo analogico a cui fa riferimento.



Address (USINT) Occorre specificare l'indirizzo del modulo da cui eseguire l'acquisizione analogica (Range da 0x00 a 0x0F). Il valore 0x00 indica il primo modulo di estensione, 0x01 il secondo e così di seguito.

Channel (USINT) Occorre specificare l'indirizzo del canale sul modulo (Range da 0x00 a 0x0F). Se viene settato un indirizzo di canale non presente, si interrompe l'esecuzione e viene settato il bit di **Fault**.

Mode (USINT) Occorre specificare il modo di gestione uscita analogica secondo la tabella sottoriportata.

Mode	Define	Acquisition
1	DA_VOLT_0_10	Voltage from 0 to 10 V
2	DA_VOLT_0_5	Voltage from 0 to 5 V
3	DA_VOLT_M10_10	Voltage from -10 to +10 V
4	DA_VOLT_M5_5	Voltage from -5 to +5 V
5	DA_CURR_0_20	Current from 0 to 20 mA
6	DA_CURR_4_20	Current from 4 to 20 mA

Value (REAL) Occorre specificare il valore di uscita espresso nella unità definita dal modo di gestione.

Done (BOOL) Dato analogico acquisito, viene attivato per un loop al termine della acquisizione analogica.

Fault (BOOL) Errore di acquisizione, viene attivato in caso di errore nella sequenza di acquisizione.

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con [SysGetLastError](#) è possibile rilevare il codice di errore.

- 9982050 Errore allocazione blocco funzione.
- 9982060 Terminato spazio memoria rilocabile, non è possibile eseguire l'FB.
- 9982070 Errore versione blocco funzione.
- 9982080 Impossibile inizializzare il modulo.
- 9982100 Il modulo indirizzato in **Address** non è presente.
- 9982110~1 Il modulo indirizzato non supporta i comandi uscita analogica.
- 9982150 Il valore da impostare sul modulo analogico non è corretto.
- 9982200 Il modo di gestione definito in **Mode** non è gestito dal modulo.
- 9982210 Errore nella gestione uscita analogica sul modulo.

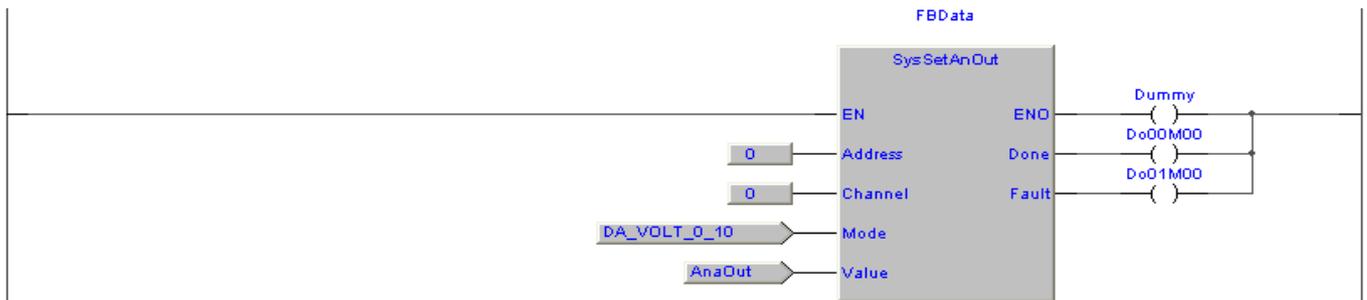
Esempi

Viene eseguita l'uscita analogica dal canale 0 del modulo 0 in modo 0-10 Volt. Se dato valido viene attivata l'uscita digitale **Do00M00** se errore di conversione viene attivata l'uscita digitale **Do01M00**. Il dato analogico da impostare in uscita nel range da 0.000 a 9.999 è presente nella variabile **AnaOut**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBData	SysSetAnOut	Auto	No	0	..	FB SysSetAnAout data
2	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable
3	AnaOut	REAL	Auto	No	0	..	Analog value (Volts)

Esempio LD (Ptp116a100)



Esempio IL

```
(* Manage analog output 0 on module 0. *)

LD 0
ST FBData.Address (* Set module address *)

LD 0
ST FBData.Channel (* Set channel *)

LD DA_VOLT_0_10
ST FBData.Mode (* Set management mode *)

LD AnaOut
ST FBData.Value (* Store the output value *)

CAL FBData (* Call the SysSetAnOut function block *)

LD FBData.Done
ST Do00M00 (* The output is active if data is set *)

LD FBData.Fault
ST Do01M00 (* The output is active if execution fault *)
```

Esempio ST

```
(* Manage analog output 0 on module 0. *)

FBData.Value:=AnaOut; (* Store the output value *)
FBData(Address:=0, Channel:=0, Mode:=DA_VOLT_0_10); (* Call the SysSetAnOut function block *)

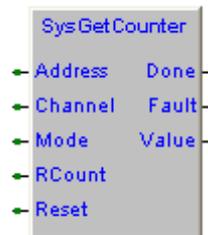
Do00M00:=FBData.Done; (* The output is active if data is set *)
Do01M00:=FBData.Fault; (* The output is active if execution fault *)
```

7.15.6 SysGetCounter, get counter

Type	Library	Version
FB	Embedded	5.0

Questo blocco funzione esegue la lettura di un contatore. Il blocco funzione può essere utilizzato per acquisire il valore del contatore presente sul modulo CPU SlimLine e dai moduli che gestiscono la funzione contatore.

E' prevista la gestione del reset valore di conteggio e della inversione conteggio. In base alla definizione di **Mode** è possibile gestire conteggio su fronte positivo, negativo o su entrambi i fronti dell'ingresso clock del contatore. Se il modulo che gestisce il contatore lo prevede è possibile anche definire comandi hardware (Ingressi logici) di reset conteggio e di inversione conteggio.



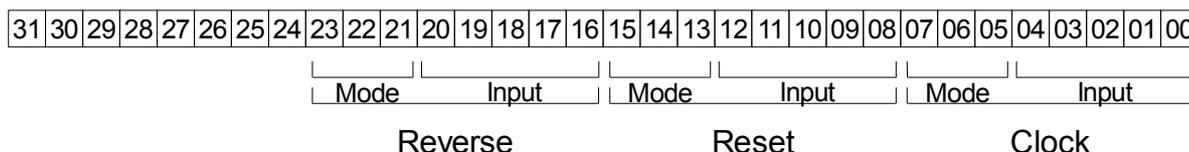
Address (USINT) Occorre specificare l'indirizzo di modulo da cui eseguire l'acquisizione encoder (Range da 0 a 255). Il valore 0 indica il primo modulo di estensione, 1 il secondo e così di seguito. L'indirizzo 255 indica il modulo CPU.

Se viene settato un indirizzo di modulo non presente, si interrompe l'esecuzione e viene settato il bit di **Fault**.

Channel (USINT) Occorre specificare l'indirizzo del canale sul modulo (Range da 0 a 15).

Se viene settato un indirizzo di canale non presente, si interrompe l'esecuzione e viene settato il bit di **Fault**.

Mode (UDINT) Modo acquisizione, espresso su 32 bits secondo lo schema riportato.



Clock	Input	Definisce ingresso logico da utilizzare come clock
	Mode	0: Conta su fronte salita 1: Conta su fronte discesa 2: Conta su entrambi i fronti
Reset	Input	Definisce ingresso logico da utilizzare come reset
	Mode	0: Non è utilizzato ingresso di reset 1: Reset counter se ingresso attivo 2: Reset conter se ingresso non attivo
Reverse	Input	Definisce ingresso logico da utilizzare come inversione conteggio
	Mode	0: Non è utilizzato ingresso di reverse 1: Inverte conteggio counter se ingresso attivo 2: Inverte conteggio conter se ingresso non attivo

Per calcolare il valore di mode si applica la formula:

$$((Reverse\ mode)*2097152)+((Reverse\ input)*65536)+((Reset\ mode)*8192)+((Reset\ input)*256)+((Clock\ mode)*32)+(Clock\ input)$$

Se viene settato un valore non corretto, si interrompe l'esecuzione e viene settato il bit di **Fault**.

RCount (BOOL) Reverse counting, attivando questo ingresso **Value** viene decrementato ad ogni variazione di conteggio.

Reset (BOOL) Attivando questo ingresso si ha il reset del valore di conteggio **Value**.

Done (BOOL) Dato counter acquisito, viene attivato per un loop al termine della acquisizione counter.

Fault (BOOL) Errore di acquisizione, viene attivato in caso di errore nella sequenza di acquisizione.

Value (UDINT) Valore contatore.

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con **SysGetLastError** è possibile rilevare il codice di errore.

- 9981050 Errore allocazione blocco funzione.
- 9981060 Terminato spazio memoria rilocabile, non è possibile eseguire l'FB.
- 9981070 Errore versione blocco funzione.
- 9981080 Impossibile inizializzare il modulo.
- 9981100 Il modulo indirizzato in **Address** non è presente.
- 9981110 Il canale definito in **Channel** non è gestito.
- 9981200~1 Il modo di gestione definito in **Mode** non è gestito dal modulo.
- 9981210 Errore nella gestione lettura counter dal modulo.

Esempi

Viene eseguita l'acquisizione del contatore dal modulo CPU di SlimLine, viene eseguito il conteggio su entrambi i fronti dell'ingresso di clock. Il valore di conteggio è trasferito nella variabile **Value**. Su fine conversione viene attivata l'uscita digitale **Do01M00** se errore di conversione viene attivata l'uscita digitale **Do02M00**.

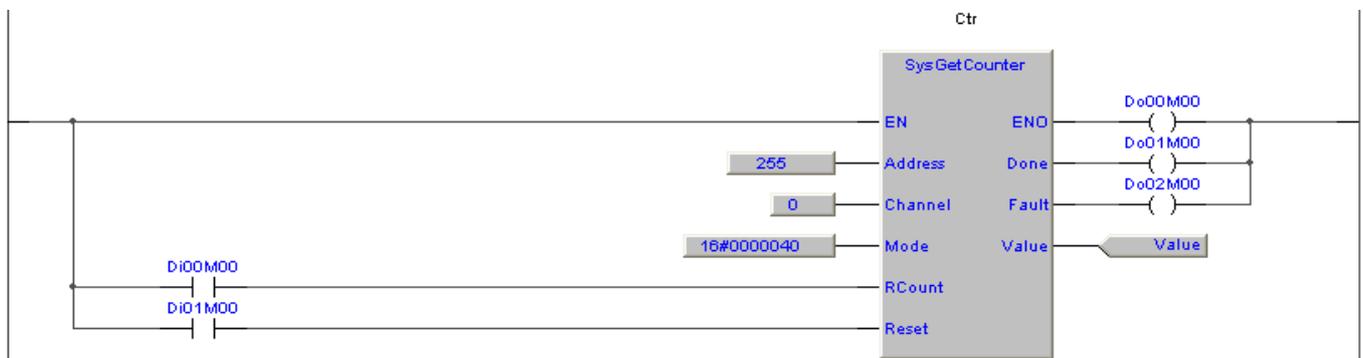
Attivando l'ingresso **Di00M00** viene eseguita l'inversione del conteggio ad ogni variazione dell'ingresso di clock viene decrementato il valore di **Value** in uscita.

Attivando l'ingresso **Di01M00** viene eseguito il reset del conteggio il valore di **Value** in uscita è resettato.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Value	UDINT	Auto	No	0	..	Counter value
2	Ctr	SysGetCounter	Auto	No	0	..	SysGetCounter FB data

Esempio LD

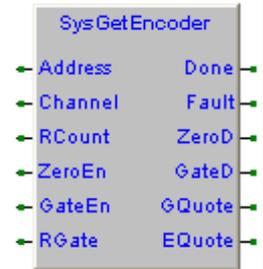


Type	Library	Version
FB	Embedded	3.0

7.15.7 SysGetEncoder, get encoder input

Questo blocco funzione esegue la lettura di un canale encoder. Il blocco funzione può essere utilizzato solo su sistemi che hanno moduli in grado di acquisire encoders incrementali.

E' prevista la gestione della tacca di zero e la possibilità di acquisire valori di quota all'interno di un segnale logico di gate.



- Address** (USINT) Occorre specificare l'indirizzo di modulo da cui eseguire l'acquisizione encoder (Range da 0 a 15). Il valore 0 indica il primo modulo di estensione, 1 il secondo e così di seguito.
Se viene settato un indirizzo di modulo non presente, si interrompe l'esecuzione e viene settato il bit di **Fault**.
- Channel** (USINT) Occorre specificare l'indirizzo del canale sul modulo (Range da 0 a 15).
Se viene settato un indirizzo di canale non presente, si interrompe l'esecuzione e viene settato il bit di **Fault**.
- RCount** (BOOL) Reverse counting, attivando questo ingresso si inverte l'incremento di quota **EQuote** in funzione della direzione di rotazione encoder.
- ZeroEn** (BOOL) Attivando questo ingresso si ha il reset della quota **EQuote** al passaggio della tacca di zero encoder.
- GateEn** (BOOL) Attivando questo ingresso sul fronte di variazione dell'ingresso di Gate viene trasferito il valore di **EQuote** in **GQuote**.
- RGate** (BOOL) Attivando questo ingresso viene gestito il fronte di disattivazione dell'ingresso Gate.
- Done** (BOOL) Dato encoder acquisito, viene attivato per un loop al termine della acquisizione encoder.
- Fault** (BOOL) Errore di acquisizione, viene attivato in caso di errore nella sequenza di acquisizione.
- ZeroD** (BOOL) Tacca di zero encoder acquisita, viene settata su acquisizione tacca di zero encoder, si resetta disattivando l'ingresso **ZeroEn**.
- GateD** (BOOL) Segnale di Gate acquisito, viene attivato per un loop alla acquisizione del segnale Gate.
- GQuote** (UINT) Quota di gate, valore di quota encoder **EQuote** memorizzata sul fronte selezionato del segnale Gate.
- EQuote** (UINT) Quota encoder, valore di quota encoder, al raggiungimento del valore minimo o massimo viene eseguito il roll over.

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con [SysGetLastError](#) è possibile rilevare il codice di errore.

- 9980050 Errore allocazione blocco funzione.
- 9980060 Terminato spazio memoria rilocabile, non è possibile eseguire l'FB.
- 9980070 Errore versione blocco funzione.
- 9980080 Impossibile inizializzare il modulo.
- 9980100 Il modulo indirizzato in **Address** non è presente.
- 9980110~2 Il modulo indirizzato non supporta i comandi acquisizione encoder.
- 9980200 Il modo di gestione definito in **Mode** non è gestito dal modulo.
- 9980210~2 Errore nella gestione lettura encoder dal modulo.

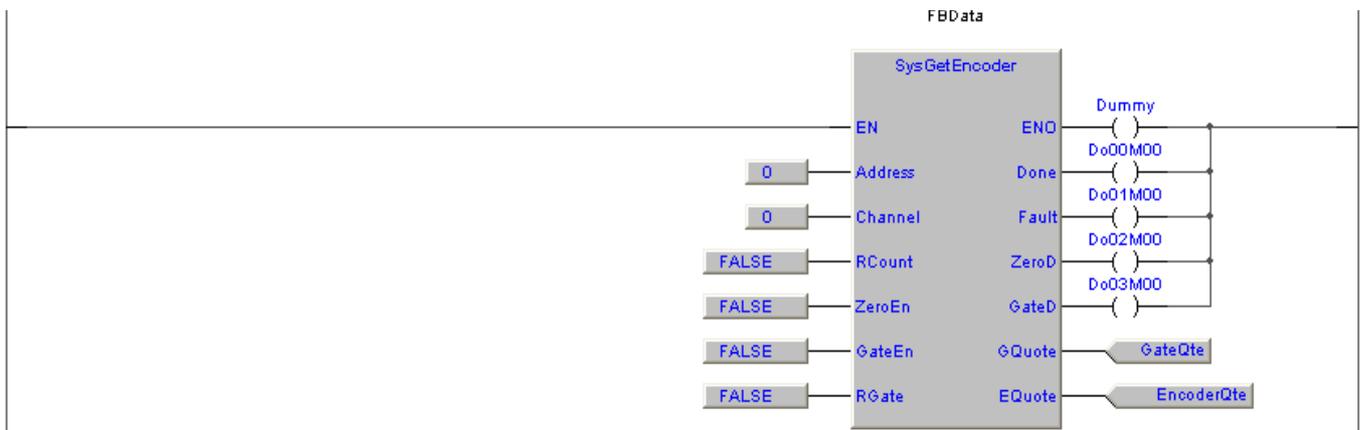
Esempi

Viene eseguita l'acquisizione dell'ingresso encoder dal canale 0 del modulo 0, il valore di quota encoder è trasferito nella variabile **EncoderQte**. Su fine conversione viene attivata l'uscita digitale **Do00M00** se errore di conversione viene attivata l'uscita digitale **Do01M00**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBData	SysGetEncoder	Auto	No	0	..	FB SysGetEncoder data
2	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable
3	GateQte	UINT	Auto	No	0	..	Gate quote
4	EncoderQte	UINT	Auto	No	0	..	Encoder quote

Esempio LD (Ptp116a100)



Esempio IL (Ptp116a100)

```
(* Acquires encoder 0 from module. *)

LD 0
ST FBData.Address (* Set module address *)

LD 0
ST FBData.Channel (* Set channel *)

LD FALSE
ST FBData.RCount (* Reverse counting *)
ST FBData.GateEn (* Gate enable *)
ST FBData.RGate (* Reverse gate *)

CAL FBData (* Call the SysGetEncoder function block *)

LD FBData.Done
ST Do00M00 (* The output is active if data is acquired *)

LD FBData.Fault
ST Do01M00 (* The output is active if execution fault *)

LD FBData.GQuote
ST GateQte (* Gate quote *)

LD FBData.EQuote
ST EncoderQte (* Encoder quote *)
```

7.15.8 SysPhrVRd, read variable from peripheral module

Type	Library	Version
Function	Embedded	5.0

Questa funzione esegue la lettura di una variabile dal modulo periferico di estensione.

Occorre definire l'indirizzo di modulo **Module**, l'indirizzo della variabile da leggere sul modulo periferico **RdAdd**, il tipo di variabile **VarType** e l'indirizzo del buffer dove trasferire il valore letto **VarAdd**.



Parametri funzione:

- Module** (USINT) Occorre specificare l'indirizzo di modulo da cui eseguire la lettura (Range da 0 a 15). Il valore 0 indica il primo modulo di estensione, 1 il secondo e così di seguito.
- RdAdd** (UINT) Indirizzo variabile da leggere come allocata sul modulo periferico.
- VarType** (USINT) Tipo variabile, come indicato nella tabella [Variable types definition](#).
- VarAdd** (UDINT) Indirizzo variabile.

La funzione ritorna:

(BOOL) **FALSE**: Errore esecuzione. **TRUE**: Funzione eseguita correttamente.

Codici di errore

In caso di errore la funzione torna **FALSE** e con [SysGetLastError](#) è possibile rilevare il codice di errore.

- 9989100 Il modulo indirizzato in **Module** non è presente.
- 9989110 Il tipo variabile definito in **VarType** non è corretto.
- 9989200 Errore durante l'esecuzione della lettura della variabile dal modulo periferico.

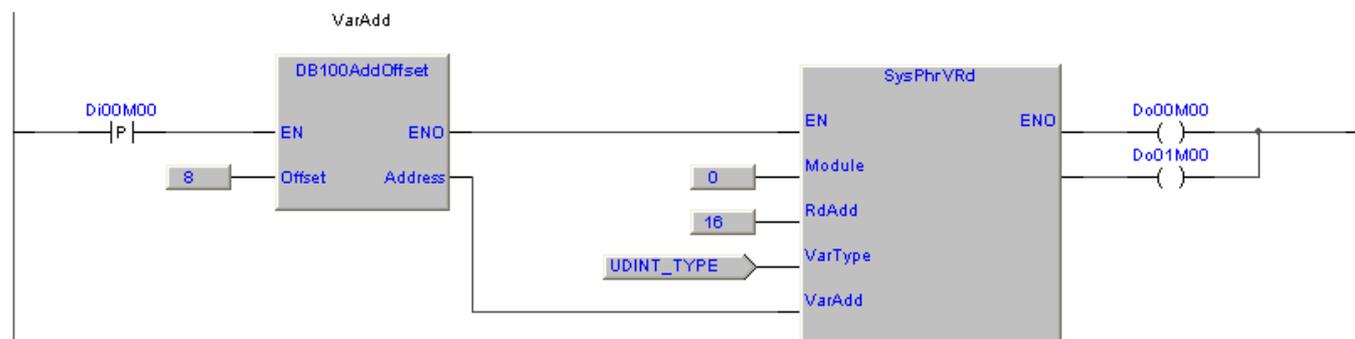
Esempi

Attivando l'ingresso **Di00M00** viene eseguita la lettura della variabile **UDINT** da indirizzo **16** dal modulo periferico **0**. Il valore della variabile è trasferito nella **DB100** ad offset **8**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	VarAdd	DB100AddOffset	Auto	No	0	..	Variable address calculation

Esempio LD

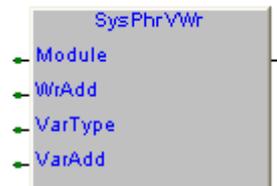


7.15.9 SysPhrVWr, write variable to peripheral module

Type	Library	Version
Function	Embedded	5.0

Questa funzione esegue la scrittura di una variabile sul modulo periferico di estensione.

Occorre definire l'indirizzo di modulo **Module**, l'indirizzo della variabile da scrivere sul modulo periferico **WrAdd**, il tipo di variabile **VarType** e l'indirizzo del buffer dove si trova il valore da scrivere **VarAdd**.



Parametri funzione:

- Module** (USINT) Occorre specificare l'indirizzo di modulo su cui eseguire la scrittura (Range da 0 a 15). Il valore 0 indica il primo modulo di estensione, 1 il secondo e così di seguito.
- WrAdd** (UINT) Indirizzo variabile da scrivere come allocata sul modulo periferico.
- VarType** (USINT) Tipo variabile, come indicato nella tabella [Variable types definition](#).
- VarAdd** (UDINT) Indirizzo variabile.

La funzione ritorna:

(BOOL) **FALSE**: Errore esecuzione. **TRUE**: Funzione eseguita correttamente.

Codici di errore

In caso di errore la funzione torna **FALSE** e con [SysGetLastError](#) è possibile rilevare il codice di errore.

- 9988100 Il modulo indirizzato in **Module** non è presente.
- 9988110 Il tipo variabile definito in **VarType** non è corretto.
- 9988200 Errore durante l'esecuzione della lettura della variabile dal modulo periferico.

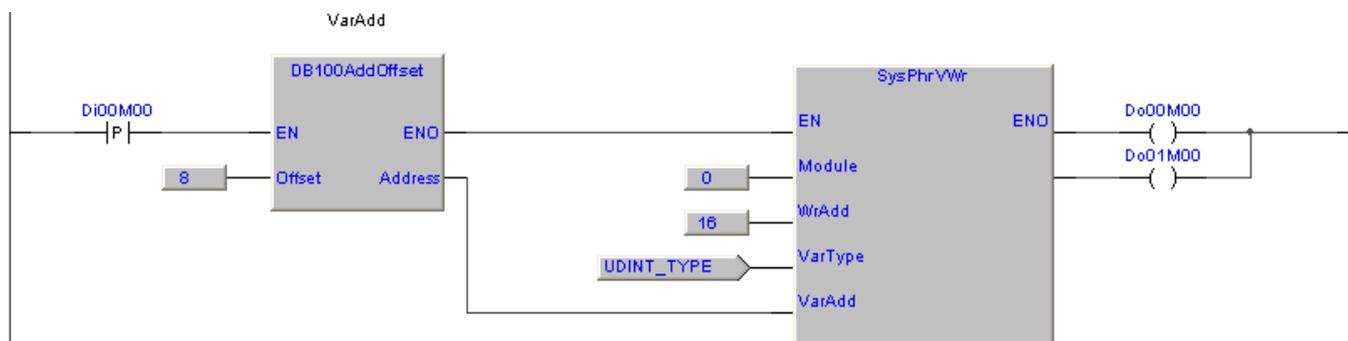
Esempi

Attivando l'ingresso **Di00M00** viene eseguita la scrittura della variabile **UDINT** ad indirizzo **16** sul modulo periferico **0**. Il valore da scrivere è presente nella **DB100** ad offset **8**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	VarAdd	DB100AddOffset	Auto	No	0	..	Variable address calculation

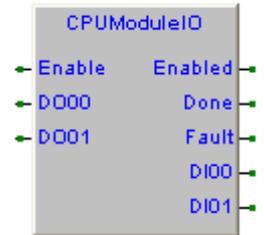
Esempio LD



7.15.10 CPUModuleIO, CPU module I/O management

Type	Library	Version
FB	PLCUtyLib	SFR054A400

Questo blocco funzione esegue la lettura dei due ingressi digitali e la gestione delle due uscite digitali presenti sul modulo CPU dello SlimLine.



- Enable** (BOOL) Abilitazione gestione I/O, attivando l'ingresso vengono acquisiti gli ingressi logici e gestite le uscite logiche. Disattivando l'ingresso vengono disattivate anche le uscite logiche sul modulo CPU.
- DO00** (BOOL) Stato della uscita logica DO00 presente sul modulo CPU.
- DO01** (BOOL) Stato della uscita logica DO01 presente sul modulo CPU.
- Enabled** (BOOL) Blocco funzione abilitato.
- Done** (BOOL) Acquisizione input e gestione output eseguita.
- Fault** (BOOL) Errore su acquisizione input e gestione output.
- DI00** (BOOL) Stato dell'ingresso logico DI00 presente sul modulo CPU.
- DI01** (BOOL) Stato dell'ingresso logico DI01 presente sul modulo CPU.

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con [SysGetLastError](#) è possibile rilevare il codice di errore.

- 10001100 Errore su esecuzione funzione lettura ingressi logici da modulo CPU
- 10001200 Errore su esecuzione funzione di set uscite logiche su modulo CPU

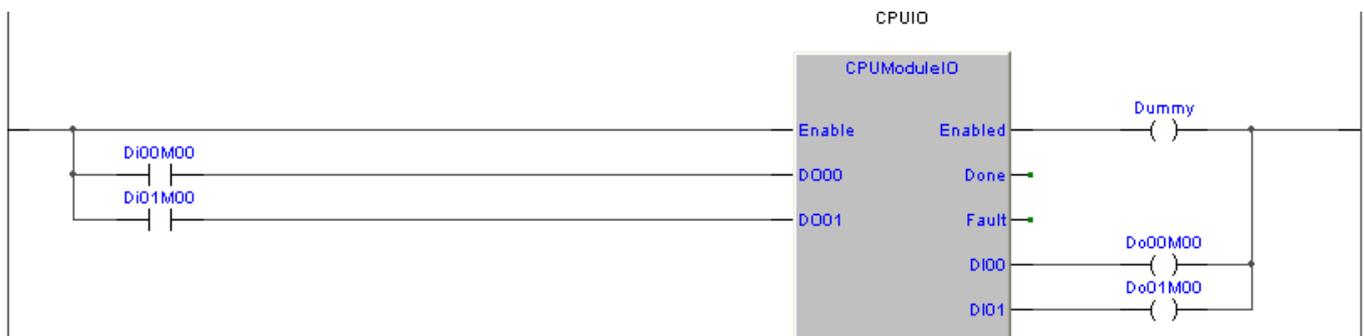
Esempi

Viene eseguita la lettura dei due ingressi digitali e la gestione delle due uscite digitali presenti sul modulo CPU. Lo stato dei due ingressi logici è copiato nelle uscite **Do00M00** e **Do01M00**. Lo delle variabili **Di00M00** e **Di01M00** è trasferito sulle due uscite logiche.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	CPUIO	CPUModuleIO	Auto	No	0	..	CPUModuleIO function block
2	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable

Esempio LD (Ptp114a200)



7.16 Funzioni ed FB di utilità generale

7.16.1 DB100AddOffset, returns DB100 address offset

Type	Library	Version
FB	ePLCUtyLib	SFR054A400

Questa funzione ritorna l'indirizzo della locazione di memoria all'interno della **DB100**, il cui offset è fornito come parametro.



Parametri funzione:

Offset (UINT) Offset locazione memoria

La funzione ritorna:

(UDINT) Indirizzo della locazione di memoria

Esempi

La variabile **ValueToPrint** è allocata nella **DB100** ad offset **8**, ogni secondo ne viene incrementato il valore e su porta seriale **COM0** viene inviata la stringa **Value is: xxxx**.

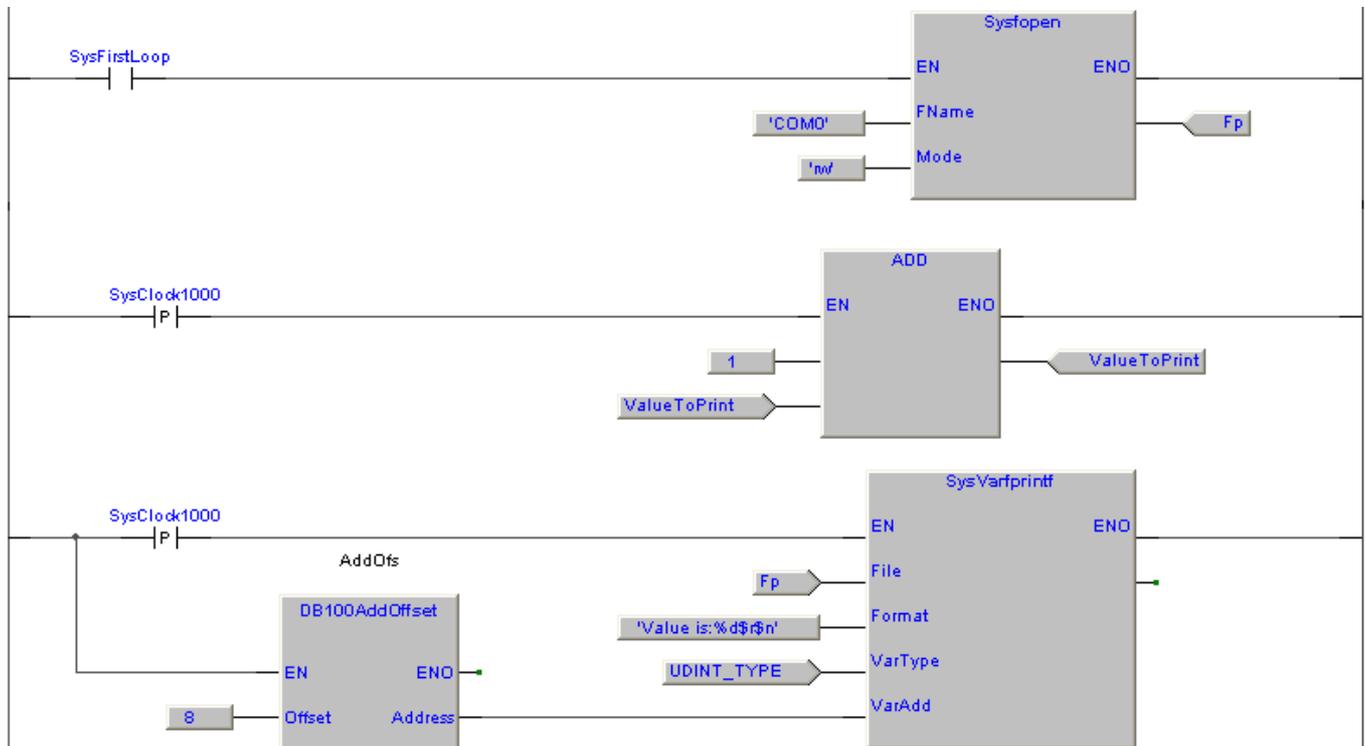
Definizione variabili globali

	Name	Type	Address	Group	Array	Init value	Attribute	Description
1	ValueToPrint	UDINT	%MD100.8		No	0	..	Value to print

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer
2	AddOfs	DB100AddOffset	Auto	No	0	..	Address offset calculation

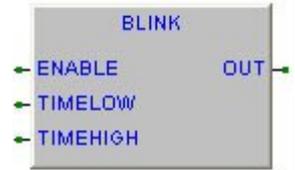
Esempio LD



Type	Library	Version
FB	PLCUtyLib	SFR054A000

7.16.2 BLINK, blink command

Questo blocco funzione gestisce una uscita lampeggiante con tempo di ciclo definibile. Attivando l'ingresso **ENABLE** l'uscita **OUT** inizia a lampeggiare con tempi di ciclo alto e basso definiti.



- ENABLE** (BOOL) Abilitazione blocco funzione, attivandolo viene gestita l'uscita OUT lampeggiante. Disattivandolo l'uscita OUT viene resettata.
- TIMELOW** (UDINT) Definisce il tempo in cui l'uscita OUT rimane nello stato logico low, espresso in mS.
- TIMEHIGH** (UDINT) Definisce il tempo in cui l'uscita OUT rimane nello stato logico high espresso in mS.
- OUT** (BOOL) Stato uscita lampeggiante.

Esempi

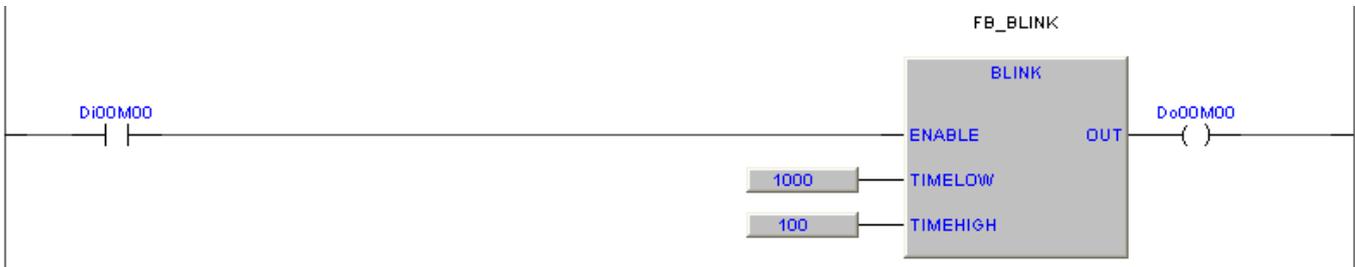
Viene impostato un lampeggio con 100 mS di tempo On e 1000 mS di tempo Off. Attivando l'ingresso digitale **Di00M00** l'uscita l'uscita digitale **Do00M00** lampeggia con i tempi definiti.

