Network Functionality in CoDeSys V2.3

Document Version 1.7

CONTENT

1 Overview

Network variables can be used for an uncomplicated data exchange between two or several PLCs without the need of PLC-specific functions. Currently the functionality of network variables is implemented for CAN and UDP networks. The values of the variables are exchanged automatically, like broadcast messages. In UDP these messages are implemented as broadcast telegrams, in CAN as PDOs. These services are **not confirmed** by the protocol, which means that it is not checked, whether a message is received by the addressee. The exchange of network variables is a 1 (sender) to n (recipients) – connection.

The **Parameter Manager** is another possibility to exchange variables. It bases on a 1 to 1 – connection, using a **confirmed** protocol. The user can check whether the message has reached the recipient. The exchange is not done automatically but by function block calls which come from the user program. In the target system the handling of the parameter lists must be implemented, basing on the download format of the Parameter Manager. If this is not the case, which means that the online functions are not available, then at least alternatively the parameter lists can be exchanged via using SDO function blocks of the network libraries.

Network Variables Lists

Network variables are defined by name in the CoDeSys project and will be exchanged during run time automatically* between the PLCs. For this purpose they are combined in so-called **network variables lists**. There can be several lists per PLC. Due to the fact that the data exchange is done via broadcast messages, one of the PLC is operating as sender and the others can receive the data. **No PLC-specific functions are necessary**, but the variable lists must be absolute consistent in all PLCs. In order to support this consistency the CoDeSys programming system provides the possibility to export a variables list during the generation process and to re-import it in the receiving PLCs.

During runtime the variables are identified by a **index/subindex** system (not to be mixed up with the index/subindex system used by the Parameter Manager !). The assignment of the indices is **semi-automatic**: each of the variables lists contains a list identifier which is defined by the user in the Properties dialog (COB-ID) and which is used as index. For UDP communication the subindex will be defined automatically corresponding to the order of the variables within the list and the number of the transfer units. For CAN communication automatically further COB-IDs will be assigned if more than 1 PDO is necessary for data transfer. Each variable list can be released for up to four different communication networks.

*** Please note:** The automatic exchange can if desired explicitly deactivated by the IEC program with the help of the PDO function blocks of the network libraries.

Parameter Lists

In the Parameter Manager **parameter lists** can be created, which contain process variables known by name or process-independent parameters (constants). These parameters are entered and provided with attributes and a index/subindex (not to be mixed up with the index/subindex system used in the network variables lists!) by the user. Each list gets a name. The lists then can be loaded on the PLC and from there can be exchanged with other PLCs. The assignement of name and index/subindex for each parameter on the source PLC allows a merely index-based write and read access by other PLCs.

The assignment of **index/subindex** is determining for the consistency of data in all exchanging PLCs and therefore must be unique within the network. The value range of index and subindex is defined in the target settings.

The Network variables function as well as the Parameter Manager function must be activated in the **Target Settings**.

For network variables enter here the used protocol, which then can be parameterized individually for each variables list.

For the Parameter Manager enter the index and subindex ranges for parameter lists and for variables lists. This means that one node not necessarily must support all indices/subindices concerning a remote access.

Please regard: If the target system does not support the online functionality of the Parameter Manager (see entry "ParameterManager" in the target file) and for this reason the parameter lists cannot be exchanged by the target system, the variables nevertheless can be exchanged using the SDO function blocks of the network libraries. In order to make this possible, the option 'Support network variables" must be activated in the target settings of the sending as well as of the receiving target systems !

Note concerning language use:

The terms "PDO", "SDO" and COB-ID" here are also used for the UDP environment; "OD" stands for "Object Dictionnary" which has been a feature of CoDeSys before V2.2 and which now is replaced by the "Parameter Manager".

2 Network Variables User Interface

For using the network variables functionality with CoDeSys, the user has to perform the following steps:

2.1 Prerequisites

You need a CoDeSys Programming System Version 2.2 (or later) and the libraries

for UDP: NetVarUdp_LIB_V23.lib, SysLibCallback.lib und SysLibSocket.lib

for CAN: NetVarCan_LIB.lib, SysLibCallback.lib und CANDrvLib.lib

Besides that you need a CoDeSys Runtime System which supports for UDP the libraries *SysLibSocket.lib* and *SysLibCallback.lib*, for CAN the library *CANDrvLib.lib*. *NetVarUdp_LIB_V23.lib* resp. *3S_CANopenManager.lib* and *3S_CanOpenNetVar.lib* are internal libraries, for which no code must be implemented in the runtime system.

