Connecting UniOP to Interbus

This Technical Note contains all the information required to connect the UniOP operator panels to an Interbus system and to take advantage from the advanced data access options supported by the UniOP Interbus communication driver.

***Important**: The Interbus connection of the UniOP operator panels is compliant with the MMI-COM standard device profile. A special application program running in the Master Controller is required to support the communication. This application program has been tested and is available for some specific system configurations (the list of the support programs currently available is in the Appendix. This Technical Note applies only to the Interbus communication driver identified by the name 'Interbus' associated to the Designer file UPLC120.DLL. This communication driver requires a panel of hardware type –0045 equipped with a communication module type TCM04.*

Contents

1. Introduction

The UniOP operator panels can be connected to an Interbus system on the Remote bus. This Technical Note describes the principal points to follow for a successful connection.
UniOP connects to the Interbus system in accordance with the MMI-COM device profile.

UniOP is always a Slave in an Interbus network and it is only able to exchange data with a single master controller.

UniOP is compliant with the MMI-COM standard device profile specification and operates through the Indirect Process Data Channel. No Parameter Channel and no Direct Process Data Channel are used. The user data is interpreted in accordance with the MMI-COM profile specification.
The UniOP Interbus interface and the MMI-COM protocol are independent of the controller.
In the MMI-COM specification data is referenced with ‘variable numbers’. A function block must be running in the Interbus master controller to map between the variable numbers and the real PLC addresses.
According to the MMI-COM profile, the operator panel, which is acting as a client, initiates data transfers; the function block running on the Interbus master controller is acting as a server.
This technical note includes the description of all PLC-specific function blocks currently available.

Connection to the Interbus network requires an optional communication module, part number TCM04 including a special communication adapter. The Remote Bus connection features two screw-clamp terminals with standard signal assignment for incoming and outgoing bus interfaces.

There are 3 main steps that you need to follow to have UniOP working in an Interbus system:

1) Add the special UniOP function blocks to the program in the Interbus master controller
2) Create the UniOP Communication Data Block in the master controller and enter in this data block information on the network configuration
3) Configure the UniOP Operator Panel with the Designer software package

All these steps will be described in greater detail in the following sections.

2. Interbus overview

Interbus is a serial bus system for transmitting data between different types of control systems and spatially distributed I/O units.
Interbus performs two important tasks:

- The cyclical transmission of process states that change quickly.
- The transmission of parameter data for complex I/O modules and specialised process devices.

The IBS system is designed as a data ring. The controller board is the central device for controlling the data ring. It exchanges data transmitted serially within the data ring with the high-level control or computer system and the low-level IBS devices. The data exchange is carried out simultaneously and cyclically in both directions (full duplex).
Each station of an IBS ring represent a repeater and to simplify system installation, the go and return lines of the ring system are implemented within one cable line.
Each device in the IBS system has an identification register (ID register). This register includes specific information about the device. In addition each device has I/O registers for the transmission of the process data.

2.1 Device Profile

The **device profile** defines the application functions that are visible through the communication.
In relation to the specific application or hardware group, the **Communication Profile** for a device limits or classifies the degrees of freedom contained in the specification of the data transfer medium.
According to the MMI-COM profile the IBS system has basically three **Data Channels** described into the **Communication Functions**.
The communication functions define how the data transport via **Process Data Channels** and the **Parameter Channel** is carried out.

The communication functions provide three types of data transmission:

- **Direct Process Data**: cyclic data which remain constant through the service period of the device; they are transmitted via the process data channel at cyclic interval and they are not acknowledge.

- **Indirect Process Data Channel:** contains data which do not remain constant through the service period of the device, but they change according to the external events or requests. The structure of these data is determined by means of the process data identifier (**PD-index**); the handshaking between two communication partners is defined using a status/control byte.

- **Parameter Channel**: permits a background communication activity during which the bytes of an error-protected protocol are transmitted sequentially. This procedure is usually used for low communication sped involving large amount of data and does not slow down the transmission of the time-critical data.

2.2 Interbus Connection Data in MMI-COM profile

The UniOP Operator Panel operates in the Interbus system as slave device on the 2-wire Remote Bus.
The panel occupies 8 bytes of Interbus data, mapped in the controller to 8 bytes in the Input area and 8 bytes in the Output area.

UniOP operates using the Indirect Process Data Channel. No Parameter Channel and no Direct Process Data Channel are used.
The user data is interpreted in accordance with the MMI-COM device profile.
The principle of operation is based on the Client-Server concept. UniOP acts as a client, the Interbus master controller as the server.
The operator panel initiates all service requests. The master controller will respond to the service requests.

The Interbus connection information is specified in Table 1.

IB Device Type	Remote bus
Data Channels Size	
Indirect process data with status word	8 bytes
Direct process data	None
Parameter channel	None
Identcode	0x2F [hex] / 47 [dec]
Function Group Specification According to the MMI-COM Profile	
B3	Variable Input
G1	Variable Request
Supported MMI-COM Services	
PD-Index 0x14	Variable transmit 1 byte
PD-Index 0x15	Variable transmit 2 bytes
PD-Index 0x16	Variable transmit 4 bytes
PD-Index 0x40	Variable request 1 byte
PD-Index 0x41	Variable request 2 bytes
PD-Index 0x42	Variable request 4 bytes

Table 1 – IBS Connection Data

2.3 Assignment of the Process data Channel

The area for the process data channel is composed of the **control/status word** with an integrated **PD-index** and a **data field**. The current structure of the data field is defined by the PD-index in the control/status word. The length of the indirect PD channel is fixed and is specified in the device description as reported in the previous chapter.
The MMI-COM profile defines an 8-bytes sized area within the indirect process data channel with the following structure:

Byte 1	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**	**Byte 8**
Status byte	*PD-index*	*Variable Number High*	*Variable Number Low*	*Data*	*Data*	*Data*	*Data*

Since the data transfer must be carried out in both directions, each UniOP connected to IBS is mapped to the controller memory with 8 bytes of input data and 8 bytes of output data.

Byte 1 and Byte 2 are together the control/status word that is composed as shown in Table 2.

Byte	Bit	Name
2	0	PD-index bit 0
	1	PD-index bit 1
	2	PD-index bit 2
	3	PD-index bit 3
	4	PD-index bit 4
	5	PD-index bit 5
	6	PD-index bit 6
	7	PD-index bit 7
1	0	Reserved
	1	Reserved
	2	Index
	3	Standard
	4	Handshake sending
	5	Handshake receiving
	6	Malfunction
	7	Online

Table 2 – Control/Status word

In the following an explanation of the bits into the status byte and refers equally to the two communication directions is included. The term "Device" means, according to the direction, either the IBS master or the MMI device.

- **Online (Bit 7)**
  This bit indicates that the device is ready to receive. No telegrams may be sent to a device that does not report online. Independent of that, such device may send data itself. Online and malfunction bit do not rule out each other.

- **Malfunction (Bit 6)**
  This bit indicates a malfunction on the device. If the device is online, a malfunction code can be inquired.

- **Handshake receiving (Bit 5)**
  This bit is used to acknowledge the reception of a telegram. This bit is inverted whenever the remote station indicates a new telegram by inverting its handshake receiving bit (Bit 4) and the receiving station has read out this new telegram. This handshake signal only indicates on the protocol level that that the transmission channel involved is free again, not that the application has already processed the telegram (the receive buffer). Therefore, the application may still classify the telegram as faulty at a later moment.

- **Handshake sending (Bit 4)**
  This bit signals a new telegram. This bit may be inverted only if it matches the handshake receiving bit (Bit 5) of the remote station and the remote station is online (Bit 7). The transmission channel then remains enabled until the remote station causes its handshake receiving bit to match the new value of its own handshake sending bit.

- **Standard bit (Bit 3)**
  This bit defines the assignment of the data field for process data transfer. In this case the meaning of the PD-index is standardised. If this bit is not set, the meaning of the subsequent field is defined in a manufacture-specific way. If, while the standard bit is set, the index bit is not set, the subsequent field is a bit by bit assignment to a key or indicator field.
  In UniOP support for Interbus, this bit is always set.

- **Index bit (Bit 2)**
  This bit defines the existence of an index field in byte 2 of the indirect process data channel's data field. If the Standard bit is set, the PD-index is transferred in the index field byte 2. If the standard bit is not set, this indicates a manufacturer-specific index.
  In UniOP support for Interbus this bit is always set.

The **PD-index** specifies the MMI-COM service being used.

The **Variable Number** specifies the number of the variables to be accessed. The interpretation of the variable numbers is not defined by the device profile; the settings in the operator panel application and the PLC application must be consistent.

2.4 The PLC Support Program

To handle the services of the MMI-Com profile a support PLC program is required. A specific function block must be executed in order to handle properly the data exchange between the master and all the slave devices. The function block will interpret the data received, will manage the handshake bit and will decode the request come from UniOP.

If valid requests from the MMI are detected, the variable number transmitted must be used to get the content of a real memory location into PLC or to write to a real address in the controller memory area. Only the controller manages the data required by UniOP in an event-controlled way.
When initialising the controller, the status/control byte of the PLC must be assigned to the value 0x8C (online, standard and index bit set).

Whenever the terminal resets the online bit, the handshake bits in the controller must be erased. This permits the terminal to synchronise itself with the controller at any time.

If the bus is properly working the **online bit** must be set. The online bit indicates that the device is ready to receive. The bit is normally set.

The **malfunction bit** indicates that a malfunction has occurred. Normally the bit is reset. If the PLC program sets the bit, UniOP will interpret it as accessing to a not existing variable.

The **handshake receive bit** is used to acknowledge the receipt of a message. This bit is inverted after the remote station signals a new message by inverting its handshake transmit bit (Bit 4) and this message has been evaluated by the station to which it has been transmitted. The bit indicates, at protocol level, that the channel is available again, but it does not indicate that the corresponding application has already evaluated the message.

