1.1 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

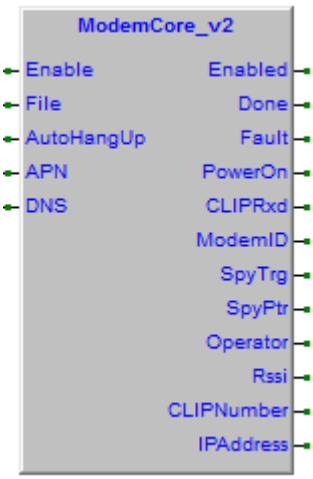
Messaggio SMS

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.

1.1.1 ModemCore_v2, modem core management

Type	Library	Version
FB	eModemLib	SFR057D000



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'FB ritorna un ***ModemID*** che deve essere passato alle FB collegate (Esempio invio SMS, ricezione SMS, ecc.). Le uscite ***SpyTrg*** e ***SpyPtr*** possono essere utilizzate per gestire la FB ***SpyData***.

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.

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 ***CLIPRxd***. Su connessione GPRS viene ritornato l'indirizzo IP assegnato dal gestore alla connessione ***IPAddress***.

Enable (BOOL)	Abilitazione blocco funzione, attivandolo viene gestito il modem.
File (FILEP)	Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.
AutoHangUp (BOOL)	Abilita l'HangUp automatico alla ricezione di una chiamata telefonica, viene comunque ritornato il CLIP.
APN (STING[32])	Definizione APN per connessione GPRS.
DNS (STING[16])	Definizione indirizzo IP server DNS.
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.
CLIPRxd (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.).
SpyTrg (UDINT)	Trigger per FB SpyData.
SpyPtr (@USINT)	Puntatore a buffer dati da spiare.
Operator (STRING[16])	Contiene la stringa con il nome dell'operatore telefonico.
Rssi (USINT)	Valore potenza segnale radio.
CLIPNumber (STRING[16])	Contiene la stringa con il numero di CLIP ricevuto.
IPAddress (STING[16])	Indirizzo IP assegnato dal gestore su connessione GPRS.

Codici di errore

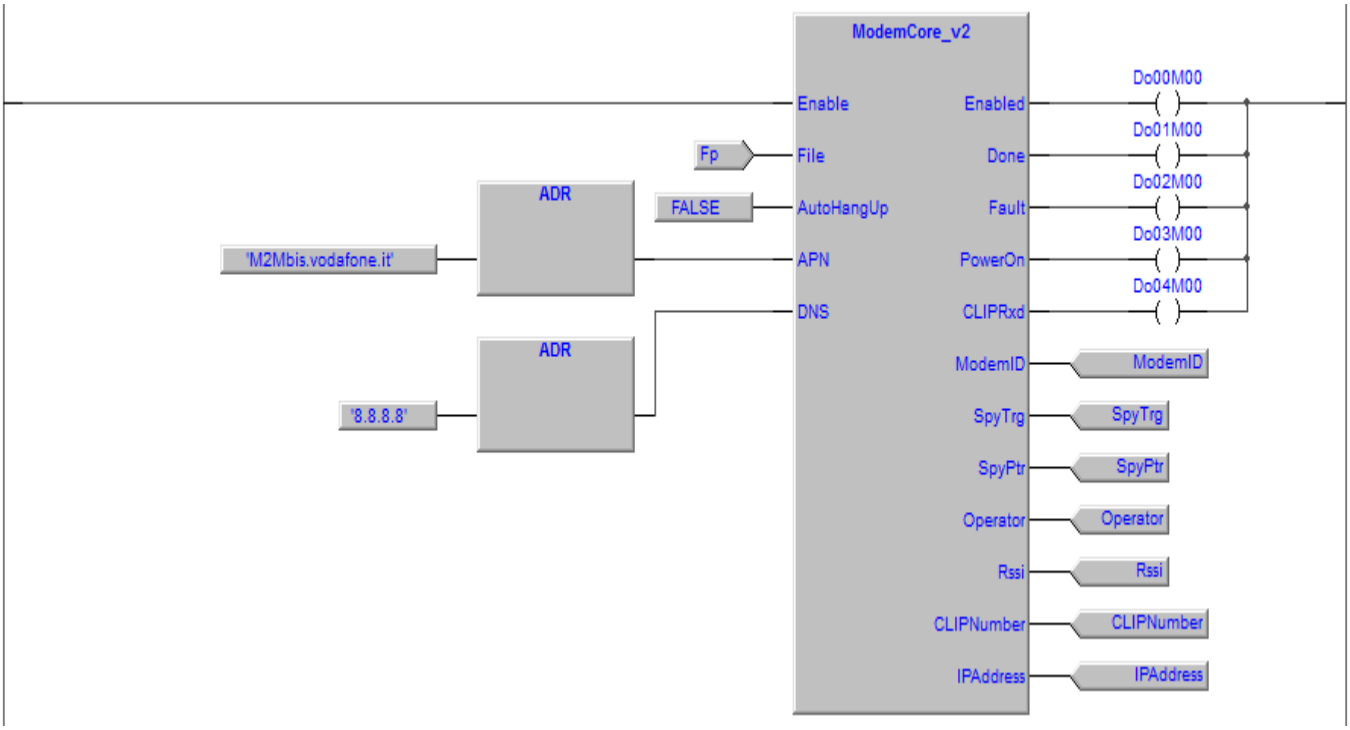
In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

