

DC303

smar

MAR / 17
DC303
VERSION 3



OPERATION & MAINTENANCE
INSTRUCTIONS MANUAL

REMOTE I/O PROFIBUS - PA



DC303ME



Specifications and information are subject to change without notice.
Up-to-date address information is available on our website.

[web: www.smar.com/contactus.asp](http://www.smar.com/contactus.asp)

INTRODUCTION

Until all types of devices are available with Profibus-PA systems will have to be of a hybrid nature accepting both Profibus and conventional signals. A mixed traditional and Profibus environment is inevitable during the transition to a Profibus technology. DC303 makes integration of Profibus and conventional I/O easy. Discrete devices such as pressure switches, push buttons, on/off valves, pumps and conveyors are integrated to the system over the Profibus-PA field-level network using DC303. The DC303 remote I/O units can be distributed into the field where they are mounted close to the conventional devices without the need to run the conventional wiring to the control room. The DC303 is an integral part of SYSTEM302 but also it integrates into other systems supporting Profibus.

The DC303 makes conventional analog and discrete inputs and outputs available using standard Profibus-PA Function blocks making the system homogenous and control strategy configuration easy as conventional I/O appears as if they were regular Profibus devices. Control loops are implemented consistently regardless of I/O being conventional or Profibus based. Only a single programming language has to be used.

The DC303 is a simple low-cost DIN-rail mounted unit. The DC303 is a single integrated easy to use piece of equipment including power, control, networking and I/O under one compact device requiring less panel space than other solutions.

Function block like Discrete Input and Discrete Output, as well FFB (Flexible Function Block) enables the DC303 to perform logic and r control functions in the field integrating the control strategy with other Profibus devices on the same network. Function blocks provide great flexibility in control strategy. Conventional discrete I/O now works together with pure Profibus devices on the same network and in the same loop. The DC303 is fully configured SYSTEM302 or any other Profibus-PA configuration tool. Function blocks provide logic such as AND, OR, NAND etc. as well as latches etc.

The DC303 may be installed close to the sensors and actuators, thereby eliminating long wire runs and associated marshalling panels and cable trays for the conventional I/O, with subsequent savings further reducing overall system cost. Use DC303 to make it possible to distribute I/O at various locations in the field and connect them via Profibus-PA. DC303 is ideal to connect motor control centers, variable speed drives, and electrical actuators and motor operated valves to Profibus-PA.

Get the best result of the DC303 by carefully reading these instructions.

NOTE

In case of using Simatic PDM as the configuration and parameterization tool, Smar recommends that the user does not apply the option "Download to Device". This function can improperly configure the field device. Smar recommends that user make the use of the option "Download to PG / PC" and then selecting the Device Menu, use the menus of the transducer, function and display blocks acting specifically, according to each menu and method for reading and writing.



WARNING

This manual applies to equipment with a serial number from 2000 onwards.

NOTE

For previous DC303 versions (with plastic cowling), pick up equivalent manual.

Waiver of responsibility

The contents of this manual abides by the hardware and software used on the current equipment version. Eventually there may occur divergencies between this manual and the equipment. The information from this document are periodically reviewed and the necessary or identified corrections will be included in the following editions. Suggestions for their improvement are welcome.

Warning

For more objectivity and clarity, this manual does not contain all the detailed information on the product and, in addition, it does not cover every possible mounting, operation or maintenance cases.

Before installing and utilizing the equipment, check if the model of the acquired equipment complies with the technical requirements for the application. This checking is the user's responsibility.

If the user needs more information, or on the event of specific problems not specified or treated in this manual, the information should be sought from Smar. Furthermore, the user recognizes that the contents of this manual by no means modify past or present agreements, confirmation or judicial relationship, in whole or in part.

All of Smar's obligation result from the purchasing agreement signed between the parties, which includes the complete and sole valid warranty term. Contractual clauses related to the warranty are not limited nor extended by virtue of the technical information contained in this manual.

Only qualified personnel are allowed to participate in the activities of mounting, electrical connection, startup and maintenance of the equipment. Qualified personnel are understood to be the persons familiar with the mounting, electrical connection, startup and operation of the equipment or other similar apparatus that are technically fit for their work. Smar provides specific training to instruct and qualify such professionals. However, each country must comply with the local safety procedures, legal provisions and regulations for the mounting and operation of electrical installations, as well as with the laws and regulations on classified areas, such as intrinsic safety, explosion proof, increased safety and instrumented safety systems, among others.

The user is responsible for the incorrect or inadequate handling of equipments run with pneumatic or hydraulic pressure or, still, subject to corrosive, aggressive or combustible products, since their utilization may cause severe bodily harm and/or material damages.

The field equipment referred to in this manual, when acquired for classified or hazardous areas, has its certification void when having its parts replaced or interchanged without functional and approval tests by Smar or any of Smar authorized dealers, which are the competent companies for certifying that the equipment in its entirety meets the applicable standards and regulations. The same is true when converting the equipment of a communication protocol to another. In this case, it is necessary sending the equipment to Smar or any of its authorized dealer. Moreover, the certificates are different and the user is responsible for their correct use.

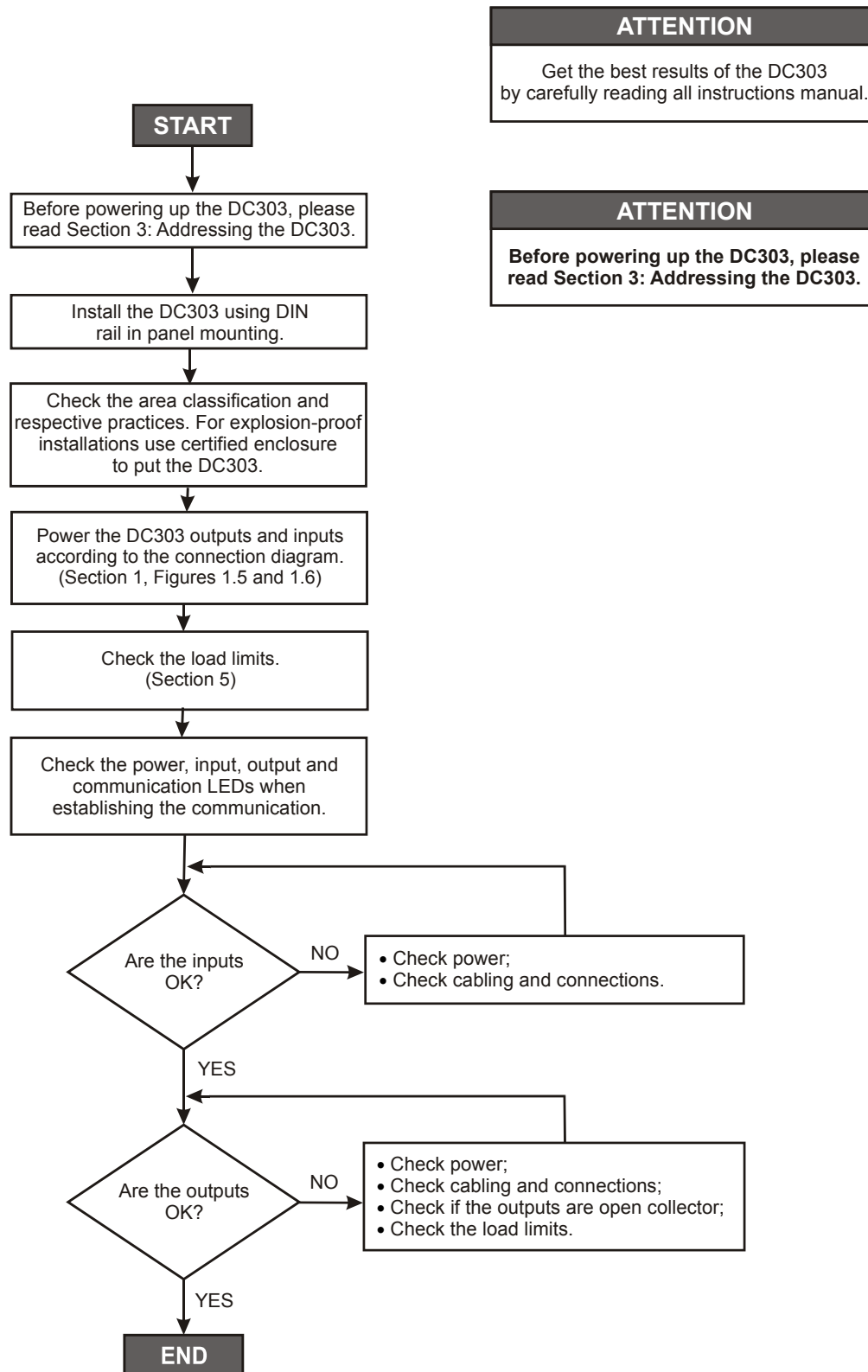
Always respect the instructions provided in the Manual. Smar is not responsible for any losses and/or damages resulting from the inadequate use of its equipments. It is the user's responsibility to know and apply the safety practices in his country.

TABLE OF CONTENTS

SECTION 1 - INSTALLATION	1.1
GENERAL	1.1
MOUNTING	1.1
ELECTRIC WIRING	1.2
TOPOLOGY AND NETWORK CONFIGURATION	1.5
GENERAL SYSTEM	1.6
SECTION 2 - OPERATION	2.1
FUNCTIONAL DESCRIPTION – ELECTRONICS	2.1
(CPU) CENTRAL PROCESSING UNIT, FRAM	2.1
COMMUNICATION CONTROLLER	2.1
POWER SUPPLY	2.1
FACTORY RESET	2.1
INPUT LATCHES	2.1
OUTPUT LATCHES	2.1
OPTICAL ISOLATION	2.1
SECTION 3 - CONFIGURATION	3.1
CONNECTING PHYSICAL SIGNALS TO DIGITAL INPUT BLOCK	3.1
CONNECTING PHYSICAL SIGNALS TO DIGITAL OUTPUT BLOCK	3.1
EXAMPLES OF APPLICATIONS	3.2
EXECUTING LOGIC WITH DC303	3.3
DESCRIPTION	3.3
STATUS	3.3
SUPPORTED MODES	3.3
SCHEMATIC	3.4
PARAMETERS	3.4
FUNCTIONS	3.10
TP TIMER PULSE	3.10
TON TIMER ON-DELAY	3.10
TOF TIMER OFF-DELAY	3.11
CTD PULSE COUNTER DOWN	3.12
CTU PULSE COUNTER UP	3.12
RS FLIP-FLOP	3.12
SR FLIP-FLOP	3.12
ERROR CODE	3.13
EXAMPLE OF APPLICATIONS	3.14
DC303 CYCLICAL CONFIGURATION	3.17
CYCLICAL DIAGNOSIS	3.19
ADDRESSING THE DC303	3.21
DOWNLOAD USING SIMATIC PDM	3.22
SECTION 4 - MAINTENANCE PROCEDURES	4.1
GENERAL	4.1
DISASSEMBLY PROCEDURE	4.1
REASSEMBLY PROCEDURE	4.1
FIRMWARE UPDATE PROCEDURE	4.2
BOARDS INTERCHANGEABILITY	4.2
ACCESSORIES	4.2
EXPLODED VIEW	4.3
SPARE PARTS	4.3
SECTION 5 - TECHNICAL SPECIFICATIONS	5.1
GENERAL	5.1
DC303 INPUTS	5.1
DESCRIPTION-INPUTS	5.1
TECHNICAL SPECIFICATIONS	5.1
DC303 OPEN COLLECTOR OUTPUTS	5.2
DESCRIPTION – OUTPUTS	5.2

TECHNICAL SPECIFICATIONS	5.2
ORDERING CODE.....	5.2
APPENDIX A – SRF – SERVICE REQUEST FORM.....	A.1
RETURNING MATERIALS.....	A.2

Installation Flowchart



ATTENTION

Get the best results of the DC303 by carefully reading all instructions manual.

ATTENTION

Before powering up the DC303, please read Section 3: Addressing the DC303.

*More information in Section 1 of DC303. Operation, Maintenance and Instructions Manual.

INSTALLATION

General

ATTENTION

Before powering up the **DC303**, please, read Section 3: Addressing the **DC303**.

The overall accuracy of measurement and control depends on several variables. Although the Profibus Remote I/O has an outstanding performance, proper installation is essential, in order to maximize its performance.

Among all factors, which may affect the accuracy, environmental conditions are the most difficult to control. There are, however, ways of reducing the effects of temperature, humidity and vibration.

Locating the Profibus Remote I/O in areas protected from extreme environmental changes can improve its performance.

In warm environments, the Profibus Remote I/O should be installed to avoid, as much as possible, direct exposure to the sun. Installation close to lines and vessels subjected to high temperatures should also be avoided.

Use of sunshades or heat shields to protect the Profibus Remote I/O from external heat sources should be considered, if necessary.

Humidity is fatal to electronic circuits. In areas subjected to high relative humidity, the protection cover must be provided.

For details of mounting, please, refer to Figure 1.1 and Figure 1.2

Mounting

Use DIN rail (TS35-DIN EN 50022 or TS32-DIN EN50035 or TS15 DIN EN50045), as shown in Figure 1.1 – Mechanical Mounting. The **DC303** can optionally be supplied preinstalled in an enclosure ready for field mounting.

