PC3000

invensys
EUROTHERM

ENG

DeviceNet user guide

CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. SCOPE

This document describes the implementation of a DeviceNet slave on a PC3000. It contains installation instructions for the hardware and software and outlines the necessary configuration details for the module. It contains sections describing the different modes of operation supported by the function block library and provides some example program fragments.
Details of the function blocks can be found in Related Documents [1].
The library containing these function blocks also contains other network related function blocks for Profibus DP Master operation. Use of those blocks can be found in Related Documents [2].

2. RELATED DOCUMENTS

Title	Document Number
[1] Fieldbus on PC3000, Function block reference	HA027900
[2] Profibus on PC3000, user guide	HA027902

3. OVERVIEW

3.1 Hardware

3.1.1 Version compatibility

The module is Version 3 but may be used in existing Version 2 and Version 1 racks. It is designed to be used with the LCM-PLUS and firmware version 3.20 or higher.

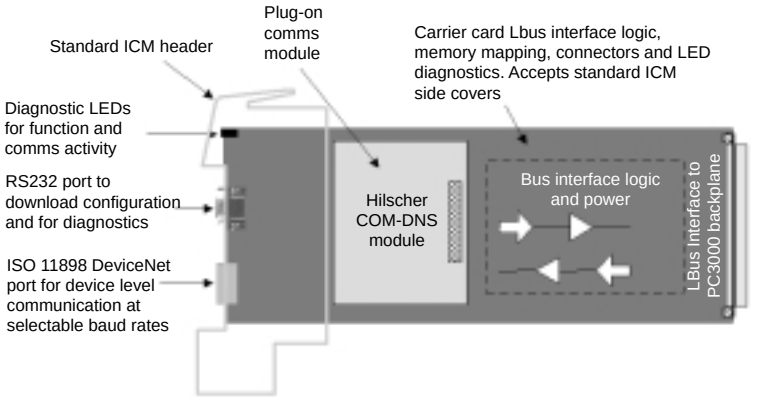
> It is not possible to use the library with earlier versions of the firmware or with a simple LCM.

3.1.2 Description

The PC3000 DeviceNet Slave module provides one isolated communications channel. It can be used in any of the first five slots in a PC3000 main rack. More than one module can be mounted in the rack and the only limitation is that they must be mounted to the right (higher slot address) of any ICM, or other Lbus, modules.
The module comprises a motherboard, which carries a plug-on COM-DNS DeviceNet module from Hilscher GmbH (see Reference 1, paragraph 6.3.1 on page 27). The motherboard provides: -

- physical mounting for the module, connectors and diagnostic LED’s
- power supply
- Lbus interface to the DPM module's dual port memory.

Figure 3-1: Module layout



3.1.3 Module Identification

A label fitted to the side of the module carries details of the serial number etc. The product code is included and should read:-
PC3000/COMM/VERSION3/DEVICENET/SLAVE

3.1.4 Connections

User connection to the module is via two connectors at the front of the module. A 9 way, D-type for configuration and diagnostics and a 5 way CombiCon connector for the DeviceNet network.

3.1.4.1 Configuration port

The top connector is a male and is an unisolated RS232 configuration port. It is for connection to a computer running the Hilscher SyCon configuration software for download and diagnostics.

Table 3-1: Configuration port connections

Pin No	Function
2	Tx
3	Rx
5	Common

3.1.4.2 DeviceNet port

The bottom connector is a 5 way CombiCon connector and is the isolated ISO 11898 DeviceNet port. The actual network connection to remote instruments is via standard DeviceNet cable, which contains both the signal and the power.

Table 3-2: DeviceNet port connections

Pin No.	Function
1	0V external power
2	CANL - data line low
3	Shield
4	CANH - data line high
5	24V external power

External termination at last unit = 124‰

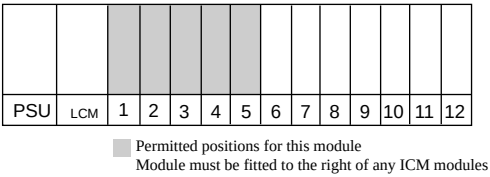
3.1.5 Configuration options

There is no hardware configuration necessary on this module. There are jumpers on the mother board for setting interrupt levels and test features but the module is shipped with the necessary jumpers fitted and these should not be altered.

3.1.6 Location

The module must be located in the main rack in one of the first five I/O positions. It must also be fitted to the right of any standard COM/PORTS4 modules.

Figure 3-2 Module location



3.1.7 Specification

This module performs the function of a DeviceNet slave.

3.1.7.1 Technical Data

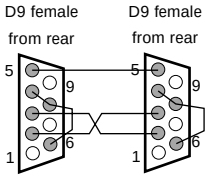
· DeviceNet Slave connection	Potential-free ISO 11898 interface
· Transmission rates	Max 500 Kbaud
· Configuration connection	Potential-linked RS232 interface
· Host interface	510 byte dual port memory
· Consumed data	255 bytes
· Produced data	255 bytes
· Diagnostics	LED's and via RS232 port
· Operating temperature	0 - 55 degrees

3.1.8 Connectors and Cables

3.1.8.1 Configuration and Diagnostics

An RS232 9 way D-type to 9 way D-type cable is available for configuration and diagnostics. The Hilscher part number is KAB-SRV. The cable is bundled with the SyCon configuration software if bought from Eurotherm. The wiring details are shown in Figure 3-3. The cable type is unimportant.

Figure 3-3: Configuration/Diagnostic cable



3.1.8.2 DeviceNet

The DeviceNet connection is via a five-way 5.08 pitch male CombiCon connector mounted on the PCB. The cable, therefore, terminates in a matching female connector. A wide variety of such connectors are available with screwless terminals, spring leaf and rising clamp terminals. They are also available with strain relief features and with cables entering at the side or at the rear. A typical simple connector with screw clamp terminals would be the Weidmuller product, part number BLZ 5.08/5.

The cables are specified by the Open DeviceNet Vendor Association (ODVA). A suitable cable for trunk lines is Belden type 3082A and for drop lines is Belden type 3084A.

3.1.9 Diagnostics

There are four LEDS on the front of the module to provide information about the module operations. These are shown in Table 3-3.

