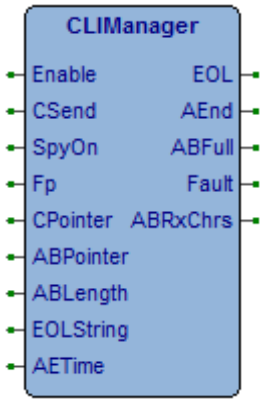
1.1.23 CLIManager, manages a command-line user interface

Type	Library
FB	eLLabUtyLib_C050



Questo FB gestisce l'interfaccia a linea di comando di un dispositivo. Occorre passare in ***Fp*** lo stream di comunicazione con il dispostivo, in ***CPointer*** la stringa di comando, attivando ***CSend*** il comando viene inviato al dispositivo e si riceve la risposta.

In ***ABPointer*** ed ***ABLength*** occorre definire l'indirizzo e la dimensione del buffer dove verrà trasferita la risposta. Se si desidera controllare la ricezione di ogni singola riga della risposta basterà definire in ***EOLString*** la stringa di terminazione riga (Esempio 'rn').

In ***AETime*** è possibile definire il tempo di fine ricezione risposta, se non sono ricevuti caratteri per il tempo definito, si attiva per un loop l'uscita ***AEnd***.

L'uscita ***EOL*** si attiva per un loop alla ricezione di una riga (Solo se definito ***EOLString***), l'uscita ***ABFull*** si attiva per un loop se il buffer di ricezione risposta è pieno. In ***ABRxChrs*** è ritornato il numero di caratteri di risposta ricevuti.

In caso di errore viene attivata per un loop di programma l'uscita ***Fault***.

Enable (BOOL) Abilitazione FB.

CSend (BOOL) Sul fronte di attivazione viene inviato al dispositivo il comando definito in ***CPointer***.

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

File (FILEP) Flusso dati ***stream*** di comunicazione con il dispositivo.

CPointer (@STRING) Puntatore alla stringa di definizione comando da inviare.

ABPointer (@STRING) Puntatore al buffer di memorizzazione della risposta.

ABLength (@STRING) Dimensione buffer di memorizzazione della risposta.

EOLString (@STRING) Puntatore alla stringa di fine linea.

AETime (UINT) Tempo di attesa fine ricezione risposta.

EOL (BOOL) Attivo per un loop su fine ricezione riga (Solo se definito **EOLString**).

AEnd (BOOL) Attivo per un loop su fine ricezione risposta (Non ricevuti caratteri per tempo **AETime**).

ABFull (BOOL) Attivo per un loop se buffr risposta pieno (Ricevuti caratteri definitri in **ABLength**).

Fault (BOOL) Attivo per un loop se errore.

ABRxChrs (UDINT) Numero di caratteri di risposta ricevuti.

Codici di errore

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

Codice	Descrizione
10082020	FB eseguito in task diversa da back.

Esempi

Viene eseguito il login su un sistema SlimLine (Userbname Admin, Password Admin) e viene acquisito il comando HwStats. La risposta al comando viene ricevuta riga per riga (Fine linea <CR><LF>). Dalla risposta ricevuta viene rilevato il numero di accensioni ***SwOn*** e la temperatura della CPU ***Temperature***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	ABuffer	STRING	Auto	[60]		..	Answer buffer
2	CaseAx	USINT	Auto	No		..	Case auxiliary
3	CaseNr	USINT	Auto	No		..	Case gestione
4	CLI	CLIManager	Auto	No		..	Command line manager
5	ErrorNr	USINT	Auto	No		..	Error number
6	Logged	BOOL	Auto	No		..	Logged in
7	Pulse	BOOL	Auto	No		..	Command pulse
8	RVars	USINT	Auto	No		..	Readed variables
9	SwOn	UDINT	Auto	No		..	Switched on times
10	TCPClient	SysTCPClient	Auto	No		..	TCP Client
11	Temperature	REAL	Auto	No		..	Internal temperature
12	TimeBf	UDINT	Auto	No		..	Time buffer (uS)

