Ethernet

Application Specific Function Block Manual

Version 14.0.1

Sheffield Automation, LLC

NOTE

Progress is an on-going commitment at Sheffield Automation, LLC. We continually strive to offer the most advanced products in the industry; therefore, information in this document is subject to change without notice. The illustrations and specifications are not binding in detail. Sheffield Automation, LLC shall not be liable for any technical or editorial omissions occurring in this document, nor for any consequential or incidental damages resulting from the use of this document.

DO NOT ATTEMPT to use any Sheffield Automation, LLC product until the use of such product is completely understood. It is the responsibility of the user to make certain proper operation practices are understood. Sheffield Automation, LLC products should be used only by qualified personnel and for the express purpose for which said products were designed.

Should information not covered in this document be required, contact the Customer Service Department, Sheffield Automation, LLC, 660 South Military Road, P.O. Box 1658, Fond du Lac, WI 54936-1658. Sheffield Automation, LLC can be reached by telephone at (920) 921–7100.

DISCLAIMER: All programs in this release (application demos, application specific function blocks (ASFB's), etc.) are provided "AS IS, WHERE IS", WITHOUT ANY WARRANTIES, EXPRESS OR IMPLIED. There may be technical or editorial omissions in the programs and their specifications. These programs are provided solely for user application development and user assumes all responsibility for their use. Programs and their content are subject to change without notice.

Release 2903

Part Number M.1301.6207

Table of Contents: Ethernet ASFB Manual

NOTES

CHAPTER 1 Application Specific Function Block Guidelines

Installation

The following guidelines are recommended ways of working with Application Specific Function Blocks (i.e. ASFBs) from Giddings & Lewis.

The Applications CD includes the ASFB package as follows:

- .LIB file(s) containing the ASFB(s)
- source .LDO(s) from which the ASFB(s) was made
- example LDO(s) with the ASFB(s) incorporated into the ladder which you can then use to begin programming from or merge with an existing application ladder

When you install the Applications CD, the ASFB paths default to:

C:\Program Files\Giddings & Lewis\Applications **v*xx.x.r***\ASFB

and

C:\Program Files\Giddings & Lewis\Applications **v*xx.x.r***\Examples

where **v*xx.x*** *is the PiCPro for Windows version number that these ASFBs and examples were built under. The* ***.r*** *is the revision number of the Application software itself.*

The .LIB files and source .LDO files are put in the ASFB subdirectory. The example .LDO files are put in the Examples subdirectory.

Revisions

The first four networks of each ASFB source ladder provide the following information:

Network 1

The first network just informs you that the ASFB is provided to assist your application development.

Network 2

The second network is used to keep a revision history of the ASFB. Revisions can be made by Giddings & Lewis personnel or by you.

The network identifies the ASFB, lists the requirements for using this ASFB, the name of the library the ASFB is stored in, and the revision history.

The revision history includes the date, ASFB version (see below), the version of PiCPro used while making the ASFB, and comments about what the revision involved.

When an ASFB is revised, the number of the first input (EN_ _ or RQ_ _) to the function block is changed in the software declarations table. The range of numbers available for Giddings & Lewis personnel is 00 to 49. The range of numbers available for you is 50 to 99. See chart below.

Revision	Giddings & Lewis revisions	User revisions
1st	EN00	EN50
2nd	EN01	EN51
.	.	.
.	.	.
.	.	.
50th	EN49	EN99

Network 3

The third network describes what you should do if you want to make a revision to the ASFB.

ASFB Input/Output Descriptions

Network 4

The fourth network describes the ASFB and defines all the inputs and outputs to the function block.

Using ASFBs

When you are ready to use the ASFB in your application, there are several approaches you can take as shown below.

- Create a new application LDO starting with the example LDO for the ASFB package. The advantage is that the software declarations table for the ASFB has been entered for you.
- If you already have an application LDO, copy and paste the example LDO into yours. The software declaration tables for both LDOs will also merge.