Disattivando l'ingresso digitale **Di00M00** l'uscita digitale **Do00M00** si azzerava immediatamente.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FB_eBLINK	BLINK	Auto	No	0	..	eBLINK (Blink out function block)

Esempio LD (Ptp114a100)



Esempio IL (Ptp114a100)

```

CAL FB_BLINK (* Call the BLINK function block *)

LD Di00M00
ST FB_BLINK.ENABLE (* Transfer the digital input to enable input *)

LD 1000
ST FB_BLINK.TIMELOW (* Set the time low *)

LD 100
ST FB_BLINK.TIMEHIGH (* Set the time high *)

LD FB_BLINK.OUT
ST Do00M00 (* Copy FB output to logic output *)
    
```

Esempio ST (Ptp114a100)

```

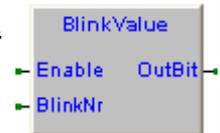
FB_BLINK(TIMELOW:=1000, TIMEHIGH:=100); (* Call the BLINK function block *)

FB_BLINK.ENABLE:=Di00M00; (* Transfer the digital input to FB enable *)
Do00M00:=FB_BLINK.OUT; (* Transfer the FB output to digital output *)
    
```

7.16.3 BlinkValue, blink out value

Type	Library	Version
FB	PLCUtyLib	SFR054A800

Questo blocco funzione gestisce una uscita lampeggiante con possibilità di definire il numero di lampeggi. Attivando l'ingresso **Enable** e definendo il numero di lampeggi in **BlinkNr**, l'uscita **OutBit** inizia a lampeggiare con il numero di lampeggi definito.



Il numero di lampeggi è definito in decine ed unità, il valore delle decine è riportato con un lampeggio lento (1 Sec), mentre il numero delle unità è riportato con un lampeggio veloce (250 mS). Una pausa di 3 Sec separa le sequenze di lampeggio.

Enable (BOOL) Abilitazione blocco funzione, attivandolo viene gestita l'uscita **OutBit** lampeggiante. Disattivandolo l'uscita viene resettata.

BlinkNr (USINT) Definisce il numero di lampeggi dell'uscita **OutBit**. Definendo tempo 0 l'uscita si disattiva.

OutBit (BOOL) Stato uscita lampeggiante.

Esempi

Attivando l'ingresso digitale **Di00M00** l'uscita l'uscita digitale **Do00M00** lampeggia con 2 lampeggi lenti (1 Sec), 3 lampeggi veloci (250 mS) ed una pausa di 3 Sec.

Disattivando l'ingresso digitale **Di00M00** l'uscita digitale **Do00M00** si azzerava immediatamente.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBData	BlinkValue	Auto	No	0	..	FB BlinkValue data

Esempio LD (Ptp114a500)



Esempio IL (Ptp114a500)

```

CAL FBData (* Call the "BlinkValue" function block *)

LD Di00M00
ST FBData.Enable (* Transfer the digital input to enable input *)

LD 23
ST FBData.BlinkNr (* Set the number of blink *)

LD FBData.OutBit
ST Do00M00 (* Copy FB output to logic output *)
    
```

Esempio ST (Ptp114a100)

```

FBData(BlinkNr:=23); (* Call the BLINK function block *)

FBData.Enable:=Di00M00; (* Transfer the digital input to FB enable *)
Do00M00:=FBData.OutBit; (* Transfer the FB output to digital output *)
    
```

7.16.4 ModbusRTUMaster, modbus Rtu master

Type	Library	Version
FB	PLCUtyLib	SFR054A500

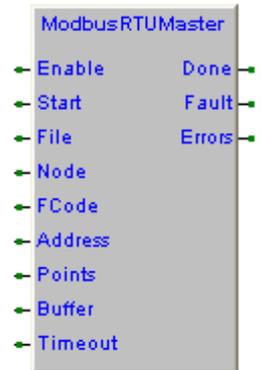
Questo blocco funzione esegue la gestione del protocollo modbus master, è possibile definire il terminale di I/O su cui effettuare la comunicazione **File**. Attivando il comando di **Start** sul terminale di I/O viene inviato un frame per eseguire la funzione modbus definita in **Function**.

Il comando **Start** se attivato in modo impulsivo permette l'esecuzione del comando definito una sola volta. Se è sempre attivo permette l'esecuzione ciclica del comando definito.

Se la funzione è una funzione di lettura, il valore delle variabili a partire dall'indirizzo definito in **Address** per il numero di variabili definito da **Points**, viene letto dal sistema slave e trasferito nelle variabili indirizzate da **Buffer**.

Se la funzione è una funzione di scrittura, il valore delle variabili presenti nel buffer di memoria indirizzato da **Buffer** per il numero di variabili definito da **Points**, è inviato al dispositivo slave che lo trasferirà nelle sue variabili a partire dall'indirizzo definito in **Address**.

Terminato il comando viene attivata per un loop l'uscita **Done**, in caso di errore esecuzione comando o tempo di esecuzione comando superiore al tempo definito in **Timeout**, viene attivata per un loop l'uscita **Fault** ed incrementato il valore in **Errors**.



- Enable** (BOOL) Comando di abilitazione blocco funzione.
- Start** (BOOL) Comando di esecuzione comando modbus.
- File** (FILEP) Flusso dati **stream** ritornato dalla funzione **Sysfopen**.
- Node** (USINT) Numero di nodo modbus su cui effettuare il comando (Range da 0 a 255).
- FCode** (USINT) Codice funzione modbus da eseguire nel comando (Range da 0 a 255).

Codice	Comando
01	Read coil status (Massimo 255 coils)
03	Read holding registers (Massimo 32 registri)
04	Read input registers (Massimo 32 registri)
0F	Force multiple coils (Massimo 255 coils)
10	Preset multiple registers (Massimo 32 registri)

- Address** (UINT) Indirizzo di allocazione variabili su sistema slave. In accordo alle specifiche modbus l'indirizzo inviato nel frame dati è (**Address-1**) (Range da 16#0001 a 16#FFFF).
- Points** (USINT) Numero di variabili consecutive su cui opera il comando (Range da 1 a 32).
- Buffer** (@USINT) Indirizzo buffer dati letti o da scrivere.
- Timeout** (UINT) Tempo massimo esecuzione comando espresso in mS. Se il comando non termina nel tempo definito viene abortito ed attivata l'uscita **Fault**.
- Done** (BOOL) Attivo per un loop al termine della esecuzione del comando.
- Fault** (BOOL) Attivo per un loop su errore esecuzione del comando.
- Errors** (UDINT) Numero di errori, incrementato ad ogni nuovo errore, raggiunto valore massimo riparte da 0.

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con [SysGetLastError](#) è possibile rilevare il codice di errore.

- 10007010 Valore di **File** non definito.
- 10007050 Timeout esecuzione.
- 10007060 Errore esecuzione.
- 10007100 Codice funzione definito in **Function** non gestito.
- 10007120 Valore di **Points** errato.
- 10007500 Errore in ricezione frame (Codice comando errato).
- 10007520 Errore in ricezione frame (CRC frame errato).
- 10007540 Errore in ricezione frame (Dati errati).

Esempi

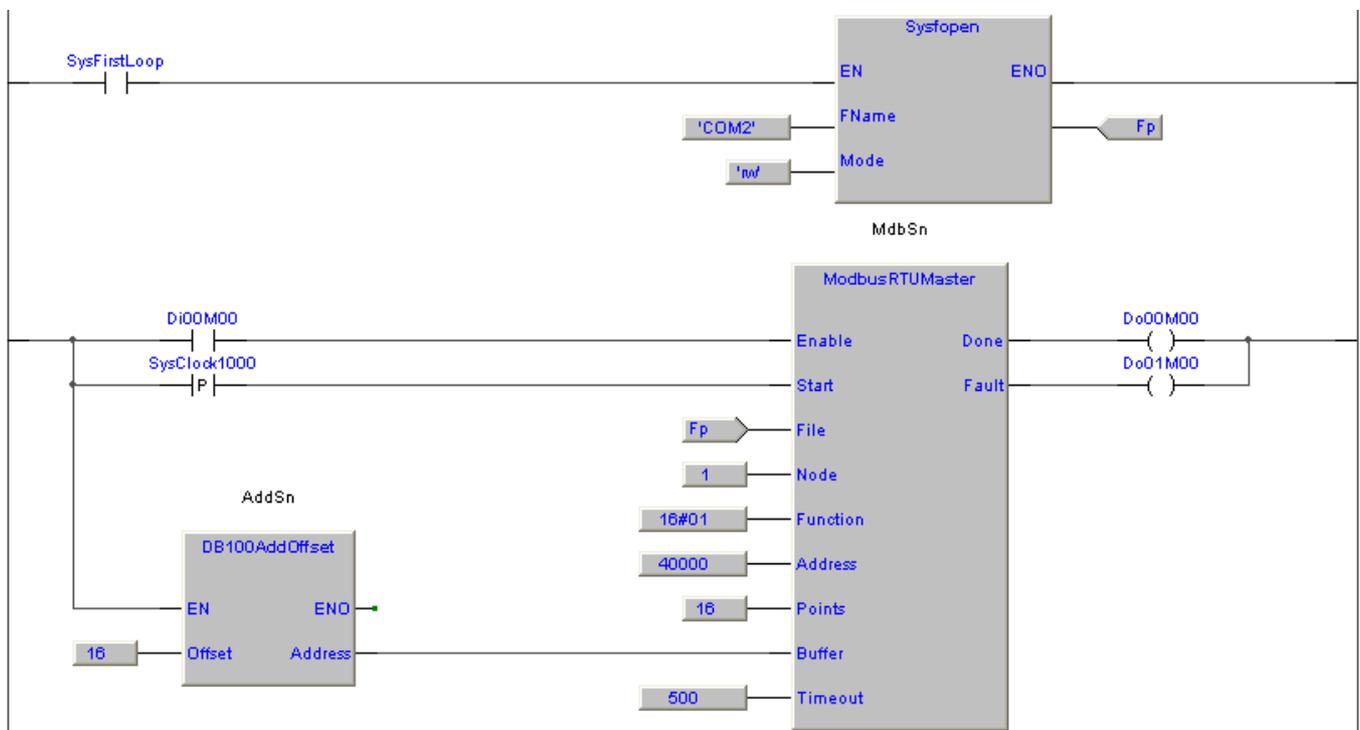
Attivando l'ingresso digitale **Di00M00** viene eseguita ogni secondo la lettura di 16 coils a partire da indirizzo 16#01 dal nodo modbus 1. Il valore dei coils letti è trasferito nella DB100 a partire da indirizzo 16. Terminata la lettura si attiverà per un loop l'uscita logica **Do00M00**.

Nella variabile BOOL ad indirizzo MX100.16 verrà trasferita la variabile BOOL MX100.0 del sistema slave, ad indirizzo MX100.17 la variabile MX100.1, ad indirizzo MX100.18 la variabile MX100.2 e così via.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer
2	MdbSn	ModbusRTUMaster	Auto	No	0	..	Modbus RTU master
3	AddSn	DB100AddOffset	Auto	No	0	..	Address offset

Esempio LD (Ptp114a300)



Type	Library	Version
FB	PLCUtyLib	SFR054A000

7.16.5 MDBRTUMASTER, modbus Rtu master

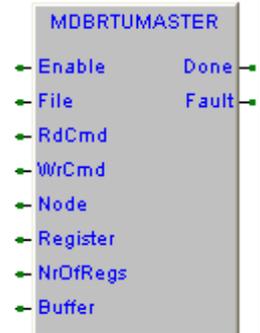
Blocco funzione obsoleto, si consiglia di utilizzare [ModbusRTUMaster](#).

Questo blocco funzione esegue la gestione del protocollo modbus master, è possibile definire il terminale di I/O su cui effettuare la comunicazione **File**.

Attivando il comando di read **RdCmd** sul terminale di I/O viene inviato un frame modbus con il comando di **Read Holding Registers** (0x03) ed il valore ritornato dei registri viene trasferito nella variabile indirizzata da **Buffer**.

Attivando il comando di write **WrCmd** sul terminale di I/O viene inviato un frame modbus con il comando di **Preset Multiple Registers** (0x10) con il valore dei registri acquisito dalla variabile indirizzata da **Buffer**.

Terminato il comando viene attivato per un loop l'uscita **Done**.



- Enable** (BOOL) Comando di abilitazione blocco funzione.
- File** (FILEP) Flusso dati **stream** ritornato dalla funzione **Sysfopen**.
- RdCmd** (BOOL) Comando di esecuzione lettura registri.
- WrCmd** (BOOL) Comando di esecuzione scrittura registri.
- Node** (USINT) Numero di nodo modbus su cui effettuare il comando (Range da 0 a 255).
- Register** (UDINT) Indirizzo di inizio lettura o scrittura registri su nodo modbus. In accordo alle specifiche modbus l'indirizzo inviato nel frame dati è (**Register-1**) (Range da 16#0001 a 16#FFFF).
- NrOfRegs** (USINT) Numero di registri consecutivi da leggere o scrivere (Range da 1 a 32).
- Buffer** (@UINT) Indirizzo buffer dati letti o da scrivere.
- Done** (BOOL) Attivo per un loop al termine della esecuzione del comando.
- Fault** (BOOL) Attivo per un loop su errore esecuzione del comando.

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con [SysGetLastError](#) è possibile rilevare il codice di errore.

- 10000010 Valore di **File** non definito.
- 10000050 Timeout esecuzione
- 20000060 Errore sequenze gestione comando
- 10000100 Valore di **Register** errato
- 10000102
- 10000200 Frame risposta a comando read in errore
- 10000300 Frame risposta a comando write in errore
- 10000400 Errore in ricezione frame (Codice comando errato)
- 10000410 Errore in ricezione frame (CRC frame errato)

Esempi

Viene eseguita la lettura ogni 100 mS di 6 registri a partire da indirizzo 16#100 dal nodo modbus 1. Il valore dei registri letti è trasferito nell'array **Registers**.

Spiando la porta seriale COM2 con un programma di emulazione terminale in grado di visualizzare i caratteri esadecimali vedremo ogni 100 mS la stringa con il comando modbus di **Read Holding Register**: 01 03 01 00 00 06 C4 34.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Terminal I/O pointer
2	Registers	UINT	Auto	[0..9]	10(0)	..	Registers area
3	Mdb	MDBRTUMASTER	Auto	No	0	..	Modbus Rtu master FB
4	Counter	UDINT	Auto	No	0	..	Messages counter
5	Errors	UDINT	Auto	No	0	..	Errors counter
6	Sm	SYSSEIALMODE	Auto	No	0	..	Serial mode
7	AFlag	BOOL	Auto	No	FALSE	..	Auxiliary flag
8	Trigger	R_TRIG	Auto	No	0	..	Raising trigger FB

Esempio ST (Ptp114a200)

```

(* ----- *)
(* OPEN THE COMMUNICATION PORT *)
(* ----- *)
(* Here open the COM2 port in read/write. *)

IF (Fp = NULL) THEN
  Fp:=Sysfopen('COM2', 'rw'); (* Terminal I/O pointer *)
END_IF;

(* ----- *)
(* INITIALIZATION *)
(* ----- *)
(* Set the serial mode. *)

IF (SysFirstLoop) THEN
  AFlag:=SysGetSerialMode(ADR(Sm), Fp); (* Get serial mode *)
  Sm.Baudrate:=57600;
  Sm.Parity:='E';
  Sm.DTRManagement:=DTR_AUTO_WO_TIMES;
  AFlag:=SysSetSerialMode(ADR(Sm), Fp); (* Set serial mode *)
END_IF;

(* ----- *)
(* MODBUS MASTER *)
(* ----- *)
(* Preset the modbus master FB parameters. *)

Mdb.Enable:=TRUE; (* Function enable *)
Mdb.File:=Fp; (* Terminal I/O pointer *)
Mdb.Node:=1; (* Node number *)
Mdb.Register:=257; (* Start register address *)
Mdb.NrOfRegs:=6; (* Number of registers *)
Mdb.Buffer:=ADR(Registers); (* Address of data buffer *)

(* Call the modbus master FB an execute read command every 100 mS. *)

Trigger(CLK:=SysClock100);
Mdb(RdCmd:=Trigger.Q);

(* Check if done or error and count them. *)

IF (Mdb.Done) THEN Counter:=Counter+1; END_IF;
IF (Mdb.Fault) THEN Errors:=Errors+1; END_IF;

(* [End of file] *)

```

7.16.6 ModbusRTUSlave, modbus Rtu slave

Type	Library	Version
FB	PLCUtyLib	SFR054A700

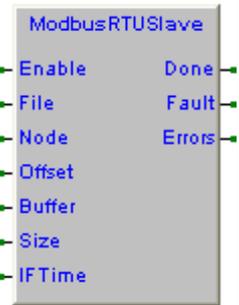
Sui sistemi SlimLine il protocollo modbus slave è già implementato dal sistema operativo, pertanto non occorre inserire blocchi funzione appositi nel programma utente. Questo blocco esegue l'override della gestione di sistema operativo e si utilizza in casi particolari, dove non è possibile utilizzare la gestione implementata nel sistema operativo.

Questo blocco funzione esegue la gestione del protocollo modbus slave, è possibile definire il terminale di I/O su cui effettuare la comunicazione **File**.

Occorre definire il nodo modbus **Node**, e l'eventuale offset di indirizzo frame modbus **Offset**. I comandi modbus ricevuti operano sul buffer di memoria il cui indirizzo è definito in **Buffer** e la dimensione in bytes è definita in **Size**.

In **IFTime** occorre definire il tempo di interframe dei comandi modbus, cioè il tempo che intercorre tra la ricezione di un comando ed il comando successivo. Su linea seriale questo tempo coincide con il tempo di ricezione di 3 caratteri al baud rate definito.

Alla ricezione di ogni comando modbus corretto si attiva per un loop l'uscita **Done**, in caso di errore comando viene attivata per un loop l'uscita **Fault** ed incrementato il valore in **Errors**.



- Enable** (BOOL) Comando di abilitazione blocco funzione.
- File** (FILEP) Flusso dati **stream** ritornato dalla funzione **Sysfopen**.
- Node** (USINT) Numero di nodo modbus (Range da 0 a 255).
- Offset** (UINT) Offset su indirizzo modbus ricevuto nel frame dati (Range da 16#0000 a 16#FFFF).
- Address** (UINT) Indirizzo di allocazione variabili su sistema slave. In accordo alle specifiche modbus l'indirizzo inviato nel frame dati è (**Address-1**) (Range da 16#0001 a 16#FFFF).
- Buffer** (@USINT) Indirizzo buffer dati su cui operano i comandi modbus.
- Size** (UINT) Dimensione in byte del buffer dati su cui operano i comandi modbus.
- IFTime** (UDINT) Tempo che intercorre tra la ricezione di un comando ed il comando successivo (uS). Se comunicazione su porta seriale il tempo deve essere definito in base al baud rate.

Baud rate	Tempo
300	112000
600	56000
1200	28000
2400	14000
4800	7000
9600	3430

Baud rate	Tempo
19200	1720
38400	860
57600	573
76800	429
115200	286

- Done** (BOOL) Attivo per un loop alla ricezione di comando modbus.
- Fault** (BOOL) Attivo per un loop su errore ricezione comando modbus.
- Errors** (UDINT) Numero di errori riscontrati. Viene incrementato ad ogni nuovo errore, raggiunto il valore massimo il conteggio riparte da 0.

Comandi supportati

Il blocco funzione supporta solo alcuni comandi previsti dal protocollo modbus, i comandi supportati sono:

Codice	Comando
01	Read coil status (Massimo 250 coils)
02	Read input status (Massimo 250 coils)
03	Read holding registers (Massimo 125 registri)
04	Read input registers (Massimo 125 registri)
05	Force single coil
06	Preset single register
08	Loopback diagnostic test
0F	Force multiple coils (Massimo 250 coils)
10	Preset multiple registers (Massimo 125 registri)

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con [SysGetLastError](#) è possibile rilevare il codice di errore. In caso di eccezione su comando modbus viene riportato il codice di errore ma non viene attivata l'uscita **Fault**.

- 10019010 Valore di **File** non definito.
- 10019060 Errore esecuzione.
- 10019100 Errore in ricezione frame (Lunghezza frame modbus troppo lunga).
- 10019110 Errore in ricezione frame (Lunghezza frame modbus troppo corta).
- 10019120 Errore in ricezione frame (CRC modbus errato).
- 10019130 Errore in ricezione frame (Lunghezza frame modbus errata).
- 10019200 Errore trasmissione frame risposta
- 10019501 Eccezione 01. **Illegal function**, comando ricevuto non è tra quelli gestiti.
- 10019502 Eccezione 02. **Illegal data address**, comando ricevuto ha indirizzo o numero dati fuori range.
- 10019503 Eccezione 03. **Illegal data value**, comando ricevuto ha campo dati fuori range.
- 10019504 Eccezione 04. **Failure in associated device**, comando ricevuto contiene imprecisioni.

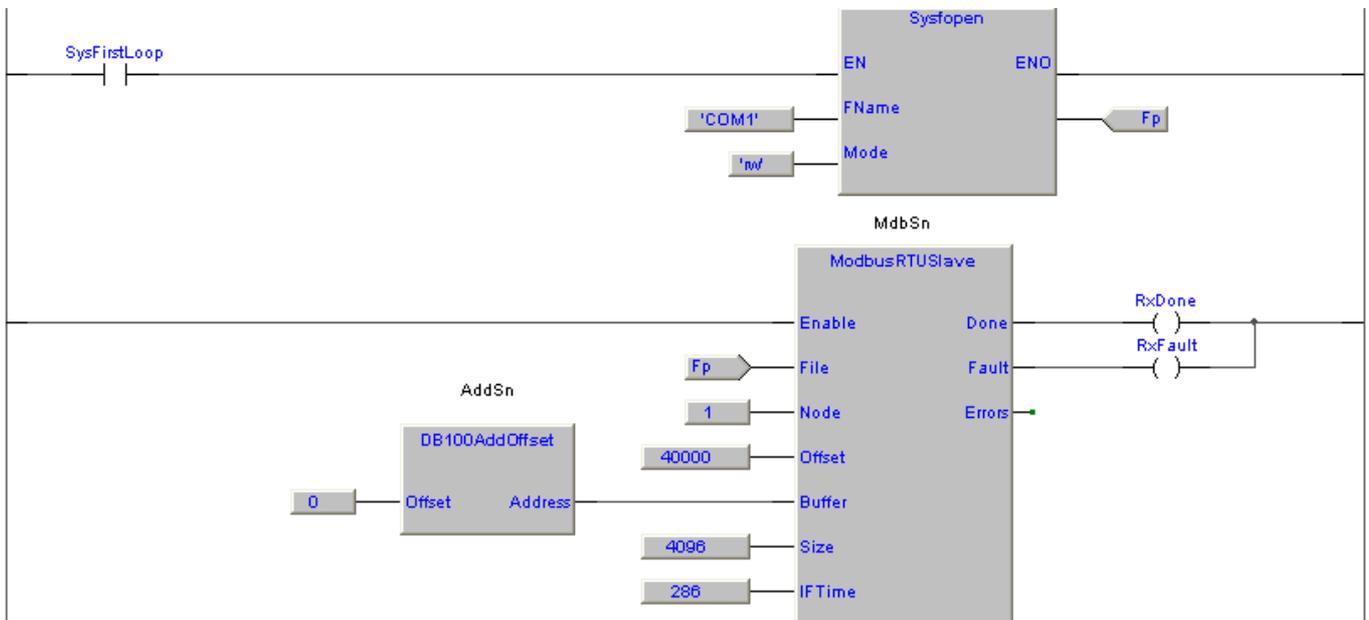
Esempi

Viene gestito il protocollo modbus slave su porta seriale **COM1**, si utilizza le impostazioni seriali di default **115200**, e **8**, **1**. I comandi modbus possono agire su tutta l'area della **DB100**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	RxDone	BOOL	Auto	No	FALSE	..	Modbus Rx command Ok
2	RxFault	BOOL	Auto	No	FALSE	..	Modbus Rx command fault
3	AddSn	DB100AddOffset	Auto	No	0	..	DB address FB
4	Fp	FILEP	Auto	No	0	..	File pointer
5	MdbSn	ModbusRTUSlave	Auto	No	0	..	Modbus RTU slave FB
6	SMode	SetSMode	Auto	No	0	..	Set serial mode FB

Esempio LD



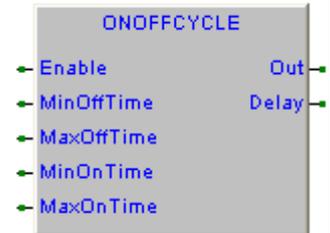
Type	Library	Version
FB	PLCUtyLib	SFR054A000

7.16.7 ONOFFCYCLE, on/off cycle with random times

Questo blocco funzione esegue la temporizzazione di un ciclo On/Off con tempi random di On e di Off definibili tra valori minimo e massimo.

Attivando il comando di **Enable** l'uscita **Out** esegue un lampeggio On/Off con tempi random compresi tra i valori minimo e massimo definiti. Disabilitando l'ingresso l'uscita Out si disattiva.

La variabile **Delay** ritorna il valore di ritardo attualmente attivo.



- Enable** (BOOL) Comando di abilitazione.
- MinOffTime** (UDINT) Valore minimo di tempo off comando (mS).
- MaxOffTime** (UDINT) Valore massimo di tempo off comando (mS).
- MinOnTime** (UDINT) Valore minimo di tempo on comando (mS).
- MaxOnTime** (UDINT) Valore massimo di tempo on comando (mS).
- Out** (BOOL) Stato comando On/Off in uscita.
- Delay** (UDINT) Valore di tempo attualmente in temporizzazione (mS).

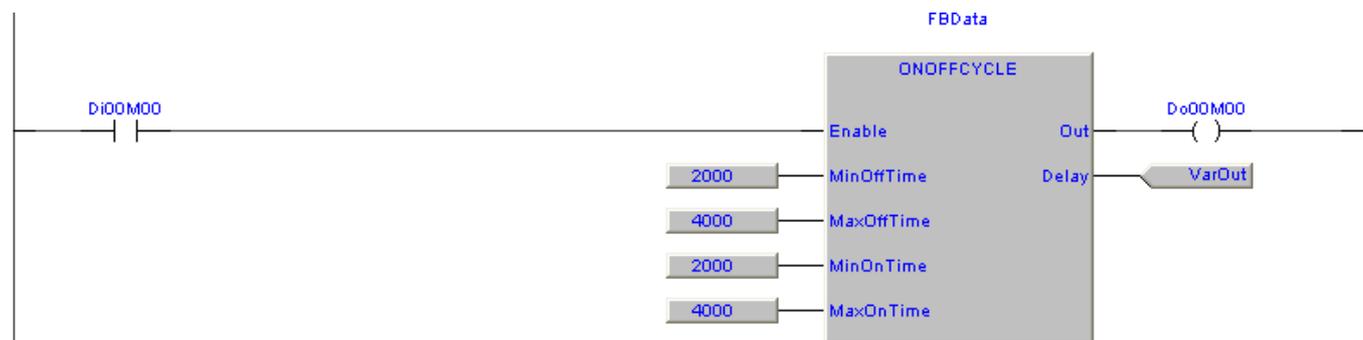
Esempi

Viene eseguito il lampeggio della uscita **Do00M00** con tempi random variabili tra i 2 ed i 4 secondi.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBData	ONOFFCYCLE	Auto	No	0	..	ONOFFCYCLE FB data
2	VarOut	UDINT	Auto	No	0	..	Variable output

Esempio LD (Ptp114a100)



Esempio IL (Ptp114a100)

```

CAL FBData (* Call the ONOFFCYCLE function block *)

LD Di00M00
ST FBData.Enable (* Transfer the digital input to Enable input *)

LD 2000
ST FBData.MinOffTime (* Set the minimum off time *)

LD 4000
ST FBData.MaxOffTime (* Set the maximum off time *)

LD 2000
ST FBData.MinOnTime (* Set the minimum on time *)

LD 4000
    
```

```
ST FBData.MaxOnTime (* Set the maximum on time *)

LD FBData.Out
ST Do00M00 (* Copy the Out value to logic output *)

LD FBData.Delay
ST VarOut (* The Delay time is copied to variable *)
```

Esempio ST (Ptp114a100)

```
FBData(); (* Call the ONOFFCYCLE function block *)

FBData.Enable:=Di00M00; (* Transfer the digital input to Enable input *)
FBData.MinOffTime:=2000; (* Set the minimum off time *)
FBData.MaxOffTime:=4000; (* Set the maximum off time *)
FBData.MinOnTime:=2000; (* Set the minimum on time *)
FBData.MaxOnTime:=4000; (* Set the maximum on time *)

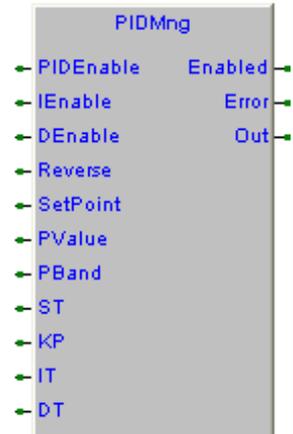
Do00M00:=FBData.Out; (* Copy the Out value to logic output *)
VarOut:=FBData.Delay; (* The Delay time is copied to variable *)
```

Type	Library	Version
FB	PLCUtyLib	SFR054A000

7.16.8 PIDMng, PID management

Questo blocco funzione esegue la regolazione PID. E' prevista la possibilità di abilitare singolarmente i vari tipi di azione (**P**)roporzionale, (**I**)ntegrativa, (**D**)erivativa.

Un comando di **Reverse** permette di invertire il segno del segnale in uscita **Out**.



- PIDEnable** (BOOL) Abilitazione regolazione PID, attivando l'ingresso si abilita la regolazione. Disattivando l'ingresso si azzerà il valore in uscita **Out**.
- IEnable** (BOOL) Abilitazione regolazione integrativa, attivando l'ingresso si abilita la regolazione integrativa.
- DEnable** (BOOL) Abilitazione regolazione derivativa, attivando l'ingresso si abilita la regolazione derivativa.
- Reverse** (BOOL) Inversione segno su valore in uscita **Out**.
- SetPoint** (REAL) Set point, il valore è espresso nell'unità di misura del processo da controllare.
- PValue** (REAL) Valore acquisito dal processo, il valore è espresso nell'unità di misura del processo da controllare.
- PBand** (REAL) Banda proporzionale, questo valore definisce il valore di errore oltre al quale la regolazione viene disabilitata forzando l'uscita **Out** al massimo **100%**. Il valore è espresso nell'unità di misura del processo da controllare.
- ST** (REAL) Tempo di scansione, occorre impostare il tempo in cui si desidera vengano eseguite le regolazioni integrativa e derivativa se abilitate, il valore è in **mS**.
- KP** (REAL) Costante proporzionale, si ricorda che più è elevato il valore più è pronta è la regolazione con un conseguente aumento del valore di overshoot. Il valore è un numero.
- IT** (REAL) Tempo integrativo, si ricorda che più è elevato il valore meno è veloce la regolazione integrativa a recuperare l'errore. Il valore è espresso in **Sec**.
- DT** (REAL) Tempo derivativo, si ricorda che più è elevato il valore più è veloce la regolazione derivativa a recuperare l'errore. Il valore è espresso in **Sec**.
- Enabled** (BOOL) Regolazione PID abilitata.
- Error** (BOOL) Errore nella esecuzione.
- OUT** (REAL) Valore di correzione in uscita dalla regolazione PID. Questo valore deve essere utilizzato per il comando del processo. Il valore è espresso in **%**. Il range è compreso tra 0 e 100 %.

Codici di errore

In caso di errore si attiva l'uscita **Error**, con [SysGetLastError](#) è possibile rilevare il codice di errore.

10012050 Non è stato definito valore di **ST**.

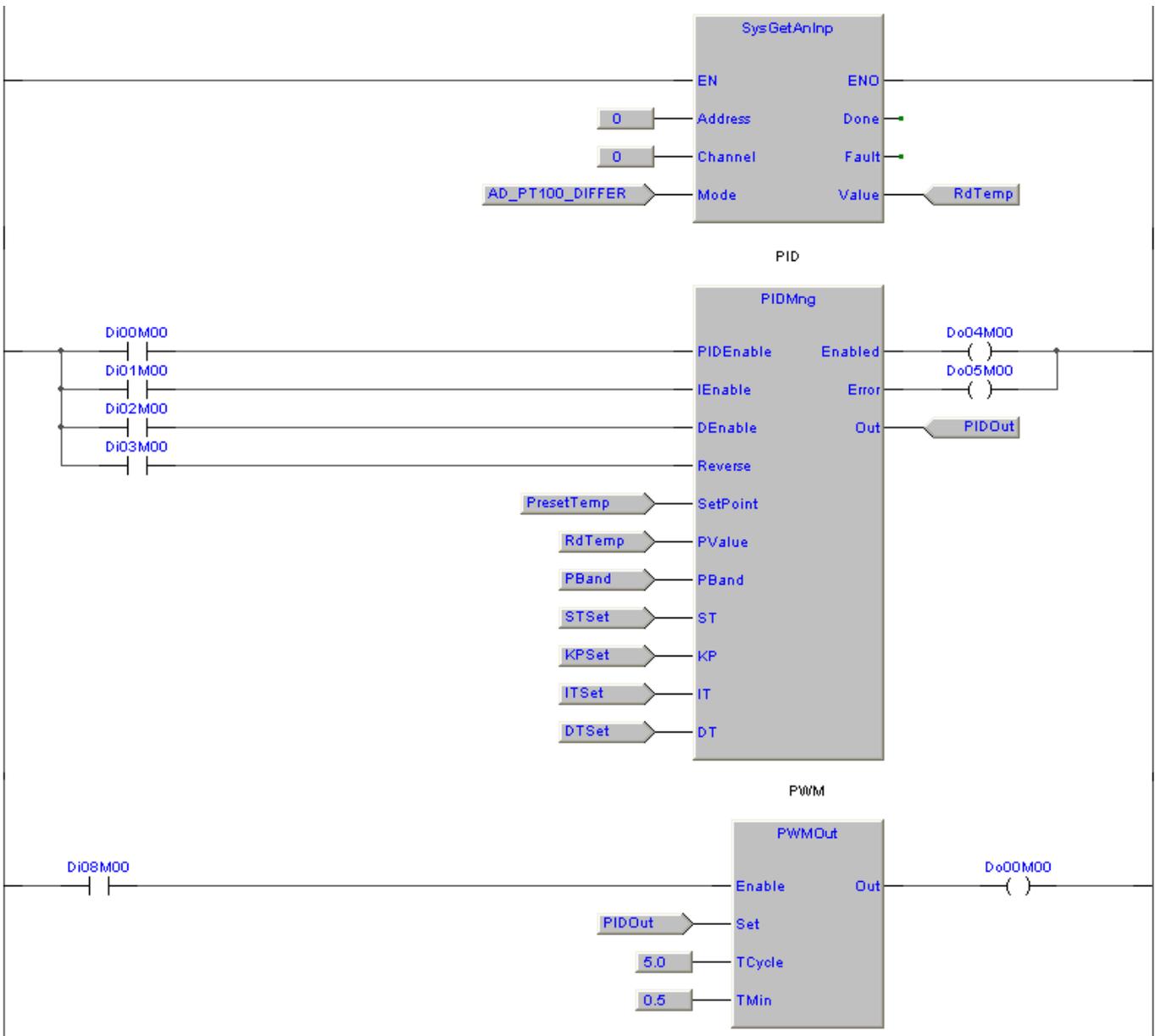
Esempi

Nell'esempio è gestita una regolazione di temperatura su di un termoriscaldatore. Viene acquisita la sonda di temperatura da una Pt100 e viene gestita una uscita PWM **Do00M00** per il comando. Le costanti del loop PID sono allocate in memoria backup e sono mantenute allo spegnimento, inoltre sono accessibili da modbus.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	RdTemp	REAL	%MD100.0	No	0	..	Read temperature (Degrees)
2	PreSetTemp	REAL	%MD100.2048	No	0	..	Set point temperature (Degrees)
3	PBand	REAL	%MD100.2052	No	0	..	Proportional band (Degrees)
4	STSet	REAL	%MD100.2056	No	0	..	Scansion time (mS)
5	KPSet	REAL	%MD100.2060	No	0	..	Proportional coefficient
6	ITSet	REAL	%MD100.2064	No	0	..	Integrative time (S)
7	DTSet	REAL	%MD100.2068	No	0	..	Derivative time (S)
8	TempRead	SysGetAnInp	Auto	No	0	..	Analog input
9	PWM	PWMOut	Auto	No	0	..	PWM management
10	PIDOut	REAL	Auto	No	0	..	PID output value (%)
11	PID	PIDMng	Auto	No	0	..	PID management

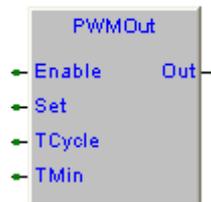
Esempio LD (Ptp114a100)



7.16.9 PWMOut, PWM output management

Type	Library	Version
FB	PLCUtyLib	SFR054A000

Questo blocco funzione esegue la gestione di una uscita PWM.



