APPLICATION SPECIFIC FUNCTION BLOCK MANUAL

Data Highway Plus™

GIDDINGS & LEWIS®

NOTE

Progress is an ongoing commitment at Giddings & Lewis. 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. Giddings & Lewis 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 Giddings & Lewis 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. Giddings & Lewis 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, Giddings & Lewis, 660 South Military Road, P.O. Box 1658, Fond du Lac, WI 54936-1658. Giddings & Lewis can be reached by telephone at (920) 921-7100.

401-55376-00

Version 0498

DB2-3489

BA22-4489

Table of Contents

NOTES

Application Specific Function Block Guidelines

Installation

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

1. Make a back up copy of the ASFB disk you receive and store the original in a safe place.
2. The disk you receive with the ASFB package will include the following:
   1. ASFBS directory containing:
      - .LIB file(s) containing the ASFB(s)
      - source .LDO(s) from which the ASFB(s) was made
   2. EXAMPLES directory containing:
      - 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

   It is recommended that you copy the .LIB and the source LDO files to your hard drive on the PC in the following way. Remember that ASFB libraries (.LIB) files and source (.LDO) files must be kept in the same directory.

   - Create a directory that will hold all ASFB LIBs and source LDOs. For example, you may have the Motion ASFB package and the Communication ASFB package. Copy the appropriate files on the disks to a directory on your PC called ASFB.

     When *you* installed PiCPro, the PiCLib statement was automatically entered in your autoexec.bat file as shown below:

     SET PICLIB=C:\PICLIB

     NOTE: If you chose to alter your PICLIB statement during installation, it will look different than what appears above.

     Now add the ASFB directory to your PICLIB = statement as shown below:

     SET PICLIB=C:\PICLIB;C:\ASFB

   - Put the example file(s) in your working directory.
     For example, if you always run PiCPro from the directory which holds all your LDO files, then copy all the ASFB example LDOs to the LDO directory.

Revisions

3. The first three networks of each ASFB source ladder provide the following information.

Network 1

The first 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 1
|...1.....
X-Name ASFB Source Revision History

Located in Library X-LIB

Requirements:
PiCPro Ver 4.0 or higher

Date	Version	Using PiCPro	Comments
MM-DD-YY	EN00	4.1	Original

Network 2

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

If you revise the ASFB, do the following:

1. Do a 'M'odule, save 'A's in order to save the original ASFB before you begin modifying.
2. Change the number on the first input to the ASFB in the software declarations table to a 50 or greater (for example, ENOO would be changed to EN50).
3. Update the revision history in network 1.

ASFB Input/Output Descriptions

Network 3

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

ASFB Description

INPUTS:

Name	Data Type	Definition
ENOO	BOOL	enables execution

OUTPUTS:

Name	Data Type	Definition
OK	BOOL	execution complete

4. 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.

     NOTE: To keep the original example LDO, use the 'save As' command. This copies the example LDO to an LDO with the application name you give it.

   - If you already have an application LDO, merge the example LDO with the application LDO using the optional LDOMERGE software package. The software declaration tables for both LDOs will also merge.

   - Enter the ASFB into your application LDO.

     NOTE: This method is not recommended if the software declarations table is lengthy. It requires that you manually enter all the inputs and outputs to the ASFB in the table. With some packages, this is time-consuming. Any structure, array, array of structures, or strings must be entered *exactly* as it appears in the original table. This is critical to the correct functioning of the ASFB.

Data Highway Plus ASFBs

1.1 Introduction

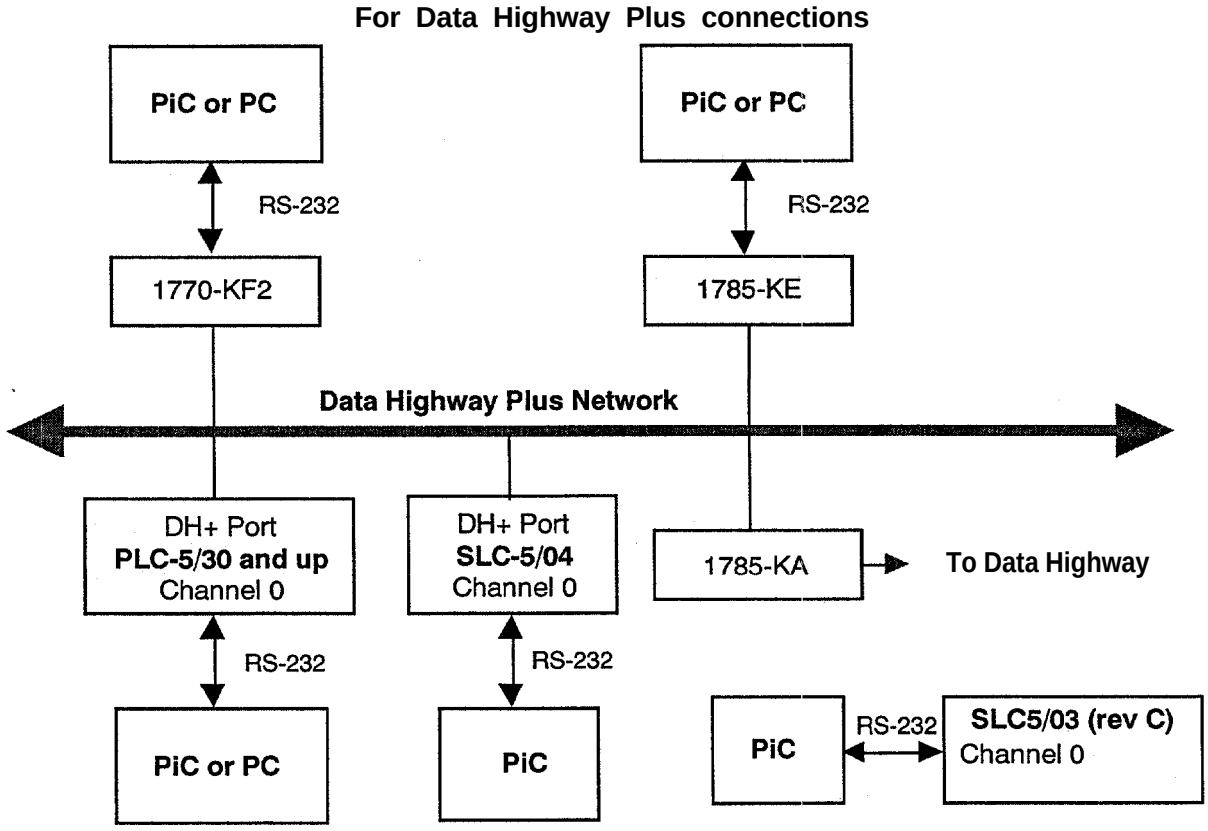
The Data Highway Plus ASFB software package from Giddings & Lewis allows the PiC to communicate to any compatible device using the Allen-Bradley DF1 full duplex serial protocol. Communication takes place through the PiC serial ports. The serial ports include the User Port on the CPU or the ports on a PiC serial communications module (2 or 4 port models available).

Compatible devices include the following as illustrated in Figure 1:

- Allen -Bradley PLC5/30 or higher
- Allen -Bradley SLC 5/03 revision 'C' or higher*
- Allen -Bradley SLC5/04*
- Allen -Bradley 1770-KF2 module
- Allen -Bradley 1785-KE module
- Any software program supporting the A-B DF1 Full Duplex serial protocol

*See the NOTE on the following page.

Figure 1. Configurations for using Data Highway Plus communications



The interface devices shown in the diagram are described below.

1770-KF2 Stand alone communication interface module that connects an RS-232 device to the Data Highway Plus or Data Highway.

1785-KE Rack mounted communications interface module that connects an RS-232 device to Data Highway Plus.

1785-KA Communication interface device that connects the Data Highway Plus with the Data Highway.

When a PiC is connected to a 1770-KF2 or 1785-KE module, the PiC can communicate over a Data Highway Plus Network. When a PiC is connected to the Channel 0 port of a PLC-5/30 or higher, the PiC can communicate with the data table area of that PLC, but not to any other stations that the PLC may be connected to on the Data Highway Plus network.

