Training document for the company-wide automation solution Totally Integrated Automation (T I A)

MODULE D2

AS- Interface / integration of a LOGO! 12/24RC logic module and AS-I interface module

We thank the company Michael Dziallas Engineering and the instructors of vocational schools as well as further persons for the support with the production of the document.

PAGE:

1.	**Forward**	5
2.	**Notes for the Operation of LOGO! Logic Modules**	7
3.	**The First Steps with LOGO!**	8
3.1	Terminals	8
3.2	Terminals from LOGO!	8
3.3	LOGO! knows the following terminals	9
3.4	Blocks and block numbers	9
3.5	Blocks	9
3.6	Logical operations	9
3.7	Block representation in display from LOGO!	10
3.8	Assignment of a block number	10
3.9	The 4 golden rules for the servicing of LOGO!	11
3.10	Overview of the menus from LOGO!	12
4.	**Example Exercise Sliding Gate Control**	13
4.1	Requirements for the gate control	13
4.2	Wiring of the gate control with LOGO! 12/24RC	14
4.3	Used components	14
4.4	Central drive and monitoring of more industrial gates	15
4.5	Layered control over the AS-Interface	15
4.6	Function block diagram of the LOGO! solution	16
5.	**Program input in LOGO!**	17
5.1	Changing into the programming operation mode	17
5.2	LOGO! changes in the program menu	17
5.3	Program entering	18
5.4	Parameterizing of a block	19
5.5	LOGO! switched into RUN	22

PAGE:

6.	**Programming of the SIMATIC S7-300**	23
6.1	New project generation	23
6.2	Hardware configuration entering	25
6.3	Driving the sliding gate over the AS-Interface	26
6.4	Control program for the driving of the sliding gate	27
6.5	Entering organizational block OB1	28

The following symbols stand for the specified modules:

Information

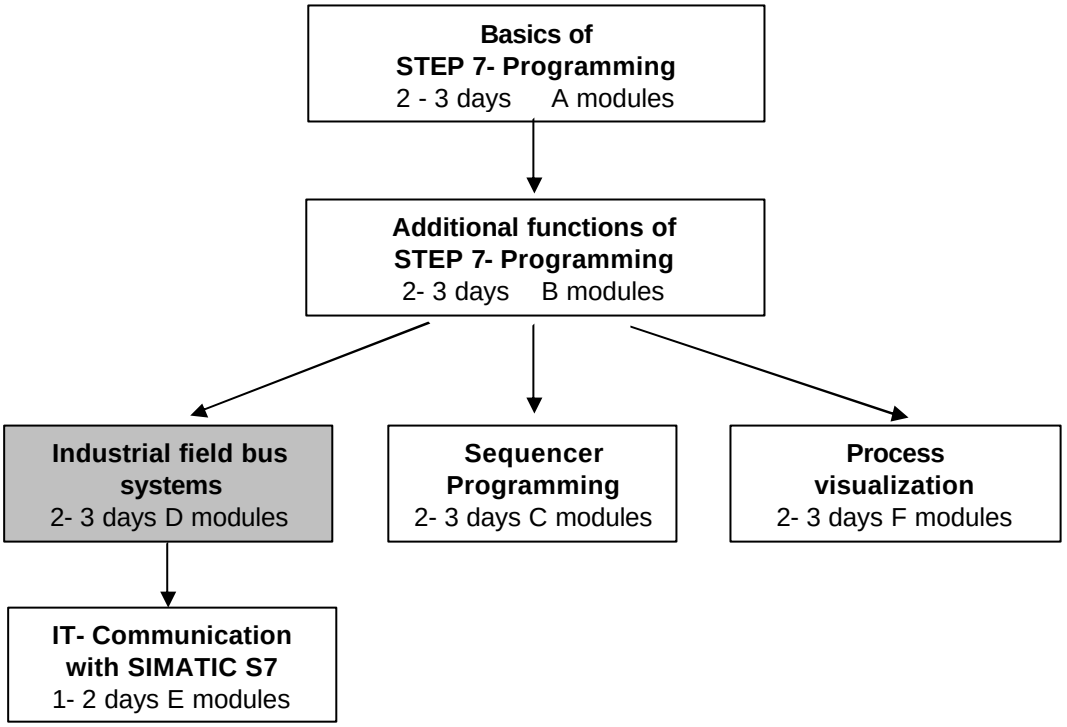
Programming

Example exercise

Notes

1. FORWARD

The module D2 is assigned content wise to **Industrial field bus systems**.



Learning goal:

In this module, the reader should learn the basic functions of the LOGO! logic module 24RCLB11 with integrated interface for the AS-Interface.
Typical task positions are executed by a pattern machine and an example project is processed in the following steps:

- Generation of a program for the LOGO! 12/24RC logic module
- Debugging of task position in RUN mode of LOGO!
- Generation of a project for a PLC SIMATIC S7-300
- Integration of ASi information from the LOGO! 12/24RC logic module in the control program of the SIMATIC S7-300CPU
- Debugging of the task position with a PLC SIMATIC S7- 300 and the LOGO! 12/24RC

Requirements:

For the successful use of this module, the following knowledge is assumed:

- Knowledge in the use of Windows 95/98/2000/ME/NT4.0
- Basics of PLC- Programming with STEP 7 (e.g. Module A3 - 'Startup' PLC programming with STEP 7)
- Commissioning of the AS-Interface with SIMATIC S7-300 (e.g. Module D1 – AS- Interface with the SIMATIC S7-300 and the CP342-2)

Required hardware and software

1 PC, Operating system Windows 95/98/2000/ME/NT4.0 with
- Minimal: 133MHz and 64MB RAM, approx. 150 MB free hard disk space
- Optimal: 500MHz and 128MB RAM, approx. 150 MB free hard disk space

2 Software STEP 7 V 5.x

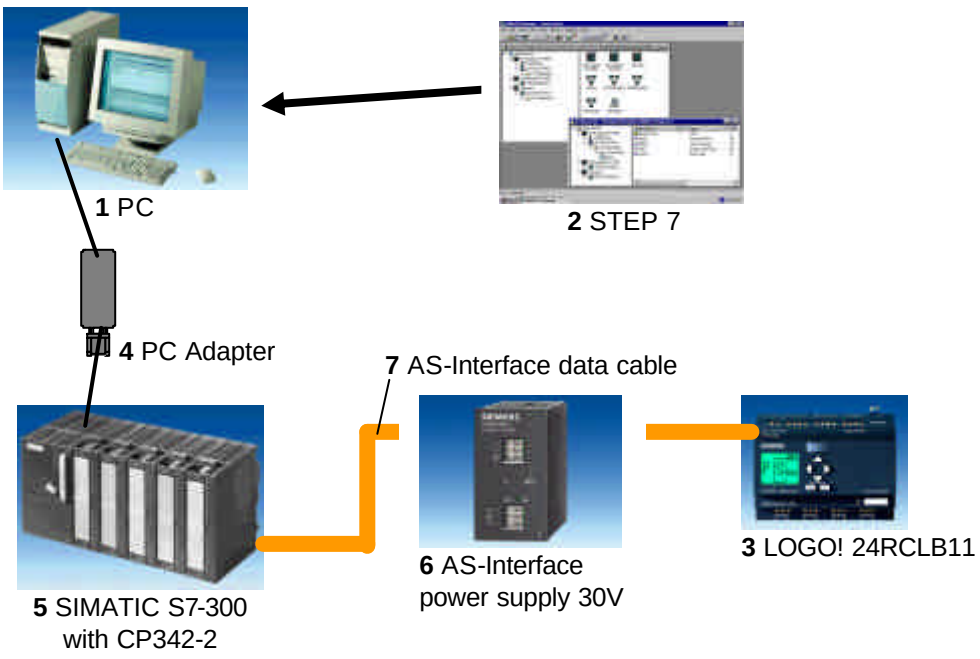
3 LOGO! 12/24RC logic module

4 MPI- Interface for the PC (e.g. PC- Adapter)

5 PLC SIMATIC S7-300
Example configuration:
- Power supply: PS 307 2A
- CPU: CPU 314
- Digital inputs: DI 16x DC24V
- Digital outputs: DO 16x DC24V / 0.5 A
- CP 342-2 AS-Interface

6 AS-Interface power supply 30V

7 AS-Interface yellow data cable



2. NOTES FOR THE OPERATION OF LOGO! LOGIC MODULES

LOGO! is the universal logic module from Siemens.

In LOGO!, the control is integrated with service- and display units.
With the service- and display units from LOGO!, you can generate programs and edit and execute system functions.

External programs over an interface from a program module or over a PC cable can be read from the programmed software LOGO!-SOFT. In addition to programming software, you can also execute a simulation of your circuit by the computer or print out overview plans.

Finished practical use basic functions (e.g. for time-delay on switch, time-delay off switch and current surge relay, time switch, binary memory bit, such as in- and outputs for device type) are already contained in the LOGO! logic module.

You solve tasks with LOGO! :

- In the house- and installation technology (e.g. house stairway lighting, outside lights, marquees, shutters, shop window lighting etc.),

- in an electrical panel builder and in machines- and apparatus constructions (e.g. gate controls, ventilation systems, industrial water pumps, etc.).

Furthermore LOGO! can be set for special control to signal preprocessing. Through the Asi-Variant, the application as a distributed peripheral is possible with separate intelligence from location for the controlling of machines and processors. Through this, control tasks can be accomplished in the LOGO! logic module in order to release the master control.

There are special variants without service units for serial uses in small machines and apparatus constructions, in an electrical panel builder and installation range. These must be downloaded over a program module or over the PC software LOGO-SOFT.