Note: With the current release of this module, the LED's do not conform to the DeviceNet standard and the differences are shown in Table 3-4.

Table 3-3: Diagnostic LED's

Label	Colour	Function		
F	Red	NET	See Table 3-4	
R	Green	RUN	On	Communication running
			Flashing non-cyclic	Parameter error
			Off	Communication stopped
1	Green	RDY	On	COM ready
			Flashing cyclic	Bootstrap loader active
			Flashing non-cyclic	Hardware or system error
			Off	Hardware error
0	Green	MOD	See Table 3-4	

Table 3-4 LED non-conformance details

DeviceNet Specification				PC3000 implementation
Function	**Colour**	**State**	**Explanation**	
NET	Red	On	Critical link failure	Red LED F On
		Flashing	Connection time out	Red LED F flashing
		Off	Device not powered	Red LED F Off
	Green	On	On-line, link ok	Red LED F Off
		Flashing	On-line, not connected	Red LED F Off
		Off	Device not powered	Red LED F Off
MOD	Red	On	Unrecoverable fault	Green LED 0 Off
		Flashing	Minor fault	Green LED 0 Off
		Off	No power	Green LED 0 Off
	Green	On	Normal operation	Green LED 0 On
		Flashing	Configuration failure	Green LED 0 flashing
		Off	No power	Green LED 0 Off

3.2 PC3000 Software Support

3.2.1 Hardware requirement

The software support for the DeviceNet module requires the installation of a 128K RAM card in the second RAM slot of the LCM-PLUS. This is the rearmost position behind the EPROM card.

IMPORTANT

The downloadable function blocks require a 128K RAM card installed in the LCM-PLUS.

3.2.2 Deliverables

For the DOS PS Tools, the PC3000 function blocks that support this module come in the form of a downloadable function block library on a single 31/2" diskette. The files supplied on the diskette must be copied to the relevant directories on the computer running the DOS PC3000 programming software. The Windows version of the tool WinPS already has the necessary FIELDBUS library to support this module. For WinPS these blocks can also be found in the Xcomms library.

3.2.2.1 PS Tools

If downloadable libraries have not been used before, some initial setting up may be needed as follows. In the following descriptions, <pc3000_home> represents the directory into which the PS Tools were initially installed, usually C:\PC3000.

- Create a directory <pc3000_home>\tpl if it does not already exist.
- Edit the file <pc3000_home>\REL_DEF_ (which should already exist) to add the line "tpl". The file should now contain:-

```
definiti
tpl
```

- Create a directory <pc3000_home>\user\standard and copy the existing files XSYMBOLS.A and XSYMGNU.R into it from <pc3000_home>\user. New versions of these files are included in the downloadable library and this procedure is to provide a back up in case it is necessary to revert to a standard configuration. The delivered files are as follows and the table defines which directories they must be copied to.

Table 3-5: PS downloadable library files

File name	Destination	Description
XSYMBOLS.A	<pc3000_home>\user	Symbol table in ASCII form
FF**.H	<pc3000_home>\user	Individual header files for the function blocks. Used during program build. There should be 26 of these files, see Table 3-6
HEADLIST.H	<pc3000_home>\user	List of function blocks in the downloadable library. For information only.
FBLOCKS.LST	<pc3000_home>	Contains the version number of the library. Used on download to check the compatibility of any previously downloaded library.
FF**.MSG	<pc3000_home>\parhelp	On-line help files for each function block. There should be 26 of these files, see Table 3-6.

File name	Destination	Description
FF**.O	<pc3000_home>\tpl	PS Tools template files for each function block. There should be 26 of these files, see Table 3-6.
XSYMGNU.R	<pc3000_home>\user	Symbols table in pre-compiled form.
LCM_RT_R.RUN	<pc3000_home>	The downloadable run-time library. This is the file that is downloaded to the PC3000.

For the *.msg, *.o and *.h file types, the filenames are:

Table 3-6 List of PS function block file names

File Name	Type Name
FF78	Profi_DPM
FF79	COM_Inf
FF7B	COM_Slv_Sta
FF7D	DevNet_S
FF7F	COM_Slv_Inf
FF83	COM_Table
FF84	COM_Dint
FF85	COM_Real
FF86	COM_Bool
FF87	COM_Str
FF89	COM_SW
FF8A	COM_Dint_8
FF8B	COM_Real_8
FF8C	COM_Dint_D
FF8D	COM_Real_D
FF8E	COM_Bool_D
FF8F	COM_SW_D
FF90	COM_Diag
FF97	COS_Dint
FF98	COS_Real
FF99	COS_Bool
FF9A	COS_Str
FF9B	COS_SW
FF9C	COS_Dint_8
FF9D	COS_Real_8

3.2.3 Library Contents

The PC3000 software support for networks is designed to be functionally compatible with the other supported communication protocols. It comprises driver function blocks, a set of remote variable types (COM_Vars) and a set of slave variable types (COS_Vars).

This document is limited to describing the DeviceNet slave support.