NOTE

The terms *slave mode* and *master/slave mode* are used throughout this document to describe which station is responsible for what action. These terms should not be confused with the master and slave terms used by Allen-Bradley when referring to the Half Duplex DF1 protocol. This package implements the Full Duplex DF1 protocol and is incompatible with the Half Duplex version.

When communicating with the DF1 protocol, the PiC can operate in either a slave mode or a master/slave mode. In the slave mode, the PiC will only receive commands from the other device. It will not initiate any transfers to other devices. In the master/slave mode, the PiC can initiate transfers to other devices as well as receive commands from them.

NOTE

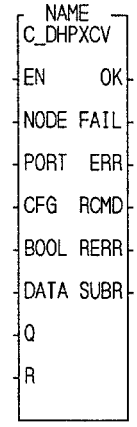
When the PiC is connected to a SLC-5, it can only operate in the slave mode .

The Data Highway Plus software includes two ASFBs that you install in PiCPro and use in your application ladder. They are nhnwn below.

C_DHPXCV

Communications_Data Highway Plus Transceiver function block

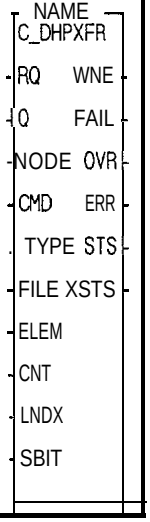
Establishes communications to a compatible device. Required for the slave mode and the master/slave mode.



C_DHPXFR

Communications-Data Highway Plus Transfer Request function block

Initiates, in conjunction with C_DHPXCV, a request to another station. Required for the master/slave mode.



1.2 Requirements

The hardware and software requirements when using the Data Highway Plus interface to communicate with a compatible device are covered in this section.

Hardware requirements

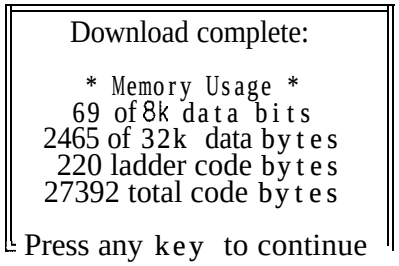
- A PiC programmable industrial computer with approximately 4K of data bytes free and approximately 25K of ladder code bytes free.

NOTE: The number of free bytes can be checked in the Download complete box in PiCPro. The box appears after you download a ladder. An example is shown on the right.

To calculate the number of free bytes, subtract the total code bytes from your processor memory capacity.

Example: If your processor memory is 64K (64 * 1024 = 65536), then

65536 - 27392 = 38144 or 37.2 K of free bytes



- A serial port (either User Port on the PiC CPU or one of four serial ports on a serial communications module.)
- A serial cable to connect the PiC to the remote device.

Cable connections

The pinouts for the various Data Highway Plus communications connections are shown below. Choose the one for your system.

PiC to a 1770-KF2

PiC CPU (1 O-pin screw terminal)			to a	1770-KF2 (25-pin female)	
GND	8	———		7	GND
RECV	9	———		2	TRANS
TRANS	10	———		3	RECV

PiC to a PC

PiC CPU (1 O-pin screw terminal)			to a	PC (9-pin female)	
GND	8	———		5	GND
RECV	9	———		3	TRANS
TRANS	10	———		2	RECV

PiC to a PLC-5/30 or higher

PiC CPU (1 O-pin screw terminal)			to a PLC-5/30 or up Channel 0 (25-pin male)		
GND	8	———		7	GND
RECV	9	———		2	TRANS
TRANS	10	———		3	RECV

PiC to a SLC-5/03 or 5/04

PiC CPU (1 O-pin screw terminal)			to aSLC-5/03 or 5/04 Channel 0 (9-pin female)		
GND	8	———		5	GND
RECV	9	———		3	TRANS
TRANS	10	———		2	RECV

1770-KF2 to a PC

1770-KF2 (25-pin female)			to a	PC (9-pin female)	
TRANS	2	———		2	RECV
RECV	3	———		3	TRANS
GND	7	———		5	GND
DSR	6	—┐ *		* ┌—	4 DTR
DCD	8	—┤		├—	6 DSR
DTR	20	—┘		└—	1 DCD
RTS	4	—┐ *		* ┌—	7 RTS
CTS	5	—┘		└—	8 CTS

GD02-4892

*These jumpers may not be necessary. Check your manual.

Software requirements

- Data Highway Plus ASFB software
- PiCPro Version 4.1 or higher
- LDOMERGE software (Optional software that allows you to merge ladders.)

PLC compatibility

The Giddings & Lewis implementation of the DF1 protocol requires the following configuration settings:

DF1 Full Duplex Serial
BCC Error Detect
Logical Binary Addressing Mode
No Embedded Responses
PLC5 Type Messages

The PiC supports the following PLC-5 Data Highway Plus commands:

Command Name	Command	Function
Word Range Read	0F	01
Word Range Write	0F	00
Read-Modify-Write	0F	26
*Typed Read	0F	68
*Typed Write	0F	67

*These commands are only supported in the slave (incoming) mode.

The device the PiC is communicating with must also support these commands. If the PiC receives a command it does not support or recognize, it will return an error response to the sender, specifically, the error code 10 hex (Illegal Command or Format).

For information, on Data Highway error codes, see Allen-Bradley's "Data Highway/Data Highway Plus Protocol and Command Set Reference Manual" Publication 1770-6.5.16 - November, 1987 or later.

1.3 Installation

The Data Highway Plus interface disk contains the files listed below. The Main group includes the ASFB library (LIB), source ladders for the ASFBs (LDOs), and remark files containing the comments in the source ladders (.REMs). The Example group includes the example LDO and REM files. The Auxiliary group contains the LIB, LDOs, and REMs for the UDFBs used in the source ladders for the ASFBs.
NOTE: It should never be necessary for you to access any of the files in the Auxiliary group. The LIB is required in order for the ASFB to work and the LDOs allow you to view the source ladders when troubleshooting if necessary.

Follow the guidelines found at the beginning of the manual. Always make a back up copy of the disk and store the original in a safe place. The recommended destination directory for each file is listed in the last column.