The **handshake transmit bit** is used to signal a new message. Inverting of this bit is allowed only when it is consistent with the handshake receive bit of the remote station and when the remote station is online. Subsequently the transmission channel will be blocked until the remote station has adapted its handshake receive bit to the new value of its own handshake transmit bit such that they are consistent again.

UniOP itself starts all services related to the terminal.
If the handshake receive bit of the terminal (IN area) is not consistent with the handshake receive bit of the controller (OUT area), then a new request has been received from UniOP which must be processed by the PLC support program.

The receipt of this new request must be acknowledge by inverting the handshake receive bit (OUT area) in the PLC and by providing back to the terminal the requested data.
The new data to be transmitted must be inserted in the transmitted data area (OUT area) of the controller and the handshake transmit bit must be inverted as well (OUT area).

3. Siemens Simatic S5 95U with Phoenix Contact IBS S5 100 CB-T

This chapter describes the application of the UniOP Operator Panels in an Interbus system based on a Simatic S5 95U/100U PLC equipped with a Phoenix Contact **IBS S5 100 CB-T** Interbus controller module.
The information necessary for the correct installation of the controller board is available in the technical documentation provided by Phoenix Contact.

The current implementation of Interbus for UniOP allows the operator panel to access a contiguous array of registers in the PLC. This array can be assigned an address either in the Merker area or in a Data Block. In accordance with the MMI-COM profile the system will map the variable objects in this PLC area. The programmer will enter addresses in the Designer software as variable indexes.

Variable Index	**Merker Addressing**
V0	MB0
V1	MB1
V2	MB2
…	…

Example 1: Variable data allocated in the Merker Area

Variable Index	**DB Addressing**
V0	DB10, DL0
V1	DB10, DR0
V2	DB10, DL1
…	…

Example 2: Variable data allocated in Data Block 10

The MMI-COM server functionality requires 3 special function blocks (FB) to be included in the PLC program. They must be called cyclically in OB1.
The Interbus data must be entered in a data block (the UniOP Comm DB). This data block specifies also which PLC memory area has been assigned to the communication between the operator panel and the PLC.

The Interbus master controller occupies a part of the address space in the Analog Process Image of the S5 PLC. The Phoenix Contact technical documentation specifies how to determine the addressing of the Interbus master board in the PLC address space.
The Interbus data can be accessed directly by the PLC user program using direct access to the I/O area (instructions L PW, T PW) or indirectly via the process image.
The IBS S5 master board is plug-and-play; no special configuration in the PLC is necessary.

Operation of the Interbus bus in a Simatic S5 95U/100U PLC system must be started by the PLC program when all the slaves are online. Note that the start-up phase of a UniOP panel may take several seconds to be completed and UniOP will not be online until started.
The sample program provides you with a simple example of network activation.

The Interbus master board allocates some system registers in the input and output area at the beginning of the I/O area assigned to the controller board (Diagnostic register, Info register and Activation register). These system registers can be used to diagnose the status of the network and to start/stop the activity of the Interbus bus.

Note: *The special Function Blocks provided with the UniOP communication driver do* ***not*** *control these system registers.*

3.1 Starting Interbus

Interbus will start when the power supply of the PLC is turned on. The Master during PLC power reads the configuration of the IBS system in a location-oriented way. The PLC recognises the address space required by Interbus and then this space is considered in the I/O address area of the entire system.
Immediately afterwards the RUN LED should light permanently. This always indicated that data is transmitted between the PLC and the IBS devices.
It is possible to start Interbus by means of the PLC program or by switching the power supply of the PLC off and on again.
The supplied S5 example does not provide any control of the IBS registers and IBS will not be started automatically if, for instance, a panel has a power fault. Please note that panel's Configuration Mode does not cause any problem to IBS chain.
Phoenix Contact provides its Master by a simple set of useful function block that could be used by the final user to set-up properly the IBS errors handling system.

3.2 Using the UniOP Function Blocks in the Master PLC

Three special function blocks are available to support the MMI-COM server capability for the Simatic S5 95U/100U PLC’s.
A data block is used to pass parameters to the function blocks.

Function Blocks are available as ready-to-run sample projects. Function and Data Blocks may be extracted from the sample projects for integration into the user’s program.
The main function block is FB97. It must be cyclically called from OB1.
It is important that FB97 is called cyclically, you should not call it only one time, as the function block only processes the requests from the UniOPs when it is called.
UniOP will not be able to communicate with the master PLC if it is in STOP mode, as the special function blocks will not be called.

Note that if you have multiple UniOPs connected to the Master PLC you do NOT need to call FB97 once for each UniOP. A single call to FB97 for every cycle of the PLC program is sufficient to process data for all the UniOPs attached to the Master PLC.

The program fragment below shows how to call FB97 from OB1.

```
      :L    KF+11      11 = UniOP Comm DB number
      :SPA FB97
NAME  :UNIBATCH
```

The call to FB97 is quite straightforward, first the Data Block number used for the UniOP Comm DB is loaded into Accumulator 1 and then the function block is called. You must always specify the UniOP Comm DB in this way before calling FB97. The UniOP Comm DB is described in the next section.
The function blocks FB96 and FB98 should not be called directly, they are used internally by FB97.

The special function blocks use the merker area MB200 - MB255 as a temporary storage area during execution. These merkers will therefore be modified when the special function blocks are called. You can access this merker area with UniOP but any access should be carefully avoided.

The PLC support program for UniOP gives the possibility to read a subset of data types in the PLC memory. In particular by configuring properly the Communication DB it is possible to select if UniOP will have access to the Merker area or to a predefined Data Block. You can choose the allocation of the communication areas with a parameter in the Communication DB.
Note that if more than one UniOP panel is connected to the master system, each UniOP can access different memory types, since the Memory Type information is included in the node-dependent part of the Communication DB.
The PLC program consists of 3 function blocks: FB96, FB97, FB98 and one data block. Only FB97 needs to be called at each PLC cycle in OB1.

3.3 Sample PLC Program

A sample PLC support program for Interbus MMI-COM communication is available. The program is provided as STEP5 project file and is based on an Interbus network including two panels: the first one will access the merker area, the second one will access the data block DB25.
The name of the sample program is **IB0950ST.S5D**.

3.4 Creating the UniOP Comm Data Block

The UniOP Communication Data Block (UniOP Comm DB) provides FB97 with all the information it needs about the Interbus addressing mode in the master board, the number of UniOP panels to process and the addresses of the Input and Output data for all the panels in the Master PLC memory.
You have to create a data block in the Master PLC that is at least 200 words long for the UniOP Comm DB.
The UniOP Comm DB has 2 distinct parts; the first part (header) contains information on the configuration of the Interbus controller board installed on the PLC while the second part contains a section for each UniOP panel connected to the network.

The header part is placed in the first 4 words of the UniOP Comm DB and has the following format:

	DL	**DR**
DW0	Number of Panels	0
DW1	0	0
DW2	0	0
DW3	0	Reserved for Internal Use

Number of Panels specifies the total number of UniOP panels that you want to let communicate with the Master PLC.

Following the header there is a section containing specific information for each operator panel.
Each UniOP panel is assigned 4 words in the UniOP Comm DB. The index "n" runs from 0 up to Number of Panels - 1.

The format of the data for a single panel has the following format:

	DL	DR
DW(4 + n)	Input Address	Output Address
DW(5 + n)	Memory Type	0
DW(6 + n)	Data Block Number	0
DW(7 + n)	Error Code For Last Request	Status Byte For Last Request

Input Address specifies the starting address in the Master PLCs memory where the Input data for this panel will be placed. This should be set according to the information available from Phoenix Contact documentation on the addressing of IBS controller board.

Output Address specifies the starting address in the Master PLCs memory where the Output data for this panel will be placed. This should be set according to the information available from Phoenix Contact documentation on the addressing of IBS controller board.

Note: *Input and Output addresses are not the start addresses of the areas reserved to the IBS controller board in the analog process image. In fact, in the word oriented address area of the PLC, two input words and one output word are assigned to the diagnostic and function registers of the IB controller board. For instance, if the controller board is placed into slot number 0, the IB base address is 64 for input and output area while the address you should insert in the Comm DB are 68 for input and 66 for output.*

Memory Type specifies the memory area that UniOP will access. Enter 0 for Data Block area or enter 2 if you want to access to merker area.

Data Block Number specifies the DB number that UniOP will access in case Memory Type = 0 has been selected.

Error Code for Last Request specifies the Error Code for the last communication request processed for this panel. You do not need to set this element. FB97 will actually write to this element to give you an indication of the error status of the communication. This field is therefore just for information. The error codes have the following meaning:

Error Code	Meaning
0	No errors
1	Panel is Offline
2	Attempting to read from a non existing variable location
3	Attempting to write to a non existing variable location

Status Byte for Last Request specifies the Status Byte for the last communication request for this panel. Every time the UniOP makes a request it includes the Status byte in the request to the PLC. This Status Byte contains handshaking information and you do not need to set this element. This field is therefore just for information.