CHAPTER 2 Ethernet ASFBs

These are the Ethernet application specific function blocks (ASFBs). These ASFBs provide drop-in functionality for Ethernet support. They are easy to use function blocks that enable the PiC or MMC to communicate and share data with other devices over an Ethernet connection. They provide support for:

1. **PiC to PiC (or MMC) data exchange**

   There are two Ethernet protocol choices provided: TCP (transmission control protocol) and UDP (user datagram protocol). TCP provides much greater confirmation of receipt than does UDP. TCP also allows the streaming of data and larger data packets. UDP can result in less network traffic because each UDP message does not have to be acknowledged (which is the case with TCP).

2. **File transfers**

   TFTP (trivial file transfer protocol) is used over UDP to transfer individual RAM-DISK files over Ethernet.

3. **Internet E-mail sending capability**

   The PiC or MMC can send a user-programmable text message to a user-specified E-mail address via SMTP (simple mail transfer protocol). The PiC or MMC does not have the capability to receive or respond to E-mail with this ASFB.

   These function blocks implement several of the procedures described in the PiCPro Function/Function Block Reference Guide "Overview of Using Ethernet-TCP/IP Function Blocks", located after the IPWRITE function block description.

Note: The OPC Server (OLE for Process Control) capability over Ethernet is not part of this package. It is available separately. The OPC Server provides a tool for the exchange of data between a PC and a PiC or MMC using the widely accepted standard, OPC. That package consists of an ASFB for the PiC or MMC and the PC-resident OPC Server software.

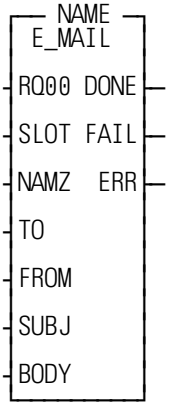
The files for this Ethernet ASFB package are listed on the following page.

Note: Every .LDO on the CD has a corresponding .REM file. The .REM file contains all the network comments found in the .LDO file. If you move a .LDO file to a different location, be sure to move its .REM file to the same directory.

E_INET.LIB	Library for the internet E-mail ASFB
E_MAIL.LDO	Source file for the E-mail (SMTP) client
E_PICXCG.LIB	Library for the data exchange ASFBs
E_TCPCL.LDO	Source file for the TCP client
E_TCPRD.LDO	Source file for the TCP data read
E_TCPSVR.LDO	Source file for the TCP server
E_TFTP.LIB	Library for the TFTP ASFBs
E_TFTPCL.LDO	Source file for the TFTP client
E_TFTPSV.LDO	Source file for the TFTP server
E_UDPCL.LDO	Source file for the UDP client
E_UDPSVR.LDO	Source file for the UDP server

Also included with the Ethernet ASFB package are example ladders for the use of the respective ASFBs as follows:

E_CLIENT.LDO	Example of the client ASFBs
E_SERVER.LDO	Example of the server ASFBs



Inputs: RQ00 (BOOL) - requests message (one shot)

SLOT (USINT) - slot number of the Ethernet module

NAMZ (STRING) - IP address of post office on LAN, zero-terminated

TO (STRING) - E-mail address of recipient

FROM (STRING) - sending PiC name (optional)

SUBJ (STRING) - subject header

BODY (STRING) - body of the message

Outputs: DONE (BOOL) - execution without error

FAIL (BOOL) - an error has occurred

ERR (INT) - error number from IP functions that occurred during execution

<<INSTANCE NAME>>:E_MAIL(RQ00 := <<BOOL>>, SLOT := <<USINT>>, NAMZ := <<INT>>, TO := <<USINT>>, FROM := <<USINT>>, SUBJ := <<USINT>>, BODY := <<USINT>>, DONE => <<BOOL>>, FAIL => <<BOOL>>, ERR => <<INT>>);

This function block enables a PiC or MMC to send a text-based e-mail message via SMTP using the Ethernet module. This function block does not allow the PiC or MMC to receive or respond to e-mail.

E_MAIL allows a PiC or MMC to send the text-based message entered at the BODY input to the SMTP address specified at the TO input. Up to 256 bytes of text may be sent as the BODY of the message, with additional bytes for the TO, FROM, and SUBJ inputs.