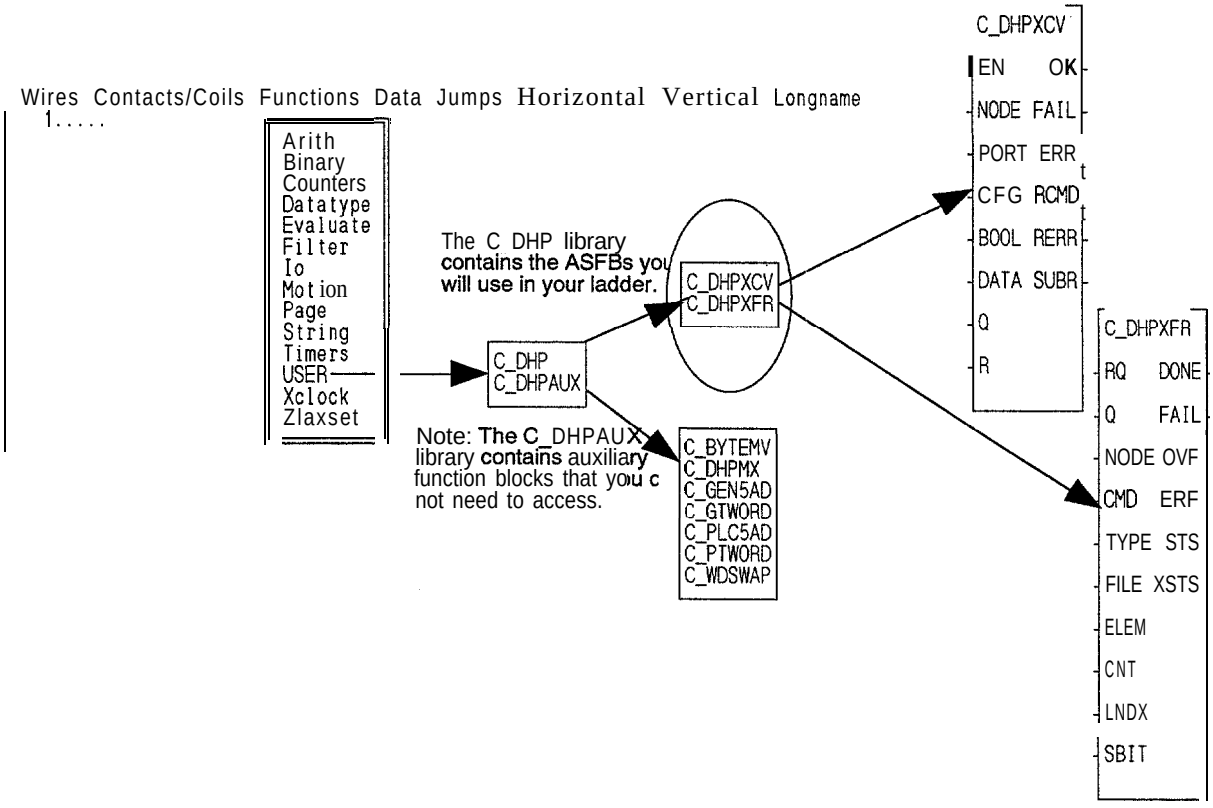
Group	File	Description	Directory
Main	C_DHP.LIB	The library containing the two application specific function blocks used to perform data highway plus communications.	ASFB
	C_DHPXCV.LDO	The source ladder for the transceiver function block.	ASFB
	C_DHPXCV.REM	The remark file for the source ladder	ASFB
	C_DHPXFR.LDO	The source ladder for the transfer request function block.	ASFB
	C_DHPXFR.REM	The remark file for the source ladder	ASFB
Example	C_DHPEX.LDO	The example LDO from which you can build a new application LDO or to which you can merge an existing one.	Working
	C_DHPEX.REM		Working
Auxiliary	C_COMMON.LIB	The Library of common communication functions	ASFB
	C_DHPAUX.LIB	The library that holds all the function blocks used in the source ladder for the ASFB.	ASFB
	C_DHPMX.LDO	Source ladder	ASFB
	C_DHPMX.REM	Remark file	ASFB
	C_GEN5AD.LDO	Source ladder	ASFB
	C_GEN5AD.REM	Remark file	ASFB
	C_GTWORD.LDO	Source ladder	ASFB
	C_GTWORD.REM	Remark file	ASFB
	C_PLC5AD.LDO	Source ladder	ASFB
	C_PLC5AD.REM	Remark file	ASFB
	C_PTWORD.LDO	Source ladder	ASFB
	C_PTWORD.REM	Remark file	ASFB
	C_WDSWAP.LDO	Source ladder	ASFB
	C_WDSWAP.REM	Remark file	ASFB
	C_BYTEMV.LDO	Source ladder	ASFB
	C_BYTEMV.REM	Remark file	ASFB

NOTE: The libraries containing the ASFBs and their source ladders must always be in the same directory.

1.4 Data Highway Plus Function Blocks

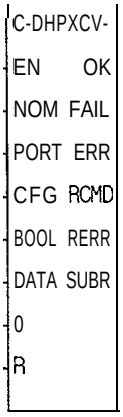
The function blocks for the Data Highway Plus interface are described in this section. When PiCPro is running, you can find the Data Highway Plus function blocks by choosing the Function menu, then USER, then C_DHP as shown in Figure 2.

Figure 2. Location of ASFBs in PiCPro



C_DHPXCV
USER/C_DHP

Communi-cations-data highway plus transceiver



Inputs:
EN (BOOL) - enables execution
NODE (USINT) - identifies this station number
PORT (STRING) - identifies the communication serial port
CFG (STRING) - configuration string for the port
BOOL (ARRAY) - boolean data area
DATA (STRUCT) - variable data area
Q (ARRAY OF STRUCT) - Request transfer message queue
R (STRUCT) - received message information

outputs:
OK (BOOL) - execution completed without error
FAIL (BOOL) - transceiver initialization failed
ERR (INT) - 0 if initialization is successful; 0 if initialization is unsuccessful
RCMD (BOOL) - energized if a message is received
RERR (USINT) - error code for last message received
SUBR (USINT) - sub error code for last message

The C_DHPXCV function block provides PiC communication capabilities for both the slave and the master/slave modes to a Data Highway Plus compatible device. The link to the device must be made through one of the PiC serial ports.

When this function is enabled, it will open the PiC serial port specified at the PORT input. This port will be configured based on the information specified at the CFG input. If the port configures properly the OK output will energize and the system will be ready to talk to the Data Highway Plus device. If a problem occurs in the open or configuration process, the FAIL output will be energized and the OK will not be set. See Appendix B in the PiCPro Software Manual for the error codes at the ERR output.

To establish communications on the Data Highway Plus network, this function block is needed only once and should be enabled every scan.

Inputs

EN The EN input is energized every scan to make a request over the Data Highway Plus network. In a typical system, this input will be wired to the vertical or power bus rail.

NOTE: De-energizing this input will cause communication to stop for this station identified at the NODE input.

NODE The NODE input specifies the station number this control will be. The number you enter must match any Data Highway Plus interface equipment this control is talking to. For example, if the PiC User Port is connected to a KF2 module that is addressed as station number 4, then 4 is entered at the NODE input.

The range of numbers that this input will accept is 0 to 255 (decimal).

IMPORTANT

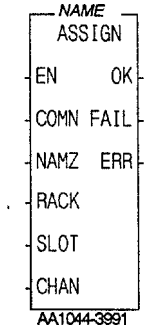
The station numbers for the Data Highway Plus network are typically in octal. The NODE input is expecting a decimal value. It is necessary for you to convert the station number from octal to decimal before entering it.

PORT The PORT input specifies which serial port this function block will use to communicate over. Place a string type variable at this input that has been initialized with the name of the port that is to be used.

For example, if the PiC User Port is being used, initialize a string as:

USER:$OO

If one of the channels on the serial communications module is being used, the name you enter at the NAMZ input of the ASSIGN function block is the name you enter in the string at the PORT input of the CDHPXCV function block.



BOOL

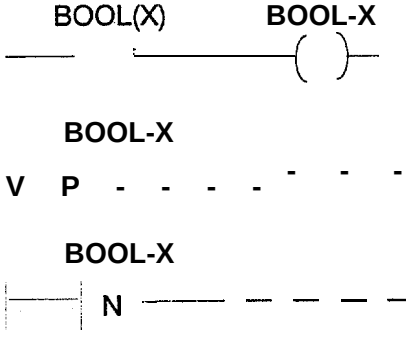
The BOOL input is an array that specifies the boolean (bit) data area that is used for any boolean (bit) transfers. The array size can range from two to 992 booleans. Choose the number of booleans in multiples of 16 (16, 32, 48, 64, . . . 992)

Although the boolean data type is not included in the DATA structure, it is necessary to place the number of booleans divided by 16 in the first field (BOOL_S) of the data structure. This is required since the BOOL array will be addressed by a remote node via an element number. Each element is 16 bits long.

IMPORTANT

Do not use a positive or negative transistional contact in your LDO with the BOOL array.

If it is necessary to set up a transistional contact with a BOOL array, use the BOOL array to energize another boolean coil. Then use this boolean for the transistional contact as shown in the example below.



DATA The DATA input is used to specify the name of the main data area. This data area is a structure with arrays that you define. It contains every data type available except booleans. The format for the structure is that for each data type that exists on the PiC, there are two fields of entry (only one for booleans).

IMPORTANT

If your application does not require a math co-processor and you are running PiCPro Version 7.0 or higher, an error message stating that a math co-processor is required will appear when downloading a program that contains this data structure. This message is generated because the data structure has data types that require a math co-processor, i.e. REAL, LWORD, ULINT, etc. You can choose to ignore this error and continue downloading.

OR

If you want to eliminate the message from appearing, you can change the following datatypes in the DATA structure.

Name	Existing Data Type	Change to:
.LWORD_D	**LWORD** (0..1)	**DINT (0..3)***
.ULINT_D	**ULINT** (0..1)	**DINT (0..3)***
.LINT_D	**LINT** (0..1)	**DINT (0..3)***
.REAL_D	REAL (0..1)	**DINT (0..1)**
.LREAL_D	**LREAL** (0..1)	**DINT (0..3)***

The data types that require a math co-processor are now eliminated from the program and, consequently, the error message will not appear. The DINT arrays you are replacing the existing data types with do not require a math co-processor.

