Connecting UniOP to Profibus DP via HMI Profile

This Technical Note contains all the information required to connect the UniOP panels to a Profibus DP system taking advantage from the flexibility and easy programmability of the UniOP 'Profibus DP HMI' communication driver.

Note: *This Technical Note applies only to the Profibus DP communication driver identified by the name "Profibus DP HMI" and associated to the Designer file UPLC142.DLL. To run this communication driver it is required a panel of hardware type –0045 equipped with a TCM08 communication module.*

Contents

1. Introduction

This Technical Note describes how to use the UniOP operator panels in a Profibus DP network according to the PNO draft standard profile 'Profile for Human Machine Interface Systems (HMI)'. This implementation is based on the draft specification document Revision 1.0 - January 1998.

A standard device profile for HMI product will allow a vendor-independent approach in the choice of HMI products to be used in a Profibus DP system.

The draft version of the HMI profile as edited by PNO is not complete and cannot guarantee inter-operability of HMI products.
The UniOP Profibus DP HMI protocol adheres to the standard as much as possible. Where the standard is lacking, the simplest solution has been chosen. The conversion to the future final release of the profile will be as smooth as possible.

2. Basic Concepts of Profibus Profile for HMI

Connecting a complex HMI device to a Profibus DP system is very demanding for the bus. Profibus DP has been designed to be the ideal solution to connect simple I/O devices to a controller. HMI products, instead, need to exchange large amounts of data with the controller not cyclically but on demand. Usually, this may accomplished with the support of an appropriate program unit running on the master controller. The concept of Device Profiles has been introduced to make these solutions more portable and vendor-independent.
The Profibus Profile for HMI defines the foundation for an application layer protocol to be hosted by a Profibus DP network. This definition is independent of the Profibus DP master in the sense that it provides the means to customize its operation for any individual Profibus DP master system.
The resulting solution is easily portable to any Profibus DP master.
The user interface of the programming tool of the HMI adapts itself to show the typical addressing model of Profibus DP controller to which the panel is going to be connected.
Customization is possible using a file that contains data for all the different controller types.
This file is UPLC142.INI and must be present in the Designer directory.

Note: *If the file UPLC142.INI is not present in the Designer directory, the Designer will only show a generic syntax for the controller.*

Operation of the UniOP protocol is based on the application layer as defined by the Profibus Profile. The UniOP protocol places request codes in the Profibus DP input buffer of the controller reserved to the panel. An application program running in the controller must process these request codes. This program takes processes the commands received by UniOP and places the requested data in the Profibus output buffer assigned to the operator panel.

2.1. Profibus Profile Details

This chapter provides details on the specification of the HMI profile and describes the subset of the request/response formats used by this implementation of the protocol.

The Profibus Profile supports a kind of Datagram, this means it is possible to retrieve multiple Index items in a single request.
The Profibus Profile can also support reading and writing to direct bits and Fast Bits.

Note: *Current version of the protocol (V3.00) does not support multiple Index items; does not support reading and writing direct bits and does not support Fast Bits.*

2.1.1. Request Format: Panel to Controller

The format of the Data Exchange data sent to the controller is described in Table 1

Byte Number	Description
0	Request Control Byte
1	Fast Bits 0…7
2	Fast Bits 8…15
3	Fast Bits 16…23
4	OrderControlBlock 1
…	
…	
4 + size of Order Control Block 1	OrderControlBlock 2
…	
…	
4 + size of Order Control Block 1 + size of Order Control Block 2	OrderControlBlock 3
…	
…	
	0

Table 1 – Request format

The request consists of a RequestControlByte, some FastBits and multiple Order Control Blocks; Order Control Blocks can have different sizes. The last Order Control Block is followed by a byte containing the value 0.

2.1.1.1. Request Control Byte

The Request Control Byte is defined as shown in Table 2.

Bit	Description
7	Request Control Bit
6	Communication Bit
5	Toggle Request Bit
4	Error Request Bit
3	Acknowledge Bits
2	
1	
0	

Table 2 – Request Control Byte

Request Bit when assembling the Request Control Byte, UniOP checks the current value of the Response Bit in the last Response from the PLC. If the Response Bit is 0 then it places a value of 1 in the Request Bit; if the Response Bit is 1 then it places a value of 0 in the Request Bit. The controller program that handles incoming requests from UniOP can monitor this bit; when it sees that the value of this bit has changed then it knows that a new Request is available from the UniOP.

COM Bit this bit is always set to 1 by UniOP. The controller program uses the Toggle Bit to determine if the communication with the UniOP is still alive. If it determines that its internal Toggle Bit variable has stopped toggling then it sets the value of the COM Bit in the image of the data from the UniOP to 0.

In this way it can signal that the data from the UniOP is not being updated and that the connection to the UniOP is down.

Toggle Bit UniOP has an internal variable called Toggle. Whenever UniOP receives the data exchange response from the controller it reads the value of the Toggle Bit in the response from the controller and sets the value of its Toggle Internal Variable to the value of the Toggle Bit in the Response from the PLC. Also whenever UniOP is composing a request to send to the controller it first toggles the value of the Toggle internal variable and places the new value in the Toggle Bit of the request. This means that the value of the Toggle Internal variable in UniOP should be constantly changing between the values 0 and 1; it can be used as a watchdog. If it stops changing its value then either the UniOP processing or the PLC processing has been stopped.