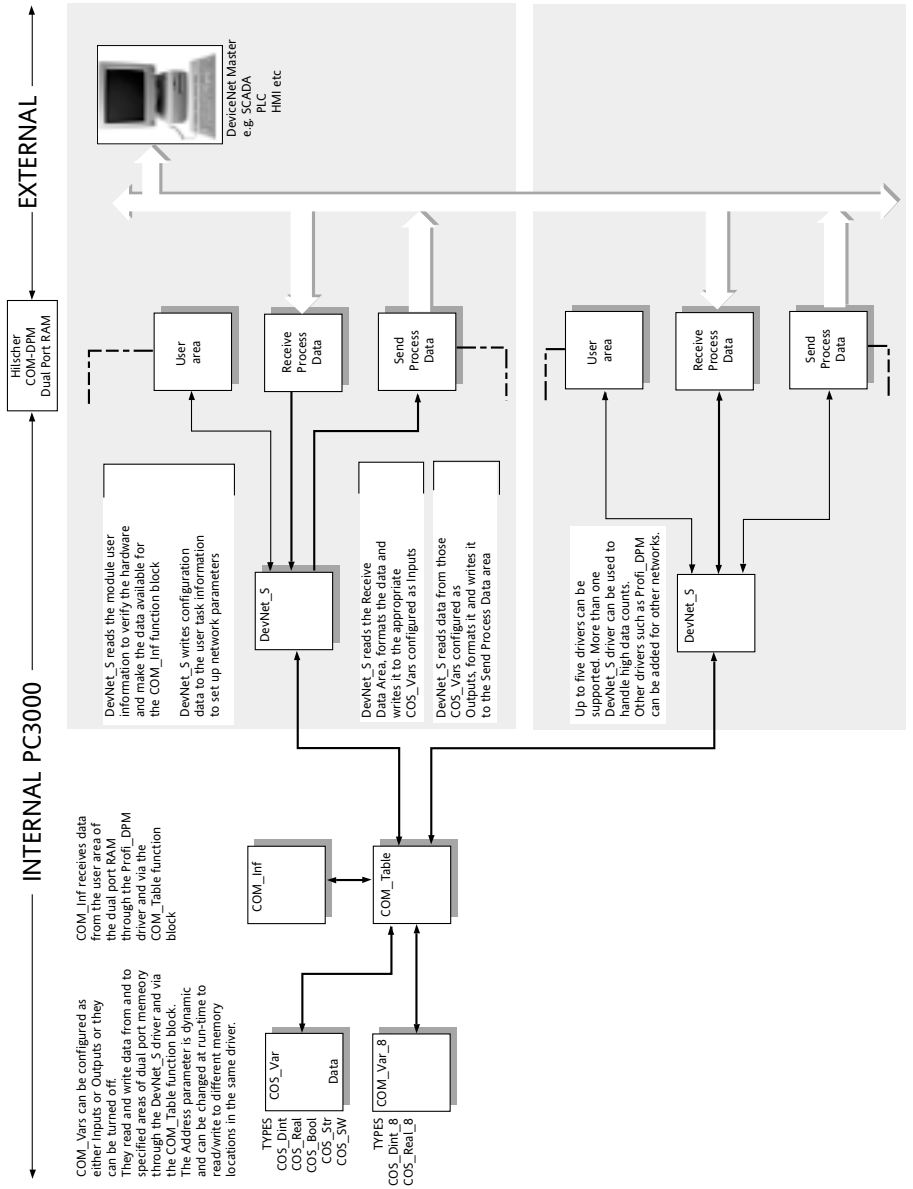
To use the DeviceNet DNS module, it is necessary to instantiate a DevNet_S driver and as many COS_Vars of the relevant types as necessary for the application. These variables can be set to Input, Output or Off. Provided a variable is not set to Off, read and write transactions between these variables and the network are carried out continuously at a rate determined by the driver task.

Because the software is a downloadable library, the hooks between these various function blocks, normally in firmware, must be created by the programmer. This is done by instantiating a COM_Table function block. Only one COM_Table is needed and it requires no configuration or wiring. It has a few output parameters but these are only for verification and diagnostics.

The DNS module provides, in its dual port memory, a set of information parameters. Some of these are used by the DevNet_S (and other drivers) to confirm that the correct module is fitted. Others may be necessary in the event of module failure and subsequent diagnosis. The set of information parameters may be obtained by instantiating a COM_Inf function block. This block obtains its data from the module installed in the specified slot. The block is not required for communications but does supply useful information if support should be required.

3.2.4 Functional Outline

Figure 3-4 System architecture



Once configured, the DNS module handles all of the network communications tasks. It extracts data from the Send Process Data area and transmits them over the network. It accepts data from the network master and places them into the Receive Process Data Area.
The PC3000 COS_Var function blocks read and write to these same Process Data areas. The memory access is controlled by a set of flags to ensure that data is always consistent i.e. all bytes read or written at the same time.
The COS_Var blocks are all registered in a table at the appropriate DevNet_S driver. The Address parameter in each COS_Var specifies which driver to use and where the data is to be found in that driver (i.e. in dual port memory). At each execution of the DevNet_S, the driver services all the COS_Vars that are registered with it.
For a COS_Var configured as an Input, it reads the dual port memory, using the handshake flags to ensure consistency, interprets the data according to the specified format and returns a value to the COS_Var. For a COS_Var configured as an Output, it formats the value and writes it to the specified area of dual port memory. The driver interprets the memory locations (as integer, real, string etc.) according to a specification included in the COM variable address string.

3.2.5 Known bugs and limitations

This release of the DeviceNet driver does not support Explicit Messaging.
There are no known bugs.

3.3 DeviceNet Slave Configuration

The PC3000 function blocks require certain parameters in the DNS module to be configured in a particular way. The module can be configured by two methods.

3.3.1 Configuration by SyCon

The DNS module can be configured using the Hilscher SyCon configuration tool. This runs on a PC running Windows 95, 98, NT4, Me or 2000 and the configuration is downloaded to the module over an RS232 link to the configuration and diagnostic port of the module. The parameters that must be set are to be found under Slave Device Settings and must be as in Table 3-7.

Table 3-7: Compulsory DNS configuration parameters

Parameter	Value
Start up behaviour after system initialisation	Controlled release of the communication by the application program.
Handshake of the process data	Buffered, device controlled.
Autobaud	Tick if required.

Other parameters such as MAC ID, Description and the specification of the input and output data areas (including Consumed size and Produced size) are all to be found under Slave Device Configuration and should be set as required.

IMPORTANT

Configuration data downloaded by SyCon are loaded into flash memory on the DNS module and become the cold start default values, i.e. on a power-up, these are the values that will be used by the module.
Also if a Reset is performed on the DeviceNet_S funtion block, these are the values that will be loaded into run-time memory.

3.3.2 Configuration by PC3000

The DevNet_S function block provides a means of configuring the DNS module without the need to use SyCon although care needs to be taken in the user program to protect the data in the event of a power cycle.

IMPORTANT

Configuration data in the PC3000 is stored in dual port RAM and loaded to the run-time memory of the DNS module only when an Init is performed.
On a power up or Reset of the DNS module, the cold start values, if any have been downloaded by SyCon, will be loaded into run-time memory. The PC3000 user program must, therefore, detect the power up and trigger an Init in order to re-instate the desired configuration.

