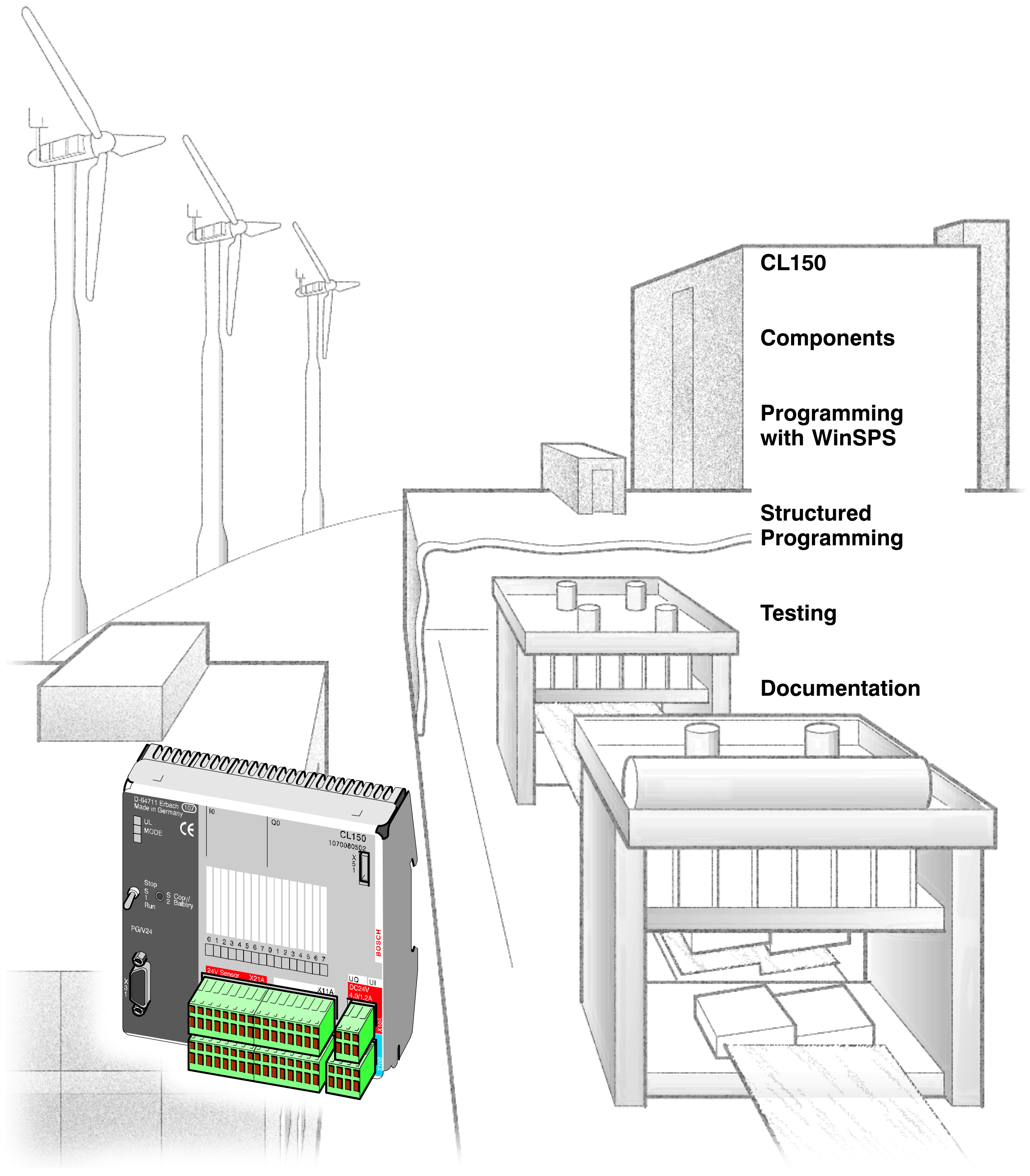
Drive and Control Technology

The world of the PLC – CL150



BOSCH
Automation Technology

The world of the PLC – CL150

1070 072 346-101 (01.04) GB

Discretionary charge 10.– EUR

Contents

CL150

This manual addresses skilled personnel with PLC know–how and technicians who are familiar with basic PLC programming.

The manual will guide you through your first PLC steps with CL150. This manual introduces the Bosch CL150 controller and provides the know–how required for handling and utilizing the CL150.

It shows you how to commission the CL 150 and start the programming tool WinSPS. You will be familiarized with the PLC instructions of the CL150 and use them in a WinSPS sample program.

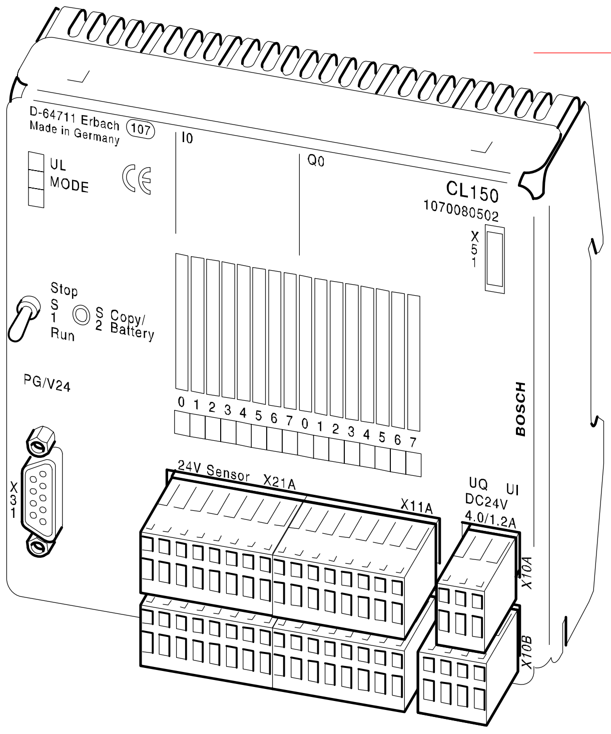
Testing and debugging the program after it has been created is often a rather time–consuming aspect in PLC programming. This manual will show you how to test and debug your program directly within the programming environment in combination with the CL150.

The symbolic and structured programming methods presented in this manual show you how to efficiently utilize the controller.

The CL150 is an economic and compact controller of the Bosch Programmable Logic Controllers (PLC) series. The CL150 is small, efficient and can be used for fast controlling tasks, for example in the automotive range, small conveying systems and in systems for the printing and paper industry.

This controller represents a cost efficient solution for even the smallest systems. With an enhancement of the decentralized I/O system components, or in a network as decentralized controller, it can handle simple tasks in assembly lines as the unit offers very good networking capabilities and a large instruction scale. The CL150 is available in a number of version for an optimal and efficient adaptation to the tasks at hand: dependent on the equipment, it comes with supplementary analog inputs and outputs or with optional operation within a Fieldbus network.

The CL150



WinSPS

Complete programming package for Windows95, 98 and NT

Project management, creation of programs, Online testing, documentation

Programming in IL, LD, FUD and SFC in accordance with DIN EN 61131

Multiuser functions for working in project teams

Multitasking for parallel program processing

Single-user and networking solutions

CL150

Basic version of the compact controller with 8 inputs and 8 outputs

Equipment variations with analog I/Os and connection to Fieldbus systems

Connection for the components of the I/O system B~IO

Extensions

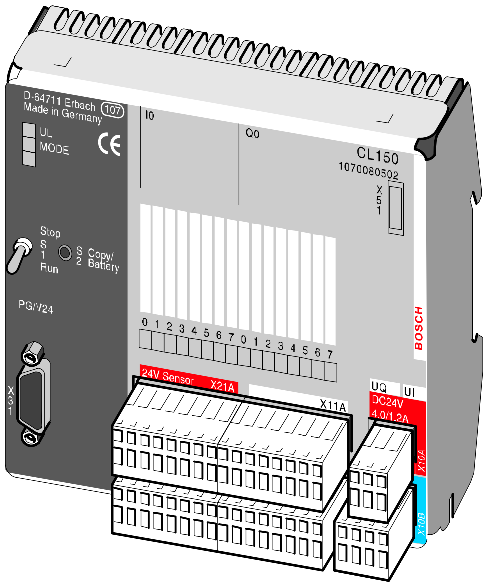
Customization of the given I/O structure with modules of the system B~IO M-

In-line operation with a maximum of 16 modules

Basic equipment of the CL150

The compact controller

- is available in several versions with analog and digital I/Os, an additional serial interface and a Fieldbus extension,
- is equipped with an expansion slot for I/O modules of the B~IO system and
- expandable with up to 16 modules.



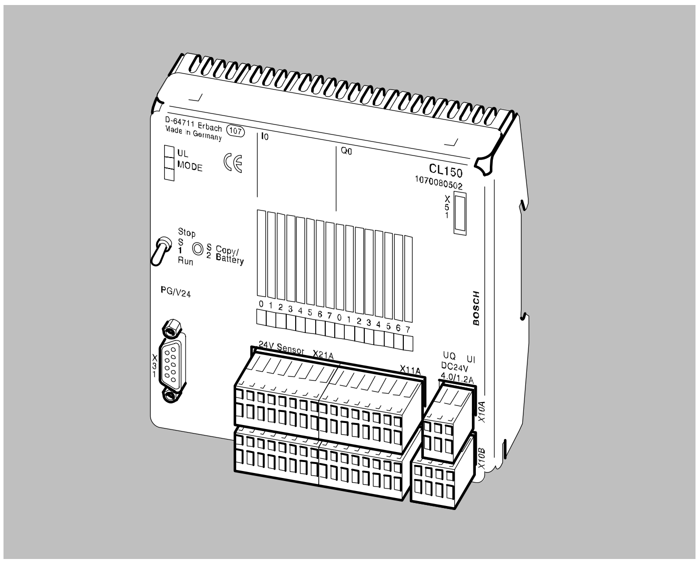
The controller has two internal memories - one of 64 KB RAM and one 64 KB Flash EPROM - which can be used individually or combined.

A battery buffers the controller's internal RAM against power failure and switching off, thus protecting the remanent areas for markers, timers, counters, data fields and data modules.

B~IO modular, available I/Os

Digital inputs	8 DI 8 inputs	16 DI 16 inputs	16 DI-3 16 inputs	
Digital outputs	8 DO 8 outputs 0,5 A	16 DO 16 outputs 0,5 A	8 DO/2A 8 outputs 2,0 A	8 DO R 8 relay-outputs 2,0 A
Combination module, digitale I/O		8 DI/DO 8 connections, bit utilization as I/O 0,5 A		
Analog I/O	4 AI_UI 4 inputs, 12 bit 0–5 V, 0–10 V, ±0,5 V, ±10 V	4 AI_UIT 4 inputs, 14 bit 0,1 V, ±1 V, ±10 V	4 AO_I 4 outputs, 16 bit —	4 AO_U 4 outputs, 12 bit 0–10 V, ±10 V

The basic devices



CL150

The CL150 has
- 8 digital inputs 24 VDC,
- 8 digital outputs 24 VDC/0.5 A,
- 2 fast 32-bit counters or
- 3 interrupt/alarm inputs,
- one programming interface V24, BUEP19E protocol.

CL151

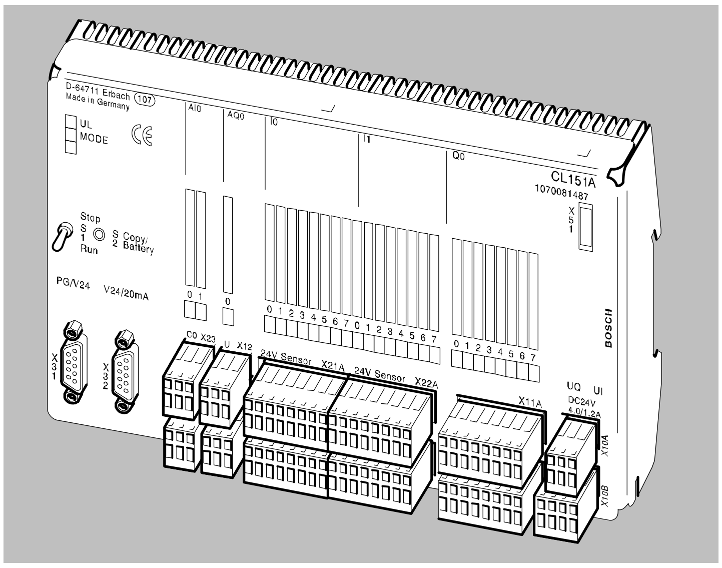
The CL151 has
- 16 digital inputs 24 VDC,
- 8 digital outputs 24 VDC/0.5 A,
- 2 fast 32-bit counters or
- 3 interrupt/alarm inputs,
- Real-time clock,
- one programming interface V24, BUEP19E protocol,
- a second serial V24 interface, 20 mA passive, protocol BUEP19E and BUEP03E.

Fieldbus interface

The CL150 and CL151 are available with the following Fieldbus interfaces:
- PROFIBUS-DP,
- CANOpen,
- InterBus-S and
- DeviceNet.

All devices with Fieldbus interface are equipped with the following features:
- 8 digital inputs 24 VDC,
- 8 digital outputs 24 VDC/0.5 A,
- 2 fast 32-bit counters or
- 3 interrupt or alarm inputs,
- Real-time clock,
- Programming interface V24, protocol BUEP19E
- CL151: second serial interface and BUEP03E protocol
- Fieldbus interface.

of CL150 and CL151



CL150A

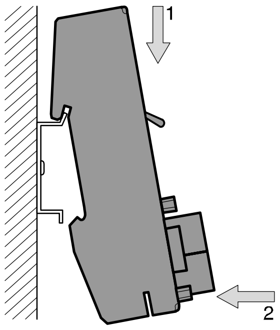
The CL150A has
- 16 digital inputs 24 VDC,
- 8 digital outputs
  24 VDC/0.5 A,
- 2 fast 32-bit counters or
- 3 interrupt/alarm inputs,
- 2 analog inputs 0-10 V,
  10 bits,
- 1 analog output ± 10 V, 0-20 mA, 12 bits,
- Real-time clock,
- one programming interface V24, BUEP19E protocol.

CL151A

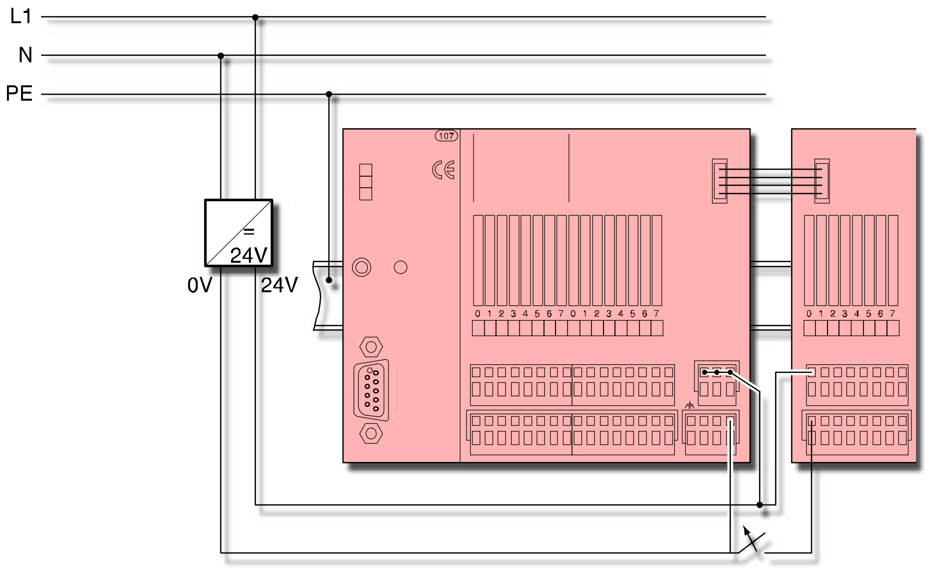
The CL151A has
- 16 digital inputs 24 VDC,
- 8 digital outputs
  24 VDC/0.5 A,
- 2 fast 32-bit counters or
- 3 interrupt/alarm inputs,
- 2 analog inputs 0-10 V,
  10 bits,
- 1 analog output ± 10 V,
  0-20 mA, 12 bits,
- Real-time clock,
- one programming interface V24, BUEP19E protocol,
- a second serial V24 interface, 20 mA passive, protocol BUEP19E and BUEP03E.

CL150 Controller Installation in the switchboard cabinet

- Mounting on rail or with screws
- Vertical, horizontal or lying mounting

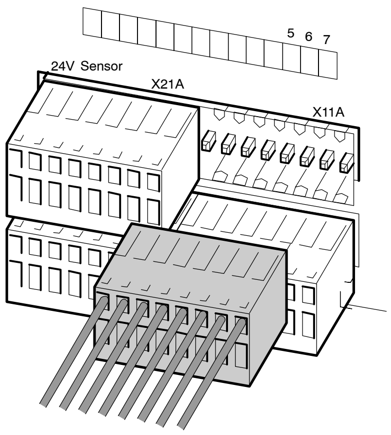


Connecting and wiring the PLC



- Fuse the voltage supply on primary side
- Connect the CL150 controller to the protective ground system PE by installing it on a grounded mounting rail
- Dimension the voltage supply for maximum current load
- Maximal permitted voltage fluctuation +20%, -15%
- Always use a power supply with safe separation compliant with DIN EN 60 742. Additional protective measures are not required if conforming.
- Wire the 24-V lines separately from lines carrying higher voltages.

The modules are supplied with removable plug terminals. Consequently there is no need to disconnect the wiring when replacing the controller or a module.



Expanding the CL 150 with I/O modules

Switch off the signal voltages and the external power supply of the controller before you connect modules.

When handling the controller or modules, make provisions for sufficient protection against electrostatic discharge that might destroy the controller or the module.

Automatic addressing

The CL150 operating system automatically assigns the start addresses to all connected I/O modules.

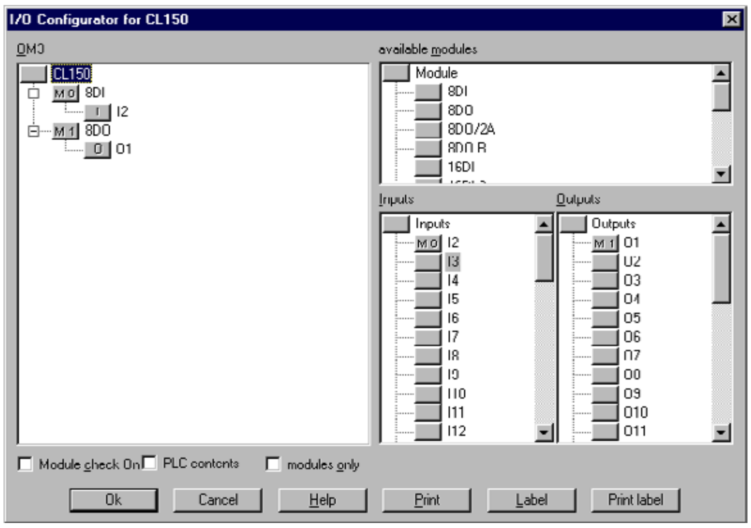
The module address is dependent on the sequence it is arranged in.

Start address for input modules is 2, for output modules it is 1.

The data length of the module is taken into consideration; word length modules are assigned to an even-numbered address.

Modules equipped with both inputs and outputs have the same input/output start addresses.

Manual addressing



The start address of all modules is customized in the WinSPS software.

In the WinSPS program, call the editor and select the menu “Edit -> I/O configuration (OB3)“.

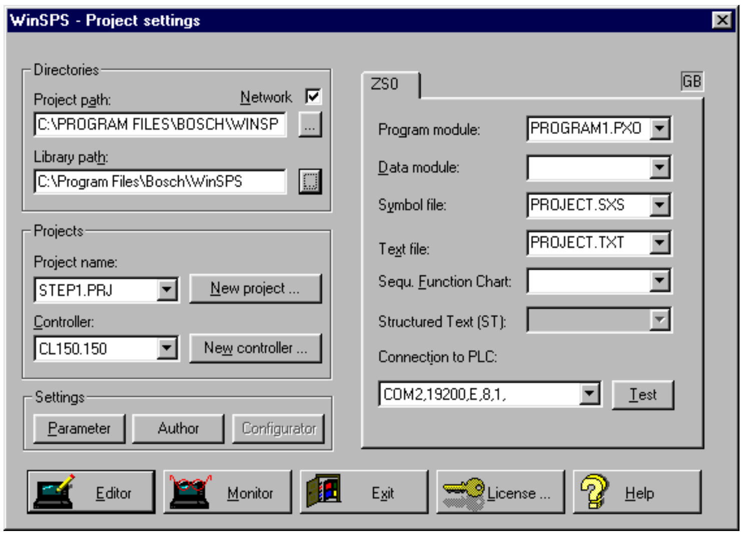
The PLC program evaluates the signal connections of and 8-input module, for example, with start address E2 via operand addresses I2.0 to I2.7.