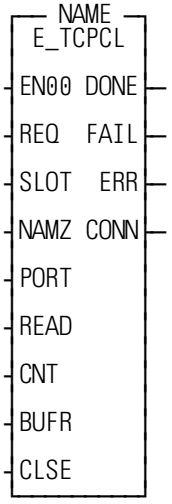
In order to use this function block, you must have an SMTP server installed on your network. The address of the SMTP Server (or the "Post Office") must be specified at NAMZ. SMTP server software may be obtained on the World Wide Web. This function block was tested using Seattle Labs SLMail 2.7 for Windows 95, available from http://www1.seattlelab.com

The FROM input is used only to specify the origin of the message. It is not for the purpose of replying to the PiC or MMC.

The DONE output indicates that the outgoing message has been successfully transferred to the SMTP server. It does not indicate successful reception of the message by its intended recipient.

The FAIL output is energized in the case of an incomplete transfer of the message to the SMTP server. If it is not energized, however, it does not necessarily indicate successful reception of the message by the intended recipient.

The ERR output specifies the type of error. Error codes are listed after E_UDPSVR in Table 2-1 Ethernet ASFB Error Codes. The ERR value of -1 could be caused by an incorrect NAMZ value or an incorrect SMTP Server configuration. The ERR value of -2 could be caused by an incorrect NAMZ value or the SMTP Server is offline.



Inputs: EN00 (BOOL) - enables function (continuous)

REQ (BOOL) - requests transfer (one-shot)

SLOT (USINT) - slot number of Ethernet module

NAMZ (STRING) - IP address or DNS name of TCP server, zero terminated

PORT (UINT) - TCP protocol port number. Choose any available TCP port above 1024

READ (BOOL) - read/write switch-read when enabled

CNT (UINT) - number of bytes to be read or written

BUFR (STRUCT) - data to exchange with TCP server

CLSE (BOOL) - close currently open socket (one-shot)

Outputs: DONE (BOOL) - execution complete without error

FAIL (BOOL) - error, execution incomplete

ERR (INT) - error number from IP functions that occurred during execution

CONN (BOOL) - established connection to server

```
<<INSTANCE NAME>>:E_TCPCL(EN00 := <<BOOL>>, REQ := <<BOOL>>,
  SLOT := <<USINT>>, NAMZ := <<STRING>>, PORT := <<UINT>>, READ
  := <<BOOL>>, CNT := <<UINT>>, BUFR := <<STRUCT>>, DONE =>
  <<BOOL>>, FAIL => <<BOOL>>, ERR => <<INT>>, CONN =>
  <<BOOL>>);
```

This function block is used to read or write data to a PiC or MMC using the Ethernet module. For PiC to PiC, the other PiC or MMC will have the E_TCPSVR function block (the server for this client). TCP is an Ethernet protocol that provides guaranteed delivery and error checking.

This function block implements the TCP Client connection described in the PiCPro Function/Function Block Reference Guide "Creating a TCP Client", located after the IPWRITE function block description.

This function block acts only as a client. It will solicit information from an external server (with E_TCPSVR) but will not service incoming requests from other controls or PCs.

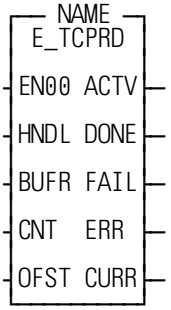
The inputs at PORT, CNT, and BUFR must be the same in both the TCP client and the TCP server it is to communicate with.

When the REQ input is energized this function block will create a socket on the protocol port specified by PORT using the Ethernet board in the slot specified by SLOT. Once a socket is created, the function block will initiate a read or a write, depending on the status of the READ input. If the READ input is set, the client will get the data contained in the BUFR input of the server with which the client is communicating and place that data in the structure at BUFR on the client side. If the READ input is not set, the client will put the data from its BUFR input into the structure at BUFR on the server side.

If the DONE output is energized, the client was successful in connecting to the server and transferring the data.

