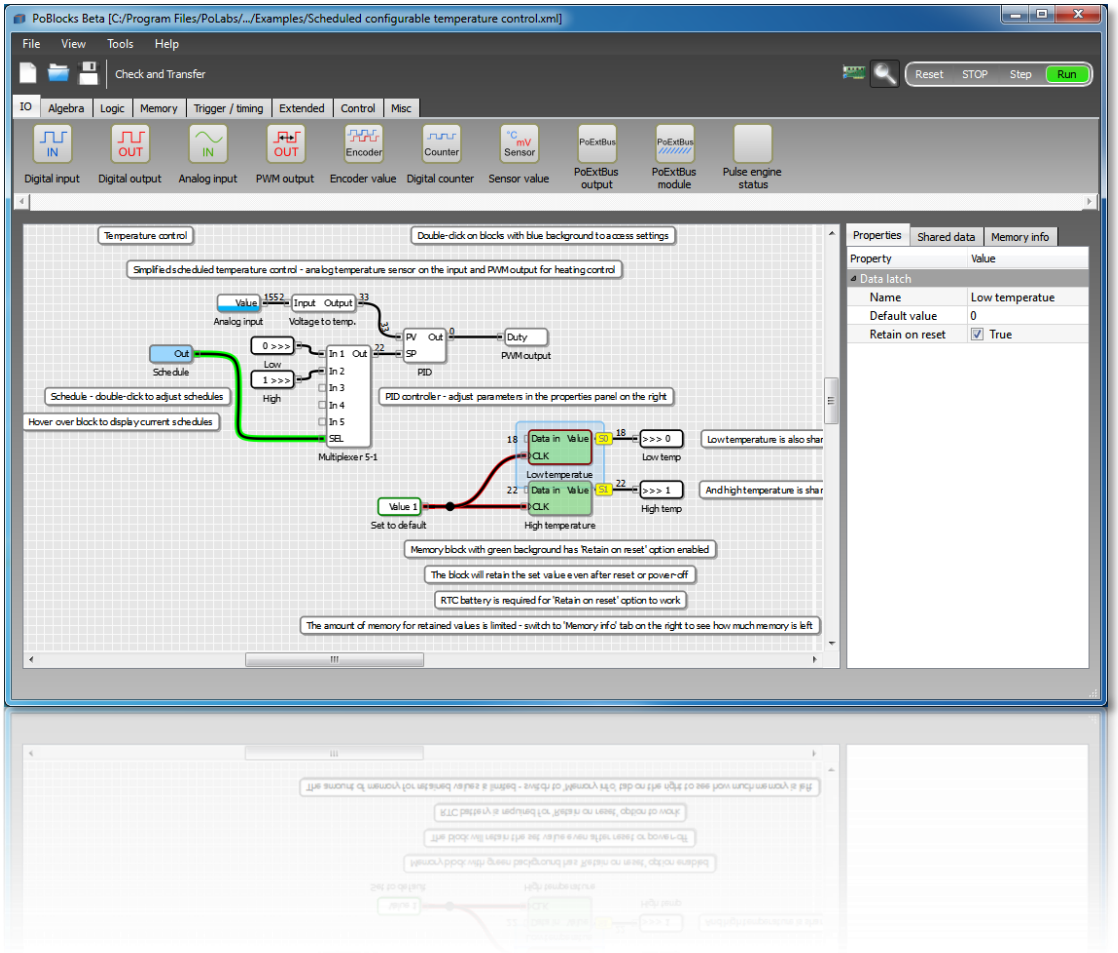
PoBlocks

User manual

Version: March 31, 2014



Please read the following notes

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice.
2. PoLabs does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of PoLabs products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of PoLabs or others. PoLabs claims the copyright of, and retains the rights to, all material (software, documents, etc.) contained in this release. You may copy and distribute the entire release in its original state, but must not copy individual items within the release other than for backup purposes.
3. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of the products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. PoLabs assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
4. PoLabs has used reasonable care in preparing the information included in this document, but PoLabs does not warrant that such information is error free. PoLabs assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
5. PoLabs devices may be used in equipment that does not impose a threat to human life in case of the malfunctioning, such as: computer interfaces, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment, and industrial robots.
6. Measures such as fail-safe function and redundant design should be taken to ensure reliability and safety when PoLabs devices are used for or in connection with equipment that requires higher reliability, for example: traffic control systems, anti-disaster systems, anticrime systems, safety equipment, medical equipment not specifically designed for life support, and other similar applications.
7. PoLabs devices shall not be used for or in connection with equipment that requires an extremely high level of reliability and safety, as for example: aircraft systems, aerospace equipment, nuclear reactor control systems, medical equipment or systems for life support (e.g. artificial life support devices or systems), and any other applications or purposes that pose a direct threat to human life.
8. You should use the PoLabs products described in this document within the range specified by PoLabs, especially with respect to the maximum rating, operating supply voltage range and other product characteristics. PoLabs shall have no liability for malfunctions or damages arising out of the use of PoLabs products beyond such specified ranges.
9. Although PoLabs endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, PoLabs products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a PoLabs product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures.
10. Usage: the software in this release is for use only with PoLabs products or with data collected using PoLabs products.
11. Fitness for purpose: no two applications are the same, so PoLabs cannot guarantee that its equipment or software is suitable for a given application. It is therefore the user's responsibility to ensure that the product is suitable for the user's application.
12. Viruses: this software was continuously monitored for viruses during production, however the user is responsible for virus checking the software once it is installed.
13. Upgrades: we provide upgrades, free of charge, from our web site at www.poscope.com. We reserve the right to charge for updates or replacements sent out on physical media.
14. Please contact a PoLabs support for details as to environmental matters such as the environmental compatibility of each PoLabs product. Please use PoLabs products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. PoLabs assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
15. Please contact a PoLabs support at support@poscope.com if you have any questions regarding the information contained in this document or PoLabs products, or if you have any other inquiries.
16. The licensee agrees to allow access to this software only to persons who have been informed of and agree to abide by these conditions.
17. Trademarks: Windows is a registered trademark of Microsoft Corporation. PoKeys, PoKeys55, PoKeys56U, PoKeys56E, PoScope, PoLabs and others are internationally registered trademarks.

1. PoBlocks description

PoBlocks is a graphical programming tool for PoKeys devices. It features an intuitive and clean interface and enables the user to quickly and easily design, deploy and debug a program that gets transferred and executed by the PoKeys device itself.

PoBlocks was developed with ease of use in mind, which means that it does not require long manuals, extended tutorials or deep knowledge to use. Although PoBlocks is simple to use, it also boasts a rich set of features - support for PoKeys basic and extended I/O interfaces support, timers, counters, configurable clock sources, algebra, memory, logic and non-linear operations, time schedule, event drums, even PID and on/off controllers, etc.

PoBlocks also features a simple to use monitor mode for debugging that gives user a good insight in how the program executes in real-time.

2. Main features

- **Simple and intuitive graphical user interface with integrated support**: just open the application and start designing your diagram. Drag the function blocks from the graphical toolbar with mouse and connect them by clicking on the input/output ports. When in doubt, hover over the block to access the integrated help.
- **Support for wide array of PoKeys peripherals**: PoBlocks gives you the access to digital inputs and outputs, analog inputs, PWM outputs, encoder inputs, digital counters, PoExtBus outputs and more just by dragging a block and selecting the pins in the property panel on the right.
- **Algebra, logic blocks**: Choose from basic algebra functions and logic functions to create simple conditional logics.
- **Memory blocks**: use JK, D, T or data latches, minimum/maximum value memories, simple RAM blocks.
- **Trigger and timing functions**: PoBlocks offers counters, signal level triggers, on-, off- and pulse-timers, etc.
- **Advanced blocks**: weekly time schedule, LCD interface support with multiple layouts, drum-style programming, process control etc. Advanced blocks enable you to quickly start controlling your process as you want it. If no block suit your needs, Custom PoIL block enables custom PoIL code execution.
- **One-click compiling and downloading**: when satisfied with your design, compile it and download it to the device with only one click.
- **Real-time debugging/monitoring**: with PoBlocks, your diagrams are simply created, then compiled and downloaded to the device with one click. When it comes to debugging or monitoring the process, simply activate the monitor mode and all outputs and connections will be populated with current values.

3. Requirements

- Windows XP, Vista, 7, 8 (other platforms coming soon)
- Visual C++ 2010 Redistributable Package installed on the target computer
- PoKeys56 or newer device (on USB or network)

4. Installation

PoBlocks application is a part of the PoKeys software installation package, available free of charge on www.poscope.com.

5. Contents

6. PoLabs PoIL core introduction

PoKeys PoIL core is a virtual 16/32-bit software processor, which interprets PoIL code in PoKeys device and has access to various PoKeys peripherals.

PoIL code is created in the compiling process in the PoBlocks application and is stored in the flash memory of the PoKeys device.

6.1. Operating modes

PoIL core has the following operating modes:

- **STOPPED**: PoIL core is stopped and no code is being executed
- **RUNNING**: PoIL core is executing code
- **EXCEPTION**: PoIL core encountered an error and execution had to be stopped
- **PAUSE**: PoIL core is temporarily halted and when restarted, continues from this point

6.2. Start-up configuration

When PoKeys device is started (power is applied), the PoIL core can be reset (initialization code executed) and PoIL code execution can start automatically. By default, PoIL code is not automatically executed on reset (Auto-start is disabled).

6.3. Cycle time

The PoIL core supports priority-based pre-emptive scheduler that switches between two (or more on later versions) tasks. Task 0 has the lowest priority and is enabled by default. Other tasks are periodic tasks that have a fixed time-period between executions. Task switching is done at 1 ms intervals or on task exit events.

Task 1 is automatically used and enabled by PoBlocks software for executing the PoIL code of the diagram when periodic mode is selected (when cycle time, greater than 0 is specified). In this mode, minimum cycle time for execution is limited by the code size, but cannot be lower than 1 ms. In non-periodic mode (cycle time set to 0), task 0 is used and cycle time depends solely on the PoIL code size.

