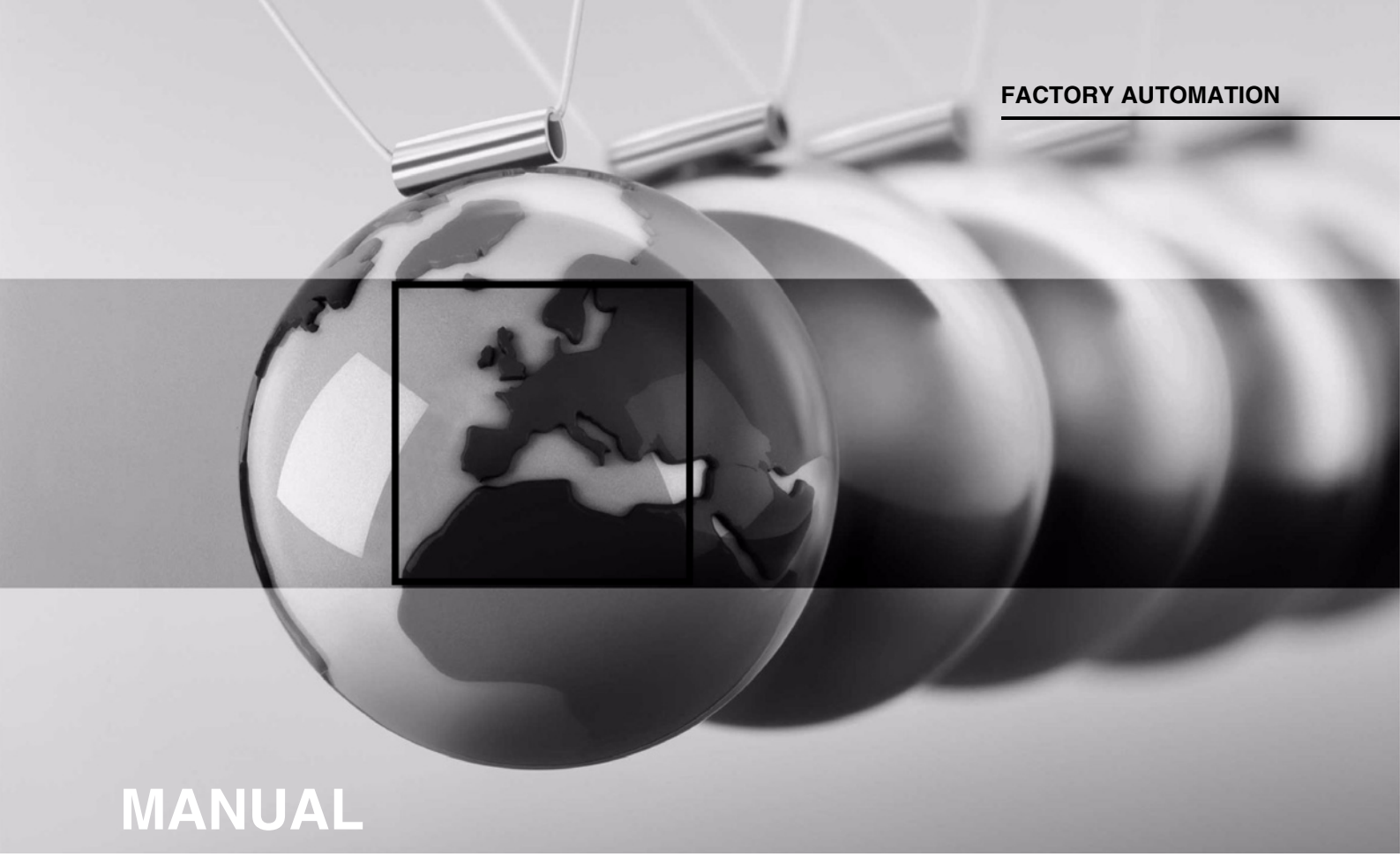


MANUAL

BUS COMMUNICATION

C-BOX 300

PEPPERL+FUCHS
SENSING YOUR NEEDS

With regard to the supply of products, the current issue of the following document is applicable: The General Terms of Delivery for Products and Services of the Electrical Industry, published by the Central Association of the Electrical Industry (Zentralverband Elektrotechnik und Elektroindustrie (ZVEI) e.V.) in its most recent version as well as the supplementary clause: "Expanded reservation of proprietorship"

Table of Contents

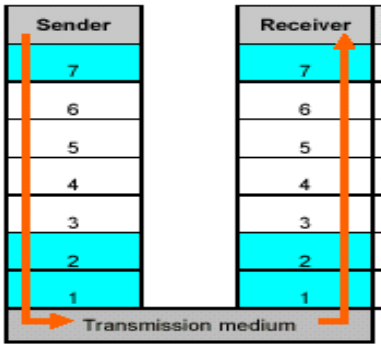
1. Overview

1.1 Description

Profibus is the world's most popular FieldBus.

Profibus is the most widely accepted international networking standard. Nearly universal in Europe and also popular in North America, South America and parts of Africa and Asia. Profibus can handle large amounts of data at high speed and serve the needs of the majority of automation applications.

Profibus was created under German Government leadership in co-operation with automation manufacturers (Siemens) in 1989. Today it is commonly found in Process Control, large assembly and material handling machines. Just a single-cable which is able to wire multi-input sensor blocks, pneumatic valves, complex intelligent devices, smaller sub-networks, operator interfaces and many other devices.



Designation and function of the layers	
Application layer	Interface to application program with application-oriented commands (read, write)
Presentation layer	Representation (coding) of data for analysis and interpretation in the next layer
Session layer	Establishing and clearing temporary station connections; synchronization of communicating processes
Transport layer	Controlling data transmission for layer 5 (transport errors, break down into packets)
Network layer	Establishing and clearing connections, avoiding network congestion
Data-link layer	Description of bus access protocol (Medium Access Control, MAC) including data security
Physical layer	Definition of the medium (hardware), coding and speed of the data transmission

The ISO/OSI reference model describes communications between the stations of a communication system: if a communication system does not require some specific functions, the corresponding layers have no purpose and are bypassed. Profibus uses layers 1, 2 and 7.

1.2 Profibus-DP

Basically Profibus is available in different versions:

- **Profibus-DP** (Decentralized Periphery)
  Multiple masters are possible with Profibus-DP, in which case each slave device is assigned to one master. This means that multiple masters can read inputs from the device but only one master can write outputs to that device.

- Profibus-FMS
  It is a peer to peer messaging format, which allows masters to communicate with one another. Just as in Profibus-DP, up to 126 nodes are available and all can be masters if desired. FMS messages consume more overhead than DP messages.

- Profibus-PA
  PA protocol is the same as the latest Profibus-DP except that voltage and current levels are reduced to meet the requirements of intrinsic safety (Class I div. II) for the process industry.

Pepperl+Fuchs supports **Profibus-DP only**, since this version has been specifically designed for factory automation. System version must be of this type.

Main features:

- Maximum Number of Nodes: 126
- Distance: 100m to 24 Km (with repeaters and fibre optic transmission)
- Baud rate: 9600 to 12M bps
- Messaging formats: Polling, Peer-to-Peer

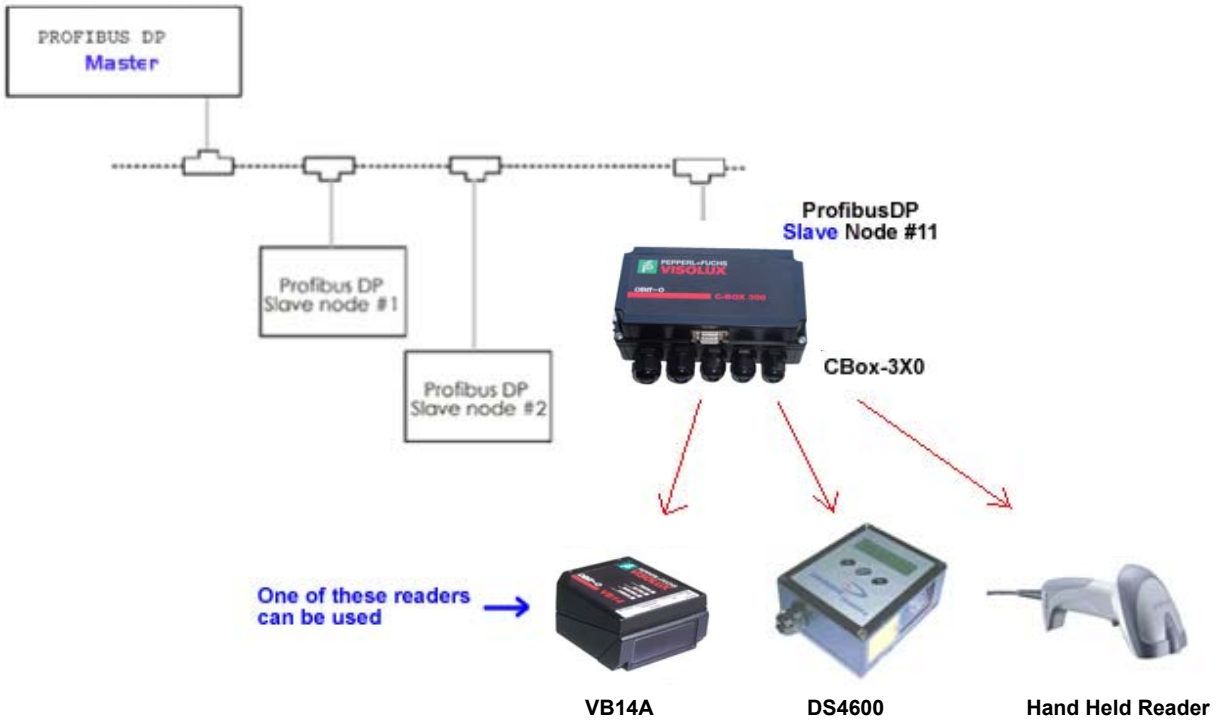
From here on we will refer to Profibus-DP only.

2. Bus Communication

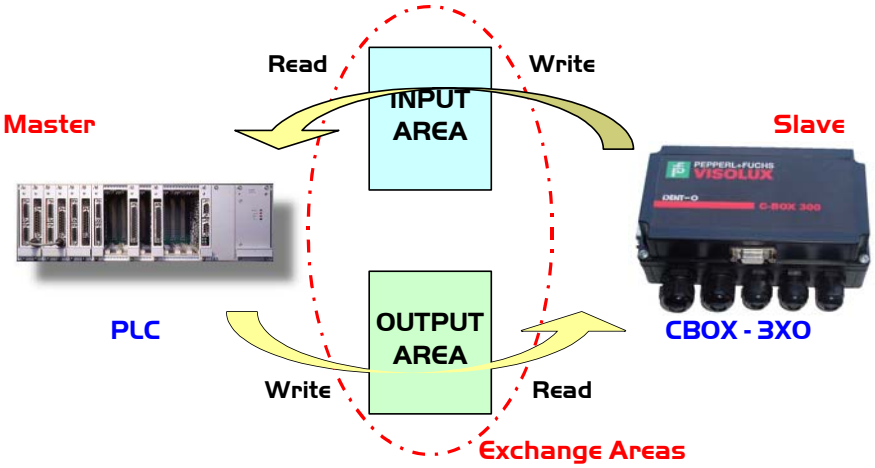
2.1 Data Exchange

Master Profibus is usually a PLC (Siemens S7 or others). Sometimes it could be a PC-based device as well.

CBox-300 device is always ***Slave*** in the Profibus network.



Basically two shared memory areas (Exchange Areas) exist between SLAVE and MASTER so both devices provide information to each other. Exchange areas are physically placed on the CBox-300 Profibus chip.



Input and output areas always refer to the Master: this means that the scanner writes to the Input buffer and the PLC writes to the Output buffer.

