PC3000

invensys
EUROTHERM

ENG

# Fieldbus library
# function block supplement

## CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

## 1. Scope

This document describes the function blocks available for ProfibusDP master and DeviceNet slave support on the PC3000. These blocks are all included in a single downloadable Fieldbus library that contains all network function blocks.
There is no support in these blocks for Explicit Messaging and the hardware only supports polled mode for data transfer.
For hardware and software installation information, and application notes refer to the following documentation.

## 2. Related Documents

| | Title | Document Number | Revision and date |
|---|---|---|---|
| [1] | Profibus Module installation guide | HA027826 | 1 |
| [2] | DeviceNet Module Installation guide | HA027827 | 1 |
| [3] | Profibus on PC3000. User Manual | HA027902 | 1 |
| [4] | DeviceNet on PC3000. User Manual | HA027903 | 1 |

## 3. HCOM_DRVS

This class contains the currently supported Fieldbus driver function block types:

Table 3-1: Function blocks in HCOM_DRVS class

| Function block name | Description |
|---|---|
| Profi_DPM | Profibus DP master driver |
| COM_Inf | Provides diagnostic information about any COM module |
| COM_Slv_Sta | Provides on-line status information about 8 slave instruments. |
| DevNet_S | DeviceNet slave drive |
| COM_Table | Support table for the Hilscher COM range of communication modules |
| COM_Slv_Inf | Provides information about a configured slave instrument. |
| COM_Diag | Provides access to slave diagnostics. |

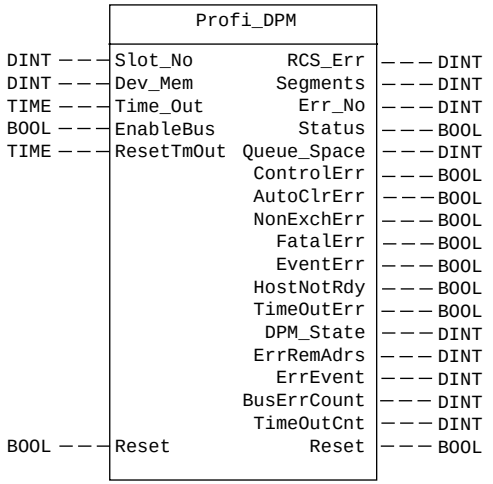
# 4. Profi_DPM FF78

## 4.1 Functional Description

A Profibus DP Master comms driver. It requires a Profibus DPM module to be resent in one of the first five rack slots and it provides the data formatting and read/write routines for any number of COM_Var function blocks.
One COM_Table function block is required to service this and any other similar comms driver. Only one COM_Table per PC3000 program is needed because it will support up to the a maximum of 5 drivers.
Information about the Hilscher module that is not available in this function block may be obtained by instantiating a COM_Inf function block with the same slot number.
The PC3000 Profibus DPM module uses a Hilscher COM-DPM module which must be configured using Hilscher Sycon PB/E software (see 18.4.2). Certain system settings that need to be made are documented in Reference 3, paragraph 18.4.3. on page 118.

## 4.2 Parameter Diagram

Figure 4-1: Profi_DPM block diagram



## 4.3 Parameter Descriptions

### 4.3.1 Slot_No

Wirable Input INTEGER. Max: 5, Min: 1
The rack slot number into which the module is installed. If more than one driver is given the same slot number, a driver error will be generated.

### 4.3.2 Dev_Mem

Wirable Input ENUMERATED INTEGER. Max: 6, Min: 0
Select the relevant setting for the installed module. Only 2K(DPM) and 8K(PB) modules are currently available for Profibus DP Master.

Table 4-1: Profi_DPM Dev_Mem enumerations

| Value | Enumeration |
|---|---|
| 0 | DPM_1K |
| 1 | DPM_2K |
| 2 | DPM_4K |
| 3 | DPM_8K |
| 4 | DPM_16K |
| 5 | DPM_32K |
| 6 | DPM_64K |

### 4.3.3 Time_Out

Wirable Input TIME. Max: 2147483647, Min: 0
Access to the dual port memory on the Hilscher module is controlled by a number of semaphore flags. If a response is not received from the module within the {Time_Out} period, an error is returned.

### 4.3.4 RCS_Err

Nonwirable Output INTEGER. Max: 255, Min: 0
The Hilscher COM-DPM operating system error byte as read from the dual port memory. A non-zero value indicates a module initialisation error. Consult technical suport or the Hilscher hotline.
If an error value and description is given below, it is taken from the Sycon on-line help file (see 18.4.2).

Table 4-2: Profi_DPM RCS_Err values

| RCS_Err Number | Description |
|---|---|
| 0 | No error |
| 4 | Task does not exist |
| 5 | Task is not initialised |
| 6 | The MCL is locked |
| 7 | The MCL rejects a send command because of an error |
| 20 | Data base not configured |
| 21 | Data base segment not configured or doesn't exist |
| 22 | Number for message wrong during download |
| 23 | Received number of data during download does not match to that in the command message |
| 24 | Sequence identifier wrong during download |
| 25 | Checksum after download and checksum in command message does not match |
| 26 | Write/Read access of data base segment |
| 27 | Download/Upload or erase of configured data base type is not allowed |
| 28 | The state of the data base segment indicated an error. Upload not possible |
| 29 | The access to the data base segment needs the boot strap loader. The boot strap loader is not present |
| 30 | Trace buffer overflow |
| 31 | Entry into trace buffer too long |
| 37 | No or wrong licence. The OEM licence of the system configurator allows only communication to devices that have the same licence inside |
| 38 | The data base created by the system configurator and the data base expected by the firmware is not compatible |
| 39 | DBM module missing |
| 40 | No command free |
| 41 | Command unknown |
| 42 | Command mode unknown |
| 43 | Wrong parameter in the command |
| 44 | Message length does not match to the parameters of the command |
| 45 | Only a MCL does use this command to the RCS |
| 50 | FLASH occupied at the moment |

| RCS_Err Number | Description |
|---|---|
| 51 | Error deleting the FLASH |
| 52 | Error writing the FLASH |
| 53 | FLASH not configured |
| 54 | FLASH timeout error |
| 55 | Access protection error while deleting the FLASH |
| 56 | FLASH size does not match or not enough FLASH |
| | memory |
| 60 | Wrong structure type |
| 61 | Wrong length of structure |
| 62 | Structure does not exist |
| 70 | No clock on the device |
| 80 | Wrong handle for the table (table does not exist) |
| 81 | Data length does not match the structure of this table |
| 82 | The data set of this number does not exist |
| 83 | This table name does not exist |
| 84 | Table full. No more entries allowed |
| 85 | Other error from DBM |
| 90 | The device info (serial number, device number and date) does already exist |
| 91 | Licence code invalid |
| 92 | Licence code does already exist |
| 93 | All memory locations for licence codes already in use |

### 4.3.5 Segments

Nonwirable Output INTEGER. Max: 2147483647, Min: 0
Messages between the LCM and the Hilscher module are buffered in segments. The number of segments remaining is read from the module and displayed in this parameter. It is read from the dual port memory Operating System Information as the SegmentCount parameter.

### 4.3.6 Err_No

Nonwirable Output INTEGER. Max: 2147483647, Min: 0
Indicates the cause of any error in the driver installation.

Table 4-3: Profi_DPM errors

| Err_No | Error | Description |
|---|---|---|
| 0 | NO_ERROR | Module is installed and operating normally. |
| 40 | ERROR_MODULE_NOT_PROFI_M | The module detected at the specified {Slot_No} is not a Profibus DP Master. |

| Err_No | Error | Description |
|---|---|---|
| 42 | ERROR_MODULE_MEMORY_INCORRECT | The module size as read from the dual port memory does not match that specified in {Dev_Mem}. |
| 43 | ERROR_MODULE_DEV_NOT_COM | The installed Hilscher module type is not a COM. |
| 44 | ERROR_MODULE_DEV_NOT_DPM | The installed Hilscher module model is not a DPM. |
| 45 | ERROR_MODULE_ID_NOT_COM | The installed Hilscher device ID is not COM. |
| 46 | ERROR_MODULE_NOT_READY | The module Ready flag is not set. |
| 47 | ERROR_MODULE_WDOG_FAIL | A module watchdog failure has been detected. Not currently supported. |
| 48 | ERROR_SLOT_ALREADY_OCCUPIED | The {Slot_No} specified already has a module installed. |
| 49 | ERROR_INIT_WRITE_FAIL_TEST_ID | A dual port memory fault was detected during initialisation. |
| 50 | ERROR_INIT_WRITE_FAIL_ORIG_ID | A dual port memory fault was detected during initialisation. |
| 51 | ERROR_NO_COM_TABLE_INSTANTIATED | No instance of a COM_Table function block was found during initialisation. This block is mandatory. |
| 63 | ERROR_TIMEOUT_WAITING_FOR_READY_FLAG | A reset has been requested and the PC3000 has tried to put the module into an OPERATE state but the Ready flag is not set. |
| 64 | ERROR_TIMEOUT_WAITING_FOR_RESET_FLAG_TO_CLEAR | A reset has been requested and actioned but the module is not re-setting its Reset flag. |
| 65 | ERROR_TIMEOUT_WAITING_FOR_MODE_OPERATE | A reset has been requested but the DPM module will not go to OPERATE mode. |
| 70 | ERROR_DO_CONFIG_UNKNOWN_STATE | The state engine controlling the configuration software got into an invalid state. The state has been reset, the configuration may not be complete. |

### 4.3.7 Status

Nonwirable Output BOOL

This parameter is set to GO (1) if {Err_No} is zero. Otherwise, this parameter is set to NOGO (0).

### 4.3.8 Queue_Space

Nonwirable Output INTEGER. Max: 100, (1024 in Xcomms libaray)
Min: 0

Requests received from COM_Vars, are queued while awaiting action. The queue allows up to 100 requests to be outstanding. This parameter indicates the space remaining in the queue.

If it drops to 0, the comms driver is overloaded. Either speed up the task rate for the driver or slow down the update rate of the COM_Vars.

Reading and writing parameters which are pre-configured as part of the data exchange on the network should not cause problems with the queue space because the driver only reads and writes to the dual port memory on the DPM module. Use of the demand data protocol, using function blocks such as COM_Dint_D, may cause queue space problems because this protocol does initiate transactions over the network with the time taken for slave instruments to respond being significant. If the demand data function blocks are configured to use the lock out feature, the queue space will not change because other requests are rejected and not queued. It will, however, result in a delayed response time at the requesting COM_Var(s).

### 4.3.9 EnableBus

Wirable Input BOOL. HostFlags: NotRdy

This parameter turns the bus communication task on the Hilscher module on and off. The start up behaviour of the module must have been configured, using Sycon PB/E (see 18.4.2), to be ‘Controlled Release by the Application’.

While the bus is disabled, any request to the driver will result in an error 60 being returned.

### 4.3.10 Reset

Nonwirable Input BOOL. HostFlags: Reset

Setting this parameter to On will perform a hard reset of the Hilscher module. Parameters are reloaded from flash memory and the module is re-initialised. The parameter is reset to Off automatically when the DPM module is running again. If {EnableBus} is On, comms requests are held until the {Reset} parameter is reset to Off.

### 4.3.11 ControlErr

Nonwirable Output BOOL. DPM Task2State: Global_Bits: Bit 0(CTRL) CONTROL-ERROR. A DPM parameterisation error which is serious enough to prevent the module control program running.
For more information see the Hilscher Protocol Interface Manual, ProfibusDP Master.

### 4.3.12 AutoClrErr

Nonwirable Output BOOL. DPM Task2State: Global_Bits: Bit 1(ACLR) AUTO-CLEAR-ERROR. The DPM has stopped communication to all slaves and reached the autoclear end state.
For more information see the Hilscher Protocol Interface Manual, ProfibusDP Master.

### 4.3.13 NonExchErr

Nonwirable Output BOOL. DPM Task2State: Global_Bits: Bit 2(NEXC) NON-EXCHANGE-ERROR. At least one slave has not reached the data exchange state and no process data is being exchanged with it. The address of the first faulty slave will be reported in (ErrRemAdrs}.
For more information see the Hilscher Protocol Interface Manual, ProfibusDP Master.

### 4.3.14 FatalErr

Nonwirable Output BOOL. DPM Task2State: Global_Bits: Bit 3(FAT) FATAL-ERROR. Because of heavy bus error, no further bus communication is possible.
For more information see the Hilscher Protocol Interface Manual, ProfibusDP Master.

### 4.3.15 EventErr

Nonwirable Output BOOL. DPM Task2State: Global_Bits: Bit 4(EVE) EVENT-ERROR. The DPM has detected bus short circuits. The number of detected events is reported in {BusErrCount}. This bit will be set when the first event is detected and will only be reset by a module reset.
For more information see the Hilscher Protocol Interface Manual, ProfibusDP Master.

### 4.3.16 HostNotRdy

Nonwirable Output BOOL. DPM Task2State: Global_Bits: Bit 5(NRDY) HOST-NOT-READY-NOTIFICATION. Indicates if the PC3000 has signalledready. This bit will be set if {EnableBus} is Off.
For more information see the Hilscher Protocol Interface Manual, ProfibusDP Master.

### 4.3.17 TimeOutErr

Nonwirable Output BOOL. DPM Task2State: Global_Bits: Bit 6(TOUT) TIMEOUT-ERROR. The DPM has detected an overrun of the control program because of rejected telegrams. It is an indication of bus short circuits when the DPM has interrupted the communication.
The number of detected timeouts is reported in {BusErrCount}. This bit will be set when the first timeout is detected and will only be reset by a module reset.
For more information see the Hilscher Protocol Interface Manual, ProfibusDP Master.

### 4.3.18 DPM_State

Nonwirable Output ENUMERATED INTEGER. DPM Task2State: DPM_State Represents the main state of the DPM control program. It should be OPERATE.
For more information see the Hilscher Protocol Interface Manual, ProfibusDP Master.

Table 4-4: Profi_DPM DPM_State values

| DPM_State | Description |
|---|---|
| 0 | Module state is OFFLINE |
| 1 | Module state is STOP |
| 2 | Module state is CLEAR |
| 3 | Module state is OPERATE |

### 4.3.19 ErrRemAdrs

Nonwirable Output INTEGER. Max: 255, Min: 0. DPM Task2State: Err_rem_adrs If a faulty slave is detected, its address will be reported in this parameter and {NonExchErr} will be set On. If more than one fault occurs, the first slave to be detected will be reported here. A value of 255 indicates that the source of the error is the DPM itself.
For more information see the Hilscher Protocol Interface Manual, ProfibusDP Master.

### 4.3.20 ErrEvent

Nonwirable Output INTEGER. Max: 255, Min: 0. DPM Task2State: Err_event
The error number associated with the faulty unit on the bus.
For more information see the Hilscher Protocol Interface Manual, ProfibusDP Master.

Table 4-5: Profi_DPM ErrEvent values

| ErrEvent | Description |
|---|---|
| 0 | No error. Device is functioning properly. |
| 2 | Station reports overflow generated by the master telegram. Check the length of the configuration and/or parameter data for the con figured slaves. Reload the data base if necessary. |
| 3 | The master has made a function request that is not recognised by the station. Check slave for compatibility. |
| 9 | No response from the slave to a read request. Check the configu ration data of the station and compare it with the physical I/O data length. |
| 17 | No response from the station. Check cable, slave address, power on etc. |
| 18 | Master not in the logical token ring. This is a device error. Check the FDL-Address of the master or high-station-Address of other master systems. Check cabling for bus short circuits. |
| 21 | Faulty parameter in request. This error is in the master telegram. Contact technical support or the Hilscher hotline. |
| 50 | USR_INTF task not found. This is a device error in the DPM module. Contact technical support. |
| 51 | No global data field. This is a device error in the DPM module. Contact technical support. |
| 52 | FDL task not found. This is a device error in the DPM module. Contact technical support. |
| 53 | PLC task not found. This is a device error in the DPM module. Contact technical support. |
| 54 | Master parameters not found. This is a device error in the DPM module. Download the DPM database again using SyCon. |
| 55 | Faulty parameter value in the master parameters. This is a configuration error. Check configuration and redownload from SyCon. |
| 56 | Slave parameters not found. This is a configuration error. Check configuration and redownload from SyCon. |

| ErrEvent | Description |
|---|---|
| 57 | Faulty parameter value in the slave parameters. This is a configuration error. Check configuration and redownload from SyCon. |
| 58 | Duplicate slave address. This is a configuration error. Check configuration and redownload from SyCon. |
| 59 | The Send Process Data offset address of one device is larger than 255. This is a configuration error. Check configuration and redownload from SyCon. |
| 60 | The Receive Process Data offset address of one device is larger than 255. This is a configuration error. Check configuration and redownload from SyCon. |
| 61 | Slave data areas are overlapping in the Send Process Data. This is a configuration error. Check configuration and redownload from SyCon. |
| 62 | Slave data areas are overlapping in the Receive Process Data. This is a configuration error. Check configuration and redownload from SyCon. |
| 63 | Unknown process data handshake. This is a configuration error. Check the warm start configuration and redownload from SyCon. |
| 64 | Free RAM exceeded. This is a device error. Contact technical sup port. |
| 65 | Faulty slave parameter data sets. This is a configuration error. Check configuration and redownload from SyCon. |
| 202 | No free segments. This is a device error. Contact technical sup port. |
| 212 | Faulty reading of the data base. This is a device error. Contact technical support. |
| 213 | Structure surrender to operating system faulty. This is a device error. Contact technical support. |

### 4.3.21 BusErrCount

Nonwirable Output INTEGER. Max: 65535, Min: 0

The number of heavy bus errors caused by, for example, bus short circuits.

The value is reported in the DPM Task2State, Bus_error_cnt parameter. For more information see the Hilscher Protocol Interface Manual, ProfibusDP Master.

### 4.3.22 TimeOutCnt

Nonwirable Output INTEGER. Max: 65535, Min: 0
The number of rejected Profibus telegrams caused by heavy bus errors.
The value is reported in the DPM Task2State, Time_out_cnt parameter. For more information see the Hilscher Protocol Interface Manual, ProfibusDP Master.

### 4.3.23 ResetTmOut

Wirable Input TIME. Max: 2147483647, Min: 0
The period of time allowed for the DPM to become operational once the bus is enabled. If the module is not Ready within this time, a ERROR_TIMEOUT_WAITING_FOR_MODE_OPERATE error is reported.

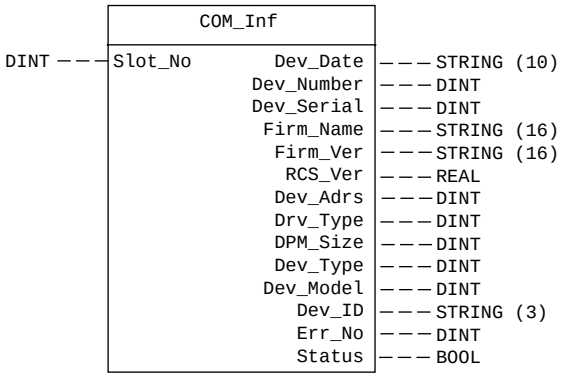
# 5. COM_Inf FF79

## 5.1 Functional Description

A function block to provide diagnostic information about any COM modules that are configured in the system. A COM driver must be installed for the module e.g. a Profi_DPM function block, and a COM_Table must be instantiated.
The data displayed by this block is read directly from the Hilscher COM module dual port memory user area.
The {Slot_No} can be changed online to read data from all installed modules. Alternatively, one block can be instantiated for each installed module.
For more information about the displayed parameters, see the Hilscher Toolkit Manual, General Definitions.

## 5.2 Parameter Diagram

Figure 5-1: COM_Inf block diagram



## 5.3 Parameter Descriptions

### 5.3.1 Slot_No

Wirable Input INTEGER. Max: 5, Min: 1
The rack slot number, 1 to 5, of the COM module. This parameter can be changed online to read data from any installed module.

### 5.3.2 Dev_Date

Nonwirable Output STRING
The date of manufacture of the installed Hilscher COM module.

### 5.3.3 Dev_Number

Nonwirable Output INTEGER. Max: 2147483647, Min: 0
The device number of the installed Hilscher COM module.

### 5.3.4 Dev_Serial

Nonwirable Output INTEGER. Max: 2147483647, Min: 0
The serial number of the installed Hilscher COM module.

### 5.3.5 Firm_Name

Nonwirable Output STRING
The name of the firmware installed in the Hilscher COM module. The first 8 bytes are the name of the loaded firmware and the last 8 bytes are the name of the board.