*The array size for these DINTs is double the original size for the 64-bit variables. This insures that the memory map for the data structure remains the same.

The first member of the structure is a UINT called BOOL_S. This is initialized to the number of booleans entered at the BOOL array divided by 16.

For each additional data type available in the PiC, two fields are defined in this structure. The first field is initialized to the number of items that are defined in the corresponding data array. The second field is the array for that data type.

NOTE: It is very important that the value in size and the size of the array are the same. The size is user adjustable from 2 to 999 elements.

Declared structure with arrays for DATA input

USERDATA	STRUCT	
BOOL_S	UINT	2
BYTE_S	UINT	2
BYTE_D	BYTE(0..1)	
WORD_S	UINT	2
WORD_D	WORD(0..1)	
DWORD_S	UINT	2
DWORD_D	DWORD(0..1)	
LWORD_S	UINT	2
LWORD_D	LWORD(0..1)	
USINT_S	UINT	2
USINT_D	USINT(0..1)	
UINT_S	UINT	2
UINT_D	UINT(0..1)	
UDINT_S	UINT	2
UDINT_D	UDINT(0..1)	
ULINT_S	UINT	2
ULINT_D	ULINT(0..1)	
SINT_S	UINT	2
SINT_D	SINT(0..1)	
INT_S	UINT	2
INT_D	INT(0..1)	
DINT_S	UINT	2
DINT_D	DINT(0..1)	
LINT_S	UINT	2
LINT_D	LINT(0..1)	
REAL_S	UINT	2
REAL_D	REAL(0..1)	
LREAL_S	UINT	2
LREAL	LREAL(0..1)	
STRING_S	UINT	2
STRING_D	STR[82](0..1)	
DATE_S	UINT	2
DATE_D	DATE(0..1)	
DANDT_S	UINT	2
DANDT_D	D_AND_T(0..1)	
TOFD_S	UINT	2
TOFD_D	T_OF_D(0..1)	
TIME_S	UINT	2
TIME_D	TIME(0..1)	
	END-STRUCT	

Q

The Q array of structures specifies a request queue area that transfer requests can be held in until they are processed. Requests are placed in this queue by the C_DHPXFR function block and retrieved by the C_DHPXCV function block. The structure placed at this input must have the format shown below.

NOTE: This array of structures must be defined in your ladder as shown below. You must place it at the Q input of each Data Highway Plus function block used. Do not assign or modify any values in this array of structures. It is for inter-function communication only.

Declared array of structures for Q input

Q	STRUCT(0..10)
NODE	USINT
:PROT	USINT
.CMD	USINT
TYPE	USINT
:FILE	UINT
ELEM	UINT
:CNT	USINT
LNDX	UINT
:STAT	USINT
.STS	USINT
.EXT_STS	USINT
.SBIT	USINT
	END-STRUCT

R

The R structure specifies a data area that information about the last unsolicited message received from the other station is placed. When an unsolicited message is received, command data is placed in the data area specified by this input. The structure placed at this input must have the format shown below.

Declared structure for R input

R	STRUCT
.NODE	USINT
.CMD	USINT
.TYPE	USINT
ELEM	UINT
:CNT	USINT
	END-STRUCT

outputs

OK The OK output when energized indicates that the transceiver portion has been started and is ready for communication. If this output does not energize, check the FAIL output and the ERR output to identify the problem.

FAIL The FAIL output when energized indicates that the transceiver initialization failed. When this output is energized, the OK will not be energized and an error code will appear at the ERR output to identify the problem.

ERR The ERR output is 0 if initialization is successful and is 0 if initialization is unsuccessful. The error codes that appear at this output are system errors. See Appendix B in the PiCPro Software Manual for a description of each error.

RCMD The RCMD output when energized indicates the transceiver has received a command from a remote station. The information describing the nature of the request will be placed in the data structure placed at the R input of this function block.

RERR
SUBR The RERR and SUBR outputs hold any error codes generated while trying to process the last request received. These outputs can be checked each time the RCMD output energizes. If no error condition exists, these outputs will be zero. If an error condition does exist, the appropriate error code will be output. The list of error codes follows.

RERR Code	Description
0	No error.
1	A message was received with an unknown or unsupported function requested. Ensure that the device talking to the PiC is using one of the commands supported by the PiC. See the PLC compatibility section in this manual.
2	A bad address format was used in a Read-Modify-Write command. Ensure that the device talking to the PiC is using a four level PLC5 type address format. NOTE: If a PLC5 is sending requests to the PiC, the PLC5 should be configured to use the Type Write and Type Read commands.

SUBR	Description
3	PLC5 address level 4 is not zero.

3	An invalid FILE number was used in a Read-Modify-Write command. Only files 3 (bit, booleans), 7 (integers), and 8 (floating point) are valid file numbers.
4	An error occurred in the read portion of a Read-Modify-Write command.

SUBR	Description
1	Unrecognized data type.
2	Request exceeds array bounds.
3	Memory transfer overflow.

5	An error occurred in the write portion of a Read-Modify-Write command.

SUBR	Description
1	Unrecognized data type.
2	Request exceeds array bounds.
3	Memory transfer overflow.

6	The packet offset field of the command frame was not zero. This function does not support multi-packet transfers and, therefore, the packet offset must be zero on all commands. The total amount of data requested could not be transferred in one block so the remote device broke the data into multiple packets. This function does not support multi-packet transfers. Try reducing the amount of data transferred per request.
7	The total number of data bytes exceeded 239 for a Word Range Write command.
8	A bad address format was used in a Word-Range-Read/Write command.

SUBR	Description
3	PLC5 address level 4 is not zero.

9	An invalid file number was used in a Word-Range-Read/Write command.
10	An error occurred while reading or writing data from a received message into memory.

SUBR	Description
1	Unrecognized data type.

2	Request exceeds array bounds.
3	Memory transfer overflow.

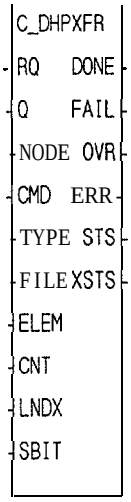
11	The total number of data bytes exceeded 244 bytes for a Word-Range-Read command.
12	The total number of data bytes exceeded 239 bytes for a Word-Range-Write command.
13	A bad address format was used in a Typed Write or Typed Read command.

SUBR	Description
3	PLC5 address level 4 is not zero.

14	Unsupported data type in a Typed Write command.
15	Unsupported data type in a Typed Read command.
16	Data type sent in Typed Write command does not match destination file data type.
17	Invalid file number in Typed Write command.
18	Invalid file number in Typed Read command.
21	Total number of data bytes exceeded 240 bytes for a Typed Write command.
22	Total number of data bytes exceeded 240 bytes for a Typed Read command.

C_DHPXFR
USER/C_DHP

Communi-cation-data highway plus transfer request



Inputs:

RQ (BOOL) - enables execution

Q (USINT) - request transfer message queue

NODE (USINT) - number of destination station

CMD (USINT) - command 0 = read, 1 = write, 2 = read, write, modify

TYPE (USINT) - PiC data type

FILE (UINT) - remote station file number

ELEM (UINT) - remote station element number

CNT (USINT) - number of elements in transfer

LNDX (UINT) - local PiC array index in array defined by TYPE input

SBIT (USINT) - starting bit number of a read-modify-write command

outputs:

DONE (BOOL) - energized if transfer is successful

FAIL (BOOL) - energized if transfer is unsuccessful

OVR (BOOL) - energized if more than 10 pending requests are made overflowing the queue

ERR (USINT) - error code if FAIL output is energized

STS (USINT) - error status returned from remote station

XSTS (USINT) - extended error status returned from remote station

The C_DHPXFR function block is used to initiate a read, write, or read-modify-write request over a PiC serial port to the remote device. When the EN input is one-shot, a Data Highway Plus transaction described by the data at the inputs is started over the serial port. If the transaction completes successfully, the DONE output is energized. If the transaction does not complete successfully, the FAIL output is energized and an error code will appear at ERR, STS, or XSTS indicating the reason for the failure.