The function block automatically sets up the required compulsory parameters correctly. Other parameters can be set in the function block are

- MAC ID
- Baud rate
- Device name
- Produced size
- Consumed size

These parameters only take effect after a warm start is performed by triggering the Init parameter of the DevNet_S driver. The Init copies the configuration data from the volatile dual port RAM into the run-time memory of the DNS module. On a power down, this data is lost and, on a subsequent power up, must be reinstated with another Init.

4. Function Block Summary

Table 4-1: Summary of function blocks in the library

Name	ID	Purpose
COM_Table	FF7F	Provides the system interface between any network comms driver and all the other function blocks that use it. This block is compulsory. It does not require configuration or wiring. One block supports up to the maximum of five net work modules.
Profi_DPM	FF78	Profibus DP Master driver. Provides all interfaces to the dual port memory of the DPM module. These include the Process Data interface for all the COM_Var function blocks, common system interface for the COM_Inf function block and the message interface for the COM_Slv_Inf, COM_Slv_Sta and COM_Diag function blocks. One of these blocks must be instantiated for each DPM hardware module present in the rack.
DevNet_S	FF7D	DeviceNet Slave driver. Provides all interfaces to the dual port memory of the DNS module. These include the Process Data interface for all the COS_Var function blocks and the common system interface for the COM_Inf function block. One of these blocks must be instantiated for each DNS hardware module present in the rack.
COM_Inf	FF79	Provides information about any installed COM module. This is useful if support should be required for the module as it includes data such as software version numbers. This block is optional and is not required for correct functioning of the communications interface.
COM_Slv_Inf	FF8F	Extracts from any network master driver (e.g. Profi_DPM) the configuration and on-line information for one network slave at a time. The information is provided by the GSD file for the slave device and by the network configurator. This block is optional.
COM_Slv_Sta	FF7B	Extracts from any network master driver (e.g. Profi_DPM) a summary of the on-line status information for 8 slave devices. Whether the slave is configured, whether it is communicating and whether there is any new diagnostic information available. This block is optional but recommended.

Name	ID	Purpose	
COM_Diag	FF90	For any network master driver (e.g. Profi_DPM), provides detailed slave diagnostic information for one slave device on demand. Includes the standard slave diagnostics and the Extended Diagnostic Data which is specific to each slave device. This block is recommended.	
COM_Dint	FF83	Read /write one integer value	**For use with network master drivers.** These blocks read/write data that are configured in the cyclic data exchange
COM_Real	FF84	Read /write one floating point value	
COM_Bool	FF85	Read /write one digital value	
COM_Str	FF86	Read /write string	
COM_SW	FF87	Read /write one status word	
COM_Dint_8	FF89	Read /write up to 8 integer values	**For use with network master drivers.** These blocks read/write data that are configured in the cyclic data exchange
COM_Real_8	FF8A	Read /write up to 8 floating point values	
COM_Dint_D	FF8E	Read /write one integer value	**For use with network master drivers.** These blocks read/write data that is not configured in the cyclic data exchange
COM_Real_D	FF8D	Read /write one floating point value	
COM_Bool_D	FF8B	Read /write one digital value	
COM_SW_D	FF8C	Read /write one status word	
COS_Dint	FF97	Read /write one integer value	**For use with network slave drivers.** These blocks read/write data that are configured in the cyclic data exchange
COS_Real	FF98	Read /write one floating point value	
COS_Bool	FF99	Read /write one digital value	
COS_Str	FF9A	Read/write a string	
COS_SW	FF9B	Read /write one status word	
COS_Dint_8	FF9C	Read /write up to 8 integer values	
COS_Real_8	FF9D	Read /write up to 8 floating point values	

5. Cyclic Data Exchange

5.1 Introduction

DeviceNet is a fast fieldbus network that provides a simple mechanism for transferring blocks of memory between a master device and, up to, 63 slave devices. DeviceNet is not concerned with the details of data type and format. Each slave can 'consume' up to 255 bytes of input data and 'produce' 255 bytes of output data. The input and output data are transferred between the slave and the master on a cyclic basis and at a rate determined by the network configuration.
The slave is responsible for the details of produced and consumed size, data types and format. The master must know this information in order to correctly interpret the data supplied by the slave and to correctly format the data supplied to the slave.

5.2 Master Configuration

The PC3000 acting as a slave will be connected to a master device, which may be a SCADA system, a PLC, an HMI or some other DeviceNet master. That master will need to be configured using whatever tools are appropriate.
Information about the PC3000 DeviceNet Slave is imported into the master configuration tool in the form of an EDS file (electronic data sheet). The EDS file supplied with the PC3000 treats the PC3000 as modular slave and allows the network configurer to assign produced and consumed memory in blocks of 8 bytes, up to a maximum of 248 bytes produced and 248 bytes consumed.

IMPORTANT

The produced and consumed sizes specified during the configuration of the master must match the configuration of the slave. If this is not true, the master will report a configuration error and communication cannot be established.
For this reason, the DevNet_S function block parameters, InSize and OutSize, must be multiples of 8 (to conform with the EDS file).

5.3 PC3000 Configuration

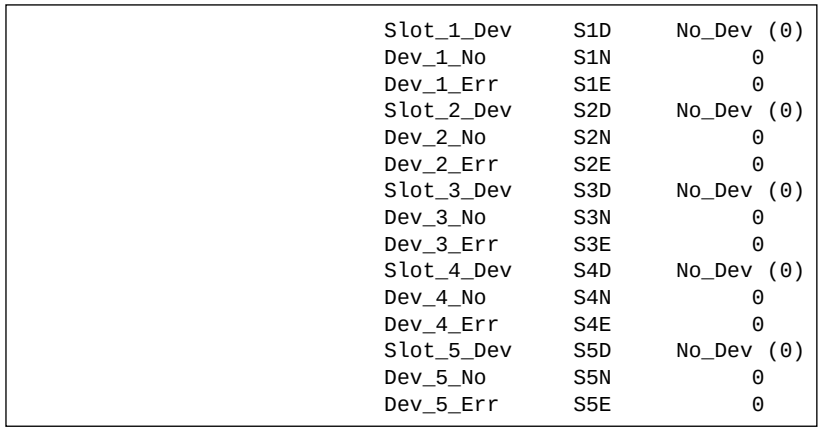
5.3.1 Compulsory function blocks

A PC3000 program with any network capability must have one COM_Table block instantiated to support the network drivers. One COM_Table will support up to the maximum of five drivers allowed. Each DeviceNet Slave module (DNS) installed in the rack requires a DevNet_S function block driver to support it.
With these two elements in place, communications can be established between a network master and the PC3000 slave. COS_Vars are then required in order to extract the data from the driver.