Dimensions of exchange areas can be set to different values by the PLC through the GSD file: the CBox-300 allow up to **32 bytes for the Input Area** and **32 bytes for the Output Area**.

2.2 VisoSetup Settings

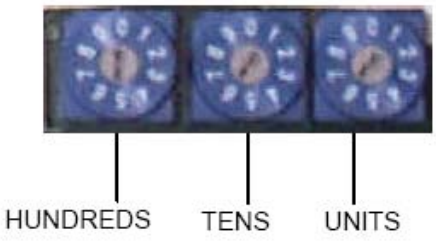
The VisoSetup SW Configuration Tool allows parameterisation of Profibus communication.

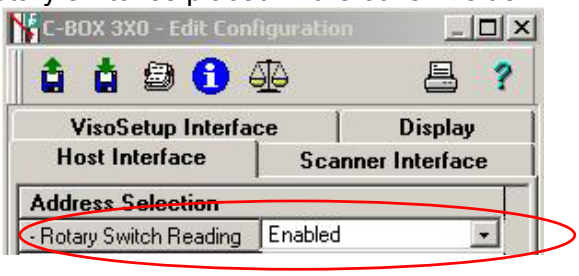
2.2.1 Bus Communication Parameters

- **Profibus Address**

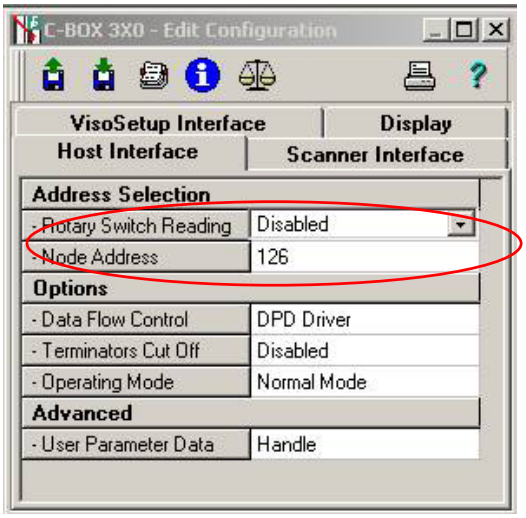
The Slave is identified over the Profibus network by setting its own address. This value, which is unique for each slave, should match the address specified while configuring your network, so that the PLC is able to detect the Slave properly.

The Node Address should be set using the rotary switches placed in the cover inside:





The valid address range is from **000** to **126**. If an invalid value is detected, the CBox-300 cannot communicate with the Profibus network. The Node Address can also be assigned through VisoSetup:



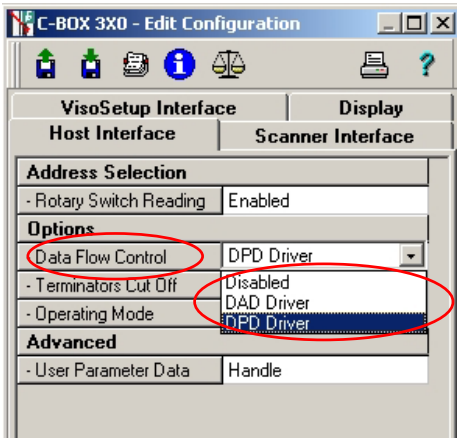
- **Profibus Baudrate**
- **Master Input Area Size / Master Output Area Size**

Network baurate is selected within PLC environment while configuring your network. CBox-300 will automatically learn the correct communication speed at run time.

Master Input Area Size and *Master Output Data Size* are forced by PLC as well.

- **Data Flow Control**

Data Flow Control enables a powerful way to manage and optimise communication with the Profibus Master. A dedicated program running on the PLC is required to take advantage of this feature (see details in the "Flow Control Mode (FCM)" paragraph).

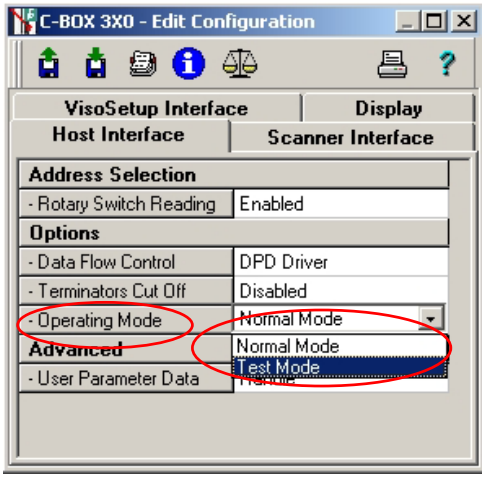


Final implementation should make use Flow Control to obtain maximum reliability and optimal synchronisation between Master and Slave. As a first approach, *Data Flow Control* = *Disable* is the suggested way to check the HW/SW configuration of the Profibus network.

- **Operating Mode**

During the startup phase of the CBox-300 installation, it is fundamental to check Profibus communication first. *Operating Mode* = *Test Mode* is the way to make CBox-300 send sequential messages to the PLC without need of additional reader connected. It is useful just to check if something is received by Master Profibus.

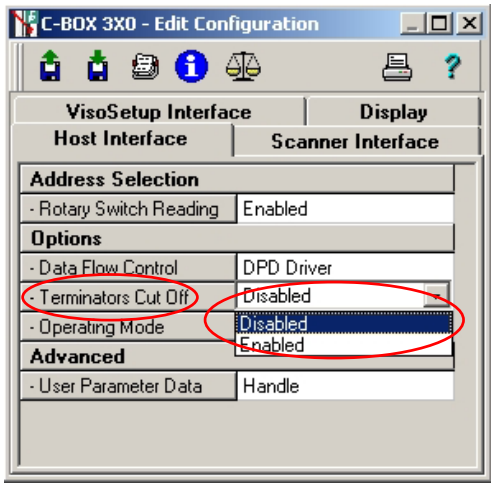
When *Operating Mode* = *Normal Mode* CBox-300 simply forwards messages from reader to PLC.



- **Terminators Cut Off**

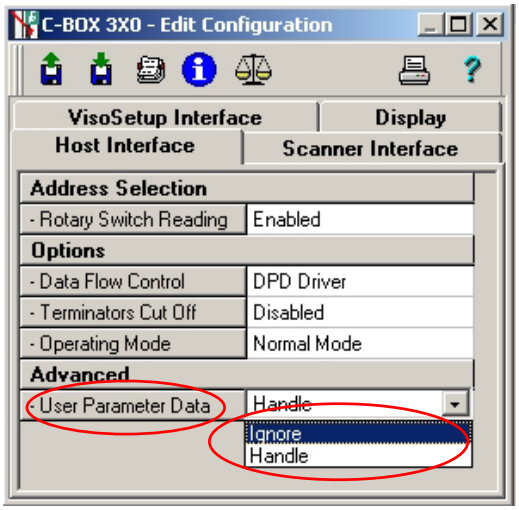
Terminator characters are needed to let CBox-300 understand when each message finishes so it can forward strings correctly to Profibus network.

If necessary, *Terminators Cut Off* = *Enabled* can avoid terminators transmission.



- **User Parameter Data**

Starting from sw v.2.00 and higher, scanner programming from PLC is allowed. The implementation takes advantage of Profibus modules within PLC environment (refer to par.3.3 *Scanner Programming via GSD File* for details). As default *User Parameter Data* = *Handle* supports the above functionality whereas *User Parameter Data* = *Ignore* does not consider any possible GSD module added to the network configuration.



2.3 Flow Control Mode (FCM)

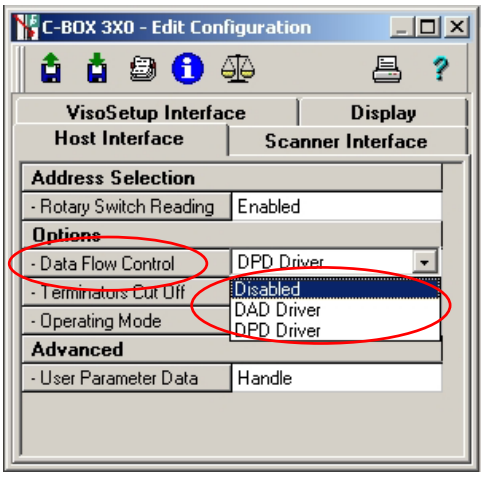
The Flow Control Mode is a powerful way to manage and optimise the communication with the Profibus Master. By enabling the FCM a few bytes of the exchange areas are reserved for driver operations and the rest are used by the application layer.

The reserved bytes are used to implement many different features such as:

- Flow-control and corresponding buffering in both directions
- Fragmentation and reassembling of data longer than the exchange area sizes
- Synchronisation of flow control numbers
- Service Access Point oriented communication
- Length information

Note: If the Flow Control is disabled, all the bytes of the exchange areas are used by the application layer. The input area is updated whenever a new reading event has to be transferred to the Master station.
→ In this situation, the Master must read the input area before it changes due to a new message, typically new barcode occurrence.
→ Moreover, two occurrences of the same barcode cannot be understood, since the input area does not change.
→ In addition, if application data is longer than input area sizes, data is automatically truncated.

FCM can be selected by means of the Data Flow Control parameter.



Basically three options are available:

- ❒ DAD Driver → *FCM compatible with "Flow Control = Anybus" used in new devices*
- ❒ DPD Driver → *FCM compatible with "Flow Control = Profibus" used in Multiplexer*
- ❒ Disable → *No Flow Control*

Although both drivers implement the same above features, two options are currently available to obtain the maximum compatibility towards different Pepperl+Fuchs family devices.

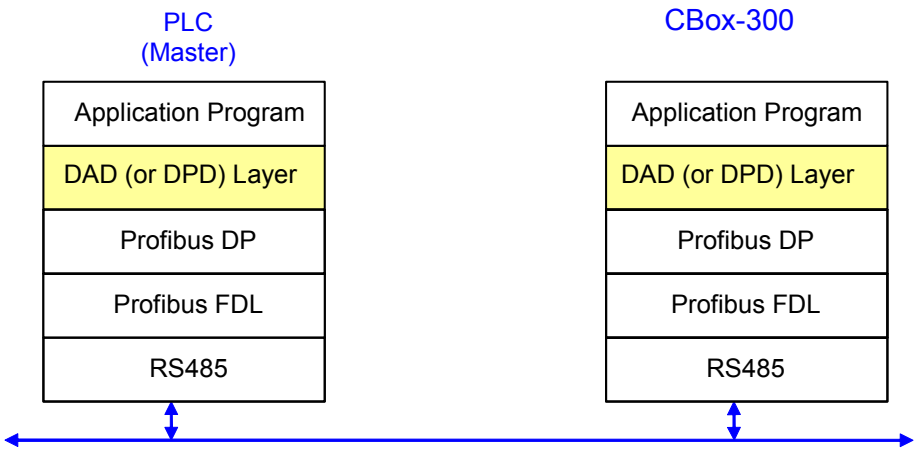
See the cross reference table below about supported FCM:

CBOX Family		**DAD Driver**	**DPD Driver**
CBox-300 Profibus-DP		X	X
VB3x Family		**DAD Driver**	**DPD Driver**
VB33-2000-P		X	X
VB33-2000-OM-P		X	X
VB34-2500-P		X	X
VB34-2500-OM-P		X	X
MX Family			
MX4000-1100			X
MX4000-1100 SB2498 12MB/s		X	

2.3.1 Flow Control drivers

The Flow Control driver is a layer that is built upon the intrinsic DP data exchange mechanism. Basically such a layer is required because the intrinsic DP Profibus mechanism is not message oriented.