Advantage of manual addressing:

The number of connected modules is fixed, that is, a disconnected module is recognized as error.

The program WinSPS

WinSPS at a glance



Project path: C:\PROGRAM FILES\BOSCH\WINSP
Library path: C:\Program Files\Bosch\WinSPS
Project name: STEP1.PRJ
Controller: CL150.150
Program module: PROGRAM1.PXO
Data module:
Symbol file: PROJECT.SXS
Text file: PROJECT.TXT
Sequ. Function Chart:
Structured Text (ST):
Connection to PLC: COM2,19200,E,8,1,

Default settings

Determining data related to the project and controls

- Project path and names,
- Controller types
- File names
- Interface configuration

Customizing the WinSPS startup parameters

Licensing information

Editor

Creating the PLC program with the

- Instruction List (IL)
- Function diagram (FBD)
- Sequential Function Chart (SFC)
- Contact plan (LD)

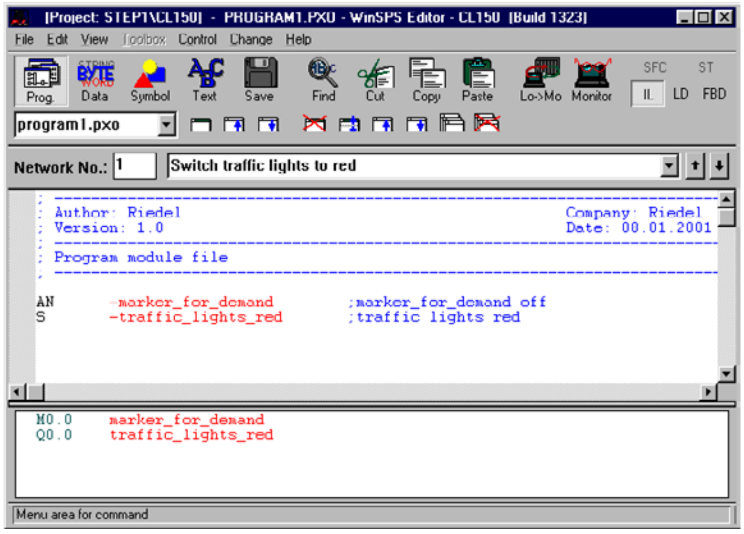
Interface test

Transferring the programs

Managing PLC memory utilization

Creating symbol lists and cross-reference lists

Printing out all PLC configuration data



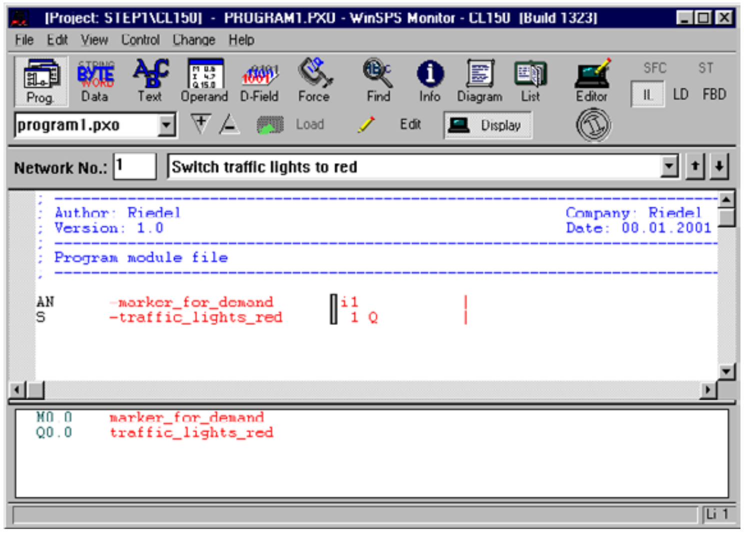
[Project: STEP1\CL150] - PROGRAM1.PXO - WinSPS Editor - CL150 [Build 1323]
File Edit View Toolbox Control Change Help
Prog
Data
Symbol
Text
Save
Find
Cut
Copy
Paste
Lo->Mo
Monitor
SFC
ST
IL
LD
FBD
Menu area for command

program1.pxo

Network No.: 1 Switch traffic lights to red

```
; Author: Riedel                                        Company: Riedel
; Version: 1.0                                          Date: 00.01.2001
; Program module file

AN      -marker_for_demand            ;marker_for_demand off
S       -traffic_lights_red           ;traffic lights red
```

```
M0.0    marker_for_demand
Q0.0    traffic_lights_red
```



[Project: STEP1\CL150] - PROGRAM1.PXO - WinSPS Monitor - CL150 [Build 1323]
File Edit View Control Change Help
Prog
Data
Text
Operand
D-Field
Force
Find
Info
Diagram
List
Editor
SFC
ST
IL
LD
FBD
Load
Edit
Display
Li 1

program1.pxo

Network No.: 1 Switch traffic lights to red

```
; Author: Riedel                                 Company: Riedel
; Version: 1.0                                   Date: 00.01.2001
; Program module file

AN      -marker_for_demand          i1
S       -traffic_lights_red         1 Q
```

```
M0.0    marker_for_demand
Q0.0    traffic_lights_red
```

Monitor

Testing the PLC program with the CL150

Monitoring

- the running PLC
- the program status
- data changes
- the I/O image

Retrieving module calls

Changing the switch state of inputs, outputs and markers in the PLC.

Installing WinSPS

Winsps.exe

Starting WinSPS

Valuable information about WinSPS

Ordering a WinSPS license per fax

Removing WinSPS from the hard disk drive, save for the licensing information

Uninstall WinSPS

Programming handb...CL150

PDF files of the controller programming manuals and information on SFC and PLC programming

Start the installation program ”SETUP.EXE” from the CD-ROM.

You can select ”SETUP.EXE” with the Explorer and start it per mouse click. Follow the instructions of the installation program.

The installation program unpacks and copies all files to the hard disk drive. The adjacent icons will be created in the ”Bosch” program group.

You do not require a license in order to be able to work with WinSPS and the CL150. However, if you want to use the program in other controllers also you must order a license key from Bosch.

Uninstalling WinSPS

Always remove a licensed WinSPS version with the program ”Uninstall WinSPS”. Initialize it with a click on the corresponding icon in the Bosch program group. Do not use the Explorer to remove WinSPS. This will delete all existing licensing information.
To recognize the license key, the new WinSPS software must be installed in the same directory as the previous version.

Programming license

You do not require a license in order to be able to work with WinSPS and the CL150. However, if you want to use the program in other controllers as well you must order a license key from Bosch. You can obtain licenses free of costs and for time-limited demo versions, as well as for full versions for single-user and networking applications.

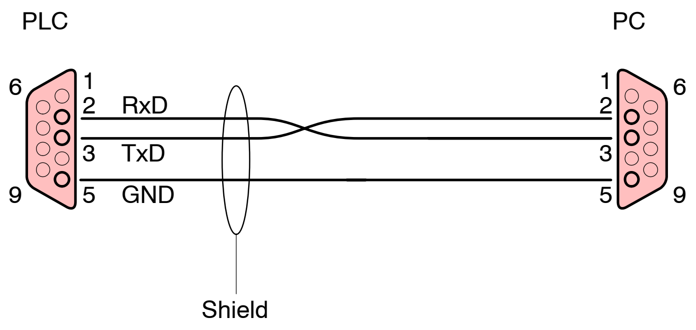
You can direct your license application to Robert Bosch GmbH by fax. A dialog box that is opened with the initial startup of WinSPS offers you information on the procedure.

All features of the WinSPS software will be fully enabled after you have entered the Bosch license key.

The license is valid only for the installed WinSPS program. If you want to use a second WinSPS installation, you must either order another license or transfer the existing license to the second version. This prevents a startup of the current WinSPS installation until you have returned the license to it.

You can use the ”License” button in the WinSPS program window ”WinSPS-project customization” for license transfer and changes.

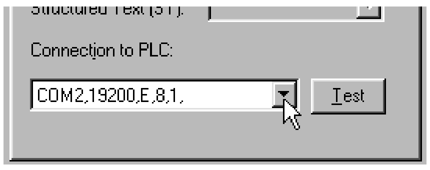
Connecting the PC to the CL150



You can use the Bosch serial cable K19 (order no. 1070 077 753) to transfer data between the CL150 and a PC. Connect the cable to a PC serial interface COM1 to COM4. The respective interface is selected in the WinSPS window ”Project settings” -> ”Connecting the PLC via:”.

Customizing communication parameters

The respective interface is selected in the WinSPS window ”Project settings” -> ”Connecting the PLC via:”. The transfer rate of the CL150 is set fixed to 19,200 baud.


Connection to PLC:

COM2,19200,E,8,1,

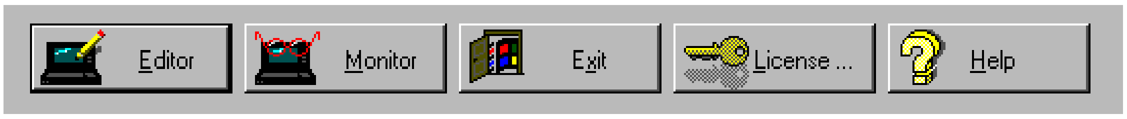
Interface test

The WinSPS software can test the function of the data communication and query the transfer rate settings. This interface test is started in the WinSPS window ”Editor”, under ”Controller -> Interface test”.

WinSPS tests the communication and returns the transfer parameters found.

If the data communication is disrupted,

- check whether the controller is switched on
- and whether all plugs are properly connected. The cable should be connected to the 9-pin X31 socket of the controller and to a PC serial interface.
- Check whether the PC detects the serial interface. You must restart Windows when you initially use the interface or if, for example, you swap the serial interface for the mouse and the controller connection. The Windows startup sequence recognizes active interfaces and assigns the connections accordingly.



Changing to the editor | **Changing to monitor mode** | **Closing the program** | **Enabling the program license** | **Calling Help**

Configuration

Starting WinSPS

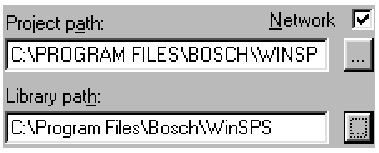
In your configuration settings you create or select the project path and name, as well as the controller name and type. Blocks and date are created for a project:

- a program module for the controller program,
- the Organization Module OM1 for starting the PLC program,
- a Data Module,
- a symbol file for the declaration of blocks and symbol operands.

Creating project directories

A directory path for the project

The path to the WinSPS program libraries

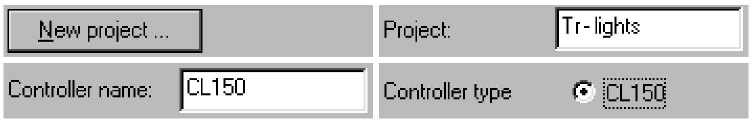


Project path: C:\PROGRAM FILES\BOSCH\WINSP ☑ Network

Library path: C:\Program Files\Bosch\WinSPS

Editing controller and project information

Project name, directory name for the controller and the controller type CL150



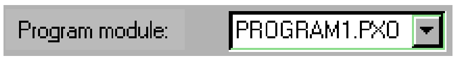
Project: Tr-lights

Controller name: CL150

Controller type ● CL150

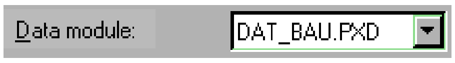
Insert the file names

The name of the module initially programmed in the editor



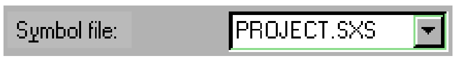
Program module: PROGRAM1.PXO

Name of a data module



Data module: DAT_BAU.PXD

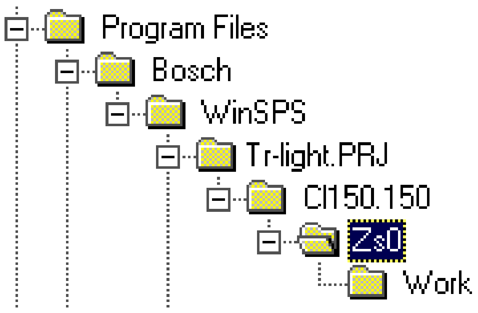
A name for the project symbol file



Symbol file: PROJECT.SXS

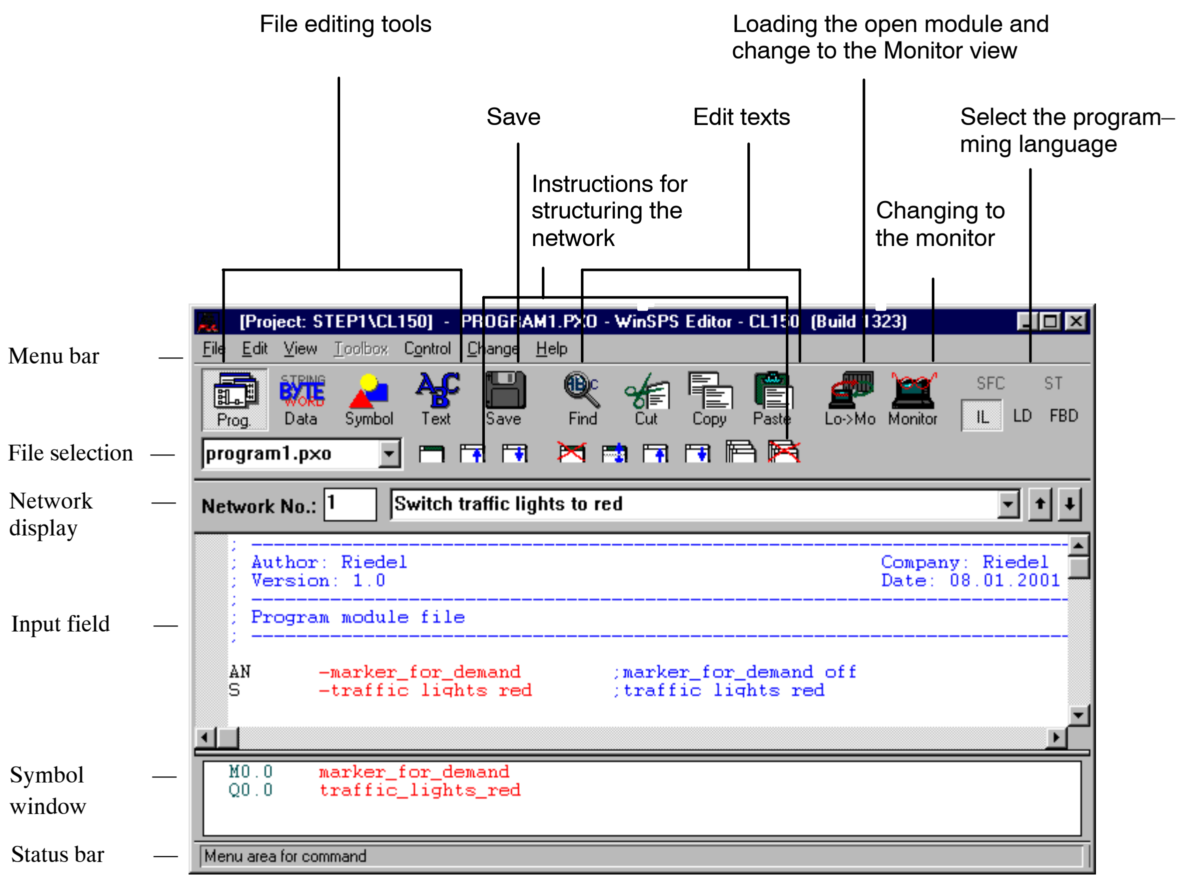
Call the editor
Files are created after confirmation

File structure in CL 150



WinSPS automatically creates all required directories after the query. You can choose names freely. It is important, however, that you subsequently enter the correct names in the symbol file.

The Editor


File editing tools
Loading the open module and change to the Monitor view
Save
Edit texts
Select the programming language
Instructions for structuring the network
Changing to the monitor
Menu bar
File selection
Network display
Input field
Symbol window
Status bar
[Project: STEP1\CL150] - PROGRAM1.PXO - WinSPS Editor - CL150 (Build 1323)
File Edit View Toolbox Control Change Help
Prog. Data Symbol Text Save Find Cut Copy Paste Lo->Mo Monitor
SFC ST IL LD FBD
program1.pxo
Network No.: 1
Switch traffic lights to red
Menu area for command

```
; ---------------------------------------------------------------------------
; Author: Riedel                                        Company: Riedel
; Version: 1.0                                          Date: 08.01.2001
; ---------------------------------------------------------------------------
; Program module file
; ---------------------------------------------------------------------------

AN      -marker_for_demand        ;marker_for_demand off
S       -traffic lights red       ;traffic lights red
```

```
M0.0    marker_for_demand
Q0.0    traffic_lights_red
```

WinSPS provides the following files for your program organization and documentation.

Symbol files for symbolic addressing and configuring the PLC program

Text files for comments and documentation

Operand and fixing files for the program test

PLC program modules for a library, for direct implementation into a PLC program.

Customizing the Editor view
You can switch the symbol bars and the network display on or off via menu item ”View”. Actuated menu items are indicated by WinSPS with a check mark.

Creating a program

Customizing the symbol file

In the symbol file you can declare the symbolic names for the operand.

Load the symbol file with the default file name into the editor.

The editor checks the syntax of all inputs and automatically formats error-free entries.

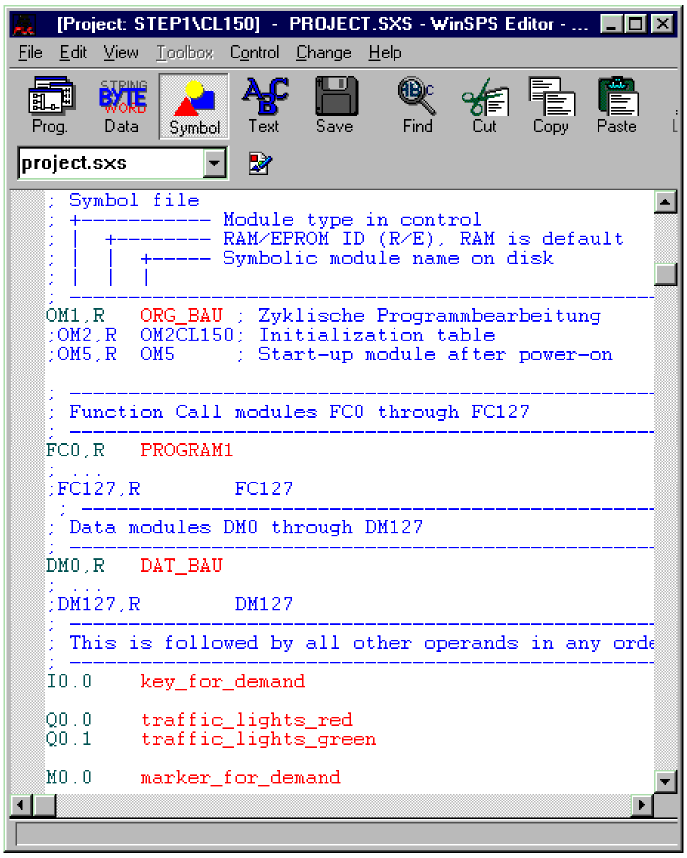
The blocks are assigned the file name under which they are stored. You must enter the following files:

- ”org_bau.pxo”, for the organization module OM1
- ”program1.pxo”, for the program module FC0.

WinSPS also requires the data module ”dat_bau”, because it has been entered in the default setting. You do not need to edit the module at this point.

For programming with symbolic operands you declare a symbolic name in the symbol file for each absolute operand such as I0.1 or M0.1.

Saving the symbol file.