- Enable** (BOOL) Abilitazione gestione uscita PWM, attivando l'ingresso si abilita la gestione. Disattivando l'ingresso si azzerava l'uscita **Out**.
- SET** (REAL) Valore di set PWM, il valore è espresso in %.
- TCycle** (REAL) Tempo di di ciclo PWM, il valore è espresso in **S**.
- TMin** (REAL) Tempo minimo comando uscita **Out**, il valore è espresso in **S**.
- Out** (BOOL) Uscita PWM.

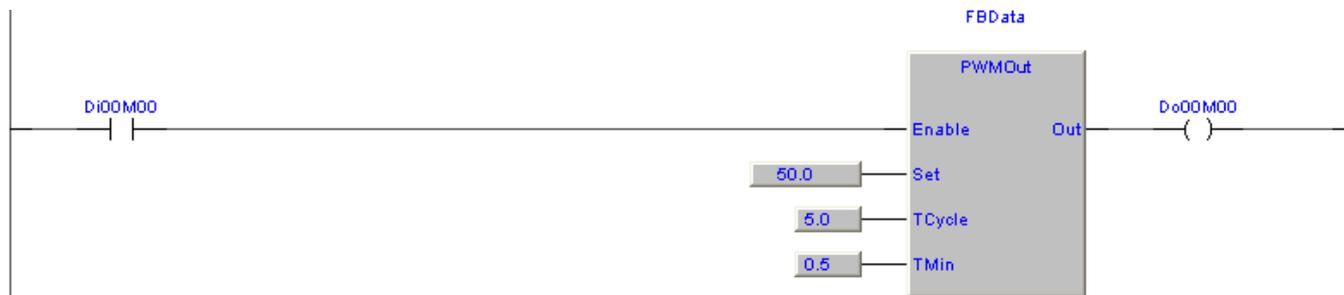
Esempi

Nell'esempio è gestita una uscita PWM definendo un tempo di ciclo di 5 secondi con un tempo minimo di 0.5 secondi. Impostando come set point il valore 50% attivando l'ingresso **Di00M00** avremo che l'uscita **Do00M00** sarà attivata per 2.5 secondi e disattivata per 2,5 secondi.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBData	PWMOut	Auto	No	0	..	FB PWMOut data

Esempio LD (Ptp114a100)



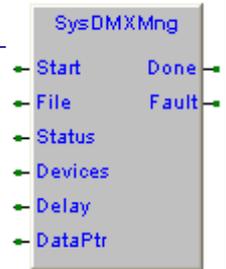
Type	Library	Version
FB	Embedd	3.0

7.16.10 SysDMXMng, DMX management

Questo blocco funzione esegue la gestione del protocollo DMX, questo è un blocco funzione protetto per utilizzarlo occorre richiedere il codice di protezione, vedi [protezione funzioni e blocchi funzione](#). E' comunque possibile utilizzarlo liberamente in modo test per 15 Min.

Attivando l'ingresso di **Start** viene inviato sulla porta seriale identificata da **File** un frame DMX che inizia con il valore di **Status** e segue con il valore di preset dei dispositivi definiti da **Devices**. Il valore di preset dei vari dispositivi deve essere caricato in un array di dati il cui indirizzo è passato in **DataPtr**. Mantenendo sempre attivo l'ingresso **Start** verranno inviati consecutivamente frames DMX.

Al termine dell'invio del comando DMX si attiverà per un loop di programma l'uscita **Done**.



- Start** (BOOL) Comando di invio frame DMX su porta seriale, si resetta automaticamente all'invio del frame.
- File** (FILEP) Flusso dati **stream** ritornato dalla funzione **Sysfopen**.
- Status** (USINT) Valore byte di stato inviato nel protocollo DMX prima dei byte di preset dispositivi.
- Devices** (UINT) Numero di dispositivi connessi al bus DMX.
- Delay** (UINT) Tempo di pausa trasmissione frames DMX (mSec)
- DataPtr** (@USINT) Pointer all'array dati valori di preset dispositivi DMX.
- Done** (BOOL) Attivo per un loop al termine dell'invio frame DMX del comando
- Fault** (BOOL) Attivo in caso di errore nella gestione.

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con [SysGetLastError](#) è possibile rilevare il codice di errore.

- 9979050 Errore allocazione blocco funzione.
- 9979060 Terminato spazio memoria rilocabile, non è possibile eseguire l"FB.
- 9979070 Errore versione blocco funzione.
- 9979085 FB protetta, terminato tempo funzionamento in modo demo.
- 9979200 Protocollo DMX non supportato dal dispositivo definito in **File**.

Esempi

Dovendo gestire puntatori a memoria è preferibile utilizzare il blocco funzione all'interno di un programma ST, nell'esempio viene attivato il protocollo DMX sulla porta seriale **COM1**. Vengono gestiti 5 dispositivi con indirizzi da 1 a 5. Il frame DMX è continuamente inviato ai dispositivi.

Attivando l'ingresso digitale **Di00M00** viene impostato il valore 0, su tutti i dispositivi. Attivando l'ingresso digitale **Di01M00** su dispositivo 1 viene impostato il valore 10, sul 2 il valore 20, e così via fino al quinto in cui viene impostato il valore 50.

Definizione variabili

	Name	Type	Address	Array	Initvalue	Attribute	Description
1	FBDMX	SysDMXMng	Auto	No	0	..	FB gestione protocollo DMX
2	DMXData	USINT	Auto	[0..4]	5(0)	..	DMX data
3	DiPls	R_TRIG	Auto	[0..1]	0,0	..	Pulse su ingresso

Esempio ST

```
(* ----- *)
(* ESEGUO APERTURA PORTA SERIALE *)
(* ----- *)
(* Here the COM1 port is opened in read/write. *)

  IF (FBDMX.File = NULL) THEN
    FBDMX.File:=Sysfopen('COM1', 'rw'); (* Port COM1 file pointer *)
  END_IF;

(* ----- *)
(* ESEGUO ATTIVAZIONE COMANDI *)
(* ----- *)
(* Attivazione comandi su input Di00M00. *)

  DiPls[0] (CLK:=Di00M00);
  IF (DiPls[0].Q) THEN
    DMXData[0]:=0; (* Preset dispositivo con indirizzo 1 *)
    DMXData[1]:=0; (* Preset dispositivo con indirizzo 2 *)
    DMXData[2]:=0; (* Preset dispositivo con indirizzo 3 *)
    DMXData[3]:=0; (* Preset dispositivo con indirizzo 4 *)
    DMXData[4]:=0; (* Preset dispositivo con indirizzo 5 *)
    FBDMX.Start:=TRUE; (* Start *)
  END_IF;

(* Attivazione comandi su input Di01M00. *)

  DiPls[1] (CLK:=Di01M00);
  IF (DiPls[1].Q) THEN
    DMXData[0]:=10; (* Preset dispositivo con indirizzo 1 *)
    DMXData[1]:=20; (* Preset dispositivo con indirizzo 2 *)
    DMXData[2]:=30; (* Preset dispositivo con indirizzo 3 *)
    DMXData[3]:=40; (* Preset dispositivo con indirizzo 4 *)
    DMXData[4]:=50; (* Preset dispositivo con indirizzo 5 *)
    FBDMX.Start:=TRUE; (* Start *)
  END_IF;

(* ----- *)
(* ESEGUO GESTIONE PROTOCOLLO DMX *)
(* ----- *)
(* Gestione protocollo DMX. *)

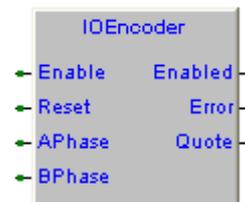
  FBDMX.Status:=0; (* Status byte *)
  FBDMX.Devices:=5; (* Number of devices *)
  FBDMX.Delay:=0; (* Interframe delay (mSec) *)
  FBDMX.DataPtr:=ADR(DMXData); (* Data array pointer *)
  FBDMX(); (* FB gestione protocollo DMX *)
  IF (FBDMX.Done) THEN FBDMX.Start:=FALSE; END_IF;

(* [End of file] *)
```

7.16.11 IOEncoder, incremental encoder over I/O

Type	Library	Version
FB	PLCUtyLib	SFR054A400

Questo blocco funzione esegue la lettura di un encoder incrementale connesso agli ingressi logici. Basta appoggiare sui due ingressi **APhase** e **BPhase** del blocco funzione i due ingressi di acquisizione del canale **A** e del canale **B** di un encoder incrementale. Il blocco funzione esegue la quadratura dei segnali, il controllo della direzione di rotazione e gestisce il valore di **Quote** in uscita.



La quadratura dei segnali esegue la moltiplicazione per 4 delle tacche encoder quindi il valore di **Quote** al termine di un giro completo dell'encoder è pari al numero di tacche encoder moltiplicato per 4.

- Enable** (BOOL) Abilitazione gestione conteggio encoder.
- Reset** (BOOL) Comando di reset quota encoder. Attivando l'ingresso si azzerava il valore di **Quote**.
- APhase** (BOOL) Ingresso canale **A** encoder.
- BPhase** (BOOL) Ingresso canale **B** encoder.
- Enabled** (BOOL) Conteggio encoder abilitato.
- Error** (BOOL) Attivo per un loop su errore acquisizione encoder. Si attiva se la frequenza di ingresso dei segnali encoder è maggiore rispetto al tempo di esecuzione del blocco funzione.
- Quote** (UDINT) Valore di quota encoder espresso in impulsi. Numero tacche giro encoder moltiplicato per 4.

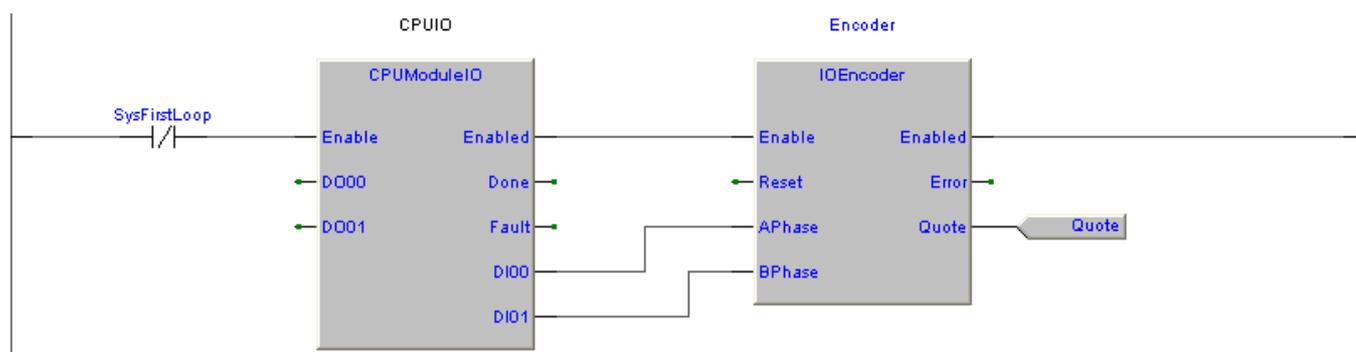
Esempi

Nell'esempio è gestita l'acquisizione di un encoder incrementale connesso agli ingressi del modulo CPU. Ruotando l'encoder il valore di **Quote** verrà incrementato se rotazione CW oppure decrementato se rotazione CCW, del numero di tacche giro moltiplicato per 4.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Encoder	IOEncoder	Auto	No	0	..	IOEncoder function block
2	CPUIO	CPUModuleIO	Auto	No	0	..	CPUModuleIO function block
3	Quote	UDINT	Auto	No	0	..	Encoder quote

Esempio LD (Ptp114a200)



7.16.12 GetISO1155Crc, calculate CRC according ISO1155

Type	Library	Version
Function	PLCUtyLib	SFR054A400

Questa funzione esegue il calcolo del CRC **Cyclic Redundancy Check**, (Controllo Ciclico di Ridondanza) su di un'area dati. Il calcolo è effettuato secondo le specifiche **ISO 1155**.

Occorre passare alla funzione l'indirizzo del buffer di memoria **Buffer** ed il numero di bytes **ByteNr** su cui eseguire il calcolo del CRC.



Buffer (@USINT) Indirizzo dell'area di memoria su cui eseguire il calcolo del CRC.

ByteNr (UINT) Numero di bytes su cui eseguire il calcolo del CRC a partire dall'indirizzo definito in **Buffer**.

CRC (UINT) Valore CRC calcolato.

Esempi

Viene calcolato il CRC di una richiesta di lettura del registro 1.8.1 da un contatore di energia elettrica secondo lo standard IEC 62056-2. Il frame di richiesta è '**<SOH>R1<STX>1.8.1()<ETX><CRC>**'.

Il valore del CRC ritornato in **CRCValue** è 16#5A (90 decimale).

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	DataFrame	STRING	Auto	[16]		..	Data frame
2	CRCValue	UINT	Auto	No	0	..	CRC Value

Esempio ST

```
(* ***** *)
(* IEC1155 CRC CALCULATION *)
(* ***** *)
(* Register read command '<SOH>R1<STX>1.8.1()<ETX><CRC>' . *)

DataFrame:='$01R1$021.8.1()$03'; (* Data frame *)
CRCValue:=GetISO1155Crc(ADR(DataFrame), 12); (* CRC Value *)
```

Type	Library	Version
FB	PLCUtyLib	SFR054A500

7.16.13 IODataExchange, exchange data by using logic I/O

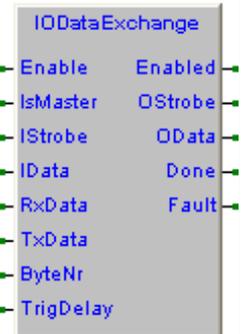
Questo blocco funzione permette lo scambio di dati tra due sistemi, uno master ed uno slave, utilizzando una connessione tramite I/O logici. Sono utilizzati due ingressi e due uscite digitali per ogni sistema, è possibile definire il numero di bytes di dati da scambiarsi.

Occorre connettere l'uscita digitale **OStrobe** di un sistema con l'ingresso digitale **IStrobe** dell'altro sistema e l'uscita **OData** con l'ingresso digitale **IData** dell'altro.

Il trasferimento dati è bidirezionale, i dati presenti nel buffer **TxData** di un sistema sono trasferiti nel buffer **RxData** dell'altro sistema e viceversa, per il numero di bytes definito in **ByteNr**. La comunicazione è verificata mediante l'invio di un CRC secondo lo standard ISO 1155.

Ad ogni **fine** trasferimento dati si attiva per un loop l'uscita **Done**, sulla sua attivazione occorre provvedere a trasferire i dati da trasmettere nel buffer di trasmissione e leggere i dati dal buffer di ricezione.

In caso di **errore** nella comunicazione si attiva per un loop l'uscita **Fault**, ed i due sistemi si risincronizzano per riprendere una nuova trasmissione.



- Enable** (BOOL) Abilitazione gestione comunicazione.
- IsMaster** (BOOL) **TRUE**: Modo master, **FALSE**: Modo slave.
- IStrobe** (BOOL) Occorre appoggiare l'ingresso digitale di strobe.
- IData** (BOOL) Occorre appoggiare l'ingresso digitale di dato.
- RxData** (UDINT) Indirizzo buffer dati ricevuti.
- TxData** (UDINT) Indirizzo buffer dati da trasmettere.
- ByteNr** (USINT) Numero bytes da scambiare con altro sistema (Da 1 a 30).
- TrigDelay** (UINT) Tempo attesa tra uscita dato **OData** ed uscita strobe **OStrobe** (Da 0 a 30 mS).
- ByteNr** (USINT) Numero bytes da scambiare con altro sistema.
- OStrobe** (BOOL) Da appoggiare su uscita digitale di strobe.
- OData** (BOOL) Da appoggiare su uscita digitale di dato.
- Done** (BOOL) Si attiva per un loop al termine dello scambio dati.
- Fault** (BOOL) Si attiva per un loop in caso di errore su scambio dati.

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con [SysGetLastError](#) è possibile rilevare il codice di errore.

- 10011080 Errore definizione valore **ByteNr**.
- 10011082 Errore definizione valore **TrigDelay**.
- 10011100~1 Timeout attesa attivazione segnale **IStrobe**.
- 10011110~1 Timeout attesa disattivazione segnale **IStrobe**.
- 10011200~1 Errore CRC dati ricevuti.

Esempi

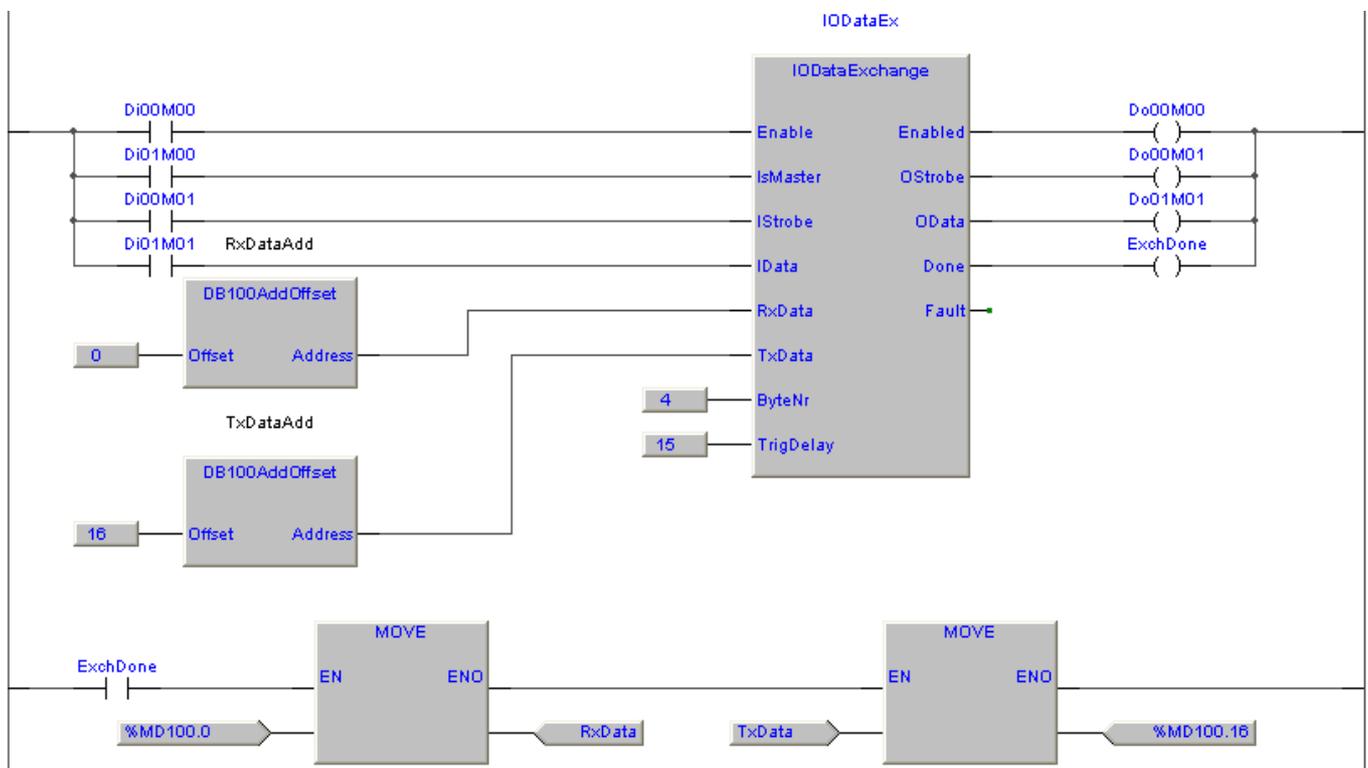
Utilizzando due sistemi attivando modo master (**Di01M00** attivo) su di un sistema e modo slave (**Di01M00** disattivo) sull'altro, è possibile eseguire lo scambio di 4 bytes di memoria tra i sistemi. I 4 bytes allocati ad indirizzo **MD100.0** di un sistema saranno trasferiti su 4 bytes allocati ad indirizzo **MD100.16** dell'altro sistema.

Al termine del trasferimento, i dati ricevuti dalla memoria **MD100.0** sono trasferiti nella variabile **RxDData**, mentre la variabile **TxDData** è trasferita in memoria **MD100.16**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	ExchDone	BOOL	Auto	No	FALSE	..	Data exchange done
2	RxDData	UDINT	Auto	No	0	..	Received data
3	TxDData	UDINT	Auto	No	0	..	Transmit data
4	RxDDataAdd	DB100AddOffset	Auto	No	0	..	RxDData address
5	TxDDataAdd	DB100AddOffset	Auto	No	0	..	TxDData address
6	IODataEx	IODataExchange	Auto	No	0	..	FB IODataExchange data

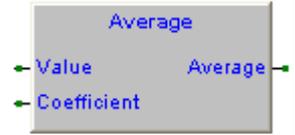
Esempio LD (Ptp121A000)



7.16.14 Average, value average

Type	Library	Version
FB	PLCUtyLib	SFR054A800

Questo blocco funzione esegue la media su di un valore. L'azione di media è definita da un parametro **Coefficient**, maggiore è il valore del parametro e maggiore sarà l'azione di media sul valore in uscita **Average**.



Value (REAL) Valore su cui effettuare l'azione di media

Coefficient (REAL) Valore del coefficiente di media.

Average (REAL) Valore mediato in uscita.

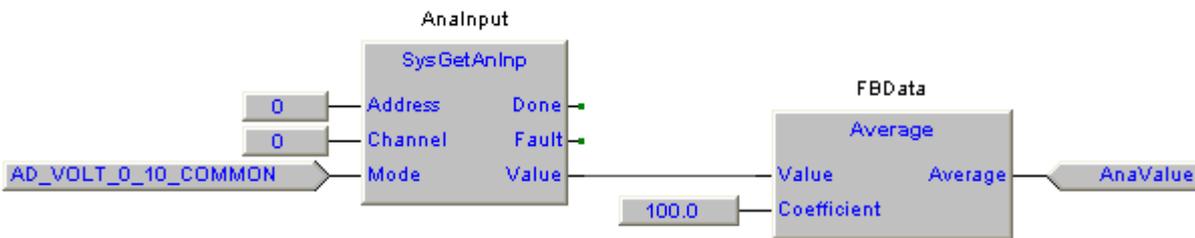
Esempi

Viene eseguita una acquisizione analogica dall'ingresso **0** del modulo con indirizzo **0**, in modo 0÷10 volt. Il valore acquisito viene mediato e poi trasferito nella variabile **AnaValue**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBDData	Average	Auto	No	0	..	FB average data
2	AnaInput	SysGetAnInp	Auto	No	0	..	FB Analog input data
3	AnaValue	REAL	Auto	No	0	..	Analog input value

Esempio LD (Ptp114A500)



7.17 Protocollo DLMS, o IEC 62056-21

Con il nuovo protocollo standard DLMS (Device Language Message Specification), la comunicazione con i sistemi di metering viene immensamente semplificata. Questo protocollo ha una struttura orientata agli **Objetti** che rende possibile leggere con la stessa identica modalità dati applicativi provenienti da contatori di diversi costruttori.

Lo standard IEC 61107 o IEC 62056-21 è uno standard internazionale che descrive il protocollo DLMS per la lettura da parte di un computer dei dati da contatori tariffari di energia elettrica, acqua e gas.

Il protocollo prevede una fase di **Sign-On** con il contatore durante la quale occorre fornire un codice di accesso (Solitamente il numero di serie del contatore), ed il contatore fornisce una password in uscita che può essere utilizzata per criptare i dati.

Terminata questa fase è possibile richiedere al contatore il valore dei suoi registri utilizzando i codici di identificazione **OBIS** a 5 caratteri (IEC 62056-61).

Per la famiglia SlimLine abbiamo sviluppato un apposito blocco funzione che automatizza tutte le operazioni, occorre passare il numero di serie del contatore ed il codice OBIS del registro da leggere. Il blocco funzione esegue il Sign-On sul contatore e ritorna il valore del registro indicato.

Interfaccia con il contatore

Per interfacciarsi con il contatore è possibile utilizzare un apposito accoppiatore ottico che si appoggia alla finestra di lettura del contatore e si connette ad una delle porte seriali dello SlimLine.

Oppure nel caso di contatori predisposti con l'uscita RS485 è possibile connettersi direttamente alla porta RS485 dello SlimLine.

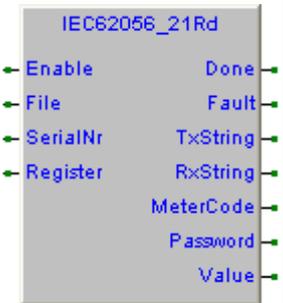


Type	Library	Version
FB	PLCUtyLib	SFR054A700

7.17.1 IEC62056_21Rd, IEC62056-21 protocol read

Questo blocco funzione esegue la gestione della lettura di registri da sistemi di metering utilizzando il protocollo IEC62056-21. Questo è un blocco funzione protetto per utilizzarlo occorre richiedere il codice di protezione, vedi [protezione funzioni e blocchi funzione](#). E' comunque possibile utilizzarlo liberamente in modo test per 30 Min.

Occorre in **SerialNr** definire il numero seriale del contatore (Utilizzato come chiave di accesso alla lettura) e in **Register** l'indirizzo del registro da leggere secondo la normativa OBIS. Se la lettura ha esito positivo viene attivata l'uscita **Done** e le variabili **MeterCode**, **Password** e **Value** sono valorizzate con i dati letti dal contatore.



Enable (BOOL)	Attivando l'ingresso viene gestita la lettura del contatore.
File (FILEP)	Pointer al file della risorsa così come ritornato dalla funzione Sysfopen .
SerialNr (STRING[16])	Numero di serie del contatore, viene utilizzato come chiave di accesso.
Register (STRING[16])	Indirizzo registro da leggere secondo la codifica OBIS.
Done (BOOL)	Viene attivato per un loop al termine della acquisizione.
Fault (BOOL)	Viene attivato per un loop in caso di errore nella sequenza di acquisizione.
TxString (STRING[32])	Contiene la stringa di comando inviata al contatore, può essere utilizzato in debug per verificare i comandi inviati.
RxString (STRING[32])	Contiene la stringa di risposta ritornata dal contatore, può essere utilizzato in debug per verificare le risposte ricevute.
MeterCode (STRING[32])	Contiene stringa con il codice del contatore acquisita durante fase di acceso (Sign-on).
Password (UDINT)	Contiene valore password acquisita dal contatore durante fase di acceso (Sign-on).
Value (STRING[32])	Contiene stringa con valore registro richiesto acquisita dal contatore.

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con [SysGetLastError](#) è possibile rilevare il codice di errore.

- 10016010 Valore di **File** non definito.
- 10016020 FB protetta, terminato tempo funzionamento in modo demo.
- 10016050 Timeout esecuzione.
- 10016070 Errore case gestione.
- 10016100~1 Errore ricezione tipo da contatore.
- 10016110~2 Errore ricezione password da contatore.
- 10016120~2 Errore ricezione valore parametro da contatore.
- 10016200 Overflow ricezione stringa da contatore.

Esempi

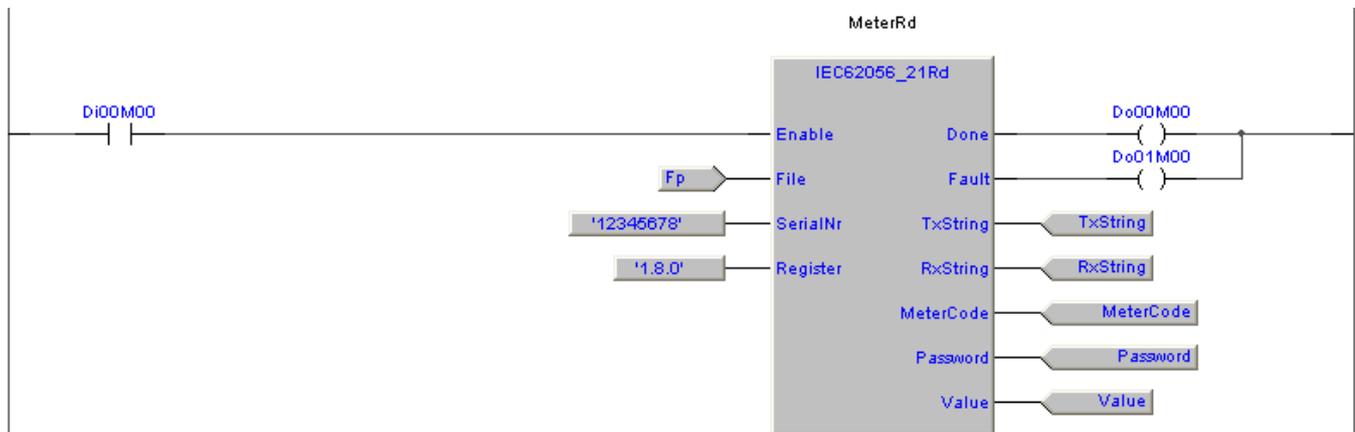
E' disponibile un programma di esempio Ptp122*000 che esegue la lettura da un contatore di energia di 3 registri. A titolo indicativo diamo un esempio di massima per la lettura di un registro.

Su attivazione dell'ingresso digitale **Di00M00** viene eseguita la lettura del registro **1.8.0** (Potenza in Kw). Se la lettura ha esito positivo si attiva per un loop l'uscita digitale **Di00M00**. Le variabili **MeterCode**, **Password** e **Value**, saranno valorizzate con i valori letti dal contatore.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer
2	MeterRd	IEC62056_21	Auto	No	0	..	IEC62056_21Rd FB data
3	TxString	STRING	Auto	[32]		..	Tx data string
4	RxString	STRING	Auto	[32]		..	Rx data string
5	MeterCode	STRING	Auto	[32]		..	Meter code
6	Password	UDINT	Auto	No	0	..	Meter password
7	Value	STRING	Auto	[32]		..	Variable read value

Esempio LD (Ptp114A000)



7.18 Funzioni ed FB gestione modem (eModemLib)

Le funzioni ed i blocchi funzione per la gestione del modem utilizzano un modem GSM connesso ad un terminale di I/O del sistema (Tipicamente è utilizzata una porta seriale). Nel modem deve essere inserita una tessera SIM **non protetta dal codice PIN**.

Per utilizzare la gestione del modem occorre importare la libreria **SFR057**00** nel proprio progetto, si rimanda al capitolo relativo all'[import delle librerie](#) per ulteriori informazioni in merito.

Nella descrizioni successive si fa riferimento alle seguenti definizioni generali.

Numero di telefono

Il numero di telefono consiste in una stringa lunga da 10 a 16 caratteri numerici conforme al seguente formato:

Prefisso internazionale senza lo zero davanti (es. +39 per Italia, +49 per Germania, +44 per Gran Bretagna ecc.)

Codice dell'operatore mobile (es. 338, 320, 347, ecc.)

Numero di telefono (es. 7589951)

Esempio: +393337589951,+3933812345,+49172123456

Messaggio SMS

Un messaggio SMS può essere lungo fino a 160 caratteri alfanumerici facenti parte del seguente set:

A...Z, a...z, 0...9, Spazio bianco, sono da evitare tutti gli altri caratteri.

Type	Library	Version
FB	eModemLib	SFR057A100

7.18.1 ModemCore, modem core management

Questo blocco funzione gestisce un modem connesso al dispositivo di I/O definito in **File**, questo è un blocco funzione protetto per utilizzarlo occorre richiedere il codice di protezione, vedi [protezione funzioni e blocchi funzione](#). E' comunque possibile utilizzarlo liberamente in modo test per 30 Min.

L'FB gestisce il dialogo con il modem, ne esegue l'inizializzazione e ne controlla lo stato, controlla se il modem è connesso alla rete GSM e ritorna l'operatore di rete **Operator** ed il livello del segnale **Rssi**. Nel caso in cui il modem si sganci dalla rete l'FB provvede al suo riaggancio automatico.

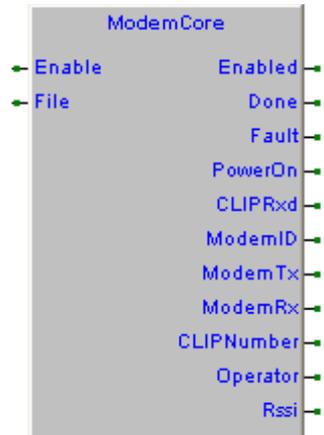
L'uscita **Done** si attiva se il modem è correttamente inizializzato, mentre l'uscita **Fault** si attiva per un loop di programma in caso di errori di gestione.

E' previsto un comando **PowerOn** per la gestione della alimentazione del modem, in questo modo l'FB può spegnere e riaccendere il modem in caso riscontri una non funzionalità dello stesso.

