

# Manual

**LioN-P  $\mu$ DCU  
Distributed Control Unit**

**LioN-P Digital  
LDMicro  
0980 ESL 393-121-DCU1  
0980 ESL 390-121-DCU1  
 $\mu$ DCU Programming Manual**

# Contents

<b>1 About This Manual</b>	<b>5</b>
1.1 General Information	5
1.2 Version information	6
<b>2 Introduction</b>	<b>7</b>
<b>3 Explanation of terms</b>	<b>8</b>
<b>4 Cyclic Data and Parameters</b>	<b>9</b>
4.1 Consuming Data	9
4.2 Producing Data	9
4.3 Data Exchange	10
4.3.1 PROFINET	10
4.3.2 EtherNet/IP	11
4.3.3 EtherCAT	13
4.4 Module Parameters	14
4.4.1 DCU startup parameter	14
<b>5 Technical Data</b>	<b>15</b>
5.1 DCU / Forcemode LED description	15
5.2 Port types	16
5.2.1 0980 ESL 393-121-DCU1 (LioN-P 8DI8DO with DCU)	16
5.2.2 0980 ESL 390-121-DCU1 (LioN-P 16DIO with DCU)	17
5.3 Electrical Specifications	17

<b>6 DCU Programming Details</b>	<b>18</b>
6.1 $\mu$ DCU / LDMicro Limitations	18
6.2 LDMicro	19
6.2.1 LDMicro Introduction	19
6.2.2 Filetypes	20
6.2.3 Datatypes	20
6.2.4 Naming conventions	21
6.2.4.1 LDMicro conventions	21
6.2.4.2 LioN-P $\mu$ DCU conventions	21
6.2.5 Available Data	22
6.2.6 Physical inputs and outputs	23
6.2.7 Direct access to cyclic bits	23
6.2.8 Reading and manipulating consuming and producing data by channel	24
6.2.9 Data Exchange	25
<b>7 DCU Web Interface</b>	<b>26</b>
7.1 Web Interface element overview	26
7.2 Username and Password	27
7.3 DCU States	27
7.4 Uploading a program into the DCU	27
7.5 Program Information	28
7.6 Autostart	28
7.7 Custom Mapping	28
7.7.1 Create mapping with the mapping dialog	29
7.7.2 Create a mapping file manually	30
7.8 Variable Forcing	30
<b>8 DCU Program and Mapping batch upload</b>	<b>32</b>
8.1 POST request for uploading files	32

8.2 Using the Perl script	33
<b>9 Standard JSON Module Information</b>	<b>34</b>
9.1 Example JSON response	34
9.2 JSON Response object structure	35
9.3 JSON Response Description	36

# 1 About This Manual

## 1.1 General Information

Please read the explanations and configuration instructions in this manual carefully before starting up the LioN-P modules. Keep this manual where it is accessible to all users.

The texts, illustrations, diagrams, and examples serve to illustrate the functionality of the LioN-P  $\mu$ DCU modules.

Please contact us if you have any detailed questions on installing and starting up the devices.

Belden Deutschland GmbH  
– Lumberg Automation™ –  
Im Gewerbepark 2  
D-58579 Schalksmühle  
Germany  
[support-automation@belden.com](mailto:support-automation@belden.com)  
[www.lumberg-automation.com](http://www.lumberg-automation.com)

Belden Deutschland GmbH – Lumberg Automation™ – reserves the right to make technical changes or changes to this manual at any time without notice.

## 1.2 Version information

Index	Created	Changed
Version number	Version 1.0	Version 1.1
Date		October 2017
Name/department		

Index	Changed	Changed
Version number	Version 1.2	
Date	January 2020	
Name/department	JGA/R&D	

*Table 1: Overview of manual revisions*

# 2 Introduction

The LioN-P  $\mu$ DCU is a multiprotocol fieldbus slave module based on the 0980 ESL 393-121 / 0980 ESL 390-121 module. It provides all the functionality of the base module, but has an additional integrated programmable logic unit. This unit can execute user programs created with a small external tool, called LDMicro. These programs are created in a ladder logic manner and are called “DCU programs”.

This allows the user to add additional control logic which is stored directly in the slave module itself and is independent from fieldbus or plc. This ranges from simple Boolean operations of input and outputs to fully autonomous (without any plc) programs.

To a plc the module appears as a normal slave module with 8 bit output data (consuming) and 16 bit input data (producing) of cyclic data.

I/Os used by the DCU program are no more controllable by the plc directly, but can be used to communicate with the plc, because the corresponding cyclic bits can be read and written by the DCU program.

## 3 Explanation of terms

DCU	Distributed Control Unit. A programmable logic for fieldbus modules
DCU program	A user written program which can be executed by a DCU (module).
LDMicro	A windows software to create DCU programs in a ladder logic manner.
Port	Physical plug on a i/o module (M12 / M8, 5 pole) e.g. named with X1, X6. The modules here have 8 ports.
Channel	A pin on a port. Defined by a port and a channel letter. E.g. X1A, X4B. Each port can have up to two channels. The channels are found on pin 4 (A) and pin 2 (B) of each port.
Consuming Data	Cyclic data which is transferred by a fieldbus from a PLC to the i/o module. E.g. for controlling an output.
Producing Data	Cyclic data which is transferred by a fieldbus from an i/o module to the PLC. E.g. for reading an input.



