PC P-NET® Interface board

PD 3920

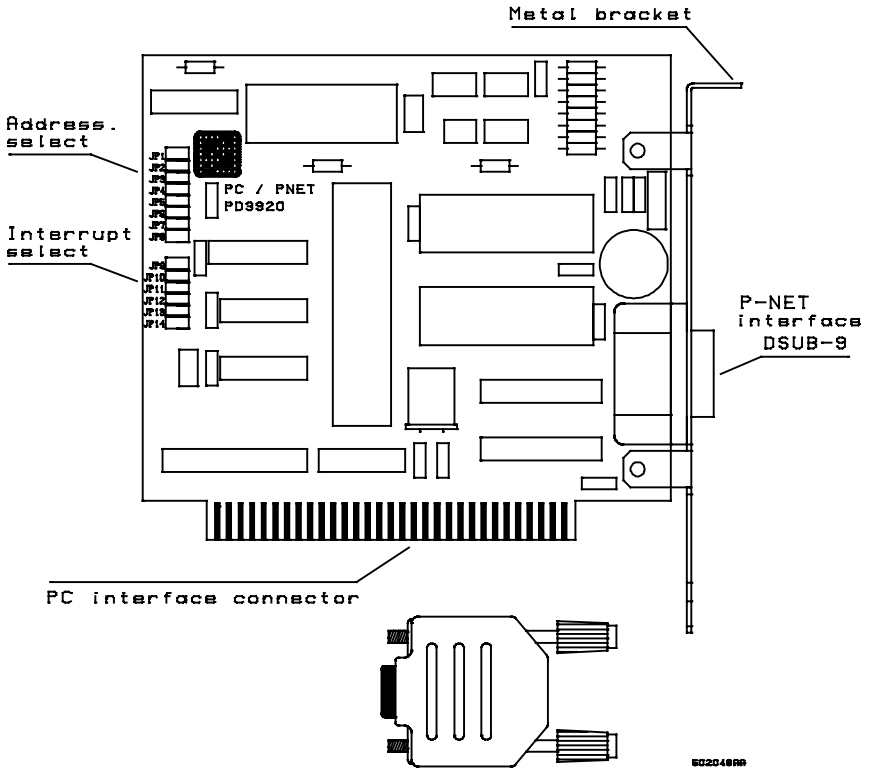
Manual

PC P-NET® interface board, PD 3920.

The PC P-NET® interface board, PD 3920 is an intelligent interface board used for communication to a fieldbus. Installing a PC P-NET® interface board, PD 3920 in a PC provides you the facility to communicate directly to a fieldbus, called P-NET®. Communicating with the P-NET® using the P-NET® protocol enables you to communicate with ALL modules connected to the P-NET®, including both interface modules and controllers, without doing any extra PC-dedicated programming in the controllers or the interface modules. The PC communicates with the PD3000 controller and interface modules with the P-NET® protocol by using a special communication driver, called P-NET® driver and an interface board, the PC P-NET® interface board PD 3920. The baud rate is fixed at 76800 baud, corresponding to the P-NET® standard.

Using the PC P-NET® interface board.

Installing the PC P-NET® interface board.



Before you install your PC P-NET interface board you must prepare your computer in the following way:

1. Turn power OFF.
2. Disconnect the power cord and all other cables from the back of your system unit.
3. Locate and remove the cover mounting screws and slide the cover of your computer off.
4. Examine the interface board to familiarize yourself with the position of the jumpers.
5. You need to set the jumpers on your interface board to ensure that the interface board is compatible with your computer. Refer to the following table for the correct jumper settings for the type of computer you are using.

Description of jumpers.