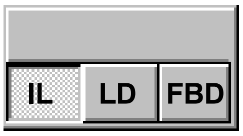


```
; Symbol file
; +----------- Module type in control
; |  +-------- RAM/EPROM ID (R/E), RAM is default
; |  |  +----- Symbolic module name on disk
; |  |  |
; ------------------------------------------------------------
OM1,R   ORG_BAU ; Zyklische Programmbearbeitung
;OM2,R  OM2CL150; Initialization table
;OM5,R  OM5     ; Start-up module after power-on

; ------------------------------------------------------------
; Function Call modules FC0 through FC127
; ------------------------------------------------------------
FC0,R   PROGRAM1
; ...
;FC127,R        FC127
 ; -----------------------------------------------------------
; Data modules DM0 through DM127
; ------------------------------------------------------------
DM0,R   DAT_BAU
; ...
;DM127,R        DM127
; ------------------------------------------------------------
; This is followed by all other operands in any orde
; ------------------------------------------------------------
I0.0    key_for_demand

Q0.0    traffic_lights_red
Q0.1    traffic_lights_green

M0.0    marker_for_demand
```

Programming the Organization Module OM1

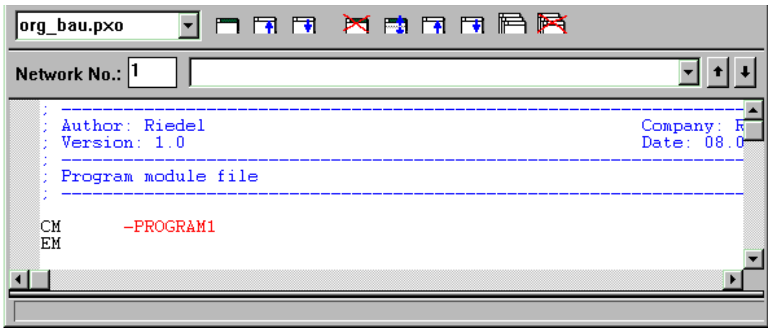
The Organization Module OM1 is the first program module to be programmed. Controlling of the program schedule is a major task of OM1. The PLC program is edited in another program module that is called by OM1.

The module call instruction is declared in the IL.
The instruction "CM" cannot be displayed in the FBD.



WinSPS relieves you of standard tasks

The file selection window displays the name of the open file.



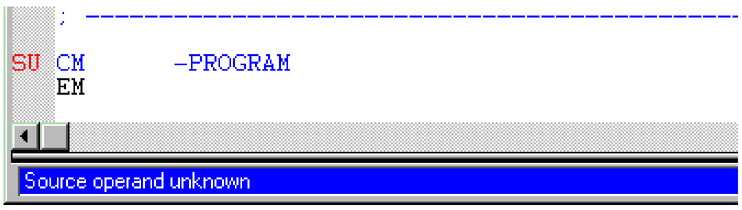
```
; ------------------------------------------------------------
; Author: Riedel                                 Company: R
; Version: 1.0                                   Date: 08.0
; ------------------------------------------------------------
; Program module file
; ------------------------------------------------------------

CM      -PROGRAM1
EM
```

New program files are automatically opened in WinSPS with a comment header that displays project information. Dependent on the file type, WinSPS supplements the file with additional text and program entries.

The program call declaration is entered in the editor

```
; Version: 1.0
  ;
--------------------------------------------------
  ; Program module file
  ;
--------------------------------------------------
  ;
> CM      FC0   ; Call of program module FC0
  EM
```

WinSPS immediately checks any program entry. While the cursor still points to the row, WinSPS automatically formats the program instruction if the entry is free of errors.

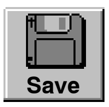


```
   ; ---------------------------------------------
SU CM        -PROGRAM
   EM
```

Source operand unknown

At the beginning of the command line WinSPS marks errors and alerts with an abbreviated error indication. After the cursor is positioned on the line, the status bar displays information on each selected line.

Alerts have an informative character and do not have to be remedied. However, they can indicate logical errors that might cause serious disruptions of the program cycle.



Save the Organization Module OM1 as "org_bau.pxo"

Creating the program module FC0

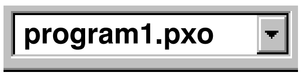
The second program module, namely FC0, is called up by the organization module OM1. It contains the PLC program.

Multiple program blocks are created for large projects to allow separate programming and testing of individual functions in the PLC program.

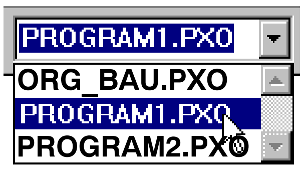
A new FC0 module must be created. You can specify ”program1.pxo” as file name.

Creating and opening the file

Enter the new file name in the file selection window and confirm with Enter.



Expand the file selection window and select the file.



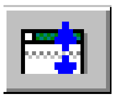
Splitting IL in networks

Split the program into networks. You can then edit it in IL and in FBD.

The network sequence corresponds with the workflow of the system. However, it can also be selected differently.

- Switching to the network view with ”View -> Networks”
- Splitting the new IL program into single networks

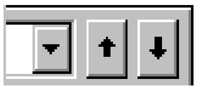
Separate the network



The instruction ”EM” is moved to a separate network because it cannot be displayed in FBD.

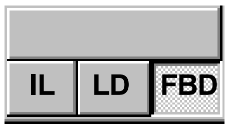
All program instructions as from the cursor position are moved to a new network. The FBD program is created in the first network.

Changing the network



You can return to the previous network with a click on the up arrow button. As an alternative, you can use the key shortcut ”Ctrl + PageUp”.

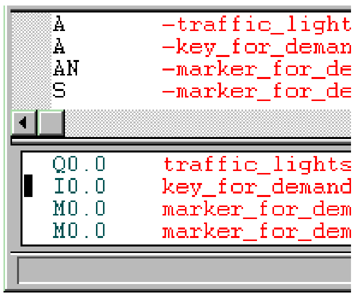
Changing the programming language to FBD



WinSPS switches to the FBD view only if the instruction sequence in the expanded network can be viewed in FBD. Otherwise, the status bar displays the message ”Cannot display row in FBD PI no.: 1”. ”PI” points to the first row of the program instruction that cannot be compiled.

The symbol window

The program editor lets you change the size or completely hide the symbol window by moving the separating line towards the input box via mouse button. The symbol window displays the symbolic names of all operands in the edit box. WinSPS continuously updates the symbolic names.



```
A         -traffic_light
A         -key_for_deman
AN        -marker_for_de
S         -marker_for_de

Q0.0      traffic_lights
I0.0      key_for_demand
M0.0      marker_for_dem
M0.0      marker_for_dem
```

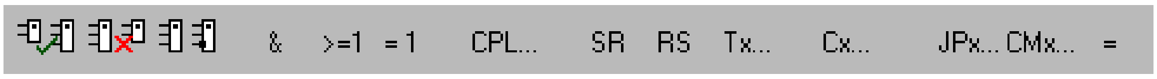
Networks in FBD

The cursor must be positioned on a new row you want to edit in the FBD input box. You can create and edit single blocks using the elements of the FBD toolbar.

To split the networks in FBD, call the network commands in the menu ”Edit -> Network instructions”. A network toolbar is only available in IL.

The new networks are inserted above or below the current network.

FBD toolbar



Inserting a logical AND

WinSPS creates networks using the two elements ”&” and ”=”.

Inserting RS/SR elements

Instead of SR, ”=” elements are used in some networks. Dependent on the logic result, this allows you to set or reset outputs, markers and similar.

Inserting additional inputs

In order to add an input pin, the cursor must be positioned on it.

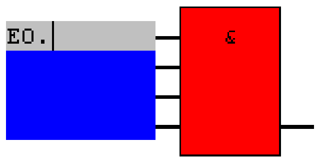
Negating inputs

To negate an input, the cursor must be positioned on the input pin.

Inserting SR elements.

The cursor must be positioned on the cross–link leading the ”=” element.

Naming I/Os



Declare the operand. The cursor must be positioned in the name box of the pin. Hit enter to move to the next field.

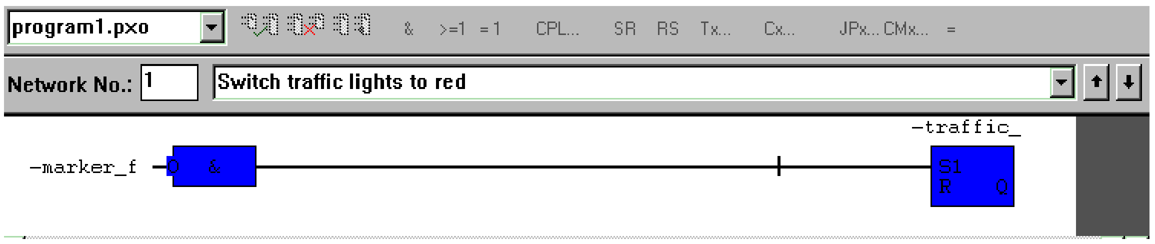
Naming the network

Insert a name in the network row, for example, ”Switch traffic lights to red”. Save the program.

To insert further RS or SR elements, position the cursor on the cross-link leading the ”=” or RS/SR element and insert the RS or SR element.

If you want to replace ”=” with an RS/SR element, insert the RS or SR element and then delete the ”=” element. Click on the ”=” element and delete it with the ”DEL” key.

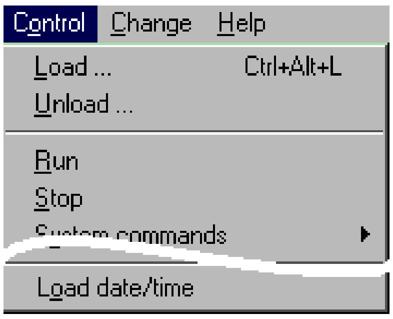
Network no. 1



Downloading the program to the controller

Do not test your program in a unit that is currently occupied by other tasks. The program might switch outputs and as a result create unexpected switching states.

Before you download the program to the PLC
- Switch on the controller
- Connect the data link to the PC and the controller.



Download the PLC program to the controller with all the modules that you have entered in the symbol file. Do not change the default settings in the "Load" window. Confirm your entries with OK.

Prior to downloading and for reasons of safety, WinSPS switches the controller to "Stop" mode and returns it to "Run" mode after the download.

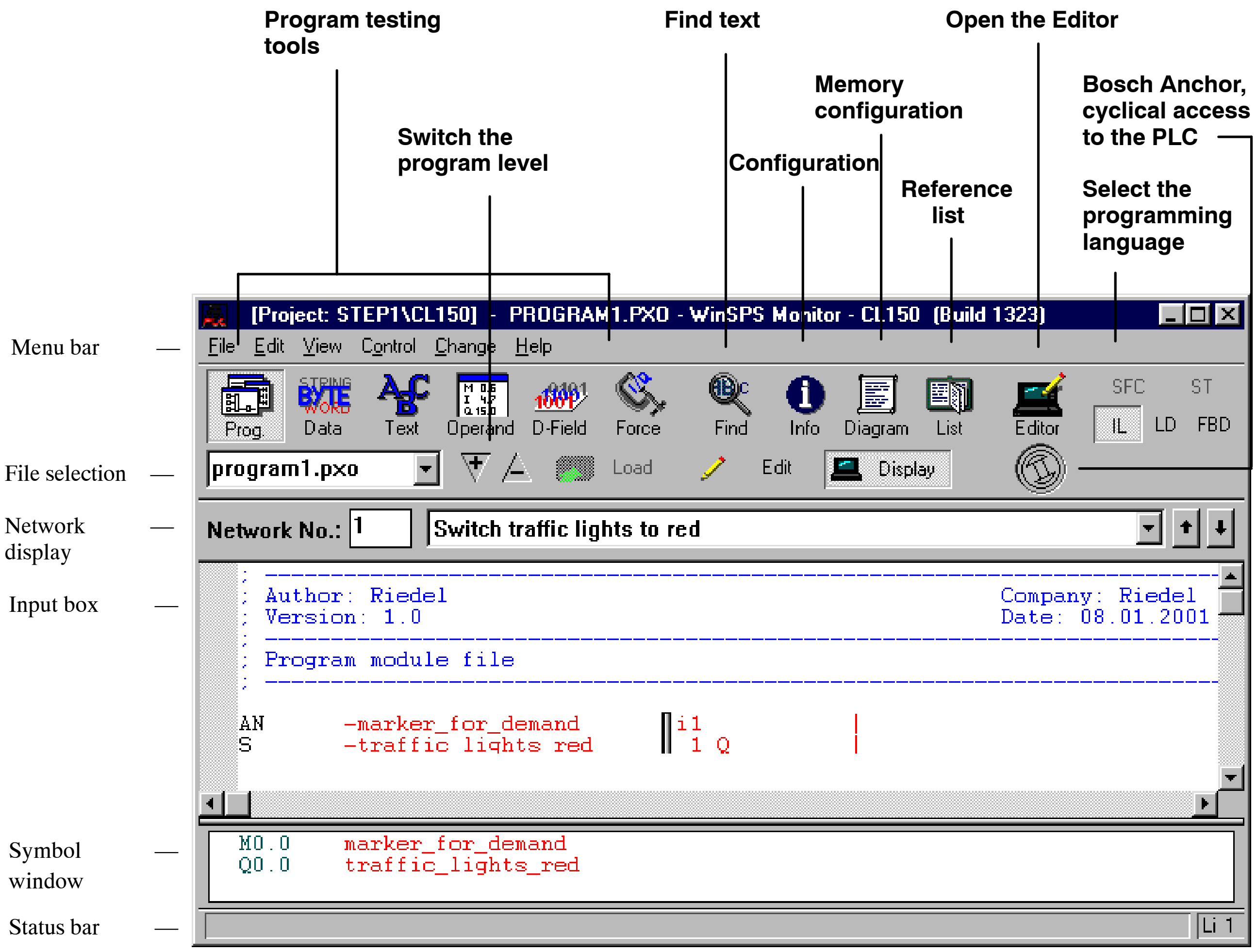
The red "Stop" LED on the front panel of the controller is switched off when the program is running on the CL150.

Open the Monitor to test the PLC program.

The transferred PLC program is tested in the Monitor.

WinSPS only starts the Monitor if
- all PLC program modules are error-free and downloaded to the controller and
- if communication is established to the switched on controller.

The Monitor



```
; ------------------------------------------------------------------------
; Author: Riedel                                      Company: Riedel
; Version: 1.0                                        Date: 08.01.2001
; ------------------------------------------------------------------------
; Program module file
; ------------------------------------------------------------------------
;
AN      -marker_for_demand      i1
S       -traffic lights red      1 Q
```

```
M0.0     marker_for_demand
Q0.0     traffic_lights_red
```

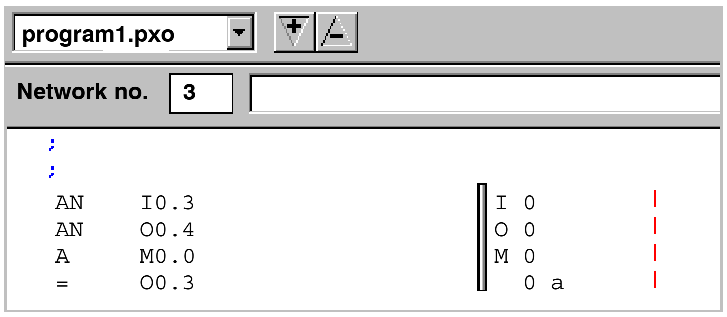
WinSPS offers some highperformance tools in the Monitor which allow you to

- test the program,
- monitor controller and program states,
- monitor the I/O image,
- retrieve module calls,
- and to switch I/O states.

Especially Operand and Fixing Files are used to this purpose.

Program module Monitor

View of the running PLC program

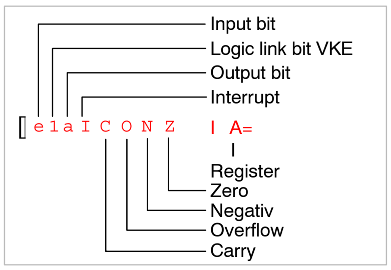


program1.pxo

Network no. 3

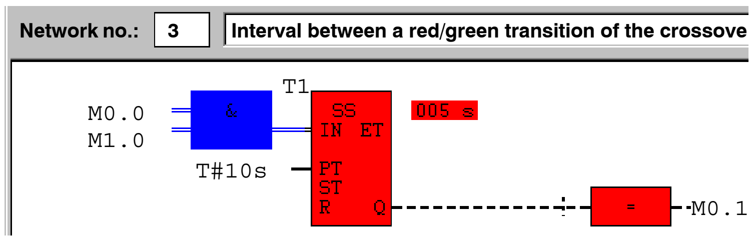
```
;
;
AN    I0.3                         I 0
AN    O0.4                         O 0
A     M0.0                         M 0
=     O0.3                           0 a
```

You can shift the separating line between the program and monitor display per mouse.



In the IL view, the current system status is displayed for each program instruction. You can customize the display of register values and the representation of variables via the menu "View -> Format ...".

FBD view of a network in the Monitor. The displayed network is cycled. The logical AND link is set. Marker M0.1 is set after a 10 s interval has expired.

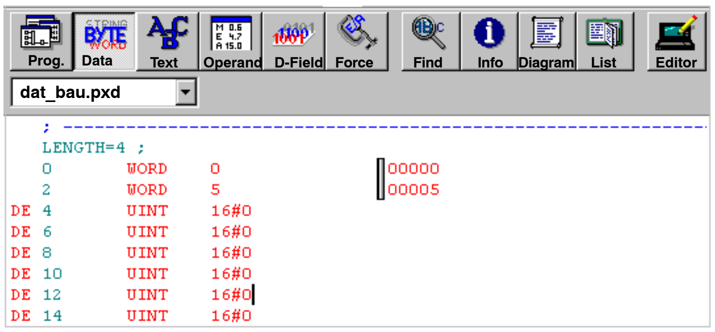


Network no.: 3 Interval between a red/green transition of the crossove

Data Module Monitor

Displays the Data Module and the current controller values.

Data Modules allow you to access and modify controller data, provided it is stored in the controller RAM area. Data Modules in the (E)EPROM area are read-only



dat_bau.pxd

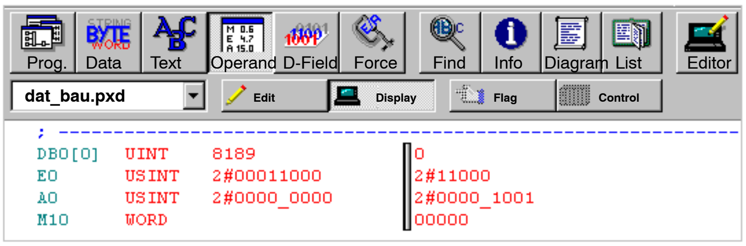
```
    ; -----------------------------------------------------------
    LENGTH=4 ;
    0          WORD     0                 00000
    2          WORD     5                 00005
DE  4          UINT     16#0
DE  6          UINT     16#0
DE  8          UINT     16#0
DE  10         UINT     16#0
DE  12         UINT     16#0
DE  14         UINT     16#0
```

The left-hand side displays the content of the Data Module, the right-hand side the current controller data.

Operand field Editor

Operand

You can display any operand on–screen and edit, mark or control it.



```
; ---------------------------------------------------------------
DB0[0]    UINT     8189                  0
E0        USINT    2#00011000            2#11000
A0        USINT    2#0000_0000           2#0000_1001
M10       WORD                           00000
```

On the same screen you can simultaneously display miscellaneous operands which cannot be displayed this way in the IL Monitor.

Edit — Editing the operand field and declaring new values.

Flag — Marking operands with changed initialization values.

When working with large PLC projects, the operand field editor offers a view of all important operand states.

Display — Display of operand values with cyclical update.

Modified operands must be marked before they are transferred to the controller.

You can edit these operand values and transfer them to the controller.

Entries marked ”Fixed” are the ones you have modified in ”Fixing Editor”, the Monitor tool.

Control — Marked operands are transferred to the controller with their new initialization values.