5.3.1.1 COM_Table

One, and only one, COM_Table function block is required to provide system support. Up to five network communications boards can be installed in a PC3000 rack and on COM_table will support all five. No configuration or user wiring is required.

Figure 5-1: COM_Table function block



5.3.1.2 DevNet_S

One DevNet_S function block is required for each DNS module installed in the rack (up to five maximum).

- Assign the driver to an appropriate task. Execution time of the driver increases with increasing numbers of slave variables registered with the driver. Whilst it is possible to service a few variables with the driver on a 10ms task, it is likely that, for systems of reasonable size, a slower task will be required.
- Set {Slot_No} to correspond to the position of the DNS module in the rack.
- Set {ErrAction} to the required value depending on how the system is to behave in the event of a comms failure.

Table 5-1: DevNet_S ErrAction parameter values

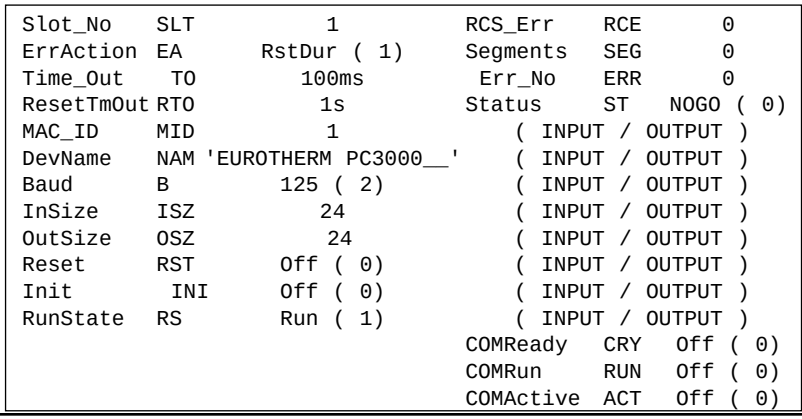
Value	Enumeration	Action
0	None	No action is taken by the driver. Input COS_Vars retain their last known values during the comms failure and are updated as soon as comms is re-established. The user program can detect comms failure from the COMActive flag and take specific action, if required.
1	RstDur	Reset for the duration. As soon as the block detects comms failure (COMActive = Off), it sets all input values to 0 and {RunState} to Off. This will set any outputs and other parameters that are wired to the input COS_Vars to 0 or Off. As soon as comms is re-established, {RunState} is set to Run, the input values are updated by the net work and control resumes normally.
2	RstWait	Reset and wait for program. As soon as the block detects comms failure (COMActive = Off), it sets all input values to 0 and {RunState} to Off. This will set any outputs and other parameters that are wired to the input COS_Vars to 0 or Off. This state is retained when comms is re-established. The user program must set {RunState} to Run before the input values are updated and control resumed.

- If the DNS module has been configured through the serial configuration and diagnostic port, it is not necessary to assign values to {MAC_ID}, {DevName}, {Baud}, {InSize} or {OutSize}. The DNS module will operate according to the cold start values. The cold start values will be copied from the flash memory of the DNS module into the DevNet_S function block if the driver is {Reset}.
- If the DNS module has not been configured through the serial configuration and diagnostic port, the following values must be assigned. Before they take effect, they must be transferred to the run-time memory of the DNS module by performing an {Init}. This would normally be executed in an SFC step at the beginning of the program but also needs to be done again on any power-up.

- Set {MAC_ID} to be the slave address
- Set {DevName} (optional)
- Set {Baud}
- Set {InSize}. This is the size, in bytes, of the Consumed Data and must match the value specified in the configuration of the network master. If the standard EDS file (PC3KDNS.EDS) is used, the value must be a multiple of eight.
- Set {OutSize}. This is the size, in bytes, of the Produced Data and must match the value specified in the configuration of the network master. If the standard EDS file (PC3KDNS.EDS) is used, the value must be a multiple of eight.

- The two timeout values, {Time_Out} and {ResetTmOut} should be changed to reflect the task to which the driver has been assigned. i.e. if a slower task than the default is used, the timeouts should be increased proportionately.
- If the DNS module has been configured through the serial configuration and diagnostic port, arrange the SFC to trigger {Reset} during startup. This is not strictly necessary because the DNS module will start working with its pre-configured setup but this setup will not be reflected in the values of {MAC_ID} etc. until a {Reset} is performed.
- If the DNS module has not been configured through the serial configuration and diagnostic port, the SFC must be set up to trigger an {Init} during startup. This is necessary to transfer the configuration to run-time memory.
- Under normal circumstances, {RunState} should be set to Run. It will automatically go to Off if comms fails and {ErrAction} has been specified. Also, the user or the program can turn it off. This does not stop the network comms (i.e. there will be no message at the master) but it does prevent the driver from updating the COS_Vars and, therefore, allows ‘off-line’ testing.

Figure 5-2: DevNet_S function block



5.3.1.3 COS_Vars

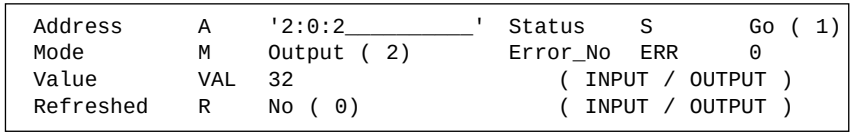
COS_Vars are function blocks, which read and write the data from the DevNet_S driver. There are a number of function blocks of different data types. The collection of instantiated blocks effectively defines the memory format of the slave because each COS_Var is associated with a particular area of the DeviceNet memory. Each COS_Var can be registered as an Input, in which case it reads data from the DeviceNet Consumed memory area, or as an Output, in which case it writes data to the DeviceNet Produced memory area. The location and size of the associated memory area is specified in the COS_Var {Address} parameter.

