IEC 61131-3 Programming (Obsolete)

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 Prefazione

Con l'ambiente di sviluppo LogicLab vengono fornite librerie contenenti oggetti (funzioni e blocchi funzione) specifiche per la soluzione di specifici problemi. L'elenco di tutti gli oggetti disponibili è consultabile on-line nel nostro sito di supporto all'indirizzo:

https://support.elsist.biz/articoli/indice-manuale-programmazione-iec-61131-3/

Gli oggetti nel tempo subiscono correzioni ed aggiornamenti per soluzionare bugs o per aggiungere nuove funzionalità, per questo in alcuni casi vengono aggiunti punti di I/O e/o modificato il tipo degli I/O esistenti. In questo caso viene modificato il nome dell'oggetto tipicamente aggiungendo l'estensione "***_vx***" al nome. Nel manuale on-line viene riportato il nuovo oggetto con la relativa descrizione, ed il vecchio oggetto viene spostato nella libreria ***eLLabObsoleteLib***, e la relativa descrizione spostata in questo manuale. Ulteriori informazioni sugli oggetti obsoleti si trova all'indirizzo:

https://support.elsist.biz/articoli/libreria-ellabobsoletelib-oggetti-obsoleti/

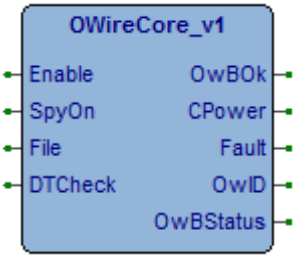
2 Funzioni ed FB gestione One-Wire (eLLab1WireLib)

Attenzione! Per utilizzare la libreria occorre importarla nel proprio progetto.

Questa libreria ha un insieme di oggetti tutti dipendenti tra di loro quindi l'intera libreria và importata nel progetto, quindi gli oggetti non sono spostati nella libreria degli obsoleti, ma si mantiene la disponibilità del download della versione precedente della intera libreria.

2.1 OWireCore_v1, One-Wire interface core

Type	Library
FB	eLLab1WireLib (v5.0.0)



Questo blocco funzione inizializza e gestisce il convertitore One-Wire connesso allo stream definito in ***File***. Il FB controlla il tipo di interfaccia connesso allo stream (Modulo CPU o porta seriale) e la gestisce.

Attivando ***Enable*** il FB esegue la gestione dell'interfaccia One-Wire, l'uscita ***OwBOk*** si attiva se il bus One-Wire funziona correttamente ed è connesso almeno un dispositivo. L'uscita ***CPower*** gestisce l'accensione del convertitore (Nel caso del modello DS9097U deve essere connessa al segnale DTR della seriale). In ***OwBStatus*** viene ritornata la condizione del bus One-Wire.

L'uscita ***Fault*** si attiva in caso di errori di gestione. L'FB ritorna ***OWID*** che deve essere passato alle FB collegate.

Enable (BOOL) Abilitazione blocco funzione, attivandolo viene gestito il convertitore Seriale/One-Wire.

SpyOn (BOOL) Attivato permette lo spionaggio del funzionamento.

File (FILEP) Flusso dati ***stream*** a cui è connessa l'interfaccia One-Wire.

DTCheck (REAL) Tempo di controllo dispositivo gestione bus One-Wire. Ogni tempo definito viene gestito un ciclo di verifica del dispositivo di gestione bus e del bus stesso. Se “0” il controllo è disabilitato.

OwOk (BOOL) Attivo se bus One-Wire funziona correttamente ed è connesso almeno un dispositivo.

CPower (BOOL) Comando alimentazione convertitore

Fault (BOOL) Attivo se errore gestione bus One-Wire.

OwID (@_OWIREDATA) One-Wire management ID, da passare alle FB collegate.

OwBStatus (USINT) Ritorna lo stato in cui si trova il bus One-Wire
0: Bus shorted
1: Bus Ok
2: Presence pulse
3: No devices

Trigger di spy

Se ***SpyOn*** attivo viene eseguita la funzione ***SysSpyData*** che permette di spiare il funzionamento della FB. Sono previsti vari livelli di triggers.

TFlags	Descrizione
16#00000001	***Lg:*** Log funzionamento.
16#40000000	***Er:*** Errori di funzionamento.

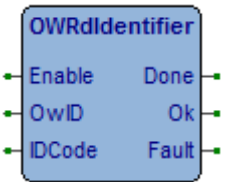
Codici di errore

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

10008010	Valore di ***File*** non definito.
10008100	Timeout esecuzione
10008200	Errore sequenza Strong Pullup.
10008400~6	Errore nelle sequenze configurazione interfaccia One-Wire.
10008500	Errore controllo interfaccia One-Wire.
10008600~2	Errore nelle sequenze indirizzamento device One-Wire.

2.2 OWRdIdentifier, One-Wire read ROM identifier

Type	Library
FB	eLLab1WireLib (v5.0.0)



Questo blocco funzione esegue la lettura del codice di identificazione di un dispositivo One-Wire, si collega al blocco funzione ***OWireCore*** di gestione convertitore One-Wire. Occorre passare ***OwD*** in uscita dal blocco funzione di gestione convertitore.

Attivando ***Enable***, viene eseguita la lettura del ROM ID dal dispositivo connesso al bus One-Wire ***Attenzione! Bisogna avere un solo dispositivo connesso al bus***. Al termine della lettura del codice si attiva l'uscita ***Done***. Se la lettura ha esito positivo si attiva per un loop di programma l'uscita ***Ok***. Gli 8 bytes del ROM ID sono trasferiti nell'array indirizzato da ***IDCode***.

Disattivando ***Enable*** si azzera ***Done*** e l'eventuale ***Fault***, per eseguire nuovamente il comando occorre riabilitare l'ingresso ***Enable***.

Enable (BOOL) Abilita il blocco funzione.

OwID (@_OWIREDATA) One-Wire management ID, fornito dalla FB ***OWireCore***.

IDCode (@USINT) Puntatore array memorizzazione ROM ID, deve essere almeno 8 bytes.

Done (BOOL) Si attiva al termine della esecuzione lettura del ROM ID.

Ok (BOOL) Attivo per un loop se lettura ROM ID eseguita correttamente.

Fault (BOOL) Attivo se errore lettura ROM ID.

Codici di errore

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

10009010 ***OwID*** non definito.

10009020 ***OwID*** non corretto.

10009200 Timeout esecuzione.

Esempi

Ecco un semplice esempio di 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 ***ROMID***.

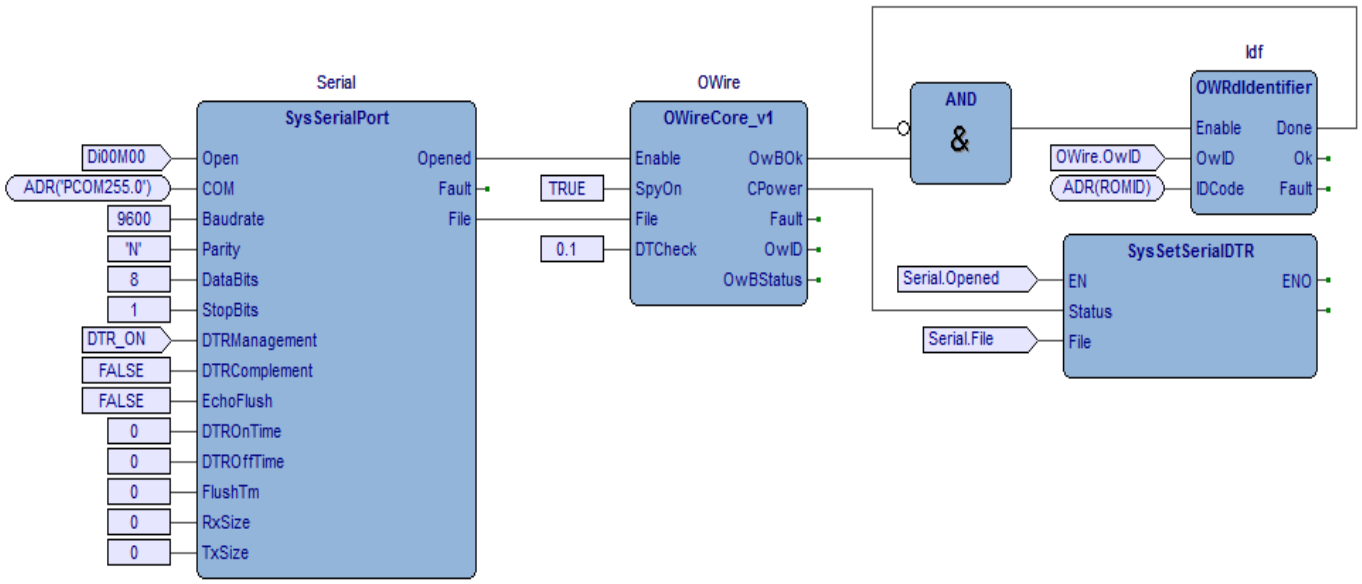
Ciclicamente viene eseguita l'acquisizione, se un TAG è inserito nel lettore viene attivato ***TAGInserted***.