BASE address jumper (JP1-JP8):

Jumper	Address	Normally used for		
		PC/XT	PC/AT	
JP1:	070 (hex)	RTC, Cmos ram	RTC,Cmos ram	Do not use!
JP2:	0F0 (hex)	Math. Co-proc.	Math. Co-proc	Do not use!
JP3:	270 (hex)		LPT2*	
JP4:	2F0 (hex)	COM2*	COM2*	
JP5:	170 (hex)			Factory setting
JP6:	1F0 (hex)		Hard Disk	
JP7:	370 (hex)	LPT1*	LPT1*	
JP8:	3F0 (hex)	COM1*	COM1*	

Interrupt selector jumper (JP9-JP14):

Jumper	Interruptnumber	Normally used for		
		PC/XT	PC/AT	
JP9:	2	RESERVED	EXTENDED	Do not use!
JP10:	3	COM2*	COM2*	Factory setting
JP11:	4	COM1*	COM1*	
JP12:	5	Fixed Disk	LPT2*	
JP13:	6	Diskette	Diskette	Do not use!
JP14:	7	LPT1*	LPT1*	

When you have determined the correct jumper settings for your computer, set the jumpers on the appropriate pair of pins. Record the settings of the jumpers for later installation of the PNETCARD driver in the CONFIG.SYS file.

After you have set the jumpers, perform the following steps to install the interface board in the expansion slot in your computer.

1. Locate a free expansion slot on your computer. You may need to refer to your computer's User's Manual to choose the correct expansion slot.
2. Remove the expansion slot cover (the metal plate on the back panel of the system unit) by first removing the screw which hold the cover in place and then carefully lifting off the slot cover completely.
3. Insert the interface board firmly in your computer's expansion slot. Make sure that the connector (the gold striped end of the interface board) is setting properly in the expansion slot groove and the bracket of the interface board is in the groove which previously held the slot cover.
4. After checking that the top of the interface board is aligned with the hole on the top of the expansion slot, replace the screw which previously held the expansion slot cover.Tighten the screw firmly so it holds the bracket of the interface board in place.
5. Replace the cover of your computer and tighten the cover mounting screws.
6. Reconnect the cables previously removed but leave the power switch for your system unit OFF.
7. Locate the 9-pin male PC P-NET connector on the back of your system unit and gently press the P-NET connector in place. Tighten the screws on the side of the 9-pin PC P-NET connector to secure it.

P-NET connection.

The P-NET fieldbus is connected in a ring circuit. It is recommended to connect the P-NET to the PC through a stub connector. The stub length must not exceed 2 meters.

Description of the 9-pin male PC P-NET connector.

Pin	Connection
1	P-NET A
2	NC
3	P-NET S
4	NC
5	P-NET B
6	NC
7	NC
8	NC
9	NC

NC = not connected

The P-NET S (shield) must be isolated from the plug housing, thus a DSUB-9 plug with a plastic house is recommended.

Electrical specification.

Power supply (via the PC bus)	5 Volt, 150 mA (max 300 mA)
Bus interface	XT/AT compatible bus-interface
P-NET interface	RS485 2 wire twisted and shielded, galvanically separated from the interface board.

The P-NET driver.

Installing the P-NET driver.

This description is valid for the following driver version:

PNETCARD.SYS: vers. 1.00.

This version is for use with the PC P-NET interface board (’PNETCARD.SYS’). The P-NET driver is a resident program, which is installed in memory when the PC is booted. To install the driver, the following line must be inserted in CONFIG.SYS:

```
DEVICE=[d:][path]driver [/base] [/int] [/masters] [/nodeaddr]
```

[d:][path]
specifies the drive and directory where the driver is to be found,

driver
specifies the driver file name, PNETCARD.SYS

base
specifies the I/O port selected with the jumper JP1-JP8 on the PD3920 interface board.

int
specifies the interrupt selected with the jumper JP9-JP14 on the PD3920 interface board.

masters
specifies the number of masters on the P-NET.

node
is the PC’s P-NET node address.

Example:

```
DEVICE=C:\PNET\PNETCARD.SYS /BASE:$170 /INT:4 /MASTERS:6 /NODE:1
```

Using the P-NET driver in a TURBO-PASCAL program.

This description is valid for the following driver versions:

PNETCARD.SYS	vers. 1.00.
PNETDRIV.86	vers. 3.08.
PNETDRIV.286	vers. 3.08.