For each block of data to be transferred (up to approximately 240 bytes per transfer), this function block must be called. Typically this function block is repeated in the ladder for each different block of data to be transferred. Each time the function block appears in the ladder, it must be declared with a different Name.

This function uses a request queue that can hold up to 10 requests before overflowing. Each request is placed on the queue by this function block and processed (removed from the queue) by the C_DHPXCV function block. The requests are sent over the serial port in the order they are received. Each request must complete before the next request will be sent. The purpose of the queuing mechanism is to allow you to make more than one request from the ladder before the previous request completes.

Inputs

RQ The RQ input is energized to make a request over the Data Highway Plus network. Use a one-shot or transitional to ensure that this input is energized for only one scan. When energized, the data from the other inputs is moved onto the queue for transmission by the C-DHPXCV function block. This function block then monitors the status flag of that queue entry waiting for the transaction to complete.

Q This array of structures is used to specify a request queue area that transfer requests can be held in until they can be processed. Requests are placed in this queue by this function block and retrieved by the C-DHPXCV function block. The structure is the same one that is used at the Q input of the C-DHPXCV function block. It should not be altered in any way since it is used for inter-function block communications only.

Declared array of structures for Q input

```
)               STRUCT(0..10)
 NODE           USINT
 PROT           USINT
 CMD            USINT
 TYPE           USINT
 FILE           UINT
.ELEM           UINT
.CNT            USINT
.LNDX           UINT
.STAT           USINT
.STS            USINT
.EXT_STS        USINT
.SBIT           USINT
                END-STRUCT
```

NODE The number entered here specifies the destination station number for the request. The range of numbers that this input will accept is 0 to 255 (decimal).

> **IMPORTANT**
>
> The station numbers on the Data Highway Plus are typically in octal. The NODE input is expecting a decimal value. It is necessary for you to convert the station number from octal to decimal before entering it.

CMD The command input specifies the kind of request to perform from those listed below. Enter the value of the command you want or the label in the last column that has been declared in the declarations table of the DHP.LDO.

Command	Value	Constant in declarations
READ (Word Range Read)	0	READ
WRITE (Word Range Write)	1	WRITE
READ-MODIFY-WRITE (Write Bit)	2	RMW

TYPE The TYPE input specifies the type of data the request is to operate on in the PiC. Enter the value of the data type you want or the label in the last column that has been declared in the declarations table of the DHP.LDO.

Data Type	# of Bits	Value (Hex)	Constant in declarations
Bool	1 *	0	XF_BOOL
Byte	8	1	XF_BYTE
Word	16	2	XF_WORD
Dword	32	3	XF_DWORD
Lword	64	4	XF_LWORD
Usint	8	5	XF_USINT
Uint	16	6	XF_UINT
Udint	32	7	XF_UDINT
Ulint	64	8	XF_ULINT
Sint	8	9	XF_SINT
Int	16	A	XF_INT
Dint	32	B	XF_DINT
Lint	64	C	XF_LINT
Real	32	D	XF_REAL
Lreal	64	E	XF_LREAL
String	(84 bytes)	F	XF_STRIN
Date	16	10	XF_DATE
Date & time	32	11	XF_DANDT
Time of day	32	12	XF_TOFD
Time	32	13	XF_TIME

*Booleans are referenced in elements of 16 bits.

FILE The FILE input specifies the destination FILE number in the remote station specified at the NODE input of this function. No type checking is done based on the file number and data may be sent to any file.

ELEM The ELEM input specifies the destination ELEMENT number in the remote station specified at the NODE input of this function block.

C N T The CNT input specifies the number of elements of the type specified at the TYPE input that are to be sent for a WRITE command or the number of elements of the type determined by the File type in the remote station that are to be received by the PiC for a READ command.
For boolean data types, this value will always represent the number of lb-bit elements to be transferred.

LNDX The LNDX input specifies the local index or array index where data will be placed or retrieved. The array specified by the TYPE input is the array that this input is the starting index for.
For boolean data types, this value will always specify the starting lb-bit element number rather than the array index.

SBIT The SBIT input specifies the starting bit number within the element specified by the ELEM input for the Read-Modify-Write command only. This value represents the starting bit for both the local and remote node.

outputs

DONE The DONE output energizes when the requested transfer has completed successfully.

FAIL The FAIL output energizes if a problem occurs in the transfer requested. If this output energizes, an error code will be present at one of the following outputs.

ERR
STS
XSTS

OVR The OVR output energizes if the request made with this function block call overflowed the request queue. Ten positions are available in the Request Queue. If this output energizes, the request is ignored and may be attempted later.

ERR The ERR output will contain a value if the FAIL output is set and an error has occurred locally (in the PiCs communications processing. See the error codes for the C_DHPXFR function block below.

STS* The STS output reports the error status code for the remote station.

XSTS* The XSTS output reports the extended error status code for the remote station.

*For additional information on Data Highway Plus error codes, see the Allen-Bradley "Data Highway/Data Highway Plus Protocol and Command Set Reference Manual," Publication 1770 - 6.5.16 - November, 1987 or later.

The following error codes can be returned from the C_DHPXFR function at the ERR output.

Code	Error	Source ladder
1	**Unrecognized data type** The value at the TYPE input is not a valid data type. See the Data type list.	C-DHPMX
2	**Request exceeds array bounds** A request was made for data which caused the array index to exceed the size of the array being accessed. The maximum index is calculated by the following formula. Max Index = CNT * Size of (ELEM) + LNDX *(units = elements * byteslelement + starting element)*	C-DHPMX
3	**Memory transfer overflow** The number of elements being transferred in one transfer caused the memory transfer routine to ovefflow when trying to calculate the number of bytes needed to complete the transfer.	C-DHPMX
17	**Transfer failed (no response)** The other station acknowledged the message but did not respond to the command within the six second timeout period. Typically this error is caused by a mismatch in the Node (station) numbers.	C-DHPXCV
18	**Incorrect response byte count** The number of bytes in the response message is incorrect.	C-DHPXCV
19	**Transfer byte count > 244 during read.** The number of bytes in a read frame is greater than 244.	C-DHPXCV
20	**Transfer byte count z-239 during write** The number of bytes in a write frame is greater than 239.	C-DHPXCV
21	**Incorrect SCR/CMD/TNS** A response is received that contains a value in the SOURCE STATION, COMMAND, or the TRANSACTION NUMBER field which does not match with expected response.	C-DHPXCV

22	**Transfer failed (no acknowledgment)**	C-DHPXCV
	The other station never acknowledged receipt of the request. Check connecting cables and operation of the other station.	
23	**Odd number of bytes**	C-DHPXCV
	The request parameters cause a request for an odd number of bytes. All requests must contain an even number of bytes or whole number of words.	
24	**Frame size exceeded**	C-DHPXCV
	A write request generated a total frame size greater than 256 bytes.	
25	**Data link layer frame size exceeded**	C-DHPXCV
	The data link layer tried to process a frame larger than 256 bytes.	
33	**Transmission queue full**	C-DHPXCV
	The last transfer request (execution of C_DHPXFR function block) caused the 10 position request queue to overflow. The request did not get into the queue and was not sent. Once a transfer request is made only nine more requests can be made before the first one completes. If a tenth transfer request is made this error is returned. NOTE: If your ladder logic is only trying to make one request at a time and this error is received, it may be that the EN input of one of the C_DHPXFR function blocks is being enabled for multiple scans instead of being one-shot. This would cause an additional transfer request each scan.	

1.5 Modes of operation

The function blocks can be used in two modes of operation.

1. The slave mode which uses the C_DHPXCV function block,
2. The master/slave mode which uses the C_DHPXCV and the C_DHPXFR function blocks.

If you are creating a new application ladder, open the C_DHPEX.LDO shown below and use the save As command to name it whatever your application will be called.

If you want to add the C_DHPEX.LDO to an existing application ladder, use the optional LDOMERGE software to combine them.

Both of these methods produce an application ladder with all the software declarations in C_DHPEX.LDO entered for you. You can also enter the declarations manually, but this is not recommended.