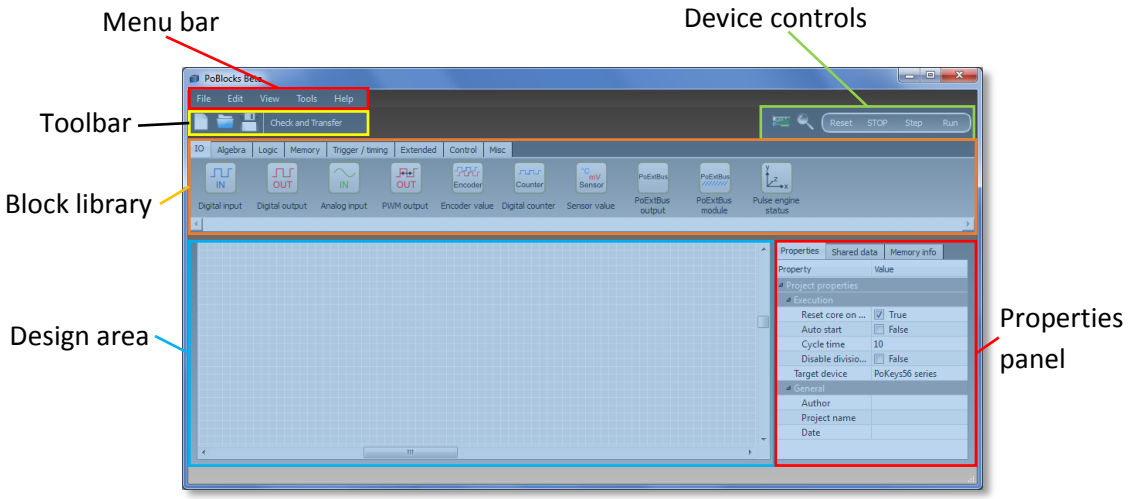
PoBlocks features an integrated Task manager that displays the target and actual cycle times of the tasks being executed (Task manager is found in PoBlocks application menu under ‘Tools > Task manager’). For more details, see ‘Task manager’ below.

6.4. PoIL programming

PoBlocks enables low-level PoIL language programming using the custom PoIL block. Contact us at support@poscope.com and request PoIL documentation document.

7. User interface

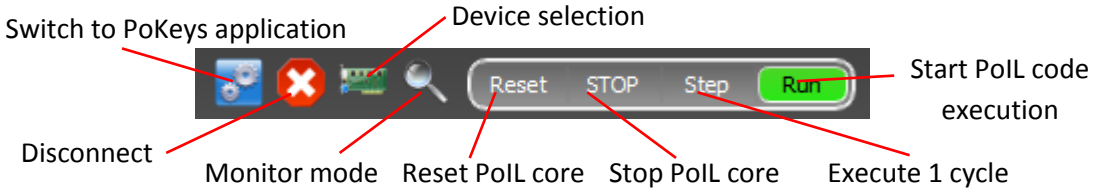
PoBlocks user interface is displayed below. Top area contains menu bar, tolbar with device controls and block library. Bottom left area is diagram design area, while the right part is reserved for properties panel that contains dynamic properties of project or currently selected block.



7.1. Toolbar and device controls

Toolbar gives the access to most-frequently used functions of the PoBlocks application. Besides creating new, opening and saving the designed diagram, 'Compile and transfer' button enables one-click compiling, checking and transferring of the diagram to the device. If the code is already running the device, a dialog appears asking user to confirm the download procedure.

Device controls panel enables user to interact with the PoIL core in the PoKeys device.



The following controls are available:

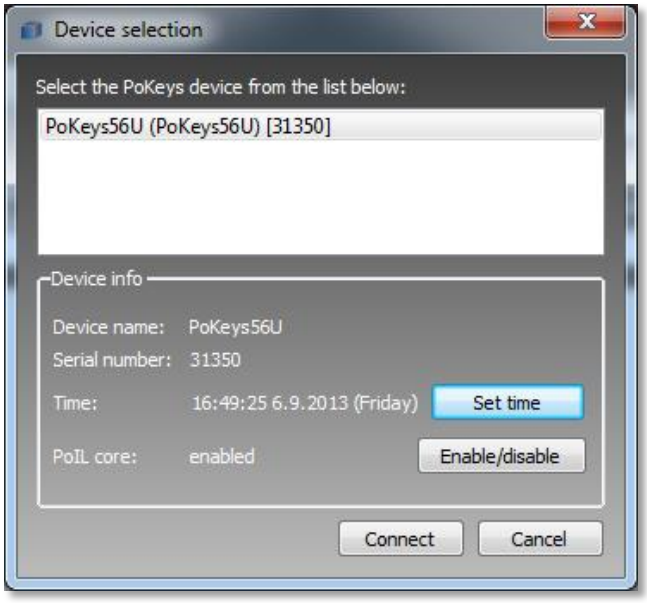
- **Switch to PoKeys application**: disconnect and open PoKeys application for advanced peripheral setup and testing. This option is only available if PoKeys application is installed to the default installation path.
- **Disconnect**: click this to disconnect from current device

- **Device selection**: this opens a device selection dialog to select a device to download the PoIL code to.
- **Monitor mode**: clicking the Monitor mode button enables or disables the real-time monitor mode (this is available after the diagram has been successfully compiled and downloaded to the device).
- **Reset PoIL core**: use this command to reset the PoIL code execution.
- **Stop PoIL core**: this command stops the PoIL code execution.
- **Execute 1 cycle (step)**: this command executes the PoIL code of the diagram once and switches the PoIL core state back into STOPPED mode.

7.2. Device selection dialog

Device selection dialog is used to select a PoKeys device to work with. All detected devices are displayed in the list and selecting a device refreshes the device information panel, giving the user information on device's name, its serial number, current time and status of PoIL core activation. All PoKeys devices are shipped with PoIL core disabled and user must enable the core once prior downloading the diagram to that device. This is accomplished by clicking the 'Enable/disable' button (note: enabling PoIL core disables keyboard macro capability of PoKeys56U devices).

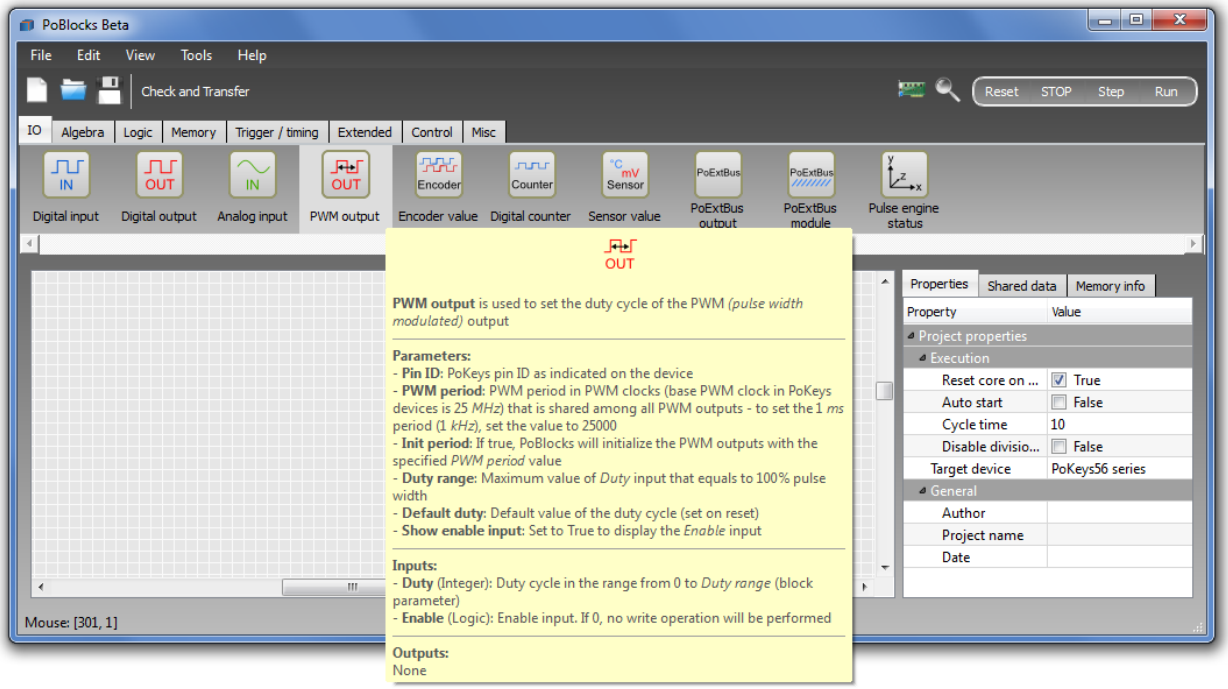
PoKeys devices with a holder for a RTC battery support RTC (real time clock) that can be used for scheduling the events. RTC in the device is set to current computer time by clicking the button 'Set time'.



7.3. Block library

Block library panel contains graphical representations of function blocks in the PoBlocks library. The library is divided into block categories and tabbed menu system is used to switch between these. Each block in the block library offers integrated help system (see the figure below) that gets activated by hovering the mouse cursor over the block.

To insert the block from the library into the diagram, drag a block using mouse cursor from the block library into the diagram.

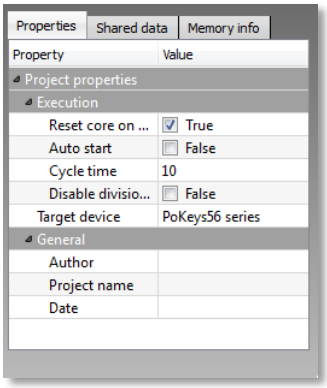


7.4. Properties panel

Properties panel contains a dynamic grid of properties based on the currently selected object in the diagram. By clicking on an empty space (without blocks), PoBlocks project properties are displayed in the grid.

7.5. Project properties

Project properties are accessed by clicking on an empty space in the designing area (an area without blocks or connections). The following options are available in the properties panel



Reset core on startup: if checked, PoIL core executes the initialization code on PoKeys device startup (defining the pin functions, setting-up the peripherals used in the diagram, etc.)

Auto start: if checked, PoIL core starts PoIL code execution immediately after the PoKeys device is powered up

Cycle time: this option sets the cycle time, defining the time between each diagram PoIL code execution

Disable division by zero exception: if checked, division by zero event does not stop PoIL code execution, but generates result 0

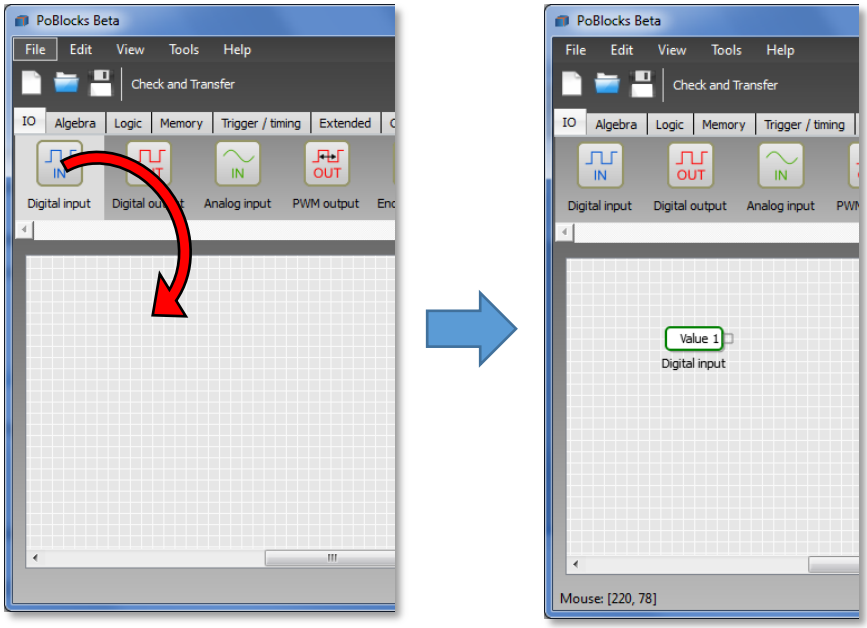
Target device: a drop-box menu used to select the device type that will be used to execute the diagram's code

General section: contains fields used for project documentation purposes

8. Using PoBlocks

8.1. Inserting blocks

Use left mouse button to drag a selected block from the block library to design area.