L'FB ritorna un **ModemID** che deve essere passato alle FB collegate (Esempio invio SMS, ricezione SMS, ecc.).

Le uscite **ModemTx** e **ModemRx** riportano i comandi inviati e ricevuti dal modem, in questo modo è possibile visualizzare in debug la comunicazione con il modem permettendo di visualizzare eventuali errori nell'interfaccia con il modem.

Su ricezione chiamata telefonica viene rilevato il CLIP del chiamante che è ritornato in uscita **CLIPNumber**, contemporaneamente ad ogni squillo del telefono si attiva per un loop di programma l'uscita **CLIPRx**.



Enable (BOOL)	Abilitazione blocco funzione, attivandolo viene gestito il modem.
File (FILEP)	Flusso dati stream ritornato dalla funzione Sysfopen .
Enabled (BOOL)	Blocco funzione abilitato.
Done (BOOL)	Modem correttamente inizializzato e funzionante.
Fault (BOOL)	Attivo per un loop di programma se errore gestione modem.
PowerOn (BOOL)	Comando di gestione uscita alimentazione modem.
CLIPRx (BOOL)	Attivo per un loop di programma ad ogni ricezione CLIP (Tipicamente ad ogni RING del modem).
ModemID (UDINT)	ID modem da passare alle FB collegate (Esempio ModemSMSsend , ModemSMSreceive , ecc.).
ModemTx (STRING[256])	Contiene la stringa di comando inviata al modem, può essere utilizzato in debug per verificare i comandi inviati al modem.
ModemRx (STRING[256])	Contiene la stringa di risposta ritornata dal modem, può essere utilizzato in debug per verificare le risposte ricevute dal modem.
CLIPNumber (STRING[16])	Contiene la stringa con il numero di CLIP ricevuto.
Operator (STRING[16])	Contiene la stringa con il nome dell'operatore telefonico.
Rssi (USINT)	Valore potenza segnale radio.

Codici di errore

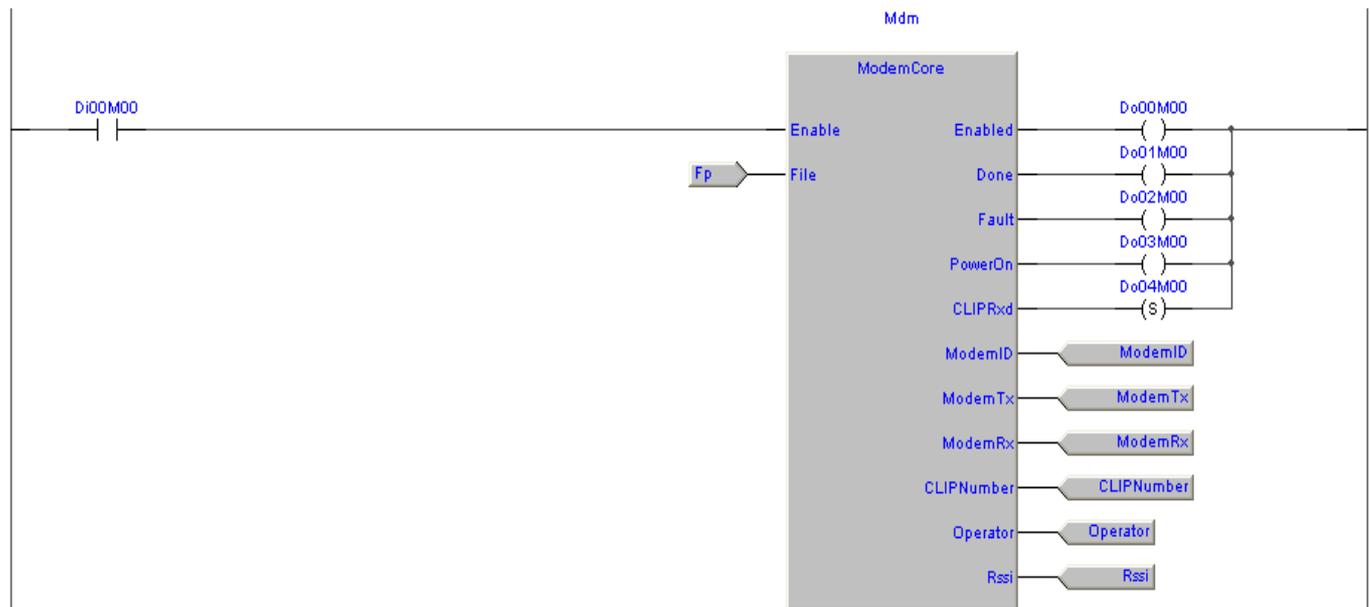
In caso di errore si attiva l'uscita **Fault**, con [SysGetLastError](#) è possibile rilevare il codice di errore.

Codice	Descrizione
10002010	Valore di File non definito.
10002020	FB protetta, terminato tempo funzionamento in modo demo.
10002050	Timeout esecuzione.
10002100~9	Errore ricezione CLIP.
10002150~9	Errore nelle sequenze power on del modem.
10002200~1	Errore nelle sequenze di controllo del modem.
10002210~7	Errore nella acquisizione dell'operatore telefonico.
10002220~2	Errore nella acquisizione del livello del segnale.
10002300~4	Errore nell'invio messaggio SMS.
10002350~8	Errore nella ricezione del messaggio SMS.

Esempi

Nell'esempio è gestito un modem connesso al terminale di I/O definito nella variabile **Fp**, per la definizione delle variabili e per una migliore comprensione del funzionamento si rimanda agli esempi successivi.

Esempio LD



7.18.2 ModemSMSReceive, receive a SMS message

Type	Library	Version
FB	eModemLib	SFR057A100

Questo blocco funzione esegue la ricezione di un messaggio SMS, si collega al blocco funzione di gestione modem **ModemCore**, occorre passare il **ModemID** in uscita dal blocco funzione di gestione modem.



Alla ricezione di un messaggio SMS si attiva per un loop di programma l'uscita **Done**, sull'uscita **SMSText** viene ritornato il messaggio ricevuto, all'uscita **CLIPNumber** della FB **ModemCore** è ritornato il numero di telefono da cui il messaggio è stato ricevuto. Il testo del messaggio ricevuto rimane presente in uscita sino alla ricezione di un altro messaggio.

- Enable** (BOOL) Abilita la ricezione dei messaggi SMS.
- ModemID** (UDINT) ID modem fornito in uscita dalla **ModemCore**.
- Done** (BOOL) Attivo per un loop se ricevuto messaggio SMS.
- Fault** (BOOL) Attivo per un loop se errore.
- Text** (STRING[160]) Testo del messaggio SMS ricevuto.

Codici di errore

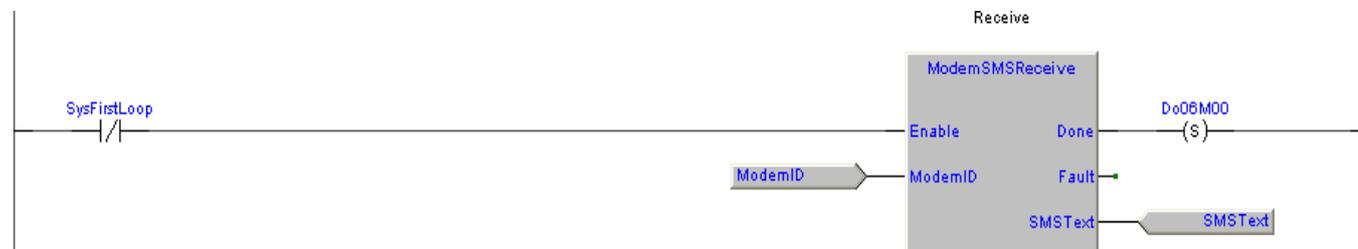
In caso di errore si attiva l'uscita **Fault**, con **SysGetLastError** è possibile rilevare il codice di errore.

- 10003010 **ModemID** non definito.
- 10003020 **ModemID** non corretto.

Esempi

Nell'esempio è gestita la ricezione di un messaggio SMS dal modem definito nella variabile **ModemID**, per la definizione delle variabili e per una migliore comprensione del funzionamento si rimanda agli esempi successivi.

Esempio LD



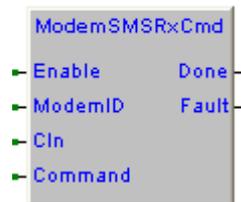
7.18.3 ModemSMSRxCmd, receive a SMS command

Type	Library	Version
FB	eModemLib	SFR057A100

Questo blocco funzione esegue la ricezione di un comando tramite un messaggio SMS, si collega al blocco funzione di gestione modem **ModemCore**, occorre passare il **ModemID** in uscita dal blocco funzione di gestione modem.

Alla ricezione di un messaggio SMS se nel testo del messaggio è presente la stringa definita in **Command**, si attiva per un loop di programma l'uscita **Done**, all'uscita **CLIPNumber** della FB **ModemCore** è ritornato il numero di telefono da cui il messaggio è stato ricevuto.

Attivando **Cin** il controllo sulla stringa definita in **Command** verrà fatto non considerando il case (Maiuscolo/minuscolo) dei caratteri.



- Enable** (BOOL) Abilita la ricezione del comando.
- ModemID** (UDINT) ID modem fornito in uscita dalla **ModemCore**.
- Cin** (BOOL) Se attivo, controllo di **Command** non considerando case (Maiuscolo/minuscolo) dei caratteri.
- Command** (STRING[32]) Testo comando da eseguire.
- Done** (BOOL) Attivo per un loop se ricevuto messaggio SMS contenente il testo indicato in **Command**.
- Fault** (BOOL) Attivo per un loop se errore.

Codici di errore

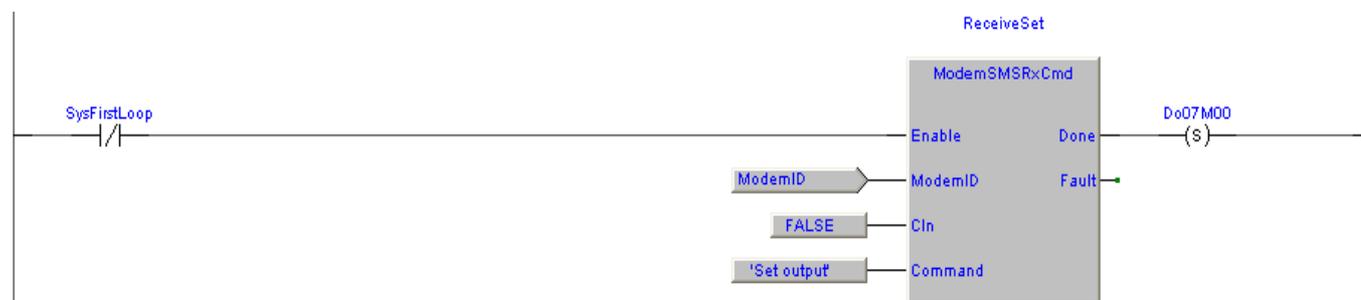
In caso di errore si attiva l'uscita **Fault**, con [SysGetLastError](#) è possibile rilevare il codice di errore.

- 10004010 **ModemID** non definito.
- 10004020 **ModemID** non corretto.

Esempi

Nell'esempio è gestita la ricezione di un messaggio SMS dal modem definito nella variabile **ModemID**, per la definizione delle variabili e per una migliore comprensione del funzionamento si rimanda agli esempi successivi.

Esempio LD

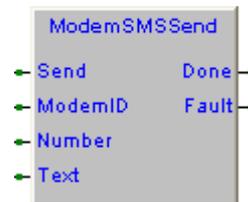


7.18.4 ModemSMSSend, send a SMS message

Type	Library	Version
FB	eModemLib	SFR057A100

Questo blocco funzione esegue l'invio di un messaggio SMS, si collega al blocco funzione di gestione modem **ModemCore**, occorre passare alla FB il **ModemID** in uscita dal blocco funzione di gestione modem.

Su fronte attivazione ingresso di **Send** viene prenotato l'invio del messaggio, non appena sarà possibile il messaggio definito in **Text** verrà inviato al numero definito in **Number**. Terminato l'invio verrà attivata per un loop di programma l'uscita **Done**.



- Send** (BOOL) Sul fronte di attivazione comanda l'invio del messaggio SMS. **Attenzione!** Il messaggio sarà inviato non appena il modem è libero per l'invio.
- ModemID** (UDINT) ID modem fornito in uscita dalla **ModemCore**.
- Number** (STRING[16]) Numero di telefono a cui eseguire l'invio del messaggio.
- Text** (STRING[160]) Testo messaggio da inviare.
- Done** (BOOL) Attivo per un loop al termine dell'invio del messaggio SMS.
- Fault** (BOOL) Attivo per un loop se errore invio messaggio SMS.

Codici di errore

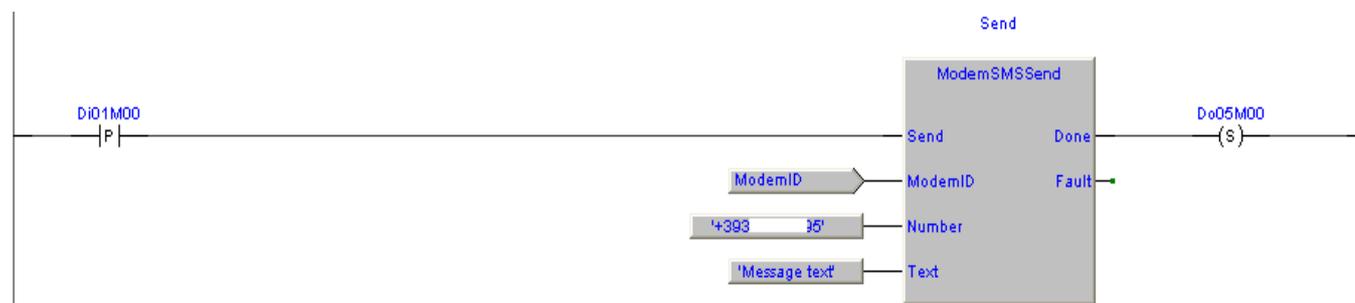
In caso di errore si attiva l'uscita **Fault**, con **SysGetLastError** è possibile rilevare il codice di errore.

- 10005010 **ModemID** non definito.
- 10005020 **ModemID** non corretto.

Esempi

Nell'esempio è gestito l'invio di un messaggio SMS sul modem definito nella variabile **ModemID**, per la definizione delle variabili e per una migliore comprensione del funzionamento si rimanda agli esempi successivi.

Esempio LD



7.19 Funzioni ed FB gestione One-Wire (ePLC1WireLib)

La rete da campo 1 Wire® è un protocollo standard basato su di un solo filo di comunicazione, come indica lo stesso nome, che include numerosi dispositivi e sensori frequentemente utilizzati nel campo dell'automazione industriale e domestica.

I dispositivi sono interconnessi da soli due fili, uno per la massa ed uno per il segnale e l'alimentazione; su questi due fili possono essere collegati tutti i dispositivi in rete scegliendo la disposizione fisica necessaria.

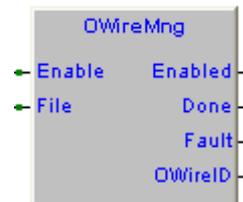
Il protocollo 1 Wire® è dotato di tutte le modalità di comunicazione che consentono di ottenere un elevato trasferimento dati ed una sicurezza intrinseca sulla loro validità. Questo avviene grazie a tecniche di indirizzamento univoche, CRC polinomiali di controllo, numerosi comandi di verifica e complessi algoritmi di gestione.

7.19.1 OWireMng, One-Wire management

Type	Library	Version
FB	ePLC1WireLib	SFR059A000

Questo blocco funzione gestisce il convertitore Seriale/One-Wire connesso al dispositivo di I/O definito in **File**, l'FB gestisce l'inizializzazione e la gestione del convertitore.

L'uscita **Done** si attiva se il convertitore è correttamente inizializzato, l'uscita **Fault** si attiva per un loop di programma in caso di errori di gestione. L'FB ritorna un **OWireID** che deve essere passato alle FB collegate (Esempio lettura ROM code, acquisizione temperatura, ecc.).



- Enable** (BOOL) Abilitazione blocco funzione, attivandolo viene gestito il convertitore Seriale/One-Wire.
- File** (FILEP) Flusso dati **stream** ritornato dalla funzione **Sysfopen**.
- Enabled** (BOOL) Blocco funzione abilitato.
- Done** (BOOL) Convertitore Seriale/One-Wire correttamente inizializzato e funzionante.
- Fault** (BOOL) Attivo per un loop di programma se errore gestione convertitore.
- OWireID** (UDINT) ID One-Wire da passare alle FB collegate (Esempio **OWRdIdentifier**, **OWRdTemperature**, ecc.).

Codici di errore

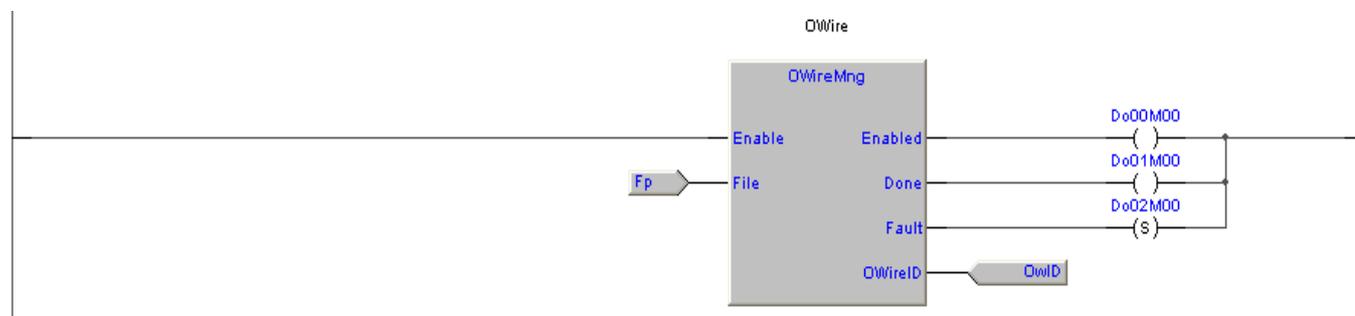
In caso di errore si attiva l'uscita **Fault**, con [SysGetLastError](#) è possibile rilevare il codice di errore.

- 10008010 Valore di **File** non definito.
- 10008200 Timeout invio stringa comando.
- 10008300~9 Errore nelle configurazione convertitore.
- 10008400~1 Errore nella verifica configurazione.
- 10008500~2 Errore nelle sequenze indirizzamento device One-Wire.
- 10008600 Errore risposta a comando di reset pulse.
- 10008601 Bus One-Wire in cortocircuito.
- 10008602 Errore dispositivi su bus One-Wire.
- 10008603 Nessun dispositivo su bus One-Wire.

Esempi

Nell'esempio è gestito un convertitore Seriale/One-Wire connesso al terminale di I/O definito nella variabile **Fp**, per la definizione delle variabili e per una migliore comprensione del funzionamento si rimanda agli esempi successivi.

Esempio LD



Type	Library	Version
FB	ePLC1WireLib	SFR059A000

7.19.2 OWRdIdentifier, One-Wire read ROM identifier

Questo blocco funzione esegue la lettura del codice di identificazione di un dispositivo One-Wire, si collega al blocco funzione **OWireMng** di gestione convertitore Seriale/One-Wire. Occorre passare **OWireID** in uscita dal blocco funzione di gestione convertitore.

Attivando il comando di **Start**, viene eseguita la lettura del ROM ID dal dispositivo connesso al bus One-Wire **Attenzione! Bisogna avere un solo dispositivo connesso al bus**. Se la lettura ha esito positivo si attiva per un loop di programma l'uscita **Done** e gli 8 bytes del codice letto sono trasferiti nell'array indirizzato.



- Enable** (BOOL) Abilita il blocco funzione.
- Start** (BOOL) Attivando l'ingresso viene eseguita la lettura del ROM ID.
- OWireID** (UDINT) ID One-Wire fornito in uscita dal blocco funzione [OWireMng](#).
- IDCode** (@USINT) Puntatore ad array di memorizzazione ROM ID letto, l'array deve essere almeno 8 bytes.
- Enabled** (BOOL) Blocco funzione abilitato.
- Done** (BOOL) Attivo per un loop al termine della lettura del ROM ID.
- Fault** (BOOL) Attivo per un loop se errore.

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con [SysGetLastError](#) è possibile rilevare il codice di errore.

- 10009010 **OWireID** non definito.
- 10009020 **OWireID** non corretto.
- 10009100 FB **OWireMng**, gestione convertitore Seriale/One-Wire, impegnata.
- 10009200~2 Errore gestione sequenze One-Wire lettura ID.

Esempi

Ecco un semplice esempio di un programma per la gestione di dispositivi iButton per il riconoscimento personale. Inserendo il TAG nel lettore viene eseguita la lettura del ROM identifier, il valore acquisito è trasferito in un array di 8 bytes allocato ad indirizzo **MD100.0**.

Ogni 100 mS viene eseguita l'acquisizione, se un TAG è inserito nel lettore viene attivata l'uscita **Do00M00**.

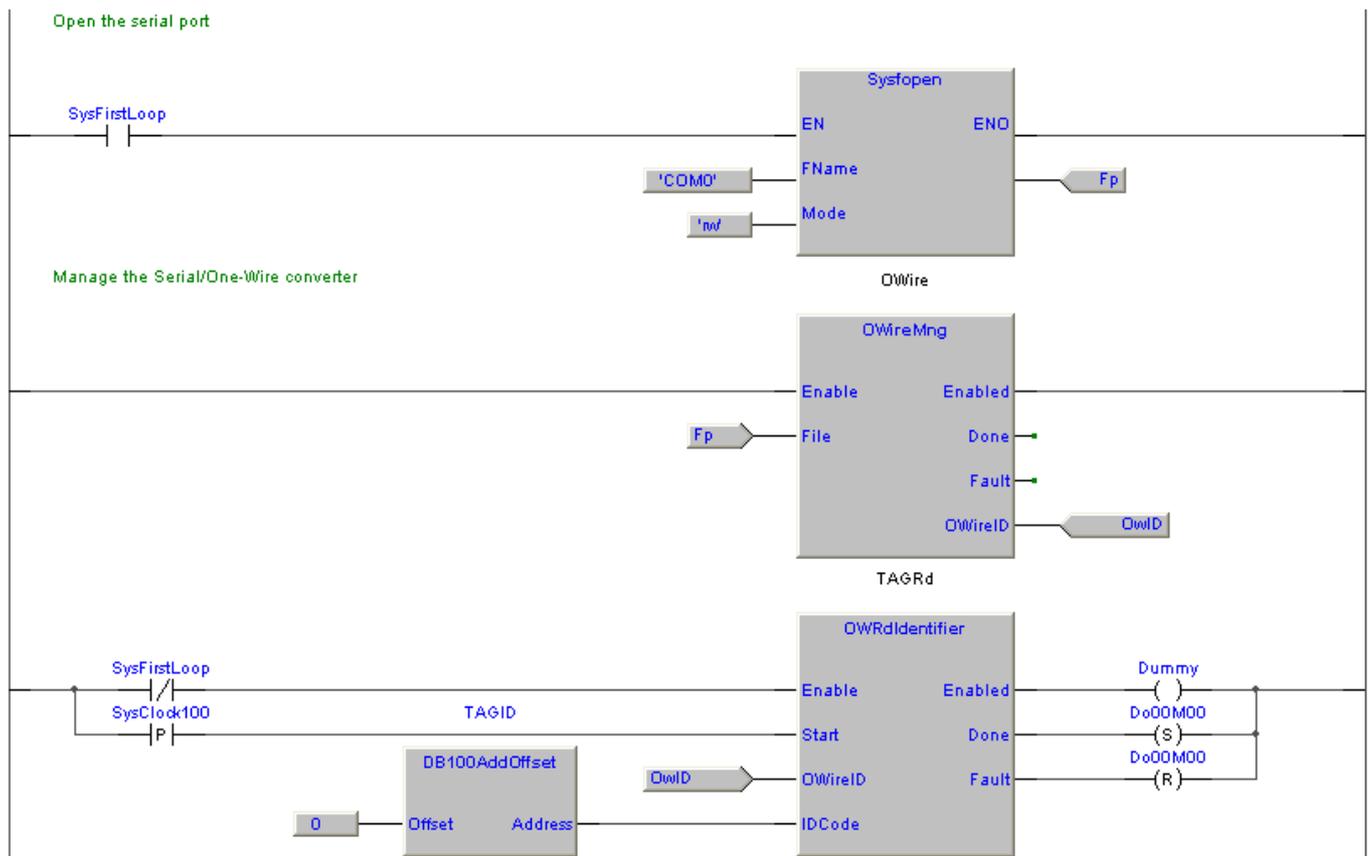
Per semplicità nel programma non viene eseguito alcun controllo sull'ID letto, ma in un sistema di controllo accessi ad esempio è possibile dall'ID letto identificare la persona ed abilitare o no l'accesso.



Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable
2	Fp	FILEP	Auto	No	0	..	File pointer
3	OwID	UDINT	Auto	No	0	..	One-Wire ID
4	OWire	OWireMng	Auto	No	0	..	One-Wire management FB
5	TAGID	DB100AddOffset	Auto	No	0	..	DB100 address TAG ID
6	TAGRd	OWRdIdentifier	Auto	No	0	..	One-Wire read identifier

Esempio LD (Ptp120A000)



Type	Library	Version
FB	ePLC1WireLib	SFR059A000

7.19.3 OWRdTemperature, One-Wire read temperature

Questo blocco funzione esegue gestione di un sensore One-Wire di acquisizione temperatura (Codice Maxim DS18B20), si collega al blocco funzione **OWireMng** di gestione convertitore Seriale/One-Wire. Occorre passare **OWireID** in uscita dal blocco funzione di gestione convertitore.



Attivando il comando di **Start**, viene eseguita la lettura del valore di temperatura dal dispositivo connesso al bus One-Wire. Se la lettura ha esito positivo si attiva per un loop di programma l'uscita **Done** e su **Temperature**, sarà riportato il valore di temperatura acquisito. L'uscita **Fault** si attiva per un loop di programma in caso di errori di gestione.

Se sul bus One-Wire è connesso un unico dispositivo possibile forzare a **0** la definizione in **IDCode** del ROM ID, se invece sul bus One-Wire sono presenti più dispositivi parallelati, occorre definire in **IDCode** l'indirizzo dell'array di 8 bytes che contiene il ROM ID del dispositivo che si vuole acquisire.

Enable (BOOL)	Abilita il blocco funzione.
Start (BOOL)	Attivando l'ingresso viene eseguita la lettura del ROM ID.
OWireID (UDINT)	ID One-Wire fornito in uscita dal blocco funzione OWireMng .
IDCode (@USINT)	Puntatore ad array definizione ROM ID dispositivo da acquisire.
Enabled (BOOL)	Blocco funzione abilitato.
Done (BOOL)	Attivo per un loop al termine della lettura temperatura.
Fault (BOOL)	Attivo per un loop se errore.
Temperature (REAL)	Valore di temperatura acquisito (°C). Range di lettura da -55 (°C) a +125 (°C). Precisione ±0.5 (°C) tra -10 (°C) e +85 (°C). Risoluzione di lettura 0.0625 (°C).

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con **SysGetLastError** è possibile rilevare il codice di errore.

- 10010010 **OWireID** non definito.
- 10010020 **OWireID** non corretto.
- 10010100 FB **OWireMng**, gestione convertitore Seriale/One-Wire, impegnata.
- 10010200~5 Errore nelle sequenze acquisizione temperatura.

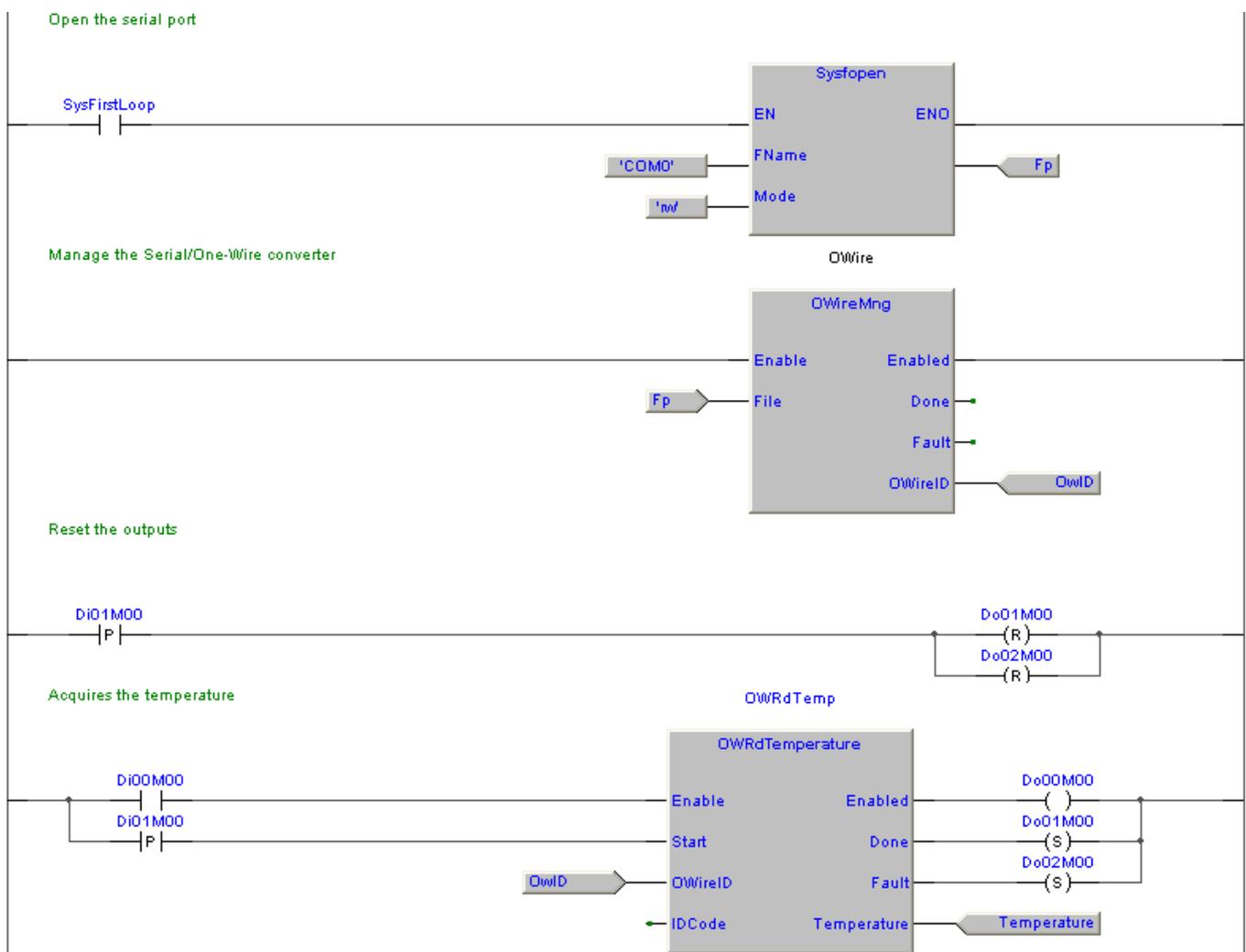
Esempi

Viene eseguita la lettura della temperatura da un dispositivo One-Wire. Non essendo definito **IDCode** viene acquisito qualsiasi dispositivo presente sul bus One-Wire **Attenzione! Deve essere presente un solo dispositivo sul bus**. Ad ogni attivazione ingresso **Di01M00** viene eseguita l'acquisizione, se esecuzione corretta viene attivata l'uscita **Do01M00** ed il valore acquisito è trasferito nella variabile **Temperature**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer
2	OwID	UDINT	Auto	No	0	..	One-Wire ID
3	OWire	OWireMng	Auto	No	0	..	One-Wire management FB
4	OWRdTemp	OWRdTempei	Auto	No	0	..	One-Wire read temperature FB
5	Temperature	REAL	Auto	No	0	..	Temperature (°C)

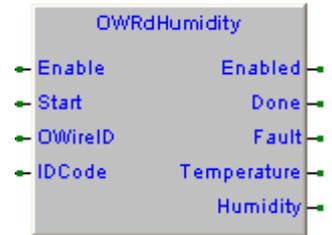
Esempio LD (Ptp120A000)



Type	Library	Version
FB	ePLC1WireLib	SFR059B000

7.19.4 OWRdHumidity, One-Wire read humidity

Questo blocco funzione esegue gestione di un sensore One-Wire di acquisizione umidità e temperatura basato su convertitore Maxim DS2438. Si collega al blocco funzione **OWireMng** di gestione convertitore Seriale/One-Wire. Occorre passare **OWireID** in uscita dal blocco funzione di gestione convertitore.



Attivando il comando di **Start**, viene eseguita la lettura del valore di temperatura dal dispositivo connesso al bus One-Wire. Se la lettura ha esito positivo si attiva per un loop di programma l'uscita **Done**, su **Temperature** e **Humidity**, saranno ritornati i valori di temperatura e umidità acquisito. L'uscita **Fault** si attiva per un loop di programma in caso di errori di gestione.

Se sul bus One-Wire è connesso un unico dispositivo è possibile forzare a **0** la definizione in **IDCode** del ROM ID, se invece sul bus One-Wire sono presenti più dispositivi parallelati, occorre definire in **IDCode** l'indirizzo dell'array di 8 bytes che contiene il ROM ID del dispositivo che si vuole acquisire.

- Enable** (BOOL) Abilita il blocco funzione.
- Start** (BOOL) Attivando l'ingresso viene eseguita la lettura del ROM ID.
- OWireID** (UDINT) ID One-Wire fornito in uscita dal blocco funzione [OWireMng](#).
- IDCode** (@USINT) Puntatore ad array definizione ROM ID dispositivo da acquisire.
- Enabled** (BOOL) Blocco funzione abilitato.
- Done** (BOOL) Attivo per un loop al termine della lettura temperatura.
- Fault** (BOOL) Attivo per un loop se errore.
- Temperature** (REAL) Valore di temperatura acquisito (°C). Range di lettura da -55 (°C) a +125 (°C). Precisione ±0.5 (°C) tra -10 (°C) e +85 (°C). Risoluzione di lettura 0.03125 (°C).
- Humidity** (REAL) Valore di umidità acquisito (RH%).

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con [SysGetLastError](#) è possibile rilevare il codice di errore.

- 10015010 **OWireID** non definito.
- 10015020 **OWireID** non corretto.
- 10015100 FB **OWireMng**, gestione convertitore Seriale/One-Wire, impegnata.
- 10015200~3 Errore nelle sequenze conversione temperatura.
- 10015300~8 Errore nelle sequenze acquisizione tensione alimentazione sensore.
- 10015400~8 Errore nelle sequenze acquisizione sensore umidità.