In the following figure the complete Stack is represented:



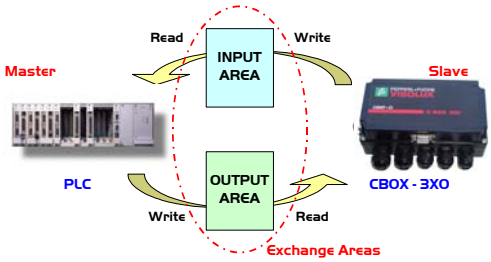
Two Flow Control drivers are available:

- DAD Driver → Compatible with *Flow Control = Anybus*
- DPD Driver → Compatible with *Flow Control = Profibus*

The DAD driver should be the preferred solution for brand new installations.

2.3.2 FCM with DAD Driver

In order to implement the flow controlled version of the driver, exchange areas must be congruently compiled in both directions.



From now on we refer to the Input Area as a buffer made up of InputAreaSize bytes:

IN[0]	IN[1]	IN[2]	…	IN[InputAreaSize - 1]

and to the Output Area as a buffer made up of OutputAreaSize bytes:

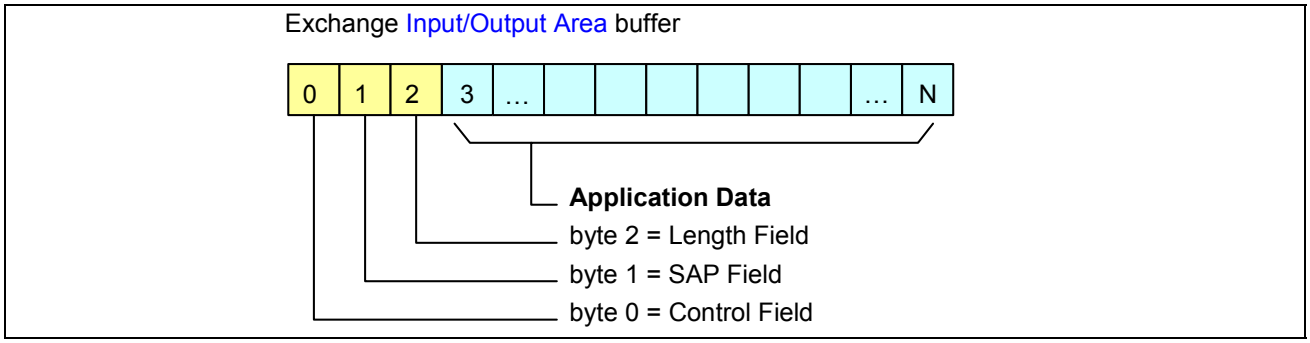
OUT[0]	OUT[1]	OUT[2]	…	OUT[OutputAreaSize - 1]

Only the first three bytes are used by the DAD Driver layer in both buffers:

Control Field (byte 0) — used to issue and control the driver primitives such as flow-control, fragmentation and resynchronisation.

Service Access Point Field (byte 1) — used to distinguish among different services and to provide future expandability. (Since this SAP definition is introduced by the DAD Driver, it must not be confused with the SAP that is defined by the international standard).

Length Field (byte 2) — contains the number of bytes used by the application layer:
Length Field ≤ (InputAreaSize – 3) for the Input Area
Length Field ≤ (OutputAreaSize – 3) for the Output Area.



The Application Data buffer holds useful information, typically the barcode messages, processed by the application program. IN[3] contains the first significant byte of the Application Data buffer (the same first byte you would see if VB14A transmitted the barcode buffer onto the Auxiliary port instead of the Profibus interface).

The structure of the application buffer and its length strictly depend on the selected data format on the VB14A. Barcode messages longer than (InputAreaSize – 3) will be split in pieces through an automatic fragmentation process (see details in the "Fragmentation and Reassembling" paragraph).

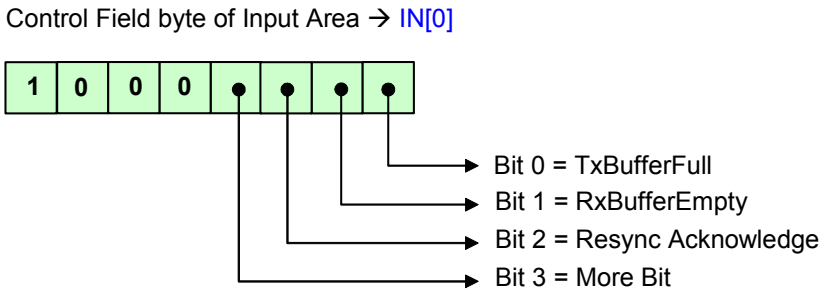
2.3.2.1 Control Field

This is the core of the flow controlled communication.

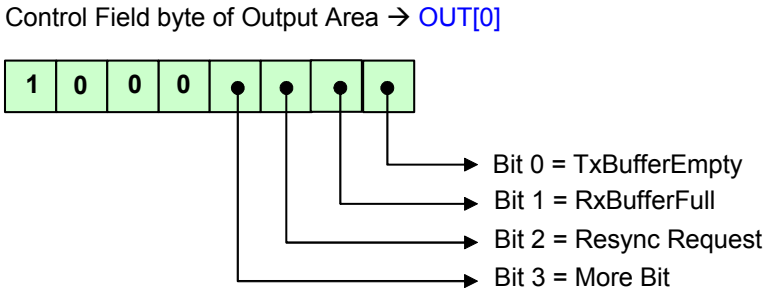
The Input Area structure reserves bit 0 and bit 1 of IN[0] for handshake purposes while the Output Area structure, which is symmetrical, reserves bit 0 and 1 of OUT[0].

At any time the Master station can make a resynchronization request by means of bit 2 of the Output Area. This process, which resets the synchronization numbers (bit 0 and bit 1 of both Input and Output areas), has to be acknowledged by the Slave on bit 2 of the Input Area.

Bit 3 is used to control a fragmentation sequence in both directions.



IN[0].bit0	**TxBufferFull** toggles when Slave has made available new Input Area data
IN[0].bit1	**RxBufferEmpty** toggles when Output Area data has been read by Slave
IN[0].bit2	**Resync Acknowledge** set to 1 as an acknowledge to a resync request. With this bit, the master can detect a slave is on line.
IN[0].bit3	**More Bit** is 1 when this is not the last piece of a fragmentation sequence while it is 0 when this is the last piece
IN[0].bit4,5,6,7	Set to **0**, **0**, **0**, **1** when DAD messaging protocol is used



OUT[0].bit0	**TxBufferEmpty** must toggle when Input Area data has been read by Master
OUT[0].bit1	**RxBufferFull** must toggle when Master makes available new Output Area data
OUT[0].bit2	**Resync Request** set to 1 for one second to resynchronize the slave. After resynchronization, all 4 handshake bits are set to 0
OUT[0].bit3	**More Bit** must be 1 when this is not the last piece of a fragmentation sequence and it must be 0 when this is the last
OUT[0].bit4,5,6,7	Set to **0**, **0**, **0**, **1** when DAD messaging protocol is used

2.3.2.2 SAP Field

SAP (Service Access Point) is an identifier that is used to implement multiple services sharing the same communication channel between two remote stations.

The following values have been defined:

- SAP = 0 Used to transfer information messages between CBox-300 and PLC
- SAP = 255 Reserved for driver services (see details in the "SAP Services" paragraph)

All other SAP values are free and they could be used by dedicated application programs after agreement between the application programs themselves.

2.3.2.3 Length Field

The Application layer uses all or a part of the remaining bytes of the Exchange Area buffers that are not used by the DAD Driver. The Length Field is introduced to keep the information of how many bytes are really used by the Application Layer.

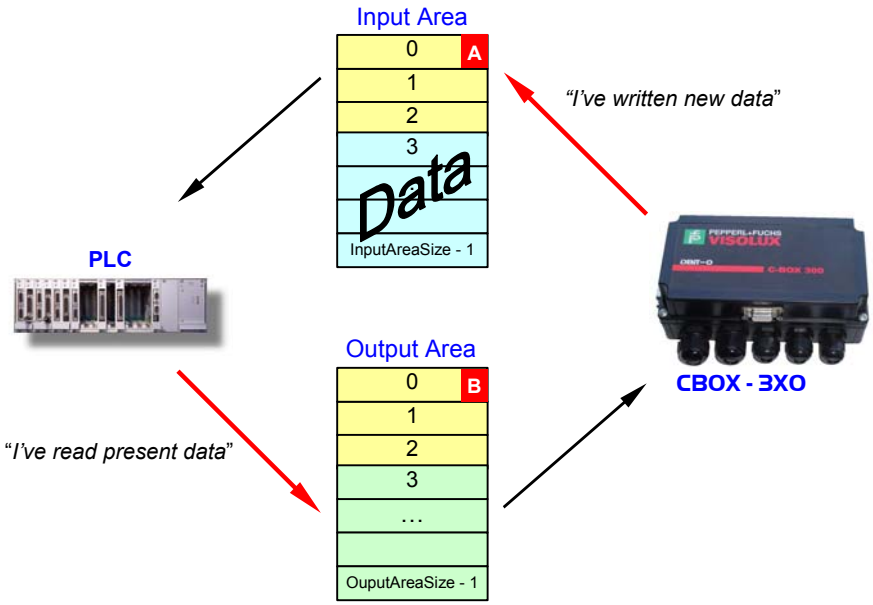
A fragment that is not the last one of a fragmentation sequence must fill this field with [InputAreaSize – 3] (or [OutputAreaSize - 3]), depending on whether it is an Input/Output fragment. Otherwise this field gets a value that is less than or equal to [InputAreaSize – 3].

2.3.2.4 Data Transmission from CBOX-300 to PLC

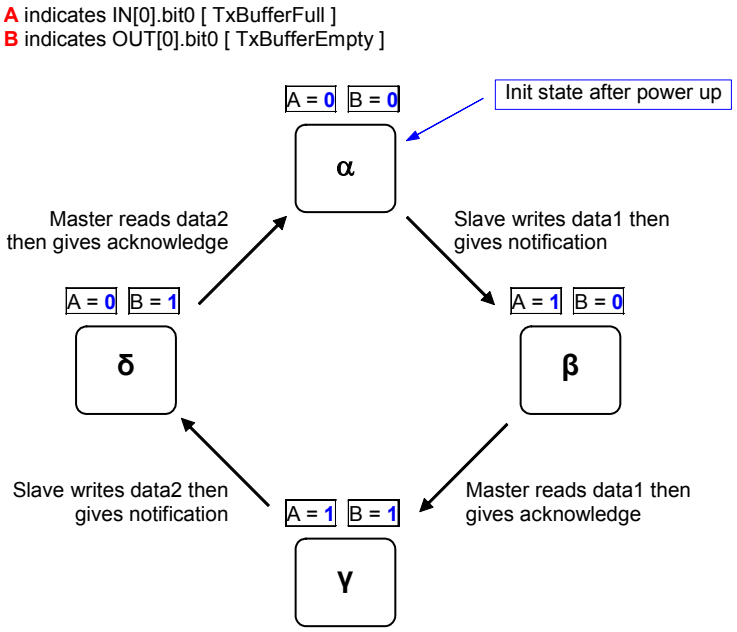
This paragraph describes how it is possible to exchange messages with flow control. The communication mechanism is simple:

- **IN[0].bit0** [A] is used by CBox-300 to notify that "*Slave has written a new data so Master can read it*"
- **OUT[0].bit0** [B] must be used by PLC to notify that "*Master has read last data so Slave can send next message*"

This happens each time bit A (or B) changes its state (toggles). Bit level doesn't matter, only the transition has to be considered.



The following state machine shows data transmission from Slave to Master. Please note that each cycle transfers two data messages.



Let's analyse a typical data exchange based on the following settings:

- Flow Control = **DAD Driver**
- Input Area Size = **16**
- Output Area Size = **8**

- After power up Input and Output areas are generally filled by zero. According to DAD driver implementation, Input area has Control Field = 80_{Hex} and SAP = 00_{Hex}.