Data Field Editor

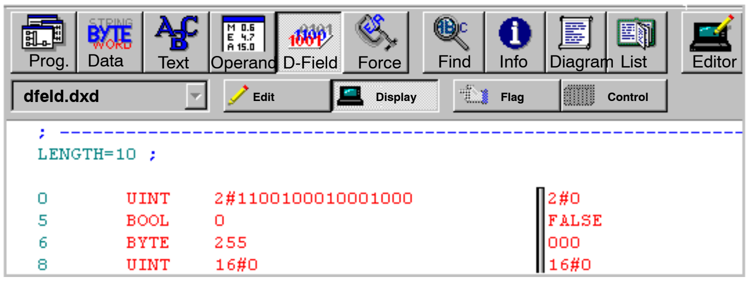
D–Feld

The Data Field Editor displays entries in the retentive data field of 8192 bytes length.

You can edit, mark and then transfer your entries to the controller.

The button functions correspond with those of the operand field editors.

Data field entries must be inserted in ascending address order.



```
; ---------------------------------------------------------------
LENGTH=10 ;

0         UINT     2#1100100010001000          2#0
5         BOOL     0                           FALSE
6         BYTE     255                         000
8         UINT     16#0                        16#0
```

The individual program modules and the complete PLC program are tested prior their execution. This is carried out by connecting the inputs to the PLC via a simulation field with diverse switches, or by simulating the connection using the Fixing Editor, the Monitor tool.

Fixing

You can fix inputs, outputs and markers.

Fixing overrides externally switched I/O states.

Fixed inputs are not displayed on the LED bar of the module, outputs, however, are shown.

Fixing is cancelled if you load a fixing without entries.

Data input in the fixing editor is:

```
Operand  Data typ  Fixing a value ; Comment
I0.3     BOOL      1              ; set I0.3 to "1"
I        USINT     2#xxxxxx1x     ; fix I0.1
```

Testing the controller program

The "Traffic light" example is tested with the fixing tool.

Force

WinSPS opens the fixing file "force.txd" when it starts the Fixing Editor.

Inputs I0.0 to I0.6 are modified in the PLC program; I0.7 is not used.

All bits which are not fixed can be masked with an "x" rather than "0" or "1".

```
I0   USINT 2#x0000000 ; reset I0.0 to I0.6
```

Two additional buttons are displayed above the working area:

Unload

Load the fixing configuration to the PLC.

Load

Unload all fixed I/O signals to the Fixing Editor.

The CL150 reactions can be monitored via the output LEDs.

The program monitor displays detailed information on all program states.

You can open a second WinSPS window to simultaneously fix inputs and monitor program changes.

Do not close the first WinSPS window and open WinSPS once again via Windows Start menu. Open the monitor directly in the configuration of the second WinSPS window. Open all files in read-only mode.

Both WinSPS programs run parallel and update data using the same data link. Therefore you can monitor program reactions parallel to the Fixing Monitor.

Accelerate the load rate after minor program modifications.

Lo–>Mo

Transfer the opened file from the editor to the PLC. Open the Monitor view.

Debugging programming errors

Editor

Correct the errors

Lo–>Mo

Documentation is an essential part of programming work. The programmer will only be able to systematically debug errors or continue to edit a program if the program is well documented.

WinSPS offers a series of functions and a Help for PLC program documentation, ranging from the creation of the new module to printing out the finished PLC program.

The programmer takes care of the major part of the documentation while he is programming. This includes:

- structuring the PLC program,
- commenting the program steps, networks and modules in IL,
- using symbolic programming.

WinSPS joins the comments in the PLC programs to form a complete documentation.

Commenting the FBD program

Program commenting is only possible in IL.

FBD only displays the comments above the network layer.

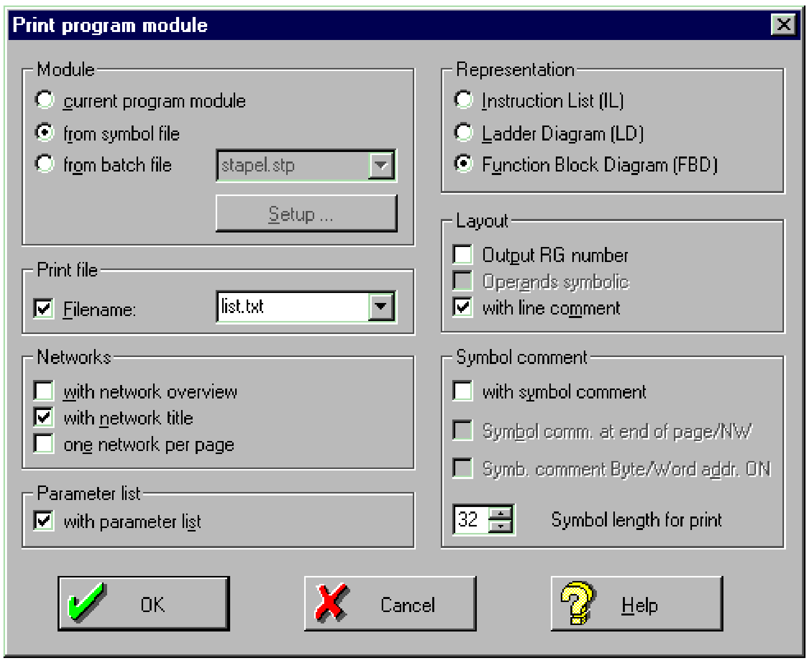
When a network is generated from IL to FBD, there must be no comment lines inserted between IL instructions.

Documenting the PLC program

The print function allows you to

- output program modules in IL, LD, SFC and FBD with and without comment texts,
- display programs with symbolic or absolute operands,
- output cross-reference lists, symbol files, network and module overviews.

Print jobs can be output to a file in order to make them available for further use in a text editor.


Print program module

Module
- current program module
- [✓ from symbol file]
- from batch file
- stapel.stp
- Setup ...

Print file
- [✓ Filename:] list.txt

Networks
- with network overview
- [✓ with network title]
- one network per page

Parameter list
- [✓ with parameter list]

Representation
- Instruction List (IL)
- Ladder Diagram (LD)
- [✓ Function Block Diagram (FBD)]

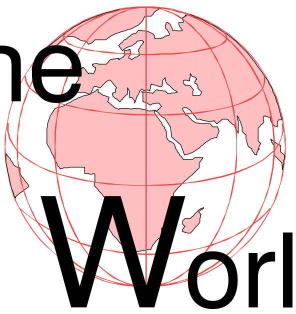
Layout
- Output RG number
- Operands symbolic
- [✓ with line comment]

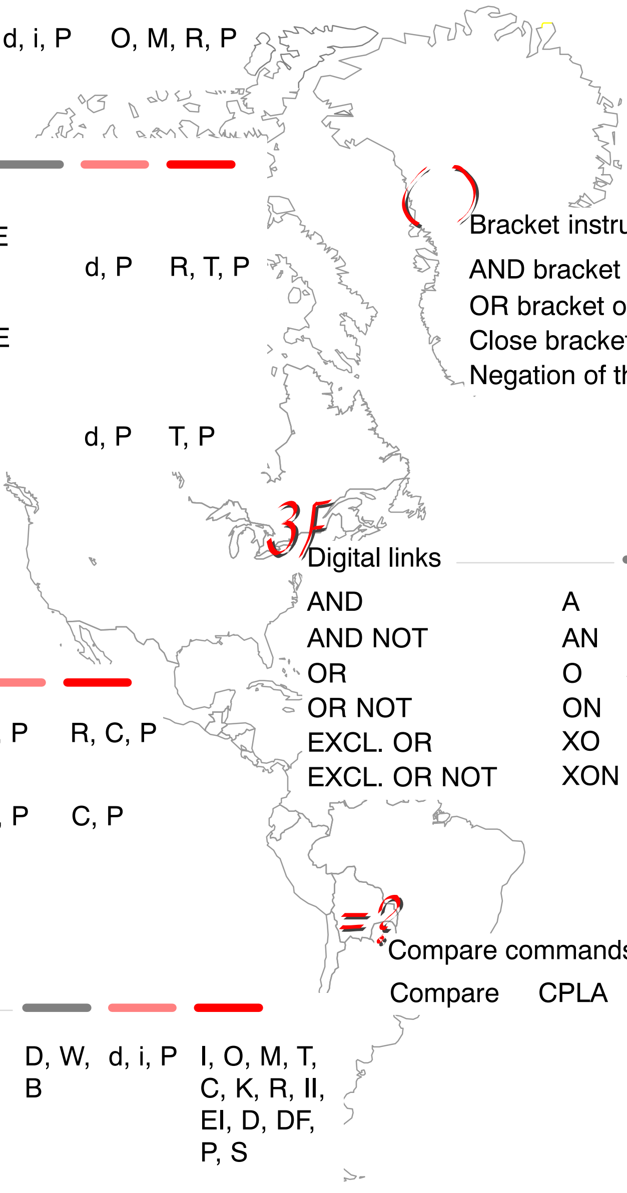
Symbol comment
- with symbol comment
- Symbol comm. at end of page/NW
- Symb. comment Byte/Word addr. ON
- 32 Symbol length for print

The stack processing function simplifies documentation tasks for large projects. The print objects are displayed in a window and released for printing in one pass.

You can customize the content of a standard print header and the print layout via diverse settings in the editor menu ”File print layout...”.

The World





Binary links

AND	A	Bit	d, i, P	I, O, M, T, C, R, P Status bits
AND NOT	AN			
OR	O			
OR NOT	ON			
Set	S	Bit	d, i, P	O, M, R, P
Reset	R			
Equal	=			

Time commands

Pulse	SP	d, P	R, T, P
extended Pulse	SPE		
On–delay	SR		
Rise–time On–delay	SRE		
Off–delay	SF		
Reset time	RT	d, P	T, P
Timer Stop	TH		

Bracket instructions

AND bracket open	(
OR bracket open	O(
Close bracket	)
Negation of the bracket content	)N

Digital links

AND	A	W, B	d, i, P	K, R
AND NOT	AN			
OR	O			
OR NOT	ON			
EXCL. OR	XO			
EXCL. OR NOT	XON			

Counter instructions

Set counter	SCY	d, P	R, C, P
Counter up	CU	d, P	C, P
Counter down	CD		
Reset counter	RCY		

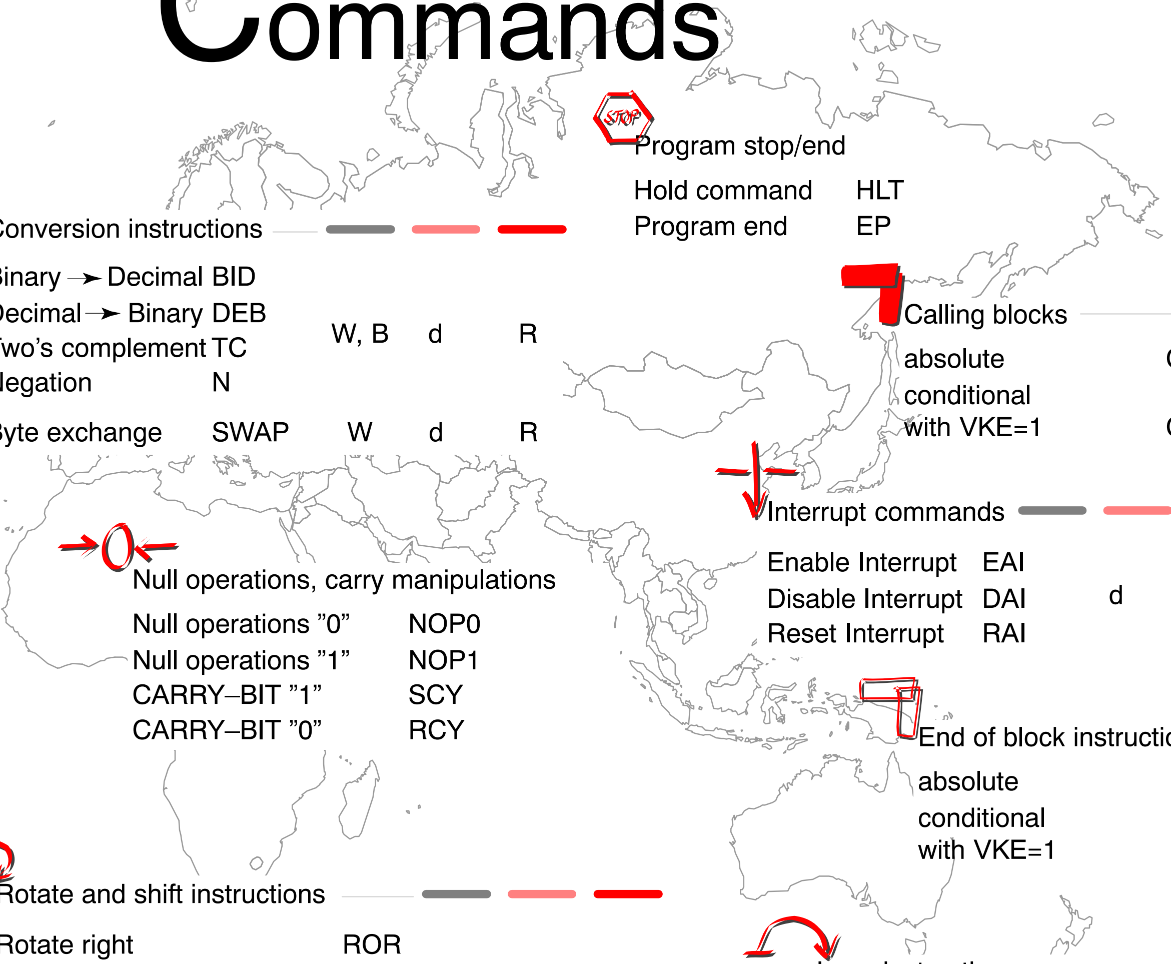
Compare commands

Compare	CPLA	W, B	d	K, R

Data transfer instructions

Load	L	D, W, B	d, i, P	I, O, M, T, C, K, R, II, EI, D, DF, P, S
Transfer	T	W, B	d, i, P	O, M, IO, EO

# of CL150 Commands



Arithmetic operations

Addition	ADD	W, B	d	K, R
Addition with carry	ADC			
Subtraction	SUB			
Subtraction with carry	SBB			
MultipliCation	MUL			
Division	DIV			
Increment	INC	W, B	d	R
Decrement	DEC			

Program stop/end

Hold command	HLT
Program end	EP

Conversion instructions

Binary → Decimal	BID	W, B	d	R
Decimal → Binary	DEB			
Two's complement	TC			
Negation	N			
Byte exchange	SWAP	W	d	R

Calling blocks

absolute	CM
conditional with VKE=1	CMC

Interrupt commands

Enable Interrupt	EAI	d	K
Disable Interrupt	DAI		
Reset Interrupt	RAI		

Null operations, carry manipulations

Null operations "0"	NOP0
Null operations "1"	NOP1
CARRY–BIT "1"	SCY
CARRY–BIT "0"	RCY

End of block instructions

absolute	EM
conditional with VKE=1	EMC

Rotate and shift instructions

Rotate right	ROR	W, B;	d	R
Rotate left	ROL			
Rotate right with CARRY	RCR			
Rotate left with CARRY	RCL			
Logical SHIFT right	SLR			
Logical SHIFT left	SLL			
Arithmetic SHIFT right	SAR			

Jump instructions

absolute	JP
conditional	JPx

Data types		Adressing		Operands					
Bit	Bit	d	direct	O	Output	EI	Extended Input	P	Parameter
B	Byte	i	indirect	EO	Extended Output	IO	Interface Output	R	Register
W	Word	P	Parameter	D	Data word	II	Interface Input	S	System area
D	Double word			DF	Data field	K	Constant	T	Timer
				I	Input	M	Marker	C	Counter

Logic & more

Binary instructions are the basic elements for logical links.
A binary command consists of the operation and the operand.

```
Operation   Operand   <; Comment>

A           I0.1      ; Logic AND with I0.1
S           O1.2      ; Set Output O1.2
=           M1.2      ; Pass RES in marker
```

Binary Operations

Logical AND

A	Query for signal “1“
AN	Query for signal “0“

Logical OR

O	Query for signal “1“
ON	Query for signal “0“

Assignment statements

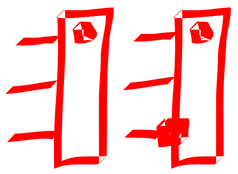
=	Assign logical link result
S	Output, marker... set signal to “1“
R	Output, marker... reset signal to “0“

Operands

```
ID, input here
|       Byte address
|       |   Bit ID, I/O
|       |   |
I       0 . 2
C       3
|       |
|       Counter ID
ID, counter here
```

Operands and valid address area

Inputs	I0.0	to I23.7
Outputs	O0.0	to O15.7
Markers	M0.0	to M191.7
Timers	T0	to T127
Counters	C0	to C63



Logical links

You can program logical links in IL, FBD and LD. For FBD and LD presentation you must work in networks.

; Comment **(* Comment *)**

Comments begin with a semicolon; or they are enclosed with (* and *).

Comments can only be edited in IL.

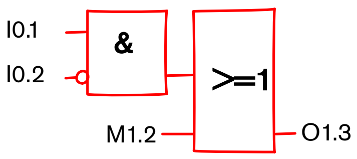
A network for binary links consists of a logical AND/OR sequence. It starts with an AND instruction or with an open bracket ”(” and ends with the assignment of the logical link result (RES).

In FBD and LD, WinSPS displays each network individually. In IL you can also work with a list view of all networks and completely without network technology.

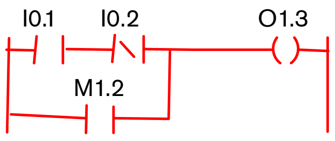
IL

```
A    I0.1    ;AND I0.1
AN   I0.2    ;AND NOT I0.2
O    M1.2    ;OR M1.2
=    O1.3    ;RSE in O1.3
```

FBD



LD



(Bracket functions)

Links are evaluated strictly in accordance with the rules of Boolean logic, that is, ”AND has priority over OR logic operations”. You can use the bracket functions and markers to change the evaluation sequence.

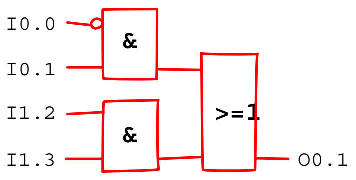
(	AND bracket open
O(	OR bracket open
)	Close bracket
)N	Negation of the bracket content

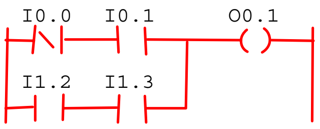
Bracket function can be nested in seven levels.

A network with bracket function can also be represented with markers.

AND has priority to OR

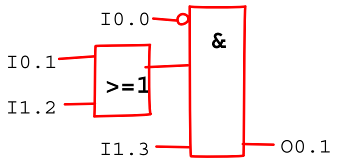
```
; Network 1
AN      I0.0
A       I0.1
O       I1.2
A       I1.3
=       O0.1
```

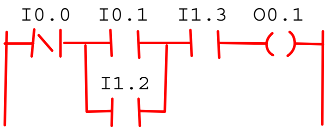




Brackets

```
; Network 2
AN      I0.0
(
A       I0.1
O       I1.2
)
A       I1.3
=       O0.1
```





Absolute or symbolic

WinSPS operates with absolute and symbolic operands. Absolute operands are called via their address, for example, I0.0.
Symbolic operands are addressed with their symbolic name. The name should describe the operand’s function to make the program more comprehensive and easier to read.

– Symbolic name
Declaration in the symbol file

Writing method ”- Symbol name”

Case sensitive

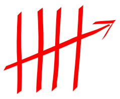
Not allowed are mutated vowels and special characters

The symbolic name of the module corresponds with its name in the module file.

Length of symbolic names
– for modules maximum 8 characters
– for other operands 32 characters

Operand	**Task**	**absolute**	**symbolic**
I0.1	Input I0.1 reports slide at the front	`AN    I0.1`	`AN  -Slide_is_Forward`
O0.3	Starts the conveyor belt	`=     O0.3`	`=   -Belt_On`
DM0	Module file ”crane_dat.pxd” with data for the project ”CRANE”	`CM    DB0`	`CM  -CRANE_DAT`
C3	Parts counter	`L  W  Z3,A`	`L  W  -Number_of_Parts,A`
D4	Maximum buffer capacity	`L  W  D4,B`	`L  W  -Max_Number_of_Parts,B`

Counters



The CL150 offers 64 counters and 128 timers. They can be started via program controls and without additional hardware.

You can program counters and timers in IL, FBD and LD.

Generating timers and counters in FBD

WinSPS automatically generates FBD modules for counters and timers. When you subsequently switch to IL, the result is a determined instruction sequence for the IL view.