## 4 Cyclic Data and Parameters

Fixed for 0980 ESL 393-121-DCU1 and default values for 0980 ESL 390-121-DCU1.

### 4.1 Consuming Data

Output	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte n	4B	4A	3B	3A	2B	2A	1B	1A
Byte n + 1	8B	8A	7B	7A	6B	6A	5B	5A

The following applies here:

- ▶ 1A ... 8A: Output of channel A (contact pin 4) of the M12 socket connections 1 to 8.
- ▶ 1B ... 8B: Output of channel B (contact pin 2) of the M12 socket connections 1 to 8.

### 4.2 Producing Data

Input	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte n	4B	4A	3B	3A	2B	2A	1B	1A
Byte n + 1	8B	8A	7B	7A	6B	6A	5B	5A

- ▶ 1A ... 8A: Actual status of channel A (contact pin 4) of the M12 socket connections 1 to 8.
- ▶ 1B ... 8B: Actual status of channel B (contact pin 2) of the M12 socket connections 1 to 8.



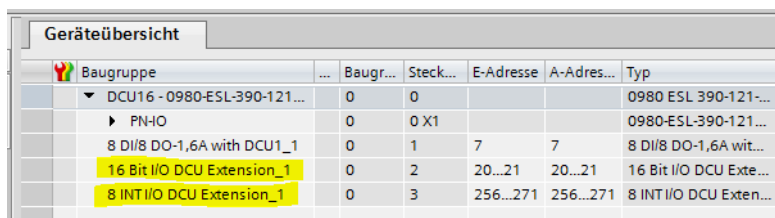
**Attention:** Depending on the selected fieldbus, the module can provide additional cyclic data bytes (e.g. diagnostic information). Please refer to the manual of the base module for information about fieldbus specific details.

## 4.3 Data Exchange

Only available in 0980 ESL 390-121-DCU1 (16 DIO Universal DCU)

- ▶ The module provides additional cyclic data purely for data exchange between a PLC and the DCU program. The DCU program can, for example, take commands and data from the PLC and respond with execution results.
- ▶ The exchange data space contains 16 bit plus 8 words (as 16 bit signed integer) in each direction.
- ▶ The data exchange bits can be written with the  $YEn$  bit variable in LDMicro.
- ▶ The data exchange bits can be read with the  $XEn$  bit variable in LDMicro.
- ▶ The symbols for the integer variables  $EIn$  and  $EOn$  allows the reading and writing of exchange data words.

### 4.3.1 PROFINET



Baugruppe	Baugr...	Steck...	E-Adresse	A-Adres...	Typ
DCU16 - 0980-ESL-390-121...	0	0			0980 ESL 390-121-...
▶ PN-IO	0	0 X1			0980-ESL-390-121...
8 DI/8 DO-1,6A with DCU1_1	0	1	7	7	8 DI/8 DO-1,6A wit...
16 Bit I/O DCU Extension_1	0	2	20...21	20...21	16 Bit I/O DCU Exte...
8 INT I/O DCU Extension_1	0	3	256...271	256...271	8 INT I/O DCU Exten...

Figure 1: DCU exchange area in TIA portal

In PROFINET the exchange area consists of two additional slots (2 and 3). Slot 2 contains the 16 bit exchange data and slot 3 the 16 byte exchange words.

### 4.3.2 EtherNet/IP

The exchange data is provided within the EtherNet/IP cyclic data. The format of this data is subject to change with the currently selected assembly (16DI/DO, 8DI/8DI, 16DI etc.).

The following cyclic data applies to the default 16DI/16DO assembly with DCU. For further information please refer to the LioN-P EtherNet/IP manual.

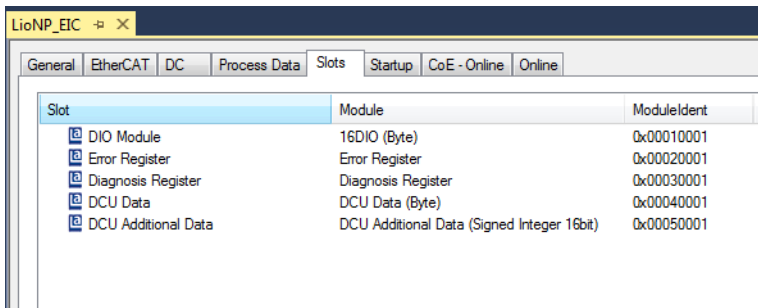
Byte	Function
0	Input Data
1	Input Data
2	General Diagnosis
3	Sensor Diagnosis
4	Reserved
5	Actuator Diagnosis 1
6	Actuator Diagnosis 2
7	DCU Bit exchange Byte 1
8	DCU Bit exchange Byte 2
9	DCU Integer exchange 1 MSB
10	DCU Integer exchange 1 LSB
11	DCU Integer exchange 2 MSB
12	DCU Integer exchange 2 LSB
:	:
24	DCU Integer exchange 8 MSB
25	DCU Integer exchange 8 LSB