Table 5-2: COS_Var variable types

Block type	Parameter type	Max value	Min value	DeviceNet memory usage (bytes)
COS_Dint	INTEGER	2147483646	-2147483647	1, 2 or 4
COS_Real	REAL	3.40282E+38	-3.40282E+38	1, 2, 4 or 8
COS_Bool	BOOL	1 (On)	0 (Off)	1 bit
COS_Str	STRING			≤ 255
COS_SW	16 off BOOL Value_0 to Value_15 1 off INTEGER Value	1 (On) 65535	0(Off) 0	1 or 2
COS_Dint_8	≤ 8 off INTEGER	2147483646	-2147483647	≤ 8 x (1, 2 or 4)
COS_Real_8	≤ 8 off REAL	3.40282E+38	-3.40282E+38	≤ 8 x (1, 2, 4 or 8)

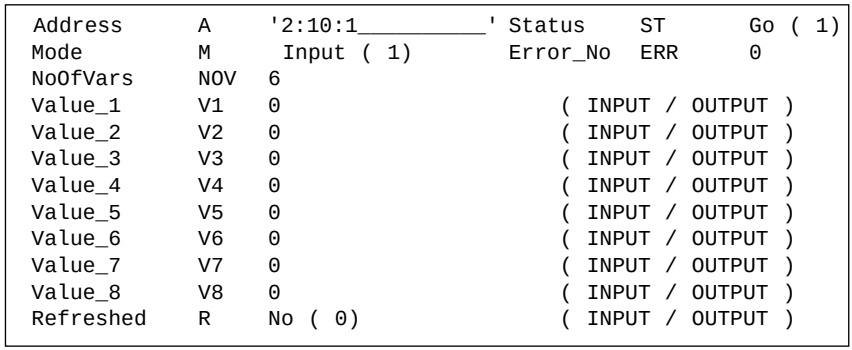
The {Address} parameter must be configured to read/write the correct number of bytes from the appropriate driver at the correct memory offset. For details of the syntax of the {Address} parameter, see the Function Block Reference.
The {Mode} parameter must be set as either Input (Consumed) or Output (Produced). It can also be set to Off which prevents the driver from updating this block or being updated by it.
If {Mode} is Output, {Value} can be wired or written to by the SFC. If {Mode} is Input, {Value} is effectively Read Only. If {Mode} is Off, {Value} can be written which is useful for forcing values during commissioning and for testing purposes.
Figure 5-3 shows a COS_Dint configured to write two bytes at offset 0 to the DNS module in slot 2.

Figure 5-3: COS_Dint function block



The COS_Dint_8 and COS_Real_8 function blocks each provide up to eight values, with identical specification and format, taken from consecutive memory locations. The {NoOfVars} parameter can limit the number of parameters written or read to less than eight. This enables the block to be used up to the memory limits without causing memory overrun errors.
Figure 5-4 shows a COS_Dint_8 configured to read 6 values from the DNS module installed in slot 2. All 6 values are one byte long and the first is at offset 10. The second is at offset 11, the third at offset 12 etc.

Figure 5-4: COS_Dint_8 function block

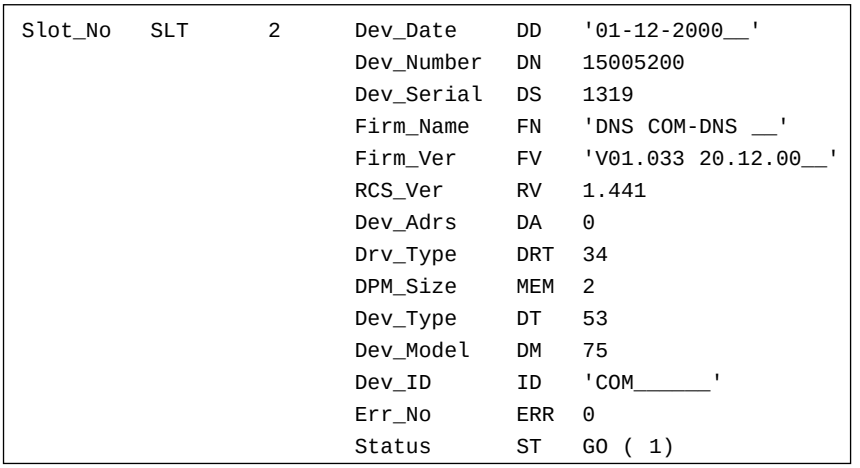


5.3.2 Optional Function Blocks

5.3.2.1 COM_Inf

This block provides information about the DNS master module itself. It provides confirmation that the module is the correct module, that its firmware and operating system are current and that it has been correctly configured.

Figure 5-5: COM_Inf function block



5.4 Example Configuration

An 8 loop controller is to communicate, as a DeviceNet slave, with a host device which will send :-

- active setpoints
- high alarm limits
- low alarm limits
- auto/manual flag for each loop (in the form of a status word)
- output power when in Manual

and receive :-

- process value
- actual output power
- alarm status word

Input and output memory is declared in blocks of 8 bytes as specified by the Electronic Data Sheet supplied with the module. The values can be passed in a number of different formats but, for the example, all real numbers will be passed as 16 bit (2 byte) integers which have been scaled by a factor of ten in order to provide one decimal place. The DeviceNet memory map might, therefore, be as shown in Table 5-3.

Table 5-3: Example memory map

Consumed Offset	Value		Produced Offset	Value
0	SP 1	Blocks 1 & 2	0	PV 1
2	SP 2		2	PV 2
...	...		...	...
14	SP 8		14	PV 8
16	OP1	Blocks 3 & 4	16	OP 1
18	OP2		18	OP 2
...	...		...	...
30	OP 8		30	OP 8
32	AL Hi 1	Block 5	32	Alm SW
34	AL Hi 2		34	Not used
36	AL Hi 3		36	Not used
38	AL Hi 4		38	Not used
...	...	Block 6		
46	AL Hi 8			
48	AL Lo 1	Blocks 7 & 8		
50	AL Lo 2			
...	...			
62	AL Lo 8			
64	AutoMan SW	Block 9		
66	Not used			
68	Not used			
70	Not used			

Table 5-4 lists the comms related function blocks that would be necessary to implement this scheme.