8.2. Removing blocks

Select one block or use lasso selection to select multiple blocks, then press 'Delete' button.

8.3. Moving blocks

Use left mouse button to move the selected block.

8.4. Copying blocks

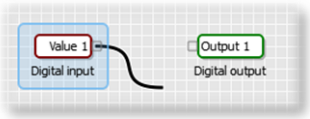
Select the blocks you want to copy by a ‘lasso tool’. Press down the left mouse button on an empty part of the diagram, then drag the mouse to select the blocks. To copy the blocks, press Ctrl+C or go to Edit > Duplicate selected.

8.5. Connecting blocks

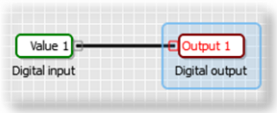
Right port of one block can only be connected to left port of another block. Left port of one block can however be connected to either right port of another block or to an existing connection.

Start connecting the blocks by clicking on one of the empty ports (a port that can accept a connection will be highlighted in green). You can then either drag the connection to another port and release the mouse button or release the button immediately and click again on the destination block's port.

1. Start at source port



2. Drag the connection

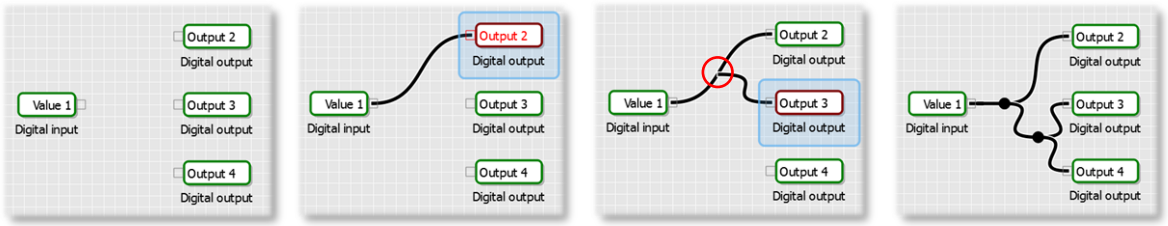


3. End at destination block

In order to route the connection around other blocks to improve the readability of the diagram, intermediate points can be added to the connection. Just click on an empty space while dragging the connection and an intermediate point will appear. Based on the connection profile, the connection will pass the intermediate point in either vertical or horizontal direction. To move the intermediate point, complete the connection, click on the intermediate point and drag it into the correct position.

Due to the properties of the block diagrams, left port (input) of each block can be connected to only one right port (output) of another (or the same) block. However, each output can be connected to multiple inputs. PoBlocks uses these rules and refuses a connection that breaks them.

In order to connect multiple inputs (left ports) to one output (right port), start by creating a base connection between one input and one output. Then, start adding another connection by clicking on an empty input and complete the connection by clicking on an existing connection.



8.6. Constant inputs

Often a simple constant value is used as an input parameter. In order to insert a constant value, right click on an empty input (left port) - constants can not be added to output (right ports). An default constant value 0 appears - to edit the value, just click on it and enter a new value. To replace the constant with a connection, simply drop the new connection to the port, containing the constant value - the constant value will be replaced by the newly established connection.

8.7. Removing connections

In order to remove an existing connection, select the connection by clicking on it and press 'Delete' button.

8.8. Configuring blocks

Basic blocks are configured using the properties panel on the right of the design area. Select a block first and the properties panel will be automatically populated with options for that block. Help for each property is included in the block help pop-up window in the block library.

Advanced (extended) blocks (those that have light blue background) like Event drum, Look-up table, Schedule, LCD UI, Custom PoIL, etc. have a dedicated block properties editor. This editor is accessed by double-clicking on a block. The description of each of these blocks can be found in the blocks section below.

9. Compile process

After PoBlocks diagram is finished, a built-in compiler is used to translate it into PoIL code that is then transferred and executed on a PoKeys device. During compiling, the diagram is checked for errors and if any are detected, a pop-up error list window will appear and compiler process will be stopped. User has to correct the errors before the diagram can be fully compiled and transferred to the PoKeys device.

Compile process executes the following operations

- The blocks are first put into a proper execution sequence based on their function and how they are connected to other blocks. PoIL core executes code of one block after the other and the proper sequence is required to obtain correct result. If the diagram consists of algebraic loops (closed loops containing blocks that can not be properly sectioned), an error is thrown and user has to insert a proper '1 T delay' block somewhere in the loop in order to instruct the compiler where to split the loop.
- After the block execution sequence is defined, PoIL code is generated for each block.
- Code of all blocks is joined and optimized.
- Compile time errors (if any) are displayed in the errors pop-up window.

10. Modes of operation (Run, Step, Stop)

When a PoKeys device is selected in the 'Device selection' dialog, an operating mode of the PoIL core is automatically displayed in device controls as a highlighted item.

STOP: PoIL core is in STOPPED state, no PoIL code is being executed

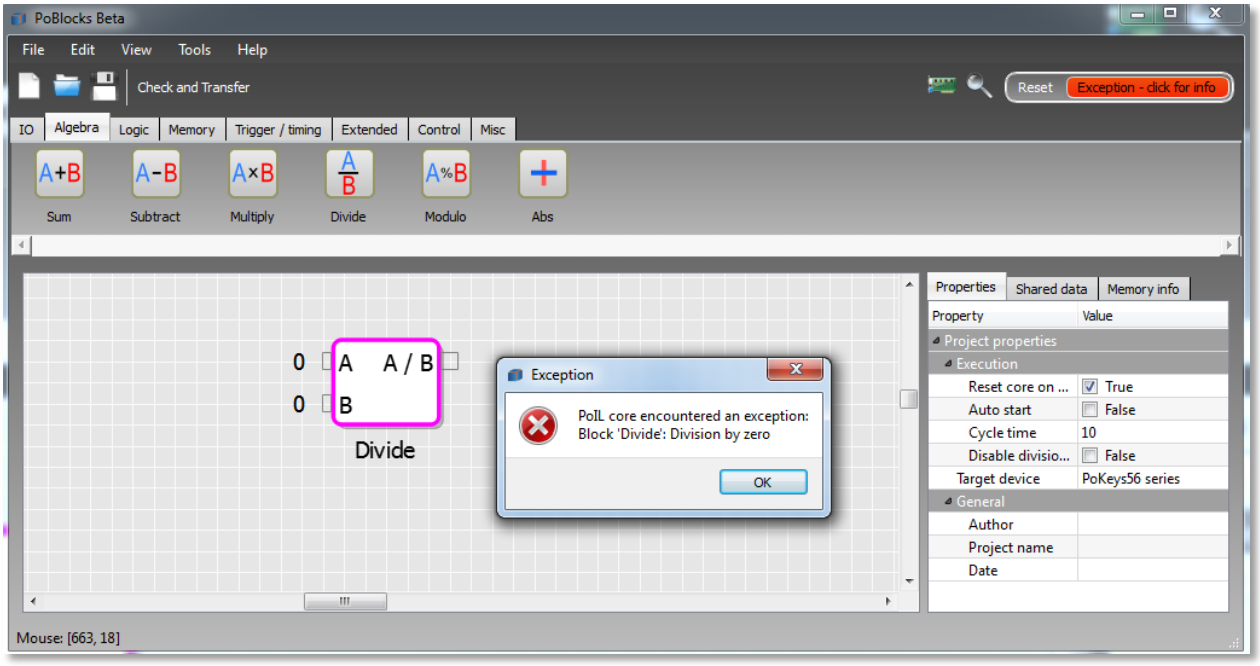
Step: After pressing Step, one cycle of PoIL diagram will be executed and the PoIL core will be put back to STOPPED mode

Run: PoIL core is running normally

Exception: PoIL core encountered an exception and had to be stopped. For more info on the cause of the exception, click the red Exception button.

11. Exceptions

The following figure displays a 'Division by zero' exception description after clicking the 'Exception' mode button in the device controls. The dialog also displays the most probable block that caused the exception.



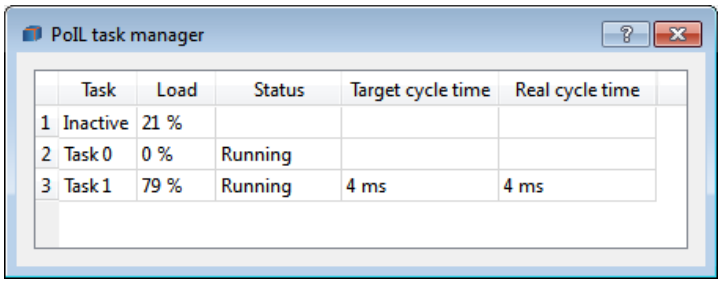
12. Monitor mode 

PoBlocks features a ‘Monitor mode’ that is used to observe the PoBlocks diagram execution on the device in nearly real-time. Monitor mode can be enabled after the diagram is compiled and transferred to the device using the ‘Monitor mode’ toggle switch in device controls section. Once activated, no changes to the diagram are allowed until the monitor mode is deactivated.

All output ports of blocks in the diagram are equipped with the output value display that extend over the connection to the other block. Connections that carry Boolean signals (On/Off, True/False) are colored in gray (inactive = Off, False) or in light green (active = On, True), giving user a visual feedback of the diagram state.

13. Task manager

PoBlocks features task manager feature that gives the user an insight into how the PoIL code is being executed. Task manager can be accessed when the connection with PoKeys device is established by clicking on Tools > Task manger.



	Task	Load	Status	Target cycle time	Real cycle time
1	Inactive	21 %			
2	Task 0	0 %	Running		
3	Task 1	79 %	Running	4 ms	4 ms

Task manager shows all tasks that are currently being executed and the basic information on tasks' performances.