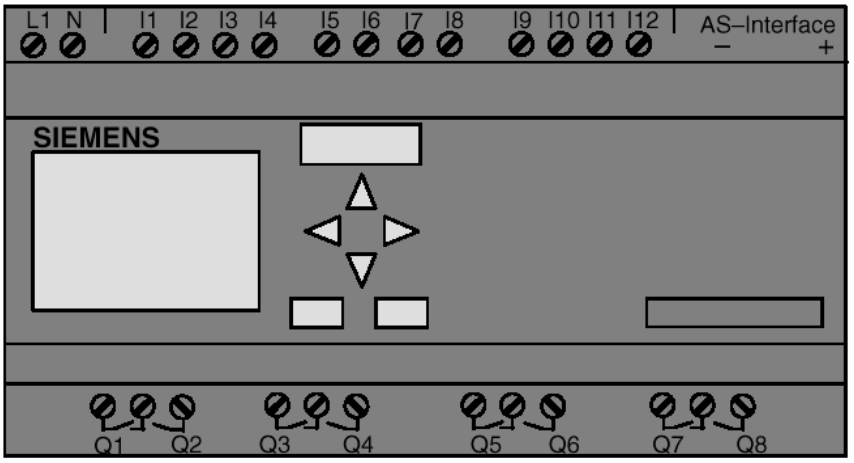
3. THE FIRST STEPS WITH LOGO!

We indicate the inputs of a circuit with programming. A LOGO! program is not really any different from a circuit diagram besides the way that it is represented. We customized the representation from the display field from LOGO!. In this chapter, we introduce to you how you convert your applications into LOGO! programs with LOGO!.

Next we introduce to you the 2 fundamental terms **terminal** and **block** and show you what is contained behind them.
In a second step, we develop a program together from a simple, conventional circuit that you can enter directly into LOGO! in the third step. After reading a few pages from the handbook, your first program will be stored execution capable in LOGO!. With the proper hardware (circuit...) you can already execute the first tests.

3.1 Terminals

LOGO! contains inputs and outputs:



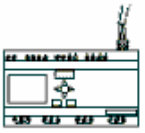
The inputs are identified with the letter I and a target. When you look at LOGO! from the front, you see the terminals for the inputs on top. The outputs are identified with a Q and a target. The terminals of the outputs can be seen on the bottom of the picture.

3.2 Terminal from LOGO!

We identify all connections and states that are found in LOGO! applications. The in- and outputs can contain the state ‘0‘ or ‘1‘. State ‘0‘ means that no voltage is applied to the inputs. State ‘1‘ means that voltage is applied. This information should be nothing new to you. We have imported the terminals hi, lo and x in order to lighten the program input for you. ‘high‘ (high) is the state ‘1‘ and ‘low‘ (low) is the state ‘0‘. When you do not want to activate an input of a block, then use the terminal ‘x‘. You will learn about what a block is on the next page.

Note
In- and outputs that are available by LOGO!...B11 over the AS-Interface bus connection are not direct physical inputs to LOGO!. Please note that the bus master appoints the in –and output devices to the AS-Interface.

3.3 LOGO! Knows the Following Terminals

Terminals			
Inputs	I1... I6I, I7 (AI1), I8 (AI2)	I1...I12	I1...I12 and Ia1...Ia4 (AS-Interface)
Outputs	Q1...Q4	Q1...Q8	Q1...Q8 and Qa1...Qa4 (AS-Interface)
lo	Signal with level '0' (Off)		
hi	Signal with level '1' (On)		
x	An available connection is not used		

3.4 Blocks and Block Numbers

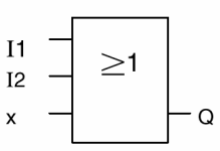
In this chapter we introduce to you how you can generate extensive circuits with the elements of LOGO! and how the blocks are operated with one another and with the in- and outputs.

3.5 Blocks

A block in LOGO! is a function which converts the input information into output information. Earlier you must have wired the individual elements in the electrical panel or junction box. By the programming you connect terminals with blocks. Therefore you simply choose from the menu **Co** the desired connection. The menu **Co** is named after the english term connector (terminal).

3.6 Logical Operations

The simplest blocks are logical operations (e.g. AND, OR).

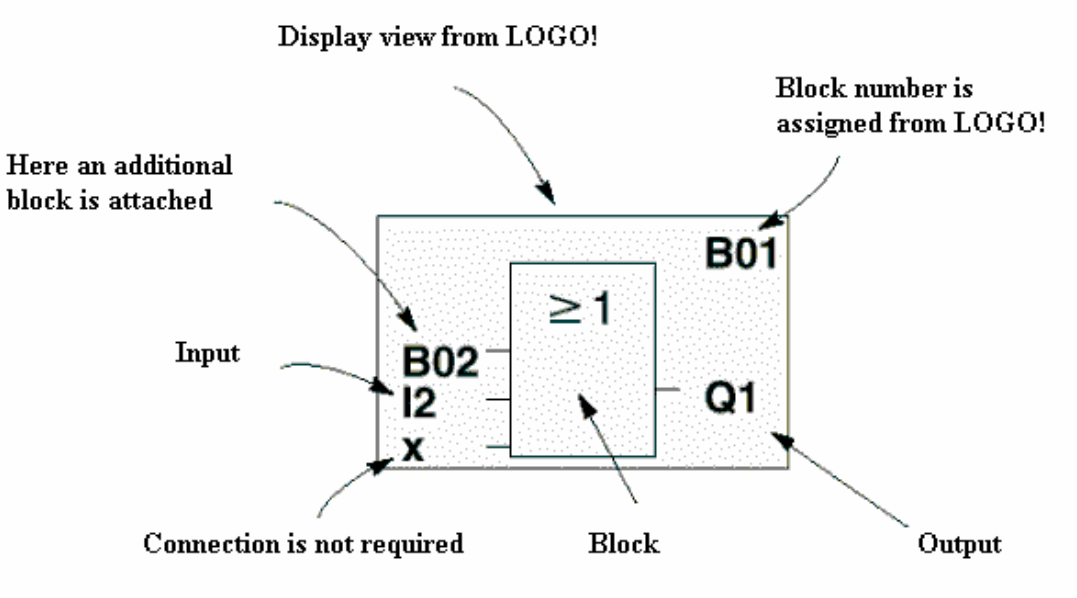


Here are the inputs I1 and I2 connected to the OR-block.
The last input of the block is not used so it is set with x.

The other special functions are just as effective (e.g. current surge relays, counters, timers)

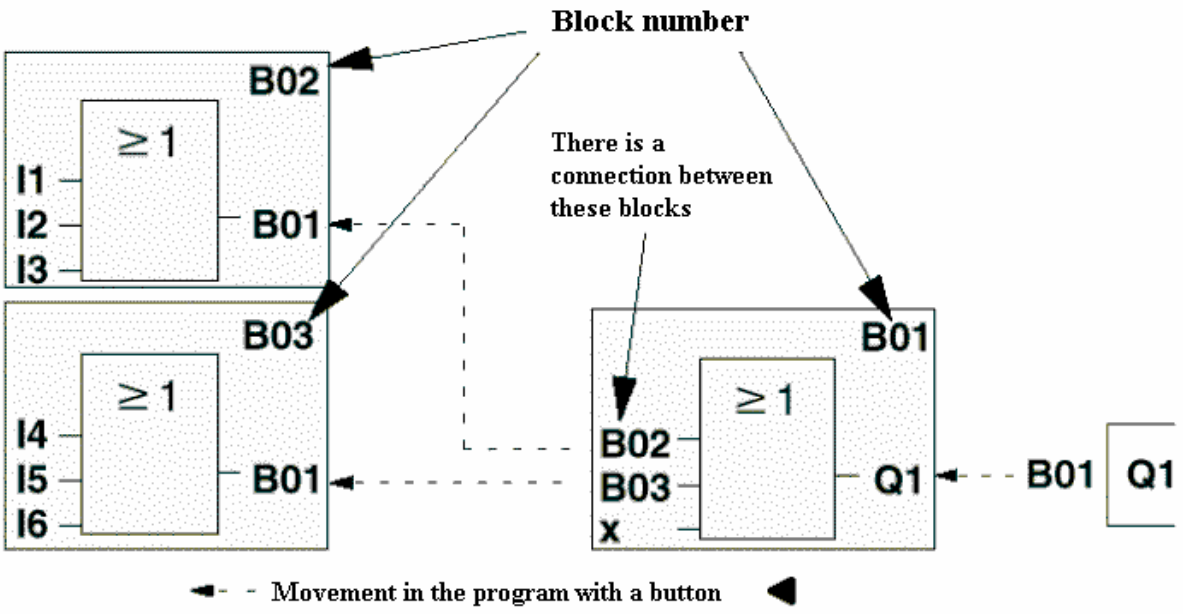
3.7 Block Representation in the Display from LOGO!

In the picture we show you a typical display Display from LOGO!. It is always only a block representation. Therefore we imported the block numbers that should help you to control the circuit relation.



3.8 Assignment of a Block Number

Whenever you insert a block into a program, LOGO! gives this block a block number. LOGO! shows you the connection between the blocks over the block numbers. The block numbers only serve for orientation help in the program.

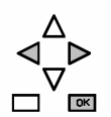


In the overview picture you see 3 display views from LOGO! that together make the program. The blocks from LOGO! are connected to one another over the block numbers.

Forward Notes **First steps with LOGO!** Example exercise Program LOGO! Program S7-300

3.9 The 4 Golden Rules for the Servicing of LOGO!

Rule 1 - the 3-Finger-Handle



Enter the circuit in the operation mode programming. You arrive in the programming operation mode in which you push simultaneously the 3 buttons **Cursor left**, **Cursor right** and **OK**.
You modify values of times and parameters in the parameter assignment operation mode. You arrive in the parameter assignment operation mode in which you simultaneously push the 2 buttons **ESC** and **OK**.

Rule 2 – Outputs and inputs

You always enter a circuit from output to input.

You can connect an output with more inputs, but you can not switch more outputs onto an input. You cannot connect an output with a proceeding input inside of the program path. Switch between memory bits or outputs for each internal feedback.