*Table 2: Cyclic producing data*

Byte	Function
0	Output Data
1	Output Data
2	DCU Bit exchange Byte 1
3	DCU Bit exchange Byte 2
4	DCU Integer exchange 1 MSB
5	DCU Integer exchange 1 LSB
6	DCU Integer exchange 2 MSB
7	DCU Integer exchange 2 LSB
:	:
20	DCU Integer exchange 8 MSB
21	DCU Integer exchange 8 LSB

*Table 3: Cyclic consuming data*

### 4.3.3 EtherCAT



Slot	Module	ModuleIdent
DIO Module	16DIO (Byte)	0x00010001
Error Register	Error Register	0x00020001
Diagnosis Register	Diagnosis Register	0x00030001
DCU Data	DCU Data (Byte)	0x00040001
DCU Additional Data	DCU Additional Data (Signed Integer 16bit)	0x00050001

Figure 2: DCU exchange area slots in TwinCAT3

Name	Online	Type	Size	> Addr...	In/Out
Diagnosis Register		UDINT	4.0	42.0	Input
DCU Inputs 0..7		USINT	1.0	46.0	Input
DCU Inputs 8..15		USINT	1.0	47.0	Input
DCU Additional Input 0		INT	2.0	48.0	Input
DCU Additional Input 1		INT	2.0	50.0	Input
DCU Additional Input 2		INT	2.0	52.0	Input
DCU Additional Input 3		INT	2.0	54.0	Input
DCU Additional Input 4		INT	2.0	56.0	Input
DCU Additional Input 5		INT	2.0	58.0	Input
DCU Additional Input 6		INT	2.0	60.0	Input
DCU Additional Input 7		INT	2.0	62.0	Input
WcState		BIT	0.1	1522.1	Input
InputToggle		BIT	0.1	1524.1	Input

Figure 3: DCU exchange variables in TwinCAT3

In EtherCAT the two exchange areas are organized as additional slots. The 8 signed integer values are directly showed as variables from type INT with size 2.

## 4.4 Module Parameters

The LioN-P  $\mu$ DCU has one additional plc parameter which controls the DCU startup behaviour.

### 4.4.1 DCU startup parameter

Disabled	The DCU starts in DISABLED state.
Lock	The DCU is disabled and cannot be started by web interface.
Run	The DCU starts in RUN state and executed the DCU program, IF there is a valid program loaded.

The screenshot shows the TIA Portal configuration interface for the DCU startup parameter. The left sidebar contains a tree view with the following structure:

- Allgemein
  - Kataloginformation
  - PROFINET-Schnittstelle [X1]
    - Allgemein
    - Ethernet-Adressen
    - Erweiterte Optionen
    - HW-Kennung
    - Identification & Maintenance
    - Baugruppenparameter**
    - HW-Kennung

The main configuration area is titled 'Baugruppenparameter' and contains two sections:

- General Parameters**
  - Report Alarms: On
  - Report Alarm UL: On
  - Force Mode: Enabled
  - Web Interface: Enabled
  - DCU Startup: Disabled (can be enabled by web interface)
- Fail Safe Configuration**
  - Fail Safe Value Port5 Ch.A: Set Low
  - Fail Safe Value Port5 Ch.B: Set Low
  - Fail Safe Value Port6 Ch.A: Set Low
  - Fail Safe Value Port6 Ch.B: Set Low
  - Fail Safe Value Port7 Ch.A: Set Low
  - Fail Safe Value Port7 Ch.B: Set Low
  - Fail Safe Value Port8 Ch.A: Set Low
  - Fail Safe Value Port8 Ch.B: Set Low

The 'DCU Startup' dropdown menu is open, showing the following options:

- Disabled (can be enabled by web interface)
- Locked (disabled and cannot be enabled by web interface)
- Disabled (can be enabled by web interface)** (highlighted)
- RUN (enabled and module starts in RUN mode)

Figure 4: DCU Startup Parameter in TIA Portal

## 5 Technical Data

### 5.1 DCU / Forcemode LED description

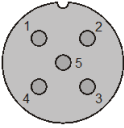
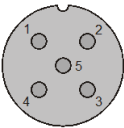
The DCU/FM LED on the module indicates the DCU and force mode status.

LED Color	Meaning
Off	DCU Disabled / No Forcemode active
Blue Flashing	DCU is running
Blue On	DCU is stopped
Red On	DCU error
Blue/Red Flashing	Force Mode ON

## 5.2 Port types

### 5.2.1 0980 ESL 393-121-DCU1 (LioN-P 8DI8DO with DCU)

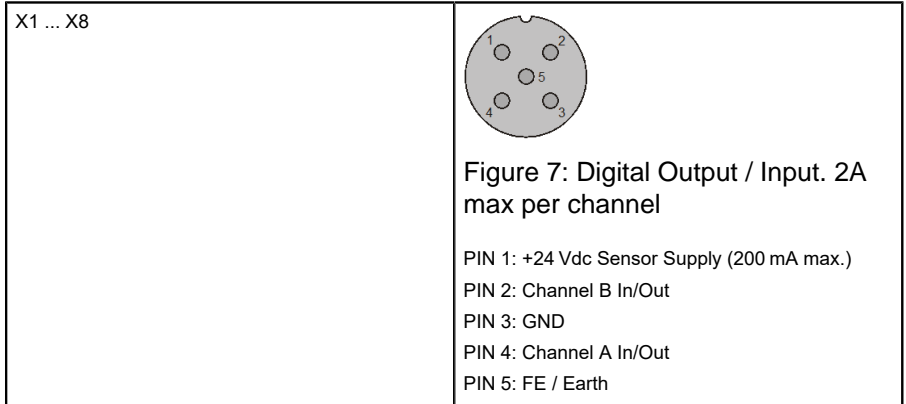
The  $\mu$ DCU has 8 digital inputs and 8 digital outputs organized in 8 ports with 2 channels each. For more details please refer to the 0980 ESL 393-121 datasheet and manual.