Codice	Descrizione
10002010	Valore di ***File*** non definito.
10002020	FB protetta, terminato tempo funzionamento in modo demo.
10002050	Timeout esecuzione.
10002080	Errore case gestione.
10002100	Errore ricezione dati da modem.
10002200~3	Errore ricezione CLIP.
10002300~3	Errore nelle sequenze power on del modem.
10002400~9	Errore nelle sequenze di controllo del modem.
10002500~7	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



1.1.2 ModemSMSReceive, receive a SMS message

Type	Library	Version
FB	eModemLib	SFR057D000

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.

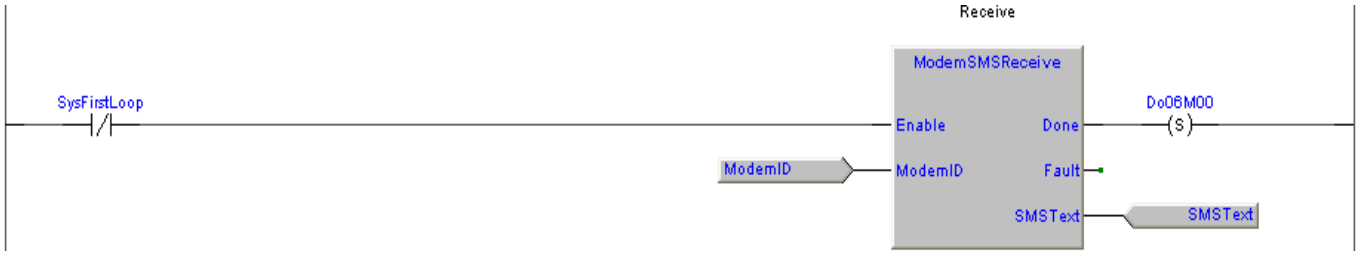
10003010 ***ModemID*** non definito.

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 *(Ptp118a200, ReceiveSMSMessage)*



1.1.3 ModemSMSRxCmd_v1, receive a SMS command

Type	Library	Version
FB	eModemLib	SFR057D000

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 (@USINT) Pointer al testo del 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.

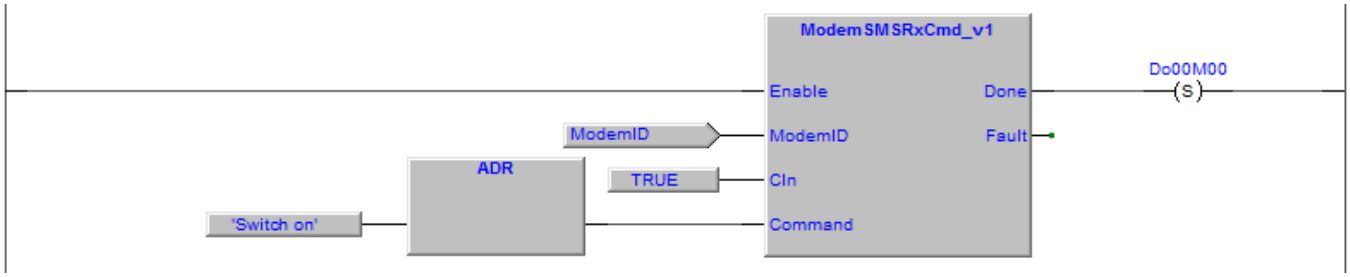
10004010 ***ModemID*** non definito.