The FAIL output is set if the function block was not successful in connecting to the server and transferring the data.

The ERR output specifies the type of error. Error codes are listed after E_UDPSVR in Table 2-1 Ethernet ASFB Error Codes.



Inputs: EN00 (BOOL) - enables function

HNDL (UINT) - socket handle from IPSOCK

BUFR (Array of BYTE) - byte data area

CNT (UINT) - number of bytes to read

OFST (UINT) - byte offset into data message to read

Outputs: ACTV (BOOL) - execution complete without error

DONE (BOOL) - error, execution incomplete

FAIL (BOOL) - error number from IP functions that

ERR (INT) - 0 if read is successful, non 0 if read failed

CURR (UINT) - number of bytes actually read

<<INSTANCE NAME>>:E_TCPRD(EN00 := <<BOOL>>, HNDL := <<UINT>>, BUFR := <<BYTE>>, CNT := <<UINT>>, OFST := <<UINT>>, ACTV => <<BOOL>>, DONE => <<BOOL>>, FAIL => <<BOOL>>, ERR => <<INT>>, CURR => <<UINT>>);

This function block is used to read data from a TCP connection via the Ethernet module.

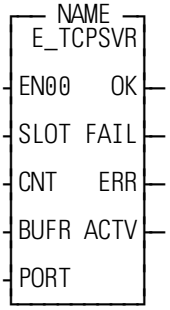
Since TCP is based on data streaming, not all of the data to be read may be available at any one time. The function block provides a convenient method of reading the stream data until the requested number of bytes are read.

When the EN00 input is energized, the function block will read data from a TCP socket defined by the HNDL input. After a successful read, if the actual number of bytes read is less than the requested number of bytes to read, the function will initiate another read. While the read(s) is active the ACTV output will be energized. When all the requested data has been read the DONE output will energize and the ACTV output will de-energize.

The EN00 input must remain energized until a DONE or FAIL is reported after which the EN00 input can be de-energized. To initiate another byte count read request the EN00 input must see a low to high transition.

The DONE output is set if the number of actual bytes read at the CURR output equals the number of requested bytes to read at the CNT input and indicates a successful read. The FAIL output is set if a read was unsuccessful.

The ERR output specifies the type of error and are listed in Table 2-1 Ethernet ASFB Error Codes.



Inputs: EN00 (BOOL) - enables function (continuous)

SLOT (USINT) - slot number of Ethernet module

CNT (UINT) - number of bytes in BUFR

BUFR (STRUCT) - data structure to be written from/read to

PORT (UINT) - TCP protocol port number

Outputs: OK (BOOL) - execution without error

FAIL (BOOL) - an error has occurred

ERR (INT) - error number from IP functions that occurred during execution

ACTV (BOOL) - currently reading or writing to input at BUFR

<<INSTANCE NAME>>:E_TCPSVR(EN00 := <<BOOL>>, SLOT := <<USINT>>, CNT := <<UINT>>, BUFR := <<STRUCT>>, PORT := <<UINT>>, OK => <<BOOL>>, FAIL => <<BOOL>>, ERR => <<INT>>, ACTV => <<BOOL>>);

This function block allows data to be read from or written to the PiC or MMC via the Ethernet module using TCP. For PiC to PiC, the other PiC or MMC will have the E_TCPCL function block (the client for this server).

This function block implements the TCP Server connection described in the PiCPro Function/Function Block Reference Guide “Creating a TCP Server" located after the IPWRITE function block description.

E_TCPSVR allows an external TCP client (with E_TCPCL) to read or write the number of bytes specified at the CNT input from or to (respectively) the structure specified at BUFR, provided the client and the server have identical inputs at CNT and BUFR.

TCP provides much greater confirmation of receipt than does UDP. If a packet is sent and not acknowledged within a certain amount of time, the packet will be resent. In addition, TCP allows for the streaming of data. Due to the addition of a sequencing number, packets are guaranteed to be reassembled in the order in which they were sent. This allows for messages larger than one packet size to be sent. The maximum amount of data that may be transferred at one time using TCP is 32,767 bytes (32KB).