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

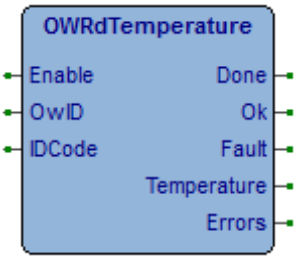
```
VAR
  Serial : SysSerialPort; (* Serial port management *)
  OWire : OWireCore_v1; (* FB One Wire management *)
  Idf : OWRdIdentifier; (* Read identifier *)
  ROMID : ARRAY[ 0..7 ] OF BYTE; (* ROM ID code *)
END_VAR
```

Esempio FBD *(PTP120A600, FBD_OWRdIdentifier)*



2.3 OWRdTemperature, One-Wire read temperature

Type	Library
FB	eLLab1WireLib (v5.0.0)



Questo blocco funzione esegue l'acquisizione di un sensore One-Wire di temperatura (Maxim DS18B20), si collega al blocco funzione ***OWireCore*** di gestione interfaccia One-Wire. Occorre passare ***OwID*** in uscita dal blocco funzione di gestione ***OWireCore***.

Attivando ***Enable***, viene eseguita la lettura del valore di temperatura dal dispositivo connesso al bus One-Wire, terminata l'acquisizione si attiva l'uscita ***Done***, se acquisizione corretta si attiva per un loop l'uscita ***Ok*** e su ***Temperature***, sarà riportato il valore di temperatura acquisito.

L'uscita ***Fault*** si attiva in caso di errori di gestione. Disattivando ***Enable*** si azzera ***Done***, per eseguire una nuova lettura occorre riabilitare l'ingresso ***Enable***.

Se sul bus One-Wire è connesso un unico dispositivo, si può evitare di settare ***IDCode*** oppure si può forzare a ***0***. Se invece sul bus One-Wire sono presenti più dispositivi parallelati, in ***IDCode*** occorre definire l'indirizzo dell'array di 8 bytes che contiene il ROM ID del dispositivo che si vuole acquisire.

Enable (BOOL) Abilita il blocco funzione.

OwID (@_OWIREDATA) One-Wire management ID, fornito in uscita dal blocco funzione ***OWireCore***.

IDCode (@USINT) Puntatore ad array definizione ROM ID dispositivo da acquisire. Se 0 viene eseguita lettura con comando Skip ROM (Viene acquisito qualsiasi device connesso).

Done (BOOL) Si attiva al termine della esecuzione lettura temperatura.

Ok (BOOL) Attivo per un loop se lettura temperatura eseguita correttamente.

Fault (BOOL) Attivo se errore lettura temperatura.

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).

Errors (UDINT) Contatore errori di esecuzione.

Codici di errore

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

10010010 ***OwID*** non definito.

10010020 ***OwID*** non corretto.

10010100 Errore in FB ***OWireCore***.

10010200 Timeout esecuzione

10010100 FB ***OWireCore***, gestione interfaccia One-Wire non disponibile.

10010300~3 Errore nelle sequenze acquisizione temperatura.

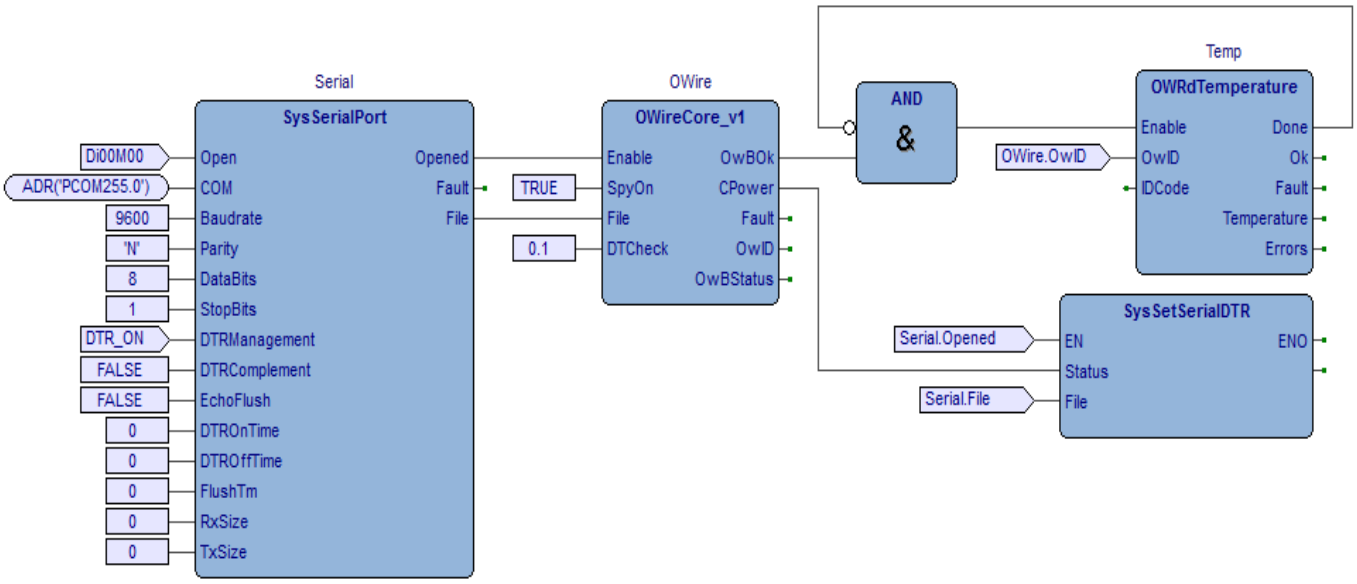
Esempi

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***. Il valore acquisito è trasferito nella variabile ***Temperature***.

Definizione variabili

```
VAR
  Serial : SysSerialPort; (* Serial port management *)
  OWire : OWireCore_v1; (* FB One Wire management *)
  Temp : OWRdTemperature; (* Read temperature *)
END_VAR
```

Esempio LD *(PTP120A400, FBD_OWRdTemperatureSkipROM)*



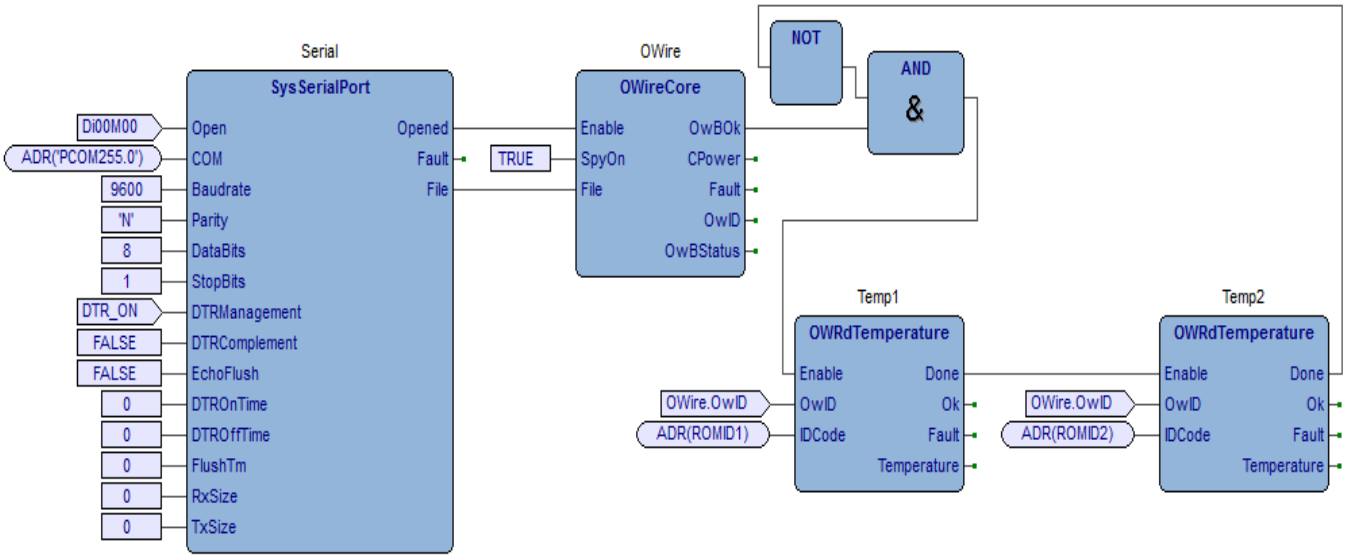
Esempi

Viene eseguita la lettura di due dispositivi One-Wire connessi alla porta presente sul modulo CPU SlimLine.

Definizione variabili

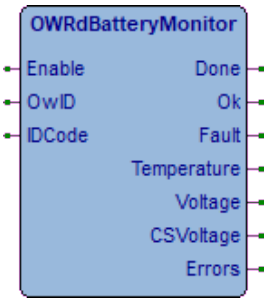
```
VAR
  Serial : SysSerialPort; (* Serial port management *)
  OWire : OWireCore; (* FB One Wire management *)
  Temp1 : OWRdTemperature; (* Read temperature *)
  Temp2 : OWRdTemperature; (* Read temperature *)
  ROMID1 : ARRAY[ 0..7 ] OF BYTE := [16#28, 16#F0, 16#87, 16#E3, 16#08, 16#00, 16#00, 16#12]; (* ROM ID *)
  ROMID2 : ARRAY[ 0..7 ] OF BYTE := [16#28, 16#46, 16#E1, 16#E3, 16#08, 16#00, 16#00, 16#2A]; (* ROM ID *)
END_VAR
```

Esempio LD *(PTP120A400, FBD_OWRdTemperature2Devices)*



2.4 OWRdBatteryMonitor, One-Wire read smart battery monitor

Type	Library
FB	eLLab1WireLib (v5.0.0)



Questo blocco funzione esegue l'acquisizione di un DS2438 Smart Battery Monitor, si collega al blocco funzione *OWireCore* di gestione interfaccia *One-Wire*. Occorre passare **OwlD** in uscita dal blocco funzione di gestione *OWireCore*.