10004020 ***ModemID*** non corretto.

Esempi

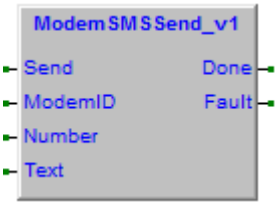
Nell'esempio è gestita la ricezione di un messaggio SMS dal modem definito nella variabile ***ModemID***, per la definizione delle variabili e per una migliore comprensione del funzionamento si rimanda agli esempi successivi.

Esempio LD *(Ptp118b000, ReceiveSMSCommand)*



1.1.4 ModemSMSSend_v1, send a SMS message

Type	Library	Version
FB	eModemLib	SFR057D000



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 (@USINT) Pointer al numero di telefono a cui eseguire l'invio del messaggio.

Text (@USINT) Pointer al 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.

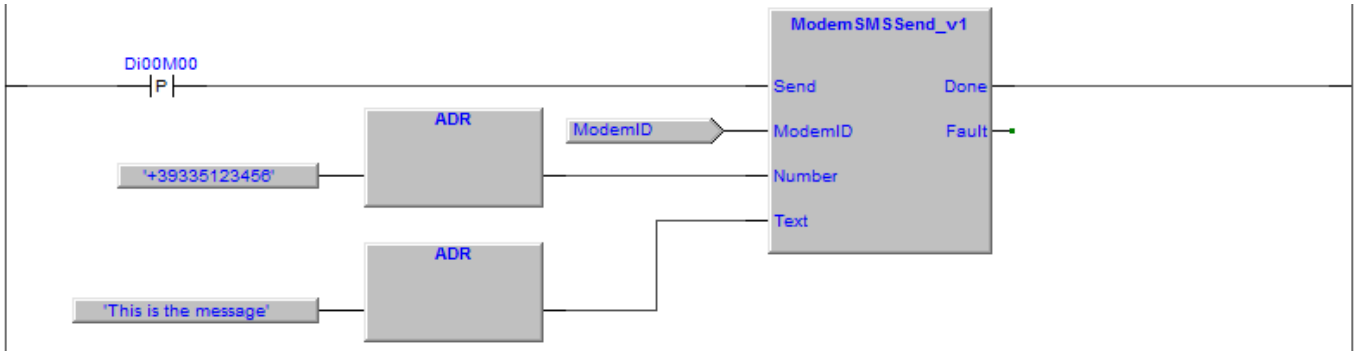
10005080 Errore case gestione.

10005100~3 Errore gestione invio SMS.

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 *(Ptp118b000, SendSMSMessage)*