Error Bit UniOP sets this bit to 1 if an error has occurred. This should never happen.

Acknowledge Bits not used in this implementation.

2.1.1.2. Order Control Block

The Order Control Block is divided in 3 parts.
The Order Control Byte specifies what type of operation is requested, i.e. READ/WRITE and how many bytes should be read or written.
The Index specifies the address in the controller where the READing or WRITEing is to take place.
Finally, for WRITE operations there is the actual data to be written to the controller.

Byte	Description
0	Order Control Byte
1	Index
…	
…	
1 + size of Index	Data for WRITE operation
…	
…	
1 + size of Index + size of Data	0

Table 3 – Order Control Block

2.1.1.2.1. Order Control Byte

The Order Control Byte has the format described in Table 4.

Bit	Description
7	Service Type
6	
5	Address Mode
4	
3	Data Length –1
2	
1	
0	

Table 4 – Order Control Byte

ServiceType specifies the operation requested. Possible values are:

0	READ	for reading 0-16 bytes
2	WRITE	for writing 0-16 bytes
1	UPLOAD	for multiple block reads of 0-16 bytes
3	DOWNLOAD	for multiple block writes of 0-16 bytes

Note: *The UniOP protocol only uses the READ and WRITE service types.*

Address Mode specifies the addressing mode for the requested operation. Possible values are:

0	NOP	not used
2	Index without cache	normal operation
1	Cache read	accesses take place through the cache
3	Cache write	the index is written to the cache

Note: *The UniOP protocol only uses the Index without Cache address mode.*

Data length –1 specifies the number of data bytes to be transferred to or from the controller with the READ or WRITE operations. A value of 0 means that 1 byte is to be read or written, a value of 15 means 16 bytes are to be read or written.

Note that the read or write operation will be performed starting from the controller memory address specified by the Index.

2.1.1.2.2. Index

In the Profibus HMI Profile the Index can take different formats depending on whether the PLC Object Reference uses the Reference fields or not. It is assumed that the program running in the controller to interpret the Index requests from the HMI implicitly knows which Object References use the Sub Reference fields and which do not.

2.1.1.2.2.1. Sub Reference 1 and 2 fields NOT used

If the Sub Reference fields are not used, then the format of the Index will be as in the table below.

Byte	Description
0	Object Reference
1	Object Offset low byte
2	Object Offset high byte
3	Object Bit Index (Only present for Bit Access, i.e. if "Bit Mode" = 1)

Table 5 – Index format

Object Reference this specifies PLC object being accessed, i.e. Input, Output, Flag, etc. It also specifies if individual bit is being accessed. The Object Reference byte has the following format:

Bit	Description
7	Bit Mode
6	PLC Object ReferenceCode
5	
4	
3	
2	
1	
0	

Table 6 – Object Reference

PLC Object Reference Code can have values between 0 and 127. Each Object in the controller will have its own unique value, e.g. 0 for Inputs, 1 for Outputs, etc.

Bit Mode used to specify whether the access is to a single bit within a PLC Object. If this bit is 1 then access is to the bit specified in the "Object Bit Index" field. If this bit is 0 then access is not to a single bit.

Object Offset this word specifies the Address Offset of the PLC object. Range is 0-32767. The Address Offset specifies the offset in the intrinsic data units for the Object type in the PLC. For example, if the PLC Object type is made of WORDs then the Address Offset will specify a WORD Offset; if the PLC Object type is made of BYTEs then the Address Offset will specify a BYTE Offset.

Object BitIndex this byte specifies the Bit Index of the bit within the Object Offset. This field is only present if the "Bit Mode" field has been set to 1. If the PLC Object type, as specified by Object Reference, is composed of BYTEs then this field may take a value between 0 and 7; if the PLC Object type is composed of WORDs then this field may take a value between 0 and 15.

2.1.1.2.2.2. Sub Reference 1 field used Sub Reference field 2 NOT used

The Sub Reference 1 field can be used to specify, for example, a Data Block number or a File Number. If it is required for a particular Object Reference then the Index will take the following format:

Byte	Description
0	Object Reference
1	Object Offset LO BYTE
2	Object Offset HI BYTE
3	Object Sub Reference 1 LO BYTE
4	Object Sub Reference 1 HI BYTE
5	Object Bit Index (Only present for Bit Access, i.e. if "Bit Mode" = 1)

Table 7

Object Sub Reference 1 This field can take a value 0 - 32767. It represents exactly a Data Block number or a File Number or some other such sub-address field.

2.1.1.2.2.3. Sub Reference 1 field and 2 used

The Sub Reference 1 field can be used to specify, for example, a Data Block number or a File Number. The Sub Reference field 2 can be used to specify some other sub address number. If they are both required for a particular Object Reference then the Index will take the following format:

Byte	Description
0	Object Reference
1	Object Offset LO BYTE
2	Object Offset HI BYTE
3	Object Sub Reference 1 LO BYTE
4	Object Sub Reference 1 HI BYTE
5	Object Sub Reference 2 LO BYTE
6	Object Sub Reference 2 HI BYTE
7	Object Bit Index (Only present for Bit Access, i.e. if "Bit Mode" = 1)

Table 8

Object Sub Reference 1 This field can take a value 0 - 32767. It represents exactly a Data Block number or a File Number or some other such sub-address field.

Object Sub Reference 2 This field can take a value 0 - 32767. It represents exactly a Data Block number or a File Number or some other such sub-address field.