Rule 3 - Cursor and cursor movement

By the entering of a circuit, it applies:

If the cursor is represented as an underline then you can **move** the **cursor**.

– With the buttons , Ü , Þ , Ý or ß you move the cursor in the circuit
– With **OK** you change to the "Terminal/Block choice"
– With **ESC** you leave the input mode of the circuit.

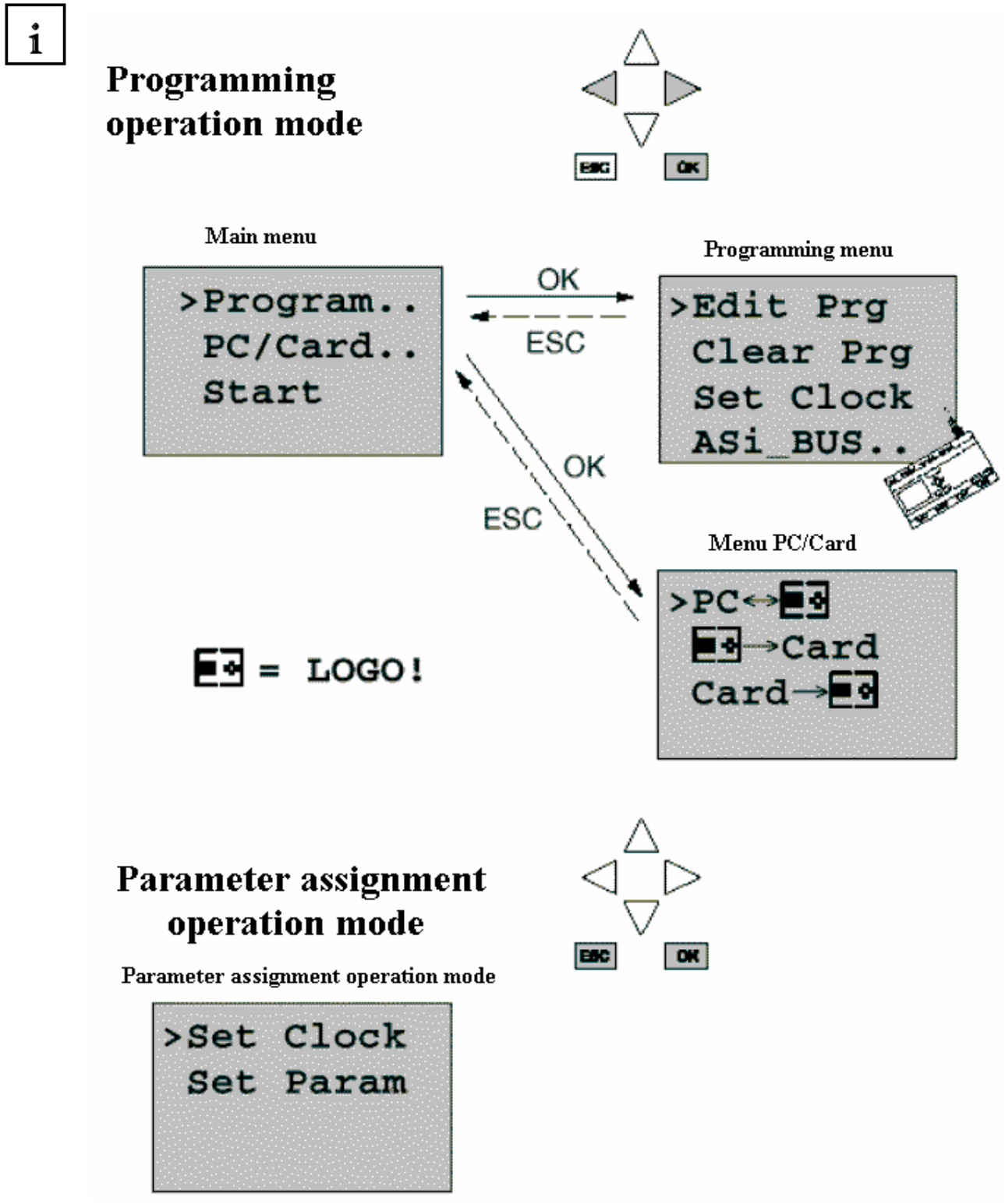
If the cursor is represented as a full block then you should **choose** a **Terminal/Block**.

– With the buttons Ý or ß you choose a Terminal/a Block
– With **OK** you accept the choice
– With **ESC** you arrive a step back

Rule 4 – Planning

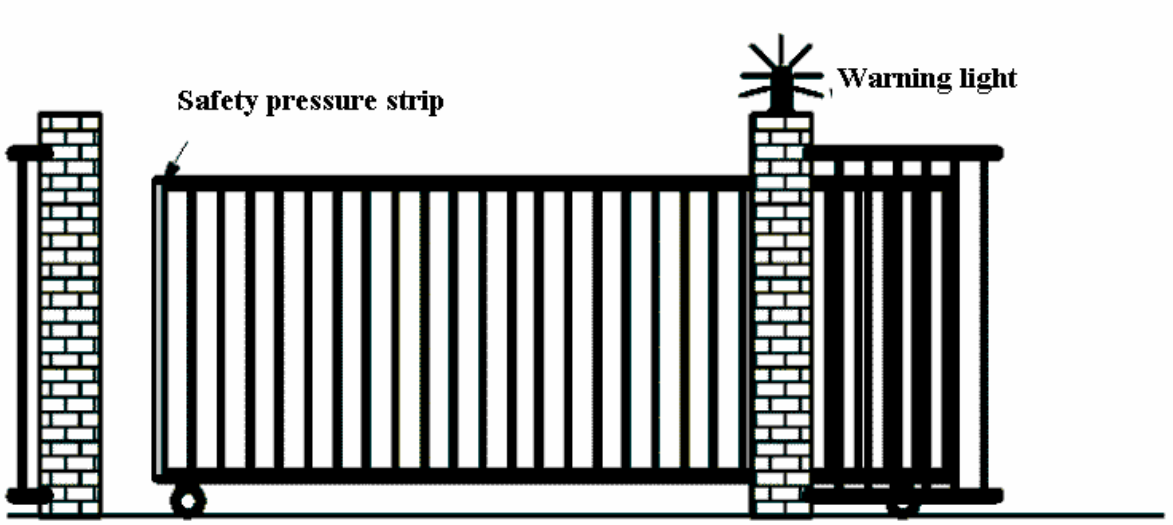
Before the entering of a circuit you should plan the entire next step on paper or program LOGO! directly with a LOGO!Soft Comfort.
LOGO! can now save an entire program.
When a circuit is not entirely entered, LOGO! can not leave the **programming** operation mode.

3.10 Overview of the Menus from LOGO!



4. EXAMPLE EXERCISE SLIDING GATE CONTROL

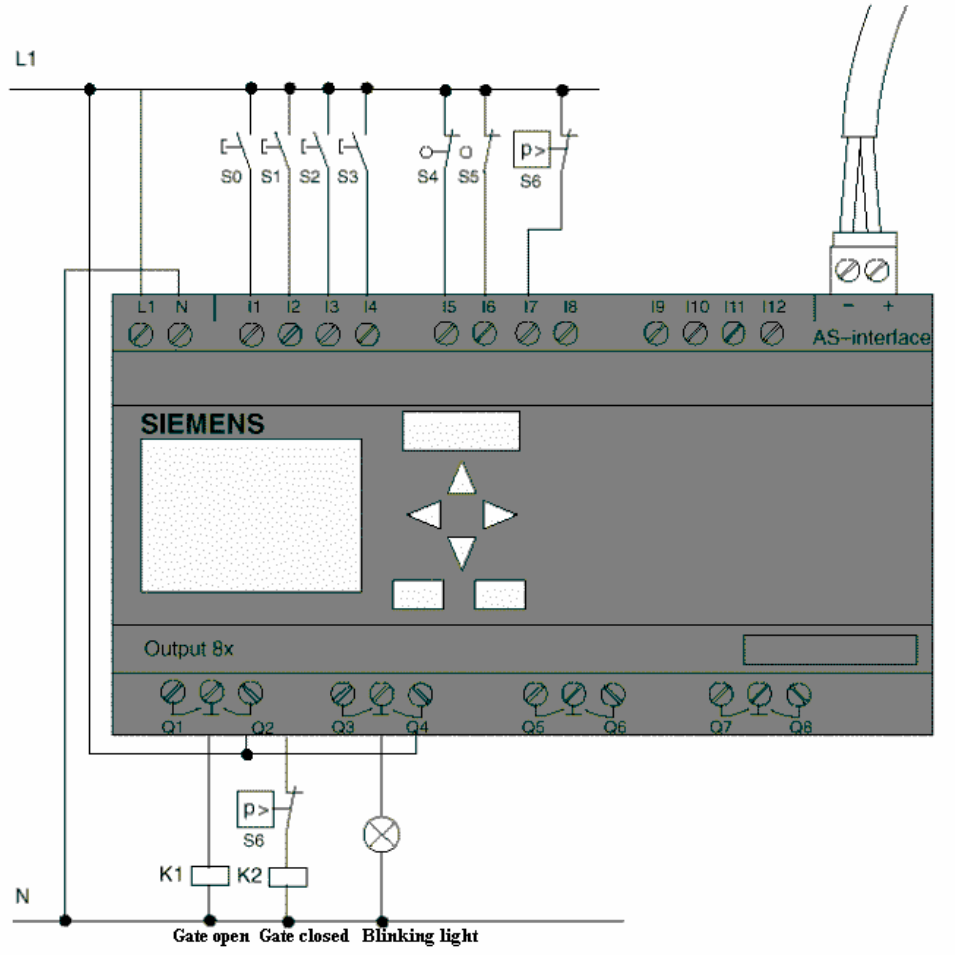
Access to a company premesis is in many cases possible from different locations. Not all gates can always be watched by Personel. Therefore they must be operated and supervised from a central control room. Additionally, it must be guaranteed that the personal can open and close a gate directly. For each gate, a LOGO! 12/24RC is used.
Over the AS-Interface, the modules are linked with one another and operated with an AS-I master. We describe to you a gate controller for a gate in this section. The other gate controllers are identically built.