3.5 Sample Application

In this section you will find the description of the Comm DB for a sample application including 2 UniOP panels attached to an IBS controller board placed in slot number 0 of an S5 95U. The Input address for the first panel is therefore set to 68 and the Output address to 66 in the analog process image. The second panel is mapped to 76 for the inputs and to 74 for the outputs. The first panel will access the merker area and the second one will access data block DB25.
The UniOP Comm DB would have the following form:

	DL	**DR**
DW0	2 (Number Of Panels)	0
DW1	0	0
DW2	0	0
DW3	0	0 (Reserved For Internal Use)
DW4	68 (Input Address)	66 (Output Address)
DW5	2 (Memory Type)	0
DW6	0 (Data Block Number)	0
DW7	0 (Error Code For Last Request)	0 (Copy of last status byte)
DW8	76 (Input Address)	74 (Output Address)
DW9	0 (Memory Type)	0
DW10	25 (Data block number)	0
DW11	0 (Error Code For Last Request)	0 (Copy of last status byte)

***Note**: Each panel included in the Interbus network must have its descriptor in the Comm DB. All descriptors have to be placed in consecutive memory locations.*

4. Siemens Simatic S5 115U CPU 943B with Phoenix Contact IBS S5 DSC/I-T

This chapter describes the application of the UniOP Operator Panels in an Interbus system based on a Simatic S5 115U CPU 943B PLC equipped with a Phoenix Contact **IBS S5 DSC/I-T** Interbus controller module.
The information necessary for the correct installation of the controller board is available in the technical documentation from Phoenix Contact.

The current implementation of Interbus for UniOP allows the operator panel to access a contiguous array of registers in the PLC. This array can be positioned either in the Merker area or in a Data Block. In accordance with the MMI-COM profile the system will map the variable objects in this PLC area. The programmer will enter addresses in the Designer software as variable indexes.

Variable Index	Merker Addressing
V0	MB0
V1	MB1
V2	MB2
…	…

Example 1: Variable data allocated in the Merker Area

Variable Index	DB Addressing
V0	DB10, DL0
V1	DB10, DR0
V2	DB10, DL1
…	…

Example 2: Variable data allocated in Data Block 10

The MMI-COM server functionality requires 3 special function blocks to be included in the PLC program. They must be called cyclically in OB1.
The Interbus data must be entered in a data block (the UniOP Comm DB). This data block specifies also which PLC memory area is assigned for the communication between the operator panel and the PLC.

The Interbus data can be accessed directly by the PLC user program using direct access to the I/O area (instructions L PW, T PW)
The IBS S5 master board is plug-and-play; no special configuration in the PLC is necessary.

The Interbus master board allocates some system registers in the input and output area at the beginning of the I/O area assigned to the controller board. These system registers can be used to diagnose the status of the network and to start/stop the activity of the Interbus network.

Note: *The special Function Blocks provided with the UniOP communication driver do* ***not*** *control these system registers.*

4.1 Starting Interbus

Interbus will start when the power supply of the PLC is turned on.
It is possible to start Interbus by means of the PLC program or by switching the power supply of the PLC off and on again.

The supplied S5 example does not provide any control of the IBS registers and IBS will not be started automatically if, for instance, a panel has a power fault. Please note that panel's Configuration Mode does not cause any problem to IBS chain.
Phoenix Contact provides a simple set of useful function blocks that could be used by the final user to properly set-up the IBS errors handling system.

4.2 Using the UniOP Function Blocks in the Master PLC

Three special function blocks are available to support the MMI-COM server capability for the Simatic S5 115U PLC’s.
Please refer to the corresponding section on S5 95U for details.

4.3 Sample PLC Program

A sample PLC support program for Interbus MMI-COM communication is available. The program is provided as a STEP5 project file and is based on an Interbus network including three operator panels: the first one will access the data block 35, the second one will access the data block DB25, the last one will access the merker area.
The name of the sample program is **IB1150ST.S5D**.

4.4 Creating the UniOP Comm Data Block

Please refer to the corresponding section in the chapter on S5 95U.

4.5 Sample Application

In this section you will find the description of the Comm DB for a sample application including 3 UniOP panels attached to an IBS controller board placed in slot number 0 of an S5 115U. The Input address for the first panel is therefore set to 60 and the Output address to 60. The second panel is mapped to 68 for the inputs and for the outputs. The 3rd panel is mapped to address 76 for Input and Output. The first panel will access the data block 35, the second one will access data block DB25, and the last one will access to the merker area.
The UniOP Comm DB would have the following form:

	DL	**DR**
DW0	2 (Number Of Panels)	0
DW1	0	0
DW2	0	0
DW3	0	0 (Reserved For Internal Use)
DW4	60 (Input Address)	60 (Output Address)
DW5	0 (Memory Type)	0
DW6	35 (Data Block Number)	0
DW7	0 (Error Code For Last Request)	0 (Copy of last status byte)
DW8	68 (Input Address)	68 (Output Address)
DW9	0 (Memory Type)	0
DW10	25 (Data block number)	0
DW11	0 (Error Code For Last Request)	0 (Copy of last status byte)
DW12	76 (Input Address)	76 (Output Address)
DW13	2 (Memory Type)	0
DW14	0	0
DW15	0 (Error Code For Last Request)	0 (Copy of last status byte)

Note: *Each panel included in the Interbus network must have its descriptor in the Comm DB. All descriptors have to be placed in consecutive memory locations.*

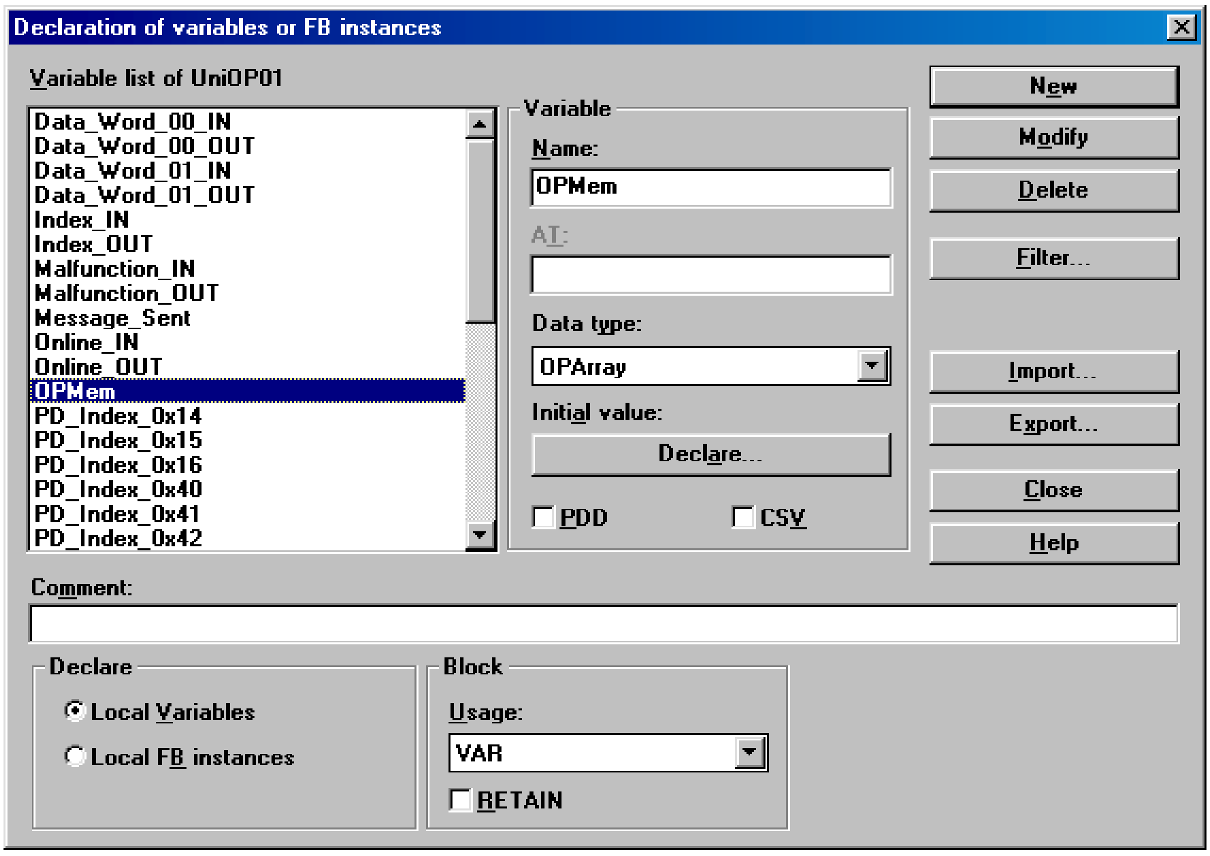
5. Phoenix Contact PCWORX with IBS ISA FC/I-T

This chapter describes the application of the UniOP Operator Panels in an Interbus system based on the Phoenix Contact Interbus Master Card **IBS ISA FC/I-T**.
The information necessary for the correct installation of the controller board is available in the technical documentation from Phoenix Contact.

The IBS card uses with the Phoenix Contact PC WORX automation software operates as an IEC1131 softlogic controller and Interbus master.
The MMI-COM server functionality requires one logical POU to be inserted in the Tasks list of the user project.
The working example provided for this controller exchanges data with the MMI using a reserved area defined in the controller memory as a new Data Type under MMI Data Types. This user memory area has been defined as an array of 1000 words; the name of the array is **OPMem** of type **OPArray**; the declaration is shown in the code below:

```
TYPE
   OPArray: ARRAY[0..999]OF WORD;
END_TYPE
```

One local variable of type **OPArray** is then declared for each panel connected as an Interbus slave as shown in Figure 1.