If you toggle between IL and FBD you must not modify the instruction sequence determined in IL via FBD.

Programming counters

Counter operands are C0 to C63.

Count starts at the rising edge of RES.

The count starts at 0 or at the declared start value.

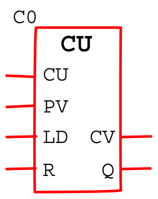
Valid count area and start values are 0 to 8191.

Counter commands

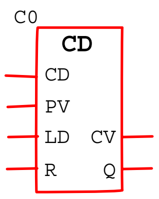
SCY	Declare the counter start value
CU	Increment
CD	Decrement
RCY	Reset counter to 0

The counter status is queried using the load command ”L”.

Up–counter



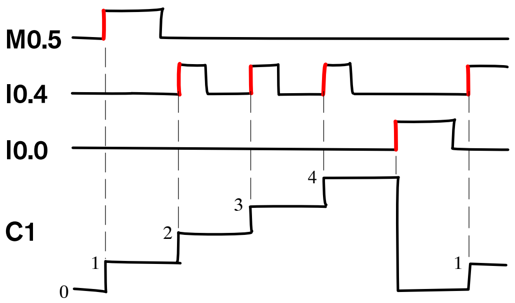
Down–counter

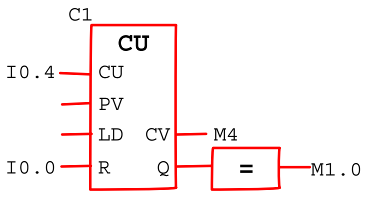


```
CU/CD:Count  condition
PV:  Start  value
LD:  Load  condition  for
     start  value
R:   Reset  condition
CV:  actual  counter  value
Q:   Counter  =  0: RES  =  0
     Counter  >  0: RES  =  1

CU/CD  and  Q  must  be
wired.
```

Up-counter CU

```
                 ; Count
A        I0.4    ; If I0.4 toggles 0 -> 1
CU       C1      ; increment C1 by 1
                 ; Set counter to 1
A        M0.5    ; If M0.5 is set
L    W   1,A     ; load counter status 1
SCY      A, C1   ; Set counter
L    W   C1, A   ; Load counter value
                 ; Reset counter
A        I0.0    ; If I0.0 toggles 0 -> 1
RCY      C1      ; reset counter
                 ; Save counter value
L    W   C1, A   ; Load counter value
T        A, M4   ; Save counter value
                 ; Query counter value 0
L    W   C1, A   ; Load counter value
A        C1      ; As long as counter is
                   not 0,
=        M1.0    ; Merker M1.0 setzen
```





Times

Programming times

Times start at the pulse edge of RES

- SP, SPE, SR and SRE at the positive edge
- SF at the negative edge

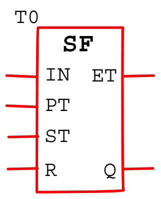
Transitions within the 1st cycle after a program start are ignored. The time period is declared in the time constant. Times are actualized in the I/O cycle. During a program cycle the time end is therefore not recognized until the next cycle.

Time commands

SP	Pulse time start
SPE	Start Pulse Extended
SR	Start Rising Edge Delay
SRE	Time start as rise-time On-Delay
SF	Start Falling Edge delay
RT	Reset time with RES=1
TH	Time Hold with RES = 1

The current time is queried using the load command ”L”.

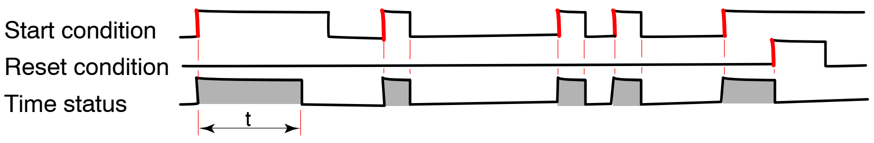
Time constant: T#10ms to T#10230s



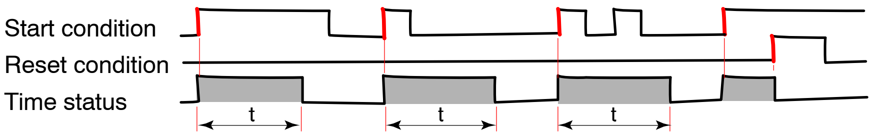
```
IN:  Start condition
PT:  Time constant
ST:  Hold condition for
      time
R:   Reset condition
ET:  Extra time
Q:   Time status

IN, PT and Q must be
wired.
```

Time diagrams

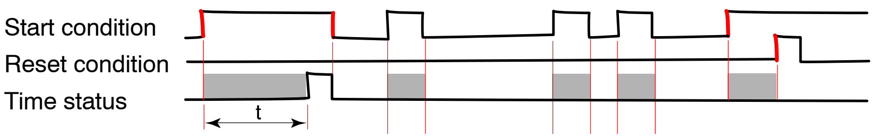
SP Start Pulse Time



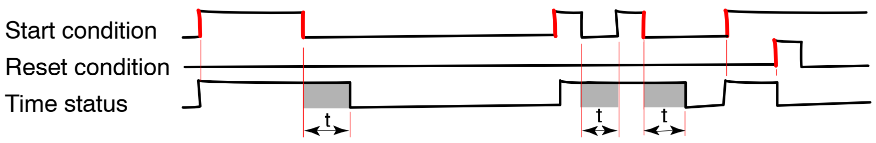
SPE Start Pulse Extended



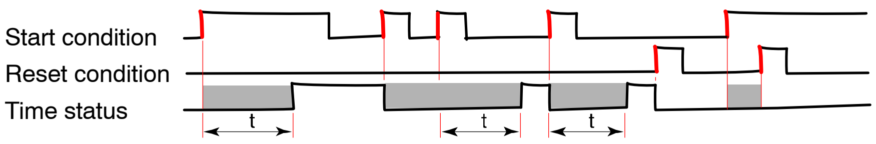
SR Start On-Delay

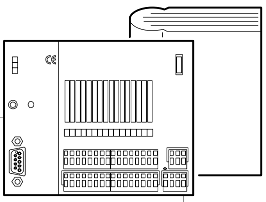


SF Time start as Off-delay



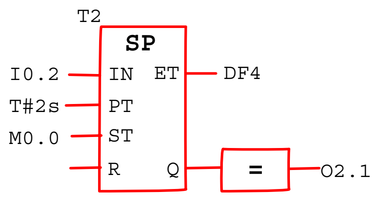
SRE Time start as Rise-time On-delay

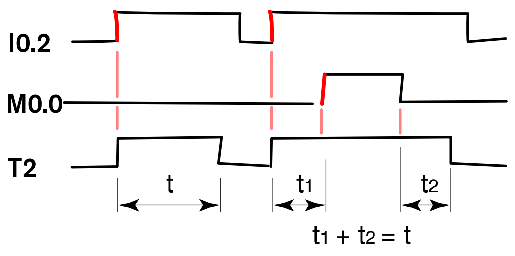




Time start as pulse SP

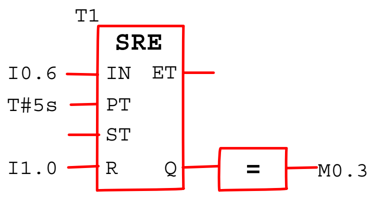
```
A       I0.2      ; I0.2 from 0->1
L    W  T#2s,A    ; Load time
SP      A,T2      ; Start time T2

A       M0.0      ; M0.0 from 0->1
TH      T2        ; Hold time counter

L    W  T2,A      ; Load time value
T    W  A,DF4     ; Transfer to DF4

L    W  T2, A     ; Load time value
A       T2        ; Time running?
=       O2.1      ; Set output
```

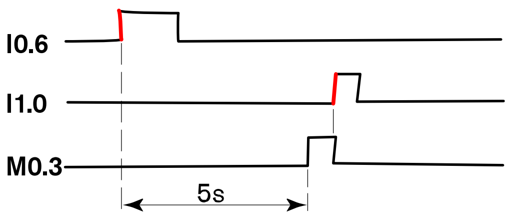




Start Time as On-delay SR

```
A       I0.6      ; I0.6 briefly from 0->1
L    W  T#5s,A    ; Delay 5 sec
SRE     A,T1      ; On delay

A       I1.0      ; I1.0 from 0 -> 1
RT      T1        ; Time off

L    W  T1,A      ; Load timer value
A       T1        ; Time running T1= 0
=       M0.3      ; Time expired:
                  ; T1= M0.3 =1
```





Calculations

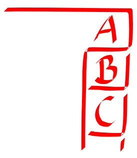
The CL150 offers digital linking, arithmetic function and compare instructions for calculations and comparison. Input data are passed as constants or register values. After command execution, the result appears in the register of the 2nd operand.

	Operation	1st Operand	2nd Operand
Digital linking	A, AN, O, ON, XO, XON	Constant registers	Register
Arithmetic operations	ADD, ADC, SUB, SBB, MUL, DIV	Constant registers	Register
Compare function	CPLA	Constant registers	Register

```
AN      B    A,B
ADD     W    14,A
        |    |  2nd operand, register
        |    1st operand, constant or register
        Byte- length or Word length
```

The status bits display supplementary information regarding the result, for example, the prefix operator or calculation errors.

Z	Zero, the result is zero
N	Negation, negative result
C	Carry, carry bit
O	Overflow, range overflow, division by 0 or result value exceeds 16 bits.

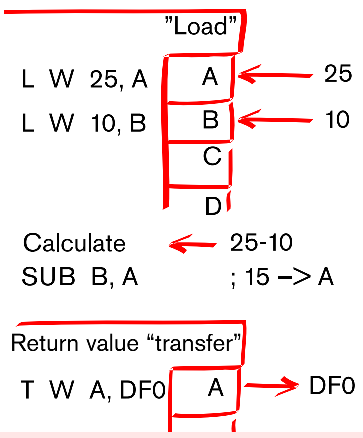


Registers are used as intermediate memory for data exchange. The CL150 operates with the four 16-bit registers A, B, C and D.

The addressing instructions ”L” and ”T” are used to exchange the values, for example, of a counter or data field, between the registers and an operand of the CL150.

The load operation ”L” loads the register with a constant or with the value of an address area.
The transfer operation ”T” returns the contents of the register to an address area.

Calculate 25 - 10 =15



Calculating operations cannot be displayed graphically. They can only be programmed in IL.

IL instructions in FBD

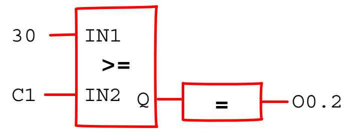
When programming in FBD, a separate network must be created for every instruction sequence that cannot be displayed in FBD. WinSPS automatically toggles the display mode from FBD to IL if a network cannot be displayed in FBD.

Compare =?

The CPLA instruction can be used for logical and arithmetic comparison. It can be programmed in IL or FBD. The compare result is evaluated via status bits.

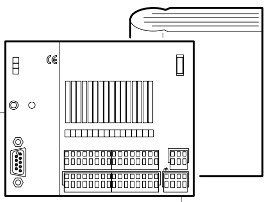
Counter setpoint 30 reached?

```
L     W   C1,A   ; Counter value
L     W   30,B   ; Setpoint 30
CPLA  W   B,A    ; Compare
AN        CY     ; A >= B?
=         O0.2   ; RES -> O0.2
```



Evaluation of the logical comparison

CPLA W B, A	Query	
A = B	A	Z
A ≠ B	AN	Z
A < B	A	CY
A ≤ B	A	Z
	O	CY
A > B	AN	CY
	AN	Z
A ≥ B	AN	CY



Calculation with Byte or Word operands

Byte operands can occupy any operand address, I3, M91, O5. Word operands must occupy even-numbered address areas only, O14, M0, I6.

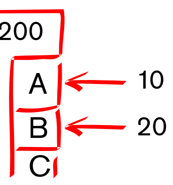
With add and subtract operations, the input and result operands occupy the same register length.

The results of multiplication instructions and the division operation for the first operand require double register length.

Multiplication with Byte and Word operands

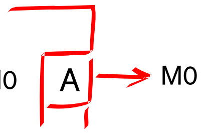
10 * 20 = 200

```
L  B   10, A
L  B   20, B
```



```
MUL  B  B,A    ; 200 -> A
```

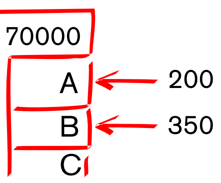
```
T  B   A, M0
```



M0 = 200

200 * 350 = 70000

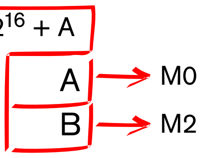
```
L  W  200, A
L  W  350, B
```



```
MUL  W  B,A     ; 70000 -> BA
```

BA = B x 2^{16} + A

```
T  W  A, M0
T  W  B, M2
```

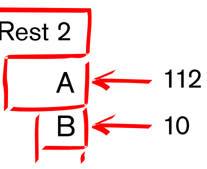


M0 = 4464, M2=1

Division with Byte and Word operands

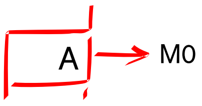
112/10 = 11 Rest 2

```
L  W  112, A
L  B     10, B
```



```
DIV  B  B,A  ; 2 -> Al, 11 -> Ar
```

```
T  W  A, M0
```

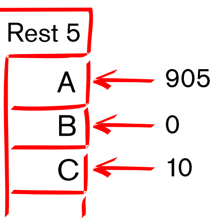


M0 = 2 x 2^8 + 11 = 523

Al: Register A left byte = remainder
Ar: Register A right byte = result

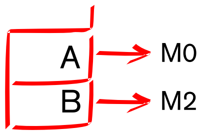
905/10 = 90 Rest 5

```
L  W  905, A
L  W      0, B
L  W    10, C
```



```
DIV  W  C,A   ; 90 -> A, 5 -> B
```

```
T  W  A, M0
T  W  B, M2
```



M0 = 90, M2 = 5

Indirect addressing of operands

With indirect addressing the operand name and address are declared separately. The operand address is first loaded via register.

Addressing M2 directly

```
L    W    M2, B
```

Addressing M2 indirectly

```
L    W    2,A
L    W    M[A],B
```

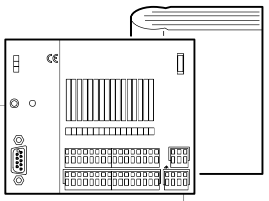
Bit operands such as I6.2 are addressed via Byte address and Bit number.

```
L   W   50,A    ; 50:8=6,  Rest 2
A       E[A]    ; Input I6.2
```

Indirect addressing is used, for example, to copy inputs to a marker area using only few commands. The Words of inputs I10 to I20 are copied to the marker words M50 to M60.

The ”JPx continue” command means: jump to program address, ”continue” if the previous value is not negative. The operations INC and DEC add/subtract the value of the 2nd operand to/from the 1st operand.

```
L    W    5,A      ; Loop counter 5
L    W    10,B     ; Start address for inputs: 10
L    W    50,C     ; Start address for markers: 50
 continue:         ; Destination address for loop
L    W    E[B],D   ; Read input value
T    W    D,M[C]   ; Write to marker area
INC  W    C,2      ; Next marker address
INC  W    B,2      ; Next input address
DEC  W    A,1      ; Loop counter -1
JPN    continue    ; Loop counter >=0? => continue
```



Constants, variables and address areas

Constants

Meaning	Representation	Area
Unsigned integer	Binary	2#0000000000000000 to 2#1111111111111111
	Decimal	0 to 65535
	Hexadecimal	16#0000 to 16#FFFF
Signed integer	Decimal	-32768 to +32767
Text	ASCII	'ABC'
Time value		T#10ms to T#10230s alternative input: T#0.r to T#1023.r, r: 0=1ms, 1=100ms, 2=1s, 3=10s

Variable types

Meaning	Prefix	Data typs	Data length	Examples
Bit	X	BOOL	1 bit	TRUE, 0
Byte	B	BYTE, SINT, USINT	8 bit	232, –126, 2#00001111
Word	W	WORD, INT, USINT	16 bit	12670, –2504, 8#376376
Douple word	D	DWORD, DINT, UDINT	32 bit	78900, –123000, 16#FEF4FEF5
Counters		CVALUE	16 bit	3
Timers		TVALUE	16 bit	T#10s
Strings		STRING(x), OSTRING(x), VSTRING(x)	variable with length x	'CHARACTER', 09, 3F, 0D,21, 3F, 7E

Address area of the load and transfer instruction

Load instruction "L"
L W 1st Operand, register

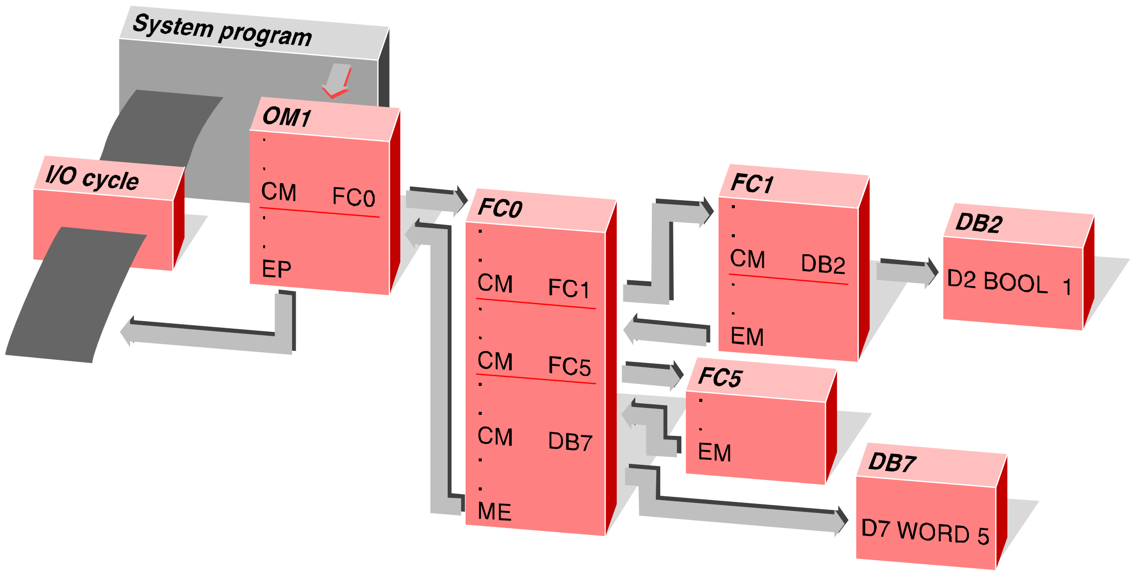
Transfer instruction "T"
T W Register, 2nd operand

1. Operand	Address area	
Input, Interface input	I0 II0	to I47 to II1
Output	O0	to O31
Marker	M0	to M151
Timers, Time value	T0 10 ms	to T127 to10230 s
Counters, Counter value	C0 0	to C63 to 8191
Constant	0	to 65535
Register	A, B, C, D	
Parameter	P0	to P31
System	S0	to S255
Data Field	DF0	to DF8191
Bytes in Data Module	D0	to D511

2. Operand	Address area	
Output, Interface Outp.	O0 IO0.0	to O31 to IO0.7
Marker	M0	to M151
System area	S0	to S255
Parameter	P0	to P31
Data Field	DF0	to DF8191
Bytes in Data Module	D0	to D511

Modules and files of the PLC program

The controlling and structuring of PLC programs, as well as a well arranged design of large controlling projects is assisted by the use of organization/program/ data modules.



Organization Modules

OM1 to OM19 represent the interface between the program and the controller.
Organization Modules

- start the PLC program cyclically,
- initialize the system area of the CL150,
- offer a variable program start;
- process error and interrupt handlers and they
- are closed with a ”EP” instruction.

Program modules

FC0 to FC127 contain the major parts of the PLC program. FC represents a Function Call.
Program modules

- can call data modules and other FCs,
- contain mainly interrelated functions of program parts,
- can be called with I/O parameters,
- are closed with a ”EP” instruction.

Data modules

DM0 to DM127 store the fixed and variable values of as well as the text information in the PLC program. Two DMs can be active concurrently in a program block.
Data modules

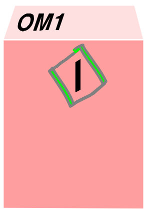
- are called by FCs or OMs,
- contain between 1 and 512 bytes of data per DM,
- must be activated prior to using them,
- are only active in the calling FC or OM,
- stay active in the FC or OM until other data modules are called.