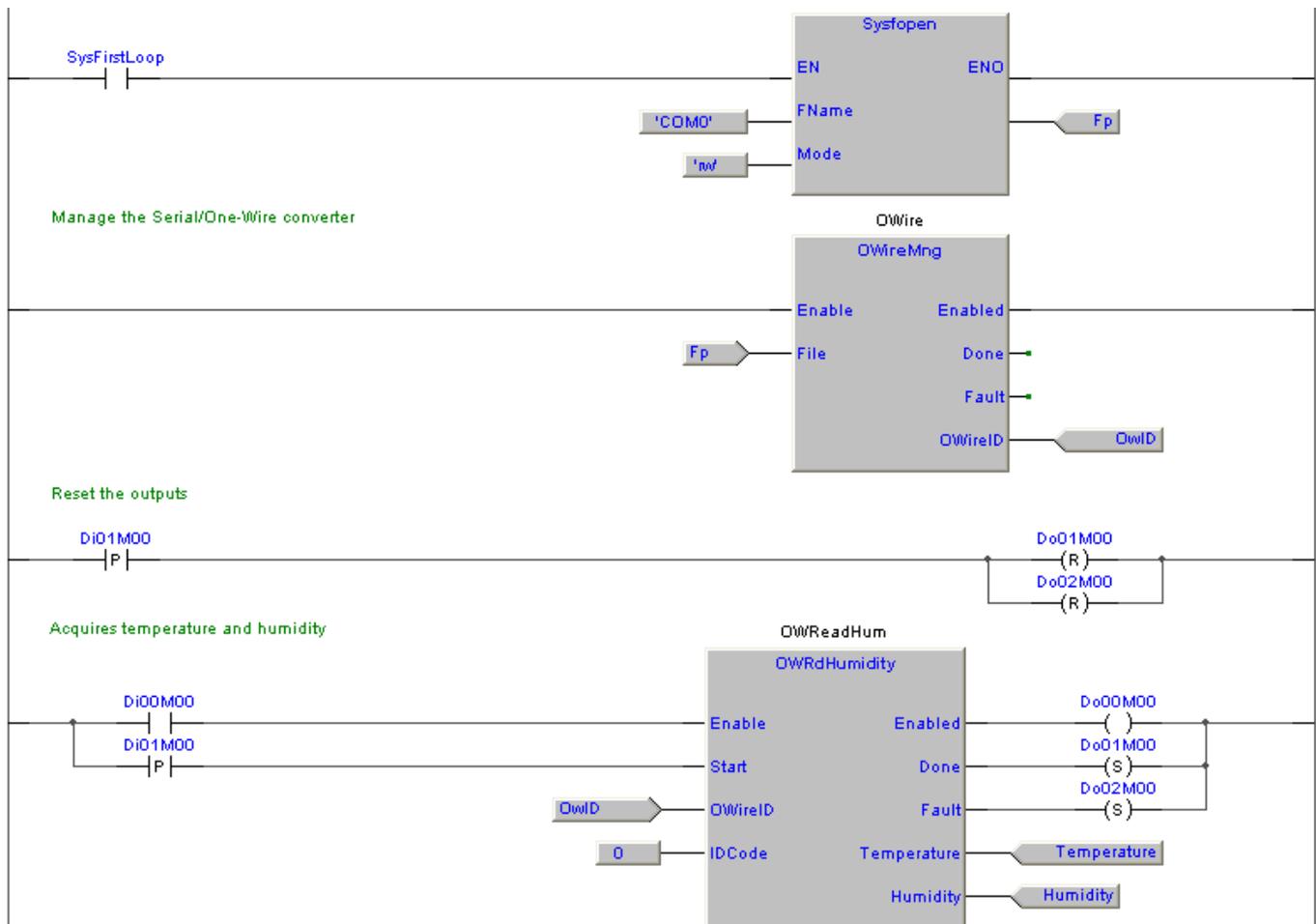
Esempi

Viene eseguita la lettura della temperatura ed umidità da un dispositivo One-Wire. Non essendo definito **IDCode** viene acquisito qualsiasi dispositivo presente sul bus One-Wire **Attenzione! Deve essere presente un solo dispositivo sul bus**. Ad ogni attivazione ingresso **Di01M00** viene eseguita l'acquisizione, se esecuzione corretta viene attivata l'uscita **Do01M00** ed i valori acquisiti sono trasferiti nelle variabili **Temperature** ed **Humidity**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	OwID	UDINT	Auto	No	0	..	One-Wire ID
2	Humidity	REAL	Auto	No	0	..	Humidity (RH%)
3	Temperature	REAL	Auto	No	0	..	Temperature (°C)
4	Fp	FILEP	Auto	No	0	..	File pointer
5	OWire	OWireMng	Auto	No	0	..	One-Wire management FB
6	OWReadHum	OWRdHumidi	Auto	No	0	..	FB read humidity data

Esempio LD (Ptp120A100)



7.20 Funzioni ed FB gestione networking

Da linguaggio IEC sono disponibili funzioni e blocchi funzione per la gestione networking.

7.20.1 SysIPReach, IP address is reachable

Type	Library	Version
FB	Embedded	6.0

Questo blocco funzione esegue controllo se un indirizzo IP è raggiungibile, se indirizzo è già presente nella tabella ARP del sistema viene ritornato **Done**.

In caso contrario viene eseguita una richiesta ARP sulla rete e se il dispositivo con l'indirizzo IP definito viene trovato, è attivata l'uscita **Done** e aggiornata la tabella ARP.

La variabile **Refresh** ritorna la percentuale di tempo trascorsa dall'ultimo aggiornamento della tabella ARP del sistema con l'indirizzo IP definito.



- Enable** (BOOL) Abilitazione blocco funzione, attivandolo si forza socket in condizione di listening.
- PeerIP** (STRING[15]) Stringa di definizione indirizzo IP di cui eseguire la ricerca.
- Done** (BOOL) Attivo se indirizzo IP è raggiungibile (Voce presente in tabella ARP).
- Fault** (BOOL) Attivo per un loop di programma se errore gestione.
- Refresh** (USINT) Percentuale di tempo da ultimo aggiornamento tabella ARP.

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con [SysGetLastError](#) è possibile rilevare il codice di errore.

Codice	Descrizione
9974050	Errore allocazione blocco funzione
9974060	Terminato spazio memoria rilocabile, non è possibile eseguire l'FB.
9974070	Errore versione blocco funzione.
9974100	Errore indirizzo IP definito PeerIP .
9974200	Errore invio richiesta ARP.

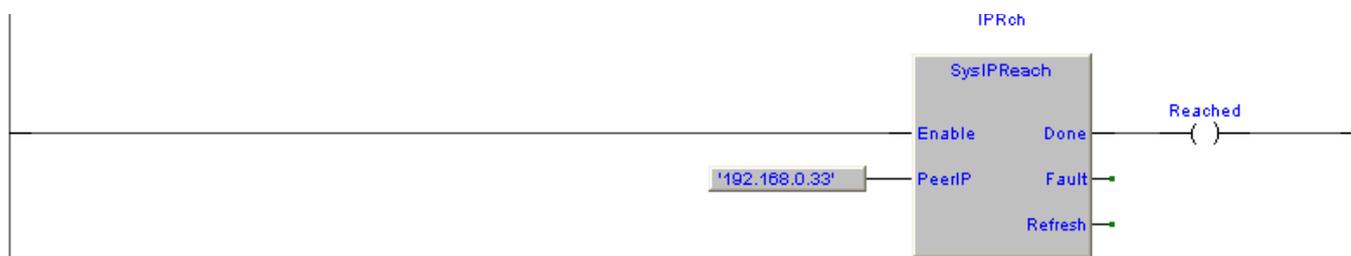
Esempi

Nell'esempio viene controllato se l'indirizzo IP **192.168.0.33** è raggiungibile, in tal caso si attiva l'uscita **Reached**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Reached	BOOL	Auto	No	FALSE	..	IP address reached
2	IPRch	SysIPReach	Auto	No	0	..	FB IP reach

Esempio LD



Type	Library	Version
FB	Embedded	6.0

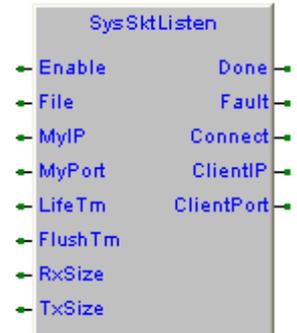
7.20.2 SysSktListen, Socket listen

Questo blocco funzione forza il socket nella condizione di listening, occorre passare alla FB un flusso dati **stream** indicato dal parametro **File**, precedentemente aperto dalla funzione **Sysfopen**.

I parametri **MyIP** e **MyPort** indicano l'interfaccia di rete e la porta su cui il socket è in ascolto, **MyIP** può essere lasciato non connesso ed in tal caso viene utilizzata l'interfaccia ethernet.

I parametri **LifeTm**, **FlushTm**, **RxSize**, **TxSize**, devono essere definiti solo se si tratta di socket TCP, possono essere lasciati non connessi nel caso di socket UDP.

L'uscita **Done** si attiva se tutti i parametri sono corretti ed il socket è stato messo in condizione di listening. Quando un client si connette (Solo socket TCP) si attiva l'uscita **Connect** e nelle variabili **ClientIP** e **ClientPort** sono ritornati l'indirizzo IP e la porta connessa al socket.



- Enable** (BOOL) Abilitazione blocco funzione, attivandolo si forza socket in condizione di listening.
- File** (FILEP) Flusso dati **stream** ritornato dalla funzione **Sysfopen**.
- MyIP** (STRING[15]) Stringa di definizione indirizzo IP interfaccia di rete su cui il socket è gestito. Lasciandolo non connesso o definendo **'255.255.255.255'** sarà utilizzata interfaccia ethernet.
- MyPort** (UINT) Porta su cui sarà posto in ascolto il socket.
- LifeTm** (UINT) Tempo di vita socket, se non sono ricevuti o inviati dati dopo il tempo definito il socket viene automaticamente chiuso (Sec) (Solo per socket TCP).
- FlushTm** (UINT) Tempo di flush dati, se non sono caricati dati nel socket dopo il tempo definito i dati presenti vengono automaticamente inviati (mS) (Solo per socket TCP).
- RxSize** (UINT) Dimensione buffer ricezione dati (Solo per socket TCP).
- TxSize** (UINT) Dimensione buffer trasmissione dati (Solo per socket TCP).
- Done** (BOOL) Attivo se socket in condizione di listening.
- Fault** (BOOL) Attivo per un loop di programma se errore gestione.
- Connect** (BOOL) Si attiva se un client si connette al socket (Solo per socket TCP).
- ClientIP** (STRING[15]) Stringa di definizione indirizzo IP del client connesso al socket (Solo per socket TCP).
- ClientPort** (UINT) Porta del client attualmente connesso al socket (Solo per socket TCP).

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con **SysGetLastError** è possibile rilevare il codice di errore.

Codice	Descrizione
9977050	Errore allocazione blocco funzione
9977060	Terminato spazio memoria rilocabile, non è possibile eseguire l'FB.
9977070	Errore versione blocco funzione.
9977100	Valore di File non definito.
9977110	Tipo di stream definito in File non corretto.
9977150	Errore indirizzo IP dispositivo client ClientIP .
9977200	Errore indirizzo IP interfaccia di rete MyIP .
9977300	Valore dimensione buffer ricezione RxSize fuori range.
9977310	Valore dimensione buffer trasmissione TxSize fuori range.
9977350	Errore nel set dei parametri TCP.
9977400	Errore set socket in listening.

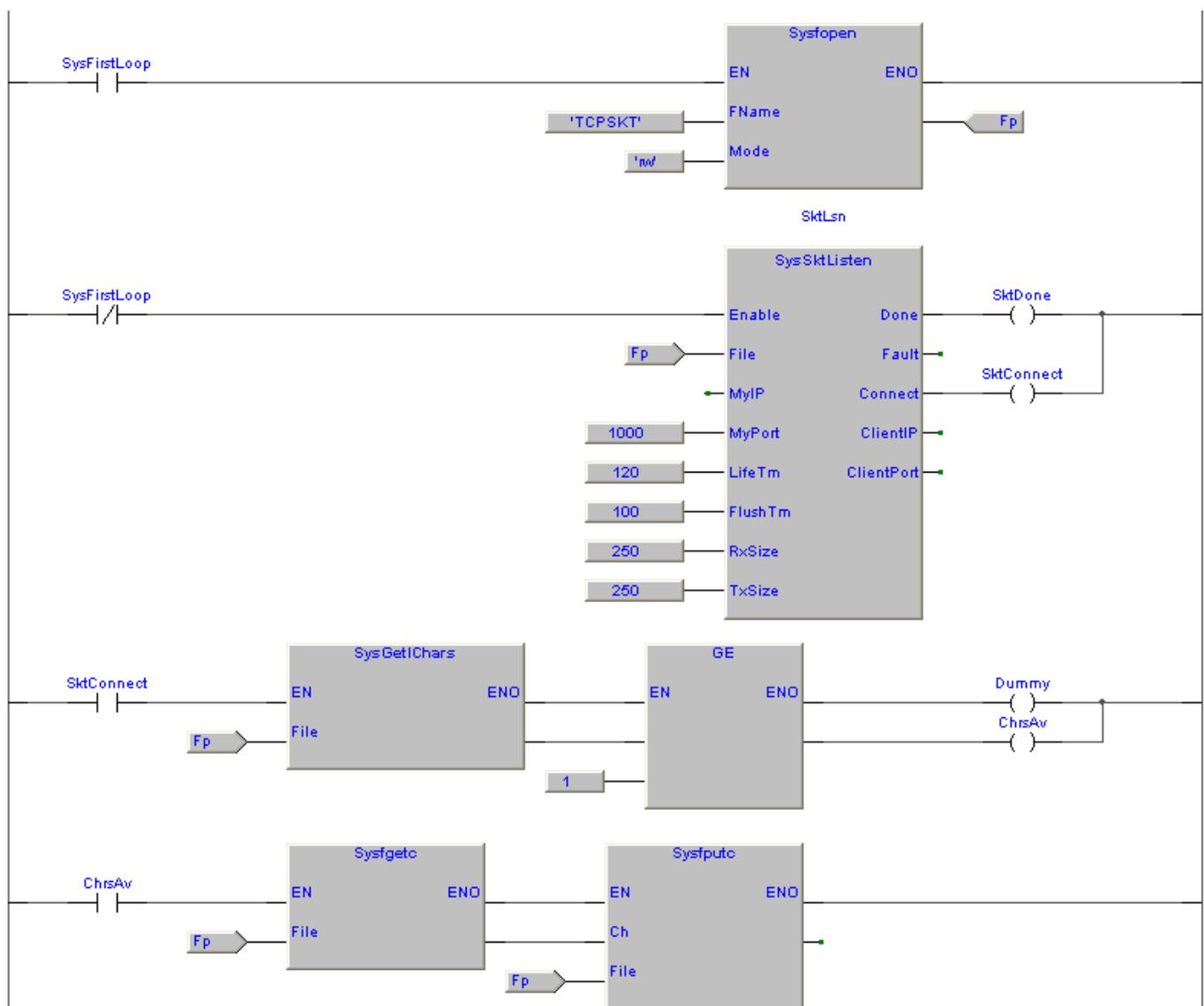
Esempi

Nell'esempio è posto in ascolto un socket TCP sulla porta 1000. Se ci si connette con un client telnet (Esempio Hyperterminal) al sistema SlimLine sulla porta 1000, si attiverà la variabile **SkTConnect**. Inviando caratteri dalla finestra del terminale telnet, si attiverà la variabile **ChrsAv**, i caratteri inviati saranno ricevuti dalla funzione **Sysfgetc** e inviati in echo sul cliente telnet dalla funzione **Sysfputc**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer
2	SkTLsn	SysSkListen	Auto	No	0	..	Socket listen FB
3	SkTDone	BOOL	Auto	No	FALSE	..	Socket done
4	SkTConnect	BOOL	Auto	No	FALSE	..	Socket connect
5	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable
6	ChrsAv	BOOL	Auto	No	FALSE	..	Characters available

Esempio LD (Ptp119a200)



Type	Library	Version
FB	Embedded	6.0

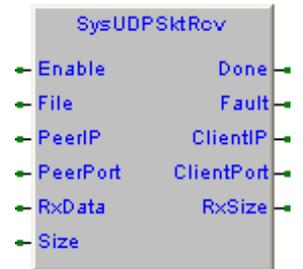
7.20.3 SysUDPSktRcv, UDP socket receive

Questo blocco funzione esegue la ricezione dati da un socket UDP, occorre passare alla FB un flusso dati **stream** indicato dal parametro **File**, precedentemente aperto dalla funzione [Sysfopen](#) ed il socket deve essere stato posto in condizione di listening dalla funzione [SysSktListen](#).

I parametri **PeerIP** e **PeerPort** indicano l'indirizzo IP e la porta da cui sono accettati i dati, se non connessi sono accettate connessioni da tutti gli IP e da tutte le porte.

Nel parametro **RxData** bisogna definire l'indirizzo del buffer di memoria in cui verranno trasferiti i dati ricevuti, ed in **Size** occorre definire la dimensione del buffer.

Su ricezione dati si attiva l'uscita **Done** e nelle variabili **ClientIP** e **ClientPort** sono ritornati l'indirizzo IP e la porta da cui sono stati ricevuti i dati, mentre in **RxSize** è ritornato il numero di bytes dati ricevuti.



- Enable** (BOOL) Abilitazione blocco funzione.
- File** (FILEP) Flusso dati **stream** ritornato dalla funzione **Sysfopen**.
- PeerIP** (STRING[15]) Stringa di definizione indirizzo IP da cui saranno accettate le connessioni. Lasciandolo non connesso o definendo **'255.255.255.255'** saranno accettate connessioni da tutti gli IP.
- PeerPort** (UINT) Porta da cui saranno accettate le connessioni, Lasciandolo non connesso o definendo **65536** saranno accettate connessioni da tutte le porte.
- RxData** (@USINT) Puntatore al buffer dove devono essere trasferiti i dati ricevuti.
- Size** (UINT) Dimensione buffer ricezione dati.
- Done** (BOOL) Attivo per un loop di programma su ricezione dati.
- Fault** (BOOL) Attivo per un loop di programma se errore gestione.
- ClientIP** (STRING[15]) Stringa di definizione indirizzo IP del client da cui sono stati ricevuti i dati.
- ClientPort** (UINT) Porta del client da cui sono stati ricevuti i dati.
- RxSize** (UINT) Numero di bytes di dato ricevuti.

Codici di errore

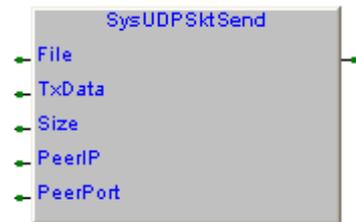
In caso di errore si attiva l'uscita **Fault**, con [SysGetLastError](#) è possibile rilevare il codice di errore.

Codice	Descrizione
9975050	Errore allocazione blocco funzione
9975060	Terminato spazio memoria rilocabile, non è possibile eseguire l'FB.
9975070	Errore versione blocco funzione.
9975100	Valore di File non definito.
9975110	Tipo di stream definito in File non corretto.
9975200	Errore indirizzo IP connessioni accettate PeerIP .
9975300	Errore ricezione dati da socket UDP
9975350	Errore indirizzo IP client ClientIP .

Type	Library	Version
Function	Embedded	6.0

7.20.4 SysUDPSktSend, UDP socket send

Questa funzione esegue la trasmissione dati su di un socket UDP, occorre passare un flusso dati **stream** indicato dal parametro **File**, precedentemente aperto dalla funzione **Sysfopen** ed il socket deve essere stato posto in condizione di listening dalla funzione **SysSktListen**.



Nel parametro **TxData** bisogna definire l'indirizzo del buffer di memoria che contiene i dati da trasmettere, ed in **Size** occorre definire il numero di bytes di dati da trasmettere.

I parametri **PeerIP** e **PeerPort** indicano l'indirizzo IP e la porta a cui saranno inviati i dati.

Se la trasmissione è riuscita verrà ritornato il numero di bytes trasmessi (Solitamente uguale al valore definito in **Size**), in caso di errore viene ritornato **EOF**.

Parametri funzione:

- File** (FILEP) Flusso dati **stream** ritornato dalla funzione **Sysfopen**.
- TxData** (@USINT) Puntatore al buffer che contiene i dati da trasmettere.
- Size** (UINT) Numero di bytes dati da trasmettere.
- PeerIP** (STRING[15]) Stringa di definizione indirizzo IP a cui trasmettere i dati.
- PeerPort** (UINT) Porta a cui trasmettere i dati.

La funzione ritorna:

(UINT) Numero bytes trasmessi, **EOF** in caso di errore

Codici di errore

In caso di errore la funzione torna con **EOF**, con **SysGetLastError** è possibile rilevare il codice di errore.

Codice	Descrizione
9976010	Valore di File non definito.
9976050	Tipo di stream definito in File non corretto.
9976100	Errore indirizzo IP connessioni accettate PeerIP .
9976200	Errore trasmissione dati su socket UDP

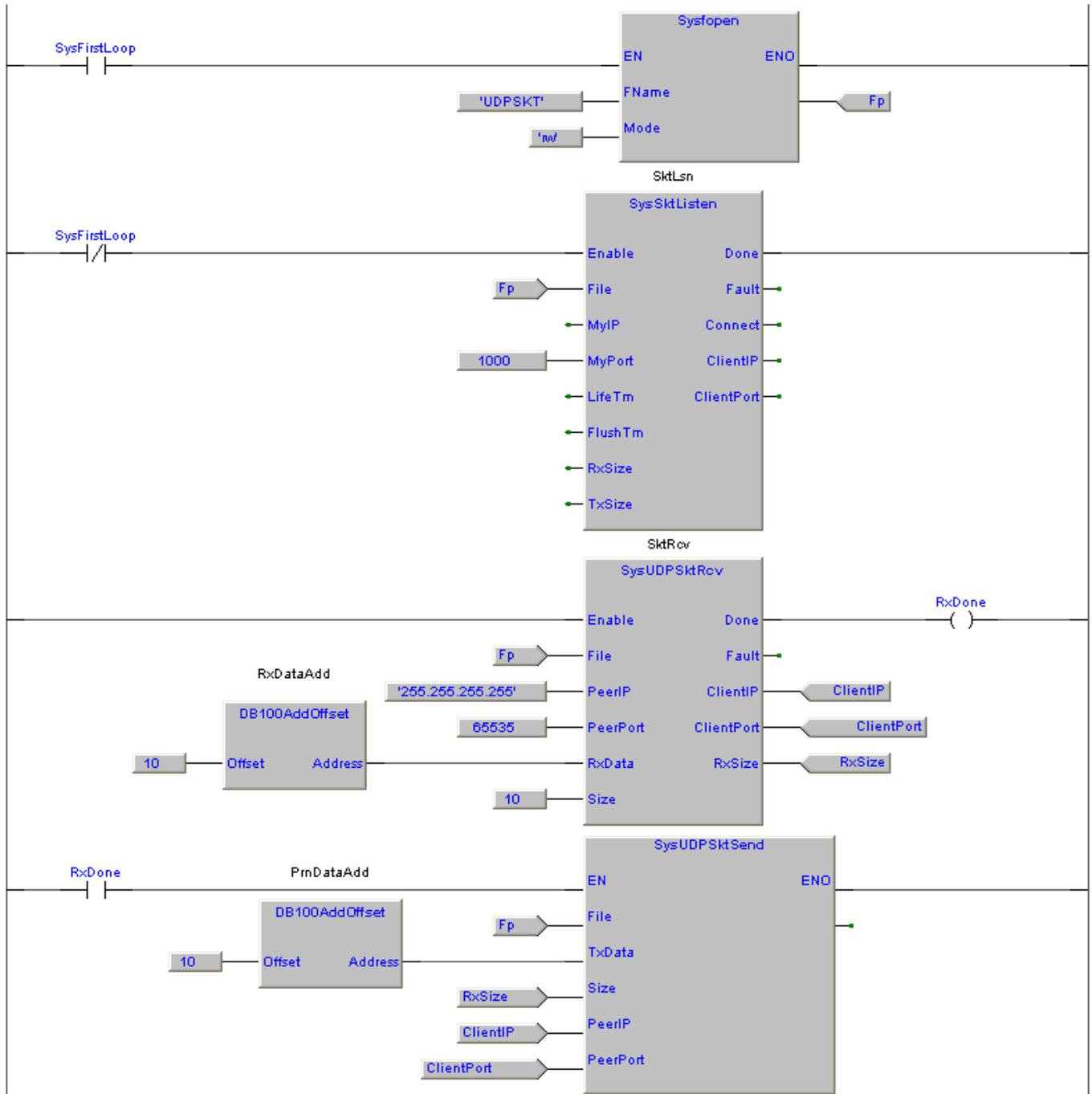
Esempi

Nell'esempio inviando caratteri al sistema SlimLine sulla porta 1000 da una connessione UDP, i caratteri inviati saranno ritrasmessi sulla porta UDP del sistema client da cui sono stati ricevuti.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	RxDone	BOOL	Auto	No	FALSE	..	Rx data done
2	ClientPort	UINT	Auto	No	0	..	Client port
3	RxSize	UINT	Auto	No	0	..	Rx bytes received
4	ClientIP	STRING	Auto	[15]		..	Client IP address
5	PrmDataAdd	DB100AddOffset	Auto	No	0	..	Print data buffer address
6	RxDataAdd	DB100AddOffset	Auto	No	0	..	Rx data buffer address
7	Fp	FILEP	Auto	No		..	File pointer
8	SktLsn	SysSktListen	Auto	No	0	..	FB socket listen
9	SktRcv	SysUDPSktRcv	Auto	No	0	..	FB socket receive

Esempio LD (Ptp119a200)



Type	Library	Version
FB	PLCUtyLib	SFR054A700

7.20.5 UDPDataTxfer, UDP data transfer

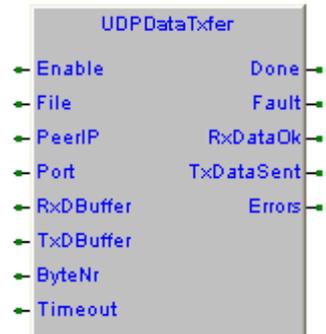
Questo blocco funzione esegue il trasferimento di un blocco di memoria tra due sistemi utilizzando una connessione UDP su rete ethernet. Occorre passare alla FB un flusso dati **stream** indicato dal parametro **File**, precedentemente aperto dalla funzione **Sysfopen** ed il socket deve essere stato posto in condizione di listening dalla funzione **SysSktListen**.

Il parametro **PeerIP** indica l'indirizzo IP del sistema con cui avviene il trasferimento dati, **Port** indica la porta tramite la quale il trasferimento avviene (Deve assumere lo stesso valore su entrambi i sistemi).

Il parametro **Timeout** definisce il tempo massimo per il trasferimento dei dati, l'invio dei dati si conclude con la ricezione di un acknowledge da parte dell'altro sistema, un ciclo di invio dati e ricezione acknowledge richiede 2 loop di esecuzione programma. Se dopo l'invio non viene ricevuto Ack in un tempo pari a **Timeout/4**, viene effettuato un altro invio e così di seguito fino allo scadere del tempo definito. Per garantire almeno 3 retries si consiglia di **impostare come tempo di timeout un valore pari a 10 volte il tempo massimo di loop** (Scegliendo quello maggiore tra i due sistemi).

L'invio dei dati è automatico sulla variazione di uno qualsiasi dei bytes del buffer di trasmissione, per garantire il controllo sul link tra i due sistemi, ogni tempo pari a **Timeout** viene comunque eseguito un invio del buffer di memoria.

Se i due sistemi sono in comunicazione si attiva l'uscita **Done**, **RxDataOk** si attiva per un loop ad ogni ricezione del buffer dati dall'altro sistema, mentre **TxDataSent** si attiva per un loop al termine della trasmissione del buffer dati verso l'altro sistema.



Enable (BOOL)	Abilitazione blocco funzione.
File (FILEP)	Flusso dati stream ritornato dalla funzione Sysfopen .
PeerIP (STRING[15])	Stringa di definizione indirizzo IP del sistema con cui avviene il trasferimento dati.
Port (UINT)	Porta tramite la quale avviene il trasferimento dati (Stesso valore su entrambi i sistemi).
RxDBuffer (@USINT)	Puntatore al buffer dove devono essere trasferiti i dati ricevuti.
TxDBuffer (@USINT)	Puntatore al buffer dove sono presenti i dati da trasmettere.
ByteNr (UINT)	Numero di bytes scambiati.
Timeout (UINT)	Tempo massimo per il trasferimento del buffer dati (mS).
Done (BOOL)	Attivo se i due sistemi sono in comunicazione tra di loro.
Fault (BOOL)	Attivo per un loop di programma se errore gestione.
RxDataOk (BOOL)	Attivo per un loop di programma su ricezione buffer dati da altro sistema.
TxDataSent (BOOL)	Attivo per un loop di programma al termine trasmissione buffer dati verso altro sistema.
Errors (UINT)	Numero di errori, incrementato ad ogni nuovo errore, raggiunto valore massimo riparte da 0.

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con **SysGetLastError** è possibile rilevare il codice di errore.

Codice	Descrizione
10014050	Valore di File non definito.
10014100	Terminato spazio memoria rilocabile, non è possibile eseguire l"FB.
10014200~1	Errore ricezione frame dati blocco di memoria.
10014300~2	Errore ricezione frame acknowledge.
10014400	Ricevuto comando non gestito.
10014500	Timeout invio frame dati blocco di memoria.
10014600	Errore sequenze di trasmissione.

Esempi

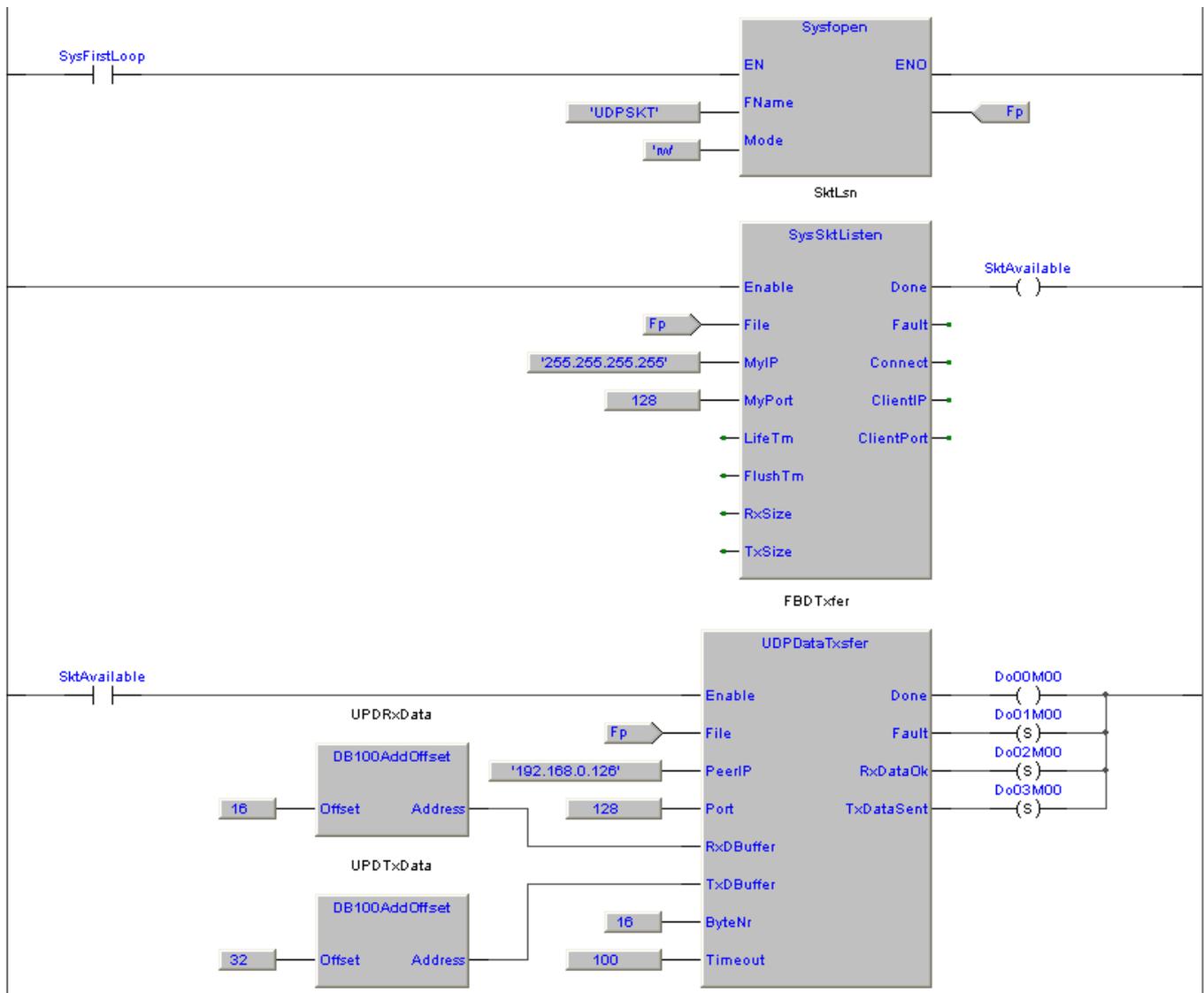
Nell'esempio viene trasferito un blocco di memoria di 16 bytes allocati su **MX100.16** verso il sistema con IP 192.168.0.126, i dati ricevuti sono trasferiti a **MX100.32**. Per testarne il funzionamento occorre trasferire lo stesso programma sul sistema con IP 192.168.0.126 definendo opportunamente la variabile **PeerIP**.

L'uscita **Do00M00** è attiva se i due sistemi sono sincronizzati tra di loro.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	SketAvailable	BOOL	Auto	No	FALSE	..	Socket available
2	UPDRxData	DB100AddOffset	Auto	No	0	..	Address of Rx data
3	UPDTxData	DB100AddOffset	Auto	No	0	..	Address of Tx data
4	Fp	FILEP	Auto	No	0	..	UDP socket
5	SketLsn	SysSketListen	Auto	No	0	..	Socket listen
6	FBDTxfer	UDPDataTxfer	Auto	No	0	..	UDP data transfer

Esempio LD (Ptp119a300)



7.21 Funzioni ed FB supporto prodotti Hw Group (eHwGSpLib)

La ditta della repubblica Ceca Hw Group <http://www.hw-group.com> produce dispositivi per Networking e prodotti per telecontrollo, monitoraggio e gestione dati.