'Inactive' task shows how much 'Load' is still available to tasks (unused processing time)

Task 0 is the primary task being executed by PoKeys device. In periodic mode this task is only used to initialize and start other tasks and does not affect the operation afterwards, while in non-periodic mode, only task 0 is used for code execution.

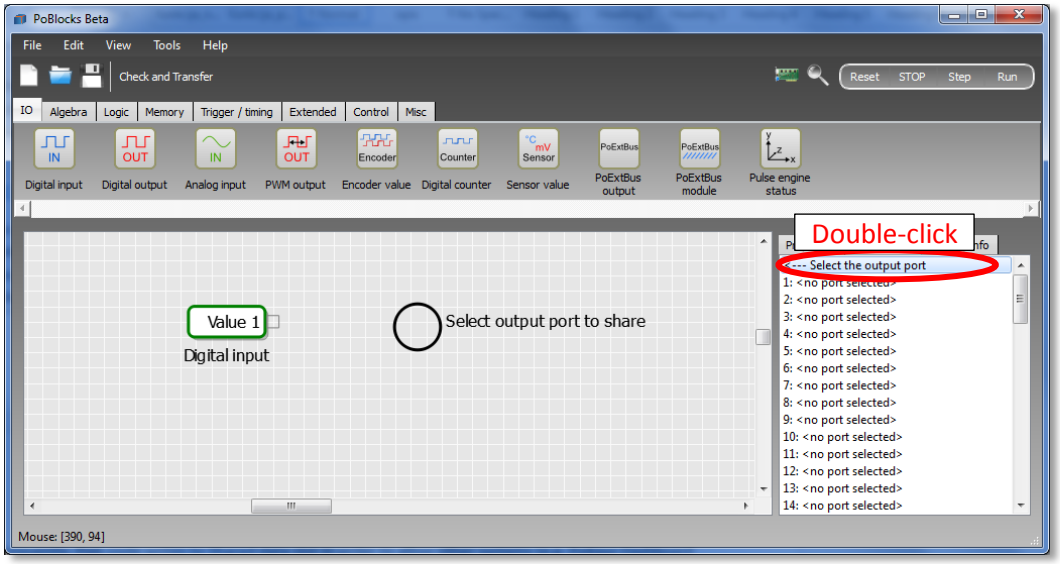
Last two columns show the 'Target cycle time' (set in the project settings) and 'Real cycle time' (true/real cycle time managed by the device).

14. Shared data

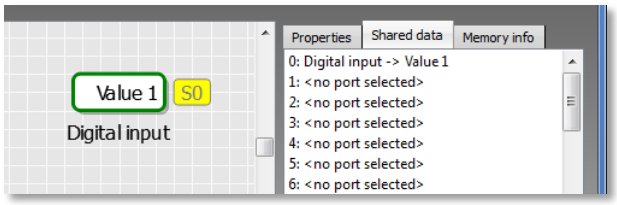
In order to share data between PoIL code and other PoKeys systems (Web dashboard and Modbus interface on Ethernet PoKeys devices, third-party applications on other devices or computers, connected to PoKeys, etc.) a shared data slots were introduced in PoKeys devices with PoIL core support. Each Shared data slot can hold 1-, 8-, 16- or 32-bit data, which can be read or written by any of the involved systems. Most frequently, PoIL code writes to Shared data slot in order to allow other systems (e.g. PoKeys Dashboard or Modbus device) or reads from Shared data slot in order to obtain data from other systems (e.g. user interaction via Dashboard or Modbus device operation).

14.1. Writing data to Shared data slot

In order to write data to shared data slot, switch to 'Shared data' tab in the properties section and double-click on an empty slot to start the slot assignment process (figure below). A circular selection will appear in the design area, instructing you to select an output port to share.



Move the cursor over to the port you want to share and click on it – this will complete the assignment process for that shared data slot. A shared port will be highlighted in yellow with the text showing 'Sx' (where x is the Shared data slot ID).

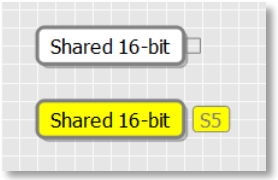


Shared slot ID (displayed as a shared slot index in the list and on every port with shared slot assigned to) is used to refer to this slot from other systems.

Certain memory objects (data latch, min/max value memory, etc.) can be used to read and write shared data slots. When a latch clock signal (for data latch) or lower/higher input value is detected, a shared data slot value is updated with a new value. If other systems write to the same shared data slot, this new value is used in PoIL code also.

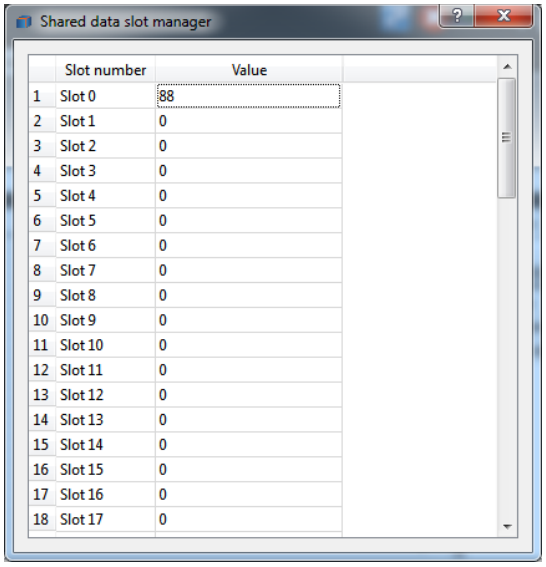
14.2. Reading data from shared data slots

Use on of the 'Shared x-bit' blocks (where x is 1, 8, 16 or 32) from the block library under the 'Misc' category. Shared data slot must then be assigned to the output port of that block (figure below illustrates a Shared 16-bit block without and with a shared data slot assigned.



Monitoring and changing of shared data slots

In order to see the values of shared data slots and/or to change it, open the Shared data slot manager (to open it, go to 'Tools > Shared slot manager' menu).



15. PoBlocks data types

In general, PoBlocks supports Boolean values (1-bit) and signed integer numbers (8-, 16- or 32-bit). While most of blocks use predefined data types for inputs and outputs, PoBlocks compiler converts between the types automatically without user intervention. Support for floating point arithmetic is not implemented, fix point arithmetic has to be used instead by the user (e.g. to obtain 0.01 resolution, all calculations must be done with numbers greater for a factor of 100, only divided or modulated by 100 when displaying the values to the user).

16. Blocks description

16.1. IO - Digital input

Description	Digital input is used to read the state of the digital input on PoKeys device
Inputs	None
Outputs	***Value*** (Logic): Digital input state
Properties	***Pin ID*** (Integer, 1 to 55): PoKeys pin ID as indicated on the device ***Init function*** (Logic): If true, PoBlocks will setup the selected pin as digital input on startup ***Inverted*** (Logic): If true, pin state will be inverted
Remarks	None

16.2. IO - Digital output

Description	Digital output is used to set the state of the digital output on PoKeys device
Inputs	***Output*** (Logic): Digital output state ***Enable*** (Logic): Enable input. If 0, no write to output will be performed
Outputs	None
Properties	***Pin ID*** (Integer, 1 to 55): PoKeys pin ID as indicated on the device ***Init function*** (Logic): If true, PoBlocks will setup the selected pin as digital input on startup ***Inverted*** (Logic): If true, pin state will be inverted ***Default value*** (Logic): The default state of the output on reset (True=1, False=0) ***Set to default on init*** (Logic): If true, the value of the output is set to *Default value* on reset ***Show enable input*** (Logic): Set to True to display the *Enable* input
Remarks	None

16.3. IO - Analog input

Description	Analog input is used to read one analog input value on PoKeys device
Inputs	None
Outputs	***Value*** (16-bit integer): Analog input value (0 to 4095)
Properties	***Pin ID*** (Integer, 41 to 47): PoKeys pin ID as indicated on the device ***Init function*** (Logic): If true, PoBlocks will setup the selected pin as digital input on startup ***Output value*** (Other type): Output value type selection
Remarks	None

16.4. IO - PWM output

Description: PWM output is used to set the duty cycle of the PWM *(pulse width modulated)* output

Inputs:
Duty (32-bit integer): Duty cycle in the range from 0 to *Duty range* (block parameter)
Enable (Logic): Enable input. If 0, no write operation will be performed

Outputs: None

Properties:
Pin ID (Integer, 17 to 22): PoKeys pin ID as indicated on the device
PWM period (Integer): PWM period in PWM clocks (base PWM clock in PoKeys devices is 25 *MHz*) that is shared among all PWM outputs - to set the 1 *ms* period (1 *kHz*), set the value to 25000
Init period (Logic): If true, PoBlocks will initialize the PWM outputs with the specified *PWM period* value
Duty range (Integer): Maximum value of *Duty* input that equals to 100% pulse width
Default duty (Integer): Default value of the duty cycle (set on reset)
Show enable input (Logic): Set to True to display the *Enable* input

Remarks: None

16.5. IO - PWM output C

Description: PWM output C is used to set the duty cycle and period of the PWM *(pulse width modulated)* outputs. PWM outputs 1 to 6 equate to pins 17 to 22 on PoKeys56E/U. **Warning: pulse width anomalies can occur when changing PWM period**.