Variable list of UniOP01
- Data_Word_00_IN
- Data_Word_00_OUT
- Data_Word_01_IN
- Data_Word_01_OUT
- Index_IN
- Index_OUT
- Malfunction_IN
- Malfunction_OUT
- Message_Sent
- Online_IN
- Online_OUT
- OPMem
- PD_Index_0x14
- PD_Index_0x15
- PD_Index_0x16
- PD_Index_0x40
- PD_Index_0x41
- PD_Index_0x42

Variable
Name: OPMem
AT:
Data type: OPArray
Initial value:
☐ PDD ☐ CSV

Comment:

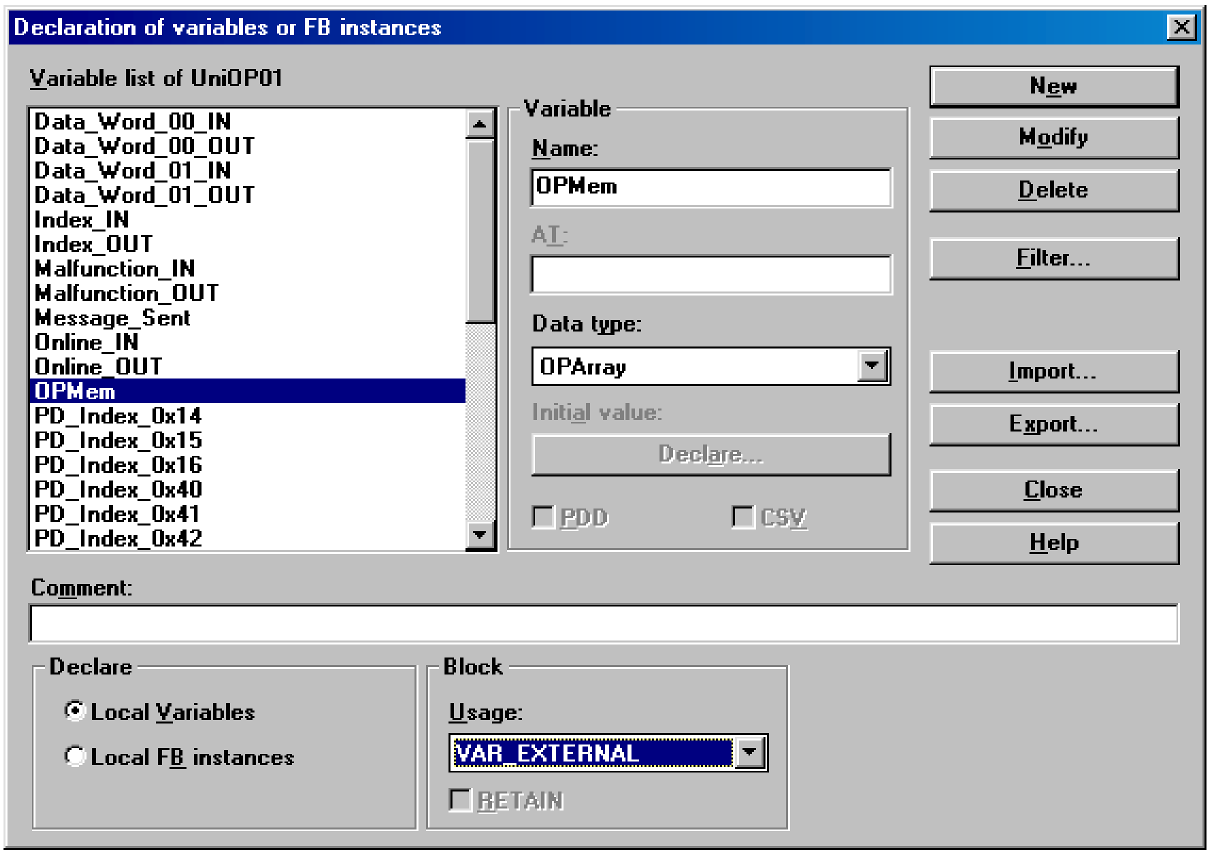
Declare
[✓ Local Variables] Local FB instances

Block
Usage: VAR
☐ RETAIN

Figure 1 – Variable declaration dialog box

5.1 How to Use Global Memory to Interface UniOP

The array allocated to interface the operator panels can be configured to be global i.e. the same for all the attached panels. If this is the case, the OPMem array of type OPArray should be declared as EXTERNAL as shown in Figure 2 by selecting the proper usage scope.



Name: OPMem
Data type: OPArray
Declare: [✓ Local Variables] Local FB instances
Usage: VAR_EXTERNAL

Figure 2 – OPMem declared as VAR_EXTERNAL

Please note that whenever you change this setting, a complete new parameterisation of the Interbus card must be performed.

5.2 Addressing Mode

In accordance with the MMI-COM profile the system will map the variable objects in this PLC area using the *Byte Inverted Order*. The programmer will enter addresses in the Designer software as variable indexes.
The table below shows the byte addressing mode.

Variable Index Byte Data Format	Array Addressing
V0	MS Byte of OPMem[0]
V1	LS Byte of OPMem[0]
V2	MS Byte of OPMem[1]
V3	LS Byte of OPMem[1]
V4	MS Byte of OPMem[2]
…	…

Table 3

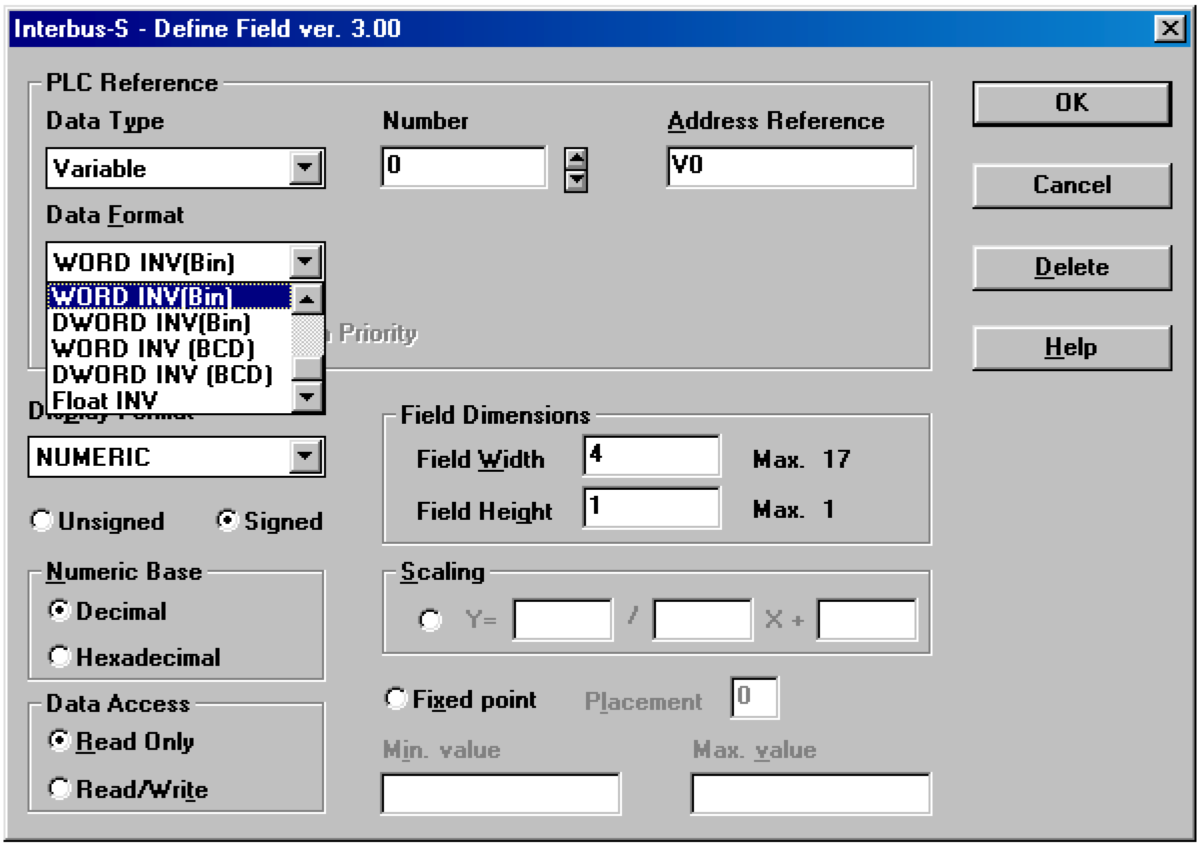
The following table shows instead the word addressing mode.

Variable Index Inverted Word Data Format	Array Addressing
V0	OPMem[0]
V2	OPMem[1]
V4	OPMem[2]
V6	OPMem[3]
V8	OPMem[4]
…	…

Table 4

Note: *Access to odd V addresses is not permitted because of the word organisation of the OPMem array.*

To display correctly the contents of the array elements, you must use the Data Format WORD INV from the Designer, as shown in the figure below.



Interbus-S - Define Field ver. 3.00

PLC Reference
Data Type: Variable
Number: 0
Address Reference: V0
Data Format: WORD INV(Bin)
WORD INV(Bin)
DWORD INV(Bin)
WORD INV (BCD)
DWORD INV (BCD)
Float INV
Display Format: NUMERIC
Unsigned [✓ Signed]
Numeric Base: [✓ Decimal] Hexadecimal
Data Access: [✓ Read Only] Read/Write
Field Dimensions
Field Width: 4 Max. 17
Field Height: 1 Max. 1
Scaling
Fixed point Placement 0
Min. value
Max. value

Figure 3 – Data Format in the Data Field Definition Dialog Box

5.3 Configuring UniOP in the Interbus Network with SYSTEM WORX

As an MMI-COM compliant device, UniOP can be directly inserted into the chain with the ID Code 47 (dec). In alternative you can connect it to the bus and perform a “Read Configuration” function from the software.
The program provided as an example includes two UniOP panels; the resulting configuration layout appears as shown in Figure 4.