This function block will service incoming TCP requests but will not send unsolicited information to other controls.

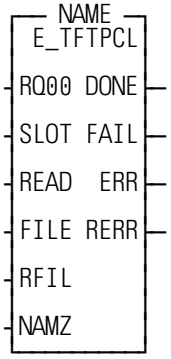
When the EN00 input is energized, the function block creates a TCP socket on the protocol port defined in PORT on the Ethernet board designated in SLOT. PORT is chosen as any available protocol port greater than 1024. Both the client and the server must have the same PORT number.

The OK output is set if the function block was successful in opening and configuring the port.

The FAIL output is set if the function block was not successful in opening and configuring the port.

The ERR output specifies the type of error. Error codes are listed after E_UDPSVR in Table 2-1 Ethernet ASFB Error Codes.

The ACTV output, when energized, indicates that the data at BUFR is currently in use by the function block and should not be read or written at this time.



Inputs: RQ00 (BOOL) - requests transfer (one-shot)

SLOT (USINT) - slot number of Ethernet module

READ (BOOL) - read/write switch - read when enabled

FILE (STRING) - filename in RAMDISK to read from/write to (Example: RAMDISK:\FILE.DAT)

RFIL (STRING) - remote file to read/write (Example: if on a PC: C:\DATA.DAT, if on a PiC or MMC: RAMDISK:\FILE.DAT)

NAMZ (STRING) - IP address of TFTP Server zero terminated

Outputs: DONE (BOOL) - execution complete without error

FAIL (BOOL) - error, execution incomplete

ERR (INT) - error number from IP functions that occurred during execution

RERR (INT) - error number from RAMDISK functions that occurred during execution

<<INSTANCE NAME>>:E_TFTPCL(RQ00 := <<BOOL>>, SLOT := <<USINT>>, CNT := <<UINT>>, READ := <<BOOL>>, FILE := <<STRING>>, RFIL := <<STRING>>, NAMZ := <<STRING>>, DONE => <<BOOL>>, FAIL => <<BOOL>>, ERR => <<INT>>, RERR => <<INT>>);

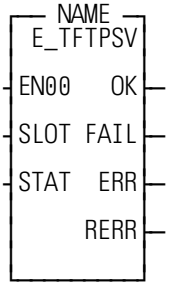
This function block allows a PiC or MMC to transfer files to and from a remote TFTP host. It allows a PiC or MMC to transfer files from its RAMDISK to a foreign host (another control (such as a PiC or MMC with E_TFTPSV) or a PC) and from a foreign host to the RAMDISK of the PiC or MMC in which the ASFB and Ethernet module are installed.

When the RQ00 input is energized, this function block will initiate a TFTP read or write, depending on the status of the READ input. If the READ input is set, the client will get the file specified at the RFIL input and place that file in the RAMDISK file specified at FILE. If the READ input is not set, the client will copy the RAMDISK file specified at FILE input into the file specified at RFIL.

If the DONE output is energized, the client was successful in connecting to the server and transferring the data.

The FAIL output is energized in the case of an IP or RAMDISK error, with the specific error being designated at the ERR and RERR outputs, respectively.

The ERR output specifies the type of error. Error codes are listed after E_UDPSVR in Table 2-1 Ethernet ASFB Error Codes. Error codes for the RERR output can be found in Appendix B of the PiCPro Software Manual.



Inputs: EN00 (BOOL) - enables execution (continuous)

SLOT (USINT) - slot number of Ethernet module

STAT (STRUCT) - status of current file transfer

Outputs: OK (BOOL) - execution without error

FAIL (BOOL) - an error has occurred

ERR (INT) - error number from IP functions that occurred during execution

RERR (INT) - error number from RAMDISK functions that occurred during execution

<<INSTANCE NAME>>:E_TFTPSV(EN00 := <<BOOL>>, SLOT := <<USINT>>, STAT := <<STRUCT>>, OK => <<BOOL>>, FAIL => <<BOOL>>, ERR => <<INT>>, RERR => <<INT>>);

This function block allows a PiC or MMC to service TFTP file transfer requests for a file on the RAMDISK.