Data field

The CL150 manages a data field with a size of 8 KB: DF0-DF8191; can be used as read/write buffer for any data. The data field can be battery buffered against power failure.

The CL150 Organization Modules

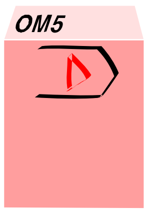
Cyclical block



The system cyclically calls the Organization Module OM1. The I/O image of the CL150 is updated prior to each call. OM1 is used mainly for program controlling. It starts the lower program module level.
OM1 must be implemented once in every PLC program.

All other Organization Modules are optional.

Startup Module

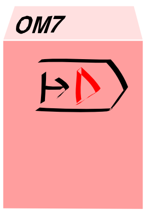


The OM5 program is processed after the CL150 is switched on and at the start of the PLC program.

Definition module

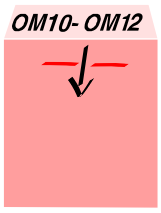
In OM2 you can edit system configurations of the CL150 such as, for example, the maximum cycle time, time and timer behavior or remanence limits. At the first start of the PLC program, OM2 writes the Initialization to the CL150 system memory.

Startup Module



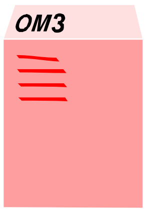
Contrary to OM5, the OM7 program is executed only on transitions from Stop to Run mode.

Interrupt Module



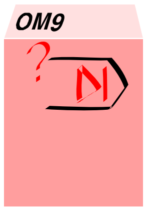
Programmed Interrupt Modules are called by the system program if the signal changes on specified interrupt inputs.

Configuration Table



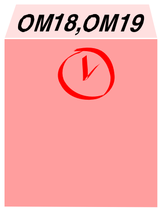
The configuration program WinSPS creates an OM3 when I/O modules of the B~IO system are manually configured.

Error Module



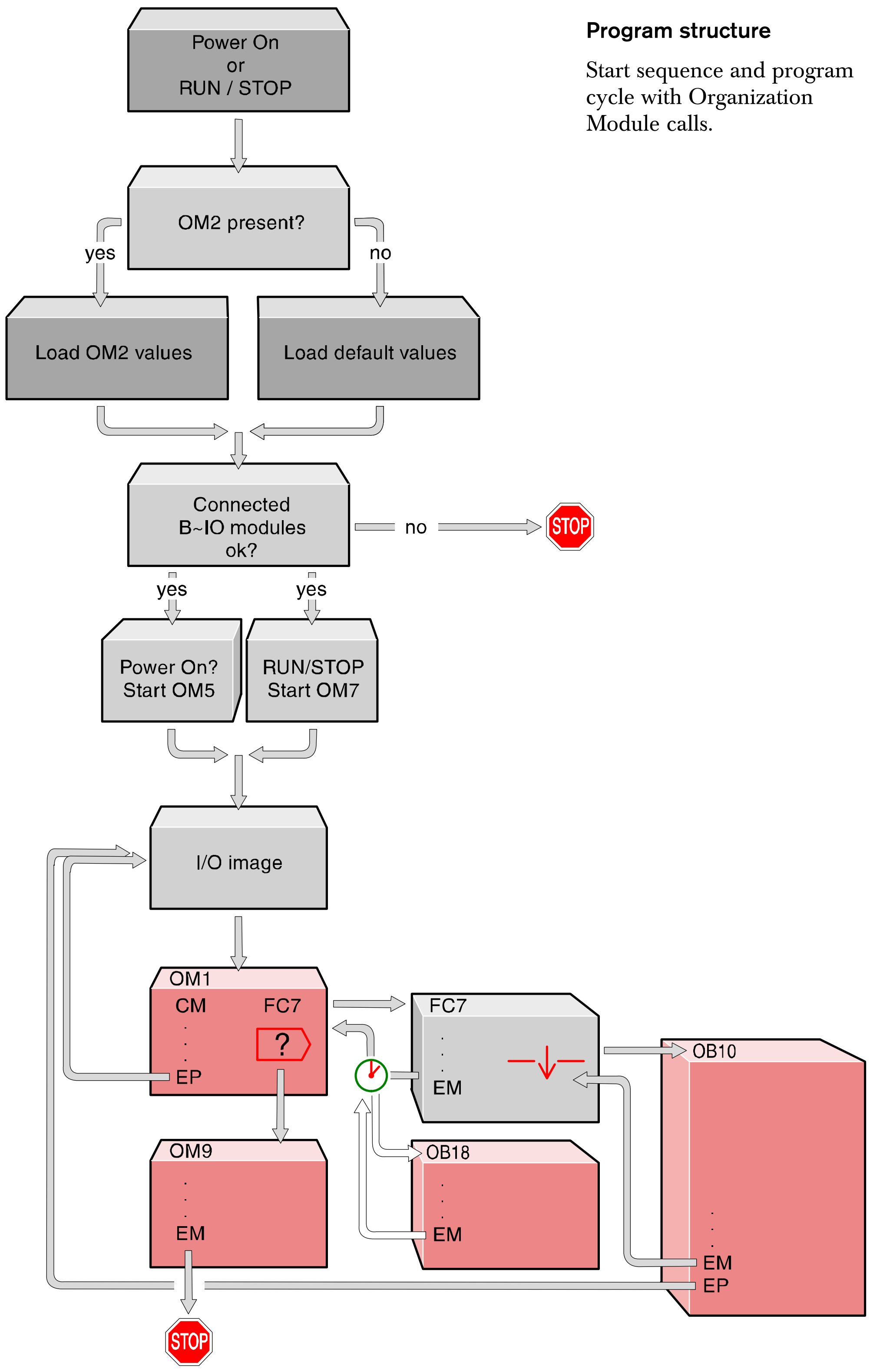
In case of control program error the OM9 program is executed before the controller changes to Stop mode. This action, for example, recovers important data from remanent marker areas.

Time controlled modules



Time controlled modules are called when a specified time interval has expired and a running program block is terminated.
Time intervals can be edited in the PLC program.

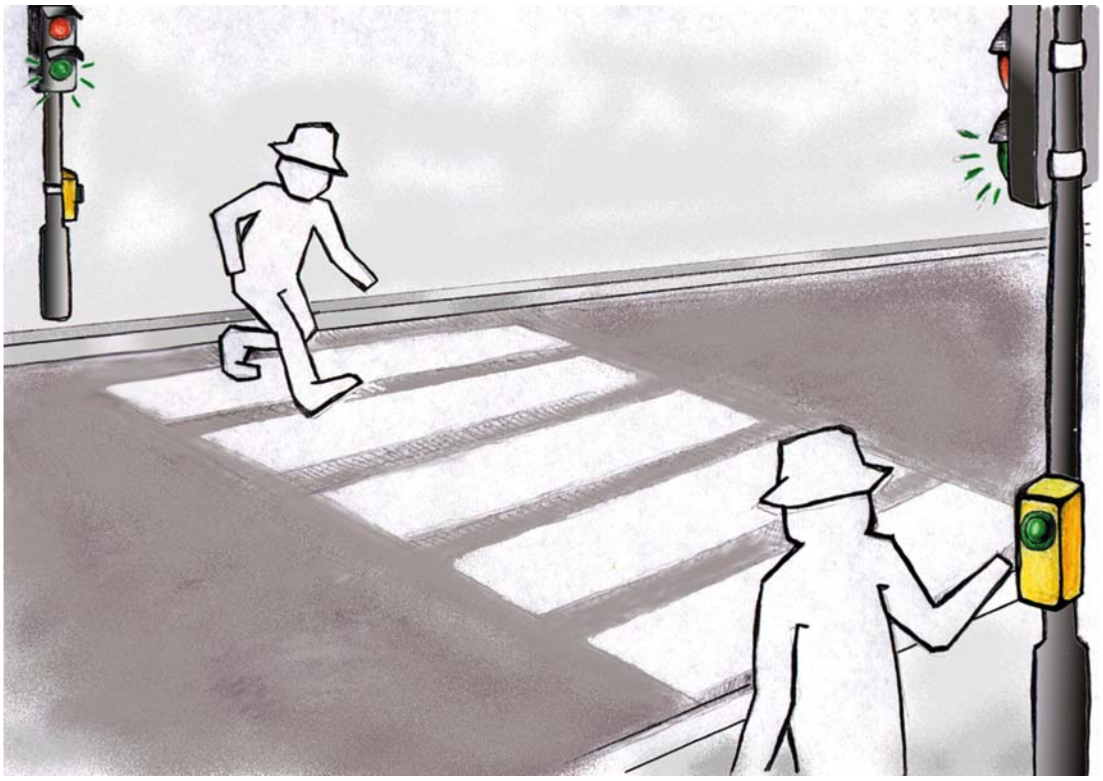
Program cycle



Program structure

Start sequence and program cycle with Organization Module calls.

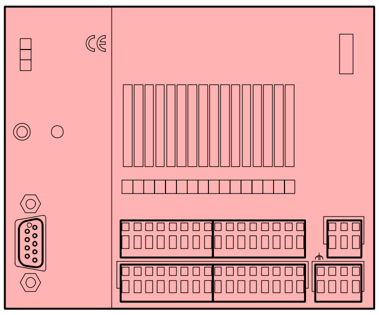
Programming with WinSPS



Using the CL150 to control crossover lights

In basic state the crossover lights should display red. The signal should toggle to green 10 seconds after a pedestrian who wishes to cross the road has actuated the request button. The pedestrian has 15 seconds to cross the road; after that time the signal toggles to red again.

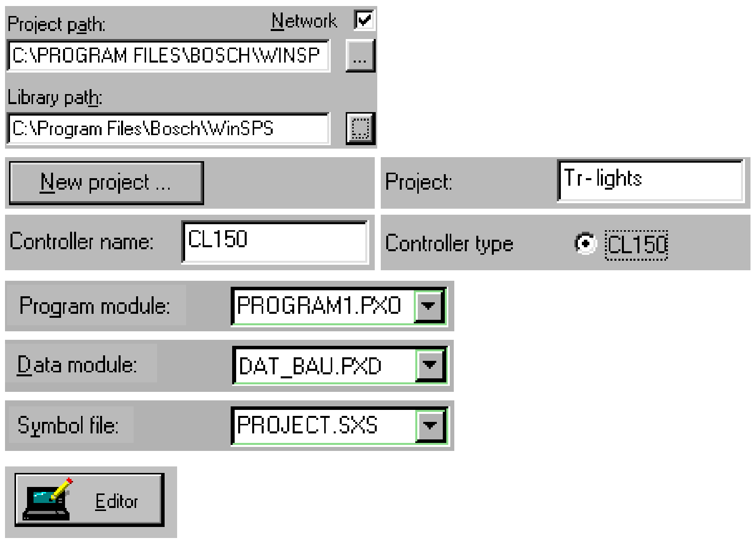
In the following chapter we will develop a control program that meets these demands. You can find helpful information on handling WinSPS in the Chapter ”The Program WinSPS”.



CL150
Basic equipment:
CL150 with 8 digital inputs and 8 digital outputs

Programming the crossover light controls

Default configuration
Open WinSPS and customize your default settings.



Project path: C:\PROGRAM FILES\BOSCH\WINSP
Network [✓]
Library path: C:\Program Files\Bosch\WinSPS
Project: Tr-lights
Controller name: CL150
Controller type: [✓ CL150]
Program module: PROGRAM1.PXO
Data module: DAT_BAU.PXD
Symbol file: PROJECT.SXS

You can choose the names freely. However, it is important that you subsequently enter the correct names in the symbol file.

After having completed your default customization, call the editor.

Call the editor

Customizing the symbol file

Symbol

Open the symbol file and enter the following:

In the row

```
OM1,R      ORG_BAU
```

(Name of your Program Module)

In the row

```
FCO,R      PROGRAM1
```

(Here you can name your second Program Module)

In the row

```
DMO,R      DAT_BAU
```

(Name of your Data Module)

Declare all other operands below. You can freely choose the sequence.

You should initially plan your I/O and marker assignments and enter them in the symbol file. You can modify them at any time.

An alert in the editor's status bar will warn you if you are using programming elements that you have not yet entered in the symbol file. This alert is cleared after you have updated the symbol file.

```
I0.0  Request_button

O0.0  Crossover_lights_red
O0.1  Crossover_lights_green

M0.0  Request_Marker
M0.1  Time_Marker_1_Crossover_red_green
M0.2  Time_Marker_2_Crossover_green_red

M1.0  Step_marker_1
M2.0  Step_marker_2

T1    Time_red_green
T2    Time_green_red
```

Programming the Organization Module OM1

The Organization Module OM1 is displayed by default when you open the editor.
If this is not the case, select the file "Org_Mod.pxo" in the file selection window.

Call the program module FC0 with the help of OM1. Insert the CM row for the module call above the EM row.

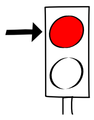
```
CM   FC0     ; Creating the program module FC0
```

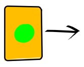
Save the file.

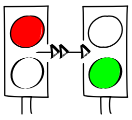
Creating the program module FC0

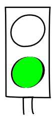
Create this program module and name it ”program1.pxo”.

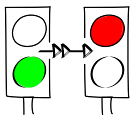
Switch to the FBD view and enter the program networks.

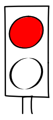

1 Set crossover signal to red

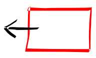

2 Evaluate the request button


3 Time delay Red -> Green


4 Switch crossover signal to green


5 Time delay Green -> Red

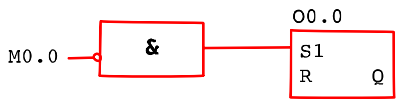

6 Switch crossover signal to red


7 Module end

The Program Module "program1.pxo"

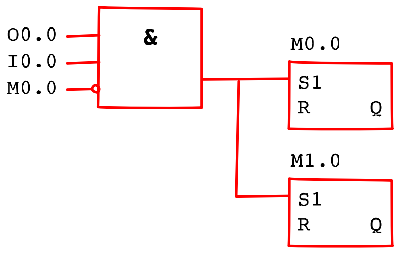
Network 1, Set crossover signal to red

```
AN     M0.0    ; Request Marker off
S      O0.0    ; Crossing signal red
```



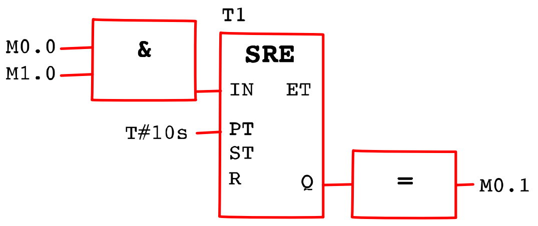
Network 2, Evaluate the request button

```
A      O0.0    ;Crossing signal is red and
A      I0.0    ;Request button actuated and
AN     M0.0    ;Request Marker not set
S      M0.0    ;Set request Marker
S      M1.0    ;Step Marker 1
```



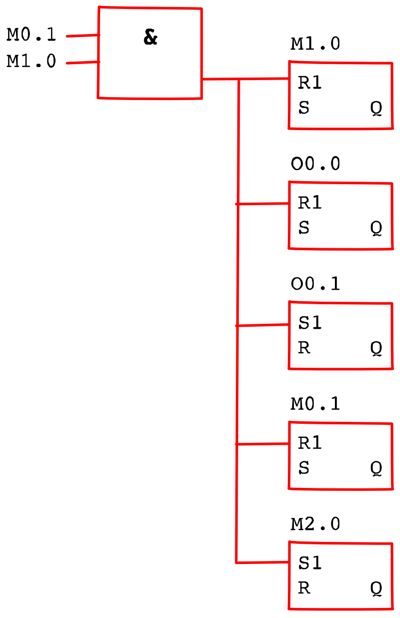
Network 3, Time delay Red -> Green

```
A      M0.0     ;Request Marker is set
A      M1.0     ;and Step Marker 1 is set
L  W   T#10s,A  ;10 seconds
SRE    A,T1     ;Rise-time delay
L  W   T1,A     ;Load timer
A      T1       ;Timer running?
=      M0.1     ;First time Marker is set
                ;if T1=0
```



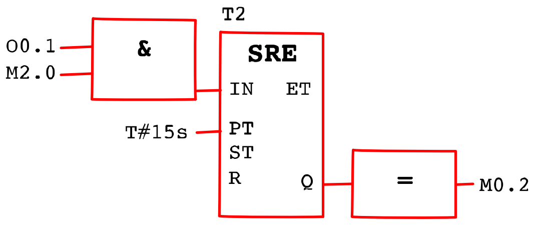
Network 4, Switch crossover signal to green

```
A      M0.1    ;First Time Marker is set
A      M1.0    ;and Step Marker 1 is set
R      M1.0    ;Reset step Marker 1
R      O0.0    ;Pedestrian red signal off
S      O0.1    ;Pedestrian crossing lights green
R      M0.1    ;Reset first Time Marker
S      M2.0    ;Step Marker 2 is set
```



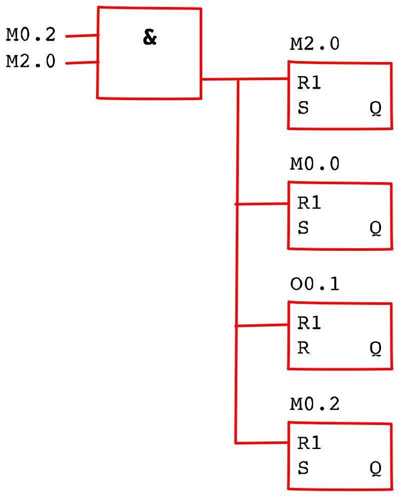
Network 5, Time delay Green -> Red

```
A      O0.1      ;Pedestrian crossing lights
                  ;green?
A      M2.0      ;Step Marker 2 is set
L  W   T#15s,A   ;15 seconds green
                  ;for pedestrians
SRE    A,T2      ;Rise-time delay
L  W   T2,A      ;Load timer
A      T2        ;Timer is running
=      M0.2      ;Second time Marker is set
                 ;if T2=1
```



Network 6, Switch crossover signal to red

```
A      M0.2      ;Second time Marker is set
A      M2.0      ;and step Marker 2 is set
R      M2.0      ;Reset step Marker 2
R      M0.0      ;Reset Request Marker
R      O0.1      ;Pedestrian crossing green signal off
R      M0.2      ;Reset second time Marker
```



Network 7, Module end

```
EM
```

EM

Download the program to the controller

Switch on the controller. Ensure that the operation switch of the CL150 is in ”Run” position. To transfer the program to the controller, select ”Load” in the menu ”Controller”. Do not make any changes in the ”Load” window. Confirm your entries with OK.

The program is compiled during this loading process. You will be warned of any errors in the program syntax.

Open the Monitor view

After you have successfully transferred your program to the controller, output O0.0 should light up after you have switched the controller to RUN mode. This output represents the red signal of the crossover lights.

Monitor

Now call the Monitor.

The rotating Bosch Anchor in the upper right of the Monitor dialog box indicates that the controller is running.

Testing the control program

You have two program testing options:
Simulate the request key by connecting a switch to input O0.0 of the controller or via Fixing function.

You only need to actuate this corresponding switch to monitor the program reaction in the Monitor view.

Force

Call Fixing.

To restart the cycle, you must set and reset input O0.0. If O0.0 stays set the process enters a kind of endless loop.

Fixing

Step	**Simulation of**	**Inputs**		**Fixing ”Load”**	**Outputs**		**Explanation**
		on	**off**		**on**	**off**	
0	Traffic signal red					o0.0	Red light is lit at program start
1	Actuate the request button	I0.0		I0.0 BOOL 1	O0.0		Request button is actuated
2	Release the request button		i0.0	I0.0 BOOL 0	O0.0		Request button is released

The additional program steps proceed without user intervention.

Programming the counter

In a second step the existing program will be extended:

The counter is to determine how often the request button has been actuated within the 10 seconds until the crossover light toggles to green signal. If a crossover light change has been requested five times or more, the green phase will subsequently be extended from 15 seconds to 30 seconds.

Open the Explorer. In your program path (C:\BOSCH\Tr-lights, if you have adhered to the default specified above), create a new subdirectory ”STEP2” at the same level as ”STEP1”. Then, copy the STEP1 subdirectories to the new directory. The first program section is now available for editing your expansion in STEP2.

