

PROVIDING SAFETY

WIJ BESCHERMEN  
uw meest belangrijke kapitaal  
**UW MEDEWERKERS**





# Safety Simplifier Manual

Design, programming, installation, maintenance and decommissioning of Safety Simplifier Systems.

*Safety Simplifier makes it easy to create safe and reliable industry and workplace environments, using wireless safety communication.*

*Safety Simplifier can be customized with many different features and functions to fit your application.*



Plug in the Simplifier Monitor in your computer to monitor and program your Safety Simplifiers wirelessly in our software Simplifier Manager.

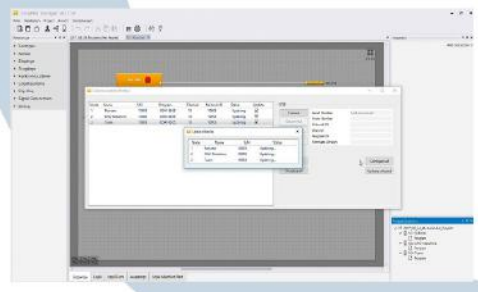


Safety

Simplifier

Simplifier Manager is the software used to create, download, and debug Safety Simplifier programs. Programs can be downloaded and debugged using a USB cable, or wirelessly using the Simplifier Monitor. Using the Simplifier Monitor eliminates the need to ever open the unit.

Debugging Safety Simplifier using the online mode displays the state of all internal signals in the system, making it easy to detect and correct logical errors in the program.



Simplifier Monitor

**WE SIMPLIFY SAFETY**



This document is the original document

All rights of this documentation are reserved by SSP North AB (Safety System Products North AB). Copies may be made for internal uses. Source code from third-parties and/or open source software has been used for some parts of the configuration software. The relevant license information available tough our home page ([www.sspn.se](http://www.sspn.se)).

SSP North AB  
Tullkammarvägen 14  
439 31 Onsala  
Sweden  
[www.sspn.se](http://www.sspn.se)

## Table of Contents

<b>1</b>	<b>Introduction .....</b>	<b>10</b>
1.1	This Manual .....	10
1.1.1	Abbreviations .....	10
1.1.2	Symbols .....	10
1.2	Intended use of Safety Simplifier .....	10
1.3	Contact .....	11
1.4	Who can implement Safety Simplifier .....	11
1.5	Warranty .....	11
1.6	Maintenance and repairs .....	11
1.7	Disposal considerations .....	11
1.8	Safety Assessment .....	12
1.9	Safety communication .....	12
<b>2</b>	<b>The Safety Simplifier System overview and features .....</b>	<b>13</b>
2.1	Features .....	14
2.1.1	Example input functions .....	14
2.1.2	Example Output Functions .....	14
2.1.3	Combined I/O.....	14
2.1.4	Redundant Relay Outputs .....	15
2.1.5	I/O outputs for “non-safety” information.....	15
2.1.6	LED information .....	15
<b>3</b>	<b>Safety Precautions when using Safety Simplifier .....</b>	<b>16</b>
3.1	Safety information regarding I/O and relay outputs of Safety Simplifier.....	19
3.1.1	The relay outputs .....	19
3.1.2	Solid-state transistor outputs .....	20
3.1.3	Safety inputs .....	22
<b>4</b>	<b>Technical data Safety Simplifier.....</b>	<b>23</b>
4.1	Safety parameters according to 610508 .....	25
<b>5</b>	<b>Dimensions and mounting .....</b>	<b>26</b>
<b>6</b>	<b>Safety Simplifier principle layout .....</b>	<b>29</b>
<b>7</b>	<b>Electrical Installation .....</b>	<b>30</b>
7.1	Making IO Connections.....	31
<b>8</b>	<b>Bus communication via radio and/or CAN-bus.....</b>	<b>32</b>
<b>9</b>	<b>Calculating PFH-d values .....</b>	<b>32</b>
<b>10</b>	<b>Calculating reaction time.....</b>	<b>34</b>
<b>11</b>	<b>Electrical diagrams for recommended connections of machines, PB:s and safety devices.....</b>	<b>35</b>
11.1	E-stop, Combo PB with indication could be used for reset, open, start, stop etc ..	35
11.2	E-stop, Combo PB with indication for Reset, Open, Start and Stop, Selectors.....	35

11.3	Two-hand connection according to EN 574.....	36
11.4	Safety devices with OSSD outputs such as: RFID- sensor/Safety Gate sensor, Scanners, Light beams and light curtains .....	36
11.5	Safety Gate limit switches .....	37
11.6	Relay outputs for contactors with feedback for supervision/monitoring .....	37
11.7	Relay outputs for machines and safety devices .....	38
11.8	OSSD outputs for machines, contactors and safety devices .....	38
11.9	The LED display menu.....	39
11.9.1	Navigation .....	39
11.9.2	Sub-menus .....	40
11.9.3	The menus.....	41
11.9.3.1	N (first level).....	41
11.9.3.2	N (second level).....	41
11.9.3.3	I/O (first level) .....	42
11.9.3.4	I/O (second level).....	43
11.9.3.5	U (first level).....	43
11.9.3.6	U (second level).....	44
11.9.3.7	U (third level) .....	44
11.9.3.8	M (first level) .....	45
11.9.3.9	M (second level) .....	45
11.9.3.10	 (Radio) (first level) .....	46
11.9.3.11	 (Radio) (second level) .....	46
11.9.3.12	CAN (first level).....	47
11.9.3.13	CAN (second level).....	47
11.9.3.14	CAN (third level) .....	48
11.9.4	Fatal error mode .....	48
11.9.4.1	Fatal error codes.....	48
11.9.5	Configuration mode .....	50
11.10	LED Display quick reference.....	51
11.10.1	First level menu information .....	51
11.10.2	Second level menu information .....	51
11.10.3	Third level menu information .....	52
<b>12</b>	<b>Introduction of the software Simplifier Manager.....</b>	<b>54</b>
<b>13</b>	<b>Simplifier Manager .....</b>	<b>54</b>
13.1	Notice .....	54
13.2	Windows 7 and 10 .....	54
13.3	Installation .....	54
13.3.1	External Tools .....	54
13.3.1.1	Compiler .....	54
13.3.1.2	PDF Viewer.....	54
13.3.1.3	Windows 7 USB Driver .....	54
13.4	User Interface Layout .....	55
13.4.1	The Main Menu.....	55
13.4.2	The Toolbar .....	58
13.4.3	Documents.....	58
13.4.4	The Toolbox .....	58
13.4.5	The Inspector.....	58
13.4.6	The Project Manager .....	58
13.5	Making your first program .....	58
13.5.1	Creating a Project .....	58
13.5.2	Hardware Configuration.....	59
13.5.3	The Logic Editor.....	60
13.5.4	Navigation in the logic graph .....	61

13.5.5	The Goal of this example.....	61
13.5.6	Adding Function Blocks.....	62
13.5.7	Moving Blocks.....	63
13.5.8	Deleting Blocks.....	63
13.5.9	Selecting Multiple Blocks.....	63
13.5.10	Copy, Cut, and Paste.....	63
13.5.11	Connections.....	63
13.5.12	Removing Connections.....	64
13.5.13	Final steps.....	65
13.5.14	Checking and Compiling.....	65
13.5.15	Saving and loading projects.....	65
<b>13.6</b>	<b>Expanding the program.....</b>	<b>66</b>
13.6.1	Local Memories.....	66
13.6.2	Adding Local Memories to the program.....	67
13.6.3	Managing Pages.....	68
13.6.4	Adding more nodes.....	70
13.6.5	Programming Node 2.....	70
13.6.6	Sending Information via Radio/CAN.....	71
<b>13.7</b>	<b>Online Mode.....</b>	<b>74</b>
<b>13.8</b>	<b>Downloading Programs.....</b>	<b>77</b>
13.8.1	Entering serial numbers.....	78
13.8.2	Download via USB.....	78
13.8.3	Download via radio with Simplifier Monitor.....	78
<b>13.9</b>	<b>Passwords.....</b>	<b>78</b>
13.9.1	Opening a password protected project file.....	79
13.9.2	Comments.....	79
13.9.3	The Comment property.....	79
13.9.4	The Comment block.....	79
<b>13.10</b>	<b>Designing systems with Radio or CAN communication.....</b>	<b>80</b>
13.10.1	Radio and CAN communication safety Notice.....	80
13.10.2	Radio.....	81
13.10.3	Selecting radio channel for radio communication.....	83
13.10.4	Designing systems that communicate via radio.....	84
13.10.4.1	Radio quality.....	84
13.10.5	Global and CAN Memories.....	86
13.10.5.1	Choosing Memory Numbers.....	86
13.10.5.2	Timeout and StartUp test.....	86
13.10.5.3	Best Practices.....	88
13.10.5.4	Controlling outputs directly via radio or CAN.....	88
<b>13.11</b>	<b>Input Functions.....</b>	<b>89</b>
13.11.1	Input States.....	90
13.11.2	Input signal types.....	91
13.11.2.1	VDC.....	91
13.11.2.2	OSSD.....	91
13.11.2.3	0V.....	91
13.11.2.4	A, B, C, D Pulses.....	91
13.11.2.5	Inverted A, B, C, D Pulses.....	91
13.11.3	Input properties.....	91
13.11.3.1	Enable ERROR Output.....	92
13.11.3.2	StartUp Test.....	92
13.11.3.3	Filter ON (ms).....	92
13.11.3.4	Filter OFF (ms).....	92
13.11.3.5	Enable Simultaneity.....	92
13.11.3.6	Simultaneity (ms).....	92
13.11.3.7	Enable Zero Time.....	93
13.11.3.8	Zero Time (ms).....	93

13.11.3.9	Comment .....	93
13.11.3.10	Terminal Count .....	93
13.11.3.11	Pin Properties .....	93
<b>13.12</b>	<b>Output Functions .....</b>	<b>94</b>
13.12.1	ERROR State .....	95
13.12.2	Feedback .....	96
13.12.3	Output signal types .....	96
13.12.3.1	VDC .....	96
13.12.3.2	0V .....	96
13.12.3.3	OSSD .....	96
13.12.3.4	A, B, C, D Pulses .....	96
13.12.3.5	Inverted A, B, C, D Pulses .....	96
13.12.4	The Advanced Output function block .....	96
<b>13.13</b>	<b>Fatal Errors .....</b>	<b>97</b>
<b>13.14</b>	<b>Logic .....</b>	<b>97</b>
13.14.1	Logical Loops .....	97
13.14.2	Unsafe and safe signals .....	98
<b>13.15</b>	<b>Serial Communication for status information .....</b>	<b>99</b>
13.15.1	Serial Encoding .....	99
13.15.1.1	Format .....	100
13.15.2	Serial Decoding .....	101
13.15.3	Serial communication via I/O Terminals .....	102
13.15.4	Configuration for communication with other PLCs .....	103
13.15.4.1	Receiving signals from other PLCs .....	103
13.15.4.2	Sending signals to other PLCs .....	103
13.15.4.3	Function blocks for communication with PLCs .....	103
<b>14</b>	<b>Function Block Reference .....</b>	<b>104</b>
<b>14.1</b>	<b>Inputs .....</b>	<b>105</b>
14.1.1	Two-Hand Device .....	105
14.1.2	Status Input .....	107
14.1.3	Selector Switch .....	108
14.1.4	Advanced Input .....	109
14.1.5	Door Sensor .....	111
14.1.6	Light Barrier .....	112
14.1.7	E Stop .....	114
14.1.8	Push Button .....	115
<b>14.2</b>	<b>Output .....</b>	<b>116</b>
14.2.1	Advanced Output .....	116
14.2.2	OSSD Output .....	117
14.2.3	Relay Output .....	118
14.2.4	Status Output .....	119
<b>14.3</b>	<b>Memory .....</b>	<b>120</b>
14.3.1	Memory .....	120
14.3.2	Reference .....	121
<b>14.4</b>	<b>Function .....</b>	<b>122</b>
14.4.1	Delay OFF .....	122
14.4.2	Delay ON .....	123
14.4.3	Single Reset .....	124
14.4.4	Sequence Reset .....	125
14.4.5	Stepping .....	126
14.4.6	Internal Input .....	127
<b>14.5</b>	<b>Logic .....</b>	<b>129</b>
14.5.1	NOT .....	129



14.5.2	AND .....	129
14.5.3	NAND.....	130
14.5.4	OR .....	132
14.5.5	NOR.....	133
14.5.6	XOR.....	134
14.5.7	XNOR .....	135
<b>14.6</b>	<b>Latches.....</b>	<b>136</b>
14.6.1	T Latch.....	136
14.6.2	SR Latch .....	137
<b>14.7</b>	<b>Signal Generators.....</b>	<b>138</b>
14.7.1	Logic 1 .....	138
14.7.2	Logic 0 .....	138
14.7.3	Square Wave .....	139
14.7.4	1Hz Blink.....	139
14.7.5	5Hz Blink.....	140
<b>14.8</b>	<b>Debug.....</b>	<b>142</b>
14.8.1	Fatal Error.....	142
14.8.2	CAN OK .....	143
14.8.3	Link OK .....	143
<b>14.9</b>	<b>Advanced .....</b>	<b>144</b>
14.9.1	Unsafe & Safe.....	144
<b>14.10</b>	<b>Non-Safety.....</b>	<b>144</b>
14.10.1	Serial Encoder .....	144
14.10.2	Serial Decoder .....	146
<b>14.11</b>	<b>Miscellaneous.....</b>	<b>147</b>
14.11.1	Comment .....	147

# Safety Simplifier

Introduction | System overview | Features | Safety  
Precautions | I/O and Relay outputs | Technical data |  
Installation | Bus communication | PFH-d-values |  
Reaction time | Electrical diagrams | LED display

## 1 Introduction

### 1.1 This Manual

This manual regards the design and use of Safety Simplifier systems.

The manual is valid until new documentation is published. The latest original instructions are available at [www.sspn.se](http://www.sspn.se).

The manual explains the function, the special safety design to consider for a safety system, the calculation of safety values for Safety Simplifier as an element of safety and how to connect and install the product.





Programming, downloading, updating and debugging Simplifier Systems is described in the software part of this manual.

#### 1.1.1 Abbreviations

CAN	Controller Area Network
DC	Diagnostic coverage
ESPE	Electro-sensitive Protective Device
EUC	Equipment Under Control
MRT	Mean Repair Time
OSSD	Output Signal Switching Device
PCB	Print Circuit Board
PFH	Mean probability of failure on demand
PFH-d	Mean probability of dangerous failure on demand
PL	Performance Level
PLC	Programmable Logic Controller
SFF	Safe Failure Fraction
SIF	Safety Instrumented Function
SIL	Safety Integrity Level
SSPN	Safety System Products North AB
USB	Universal Serial Bus

#### 1.1.2 Symbols

Different symbols are used throughout this manual and it is important to understand the meaning of each.

 <b>Warning!</b>	This symbol indicates a situation that presents immediate danger, and if not prevented will result in serious injury or death.
 <b>Caution!</b>	This symbol indicates a situation that presents potential danger, and if not prevented may result in injury or death.
 <b>Notice!</b>	This symbol indicates a potential danger, which may lead to damage to property and equipment if not prevented.
 <b>Note</b>	This symbol indicates useful tips and information.

### 1.2 Intended use of Safety Simplifier

The Safety Simplifier is intended for use in machinery covered by the Machinery Directive 2006/42/EC and for safety circuits according to ISO 13849-1 and IEC 61508 up to PL e, Cat. 4 and SIL 3. See technical data and CE declaration for Safety Simplifier for more information on intended use.

## 1.3 Contact

In case of a hazardous malfunction of a safety simplifier please contact SSP North AB using the contact information on the last page of this document. All necessary contact information can also be found at [www.sspnorth.se](http://www.sspnorth.se). SSP North AB can also be contacted through their distribution network as they are responsible to forward information which can be vital for the correct use and to prevent hazardous situations.

## 1.4 Who can implement Safety Simplifier

The Safety Simplifier may only be assembled, installed, programmed, commissioned, maintained and decommissioned by personnel of sufficient competence. Competent personnel are people who:

- have knowledge in operating the equipment in systems which Safety Simplifier will be a part of
- have knowledge in Safety Standards such as the machinery directive and other standards applicable to the application
- have knowledge in basic electronics
- have knowledge in the programming of Safety PLC-Systems
- have got training in using Safety Simplifier and Simplifier Manager
- have read and understood the contents of this manual

When reading the software part of this manual, it is recommended to have Simplifier Manager running on a computer nearby, and one or more Safety Simplifier units to test with. This is so the user can follow along the examples and get an understanding of how the functions work in practice. How to install Simplifier Manager is described in chapter 3.3.

## 1.5 Warranty

The SSP North AB products are covered by a warranty against material, construction and manufacturing faults. During the guarantee/warranty period, SSP North AB may replace the product or faulty parts. Work under guarantee/warranty must be carried out by SSP North AB or by an authorized service centre specified by SSP North AB.

The following faults are not covered by the guarantee/ warranty:

- Faults due to wear and tear from normal use
- Failure of parts of a consumable nature
- Failure of products that have been subject to unauthorized modifications
- Faults resulting from incorrect installation
- Faults resulting from incorrect use
- Water/moisture damage caused by environment exceeding IP65

## 1.6 Maintenance and repairs

Please follow the following guidelines for maintaining the product:

- Keep the product in a dry, clean place according to IP65
- Wipe off dust using a slightly damp, clean cloth



Repairs must only be carried out by SSP North AB or qualified personnel authorised by SSP North AB. Incorrect maintenance or repairs may cause unintended malfunction. Contact your representative if you require service or other assistance.

## 1.7 Disposal considerations

Please follow these guidelines for disposing the product:

- Check the mission time for using Safety Simplifier in safety systems
- For disposals, comply with the regulations in your region for electronics and other parts of the Safety Simplifier

## 1.8 Safety Assessment

Before designing a safety system, a safety assessment needs to be done in accordance with the Machinery directive and other applicable standards.

The functional safety is guaranteed for the Safety Simplifier as an element of a system. The design and the programming of a safety system using Safety Simplifier and the software Simplifier Manager is the sole responsibility of the installer/user. The complete system must be tested with all connected elements to verify that it fulfils the safety requirements. Any change of the safety system requires new testing and documentation.

The Safety Simplifier system is designed to achieve up to SIL 3, SILCL 3, PL e, Cat. 4, Type 4 in accordance with the IEC 61508 and ISO 13849-1. The SIL and PL of an application using Safety Simplifier also must include the parameters of the safety devices, machine control systems and other relevant equipment. Standards that are relevant for a certain application must also be followed. See the CE declaration for Safety Simplifier for information about applicable standards for which the Safety Simplifier is approved for and intended for.

## 1.9 Safety communication

The CAN and radio safety communication channels fulfil IEC 61784-3:2016 and are only used as physical layers.

## 2 The Safety Simplifier System overview and features

The Safety Simplifier system is a PLC-system, designed to be a part of an Equipment Under Control (EUC) requiring a Safety Integrated Level (SIL) up to SIL 3 according to IEC 61508 and Ple, cat 4 according to 13849-1. The Safety Simplifier is a Safety PLC intended to solve logic for one or several safety functions in an EUC.

The safety connections to safety devices and machines are intended to be done via the I/O: s of Safety Simplifier. The Safety Simplifier/s can also communicate with each other via radio and /or CAN-bus cable for the safety logic.

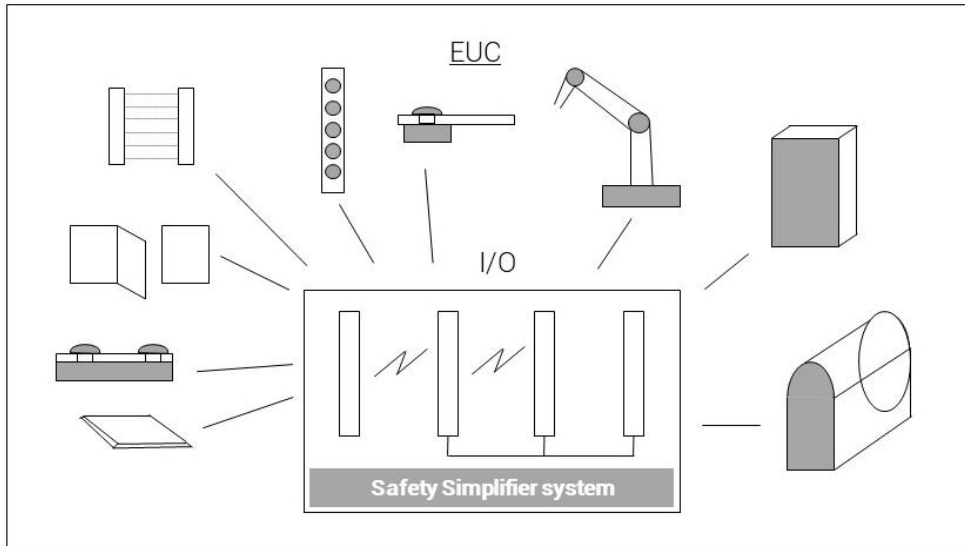


Figure 2.1 The Simplifier system as part of an EUC

The configuration software Simplifier Manager is used for:

- Programming and compiling logic that can be run on Safety Simplifier systems
- Configuring Safety Simplifier devices as systems
- Monitoring and debugging Safety Simplifier systems
- Generating documentation about Safety Simplifier system



Configuration of Safety Simplifier devices is performed either via USB using the micro USB connector inside the Safety Simplifier, or wirelessly using Simplifier Monitor.

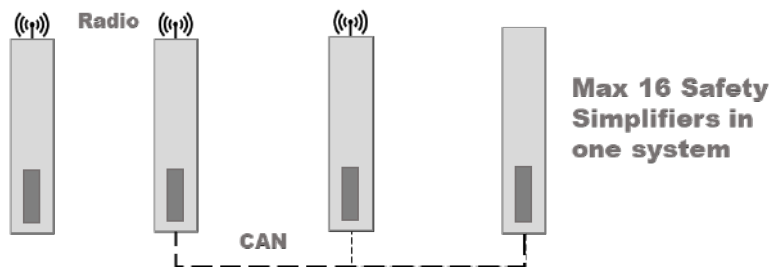


Figure 2.2 Example of a Simplifier system of four units, using a combination of CAN and radio to communicate.

## 2.1 Features

- Up to 14 transistor I/O configurable as inputs or outputs (or as a combination of both)
- Up to 2x2 doubled relay outputs
- An optional LED display for failure codes and status information
- 2.4 GHz radio and/or CAN-bus for safety communication between up to 16 units
- Spring terminals for I/O connections
- IP 65 housing
- Two openings on each short side for M12- contacts or cable glands M16
- Optional opening for PB: s, selectors, E-stop etc
- USB and/or radio communication for Simplifier Manager

### 2.1.1 Example input functions

- Emergency stop switches
- Two hand stations according to EN 574
- Safety gate limit switches
- Reset devices
- Light beam devices
- Light curtain devices
- Scanners
- Enabling switches
- Hold to run device
- Operating mode selector devices
- Safety mats and strips
- RFID-sensors
- NC contacts for supervising of relays controlled from outputs
- Eight coded signals (four regular, four inverted) from the same unit
- Two coded signals (one regular and one inverted) from other units
- Voltage window and inverted voltage window
- Specific voltage comparator

### 2.1.2 Example Output Functions

- 10 - 30 VDC outputs for machines
- 10 - 30 VDC OSSD outputs with short circuit detection between up to 14 outputs
- 10 - 30 VDC for 8 safety coded signal within one Safety Simplifier
- 10 - 30 VDC for 2 safety coded signals between Safety Simplifiers

### 2.1.3 Combined I/O

Combined I/O are I/O that can be used as safe inputs and non-safe outputs at the same time.

- Reset pushbuttons with lamp indication
- Start and stop push buttons with lamp indication
- Other devices or push buttons with lamp indication

## 2.1.4 Redundant Relay Outputs

Two optional potential free redundant relay outputs for safety.

The relay outputs can for example be used for:

- Safety inputs on lrb:s and other machines
- The control of door locking devices and other safety mechanisms

## 2.1.5 I/O outputs for “non-safety” information

All transistor I/O:s can be used for non-safety as for example:

- Serial information for up to 32 internal signals from logic. This can be used between two different Safety Simplifier systems or between Safety Simplifier units and other PLCs to exchange non-safety information without the need of a common application program.
- Indication and status information

All relay outputs can be used for non-safety as for example:

- Indication and status information

## 2.1.6 LED information

See chapter 11.9 The LED display menu on page 39 for a detailed description of how the LED display menu is operated.

The LED-information on a Safety Simplifier shows:

- Node number and number of nodes in a system
- I/O status on actual node and other nodes in a system
- Supply voltage level for actual node and other nodes in a system
- Global Memory status on actual node and other nodes in a system
- Radio channel and communication status on actual node and other nodes in a system
- Indication of nodes with CAN communication and CAN memory status and on actual node and other nodes in a system
- Error codes
- I/O errors



### 3 Safety Precautions when using Safety Simplifier



Read this information carefully as it is essential information on how to design and use the Safety Simplifier System in a safe way.

The Safety Simplifier system has three modes of operation:

1. Normal mode: The Safety Simplifier PLC controls the outputs according to inputs and logic.
2. Safe state: a state during which outputs are in safe state.
3. PLC Configuration mode. The Safety Simplifier is off-line, and all outputs are in safe state.

This is the only mode where a new configuration can be downloaded from a PC.

Safe state is defined as a state when outputs are turned off (goes low = 0V) and the output relays are deenergised (all contacts open).

In a system of more than one Safety Simplifier, if a node loses communication from another node, it will consider all the safety information from that node as 0 (off, open, deenergised). It is important to design the system so that loss of communication results in the relevant outputs turning off (0V).

If an internal dangerous failure is detected the affected Safety Simplifier unit goes to safe state. The safety communication via radio and CAN-bus is turned off. The other nodes in the system will lose connection to the affected Safety Simplifier unit and all safety information from that unit are set to 0. Information on the cause of the dangerous failure is available on the LED-panel and via USB on the actual unit. Information is not available wirelessly via radio or via CAN as this communication is shut off.

If an external dangerous failure is detected (such as a short circuit on an OSSD output or an input that detects an invalid coded signal), the affected inputs and/or outputs turn off. The error information is available on the LED-panel, wirelessly, via CAN and USB, and as an internal signal in logic. If more actions are required, the logic can be programmed to set the device in safe state, turn off other functions, or perform other actions, in case an input or output function detects an external error. This choice must be taken by the system integrator.

The system is designed for applications where; 0V, open circuit, logic "0", low signal, loss of radio signal, loss of bus communication etc generates a safe state (stop/off). The application must be designed according to the "de-energisation" principle, meaning that stop functions shall operate by de-energizing as well as with safe state conditions.

Safe state = "0". A fault in the system can set inputs, outputs, memories etc. to logic "0" which is regarded as a safe state. Logic "1" must therefore normally not be used to generate a safe state (stop/off). An exception is a dual or multiple channel function with logic "1" combined with logic "0". It is essential that loss of radio communication and CAN-bus communication lead to safe state. This must be the principle when programming a system with two or more Safety Simplifiers communicating between each other through radio or CAN-bus.

For control devices which starts a function when they are actuated it is important to use the start-up function for the inputs connected to the devices and to use the start-up function for global and CAN memories used between Safety Simplifiers. The start-up function requires a control device to be released and activated again after loss of power, loss of communication and loss of one or more input signals.

In a safety system using two or more Safety Simplifiers it is important that it is planned how to handle a loss and return of communication. If a Safety Simplifier loses communication signals only those functions depending on the signals/memories in the communication will react. All other functions will continue to work. When the communication comes back after a loss it is up to the programmer of the system to decide the actions. The programmer can for example require every function to stop if one or more Safety Simplifiers loses communication. The programmer can also select to require a restart after return of communication. This must be decided by the risk analysis for the system.

The Safety Simplifier System offers a set of function blocks intended for different safety functions such as two-hand, monitoring of dual channel input, dual OSSD outputs etc. It is strongly recommended that these function blocks, which are checked and certified, are being used. See the software part of this manual for more information.

All logic for safety shall be designed in such a way that no hazardous conditions are caused if a unit or outputs goes to safe state.

The Safety Simplifier unit/s shall be restarted at least once per year to fulfil the safety requirements. If the application where it is used requires restarts more often than that, that requirement shall be fulfilled.

All programming, logics and connections for safety shall be done in accordance to the manuals for Safety Simplifier and to the actual safety requirements and legislation for the EUC.

To identify a Safety Simplifier each one has a unique serial number (32 bits) for the identification. The software for a system of 2 – 16 Safety Simplifier also has a network id based on all the Safety Simplifiers serial number in a system to have a safe communication.

Each Safety Simplifier must have the same firmware to communicate (se Safety Simplifier Manual).

Every application program for Safety Simplifier has a unique code to verify the right program. Any change in a program will change the code.

Since Safety Simplifier is a system for the control of safety functions it is vital that personnel involved in design, programming and maintenance have sufficient knowledge about the system and knowledge in the field of machinery safety for the actual machinery. It is also important that the safety system is carefully tested and documented.

Reprogramming of an existing program can be necessary, and this can be carried out a long time after the original programming was done. It is then important that the programmer is familiar with the system, the hardware application, the program code and is sure about the intention with the revision. It is also important that the modification is carefully tested and documented.

Download of application programs shall be password protected and this can be set in Simplifier Manager. The intention is that the password is kept in secret by a responsible person who gives permission for revision of programs. If the password gets commonly known, it should be changed.

The most important part before a machine or other safety application is taken in use is to verify correct behaviour according to the specification and the risk analysis by testing all the safety functions in the safety system. Since many design faults are difficult to find only by a practical test it is also necessary to make a review of drawings and PLC program. It is recommended that tests and verifications also are done by person/s other than the designer. This is one way to prevent faults in design and systematic failures.

The safety Simplifier shall only be used in an environment which is in line with the environment conditions stated in the technical data.

The Safety Simplifier has I/O and logic signals for safety and for non-safety.

The safety functions are monitored and controlled by both processors.

The non-safety is optimised for non-safety functions and flexibility. These shall not be used for safety as they are not controlled and monitored by both processors. Examples of this is the information communication between Safety Simplifiers, via the Simplifier Monitor to a computer as well as the information on the LED-display.

In the programming software Simplifier Manager, all non-safety signals are indicated by a dotted white line, while safety signals are coloured solid yellow. Non-safety function blocks are coloured grey.

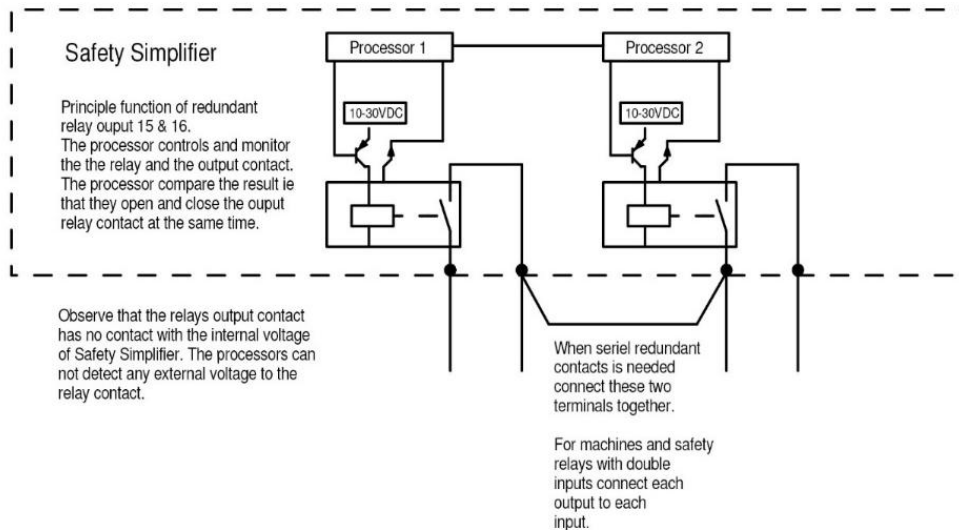
The safety functions in a Safety Simplifier are for example:

- Up to 2x2 double relay outputs
- Up to 14 I/O selectable as input, output or a combination
- Detection of voltage level from 0 – 30 VDC
- Pulsed output signals
- Distinguishing between different pulsed input signals
- Short circuit detection on inputs and outputs
- Detection of ON and OFF conditions for inputs and outputs
- Up to 14 OSSD outputs with short circuit detection
- 16 Global memories for radio and CAN-buss communication
- 16 extra CAN-memories on CAN-buss

## 3.1 Safety information regarding I/O and relay outputs of Safety Simplifier

### 3.1.1 The relay outputs

Two optional, potential free, doubled safety relay outputs can be used to control safety functions up to SIL 3/PLe/cat 4.



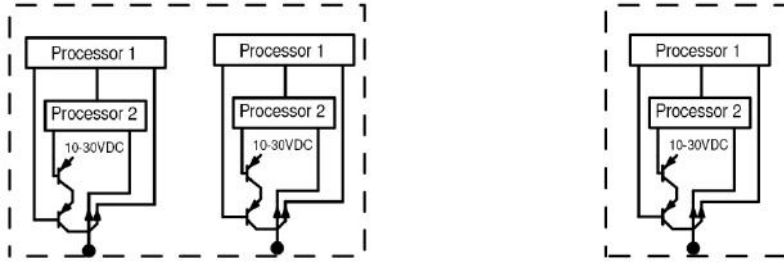
A relay output consists of two separate relays each having one NO contact connected to terminals. Each relay is controlled and supervised by a processor. The processors compare the result with each other. The processors check that the contact in each relay is opened every time a relay is deenergised. The output 15 (terminal 23-24 and 33-34) control a safety function up to SIL 3, PLe cat 4. Output 16 (terminal 43-44 and 53-54) has the same function.



**Caution!**

Both output contacts from 15 respectively from 16 must be used in a safety function to achieve a redundant safety output of category 3 or 4. A relay output cannot check if there is a short circuit over a contact in for example a cable. Short circuit detection needs to be performed at the device connected to the relay output contacts. For relays the number of operations and braking capacity have an impact on the PFH-d value. For low number of operations and braking capacity have an impact on the PFH-d value. For low number of operations as for E-stop a higher current can be switched by the relays during its lifetime. For high numbers of operations, the switched current should be lower. See chapter 4 Technical data Safety Simplifier (page 23) and chapter 9 Calculating PFH-d values (page 32).

### 3.1.2 Solid-state transistor outputs



Up to SIL 3, Ple, Cat 4 redundant output

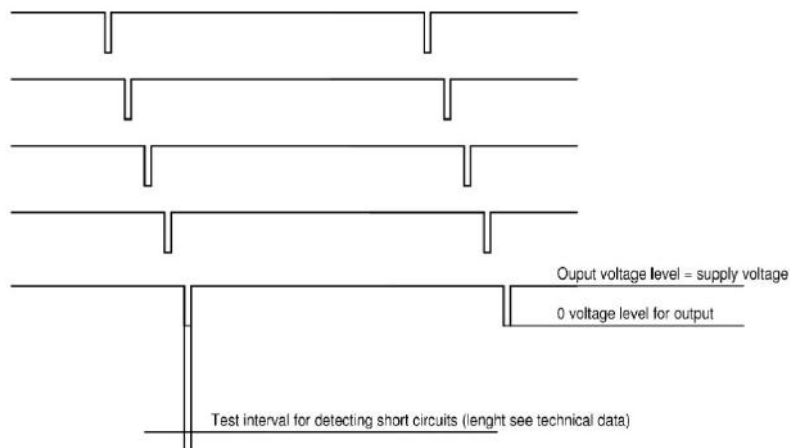
Up to SIL 3, Pld, Cat 4 redundant output

Each output has two transistors in series and each one is controlled by each processor. Each output is therefore a redundant output up to a certain degree, (see PFH-d values and a note below). Both processors are also checking the signal transmitted from the output. This way both CPU:s check the actual voltage level and coded signals sent out from the output. A single output which detects an external short circuit can turn of itself but not the external short circuit. We recommend double static outputs for SIL3, Ple, cat 4.

Any of the 14 I/O can be set in the Simplifier Manager to different output types in ON and OFF state. The selectable output types for an I/O are:

1. VDC/+V (maximum 0.5V less than supply voltage)
2. 0 VDC
3. OSSD = output voltage level is 0,5 V less than supply voltage and this type of output also detects a short circuit to other OSSD outputs in the same unit and to internal and external fixed voltages. If there is an external short circuit the relevant outputs will be shut off, set to OFF state or to fatal error depending on the type of short circuit and on how the application program is done. See the Safety Manager manual for more information. The check of the output is done during a very short time (less than 150 micro seconds) when the voltage level shall go down to 0V. The OSSD outputs are commonly used in the industry in safety devices and for stop signals to machines. Up to 14 OSSD output can be selected in a Safety Simplifier.

