EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH
CERN — PS DIVISION

CERN-PS-CO-2001-xx

A UNIX Fetch/Write Client Interface for SIMATIC PLCs

F. Locci, G. Mornacchi

Geneva, Switzerland
29 March, 2001

1. Introduction

We describe the client implementation, running under UNIX-like operating system, of the SIMATIC Fetch/Write protocol [1]. The implementation is intended to provide in a simple way communications, from a UNIX-like machine such as a LynxOS based DSC, with a SIMATIC S5 PLC.

This work is part of the upgrade of the ISOLDE front-end controls [2]; in particular it is part of the developments related to the controls of the vacuum and stepping motors. This equipment is driven by an old SIMATIC S5 based set of PLCs originally interfaced to a PC via a SINEC L2 (Siemens proprietary) bus. Because:

1) Ethernet has been selected by the project [2] to connect the existing PS control system front-end, i.e. a DSC, to PLCs, and

2) we do not want to modify the existing S5 PLC software.

The simplest way to satisfy both requirements is represented by a communication protocol which provides elementary read/write access to locations in a PLC data block. The fetch/write protocol is such a tool and it is implemented by the Siemens Ethernet couplers for both S5 and S7 series PLCs.

2. Protocol Description

The client fetch/write protocol is based on transactions: the client sends a request command specifying the required operation (fetch or write), the related data block in the PLC, the offset within the data block and the amount of 16 bit words to be transferred. The above information is packed in a (request) header by the client, in case of a write the data to be written into the PLC follow the header. The PLC, actually the Ethernet coupler, executes the command and replies with a message consisting of an (acknowledge) header, identifying the request and reporting the status of the original command, and the requested data when the original command was a fetch. The following tables define the request and acknowledgement headers for both the fetch and write commands (from [1]):

	Write Request Frame		**Write Acknowledgement Frame**	
Byte #	*Field Name*	*Value*	*Field Name*	*Value*
0	System ID	"S"	System ID	"S"
1	….	"5"		"5"
2	Header Length	16d	Header Length	16d
3	ID Op Code	1	ID OP Code	1
4	Op Code Length	3	Op Code Length	3
5	Op Code Length	3	Op Code Length	4
6	ORG Field	3	Ack Field	0Fh
7	Org Field length	8	S Field Length	3
8	ORG ID…		Error Number	
9	DB Number		Empty Field	FFh
10	Start…	High Part	Length Empty Field	7
11	…Address	Low part		
12	Number of..	High Part	Free	
13	..words..	Low Part		
14	Empty Field	FFh		
15	Empty Field Size	2		
16	Data up to 64K bytes			

Table 1. Write command headers.

The fetch/write protocol runs on top of the ISO-on-TCP protocol, RFC1006, as implemented by the Siemens S5/S7 Ethernet couplers. On the LynxOS/UNIX side the fetch/write client runs on top of the rfc1006 implementation [3] developed specifically for the “industrial components integration” project.

	Read Request Frame		Read Acknowledgement Frame	
Byte #	*Field Name*	*Value*	*Field Name*	*Value*
0	System ID	"S"	System ID	"S"
1	….	"5"		"5"
2	Header Length	16d	Header Length	16d
3	ID Op Code	1	ID OP Code	1
4	Op Code Length	3	Op Code Length	3
5	Op Code	5	Op Code	6
6	ORG Field	3	Ack Field	0Fh
7	Org Field length	8	S Field Length	3
8	ORG ID…		Error Number	
9	DB Number		Empty Field	FFh
10	Start…	High Part	Length Empty Field	7
11	…Address	Low part		
12	Number of..	High Part	Free	
13	..words..	Low Part		
14	Empty Field	FFh		
15	Empty Field Size	2		
			Data up to 64KB (if Error number = 0)	

3. Fetch/Write Client API

The fetch/write client is accessible through an API including open/close functions and fetch/write commands.