X1 ... X4	 <p><b>Figure 5: Digital Input</b></p> <p>PIN 1: +24 Vdc Sensor Supply (200 mA max.)          PIN 2: Input Channel B          PIN 3: GND          PIN 4: Input Channel A          PIN 5: FE / Earth</p>
X5 ... X8	 <p><b>Figure 6: Digital Output / 2 A max.</b></p> <p>PIN 1: n. c.          PIN 2: Output Channel B          PIN 3: GND          PIN 4: Output Channel A          PIN 5: FE / Earth</p>



### 5.2.2 0980 ESL 390-121-DCU1 (LioN-P 16DIO with DCU)

This module has 16 universal digital channels, organized in 8 ports with 2 channels each. Each channel can be used as a digital input or digital output.



## 5.3 Electrical Specifications

Please refer to 0980 ESL 393-121-DCU1 or 0980 ESL 930-121-DCU1 datasheet.

## 6 DCU Programming Details

### 6.1 $\mu$ DCU / LDMicro Limitations

Max. Rungs	99
Max. Bits	99
Max. Integers	99
Max. Lines (compiled program)	4096
Average $\mu$ DCU Cycle Time	10 ms

## 6.2 LDMicro

Open source ladder logic programming tool:

LDMicro download: <http://cq.cx/ladder.pl#dl>

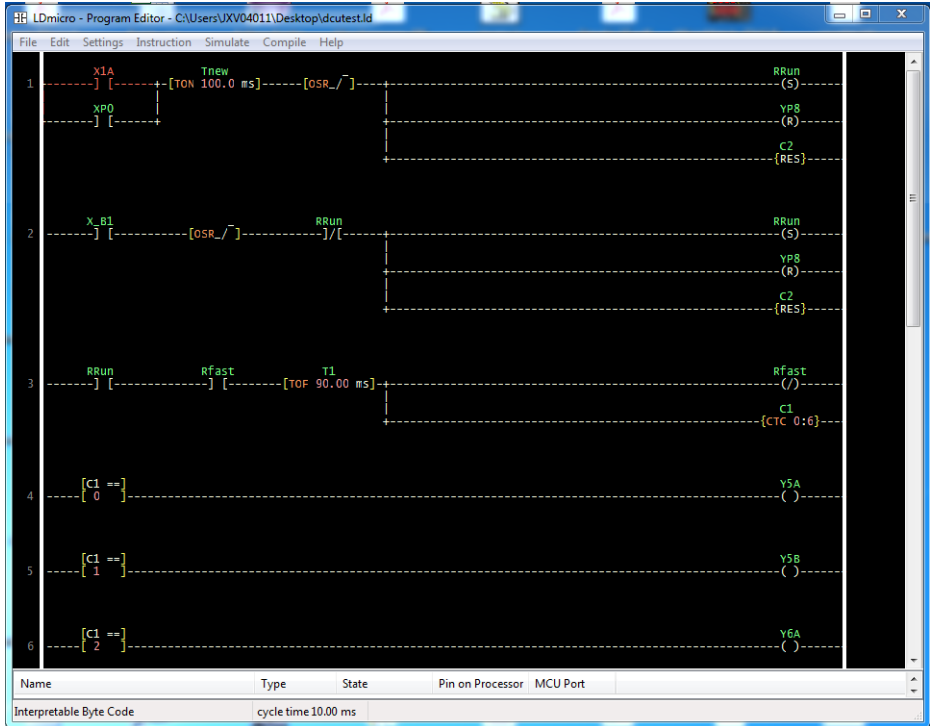


Figure 8: LDMicro user interface

### 6.2.1 LDMicro Introduction

With LDMicro the user can create programs in a Ladder Diagram style according to EN 61131 - 3. Here all elements of the program are arranged on horizontal lines (Rungs). Rungs are always executed from left to right without a guaranteed Rung order. This concept is derived from hardwired relay circuits.

LDMicro offers a large number of instructions such as:

- ▶ Bit operations such as contacts, coils, set/reset
- ▶ Edge Detection
- ▶ Timers and turn on/off delays
- ▶ Up/Down/Circular counters
- ▶ 16 bit signed arithmetic operations

DCU programs created with LDMicro are able to:

- ▶ Control all inputs and outputs of the module
- ▶ React to diagnostic events (short-circuit, undervoltage etc.)
- ▶ Communicate with a connected PLC
- ▶ Share information on the network

### 6.2.2 Filetypes

Program files for LDMicro are named with .ld. Those files can be loaded, edited and saved via LDMicro applicaton.

To compile a program for the DCU, first select the correct target type under **Settings > Microcontroller > Interpretable Bytecode**.