Drawing shows 5 OSSD outputs with short circuit detection. Up to 14 OSSD outputs can be selected



from a Safety Simplifier. A short circuit between any of these or to external voltage, is detected as an external fault. A short circuit to one or more outputs will set the output to 0 volt. Other outputs it the same function block will be set to OFF state. The fault will be indicated on the LED panel.

The test interval is very short (below 150 micro seconds). A conductor or a relay connected to an OSSD output will normally not fall during this time. See technical data for test interval specification.

A, B, C and D-pulses are 4 different unique pulse trains which can be generated. These can also be generated as inverted and they are then called Inv A etc. This way up to 8 different pulse trains can be selected.

These signals are used to detect that the same or inverted signal comes back on an input on the same Safety Simplifier. This way it is easy to connect cables via safety devices and back to the Safety Simplifier to check that the function is right and that there are no short circuits between the cables when they are coming back to the inputs. If there is a short circuit between any of these different signals or to fixed voltage it can be detected on the outputs and the inputs. If there is an external short circuit the relevant outputs can be shut off, set to OFF state or to fatal error depending on the type of short circuit and on how the application program is done.

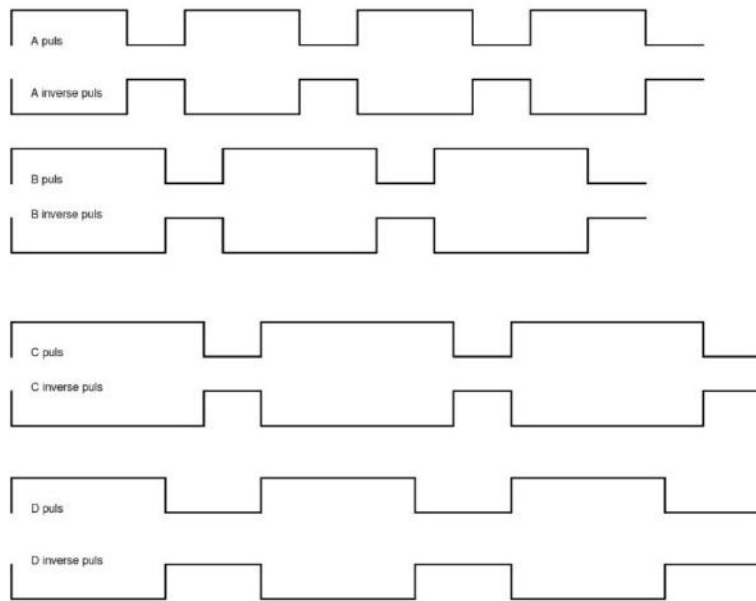


Figure 3.1: Pulse trains for A, B, C, and D pulses

See the Safety Manager manual for more information.



**Note**

If there is a short circuit to an external voltage to an I/O this cannot be disconnected by Safety Simplifier. Safety Simplifier can only disconnect other outputs associated with the programmed function of the detected short circuit I/O or be programmed to go to fatal error.

If an output is neither in ON or OFF state it is in ERROR state. This is the case if an external fault is detected except for detected short circuits to 0V which leads to fatal error or that the external fuse shuts off. The default action in case of a detected external fault is to go to Error state and to set the outputs with a detected failure to 0V and the other output/outputs in a group to OFF state. The LED will go to orange for the outputs which are in error state and the one/s with fault will flash twice as often. If error is gone on the outputs the cause for stop will still flash with same speeds until a restart of outputs is done. Additional actions in case of external fault can be programmed by the programmer of the safety system using Simplifier Manager. The other actions can for example be to go to fatal error or any other programmed functions.

### 3.1.3 Safety inputs

For details on programming safety inputs see chapter 13.11 Input Functions (page 89).

Each input is a voltage input and both CPU: s read the voltage level. Voltages above 80% of power supply voltage are considered a logical 1, and voltages below 20% of power supply voltage are considered a logical 0. Some inputs can also be set to a specific voltage level (specified in V).

When configuring inputs, an ON-state and an OFF-state signal type must be specified. The allowed states you can check in the matrix for recommended ON and OFF selections (see chapter 13.11.2 Input signal types on page 91).

ON and OFF states can be:

1. 0V
2. VDC (= Supply voltage)
3. Any of the A, B, C and D pulse trains or inverted within the same Safety Simplifier
4. E or inverted E-signal between Safety Simplifiers.

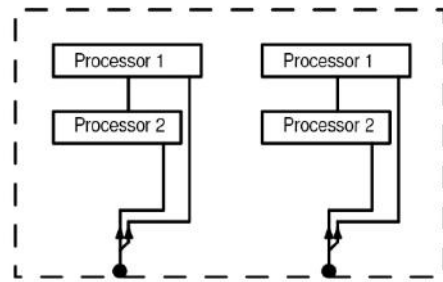


Figure 3.2: SIL 3, PLe, CAT 4 redundant input

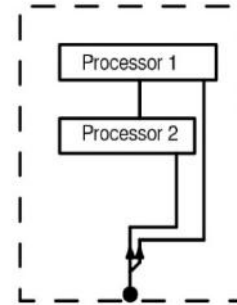
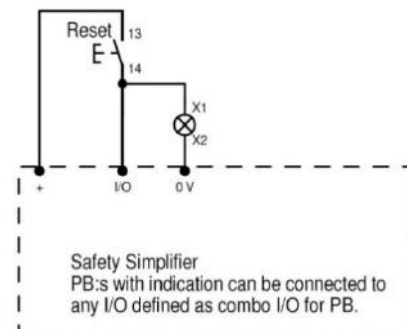


Figure 3.3: SIL 3, PLe, Cat 4 redundant input



Using a single input with 0V as the ON state signal (normally closed) as a safety input violates the de-energisation principle, as a disconnected cable will result in the input going to ON state. Always combine this type of input with a normally open input to make sure a disconnected cable does not lead to ON state.

For ON and OFF state, we also recommend active signals (coded pulsed signals) in both ON and OFF state (see matrix for recommended combinations). Active signals give the possibility to detect short circuits from external voltage.



## 4 Technical data Safety Simplifier

Recommended fuse	Fast Acting Fuse 2,5 A
Power supply	10 - 30 VDC
Current consumption except load on outputs	90 mA inactive mode, LED and relays not activated 110 mA active mode with LED, relays not activated 210 mA active mode, LED, I/O and 4 relays activated
Maximum output current transistor output	600 mA
Voltage drop when output 600 mA	0.5 V below supply voltage
Maximum total output current transistors	2000 mA
I/O	Selectable as Input, Output or as a combined I/O
I/O as Input	10 – 30 VDC (for PNP sensors)
I/O as Output transistor	10-30 VDC
I/O as OSSD output with self-test during	<150µs
I/O as combined Input and Output for reset	10-30 VDC
Relay contacts (S16 only)	Gold plated AgSnO <sub>2</sub>
DC13	2A
Relay contacts (inductive/resistive load)	Max 2A (1260 op/year for 10 years) Max 49 VDC
Relay contacts (inductive/resistive load)	Max 0,5 A (3780 op/year for 10 years) Max 49 VDC
Relay contacts (inductive/resistive load)	Max 0,2 A (100 800 op/year for 10 years) Max 49 VDC
Operating temperature	-20 °C to + 65 °C
Operating altitude	Up to 2000 m
Storage temperature	-30 °C to + 70 °C
Relative operating humidity	10 - 90 %
Terminals (spring)	32 (max 0,5 mm <sup>2</sup> or 24 AWG)
Vibration in accordance with the standard	EN 60068-2-6
Frequency	5-150 Hz
Acceleration	1 g
Shock stress in accordance with the standard	EN 60068-2-27
Acceleration	15 g
Duration	11ms
Relative operating humidity	10 - 90 %
Mounting surface size	253 x 40 x 44 mm
Maximum size housing	253 x 42 x 44 mm
Weight S14RBLD/S16RBLD without holes/plugs	About 235/265 g
Enclosure material	PC + ABS, UL-certified material, ANSI/UL 94
Degree of protection, IEC 60529	IP 65
Ventilation enclosure	GORE-TEX membrane
Maximum altitude	1000 m
USB contact	micro USB with 180-degree DIP package
Lifetime	Calculations for PFH-d 61508 up to 10 years Calculations for PFH-d 13849-1 up to 20 years
Radio frequency channel number 1	2405 MHz
Radio frequency channel number 2	2410 MHz



Radio frequency channel number 3	2415 MHz
Radio frequency channel number 4	2420 MHz
Radio frequency channel number 5	2425 MHz
Radio frequency channel number 6	2430 MHz
Radio frequency channel number 7	2435 MHz
Radio frequency channel number 8	2440 MHz
Radio frequency channel number 9	2445 MHz
Radio frequency channel number 10	2450 MHz
Radio frequency channel number 11	2455 MHz
Radio frequency channel number 12	2460 MHz
Radio frequency channel number 13	2465 MHz
Radio frequency channel number 14	2470 MHz
Radio frequency channel number 15	2475 MHz
Radio frequency channel number 16	2480 MHz

Technical data CAN card for Safety Simplifier

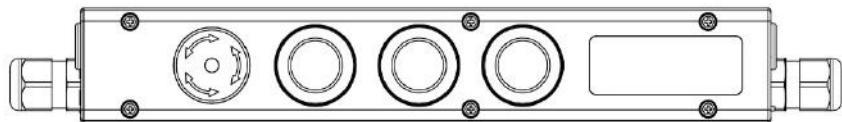
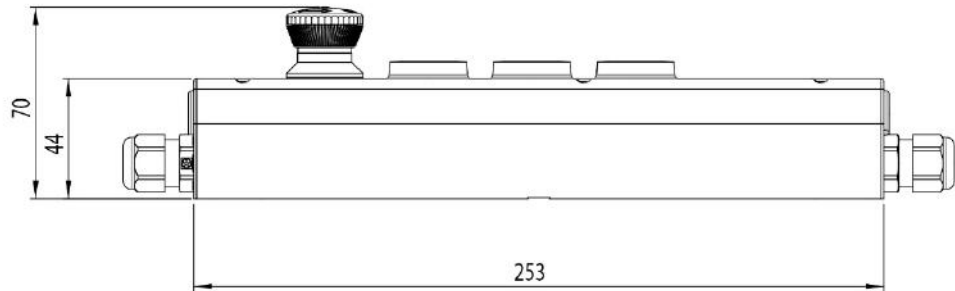
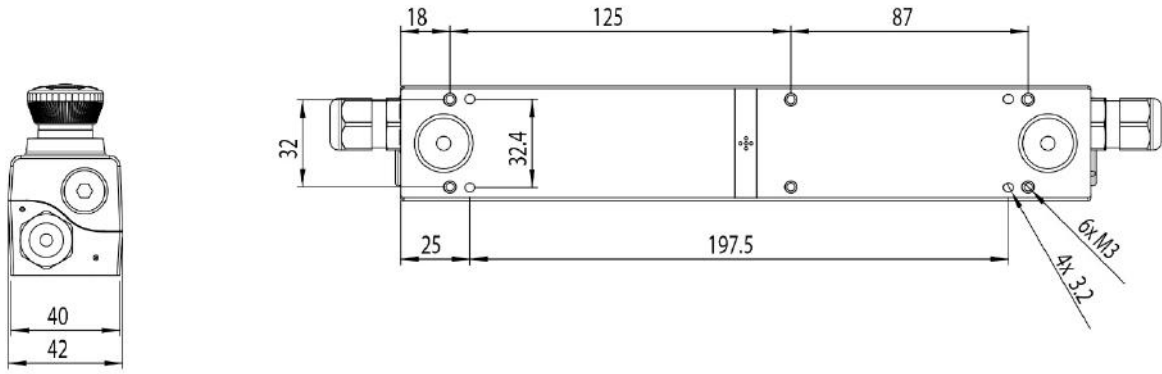
Power supply	10 – 30 VDC
Current consumption	0,02 A at 24 VDC
Terminals (spring)	7 (max 0.5 mm <sup>2</sup> )
Selector for resistor. Shall be set to ON at CAN-card for each end of CAN bus	Position ON = 120 Ohm between H & L Position 1 = no resistor

Data Rate	Trunk Distance	Stub length
		Units connected on a Stub must not have termination resistors fitted
125kpbs	250m	2.4m
250kpbs	160m	1.2m
500kpbs	70m	0.6m

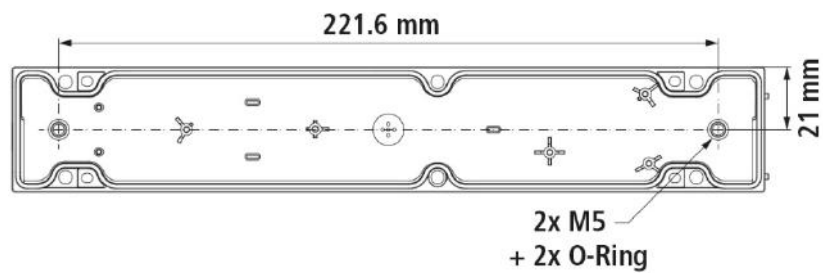
## 4.1 Safety parameters according to 610508

	Input	Logic	Single transistor output	Double transistor output	Relay output	
PFH	4.6E-09	2.12E-09	1.34E-09	7.76E-10	3.87E-09	1/h
$\lambda$ DD	5.97E+02	4.05E+02	1.89E+02	2.39E+02	5.03E+02	(10e-9)/h
DC	99%	99%	99%	99%	99%	
Diagnostic test interval	Continuous					
Lifetime	10					years
Proof test	10					years
MRT	8					hours
Type	B	B	B	B	B	
SFF	99.5%	99.5%	99.5%	99.5%	99.5%	
HFT	1	1	1	1	1	faults

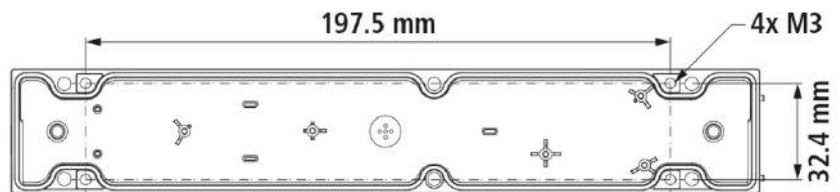
## 5 Dimensions and mounting





Bottom plastic part of Safety Simplifier showing two mounting holes for M5 screws with sealing and distances in between (see Table 5.1: Accessories for Safety Simplifier on page 28).



Bottom plastic part of Safety Simplifier showing four mounting holes for M3 screws (see Table 5.1: Accessories for Safety Simplifier on page 28).



## Mounting with plates

	<p>Bracket small</p>	<p>Aluminium 253 x 40 x 3 mm Incl 4 M3 screws for Safety Simplifier</p>	<p>Article No. SP-N-88-850-01</p>
	<p>Bracket large</p>	<p>Aluminium 338 x 40 x 3 mm Incl 6 M3 screws for Safety Simplifier</p>	<p>Article No. SP-N-88-850-02</p>

### Mounting with small bracket

The small bracket with the same length as the Safety Simplifier is mounted on a surface using two screws. The Safety Simplifier is then mounted on the bracket with the four included M3 screws (see Table 5.1: Accessories for mounting and installing Safety Simplifier on page 28).

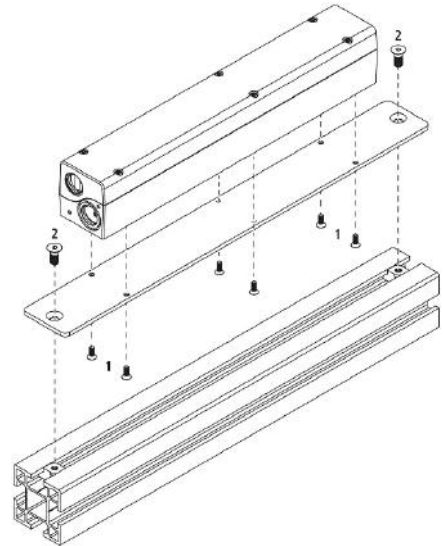
### Mounting with large bracket

The large bracket is mounted with the six included M3 screws (1), directly on the bottom side of Safety Simplifier. For mounting the bracket with Safety Simplifier there are two 5 mm holes for M5 screws (2).

### Mounting without plates

The Safety Simplifier has four holes for mounting the bottom part directly on a surface. The holes are intended for M3 screws.

In the middle of the Safety Simplifier box there are two prepared hole possibilities. Two holes can be made by pressing out the plastic parts with a screwdriver. To keep the sealing level when using these holes, the unit must be mounted with our M5 screws including sealing.



### Joining two Safety Simplifier enclosures together

Two or more Safety Simplifier enclosures can be joined together (see Figure 5.1) using the special connection screw (see Table 5.1: Accessories for mounting and installing Safety Simplifier on page 28). This is done by aligning the bottom part of one Safety Simplifier with the top of another and screwing the connection screw through the bottom Safety Simplifier into the top Safety Simplifier. Make sure the alignment pins of the bottom Simplifier are aligned with the corresponding holes in the top Simplifier.

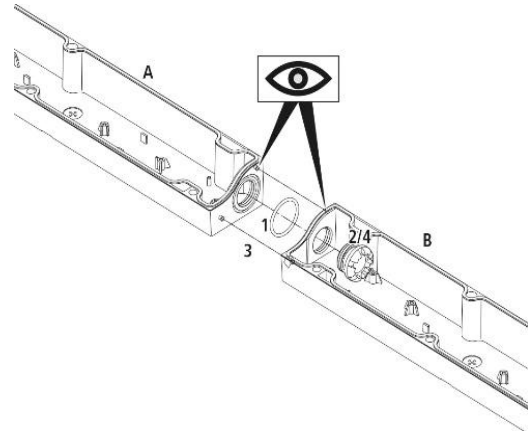


Figure 5.1: Joining two Safety Simplifier enclosures using the connection screw (see Table 5.1: Accessories for Safety Simplifier on page 28).

A: top Simplifier

B: bottom Simplifier

1: O-ring sealing

2: Connection screw

3: Alignment pins



**Note** Do not forget the O-ring sealing when joining two Safety Simplifier together with the connection screw, as shown in Figure 5.1. Make sure the alignment pins of the bottom Simplifier are aligned with the corresponding holes in the top Simplifier.




	Screw with sealing	M5 with O-ring for mounting of Safety Simplifier	SP-N-88-001-89
	M16 Plug (yellow)	Plug with O-ring for mounting from outside Plug for inside mounting on top part when two boxes are connected together	SP-N-88-001-73 SP-N-88-001-60
	Connection screw with sealing	Plastic part with threads M16 and O-ring	SP-N-88-001-90

Table 5.1: Accessories for mounting and installing Safety Simplifier

## 6 Safety Simplifier principle layout

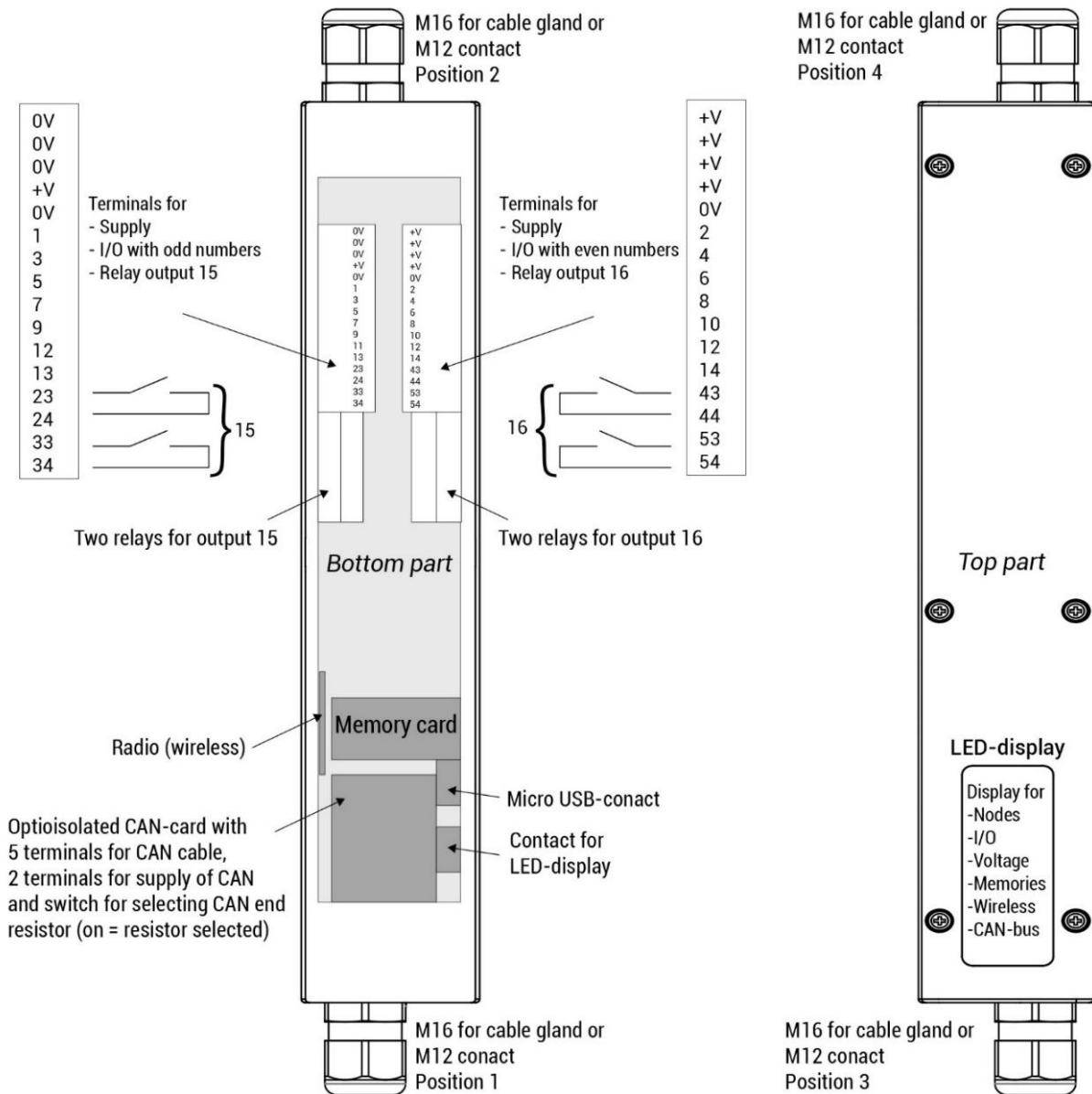



Figure 5.1

The internal and external layout of Safety Simplifier

### Conductor cross section

- single	0.08...0.5 mm <sup>2</sup>
- fine wire	0.08...0.5 mm <sup>2</sup>
- fine wired (wire sleeve with plastic collar)	0.08...0.5 mm <sup>2</sup>
- fine wired (wire sleeve without plastic collar)	0.25 mm <sup>2</sup>
stripping length	5-6 mm <sup>2</sup>

## 7 Electrical Installation



**Warning!**

Make sure that supply voltage is always off or disconnected when installing Safety Simplifier and other external devices. Failure to do so risks unintentionally starting dangerous equipment which can result in serious injury, death, and damage to equipment.

Never use the Safety Simplifier in an installation before the installation is verified according to the specified function according to the risk analysis.

Make sure cables and connected devices to Safety Simplifier such as sensors, Pushbuttons, selectors etc are isolated for 240V. The Safety Simplifier is only intended for low voltage system 10 – 30 VDC. Voltage outside this range may cause unintended hazardous behaviour.

The Safety Simplifier has internal overcurrent protection but should always be protected by an external fuse (see technical data (6)).

The Safety Simplifier is intended to be used in both 12V and 24V DC systems, however any voltage in the range 10-30V can be used. The accepted voltage range can be limited by settings in the software, if external sensors and equipment require a more limited voltage range. If the voltage goes outside the specified range the unit will enter safe state and indicate a voltage error on the LED display.

As the Safety Simplifiers communicate with each other using radio or optical isolated CAN-bus, each one can be set to different voltage levels if needed.

The terminals in the Safety simplifier are spring terminals which can stand high vibrations. For supply voltage there are several terminals to be used for both in and output supply.

The Safety Simplifier is intended for applications which fulfil IEC-EN 60204-1:

- Transformers shall be used for supplying the control circuits
- For electrical safety reasons and to be able to detect safety critical earth faults in single channel circuits, the 0V terminal must be connected to protective bonding circuit (see EN 60 204-1, 9.4.3.1 method a)

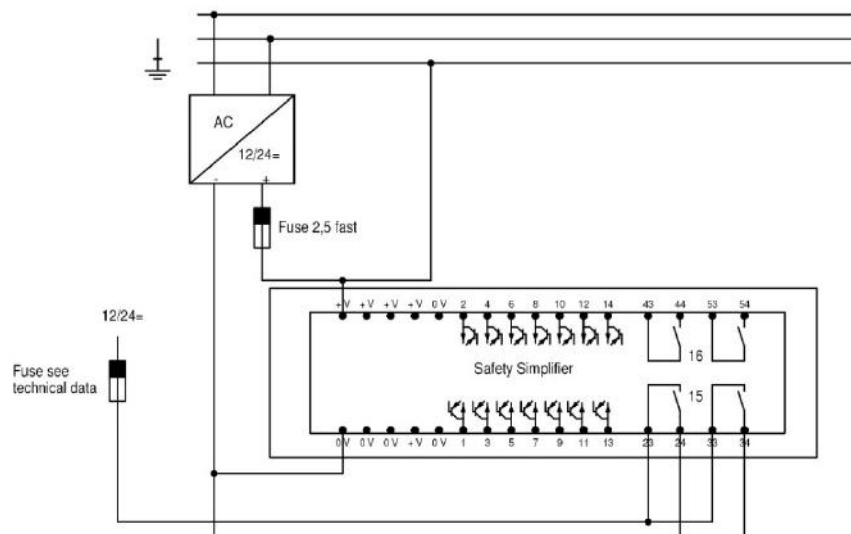


Figure showing fusing and power supply to Safety Simplifier and external relays/contactors

## 7.1 Making IO Connections

The IO clamping terminals on Safety Simplifier can be opened and closed by using a small flathead screwdriver (see Figure 7.1).

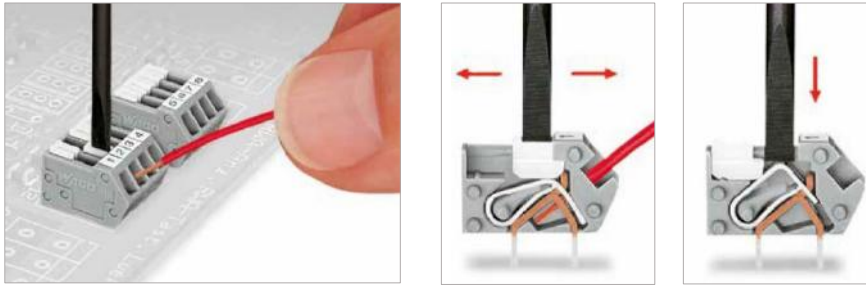


Figure 7.1: Opening and closing an IO terminal



**Note**

Only use stranded wires when making connections to the terminals for the best connection quality.



## 8 Bus communication via radio and/or CAN-bus

Both S14 and S16 can have a radio card and a CAN card for bus communication. A system of up to 16 Safety Simplifiers can use 16 Global Memories (GM) from any other Safety Simplifier for safety functions as well as for information reasons. If there is a radio card and a CAN card in a Safety Simplifier both as a default setting are sending the status of the GM:s. A safety Simplifier will use the status from the fastest bus. The wireless system will repeat the GM:s between each other. A CAN bus only sends the GM: s to the ones connected to the same CAN bus cable. A CAN bus have however 16 extra CAN memories which are sent on the CAN bus only. See the programming manual to see how to use the GM: s and CAN memories for safety functions and for non-safety functions.

The wireless card needs to be ordered from beginning for a Safety Simplifier. A CAN card can be mounted anytime. It is easy to connect a CAN card into a Safety Simplifier and it is fixed by pushing it into place and secured with a screw.

The terminals for the voltage supply on the CAN card is named +V and -V. These are used when the CAN is supplied from the terminals in a safety Simplifier. This supply is then connected to terminal 2+ and 3 -which is to be connected to the CAN-bus for other units. It is important that the supply is only connected to units with the same ground. Normally only one connection is needed. 1S is intended for the shield in a CAN bus cable if needed. 4H and 5L is for the CAN-bus signals which are optocoupled to the CPU CAN bus connection. The numbers from 1 to 5 is in line with the standard for M12 contacts to the CAN bus. The CAN card also has a switch for connecting a resistor 120 Ohm which is needed on both ends of a CAN bus cable. Up to 16 units can be connected to each other on CAN bus

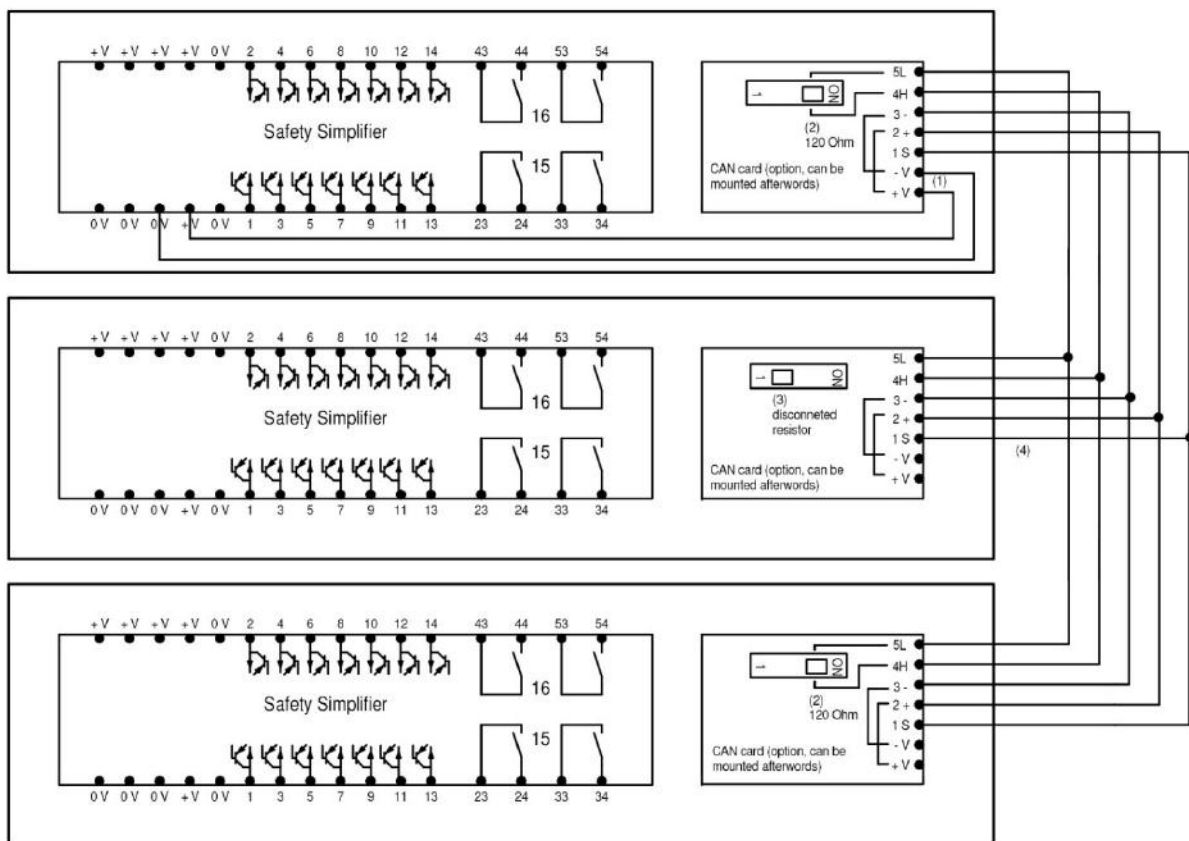


Figure 8.1: Wiring diagram for CAN-bus connections between Safety Simplifier

## 9 Calculating PFH-d values

PFH-d value for Safety Simplifiers:

- Single conductor input PFH-d:  $4.60 \times 10^{-9}$

- Double conductor input PFH-d:  $4.60 \times 10^{-9}$
- Logic: PFH-d:  $2.39 \times 10^{-9}$  (including power supply)
- Single conductor transistor output PFH-d:  $1.78 \times 10^{-9}$

Double conductor transistor output PFH-d:  $0.776 \times 10^{-9}$

- Double conductor output PFH-d:  $3.87 \times 10^{-9}$  : 1260 operations/year at max switched current 2A
- Double conductor output PFH-d:  $3.87 \times 10^{-9}$  : 3780 operations/year at max switched current 0.5A
- Double conductor output PFH-d:  $1.09 \times 10^{-9}$  : 100 800 operations/year at max switched current 0.2A

Formula to count the PFH-d value for each safety channel:

- Single Safety Simplifier: PFH-d = Input + logic + output

Multiple Safety Simplifiers with radio communication with repeating information:

- PFH-d = Input + n x logic + output (n = numbers of Safety Simplifiers in the system)

Multiple Safety Simplifiers with radio communication without repeating information:

- PFH-d = Input + 2 x logic + output

Multiple Safety Simplifiers using CAN-bus for safety communication:

- PFH-d = Input + 2 x logic + output

Multiple safety Simplifiers connected to each other via inputs and outputs

- PFH-d = (Input + logic + output) x m (m = numbers of units from input to output)

Life time used in above calculations based on EN61508: 10 years

Life time according to table in EN13849-1 for PFH-d: 20 years

## 10 Calculating reaction time

For inputs, there are 4 types:		For logic	For output:
VDC/OSSD	T(I)=1ms	T(L)=2ms	Transistor: T(O) = 1ms Relay: T(O) = 11ms
A-pulse	T(I)=10ms		
B-pulse	T(I)=12ms		
C/D-pulse	T(I)=14ms		
E-pulse	T(I)=20ms		
Window	T(I)=1ms		

Selected Delay off for input: T(DI)

Selected Delay off in logic: T(DL)

Selected Delay off for output: T(DO)

Response time(ms) = T(I)+T(DI)+T(L)+T(DL)+T(O)+T(DO)

Two or more Safety Simplifiers

For two to 16 Safety Simplifiers you need to add the communication time, T(C), which is selected in the Simplifier Manager. You can select between long or short response time for the used memories.

Example

An input in one Safety Simplifier is used via a global memory (GM) through radio/CAN for an output in one or more other safety Simplifiers.

Formula for response time =

Safety Simplifier nod 1 + Radio and/or CAN + Safety Simplifier nod 2-16 (using GM in 1)

(Input, delays, logic) + (Communication response time: short or long) + (Logic, delays and Output)

$T(I) + T(D \text{ OFF } I) + T(L) + T(D \text{ OFF } L) + T(C \text{ long or short}) + T(L) + T(D \text{ OFF } L) + T(O) + T(D \text{ OFF } O)$



**Note**

It is strongly recommended always to go directly from one unit to another unit via a global memory, in this case a global memory in node 1 is used in any or all other nodes 2-16 for an output.

If a GM in nod 1 is used to activate a GM in node 2 and this GM2 is used in node 3 you will have to add the response time in logic 2 and an extra communication time (above + T(L)+ T(D OFF L) + T(C long or short).