### 5.3.6 Firm_Ver

Nonwirable Output STRING
The version number of the firmware installed on the Hilscher COM module. It comprises a version number and a date.

### 5.3.7 RCS_Ver

Nonwirable Output REAL. Max: 3.40282E+38, Min: 3.40282E+38
The version number of the Hilscher COM module operating system.

### 5.3.8 Dev_Adrs

Nonwirable Output INTEGER. Max: 2147483647, Min: 0
This value should be 0. It is not used in the PC3000 implementation.

### 5.3.9 Drv_Type

Nonwirable Output INTEGER. Max: 2147483647, Min: 0
This value describes the way in which the dual port memory access is controlled. It should be 0x42 or 0x43 (decimal 66 or 67). This corresponds to Buffered, Host Controlled with a Bitwise handshake Mailbox.
If this value is not 66 or 67, the module has not been configured correctly. Use the Sycon PB/E software to set the Process Data Handshake to Buffered, Host Controlled.
The default value is 50 which signifies Inconsistent, Uncontrolled access. The consequence of this would be the block will not work because various flags used to control access to the data will not work.

### 5.3.10 DPM_Size

Nonwirable Output INTEGER. Max: 2147483647, Min: 0
The actual size of the Hilscher module dual port memory in K bytes.

### 5.3.11 Dev_Type

Nonwirable Output INTEGER. Max: 2147483647, Min: 0
The value of this parameter should be 53 signifying a Hilscher COM module. Any other value indicates an error in reading the dual port memory or a fault in the module.

### 5.3.12 Dev_Model

Nonwirable Output INTEGER. Max: 2147483647, Min: 0
This value indicates the type of COM module fitted. It should be 56 signifying a ProfibusDP master module or 75 signifying a DeviceNet slave module. Another value indicates a different module type, an error reading the memory or a module fault.67 COM-PB is supported by the Xcomms library.
Possible model numbers are:

Table 5-1: COM_Inf Dev_Model values

| Model Number | Description | |
|---|---|---|
| 49 | COM 10/11 | for standard protocols |
| 50 | COM 12 | for Profibus FMS |
| 55 | COMDPS | for Profibus DP Slave |
| 56 | COMDPM | for Profibus DP Master |
| 57 | COM | for CAN SDS |
| 67 | COM-PB | for Profibus Combi-Master |
| 75 | COM-DNS | for DeviceNet Slave |

### 5.3.13 Dev_ID

Nonwirable Output STRING
The Device Identifier of the installed Hilscher module. It should be COM. Any other value signifies an error reading the memory or a module fault.

### 5.3.14 Err_No

Nonwirable Output INTEGER. Max: 2147483647, Min: 1
The Err_No passed from the PC3000 COM driver e.g. Profi_DPM. This value is also available at that driver.
An error of -1 indicates that no driver is installed for the specified {Slot_No}.

Table 5-2: COM_Inf Err_No values

| Err_No | Error | Description |
|---|---|---|
| 0 | NO_ERROR | Module is installed and operating normally. |
| 16 | CS_ERR_NO_REMOTE_PARAM_SERV | There is no Profi_DPM driver assigned to the specified slot. |
| 40 | ERROR_MODULE_NOT_NETWORK_MOTHER_BOARD | The module detected at the specified {Slot_No} is not a Network Card. |
| 42 | ERROR_MODULE_MEMORY_INCORRECT | The module size as read from the dual port memory does not match that specified in {Dev_Mem}. |
| 43 | ERROR_MODULE_DEV_NOT_COM | The installed Hilscher module type is not a COM. |
| 44 | ERROR_MODULE_DEV_NOT_DPM | The installed Hilscher module model is not a DPM. |
| 45 | ERROR_MODULE_ID_NOT_COM | The installed Hilscher device ID is not COM. |
| 46 | ERROR_MODULE_NOT_READY | The module Ready flag is not set. |
| 47 | ERROR_MODULE_WDOG_FAIL | A module watchdog failure has been detected. Not currently supported. |
| 48 | ERROR_SLOT_ALREADY_OCCUPIED | The {Slot_No} specified already has a module installed. |

| Err_No | Error | Description |
|---|---|---|
| 49 | ERROR_INIT_WRITE_FAIL_TEST_ID | A dual port memory fault was detected during initialisation. |
| 50 | ERROR_INIT_WRITE_FAIL_ORIG_ID | A dual port memory fault was detected during initialisation. |
| 51 | ERROR_NO_COM_TABLE_INSTANTIATED | No instance of a COM_Table function block was found during initialisation. This block is mandatory. |
| 55 | ERROR_MODULE_DEV_NOT_DNS | The installed Hilscher module model is not a DNS. |
| 63 | ERROR_TIMEOUT_WAITING_FOR_READY_FLAG | A Reset or an Init has been performed and the module Ready flag is not set. |
| 64 | ERROR_TIMEOUT_WAITING_FOR_RESET_FLAG_TO_CLEAR | A reset has been requested and actioned but the module is not re-setting its Reset flag. |
| 65 | ERROR_TIMEOUT_WAITING_FOR_MODE_OPERATE | A reset has been requested but the DPM module will not go to OPERATE mode. |
| 70 | ERROR_DO_CONFIG_UNKNOWN_STATE | The state engine controlling the configuration software got into an invalid state. The state has been reset, the configuration may not be complete. |

### 5.3.15 Status

Nonwirable Output BOOL
Set to GO (1) if {Err_No} is zero, otherwise set to NOGO (0).

# 6. COM_Slv_Sta FF7B

## 6.1 Functional Description

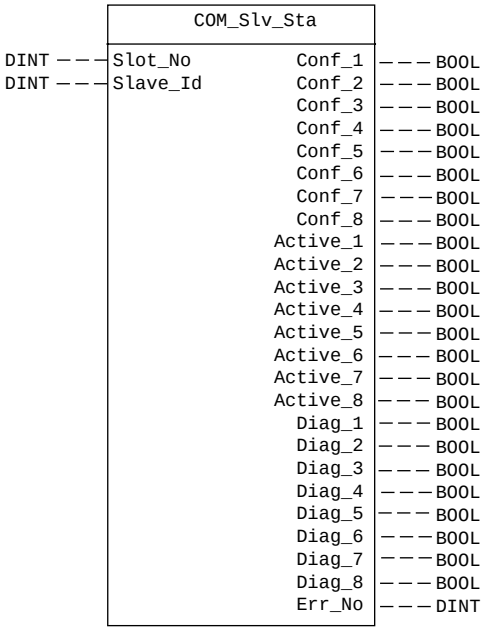
A function block to display the network status of Profibus slaves. Eight consecutive slaves are displayed, starting at a specified slave address.

The block indicates whether a slave is configured on the network, whether it is communicating and whether there is any current diagnostic information available for it.

Although individual COM_Vars will report an error if a slave stops communicating, the validity of all data relating to a particular slave is best tested using the Active flags in this block.

## 6.2 Parameter Diagram

Figure 6-1: COM_Slv_Sta parameter diagram



## 6.3 Parameter Descriptions

### 6.3.1 Slot_No

Wirable Input INTEGER. Max: 5, Min: 1
The PC3000 rack position, 1 to 5, of the Profibus master module. It can be changed dynamically.

### 6.3.2 Slave_Id

Wirable Input INTEGER. Max: 128, Min: 0
The slave address relating to the {Conf_1}, {Active_1} and {Diag_1} outputs. The other outputs correspond to successive slave addresses starting at {Slave_Id}.

### 6.3.3 Conf_1 to Conf_8

Nonwirable Output BOOL
Indicates whether the instrument at this address is configured on the network. {Conf_1} relates to the slave at address {Slave_Id}; {Conf_2} relates to the slave at address {Slave_Id}+1; etc.

Table 6-1: COM_Slv_Sta Conf_N values

| Value | Description |
|---|---|
| 0 | This slave is not configured. |
| 1 | This slave is configured. |

### 6.3.4 Active_1 to Active_8

Nonwirable Output BOOL
Indicates whether the instrument at this address is communicating on the network. {Active_1} relates to the slave at address {Slave_Id}; {Active_2} relates to the slave at address {Slave_Id}+1; etc.
Note that this parameter will only become true when an explicit read or write is made to the dual port memory and a successful network transaction has occurred with this slave.

Table 6-2: COM_Slv_Sta Active_N values

| Value | Description |
|---|---|
| 0 | This slave is not communicating. |
| 1 | This slave is communicating. |

### 6.3.5 Diag_1 to Diag_8

Nonwirable Output BOOL
Indicates whether the master module has any current diagnostic information for the instrument at this address. {Diag_1} relates to the slave at address {Slave_Id}; {Diag_2} relates to the slave at address {Slave_Id}+1; etc.
The information can be read using a COM_Diag function block. The flag is reset when the information has been read.

Table 6-3: COM_Slv_Sta Diag_N values

| Value | Description |
|---|---|
| 0 | There is no diagnostic available for this instrument. |
| 1 | There is diagnostic data available for this instrument. Use a COM_Diag function block to read it. |

### 6.3.6 Err_No

Nonwirable Output INTEGER. Max: 255, Min: 0
Reports errors in the function block operation.

Table 6-4: COM_Slv_Sta error numbers

| Err_No | Error | Description |
|---|---|---|
| 0 | NO_ERROR | The block is operating normally. |
| 16 | CS_ERR_NO_REMOTE_PARAM_SERV | There is no communications driver installed in the specified {Slot_No}. |
| 51 | ERROR_NO_COM_TABLE_INSTANTIATED | There is no COM_Table function block. This block is compulsory for fieldbus applications and is needed to support the Profi_DPM function block. |
| 127 | ERROR_INVALID_SLOT_ADDRESS | {Slot_No}must be in the range 1 - 5. |

# 7. DevNet_S FF7D

## 7.1 Functional Description

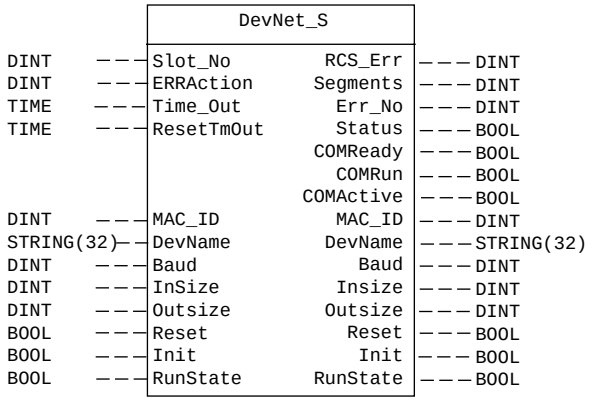
A DeviceNet Slave comms driver. It requires a DeviceNet DNS module to be present in one of the first five rack slots.
One COM_Table function block is required to service this and any other similar comms driver. One COM_Table per program.
Information about the module that is not available in this function block may be obtained by instantiating a COM_Inf function block with the same slot number.
The DeviceNet DNS module uses a Hilscher COM-DNS module.
It may be configured using the Hilscher SyCon software in which case certain system settings need to be made and these are documented in the User Manual for this module.
Alternatively, the configuration can be done by this function block in which case, the system settings are made automatically.

**IMPORTANT NOTE**

A configuration downloaded from SyCon is kept in non-volatile memory on the DNS module and is copied to the run-time memory, becoming the cold-start default, on any power up.
If this function block is used to configure the module, the data is held in the volatile dual port memory. On a power down, the data in the dual port memory is lost. On a power up, the default settings, if any, are loaded from non-volatile memory (not from this block!). It is not until this block initialises the DPM module (via the {Init} parameter) that the proper configuration is regained. It is, therefore, important that the user program detects a power up condition and triggers the {Init} parameter.

## 7.2 Parameter Diagram

Figure 7-1: DevNet_S block diagram




## 7.3 Parameter Descriptions

### 7.3.1 Slot_No

Wirable Input INTEGER. Max: 5, Min: 1
The rack slot number into which the module is installed. If more than one driver is given the same slot number, an error will be generated.

### 7.3.2 ErrAction

Wirable Input ENUMERATED INTEGER
Defines the action to be taken in the event of a comms failure. The DeviceNet slave detects the presence or absence of communications activity on the network and can take specified action on the cessation of comms activity.

Table 7-1: DevNet_S ErrAction enumerations

| Value | Enumeration | Description |
|---|---|---|
| 0 | None | The input memory, and hence the COS variables, retain the last known values for the duration of the comms failure. Special action may be taken in the user program by reference to the {COMActive} flag. |

| Value | Enumeration | Description |
|---|---|---|
| 1 | RstDur | The entire input memory, and hence all the COS variables, are reset to 0 for the duration of the comms failure. {RunState} is set to Off. When comms is reestablished, {RunState} is set to Run and values will be updated . |
| 2 | RstWait | On comms failure, the input memory and, hence, all COS variables, are set to 0 and {RunState} is set Off. The user program must detect comms, using {COMActive}, and set {RunState} to Run before values are again updated. |

### 7.3.3 Time_Out

Wirable Input TIME. Max: 2147483647, Min: 0

Access to the dual port memory on the Hilscher module is controlled by a number of semaphore flags. If a response is not received from the module with {Time_Out}, an error is returned.

### 7.3.4 ResetTmOut

Wirable Input TIME. Max: 2147483647, Min: 0

The period of time allowed for the module to become operational after a {Reset} or an {Init}.

### 7.3.5 RCS_Err

Nonwirable Output INTEGER. Max: 2147483647, Min: 0

The Hilscher COM-DPM operating system error byte as read from the dual port memory.

A nonzero value indicates a module initialisation error. Consult technical support or the Hilscher hotline.

If an error value and description is given below, it is taken from the Sycon online help file. (see 18.4.2)

Table 7-2: DevNet_S RCS_Err values

| RCS_Err Number | Description |
|---|---|
| 0 | No error |
| 4 | Task does not exist |
| 5 | Task is not initialised |
| 6 | The MCL is locked |
| 7 | The MCL rejects a send command because of an error |
| 20 | Data base not configured |

| RCS_Err Number | Description |
|---|---|
| 21 | Data base segment not configured or doesn't exist |
| 22 | Number for message wrong during download |
| 23 | Received number of data during download does not match to that in the command message |
| 24 | Sequence identifier wrong during download |
| 25 | Checksum after download and checksum in command message does not match |
| 26 | Write/Read access of data base segment |
| 27 | Download/Upload or erase of configured data base type is not allowed |
| 28 | The state of the data base segment indicated an error. Upload not possible |
| 29 | The access to the data base segment needs the boot strap loader. The boot strap loader is not present |
| 30 | Trace buffer overflow |
| 31 | Entry into trace buffer too long |
| 37 | No or wrong licence. The OEM licence of the system configurator allows only communication to devices that have the same licence inside |
| 38 | The data base created by the system configurator and the data base expected by the firmware is not compatible |
| 39 | DBM module missing |
| 40 | No command free |
| 41 | Command unknown |
| 42 | Command mode unknown |
| 43 | Wrong parameter in the command |
| 44 | Message length does not match to the parameters of the command |
| 45 | Only a MCL does use this command to the RCS |
| 50 | FLASH occupied at the moment |
| 51 | Error deleting the FLASH |
| 52 | Error writing the FLASH |
| 53 | FLASH not configured |
| 54 | FLASH timeout error |
| 55 | Access protection error while deleting the FLASH |
| 56 | FLASH size does not match or not enough FLASH memory |
| 60 | Wrong structure type |
| 61 | Wrong length of structure |

| RCS_Err Number | Description |
|---|---|
| 62 | Structure does not exist |
| 70 | No clock on the device |
| 80 | Wrong handle for the table (table does not exist) |
| 81 | Data length does not match the structure of this table |
| 82 | The data set of this number does not exist |
| 83 | This table name does not exist |
| 84 | Table full. No more entries allowed |
| 85 | Other error from DBM |
| 90 | The device info (serial number, device number and date) does already exist |
| 91 | Licence code invalid |
| 92 | Licence code does already exist |
| 93 | All memory locations for licence codes already in use |

### 7.3.6 Segments

Nonwirable Output INTEGER. Max: 2147483647, Min: 0
Messages between the LCM and the Hilscher module are buffered in segments. The number of segments remaining is read from the module and displayed in this parameter.

### 7.3.7 Err_No

Nonwirable Output INTEGER. Max: 2147483647, Min: 0
Indicates the cause of any error in the driver installation.

Table 7-3: DevNet_S Err_No values

| Err_No | Error | Description |
|---|---|---|
| 0 | NO_ERROR | Module is installed and operating normally. |
| 40 | ERROR_MODULE_NOT_NETWORK_MOTHER_BOARD | The module detected at the specified {Slot_No} is not a Network Card. |
| 43 | ERROR_MODULE_DEV_NOT_COM | The installed Hilscher module type is not a COM. |
| 45 | ERROR_MODULE_ID_NOT_COM | The installed module ID is not COM. |
| 46 | ERROR_MODULE_NOT_READY | The module Ready flag is not set. |

| Err_No | Error | Description |
|---|---|---|
| 47 | ERROR_MODULE_WDOG_FAIL | A module watchdog failure has been detected. Not currently supported. |
| 48 | ERROR_SLOT_ALREADY_OCCUPIED | The {Slot_No} specified already has a module installed. |
| 49 | ERROR_INIT_WRITE_FAIL_TEST_ID | A dual port memory fault was detected during initialisation. |
| 50 | ERROR_INIT_WRITE_FAIL_ORIG_ID | A dual port memory fault was detected during initialisation. |
| 51 | ERROR_NO_COM_TABLE_INSTANTIATED | No instance of a COM_Table function block was found during initialisation. This block is mandatory. |
| 55 | ERROR_MODULE_DEV_NOT_DNS | The installed Hilscher module model is not a DNS. |
| 56 | ERROR_MAX_NO_COS_BLOCKS_EXCEEDED | Up to 512 COS_Vars can be registered with a DevNet_S driver. |
| 58 | ERROR_INVALID_INPUT_DATA_SIZE | The Input Data Size must be non-negative and ≤ 255. |
| 59 | ERROR_INVALID_OUTPUT_DATA_SIZE | The Output Data Size must be non-negative and ≤ 255. |
| 60 | ERROR_BUS_NOT_ENABLED | The driver is not in {RunState}= Run |
| 61 | ERROR_TIMEOUT_WAITING_FOR_COM | The module's COM host flag is not set. |
| 62 | ERROR_TIMEOUT_WAITING_FOR_ACCESS | The module is not releasing the dual port memory. |

| Err_No | Error | Description |
|---|---|---|
| 63 | ERROR_TIMEOUT_WAITING_FOR_READY_FLAG | |
| | | A Reset or an Init has been performed and the module Ready flag is not set. |
| 64 | ERROR_TIMEOUT_WAITING_FOR_RESET_FLAG_TO_CLEAR | |
| | | A reset has been requested and actioned but the module is not re-setting its Reset flag. |
| 65 | ERROR_TIMEOUT_WAITING_FOR_MODE_OPERATE | |
| | | The device is not in mode Operate. |
| 66 | ERROR_TIMEOUT_WAITING_FOR_INIT_FLAG_TO_CLEAR | |
| | | An Init has been performed but the device is not releasing the INIT flag. |
| 68 | ERROR_TIMEOUT | Unspecified timeout error |
| 69 | ERROR_COMMS_NOT_ACTIVE | There is no comms activity on the network |
| 70 | ERROR_DO_CONFIG_UNKNOWN_STATE | The state engine controlling the configuration software got into an invalid state. The state has been reset, the configuration may not be complete. |

### 7.3.8 Status

Nonwirable Output BOOL

This parameter is set to GO (1) if {Err_No} is zero. Otherwise, this parameter is set to NOGO (0).

### 7.3.9 COMReady

Nonwirable Output BOOL HostFlags: Ready

The value of the Hilscher HostFlags Ready bit as read from dual port memory. Indicates that the device is configured and ready to run.

### 7.3.10 COMRun

Nonwirable Output BOOL HostFlags: Run
The value of the Hilscher HostFlags Run bit as read from dual port memory. Indicates that the device is running.

### 7.3.11 COMActive

Nonwirable Output BOOL HostFlags: Com
The value of the Hilscher HostFlags Com bit as read from dual port memory. Indicates that the process data exchange is active. When this value goes Off, the action specified by {ErrAction} is activated.

### 7.3.12 MAC_ID

Nonwirable Input INTEGER. Max: 63, Min: 0
The MAC_Id assigned to this slave device. This value is read from the module's cold start values on a {Reset} or written to the run-time memory area on an {Init}.

