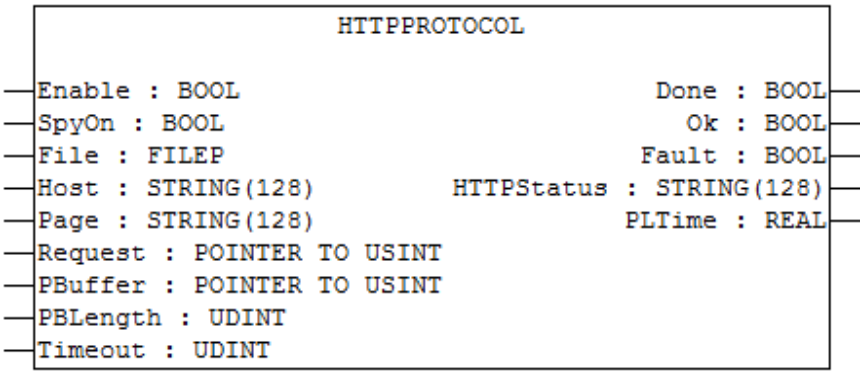
1.1.1 HTTPProtocol, HTTP protocol management

Type	Library
FB	eCDSNetworkLib_A000

Questo blocco funzione esegue la richiesta di una pagina web con il protocollo HTTP. Attivando l'ingresso ***Enable*** viene inviata la richiesta HTTP sullo stream di I/O definito in ***File***. Ricevuta la pagina si attiva per un loop di programma l'uscita ***Ok***.



Nel parametro ***Host*** occorre definire il nome host del server HTTP, mentre in ***Page*** occorre definire la pagina da richiedere. La pagina viene richiesta con i parametri definiti nel buffer ***Request***. La pagina richiesta viene ritornata nel buffer definito in ***Pbuffer***.

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***.

Enable (BOOL)	Comando attivazione richiesta pagina.
SpyOn (BOOL)	Se attivo permette di spiare il funzionamento della FB.
File (FILEP)	Flusso dati stream ritornato dalla funzione ***Sysfopen***.
Host (STRING[128])	Stringa di definizione Host name.
Page (STRING[128])	Stringa di definizione pagina web richiesta.
Request (POINTER)	Indirizzo buffer dati da inviare con la richiesta.
PBuffer (POINTER)	Indirizzo buffer pagina ricevuta.
PBLength (UDINT)	Dimensione buffer di pagina.
Timeout (UINT)	Timeout esecuzione richiesta pagina (mS).
Done (BOOL)	Attivo a fine esecuzione, si attiva anche in caso di ***Fault***.
Ok (BOOL)	Attivo per un loop di programma su ricezione pagina.
Fault (BOOL)	Attivo per un loop di programma se errore gestione.
HTTPStatus (STRING[128])	Status risposta HTTP ricevuta.
PLTime (REAL)	Tempo impiegato per caricamento pagina (S).

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	***Tx:*** Invio richiesta HTTP.
16#00000002	***Rx:*** Ricezione risposta HTTP.
16#10000000	***Lg:*** Messaggio di log.
16#20000000	***Wr:*** Messaggio di warning.
16#40000000	***Er:*** Messaggio di errore.

Codici di warning/errore

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

10055010	Er	Valore di ***File*** non definito
10054100	Er	Timeout esecuzione richiesta HTTP
10054200	Er	Campo request troppo lungo
10054300	Er	Errore ricezione HTTP Status
10054301	Wr	Campo HTTP Status troppo grande è stato troncato
10054350	Wr	Pagina ricevuta troppo grande è stata troncata

Esempi

Nell'esempio viene eseguita la richiesta della pagina **www.slimline.altervista.org/Tutorials/HTTPProtocol.php**. Sono passati alla pagina 2 parametri ***Dividend*** e ***Divisor***, la pagina è postata su Altervista ed al suo interno un semplice script PHP esegue la divisione tra i due valori passati e ritorna il risultato.

Ecco il codice del file script ***HTTPProtocol.php***.

<?php echo "The result is: ".$_REQUEST["Dividend"]/$_REQUEST["Divisor"]; ?>

Come si vede nel file di script PHP il valore delle variabili passate nella stringa ***Request*** viene assegnato alle due varibili PHP $_REQUEST["Dividend"] e $_REQUEST["Divisor"]. Il risultato della divisione viene ritornato nella pagina ed andrà a valorizzare la variabile ***PBuffer***.

Definizione variabili

```
VAR
	Fp: ARRAY [0..1] OF FILEP; (* File pointer *)
	OShot: R_TRIG; (* One shot *)
	Delay: TON; (* Delay timer *)
	DNS: DNSProtocol; (* DNS protocol *)
	HTTP: HTTPProtocol; (* HTTP protocol *)
	UDPCSk: SysUDPClient; (* UDP client socket *)
	TCPCSk: SysTCPClient; (* TCP client socket *)
	DomainIP: STRING(16); (* Domain IP *)
	RTime: UDINT; (* Reference time (uS) *)
	TTL: UDINT; (* Time to live (S) *)
	Request: STRING(32):='Dividend=100&Divisor=5'; (* Request string *)
	PBuffer: STRING(128); (* Page buffer *)
END_VAR
```

Esempio LD *(PTP131A000, LD_HTTPProtocol)*