Input Area	**80**$_{Hex}$ **00**$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$
Output Area	00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$

- Also PLC must set the Control Field of Output area properly, as long as DAD messaging protocol is utilised.

Input Area	80$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$
Output Area	**80**$_{Hex}$ **00**$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$

- VB14A reads a barcode "123456". Let's assume standard data formatting with <STX> as header and <CR><LF> as terminators. C-Box toggles bit **A**.

Input Area	**81**$_{Hex}$ 00$_{Hex}$ **09**$_{Hex}$ **02**$_{Hex}$ **31**$_{Hex}$ **32**$_{Hex}$ **33**$_{Hex}$ **34**$_{Hex}$ **35**$_{Hex}$ **36**$_{Hex}$ **0D**$_{Hex}$ **0A**$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$
Output Area	80$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$

- PLC detects transition of bit **A** so now it can read incoming data (it copies 9 bytes in its memory from IN[3] on) then toggles bit **B** as acknowledge.
  ***Note**: before the acknowledge, all further barcodes read by VB14A are buffered.*

Input Area	81$_{Hex}$ 00$_{Hex}$ 09$_{Hex}$ 02$_{Hex}$ 31$_{Hex}$ 32$_{Hex}$ 33$_{Hex}$ 34$_{Hex}$ 35$_{Hex}$ 36$_{Hex}$ 0D$_{Hex}$ 0A$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$
Output Area	**81**$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$

- VB14A reads a barcode "10DL" and toggles bit **A**.

Input Area	**80**$_{Hex}$ 00$_{Hex}$ **07**$_{Hex}$ **02**$_{Hex}$ **31**$_{Hex}$ **30**$_{Hex}$ **44**$_{Hex}$ **4C**$_{Hex}$ **0D**$_{Hex}$ **0A**$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$
Output	81$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$ 00$_{Hex}$

Area	

- PLC reads new data message (it copies 7 bytes in its memory from IN[3] on) then toggles bit **B** as acknowledge.

Input Area	80_{Hex}	00_{Hex}	07_{Hex}	02_{Hex}	31_{Hex}	30_{Hex}	44_{Hex}	$4C_{Hex}$	$0D_{Hex}$	$0A_{Hex}$	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}
Output Area	**80_{Hex}**	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}							

- VB14A performs *No Read* and toggles bit **A**. Let's assume <CAN> as *Global No Read* character.

Input Area	**81_{Hex}**	00_{Hex}	**04_{Hex}**	**02_{Hex}**	**18_{Hex}**	**$0D_{Hex}$**	**$0A_{Hex}$**	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}
Output Area	80_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}							

- PLC reads new data message (it copies 4 bytes in its memory from IN[3] on) then toggles bit **B** as acknowledge.

Input Area	81_{Hex}	00_{Hex}	04_{Hex}	02_{Hex}	18_{Hex}	$0D_{Hex}$	$0A_{Hex}$	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}
Output Area	**81_{Hex}**	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}							

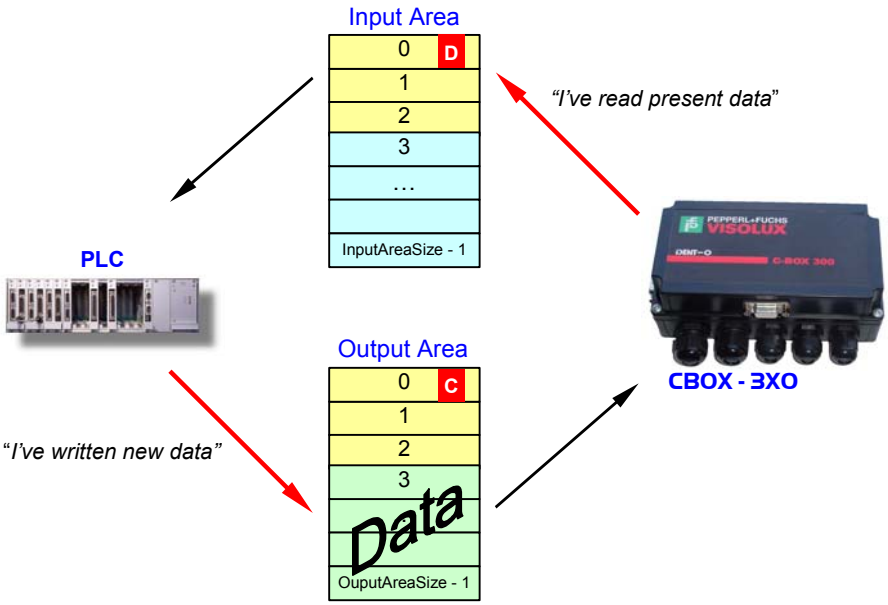
- Data exchange continues…

2.3.2.5 Data Transmission from PLC to CBOX-300

Analogous to the previous paragraph, flow control works even when data are coming from the Master towards the Slave. The communication mechanism is based on the same concepts:

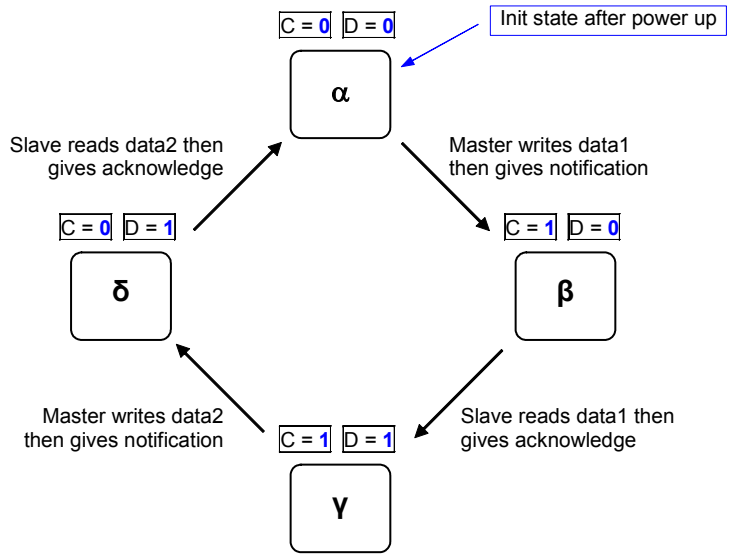
- **OUT[0].bit1** [C] must be used by PLC to notify that "*Master has written new data so Slave can read it*"
- **IN[0].bit1** [D] is used by CBox-300 to notify that "*Slave has read data so Master can send next message*"

This happens each time bit C (or D) changes its state (toggles). Bit level doesn't matter, only the transition has to be considered.



The following state machine shows data transmission from Master to Slave. Please note that each cycle transfers two data messages.

C indicates OUT[0].bit1 [RxBufferFull]
D indicates IN[0].bit1 [RxBufferEmpty]

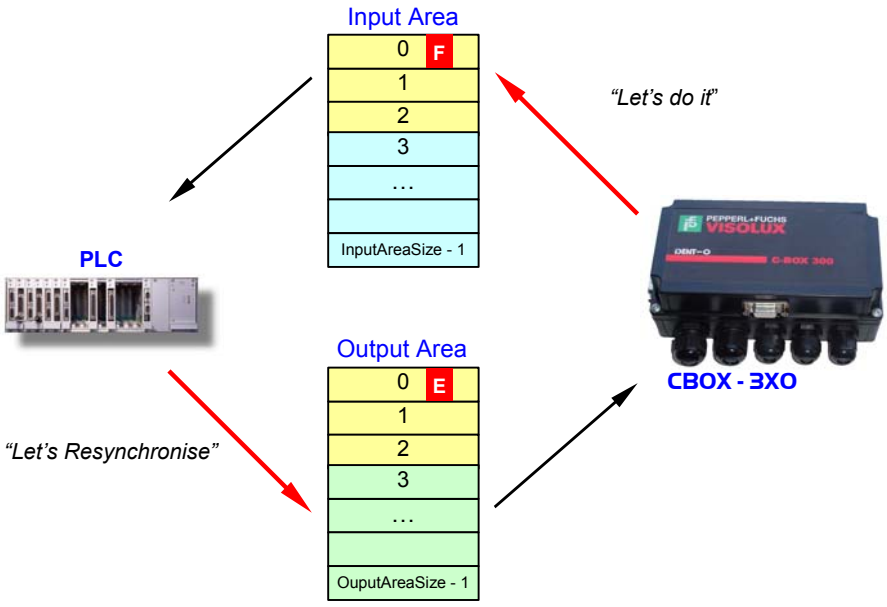


2.3.2.6 Resynchronisation

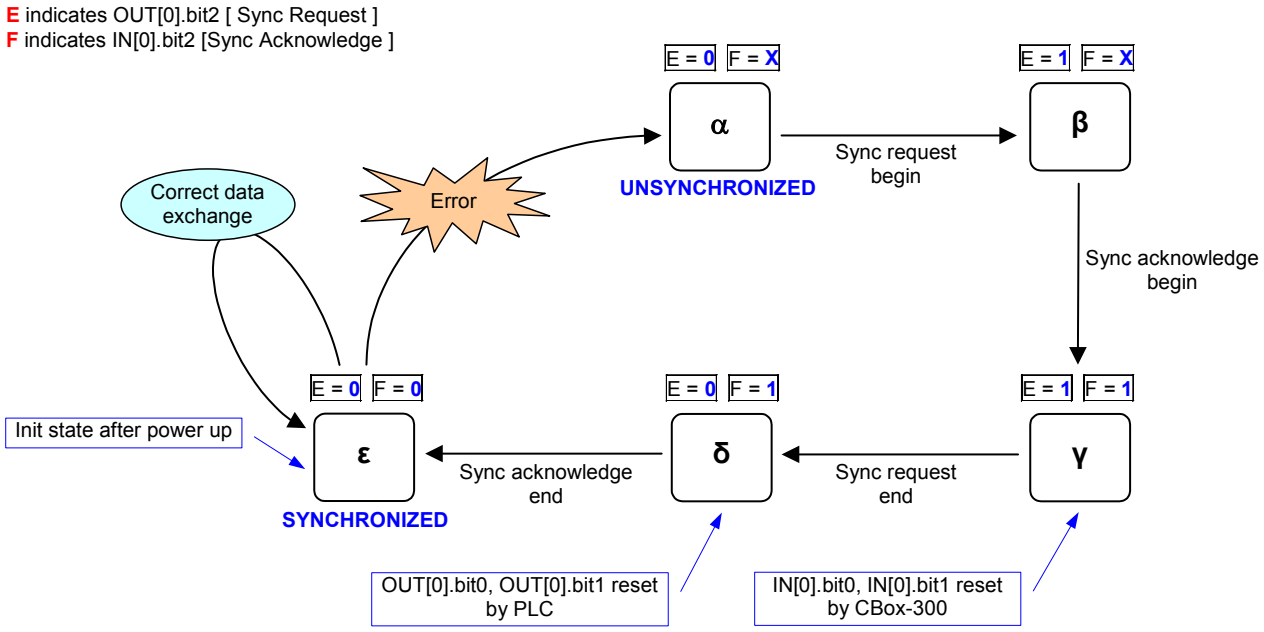
The resynchronisation process restarts the messaging protocol from a predefined state.
It may be used either at the Master startup to detect if the Slave is on line or during normal operations in case of errors requiring a protocol reset procedure.

The process is based on bit two:

- **OUT[0].bit2** [E] must be used by PLC to request the Resynchronisation
- **IN[0].bit2** [F] is used by CBox-300 to acknowledge the request



The following state machine shows the resynchronisation cycle, requested by the PLC and performed together with CBox-300:



Let's analyse the resynchronisation process, starting from the previous data exchange discussed in "Data Transmission from CBOX-300 to PLC"…

Input Area	81_{Hex}	00_{Hex}	04_{Hex}	02_{Hex}	18_{Hex}	$0D_{Hex}$	$0A_{Hex}$	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}
Output Area	81_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}							

- PLC requests resynchonisation by setting bit E = 1.