### 7.3.13 DevName

Nonwirable Input STRING
The DeviceName assigned to this slave device. This value is read from the module's cold start values on a {Reset} or written to the run-time memory area on an {Init}.

### 7.3.14 Baud

Nonwirable Input ENUMERATED INTEGER
The baud rate assigned to this slave device. This value is read from the module's cold start values on a {Reset} or written to the run-time memory area on an {Init}.

Table 7-4: DevNet_S Baud values

| Baud | Value |
|---|---|
| 0 | 500 |
| 1 | 250 |
| 2 | 125 |
| 3 | Auto baud |

### 7.3.15 InSize

Wirable Input INTEGER. Max: 255, Min: 0
The Consumed size assigned to this slave device. This value is read from the module's cold start values on a {Reset} or written to the run-time memory area on an {Init}. This value must match the value specific in the configuration of the DeviceNet master.

### 7.3.16 OutSize

Wirable Input INTEGER. Max: 255, Min: 0
The Produced size assigned to this slave device. This value is read from the module's cold start values on a {Reset} or written to the run-time memory area on an {Init}. This value must match the value specific in the configuration of the DeviceNet master.

### 7.3.17 Reset

Nonwirable Input BOOL
Performs a cold start of the module by setting the DevFlag Reset bit. The module is reinitialised and values copied from the cold start area to {MAC_ID}, {DevName}, {Baud}, {InSize} and {OutSize}.
Cold start values can be programmed into the device using the Hilscher configuration tool, SyCon, but it is not necessary. Warm start values for these parameters can be set in the block and an {Init} performed to copy them to the runtime values.

### 7.3.18 Init

Nonwirable Input BOOL
Performs a warm start of the module by setting the DevFlag Init bit. The module is reinitialised and values copied from {MAC_ID}, {DevName}, {Baud}, {InSize} and {OutSize} to the runtime locations.
Default cold start values can be programmed into the device using the Hilscher configuration tool, SyCon, and then loaded to the runtime area with a {Reset}.

### 7.3.19 RunState

Nonwirable Input BOOL
Specifies whether values are copied between the dual port memory and the COS variables. It can be set by the user program but can also be set by the block depending on the value of {ErrAction}.

Table 7-5 DeviNet_S RunState values

| RunState | Sense | Description |
|---|---|---|
| 0 | Off | Network comms may still be running but values are not copied from the dual port memory to the COS variables or vice versa. This state is automatically entered on a comms fail if {ErrAction} is either RstDur or RstWait. This state allows the user to write to COS_Vars that are configured as Input and is, therefore, useful for forcing values for test purposes. |

| RunState | Sense | Description |
|---|---|---|
| 1 | Run | Values are copied between the dual port memory and COS variables. This is the normal state. This value is set automatically on recovery from comms failure if {ErrAction} is set to RstDur. |

# 8. COM_Table FF7F

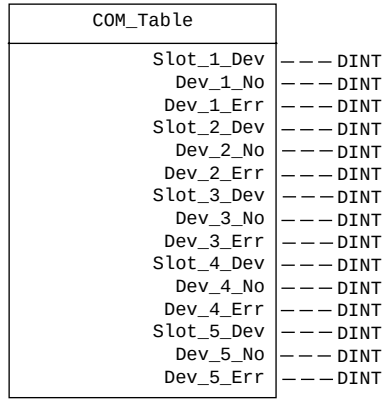
## 8.1 Functional Description

This is a downloadable support table for the Hilscher COM range of communication modules. One instance of this block must be created if one or more COM modules are to be used. It is a table of pointers which permit the different function blocks to pass data between them without the need for user wiring in the application program.

It provides the linkage mechanism between the COM_Vars, COS_Vars, COM drivers and COM_Inf function blocks. The parameters displayed are for verification and diagnostic purposes only and are also available at either the corresponding COM driver or the COM_Inf function block. A COM driver will show an error if this block does not exist.

## 8.2 Parameter Diagram

Figure 8-1: COM_Table block diagram




## 8.3 Parameter Descriptions

There are three parameters for each of the five possible slot positions. The parameters are provided for diagnostic purposes only.

### 8.3.1 Slot_1_Dev to Slot_5_Dev

Nonwirable Output ENUMERATED INTEGER. Max: 5, Min: 0
The type of COM driver assigned to the slot. This information is obtained from the COM driver function block assigned to the slot, it is not read from the hardware itself. Possible values are as follows.

Table 8-1: COM_Table Slot_n_Dev enumerations

| Value | Enumeration | Description |
|---|---|---|
| 0 | No_Dev | No driver assigned to this slot |
| 1 | COM_DPM | A Profi_DPM driver is assigned to this slot |
| 2* | COM_DPS | A Profi_DPS driver is assigned to this slot |
| 4* | COM_DNM | A DevNet_M driver is assigned to this slot |
| 5 | COM_DNS | A DevNet_S driver is assigned to this slot |

** Not available yet*

### 8.3.2 Dev_1_No to Dev_5_No

Nonwirable Output INTEGER. Max: 2147483647, Min: 0
The Device Number of the module installed in this slot. This value is read from the COM driver assigned to this slot. It is the DeviceNumber parameter in the Device Information area of the dual port memory.

### 8.3.3 Dev_1_Err to Dev_5_Err

Nonwirable Output INTEGER Max: 255, Min: 0
The {Err_No} of the COM driver assigned to this slot. See paragraph 4.3.6 on page 5 or paragraph 7.3.7 on page 25 for possible values.
If there is no COM driver assigned to the slot, the block displays CS_ERR_NO_REMOTE_PARAM_SERV, error number 16.

# 9. COM_Slv_Inf FF8F

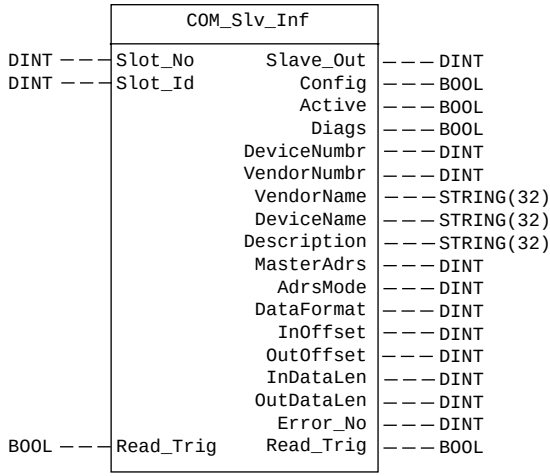
## 9.1 Functional Description

A function block to read information about individual slave instruments from the Profi_DPM driver.
The information obtained is the information provided by the configurer of the network and supplied to the SyCon PB/E software (see 18.4.2) when configuring the master module. Some of the information comes from the GSD file for each slave and some is provided by the configurer.

The function block does not obtain the data continuously but only when a Read trigger is set. This minimises unnecessary overhead on the system. Although {Config}, {Active} and {Diags} are available in this block, they are better obtained from the COM_Slv_Sta block which doesn't need triggering.

## 9.2 Parameter Diagram

Figure 9-1: COM_Slv_Inf parameter diagram




## 9.3 Parameter Descriptions

### 9.3.1 Slot_No

Wirable Input INTEGER. Max: 5, Min: 1
The rack slot number, 1 to 5, of the installed Profi_DPM driver. This value may be changed dynamically.

### 9.3.2 Slave_Id

Wirable Input INTEGER. Max: 128, Min: 0
The ID of the slave instrument for which data is required. This value may be changed dynamically. Thus, one function block may be used to scan all the slaves or one function block may be instantiated for each slave.

### 9.3.3 Slave_Out

Nonwirable Output INTEGER. Max: 128, Min: 0
Indicates the slave ID for which the displayed data is valid. This value is only updated when {Read_Trig} is set and the relevant data is obtained.

### 9.3.4 Error_No

Nonwirable Output INTEGER. Max: 2147483647, Min: 0
Will be nonzero if there is a failure during operation of the block.
Error numbers appearing here may be reported by the block itself, by the Profi_DPM driver or by the internal COM_Str function block which is used to read and write the messages. Errors not shown in Table 9-1 will be found in Table 12-7 on page 54.

Table 9-1: COM_Slv_Inf error numbers

| Error_No | Error | Description |
|---|---|---|
| 0 | NO_ERROR | Either no request has yet been made or the last request ended successfully. |
| 16 | CS_ERR_NO_REMOTE_PARAM_SERV | There is no communications driver installed in the specified {Slot_No}. |
| 51 | ERROR_NO_COM_TABLE_INSTANTIATED | There is no COM_Table function block. This block is compulsory for fieldbus applications and is needed to support the Profi_DPM function block. |
| 60 | ERROR_BUS_NOT_ENABLED | The bus is disabled at the COM driver and the DPM module is, therefore, not responding to requests. |

| Error_No | Error | Description |
|---|---|---|
| 68 | ERROR_TIMEOUT | Timeout waiting for the DPM module to respond to the request for information. The timeout period is set at the Profi_DPM driver in the {TimeOut} parameter. |
| 142 | ERROR_SLAVE_ADDRESS_NOT_CONFIGURED | The slave address, for which information is requested, is not configured as part of the network. Either reconfigure the DPM module to include the specified slave or request information for a configured slave. |

### 9.3.5 Config

Nonwirable Output BOOL
Indicates whether the specified slave {Slave_Out} is configured on the network.

Table 9-2: COM_Slv_Inf Config values

| Value | Description |
|---|---|
| 0 | This slave instrument is not configured on the network. |
| 1 | This slave instrument is configured on the network. |

### 9.3.6 Active

Nonwirable Output BOOL
Indicates whether the specified slave {Slave_Out} is communicating on the network. Note that this parameter will only become true when an explicit read or write is made to the dual port memory and a successful network transaction has occurred with this slave.

Table 9-3: COM_Slv_Inf Active values

| Value | Description |
|---|---|
| 0 | This slave instrument is not communicating on the network. |
| 1 | This slave instrument is communicating on the network. |

### 9.3.7 Diags

Nonwirable Output BOOL
Indicates whether there are new diagnostics for the specified slave, {Slave_Out}. The diagnostics can be read using a COM_Diag function block and the act of reading them will reset this bit.
Only updated on a {Read_Trig}.

Table 9-4: COM_Slv_Inf Diags values

| Value | Description |
|---|---|
| 0 | There are no new diagnostics for {Slave_Out}. |
| 1 | There are new diagnostics available for {Slave_Out}. |

### 9.3.8 DeviceNumbr

Nonwirable Output INTEGER. Max: 2147483647, Min: 0
The Device Number of this instrument. This data is extracted, by the Sycon PB/E software (see 18.4.2), from the GSD file for this slave instrument.

### 9.3.9 VendorNumbr

Nonwirable Output INTEGER. Max: 2147483647, Min: 0
The Vendor Number of this instrument. This data is extracted, by the Sycon PB/E software (see 18.4.2), from the GSD file for this slave instrument.

### 9.3.10 VendorName

Nonwirable Output STRING
The Vendor Name for this instrument. This data is extracted, by the Sycon PB/E software (see 18.4.2), from the GSD file for this slave instrument.

### 9.3.11 DeviceName

Nonwirable Output STRING
The Device Name of this instrument. This data is extracted, by the Sycon PB/E software (see 18.4.2), from the GSD file for this slave instrument.

### 9.3.12 Description

Nonwirable Output STRING
A text string describing this instrument (e.g. Zone_1_Temp) which was supplied by the network configurer.

### 9.3.13 MasterAdrs

Nonwirable Output INTEGER. Max: 128, Min: 0
The address of the master which configured this slave.

### 9.3.14 AdrsMode

Nonwirable Output ENUMERATED INTEGER
The memory addressing mode specified by the user when the DPM module was configured.

Table 9-5: COM_Slv_Inf AdrsMode vlaues

| Value | Description |
|---|---|
| 0 | None |
| 1 | Byte |
| 2 | Word |

### 9.3.15 DataFormat

Nonwirable Output ENUMERATED INTEGER
The data format mode specified by the user when the DPM module was configured.

Table 9-6: COM_Slv_Inf DataFormat values

| Value | Description |
|---|---|
| 0 | None |
| 1 | Little Endian |
| 2 | Big Endian |

### 9.3.16 InOffSet

Nonwirable Output INTEGER. Max: 2147483647, Min: 0
A diagnostic parameter. The offset into the Receive Process Data area in the DPM dual port RAM where the data for this slave may be found.
The information is not necessary for ordinary usage of the Profibus system. The Profi_DPM driver function block uses this information when validating variable addresses and processing communication requests.
A value of -1 indicates the slave is not configured.

### 9.3.17 OutOffSet

Nonwirable Output INTEGER. Max: 2147483647, Min: 0
A diagnostic parameter. The offset into the Send Process Data area in the DPM dual port RAM where the data for this slave may be found.
The information is not necessary for ordinary usage of the Profibus system. The Profi_DPM driver function block uses this information when validating variable addresses and processing communication requests.
A value of -1 indicates the slave is not configured.

### 9.3.18 InDataLen

Nonwirable Output INTEGER. Max: 2147483647, Min: 0
A diagnostic parameter. The total data length of the slave's Input data. The Profi_DPM driver function block uses this information when validating variable addresses and processing communication requests.
A value of -1 indicates the slave is not configured.

### 9.3.19 OutDataLen

Nonwirable Output INTEGER. Max: 2147483647, Min: 0
A diagnostic parameter. The total data length of the slave's Output data. The Profi_DPM driver function block uses this information when validating variable addresses and processing communication requests.
A value of -1 indicates the slave is not configured.

### 9.3.20 Read_Trig

Nonwirable Input BOOL
Set this parameter On to read the data for the specified slave. It will automatically reset.

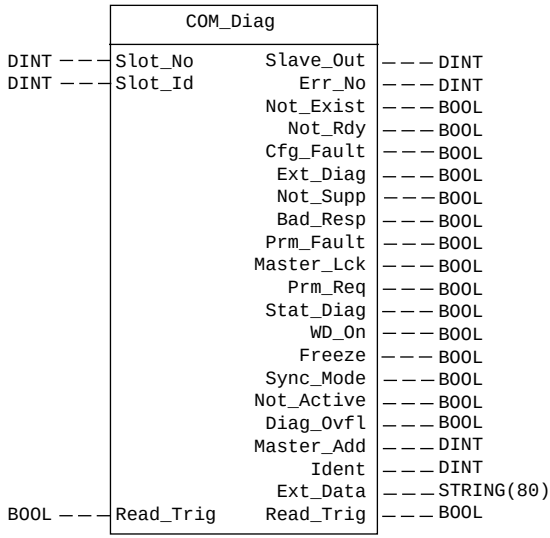
# 10. COM_Diag FF90

## 10.1 Functional Description

COM_Diag FF90
Read DP-Slave Diagnostic Information
This function block will read the slave diagnostics for one slave device on demand. The existence of new diagnostic information for a slave can be determined from a COM_Slv_Sta block or a COM_Slv_Inf block.
Profibus slave diagnostics consist of six bytes of data specified by the Profibus standard followed by up to 63 bytes of device specific data, the ext_diag_data, specific to the slave device.
This function block interprets the first six bytes as per the Profibus specification and outputs the ext_diag_data as a string, {Ext_Data}, which can be interpreted by the PC3000 user program.
The first byte of {Ext_Data} is a header byte and contains the total length of the ext_diag_data including the header byte.

## 10.2 Parameter Diagram

Figure 10-1: COM_Diag block diagram



## 10.3 Parameter Descriptions

### 10.3.1 Slot_No

Wirable Input INTEGER. Max: 5, Min: 1
The rack slot number of the installed Profi_DPM driver. This value may be changed dynamically.

### 10.3.2 Slave_ID

Wirable Input INTEGER. Max: 128, Min: 0
The ID of the slave instrument for which data is required. This value may be changed dynamically. Thus, one function block may be used to get diagnostics for all slaves or a function block may be instantiated for each slave.

### 10.3.3 Slave_Out

Nonwirable Output INTEGER. Max: 128, Min: 0
Indicates the slave ID for which the displayed data is valid. This value is only updated when {Read_Trig} is set and the relevant data is obtained.

### 10.3.4 Err_No

Nonwirable Output INTEGER Max: 2147483647, Min: 0
Will be non-zero if there is a failure during operation of the block.
Error numbers appearing here may be reported by the block itself, by the Profi_DPM driver or by the internal COM_Str function block which is used to read and write the messages. Errors not shown in Table 10-1 will be found in Table 12-7 on page 54.

Table 10-1: COM_Slv_Inf error numbers

| Error_No | Error | Description |
|---|---|---|
| 0 | NO_ERROR | Either no request has yet been made or the last request ended successfully. |
| 16 | CS_ERR_NO_REMOTE_PARAM_SERV | There is no communications driver installed in the specified {Slot_No}. |
| 51 | ERROR_NO_COM_TABLE_INSTANTIATED | There is no COM_Table function block. This block is compulsory for fieldbus applications and is needed to support the Profi_DPM function block. |

| Error_No | Error | Description |
|---|---|---|
| 60 | ERROR_BUS_NOT_ENABLED | The bus is disabled at the COM driver and the DPM module is, therefore, not responding to requests. |
| 68 | ERROR_TIMEOUT | Timeout waiting for the DPM module to respond to the request for information. The timeout period is set at the Profi_DPM driver in the {TimeOut} parameter. |
| 142 | ERROR_SLAVE_ADDRESS_NOT_CONFIGURED | The slave address, for which information is requested, is not configured as part of the network. Either recon figure the DPM module to include the specified slave or request information for a configured slave. |

### 10.3.5 Not_Exist

Nonwirable Output BOOL. (Profibus Diag_Data: Station_status_1: Bit 0)
This bit is set by the DPM module if the respective slave can not be reached over the line.
If this bit is set, the rest of the diagnostic bits contain the last known state or the initial value of 0. The slave sets this bit to zero when communication is established.

### 10.3.6 Not_Rdy

Nonwirable Output BOOL. (Profibus Diag_Data: Station_status_1: Bit 1)
This bit is set by the slave if it is not yet ready for data transfer.

### 10.3.7 Cfg_Fault

Nonwirable Output BOOL. (Profibus Diag_Data: Station_status_1: Bit 2)
This bit is set by the slave if the last received configuration data from the master is different from that which the slave has determined.

### 10.3.8 Ext_Diag

Nonwirable Output BOOL. (Profibus Diag_Data: Station_status_1: Bit 3)
This bit is set by the slave. If set to 1, it indicates that a diagnostic entry exists in the slave specific diagnostic area (ext_diag_data). If it is set to 0, a status message can still exist in ext_diag_data.

### 10.3.9 Not_Supp

Nonwirable Output BOOL. (Profibus Diag_Data: Station_status_1: Bit 4)
This bit is set by the slave when a function is requested which is not supported by the slave.

### 10.3.10 Bad_Resp

Nonwirable Output BOOL. (Profibus Diag_Data: Station_status_1: Bit 5)
This bit is set by the DPM when an invalid response is received from the slave. The slave resets it to 0.

### 10.3.11 Prm_Fault

Nonwirable Output BOOL. (Profibus Diag_Data: Station_status_1: Bit 6)
This bit is set by the slave when a faulty parameter frame is received, e.g. wrong length, wrong Ident_Number, invalid parameters.

### 10.3.12 Master_Lck

Nonwirable Output BOOL. (Profibus Diag_Data: Station_status_1: Bit 7)
The slave has been parameterised by a different master. This bit is set by the DPM if the address in (Master_Add} is not 255 and not equal to its own address. The slave sets the bit to 0.

### 10.3.13 Prm_Req

Nonwirable Output BOOL. (Profibus Diag_Data: Station_status_2: Bit 0)
The slave sets this bit to 1 to request reparameterisation and reconfiguration. The bit remains set until the parameterisation is finished.

### 10.3.14 Stat_Diag

Nonwirable Output BOOL. (Profibus Diag_Data: Station_status_2: Bit 1)
If the slave sets this bit, the DPM will fetch diagnostic information continuously until the bit is reset. For example, the slave will set this bit if it is not able to provide valid user data.

### 10.3.15 WD_On

Nonwirable Output BOOL. (Profibus Diag_Data: Station_status_2: Bit 3)
This bit is set by the slave as soon as its watchdog control has been activated.

### 10.3.16 Freeze

Nonwirable Output BOOL. (Profibus Diag_Data: Station_status_2: Bit 4)
This bit is set by the slave when it receives a Freeze control command.