It is also possible to set the cycle time (**Settings > MCU Parameters**. A cycle time of 10 ms or above is recommended.

Then choose from menu **Compile > Compile as...**, and select a location and name where the compiled program should be stored. The program will then be compiled. The result is an .int file.

This file can now be uploaded into the DCU.

### 6.2.3 Datatypes

LDMicro knows the following datatypes:

Bit	0 or 1
Int	16 bit signed integer (-32768 to +32767)
T	Timer
C	Counter

## 6.2.4 Naming conventions

### 6.2.4.1 LDMicro conventions

There are 3 types of bits with a mandatory naming convention:

Type	Convention	Example
Input Bit	Must start with "X"	X1A, X5P
Output Bit	Must start with "Y"	Y2B, Y3P
Internal Relay	Must start with "R"	R1, RRun, RStart

### 6.2.4.2 LioN-P µDCU conventions

Type	Convention	Example
Physical IO Input Data	X followed by port number and channel	X1A, X5B
Physical IO Output Data	Y followed by port number and channel	Y2B, Y7A
Cyclic data to PLC	Y followed by "P" and bit number	YP5, YP15
Cyclic data from PLC	X followed by "P" and bit number	XP0, XP6
Special bits	"X" or "Y" followed by <u>  </u> (underline) and a name	X_DIA, Y_STOP
Integer values for IOs	IN or OUT followed by byte no.	IN1, IN2, OUT1, OUT2
Integer values for special information	<u>  </u> (underline) followed by a name	_SCS, _CE1

## 6.2.5 Available Data

This data is available directly in LDMicro programs. Just name a bit or integer variable in LDMicro according to the following list.

Symbol	Direction	Type	Description
<b>Basic Input/Output Data</b>			
Xn[A/B]	Input	Bit	Read digital input state from port n (1...8). Channel A or B
Yn[A/B]	Output	Bit	Write digital output state to port n (1...8). Channel A or B
OUT[0 1]	Output	Int	Write 8 output states from INT. (0=X1...X4, 1= X5...X8)
IN[0 1]	Input	Int	Read 8 input states as INT. (0=X1...X4, 1= X5...X8)

<b>Data exchange (with PLC)</b>			
XPn	Input	Bit	Read consuming bit from PLC. n = 0...15
YPn	Output	Bit	Write producing bit to PLC. n = 0...15
XCn[A/B]	Input	Bit	Read consuming data from PLC for port n (1...8). Channel A or B
YPn[A/B]	Input	Bit	Write producing data to PLC for port n (1...8). Channel A or B
XEn <sup>*</sup>	Input	Bit	Read data exchange bit n (0...15)
YEn <sup>*</sup>	Output	Bit	Write data exchange bit n (0...15)
EIn <sup>*</sup>	Input	Int	Data exchange value from PLC..n=0...7
EOn <sup>*</sup>	Out	Int	Data exchange value to PLC. n=0...7

<b>Diagnostic Information</b>			
X_DIA	Input	Bit	Diagnosis Master Bit
X_SCS	Input	Bit	Sensor Diagnosis Bit
X_SCA	Input	Bit	Actuator Diagnosis Bit
X_LVS	Input	Bit	Sensor Supply Voltage fault
X_LVA	Input	Bit	Actuator Supply Voltage fault
X_COMM	Input	Bit	Cyclic connection to PLC established
_SCS	Input	Int	Sensor short circuit information per Port
_CE1	Input	Int	Channel Error LSB
_CE2	Input	Int	Channel Error MSB

DCU Control			
Y_STOP	Output	Bit	Causes the DCU to STOP
Y_DIS	Output	Bit	Causes the DCU to DISABLE itself
Special			
_Pn	Output	Int	Data to publish. n = 0...31
_MSG	Output	Int	Show Message with corresponding number on web gui
X_Bn	Input	Bit	Virtual Button on web gui pressed. n = button number 1...10
X_FIRST	Input	Bit	Is set only on first program cycle after power-up or reset.

\* Only available in 0980 ESL 390-121-DCU1 (16 DIO Universal)

### 6.2.6 Physical inputs and outputs

The symbol  $X_nA/X_nB$  allows the DCU program to read directly the corresponding physical input. A *contact* named with this symbol would be interpreted as *closed* if the corresponding input pin is shorted with +24 VDC (e.g. Pin 1).

The symbol  $Y_nA/Y_nB$  can be used to control directly a physical digital output. A *coil* which is named with this symbol would activate the corresponding output pin which is set on +24 VDC.

I/Os which are used in a DCU program are disconnected from the corresponding cyclic data to and from the plc. However, this cyclic data can still be read and manipulated by the DCU program, in order to communicate or exchange information with the plc.

I/Os which are NOT used can still be directly controlled by a plc.

### 6.2.7 Direct access to cyclic bits

The module provides 16 bit of cyclic input data to the plc (producing data), which is represented in the DCU program by the symbol  $YP_n$ , where n is the bit number of the cyclic bit ranging from 0 to 15. A *coil* which is named with this symbol would control the corresponding cyclic bit in den module's producing data.

Only cyclic bits which are disconnected from physical I/Os (because they are used in a DCU program) can be manipulated in this way.