General functional test requirements:

A functional test (automatic or manual) to detect failures shall be performed within the following test intervals:

a) at least every month for

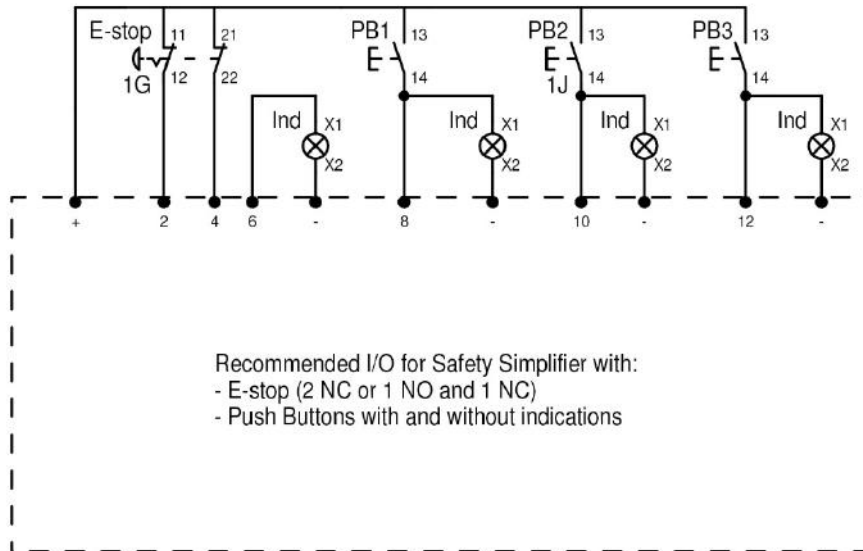
PL e with Category 3 or Category 4 (according to EN ISO 13849-1) or  
SIL 3 with HFT (hardware fault tolerance) = 1 (according to EN 62061);

b) at least every 12 months for

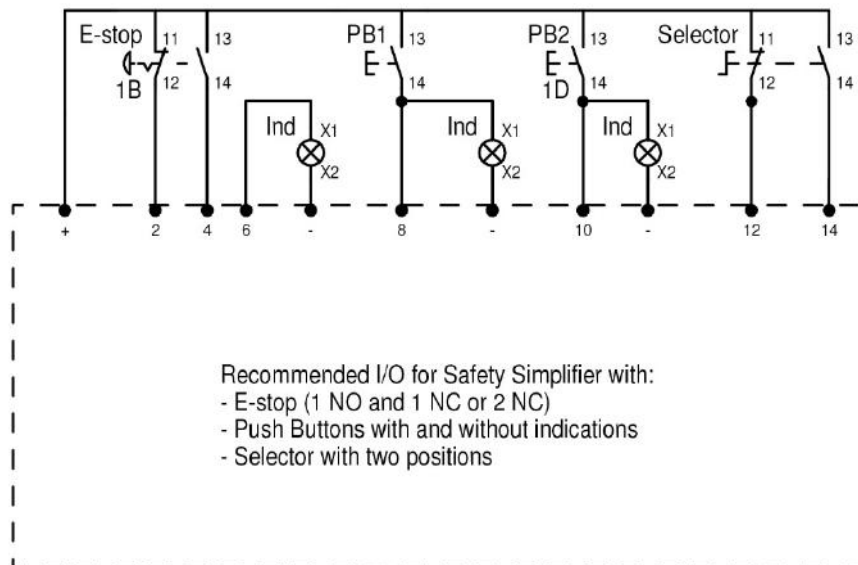
PL d with Category 3 (according to EN ISO 13849-1) or  
SIL 2 with HFT (hardware fault tolerance) = 1 (according to EN 62061).

## 11 Electrical diagrams for recommended connections of machines, PB:s and safety devices

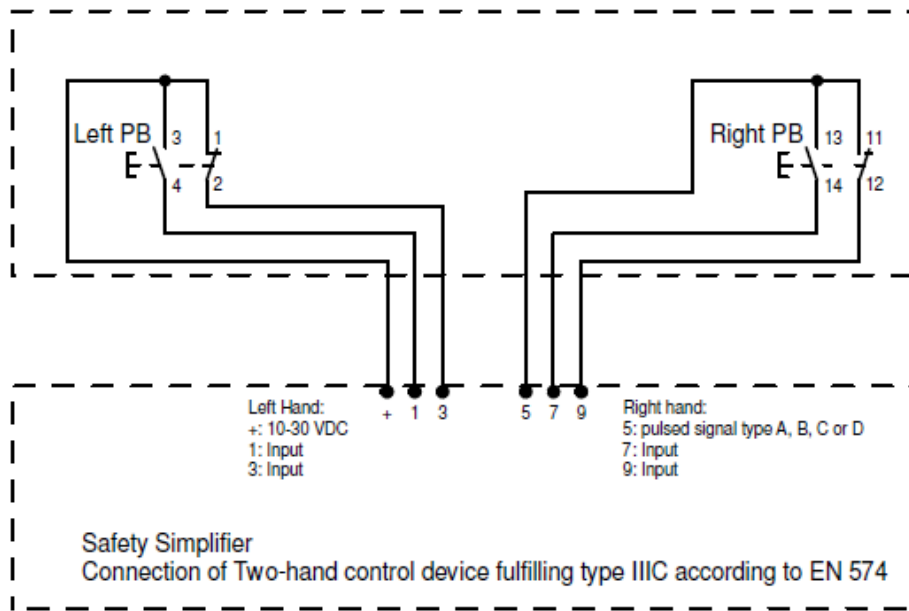
### 11.1 E-stop, Combo PB with indication could be used for reset, open, start, stop etc



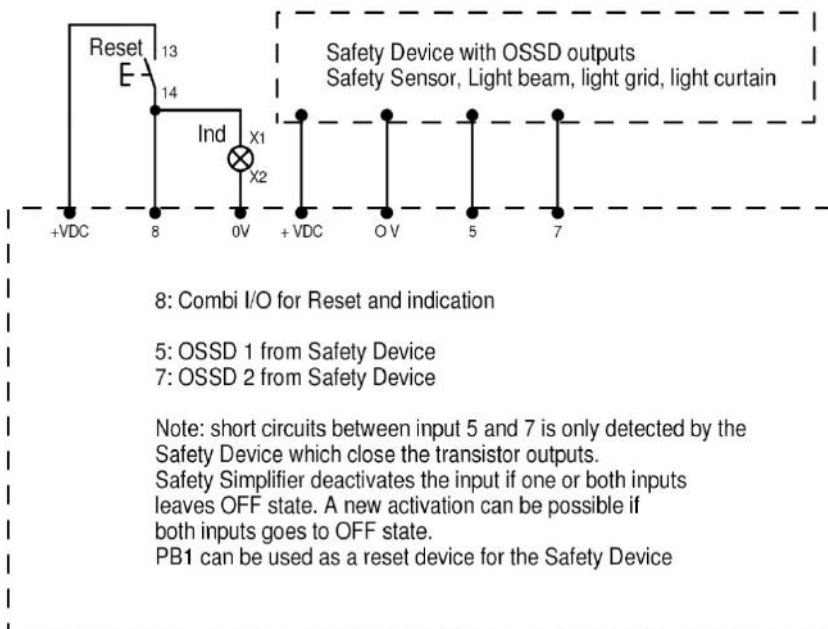
### 11.2 E-stop, Combo PB with indication for Reset, Open, Start and Stop, Selectors



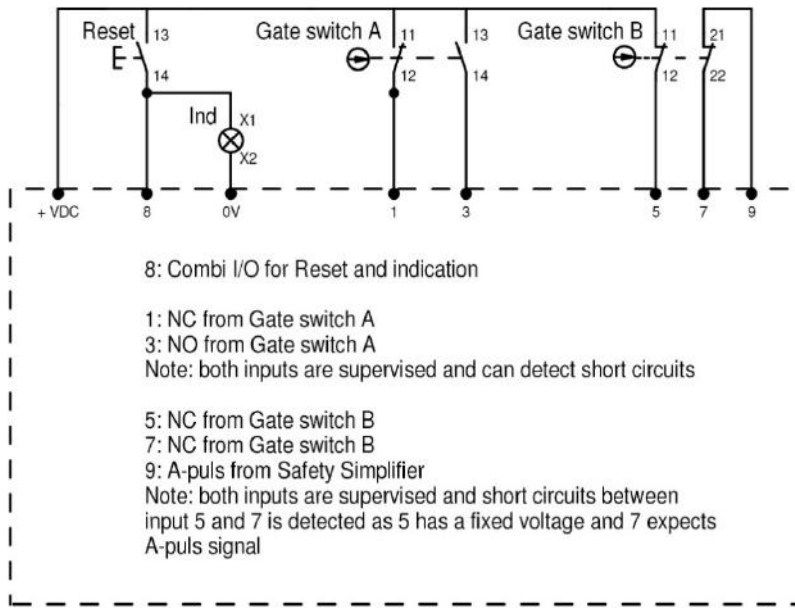
### 11.3 Two-hand connection according to EN 574



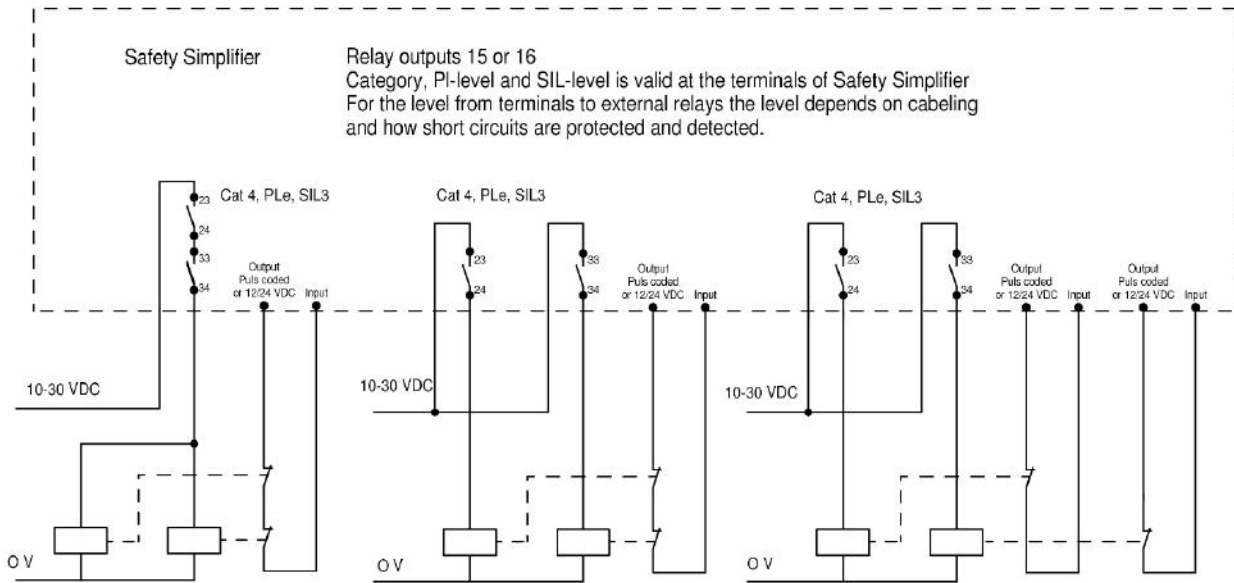
### 11.4 Safety devices with OSSD outputs such as: RFID-sensor/Safety Gate sensor, Scanners, Light beams and light curtains



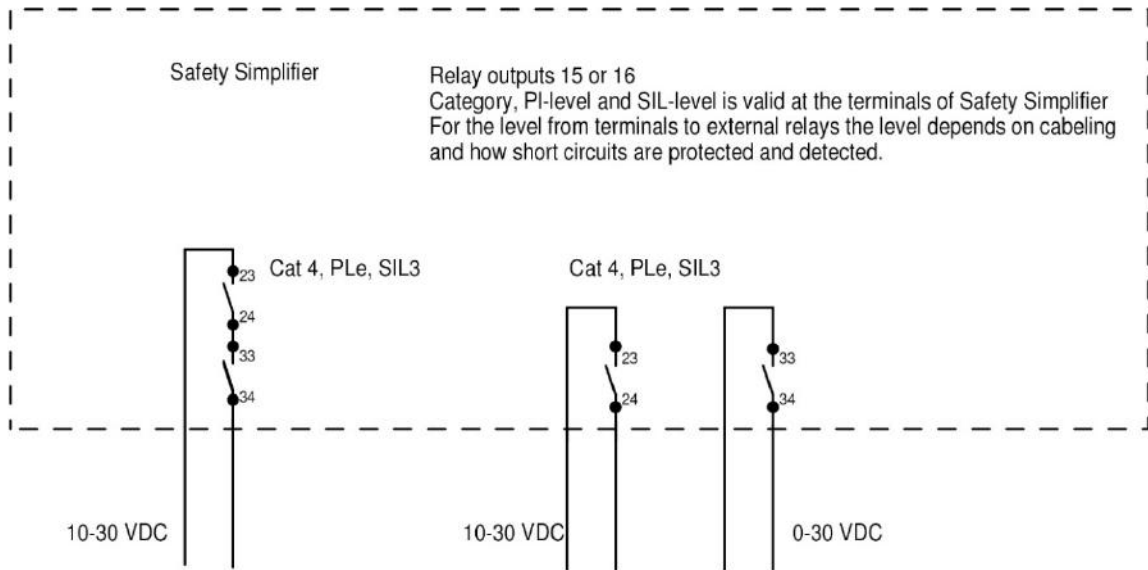
## 11.5 Safety Gate limit switches



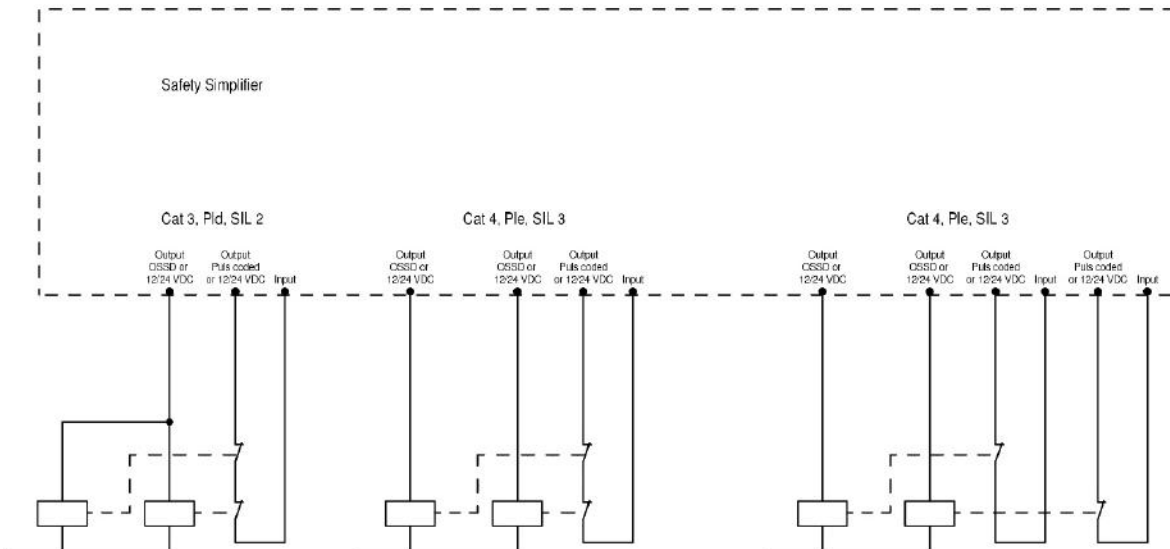
## 11.6 Relay outputs for contactors with feedback for supervision/monitoring



## 11.7 Relay outputs for machines and safety devices



## 11.8 OSSD outputs for machines, contactors and safety devices



## 11.9 The LED display menu

The front panel LED display menu displays useful information about the node and the system it is in. The different parts of the display can be seen in Figure 11.1:

1. The capacitive push button. This button can be used to navigate between the different menus of the display.
2. Here the currently selected menu is displayed (the meaning of each menu is described below).
3. Here different information is displayed, depending on which menu is selected.

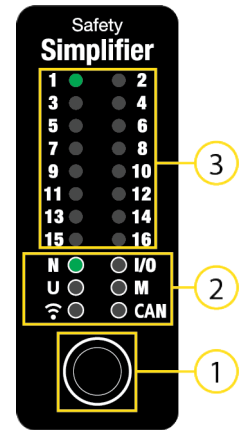


Figure 11.1

*The different parts of the LED display menu*

### 11.9.1 Navigation

To switch between the different menus the capacitive push button can be used. A firm short press (<1s) on the button cycles through the menus in the following order:

N ► I/O ► U ► M ►  ► CAN

Figure 11.2 shows the order of the menus on the display itself.

Each press of the capacitive button moves the menu to the next menu. Pressing the button in the CAN menu cycles the menu back to the N menu (this means that to go from the I/O menu to the N menu, the button must be pressed 5 times to cycle through all the menus back to N).

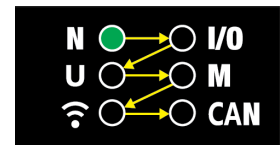


Figure 11.2

*The order the display menus are shown in.*



## 11.9.2 Sub-menus

Each menu also has one or more so called sub-menus. These can be reached by pressing and holding the button for 1 second. When jumping to a sub-menu, the LED indicating the menu will change colour from green (first level), to red (second level), to orange (third level). To go back to the first level menu, hold the button again until all sub-menus have been cycled through, and the LED turns green again.

In Figure 11.3, the first sub menu of the N menu is shown. This menu displays the nodes in the system that this node has either CAN or radio connection to. A green LED indicates connection, and a red LED indicates no connection. The unit has connection to node 1 (which is itself) and node 2, but not to node 3.

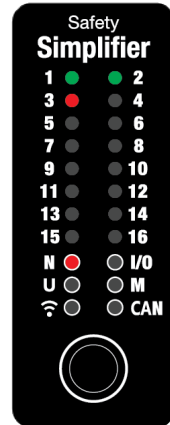


Figure 11.3

The first sub-menu  
of the N menu.




**Note**

The **U** and **CAN** menus have two sub-menus (three levels), while all other menus only have one sub-menu (two levels).

When in a sub-menu, the first level menu **cannot** be changed by a short press on the button. To cycle between the first level menus (i.e. from N, to I/O, to U, etc) the menu must be in the first level (i.e. the menu LED must be green). Short presses in a sub-menu changes the information displayed in that sub menu, for example in some menus to switch between looking at information from different nodes.

### 11.9.3 The menus

The following chapters describe what type of information each menu and its submenus display, and how the display can look in different menus.

 <b>Note</b>	<p>A long press of the capacitive push button is used to go to different levels of the menu.</p> <p>The display illustrations in the following chapters are examples, and will differ depending on the configuration, installation, and state of the system.</p>
--	--

#### 11.9.3.1 N (first level)

Displays the node number of the current unit (see Figure 11.4). This is the default menu when the unit starts up. The display will also jump back to this menu after not being touched for a couple of minutes.

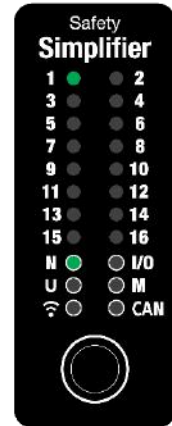



Figure 11.4

#### 11.9.3.2 N (second level)

Displays the connection to the other nodes in the system (note that the **N** LED is now red, indicating that the menu is in the first sub-menu of the **N** menu). If this unit has radio **or** CAN connection to another unit, it is displayed in green. If it does not have either radio or CAN connection, it is red.

In Figure 11.5, the unit has connection with node 1 (itself) and node 2 (via either radio or CAN). It has no connection to node 3. Having this type of connection means that the global memories between these nodes will always be heard. To see if the unit has connection via radio or CAN specifically, see the  and CAN menu descriptions further down in this chapter.

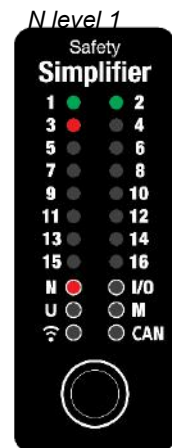


Figure 11.5

*N level 2*

### 11.9.3.3 I/O (first level)

This menu displays the state of the IO on the unit. An IO that is in the OFF state is shown in red, an IO that is in the ON state is shown in green, and the IO that are not programmed are turned off (black). In Figure 11.6, IO 2 and 4 are in the ON state, IO 6 and 8 are in the OFF state, relay 15 is in the OFF state, and relay 16 is in the ON state.

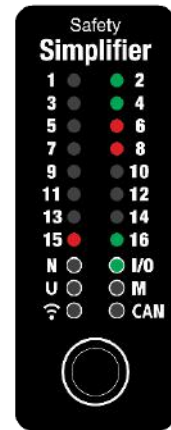


Figure 11.6

I/O level 1

External errors on inputs and outputs are indicated on the display by lighting the number corresponding to the terminal in orange. When an external error is detected the display will automatically jump to the I/O menu until the error has disappeared and been reset. If the error is currently present (i.e. the short circuit is still present, and the terminal is still detecting an external voltage), the LED is constant orange. When the error has disappeared and has not been reset in logic, the terminal LEDs blink orange.

Terminals that are in the same input/output function group as another terminal with an external error, are indicated by a short, fast, orange blink. This means that the terminal itself does not have an error but is in an IO function that has an error.

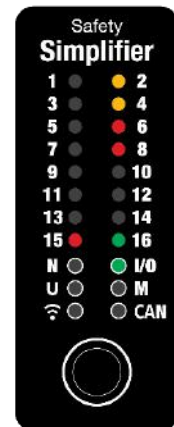


Figure 11.7

Terminal 2 and 4 are detecting external errors

### 11.9.3.4 I/O (second level)

This menu displays the state of the IO (both inputs and outputs) on the unit. An IO that is in the OFF state is shown in red, an IO that is in the ON state is shown in green, and the IO that are not programmed are turned off (black). In Figure 11.8, IO 2 and 4 are in the ON state, IO 6 and 8 are in the OFF state, relay 15 is in the OFF state, and relay 16 is in the ON state.

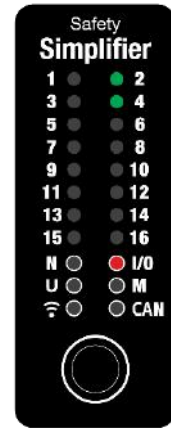


Figure 11.8

I/O level 2

### 11.9.3.5 U (first level)

Shows the voltage of the unit. The voltage is the sum of the LED numbers (for example on Figure 11.9, the voltage is 8+16=24V). When the voltage is less than or equal to 16, only one LED is on. If the voltage is close to the specified max/min voltage of the unit, the voltage will show in red instead of green.

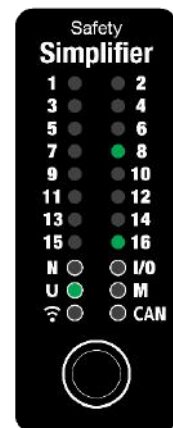


Figure 11.9

U level 1

### 11.9.3.6 U (second level)

Shows the voltage of the other units in the system. The voltage is the sum of the LED numbers (for example on Figure 11.10, the voltage is 8+16=24V). A short press on the button when in this level switches between the different nodes in the system. A short blink on an LED indicates the number of the node being looked at.

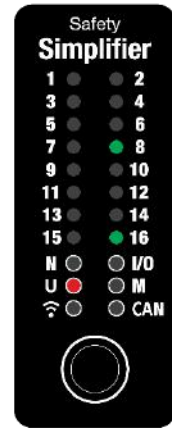


Figure 11.10

U level 2

### 11.9.3.7 U (third level)

The third level of the U menu indicates the temperature inside the enclosure of the units in the system. The temperature is the sum of the LEDs on the display (for example on Figure 11.11 the voltage is 12+15+16=43°C). A short press on the button when in this level switches between the different nodes in the system. A short blink on an LED indicates the number of the node being looked at.

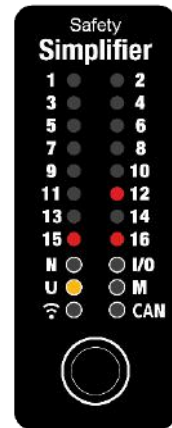


Figure 11.11

U level 3



**Note**

The temperature is measured directly on the PCB which is warmer than the ambient temperature due to heat generated from different components.

### 11.9.3.8 M (first level)

This menu displays the state of the 16 global memories from this unit. A green LED indicates that the memory is high. A red LED indicates that the memory is OFF. If the LED is turned off it means the memory is not configured in logic. In Figure 11.12 global memory 1 is high, global memory 6 and 8 are low, and the rest are not configured.

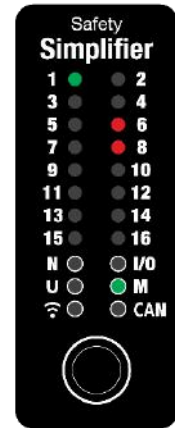


Figure 11.12

M level 1

### 11.9.3.9 M (second level)

This menu shows the state of the global memories of all units in the system. A short press on the button when in this level switches between the different nodes in the system. A short blink on an LED indicates the number of the node being looked at.

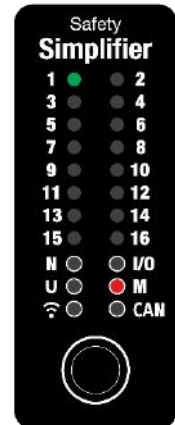


Figure 11.13

M level 2

### 11.9.3.10 📶 (Radio) (first level)

This menu displays the radio channel the unit is on. In Figure 11.14 the unit is on channel 5.

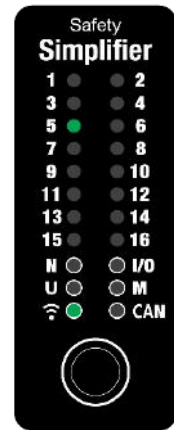


Figure 11.14

Radio level 1

### 11.9.3.11 📶 (Radio) (second level)

The second level displays the nodes that this unit has radio connection to. A green LED indicates good connection. An orange LED indicates OK/poor connection. A red LED indicates no connection. In Figure 11.15 the node has connection to node 1 (which in this case is itself), it has poor connection to node 2, and no connection to node 3.

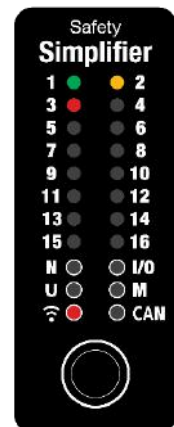


Figure 11.15

Radio level 2

### 11.9.3.12 CAN (first level)

This menu shows which nodes this unit has CAN connection to. In Figure 11.16 the unit has CAN connection to node 1 (itself), and node 2, indicated by green. It does not have CAN connection to node 3, indicated by red.

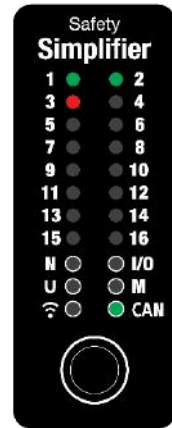


Figure 11.16

CAN level 1

### 11.9.3.13 CAN (second level)

This menu shows the state of the 16 CAN memories on this unit. A green LED indicates that the memory is high. A red LED indicates that the memory is OFF. If the LED is turned off it means the memory is not configured in logic. In Figure 11.17 CAN memory 1 and 16 are high, and CAN memory 15 is low.

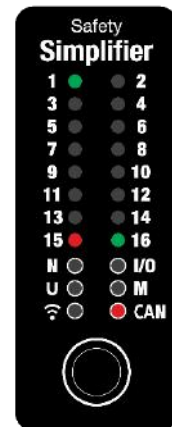


Figure 11.17

CAN level 2



### 11.9.3.14 CAN (third level)

This menu indicates which baud-rate the unit is using on LED 1, 2, or 5:

- 1: 125kbit
- 2: 250kbit
- 3: 500kbit

In Figure 11.18 the unit is using a baud-rate of 250kbit (LED 2 is green).

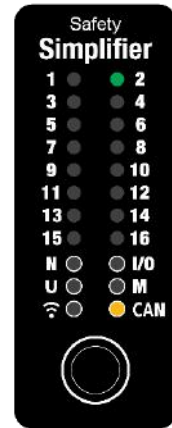


Figure 11.18

CAN level 3

### 11.9.4 Fatal error mode

When an internal or external error is detected, the unit can enter error state. In this state all I/O are turned off, the radio and CAN communication stops, and the unit displays an error code on the LED display.

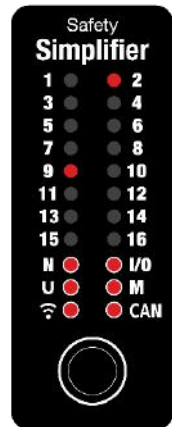


Figure 11.19

Display of a unit that is in fatal error state.

#### 11.9.4.1 Fatal error codes

The error codes are indicated on the LED display by all six menu LEDs lighting in red (N, I/O, U, M, etc), and one or more numbered LEDs lighting up in red (in Figure 11.19 LEDs 2 and 9 are lighting, indicating an undervoltage error). The error can be decoded by looking at the display and seeing which LEDs are lighting up and finding the corresponding row in Table 11.1: Fatal error codes and descriptions on page 49.

Display LEDs	Code	Name	Description	Solution
2	0x0002	NO_NETWORK_ID	No configured network ID.	Download a new configuration to the unit.
1 & 4	0x0009	SETTINGS_SERIAL_NUMBERS	Invalid serial number settings.	Download a new configuration to the unit.
2 & 4	0x000A	SIO_OVER_CURRENT_IRQ	Overcurrent protection. There is an output connected directly to ground.	Disconnect the faulting terminal.
1 & 2 & 3 & 4	0x000F	BAD_CFG_NODE_NR	Invalid node number settings.	Download a new configuration to the unit.
9	0x0100	PWR_SUPPLY_TOO_HIGH	Voltage is higher than the configured max voltage.	Check power supply, and project settings.
9 & 2	0x0102	PWR_SUPPLY_TOO_LOW	Voltage is lower than the configured min voltage.	Check power supply, and project settings.
15 & 16 & 1-8	0xC001 to 0xC0FF	USER_FATAL_ERROR	User fatal error triggered from logic.	See program logic for this unit.

Table 11.1: Fatal error codes and descriptions

### 11.9.5 Configuration mode

When a unit enters configuration mode it is indicated on the display by lighting the bottom six menu LEDs, and LED 1, in orange, as shown in Figure 11.20.

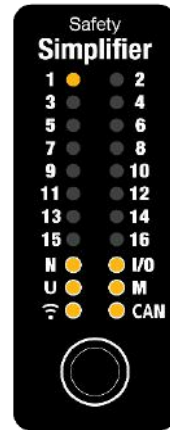


Figure 11.20

*Display of a unit in configuration mode.*

## 11.10 LED Display quick reference

### 11.10.1 First level menu information

When in the first level of a menu the menu LED is green. Switch between these menus by a firm quick press on the capacitive button.

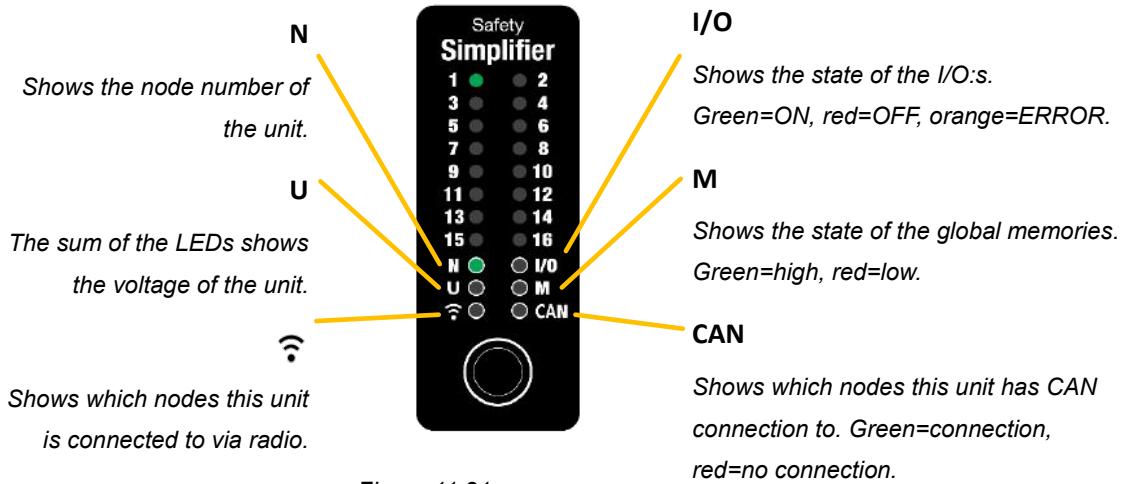


Figure 11.21

First level information

### 11.10.2 Second level menu information

To go to the second level of a menu, press and hold the button for one second. To switch back to the main menu, press and hold the button again (twice if you are in the **U** or **CAN** menus). When in the second level of a menu, the menu LED is red. Some second level menus display information from other nodes in the system. A firm short press on the button in those menus switches between the different nodes.

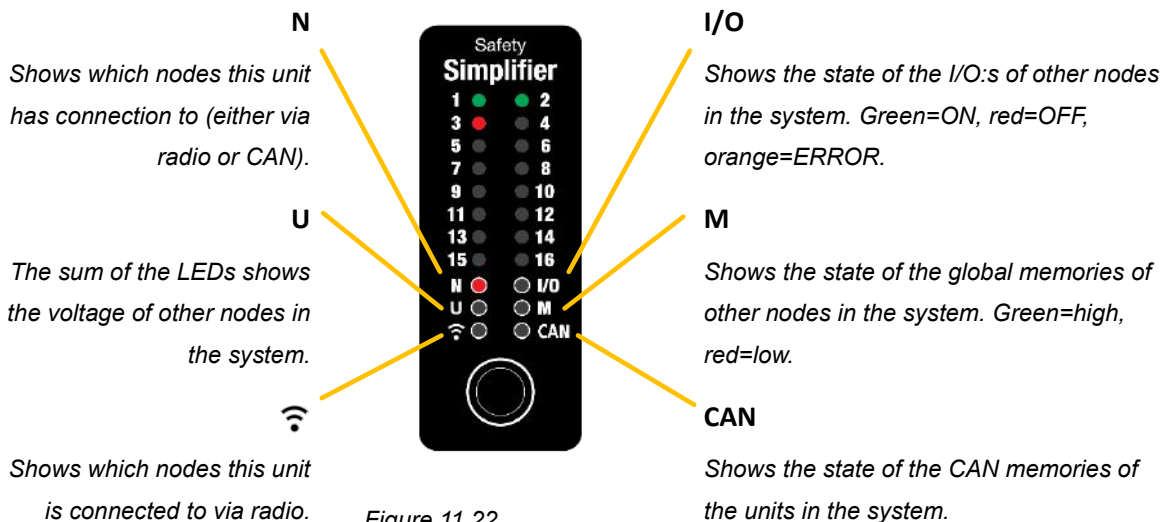


Figure 11.22

Second level information.

### 11.10.3 Third level menu information

The U and CAN menus have a third menu level. The third level menu is indicated by the LED turning orange.

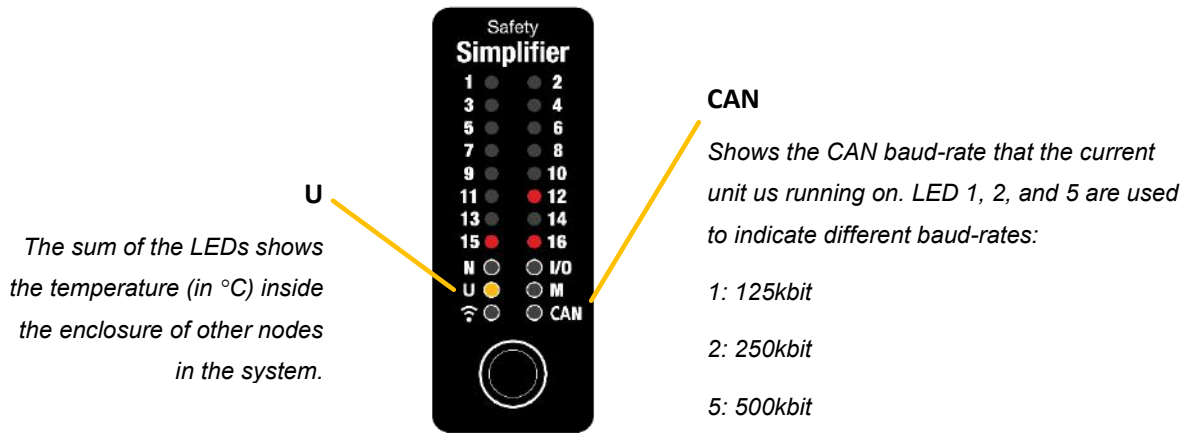


Figure 11.23

Third level information.

# Software

Using Simplifier Manager to design and  
program Safety Simplifier Systems

## 12 Introduction of the software Simplifier Manager

This part of the manual aims to introduce the user to the basics of programming and debugging Simplifier systems with the help of Simplifier Manager. The following chapters introduces Simplifier Manager and how to make programs for Simplifier units and systems. It guides the user through the steps of making their first program, and the following chapters go into detail of how to design and debug more complex systems, as well as common safety mistakes and how to avoid them.

## 13 Simplifier Manager

### 13.1 Notice

Simplifier Manager is a tool to help the developer develop safety applications. While Simplifier Manager can and does detect some common logic errors and other systematic errors, it does not guarantee that all errors are detected. All Simplifier systems and their programs must be thoroughly tested and verified against the function specification before being used in any safety application.

### 13.2 Windows 7 and 10

The current version of Simplifier Manager supports both Windows 7 and 10, however on Windows 7, to connect to USB devices like the Simplifier Monitor, a special USB driver must be installed. No extra driver is needed when using Windows 10. See chapter 13.3.1.3 for details on how to install the driver for Windows 7.

### 13.3 Installation

The Safety Simplifier Manger is available to our website [ssp.se](http://ssp.se).

To install Simplifier Manager, double click the "SimplifierManagerSetup.msi" file. It guides the user through the installation process, where the user can select install location. The same process is used to update Simplifier Manager.