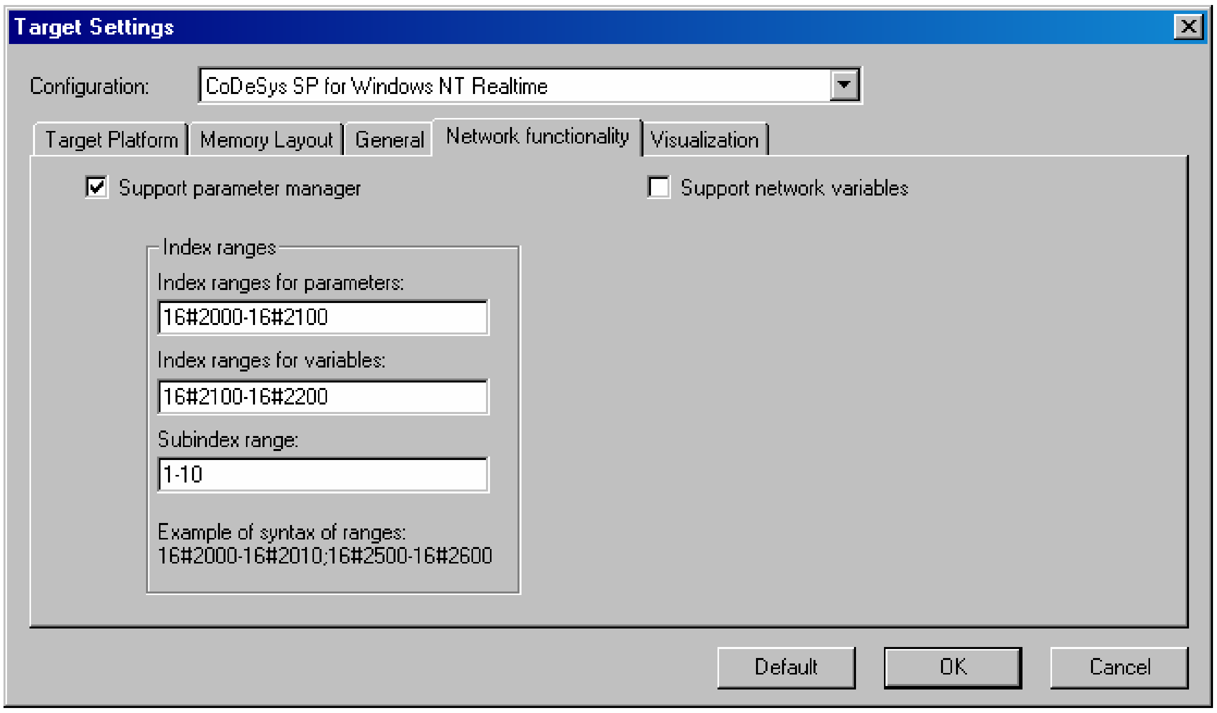
NetVarUdp_LIB_V23.lib is an extension of the library NetVarUdp_Lib.lib. It contains the structures which are needed for acknowledged transfer and diagnosis The library can be used with CoDeSys 2.3 or later versions. NetVarUdp_Lib.lib also still can be used, but does not provide the new features.

CoDeSys automatically will include the appropriate libraries if the defined network type is “UDP” and if the libraries are available in the libraries folder. (Not yet working for network type “CAN”. Also CoDeSys will automatically generate the required initialization code and the calls of the network POUs at the start and the end of each cycle.

Please regard:, The POUs and structures of the NetVar-libraries must not be called explicitly. For this reason they are not described in detail here. Just the NetVarSDO_Udp functionblock is intended to get called explicitly. How to use this module is described in chapter 3.3.

2.2 Target Settings

Activate the option **Support network variables** in the dialog box 'Target Settings' (tab 'Networkfunctionality'). At **Names of supported networkinterfaces** enter the name of the desired network, e.g. UDP. The required libraries will then automatically be included in the project.


Target Settings

Configuration: CoDeSys SP for Windows NT Realtime

[✓] Support parameter manager

[ ] Support network variables

Index ranges

Index ranges for parameters: 16#2000-16#2100

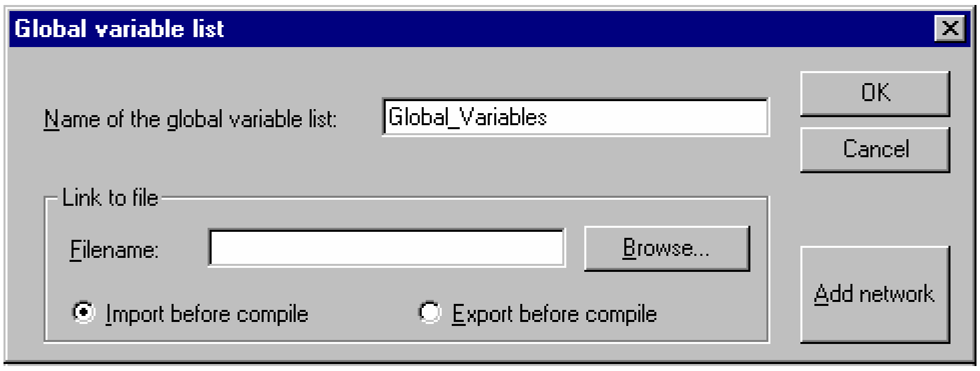
Index ranges for variables: 16#2100-16#2200