2.1.1.2.3. About the Request Data Format for Write Operations

The actual request data for write operations, i.e. the data values to be written to the PLC memory will immediately follow the Index that defines write address. The format of the request data will be a raw byte stream. For example, if the panel wants to write the value of 3 WORD sized items then the request byte stream will be composed of 6 bytes of data.
If the WORD units in the PLC are HI-LO ordered (big endian) then the data in the byte stream will also be HI-LO ordered; if, however, the WORD data units in the PLC are LO-HI (little endian) ordered then the data in the byte stream will be LO-HI ordered.

2.1.2. Response Format: Controller to Panel

The Response format from the Controller has the format shown in Table 9.

Byte	Description
0	Response Status Control Byte
1	Fast Bits 0…7
2	Fast Bits 8…15
3	Fast Bits 16…23
4	Data
6	Data
7	Data

Table 9

2.1.2.1. Response Status Byte

The Response Status Byte has the format shown in Table 10

Bit	Description
7	Response Status Bit
6	Communication Bit
5	Toggle Response Bit
4	Error Response Bit
3 2 1 0	Action Bits

Table 10

Response Status the PLC program uses this flag to inform the HMI panel that the request has been processed and that the response data is available. When the PLC program has finished inserting the response data in the data exchange packet, it will check the current value of the Response Status Bit and toggle it, placing the new value back in the response data. In this way the panel, which is monitoring the value of this bit, will be able to determine that its value has changed, and will thus know that the response data is available.

Communication this bit is always set to 1 by the PLC.

Toggle Response the PLC program has an internal variable called Toggle. Whenever the PLC receives the data exchange response from the HMI panel, it reads the value of the Toggle Bit in the response from the panel and sets the value of its Toggle internal variable to the value of the Toggle Bit in the Response from the UniOP. Also, whenever the PLC is composing a response packet to send to the panel, it first toggles the value of the Toggle internal variable and places the new value in the Toggle Bit of the response. This means that the value of the Toggle internal variable in the PLC should be constantly flipping between 0 and 1; it is thus used as a watchdog. If it stops flipping its value then either the processing in the panel or the processing in the PLC has broken down.

Error Response the PLC sets this bit to 1 if an error has occurred. For example, if the HMI panel request packet included an Index that is not valid for the PLC.

Action Bits they are not used by the PLC.

2.1.2.2. Data Format in response Block

The actual response data, i.e. the values of the PLC variables requested by the HMI panel, will immediately follow the fast bits. The format of the response data will be a raw byte stream.
For example, if the panel has requested the value of 3 WORD sized items then the response data stream will be composed of 6 bytes of data.

If the WORD units in the PLC are big endian then the data in the byte stream will also be HI-LO ordered; if the WORD data units in the PLC are little endian then the data in the byte stream will be LO-HI ordered.

2.2. The initialization file

The UPLC142.DLL file requires a special initialization file for proper operation. The file must be present in the Designer installation directory.

Note: *If the initialization file is missing, the communication driver Profibus DP HMI will not be listed in the list of available drivers ('Project/Change Controller Driver…').*

The UPLC142.INI file is divided into a certain number of sections where a PLC model and the corresponding data types can be configured.

Data for a maximum of 50 different PLC types can be specified in this INI file. However, the byte size of this INI file should not exceed 64KBytes.

The data for a PLC model is marked by a [PLCxxx] section where xxx is a decimal number between 1 and 50. i.e. for the 1st PLC type it would be [PLC1], for the second [PLC2] and so on. Please note that there should be no gaps in the PLC number, i.e. [PLC1]..[PLC3] is not a valid sequence.

The data within an individual PLC section has the following format:

```
Name =<plcname>
DirectAccess =<yesorno>
DefModel =<yesorno>
NumericFormat=<numericformat>
ShowDataFormat=<yesorno>
VarType1 =<description>,<brief>,<vt>,<oi>
VarType2 =<description>,<brief>,<vt>,<oi>
VarType3 =<description>,<brief>,<vt>,<oi>
VarType4 =<description>,<brief>,<vt>,<oi>
VarType5 =<description>,<brief>,<vt>,<oi>
VarType6 =<description>,<brief>,<vt>,<oi>
```

where:

`Name =<plcname>` is the name of the PLC model with a maximum of 30-character string;

`DirectAccess =<yesorno>` specifies if the Direct Access feature is enabled or disabled; except for special needs, it is strongly recommended to leave this option disabled;

`DefModel =<yesorno>` allows to specifies the default controller model proposed by Designer when creating a new project with the Profibus DP HMI driver; only one controller configured in the "ini" file should contain in its description this parameter;

`NumericFormat=<numericformat>` allows to specifies the numeric based used for the Object Offset parameters; the selection is done for the controller as a whole and not for individual controller data items; if this option is set to decimal then Object Offset and Sub Reference fields are coded in decimal for all the controller data types, if this option is set to hexadecimal then Object Offset

	and Sub Reference fields are coded in hexadecimal for all the controller data types; a value of “1” means decimal, a value of “2” means hexadecimal;
`ShowDataFormat=<yesorno>`	allows to specifies whether the Logical Address string visible in the Define Field dialog box should or should not contain a character that specifies the data format; the selection is done for the controller as a whole and not for individual controller data items; a value of “1” means that the data format indicator character is inserted in the “Address” box; a value of “0” means that the data format indicator is not included in the “Address” box; the data format indicator is “X” for bit, “B” for byte, “W” for word and “D” for double word;
`<description>`	is the full name of the variable type with a maximum of 30-character string;
`<brief>`	is a brief mnemonic for the variable type with a maximum of 4-character string;
`<vt>`	is a decimal value that defines the variable type capabilities; it will take one of the values shown in Table 11.