Input Area	81_{Hex}	00_{Hex}	04_{Hex}	02_{Hex}	18_{Hex}	$0D_{Hex}$	$0A_{Hex}$	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}
Output Area	**85_{Hex}**	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}							

- CBox-300 detects the request and so it resets **IN[0].bit0** and **IN[0].bit1**. Then it gives an acknowledge back to the PLC by means of bit F.

Input Area	**84_{Hex}**	00_{Hex}	04_{Hex}	02_{Hex}	18_{Hex}	$0D_{Hex}$	$0A_{Hex}$	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}
Output Area	85_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}							

- PLC has to reset **OUT[0].bit0** and **OUT[0].bit1** before completing its request with bit E = 0.

Input Area	84_{Hex}	00_{Hex}	04_{Hex}	02_{Hex}	18_{Hex}	$0D_{Hex}$	$0A_{Hex}$	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}
Output Area	**80_{Hex}**	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}							

- CBox-300 completes the acknowledge process by setting bit F = 0.

Input Area	**80_{Hex}**	00_{Hex}	04_{Hex}	02_{Hex}	18_{Hex}	$0D_{Hex}$	$0A_{Hex}$	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}
Output Area	80_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}							

- Now Flow Control has been returned to a predefined state. All data exchange bits in the Control Field are surely zero and data transmission can proceed safely.

2.3.2.7 Fragmentation and Reassembling

The fragmentation process is activated whenever Application Data cannot be contained in the related exchange area. Basically long messages are split into pieces which are transmitted separately. Reassembling allows the reconstruction of the whole messages.

CBox-300 already implements these functions in the DAD layer, while the PLC needs a congruent management.

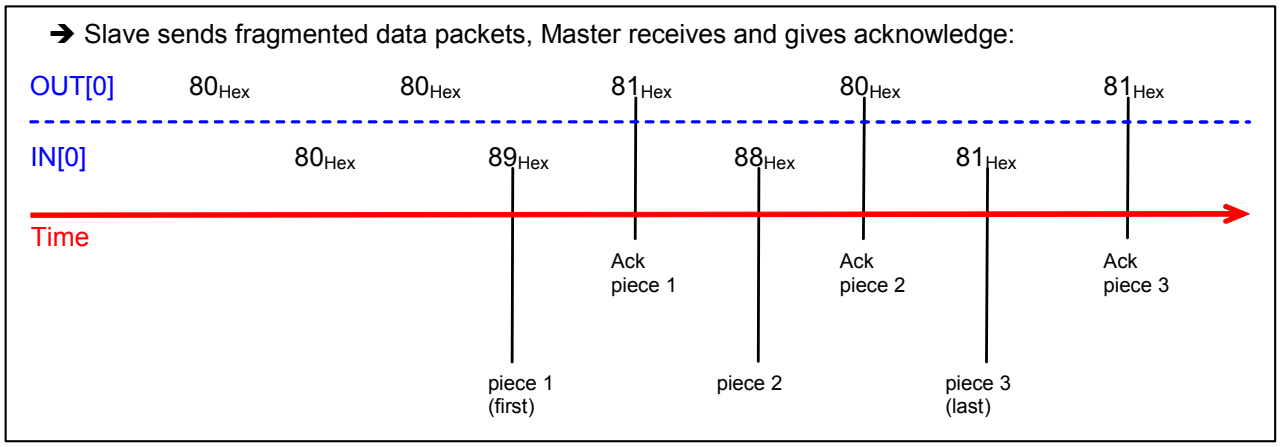
The fragmentation is based on the **More Bit** (bit 3) in the Control Field byte.

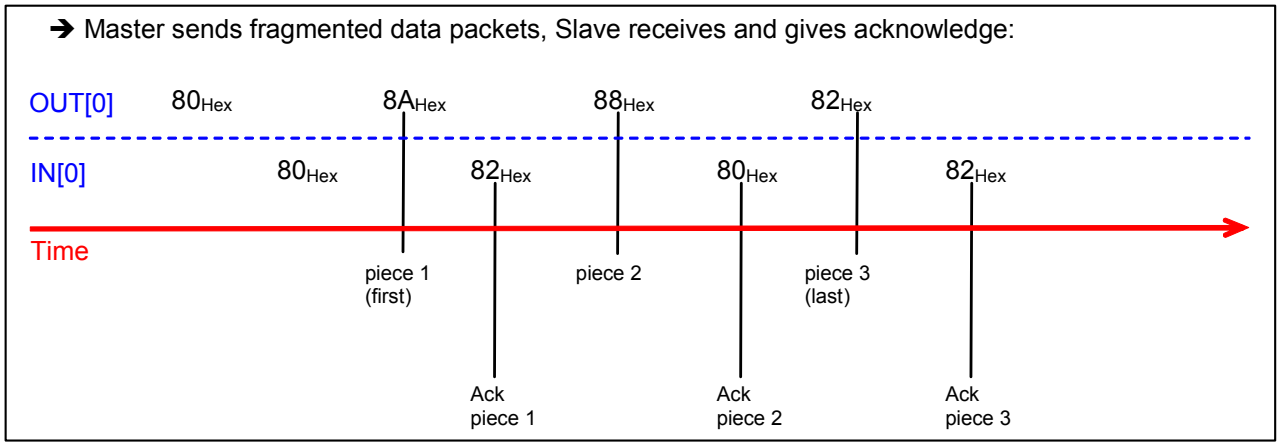
More Bit = 0 indicates that all the information is included within the current message. When Application Data is longer than (exchange area size – 3), the first partial message is transmitted having More Bit = 1. Following fragments keep More Bit = 1 and only the last piece will have More Bit = 0 again. Thanks to this mechanism, the receiver station may detect the last piece and so reassemble the entire information.

Some notes:

- CBox-300 can manage application messages up to 256 bytes
- Intermediate fragments have Length Field = (exchange area size – 3)
- Last fragment has Length Field ≤ (exchange area size – 3)
- Bit0 and bit1 of both Input and Output areas are independently managed for any fragment

The following figures show how the Control Byte changes according to the fragmentation process. Both data flow directions are considered.





Let's analyse a fragmented data exchange based on the following settings:

- Flow Control = **DAD Driver**
- Input Area Size = **16**
- Output Area Size = **8**

- After power up Input and Output areas are generally filled by zero. According to DAD driver implementation, Input area has Control Field = 80_{Hex} and SAP = 00_{Hex}.

Input Area	**80**Hex **00**Hex 00Hex 00Hex 00Hex 00Hex 00Hex 00Hex 00Hex 00Hex 00Hex 00Hex 00Hex 00Hex 00Hex 00Hex
Output Area	00Hex 00Hex 00Hex 00Hex 00Hex 00Hex 00Hex 00Hex

- Also PLC must set the Control Field of Output area properly, as long as DAD messaging protocol is utilised.

Input Area	80Hex 00Hex 00Hex 00Hex 00Hex 00Hex 00Hex 00Hex 00Hex 00Hex 00Hex 00Hex 00Hex 00Hex 00Hex 00Hex
Output Area	**80**Hex **00**Hex 00Hex 00Hex 00Hex 00Hex 00Hex 00Hex

- VB14A reads a barcode with content "*1234567890abcde1234567890abcde*". Let's assume standard data formatting with <STX> as header and <CR><LF> as terminators. In this condition since the whole message cannot be included in Input Area, CBox-300 transmits first fragment "<STX>*1234567890ab*" only (setting More Bit = 1) then it toggles bit A.

Input Area	**89**Hex 00Hex **0D**Hex **02**Hex **31**Hex **32**Hex **33**Hex **34**Hex **35**Hex **36**Hex **37**Hex **38**Hex **39**Hex **30**Hex **61**Hex **62**Hex
Output Area	80Hex 00Hex 00Hex 00Hex 00Hex 00Hex 00Hex 00Hex

- PLC detects transition of bit **A** so now it can read first incoming fragment (it copies 13 bytes in its memory from IN[3] on) then toggles bit **B** as acknowledge.

Input Area	89_{Hex}	00_{Hex}	$0D_{Hex}$	02_{Hex}	31_{Hex}	32_{Hex}	33_{Hex}	34_{Hex}	35_{Hex}	36_{Hex}	37_{Hex}	38_{Hex}	39_{Hex}	30_{Hex}	61_{Hex}	62_{Hex}
Output Area	**81_{Hex}**	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}								

- C-Box detects transition of bit **B** so it sends second fragment "*cde1234567890*" (still More Bit = 1) and toggles bit **A**.

Input Area	**88_{Hex}**	00_{Hex}	**$0D_{Hex}$**	**63_{Hex}**	**64_{Hex}**	**65_{Hex}**	**31_{Hex}**	**32_{Hex}**	**33_{Hex}**	**34_{Hex}**	**35_{Hex}**	**36_{Hex}**	**37_{Hex}**	**38_{Hex}**	**39_{Hex}**	**30_{Hex}**
Output Area	81_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}								

- PLC reads second fragment (it copies 13 bytes in its memory from IN[3] on) then toggles bit **B** as acknowledge.

Input Area	88_{Hex}	00_{Hex}	$0D_{Hex}$	63_{Hex}	64_{Hex}	65_{Hex}	31_{Hex}	32_{Hex}	33_{Hex}	34_{Hex}	35_{Hex}	36_{Hex}	37_{Hex}	38_{Hex}	39_{Hex}	30_{Hex}
Output Area	**80_{Hex}**	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}								

- CBox sends third (last) fragment "*abcde<CR><LF>*" (finally More Bit = 0) and toggles bit **A**.

Input Area	**81_{Hex}**	00_{Hex}	**07_{Hex}**	**61_{Hex}**	**62_{Hex}**	**63_{Hex}**	**64_{Hex}**	**65_{Hex}**	**$0D_{Hex}$**	**$0A_{Hex}$**	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}
Output Area	80_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}								

- PLC reads last fragment (it copies 7 bytes in its memory from IN[3] on) and now the reassembling can be completed. Then it toggles bit **B** as acknowledge.

Input Area	81_{Hex}	00_{Hex}	07_{Hex}	61_{Hex}	62_{Hex}	63_{Hex}	64_{Hex}	65_{Hex}	$0D_{Hex}$	$0A_{Hex}$	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}
Output Area	**81_{Hex}**	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}	00_{Hex}								

- Whole message has been completely transmitted.

2.3.2.8 SAP Services

FLUSH QUEUE is the unique driver service currently available. It performs flushing of the internal queues and may be issued at any time.

FLUSH QUEUE Service

Request: Flush data buffers (issued by the Master station to CBox-300)
Action: Flush all information from previous decoding phases
Response: Command accepted / Command rejected (generated by CBox-300 toward Master)

Application data areas must be formatted as follows:

Request Command	**byte 3**	**byte 4**
Flush data buffer	**'['** (5B Hex)	**'F'**(46 Hex)

Response Command	**byte 3**	**byte 4**
Command accepted	**'A'** (41 Hex)	' '(20 Hex)
Command rejected	**'C'** (43 Hex)	' '(20 Hex)

2.3.2.9 DAD internal queues

CBox-300 has two internal queues (one for each direction) to keep the application events: input queue and output queue.

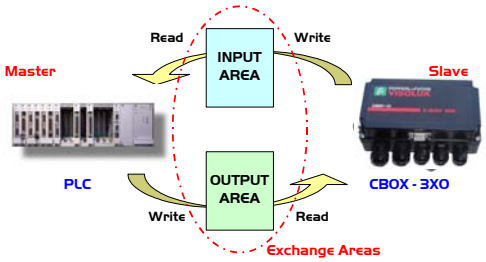
The input queue is used when a new message (generally a barcode) has to be transmitted by CBox-300 before the Master station has generated all the acknowledge handshakes for each previous transmission.
The output queue is rarely used at the moment.