Invio di un messaggio a più numeri

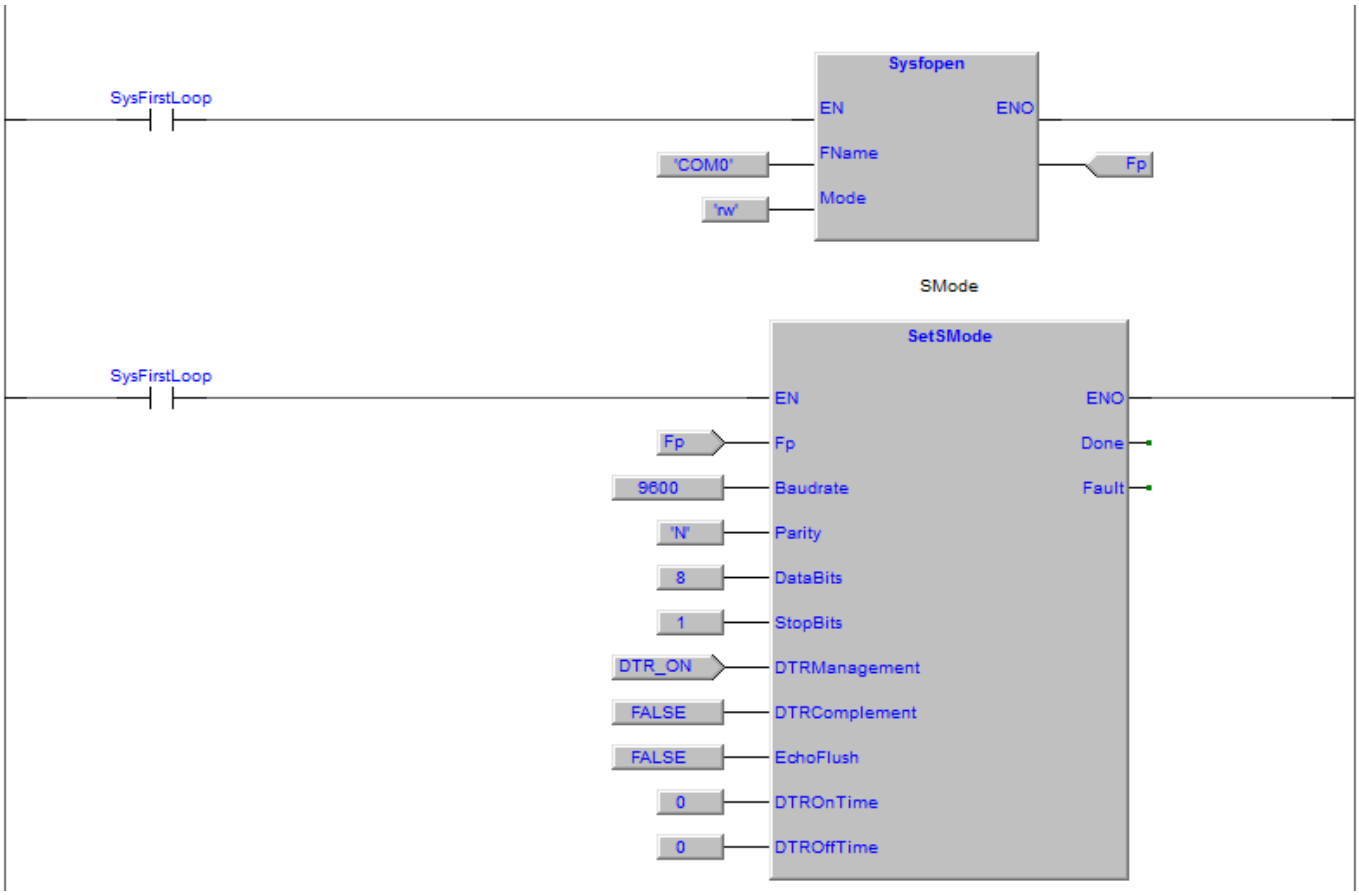
Molte volte succede di dover inviare su un evento (Esempio ingresso digitale) diversi messaggi SMS ognuno con il proprio testo a diversi numeri telefonici. Il blocco funzione ***SMSSend*** permette la gestione dell'invio sullo stesso evento di più messaggi, ogni messaggio è contraddistinto dal numero di telefono e dal testo del messaggio.

