## 1.1 Libreria scambio dati su rete (eLLabDataExchLib)

***Attenzione! Per utilizzare la libreria occorre importarla nel proprio progetto***. Vedere capitolo relativo all'import delle librerie.

Questa libreria permette di gestire lo scambio dati tra due sistemi utilizzando una connessione in rete TCP/IP. Utilizzando questa libreria è possibile scambiare dati tra sistemi (Un master e molti slaves) sfruttando le conessioni di rete esistenti.

### 1.1.1 TCPDEXCHNODEDEFS, struttura definizione parametri nodo

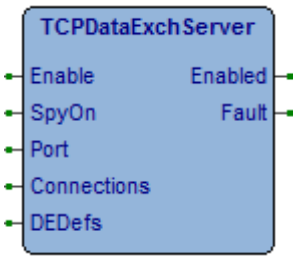
Questa struttura utilizzata dai blocchi funzione che eseguono lo scambio dati. Nella struttura oltre alle informazioni sui nodi sono definiti anche i buffers dei dati in scambio.

| Name | Type | | Description |
|---|---|---|---|
| NodeID | USINT | RW | Node ID. Numero di nodo del sistema, utilizzato per verificare la corrispondenza dei dati. |
| AutoTxD | BOOL | RW | Auto Tx data. Se attivato, in caso di variazione, viene forzato l'invio automatico del TxBuffer verso l'altro sistema. |
| TxData | BOOL | RW | Tx data send. Settandolo da programma utente si forza l'invio del TxBuffer verso l'altro sistema. ***Il comando rimane attivo fino alla avvenuta trasmissione*** poi viene automaticamente resettato. |
| Active | BOOL | R | Data exchange active. Si attiva se scambio dati sul nodo è attivo. |
| RxOk | BOOL | R | Rx data Ok. Settato su ricezione dati dall'altro sistema, i dati sono stati trasferiti in RxBuffer. Deve essere resettato da programa utente. |
| TxHeartbeat | USINT | RW | Tempo in secondi di invio frame di heartbeat. Ogni tempo definito viene inviato un frame dati all'altro sistema. |
| RxHeartbeat | USINT | R | Tempo in secondi di ricezione frame di heartbeat, viene ricevuto il valore di ***TxHeartbeat*** inviato dall'altro sistema. Ogni tempo definito deve essere ricevuto un frame dati all'altro sistema. |
| RxBuffer | @BYTE | RW | Rx buffer address. Indirizzo di allocazione del buffer dati ricevuti dall'altro sistema. |
| RxLength | UDINT | RW | Rx buffer length. Dimensione del buffer dati ricevuti dall'altro sistema. Deve coincidere con TxLength definito sull'altro sistema. |
| TxBuffer | @BYTE | RW | Tx buffer address. Indirizzo di allocazione del buffer dati da inviare verso l'altro sistema. |
| TxLength | UDINT | RW | Tx buffer length. Dimensione del buffer dati da inviare all'altro sistema. Deve coincidere con RxLength definito sull'altro sistema. |
| RxPackets | UDINT | R | Rx packets. Counter pacchetti dati ricevuti dall'altro sistema. |
| TxPackets | UDINT | R | Tx packets. Counter pacchetti dati inviati all'altro sistema. |
| TxTime | REAL | R | Tempo in secondi di invio dati all'altro sistema. |
| CErrors | UDINT | R | Communication errors. Counter errori di comunicazione con l'altro sistema. |

I campi RW devono essere impostati dal programma utente, i campi R ***non devono essere modificati*** da programma utente, possono solo essere letti.

### 1.1.2 TCPDataExchServer, TCP data exchange (Server)

| Type | Library |
|---|---|
| FB | eLLabDataExchLib_A000 |




Questo blocco funzione gestisce lo scambio dati su rete TCP agendo come server. Attivando l'FB si pone in ascolto il server sulla sulla porta definita in ***Port***, il server accetta il numero contemporaneo di connessioni dai client definite in ***Connections***. In ***DEDefs*** occorre definire l'indirizzo dell'array di strutture ***TCPDEXCHNODEDEFS*** che identificano i nodi slave che possono connettersi al server.

La gestione della comunicazione andrà gestita tramite le variabili nella struttura dati. Alla connessione di un client viene cercato nell'array indicato da ***DEDefs*** il ***NodeID*** del messaggio ricevuto ed i due sistemi si "legano" per lo scambio dati, il bit ***Active*** verrà attivato.