Tutti i prodotti Hw Group dispongono di connettività su rete ethernet con protocolli TCP/IP, UDP, SNMP, ed per facilitare la connessione di questi prodotti con l'ambiente di sviluppo LogicLab, sono fornite funzioni e blocchi funzioni specifici.

Type	Library	Version
FB	eHwGSpLib	SFR060A000

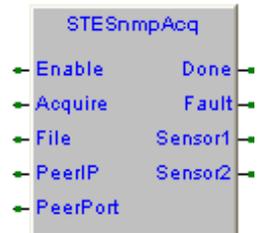
7.21.1 STESnmpAcq, STE thermometer acquisition over SNMP

Questo blocco funzione gestisce l'acquisizione del valore di temperatura delle due sonde connesse al termometro IP STE. La connessione tra il termometro e lo SlimLine avviene su rete ethernet utilizzando il protocollo SNMP.

Occorre passare alla FB un flusso dati **stream** indicato dal parametro **File**, precedentemente aperto dalla funzione **Sysfopen** come socket UDP e posto in ascolto con il blocco funzione **SysSktListen**.

Su fronte attivazione del comando **Acquire**, viene effettuata la lettura SNMP dal termometro STE definito da indirizzo IP **PeerIP**, su porta **PeerPort** (Di default la porta SNMP è la 161). Se il comando **Acquire** è mantenuto attivo, la lettura viene effettuata in modo ciclico.

L'uscita **Done** si attiva per un loop al termine della acquisizione dei due valori di temperatura.



- Enable** (BOOL) Abilitazione blocco funzione.
- Acquire** (BOOL) Comando acquisizione termometro STE. Sul fronte di attivazione si esegue la lettura del valore di temperatura. Mantenuto attivo la lettura viene effettuata in modo ciclico.
- File** (FILEP) Flusso dati **stream** ritornato dalla funzione **Sysfopen**.
- PeerIP** (STRING[15]) Stringa di definizione indirizzo IP del termometro IP.
- PeerPort** (UINT) Porta utilizzata per la connessione (Di default la porta SNMP è la 161).
- Done** (BOOL) Attivo per un loop di programma su fine lettura dati.
- Fault** (BOOL) Attivo per un loop di programma se errore gestione.
- Sensor1** (REAL) Valore di temperatura letto dal sensore 1 (°C).
- Sensor2** (REAL) Valore di temperatura letto dal sensore 2 (°C).

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con **SysGetLastError** è possibile rilevare il codice di errore.

- 10013010 Valore di **File** non definito.
- 10013050 Timeout esecuzione.
- 10013060 Errore case gestione lettura.
- 10013100 Errore controllo indirizzo IP del dispositivo STE.
- 10013120 Errore ricezione dati da dispositivo STE.
- 10013200~1 Errore lettura sensore 1.
- 10013300~1 Errore lettura sensore 2.

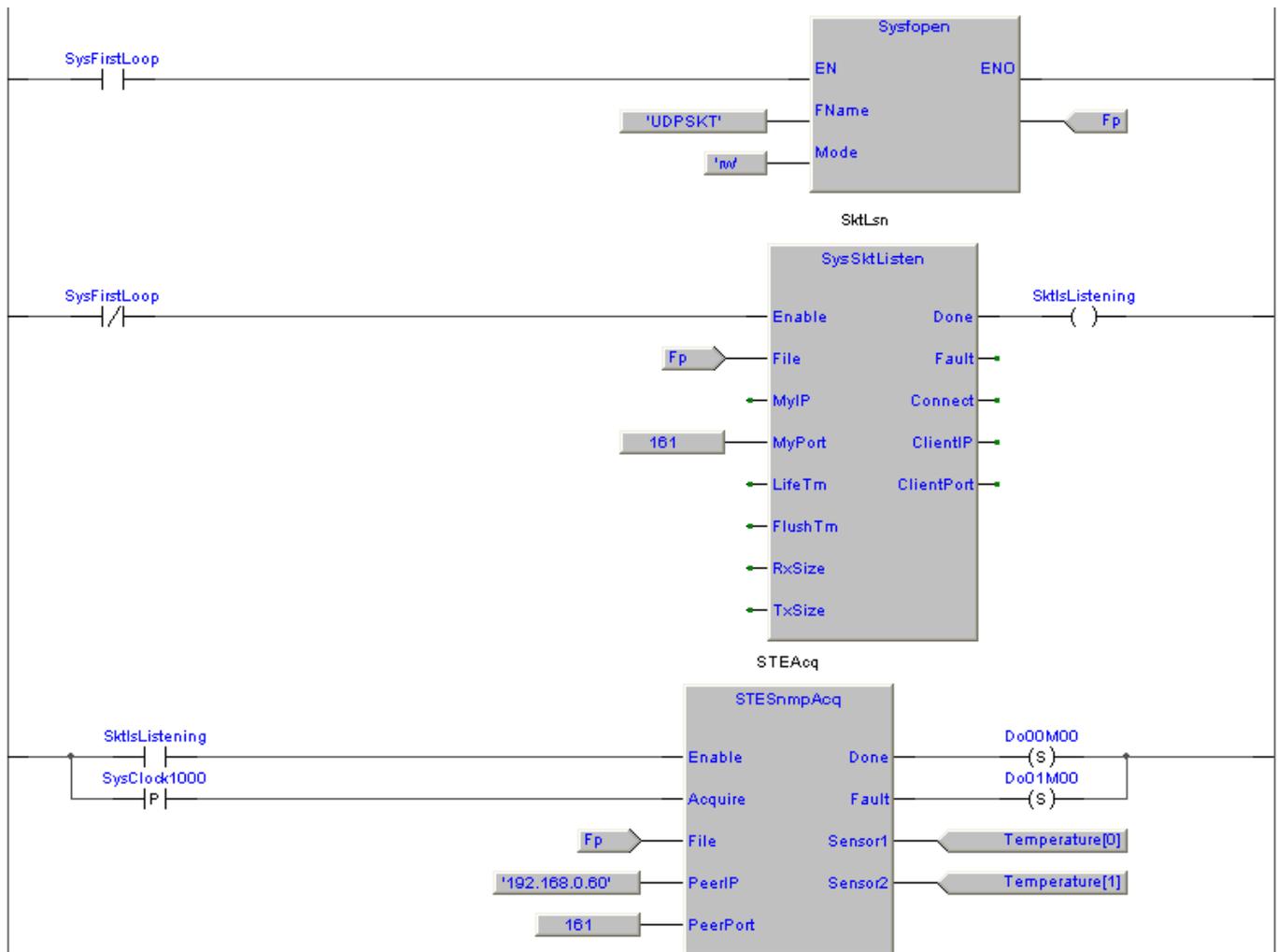
Esempi

Nell'esempio è gestita l'acquisizione dei due valori di temperatura da un termometro STE ogni secondo. Il valore di temperatura in gradi centigradi è ritornato sulle variabili **Temperature[0]** e **Temperature[1]**. L'uscita logica **Do00M00** si attiva alla prima esecuzione della acquisizione, mentre l'uscita logica **Do01M00** si attiva in caso di errore di acquisizione.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	SktIsListening	BOOL	Auto	No	FALSE	..	Socket is listening
2	Temperature	REAL	Auto	[0..1]	2(0)	..	Temperature
3	Fp	FILEP	Auto	No	0	..	UDP socket
4	STEAcq	STESnmpAcq	Auto	No	0	..	STE acquisition
5	SktLsn	SysSktListen	Auto	No	0	..	FB socket listen data

Esempio LD



Type	Library	Version
FB	eHwGSpLib	SFR060A100

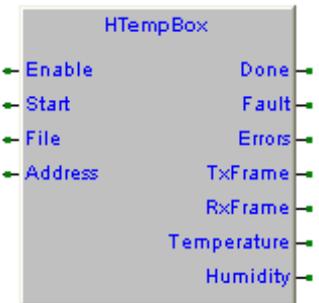
7.21.2 HTempBox, HTemp Box2-485 acquisition

Questo blocco funzione gestisce l'acquisizione del valore di temperatura e di umidità da un HTemp Box2-485 connesso con lo SlimLine in porta seriale RS485.

Occorre passare alla FB il puntatore al file di porta seriale **stream** indicato dal parametro **File**, precedentemente aperto dalla funzione [Sysfopen](#).

Su fronte attivazione del comando **Start**, viene effettuata la lettura del valore di temperatura e umidità dal convertitore HTemp connesso alla porta seriale definita. Se il comando **Start** è mantenuto attivo, la lettura viene effettuata in modo ciclico.

L'uscita **Done** si attiva per un loop al termine della acquisizione dei valori di temperatura e di umidità.



Enable (BOOL)	Abilitazione blocco funzione.
Start (BOOL)	Comando acquisizione da HTemp. Sul fronte di attivazione si esegue la lettura del valore di temperatura e umidità. Mantenuto attivo la lettura viene effettuata in modo ciclico.
File (FILEP)	Flusso dati stream ritornato dalla funzione Sysfopen .
Address (STRING[1])	Stringa di definizione indirizzo dispositivo.
Done (BOOL)	Attivo per un loop di programma su fine lettura dati.
Fault (BOOL)	Attivo per un loop di programma se errore gestione.
Errors (UDINT)	Numero di errori, incrementato ad ogni nuovo errore, raggiunto valore massimo riparte da 0.
TxFrame (STRING[8])	Contiene il frame inviato al dispositivo, può essere utilizzato in debug.
RxFrame (STRING[16])	Contiene il frame ricevuto dal dispositivo, può essere utilizzato in debug.
Temperature (REAL)	Valore di temperatura (°C).
Humidity (REAL)	Valore di umidità (%).

Codici di errore

In caso di errore si attiva l'uscita **Fault**, il valore di **Errors** si incrementa e con [SysGetLastError](#) è possibile rilevare il codice di errore.

- 10032010 Valore di **File** non definito.
- 10032050 Timeout esecuzione.
- 10032060 Errore case gestione lettura.
- 10032100 Errore controllo indirizzo del dispositivo HTemp.
- 10032120 Errore lettura valore di temperatura.
- 10032150 Errore lettura valore di umidità.
- 10013200~1 Errore nella gestione del protocollo di comunicazione.

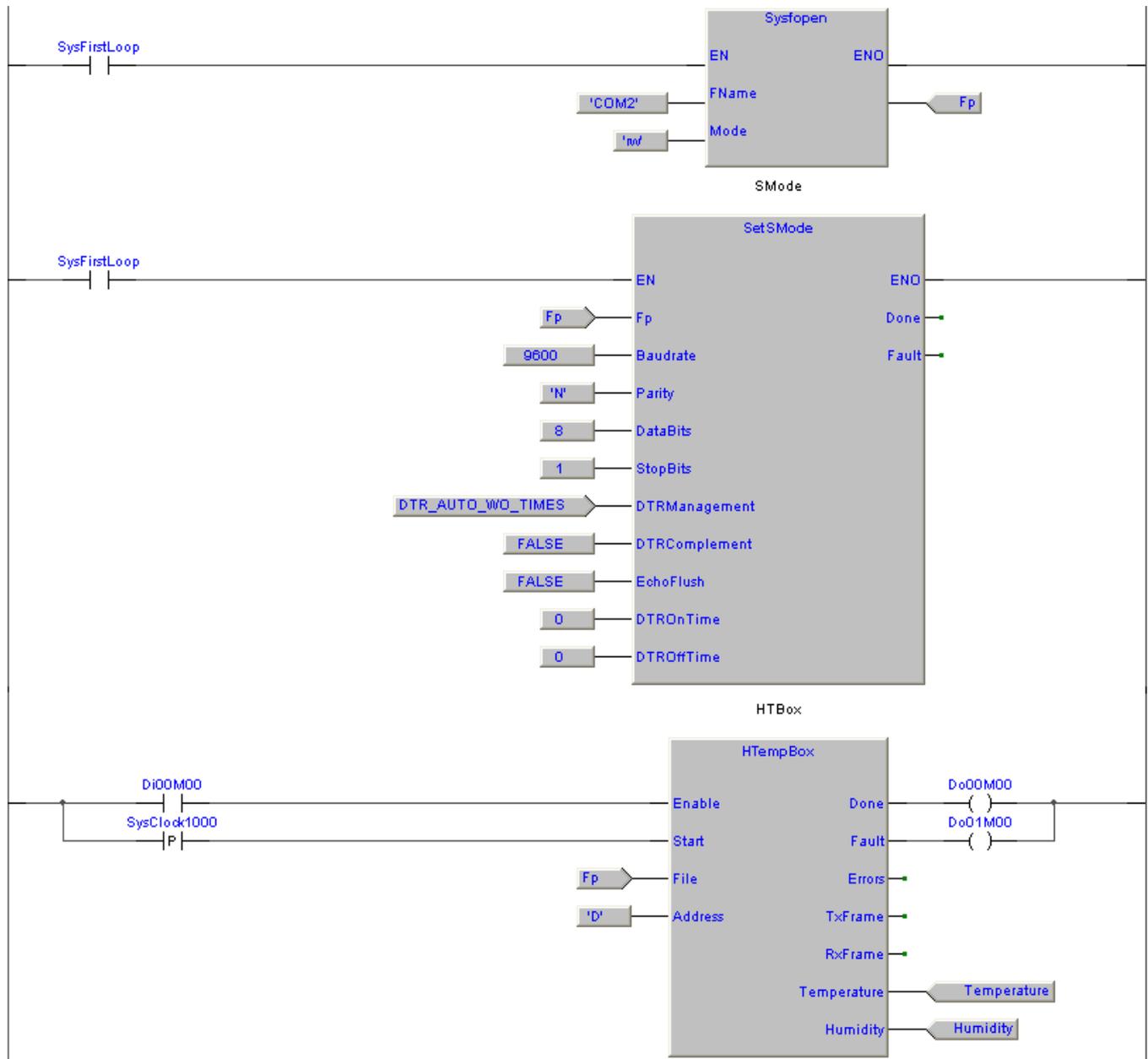
Esempi

Nell'esempio è gestita l'acquisizione del valore di temperatura e di umidità da un HTemp ogni secondo. Il valore di temperatura in gradi centigradi è ritornato nella variabile **Temperature** ed il valore di umidità in % è ritornato nella variabile **Humidity**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer
2	SMode	SetSMode	Auto	No	0	..	FB set serial mode data
3	HTBox	HTempBox	Auto	No	0	..	FB HTempBox data
4	Temperature	REAL	Auto	No	0	..	Temperature value (°C)
5	Humidity	REAL	Auto	No	0	..	Humidity value (%)

Esempio LD



7.22 Funzioni ed FB supporto protocollo NMEA (eNMEALib)

NMEA 0183 (O più comunemente NMEA) è uno standard di comunicazione di dati utilizzato soprattutto in nautica e nella comunicazione di dati satellitari GPS. L'ente che gestisce e sviluppa il protocollo è la National Marine Electronics Association.

Questo protocollo si basa sul principio che la fonte, detta talker, può soltanto inviare i dati (sentences) e la ricevente, detta listener, può soltanto riceverli.

La libreria **eNMEALib** fornisce una serie di funzioni e blocchi funzione per gestire le sentenze NMEA, in pratica è possibile realizzare programmi con l'ambiente di sviluppo LogicLab che si comportano come listeners di sentenze NMEA.

7.22.1 NMEASInterface, NMEA system interface

Type	Library	Version
FB	eNMEALib	SFR061A000

Questo blocco funzione gestisce l'interfaccia verso un dispositivo che invia sentenze NMEA connesso al dispositivo di I/O definito in **File**, questo è un blocco funzione protetto per utilizzarlo occorre richiedere il codice di protezione, vedi [protezione funzioni e blocchi funzione](#). E' comunque possibile utilizzarlo liberamente in modo test per 30 Min.



L'FB riceve le sentenze NMEA dal dispositivo, ne controlla il prefisso comparandolo con la stringa definita in **DID**, controlla se la sentenza ricevuta è corretta (Controllo sul CRC). L'uscita **Done** si attiva per un loop ad ogni ricezione di sentenza NMEA corretta.

L'FB ritorna un **NMEAID** che deve essere passato alle FB collegate (FB di gestione sentenze NMEA). L'uscita **RxString** riportano la stringa ricevuta dal dispositivo, in questo modo è possibile visualizzare in debug la comunicazione con il dispositivo permettendo di visualizzare eventuali errori.

- Enable** (BOOL) Abilitazione blocco funzione.
- File** (FILEP) Flusso dati **stream** ritornato dalla funzione **Sysfopen**.
- DID** (STRING[2]) Stringa di definizione prefisso dispositivo.
- Done** (BOOL) Attivo per un loop di programma su ricezione sentenza corretta.
- Fault** (BOOL) Attivo per un loop di programma se errore gestione.
- NMEAID** (UDINT) ID interfaccia con sistema NMEA da passare alle FB collegate.
- RxString** (STRING[82]) Contiene la stringa ricevuta dal dispositivo, può essere utilizzato in debug per verificare le sentenze ricevute.

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con [SysGetLastError](#) è possibile rilevare il codice di errore.

- 10017010 Valore di **File** non definito.
- 10017020 FB protetta, terminato tempo funzionamento in modo demo.
- 10017050 Timeout esecuzione.
- 10017070 Errore case gestione.
- 10017100~4 Errore ricezione sentenza NMEA.

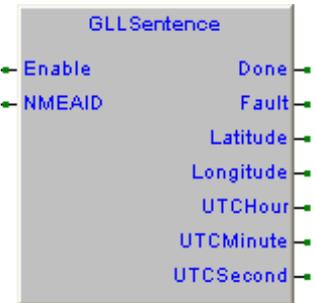
Type	Library	Version
FB	eNMEALib	SFR061A000

7.22.2 GLLSentence, Geographic Position sentence

Questo blocco funzione esegue la ricezione della sentenza GLL Geographic Position, si collega al blocco funzione **NMEASInterface** di gestione dispositivo NMEA. Occorre passare **NMEAID** in uscita dal blocco funzione di gestione dispositivo.

La sentenza GLL contiene le informazioni di latitudine, longitudine ora e fix, esempio di sentenza **\$IIGLL,4419.0173,N,00829.6653,E,084550.00,A,2*09**.

L'FB controlla correttezza dei campi della sentenza e ne estrae le informazioni di latitudine, longitudine e tempo. L'uscita **Done** si attiva per un loop ad ogni ricezione di sentenza GLL corretta.



- Enable** (BOOL) Abilitazione blocco funzione.
- NMEAID** (UDINT) ID interfaccia con sistema NMEA fornito in uscita dal blocco funzione **NMEASInterface**.
- Done** (BOOL) Attivo per un loop di programma su ricezione sentenza GLL corretta.
- Fault** (BOOL) Attivo per un loop di programma se errore sentenza.
- Latitude** (REAL) Valore di latitudine indicato nella sentenza, il valore è espresso in frazione di gradi. Valori positivi indicano latitudine nord, valori negativi latitudine sud.
- Longitude** (REAL) Valore di longitudine indicato nella sentenza, il valore è espresso in frazione di gradi. Valori positivi indicano latitudine est, valori negativi latitudine ovest.
- UTCHour** (USINT) Valore di ora UTC indicato nella sentenza.
- UTCMinute** (USINT) Valore di minuti UTC indicato nella sentenza.
- UTCSecond** (USINT) Valore di secondi UTC indicato nella sentenza.

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con **SysGetLastError** è possibile rilevare il codice di errore.

- 10018010 **NMEAID** non definito.
- 10018020 **NMEAID** non corretto.
- 10018100~2 Errore nel valore di latitudine.
- 10018200~2 Errore nel valore di longitudine.
- 10018300~2 Errore nel valore ora UTC.

Esempi

E' disponibile un programma di esempio Ptp123*000 che gestisce l'interfaccia verso un navigatore satellitare con l'interpretazione di alcune sentenze NMEA.

Nell'esempio riportato è gestita la ricezione di una sentenza GLL.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer
2	GLL	GLLSentence	Auto	No	0	..	GL sentence reception FB data
3	NMEAID	UDINT	Auto	No	0	..	NMEA interface ID
4	NMEARx	NMEASInterfa	Auto	No	0	..	NMEA device interface FB data
5	RxGPS	STRING	Auto	[32]		..	Rx data from GPS device
6	SMode	SetSMode	Auto	No	0	..	Serial mode FB data

Esempio LD

7.22.3 MWVSentence, Wind Speed and Angle sentence

Type	Library	Version
FB	eNMEALib	SFR061A000

Questo blocco funzione esegue la ricezione della sentenza MWV wind speed and angle, si collega al blocco funzione **NMEASInterface** di gestione dispositivo NMEA. Occorre passare **NMEAID** in uscita dal blocco funzione di gestione dispositivo.

La sentenza MWV contiene le informazioni di velocità e direzione del vento, esempio di sentenza **\$IIMWV,120.09,R,4.53,N,A*35**.

L'FB controlla correttezza dei campi della sentenza e ne estrae le informazioni di velocità e direzione. L'uscita **Done** si attiva per un loop ad ogni ricezione di sentenza MWV corretta.



- Enable** (BOOL) Abilitazione blocco funzione.
- NMEAID** (UDINT) ID interfaccia con sistema NMEA fornito in uscita dal blocco funzione **NMEASInterface**.
- Done** (BOOL) Attivo per un loop di programma su ricezione sentenza MWV corretta.
- Fault** (BOOL) Attivo per un loop di programma se errore sentenza.
- WSpeed** (REAL) Valore di velocità del vento (Nodi).
- WPAngle** (REAL) Valore angolo polare (Relative) direzione vento.

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con **SysGetLastError** è possibile rilevare il codice di errore.

- 10020010 **NMEAID** non definito.
- 10020020 **NMEAID** non corretto.
- 10020100 Errore nel valore di velocità vento.

7.23 Funzioni ed FB supporto inverter Power One (ePowerOneLib)

Power One è uno dei principali produttori mondiale di sistemi di alimentazione. Power One è di diritto anche nel settore delle energie alternative con applicazioni per sistemi eolici ed inverter fotovoltaici. Oggi una strategia convincente nell'ambito delle energie alternative non può prescindere dallo sviluppo di soluzioni per il risparmio energetico.

La linea di Inverter fotovoltaici Aurora, comprende sia modelli per la connessione in rete sia isolati, con o senza trasformatore e concepiti per applicazioni da esterno e da interno. Tutti i prodotti della gamma si posizionano per soluzioni di progetto e tecnologia costruttiva ai vertici del mercato e sono caratterizzati da elevatissima affidabilità, innovazione ed efficienza.

Inverter Aurora

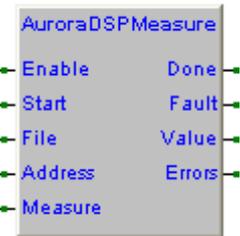
Alta efficienza di conversione e estrema facilità di manutenzione grazie alla possibilità di inserzione e disinserzione rapida dei moduli fotovoltaici. La scalabilità del sistema grazie alla architettura "Add-on" consente di coprire un'ampia gamma di applicazioni (fino a 300kW su singolo armadio).

Disponibile anche la versione senza trasformatore BT per connessione diretta ad una cella di media tensione (con trasf. MT).

Type	Library	Version
FB	ePowerOneLib	SFR062A000

7.23.1 AuroraDSPMeasure, Aurora measure request to DSP

Questo blocco funzione esegue la lettura delle misure dal DSP di un inverter Aurora della Power One, connesso al dispositivo di I/O definito in **File**. Questo è un blocco funzione protetto per utilizzarlo occorre richiedere il codice di protezione, vedi [protezione funzioni e blocchi funzione](#). E' comunque possibile utilizzarlo liberamente in modo test per 30 Min. Viene utilizzato il FB **CRCPolinomial** per il calcolo del CRC dei frame dati da e verso l'inverter.



La connessione con gli inverters è in RS485 multidrop, occorre definire in **Address** l'indirizzo dell'inverter con cui si vuole dialogare. In **Measure** occorre indicare il codice della misura da leggere (Vedi codici misura).

Attivando l'ingresso Start viene eseguita la lettura della misura indicata, terminata la lettura viene attivata per un loop l'uscita **Done**, in caso di errore esecuzione viene attivata per un loop l'uscita **Fault** ed incrementato il valore in **Errors**.

- Enable** (BOOL) Comando di abilitazione blocco funzione.
- Start** (BOOL) Comando di esecuzione lettura misura.
- File** (FILEP) Flusso dati **stream** ritornato dalla funzione **Sysfopen**.
- Address** (USINT) Indirizzo inverter (Range da 0 a 255).
- Measure** (USINT) Tipo misura da effettuare su inverter (Vedi codici misura).
- Done** (BOOL) Attivo per un loop al termine della esecuzione del comando.
- Fault** (BOOL) Attivo per un loop su errore esecuzione del comando.
- Value** (REAL) Valore misura acquisito da inverter (E' nella relativa unità di misura).
- Errors** (UDINT) Numero di errori, incrementato ad ogni nuovo errore, raggiunto valore massimo riparte da 0.

Codici misura

Nella variabile **Measure** occorre definire il codice della misura da effettuare dall'inverter secondo la tabella.

Codice	Descrizione	Um
1	Grid Voltage (For three-phases systems is the mean)	V
2	Grid Current (For three-phases systems is the mean)	A
3	Grid Power (For three-phases systems is the mean)	W
4	Frequency (For three-phases systems is the mean)	Hz
5	Vbulk (For Inverter with more Bulk is the sum)	V
6	Ileak (Dc/Dc)	A
7	Ileak (Inverter)	A
21	Inverter Temperature	°C
22	Booster Temperature	°C
23	Input 1 Voltage (Input Voltage for single channel module)	V
25	Input 1 Current (Input Current for single channel module)	A
26	Input 2 Voltage (Input Voltage for single channel module)	V
27	Input 2 Current (Input Current for single channel module)	A
28	Grid Voltage (Dc/Dc)	V
29	Grid Frequency (Dc/Dc)	Hz

Codice	Descrizione	Um
30	Isolation Resistance (Riso)	
31	Vbulk (Dc/Dc)	V
32	Average Grid Voltage (VgridAvg)	V
33	VbulkMid	V
34	Power Peak	W
35	Power Peak Today	W
36	Grid Voltage neutral	V
37	Wind Generator Frequency	Hz
38	Grid Voltage neutral-phase	V
39	Grid Current phase r	A
40	Grid Current phase s	A
41	Grid Current phase t	A
42	Frequency phase r	Hz
43	Frequency phase s	Hz
44	Frequency phase t	Hz
45	Vbulk +	V
46	Vbulk -	V
47	Supervisor Temperature	°C
48	Alim. Temperature	°C
49	Heat Sink Temperature	°C
61	Grid Voltage phase r	V
62	Grid Voltage phase s	V
63	Grid Voltage phase t	V

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con [SysGetLastError](#) è possibile rilevare il codice di errore.

- 10030010 Valore di **File** non definito.
- 10030020 FB protetta, terminato tempo funzionamento in modo demo.
- 10030050 Timeout esecuzione.
- 10030070 Errore case gestione.
- 10030100 Errore CRC risposta da inverter Aurora.
- 10030200 Errore ricezione "Transmission state" da inverter Aurora.
- 10030251 Errore da inverter Aurora "Command is not implemented".
- 10030252 Errore da inverter Aurora "Variable does not exist".
- 10030253 Errore da inverter Aurora "Variable value is out of range".
- 10030254 Errore da inverter Aurora "EEProm not accessible".
- 10030255 Errore da inverter Aurora "Not Toggled Service Mode".
- 10030256 Errore da inverter Aurora "Can not send the command to internal micro".
- 10030257 Errore da inverter Aurora "Command not Executed".
- 10030258 Errore da inverter Aurora "The variable is not available, retry".

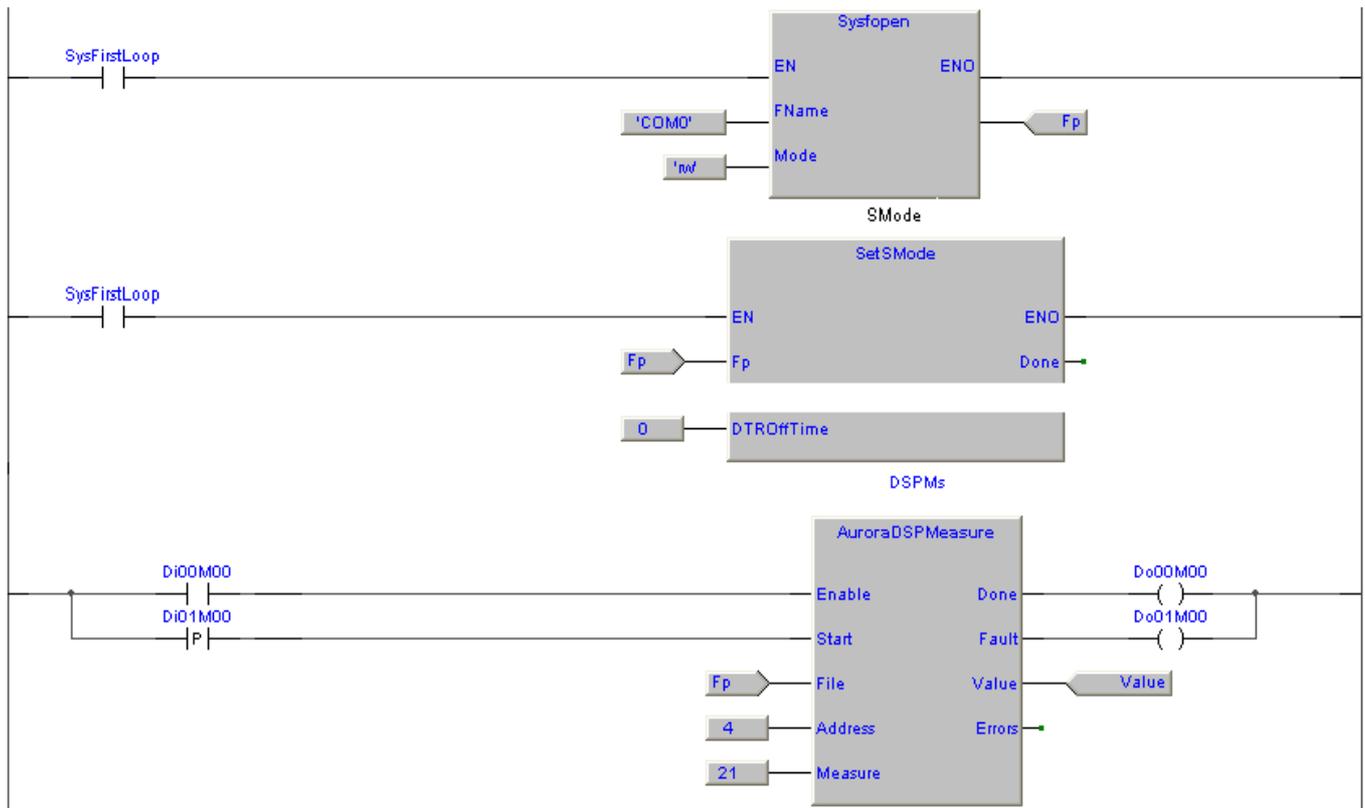
Esempi

Viene eseguita la lettura della misura 21 (Inverter Temperature) dall'inverter con indirizzo 4, il valore ritornato è trasferito nella variabile **Value**. Di default la porta seriale v  impostata a **19200, n 8, 1**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Terminal I/O file pointer
2	SMode	SetSMode	Auto	No	0	..	FB set serial mode
3	DSPMs	AuroraDSPMeasure	Auto	No	0	..	FB Aurora DSP measure
4	Value	REAL	Auto	No	0	..	Value read from inveter

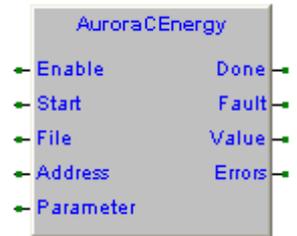
Esempio LD



Type	Library	Version
FB	ePowerOneLib	SFR062A000

7.23.2 AuroraCEnergy, Aurora cumulated energy reading

Questo blocco funzione esegue la lettura della energia generata da un inverter Aurora della Power One, connesso al dispositivo di I/O definito in **File**. Questo è un blocco funzione protetto per utilizzarlo occorre richiedere il codice di protezione, vedi [protezione funzioni e blocchi funzione](#). E' comunque possibile utilizzarlo liberamente in modo test per 30 Min. Viene utilizzato il FB [CRCPolinomial](#) per il calcolo del CRC dei frame dati da e verso l'inverter.



La connessione con gli inverters è in RS485 multidrop, occorre definire in **Address** l'indirizzo dell'inverter con cui si vuole dialogare. In **Parameter** occorre indicare il parametro da leggere (Vedi codici parametro).

Attivando l'ingresso Start viene eseguita la lettura della misura indicata, terminata la lettura viene attivata per un loop l'uscita **Done**, in caso di errore esecuzione viene attivata per un loop l'uscita **Fault** ed incrementato il valore in **Errors**.

- Enable** (BOOL) Comando di abilitazione blocco funzione.
- Start** (BOOL) Comando di esecuzione lettura misura.
- File** (FILEP) Flusso dati **stream** ritornato dalla funzione **Sysfopen**.
- Address** (USINT) Indirizzo inverter (Range da 0 a 255).
- Parameter** (USINT) Codice parametro da acquisire da inverter (Vedi codici parametro).
- Done** (BOOL) Attivo per un loop al termine della esecuzione del comando.
- Fault** (BOOL) Attivo per un loop su errore esecuzione del comando.
- Value** (UDINT) Valore parametro acquisito da inverter (E' nella relativa unità di misura).
- Errors** (UDINT) Numero di errori, incrementato ad ogni nuovo errore, raggiunto valore massimo riparte da 0.

Codici parametro

Nella variabile **Parameter** occorre definire il codice del parametro da leggere dall'inverter secondo la tabella.

Codice	Descrizione	Um
0	Daily energy	Kw
1	Weekly Energy	Kw
3	Month Energy (Energy from the first day of current calendar month)	Kw
4	Year Energy (Energy from the first day of current calendar year)	Kw
5	Total Energy (Total lifetime)	Kw
6	Partial Energy (Cumulated since reset)	Kw

Codici di errore

In caso di errore si attiva l'uscita **Fault**, con [SysGetLastError](#) è possibile rilevare il codice di errore.

- 10031010 Valore di **File** non definito.
- 10031020 FB protetta, terminato tempo funzionamento in modo demo.
- 10031050 Timeout esecuzione.
- 10031060 Codice parametro errato.
- 10031070 Errore case gestione.
- 10031100 Errore CRC risposta da inverter Aurora.