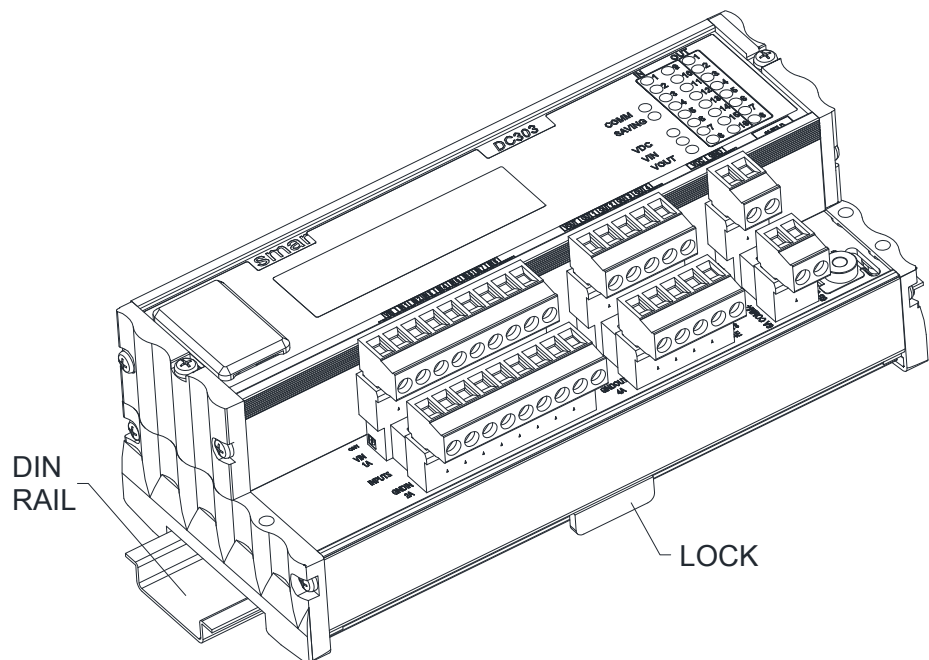
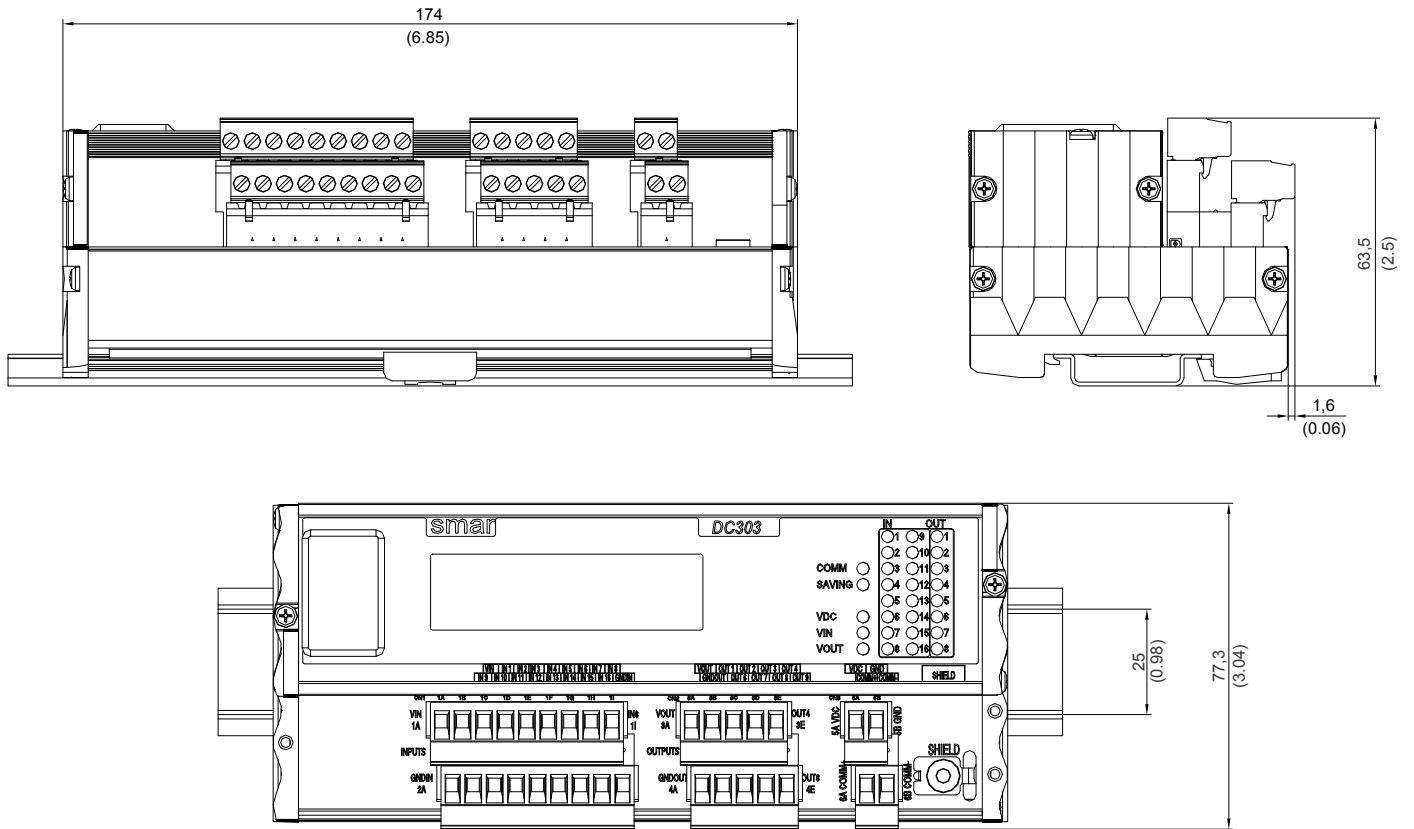


Figure 1.1- Mechanical Mounting



NOTE

The measurements are in mm.

Electric Wiring

Access the wiring block by the front View with label for inputs, outputs power supply and bus connection. The connections are made using the screws.

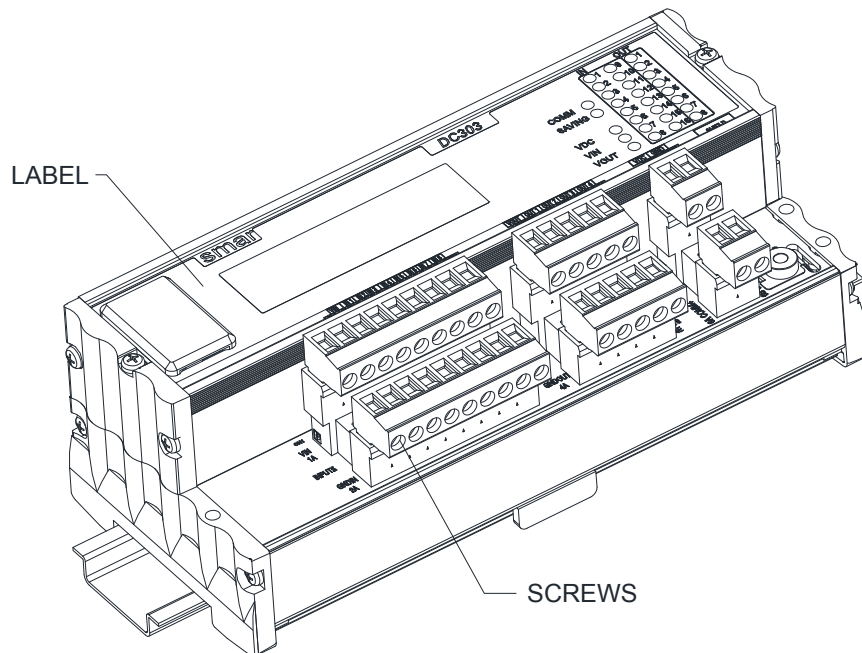


Figure 1.3 - Terminal Block Connections

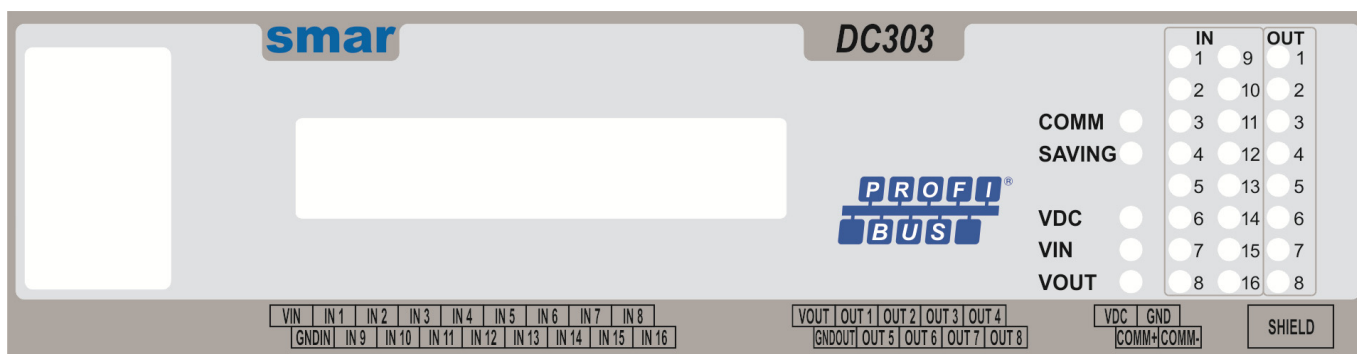


Figure 1.4 – Label DC303

The following table describes the terminal numbers for DC303.

Upper (U)	Lower (U)	Remark
VIN (1A)	GNDIN (2A)	Auxiliary power for inputs
IN1 (1B)	IN9 (2B)	Digital Inputs
IN2 (1C)	IN10 (2C)	
IN3 (1D)	IN11 (2D)	
IN4 (1E)	IN12 (2E)	
IN5 (1F)	IN13 (2F)	
IN6 (1G)	IN14 (2G)	
IN7 (1H)	IN15 (2H)	
IN8 (1I)	IN16 (2I)	
VOUT (3A)	GNDOUT (4A)	Auxiliary power to drive outputs
OUT1 (3B)	OUT5 (4B)	Digital Outputs
OUT2 (3C)	OUT6 (4C)	
OUT3 (3D)	OUT7 (4D)	
OUT4 (3E)	OUT8 (4E)	
VDC (5A)		Main power
GND (5B)		
	COMM+ (6A)	Profibus PA communication signal.
	COMM- (6B)	

Table 1.1 - Terminal Block Connections

The used connections should be plugged accordingly. For examples, please see the Figure 1.5 and Figure 1.6.

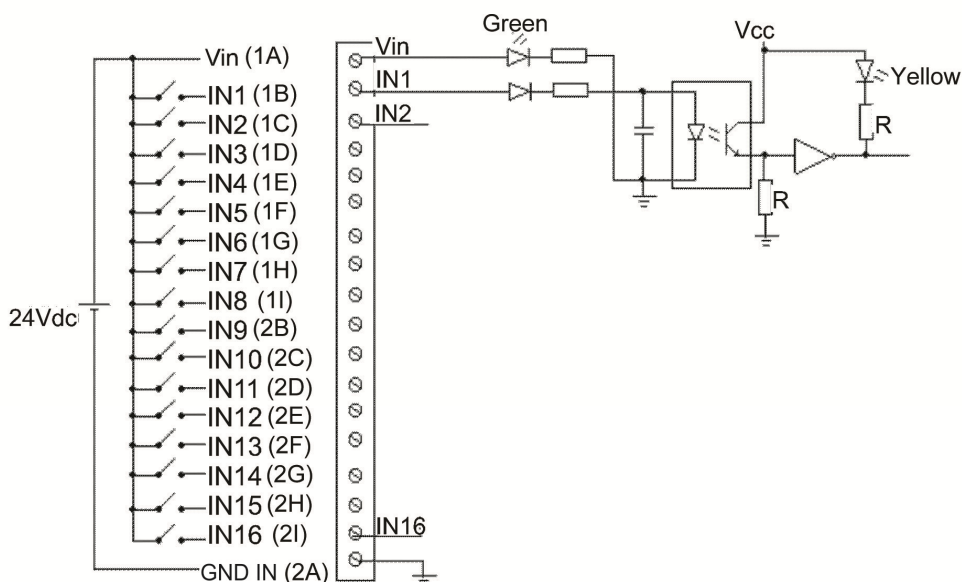


Figure 1.5 – Example of Input Connections

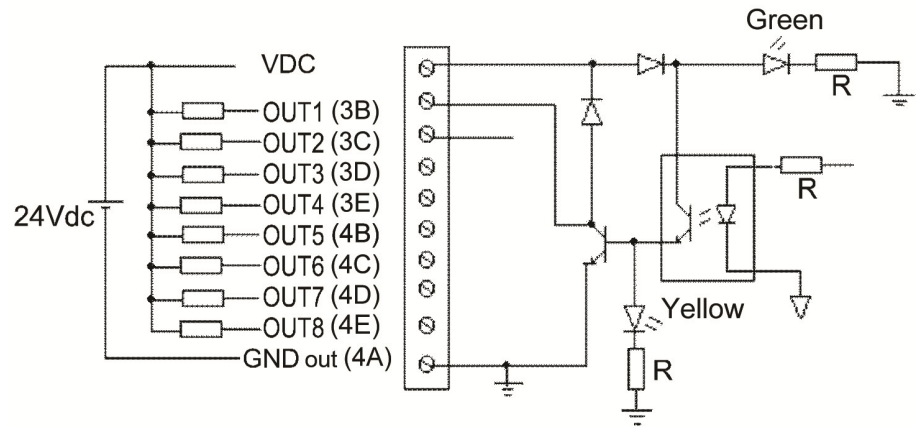


Figure 1.6 – Example of Output Connections

The **DC303** is a non bus-powered device. The **DC303** uses a 31.25 Kbit/s voltage mode option for the physical signaling. Various types of Profibus-PA devices may be connected on the same bus, being bus-powered or non-bus-powered. When bus-powered, the devices must use the same signaling.

The total of equipment on a Profibus-PA network depends on the area classification, total consumption each device on the bus, distances used, etc.

The DC303 is a non bus-powered device.

In hazardous area, the number of devices may be limited by intrinsically safe restrictions.

The **DC303** is protected against reverse polarity, and can withstand ± 35 VDC without damage.

NOTE

Please refer to the General Installation, Operation and Maintenance Manual for more details.

WARNING	
HAZARDOUS AREAS	
In hazardous zones with intrinsically safe or non-incendive requirements, the circuit entity parameters and applicable installation procedures must be observed.	
Cable access to wiring connections is obtained by the two conduit outlets. Conduit threads should be sealed by means of code-approved sealing methods.	

Topology and Network Configuration

Bus topology (See Figure 1.7 - Bus Topology) and tree topology (See Figure 1.8 - Tree Topology) are supported. Both types have a trunk cable with two terminations. The devices are connected to the trunk via spurs. The spurs may be integrated in the device giving zero spur length.

A spur may connect more than one device, depending on the length. Active couplers may be used to extend spur length.

Active repeaters may be used to extend the trunk length.

The total cable length, including spurs, between any two devices in the Profibus-PA should not exceed 1900 m.

WARNING	
POWER SUPPLIES	
If there are requirements for power supply isolation between inputs and outputs, it is recommended to use at least two power supplies, one for inputs and another one for outputs and Vdc.	
If the application does not require isolation between inputs and outputs, only one power supply could be used for inputs, outputs and Vdc.	
Inputs and outputs are optically isolated from each other.	