Attivando **Enable**, viene eseguita la lettura del valore di temperatura, tensione e tensione shunt dal dispositivo, terminata l'acquisizione si attiva l'uscita **Done**, se acquisizione corretta si attiva per un loop l'uscita **Ok** e sulle variabili **Temperature**, **Voltage** e **CSVoltage** saranno ritornati i valori acquisiti.

L'uscita **Fault** si attiva in caso di errori di gestione. Disattivando **Enable** si azzera **Done**, per eseguire una nuova lettura occorre riabilitare l'ingresso **Enable**.

Se sul bus *One-Wire* è connesso un unico dispositivo, si può evitare di settare **IDCode** oppure si può forzare a 0. Se invece sul bus *One-Wire* sono presenti più dispositivi parallelati, in **IDCode** occorre definire l'indirizzo dell'array di 8 bytes che contiene il ROM ID del dispositivo che si vuole acquisire.

Descrizione

Enable (*BOOL*) Comando abilitazione.

OwlD (*@_OWIREDATA*)) *One-Wire management ID*, fornito dalla FB *OWireCore*.

IDCode (*@BYTE*) Puntatore ad array definizione ROM ID dispositivo da acquisire. Se 0 viene eseguita lettura con comando *Skip ROM* (Viene acquisito qualsiasi device connesso).

Done (*BOOL*) Si attiva al termine della esecuzione comando.

Ok (*BOOL*) Attivo per un loop se temperatura acquisita correttamente.

Fault (*BOOL*) Attivo per un loop se errore esecuzione.

Temperature (*REAL*) Valore di temperatura acquisito (°C). Range di lettura da -55÷125 (°C). Risoluzione di lettura 0.03125 (°C).

Voltage (*REAL*) Valore di tensione acquisito (V). Range di lettura da 0÷10 (V). Risoluzione di lettura 10 (mV).

CSVoltage (*REAL*) Valore di tensione su shunt esterno (mV). Range di lettura da -1.248÷1.248. Risoluzione di lettura 0.2441(mV).

Errors (*UDINT*) Contatore errori di esecuzione.

Errori

10117010	*OwlD* non definito.
10117020	*OwlD* non corretto.
10117100	Errore in FB *OWireCore*.
10117200	Timeout esecuzione.
10117300~6	Errore nelle sequenze acquisizione.

Esempi

Rimando agli esempi della OWRdTemperature.

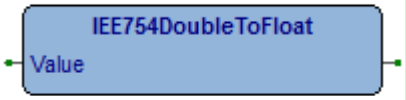
3 Libreria oggetti obsoleti (eLLabObsoleteLib)

Questa libreria contiene tutti gli oggetti che sono diventati obsoleti con il tempo, e sono stati superati da nuove versioni. Il manuale di questi oggetti non è più disponibile in linea sul nostro sito di supporto.

3.1 IEE754DoubleToFloat, IEE754 double to float

Type	Library
Function	eLLabObsoleteLib

Questa funzione esegue la conversione di un numero REAL nel formato IEE754 double (64 bits) nel formato IEE754 float (32 bits).



Parametri funzione:

Value (@BYTE) Pointer all'array valore IEE da convertire.

La funzione ritorna:

(REAL) Valore variabile convertita.

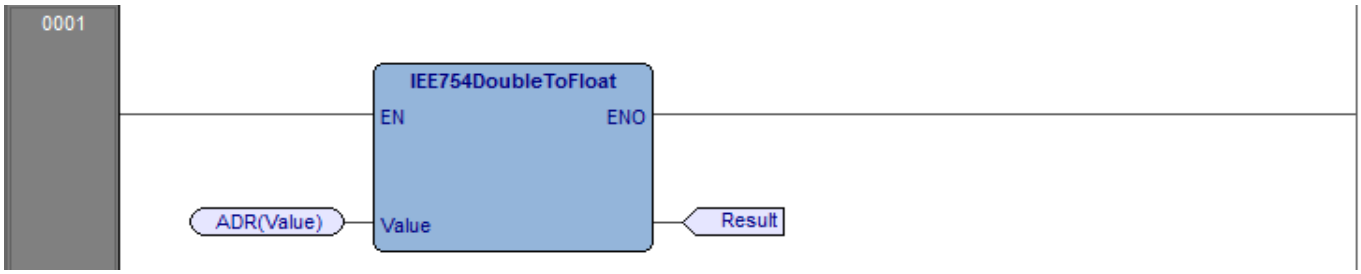
Esempi

Viene eseguita la conversione del valore double IEE754 “3FFB88CE703AFB7F” e si ottiene come risultato il valore REAL “1.7209”.

Definizione variabili

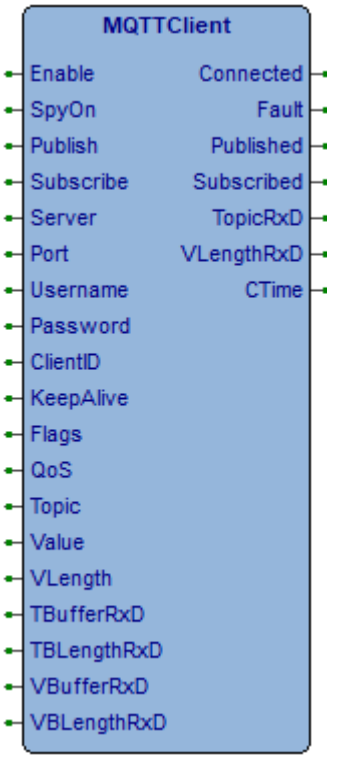
	Name	Type	Address	Array	Init value	Attribute	Description
1	Value	BYTE	Auto	[0..7]	[16#3F, 16#F	..	Double value
2	Result	REAL	Auto	No		..	Result

Esempio LD



3.2 MQTTClient, client for a MQTT server

Type	Library
FB	eLLabObsoleteLib



Questo blocco funzione permette di gestire la connessione ad un server utilizzando il protocollo MQTT (Message Queuing Telemetry Transport) di IBM che l'Organization for the Advancement of Structured Information Standards (OASIS) ha dichiarato come lo standard di riferimento per la comunicazione per l'Internet delle Cose. Il protocollo adotta un meccanismo di pubblicazione e sottoscrizione per scambiare messaggi tramite un appostivo "message broker" che fa da server. Il FB permette sia di pubblicare che di sottoscrivere topics sul broker.

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 su tutti i broker che non richiedono autenticazione (Username e Password non definiti). Sui broker autenticati il FB funziona in modo test per 30 Min.

Con il comando ***Enable*** si forza la connessione al Server (Message broker) sulla porta ***Port***, utilizzando ***Username*** e ***Password*** definite. Se la connessione và a buon fine si attiva l'uscita ***Connected***. A connessione avvenuta ogni 1/3 del tempo definito in ***KeepAlive*** viene inviato un comando di ping per mantenere la connessione. Se il broker non riceve il ping nel tempo definito in ***KeepAlive*** ritiene la connessione terminata.

Il comando di ***Publish*** permette di pubblicare sul broker il ***Topic*** indicato con relative ***Value***, ***Flags*** e ***QoS***. Se la pubblicazione và a buon fine si attiva per un loop l'uscita ***Published***.

Il comando di ***Subscribe*** permette di sottoscrivere sul broker il ***Topic*** indicato con relative ***Flags*** e ***QoS***. Se la sottoscrizione và a buon fine si attiva per un loop l'uscita ***Subscribed***. Ad ogni sottoscrizione del valore di un topic sul broker, il broker ne invia il valore a tutti i client che lo hanno sottoscritto. Il FB riceve sia il nome del topic ritornato in ***TBufferRxD*** che il valore ritornato in ***VbufferRxD*** ed attiva per un loop ***TopicRxD***. In ***VLengthRxD*** è ritornata la lunghezza del valore di topic ricevuto.

In ***CTime*** è ritornato il tempo di comunicazione con broker, rilevato alla connessione ed a ogni esecuzione di ping. O sulla pubblicazione di topic solo se QoS è 1. In caso di errore viene eseguita una disconnessione e riconnessione al broker e si attiva per un loop l'uscita ***Fault***.

Descrizione

Enable (*BOOL*) Comando abilitazione connessione al server *MQTT message broker*.

SpyOn (*BOOL*) Se attivo permette di spiare il funzionamento della FB.

Publish (*BOOL*) Comando pubblicazione di un topic sul server.

Subscribe (*BOOL*) Comando sottoscrizione di un topic sul server.

Server (*@STRING*) Puntatore stringa definizione server MQTT.

Port (*UINT*) Numero porta TCP a cui connettersi.

Username (*@STRING*) Puntatore stringa definizione nome utente (1).

Password (*@STRING*) Puntatore stringa definizione password accesso (1).

User (*@STRING*) Puntatore stringa definizione nome utente.

Password (*@STRING*) Puntatore stringa definizione password.

ClientID (*@STRING*) Puntatore stringa definizione identificativo client (Massimo 23 caratteri) (1).

KeepAlive (*REAL*) Tempo di vita connessione da parte del broker (S).

Flags (*DWORD*) Flags connessione, pubblicazione, sottoscrizione.

QoS (*USINT*) Quality of Service.

Topic (*@STRING*) Puntatore stringa definizione nome topic (Pubblicazione/Sottoscrizione).