Subindex range: 1-10

Example of syntax of ranges: 16#2000-16#2010;16#2500-16#2600

2.3 Settings in the Global Variables Lists

Add a new global variables list. Here you define the variables, which you want to exchange with the other PLCs. Use the command 'Object Properties' to open the following dialog:



Global variable list

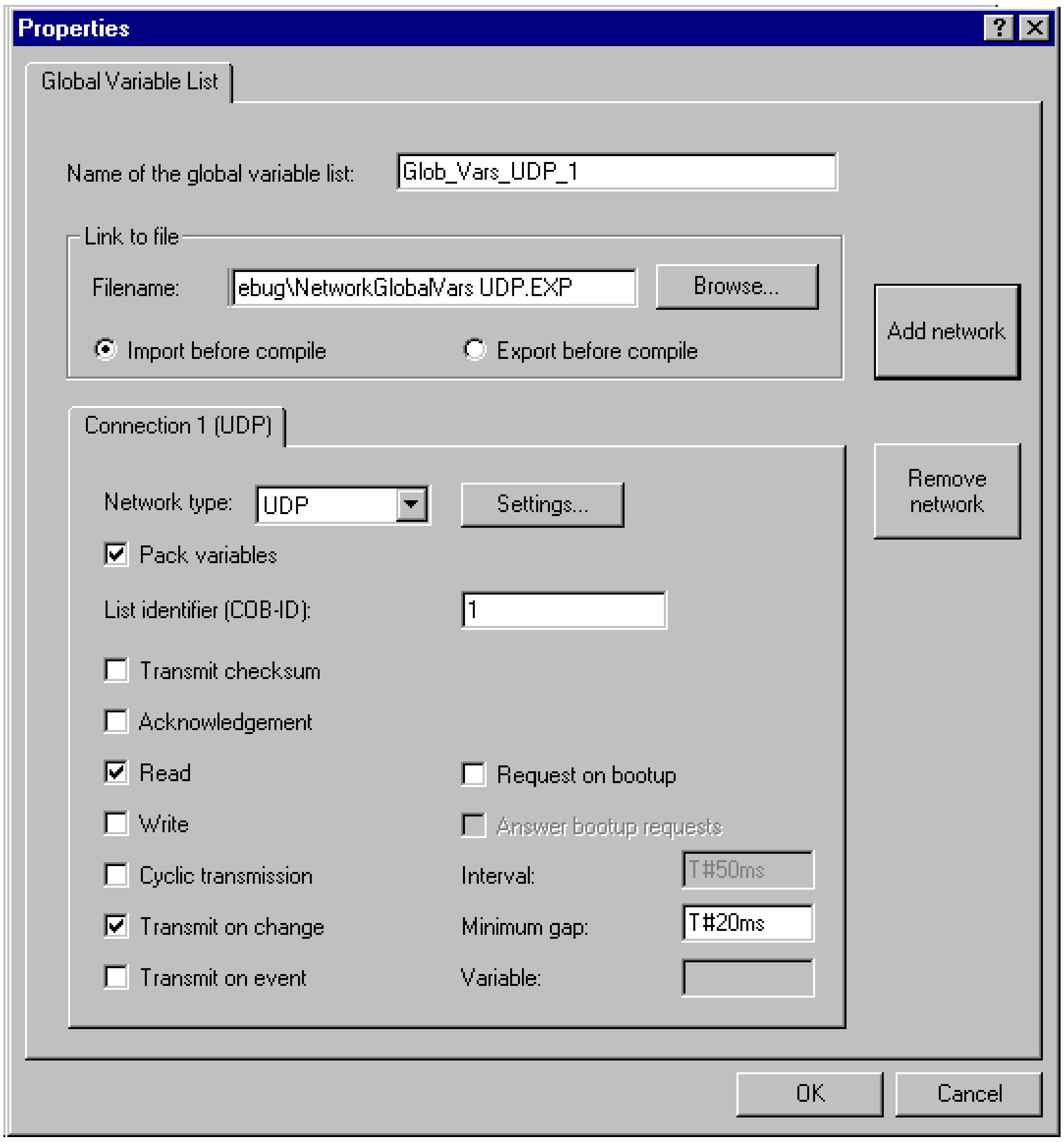
Name of the global variable list: Global_Variables

Link to file

Filename:

[✓ Import before compile] Export before compile

Click on the button **Add Network** to get the options for defining the network properties of the current variables list. If you have set up several network connections, then you can configure several connections for each Global variable list.