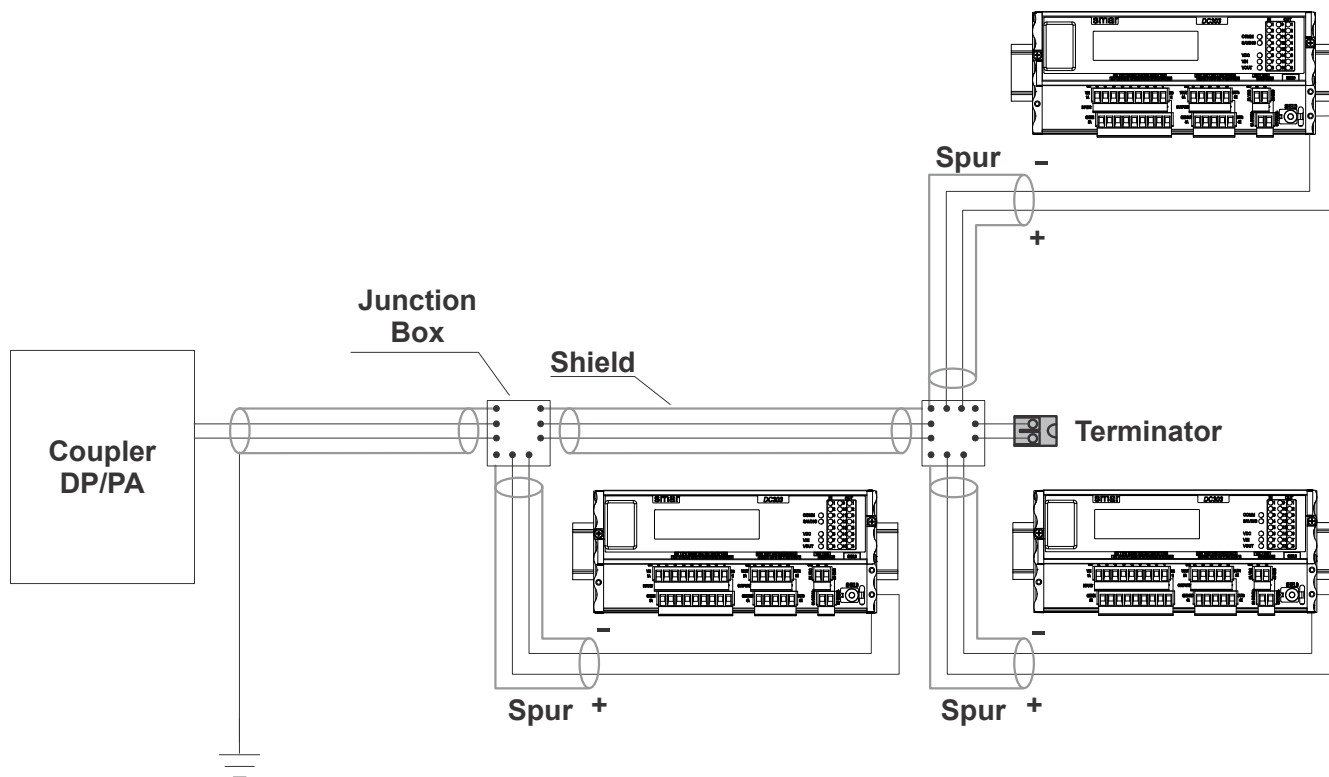


Figure 1.7 - Bus Topology

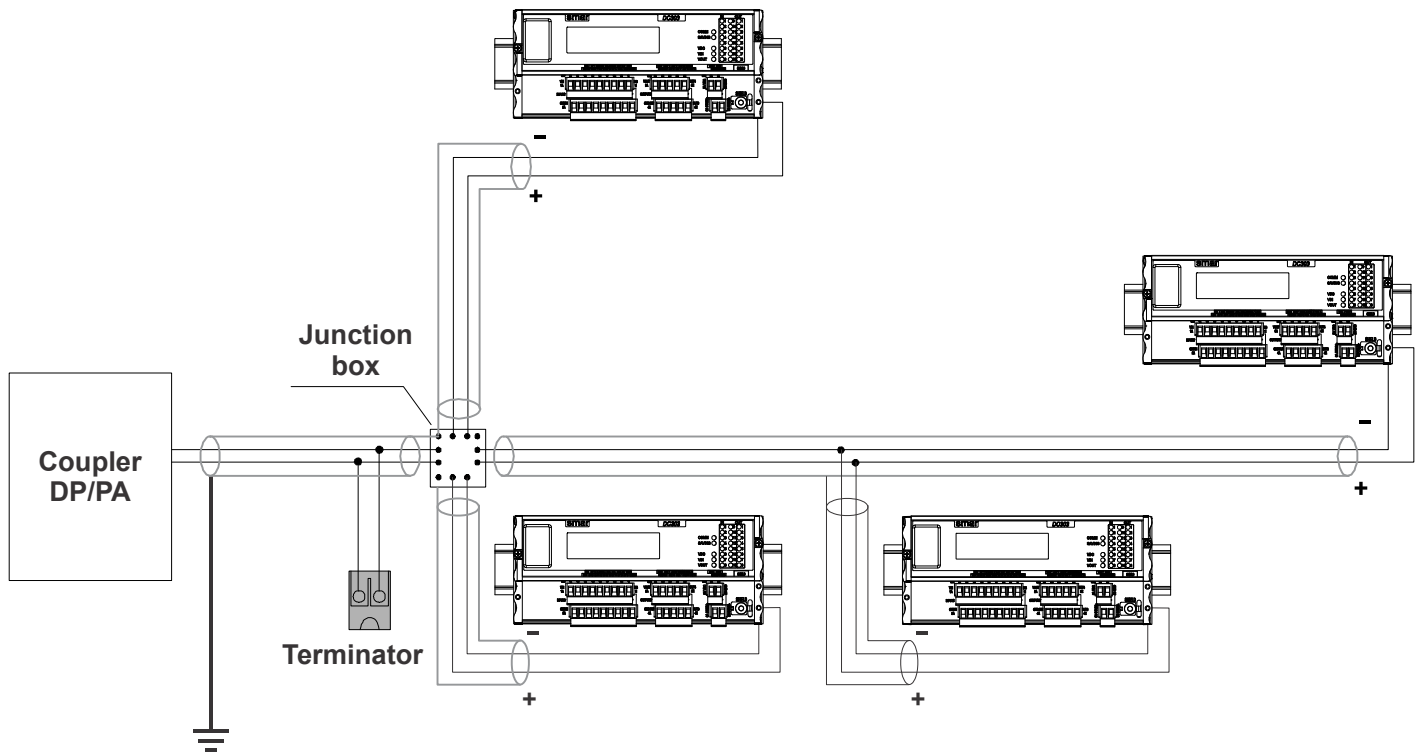


Figure 1.8 - Tree Topology

General System

According to the figure below, we can see a general network topology where the **DC303** is integrated in a simple Profibus network.

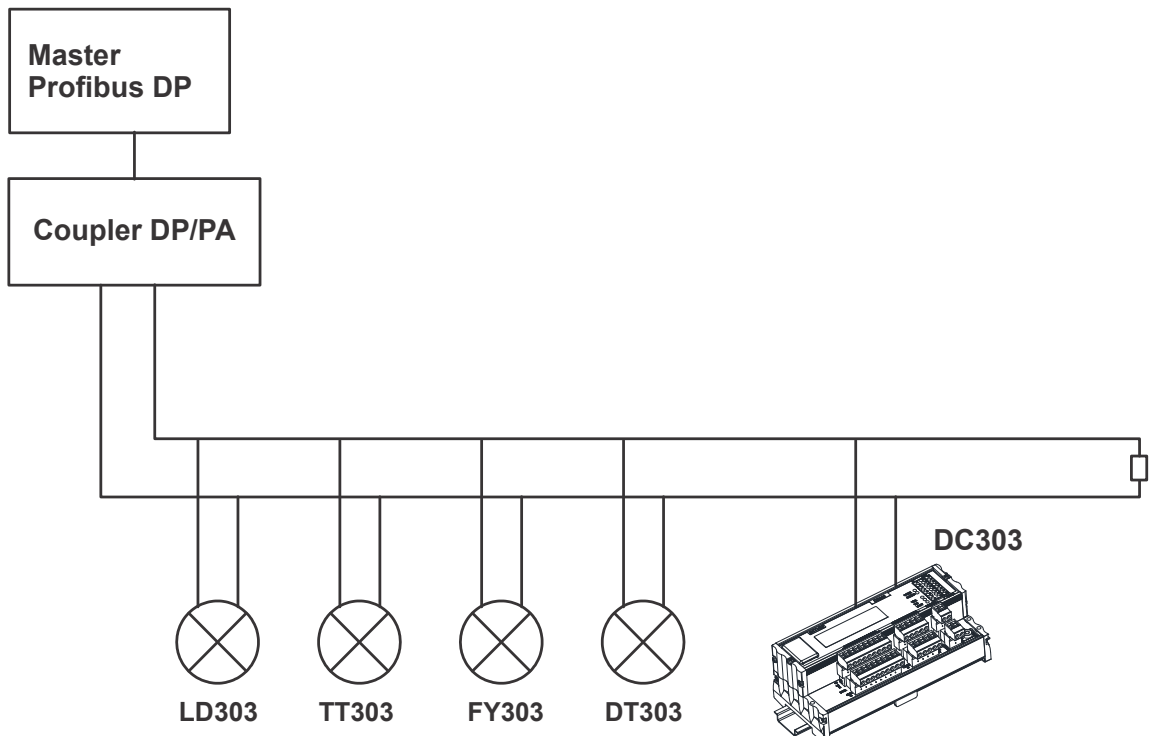


Figure 1.9 – DC303 and a general Profibus System

Section 2

OPERATION

The **DC303** accepts up to 16 optically isolated inputs and up to 8 open collector outputs. It is therefore ideal for interfacing existing discrete points to a Fieldbus system.

Function Blocks provide great flexibility to control strategies. The **DC303** has 16 DIs and 8 DOs.

The conventional discrete I/Os work together with Fieldbus devices integrated in the same network and in the same control loop.

Output function blocks include standard Profibus-PA safety mechanism in case of failures.

Inputs and Outputs are isolated from each other and are accessed via communication network through the function blocks channel. The leds are used to indicate the I/Os status. The use of Functional Blocks make the system homogeneous in a way that the device with conventional discrete and analog I/Os can be available in order to make the control strategies configuration easy.

Functional Description - Electronics

Refer to the block diagram (See Figure 2.1 – **DC303 Block Diagram**). The function of each block is described below.

(CPU) Central Processing Unit, FRAM

The CPU is the intelligent part of the **DC303**, being responsible for the management and operation of the execution block, self-diagnosis and communication. The program and the temporary data are stored in a FRAM memory. In the absence of energy, the data stored in the FRAM is not lost. The FRAM memory also stores the non-volatile data that will be used later. Examples of such data are: calibration, configuration and identification data.

Communication Controller

It monitors line activity, modulates and demodulates the signal from network line.

Power Supply

Takes power of the loop-line to power the Discrete Controller circuitry.

Factory Reset

The factory reset inscription can be found on the superior left side of the **DC303** housing. In order to accomplish this operation, simply short-circuit the contacts on the circuit board, turn-on the **DC303** in this short-circuit condition, and keep the key until the saving led goes to on state.

They are two mechanical contacts to perform the factory reset.

Input Latches

They are latches to hold the condition of inputs.

Output Latches

They are latches to hold the condition of outputs.

Optical Isolation

Optical isolation for inputs and outputs.

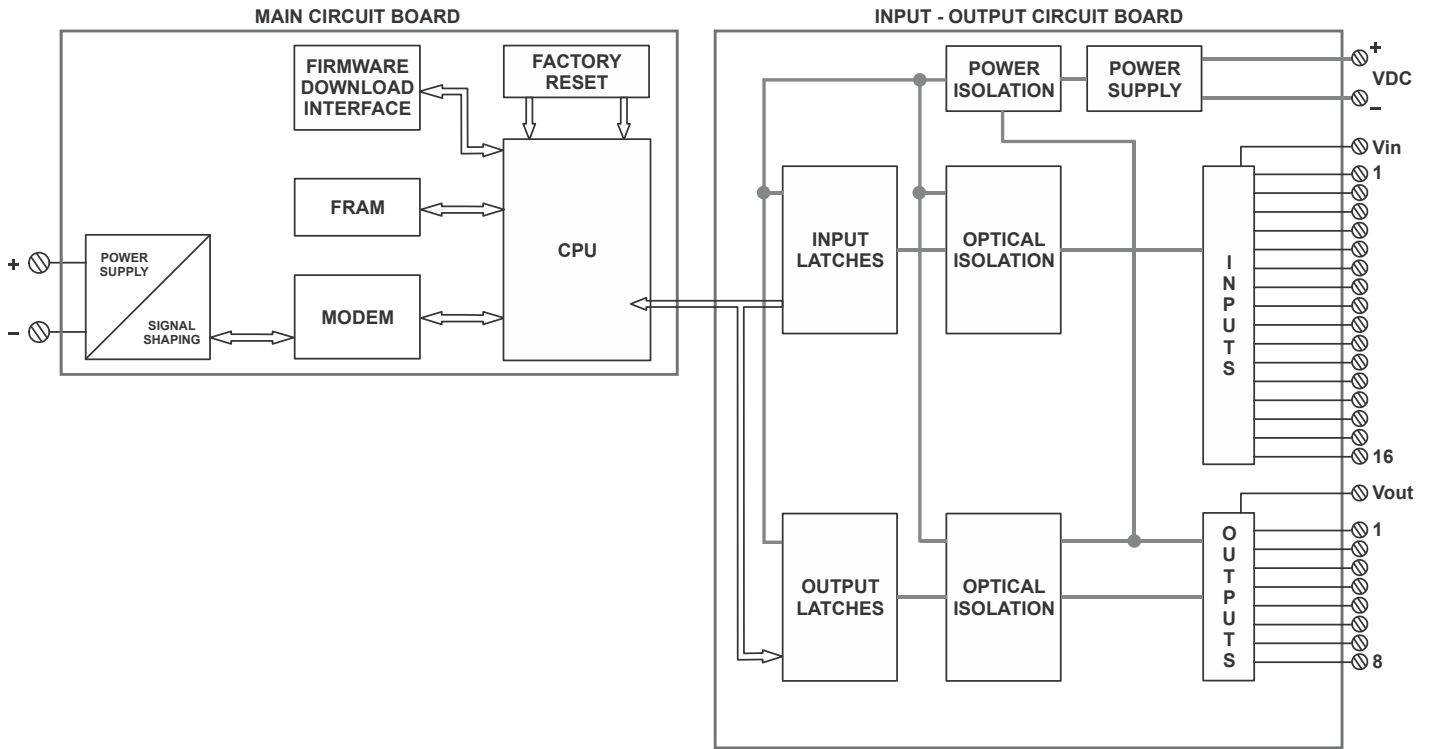


Figure 2.1 – DC303 Block Diagram

Section 3

CONFIGURATION

The **DC303** may be configured using SYSTEM302 from Smar or a third party configuration tool based on EDDL or FDT/DTM.

The **DC303** has DO (Discrete Output Block) and DI (Discrete Input Block). In addition, there is a built-in flexible function block to execute logics with Boolean resources, timers, counters, etc.

Function Blocks are not covered in this manual. For explanation and details of function blocks, see the *Function Blocks Manual*.

Connecting physical signals to Digital Input Block

The DI block takes the discrete input data, selected by channel number, and makes it available to other function blocks at its output.

For details, please see the Function Block Manual.

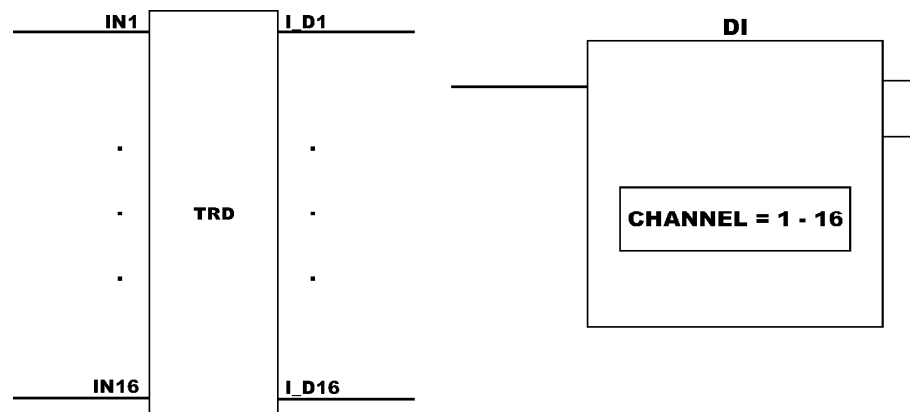


Figure 3.1 - DC303 and DI Block connections

Connecting physical signals to Digital Output Block

The DO block converts the value in SP_D to something useful for the hardware through the CHANNEL selection.