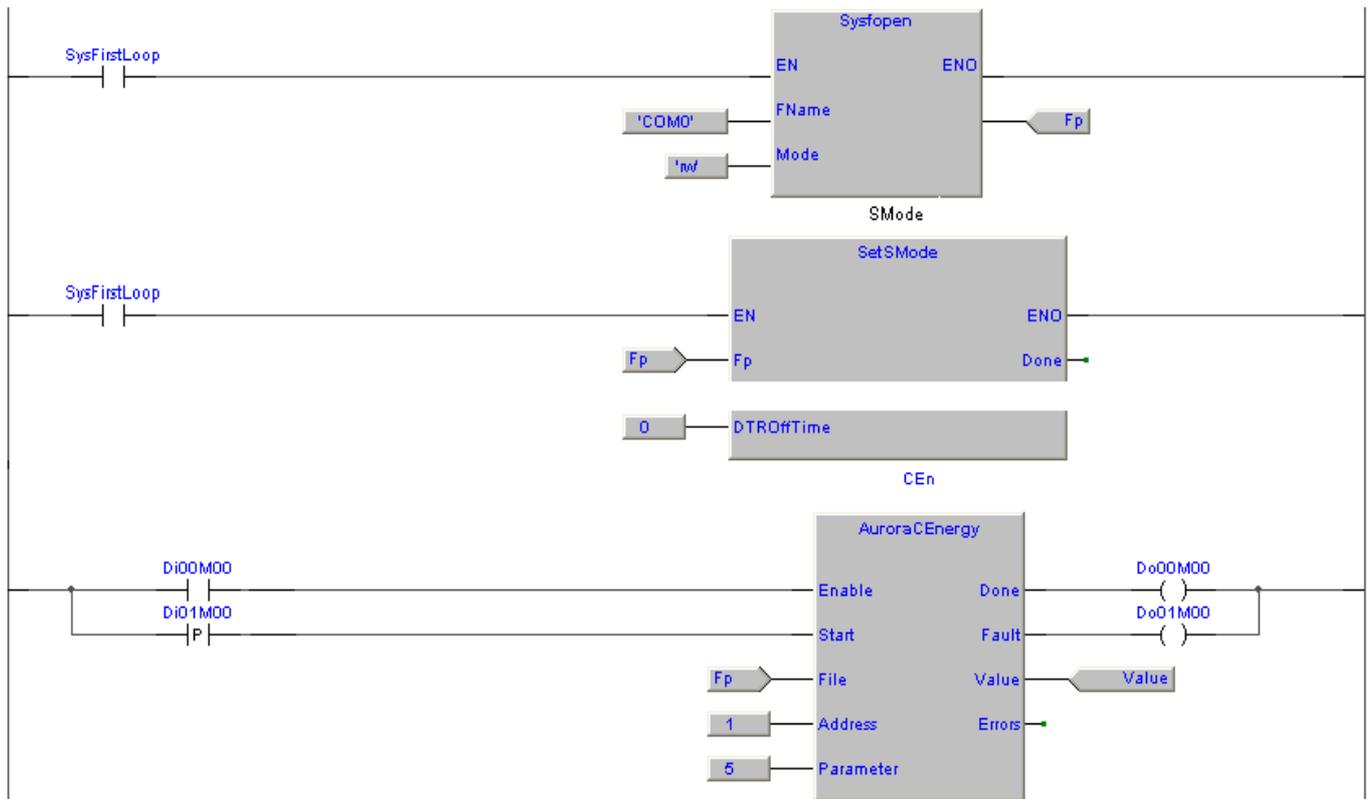
Esempi

Viene eseguita la lettura del totale energia prodotta dall'inverter con indirizzo 1, il valore ritornato è trasferito nella variabile **Value**. Di default la porta seriale v  impostata a **19200, n 8, 1**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Terminal I/O file pointer
2	SMode	SetSMode	Auto	No	0	..	FB set serial mode
3	Value	UDINT	Auto	No	0	..	Value read from inveter
4	CEn	AuroraCEnergy	Auto	No	0	..	FB Aurora cumulated energy

Esempio LD

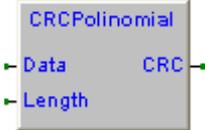


Type	Library	Version
FB	ePowerOneLib	SFR062A000

7.23.3 CRCPolynomial, CRC polinomial calculation

Questo blocco funzione esegue il calcolo del CRC su di un array di dati in accordo allo standard CCITT. Questo tipo di CRC è utilizzato come controllo sul frame dati da e verso l'inverter Aurora.

In **Data** occorre indicare l'indirizzo dell'array di dati su cui calcolare il CRC, **Length** indica la lunghezza dell'array in byte. In **CRC** è ritornato il valore del CRC calcolato.



Data (@USINT) Indirizzo array dati su cui eseguire il calcolo del CRC.

Length (USINT) Lunhezza array in bytes.

CRC (UINT) CRC calcolato.

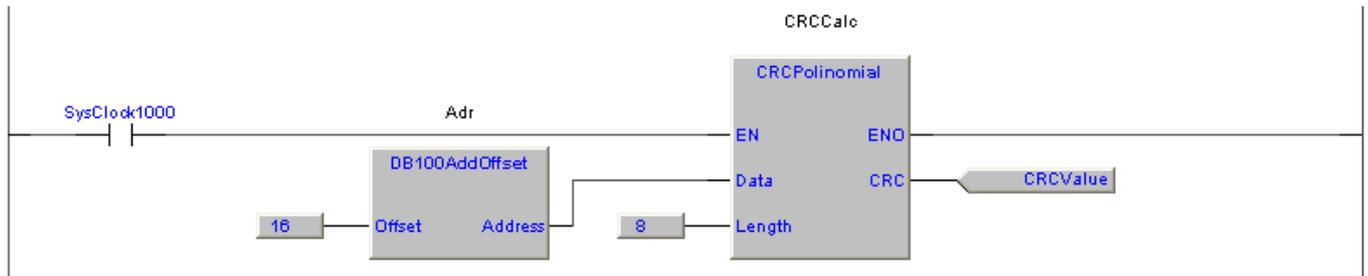
Esempi

Viene eseguito il calcolo del CRC su di un array di dati. Supponendo di avere un array di 8 bytes allocato in DB100 all'indirizzo 16 che contiene i valori 16#04, 16#3B, 16#1B, 16#00, 16#00, 16#00, 16#00, 16#00. Il CRC calcolato sarà 16#5AF0.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Adr	DB100AddOffset	Auto	No	0	..	FB DB100 address
2	CRCValue	WORD	Auto	No	0	..	CRC calculated
3	CRCCalc	CRCPolynomial	Auto	No	0	..	FB CRC calculation

Esempio LD



8 Tips and tricks

8.1 Swap variabile DWORD

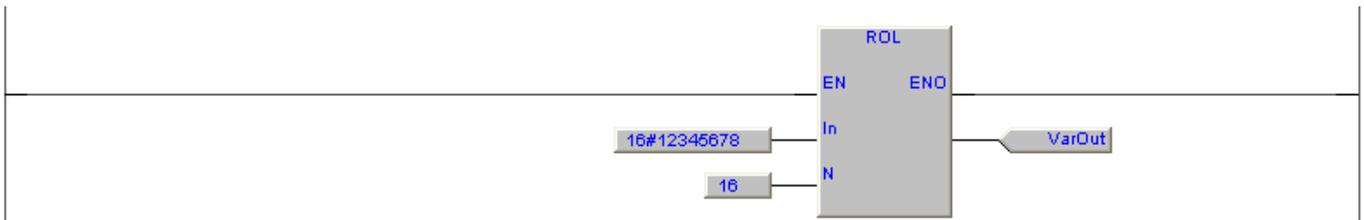
Ecco come utilizzare la funzione **ROL** per eseguire lo swap su variabile **DWORD**.

Nell'esempio il valore **16#12345678** viene trasformato nel valore **16#56781234** e trasferito nella variabile **VarOut**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	VarOut	DWORD	Auto	No	0	..	Output variable

Esempio LD



Esempio IL

```
LD 16#12345678
ROL 16
ST VarOut (* Output variable *)
```

Esempio ST

```
VarOut:=ROL(16#12345678, 16); (* Output variable *)
```

8.2 Swap variabile WORD

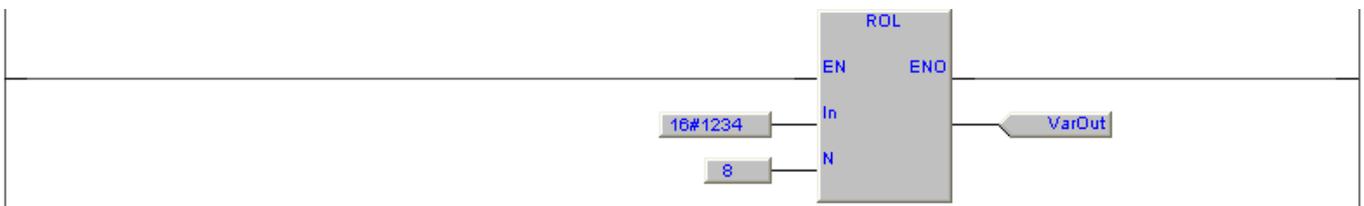
Ecco come utilizzare la funzione **ROL** per eseguire lo swap su variabile **WORD**.

Nell'esempio il valore **16#1234** viene trasformato nel valore **16#3412** e trasferito nella variabile **VarOut**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	VarOut	WORD	Auto	No	0	..	Output variable

Esempio LD



Esempio IL

```
LD 16#1234
ROL 8
ST VarOut (* Output variable *)
```

Esempio ST

```
VarOut:=ROL(16#1234, 8); (* Output variable *)
```

8.3 Swap variabile BYTE

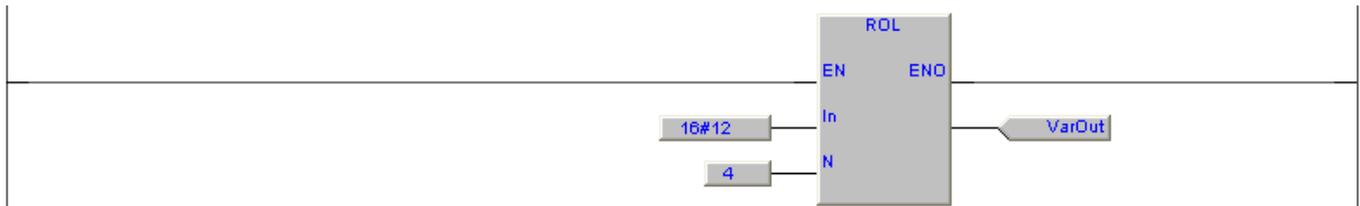
Ecco come utilizzare la funzione **ROL** per eseguire lo swap su variabile **BYTE**.

Nell'esempio il valore **16#12** viene trasformato nel valore **16#21** e trasferito nella variabile **VarOut**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	VarOut	BYTE	Auto	No	0	..	Output variable

Esempio LD



Esempio IL

```
LD 16#12
ROL 4
ST VarOut (* Output variable *)
```

Esempio ST

```
VarOut:=ROL(16#12, 4); (* Output variable *)
```

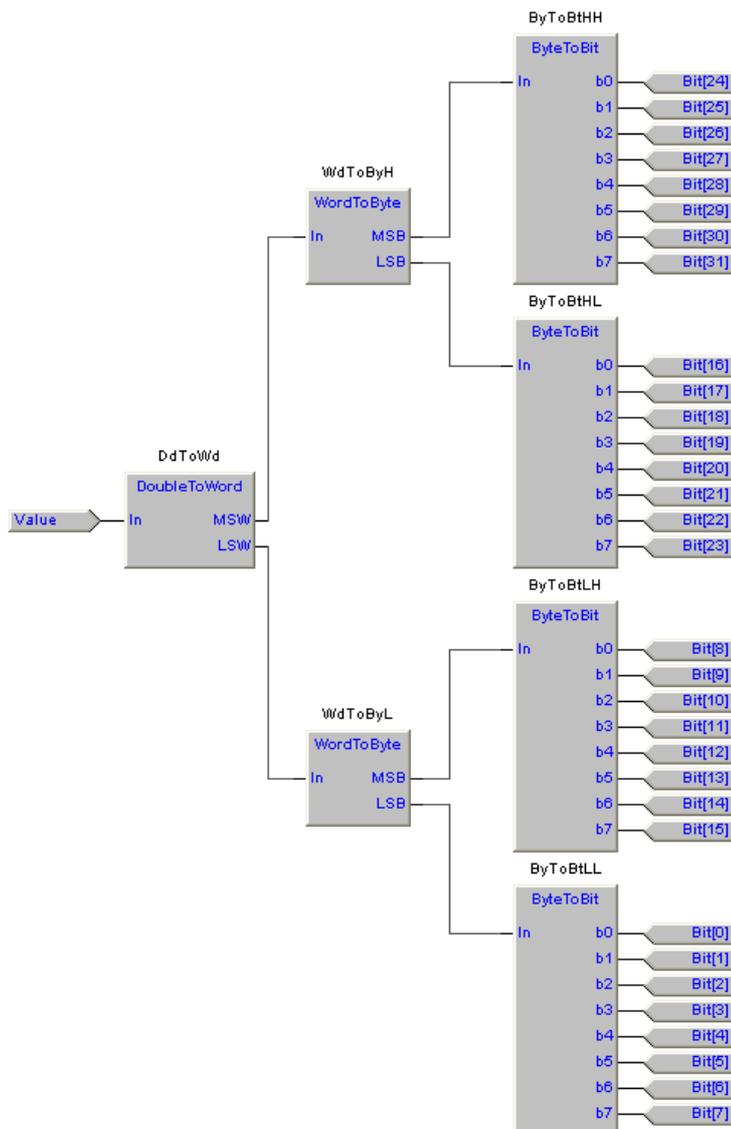
8.4 Espandere DWORD in 32 BOOL

Ecco un come utilizzando i blocchi funzione DoubleToWorld, WordToByte, ByteToBit sia possibile espandere una variabile **DWORD** in 32 variabili **BOOL**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	ByToBtHH	ByteToBit	Auto	No	0	..	Byte to bits
2	ByToBtHL	ByteToBit	Auto	No	0	..	Byte to bits
3	ByToBtLH	ByteToBit	Auto	No	0	..	Byte to bits
4	ByToBtLL	ByteToBit	Auto	No	0	..	Byte to bits
5	WdToByH	WordToByte	Auto	No	0	..	Word to bytes
6	WdToByL	WordToByte	Auto	No	0	..	Word to bytes
7	DdToWd	DoubleToWorld	Auto	No	0	..	Double to word
8	Bit	BOOL	Auto	[0..31]	FALSE,31 (0)	..	Bit status
9	Value	DWORD	Auto	No	0	..	Value to convert

Esempio FBD (Ptp114a200)



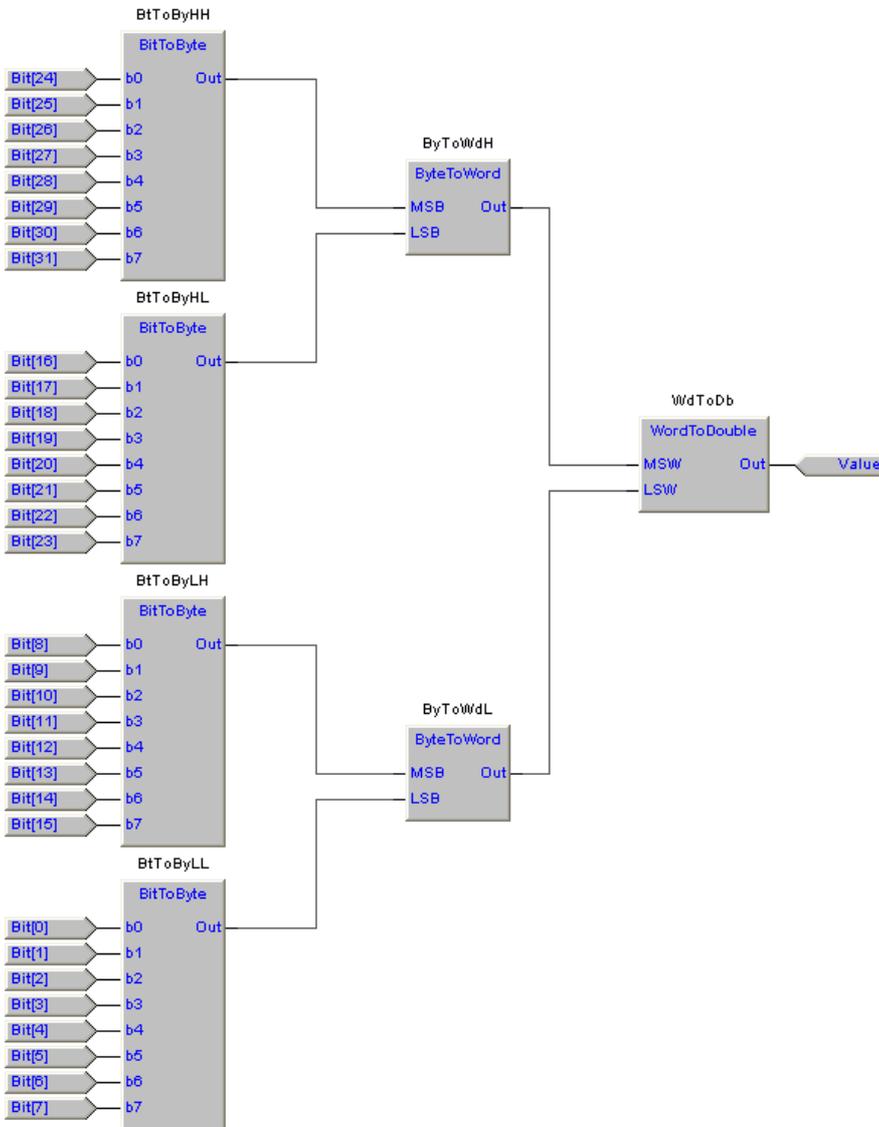
8.5 Comprimere 32 BOOL in DWORD

Ecco come utilizzando i blocchi funzione [BitToByte](#), [ByteToWord](#), [WordToDouble](#) sia possibile comprimere 32 variabili **BOOL** in una variabile **DWORD**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
2	BtToByHH	BitToByte	Auto	No	0	..	Bits to byte
3	BtToByLH	BitToByte	Auto	No	0	..	Bits to byte
4	BtToByLL	BitToByte	Auto	No	0	..	Bits to byte
5	ByToWdH	ByteToWord	Auto	No	0	..	Byte to word
6	ByToWdL	ByteToWord	Auto	No	0	..	Byte to word
7	WdToDb	WordToDouble	Auto	No	0	..	Word to double
8	Value	DWORD	Auto	No	0	..	Value to convert
9	Bit	BOOL	Auto	[0..31]	FALSE,31(0)	..	Bit status

Esempio FBD (Ptp114a200)



8.6 Definire caratteri ascii non stampabili

Nella gestione di protocolli di comunicazione e/o per impostare modalità di stampa su stampanti necessita eseguire l'output di caratteri ascii non stampabili, cioè caratteri con codici inferiori al 16#20 o superiori al 16#7F.

Per la definizione dei caratteri ascii stampabili basta includere tra apici singoli i caratteri (Esempio 'abcd').

Per i caratteri non stampabili, occorre anteporre al valore esadecimale del carattere il simbolo \$, quindi per il carattere di <STX> 16#02 avremo la definizione '\$02', per <ETX> '\$03' e così di seguito.

Ricordo che alcuni caratteri di controllo come il line feed codice 16#0A sia possibile definirli sia come '\$0A' che come '\$I'. Il carriage return codice 16#0D sia possibile definirlo sia come '\$OD' che come '\$r'. Riporto tabella esplicativa.

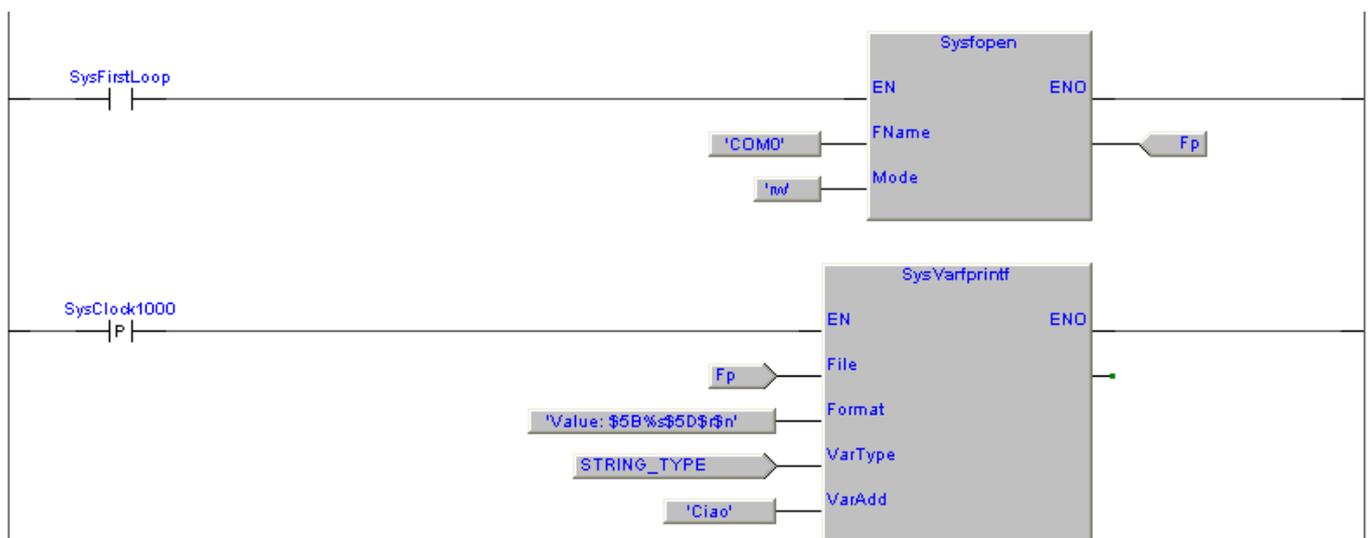
Sequenza	Significato	Esadecimale	Esempio
\$\$	Carattere \$	16#24	'I paid \$\$5 for this'
\$'	Apostrofo	16#27	'Enter \$'Y\$' for YES'
\$I	Line feed	16#0A	'next \$I line'
\$r	Carriage return	16#0D	'Hello\$r'
\$n	New line	16#0D0A	'This is a line\$n'
\$p	New page	16#0C	'last line \$p first line'
\$t	Tabulazione	16#09	'name\$tsize\$tdate'
\$hh		16#hh	'ABCD = \$41\$42\$43\$44'

Ecco un esempio di utilizzo della funzione **SysVarprintf** per definire oltre ai caratteri stampabili anche i caratteri non stampabili ed inviarli verso lo stream di uscita. In questo esempio viene inviato verso la porta seriale **COM0** la stringa [**Ciao**] seguita da carriage return e line feed.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer

Esempio LD



8.7 Rx/Tx dati su stream

Come si è visto con la funzione **Sysfopen** è possibile definire un collegamento tra una risorsa di I/O ed un flusso di dati **stream** da cui è possibile gestire la ricezione e/o la trasmissione di dati.

Per la ricezione dei dati in ingresso dallo stream si utilizza la funzione per controllo se dati presenti **SysGetIChars** e la funzione per la lettura degli stessi **Sysfgetc**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	RxString	STRING	Auto	[32]		..	Rx string
2	File	FILEP	Auto	No	0	..	File pointer
3	Ptr	@USINT	Auto	No	0	..	String pointer

Esempio ST

```
(* Rx data from stream. *)

Ptr:=ADR(RxString); (* String pointer *)

WHILE (TO_BOOL(SysGetIChars(File))) DO
  @Ptr:=TO_USINT(Sysfgetc(File)); (* Rx string *)
  Ptr:=Ptr+1; (* String pointer *)

  (* Check if string pointer overflow. *)

  IF (Ptr > ADR(RxString)+31) THEN EXIT; END_IF;
END_WHILE;
```

Per la trasmissione dei dati in uscita dallo stream si utilizza la funzione che controlla se spazio disponibile **SysGetOSpace**, e se lo spazio è sufficiente a contenere la stringa, o come nell'esempio, se il buffer di uscita è vuoto è possibile trasferire la stringa sullo stream con la funzione **SysVarfprintf**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	TxString	STRING	Auto	[32]		..	Rx string
2	File	FILEP	Auto	No	0	..	File pointer
3	i	UINT	Auto	No	0	..	Auxiliary counter

Esempio ST

```
(* Tx data to stream. *)

IF (TO_UDINT(SysGetOSpace(File)) = SysGetTxBSize(File)) THEN
  i:=TO_UINT(SysVarfprintf(File, '%s', STRING_TYPE, ADR(TxString)));
END_IF;
```

9 Appendici

9.1 Tabella istruzioni IL

Istruzione	Operandi	Descrizione
LD	Tutti	Carica il valore operando nell'accumulatore
LDN	Tutti	Carica il valore negato operando nell'accumulatore
ST	Tutti	Trasferisce il valore dell'accumulatore nell'operando
STN	Tutti	Trasferisce il valore negato dell'accumulatore nell'operando
S	BOOL	Setta l'operando (Accetta solo BOOL) se l'accumulatore è TRUE
R	BOOL	Resetta l'operando (Accetta solo BOOL) se l'accumulatore è TRUE
AND	Tutti meno REAL	AND a bit tra accumulatore e valore operando, risultato in accumulatore
ANDN	Tutti meno REAL	AND a bit tra accumulatore e valore negato operando, risultato in accumulatore
OR	Tutti meno REAL	OR a bit tra accumulatore e valore operando, risultato in accumulatore
ORN	Tutti meno REAL	OR a bit tra accumulatore e valore negato operando, risultato in accumulatore
XOR	Tutti meno REAL	XOR a bit tra accumulatore e valore operando, risultato in accumulatore
XORN	Tutti meno REAL	XOR a bit tra accumulatore e valore negato operando, risultato in accumulatore
NOT		Esegue l'inversione a bit del valore in accumulatore
ADD	Tutti meno BOOL	Somma tra accumulatore e valore operando, risultato in accumulatore
SUB	Tutti meno BOOL	Sottrazione tra accumulatore e valore operando, risultato in accumulatore
MUL	Tutti meno BOOL	Moltiplicazione tra accumulatore e valore operando, risultato in accumulatore
DIV	Tutti meno BOOL	Divisione tra accumulatore e valore operando, risultato in accumulatore
MOD	Tutti meno BOOL	Ritorna il modulo della divisione nell'accumulatore
GT	Tutti meno BOOL	Controlla se accumulatore > operando, risultato (BOOL) in accumulatore
GE	Tutti meno BOOL	Controlla se accumulatore >= operando, risultato (BOOL) in accumulatore
EQ	Tutti meno BOOL	Controlla se accumulatore = operando, risultato (BOOL) in accumulatore
NE	Tutti meno BOOL	Controlla se accumulatore <> operando, risultato (BOOL) in accumulatore
LE	Tutti meno BOOL	Controlla se accumulatore <= operando, risultato (BOOL) in accumulatore
LT	Tutti meno BOOL	Controlla se accumulatore < operando, risultato (BOOL) in accumulatore
JMP	Etichetta	Salta incondizionatamente su etichetta
JMPC	Etichetta	Salta su etichetta se accumulatore diverso da zero
JMPCN	Etichetta	Salta su etichetta se accumulatore uguale a zero
CAL	FB	Esegue incondizionatamente il blocco funzione
CALC	FB	Esegue blocco funzione se accumulatore diverso da zero
CALCN	FB	Esegue blocco funzione se accumulatore uguale a zero
RET		Ritorna incondizionatamente al programma che ha eseguito CALL
RETC		Ritorna al programma che ha eseguito CALL se accumulatore diverso da zero

9.2 Operatori linguaggio ST

Nella tabella seguente sono riportati gli operatori utilizzabili nel linguaggio ST. Gli operatori sono riportati in tabella in base alla loro priorità, dall'alto verso il basso, quindi le parentesi hanno priorità maggiore su tutti gli altri operatori.

Operazione	Simbolo	Esempio
Parenthesization	(Espressione)	
Function evaluation	Funzione(Argomenti)	LN(A), MAX(X,Y), etc.
Negation	-	
Complement	NOT	
Exponentiation	**	
Multiply	*	
Divide	/	
Modulo	MOD	
Add	+	
Subtract	-	
Comparison	< , > , <= , >=	
Equality	=	
Inequality	<>	
Boolean AND	&	
Boolean AND	AND	
Boolean Exclusive OR	XOR	
Boolean OR	OR	

9.3 Statements linguaggio ST

Nella tabella seguente sono riportati gli operatori utilizzabili nel linguaggio ST. Gli operatori sono riportati in tabella in base alla loro priorità, dall'alto verso il basso, quindi le parentesi hanno priorità maggiore su tutti gli altri operatori.

Operazione	Esempio
Assignment	<code>A:=B; CV:=CV+1; C:=SIN(X);</code>
FB Invocation and output usage	<code>CMD_TMR(IN:=%IX5, PT:=T#300ms); A:=CMD_TMR.Q;</code>
RETURN	<code>RETURN;</code>
IF	<code>D:=B*B-4*A*C; IF D < 0.0 THEN NROOTS:=0; ELSIF D=0.0 THEN NROOTS:=1; X1:=-B/(2.0*A); ELSE NROOTS:=2; X1:=(-B+SQRT(D))/(2.0*A); X2:=(-B-SQRT(D))/(2.0*A); END_IF;</code>
CASE	<code>TW:=BCD_TO_INT(THUMBWHEEL); TW_ERROR:=0; CASE TW OF 1,5: DISPLAY:=OVEN_TEMP; 2: DISPLAY:=MOTOR_SPEED; 3: DISPLAY:=GROSS-TARE; 4,6..10: DISPLAY:=STATUS(TW-4); ELSE DISPLAY:=0; TW_ERROR:=1; END_CASE; QW100:=INT_TO_BCD(DISPLAY);</code>
FOR	<code>J:=101; FOR I:=1 TO 100 BY 2 DO IF WORDS[I]='KEY' THEN J:=I; EXIT; END_IF; END_FOR;</code>
WHILE	<code>J:=1; WHILE J <= 100 & WORDS[J]<>'KEY' DO J:=J+2; END_WHILE;</code>
REPEAT	<code>J:=-1; REPEAT J:=J+2; UNTIL J=101 OR WORDS[J]='KEY' END_REPEAT ;</code>
EXIT	<code>EXIT;</code>
Empty Statement	<code>;</code>

9.4 Conversione tipo dati

Nella stesura di programmi capita frequentemente di dover eseguire una conversione di tipo (Detto anche **casting**), in pratica ci si riferisce all'operazione di passaggio di una variabile da un tipo di dato ad un altro. Nella programmazione IEC con LogicLab sono previste apposite funzioni di conversione di tipo, vediamole.

Name	Input type	Output type	Function
DINT_TO_INT	DINT	INT	Converts a long integer (32 bits, signed) into an integer (16 bits, signed)
INT_TO_DINT	INT	DINT	Converts an integer (16 bits, signed) into a long integer (32 bits, signed)
TO_BOOL	Any	BOOL	Converts any data type into a boolean
TO_SINT	Any	SINT	Converts any data type into a short integer (8 bits, signed)
TO_USINT	Any	USINT	Converts any data type into an unsigned short integer (8 bits, unsigned)
TO_INT	Any	INT	Converts any data type into an integer (16 bits, signed)
TO_UINT	Any	UINT	Converts any data type into an unsigned integer (16 bits, unsigned)
TO_DINT	Any	DINT	Converts any data type into a long integer (32 bits, signed)
TO_UDINT	Any	UDINT	Converts any data type into an unsigned long integer (32 bits, unsigned)
TO_REAL	Any	REAL	Converts any data type into a floating point (32 bits, signed)

Esempi

Conversione di una variabile di tipo DINT in una variabile di tipo USINT nei diversi linguaggi di programmazione. Naturalmente se il valore della variabile **VarDINT** supera il valore 255 (Limite della variabile **VarUSINT**), verrà ritornato il resto della divisione per il limite.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	VarUSINT	USINT	Auto	No	0	..	USINT variable
2	VarDINT	DINT	Auto	No	0	..	DINT variable

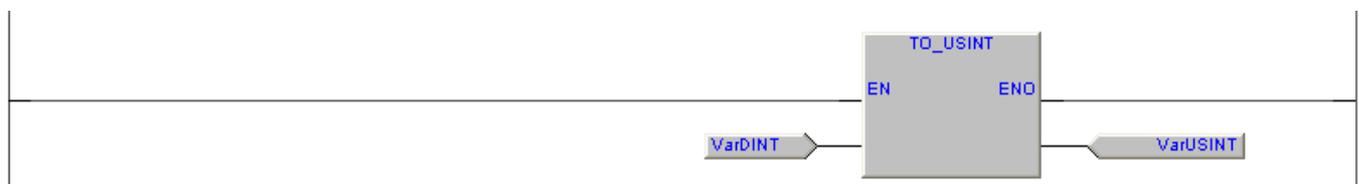
Esempio IL

```
LD VarDINT (* DINT variable *)
TO_USINT
ST VarUSINT (* USINT variable *)
```

Esempio FBD



Esempio LD



Esempio ST

```
VarUSINT:=TO_USINT(VarDINT); (* USINT variable *)
```

9.5 Errori di esecuzione

Alcune funzione e/o blocchi funzione possono avere errori di esecuzione, tipicamente è ritornata una flag di fault che indica l'errore. Per acquisire il codice di errore utilizzare la funzione [SysGetLastError](#), ritorna il codice dell'ultimo errore, va eseguita immediatamente dopo la funzione di cui si vuole controllare l'errore. Da debug è possibile controllare il valore della variabile [SysLastError](#) che sarà **0** se nessun errore oppure valore dell'ultimo errore riscontrato nella esecuzione del programma.