### 10.3.17 Sync_Mode

Nonwirable Output BOOL. (Profibus Diag_Data: Station_status_2: Bit 5)
This bit is set by the slave when it receives a Sync control command.

### 10.3.18 Not_Active

Nonwirable Output BOOL. (Profibus Diag_Data: Station_status_2: Bit 7)
This bit is set by the DPM when the slave has been marked inactive within the slave parameter set and has been removed from cyclic processing. The slave always sets this bit to 0.

### 10.3.19 Diag_Ovfl

Nonwirable Output BOOL. (Profibus Diag_Data: Station_status_3: Bit 7)
If this bit is set, there is more diagnostic information than specified in ext_diag_data.
e.g. the slave will set this bit if there are more channel diagnostics than it can enter in its send buffer.
e.g. the DPM will set this bit if the slave sends more diagnostic information than it can enter in its diagnostic buffer.

### 10.3.20 Master_Add

Nonwirable Output INTEGER. Max: 128, Min: 0. (Profibus Diag_Data: Diag_Master_Add)
The address of the master which parameterised this slave. If none of the masters parameterised the slave, the slave inserts the address 255 here.

### 10.3.21 Ident

Nonwirable Output INTEGER. Max: 2147483647, Min: 0. (Profibus Diag_Data: Ident_Number)
The manufacturers identifier for the slave device. Can be used for verification or for exact identification of ext_diag_data structure.

### 10.3.22 Ext_Data

Nonwirable Output STRING. (Profibus Diag_Data: Ext_Diag_Data)
Contains ext_diag_data, the device specific part of the diagnostics. The first byte is a header containing its length, including the header byte itself. Consult the slave documentation for the interpretation of the other data.

### 10.3.23 Read_Trig

Nonwirable Input BOOL
Set this parameter to On to trigger a read of the diagnostic information. The existence of new diagnostic information is best determined from a {COM_Slv_Sta} function block.
The parameter value will return to 0 automatically when the transaction is complete.

## 11. HCOM_VARS

This class contains the function block types used to communicate with a remote network device:

Table 11-1: Function blocks in HCOM_VARS class

| Function block name | Description |
|---|---|
| COM_Bool | For communicating with a single discrete parameter. |
| COM_Dint | For communicating with a single integer parameter. |
| COM_Real | For communicating with a single real parameter. |
| COM_Str | For communicating with a single string parameter. |
| COM_SW | For communicating with a single status word. |
| COM_Dint_8 | Used to communicate with up to 8 integer parameters. |
| COM_Real_8 | Used to communicate with up to 8 real parameters. |
| COM_Bool_D | Communicating to a discrete parameter using the demand data protocol |
| COM_Dint_D | Communicating to an integer parameter using the demand data protocol |
| COM_Real_D | Communicating to a real parameter using the demand data protocol |
| COM_SW_D | Communicating to a status word using the demand data protocol |

## 12. COM_Var FF83, FF84,FF85,FF86,FF87

There are five function blocks to handle discrete parameters that have been configured as part of the cyclic data exchange. They are

Table 12-1: COM_Var type list

| Name | ID |
|---|---|
| COM_Dint | FF83 |
| COM_Real | FF84 |
| COM_Bool | FF85 |
| COM_Str | FF86 |
| COM_SW | FF87 |

Except for {New_Value} and {Value} type and for some differences in addressing format, the blocks are identical in function and use. This chapter describes these blocks as a generic class, highlighting the small differences that do exist.

### 12.1 Functional Description

Used to communicate with a single parameter in a remote network device. The parameter must have been configured as part of the input/output data exchange of the network. (For demand data transaction, use a COM_Var_D function block, see chapter 14 on page 81).

These blocks are very similar in function and use to the standard Remote_Var function blocks in the PC3000 function block library. The main difference is that, while a Remote_Var communicates with a slave instrument over a communications link, the COM_Var function blocks only read and write data into the dual port process data memory of the DPM module. It is the responsibility of that module to map the process data onto the connected slaves on the Profibus network.

It is not necessary to know the memory mapping of the DPM process data but a knowledge of the individual slaves is required to determine how the memory is used for each slave. For instance, the Eurotherm 2400 range has a software tool called GSD File Editor which is used to specify the memory mapping.

### 12.2 Parameter Diagram

Figure 12-1 describes the COM_Dint, COM_Real, COM_Bool and COM_Str function blocks. The only difference between them is the data type of {New_Value} and {Value}. For this reason, the data type is not shown in the diagram.

Figure 12-1: COM_Var block diagram

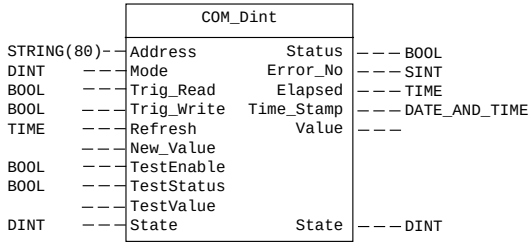
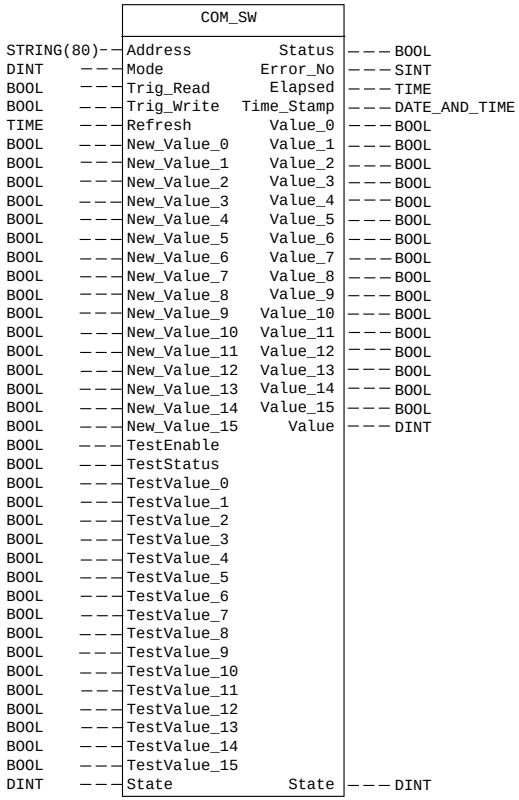



Figure 12-2 shows the COM_SW function block. The block is functionally identical to the COM_Dint but each bit in the data is brought out separately. The {Value} parameter itself is just the integer representation of the status word.

Figure 12-2: COM_SW block diagram



## 12.3 Parameter Descriptions

### 12.3.1 Address

Wirable Input STRING

The address of the remote variable expressed as a memory location in the slave instrument. It specifies the COM module responsible, the slave instrument address, the start memory address, the number of bytes to read/write or the bit number and the data format. This parameter may be changed dynamically to read/write different parameters.

There must be an instance of the appropriate COM driver (e.g. Profi_DPM) assigned to the slot number used in the address.

The string has four fields separated by colons (‘:’) e.g. ‘1:23:4:2’. The first three fields are exactly the same for all the function block types. The validity of field 4 depends on the function block type. These fields are described in detail in the following tables.

Table 12-2: COM_Var Address syntax (fields 1 -3)

| No | Field | Description |
|---|---|---|
| 1 | Slot address | The first character in the string will be a numeral in the range 1 - 5. It is the slot address of the COM module which will service this parameter.<br>It must be followed by a ‘:’. |
| 2 | Slave address | Specifies the slave instrument address, 1-128. The address may be prefixed with 'X'|'x' to indicate an address in hex notation. The field must be terminated by ':'.<br>For the Xcomms library slave address 0 is used to access data across slave boundaries |
| 3 | Offset | Specifies the offset of the data in the slave's input or output data. Which data area to use is determined by whether this function block is reading or writing. The value may be prefixed with 'X'|'x' to indicate it is hex.<br>The offset of the data is obtained from the slave instrument's documentation or, in the case of configurable slaves, from the configuration software e.g. GSD File Editor, the software for Eurotherm 2400, 2500 etc.<br>If the parameter is set to Read, it will read data from |

| | | |
|---|---|---|
| | | the Input Data space of the DPM module.<br>If the parameter is set to Write, it will write data to the Output Data space of the DPM module.<br>Note that a particular offset into the input data space will not usually contain the same parameter as the same offset into the output data space. E.g. it will usually be necessary to read and write the setpoint (say) of an instrument at different offsets.<br>Field 3 must be terminated by a ':'.<br>For the Xcomms library with slave address 0, the total offset in memory from the first slave is required to give an absolute offset. |
| 4 | Length or Bit number | This field specifies either a number of bytes to read/write or a bit number to read/write.<br>The field may have one of the following prefixes.<br>None Decimal number of bytes<br>X\|x Hex number of bytes<br>B\|b Decimal bit number<br>[B\|b][X\|x] Hex bit number<br>Bit numbers are not limited to 0 to 7. Higher bit numbers may be specified, as in a 32 bit status word for instance, and the driver will read/write the appro priate byte with the relevant bit number.<br>The following are the valid entries in this field for the different block types.<br>COM_Dint Either length (1, 2 or 4) or bit number<br>COM_Real Length only (1,2,4 or 8)<br>COM_Bool Bit number only<br>COM_Str Length only (≤ 255)<br>COM_SW Length only (1 or 2) |
| | Byte & Word ordering | Additional characters may be appended to field 4 to swap bytes and/or words in the data to account for different byte ordering.<br>~ Swaps pairs of bytes so that 123456 becomes 214365<br>^ Reverses the order of the words so that 123456 becomes 563412<br>One or both of these may be required to account for differences in memory usage as for Intel and |

| | | |
|---|---|---|
| | | Motorola. If both are used, the order is unimportant, the end result being that the order of the bytes is completely reversed so that 123456 becomes 654321.<br>Note: If either of these characters is used with an odd length, a null character is appended to the data before the ordering is applied. When a bit number is specified instead of a length, the length is calculated and then rounded up, if necessary. |
| | Data format | Different formats apply to different function block types. See Table 12-3. |

Table 12-3: COM_Var Address syntax. Data formats

| **Function Block Type** | **Field 4 syntax** |
|---|---|
| COM_Dint | No valid format characters |
| COM_Real | [E\|e][+\|]n - Where 0 ≤ n ≤ 9. Specifies an integer to be interpreted in exponent mode. Length must be 1\|2\|4.<br>[L\|l]m,n - Where m and n are real numbers. Specifies an integer to be interpreted in limits mode. Length must be 1\|2\|4. The integer limits to which m and n apply depend on the specified length.<br>Length 1: m ≡ 0, n ≡ 255 ($2^8$)<br>Length 2: m ≡ 0, n ≡ 65535 ($2^{16}$)<br>Length 4: m ≡ 0, n ≡ 4294967295 ($2^{32}$)<br>p IEEE 32 bit. Length must be 4.<br>P IEEE 64 bit. Length must be 8. |
| COM_Bool | No valid format characters |
| COM_Str | No valid format characters |
| COM_SW | No valid format characters |

The examples in Table 12-4 are all described as read transactions. The address for a write transaction is exactly the same structure.

Table 12-4: COM_Var Address examples

| Filter type | Address | Meaning |
|---|---|---|
| COM_Dint | 1:5:0:2 | From DPM module in slot 1, slave 5, offset 0, read 2 bytes. Interpret as a 16 bit integer. |
| COM_Dint | 1:5:2:b7 | From DPM module in slot 1, slave 5, offset 0, read 1 byte and extract bit 7. {Value} will be 1 or 0 depending on the state of the bit. |
| COM_Dint | 3:x20:6:4~ | From DPM module in slot 3, slave 32 (hex 20), offset 6, read 4 bytes and reverse order of pairs of bytes. Interpret as a 32 bit integer. |
| COM_Real | 2:23:8:2 | From DPM module in slot 2, slave 23, offset 8, read 2 bytes. Interpret as a 16 bit integer and convert to floating point. |
| COM_Real | 2:23:8:2E1 | From DPM module in slot 3, slave 23, offset 8, read 2 bytes. Interpret as a 16 bit integer, convert to floating point and divide by 10. |
| COM_Real | 4:97:16:4p | From DPM module in slot 4, slave 97, offset 16, read 4 bytes and interpret as an IEEE_32 floating point number. |
| COM_Bool | 1:2:4:b7 | From DPM module in slot 1, slave 2, offset 4, read 1 byte and extract bit 7. |
| COM_Bool | 1:2:4:b15 | From DPM module in slot 1, slave 2, offset 4, read 2 bytes and extract bit 15 (bit 7 in the byte at offset 5). |
| COM_Bool | 1:2:4:b15~ | From DPM module in slot 1, slave 2, offset 4, read 2 bytes, swap their order and extract bit 15 (bit 7 in the byte at offset 4). |
| COM_Bool | 1:2:4:bx38 | From DPM module in slot 1, slave 2, offset 4, read 8 bytes and extract bit 56 (hex 38. bit 0 in the byte at offset 11). |
| COM_Str | 2:5:6:8~^ | From DPM module in slot 2, slave 5, offset 6, read 8 bytes. Swap pairs of bytes and reverse words (to reverse the string completely). |
| COM_SW | 2:16:4:2 | From DPM module in slot 2, slave 16, offset 4, read 2 bytes and interpret as a 16 bit status word. |

### 12.3.2 Mode

Wirable Input ENUMERATED INTEGER
Determines the manner in which a communication transaction is initiated. Modes supported are:

Table 12-5: COM_Var Mode enumerations

| Value | Enumeration | Description |
|---|---|---|
| 0 | Demand | Read/Write on change of {State}, {Trig_Read} or {Trig_Write}. |
| 1 | R_Cont | Read continually at rate defined by {Refresh}. |
| 2 | W_Cont | Write continually at rate defined by {Refresh}. |
| 3 | Change | Write if {New_Value} <> {Value} but no more frequently than at the rate defined by {Refresh}. |

### 12.3.3 Trig_Read

Wirable Input BOOL
This parameter may be used to force a Read transaction. Read is initiated when {Trig_Read} is set to 'On'. It must be reset to 'Off' then 'On' to force another read. This parameter is useful when triggering a read transaction by soft wiring because the {State} parameter can not be wired.

### 12.3.4 Trig_Write

Wirable Input BOOL
This parameter may be used to force a Write transaction. Write is initiated when {Trig_Write} is set to 'On'. It must be reset to 'Off' then 'On' to force another write. . This parameter is useful when triggering a write transaction by soft wiring because the {State} parameter can not be wired.

### 12.3.5 Refresh

Wirable Input TIME Max: 1728000000, Min: 100
The rate at which 'R_Cont' and 'W_Cont' transactions will occur. As an example, if {Refresh} is set to '10s' and {Mode} is 'R_Cont' a new value will be read from the dual port memory every 10s.
This parameter does not affect the rate at which data is transferred to the slave across the network.
It also defines the fastest rate at which data will be written when {Mode} is 'Change'. i.e. for a write to occur, {New_Value} must be different to {Value}and the {Refresh} time must expire.

### 12.3.6 New_Value

Wirable Input. Parameter type and limits vary with block type as in Table 12-6.

Table 12-6: COM_Var New_Value parameter type and limits

| Block type | Parameter type | Max value | Min value |
|---|---|---|---|
| COM_Dint | INTEGER | 2147483646 | -2147483647 |
| COM_Real | REAL | 3.40282E+38 | -3.40282E+38 |
| COM_Bool | BOOL | 1 (On) | 0 (Off) |
| COM_Str | STRING(255) | N/A | N/A |
| COM_SW | BOOL<br>16 parameters :-<br>New_Value_0 to<br>New_Value_15 | 1 (On) | 0 (Off) |

The value that will be written to dual port memory on a Write transaction. If the transaction is successful, the value will be copied to {Value}.
COM_Str note: For COM_Str function blocks, the actual value written may depend on the length parameter encoded in {Address}. If {New_Value} is longer than the specified length, it will be truncated and only the specified length will be written. If {New_Value} is shorter than the specified length, only those characters in {New_Value} will be written: it will not be padded.

### 12.3.7 Status

Nonwirable Output BOOL
This parameter is set to Go (1) if {Error_No} is zero, otherwise it is set to NOGO (0).

### 12.3.8 Error_No

Nonwirable Output INTEGER Max: 255, Min: 0
This parameter indicates configuration and/or communication problems. It is only updated when a comms transaction is attempted. Driver errors are also reported in this parameter and paragraph 4.3.6 on page 5 describes those errors. Not all of the errors in the following table are relevant to all function block types.

Table 12-7: COM_Var error numbers

| Error_No | Error | Description |
|---|---|---|
| 0 | NO_ERROR | Either the block is functioning normally or no comms transaction has yet taken place. |
| 1 | CS_ERROR_NO_ADDRESS | The {Address} parameter is empty. |
| 11 | CS_ERROR_ILLEGAL_SLOT | The slot number specified in {Address} is outside the range 1 to 5. |
| 16 | CS_ERROR_NO_REMOTE_PARAMETER_SERVER | The slot number specified in {Address} does not have a COM driver attached to it. |
| 40 | ERROR_MODULE_NOT_PROFI_M | Reported by the COM driver. The hardware module detected at the specified {Slot_No} does not match the driver. |
| 42 | ERROR_MODULE_MEMORY_INCORRECT | The module size as read from the dual port memory does not match that specified in the COM driver. |
| 43 | ERROR_MODULE_DEV_NOT_COM | The installed Hilscher module type does not match the driver. |
| 44 | ERROR_MODULE_DEV_NOT_DPM | The installed Hilscher module model does not match the driver. |
| 45 | ERROR_MODULE_ID_NOT_COM | The installed Hilscher device ID does not match the driver. |
| 46 | ERROR_MODULE_NOT_READY | The module Ready flag is not set. |
| 48 | ERROR_SLOT_ALREADY_OCCUPIED | There is a conflict because two COM drivers are assigned to the same module slot. |

| Error_No | Error | Description |
|---|---|---|
| 49 | ERROR_INIT_WRITE_FAIL_TEST_ID | A dual port memory fault was detected during initialisation. |
| 50 | ERROR_INIT_WRITE_FAIL_ORIG_ID | A dual port memory fault was detected during initialisation. |
| 51 | ERROR_NO_COM_TABLE_INSTANTIATED | No instance of a COM_Table function block was found during initialisation. This block is mandatory. |
| 60 | ERROR_BUS_NOT_ENABLED | The bus is disabled at the COM driver. No comms transactions can be initiated. |
| 61 | ERROR_TIMEOUT_WAITING_FOR_COM | The Comms flag in the Hilscher module is not set indicating that there is no comms activity on the bus. Check DPM configuration, slave addresses, cabling etc. |
| 62 | ERROR_TIMEOUT_WAITING_FOR_ACCESS | The COM driver requested access to the dual port memory but it was not granted within the time_out period. A hard reset may clear the problem. |
| 63 | ERROR_TIMEOUT_WAITING_FOR_READY_FLAG | The module Ready flag is not set. Reset of the module cannot proceed. A power cycle may be necessary to clear the problem. |

| Error_No | Error | Description |
|---|---|---|
| 64 | ERROR_TIMEOUT_WAITING_FOR_RESET _FLAG_TO_CLEAR | The COM driver has set the Reset flag but the flag was not cleared. A power cycle may be necessary to clear the problem. |
| 65 | ERROR_TIMEOUT_WAITING_FOR_MODE_OPERATE | The module cannot get the configuration data during initialisation unless the DPM module is in OERATE mode. The module will not go to that state. A power cycle may be necessary to clear the problem. |
| 70 | ERROR_DO_CONFIG_UNKNOWN_STATE | The state machine controlling the module configuration is found to be in an unknown state. Configuration is not guaranteed to be complete. |
| 80 | ERROR_MBX_HNDLR_DEVICE_NOT_RUNNING | The message handler does not function unless the DPM module is running. Try enabling the bus at the driver. |
| 81 | ERROR_MBX_HNDLR_CLEARING_MBX | TimeOut while trying to clear the mailbox. DPM module needs resetting. |
| 82 | ERROR_MBX_HNDLR_TIMEOUT_WAITING_FOR_MAILBOX | TimeOut while waiting for the mailbox to become available. DPM module needs resetting. |

| Error_No | Error | Description |
|---|---|---|
| 83 | ERROR_MBX_HNDLR_TIMEOUT_WAITING_FOR_RESPONSE | TimeOut while waiting for a response to the last message. DPM module needs resetting. |
| 84 | ERROR_MBX_HNDLR_UNKNOWN_STATE | The function block state machine controlling the mailbox was found to be in some unknown state. It has been reinitialised. |
| 85 | ERROR_MBX_HNDLR_NOINIT | A timeout occurred while waiting for the mailbox system to initialise. The mailbox system is, therefore, not properly initialised. A power cycle may be necessary to clear the problem. |
| 128 | ERROR_START_OUTSIDE_MEMORY | The offset specified in the {Address} parameter is outside the data space for that specified slave.. |
| 129 | ERROR_END_OUTSIDE_MEMORY | The offset specified in the {Address} parameter is legal but the data length takes the transaction outside the valid memory area. |
| 130 | ERROR_INVALID_CHARACTER_IN_ADDRESS | The {Address} parameter contains in invalid character. |
| 131 | ERROR_NUMBER_OUT_OF_RANGE | The {Address} parameter contains an invalid exponent value for a real number with E format. The valid range is -9 to +9. |

| Error_No | Error | Description |
|---|---|---|
| 132 | ERROR_UNSUPPORTED_DATA_TYPE | The driver reports that the data type of the COM_Var is not supported. This is a system error and should be reported. |
| 135 | ERROR_INVALID_LENGTH | The {Address} parameter contains an invalid length for this parameter type. |
| 136 | ERROR_INVALID_ADDRESS_FOR_THIS_PARAMETER_TYPE | The {Address} parameter is invalid for this parameter type. |
| 137 | ERROR_INVALID_FLOAT_MODE | The {Address} parameter contains an invalid character where the mode character was expected. |
| 138 | ERROR_INVALID_BIT_NUMBER | The {Address} parameter contains a bit number specification which is invalid for this parameter type. |
| 139 | ERROR_NO_SLAVE_ADDRESS | No slave address in the {Address} parameter. |
| 140 | ERROR_SLAVE_ADDRESS_OUTSIDE_RANGE | Slave addresses are limited to 0 to 127. |
| 141 | ERROR_NO_ADDRESS_OFFSET | No offset specified in {Address} parameter. |
| 142 | ERROR_SLAVE_ADDRESS_NOT_CONFIGURED | The specified slave address is not configured in the net work. |
| 143 | ERROR_PORT_DELIMITER_NOT_FOUND | No colon after the slot number in {Address} parameter. |
| 144 | ERROR_SLAVE_DELIMITER_NOT_FOUND | No colon after the slave address in {Address} parameter. |

| Error_No | Error | Description |
|---|---|---|
| 145 | ERROR_ADDRESS_DELIMITER_NOT_FOUND | No colon after the offset address in {Address} parameter. |
| 146 | ERROR_NO_LENGTH_OR_BIT_NUMBER | No length or bit number specified in {Address} parameter. |
| 147 | ERROR_INVALID_MULTIELEMENT | The multielement variable has elements of different sizes. |
| 150 | ERROR_SLAVE_INACTIVE | The specified slave is not communicating. |
| 255 | ERROR_NONE_MSGE_REQUEST | This is not an error. It signifies that an address has been detected which contains the message request character and this error number is used to return the information to the driver. Message requests are used with the DPM mailbox to obtain configuration data, diagnostics etc. |