A TURBO PASCAL file is included with the driver. This file contains different procedures for use in the program. These procedures perform opening and closing the driver, loading and storing data via the driver and converting of the various data types to/from TURBO PASCAL and the interface modules including the PD3000 controller. Each of these procedures are described below.

Data exchange.

When using the driver from a TURBO-PASCAL program (ver. 4.0 later), the following record must be used to exchange variables with the driver and the program. The variable of the record type must hold the correct data before data are loaded/stored. The procedures to load/store data via the driver are listed below and are delivered in a PNETUNIT.PAS file together with the driver.

```
TYPE
  buf56 = ARRAY[0..55] OF BYTE;

P_Net_Block = RECORD
            P_Net_No    : STRING[25];
            SoftWire_No : WORD;
            AdrLow      : WORD;
            Offset      : INTEGER;
            DataReady   : INTEGER;
            ErrorCode   : INTEGER;
            NumOfByte   : INTEGER;
            PnetData    : buf56;
          END;
```

P_Net_No

This field must hold the complete node address for the module you want to access. The complete node address is given as a sequence of P-NET node addresses and port numbers, calculated from the PC. The first byte in the string (P_Net_No[0]) holds the number of bytes in the complete node address. The P-NET node addresses and port numbers must be in hexadecimal (NOT ASCII).

SoftWire_No

This field must hold the softwire number for the variable you want to access. The softwire number can be a softwire number in a PD3000 controller or it can be the register address in an interface module. When using absolute addressing, this field must hold the higher two byte of the address. The lower two byte of the address must be stored in AdrLow.

AdrLow

When using absolute addressing, this field must hold the lower two byte of the address. The higher two byte of the address must be stored in SoftWire_No.

Offset

This field holds an offset, in byte, to the data from the specified SoftWire_No or address. The Offset can be used when accessing a complex variable, e.g. array or record.

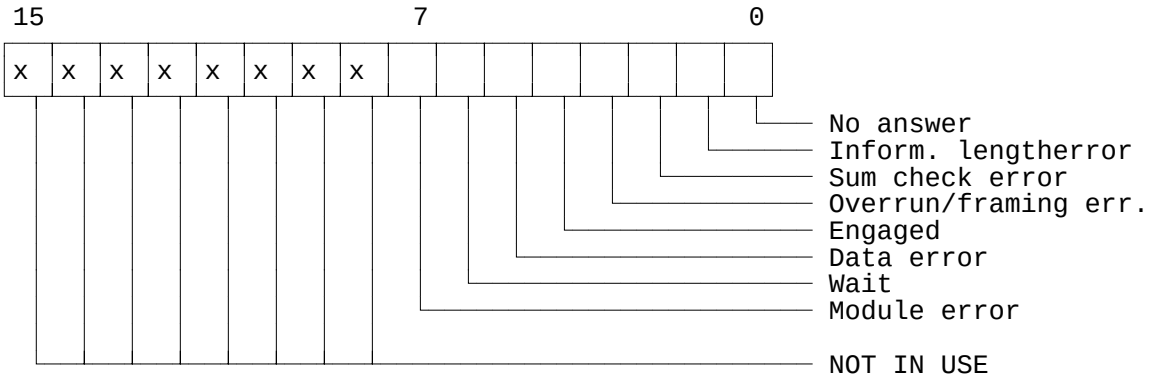
DataReady

This field denotes if the transmissions to/from the P-NET are ready. DataReady has the following meaning:

0: The transmission has not finished yet, the data are not loaded/stored.
1: Data are loaded/stored and the driver is ready.
2: Reserved.
3: Reserved.

ErrorCode

This field denotes the errorcode from the driver as a result of the transmission. The errorcode consists of 16 bit (an integer), where each bit has the following meaning:



Combined error bits with a separate interpretation:

bit 0..1	($0003):	Time out in "Answer comes later"
bit 0..3	($000F):	Error on other net.
bit 0..4	($001F):	Too many "Engaged/Wait"
bit 0..3,5	($002F):	Buffer full/empty

Other combinations of the above error bits are a result of more than one error from the transmission and must be interpreted in a special way.