4.1 Requirements for the Gate Control

Each gate is opened or closed by means of a cord operated switch. The gate is therefore opened and or closed completely. The gate is opened and closed from the gate keeper logs over the AS-I bus connection. The state GATE OPENED or GATE CLOSED is displayed. A blinking light is turned on for 5 seconds before the beginning and during the movement of the gate. It is guaranteed by a safety pressure strip that no people are injured or things are crushed by the closing of the gate.

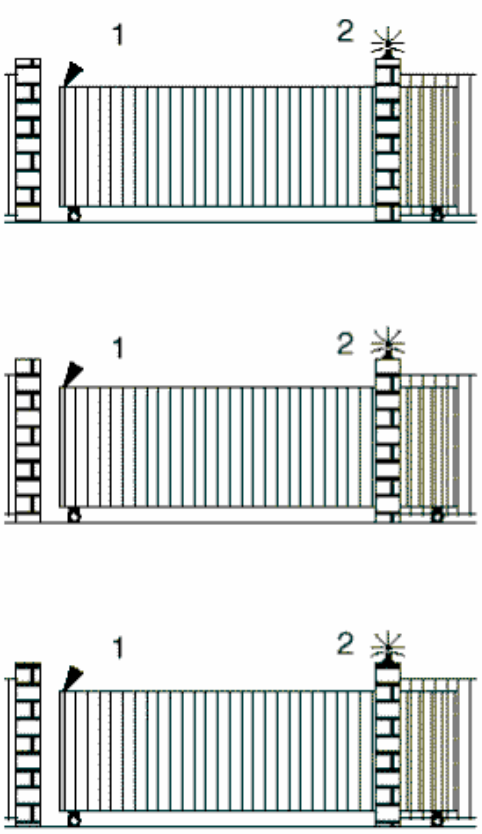
4.2 Wiring of the gate control with the LOGO! 24RCLB11



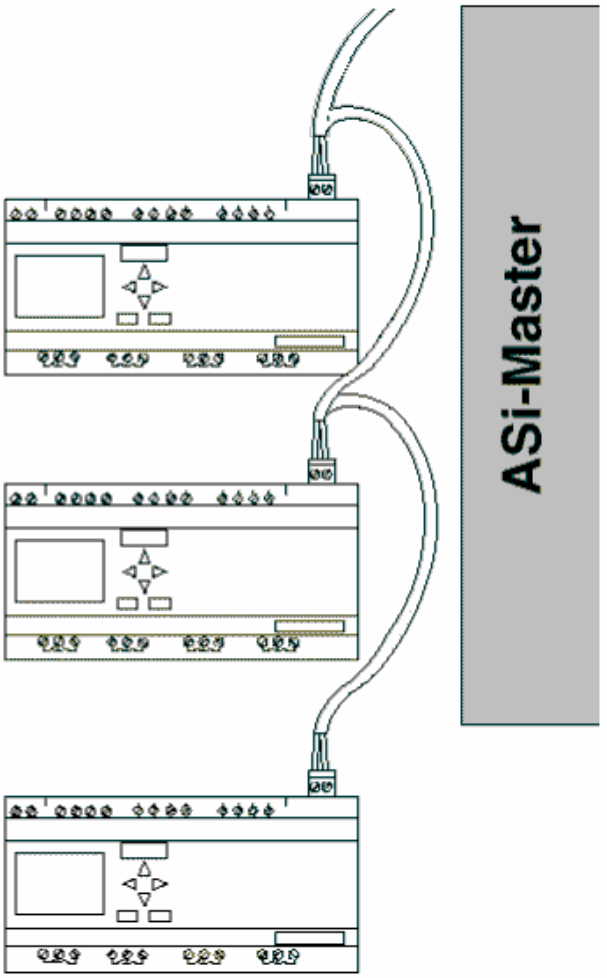
4.3 Used Components

K1 Main contact open
K2 Main contact closed
S0 (Normally open) Cord switch GATE-OPEN
S1 (Normally open) Cord switch GATE-CLOSE
S2 (Normally open) Button GATE-HAND-OPEN
S3 (Normally open) Button GATE-HAND-CLOSE
S4 (Normally closed) Position switch GATE OPENED
S5 (Normally closed) Position switch GATE CLOSED
S6 (Normally closed) Safety pressure strip

4.4 Central Drive and Monitoring of more Industrial Gates



1 Safety pressure switch
2 Blinking light

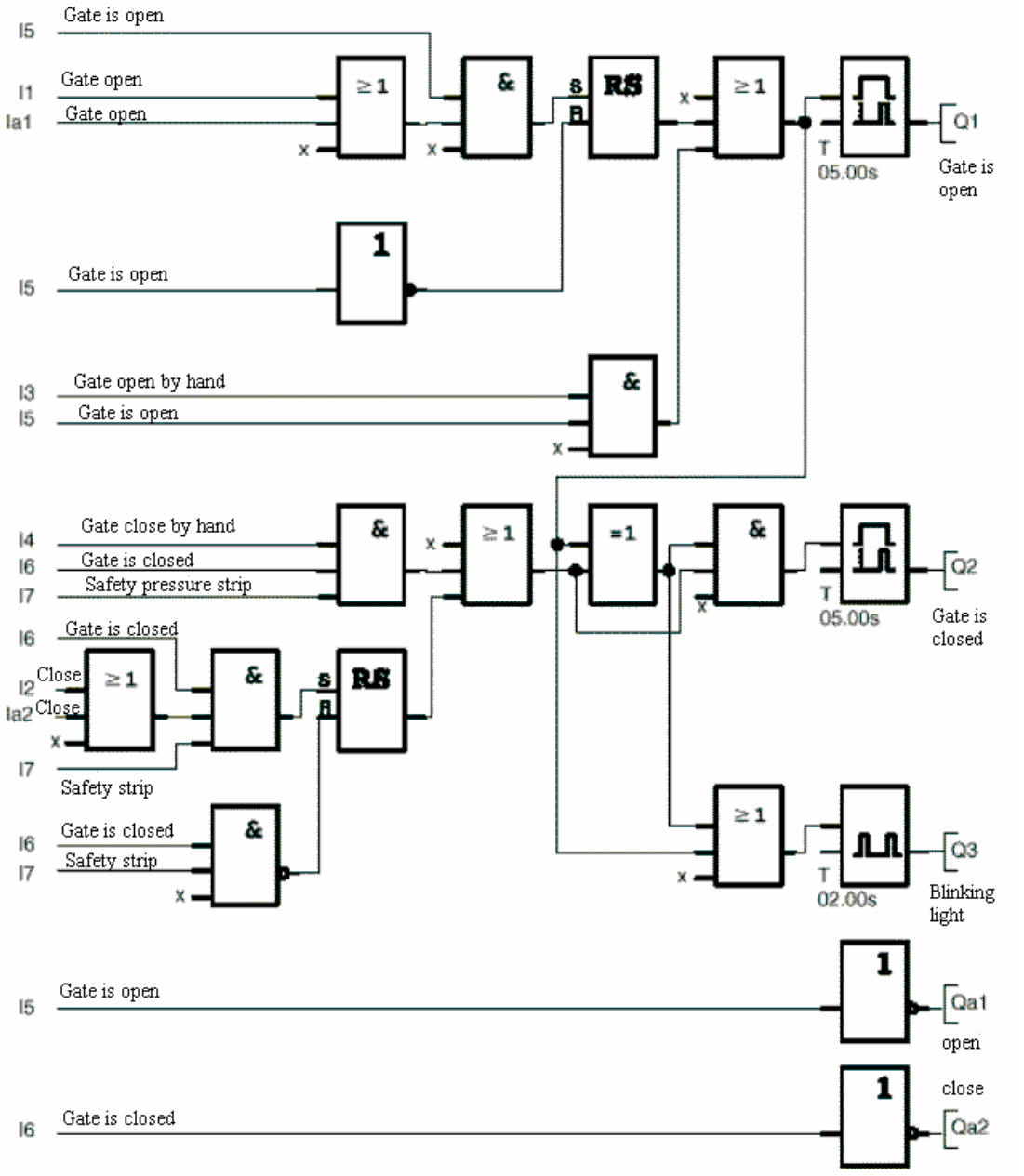


4.5 Layered Control over the AS-Interface

Qa1 Position switch GATE OPENED
Qa2 Position switch GATE CLOSED
Ia1 External button GATE OPEN
Ia2 External button GATE CLOSE

Note
After the AS-I slave address in the LOGO!RCLB11 is used, a different address area is assigned in the AS-I-Master.

4.6 Function Block Diagram LOGO! Solution



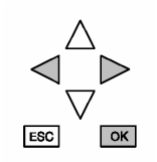
Through the start button GATE OPEN or GATE CLOSE, the moving of the gate is instructed as long as the opposing direction is not switched on. The end of the movement occurs by the end switch. The closing of the gate is also broken by the safety strip.

5. PROGRAM INPUT IN LOGO!

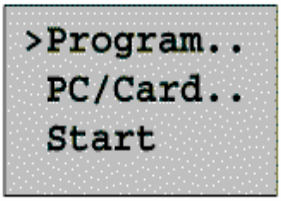
You composed a circuit and now would like to enter it into LOGO!. We will show you how this works by means of the entering of a program.

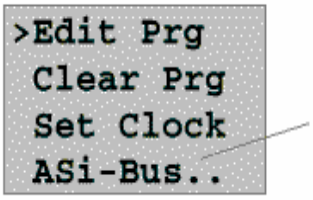
5.1 Changing into the Operation Mode "Programming"

You connected LOGO! to the network and turned on the power. On the Display, you should see the following display:



Switch LOGO! into the programming operation mode.
Then push the buttons Ü , Þ and **OK** simultaneously.