Table 5-4: Example function block list

Type	Name (say)	Configuration	Description
COM_Table	Table	No config	Network resource
DevNet_S	DNS	Slot number (1 say)	DeviceNet slave driver
COS_Real_8	SP	Input 1:0:2E1	8 off setpoints
COS_Real_8	PV	Output 1:0:2E1	8 off process values
COS_Real_8	OP_Out	Output 1:16:2E1	8 off actual outputs
COS_Real_8	OP_In	Input 1:16:2E1	8 off manual outputs
COS_Real_8	ALH	Input 1:32:2E1	8 off alarm limit hi
COS_Real_8	ALL	Input 1:48:2E1	8 off alarm limit lo
COS_SW	AM	Input 1:64:1	8 off auto/manual flags
COS_SW	AL	Output 1:32:2 8 off low alarm flags	8 off high alarm flags

Figure 5-6: Example function block wiring

```
(* Make PID input connections.  Output wiring depends on Auto/Manual
mode *)
PID1.Setpoint     := SP.Value_1 ;
PID1.Manual       := AM.Value_0 ;
PID1.Output       := SEL_REAL(G :=PID1.Manual,IN0 :=PID1.Output,IN1
                              :=OP_In.Value_1);
PID2.Setpoint     := SP.Value_2 ;
PID2.Manual       := AM.Value_1 ;
PID2.Output       := SEL_REAL(G :=PID2.Manual,IN0 :=PID2.Output,IN1
                  :=OP_In.Value_2);
                  (* …… and so on *)
PID8.Setpoint     := SP.Value_8 ;
PID8.Manual       := AM.Value_7 ;
PID8.Output       := SEL_REAL(G :=PID8.Manual,IN0 :=PID8.Output,IN1
                  :=OP_In.Value_8);

(* Wire the high alarm output flags in the low byte of the AL status
word *)
AL.Value_0        := PID1.Process_Val > ALH.Value_1 ;
AL.Value_1        := PID2.Process_Val > ALH.Value_2 ;
(* …… and so on *)
AL.Value_7        := PID8.Process_Val > ALH.Value_8 ;

(* Wire the low alarm output flags in the high byte of the AL status
word *)
AL.Value_8        := PID1.Process_Val < ALL.Value_1 ;
AL.Value_9        := PID2.Process_Val < ALL.Value_2 ;
(* …… and so on *)
AL.Value_15       := PID8.Process_Val < ALL.Value_8 ;


(* Wire the actual process values *)
PV.Value_1        := PID1.Process_Val ;
PV.Value_2        := PID2.Process_Val ;
(* …… and so on *)
PV.Value_8        := PID8.Process_Val ;


(* Wire the actual output power values *)
OP_Out.Value_1    := PID1.Output ;
OP_Out.Value_2    := PID2.Output ;
(* …… and so on *)
OP_Out.Value_8    := PID8.Output ;
```

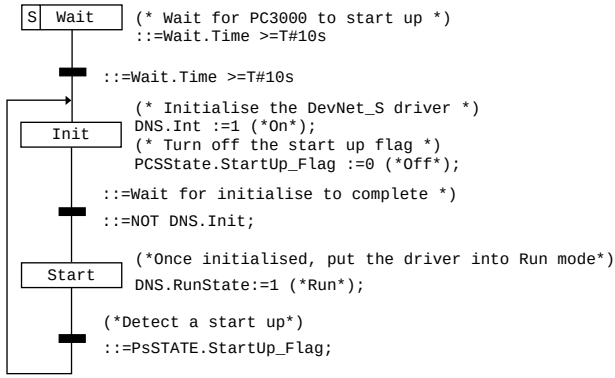
The DevNet_S driver will be configured in a similar way to Figure 5-7, certainly with respect to InSize and OutSize, remembering that {InSize} and {OutSize} must be multiples of eight.

Figure 5-7: DevNet_S configuration

Slot_No	SLT	2	RCS_Err	RCE	0
ErrAction	EA	None (0)	Segments	SEG	6
Time_Out	TO	100ms	Err_No	ERR	0
ResetTmOut	RTO	2s	Status	ST	GO(1)
MAC_ID	MID	1	(INPUT / OUTPUT)		
DevName	NAM	'Example Program___'(INPUT / OUTPUT)			
Baud	B	125(2)	(INPUT / OUTPUT)		
InSize	ISZ	72	(INPUT / OUTPUT)		
OutSize	OSZ	40	(INPUT / OUTPUT)		
Reset	RST	Off (0)	(INPUT / OUTPUT)		
Init	INI	Off (0)	(INPUT / OUTPUT)		
RunState	RS	Run (1)	(INPUT / OUTPUT)		
			COMReady	CRY	On(1)
			COMRun	RUN	On(1)
			COMActive	ACT	Off(0)

The SFC will include, as part of the start up sequence, the initialisation of the DevNet_S driver. Also, a power-up must be detected and the driver re-initialised. A piece of SFC similar to Figure 5-8 would serve this pupose.

Figure 5-8: SFC initialisation sequence



6. Glossary of Terms and References

6.1 PC3000 terms

BOOL — A two state parameter. Usually either On or Off but may have other 'senses'. e.g. Yes/No or True/False.

Cold start value — The value assigned to a parameter at program generation. All parameters have default cold start values that are assigned when a function block is created. These values can be changed by the programmer. At run time, when the PC3000 executes a cold start, all parameters are assigned their cold start values.

Downloadable — A function block that is not present in the PC3000 firmware. These blocks are compiled as part of the user program and downloaded at that time. They are separate from the user program, however, in that they are loaded to the last RAM card (the 3rd in version 1 LCMs and the 2nd in version 2 LCMs and LCM-PLUSs).

ENUMERATED — An integer parameter, with a limited number of valid values, for which each value is represented by a text string.

Input — A parameter that can be written to by the PS, the SFC or by wiring.

Input/Output — A parameter that can be written to by the PS or the SFC and which the block itself can also change. It usually can not be wired.

INTEGER — A parameter that can store any whole number. Range is -2147483648 to +2147483647.

Microcell — PC3000 programming and configuration software that runs on a PC in an OS/2 environment. Also provides simple SCADA functions.

Non-wirable — A parameter that can not be the destination of a wiring statement. These parameters do not appear on the Wiring Edit screen. However, non-wirable inputs (which are often input/outputs) may be written to by the SFC.