When enabled, the E_TFTPSV function block allows a remote TFTP client to read and write files to and from the RAMDISK of a PiC or MMC in which the ASFB and Ethernet module are installed. The remote TFTP client could be either a PiC or MMC with the E_TFTPCL function block or a PC with a TFTP client.

When the EN00 input is energized, the function block opens a UDP server socket on port 69 (this is the accepted TFTP port). Incoming TFTP requests are serviced one-at-a-time, with new requests being ignored if made before the completion of a previously requested transfer. If multiple simultaneous transfers are needed, the function block must be edited to accommodate this; the function block only serves one request at a time.

If a remote TFTP client attempts to write to a file that already exists on the RAMDISK of the PiC or MMC with the E_TFTPSV function block, the existing file will be erased and replaced by the new file.

The STAT structure shows the status of the current file transfer. The structure format is as follows:

STAT STRUCT		Status of the current file transfer
.Active	BOOL	Indicates that a file transfer is active
.ReadWrite	BOOL	0 = file being read, 1 = file being writtten
.ClientIPAddr	STRING[25]	IP Address of the Client accessing the the file
.FileName	STRING[80}	Name of file being accessed
.CheckSum	INT	Checksum, used to verify the size of the Structure, set the Initial Value to 12345
	END_STRUCT	

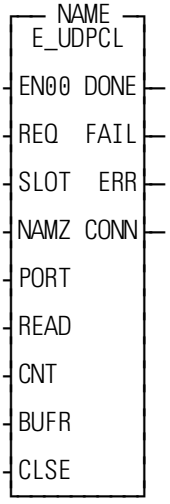
IMPORTANT

The last data variable CheckSum must be included in the structure with the initial value set to 12345. This memory location with a known value is used by the ASFB to verify the size of the structure. If the structure is not the correct size, an error will be reported upon initialization.

The OK output is set if the function block is successful in opening and configuring a UDP socket and reset in the event of a failure.

The FAIL output is energized in the case of an IP or RAMDISK access error, with the specific error being designated at the ERR and RERR outputs, respectively.

The ERR output specifies the type of error. Error codes are listed after E_UDPSVR in Table 2-1 Ethernet ASFB Error Codes. Error codes for the RERR output can be found in Appendix B of the PiCPro Software Manual.



Inputs: EN00 (BOOL) - enables function (continuous)

REQ (BOOL) - requests transfer (one-shot)

SLOT (USINT) - slot number of Ethernet module

NAMZ (STRING) - IP address or DNS name of UDP server, zero terminated

PORT (UINT) - UDP protocol port number.

READ (BOOL) - read/write switch - read when enabled

CNT (UINT) - number of bytes to be read or written

BUFR (STRUCT) - data to exchange with UDP server

CLSE (BOOL) - closes socket (one-shot)

Outputs: DONE (BOOL) - execution complete without error

FAIL (BOOL) - error, execution incomplete

ERR (INT) - error number from IP functions that occurred during execution

CONN (BOOL) - socket open (connection is active)

<<INSTANCE NAME>>:E_UDPCL(EN00 := <<BOOL>>, REQ := <<BOOL>>, SLOT := <<USINT>>, NAMZ := <<STRING>>, PORT := <<UINT>>, READ := <<BOOL>>, CNT := <<UINT>>, BUFR := <<STRUCT>>, CLSE := <<BOOL>>, DONE => <<BOOL>>, FAIL => <<BOOL>>, ERR => <<INT>>, CONN => <<BOOL>>);

This function block allows data to be read from or written to a PiC or MMC via the Ethernet module using UDP. For PiC to PiC, the other PiC or MMC will have the E_UDPSVR function block (the server for this client).

This function block implements the UDP Client connection described in the PiCPro Function/Function Block Reference Guide "Creating a UDP Client (Connectionless)", located after the IPWRITE function block description.

This function block acts only as a client. It will solicit information from server function blocks but will not service incoming requests from other controls or PCs.