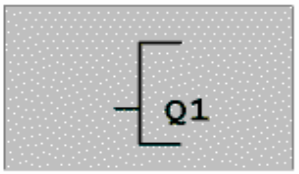
The program menu of LOGO!

The entering ASi-Bus only occurs by LOGO!...LB11-Variants

By the first activator of the first cell, you see a ">". With the cursor buttons (Ý, ß), move the ">" back and forth. Move the ">" to "Program.." and push the button **OK**.

5.2 LOGO! changes in the Programming menu

Here you can also move the ">" with the cursor buttons (Ý, ß). Activate the ">" on "Edit Prg" (for program editing) and push the button **OK**.

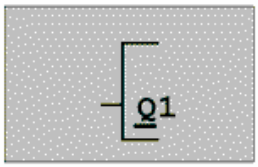


The first output of LOGO!

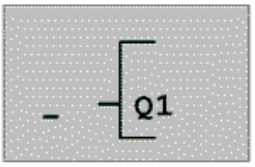
LOGO! now shows you the first output:
With the cursor buttons (Ý, ß), you can choose the other outputs. Now you begin with the entering of your circuit.

5.3 Program entering

We only input the program (and that was the output to the input). At the beginning, LOGO! shows the output:

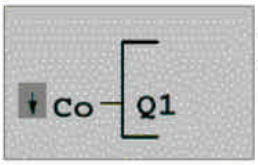


The first output of LOGO!

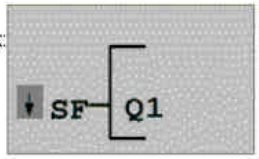


The cursor displays where you find yourself in the program.

Under the Q of Q1, you see an underscore. We call the underscore a **cursor**. The cursor shows the activator of where you find yourself in the program. You can move the cursor with the cursor buttons. Now push the cursor button **cursor left**.
By this activator you enter only the first block. You change into the input mode when you push the button **OK**.

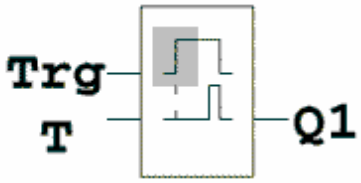


The cursor is represented as a full block: You can choose a terminal or a block



In the list SF, you find the blocks for special functions

The cursor no longer has the form of an underscore but blinks as a full block. LOGO! simultaneously offers you different choice possibilities. Choose SF (push **cursor down**, until SF appears) and push the button **OK**. LOGO! now shows you the first block from the list of the special functions (SF):



By choosing a block for a special or basic function, LOGO! shows the block of the function. The cursor is in the block and has the form of a full block. With the buttons ▼ or ▲ , you choose the desired block.

The block for the on delay contains 3 inputs. The top input is the trigger input (Trg). You start the on delay over this input. In our example, the on delay from the OR-Block B02 is started. You activate the time for the on delay over the parameter T.

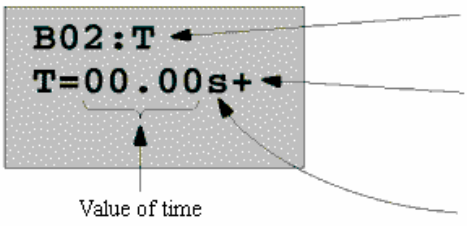
Note
Indicate the individual logical functions with the block numbers of your LOGO! program from the function block diagram (Page 10). An error search or modification of the program is facilitated.

5.4 Parameterizing a Block

Enter now the time T for the on delay:

1. When the cursor does not yet lay under the T, move it under the **T** with the help of the cursor button.
2. To change into the input mode: Push button **OK**

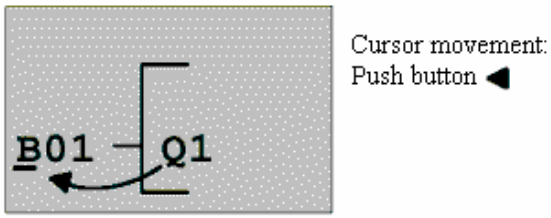
LOGO! shows the parameter window by the parameters:



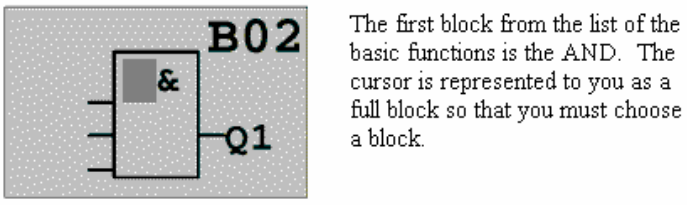
To modify the time value:

- **With the buttons ◀ and ▶ , you move the cursor back and forth**
- **With the buttons ▲ and ▼ ,you modify the value of the actuator**
- **When you have entered the time value, press the button OK**

Move the cursor under the B from B01 (B01 is the block number of the time block).
Push the **cursor left** button two time in order to position yourself **Trg**.

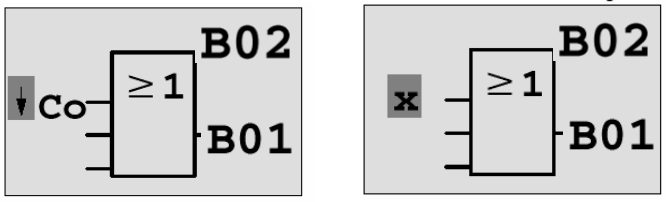


Push button **OK** .
Choose **BF** for basic functions with the **cursor down** button.
Accept with **OK** (The block B02 is displayed).



Choose an OR (OR function) with the **cursor down** button.
Accept with **OK**.

First push the **OK** button (CO appears) for the input of the OR-Block and then push **OK** one more time (x appears) and then a third time **OK.** You insert a **x** for an unused place. The cursor springs further to the second input of the OR-Block.



Push the **OK** button on the second input and change to the **SF** (special functions) with the **cursor down**.
Accept your choice with **OK**.
Choose an **RS** flipflop with the cursor button (Ý, ß) and accept with **OK**.

You are now in block B03.
Add an **AND-Block** (AND function) to the Set-input of the flipflop with **OK**, **cursor under**, **BF** for basic functions and one more time on **OK**.
Accept your choice with **OK**.

You are now in block B04.
Insert the input **I5** to the first input of the AND-Block with **OK** (CO appears) and once more with **OK** (x displays) and then push the cursor button (Ý, ß).
Accept with **OK**.
Insert an **OR-Block** (OR function) on the second input with **OK**, **cursor left**, **BF** (basic functions) and once more **OK** and **cursor down** .
Accept with **OK**.

You are now in block B05.
Insert the input **I1** on the first input of the OR-Block with **OK** (CO appears) and once more **OK** (x appears) and then the cursor button (Ý, ß).
Accept with **OK**.
Insert the input **Ia1** on the second input of the OR-Block with **OK** (CO appears) and once more **OK** (x appears) and then the cursor button (Ý, ß).
Accept with **OK**.
Insert a **x** (unused place) on the third input of the OR-Block with **OK** (CO appears) and once more **OK** (x appears) and a third time **OK**.

Block B05 is closed and you find yourself further in block B04.
Insert an **x** (unused place) on the third input of the AND-Block.

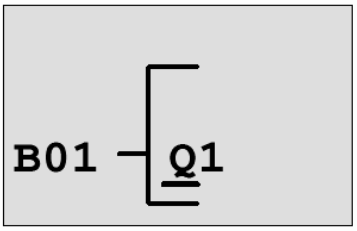
Block B04 is closed and you find yourself on the reset input in block B03.
Insert a **NOT-Block** (Negation) on the reset input with **OK**, **cursor down**, **BF** (basic functions) and once more on **OK** and two times on **cursor under**.
Accept with **OK**.

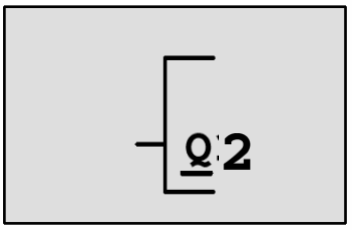
You are now in block B06.
Insert the input **I5** on the input of the NOT-Block with **OK** (CO appears) and once more **OK** (x appears) and the cursor button (Ý, ß).
Accept with **OK**.

Block B06 is closed and you find yourself on the par input in block B03.
A retentitivity of the flipflop can be set with **OK** and the cursor button (Ý, ß).
Accept with **OK**.

Block B03 is closed and you find yourself on the third input in block B02.
Insert an **AND-BLOCK** on the third input of the OR-Block with **OK**, **cursor down**, **BF** (basic functions) and once more on **OK**.
Accept with **OK**.

You are now in block B05.
Insert the input **I3** on the first input of the AND-Block with **OK** (CO appears) and once more **OK** (x appears) and then the cursor button (Ý, ß).
Accept with **OK**.
Insert the input **I5** on the second input of the AND-block with **OK** (CO appears) and once more **OK** (x displays) and the cursor button (Ý, ß).
Accept with **OK**.
Insert an **x** (unused place) on the third input of the AND-block with **OK** (CO appears) and once more on **OK** (x appears) and a third time on **OK**.
The inputs for the output Q1 are now finished.





Now choose the output Q2 with the cursor buttons (Ý, ß).
Push **cursor left** and enter the program for program Q2 (Page 10).
Notice that accesses are programmed here also from the outputs of available blocks.
You find already programmed blocks under **BN** (block number, click one time by **CO** with **cursor up**).
Enter the programs for the outputs Q3, Qa1 and Qa2.

Now the program entering for the sliding gate control in LOGO! is closed.