Variable Type Code	Description	Direct Bit Access
1	1 Field BYTE	No
2	1 Field WORD	No
3	1 Field Inverted WORD	No
4	2 Field BYTE	No
5	2 Field WORD	No
6	2 Field Inverted WORD	No
7	3 Field BYTE	No
8	3 Field WORD	No
9	3 Field Inverted WORD	No
10	1 Field BYTE	Yes
11	1 Field WORD	Yes
12	1 Field Inverted WORD	Yes
13	2 Field BYTE	Yes
14	2 Field WORD	Yes
15	2 Field Inverted WORD	Yes
16	3 Field BYTE	Yes
17	3 Field WORD	Yes
18	3 Field Inverted WORD	Yes
19	1 Field DWORD	No
20	2 Fields DWORD	No
21	3 Fields DWORD	No
22	1 Field DWORD	Yes
23	2 Fields DWORD	Yes
24	3 Fields DWORD	Yes
25	1 Field Inverted DWORD	No

Table 11

Depending on the configured variable type, the associated parameters will change in the way needed to match the PLC native addressing mode.

Consider the Allen-Bradley addressing mode as an example. The A-B PLCs have a two-level addressing mode for the Integer Data Type: the first is the file number; the second is the offset inside the file. In order to properly address these data we need a “2-fields” address.
The addressing inside an Integer File is word-oriented. In addition the word format is the big endian (high byte first).
Considering all these three conditions, the Variable Code we need to configure for Integer file access is 5.

`<oi>` is a decimal value that specifies the Object Index ID as explained in the Profibus Profile. This code is used to distinguish different Data Type configured for the same PLC. This parameter assumes values from 1 to 6.

3. Configuring Designer

Once the INI files has been properly configured, you will have to select the PLC model from the list presented by the Designer software in the Controller Setup dialog box.

If the INI file has not been configured properly or if the user needs to define a special PLC access mode, one additional PLC Type is available.
It is called “User Defined” and it is shown as default on the “Controller Setup” dialog box as in Figure 1.

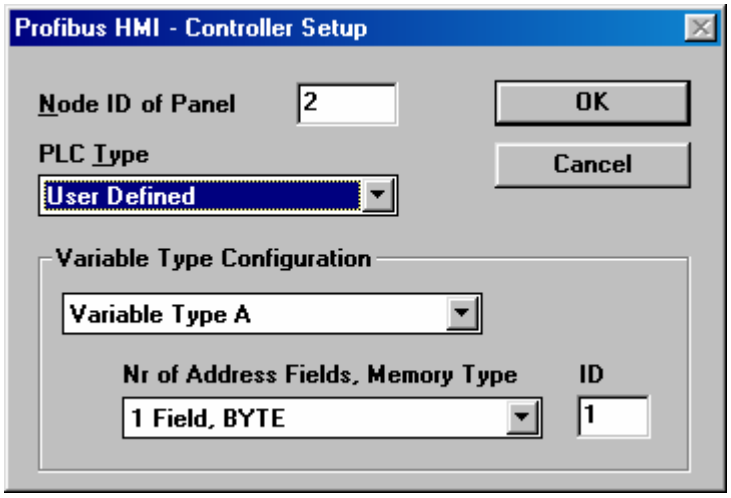

Figure 1 – Controller Setup Dialog Box

The “Variable Type Configuration” section contains all the information the Designer software needs for each generic variable called “A”, “B”, “C”, and so on.

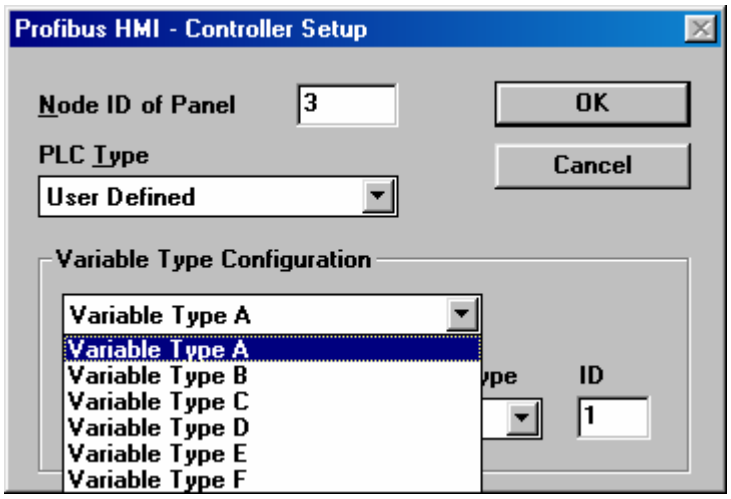

Figure 2

For each variable the Number of Address field and the ID code should be defined.

This procedure is only important when a particular access mode is needed or when the INI file does not contain the correct definition for the PLC.

Once the INI file has been properly configured from the “PLC Type” list box is available the complete list of the configured PLCs. They correspond to the INI sections called [PLCx].
The list box contains as PLC identifier the string typed in the INI file as `<plcname>`. As an example Figure 3 shows names for Allen-Bradley and Siemens.