The queues are sized in the following way:
- 300 elements of 32 bytes each for input queue (data flow from Slave to Master)
- 20 elements of 32 bytes each for output queue (data flow from the Master to Slave)

The queues may be flushed by the Master station through the SAP=255 primitive. This is generally done at the Master startup if the Master station wants to cancel all the previous buffers that were generated before its startup. However, the Master station is free to decide not to cancel them.

2.3.3 FCM with DPD Driver

In order to implement the flow controlled version of the driver, exchange areas must be congruently compiled in both directions.

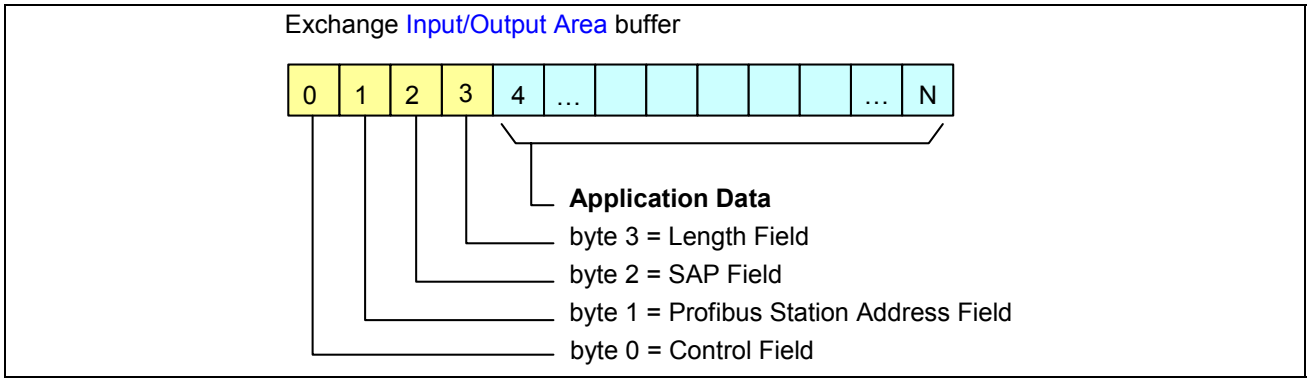


This driver supports all the features listed in the “FCM with DAD Driver” paragraph and implements all the handshake mechanisms previously discussed.

Differently from DAD Driver, DPD implement the further Profibus Station Address Field and Control Field must have bit7 set to zero (see par. 2.3.3.1).

The first **four** bytes are used by the DPD Driver layer in both buffers as follows:

Control Field (byte 0)	used to issue and control the DPD Driver primitives such as flow-control, fragmentation and resynchronisation.
Profibus Station Address Field (byte 1)	contains the information of the selected CBox-300 address of the Profibus network.
Service Access Point Field (byte 2)	used to distinguish among different services and to provide future expandability. (Since this SAP definition is introduced by the DPD Driver, it must not be confused with the SAP that is defined by the international standard).
Length Field (byte 3)	contains the number of bytes used by the application layer: Length Field ≤ (InputAreaSize – 4) for the Input Area Length Field ≤ (OutputAreaSize – 4) for the Output Area.

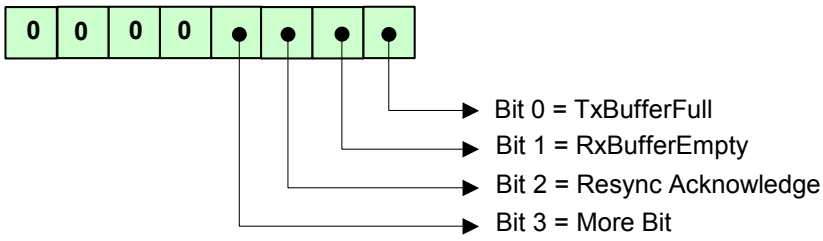


The Application Data buffer holds the information starting from byte IN[4] and the fragmentation process is managed by CBox-300 when barcode messages are longer than (InputAreaSize – 4).

2.3.3.1 Control Field

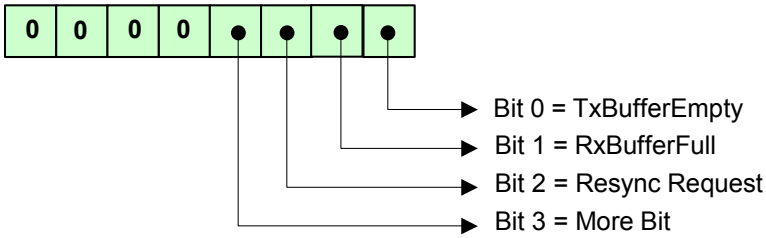
Differently from DAD Driver, just note that Control Field must have bit7 set to zero.

Control Field byte of Input Area → IN[0]



IN[0].bit0	**TxBufferFull** toggles when Slave has made available new Input Area data
IN[0].bit1	**RxBufferEmpty** toggles when Output Area data has been read by Slave
IN[0].bit2	**Resync Acknowledge** set to 1 as an acknowledge to a resync request. With this bit, the master can detect a slave is on line.
IN[0].bit3	**More Bit** is 1 when this is not the last piece of a fragmentation sequence while it is 0 when this is the last piece
IN[0].bit4,5,6,7	Set to **0**, **0**, **0**, **0** when DPD messaging protocol is used

Control Field byte of Output Area → OUT[0]



OUT[0].bit0	**TxBufferEmpty** must toggle when Input Area data has been read by Master
OUT[0].bit1	**RxBufferFull** must toggle when Master makes available new Output Area data
OUT[0].bit2	**Resync Request** set to 1 for one second to resynchronize the slave. After resynchronization, all 4 handshake bits are set to 0
OUT[0].bit3	**More Bit** must be 1 when this is not the last piece of a fragmentation sequence and it must be 0 when this is the last
OUT[0].bit4,5,6,7	Set to **0**, **0**, **0**, **0** when DPD messaging protocol is used

3. Network Configuration

3.1 GSD file

A GSD file is a readable ASCII text file that contains a complete description of the specific device. Basically GSD includes both general info (i.e. vendor and device name, hw/sw releases) and device specific info (Input and Output area size, communication parameters, scanner setup parameters and so on).

Powerful configuration tools (i.e. Siemens SIMATIC Manager) are available to setup a Profibus network. Based on the GSD files, these allow easy configuration of Profibus networks with devices from different manufacturers.

Note: firstly a GSD file must be installed into the PLC environment in order to let a new device be identified and to work on the Profibus network. Follow the instructions in the "*GSD Installation*" paragraph.

CBox-300 is equipped with the following files:

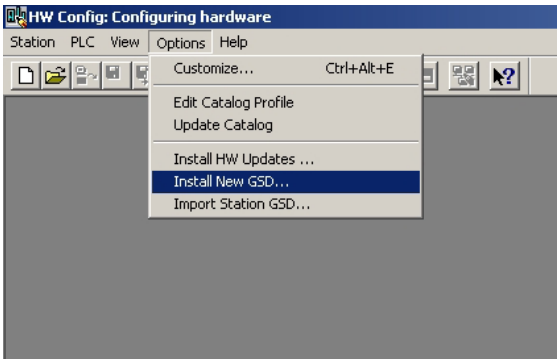
- P+F_071E.GSD } Device description file
- PFCB3.DIB
- PFCB3_DI.DIB } Custom identification icons
- PFCB3_SF.DIB

The GSD file is a certified part of the device and must not be changed manually. This file is also not changed by the configuration tool.

3.2 GSD Installation

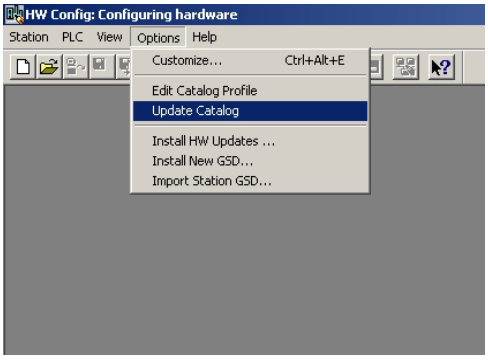
The first action that must be done is to add the CBox-300 as a new Profibus-DP Slave among the catalogue of suitable devices for the PLC.

1. *Install the new GSD file...*

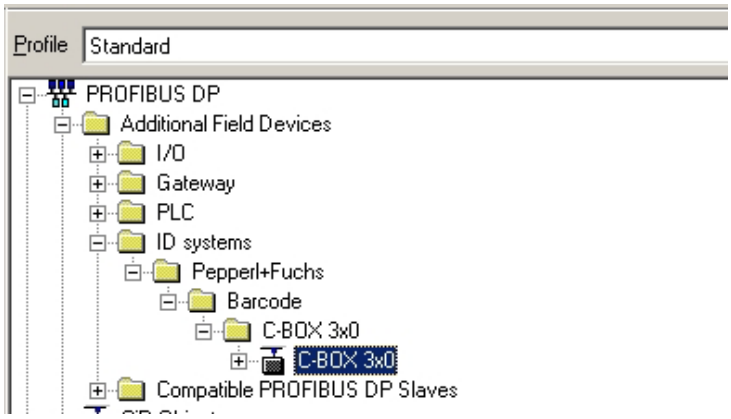


2. *Find the GSD file…*
   (GSD and *.DIB files must be in the same directory)

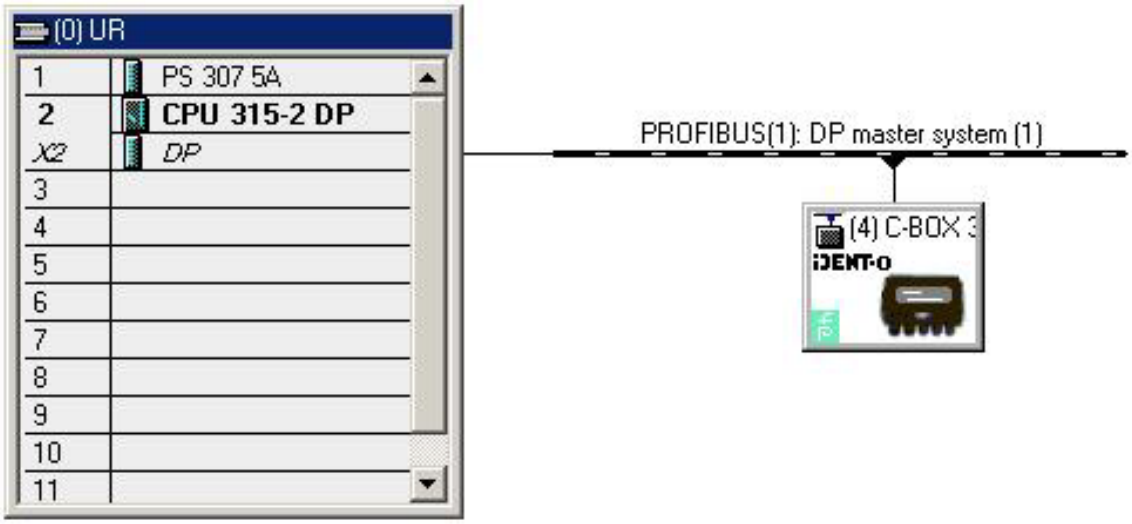
3. *Update Catalogue…*



4. *Find new device…*
   A new *C-BOX 3x0* device will appear in the PLC catalogue under *Profibus-DP → Additional Field Devices → ID Systems → Pepperl+Fuchs →* Barcode→ *C-BOX 3x0* folder



5. *Insert the device in the Profibus network…*
   The easy drag&drop function allows inserting the *C-BOX 3x0* device in your own network



3.3 Scanner Programming via GSD File

The major benefit of the GSD file is to setup the scanner by means of the Profibus Master without the using the VisoSetup program: it is possible to select the barcodes, the scanner operating mode and many other relevant parameters.