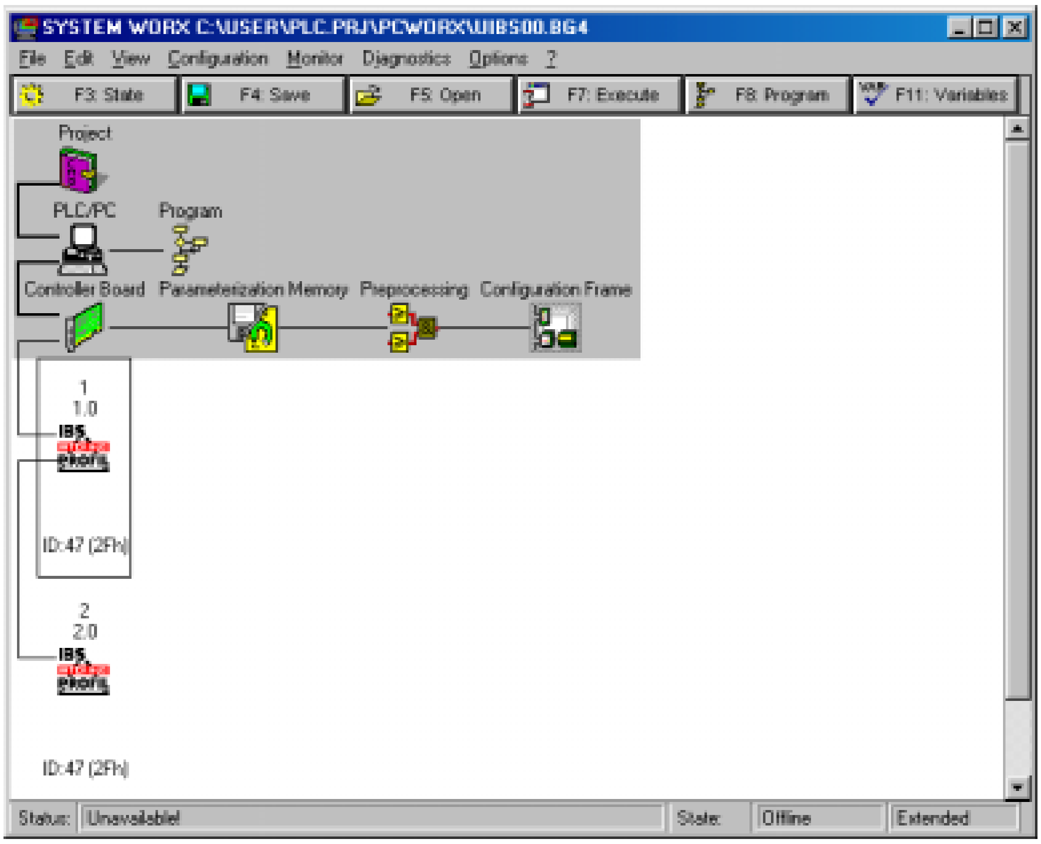


Figure 4 – SYSTEM WORX configuration of the sample program

Configuring UniOP in SYSTEM WORX means assigning to each panel the proper “Process Data Description”.
Depending on which panel is handled, the user have to manually assign the I/O variables defined in the project to the Input and Output buffer of each panel.

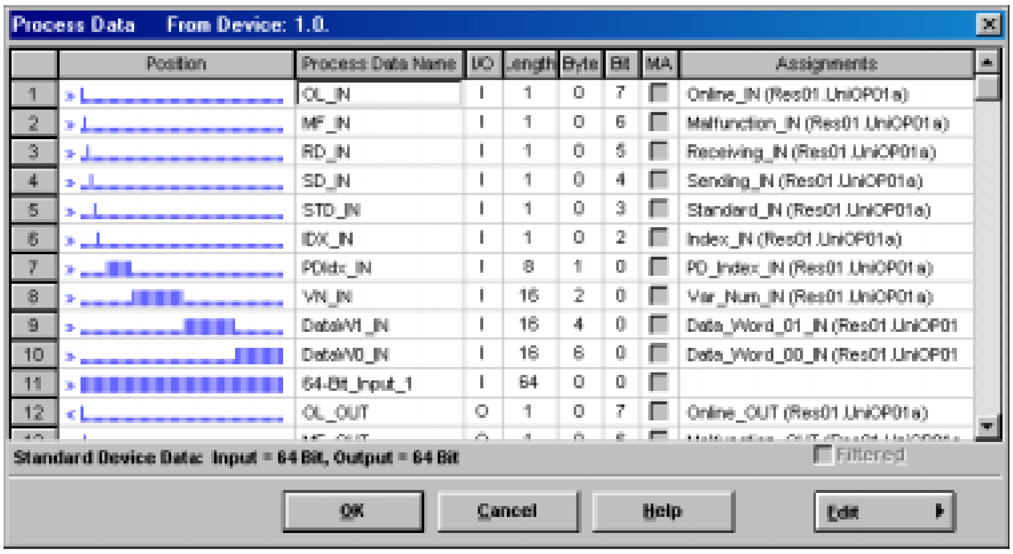
The 64-bit Input buffer of UniOP is divided in variables; each variable is assigned a Process Data Name. The user will have to define the association between Process Data Names and the corresponding variables defined in the PLC program (see below Chapter 5.4 Using the UniOP POU in PC WORX Automation Software).

Table 5 below shows the mapping between the Process Data Name and the variables for the sample program.

Process Data Name	I/O	Length	Byte	Bit	MA	Assignments
OL_IN	I	1	0	7		Online_IN (Res01.UniOP01a)
MF_IN	I	1	0	6		Malfunction_IN (Res01.UniOP01a)
RD_IN	I	1	0	5		Receiving_IN (Res01.UniOP01a)
SD_IN	I	1	0	4		Sending_IN (Res01.UniOP01a)
STD_IN	I	1	0	3		Standard_IN (Res01.UniOP01a)
IDX_IN	I	1	0	2		Index_IN (Res01.UniOP01a)
PDIdx_IN	I	8	1	0		PD_Index_IN (Res01.UniOP01a)
VN_IN	I	16	2	0		Var_Num_IN (Res01.UniOP01a)
DataW1_IN	I	16	4	0		Data_Word_01_IN (Res01.UniOP01a)
DataW0_IN	I	16	6	0		Data_Word_00_IN (Res01.UniOP01a)
OL_OUT	O	1	0	7		Online_OUT (Res01.UniOP01a)
MF_OUT	O	1	0	6		Malfunction_OUT (Res01.UniOP01a)
RD_OUT	O	1	0	5		Receiving_OUT (Res01.UniOP01a)
SD_OUT	O	1	0	4		Sending_OUT (Res01.UniOP01a)
STD_OUT	O	1	0	3		Standard_OUT (Res01.UniOP01a)
IDX_OUT	O	1	0	2		Index_OUT (Res01.UniOP01a)
PDIdx_OUT	O	8	1	0		PD_Index_OUT (Res01.UniOP01a)
VN_OUT	O	16	2	0		Var_Num_OUT (Res01.UniOP01a)
DataW1_OUT	O	16	4	0		Data_Word_01_OUT (Res01.UniOP01a)
DataW0_OUT	O	16	6	0		Data_Word_00_OUT (Res01.UniOP01a)

Table 5 – Process Data Description and Assignments

Table 5 describes the assignments done for the panel handled by the instance UniOP01a of the program UniOP01. The second panel will have a Process Data Description similar to this one with the only exception that the instance UniOP01b will handle the variables coming from the second operator panel.
Figure 5 shows the dialog box where the assignment is done for the first panel.



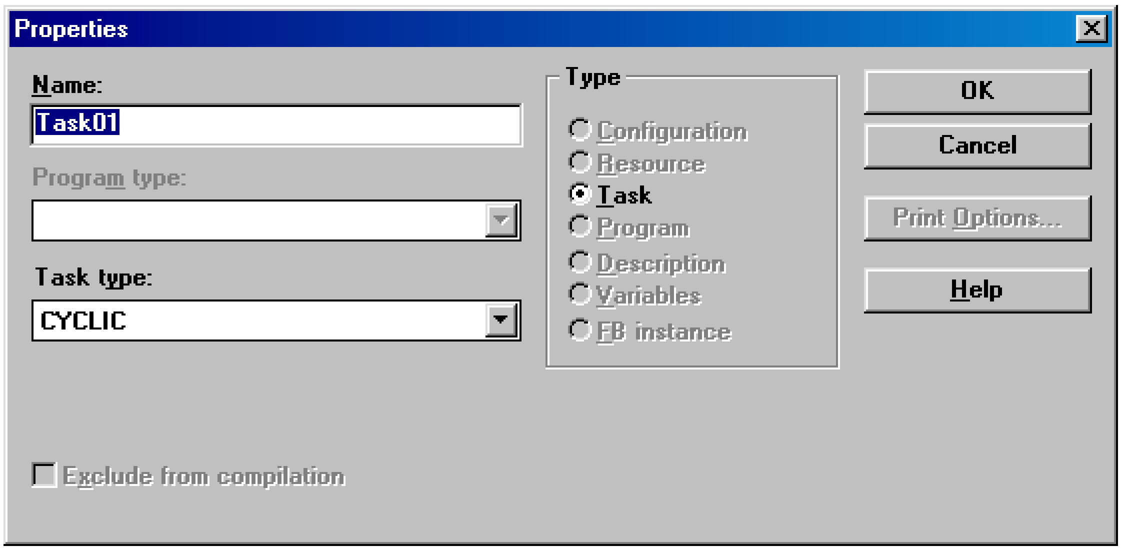
	Position	Process Data Name	I/O	Length	Byte	Bit	MA	Assignments
1		OL_IN	I	1	0	7	☐	Online_IN (Res01.UniOP01a)
2		MF_IN	I	1	0	6	☐	Malfunction_IN (Res01.UniOP01a)
3		RD_IN	I	1	0	5	☐	Receiving_IN (Res01.UniOP01a)
4		SD_IN	I	1	0	4	☐	Sending_IN (Res01.UniOP01a)
5		STD_IN	I	1	0	3	☐	Standard_IN (Res01.UniOP01a)
6		IDX_IN	I	1	0	2	☐	Index_IN (Res01.UniOP01a)
7		PDIdx_IN	I	8	1	0	☐	PD_Index_IN (Res01.UniOP01a)
8		VN_IN	I	16	2	0	☐	Var_Num_IN (Res01.UniOP01a)
9		DataW1_IN	I	16	4	0	☐	Data_Word_01_IN (Res01.UniOP01
10		DataW0_IN	I	16	6	0	☐	Data_Word_00_IN (Res01.UniOP01
11		64-Bit_Input_1	I	64	0	0	☐	
12		OL_OUT	O	1	0	7	☐	Online_OUT (Res01.UniOP01a)

Standard Device Data: Input = 64 Bit, Output = 64 Bit

Figure 5 – Process Data

5.4 Using the UniOP POU in PC WORX Automation Software

The UniOP support program for PC WORX is made of only one POU called UniOP01 written in ST language.
One task to handle UniOP communication is created with the name Task01 and Task Type CYCLIC execution as shown in Figure 6.