Value (*@STRING*) Puntatore valore topic (Pubblicazione).

VLength (*UDINT*) Lunghezza valore topic (Pubblicazione).

TBufferRxD (*@STRING*) Puntatore buffer nome topic ricevuto da broker.

TBLengthRxD (*UDINT*) Dimensione buffer nome topic ricevuto da broker.

VBufferRxD (*@STRING*) Puntatore buffer valore topic ricevuto da broker.

VBLengthRxD (*UDINT*) Dimensione buffer valore topic ricevuto da broker.

Connected (*BOOL*) Si attiva se connessione al server MQTT avvenuta.

Fault (*BOOL*) Attivo per un loop se errore esecuzione.

Published (*BOOL*) Attivo per un loop su pubblicazione di un topic sul server.

Subscribed (*BOOL*) Attivo per un loop su sottoscrizione di un topic sul server.

TopicRxD (*BOOL*) Attivo per un loop su ricezione di un topic dal server.

VLengthRxD (*UINT*) Lunghezza valore topic ricevuto dal broker.

CTime (*Real*) Tempo di comunicazione con il broker (S).

Nota (1): Se nella stringa è presente il carattere $, và indicato come $$ (Esempio Us$er diventa Us$$er).

Esempi

Abilitando da debug ***Enable*** il FB si connette al broker ***broker.mqttdashboard.com***, un broker gratuito e senza autenticazione (Quindi il FB può essere usato gratuitamente), se la connessione và a buon file si attiva l'uscita ***Connected***, ed esegue la pubblicazione dello stato dalla variabile ***Command*** sul topic ***IOCommand***. Il programma si sottoscrive allo stesso topic così ne riceve dal broker lo stato ad ogni variazione, e lo stato ricevuto viene trasferito sulla variabile ***Status***.

Agendo da debug sulla variabile ***Command*** ci ritroveremo lo stesso stato sulla variabile ***Status***, passando attraverso la pubblicazione sul broker ed alla relativa ricezione della notifica in sottoscrizione. In questo articolo informazioni su come interagire con l'esempio da Smartphone.

Nota: non essendoci autenticazione chiunque pubblichi qualcosa sul topic IOCommand và in conflitto con quanto pubblicato dal programma, quindi per evitare conflitti conviene modificare il nome del topic e l'identificativo utente. Ci può solo essere un utente con lo stesso identificativo sullo stesso broker.

```
PROGRAM ST_MQTTClient
VAR
    Enable : BOOL; (* Enable command *)
    Command : BOOL; (* Command status *)
    Status : BOOL; (* Received topic status *)
    CPulse : BOOL; (* Command pulse *)
    CaseNr : USINT; (* Program case *)
    TimeBf : UDINT; (* Time buffer (uS) *)
    Topic : STRING[ 32 ] := 'IOCommand'; (* Topic name *)
    TopicRxD : STRING[ 64 ]; (* Topic received buffer *)
    ValueRxD : STRING[ 300 ]; (* Value received buffer *)
    MQTT : MQTTClient; (* MQTT client FB *)
END_VAR

// ********************************************************************************
// PROGRAM "ST_MQTTClient"
// ********************************************************************************
// This program connects to a free broker, it publishes the state of the BOOL
// variable "Command" on the "IOCommand" topic. It subscribes to the same topic
// so receives the topic status from the broker and transfer it to the BOOL
// variable "Status".
// --------------------------------------------------------------------------------

    // ----------------------------------------------------------------------------
    // INITIALIZATION
    // ----------------------------------------------------------------------------
    // Program initializations.
```

```
    IF (SysFirstLoop) THEN
        MQTT.SpyOn:=TRUE; //Spy active
        MQTT.Server:=ADR('broker.mqttdashboard.com'); //Broker URL
        MQTT.Port:=1883; //Broker TCP port
        MQTT.Username:=NULL; //Broker username
        MQTT.Password:=NULL; //Broker password
        MQTT.ClientID:=ADR('Elsist'); //Client identifier
        MQTT.KeepAlive:=180; //Keep alive time (S)
        MQTT.Flags:=16#00000000; //Connection/Publish/Subscribe flags
        MQTT.QoS:=1; //Quality of Service
        MQTT.TBufferRxD:=ADR(TopicRxD); //Topic buffer (Received)
        MQTT.TBLengthRxD:=SIZEOF(TopicRxD); //Topic buffer length (Received)
        MQTT.VBufferRxD:=ADR(ValueRxD); //Value buffer (Received)
        MQTT.VBLengthRxD:=SIZEOF(ValueRxD); //Value buffer length (Received)
    END_IF;

    // ------------------------------------------------------------------------
    // MQTT CLIENT MANAGEMENT
    // ------------------------------------------------------------------------
        // MQTTClient management.

    MQTT(); //MQTTClient management
    IF (MQTT.Fault) THEN MQTT.Enable:=FALSE; CaseNr:=0; RETURN; END_IF;
    IF NOT(MQTT.Connected) THEN CaseNr:=0; END_IF;

    // ------------------------------------------------------------------------
    // TOPIC VALUE RECEPTION
    // ------------------------------------------------------------------------
    // Check if a topic value has been received.

    IF (MQTT.TopicRxD) THEN

        // Check if the received topic is the expected.

        IF (SysStrFind(ADR(TopicRxD),ADR(Topic), FIND_DEFAULT) <> NULL) THEN

            // The received topic is the expected, so check its value.

            IF (SysStrFind(ADR(ValueRxD), ADR('Off'), FIND_DEFAULT) <> NULL) THEN
Status:=FALSE; END_IF;
            IF (SysStrFind(ADR(ValueRxD), ADR('On'), FIND_DEFAULT) <> NULL) THEN
Status:=TRUE; END_IF;
        END_IF;
    END_IF;

    // ------------------------------------------------------------------------
    // PROGRAM CASES
    // ------------------------------------------------------------------------
    // Program cases.

    CASE (CaseNr) OF

        // ------------------------------------------------------------------------
        // CONNECTION TO BROKER
        // ------------------------------------------------------------------------
        // Check if connected to broker.

        0:
        MQTT.Enable:=Enable; //Connection enable
        IF NOT(MQTT.Connected) THEN RETURN; END_IF;
```

```
        // Client connected to broker, manage the subscription

        MQTT.Topic:=ADR(Topic); //Topic name
        MQTT.Subscribe:=TRUE; //Topic subscribe
        CaseNr:=CaseNr+1; //Program case

        // ------------------------------------------------------------------
        // Waits for subscription.

        1:
        MQTT.Subscribe:=FALSE; //Topic subscribe
        IF NOT(MQTT.Subscribed) THEN RETURN; END_IF;
        CaseNr:=10; //Program case

        // ------------------------------------------------------------------
        // TOPIC PUBLISH
        // ------------------------------------------------------------------
        // Checks if the command status change and publish it.

        10:
        IF (Command = CPulse) THEN RETURN; END_IF;
        CPulse:=Command; //Command pulse

        // Defines the topic to publish.

        MQTT.Topic:=ADR(Topic); //Topic name

        // Defines the value to publish.

        IF NOT(Command) THEN MQTT.Value:=ADR('Off'); END_IF;
        IF (Command) THEN MQTT.Value:=ADR('On'); END_IF;

        // Defines length of the value.

        MQTT.VLength:=Sysstrlen(MQTT.Value); //Value length
        MQTT.Publish:=TRUE; //Topic publish
        CaseNr:=CaseNr+1; //Program case

        // ------------------------------------------------------------------
        // Waits for the publishing.

        11:
        MQTT.Publish:=FALSE; //Topic publish
        IF NOT(MQTT.Published) THEN RETURN; END_IF;
        CaseNr:=10; //Program case
    END_CASE;

// [End of file]
```

4 HTTPClient, HTTP client

Type	Library
FB	eLLabObsoleteLib

L'HTTP è un protocollo che lavora con un'architettura di tipo client/server, il client esegue una richiesta ed il server restituisce la risposta mandata da un altro host. Questo blocco funzione esegue la richiesta in modalità client (Proprio come il browser), per gestire la connessione con il protocollo HTTP utilizzare il FB SysTCPClient collegando tra di loro gli I/O File. Per gestire la connessione con il protocollo HTTPS occorre inserire tra la connessione degli I/O File il FB SysTLSClient.

Attivando Enable viene attivata l'uscita Connect che comanda la connessione al server, a connessione avvenuta viene inviata la richiesta HTTP della pagina definita in Page all'HostName. La pagina viene richiesta con i parametri definiti nel buffer Request passati secondo la definizione di RMethod (GET, POST, PUT).

In DBSize occorre definire la dimensione del buffers che l'FB alloca (Con SysRMalloc) per la gestione dei pacchetti TCP in trasmissione e ricezione. La dimensione minima è 256 bytes, aumentando la dimensione si velocizza il trasferimento (Sono effettuati meno frazionamenti). E' inutile definire lunghezze superiori al limite del pacchetto TCP 1500 bytes.

L'header e la pagina sono ricevuti in frammenti successivi in base al tipo di trasferimento del server, ad ogni ricezione di un frammento DBChars viene valorizzato (Per un solo loop di esecuzione) con il numero di bytes ricevuti che possono essere letti dal buffer all'indirizzo DBAddress. (HPSelector indica il tipo di dati ricevuti, FALSE dati header, TRUE dati pagina). In questo modo è possibile ricevere qualsiasi dimensione di dati, sarà cura del programma utente trasferire i dati ricevuti in una stringa o in un file.