Different scanner parameters are grouped in Project Modules that can be used to create custom configurations. The User should include the desired modules in his own PLC project and select the proper values for each parameter. Once completed, the PLC will automatically program the scanner at each power on and also any time it is re-discovered on the network (reconnected).

Since the scanner configuration is stored in the PLC, scanner replacement operations become extremely easy and quick.

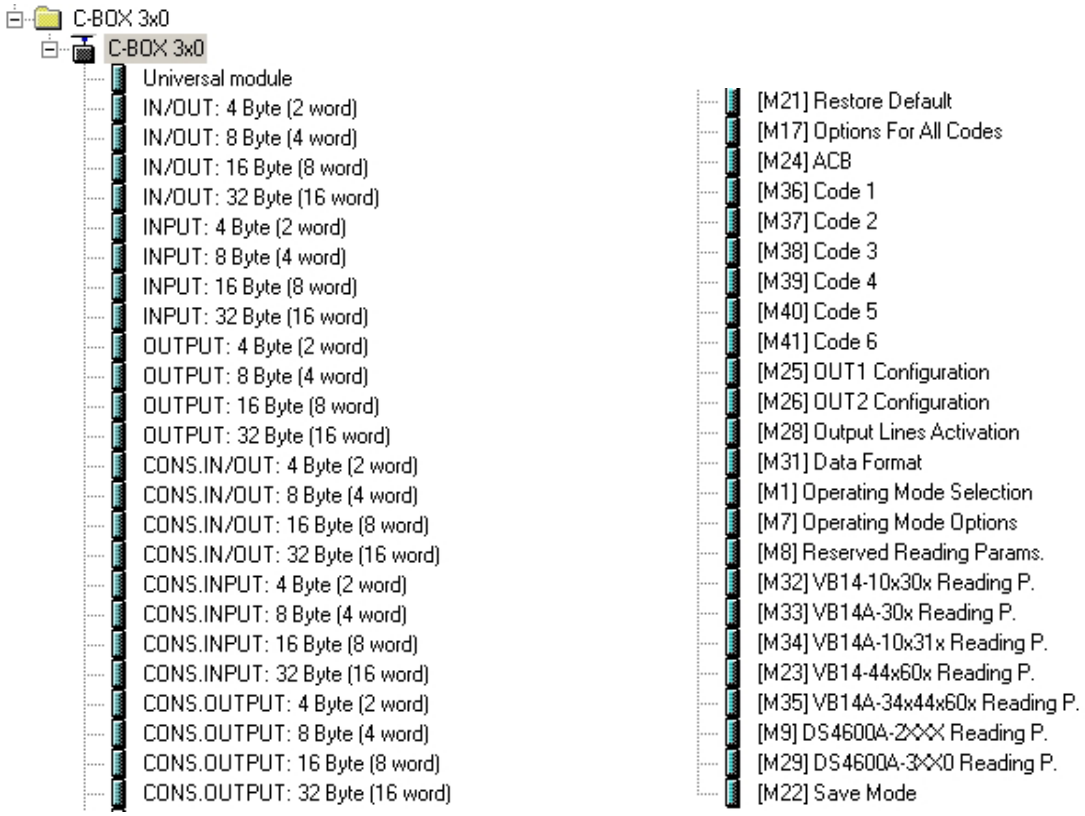
Following description refers to GSD version 1.02.

3.3.1 Project Modules

Three types of modules are available:

- **[CONS. IN / OUT: xx Byte]** modules
  [CONS. INPUT: xx Byte] modules
  [CONS. OUTPUT: xx Byte] modules → I/O module with Data Consistency over the whole string
- **[INPUT: xx Byte]** modules
  [OUTPUT: xx Byte] modules → I/O module with Data Consistency over bytes/words
- **[Mxx]** modules → Scanner Parameters module

Modules overview:



At least one I/O module is required to set the dimensions of both the Input and Output exchange areas. If needed, one or more I/O modules can be combined.

Scanner parameterisation is optional.

Some notes:

→ all of the available modules are generally executed in the same order they are inserted.

→ [M21] Restore Default module, independently from its position, is executed first.
This module allows a congruent configuration.

→ [M22] Save Mode module, independently from its position, is executed as last command.
Typically this module is not necessary.

Let's see the details of each module:

[M21] Restore Default module

Parameter	*Options*
<No Parameters>	<No options>

[M17] Options For All Codes module

Parameter	*Options*
Multi Label	"Disabled" "Enabled"
Global No Read Character	"Disabled" "NUL", "SOH", … , " }" , "~"

[M24] ACB module

Parameter	*Options*
2/5 Interleaved	"Disabled" "Enabled"
Code 39 and derivates	"Disabled" "Enabled"
Codabar	"Disabled" "Enabled"
Code 128 / EAN 128	"Disabled" "Enabled"
EAN / UPC	"Disabled" "Enabled"
Code 93	"Disabled" "Enabled"

[M11] Code 1 module
[M12] Code 2 module
[M13] Code 3 module
[M14] Code 4 module
[M15] Code 5 module
[M16] Code 6 module

Parameter	*Options*
Code Type	"Code 128" "Interleaved 2 of 5" "Code 39" "Code EAN 128" "EAN-13" "EAN-8" "UPC-A" "UPC-E" "All EAN_UPC" "CODABAR" "Code 93"
Check Digit	"Disabled" "Enabled"
Check Digit Tx	"Disabled" "Enabled"

Label Length	"Variable" "1" … "48"

[M25] OUT1 Configuration module
[M26] OUT2 Configuration module

Parameter	*Options*
Event	"No Read" "Right" "Wrong" "Wrong/No Read" "Wrong/Right" "Right/No Read"
Idle State	"Normally Open" "Normally Closed"
Mode	"Disabled" "Level" "10 ms pulse" "20 ms pulse" "30 ms pulse" "40 ms pulse" "50 ms pulse" "60 ms pulse" "70 ms pulse" "80 ms pulse" "90 ms pulse" "100 ms pulse" "110 ms pulse" "120 ms pulse" "130 ms pulse" "140 ms pulse" "150 ms pulse" "160 ms pulse" "170 ms pulse" "180 ms pulse" "190 ms pulse" "200 ms pulse" "210 ms pulse" "220 ms pulse" "230 ms pulse" "240 ms pulse" "250 ms pulse" "300 ms pulse" "350 ms pulse" "400 ms pulse" "450 ms pulse" "500 ms pulse" "550 ms pulse" "600 ms pulse" "650 ms pulse" "700 ms pulse" "750 ms pulse" "800 ms pulse" "850 ms pulse" "900 ms pulse" "950 ms pulse" "1000 ms pulse" "1050 ms pulse" "1100 ms pulse" "1050 ms pulse" "1200 ms pulse" "1050 ms pulse"

	"1300 ms pulse" "1050 ms pulse" "1400 ms pulse" "1050 ms pulse" "1500 ms pulse"

[M28] Output Lines Activation module

Parameter	*Options*
Output Line Active	"On Decoding" "After Reading Phase Off"

[M31] Data Format module

Parameter	*Options*
Header 1	"Disabled" "NUL", "SOH", … , " }" , "~"
Header 2	"Disabled" "NUL", "SOH", … , " }" , "~"
Header 3	"Disabled" "NUL", "SOH", … , " }" , "~"
Header 4	"Disabled" "NUL", "SOH", … , " }" , "~"
Code Field Length	"Variable" "Code Length" "1", "2", … , "49", "50"
Data Justification	"Disabled" "NUL", "SOH", … , " }" , "~"
Fill Character	"NUL", "SOH", … , " }" , "~"
Reader Failure Char.	"NUL", "SOH", … , " }" , "~"
Data Packet Separator 1	"Disabled" "NUL", "SOH", … , " }" , "~"
Data Packet Separator 2	"Disabled" "NUL", "SOH", … , " }" , "~"
Info Field Separator 1	"Disabled" "NUL", "SOH", … , " }" , "~"
Info Field Separator 2	"Disabled" "NUL", "SOH", … , " }" , "~"
Code Identifier Tx	"Disabled" "Enabled"
Header Tx Start	"With Data" "After Reading Phase On"
Data Tx Start	"On Decoding" "After Reading Phase Off"

[M1] Operating Mode Selection module

Parameter	*Options*
Operating Mode	"On Line" "Serial On Line" "Automatic" "Test"

[M7] Operating Mode Options module

Parameter	*Options*
External Trigger	"Standard" "Inverse"
External Trigger Filter	"Disabled" "Enabled"
Serial Start	"NUL", "SOH", … , " }" , "~"
Serial Stop	"NUL", "SOH", … , " }" , "~"
Reading Phase Timeout	"Disabled" "30ms" "60ms" "80ms" "100ms" "200ms" "300ms" "400ms" "500ms" "600ms" "700ms" "800ms" "900ms" "1 s" "1.5 s" "2 s" "2.5 s" "3 s" "4 s"
Reading Phase Off	Trigger off / Serial Stop" "Always Timeout"
Automatic Threshold	"5" "10" "15" "20" "25" "30" "35" "40" "45" "50" "100" "150" "200" "250" "300" "350" "400"
Test On	"Good Read Rate" "Positioning Quality"

[M8] Reserved Reading Params. module

Parameter	*Options*
Beam Shutter	"Disabled" "Triggered" "Enabled"
Overflow	"0.87 µs" "1.74 µs" "2.60 µs" "3.47 µs" "4.34 µs" "5.21 µs" "6.08 µs" "6.94 µs" "7.81 µs" "8.68 µs" "9.55 µs" "10.42 µs" "12.15 µs" "13.88 µs" "15.62 µs" "16.50 µs" "18.23 µs" "19.96 µs" "21.70 µs" "23.44 µs" "25.17 µs" "26.91 µs" "28.44 µs" "30.38 µs" "32.98 µs" "35.59 µs" "38.19 µs" "40.80 µs" "44.27 µs" "47.74 µs" "51.21 µs" "55.55 µs" "59.89 µs" "64.23 µs" "68.57 µs" "72.91 µs" "77.25 µs" "81.59 µs" "86.80 µs" "92.01 µs" "97.22 µs" "102.4 µs" "107.6 µs" "110.7 µs"

[M32] VB14-10x30x Reading P. module

Parameter	*Options*
Beam Shutter	"Disabled" "Triggered" "Enabled"
Overflow	"0.87 µs" "1.74 µs" "2.60 µs" "3.47 µs" "4.34 µs" "5.21 µs" "6.08 µs" "6.94 µs" "7.81 µs" "8.68 µs" "9.55 µs" "10.42 µs" "12.15 µs" "13.88 µs" "15.62 µs" "16.50 µs" "18.23 µs" "19.96 µs" "21.70 µs" "23.44 µs" "25.17 µs" "26.91 µs" "28.44 µs" "30.38 µs" "32.98 µs" "35.59 µs" "38.19 µs" "40.80 µs" "44.27 µs" "47.74 µs" "51.21 µs" "55.55 µs" "59.89 µs" "64.23 µs" "68.57 µs" "72.91 µs" "77.25 µs" "81.59 µs" "86.80 µs" "92.01 µs" "97.22 µs" "102.4 µs" "107.6 µs" "110.7 µs"
Code Resolution	“Standard” “High” “Toggle”