The C_DHPEX.LDO

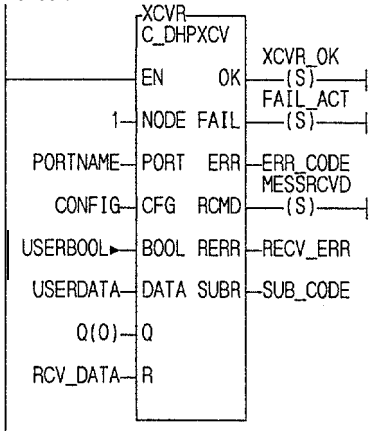
```
Workstation    Processor Module Declarations Network    Element    View
|...1.......
Initialize Data Highway+ communications (DF1 protocol) over the User Port.

Define this node as Node 1. NODE-NUM = 1
Port is setup for 9600 baud, 8 data bits, No parity, 1 Stop Bit.

If a KF2 module is being used, set the node number to match the KF2 module's
station number. Remember that the KF2 station number is in octal.

NAME and CONFIG are initialized with initial values under software
declarations. The format of the CONFIG string is the same as the NAMZ input
of the OPEN function found in IO.LIB

If one of the channels on the 4 channel serial module is going to be used, it
is necessary to use the ASSIGN function to associate the string NAME with the
actual port on the serial module. See the ASSIGN function in the software
manual.
```



Software Declarations Table for C_DHPEX.LDO

This section contains the declarations for the C_DHPXCV function block and some of its inputs and outputs.

Array at BOOL input

Structure at DATA input

The first member (BOOL_S) is the size of the BOOL array above. The remaining members are in pairs for each PiC data type - the first is the size and the second is the data type array. The number of elements (2 to 999) you enter in the array determines the number entered in size.

Name	Type	I/O Pt.	Init. Val.	Long Name	Top
CVR	<fb>C_DHPXCV				
PORTNAME	STRING[10]		USER:$00	(Long Names	
CONFIG	STRING[15]		9600,N,8,1►	omitted here	
CVR_OK	BOOL			for clarity.	
CVR_FL	BOOL			View them	
CVR_ERR	INT			in PiCPro.)	
MESSRCVD	BOOL				
RECV_ERR	USINT				
SUB_CODE	USINT				
USERBOOL	BOOL(0..31)				
USERDATA	STRUCT				
.BOOL_S	UINT		2		
.BYTE_S	UINT		2		
.BYTE_D	BYTE(0..1)				
.WORD_S	UINT		2		
.WORD_D	WORD(0..1)				
.DWORD_S	UINT		2		
.DWORD_D	DWORD(0..1)				
.LWORD_S	UINT		2		
.LWORD_D	LWORD(0..1)				
.USINT_S	UINT		2		
.USINT_D	USINT(0..1)				
.UINT_S	UINT		2		
.UINT_D	UINT(0..1)				
.UDINT_S	UINT		2		
.UDINT_D	UDINT(0..1)				
.ULINT_S	UINT		2		
.ULINT_D	ULINT(0..1)				
.SINT_S	UINT		2		
.SINT_D	SINT(0..1)				
.INT_S	UINT		2		
.INT_D	INT(0..1)				
.DINT_S	UINT		2		
.DINT_D	DINT(0..1)				
.LINT_S	UINT		2		
.LINT_D	LINT(0..1)				
.REAL_S	UINT		2		
.REAL_D	REAL(0..1)				
.LREAL_S	UINT		2		
.LREAL	LREAL(0..1)				
.STRING_S	UINT		2		
.STRING_D	STR[82](0..1)				
.DATE_S	UINT		2		
.DATE_D	DATE(0..1)				
.DANDT_S	UINT		2		
.DANDT_D	D_AND_T(0..1)				
.TOFD_S	UINT		2		
.TOFD_D	T_OF_D(0..1)				
.TIME_S	UINT		2		
.TIME_D	TIME(0..1)				
	END-STRUCT				

Q Array of structures The C_DHPXCV function block retrieves transfer information placed in this queue by the C-DHPXFR function block when working in the master/slave mode.	Q	STRUCT(0..10)	
	.NODE	USINT	
	.PROT	USINT	
	.CMD	USINT	
	.TYPE	USINT	
	.FILE	UINT	
	.ELEM	UINT	
	.CNT	USINT	
	.LNDX	UINT	
	.STAT	USINT	
	.STS	USINT	
	.EXT_STS	USINT	
	.SBIT	USINT	
		END-STRUCT	
R structure	RCV_DATA	STRUCT	
	.NODE	USINT	
	.CMD	USINT	
	.TYPE	USINT	
	.ELEM	UINT	
	.CNT	USINT	
		END-STRUCT	
Constants for CMD input on C-DHPXFR	READ	USINT	0
	WRITE	USINT	1
	RMW	USINT	2
Constants for TYPE input on C_DHXPFR	XF_BOOL	USINT	0
	XF_BYTE	USINT	1
	XF_WORD	USINT	2
	XF_DWORD	USINT	3
	XF_LWORD	USINT	4
	XF_USINT	USINT	5
	XF_UINT	USINT	6
	XF_UDINT	USINT	7
	XF_ULINT	USINT	8
	XF_SINT	USINT	9
	XF_INT	USINT	16#A
	XF_DINT	USINT	16#B
	XF_LINT	USINT	16#C
	XF_REAL	USINT	16#D
	XF_LREAL	USINT	16#E
	XF_STRIN	USINT	16#F
	XF_DATE	USINT	16#10
	XF_DANDT	USINT	16#11
	XF_TOFD	USINT	16#12
	XF_TIME	USINT	16#13
	end-table	void	
	=Alt-M mod attrib==Press F10 to exit==Alt-E ent field=Bottor		

NOTE

In the master/slave mode, a C-DHPXFR function block must be called for each block of information transferred.

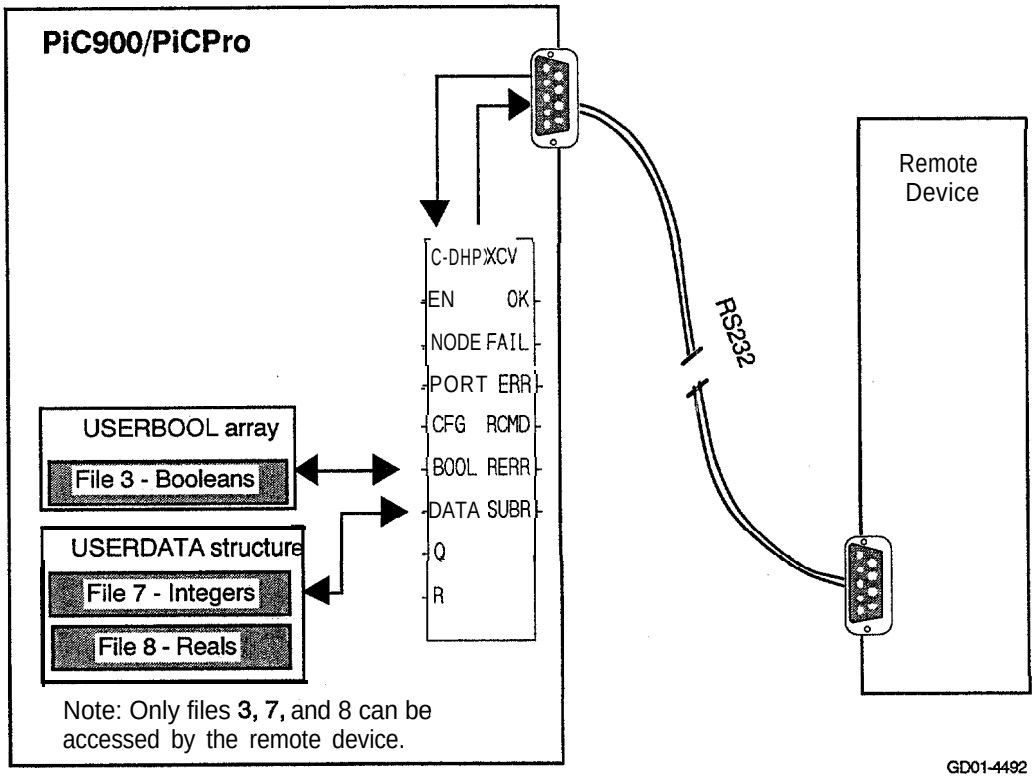
Each occurrence of the C_DHPXFR function block in your application ladder must be declared with a different name in the software declarations table.