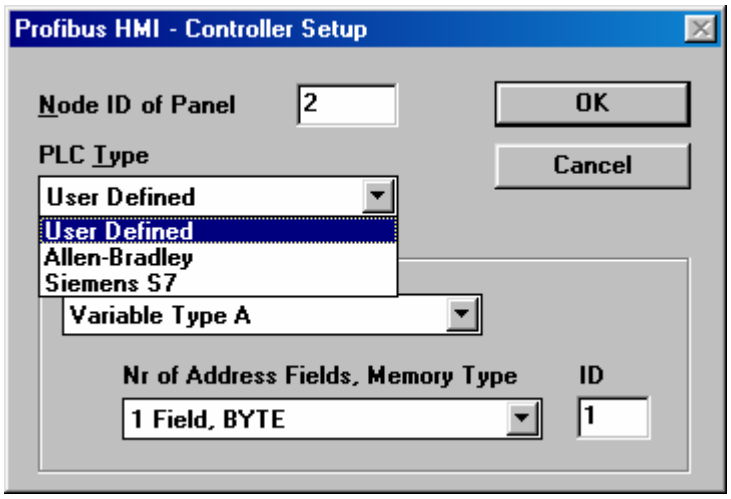

Figure 3

When the PLC Type is selected, the Variable Type Configuration area contains all the information from the INI file and no user modifications are required. Please note that "Nr. Of Address Fields" and "ID" fields are grayed-out as in Figure 4.

The last information that must be configured in the "Controller Setup" dialog box is the "Node ID of Panel" which correspond to the Profibus DP Slave node number.

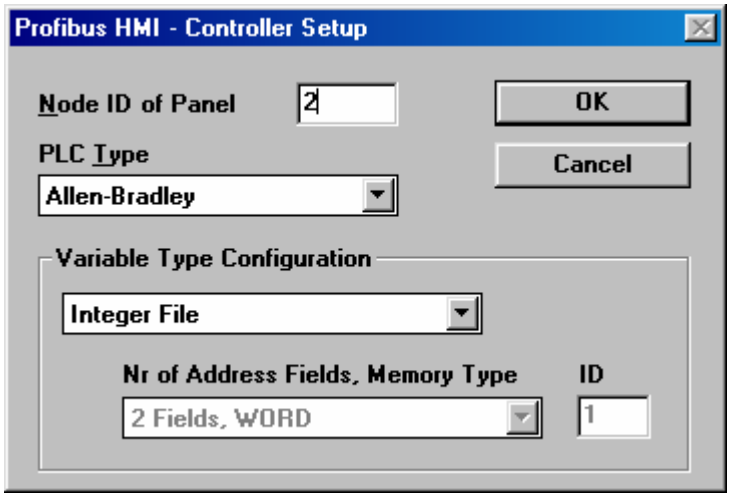

Figure 4

4. Connecting UniOP to Allen-Bradley Profibus DP Master

This chapter describes all the steps you have to follow in order to establish a successful connection between UniOP and Allen-Bradley Profibus DP Master System.
Functionality has been tested with the Profibus DP master module SST-PFB-SLC from SSTech, used with an Allen-Bradley SLC processor (SLC 5/03 or higher).
The Profibus DP I/O buffer of the HMI panel must be mapped in the M1/M0 area..

4.1. Configuring the Initialization File

The UPLC142.INI file sections are configured as described below:

```
[PLC1]
Name                = Allen-Bradley
DirectAccess        = 0
NumericFormat       = 1
ShowDataFormat      = 0
VarType1            = Integer File, N, 5, 1
```

This example will let you access data from the Integer Files (N).

4.2. Configuring Designer

Once the INI file has been correctly configured, the Designer software simply needs the selection of the “Allen-Bradley” PLC model in the “Controller Setup” dialog box.

The “Define Field” dialog box will appear as shown in Figure 5 where the “Sub Reference 1” field contains the File Number.
The field “Number” contains the word number in the selected file.