Click on Project Name in the WinSPS default configuration. The box below should now display the STEP1 and STEP2 fields. Click on STEP2 and open the editor.

First again, customize the symbol file.

```
I0.0  Request_button

O0.0  Crossover_lights_red
O0.1  Crossover_lights_green

M0.0  Request_marker
M0.1  Time_Marker_1_Crossover_red_green
M0.2  Time_Marker_2_Crossover_green_red

M0.3  Counter_Marker
M0.4  Time_meas_2_cross_green_red_Counter_0
M0.5  Marker_to_set_Counter
M0.6  Marker_for_T2
M0.7  Marker_for_T3

M1.0  Step_marker_1
M2.0  Step_marker_2

T1    Time_red_green
T2    Time_green_red
T3    Time_green_red_Counter_not_0

C1    Request_Counter
```

You do not need to edit the Organization Module OM1.

New Network
Now, open the program module FC0 you have named ”program1.pxo” and enter your changes.

Here, you must recreate the networks displayed in black color, or modify them if they already exist. The new program parts to be changed are highlighted in the source code below.

1 Set crossover signal to red

2 Evaluate the request button

3 Time delay Red -> Green

4 Set counter and decrement

5 Switch crossover signal to green

6 Time delay Green -> Red

7 Determine the T2 Marker

8 Time delay Red -> Green, if counter = 0

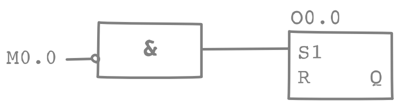
9 Determine the T3 Marker

10 Switch crossover signal to red

11 Module end

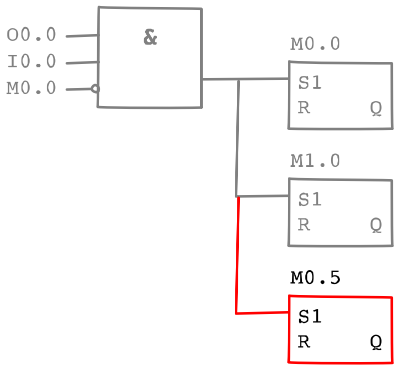
Network 1, Set crossover signal to red

```
AN    M0.0    ; Request Marker off
S     O0.0    ; Crossing signal red
```



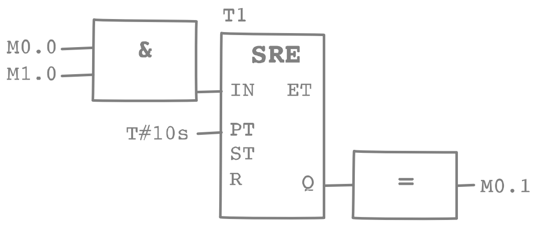
Network 2, Evaluate the request button

```
A     O0.0    ;Pedestrian crossing signal is red and
A     I0.0    ;Request button actuated and
AN    M0.0    ;Request Marker not set
S     M0.0    ;Set request Marker
S     M1.0    ;Step Marker 1
S     M0.5    ;Marker is set
```



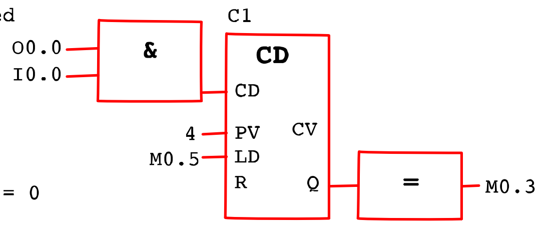
Network 3, Time delay Red -> Green

```
A     M0.0     ;Request Marker is set
A     M1.0     ;and Step Marker 1 is set
L  W  T#10s,A  ;10 seconds
SRE   A,T1     ;Rise-time delay
L  W  T1,A     ;Load timer
A     T1       ;Timer running?
=     M0.1     ;First time Marker is set
               ;if T1=0
```



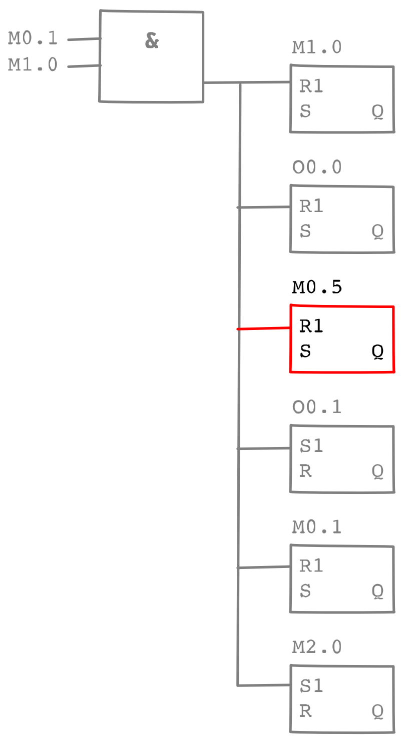
Network 4, Set counter and decrement

```
A     I0.0     ;Pedestrian crossing lights red
A     O0.0     ;Request button actuated
CD    C1       ;Counter  = Counter - 1
A     M0.5     ;Marker is set
L  W  4,A      ;Load counter status 4
SCY    A,C1    ;Set counter
L  W  C1,A     ;Transfer counter to A
A     C1       ;Counter 1 not 0 | Counter  1 = 0
=     M0.3     ;M0.3 = 1        | M0.3 = 0
```



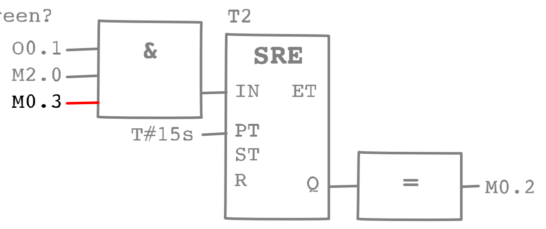
Network 5, Switch crossover signal to green

```
A      M0.1    ;First time Marker is set
A      M1.0    ;and Step Marker 1 is set
R      M1.0    ;Reset step Marker 1
R      O0.0    ;Pedestrian red signal off
R      M0.5    ;Reset set Marker
S      O0.1    ;Pedestrian crossing lights green
R      M0.1    ;Reset first time Marker
S      M2.0    ;Step Marker 2 is set
```



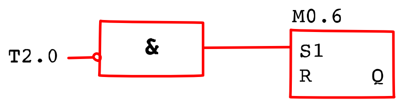
Network 6, Time delay Green -> Red

```
A      O0.1     ;Pedestrian crossing lights green?
A      M2.0     ;Step Marker 2 is set
A      M0.3     ;Counter Marker is set
L   W  T#15s,A  ;15 seconds green
                ;for pedestrians
SRE    A,T2     ;Rise-time delay
L   W  T2,A     ;Load timer
A      T2       ;Timer is running
=      M0.2     ;Second time Marker is set
                ;if T2=0
```



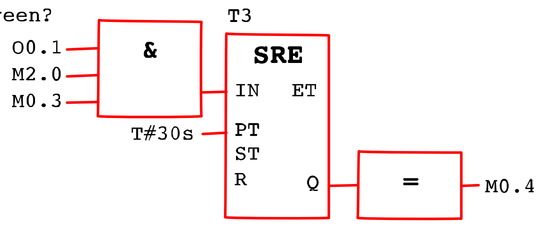
Network 7, Determine the T2 Marker

```
AN     T2      ;If the timer is running, T2 = 0
S      M0.6    ;M0.6 is set
```



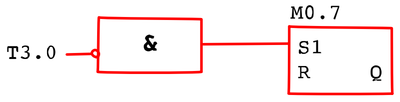
Network 8, Time delay Red -> Green, if counter = 0

```
A      O0.1     ;Pedestrian crossing lights green?
A      M2.0     ;Step Marker 2 is set
A      M0.3     ;Counter Marker is set
L   W  T#30s,A  ;30 seconds green
                ;for pedestrians
SRE    A,T3     ;Rise-time delay
L   W  T3,A     ;Load timer
A      T3       ;Timer is running
=      M0.4     ;Second time Marker is set
                ;if T3=0
```



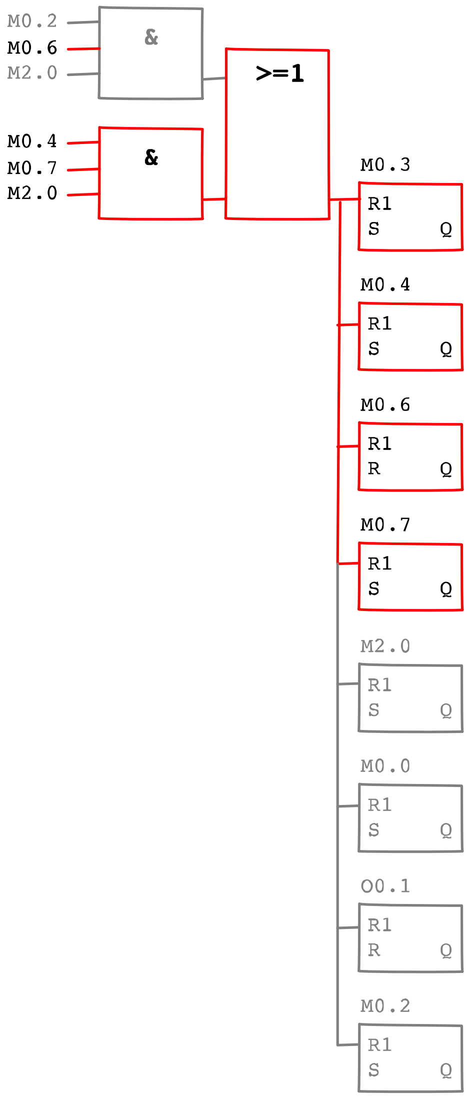
Network 9, Determine the T3 Marker

```
AN    T3     ;When the timer is running, T2 = 0
S     M0.7   ;M0.6 is set
```



Network 10, Switch crossover signal to red

```
A     M0.2   ;Second time Marker is set
A     M0.6   ;and T2 Marker is set
A     M2.0   ;and second step Marker is set
O     M0.4   ;or fourth time Marker is set
A     M0.7   ;and T3 Marker is set
A     M2.0   ;and second step Marker is set
R     O0.1   ;Pedestrian green signal off
R     M0.0   ;Reset Request Marker
R     M0.2   ;Reset second time Marker
R     M0.3   ;Reset counter Marker
R     M0.4   ;Reset fourth time Marker
R     M0.6   ;Reset T2 Marker
R     M0.7   ;Reset T3 Marker
R     M2.0   ;Reset step Marker 2
```



Network 11, Module end

```
EM
```

EM

Loading, testing and modifying the program

Download the program to the controller with ”Controller -> load”. Now open the Monitor and test the program by setting and enabling I0.0 six times. However, repeat this sequence only once or twice.

You should now make a few changes in the program in order to familiarize yourself with WinSPS and the CL150.

Open the Editor again.

Edit

Slightly modify the program and reload it to the controller.

You can, for example, extend or reduce times; or, you can add comments to disable markers that, on first sight, appear to you to be superfluous. The program process will then be different - monitor the reactions in the Monitor and find out why the program reacts this way.

In short, feel free to play around with the program. Maybe you will come up with a solution you like better than the existing one!

Structured programing

Structured programming is used to split the program into clearly organized, functionally and technologically interrelated modules for your PLC configuration.

Each one of the modules carries out a partial operation in the PLC configuration.

The advantages of structured programming are:

- a clearly organized program structure,
- programming and testing of subroutines,
- testing and editing of PLC jobs by the project team members
- multiple use of program modules
- simplified troubleshooting
- clear readability
- shorter programs

Organization Modules and Program Modules are available to realize structured programming.

Organization Modules (OMs) are called directly by the CL150 operating system. Therefore, they are mainly used for jobs closely related to the system.

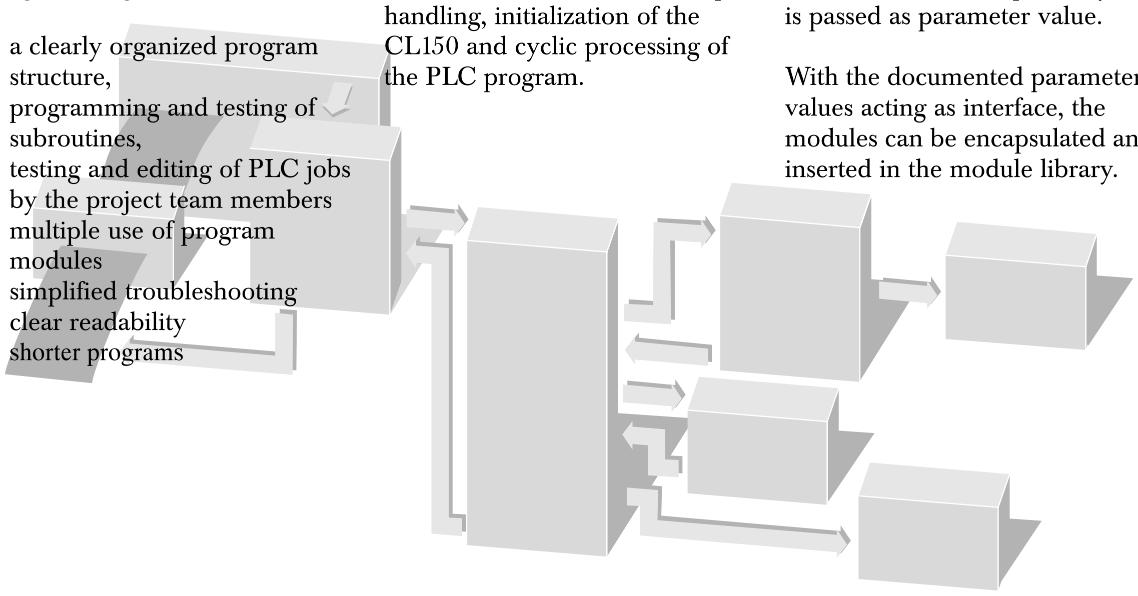
The tasks of Organization Modules include error/interrupt handling, initialization of the CL150 and cyclic processing of the PLC program.

The PLC program is edited in the program modules.

Each one of the program modules should describe a functionally interrelated task. It can call up further program modules to process part of this task.

Program modules can be configured. When the module is called, the data required by it is passed as parameter value.

With the documented parameter values acting as interface, the modules can be encapsulated and inserted in the module library.



Module call with parameters

Input parameters are passed to the program module when it is called. The subordinate module returns the results in output parameters.

WinSPS manages

Input parameters	VAR_INPUT
Output parameters	VAR_OUTPUT
I/O parameters	VAR_IN_OUT

The number of parameters is returned when the module is called.

```
;Module  call                ;Module  FC1

CM      FC1,2                A       I0.0
P0      M1.1                 AN      P0
P1   W  M20                  =       O0.2


                             L    W  P1,A
                             INC  W  A,1
                             T    W  A,P1
```

Structuring the control program

The network of a sample program shows you how to configure and call a module. The crossover light function is not changed in this process.

We shall use Network no. 3 ”Time delay red -> green”. The master program calls and transfers the timer function and the parameters to a separate program module.

Customizing the symbol file

In the symbol file, you must specify the new program module you are using for this configuration.
The new module will be called ”timer1.pxo”.
Enter the following row below the row calling the module ”program1.pxo”

```
FC1,R     TIMER1
```

Save the symbol file.

Configuring a module

First, we shall create the program module ”timer.pxo”.

This program module is to be called with three parameters. First, create the new module and then customize your parameter data.

You can edit the parameters in a separate window of the WinSPS Editor under “Edit -> Edit Parameter List...”.

Input data for the parameter header:

```
P0 Bool Step_Marker_1

VAR_INPUTP

P1 Bool Request Marker

VAR_INPUTP

P2 Bool Time_Marker_1

VAR_OUTPUT
```

The new program module “timer1.pxo” consists of two networks

1 Timer for red-green delay

2 Module end

Copy network 3 from the module ”program1.pxo” to the new module and replace

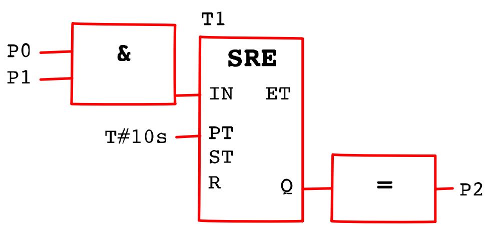
M0.0	with	P0
M1.0	with	P1 and
M0.1	with	P2.

Save the completed module ”timer1.pxo”.

The new program module ”timer1.pxo”

Network 1, Timer for red > green delay

```
A      P0        ;Request marker is set
A      P1        ;and step marker 1 is set
L  W   T#10s,A  ;10 seconds period
SRE    A,T1      ;Rise-time delay
L  W   T1,A      ;Load time
A      T1        ;Time running?
=      P2        ;First timer marker is set
                 ;if T1=1
```



Network 2, Module end

```
EM
```

EM

Customizing the program module ”program1.pxo”

Now the first program module ”program1.pxo” remains to be customized:

Delete the contents of Network 3 without deleting the network itself.

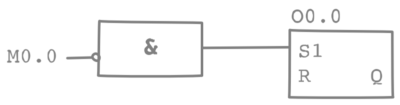
Subsequently, insert the parameter list by selecting ”Edit -> Call parameter list”.

WinSPS queries the parameter list in module ”timer1.pxo” and inserts it with its symbolic operands.

Replace these symbolic operands with the absolute parameters M0.0, M1.0 and M0.1.

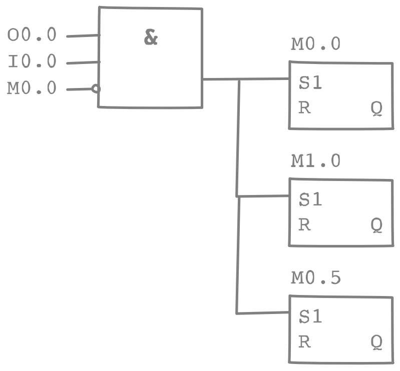
Network 1, Set crossover signal to red

```
AN     M0.0    ; Request Marker off
S      O0.0    ; Crossing signal red
```



Network 2, Evaluate the request button

```
A      O0.0    ;Pedestrian crossing signal is red and
A      I0.0    ;Request button actuated and
AN     M0.0    ;Request Marker not set
S      M0.0    ;Set request Marker
S      M1.0    ;Step Marker 1
S      M0.5    ;Marker is set
```

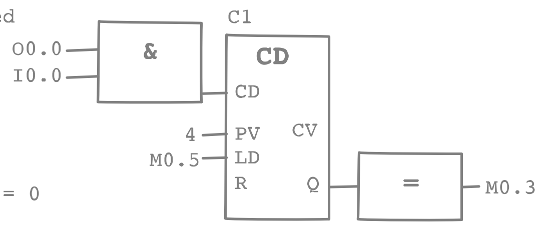


Network 3, time delay red > green via module call

```
CM       FC1,3
;
P0     M0.0    ;VAR_INPUT
P1     M1.0    ;VAR_INPUT
P2     M0.1    ;VAR_OUTPUT
```

Network 4, Set counter and decrement

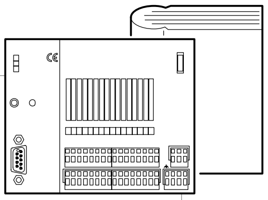
```
A      O0.0     ;Pedestrian crossing lights red
A      I0.0     ;Request button actuated
CD     C1       ;Counter  = Counter - 1
A      M0.5     ;Marker is set
L  W   4,A      ;Load counter status 4
SCY     A,C1    ;Set counter
L  W   C1,A     ;Transfer counter to A
A      C1       ;Counter 1 not 0 | Counter  1 = 0
=      M0.3     ;M0.3 = 1        | M0.3 = 0
 .                                            .
 .                                            .
 .                                            .
```



Network 5 to Network 11, Module end

```
EM
```

EM



Memory structure

The CL150 is equipped with RAM and Flash EPROM memory modules.

The RAM area is a volatile read-write memory that must be battery buffered to protect against power failure, thus protecting against the loss of the PLC program and remanent data.

EPROM memory is non-volatile, which means that program and data modules loaded to the EPROM are still available after a power failure. However, the actual data of remanent areas and of the data field are lost if no buffer battery is installed.