Slave mode

In the slave mode the PiC will receive commands from other devices but will not initiate any transfers. Only one function block, C_DHPXCV, is required to enable the slave mode. It is responsible for all communications to and from the PiC for Data Highway Plus support. As shown in Figure 3 below, the only data types available in the slave mode are booleans, integers, and reals.

Remember that when operating in the slave mode, the PiC cannot initiate any communications; it can only respond to requests from the remote device.

Figure 3. Slave mode



All data being sent to or retrieved from the PiC will have a File number associated with it. Although this file number has no direct equivalent in the PiC, it is used to determine where to place or retrieve data. The following list contains the valid File numbers and the PiC memory areas these files correspond to.

File #	Type of data	PiC memory area	Example
3	boolean (bit)	BOOL input of C-DHPXCV	USERBOOL (0)
7	integer	DATA input of C-DHPXCV	USERDATA.INT_D (0)
8	floating point	DATA input of C-DHPXCV	USERDATA.REAL_D (0)

The PiC will only respond to requests directed at one of these files. Requests made to any other file number will generate an error response to the device that made the request.

To enable the PiC to communicate as a Data Highway Plus slave, the C_DHPXCV function block must be entered in your application ladder once. Enable it every scan. Each input must have the appropriate variable attached to it.

There are some limitations when using the slave mode.

1. Only the data sent to files 3, 7, and 8 will be accepted by the PiC. Requests accessing any other file number will cause an error response to be generated.
2. Each data transfer must complete in one packet. Multiple packets are not supported. This limits the amount of data that may be sent per transfer to 244 bytes for WRITEs and 239 bytes for READs.
   NOTE: WRITE and READ here refer to the action of the remote device, not the PiC.
3. PLC-5 Logical Binary addressing scheme is supported. Logical ASCII addressing is not supported.

Slave mode setup

1. Determine how many booleans (bit type) will be needed for your application and round that number up to the nearest multiple of 16. The maximum number of booleans is limited to 992.

   Example: If 50 booleans are required, the next highest multiple of 16 is 64. 64 booleans need to be created.
2. Modify the size of the USERBOOL array in the software declaration table by setting it to the size determined in step 1. In the software declarations table, place the cursor on the data item named USERBOOL and press <Alt> A to enter the array length.

   For the example, 64 is entered in the Array Length box.
3. The size of the boolean array (USERBOOL) must be placed in the first field of the USERDATA data structure: BOOL-S (boolean size). Place the cursor on the Init. Val. column for the BOOL-S data item and press <Alt> E to enter the number. Calculate the number to enter by dividing the number determined in step 1 by 16.

   For the example, the number is 4.
4. Determine how many integers will be needed for your application. The acceptable range is from 1 to 999. If no integer data is required, go to step 7.
5. Modify the size of the INT-D array in the USERDATA data structure. In the software declarations table, place the cursor on the data item INT-D and press <Alt> A. Enter the number of integers required.
6. The size of the integer array must be placed in the field just above the INT-D array, namely INT S (integer size). Place the cursor in the Init. Val. column for the INT_S data-item, press <Alt>E, and enter the correct value.
7. Determine how many reals (floating point numbers) will be needed for your application (up to 999). If no real data is required, go to step 10.
8. Modify the size of the REAL-D array in the USERDATA data structure. In the software declarations table, place the cursor on the REAL-D data item and <Alt> A. Enter the number of reals required.
9. The size of the REAL array must be placed in the field just above the REAL-D array in the REAL-S. Place the cursor on the Init. Val. column for the REAL-S data item, press <Alt>E and enter the correct value.

10. Determine the node (station) number for the PiC and place that number at the NODE input of the C_DHPXCV function block.

11. Determine which serial port is going to be used for the Data Highway Plus communications. If the User Port is going to be used, initialize a string type variable as follows:

    PORTNAME STRING(lO) "USER:$00"

12. Determine the proper communications configuration for the serial port. Then assign a string type variable an Initial Value that when placed at the CFG input will configure the communications channel. See the CFGZ input on the CONFIG function in the PiCPro Software Manual for more information on the configuration string.

    Example: For 9600 baud, No parity, 8 data bits, 1 stop bit, No handshake
    CONFIG STRING(15) "9600,N,8,1,N,$00"

    Example: For 19200 baud, Even parity, 8 data bits, 1 stop bit, No handshake
    CONFIG STRING(15) "19200,E,8,1,N,$00"

Master/Slave mode

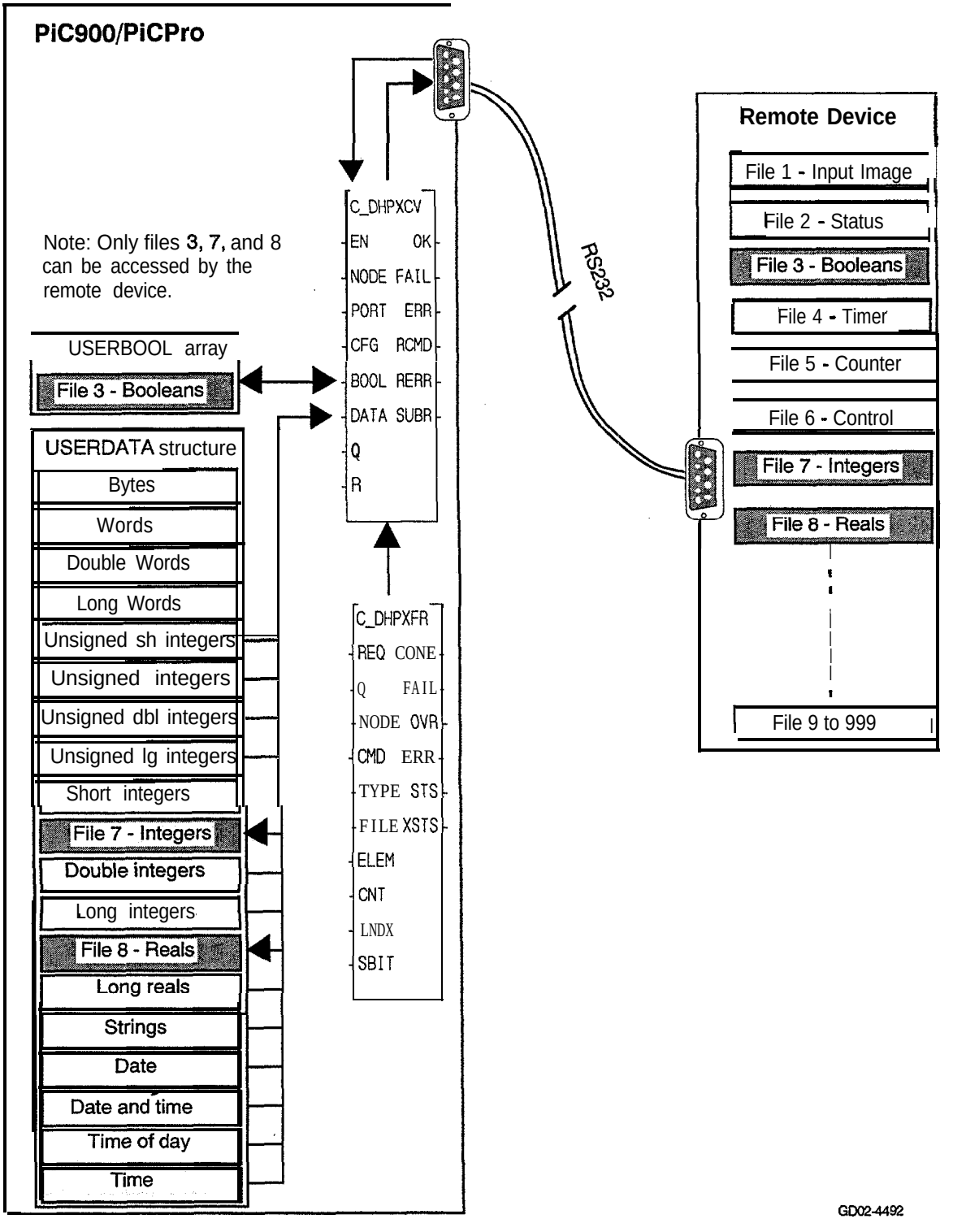
In the master/slave mode the PiC can receive commands from and initiate communications to other devices. See Figure 4. A data transfer is requested through the use of the C_DHPXFR function block in the application ladder. C-DHPXFR has several inputs which allow a single transaction to be described. For each transaction a call to C-DHPXFR must be made. Typically after a transaction completes another request is made. Up to 10 requests can be made before the first one finishes. These requests are placed in a queue by the C-DHPXFR function block and processed by the C-DHPXCV function block one at a time.