#### 13.3.1 External Tools

##### 13.3.1.1 Compiler

When Simplifier Manager is started for the first time it asks the user to install the GCC compiler (if it is not installed already), which is needed for compiling and downloading programs to units. Simplifier Manager can still be used to make programs if the user decides to not install the compiler, but programs cannot be compiled or downloaded.

##### 13.3.1.2 PDF Viewer

For displaying project reports and other documents generated by Simplifier Manager, a PDF viewer such as Adobe Acrobat must be installed.

##### 13.3.1.3 Windows 7 USB Driver

Contact your representative for details on installing the driver.

## 13.4 User Interface Layout

Apart from the standard menu and toolbar, Simplifier Manager has four main parts; the middle area containing open “Documents”, the Toolbox, the Inspector, and the Project Manager. If any panel is closed, it can be opened again from the View menu. Restarting the program resets the view to default.

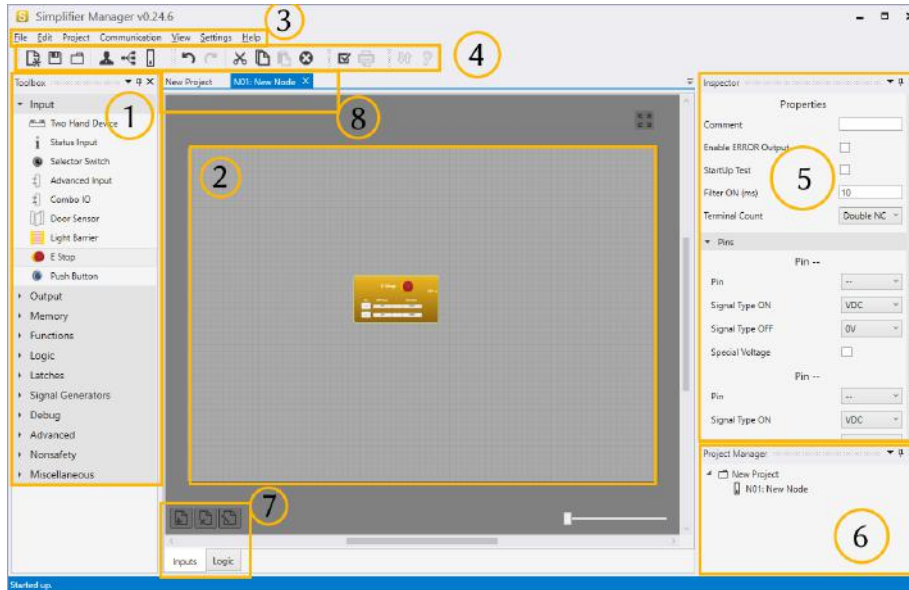


Illustration 13.1: The layout of Simplifier Manager.

1. Toolbox  
Here all the different logic blocks can be found.
2. Logic graph  
This is where all logic is created.
3. Main Menu  
Here all the standard functions can be found.
4. Toolbar  
Here the most frequently used functions can be found.
5. Inspector  
Here properties for blocks in the logic can be changed.
6. Project Manager  
Here an overview of the project is displayed.
7. Page controls  
These are the controls for managing pages for the currently open logic document.
8. Documents  
Here the different open documents are displayed.

### 13.4.1 The Main Menu

The main menu is the standard Windows style menu found in the top left corner of the program. Here all different commands can be found.

File



The file menu contains commands for managing files and projects.

- **New Project (Ctrl+N)**  
Opens the project settings dialog for creating a new project.
- **Open Project (Ctrl+O)**  
Opens the default windows file selection dialog, where an already existing simple project file can be selected and opened.
- **Save (Ctrl+S)**  
Saves the current project, overwriting the currently open project file. If the project has not been saved before, the save file dialog will display, letting the user select a directory and a name for the project file. **Note** that the project file is the same as the file name, and if a new name is selected for the file, the project will change its name to the new name.
- **Save As (Ctrl+Shift+S)**  
Opens the save file dialog where the user select a new directory and name and saves the project in that location.
- **Close Project**  
Closes the current project. If the project has not been saved, the user is asked if the project should be saved, or if the changes should be discarded.
- **Exit (Alt+F4)**  
Closes Simplifier Manager. If the project has not been saved, the user is asked if the project should be saved, or if the changes should be discarded.

#### Edit commands

The Edit menu contains commands for editing the blocks in the currently open graph editor.

- **Undo (Ctrl+Z)**  
Undoes the last undoable change in the current node. **Note** that some changes in the project are not undoable.
- **Redo (Ctrl+Y)**  
Redoes the last undone change.
- **Cut (Ctrl+X)**  
Cuts the currently selected blocks from the graph editor. This is equivalent to doing Copy+Delete.
- **Copy (Ctrl+C)**  
Copies the currently selected blocks to the clipboard.
- **Paste (Ctrl+V)**  
Pastes the blocks that are in the clipboard, to the currently open graph editor.
- **Delete (Delete)**  
Deletes the currently selected elements from the graph editor.

#### Project

The project menu contains commands for editing the currently open project.

- **New Node**  
Adds a new node to the project.
- **Project Settings**  
Opens the Project Settings dialog, where the project settings (Project Name, Responsible Person, Organisation, Description), and radio settings (Channel, Short Timeout, Long Timeout) can be changed.
- **Hardware Configuration**  
Opens the Hardware Configuration document.
- **Manage Passwords**  
Opens the Password Manager, where passwords can be set and changed.

- **Compile**  
Checks and compiles the program. If the logic contains any errors, the list of errors displays.
- **Generate project PDF**  
Generates the project report PDF and opens it in the default PDF viewer on the computer.

### Communication

The Communication menu contains various communication related commands.

- **Open Communication Window**  
This command opens the Communication Window, where programs can be downloaded to units via the Simplifier Monitor, or via USB.
- **Start Online Mode**  
Starts or stops the Online Mode. This command is only available when the project is compiled, and all serial numbers are specified.
- **Update Monitor**  
Updates the firmware of the connected Simplifier Monitor.
- **Update Firmware**  
Opens the dialog for updating firmware via radio.
- **Set Password (USB)**  
Opens the dialog to set password via USB.

### View

The view menu contains commands for opening different panels in the program.

- **Inspector**  
Opens the Inspector panel and sets focus to it.
- **Project Manager**  
Opens the Project Manager panel and sets focus to it.
- **Toolbox**  
Opens the Toolbox panel and sets focus to it.
- **Online Overview**  
Opens the Online Overview Document. This can only be done when in Online Mode.

### Help

From the Help menu the Manual can be opened.

- **Open Manual**  
Opens this manual

## 13.4.2 The Toolbar

The Toolbar is displayed right below the Main Menu and contains many of the most commonly used commands.

## 13.4.3 Documents

Documents in Simplifier Manager much like tabs in modern Web Browsers. When a document is opened it will display in the middle area of the window. You can switch documents by clicking on the different documents in the document header section.

Most of the work done in Simplifier Manager will be done in some type of Document (for example, the program logic is edited in a document).

## 13.4.4 The Toolbox

This panel houses all the different programming blocks available when programming. It is empty when a logic document is not open.

## 13.4.5 The Inspector

When a block is selected, its properties can be edited in the Inspector.

## 13.4.6 The Project Manager

Here an overview of the project is displayed. It displays the nodes in the system. Double clicking on a node in the project tree opens the node logic document. Double clicking on the project opens the project overview document.

# 13.5 Making your first program

## 13.5.1 Creating a Project

The first thing the user will see is the Home Screen document, from which a new project can be created. Clicking on Create New Project brings up the Project Settings window, where several different settings can be changed. For now, we will only discuss the settings under Project settings, the Radio Settings will be discussed in detail in later chapters. Change Project Name, Responsible Person, Organisation, and Description as you wish. Leave Channel, Timeout Short and Timeout Long as they are (radio channels are discussed in chapter 0 on page 80, and timeouts are discussed in chapter 13.10.5.2 on page 86). Click OK to create the project. These settings can be changed at any time by going to Project→Change Settings. Note that when saving projects, the project name will be the file name, followed by the file extension “.simple”.

The screenshot shows a 'Project Settings' dialog box with the following fields and options:

- Project Settings:**
  - Project Name:
  - Responsible Person:
  - Organisation:
  - Description:
- Radio Settings:**
  - Channel:
  - Timeout Short (ms):
  - Timeout Long (ms):
- CAN Settings:**
  - CAN Baud Rate:

Buttons:

Illustration 13.2: The Project Settings dialog, where the project settings can be changed.

### 13.5.2 Hardware Configuration

When a project is created the Hardware Configuration document is shown. To add a node to the project, click the Add New Node button, or click Add Node in the toolbar. This will open the Node Settings dialog, where the node settings can be changed. The most important setting is the Type setting, which says if the node has relays or not. This must match the type of the node that is being used. If the node type does not match, the program cannot be downloaded to the unit. Pressing OK will add a Simplifier unit to the hardware configuration. To open the settings again for the new node, right-click the node in the Project Overview document (or in the Project Manager) and select Change Settings. Open the node logic by double clicking on the Simplifier, or right click and select Program.

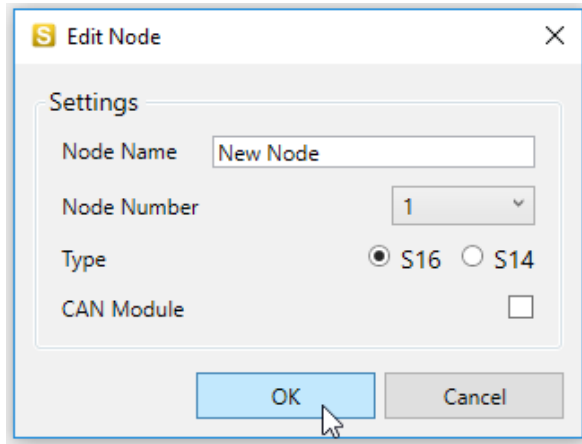


Illustration 13.3: The node settings dialog. Here the settings of the node can be changed.

### 13.5.3 The Logic Editor

Opening the program for node 1 displays the logic graph document. This is the document where all logic programming is done. Note that the Toolbox has changed and is now filled with different categories of function blocks. Function blocks are the building blocks used to program nodes. See chapter 14 for a complete list of all function blocks available in Simplifier Manager.

The node logic document contains the program for the selected unit. The program can be split over different so-called pages. One page is printed as a landscape oriented A4 in the project report PDF. The pages can be seen in the bottom of the document, along with the buttons for adding, removing and renaming pages.

It is recommended to split the logic over different pages, so it is easier for a reader to navigate and understand. The page name can be used to describe what kind of functions the page contains. Clicking the rename page button brings up the dialog for changing the page name. Change the name of this page to Logic.

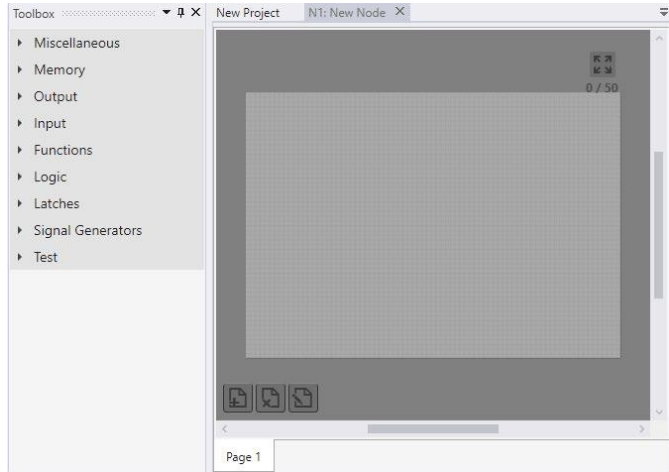


Illustration 13.4: The logic diagram (right) and the toolbox with all categories collapsed (left)



Illustration 13.5 The rename page button.

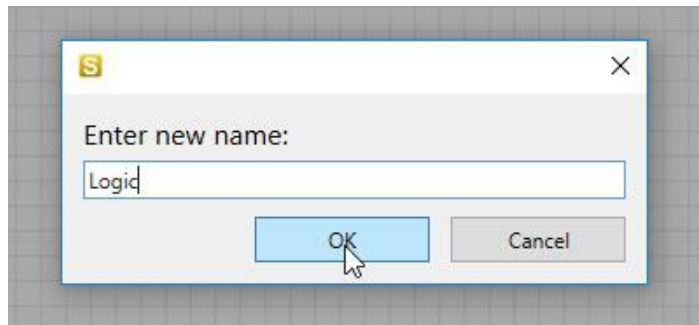


Illustration 13.6: The dialog for renaming pages.

### 13.5.4 Navigation in the logic graph

Before we continue it is good to understand how to navigate in the logic graph. To zoom in and out, use the scroll wheel on your mouse. To center the graph in the view click the button in the top-right corner. The numbers under the button display how many blocks have been placed in this page. Holding down the right mouse button and dragging with the mouse will pan the graph. The scroll bars on the bottom and the right side can also be used for scrolling horizontally or vertically.

### 13.5.5 The Goal of this example

In the coming chapters we will create a single node system with a basic E-Stop function with a reset function, for controlling a relay output. To accomplish this, we will use four different function blocks; E Stop, Push Button, Single Reset, and Relay Output. To activate the relay, the E-Stop must be pulled out and the Start Push Button pressed. Pressing the E Stop will stop the machine. The resulting function from the input terminals on the Simplifier to the output relay is up to SIL3, PLe cat 4 according to 61508 and 13849-1, and is useable in safety applications.

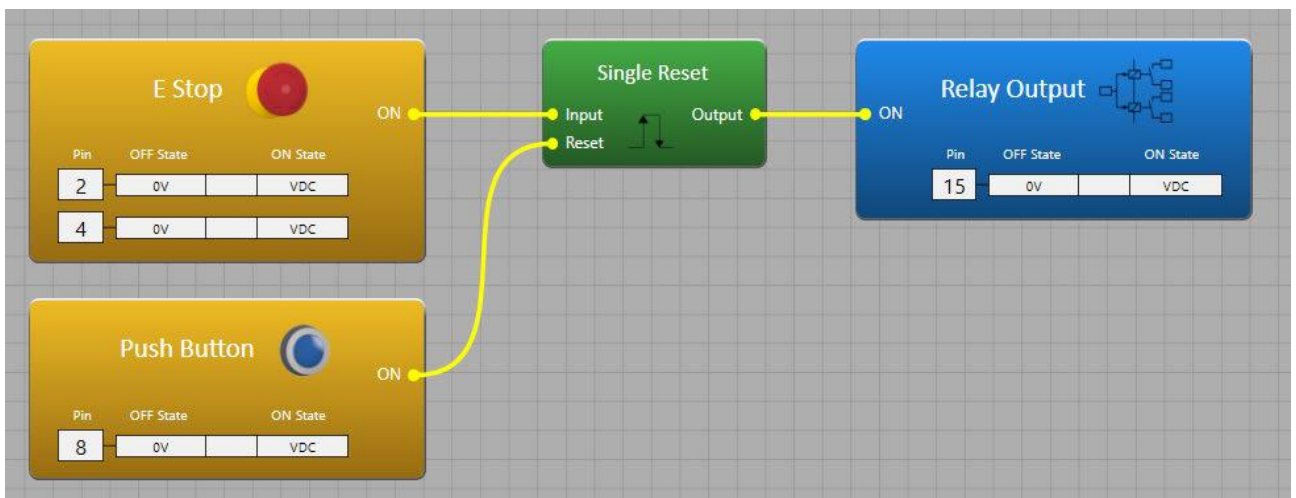


Illustration 13.7: How the function will look at the end of this chapter.

Afterwards we will expand the program by adding another node to the system, that will communicate with node 1 via radio. Both nodes will each have an E-Stop, and the relay output should fall if any E-Stop is pressed. Node 1 will still have the push button to reset the function. The machine cannot start if an E-Stop is pressed, or if no radio communication between the nodes exist. Loss of radio communication will yield the same result as the E-Stop being pressed, which means the function must be reset with the push button if radio link is lost. The resulting function from input to outputs is up to SIL 3, PLe cat 4 according to ISO 61508 and ISO 13849-1.

### 13.5.6 Adding Function Blocks

First, we will add an E-Stop to the graph. Expand the Input category in the toolbox, and with the left mouse button, click and drag an E-Stop from the Toolbox to the logic graph. When the block is created it will be selected, indicated by the yellow outline. Clicking on a block will also select it. Clicking on the graph background will deselect all selected blocks. When selecting the E-Stop block, notice that the Inspector now contains some editable properties; Comment, Enable ERROR Output, StartUp Test, Filter ON (ms), and Terminal Count. Under the Pins category are also settings for each pin. These are all the settings that can be changed for this block. Terminal signal types are discussed in chapter 13.11, along with ON state and OFF state. For now, it is enough to know that an input ON output goes high when all pins receive their ON state signal. In the ON state the ON output will output

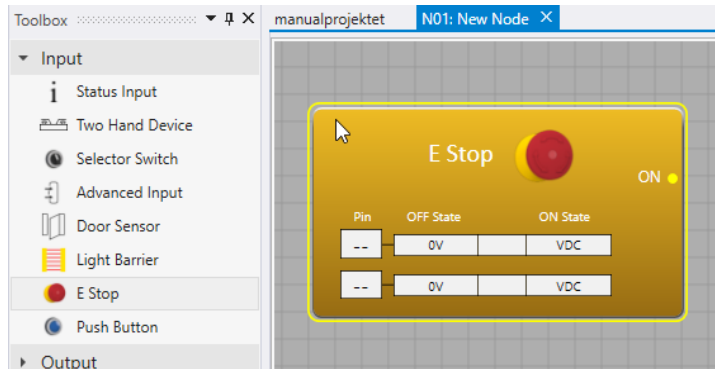


Illustration 13.9: Dragging an E Stop from the toolbox to the graph.

a 1 to the logic, meaning in this example that the E Stop is not pressed/pulled out/OK. In the inspector Change the first terminal number to 2, and the second terminal number to 4 (as shown in Illustration). This will program terminal 2 and 4 as an input, and the logical input signal can be used in the graph. Notice that the terminal number is displayed under Pin in the function block.

The Terminal Count property can be used to make the input a single channel input. We will keep it as a Double NC (Normally Closed contacts).

The Filter (ms) is for filtering the ON signal (which means it only filters the signals when the E Stop is pulled out/reset). It does not Filter the OFF signal, so when the button is pressed there is only the predefined filter of 2 ms before the function goes to OFF state (see chapter 13.11 for a description of the input function).

Enable Error enables the error output connector on the block. This can be used for indication or other logic.

For every channel of an input, a Terminal Number must be selected. Any terminal between 1-14 can be used for any input function. For the E Stop, the Signal Type ON must also be defined for both terminals.

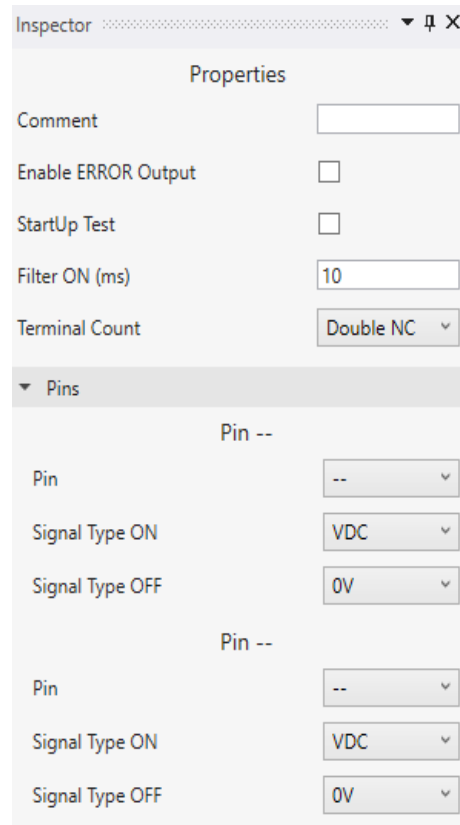


Illustration 13.8: The changeable properties of the E Stop block.

### 13.5.7 Moving Blocks

To move blocks around in the graph, hold down left mouse button and drag the block. Selecting multiple blocks and dragging moves all selected blocks.

### 13.5.8 Deleting Blocks

To delete a block from the logic graph, select it by left-clicking on it, and press Delete on the keyboard, or right click on it and select Delete. The delete button in the Toolbar can also be used, or by going to Edit→Delete.

### 13.5.9 Selecting Multiple Blocks

To select multiple blocks at a time, hold Ctrl on the keyboard and click the blocks that should be selected. Notice that when more than one blocks are selected, the inspector is empty. Clicking on a block that is already selected while holding the Ctrl key will deselect the block.

### 13.5.10 Copy, Cut, and Paste

To copy the currently selected elements, go to Edit→Copy, or press Ctrl+C. To cut the selected elements, go to Edit→Cut, or press Ctrl+X. To paste the elements currently in the clipboard, go to Edit→Paste, or press Ctrl+V.

### 13.5.11 Connections

The program logic is built by creating blocks and connecting internal signals between them. An internal signal is a binary value that can be either 1 or 0, that is represented with a line in the logic. Note that safety signals are coloured yellow, and unsafe signals are dashed and coloured white. Unsafe and safe signals are discussed in more detail in chapter 13.14.2 . For now, we will only use safe signals (yellow).

A connection is created by dragging from a so-called output connector, to an input connector. A block can have zero or more input and output connectors. Connectors on the left side of a block are input connectors, and connectors on the right side of a block are output connectors. This makes the logic flow from left to right, where outputs from blocks on the left side are inputs to blocks to the right of them.

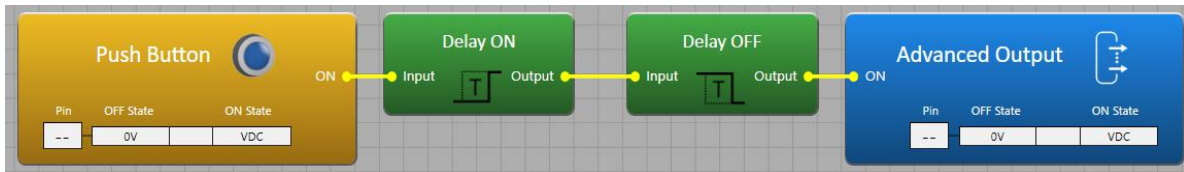


Illustration 13.10: Example of a series of blocks connected with connections.



## Adding a reset function

To create a reset function for the E Stop, expand the Functions category in the Toolbox, and drag a Single Reset block to the graph. The Single Reset function is a monitored reset of the input signal. When the input signal is 1, the Reset input of the block must go 0→1→0 for the output to go to 1 (see chapter 14.4.3 on page 124 for a description of the Single Reset function). With the Single Reset block selected, go to the Inspector and disable Indication. This removes the Lamp output from the block, which we will not use.

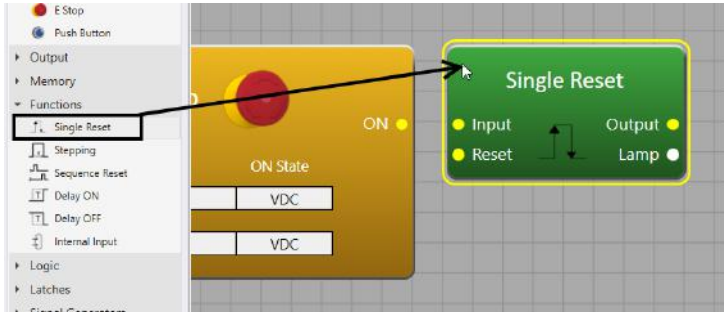


Figure 13.1: Dragging a Single Reset to the logic diagram.

To connect the signal from the E Stop to the Single Reset, drag from the yellow output connector hotspot (ON) on the E Stop block, to the input connector on the Single Reset block. A yellow line (from here on referred to as a connection) follows the mouse. To create the connection, release the mouse button when the connection has snapped to the connector on the Single Reset block. This creates a connection between the blocks, which means the Single Reset block will use the ON signal from the E Stop as input to its function.

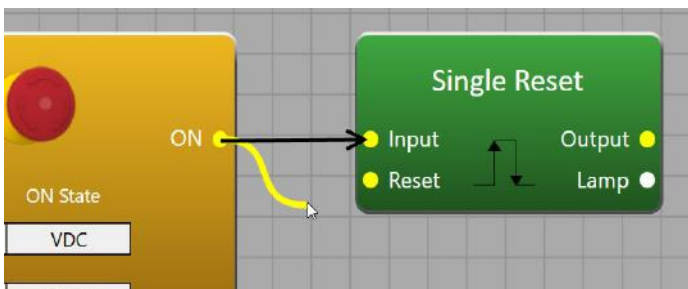


Illustration 13.11: A connection is created by dragging from an output connector to an input connector.

### 13.5.12 Removing Connections

To remove a connection, right click on a connector and select "remove connection". Doing this on an output connector will delete all connections coming from that connector.

### 13.5.13 Final steps

To complete the E Stop function, expand the Input category and drag a Push Button block to the graph. The Push Button has four settings; Indication, Terminal Number, Signal Type and Comment. If indication is enabled, an input connector will appear on the block, allowing for indication on the same pin as the input. The Terminal Number can be set to any free input terminal. When changing this, notice the terminal numbers 2 and 4 are not available, as they are used for the E Stop input. Assign the push button to pin 8 and connect its ON output to the Reset input on the Single Reset block. If you have an S16 unit, you can drag a Relay Output to the graph and change its Terminal Number to 15 or 16. If you have an S14 unit you can drag out an OSSD Output instead and configure the output terminals to any two available pins. Note that pins 2, 4 and 8 are not available as they are in use by the input functions.

### 13.5.14 Checking and Compiling

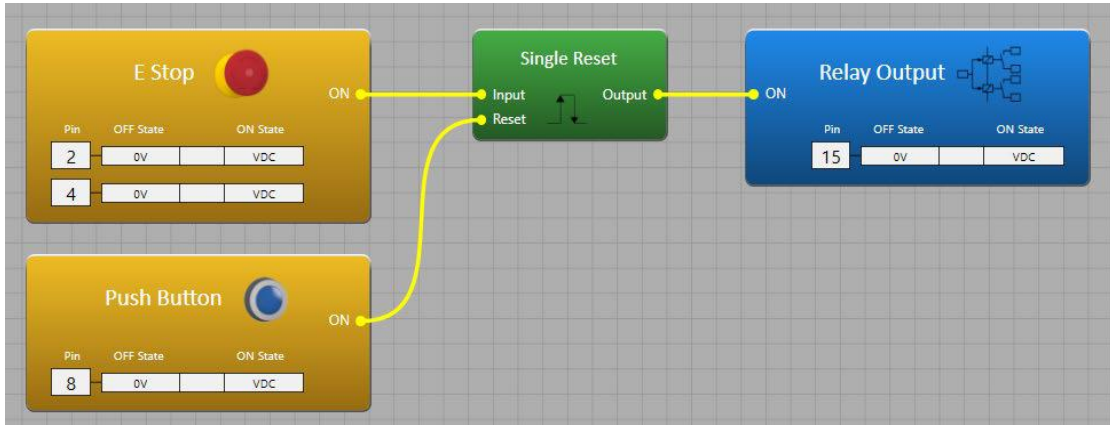


Illustration 13.12: The finished E Stop program. For units without relays (S14) an OSSD Output can be used.

To verify that this program is valid, click the "Compile" button in the toolbar (or press Ctrl+B on the keyboard). If the compiler is installed, the program will be checked for errors and then compile, displaying the program checksum when finished. If the compiler is not installed, the program will only be checked for errors, but is not compiled.

If there is an error in the logic, a message will tell the user that there is an error in logic, and the error list will open. The error list contains all errors in the logic, for example inputs with unassigned terminals, connectors that are missing connections, and other errors.

This program can now be downloaded to a system. See chapter 13.8 for a description of how to download programs to systems.

### 13.5.15 Saving and loading projects

To save the project, go to File→Save, or press Ctrl+S. The Save file dialog is displayed. The Project name is already entered as the file name. Be aware that changing the saved file name will also change the name of the project. The saved file is saved in the Simplifier Manager project file format (with the ".simple" file extension).

To open a file, go to File→Open, or press Ctrl+O. The load file dialog is displayed. Here a Simplifier Manager project file can be selected to load. If the project file is password protected, a dialog shows prompting the user to enter the password for the project. If the wrong password is entered, the project cannot be loaded. Passwords are discussed in chapter 13.9.

## 13.6 Expanding the program

If we want another E Stop somewhere else, we need to add a second node to the project and configure the function to work with both nodes. This is done by sending the E Stop signal via a so-called Global Memory. Before we do this, it is good to understand how memories work in general.

### 13.6.1 Local Memories

A Local Memory works just like a connection. It references a signal in the program, so it can be used in multiple places. The local memory is free to use, meaning it does not take any memory in the compiled program, it only references the signal it is connected to. Replacing a connection with a local memory will compile into the same program (this can be seen by compiling two projects and comparing their checksums). In the case where a signal is used as input for many blocks, a local memory can reduce the amount of crossed connections, making the logic easier to understand. They are also necessary for connecting signals across pages.

To create a Local Memory, expand the Memory category in the Toolbox, and drag out a Memory to the graph. This memory is by default a Local Memory but can be changed by changing the Memory Type. The Memory Type indicates which type of memory this is. Global and CAN memories can be referenced across different nodes in the system (they are discussed in chapter 13.6.6), while local memories are only used within the node itself. The tag inside the square brackets next to the Memory block indicate what type of memory it is. For now, we will focus on local memories.

To change the name of the memory, change the Memory Name property in the inspector. The memory name can be any UTF-8 string of characters, but for readability it is recommended that a consistent naming convention like snake\_case or PascalCase is used. For these examples snake\_case is used. The memory name is used when

referencing the memory. It is recommended to name the memory after what signal is being sent to it (for example, when connecting an E Stop to a memory, the memory should be named e\_stop or something similar that describes what the actual signal is, or where it comes from).

To reference a memory, drag a Reference to the logic graph, and select the source memory that you want to reference to in the Inspector (see Illustration 13.15).



Illustration 13.13: A local memory source, with the name "New Memory".

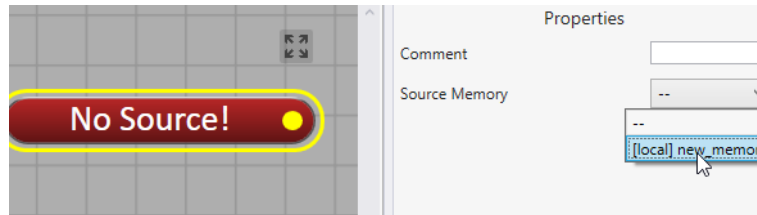


Illustration 13.14 Selecting a source memory for an unlinked reference.

### 13.6.2 Adding Local Memories to the program

Let's now replace the connections we have made from our input functions with Local Memories. First, we need to remove the old connections. To do this, right click on either the input or output connector and select Delete Connections.

Now we want to replace the old connections with Local Memories. To do this, drag a Memory from the toolbox to the graph, and change its name to "e\_stop". Now connect the E Stop ON output to the memory. Do the same for the Push Button and name the memory "reset\_button".

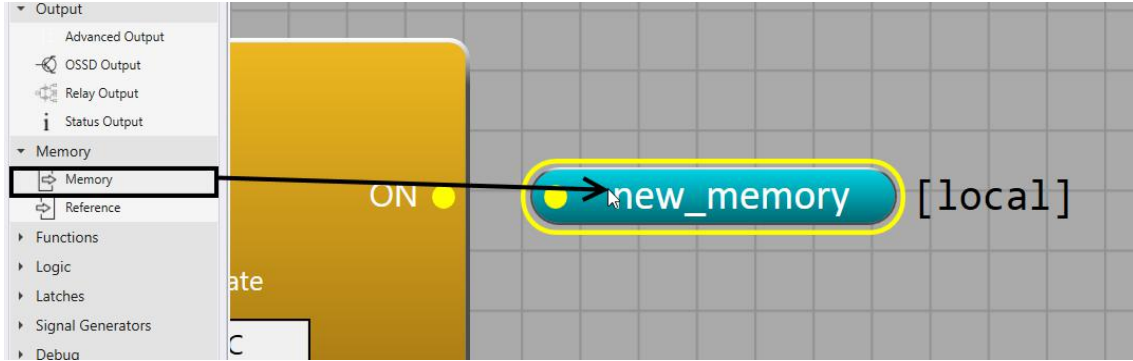


Illustration 13.15: Dragging a Memory to the graph.

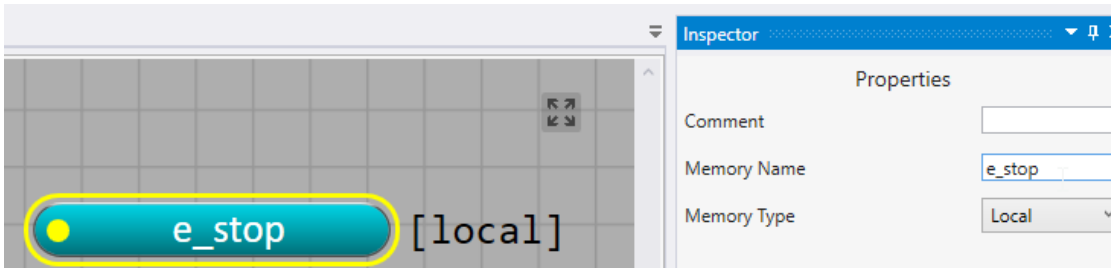


Illustration 13.17: Changing the memory name to e\_stop.

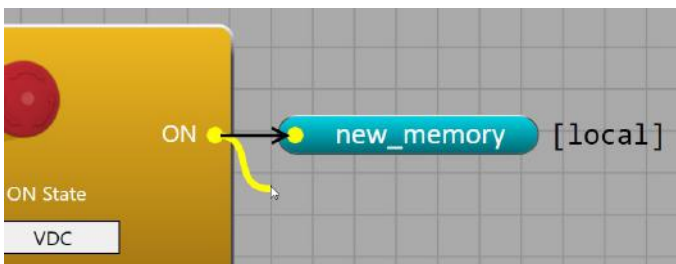


Illustration 13.16: Connecting the E Stop to the memory.

### 13.6.3 Managing Pages

The program we have currently is very simple, so we don't *need* more than one page. However, it is good to understand how pages work, so we will go through the process of working with pages here. This process can then be applied to larger projects.

To add another page to the logic, press the Add Page button in the lower left corner of the document. A dialog will show where a new name for the page can be entered. Name the page to "Logic".

To switch between pages, click the tabs in the bottom of the document. To rename a page, click the rename page button in the bottom of the document.

Switch back to the "Page 1" page, press the rename page button and rename it to "Input". Now we have two pages; "Input" and "Logic".

Now we need to move the reset and output blocks from the Input page to the Logic page. The easiest way to achieve this is to cut and paste the element. With the Ctrl key pressed, select the Single Reset block, and the Relay Output block. Cut the elements by going to Edit→Cut, pressing Ctrl+X on the keyboard, or pressing the Cut button on the toolbar. The elements are now in the clipboard and can be pasted somewhere else. Go to back to the Logic page and paste the elements by going to Edit→Paste, by pressing the Paste button in the toolbar, or by pressing Ctrl+V on the keyboard. Move the blocks so they are in the center of the graph.

Now to make the function like it was before, we need to reference the "e\_stop" and "start\_button" memories in the Logic page and connect them to the single reset block. To reference a memory, drag in a Reference from the Memory category in the Toolbox. Notice it does not have a source signal yet. With the Memory Reference selected, go to the Inspector and in the Source Memory drop down and select the "e\_stop" local memory. Notice that the name and colour of the Reference changes to match the source memory. Connect the output of the memory reference to the "Input" input on the Single Reset block. Now drag another reference to the graph, set its Source Memory to the "start\_button" memory, and connect it to the "Reset" input (see Illustration 13.23)

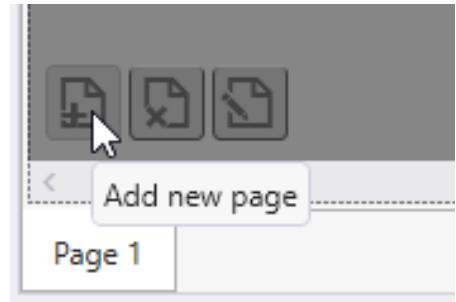


Illustration 13.18: The add page button.

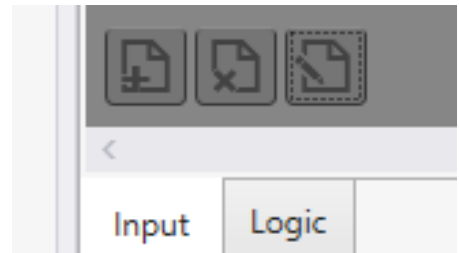


Illustration 13.19: The Input and Logic pages.

The function is now equivalent to the function before, and compiling yields the same program checksum as before.