Properties

Name: Task01

Program type:

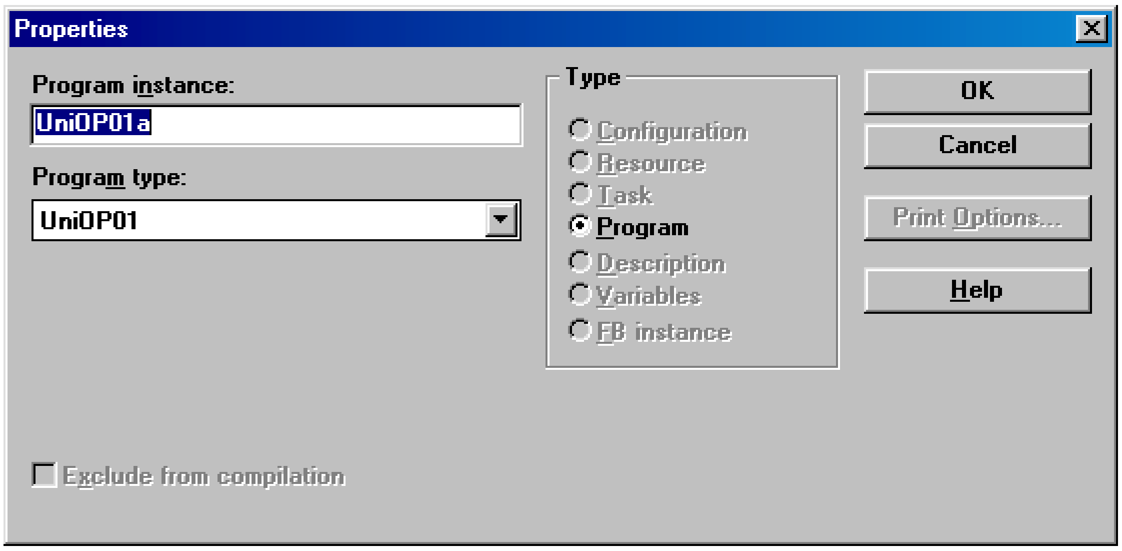
Task type: CYCLIC

Type: ( ) Configuration ( ) Resource [✓ Task] ( ) Program ( ) Description ( ) Variables ( ) FB instance

☐ Exclude from compilation

Figure 6 – Task01 properties dialog box

You can include into Task01 one or more instances of the program UniOP01. Figure 7 shows the definition for one instance of UniOP01 called UniOP01a.


Properties

Program instance: UniOP01a

Program type: UniOP01

Type: ( ) Configuration ( ) Resource ( ) Task [✓ Program] ( ) Description ( ) Variables ( ) FB instance

☐ Exclude from compilation

Figure 7 – Program instance definition in Task01

The sample program provided as a template has been written to support two UniOP panels. To add one or more additional operator panels to the Interbus network, just create new instances of the UniOP01 program in Task01.
Figure 8 shows the project tree configured for two operator panels handled by instance UniOP01a and UniOP01b.

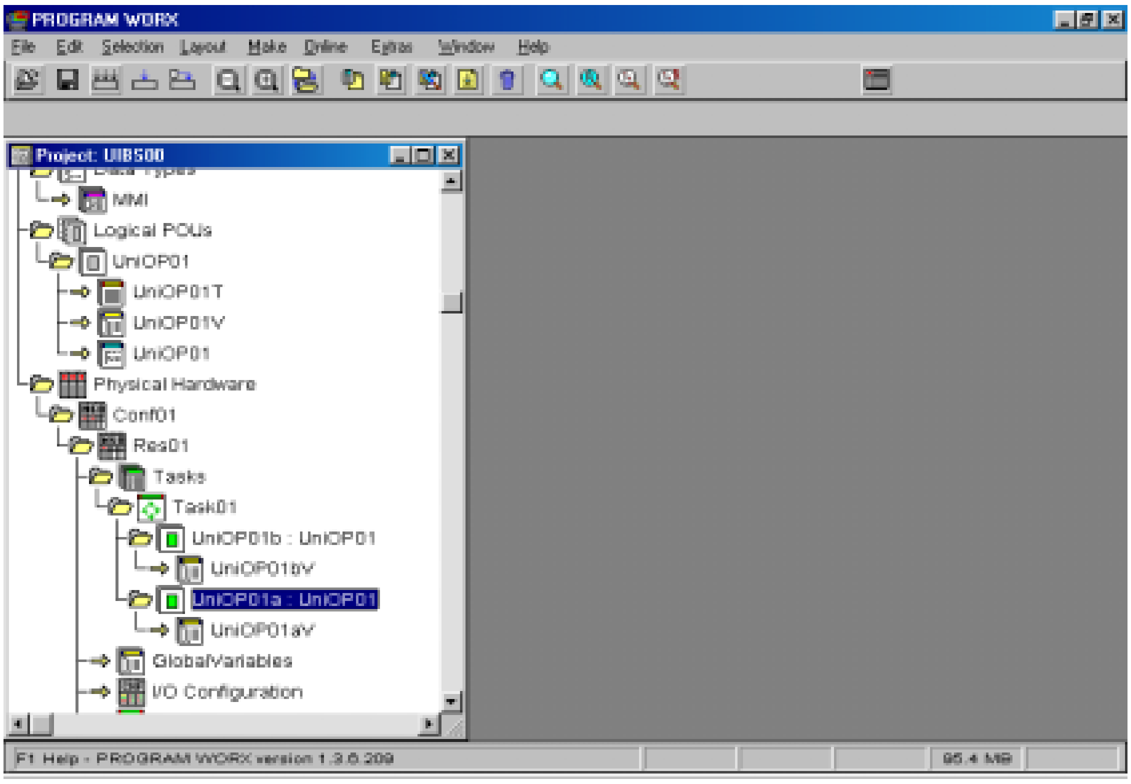


Figure 8 – Project tree of the sample UniOP support program.

6. Siemens Simatic S7 300 Series with Phoenix Contact IBS S7 300 BC-T and IBS S7 300 DSC-T

This chapter describes the application of the UniOP Operator Panels in an Interbus system based on a Simatic S7 300 PLC equipped with a Phoenix Contact **IBS S7 300 BC-T/IBS S7 300 DSC-T** Interbus controller modules.
The information necessary for the correct installation of the controller board is available in the technical documentation from Phoenix Contact and will not be reproduced here.

6.1 Addressing Mode

The current implementation of Interbus for UniOP allows the operator panel to access a contiguous array of registers in the PLC. This array can be positioned either in the Merker area or in a Data Block. In accordance with the MMI-COM profile the system will map the variable objects in the selected PLC data area. The programmer will enter PLC addresses in the Designer software as variable indexes.

Variable Index Byte	**Merker Addressing**
V0	MB0
V1	MB1
V2	MB2
…	…

Example 1: Variable data allocated in the Merker Area

Variable Index Byte	**DB Addressing**
V0	DBB0
V1	DBB1
V2	DBB2
…	…

Example 2: Variable data allocated in Data Block 10

Table 6

Variable Index Word	**Merker Addressing**
V0	MB0 – MB1
V2	MB2 – MB3
V4	MB4 – MB5
…	…

Example 1: Variable data allocated in the Merker Area

Variable Index Word	**DB Addressing**
V0	DBB1 – DBB0
V2	DBB3 – DBB2
V4	DBB5 – DBB4
…	…

Example 2: Variable data allocated in Data Block 10

Table 7

The MMI-COM server functionality requires some special function blocks to be included in the PLC program. They must be called cyclically in OB1 to allow data exchange between the PLC and the HMI panel.

6.2 Starting Interbus

Interbus will start when the power supply of the PLC is turned on.
It is possible to start Interbus by means of the PLC program or by switching the power supply of the PLC off and on again.

The S7 sample project does not provide any control of the IBS registers and IBS will not be started automatically if, for instance, a panel has a power fault. Please note that when an operator panel is in Configuration Mode it does not interfere with the proper operation of the IBS chain.
Phoenix Contact provides a set of function blocks that could be used by the user to properly set-up the IBS errors handling system.

6.3 UniOP Support Program in the Master PLC for IBS S7 300 BC-T

The Interbus data can be accessed directly by the PLC user program using standard Function Blocks provided by Phoenix with the Interbus hardware.

The PLC support program for UniOP uses the following Phoenix Function Blocks:

IB_INI FC20
The IBI_INI function block is called during start-up (OB100). It synchronises the PLC CPU with the controller board. It returns when the controller board is ready for operation.

In the sample program provided the function block is called with the following syntax.

```
CALL  FC    20
       IB_ADR   :=256
       IBDB     :=DB20
       ZEIT     :=T1
       IN_BYTES :=8
       OUT_BYTES:=8
       ERR      :=MB9
```

IB_READ FB21
At the beginning of the program cycle (OB1), the IB_READ function block reads the input data of the controller board to the CPU memory.