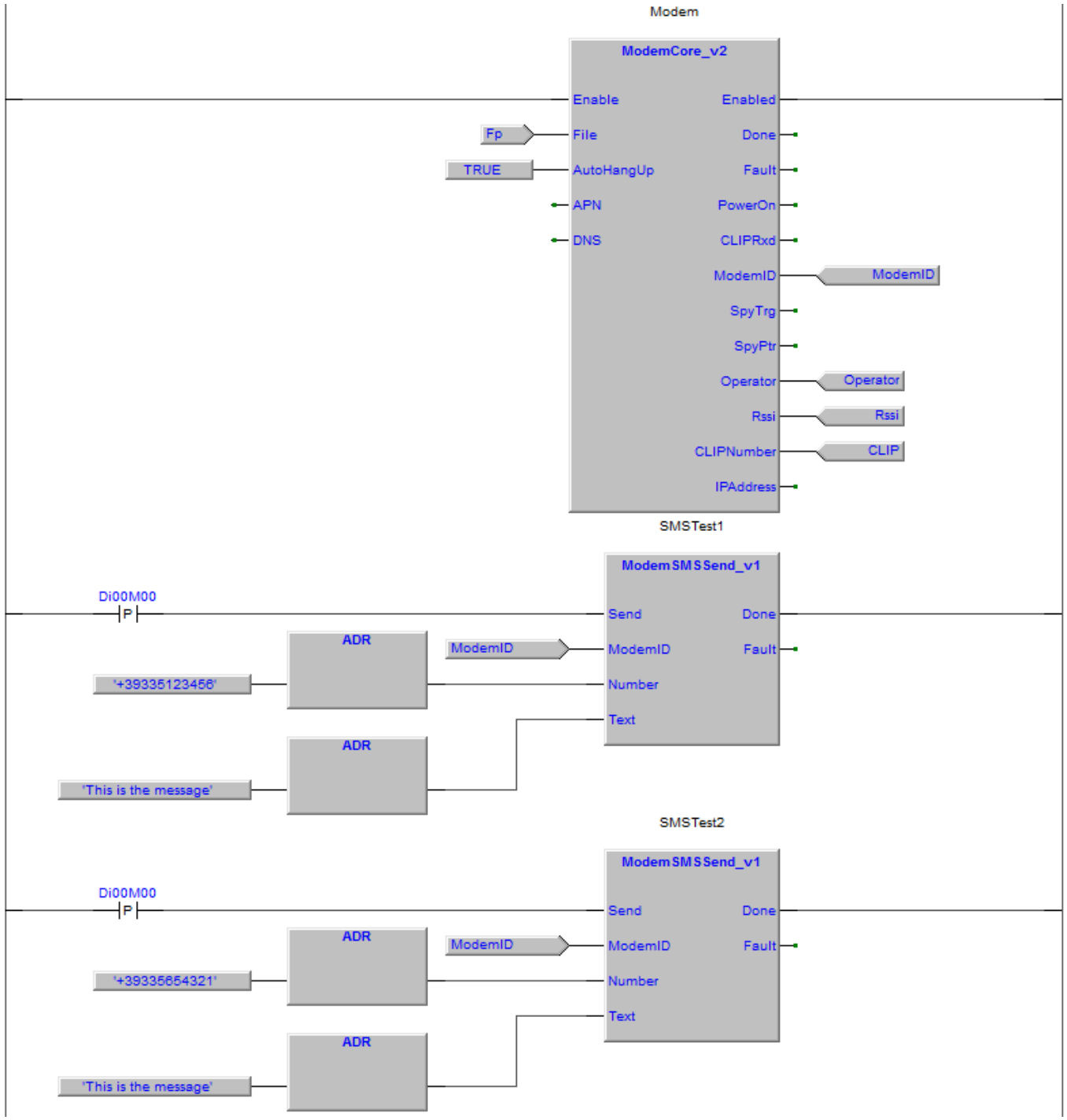
Nell'esempio è gestito l'invio di un messaggio a più numeri di telefono. Come si può vedere attivando l'ingresso digitale ***Di00M00*** verranno inviati due messaggi a due diversi numeri di telefono. E' possibile aggiungere altri rami con il blocco funzione ***SMSSend*** ognuno con il proprio messaggio e numero di telefono sempre attivati da ***Di00M00***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Rssi	USINT	Auto	No	0	..	RSSI value
2	ModemID	UDINT	Auto	No	0	..	Modem ID
3	CLIP	STRING	Auto	[32]		..	CLIP received number
4	Operator	STRING	Auto	[16]		..	Operator value
5	Rx	STRING	Auto	[32]		..	Modem Rx string
6	Tx	STRING	Auto	[32]		..	Modem Tx string
7	Fp	FILEP	Auto	No	0	..	Serial pointer
8	Modem	ModemCore_v1	Auto	No	0	..	Modem core FB
9	SMSTest1	ModemSMSSend_v1	Auto	No	0	..	Modem SMS send FB
10	SMSTest2	ModemSMSSend_v1	Auto	No	0	..	Modem SMS send FB
11	SMode	SetSMode	Auto	No	0	..	Serial mode FB

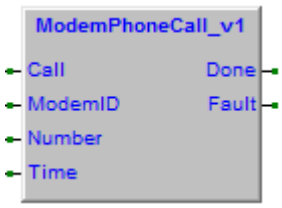
Esempio LD *(Ptp118b000, MultipleSMSSend)*