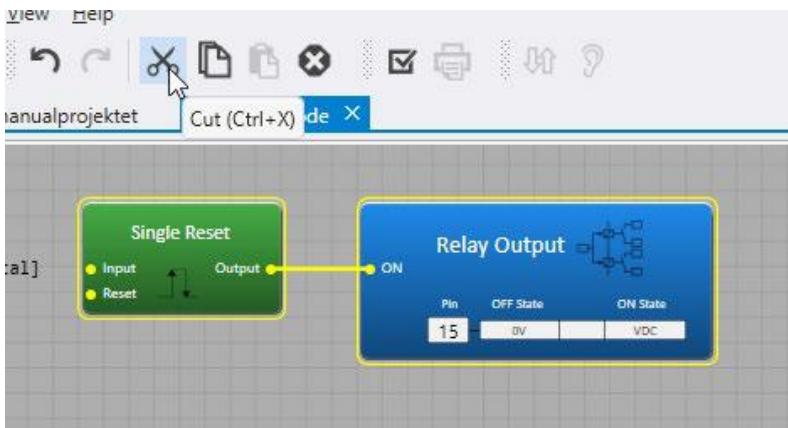


Illustration 13.20: Pressing the "Cut" toolbar button.

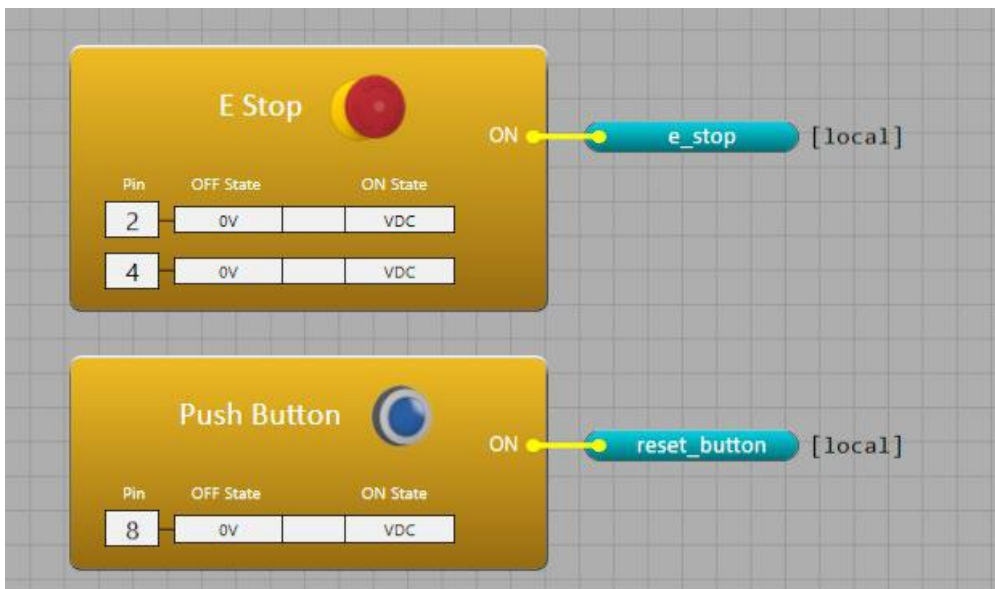


Illustration 13.22: Page 1 ("Input")

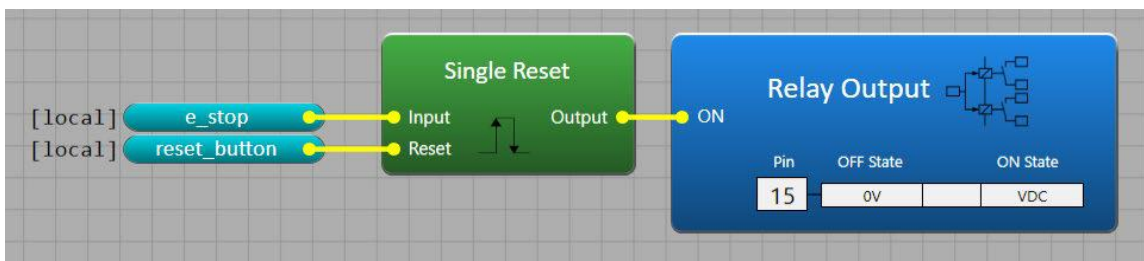


Illustration 13.21: The finished reset and E-stop function.

### 13.6.4 Adding more nodes

To add a second node to this project, click the Add Node button in the toolbar, or go to Project→Add Node. A new empty graph document will open. Here node 2 can be programmed in the same manner node 1 was programmed. To switch between the logic of node 1 and node 2, the document tabs can be used. A document can be closed by clicking the X button on the document tab, right clicking and selecting Close, or by clicking on it with the middle mouse button. If a logic document has been closed it can be opened by right clicking the node in the Project Manager, and selecting Program, or double clicking on the Program leaf under the node.

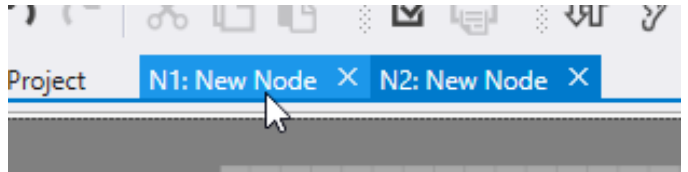


Illustration 13.23: The headers for the logic document of node 1 and node 2. Switching between documents is done by clicking the headers.

### 13.6.5 Programming Node 2

Node 2 will have an E-Stop just like node 1, but instead of sending the signal to a local memory, the signal will be sent over radio using so called Global Memories. First switch to node 1 page 1. While holding the Ctrl key, left click to select the E Stop and the e\_stop local memory. Copy the elements by going to Edit→Copy or press Ctrl+C on the keyboard.

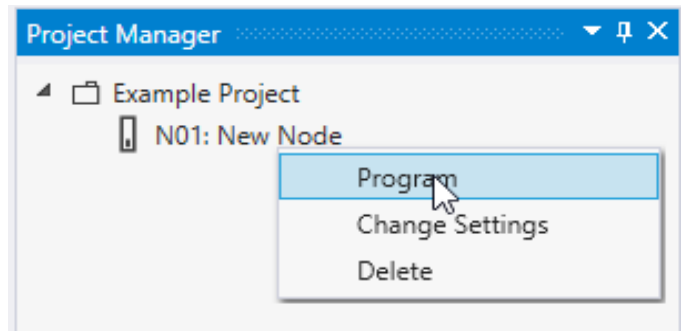


Illustration 13.24: The program for a node can be opened by double clicking on the node, or right clicking the node in the Project Manager, and selecting "Program".

Go back to the logic document of node 2 and paste by pressing Ctrl+V, or by going to Edit→Paste. The pasted blocks will have the same settings as the copied blocks.

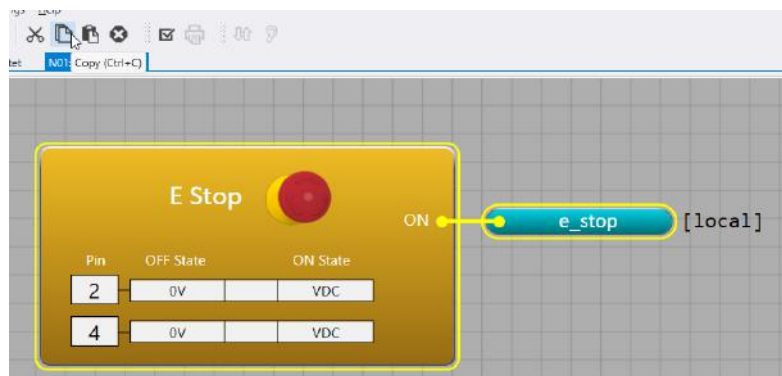


Illustration 13.25: Copy the E Stop and the memory in node 1.

### 13.6.6 Sending Information via Radio/CAN

To send safety information between two nodes via Radio or CAN, Global Memories and CAN memories can be used. See chapter 13.10 and chapter 13.10.5 for a more detailed explanation of how Global and CAN memories work.

In the logic for node 2, with the "E-stop" memory selected, change the Memory Type to Global. Notice the colour of the memory changes to gold, and the text "[unassigned]" appears next to it. This means the memory has not been assigned a memory number. Note that a memory cannot be used in another node until it has been assigned a memory number.

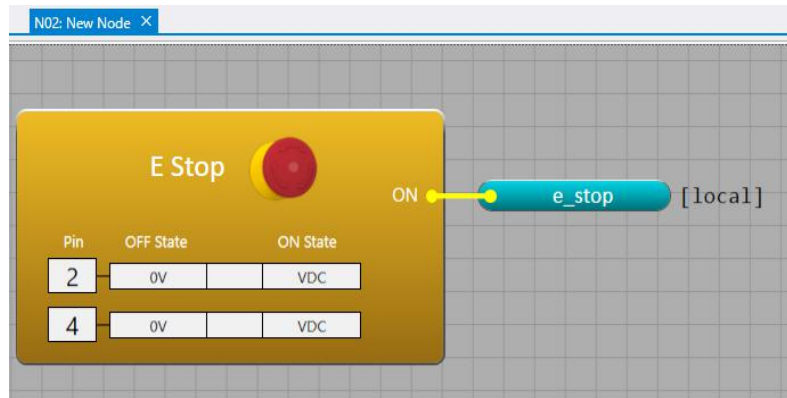


Illustration 13.26: Pasting the elements in node 2.

The memory now has some new properties in the Inspector; Number, StartUp Test, Do Not Repeat, and Reaction Time. These settings are discussed more in chapter 13.10.

Change Number to GM01. The text next to the memory now says "[N02.GM01]". Leave the other settings as they are for now (these properties are discussed in chapter 13.10.5.2).

Change the memory type of the "start\_button" memory to global as well and assign it to GM02. For this memory it is also crucial to enable StartUp test. Again, these properties are discussed in detail in chapter 13.10.5.2.

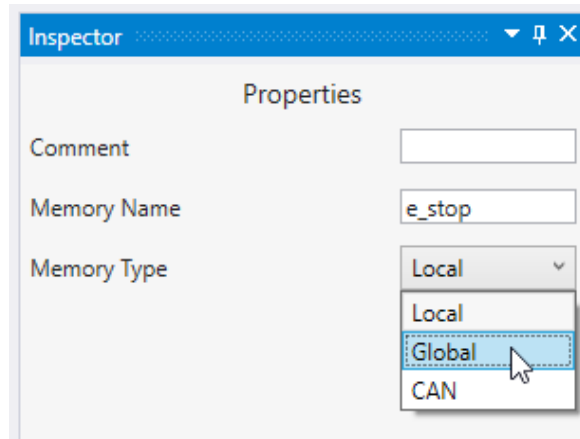


Illustration 13.27: Changing Memory Type to Global.



These memories can now be referenced in node 1, the same way local memories were referenced. Switch to the logic for node 1 and go to page 2. Drag in a Memory Reference from the Memory category in the toolbox. The "[N2.GM01] e\_stop" and "[N2.GM02] reset\_button" Global Memories from node 2 are now visible in the Source Memory drop down of the Memory Reference in node 1. Selecting the "[N2.GM01] e\_stop" will change the colour of the reference to gold, as well as show where the memory comes from in the brackets to the left of the reference, in our case [N2.GM01], which means node 2, Global Memory 1.

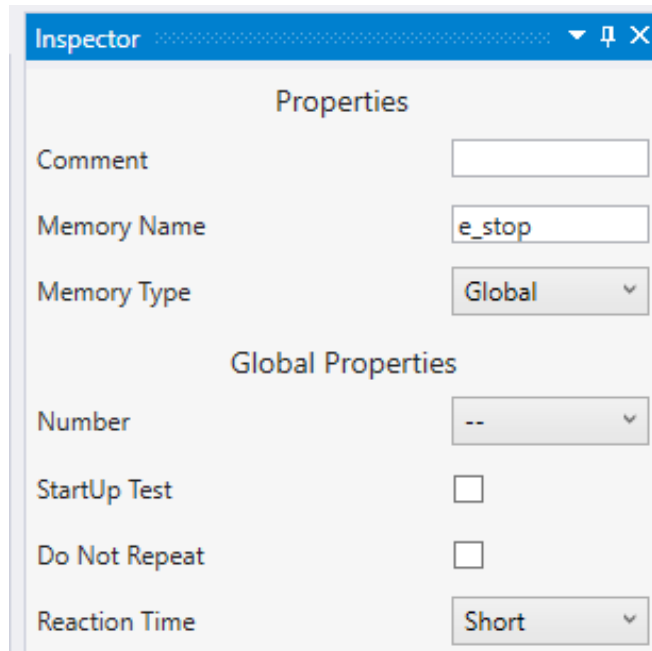


Illustration 13.28: The inspector after changing memory type to Global.

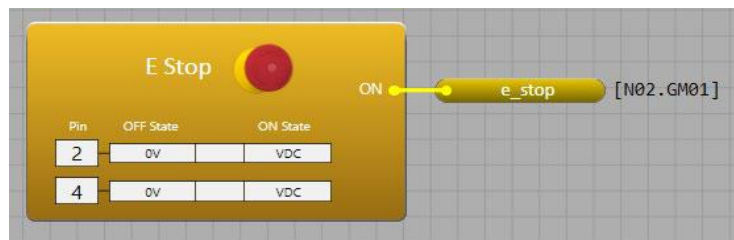


Illustration 13.29: The logic in node 2.

To combine the E Stops from node 1 and node 2, drag in an AND block from the Logic category in the Toolbox. Connect the "[N2.GM1] e\_stop" reference to the "1" input of the AND block and connect the "[local] e\_stop" reference that we created earlier to the "2" input of the AND block. Connect the Output of the AND block to the Input of the Single Reset block. Connect the Output of the AND block to the Input of the Single Reset block. Connect the Output of the AND block to the Input of the Single Reset block.

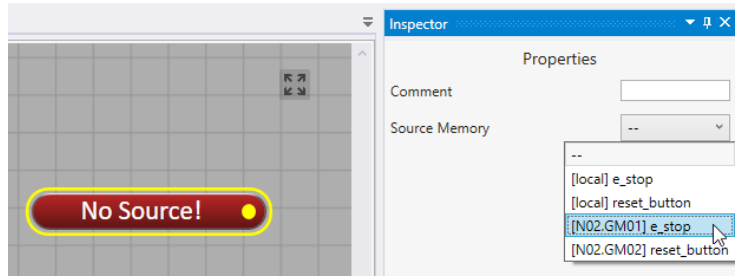


Illustration 13.30: A memory reference in node 1, and all its possible memory sources.

Finally, to make the Push Button in node 2 also start the machine, we will make a similar function to the one we just did, except we will use an OR block instead of an AND block (we can press either the button on node 1 or the button on node 2). See Illustration 13.33



Illustration 13.31: A reference with N02.GM01 as source memory. Note the color and tag match the source memory.

The function is now finished. Compile, download and go Online to see how the program works.

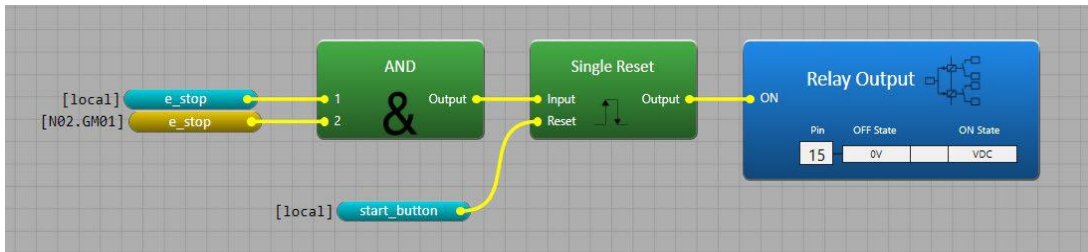


Illustration 13.32: The finished logic in node 1 page 2 ("Logic").

## 13.7 Online Mode

Simplifier Manager comes with an Online Mode debugging tool, that listens to systems and displays information about radio communication, CAN communication, I/O, Memories, voltage, temperature and internal signals, for debugging and monitoring systems.

The online mode can be used via USB directly to a Simplifier, or via radio using Simplifier Monitor.

To start Online Mode on a system, Simplifier Manager must have the same compiled program, and must have the correct serial numbers for all the units in the system. If it does, the ear icon in the Toolbar can be clicked to start the Online Mode. A dialog is displayed (see Illustration 13.34), where the user can select which device type to use (via USB or via radio with Simplifier Monitor).

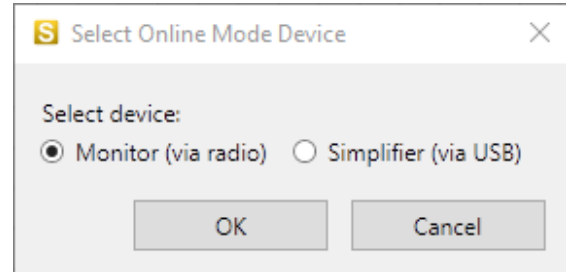


Illustration 13.33: Illustration 13.40 The dialog where the online device can be selected.

To see the state of all internal signals, simply open the logic document of a node. The high and low states of all signals is visualized by the colors red (LOW) and green (HIGH). When using the monitor, if the node turns off or is too far away (loss of radio contact) the signals in the logic turn blue, indicating that the node is no longer heard.

To get a summary of the state of the system, the Online Overview document can be opened by going to View → Online Overview. In the top left side of the document, a list of all the nodes in the system is displayed along with their voltage and temperature. The signal quality can also be seen here (note that the signal quality displayed here only affects the quality of the online mode and does not represent the quality of the safety communication between the nodes). Clicking on a node in this list displays the I/O and Global Memory status of the selected node in the top right side.

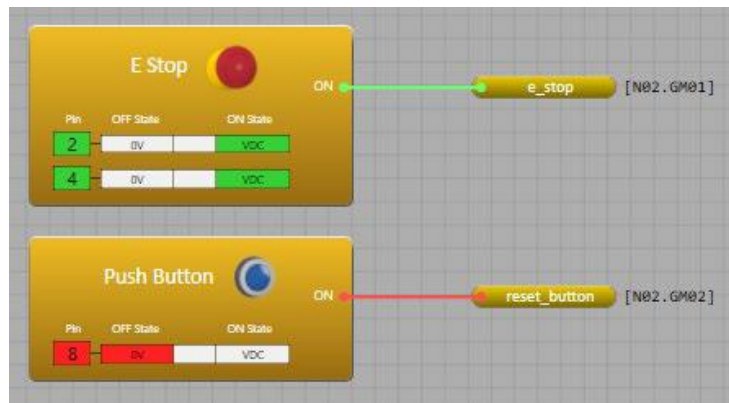


Illustration 13.34: Online mode on an E Stop in ON state, and a push button in OFF state. The green line indicates that the signal is 1 (HIGH), and the red line indicate that the signal is 0 (LOW).

Input and Output blocks show the current state of the terminals inside the block. If a terminal is in OFF state, the OFF-state signal background is coloured red. If a pin is in ON state, the ON state signal square background is coloured green. If a pin is in ERROR state, the middle square will be coloured orange, and the text "ERROR" will display.

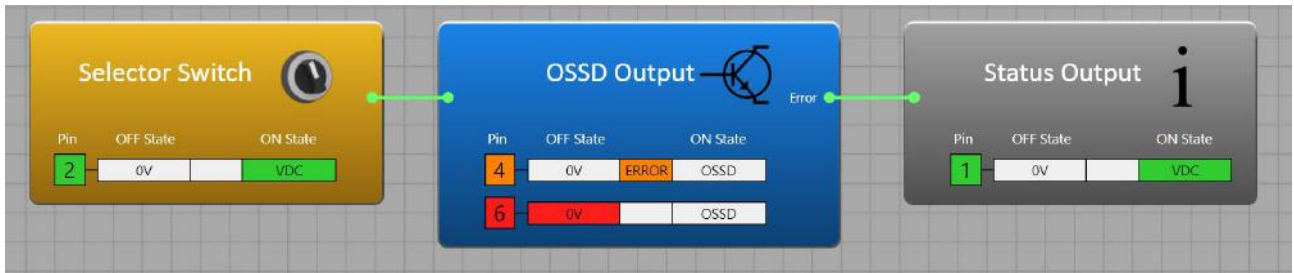


Illustration 13.35: Online on a Selector Switch, OSSD Output and a Status Output. The OSSD Output has an external error on terminal 4 (short circuit to 24V).

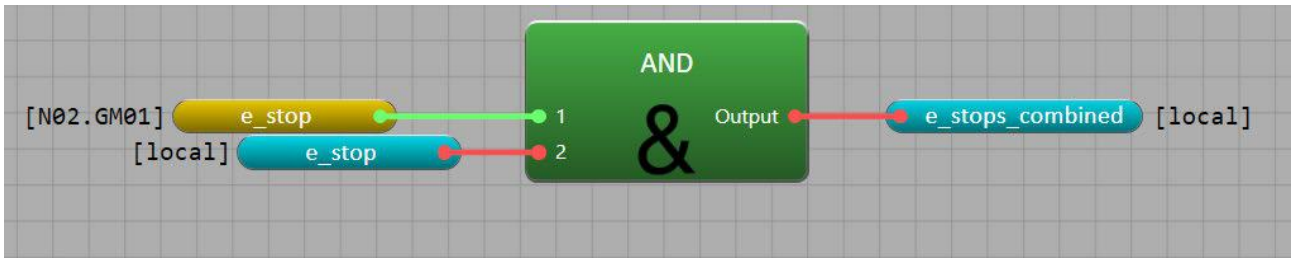


Illustration 13.37: Online mode on a reference to a Global Memory.

### Online Mode Notice

If online mode does not start, make sure the system is running on the correct channel, and that the program has been downloaded to all the units in the system. Also check that node 1 is running and has contact to the other nodes in the system. Downloading or reconfiguring the system again from the Communication Window can also solve the problem.

### Limitations

The status information from nodes in a system is sent periodically, and the update frequency depends on how many nodes there are in a system. A system with two nodes will therefore have a much faster update speed of the status information than a system with 16 nodes. This will result in the debug signals of larger systems updating more slowly than in smaller systems, and in systems of 16 nodes a delay of a couple of seconds can be seen.

Signals (internally in the unit) that change faster than the update speed of the status information over radio/USB cannot be seen by Simplifier Manager and will be lost (due to the sampling theorem). This occurs at a bigger scale in larger systems, due to the longer update time. For example, a quick press of a push button might not be seen on the online mode, even though the signal was registered internally in the unit.

Internal logic signals are sent from nodes split into chunks, which means only some are updated at a time. This can in some rare cases result in some functions looking like they are not behaving correctly. For example, an AND block could have all inputs green = 1, and the output red = 0. There is nothing wrong with the AND function, the output signal has just not been updated yet. This will only happen the instant when a signal in the logic changes and does not happen for signals that have not changed. Poor radio connection to the node will greatly increase the risk of this happening. To limit this effect, be close to the systems/nodes that are being monitored.

When using Global Memories in the logic a similar effect can be seen as well. The Global memory signals are received much faster than the internal signals (since the global memories are taken directly from the safety information, which has a much higher update frequency). When a Global

memory changes its value, there could be some time before the internal signals are updated, making the function look like it is not behaving as it should. The CAN memories do not suffer from this problem as much, since they are not sent on the radio as safe signals, but as debug data together with the internal signals.

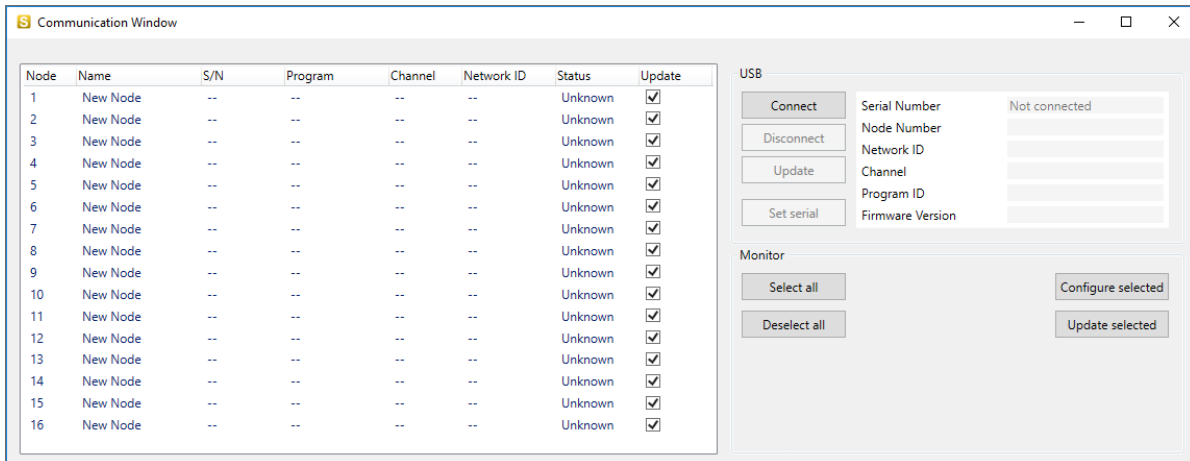
#### **CAN and Online Mode**

Debug data is sent via CAN to other units in the CAN network. However, since debug information is not repeated, the information from CAN only nodes (nodes without radio module) can only be seen when running online mode via USB on those nodes, or via a node that is connected via CAN to them.

## 13.8 Downloading Programs

This chapter is subject to change.

All programs are downloaded from the Communication Window, which can be opened by clicking the Communication button in the toolbar. To open the Communication Window the project must be compiled. Before downloading, the Serial Number of every node in the system must be specified.



*Illustration 13.38 The communication window. All downloading and updating of Simplifiers is done from this view.*

### 13.8.1 Entering serial numbers

There are two ways that a serial number can be entered. The serial numbers can be specified by right clicking the node in the list of nodes on the left side and selecting Edit serial number. It can also be specified by connecting the unit via USB, selecting the node in the list, and pressing the Set serial button. To change an already entered serial number, right click the node and select Edit Serial Number. To change the serial with USB, the already existing Serial Number must be removed. This is done by right clicking the node and selecting Clear serial number.

### 13.8.2 Download via USB

To download programs to a unit via USB, connect your micro USB cable to each unit and update using the communication window. Whenever the program is changed, all units must be updated to be able to communicate with each other.

#### Downloading via USB Notice

When downloading via USB the power supply from the PC is often powerful enough. It is however recommended to use an external power supply, as sometimes the power supply from for example a laptop running on battery is not sufficient.

### 13.8.3 Download via radio with Simplifier Monitor

To download a program to a system with the Simplifier Monitor, first connect the Simplifier Monitor to a USB port on the computer. The first time the program is downloaded to the system, it must be configured with the right settings. This is done by pressing the Configure selected button. After the units are configured, they will be updated automatically. Future updates are done with the Update selected button.

If some units are too far away to be updated, they can be updated individually by selecting them in the list, and pressing the Update selected button.

Running the Configure selected command often resolves most issues with online/downloading.

## 13.9 Passwords

The project file can be password protected. A password protected project file can only be opened if the correct password is entered.

To password protect the project file, open the Password Manager by going to Project→Manage Passwords. Entering a password in the text box and pressing the OK button displays a warning dialog. Clicking OK sets a password for the project.

Note that the project must be saved to password protect the file.

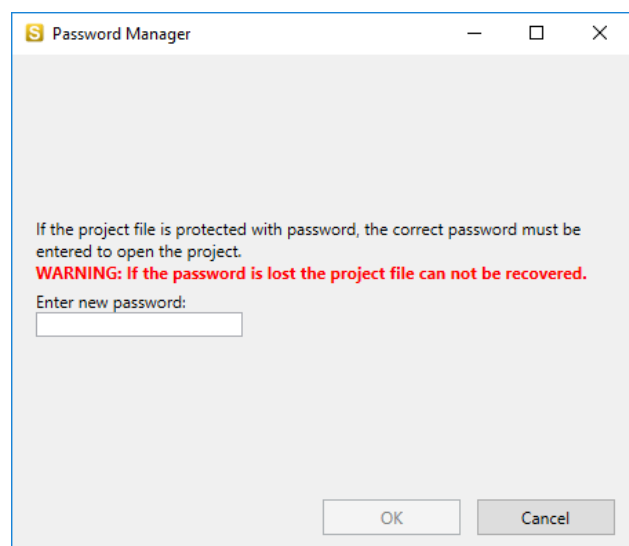
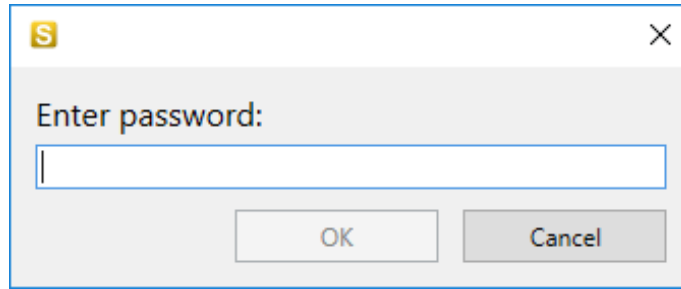


Illustration 13.40: The Password Manager dialog

### 13.9.1 Opening a password protected project file

To open a password protected project file, open the file like usual. If the project file is password protected, a dialog will show, prompting the user to enter the password.



*Illustration 13.41: The dialog for entering passwords to open password protected projects.*

### 13.9.2 Comments

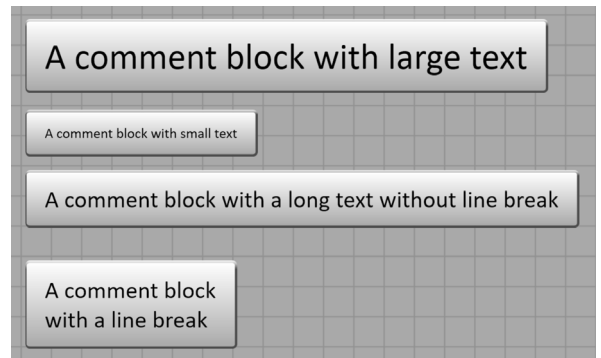
There are two different types of comments; the Comment property of all blocks, or the Comment block.

### 13.9.3 The Comment property

The comment property of all blocks can be changed to add a description for the block. The text is displayed above the block in the graph. The comment property does not change the compiled program in any way.

### 13.9.4 The Comment block

The comment block can be used to add longer descriptions to logic. It does not change the compiled program in any way. See chapter 14.11.1  
To add a comment block, expand the Miscellaneous category in the Toolbox, and drag a Comment block to the logic. The Comment block has 2 properties; Text, and Size. The size can be set between 12 and 40, which changes the size of the text in the block. The Text property sets the text that should be displayed in the block. Pressing the Enter key will create a line break in the text.



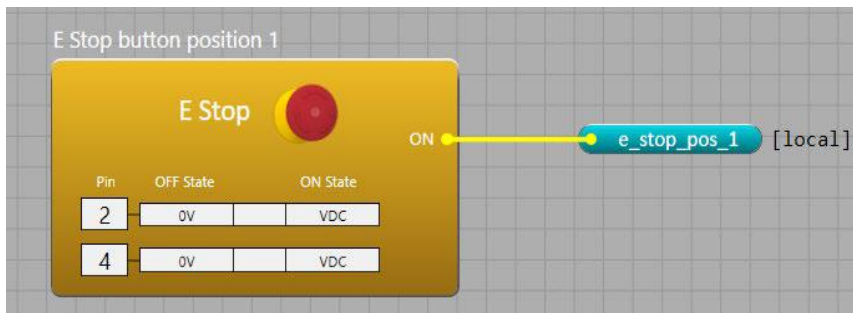
*Illustration 13.42: Comment blocks with different settings*



## 13.10 Designing systems with Radio or CAN communication

### 13.10.1 Radio and CAN communication safety Notice

Since neither CAN nor radio are 100% reliable ways of communication (a CAN cable can be cut off or disconnected at any time, a radio link can be disrupted by radio interference, or a node can simply lose power and stop communicating), the safety applications that use these types of communication must be developed with these problems in mind. Loss and return of radio communication must be considered the same as loss and return of power, and therefore must always result in all affected outputs and equipment going to safe state. Simplifier Manager helps to prevent some common errors when programming, but it is ultimately up to the developer to follow these principles when programming and installing the system. The following chapters discuss how to program systems that communicate via radio or CAN, and the most common errors and how to avoid them.



*Illustration 13.43: An E Stop block with the comment property set to "E Stop button position 1".*

### 13.10.2 Radio

The Safety Simplifier can communicate to other Safety Simplifiers via the optional radio module. The wireless protocol that is used is a special protocol similar to IEEE 802.15.4. This protocol shares the same band as Wi-Fi (2.4GHz), and has the same 16 channels as 802.15.4, numbered 11-26. Channels for Safety Simplifier Systems are numbered 1-16, where the simplifier channels 1-16 correspond to the IEEE 802.15.4 channels as seen in the table below. (Simplifier channel + 10 = IEEE 802.15.4 channel).

Simplifier Channel	IEEE 802.15.4 Channel	Frequency (in MHz)
1	11	2405
2	12	2410
3	13	2415
4	14	2420
5	15	2425
6	16	2430
7	17	2435
8	18	2440
9	19	2445
10	20	2450
11	21	2455
12	22	2460
13	23	2465
14	24	2470
15	25	2475
16	26	2480

Table 13.1: How Simplifier channels correspond to IEEE 802.15.4 channels

To be able to communicate with other nodes, all the nodes must be running the same program, and all nodes must know the serial number of all other nodes in the system. This is to make sure that only the units that are in the system can share information to each other. In Simplifier Manager, before a program can be downloaded, all serial numbers must be specified by the user.

A disturbance in the radio will never cause a safe signal to go high, as a heavy checksum is used on all information. A very strong disturbance for a long time however, can disrupt the system for longer than the specified timeout, which will cause a timeout and affected memories will be set to 0 at the receiving nodes. Timeouts are discussed further in chapter 13.10.5.2.

When communicating via radio, the nodes send their packets in order from node 1 to node 16. A complete update cycle where all nodes have sent a packet once is called a frame. The frame time depends on how many nodes are in the system.

Nodes in system	Frame Time (ms)
2	4.25
3	6.38
4	8.50
5	11.25
6	13.88
7	16.63
8	20.00
9	23.06
10	26.25
11	29.56
12	33.75
13	37.38
14	41.13
15	45.94
16	50.00

Frame time is the minimum response time over radio. The maximum response time is determined by the Long or Short timeout specified in the project. The safety signals will never exceed the timeout specified.

The average response time always depends on the installation and radio environment at that location. Generally, with good radio quality, the average response time can be expected to be somewhere between one or two frame times. However, the radio quality is never constant, and disturbances can cause the average response time to vary over time.

The protocol does not implement any resending/acknowledgement functionality. Instead, all nodes in the system repeat all information they hear to each other, so if a node loses a packet from another node, it still has a chance to hear the repeated information from other nodes in the system. Each node also sends out the age of the information, so other nodes know how old the data is. This is so all the timeout deadlines can be met.

When node 1 starts up it starts sending packets immediately. The other nodes in the system scan all channels listening for information from node 1. If a node hears node 1 directly, or repeated information from node 1 via another node, they will join the network. This means a direct contact with node 1 is not necessary for a node to join the network, but node 1 must be in the network. Turning off node 1 will cause all nodes to stop repeating information from node one and go back to scanning mode. Turning off any other node only stops actions depending on the turned off node.

### 13.10.3 Selecting radio channel for radio communication

The radio channel is selected in the Program Settings dialog in Simplifier Manager. To find the best channel for communication, the built-in channel scanner in Simplifier Manager can be used. Note that this requires a Simplifier Monitor. The channel scanner checks all 16 channels for overlapping Wi-Fi signals and other radio disturbances that would make the channel unsuitable for communication. Without a Simplifier Monitor, a suitable channel can be found by analysing the different Wi-Fi networks in the area to see which channels are occupied. There are several ways to do this. Smartphones running the Android operating system can download a Wi-Fi analysis application to scan the 2.4G band for Wi-Fi networks (this is not available for Apple smartphones). Some programs exist for Windows that do this as well. This method however does not detect other disturbances that can disrupt radio communication. Using the Simplifier Monitor together with the build-in channel scanner is the most reliable way of finding a suitable radio channel.



The Simplifier channels 1-16 do not correspond to the Wi-Fi channels 1-14. Each Wi-Fi channel is 22MHz wide, covering several Simplifier channels.