### 12.3.9 Elapsed

Nonwirable Output TIME Max: 1728000000, Min: 0

Indicates the time since the last successful transaction. It is only relevant in the 'R_Cont' and 'W_Cont' modes.

Time will increment from zero to the {Refresh} time. If the transaction fails, the value will be frozen at the time the time out occurred.

### 12.3.10 Time_Stamp

Nonwirable Output DATE_AND_TIME Max: 2147483646, Min: 0

The date and time of the last successful transaction. It is set when {State} reverts to 'OK'.

### 12.3.11 Value

Nonwirable Output . Parameter type and limits vary with function block type as in Table 12-8.

Table 12-8: COM_Var Value parameter type and limits

| Block type | Parameter type | Max value | Min value |
|---|---|---|---|
| COM_Dint | INTEGER | 2147483646 | -2147483647 |
| COM_Real | REAL | 3.40282E+38 | -3.40282E+38 |
| COM_Bool | BOOL | 1 (On) | 0 (Off) |
| COM_Str | STRING(255) | N/A | N/A |
| COM_SW | BOOL x 16 Value_0 to Value_15 and INTEGER x 1 Value | 1 (On)<br><br>65535 | 0 (Off)<br><br>0 |

This is the last value read from, or written to the dual port memory.
In the case of a successful write, {Value} is set equal to {New_Value}. If the transaction fails, {Value} will remain at the last value read or written.
For a COM_SW, the {Value} parameter is the integer value of the status word.

### 12.3.12 State

Nonwirable Input ENUMERATED INTEGER
Indicates the progress of a comms transaction. Its value will become ‘Read’ or ‘Write’ on a request. It will change to ‘Pending’ as the request is processed by the COM driver and then to ‘Ok’ or ‘Error’ when finished. If ‘Error’, the parameter {Err_No} will provide information about the cause of the error.
If {Mode} is set to 'Demand', this parameter can be used, by assignment in the sequence program, to initiate a read or write transaction. Typically, the value of {State} will be set to either 3 (Write) or 4 (Read) in an SFC step and the following transition will wait for the value of {State} to change to either 0 (Ok) or 2 (Error).
States are:

Table 12-9: COM_Var State enumerations

| Value | Enumeration | Description |
|---|---|---|
| 0 | Ok | Transcation was successful |
| 1 | Pending | Transcation in progress |
| 2 | Error | Transcation failed |
| 3 | Write | Write request |
| 4 | Read | Read request |

### 12.3.13 TestEnable

Wirable Input BOOL

Disconnects the function block from the comms network and enables testing of the application program to proceed using test values to simulate the comms.

With {TestEnable} set to On, comms requests are still submitted to the driver function block where the {Address} is verified. Errors will be returned by the driver if a DPM module is not present or not configured or if the {Address} does not verify. Provided that the driver does not return an error, the test mode functions are summarised in Table 12-10.

Table 12-10: Test mode functions

| | **TestStatus = Go** | **TestStatus = NOGO** |
|---|---|---|
| Write | {New_Value} -> {Value}<br>{Status} -> Go<br>{Error_No} -> 0<br>{State} -> 0 (* OK *)<br>{Time_Stamp} -> Current time | {New_Value} -> Not copied<br>{Status} -> NOGO<br>{Error_No} -> 255<br>{State} -> 2 (* Error *)<br>{Time_Stamp} -> Not set |
| Read | {Test_Value} -> {Value}<br>{Status} -> Go<br>{Error_No} -> 0<br>{State} -> 0 (* OK *)<br>{Time_Stamp} -> Current time | {Test_Value} -> Not copied<br>{Status} -> NOGO<br>{Error_No} -> 255<br>{State} -> 0 (* Error *)<br>{Time_Stamp} -> Not set |

### 12.3.14 TestStatus

Wirable Input BOOL

The value that will be copied to {Status} if {TestEnable} is set to On and either a Read or a Write is triggered.

The value of this parameter also determines the other actions that take place when {TestEnable} is On. See {TestEnable} for more details.

### 12.3.15 TestValue

Table 12-11: COM_Var TestValue parameter type and limits

| Block type | Parameter type | Max value | Min value |
|---|---|---|---|
| COM_Dint | INTEGER | 2147483646 | -2147483647 |
| COM_Real | REAL | 3.40282E+38 | -3.40282E+38 |
| COM_Bool | BOOL | 1 (On) | 0 (Off) |
| COM_Str | STRING(255) | N/A | N/A |
| COM_SW | BOOL x 16<br>TestValue_0 to TestValue_15 | 1 (On) | 0 (Off) |

The value that will be copied to {Value} if a Read is triggered with {TestEnable} set to On and {TestStatus} is Go.
COM_Str note: If {TestValue} is longer than the length specified in {Address}, it will be truncated. If it is shorter it will be right padded with last known values or otherwise null characters.

## 13.COM_Var_8 FF89, FF8A

There are two function blocks to handle multiple parameters that have been configured as part of the cyclic data exchange. They are

Table 13-1: COM_Var_8 type list

| Name | ID |
|---|---|
| COM_Dint_8 | FF89 |
| COM_Real_8 | FF8A |

Except for {New_Value} and {Value} type and for some differences in addressing format, the blocks are identical in function and use. This chapter describes these blocks as a generic class, highlighting the small differences that do exist.

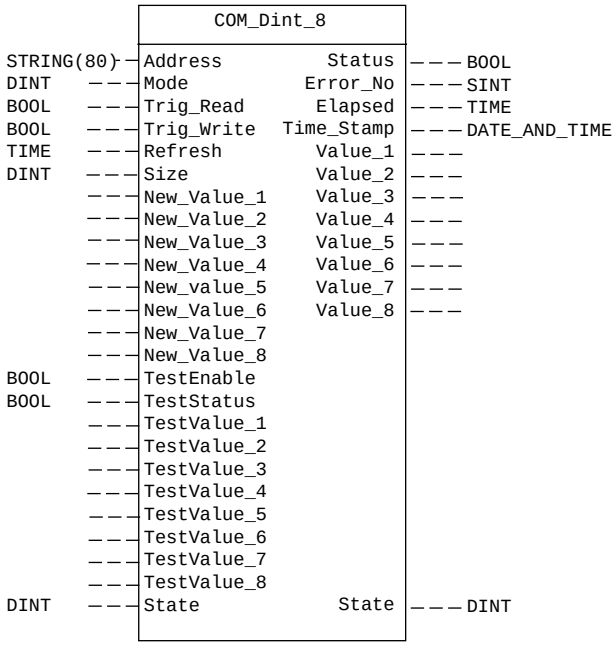
### 13.1 Functional Description

Used to communicate with up to 8 parameters in a remote network device. The parameters must have been configured as part of the input/output data exchange of the network. For demand data transaction, use a COM_Var_D function block.
The blocks are very similar in function and use to the standard Remote_Var function blocks in the PC3000 function block library. The main difference is that, while a Remote_Var blocks communicate with a slave instrument over a communications link, the COM_Var function blocks only read and write data into the dual port process data memory of the DPM module. It is the responsibility of that module to map the process data onto the connected slaves on the Profibus network.
It is not necessary to know the memory mapping of the DPM process data but a knowledge of the individual slaves is required to determine how the memory is used for each slave. For instance, the Eurotherm 2400 range has a software tool called GSD File Editor which is used to specify the memory mapping.

### 13.2 Parameter Diagram

Figure 13-1 describes both the COM_Dint_8 and the COM_Real_8 function blocks. The only difference between them is the data type of {New_Value} and {Value}. For this reason, the data type is not shown in the diagram.

Figure 13-1: COM_Var_8 block diagram



## 13.3 Parameter Descriptions

### 13.3.1 Address

Wirable Input STRING

The address of the remote variables expressed as a memory location in the slave instrument. It specifies the COM module responsible, the slave instrument address, the start memory address, the number of bytes to read/write for each parameter or the bit number and the data format. This parameter may be changed dynamically to read/write different parameters.

There must be an instance of the appropriate COM driver (e.g. Profi_DPM) assigned to the slot number used in the address.

The string has four fields separated by colons (':') e.g. '1:23:4:2'. The first three fields are exactly the same for both function block types. The validity of field 4 depends on the function block type. These fields are described in detail in the following tables.

Table 13-2: COM_Var_8 Address syntax (fields 1 -4)

| No | Field | Description |
|---|---|---|
| 1 | Slot address | The first character in the string will be a numeral in the range 1 - 5. It is the slot address of the COM module which will service these parameters.<br>It must be followed by a ':'. |
| 2 | Slave address | Specifies the slave instrument address, 1-128. The address may be prefixed with 'X'\|'x' to indicate an address in hex notation. All parameters in one block must originate in the same slave. The field must be terminated by ':'.<br>For the Xcomms library slave address 0 is used to access data across slave boundaries |
| 3 | Offset | Specifies the starting offset of the data in the slave's input or output data. Which data area to use is determined by whether this function block is reading or writing. The value may be prefixed with 'X'\|'x' to indicate it is hex.<br>The data at this offset is read into {Value_1}. Offsets for subsequent values are calculated by adding the length from field 4.<br>The offset of the data is obtained from the slave instrument's documentation or, in the case of configurable slaves, from the configuration software e.g. GSD File Editor, the software for Eurotherm 2400, 2500 etc. |

| No | Field | Description |
|---|---|---|
| 3 | | If the block is set to Read, it will read data from the Input Data space of the DPM module.<br>If the block is set to Write, it will write data to the Output Data space of the DPM module.<br>Note that a particular offset into the input data space will not usually contain the same parameters as the same offset into the output data space. E.g. it will usually be necessary to read and write the setpoint (say) of an instrument at different offsets.<br>Field 3 must be terminated by a ':'.<br>For the Xcoms libraray with slave address 0, the total offset in memory from the first slave is required to give an absolute offset. |
| 4 | Length or Bit number | This field specifies either a number of bytes to read/write or a bit number to read/write. The field may have one of the following prefixes.<br>None - Decimal number of bytes<br>X\|x - Hex number of bytes<br>B\|b - Decimal bit number<br>[B\|b][X\|x] - Hex bit number<br>Bit numbers are not limited to 0 to 7. Higher bit numbers may be specified, as in a 32 bit status word for instance, and the driver will read/write the appropriate byte with the relevant bit number.<br>Note: Bits 0-7 are in byte 1, bits 8-15 are in byte 2 etc. Depending on the byte ordering of the data, status words will often have the most significant byte first (i.e. bits 8-15 will be in byte 1). In order to address the bits in the logical way, it will be necessary to use the byte swap option '~' as described below.<br>The following are the valid entries in this field for the different block types.<br>COM_Dint_8 Either length (1, 2 or 4) or bit number<br>COM_Real_8 Length only (1, 2, 4 or 8) |

| No | Field | Description |
|---|---|---|
| | Byte and Word ordering | Additional characters may be appended to field 4 to swap bytes and/or words in the data to account for different byte ordering.<br>~ Swaps pairs of bytes so that 123456 becomes 214365<br>^ Reverses the order of the words so that 123456 becomes 563412<br>One or both of these may be required to account for differences in memory usage as for Intel and Motorola. If both are used, the order is unimportant, the end result being that the order of the bytes is completely reversed so that 123456 becomes 654321.<br>Note: If either of these characters is used with an odd length, a null character is appended to the data before the ordering is applied. When a bit number is specified instead of a length, the length is calculated and then rounded up, if necessary. |
| | Data format | Different formats apply to different function block types. See Table 13-3. |

Table 13-3: COM_Var_8 Address syntax. Data formats

| Function Block Type | Field 4 syntax |
|---|---|
| COM_Dint_8 | No valid format characters |
| COM_Real_8 | [E\|e][+\|]n - Where $0 \leq n \leq 9$. Specifies an integer to be interpreted in exponent mode. Length must be 1\|2\|4.<br>[L\|l]m,n - Where m and n are real numbers. Specifies an integer to be interpreted in limits mode. Length must be 1\|2\|4. The integer limits to which m and n apply depend on the specified length.<br>Length 1: $m \equiv 0$, $n \equiv 255$ ($2^8$)<br>Length 2: $m \equiv 0$, $n \equiv 65535$ ($2^{16}$)<br>Length 4: $m \equiv 0$, $n \equiv 4294967295$ ($2^{32}$)<br>p - IEEE 32 bit. Length must be 4.<br>P - IEEE 64 bit. Length must be 8. |

The examples in Table 13-4 are all described as read transactions. The address for a write transaction is exactly the same structure.

Table 13-4: COM_Var_8 Address examples

| FB type | Address | Meaning |
|---|---|---|
| COM_Dint_8 | 1:5:0:2 | From DPM module in slot 1, slave 5:-<br>at offset 0, read 2 bytes → Value_1<br>at offset 2, read 2 bytes → Value_2<br>at offset 4, read 2 bytes → Value_3<br>...<br>at offset 14, read 2 bytes → Value_8<br>Interpret values as a 16 bit integers. |
| COM_Dint_8 | 1:5:2:b7 | From DPM module in slot 1, slave 5:-<br>at offset 0, read 1 byte and extract bit 7→ Value_1<br>at offset 1, read 1 byte and extract bit 7→ Value_2<br>at offset 2, read 1 byte and extract bit 7→ Value_3<br>...<br>at offset 7, read 1 byte and extract bit 7 →Value_8<br>{Value_N} will be 1 or 0 depending on the state of the bit.<br>Note: See also the next example. |
| COM_Dint_8 | 1:5:2:b15~ | From DPM module in slot 1, slave 5:-<br>at offset 0, read 2 bytes, swap & and extract bit 7 → Value_1<br>at offset 1, read 2 bytes, swap & and extract bit 7 → Value_2<br>at offset 2, read 2 bytes, swap & and extract bit 7 → Value_3<br>...<br>at offset 7, read 2 bytes, swap & and extract bit 7 → Value_8<br>{Value_N} will be 1 or 0 depending on the state of the bit.<br>Note: This instruction will actually read bit 7 of the first byte, same as the previous example, but will step on 2 bytes each time instead of 1 byte. |

| FB type | Address | Meaning |
|---|---|---|
| COM_Dint_8 | 3:x20:6:4~ | From DPM module in slot 3, slave 32 (hex 20):-<br>at offset 6, read 4 bytes and swap bytes→ Value_1<br>at offset 10, read 4 bytes and swap bytes<br>→ Value_2<br>at offset 14, read 4 bytes and swap bytes<br>→ Value_3<br>...<br>at offset 36, read 4 bytes and swap bytes<br>→ Value_8<br>Interpret as a 32 bit integers. |
| COM_Real_8 | 2:23:8:2 | From DPM module in slot 2, slave 23:-<br>at offset 8, read 2 bytes → Value_1<br>at offset 10, read 2 bytes → Value_2<br>at offset 12, read 2 bytes → Value_3<br>...<br>at offset 22, read 2 bytes → Value_8<br>Interpret as a 16 bit integers and convert to floating point. |
| COM_Real_8 | 2:23:8:2E1 | From DPM module in slot 2, slave 23:-<br>at offset 8, read 2 bytes → Value_1<br>at offset 10, read 2 bytes → Value_2<br>at offset 12, read 2 bytes → Value_3<br>...<br>at offset 22, read 2 bytes → Value_8<br>Interpret as 16 bit integers, convert to floating point and divide by 10. |
| COM_Real_8 | 4:97:16:4p | From DPM module in slot 4, slave 97:-<br>at offset 16, read 4 bytes → Value_1<br>at offset 20, read 4 bytes → Value_2<br>at offset 24, read 4 bytes → Value_3<br>...<br>at offset 44, read 4 bytes → Value_8<br>Interpret values as IEEE_32 floating point numbers. |

### 13.3.2 Mode

Wirable Input ENUMERATED INTEGER
Determines the manner in which a communication transaction is initiated. Modes supported are:

Table 13-5: COM_Var_8 Mode enumerations

| Value | Enumeration | Description |
|---|---|---|
| 0 | Demand | Read/Write on change of {State}, {Trig_Read} or {Trig_Write}. |
| 1 | R_Cont | Read continually at rate defined by {Refresh}. |
| 2 | W_Cont | Write continually at rate defined by {Refresh}. |
| 3 | Change | Write if {New_Value} <> {Value} but no more frequently than at the rate defined by {Refresh}. |

### 13.3.3 Trig_Read

Wirable Input BOOL
This parameter may be used to force a Read transaction. Read is initiated when {Trig_Read} is set to 'On'. It must be reset to 'Off' then 'On' to force another read. This parameter is useful when triggering a read transaction by soft wiring because the {State} parameter can not be wired.

### 13.3.4 Trig_Write

Wirable Input BOOL
This parameter may be used to force a Write transaction. Write is initiated when {Trig_Write} is set to 'On'. It must be reset to 'Off' then 'On' to force another write. . This parameter is useful when triggering a write transaction by soft wiring because the {State} parameter can not be wired.

### 13.3.5 Refresh

Wirable Input TIME Max: 1728000000, Min: 100
The rate at which 'R_Cont' and 'W_Cont' transactions will occur. As an example, if {Refresh} is set to '10s' and {Mode} is 'R_Cont' a new value will be read from the dual port memory every 10s.
It also defines the fastest rate at which transactions will occur when {Mode} is 'Change' even if {New_Value} has changed.

### 13.3.6 Size

Wirable Input INTEGER Max: 8, Min: 0
Specifies how many of the 8 values are actually used. The driver checks that reads and writes are not outside the process data of the slave. To avoid writing to or reading from illegal areas of memory, the number of parameters written/read can be limited by setting this parameter to a value less than 8.