Likewise, the 8 bit of cyclic output data from the plc (consuming data) can be read by a DCU program with the  $XP_n$  symbol, where  $n$  is the cyclic bit number ranging from 0 to 7.

This allows the DCU program to react on events triggered by the plc.

### 6.2.8 Reading and manipulating consuming and producing data by channel

The  $XC_nA$  /  $XC_nB$  symbols read the consuming cyclic bit received from the PLC which controls the specified output channel. Even if the channel is controlled by the DCU and therefore not by the PLC directly, the DCU program is able to react on the state of this bit.

The  $YP_nA$  /  $YP_nB$  symbols manipulate the producing data for the specified channel which is sent to the PLC via the cyclic data. With this a DCU program can simulate an input state to the PLC independently from the real input state of the channel (input simulation).

Physical Port/ Channel	Port 1/A	Port 1/B	Port 2/A	Port 2/B	Port 3/A	Port 3/B	Port 4/A	Port 4/B
Read consuming bit	XC1A	XC1B	XC2A	XC2B	XC3A	XC3B	XC4A	XC4B
Manipulate producing bit	YP1A	YP1B	YP2A	YP2B	YP3A	YP3B	YP4A	YP4B

Physical Port/ Channel	Port 5/A	Port 5/B	Port 6/A	Port 6/B	Port 7/A	Port 7/B	Port 8/A	Port 8/B
Read consuming bit	XC5A	XC5B	XC6A	XC6B	XC7A	XC7B	XC8A	XC8B
Manipulate producing bit	YP5A	YP5B	YP6A	YP6B	YP7A	YP7B	YP8A	YP8B



### 6.2.9 Data Exchange

Only available in 0980 ESL 390-121-DCU1 (16 DIO Universal DCU)

- ▶ The module provides additional cyclic data purely for data exchange between a PLC and the DCU program. The DCU program can, for example, take commands and data from the PLC and respond with execution results.
- ▶ The exchange data space contains 16 bit plus 8 words (as 16 bit signed integer) in each direction.
- ▶ The data exchange bits can be written with the  $YE_n$  bit variable in LDMicro.
- ▶ The data exchange bits can be read with the  $XE_n$  bit variable in LDMicro.
- ▶ The symbols for the integer variables  $EIn$  and  $EOn$  allows the reading and writing of exchange data words.

# 7 DCU Web Interface

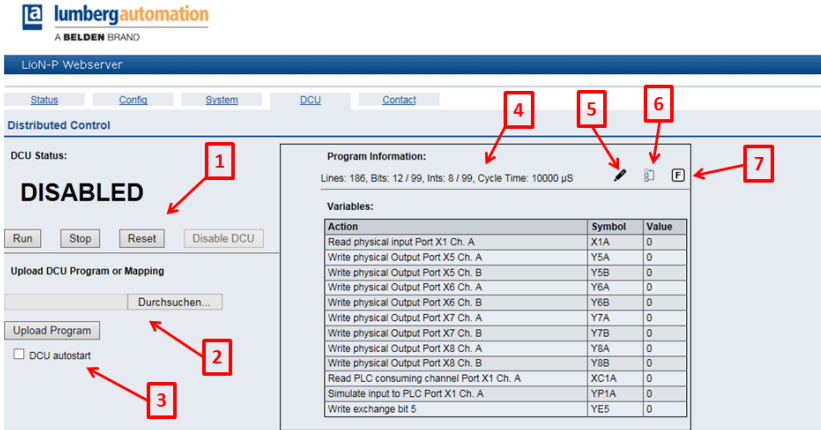


Figure 9: DCU web interface

The DCU web interface allows the user to upload programs to the DCU and to control the DCU state.

## 7.1 Web Interface element overview

1. Shows the current DCU state and provides buttons to control the DCU.
2. Upload a DCU program or mapping file.
3. If the autostart box is checked, the DCU will run automatically after power up, if a valid program is present.

4. Shows basic program information and a list of all module functions currently used by the DCU program by its variables. It also displays the current value of each variable.
5. Change the number format of the value column for INT variables
6. Opens the mapping dialog for creating a custom variable mapping.
7. Activate/deactivate variable forcing.

## 7.2 Username and Password

To change the DCU state or upload programs, "WRITE" or "ADMIN" privileges are required. The default password for user "admin" is "private".

## 7.3 DCU States

There are the following DCU states:

State on web site	Description
NO PROGRAM	There is no program loaded or the uploaded file is not a valid program.
LOCKED	The DCU is locked by the master (PLC) configuration.
DISABLED	The DCU is disabled. No program is running and the DCU has no control over the inputs and outputs. <b>The module acts as a normal digital I/O module.</b>
<b>STOP</b>	The DCU controls the inputs and outputs that are used in the loaded program, but the program is stopped. All other inputs and outputs can still controlled by the master.
<b>RUN</b>	The DCU controls the inputs and outputs that are used in the loaded program, and the program is executed. All other inputs and outputs can still be controlled by the master.

## 7.4 Uploading a program into the DCU

Programs which are created and compiled with LDMicro can be directly uploaded into the plc. Choose the program file (.int) and press upload.

Program upload is NOT allowed when the DCU is in RUN mode. WRITE or ADMIN user rights are needed to upload a DCU program.

It is also possible to upload a variable mapping file (.map).