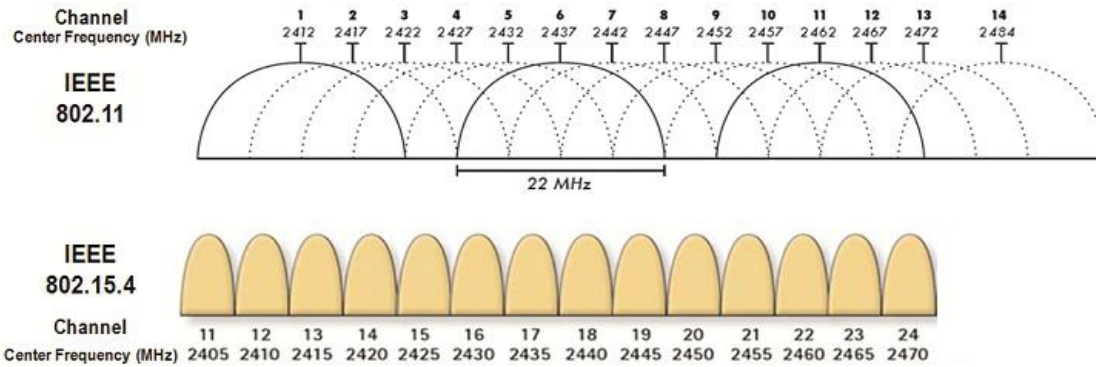


Figure 13.2: Overlap of Wi-Fi (IEEE 802.11) channels and IEEE 802.15.4 channels. Note how for example Wi-Fi channel 6 overlaps 802.15.4 channels 16 through 19 (which means Simplifier channels 6 through 9).

### 13.10.4 Designing systems that communicate via radio

In general, to achieve the best possible radio quality there are three rules:

1. Every node in the system should have good radio connection to at least three other nodes in the system
2. Every node should have good radio connection to the previous node (node 2 should hear node 1, node 3 should hear node 2, etc)
3. Node 1 should have connection to the last node in the system.

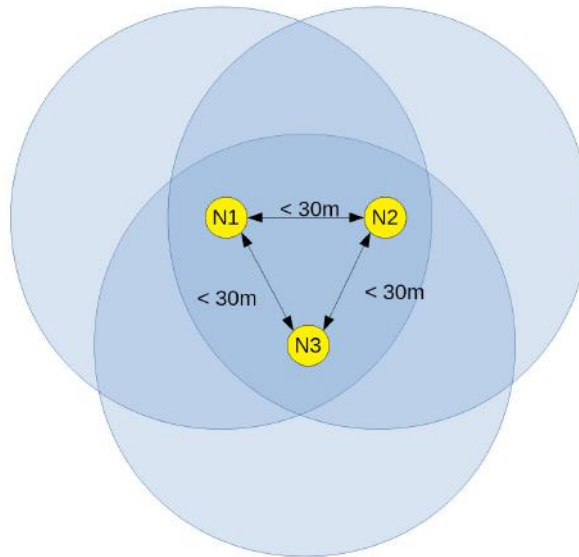
For systems with 2, 3, or 4 nodes, each node should have a good radio connection to every other node in the system.

To achieve a good radio connection between two given nodes, the nodes must be within a certain range of each other. This range depends on walls and other obstructions.

#### 13.10.4.1 Radio quality

For the system in Figure 13.3, a very good radio quality can be expected. All nodes are within range to each other and hear each other directly. This means that no node in the system relies on repeated information.

If a radio disturbance makes for example node 3 miss the information sent by node 1, it still has a chance to hear the repeated information from node 2. This makes the system more resistant (but of course not immune) to random disturbances, and thus more reliable.



*Figure 13.3: An example of a system with good radio quality. Blue circles indicate signal range, arrows indicate direct radio contact. Note that the recommended distance between nodes depend on walls and other obstructions between the nodes.*

In Figure 13.4 nodes 1, 2, and 3 have direct radio contact to each other, and 3, 4, and 5 have direct radio contact to each other. This system will function. However, it violates some of the rules given in the beginning of this chapter; node 1, 2, 4, and 5 only hear two other nodes. This results in a less reliable system. It also violates the rule that node 1 should hear the last node in the system. The result of this is slower reaction times, and less reliability.

**Reliability**

If node 3 loses power or stops radio communication due to an error, no repeated information from node 1 will reach node 4 and 5, and these nodes will leave the network. This makes the network rely completely on node 3 to work.

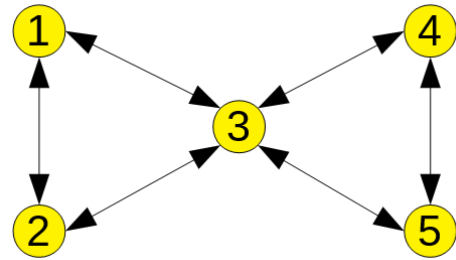
**Reaction Times**

The information from nodes 4 and 5 to nodes 1 and 2 will be delayed by the repeating function of node 3, as it must repeat "backwards". This increases the average reaction times in the system, but it does not increase the maximum reaction time.

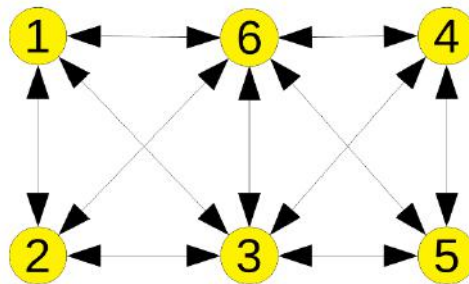
**Solution**

To solve these two issues for the system in Figure 13.4, the system must be changed to follow the rules given at the start of chapter 13.10.4. To achieve this, we can simply add a new node (node 6) and put it next to node 3 (see Figure 13.5).

Now all rules are followed. Node 1 hears the last node in the system (node 6), and every node in the system hears at least 3 other nodes, where one is always the previous node.



*Figure 13.4: A system that relies on the repeating function of node 3. If node 3 loses power, nodes 4 and 5 will lose connection to node 1 and will leave the network.*



*Figure 13.5: This system still relies on the repeating function, but it has redundancy. If node 3 stops transmitting radio, node 4 and 5 will still be able to hear node 1 and 2 via node 6.*

## 13.10.5 Global and CAN Memories

As mentioned in earlier chapters, Global Memories and CAN Memories can be used to send safety information between nodes in a system via Radio or CAN. The Global Memories are sent via both radio and CAN, while the CAN memories are only sent via CAN. A maximum of 16 Global Memories can be defined for each node, and an additional 16 CAN Memories can be defined for each node. This means a maximum of 16 signals can be sent from each node via radio, and a maximum of 32 signals can be sent from each node via CAN. All Global and CAN memories can be used by any node in the system.

### 13.10.5.1 Choosing Memory Numbers

There is no difference between the memories in the radio communication. Either all memories from a node are received, or no memories are received, and all memories are received at the same time. This means that no memory number is better than another, and there are theoretically no rules for assigning memory numbers. There are however some good guides to follow, to make debugging easier.

It is recommended that similar functions are assigned to the same memory number in every node. For example, it is good practice to put all E-Stops on GM1, and all Reset buttons on GM6 for all nodes that have these functions. This makes debugging easier with the display, as the status of all E-Stops is displayed in the same position for all nodes.

It is also recommended that the memory numbers follow the I/O:s. If a Push Button is assigned to terminal 6, it makes sense that it is sent on GM Number 6 as well.

These are only guidelines and cannot always be followed for all applications. The user is free to make their own rules for how memories are assigned, but it should be as consistent as possible, and make sense for the application.

### 13.10.5.2 Timeout and StartUp test

For every Global or CAN memory, apart from Memory Number, two more properties exist; Timeout and StartUp test. These properties are important for the safety communication and determine how the receiving node uses the Memory signal. There is also a third property for Global memories called Do Not Repeat, which is discussed further down in this chapter.

#### Timeout

The response time of a complete function that is sent over radio (Input → Logic → Output) is given by  $T_{input} + T_{logic} + T_{timeout} + T_{logic} + T_{output}$ . The  $T_{timeout}$  time is given by the timeout specified on the memory that is being used to send the signal. Generally, the longer the communication timeout, the more reliable and fault tolerant radio communication between nodes. Setting shorter timeout guarantees shorter maximum response times for safety signals sent over radio, but also results in less robust communication over radio. The question that should determine the timeout values is "What is the maximum theoretical allowed response time for the safety device I am using?". For an E Stop, the maximum response time can be set to higher values (between 500ms to 1000ms is acceptable). For a Light Curtain/ESPE device, the response time requirement is much stricter, so a timeout of 200ms or less can be required. This means that the E Stop would be more resistant to radio disturbances than the ESPE but would also have a longer theoretical maximum reaction time than the ESPE. Reaction time must always be weighed against communication robustness when choosing radio timeouts.

Two different values for timeout can be set for a project; Long and Short. These are default set to Long = 500ms, and Short = 200ms. For each global or CAN memory one of these values can be specified. This is so that safety devices that require shorter timeouts (such as light curtains) can use the short timeout value, and other devices that allow longer reaction times can use the long timeout value. Imagine that a node (A) receives a radio packet from another node (B). Two countdowns start internally in node (A); one for the long timeout and one for the short timeout. If the long timeout is set to 500ms and the short timeout is set to 200ms, the long timer is set to 500ms and the short timer is set to 200ms, and they start counting down immediately. When the timer for short timeout reaches zero all memories from node (B) with short timeout are internally set to zero at the receiver node (A). The same happens when the long timeout reaches zero for the memories with long timeout. Every time a packet is received from node (B) both timers are restarted at their initial values.

Repeated information does not change this behaviour, as the timing information about all data is repeated as well. If a third node (C) sends data to node (B), and node (B) repeats it to node (A), node

(A) sets its countdown timers for node (C) according to when the data was sent, and not when it was received.

The Long and Short times are defined in the Project Settings Window. The Timeout values are used for both the radio and CAN communication. These values determine the maximum reaction time over the radio and CAN link (via radio however, the average reaction time could be much lower, depending on the radio quality and installation).

When calculating response times from an input to an output on another node, the Timeout value specified for the memory must be added to the total reaction time.

It can be tempting to set the Timeout values to very short times, but while a lower Timeout will lower the maximum reaction times on paper, the average reaction time will still be the same. Shorter timeouts will also result in less reliable communication. To achieve the shortest possible reaction times, use fewer nodes in the system, and use CAN instead of radio. For CAN communication the timeout can be set to lower values, since CAN communication is often much more reliable than radio communication. Generally, for systems using only CAN the short timeout can be set to  $2 \cdot NodeCount + 5$  and be used on all memories without any issues.

#### StartUp Test

When a node loses and regains communication to a node with a memory with StartUp test enabled, it must receive an active 0 from the sender before it can set the memory to 1 again. This can be used for example for using Two-Hand devices via two Safety Simplifier. If radio contact is lost and regained while the Two-Hand device is actuated, the receiver will not start the machine again before the Two-Hand device is released and actuated again. Generally, this setting is used for active signals that require an operator to keep high (for example actuating a button or Two-Hand device). StartUp test should also be enabled when a memory is directly controlling an output on another unit. Losing and regaining radio connection stops the machine, and it will not turn on again until an active 0 has been received from the sending node.

#### Do Not Repeat

This option makes the memory not repeated by other nodes, so the signal is only sent directly to the receiver.



### 13.10.5.3 Best Practices

For the best safety communication response times, it is crucial that safety signals only go over radio once. Using a global memory from another node in logic and sending the result on a new memory effectively doubles the reaction time added from radio, as the memory must go via the radio twice.

### 13.10.5.4 Controlling outputs directly via radio or CAN

When using any type of communication, the idea that loss and return of communication must not result in dangerous function must be applied to all simplifier safety systems. When a hold to run device is controlling a safety output directly using a Global or a CAN memory it is crucial to enable the start-up function for the memory. This way one have to release and activate the hold to run device in case of communication or power failure in the Safety Simplifier which uses the memory.

For other safety devices it is not recommended to use the start-up memory for the same action. In case a stop is initiated from communication or power failure in the Safety Simplifier unit controlling a machine function a new start is normally required. This can easily be done by using a reset function in that unit and to have start(reset) buttons in any of or in all the Safety Simplifier units. To give a new start is common requirement after communication or power failure.

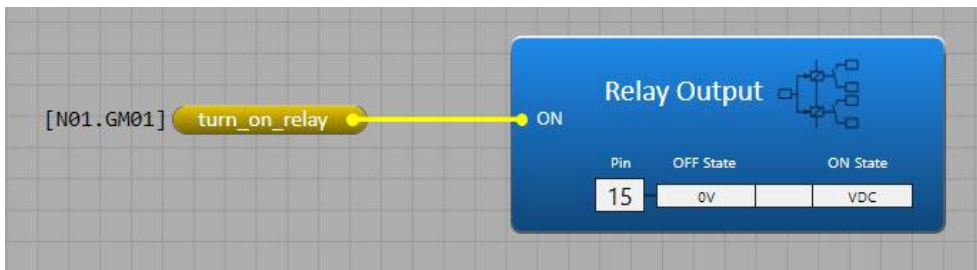


Illustration 13.44: Driving a relay output in node 2 directly using a global memory from node 1.

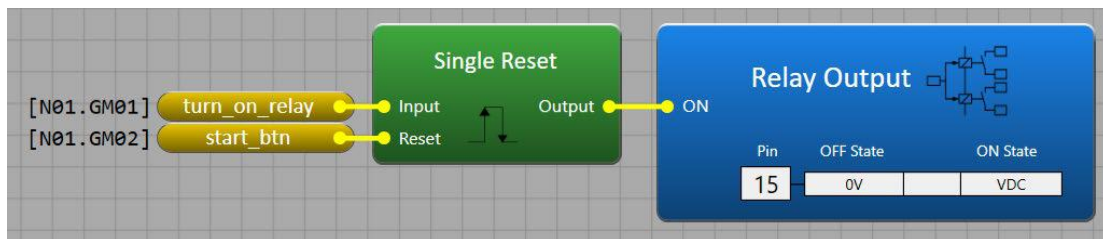


Illustration 15.64: Driving a relay output in node 2 via a reset block using a global memory from node 1.

## 13.11 Input Functions

Input functions in Safety Simplifier are created in a slightly different way from most other PLCs. All input function blocks in Simplifier Manager (E Stop, Gate, Light Barrier, etc.) are based on the same underlying function. Understanding this function is crucial for understanding the different input blocks in Simplifier Manager. With this knowledge, using the Advanced Input, the user can also create their own custom, application specific input functions.

An input can be in one of three so-called "states": ON, OFF, and ERROR. In the OFF state, the ON output is 0. In the ON state the ON output is 1. It is important to know that the ON output will only output a logical 1 when the function is in the ON state. In the ERROR state, the ON output is 0, and the optional ERROR output is 1.

To better understand how input functions work, first we will look at an E Stop input and see how it behaves.

To start with, notice on the E Stop block that each pin has an OFF-State signal and an ON-State signal (signals types are discussed in chapter 13.11.2). When the correct physical OFF state signal is received on all terminals of the Input function, the input goes to OFF state. In this state the block outputs a logical 0 (LOW) signal. When the correct physical ON state signal is received on all terminals of the input function the function goes to ON state. In the ON state, the ON output will output a logic 1. For an E Stop this means that both NC contacts are closed, and the E-Stop is OK.

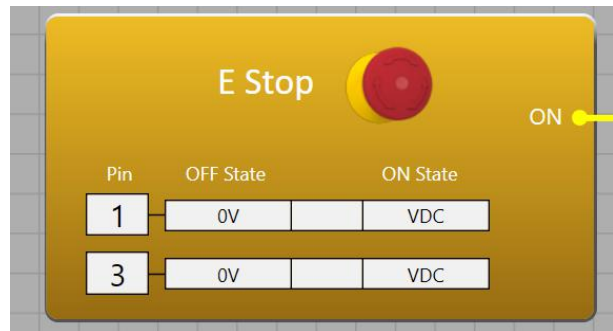
When the E Stop is pressed, the contacts open and pins 1 and 3 read 0V (open contact), which means all pins receive their OFF-state signal, and the function goes to OFF-state. The ON output now outputs a logic 0 (LOW) signal. The ON output only outputs a 1 when the function is in ON state. This is true for all safety input functions.

To go to the ON state, the E-Stop must be pulled out. If both contacts close successfully the pins will read their ON state signal, and the function goes back to ON state. However, if one contact is broken and does not close, one of the terminals receives its ON state signal while the other receives its OFF-state signal, and the function will go to ERROR state (due to a multi-channel error). In the ERROR state, the ON output is 0. If the ERROR output is enabled, it will output a logic 1. The function can never go to ON state from ERROR state. It must first receive a valid OFF state, in this case, 0V on both pin 1 and 3.

The function will go to ERROR state for two different reasons; multi-channel errors, and signal type errors.

A multi-channel error means that two input channels differ for too long (one is in OFF state and one is in ON state).

Signal type errors are when for example the ON state signal configured to be an A-Pulse, but the terminal is receiving constant high 24V signal (VDC).



*Illustration 13.45: An E Stop using terminals 1 and 3, where 0V is the OFF state signal, and VDC is the ON state signal.*

### 13.11.1 Input States

To visualise input states, see the illustrations on this page. The E Stop is configured to use terminals 1 and 3. The ON state signals are VDC for both, so the function expects a high signal to reach ON state.

Pressing and releasing the E-Stop shows which signals are received on the pins, and how the function switches states. A two-channel error can be normally simulated by slowly releasing the E-Stop until only one contact closes (see 12.79: Illustration). The function will go to ERROR state if the terminal receive the wrong signal for longer than the specified Simultaneity time, which for the E Stop block is set to 500ms internally. Whenever the function is in ERROR state the ERROR output will output a logic 1. Releasing the E-Stop completely after the two-channel error, results in both pins receiving their correct ON state signals again. However, the function does not go to ON state, because the function has not gotten a valid OFF state since it was in ERROR state. Pressing the E-Stop opens both contacts and the function receives a correct OFF state. Now when releasing the button, the function can go to ON state.

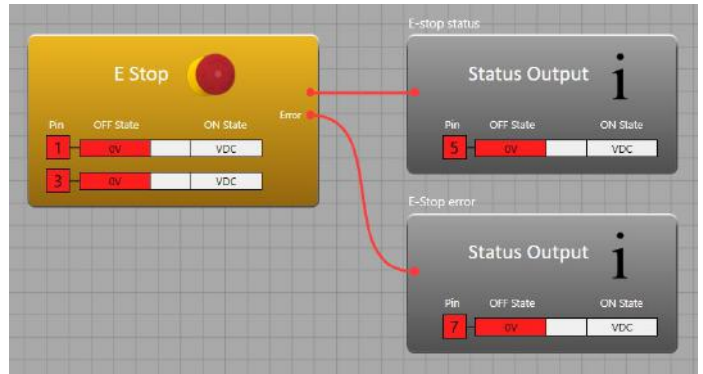


Illustration 13.46: E Stop in OFF state. Terminal 1 and 3 are receiving their correct OFF state signal.

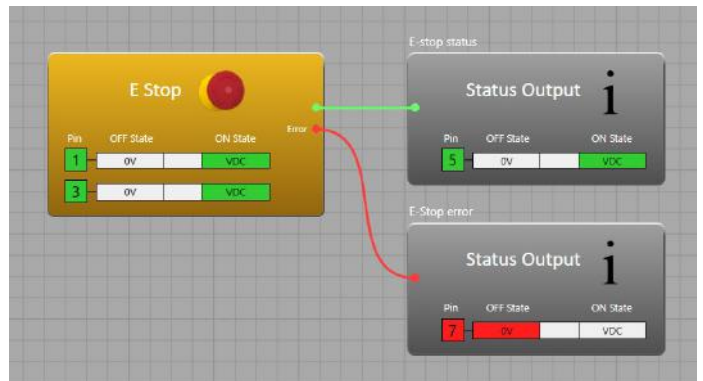


Illustration 13.49: E Stop in ON state (released). Both terminals receive their correct ON state signal (VDC).

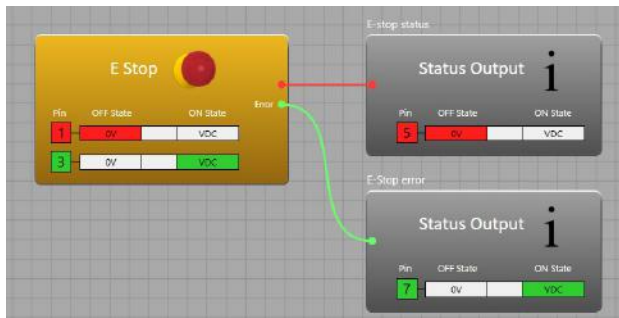


Illustration 13.48: E Stop in ERROR state, due to a two channel error between terminal 1 and 3.

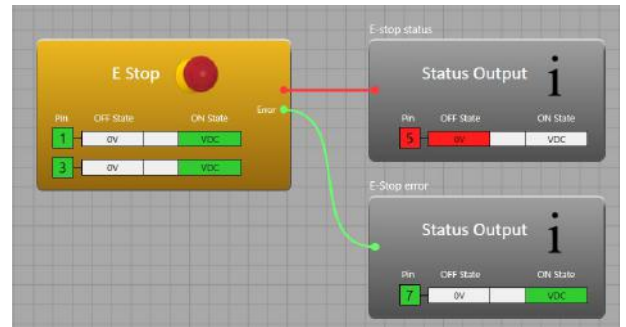


Illustration 13.50: E Stop with a persistent two-channel error. The function cannot go from ERROR state to ON state, even though the correct signals are received on the terminals.

### 13.11.2 Input signal types

A signal type defines the shape of the signal the terminal expects. It can be a constantly high (VDC) or low signal (0V), or pulse, as described later in this chapter. The signal type defined for a terminal determines if the terminal receives its ON state signal or OFF state signal. If the terminal receives a signal that is neither the ON state signal nor the OFF-state signal, the terminal will indicate ERROR. On the LED display the terminal states can be seen on the I/O menu. When a terminal is in OFF state the terminal number LED is red. When a terminal is in ON state the LED is green. When the terminal is in ERROR state the LED it is indicated by blinking orange.

#### Input voltage levels

The signal type does not define the voltage levels that the terminal considers a digital 1 (HIGH) and digital 0 (LOW). The voltage level that is considered a high signal is default set to 80% of the voltage supply to the unit, and the voltage level that is considered a low signal is default set to 20% of the voltage supply to the unit. This is so the same program works if the unit is used with any power supply voltage, for example 12V or 24V. If a specific voltage is required for a certain input, it can be specified directly on that input (in Volt). For example, the unit could be configured with the default values to consider 80% a high signal and 20% a low signal, but an input can be configured to consider 20V a high signal and 10V a low signal. The values specified directly for individual terminals are independent of the supply voltage.

When the input signal goes from low to high or high to low, the logical value is determined based on a hysteresis between the specified voltage values. The signal is logically 1 (HIGH) if the input voltage is above the specified high voltage percentage. The signal goes to 0 (LOW) only when the voltage goes below 20%.

#### 13.11.2.1 VDC

The VDC signal is defined as a constant high signal.

#### 13.11.2.2 OSSD

The OSSD signal is defined as a constant high signal, where short test pulses are allowed.

#### 13.11.2.3 0V

The 0V signal is defined as a constant low signal.



This signal type must never be used by itself as an ON-state signal, as it cannot handle a disconnection on its own. An input function where one terminal is configured to use 0V as ON state, must have at least one more terminal configured to have an active signal as its ON state signal type (VDC, OSSD, any pulse signal).

#### 13.11.2.4 A, B, C, D Pulses

The A Pulse, B Pulse, C Pulse and D Pulse are unique pulse forms that are uniquely distinguishable between each other. The A/B/C/D pulses must be generated by the unit that is using them, thus they cannot be connected between units.

A short circuit between any normal and inverted A/B/C/D pulses or OSSD is detected by all involved inputs and outputs.

#### 13.11.2.5 Inverted A, B, C, D Pulses

The Inverted A, B, C, and D pulses are the exact inverted versions of the A, B, C, and D Pulse described in the chapter above. The pulses must be generated by the unit that is using them, thus they cannot be connected between units.

A short circuit between any normal and inverted A/B/C/D pulses or OSSD is detected by all involved inputs and outputs.

### 13.11.3 Input properties

This chapter describes all the properties that can be changed for an input function. A predefined input function block like E Stop may only use some of these properties. For the Advanced Input function all the properties are available for the user to change.

### 13.11.3.1 Enable ERROR Output

The Enable ERROR Output property can be changed to enable or disable the ERROR output. The ERROR output goes to 1 whenever the function is in ERROR state. It is 0 when the function is in OFF state or ON state. The ERROR signal can be used in logic for indication, or for setting other functions in safe state.

### 13.11.3.2 StartUp Test

If StartUp test is enabled, the function must get a valid OFF state before it can go to ON state after start-up/power on. For example, a Safety Door, enabling StartUp test means the door must be opened and closed each time the system is started power on, to test a complete operation cycle and that all components of the function work as expected.

When start-up function is selected it requires OFF state before ON state. This means OFF state will always be required at power on, after loss and return of power and after loss and return of communication (radio or CAN). For manual hold to run devices such as two-hand control, enabling devices and hold to run PB:s, this is a normal requirement to release before activation.

It is normally not needed to activate this for other safety devices such as E-stops and interlocked doors. If start-up is selected for these devices an E-stop would be required to be pressed and released and the doors would be required to be opened and closed at power on as well as at loss and return of power/communication.

In other cases, to use the start-up function depends on the installation and the risk analysis. For a stop function in a separate Safety Simplifier a start-up function with a reset can be required in case of loss and return of power/communication.

### 13.11.3.3 Filter ON (ms)

Filter On means that the inputs need to be in ON-state this time to go to ON state. The Filter ON property filters the ON state signal for all terminals in this function, when going from OFF state to ON state. The logical signal from this block will be delayed for this time when the function is going from OFF state to ON state. This filter does not affect the transition from ON state to OFF state or ERROR state.

The reason for selecting Filter on is to prevent debouncing contacts from causing error state. If for example one of two contacts debounce an input block could go to Error state. For PB:s a recommendation from the producer can be 10 ms debounce time.

Note that filtering is applied to all terminals of the functions block. This includes terminals that are configured to use pulsed signals as well. The filter is not applied to the actual physical signal, but to the terminal value itself by the function. When a terminal starts receiving its ON state signal, the input function it is part of will not consider that terminal to be in ON state until the Filter ON time has passed.

### 13.11.3.4 Filter OFF (ms)

Filter OFF means that the inputs need to be in OF-state this time to go to OFF state. The Filter OFF property filters the OFF-state signal for all terminals in this function, when going from ON state to either OFF state or ERROR state. The logical signal from this block will be delayed for this time when the function is leaving ON state. For an E Stop this means the reaction time when pressing the E Stop.

Note that filtering is applied to all terminals of the function block. This includes terminals that are configured to use pulsed signals as well. The filter is not applied to the actual physical signal, but to the terminal value itself by the function. When a terminal stops receiving its ON state signal, the input function it is part will keep considering that terminal to be in ON state until the Filter OFF time has passed.

### 13.11.3.5 Enable Simultaneity

This property enables the Simultaneity property. When disabled, no simultaneity is required on the input. When enabled, the Simultaneity (ms) property can be changed (see chapter 13.11.3.6).

### 13.11.3.6 Simultaneity (ms)

The Simultaneity property determines how long two terminals can differ when going from OFF state to ON state. If they are in different states for longer than this time, the input function goes to ERROR state, and waits for a new OFF state (as explained in the end of chapter 13.11.1). For a two-hand device maximum 0.5 seconds is allowed between each hand from OFF to ON-state to fulfil the safety

requirement for two-hand devices. The Simultaneity between two or more sensors is also often used for bypassing of light grids when material shall pass into a safeguarded cell.

For other devices the simultaneity can be useful to detect malfunctions. In an E-stop if one out of two contacts would have a delayed action, this could be detected before it fails completely.

#### **13.11.3.7 Enable Zero Time**

This property enables the Zero Time property, which, when enabled, requires the input to be in OFF state for a certain amount of time before it can go to ON state (see chapter 13.11.3.8).

#### **13.11.3.8 Zero Time (ms)**

The Zero Time property tells the function that the terminals must be in OFF state for longer than this time, before the function can go to ON state again. This is required whenever the function leaves ON state, or when the function is in ERROR state. This property can be used for example with a Two-Hand Device input, where the Two-Hand device must be released for at least this time before it can be used again.

#### **13.11.3.9 Comment**

The Comment property can be changed to add text above the block in the logic graph. This is for giving names or descriptions to blocks and does not affect the compiled program in any way.

#### **13.11.3.10 Terminal Count**

This property sets how many channels this input should have. For most applications, the Terminal Count is set to Single (for inputs like Push Buttons), Double (for safety sensors like Gate Sensors or Light Barriers, or for E Stops), or in some cases Quadruple (for example for Two-Hand devices). For the Advanced Input function, this can be set to a maximum of Octuple (8 channels).

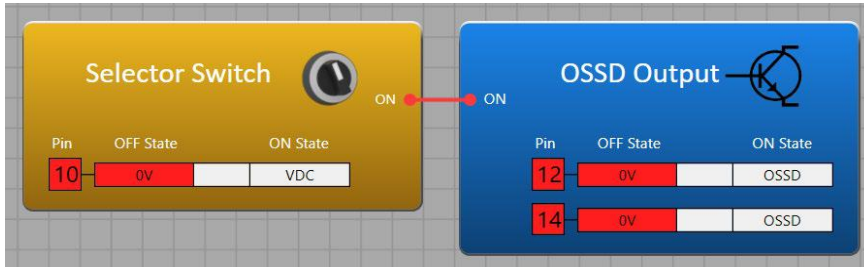
#### **13.11.3.11 Pin Properties**

Each pin has 5 properties; Pin, Signal Type ON, Signal Type OFF, Special Voltage, High Voltage, and Low Voltage. The Pin property decides which terminal number that should be used. The Signal Type ON and Signal Type OFF determines what the ON state and OFF state signals should be for this terminal. Enabling special voltage enables the High Voltage and Low Voltage properties. These determine which voltage levels should be used as a logical high and low signal for this pin. The voltages are independent of the supply voltage.

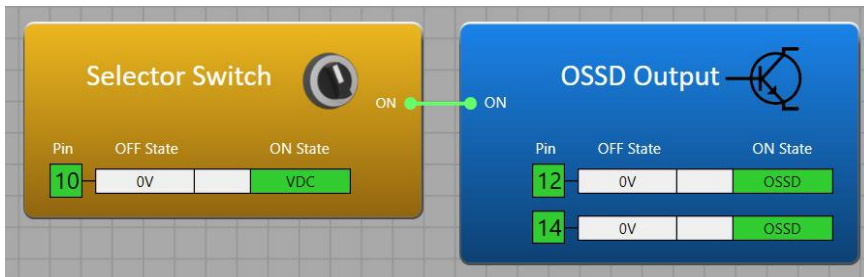
## 13.12 Output Functions

All output functions derive from the same output function. Understanding the output function is important for using any output block. This is also necessary for creating specialized outputs using the Advanced Output function.

The output functions can be in three different states; OFF, ON, and ERROR, and can only go to ON state from OFF state. A logical 1 on the ON Input connector drives the function to ON state, if no external errors have been detected, and if all other criteria are met, which are discussed further in this chapter.



*Illustration 13.51: OSSD Output in OFF state, outputting 0V on pins 12 and 14.*



*Illustration 13.52: OSSD Output in ON state, outputting an OSSD signal on pins 12 and 14.*

When the function is in the OFF state, the terminals try to output the defined OFF state signals on its pins (the OFF-state signals can be any output signal type. All signal types are described in chapter 13.12.3). In most cases this is 0V, however it is possible to have any signal type as OFF state. In the ON state the terminals to output ON state signal.

If the terminal reads a different signal from the one that it is trying to output, it means there is an external error (for example a short circuit), and the function will go to ERROR state.

### 13.12.1 ERROR State

In the ERROR state, the terminals that have detected an error will output 0V, regardless of the configured OFF state signal on that terminal (OFF state could be an active signal). The pins that have not detected an error will output their OFF-state signal. To terminals that have detected an error will keep outputting 0V until the function leaves ERROR state. The function will leave ERROR state when all external faults have disappeared (if the ERROR Reset is not enabled).



Illustration 13.54: OSSD Output in ERROR state due to an external error on terminal 12 (short circuit to 24V).

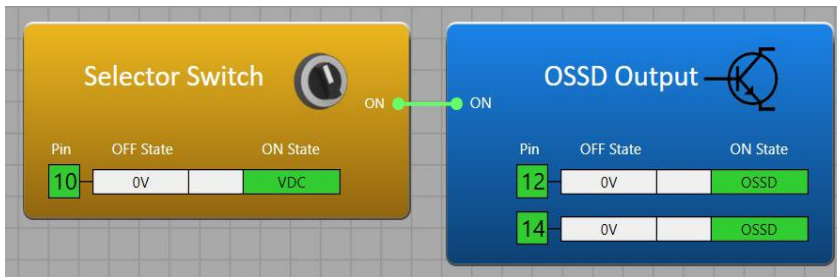


Illustration 13.53: When the external error disappears, the output goes to ON state again.

Enabling the Reset ERROR input changes this behaviour. If the Reset ERROR input is enabled, the function will stay in ERROR state when all external errors have disappeared, until a logical 1 is given on the Reset ERROR input. If the ERROR Reset input is not enabled, the function leaves ERROR state as soon as all external errors have disappeared.

Note that the Reset ERROR input does not require a flank to reset the error. A constant logical 1 is enough.

Note also that the Reset ERROR input cannot be used as a reset function for the ON signal. If no ERROR has been detected, the output will go to ON state as soon as the ON input gets a logical 1, regardless of the signal received on the Reset ERROR input.

Settings the Reset ERROR input to constant logical 1, will result in the same function as disabling the Reset ERROR input (without the Reset ERROR signal, the output is always reset after errors).

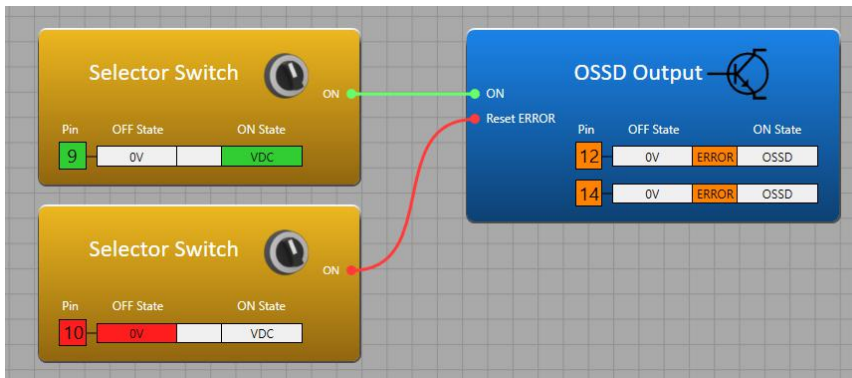


Illustration 13.55: OSSD Output where the Reset ERROR input is enabled. The function will not leave ERROR state until a logical 1 is received on the Reset ERROR input after the external errors have disappeared.



### **13.12.2 Feedback**

When the Feedback input is enabled, the function can only go to ON state when a high flank (0→1) is received on the ON input, and the Feedback input is a logical 1. If the Feedback input is 0, the function will go to ERROR state until the ON signal goes to logical 0, and the error is reset by the Reset ERROR signal (if enabled). If the function is in ON state and an external error is detected. When leaving ERROR state, the function cannot go to ON state again until the ON signal has gotten a logical 0 and then a 1.

Fatal errors can be reset by the touch button the LED panel. If the cause of the fatal error is gone it will not jump into fatal error after reset.

### **13.12.3 Output signal types**

The following chapters describe all available output signal types that can be configured for a terminal.

#### **13.12.3.1 VDC**

The VDC signal is a constant high signal. The output voltage is the power supply voltage to the unit depending on load minus 0.5V. When outputting a VDC signal, the terminal can never detect a short circuit to another type of signal. A short circuit directly to 0V/ground will trigger the over current protection and the unit will go to fatal error state.

#### **13.12.3.2 0V**

The 0V signal is a constant low signal.

#### **13.12.3.3 OSSD**

A terminal configured with OSSD will output an OSSD signal. An OSSD signal will have short interruptions (less than 0,15 ms) to check if there is an external short circuit to the input. During the interruption the output shall go down to 0V. Short circuits to fixed voltage and to other OSSD outputs in the same unit can be detected. OSSD outputs are not recommended for capacitive loads.

#### **13.12.3.4 A, B, C, D Pulses**

The A/B/C/D pulse is a coded signal that can be used as input in the same unit. A short circuit to fixed voltage or to any other A/B/C/D pulse or inverted A/B/C/D pulse in the same unit is detected on all outputs and inputs.

#### **13.12.3.5 Inverted A, B, C, D Pulses**

The inverted A/B/C/D pulse is a coded signal that can be used as input in the same unit. The inverted A/B/C/D pulses are the inverted versions of the A/B/C/D pulses. A short circuit to fixed voltage or to any other A/B/C/D pulse or inverted A/B/C/D pulse in the same unit is detected on all outputs and inputs.