### 13.3.7 New_Value_1 to New_Value_8

Wirable Input. Parameter type and limits vary with block type as in Table 13-6.

Table 13-6: COM_Var_8 New_Value data types

| Block type | Parameter type | Max value | Min value |
|---|---|---|---|
| COM_Dint_8 | INTEGER | 2147483646 | -2147483647 |
| COM_Real_8 | REAL | 3.40282E+38 | -3.40282E+38 |

The values that will be written to dual port memory on a Write transaction. If the transaction is successful, the values will be copied to {Value_1} to {Value_8}.

### 13.3.8 Status

Nonwirable Output BOOL
This parameter is set to Go (1) if {Error_No} is zero, otherwise it is set to NOGO (0).

### 13.3.9 Error_No

Nonwirable Output INTEGER Max: 255, Min: 0
This parameter indicates configuration and/or communication problems. It is only updated when a comms transaction is attempted. Driver errors are also reported in this parameter and 4.3.6 on page 5 describes those errors.

Table 13-7: COM_Var_8 error numbers

| Error_No | Error | Description |
|---|---|---|
| 0 | NO_ERROR | Either the block is functioning normally or no comms transaction has yet taken place. |
| 1 | CS_ERROR_NO_ADDRESS | The {Address} parameter is empty |

| Error_No | Error | Description |
|---|---|---|
| 11 | CS_ERROR_ILLEGAL_SLOT | The slot number specified in {Address} is outside the range 1 to 5. |
| 16 | CS_ERROR_NO_REMOTE_PARAMETER_SERVER | The slot number specified in {Address} does not have a COM driver attached to it. |
| 40 | ERROR_MODULE_NOT_PROFI_M | Reported by the COM driver. The hardware module detected at the specified {Slot_No} does not match the driver. |
| 42 | ERROR_MODULE_MEMORY_INCORRECT | The module size as read from the dual port memory does not match that specified in the COM driver. |
| 43 | ERROR_MODULE_DEV_NOT_COM | The installed Hilscher module type does not match the driver. |
| 44 | ERROR_MODULE_DEV_NOT_DPM | The installed Hilscher module model does not match the driver. |
| 45 | ERROR_MODULE_ID_NOT_COM | The installed Hilscher device ID does not match the driver. |
| 46 | ERROR_MODULE_NOT_READY | The module Ready flag is not set. |
| 48 | ERROR_SLOT_ALREADY_OCCUPIED | There is a conflict because two COM drivers are assigned to the same module slot. |
| 49 | ERROR_INIT_WRITE_FAIL_TEST_ID | A dual port memory fault was detected during initialisation. |
| 50 | ERROR_INIT_WRITE_FAIL_ORIG_ID | A dual port memory fault was detected during initialisation. |

| Error_No | Error | Description |
|---|---|---|
| 51 | ERROR_NO_COM_TABLE_INSTANTIATED | No instance of a COM_Table function block was found during initialisation. This block is mandatory. |
| 60 | ERROR_BUS_NOT_ENABLED | The bus is disabled at the COM driver. No comms transactions can be initiated. |
| 61 | ERROR_TIMEOUT_WAITING_FOR_COM | The Com flag in the Hilscher module is not set indicating that there is no comms activity on the bus. Check DPM configuration, slave addresses, cabling etc. |
| 62 | ERROR_TIMEOUT_WAITING_FOR_ACCESS | The COM driver requested access to the dual port memory but it was not granted within the time-out period. A hard reset may clear the problem. |
| 63 | ERROR_TIMEOUT_WAITING_FOR_READY_FLAG | The module Ready flag is not set. Reset of the module cannot proceed. A power cycle may be necessary to clear the problem. |
| 64 | ERROR_TIMEOUT_WAITING_FOR_RESET _FLAG_TO_CLEAR | The COM driver has set the Reset flag but the flag was not cleared. A power cycle may be necessary to clear the problem. |

| Error_No | Error | Description |
|---|---|---|
| 65 | ERROR_TIMEOUT_WAITING_FOR_MODE_OPERATE | The module cannot get the configuration data during initialisation unless the DPM module is in OERATE mode. The module will not go to that state. A power cycle may be necessary to clear the problem. |
| 70 | ERROR_DO_CONFIG_UNKNOWN_STATE | The state machine controlling the module configuration is found to be in an unknown state. Configuration is not guaranteed to be complete. |
| 80 | ERROR_MBX_HNDLR_DEVICE_NOT_RUNNING | The message handler does not function unless the DPM module is running. Try enabling the bus at the driver. |
| 81 | ERROR_MBX_HNDLR_CLEARING_MBX | TimeOut while trying to clear the mailbox. DPM module needs resetting. |
| 82 | ERROR_MBX_HNDLR_TIMEOUT_WAITING_FOR_MAILBOX | TimeOut while waiting for the mailbox to become available. DPM module needs resetting. |
| 83 | ERROR_MBX_HNDLR_TIMEOUT_WAITING_FOR_RESPONSE | TimeOut while waiting for a response to the last message. DPM module needs resetting. |

| Error_No | Error | Description |
|---|---|---|
| 84 | ERROR_MBX_HNDLR_UNKNOWN_STATE | The function block state machine controlling the mailbox was found to be in some unknown state. It has been reinitialised. |
| 85 | ERROR_MBX_HNDLR_NOINIT | A timeout occurred while waiting for the mailbox system to initialise. The mailbox system is, therefore, not properly initialised. A power cycle may be necessary to clear the problem. |
| 128 | ERROR_START_OUTSIDE_MEMORY | The offset specified in the {Address} parameter is outside the data space for that specified slave. |
| 129 | ERROR_END_OUTSIDE_MEMORY | The offset specified in the {Address} parameter is legal but the data length takes the transaction outside the valid memory area. |
| 130 | ERROR_INVALID_CHARACTER_IN_ADDRESS | The {Address} parameter contains in invalid character. |
| 131 | ERROR_NUMBER_OUT_OF_RANGE | The {Address} parameter contains an invalid exponent value for a real number with E format. The valid range is 9 to +9. |
| 132 | ERROR_UNSUPPORTED_DATA_TYPE | The driver reports that the data type of the COM_Var is not supported. This is a system error and should be reported. |

| Error_No | Error | Description |
|---|---|---|
| 135 | ERROR_INVALID_LENGTH | The {Address} parameter contains an invalid length for this parameter type. |
| 136 | ERROR_INVALID_ADDRESS_FOR_THIS_PARAMETER_TYPE | The {Address} parameter is invalid for this parameter type. |
| 137 | ERROR_INVALID_FLOAT_MODE | The {Address} parameter contains an invalid character where the mode character was expected. |
| 138 | ERROR_INVALID_BIT_NUMBER | The {Address} parameter contains a bit number specification which is invalid for this parameter type. |
| 139 | ERROR_NO_SLAVE_ADDRESS | No slave address in the {Address} parameter. |
| 140 | ERROR_SLAVE_ADDRESS_OUTSIDE_RANGE | Slave addresses are limited to 0 to 127. |
| 141 | ERROR_NO_ADDRESS_OFFSET | No offset specified in {Address} parameter. |
| 142 | ERROR_SLAVE_ADDRESS_NOT_CONFIGURED | The specified slave address is not configured in the net work. |
| 143 | ERROR_PORT_DELIMITER_NOT_FOUND | No colon after the slot number in {Address} parameter. |
| 144 | ERROR_SLAVE_DELIMITER_NOT_FOUND | No colon after the slave address in {Address} parameter. |
| 145 | ERROR_ADDRESS_DELIMITER_NOT_FOUND | No colon after the offset address in {Address} parameter. |
| 146 | ERROR_NO_LENGTH_OR_BIT_NUMBER | No length or bit number specified in {Address} parameter. |

| Error_No | Error | Description |
|---|---|---|
| 147 | ERROR_INVALID_MULTIELEMENT | The multielement variable has elements of different sizes. |
| 150 | ERROR_SLAVE_INACTIVE | The specified slave is not communicating. |
| 255 | ERROR_NONE_MSGE_REQUEST | This is not an error.<br>It signifies that an address has been detected which contains the message request character and this error number is used to return the information to the driver.<br>Message requests are used with the DPM mailbox to obtain configuration data, diagnostics etc. |

### 13.3.10 Elapsed

Nonwirable Output TIME Max: 1728000000, Min: 0
Indicates the time since the last successful transaction. It is only relevant in the 'R_Cont' and 'W_Cont' modes.
Time will increment from zero to the {Refresh} time. If the transaction fails, the value will be frozen at the time the time out occurred.

### 13.3.11 Time_Stamp

Nonwirable Output DATE_AND_TIME Max: 2147483646, Min: 0
The date and time of the last successful transaction. It is set when {State} reverts to 'OK'.

### 13.3.12 Value_1 to Value_8

Nonwirable Output. . Parameter type and limits vary with block type as in Table 13-8.

Table 13-8: COM_Var_8 Value data types

| Block type | Parameter type | Max value | Min value |
|---|---|---|---|
| COM_Dint_8 | INTEGER | 2147483646 | -2147483647 |
| COM_Real_8 | REAL | 3.40282E+38 | -3.40282E+38 |

This is the last value read from, or written to the dual port memory.
In the case of a successful write, {Value_N} is set equal to {New_Value_N}. If the transaction fails, {Value_N} will remain at the last value read or written.

### 13.3.13 State

Nonwirable Input ENUMERATED INTEGER
Indicates the progress of a comms transaction. Its value will become ‘Read’ or ‘Write’ on a request. It will change to ‘Pending’ as the request is processed by the COM driver and then to ‘Ok’ or ‘Error’ when finished. If ‘Error’, the parameter {Err_No} will provide information about the cause of the error.
If {Mode} is set to 'Demand', it can be used, by assignment in the sequence program, to initiate a read or write transaction. States are:

Table 13-9: COM_Var_8 State enumerations

| Value | Enumeration | Description |
|---|---|---|
| 0 | Ok | Transaction was successful. |
| 1 | Pending | Transaction failed. |
| 2 | Error | Transaction in progress. |
| 3 | Write | Write request. |
| 4 | Read | Read request. |

### 13.3.14 TestEnable

Wirable Input BOOL

Disconnects the function block from the comms network and enables testing of the application program to proceed using test values to simulate the comms.

With {TestEnable} set to On, comms requests are still submitted to the driver function block where the {Address} is verified. Errors will be returned by the driver if a DPM module is not present or not configured or if the {Address} does not verify.

Provided that the driver does not return an error, the test mode functions are summarised in Table 13-10: Test mode functions.

Table 13-10: Test mode functions

| | TestStatus = Go | TestStatus = NOGO |
|---|---|---|
| Write | {New_Value_N} -> {Value_N}<br>{Status} -> Go<br>{Error_No} -> 0<br>{State} -> 0 (* OK *)<br>{Time_Stamp} -> Current time | {New_Value_N} -> Not copied<br>{Status} -> NOGO<br>{Error_No} -> 255<br>{State} -> 2 (* Error *)<br>{Time_Stamp} -> Not set |
| Read | {Test_Value_N} -> {Value_N}<br>{Status} -> Go<br>{Error_No} -> 0<br>{State} -> 0 (* OK *)<br>{Time_Stamp} -> Current time | {Test_Value_N} -> Not copied<br>{Status} -> NOGO<br>{Error_No} -> 255<br>{State} -> 0 (* Error *)<br>{Time_Stamp} -> Not set |

### 13.3.15 TestStatus

Wirable Input BOOL

The value that will be copied to {Status} if {TestEnable} is set to On and either a Read or a Write is triggered.

The value of this parameter also determines the other actions that take place when {TestEnable} is On. See {TestEnable} for more details.

### 13.3.16 TestValue_1 to TestValue_8

Table 13-11: COM_Var_8 TestValue parameter type and limits

| Block type | Parameter type | Max value | Min value |
|---|---|---|---|
| COM_Dint_8 | INTEGER | 2147483646 | -2147483647 |
| COM_Real_8 | REAL | 3.40282E+38 | -3.40282E+38 |

The value that will be copied to {Value_N} if a Read is triggered with {TestEnable} set to On and {TestStatus} is Go.

## 14. COM_Var_D FF8B, FF8C, FF8D, FF8E

There are four function blocks to handle discrete parameter comms using the Demand Data protocol. They are

Table 14-1: COM_Var_D type list

| Name | ID |
|---|---|
| COM_Bool_D | FF8B |
| COM_SW_D | FF8C |
| COM_Real_D | FF8D |
| COM_Dint_D | FF8E |

Except for {New_Value} and {Value} type and for some differences in addressing format, the blocks are identical in function and use. This chapter describes these blocks as a generic class, highlighting the small differences that do exist.

### 14.1 Functional Description

Used to communicate with a single parameter in a remote network device using the Demand Data protocol. This protocol can be used to communicate with parameters that have not been preconfigured as part of the network process data exchange. Because the Demand Data protocol is a sequence of writes and reads with pauses to wait for the slave response, the time taken for data to be transferred using this method is considerably longer than for configured network data.
This is especially so for COM_Bool_D when writing because the block does a read before write and two sets of transactions are, therefore, required.
At least 13 program execution cycles are needed but, with varying slave response times, the total time taken may be considerably longer. If the LockOut feature is enabled, no other comms transaction can take place with the device during this time.
Different Profibus devices use the Demand Data in different ways but the Eurotherm range of 2400, 2500, TU stacks and drives all use a similar mechanism and this block may be used for those instruments. The actual demand data mechanism is described in Reference 3, page 118 and the examples there can be used in more general cases.
The information for this block is taken from the Series 2000 ProfibusDP Communications Handbook, HA026290.
The parameter mnemonics are listed in the relevant manual for the slave device. e.g. for Eurotherm Controls Series 2000, the manual identified above contains the relevant information.

## 14.2 Parameter Diagram

Figure 14-1 describes the COM_Dint_D, COM_Real_D and COM_Bool_D function blocks. The only difference between them is the data type of {New_Value} and {Value}. For this reason, the data type is not shown in the diagram.

Figure 14-1: COM_Var_D block diagram

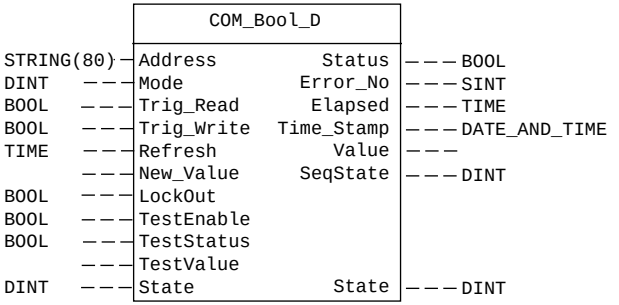
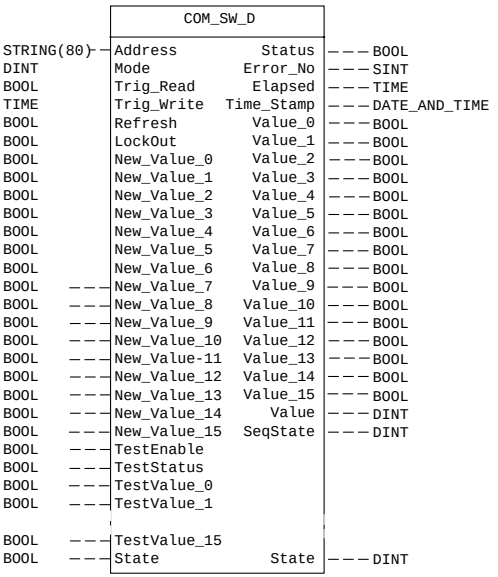



Figure 14-2 shows the COM_SW_D function block. The block is functionally identical to the other COM_Var_D blocks but each bit in the data is brought out separately. The {Value} parameter itself is just the integer representation of the status word.

Figure 14-2: COM_SW_D block diagram



## 14.3 Parameter Descriptions

### 14.3.1 Address

Wirable Input STRING

The address of the remote variable. It specifies the COM module responsible, the slave address and the parameter ID. Optionally, it can specify a bit number and a data format. This parameter may be changed dynamically to read/write different parameters.

There must be an instance of the appropriate COM driver (e.g. Profi_DPM) assigned to the slot number used in the address.

The string has three fields separated by colons (‘:’) e.g. ‘1:23:4’. The three fields are exactly the same for all the function block types but the validity of various options in field 3 depends on the function block type. These fields are described in detail in the following tables.

Table 14-2: COM_Var_D Address syntax

| No | Field | Description |
|---|---|---|
| 1 | Slot address | The first character in the string will be a numeral in the range 1 - 5. It is the slot address of the COM module which will service this parameter.<br>It must be followed by a ‘:’. |
| 2 | Slave address | Specifies the slave instrument address, 1-128. The address may be prefixed with 'X'\|'x' to indicate an address in hex notation. The field must be terminated by ':'. |
| 3 | Parameter ID | Specifies the parameter ID that is to be read or written. This information must be obtained from the documentation for the slave instrument in question. |
| | Bit number | If the required data is a bit in a parameter, the bit number is appended to the parameter ID. It must begin with ‘B’\|’b’ and may be followed by ‘X’\|’x’ to indicate a hex value.<br>Only bit numbers 0 to 15 are valid because the block only supports 16 bit data types.<br>The following are the valid entries in this field for the different block types.<br>COM_Dint Bit number is valid<br>COM_Real Bit number is invalid<br>COM_Bool Bit number is required<br>COM_SW Bit number is invalid |

| No | Field | Description |
|---|---|---|
| 3 | Byte ordering | An additional character may be appended to field 3 to swap bytes in the data to account for different byte ordering.<br>~ Swaps bytes so that 12 becomes 21<br>This may be required to account for differences in memory usage as for Intel and Motorola.<br>Note: Bits 0-7 are in byte 1, bits 8-15 are in byte 2 etc. Depending on the byte ordering of the data, status words will often have the most significant byte first (i.e. bits 8-15 will be in byte 1). In order to address the bits in the logical way, it will be necessary to use the byte swap option. |
| | Data format | Different formats apply to different function block types. See Table 14-3. |

Table 14-3: COM_Var_D Address format characters

| Function Block Type | Field 3 syntax |
|---|---|
| COM_Dint_D | No valid format characters |
| COM_Real_D | [E\|e][+\|]n - Where 0 ≤ n ≤ 9. Specifies an integer to be interpreted in exponent mode. Length must be 1\|2\|4.<br>[L\|l]m,n - Where m and n are real numbers. Specifies an integer to be interpreted in limits mode. Length must be 1\|2\|4. The integer limits to which m and n apply depend on the specified length.<br>Length 1: m ≡ 0, n ≡ 255 ($2^8$)<br>Length 2: m ≡ 0, n ≡ 65535 ($2^{16}$)<br>Length 4: m ≡ 0, n ≡ 4294967295 ($2^{32}$)<br>p - IEEE 32 bit. Length must be 4.<br>P - IEEE 64 bit. Length must be 8. |
| COM_Bool_D | No valid format characters |
| COM_SW_D | No valid format characters |

Table 14-4: COM_Var_D Address examples

| FB Type | Address | Meaning |
|---|---|---|
| COM_Bool_D | 1:3:76b15 | From DPM module in slot 1, slave 3, read parameter ID 76 and extract bit 15. |
| COM_Real_D | 2:74:3E1 | From DPM module in slot 2, slave 74, read parameter ID 3. Interpret the data as an integer and divide by 10. |
| COM_Dint_D or COM_SW_D | 3:21:234~ | From DPM module in slot 3, slave 21, read parameter ID 234. Swap the byte order and interpret as an integer Or as a bit field. |

### 14.3.2 Mode

Wirable Input ENUMERATED INTEGER
Determines the manner in which a communication transaction is initiated. Modes supported are:

Table 14-5: COM_Var_D Mode enumerations

| Value | Enumeration | Description |
|---|---|---|
| 0 | Demand | Read/Write on change of {State}, {Trig_Read} or {Trig_Write}. |
| 1 | R_Cont | Read continually at rate defined by {Refresh}. |
| 2 | W_Cont | Write continually at rate defined by {Refresh}. |
| 3 | Change | Write if {New_Value} <> {Value} but no more frequently than at the rate defined by {Refresh}. |

### 14.3.3 Trig_Read

Wirable Input BOOL
This parameter may be used to force a Read transaction. Read is initiated when {Trig_Read} is set to 'On'. It must be reset to 'Off' then 'On' to force another read. This parameter is useful when triggering a read transaction by soft wiring because the {State} parameter can not be wired.