## 7.5 Program Information

The box on the right shows some information about the currently loaded program. The variable table shows all variables in the program which are mapped to a module function. Here are also located the buttons for changing the number format, mapping dialog and variable forcing.

## 7.6 Autostart

If the autostart checkbox is checked, the DCU will automatically start in RUN mode if the module is powered on and if there is a valid program loaded.

## 7.7 Custom Mapping

Typically, the mapping between a variable and the corresponding module function is implicitly supposed by the variable name, according to the table in chapter 5.5. For example, a variable named Y5B is automatically connected to the module function “Set physical output Port 5 Channel B”. If this approach is used, no further mapping is needed.

For a better readability of the DCU program, it can be useful to name variables according to their function in the application. For example, if an output should control a green LED, the variable can be Y\_LEDGreen.

Such a variable name is not known by the module and therefore not connected to any function and this variable will not be shown in the variable list, but the mapping between this variable and a physical output can be made manually via the variable mapping

### 7.7.1 Create mapping with the mapping dialog

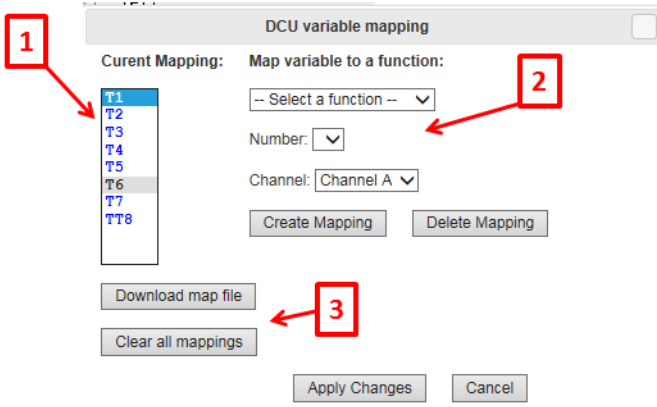


Figure 10: Variable mapping dialog

The mapping dialog shows a list of all variables in the current program (which are not mapped automatically by their names). [1]

Each of these variables can be connected to a device function by selecting the variable. After that, the function dropdown list fills with possible functions for this variable type.

After selecting a function, it can be necessary to specify it. For example choose a number or a port and channel. [2]

A click on **Create Mapping** will connect the variable with this function. This connection is also visible in the variable list.

The button **Delete Mapping** breaks an existing mapping.

If all needed variables are mapped, click on **Apply Changes** to send the changes to the module.

The variable list on the DCU page now shows also the newly mapped variables with the original function name in brackets.

For backup, reuse or external editing, the current mapping can be downloaded as a file. [3]

The button **Clear all mappings** deletes all existing mappings in the device.

## 7.7.2 Create a mapping file manually

A variable mapping can also be created by simply uploading a mapping file. The file is structured as follows:

```
[CustomName];[Symbol]\r\n
```

### Example

```
XStart;X1A
```

```
YLED;Y5B
```

The file should be terminated by the extension ".map".

## 7.8 Variable Forcing




Program Information:		
Lines: 186, Bits: 12 / 99, Ints: 8 / 99, Cycle Time: 10000 µs		  
Variables:		
Action	Symbol	Value
Read physical input Port X1 Ch. A	X1A	0 <input type="checkbox"/>
Write physical Output Port X5 Ch. A	Y5A	1 <input checked="" type="checkbox"/>
Write physical Output Port X5 Ch. B	Y5B	0 <input type="checkbox"/>
Write physical Output Port X6 Ch. A	Y6A	0 <input type="checkbox"/>
Write physical Output Port X6 Ch. B	Y6B	0 <input type="checkbox"/>
Write physical Output Port X7 Ch. A	Y7A	0 <input type="checkbox"/>
Write physical Output Port X7 Ch. B	Y7B	0 <input type="checkbox"/>
Write physical Output Port X8 Ch. A	Y8A	1 <input checked="" type="checkbox"/>
Write physical Output Port X8 Ch. B	Y8B	0 <input type="checkbox"/>
Read PLC consuming channel Port X1 Ch. A	XC1A	0 <input type="checkbox"/>
Simulate input to PLC Port X1 Ch. A	YP1A	0 <input type="checkbox"/>
Write exchange bit 5	YE5	1 <input checked="" type="checkbox"/>

Figure 11: Variable list with active forcing

All variables of a DCU program which are mapped to a module function can be forced. That means that the value can be manually modified from the web interface directly.

Input variables (variables which are filled with data from the module) can be forced to a specific value which is then read by the DCU program. So input data to a DCU program can be manipulated e.g. for testing reasons.

Output variables (variables which are written by a DCU program to modify the module state) can be forced to a specific value which is directly passed to the module function, the variable is mapped with. So this function can be directly manipulated. Output variables are only processed if the DCU is in RUN state.

To start variable forcing, the user clicks on the button marked "F" in the program information section. The buttons turn to "X". A click on the button again will end the variable forcing.

All variables in the list will get an additional "F" button in the value column. A click on this button opens a small dialog which allows the user to enter a force value for this variable. Bit variables only offers a "0" and "1" button, since bit variables only can be set to 0 or 1.

For Int Variables a number can be entered.

The button marked with "X" will terminate the forcing for this variable immediately.