Properties

Global Variable List

Name of the global variable list: Glob_Vars_UDP_1

Link to file

Filename: ebug\NetworkGlobalVars UDP.EXP

[✓ Import before compile] Export before compile

Connection 1 (UDP)

Network type: UDP

☑ Pack variables

List identifier (COB-ID): 1

☐ Transmit checksum

☐ Acknowledgement

☑ Read

☐ Request on bootup

☐ Write

☐ Answer bootup requests

☐ Cyclic transmission

Interval: T#50ms

☑ Transmit on change

Minimum gap: T#20ms

☐ Transmit on event

Variable:

The options and their meaning:

Network type: one of the network types which are defined in the Target Settings dialog (tab 'Networkfunctionality').

The **Settings for UDP** provide the following configuration options:

Use standard: If this button is pressed, Port 1202 will be used for the exchange with other PLCs in the network. The broadcast/multicast address will be set to "255 . 255 . 255 . 255", which means that the exchange will be done with all nodes within the network.

Port: Port which is to be used alternatively to the standard (see above). Attention: Must be set identically at all participating nodes. This value automatically will be taken on for all further connections which might be defined on further tabs in the configuration dialog.

Broadcast/Multicast address: This address will be used alternatively to the standard (see above) for a subnetwork (e.g. "197 . 200 . 100 . 255" would affect all nodes with the IP addresses 197 . 200 . 100 . x)

In the **Settings** for **CAN** enter the **Controller Index** of the controller, which will be used for sending the network variables.

Please regard: On Win32 systems the subnet mask of the TCP/IP configuration must match with the set Broadcast/Multicast address of the network variables, in order to be able to use the network variables functionality.

Pack Variables: If this option is activated, as many variables as possible will be packed in one transmission unit. For UDP a transmission unit has a maximum size of 256 Bytes, for CAN 8 Bytes. If one transmission unit is to small to catch all variables, then further units will be generated (Regard this when defining the COB-IDs for the variables lists, see below).
If the option is deactivated, each variable will be packed in a separate transmission unit.

List identifier (COB-ID): An unique identifier for a variables list, which is useful to mark the lists which should be exchanged between several projects. All lists which have the same identifier will be exchanged. But assure that the lists, which have the same identifier, also contain identical definitions ! For this it is recommended to use the feature **Link to file** to export the desired variable list from one project and to import it into the other projects.
ATTENTION: Regard the size of the data when you define the list identifer, especially for CAN networks: One transfer unit (PDO) max. can contain 8 bytes of data. If a variable list must be splitted up to several PDOs because of its size, these PDOs automatically will get assigned additional COB-IDs, which are numbered consecutively, starting with that which is defined in the Properties dialog of the list. Thus there is the risk of generating double defined IDs.

Include Checksum: (not supported for CAN): A checksum will be added to each packet which is sent. The checksum will be checked by the receiver to make sure that the variable definitions of sender and receiver are identic. A packet with a non-matching checksum will not be accepted and – if this is configured ('Use acknowledge transfer', see below) – will be acknowledged negatively.

Use acknowledged transfer: (not supported for CAN): Each message will be acknowledged by the receiver. As soon as the sender does not get at least one acknowledgement within a cycle, an error will be produced which in case of an UPD-network will be written to the diagnosis structure of the NetVarUdp_LIB_V23.lib (see chapter 3.4).

Read: The variables in the list are read; if the option is deactivated, further variables sent over the net will be ignored.

Request at Bootup: (not supported for CAN): If the local node is a "reading" node (Option 'Read' activated), then as soon as it gets re-booted the actual variable values will be requested from all writing nodes and will be sent by those, independently of any other transmit conditions (time, event), which normally trigger the communication. Precondition: In the configuration of the writing nodes the option 'Answer Bootup requests' must be activated ! (see below).

Write: The values of the variables in this list should be sent to other PLCs. It is recommended to set only one of the options "Read" or "Write" for a variables list. If some variables of a project should be read <u>and</u> written, then use two variables lists: one for reading, one for writing. Besides that it is recommended to have only one PLC in a network which is used to send a certain variables list.

Answer Bootup requests: (not supported for CAN): If the local node is a "writing" node (Option 'Write' activated), then each request of a reading node which is sent by it at bootup (Option Request on Bootup, see above), will be answered. That means that the actual variable values will be transmitted even if none of the other defined transmission triggers (time or event) would force this at this moment.

Cyclic transmission: Variables are written within the intervals specified after **Interval**. (time notation e.g. T#70ms).

Transmit on change: Variables are written only when their values change; an entry after Minimum can, however, set a minimum time lapse between transfers.