[M33] VB14A-30x Reading P. module

Parameter	*Options*
Beam Shutter	"Disabled" "Triggered" "Enabled"
Overflow	"0.87 µs" "1.74 µs" "2.60 µs" "3.47 µs" "4.34 µs" "5.21 µs" "6.08 µs" "6.94 µs" "7.81 µs" "8.68 µs" "9.55 µs" "10.42 µs" "12.15 µs" "13.88 µs" "15.62 µs" "16.50 µs" "18.23 µs" "19.96 µs" "21.70 µs" "23.44 µs" "25.17 µs" "26.91 µs" "28.44 µs" "30.38 µs" "32.98 µs" "35.59 µs" "38.19 µs" "40.80 µs" "44.27 µs" "47.74 µs" "51.21 µs" "55.55 µs" "59.89 µs" "64.23 µs" "68.57 µs" "72.91 µs" "77.25 µs" "81.59 µs" "86.80 µs" "92.01 µs" "97.22 µs" "102.4 µs" "107.6 µs" "110.7 µs"
Motor Control	“Motor Off” “Speed_2” “Speed_3”
Code Reading Condition	“Standard” “Difficult” “Toggle”
Serial Motor Start	“Disabled” "NUL”, “SOH”, … , ” }” , “~"
Serial Motor Stop	“Disabled” "NUL”, “SOH”, … , ” }” , “~"

[M34] VB14A-10x31x Reading P. module

Parameter	*Options*
Beam Shutter	"Disabled" "Triggered" "Enabled"
Overflow	"0.87 µs" "1.74 µs" "2.60 µs" "3.47 µs" "4.34 µs" "5.21 µs" "6.08 µs" "6.94 µs" "7.81 µs" "8.68 µs" "9.55 µs" "10.42 µs" "12.15 µs" "13.88 µs" "15.62 µs" "16.50 µs" "18.23 µs" "19.96 µs" "21.70 µs" "23.44 µs" "25.17 µs" "26.91 µs" "28.44 µs" "30.38 µs" "32.98 µs" "35.59 µs" "38.19 µs" "40.80 µs" "44.27 µs" "47.74 µs" "51.21 µs" "55.55 µs" "59.89 µs" "64.23 µs" "68.57 µs" "72.91 µs" "77.25 µs" "81.59 µs" "86.80 µs" "92.01 µs" "97.22 µs" "102.4 µs" "107.6 µs" "110.7 µs"
Motor Control	“Motor Off” “Speed_3” “Speed_4”
Code Reading Condition	“Standard” “Difficult” “Toggle”
Serial Motor Start	“Disabled” "NUL”, “SOH”, … , ” }” , “~"
Serial Motor Stop	“Disabled” "NUL”, “SOH”, … , ” }” , “~"

[M23] VB14-44x60x Reading P. module

Parameter	*Options*
Beam Shutter	"Disabled" "Triggered" "Enabled"
Overflow	"0.87 µs" "1.74 µs" "2.60 µs" "3.47 µs" "4.34 µs" "5.21 µs" "6.08 µs" "6.94 µs" "7.81 µs" "8.68 µs" "9.55 µs" "10.42 µs" "12.15 µs" "13.88 µs" "15.62 µs" "16.50 µs" "18.23 µs" "19.96 µs" "21.70 µs" "23.44 µs" "25.17 µs" "26.91 µs" "28.44 µs" "30.38 µs" "32.98 µs" "35.59 µs" "38.19 µs" "40.80 µs" "44.27 µs" "47.74 µs" "51.21 µs" "55.55 µs" "59.89 µs" "64.23 µs" "68.57 µs" "72.91 µs" "77.25 µs" "81.59 µs" "86.80 µs" "92.01 µs" "97.22 µs" "102.4 µs" "107.6 µs" "110.7 µs"
Scanning Speed	“Speed_1” “Speed_2” “Speed_3” “Speed_4”
Code Reading Condition	“Standard” “Difficult” “Toggle”

[M35] VB14A-34x44x60x Reading Params. module

Parameter	*Options*
Beam Shutter	"Disabled" "Triggered" "Enabled"
Overflow	"0.87 µs" "1.74 µs" "2.60 µs" "3.47 µs" "4.34 µs" "5.21 µs" "6.08 µs" "6.94 µs" "7.81 µs" "8.68 µs" "9.55 µs" "10.42 µs" "12.15 µs" "13.88 µs" "15.62 µs" "16.50 µs" "18.23 µs" "19.96 µs" "21.70 µs" "23.44 µs" "25.17 µs" "26.91 µs" "28.44 µs" "30.38 µs" "32.98 µs" "35.59 µs" "38.19 µs" "40.80 µs" "44.27 µs" "47.74 µs" "51.21 µs" "55.55 µs" "59.89 µs" "64.23 µs" "68.57 µs" "72.91 µs" "77.25 µs" "81.59 µs" "86.80 µs" "92.01 µs" "97.22 µs" "102.4 µs" "107.6 µs" "110.7 µs"
Motor Control	“Motor Off” “Speed_2” “Speed_3” “Speed_4”
Code Reading Condition	“Standard” “Difficult” “Toggle”
Serial Motor Start	“Disabled” "NUL”, “SOH”, … , ” }” , “~"
Serial Motor Stop	“Disabled” "NUL”, “SOH”, … , ” }” , “~"

[M9] DS4600A-2XXX Reading P. module

Parameter	*Options*
Beam Shutter	"Disabled" "Triggered" "Enabled"
Overflow	"0.8 µs" "1.6 µs" "2.4 µs" "3.2 µs" "4.0 µs" "4.8 µs" "5.6 µs" "6.4 µs" "7.2 µs" "8.0 µs" "8.8 µs" "9.6 µs" "10.4 µs" "12.0 µs" "13.6 µs" "15.2 µs" "16.8 µs" "18.4 µs" "20.0 µs" "21.6 µs" "23.2 µs" "24.8 µs" "26.4 µs" "28.0 µs" "29.6 µs" "31.2 µs" "32.8 µs" "34.4 µs" "36.0 µs" "37.6 µs" "39.2 µs" "40.8 µs" "42.4 µs" "44.0 µs" "45.6 µs" "47.2 µs" "48.8 µs" "50.4 µs" "52.8 µs" "55.2 µs" "57.6 µs" "60.0 µs" "62.4 µs" "64.8 µs" "68.0 µs"
Code Resolution	“Standard” “High” “Toggle”
Motor Control	“Motor On” “Motor Off”
Code Reading Condition	“Standard” “Difficult” “Very Difficult”
Noise Reduction	“Off” “On” “Toggle”

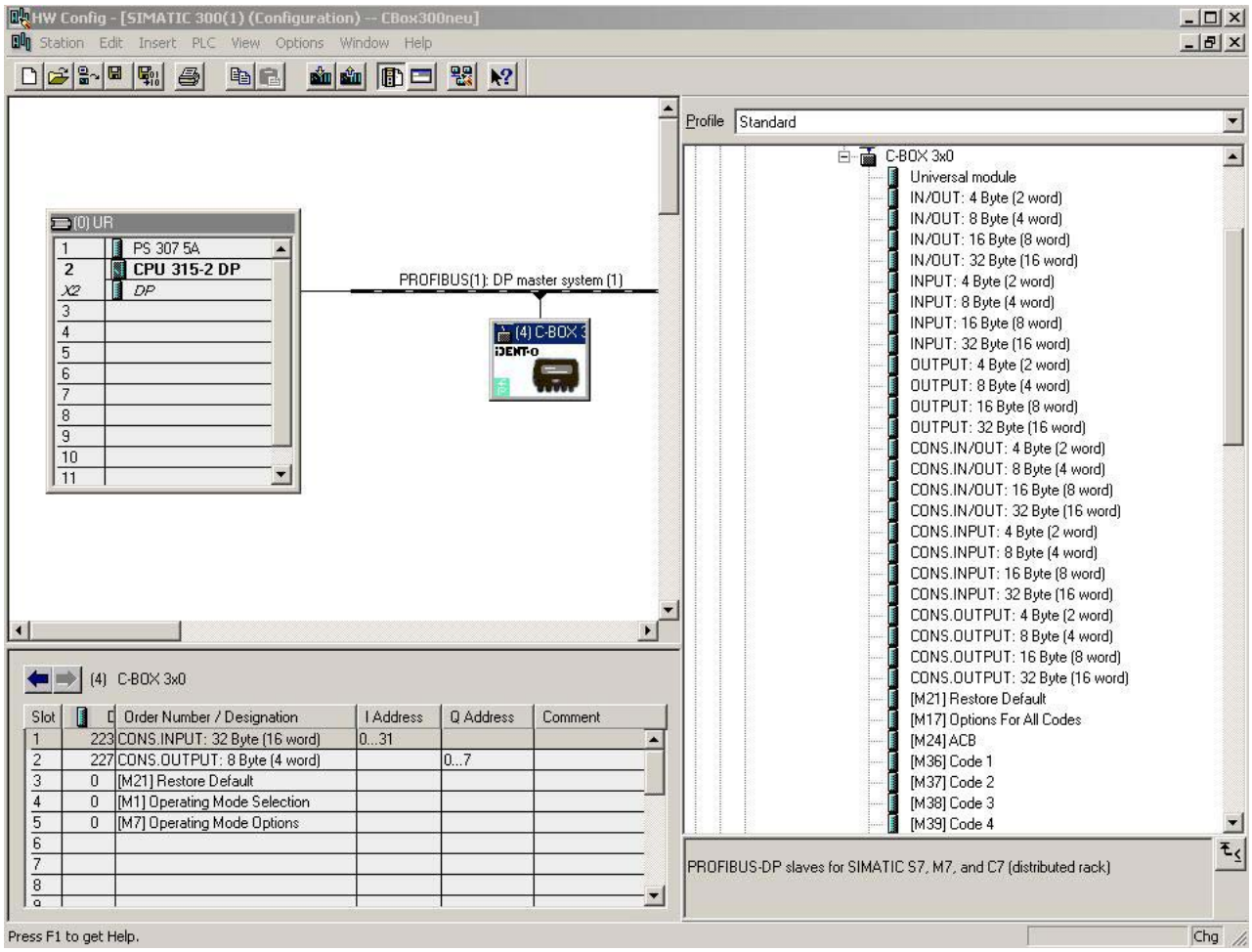
[M29] DS4600A-3XX0 Reading P. module

Parameter	*Options*
Beam Shutter	"Disabled" "Triggered" "Enabled"
Automatic Overflow Ratio	"3", "4", … , "99", "100"
Code Resolution	“Standard” “High” “Toggle”
Motor Control	“Motor On” “Motor Off”
Code Reading Condition	“Standard” “Difficult” “Very Difficult”
Noise Reduction	“Off” “On” “Toggle”
Code Reconstruction	“Enabled” “Disabled”

[M22] Save Mode module

Parameter	*Options*
Save Mode	"Save In EEPROM (permanent)" "Save In RAM Only (volatile)"

A typical scanner programming project is as follows:



Here you have:

[CONS INPUT: 32 Byte] module to set the Input exchange areas.
[CONS OUTPUT: 8 Byte] module to set the Output exchange areas.
[M21] module to restore the scanner configuration to Default.
[M1] module to select the scanner operating mode, i.e. OnLine mode.
[M7] module to indicate options such as Timeout

FACTORY AUTOMATION – SENSING YOUR NEEDS

Worldwide Headquarters
Pepperl+Fuchs GmbH
68307 Mannheim · Germany
Tel. +49 621 776-0
E-mail: info@de.pepperl-fuchs.com

USA Headquarters
Pepperl+Fuchs Inc.
Twinsburg, Ohio 44087 · USA
Tel. +1 330 4253555
E-mail: sales@us.pepperl-fuchs.com

Asia Pacific Headquarters
Pepperl+Fuchs Pte Ltd.
Company Registration No. 199003130E
Singapore 139942
Tel. +65 67799091
E-mail: sales@sg.pepperl-fuchs.com

www.pepperl-fuchs.com

PEPPERL+FUCHS
SENSING YOUR NEEDS

TDOCT-1144_ENG 198586
12/2006