In the sample program provided the function is called with the following syntax.

```
CALL  FB    21 , DB21
       IBDB   :=DB20
       COPY_OK:=M10.0
       ERROR  :=DB10.DBW56
       DEST   :=P#DB30.DBX 0.0 BYTE 200
```

IB_WRITE FB22
At the end of the program cycle (OB1) the IB_WRITE function block transfers the output data from the CPU memory to the controller board. With the SOURCE parameter the user can indicate the data source.

In the sample program provided the function is called with the following syntax.

```
CALL  FB    22 , DB22
       IBDB   :=DB20
       CONS   :=M10.1
       COPY_OK:=M10.2
       ERROR  :=DB10.DBW58
       SOURCE :=P#DB31.DBX 0.0 BYTE 200
```

Note: *For the UniOP application it is of vital importance to enable the Consistency Mode by setting to "1" the input parameter* `CONS`*. The value "1" ensures that the input and the output data is consistently written or read in the same Interbus cycle. The setting also applies to the IB_READ function block.*

Complete description of the Phoenix function blocks is available from Phoenix Contact in the document **IBS S7 300 BC SWD UM E**.

Additionally to the standard function blocks, the OB1 must include cyclical call to the UniOP function block that has been named FC7 in the sample project.

The UniOP function block has to be called with the following syntax:

```
CALL  FC     7
       DataType:=B#16#1      // DataType. 1 = Merker; 0 = Data Block
       DBNumber:=DB10        // DB Number
       INAddr  :=W#16#0      // Interbus input address
       OUTAddr :=W#16#0      // Interbus output address
       DB_IN   :=DB30        // DB used in FB21
       DB_OUT  :=DB31        // DB used in FB22
```

If you have more than one UniOP panel connected as Interbus slaves, you will have to issue a call to FC 7 for each of the HMI panels that have to be serviced.

The FC7 function is parameterised according the following description:

DataType	Defines the access area for the current panel. Value 1 means that UniOP panel will access to the PLC merker area according with the addressing mode explained in Table 6 and Table 7.
INAddr	Defines the Input offset address of the Input buffer of the current panel. It has to be manually calculated by the user according with the Interbus network layout.
OUTAddr	Defines the Output offset address of the Output buffer of the current panel. It has to be manually calculated by the user according with the Interbus network layout.
DN_IN	Defines the Data Block number used as temporary data storage by the IB_READ function.
DN_OUT	Defines the Data Block number used as temporary data storage by the IB_WRITE function.

Note: *The special Function Block FC 7 provided with the UniOP communication driver, does* ***not*** *handle in any way the Interbus diagnostic area.*

6.4 UniOP Support Program in the Master PLC for IBS S7 300 DSC-T

The Interbus data can be accessed directly by the PLC user program using standard Functions provided by Phoenix with the Interbus hardware.

The PLC support program for UniOP uses the following Phoenix Functions:

INIT_IB FC20

The IBI_INI function block is called during start-up (OB100). It synchronises the PLC CPU with the controller board. Once the controller board and the INTERBUS system have been started, the function will be terminated. After the controller board has been parameterized and the INTERBUS system has been started, the function resets the activation bit (BUSY = 0). The result is displayed in the RET bit. To call FC 20, set this bit memory bit on a positive edge. After processing, the function resets the bit.

In the sample program provided the function block is called with the following syntax:

```
UN     M      10.0
S      M      10.0

CALL   FC     20
 IBDB          :=DB20
 COM_ADR       :=256
 DIAG_STATE    :=0
 DIAG_PARA     :=0
 FKN_START     :=0
 FKN_PARA      :=0
 FKN_STATE     :=0
 MEM_READ      :=21
 MEM_WRITE     :=22
 LOAD          :=0
 BOOT          :=0
 MODE          :=0
 TIMER_NR      :=T1
 SOURCE        :=0
 CONFIGURATION:=DW#16#8000000F
 RET           :=M10.1
 BUSY          :=M10.0
```

MEM_READ FC21

The FC 21 MEM_READ function reads data from the controller board and copies it to the destination area of the PLC to be indicated by the user. It has to be called at the beginning of the user program in the OB1 block. To call FC 21, set this bit memory bit on a positive edge. After processing, the function resets the bit. When the call parameter SOURCE is set to 0, the FC copies data from Interbus to a Data Block.

In the sample program provided the function is called with the following syntax:

```
UN     M      10.2
S      M      10.2

CALL   FC     21
 IBDB         :=DB20
 MODE         :=0
 SOURCE       :=0
 DEST_AREA    :=5
 DEST_AREA_NR:=30
 DEST_OFFSET :=0
 DEST_LENGTH :=20
 RET          :=M10.3
 BUSY         :=M10.2
```

MEM_WRITE FC22
The FC 22 MEM_WRIT function writes data from the source area of the PLC to the specified destination area on the controller board. It has to be called at the end of the user program in the OB1 block. To call FC 22, set this bit memory bit on a positive edge. After processing, the function resets the bit. When the calling parameter DESTINATION is set to 0, the FC copies data from a Data block to the Interbus.

In the sample program provided the function is called with the following syntax:

```
UN     M      10.4
S      M      10.4

CALL   FC     22
 IBDB              :=DB20
 MODE              :=0
 SOURCE_AREA       :=5
 SOURCE_AREA_NR:=31
 SOURCE_OFFSET :=0
 SOURCE_LENGTH :=20
 DESTINATION       :=0
 RET               :=M10.5
 BUSY              :=M10.4
```

Note: *For the UniOP application it is of vital importance to enable the Consistency Mode by setting to "1" the input parameter* `CONS`*. The value "1" ensures that the input and the output data is consistently written or read in the same Interbus cycle. The setting also applies to the IB_READ function block.*

Complete description of the Phoenix function blocks is available from Phoenix Contact in the document **IBS S7 300 DSC SWD UM E**.

Additionally to the standard function blocks, the OB1 must include cyclical call to the UniOP function block that has been named FB7 in the sample project. The FB7 works on local data stored in two data blocks; the Phoenix Functions provide the copy process from the Interbus at the beginning of the PLC cycle and to the Interbus at the end of the PLC cycle.

The UniOP function block has to be called with the following syntax:

```
CALL   FB     7 , DB7
 DataType:=B#16#1      // DataType. 1 = Merker; 0 = Data Block
 DBNumber:=DB10        // DB Number
 INAddr   :=W#16#0     // Interbus Input Address
 OUTAddr  :=W#16#0     // Interbus Output Address
 DB_IN    :=DB30       // DB used in FB21
 DB_OUT   :=DB31       // DB used in FB22
```

The instance data blocks for the different calls to the FB7 function block, are automatically created by the Step7 software when the operator confirm the code line: `CALL   FB     7 , DB7.`
The Step7 software can recognize that the DB is not present and it asks for an automatic creation of the data block. Each instance data block uses 12 bytes.
A second panel can be inserted in the configuration simply making a new call to the FB7.
The example program is ready to run two UniOP panels.

The FB7 function is parameterised according the following description:

DataType	Defines the access area for the current panel. Value 1 means that UniOP panel will access to the PLC merker area according with the addressing mode explained in Table 6 and Table 7.
INAddr	Defines the Input offset address of the Input buffer of the current panel. It has to be manually calculated by the user according with the Interbus network layout.
OUTAddr	Defines the Output offset address of the Output buffer of the current panel. It has to be manually calculated by the user according with the Interbus network layout.
DN_IN	Defines the Data Block number used as temporary data storage by the IB_READ function.
DN_OUT	Defines the Data Block number used as temporary data storage by the IB_WRITE function.

Note: *The special Function Block FB 7 provided with the UniOP communication driver, does* ***not*** *handle in any way the Interbus diagnostic area.*

The Support program for the Phoenix Controller card IBS S7 300 DSC-T is designed to use a Function Block instead of a Function.
A function block is a block “with memory." It is assigned a data block as its memory (instance data block). The parameters that are transferred to the FB and the static variables are saved in the instance DB. Temporary variables are saved in the local data stack. Data saved in the instance DB are not lost when execution of the FB is complete. Data saved in the local data stack are, however, lost when execution of the FB is completed.
A function is a logic block “without memory." Temporary variables belonging to the FC are saved in the local data stack. This data is then lost when the FC has been executed. To save data permanently, functions can also use shared data blocks. Since an FC does not have any memory of its own, you must always specify actual parameters for it. You cannot assign initial values for the local data of an FC.
The Function Block FB7 implements a consistency check of the frame buffer received from the panel and can be freely used also for the other S7 Phoenix equipment (IBS S7 300 BC-T). Consistency check can be only

6.5 Sample PLC Program

A sample PLC support program for Interbus MMI-COM communication is available. The program is provided as an archived STEP7 project file and is based on an Interbus network including two operator panels: the first one will access the Merker Area, the second one will access the data block DB10.
The name of the sample program is **INTS02**.

6.6 Sample Application

In this section you will find the copy of the OB1 block programmed in the sample application including 2 UniOP panels attached to an IBS controller board placed in slot number 0 of an S7 314 CPU.
The Input address for the first panel is therefore set to 0 and the Output address to 0. The second panel is mapped to 8 for the inputs and for the outputs.