The inputs at PORT, CNT, and BUFR must be the same in both the UDP client and the UDP server it is to communicate with.

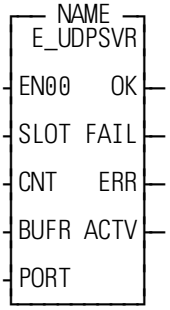
Due to UDP packet size, up to 512 bytes of data may be transferred per use of this function block. As a result, CNT must be less than or equal to 512. If any of the TCP benefits are required or if more than 512 bytes must be transferred per message, use the E_TCPSVR (for the server) and E_TCPCL (for the client) function blocks.

When the REQ input is energized this function block will create a socket on the protocol port specified by PORT using the Ethernet board in the slot specified by SLOT. Once a socket is created, the function block will initiate a read or a write, depending on the status of the READ input. If the READ input is set, the client will get the data contained in the BUFR input of the server with which the client is communicating and place that data in the structure at BUFR on the client side. If the READ input is not set, the client will put the data from its BUFR input into the structure at BUFR on the server side.

If the DONE output is energized, the client was successful in connecting to the server and transferring the data.

The FAIL output is set if the function block was not successful in connecting to the server and transferring the data.

The ERR output specifies the type of error. Error codes are listed after E_UDPSVR in Table 2-1 Ehthernet ASFB Error Codes.



Inputs: EN00 (BOOL) - enables function (continuous)

SLOT (USINT) - slot number of Ethernet module

CNT (UINT) - number of bytes to be read or written

BUFR (STRUCT) - data to be read/space to be written to

PORT (UINT) - UDP protocol port number. Choose any available UDP port above 1024.

Outputs: OK (BOOL) - execution without error

FAIL (BOOL) - an error has occurred

ERR (INT) - error number from IP functions that occurred during execution

ACTV (BOOL) - currently reading or writing to input at BUFR

<<INSTANCE NAME>>:E_UDPSVR(EN00 := <<BOOL>>, SLOT := <<USINT>>, CNT := <<UINT>>, BUFR := <<STRUCT>>, PORT := <<UINT>>, OK => <<BOOL>>, FAIL => <<BOOL>>, ERR => <<INT>>, ACTV => <<BOOL>>);

This function block allows data to be read from or written to the PiC or MMC via UDP using the Ethernet module. For PiC to PiC, the other PiC or MMC will have the E_UDPCL function block (the client for this server).

This function block implements the UDP Server connection described in the PiCPro Function/Function Block Reference Guide "Creating a UDP Server (Connectionless)", located after the IPWRITE function block description.

E_UDPSVR allows an external UDP client (with E_UDPCL) to read or write the number of bytes specified at the CNT input from or to (respectively) the structure specified at BUFR, provided the client and the server have identical inputs at CNT and BUFR.

Due to UDP packet size, up to 512 bytes of data may be transferred per use of this function block. As a result, CNT must be less than or equal to 512. If any of the TCP benefits are required or if more than 512 bytes must be transferred per message, use the E_TCPSVR (for the server) and E_TCPCL (for the client) function blocks.

This function block will service incoming UDP requests but will not send unsolicited information to other controls.

When the EN00 input is energized, the function block creates a UDP socket on the protocol port defined in PORT on the Ethernet board designated in SLOT. PORT is chosen as any available protocol port greater than 1024. Both the client and the server must have the same PORT number.

The OK output is set if the function block was successful in opening and configuring the port.

The FAIL output is set if the function block was not successful in opening and configuring the port.

The ERR output specifies the type of error. Error codes are listed in Table 2-1 Ethernet ASFB Error Codes.

The ACTV output, when energized, indicates that the data at BUFR is currently in use by the function block and should not be read or written at this time.