Open

s = FwOpen(char *ip, int port, char *dst, char *src, long ts)

Input	Parameter	Type	Meaning
	Ip	Char *	IP address of destination TCP coupler
	Port	Int	Port number of ISO-on-TCP server
	Dst	Char *	TSAP destination string
	Src	Char *	TSAP source string
	Ts	Long	Time out in seconds
Output	**Return**		
	s	Int	Socket number of the established connection if >0. Error code otherwise

FwOpen opens a connection with the PLC, at the entry points specified by the source and destination TSAPs. If successful it returns the local socket number identifying the connection. It is merely a redefinition of the rfcsock rfcConnect call.

Close

r = fwClose(int s, int shutdown, char *dst, char *src)

Input	Parameter	Type	Meaning
	s	int	Socket number as returned by fwOpen
	shutdown	Int	Control flag
	Dst	Char *	TSAP destination string
	Src	Char *	TSAP source string
Output	**Return**		

	r	Int	Status code: 0 if successful, <0 otherwise

FwClose requests the remote PLC to close the connection identified by the socket number s. It is a redefinition of the rfcsock rfcDisconnect function call.

Fetch

R = fwFetch(int s, unsigned char dbNumber, unsigned short startAddress, unsigned short size, unsigned char *data, int ts)

Input	Parameter	Type	Meaning
	s	int	Socket number, as returned by fwOpen
	dbNumber	Unsigned char	Data block number in PLC
	startAddress	Unsigned short	Offset (in 16-bit words) within the data block
	Size	Unsigned short	Number of 16-bit words to fetch from PLC
	data	Unsigned char *	User data array (filled with the fetch result)
	ts	Int	Time out in seconds
Output	**Return**		
	r	Int	Status code: 0 if success, non 0 otherwise

FwFetch asks the Ethernet/TCP coupler to fetch “size” words from the PCL data block “dbNumber” starting at the (word) offset “startAddress”. The fetched values are then returned in the user “data” array.

Write

R = fwWrite(int s, unsigned char dbNumber, unsigned short startAddress, unsigned short size, unsigned char *data, int ts)

Input	Parameter	Type	Meaning
	s	int	Socket number, as returned by fwOpen
	dbNumber	Unsigned char	Data block number in PLC
	startAddress	Unsigned short	Offset (in 16-bit words) within the data block
	Size	Unsigned short	Number of 16-bit words to fetch from PLC
	data	Unsigned char *	User data array (data to be written to the PLC)
	ts	Int	Time out in seconds
Output	**Return**		
	r	Int	Status code: 0 if success, non 0 otherwise

FwWrite writes “size” words, as given in the user “data” array, into the PLC data block “dbNumber” starting at the (word) offset “startAddress”.

Errors

All the API functions return 0 in case of a successful completion, otherwise they may return the following error codes:

PS Error code	Meaning	Explanation
EQP_PROTOCOL	Invalid header	An invalid fetch/write protocol header was received from the PLC
EQP_MEMALLOC	No dynamic space	The library could not allocate dynamic memory via the malloc syscall
EQP_ARRDIMDIFF	Bad reply (fwFetch only)	The PLC did not return the requested number of words
Other errors		As returned by the underlying rfcsock library

Performance

Results from some simple performance measurement are shown below. These apply to two different configurations:

1) a DSC connected to a S5 PLC system via the Ethernet coupler CP 1430 TCP.

Fetch	Number of words	Times (ms)
	1	35.5
	10	37.5
Write		
	1	36
	10	37.5

2) a DSC connected to a S7-400 PLC via the YYYY Ethernet coupler.

Fetch	Number of Words	Time (ms)
	1	12.5
	10	12.6
	50	13.1
	100	13.6
	200	16.8
	250	195
	300	195
	400	198
	500	395

References

1) SIMATIC CP 1430 TCP coupler. Appendix B
2) J. Allard et al. Consolidation of the ISOLDE controls. PS/CO 2001-xx
3) F. Locci. Rfcsock package (iso-on-tcp client library)