Al termine dell'invio della richiesta si attiva per un loop di programma l'uscita RSent, su ricezione pagina si attiva per un loop di programma l'uscita PLoad, in HLength e PLength sono ritornate le lunghezze dell'header e della pagina ricevuti, mentre in PLTime il tempo necessario per l'intera richiesta.

In caso di errore esecuzione o tempo di esecuzione comando superiore al tempo definito in Timeout, viene attivata per un loop di programma l'uscita Fault. L'uscita Done si attiva al termine della esecuzione della richiesta e su errore. Per acquisire nuovamente la pagina occorre disabilitare e poi riabilitare l'ingresso Enable.

Descrizione

Enable (BOOL) Comando richiesta pagina. Per eseguire una nuova richiesta occorre disattivare e riattivare l'ingresso.

SpyOn (BOOL) Se attivo permette di spiare il funzionamento del FB (Vedi articolo).

File (eFILE) Stream di I/O utilizzato per la connessione.

RMethod (USINT) Metodo gestione richiesta (0=GET, 1=POST, 2=PUT).

HostName (@STRING) Nome del server utilizzato nella richiesta.

HostPort (UINT) Numero porta TCP a cui connettersi (Default 80).

Page (@STRING) Stringa di definizione pagina richiesta.

Request (@STRING) Indirizzo buffer dati da inviare con la richiesta.

Header (@STRING) Indirizzo stringa header, se NULL viene inviato l'header standard. Se definito occorre indicare l'header completo da inviare con la richiesta al server.

DBSize (UINT) Dimensione buffers Rx/Tx allocati da FB (SysRMalloc). Minimo 256 massimo 1500.

Timeout (REAL) Timeout esecuzione richiesta pagina (S).

Connect (BOOL) Comando di connessione al server, su attivazione occorre gestire la connessione.

Done (BOOL) Attivo a fine esecuzione, si attiva anche in caso di Fault. L'uscita rimane attiva fino a quando non viene settato Enable:=FALSE.

Fault (BOOL) Attivo per un loop di programma se errore gestione.

RSent (BOOL) Attivo per un loop di programma al termine dell'invio richiesta HTTP.

HPSelector (BOOL) FALSE: Si stanno ricevendo dati di header. TRUE: Si stanno ricevendo dati di pagina.

PLoad (BOOL) Attivo per un loop di programma su fine ricezione dati. Per acquisire dati (Header/Pagina) riferirsi a

HPSelector. La lunghezza dei dati ricevuti è indicata in DBChars.

DBAddress (@STRING) Indirizzo buffer dati ricevuti allocato da FB (Con SysRMalloc) di dimensione DBSize.

DBChars (UDINT) Bytes di pagina ricevuti, viene valorizzato per un loop. Ad ogni valorizzazione occorre estrarre i dati (Header/Pagina) da DBAddress.

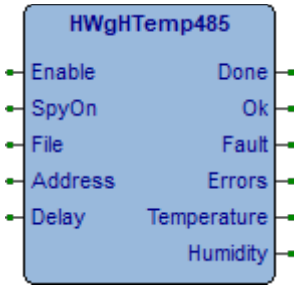
HLength (UDINT) Dimensione header ricevuto.

PLength (UDINT) Dimensione pagina ricevuta.

PLTime (REAL) Tempo impiegato per richiesta pagina (S).

5 HWgHTemp485, HWgroup HTemp-485 acquisition

Type	Library
FB	eLLabObsoleteLib



Questo blocco funzione gestisce la lettura dei sensori di temperatura ed umidità HTemp-485 della HW group con il protocollo seriale RS485. Occorre passare alla FB in ***File*** il puntatore allo stream di comunicazione con il dispositivo.

Attivando l'ingresso ***Enable*** viene effettuata la lettura dei valori di temperatura ed umidità dal dispositivo indirizzato in ***Address***, terminata l'esecuzione si attiva l'uscita ***Done***. Se il comando ha avuto esito positivo si attiva l'uscita ***Ok***, in caso contrario si attiva l'uscita ***Fault***. Per eseguire nuovamente il comando occorre disabilitare e riabilitare l'ingresso ***Enable***. L'ingresso ***SpyOn*** se attivo permette di spiare il funzionamento della FB.

Il blocco funzione è realizzato per permetterne la connessione in cascata, è possibile connettere al ***Done*** di un FB l'***Enable*** di un'altro e così di seguito condividendo lo stesso stream di comunicazione.

Enable (*BOOL*) Comando abilitazione acquisizione sensore. Per rieseguire il comando disabilitare e poi riabilitare questo ingresso.

SpyOn (*BOOL*) Se attivo permette di spiare il funzionamento della FB (Vedi articolo).

File (*eFILEP*) Flusso dati stream da utilizzare per la comunicazione.

Address (*STRING[1]*) Indirizzo sensore, è posibile acquisire in multidrop fino a 25 sensori con indirizzo da A a Z escluso T.

Delay (*REAL*) Tempo di pausa dopo l'esecuzione del comando espresso in S.

Done (*BOOL*) Si attiva al termine della esecuzione comando e rimane attiva fino alla disabilitazione di *Enable*.

Ok (*BOOL*) Attivo per un loop se esecuzione comando corretta.

Fault (*BOOL*) Attivo per un loop se errore esecuzione comando.

Errors (*UDINT*) Numero di errori, incrementato ad ogni nuovo errore, raggiunto valore massimo riparte da 0.

Temperature (*REAL*) Valore di temperatura (°C).

Humidity (*REAL*) Valore umidità (%).

5.1 Trigger di spy

Se *SpyOn* attivo è possibile utilizzare la console di spionaggio per verificare il funzionamento della FB. Sono previsti vari livelli di triggers.

16#00000001 ***Tx:*** Invio frame verso sensore.
16#00000002 ***Rx:*** Ricezione frame da sensore.
16#40000000 ***Er:*** Messaggio di errore.

5.2 Esempi

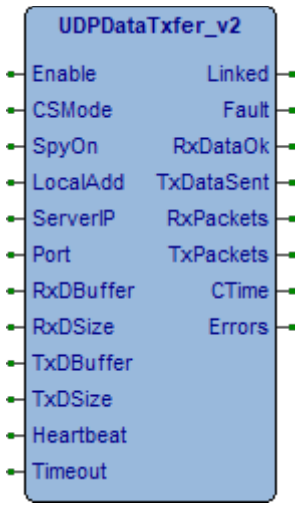
Nell'esempio è gestita l'acquisizione di un sensore HTem-485 con indirizzo "E" (Indirizzo di default), connesso alla porta seriale RS485.

```
PROGRAM ST_HWgHTemp485
VAR
    Sp : SysSerialPort; (* Serial port *)
    Sensor : HWgHTemp485; (* HTemp sensor *)
END_VAR

// ********************************************************************************
// PROGRAM "ST_HWgHTemp485"
// ********************************************************************************
// A HWgroup HTemp-485 sensor connected to the COM2 serial port is acquired.
// --------------------------------------------------------------------------------

    // ----------------------------------------------------------------------------
    // INITIALIZATION
    // ----------------------------------------------------------------------------
```

```
    // Program initializations.

    IF (SysFirstLoop) THEN

        // Serial port settings.

        Sp.COM:=ADR('COM2'); //COM port definition
        Sp.Baudrate:=9600; //Baudrate
        Sp.Parity:='N'; //Parity
        Sp.DataBits:=8; //Data bits
        Sp.StopBits:=1; //Stop bits
        Sp.DTRManagement:=DTR_AUTO_WO_TIMES; //DTR management
        Sp.DTRComplement:=FALSE; //DTR complement
        Sp.EchoFlush:=FALSE; //Received echo flush
        Sp.DTROnTime:=0; //DTR On time delay (mS)
        Sp.DTROffTime:=0; //DTR Off time delay (mS)
        Sp.FlushTm:=0; //Flush time (mS)
        Sp.RxSize:=0; //Rx buffer size
        Sp.TxSize:=0; //Tx buffer size

        // HTemp sensor settings.

        Sensor.SpyOn:=TRUE; //Spy On
        Sensor.Address:='E'; //Sensor address
        Sensor.Delay:=1.0; //Delay time (S)
    END_IF;

    // ------------------------------------------------------------------------
    // SENSOR ACQUISITION
    // ------------------------------------------------------------------------
    // Sensor acquisition.

    Sp(Open:=TRUE); //Serial port management
    Sensor.File:=Sp.File; //File pointer
    Sensor(); //HTemp sensor
    Sensor.Enable:=Sp.Opened AND NOT(Sensor.Done); //Acquisition enable

// [End of file]
```

6 UDPDataTxfer, data transfer by UDP connection

Type	Library
FB	eLLabObsoleteLib



Questo blocco funzione esegue il trasferimento di un blocco di memoria tra due sistemi utilizzando una connessione UDP su rete ethernet. Su un sistema andrà istanziato il FB configurato in modalità client (***CSMode***:=FALSE), mentre sull'altro sistema andrà istanziato il FB configurato in modalità server (***CSMode***:=TRUE). I parametri ServerIP e Port da impostare solo sul sistema client indicano l'indirizzo IP e la porta del sistema server con cui avviene il trasferimento dati.

L'uscita ***Linked*** si attiva quando si è verificata la corretta connessione tra i due sistemi, e per garantire che i due sistemi siano sempre correttamente interconnessi anche se non vi è variazione nei dati da trasmettere viene comunque inviato un pacchetto dati ogni tempo definito in ***Heartbeat***.