Per inviare i dati al sistema client occorre attivare il bit ***TxData***, il bit è automaticamente resettato a fine trasmissione. Se è attivo il bit ***AutoTxD*** su variazione dei dati nel buffer Tx, i dati sono automaticamente inviati al client. Su ricezione dati da un client avremo il bit ***RxOk*** attivo per un loop.

| | |
|---|---|
| **Enable** (BOOL) | Comando attivazione server. |
| **SpyOn** (BOOL) | Se attivo permette di spiare il funzionamento. |
| **Port** (UINT) | Porta TCP su cui porre in ascolto il server. |
| **Connections** (USINT) | Numero di connessioni contemporanee gestite. |
| **DEDefs** (@TCPDEXCHNODEDEFS) | Indirizzo allocazione array struttura ***TCPDEXCHNODEDEFS*** di definizione nodi client. |
| **Enabled** (BOOL) | Blocco funzione abilitato. |
| **Fault** (BOOL) | Attivo per un loop di programma se errore gestione. |

## 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 | ***Rx:*** Sequenze ricezione frame. |
| 16#00000002 | ***Tx:*** Sequenze trasmissione frame. |
| 16#40000000 | ***Er:*** Errore gestione. |

## Codici di errore

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

| Codice | Descrizione |
|---|---|
| 10061020 | FB eseguita in una task diversa dalla task di background. |
| 10061025 | ***DEDefs*** non impostato correttamente. |
| 10061030 | ***Connections*** non impostato correttamente. |
| 10061050 | E' stato modificato il numero di connessioni (***Connections***) definito. |
| 10061060~1 | Errore nella allocazione memoria gestione nodi funzione ***SysRMAlloc***. |
| 10061070 | ***Connections*** non impostato correttamente. |
| 10061100 | Il client ha chiuso la connessione. |
| 10061110 | Timeout invio heartbeat. |
| 10061120 | Timeout ricezione heartbeat. |
| 10061130 | Timeout esecuzione. |
| 10061200 | Errore nella allocazione memoria copia buffer Tx funzione ***SysRMAlloc***. |
| 10061210 | E' stata modificata la dimensione del buffer di Tx (***TxLength***). |
| 10061300~3 | Errore sequenze ricezione frame dati da client. |
| 10061310 | Lunghezza dati ricevuti non corretta (***TxLenght*** del client diverso da ***RxLength***). |
| 10061400 | Errore sequenze trasmissione frame dati verso client. |
| 10061900 | Errore case gestione. |

## Esempi

Nell'esempio viene gestito un server che gestisce lo scambio con due sistemi client.

### Definizione variabili

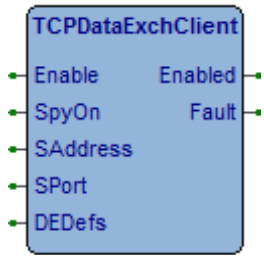
```
VAR
  C1RxBuffer : ARRAY[ 0..7 ] OF BYTE; (* Rx buffer (Client 1) *)
  C1TxBuffer : ARRAY[ 0..7 ] OF BYTE; (* Tx buffer (Client 1) *)
  C2RxBuffer : ARRAY[ 0..7 ] OF BYTE; (* Rx buffer (Client 2) *)
  C2TxBuffer : ARRAY[ 0..7 ] OF BYTE; (* Tx buffer (Client 2) *)
  DEDefs : ARRAY[ 0..1 ] OF TCPDEXCHNODEDEFS; (* Data exchange node definitions *)
  TCPServer : TCPDataExchServer; (* TCP data exchange server *)
END_VAR
```