For details, please see the Function Blocks Manual.

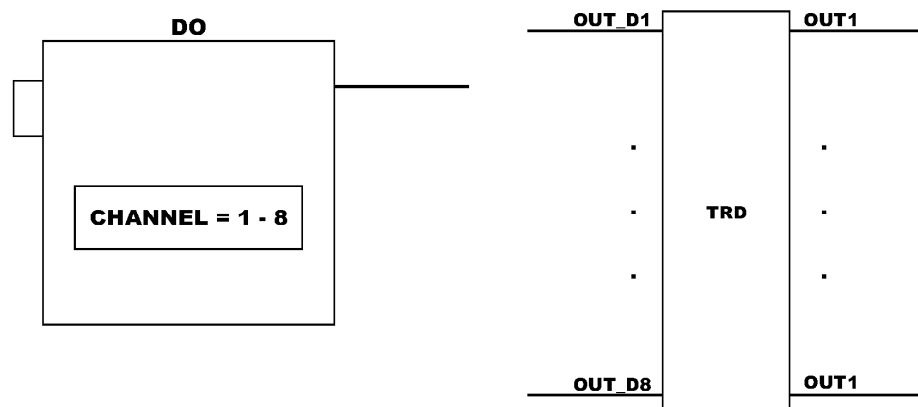


Figure 3.2 - DC303 and DO Block connections

Examples of Applications

Application 1: From the computer the input and output can be manipulated.

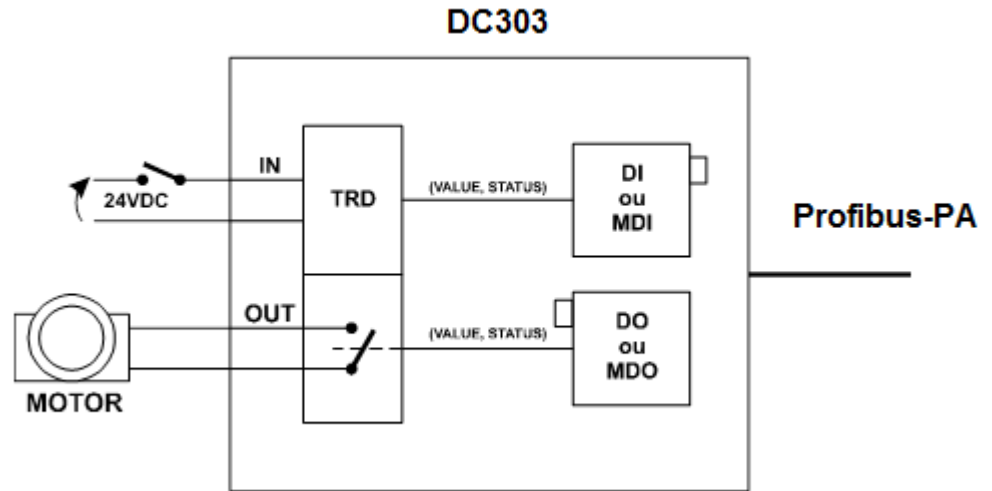


Figure 3.3 - DC303 – Application 1

Application 2: Distributed control (Level limit will start a motor, a pump, or open/close an on/off valve).

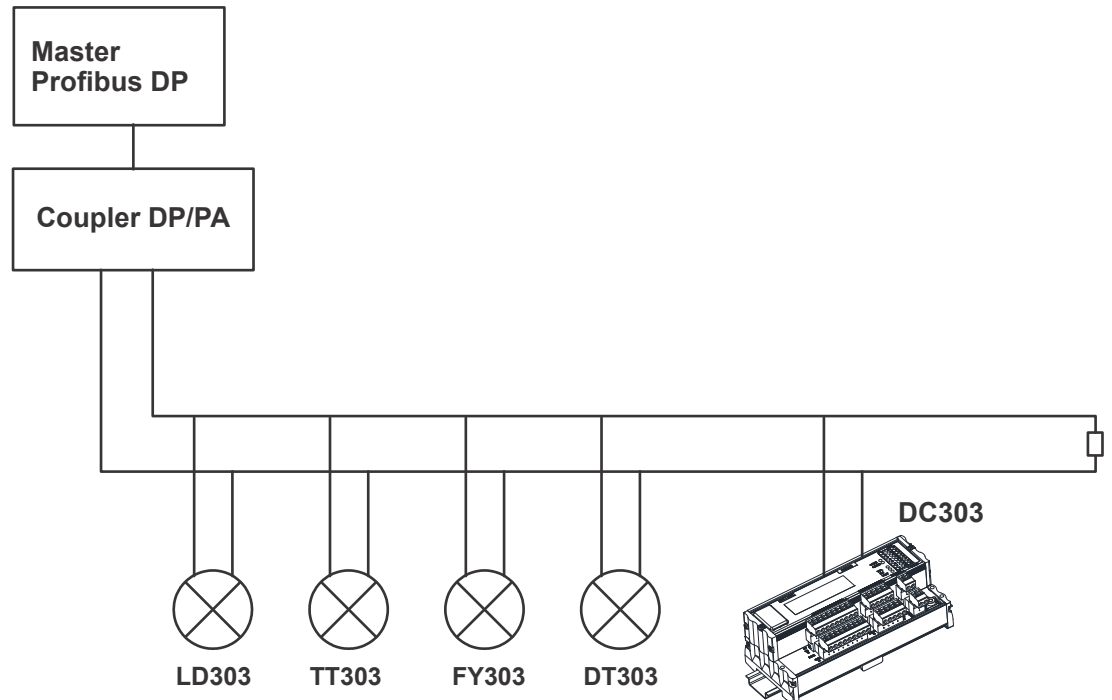


Figure 3.4- DC303 – Application 2

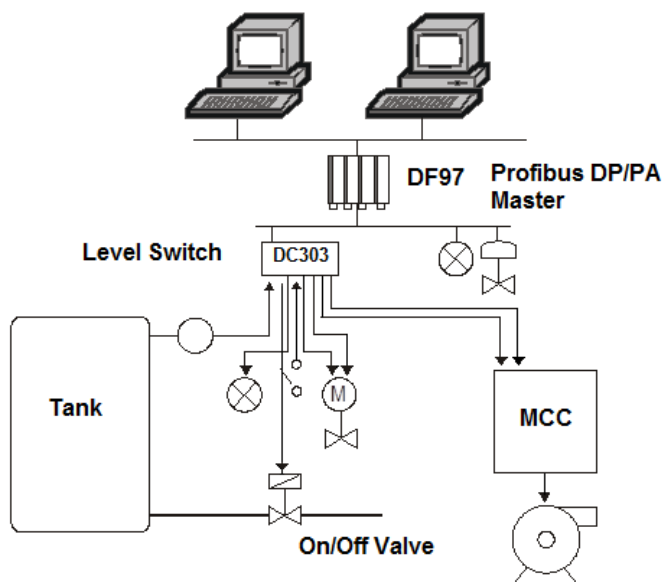
Application 3: General Application for Level and MCC control.

Figure 3.5 - DC303 – Application 3

Executing Logic with DC303

Description

The **DC303** is designed to execute logic. In this case the discrete outputs blocks (DOs) will not act physically on the hardware. The transducer block has a flexible built-in function block (FFB) that can work with up to 8 discrete inputs coming from the Profibus network via parameters SP_D from DOs blocks. On the transducer block these parameters are reference as IN_D1 to IN_D8. The transducer block can also provide 8 discrete outputs to the Profibus network, through the OUT_D8 OUT_D1 parameters that are available via OUT_D from the DI blocks (DI1 to DI8). The FFB can receive up to 16 discrete inputs via hardware and also provides 8 discrete outputs via hardware. In this situation the blocks DOs and DIs should be in automatic mode (block mode). When the FFB is enable in the transducer block (via parameter TRD_FFB_ENABLED), the blocks DI9 to DI16 are configured to "Out of Service".

When the FFB is disabled, the **DC303** works with 16 DI blocks and 8 DO blocks, reading its 16 discrete inputs and writing on their 8 hardware outputs, respectively.

Status indication for the inputs depends on the I/O subsystem.

The FFB block provides logic such as AND, OR, XOR and NOT and functions such as Timer On-Delay, Timer Off-Delay, Timer Pulse, Pulse Counter Down (CTD), Pulse Counter Up (CTU), RS Flip-Flop and SR Flip-Flop. The logic is done using the discrete inputs (IN_Dx) variables from Profibus network via DOs (SP_D), the output parameters for the Profibus network (OUT_Dx via DIs), the input discrete variables from hardware, the output discrete variables from hardware, the failsafe (FSx) values and the auxiliary bit variables (AUX's).

Status

The outputs status OUT_Dx will be according to:

- Input failure – Bad: Device Failure;
- Power up – Bad: Device Failure.

In the logic, a status that is greater or equal to 0x80 is considered to be true and a status that is less than 0x80 is considered to be false.

Supported Modes

O/S, and AUTO.

The changes on the Logic Lines and its parameters depend on the selection of CHANGE_OPTION

Schematic

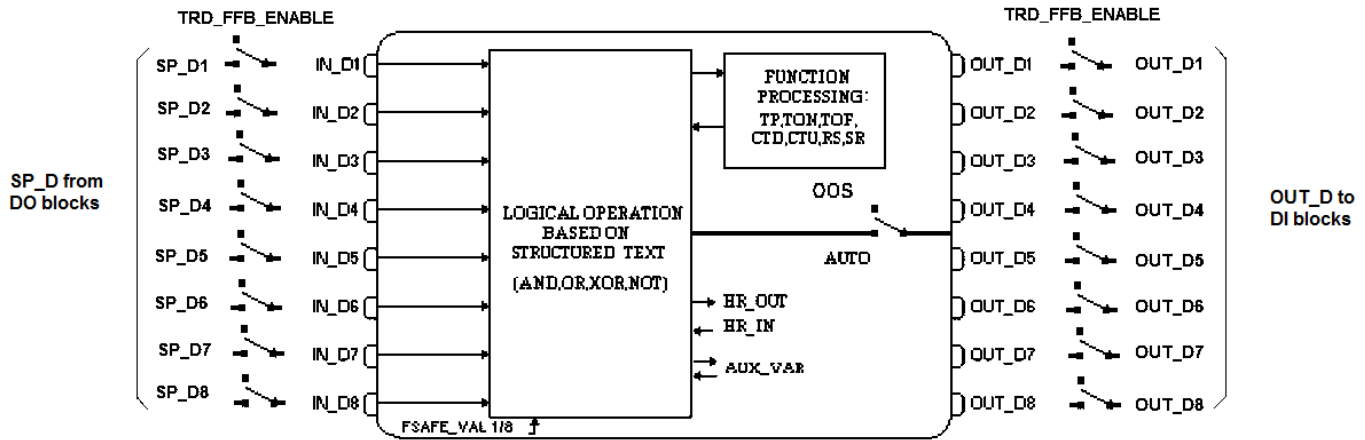


Figure 3.6 - Schematic

Parameters

Idx	Parameter	Data Type/ (Length)	Valid Range/ Options	Default Value	Units	Store/ Mode	Description
16	TRD_FFB_ENABLE	Unsigned8	0- Disabled ; 1- Enabled	0- Disabled		S	Allows the execution of FFB
17	IN_D1	DS-34				D	Discrete Input 1 for FFB. It comes from SP_D (DO1).
18	IN_D2	DS-34				D	Discrete Input 2 for FFB. It comes from SP_D (DO2).
19	IN_D3	DS-34				D	Discrete Input 3 for FFB. It comes from SP_D (DO3).
20	IN_D4	DS-34				D	Discrete Input 4 for FFB. It comes from SP_D (DO4).
21	IN_D5	DS-34				D	Discrete Input 5 for FFB. It comes from SP_D (DO5).
22	IN_D6	DS-34				D	Discrete Input 6 for FFB. It comes from SP_D (DO6).
23	IN_D7	DS-34				D	Discrete Input 7 for FFB. It comes from SP_D (DO7).
24	IN_D8	DS-34				D	Discrete Input 8 for FFB. It comes from SP_D (DO8).
25	FSTATE_VAL_D1	Unsigned8		0		S	The preset discrete value to use in failure for hardware output 1.
26	FSTATE_VAL_D2	Unsigned8		0		S	The preset discrete value to use in failure for hardware output 2.
27	FSTATE_VAL_D3	Unsigned8		0		S	The preset discrete value to use in failure for hardware output 3.
28	FSTATE_VAL_D4	Unsigned8		0		S	The preset discrete value to use in failure for hardware output 4.
29	FSTATE_VAL_D5	Unsigned8		0		S	The preset discrete value to use in failure for hardware output 5.
30	FSTATE_VAL_D6	Unsigned8		0		S	The preset discrete value to use in failure for hardware output 6
31	FSTATE_VAL_D7	Unsigned8		0		S	The preset discrete value to use in failure for hardware output 7.

Idx	Parameter	Data Type/ (Length)	Valid Range/ Options	Default Value	Units	Store/ Mode	Description
32	FSTATE_VAL_D8	Unsigned8		0		S	The preset discrete value to use in failure for hardware output 8.
33	OUT_D1	DS-34				D	The calculated discrete output variable 1 of the block in AUTO mode and copied to DI1
34	OUT_D2	DS-34				D	The calculated discrete output variable 2 of the block in AUTO and copied to DI2.
35	OUT_D3	DS-34				D	The calculated discrete output variable 3 of the block in AUTO mode and copied to DI3.
36	OUT_D4	DS-34				D	The calculated discrete output variable 4 of the block in AUTO mode and copied to DI4
37	OUT_D5	DS-34				D	The calculated discrete output variable 5 of the block in AUTO and copied to DI5
38	OUT_D6	DS-34				D	The calculated discrete output variable 6 of the block in AUTO mode and copied to DI6
39	OUT_D7	DS-34				D	The calculated discrete output variable 7 of the block in AUTO mode and copied to DI7
40	OUT_D8	DS-34				D	The calculated discrete output variable 8 of the block in AUTO mode and copied to DI8.
41	AUX_01_16	Bitstring(2)				D/ OS	Auxiliary bit enumerated variable 01_16.
42	AUX_17_32	Bitstring(2)				D/ OS	Auxiliary bit enumerated variable 17_32.
43	AUX_33_48	Bitstring(2)				D/ OS	Auxiliary bit enumerated variable 33_48.
44	AUX_49_64	Bitstring(2)				D/ OS	Auxiliary bit enumerated variable 49_64.
45	AUX_65_80	Bitstring(2)				D/ OS	Auxiliary bit enumerated variable 65_80.
46	AUX_81_96	Bitstring(2)				D/ OS	Auxiliary bit enumerated variable 81_96.
47	TON_PST	16 Floats	Positive	0	sec	S/ OS	Array of 16 float elements where the user can set the PST timer duration in seconds for each Timer ON Delay.
48	TON_CTA	16 Floats		0	sec	D	Array of 16 float elements where the user can read the lapsed time until the PST timer duration in seconds for each Timer ON Delay.
49	TON_OUT	Bitstring(2)				D	A bit enumerated that indicates the timer output states.
50	TOFF_PST	16 Floats	Positive	0	sec	S/ OS	Array of 16 float elements where the user can set the PST timer duration in seconds for each Timer OFF Delay.
51	TOFF_CTA	16 Floats		0	sec	D	Array of 16 float elements where the user can read the lapsed time until the PST timer duration in seconds for each Timer OFF Delay.
52	TOFF_OUT	Bitstring(2)				D	A bit enumerated that indicates the timer output states.
53	TP_PST	16 Floats	Positive	0	sec	S/ OS	Array of 16 float elements where the user can set the PST timer duration in seconds for each Timer Pulse.
54	TP_CTA	16 Floats		0	sec	D	Array of 16 float elements where the user can read the lapsed time until the PST timer duration in seconds for each Timer Pulse.
55	TP_OUT	Bitstring(2)				D	A bit enumerated that indicates the timer output states.
56	CTU_PST	16 Unsigned32	Positive	0	None	S/ OS	Array of 16 unsigned integer32 elements where the user can set the PST value of each pulse counter. The counter will increment from zero to PST value.