We now quit the program entering. This is accomplished by:

1. Return to the program menu: Push **ESC** button.

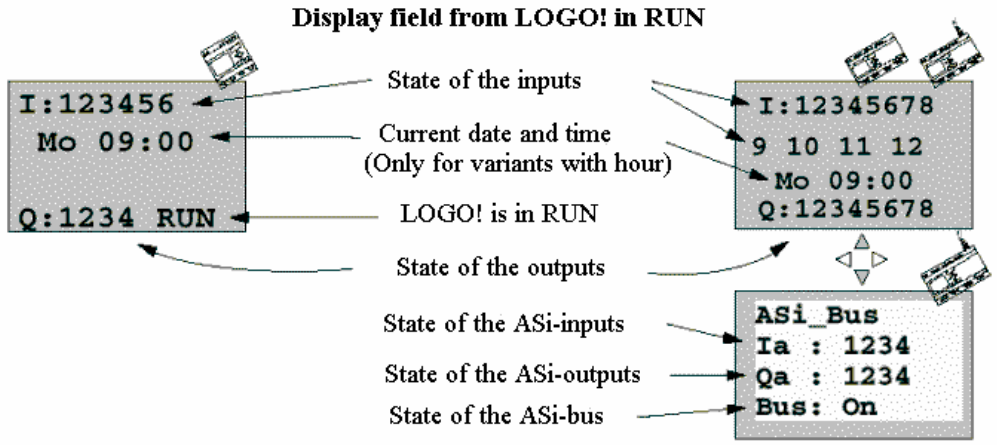
When you have not returned to the program menu then you forgot to completely toggle a block. LOGO! shows you the activator in which you forgot something in the program. (LOGO! only accepts fully completed programs).

2. Return to the main menu: Push **ESC** button.

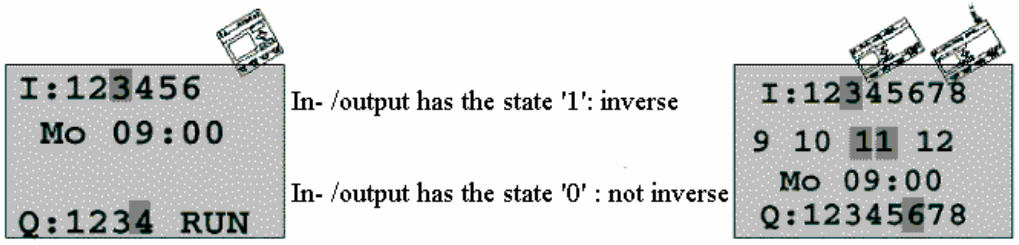
5.5 Switching LOGO! into RUN

1. Move '**>**' to 'Start' : with the cursor button (Ý, ß).
2. Accept Start: Push **OK** button .

LOGO! goes into RUN. LOGO! display the following Display in RUN:



LOGO! executes the program in RUN. Next LOGO! reads the states of the inputs, detects the states of the outputs which are associated with your entered program and switches the relays by the outputs on or off..
LOGO! displays the state of an input or an output.



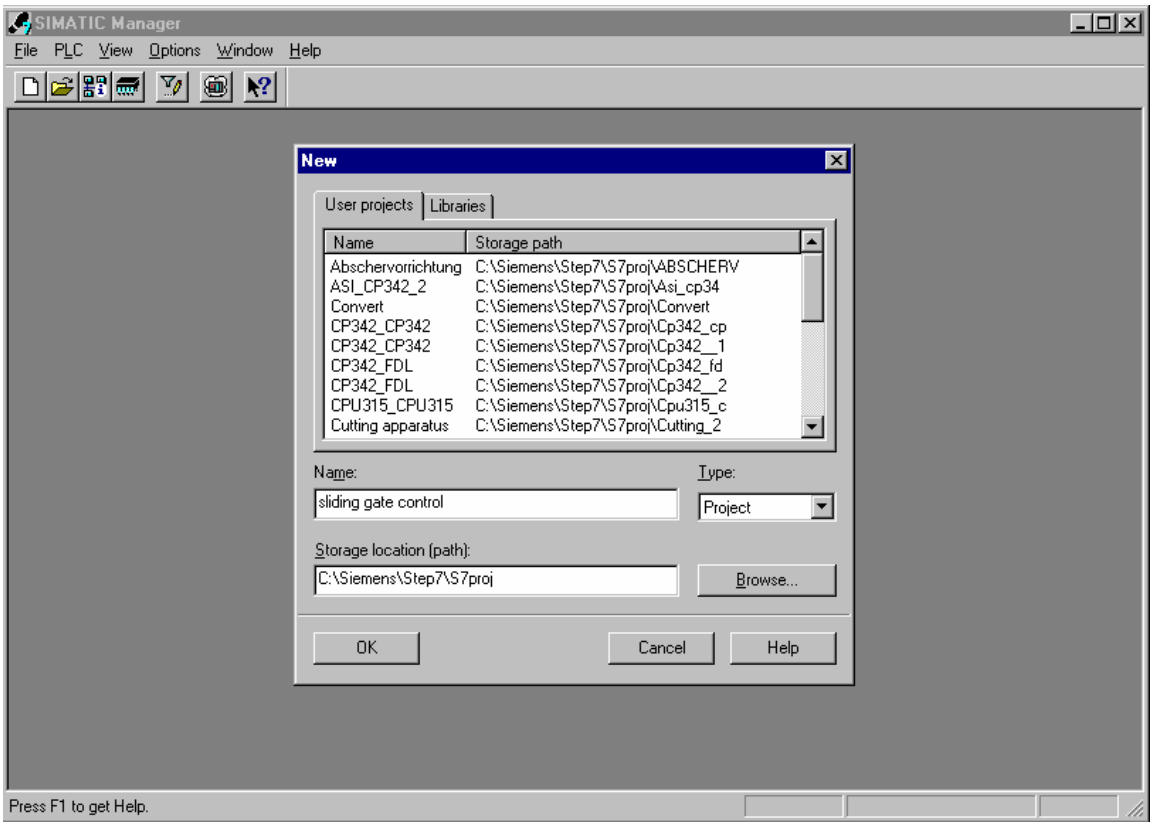
You can now test the sliding gate control on LOGO!

Note the limit switch for GATE OPEN or GATE CLOSED and the safety pressure strip have a normally closed function. With the hand button OPEN or CLOSE, the motion of the gate is processed as long as the button is activated. By the processing of the gate, the signal lamp blinks first for 5 seconds before the gate movement begins (this means that the hand button must be held for at least this long). With this setting, the gate opens or closes in automatic operation mode.