Table 2-1. Ethernet ASFB Error Codes

ERR#	Description	ERR#	Description
0	No error	**40**	Destination address required
1	Not owner	**41**	Protocol wrong type for socket
2	No such file or directory	**42**	Protocol not available
3	No such process	**43**	Protocol not supported
4	Interrupted system call	**44**	Socket type not supported
5	I/O error	**45**	Operation not supported on socket
6	No such device or address	**46**	Protocol family not supported
7	Arg list too long	**47**	Address family not supported
8	Exec format error	**48**	Address already in use
9	Bad file number	**49**	Can’t assign requested address
10	No children	**50**	Socket operation on non-socket
11	No more processes	**51**	Network is unreachable
12	Not enough core	**52**	Network dropped connection on reset
13	Permission denied	**53**	Software caused connection abort
14	Bad address	**54**	Connection reset by peer
15	Directory not empty	**55**	No buffer space available
16	Mount device busy	**56**	Socket is already connected
17	File exists	**57**	Socket is not connected
18	Cross-device link	**58**	Can’t send after socket shutdown
19	No such device	**59**	Too many references: can’t splice
20	Not a directory	**60**	Connection timed out
21	Is a directory	**61**	Connection refused
22	Invalid argument	**62**	Network is down

23	File table overflow	**63**	Text file busy
24	Too many files open	**64**	Too many levels of symbolic links
25	Not a typewriter	**65**	No route to host
26	File name too long	**66**	Block device required
27	File too large	**67**	Host is down
28	No space left on device	**68**	Operation now in progress
29	Illegal seek	**69**	Operation already in progress
30	Read-only file system	**70**	Operation would block
31	Too many links	**71**	Function not implemented
32	Broken pipe	**72**	Operation cancelled
33	Resource deadlock avoided	**1000**	There is a non-zero terminated string which requires zero termination or a zero length string.
34	No locks available	**1001**	There is a CNT input which is too large.
35	Unsupported value	**1002**	The SLOT number requested does not contain an Ethernet board.
36	Message size	**1003**	Either the firmware does not support TCP/IP or there is no Ethernet board in the rack.
37	Argument too large	**1004**	The IPZ buffer is too small.
38	Result too large	**1005**	A TCP/IP function was terminated due to a TCP/IP stack failure. The socket the function block is using is no longer valid. *

Table 2-2. Ethernet ASFB Error Codes

ERR#	Description	ERR#	Description
-2	No response from SMTP server	38	Result too large
-1	Client/server settings do not match	39	Error not defined
0	No error	40	Destination address required
1	Not owner	41	Protocol wrong type for socket
2	No such file or directory	42	Protocol not available
3	No such process	43	Protocol not supported
4	Interrupted system call	44	Socket type not supported
5	I/O error	45	Operation not supported on socket
6	No such device or address	46	Protocol family not supported
7	Arg list too long	47	Address family not supported
8	Exec format error	48	Address already in use
9	Bad file number	49	Can not assign requested address
10	No children	50	Socket operation on non-socket
11	No more processes	51	Network is unreachable
12	Not enough core	52	Network dropped connection on reset
13	Permission denied	53	Software caused connection abort
14	Bad address	54	Connection reset by peer
15	Directory not empty	55	No buffer space available
16	Mount device busy	56	Socket is already connected
17	File exists	57	Socket is not connected
18	Cross-device link	58	Can not send after socket shutdown
19	No such device	59	Too many devices; can not splice
20	Not a directory	60	Connection timed out
21	Is a directory	61	Connection refused
22	Invalid argument	62	Network is down
23	File table overflow	63	Text file busy
24	Too many files open	64	Too many levels of symbolic link
25	Not a typewriter	65	No route to host
26	File name too long	66	Block device required
27	File too large	67	Host is down
28	No space left on device	68	Operation now in progress
29	Illegal seek	69	Operation already in progress
30	Read-only file system	70	Operation would block
31	Too many links	71	Function not implemented
32	Broken pipe	72	Operation cancelled
33	Resource deadlock avoided	1000	String requires zero-termination
34	No locks available	1001	CNT input too large
35	Unsupported value	1002	SLOT number requested does not contain and Ethernet module
36	Message size	1003	Firmware does not support TCP/IP or no Ethernet module in rack
37	Argument too large	1004	IPZ buffer too small

NOTES

Index

NOTES