Esempio ST *(PTP114A660, ST_CLIManager)*

```
(* Eseguo inizializzazioni. *)

IF (SysFirstLoop) THEN

  (* Inizializzo TCP client. *)

  TCPClient.PeerAdd:=ADR('192.168.0.230'); (* Peer address *)
  TCPClient.PeerPort:=23; (* 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 *)

  (* Inizializzo interfaccia comandi. *)

  CLI.SpyOn:=TRUE; (* Spy On *)
  CLI.ABPointer:=ADR(ABuffer); (* Answer buffer pointer *)
  CLI.ABLength:=SIZEOF(ABuffer); (* Answer buffer length *)
  CLI.AETime:=1000; (* Answer end time (mS) *)

  (* Inizializzo variabili. *)

  TimeBf:=SysGetSysTime(TRUE); (* Time buffer (uS) *)
END_IF;

(* ---------------------------------------------------------------------- *)
(* CONNESSIONE AL SERVER TELNET                                           *)
(* ---------------------------------------------------------------------- *)
(* Manage the TCP client. *)

TCPClient(); (* TCPClient *)
CLI.Fp:=TCPClient.File; (* File pointer *)
CLI(Enable:=TCPClient.Connected); (* Command interface *)
CLI.CSend:=FALSE; (* Command send *)

(* ---------------------------------------------------------------------- *)
(* GESTIONE FAULT                                                         *)
(* ---------------------------------------------------------------------- *)
(* Gestione fault. *)

IF (ErrorNr <> 0) THEN CaseNr:=0; END_IF;

(* ---------------------------------------------------------------------- *)
(* GESTIONE TIMEOUT SEQUENZA                                              *)
(* ---------------------------------------------------------------------- *)
```

```
(* Viene gestito il timeout esecuzione sequenze. *)

IF ((CaseNr = 0) OR (CaseNr <> CaseAx)) THEN TimeBf:=SysGetSysTime(TRUE); END_IF;
IF ((SysGetSysTime(TRUE)-TimeBf) > 10000000) THEN ErrorNr:=10; RETURN; END_IF;
CaseAx:=CaseNr; (* Case auxiliary *)

(* ---------------------------------------------------------------------- *)
(* GESTIONE SEQUENZE                                                      *)
(* ---------------------------------------------------------------------- *)
(* Gestione sequenze programma. *)

CASE (CaseNr) OF

  (* ---------------------------------------------------------------------- *)
  (* CONNESSIONE AL SERVER                                                  *)
  (* ---------------------------------------------------------------------- *)
  (* Eseguo connessione al server. *)

  0:
  TCPClient.Connect:=FALSE; (* Connessione al server *)
  IF (Di00CPU = Pulse) THEN RETURN; END_IF;
  Pulse:=Di00CPU; (* Command pulse *)
  IF NOT(Di00CPU) THEN RETURN; END_IF;

  (* Eseguo connessione al server. *)

  RVars:=0; (* Readed variables *)
  ErrorNr:=0; (* Error number *)
  TCPClient.Connect:=TRUE; (* Connessione al server *)
  CaseNr:=CaseNr+1; (* Case gestione *)

  (* ---------------------------------------------------------------------- *)
  (* Controllo se socket connesso attendo prompt. *)

  1:
  IF NOT(TCPClient.Connected) THEN RETURN; END_IF;
  IF NOT(CLI.AEnd) THEN RETURN; END_IF;
  IF (SysStrFind(ADR(ABuffer), ADR('Login:'), FIND_GET_END) = NULL) THEN ErrorNr:=20; RETURN; END_IF;

  (* Invio login ed attendo richiesta password. *)

  CLI.CPointer:=ADR('Admin$r'); (* Invio Username *)
  CLI.EOLString:=ADR('Password:'); (* Attendo stringa "Password:" *)
  CLI.CSend:=TRUE; (* Command send *)
  CaseNr:=CaseNr+1; (* Case gestione *)

  (* ---------------------------------------------------------------------- *)
  (* Controllo se ricevuto richiesta password. se fine risposta errore. *)

  2:
  IF (CLI.AEnd) THEN ErrorNr:=30; RETURN; END_IF;
  IF NOT(CLI.EOL) THEN RETURN; END_IF;
  CLI.CPointer:=ADR('Admin$r'); (* Invio Password *)
  CLI.EOLString:=NULL; (* End of line string *)
  CLI.CSend:=TRUE; (* Command send *)
  CaseNr:=CaseNr+1; (* Case gestione *)

  (* ---------------------------------------------------------------------- *)
  (* Attendo prompt di sistema "[Admin]>". Se il buffer di appoggio è *)
  (* piccolo potrebbe essere nella prima ricezione su buffer pieno. *)

  3:
  IF (CLI.ABFull OR CLI.AEnd) THEN
    IF (SysStrFind(ADR(ABuffer), ADR('[Admin]>'), FIND_GET_END) <> NULL) THEN Logged:=TRUE; END_IF;
  END_IF;

  (* Controllo se fine ricezione risposta. Si deve esere loggati. *)

  IF NOT(CLI.AEnd) THEN RETURN; END_IF;
  IF NOT(Logged) THEN ErrorNr:=40; RETURN; END_IF;
  CaseNr:=10; (* Case gestione *)

  (* ---------------------------------------------------------------------- *)
  (* USCITA COMANDO                                                         *)
  (* ---------------------------------------------------------------------- *)
  (* Invio comando richiesta status. Ogni riga di risposta al comando *)
  (* termina con <CR><LF>. *)
```

```
    10:
    CLI.CPointer:=ADR('HwStats$r');
    CLI.EOLString:=NULL; (* End of line string *)
    CLI.EOLString:=ADR('$r$n'); (* End of line string *)
    CLI.CSend:=TRUE; (* Command send *)
    CaseNr:=CaseNr+1; (* Case gestione *)

    (* ------------------------------------------------------------------ *)
    (* Controllo risposte. *)

    11:
    IF (CLI.EOL OR CLI.AEnd) THEN
      IF (SysVarsscanf(SysStrFind(ADR(ABuffer), ADR('Switched on times:..'), FIND_GET_END), '%d', UDINT_TYPE,
ADR(SwOn))) THEN RVars:=RVars+1; END_IF;
      IF (SysVarsscanf(SysStrFind(ADR(ABuffer), ADR('Internal temp. [C]:.'), FIND_GET_END), '%f', REAL_TYPE,
ADR(Temperature))) THEN RVars:=RVars+1; END_IF;
    END_IF;

    (* Controllo se fine ricezione risposta. Devo avere 2 variabili. *)

    IF NOT(CLI.AEnd) THEN RETURN; END_IF;
    IF (RVars <> 2) THEN ErrorNr:=50; RETURN; END_IF;
    CaseNr:=0; (* Case gestione *)
  END_CASE;

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