CL150 operating modes

The CL150 operates in two modes: with and without battery. In battery mode, all memory data and the setting of the real-time clock are retained when the power supply is switched off. After the power supply is switched on again the controller picks up operation with the values contained in the memory before power was switched off.

With operation without battery, the RAM and data memory restart in an undefined state after power is switched on again. The real-time clock is set to 01.01.00 / 00:00, the weekday is not defined. All modules contain their initialization values. User data are deleted.

You declare the operating mode of the CL150 in the Initialization Flag DW02 bit 7, of the Initialization Module OM2:

DW0, bit 7=0	no battery
DW0, bit 7=1	battery mode

Delivery state of the CL150: no battery mode.

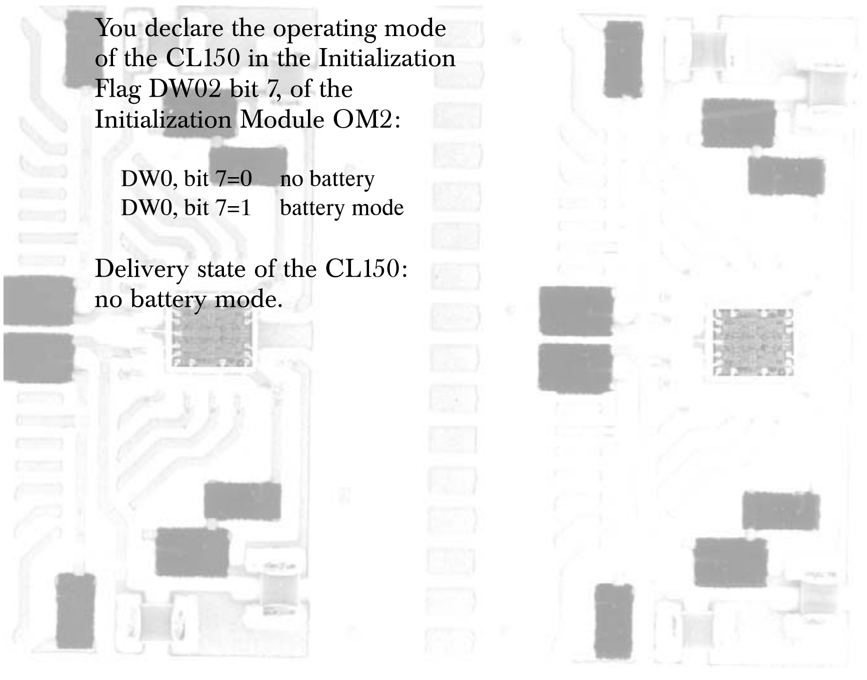
Remanent behavior

The CL150 provides a remanent memory area for storing operand values. The memory is battery buffered and protects data in case of power failure and operating mode transitions Run/Stop and Network On. This remanence area can be modified in OM2.

Without changes in OM2, the remanent memory area contains:

Marker M76	to M152
Counter C32	to C63
Times T64	to T127

The complete Data Field, Data Modules and Monitor Fixings also stay remanent.



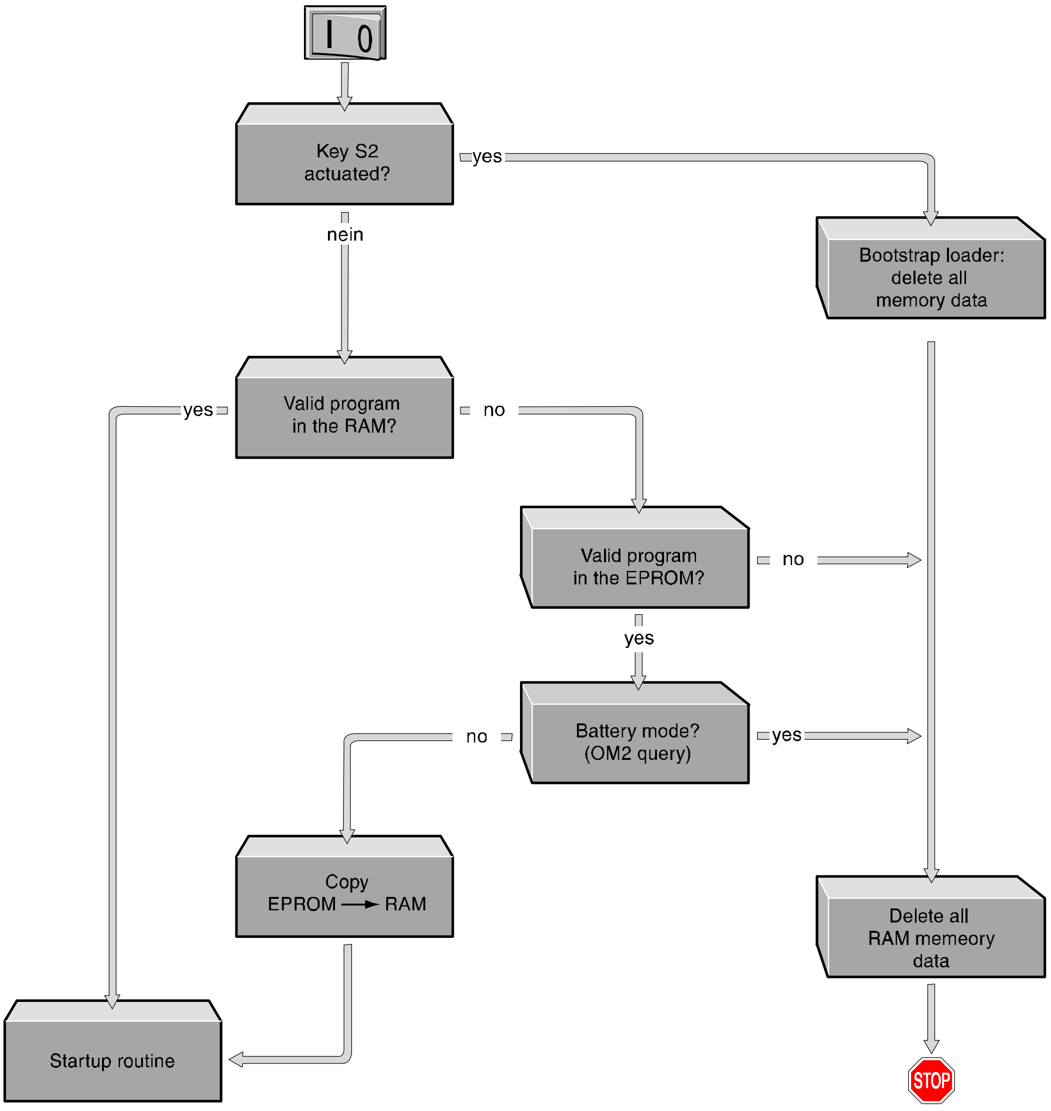
CL150 startup modes

The CL150 can load new programs from EPROM memory, independent of a programming device.

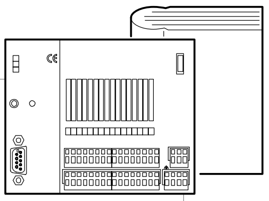
After it is switched on, the CL150 first tries to locate a valid program in the RAM or EPROM memory.

If a program exists there the CL150 loads it into the RAM.

Load program



I 0
Key S2 actuated?
yes
nein
Bootstrap loader: delete all memory data
Valid program in the RAM?
yes
no
Valid program in the EPROM?
no
yes
Battery mode? (OM2 query)
no
yes
Copy EPROM → RAM
Delete all RAM memeory data
Startup routine
STOP



In the CL150 controller you can utilize two fast hardware counters or, as an alternative, three interrupt inputs. The controllers CL150A and CL151A can be used to process analog values.

CL150 interfaces

Onboard hardware counters

These are two fast 32-bit up/down hardware counters. They can count pulses, regardless of the PLC I/O cycle. Both counters can process their own tasks. You can configure I0.0/ I0.1 and I0.2/I0.3 as fast counters.
The default setting of both counters is specified in the Organization Module OM2. When using these counters you must implement this Organization Module in the PLC program.

Declared is
- the counter default values,
- an upper and lower setpoint,
- outputs to be set instantaneously after the setpoint is reached,
- control data for edge detection and default count direction.

Interrupt inputs

With the help of the three alarm/ interrupt inputs I0.0 to I0.2 the PLC can react instantaneously, regardless of program signal transitions at the input.

An input signal transition from 0 to 1 triggers a peripheral interrupt in the PLC. This interrupt stops the current PLC program process and starts one of the three Organization Modules OM10, OM11 or OM12.

This input function is enabled if the interrupts are enabled and if OM10 to OM12 are implemented in the program.

One Organization Module is assigned to each interrupt input. The first input has the highest priority. This means that the program in Organization Module OM10 can not be interrupted by an interrupt of the second or third input.

The PLC program continues at the break point after it has processed the interrupt request.

Analog data processing

X24

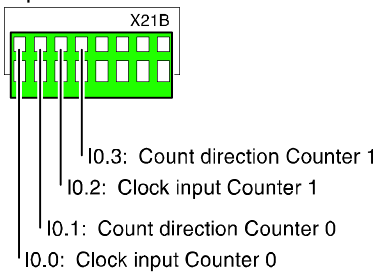
The CL150A and CL151A are equipped with two analog inputs and one analog output. Without additional modules they can therefore process measuring data, for example, for monitoring temperature, filling levels and pressure or tasks for controlling motor controllers or slide valves.

The analog signals are connected to the X23/X24 interfaces. These analog inputs operate with a resolution of 10 bits and in a voltage range of 0 to 10 V.

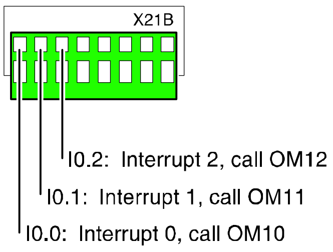
At the analog output the signal is made available on two cables as voltage value (X12) or current value (X13). The nominal range of the voltage signal is 0...10 V or -10...+10 V, that of the current signal is 0...20 mA.

The system area of the controller contains the data for analog data processing.

Inputs


X21B
I0.3: Count direction Counter 1
I0.2: Clock input Counter 1
I0.1: Count direction Counter 0
I0.0: Clock input Counter 0

Inputs


X21B
I0.2: Interrupt 2, call OM12
I0.1: Interrupt 1, call OM11
I0.0: Interrupt 0, call OM10

Technical data of the CL15x, CL15xA

Classification:
Control systems of the lower and medium performance class

Module rack dimensions (WxHxD) in mm 123/184 x 105 x 38 without connectors

Weight 350 g / 500 g
24-V voltage supply

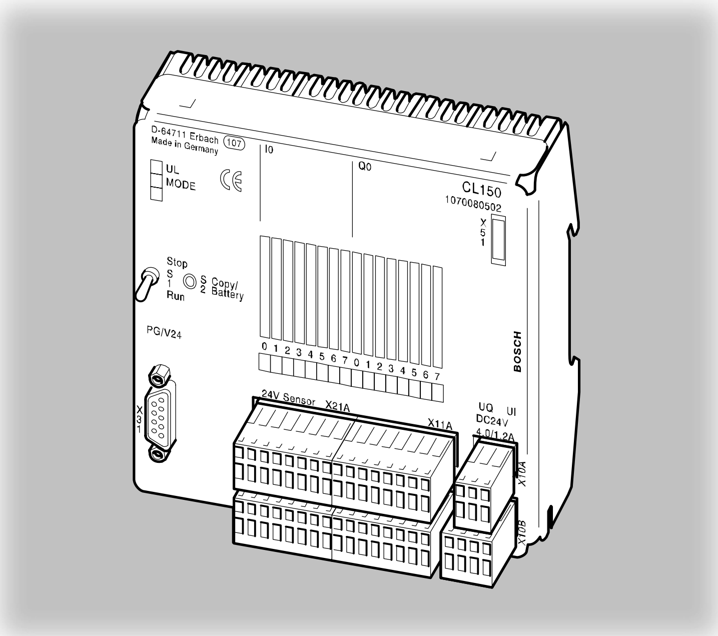
Current consumption up to 1.2 A, inrush current up to 25 A

Editing time
Bit instruction min. 0.6µs
Word instruction min. 0.6µs
Module instruction min. 46 µs

I/O image in 0.6 to 1.9 ms

64 KB RAM program memory
and
64 KB Flash EPROM

RAM buffering with lithium battery



12 Organization Modules,
OM1 to OM3, OM5, OM7, OM9 to OM12, OM17 to OM19

128 Program Modules,
FC0 to FC127

128 Data Modules,
DM0 to DM127

8 kByte Data field,
DF0 to DF8191

256 bytes system area,
S0 to S255

Nesting depth
32 blocks

7 bracket levels

Address format: BIT, BYTE, WORD, DOUBLEWORD for constants

384 digital inputs
I0.0 to I47.7

256 digital outputs,
O0.0 to O31.7

1216 Markers,
M0.0 to M151.7

4 registers, Word length

128 timers,
T0 to T127

64 counters,
C0 to C63

2 fast counters,
32 bits, max. 10 kHz

Analog inputs:
CL15xA: 2
Analog outputs
CL15xA: 1

Literature

- Bosch CL150, CL151, CL150A, CL151A -DP, -CAN, -IBS, -DEV Controller Manual / Operation List Order no. 1070 072 188

Training

The Bosch Training Center ”AT-didactic” offers product training related to the economic use and operation of industrial controlling techniques.

- Programmable Logic Controllers (PLC)
- Numeric Controls (CNC)
- Robot Controls (RC)
- Electrical Servo-drive systems
- Welding controls

You can order the detailed training program by calling Germany +49 (0) 60 62 / 78-6 02 or per fax +49 (0) 60 62 / 78-8 33

Glossary

Absolute Jump	The jump is carried out regardless of the previous logic link result. See also Jump Instruction.
AND circuit	Serial logic contact link. The logic result is only 1 if all contacts are on current, that is, if all contact signals are 1.
ASCII	American Standard Code of Information Interchange - standardized code for character display
Battery RAM	Read / Write Memory. Buffered against power failure.
Baud rate	Dimension for the speed of data transfer
Binary	Numerals, data and information are displayed only with the characters 0 and 1.
Bit	Smallest unit that describes a "0" or "1" status.
Byte	1 Byte = 8 bit
Central unit	The PLC core - consists of the controlling unit and the calculation unit
CPU	Central Processing Unit – see Central Unit
Compare function	Logical or arithmetic comparison of operands (BIT, BYTE, WORD etc.)
Conditional jump	The jump is only carried out if the previous logic link result is 1. See also Jump Instruction.
Contact plan, LAD	Represents the control sequence with NO and NC contacts as common in relay technology.
Cross-reference	Determines all blocks and program rows for an operand which is used in the PLC program.
Cycle time	Time required for one cycle of the user program
Editor	User program which assists the following actions: input, modification, correction, saving and output of PLC programs.
EPROM	Erasable Programmable Read Only Memory. The memory content is non-volatile. However, it can be erased if required.
Fixing	Setting or resetting I/Os with the help of a programming device, regardless of the respective program dependent I/O status.
Function block language, FBL	Function block representing logical networks.
Hardware	All devices and accessories of a PLC, for example, the programming device, controller, printer, cables etc.
Input image	Memory for the "0" or "1" status of all PLC inputs
Instruction	The smallest self-contained step of a PLC program, for example, I/O links AND, OR etc.
Instruction List, STL	Presentation of a PLC program in which the instructions are listed in rows one below the other.
Interrupt input	Input with the highest priority. When the interrupt input is set, the normal program cycle is interrupted to process an interrupt routine.

Jump instruction	With a jump instruction the user can force the PLC program to quit at a defined point and perform a jump to continue program processing at another point of the program. There is a differentiation between absolute and conditional jump instructions.
LED	Light Emitting Diode - Luminescent diode optically displaying operating states
Marker	Memory for intermediate values with the status Set or Not Set.
Marker area	Number of available markers. The CL150 is equipped with a marker area of 152 markers
Monitor mode	Displays internal and external switching states on the programming device. See also Online.
Normally closed contact (NC)	Breaks the current circuit when it opens.
Normally open contact (NO)	The current circuit is connected when this contact closes.
Offline	The programming device is not connected to the PLC. PLC programs can be created without the controller.
Online	The programming device is connected to the PLC. While the machine or system which is to be controlled is in operation, the I/O status and the logic link results can be monitored on the programming device. See also Monitor mode.
Operands	This refers to inputs, outputs, markers, timers and counters. Addresses are part of the instruction set.
Operation	Part of the control instruction set which determines how to interlink operands, for example, with AND, OR, LOAD, ADD, COMPARE operations.
Optocoupler	Module for the separation of current circuits. The electrical signals of the primary current circuit are converted into a light signal. These signals are subsequently converted back into the original electrical signal in the secondary circuit, thus decoupling the secondary current circuit from the primary. This increases the PLC's immunity to interference.
OR circuit	Parallel logic link of two contacts. If one of the contact signals = “1“, it is under current, the logic result is a 1.
Output, short circuit-proof	Output on which the output current is retained within permitted limits in error case.
Output image	Result memory for logic link results that are passed to the outputs at the end of a PLC program.
Output signal	Current/voltage value for controlling series control units, power contactors, motors or valves.
Peripheral devices	All devices which can be connected to a PLC; for example, printers or text displaying devices
PLC	Programmable Logic Controller
Potential separation	Distribution of I/O signal lines to multiple current circuits with multiple reference points.
Program	Command set for the solution of certain tasks.
Program address	Identifier of program segments
Program memory	Semiconductor modules for storing user programs; for example, EPROM or RAM.

Program processing controlled by interrupt	The interrupt routine OM10, OM11 or OM12 is processed when the interrupt input is set. This interrupts cyclical PLC program processing.
RAM	Random Access Memory - Memory module for read / write operations. The memory is volatile in case of power failure if not battery buffered.
Remanent memory	Non-volatile memory. The content is remanent in case of power failure.
Serial interface	Transmits serial data over a single data line
Software	Collective term for programs
STL	See Instruction List
Transfer protocol	Standardized protocol for data communication. The data format and parameters of the communication are fixed. The aim is high communication reliability.

Index

Bosch Automation Technology

Industrial hydraulics
Robert Bosch GmbH
Geschäftsbereich
Automationstechnik
Industriehydraulik
Postfach 30 02 40
D-70442 Stuttgart
Fax +49 (0) 7 11 8 11-18 57

Drive and control technology
Robert Bosch GmbH
Geschäftsbereich
Automationstechnik
Antriebs- und Steuerungstechnik
Postfach 11 62
D-64701 Erbach
Fax +49 (0) 60 62 78-4 28

Mobile hydraulics
Robert Bosch GmbH
Geschäftsbereich
Automationstechnik
Mobilhydraulik
Postfach 30 02 40
D-70442 Stuttgart
Fax +49 (0) 7 11 8 11-17 98

Tightening and press-fit systems-
Robert Bosch GmbH
Geschäftsbereich
Automationstechnik
Schraub- und Einpress-Systeme
Postfach 11 61
D-71534 Murrhardt
Fax +49 (0) 71 92 22-1 81

Pneumatics
Robert Bosch GmbH
Geschäftsbereich
Automationstechnik
Pneumatik
Postfach 30 02 40
D-70442 Stuttgart
Fax +49 (0) 7 11 8 11-2 45 30

Deburring technology
Robert Bosch GmbH
Geschäftsbereich
Automationstechnik
Entgrattechnik
Postfach 30 02 07
D-70442 Stuttgart
Fax +49 (0) 7 11 8 11-3 34 75

Assembly technology
Robert Bosch GmbH
Geschäftsbereich
Automationstechnik
Montagetechnik
Postfach 30 02 07
D-70442 Stuttgart
Telefax +49 (0) 7 11 8 11-77 77

AT-didactic
Robert Bosch GmbH
Geschäftsbereich
Automationstechnik
didactic
Berliner Straße 25
D-64701 Erbach
Fax +49 (0) 60 62 78-8 33

Subject to technical modifications

Your concessionary

BOSCH

Robert Bosch GmbH
Geschäftsbereich
Automationstechnik
Antriebs- und Steuerungstechnik
Postfach 11 62
D-64701 Erbach
Fax +40 (0) 60 62 78-4 28

1070 072 346-101 (01.04) GB · HB IN · AT/PLS · Printed in Germany