Modem
ModemCore_v2
Enable
Enabled
Fp
File
Done
TRUE
AutoHangUp
Fault
APN
PowerOn
DNS
CLIPRxd
ModemID
ModemID
SpyTrg
SpyPtr
Operator
Operator
Rssi
Rssi
CLIPNumber
CLIP
IPAddress
SMSTest1
ModemSMSSend_v1
Di00M00
P
Send
Done
ADR
ModemID
ModemID
Fault
'+39335123456'
Number
Text
ADR
'This is the message'
SMSTest2
ModemSMSSend_v1
Di00M00
P
Send
Done
ADR
ModemID
ModemID
Fault
'+39335654321'
Number
Text
ADR
'This is the message'

1.1.5 ModemPhoneCall_v1, executes a phone call

Type	Library	Version
FB	eModemLib	SFR057D000



Questo blocco funzione esegue una chiamata telefonica al numero indicato, si collega al blocco funzione di gestione modem ***ModemCore***, occorre passare il ***ModemID*** in uscita dal blocco funzione di gestione modem.

Su fronte attivazione ingresso di ***Call*** viene prenotata l'esecuzione della chiamata, non appena sarà possibile verrà eseguita la chiamata al numero definito in ***Number***. Terminato il tempo definito in ***Time*** la chiamata verrà terminata. Se non vi sono problemi verrà attivata per un loop di programma l'uscita ***Done***.

Call (BOOL) Sul fronte di attivazione comanda 'esecuzione della chiamata. ***Attenzione!*** La chiamata verrà eseguita non appena il modem è libero.

ModemID (UDINT) ID modem fornito in uscita dalla ***ModemCore***.

Number (@USINT) Pointer al numero di telefono da chiamare.

Time (UINT) Tempo di durata chiamata (Sec).

Done (BOOL) Attivo per un loop al termine della esecuzione chiamata.

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

Codici di errore

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

10037010 ***ModemID*** non definito.

10037020 ***ModemID*** non corretto.

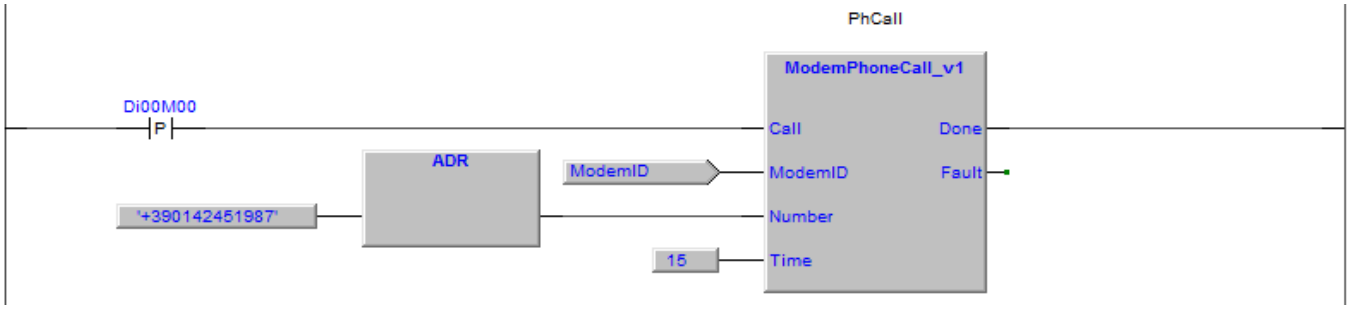
10037080 Errore case gestione.

10037100~3 Errore gestione chiamata telefonica.

Esempi

Nell'esempio è gestita una chiamata al numero indicato. Dopo 15 secondi la chiamata viene conclusa.

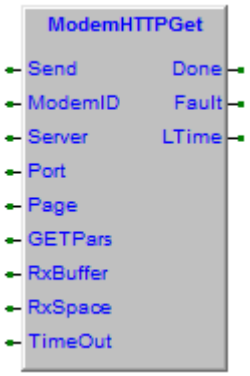
Esempio LD *(Ptp118b000, MakePhoneCall)*



1.1.6 ModemHTTPGet, executes a HTTP Get request

Type	Library	Version
FB	eModemLib	SFR057D000

Questo blocco funzione esegue una richiesta HTTP inserendo in linea dei parametri GET, si collega al blocco funzione di gestione modem ***ModemCore***, occorre passare il ***ModemID*** in uscita dal blocco funzione di gestione modem.