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 al minimo 2 loop di esecuzione programma. Se dopo l'invio non viene ricevuto acknowledge entro il tempo definito viene generato errore.

L'invio dei dati è automatico sulla variazione di uno qualsiasi dei bytes del buffer di trasmissione, ***RxDataOk*** si attiva per un loop ad ogni ricezione dati dall'altro sistema, ***TxDataSent*** si attiva per un loop al termine della trasmissione dei dati verso l'altro sistema.

Enable (BOOL) Comando abilitazione.

CSMode (BOOL) Client/Server mode. FALSE:Modalità UDP client, TRUE:Modalità UDP server.

SpyOn (BOOL) Se attivo permette di spiare il funzionamento della FB (Vedi articolo).

LocalAdd (@STRING) Range indirizzi IP da cui è accettata la connessione. La connessione è accettata se indirizzo IP del peer in AND con il valore non viene modificato. Default '0.0.0.0': connessione accettata da tutti gli indirizzi IP.

ServerIP (@STRING) Indirizzo IP (o URL) del sistema server con cui scambiare dati. Occorre definirlo solo sul sistema configurato come client (CSMode:=FALSE).

Port (UINT) Numero porta UDP del sistema server con cui scambiare dati. Occorre definirlo solo sul sistema configurato come client (CSMode:=FALSE).

RxDBuffer (@BYTE) Indirizzo buffer dove devono essere trasferiti i dati ricevuti.

RxDSize (UDINT) Dimensione buffer dati ricevuti. Deve coincidere con il valore di TxDSize dell'altro sistema.

TxDBuffer (@BYTE) Indirizzo buffer dati da inviare. Ad ogni variazione del contenuto del buffer l'intero buffer è inviato all'altro sistema.

TxDSize (UDINT) Dimensione buffer dati da inviare. Deve coincidere con il valore di RxDSize dell'altro sistema.

Heartbeat (REAL) Tempo massimo in cui deve esserci uno scambio dati tra i due sistemi espresso in secondi. Se non vi è variazione nei dati da trasmettere ogni tempo definito il sistema client invia un pacchetto dati. Deve essere definito lo stesso valore in entrambi i sistemi.

Timeout (REAL) Tempo massimo ricezione acknowledge su invio dati espresso in secondi. Se a seguito di un invio dati nel tempo definito non viene ricevuto l'acknowledge si genera errore. Deve essere definito lo stesso valore in entrambi i sistemi.

Linked (BOOL) Si attiva se i due sistemi client e server sono correttamente interconnessi.

Fault (BOOL) Attivo per un loop se errore esecuzione.

RxDataOk (BOOL) Attivo per un loop alla ricezione dati.

TxDataSent (BOOL) Attivo per un loop su trasmissione dati.

RxPackets (UDINT) Contatore pacchetti ricevuti.

TxPackets (UDINT) Contatore pacchetti trasmessi.

CTime (REAL) Tempo comunicazione (S).

Errors (UDINT) Numero di errori.

6.1 Trigger di spy

Se *SpyOn* attivo è possibile utilizzare la console di spionaggio per verificare il funzionamento della FB. Sono previsti vari livelli di triggers.

16#00000001 **Tx:** Invio frame verso sensore.

16#00000002 **Rx:** Ricezione frame da sensore.

16#80000000 **Er:** Messaggio di errore.

6.2 Esempi

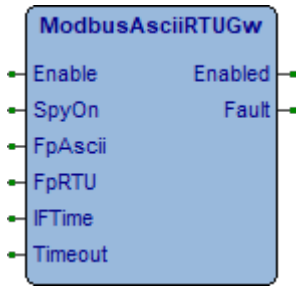
Nell'esempio è gestita l'acquisizione di un sensore HTem-485 con indirizzo "E" (Indirizzo di default), connesso alla porta seriale RS485.

```
PROGRAM ST_UDPDataTxfer_v2
VAR
 CRxDBuffer : ARRAY[0..7] OF BYTE; (* Rx data buffer (Client) *)
    CTxDBuffer : ARRAY[0..3] OF BYTE; (* Tx data buffer  (Client) *)
    SRxDBuffer : ARRAY[0..3] OF BYTE; (* Rx data buffer (Server) *)
    STxDBuffer : ARRAY[0..7] OF BYTE; (* Tx data buffer (Server) *)
    CDataTxF : UDPDataTxfer_v2; (* UDP data transfer (Client) *)
    SDataTxF : UDPDataTxfer_v2; (* UDP data transfer (Server) *)
END_VAR

// ********************************************************************************
// PROGRAM "ST_UDPDataTxfer_v1"
// ********************************************************************************
// Exchanges data using a localhost UDP connection.
// --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------
    // CLIENT MODE OPERATION
    // --------------------------------------------------------------------------
    // UDP data transfer initialization (Client).

    IF (SysFirstLoop) THEN
        CDataTxF.Enable:=TRUE; //Enable
        CDataTxF.SpyOn:=TRUE; //Spy active
        CDataTxF.CSMode:=FALSE; //FALSE:Client, TRUE:Server
        CDataTxF.LocalAdd:=ADR('0.0.0.0'); (* Local address *)
        CDataTxF.ServerIP:=ADR('127.0.0.1'); //Server address
        CDataTxF.Port:=10000; //Client/Server port number

        CDataTxF.RxDBuffer:=ADR(CRxDBuffer); //Rx data buffer
        CDataTxF.RxDSize:=SIZEOF(CRxDBuffer); //Rx data size

        CDataTxF.TxDBuffer:=ADR(CTxDBuffer); //Tx data buffer
        CDataTxF.TxDSize:=SIZEOF(CTxDBuffer); //Tx data size

        CDataTxF.Heartbeat:=2.0; //Heartbeat time (S)
        CDataTxF.Timeout:=3.0; //Timeout time (S)
    END_IF;

    // Execute the data transfer.

    CDataTxF(); //UDP data transfer (Client)

    // --------------------------------------------------------------------------
    // SERVER MODE OPERATION
    // --------------------------------------------------------------------------
    // UDP data transfer initialization (Server).

    IF (SysFirstLoop) THEN
```

```
        SDataTxF.Enable:=TRUE; //Enable
        SDataTxF.SpyOn:=TRUE; //Spy active
        SDataTxF.CSMode:=TRUE; //FALSE:Client, TRUE:Server
        SDataTxF.LocalAdd:=ADR('0.0.0.0'); (* Local address *)
        SDataTxF.ServerIP:=eNULL; //Server address
        SDataTxF.Port:=10000; //Client/Server port number

        SDataTxF.RxDBuffer:=ADR(SRxDBuffer); //Rx data buffer
        SDataTxF.RxDSize:=SIZEOF(SRxDBuffer); //Rx data size

        SDataTxF.TxDBuffer:=ADR(STxDBuffer); //Tx data buffer
        SDataTxF.TxDSize:=SIZEOF(STxDBuffer); //Tx data size

        SDataTxF.Heartbeat:=2.0; //Heartbeat time (S)
        SDataTxF.Timeout:=3.0; //Timeout time (S)
    END_IF;

    // Execute the data transfer.

     SDataTxF(); //UDP data transfer (Server)

// [End of file]
```

7 ModbusAsciiRTUGw, modbus ascii from/to RTU gateway

Type	Library
FB	eLLabObsoleteLib



Questo blocco funzione esegue la conversione di protocollo tra il Modbus Ascii ed il Modbus RTU. Il protocollo Ascii ricevuto dallo stream ***FpAscii*** viene convertito in RTU e trasmesso sullo stream ***FpRTU*** e viceversa. E' quindi possibile fare dialogare tra di loro sistemi che utilizzano i due tipi di protocollo diversi.

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.

I frames modbus ricevuti vengono controllati verificando il LRC per i frame Ascii ed il CRC per i frames RTU. In caso di errore comando viene attivata per un loop l'uscita ***Fault***.

Enable (*BOOL*) Comando di abilitazione blocco funzione.

SpyOn (*BOOL*) Se attivo permette di spiare il funzionamento della FB (Vedi articolo).

FpAscii (e*FILEP*) Flusso dati stream da per la gestione del protocollo Modbus Ascii.

FpRTU (e*FILEP*) Flusso dati stream da per la gestione del protocollo Modbus RTU.

IFTime (*UDINT*) Tempo ricezione caratteri (µS) riferito al protocollo RTU, se comunicazione su porta seriale il tempo deve essere definito in base al baud rate (Vedi tabella). Nel caso di comunicazione su rete ethernet è possibile definire il valore minimo.

Timeout (*DSINT*) Tempo massimo gestione pacchetto (mS).

Enabled (*BOOL*) *Blocco funzione abilitato*.

Fault (*BOOL*) Attivo per un loop se errore esecuzione comando.

7.1 Trigger di spy

Se *SpyOn* attivo è possibile utilizzare la console di spionaggio per verificare il funzionamento della FB. Sono previsti vari livelli di triggers.

16#00000001 ***Ax:*** Frame Modbus Ascii ricevuto.

16#00000002 ***Rx:*** Frame Modbus RTU ricevuto.

16#40000000 ***Er:*** Errore di esecuzione.