Inputs:
Period (32-bit integer): PWM period in PWM clocks (base PWM clock in PoKeys devices is 25 *MHz*) that is shared among all PWM outputs - to set the 1 *ms* period (1 *kHz*), set the value to 25000
Update period (Logic): Update input. On rising edge, PWM period will be updated
Duty 1-6 (32-bit integer): Duty cycle for PWM output 1-6 in percent
Enable (Logic): Enable input. If 0, no write operation will be performed

Outputs: None

Properties:
Default period (Integer): Default PWM period (set on reset)
Show enable input (Logic): Set to True to display the *Enable* input
Default duty 1-6 (Integer): Default value of the duty cycle in percent

Remarks: None

16.6. IO - Encoder value

Description: Encoder value reads the value of the encoder counter

Inputs: None

Outputs:
Encoder (32-bit integer): Encoder counter value

Properties:
Encoder ID (Integer, 1 to 26): PoKeys encoder ID (0 to 25)
Channel A pin (Integer, 1 to 55): PoKeys pin ID as indicated on the device, used for encoder A channel signal

Channel B pin (Integer, 1 to 55): PoKeys pin ID as indicated on the device, used for encoder B channel signal
Multiplier 4x (Logic): Activate the 4x multiplier for encoder signals - encoder counter is incremented on any change of the A or B signals
Init encoder (Logic): If true, PoBlocks will initialize the encoder with the specified settings on startup
Clear on start (Logic): If true, encoder counter will be reset on startup
Show reset input (Logic): If true, reset input is displayed

Remarks None

16.7. IO - Digital counter

Description Digital counter reads the value of the digital counter. Digital counter enables counting signals of higher frequencies

Inputs None

Outputs ***Value*** (32-bit integer): Digital counter value

Properties ***Counter pin*** (Integer, 1 to 55): PoKeys pin ID as indicated on the device, used as digital counter input. Not all pins support digital counters - advise PoKeys manual
Use direction pin (Logic): If True, pin specified by *Direction pin* is used to define the count direction
Direction pin (Integer, 1 to 55): PoKeys pin ID as indicated on the device, used as digital counter direction input
Rising edge (Logic): If true, counter will be incremented on rising signal edges
Falling edge (Logic): If true, counter will be incremented on falling signal edges
Clear on start (Logic): If true, counter will be reset on startup
Show reset input (Logic): If true, reset input is displayed

Remarks None

16.8. IO - Sensor value

Description Sensor value outputs the current value of the selected sensor

Inputs None

Outputs ***Value*** (32-bit integer): Sensor value
Sensor OK (Logic): Sensor status - 0 if sensor is inactive or error occured

Properties ***Sensor ID*** (Integer, 1 to 27): PoKeys sensor ID as configured in PoKeys configuration - I2C sensors have IDs between 1 and 10, 1-wire sensors have IDs between 11 and 20, analog sensors have IDs between 21 and 27.

Remarks None

16.9. IO - PoExtBus output

Description PoExtBus output sets the state of one output on PoExtBus module
Inputs ***Output*** (Logic): Output state
Enable (Logic): Enable input. If 0, no write operation will be performed
Outputs None
Properties ***Module number*** (Integer, 1 to 10): PoExtBus module ID (1-10)
Module output (Integer, 1 to 8): PoExtBus output ID (1-8)
Default value (Logic): The default state of the output on reset (True=1, False=0)
Set to default on init (Logic): If true, the value of the output is set to *Default value* on reset
Show enable input (Logic): Set to True to display the *Enable* input
Remarks None

16.10. IO - PoExtBus module

Description PoExtBus module sets the state of 8 PoExtBus outputs on selected PoExtBus module
Inputs ***Output 1-8*** (Logic): Output state
Enable (Logic): Enable input. If 0, no write operation will be performed
Outputs None
Properties ***Module number*** (Integer, 1 to 10): PoExtBus module ID (1-10)
Set to default on init (Logic): If true, the value of the output is set to *Default value* on reset
Output 1-8 default (Logic): The default state of the output on reset (True=1, False=0)
Show enable input (Logic): Set to True to display the *Enable* input
Remarks None

16.11. IO - Pulse engine status

Description Pulse engine status - current position of the pulse engine
Inputs None
Outputs ***x*** (32-bit integer): x axis
y (32-bit integer): y axis
z (32-bit integer): z axis
Properties None
Remarks

16.12. Algebra - Sum

Description Sum of inputs
Inputs ***A*** (32-bit integer): First variable A
B (32-bit integer): Second variable B

C (32-bit integer): Third variable C
D (32-bit integer): Forth variable D
Outputs ***Sum*** (32-bit integer): Sum of A and B
Properties ***Inputs*** (Integer, 2 to 4): Number of inputs
Remarks None

16.13. Algebra - Subtract

Description Difference of two inputs
Inputs ***A*** (32-bit integer): First variable A
B (32-bit integer): Second variable B
Outputs ***A - B*** (32-bit integer): Difference of A and B (A - B)
Properties None
Remarks None

16.14. Algebra - Multiply

Description Product of inputs
Inputs ***A*** (32-bit integer): First variable A
B (32-bit integer): Second variable B
C (32-bit integer): Third variable C
D (32-bit integer): Forth variable D
Outputs ***Product*** (32-bit integer): Product of inputs
Properties ***Inputs*** (Integer, 2 to 4): Number of inputs
Remarks None

16.15. Algebra - Divide

Description Division of two inputs
Inputs ***A*** (32-bit integer): First variable A
B (32-bit integer): Second variable B
Outputs ***A / B*** (32-bit integer): Division result of A and B (A / B)
Properties None
Remarks Division by zero triggers a Division by zero exception in PoIL core. To stop this from happening, 'Disable division by zero exception' must be enabled in project properties

16.16. Algebra - Modulo

Description Modulo operation of two inputs
Inputs ***A*** (32-bit integer): First variable A
B (32-bit integer): Second variable B
Outputs ***A mod B*** (32-bit integer): Modulo operation result (A modulo B)

Properties None
Remarks None

16.17. Algebra - Abs

Description Absolute value of the input signal
Inputs ***Input*** (32-bit integer): Input variable
Outputs ***Output*** (32-bit integer): Absolute value of the input variable
Properties None
Remarks None

16.18. Algebra - Min

Description Min Minimum of inputs
Inputs ***A*** (32-bit integer): First variable A
B (32-bit integer): Second variable B
C (32-bit integer): Third variable C
D (32-bit integer): Forth variable D
Outputs ***Min*** (32-bit integer): Minimum value of inputs
Properties ***Inputs*** (Integer, 2 to 4): Number of inputs
Remarks None

16.19. Algebra - Max

Description Max Maximum of inputs
Inputs ***A*** (32-bit integer): First variable A
B (32-bit integer): Second variable B
C (32-bit integer): Third variable C
D (32-bit integer): Forth variable D
Outputs ***Max*** (32-bit integer): Maximum value of inputs
Properties ***Inputs*** (Integer, 2 to 4): Number of inputs
Remarks None

16.20. Logic - NOT

Description NOT : negation logic
Inputs ***Input*** (Logic): Input signal
Outputs ***!Input*** (Logic): Negated input signal
Properties None
Remarks None

16.21. Logic - AND

Description	AND : logical AND operation
Inputs	***A*** (Logic): A signal input
	B (Logic): B signal input
	C (Logic): C signal input
	D (Logic): D signal input
Outputs	***AND*** (Logic): Result of logical AND operation on input signals
	NAND (Logic): Negated result of logical AND operation
Properties	***Show negated output*** (Logic): Set to True to display the *NAND* output
	Inputs (Integer, 2 to 4): Number of logical inputs
Remarks	None

16.22. Logic - OR

Description	OR : logical OR operation
Inputs	***A*** (Logic): A signal input
	B (Logic): B signal input
	C (Logic): C signal input
	D (Logic): D signal input
Outputs	***OR*** (Logic): Result of logical OR operation on input signals
	NOR (Logic): Negated result of logical OR operation
Properties	***Show negated output*** (Logic): Set to True to display the *NOR* output
	Inputs (Integer, 2 to 4): Number of logical inputs
Remarks	None

16.23. Logic - XOR

Description	XOR : logical XOR operation
Inputs	***A*** (Logic): A signal input
	B (Logic): B signal input
Outputs	***XOR*** (Logic): Result of logical XOR operation on input signals
	EQ (Logic): Negated result of logical XOR operation
Properties	***Show negated output*** (Logic): Set to True to display the *EQ* output
Remarks	None

16.24. Logic - Compare (GT)

Description	Compare (GT) checks whether *Value* is **greater** than *Reference*
Inputs	***Value*** (32-bit integer): Input signal
	Reference (32-bit integer): Reference signal
Outputs	***Result*** (Logic): Result of comparison - 1 if condition is met, 0 otherwise

!Result (Logic): Negated result of the comparison
Properties ***Show negated output*** (Logic): Set to True to display the negated result output
Remarks None

16.25. Logic - Compare (GE)

Description Compare (GE) checks whether *Value* is **greater than or equal** to *Reference*
Inputs ***Value*** (32-bit integer): Input signal
Reference (32-bit integer): Reference signal
Outputs ***Result*** (Logic): Result of comparison - 1 if condition is met, 0 otherwise
!Result (Logic): Negated result of the comparison
Properties ***Show negated output*** (Logic): Set to True to display the negated result output
Remarks None

16.26. Logic - Compare (EQ)

Description Compare (EQ) checks whether *Value* is **equal** to *Reference*
Inputs ***Value*** (32-bit integer): Input signal
Reference (32-bit integer): Reference signal
Outputs ***Result*** (Logic): Result of comparison - 1 if condition is met, 0 otherwise
!Result (Logic): Negated result of the comparison
Properties ***Show negated output*** (Logic): Set to True to display the negated result output
Remarks None

16.27. Logic - Compare (LE)

Description Compare (LE) checks whether *Value* is **lower than or equal** to *Reference*
Inputs ***Value*** (32-bit integer): Input signal
Reference (32-bit integer): Reference signal
Outputs ***Result*** (Logic): Result of comparison - 1 if condition is met, 0 otherwise
!Result (Logic): Negated result of the comparison
Properties ***Show negated output*** (Logic): Set to True to display the negated result output
Remarks None

16.28. Logic - Compare (LT)

Description Compare (LT) checks whether *Value* is **lower** than *Reference*
Inputs ***Value*** (32-bit integer): Input signal
Reference (32-bit integer): Reference signal
Outputs ***Result*** (Logic): Result of comparison - 1 if condition is met, 0 otherwise
!Result (Logic): Negated result of the comparison
Properties ***Show negated output*** (Logic): Set to True to display the negated result output

Remarks None

16.29. Logic - Compare (NE)

Description Compare (NE) checks whether *Value* is **not equal** to *Reference*
Inputs ***Value*** (32-bit integer): Input signal
Reference (32-bit integer): Reference signal
Outputs ***Result*** (Logic): Result of comparison - 1 if condition is met, 0 otherwise
!Result (Logic): Negated result of the comparison
Properties ***Show negated output*** (Logic): Set to True to display the negated result output
Remarks None

16.30. Memory - Set/Reset latch

Description Set/Reset latch simulates an (asynchronous) SR (Set/Reset) latch
Inputs ***S*** (Logic): Set input signal
R (Logic): Reset input signal
Outputs ***Q*** (Logic): Latch state
Properties ***Default value*** (Logic): Default value on reset
Retain on reset (Logic): If True, the value is saved to battery-backed RAM. Also disables default value setting.
Initial value (Logic): The initial value of the retained block after it gets uploaded to the device.
Remarks None

16.31. Memory - JK latch

Description JK latch simulates an (asynchronous) JK latch
Inputs ***J*** (Logic): J input signal
K (Logic): K input signal
Outputs ***Q*** (Logic): Latch state
Properties ***Default value*** (Logic): Default value on reset
Retain on reset (Logic): If True, the value is saved to battery-backed RAM.
Initial value (Logic): The initial value of the retained block after it gets uploaded to the device.
Remarks None