### **13.12.4 The Advanced Output function block**

The Advanced Output block can be used for creating a specialized output function. 1-8 channels can be used and configured freely. The output can use relay output and transistor outputs in the same function. Any signal types can be used for ON and OFF state.

## 13.13 Fatal Errors

Many internal or external failures will result in the unit going to Fatal Error (Safe State). In Fatal Error state, the unit does not communicate via radio or CAN, and all I/O are set to 0V (not OFF state). The lower 6 LEDs on the LED panel will glow red, and the numbered LEDs will display an error code. The code is binary coded on the display. The unit can only leave Fatal Error state by power cycling the unit. A fatal error can also be triggered from inside logic, for certain signals (for example, an error on a relay output could set the entire unit in fatal error (Safe State)). This is done using the Fatal Error block. There are a total of 256 different error codes that can be selected for each node.

## 13.14 Logic

The logic is guaranteed to be executed every millisecond in a unit ( $\pm 1\%$ ). This means the time resolution for all logic is 1 millisecond. This is also what the unit uses to keep track of timings in the logic. This means that delays and times in the logic will never de-sync (See **Error! Reference source not found.**).

### 13.14.1 Logical Loops

A logic loop happens when a signal that is defined later in the logic chain is used earlier in the chain. Or in other words, a function input signal depends on the function output signal (commonly known as feedback). Since the output signal value is unknown before using the inputs, the previous value of the output signal will be used. This results in one logic cycle delay of the output signal (per loop). This can be done in logic when using Local memories but is not allowed with regular connections. Since the Local memory is only a reference to a signal it will not create loops by default. Only if the local memory is referenced in the chain before it is defined will a loop be created.



Creating loops in the logic can cause unintended behaviour of the logic and will cause higher maximum response times and different PFH-d values.

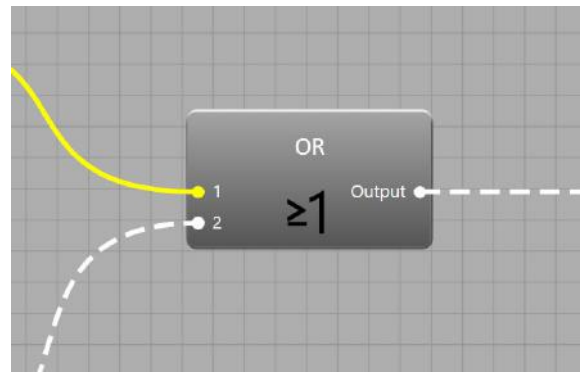
---

### 13.14.2 Unsafe and safe signals

In the logic there are two types of signals; Unsafe signals, and safe signals. The safe signals are the only signals that can be used for safety applications. These can drive safety outputs high. Safety signals always originate from a safe source, like a safety input, or an internally generated signal (Logic 1 or Logic 0). Note that the signal is not guaranteed to be logically safe. The developer of the application is always responsible to ensure that the program logic is safe. For example, a Logic 1 (constantly high signal) is considered safe by the program but driving a relay output with the signal directly is not safe, as there is no safety devices that can turn the output off (this can be seen also as an infinite reaction time).

An unsafe signal is a status / info signal, that cannot be used for safety applications. Unsafe signals can for example drive a Status Output, turn on a Push Button Lamp, or be sent over a serial bus to other PLCs. Whenever an unsafe signal in any way affects a safe signal, the resulting output becomes an unsafe signal. For example, connecting a safe signal and an unsafe signal to an OR block, makes the OR block unsafe, and its output is unsafe (see Illustration 13.58).


If there is a safety application with an output that should (logically) be able to be turned off using an unsafe signal, the Unsafe & Safe block can be used. This block is an AND function between an unsafe signal and a safe signal. All safety must come from the safe input of the block but turning off the unsafe input can also turn off the output. See chapter 14.9.1 on page 144 for a description on how to use the block.



*Illustration 13.56: A safe and an unsafe signal affecting an OR block, making the block unsafe (indicated by the block becoming grey, and its output becoming unsafe).*

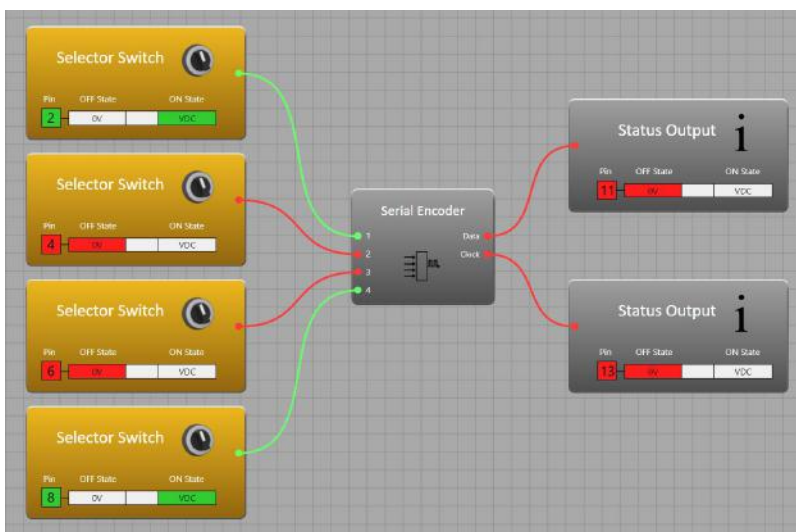
## 13.15 Serial Communication for status information

The Safety Simplifier comes with two function blocks; the Serial Encoder and the Serial Decoder, for synchronous serial communication via the I/O terminals on the Simplifier. Up to 32 signals can be sent and received with each block. The blocks must be configured correctly to encode and decode the information. The configured terminals can then be connected to another Simplifier, or another PLC.

 <b>Warning!</b>	<p>The serial information is not safety information, as indicated by the grey colour of the function block in Simplifier Manager. These signals are only intended to be used for status information and should never be used for any safety function.</p>
--	---

### 13.15.1 Serial Encoding

The Serial Encoder takes 3-32 input signals and encodes them to two outputs; Data and Clock. These signals can then be connected to two Status Outputs and connected externally to another PLC via the terminals on the Simplifier.



*Illustration 13.57: An example of how to connect the Serial Encoder. The terminals 11 and 13 are freely configurable, and any terminal (1-14) can be used.*

### 13.15.1.1 Format

The data is sent least significant bit first, and most significant bit last (the bit on input 1 is sent first, then input 2, then input 3 etc.). The data output changes value on clock high flank and is clocked on clock low flank (as shown in Figure 13.6).

The data is sent in "packets" of bits. The packets are separated by the time given by Inter-character Duration. During this time both Data and Clock are low. There is always an extra time (equal to the Half Bit Time) after the Inter character duration before each packet. This lets the Serial Decoder block use the same Inter character duration setting as the Serial Encoder.

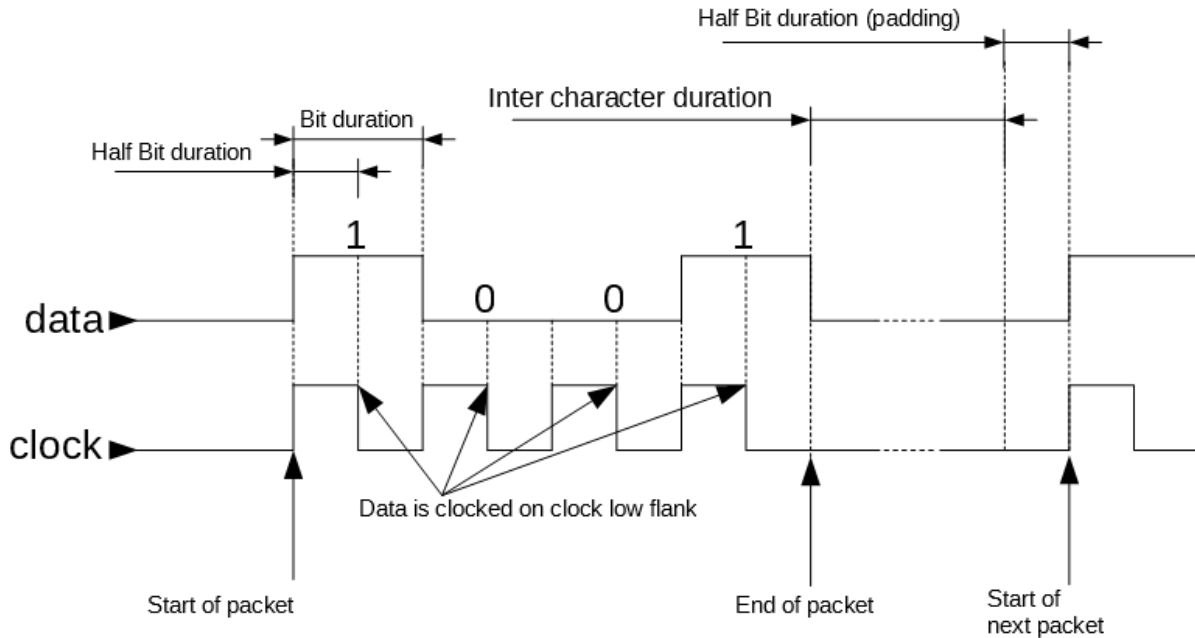


Figure 13.6 Serial Encoder timing diagram

The Serial Encoder has three properties; Inter-character Duration (ms), Half Bit Duration (ms), and Input Count.

#### Inter-character Duration

This is the time the clock and data signals will be low between each packet of data. This is so the receiver knows when one packet ends and the next begins. This must be greater than the Half Bit Duration.

#### Half Bit Duration

This is how long the clock signal is high for each bit sent out.

#### Input Count

This sets how many signals will be encoded. Up to 32 signals can be encoded.

### 13.15.2 Serial Decoding

The Serial Decoder can be configured to decode a signal encoded by a Serial Encoder from another Simplifier, or from another PLC. The Serial Decoder has three properties; Inter-character Duration (ms), Timeout (ms), and Output Count. The Inter-character duration tells the function how much time that should indicate a new packet.

#### Inter-Character Duration

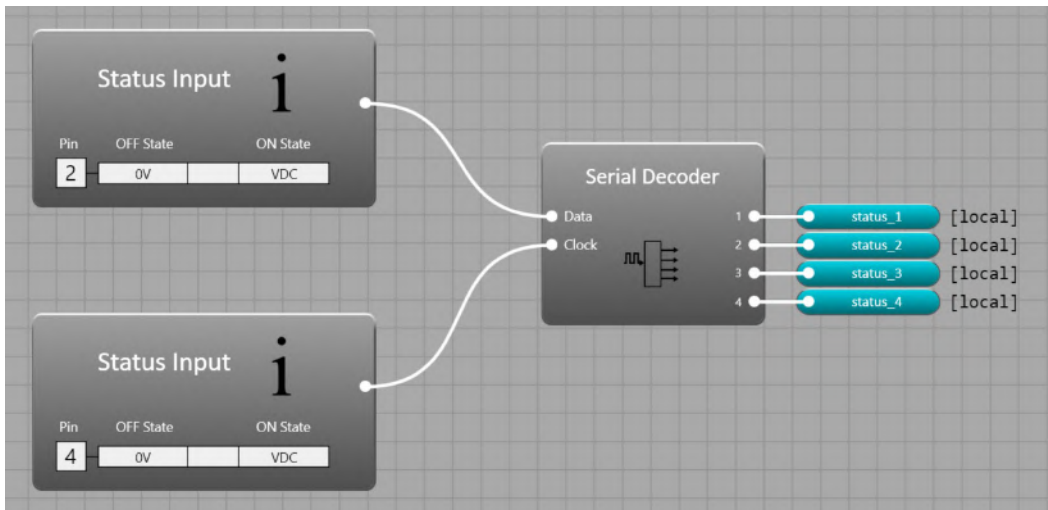
The intercharacter duration should be set to the same value as the intercharacter duration of the Serial Encoder that is encoding the signal.

#### Timeout

The timeout can be set to 1000 ms (1 second), or 10 times the Inter-Character duration (whichever is highest). If the clock-signal is low for this time (for example if the cable is disconnected), all the outputs from the Serial Decoder will be set to 0.

#### Output Count

This is how many signals the encoded data contains. Select the same amount as inputs on the connected Serial Encoder.



*Illustration 13.58: An example of how to connect a Serial Decoder in Simplifier Manager. The Data and Clock signals are externally connected to the data and clock outputs from the sending unit.*

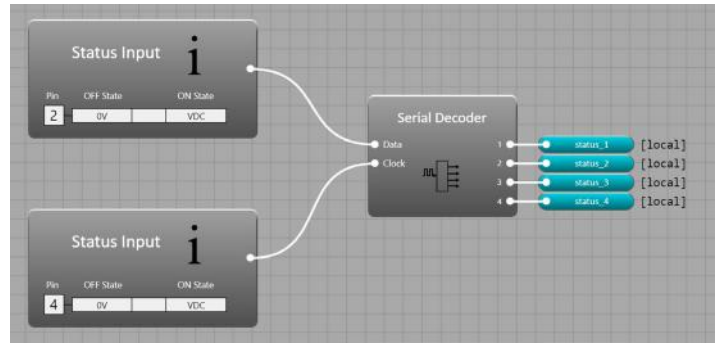
### 13.15.3 Serial communication via I/O Terminals

To send data via the I/O terminals to other Simplifiers, the properties of both the Serial Encoder and the Serial Decoder can be left as they are. The default values for these properties are optimized for communication between Simplifiers via I/O.

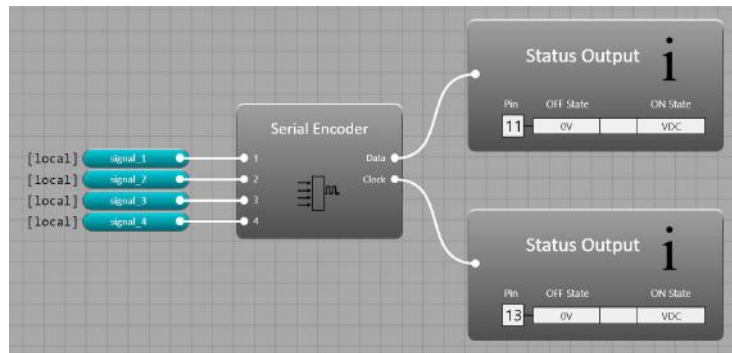
Input Count and Output Count can be set to however many signals that should be sent.

The output terminal connected to the Data output of the Serial Encoder, should be connected externally to the input terminal connected to the Data input of the Serial Decoder on the receiving Simplifier. The same should be done for the Clock signal, as seen in Illustration.

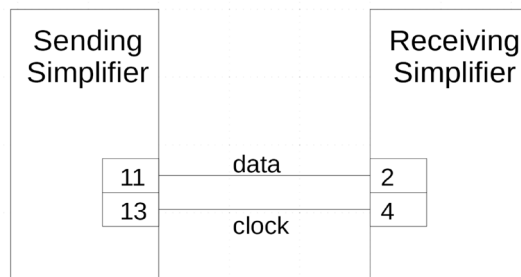
Note that any transistor IO can be used. For this example, the sender is using terminal 11 for the data output and terminal 13 for the clock output. The receiver is using terminal 2 for the data input and terminal 4 for the clock input.



*Illustration 13.59: An example of a receiving node. Terminal 2 has been configured as the Data terminal, and terminal 4 has been configured as the Clock terminal.*



*Illustration 13.60: An example of a sending node. Terminal 11 has been configured as the Data output terminal, and terminal 13 has been configured as the Clock output terminal.*



*Illustration 13.61: How to externally connect to simplifiers that communicate serial information via IO Terminals. Note that the terminal numbers are arbitrary. Any transistor IO can be used.*

## **13.15.4 Configuration for communication with other PLCs**

### **13.15.4.1 Receiving signals from other PLCs**

To receive serial data from other PLCs, the Serial Decoder can be used. See the documentation for the other PLC for details on how the encoding function works for that PLC.

### **13.15.4.2 Sending signals to other PLCs**

To send serial data to other PLCs, the Serial Encoder can be used. See the documentation for the other PLC for details on how the decoding function works for that PLC. Make sure the receiving PLC is configured correctly to receive the signals.

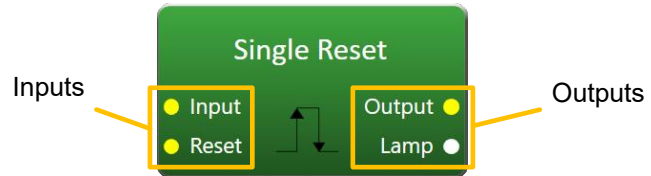
### **13.15.4.3 Function blocks for communication with PLCs**

Function blocks for serial communication with other PLC systems (for example Siemens, Beckhoff and Rockwell) are available for download on our homepage.

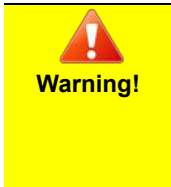


## 14 Function Block Reference

This chapter describes the function of every function block in simplifier manager. Each block is described by specifying its inputs, outputs, and properties. Inputs are signals that the block needs to perform its function (see Figure 14.1). These are the signals that the function block will perform its function on. Outputs are signals that are generated by the block, that can be used as inputs for other blocks.



*Figure 14.1 The inputs and outputs of a function block. Note that the yellow inputs and outputs are for safety signals, and the white inputs and outputs are for non-safety signals.*



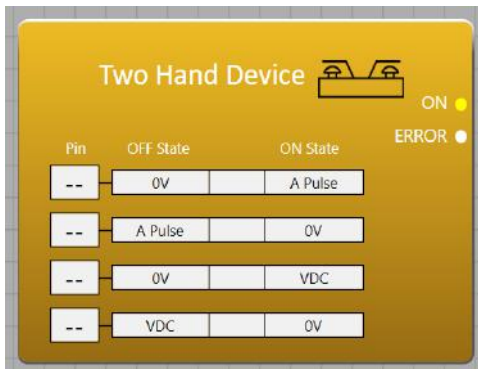
All blocks that are coloured grey are “non-safety” blocks, meaning they must never be used for safety applications. These blocks are intended for status and indication only.

There are two types of internal signals that can be used: safe and unsafe. Safe signals are coloured solid yellow and can be used in safety functions. Unsafe signals are dotted white and must never be used for safety functions.

## 14.1 Inputs

### 14.1.1 Two-Hand Device

Block diagram



#### Description

This block can be used for a Two-Hand device according to EN 574 input.

Pins: 2 or 4

Inputs

None

Outputs

- **ON**  
This output is 1 (HIGH) when the function is in ON state, which is when all pins receive their configured ON-state signal type and all timing requirements are met. It is 0 (LOW) when any criteria for the function to be in ON state are not met.
- **ERROR**  
This output is 1 (HIGH) when the function is in ERROR state.

#### Properties

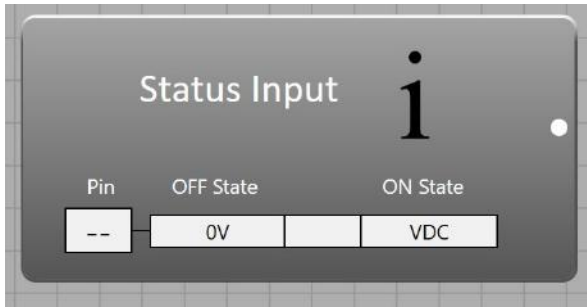
Properties	
Comment	<input type="text"/>
Enable ERROR Output	<input type="checkbox"/>
StartUp Test	<input checked="" type="checkbox"/>
Filter ON (ms)	<input type="text" value="10"/>
Simultaneity (ms)	<input type="text" value="450"/>
Enable Zero Time	<input checked="" type="checkbox"/>
Zero Time (ms)	<input type="text" value="100"/>
Terminal Count	<input type="text" value="Quad NC-NC"/>

- **Comment**  
Changes the comment text above this block.
- **Enable ERROR Output**  
Enables or disables the ERROR output connector.

- **StartUp Test (required for two-hand)**  
This property enables or disables the StartUp test check. If StartUp Test is enabled, the function cannot go to ON state directly at start-up but needs a valid OFF state first.
- **Filter ON (ms)**  
Filters the 0→1 flank of pins, so they must be in ON state for longer than this time.
- **Simultaneity (ms)**  
All pins must go to ON state within this time for the function to go to ON state. If this criterion is failed the input needs all pins to go to OFF state before a new attempt can be made.
- **Enable Zero Time**  
Enables or disables the Zero Time property.
- **Zero Time (ms)**  
If enabled, requires all the pins to be in OFF state for this time before the function can go to ON state.
- **Terminal Count**  
Changes the input function between Quad NC-NO and Double NO.
- **Pins**  
These settings can be changed for each pin:
- **Pin**  
The pin number to use for this pin input.
- **Signal Type ON**  
Which signal type should be considered as ON state.
- **Signal Type OFF**  
Which signal type should be considered as OFF state.

## 14.1.2 Status Input

### Block Diagram



### Description

This block can be used to evaluate a status signal input and use it in logic.

Pins: 1

Inputs

None

Outputs

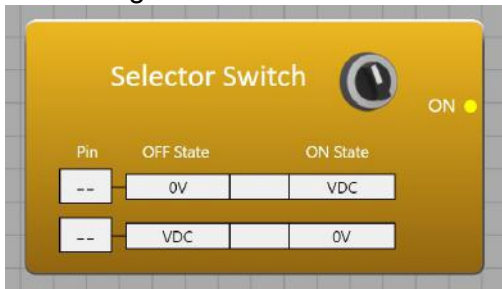
- ON  
This output turns on when a logical HIGH analogue signal is received on the configured pin.  
**Note** that this output is an unsafe signal, and thus cannot affect any safety outputs.

### Properties

- Comment  
Changes the comment text above this block.
- Terminal Number  
The pin number to use.

### 14.1.3 Selector Switch

#### Block Diagram



#### Description

This block can be used to evaluate a selector switch input. The switch can be of different types: Single NO, Double NO, Double NC/NO.

Pins: 1-2

Inputs

None

Outputs

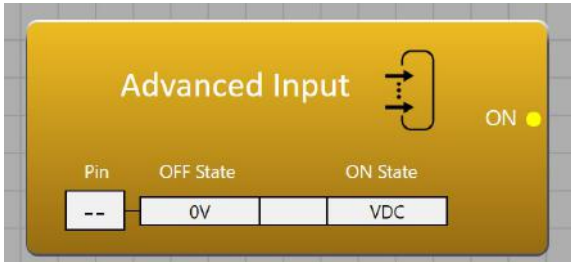
- ON  
This output is 1 (HIGH) when the selector switch is in the state.
- ERROR  
This output is 1 (HIGH) when the function is in ERROR state, and 0 (LOW) if the function is in ON or OFF state.

#### Properties

- Comment  
Changes the comment text above this block.
- Enable ERROR Output  
Enables the ERROR output.
- Filter ON (ms)  
Filters the 0→1 flank of pins, so they must be in ON state for longer than this time.
- StartUp Test  
This property enables or disables the StartUp test check. If StartUp Test is enabled, the function cannot go to ON state directly at start-up (power on) but needs a valid OFF state first.
- Input Type  
Changes which input type this block is (Single NO, Double NO, Double NC/NO)

## 14.1.4 Advanced Input

### Block Diagram



### Description

This block can be used to create a custom input function to fit a specific device or application. The input function is described in chapter 13.11 Input Functions on page 89.

Pins: 1-8

### Inputs

None

### Outputs

- **ON**  
This output is 1 (HIGH) when the function is in ON state, and 0 (LOW) when the function is in OFF state or ERROR state.
- **ERROR**  
This output is 1 (HIGH) when the function is in ERROR state, and 0 (LOW) if the function is in ON or OFF state.

### Properties

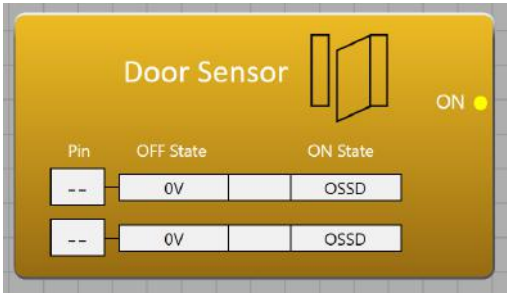
Properties	
Comment	<input type="text"/>
Enable ERROR Output	<input type="checkbox"/>
StartUp Test	<input type="checkbox"/>
Filter ON (ms)	<input type="text" value="0"/>
Filter OFF (ms)	<input type="text" value="0"/>
Enable Simultaneity	<input type="checkbox"/>
Enable Zero Time	<input type="checkbox"/>
Channel Count	Single <input type="button" value="v"/>
▼ Pins	
Pin --	
Pin	-- <input type="button" value="v"/>
Signal Type ON	VDC <input type="button" value="v"/>
Signal Type OFF	0V <input type="button" value="v"/>
Special Voltage	<input type="checkbox"/>

- **Comment**  
Changes the comment text above this block.
- **Enable ERROR Output**  
Enables the ERROR output connector.

- **StartUp Test**  
This property enables or disables the StartUp test check. If StartUp Test is enabled, the function cannot go to ON state directly at start-up (power on) but needs a valid OFF state first.
- **Filter ON (ms)**  
Filters the 0→1 flank of pins, so they must be in ON state for longer than this time.
- **Simultaneity (ms)**  
All pins must go to ON state within this time for the function to go to ON state. If this criterion is failed the input needs all pins to go to OFF state before a new attempt can be made.
- **Enable Zero Time**  
Enables or disables the Zero Time property.
- **Zero Time (ms)**  
If enabled, requires all the pins to be in OFF state for this time before the function can go to ON state.
- **Terminal Count**  
Sets the terminal count between 1 and 8.
- **Pins**  
These settings can be changed for each pin:
- **Pin**  
The pin number to use for this pin input.
- **Signal Type ON**  
Which signal type should be considered as ON state.
- **Signal Type OFF**  
Which signal type should be considered as OFF state.

## 14.1.5 Door Sensor

### Block Diagram



### Description

This block can be used to evaluate a door or gate sensor dual input.

Pins: 2

Inputs

None

Outputs

- ON  
This output is 1 (HIGH) when both pins receive their ON state signal.
- Outputs

### Properties

Properties	
Comment	<input type="text"/>
Enable ERROR Output	<input type="checkbox"/>
StartUp Test	<input type="checkbox"/>
Filter ON (ms)	<input type="text" value="10"/>
▼ Pins	
Pin --	
Pin	<input type="text" value="--"/>
Signal Type ON	<input type="text" value="OSSD"/>
Signal Type OFF	<input type="text" value="0V"/>
Pin --	
Pin	<input type="text" value="--"/>
Signal Type ON	<input type="text" value="OSSD"/>
Signal Type OFF	<input type="text" value="0V"/>

- Comment  
Changes the comment text above this block.
- Enable ERROR Output  
Enables the ERROR output connector.
- StartUp Test  
This property enables or disables the StartUp test check. If StartUp Test is enabled, the function cannot go to ON state directly at start-up, (power on) but needs a valid OFF state first.
- Filter ON (ms)  
Filters the 0→1 flank of pins, so they must be in ON state for longer than this time.



## 14.1.6 Light Barrier

### Block Diagram



### Description

This block can be used to evaluate a Light Barrier/ESPE input device.

Pins: 2

Inputs

None

Outputs

- **ON**  
This output is 1 (HIGH) when the function is in ON state. This output is 0 (LOW) when the function is in OFF or ERROR state.
- **ERROR**  
This output is 1 (HIGH) when the function is in ERROR state.

### Properties

Properties	
Comment	<input type="text"/>
Enable ERROR Output	<input type="checkbox"/>
StartUp Test	<input type="checkbox"/>
Filter ON (ms)	<input type="text" value="10"/>
▼ Pins	
Pin --	
Pin	-- ▼
Signal Type ON	OSSD ▼
Signal Type OFF	0V ▼
Pin --	
Pin	-- ▼
Signal Type ON	OSSD ▼
Signal Type OFF	0V ▼

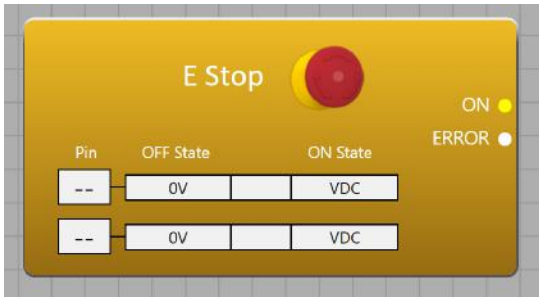
- **Comment**  
Changes the comment text above this block.
- **Enable ERROR Output**  
Enables the ERROR output connector.
- **StartUp Test**  
This property enables or disables the StartUp test check. If StartUp Test is enabled, the

function cannot go to ON state directly at start-up, (power on) but needs a valid OFF state first.

- Filter ON (ms)  
Filters the 0→1 flank of pins, so they must be in ON state for longer than this time.

## 14.1.7 E Stop

### Block Diagram



### Description

This block can be used to evaluate an E-stop input. The E-stop can be of different types: Single NC, Double NC, Double NC/NO.

Pins: 1-2

### Inputs

- Comes from the pins/terminals

### Outputs

- ON  
This output is 1 (HIGH) when the function is in ON state. This output is 0 (LOW) when the function is in OFF or ERROR state.

### Properties

Properties	
Comment	<input type="text"/>
Enable ERROR Output	<input checked="" type="checkbox"/>
StartUp Test	<input type="checkbox"/>
Filter ON (ms)	<input type="text" value="10"/>
Terminal Count	<input type="text" value="Double NC"/>

- Comment  
Changes the comment text above this block.
- Enable ERROR Output  
Enables the ERROR output.
- Filter ON (ms)  
Filters the 0→1 flank of pins, so they must be in ON state for longer than this time.
- StartUp Test  
This property enables or disables the StartUp test check. If StartUp Test is enabled, the function cannot go to ON state directly at start-up (power on) but needs a valid OFF state first.
- Input Type  
Changes which input type this block is (Single NC, Double NC, Double NC/NO)

## 14.1.8 Push Button

### Block Diagram



### Description

This block can be used to evaluate a single input push button with an indication output on the same pin (for example a reset/start push button with indication). This is accomplished by quickly switching between using the transistor IO as an input and an output (1.5ms output, 0.5ms input).



Note

See chapter 11.1 on page 35 for electrical drawing for reset function.

Pins: 1

### Inputs

- **Lamp**  
This input can be used to turn on the indication on the physical button.

### Outputs

- **ON**  
This output is 1 (HIGH) when the function is in ON state (when the button is pressed). This output is 0 (LOW) when the function is in OFF state.

### Properties

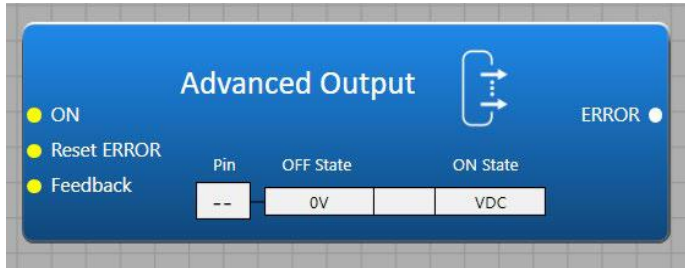
Properties	
Comment	<input type="text"/>
Enable Lamp Input	<input checked="" type="checkbox"/>
Filter (ms)	<input type="text" value="10"/>
Terminal Number	<input type="text" value="--"/>

- **Comment**  
Changes the comment text above this block.
- **Enable Lamp Input**  
This enables or disables the Lamp input.
- **Filter ON (ms)**  
This filters the 0→1 flank of the input for the specified amount. This means the pin must be high for longer than the specified Filter ON time for the function to go to ON state.
- **Terminal Number**  
Selects the pin that should be used.

## 14.2 Output

### 14.2.1 Advanced Output

#### Block Diagram



#### Description

This block can be used to create a specialized output function. 1-8 pins can be used, and any valid combination of signal types can be used. This output can also combine transistor outputs and relay outputs in the same function (an error on the transistor output will turn off the relay output).

Pins: 1-8

#### Inputs

- **ON**  
A logical 1 (HIGH) signal on this input tries to turn on the output pins so they output their ON state signal, unless some other criteria fail.
- **Reset ERROR**  
If this input is enabled, the function does not leave ERROR state until a logical 1 has been received on this input. There are no flank requirements on the reset signal; as soon as a logical 1 (HIGH) signal is received the function will leave ERROR state.
- **Feedback**  
If enabled, this signal must be logical 1 (HIGH) when the ON signal goes high to turn on the output. If this signal is 0 (LOW) when the ON signal turns on, the output function goes to ERROR state.

#### Outputs

- **ERROR**  
This output is 1 (HIGH) when the function is in ERROR state.

#### Properties

- **Comment**  
Changes the comment text above this block.
- **Enable ERROR Output**  
Enables the ERROR output.
- **Enable Feedback**  
Enables the Feedback input.
- **Enable Error Reset**  
Enables the Error Reset input.
- **Channel Count**  
Specifies how many terminals should be used in this function.

## 14.2.2 OSSD Output

### Block Diagram

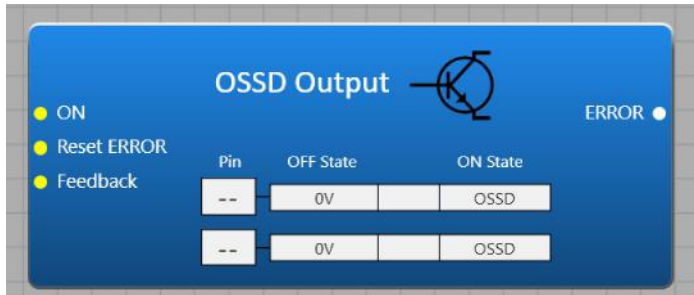


Figure 14.2 Block diagram for OSSD output with ERROR output, Reset ERROR and Feedback enabled.

### Description

The OSSD Output can be used to output OSSD signals on any of the 14 transistor IO. An OSSD signal from any pin detects a short circuit between an OSSD signal on any other pin and to external fixed voltage.

Pins: 1-8



Electrical diagrams for IO connections can be found in chapter 11.4 (page 36) and chapter 11.8 (page 38).

### Inputs

- **ON**  
A logical 1 (HIGH) signal on this input tries to turn on the output pins so they output their ON state signal, unless some other criteria fail.
- **Reset ERROR**  
If this input is enabled, the function does not leave ERROR state until a logical 1 has been received on this input. There are no flank requirements on the reset signal; as soon as a logical 1 (HIGH) signal is received the function will leave ERROR state.
- **Feedback**  
If enabled, this signal must be logical 1 (HIGH) when the ON signal goes high to turn on the output. If this signal is 0 (LOW) when the ON signal turns on, the output function goes to ERROR state.

### Outputs

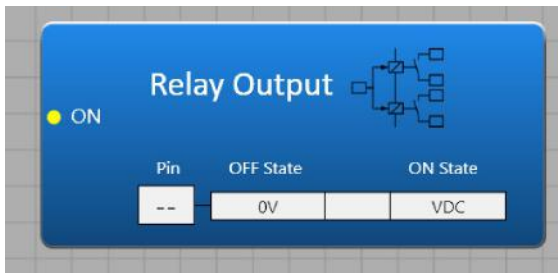
- **ERROR**  
This output is 1 (HIGH) when the function is in ERROR state.

### Properties

- **Comment**  
Changes the comment text above this block.
- **Enable ERROR Output**  
Enables the ERROR output.
- **Enable Feedback**  
Enables the Feedback input.
- **Enable Error Reset**  
Enables the Error Reset input.
- **Channel Count**  
Specifies how many terminals should be used in this function.

## 14.2.3 Relay Output

### Block Diagram



### Description

This block can be used to configure a relay output on an S16 unit. Technical data for relays can be found in chapter 4 Technical data Safety Simplifier on page 23.

Pins: 1 (relay terminal 15 or 16)



Note

Electrical diagrams for IO connections can be found in chapter 11.6 (page 37) and chapter 11.7 (page 38).

### Inputs

- **ON**  
A logical 1 (HIGH) signal on this input tries to turn on the output pins so they output their ON state signal, unless some other criteria fail.
- **Feedback**  
If enabled, this signal must be logical 1 (HIGH) when the ON signal goes high to turn on the output. If this signal is 0 (LOW) when the ON signal turns on, the output function goes to ERROR state.

### Outputs

- **ERROR**  
This output is 1 (HIGH) when the function is in ERROR state.

### Relays outputs contacts 15:

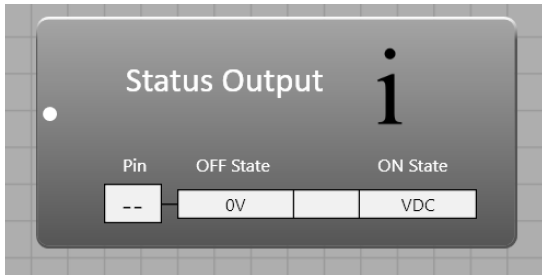
- Relay 15, 1NO pin 23-24 and 1NO pin 33-34 (both must be used for a redundant function)
- Relays output contacts, 1NO pin 43-44 and 1NO pin 53-54 (both must be used for a redundant function)

### Properties

- **Comment**  
Changes the comment text above this block.
- **Enable Feedback**  
Enables or disables the Feedback input.
- **Relay Number**  
Configures which relay should be used (15 or 16).