In the master mode, any data in the PiC may be written to any File number in the other device. It is your responsibility to make sure that the data formats are compatible. Also, any data read from another device may be place in any of the data types in the PiC.

All data transferred to or from the PiC is placed in or retrieved from the data structure placed at the DATA input of the C-DHPXCV function block. Bit data is an exception. It is placed in the boolean array at the BOOL input.

To enable the PiC to communicate as a Data Highway Plus master, the C-DHPXCV function block must be present in the ladder. It should be placed in the ladder once and enabled every scan. Each input of the C-DHPXCV function block must have the appropriate variable attached to it.

Figure 4. Master/Slave Mode



Master/Slave mode setup

1. Determine the node (station) number for the PiC and place that number at the NODE input of the C_DHPXCV function block.
2. Determine which serial port is going to be used for the Data Highway Plus communications. If the User Port is going to be used, initialize a string type variable as follows:

   PORTNAME STRING(lO) "USER:$00"
3. Follow the steps listed at the slave mode setup.

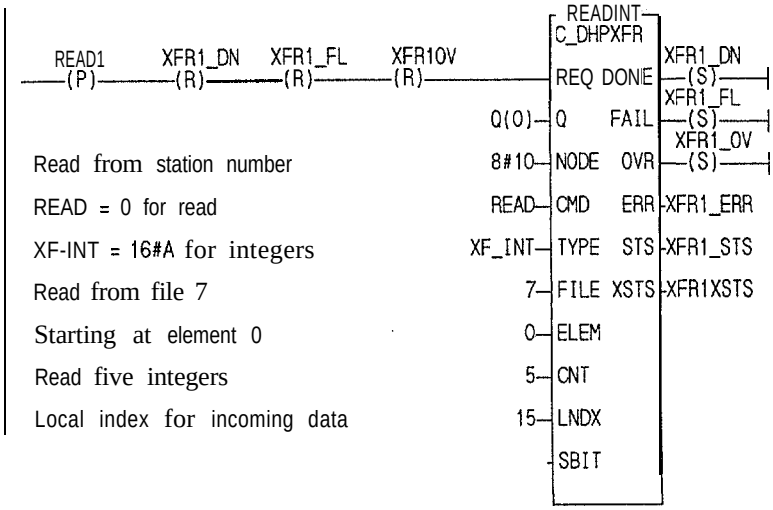
1.6 Examples

The following examples illustrate a READ, a WRITE, and a READ-MODIFY-WRITE request made by the PiC in the master/slave mode of operation.

EXAMPLE 1 - READ

An example of a READ request made by the PiC to another station.

This example will read five integers from station (NODE) 10 (octal) or 8 (decimal). The five integers read will be placed in the INT_D( ) array of the USERDATA data structure starting at element 3.5.



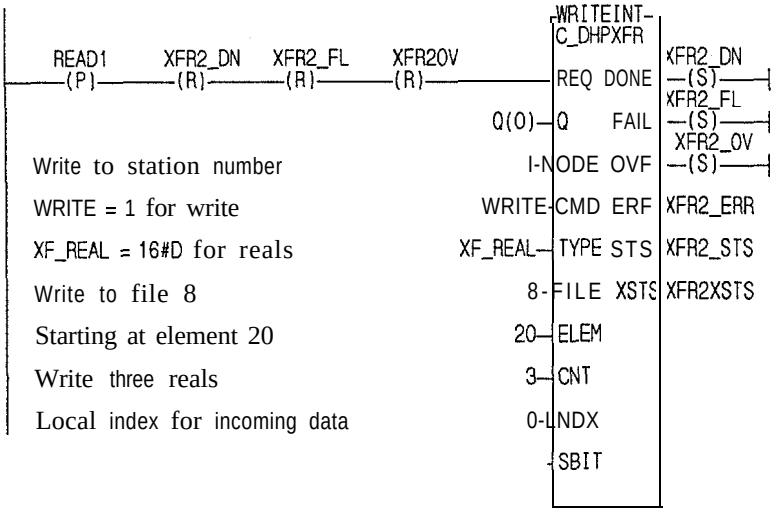
NOTE: In PLC addressing terminology, this is equivalent to:

N7:0
5 integers

EXAMPLE 2 - WRITE

An example of a WRITE request made by the PiC to another station.

Three reals from the PiC will be written to the station (NODE) 1. The three reals will be taken from the USERDATA.REAL_D ( ) array starting at element 0 and sent to file 8 of the other device starting at element 20.



NOTE: In PLC addressing terminology, this is equivalent to:

F8:20
3 reals

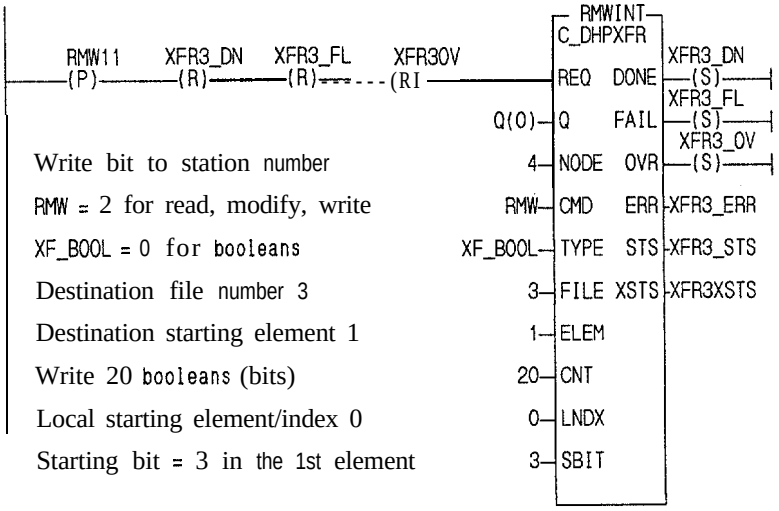
EXAMPLE 3 - READ-MODIFY-WRITE

An example of a READ-MODIFY-WRITE request made by the PiC to another station.

Twenty booleans from the PiC will be written to the station (NODE) 4. The 20 booleans will be taken from the USERBOOL ( ) array at the BOOL input of the C_DHPXCV function block and sent to file 3, element 1, starting with bit 3 in the second word (16 bit group.)

When boolean type data is being transferred, the ELEM input refers to which block of 16 booleans to start at. The SBIT input specifies which bit (starting with the low bit) within the starting element to transfer to or from.

NOTE: The SBIT input is the starting bit number for both the source and destination stations, but the local element number LNDX and the destination element ELEM can be different.



NOTE: In PLC addressing terminology, this is equivalent to:

B3:1
starting at bit 3
next 20 bits

The diagrams below help to visualize the transfer in example 3. Before the transfer occurs, all the bits in the source are ON or set and all the bits in the destination are OFF or cleared. Twenty bits are going to be written to station (node) 4. The bit data is taken from the PiC starting at USERBOOL (3) and ending at USERBOOL (22). It will be sent to the third bit of element 1 or File 3 bit 19.

PiC USERBOOL source

Element 0	Element 1	Element 2
11111111 11111111	11111111 11111111	11111111 11111111
USERBOOL (0)	USERBOOL (16)	USERBOOL (32)

Remote device station 4 destination

Element 0	Element 1	Element 2
00000000 00000000	00000000 000000000	0000000 00000000
File 3 Bit 0	File 3 Bit 16	File 3 Bit 32

Remote device station 4 after transfer

Element 0	Element 1	Element 2
00000000 00000000	0001111111111111	11111110 00000000
File 3 Bit 0	File 3 Bit 16	File 3 Bit 32

NOTES

Index