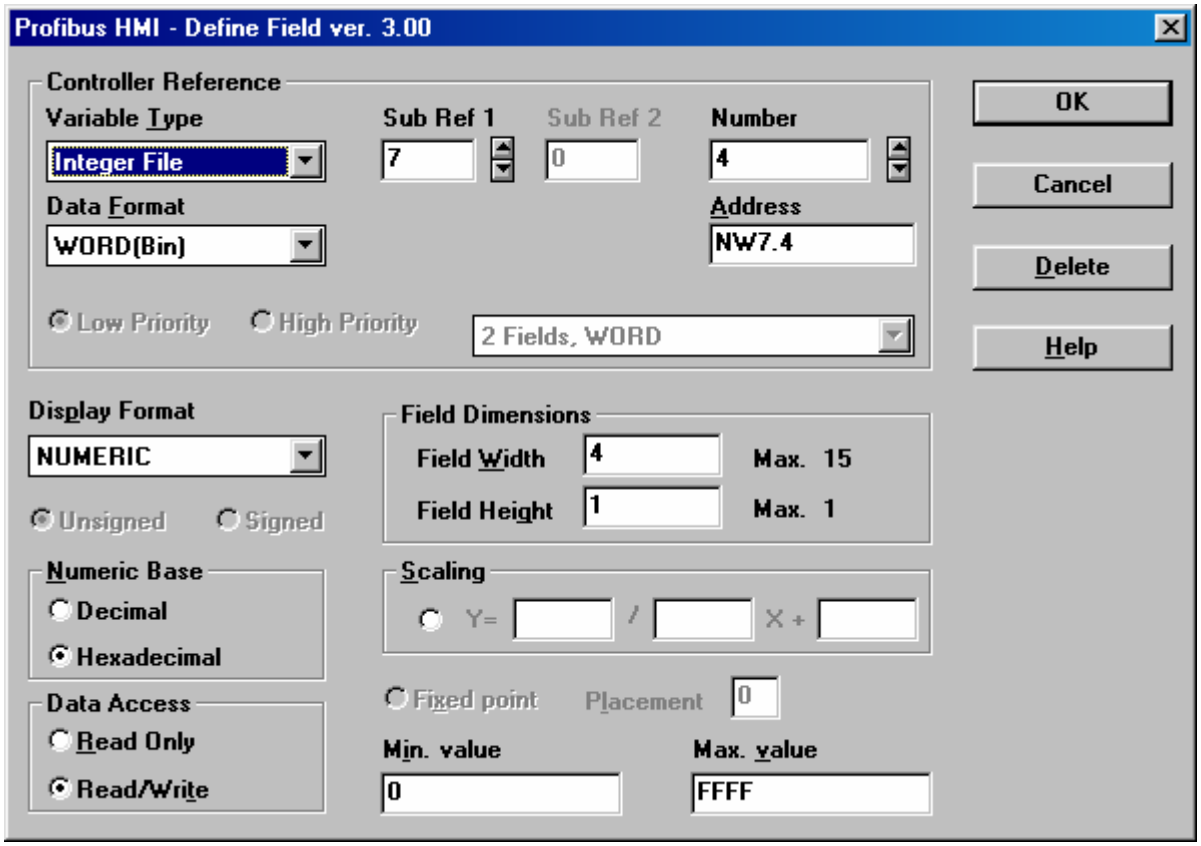

Figure 5

4.3. Using the PLC Support Program

The PLC support program for Allen-Bradley SLC 5/03 processor is available as a working example. This PLC Program support does not implement all of the functionality possible with the HMI Profile.

It is composed of only one ladder file (LAD 255 – DPSERVICES) which should be called at every scan cycle in the main ladder file of the user program as shown in Figure 6.

The information of the Profibus DP configuration of the HMI panels must be provided to the PLC support program using one integer file reserved for this purpose, which should be filled by the user only for the requested information (N255 – DPCONFIG).

Note: *The current version of the PLC Program is 01. It supports up to 32 connected panels and* ***only Integer File*** *access is allowed. No support is provided for Direct Bit Access and for Fast Bits. Generally speaking, the PLC program provided has to be considered as a working example and a good starting point to improve the interface capabilities.*

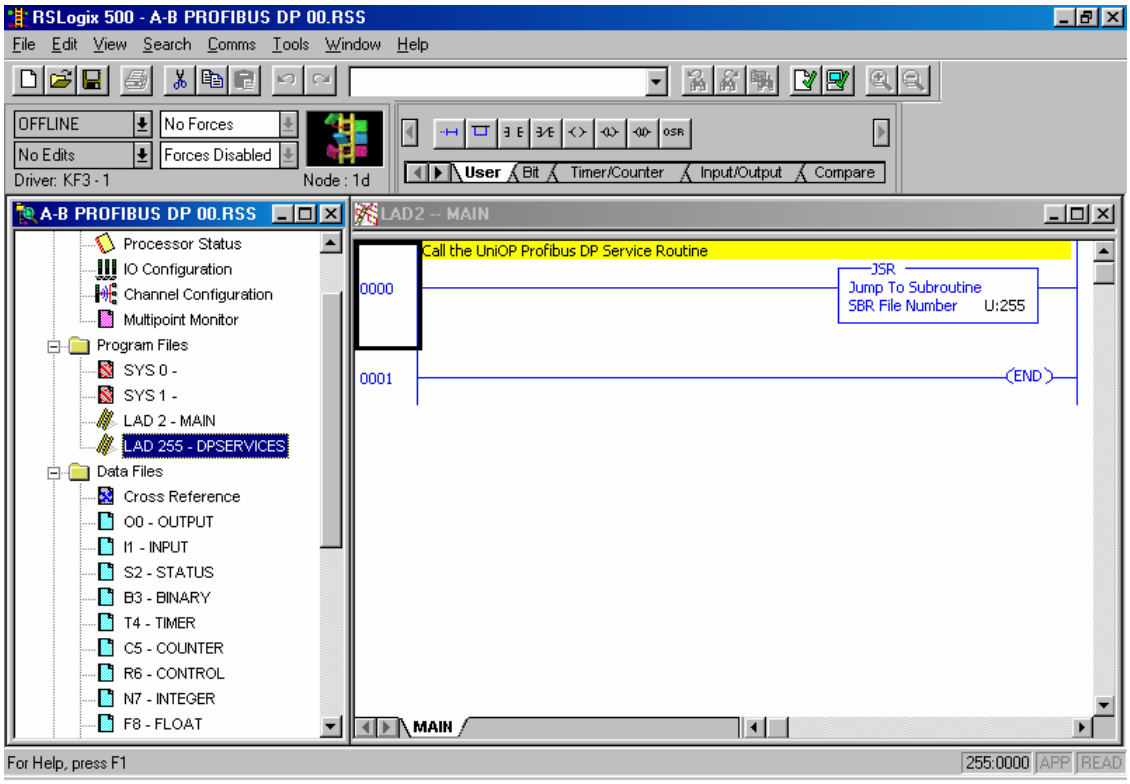

Figure 6

Table 12 shows only the required fields that have to be filled by the user in N255 configuration file. All the rest of the file must be considered as RESERVED.