### 14.3.4 Trig_Write

Wirable Input BOOL

This parameter may be used to force a Write transaction. Write is initiated when {Trig_Write} is set to 'On'. It must be reset to 'Off' then 'On' to force another write. . This parameter is useful when triggering a write transaction by soft wiring because the {State} parameter can not be wired.

### 14.3.5 Refresh

Wirable Input TIME Max: 1728000000, Min: 100

The rate at which 'R_Cont' and 'W_Cont' transactions will occur. As an example, if {Refresh} is set to '10s' and {Mode} is 'R_Cont' a new value will be read from the dual port memory every 10s.

It also defines the fastest rate at which transactions will occur when {Mode} is 'Change' even if {New_Value} has changed.

### 14.3.6 LockOut

Wirable Input BOOL

Setting this parameter to On forces comms requests from other COM_Vars to be rejected by the driver while the demand data exchange is in progress i.e. until the {State} of this block returns to 0 (Ok) or 2 (Error).

This is necessary because some slave instruments use part of the normal data exchange memory space for the demand data transaction and, therefore, parameters in the process data map may become corrupted during a demand data transaction. For instance, the Eurotherm TU range of stacks uses exclusively demand data for writing parameters and the demand data overlaps the process data space. This parameter should be set On for applications involving TU stacks.

### 14.3.7 New_Value

Wirable Input. Parameter type and limits vary with block type as in Table 14-6.

Table 14-6: COM_Var_D New_Value parameter type and limits

| Block type | Parameter type | Max value | Min value |
|---|---|---|---|
| COM_Dint_D | INTEGER | 2147483646 | -2147483647 |
| COM_Real_D | REAL | 3.40282E+38 | -3.40282E+38 |
| COM_Bool_D | BOOL<br>15 parameters:-<br>New_Value_0 to<br>New_Value_15 | 1 (On) | 0 (Off) |

The value that will be written to dual port memory on a Write transaction. If the transaction is successful, the value will be copied to {Value}.

### 14.3.8 Status

Nonwirable Output BOOL
This parameter is set to Go (1) if {Error_No} is zero, otherwise it is set to NOGO (0).

### 14.3.9 Error_No

Nonwirable Output INTEGER Max: 257, Min: 0
This parameter indicates configuration and/or communication problems. It is only updated when a comms transaction is attempted. Driver errors are also reported in this parameter and 4.3.6 on page 5 describes those errors.

Table 14-7: COM_Var_D error codes

| No | Error | Description |
|---|---|---|
| 0 | NO_ERROR | Either the block is functioning normally or no comms transaction has yet taken place. |
| 1 | CS_ERROR_NO_ADDRESS | The {Address} parameter is empty. |
| 11 | CS_ERROR_ILLEGAL_SLOT | The slot number specified in {Address} is outside the range 1 to 5. |
| 16 | CS_ERROR_NO_REMOTE_PARAMETER_SERVER | The slot number specified in {Address} does not have a COM driver attached to it. |
| 40 | ERROR_MODULE_NOT_PROFI_M | Reported by the COM driver. The hardware module detected at the specified {Slot_No} does not match the driver. |
| 42 | ERROR_MODULE_MEMORY_INCORRECT | The module size as read from the dual port memory does not match that specified in the COM driver. |
| 43 | ERROR_MODULE_DEV_NOT_COM | The installed Hilscher module type does not match the driver. |

| No | Error | Description |
|---|---|---|
| 44 | ERROR_MODULE_DEV_NOT_DPM | The installed Hilscher module model does not match the driver. |
| 45 | ERROR_MODULE_ID_NOT_COM | The installed Hilscher device ID does not match the driver. |
| 46 | ERROR_MODULE_NOT_READY | The module Ready flag is not set. |
| 48 | ERROR_SLOT_ALREADY_OCCUPIED | There is a conflict because two COM drivers are assigned to the same module slot. |
| 49 | ERROR_INIT_WRITE_FAIL_TEST_ID | A dual port memory fault was detected during initialisation. |
| 50 | ERROR_INIT_WRITE_FAIL_ORIG_ID | A dual port memory fault was detected during initialisation. |
| 51 | ERROR_NO_COM_TABLE_INSTANTIATED | No instance of a COM_Table function block was found during initialisation. This block is mandatory. |
| 60 | ERROR_BUS_NOT_ENABLED | The bus is disabled at the COM driver. No comms transactions can be initiated. |
| 61 | ERROR_TIMEOUT_WAITING_FOR_COM | The Com flag in the Hilscher module is not set indicating that there is no comms activity on the bus. Check DPM configuration, slave addresses, cabling etc. |

| No | Error | Description |
|---|---|---|
| 62 | ERROR_TIMEOUT_WAITING_FOR_ACCESS | The COM driver requested access to the dual port memory but it was not granted within the time-out period. A hard reset may clear the problem. |
| 63 | ERROR_TIMEOUT_WAITING_FOR_READY_FLAG | The module Ready flag is not set. Reset of the module cannot proceed. A power cycle may be necessary to clear the problem. |
| 64 | ERROR_TIMEOUT_WAITING_FOR_RESET _FLAG_TO_CLEAR | The COM driver has set the Reset flag but the flag was not cleared. A power cycle may be necessary to clear the problem. |
| 65 | ERROR_TIMEOUT_WAITING_FOR_MODE_OPERATE | The module cannot get the configuration data during initialisation unless the DPM module is in OPERATE mode. The module will not go to that state. A power cycle may be necessary to clear the problem. |
| 70 | ERROR_DO_CONFIG_UNKNOWN_STATE | The state machine controlling the module configuration is found to be in an unknown state. Configuration is not guaranteed to be complete. |

| No | Error | Description |
|---|---|---|
| 80 | ERROR_MBX_HNDLR_DEVICE_NOT_RUNNING | The message handler does not function unless the DPM module is running. Try enabling the bus at the driver. |
| 81 | ERROR_MBX_HNDLR_CLEARING_MBX | TimeOut while trying to clear the mailbox. DPM module needs resetting. |
| 82 | ERROR_MBX_HNDLR_TIMEOUT_WAITING_FOR_MAILBOX | TimeOut while waiting for the mailbox to become available. DPM module needs resetting. |
| 83 | ERROR_MBX_HNDLR_TIMEOUT_WAITING_FOR_RESPONSE | TimeOut while waiting for a response to the last message. DPM module needs resetting. |
| 84 | ERROR_MBX_HNDLR_UNKNOWN_STATE | The function block state machine controlling the mailbox was found to be in some unknown state. It has been reinitialised. |
| 85 | ERROR_MBX_HNDLR_NOINIT | A timeout occurred while waiting for the mailbox system to initialise. The mailbox system is, therefore, not properly initialised. A power cycle may be necessary to clear the problem. |
| 128 | ERROR_START_OUTSIDE_MEMORY | The offset specified in the {Address} parameter is outside the data space for that specified slave. |

| No | Error | Description |
|---|---|---|
| 129 | ERROR_END_OUTSIDE_MEMORY | The offset specified in the {Address} parameter is legal but the data length takes the transaction outside the valid memory area. |
| 130 | ERROR_INVALID_CHARACTER_IN_ADDRESS | The {Address} parameter contains in invalid character. |
| 131 | ERROR_NUMBER_OUT_OF_RANGE | The {Address} parameter contains an invalid exponent value for a real number with E format. The valid range is -9 to +9. |
| 132 | ERROR_UNSUPPORTED_DATA_TYPE | The driver reports that the data type of the COM_Var is not supported. This is a system error and should be reported. |
| 135 | ERROR_INVALID_LENGTH | The {Address} parameter contains an invalid length for this parameter type. |
| 136 | ERROR_INVALID_ADDRESS_FOR_THIS_PARAMETER_TYPE | The {Address} parameter is invalid for this parameter type. |
| 137 | ERROR_INVALID_FLOAT_MODE | The {Address} parameter contains an invalid character where the mode character was expected. |
| 138 | ERROR_INVALID_BIT_NUMBER | The {Address} parameter contains a bit number specification which is invalid for this parameter type. |
| 139 | ERROR_NO_SLAVE_ADDRESS | No slave address in the {Address} parameter. |

| No | Error | Description |
|---|---|---|
| 140 | ERROR_SLAVE_ADDRESS_OUTSIDE_RANGE | Slave addresses are limited to 0 to 127. |
| 141 | ERROR_NO_ADDRESS_OFFSET | No offset specified in {Address} parameter. |
| 142 | ERROR_SLAVE_ADDRESS_NOT_CONFIGURED | The specified slave address is not configured in the net work. |
| 143 | ERROR_PORT_DELIMITER_NOT_FOUND | No colon after the slot number in {Address} parameter. |
| 144 | ERROR_SLAVE_DELIMITER_NOT_FOUND | No colon after the slave address in {Address} parameter. |
| 145 | ERROR_ADDRESS_DELIMITER_NOT_FOUND | No colon after the offset address in {Address} parameter. |
| 146 | ERROR_NO_LENGTH_OR_BIT_NUMBER | No length or bit number specified in {Address} parameter. |
| 147 | ERROR_INVALID_MULTIELEMENT | The multi element variable has elements of different sizes. |
| 148 | ERROR_NO_PARAMETER TAG | There is no parameter tag number in the {Address} parameter. |
| 150 | ERROR_SLAVE_INACTIVE | The specified slave is not communicating. |
| 255 | ERROR_NONE_MSGE_REQUEST | This is not an error. It signifies that an address has been detected which contains the message request character and this error number is used to return the information to the driver. |

| No | Error | Description |
|---|---|---|
| | | Message requests are used with the DPM mailbox to obtain configuration data, diagnostics etc. |
| 256 | ERROR_INVALID_TAG_NUMBER | This error is generated when the slave returns an error code 0. It indicates an invalid tag number. |
| 257 | ERROR_SLAVE_RESPONSE | This error is returned by the slave when an invalid command is received. e.g. Writing to a read only parameter OR Writing an out of range value. |

### 14.3.10 Elapsed

Nonwirable Output TIME Max: 1728000000, Min: 0
Indicates the time since the last successful transaction. It is only relevant in the 'R_Cont' and 'W_Cont' modes.
Time will increment from zero to the {Refresh} time. If the transaction fails, the value will be frozen at the time the time out occurred.

### 14.3.11 Time_Stamp

Nonwirable Output DATE_AND_TIME Max: 2147483646, Min: 0
The date and time of the last successful transaction. It is set when {State} reverts to 'OK'.

### 14.3.12 Value

Nonwirable Output. . Parameter type and limits vary with function block type as in Table 14-8.

Table 14-8: COM_Var Value parameter type and limits

| Block type | Parameter type | Max value | Min value |
|---|---|---|---|
| COM_Dint | INTEGER | 2147483646 | -2147483647 |
| COM_Real | REAL | 3.40282E+38 | -3.40282E+38 |
| COM_Bool | BOOL | 1 (On) | 0 (Off) |
| COM_SW | BOOL x 16 Value_0 to Value_15 and INTEGER x 1 Value | 1 (On)<br><br>65535 | 0 (Off)<br><br>0 |

This is the last value read from, or written to the dual port memory.
In the case of a successful write, {Value} is set equal to {New_Value}. If the transaction fails, {Value} will remain at the last value read or written.

### 14.3.13 State

Nonwirable Input ENUMERATED INTEGER
Indicates the progress of a comms transaction. Its value will become 'Read' or 'Write' on a request. It will change to 'Pending' as the request is processed by the COM driver and then to 'Ok' or 'Error' when finished. If 'Error', the parameter {Err_No} will provide information about the cause of the error.
If {Mode} is set to 'Demand', it can be used, by assignment in the sequence program, to initiate a read or write transaction. States are:

Table 14-9: COM_Var_D State enumerations

| Value | Enumeration | Description |
|---|---|---|
| 0 | Ok | Transaction was successful. |
| 1 | Pending | Transaction failed. |
| 2 | Error | Transaction in progress. |
| 3 | Write | Write request. |
| 4 | Read | Read request. |

### 14.3.14 SeqState

Nonwirable Output ENUMERATED INTEGER
A diagnostic parameter. The Demand Data exchange is a sequence of reads and writes to and from dual port memory with pauses to wait for the slave response. This parameter indicates which step in the sequence is currently active. Not all states are relevant to all parameter types.

Table 14-10: COM_Var_D SeqState enumerations

| Value | Enumeration | Meaning |
|---|---|---|
| 0 | SEQ_STATE_IDLE | The sequence is not currently executing. No request. |
| 1 | SEQ_STATE_INIT | A request has been received. The sequence is initialising. |
| 2 | SEQ_STATE_ERROR | An error has occurred during the data exchange. See {Error_No} for details. |

| Value | Enumeration | Meaning |
|---|---|---|
| 3 | SEQ_STATE_DONE | The sequence is complete and is tidying up. |
| 4 | SEQ_STATE_WRT_CLEAR | Write a null command to clear any previous data. |
| 5 | SEQ_STATE_WAIT_WRT_CLEAR | Wait for the null command to write. |
| 6 | SEQ_STATE_WRT_REQ | Write the command to the send data area. |
| 7 | SEQ_STATE_WAIT_WRT_REQ | Wait for the command write to complete. |
| 8 | SEQ_STATE_READ_REQ | Read the slave response from the input data area. |
| 9 | SEQ_STATE_WAIT_READ_REQ | Wait for the read slave response to complete. |
| 10 | SEQ_STATE_READ_DATA | Read the data from the response. |
| 11 | SEQ_STATE_WAIT_READ_DATA | Wait for the data read to complete. |
| 12 | SEQ_STATE_WRT_TERM | Write a null command to terminate the transaction. |
| 13 | SEQ_STATE_WAIT_WRT_TERM | Wait for the terminating null command to write. |
| 14 | SEQ_STATE_UNLOCK | Issue a read command to unlock the driver. |
| 15 | SEQ_STATE_WAIT_UNLOCK | Wait for the unlock command to complete. |
| 16 | SEQ_STATE_WRT_CUR_DATA | Copy the parameter data in the input buffer to the output buffer. |
| 17 | SEQ_STATE_WAIT_WRT_CUR_DATA | Wait for the parameter data copy to complete. |
| 18 | SEQ_STATE_WRT_NEW_DATA | Overwrite parameter data in output buffer with new data. |
| 19 | SEQ_STATE_WAIT_WRT_NEW_DATA | Wait for the parameter data write to complete. |
| 20 | SEQ_STATE_WRT_REQ_NEW_DATA | Put a write command in the output buffer. |

| Value | Enumeration | Meaning |
|---|---|---|
| 21 | SEQ_STATE_WAIT_WRT_REQ_NEW_DATA | Wait for the write command to complete. |
| 22 | SEQ_STATE_READ_REQ_NEW_DATA | Read the slave response to the write command. |
| 23 | SEQ_STATE_WAIT_READ_REQ_NEW_DATA | Wait for the read slave response to complete. |

### 14.3.15 TestEnable

Wirable Input BOOL
Disconnects the function block from the comms network and enables testing of the application program to proceed using test values to simulate the comms.
With {TestEnable} set to On, comms requests are still submitted to the driver function block where the {Address} is verified. Errors will be returned by the driver if a DPM module is not present or not configured or if the {Address} does not verify.
Provided that the driver does not return an error, the test mode functions are summarised in Table 14-11.

Table 14-11: Test mode functions

| | TestStatus = Go | TestStatus = NOGO |
|---|---|---|
| Write | {New_Value} -> {Value}<br>{Status} -> Go<br>{Error_No} -> 0<br>{State} -> 0 (* OK *)<br>{Time_Stamp} -> Current time | {New_Value} -> Not copied<br>{Status} -> NOGO<br>{Error_No} -> 255<br>{State} -> 2 (* Error *)<br>{Time_Stamp} -> Not set |
| Read | {Test_Value} -> {Value}<br>{Status} -> Go<br>{Error_No} -> 0<br>{State} -> 0 (* OK *)<br>{Time_Stamp} -> Current time | {Test_Value} -> Not copied<br>{Status} -> NOGO<br>{Error_No} -> 255<br>{State} -> 0 (* Error *)<br>{Time_Stamp} -> Not set |

### 14.3.16 TestStatus

Wirable Input BOOL

The value that will be copied to {Status} if {TestEnable} is set to On and either a Read or a Write is triggered.

The value of this parameter also determines the other actions that take place when {TestEnable} is On. See {TestEnable} for more details.

### 14.3.17 TestValue

Table 14-12: COM_Var_D TestValue parameter type and limits

| Block type | Parameter type | Max value | Min value |
|---|---|---|---|
| COM_Dint_D | INTEGER | 2147483646 | -2147483647 |
| COM_Real_D | REAL | 3.40282E+38 | -3.40282E+38 |
| COM_Bool_D | BOOL | 1 (On) | 0 (Off) |
| COM_SW_D | BOOL x 16<br>TestValue_0 to TestValue_15 | 1 (On) | 0 (Off) |

The value that will be copied to {Value} if a Read is triggered with {TestEnable} set to On and {TestStatus} is Go.

## 15. HCOS_VARS

This class contains the function block types used for communication from a remote network device:

Table 15-1: Function blocks in HCOS_VARS class

| Function block name | Description |
|---|---|
| COS_Bool | For communicating with a single discrete parameter. |
| COS_Dint | For communicating with a single integer parameter. |
| COS_Real | For communicating with a single real parameter. |
| COS_Str | For communicating with a single string parameter. |
| COS_SW | For communicating with a single status word. |
| COS_Dint_8 | Used to communicate with up to 8 integer parameters. |
| COS_Real_8 | Used to communicate with up to 8 real parameters. |

# 16. COS_Var FF97, FF98, FF99, FF9A, FF9B

There are five function blocks to handle discrete parameters that have been configured as part of the cyclic data exchange. They are

Table 16-1: COS_Var type list

| Name | ID |
|---|---|
| COS_Dint | FF97 |
| COS_Real | FF98 |
| COS_Bool | FF99 |
| COS_Str | FF9A |
| COS_SW | FF9B |

Except for the data type of {Value} and for some differences in addressing format, the blocks are identical in function and use. This chapter describes these blocks as a generic class, highlighting the small differences that do exist.

## 16.1 Functional Description

These blocks are used to write data to and extract data from the Output (Produced) and Input (Consumed) data areas of the fieldbus interface. A fieldbus slave (e.g. DeviceNet DNS) module must be present in the rack and supported by the appropriate driver function block (e.g. DevNet_S). These function blocks read and write data values that have been configured as part of the cyclic data exchange on the network.

Each of these blocks registers itself dynamically with the driver function block identified by the {Address} parameter. Up to 512 COS_Var and COS_Var_8 function blocks can be registered with a single driver. The registration is constantly checked and, therefore, the {Address} parameter can be changed at run-time to re-allocate a COS_Var to a different driver or to a different memory location in the same driver. Note: this is unlike the standard PC3000 Slv_Vars which register at start-up and cannot be changed dynamically.

Each COS_Var can be specified as either Input (Consumed), Output (Produced) or Off. This property is also dynamic.

At each execution of the driver function block, it scans through its list of registered COS_Vars and COS_Var_8's and

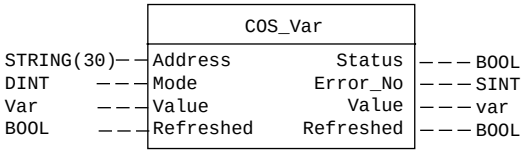
- updates those registered as Input, with new values from the Input Process Data area
- updates the Output Process Data area with new data from those registered as Output
- ignores any which are registered as Off.

The driver, therefore, becomes more heavily loaded with increasing numbers of COS_Vars and COS_Var_8's and care should be exercised when assigning it to an unnecessarily fast task.

## 16.2 Parameter Diagram

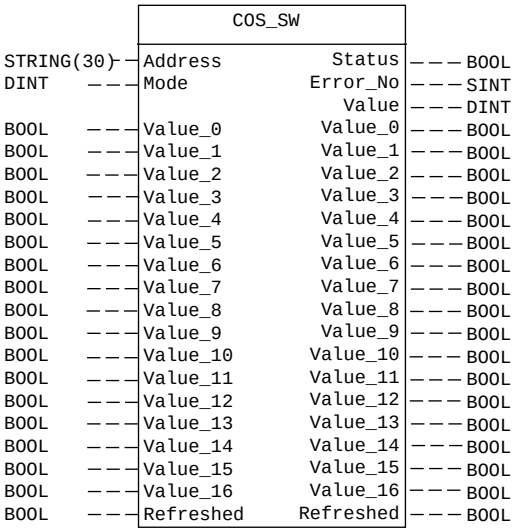
The figure describes the COS_Dint, COS_Real, COS_Bool and COS_Str function blocks. The only difference between them is the data type of {Value} which has, therefore, been omitted from the diagram.