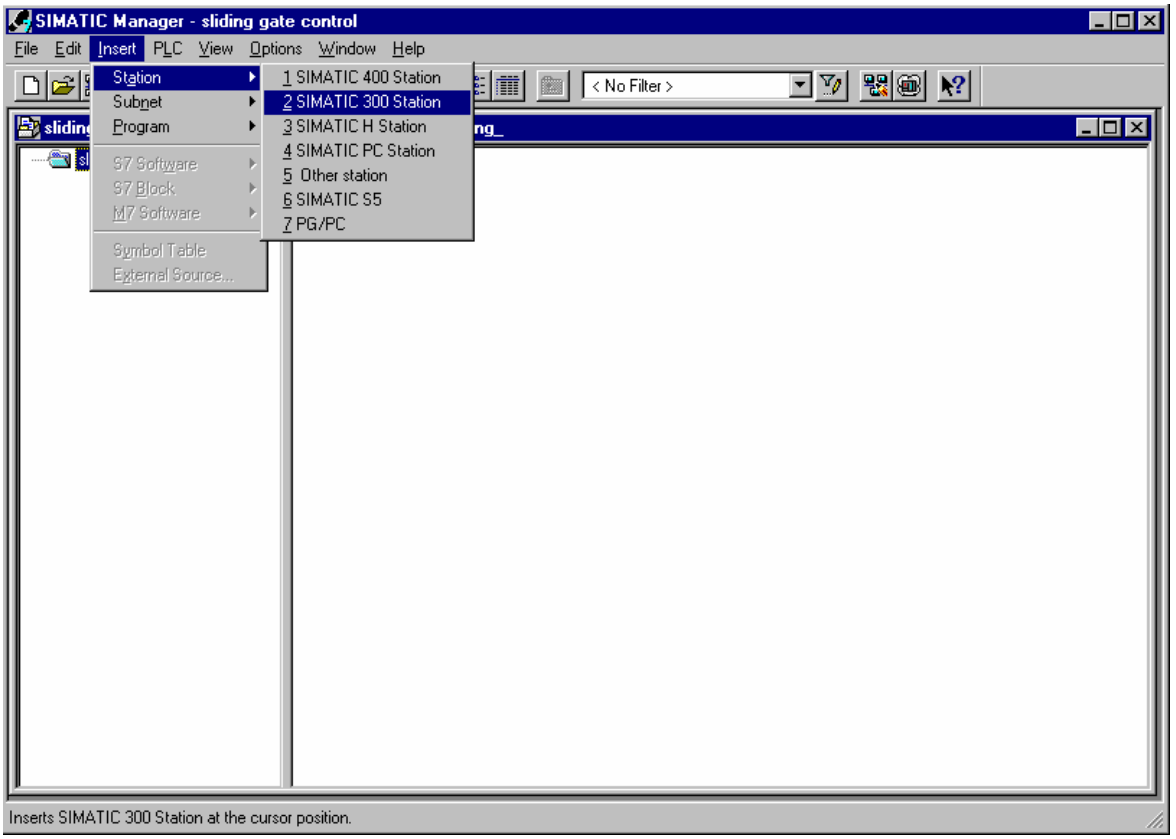
6. STEP 7 PROJECT GENERATION

6.1 New Project Generation

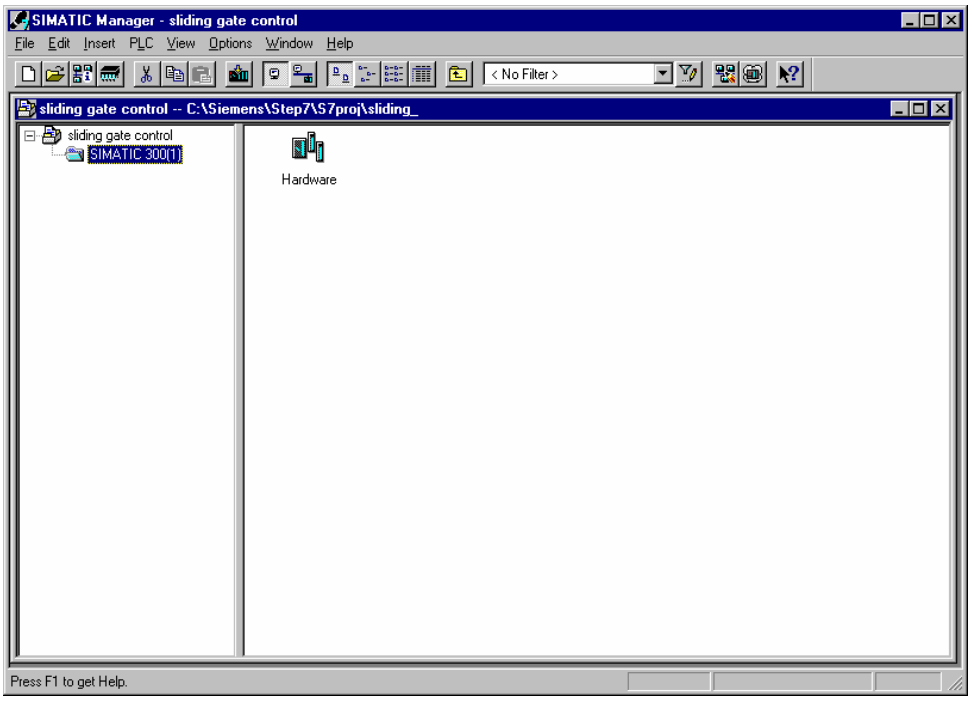
Start the **SIMATIC Manager** and generate a new project.
Input **sliding gate control** for the name.
Accept it with **OK**.



Then insert a **SIMATIC 300-Station**.



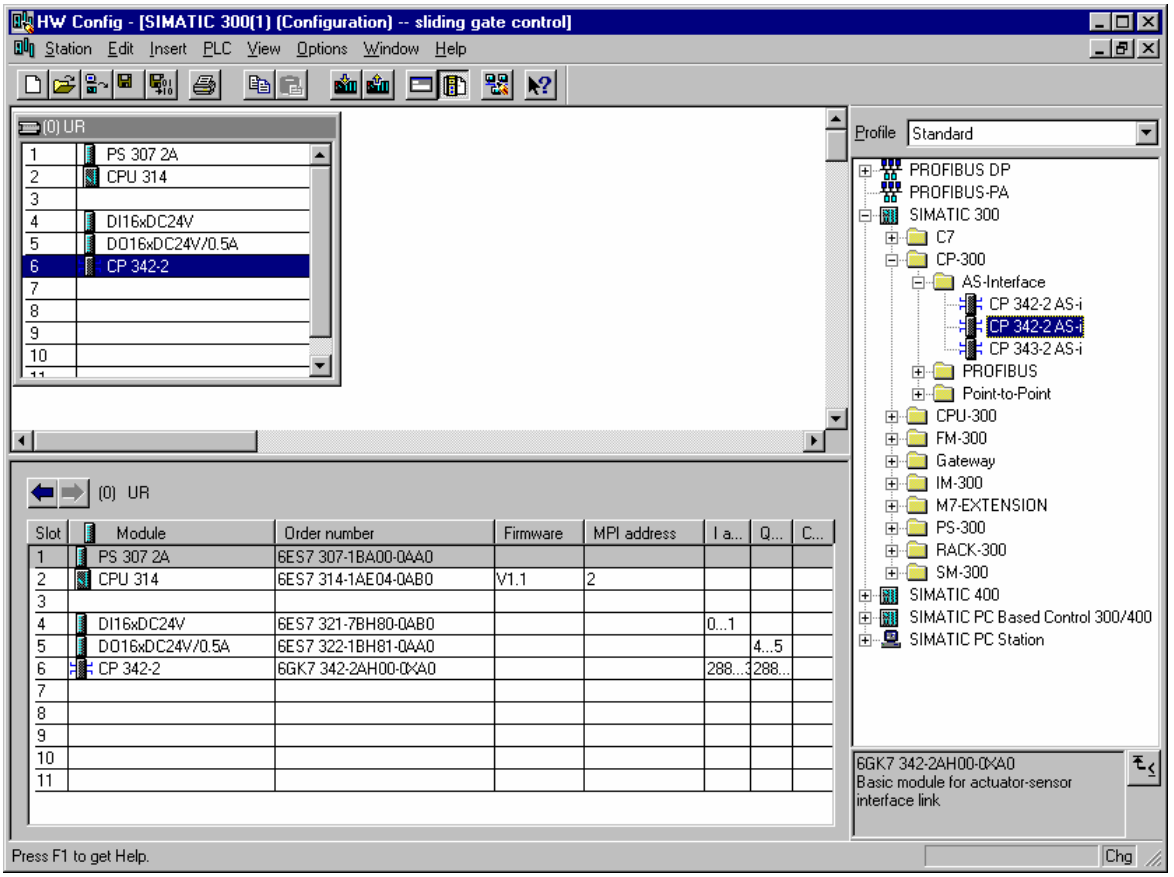
Choose **SIMATIC 300(1)** Station in the left window (first clock on the **+** by the project name sliding gate control) and open the hardware configuration through a double click on **Hardware**.



6.2 Hardware Configuration Entering

Enter the hardware configuration for your available controller.
Note the addresses for the in- and outputs and for the communication processor for the AS-Interface.

Save and compile your hardware configuration.
Download the hardware into the CPU.
Close the window for the hardware configuration.

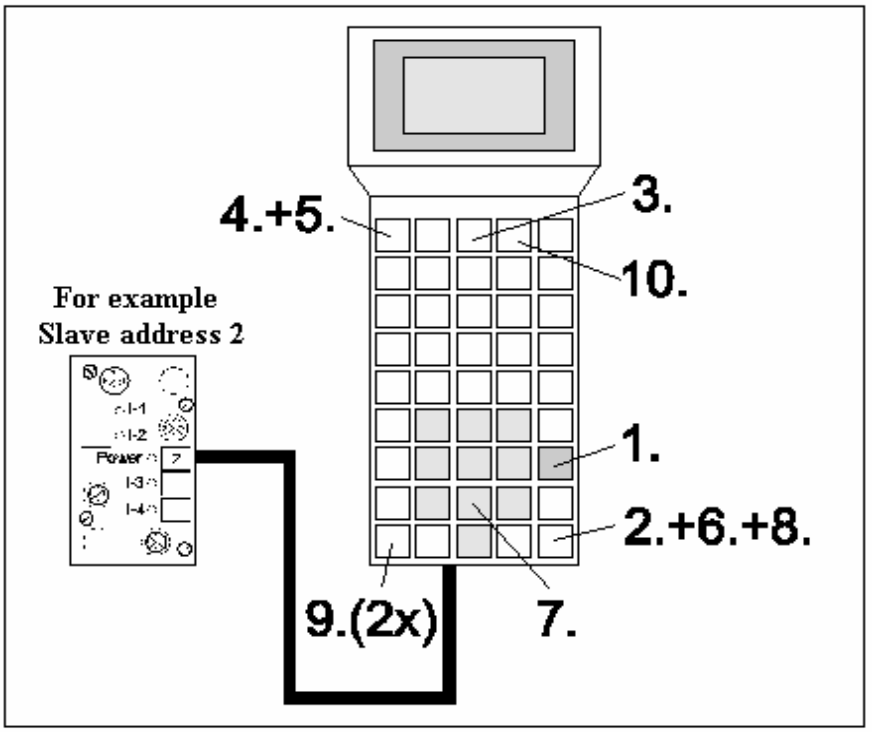


6.3 Driving of the Sliding Gate over the AS-Interface

The LOGO! 12/24RC is connected with the AS-I-Master over the AS-Interface.

One must first assign an AS-I slave address to the LOGO!- Module over an external program device. A memory area of 4 bits for the inputs and 4 bits for the outputs is allocated in the AS-I-Master for the LOGO!- Module.

In our example the LOGO!- Module is assigned the address 9.



LOGO!-Module connected with the PSG.

1. Turn on PSG (START)
2. Activation (ENTER)
3. Choose ‘Master‘ (F3)
4. Choose ‘Single operation’ (F1)
5. Choose ‘New address’ (F1)
6. Activate AS-I address (ENTER)
7. Enter new address (e.g.: 2)
8. Activate inputs (ENTER)
9. Return to the main menu (2x ESC)
10. Turn off PSG (F4)

In our program example, the opening and closing of the gate is accomplished with the button Gate1-OPEN with the input I0.0 and the button Gate1-CLOSE with the input I0.1.
The indicator light for the gate open or gate closed should be accomplished over the outputs Q4.0 and Q4.1 for the display.
The input signals of the AS-Interface should be downloaded to the input byte 10 in the process-image. This stands in the address area of the CP342-2 which can be read from the hardware configuration (here from PIW 288).
The output signals for the AS-Interface are readout to the output byte 20 from the process-image. These must be again read into the address area of the CP342-2 which can be read from the hardware configuration and then written (here to PQW288).

The following table should clear up any coherences.