Idx	Parameter	Data Type/ (Length)	Valid Range/ Options	Default Value	Units	Store/ Mode	Description
57	CTU_CTA	16 Unsigned32		0	None	D	Array of 16 unsigned integer32 elements where the user can read the incremented value of each pulse counter.
58	CTU_OUT	Bitstring(2)				D	A bit enumerated that indicates the counter output states.
59	CTD_PST	16 Unsigned32	Positive	0	None	S/ OS	Array of 16 unsigned integer32 elements where the user can set the PST value of each pulse counter. PST is a preset value since the counter will decrement until zero.
60	CTD_CTA	16 Unsigned32		0	None	D	Array of 16 unsigned integer32 elements where the user can read the decremented value of each pulse counter.
61	CTD_OUT	Bitstring(2)				D	A bit enumerated that indicates the counter output states.
62	RS_OUT	Bitstring(2)				D	A bit enumerated that indicates the RS Flip-Flop output states.
63	SR_OUT	Bitstring(2)				D	A bit enumerated that indicates the SR Flip-Flop output states.
64	LOGIC_01	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 1.
65	LOGIC_02	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 2.
66	LOGIC_03	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 3.
67	LOGIC_04	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 4.
68	LOGIC_05	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 5.
69	LOGIC_06	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 6.
70	LOGIC_07	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 7.
71	LOGIC_08	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 8.
72	LOGIC_09	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 9.
73	LOGIC_10	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 10.
74	LOGIC_11	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 11.
75	LOGIC_12	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 12.
76	LOGIC_13	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 13.
77	LOGIC_14	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 14.
78	LOGIC_15	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 15.
79	LOGIC_16	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 16.
80	LOGIC_17	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 17.
81	LOGIC_18	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 18.
82	LOGIC_19	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 19.
83	LOGIC_20	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 20.
84	LOGIC_21	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 21.
85	LOGIC_22	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 22.
86	LOGIC_23	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 23.
87	LOGIC_24	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 24.
88	LOGIC_25	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 25.
89	LOGIC_26	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 26.
90	LOGIC_27	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 27.
91	LOGIC_28	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 28.
92	LOGIC_29	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 29.
93	LOGIC_30	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 30.
94	LOGIC_31	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 31.
95	LOGIC_32	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 32.
96	LOGIC_33	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 33.

Idx	Parameter	Data Type/ (Length)	Valid Range/ Options	Default Value	Units	Store/ Mode	Description
97	LOGIC_34	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 34.
98	LOGIC_35	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 35.
99	LOGIC_36	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 36.
100	LOGIC_37	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 37.
101	LOGIC_38	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 38.
102	LOGIC_39	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 39.
103	LOGIC_40	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 40.
104	LOGIC_41	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 41.
105	LOGIC_42	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 42.
106	LOGIC_43	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 43.
107	LOGIC_44	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 44.
108	LOGIC_45	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 45.
109	LOGIC_46	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 46.
110	LOGIC_47	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 47.
111	LOGIC_48	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 48.
112	LOGIC_49	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 49.
113	LOGIC_50	VisibleString(24)		Spaces	Na	S/ OS	Logical Line Command 50.
114	LOGIC_CHECK	Unsigned8	0 - Enable., 1 - Checked. 2- Changed but not checked yet.	1 - Checked.	Na	D/OS	Allows the check for logic line.
115	ERROR_LINE	Unsigned8	0-50	1	Na	S	Indicates the logic line where there is an error.
116	ERROR_CODE	Unsigned8	0 - Logic Ok. 1 - Exceed String Length or string not valid. 2 - Non valid operand. 3 - No implemented logic or missing ',' 4 - Missing parentheses or argument not valid. 5 - Non valid resource. 6 - Argument not valid. 7 - Function not valid 8 - Non available resource. 9 - Non valid attribution. 10 - First Argument not valid. 11- Second Argument not valid.	3 - No implemented logic or missing ','	Na	S	Indicated the code for the error in the logic line.
117	CHANGE_OPTION	Unsigned8	0 - Logic parameter changes are only allowed in Out of Service. 1 - Always accept Logic parameter changes.	0 - Logic parameter changes are only allowed in Out of Service.	Na	S	Enable logic parameter changes independent of Mode Block parameter

The following table describes the Logic Operation and Command line and the correspondent Symbols used in the logic line:

Logic Operation and Command line	Symbol - description
AND	&
OR	
XOR	^
NOT	!
EQUAL	=
(arg1,arg2)	To define function arguments
;	End of logic line

The logic does NOT (!) work only with simple variables. Example:
 OUT1=!IN1;

Note that it is not allowed to have, for example,
 OUT1=!TP01(IN1);
 To use it this way, we should have:
 A01= TP01(IN1);. -> OUT1=!A01;

The logic is always executed line by line and from left to right in the logic line. Spaces are not allowed between the characters. **Empty lines are not allowed between logic lines and the implementation of logic lines must be in sequence.**

After writing the logic into the LOGIC_XX (XX:01 -> XX:50) parameters, the user needs to select the option "Enable" in the parameter LOGIC_CHECK in order to verify the errors. **When the logic is configured using the downloading process, it is necessary to configure first the LOGIC_XX (XX:01 -> XX:50) parameters and then the LOGIC_CKECK parameter. This sequence is fundamental to performing the check.**

The following table shows the mnemonic for each block parameter used in the logic lines. The mnemonic must be in capital letters:

Parameter	Mnemonic
HW_IN.Value1	I01
HW_IN.Value2	I02
HW_IN.Value3	I03
HW_IN.Value4	I04
HW_IN.Value5	I05
HW_IN.Value6	I06
HW_IN.Value7	I07
HW_IN.Value8	I08
HW_IN.Value9	I09
HW_IN.Value10	I10
HW_IN.Value11	I11
HW_IN.Value12	I12
HW_IN.Value13	I13
HW_IN.Value14	I14
HW_IN.Value15	I15
HW_IN.Value16	I16
HW_IN.Status	SI
HW_OUT.Status	SO
HW_OUT.Value1	O1
HW_OUT.Value2	O2
HW_OUT.Value3	O3
HW_OUT.Value4	O4
HW_OUT.Value5	O5

Parameter	Mnemonic
HW_OUT.Value6	O6
HW_OUT.Value7	O7
HW_OUT.Value8	O8
IN_D1.Status	IN1S
IN_D2.Status	IN2S
IN_D3.Status	IN3S
IN_D4.Status	IN4S
IN_D5.Status	IN5S
IN_D6.Status	IN6S
IN_D7.Status	IN7S
IN_D8.Status	IN8S
IN_D1.Value	IN1
IN_D2.Value	IN2
IN_D3.Value	IN3
IN_D4.Value	IN4
IN_D5.Value	IN5
IN_D6.Value	IN6
IN_D7.Value	IN7
IN_D8.Value	IN8
OUT_D1.Status	SOUT1
OUT_D2.Status	SOUT2
OUT_D3.Status	SOUT3
OUT_D4.Status	SOUT4
OUT_D5.Status	SOUT5
OUT_D6.Status	SOUT6
OUT_D7.Status	SOUT7
OUT_D8.Status	SOUT8
OUT_D1.Value	OUT1
OUT_D2.Value	OUT2
OUT_D3.Value	OUT3
OUT_D4.Value	OUT4
OUT_D5.Value	OUT5
OUT_D6.Value	OUT6
OUT_D7.Value	OUT7
OUT_D8.Value	OUT8
FSTATE_VAL_D1	FS1
FSTATE_VAL_D2	FS2
FSTATE_VAL_D3	FS3
FSTATE_VAL_D4	FS4
FSTATE_VAL_D5	FS5
FSTATE_VAL_D6	FS6
FSTATE_VAL_D7	FS7
FSTATE_VAL_D8	FS8
AUX_01_16	A01-A16
AUX_17_32	A17-A32
AUX_33_48	A33-A48
AUX_49_64	A49-A64
AUX_65_80	A65-A80
AUX_81_96	A81-A96

Parameter	Mnemonic
TON	TON01-TON16
TOFF	TOF01-TOF16
TP	TP01-TP16
CTU	CTU01-CTU16
CTD	CTD01-CTD16
RS	RS01-RS16
SR	SR01-SR16

Functions

For each type of function there are 16 available resources and the user can use only once each resource. To use the function results, the user can make attribution for auxiliary bits.

TP TIMER PULSE

This function generates a fixed time pulse in the output timer for every rising (false to true transition) on the input timer. The pulse width is determined by TP_PST parameter in seconds. Transitions in the input timer will be ignored while the pulse is active. The current time is available in the TP_CTA parameter.

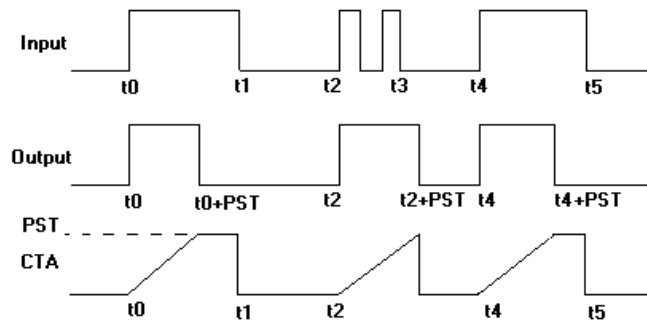


Figure 3.7 - Timer Pulse Function – timing diagrams

The syntax for Timer Pulse is: **TPxx(arg)**

Where, xx is the used resource from 01 to 16 and arg is the function argument and it must be a simple variable. Examples:

```
O1=TP01(IN1);
OUT1= TP01(A05);
OUT3=TP08(FS1);
```

For example, the following examples are not allowed in the logic line:

O1=TP01(IN1&IN2);: note that the argument is a result of an operation, it is not allowed.

O1=TP10(!IN1);: note that the argument is a result of NOT function, it is not allowed.

O1=TP10(CTD01(IN1,IN2));: note that the argument is a result of a function, it is not allowed.

TON TIMER ON-DELAY

This function delays the timer output of going to true for a period of time after the input has moved to true. This period is configured by TON_PST parameter in seconds. If the input goes to false before the PST time, the output timer will remain in false. The CTA parameter will show the remainder time until PST value.

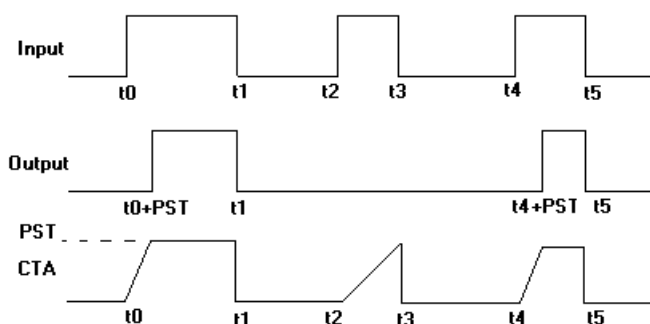


Figure 3.8 - Timer On-Delay Function – timing diagrams

The syntax for Timer On-Delay is: TONxx(arg)

Where, xx is the used resource from 01 to 16 and arg is the function argument and it must be a simple variable . Examples:

```
O1=TON01(IN1)&SI;
OUT1= TON01(A05);
OUT3=TON08(FS1);
```

For example, the following examples are not allowed in the logic line:

- O1=TON01(IN1&IN2);: note that the argument is a result of an operation, it is not allowed.
- O1=TON10(!IN1);: note that the argument is a result of NOT function, it is not allowed.
- O1=TON10(CTD01(IN1,IN2));: note that the argument is a result of a function, it is not allowed.

TOF TIMER OFF-DELAY

This function extends the true state of timer input for a determined period of time for the output timer. This period is configured by TOF_PST parameter in seconds. If the input goes to true before the out goes to false, the out will stay on true and the time period will begin to count again at the moment when the input goes to false. The CTA parameter will show the remainder time until PST value.

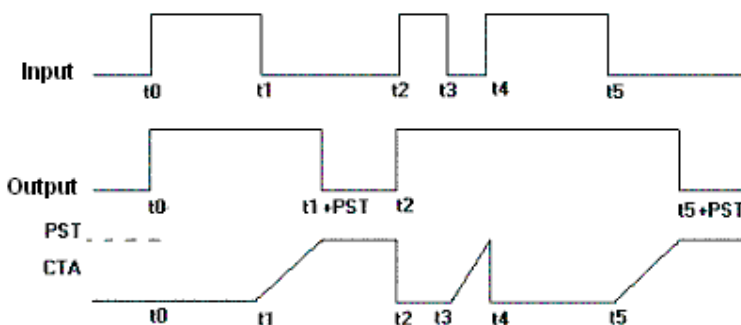


Figure 3.9 - Timer OFF-Delay Function – timing diagrams

The syntax for Timer Off-Delay is: TOFxx(arg)

Where, xx is the used resource from 01 to 16 and arg is the function argument and it must be a simple variable .Examples:

```
O1=TOF01(IN1)&SI;
OUT1= TOF01(A05);
OUT3=TOF08(FS1);
```

- For example, the following examples are not allowed in the logic line:
- O1=TOF01(IN1&IN2);: note that the argument is a result of an operation, it is not allowed.
- O1=TOF10(!IN1);: note that the argument is a result of NOT function, it is not allowed.
- O1=TOF10(CTD01(IN1,IN2));: note that the argument is a result of a function, it is not allowed.

CTD PULSE COUNTER DOWN

This function is used to count rising transitions (from false to true) in the counter input(arg1). Every time it is seeing a rising transition the internal counter accumulator (CTA) decrements of one. When the CTA reaches zero the counter output will go to true. The counter value will be preset for PST. A transition from false to true in the second argument(arg2) presets the counter.

The syntax for CTD is: CTDxx(arg1,arg2)

Where, xx is the used resource from 01 to 16 and arg1 and arg2 are the function arguments and they must be simple variables. Examples:

```
O3=CTD10(IN1,IN2);
OUT1=CTD03(A11,A14)&SI;
```

For example, the following examples are not allowed in the logic line:

O1=CTD01(IN1&IN2,IN3);: note that the argument is a result of an operation, it is not allowed.

O1=CTD10(!IN1,IN3);: note that the argument is a result of NOT function, it is not allowed.

O1=CTD10(TP01(IN1),IN2);: note that the argument is a result of a function, it is not allowed.