Memory Location	Description
N255:0	Slot Number where the DP Scanner is inserted in the PLC rack
N255:1	Number of HMI panels connected
N255:2	Sequence Type
N255:100	M1 Offset for Input Buffer of the 1st panel
N255:101	M1 Offset for Input Buffer of the 2nd panel
N255:102	M1 Offset for Input Buffer of the 3rdt panel
…	….
N255:150	M0 Offset for Output Buffer of the 1st panel
N255:151	M0 Offset for Output Buffer of the 2nd panel
N255:152	M0 Offset for Output Buffer of the 3rd panel
…	…

Table 12

DeviceNet Scanner slot number DeviceNet Scanner Module slot number (1...32).

Number of connected UniOPs Number of HMI panels connected to Profibus DP (1...32). For each HMI panel, 2 words in memory are assigned for the definition of M0 and M1 offsets.

Sequence Type Specifies the way in which HMI panels are handled by the PLC support program. If Sequence Type is set to 0, the program will handle data exchange with one panel per call. When called again, it will handle data exchange with the next HMI panel in the list and so on, until all have been serviced, then it will restart from the first. If Sequence Type is set to 1, the PLC Support Program will handle data exchange with all HMI panels in the same cycle.
This means that if Sequence Type is set to 1, the requests from the panels will be processed faster but the execution time of the PLC program will be longer. If the increased execution time of the PLC program may cause problems for your application, you should set Sequence Type to 0.

Mx Offset Specifies the offset in words from the beginning of M0 and M1 Data Files, at which data for/from a particular HMI panel should be written/read. Data provided here must coincide with the offset specified for produced/consumed data in the Profibus DP configuration program used to program the master module. Numbering of the HMI panels used in the above table is arbitrary and it is not related to their address in the Profibus DP network.

5. Connecting UniOP to Simatic S7 Profibus DP Master Systems

This chapter describes all the steps you have to follow in order to establish a successful connection between UniOP and Simatic S7 Profibus DP Master systems.
Functionality has been tested with the Simatic S7 315-2DP CPU model.

Note: *The UniOP panels have to be configured in the Master controller using 32 bytes for the Input buffer and 32 bytes for the Output buffer. Different buffer sizes are not supported. All the 32 bytes of each buffer have to be configured in a way they result contiguous; the start address of the two buffers can be different.*

5.1. Configuring the Initialization File

The UPLC142.INI file section for Simatic S7 controllers is configured as described below:

```
[PLC2]
Name                = Siemens S7
DirectAccess        = 0
NumericFormat       = 1
ShowDataFormat      = 1
VarType1            = Data Block, D, 4, 1
VarType2            = Flag, M, 1, 2
DefModel            = 1
```

Note that both Data Types are byte native and the “Data Block” data type needs a “two levels” addressing mode.

5.2. Configuring Designer

Once the initialization file has been properly configured, the Designer define filed dialog box changes according to the specification given in the configuration file.
From “Project/Controller Setup” the Simatic S7 controller model has to be selected in case it is not the default model. The node number assigned to the UniOP panel in the Simatic Hardware configuration has to be entered in the “Node ID of Panel” box.

The Define Filed Dialog box appears like presented in the next Figure 7.

Simatic S7 controllers use an inverted byte order in word and double word data formats. The correct representation of these data types on the UniOP page is obtained using the WORD INV and the DOUBLE WORD INV data format.

5.3. Using the PLC Support Program

Communication between Simatic S7 controller and UniOP panel is handled by a PLC support program that has to be running on the PLC and called in the OB1 organization block.

Note: *The current version of the PLC Program is “A”. It supports access to **Merker** Area and **Data Block** area. No support is provided for Direct Bit Access and for Fast Bits. Generally speaking, the PLC program provided has to be considered as a working example and a good starting point to improve the interface capabilities.*

The PLC support program is made of one Function Block called FB9 in the available example and some instances Data Block; each panel connected to the Profibus DP bus needs one different instance Data Block.