### Esempio ST

```
  (* Program initialization *)

  IF (SysFirstLoop) THEN

    (* Set the client 1 definitions. *)

    DEDefs[0].NodeID:=1; (* Node ID *)
    DEDefs[0].AutoTxD:=TRUE; (* Automatic Tx data send *)
    DEDefs[0].RxBuffer:=ADR(C1RxBuffer); (* Rx buffer address *)
    DEDefs[0].RxLength:=SIZEOF(C1RxBuffer); (* Rx buffer length *)
    DEDefs[0].TxBuffer:=ADR(C1TxBuffer); (* Tx buffer address *)
    DEDefs[0].TxLength:=SIZEOF(C1TxBuffer); (* Tx buffer length *)
    DEDefs[0].TxHeartbeat:=5; (* Tx heartbeat time (S) *)

    (* Set the client 2 definitions. *)

    DEDefs[1].NodeID:=2; (* Node ID *)
    DEDefs[1].AutoTxD:=TRUE; (* Automatic Tx data send *)
    DEDefs[1].RxBuffer:=ADR(C2RxBuffer); (* Rx buffer address *)
    DEDefs[1].RxLength:=SIZEOF(C2RxBuffer); (* Rx buffer length *)
    DEDefs[1].TxBuffer:=ADR(C2TxBuffer); (* Tx buffer address *)
    DEDefs[1].TxLength:=SIZEOF(C2TxBuffer); (* Tx buffer length *)
    DEDefs[1].TxHeartbeat:=10; (* Tx heartbeat time (S) *)

    (* Server configuration. *)

    TCPServer.SpyOn:=TRUE; (* Spy command *)
    TCPServer.Port:=10000; (* Peer port *)
    TCPServer.Connections:=2; (* Accepted connections *)
    TCPServer.DEDefs:=ADR(DEDefs); (* Data exchange definitions *)
  END_IF;

  (* Manage data exchange server. *)

  TCPServer(Enable:=TRUE); (* Manage the server *)

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

### 1.1.3 TCPDataExchClient, TCP data exchange (Client)

| Type | Library |
|---|---|
| FB | eLLabDataExchLib_A000 |




Questo blocco funzione gestisce lo scambio dati su rete TCP agendo come client. Su attivazione l'FB si connette al server definito in ***SAddress*** alla porta definita in ***Sport***. In ***DEDefs*** occorre definire l'indirizzo della struttura ***TCPDEXCHNODEDEFS*** che identifica il nodo slave nella connessione con il server.

La gestione della comunicazione andrà gestita tramite le variabili nella struttura dati. Se la connessione al server viene stabilita (Il ***NodeID*** corrisponde con uno di quelli accettati dal server) i due sistemi si "legano" per lo scambio dati ed il bit ***Active*** della struttura dati verrà attivato.

Per inviare i dati al sistema server occorre attivare il bit ***TxData***, il bit è automaticamente resettato a fine trasmissione. Se è attivo il bit ***AutoTxD*** su variazione dei dati nel buffer Tx, i dati sono automaticamente inviati al server. Su ricezione dati da un server avremo il bit ***RxOk*** attivo per un loop.

**Enable** (BOOL) Comando attivazione client.

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

**SAddress** (@USINT) Indirizzo IP o URL del server a cui connettersi.

**SPort** (UINT) Porta TCP a cui connettersi sul server.

**DEDefs** (@TCPDEXCHNODEDEFS) Indirizzo allocazione struttura ***TCPDEXCHNODEDEFS*** di definizione nodo.

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

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

## 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 | ***Rx:*** Sequenze ricezione frame. |
| 16#00000002 | ***Tx:*** Sequenze trasmissione frame. |
| 16#40000000 | ***Er:*** Errore gestione. |

## Codici di errore

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

| Codice | Descrizione |
|---|---|
| 10062020 | FB eseguita in una task diversa dalla task di background. |
| 10062025 | ***DEDefs*** non impostato correttamente. |
| 10062100 | Manca connessione al server. |
| 10062110 | Timeout invio heartbeat. |
| 10062120 | Timeout ricezione heartbeat. |
| 10062130 | Timeout esecuzione. |
| 10062200 | Errore nella allocazione memoria copia buffer Tx funzione ***SysRMAlloc***. |
| 10062210 | E' stata modificata la dimensione del buffer di Tx (***TxLength***). |
| 10062300~1 | Errore sequenze ricezione frame dati da server. |
| 10062310 | Lunghezza dati ricevuti non corretta (***TxLenght*** del server diverso da ***RxLength***). |
| 10062400 | Errore sequenze trasmissione frame dati verso server. |
| 10062900 | Errore case gestione. |

## Esempi

Nell'esempio viene gestito un client che gestisce lo scambio con un sistema server. Come si vede viene definito come indirizzo del serve l'IP 172.0.0.1 che è il localhost, in questo modo è possibile sullo stesso modulo CPU fare dialogare il FB TCPDataExchClient con il TCPDataExchServer.

### Definizione variabili

```
VAR
  TxBuffer : ARRAY[ 0..7 ] OF BYTE; (* Tx buffer address *)
  RxBuffer : ARRAY[ 0..7 ] OF BYTE; (* Rx buffer address *)
  DEDefs : TCPDEXCHNODEDEFS; (* Data exchange node definitions *)
  TCPClient : TCPDataExchClient; (* TCP data exchange client *)
END_VAR
```

### Esempio ST

```
  (* Program initialization *)

  IF (SysFirstLoop) THEN

    (* Set the client definitions. *)

    DEDefs.NodeID:=1; (* Node ID *)
    DEDefs.AutoTxD:=TRUE; (* Automatic Tx data send *)
    DEDefs.RxBuffer:=ADR(RxBuffer); (* Rx buffer address *)
    DEDefs.RxLength:=SIZEOF(RxBuffer); (* Rx buffer length *)
    DEDefs.TxBuffer:=ADR(TxBuffer); (* Tx buffer address *)
    DEDefs.TxLength:=SIZEOF(TxBuffer); (* Tx buffer length *)
    DEDefs.TxHeartbeat:=1; (* Tx heartbeat time (S) *)

    (* Client configuration. *)

    TCPClient.SpyOn:=TRUE; (* Spy command *)
    TCPClient.SAddress:=ADR('127.0.0.1'); (* Server address *)
    TCPClient.SPort:=10000; (* Server port *)
    TCPClient.DEDefs:=ADR(DEDefs); (* Client definitions *)  END_IF;
  END_IF;

  (* Manage data exchange client. *)

  TCPClient(Enable:=TRUE); (* Manage the client *)

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