CTU PULSE COUNTER UP

This function is used to count rising transitions (from false to true) in the counter input(arg1). Every time it is seeing a rising transition the internal counter accumulator (CTA) increments of one. When the CTA reaches the preset value PST, the counter output will go to true. A transition from false to true in the second argument(arg2) resets the counter.

The syntax for CTU is: CTUxx(arg1,arg2)

Where, xx is the used resource from 01 to 16 and arg1 and arg2 are the function arguments and they must be simple variables. Examples:

```
O3=CTU10(IN1,IN2);
OUT1=CTU03(A11,A14)&SI;
```

For example, the following examples are not allowed in the logic line:

O1=CTU01(IN1&IN2,IN3);: note that the argument is a result of an operation, it is not allowed.

O1=CTU10(!IN1,IN3);: note that the argument is a result of NOT function, it is not allowed.

O1=CTU10(TP01(IN1),IN2);: note that the argument is a result of a function, it is not allowed.

RS FLIP-FLOP

This function has the following operation table:

R(arg1)	S(arg2)	OUT
0	0	Last state
0	1	1
1	0	0
1	1	0

The syntax for RS Flip-Flop is: RSxx(arg1,arg2)

Where, xx is the used resource from 01 to 16 and arg1 and arg2 are the function arguments and they must be simple variables. Examples:

```
O3=RS10(IN1,IN2);
OUT1=RS03(A11,A14)&SI;
```

For example, the following examples are not allowed in the logic line:

O1=RS01(IN1&IN2,IN3);: note that the argument is a result of an operation, it is not allowed.

O1=RS10(!IN1,IN3);: note that the argument is a result of NOT function, it is not allowed.

O1=RS10(TP01(IN1),IN2);: note that the argument is a result of a function, it is not allowed.

SR FLIP-FLOP

This function has the following operation table:

S(arg1)	R(arg2)	OUT
0	0	Last state

S(arg1)	R(arg2)	OUT
0	1	0
1	0	1
1	1	1

The syntax for SR Flip-Flop is: SRxx(arg1,arg2)

Where, xx is the used resource from 01 to 16 and arg1 and arg2 are the function arguments and they must be simple variables. Examples:

```
O3=SR10(IN1,IN2);
OUT1=SR03(A11,A14)&SI;
```

For example, the following examples are not allowed in the logic line:

O1=SR01(IN1&IN2,IN3);: note that the argument is a result of an operation, it is not allowed.

O1=SR10(!IN1,IN3);: note that the argument is a result of NOT function, it is not allowed.

O1=SR10(TP01(IN1),IN2);: note that the argument is a result of a function, it is not allowed.

Error Code

Some examples of error conditions:

Error Code: "Exceed String Length or string not valid."

```
a) OUT1=IN1&IN2&IN2|IN4^IN5|IN6;
```

Note that they are 29 characters on the string and the maximum allowed is 24.

```
b) OUT1=IN1&in2;
```

Note that the logic is case sensitive. All characters must be in capital letters.

Error Code: "Non valid operand."

```
OUT1=IN1%IN2;
```

Note that the % is not allowed. See the table that describes the Logic Operation and Command line.

Error Code: "No implemented logic or missing ';' ."

```
OUT1=IN1
```

Note that the "; " is missing at the end of the logic line.

Error Code: "Missing parentheses or argument not valid."

```
OUT1=TP10(IN1;
```

Note that the parentheses is missing in the timer pulse function.

Error Code: "Non valid resource."

```
OUT1=TP18(IN1);
```

Note that there are 16 resources for each function

Error Code: "Argument not valid."

```
OUT1=TP10(IN10);
```

Note that there are only 8 inputs. IN10 is not a valid argument.

Error Code: "Function not valid."

```
OUT1=TR10(IN1);
```

Note that TR is not a valid function.

Error Code: "Non available resource."

```
OUT1=TP10(IN1);
```

```
A03=TP10(IN7);
```

Note that there are 16 resources for each function. The resource 10 for the timer has already been used and can not be used again. However, the result of the function can be assigned to an auxiliary variable and this auxiliary variable can be used several times.

```
A03=TP10(IN7);
```

```
A03=TP10(IN7);
```

Error Code: "Non valid attribution."

```
IN1=IN2^TP03(IN4);
```

Note that is not allowed attribution to inputs.

Error Code: "First Argument not valid."

OUT1=CTD01(!IN1,IN2);

Note that the arguments are necessarily simple variables and not functions.

OUT1=RS11(IN15,IN2);

Note that the first argument is not allowed.

Error Code: "Second Argument not valid."

a) OUT1=CTD01(IN1,!IN2);

Note that the arguments are necessarily simple variables and not functions.

b) OUT1=RS11(IN1,IN20);

Note that the second argument is not allowed.

Example of applications

- 1) According to the next figure, we have an industrial application where the aim is to fill up the bottles with a chemical fluid. The conveyor moves the bottles up to the filling direction and then the bottle is detected by a sensor. The conveyor must stop and open the valve of filling and the level is detected by another sensor. After detecting the level, the system must wait for 10 seconds and then move the conveyor again until the next bottle.

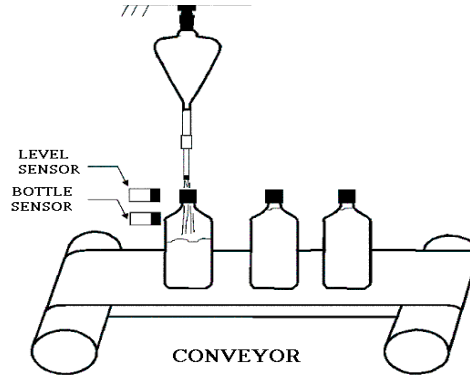


Figure 3.10 - Filling of bottles with a chemical fluid

Using the Flexible Function Block we have the following definitions:

- The conveyor will be turned on using the hardware output 01 (O1);
- The fluid valve will be turned on using the hardware output 02 (O2);
- The bottle sensor will be connected to the hardware input 01 (I01);
- The level sensor will be connected to the hardware input 02 (I02);
- The power system will be connected to the hardware input 03 (I03);

We have the following configuration:

TON_PST resource [01] = 10.0s.

LOGIC_01 A01=TON01(I02);

LOGIC_02 O1=I03&!I01|A01;

LOGIC_03 O2=I01&!I02;

Making an analogy to ladder programming, we have:

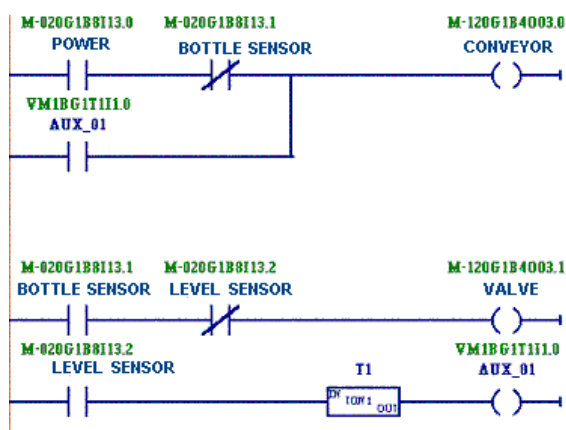


Figure 3.11 - Analogy to ladder programming

- 2) In the following application we have the control of steps to operate an electro-mechanical balance, that weights phosphatic stone .

The weight process is done by boat-load, the system executes one full weight cycle each interval time of 20 seconds. See the following figure:

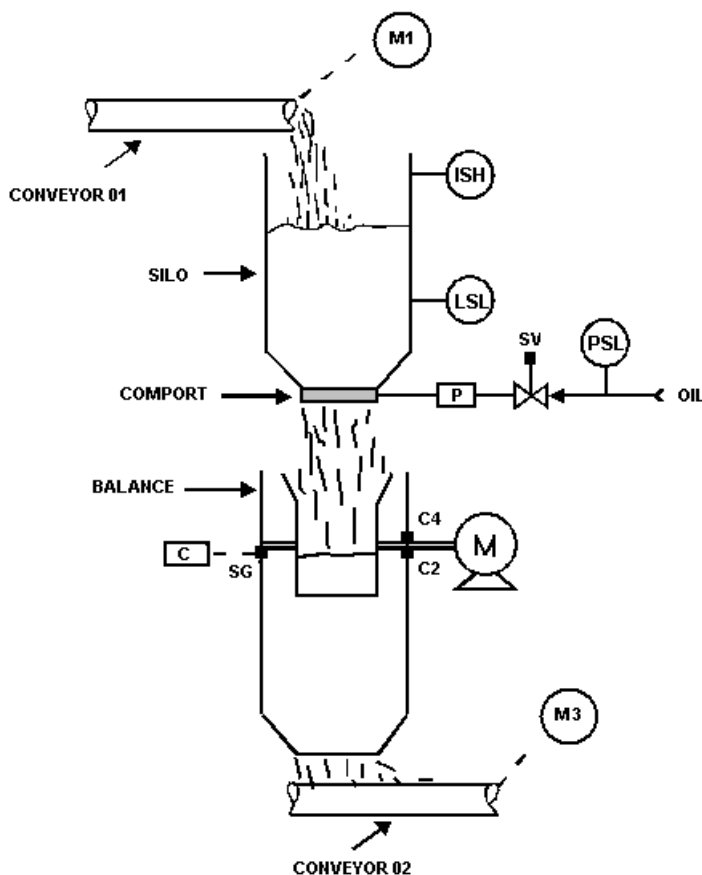


Figure 3.12 – Application with an electro-mechanical balance

- M1 and M3 - Motors for the conveyors
- C2 and C4 - Limit Switches
- LSH - High Level Sensor
- LSL - Low Level Sensor
- SG - Load Cell
- SV - Solenoid Valve

- M - Bucket Motor
- P - Comport Piston
- C - Weight Circuit

Process:



The system requires the following conditions to startup:

- The phosphatic stone level (LSL non activated);
- Oil Pressure (PSL on);
- Conveyor 02 active (M3 on);
- Bucket in initial position (C4 on);



After the initial conditions, we note:

- By activating the power switch, the comport opens, and the bucket starts to get loaded;
- After reaching the desired weight, the comport closes. After 5 seconds, the bucket rotates 180° and unload the product into the conveyor 02.

Note:

-  This new position will be detected by C2 and after 5 seconds, the bucket will have to return to initial position, which will be detected by C4.
-  After the bucket returns to the initial position, a new weight cycle begins.

Comment:

-  The operation sequence must be stopped if any requirement is not satisfied.
-  The silo comport is activated by a hydraulic piston.

Using the Flexible Function Block we have the following definitions:

- LSL will be connected to the hardware input 01 (I01);
- LSH will be connected to the hardware input 02 (I02);
- PSL will be connected to the hardware input 03 (I03);
- C2 will be connected to the hardware input 04 (I04);
- C4 will be connected to the hardware input 05 (I05);
- Power will be connected to the hardware input 06 (I06);
- M3 will be connected to the hardware input 07 (I07);
- M will be activated by hardware output 01 (O1);
- The Comport will be activated by hardware output 02 (O2);
- M1 will be activated by hardware output 03 (O3);

We have the following configuration:

```

TON_PST resource [01] = 5.0s.
LOGIC_01    A01=!I01&I03&I07&I05;
LOGIC_02    A02=I06&RS01(I02,I01);
LOGIC_03    O3=A02&I03;
LOGIC_04    A03=I03&I07;
LOGIC_05    O2=I06&A03&I04;
LOGIC_06    O1=TON01(I04)&!I05&A03;
    
```

3) Using Fault-State values:

Lets suppose we have the following condition:

- A01 receives the logic between the status for discrete inputs as follows:
A01=IN1S&IN2S;
when the status is bad for one of these inputs, then, A01=false(0),
otherwise, A01=true (1);
- FS1 is the fault-state value for O1;
- A02 is the bit containing the logic for O1;

We have the following table between the FS1, A01 and A02:

FS1	A01	A02	O1
0	0	0	0
0	0	1	0
0	1	0	0

FS1	A01	A02	O1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Then,

```
A03=!FS1&A01&A02;
A04=FS1&!A01&!A02;
A05=FS1&!A01&A02;
A06=FS1&A01&A02;
O1=A03|A04|A05|A06;
```

DC303 Cyclical Configuration

PROFIBUS-DP as well as PROFIBUS-PA foresees protocol mechanisms against communication failures and errors and, as an example, during the initialization, several errors sources are verified. After the power up the field equipments (slaves) are ready for the cyclical data exchange with the Class1 master, but, for that, the master parameterization for the correspondent slave must be correct. This information is obtained through the GSD files, which should be one for each device. Through the commands below, the master executes every initialization process with PROFIBUS-PA devices:

- **Get_Cfg**: carries the slaves' configuration and verifies the net configuration;
- **Set_Prm**: writes in the slaves' parameters and executes net parameterization services;
- **Set_Cfg**: configures the slaves according to inputs and outputs;
- **Get_Cfg**: a second command, where the master will verify the slaves' configuration.

All these services are based on the information obtained of GSD slaves' files.

The GSD file of **DC303** presents details of hardware revision and software, bus timing of the device and information on cyclical data exchange.

The **DC303** has 16 DI and 8 DO blocks.

Most of the PROFIBUS configurators use 2 directories. These directories must have the GSD's and bitmap's files of several manufacturers.

The GSD and bitmap's files for Smar devices can be purchased via internet in www.smar.com.

See below a typical example with the necessary steps to the integration of a **DC303** device in a PA system and that can be extended for any device:

- Copy the GSD file of the device for the search directory of the PROFIBUS configurator, usually named GSD.
- Copy the bitmap file of the device for the search directory of the PROFIBUS configurator, usually named BMP.
- Once the master is chosen, the communication rate must be chosen, remembering that when we had the couplers, we can have the following rates: 45.45 kbits/s (Siemens), 93.75 kbits/s (P+F) and 12 Mbits/s (P+F, SK2) .If we had the link device, it can be up to 12 Mbits/s.
- Add the **DC303**, specifying the address in the bus.
- Choose the cyclical configuration via parameterization with the GSD file, dependent of the application. For each DO and DI block, the **DC303** provides two bytes to the Profibus DP Master, one the discrete value and one status byte.
- Please see the cyclic options for **DC303**:

```
;Empty module
Module = "EMPTY_MODULE"           0x00 ;
EndModule
;
```

```

;
;Modules for Discrete Output Block
Module = "SP_D"                0xA1 ;

EndModule
Module = "SP_D+RB_D"           0xC1, 0x81, 0x81, 0x83 ;

EndModule
Module = "SP_D+CB_D"           0xC1, 0x81, 0x82, 0x92 ;

EndModule
Module = "SP_D+RB_D+CB_D"      0xC1, 0x81, 0x84, 0x93 ;

EndModule
Module = "RIN_D+ROUT_D"        0xC1, 0x81, 0x81, 0x8C ;

EndModule
Module = "RIN_D+ROUT_D+CB_D"   0xC1, 0x81, 0x84, 0x9C ;

EndModule
Module = "SP_D+RB_D+RIN_D+ROUT_D+CB_D"  0xC1, 0x83, 0x86, 0x9F ;

EndModule

;Modules for Discrete Input Block

Module = "OUT_D"                0x91 ;

EndModule

```

- The watchdog condition can also be activate, where after the communication loss detection for the slave device with the master, the equipment can change to a fail-safe condition.