NumOfByte

This field must hold the number of byte you want to load or store. When accessing a PD3000 controller, you must be careful not to access more than one byte if you have an **odd** address (absolute addressing or odd offset). If you do so, the controller will take a reset (restriction in the 68000 microprocessor in the controller).

PnetData

This field holds the data that you want to load/store via P-NET. The data must be converted and stored in the array before they are stored via P-NET, and after loading they must be converted before they are stored in the variables in the PC. These conversion procedures between TURBO PASCAL and a PD3000 controller (IEEE format) and other interface modules are listed below and are delivered in a PNETUNIT.PAS file together with the driver.

Communication procedures.

InitDriver

Before the driver is used, the interface board or the serial port and the driver must be initialized. This is done by calling the function **InitDriver**. This function is a boolean and is only called once in the program. The function returns true if the driver is installed.

CloseDriver

Before the program is terminated, the driver must be closed. This is done by calling the procedure **CloseDriver**. The driver is opened as a file by InitDriver and when closed, its DOS file handle is freed for reuse.

LoadPNetData(VAR PNB: P_Net_Block)

This procedure loads data via the P-NET. The P_Net_No, SoftWire_No and number of byte must be set up in a variable of type P_Net_Block, before the procedure is called. The loaded data are stored in PNB.PnetData.

StorePNetData(VAR PNB: P_Net_Block)

This procedure stores data via the P-NET. The P_Net_No, SoftWire_No and number of byte and the data must be set up in a variable of type P_Net_Block, before the procedure is called.

AndPNetData(VAR PNB: P_Net_Block)

This procedure performs a bitwise AND via the P-NET, with one operand being the data in P_Net_Block.PnetData and the other operand being data on the net, specified by the P_Net_No and SoftWire_No. The P_Net_No, SoftWire_No and number of byte must be set up in a variable of type P_Net_Block, before the procedure is called.

OrPNetData(VAR PNB: P_Net_Block)

This procedure performs a bitwise OR via the P-NET, with one operand being the data in P_Net_Block.PnetData and the other operand being data on the net, specified by the P_Net_No and SoftWire_No. The P_Net_No, SoftWire_No and number of byte must be set up in a variable of type P_Net_Block, before the procedure is called.

AbsLoadPNetData(VAR PNB: P_Net_Block)

This procedure loads data via the P-NET. The P_Net_No, absolute address and number of byte must be set up in a variable of type P_Net_Block, before the procedure is called. The loaded data are stored in PNB.PnetData.

AbsStorePNetData(VAR PNB: P_Net_Block)

This procedure stores data via the P-NET. The P_Net_No, absolute address, number of byte and the data must be set up in a variable of type P_Net_Block, before the procedure is called.

AbsAndPNetData(VAR PNB: P_Net_Block)

This procedure performs a bitwise AND via the P-NET, with one operand being the data in P_Net_Block.PnetData and the other operand being data on the net, specified by the P_Net_No, SoftWire_No and AdrLow. The P_Net_No, absolute address and number of byte must be set up in a variable of type P_Net_Block, before the procedure is called.

AbsOrPNetData(VAR PNB: P_Net_Block)

This procedure performs a bitwise OR via the P-NET, with one operand being the data in P_Net_Block.PnetData and the other operand being data on the net, specified by the P_Net_No, SoftWire_No and AdrLow. The P_Net_No, absolute address and number of byte must be set up in a variable of type P_Net_Block, before the procedure is called.

SetPNetNumber(Number: INTEGER);

This procedure set the PC’s P-NET node address to Number.

GetPNetNumber(VAR Number: INTEGER);

This procedure returns the PC's node address (P-NET number).

SetMasters(Number: INTEGER);

This procedure defines the number of masters on the P-NET connected to the PC. This is only valid if the PD3920 PC P-NET interface board is installed in the PC. IF the PD3920 interface board is not installed, this procedure have no effect.

GetMasters(VAR Number: INTEGER);

This procedure returns the number of masters on the P-NET connected to the PC. This is only valid if the PD3920 PC P-NET interface board is installed in the PC. If the PD3920 interface board is not installed, this procedure have no effect.