Transmit on event: (not supported for CAN): The variables of the list will be written as soon as the variable inserted at **Variable** gets TRUE.

Besides simple data types, a variable list may also contain structures and arrays. Elements of these compound data types are sent individually

If a variable list is bigger than the PDO size of the respective network, it is sent using several PDOs. This means that it can not be assured that all data of a variable list is received in the same cycle. Portions of the variable list may arrive in different PLC cycles. This can happen also for variables of array or structure data types.

2.4 Internals

2.4.1 Format of a telegram (UDP)

An UDP network-variable-telegram has the following format (The terms "PDO" and "SDO" also in this context are used for the transfer units):

Offset	Size	Name	Content
0	4	Identity	Contains the CoDeSys protocol identity code ‚3S-0'. (Byte0 = '0'; Byte1 = '-'; Byte2 = 'S'; Byte3 = '3')
4	4	ID	Type of the message. For PDO-messages (network variables) the value is always 0, for SDO- messages (Parameter list elements) the value is of enumeration type NetVarOD_Service_Udp (see below).
8	2	Index	This Feld contains the base identifier (COB-ID), this means the number of the network variable list.
10	2	SubIndex	If the option „pack variables „ is not activated, the single network variables are sent in separate messages. Each message contains a subindex. This enables the receiving controller to find the correct variable.
12	2	Items	Contains the number of variables.
14	2	Length	This is the total size of the message (Header + Data)
16	2	Counter	The Counter counts the number of sent telegrams.
18	1	Flags	Bit0: Send-acknowledgement desired Bit1: Check of checksum desired (see below) Bit2: Invalid checksum (see below)
19	1	Checksum	Checksum of the datagram, is checked on request (see above)
20	Max. 256	Data	In the data area, CoDeSys appends the values of the network variables.

Enumeration NetVarOD_Service_Udp:

```
TYPE NetVarOD_Service_Udp :
(
      ODStateFree := 0,
      ODReadRequest,
      ODReadRequestReply,
      ODWriteRequest,
      ODWriteRequestReply,
      ODAcknowledgement,
      ODBootUpRequest,

      ODErrorReplyUnnown := 20,
      ODErrorReplyItems,
      ODErrorReplyAccess,
      ODErrorReplyIdx,
      ODErrorReplySub,
      ODErrorReplyLen,

      ODErrorWrongService,
      ODErrorTimeOut,
      ODErrorReplyCPUStopped

      ODStateBusy := 100
);
```

With SDOs, which are used for network variables created by the Parameter Manager, confirmed messages are used to read and write the values:

ODReadRequest - ODReadRequestReply, ODWriteRequest - ODWriteRequestReply.

Additional, there are code for error cases. The state of ta SDO-transmission is contained in the output variable nStatus of POU *NetVarSDO_Udp* (enumeration *NetVarOD_Service_Udp*, see above).

2.4.2 Generated Calls

Automatically an **implicit variable list** will be created in the CoDeSys project. For each PDO (sending or receiving) an array entry of type *NetVarPDO_Rx_UDP* resp. *NetVarPDO_Tx_UDP* will be inserted.

For debug purposes a file containing the generated declarations will be created. The file is named *NetworkGlobalVars_UDP.exp* and will be found in the compile files directory.

Also **initialization code** is generated, which will initialize the created data. The initialization code will be called by the GlobalInit POU, which in turn is called immediately after a download in order to initialize the project data.

At the begin and at the end of a task implicit calls of library functions will be generated:

For each task which uses network variables initially a receive call will be generated for each PDO:

```
pNetVarPDO_Rx_Udp[0]();
pNetVarPDO_Rx_Udp[1]();
...
```

For each task which uses network variables initially a (sole) instance of the POU NetVarManager_Udp_FB bzw. NetVarManager_Can_FB will be called:

```
NetVarManager_UDP();
```

For each task which uses network variables at the end a send call will be generated for each PDO:

```
pNetVarPDO_Rx_Udp[0]();
pNetVarPDO_Rx_Udp[1]();
...
```

3 Parameter Manager

The Parameter Manager (for operating see CoDeSys User Manual or Online Help) makes possible a connection between two PLCs PLC_A and PLC_B. PLC_A defines one or several parameter lists, i.e. tables containing project variables which can be accessed by an index and subindex. PLC_B can use SDO services or function blocks to read or write the variables' values from a parameter list of PLC_A.