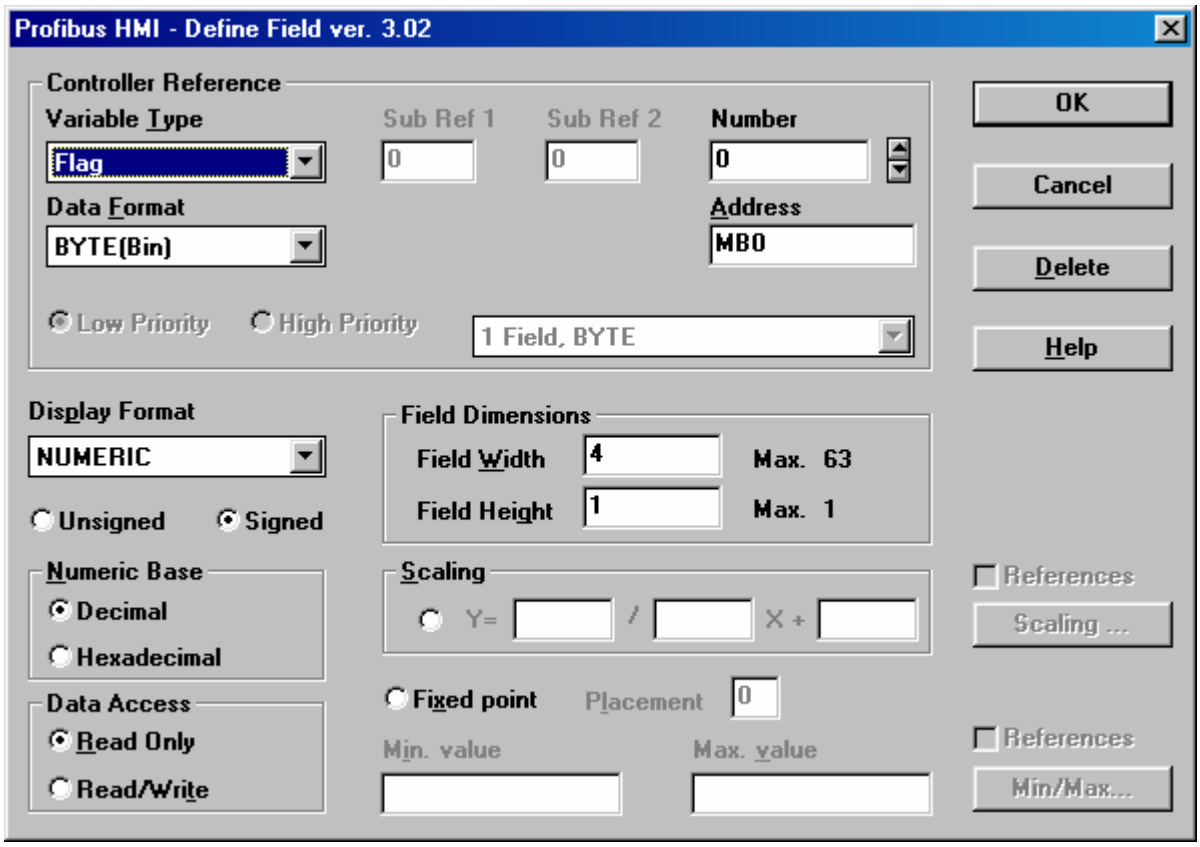

Figure 7

5.3.1. How to call the Function Block in the OB1 Block

The Function Block FB9 has to be called in the OB1 block, once for each connected panel.
The FB9 block has two input parameters: the offset address of the Profibus Input buffer assigned to the panel and the offset address of the Output buffer assigned to the panel in the bus. Addresses have to be expressed in exadecimal numeric base.

The FB9 syntax call is:

```
CALL   FB      9 , DB9
 INAddr :=W#16#20
 OUTAddr:=W#16#20
```

for a UniOP panel configured with 32+32 bytes starting from the Input address 32 and Output address 32.
The instance data blocks for the different calls to the FB9 function block, are automatically created by the Step7 software when the operator confirm the code line: `CALL  FB  9 , DB9.`
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 20 bytes.
A second panel can be inserted in the configuration simply making a new call to the FB9.
The example progrma is ready to run two UniOP panels.

Note: *the PLC support prtogram does not provide any error handling method concerning theProfibus DP parter missing situations.*

Appendix A. Communication Error Codes

Current communication status is displayed in the System Menu of the UniOP panel. Beside the string describing current state of the communication, there is an additional error code representing the last (which may be not the current one) error encountered. The codes are the following:

00	No error	
04	Error Bit is Set in the Response Status Byte from controller	Indicated that an error condition is flagged from the controller to the panel. This is probably due to wrong address used in the Designer project.
05	Low level protocol error	Indicates a Time Out in Data Packet or Status Request. The communication line has been broken or the PLC has had a power fail.
07	Internal failure	This should never happen. It indicates a hardware problem or software low level problem.
08	Communication Time Out	Indicates that the communication line has been broken or the PLC has had a power fail.
09	Failed to enter data exchange	Indicates that the UniOP failed to enter the data exchange state. This happen turning on the UniOP with PLC powered off. May also indicate that there are two or more slaves with the same node number.
12	Handshaking Error	Toggle Bit in the Response Control Block from the PLC is not different to toggle bit in request control byte.

Appendix B. Technical Data and Connection Information

The main technical information on the UniOP Profibus DP Slave interface using TCM08 is shown in the table below:

Baudrate	9.6 Kb to 12 Mb
Buffer size	8/16/32 bytes
Slave address	Software configurable
Optical insulation	Yes
Profibus Connector	Standard, 9 pin female sub-D type

The Aux Port is used for the Profibus DP communication. When the TCM08 communication module is mounted, the Aux Port becomes a standard Profibus connector. A simple point to point connection can be performed with the cable CA128.
In all other cases the usage of appropriate bus connectors such as the Siemens 6ES7972-0BA20-0XA0 and 6ES7972-0BB20- 0XA0 is recommended.

Appendix C. Requirements and Compatibility

This version of Profibus DP is included in the Designer DLL file UPLC142.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 TCM08 is required.
The UniOP panel must have hardware type –0045; firmware version number 4.10 or higher is required to support operation with the TCM08 modules.

Direct Access is currently not supported.