Figure 16-1: COS_Var generic block diagram




The following figure shows the COM_SW function block. The block is functionally identical to a COS_Dint but each bit in the data is brought out separately. The {Value} parameter itself is just the integer representation of the data.

Figure 16-2: COS_SW block diagram



## 16.3 Parameter Descriptions

### 16.3.1 Address

Wirable Input STRING

The address of the slave data expressed as a memory location in the Process Data area of the DeviceNet interface. It specifies the COM module location, the offset into the Process Data, the number of bytes to read or write (or a bit number) and the data format. The parameter can be changed dynamically to reassign the block to a different driver or a different memory location on the same driver.

There must be an instance of the appropriate COM driver (e.g. DevNet_S) assigned to slot number specified in the {Address}.

The parameter has three fields separated by colons (':') e.g. '1:4:2'. Format information is appended to field 3 and varies between block types.

Table 16-2: COS_Var Address syntax

| Field No | Field Name | Description |
|---|---|---|
| 1 | Slot address | The first character must be a number in the range 1 – 5. It is the slot address of the COM module which will service this parameter. An appropriate COM driver must be assigned to this slot.<br>Field 1 must be followed by a ':'. |
| 2 | Offset | Specifies the offset of the data in the slave's input or output data. Which data area to use is determined by whether this function block is set as Input or Output. The value may be prefixed with 'X'\|'x' to indicate it is hex.<br>The offset of the data is entirely in the hands of the programmer. Care should be taken to ensure there is no overlap, unless it is deliberate, because there is NO checking of this.<br>If the parameter {Mode} is set to Input, it will read data from the Input Data area of the COM module.<br>If the parameter {Mode} is set to Output, it will write data to the Output Data area of the COM module.<br>Field 2 must be terminated by a ':'. |

| Field No | Field Name | Description |
|---|---|---|
| 3 | Data length/Bit number | Field 3 specifies either a number of bytes to read/write or a bit number to read/write. The value can be preceded by any of the following:<br>None - Decimal number of bytes<br>X\|x - Hex number of bytes<br>B\|b - Decimal bit number<br>[B\|b][X\|x] - Hex bit number<br><br>Bit numbers are not limited to 0 to 7. Higher bit numbers may be specified, as in a 32 bit status word for example, and the driver will read/write the appropriate byte with the relevant bit number.<br>The following are valid entries in this field for the different block types.<br>COM_Dint Length (1, 2 or 4) or a bit number<br>COM_Real Length only (1, 2, 4 or 8)<br>COM_Bool Bit number only<br>COM_Str Length only (≤ 255)<br>COM_SW Length only (1 or 2) |
| | Byte and Word ordering | Additional characters may be appended to filed 3 to swap bytes and/or words in the data.<br>~ Swaps pairs of bytes so that '123456' becomes '214365'<br>^ Reverses the word order so that '123456' becomes '563412'<br>One or both of these may be required to account for differences in memory usage as for Intel and Motorola. If both are used, the order is unimportant, the end result is that the order of the bytes is completely reversed so that '123456' becomes '654321' |

| | | |
|---|---|---|
| | | Note: If either of these characters is used with an odd length, a null character is appended to the data being the ordering is applied. When a bit number is specified instead of a length, the length is calculated and rounded up as necessary. |
| | Data format | Different formats apply to different function block types. |

The following table describes the valid format characters for the various block types.

Table 16-3: COS_Var Address format characters

| **Function block type** | **Format characters** |
|---|---|
| COS_Dint | No valid format characters |
| COS_Real | [E\|e][+\|-]n Where 0 ≤ n ≤ 9. Specifies exponent mode. Length must be 1\|2\|4<br>[L\|l]m,n Where m and n are real numbers. Specifies limits mode. Length must be 1\|2\|4.<br>The integer limits to which m and n apply depend on the specified length.<br>Length 1: m ≡ 0 n ≡ 256 ($2^8$)<br>Length 2: m ≡ 0 n ≡ 65536 ($2^{16}$)<br>Length 4: m ≡ 0 n ≡ 4294967295 ($2^{32}$)<br>p (lower case) IEEE 32 bit floating point. Length must be 4<br>P (upper case) IEEE 64 bit floating point. Length must be 8 |
| COS_Bool | No valid format characters |
| COS_Str | No valid format characters |
| COS_SW | No valid format characters |

Some example addresses. They are described as read transactions but the addresses would be the same for write transactions.

Table 16-4: COS_VAR Address examples

| FB Type | Address | Meaning |
|---|---|---|
| COS_Dint | 1:0:2 | From module in slot 1, read 2 bytes at offset 0 and interpret as an integer. |
| | 1:2:b7 | From module in slot 1, read 1 byte at offset 2 and extract bit 7. Value will be integer 1 or 0. |
| | 3:x20:4~ | From module in slot 3, read 4 bytes at offset 32 (20h), swap byte order and interpret the result as an integer. |
| COS_Real | 2:8:2E1 | From module in slot 2, read 2 bytes at offset 8, interpret as an integer, divide by 10 and convert to floating point. |
| | 1:10:4p | From module in slot 1, read 4 bytes at offset 10 and interpret as an IEEE floating point number. |
| | 2:0:4~L0,100 | From module in slot 2, read 4 bytes at offset 0, swap the byte order and interpret as an integer. Scale the integer to the range 0 – 100 (0 = 0 and $2^{32}$ = 100) and convert to floating point. |
| | 1:0:2 | From module in slot 1, read 2 bytes at offset 0, interpret as an integer and convert to floating point. (Same as E0) |
| COS_Bool | 1:0:b0 | From module in slot 1, read 1 byte at offset 0 and extract bit 0. Value will be integer 1 or 0. |
| | 1:2:b15~ | From module in slot 1, read 2 bytes at offset 2, swap them and extract bit 15. Value will be integer 1 or 0. |
| | 3:x10:bx12 | From module in slot 3, read 3 bytes at offset 16 (10h)and extract bit 18 (12h). Value will be integer 1 or 0. |
| COS_Str | 2:6:10~^ | From module in slot 2, read 10 bytes at offset 6, swap bytes and reverse words i.e. (reverse byte order completely). |
| COS_SW | 3:2:1 | From module in slot 3, read 1 byte at offset 2 and interpret as a bit field. Only Value_0 to Value_7 will be valid in the function block. |

### 16.3.2 Mode

Wirable Input ENUMERATED INTEGER
Specifies whether this variable is Off, Consumed or Produced.

Table 16-5: COS_Var Mode enumerations

| Value | Enumeration | Description |
|---|---|---|
| 0 | Off | This parameter is not being updated by the associated slave comms driver. This allows individual variables to be disconnected from the network. The slave comms driver's {RunState} parameter disconnects all variables. |
| 1 | Input | The value of this parameter is being written by the associated slave comms driver from data received over the network. Parameters such as setpoints and outputs can be wired FROM this parameter. |
| 2 | Output | The value of this parameter is being written to the associated slave comms driver for transmission over the network. Parameters such as process values and other inputs can be wired TO this variable. |

### 16.3.3 Status

Nonwirable Output BOOL
This parameter is set to GO (1) if {Err_No} is zero. Otherwise, this parameter is set to NOGO (0).

### 16.3.4 Error_No

Nonwirable Output INTEGER. Max: 255, Min: 0
This parameter indicates configuration and/or communication problems.

Table 16-6: COS_Var Error_No values

| Value | Enumeration | Description |
|---|---|---|
| 0 | NO_ERROR | Either the block is functioning normally or no comms trans action has yet taken place. |
| 1 | CS_ERROR_NO_ADDRESS | The {Address} parameter is empty. |
| 11 | CS_ERROR_ILLEGAL_SLOT | The slot number specified in {Address} is outside the range 1 to 5. |
| 16 | CS_ERROR_NO_REMOTE_PARAMETER_SERVER | The slot number specified in {Address} does not have a COM driver attached to it. |
| 40 | ERROR_MODULE_NOT_NETWORK_MOTHER_BOARD | The module detected at the specified {Slot_No} is not a Network Card. |
| 42 | ERROR_MODULE_MEMORY_INCORRECT | The module size as read from the dual port memory does not match that specified in the COM driver. |
| 43 | ERROR_MODULE_DEV_NOT_COM | The installed Hilscher module type does not match the driver. |
| 45 | ERROR_MODULE_ID_NOT_COM | The installed Hilscher device ID does not match the driver. |
| 46 | ERROR_MODULE_NOT_READY | The module Ready flag is not set. |
| 48 | ERROR_SLOT_ALREADY_OCCUPIED | There is a conflict because two COM drivers are assigned to the same module slot. |

| Value | Enumeration | Description |
|---|---|---|
| 49 | ERROR_INIT_WRITE_FAIL_TEST_ID | A dual port memory fault was detected during initialisation. |
| 50 | ERROR_INIT_WRITE_FAIL_ORIG_ID | A dual port memory fault was detected during initialisation. |
| 51 | ERROR_NO_COM_TABLE_INSTANTIATED | No instance of a COM_Table function block was found during initialisation. This block is mandatory. |
| 52 | ERROR_TIMEOUT_WAITING_FOR_DEVICE_TO_RUN | Comms module is not running. |
| 55 | ERROR_MODULE_DEV_NOT_DNS | Comms module is not a DeviceNet slave. |
| 56 | ERROR_MAX_NO_COS_BLOCKS_EXCEEDED | Max number of COS function blocks (512) per driver exceeded. |
| 57 | ERROR_DRIVER_DOES_NOT_SUPPORT_SLAVE | The driver in the specified slot does not support COS variables. |
| 58 | ERROR_INVALID_INPUT_DATA_SIZE | The specified Consumed data size exceeds the physical memory of the module. |
| 59 | ERROR_INVALID_OUTPUT_DATA_SIZE | The specified Produced data size exceeds the physical memory of the module. |
| 61 | ERROR_TIMEOUT_WAITING_FOR_COM | The Com flag in the Hilscher module is not set indicating that there is no comms activity on the bus. |

| Value | Enumeration | Description |
|---|---|---|
| 62 | ERROR_TIMEOUT_WAITING_FOR_ACCESS | The COM driver requested access to the dual port memory but it was not granted within the time_out period. |
| 63 | ERROR_TIMEOUT_WAITING_FOR_READY_FLAG | The module Ready flag is not set. Reset of the module cannot proceed. |
| 64 | ERROR_TIMEOUT_WAITING_FOR_RESET _FLAG_TO_CLEAR | The COM driver has set the Reset flag but the flag was not cleared. |
| 65 | ERROR_TIMEOUT_WAITING_FOR_MODE_OPERATE | The module cannot get the configuration data during initialisation unless the relevant task is running on the module. This task is not running. |
| 66 | ERROR_TIMEOUT_WAITING_FOR_INIT_FLAG_TO_CLEAR | An {Init} has been performed on the comms slave driver and has timed out. |
| 70 | ERROR_DO_CONFIG_UNKNOWN_STATE | The state machine controlling the module configuration is found to be in an unknown state. Configuration is not guaranteed to be complete. |
| 128 | ERROR_START_OUTSIDE_MEMORY | The {Address} parameter contains an invalid offset address. |

| Value | Enumeration | Description |
|---|---|---|
| 129 | ERROR_END_OUTSIDE_MEMORY | The {Address} parameter contains a valid offset address but the data length takes the transaction out side the valid memory area. |
| 130 | ERROR_INVALID_CHARACTER_IN_ADDRESS | The {Address} parameter contains in invalid character. |
| 131 | ERROR_NUMBER_OUT_OF_RANGE | The {Address} parameter contains an invalid exponent value for a real number with E format. The valid range is -9 to +9. |
| 132 | ERROR_UNSUPPORTED_DATA_TYPE | The driver reports that the data type of the COS_var is not supported. This is a system error and should be reported. |
| 133 | ERROR_NO_LOW_LIMIT | An 'L' format has been specified without a low limit. |
| 134 | ERROR_NO_HIGH_LIMIT | An 'L' format has been specified without a high limit. |
| 135 | ERROR_INVALID_LENGTH | The {Address} parameter contains an invalid length for this parameter type. |
| 136 | ERROR_INVALID_ADDRESS_FOR_THIS_PARAMETER_TYPE | The {Address} parameter is invalid for this parameter type. |
| 137 | ERROR_INVALID_FLOAT_MODE | The {Address} parameter contains an invalid character where the mode character was expected. |

| Value | Enumeration | Description |
|---|---|---|
| 138 | ERROR_INVALID_BIT_NUMBER | The {Address} parameter contains a bit number specification which is invalid for this parameter type. |
| 141 | ERROR_NO_ADDRESS_OFFSET | No offset specified in {Address} parameter. |
| 143 | ERROR_PORT_DELIMITER_NOT_FOUND | No colon after the port number in {Address} parameter. |
| 145 | ERROR_ADDRESS_DELIMITER_NOT_FOUND | No colon after the offset address in {Address} parameter. |
| 146 | ERROR_NO_LENGTH_OR_BIT_NUMBER | No length or bit number specified in {Address} parameter. |
| 147 | ERROR_INVALID_MULTIELEMENT | The multi-element variable has elements of different sizes. |

### 16.3.5 Value

Wirable Input. Parameter type and limits vary with function block type as in the table.

Table 16-7: COS_Var data types and limits

| Block type | Parameter type | Max value | Min value |
|---|---|---|---|
| COS_Dint | INTEGER | 2147483646 | -2147483647 |
| COS_Real | REAL | 3.40282E+38 | -3.40282E+38 |
| COS_Bool | BOOL | 1 (On) | 0 (Off) |
| COS_Str | STRING | Maximum length 255 | |
| COS_SW | 16 off BOOL Value_0 to Value_15<br>1 off INTEGER Value | 1 (On)<br>65535 | 0 (Off)<br>0 (Off) |

The value that is being read or written provided that the {Mode} is not Off, that {Error_No} is zero and the associated slave comms driver has {RunState} set to Run.
For a COS_SW, the {Value} parameter is the integer value of the status word.

### 16.3.6 Refreshed

Nonwirable Input BOOL
Only relevant if {Mode} is set to Input. Indicates that the {Value} has just been changed by the driver. The user program must reset this value to Off in order to detect the next change.

## 17. COS_Var_8 FF9C, FF9D

There are two function blocks to handle multiple parameters that have been configured as part of the cyclic data exchange. They are

Table 171: COS_Var_8 block types

| Name | ID |
|---|---|
| COS_Dint_8 | FF9C |
| COS_Real_8 | FF9D |

Except for the data type of {Value} and for some differences in addressing format, the blocks are identical in function and use. This chapter describes these blocks as a generic class, highlighting the small differences that do exist.

### 17.1 Functional Description

These blocks are used to write data to and extract data from the Output (Produced) and Input (Consumed) data areas of the fieldbus interface. A fieldbus slave (e.g. DeviceNet DNS) module must be present in the rack and supported by the appropriate driver function block (e.g. DevNet_S). These function blocks read and write up to eight data values that have been configured as part of the cyclic data exchange on the network.

Each of these blocks registers itself dynamically with the driver function block identified by the {Address} parameter. Up to 512 COS_Var and COS_Var_8 function blocks can be registered with a single driver. The registration is constantly checked and, therefore, the {Address} parameter can be changed at run-time to re-allocate a COS_Var_8 to a different driver or to a different memory location in the same driver. Note: this is unlike the standard PC3000 Slv_Vars which register at start-up and cannot be changed dynamically.

Each COS_Var_8 can be specified as either Input (Consumed), Output (Produced) or Off. This property is also dynamic.

At each execution of the driver function block, it scans through its list of registered COS_Vars and COS_Var_8's and

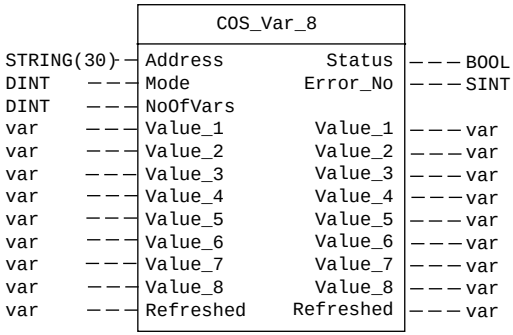
- updates those registered as Input, with new values from the Input Process Data area
- updates the Output Process Data area with new data from those registered as Output
- ignores any which are registered as Off.

The driver, therefore, becomes more heavily loaded with increasing numbers of COS_Vars and COS_Var_8's and care should be exercised when assigning it to an unnecessarily fast task.

## 17.2 Parameter Diagram

The figure describes the COS_Dint_8 and COS_Real_8 function blocks. The only difference between them is the data type of {Value} which has, therefore, been omitted from the diagram.

Figure 17-1: COS_Var_8 block diagram




## 17.3 Parameter Descriptions

The parameters for these blocks are largely identical to those for the discrete COS_Dint and COS_Real function blocks described in 16.3 and reference should be made to that section. The following notes only address any variations.

### 17.3.1 Address

The {Address} format is identical to that for the discrete blocks. The Offset and Length specified in this parameter apply to {Value_1}. {Value_2} etc. are read (written) by successively adding Length to the Offset.
e.g. {Address} = '1:4:2'. Value_1 is at offset 4, Value_2 is at offset 6 etc.
Process Data area

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Value_1 | | Value_2 | | Value_3 | | Value_4 | | Value_5 | | Value_6 | | Value_7 | | Value_8 | |

e.g. {Address} = '1:2:4'. Value_1 is at offset 2, Value_2 is at offset 6 etc.
Process Data area

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Value_1 | | | | Value_2 | | | | Value_3 | | | | Value_4 | | | | etc | |

Where a bit number is specified, the length is deduced from the bit number and then used to increment for each successive value.

### 17.3.2 NoOfVars

Wirable Input INTEGER. Max: 8, Min: 0
Can be used to limit the number of values to less than the maximum of eight. This is necessary to avoid potential memory violation errors or possible overlap between this block and other blocks. Although no checking is implemented to prevent overlap between different blocks, the specified offset and length are checked against the Produced and Consumed memory sizes specified in the driver. If these sizes are exceeded, an error will be reported.

### 17.3.3 Refreshed

Nonwirable Input BOOL
Only relevant if {Mode} is set to Input. This is set On if any of the Value_1 to Value_8 are changed by the driver. This is useful as an edge detector. The user program must turn the value Off so that the next edge can be detected.

# Glossary of Terms

## 18.1 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 LCM's and the 2nd in version 2 LCM's and LCM-PLUS's). |
| 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, Windows 2000 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. |

## 18.2 Profibus terms

| | |
|---|---|
| DP | Decentralised Periphery (Distributed control). |
| GSD | Device Data Base file, equivalent to an electronic device data sheet. Contains information about a Profibus slave that the master will use when it configures the network. 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. |
| Input data | Data in the slave that comes from device inputs, or similar parameters, and is read by the master, e.g. process value. |
| Output data | Data in the slave that is written by the master and copied to device outputs or similar parameters, e.g. controller setpoint. |
| Receive Process Data | The area of dual port memory in the master that contains copies of the Input data for all the configured slaves. During the cyclic data exchange, values are read from all slaves and copied to this area. |
| Send Process Data | The area of dual port memory in the master that contains copies of the Output data for all the configured slaves. During the cyclic data exchange, the values in this area are written to the appropriate slaves. |

| | |
|---|---|
| Demand data Exchange | A mechanism whereby data that is not configured as part of the cyclic data exchange can be written or read on command. |

## 18.3 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 COMDNS.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. |

## 18.4 Other terms and references

### 18.4.1 Reference 1

| | |
|---|---|
| Hilscher GmbH | Manufacturer of Fieldbus interface cards and modules.<br>Hilscher Gesellschaft für Systemautomation mbH<br>Rheinstraße 78<br>D-65795 Hattersheim<br>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 |

### 18.4.2 Reference 2

| | |
|---|---|
| Sycon PB/E | System Configuration software for the Hilscher DPM module. Runs on a PC under Windows 95, Windows 98, Windows NT or Windows 2000 and is used to configure the Profibus network. Supplied by Hilscher GmbH, see 18.4.1. |

### 18.4.3 Reference 3

| | |
|---|---|
| User Manual | Eurotherm document, HA027902<br>Profibus on PC3000, User Manual<br>Eurotherm document, HA027903<br>DeviceNet on PC3000, User Manual |

## 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

ED 29

**http://www.eurotherm.co.uk**

CE

ENG



HA027900