Su fronte attivazione ingresso ***Send*** viene eseguita la connessione al ***Server*** HTPP sulla porta ***Port***, e richiesta la pagina ***Page***. La pagina viene richiesta con i parametri GET definiti nel buffer puntato da ***GETPars***. La pagina ricevuta dal server viene trasferita nel buffer puntato da ***RxBuffer***, i dati ricevuti che superano la dimensione del buffer ***RxSpace*** sono scartati.

Terminata la ricezione della pagina richiesta si attiva per un loop l'uscita ***Done***, se pagina non è ritornata nel tempo definito in ***TimeOut*** l'esecuzione termina con errore.

Send (BOOL) Sul fronte di attivazione comanda richiesta pagina HTTP. ***Attenzione!*** La richiesta verrà eseguita non appena il modem è libero.

ModemID (UDINT) ID modem fornito in uscita dalla ***ModemCore***.

Server (@USINT) Pointer alla stringa di definizione del server HTTP.

Port (UINT) Porta su cui effettuare la connessione.

Page (@USINT) Pointer alla stringa di definizione pagina richiesta.

GETPars (@USINT) Pointer alla stringa parametri GET da inserire nella richiesta.

RxBuffer (@USINT) Pointer al buffer di ricezione pagina richiesta.

RXSpace (UINT) Dimensione in bytes del buffer di ricezione pagina.

TimeOut (UINT) Tempo massimo attesa ricezione pagina (mS).

Done (BOOL) Attivo per un loop al termine della ricezione pagina.

Fault (BOOL) Attivo per un loop se errore ricezione pagina.

LTime (UDINT) Tempo di caricamento pagina (mS).

Codici di errore

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

10042010 ***ModemID*** non definito.

10042020 ***ModemID*** non corretto.

10042080 Errore case gestione.

10042100~8 Errore gestione richiesta pagina da server HTTP.

Esempi

L'utilizzo tipico di questa FB è di collegarsi ad un server HTTP dove sono eseguite pagine script (ASP, PHP, Phyton, ecc) richiedendo la pagina fornendo in linea alla chiamata dei parametri in GET. Lo script della pagina potrà eseguire operazioni con i dati passati in linea e ritornare dei valori come risultato. Ecco un semplice esempio che tramite una pagina script PHP sul server permette di inviare una Email.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Pulse	BOOL	Auto	No	FALSE	..	Auxiliary pulse
2	Result	BOOL	Auto	No	FALSE	..	Result flag
3	ABf	INT	Auto	No	0	..	Auxiliary buffer
4	RxD	STRING	Auto	[32]		..	HTTP Rx data
5	TxD	STRING	Auto	[128]		..	HTTP Tx data
6	Mdm	ModemCore_v2	Auto	No	0	..	Modem management
7	HTTP	ModemHTTPGet	Auto	No	0	..	HTTP Get
8	Sm	SYSSERIALMODE	Auto	No	0	..	Serial mode