SetBaudRate(Baud: WORD);

This procedure defines the baudrate on the P-NET connected to the PC's serial port. This is only valid if you use the serial port otherwise this procedure have no effect.

GetBaudRate(VAR Baud: WORD);

This procedure returns the baudrate on the P-NET connected to the PC's serial port. This is only valid if you use the serial port otherwise this procedure have no effect.

SetPort(Port: WORD);

This procedure defines the communication port for the P-NET. This is only valid if you use the serial port's otherwise this procedure have no effect. Valid ports are 1 for COM1 and 2 for COM2.

GetPort(VAR Port: WORD);

This procedure returns the port number. If you use a serial driver port are 1 for COM1, and 2 for COM2. If you use PC P-NET interface board Port is the BasePort.

GetDriverVersion(VAR Major, Minor: INTEGER);

This procedure loads the driver version from the P-NET driver. This is only valid if you use the PC P-NET interface board otherwise this procedure have no effect.

GetDriverType: INTEGER;

This function returns the driver type. The following types is defined:

1: Serial driver for 80286/386/486
2: Serial driver for 8088/86
3: Driver for PC P-NET interface board

GetPNetCardVersion(VAR Major, Minor: INTEGER);

This procedure loads the software version from the PD3920 PC P-NET interface board. If used on a system using the serial port, the result is (0,0).

Conversion procedures.

ByteFromPd3000(VAR PD: buf56; VAR b: BYTE)

This procedure converts data from the field variable **PnetData** into a byte. The procedure must be called after loading a byte via the P-NET.

ByteToPd3000(VAR b: BYTE; VAR PD: buf56)

This procedure converts a byte to the field variable **PnetData**. The procedure must be called before a byte is stored via the P-NET.

IntegerFromPd3000(VAR PD: buf56; VAR Int: INTEGER)

This procedure converts data from the field variable **PnetData** into an integer. The procedure must be called after loading an integer via the P-NET.

IntegerToPd3000(VAR Int: INTEGER; VAR PD: buf56)

This procedure converts an integer to the field variable **PnetData**. The procedure must be called before an integer is stored via the P-NET.

WordFromPd3000(VAR PD: buf56; VAR W: WORD)

This procedure converts data from the field variable **PnetData** into a word. The procedure must be called after loading a word via the P-NET.

WordToPd3000(VAR W: WORD; VAR PD: buf56)

This procedure converts a word to the field variable **PnetData**. The procedure must be called before a word is stored via the P-NET.

LongIntFromPd3000(VAR PD: buf56; VAR LI: LongInt)

This procedure converts data from the field variable **PnetData** into a longinteger. The procedure must be called after loading a longinteger via the P-NET.

LongIntToPd3000(VAR LI: LongInt; VAR PD: buf56)

This procedure converts a longinteger to the field variable **PnetData**. The procedure must be called before a longinteger is stored via the P-NET.

RealFromPd3000(VAR PD: buf56; VAR R: REAL)

This procedure converts data from the field variable **PnetData** into a real. The procedure must be called after loading a real via the P-NET.

RealToPd3000(VAR R: REAL; VAR PD: buf56)

This procedure converts a real to the field variable **PnetData**. The procedure must be called before a real is stored via the P-NET.

LongRealFromPd3000(VAR PD: buf56; VAR R: REAL)

This procedure converts data from the field variable **PnetData** into a real. The procedure must be called after loading a longreal via the P-NET.

LongRealToPd3000(VAR R: REAL; VAR PD: buf56)

This procedure converts a real to the field variable **PnetData**. The procedure must be called before a longreal is stored via the P-NET.

OldRealFromPd3000(VAR PD: buf56; VAR R: REAL)

This procedure converts data from the field variable **PnetData** into a real. The procedure must be called after loading an oldreal via the P-NET.

OldRealToPd3000(VAR R: REAL; VAR PD: buf56)

This procedure converts a real to the field variable **PnetData**. The procedure must be called before an oldreal is stored via the P-NET.