Output — A parameter which can only be written to by the block itself. It can be read by the PS, the SFC or wiring.

PS — PC3000 Programming Software that runs on a PC in a DOS environment or in a DOS window under Windows 3.1, Windows 95, Windows 98, Windows NT or OS/2. Has no SCADA functionality.

REAL — A parameter that can store any real number including decimal parts. Maximum range is $\pm 3.4 * 10^{38}$.

SFC — Sequential Function Chart. The graphical language used to describe sequential logic.

ST — Structured Text. The text language used for wiring and within SFC steps and transtitions.

STRING	A parameter consisting of between one and 255 bytes. Usually used for text messages but can also be used as a data array.
Task	A function block executes at regular intervals determined by the task to which it is assigned. Between two and seven tasks may be present in an application program with execution rates of between 5ms and several minutes.
Win PS	PC3000 Programming Software that runs on a PC in a Windows environment under Windows 98, Windows NT or Windows 2000.
Wirable	A parameter that can be the destination of a wiring statement. These parameters appear on the Wiring Edit screen for the function block.
Wiring	The connection between function block parameters which is executed continuously at the same rate as the destination function block.

6.2 DeviceNet terms

EDS	Electronic Data Sheet. A text file which contains information about a DeviceNet slave that the master will use when it configures the network. The one issued with the PC3000 DNS module is PC3KDNS.EDS.
Cyclic data exchange	The mechanism whereby the specified Input data is read from all the slaves and the specified Output data is written to all the slaves.
Produced data	Data in the slave that comes from device inputs, or similar parameters, and is read by the master, e.g. process value. Date produced by a slave is consumed at the master.
Consumed data	Data in the slave that is written by the master and copied to device outputs or similar parameters, e.g. controller setpoint. Data produced by the master is consumed at the slave.
Receive Process Data	The area of dual port memory in the slave that contains data produced by the master and, hence, consumed by the slave. Values are written here by the master on every cyclic data exchange.
Send Process Data	The area of dual port memory in the slave that contains data produced by the slave and, hence, consumed by the master. The master reads this data on every cyclic data exchange.

6.3 References

6.3.1 Reference 1

Hilscher GmbH Manufacturer of Fieldbus interface cards and modules.
Hilscher Gesellschaft für Systemautomation mbH
Rheinstraße 78
D-65795 Hattersheim
Germany
Tel: +49 (0) 6190/9907-0
Fax: +49 (0) 6190/9907-50
Hotline:+49 (0) 6190/9907-99 or e-mail hotline@hilscher.com
web: www.hilscher.com

6.3.2 Reference 2

SyCon	System Configuration software for the Hilscher modules. Runs on a PC under Windows 95/98 and is used to con figure fieldbus networks. Supplied by Hilscher GmbH, see 6.3.1.

6.3.3 Reference 3

Function Block Reference	Eurotherm document, HA027900 Fieldbus on PC3000, Function Block Reference

6.3.4 Reference 4

General definitions	Hilscher document Toolkit Manual, General Definitions

6.3.5 Reference 5

Protocol interface manual	Hilscher documents Protocol Interface Manual, DeviceNet Slave

INTERNATIONAL SALES AND SERVICE

AUSTRALIA
Eurotherm Pty. Ltd.
Telephone Sydney (+61 2) 96348444
Fax (+61 2) 96348555

AUSTRIA
Eurotherm GmbH
Telephone Vienna (+43 1) 7987601
Fax (+43 1) 7987605

BELGIUM
Eurotherm B.V.
Telephone Antwerp (+32) 85 274080
Fax (+32) 85 274081

BRAZIL
Ero Electronic do Brasil Ind. e Com Ltda.
Telephone (+19) 3237 3413
Fax (+19) 3234 7050

DENMARK
Eurotherm Danmark A/S
Telephone Copenhagen (+45 70) 234670
Fax (+45 70) 234660

FINLAND
Eurotherm Finland
Telephone (+358) 22506030
Fax (+358) 22503201

FRANCE
Eurotherm Automation SA
Telephone Lyon (+33 478) 664500
Fax (+33 478) 352490

GERMANY
Eurotherm Deutschland GmbH
Telephone Limburg (+49 6431) 2980
Fax (+49 6431) 298119
Also regional offices

HONG KONG
Eurotherm Limited
Telephone Hong Kong (+852) 28733826
Fax (+852) 28700148
Telex 0802 69257 EIFEL HX

INDIA
Eurotherm India Limited
Telephone Chennai (+9144) 4961129
Fax (+9144) 4961831

IRELAND
Eurotherm Ireland Limited
Telephone Naas (+353 45) 879937
Fax (+353 45) 875123

ITALY
Eurotherm S.r.l
Telephone Como (+39 31) 975111
Fax (+39 31) 977512
Telex 380893 EUROTH I

JAPAN
Densei-Lambda K.K.
Eurotherm Division
Telephone Tokyo (+81 3) 5714 0620
Fax (+81 3) 5714 0621

KOREA
Eurotherm Korea Limited
Telephone Seoul (+82 31) 2868507
Fax (+82 31) 2878508

NETHERLANDS
Eurotherm B.V.
Telephone Alphen a/d Ryn (+31 172) 411752
Fax (+31 172) 417260

NORWAY
Eurotherm A/S
Telephone Oslo (+47 67) 592170
Fax (+47 67) 118301

SPAIN
Eurotherm España SA
Telephone (+34 91) 6616001
Fax (+34 91) 6619093

SWEDEN
Eurotherm AB
Telephone Malmo (+46 40) 384500
Fax (+46 40) 384545

SWITZERLAND
Eurotherm Produkte (Schweiz) AG
Telephone (+41 55) 4154400
Fax (+41 55) 4154415

UNITED KINGDOM
Eurotherm Limited
CONTROLS and DATA MANAGEMENT
Telephone Worthing (+44 1903) 695888
Fax (+44 1903) 695666
PROCESS AUTOMATION
Telephone Worthing (+44 1903) 205277
Fax (+44 1903) 236465

U.S.A
Eurotherm Inc.
Telephone Leesburg (+1 703) 443 0000
Fax (+1 703) 669 1300
Web www.eurotherm.com

http://www.eurotherm.co.uk

ED 29

CE

ENG

HA027903