16.32. Memory - D latch

Description D latch simulates an (asynchronous) D latch
Inputs ***D*** (Logic): D input signal
E (Logic): E(nable) input signal

Outputs ***Q*** (Logic): Latch state
Properties ***Default value*** (Logic): Default value on reset
Retain on reset (Logic): If True, the value is saved to battery-backed RAM.
Initial value (Logic): The initial value of the retained block after it gets uploaded to the device.
Remarks None

16.33. Memory - JK flip-flop

Description JK flip-flop simulates a JK flip-flop
Inputs ***J*** (Logic): J input signal
CLK (Logic): Clock input signal
K (Logic): K input signal
Outputs ***Q*** (Logic): Flip-flop state
Properties ***Default value*** (Logic): Default value on reset
Retain on reset (Logic): If True, the value is saved to battery-backed RAM.
Initial value (Logic): The initial value of the retained block after it gets uploaded to the device.
Remarks None

16.34. Memory - D flip-flop

Description D flip-flop simulates a D flip-flop
Inputs ***D*** (Logic): D input signal
CLK (Logic): Clock input signal
Outputs ***Q*** (Logic): Flip-flop state
Properties ***Default value*** (Logic): Default value on reset
Retain on reset (Logic): If True, the value is saved to battery-backed RAM.
Initial value (Logic): The initial value of the retained block after it gets uploaded to the device.
Remarks None

16.35. Memory - T flip-flop

Description T flip-flop simulates a T flip-flop
Inputs ***T*** (Logic): T input signal
CLK (Logic): Clock input signal
Outputs ***Q*** (Logic): Flip-flop state
Properties ***Default value*** (Logic): Default value on reset
Retain on reset (Logic): If True, the value is saved to battery-backed RAM.
Initial value (Logic): The initial value of the retained block after it gets uploaded to the device.
Remarks None

16.36. Memory - Data latch

Description: Data latch simulates a data latch. Similar to D flip-flop, but supports integers for input (D flip-flop uses logical signals only)

Inputs:
Data in (32-bit integer): Integer data input
CLK (Logic): Clock input signal

Outputs:
Value (32-bit integer): Stored integer data
Value F (32-bit integer): Stored integer data (on falling edge) in bi-directional mode

Properties:
Default value (Integer): Default reset value
Bi-directional (Logic): If True, the latch samples on both the rising edge and falling edge.
Retain on reset (Logic): If True, the value is saved to battery-backed RAM. Also disables default value setting.
Initial value (Integer): The initial value of the retained block after it gets uploaded to the device.

Remarks: None

16.37. Memory - 1 T delay

Description: 1 T delay simulates a one-cycle delay

Inputs:
In (32-bit integer): Integer data input

Outputs:
Q (32-bit integer): Integer data output

Properties:
Default value (Integer): Default reset value

Remarks: None

16.38. Memory - MIN memory

Description: MIN memory remembers the minimum value of the *Data in* signal. Use *CLK* input to reset the memory.

Inputs:
Data in (32-bit integer): Integer data input
CLK (Logic): Clock input signal

Outputs:
Value (32-bit integer): Stored minimum value

Properties:
Default value (Integer): Default reset value
Retain on reset (Logic): If True, the value is saved to battery-backed RAM. Also disabled default value setting.
Initial value (Integer): The initial value of the retained block after it gets uploaded to the device.

Remarks: None

16.39. Memory - MAX memory

Description MAX memory remembers the maximum value of the *Data in* signal. Use *CLK* input to reset the memory.

Inputs ***Data in*** (32-bit integer): Integer data input
CLK (Logic): Clock input signal

Outputs ***Value*** (32-bit integer): Stored maximum value

Properties ***Default value*** (Integer): Default reset value
Retain on reset (Logic): If True, the value is saved to battery-backed RAM. Also disabled default value setting.
Initial value (Integer): The initial value of the retained block after it gets uploaded to the device.

Remarks None

16.40. Memory - RAM (8-bit)

Description RAM (8-bit)

Inputs ***Data*** (8-bit integer): Integer data input
Address store (16-bit integer): Address of the destination memory
CLK store (Logic): Clock input signal for storing data
Address load (16-bit integer): Address of the destination memory
CLK load (Logic): Clock input signal for retrieving data

Outputs ***Data out*** (8-bit integer): Stored integer data (on falling edge) in bi-directional mode

Properties ***Memory size*** (Integer, 1 to 255): Number of memory cells
Retain on reset (Logic): If True, the value is saved to battery-backed RAM. Also disables default value setting.
Initial value (Integer, 0 to 255): The initial value of the retained block after it gets uploaded to the device.
Initial RAM value (Integer, 0 to 255): The initial value of the retained block after it gets uploaded to the device.

Remarks None

16.41. Memory - RAM (16-bit)

Description RAM (16-bit)

Inputs ***Data*** (16-bit integer): Integer data input
Address store (16-bit integer): Address of the destination memory
CLK store (Logic): Clock input signal for storing data
Address load (16-bit integer): Address of the destination memory
CLK load (Logic): Clock input signal for retrieving data

Outputs ***Data out*** (16-bit integer): Stored integer data (on falling edge) in bi-directional mode

Properties ***Memory size*** (Integer, 1 to 255): Number of memory cells
Retain on reset (Logic): If True, the value is saved to battery-backed RAM. Also disables default value setting.

Initial value (Integer, 0 to 65535): The initial value of the retained block after it gets uploaded to the device.
Initial RAM value (Integer, 0 to 65535): The initial value of the retained block after it gets uploaded to the device.

Remarks None

16.42. Memory - RAM (32-bit)

Description RAM (32-bit)

Inputs ***Data*** (32-bit integer): Integer data input
Address store (16-bit integer): Address of the destination memory
CLK store (Logic): Clock input signal for storing data
Address load (16-bit integer): Address of the destination memory
CLK load (Logic): Clock input signal for retrieving data

Outputs ***Data*** (32-bit integer): Stored integer data (on falling edge) in bi-directional mode

Properties ***Memory size*** (Integer, 1 to 255): Number of memory cells
Retain on reset (Logic): If True, the value is saved to battery-backed RAM. Also disables default value setting.
Initial value (Integer): The initial value of the retained block after it gets uploaded to the device.
Initial RAM value (Integer): The initial value of the retained block after it gets uploaded to the device.

Remarks None

16.43. Memory - Sample/hold

Description Sample/hold simulates a sample and hold element. When the sample input is enabled, the block operates in transparent mode. When sample input is disabled, the output retains the last value.

Inputs ***Data in*** (32-bit integer): Integer data input
Sample (Logic): Sample input signal

Outputs ***Value*** (32-bit integer): Stored integer data

Properties ***Default value*** (Integer): Default reset value
Retain on reset (Logic): If True, the value is saved to battery-backed RAM. Also disables default value setting.
Initial value (Integer): The initial value of the retained block after it gets uploaded to the device.

Remarks None

16.44. Trigger / timing - Clock source

Description Clock source
Inputs ***Half time-period*** (32-bit integer): Dynamic *Half time-period* optional input
Outputs ***Clock*** (Logic):
Properties ***Half time-period*** (Integer, 1 to 3600000): Half of the clock time period (in milliseconds - set to 500 for 1 Hz clock)
Show dynamic input (Logic): If set to True, Half-time period input is enabled
Remarks None

16.45. Trigger / timing - Rising edge

Description Rising edge triggers the output when the *Input* signal value changes from 0 to 1 (rises)
Inputs ***Input*** (Logic): Input signal
Outputs ***Change*** (Logic): Output indicating a change in the input signal
Properties None
Remarks None

16.46. Trigger / timing - Falling edge

Description Falling edge triggers the output when the *Input* signal value changes from 1 to 0 (falls)
Inputs ***Input*** (Logic): Input signal
Outputs ***Change*** (Logic): Output indicating a change in the input signal
Properties None
Remarks None

16.47. Trigger / timing - Up counter

Description Up counter counts the *Clock* input positive (rising) changes from 0 to value, specified by *PV* input
Inputs ***Clock*** (Logic): Counter pulse signal
Reset (Logic): When 1, resets the counter value *Value* to 0
PV (32-bit integer): Maximum counter value
Outputs ***Q*** (Logic): Indicates whether counter has reached maximum value
Value (32-bit integer): Current counter value
Properties ***Default value*** (Integer): Default reset value of the counter
Retain on reset (Logic): If True, the value is saved to battery-backed RAM. Also disables default value setting.
Initial value (Integer): The initial value of the retained block after it gets uploaded to the device.

Remarks None

16.48. Trigger / timing - Down counter

Description Down counter decrements the counter on the *Clock* input positive (rising) changes from value, specified by *PV* input, to 0

Inputs ***Clock*** (Logic): Counter pulse signal
Load PV (Logic): When 1, counter value *Value* is loaded with value of *PV*
PV (32-bit integer): Initial counter value

Outputs ***Q*** (Logic): Indicates whether counter has reached 0
Value (32-bit integer): Current counter value

Properties ***Default value*** (Integer): Default reset value of the counter
Retain on reset (Logic): If True, the value is saved to battery-backed RAM. Also disables default value setting.
Initial value (Integer): The initial value of the retained block after it gets uploaded to the device.

Remarks None

16.49. Trigger / timing - Up/down counter

Description Up/down counter increments the counter vaue on the *Clock up* input positive (rising) changes and decrements the counter on the *Clock down* input positive (rising) changes. The counter value *Value* is limited to the range between (including) 0 to *PV*

Inputs ***Clock up*** (Logic): Counter increasing pulse signal
Clock down (Logic): Counter decreasing pulse signal
Reset (Logic): When 1, resets the counter value *Value* to 0
Load PV (Logic): When 1, counter value *Value* is loaded with value of *PV*
PV (32-bit integer): Initial/maximum counter value

Outputs ***QU*** (Logic): Indicates whether counter has reached maximum value
QD (Logic): Indicates whether counter has reached 0
Value (32-bit integer): Current counter value

Properties ***Default value*** (Integer): Default reset value of the counter
Retain on reset (Logic): If True, the value is saved to battery-backed RAM. Also disables default value setting.
Initial value (Integer): The initial value of the retained block after it gets uploaded to the device.