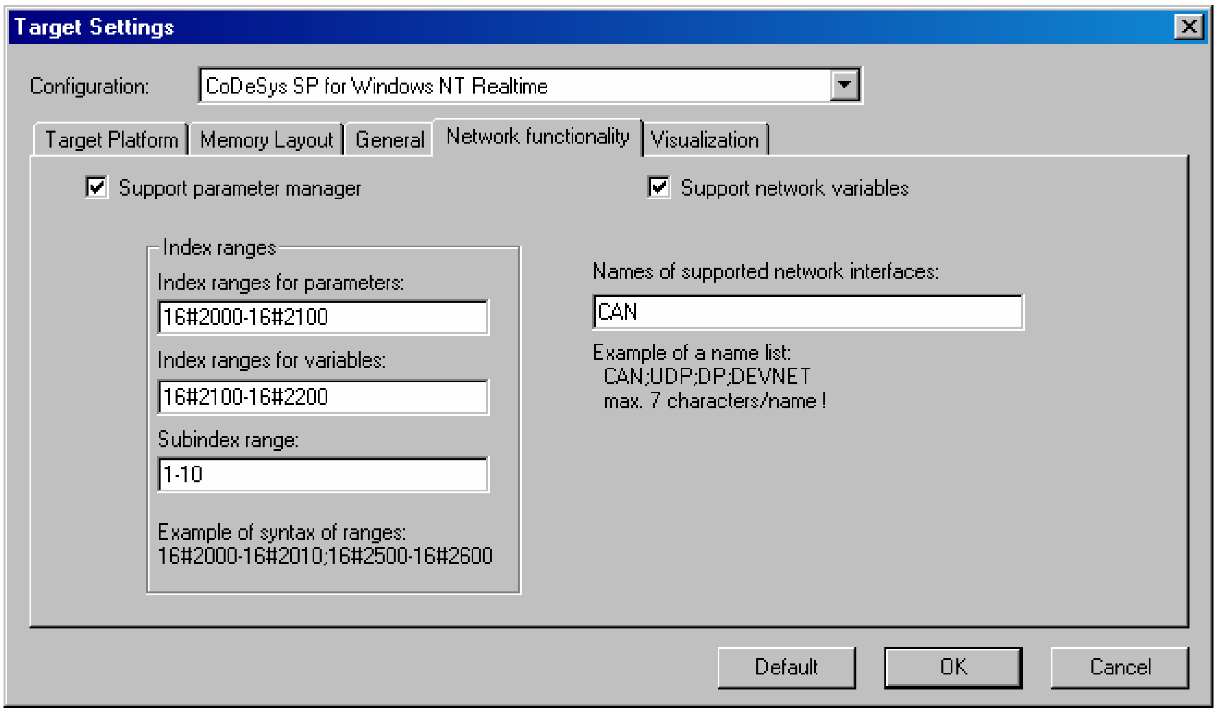
Currently this feature is implemented in the library *NetVarUdp_LIB.lib* (*NetVarSDO_Udp*).

3.1 Prerequisites

The prerequisites are the same as for the Network Variables (see chapter 2.1).

CoDeSys will automatically generate the initialization code and the calls for the Parameter Manager POUs at the start and the end of each cylce. These are necessary to recognize and answer requests from other PLCs.

3.2 Target Settings



Configuration: CoDeSys SP for Windows NT Realtime

[✓] Support parameter manager

[✓] Support network variables

Index ranges

Index ranges for parameters: 16#2000-16#2100

Index ranges for variables: 16#2100-16#2200

Subindex range: 1-10

Example of syntax of ranges:
16#2000-16#2010;16#2500-16#2600

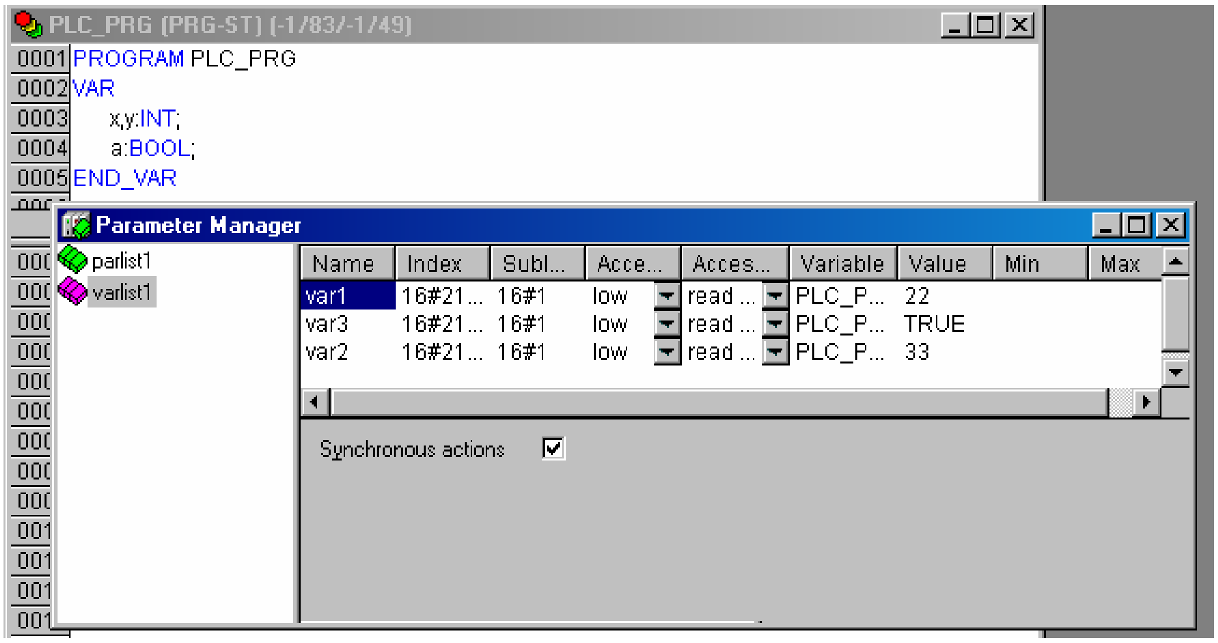
Names of supported network interfaces: CAN