7.2 Esempi

L'esempio può essere eseguito su di un sistema SlimLine. Tramite una connessione TCP in localhost un FB *ModbusMaster_v1* comunica in Ascii con il FB *ModbusAsciiRTUGw* che converte in RTU e lo invia sulla porta seriale COM0. Collegando con un cavo cross le seriali COM0 con la COM1 del sistema lo slave Modbus integrato risponderà alla richiesta in RTU. Il mesaggio di risposta e verrà ritornato al ModbusMaster convertito in ascii.

```
PROGRAM ST_ModbusAsciiRTUGw
VAR
    i : UDINT;     (* Aux counter *)
    CaseNr : USINT; (* Program case *)
    Errors : UDINT;  (* Error counter *)
    WHRegs : ARRAY[0..7] OF WORD; (* Holding registers (Write) *)
    RHRegs : ARRAY[0..7] OF WORD; (* Holding registers (Read) *)
    Sp : SysSerialPort; (* Serial port *)
    Mdb : ModbusMaster_v1; (* Modbus master FB *)
    TCPClient : SysTCPClient; (* TCP client management *)
    MdbGw : ModbusAsciiRTUGw; (* Modbus ascii to RTU gateway *)
    TCPServer : SysTCPServer; (* TCPServer management *)
END_VAR

// ********************************************************************************
// PROGRAM "ST_ModbusAsciiRTUGw"
// ********************************************************************************
```

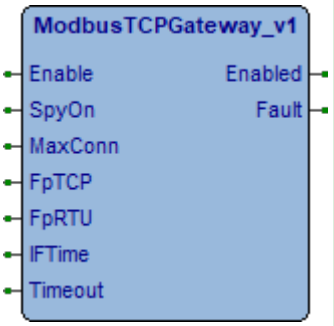
```
// A ModbusMaster communicate in modbus ascii  with the ModbusAsciiRTUGw FB
// using a localhost TCP connection. The FB converts the protocol to modbus RTU
// and communicate to the embedded modbus slave by using a serial connection.
// ------------------------------------------------------------------------------

    // ------------------------------------------------------------------------
    // INITIALIZATION
    // ------------------------------------------------------------------------
    // Program initializations.

    IF (SysFirstLoop) THEN

        // Serial port settings.

        Sp.COM:=ADR('COM0'); //COM port definition
        Sp.Baudrate:=115200; //Baudrate
        Sp.Parity:='E'; //Parity
        Sp.DataBits:=8; //Data bits
        Sp.StopBits:=1; //Stop bits
        Sp.DTRManagement:=DTR_AUTO_WO_TIMES; //DTR management
        Sp.DTRComplement:=FALSE; //DTR complement
        Sp.EchoFlush:=FALSE; //Received echo flush
        Sp.DTROnTime:=0; //DTR On time delay (mS)
        Sp.DTROffTime:=0; //DTR Off time delay (mS)
        Sp.FlushTm:=0; //Flush time (mS)
        Sp.RxSize:=0; //Rx buffer size
        Sp.TxSize:=0; //Tx buffer size

        // TCP server settings.

        TCPServer.FilesArr:=ADR(MdbGw.FpAscii); //Files array
        TCPServer.LocalAdd:=ADR('0.0.0.0'); //Local address
        TCPServer.LocalPort:=3100; //Local port
        TCPServer.MaxConn:=1; //Accepted connections
        TCPServer.FlushTm:=50; //Flush time (mS)
        TCPServer.LifeTm:=60; //Life time (S)
        TCPServer.RxSize:=256; //Rx buffer size
        TCPServer.TxSize:=256; //Tx buffer size

        // TCP client settings.

        TCPClient.PeerAdd:=ADR('127.0.0.1'); //Peer address
        TCPClient.PeerPort:=3100; //Peer port
        TCPClient.LocalAdd:=ADR('0.0.0.0'); //Local address
        TCPClient.LocalPort:=0; //Local port
        TCPClient.FlushTm:=50; //Flush time (mS)
        TCPClient.LifeTm:=20; //Life time (S)
        TCPClient.RxSize:=128; //Rx buffer size
        TCPClient.TxSize:=128; //Tx buffer size

        // Modbus master settings.

        Mdb.SpyOn:=TRUE; //Spy On
        Mdb.Type:=1; //Modbus type (Ascii)
        Mdb.Node:=1; //Device modbus node
        Mdb.Timeout:=0.5; //Timeout time (S)
        Mdb.Delay:=1.0; //Delay time (S)

        // Modbus master settings.

        MdbGw.SpyOn:=TRUE; //Spy On
        MdbGw.IFTime:=286; //Device modbus node
```

```
    MdbGw.Timeout:=100; //Timeout time (mS)
END_IF;

// ------------------------------------------------------------------------
// MODBUS MANAGEMENT
// ------------------------------------------------------------------------
// Serial port management.

Sp(Open:=TRUE); //Serial port management
MdbGw.FpRTU:=Sp.File; //Modbus RTU file pointer

// TCP server management.

TCPServer(Enable:=TRUE); //TCPServer management

// TCP client management.

TCPClient(Connect:=TRUE); //TCPClient management
Mdb.File:=TCPClient.File; //File pointer

// Manage the modbus master communication.

Mdb(); //Modbus master
Mdb.Enable:=NOT(Mdb.Done); //Modbus enable

// Manage the modbus gateway.

MdbGw(Enable:=SysFIsOpen(Mdb.File)); //Modbus ascii to RTU gateway
IF NOT(SysFIsOpen(Mdb.File)) THEN Mdb.Enable:=FALSE; CaseNr:=0; RETURN; END_IF;

// ------------------------------------------------------------------------
// PROGRAM CASES
// ------------------------------------------------------------------------
// Manage the program cases.

CASE (CaseNr) OF

    // --------------------------------------------------------------------
    // Execute a Preset multiple registers.

    0:
    Mdb.FCode:=16#10; //Modbus function code
    Mdb.Address:=40000+(16/2); //Modbus register address
    Mdb.Points:=SIZEOF(WHRegs)/2; //Modbus register points
    Mdb.Buffer:=ADR(WHRegs); //Memory buffer address
    IF NOT(Mdb.Done) THEN RETURN; END_IF;
    CaseNr:=CaseNr+1; //Program case

    // --------------------------------------------------------------------
    // Execute a Read holding registers.

    1:
    Mdb.FCode:=16#03; //Modbus function code
    Mdb.Address:=40000+(16/2); //Modbus register address
    Mdb.Points:=SIZEOF(RHRegs)/2; //Modbus register points
    Mdb.Buffer:=ADR(RHRegs); //Memory buffer address
    IF NOT(Mdb.Done) THEN RETURN; END_IF;
    CaseNr:=0; //Program case

    // Checks if read values are correct and change them.

    FOR i:=0 TO ((SIZEOF(WHRegs)/SIZEOF(WHRegs[0]))-1) DO
```

```
            IF (RHRegs[i] <> WHRegs[i]) THEN Errors:=Errors+1; END_IF;
            WHRegs[i]:=TO_WORD(SysGetRandom(TRUE)*65535.0); //Mapped holding register
        END_FOR;
    END_CASE;

// [End of file]
```

8 ModbusTCPGateway, modbus TCP gateway

Type	Library
FB	eLLabObsoleteLib



Questo blocco funzione opera come gateway Modbus tra una connessione modbus TCP ed una connessione seriale Modbus RTU. Il FB può gestire più streams Modbus TCP, per questo in ***FpTCP*** occorre passare l'array di streams di I/O da cui arrivano le richieste Modbus TCP. In ***FpRTU*** occorre passare lo stream verso cui sono inviate le richieste convertite in Modbus RTU. In ***MaxConn*** viene definito il numero di connessioni massime gestite sullo stream Modbus TCP.

In ***IFTime*** occorre definire il tempo di pausa su ricezione caratteri da porta seriale (Modbus RTU). In ***Timeout*** si definisce il tempo massimo di esecuzione di un comando Modbus TCP (Da quando il comando viene ricevuto a quando dopo la conversione in RTU viene ritornata la risposta).

L'ingresso ***SpyOn*** se attivo permette di spiare il funzionamento della FB. In caso di errore esecuzione o tempo di esecuzione superiore al tempo definito in *Timeout*, viene attivata per un loop di programma l'uscita ***Fault***.

Il FB può gestire più connessioni TCP contemporanee, quando da una connessione ha ricevuto un comando Modbus completo lo inoltra sullo stream di gestione Modbus RTU, attende la risposta e la reinvia sulla connessione TCP da cui ha ricevuto la richiesta.

Enable (*BOOL*) Abilitazione blocco funzione.

SpyOn (*BOOL*) Se attivo permette di spiare il funzionamento della FB (Vedi articolo).

MaxConn (*USINT*) Numero massimo di connessioni contemporanee accettate dal server. Deve essere uguale al numero di files definiti.

FpTCP (*@FILEP*) Pointer ad array streams di I/O connessioni Modbus TCP. Occorre definire un numero di streams pari al numero di connessioni contemporanee accettate.

FpRTU (*@FILEP*) Stream di I/O connessione Modbus RTU.

IFTime (*UDINT*) Tempo ricezione caratteri (µS) deve essere definito in base al baud rate (Vedi tabella sotto).

Timeout (*UINT*) Tempo massimo esecuzione comando espresso in mS. Se il comando non termina nel tempo definito viene abortito ed attivata l'uscita *Fault*.

Enabled (*BOOL*) Blocco funzione abilitato.

Fault (*BOOL*) Attivo per un loop se errore esecuzione.