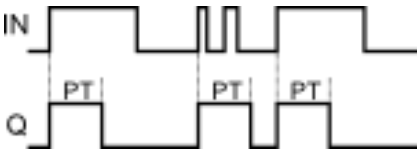
Remarks None

16.50. Trigger / timing - Pulse timer

Description Pulse timer triggers on *IN* high input signal and turns off after the period defined by *PT*, uneffected by the *IN* signal state during this period as illustrated below

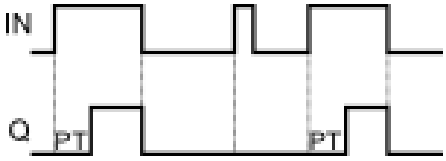
Inputs ***IN*** (Logic): Timer activation signal input

PT (32-bit integer): Timer period signal input (in ms)
Outputs ***Q*** (Logic): Timer activation status signal
ET (32-bit integer): Timer current time signal (in ms)
Properties None
Remarks



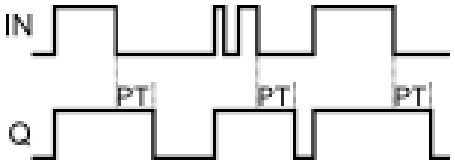
16.51. Trigger / timing - On timer

Description On timer or (on delay timer) starts counting time on *IN* input signal rising edge and turns on after the period defined by *PT*. The timer is reset if the input signal *IN* goes to low state as illustrated below
Inputs ***IN*** (Logic): Timer activation signal input
PT (32-bit integer): Timer period signal input (in ms)
Outputs ***Q*** (Logic): Timer activation status signal
ET (32-bit integer): Timer current time signal (in ms)
Properties None
Remarks



16.52. Trigger / timing - Off timer

Description Off timer or (off delay timer) activates on *IN* high input signal state. The timer starts counting on the *IN* input signal falling edge and turns off after the period defined by *PT*. The counter is reset if the input signal *IN* goes to high state and the timer stays activated as illustrated below
Inputs ***IN*** (Logic): Timer activation signal input
PT (32-bit integer): Timer period signal input (in ms)
Outputs ***Q*** (Logic): Timer activation status signal
ET (32-bit integer): Timer current time signal (in ms)
Properties None
Remarks

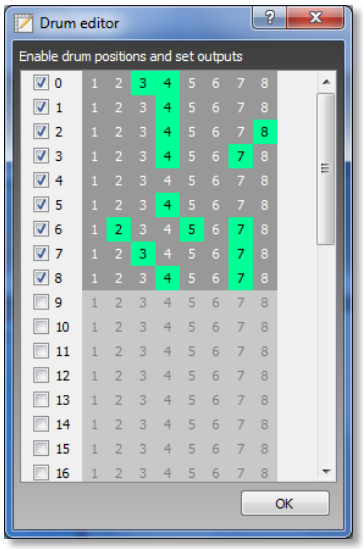


16.53. Trigger / timing - Time

Description	Time is used to read current date and time
Inputs	None
Outputs	***Second*** (8-bit integer): Seconds ***Minute*** (8-bit integer): Minute ***Hour*** (8-bit integer): Hour ***Day of month*** (8-bit integer): Day of month ***Month*** (8-bit integer): Month ***Year*** (16-bit integer): Year ***Day of week*** (8-bit integer): Day of week
Properties	None
Remarks	None

16.54. Extended - Event drum

Description	Event drum resembles a mechanical contact drum and enables easy-to-use programming of various output sequences. Double-click on block to edit configuration.
Inputs	***Position*** (32-bit integer): Drum position input (in the range from 0 to number of entries - 1)
Outputs	***Out 1-8*** (Logic): Drum output 1-8 signal
Properties	None
Remarks	Event drum editor (see figure below) is used to define outputs at each drum slot. Left-click to toggle output (Green = activated) and use check boxes on the left to set the number of slots on the drum.



16.55. Extended - Look-up table (byte)

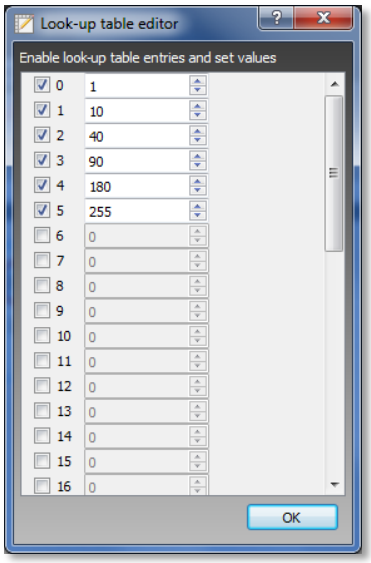
Description Look-up table (byte) is used to select data from the table. Double-click on block to edit configuration.

Inputs ***Position*** (32-bit integer): Look-up table entry index

Outputs ***Out*** (8-bit integer): Entry value

Properties None

Remarks Use look-up table editor to set values of the entries. Use check boxes on the left to set number of entries in the table.



16.56. Extended - Schedule

Description Schedule is used to setup activation schedule by configuring multiple intervals. To setup the intervals, double click on the block in the scheme.

Inputs None

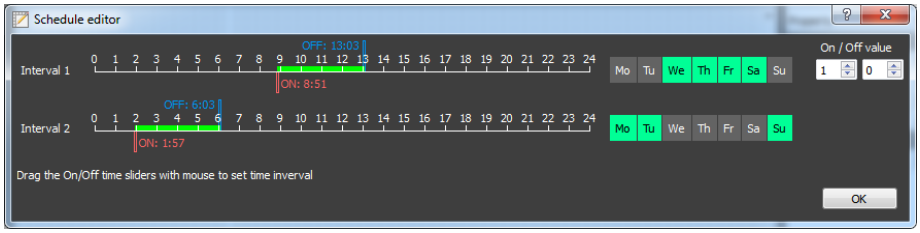
Outputs ***Out*** (8-bit integer): Activation signal
Out 1 - 10 (8-bit integer): Interval-specific optional ouputs.
Rule 1 - 10 (32-bit integer): interval-specific optional rule access, that can be used to access and modify rule data using shared slots.

Properties ***Separate outputs*** (Logic): When enabled, each interval rule drives its own output, Out is still the OR-ed value of all.
Expose rules values (Logic): When enabled, each interval rule is connected to the output, allowing remote changes to the intervals.
Rules (Integer, 1 to 10): Number of scheduler rules
Retain on reset (Logic): If True, the value is saved to battery-backed RAM. Also disables default value setting.

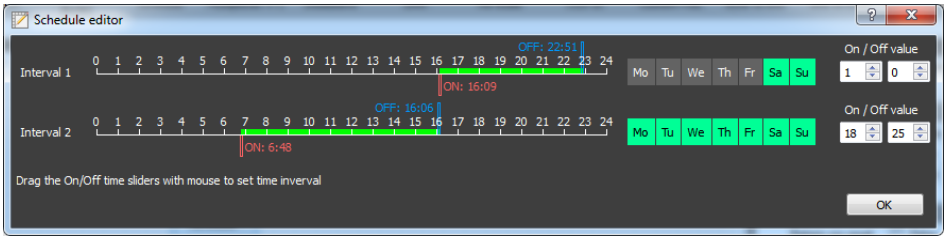
Remarks Use schedule editor to configure the time-based output activation rules. Up to 10 rules can be defined for a single Schedule block a slider-based interface. Double-click

the Schedule block to access the editor. Each rule defines one time interval during which the output is activated. Use left mouse button to drag the ON and OFF sliders to appropriate positions and select weekdays that the rule is valid on. If OFF slider is positioned to the left of the ON slider, ON slider time refers to the time in the next day.
By default, 2 rules are available for setup (to increase the number of rules, change the 'Rules' property of the block).

With separate outputs option set to 'False', the block outputs single value, based on the On/Off values, defined in the Schedule block editor (in upper-right corner).



If separate outputs option is enabled (set to 'True'), the Schedule block outputs separate values for each defined rule with their respective On and Off values, set in the fields on the right side of the dialog. The 'Out' output outputs a logical 1/0 value, indicating whether any of the rules is active or not.



In order to allow remote schedule modification, 'Expose rules values' option is available. Additional rules outputs appear on the block, which can be connected to Shared data slots. The rule is bit-wise encoded in a 32-bit value as described in the list below:

- Bits 0-5: onMinute
- Bits 6-10: onHour
- Bits 11-16: offMinute
- Bits 17-21: offHour
- Bits 22-28: bit-encoded week days
- Bits 29-31: unused

16.57. Extended - Multiplexer n-1

Description Multiplexer n-1 routes an input, specified by the *SEL* input, to output. Unconnected input will be resolved as 0.

Inputs	***SEL*** (8-bit integer): SEL
	In 1-10 (32-bit integer): Input 1-10
Outputs	***Out*** (32-bit integer): Value of the selected input
Properties	***Inputs*** (Integer, 2 to 10): Number of inputs
Remarks	None

16.58. Extended - Deadband

Description	Deadband adds symetrical deadband effect to the input signal.
Inputs	***Input*** (32-bit integer): Input signal
	Deadband (32-bit integer): (Half-)deadband width
Outputs	***Output*** (32-bit integer): Signal with deadband
Properties	None
Remarks	None

16.59. Extended - Limit

Description	Limit adds limit effect to the input signal.
Inputs	***Input*** (32-bit integer): Input signal
	Out min (32-bit integer): Minimum output signal value
	Out max (32-bit integer): Maximum output signal value
Outputs	***Output*** (32-bit integer): Limited signal
Properties	None
Remarks	None

16.60. Extended - Re-scale

Description	Re-scale rescales the input signal using the specified ranges
Inputs	***Input*** (32-bit integer): Input signal
Outputs	***Output*** (32-bit integer): Rescaled signal
Properties	***Input min*** (Integer): Minimum value of the input signal
	Input max (Integer): Maximum value of the input signal
	Output min (Integer): Output signal at the minimum input signal
	Output max (Integer): Output signal at the maximum input signal
	Limit output (Logic): Limit output to the specified range
Remarks	None

16.61. Extended - LCD UI

Description	LCD UI is used to configure user interface on alphanumeric LCD. LCD UI supports multiple LCD layouts (LCD contents), which can be selected/switched using the 'Layout' input. Each layout can contain different static (text) and dynamic (block

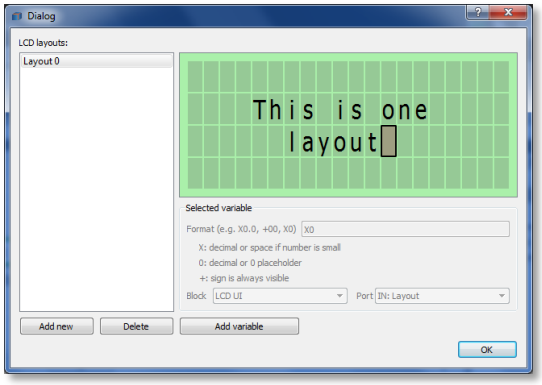
output values) content.
LCD UI block must be stimulated on the refresh input in order to refresh the contents of the LCD. LCD refresh is a time-consuming operation and is not done by PoBlocks or PoIL code by itself – user has to define a proper refresh time using clock signal block or generate a signal to refresh the LCD otherwise.
Double-click on block to edit configuration.