## 14.2.4 Status Output

### Block Diagram



### Description

This block can be used to output status signals on a transistor IO.

This block is for status/non-safety/indication only and cannot be used for any safety applications.

Pins: 1



This block is intended for stats/information purposes only and must not be used for safety functions.

### Inputs

- ON  
A logical 1 (HIGH) signal on this input turns the output ON, outputting a constantly high signal (VDC).

### Outputs

None.

### Properties

- Comment  
Changes the comment text above this block.
- Terminal Number  
Which terminal to use.



## 14.3 Memory

### 14.3.1 Memory

#### Block Diagram



#### Description

The memory block can be used to send signals internally (with local memories) or via radio/CAN to other nodes (using Global/CAN memories). Changing the memory type changes the memory to a Local/Global/CAN memory.

A Global memory is sent via Radio and CAN. This is so that if a CAN cable is connected to a unit, it will automatically start sending its information to the other nodes it is connected to.

A CAN memory is sent only via CAN. This can be used if more information needs to be sent when using CAN.

#### Inputs

- The input to the memory is the signal that will be sent internally/via radio and CAN.

#### Outputs

- None.

#### Properties

- **Comment**  
Changes the comment text above this block.
- **Memory Name**  
This sets the name of the memory. This is what is used to reference the memory in other places. Using something descriptive makes it easier to find and use.
- **Memory Type**  
Changes the memory type of this memory (local/global/CAN).

#### Global Memory properties

- **Number**  
Selects which memory number to use. (only not used numbers can be selected)
- **StartUp Test**  
This specifies if a loss of radio communication requires an active 0 to be received at the receiver before it can output a 1 again
- **Do Not Repeat**  
This specifies if the memory should be repeated by other nodes or not.
- **Reaction Time**  
This specifies if the references to this memory should have long or short reaction time. The reaction time values are specified in the project settings.

#### CAN Memory properties

- **Number**  
Specifies which CAN memory number (CM01-CM16) should be used. These numbers are separate from the global memory numbers.
- **StartUp Test**  
This specifies if a loss of CAN communication requires an active 0 to be received at the receiver before it can output a 1 again.
- **Reaction Time**  
This specifies if the references to this memory should have long or short timeout. The timeout values are specified in the project settings.

## 14.3.2 Reference

### Block Diagram



Illustration 14.2: Memory reference without source.



Illustration 14.1: Memory reference with a configured source.

### Description

This block can be used to reference a Global, CAN, or local Memory signal. When referencing a local memory this block has no memory impact on the compiled program.

### Inputs

None

### Outputs

- The output of this block outputs the same signal as the input signal to the source memory. For local memories there is no memory overhead or delay, as the signal is referenced directly internally. Some communication delays occur when referencing memories from other nodes.

### Properties

- Comment  
Changes the comment text above this block.
- Source Memory  
Selects which memory source to reference.



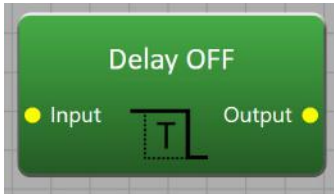
Note

See chapter 3 Safety Precautions when using Safety Simplifier on page 16.

## 14.4 Functions

### 14.4.1 Delay OFF

Block Diagram

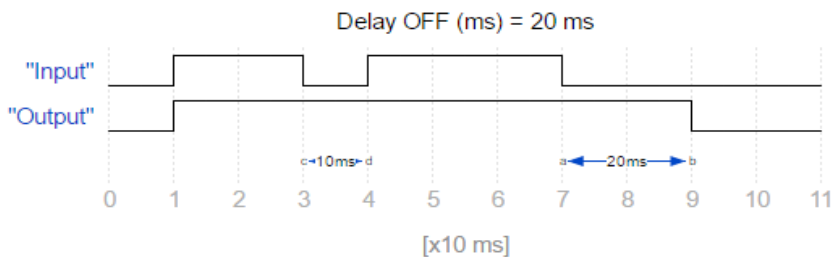


#### Description

This block can delay the 1→0 flank of the input signal for a specified amount of time. The input signal must be low for the specified Delay OFF time for the output to turn off.



This block directly affects the maximum reaction time of the function.



*Illustration 14.3: A Delay OFF block configured with Delay OFF=20ms. It has no delay on. The 0→1 flank pass right through the block, while the 1→0 flanks are delayed for the specified amount (here 20ms). Note that the first 1→0→1 pulse on the input does not turn off the output, as the output requires the input to be low for at least the specified Delay OFF time before it turns off.*

#### Inputs

- Input  
The input signal to delay the 1→0 flank of.

#### Outputs

- Output  
The delayed signal.

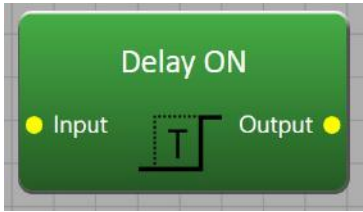
#### Properties

Properties	
Comment	<input type="text"/>
Delay OFF (ms)	<input type="text" value="100"/>

- Comment  
Changes the comment text above this block.
- Delay OFF (ms)  
The time in milliseconds to delay the 1→0 flank of the input signal.

## 14.4.2 Delay ON

### Block Diagram



### Description

This block can delay the 0→1 flank of the input signal for a specified amount of time. This can be used to filter away signal pulses that are shorter than the specified Delay ON time, since the input is required to be 1 (HIGH) for the specified Delay ON time for the output to go to 1 (HIGH).

### Diagrams

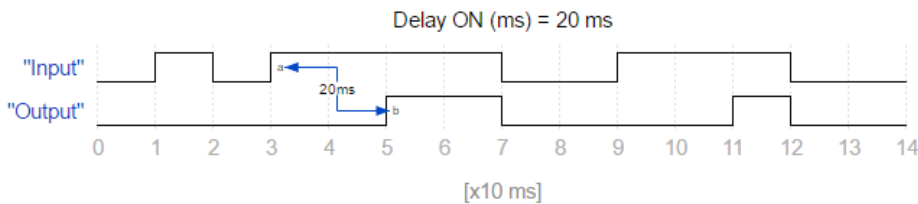


Figure 14.3 Timing diagram for a Delay ON block with Delay ON set to 20ms. The block has no delay off. Note the first pulse on the input does not turn the output high, as it requires the input to be high for longer than 20ms.

### Inputs

- Input  
The input signal to delay the 0→1 flank of.

### Outputs

- Output  
The delayed signal.

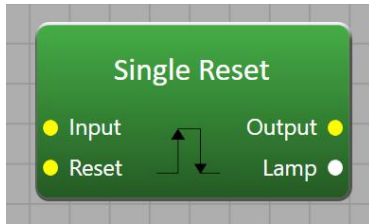
### Properties

Properties	
Comment	<input type="text"/>
Delay ON (ms)	<input type="text" value="100"/>

- Comment  
Changes the comment text above this block.
- Delay ON (ms)  
The time in milliseconds to delay the 0→1 flank of the input signal.

### 14.4.3 Single Reset

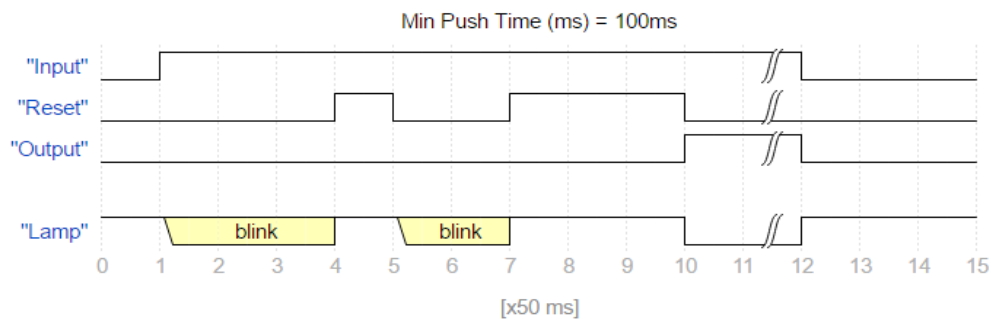
#### Block Diagram



#### Description

This is intended for setups and maintenance of machine. This block can be used to reset a safety device input, or some other signal that needs reset confirmation. To turn on the output, the input must receive a constantly high signal, and the reset must receive a 0→1→0 pulse, where the signal is 1 (HIGH) for longer than the specified Min Push Time milliseconds. If the input signal goes low at any time during or after the reset operation, the output turns off. The lamp output outputs a standard reset push button indication signal (ca 1Hz when inputs is high), that can be connected directly to the Indication input of a push button block.

#### Diagram



## Inputs

- **Input**  
This is where the signal that should be reset is connected. If this goes low at any time during or after the reset operation, the output will go low.
- **Reset**  
This input must receive a 0→1→0 flank sequence for the output to turn on.

## Outputs

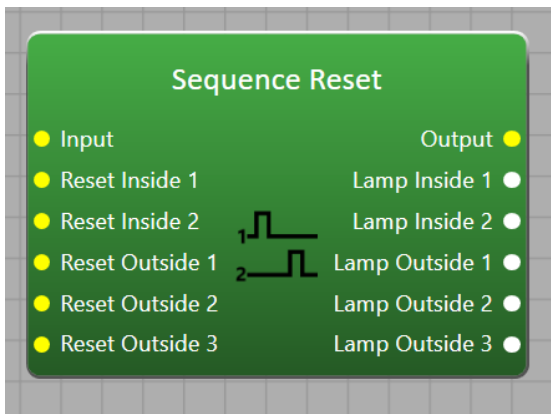
- **Output**  
This turns on if a valid reset sequence is received on the Reset input, and the Input input is 1 (HIGH). If the Input input goes low at any time, the output turns off.

## Properties

- **Comment**  
Changes the comment text above this block.
- **Min Push Time (ms)**  
The reset input must be pressed for this amount of time for the reset to be valid, and the output to go high.
- **Enable Lamp Output**  
This enables or disables the Lamp output connector.

## 14.4.4 Sequence Reset

### Block Diagram



### Description

This block can be used for reset functions that require several reset operations in sequence. It is constructed so that some reset buttons can be “inside” an enclosure. This means that during the reset of the inside buttons, the Input can go low. This is because if for example there is a gate that needs to be reset and there are reset buttons inside the enclosure, the gate must be opened to enter the enclosure and press the buttons and then opened again to leave the enclosure. After resetting the first outside button, the input must be 1 (HIGH) for the rest of the reset operation. If the input goes low during this time, the whole reset operation must be repeated from the beginning.

### Inputs

- **Input**  
This is the input signal that needs to be reset.
- **Reset Inside 0-8**  
These inputs allow the Input signal to go low during reset operation.

- **Reset Outside 1-8**  
These inputs do not allow the input signal to go low during operation.

#### Outputs

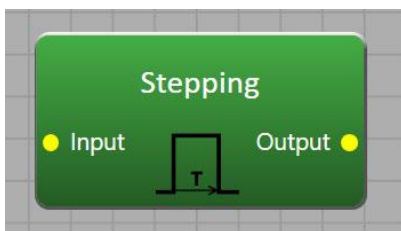
- **Output**  
After the reset operation is complete, this output goes to 1 (HIGH). Whenever the input falls, the output is 0 (LOW) and a new reset operation must be completed (except during the inside reset operation phase).
- **Lamp Inside/Outside (0-8)**  
These signals indicate which reset button is the next one in the sequence. A slow blink (500ms high, 500ms low) indicate that the button is next in the reset operation.

#### Properties

- **Comment**  
Changes the comment text above this block.
- **Max Time Between (ms)**  
This is the maximum time allowed between two reset operations. If this time is exceeded, the reset operation must be restarted.
- **Min Push Time (ms)**  
This is the minimum time a reset input must be high to count as a valid reset. This means the reset button must be pressed for at least this amount of time.
- **Buttons Inside**  
Specifies how many buttons should be inside (minimum 0, maximum 8). If this is set to 0 only outside reset buttons will be used.
- **Buttons Outside**  
Specifies how many buttons should be outside (minimum 1, maximum 8). The function always requires a button outside to finish the reset operation (the reset operation cannot be finished if there is someone inside, so the last button must be outside).

### 14.4.5 Stepping

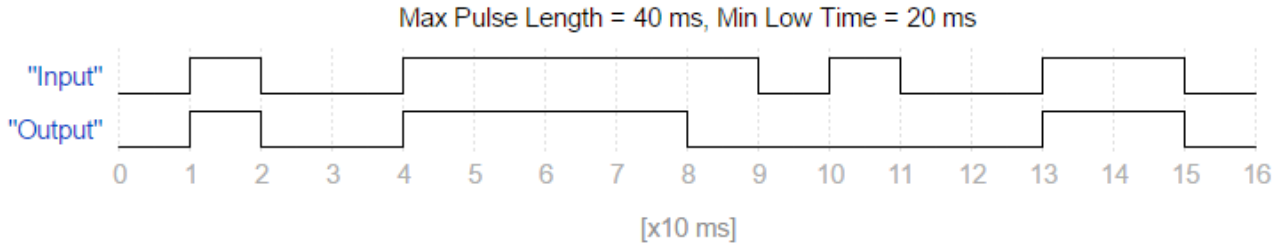
#### Block Diagram



#### Description

This block is intended for setup and maintenance. The block creates a pulse from an input signal. It ensures that the output is never high for longer than the specified Max Pulse Length time. After a pulse has been generated, another one cannot be generated until the specified Min Low Time (ms) has passed. If the input goes low before the specified Max Pulse Length time has passed, the output goes low and another pulse cannot be generated until the specified Min Low Time has passed.

Inputs



- Input  
The input that should generate the pulse.

Outputs

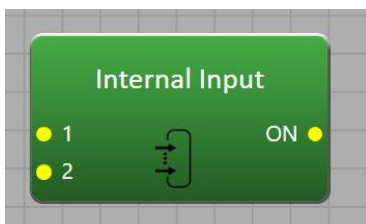
- Output  
Outputs the generated pulse.

Properties

- Comment  
Changes the comment text above this block.
- Max Pulse Length (ms)  
The maximum length of the pulse. The pulse is never longer than this.
- Min Low Time (ms)  
The output is low for this time before a new pulse can be generated.

### 14.4.6 Internal Input

Block Diagram



Description

This block can be used to combine several signals as an input internally. The function behaves the same way as a regular input function, but instead of using physical pins on the device it uses internal signals. This can be used to for example synchronize two separate two-hand device inputs operated at two different locations to ensure that they are pressed within a certain time, or to synchronize two or more global memories from other units.



Note

See chapter 13.11 Input Functions on page 89, and chapter 14.1.4 Advanced Input on page 109, for descriptions on how the input function works.



## Examples

Figure 14.4 describes how to connect two Two Hand Device inputs to an Internal Input to synchronize them. This ensures that both are pressed within a certain time of each other. If one is released, both must be released and pressed to restart the function again.

### Inputs

- 1-16  
The inputs to use in the function.

### Outputs

- ON  
Outputs 1 (HIGH) if the function is in ON state. Outputs 0 if the function is in OFF state.
- ERROR  
Outputs 1 (HIGH) if the function is in ERROR state.

### Properties

- Comment  
Changes the comment text above this block.
- Enable ERROR Output  
Enables the ERROR output connector.
- StartUp Test  
This property enables or disables the StartUp test check. If StartUp Test is enabled, the function cannot go to ON state directly at start-up (power on) but needs a valid OFF state first.
- Filter ON (ms)  
Filters the 0→1 flank of pins, so they must be in ON state for longer than this time.
- Simultaneity (ms)  
All pins must go to ON state within this time for the function to go to ON state. If this criterion is failed the input needs all pins to go to OFF state before a new attempt can be made.
- Enable Zero Time  
Enables or disables the Zero Time property.
- Zero Time (ms)  
If enabled, requires all the pins to be in OFF state for this time before the function can go to ON state.
- Terminal Count  
Sets the terminal count between 1 and 8.

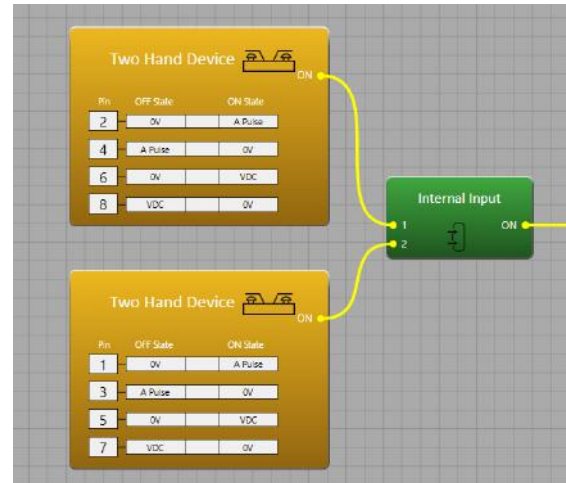


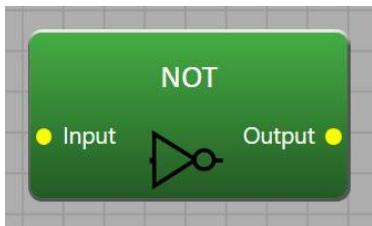
Figure 14.4

*Two Two-hand device inputs synchronized using an Internal Input block.*

## 14.5 Logic

### 14.5.1 NOT

Block Diagram



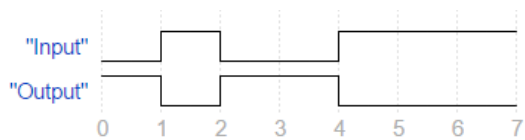
Description

This block inverts a signal.

Truth table

Input	Output
0	1
1	0

Diagrams



Inputs

- Input  
The signal to invert.

Outputs

- Output  
The inverted signal.

Properties

- Comment  
Changes the comment text above this block.

### 14.5.2 AND

Block Diagram



### Description

This block performs the boolean AND function between its inputs and outputs the result on the output.

### Truth table for 8 inputs

1	2	3	4	5	6	7	8	Output
0	x	x	x	x	x	x	x	0
x	0	x	x	x	x	x	x	0
x	x	0	x	x	x	x	x	0
x	x	x	0	x	x	x	x	0
x	x	x	x	0	x	x	x	0
x	x	x	x	x	0	x	x	0
x	x	x	x	x	x	0	x	0
x	x	x	x	x	x	x	0	0
1	1	1	1	1	1	1	1	1

### Inputs

- 1-16  
The signals that should be used in the function.

### Outputs

- Output  
The result of the function.

### Properties

- Comment  
Changes the comment text above this block.
- Boolean Function Type  
This changes the function to another boolean function (AND, NAND, OR, NOR, XOR, or XNOR).
- Inputs  
This sets how many inputs should be used. 2-16 can be configured.

## 14.5.3 NAND

### Block Diagram



### Description

This block performs the boolean NAND function between its inputs and outputs the result on the output.

### Truth table for 8 inputs

1	2	3	4	5	6	7	8	"output"
0	x	x	x	x	x	x	x	1
x	0	x	x	x	x	x	x	1
x	x	0	x	x	x	x	x	1
x	x	x	0	x	x	x	x	1
x	x	x	x	0	x	x	x	1
x	x	x	x	x	0	x	x	1
x	x	x	x	x	x	0	x	1
x	x	x	x	x	x	x	0	1
1	1	1	1	1	1	1	1	0

### Inputs

- 1-16  
The signals that should be used in the function.

### Outputs

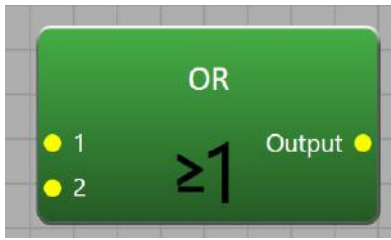
- Output  
The result of the function.

### Properties

- Comment  
Changes the comment text above this block.
- Boolean Function Type  
This changes the function to another boolean function (AND, NAND, OR, NOR, XOR, or XNOR).
- Inputs  
This sets how many inputs should be used. 2-16 can be configured.

## 14.5.4 OR

### Block Diagram



### Description

This block performs the boolean OR function between its inputs and outputs the result on the output.

### Truth table for 8 inputs

1	2	3	4	5	6	7	8	"output"
0	0	0	0	0	0	0	0	0
1	x	x	x	x	x	x	x	1
x	1	x	x	x	x	x	x	1
x	x	1	x	x	x	x	x	1
x	x	x	1	x	x	x	x	1
x	x	x	x	1	x	x	x	1
x	x	x	x	x	1	x	x	1
x	x	x	x	x	x	1	x	1
x	x	x	x	x	x	x	1	1

### Inputs

- 1-16  
The signals that should be used in the function.

### Outputs

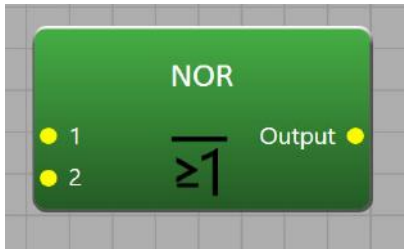
- Output  
The result of the function.

### Properties

- Comment  
Changes the comment text above this block.
- Boolean Function Type  
This changes the function to another boolean function (AND, NAND, OR, NOR, XOR, or XNOR).
- Inputs  
This sets how many inputs should be used. 2-16 can be configured.

## 14.5.5 NOR

### Block Diagram



### Description

This block performs the boolean NOR function between its inputs and outputs the result on the output.

Truth table for 8 inputs

1	2	3	4	5	6	7	8	"output"
0	0	0	0	0	0	0	0	1
1	x	x	x	x	x	x	x	0
x	1	x	x	x	x	x	x	0
x	x	1	x	x	x	x	x	0
x	x	x	1	x	x	x	x	0
x	x	x	x	1	x	x	x	0
x	x	x	x	x	1	x	x	0
x	x	x	x	x	x	1	x	0
x	x	x	x	x	x	x	1	0

### Inputs

- 1-16  
The signals that should be used in the function.

### Outputs

- Output  
The result of the function.

### Properties

- Comment  
Changes the comment text above this block.
- Boolean Function Type  
This changes the function to another boolean function (AND, NAND, OR, NOR, XOR, or XNOR).
- Inputs  
This sets how many inputs should be used. 2-16 can be configured.

## 14.5.6 XOR

### Block Diagram



### Description

This block performs the boolean XOR function between its inputs and outputs the result on the output.

### Truth table for 3 inputs

1	2	3	"output"
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

### Inputs

- 1-16  
The signals that should be used in the function.

### Outputs

- Output  
The result of the function.

### Properties

- Comment  
Changes the comment text above this block.
- Boolean Function Type  
This changes the function to another boolean function (AND, NAND, OR, NOR, XOR, or XNOR).
- Inputs  
This sets how many inputs should be used. 2-16 can be configured.

## 14.5.7 XNOR

### Block Diagram



### Description

This block performs the boolean XNOR function between its inputs and outputs the result on the output.

### Truth table for 3 inputs

1	2	3	"output"
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

### Inputs

- 1-16  
The signals that should be used in the function.

### Outputs

- Output  
The result of the function.

### Properties

- Comment  
Changes the comment text above this block.
- Boolean Function Type  
This changes the function to another boolean function (AND, NAND, OR, NOR, XOR, or XNOR).
- Inputs  
This sets how many inputs should be used. 2-16 can be configured.



## 14.6 Latches

### 14.6.1 T Latch

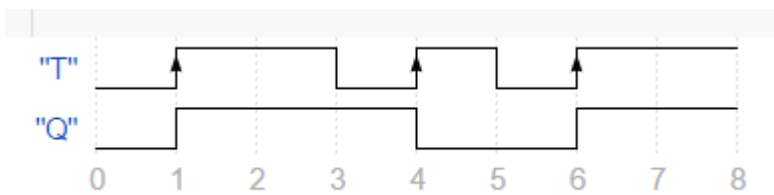
Block Diagram



Description

This block toggles the value of the output Q when a 0→1 flank is received on the T input.

Diagrams



Inputs

- T  
A high flank (0→1) on this input toggles the value of the output Q.

Outputs

- Q  
Toggles its value when a high flank is received on T.

Properties

- Comment  
Changes the comment text above this block.

## 14.6.2 SR Latch

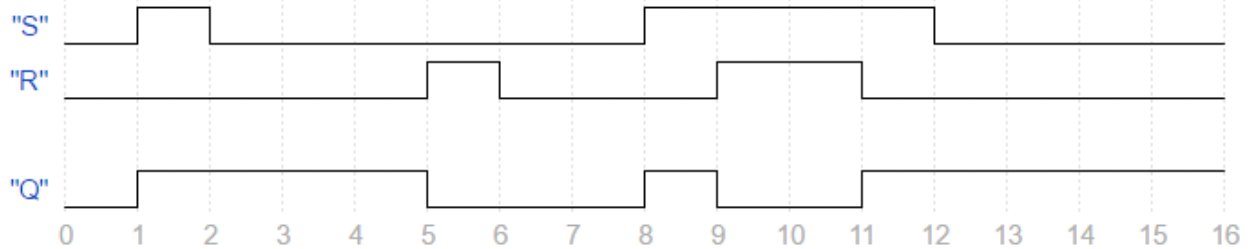
### Block Diagram



### Description

This block performs a set/reset latch function between its inputs. The output Q is latched high if S is 1 (HIGH). The output is latched 0 (LOW) if R is 1 (HIGH). If both R and S are 1 (HIGH) the output is 0 (LOW). If both inputs are 0, the output keeps the last value.

### Diagrams



### Inputs

- S  
Sets the output Q high.
- R  
Sets the output Q low.

### Outputs

- Q  
This output is 1 if S is 1, 0 if R is 1, and 0 if both R and S are 1.

### Properties

- Comment  
Changes the comment text above this block.

## 14.7 Signal Generators

### 14.7.1 Logic 1

Block Diagram



#### Description

This block generates a constant 1 (HIGH) signal on its output.

Inputs

None

Outputs

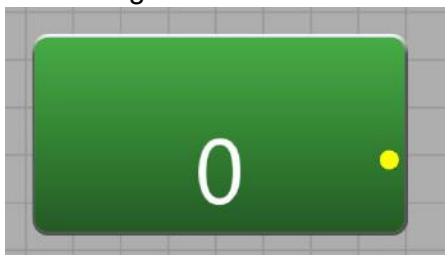
- The output outputs a constant 1 (HIGH) signal.

Properties

- Comment  
Changes the comment text above this block.

### 14.7.2 Logic 0

Block Diagram



#### Description

This block generates a constant 0 (LOW) signal on its output.

Inputs

None

Outputs

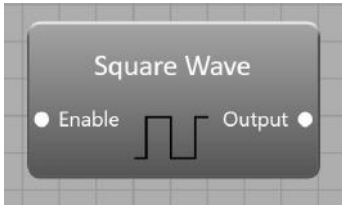
- The output outputs a constant 0 (LOW) signal.

Properties

- Comment  
Changes the comment text above this block.

### 14.7.3 Square Wave

#### Block Diagram



#### Description

This block generates a square wave clocking signal on its output. The Enable input can be used to enable or disable the square wave (0 = disabled, 1 = enabled). Note that this signal is not for safety and is intended for indication and status purposes only.

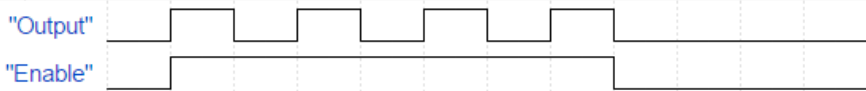
The square wave form is defined by specifying how long it should be high (Time High (ms)) and how long it should be low (Time Low (ms)).

If the Enable input is disabled, the block outputs a square wave constantly.



Note that this block is a non-safety block and is intended for indication and status purposes only.

#### Diagrams



#### Inputs

- **Enable**  
When this input receives a 0 (LOW) signal the output is constant 0. When this input receives a 1 (HIGH) signal, the output outputs a square wave defined by the user specified Time High and Time Low properties.

#### Outputs

- **Output**  
Outputs 0 (LOW) if Enable is 0. Outputs a square wave if Enable is 1.

#### Properties

- **Comment**  
Changes the comment text above this block.
- **Enable Activate Input**  
Enables or disables the Enable input.
- **Time High (ms)**  
Specifies how long the square wave should be high for.
- **Time Low (ms)**  
Specifies how long the square wave should be low for.

### 14.7.4 1Hz Blink

#### Block Diagram



### Description

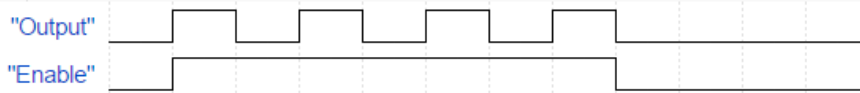
This block generates a 1Hz square wave (500ms low, 500ms high) clocking signal on its output. The Enable input can be used to enable or disable the square wave (0 = disabled, 1 = enabled). If the Enable input is disabled, the block outputs a square wave constantly.



#### Note

Note that this block is a non-safety block and is intended for indication and status purposes only.

### Diagrams



### Inputs

- Enable  
When this input receives a 0 (LOW) signal the output is constant 0. When this input receives a 1 (HIGH) signal, the output outputs a square wave defined by the user specified Time High and Time Low properties.

### Outputs

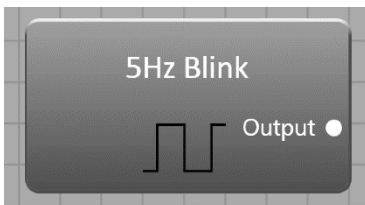
- Output  
Outputs 0 (LOW) if Enable is 0. Outputs a square wave if Enable is 1.

### Properties

- Comment  
Changes the comment text above this block.
- Enable Activate Input  
Enables or disables the Enable input.

## 14.7.5 5Hz Blink

### Block Diagram



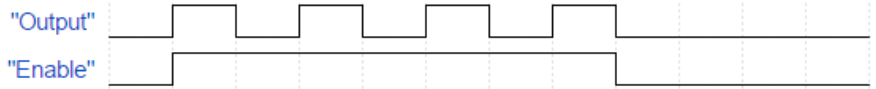
### Description

This block generates a 5Hz square wave (100ms low, 100ms high) clocking signal on its output. The Enable input can be used to enable or disable the square wave (0 = disabled, 1 = enabled). If the Enable input is disabled, the block outputs a square wave constantly.



Note that this block is a non-safety block and is intended for indication and status purposes only.

### Diagrams



### Inputs

- **Enable**  
When this input receives a 0 (LOW) signal the output is constant 0. When this input receives a 1 (HIGH) signal, the output outputs a square wave defined by the user specified Time High and Time Low properties.

### Outputs

- **Output**  
Outputs 0 (LOW) if Enable is 0. Outputs a square wave if Enable is 1.

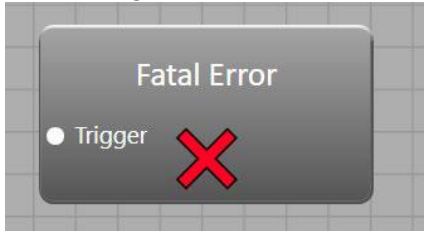
### Properties

- **Comment**  
Changes the comment text above this block.
- **Enable Activate Input**  
Enables or disables the Enable input.

## 14.8 Debug

### 14.8.1 Fatal Error

#### Block Diagram



#### Description

This block can be used to trigger the unit to enter fatal error (safe state) when the input signal is 1 (HIGH). The specified error code is displayed on LED 1-8 on the LED display. LED 15 and 16 lights up to indicate that a user error has been triggered from logic. LED 9-13 are switched off.

#### Inputs

- **Trigger**  
When a logic 1 (HIGH) signal is received on this input, the unit will enter fatal error state (safe state), turning off all outputs and displaying the specified error code on the LED display.

#### Outputs

- None.

#### Properties

- **Comment**  
Changes the comment text above this block.
- **Error Code**  
The error code that should be displayed as binary on the LED display. Setting this to 0 disables the fatal error.

## 14.8.2 CAN OK

### Block Diagram



#### Description

This block outputs a 1 (HIGH) signal when the node has direct CAN connection to the selected other node number.

#### Inputs

- None.

#### Outputs

- Outputs a 1 (HIGH) signal if the node has direct CAN connection to the selected node.

#### Properties

- Comment  
Changes the comment text above this block.
- To Node Number  
The node to check the CAN connection to.

## 14.8.3 Link OK

### Block Diagram



#### Description

This block outputs a 1 (HIGH) signal when the node has radio OR CAN connection to the selected other node number.

#### Inputs

- None.

#### Outputs

- Outputs a 1 (HIGH) signal if the node has radio OR CAN connection to the selected node.

#### Properties

- Comment  
Changes the comment text above this block.
- To Node Number  
The node to check the Link connection to.



## 14.9 Advanced

### 14.9.1 Unsafe & Safe

Block Diagram



#### Description

This block can be used if there is an unsafe signal that the programmer wants to be able to turn off a safety output or a condition to accept a station to go to ON. The output is a logical AND function between the inputs.



It is crucial that all safety equipment and signals are connected through the “Safe” input and not the “Unsafe” input. The “Unsafe” input is only supposed to take an extra unsafe signal that can turn on or off the output if the “Safe” input is high.



Safety calculations cannot be made for applications where this block is used. It is completely up to the system designer to manually perform the necessary calculations for applications using this block.

#### Inputs

- Safe  
The safe part of the application must be connected to this input.
- Unsafe  
This is the unsafe signal that should turn the output off or allow the output on.

#### Outputs

- Output  
This output is 1 when both inputs are 1.

#### Properties

- Comment  
Changes the comment text above this block.

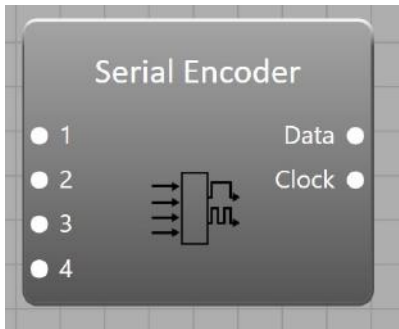
## 14.10 Non-Safety



All blocks in the non-safety category must only be used for status and information purposes.

### 14.10.1 Serial Encoder

Block Diagram



### Description

See chapter 13.15 Serial Communication for status information on page 99 for more info.

This block can encode several signals into a Data and Clock synchronous serial output. Up to 32 signals can be encoded. Connect the Data and Clock outputs to status output blocks to send the information via cable.

To deserialize the information a Serial Decoder block can be used (see page 146).

### Inputs

- 1-32  
The signals to serialize.

### Outputs

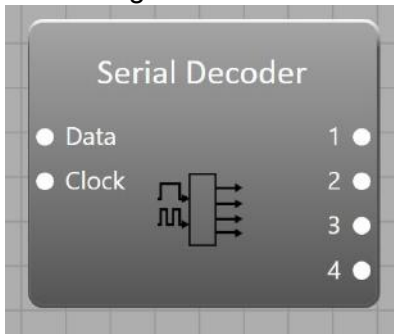
- Data  
The synchronous data output.
- Clock  
The clock output.

### Properties

- Comment  
Changes the comment text above this block.
- Intercharacter Duration (ms)  
The time between each packet.
- Half Bit Duration (ms)  
The time that the clock signal is high for each bit sent.
- Inputs  
The number of signals to be encoded. This sets how many input signals are on the block.

## 14.10.2 Serial Decoder

### Block Diagram



### Description

See chapter 13.15 Serial Communication for status information on page 99 for more info.

This block can be used to deserialize a synchronous serial stream of data. This can be the output from another Simplifier (see page 144), or from another PLC.

### Inputs

- Data  
The synchronous data signal to decode.
- Clock  
The clock to clock the data.

### Outputs

- 1-32  
The decoded data signals.

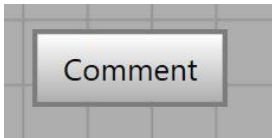
### Properties

- Comment  
Changes the comment text above this block.
- Intercharacter Duration (ms)  
This time specifies how long the pause between packets is.
- Outputs  
The number of signals to decode. This sets how many output signals are on the block.

## 14.11 Miscellaneous

### 14.11.1 Comment

Block Diagram



#### Description

This block can be used to put text comments in the logic graph. It does not affect the compiled logic.

Inputs

None

Outputs

- None.

#### Properties

- Text  
Changes the comment text.
- Text Size  
Changes the comment text size.

Contact

Safety System Products North AB

[www.sspnorth.se](http://www.sspnorth.se)

[Info@sspnorth.se](mailto:Info@sspnorth.se)

Tullkammarvägen 14

S-439 31 Onsala

Sweden