A forced variable in the list is marked with a yellow background.

## 8 DCU Program and Mapping batch upload

The DCU program and mapping files can also be uploaded directly via an http POST request. An example Perl script is provided which can be directly used for batch upload.

### 8.1 POST request for uploading files

URI	/upload?cmd=store&fullpage=false	
Method	POST	
MIME Type	multipart/form-data	
Form fields	path	dcu
	submit	upload
	file	[file to upload] (as application/octet-stream)
Filename	dc.int for program file dc.map for mapping file	



## 8.2 Using the Perl script

A Perl script (transfer.pl) for batch uploading files to the DCU is provided with a sample batch file which shows how to use it.

This line uploads a dcu program file (dc.int) to the module with the IP-Address 192.168.1.20:

```
perl -w .\transfer.pl -s dc.int -t dcu 192.168.1.20 -a IO-Device:admin:private
```

The same for the mapping file:

```
perl -w .\transfer.pl -s dc.map -t dcu 192.168.1.20 -a IO-Device:admin:private
```

Eventually, username and / or password must be adjusted to match the real module configuration. Any user with at least "WRITE" privileges can be used.

IO-Device is the realm name and should be left unchanged.

## 9 Standard JSON Module Information

The LioN-P modules offer a machine readable interface for the most important data. This data can be obtained via a http GET request. The answer is a JSON object.

In case of a DCU this data also contains the DCU public and exchange variables.

Request Method	http GET
Request URI	/info.json
Response Format	JSON

### 9.1 Example JSON response

```
{"name": "0980 ESL 390-121-DCU1", "fw-version": "V2.1.0.2-2.0",  
 "hw-version": "V1.0", "mac": "3C:B9:A6:00:17:00", "bus": 0, "failsafe": 0,  
 "inputs": [3, 0], "outputs": [0, 0], "consuming": [0, 0], "producing":  
 [0, 0], "diag": [0, 0, 0, 0], "dcu": {"state": 1, "autostart": 0, "public":  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
 "consuming_bits": [0, 0], "consuming_ints": [0, 0, 0, 0, 0, 0, 0],  
 "producing_bits": [0, 0], "producing_ints": [0, 0, 0, 0, 0, 0, 0]}
```

## 9.2 JSON Response object structure

```
▼ object {11}
  name : 0980 ESL 300-121-DCU1
  fw-version : F10014
  hw-version: : 4.1
  mac : 3C:B9:A6:00:17:00
  bus : 0
  ▶ inputs [2]
  ▶ outputs [2]
  ▶ consuming [2]
  ▶ producing [2]
  ▶ diag [4]
  ▼ dcu {7}
    state : 0
    autostart : 0
    ▶ public [6]
    ▶ consuming_bits [2]
    ▶ consuming_ints [8]
    ▶ producing_bits [2]
    ▶ producing_ints [8]
```

## 9.3 JSON Response Description

Fieldname	Datatype	Description
name	String	Name of the module
fw-version	String	Firmware Version
hw-version	String	Hardware Version
mac	String	MAC Address of the module
bus	Number	0 = Not Connected to fieldbus 1 = Connected to fieldbus
failsafe	Number	0 = Normal output operation 1 = Outputs in failsafe state
inputs	Number[2]	LSB = Physical input state Port X1-X4 MSB= Physical input state Port X5-X8
outputs	Number[2]	LSB = Physical output state Port X1-X4 MSB= Physical output state Port X5-X8
consuming	Number[2]	Consuming data from PLC
producing	Number[2]	Producing data to PLC
diag	Number[4]	Contains diagnostic information of the module Byte 0: <ul style="list-style-type: none"> <li>▶ Bit 0 = System/Sensor voltage supply fault (U<sub>S</sub>)</li> <li>▶ Bit 1 = Actuator voltage supply fault (U<sub>L</sub>)</li> <li>▶ Bit 2 = Sensor short circuit detected</li> <li>▶ Bit 3 = Actuator overload</li> <li>▶ Bit 6 = Forcemode active</li> <li>▶ Bit 7 = Internal module fault (IO data invalid!)</li> </ul> Byte 1 = Sensor short circuit port 1-8 Byte 2 = Actuator short circuit port 1-4 (Channel A, B) Byte 3 = Actuator short circuit port 5-8 (Channel A, B)
dcu	Object	<b>(only available on DCU modules)</b>

Fieldname	Datatype	Description
dcu/state	Number	Current state of the DCU: <ul style="list-style-type: none"> <li>▶ 0 = LOCKED</li> <li>▶ 1 = NO PROGRAM</li> <li>▶ 2 = DISABLED</li> <li>▶ 3 = STOP</li> <li>▶ 4 = RUN</li> <li>▶ 5 = ERROR</li> </ul>
dcu/autostart	Number	Is 1 if the local autostart is enabled
dcu/public	Number[32]	Contains all values of the DCU public variables _P0 - _P31
dcu/ consuming_bits	Number[2]	16 dcu exchange bits set by PLC
dcu/producing_bit	Number[2]	16 dcu exchange bits set by DCU program
dcu/ consuming_ints	Number[8]	16 dcu exchange words (16 bit signed integer) set by PLC
dcu/producing_ints	Number[8]	16 dcu exchange words set by DCU program