Example of a name list:
CAN;UDP;DP;DEVNET
max. 7 characters/name !

Activate the option **Support parameter manager** in the dialog 'Target Settings', tab 'Network functionality'. Enter the **Index ranges for parameters** and **variables** (parameter lists of type 'Parameter' resp. 'Variables') and the **Subindex range**.

In the Resources tree you will now find a new node 'Parameter Manager'. Here you can enter project variables and process-independent parameters in parameter lists. These variables will be accessible for other PLCs in the network by using the SDO Read and Write services. (SDOs for parameter lists of type 'Parameter' are not supported by the network library.)

Please regard: If the target system does not support the online functionality of the Parameter Manager, additionally activate the option 'Support network variables'. In this case the variables, which are organized in the Parameter Manager, can be exchanged with the aid of the SDO function blocks of the network libraries.



```
PROGRAM PLC_PRG
VAR
    x,y:INT;
    a:BOOL;
END_VAR
```

Name	Index	Subl...	Acce...	Acces...	Variable	Value	Min	Max
var1	16#21...	16#1	low	read ...	PLC_P...	22		
var3	16#21...	16#1	low	read ...	PLC_P...	TRUE		
var2	16#21...	16#1	low	read ...	PLC_P...	33		

Synchronous actions ✓

3.3 Reading and Writing Parameter List Entries

In the user program the parameter list of another PLC can be accessed by using the function block (FB) **NetVarSDO_Udp**. The function block is defined in the library NetVarUdp_LIB_V23.lib as follows:

```
FUNCTION_BLOCK NetVarSDO_Udp
VAR_INPUT
bReadData          : BOOL;
bWriteData         : BOOL;
nIndex             : INT;
nSubIndex          : INT;
pData              : POINTER TO ARRAY[0..255] OF BYTE;
nLen               : INT;
nItems             : INT;
stIPAddressClient  : STRING(20);
TimeOut            : TIME := t#500ms;
END_VAR
VAR_OUTPUT
nStatus            : NetVarOD_Service_Udp;
END_VAR
```

The parameters have the following meanings:

-	bReadData:	TRUE reads data from the other PLC.
-	bWriteData:	TRUE writes data to the other PLC.
-	nIndex:	Index in the parameter list of the other PLC.
-	nSubIndex:	SubIndex in the parameter list of the other PLC.
-	pData:	Pointer to a buffer which contains the data to be sent resp. to be received, e.g. ADR(a).
-	nLen:	Lenght of the buffer containing the data to be sent resp. to be received, e.g. LEN(a).
-	nItems:	Number of elements, if it is an ARRAY, otherwise 1.
-	stIPAddressClient:	IP-Address of the other PLC, coded as an DWORD.
-	TimeOut:	Wait time, after which an answer from the other PLC is expected.

Return: nStatus contains the status of the request:

```
TYPE NetVarOD_Service_Udp :
(
ODStateFree := 0,
ODReadRequest,
ODReadRequestReply,
ODWriteRequest,
ODWriteRequestReply,
      ODAcknowledgement,
      ODBootUpRequest,

ODErrorReplyUnnown := 20,
ODErrorReplyItems,
ODErrorReplyAccess,
ODErrorReplyIdx,
ODErrorReplySub,
ODErrorReplyLen,

ODErrorWrongService,
ODErrorTimeOut,
      ODErrorReplyCPUStopped,

ODStateBusy := 100
);
END_TYPE
```

The values of nstatus have the following meaning:

(Regard that the FB is used internally for answering requests which have been created via calls of the same FB on another controller. The user will makes use of the FB in the application. Only some of the status information is of interest for the user resp. visible at all.)