Inputs	IN / OUT				IN / OUT				Address	Outputs
PII	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	CP342-2 (PI/PQ)	PIQ
	In4	In3	In2	In1	In4	In3	In2	In1		
	Out4	Out3	Out2	Out1	Out4	Out3	Out2	Out1		
10	Occupied				Slave01				288	20
11	Slave02				Slave03				289	21
12	Slave04				Slave05				290	22
13	Slave06				Slave07				291	23
14	Slave08				Slave09				292	24
15	Slave10				Slave11				293	25
16	Slave12				Slave13				294	26
17	Slave14				Slave15				295	27
18	Slave16				Slave17				296	28
19	Slave18				Slave19				297	29
20	Slave20				Slave21				298	30
21	Slave22				Slave23				299	31
22	Slave24				Slave25				300	32
23	Slave26				Slave27				301	33
24	Slave28				Slave29				302	34
25	Slave30				Slave31				303	35

The slave address 9 is to access our LOGO! 12/24RC.
The AS-I outputs of the LOGO!-Slave Nr.9 are assigned in the process-image of the input table of the SIMATIC S7-300 CPU as follows:

Qa1 = I14.0, Qa2 = I14.1, Qa3 = I14.2 Qa4 = I14.3

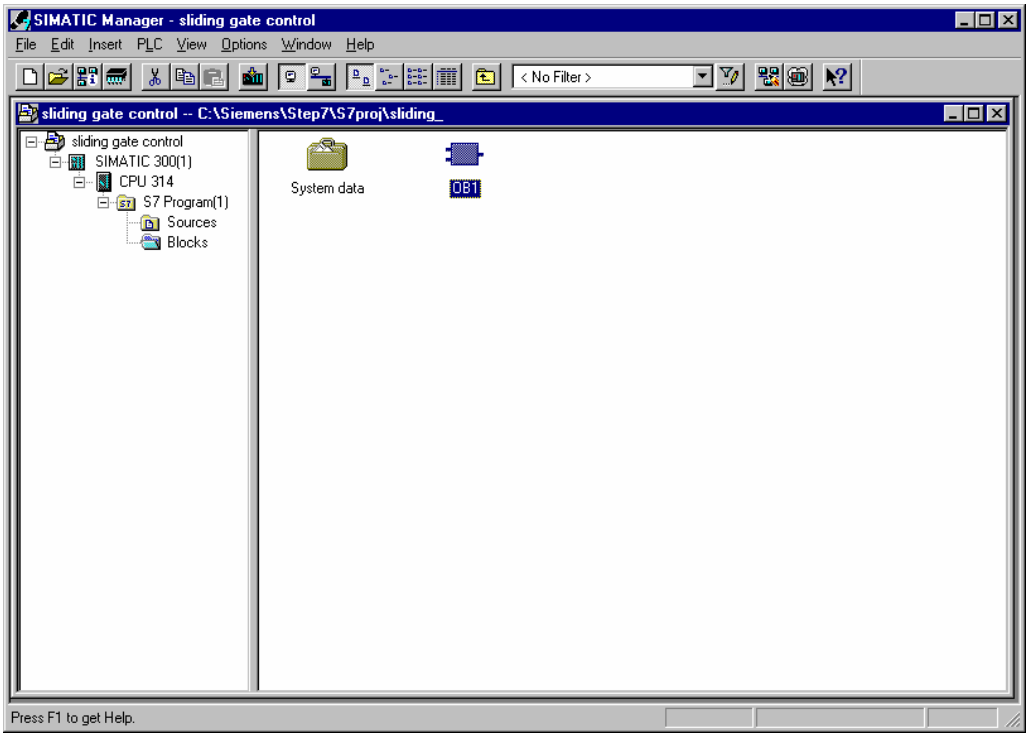
The AS-I outputs of the LOGO!-Slave Nr.9 are assigned in the process-image of the output table of the SIMATIC S7-300 CPU as followed:

Ia1 = Q24.0, Ia2 = Q24.1, Ia3 = Q24.2, Ia4 = Q24.3

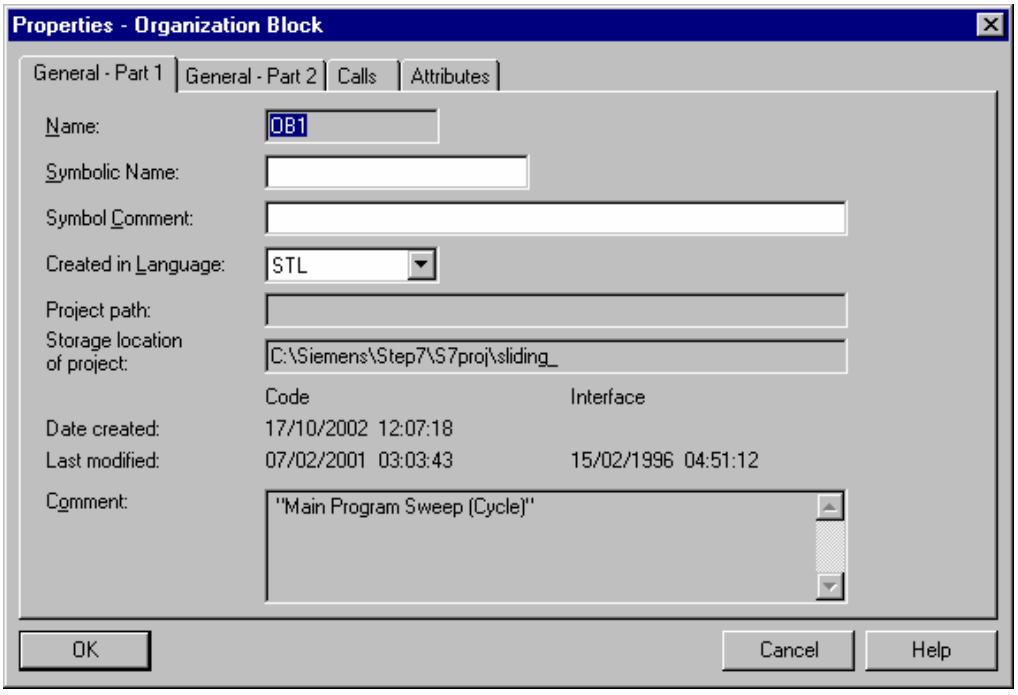
Note:
An output from the slave is an input for the master.
An input to the slave is an output from the master.

6.4 Control Program for the Driving of the Sliding Gate

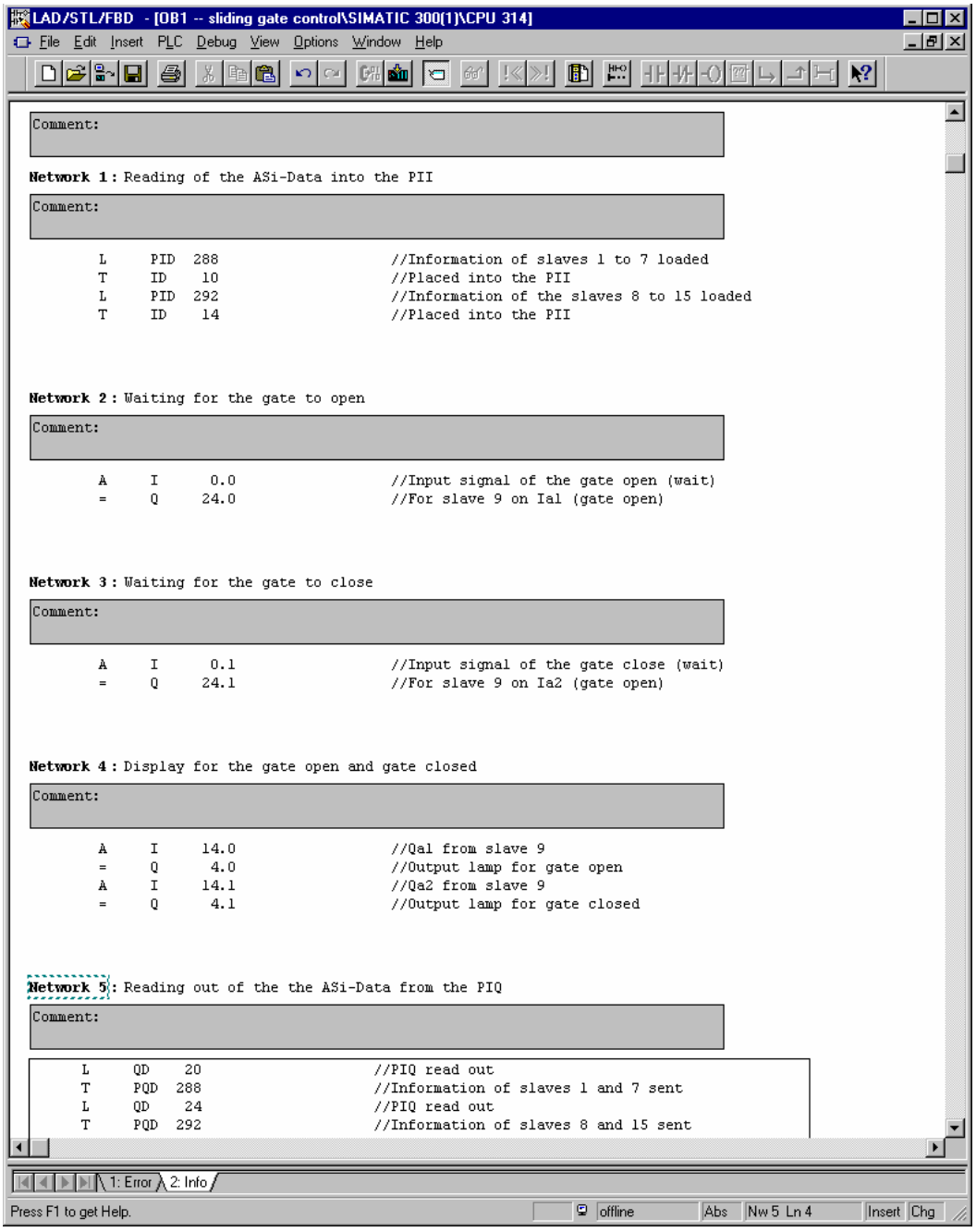
Extend the directory tree of the SIMATIC 300 station to the folder **Blocks**.
Double click on **OB1**.



Accept the properties with **OK**.



6.5 Entering Organization Block OB1



After the entering of the OB1 **save** and **download** it into the CPU.

Switch the CPU to RUN and test your program.