Esempio ST *(Ptp118b000, EmailWithHTTPGet)*

```
(* ********************************************************************** *)
(* PROGRAM "EmailWithHTTPGet"                                             *)
(* ********************************************************************** *)
(* This program sends an Email by contacting a PHP script.                *)
(* ---------------------------------------------------------------------- *)

    (* ---------------------------------------------------------------------- *)
    (* INITIALIZATION                                                         *)
    (* ---------------------------------------------------------------------- *)
    (* General initializations. *)

    IF (SysFirstLoop) THEN

        (* Opening the serial port on wich the modem is connected. *)

        Mdm.File:=Sysfopen('COM0', 'rw'); (* COM port file pointer *)

        (* Set the serial communication mode. *)

        ABf:=SysGetSerialMode(ADR(Sm), Mdm.File);
        Sm.Baudrate:=115200; (* Baudrate *)
        Sm.Parity:='N'; (* Parity *)
        Sm.DataBits:=8; (* Data bits *)
        Sm.StopBits:=1; (* Stop bits *)
        Sm.DTRManagement:=DTR_OFF; (* DTR management *)
        Sm.DTRComplement:=FALSE; (* DTR complement *)
        Sm.EchoFlush:=TRUE; (* Flush the echo caracters *)
        ABf:=SysSetSerialMode(ADR(Sm), Mdm.File);

        (* Modem parameters. Please change them according your requirements. *)

        Mdm.AutoHangUp:=TRUE; (* Auto hangup *)
        Mdm.APN:=ADR('M2Mbis.vodafone.it'); (* APN definition *)
        Mdm.DNS:=ADR('8.8.8.8'); (* DNS definition *)

        (* HTTP parameters. Please change them according your requirements. *)

        HTTP.Server:=ADR('www.MyServer.com'); (* HTTP server *)
        HTTP.Port:=80; (* HTTP port *)
        HTTP.Page:=ADR('eMailSend.php'); (* Page name *)
        HTTP.TimeOut:=5000; (* Connection timeout (mS) *)
        HTTP.GETPars:=ADR(TxD); (* GET parameters pointer *)
        HTTP.RxBuffer:=ADR(RxD); (* Rx buffer pointer *)
        HTTP.RxSpace:=SIZEOF(RxD); (* Rx buffer space *)
    END_IF;

    (* ---------------------------------------------------------------------- *)
    (* FBs MANAGEMENT                                                         *)
    (* ---------------------------------------------------------------------- *)
```

```
    (* Manage the modem core. *)

    Mdm(Enable:=TRUE); (* Modem core management *)
    HTTP.ModemID:=Mdm.ModemID; (* Modem ID *)

    (* Manage the HTTP connection. *)

    HTTP();
    HTTP.Send:=FALSE;

    (* ---------------------------------------------------------------------- *)
    (* GET THE HTTP PAGE AND SEND THE EMAIL                                   *)
    (* ---------------------------------------------------------------------- *)
    (* On digital input activation the HTTP page is requested. *)

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

        IF (Pulse) THEN
            ABf:=SysVarsnprintf(ADR(TxD), 128, 'To=%s', STRING_TYPE, ADR('sbertana@elsist.it'));
            ABf:=SysVarsnprintf(ADR(TxD)+LEN(TxD), 128-LEN(TxD), '$26Subject=%s', STRING_TYPE,
ADR('Subject'));
            ABf:=SysVarsnprintf(ADR(TxD)+LEN(TxD), 128-LEN(TxD), '$26Text=%s', STRING_TYPE, ADR('Text'));
             HTTP.Send:=TRUE;
        END_IF;
    END_IF;

    (* Check the HTTP request answer. *)

    IF (HTTP.Done) THEN
        IF (FIND(RxD, 'Ok') = 0) THEN Result:=FALSE; ELSE Result:=TRUE; END_IF;
    END_IF;

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

Vediamo di spiegare come funziona il programma, sul fronte di attivazione dell'ingresso ***Di00M00*** viene effettuata la connessione al server ***www.MyServer.com*** e richiesta l pagina ***eMailSend.php***. In linea alla chiamata della pagina sono posti i campi GET **To=sbertana@elsist.it&Subject=Subject&Text=Text**. Da notare che il carattere “&” è stato sostituito da “$26”.

La pagina ***eMailSend.php*** sul server contiene uno script PHP del tipo:

```
<?php
    $Headers="MIME-Version: 1.0\n";
    $Headers.="Content-type: text/html; charset=iso-8859-1\n";
    $Headers.="To: ".$_REQUEST['To']."\n";
    $Headers.="From: \n";
    mail($_REQUEST['To'], $_REQUEST['Subject'], $_REQUEST['Text'], $Headers);
    echo "Ok";
    exit();
?>
```

Come si vede lo script invia una mail tramite il comando mail i cui campi sono valorizzati dai parametri posti in GET alla richiesta. Lo statement ***$_REQUEST['To']*** è valorizzato con il testo definito in ***To=***, ***$_REQUEST['Subject']*** è valorizzato con il testo definito in ***Subject=***, e così via. Come si vede è possibille passare allo script tutti i valori che si desiderano.

Il valore di ritorno dallo script definito con lo statement echo verrà trasferito nel buffer ***RxD*** del nostro programma e quindi è possibile operare su di esso con le istruzioni di gestione stringa.