- ODStateFree: Displayed as soon as the FB has been called with bReadData=FALSE and bWriteData=FALSE.
- ODReadRequest: Set by the FB if the user has called the FB with bReadData=TRUE. (Rising edge)
- ODReadRequestReply: Set by the FB after having received an answer on a ReadRequest.
- ODWriteRequest: Set as soon as the FB has been called with bWriteData=TRUE. (Rising edge)
- ODWriteRequestReply: Set as soon as the FB has got an answer on a write request.
- ODAcknowledgement: Never used by the FB. Will be added to a telegram if a confirmation has been configured for a Receive PDO.
- ODBootUpRequest: Never set by the FB. Will be added to a telegram only if a bootup request has been configured for a receive telegram.
- ODErrorReplyUnknown: not used, historic.
- ODErrorReplyItems: Set by the FB as soon as an error of this type has been received as an answer on a write or read request. The error will be created by the communication partner if the number of items (array or structure elements not identic on both sides) does not match the request.
- ODErrorReplyAccess: Set by the communication partner if a request cannot be executed, because there is no write or read acces on the entry.
- ODErrorReplyIdx: Set by a communication partner if the request references an index which does not exist at the communication partner.
- ODErrorReplySub: Not used, reserved.

- ODErrorLen: Set by the communication partner if the data length does not match the projected data length.
- ODErrorWrongService: not used, reserved for future versions.
- ODErrorTimeOut: Set by the FB, if no answer matching the request has been received within the timeout which was passed when calling the FB.
- ODErrorReplyCPUStopped: Set by a communication partner if the CPU stopped.
- ODStateBusy: not used, reserved.

3.4 Diagnosis for UDP Networks

The structure NetVarUDPDiagStruct which is part of the library NetVarUDP_V23.lib can be used to get diagnostic information for the exchange of network variables in UDP networks.

The structure components:

Component	**Data type**	**Description**
nSendCount	UDINT	Number of already sent telegrams
tLastSend	TIME	Time of last sending
nReceiveCount	UDINT	Number of already received telegrams
tLastReceive	TIME	Time of last receiving
nWriteCount	UDINT	Number of telegrams received from others
sLastError	NetVarUDPError	Error state, see below
tLastError	TIME	Time of last error
nErrorCount	UINT	Number of already occurred errors
nAcknowledges	UINT	Number of sent acknowledgements

Components of **Enumeration NetVarUDPError**:

NetVarUDPError_NOERROR := 0,	No error
NetVarUDPError_UDPSENDDATA,	Error at sending of a telegram
NetVarUDPError_UDPSENDACKN,	Error at sending of the acknowledgement
NetVarUDPError_NOACKNOWLEDGEMENT,	No acknowlegement received, although option 'Acknowlegement' is activated in the configuration dialog for the network variables
NetVarUDPError_NOTACKNOWLEDGED,	Acknowledgement received, but it is negative, i.e. the sent telegram was not ok
NetVarUDPError_UDPSENDBOOTUPREQUEST,	Error at sending of the Bootup Request (Request for actual data at bootup, Option 'Request at Bootup')
NetVarUDPError_UDPANSWERBOOTUPREQUEST,	Error at answering the Bootup Request (Request for actual data at bootup, Option 'Request at Bootup')
NetVarUDPError_CHECKSUM,	Checksums are different

NetVarUDPError_UDPNETWORKMANAGER_NOT_READY := 20,	Network manager is not initialized correctly -> cannot execute send order (e.g. because sockets cannot be opened)
NetVarUDPError_UNKNOWN := 100	currently not used

Change History

Version	Description	Editor	Date
1.0	Release		08.04.2002
1.1	Chapter 2.4 ('Format of a telegram') added		
1.2	Chapter 2.4: new title, 2.4.2 ("Generated calls") added		
1.3	Rework, especially resp. Parameter Manager; new options in Network variables dialog		
1.4	NetVarUPD_Lib_V23.lib		
1.5	Chap. 3.3: Description for nstatus values added	AF	17.06.2005
1.5	Release + Adaptation to new template	MN	17.06.2005
1.6	Chap.1.and 2.3: Attention-note removed; „Please regard“-notes added in Chap.1 below ‚Networkvariableslists’ and in Chap. 2.1 (bActive #4964, online change possible since 2.3.4.0);	MN	25.07.2005
1.6	Review: Chap. 3.3 nstatus: ODErrorReplyCPUStopped and ODAcknowledgement are missing	SM	02.08.2005
1.6	Review: corrections of Review points, see above; Release	MN	03.08.2005
1.7	Chap. 2.3, Correction at „Use acknowledged transfer” (#5626), Release with CoDeSys V2.3.6.0	MN	07.12.2005