The **DC303** also has the Empty module for application where is not necessary all function blocks. There is an order for cyclic communication as follow:
DO_1, DO_2,...DO_8, DI_, DI_2, ...DI_16.

For example, where is necessary to work only with DOs blocks, we have:
DO_1, DO_2, DO_3, DO_4, DO_5, DO6, DO_7, DO_8, EMPTY_MODULE, EMPTY_MODULE,
EMPTY_MODULE, EMPTY_MODULE, EMPTY_MODULE, EMPTY_MODULE, EMPTY_MODULE,
EMPTY_MODULE.

Suppose now the application will work with DOs blocks and only DI2:
DO_1, DO_2, DO_3, DO_4, DO_5, DO6, DO_7, DO_8, EMPTY_MODULE, DI_2,
EMPTY_MODULE, EMPTY_MODULE, EMPTY_MODULE, EMPTY_MODULE, EMPTY_MODULE,
EMPTY_MODULE.

If the DO blocks are in AUTO, then the device will receive the value and status of the discrete setpoint of the class 1 master and the user will also be able to write in this value via class 2 master. In this case, the setpoint status should always be equal to 0x80 ("good") and the following configurations can be chosen:

- SP_D
- SP_D+RB_D
- SP_D+RB_D+CB_D

If the DO blocks are in RCAS, then the device will receive the value and status of the discrete setpoint only via class 1 master. In this case, the setpoint status should always be equal to 0xc4 ("IA"). The following configurations can be chosen:

- SP_D
- SP_D+RB_D
- SP_D+RB_D+CB_D
- RIN_D+ROUT_D
- RIN_D+ROUT_D+CB_D
- SP_D+RB_D+RIN_D+ROUT_D+CB_D

Cyclical Diagnosis

Via cyclic communication is possible to verify diagnostics from the **DC303** using the Profibus Master Class 1 or even via acyclic communication via Master Class 2. The Profibus-PA devices provide up to 4 standard diagnoses bytes via Physical Block (See fig. 3.13. and 3.14) and when the most significant bit of the fourth Byte is "1", the diagnose will extend the the information in more 6 bytes. These Diagnosis bytes can also be monitored via cyclic tools.

Len of status bytes	Status Type	Physical Block Slot	Status		From Physical Block	
			Appears	Dissapears	Standard Diagnostic	Extended Diagnostic
08 - Standard Diag 0E - Ext Diag	FE	01	01 - Appears	02- Dissapears	4 bytes	6 bytes vendor specific

When bit 55 (byte 4, MSB) is "1":
the device has extended diagnostic

Figure 3.13 – Cyclical Diagnosis

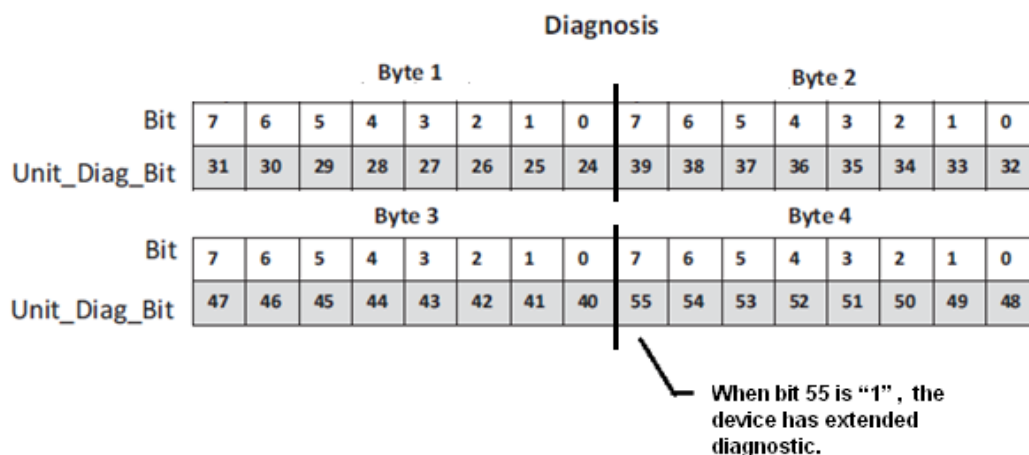


Figure 3.14 – Cyclic Diagnosis mapping for 4 bytes of Physical Block.

Unit_Diag_bit is described in the GSD file Profibus-PA device.

See below a description part of a GSD file for the 4 bytes and more detail:

```

;----- Description of device related diagnosis: -----
;
Unit_Diag_Bit(16) = "Error appears"
Unit_Diag_Bit(17) = "Error disappears"
;
;Byte 01
Unit_Diag_Bit(24) = "Hardware failure electronics"
Unit_Diag_Bit(25) = "Not used 25"
Unit_Diag_Bit(26) = "Not used 26"
Unit_Diag_Bit(27) = "Not used 27"
Unit_Diag_Bit(28) = "Memory error"
Unit_Diag_Bit(29) = "Not used 29"
Unit_Diag_Bit(30) = "Device not initialized"
Unit_Diag_Bit(31) = "Device initialization failed"

;Byte 02
Unit_Diag_Bit(32) = "Not used 32"
Unit_Diag_Bit(33) = "Not used 33"
Unit_Diag_Bit(34) = "Configuration invalid"
    
```

Unit_Diag_Bit(35) = "Restart"
Unit_Diag_Bit(36) = "Coldstart"
Unit_Diag_Bit(37) = "Maintenance required"
Unit_Diag_Bit(38) = "Not used 38"
Unit_Diag_Bit(39) = "Ident_Number violation"

;Byte 03
Unit_Diag_Bit(40) = "Not used 40"
Unit_Diag_Bit(41) = "Not used 41"
Unit_Diag_Bit(42) = "Not used 42"
Unit_Diag_Bit(43) = "Not used 43"
Unit_Diag_Bit(44) = "Not used 44"
Unit_Diag_Bit(45) = "Not used 45"
Unit_Diag_Bit(46) = "Not used 46"
Unit_Diag_Bit(47) = "Not used 47"

;byte 04
Unit_Diag_Bit(48) = "Not used 48"
Unit_Diag_Bit(49) = "Not used 49"
Unit_Diag_Bit(50) = "Not used 50"
Unit_Diag_Bit(51) = "Not used 51"
Unit_Diag_Bit(52) = "Not used 52"
Unit_Diag_Bit(53) = "Not used 53"
Unit_Diag_Bit(54) = "Not used 54"
Unit_Diag_Bit(55) = "Extension Available"

; extend diag
Unit_Diag_Bit(56) = "Transducer Block in Out of Service"
Unit_Diag_Bit(57) = "FFB is active in Transducer Block"
Unit_Diag_Bit(58) = "Error Code active in FFB"
Unit_Diag_Bit(59) = "Power Supply Failure for Inputs"
Unit_Diag_Bit(60) = "Power Supply Failure for Outputs"
Unit_Diag_Bit(61) = "Addr Jumper is active"
Unit_Diag_Bit(62) = "Not used 62"
Unit_Diag_Bit(63) = "Device is writing lock"

Unit_Diag_Bit(64) = "Simulation Active in DI 1 and/or DI 2 Block"
Unit_Diag_Bit(65) = "Simulation Active in DI 3 and/or DI 4 Block"
Unit_Diag_Bit(66) = "Simulation Active in DI 5 and/or DI 6 Block"
Unit_Diag_Bit(67) = "Simulation Active in DI 7 and/or DI 8 Block"
Unit_Diag_Bit(68) = "Simulation Active in DI 9 and/or DI 10 Block"
Unit_Diag_Bit(69) = "Simulation Active in DI 11 and/or DI 12 Block"
Unit_Diag_Bit(70) = "Simulation Active in DI 13 and/or DI 14 Block"
Unit_Diag_Bit(71) = "Simulation Active in DI 15 and/or DI 16 Block"

Unit_Diag_Bit(72) = "Fail Safe Active in DI 1 and/or DI 2 Block"
Unit_Diag_Bit(73) = "Fail Safe Active in DI 3 and/or DI 4 Block"
Unit_Diag_Bit(74) = "Fail Safe Active in DI 5 and/or DI 6 Block"
Unit_Diag_Bit(75) = "Fail Safe Active in DI 7 and/or DI 8 Block"
Unit_Diag_Bit(76) = "Fail Safe Active in DI 9 and/or DI 10 Block"
Unit_Diag_Bit(77) = "Fail Safe Active in DI 11 and/or DI 12 Block"
Unit_Diag_Bit(78) = "Fail Safe Active in DI 13 and/or DI 14 Block"
Unit_Diag_Bit(79) = "Fail Safe Active in DI 15 and/or DI 16 Block"

Unit_Diag_Bit(80) = "Fail Safe Active in DO 1 and/or DO 2 Block"
Unit_Diag_Bit(81) = "Fail Safe Active in DO 3 and/or DO 4 Block"
Unit_Diag_Bit(82) = "Fail Safe Active in DO 5 and/or DO 6 Block"
Unit_Diag_Bit(83) = "Fail Safe Active in DO 7 and/or DO 8 Block"
Unit_Diag_Bit(84) = "Simulation Active in DO 1 and/or DO 2 Block"
Unit_Diag_Bit(85) = "Simulation Active in DO 3 and/or DO 4 Block"
Unit_Diag_Bit(86) = "Simulation Active in DO 5 and/or DO 6 Block"
Unit_Diag_Bit(87) = "Simulation Active in DO 7 and/or DO 8 Block"

Unit_Diag_Bit(88) = "DI 1 and/or DI 2 Block: Out of Service"
Unit_Diag_Bit(89) = "DI 3 and/or DI 4 Block: Out of Service"
Unit_Diag_Bit(90) = "DI 5 and/or DI 6 Block: Out of Service"

- Unit_Diag_Bit(91) = "DI 7 and/or DI 8 Block: Out of Service"
- Unit_Diag_Bit(92) = "DI 9 and/or DI 10 Block: Out of Service"
- Unit_Diag_Bit(93) = "DI 11 and/or DI 12 Block: Out of Service"
- Unit_Diag_Bit(94) = "DI 13 and/or DI 14 v: Out of Service"
- Unit_Diag_Bit(95) = "DI 15 and/or DI 16 Block: Out of Service"

- Unit_Diag_Bit(96) = "DO 1 Block: Out of Service"
- Unit_Diag_Bit(97) = "DO 2 Block: Out of Service"
- Unit_Diag_Bit(98) = "DO 3 Block: Out of Service"
- Unit_Diag_Bit(99) = "DO 4 Block: Out of Service"
- Unit_Diag_Bit(100) = "DO 5 Block: Out of Service"
- Unit_Diag_Bit(101) = "DO 6 Block: Out of Service"
- Unit_Diag_Bit(102) = "DO 7 Block: Out of Service"
- Unit_Diag_Bit(103) = "DO 8 Block: Out of Service"

Addressing the DC303

1. During the initialization procedure (ie, when powering the **DC303**), if the simulate jumper is configured to "on", the physical address is shown on the outputs and this way, the user can verify the node address on the outputs 1-7, through the LEDs. When the led is on, the address bit represents "1". Output 1 is the least significant bit and the output 7 represents the most significant bit.
2. The output 8 is used to indicates the Identifier_Number_Selector and when is "on" (on state) indicates "Manufacturer Specific" and "off" (off state), indicates Profile Specific".
3. When in "Manufacturer Specific", the Identifier Number is 0x0dca. Once the Identifier_Number_Selector is changed from " Profile Specific " to "Manufacturer Specific" or vice-versa, you must wait 5 seconds while it is saved and the turn off the **DC303** and then the identifier is updated in the level of communication. If the equipment is in "Profile Specific" and using the GSD file Identifier Number equals 0x0dca, the acyclic communication will work well with tools based on EDDL, FDT/DTM, but no cyclic communication with the Profibus-DP master will get success.

ATTENTION

If necessary to execute a reset procedure or a power down and then a power up procedure, if the simulate jumper is on, please, disconnect the outputs since the **DC303** will activate the outputs according to the physical address.

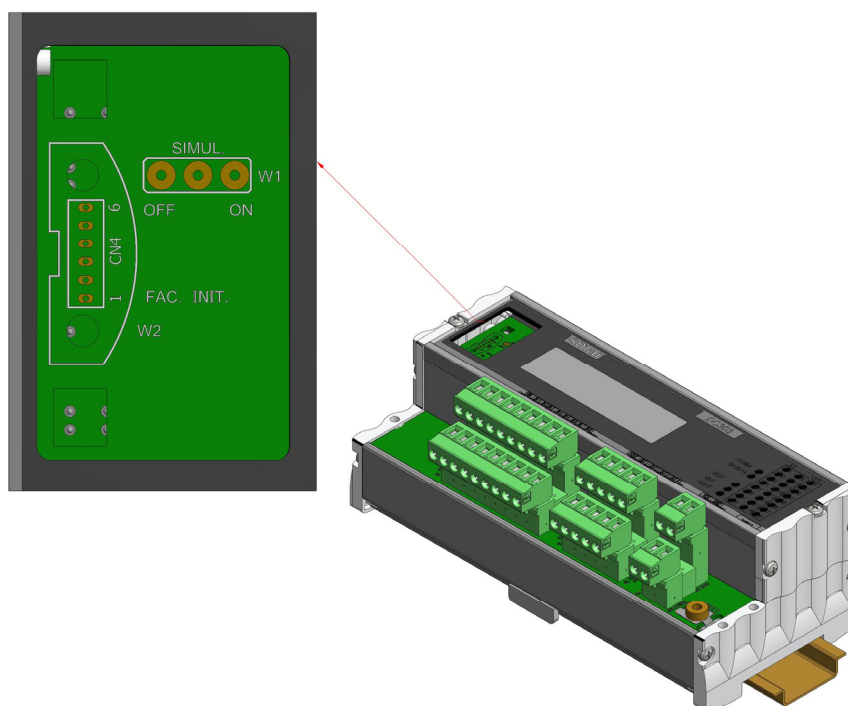


Figure 3.15- Simulate Jumper - DC303

4. To change the address you can use any EDDL-based tool (such as Simatic PDM) or FDT / DTM. For this procedure the user must know the current address of the **DC303**.

Download using Simatic PDM

ATTENTION

In the case of using the Simatic PDM to configure the **DC303**, before using the "Download to Device" command, please, execute a command to "Upload to PC/PG" and also, configure the parameter CHANGE_OPTION to the option "Always accept Logic parameter changes."

Section 4

MAINTENANCE PROCEDURES

General

SMAR **DC303** Profibus-PA Remote I/O are extensively tested and inspected before delivery to the end user. Nevertheless, during their design and development, consideration was given to the possibility of repairs by the end user, if necessary.

In general, it is recommended that the end user do not try to repair printed circuit boards. Instead, he should have spare circuit boards, which may be ordered from SMAR whenever necessary.