ID	Range errore	Funzione o FB
9958	9958000 ÷ 9958999	Sysfseek , file seek
9959	9959000 ÷ 9959999	Sysfilelength , file length
9960	9960000 ÷ 9960999	Sysrename , file rename
9961	9961000 ÷ 9961999	Sysremove , file remove
9962	9962000 ÷ 9962999	SysFOBfFlush , file output buffer flush
9963	9963000 ÷ 9963999	SysFOBfClear , file output buffer clear
9964	9964000 ÷ 9964999	SysFIBfClear , file input buffer clear
9965	9965000 ÷ 9965999	SysGetTxBSize , get file Tx output buffer size
9966	9966000 ÷ 9966999	SysGetRxBSize , get file Rx input buffer size
9967	9967000 ÷ 9967999	SysGetOSpace , get output available space on file
9968	9968000 ÷ 9968999	SysGetIChars , get input available characters from file
9969	9969000 ÷ 9969999	Sysfwrite , write data to file
9970	9970000 ÷ 9970999	Sysfread , read data from file
9971	9971000 ÷ 9971999	Sysfputc , put character to file
9972	9972000 ÷ 9972999	Sysfgetc , get character from file
9973	9973000 ÷ 9973999	Sysfclose , file close
9974	9974000 ÷ 9974999	SysIPReach , IP address is reachable
9975	9975000 ÷ 9975999	SysUDPSktRcv , UDP socket receive
9976	9976000 ÷ 9976999	SysUDPSktSend , UDP socket send
9977	9977000 ÷ 9977999	SysSktListen , socket listen
9978	9978000 ÷ 9978999	SysGetCrc , get CRC value
9979	9979000 ÷ 9979999	SysDMXMng , DMX management
9980	9980000 ÷ 9980999	SysGetEncoder , get encoder input
9981	9981000 ÷ 9981999	SysGetCounter , get counter
9982	9982000 ÷ 9982999	SysSetAnOut , set analog output
9983	9983000 ÷ 9983999	SysGetAnInp , get analog input
9984	9984000 ÷ 9984999	SysSetPhrDO , set peripheral digital output
9985	9985000 ÷ 9985999	SysGetPhrDI , get peripheral digital input
9986	9986000 ÷ 9986999	SysETimeToDate , epoch time to date conversion
9987	9987000 ÷ 9987999	SysDateToETime , date to epoch time conversion
9988	9988000 ÷ 9988999	SysPhrVWr , write variable to peripheral module
9989	9989000 ÷ 9989999	SysPhrVRd , read variable from peripheral module
9990	9990000 ÷ 9990999	SysPhrInfos , get infos from peripheral modules
9991	9991000 ÷ 9991999	SysPCodeAccept , accepts the protection code
9992	9992000 ÷ 9992999	SysSetSerialDTR , set DTR signal status

9993	9993000 ÷ 9993999	SysGetSerialCTS , get serial CTS signal status
9994	9994000 ÷ 9994999	SysSetSerialMode , set serial mode
9995	9995000 ÷ 9995999	SysGetSerialMode , get serial mode
9996	9996000 ÷ 9996999	Sysfopen , file open
9997	9997000 ÷ 9997999	SysVarsnprintf , variable print to string
9998	9998000 ÷ 9998999	SysVarfprintf , variable print to file
9999	9999000 ÷ 9999999	SysVarsscanf , extracts values from string
10000	10000000 ÷ 10000999	MDBRTUMASTER , modbus Rtu master
10001	10001000 ÷ 10001999	CPUModuleIO , CPU module I/O management
10002	10002000 ÷ 10002999	ModemCore , modem core management
10003	10003000 ÷ 10003999	ModemSMSReceive , receive a SMS message
10004	10004000 ÷ 10004999	ModemSMSRxCmd , receive a SMS command
10005	10005000 ÷ 10005999	ModemSMSSend , send a SMS message
10006	10006000 ÷ 10006999	SetSMMode , Set serial mode
10007	10007000 ÷ 10007999	ModbusRTUMaster , modbus RTU master
10008	10008000 ÷ 10008999	OwireMng , One-Wire management
10009	10009000 ÷ 10009999	OWRdIdentifier , One-Wire read ROM identifier
10010	10010000 ÷ 10010999	OWRdTemperature , One-Wire read temperature
10011	10011000 ÷ 10011999	IODataExchange , exchange data by using logic I/O
10012	10012000 ÷ 10012999	PIDMng , PID management
10013	10013000 ÷ 10013999	STESnmpAcq , STE termometer acquisition over SNMP
10014	10014000 ÷ 10014999	UDPDataTxfer , UDP data transfer
10015	10015000 ÷ 10015999	OWRdHumidity , One-Wire read humidity
10016	10016000 ÷ 10016999	IEC62056_21Rd , IEC62056-21 protocol read
10017	10017000 ÷ 10017999	NMEASInterface , NMEA system interface
10018	10018000 ÷ 10018999	GLLSentence , Geographic Position sentence
10019	10019000 ÷ 10019999	ModbusRTUSlave , modbus Rtu slave
10020	10020000 ÷ 10020999	MWVSentence , Wind Speed and Angle sentence
10030	10030000 ÷ 10030999	AuroraDSPMeasure , Aurora measure request to DSP
10031	10031000 ÷ 10031999	AuroraCEnergy , Aurora cumulated energy reading
10032	10032000 ÷ 10032999	HTempBox , HTemp Box2-485 acquisition

9.6 Tabella codici Ascii

9.6.1 Tabella codici ASCII standard

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	\$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(72	48	H	104	68	h
9	09	Horizontal tab	41	29)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	
29	1D	Group separator	61	3D	=	93	5D]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

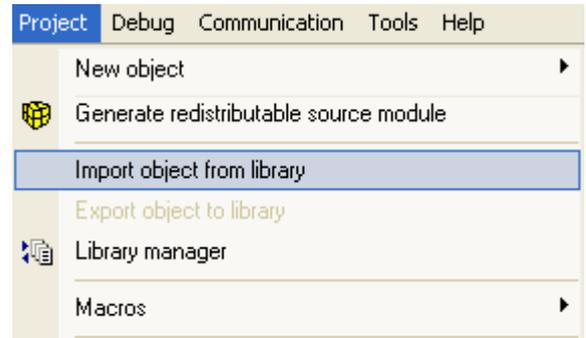
9.6.2 Tabella codici ASCII estesi

Dec	Hex	Char									
128	80	Ç	160	A0	á	192	C0	Ł	224	E0	α
129	81	ù	161	A1	í	193	C1	ł	225	E1	β
130	82	é	162	A2	ó	194	C2	ŧ	226	E2	Γ
131	83	â	163	A3	ú	195	C3	ł	227	E3	π
132	84	ä	164	A4	ñ	196	C4	—	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	+	229	E5	σ
134	86	ã	166	A6	ª	198	C6	ł	230	E6	μ
135	87	ç	167	A7	°	199	C7	ł	231	E7	τ
136	88	ê	168	A8	¿	200	C8	ł	232	E8	φ
137	89	ë	169	A9	ƒ	201	C9	ƒ	233	E9	θ
138	8A	è	170	AA	¬	202	CA	ł	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	ƒ	235	EB	δ
140	8C	î	172	AC	¼	204	CC	ł	236	EC	∞
141	8D	ì	173	AD	ı	205	CD	=	237	ED	∞
142	8E	Ä	174	AE	«	206	CE	ł	238	EE	ε
143	8F	Å	175	AF	»	207	CF	ł	239	EF	∩
144	90	É	176	B0	☐	208	D0	ł	240	FO	≡
145	91	æ	177	B1	☐	209	D1	ƒ	241	F1	±
146	92	Æ	178	B2	☐	210	D2	ƒ	242	F2	≥
147	93	ó	179	B3		211	D3	ł	243	F3	≤
148	94	ö	180	B4	ł	212	D4	ł	244	F4	[
149	95	ò	181	B5	ł	213	D5	ƒ	245	F5]
150	96	û	182	B6	ł	214	D6	ƒ	246	F6	÷
151	97	ù	183	B7	ƒ	215	D7	ł	247	F7	∞
152	98	ÿ	184	B8	ƒ	216	D8	ł	248	F8	°
153	99	ÿ	185	B9	ł	217	D9	ł	249	F9	•
154	9A	Û	186	BA		218	DA	ƒ	250	FA	·
155	9B	◊	187	BB	ƒ	219	DB	■	251	FB	√
156	9C	£	188	BC	ł	220	DC	■	252	FC	π
157	9D	¥	189	BD	ł	221	DD	■	253	FD	z
158	9E	ℳ	190	BE	ł	222	DE	■	254	FE	■
159	9F	f	191	BF	ł	223	DF	■	255	FF	□

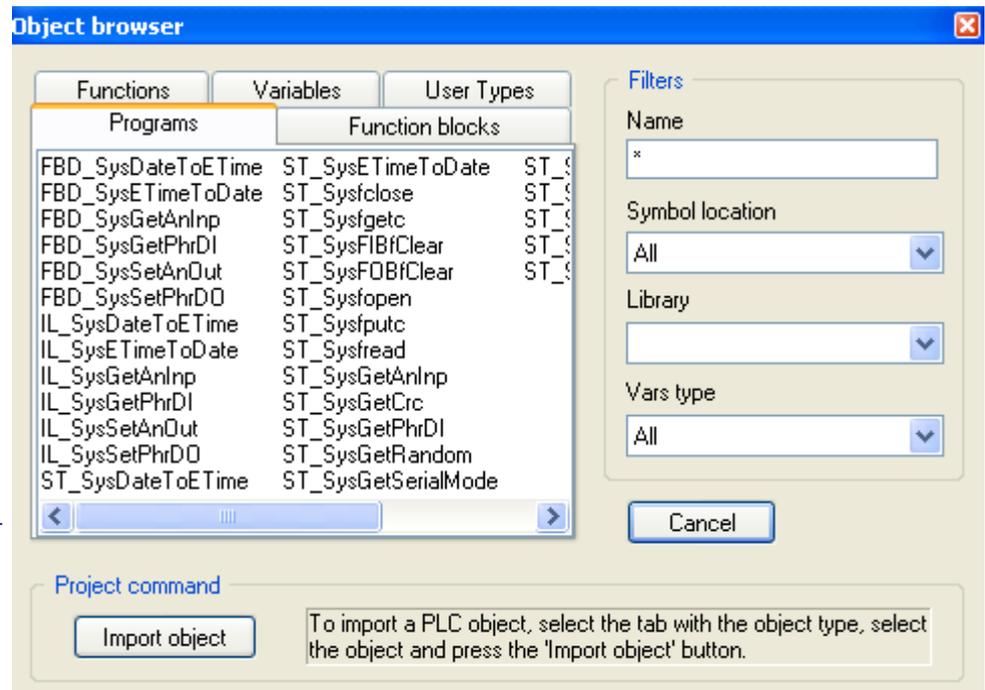
10 Esempi di programmazione

10.1 Biblioteca esempi

Per permettere all'utente di disporre di esempi da utilizzare per lo sviluppo dei propri programmi quasi tutti gli esempi riportati sul manuale sono forniti in programmi dimostrativi. I programmi dimostrativi sono codificati con il suffisso PTP, accanto ad ogni esempio. Se si desidera includere nel proprio progetto un file di esempio del manuale occorre dal menù **Project** scegliere la voce **Import object from library**. Si aprirà un dialog box che permette di selezionare la libreria da cui estrarre il programma da importare.



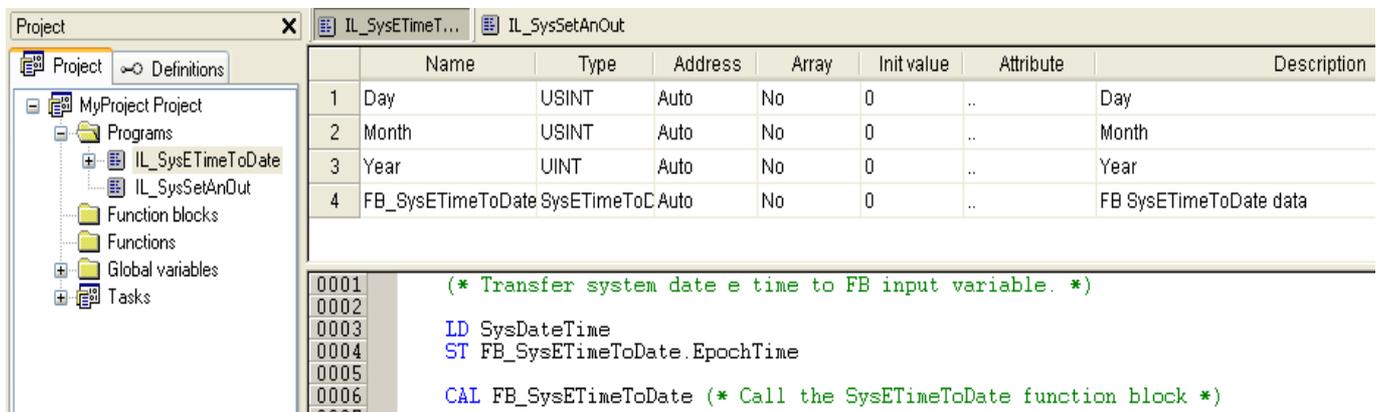
Scegliere il file di libreria desiderato (Esempio **Ptp116*000.pll**) si aprirà la finestra con l'elenco degli oggetti presenti all'interno da cui sarà possibile selezionare gli oggetti desiderati.



Evidenziando gli oggetti e agendo sul tasto **Import Object**, gli oggetti selezionati saranno inclusi nel proprio progetto.

Oltre ai programmi è possibile importare dalla libreria anche le variabili. In questo modo si potranno importare tutte le definizioni degli I/O logici come indicato nella [tabella di definizione](#).

Una volta inclusi nel progetto sarà possibile utilizzarli direttamente, oppure con con semplici operazioni di cut and paste incollare parti di codice sorgente dal progetto di esempio.



10.2 Definizioni I/O logici negli esempi

Tutti gli esempi riportati in questo manuale sono stati realizzati utilizzando una configurazione con un modulo CPU Slim line tipo MPS046A100 abbinato ad un modulo Mixed I/O PCB122*100. Tutti i 20 I/O del modulo con indirizzo 0 sono stati abbinati a variabili mnemoniche. Gli ingressi sono denominati **Di0xM00** e le uscite **Do0xM00** come evidente nella tabella di definizione

	Name	Type	Address	Group	Array	Init value	Attribute	Description
1	Di00M00	BOOL	%IX0.0		No	FALSE	..	Input 00, Module address 0
2	Di01M00	BOOL	%IX0.1		No	FALSE	..	Input 01, Module address 0
3	Di02M00	BOOL	%IX0.2		No	FALSE	..	Input 02, Module address 0
4	Di03M00	BOOL	%IX0.3		No	FALSE	..	Input 03, Module address 0
5	Di04M00	BOOL	%IX0.4		No	FALSE	..	Input 04, Module address 0
6	Di05M00	BOOL	%IX0.5		No	FALSE	..	Input 05, Module address 0
7	Di06M00	BOOL	%IX0.6		No	FALSE	..	Input 06, Module address 0
8	Di07M00	BOOL	%IX0.7		No	FALSE	..	Input 07, Module address 0
9	Di08M00	BOOL	%IX0.8		No	FALSE	..	Input 08, Module address 0
10	Di09M00	BOOL	%IX0.9		No	FALSE	..	Input 09, Module address 0
11	Di10M00	BOOL	%IX0.10		No	FALSE	..	Input 10, Module address 0
12	Di11M00	BOOL	%IX0.11		No	FALSE	..	Input 11, Module address 0
13	Do00M00	BOOL	%QX0.0		No	FALSE	..	Output 00, Module address 0
14	Do01M00	BOOL	%QX0.1		No	FALSE	..	Output 01, Module address 0
15	Do02M00	BOOL	%QX0.2		No	FALSE	..	Output 02, Module address 0
16	Do03M00	BOOL	%QX0.3		No	FALSE	..	Output 03, Module address 0
17	Do04M00	BOOL	%QX0.4		No	FALSE	..	Output 04, Module address 0
18	Do05M00	BOOL	%QX0.5		No	FALSE	..	Output 05, Module address 0
19	Do06M00	BOOL	%QX0.6		No	FALSE	..	Output 06, Module address 0
20	Do07M00	BOOL	%QX0.7		No	FALSE	..	Output 07, Module address 0

10.3 FB gestione lettura/scrittura registri

Per meglio comprendere il funzionamento delle funzioni di gestione I/O su dispositivo ecco l'implementazione di un blocco funzione per la lettura e scrittura di registri da porta seriale. Il programma in codice sorgente **FBRdWrRegisters** si trova nel **Ptp116*000**.

Definizione variabili

	Class	P..	Name	Type	Array	Init value	Attribute	Description
1	VAR		CaseNr	UINT	No	0	..	Case number
2	VAR		Ch	INT	No	0	..	Character
3	VAR_INPUT	0	Fp	FILEP	No	0	..	Terminal I/O pointer
4	VAR_INPUT	1	RdRegs	UINT	[0..3]	4(0)	..	Read registers
5	VAR_OUTPUT	0	WrRegs	UINT	[0..3]	4(0)	..	Write registers
6	VAR		RxValue	UINT	No	0	..	Received value
7	VAR		RxHexData	USINT	No	0	..	Received hexadecimal data
8	VAR		CaseBack	UINT	No	0	..	Case back
9	VAR		Digit	USINT	No	0	..	Digits counter
10	VAR		Address	UINT	No	0	..	Register address
11	VAR_EXTERNAL		SysTime	UDINT	No	0	..	System time
12	VAR		TimBuff	UDINT	No	0	..	Time buffer

Codice sorgente blocco funzione in ST

```
(* ***** *)
(* READ/WRITE REGISTERS FUNCTION BLOCK *)
(* ***** *)

(* ----- *)
(* TIMEOUT MANAGEMENT *)
(* ----- *)
(* If the command does not terminate in 1 Second it's aborted. *)

IF (CaseNr = 0) THEN TimBuff:=SysTime; END_IF;
IF (SysTime > (TimBuff+1000)) THEN CaseNr:=0; END_IF;

(* ##### *)
(* MANAGEMENT CASES *)
(* ##### *)

CASE (CaseNr) OF

  (* ----- *)
  (* WAITING READ/WRITE COMMAND *)
  (* ----- *)
  (* The read command is "?Rxx" followed by <CR>. *)

  16#0000:
  WHILE (TO_BOOL(SysGetIChars(Fp))) DO
    Ch:=Sysfgetc(Fp); (* Character *)

    (* The read command starts with '?' character. *)

    IF (Ch = 63) THEN CaseNr:=16#0100; RETURN; END_IF;

    (* The write command starts with '!' character. *)

    IF (Ch = 33) THEN CaseNr:=16#0110; RETURN; END_IF;
  END_WHILE;

  (* ----- *)
  (* MANAGE THE READ COMMAND *)
  (* ----- *)
  (* The read command proceed with 'R' character and with the register *)
  (* adress. *)

  16#0100:
  WHILE (TO_BOOL(SysGetIChars(Fp))) DO
    IF (Sysfgetc(Fp) = 82) THEN
      CaseBack:=CaseNr+1; (* Case back *)
      CaseNr:=16#0200; (* Set the case number to receive address *)
      RETURN;
    END_IF;
```

```

END_WHILE;

(* ----- *)
(* Print out the register address and value. *)

16#0101:
Address:=RxValue; (* Register address *)
IF (Address <= 3) THEN
    Ch:=SysVarfprintf(Fp, 'Address:%04X ', UINT_TYPE, ADR(Address));
    Ch:=SysVarfprintf(Fp, 'Value:%04X$r$n', UINT_TYPE, ADR(RdRegs[Address]));
END_IF;
CaseNr:=0; (* Case number *)

(* ----- *)
(* MANAGE THE WRITE COMMAND *)
(* ----- *)
(* The write command proceed with 'W' character, the register address *)
(* and the value to be written. *)

16#0110:
WHILE (TO_BOOL(SysGetIChars(Fp))) DO
    IF (Sysfgetc(Fp) = 87) THEN
        CaseBack:=CaseNr+1; (* Case back *)
        CaseNr:=16#0200; (* Set the case number to receive address *)
        RETURN;
    END_IF;
END_WHILE;

(* ----- *)
(* Receive the value. *)

16#0111:
Address:=RxValue; (* Register address *)
CaseBack:=CaseNr+1; (* Case back *)
CaseNr:=16#0200; (* Set the case number to receive value *)

(* ----- *)
(* Print out the register address and value. *)

16#0112:
IF (Address <= 3) THEN
    Ch:=SysVarfprintf(Fp, 'Address:%04X ', UINT_TYPE, ADR(Address));
    Ch:=SysVarfprintf(Fp, 'Value:%04X$r$n', UINT_TYPE, ADR(RxValue));
    RdRegs[Address]:=RxValue;
END_IF;
CaseNr:=0; (* Case number *)

(* ----- *)
(* MANAGE THE DATA RECEPTION *)
(* ----- *)
(* In the following cases the data is received, reception ends when a *)
(* ' ' or '\r' is been received, or if the maximum of an integer (4 *)
(* digits) is received. *)
(* ----- *)
(* Initialize the data reception. *)

16#0200:
Digit:=0; (* Digits counter *)
RxValue:=0; (* Received value *)
CaseNr:=CaseNr+1; (* Case number *)

(* ----- *)
(* Data reception. *)

16#0201:
WHILE (TO_BOOL(SysGetIChars(Fp))) DO

    Ch:=Sysfgetc(Fp); (* Character *)

    (* Skips the ' ' characters before the number. *)

    IF ((Digit =0) AND (Ch = 32)) THEN RETURN; END_IF;

    (* If ' ' or '\r' is been received continue. *)

    IF (Ch = 13) THEN CaseNr:=CaseBack; RETURN; END_IF;
    IF (Ch = 32) THEN CaseNr:=CaseBack; RETURN; END_IF;

```

```
(* Check if is an hexadecimal character, if not is an error. *)

IF ((Ch < 16#30) OR (Ch > 16#39)) AND ((Ch < 16#41) OR (Ch > 16#46)) THEN
    CaseNr:=0; (* Case number *)
    RETURN;
END_IF;

(* Convert the received character to a number. *)

IF ((Ch >= 16#30) AND (Ch <= 16#39)) THEN RxHexData:=TO_USINT(Ch)-16#30; END_IF;
IF ((Ch >= 16#41) AND (Ch <= 16#46)) THEN RxHexData:=TO_USINT(Ch)-16#37; END_IF;

(* Transfer data received in RxValue. *)

CASE (Digit) OF
    16#00: RxValue:=RxHexData; (* Received value *)
    16#01: RxValue:=(RxValue*16)+RxHexData; (* Received value *)
    16#02: RxValue:=(RxValue*16)+RxHexData; (* Received value *)
    16#03: RxValue:=(RxValue*16)+RxHexData; (* Received value *)
ELSE
    CaseNr:=0; (* Case number *)
    RETURN;
END_CASE;

    Digit:=Digit+1; (* Digits counter *)
END_WHILE;

END_CASE;

(* [End of file] *)
```

10.3.1 Utilizzo FB Lettura/Scrittura registri

Per utilizzare il blocco funzione è possibile realizzare un semplice programma in ST che lo richiama.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer
2	FBData	RWRegisters	Auto	No	0	..	FB data
3	RdRegs	UINT	Auto	[0..3]	4(0)	..	Read registers
4	WrRegs	UINT	Auto	[0..3]	4(0)	..	Write registers
5	i	USINT	Auto	No	0	..	Aux counter

Esempio ST

```
(* ***** *)
(* READ/WRITE REGISTERS FUNCTION BLOCK USE *)
(* ***** *)

(* Here the COM0 port is opened in read/write. *)

IF (FBData.Fp = NULL) THEN
    FBData.Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
END_IF;

(* If the COM0 port is opened the function block is executed. *)

IF (FBData.Fp <> NULL) THEN
    FBData();
END_IF;

(* Copy the read/write registers. *)

FOR i:=0 TO (3) BY 1 DO
    FBData.RdRegs[i]:=RdRegs[i];
    WrRegs[i]:=FBData.WrRegs[i];
END_FOR;

(* [End of file] *)
```

10.4 FB gestione ciclo On/Off con tempi random

Esempio di blocco funzione che utilizzando le funzioni [SysGetSysTime](#) e [SysGetRandom](#) permette di realizzare la gestione di una uscita lampeggiante con tempi di On/Off variabili in modo random entro tempi minimo e massimo definiti. Il programma in codice sorgente **FBOffCycle** si trova nel **Ptp116*000**.

Definizione variabili

	Class	P..	Name	Type	Array	Init value	Attribute	Description
1	VAR		DelayTm	UDINT	No	0	..	Delay time (uS)
2	VAR		TimeBf	UDINT	No	0	..	Time buffer (uS)
3	VAR		TimePsd	UDINT	No	0	..	Time passed (uS)
4	VAR_INPUT	1	MinOffTime	UDINT	No	0	..	Minimum off time (mS)
5	VAR_INPUT	2	MaxOffTime	UDINT	No	0	..	Maximum off time (mS)
6	VAR_INPUT	3	MinOnTime	UDINT	No	0	..	Minimum on time (mS)
7	VAR_INPUT	4	MaxOnTime	UDINT	No	0	..	Maximum on time (mS)
8	VAR_OUTPUT	0	Out	BOOL	No	FALSE	..	Output status
9	VAR_OUTPUT	1	Delay	UDINT	No	0	..	Current delay time (mS)
10	VAR_INPUT	0	Enable	BOOL	No	FALSE	..	FB enable

Codice sorgente blocco funzione in ST

```
(* ----- *)
(* GESTIONE ABILITAZIONE *)
(* ----- *)
(* Esego gestione abilitazione blocco funzione. *)

IF (NOT(Enable)) THEN
    Out:=FALSE; (* Output status *)
    TimeBf:=SysGetSysTime(TRUE); (* Time buffer (uS) *)
    Delay:=MinOffTime; (* Current delay time (mS) *)
    DelayTm:=Delay*1000; (* Delay time (uS) *)
    RETURN;
END_IF;

(* ----- *)
(* GESTIONE TEMPORIZZAZIONE ON/OFF *)
(* ----- *)
(* Esego controllo se variazione stato uscita ed eseguo salvataggio tempo e *)
(* calcolo del relativo tempo di attesa. *)

TimePsd:=SysGetSysTime(TRUE)-TimeBf; (* Time passed (uS) *)
TimeBf:=SysGetSysTime(FALSE); (* Time buffer (uS) *)

IF (DelayTm > TimePsd) THEN
    DelayTm:=DelayTm-TimePsd; (* Delay time (uS) *)
ELSE
    (* Esego inversione uscita e calcolo il relativo tempo di ritardo. *)

    Out:=Out XOR TRUE; (* Output status *)
    IF (NOT(Out)) THEN
        (* Calcolo tempo di off, viene utilizzata la formula: *)
        (* TOff=MinOffTime+((MaxOffTime-MinOffTime)*Numero Random) *)

        DelayTm:=(MinOffTime*1000)+((MaxOffTime-MinOffTime)*TO_UDINT(SysGetRandom(TRUE)*1000)); *)
    ELSE
        (* Calcolo tempo di on, viene utilizzata la formula: *)
        (* TOn=MinOnTime+((MaxOnTime-MinOnTime)*Numero Random) *)

        DelayTm:=(MinOnTime*1000)+((MaxOnTime-MinOnTime)*TO_UDINT(SysGetRandom(TRUE)*1000));
    END_IF;

    (* Esego calcolo valore tempo attuale in temporizzazione. *)

    Delay:=DelayTm/1000; (* Current delay time (mS) *)
END_IF;

(* [End of file] *)
```

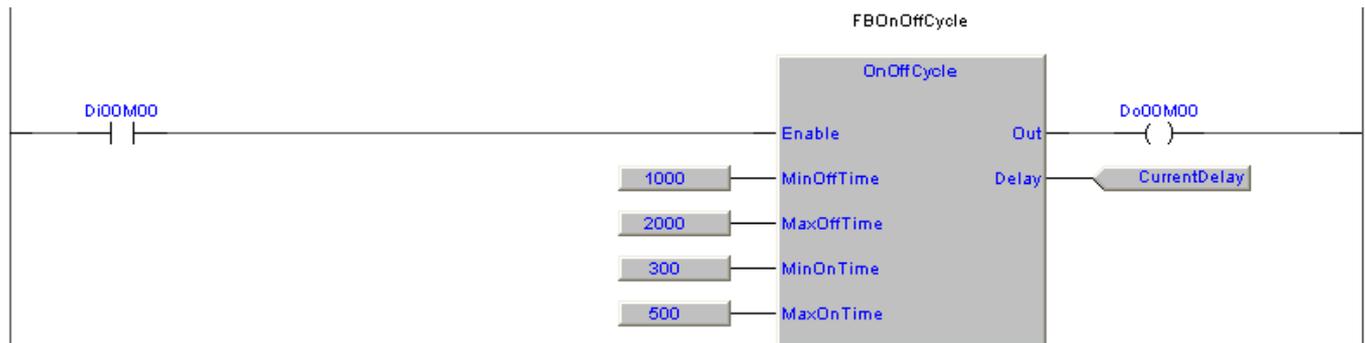
10.4.1 Utilizzo FB ciclo On/Off con tempi random

Per utilizzare il blocco funzione è possibile realizzare un semplice programma in LD che lo richiama. Nell'esempio che riporto avremo il lampeggio della uscita **Do00M00** con tempi di **Off** variabile tra 1 e 2 secondi, mentre il tempo di **On** è variabile tra 300 e 500 mS.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	CurrentDelay	UDINT	Auto	No	0	..	
2	FBOnOffCycle	OnOffCycle	Auto	No	0	..	

Esempio LD



10.5 Blocco funzione MiniDMX

Ecco un esempio su come realizzare un blocco funzione per la gestione di un mini dimmer a 3 canali RGB utilizzando il linguaggio ST. Il prodotto in questione utilizza 4 indirizzi DMX:

- 1) Preset uscita comando colore rosso.
- 2) Preset uscita comando colore verde.
- 3) Preset uscita comando colore blu.
- 4) Preset valore dimmer su tutte le uscite.



Il blocco funzione viene chiamato **MiniDMX**, sulla destra è possibile vedere come verrà visualizzato il blocco funzione nel proprio programma LD. Il programma in codice sorgente del blocco funzione e del programma di esempio si trova nel **Ptp117*000**.

Sull'ingresso **Address** occorre definire l'indirizzo del dimmer, cioè l'indirizzo DMX a cui corrisponde il preset del colore rosso, gli altri valori saranno mappati sui 3 indirizzi successivi. Sull'ingresso **DataPtr** occorre trasferire l'indirizzo dell'array dati DMX.

Red, definisce il valore dell'uscita di colore rosso. **Green**, definisce il valore dell'uscita di colore verde. **Blue**, definisce il valore dell'uscita di colore blu. **Dimmer** definisce il valore di dimmering delle uscite.

Definizione variabili

	Class	P..	Name	Type	Array	Init value	Attribute	Description
1	VAR		Ptr	@USINT	No	0	..	Pointer array comandi DMX
2	VAR_INPUT	0	Address	UINT	No	0	..	Indirizzo modulo MiniDMX
3	VAR_INPUT	1	DataPtr	@USINT	No	0	..	Address array comandi DMX
4	VAR_INPUT	2	Red	USINT	No	0	..	Preset colore rosso
5	VAR_INPUT	3	Green	USINT	No	0	..	Preset colore verde
6	VAR_INPUT	4	Blue	USINT	No	0	..	Preset colore blu
7	VAR_INPUT	5	Dimmer	USINT	No	0	..	Preset dimmer

Codice sorgente ST (Ptp117a100)

```
(* ***** *)
(* FB GESTIONE DIMMER DMX (MiniDMX) *)
(* ***** *)
(* Il driver MiniDMX gestisce 3 uscite RGB. Il driver utilizza 4 canali DMX, *)
(* *)
(* 1: Comando segnale rosso *)
(* 2: Comando segnale verde *)
(* 3: Comando segnale blu *)
(* 4: Comando dimmer su tutti i segnali *)
(* ----- *)
(* Preset pointer su array comandi DMX. *)

Ptr:=TO_UDINT(DataPtr)+Address-1; (* Pointer array comandi DMX *)
@Ptr:=Red; (* Preset colore rosso *)

Ptr:=Ptr+1; (* Pointer array comandi DMX *)
@Ptr:=Green; (* Preset colore verde *)

Ptr:=Ptr+1; (* Pointer array comandi DMX *)
@Ptr:=Blue; (* Preset colore blu *)

Ptr:=Ptr+1; (* Pointer array comandi DMX *)
@Ptr:=Dimmer; (* Preset dimmer *)

(* [End of file] *)
```

10.5.1 Programma DMXUse

Questo programma istanzia e gestisce il blocco funzione **SysDMXMng**. Viene valorizzata la variabile globale **DMXDataPtr** che contiene l'indirizzo dell'array dati DMX che dovrà essere passato al blocco funzione di gestione **MiniDMX**.

Definizione variabili globali

	Name	Type	Address	Group	Array	Init value	Attribute	Description
1	DMXData	USINT	Auto		[0..9]	10(0)	..	DMX data array
2	Add1Red	USINT	Auto		No	0	..	MiniDMX con Address 1 (Rosso)
3	Add1Green	USINT	Auto		No	0	..	MiniDMX con Address 1 (Verde)
4	Add1Blue	USINT	Auto		No	0	..	MiniDMX con Address 1 (Blu)
5	Add1Dimmer	USINT	Auto		No	0	..	MiniDMX con Address 1 (Dimmer)
6	DMXDataPtr	@USINT	Auto		No	0	..	DMX data pointer
7	Add5Red	USINT	Auto		No	0	..	MiniDMX con Address 5 (Rosso)
8	Add5Green	USINT	Auto		No	0	..	MiniDMX con Address 5 (Verde)
9	Add5Blue	USINT	Auto		No	0	..	MiniDMX con Address 5 (Blu)
10	Add5Dimmer	USINT	Auto		No	0	..	MiniDMX con Address 5 (Dimmer)

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBDMX	SysDMXMng	Auto	No	0	..	FB gestione protocollo DMX

Codice sorgente ST (Ptp117a100)

```
(* ----- *)
(* ESEGUO APERTURA PORTA SERIALE *)
(* ----- *)
(* Here the COM1 port is opened in read/write. *)

IF (FBDMX.File = NULL) THEN
  FBDMX.File:=Sysfopen('COM1', 'rw'); (* Port COM1 file pointer *)
END_IF;

(* Valorizzo pointer dati DMX per passarlo alle FB gestione DMX. *)

DMXDataPtr:=ADR(DMXData); (* DMX data pointer *)

(* ----- *)
(* ESEGUO GESTIONE PROTOCOLLO DMX *)
(* ----- *)
(* Gestione protocollo DMX. *)

FBDMX.Start:=TRUE; (* Start *)
FBDMX.Status:=0; (* Status byte *)
FBDMX.Devices:=10; (* Number of devices *)
FBDMX.Delay:=100; (* Interframe delay (mSec) *)
FBDMX.DataPtr:=DMXDataPtr; (* Data array pointer *)
FBDMX(); (* FB gestione protocollo DMX *)

(* [End of file] *)
```

Questo programma utilizza due blocchi funzione **MiniDMX** a cui viene passata la variabile globale **DMXDataPtr** che contiene l'indirizzo dell'array dati DMX. Impostando i valori desiderati nelle variabili globali di preset essi verranno trasferiti nei relativi apparecchi MiniDMX con l'indirizzo definito

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Dmx1	MiniDMX	Auto	No	0	..	
2	Dmx2	MiniDMX	Auto	No	0	..	

Codice sorgente LD (Ptp117a100)