Inputs ***Layout*** (Logic): Layout selection input
Refresh (Logic): Refresh input. LCD interface will be refreshed on low-to-high transition

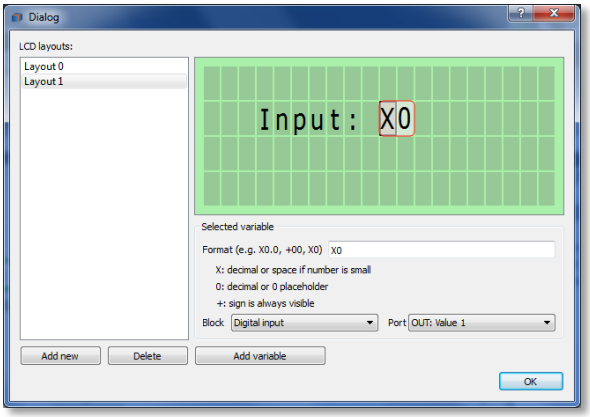
Outputs None

Properties ***Use secondary pins*** (Logic): If True, LCD will be initialized on secondary pins as described in PoKeys manual
LCD rows (Integer, 1 to 4): Number of rows in the LCD display
LCD columns (Integer, 1 to 20): Number of columns in the LCD display

Remarks LCD UI editor is split into layouts list on the left and a current layout editor on the right. Start by adding a new layout using 'Add new' button on the bottom. Select the new layout and place static or dynamic contents to LCD.
To place static contents (text), click on the position on the LCD simulator and enter the text. To navigate with keyboard, use arrow keys.



Dynamic content is placed using the 'Add variable' button on the bottom. Move cursor to the position where you want to position the dynamic contents and press 'Add variable'. A grey rounded rectagle will appear – click on it to edit the display properties, use the 'Format' field to enter the number format and 'Block', 'Port' drop-down boxes to select the data source.



16.62. Extended - Custom PoIL

Description	Custom PoIL for custom PoIL code - double-click on block to edit code
Inputs	Up to 10 inputs of various types
Outputs	Up to 10 outputs of various types
Properties	None
Remarks	None

16.63. Extended - Byte to bits

Description	Byte to bits extracts individual bits from the byte variable on input, bit 0 being least-significant one
Inputs	***In*** (8-bit integer): Input variable
Outputs	***Bit 0-7*** (Logic): Bit 0 (LSb) - Bit 7 (MSb)
Properties	None
Remarks	None

16.64. Extended - Bits to byte

Description	Bits to byte joins individual bits to the byte variable on output, bit 0 being least-significant one
Inputs	***Bit 0*** (Logic): Bit 0 (LSb) - Bit 7 (MSb)
Outputs	***Out*** (8-bit integer): Output variable
Properties	None
Remarks	None

16.65. Control - On/off

Description	On/off
Inputs	***PV*** (32-bit integer): Process Variable input (current value of the controlled variable) ***SP*** (32-bit integer): Set Point
Outputs	***Out*** (32-bit integer): On/Off controller output
Properties	***Gain*** (Integer): Controller gain ***Hysteresis*** (Integer): Half of the controller hysteresis - the difference between turn-on and turn-off points . The controller turns on at PV > SP+hysteresis/2 and turns off at PV < SP-hysteresis/2
Remarks	None

16.66. Control - PID

Description	PID (PID controller) is an implementation of a standard PID controller with filtered D-part and output limits with integral anti-windup protection.
Inputs	***PV*** (32-bit integer): Process Variable input (current value of the controlled variable) ***SP*** (32-bit integer): Set Point
Outputs	***Out*** (32-bit integer): PID controller output (in the range from *Out min* to *Out max*
Properties	***K*** (Floating point): Proportional gain ***Ti*** (Floating point): Integral time constant in seconds ***Td*** (Floating point): Derivative time constant in seconds ***Tf*** (Floating point): Derivative filter time constant - set to a fraction of *Td* ***Out min*** (Integer): Output lower limitation (minimum output value) ***Out max*** (Integer): Output upper limitation (maximum output value)
Remarks	When cycle time of the project is set to 0, PID block can not be used in the diagram due to improper timing.

16.67. Communication - 1-wire R/W

Description	1-wire write and read operation block
Inputs	***!CLK*** (Logic/clock): Inverted clock input signal ***Write 1 - 10*** (8-bit integer): Bytes 1 - 10 to write to 1-wire device
Outputs	***Busy*** (Logic): Busy signal ***Read 1 - 10*** (8-bit integer): Bytes 1 - 10 that are read from 1-wire device
Properties	***Write count*** (Integer, 0 to 10): Number of bytes to write to 1-wire bus ***Read count*** (Integer, 0 to 10): Number of bytes to read from 1-wire bus
Remarks	Communication is started on **falling** edge of the !CLK input in order to support chaining multiple communication blocks using busy signals - when one device completes the communication session, its Busy signal goes from 1 to 0, which then triggers a communication session on another block in the chain.

16.68. Communication - I2C read

Description	I^2C read operation block
Inputs	***!CLK*** (Logic/clock): Inverted clock input signal ***Address*** (8-bit integer): I^2C device address
Outputs	***Busy*** (Logic): Busy signal ***Error*** (Logic): Error signal ***Read 1 - 10*** (8-bit integer): Bytes 1 - 10 to be read from I2C device
Properties	***Read count*** (Integer, 0 to 10): Number of bytes to read from I^2C device
Remarks	Communication is started on **falling** edge of the !CLK input in order to support chaining multiple communication blocks using busy signals - when one device completes the communication session, its Busy signal goes from 1 to 0, which then triggers a communication session on another block in the chain.

16.69. Communication - I2C write

Description	I^2C write operation block
Inputs	***!CLK*** (Logic/clock): Inverted clock input signal ***Address*** (8-bit integer): I^2C device address ***Write 1 - 10*** (8-bit integer): Bytes 1 - 10 to write to I2C device
Outputs	***Busy*** (Logic): Busy signal ***Error*** (Logic): Error signal
Properties	***Write count*** (Integer, 0 to 10): Number of bytes to write to I^2C device
Remarks	Communication is started on **falling** edge of the !CLK input in order to support chaining multiple communication blocks using busy signals - when one device completes the communication session, its Busy signal goes from 1 to 0, which then triggers a communication session on another block in the chain.

16.70. Misc - Comment

Description	Comment is used to enter comments
Inputs	None
Outputs	None
Properties	***Comment***: Comment text
Remarks	None

16.71. Misc - To

Description	To is used to route signals across the schematic. To connect *To* and *From* blocks, identical *Link ID* must be specified.
Inputs	***>>>***: Input port of the link
Outputs	None
Properties	***Link ID*** (Integer): Link ID - see block description
Remarks	Only one 'To' block can be specified for one *Link ID*, while multiple 'From' blocks can

be specified for the same *Link ID*

16.72. Misc - From

Description	From is used to route signals across the schematic
Inputs	None
Outputs	***>>>***: Output port of the link
Properties	***Link ID*** (Integer): Link ID - see block description
Remarks	Only one 'To' block can be specified for one *Link ID*, while multiple 'From' blocks can be specified for the same *Link ID*

16.73. Misc - Shared 1-bit

Description	Shared 1-bit simulates a simple memory latch for use with shared slots.
Inputs	None
Outputs	(Logic): Stored integer data
Properties	***Default value*** (Logic): Default reset value ***Retain on reset*** (Logic): If True, the value is saved to battery-backed RAM. Also disables default value setting. ***Initial value*** (Logic): The initial value of the retained block after it gets uploaded to the device.
Remarks	None

16.74. Misc - Shared 8-bit

Description	Shared 8-bit simulates a simple memory latch for use with shared slots.
Inputs	None
Outputs	(8-bit integer): Stored integer data
Properties	***Default value*** (Integer): Default reset value ***Retain on reset*** (Logic): If True, the value is saved to battery-backed RAM. Also disables default value setting. ***Initial value*** (Integer): The initial value of the retained block after it gets uploaded to the device.
Remarks	None

16.75. Misc - Shared 16-bit

Description	Shared 16-bit simulates a simple memory latch for use with shared slots.
Inputs	None
Outputs	(16-bit integer): Stored integer data
Properties	***Default value*** (Integer): Default reset value ***Retain on reset*** (Logic): If True, the value is saved to battery-backed RAM. Also

disables default value setting.
Initial value (Integer): The initial value of the retained block after it gets uploaded to the device.

Remarks None

16.76. Misc - Shared 32-bit

Description Shared 32-bit simulates a simple memory latch for use with shared slots.

Inputs None

Outputs (32-bit integer): Stored integer data

Properties ***Default value*** (Integer): Default reset value
Retain on reset (Logic): If True, the value is saved to battery-backed RAM. Also disables default value setting.
Initial value (Integer): The initial value of the retained block after it gets uploaded to the device.

Remarks None

17.Grant of license

The material contained in this release is licensed, not sold. PoLabs grants a license to the person who installs this software, subject to the conditions listed below.

Access

The licensee agrees to allow access to this software only to persons who have been informed of and agree to abide by these conditions.

Usage

The software in this release is for use only with PoLabs products or with data collected using PoLabs products.

Copyright

PoLabs claims the copyright of, and retains the rights to, all material (software, documents, etc.) contained in this release.
You may copy and distribute the entire release in its original state, but must not copy individual items within the release other than for backup purposes.

Liability

PoLabs and its agents shall not be liable for any loss or damage, howsoever caused, related to the use of PoLabs equipment or software, unless excluded by statute.

Fitness for purpose

No two applications are the same, so PoLabs cannot guarantee that its equipment or software is suitable for a given application. It is therefore the user's responsibility to ensure that the product is suitable for the user's application.

Mission Critical applications

Because the software runs on a computer that may be running other software products, and may be subject to interference from these other products, this license specifically excludes usage in 'mission critical' applications, for example life support systems.

Viruses

This software was continuously monitored for viruses during production, however the user is responsible for virus checking the software once it is installed.

Support

No software is ever error-free, but if you are unsatisfied with the performance of this software, please contact our technical support staff, who will try to fix the problem within a reasonable time.

Upgrades

We provide upgrades, free of charge, from our web site at www.poscope.com. We reserve the right to charge for updates or replacements sent out on physical media.

Trademarks

Windows is a registered trademark of Microsoft Corporation. PoKeys, PoKeys55, PoKeys56U, PoKeys56E, PoScope, PoLabs and others are internationally registered trademarks.