TROUBLESHOOTING	
SYMPTOM	PROBABLE SOURCES OF TROUBLE
NO QUIESCENT CURRENT	Profibus Remote I/O Connections: Check wiring polarity and continuity. Power Supply: Check power supply output. The voltage at the DC303 Profibus-PA terminals must be between 9 and 32 Vdc. Electronic Circuit Failure: Check the boards for defect by replacing them with spare ones.
NO COMMUNICATION	Network Connections: Check the network connections: devices, power supply, and terminators. Network Impedance: Check the network impedance (power supply impedance and terminators). Controller Configuration: Check configuration of communication parameters of controller. Network Configuration: Check communication configuration of the network. Electronic Circuit Failure: Try to replace the controller circuit with spare parts.
INCORRECT INPUTS	Input Terminals Connection: Check wiring polarity and continuity. Power supply for Inputs: Check power supply. The voltage must be between 18 and 30 Vdc and the typical consumption when all input is ON is 120 mA.
INCORRECT OUTPUTS	Output Terminals Connection: Check wiring polarity and continuity. Power supply for Outputs: Check power supply. The voltage must be between 20 and 30 Vdc and the maximum current per output is 0.5 A.

Disassembly Procedure

Refer to the Figure 4.1 **DC303** Exploded view. Make sure to disconnect power supply before disassembling the **DC303**.



WARNING

The boards have CMOS components, which may be damaged by electrostatic discharges. Observe correct procedures for handling CMOS components. It is also recommended to store the circuit boards in electrostatic-proof cases.

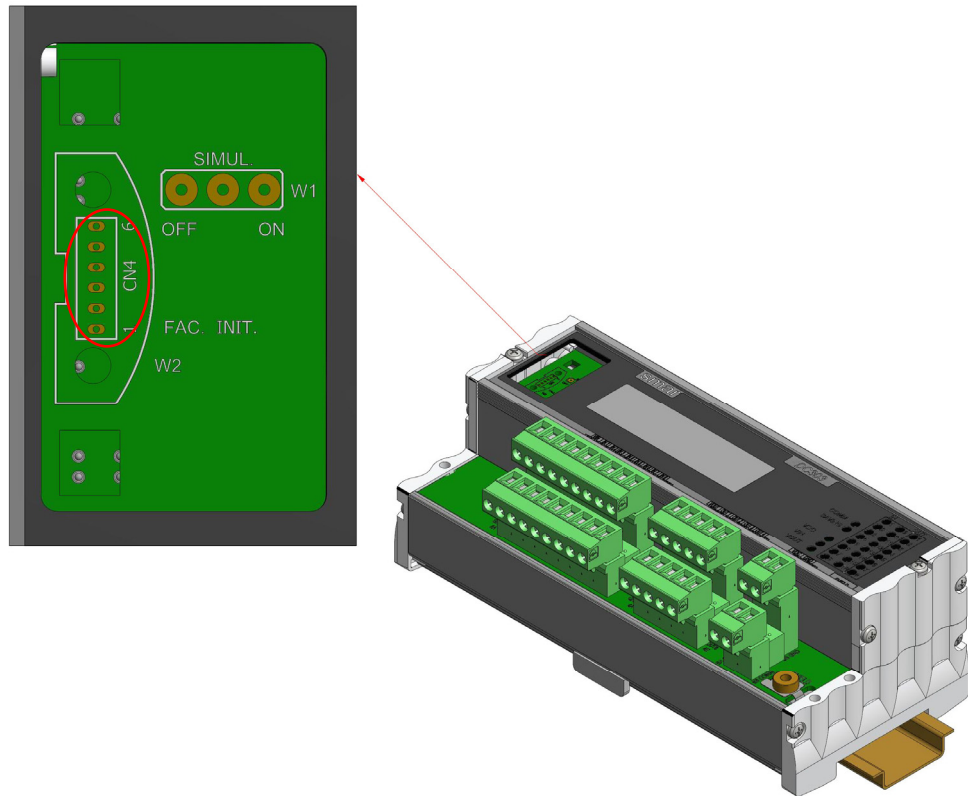
Loose the lateral locks that attach the housing cover and then the main lock. You will have the access to the main circuit board and the I/O electronic board. Gently pull out the main board. To remove the electronic boards, first unscrew the screws that anchor them to the housing, and gently pull out the boards.

Reassembly Procedure

- Put the boards into housing.
- Anchors the board with their screws.
- Make sure all inter connecting pins are connected.
- Observe the LEDs mounting positions, gently lock the housing cover in the lateral locks and the main lock.

Firmware Update Procedure

For firmware update of the **DC303** equipment see **FDI302PLUS** manual, visit website Smar: www.smar.com



NOTA

The FDI302plus should be connected to the CN4 connector

Boards Interchangeability

Main and I/O boards can be changed independently.

Accessories

ACCESSORIES	
ORDERING CODE	DESCRIPTION
PBI-PLUS	USB Interface for Profibus PA.
SYSCON	System Configuration Tool.
PS302	Power Supply.
BT302	Terminator.
FDI-302-2	Field Device Interface - Foundation Profibus & PROFIBUS PA.
DF47	Intrinsic Safety Barrier for Fieldbus.

Exploded view

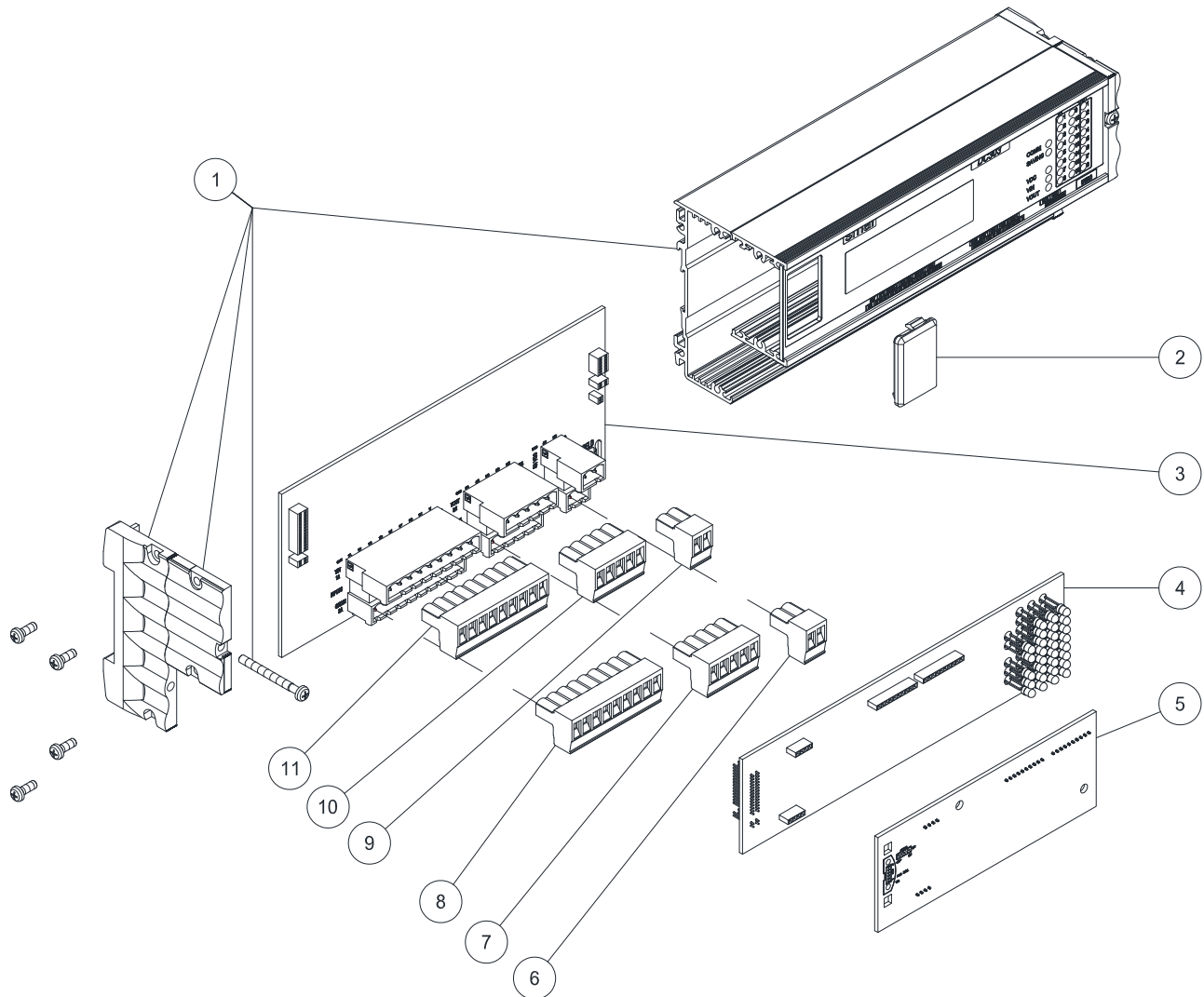


Figure 4.1 – DC Exploded view

Spare Parts

SPARE PARTS		
NAME	POSITION	CODE
Housing	1	400 - 1371
Housing Cover	2	400 - 1372
I/O Board (GLL 1469)	3	400 - 1373
Font Board (GLL 1468)	4	400 - 1374
Main board (GLL 1467)	5	400 - 1375
2-way terminal numbered 6A to 6B	6	400 - 1376
2-way terminal numbered 5A to 5B	9	400 - 1377
5-way terminal numbered 4A to 4E	7	400 - 1378
5-way terminal numbered 3A to 3E	10	400 - 1379
9-way terminal numbered 2A to 2I	8	400 - 1380
9-way terminal numbered 1A to 1I	11	400 - 1381

Section 5

TECHNICAL SPECIFICATIONS

General

Signal (Communication)	Digital only. Profibus-PA, 31.25 Kbits/s voltage mode, in compliance with IEC 61158-2.
Power Supplies	If there are requirements for power supply isolation between inputs and outputs, it is recommended to use at least two power supplies, one for inputs and another one for outputs and Vdc. If the application does not require isolation between inputs and outputs, only one power supply could be used for inputs, outputs and Vdc. Inputs and outputs are optically isolated from each other.
Current consumption quiescent	150 mA from Vdc power supply.
Turn-on Time	Approximately 10 seconds.
Update Time	Approximately 60 ms. The update time is related to the update of the inputs and outputs.
Output Impedance	Non-intrinsic safety from 7.8 kHz - 39 kHz should be greater or equal to 3 k Ω . Intrinsic safety output impedance (assuming an IS barrier in the power supply) from 7.8 kHz - 39 kHz should be greater or equal to 400 Ω .
Function Blocks	Up to 16 Discrete Input Function Blocks (DIs) and up to 8 Discrete Output Function Blocks (DOs). The DC303 has a Built-in Flexible Function Block (FFB) for logic execution such as: AND, OR, XOR and NOT. Functions as: Timer On-Delay, Timer Off-Delay, Timer Pulse, Pulse Counter Down (CTD), Pulse Counter Up (CTU), Flip-Flop RS and Flip-Flop SR.
Vibration Effect	Meets SAMA PMC 31.1.
Temperature Limits	Operation: -40 to 85°C (-40 to 185 °F); Storage: -40 to 110°C (-40 to 230 °F).
Housing	Protection: it has IP20 rating (finger protected) and meets VBG4 and other European accident prevention requirements.
Configuration	Via Profibus Communication using tools based on EDDL or FDT/TM.
Mounting	Using DIN rail (TS35-DIN EN 50022 or TS32-DIN EN50035 or TS15-DIN EN50045).

DC303 Inputs

Description-Inputs

The input module senses the DC input voltage and converts it into a True (ON) or False (OFF) logic signal. It has 1 optically isolated group of 16 inputs to detect 24 Vdc.

In case of failure of input power supply there will be indication in the cyclic diagnose bytes.

Technical specifications

Architecture	Number of Inputs is 16.
Isolation, groups are individually isolated	Optical Isolation up to 5000 Vac.
External Power	Voltage Source for Inputs 18 - 30 Vdc.
Typical Consumption per group (all inputs ON)	120 mA.
Power Indicator	Green LED.
Inputs	ON State Level (True Logic) 15 - 30 Vdc.
	OFF State Level (False Logic) 0 - 5 Vdc.
Typical Impedance	3.9 k Ω .
Status display	Red LED.
Switching Information	Time from "0" to "1": 30 μ s.
	Time from "1" to "0": 50 μ s.
Wire	One wire 14 AWG (2 mm ²).
	Two wires 20 AWG (0.5 mm ²).

DC303 Open Collector Outputs

Description – Outputs

The outputs are designed with open collector NPN transistors that are able to drive relays, solenoids and other DC loads with up to 0.5 A per output. All channels within a group share the same ground whereas groups are isolated from each other and the Fieldbus network.

In case of failure of output power supply there will be indication in the cyclic diagnose bytes.


Technical specifications

Architecture	Number of Outputs 8.
Isolation	Optical Isolation up to 5000 Vac.
External Power	Voltage Source for Outputs 20 to 30 Vdc.
Maximum Consumption	35 mA.
Power Indicator	Green LED.
Outputs	Maximum Switched Voltage 30 Vdc.
	Maximum Saturation Voltage 0.55 V @ 0.5 A.
	Maximum Current per Output 0.5 A.
	Status Display Yellow LED.
	Indicator Logic ON when the transistor is on.
	Maximum Leakage Current 100 µA @ 35 Vdc.
Output Status During: Power-Up Firmware Download Configuration Download	OFF.
Independent Protection per Output	Thermal Shutdown 165 °C.
	Thermal Hysteresis 15 °C.
	Over-Current Protection 1.3 A @ 25 Vdc maximum.
Wire	One wire 14 AWG (2 mm ²).
	Two wires 20 AWG (0.5 mm ²).

Ordering Code

MODEL	DESCRIPTION
DC303-Profibus-PA Remote I/O	- 1 Group of 16 24Vdc optically isolated inputs. - 1 group of 8 optically isolated open collector outputs.

Appendix A

	SRF – Service Request Form		
	Profibus-PA Remote Input and Output		
GENERAL DATA			
Model:	DC303		
Serial Number:	_____		
TAG:	_____		
Channels being used in DC303:	INPUT	1-4 () 9-12 ()	
		5-8 () 13-16 ()	
	OUTPUT	1-4 () 5-8 ()	
Configuration:	PC ()	Software: _____	Version: _____
INSTALLATION DATA			
Type/Model/Manufacturer of device connected to DC303:	_____		
PROCESS DATA			
Hazardous Area Classification:	() Yes, please specify: _____		
	() No		
	More details: _____		
Interference types present in the area:	No interference ()	Temperature ()	Vibration () Other: _____
Environment Temperature:	From _____°C up to _____°C.		
OCCURRENCE DESCRIPTION			

SERVICE SUGGESTION			
Adjustment ()	Cleaning ()	Preventive Maintenance ()	Update / Up-grade ()
Other: _____			
USER INFORMATION			
Company:	_____		
Contact:	_____		
Title:	_____		
Section:	_____		
Phone:	_____	_____	Extension: _____
E-mail:	_____	_____	Date: ____/____/____
For warranty or non-warranty repair, please contact your representative. Further information about address and contacts can be found on www.smar.com/contactus.asp .			

Returning Materials

If necessary to return the **DC303** to SMAR, simply contact our office, informing the defective instrument serial number, and return it to our factory.

In order to speed up analysis and solution of the problem, the defective item should be returned with a description of the failure observed, with as much details as possible. Other information concerning the instrument operation, such as service and process conditions, is also helpful.

Instruments returned or to be revised outside the guarantee term should be accompanied by a purchase order or a quote request.