8.1 IFTime, tempo ricezione caratteri

Baud rate	Tempo	Baud rate	Tempo	Baud rate	Tempo
300	112000	4800	7000	57600	573
600	56000	9600	3430	76800	429
1200	28000	19200	1720	115200	286
2400	14000	38400	860		

8.2 Trigger di spy

Se *SpyOn* attivo è possibile utilizzare la console di spionaggio per verificare il funzionamento della FB. Sono previsti vari livelli di triggers.

16#00000001 ***Tx:*** Invio frame comando modbus RTU.

16#00000002 ***Rx:*** Ricezione frame risposta modbus RTU.

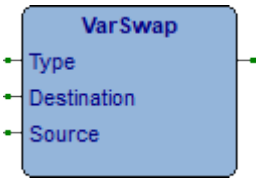
8.3 Esempi

Semplice gateway Modbus TCP/RTU, accetta fino a 3 connessioni TCP su porta 2000 reindirazzondole come Modbus RTU sulla porta seriale COM2.

```
PROGRAM ST_ModbusTCPGateway_v1
VAR
    Fp : ARRAY[0..2] OF FILEP; (* File pointer *)
    MTCPGw : ModbusTCPGateway_v1; (* Modbus TCP gateway *)
    TCPServer : SysTCPServer; (* TCP server *)
    SPort : SysSerialPort; (* Serial port *)
END_VAR

// ********************************************************************************
// PROGRAM "ST_ModbusTCPGateway_v1"
// ********************************************************************************
// The program operates as a Modbus gateway, accepts a Modbus TCP connection on
// port 2000 and converts Modbus TCP requests on Modbus RTU.
// --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------
    // INITIALIZATIONS
    // --------------------------------------------------------------------------
    // Program initializations.

    IF (SysFirstLoop) THEN

        // Initialize serial port.

        SPort.COM:=ADR('COM2'); //Serial port
        SPort.Baudrate:=19200; //Baud rate
        SPort.Parity:='E'; //Parity
        SPort.DataBits:=8; //Data bits
        SPort.StopBits:=1; //Stop bits
        SPort.DTRManagement:=DTR_AUTO_WO_TIMES; //DTR management

        // Initialize socket server.

        TCPServer.FilesArr:=ADR(Fp); //File array
        TCPServer.MaxConn:=3; //Number of connections accepted
        TCPServer.LocalAdd:=ADR('0.0.0.0'); //Local address
        TCPServer.LocalPort:=2000; //Local port
        TCPServer.LifeTm:=60; //TSocket life time (S)
        TCPServer.FlushTm:=10; //Socket flush time (mS)
        TCPServer.RxSize:=512; //Rx size buffer
        TCPServer.TxSize:=512; //Tx size buffer

        // Initialize Modbus TCP gateway.

        MTCPGw.IFTime:=1720; //Interframe time
        MTCPGw.Timeout:=500; //Command execution timeout (mS)
    END_IF;

    // --------------------------------------------------------------------------
    // MODBUS TCP/RTU GATEWAY
    // --------------------------------------------------------------------------
    // Execute the Modbus TCP/RTU gateway.

    SPort(Open:=TRUE); //Serial port management
    TCPServer(Enable:=TRUE); //TCP server management
    MTCPGw.MaxConn:=TCPServer.MaxConn; //Number of connections accepted
    MTCPGw.FpRTU:=SPort.File; //File pointer (Modbus RTU)
    MTCPGw.FpTCP:=ADR(Fp); //File pointer (Modbus TCP)
    MTCPGw(Enable:=(TCPServer.ConnPeers <> 0)); //Modbus TCP gateway

// [End of file]
```

9 VarSwap, swap variable value

Type	Library
FB	eLLabObsoleteLib



Questa funzione trasferisce eseguendone lo swap il valore della variabile ***Destination*** nella variabile ***Source*** in base al tipo ***Type*** definito. Gli indirizzi di Source e Destination possono coincidere rendendo possibile lo swap sul valore diretto di una variabile.

Type (*VR_TYPE*) Tipo di variabile .

Destination (*PVOID*) Indirizzo variabile di destinazione.

Source (*PVOID*) Indirizzo variabile sorgente. .

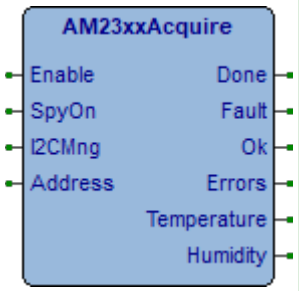
9.1 Esempi

Nell'esempio sono eseguite alcune operazioni di swap su variabili.

```
PROGRAM ST_VarSwap
VAR
    i : BOOL; (* Auxiliary variable *)
    ByteData : ARRAY[ 0..1 ] OF BYTE; (* Byte data example *)
    WordData : ARRAY[ 0..2 ] OF WORD; (* Word data example *)
    DWordData : ARRAY[ 0..2 ] OF DWORD; (* DWord data example *)
END_VAR

// ********************************************************************************
// PROGRAM "ST_VarSwap"
// ********************************************************************************
// This program shows the use of VarSwap function.
// --------------------------------------------------------------------------------

    // ----------------------------------------------------------------------------
    // SOME SWAPS
    // ----------------------------------------------------------------------------
    // Swap BYTE, Source:=16#12, Destination:=16#21.

    ByteData[0]:=16#12; //Byte data example
    i:=VarSwap(VR_TYPE#BYTE_TYPE, ADR(ByteData[1]), ADR(ByteData[0]));

    // Swap WORD, Source:=16#1234, Destination:=16#3412.

    WordData[0]:=16#1234; //Word data example
    i:=VarSwap(VR_TYPE#WORD_TYPE, ADR(WordData[1]), ADR(WordData[0]));

    // Swap LSB of WORD, Source:=16#3412, Destination:=16#3421.

    WordData[2]:=WordData[1];
    i:=VarSwap(VR_TYPE#BYTE_TYPE, ADR(WordData[2]), ADR(WordData[1]));

    // Swap DWORD, Source:=16#12345678, Destination:=16#56781234.

    DWordData[0]:=16#12345678; //DWord data example
    i:=VarSwap(VR_TYPE#DWORD_TYPE, ADR(DWordData[1]), ADR(DWordData[0]));

    // Swap MSW of WORD, Source:=16#56781234, Destination:=16#78561234.

    DWordData[2]:=DWordData[1];
    i:=VarSwap(VR_TYPE#WORD_TYPE, ADR(DWordData[2])+2, ADR(DWordData[1])+2);

// [End of file]
```

10 AM23xxAcquire, Aosong AM23xx sensor acquisition

Type	Library
FB	eLLabObsoleteLib



Questo blocco funzione esegue la lettura del valore di umidità e temperatura da un sensore AM2315 della Aosong. Può essere utilizzato in cascata con altri FB della libreria, collegando il Done di un FB con Enable di quello successivo è possibile creare catene di FB che condividono lo stesso bus I2C.

In I2CMng occorre passare l'indirizzo del FB I2CBusManager di gestione bus I2C, in Address definire l'indirizzo I2C del sensore da acquisire. Eseguita l'acquisizione si attiva l'uscita Done e se valori acquisiti si attiva l'uscita Ok ed in uscita si avranno i dati acquisiti.

Il sensore richiede una pausa tra le acquisizioni, quindi anche se eseguito ciclicamente il FB gestisce la corretta temporizzazione sulle acquisizioni.

Enable (BOOL) Comando abilitazione gestione. Da collegare a Done del FB precedente se utilizzato in cascata.

SpyOn (BOOL) Se attivo permette di spiare il funzionamento della FB (Vedi articolo).

I2CMng (@I2CBusManager) Indirizzo istanza FB I2CBusManager di gestione bus I2C.

Address (USINT) Indirizzo I2C del sensore (Default 16#44).

Done (BOOL) Esecuzione terminata, rimane attivo fino alla disabilitazione di Enable. Da collegare ad Enable del FB successivo se utilizzato in cascata.

Fault (BOOL) Attivo per un loop se errore acquisizione.

Ok (BOOL) Attivo per un loop su acquisizione dati.

Errors (UDINT) Ritorna conteggio numero di errori acquisizione.

Temperature (REAL) Valore di temperatura (°C).

Humidity (REAL) Valore umidità (%).

10.1 Esempi

Nell'esempio viene eseguita l'acquisizione del valore di temperatura e di umidità da un sensore AM2315 della Aosong connesso al bus I2C di estensione del modulo SlimLine.

```
PPROGRAM ST_AM23xxAcquire
VAR
    I2CBus : I2CBusManager; (* I2C bus management *)
    Sensor : AM23xxAcquire; (* Sensor acquisition *)
END_VAR

// ********************************************************************************
// PROGRAM "ST_AM23xxAcquire"
// ********************************************************************************
// Temperature and humidity acquisistion from a Aosong AM2320B attached to
// the extension bus.
// --------------------------------------------------------------------------------

    // --------------------------------------------------------------------------
    // INITIALIZATION
    // --------------------------------------------------------------------------
    // First program loop initialization.

    IF (SysFirstLoop) THEN

        // Initialize the I2C management.

        I2CBus.I2CMode:=0; //I2C mode selection

        // Initialize the sensor acquisition.

        Sensor.SpyOn:=TRUE; //Spy on
```

```
        Sensor.Address:=16#5C; //I2C Address
        Sensor.I2CMng:=ADR(I2CBus); //I2C bus managemen
    END_IF;

    // ------------------------------------------------------------------------
    // SENSOR ACQUISITION
    // ------------------------------------------------------------------------
    // The sensor is acquired.

    Sensor(Enable:=NOT(Sensor.Done)); //Sensor acquisition

// [End of file]
```