The OB1 block has the following structure:

```
//     Standard Function Call to fetch data from the bus
//
       CALL  FB    21 , DB21
        IBDB   :=DB20
        COPY_OK:=M10.0
        ERROR  :=DB10.DBW56
        DEST   :=P#DB30.DBX 0.0 BYTE 200 // Temp DB for incoming data
//
//     First operator panel
//
       CALL  FC     7
        DataType:=B#16#1      // DataType. 1 = Merker; 0 = Data Block
        DBNumber:=DB10        // DB Number
        INAddr  :=W#16#0      // Address of Interbus inputs
        OUTAddr :=W#16#0      // Address of Interbus outputs
        DB_IN   :=DB30        // DB used by FB21
        DB_OUT  :=DB31        // DB used by FB22
//
//    Second operator panel
//
       CALL  FC     7
        DataType:=B#16#0
        DBNumber:=DB10
        INAddr  :=W#16#8
        OUTAddr :=W#16#8
        DB_IN   :=DB30
        DB_OUT  :=DB31


// ### USER PROGRAM STARTING POINT ###


// ### USER PROGRAM END ###

//     Standard Function Call to copy data to the bus
//
       CALL  FB    22 , DB22
        IBDB   :=DB20
        CONS   :=M10.1
        COPY_OK:=M10.2
        ERROR  :=DB10.DBW58
        SOURCE :=P#DB31.DBX 0.0 BYTE 200 // Temp DB for outgoing data

       BE
```

7. Configuring UniOP with Designer for Interbus

To configure a UniOP operator panel with the Designer software for use with Interbus, follow the procedure described in this chapter.

1) Select the option "Project/Change Communication Driver" and choose "Interbus-S". There is no need to enter additional parameters in the Controller Setup dialog box as shown in Figure 9.

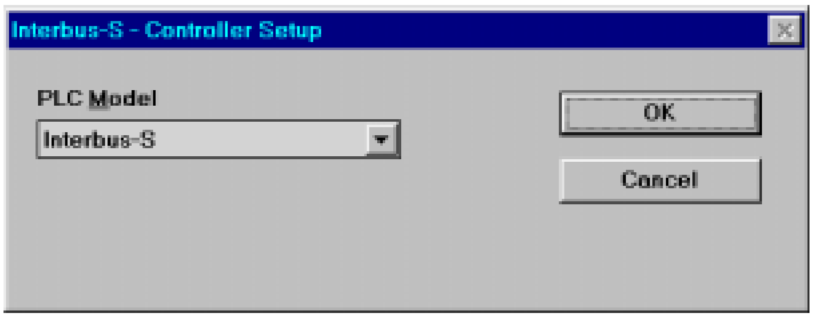


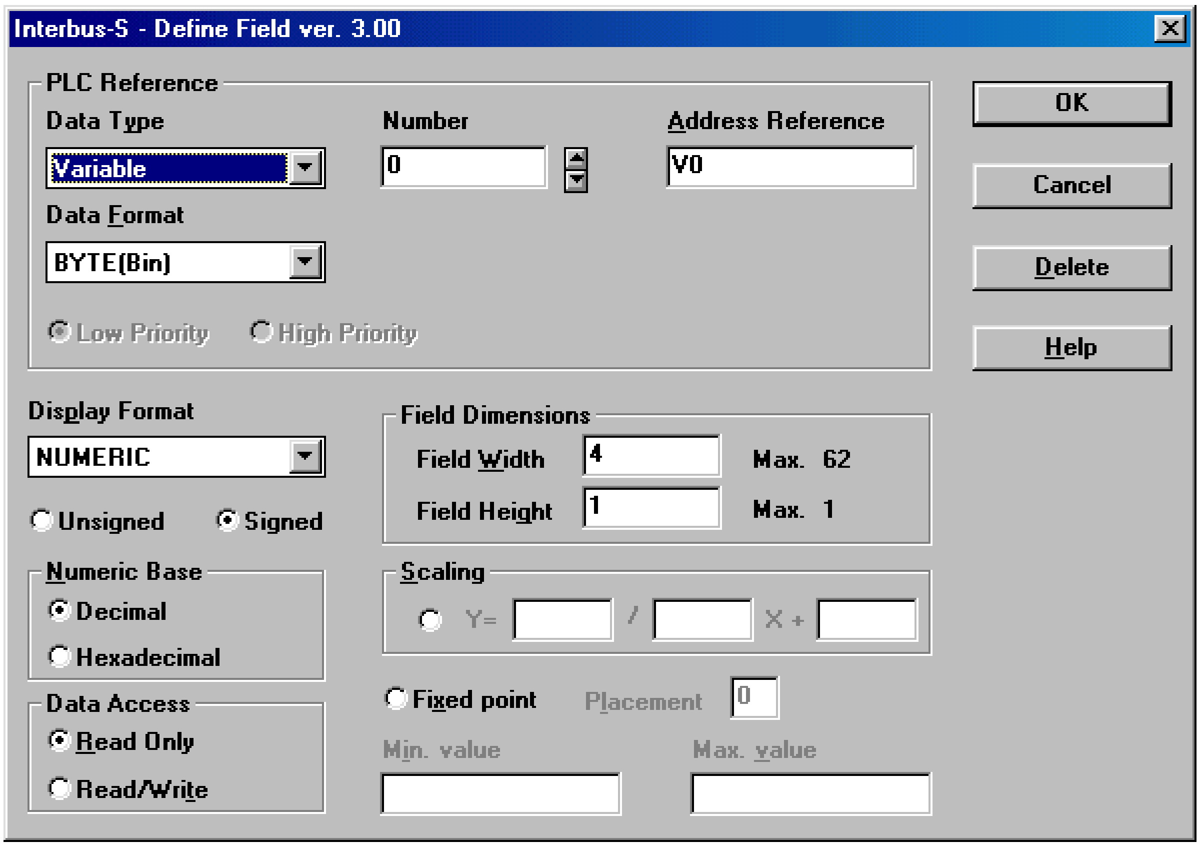
Figure 9 – Controller Setup

2) The Real Time Clock information in the Reserved Data Area (RDA) is coded in BCD

3) The page number displayed and the page number requested in the RDA are coded in binary.

4) The RDA can be freely positioned within the PLC memory.

The Define Field dialog box contains only the Data Type "Variable" and the user can select the desired offset. The PLC program will interpret the offset value according to the data definition selected by the programmer.
“Variable” memory”, depending on the configuration, can be mapped in Data Blocks or Merkers (Simatic S5) or in a user-defined memory array (PCWORX).
Figure 10 shows the Data Field dialog box for the Interbus communication driver.


Interbus-S - Define Field ver. 3.00

PLC Reference
Data Type: Variable
Number: 0
Address Reference: V0
Data Format: BYTE(Bin)
[✓ Low Priority] High Priority

Display Format: NUMERIC
Unsigned [✓ Signed]

Numeric Base: [✓ Decimal] Hexadecimal

Data Access: [✓ Read Only] Read/Write

Field Dimensions
Field Width: 4 Max. 62
Field Height: 1 Max. 1

Scaling: Y= / X +

Fixed point Placement: 0
Min. value
Max. value

Figure 10 – Data Field dialog box

8. Summary

Here is a summary of what you need to do to get your UniOP up and running as an Interbus slave.

1. Set-up the Interbus master controller for operation.
2. Determine the address space assigned by the Interbus network to the operator panels. Refer to the available Phoenix Contact documentation.
3. Include the UniOP function blocks in the PLC program.
4. Make sure the UniOP support program is called at each PLC cycle.
5. Configure the UniOP Comm DB (if required) or the calls to the function blocks to match the Input and Output address specified for the UniOP panels and other parameters.
6. Create the project files for the UniOP panels.
7. Start the program in the PLC.

Appendix 1 - Communication Error Codes

This section is applicable to all master systems and it is PLC independent.
Current communication status is displayed in the System Menu of the UniOP.
A message and a numeric error code describe the communication error status. The message describes the current communication status. The number shows the code of the current communication error and, if the communication is currently correct, the code of the last error encountered. When the error code 0 is shown, it means there have been no communication errors since this system start-up.
The codes are the following:

Code	Description	Notes
00	No errors	There are no communication errors and there have been no errors since start-up.
04	Response NAK	Indicates that the partner device has set the malfunction bit in the Interbus status byte. It means that panel is attempting to access to a non existing variable
05	Timeout	The Interbus master did not respond within the predefined period of time.
06	Ill formed response	Means that a wrong command ID or a wrong variable number was used.
07	Internal software error	Should never happen; it indicates an error in the communication driver software.

Appendix 2 - Technical Data and Connection Information

The main technical information on the UniOP Interbus interface using TCM04 is shown in the table below:

IB Device Type	Remote Bus
Data Channel Size	8 bytes – Indirect process data with status word
Identification Code	0x2F [hex] / 47 [dec]
Profile Compatibility	MMI-COM
Function Group Specification	B3 G1
Optical insulation	Yes

The Aux Port is used for the Interbus communication. When the TCM04 communication module is mounted, the Aux Port becomes an Interbus connector. The physical connection to the Interbus bus requires an adapter type TCM-04-IBT.
A simple point to point connection can be performed with the cable drawing CA161 as a reference.

Appendix 3 - Requirements and Compatibility

This version of Interbus is contained in the Designer DLL file UPLC121.DLL. The initial release level is 3.00 for the communication driver and 5.00 for the DLL (both version numbers can be seen in the Change Controller Driver dialog box of the Designer software).

A communication module of type TCM04 is required. The physical connection to the Interbus bus requires an adapter type TCM-04-IBT.
The UniOP panel must have hardware type –0045 and firmware version number 4.20 or higher to support the TCM04 modules.

Appendix 4 - Available Support Programs

PLC support programs are available to reduce application set-up time to users.
You will find below a list of the configurations that have been tested and are available for use as of now.

The following programs are currently available:

Siemens Simatic S5 AG95U using Phoenix Contact S5 100 CB-T
Siemens Simatic S5 AG115U CPU 943B using Phoenix Contact S5 DSC /I-T
Siemens Simatic S7 314 with Phoenix Contact IBS S7 300 BC-T
Phoenix Contact PCWORX with IBS ISA FC/I-T

Other configurations, though possible, have not been tested. The required support programs are not available.